International Journal of Grid and Distributed Computing
Vol. 9, No. 12 (2016), pp.63-74
http://dx.doi.org/10.14257/ijgdc.2016.9.12.06

Research on a New Genetic Algorithm Model in Cloud Computing

Song Li

Institute of Engineering, Mudanjiang Normal University
Mudanjiang 157011, China
songli1111@163.com

Abstract

As to the problems that can been solved by genetic algorithm in Big Data, genetic algorithm research has profound significance. To meet the need of social development, many researchers have been doing a lot of work in thought and theory of that and come up with specific thought and theory. In the paper, firstly look up papers about the implementation rationale, model and specific implementation of genetic algorithm in HADOOP. Summarise two implementation models of genetic algorithm in HADOOP, vertical strategy and horizontal strategy. And deeply analyze the two implementation models, their design and implementation. Based on above, come up with improved model of genetic algorithm in HADOOP, its detailed design and general implementation steps. The improved implementation model is tested by test data to verify its correctness and validity. Experimental results prove to be that the improved implementation model of this paper is correct and valid. Finally, the improved implementation model of this paper is applied to solving facility location of emergency system.

Keywords*: Cloud Computing, Big Data, HADOOP, Genetic Algorithm*

1. Introduction

Cloud computing provides a new IT resource supply way. To be simple, it provides a data information center [1-3], where computer resources are characteristic of self-government, scalable deployment, configuration, allocation, re-distribution and resource re-cycling, together with automatic installation and deployment of applications. Hardware, software and service finally constitute cloud computing [4-8]. It has hosting capabilities of various working loads, including bulk processing of jobs, user-based interactive application. With rapid deployment of physical machine or virtue machine, it completes deployment of the system quickly and improves its storage capacity. It supports redundant, self-restoring and highly loose coupling programming model as for working load to recover itself from multiple unavoidable hardware/software malfunction. It helps realize real-time monitoring of resources as to know better their usage; it re-allocates resources if necessary in order to balance resources [9-10].

From the part of problems which can be solved with genetic algorithm in big data environment, researches on genetic algorithm in cloud computing environment have far-reaching significances [11-12]. To adapt to social advancement, lots of researchers have conducted related theoretical exploration and learning, proposing the idea and methodology about solving big data problem with genetic algorithm in cloud computing condition. However, such idea and methodology are studied at a front stage, requiring further practices to improve and perfect them [13-14]. The main idea of genetic algorithm is: transform in coding form the problem of which the best solution is got in solution space into corresponding chromosome or individual in biological space; then apply biological space genetic mechanism like cross-over, mutation to make chromosome or

ISSN: 2005-4262 IJGDC

gene evolve genetically to make evolved chromosome with good features; next decode the chromosome and transform it to relative solution in solution space, letting the obtained solution gradually close to local or global optimal solution.

2. Implementation of Genetic Algorithm in Cloud Computing

2.1 Vertical Strategy Implementation Model of Genetic Algorithm

Vertical strategy implements genetic algorithm in Hadoop with the core concept: divide massive gene population into several segments and assign to Map/Reduce to implement; the whole evolution process of genetic algorithm is carried out at one Map/Reduce stage to enhance convergence and computing speed, i.e. clustering nodes in Hadoop platform own their respective populations. Vertical strategy's model separates the whole evolution of genetic algorithm into Map and Reduce stage. Input data only refers to population size and number of Map [15-16]. It is shown in Figure1.

