Neural Comput & Applic (2014) 25:1785–1799
DOI 10.1007/s00521-014-1669-y

ORIGINAL ARTICLE

Hybrid particle swarm optimization for parameter estimation of Muskingum model

Aijia Ouyang · Kenli Li · Tung Khac Truong · Ahmed Sallam · Edwin H.-M. Sha

Received: 27 July 2013 / Accepted: 5 July 2014 / Published online: 22 July 2014

Abstract The Muskingum model is the most widely used and efficient method for flood routing in hydrologic engineering; however, the applications of this model still suffer from a lack of an efficient method for parameter estimation. Thus, in this paper, we present a hybrid particle swarm optimization (HPSO) to estimate the Muskingum model parameters by employing PSO hybridized with Nelder–Mead simplex method. The HPSO algorithm does not require initial values for each parameter, which helps to avoid the subjective estimation usually found in traditional estimation methods and to decrease the computation for global optimum search of the parameter values. We have carried out a set of simulation experiments to test the proposed model when applied to a Muskingum model, and we compared the results with eight superior methods. The results show that our scheme can improve the search accuracy and the convergence speed of Muskingum model for flood routing; that is, it has higher precision and faster convergence compared with other techniques.

Keywords Particle swarm optimization · Nelder–Mead simplex method · Muskingum model · Hybrid algorithm · Parameter estimation

1 Introduction

Flood routing is a fundamental step in disaster managements; thus, we carry out intensive research in this area and we figure out that flood routing procedures can be classified into two types: hydrologic methods and hydraulic ones. Hydrologic methods employ the basic principle of continuity and the relationship between the discharge and the temporary storage dedicated for the excess volumes of water during the flooding periods. On the other hand, hydraulic methods employ approximate solutions for gradually varied and unsteady flow in open channels, based on either the convection-diffusion equations or the one-dimensional Saint-Venant equations.

Compared with hydrologic techniques, the hydraulic methods usually present a more accurate description of the flood wave profile, whereas these methods are restricted to particular applications because of their higher requirements for computing technologies. In fact, the hydrologic routing approaches are relatively easy to execute and give fairly precise results. Among all the hydrologic models used for flood routing, the Muskingum model is the most widely used in view of its simplicity.

Muskingum model was designed for the first time by McCarthy for flood control and management in the Muskingum River, Ohio. According to a large group of water

A. Ouyang · K. Li (✉)
College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China
e-mail: lkl@hnu.edu.cn

A. Ouyang
e-mail: oyaj@hnu.edu.cn

T. K. Truong
Faculty of Information Technology, Industrial University of Hochiminh City, Hochiminh, Vietnam
e-mail: truongkhactung@yahoo.com

A. Sallam
Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt
e-mail: A.Sallam@hnu.edu.cn

E. H.-M. Sha
College of Computer Science, Chongqing University, Chongqing 400044, China
e-mail: edwinsha@cqu.edu.cn

resources engineers, the Muskingum model is a very valuable tool for flood forecasting and disaster management. But they indicate that the most difficult task is to estimate parameters of employing the Muskingum model because these parameters can not be graphically derived from historical inflow or outflow hydrography.

In the past years, some researchers adopted many optimization methods to optimize the parameters of the Muskingum model. In 1997, Mohan [1] used genetic algorithm to estimate these parameters and verified its performance in comparison with another method proposed by Yoon and Padmanabhan [2]. After a while, Geem [3] introduced the Broydene-Fletchere-Goldfarbe-Shanno (BFGS) method, which searches the solution area based on gradients to estimate the Muskingum parameters. Later, another remarkable approach was proposed by Chen [4] where he applied Gray-encoded accelerating genetic algorithm (GAGA) to optimize these parameters. In 2009, Chu [5] applied PSO to estimate the parameters from another perspective. Afterward, Luo [6] utilized an immune clonal selection algorithm (ICSA) for the same task. In 2011, Barati [7] used Nelder–Mead simplex method (NMSM) to estimate these parameters. At the same time, Xu et al. [8] adopted differential evolution to estimate these parameters. BFGS and NMSM belong to local optimization algorithms, while the rest ones of the aforementioned methods belong to global optimization algorithms.

PSO is a swarm intelligence algorithm based on the imitation of social interaction and creatures' communication such as bird flocks and fish schools [9]. PSO shares many similarities with swarm intelligence optimization algorithms and has been proved to be an effective approach, which can tackle a variety of difficult optimization problems. In [10], an improved cooperative PSO was applied to train the feedforward neural network. In 2012, Ghosh [11] proposed an inertia-adaptive PSO with particle mobility factor to optimize the global optimization problems. Meanwhile, Wang [12] used a converging linear PSO to train support vector data descriptors. Then, Jia [13] combined multi-objective PSO with Pareto-optimal solutions to solve the batch processes problem. Another important application for PSO was proposed by Lee [14] using a hybrid GA-PSO for network decomposition. In addition, PSO was applied to the optimization of fuzzy controllers [15]. Recently, Altun [16] solved the problem of cost optimization of mixed feeds by a PSO.

In recent years, there are many excellent variants of PSO have been designed, e.g., comprehensive learning PSO [17], DMS-PSO [18–20], adaptive PSO [27], orthogonal learning PSO [28], PSO with an aging leader and challengers [29], cooperatively coevolving PSO [30], distance-based locally informed PSO [31], quantum-based PSO [32], and parallel hybrid PSO [33]. In this paper, we combine the PSO with the NMSM to optimize the parameters of the Muskingum model.

The remainder of our work is organized as follows. We discuss the background of Muskingum models in Sect. 2. We then describe the main principles of the PSO and the NMSM and present a hybrid PSO in Sect. 3. Section 4 consists of experimental results and analyses. Section 5 provides the conclusions of the paper.

2 Muskingum models

In this section, we introduce the Muskingum model in order to describe the flood routing. Using the continuity equations [4], we introduce the flow conditions with two different locations on the same river as follows:

$$\frac{\mathrm{dW}}{\mathrm{d}t} = I - Q, \tag{1}$$

$$W = k(xI + (1 - x)Q), \tag{2}$$

In the two equations above, I denotes the inflow discharges and Q denotes the outflow discharges, W denotes the storage volume, k and x represent the model parameters, and t denotes the time.

To forecast a flood wave, we have deduced the following routing equation Eq. (3) by combining the Eqs. (1) and (2):

$$Q(i) = c_1 I(i) + c_2 I(i-1) + c_3 Q(i-1),\ i = 2, \cdots, n, \tag{3}$$

where $I(i)$ and $Q(i)$ describe the observed inflow discharge and the observed outflow discharge, respectively, at a time interval t_i, n is the maximum time point in all time; and c_1, c_2, and c_3 are constant values and must satisfy the following three equations [34]:

$$c_1 = \frac{0.5\Delta t - kx}{k - kx + 0.5\Delta t}, \tag{4}$$

$$c_2 = \frac{0.5\Delta t + kx}{k - kx + 0.5\Delta t}, \tag{5}$$

$$c_3 = \frac{k - kx - 0.5\Delta t}{k - kx + 0.5\Delta t}, \tag{6}$$

where Δt is the time step.

From Eqs. (4–6), we can obtain Eqs. (7) and (8):

$$k = \frac{c_2 + c_3}{c_1 + c_2}\Delta t, \tag{7}$$

$$x = \frac{c_1 + c_2}{2(c_2 + c_3)} + \frac{c_1}{c_1 - 1}, \tag{8}$$

We apply the following fitness function so as to estimate these parameters efficiently,

$$\min\ f(k,x)=\sum_{i=1}^{n-1}|(kx-0.5\Delta t)I(i+1)-(kx+0.5\Delta t)I(i)+Q(i+1)+[-k(1-x)+0.5\Delta t]Q(i)| \tag{9}$$

3 Hybrid particle swarm optimization algorithm

In this section, first, we introduce the standard PSO algorithm, followed by an explanation of the NMSM. Finally, our new hybrid PSO (HPSO) algorithm is presented.

