A Novel Parallel Approach of Cuckoo Search using MapReduce

Xingjian Xu[1*], Zhaohua Ji[1, 2, 3†], Fangfang Yuan[1], Xiaoqin Liu[4]

[1] College of Computer and Information Engineering, Inner Mongolia Normal University, Hohhot, China
[2] XingAn Vocational & Technial College, Wulanhaote, China
[3] DaMin seed industry co., LTD, Wulanhaote, China
[4] The Inner Mongolia Autonomous Region People's Hospital, Hohhot, China
dotswing@gmail.com, jzh978@163.com, 452983723@qq.com, xiaoqinliu6049@haodf.com

Abstract – Optimization is a very common problem both in theoretical and practical scenes. However, at the same time, it is also been classified as a NP-hard problem in most cases, complex and hardware resource consuming. In order to evaluate the optimal value in a limited time and with a limited computing resource, people often choose metaheuristic methods to search it. Cuckoo search is a newly proposed metaheuristic method, which is suggested very compromising in recent works. In this paper, we invent a novel approach to accommodate cuckoo search to the analysis of big data by using latest cloud computing technical, which is called MapReduce paradigm. During the implementation of our method, we use HADOOP platform as the backend MapReduce engine. At the last part, through a series of simulation experiments, we prove that our approach has a much better runtime performance when processing large dataset.

Index Terms – MapReduce, Cuckoo search, Metaheuristic, Big data, Cloud computing

1. Introduction

Many problems can be finally generalized as the problem of optimization: evaluating the optimal value in a restricted range, or find the best solution in all feasible solutions. There does indeed exist a theoretical method to get the optimal value or the best solution, in the worst case, we can check all the possible ones to select the best one, which is obliviously a time-consuming approach. This kind of method is called exact method, providing you the best solution, but impractical for its extremely low efficiency. Worse more, the optimization is a NP-hard problem and cannot be solved in a relatively short time[1], so finding the best solution with a satisfied efficiency is theoretical impossible. But in most scenes, people must get the best solution in a vey limited time; otherwise it will become meaningless through the evolving of the environment. So people have to make a concession by using the metaheuristic search, which will only promise you a suboptimal solution, but need much less running time and computing resource. Through they cannot ensure the solution is optimal, however, it is sufficient good (good enough) for the most practical problems, especially with imperfect priori information.

A lot of metaheuristic search algorithms have been proposed in past decades and make a big success, no matter in the field of theoretical mathematics or engineering applications, the most famous of which are genetic algorithm (GA)[2], simulated annealing (SA)[3], particle swarm optimization (PSO)[4], and so on. In order to make the search process convergence as fast and efficient as possible, the former researchers mainly denote their efforts in such two aspects: by proposing better algorithms and by parallelization of those algorithms[5].

A. Cuckoo search algorithm

Cuckoo search algorithm is a newly invented metaheuristic search method by Yang[6]. This method seems outperform other methods, giving a better quality solution in less running time, as reported in related review and comparison[7]. The original approach is a sequential algorithm, which is written in MATLAB code and hard to be re-implemented as a parallel one in traditional ways, since every generation (iteration) depends on the last iteration calculated result in the process of convergence to the optimal values.

Like most other metaheuristic search algorithms, cuckoo search is also inspired by the natural phenomenon, getting its original idea from some cuckoo species. It mimics the obligate brood parasitism[8] of cuckoo species: they lay their eggs in other species host birds' nests and deceive the host to brood for them. In addition, the solution is represented by egg in nest. When the cuckoo lays one egg, it means a new solution is generated. The ultimate goal is to replace the not-so-good solution with a newly generated, potentially better solution. The new solution is closely related with the former one, normally evolving by some kind of random walk. Cuckoo search rely on such below idealized rules:

1. A cuckoo only lays one egg in a randomly chosen nest each time;
2. The nests with most similar eggs will remained to the next generation;
3. The total number of all nests is not changed with the generation. The cuckoo egg is discovered by the host bird

[*] Xingjian Xu: Enrolled postgraduate student, studying at the College of Computer and Information Engineering, Inner Mongolia Normal University.
[†] Zhaohua Ji: Corresponding author, Master supervisor and Professor of College of Computer and Information Engineering, Inner Mongolia Normal University.

with a fixed probability $P_a \in (0,1)$ and will be dumped from the next generation.

Compared with other population or agents based search algorithm, cuckoo search has less parameter to assign. The specific parameters need to adjust will be described in the Method part below.

