Hindawi
Journal of Optimization
Volume 2017, Article ID 4685923, 9 pages
https://doi.org/10.1155/2017/4685923

Research Article

A Novel Distributed Quantum-Behaved Particle Swarm Optimization

Yangyang Li,[1] Zhenghan Chen,[1] Yang Wang,[1] Licheng Jiao,[1] and Yu Xue[2]

[1]*Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Joint International Research Laboratory of Intelligent Perception and Computation, Xidian University, Xi'an, Shaanxi Province 710071, China*
[2]*School of Computer and Software, Nanjing University of Information Science and Technology (NUIST), Nanjing 210044, China*

Correspondence should be addressed to Yangyang Li; lyy_791@163.com

Received 29 December 2016; Accepted 4 April 2017; Published 3 May 2017

Academic Editor: Gexiang Zhang

Quantum-behaved particle swarm optimization (QPSO) is an improved version of particle swarm optimization (PSO) and has shown superior performance on many optimization problems. But for now, it may not always satisfy the situations. Nowadays, problems become larger and more complex, and most serial optimization algorithms cannot deal with the problem or need plenty of computing cost. Fortunately, as an effective model in dealing with problems with big data which need huge computation, MapReduce has been widely used in many areas. In this paper, we implement QPSO on MapReduce model and propose MapReduce quantum-behaved particle swarm optimization (MRQPSO) which achieves parallel and distributed QPSO. Comparisons are made between MRQPSO and QPSO on some test problems and nonlinear equation systems. The results show that MRQPSO could complete computing task with less time. Meanwhile, from the view of optimization performance, MRQPSO outperforms QPSO in many cases.

1. Introduction

With the development of information science, more and more data is stored, such as web content and bioinformatics data. For this reason, many basic problems have become more and more complex, which makes great troubles to current intelligent algorithm. As one of the most important issues in artificial intelligence, optimization problem in real-world applications also becomes harder and harder to be solved.

In the past 30 years, evolutionary algorithm is becoming one of the most effective intelligent optimization methods. In order to face the new challenge, distributed evolutionary algorithms (dEAs) have been blossomed rapidly. The paper [1] provides a comprehensive survey of the distributed EAs and some models are summarized. Master-slave, island, cellular, hierarchical, pool, coevolution, and multiagent models are listed and introduced. And the different models are analyzed from aspects like parallelism level, communication cost, scalability, and fault-tolerance. And some hotspots of dEAs, such as cloud and MapReduce-based implementations and GPU and CUDA-based implementations, are listed. But no results of dEAs on distributed computing devices are reported. Cloud can be applied in many aspects, and [2–8] have realized various specific applications of cloud. The paper [9] gives a review of the parallel and distributed genetic algorithms in graphics processing unit (GPU). Some works along this idea are reported, such as [10–12]. Cloud and MapReduce is a new and effective technology to deal with big data, which is proposed by Google in 2004 [13]. To respond to the requirement of parallelization and distribution, this physical platform is very convenient to deploy an algorithm to update to be parallel. The programmers only need to consider the map function and reduce function, and the other details are provided by the model itself. Many practical problems are solved with MapReduce model and cluster of servers, such as path problem in large-scale networks [14], seismic signal analysis [15], image segmentation [16], and location recommendation [17]. But the study about MapReduce-based

EAs is still in initial stage. Although some genetic algorithms [18–23] and particle swarm optimization realized by MapReduce [24] have been proposed. There are still many kinds of EAs which are not implemented with distributed model and parallel potential of these algorithms is not released. Based on these considerations, in our previous work [25], MapReduce is combined with coevolutionary particle swarm optimization, which shows that MapReduce-based CPSO obtain much better performance than CPSO. In another work [26], the quantum-behaved particle swarm optimization is transformed on MapReduce successfully. And the idea of this paper is based on it and continues extending that the background is introduced and a practical application is added.

Quantum mechanics and trajectory analysis gained extensive attention of scholars recently and sparkled in many areas, such as image segmentation [27], neural network [28], and population-based algorithms [29, 30]. In [31], Zhang presents a systematic review of quantum-inspired evolutionary algorithms. Quantum-behaved particle swarm optimization is a kind of PSO proposed by Sun et al. in 2004 [32]. Inspired by movement of particle in quantum space, a new reproduction operator of solution is proposed in this algorithm. Because a particle could arrives at any location in quantum space with a certain probability, a new solution at any location in feasible space also could be generated with a certain probability in QPSO. This mechanism is helpful for particles to avoid falling in local optimum. Some more detailed analysis has been reported in [33]. Unfortunately, when the algorithm faces large-scale and complex problem, the increasing computational cost becomes the bottleneck of this algorithm and without enough computing resource premature phenomenon could not been avoided, which urges the original algorithm to be parallel.

In order to follow this trend and enhance the capabilities of a standard QPSO, the MapReduce quantum-behaved particle swarm optimization is developed. The MRQPSO transplants the QPSO on MapReduce model and makes the QPSO parallel and distributed through partitioning the search space. Through the comparisons of MRQPSO and standard QPSO, it could be found that the proposed MRQPSO decreases the time of same function evaluations. And on some test problems QPSO increases the performance of solution and is more robust than QPSO.

