Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 19(3): 155-160, Sep. 2021 Regular paper

Cooperative Coevolution Differential Evolution Based on Spark for Large-Scale Optimization Problems

Xujie Tan[1], **Hyun-Ae Lee**[2], **and Seong-Yoon Shin**[2*], ***Member, KIICE***

[1]School of Computer and Big Data Science, Jiujiang University, Jiujiang 332005, China
[2]School of Computer Information & Communication Engineering, Kunsan National University, Kunsan 54150, South Korea

Abstract

Differential evolution is an efficient algorithm for solving continuous optimization problems. However, its performance deteriorates rapidly, and the runtime increases exponentially when differential evolution is applied for solving large-scale optimization problems. Hence, a novel cooperative coevolution differential evolution based on Spark (known as SparkDECC) is proposed. The divide-and-conquer strategy is used in SparkDECC. First, the large-scale problem is decomposed into several low-dimensional subproblems using the random grouping strategy. Subsequently, each subproblem can be addressed in a parallel manner by exploiting the parallel computation capability of the resilient distributed datasets model in Spark. Finally, the optimal solution of the entire problem is obtained using the cooperation mechanism. The experimental results on 13 high-benchmark functions show that the new algorithm performs well in terms of speedup and scalability. The effectiveness and applicability of the proposed algorithm are verified.

Index Terms: Cooperative coevolution, Differential evolution, Large-scale optimization, Resilient distributed datasets

I. INTRODUCTION

The differential evolution(DE) algorithm is a global optimization algorithm based on real coding [1]. Owing to its simplicity, efficiency, and global parallelism, the DE algorithm has been successfully applied in industrial design and engineering optimization fields in recent years; additionally, researchers have improved and innovated the DE algorithm and achieved some achievements. For example, Brest et al. [2] constructed an adaptive method for the control parameters and proposed an adaptive DE algorithm (jDE). Wang et al. proposed a compound DE algorithm (CoDE) [3], which randomly combines three selected mutation strategies and three groups of control parameters. These studies are primarily focused on low-dimensional problems (30 dimensions). However, when addressing high-dimensional problems (1000 dimensions), the performances of these DE algorithms will deteriorate significantly, and the search time increases exponentially with the dimension; as such, the problems are extremely difficult to solve and dimensionality issues persist [4].

To solve the high-dimensional optimization problem effectively, scholars have proposed different strategies, among which the representative one is cooperative co-evolution (CC) [5]; CC adopts the divide-and-conquer concept. First, the high-dimensional complex problem is decomposed into simple low-dimensional subproblems. Second, each subproblem is solved separately. Finally, using all child synergy mechanisms, the solution for the entire issue is obtained. The random grouping strategy proposed by Yang et al. [6] allows two related variables to be categorized under the same group with a high probability of obtaining more accurate solutions

Received 22 June 2021, **Revised** 08 Separtment 2021, **Accepted** 14 September 2021
***Corresponding Author** Seong-Yoon Shin (E-mail: s3397220@kunsan.ac.kr, Tel: +82-63-469-4860)
School of Computer Information & Communication Engineering, Kunsan National University, Kunsan, 54150 Korea.

Open Access **https://doi.org/10.6109/jicce.2021.19.3.155** print ISSN: 2234-8255 online ISSN: 2234-8883

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Copyright ⓒ The Korea Institute of Information and Communication Engineering

in large-scale optimization problems. Researchers have applied CC to many fields, such as large-scale black-box optimization problems [7], SCA(strategic conflict avoidance) [8], FII (fast interdependency identification) [9], CCPSO (cooperative coevolving particle swarm optimization) [10], and DG2 (differential grouping 2) [11]. However, to solve high-dimensional optimization problems, they adopted a serial method, which requires a long computation time and does not readily provide satisfactory solutions within a reasonable time frame. In recent years, cloud computing has been successfully applied to the field of large-scale information processing, such as machine learning [12], ant colony algorithm [13], CRFS (Conditional Random Fields) [14], differential evolution algorithm [15, 16], graph data analysis [17], and classification algorithms [18].

