Distributed hybrid rice algorithm based on hadoop

Lingyu Yan, Qixing Luo, Xingpeng Deng, Hongwei Chen and Zhiwei Ye

May 25, 2018

Distributed hybrid rice algorithm based on hadoop

Lingyu Yan, Qixing Luo, Xingpeng Deng, Hongwei Chen, Zhiwei Ye*
School of Computer Science
Hubei University of Technology
Wuhan, China
yanlingyu@hbut.edu.cn, qxluo@163.com, xpdeng@163.com, chw2001@sina.com, weizhiye121@163.com

***Abstract*—Hybrid Rice Optimization Algorithm(HRO) is a novel swarm intelligence optimization algorithm based on heuristic evolutionary computation, which is inspired by the cultivation process of three line hybrid rice. In this paper, we design and implement distributed hybrid rice algorithm based on Hadoop, proposing two distributed schemes. Through experiment, we select the better one for further research. By tuning the size of swarm size, we implement comparison experiment on our scheme and traditional HRO. Experiment demonstrate that our scheme shows better performance. What's more, with the increase of nodes in Hadoop scheme, our scheme shows obvious advantage.**

***Index Terms*—Hybrid Rice Optimization Algorithm(HRO), Hadoop, Distributed HRO**

I. Introduction

Three-line algorithm in hybrid rice cultivation divides rice gene into three parts, sterile line, maintainer line, restorer line. The sterile line is also called wild abortive, which is unable to breed by itself. The maintainer line is used to hybridize the sterile line. The restorer line maintains themselves. The sterile line hybridized with the maintainer line is an evolutionary process, and the restorer line maintains themselves is a swarm search process.

HRO[1,2,3] takes the advantage of evolutionary computing and swarm intelligence from three line algorithm. In each iteration, the population of HRO is sorted by the fitness from best to worst. Then the population is divided into three parts, namely the worst 1/3 of the group are selected as sterile line, the best 1/3 are employed as maintainer line and the other 1/3 acts as restorer line. Three measures may be taken to produce the next generation, namely hybridization, renew and selfing. (1) Hybrid Operation: select two species of the group with the great difference in characters. The hybrid operation between the two species may generate a new species with the advantage of parents. (2) Self intersection operation: it is the behavior of restorer line. This behavior makes all the restorer close to the best one. The genes of the selected restorer will close to the best one of all. (3) Renewal operation: The renewing behavior is to reset the restorer which haven't updated any member of sterile line. The genes of the selected maintainer will be changed to a random value in the range of search space. If hybridization operation does not meet stop condition, it will continue to iterate, or else renews the restorer line.

Although HRO is simple and easy to be implemented, with the coming of big data era, the efficiency of HRO need to be improved to fulfill the high efficiency requirement among large data scale. Its drawback mainly derives from two aspects as follows: 1) HRO is an evolutionary algorithm which involves large amount of individual computation, so, it always leads to high time complexity with large size of individual amount in swarm and large number of iteration. 2) HRO involves many parameters which is always set empirically, leading to high randomness of HRO. Therefore, only one times of execution of HRO merely yields to satisfactory results, while many times of execution is needed to provide candidates for satisfactory results, increasing the work amount of experiments.

To improve the adaptivity of HRO, in this paper, we propose a distributed HRO, which considers the advantage of MapReduce distributed parallel computing framework[25]. Utilizing the high computation speed of distributed parallel computing, we solve the high time consumption problem of HRO.

The remainder of this paper is organized as follows. In section 2, we present the distributed HRO. In section 3, extensive experiments are conducted to demonstrate the effectiveness and efficiency of the proposed algorithm. Finally, we draw a conclusion in section 4.

II. Approach

To utilize the high computation speed of distributed parallel computing, we need to match the HRO with Map and Reduce, and design a scheme for every Map-Reduce to be implemented individually. Here we propsed two scheme for Distributed HRO, shallow Distributed HRO and deep Distributed HRO.

A. Shallow Distributed HRO

We divide the swarm into several parts, and every parts is passed to different Map and Reduce operation. That means, each node in distributed HRO has its own sub-swarm. Fig. 1 shows the framework of shallow Distributed HRO.

(1) Swarm initialization

In this stage, we need to divide the whole swarm into several parts, making sure the number N of initialization swarm. It means we implement N times of independent HRO in one Distributed HRO in parallel, and every node in the scheme implements one time of HRO respectively. Since each sub-swarm corresponds to one Map and Reduce operation and initialization of each sub-swarm is implemented on the Mapper, so the number N in Swarm initialization is also the number of Mapper.