The rest of this paper is organized as follows. Section 2 introduces the PSO and QPSO. Section 3 gives a brief presentation of MapReduce model. Section 4 describes the details of QPSO implementing on MapReduce. In Section 5, we show and analyze results of experiments, including the comparison with QPSO. Finally Section 6 concludes the work in this paper.

2. PSO and QPSO

2.1. The Particle Swarm Optimization Algorithm. Inspired by bird and fish flocks, Kennedy and Eberhart proposed PSO algorithm in 1995 [34]. This algorithm is a population-based intelligent search algorithm. In order to find the food as quickly as possible, the birds in a flock would trace their companions that are near to the food firstly. Then they would determine accurate area of food. The individual of PSO searches the optimum like the bird in a flock. Each particle has velocity and position. And the two parameters would be updated according to best value and global best value of the particles. The velocity and position of particle i at the dimension j are presented by v_{ij} and x_{ij}, respectively. The updating equation could be described as

$$\begin{aligned} v_{ij}(t+1) &= w v_{ij}(t) + c_1 r_1 \left(pB_{ij}(t) - x_{ij}(t)\right) \\ &\quad + c_2 r_2 \left(gB_{ij}(t) - x_{ij}(t)\right), \\ x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1), \end{aligned} \tag{1}$$

where $v_{ij}(t)$ and $x_{ij}(t)$ are the velocity and position. t represents the tth iteration. pB and gB are best value and global best value of the particle, respectively. r_1 and r_2 are random number uniformly distributed in $[0, 1]$. w, c_1, and c_2 are three parameters of the algorithm. w is initial weight proposed by Shi and Eberhart in 1998 [35] to control the balance of local and global optimum. c_1 and c_2 are the accelerated coefficients or learning factors. Usually, $c_1 = c_2 = 2$.

From the above equations, it could be found that few parameters are used in PSO, which makes PSO easy to be controlled and used. Meanwhile, it has better convergence performance and quicker convergence speed. These advantages make the PSO algorithm gain a lot of research attention. However, the PSO is not a global optimization algorithm [36]. The limited velocity constrains the search space in a limited area. So the PSO could not always find the global optimum. In other words, the premature convergence is the most serious drawback of the PSO.

2.2. The Quantum-Behaved Particle Swarm Optimization Algorithm. To overcome the shortcoming of the original PSO algorithm, Sun et al. proposed the quantum-behaved particle swarm optimization (QPSO) in 2004 [32]. This algorithm has a more superior performance comparing to the PSO. QPSO algorithm transfers the search space from classical space to quantum space. Particles can appear at any position, which implement the full search in the solution space.

According to uncertainty principle, the velocity and position of a particle cannot be determined simultaneously. In quantum space, a probability function of the position where a particle appears could be obtained from Schrodinger equation. The true position of one particle could be measured by Monte Carlo method. Based on these ideas, in QPSO, a local attractor is constructed by particle best solutions and global best solution as (2) for each particle.

$$P_{ij}(t+1) = \phi_j(t) * pB_{ij}(t) + \left(1 - \phi_j\right) * gB_{ij}(t), \tag{2}$$

where $P_{ij}(t)$ is a local attractor of the particle i at the dimension j. $\varphi_j(t)$ is random number distributed in $[0, 1]$. $pB_{ij}(t)$ is the particle best solution. $gB_{ij}(t)$ is the current global best solution.

The position of the particle is updated by

$$X_{ij}(t+1) = P_{ij}(t+1) \pm \alpha * \left|mbest - X_{ij}(t)\right| * \ln\left(\frac{1}{u}\right), \quad (3)$$

where α is the only parameter in the algorithm called creativity coefficient, which is a positive real number, to adjust the balance of local and global search. The definition of α refers to (4). $\text{Iter}_{\max}$ is the maximum number of iterations. u is random number distributed in $[0, 1]$, and *mbest* is the mean position and defined as follows:

$$\alpha = \frac{(1 - 0.5) * (\text{Iter}_{\max} - t)}{\text{Iter}_{\max}} + 0.5, \quad (4)$$

$$mbest = \left(\frac{1}{M}\sum_{i=1}^{M} P_{i1}, \frac{1}{M}\sum_{i=1}^{M} P_{i2}, \ldots, \frac{1}{M}\sum_{i=1}^{M} P_{id}, \right), \quad (5)$$

where M is the size of population. P_i is the global extremum of particle i.

In QPSO, first step is initializing the population randomly, which concludes the position of each particle, particle best value, and global best value. Next, calculate the mean position of jth dimension according to (5). Then, particle is evaluated again and the best and global best solution of the particles would be updated according to the fitness value. After that, the particle is updated as (2) and (3). When the number of iterations or accuracy requirements is satisfied, stop running and output the optimum.

Although the QPSO algorithm is superior to PSO, it still has some disadvantages. Because the particles in the QPSO fly discretely, the narrow area where the optimum is may be missed. In the case of too much computation, QPSO may spend too much time.