Researchers have proposed distributed differential evolution algorithms based on the MapReduce model of Google's open-source platform, Hadoop [19, 20]. It was discovered that the MapReduce model is a general batch computing model that lacks an effective mechanism for parallel computing data sharing and cannot provide effective support for iterative computing. Therefore, the differential evolution algorithm based on the MapReduce model necessitates data exchange through frequent file reading and writing, which reduces its efficiency [21].

The spark cloud platform is a distributed data processing framework proposed by researchers at Berkeley University [22] and has been successfully applied in many fields. A new data abstraction model based on Spark, i.e., the resilient distributed dataset (RDD), was developed, and it effectively supported iterative computation relational queries. Because the RDD model is based on in-memory computing, it avoids the disadvantages of the MapReduce model, which is less efficient as it frequently reads and writes disk data.

Based on the Spark cloud platform, a cooperative differential evolution algorithm, the SparkDECC (Spark cloud platform based cooperative differential evolution) algorithm, is proposed herein. It adopts the divide-and-conquer strategy, decomposes a high-dimensional optimization problem into low-dimensional subproblems using the random grouping strategy, and encapsulates the problems into an RDD. In RDD, each subproblem evolves independently and in parallel for several generations. All subproblems are combined into complete problems using the synergistic mechanism and the optimal individual. In this study, the SparkDECC algorithm was evaluated using 13 standard functions. The experimental results show that the proposed algorithm is effective and feasible.

The remainder of this paper is organized as follows: Section II introduces the DE. Our proposed DE algorithm, namely SparkDECC, is presented in detail in Section III. Section IV provides the experimental results. Section V presents the conclusions and future work.

II. DE

DE is used to solve the global optimization problem based on real coding. The objective function is expressed as follows [23]:

Minimize $f(x)$
s.t. $x = (x_1, x_2, \ldots, x_D), \quad x_i \in [\min, \max]$

D indicates the dimension; *min* and *max* are the ranges of the solution space.

The population x is propagated randomly as follows:

$$x_{i,j} = \min + rnd\,(\max - \min), \tag{1}$$

where $rnd \in [0, 1]$ is a random number.

After the population is initialized by formula (1), the population promotes generations via the following three procedures.

A. Mutation

At G generations, the mutant vector v_i can be produced as follows:

$$v_1 = x_{r_1} + F_i(x_{r_2}\, x_{r_3}), \tag{2}$$

where $i = 1, 2, \ldots, NP$; r_1, r_2, and $r_3 \in [1, NP]$ are random different integers. The control parameter $F_i \in [0, 1]$.

B. Crossover

After mutation, a crossover is performed on x_i and v_i to generate u_i. The binomial crossover is defined as follows:

$$u_{i,j} = \begin{cases} v_{i,j},\ if\,(rnd \le CR_i\, or\ j = j_{rnd}) \\ x_{i,j}, otherwise \end{cases}, \tag{3}$$

where $j = 1, 2, \cdots, D$, $rnd \in [0, 1]$, $j_{rnd} \in [1, D]$, the condition $j = j_{rnd}$ ensures that the vector u receives at least one variable from v, and $CR \in [0, 1]$.

C. Selection

Finally, the best vector survives in the next generation by comparing the value of the function. The greedy selection scheme is described as follows:

$$x_i = \begin{cases} u_i, if\,(f(u_i) \le f(x_i)) \\ x_i, otherwise \end{cases}, \tag{4}$$

where $f(.)$ is a function.

The DE algorithm repeatedly performs the three-step generation until a termination criterion is satisfied.

III. SparkDECC

A. Spark

To support iterative computing more efficiently, the Spark platform extends the MapReduce cloud model [24]. The Spark platform provides two important abstractions: RDD and accumulators. RDD is a fault-tolerant parallel data structure that provides a read-only partitioned set of records that coexist in memory. Broadcast is a shared variable that caches data to each node, eliminating the necessity for data transfer, thereby reducing communication overhead and improving communication performance. The Spark API provides two types of operations for the RDD, namely transformations and actions. In transformations, the same activity is performed for each data partition, and a new RDD is returned. The action operator triggers the operation on the RDD and returns the value to the master control node. The internal implementation mechanism of the RDD is based on the iterator, which renders the data access more efficient, avoids the memory consumption of the intermediate results, and results in a more efficient and fast iterative calculation.

