Water Cycle Algorithm with Fuzzy Logic for Dynamic Adaptation of Parameters

Eduardo Méndez[1], Oscar Castillo[1], José Soria[1], Patricia Melin[1(✉)], and Ali Sadollah[2]

[1] Tijuana Institute of Technology, Calzada Tecnologico s/n, Tijuana, Mexico
{ocastillo, pmelin}@tectijuana.mx
[2] Nanyang Technological University, Singapore, Singapore

Abstract. This paper describes the enhancement of the Water Cycle Algorithm (WCA) using a fuzzy inference system to adapt its parameters dynamically. The original WCA is compared regarding performance with the proposed method called Water Cycle Algorithm with Dynamic Parameter Adaptation (WCA-DPA). Simulation results on a set of well-known test functions show that the WCA can be improved with a fuzzy dynamic adaptation of the parameters.

Keywords: WCA · Fuzzy logic · Optimization

1 Introduction

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

Using a fuzzy inference system 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 fuzzy inference system (FIS) similar to the one in [9] was developed.

A comparative study was conducted which highlights the similarities and differences with other hierarchy based meta-heuristics. And a performance study between the proposed Water Cycle Algorithm with Fuzzy Parameter Adaptation (WCA-FPA), the original WCA, and the Gravitational Search Algorithm with Fuzzy Parameter Adaptation (GSA-FPA) [16] 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 Water Cycle Algorithm (WCA) is described. In Sect. 3, some similarities with other meta-heuristics are highlighted. Section 4 it's about how to improve the WCA with fuzzy parameter adaptation. A comparative study is presented in Sect. 5 and, finally in Sect. 6 some conclusions and future work are given.

© Springer International Publishing AG 2017
G. Sidorov and O. Herrera-Alcántara (Eds.): MICAI 2016, Part I, LNAI 10061, pp. 250–260, 2017.
DOI: 10.1007/978-3-319-62434-1_21

2 The Water Cycle Algorithm

The water cycle algorithm is a population-based and nature-inspired meta-heuristic, where a population of streams is formed from rain water drops. This population of streams follows a behavior inspired by the hydrological cycle. In which streams and rivers flow downhill towards the sea. This process of flowing downhill is a simplified form of the runoff process of the hydrologic cycle. After the runoff, some of the streams are evaporated, and some new streams are created from rain as part of the hydrologic cycle.

2.1 The Landscape

There are some landforms involved in the hydrologic cycle, but in the WCA only three of them are considered, which are streams, rivers, and the sea. In this subsection, the structure, preprocessing and initialization of the algorithm are described.

In the WCA an individual (a.k.a. stream), it's an object which consists of n variables grouped as a n-dimensional column vector:

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

The number of streams and rivers are defined by the equations:

$$N_{sr} = \text{Number of Rivers} + \underbrace{1}_{sea}, \tag{2}$$

$$N_{streams} = N - N_{sr}, \tag{3}$$

Where N is the size of the population of streams, N_{sr} is a value established as a parameter of the algorithm and N streams is the number of remaining streams.

The whole population of N streams is represented as a $N \times n$ matrix:

$$\mathbf{X} = \begin{matrix} sea \\ rivers \\ \\ \\ \\ streams \\ \\ \\ \end{matrix} \left\{ \begin{matrix} \left[\begin{matrix} \mathbf{x}_1^{\mathrm{T}} \end{matrix} \right] \\ \left\{ \begin{matrix} \mathbf{x}_2^{\mathrm{T}} \\ \mathbf{x}_3^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_{N_{sr}}^{\mathrm{T}} \end{matrix} \right. \\ \left\{ \begin{matrix} \mathbf{x}_{N_{sr}+1}^{\mathrm{T}} \\ \mathbf{x}_{N_{sr}+2}^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_{N_{sr}+N_{streams}}^{\mathrm{T}} \end{matrix} \right. \end{matrix} \right. = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \ldots & x_{Nn} \end{bmatrix}. \tag{4}$$

In matrix (4) the individuals are ordered by their fitness, and the fitness of each stream is obtained by:

$$\mathbf{f}_i = \mathrm{f}(\mathbf{x}_i) = \mathrm{f}(x_{i1}, x_{i2}, \ldots, x_{in}), \quad \text{for} \quad i = 1, 2, 3, \ldots, N, \tag{5}$$

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

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

$$\mathbf{so}_i = \left\lfloor \left| \frac{\mathbf{f}_i}{\sum_{j=1}^{N_{sr}} \mathbf{f}_j + \varepsilon} \right| \cdot N_{streams} \right\rfloor, \quad n = 1, 2, \ldots, N_{sr}, \tag{6}$$

$$\mathbf{so}_1 \leftarrow \mathbf{so}_1 + \left(N_{streams} - \sum_{i=1}^{N_{sr}} \mathbf{so}_i \right), \tag{7}$$

where $\varepsilon \approx 0$. The idea behind the Eq. (6) 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 mayor; therefore, the stream-flow magnitude of rivers is inversely proportional to its fitness in the case of minimization problems.