3. MapReduce

MapReduce [13] is a programming model proposed by Dean and Ghemawat. Inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages, this model is created for processing the large-scale data in parallel. The infrastructure of MapReduce provides detailed implementations of communications, load balancing, fault-tolerance, resource allocation, file distribution, and so forth [1]. Programmers do not need a lot of knowledge and experiments about parallel and distributed programming. They only need to pay attention to *map* and *reduce* function which the model consists of and then can implement algorithm to parallelization easily.

In this model, the computation takes a set of key/value pairs. The *map* function processes the *input* key/value pairs and then emits new lists of key/value pairs, called *intermediate* key/value pairs. The type of these two lists may be from different domain. The *map* function is called independently, and the parallelization is implemented in this way. After all *map* functions' processions are completed, the *reduce* function is called. *Intermediate* key/value pairs are grouped and passed to reduce function. The reduce phase merges and integrates these intermediate key/value pairs and outputs the *output* key/value pairs finally. And the type of *intermediate* and *output* pairs must be the same. The type of *map* and *reduce* functions can be written as follows:

$$\begin{aligned} \text{map:}\ & (k1, v1) \longrightarrow \text{list}(k2, v2), \\ \text{reduce:}\ & (k2, \text{list}(v2)) \longrightarrow \text{list}(v2). \end{aligned} \quad (6)$$

Because Google has not released the system to the public, Hadoop, the Apache Lucene project developed, has been used generally. This Java-based open-source platform is a clone of MapReduce infrastructure, and we can use it to design and implement our distributed computation.

4. The MRQPSO Algorithm

The particle swarm optimization algorithm [34] is one of the popular evolutionary algorithms. It has attracted much attention because of the merits of simple concept, rapid convergence, and good quality of solution. However, this algorithm is bothered by some weakness, such as premature phenomenon. Focusing on the shortcoming of original PSO, Sun et al. proposed an uncertain and global random algorithm named quantum-behaved particle swarm optimization (QPSO) in 2004 [32]. The new one puts the search space into quantum space to let the particle move to any location with different probability. Through this strategy, the premature phenomenon could be solved to a certain degree.

Although the QPSO has satisfying progress on premature phenomenon, it has not been prepared to challenge of problems with complex landscape or needing huge computation to be solved. Due to the particles of the QPSO flying discretely, they may miss the narrow area where the global optimum is. And as the problem is getting complex, the computational cost increases. So we implement the QPSO parallel and distributed by transplanting the algorithm on MapReduce model and we name this algorithm MRQPSO. The framework of MRQPSO could be described as in Algorithm 1, and the flowchart is shown as in Figure 1.

The proposed MRQPSO partitions the search space into many subspaces. For n-dimension search space, the range of each dimension is cut to be m_i parts ($0 < i < n$); then the space-partition is completed, and $m_1 * m_2 * \cdots * m_n$ subspace is obtained [25]. Then using several servers, several mappers perform QPSO on different subspace in parallel and independently. After all the mappers finished the calculation, the reducer merges and integrates the immediate value and outputs the best solution. The space-partition helps the particle get distributed uniformly, which ensures all areas have the particles fall in at the initialization phase. It is effective to avoid the particles overflying the narrow zone where the optimum may lie. And the parallel mappers could help MRQPSO to save time cost.

4.1. MRQPSO Map Function. Algorithm 1 shows the pseudo code of the map function of proposed MRQPSO. *pbest* is the position of particle with best value and *gbest* is the

Step 1. Divide feasible space into several subspace;
Step 2. Construct Mapper which performs QPSO on one subspace and outputs the obtained optimal solution on this subspace;
Step 3. Construct Reducer which selects the best optimal solution on different subspace from mapper;
Step 4. Output the best optimal solution and its functional value.

Algorithm 1: Framework of MRQPSO.

```
function mapper (key, value) {
    initialize the positions of all particles
    evaluate the function values of positions then select the pbest and gbest
    // update the particle
    while the termination condition is not met {
        calculate the mbest and α
        for each particle {
            update the pbest and gbest
            for each dimension {
                update position
            }
        }
        calculation + 1
    }
    emit a message (ID of gbest, string of gbest and fitness)
}
```

Algorithm 2: MRQPSO map.

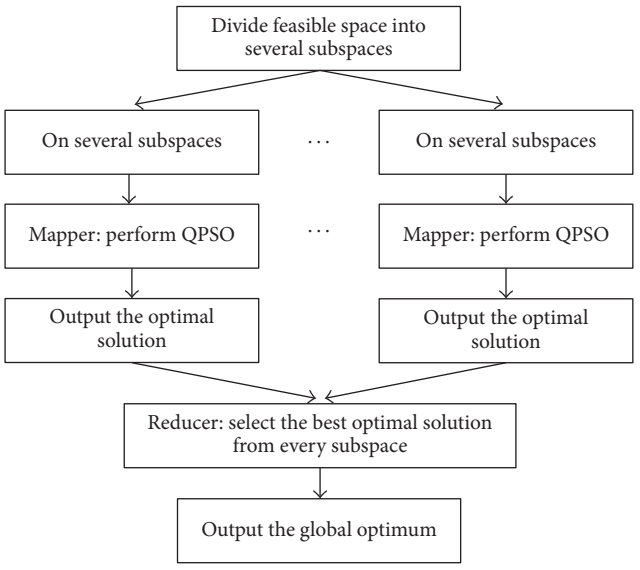

Figure 1: Flowchart of MRQPSO.