DE is a population-based evolutionary algorithm with inherent parallelism. Therefore, DE can fully integrate the parallelism of Spark. Spark initializes the population parallelized in the master control node via parallelization and stores it in memory via the key-value method, i.e.,

$$[key_i, value_i], \quad i = 1, 2, \cdots, m,$$

where m is the number of subpopulations, key_i is an integer that represents the number of the ith subpopulation, and $value_i$ is the value of the specific implementation of the ith subpopulation of DE on Spark, as shown in Fig. 1.

Spark abstracts the data in memory into the RDD using key–value pairs, preserves subpopulations in different nodes based on the value of key_i, uses the parallel operator of the RDD to evolve several generations of each subpopulation in parallel, and then generates a new population by merging with the operator's collection. At the end of the cycle, the optimal value of the entire population is obtained through action operator reduction.

B. SparkDECC

CC frameworks can effectively address large-scale optimization problems. However, as the population size increases, the time required by the CC framework increases rapidly. To improve the convergence speed of the CC framework, the advantages of cloud computing are combined with the CC framework, and a CC algorithm based on SparkDECC is proposed.

The SparkDECC algorithm first decomposes a high-dimensional problem into several low-dimensional subproblems via the random grouping method, where one subproblem corresponds to a subpopulation, and the position information of each subproblem in the entire problem is preserved. Based on the key_i value, the low-dimensional subpopulations are distributed to the corresponding partitions in the RDD, and the subpopulations in each partition execute the mutation and cross selection of the DE algorithm in parallel. When calculating the individual fitness value of the subpopulations, the optimal individuals of the final round are selected to form a complete population and perform a local optimization. After several generations of evolution in the corresponding partition, the low-dimensional subpopulation is merged into a new complete population based on its location information; the flow chart of the SparkDECC algorithm for returning the optimal individual through a global search is shown in Fig. 2.

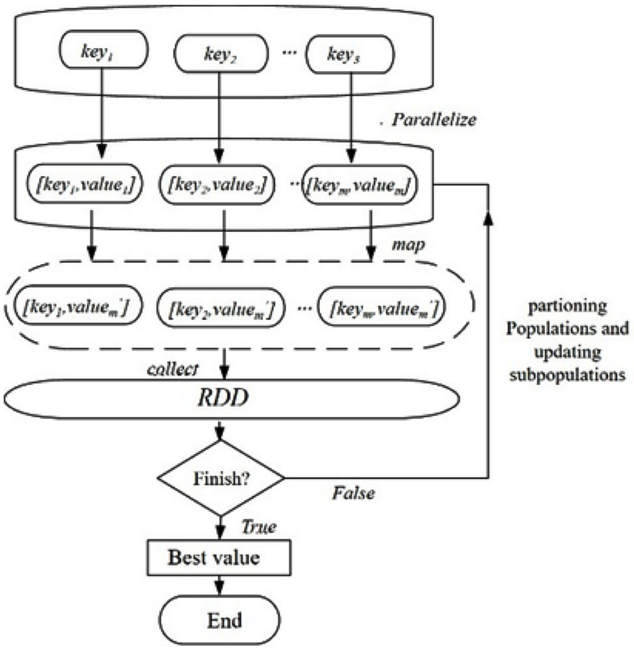

Fig. 1. DE Based On Spark.

