A Multi-Core CPU and Many-Core GPU Based Fast Parallel Shuffled Complex Evolution Global Optimization Approach

Guangyuan Kan, Tianjie Lei, Ke Liang, Jiren Li, Liuqian Ding, Xiaoyan He, Haijun Yu, Dawei Zhang, Depeng Zuo, Zhenxin Bao, Mark Amo-Boateng, Youbing Hu, and Mengjie Zhang

Abstract—In the field of hydrological modelling, the global and automatic parameter calibration has been a hot issue for many years. Among automatic parameter optimization algorithms, the shuffled complex evolution developed at the University of Arizona (SCE-UA) is the most successful method for stably and robustly locating the global "best" parameter values. Ever since the invention of the SCE-UA, the profession suddenly has a consistent way to calibrate watershed models. However, the computational efficiency of the SCE-UA significantly deteriorates when coping with big data and complex models. For the purpose of solving the efficiency problem, the recently emerging heterogeneous parallel computing (parallel computing by using the multi-core CPU and many-core GPU) was applied in the parallelization and acceleration of the SCE-UA. The original serial and proposed parallel SCE-UA were compared to test the performance based on the Griewank benchmark function. The comparison results indicated that the parallel SCE-UA converged much faster than the serial version and its optimization accuracy was the same as the serial version. It has a promising application prospect in the field of fast hydrological model parameter optimization.

Index Terms—parameter optimization, SCE-UA, multi-core CPU, many-core GPU, parallel computing

1 Introduction

A brief introduction. The difficulties involved in calibrating conceptual watershed hydrological models have, in the past, been partly attributable to the lack of robust optimization tools. After the global optimization method known as the SCE-UA (shuffled complex evolution method developed at the University of Arizona) has been proposed, it has shown promise as an effective and robust optimization technique for calibrating watershed hydrological models for more than 20 years [8]. Although SCE-UA has been widely applied in various scientific researches and engineering practices, however, the poor computational efficiency for complex problems limited its further application. In the following paragraphs, we gave a brief review of the development of the SCE-UA algorithm, and analyzed the pros and cons of it. Other optimization algorithms, such as genetic algorithm and particle swarm optimization, and the recently emerging heterogeneous parallel computing based acceleration technique were also reviewed. At last, we discussed the possibility of accelerating the SCE-UA by using the heterogeneous computing technology and proposed a heterogeneous computing based parallel SCE-UA method.

The development of SCE-UA algorithm. The SCE-UA algorithm was originally proposed for solving the conundrums encountered in the global, effective, and robust parameter calibration of the hydrological model [5], [6]. The SCE-UA is based on a synthesis of four concepts that have been proved successful for global optimization: (a) combination of probabilistic [9] and deterministic [14] approaches; (b) clustering; (c) systematic evolution of a complex of points spanning the space, in the direction of global improvement; and (d) competitive evolution [7]. It has been successfully applied in hydrological model calibration [6], [7], [8], [11], [12], [13] and other global optimization problems ([4], [29]; Yang et al., 2015). In addition to research and engineering applications, the SCE-UA was continuously improved by many researchers. Yapo et al. [27] proposed a multi-objective MOCOM-UA algorithm to spread the functionality of the successful SCE-UA single-objective algorithm. Vrugt et al. (Vrugt et al., 2003) replaced the Nelder-Mead downhill simplex search of SCE-UA by a Markov chain Monte Carlo Metropolis Hastings algorithm to develop the SCEM-UA

- *Guangyuan Kan, Tianjie Lei, Jiren Li, Liuqian Ding, Xiaoyan He, Haijun Yu, and Dawei Zhang, are with the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Research Center on Flood & Drought Disaster Reduction of the Ministry of Water Resources, China Institute of Water Resources and Hydropower Research, Beijing 100038, P. R. China. E-mail: {kanguangyuan, hxyiwhr, zhangdaweiscip}@126.com, {leitj, dinglq}@iwhr.com, ljrrsc@163.com, leivi@gmx.com.*
- *Ke Liang is with the College of Hydrology and Water Resources, Hohai University, Nanjing 210098, P. R. China. E-mail: 469524865@qq.com.*
- *Depeng Zuo is with the College of Water Sciences, Beijing Normal University, Beijing 100875, P. R. China. E-mail: dpzuo@bnu.edu.cn.*
- *Zhenxin Bao is with the Nanjing Hydraulic Research Institute, Nanjing, 210029, P. R. China. E-mail: zxbao@nhri.cn.*
- *Mark Amo-Boateng is with the University of Energy and Natural Resources, Sunyani 00233, Ghana. E-mail: yaw_mark@yahoo.com.*
- *Youbing Hu is with the Hydrologic Bureau (Information Center) of the Huaihe River Commission, Bengbu 233001, P. R. China. E-mail: 598564865@qq.com.*
- *Mengjie Zhang is with the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Department of Water Resources (DWR), China Institute of Water Resources and Hydropower Research, Beijing 100038, P. R. China. E-mail: zhangmengjieae@126.com.*

Manuscript received 4 Jan. 2016; revised 25 May 2016; accepted 26 May 2016. Date of publication 14 June 2016; date of current version 18 Jan. 2017. Recommended for acceptance by A. Dubey.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2016.2575822

algorithm. The SCEM-UA can obtain the post probability distribution of parameters for optimization and uncertainty assessment of hydrological model parameters. Cooper et al. [3] considered hydrologic process-based parameter constraints into the SCE-UA and suggested a constrained SCE-based calibration procedure for the hydrological model parameter optimization. Chu et al. [1], [2] developed a SP-UCI method to improve the performance of the SCE-UA on high-dimensional optimization problems. The SP-UCI is developed to treat complexity caused by high dimensionalities. It features a slop-based searching kernel and a scheme of maintaining the particle population's capability of searching over the full search space.

The pros and cons of SCE-UA algorithm and the motivation to parallelize it. A large number of benchmarking and real world applications have proved that the SCE-UA method is a powerful and effective global optimization algorithm. Compared with other optimization algorithms, such as genetic algorithm (GA) and particle swarm optimization (PSO) [25], it is able to consistently locate the global optimum of the problem and appears to be capable of robustly and effectively solving a various kinds of optimization problems. However, in real world applications, one of the disadvantages of the SCE-UA algorithm is that the computational efficiency is not satisfactory for complex optimization problems. As for the hydrological model parameter calibration problem, it consumes much computation time when the observed hydrological dataset is very large. This is because the computation load of the objective function increases significantly with the increasing of the hydrological dataset. When coping with high dimensional optimization problems, the number of complexes should be increased to deal with the large number of local minima and ensure a higher probability to converge into the global optimum. Large number of complexes require more objective function evaluations to converge and consume much more computation time. Many researches indicated that the best way to solve the computational efficiency problem is the using of the parallel computing technology. The SCE-UA algorithm is an inherently parallel process and is highly suited to be accelerated by parallel computing technology. Therefore, the parallelization of the SCE-UA algorithm is very necessary.

The application of parallel computing in enhancing optimization algorithms. Many previous literatures have successfully applied the parallel computing in enhancing optimization algorithm efficiency. Humphrey et al. [10] applied the cloud computing into the calibration of the famous SWAT hydrological model. Their cloud-based system calibrated the watershed model in 43.32 minutes using 16 cloud cores (15.78x speedup), 11.76 minutes using 64 cloud cores (58.13x speedup), and 5.03 minutes using 256 cloud cores (135.89x speedup). They believed that such speed-ups offer the potential toward real-time interactive model creation with continuous calibration, ushering in a new paradigm for watershed modeling. Zhang et al. [28] discussed genetic algorithms from an architectural perspective, offering a general analysis of performance of GAs on multi-core CPUs and on many-core GPUs. Based on the widely used Parallel GA (PGA) schemes, they proposed the best one for each architecture. Umbarkar et al. [22] applied the multithreaded parallel dual population genetic algorithm (MPDPGA) for unconstrained function optimizations on multi-core system to solve the problem of premature convergence and helped in early convergence of the algorithm. Ouyang et al. [15] designed a parallel hybrid algorithm by hybridizing the PSO and CGM with CUDA and speeded up the computing of general linear equations. Although the parallel computing has been widely used in the acceleration of classical optimization algorithms such as GA and PSO, the report on the parallel computing based SCE-UA family algorithm is rare. It can only be found in the Web of Science that Vrugt et al. (Vrugt et al., 2006) proposed a parallel implementation of the SCEM-UA for solving the complex optimization problem. It utilized the computational power of a distributed CPU computer system (however with no GPU).

Objective of this research. For the purpose of improving the computational efficiency of the SCE-UA algorithm, a multi-core CPU and many-core GPU based fast parallel SCE-UA global optimization approach was proposed in this paper. The recently emerging heterogeneous computing (parallel computing by using the multi-core CPU and many-core GPU) was applied in the parallelization and acceleration of the SCE-UA. The original serial and proposed parallel SCE-UA were compared to test the performance based on the Griewank benchmark function. The comparison results indicated that the parallel SCE-UA converged much faster than the serial version and its optimization accuracy was the same as the serial version. It has a promising application prospect in the field of fast hydrological model parameter optimization.

2 Methodology

2.1 Principle of the Original Serial SCE-UA

2.1.1 The Main Loop

The SCE-UA strategy combines the strengths of the controlled random search algorithms [16], [17], [18] with the concept of competitive evolution and complex shuffling. The SCE strategy is presented below and is illustrated in Fig. 1.