B. MapReduce paradigm

With the explosion growth of accessible data, the analyzed dataset is becoming bigger and bigger. The traditional parallel methods are not so smart when processing large or huge dataset, due to the fault task recovery, the Input/Output (IO) bottleneck, the network communication latency and many more unavoidable defects. So the concept of cloud computing[9] is widely adopted when analyzing large data. Cloud computing is essentially a particular form of distributed computing. Numerous and relative cheap commodity server nodes is its most remarkable characteristic.

MapReduce[10] is one of the most popular cloud computing technical, which is specially tailored for processing large dataset. The model is originally inspired by the map and reduce primitives occurring in some functional language, such as LISP[11] and HASKELL[12]. A MapReduce program has two main procedures: MAP and REDUCE. The raw data is firstly been transformed and filtered to a series of key-value pairs by the MAP function. The key-value pairs with the same key then will be collected together and feed to the same REDUCE function. Finally, the REDUCE function will output the analysis results. MAP functions and REDUCE functions will be fired concurrently without any synchronized global lock, which is one of the key points to make MapReduce a big success. The simplified MapReduce procedure is demonstrated as below.

$$(k_1, v_1) \xrightarrow{MAP} (k_2, v_2)$$
$$(k_2, list(v_2)) \xrightarrow{REDUCE} (v_3)$$

2. Methods

We use HADOOP[13] platform to carry out our method. HADOOP is a popular distribute computing framework, which implements MapReduce, and often referenced as the standard implementation of MapReduce by the open source community. It also provides necessary utilities to handle the headaches of job control, distributed file management (HDFS) and failure task recovery and task scheduling (YARN). All HADOOP business logic is build on the top of a fundamental assumption that hardware cluster nodes may fail and the platform must handle that. Its robustness and scalability accelerate the populating of HADOOP in some deep degree.

The principle of our tool combines the dived-and-conquer algorithm design technical and MapReduce programing paradigm together, applying them to the sequential cuckoo search method. The main workflow of our approach is described as below.

A. Divide the searching domain

The search domains may have different spatial shapes, but all of them can be generalized or simplified to be a 2-dimension rectangle shape, such as the searching domain of Rosenbrock function in Figure 1. In this procedure, the search domain will be divided into a series of cellular, without any gaps. The concrete divided method may have many isoforms[14]. In our implementation, we choose the simplest isoforms that directly divided the rectangle to n^2 sub-rectangle parts, uniformly both in width and length, and thus no extra overlapping. The idea of cellular dividing is originally from Tomassini's study[15] about complex system.

B. MAP procedure

The ordered id of each cellular is used as key; positions of the left upper corner will be used as the corresponding value. Every key-value pair will repeat emits for r times. Through this, in the next procedure, the same cellular will be searched for r times in order to avoid trapped in local best solution, losing the opportunity to convergence to global best solution. We found that r takes 4 enough in most cases, even if r takes larger values, some local best solutions are still be regarded as the global best ones. The cuckoo search itself it not 100 percent perfect, so it is impossible to overcome some inherent infects through the repeat search.

C. REDUCE procedure

As described above, in every REUCE procedure, the same cellular will be performed a standard cuckoo search independently for r times. The same REDUCE function process the same cellular ensures the underlying data locality, which will significantly decrease the network communication time. Data locality is the key aspect that keeps our method running faster than others[16]. Since the width and length of every cellular hold the same values, each time the REDUCE function only need to search a very small domain. The best result of them will be output.

The main parameters of cuckoo search are chosen as below:

1) *Nests count*: through the investigation of previous work[6], we set nests count to 20. More nests will not make our method to be more accuracy, but increasing its running time extremely;
2) *Max generation*: there are also no more accuracy benefits when the max generation set too high, since the search procedure has already converged before max generation has reached. Setting it to 2500 seems a good enough idea for most cases;
3) *Eggs dump probability* P_a: set it to 0.35. When it is set too high (>0.6), more flexibility occurs in the eggs quality in nests, and the whole process converges very slowly and unstable;
4) *Random walk*: The current position $x_{current}$ move to next possible position x_{next} guided by some kind of random walk P (Equation 1). The parameter a is the step size of this random walk, which we set it to 0.001 of the width

of the searching domain. Since we search a cellular for 4 times each, the two cuckoo search processes use Levy flight (Equation 2) and the others two search processes use normal distribution random walk. The mixture usage of Levy flight and normal distribution random walk make sure our method will have a combining advantage, ignoring the infects of the two random walks, respectively. We found that the combining of Levy flight and normal distribution random walk indeed improve the accuracy and its stabilization of final solutions.

$$x_{next} = x_{current} + a \oplus P \quad (1)$$

$$Levy \quad u = t^{-l}, (1 < l \leq 3) \quad (2)$$

D. Calculate the final result

Once we collected all the best results of every cellular, the final stage will be simply compare the best results among different cellular and output the minimum value of them as the global best solution. Compared with the original cuckoo search, our method explorer every part of search domain to ensure the better solution to be selected in the final stage.