position of solution with global best value. Several subspaces are saved as records and form data block. The mapper is called when a block starts a QPSO procedure. The input key/value pairs denote the massages of data block. The key is the ID of one record and the value is the string of search space. Then the mappers start to process the QPSO in every block independently. Once a block has been explored, another block will follow up immediately. Under ideal conditions, the larger the number of mappers is, the fewer process a single mapper processes, and the fuller the parallelization is. However, the mapper would spend time to be started in fact. If the data is big enough, the starting time can be neglected. But in our experiments, it will influence the results more or less.

After being processed by mappers, the immediate key/value pairs change to denote the information of *gbest* and global optimum of current data block, showed as Algorithm 2. And then the immediate key/value pairs are ready to transport to the reduce phase.

4.2. MRQPSO Reduce Function. The reduce function is in charge of merging and integrating the information which the mapper emitted. As Algorithm 3 shows, the reducer of MRQPSO is used to select the minimum from all subspaces. The mappers produced and transported the immediate key/value pairs which are received by the reducer after all mappers completed their work. At the reduce phase, all *gbest* and corresponding fitness of blocks are compared with each other. And the minimum of them is selected and outputted finally.

5. Experiment Result and Analysis

5.1. Performance of MRQPSO on Test Problems. To validate the proposed MRQPSO algorithm, we selected 8 functions to evaluate the ability of solving complex problems firstly.

Table 1: Benchmark functions used in this paper.

	Function	Description
Composition functions	F1	Composition function 1 ($n = 5$, rotated)
	F2	Composition function 2 ($n = 3$, unrotated)
	F3	Composition function 3 ($n = 3$, rotated)
	F4	Composition function 4 ($n = 3$, rotated)
	F5	Composition function 5 ($n = 3$, rotated)
	F6	Composition function 6 ($n = 5$, rotated)
	F7	Composition function 7 ($n = 5$, rotated)
	F8	Composition function 8 ($n = 5$, rotated)

```
function reducer (key, value) {
    combine the message emitted from mapper
    // get the gbest and global optimum
    for each data block {
        if fitness < global best {
            global best = fitness
        }
    }
    emit gbest and global optimum
}
```

Algorithm 3: MRQPSO reduce.

The scalable optimization problems are proposed in the CEC 2013 Special Session on Real-Parameter Optimization [37] and listed in Table 1. All the test composition functions are in the same search range: $[-100, 100]^D$. And they are all minimization problem with global optimum zeros. n is the number of kinds of basic benchmark functions.

Some parameter settings and environment are listed as follows.

We compared our proposed MRQPSO with original QPSO algorithm to test the optimization performance. Each function is run for 20 independent times and all the results are recorded in Tables 2–5. All experiments are run for $2^{13} \times 900$ function evaluations. $D = 10$.

(1) QPSO: this algorithm transforms search space from classic space to quantum space. In quantum space, particles can appear at any position and avoid premature convergence to some degrees. The population size of QPSO is 10.

(2) MRQPSO: this algorithm is a QPSO implementation on the MapReduce model, which achieves the parallel and distributed QPSO. The population sizes of MRQPSO were 10, 20, and 30, respectively, denoted by s on Table 2. The search space is partitioned into 2^{11} blocks averagely.

All experiments are run on VMware Workstation virtual machines version 12.0.0: one processor, 1.0 GB RAM. Hadoop version 1.1.2 and Java 1.7 were used in MapReduce experiments; we used three virtual machines while serial algorithm used one. CPU is core i7. Programming language is Java.

In Table 2, the best, worst, mean value, standard derivation, and running time of MRQPSO with different population sizes are listed. According to the results, as the population size becomes larger, the solutions become worse. It may be because that when the number of particles increases, the number of iterations of each particle decreased for the same function evaluations, which is not helpful to improve accuracy.

5.2. Comparison with QPSO on Test Problems. The results of MRCPSO are compared with QPSO algorithm in Tables 3 and 4. The population size is set to 10. We show two columns for each item to compare two algorithms clearly. From Table 3, MRQPSO has a better solution almost on all items. For F2 and F3, although the best value of QPSO is lower, MRQPSO meet this value nearly. Since these two values are close, one can consider that two algorithms are trapped into the same local optimum. And due to the less number of iterations of each particle, the MRQPSO cannot converge to a lower point like QPSO. In general, the MRQPSO has a better performance on mean value and standard derivation; this suggests the MRQPSO is more capable of searching for the optimum and overcoming the premature phenomenon and is more robust and steady than original QPSO.