3.1 Particle swarm optimization algorithm

3.1.1 The fundamental principle of PSO

Based on its unique search mechanism, PSO algorithm randomly initializes the particle swarm within the feasible solution space and velocity space in the first place. Put in another way, the initial position and velocity of the particle can be determined. The position can be subsequently adopted as to characterize the problem solution. For instance, the position and velocity of the particle in the i place within a d-dimensional search space can be expressed as $X_i=[x_{i,1},x_{i,2},\cdots,x_{i,d}]$ and $V_i=[v_{i,1},v_{i,2},\cdots,v_{i,d}]$, respectively. The best position of each particle at the time t, i.e., (*pbest*), $P_i=[p_{i,1},p_{i,2},\cdots,p_{i,d}]$, and the best known position of swarm, i.e., $(gbest)P_g$, can be ascertained through evaluating the objective function of each particle. The velocity and position of each particle can be updated using the following equations:

$$v_{i,j}(t+1)=\omega v_{i,i}(t)+c_1r_1[p_{i,j}-x_{i,j}(t)]+c_2r_2[p_{g,j}-x_{i,j}(t)] \tag{10}$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1),j=1,2,\cdots,d \tag{11}$$

Here, ω is the inertia weight factor, ω is a linearly decreasing, which changes from 0.9 to 0.4. c_1 and c_2 are positive acceleration constants, n_1 and n_2 refer to random numbers uniformly distributed between 0 and 1. In addition, the migration of particles can be appropriately limited if we make some settings to the particles? velocity range, i.e., $[v_{\min},v_{\max}]$ and their position interval, i.e., $[x_{\min},x_{\max}]$.

As can be seen from the above equations, updating the velocity of particles has to go through three stages.

Firstly, the effect of the current velocity of the particle gets reflected. By taking the current status of particles into consideration, the algorithm can strike a balance between its global and local search ability;

Secondly, the effect of the cognition modal can be checked. In actuality, this is the very impact of the memory of the particle itself, which enables the particle to do global search in order to avoid local optimum;

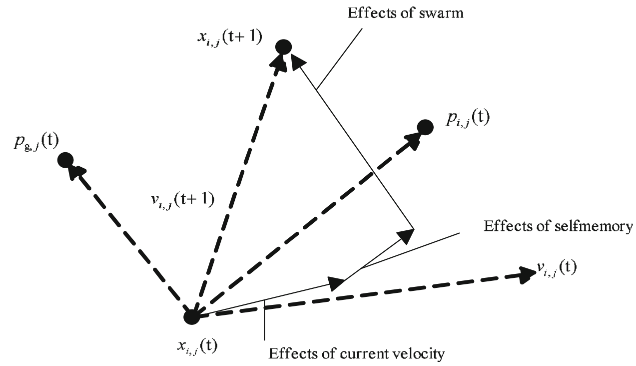

Fig. 1 The process of position update with one particle

Thirdly, the effect of the social modal can be seen. In other words, the impact of the swarm information demonstrates the information sharing among particles. Under the combined effects of these three stages, the particle is able to use the information sharing mechanism according to historical experience and constantly adjusts its position in order to find a better solution [35].

The above-mentioned way of updating particle positions in every generation in PSO algorithm is described in Fig. 1.

PSO algorithm has two versions, namely the global version and the local version. In the global version, the two extreme values of particle tracking are the best location of the particle and that of the whole population. In the local version, the particle only tracks its own optimal position. Instead of tracking the best position of the population, the particle tracks the best location of all the particles in the topology field, whose equation of updating its velocity can be seen as follows:

$$v_{i,j}(t+1)=wv_{i,j}(t)+c_1r_1[p_{i,j}-x_{i,j}(t)]+c_2r_2[p_{l,j}-x_{i,j}(t)] \tag{12}$$

Here, $P_l=[p_{l,1},p_{l,2},\ldots,p_{l,d}]$ is the best location in the local field. A global version PSO algorithm is adopted in the present paper.

3.1.2 The flow of PSO algorithm

The process of basic PSO algorithm is as follows.

Step 1 Initialize randomly the position and velocity of each particle in the population. If the search space is d-dimensional, each particle contains d variables.

Step 2 Evaluate all the particles in the population and store their current positions and fitness values in their *pbest*. The positions and fitness values of individuals with optimal fitness values in *pbest* stored in *gbest*.

Step 3 Update the position and velocity of each particle in accordance with Eqs. (10) and (11), respectively.

Step 4 Evaluate all the particles in the population.

Step 5 Compare the current fitness value of each particle in the whole population with the fitness value of its *pbest*. If the current fitness value is superior, use the particles current position and fitness value to update its *pbest*.

Step 6 Compare the fitness values of all the and *pbest*, *gbest* and update *gbest*.

Step 7 If the termination criterion is met, output *gbest* and the fitness value. Meanwhile, stop updating the algorithm. Otherwise, go to Step 3.

3.1.3 Features of PSO algorithm

To sum up, PSO algorithm has the following advantages.

1. The algorithm has wide universality and does not rely on the information of the problem.
2. It uses swarm search and has memory capacity, which can retain the optimal information of local individuals and the global population.
3. Its principle is simple and easy to implement.
4. Using cooperative search, it utilizes the local information of individuals and global information of swarm to guide the search.

There is no denying that PSO algorithm has some shortcomings, to be more detailed,

1. Its local search ability is poor with low accuracy.
2. The algorithm cannot guarantee that the global optimal solution can be searched, which is highly likely to fall into local optimum.
3. The performance of the searching algorithm is, to some extent, dependent on the parameters.
4. The theory behind the algorithm is yet to be improved, in particular, the principle guiding the algorithm design is missing [36].

3.2 Nelder–Mead simplex method

3.2.1 Basic principle of NMSM

Nelder–Mead simplex method (NMSM),also known as variable polyhedron search method, is a traditional direct unconstrained optimization algorithm. The advantages of NMSM lie in its low computational complexity, fast search speed, and strong local search ability [37]. The basic ideas of NMSM are as follows: In the N-dimensional space, a polyhedron is composed of $N+1$ vertexes. Then, the fitness value of each vertex is calculated, and therefore, the vertexes with the best value, the sub-optimal value, and the worst value can be identified. Furthermore, through strategies of reflection, expansion, shrink, and contraction, a vertex with a better value gets found to replace the worst vertex. Ultimately, a new polyhedron can be generated. By repeating this iteration, an optimal or close-to-optimal solution will be found. NMSM starts with a randomly generated initial simplex and takes the following steps to constantly change the shape of simplex: First, find the vertex with the maximum value of the objective function W, the vertex with the minimum value B, and the one with the second largest value N_w, respectively; Second, calculate the values of C, the centroid of vertexes which totals D, except for Vertex W, and further the value of reflection point of C on the part of W. If the objective function value at the location of R is smaller than that located at B, one outward expansion in the same direction needs to be executed; otherwise, if f_R is greater than the function value of N_w, execute one inward contraction in the direction accordingly. When detecting the trough, the method executes one inward contraction in all directions. This can help the whole simplex get close to the lowest point and proceed to the next iteration [39].

3.2.2 Flow of NMSM

The NMSM [38] was proposed by John Nelder and Roger Mead (1965) to obtain a minimization of a value for an fitness function in a many dimensional space. NMSM was applied for solving the optimization problems since it can obtain an accurate value for fitness functions. Let $f(x)$ denotes the fitness function, and x_i denotes the group of points in the current simplex, $i \in \{1,\ldots,N+1\}$. NMSM procedure has the following steps:

Step 1: (Order) The inequality group $f(x_1) \le f(x_2) \le \cdots \le f(x_i) \le \cdots \le f(x_{N+1})$ must be satisfied for the $N+1$ vertices.

Step 2: (Reflect) First, calculate the reflection point x_R of the simplex by $x_R = \bar{x} + \alpha(\bar{x} - x_{N+1})$, $(\alpha > 0)$, here, $\bar{x} = \frac{1}{N}\sum_{i=1}^{N}$ is the centroid of the n best points then compute $f_R = f(x_R)$. If $f_1 \le f_R \le f_N$, accept the reflected point x_R and end the iteration.

Step 3: (Expand) If $f_R < f_1$, compute the expansion point x_E of the simplex by $x_E = \bar{x} + \beta(x_R - \bar{x})$,$(\beta > 1$ and $\beta > \alpha)$. Calculate $f_E = f(x_E)$. If $f_E < f_R$, accept x_E and end the iteration; if $f_E \ge f_R$, accept x_R and end the iteration.

Step 4: (Contract) If $f_R \geq f_N$, execute a contraction operation between $\bar{x}$ and the better of x_R and x_{n+1}. a. Outside contraction: If $f_N \leq f_R < f_{n+1}$, compute the outside contraction point x_C by $x_C = \bar{x} + \gamma(x_R - \bar{x}), \quad (0 < \gamma < 1)$. Calculate $f_C = f(x_C)$. If $f_C < f_R$, accept x_C and end the iteration; otherwise, continue with **Step 5**. b. Inside contraction: If $f_R \geq f_{N+1}$, compute the inside contraction point x_{CC} by $x_{CC} = \bar{x} - \gamma(\bar{x} - x_{N+1})$. Calculate $f_{CC} = f(x_{CC})$. If $f_{CC} < f_R$, accept x_{CC} and end the iteration; or else continue with **Step 5**.