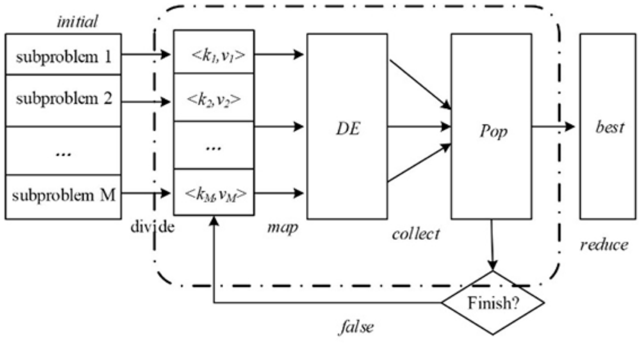

Fig. 2. Flowchart of SparkDECC algorithm

IV. EXPERIMENTAL STUDY

A. Benchmark Functions and Experimental Setting

To test the performance of the SparkDECC algorithm for solving large-scale optimization problems, we performed experiments using 13 test functions selected from the literature [25]. Among them, f_1~f_8 are unimodal functions, f_9~f_{13} are multimodal functions, f_4 and f_5 are non-decomposable functions, and all other functions are decomposable.

The spark cloud model was used in this experiment. The configuration of each node was as follows: 64-bit Core-i7 CPU, 3.4 Hz main frequency, 8 G memory, Ubuntu 13.10 operating system, Hadoop 2.2.0, Spark 1.2.0, IntelliJIdea 14.1.2 programming environment, and languages Scala and Java.

To verify the performance of SparkDECC and the factors that affect its performance, the dimension of the problem in SparkDECC was set to 1000, the dimension of the subproblem was set to 100, the problem size NP = 100, F = 0.5, Cr = 0.9, and the algebra of independent operations of each subpopulation was 100. The scalability of the SparkDECC algorithm was verified while the other parameters remained unchanged.

B. Experimental results and comparisons with other DE variants

Table 1 shows a comparison between SparkDECC and OXDE [26], CoDE, jDE, and PSO [27] in terms of the average optimal value and standard variance. The parameter settings of the five algorithms were consistent, and each algorithm was performed 25 times independently. The Wilcoxon rank-sum test was used to analyze the experimental results of the four algorithms. The significance level was 0.05, where –, +, and ≈ represent inferior, superior, and equal, respectively.

Table 1 shows a comparison between SparkDECC and each of three other DE algorithms. The results show that SparkDECC can converge rapidly to the optimal result for seven functions, namely f_1, f_5, f_6, f_{10}, f_{11}, f_{12}, and f_{13}, and the experimental data are superior to those of the other three algorithms. The convergence of the SparkDECC algorithm in f_3, f_8, and f_9 is stagnant, and the experimental results are inferior to those of the other algorithms. The performance of each algorithm is similar for the non-decomposable function f_4. The results of f_2 for jDE is inferior and superior to those of OXDE and CODE, respectively. The experimental results of f_7 with a noise function were inferior to those of CoDE, superior to jDE, and comparable to OXDE.

The acceleration ratio [20] is an effective index for measuring the parallelism of the algorithm; it is expressed as shown in Eq. (5).

$$S_k(M_n) = \frac{T_k(1)}{T_k(M_n)}, \tag{5}$$

where $T_k(1)$ represents the average time of k independent runs on a partition, and $T_k(M_n)$ represents the average time of k independent runs for n partitions. In the experiment, f_1 and f_3 were selected for the unimodal functions, and f_9 and f_{11} were selected for the multimodal functions. According to the four high-dimensional optimization functions, three different evaluation times, namely 5E6, 2.5E6, and 5E5, were set, which were independently executed 10 times. The acceleration ratio in Fig. 3 shows that when testing the high-dimensional optimization function of the SparkDECC algorithm, as the number of partitions increased, the execution

Table 1. Comparison of SparkDE, OXDE, CoDE, jDE, and PSO algorithms for solving results