The Eq. (6) has been changed from the original proposed in [2], a floor function replaced the round function, a value of ε was added to the divisor, and the Eq. (7) was also add to handle the remaining streams. These changes are for 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 among them.

2.2 The Run-Off Process

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

$$\mathbf{x}_{stream}^{i+1} = \mathbf{x}_{stream}^{i} + \mathbf{r} \cdot C \cdot \left(\mathbf{x}_{sea}^{i} - \mathbf{x}_{stream}^{i} \right), \tag{8}$$

$$\mathbf{x}_{stream}^{i+1} = \mathbf{x}_{stream}^{i} + \mathbf{r} \cdot C \cdot \left(\mathbf{x}_{river}^{i} - \mathbf{x}_{stream}^{i} \right), \tag{9}$$

$$\mathbf{x}_{river}^{i+1} = \mathbf{x}_{river}^{i} + \mathbf{r} \cdot C \cdot \left(\mathbf{x}_{sea}^{i} - \mathbf{x}_{river}^{i} \right), \tag{10}$$

for $i = 1, 2, \ldots, N_{it}$, where N_{it} and Care parameters of the algorithm, and $\mathbf{r}$ is an n-dimensional vector of independent and identically distributed (i.i.d) values, that follow a uniform distribution:

$$\mathbf{r} \sim \mathrm{U}(0, 1)^n. \tag{11}$$

The Eq. (8) defines the movement of streams which flow directly to the sea, Eq. (9) is for streams which flow towards the rivers, and Eq. (10) is for the rivers flow toward the sea. A value of $C > 1$ enables streams to flow in different directions towards the

rivers or sea. Typically, the value of C is chosen from the range $[1, 2]$ being two the most common.

2.3 Evaporation and Precipitation Process

The run-off process of the WCA consists of moving indirectly towards the global best (sea). Algorithms focused on following the global best although they are 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 re-initialization is similar to the cooling down and heating up re-initialization 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:

$$\underbrace{|\mathbf{x}_{sea} - \mathbf{x}_{river}| < d_{\max}}_{\text{evaporation criterion}}, \tag{12}$$

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

$$\underbrace{\mathbf{x}_{stream} = \mathbf{b}_{lower} + \mathbf{r} \cdot \left(\mathbf{b}_{lower} - \mathbf{b}_{upper}\right)}_{\text{raining around the search space}}, \tag{13}$$

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

3 Similarities and Differences with Other Meta-Heuristics

The WCA has some similarities with other meta-heuristics 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]. And in [12], WCA is compared with two nature-inspired meta-heuristics: the Intelligent Water Drops (IWD) [5] and Water Wave Optimization (WWO) [18]. So far similarities and differences with population-based and nature-inspired meta-heuristics have been studied, in this subsection, WCA is compared with two meta-heuristics 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 on the stream order. Therefore, WCA hierarchy it's not a nearly regular tree.

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 towards 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 run-off 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 gray wolves in nature. Four types of gray 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 positions update depends on 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) to (14) for updating the positions of the streams. The other one is the parameter d_{max} used in Eq. (12) 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 those 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 parameter C.

A single-input and multiple-output (SIMO) Mamdani's Fuzzy Inference System [8] was developed. The system consists of the input Iteration and the outputs Cstream and Criver. In Fig. 1, the layout of the fuzzy inference system 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 (Criver) for Eq. (15) which is for the flow of rivers.

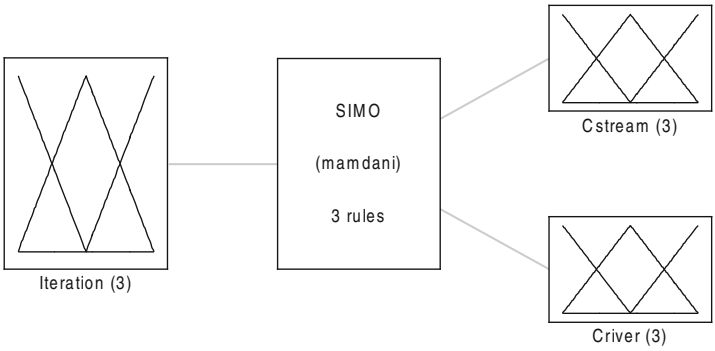

Fig. 1. Single-input and multiple-output Mamdani's Fuzzy Inference System

The input Iteration is calculated by the equation:

$$Iteration = \frac{i}{N_{it}}, \quad \text{for} \quad i = 1, 2, \ldots, N_{it}. \tag{14}$$