The notable advantage in time is presented in Table 4. From this chart, the MRQPSO is more effective in saving time cost. And it seems that the more time QPSO spend, the more advantage MRQPSO has. Normally, it takes some time to start mapper. When a problem is so simple that the serial algorithm processes fast, the outstanding benefit of rapid convergence may weaken, such as F1–F3. But when search time gets longer, the mapper starting time even can be negligible, such as F4–F7, where the MRQPSO programs' running time reduced to half compared to the QPSO.

To summarize, we can discover that the MRQPSO has better solution performance and cost less running time. The proposed MRQPSO is more suitable and effective for dealing with complex problems.

5.3. Comparisons on Nonlinear Equation Systems. Nonlinear equation systems arise in many areas, such as economics [38],

Table 2: Performance of MRQPSO.

Fun	s	Min value	Max value	Mean function value	St. d	Mean running time (ms)
F1	10	$3.17E+01$	$1.23E+02$	$8.10E+01$	$3.26E+01$	52098
	20	$2.98E+01$	$1.21E+02$	$9.34E+01$	$2.86E+01$	55393
	30	$4.16E+01$	$1.37E+02$	$1.07E+02$	$3.11E+01$	63067
F2	10	$4.26E+01$	$1.36E+02$	$9.07E+01$	$2.09E+01$	57123
	20	$1.14E+02$	$2.50E+02$	$1.84E+02$	$3.58E+01$	59785
	30	$1.82E+02$	$4.53E+02$	$2.98E+02$	$6.93E+01$	68330
F3	10	$1.78E+02$	$4.76E+02$	$3.65E+02$	$7.50E+01$	58311
	20	$1.88E+02$	$7.83E+02$	$5.54E+02$	$1.57E+02$	61245
	30	$4.84E+02$	$9.22E+02$	$7.58E+02$	$1.12E+02$	68463
F4	10	$1.04E+02$	$1.13E+02$	$1.09E+02$	$1.75E+00$	942736
	20	$1.08E+02$	$1.14E+02$	$1.11E+02$	$1.39E+00$	978615
	30	$1.07E+02$	$1.16E+02$	$1.13E+02$	$2.46E+00$	1108999
F5	10	$1.07E+02$	$1.15E+02$	$1.12E+02$	$2.62E+00$	903781
	20	$1.10E+02$	$1.18E+02$	$1.14E+02$	$2.61E+00$	984774
	30	$1.10E+02$	$1.20E+02$	$1.16E+02$	$2.89E+00$	1134965
F6	10	$1.06E+02$	$1.11E+02$	$1.08E+02$	$1.09E+00$	982763
	20	$1.04E+02$	$1.11E+02$	$1.08E+02$	$2.04E+00$	1053224
	30	$1.09E+02$	$1.16E+02$	$1.12E+02$	$1.94E+00$	1183599
F7	10	$1.54E+02$	$3.42E+02$	$2.37E+02$	$5.08E+01$	970225
	20	$1.61E+02$	$3.54E+02$	$2.49E+02$	$5.23E+01$	1056708
	30	$1.92E+02$	$3.71E+02$	$2.91E+02$	$4.68E+01$	1239350
F8	10	$1.04E+02$	$1.23E+02$	$1.14E+02$	$5.16E+00$	76354
	20	$1.08E+02$	$1.25E+02$	$1.15E+02$	$3.93E+00$	81578
	30	$1.15E+02$	$1.43E+02$	$1.28E+02$	$8.02E+00$	89552

Table 3: Comparison between MRQPSO and QPSO on the optimum.

Fun	Min value		Max value		Mean function value		St. d	
	MR QPSO	QPSO	MR QPSO	QPSO	MR QPSO	QPSO	MR QPSO	QPSO
F1	**3.17E + 01**	$2.13E+02$	**1.23E + 02**	$8.32E+02$	**8.10E + 01**	$4.45E+02$	**3.26E + 01**	$1.51E+02$
F2	$4.26E+01$	**3.69E + 01**	**1.36E + 02**	$7.96E+02$	**9.07E + 01**	$3.15E+02$	**2.09E + 01**	$2.22E+02$
F3	$1.78E+02$	**9.66E + 01**	**4.76E + 02**	$9.95E+02$	**3.65E + 02**	$4.97E+02$	**7.50E + 01**	$2.60E+02$
F4	**1.04E + 02**	$1.28E+02$	**1.13E + 02**	$2.26E+02$	**1.09E + 02**	$2.14E+02$	**1.75E + 00**	$2.07E+01$
F5	**1.07E + 02**	$1.10E+02$	**1.15E + 02**	$2.26E+02$	**1.12E + 02**	$2.11E+02$	**2.62E + 00**	$2.42E+01$
F6	**1.06E + 02**	$1.06E+02$	**1.11E + 02**	$3.27E+02$	**1.08E + 02**	$2.40E+02$	**1.09E + 00**	$7.87E+01$
F7	**1.54E + 02**	$5.06E+02$	**3.42E + 02**	$7.07E+02$	**2.37E + 02**	$5.80E+02$	**5.08E + 01**	$6.09E+01$
F8	**1.04E + 02**	$1.42E+02$	**1.23E + 02**	$1.07E+03$	**1.14E + 02**	$5.81E+02$	**5.16E + 00**	$2.60E+02$

Table 4: Comparison between MRQPSO and QPSO on the running time.