F	OXDE		CoDE		jDE		PSO		SparkDECC	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	4.76E+02	1.67E+02	1.50-05	1.74E-05	2.46E-06	1.23E-05	2.30E+06	2.79E+04	**5.85E-13**	**1.62E-13**
f_2	5.27E+01	6.89E+00	8.89E-01	9.74E-01	**1.74E-10**	**8.62E-10**	7.16E+04	3.30E+04	6.60E-07	1.00E-07
f_3	1.54E+06	2.33E+05	**4.50E+04**	**6.12E+03**	7.54E+04	1.48E+04	8.68E+08	6.51E+07	5.31E+07	7.20E+06
f_4	2.59E+01	1.84E+00	**2.67E+01**	**1.95E+00**	5.04E+01	4.58E+00	4.01E+02	7.98E+00	9.76E+01	2.22E-01
f_5	2.28E+04	7.08E+03	2.39E+03	2.12E+02	2.18E+03	2.21E+02	9.23E+12	1.29E+12	**1.62E+03**	**1.62E+02**
f_6	7.86E+03	8.86E+02	5.26E+01	6.26E+01	1.06E+04	2.98E+03	2.30E+06	1.29E+05	**1.60E-01**	**4.73E-01**
f_7	5.68E+00	7.25E-01	**9.06E-01**	**1.03E-01**	2.91E+01	1.60E+01	3.48E+13	3.87E+12	3.62E+00	1.67E-01
f_8	-4.17E+05	7.27E+02	-1.89E+05	5.14E+03	**-4.19E+05**	**3.18E-01**	-1.34E+05	4.51E+03	-6.11E+04	1.18E+03
f_9	4.94E+02	5.32E+01	4.59E+03	2.10E+02	**2.98E+00**	**4.03E+00**	2.33E+06	1.35E+05	1.10E+04	3.93E+01
f_{10}	6.49E+00	2.79E-01	2.25E+00	1.82E-01	5.12E+00	7.06E-01	2.15E+01	3.77E-02	**4.55E-08**	**8.16E-09**
F_{11}	5.02E+00	1.19E+00	7.70E-03	2.29E-02	8.91E-01	7.39E-001	5.75E+02	4.15E+01	**3.54E-14**	**1.03E-14**
f_{12}	3.01E+00	7.11E-11	5.75E-02	4.85E-02	1.32E+06	2.20E+06	6.99E+12	7.75E+11	**7.46E-04**	**2.58E-03**
f_{13}	1.94E+03	3.40E+02	1.15E+02	7.53E+01	1.14E+07	8.17E+06	7.87E+12	8.78E+11	**8.79E-04**	**3.04E-03**
−/+/ ≈	8/3/2		7/4/2		7/4/2		12/1/0			

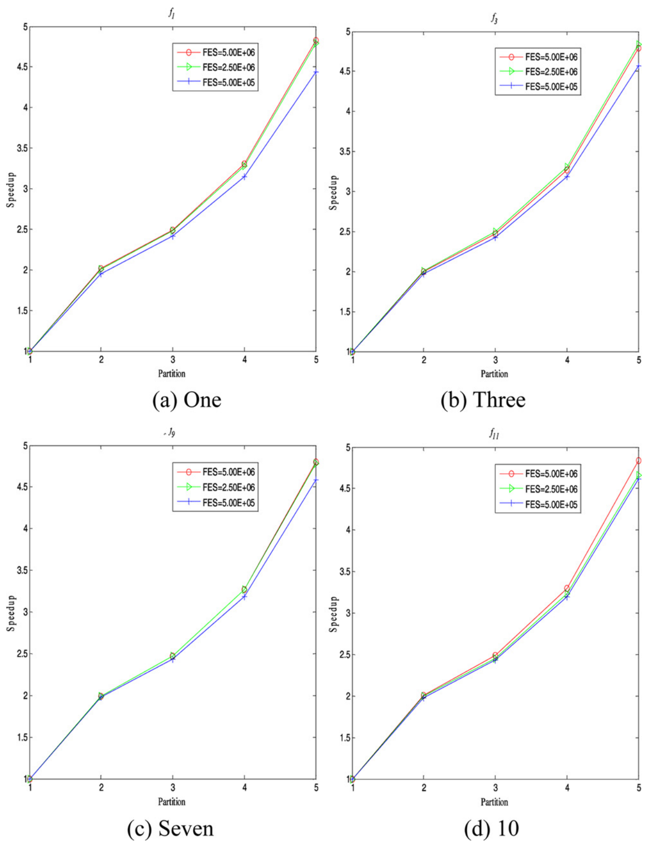

Fig. 3. Acceleration ratio.

time of the algorithm gradually decreased, and the acceleration effect improved. When the number of partitions was increased to five, the acceleration ratio was approximately five, which is consistent with the analysis of time complexity provided in Section 3.2. In addition, the relationship between the acceleration curve of the function and the evaluation times of the function was not clear, which indicates that the acceleration performance was stable.