The range of the first output Cstream had been chosen as the interval $[1, 2.2]$, and for the output Criver the interval $[1.6, 4]$, the details of the membership functions are shown in Fig. 2. The idea of using higher values for these parameters is to favor the

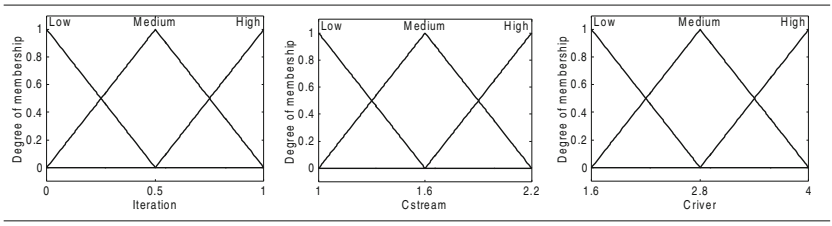

Fig. 2. Membership functions

exploration in the run-off process at an early stage, since having a greater value than two means that there is a higher probability of moving beyond the rivers or the sea. For the special case of Criver, it also helps to prevent the evaporation and precipitation processes.

The fuzzy rules are simple, the algorithm begins with higher values of C favoring exploration and slowly decreases to favor exploitation.

5 Experimental Setting

To test the performance of WCA-FPA and for comparison, we used a diverse subset of 10 unconstrained test functions, shown in Table 1 and Fig. 3. All the functions are well-known minimization problems previously used as benchmarks in the literature.

Table 1. Rules of the fuzzy inference system

1.	If (Iteration is Low) then (Cstream is High) (Criver is High) (1)
2.	If (Iteration is Medium) then (Cstream is Medium) (Criver is Medium) (1)
3.	If (Iteration is High) then (Cstream is Low) (Criver is Low) (1)

Experiments were carried out in 10 and 30 dimensions (variables), using a sample size of 50 for each function. As a measure of performance, the mean and standard deviation fitness of the best-obtained solution were considered and used for hypothesis testing.

Table 2. Set of 10 unconstrained test functions

No.	Function		Range
f_1	Egg Holder	$f(\mathrm{x}) = \sum_{i=1}^{n-1}\left[-(x_{i+1}+47)\sin\sqrt{\left\lvert x_{i+1}+\frac{x_i}{2}+47\right\rvert} - x_i\sin\sqrt{\lvert x_i-(x_{i+1}+47)\rvert}\right]$	$-512 \le x_i \le 512$
f_2	Rana	$f(\mathrm{x}) = \sum_{i=1}^{n-1} x_i\sin\left(\left[\sqrt{\lvert x_{i+1}-x_i+1\rvert}\right)\cos\left(\sqrt{\lvert x_{i+1}+x_i+1\rvert}\right)\right.$ $\left. +(x_{i+1}+1)\sin\left(\sqrt{\lvert x_{i+1}+x_i+1\rvert}\right)\cos\left(\sqrt{\lvert x_{i+1}-x_i+1\rvert}\right)\right]$	$-500 \le x_i \le 500$
f_3	Michalewicz	$f(\mathrm{x}) = -\sum_{i=1}^{n}\sin(x_i)\cdot\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}, \quad m=0$	$0 \le x_i \le \pi$
f_4	Rosenbrock	$f(\mathrm{x}) = \sum_{i=1}^{n-1}\left[(x_i-1)^2+100(x_{i+1}-x_i^2)^2\right]$	$-5 \le x_i \le 5$
f_5	De Jong	$f(\mathrm{x}) = \sum_{i=1}^{n} x_i^2$	$-5.12 \le x_i \le 5.12$
f_6	Schewefel	$f(\mathrm{x}) = -\sum_{i=1}^{n} x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$-500 \le x_i \le 500$
f_7	Ackley	$f(\mathrm{x}) = -20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+\exp(1)$	$-32.768 \le x_i \le 32.768$
f_8	Rastrigin	$f(\mathrm{x}) = -10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	$-5.12 \le x_i \le 5.12$
f_9	Griewank	$f(\mathrm{x}) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$-600 \le x_i \le 600$
f_{10}	Yang	$f(\mathrm{x}) = \left(\sum_{i=1}^{n}x_i\right)\exp\left[-\sum_{i=1}^{n}\sin(x_i^2)\right]$	$-2\pi \le x_i \le 2\pi$