Step 5: (Shrink) Compute the N points by $v_i = x_1 + \sigma(x_i - x_1)(0 < \sigma < 1)$. Also calculate $f(v_i), \quad i = 2, \cdots, N+1$. The unordered vertices of the simplex at the next iteration comprise of $x_1, v_2, \cdots, v_{N+1}$. Furthermore, go to **Step 1**.

3.3 Hybrid particle swarm optimization (HPSO) algorithm

3.3.1 Feasibility analysis of hybrid algorithm

1. Organic integration of the mechanism: NMSM is a deterministic method for optimization while PSO is an algorithm based on random distribution. The organic integration of NMSM and PSO not only enriches the searching behavior in the optimization process, but also improves the search ability and efficiency of HPSO in a bid to obtain high-quality solutions [37].
2. Good combination of operation: compared with other swarm intelligence algorithms, HPSO is higher in search efficiency and faster in outlining the shape of the objective function, which provides a good initial point for NMSM and gives full play to the powerful local search ability of NMSM. Thereby, NMSM and PSO can be organically combined.
3. Wide applicability: PSO and NMSM do not require derivation or other auxiliary information. The embedding of NMSM in PSO does not limit but instead, enhances the applicability of HPSO algorithm.
4. Characteristics of parallel computing: Both PSO and NMSM have the feature of parallel computing; hence, it is very suitable to combine the two methods.

3.3.2 Hybrid strategy

Based on PSO process, NMSM that constitutes of HPSO algorithm is introduced to improve the local fine-tuning of algorithms and increase PSO algorithm's probability to converge the global optimal solution. In each iteration, PSO algorithm is first used to perform the global optimization, followed by simplex method to conduct the local search of some elite particles among the particle swarm in the domain featuring good solutions find out a better solution [39]. Specifically,

1. PSO algorithm: as PSO algorithm has the powerful global search ability, it is easy to for the particle swarm to search the surrounding area with the global optimal solution after PSO operation. Based on this, NMSM is utilized to perform the local search so as to obtain optimal solution with higher precision.
2. NMSM: P (population size) swarm particles optimized by PSO in each iteration are sorted according to their fitness value. The first S selected particles with the best fitness value constitute the NMSM graphics with S vertexes. $(S-1)$ vertexes $X_1, X_2, \cdots, X_{S-1}$ with the best response get selected from S vertexes, and the centroid of $S-1$ vertexes X_c is calculated. The rest vertexes X_s pass through the centroid X_c. X_S' scalability mapping generates X vertexes to constitute a new NMSM graphics. S new particles are generated with the repeated method above. Fitness value of each updated particle is calculated, from which particles with the best response is selected to replace the best individuals in the original swarm, and constitute the next generation with the rest individuals in the original swarm. The elite individuals in the population, after repeated iterations with the simplex method, find out an approximate optimal position. In addition, the search accuracy of the algorithm and the probability to find out the optimal solution sooner can be increased. In general, S , the number of individuals, should not be too large. The ratio of S and (S/P) between 10 % and 20 % is appropriate. Due to the fast convergence speed of PSO in the early evolution, the NMSM with small probability is conducted for local search to find out the optimal solution, in order to lower the amount of calculation and improve the computational efficiency. In the late evolution, when swarm enters the development stage of the local search, the NMSM optimization search with large probability is adopted. Following this idea, an adaptive strategy model based on the evolution stages is given in the following section [37]. The evolutionary process is divided into three stages:

The first stage: $\tau \in [0, T_1] : T_1 = aT$
The second stage:$\tau \in [T_1, T_2] : T_2 = (1-a)T$
The third stage:$\tau \in [T_2, T]$

Table 1 Probability called of NMSM in different stages

τ	$[0, T_1]$	$(T_1, T_2]$	$(T_2, T]$
P	0.05	0.10	0.15

T is the maximum evolution algebra, τ is evolution algebra, and the value of a is usually set as 0.382. p, the probability of NMSM used in each stage, can be seen in Table 1.

3.3.3 *Characteristics of hybrid algorithm*

Based on the framework of PSO, hybrid algorithm introduces the NMSM to perform repeated simplex search and iteration on some elite particles in the swarm. Characteristics of this method are as follows:

1. Due to its own defects of the inherent mode, every search algorithm with single structure and mechanism is generally difficult to realize highly efficient optimization, and so is PSO algorithm. HPSO algorithm is better in optimization performance than single PSO algorithm and another optimization method.
2. PSO, an algorithm based on random search, has a strong ability of global optimization but gradually slow convergence speed in later algorithm stage. NMSM, a deterministic and descent method, has a superior local search ability. Using the polyhedral reflection, expansion, compression, and other properties, it is able to quickly find out the local optimal solution. The two algorithms complement each other, and therefore, their combination is conducive to the improvement of the global and local search ability and efficiency.
3. NMSM is simple featuring low complexity, and fast speed [37]. Probability in stage is used to perform simplex iteration and update on some elite particles in the swarm, with a limited number of particles involved. Therefore, the hybrid algorithm combining NMSM algorithm and PSO algorithm does not require much computation.

3.3.4 *Flow of HPSO*

Based on the previous work, the process of HPSO is described as follows:

Step 1 assign values parameters of HPSO, including population size of particle swarm P, compressibility factor K, acceleration factor φ_1, φ_2, probability P of NMSM, and the maximum algebra T required for evolution computation;

Step 2 initialize population, generate initial velocity and position of particles, and calculate particles' fitness value according to the evaluation function;

Step 3 set the present position of particle as *pbest*, and the best position of the initial group as *gbest*;

Step 4 update the velocity and position of each particle and evaluate the particles fitness value;

Step 5 sort the particle swarm according to the fitness value; for the first S elite particles, use NMSM with the possibility P to perform repeated iterations and updates on the first S selected excellent individuals; calculate the fitness value of each updated particle; replace the best individuals of the original swarm with the particle with the best response, and ultimately, constitute the next negation with the remaining individuals of the original group;

Step 6 compare the fitness value of new individuals and their *pbest* fitness value; if the fitness value of the particle is better than *pbest* fitness value, set *pbest* as the new position;

Step 7 compare the fitness value of new individuals with that of *gbest*; if the fitness value of the particle is better than *gbest* fitness value, set *gbest*as the new position;

Step 8 if the evolution computation of the particle swarm achieves the allowed maximum algebra T or the evaluation value is less than the set accuracy, output the optimal result *gbest* and iteration ends. Otherwise, if $\tau = \tau + 1$, search continues from step 4.

From the above, we can see that PSO searches the approximate space of the optimal solution located, then it gives the solution to NMSM for further depth search. The roles of the two algorithms are different; the most important role of HPSO is to strengthen the division and the cooperation through the merging and the recombination between PSO and NMSM [39]. In a word, PSO performs an exploration while using NMSM for an exploitation to make HPSO search the optimal solution very precisely and faster.

4 Experiments and result analyses

4.1 simulation results of real example

The Muskingum model investigated in our paper is from Ref. [40], where the model was in the south canal between Chenggou river and Linqing river in China, and the time interval Δt = 12 h. The detailed information can be found in Ref. [41].

The parameters k and x are required in this model, and the significance of these parameters can be clearly seen in Eqs. (4–6).

The parameters k and x play an important role in the Muskingum model. In our study, k and x are optimized with respect to the same criterion, termed the sum of the least residual absolute value, and the form of the fitness function is shown in Eq. (9).