Step 0. Initialize. Select $p \geq 1$ and $m \geq n+1$, where $p =$ number of complexes and $m =$ number of points in each complex. Compute the sample size $s = p \times m$.

Step 1. Generate sample. Sample s points $x_1, \ldots, x_s$ in the feasible space $\Omega \subset \mathrm{R}^n$. Compute the function value f_i at each point x_i. In the absence of prior information, use a uniform sampling distribution.

Step 2. Rank points. Sort the s points in order of increasing function value. Store them in an array $D = \{x_i, f_i, i = 1, \ldots, s\}$, so that $i = 1$ represents the point with the smallest function value.

Step 3. Partition into complexes. Partition D into p complexes $A^1, \ldots, A^p$, each containing m points, such that:

$$A^k = \left\{ x_j^k, f_j^k \middle| x_j^k = x_{k+p(j-1)}, f_j^k = f_{k+p(j-1)}, j = 1, \ldots, m \right\}.$$

Step 4. Evolve each complex. Evolve each complex $A^k, k = 1, \ldots, p$ according to the competitive complex evolution (CCE) algorithm outlined separately in Section 2.1.2.

Step 5. Shuffle complexes. Replace $A^1, \ldots, A^p$ into D, such that $D = \{A^k, k = 1, \ldots, p\}$. Sort D in order of increasing function value.

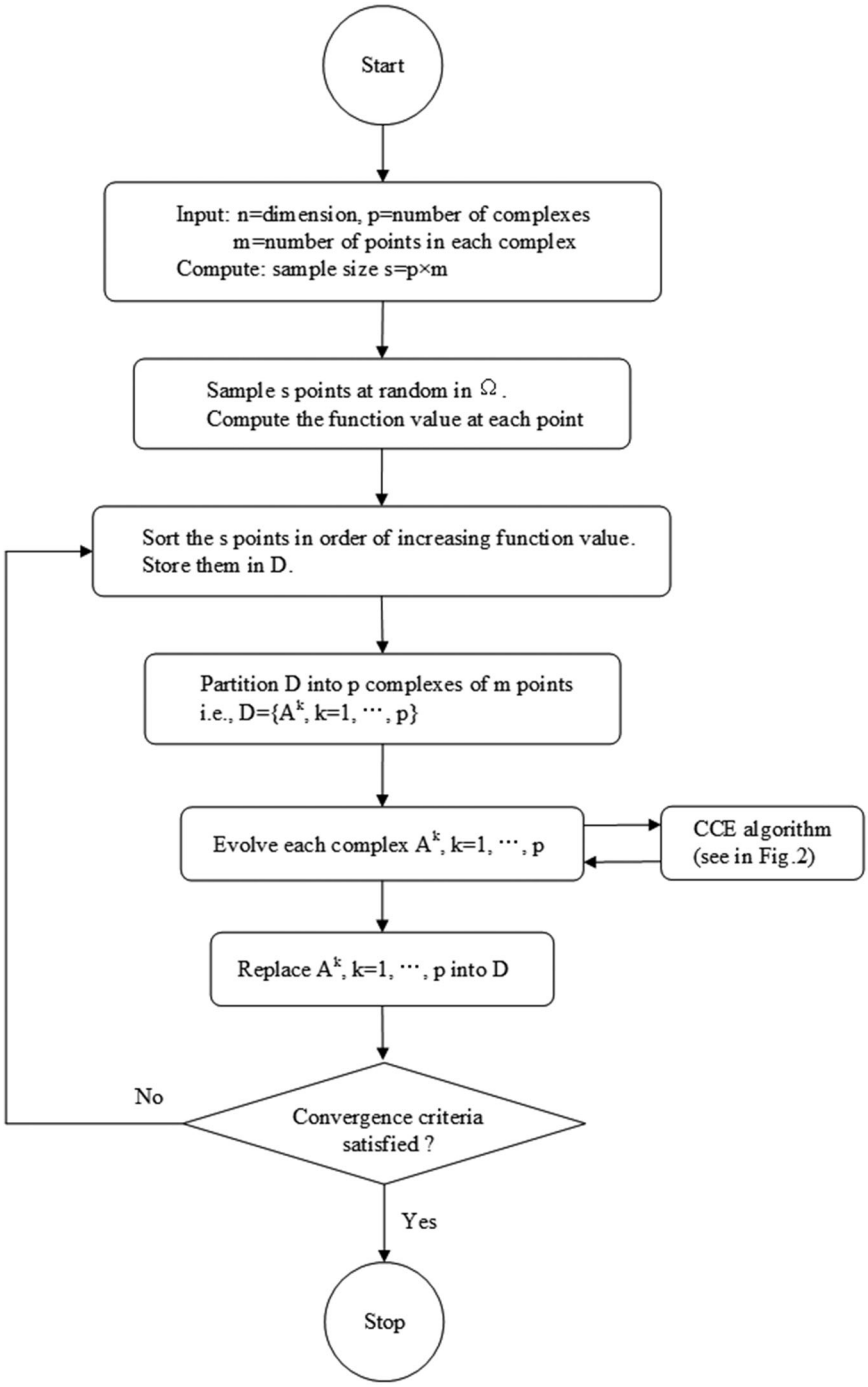


Fig. 1. Flow chart of the serial SCE-UA method.

Step 6. Check convergence. If the convergence criteria are satisfied, stop; otherwise, return to Step 3 [7].

The serial SCE-UA algorithm (named SCPU-SCE-UA) is implemented by using the PGI Visual Fortran 14.10 compiler.

2.1.2 The CCE Algorithm

The competitive complex evolution (CCE) algorithm required for the evolution of each complex in Step 4 of the SCE-UA method is presented below and is illustrated in Fig. 2.

Step 0. Initialize. Select q, α, and β, where $2 \leq q \leq m$, $\alpha \geq 1$, and $\beta \geq 1$.

Step 1. Assign weights. Assign a triangular probability distribution to A^k; i.e., $p_t = 2(m+1-t)/m(m+1), t = 1, \ldots, m$. The point x_1^k has the highest probability, $p_1 = 2/(m+1)$. The point x_m^k has the lowest probability, $p_m = 2/m(m+1)$.

Step 2. Select parents. Randomly choose q distinct points $u_1, \ldots, u_q$ from A^k according to the probability distribution specified above (the q points define a subcomplex). Store them in array $B = \{u_l, v_l, l = 1, \ldots, q\}$, where v_l is the function value associated with point u_l. Store in L the locations of A^k which are used to construct B.

Step 3. Generate offspring.

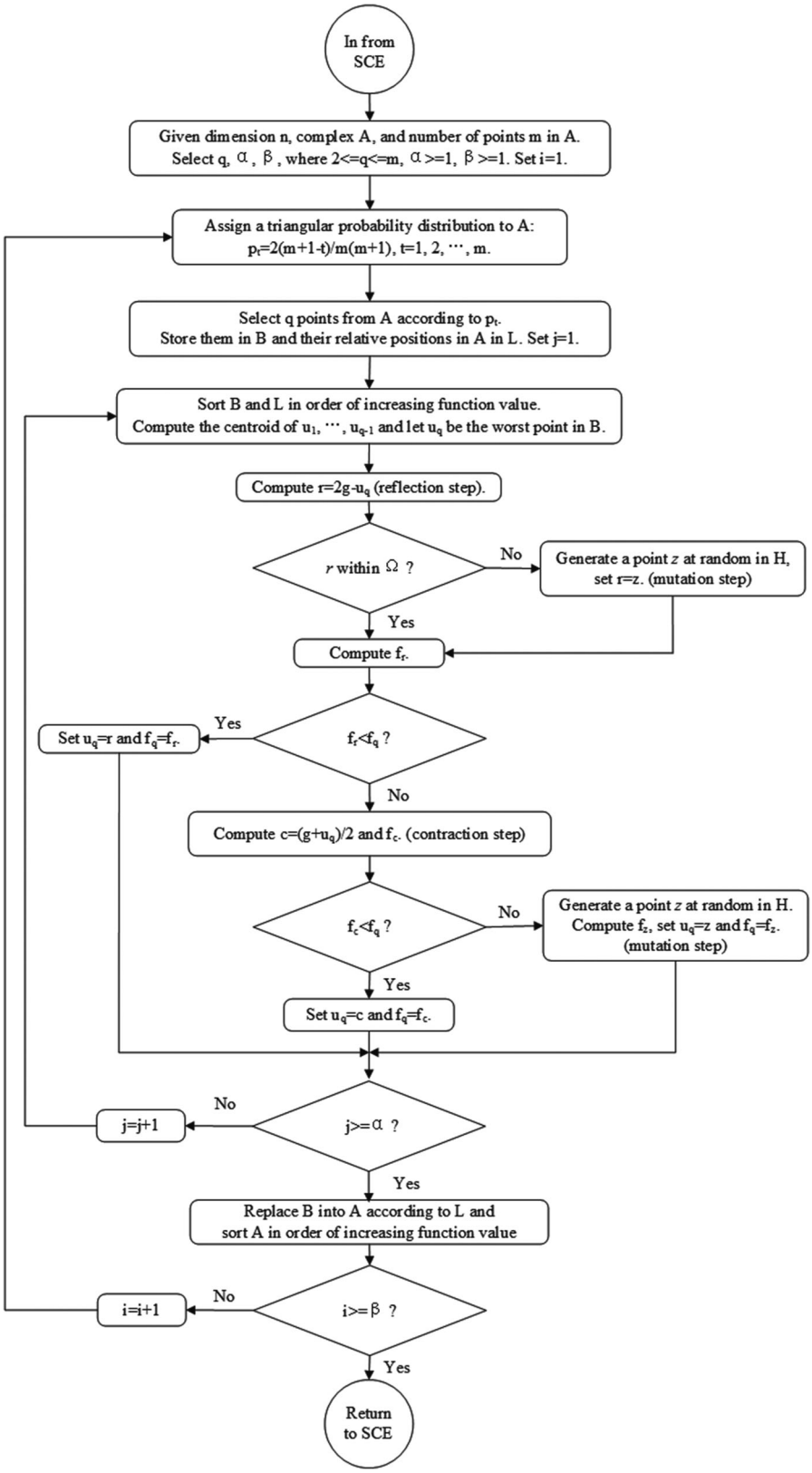


Fig. 2. Flow chart of the CCE algorithm.