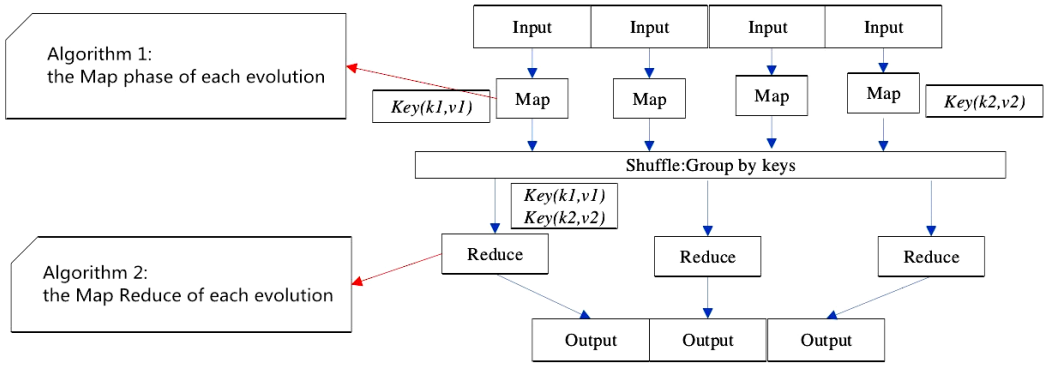


Figure 1. Vertical Strategy Implementation Model of Genetic Algorithm

The pseudo code of algorithm 1 and algorithm 2 are as follows in Figure1:

```
Algorithm 1:
Input data: population size popSize
void Map(key,value)
{
1. Current population algebra currentGen=0
2. Generating initial population pop
3. To calculate initial population fitness fitness
4. while current <  Last iteration do
5. To calculate the next generation gene population pop=nextGeneration (POP)
6. To calculate the population fitness of the next generation fitness
7. currentGen=currentGen+1
8.  end while
9. Find the optimal solution after evolution optimal
10.Output optimal solution to HDFS;

Algorithm 2:
void Reduce(key,values)
{
   1. for(optimal : values)
   2.Whether the optimal is the optimal solution
   3 .end for
```

2.2 Horizontal Strategy Implementation Model of Genetic Algorithm

The core thinking of horizontal strategy implementing genetic algorithm in Hadoop is: each evolutional task of genetic algorithm is assigned to Map/Reduce stage to carry out as to improve convergence and computing speed. Horizontal strategy's model divides the evolution of genetic algorithm into Map, Partitioner and Reduce stage. Of them, Map stage completes calculation of all genetic individual fitness and judges if there's better solution than current optimal one. If yes, replace the current optimal solution and save to HDFS; Partitioner stage divides randomly gene populations which are used as input data at Reduce stage; Reduce stage completes operations of genetic algorithm like selection, cross-over, and mutation [17]. It is shown in Figure2.

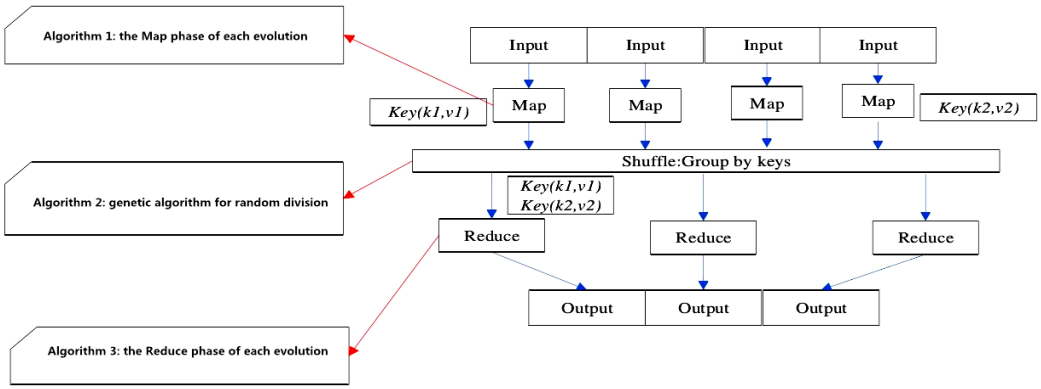


Figure 2. Horizontal Strategy Implementation Model of Genetic Algorithm

The pseudo code of algorithm 1, algorithm 2 and algorithm 3 are as follows in Figure2:

```
Algorithm 1:
void Map(key,value)
{
1 . Individual=IndividualRepresentation(key)
2.fitness=CalculateFitness(Individual)
3. Emit(Individual,fitness)
4. if fitness>max then
5. max=fitness
6. maxInd=Individual
7.end if
8. if  All the genetic individuals were treated then
9.Save the current optimal gene individual maxInd to HDFS
10. end if
}
Algorithm 2:
int getPartition(key,value,numReducers)
{
    1.return RandomInt(0,numReducers-1)
}
Algorithm 3:
void Reduce(key,values)
{
   1.While values.hasNext() do
  2. Individual=IndividualRepresentation(key)
   3. fitness=values.getValue()
```

```
    4. if processed<tSize then
    5.  tournArray[tSize+processed%tSize]=individual
    6. else
    7.  SelectAandCrossover()
    8. Mutation()
    9. end if
   10. processed=processed+1
   11. if   all of the individual genes are treated then
   12. for k=1 to tSize do
   13.SelectAandCrossover()
   14.  Mutation()
   15.  processed=processed+1
   16.end for
   17.end if
   18.end while
  };
```