Fun	Mean running time (ms)	
	MRQPSO	QPSO
F1	**52098**	60704
F2	**57123**	71371
F3	**58311**	72962
F4	**942736**	1811935
F5	**903781**	1739438
F6	**982763**	1855480
F7	**970225**	1945936
F8	**76354**	104103

engineering [39], chemistry [40], mechanics [41], medicine [42], and robotics [43], widely.

Generally, a nonlinear equation system could be described as [44]

$$\begin{aligned} e_1\left(\overrightarrow{x}\right) &= 0, \\ e_2\left(\overrightarrow{x}\right) &= 0, \\ &\vdots \\ e_M\left(\overrightarrow{x}\right) &= 0, \end{aligned} \tag{7}$$

where $\overrightarrow{x} \in \Omega^D$.

Table 5: Comparison between MRQPSO and QPSO on nonlinear equation systems.

		Fun 1		Fun 2		Fun 3	
		QPSO	MRQPSO	QPSO	MRQPSO	QPSO	MRQPSO
Value	Mean	$2.88E-04$	**3.77E − 05**	$1.37E-03$	**0.00E + 00**	**7.47E − 06**	$1.37E-05$
	Max	$4.71E-03$	**1.11E − 04**	$1.52E-02$	**0.00E + 00**	**2.09E − 05**	$3.53E-05$
	Min	**7.22E − 16**	$4.73E-06$	**0.00E + 00**	**0.00E + 00**	**9.10E − 07**	$1.28E-06$
Time	Mean	152362	**94077**	123579	**79761**	115000	**75855**
	Max	160675	**96551**	129658	**86161**	116815	**80671**
	Min	144371	**93284**	119379	**77345**	110133	**73275**

M is the number of equations. D is the dimension of variable. e_i is the ith equation in the system. Usually, at least one equation is nonlinear. If one solution could give all the equations in the system true statement, this solution is an optimal solution of this equation system.

In order to obtain the optimal solutions of a nonlinear equation system, an optimization problem like (8) or (9) could be constructed. Optimal solutions of (8) or (9) are the optimal solutions of nonlinear equation system (7)

$$\begin{aligned} \min \quad & \sum_{i=1}^{M} \left| e_i\left(\overrightarrow{x}\right)\right| \\ \text{where} \quad & \overrightarrow{x} \in \Omega^D \end{aligned} \tag{8}$$

or

$$\begin{aligned} \min \quad & \sum_{i=1}^{M} e_i^2\left(\overrightarrow{x}\right) \\ \text{where} \quad & \overrightarrow{x} \in \Omega^D. \end{aligned} \tag{9}$$

In this article, optimization problem like (8) is used to deal with one nonlinear equation system. And MRQPSO are compared with QPSO on Fun 1–Fun 3. The details of these three problems are listed as follows.

Fun 1:

$$\begin{aligned} \sum_{i=1}^{D} x_i^2 - 1 = 0, \\ \left|x_1 - x_2\right| + \sum_{i=3}^{D} x_i^2 = 0, \end{aligned} \tag{10}$$

where D is 20. Feasible area is $[-1, 1]^{20}$. Both equations are nonlinear. And 2 theoretical optimal solutions exist.

Fun 2:

$$\begin{aligned} x_1^2 + x_3^2 = 1, \\ x_2^2 + x_4^2 = 1, \\ x_5 x_3^3 + x_6 x_4^3 = 0, \\ x_5 x_1^3 + x_6 x_2^3 = 0, \\ x_5 x_1 x_3^2 + x_6 x_4^2 x_2 = 0, \\ x_5 x_3 x_1^2 + x_6 x_2^2 x_4 = 0, \end{aligned} \tag{11}$$

where D is 6. Feasible area is $[-1, 1]^6$. All the six equations are nonlinear. And infinite theoretical optimal solutions exist.

Fun 3:

$$\begin{aligned} \left(x_k + \sum_{i=1}^{D-k-1} x_i x_{i+k}\right) x_D - c_k = 0, \quad 1 \le k \le D-1, \\ \sum_{l=1}^{D-1} x_l + 1 = 0, \end{aligned} \tag{12}$$

where D is 20. Feasible area is $[-1, 1]^{20}$. In the system, one equation is linear and other nineteen equations are nonlinear. Infinite theoretical optimal solutions exist.

Some parameters and environment used on solving nonlinear equation system are listed as follows. Each algorithm is performed 20 times on each problem independently. All experiments are run for $2^{10} \times 1000$ function evaluations. The population size is 10. And search space of MRQPSO is partitioned into 2^{10} blocks. Results from QPSO and QPSO are compared and reported in Table 4.

Two aspects are considered in the comparisons. One is running time of the two algorithms. The other is the obtained minimized objective function value. The results are reported in Table 4. The better results are marked with blackbody.