To facilitate the experiments, we used matlab2012a to program a m-file for implementing the algorithms on a PC with a 32-bit windows 7 operating system, a 2GB RAM, and a CPU of Pentium Dual-core with 2.7 GHz. The standard errors have been measured for 30 independent runs. For all the intelligence algorithms, the two parameters are set as follows: population size $N_p = 20$, the maximum number of iterations Max_{IT} = 1,000. These parameters of HPSO, CLPSO, DMS-PSO, and PSO used are set the same: the acceleration constants $c_1 = 2.1$, $c_2 = 2.1$; the inertia weight factor ω is a linearly decreasing, which changes from 0.9 to 0.4. These parameters of SaDE, DPSDE, and DE used are set the same: constriction factor $F_c = 0.5$, crossover rate $P_c = 0.6$, These parameters of GAGA,

Fig. 2 The comparison of fitness values between HPSO and other methods

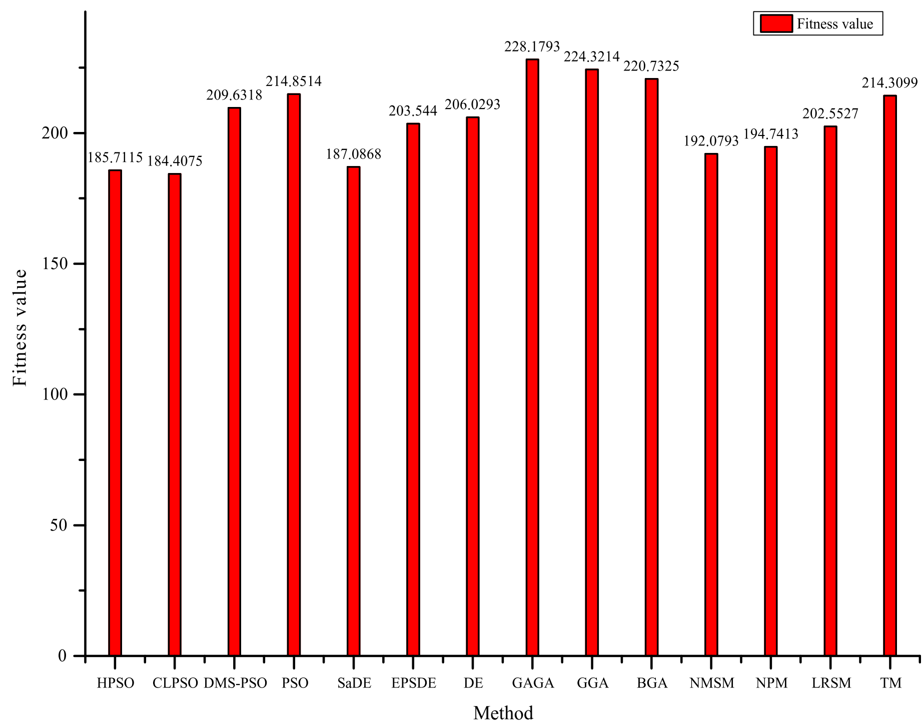

Table 2 A comparison of flood routing between different methods

Method	EN	k	x	AAE	ARE (%)	Fitness	SD
HPSO	2,000	12.7800	−0.0930	**6.8447**	**2.0151**	185.7115	**6.5320E−02**
CLPSO	10,800	12.7889	−0.0681	7.1628	2.0187	**184.4075**	3.4987E−01
DMS-PSO	9,200	13.2508	−0.5244	7.0358	2.0460	209.6318	1.0773E+01
PSO	12,000	12.9801	−0.6514	7.2463	2.1093	214.8514	1.2523E+01
SaDE	**1,400**	13.4552	−0.0511	7.2302	2.1375	187.0868	3.0984E+00
EPSDE	4,200	14.1001	−0.3509	7.0983	2.0761	203.5440	1.5956E+01
DE	5,000	12.8011	−0.5007	7.1317	2.0734	206.0293	1.2925E+01
GAGA	31,200	13.2104	−0.9048	7.3471	2.1275	228.1793	4.0085E+01
GGA	33,600	12.9876	−0.8612	7.2919	2.1136	224.3241	1.7916E+01
BGA	43,800	14.7085	−0.6011	7.2390	2.1084	220.7325	1.0453E+01
NMSM	–	12.4569	−0.1984	7.0001	2.0421	192.0793	6.3793E+00
NPM	–	12.4471	−0.2616	7.1021	2.0811	194.7413	3.2002E+00
LRSM	–	11.7916	−0.3520	7.4119	2.1941	202.5527	1.0015E+01
TM	–	12.0000	0.1000	8.7407	2.6305	214.3099	3.3806E+01

GGA, and BGA used are set the same: selection rate is from Roulette, crossover rate $P_c = 0.8$, mutation rate $P_m = 0.05$.

We made a comparison between using our new proposed model HPSO and another nine evolution algorithms PSO [5], CLPSO [17], DMS-PSO [20], EPSDE [21, 22], SaDE [23, 24], DE [8], GAGA [4], GGA [25] and BGA [26], and the comparison results show that: for HPSO, the average absolute error (AAE) is 6.8447, and the average relative error (ARE) is 2.0151. For CLPSO, DMS-PSO, and PSO, the AAE is 7.1628, 7.0358, 7.2463, and the ARE is 2.0187, 2.0460, 2.1093, respectively. For SaDE, EDSDE, and DE, the AAE is 7.2302, 7.0983, 7.1317, and the ARE is 2.1375, 2.0761, 2.0734, respectively. For GAGA, GGA, and BGA, the AAE is 7.3471, 7.2919, 7.2390, and the ARE is 2.1275, 2.1136, 2.1084, respectively. The experimental data for the model are listed in Table 1 in detail. These results proved that HPSO has higher precision compared with the above three different algorithms.

Table 3 Statistic data of description

Method	Run	Min	Mean	Max	SD
HPSO	30	185.6915	185.7115	185.9915	6.5320E−02
CLPSO	30	184.3140	184.4075	185.7166	3.4987E−01
DMS-PSO	30	185.6318	209.6318	215.6318	1.0773E+01
PSO	30	185.5521	214.8514	246.7514	1.2523E+01
SaDE	30	185.0868	187.0868	193.0868	3.0984E+00
EPSDE	30	185.5440	203.5440	228.5440	1.5956E+01
DE	30	185.0293	206.0293	226.0293	1.2925E+01
GAGA	30	185.1794	228.1793	278.1793	4.0085E+01
GGA	30	185.3251	224.3241	234.3251	1.7916E+01
BGA	30	184.8425	220.7325	230.7226	1.0453E+01
NMSM	30	185.0793	192.0793	203.0793	6.3793E+00
NPM	30	185.1246	194.7413	200.6557	3.2002E+00
LRSM	30	185.5040	202.5527	247.5528	1.0015E+01
TM	30	188.2015	214.3099	330.2966	3.3806E+01

On the other hand, we also perform the same test on the four conventional methods, and the test results are as follows: for NMSM [7], the AAE is 6.9878, and the ARE is 2.0570. For the nonlinear programming method (NPM) [42], the AAE is 7.1021, and the ARE is 2.0811. For the least residual square method (LRSM) [34], the AAE is 7.4119, and the ARE is 2.1941. And for test method (TM) [34], the AAE is 8.7407, and the ARE is 2.6305. These results also show that HPSO has higher precision compared with these conventional methods, such as the NPM, the LRSM, and the TM (Fig. 2).

It is shown in Tables 2, 3 and Fig. 2, the fitness value f is 185.7115 for HPSO, it is the second best fitness value among the 14 methods. The evaluation number (EN) of the fitness function is 2000; it is the second smallest number in terms of function evaluation among the 14 methods. From Fig. 3, we can see that HPSO is second fastest method among the 14 methods behind SaDE in terms of convergent speed.