3. Implementation Model of Improved Genetic Algorithm in Cloud Computing

3.1. Analyze Implementation Model of Existing Genetic Algorithm in Cloud Computing

The implementation model of vertical strategy's genetic algorithm has features like: the entire evolution is completed by one Map/Reduce task; clusters at each stage have own gene populations; the selection, cross-over and mutation operation of the genetic algorithm mainly happen at Map stage.

The implementation model of horizontal strategy's genetic algorithm has features as follows: each evolution is accomplished by a new Map/Reduce task; nodes in clusters commonly have one gene population, which is used as data communication medium between the last and next generation through HDFS file system; the selection, cross-over and mutation operation of this genetic algorithm mainly happen at Reduce stage.

We can see the implementation models of vertical strategy's and horizontal strategy's genetic algorithm have a common shortcoming: the selection, cross-over and mutation operation of genetic algorithm concentrate at Map or Reduce stage, limiting the parallelization degree of genetic algorithm in Hadoop; besides, the implementation model of horizontal strategy completes iteration each time by a new Map/Reduce task, with the aid of HDFS file system for data communication between the last and next generation. In that case, plentiful small files will generate; reading those files will lead to tremendous Seeks and Hopping from DataNode to DataNode to retrieve files, which is a very low-efficient access mode, reducing performance of the algorithm. Meanwhile, a great deal of HDFS files' read/write operations to decrease the performance of the method.

3.2. Implementation Model of the Improved Genetic Algorithm

Due to weakness of the implementation models of vertical and horizontal strategy's respective genetic algorithm, we need to present a new model strategy carrying forward strengths of those models, letting different Map/Reduce to complete the selection, cross-over and mutation of genetic algorithm and calculation of current optimal solution. In the course of each Map/Reduce task implementation, nodes possess their own gene population. The evolution of genetic algorithm every time is completed through new selection, cross-over and mutation and calculation of current optimal solution by

Map/Reduce chain, with Hadoop-based HBase as data communication medium between last and next generation. The proposed new strategy helps improve parallel degree of genetic algorithm in Hadoop and avoid lots of read-write operations of HDFS files, advancing on the whole the efficiency of genetic algorithm solving problems. The selection, crossover and mutation of genetic algorithm and the Map/Reduce design of the current optimal solution. Pseudo code is shown:

(1) the selection phase of the Map/Reduce

```
void Map(key,value)
{
1. pop=value
2. parents=select(pop)
}
void Reduce(key,values)
{
1: do not any processing directly to the output, as the next phase of the Map input
};
```

(2) the cross phase Map/Reduce

```
void Map(key,value)
{1. pop=value
 2. offsprings=cross(pop)
}
void Reduce(key,values)
{
1: do not any processing directly to the output, as the next phase of the Map input
};
```

(3) the mutation stage of Map/Reduce

```
void Map(key,value)
{1. pop=value
 2. offsprings=mutation(pop)
3. Set the key value of the output to a fixed value
};
```

(4) calculated the current generation of the optimal solution of the Map/Reduce

```
void Map(key,value)
{
1. pop=value
2. To calculate the optimal solution of the population of genes in the
   contemporary, and to deposit the HBase database
 };
```

3.3 Map/Reduce Communication Scheme of Implementation Model of Improved Genetic Algorithm at Each Stage

3.3.1 Structural Design of Data used for Map/Reduce Interaction at each Stage. Considering features of the implementation model of the proposed improved genetic algorithm and general approach for Map/Reduce solving problem in Hadoop, the improved method's implementation model needs preserve these data: gene population size popSize, gene population data pops (including gene bit data geneBits and gene individual fitness), current population optimalGene and according optimalFitness, current popGeneration. The specific data structure format is as follows:

```
Struct
{
type  popGeneration;
   type  popSize;
   struct pops
  {
          type geneBits;
          type fitness;
}
      type optimalGene;
      type  optimaFitness;
};
```

3.3.2 Persistent Data to Structural Design of Data Table in HBase. According to devised data structure in (3.3.1) and characteristics and design requirement of HBase, data table structure in HBase is row_id and row containing a few column clusters. The specific description of the cluster is shown in table1.