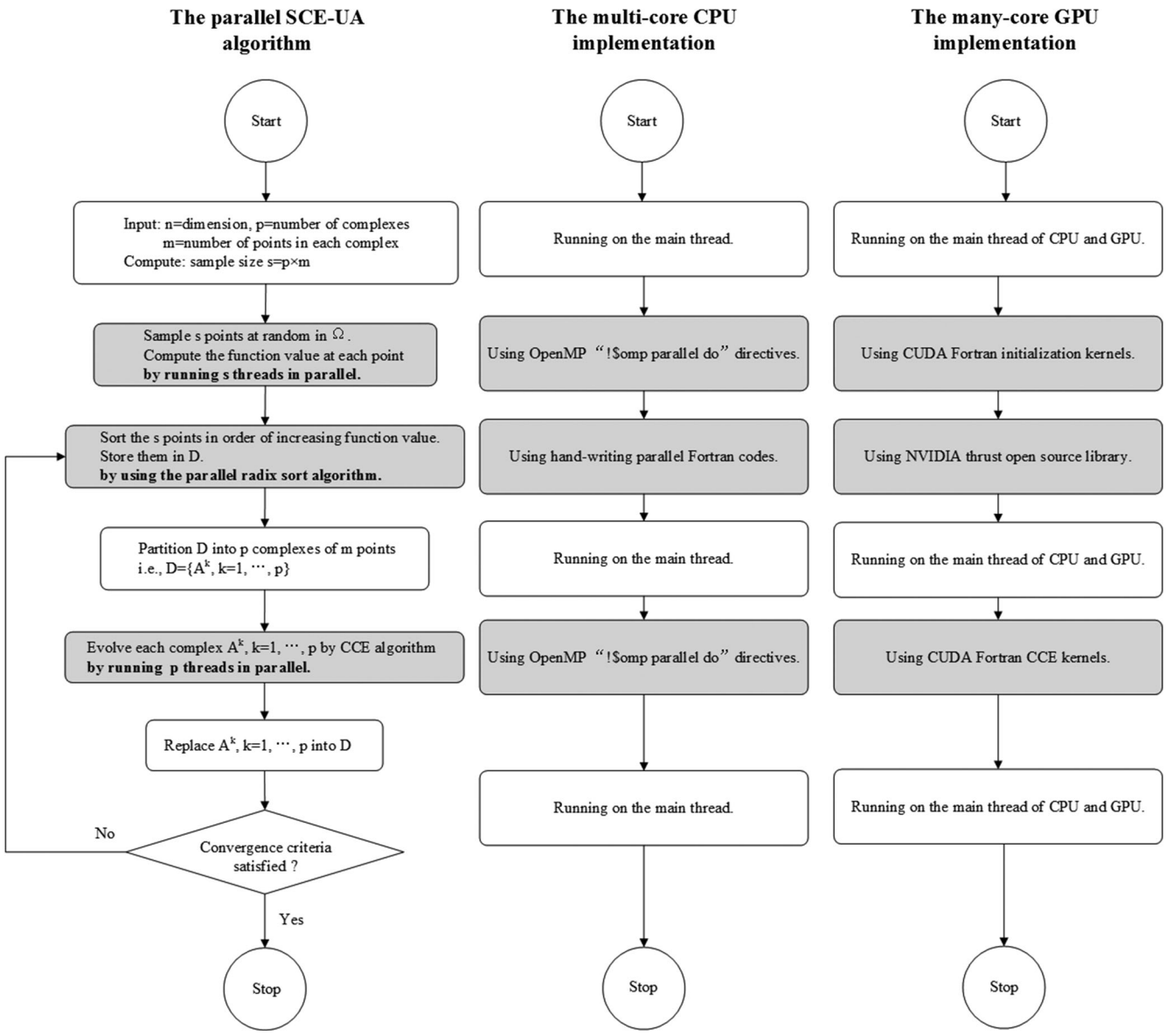


Fig. 3. Flow chart of the parallel SCE-UA method.

(a) Sort B and L so that the q points are arranged in order of increasing function value. Compute the centroid g using the expression:

$$g = \sum_{l=1}^{q-1} u_l \Big/ (q-1).$$

(b) Compute the new point $r = 2g - u_q$ (reflection step).

(c) If r is within Ω, compute the function value f_r, and go to Step (d); else, compute the smallest hypercube $H \subset \mathrm{R}^n$ that contains A^k, randomly generate a point z within H, compute f_z, set $r = z$, and set $f_r = f_z$ (mutation step).

(d) If $f_r < f_q$, replace u_q by r, go to Step (f); else, compute $c = (g + u_q)/2$ and f_c (contraction step).

(e) If $f_c < f_q$, replace u_q by c, go to Step (f); else, randomly generate a point z within H and compute f_z (mutation step). Replace u_q by z.

(f) Repeat Steps (a) through (e) α times, where $\alpha \geqslant 1$ is a user-specified parameter.

Step 4. Replace parents by offspring. Replace B into A^k using the original locations stored in L. Sort A^k in order of increasing function value.

Step 5. Iterate. Repeat Steps (1) through (4) β times, where $\beta \geq 1$ is a user-specified parameter which determines how many offspring should be generated (how far each complex should evolve) [7].

2.2 Principle of the Improved Parallel SCE-UA

Principle of parallel SCE-UA algorithm is stated as follows and illustrated in Fig. 3 (the parallel parts are colored in gray). The parallel SCE-UA is implemented on multi-core CPU and many-core GPU. The multi-core CPU version is named MCPU-SCE-UA. The many-core GPU version is named GPU-SCE-UA.

Step 0. Initialize. Select $p \geqslant 1$ and $m \geqslant n+1$, where $p =$ number of complexes and $m =$ number of points in each complex. Compute the sample size $s = p \times m$.

Step 1. Generate sample. Sample s points $x_1, \ldots, x_s$ in parallel in the feasible space $\Omega \subset \mathrm{R}^n$ by creating s threads. In

thread $i(i = 1, \ldots, s)$, generate one point x_i and compute its function value f_i. As for the random number generation, generate s random seeds for the s threads. In each thread, the random number is generated based on its corresponding random seed. For the purpose of generating random numbers with good statistical characteristic in parallel, use the Mersenne Twister random number generator instead of the widely used linear congruential random number generator [20], [21], [22], [23]. In the absence of prior information, use a uniform sampling distribution.

Step 2. Rank points. Sort the s points in order of increasing function value. Store them in an array $D = \{x_i, f_i, i = 1, \ldots, s\}$, so that $i = 1$ represents the point with the smallest function value. Here we use the parallel radix sort algorithm to implement the parallel sorting [19], [21]. For MCPU-SCE-UA, use the hand-writing Fortran codes with OpenMP supported. For GPU-SCE-UA, we call the radix sort API from the NVIDIA thrust open source library to implement the parallel sorting.

Step 3. Partition into complexes. Partition D into p complexes $A^1, \ldots, A^p$, each containing m points, such that:

$$A^k = \left\{ x_j^k, f_j^k \middle| x_j^k = x_{k+p(j-1)}, f_j^k = f_{k+p(j-1)}, j = 1, \ldots, m \right\}.$$

Step 4. Evolve complexes in parallel. Evolve each complex $A^k, k = 1, \ldots, p$ according to the competitive complex evolution (CCE) algorithm outlined separately in Section 2.1.2. Here we create p threads, each thread contains a complex and implements a CCE optimization process. These p complexes are evolved in parallel.

Step 5. Shuffle complexes. Replace $A^1, \ldots, A^p$ into D, such that $D = \{A^k, k = 1, \ldots, p\}$. Sort D in order of increasing function value. The sorting process adopts the parallel radix sort algorithm described in Step 2.

Step 6. Check convergence. If the convergence criteria are satisfied, stop; otherwise, return to Step 3.

The MCPU-SCE-UA is implemented by using the PGI Visual Fortran 14.10 with OpenMP directives and clauses. The parallel computing regions are implemented by "for loops" decorated by the OpenMP "!$omp parallel do" directives. Variables are further decorated by the OpenMP "private", "shared", and "reduction" clauses to define their locations and accessing pattern. The GPU-SCE-UA is implemented by using the PGI CUDA Fortran 14.10. The parallel computing regions are implemented by CUDA Fortran codes run on GPU cores, which is usually called "CUDA kernels". The variables of the GPU version is designed to locate in various kinds of GPU memory hierarchies for the purpose of better memory accessing efficiency. For constant scalar variables, we locate them in the constant memory. For constant arrays, we locate them in the cached texture memory. For arrays which contain small number of elements and transfer frequently between CPU memory and GPU memory, we locate them in pinned memory. For arrays which contain large number of elements and change frequently, we locate them in the off-chip global memory.