From Table 5, it could be found that both the two algorithms have good performance on objective function value. On Fun 1 and Fun 2, MRQPSO have slight advantage than QPSO on mean and max value. On Fun 3 and min value, QPSO has advantage than QPSO. Here it seems that QPSO obtained much better result on min value of Fun 1. But actually, both of the solutions obtained by MRQPSO and QPSO are very close to the theoretical optimal solutions. But in MRQPSO, the computing resource is assigned on different areas. So during the latter search of MRQPSO, no computing resource as much as QPSO could be used to improve accuracy. This may be the reason for the worse performance of MRQPSO on min value.

But from the view of time cost, it is clear that MRQPSO outperformed QPSO on all the cases. And the advantage is significant. Because three virtual devices are used to evaluate solutions in feasible space at the same time, the computing task could be completed with less time.

6. Conclusion

This paper developed a MRQPSO algorithm and implemented serial QPSO into the MapReduce model, speculative parallelization, and distribution of QPSO. The proposed method was applied to solve the composition benchmark functions and nonlinear equation systems and got satisfactory solutions basically. Moreover, from the comparisons between MRQPSO and QPSO, the results showed us the parallel one outperformed the serial one on both quality of solution and time cost. MRQPSO can be considered as a suitable algorithm to solve large-scale and complex problems. In order to solve more complex practical problems, a cluster with more servers is needed to be constructed and used to test the performance of MRQPSO, which would be a further work.

Consent

Informed consent was obtained from all individual participants included in the study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61272279, 61272282, 61371201, and 61203303), the Program for New Century Excellent Talents in University (no. NCET-12-0920), the Program for New Scientific and Technological Star of Shaanxi Province (no. 2014KJXX-45), the National Basic Research Program (973 Program) of China (no. 2013CB329402), the Program for Cheung Kong Scholars and Innovative Research Team in University (no. IRT_15R53), and the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) (no. B07048).

References

[1] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan et al., "Distributed evolutionary algorithms and their models: a survey of the state-of-the-art," *Applied Soft Computing Journal*, vol. 34, pp. 286–300, 2015.

[2] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2016.

[3] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[4] Z. Fu, F. Huang, X. Sun, A. Vasilakos, and C.-N. Yang, "Enabling Semantic Search based on Conceptual Graphs over Encrypted Outsourced Data," *IEEE Transactions on Services Computing*, no. 99, pp. 1–1, 2016.

[5] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. E98B, no. 1, pp. 190–200, 2015.

[6] Q. Liu, W. Cai, J. Shen, Z. Fu, X. Liu, and N. Linge, "A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment," *Security and Communication Networks*, vol. 9, no. 17, pp. 4002–4012, 2016.

[7] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments," *Knowledge-Based Systems*, vol. 115, pp. 123–132, 2017.

[8] Z. Fu, X. Sun, S. Ji, and G. Xie, "Towards efficient content-aware search over encrypted outsourced data in cloud," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, (IEEE INFOCOM '16)*, San Francisco, Calif, USA, April 2016.

[9] F. M. Johar, F. A. Azmin, M. K. Suaidi et al., "A review of genetic algorithms and parallel genetic algorithms on graphics processing unit (GPU)," in *Proceeding of the 2013 IEEE International Conference on Control System, Computing and Engineering, (ICCSCE '13)*, pp. 264–269, December 2013.

[10] Q. Yu, C. Chen, and Z. Pan, "Parallel genetic algorithms on programmable graphics hardware," in *Proceeding of the International Conference on Advances in Natural Computation*, vol. 3612, pp. 1051–1059, Springer-Verlag.

[11] S. Tsutsui and N. Fujimoto, "Solving quadratic assignment problems by genetic algorithms with GPU computation," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '09)*, pp. 2523–2530, Montreal, Canada, July 2009.

[12] P. Pospíchal, "GPU-based acceleration of the genetic algorithm," *Gecco Competition*, 2009.

[13] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[14] S. Aridhi, P. Lacomme, L. Ren, and B. Vincent, "A MapReduce-based approach for shortest path problem in large-scale networks," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 151–165, 2015.

[15] T. G. Addair, D. A. Dodge, W. R. Walter, and S. D. Ruppert, "Large-scale seismic signal analysis with Hadoop," *Computers and Geosciences*, vol. 66, no. 2, pp. 145–154, 2014.

[16] X. Li, J. Song, F. Zhang, X. Ouyang, and S. U. Khan, "MapReduce-based fast fuzzy c-means algorithm for large-scale underwater image segmentation," *Future Generation Computer Systems*, vol. 65, pp. 90–101, 2016.

[17] F. Wang, X. Meng, and Y. Zhang, "An adaptive user preferences elicitation scheme for location recommendation," *Chinese Journal of Electronics*, vol. 25, no. 5, pp. 943–949, 2016.

[18] A. Verma, X. Llorà, D. E. Goldberg, and R. H. Campbell, "Scaling genetic algorithms using MapReduce," in *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications (ISDA '09)*, pp. 13–18, Pisa, Italy, December 2009.