Table 1. The Cluster Information of Data Sheet in the Structure

Cluster name	Cluster name type	Cluster type	Cluster description
popGeneration	String	Byte	Population generation
popSize	String	Byte	Population size
pops	String	Byte	Gene population
optimalGene	String	Byte	Current population optimal individual

3.3.3 Data Protocol for Map/Reduce Interaction at Each Stage.

(1) Data protocol for interaction between initial operation period and that when current population's optimal solution is computed.

Generally, Map/Reduce output data format is <key,value> pair; so the output data format is also <key,value> at initialization stage; the difference is key here is a fixed figure like number 0, 1, 2, 3 etc or fixed character string; while value means object of data structure defined in part (3.3.1).

The input at the stage when calculating current population's optimal solution is input at initialization operation stage, data format, key and value keeping the same.

(2) Data protocol for interaction between the stage when current population's optimal solution is computed and selection stage

The output data and data format at current population's optimal solution calculating stage are the same with that at initialization operation stage.

At selection stage, input data format is <key,value>, where key is valued with 0, 1, 2, …, n (n is defined differently for different solving problem); value is always value of output data <key,value> at the stage of calculating current population's optimal solution.

(3) Data protocol for interaction among selection, cross-over and mutation period

At the stage of genetic algorithm selection, cross-over and mutation, data used are all output at the selection stage; so the interactive data protocol among them is similar.

At selection stage, n gene populations in <key,value> utilize selection operator for selection operation to produce relative <key,value> as input or output data at cross-over and mutation stage.

(4) Data protocol for interaction between mutation stage and selection stage in next evolution

At mutation stage, combine n resultant gene individuals of <key,value> after mutation operation with gene population of last generation; the merging rule is survival of the fittest in the theory of evolution: substitute inferior gene individuals in parent population with superior gene individuals in offspring population; the generation number of merged gene population is equal to the number of last population plus 1, which is stored in HBase table designed in part (3.3.2), as input data at selection stage in next evolution.

(4)Linking of various Map/Reduce stages of the genetic algorithm

Solution-seeking problem of the genetic algorithm generally needs iterations, requiring evolution of gene population for several times; moreover, the implementation model of the improved genetic algorithm in Hadoop assigns to different Map/Reduce the stage when calculating current population's optimal solution, selection stage, cross-over stage and mutation stage. It's required to link up each Map/Reduce stage. Hence we give the plan for linking various Map/Reduce stages. It is shown in Figure3.