2.3 Griewank Benchmark Function

The serial and parallel SCE-UA methods were tested based on the Griewank benchmark problems. The performances are compared and the sensitivity is analyzed based on the testing results. Because the calculation of the Griewank benchmark function is very fast and cannot mimic the heavy computational burden of the hydrological model simulation, we added four kinds of arithmetic operations after the calculation of the Griewank function for each objective function evaluation. These operations include 10^6 additions, 10^6 subtractions, 10^6 multiplications, and 10^6 divisions of a dummy temporary variable. For the GPU-SCE-UA, the time consumed by the memory transfer is considered into the total execution time to ensure a fair comparison between the CPU and GPU versions.

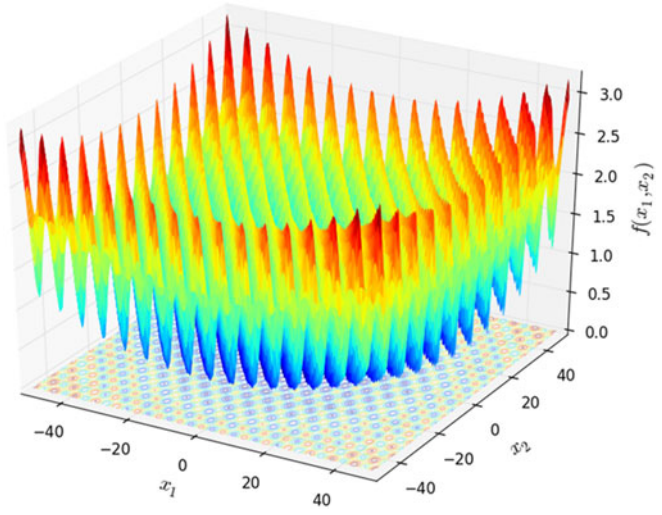


Fig. 4. The two dimensional Griewank function.

The Griewank benchmark function is a relatively hard problem for optimization algorithms. It has lots of local minima which confusing the algorithm and making the problem much harder. The Griewank function is as follows:

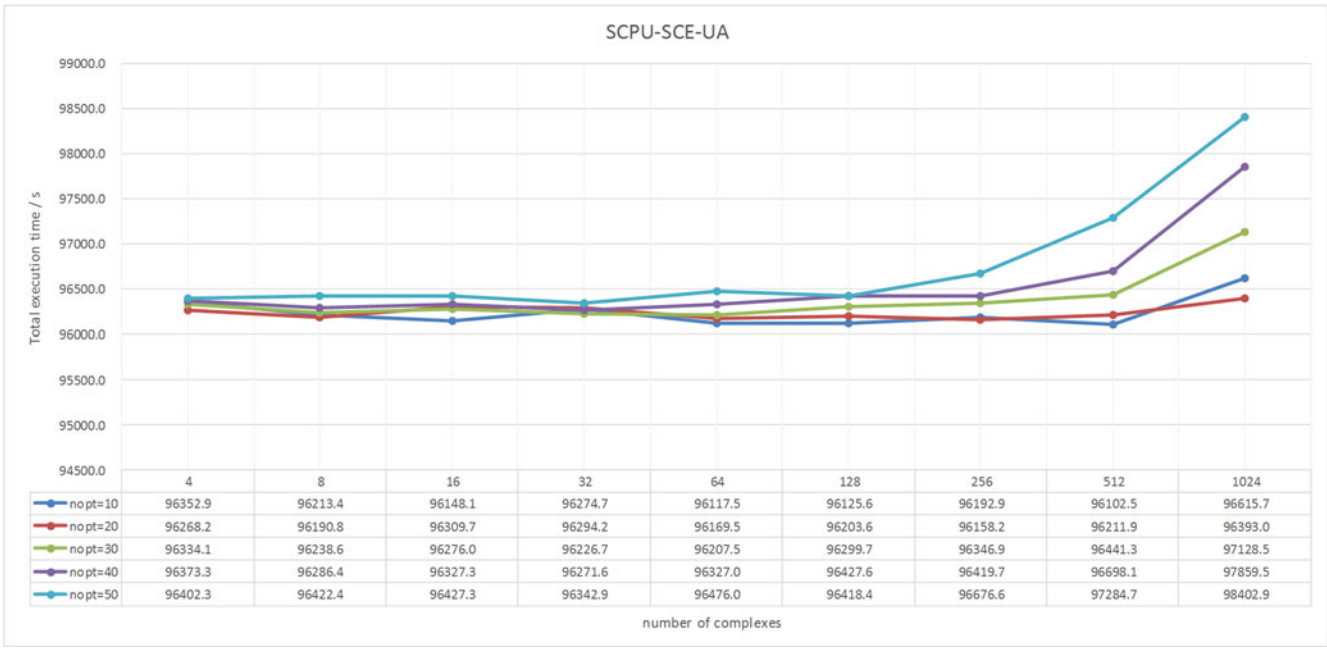
$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{d} - \prod_{i=1}^{n} \cos\left(x_i \Big/ \sqrt{i}\right) + 1$$

$$-600 \le x_i \le 600, i = 1, \ldots, n, \; d = 600$$

The global minimum is 0 and is at the origin [7]. As an example, the two dimensional Griewank benchmark function is demonstrated in Fig. 4.

2.4 Hardware Used in This Study

In this study, we used an Intel Xeon E5-2640v2 CPU and a NVIDIA Tesla K40c GPU. The Intel Xeon Processor E5-2640v2 is a 8 physical cores Ivy Bride high end workstation and server CPU with 20 MB LLC, 7.2 GT/s QPI, 2.0 GHz base clock, and 95 W TDP. It supports the Intel Hyper-Threading Technology which delivers two processing threads per physical core. Highly threaded applications can get more work done in parallel, completing tasks sooner. It also supports Intel Turbo Boost Technology which dynamically increases the processor's frequency as needed, by taking advantage of thermal and power headroom to give you a burst of speed when you need it, and increase energy efficiency when you don't. The NVIDIA Tesla K40 graphics processing unit (GPU) active accelerator is a PCI Express, dual-slot full height form factor computing module



	4	8	16	32	64	128	256	512	1024
nopt=10	96352.9	96213.4	96148.1	96274.7	96117.5	96125.6	96192.9	96102.5	96615.7
nopt=20	96268.2	96190.8	96309.7	96294.2	96169.5	96203.6	96158.2	96211.9	96393.0
nopt=30	96334.1	96238.6	96276.0	96226.7	96207.5	96299.7	96346.9	96441.3	97128.5
nopt=40	96373.3	96286.4	96327.3	96271.6	96327.0	96427.6	96419.7	96698.1	97859.5
nopt=50	96402.3	96422.4	96427.3	96342.9	96476.0	96418.4	96676.6	97284.7	98402.9

Fig. 5. Total execution time of the SCPU-SCE-UA.

comprised of a single GK110B GPU. The K40 active accelerator is designed for workstations and servers, offers a total of 12 GB of GDDR5 on-board memory, and supports PCI Express Gen 3. It processes 2880 GPU processor cores with a base clock of 745 MHz and a maximum boost clock of 875 MHz. In this study, the GPU clock is set to 875 MHz by using the NVIDIA-SMI command. The ECC memory option is off by using the NVIDIA control panel in this research.

3 Results and Discussion

3.1 The Number of Objective Function Evaluations and the Optimization Accuracy

In this research, for the purpose of comparing performance of real-world applications, both computation time (related to the number of objective function evaluations) and objective function values (related to the optimization accuracy) are concerned. For the purpose of a fair comparison of the total execution time and algorithm sensitivity, we should fix the number of objective function evaluations for each optimization computation. After numerical experiment, we choose 5×10^6 as the number of objective function evaluations for every optimization computation to ensure that all optimization process can converge to the same global optimum. Therefore, the testing is based on a fixed computation total load of 5×10^6 objective function evaluations. We will change the number of decision variables (i.e., the dimension of the problem) and the number of complexes (i.e., the parallelism of the algorithm) to test the performances of SCPU-SCE-UA, MCPU-SCE-UA, and GPU-SCE-UA methods.