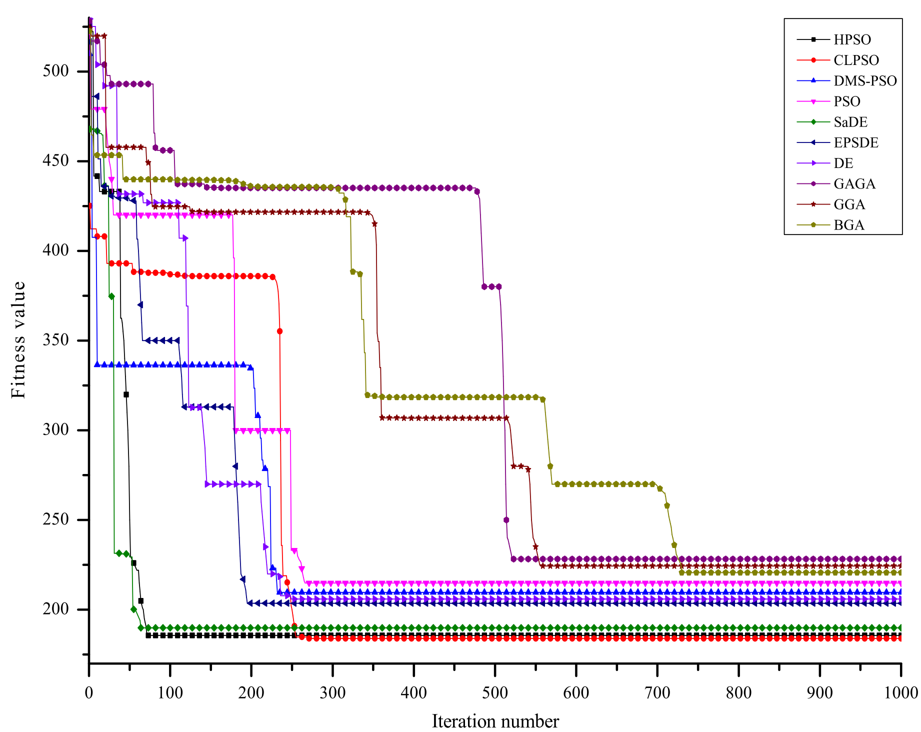

Fig. 3 The comparison of convergence between HPSO and other methods

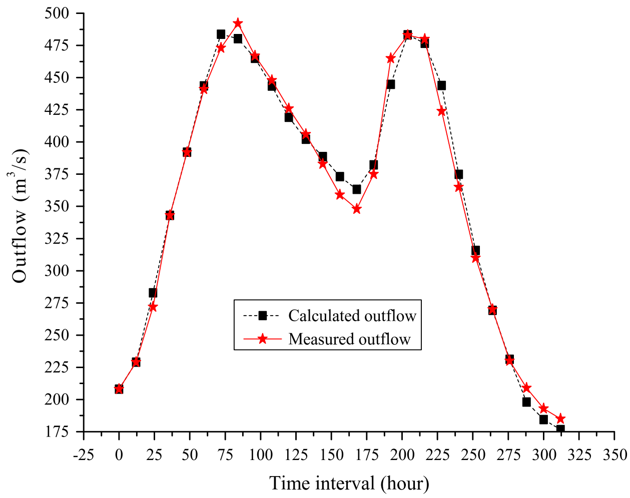

Fig. 4 The simulation results of HPSO for 1960 flood routing

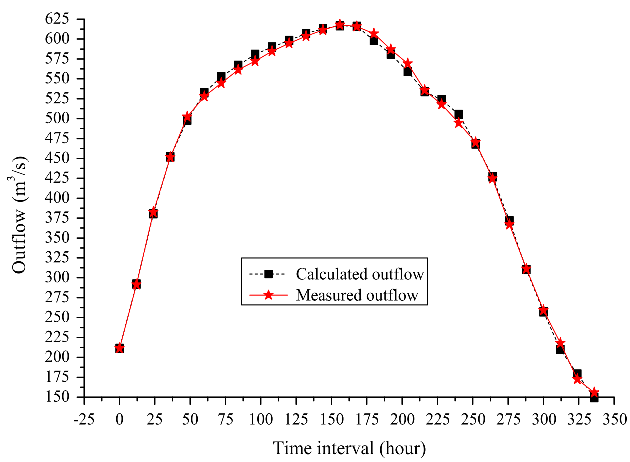

Fig. 5 The simulation results of HPSO for 1961 flood routing

The experimental results of the HPSO for the example in the practice are shown in Figs. 2, 3 and Tables 2, 3. In terms of evaluation number of objective function, SaDE is the best in 14 methods, HPSO ranked as the second best method, behind SaDE and ahead of the other methods. In terms of objective function value, CLPSO is the best in 14 methods, HPSO ranked as the second best method, behind CLPSO and ahead of the other methods. In terms of error and standard deviation, HPSO dominating it all the methods. We can conclude that the results obtained by our proposed HPSO are satisfactory in terms of precision and convergence.

Figure 4 gives the measured discharges and calculated discharges for the Muskingum model by the HPSO for 1960. Figure 5 gives the measured discharges and calculated ones for the Muskingum model by the HPSO for 1961. Figure 6 gives the measured discharges and calculated ones for the Muskingum model by the HPSO for 1964.

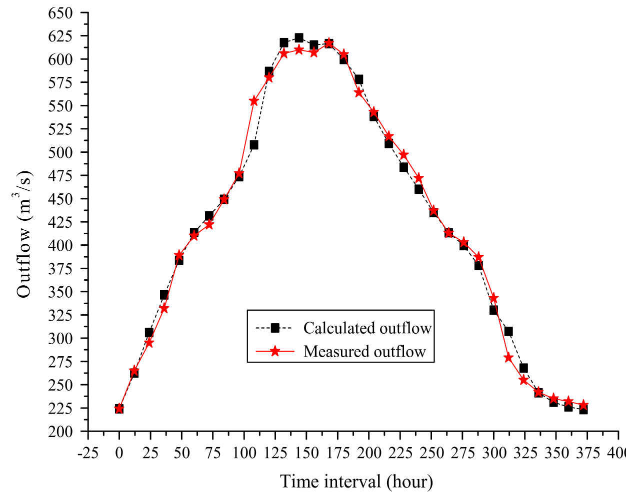

Fig. 6 The simulation results of HPSO for 1964 flood routing

From Table 2 and Figs. 2, 4, 5, and 6, we can see clearly that the simulation results obtained with our HPSO are satisfactory in terms of accuracy. The HPSO has been proved to be an efficient method to minimize the fitness function for the Muskingum model.

Tables 4, 5, and 6 show separately the comparisons of the best corresponding computed outflows obtained from various techniques such as TM, LRSM, NPM, NMSM, BGA, GGA, GAGA, DE, EPSDE, SaDE, PSO, DMS-PSO, CLPSO, and HPSO for 1960, 1961, and 1964. By comparing the results from the above three tables, HPSO outperforms the other algorithms in the three different periods.

As seen in the experimental results, the HPSO ranks as the first, first, first, second, second, second, respectively, out of 14 advanced methods in terms of absolute error, relative error, standard deviation, number of function evaluation, fitness value, convergent speed. To sum up: the overall performance of HPSO is very good.

4.2 Statistical results of p value

The Wilcoxon signed rank test was proposed by FWilcoxon in 1945. When there is concrete numerical value for the differences between the paired data of two groups, the signed test only adopts the positive ($R > 0$) and negative ($R < 0$) information. Information of differences in e size is not used. The Wilcoxon signed ranks test not only takes into consideration the positive and negative signs but also the differential size, so it has higher efficiency than the signed test [44].

The steps of this method are as follows:

Step 1 calculate the d_i value of every paired data, sort the d_i absolute value from small to large, and use the mean rank if the d_i value is equal.