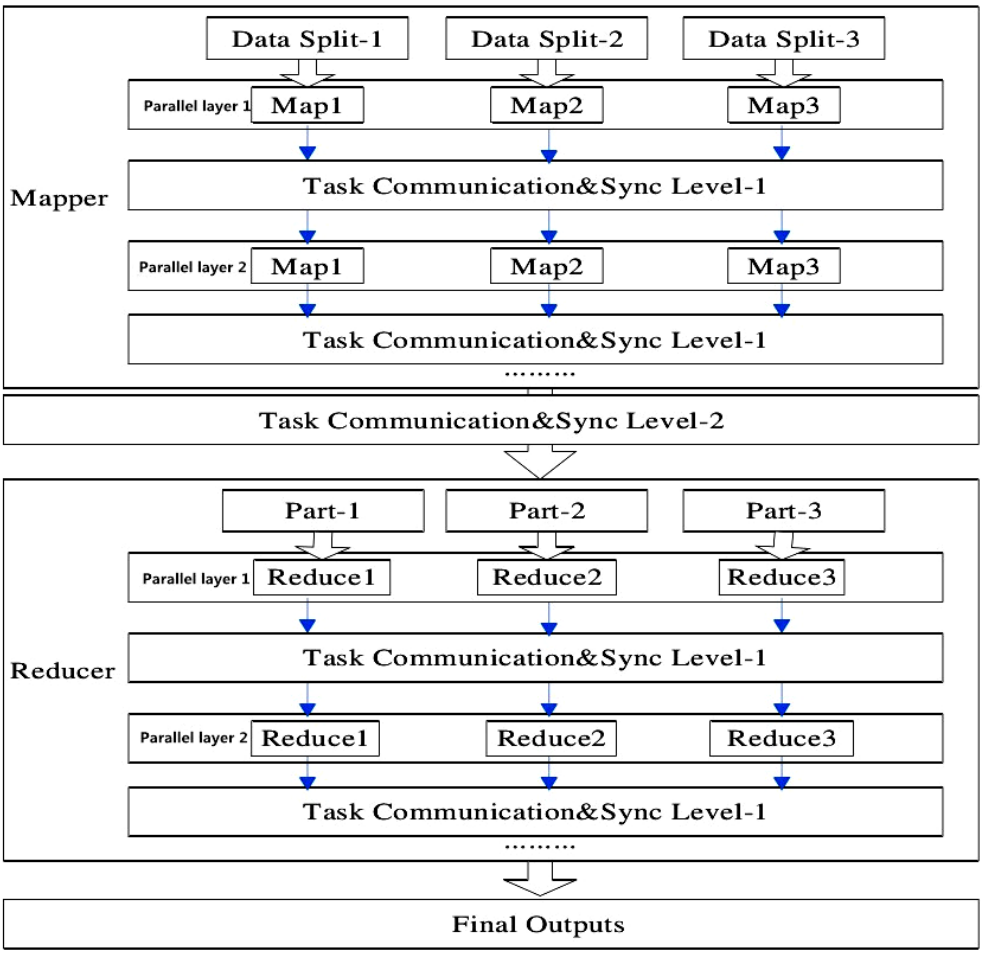


Figure 3. Multi Map/Reduce co Grounding Principle Diagram

3.4 Steps for Solution Problem by the Implementation Model of the Improved Genetic Algorithm

According to ideas of the implementation model of the improved genetic algorithm and the genetic algorithm, the former realizes solution problem in following steps:

(1) Set all parameters of genetic algorithm including genetic operator's parameters like popSize, generationNum;

(2) Design data structure to generate initial population

(3) Calculate population fitness; acquire optimal solution of current population and save it; duplicate current population into n copies as output data at this stage;

(4) Perform selection operation of n gene populations output in step (3) to form results after n selection operations as output data at this stage;

(5) As per operators perform cross-over operation of gene populations output in step (4) after n selection operations to get results of n cross-over operations as output data at this stage;

(6) As per operators perform mutation operation of gene populations output in step (5) after n cross-over operations to get results of n mutation operations as output data at this stage;

(7) Merge results after n mutation operations output in step (6) and parent population, in the idea of selecting the superior and eliminating the inferior; replace bad gene individuals in parent population with good ones in child group, with gene population size identical;

(8) Determine if target condition or iteration condition is met; if yes, terminate evolution; otherwise, skip back to step (3) for next evolution.

4. Experimental Analysis and Results

4.1Experimental Data and Experimental Platform

4.1.1 Experimental Data. Job shop scheduling problem (JSSP) is used to solve the problem [18]. Experimental data is from Beasley OR-Library J. website. The experimental data are shown in table2.

Table 2. JSSP Problem Instance Data

Name	Jobs	Machine	OptimalMakespan
FT10	10	10	820

4.1.2 Experimental platform. The experimental platform consists of 5 sets of desktop processors, each of which is configured as (i7-4 Cores 2.8 GHz, 8GB DDR3 RAM, 500GB HDD).

4.1.3 Experimental parameter setting. The choice of operation used cross era elite selection strategy, crossover probability $P_c = 0.7$, mutation probability $P_m = 0.01$.

4.2 Experimental results and analysis

The implementation model of the improved genetic algorithm realizing the solution FT10. It is shown in Figure4.