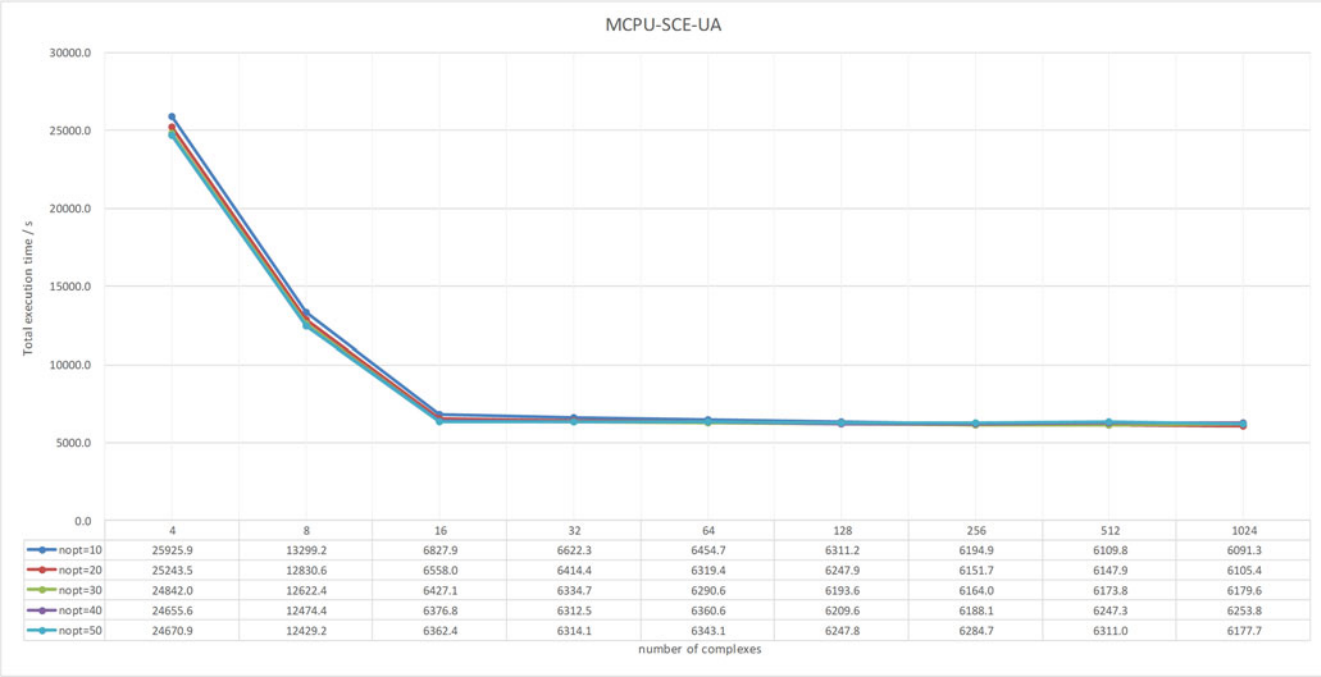
3.2 Total Execution Time Comparison

3.2.1 SCPU-SCE-UA

The total execution time of the SCPU-SCE-UA is demonstrated in Fig. 5. As we can see, with the increasing of the number of complexes or decision variables, the total execution time changes not too much. The execution time varies from 96102.5s to 98402.9s. The change degree is less than 2.5 percent. This is because that with a fixed number of objective function evaluations (5×10^6 in this research), the total computation task is fixed. The serial SCE-UA cannot make full use of the parallel computation capability of multi-core CPU. With the increasing of the number of complexes and decision variables, the total execution time changes very little.

3.2.2 MCPU-SCE-UA

The total execution time of the MCPU-SCE-UA is demonstrated in Fig. 6. The total execution time varies from 6091.3s to 25925.9s. As we can see, with the increasing of the number of complexes, the total execution time decreases. This is because that under the condition of the fixed number of objective function evaluations, for larger number of complexes, the multi-core CPU creates more threads run in parallel to further speed up the execution. However, when the number of complexes is larger than 16 (i.e. creating more than 16 threads), the total execution time do not significantly decrease furthermore. This is because that the CPU has only 16 cores (actually 8 physical cores, i.e. 16 logical cores supported by hyper-threading technology). It has reached the upper boundary of the speed up capability when creating 16 threads. Therefore, more threads will no longer speed up the execution any more. With the increasing of the number of decision variables, the total execution time changes not too much. This is because that the range of the test problem dimensionality is from 10 to 50. The upper boundary of the dimensionality is not large enough to reflect the effect of the dimension to the execution speed. We do not test very high dimension problem because when dimension becomes very large, the CPU memory is not enough (32GB in this research). High dimension problem usually causes the memory overflow runtime error.

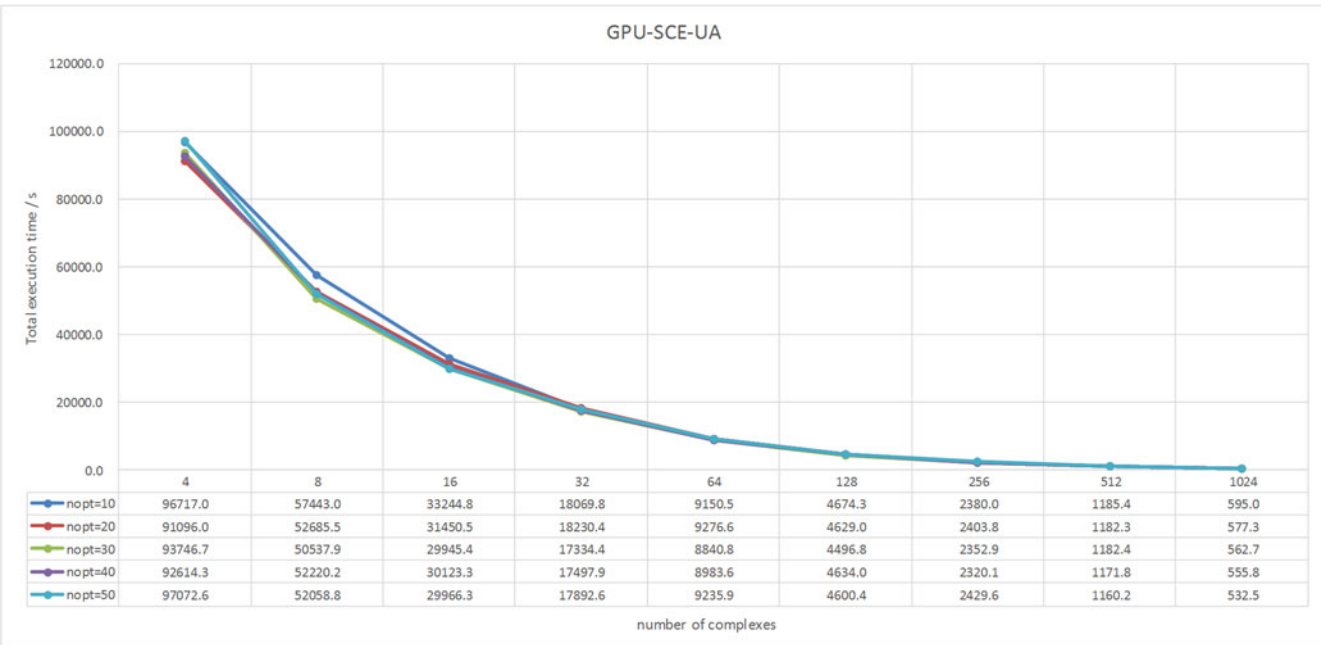


	4	8	16	32	64	128	256	512	1024
nopt=10	25925.9	13299.2	6827.9	6622.3	6454.7	6311.2	6194.9	6109.8	6091.3
nopt=20	25243.5	12830.6	6558.0	6414.4	6319.4	6247.9	6151.7	6147.9	6105.4
nopt=30	24842.0	12622.4	6427.1	6334.7	6290.6	6193.6	6164.0	6173.8	6179.6
nopt=40	24655.6	12474.4	6376.8	6312.5	6360.6	6209.6	6188.1	6247.3	6253.8
nopt=50	24670.9	12429.2	6362.4	6314.1	6343.1	6247.8	6284.7	6311.0	6177.7

Fig. 6. Total execution time of the MCPU-SCE-UA.

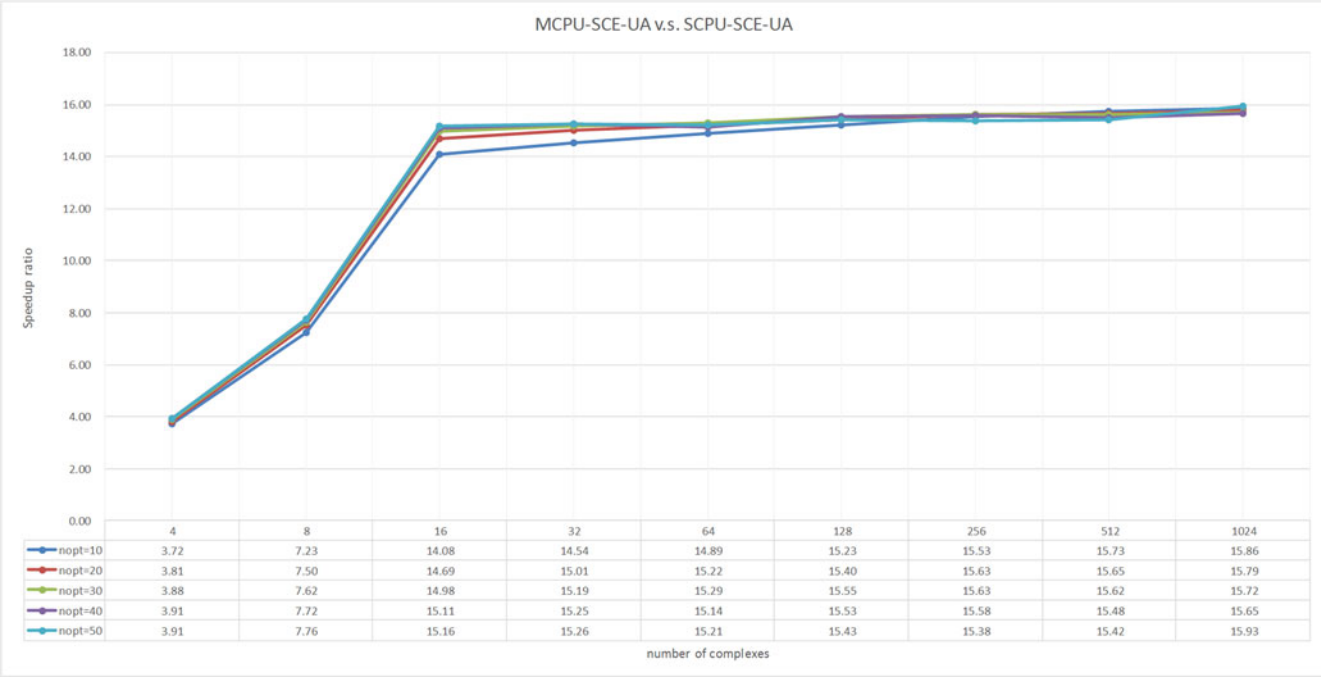
3.2.3 GPU-SCE-UA

The total execution time of the GPU-SCE-UA is demonstrated in Fig. 7. The total execution time varies from 532.5s to 97072.6s. As we can see, with the increasing of the number of complexes, the total execution time decreases significantly. This is because that under the condition of the fixed number of objective function evaluations, for larger number of complexes, the GPU creates more threads run in parallel to speed up the execution significantly. The many-core GPU can create much more threads run in parallel than the multi-core CPU. Therefore, for larger number of complexes, the GPU version runs much faster than the serial CPU and parallel CPU versions. With the increasing of the number of decision variables, the total execution time changes not too much. This is because that the range of the problem dimensionality is from 10 to 50. The upper boundary of the dimensionality is not large enough to reflect the effect of the dimension to the execution speed. We do not test very high dimension problem because when dimension becomes very large, the GPU memory is not enough (12GB in this research). High dimension problem usually causes the memory overflow runtime error.