Table 4 A comparison of calculated outflow with different methods for 1960

Time (h)	Inflow (m^3/s)	Outflow (m^3/s)	Routed outflow(m^3/s)													
			TM	LRSM	NPM	NMSM	BGA	GGA	GAGA	DE	SaDE	EPSDE	PSO	DMS-PSO	CLPSO	HPSO
0	197	208	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000
12	269	229	220.7143	235.2919	232.6250	230.8372	234.8738	235.9370	236.7847	233.4807	225.3514	227.3193	235.0319	231.1319	226.7229	229.0095
24	349	272	280.4286	287.8842	285.2355	284.3181	286.3237	286.5915	287.1504	284.9789	279.3906	279.2429	286.3775	282.8995	281.5158	282.9231
36	417	343	346.4286	345.5679	343.5754	343.6759	343.3390	342.8146	342.9741	342.3631	340.4491	338.4016	343.2845	341.0325	342.7506	343.0137
48	432	392	400.1429	390.4090	390.3128	391.5022	388.7209	387.6079	387.2248	388.7272	392.0654	389.0654	388.5758	389.0128	393.2296	392.0503
60	506	441	441.7143	448.1086	445.6765	444.8894	446.6021	446.7974	447.2923	445.3698	440.3985	440.1279	446.6449	443.4769	442.3989	443.6356
72	518	473	490.8571	482.0992	482.0571	483.1274	480.6269	479.6351	479.2861	480.6546	483.7077	481.0484	480.4973	480.9420	484.7058	483.6364
84	468	492	490.8571	474.4797	476.5567	478.5632	473.9860	472.5942	471.7315	475.0987	483.2082	480.1717	473.7890	477.1093	482.6859	480.2872
96	434	467	465.1429	463.1460	464.2295	464.4761	463.9561	463.9647	463.7786	464.5004	466.4138	466.7894	463.9496	465.3130	465.4300	464.9838
108	423	448	440.2857	443.8639	444.0625	443.6257	444.6556	445.0998	445.2247	444.7336	443.6882	444.9377	444.7124	444.7402	443.0794	443.4827
120	386	426	419.5714	417.2113	418.3972	418.6884	418.0694	418.0592	417.8485	418.6661	420.8220	421.1808	418.0597	419.5618	419.7643	419.2544
132	385	406	397.1429	403.6618	403.4570	402.6598	404.5102	405.1978	405.4806	404.3735	401.8381	403.6055	404.6017	403.9982	401.3567	402.1964
144	372	383	387.2857	388.5014	388.8445	388.6967	389.0593	389.2694	389.2849	389.2214	389.1732	389.8453	389.0844	389.4121	388.6609	388.7469
156	363	359	372.5714	372.8207	373.0796	373.0497	373.1327	373.2163	373.2011	373.2587	373.4587	373.7658	373.1418	373.4266	373.1448	373.1275
168	368	348	363.2857	363.5006	363.3441	363.3175	363.3716	363.3621	363.3860	363.2934	363.0430	362.9668	363.3715	363.1786	363.1931	363.2485
180	427	375	379.1429	386.2316	384.2210	383.3499	385.2801	385.6438	386.1256	384.2514	379.5034	379.8192	385.3420	382.6214	380.9614	382.2055
192	530	465	441.5714	451.0869	447.6796	446.5087	449.0672	449.4033	450.1191	447.3377	440.1757	439.9633	449.1354	444.6656	442.9177	444.7188
204	478	483	496.5714	476.4923	478.7620	481.2213	475.5969	473.8311	472.8003	476.8248	486.4592	482.4877	475.3496	479.1008	486.1255	483.2358
216	469	480	476.8571	476.1021	476.3971	476.4900	476.2881	476.2667	476.2075	476.4375	477.0332	477.0721	476.2832	476.6663	476.7883	476.6409
228	391	424	449.8571	437.9008	440.6520	442.1197	438.8347	438.0838	437.3336	440.2548	447.5501	446.4426	438.7169	442.5666	445.8004	443.8206
240	313	365	378.1429	369.8692	372.4879	373.5052	371.2648	370.8987	370.3099	372.5994	378.4437	378.3195	371.1983	374.6877	376.4407	374.9383
252	264	310	313.8571	313.8939	315.3507	315.3496	315.4269	315.7438	315.6033	316.1437	317.7527	319.0723	315.4581	317.1379	316.1344	315.8692
264	228	270	266.8571	268.1892	269.2125	269.0521	269.4790	269.8482	269.8022	269.9753	270.6433	271.9613	269.5201	270.6266	269.3646	269.3390
276	191	230	229.4286	229.9141	230.9977	230.9408	231.1290	231.4162	231.3301	231.6597	232.6943	233.8130	231.1590	232.3826	231.4408	231.2999
288	167	209	195.2857	197.5687	198.2007	197.9232	198.6035	198.9954	199.0258	198.9020	198.7977	200.0488	198.6503	199.2515	197.8492	198.0130
300	167	193	179.0000	186.0306	185.7777	184.9178	186.9120	187.6466	187.9547	186.7487	183.9786	185.8559	187.0100	186.3221	183.4950	184.4065
312	164	185	173.5714	177.3930	177.3448	176.8776	177.9657	178.3842	178.5430	177.9209	176.5145	177.6152	178.0208	177.7500	176.1528	176.6317

Table 5 A comparison of calculated outflow with different methods for 1961

Time (h)	Inflow (m^3/s)	Outflow (m^3/s)	Routed outflow (m^3/s)													
			TM	LRSM	NPM	NMSM	BGA	GGA	GAGA	DE	SaDE	EPSDE	PSO	DMS-PSO	CLPSO	HPSO
0	261	228	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000	228.0000
12	389	300	288.1429	305.2611	296.1246	298.7340	303.4148	304.3856	305.5072	301.1426	290.1567	291.3061	303.5733	297.5008	293.2001	300.8365
24	462	382	384.4286	382.4436	379.9467	380.5804	379.9179	379.2440	379.3688	378.8948	377.2581	374.7768	379.8445	377.5307	379.8018	380.3428
36	505	444	451.4286	445.6433	445.2389	445.2765	443.5769	442.6976	442.5688	443.0815	443.9362	441.2420	443.4693	442.5581	445.7520	444.5719
48	525	490	493.2857	486.6125	487.3443	487.0721	485.1521	484.3269	484.0925	485.0141	486.9808	484.6643	485.0465	485.0123	488.0993	486.2574
60	543	513	520.1429	517.4681	517.3572	517.3546	516.5714	516.1768	516.1118	516.3698	516.8273	515.6320	516.5228	516.1661	517.6048	517.0287
72	556	528	538.1429	535.4206	535.5231	535.4637	534.6739	534.3063	534.2248	534.5467	535.1885	534.1142	534.6278	534.4475	535.8044	535.1317
84	567	543	551.1429	548.4016	548.5895	548.5074	547.7149	547.3581	547.2699	547.6174	548.3330	547.3070	547.6698	547.5601	548.8843	548.1729
96	577	553	563.0000	560.7514	560.8660	560.8091	560.1577	559.8588	559.7893	560.0634	560.6180	559.7503	560.1200	559.9965	561.1024	560.5348
108	583	564	571.8571	568.9010	569.3410	569.1899	568.3433	567.9960	567.8839	568.3241	569.2900	568.3403	568.2983	568.3814	569.6912	568.8288
120	587	573	578.7143	576.2413	576.6507	576.5134	575.8066	575.5226	575.4258	575.8054	576.6472	575.8800	575.7696	575.8741	576.9492	576.2113
132	595	581	585.2857	584.3573	584.2479	584.2657	583.9914	583.8433	583.8258	583.8958	583.9965	583.5348	583.9735	583.7895	584.3242	584.1527
144	597	588	591.5714	589.5817	589.9603	589.8370	589.2699	589.0492	588.9678	589.2868	590.0045	589.4197	589.2409	589.3668	590.2073	589.5937
156	597	594	594.4286	592.9220	593.2700	593.1605	592.7331	592.5757	592.5097	592.7681	593.3617	592.9595	592.7121	592.8596	593.4654	592.9762
168	589	592	593.8571	591.9398	592.7066	592.4819	591.9490	591.7997	591.6925	592.1107	593.1312	592.8331	591.9272	592.3897	592.9993	592.2468
180	556	584	580.4286	575.0933	577.6618	576.9220	575.4537	575.1071	574.7775	576.0609	579.2565	578.7140	575.4000	577.0558	578.5353	576.2672
192	538	566	558.8571	560.3601	560.7417	560.6591	561.1100	561.3821	561.3957	561.3387	561.3429	562.2254	561.1422	561.6136	560.6460	560.8417
204	516	550	539.7143	540.5097	541.2166	541.0400	541.2956	541.5193	541.4903	541.5993	542.0149	542.8153	541.3205	541.9985	541.2347	541.1357
216	486	520	517.1429	517.5275	518.6532	518.3614	518.5114	518.7433	518.6731	518.9419	519.7843	520.6890	518.5356	519.5286	518.7686	518.4064
228	505	504	501.1429	510.1953	507.3349	508.1958	510.7375	511.5620	512.0138	510.2490	506.0527	507.9620	510.8518	509.3133	506.0566	509.3042
240	477	483	496.7143	491.5939	493.9103	493.2394	491.8253	491.4692	491.1635	492.3543	495.2995	494.6807	491.7714	493.2344	494.7284	492.6113
252	429	461	465.0000	457.5135	461.1862	460.1301	458.0721	457.5965	457.1292	458.9489	463.4891	462.7731	457.9978	460.3794	462.4212	459.2111
264	379	420	423.8571	420.3691	423.1991	422.4125	421.4914	421.4464	421.1485	422.3047	425.3357	425.7408	421.4740	423.5331	423.9268	421.9825
276	320	368	373.8571	370.2837	373.4978	372.6090	371.6759	371.6795	371.3517	372.6230	375.9859	376.6088	371.6630	374.0397	374.2861	372.1679
288	263	318	317.4286	315.3806	318.1614	317.4049	316.9017	317.0520	316.7969	317.7841	320.4797	321.4564	316.9089	319.0676	318.7409	317.1504
300	220	271	266.4286	268.0289	269.4001	269.0581	269.5707	270.0127	269.9581	270.1633	270.9575	272.5342	269.6199	270.9407	269.4295	269.2509
312	182	234	223.7143	225.5295	226.6486	226.3756	226.9423	227.3748	227.3441	227.4567	228.0006	229.5010	226.9914	228.1193	226.6227	226.5950
324	167	193	192.5714	198.6226	197.8320	198.1122	199.8490	200.5772	200.7989	199.9282	198.0410	200.0577	199.9428	199.8661	197.1302	198.8513
336	152	178	170.1429	171.8417	172.0566	172.0202	172.5226	172.7959	172.8269	172.7028	172.5304	173.3850	172.5556	172.9048	171.9190	172.2270