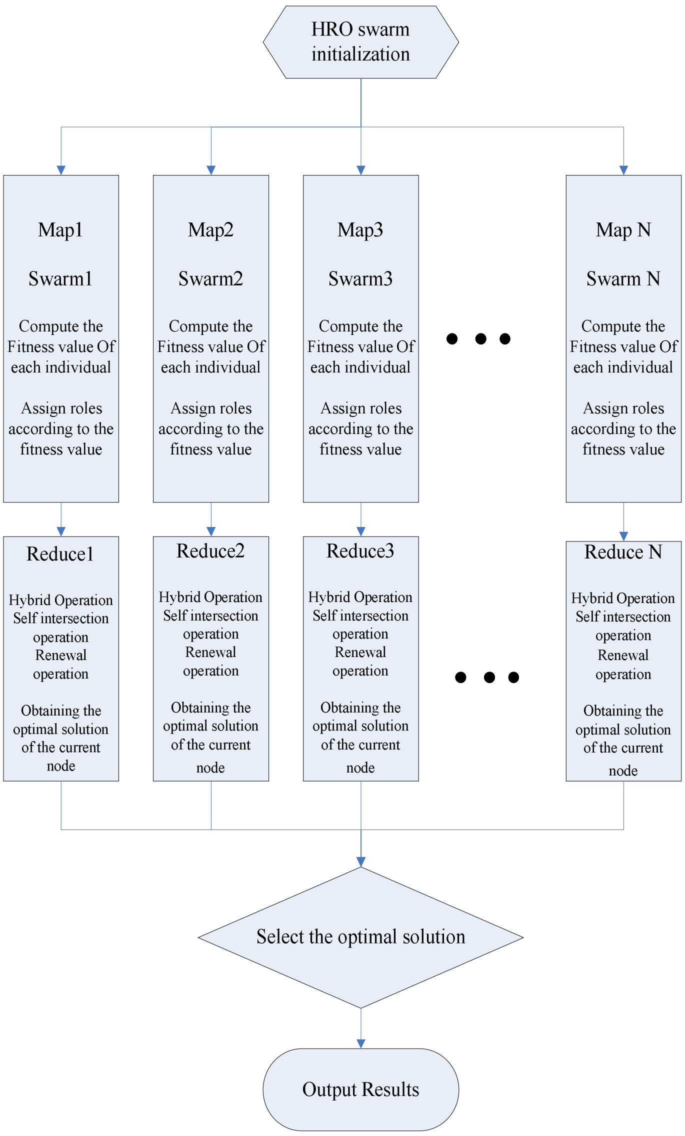

Fig. 1 Framework of Shallow Distributed HRO

(2) Map operation stage

In this stage, we need to initialize each sub-swarm, including evaluating fitness function, calculating fitness for each sub-swarm, and assigning roles according to the fitness value. Then we the maximum of hybrid times, and iteration parameter, and delivering the results to Reduce.

(3) Reduce operation stage

In this stage, we implement the swarm evolutionary of HRO. According to the role assignment from Mapper, we implement Hybrid Operation, Self intersection operation, Renewal operation, and select the optimal solution in one iteration. Once the iteration achieves the preset parameter, we get the optimal solution of the current node.

(4) Output results stage

Comparing the N optimal solution of each node, we get the global optimal solution according to the rank of fitness value.

B. Deep Distributed HRO

We implement each evolutionary computation on different Map/Reduce operation. It divides one evolutionary computation task into two Map/Reduce stage. The first Map/Reduce rank the individuals according to fitness value, the second Map/Reduce assign roles and implement evolutionary computation. The framework of Deep Distributed HRO is shown in Fig. 2.