3. Results and Discussion

Our experiments is performed on a cluster of two nodes, each of them has the same hardware configuration, comprised by 24 CPU cores (Intel Xeon E5-2630) and 44 gigabytes system memory. The operate system is both LINUX CENTOS version 6. In this paper, we check four popular metaheuristic search programs in our experiments: Cuckoo search based on Map Reduce (MR-CUCK), sequential Cuckoo search (CUCK) [6], parallel GA (p-GA)[14], and parallel PSO(p-PSO)[14]. MR-CUCK use 4 processes, 2 processes in each node. P-GA and p-PSO is not distributed parallel program, so we adjust them to use 4 processes in a single node, respectively. People often take some test functions to benchmark mateheuristic search program. Thus we choose the below three mainstream test functions:

1) Sphere function (Equation 3)

$$f(x) = \sum_{i=1}^{n} x_i^2 \quad (3)$$

2) Rosenbrock function (Equation 4, Figure 1)

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad (4)$$

3) Matyas function (Equation 5)

$$f(x,y) = 0.26(x^2 + y^2) - 0.48xy \quad (5)$$

Their corresponding theoretical optimal values (minimums) and search domains can be found in Fogel's handbook[17].

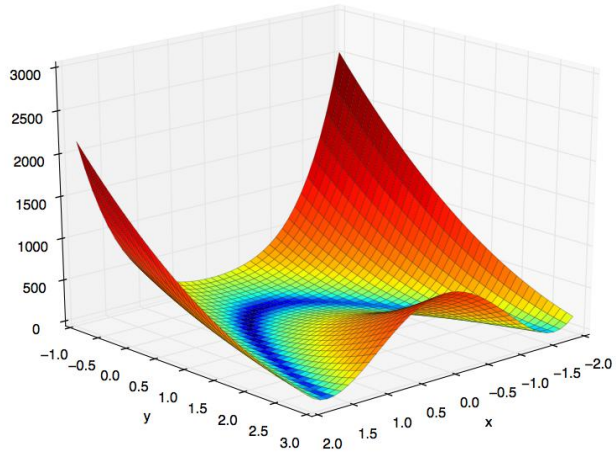

Figure 1 Surface of Rosenbrock function, getting its minimal value z = 0 at (x, y) = (1, 1)

A. Running time

The running time comparisons are depicted in Figure 2. We can see that our tool has a much less running time. That is mainly because HADOOP spreads all the computation tasks uniformly across CPU cores and rearrange them dynamically based on the real time workload information of every cluster server nodes. The iteration results are also past to next generation calculation asynchronously and efficiently, eliminating the IO communication headache at the best level. The minor square deviation of running time also suggested that our tool is less vulnerable to be trapped to the local optimal, since all the final results convergence to a single point.

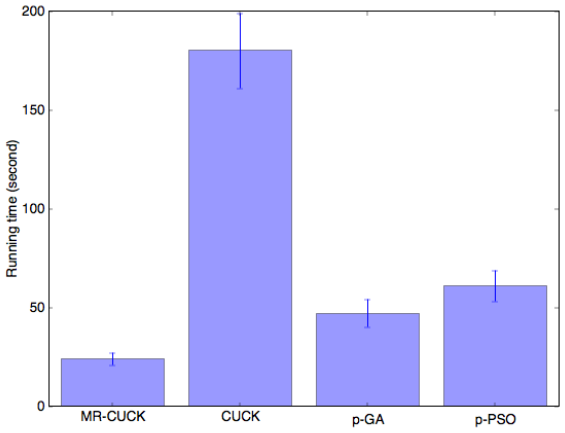

Figure 2 Running time of four methods (the sum of 100 repeat evaluation counts for a running time record, and 100 counted running time records for each method)

B. Accuracy

The accuracy comparisons are depicted in Figure 3. The accuracy of each method is evaluated by the summing the overall Euclidean distances between the result point location and the theoretical minimal point. The final sum is called 'mistake', which will reflect the results accuracy quality of

different methods, and the method that has a less mistake value is more accuracy. Our tool gives a more stabilized (the standard error is relative small) suggested optimal points, which means that our tool is often more reliable than the remained three methods, and you cannot trust a method that give you a reliable best solution at 50% chance. Since our algorithm explores a more spacious searching domain, and the final result is a summary of all divided area, it is not hard to understand the better stabilization of result solution.