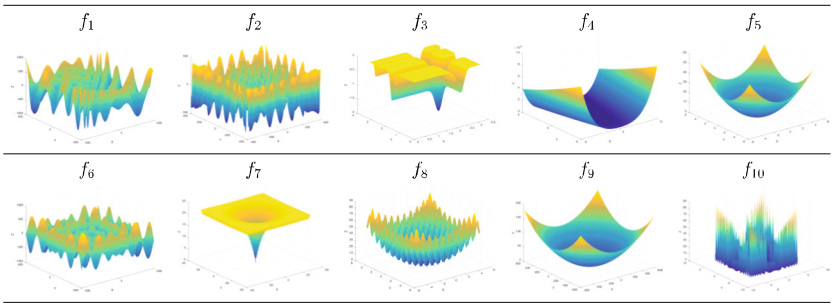

Fig. 3. Test functions surface plots for 2-dimensions

The comparisons are against the original WCA and the Gravitational Search Algorithm with Fuzzy Parameter Adaptation (GSA-FPA).

For a fair comparison, a black-box optimization approach is adopted, using the same fixed parameter settings for all the problems, although the algorithms are allowed to adapt some of its parameters at running time (Table 2).

The fixed parameters are: population size $N = 25$ and number of iterations $N_{it} = n \cdot 10^4/N$; for the water cycle algorithms $N_{sr} = 4$ and $d_{max} = 0.01$; and for the GSA-FPA an initial $G_0 = 100$.

All experiments were performed using MATLAB with 64 bits' double precision.

Tables 3 and 4 summarize the statistics results for 10-dimensional and 30-dimensional problems respectively.

Table 3. Summary statistics for 10-dimensional problems

No.	Samples means			Samples deviations		
	μ_{WCA}	$\mu_{WCA-FPA}$	$\mu_{GSA-FPA}$	σ_{WCA}	$\sigma_{WCA-FPA}$	$\sigma_{GSA-FPA}$
f_1	-3.54×10^3	-4.02×10^3	-1.38×10^3	2.97×10^2	2.07×10^2	2.41×10^2
f_2	-1.87×10^3	-1.97×10^3	-8.26×10^2	9.92×10^1	3.67×10^1	1.60×10^2
f_3	-7.73	-9.24	-2.73	7.44×10^{-1}	3.64×10^{-1}	4.11×10^{-1}
f_4	1.67	1.18	2.27	3.27	1.78	3.62
f_5	2.64×10^{-10}	7.76×10^{-10}	2.02×10^{-39}	2.38×10^{-10}	4.68×10^{-10}	9.03×10^{-39}
f_6	-3.54×10^3	-3.70×10^3	-1.45×10^3	2.66×10^2	1.72×10^2	2.70×10^2
f_7	2.15×10^{-5}	3.12×10^{-5}	4.44×10^{-15}	9.24×10^{-6}	1.07×10^{-5}	0
f_8	5.37	5.57×10^{-1}	6.71	2.63	7.56×10^{-1}	5.08
f_9	1.25×10^{-1}	8.93×10^{-2}	1.05×10^{-2}	9.40×10^{-2}	5.28×10^{-2}	9.22×10^{-3}
f_{10}	1.45×10^{-3}	8.64×10^{-4}	8.50×10^{-4}	3.49×10^{-4}	3.44×10^{-4}	2.67×10^{-4}

Samples size: 50 Parameters: N = 25 n = 10 N_{it} = 4000

Table 4. Summary statistics for 30-dimensional problems

No.	Samples means			Samples deviations		
	μ_{WCA}	$\mu_{WCA-FPA}$	$\mu_{GSA-FPA}$	σ_{WCA}	$\sigma_{WCA-FPA}$	$\sigma_{GSA-FPA}$
f_1	-8.97×10^3	-1.15×10^4	-2.65×10^3	8.76×10^2	5.90×10^2	5.17×10^2
f_2	-4.84×10^3	-5.77×10^3	-1.44×10^3	3.00×10^2	1.76×10^2	2.53×10^2
f_3	-1.94×10^1	-2.44×10^1	-1.66×10^1	1.67	1.29	2.03
f_4	8.48	1.75×10^1	6.10×10^1	8.53	1.02×10^1	3.53×10^2
f_5	1.44×10^{-9}	2.69×10^{-9}	5.24×10^{-1}	8.16×10^{-10}	1.41×10^{-9}	3.71
f_6	-9.24×10^3	-1.07×10^4	-2.50×10^3	5.85×10^2	4.58×10^2	5.08×10^2
f_7	6.82	4.29×10^{-5}	8.42×10^{-15}	8.73	1.17×10^{-5}	1.37×10^{-15}
f_8	1.33×10^2	9.41	9.77×10^1	3.16×10^1	3.14	4.08×10^1
f_9	1.08×10^{-1}	7.40×10^{-2}	1.33×10^{-3}	1.15×10^{-1}	6.94×10^{-2}	5.20×10^{-3}
f_{10}	1.08×10^{-11}	1.36×10^{-11}	7.90×10^{-12}	1.07×10^{-12}	1.92×10^{-12}	1.00×10^{-12}