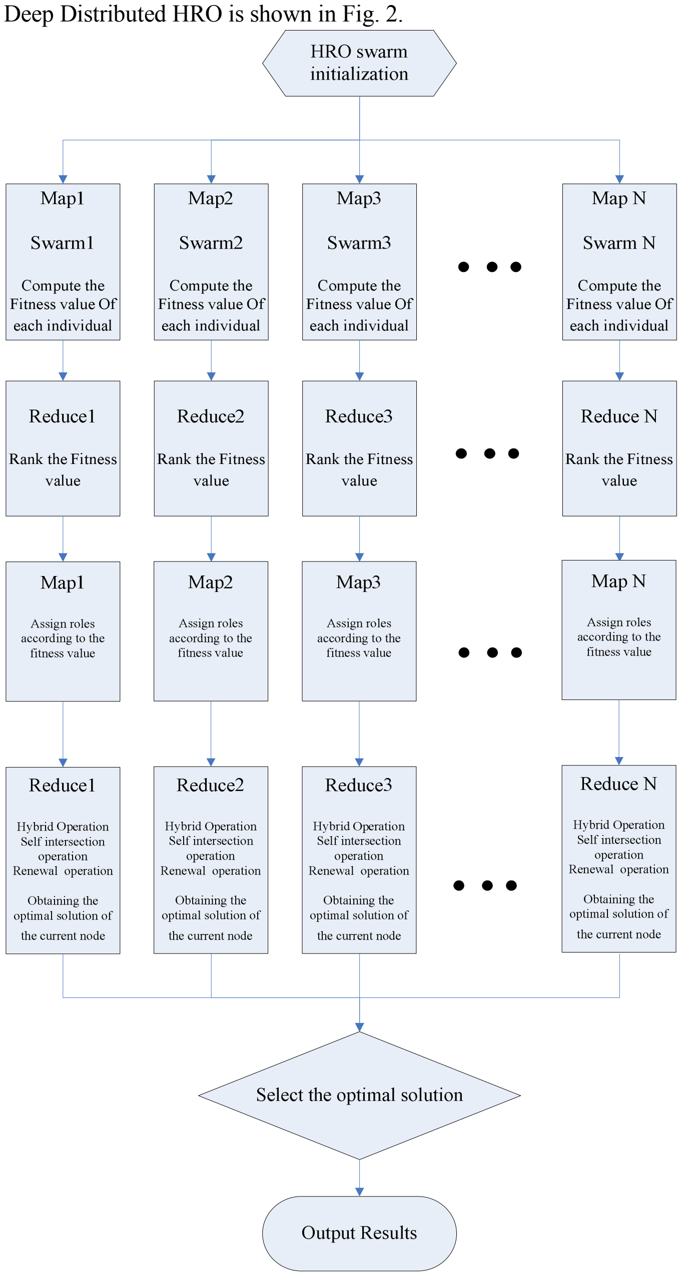

Fig. 2 Framework of Deep Distributed HRO

(1) Swarm initialization

This stage is the same as the shallow distributed HRO.

(2) Evolution of the swarm

Using first Map part to calculate fitness value, then rank the individuals in Reduce part according to the fitness value. Using the second Map part to assign roles to individuals, then implement Hybrid Operation, Self intersection operation, Renewal operation. Once the iteration achieves the condition of convergence, we get the optimal solution.

(3) Transmission of intermediate results

Transmit the fitness value of Mappers to corresponding Reducer to get fitness rank. Then transmit the rank list to the second Mapper to assign roles, and do Hybrid Operation, Self intersection operation, Renewal operation to get the optimal solution.

(4) Output results stage

Comparing the N optimal solution of each node, we select the global optimal solution according to the rank of fitness value.

III. EXPERIMENT

A. Experiment environment

We implement our scheme on hadoop platform, which is constructed on two PC machines, including five hadoop cluster nodes. One PC is considered as master node, another PC supports four slave nodes. Topology diagram of distributed system is shown in Fig. 3.

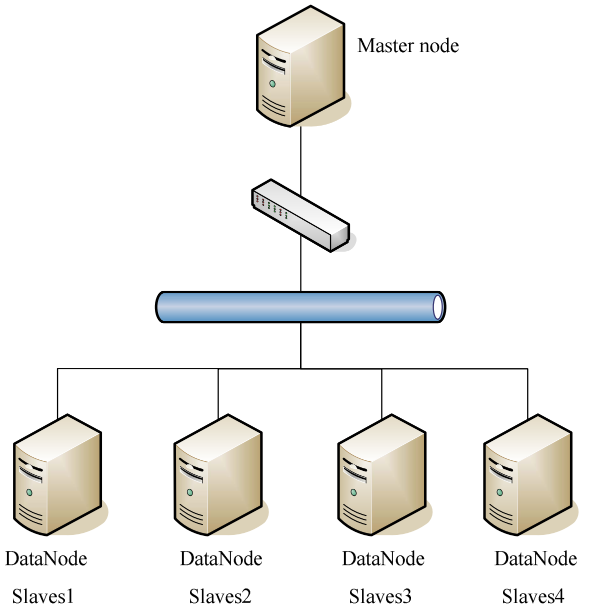

Fig. 3 Topology diagram of distributed system

B. Experiment results

Firstly, we compare the Shallow distributed HRO and Deep Distributed HRO using F11 test function in CEC2015. Fig. 4 shows the time consumption comparison of the two scheme.