	4	8	16	32	64	128	256	512	1024
nopt=10	96717.0	57443.0	33244.8	18069.8	9150.5	4674.3	2380.0	1185.4	595.0
nopt=20	91096.0	52685.5	31450.5	18230.4	9276.6	4629.0	2403.8	1182.3	577.3
nopt=30	93746.7	50537.9	29945.4	17334.4	8840.8	4496.8	2352.9	1182.4	562.7
nopt=40	92614.3	52220.2	30123.3	17497.9	8983.6	4634.0	2320.1	1171.8	555.8
nopt=50	97072.6	52058.8	29966.3	17892.6	9235.9	4600.4	2429.6	1160.2	532.5

Fig. 7. Total execution time of the GPU-SCE-UA.



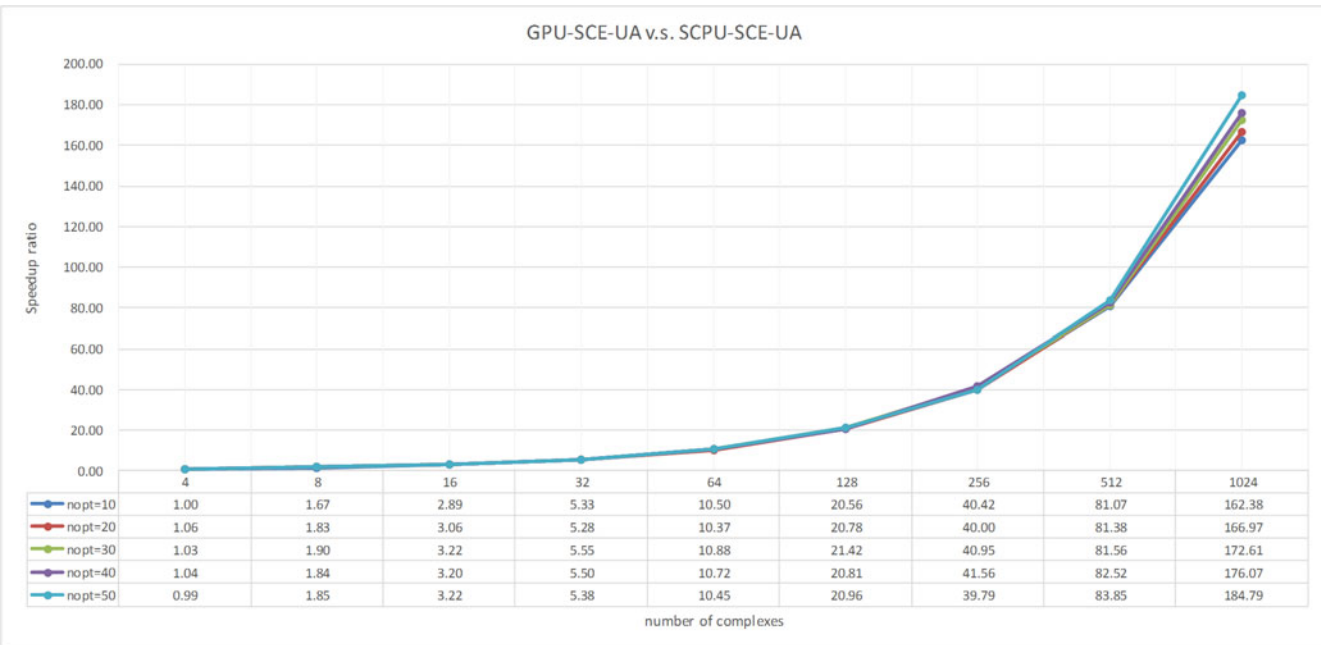
	4	8	16	32	64	128	256	512	1024
nopt=10	3.72	7.23	14.08	14.54	14.89	15.23	15.53	15.73	15.86
nopt=20	3.81	7.50	14.69	15.01	15.22	15.40	15.63	15.65	15.79
nopt=30	3.88	7.62	14.98	15.19	15.29	15.55	15.63	15.62	15.72
nopt=40	3.91	7.72	15.11	15.25	15.14	15.53	15.58	15.48	15.65
nopt=50	3.91	7.76	15.16	15.26	15.21	15.43	15.38	15.42	15.93

Fig. 8. Speedup ratio of MCPU-SCE-UA versus SCPU-SCE-UA.

3.3 Speedup Ratio Comparison

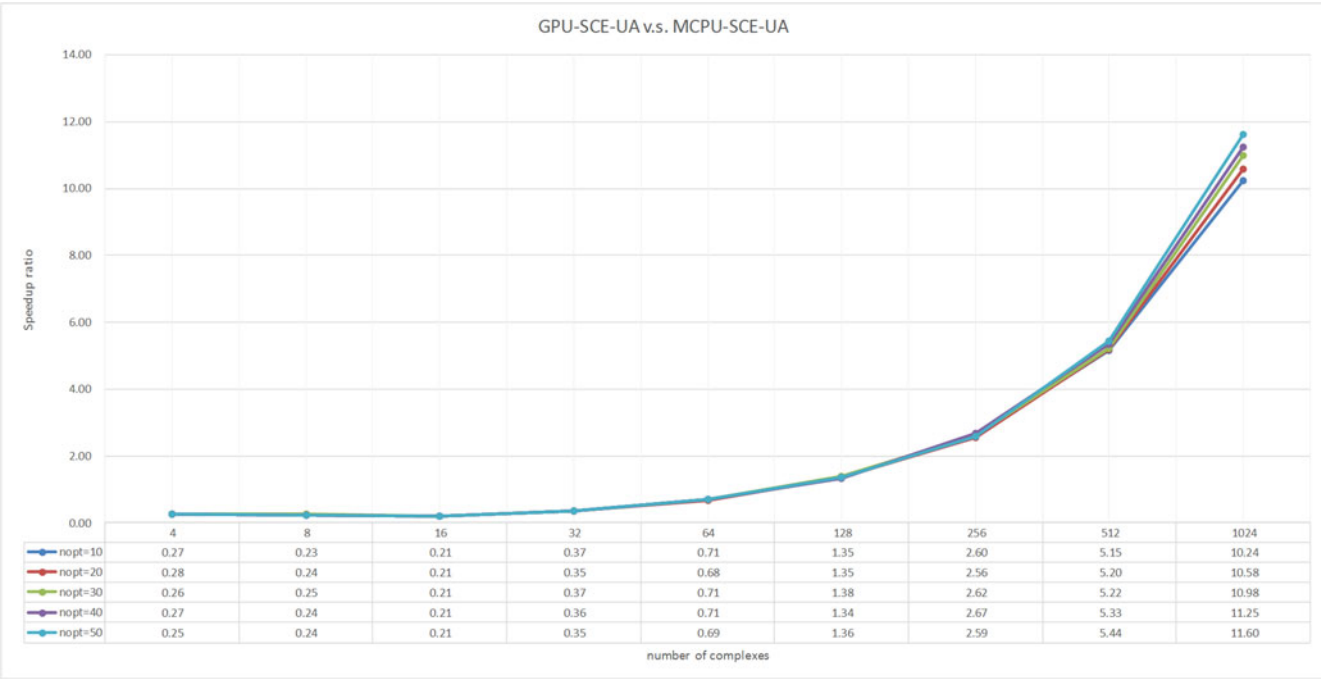
3.3.1 *MCPU-SCE-UA versus SCPU-SCE-UA*

The speedup ratio of the MCPU-SCE-UA verses the SCPU-SCE-UA is demonstrated in Fig. 8. The speedup ratio ranges from 3.72x to 15.93x. It can be seen from Fig. 8 that the speedup ratio increases when the number of complexes increases from 4 to 16. For the number of complexes larger than 16, the change degree of the speedup ratio is very small. With the increasing of the number of complexes, more threads are created. However, there are only 16 CPU cores and do not have enough CPU cores to faster execute large number of threads in parallel. Therefore, for number of complexes larger than 16, the speedup ratio cannot be significantly increased further more. With the increasing of the number of decision variables, the speedup ratio changes not too much. This is because that the range of the test problem dimensionality is from 10 to 50. The upper boundary of the dimensionality is not large enough to reflect the effect of the dimension to the speedup ratio. We do not test very high dimension problem because when dimension becomes very large, the CPU memory is not enough (32GB in this research). High dimension problem usually causes the memory overflow runtime error.