Samples size: 50 Parameters: $N = 25$ n = 30 $N_{it} = 12000$

Table 5 shows the two-sample Z-test results, which are interpreted as follows:

Table 5. Two-sample Z-test results

No.	10-dimensional				30-dimensional			
	z_{WSA}		$z_{GSA-FPA}$		z_{WSA}		$z_{GSA-FPA}$	
f_1	9.33	$+^{99\ \%}$	58.82	$+^{99\ \%}$	16.87	$+^{99\ \%}$	79.65	$+^{99\ \%}$
f_2	6.40	$+^{99\ \%}$	49.08	$+^{99\ \%}$	18.77	$+^{99\ \%}$	99.26	$+^{99\ \%}$
f_3	12.89	$+^{99\ \%}$	83.90	$+^{99\ \%}$	16.83	$+^{99\ \%}$	22.79	$+^{99\ \%}$
f_4	0.93	$\approx$	1.91	$+^{95\ \%}$	−4.81	$-^{99\ \%}$	0.87	$\approx$
f_5	−6.91	$-^{99\ \%}$	−11.74	$-^{99\ \%}$	−5.43	$-^{99\ \%}$	1.00	$\approx$
f_6	3.58	$+^{99\ \%}$	49.84	$+^{99\ \%}$	14.18	$+^{99\ \%}$	85.01	$+^{99\ \%}$
f_7	−4.85	$-^{99\ \%}$	−20.59	$+^{99\ \%}$	5.53	$+^{99\ \%}$	−25.82	$-^{99\ \%}$
f_8	12.45	$+^{99\ \%}$	8.46	$+^{99\ \%}$	27.49	$+^{99\ \%}$	15.26	$+^{99\ \%}$
f_9	2.36	$+^{99\ \%}$	−10.39	$+^{99\ \%}$	1.79	$+^{95\ \%}$	−7.39	$-^{99\ \%}$
f_{10}	8.48	$+^{99\ \%}$	−0.23	$\approx$	−8.95	$-^{99\ \%}$	−18.60	$-^{99\ \%}$

Sample size = 50 $z_{99\ \%} = \pm\ 2.33$ $z_{95\ \%} = \pm\ 1.645$

For example, for the function 4 (i.e. Rosenbrok's) as a 10-dimensional problem the z-score against the WCA is 0.93 which means there is not enough evidence to reject the null hypothesis of a right tailed test, hence the results are similar (i.e. $\approx$).

Same problem but against the GSA-FPA is a z-score of 1.91 which means there is enough evidence with a 95% of confidence to accept the alternative hypothesis that the WCA-FPA is significantly better (i.e. $+^{95\%}$).

Now the same function but as a 30-dimensional problem with a z-score of -4.81 in a left-tailed test the original WCA is significantly better than our proposal with a 99% of confidence (i.e. $-^{99\%}$).

6 Conclusions

From the hypothesis test results can be concluded that dynamically adapting the parameter "C" can significantly improve the performance of the water cycle algorithm. Although for most of the tests problems our proposal is significantly better with a 99% of confidence it seems to be something in common with functions in which it did worse. Those functions are either unimodal or have a predominant valley, which could mean that our proposal has some exploitation problems. This problem could be solved by setting a smaller value for d_{max}, or by extending the fuzzy adaptive algorithm to also adapt the parameter "d" in a non-linear manner.

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 **220**(4598), 671–680 (1983)
4. Goldberg, D.E.: Genetic Algorithms in Search. Optimization and Machine Learning. Addison-Wesley Longman Publishing Co. Inc., Boston (1989)
5. Hosseini, H.S.: Problem solving by intelligent water drops. In: IEEE Congress on Evolutionary Computation, CEC 2007, 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: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 101–106 (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. Sombra, A., et al.: A new gravitational search algorithm using fuzzy logic to parameter adaptation. In: 2013 IEEE Congress on Evolutionary Computation, pp. 1068–1074 (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)