It is obvious that Deep distributed HRO is superior to Shallow distributed HRO with the increase of swarm size. So we choose the Deep distributed HRO to implement our distributed HRO, and do comparison experiment with traditional HRO.

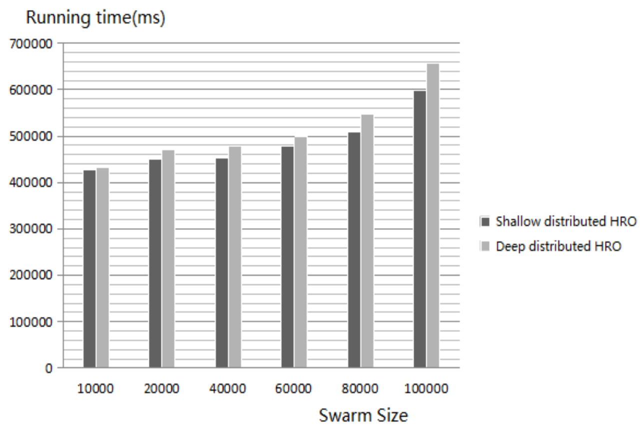

Fig. 4 Time consumption comparison of Shallow distributed HRO and Deep Distributed HRO.

To evaluate the performance Distributed HRO, we adopt test function F3 and F11 in CEC2015 to test the optimization ability and calculation speed. In the process of initalization, we set the size of swarm as 10000, 20000, 40000, 60000, 80000, 100000 respectively. The number of iteration is set to be 100.

Fig. 5 shows the time consumption comparison of HRO and Distributed HRO using F3 as test function. Fig. 6 shows the time consumption comparison of HRO and Distributed HRO using F11 as test function.

From the two comparison figures, we find that the running time increses with the increase of Swarm size. When the Swarm size is small, tradition HRO is more efficient than Distributed HRO, this derives from the communication time cost and the read/write time cost in distributed system. However, when the Swarm size is large, Distributed HRO shows better performance than tradition HRO.

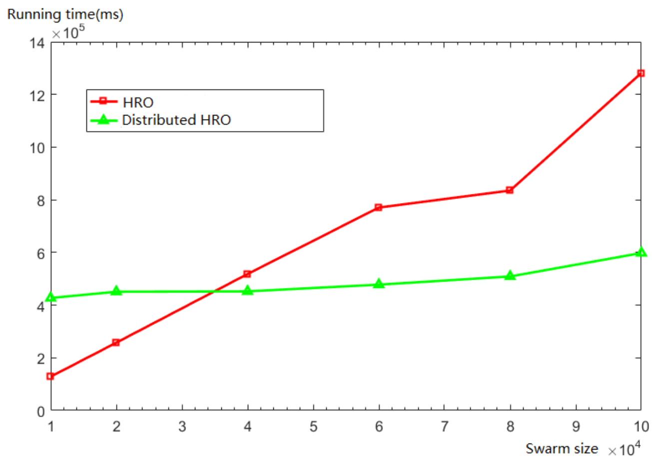

Fig. 5 Time consumption comparison of HRO and Distributed HRO using F3 as test function.

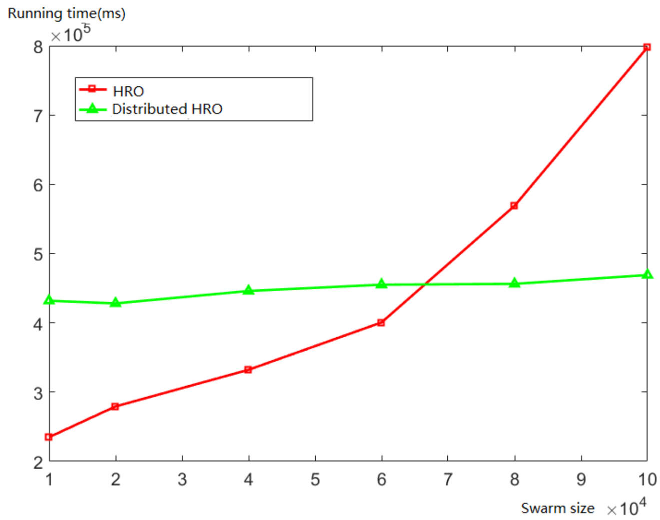

Fig. 6 Time consumption comparison of HRO and Distributed HRO using F11 as test function.