Results prove the correctness of the implementation model of the improved genetic algorithm; also experiment validates the effect of the population size on the optimal solution acquired by the improved algorithm's implementation model when

population size $P=10^5, P=10^6, P=10^7$, which suggests that the model-based genetic algorithm has good adaption to solution problem of bigger and bigger scale.

Secondly, we compare the improved genetic algorithm implementation model and that of vertical strategy's method. Table3, Table4, Table5 and Table6 show the experimental results by two methods solving FT10.

When the genetic algorithm realized by the improved genetic algorithm's implementation model seeks solution of FT10, it takes 236,77 seconds when evolution iteration is 30 and population size is 10^5; Map takes 214.58 seconds, 89.54% of the total; Reduce takes 10.93 seconds, 4.07% of the total; Overhead takes 14.89 seconds, 6.89% of the total. When the method realized by the implementation model of vertical strategy's genetic algorithm solves FT10, it takes 367.09 seconds when evolution iteration is 30; Map takes 341.66seconds, 94% of the total; Reduce takes 8.97 seconds, 3.02% of the total; Overhead takes 18.89 seconds, 4.89% of the total. On the whole, the improved genetic algorithm implementation model performs better than the other. To compare them more subjectively, Table 6 lists out mean Map time, mean Reduce time and mean Overhead time by the former is less than the later when both solve FT10 with evolution iteration=30, demonstrating correctness and effectiveness of the implementation model of the proposed genetic algorithm.

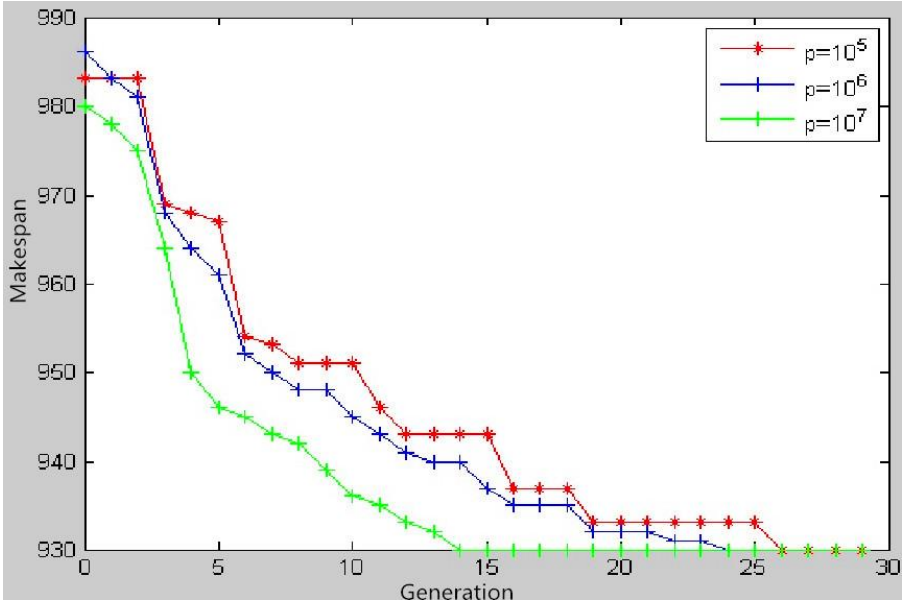


Figure 4. The Implementation Model of the Improved Genetic Algorithm Realizing the Solution FT10

Table 3. Vertical Strategy Genetic Algorithm for Solving FT10 Time and Map/Reduce Calculation Time and Percentage

Task completion time	Map()	Reduce()	Overhead
367.09	341.66	8.97	18.89
100%	94%	3.02%	4.89%

Table 4. Improved Genetic Algorithm Model of the Genetic Algorithm to Solve the Completion Time of FT10 and the Entire Map, Reduce Calculation time and Percentage

Task completion time	Map()	Reduce()	Overhead
236,77	214.58	10.93	14.89
100%	89.54%	4.07%	6.89%

Table 5. Improved Genetic Algorithm to Achieve the Model and the Vertical Strategy of the Genetic Algorithm to Achieve the Genetic Algorithm to Complete the Entire Task using the Reduce, Map and Overhead Number