[19] C. Jin, C. Vecchiola, and R. Buyya, "MRPGA: an extension of MapReduce for parallelizing genetic algorithms," in *Proceedings of the IEEE 4th International Conference on eScience (eScience '08)*, pp. 214–221, Indianapolis, Ind, USA, December 2008.

[20] X. Llorà, A. Verma, R. H. Campbell, and D. E. Goldberg, "When huge is routine: scaling genetic algorithms and estimation of distribution algorithms via data-intensive computing," *Studies in Computational Intelligence*, vol. 269, pp. 11–41, 2010.

[21] K. Tagawa and T. Ishimizu, "Concurrent differential evolution based on MapReduce," *International Journal of Computers*, vol. 4, no. 4, pp. 161–168, 2010.

[22] C. Zhou, "Fast parallelization of differential evolution algorithm using MapReduce," in *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, (GECCO '10)*, pp. 1113–1114, July 2010.

[23] B. Wu, G. Wu, and M. Yang, "A MapReduce based ant colony optimization approach to combinatorial optimization problems," in *Proceedings of the 8th International Conference on Natural Computation (ICNC '12)*, pp. 728–732, May 2012.

[24] A. W. McNabb, C. K. Monson, and K. D. Seppi, "Parallel PSO using MapReduce," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation, (CEC '07)*, pp. 7–14, September 2007.

[25] Y. Wang, Y. Li, Z. Chen, and Y. Xue, "Cooperative particle swarm optimization using MapReduce," *Soft Computing*, pp. 1–11, 2016.

[26] Y. Li, Z. Chen, Y. Wang, and L. Jiao, "Quantum-behaved particle swarm optimization using mapreduce," *Bio-Inspired Computing—Theories and Applications*, vol. 682, pp. 173–178, 2016.

[27] J. Li, J. Dang, and Y. Wang, "Medical image segmentation algorithm based on quantum clonal evolution and two-dimensional tsallis entropy," *Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao/Journal of Computer-Aided Design and Computer Graphics*, vol. 26, no. 3, pp. 465–471, 2014.

[28] O. P. Patel and A. Tiwari, "Novel quantum inspired binary neural network algorithm," *Sādhanā*, vol. 41, no. 11, pp. 1299–1309, 2016.

[29] L. Jiao, Y. Li, M. Gong, and X. Zhang, "Quantum-inspired immune clonal algorithm for global optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 5, pp. 1234–1253, 2008.

[30] Y. Li, R. Xiang, L. Jiao, and R. Liu, "An improved cooperative quantum-behaved particle swarm optimization," *Soft Computing*, vol. 16, no. 6, pp. 1061–1069, 2012.

[31] G. Zhang, "Quantum-inspired evolutionary algorithms: a survey and empirical study," *Journal of Heuristics*, vol. 17, no. 3, pp. 303–351, 2011.

[32] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the IEEE 2004 Congress on Evolutionary Computation*, pp. 325–331, 2004.

[33] J. Sun, W. Fang, X. Wu, V. Palade, and W. Xu, "Quantum-behaved particle swarm optimization: analysis of individual particle behavior and parameter selection," *Evolutionary Computation*, vol. 20, no. 3, pp. 349–393, 2012.

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks (ICNN '95)*, pp. 1942–1948, Perth, Australia, 1995.

[35] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 69–73, Anchorage, Alaska, USA, May, 1998.

[36] F. van den Bergh, *An analysis of particle swarm optimizers [Ph.D. dissertation]*, Department of Computer Science at the University of Pretoria, Pretoria, South Africa, 2002.

[37] J. J. Liang, B. Y. Qu, P. N. Suganthan, and G. Alfredo, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," Tech. Rep. 201212, Computational Intelligence Laboratory, Singapore; Zhengzhou University, Zhengzhou, China; Nanyang Technological University, January 2013.

[38] C. Hipp and M. Plum, "Optimal investment for insurers," *Insurance: Mathematics and Economics*, vol. 27, no. 2, pp. 215–228, 2000.

[39] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[40] C. Patrascioiu and C. Marinoiu, "The applications of the nonlinear equations systems algorithms for the heat transfer processes," in *Proceedings of the 12th WSEAS International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems (MAMECTIS '10)*, pp. 30–35, 2010.

[41] B. P. Mann and N. D. Sims, "Energy harvesting from the nonlinear oscillations of magnetic levitation," *Journal of Sound and Vibration*, vol. 319, no. 1-2, pp. 515–530, 2009.

[42] N. Filipovic, Z. Teng, M. Radovic, I. Saveljic, D. Fotiadis, and O. Parodi, "Computer simulation of three-dimensional plaque formation and progression in the carotid artery," *Medical & Biological Engineering & Computing*, vol. 51, no. 6, pp. 607–616, 2013.

[43] C. L. Collins, "Forward kinematics of planar parallel manipulators in the Clifford algebra of P2," *Mechanism and Machine Theory*, vol. 37, no. 8, pp. 799–813, 2002.

[44] W. Song, Y. Wang, H.-X. Li, and Z. Cai, "Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 414–431, 2015.