IV. CONCLUSION

In this paper, we design and implement distributed hybrid rice algorithm based on Hadoop. First, we propose two distributed schemes, Shallow distributed HRO and Deep distributed HRO. Through experiment, we select Deep distributed HRO for further research. By tuning the size of swarm size, we implement comparison experiment on our scheme and traditional HRO. Experiment demonstrate that our scheme shows better performance on both solution searching and running efficiency.

ACKNOWLEDGMENT

Thanks for the funding supported by the National Natural Science Foundation of China (No. 61502155, No. 61602161, No. 61602162, No. 61772180), Natural Science Foundation of Hubei University of Technology(No. BSQD15029), Hubei Undergraduate Training Program for Innovation and Entrepreneurship (No. 201702112038), Collaborative education project of Industry university cooperation of Ministry of Education(No. 201701003076).

REFERENCES

[1] Ye Z, Ma L, Chen H. “A hybrid rice optimization algorithm”, International Conference on Computer Science & Education. Japan, IEEE, 2016, pp169-174.

[2] Liu C, Wang C, Hu J, et al. “’Improved K-means algorithm based on hybrid rice optimization algorithm”, IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. IEEE, 2017,pp 788-791.

[3] Hou Y, Ye Z, Xu W, et al. “Remote sensing textual image classification based on extreme learning machine and hybrid rice optimization algorithm”, IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. IEEE, 2017, pp 777-781..

[4] Aridhi S, Lacomme P, Ren L, et al. “A MapReduce-based approach for shortest path problem in large-scale networks”[J]. Engineering Applications of Artificial Intelligence, 2015, 41(C), pp151-165.

[5] Mohammed E A, Far B H, Naugler C. “Applications of the MapReduce programming framework to clinical big data analysis: current landscape and future trends”. Biodata Mining, 2014, 7(1),pp 2..

[6] Liu X, Wang X, Matwin S, et al. “Meta-MapReduce for scalable data mining”. Journal of Big Data, 2015, 2(1), pp14.

[7] Lemnaru C, Cuibus M, Bona A, et al. “A Distributed Methodology for Imbalanced Classification Problems“, International Symposium on Parallel and Distributed Computing. IEEE, 2012, pp164-171.

[8] Lu Q, Li S, Zhang W, et al. “A genetic algorithm-based job scheduling model for big data analytics”. Eurasip Journal on Wireless Communications & Networking, 2016, 2016(1), pp1-9.

[9] Kondekar R, Gupta A, Saluja G, et al. “A MapReduce based hybrid genetic algorithm using island approach for solving time dependent vehicle routing problem”, International Conference on Computer & Information Science. IEEE, 2012, pp263-269.

[10] Dean J, Ghemawat S, “MapReduce: simplified data processing on large clusters,” Communications of the ACM, Vol. 51, pp. 107-113, 2008.

[11] Sadasivam G S, Selvaraj D. “A novel parallel hybrid PSO-GA using MapReduce to schedule jobs in Hadoop data grids”, Nature and Biologically Inspired Computing. IEEE, 2010, pp377-382.

[12] Shvachko K, Kuang H, Radia S, et al. “The Hadoop Distributed File System”, IEEE, Symposium on MASS Storage Systems and Technologies. IEEE Computer Society, 2010, pp1-10.

[13] Chávez F, Fernández F, Benavides C, et al. “ECJ+HADOOP: An Easy Way to Deploy Massive Runs of Evolutionary Algorithms”, European Conference on the Applications of Evolutionary Computation. Springer International Publishing, 2016, pp91-106.

[14] Liao G, Datta K, Willke T L. “Gunther: search-based auto-tuning of mapreduce”, International Conference on Parallel Processing. 2013, pp406-419.

[15] Liu C, Zeng D, Yao H, et al. “MR-COF: A Genetic MapReduce Configuration Optimization Framework”, International Conference on Algorithms and Architectures for Parallel Processing. Springer International Publishing, 2015, pp344-357.

[16] Al-Madi N, Ludwig S A. “Scaling Genetic Programming for data classification using MapReduce methodology”, Nature and Biologically Inspired Computing. IEEE, 2013, pp132-139.