	4	8	16	32	64	128	256	512	1024
nopt=10	1.00	1.67	2.89	5.33	10.50	20.56	40.42	81.07	162.38
nopt=20	1.06	1.83	3.06	5.28	10.37	20.78	40.00	81.38	166.97
nopt=30	1.03	1.90	3.22	5.55	10.88	21.42	40.95	81.56	172.61
nopt=40	1.04	1.84	3.20	5.50	10.72	20.81	41.56	82.52	176.07
nopt=50	0.99	1.85	3.22	5.38	10.45	20.96	39.79	83.85	184.79

Fig. 9. Speedup ratio of GPU-SCE-UA versus SCPU-SCE-UA.



	4	8	16	32	64	128	256	512	1024
nopt=10	0.27	0.23	0.21	0.37	0.71	1.35	2.60	5.15	10.24
nopt=20	0.28	0.24	0.21	0.35	0.68	1.35	2.56	5.20	10.58
nopt=30	0.26	0.25	0.21	0.37	0.71	1.38	2.62	5.22	10.98
nopt=40	0.27	0.24	0.21	0.36	0.71	1.34	2.67	5.33	11.25
nopt=50	0.25	0.24	0.21	0.35	0.69	1.36	2.59	5.44	11.60

Fig. 10. Speedup ratio of GPU-SCE-UA versus MCPU-SCE-UA.

3.3.2 GPU-SCE-UA versus SCPU-SCE-UA

The speedup ratio of the GPU-SCE-UA verses the SCPU-SCE-UA is demonstrated in Fig. 9. The speedup ratio ranges from 0.99x to 184.79x. It can be seen from Fig. 9 that the speedup ratio increases when the number of complexes increases. For large number of complexes, the speedup ratio increases significantly. With the increasing of the number of complexes, more threads are created. The Tesla K40 GPU has 2880 cores to execute large number of threads in parallel and speed up the execution very much. With the increasing of the number of decision variables, the speedup ratio changes not too much. This is because that the range of the problem dimensionality is from 10 to 50. The upper boundary of the dimensionality is not large enough to reflect the effect of the dimension to the speedup ratio. We do not test very high dimension problem because when dimension becomes very large, the GPU memory is not enough (12GB in this research). High dimension problem usually causes the memory overflow runtime error.

3.3.3 GPU-SCE-UA versus MCPU-SCE-UA

The speedup ratio of the GPU-SCE-UA verses the MCPU-SCE-UA is demonstrated in Fig. 10. The speedup ratio ranges from 0.21x to 11.60x. It can be seen from Fig. 10 that the speedup ratio increases when the number of complexes increases. For small number of complexes ($\leqslant$64 in this research), the GPU version runs slower than the multi-core CPU version (speedup ratio $<$1.00x). For large number of complexes ($\geq$ 128 in this research), the GPU version runs faster than the multi-core CPU version (speedup ratio $>$1.00x). This is because that for small number of complexes ($\leqslant$64 in this research), the number of threads executed in parallel is small. The parallelism of the program is also small. Considered that the GPU clock speed is much lower than the CPU clock speed (0.875GHz versus 2.0GHz). Therefore the CPU version runs faster. For large number of complexes, the parallelism of the program increases significantly. Although the clock speed of the GPU is smaller than the CPU, however, the number of GPU cores is much larger than the CPU (2880 versus 16). For large number of threads, the parallel GPU program can make full use of all GPU cores and runs much faster than the CPU version. With the increasing of the number of decision variables, the speedup ratio changes not too much. This is because that the range of the problem dimensionality is from 10 to 50. The upper boundary of the dimensionality is not large enough to reflect the effect of the dimension to the speedup ratio. We do not test very high dimension problem because when dimension becomes very large, the CPU and GPU memory is not enough (32 GB and 12 GB in this research). High dimension problem usually causes the memory overflow runtime error.

4 Conclusions

In this paper, we deeply analyzed a famous global optimization method – SCE-UA and proposed a faster parallel SCE-UA method. The parallel SCE-UA was implemented on multi-core CPU and many-core GPU hardware devices by using the PGI Visual Fortran 14.10 with OpenMP and CUDA Fortran, respectively. The parallel SCE-UA runs much faster than the serial version. Its optimization accuracy is same as the serial version. Apart from the results mentioned above, three conclusions can be drawn here:

(1) The parallel SCE-UA can make full use of the computation capability of the parallel devices such as multi-core CPU and many-core GPU. By using the OpenMP and CUDA Fortran software development tools, the program implementations have good efficiency and satisfactory effectiveness when applied for the Griewank benchmark function.

(2) The comparison results indicated that the parallel SCE-UA converged much faster than the serial version and its optimization accuracy was the same as the serial version. The many-core GPU version runs faster when applied to problem with large number of complexes ($\geq$ 128 in this research). The multi-core CPU version runs faster when applied to problem with smaller number of complexes ($<$ 128 in this research). The parallel SCE-UA has a promising application prospect in the field of fast hydrological model parameter optimization.

(3) Although the parallel SCE-UA achieved good optimization accuracy and satisfactory convergence speed, it still has some deficiencies such as high memory requirement. The algorithm and the program should be further improved and optimized in future researches. With the improved program, the effect of the high dimensionality should also be tested in the future.

Acknowledgments

This research was funded by the IWHR Research & Development Support Program (JZ0145B052016) IWHR Scientific Research Projects of Outstanding Young Scientists "Research and application on the fast global optimization method for the Xinanjiang model parameters based on the high performance heterogeneous computing" (No. KY1605), China Postdoctoral Science Foundation on Grant (Grant NO.: 2016M591214), Specific Research of China Institute of Water Resources and Hydropower Research (Grant Nos. Fangji 1240), the Third Sub-Project: Flood Forecasting, Controlling and Flood Prevention Aided Software Development - Flood Control Early Warning Communication System and Flood Forecasting, Controlling and Flood Prevention Aided Software Development for Poyang Lake Area of Jiangxi Province (0628-136006104242, JZ0205A432013, SLXMB200902), the NNSF of China, Study on the integrated assessment model for risk and benefit of dynamic control of reservoir waterlevel in flood season (No. 51509268), and the NNSF of China, Numerical Simulation Technology of Flash Flood based on Godunov Scheme and Its Mechanism Study by Experiment (No. 51509263). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. Guangyuan Kan, Tianjie Lei, and Ke Liang contributed equally to this work. Guangyuan Kan is the corresponding author.

References

[1] W. Chu, X. Gao, and S. Sorooshian, "A new evolutionary search strategy for global optimization of high-dimensional problems," *Inf. Sci.*, vol. 181, no. 22, pp. 4909–4927, 2011.

[2] W. Chu, T. Yang, and X. Gao, Comment on "High-dimensional posterior exploration of hydrologic models using multi-try DREAM (ZS) and high-performance computing" by Eric Laloy and Jasper A. Vrugt, *Water Resour. Res.*, vol. 50, no. 3, pp. 2775–2780, 2014.

[3] V. Cooper, V. Nguyen, and J. Nicell, "Calibration of conceptual rainfall-runoff models using global optimization methods with hydrologic process-based parameter constraints", *J. Hydrol.*, vol. 334, no. 3/4, pp. 455–466, 2007.

[4] J. Dong, C. Zheng, G. Kan, J. Wen, M. Zhao, and J. Yu, "Applying the ensemble artificial neural network-based hybrid data-driven model to daily total load forecasting",*Neural Comput. Appl.*, vol. 26, no. 3, pp. 603–611, 2015.

[5] Q. Duan "A global optimization strategy for efficient and effective calibration of hydrologic models", *Ph.D. dissertation, Dept. Hydrol. Water Resour.* Univ. Arizona, Tucson, AZ., 1991.

[6] Q. Duan, S. Sorooshian, and V. Gupta, "Effective and efficient global optimization for conceptual rainfall-runoff models", *Water Resour. Res.*, vol. 28, no. 4, pp. 1015–1031, 1992.

[7] Q. Duan, V. Gupta, and S. Sorooshian, "A Shuffled complex evolution approach for effective and efficient global minimization", *J. Optim. Theory Appl.*, vol. 76, no. 3, pp. 501–521, 1993.

[8] Q. Duan, S. Sorooshian, and V. Gupta, "Optimal use of the SCE-UA global optimization method for calibrating watershed models", *J. Hydrol.*, vol. 158, no. 3/4, pp. 265–284, 1994.

[9] J. Holland, *Adaption in Natural and Artificial Systems* Univ. Michigan Press. Ann Arbor, MI. 1975.

[10] M. Humphrey, N. Beekwilder, J. Goodall, and M. Ercan, "Calibration of watershed models using cloud computing", in *Proc. 8th IEEE Int. Conf. eScience 2012*", Chicago Illinois, Oct. 2012, pp. 8–12.

[11] G. Kan, et al., "Improving event-based rainfall-runoff simulation using an ensemble artificial neural network based hybrid data-driven model", *Stoch. Environ. Res. Risk Assess.*, vol. 29, no. 5, pp. 1345–1370, 2015.

[12] G. Kan, et al., "A new hybrid data-driven model for event-based rainfall-runoff simulation", in *Proc. Neural Comput. Appl.*, 2016, pp. 1–16, doi: 10.1007/s00521-016-2200-4.

[13] Z. Li, G. Kan, C. Yao, Z. Liu, Q. Li, S. Yu, "An improved neural network model and its application in hydrological simulation", *J. Hydrol. Eng.*, vol. 19, no, 10, 04014019-1–04014019-17, 2014.