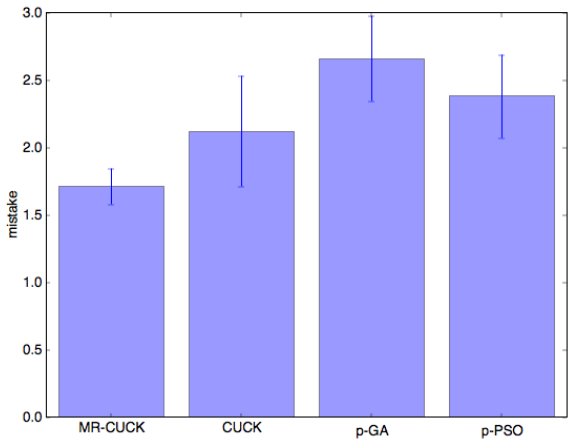

Figure 3 Results accuracy of four methods (100 repeats for each method)

4. Conclusion

Through the discussion of results above, MapReduce is proved to be a valuable sharp knife to break out the optimization problems. With a finicky implementation, algorithm can easily acquires a significant speed up. But for those scenes that only need very little computing resource, perhaps MapReduce is not the best suitable programing model, since the startup and ending cleanup operations also require some time.

By combining the MapReduce, many sequential or traditional parallel algorithms could be quickly refactoring as a modern cloud computing program, with the ability to face the challenges brought by large dataset analyzing. What is better still, some HADOOP computing clusters are now accessible as a public service with an affordable price, such as Amazon EMR (http://aws.amazon.com/elasticmapreduce/). The building and maintains of HADOOP cluster is not a easy job, so the public HADOOP cluster service will definitely drive more people to enjoy the benefits of MapReduce and HADOOP.

Acknowledgment

This work is fully supported by MOE (Ministry of Education in China) Project of Humanities and Social Sciences (ProjectNo.11YJA910003).

References

[1] S. P. Boyd and L. Vandenberghe, Convex Optimization. 2004.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, Apr. 2002.

[3] H. Szu and R. Hartley, "Fast simulated annealing," Phys. Lett. A, vol. 122, no. 3, pp. 157–162, 1987.

[4] C. Sammut and G. I. Webb, Eds., Encyclopedia of Machine Learning. Boston, MA: Springer US, 2010.

[5] E. Alba, Parallel Metaheuristics: A New Class of Algorithms. Wiley-Interscience, 2005, p. 576.

[6] X.-S. Yang and S. Deb, "Cuckoo Search via Lvy flights," in Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, 2009, pp. 210–214.

[7] P. Civicioglu and E. Besdok, "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," Artif. Intell. Rev., vol. 39, no. 4, pp. 315–346, Jul. 2011.

[8] B. G. Stokke, A. Moksnes, and E. Røskaft, "OBLIGATE BROOD PARASITES AS SELECTIVE AGENTS FOR EVOLUTION OF EGG APPEARANCE IN PASSERINE BIRDS," Evolution (N. Y)., vol. 56, no. 1, pp. 199–205, Jan. 2002.

[9] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, p. 50, Apr. 2010.

[10] J. Dean and S. Ghemawat, "MapReduce," Commun. ACM, vol. 51, no. 1, p. 107, Jan. 2008.

[11] P. H. Winston and B. K. Horn, "LISP. Second edition," Jan. 1986.

[12] P. Hudak, T. Johnsson, D. Kieburtz, R. Nikhil, W. Partain, J. Peterson, S. Peyton Jones, P. Wadler, B. Boutel, J. Fairbairn, J. Fasel, M. M. Guzmán, K. Hammond, and J. Hughes, "Report on the programming language Haskell," ACM SIGPLAN Not., vol. 27, no. 5, pp. 1–164, May 1992.

[13] T. White, Hadoop: The Definitive Guide. 2012.

[14] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: Recent advances and new trends," Int. Trans. …, vol. 20, no. 1, pp. 1–48, Jan. 2013.

[15] M. Tomassini, "Simulating Complex Systems by Cellular Automata," 2010.

[16] C. Zhou, "Fast parallelization of differential evolution algorithm using MapReduce," in Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10, 2010, p. 1113.

[17] D. B. Fogel, "Evolutionary algorithms in theory and practice," Complexity, vol. 2, no. 4, pp. 26–27, Mar. 1997.