Table 6 A comparison of calculated outflow with different methods for 1964

Time (h)	Inflow (m^3/s)	Outflow (m^3/s)	Routed outflow (m^3/s)													
			TM	LRSM	NPM	NMSM	BGA	GGA	GAGA	DE	SaDE	EPSDE	PSO	DMS-PSO	CLPSO	HPSO
0	259	224	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000	224.0000
12	306	265	262.4286	264.8837	263.3968	263.0933	263.7256	263.6818	263.9257	262.9802	260.4555	259.8548	263.7304	261.8754	261.8365	262.4138
24	351	295	307.1429	308.2398	306.8613	306.7240	306.9738	306.8009	306.9795	306.2893	304.3660	303.4561	306.9609	305.3076	305.7744	306.1646
36	400	332	349.0000	348.2936	346.8610	346.9439	346.6768	346.2898	346.4010	345.9755	344.6303	343.1319	346.6362	345.0219	346.2946	346.4730
48	431	389	389.4286	383.5294	382.8191	383.5385	381.8229	381.0084	380.8398	381.5059	382.8031	380.3993	381.7213	381.2334	384.2311	383.6370
60	431	410	419.0000	411.9694	412.2223	413.0822	411.0880	410.3534	410.0453	411.2513	414.0214	412.1441	410.9900	411.6779	414.5050	413.5935
72	455	422	431.8571	432.5873	431.8468	431.7559	431.9302	431.8532	431.9548	431.5618	430.4789	430.0323	431.9254	431.0294	431.2201	431.4470
84	476	449	451.5714	449.7621	449.2023	449.4222	448.8803	448.5571	448.5387	448.6148	448.6318	447.5874	448.8420	448.2975	449.4488	449.3299
96	499	477	474.8571	474.4060	473.7378	473.7914	473.6321	473.4380	473.4849	473.3057	472.7214	471.9861	473.6114	472.8653	473.5108	473.5791
108	546	555	506.1429	510.7741	509.2090	508.6393	509.8888	510.0724	510.4117	509.0929	505.7108	505.6911	509.9239	507.8561	506.9421	507.8015
120	643	580	576.2857	594.9509	591.3930	589.1035	594.2654	595.5960	596.6953	592.4113	581.8431	584.2342	594.4649	589.3061	583.7579	586.7121
132	645	606	625.5714	615.3793	615.6864	616.9328	614.0392	612.9614	612.5206	614.2466	618.1961	615.4212	613.8959	614.8244	618.9631	617.6527
144	629	610	629.2857	619.9270	620.7398	621.8855	619.2528	618.3771	617.9206	619.7045	623.9220	621.8508	619.1323	620.5977	624.0381	622.7366
156	613	607	619.0000	612.9892	613.6815	614.4178	612.7347	612.2088	611.8990	613.1086	616.0071	614.8298	612.6609	613.7987	615.8929	615.0255
168	629	617	615.8571	617.6831	617.1413	616.9168	617.4129	617.5015	617.6279	617.1362	615.8866	615.9465	617.4284	616.7000	616.2886	616.6135
180	565	605	607.2857	593.9558	596.3398	597.9745	594.2809	593.2970	592.5271	595.5278	602.9004	601.0519	594.1351	597.6382	601.7069	599.6261
192	556	564	573.8571	578.9609	579.0451	578.4216	579.8815	580.4723	580.6698	579.8948	578.1820	579.7854	579.9582	579.7681	577.5342	578.1463
204	501	543	542.5714	534.1815	536.1199	537.1498	534.8451	534.3199	533.7926	535.8455	540.9736	540.2035	534.7626	537.4731	539.7370	538.3463
216	477	517	506.1429	508.9280	509.5420	509.2030	510.0257	510.4701	510.5226	510.3126	510.0104	511.3956	510.0796	510.6316	509.0274	509.2563
228	451	497	481.0000	483.0966	483.7948	483.5402	484.1704	484.5556	484.5720	484.5025	484.5379	485.7927	484.2159	484.9040	483.5363	483.6649
240	425	472	456.7143	459.8153	460.4774	460.0999	461.0149	461.5051	461.5656	461.3237	460.9634	462.4864	461.0744	461.6643	459.8928	460.1515
252	400	437	431.2857	434.7310	435.3509	434.9313	435.9426	436.4624	436.5408	436.2288	435.6997	437.2879	436.0063	436.5281	434.6383	434.9472
264	395	413	409.1429	414.4520	414.4098	413.7608	415.2736	415.8603	416.0784	415.2235	413.2973	414.8486	415.3507	415.0030	412.7673	413.4280
276	387	403	397.8571	399.4551	399.6357	399.4408	399.9050	400.1231	400.1699	399.9850	399.6205	400.2610	399.9322	400.0507	399.2468	399.4099
288	346	387	379.8571	375.2829	376.6674	377.2297	375.9882	375.7722	375.4528	376.6949	379.8557	379.7302	375.9501	377.8063	378.8183	377.9994
300	260	343	333.1429	324.7933	327.6527	328.6798	326.4288	326.1059	325.4905	327.8823	334.0217	334.0910	326.3663	330.1379	331.7601	330.2038
312	294	279	293.4286	313.3368	311.6088	309.1717	314.7721	316.6351	317.6062	313.8118	304.8412	309.2482	315.0283	311.9123	304.5931	307.3614
324	242	255	274.8571	263.1478	265.1164	266.5520	263.3017	262.4104	261.7464	264.3352	270.6718	268.9352	263.1710	266.1033	269.7625	267.9579
336	226	242	241.1429	240.4887	240.9883	241.0695	240.9057	240.9396	240.8645	241.1553	241.9432	242.1962	240.9066	241.5203	241.4605	241.2876
348	222	235	229.4286	231.3993	231.4475	231.2067	231.7711	232.0026	232.0774	231.7840	231.1400	231.7732	231.8011	231.7457	230.8726	231.1060
360	219	232	224.8571	226.5026	226.5327	226.3316	226.8025	226.9936	227.0570	226.8082	226.2593	226.7788	226.8273	226.7693	226.0472	226.2440
372	216	228	221.8571	223.5026	223.5327	223.3316	223.8025	223.9936	224.0570	223.8082	223.2593	223.7788	223.8273	223.7693	223.0472	223.2440

Table 7 Results of the Wilcoxon signed ranks test

Comparison	R^+	R^-	p value
HPSO versus CLPSO	0	465	9.31323E−10
HPSO versus DMS-PSO	450	15	1.16415E−07
HPSO versus PSO	459	6	1.21072E−08
HPSO versus SaDE	234	231	0.492127506
HPSO versus EPSDE	399	66	0.000127527
HPSO versus DE	444	21	3.13856E−07
HPSO versus GAGA	387	78	0.000411564
HPSO versus GGA	450	15	1.08033E−07
HPSO versus BGA	420	45	3.72529E−09
HPSO versus HMSM	462	3	1.45854E−05
HPSO versus NPM	459	6	3.72529E−09
HPSO versus LRSM	452	13	1.30385E−08
HPSO versus TM	465	0	9.31323E−10

Step 2 recover the positive and negative number after the numbered rank, gain positive rank sum R^+ and negative rank sum R^-, and select the smaller one from R^+ and R^- as the statistic T value of the Wilcoxon signed ranks test.

Step 3 calculate p value and draw conclusions.

In this experiment, each algorithm independently runs 30 times. HPSO algorithm is compared with other 13 algorithms, respectively, to test their performance differences.

In this study, R^+ and R^- are defined as follows:

$$R^+ = \sum_{d_i > 0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{13}$$

$$R^- = \sum_{d_i < 0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{14}$$

Here, R^+ indicates that the value of HPSO is better than that of the other algorithm, whereas R^- refers to the opposite scenario; and $d_i = 0$ indicates that two algorithms have the same test results.

P value is the probability of the observation results of the sample and the more extreme results when the null hypothesis is true. If the p value is extremely small, it means that the probability of the conditions when the null hypothesis is true is very low. If such conditions take place, we have reason to refuse the null hypothesis according to the small probability principle. The smaller p value, we have the more adequate reason to refuse the null hypothesis. In conclusion, the smaller p value, the more significant the results are.

In Table 7, while comparing HPSO with CLPSO, we set the null hypothesis H0 as HPSO better than CLPSO, and the alternative hypothesis H1 as HPSO not better than CLPSO. When comparing HPSO with other algorithms, respectively, the hypothesis is opposite, namely the null hypothesis H0 indicates that HPSO is not better than the other algorithm, whereas the alternative hypothesis H1 indicates that HPSO outperforms the other algorithm. Therefore, one-side test should be used. During the analysis with SPSS software, the significance value of the accuracy (one-side) is selected, with 0.05 chosen as the significance level in this study. According to Table 7, by comparison of HPSO and CLPSO, p value is far less than 0.05, so we reject H0 and accept H1 that CLPSO outperforms HPSO. Comparing HPSO with SaDE, we accept H0, with the p value higher than 0.05, suggesting that HPSO is not better than SaDE. While comparing HPSO with the other 11 algorithms, the p value is found far less than 0.05 all the time, so we accept H1 that HPSO is better than the other 11 algorithms, and the differences are statistically significant. HPSO is thus proven very effective.