[14] J. Nelder and R. Mead, "A simplex method for function minimization", *J. Comput.*, vol. 7, no. 4, pp. 308–313, 1965.

[15] A. Ouyang, Z. Tang, X. Zhou, Y. Xu, G. Pan, and K. Li, "Parallel hybrid PSO with CUDA for ID heat conduction equation", *Comput. Fluids*, vol. 110, no. 30, pp. 198–210, 2015.

[16] W. Price, *A Controlled Random Search Procedure for Global Optimization, Toward Global Optimization 2*, L. C. W. Dixon and G. P. Szegö, Eds. North-Holland, Amsterdam, Holland, 71–84, 1978.

[17] W. Price, "Global optimization by controlled random search", *J. Optim. Theory Appl.*, vol. 40, no. 3, pp. 333–348, 1983.

[18] W. Price, "Global optimization algorithms for a CAD workstation", *J. Optim. Theory Appl.*, vol. 55, no. 1, pp. 133–146, 1987.

[19] S. Lee, M. Jeon, D. Kim, and A. Sohn, "Partitioned Parallel Radix Sort", *J. Parallel Distrib. Comput.*, vol. 62, no. 4, pp. 656–668, 2002.

[20] S. Harase, "On the F2-linear relations of Mersenne Twister pseudorandom number generators", *Math. Comput. Simul.*, vol. 100, pp. 103–113, 2014.

[21] T. Thanakulwarapas and J. Werapun, "An optimized bitonic sorting strategy with midpoint-based dynamic communication", *J. Parallel Distrib. Comput.* vol. 84, pp. 37–50, 2015.

[22] A. Umbarkar, M. Joshi, and W. Hong, "Multithreaded parallel dual population genetic algorithm (MPDPGA) for unconstrained function optimizations on multi-core system", *Appl. Math. Comput.*, vol. 243, no. 15, pp. 936–949, 2014.

[23] V. Demchik, "Pseudo-random number generators for Monte Carlo simulations on ATI Graphics Processing Units," *Comput. Phys. Commun.*, vol. 182, no. 3, pp. 692–705, 2011.

[24] J. Vrugt, B. Nualláin, B. Robinson, W. Bouten, S. Dekker, and P. Sloot, "Application of parallel computing to stochastic parameter estimation in environmental models," *Comput. Geosci.*, vol. 32, no. 8, pp. 1139–1155, 2005.

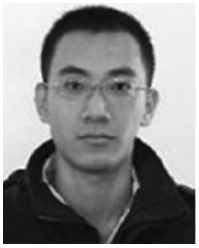
[25] D. Xu, W. Wang, K. Chau, C. Cheng, and S. Chen, "Comparison of three global optimization algorithms for calibration of the Xinanjiang model parameters", *J. Hydroinformatics*, vol. 15, no. 1, pp. 174–193, 2013.

[26] T. Yang, X. Gao, S. Sellars, and S. Sorroshian, "Improving the multi-objective evolutionary optimization algorithm for hydropower reservoir operations in the California Oroville-Thermalito complex", *Environ. Modelling Softw.*, vol. 69, pp. 262–279, 2014.

[27] P. Yapo, V. Gupta, S. Sorooshian, "Multi-objective global optimization for hydrologic models", *J. Hydrology*, vol. 204, pp. 83–97, 1998.

[28] L. Zhang, Y. Lu, M. Guo, S. Guo, C. Xu, "Architecture-based design and optimization of genetic algorithms on multi- and many-core systems", *Future Gener. Comput. Syst.*, vol. 38, pp. 75–91, 2013.

[29] S. Zhang, J. Shi, "A microwave wetland surface emissivity calibration scheme using SCE-UA algorithm and AMSR-E brightness temperature data", *Proc. 3rd Int. Conf. Environ. Sci. Inf. Appl. Technol.*, 2011, vol. 10, pp. 2731–2739.



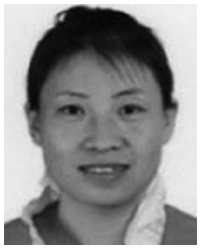
Guangyuan Kan received the BS and PhD degrees from the College of Hydrology and Water Resources, Hohai University in Nanjing, China, in 2007 and 2014, respectively. Now he works towards the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Research Center on Flood & Drought Disaster Reduction of the Ministry of Water Resources, China Institute of Water Resources and Hydropower Research. His research interests include watershed hydrological modeling, optimization algorithms, numerical techniques, big hydrological data and data mining, climate weather modeling, development tools and libraries, machine learning, and parallel computing.

Tianjie Lei received the BS, MS, and PhD degrees from the Henan Polytechnic University and Beijing Normal University in China, in 2008, 2011, and 2015, respectively. Now he works towards the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Research Center on Flood & Drought Disaster Reduction of the Ministry of Water Resources, China Institute of Water Resources and Hydropower Research. His research interests include image processing and information extraction of Unmanned Aerial Vehicle Remote Sensing, machine learning, climatic and ecological model modeling, socioeconomic and ecological assessment of drought impacts.

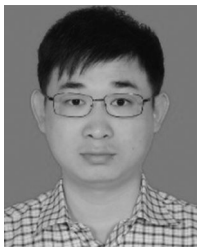
Ke Liang received the BS degrees from the College of Hydrology and Water Resources, Hohai University in Nanjing, China, in 2014. Now she is a second grade graduate student of Hohai University. Her research interests include hydrological and physical law simulating, watershed hydrological modeling.

Jiren Li received the BS, MS, and PhD degrees from the Hohai University in Nanjing, China, and Universitē de Montpellier, France, in 1967, 1982, and 1987, respectively. Now he works towards the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Research Center on Flood & Drought Disaster Reduction of the Ministry of Water Resources, China Institute of Water Resources and Hydropower Research. His research interests include watershed hydrological modeling, climate weather modeling, development tools and libraries, and parallel computing.

Liuqian Ding received the BS and MS degrees from the Zhengzhou University in Zhengzhou, China and China Institute of Water Resources and Hydropower Research, Beijing, China, in 1985 and 1988, respectively. Now he works towards the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Research Center on Flood & Drought Disaster Reduction of the Ministry of Water Resources, China Institute of Water Resources and Hydropower Research. His research interests include flood defense and parallel computing.

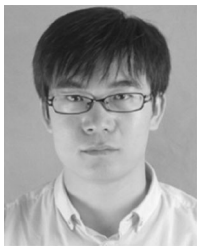


Xiaoyan He received the BS, MS, and PhD degrees from the Taiyuan University of Technology, China and China Institute of Water Resources and Hydropower Research, China in 1995, 2001, and 2009, respectively. Now she works towards the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Research Center on Flood & Drought Disaster Reduction of the Ministry of Water Resources, China Institute of Water Resources and Hydropower Research. Her research interests include watershed hydrological modeling, numerical techniques, big hydrological data and data mining, and parallel computing.



Haijun Yu received the BE and PhD degrees from the School of Civil Engineering and Transportation, South China University of Technology in Guangzhou, China, in 2010 and 2015, respectively. Now he works towards the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Research Center on Flood & Drought Disaster Reduction of the Ministry of Water Resources, China Institute of Water Resources and Hydropower Research. His research interests include hydrodynamic model, urban water management and urban drainage and waterlogging.

Dawei Zhang received the PhD degrees from the Tsinghua University in Beijing, China, in 2008. Now he works towards the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Research Center on Flood & Drought Disaster Reduction of the Ministry of Water Resources, China Institute of Water Resources and Hydropower Research. His research interests include hydrological process simulation, hydrodynamic modeling and flood risk analysis.



Depeng Zuo received the BS degrees from the College of Hydrology and Water Resources, Hohai University in Nanjing of China in 2007, and Ph.D. in College of Water Sciences, Beijing Normal University in Beijing of China in 2012, respectively. Now he works towards the College of Water Sciences, Beijing Normal University. His research interests include hydrological modeling, ecohydrology, drought assessment, and climate change impacts on water resources.

Dr Zhenxin Bao is an engineer at Water Resources Department of the Nanjing Hydraulic Research Institute. His research interests include climatic change impacts on hydrology and water resources, hydrological modelling, and statistical applications in hydrology and water resources. He has published more than 20 peer-reviewed papers including on some international prestigious SCI journals, such as Journal of Hydrology and Hydrological Processes.

Mark Amo-Boateng received the BS, MS, and PhD degrees from the KNUST, Ghana and Hohai University, China, in 2009, 2010, and 2014, respectively. Now he works towards the University of Energy and Natural Resources. His research interests include watershed hydrological modeling, optimization algorithms, numerical techniques, development tools and libraries, and parallel computing.

Youbing Hu received the BS, MS, and PhD degrees from the School of Earth and Environment, Anhui University of Science & Technology in Huainan, China, in 2008, and from the College of Hydrology and Water Resources, Hohai University in Nanjing, China, in 2010, and 2014, respectively. Now he works towards the Hydrologic Bureau (information center), the Huai River Commission. His research interests include watershed hydrological modeling, hydrological forecasting, hydrological data mining, and parallel computing.

Mengjie Zhang received the BS, and MS degrees from the College of Hydrology and Water Resources, Hohai University in Nanjing, China, in 2012, 2015, respectively. Now she works towards the State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Department of Water Resources(DWR), China Institute of Water Resources and Hydropower Research. Her research interests include watershed hydrological modeling, hydrologic forecasting, climate weather modeling, water disaster and water security.