Model class	Map number	Reduce number	Overhead number
Improved algorithm model	10556	678	140
Original algorithm model	5422	378	40

Table 6. Improved Genetic Algorithm to Achieve the Model and the Vertical Strategy of the Genetic Algorithm to Achieve the Genetic Algorithm to Complete the Entire Task Map, Reduce, and the Average Running Time of Overhead

Model class	Average Map time	Average Reduce time	Average Overhead time
Improved algorithm model	0.023422	0.018887	0.115874
Original algorithm model	0.068772	0.023334	0.387476

5. Conclusion

In this paper, through in-depth investigations on two implementation models of genetic algorithm in Hadoop cloud computing platform and their respective design principle and realization technology, the paper concluded their shortcomings. With Hadoop infrastructure and relative technology, the paper developed the implementation model of the improved genetic algorithm here in Hadoop cloud computing platform, which was discussed in details and general design and implementation plan of the model was provided for each stage. Moreover, steps involving the solving problem by the improved genetic algorithm model were listed. Finally, through numerical experiments, it proved the correctness and effectiveness of the implementation model of the proposed method in Hadoop cloud computing platform. Experimental results indicated that the model here is correct and useful, living up to the expected research objective of the paper.

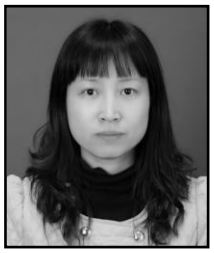
Acknowledgement

This work was supported by Mudanjiang Normal College science and technology research projects. No. QY2014002

References

[1] F.-H. Huang, "Cloud environment research of hyperspectral image classification based on parallel support vector machines", Fujian Normal University, **(2014)**.
[2] Z.-G. Ren, "Massive data processing in the field of cloud computing and its key technology oriented research", Nanjing University of Science and Technology, **(2013)**.
[3] Y.-Z. Han, "Study on reactive power optimization control system of oil field distribution network based on cloud computing", Harbin Institute of Technology, **(2014)**.
[4] Z.-X. Huang, "Research of intelligent power grid dispatching system based on cloud computing", Tianjin University of Technology, **(2014)**.
[5] D. Wei, "Research on task scheduling based on dual fitness genetic algorithm in cloud environment", Dalian Maritime University, **(2014)**.
[6] Z.-Y. Zhong, "Genetic and text classification research based on distributed feedback", Beijing University of Posts and Telecommunications, **(2014)**.
[7] M.-Y. Pan, "Design of data processing system based on Hadoop", Hebei University of Engineering, **(2014)**.
[8] Z.-J. Xu, "Research on parallel clustering algorithm of data stream", Qufu Normal University, **(2015)**.
[9] D.-J. Liang, "Research on task scheduling strategy in the cloud environment", Beijing University of Technology, **(2015)**.
[10] X.-X. Ren, "Job scheduling research based on Hadoop platform", Tianjin Normal University, **(2011)**.
[11] H.-L. Shi, "Cloud computing task scheduling research", Nanjing University of Science and Technology, **(2012)**.
[12] G.-H. Zhang, "Hadoop parallel genetic algorithm research and application in the emergency facility location", Internet world, vol. 8, **(2013)**, pp. 11-14.
[13] G.-H. Zhang, "Research on parallel genetic algorithm and its application in emergency facility location", Information technology and information technology, no.16901, **(2014)**, pp. 81-85.
[14] L. Lu, "Research on job scheduling algorithms in cloud environment", Nanjing University of Science and Technology, **(2013)**.
[15] D. Keco and A. Subasi, "Parallelization of genetic algorithms using Hadoop MapReduce", Southeast Europe Journal of Soft Computing, **(2012)**, pp. 1-5.
[16] C. Jin, C. Vecchiola and R. Buyya, "An extension ofMap-Reduce for parallelizing genetic algorithms", IEEE Fourth International Conference on eScience, eScience, **(2008)**.
[17] A. Verma, X. Llor`a, D. E. Goldberg and R. H. Campbell, "Scaling Genetic Algorithms Using MapReduce", International Conference, **(2009)**.
[18] http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

Author



Song Li, She is an associate professor at Institute of Engineering of Mudanjiang Normal University. She is in the research of computer application.