5 Conclusion

In our paper, we have developed a new hybrid PSO (HPSO) heuristic algorithm so as to estimate the parameters of the Muskingum model. HPSO has better possessing capabilities than the GAs and is much easier to implement. We have conducted intensive experiments to compare the key performance of our presented algorithm with other superior estimation methods. The HPSO has the advantage that it does not require assumptions of the initial values of the model parameters compared with conventional methods. Furthermore, the results demonstrate that HPSO can achieve a higher degree of accuracy and faster convergent speed to estimate the Muskingum model parameters, which leads to accurate predictions of outflow and guarantees the effectiveness of outflow forecasting. It is worth to mention that no derivative is required by the HPSO method; moreover, HPSO will produce a better solution via making full use of advantages of HPSO and NMSM. In a word, HPSO is an effective and feasible parameter estimation method for Muskingum model, and it can be widely applied in the field of hydrology and hydraulics.

Acknowledgments This paper is partially funded by the Key Program of National Natural Science Foundation of China (Grant No. 61133005), and the National Natural Science Foundation of China (Grant Nos. 90715029, 61070057, 60603053, 61103047), and the Ph.D. Programs Foundation of Ministry of Education of China (20100161110019). Meanwhile, the paper was supported by the Research Foundation of Education Bureau of Hunan Province (No.13C333), the Project and the Science and Technology Research Foundation of Hunan Province (Grant No. 2014GK3043).

References

1. Mohan S (1997) Parameter estimation of non-linear Muskingum models using genetic algorithm. J Hydrol Eng 123(2):137–142
2. Yoon J, Padmanabhan G (1993) parameter estimation of linear and nonlinear muskingum models. J Water Resour Plan Manag 119(5):600–610
3. Geem ZW (2006) Parameter estimation for the nonlinear Muskingum model using the NMSM technique. J Irrig Drain Eng 132(5):474–478
4. Chen J, Yang X (2007) Optimal parameter estimation for Muskingum model based on Gray-encoded accelerating genetic algorithm. Commun Nonlinear Sci Numer Simul 12(5):849–858
5. Chu HJ, Chang LC (2009) Applying particle swarm optimization to parameter estimation of the nonlinear Muskingum model. J Hydrol Eng 14(9):1024–1027
6. Luo J, Xie J (2010) Parameter estimation for nonlinear Muskingum model based on immune clonal selection algorithm. J Hydrol Eng 15(10):844–851
7. Barati R (2011) Parameter estimation of nonlinear Muskingum models using Nelder–Mead simplex algorithm. J Hydrol Eng 16(11):946–954
8. Xu DM, Qiu L, Chen SY (2011) Estimation of nonlinear Muskingum model parameter using differential evolution. J Hydrol Eng 17(2):348–353
9. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4. Piscataway, NJ: IEEE Press, pp 1942–1948.
10. Chen D, Zhao C, Zhang H (2011) An improved cooperative particle swarm optimization and its application. Neural Comput Appl 20(2):171–182
11. Ghosh S, Das S, Kundu D, Suresh K, Panigrahi B, Cui Z (2012) An inertia-adaptive particle swarm system with particle mobility factor for improved global optimization. Neural Comput Appl 21(2):237–250
12. Wang H, Zhao G, Li N (2012) Training support vector data descriptors using converging linear particle swarm optimization. Neural Comput Appl 21(6):1099–1105
13. Jia L, Cheng D, Chiu MS (2012) Pareto-optimal solutions based multi-objective particle swarm optimization control for batch processes. Neural Comput Appl 21(6):1107–1116
14. Lee WP, Hsiao YT (2012) Inferring gene regulatory networks using a hybrid ga-pso approach with numerical constraints and network decomposition. Inf Sci 188:80–99
15. Castillo O, Martínez-Marroquín R, Melin P, Valdez F, Soria J (2012) Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. Inf Sci 192:19–38
16. Altun AA, Sahman MA (2013) Cost optimization of mixed feeds with the particle swarm optimization method. Neural Comput Appl 22(2):383–390
17. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295
18. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer. In Swarm intelligence symposium, 2005. SIS 2005. Proceedings 2005 IEEE (pp. 124–129).
19. Liang JJ, Suganthan PN, Chan CC, Huang VL (2006) Wavelength detection in FBG sensor network using tree search DMS-PSO. IEEE Photonics Technol Lett 18(12):1305–1307
20. Zhao SZ, Suganthan PN, Das S (2010) Dynamic multi-swarm particle swarm optimizer with sub-regional harmony search. 2010 IEEE congress on in evolutionary computation (CEC) (pp. 1–8).
21. Mallipeddi R, Suganthan PN (2011) Ensemble differential evolution algorithm for CEC2011 problems. 2011 IEEE Congress on evolutionary computation (CEC). IEEE 2011:1557–1564
22. Derrac J, Garcia S, Hui S, Herrera F, Suganthan P N (2013) Statistical analysis of convergence performance throughout the evolutionary search: a case study with SaDE-MMTS and Sa-EPSDE-MMTS. In Differential evolution (SDE), 2013 IEEE symposium on (pp. 151–156)
23. Huang VL, Qin AK, Suganthan PN (2006) Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In Proceedings of IEEE congress on evolutionary computation (pp. 17–24).
24. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
25. Yang X, Yang Z, Lu G, Li J (2005) A gray-encoded, hybrid-accelerated, genetic algorithm for global optimizations in dynamical systems. Commun Nonlinear Sci Numer Simul 10(4):355–363
26. Achiche S, Baron L, Balazinski M (2004) Real/binary-like coded versus binary coded genetic algorithms to automatically generate fuzzy knowledge bases: a comparative study. Eng Appl Artif Intell 17(4):313–325
27. Zhu Z, Zhou J, Zhen J, Shi YH (2011) DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm. IEEE Trans Evol Comput 15(5):643–658
28. Zhan ZH, Zhang J, Li Y, Shi YH (2011) Orthogonal learning particle swarm optimization. IEEE Trans Evol Comput 15(6):832–847
29. Chen WN, Zhang J, Lin Y, Chen N, Zhan ZH, Chung HH, Li Y, Shi YH (2013) Particle swarm optimization with an aging leader and challengers. IEEE Trans Evol Comput 17(2):241–258
30. Li X, Yao X (2012) Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans Evol Comput 16(2):210–224
31. Qu B, Suganthan P, Das S (2013) A distance-based locally informed particle swarm model for multimodal optimization. IEEE Trans Evol Comput 17(3):387–402
32. Ho S, Yang S, Ni G, Huang J (2013) A quantum-based particle swarm optimization algorithm applied to inverse problems. IEEE Trans Magn 49(5):2069–2072
33. Ouyang A, Tang Z, Zhou X, Xu Y, Pan G, Li K (2014) Parallel hybrid PSO with CUDA for lD heat conduction equation. Comput Fluids doi:10.1016/j.compfluid.2014.05.020
34. Ouyang A, Tang Z, Li K, Sallam A, Edwin S (2014) Estimating parameters of Muskingum model using an adaptive hybrid PSO algorithm. Int J Pattern Recogn Artif Intell 28(1):1–29
35. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20(1):89–99
36. Liu B, Wang L, Liu Y, Qian B, Jin Y-H (2010) An effective hybrid particle swarm optimization for batch scheduling of polypropylene processes. Comput Chem Eng 34(4):518–528
37. Ren X, Hao R, Sun Z, Shi B (2010) Quantum behaved particle swarm optimization algorithm based on simplex method. Microelectron Comput 27(1):154–157
38. Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7(4):308–313
39. Chen J, Ren Z, Fan X (2006) A hybrid optimized algorithm based on improved simplex method and particle swarm optimization. Proceeding fo the 25th Chinese control conference, 7–11 August, 2006, Harbin, Heilongjiang.
40. Zhao RJ (1992) The xinanjiang model applied in china. J Hydrol 135(1–4):371–381

41. Das A (2004) Parameter estimation for Muskingum models. J Irrig Drain Eng 130(2):140–147
42. Jin J, Ding J (2000) Genetic algorithm and its applications for water science. Sichuan University Press, Sichuan
43. Abdul-Rahman OA, Munetomo M, Akama K (2013) An adaptive parameter binary-real coded genetic algorithm for constraint optimization problems. Performance analysis and estimation of optimal control parameters. Inf Sci 233:54–86
44. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18