V. CONCLUSIONS

In this study, a new collaborative cloud differential evolution algorithm (SparkDECC) based on Spark was developed using a new iterative cloud computing model. The SparkDECC algorithm decomposes a high-dimensional problem into multiple low-dimensional subproblems of the same dimension via the random grouping strategy, and each subproblem corresponds one-to-one with the partitions in the RDD model. Each subproblem executes the DE algorithm in parallel. After the subproblems have evolved independently for several generations, the optimal individuals are updated to improve the diversity of the population. SparkDECC was implemented on the Spark cloud model using Scala language. Based on a comparison involving 13 standard test functions, the results showed that SparkDECC exhibited high accuracy, high speed, and good scalability. The experimental results showed that the acceleration ratio exhibited an almost linear relationship with the number of partitions, indicating a favorable acceleration effect.

In future studies, new grouping strategies based on SparkDECC will be investigated, and synergistic mechanism of stragegies will be improved continuously to improve the convergence efficiency and solution precision of the algorithm.

REFERENCES

[1] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341-359, 1997. DOI: 10.1023/A:1008202821328.

[2] J. Brest, S. Greiner, B. Boskovic B, M. Mernik, and V. Zumer, “Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646-657, 2006. DOI: 10.1109/TEVC.2006.872133.

[3] Y. Wang, Z. Cai, and Q. Zhang, “Differential evolution with composite trial vector generation strategies and control parameters,” *IEEE Transactions on Evolutionary Computation*, vol 15, no. 1, pp. 55-66, 2011. DOI: 10.1109/TEVC.2010.2087271.

[4] F. V. d. Bergh and A. P. Engelbrecht, “A cooperative approach to particle swarm optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, 2004. DOI: 10.1109/TEVC.2004.826069.

[5] M. A. Potter and K. A. De Jong, “A cooperative coevolutionary approach to function optimization,” in *Proceedings of International Conference on Parallel Problem Solving from Nature*, vol. 866, pp. 249-257, 1994. DOI: 10.1007/3-540-58484-6_269.

[6] Z. Yang, K. Tang, and X. Yao, “Large scale evolutionary optimization using cooperative coevolution,” *Information Sciences*, vol. 178, no. 15, pp. 2985-2999, 2008. DOI: 10.1016/j.ins.2008.02.017.

[7] Y. Mei, M. N. Omidvar, X. Li, and X. X. Yao, “A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization,” *ACM Transactions on Mathematical Software*, vol. 42, no. 2, pp. 1-42, 2016. DOI: 10.1145/2791291.

[8] X. Guan, X. Zhang, J. Wei, I. Hwang, Y. Zhu, and K. Cai, “A strategic conflict avoidance approach based on cooperative coevolutionary with the dynamic grouping strategy,” *International Journal of Systems Science*, vol. 47, no. 9, pp. 1995-2008, 2016. DOI: 10.1080/00207721.2014.966282.

[9] X. M. Hu, F. L. He, W. E. Chen, and J. Zhong, “Cooperation coevolution with fast interdependency identification for large scale optimization,” *Information Sciences*, vol. 381, pp. 142-160, 2017. DOI: 10.1016/j.ins.2016.11.013.

[10] X. Li and X. Yao, “Cooperatively coevolving particle swarms for large scale optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210-224, 2012. DOI: 10.1109/TEVC.2011.2112662.

[11] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, “DG2: a faster and more accurate differential grouping for large-scale black-box optimization,” *IEEE Transactions on Evolutionary Computation*,

vol. 21, no. 6, pp. 929-942, 2017. DOI: 10.1109/TEVC.2017.2694221.

[12] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, DB Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "Mllib: machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp 1235-1241, 2016. DOI:10.5555/2946645. 2946679.

[13] L. Wan, G. Zhang, H. Li, and C. Li, "A novel bearing fault diagnosis method using Spark-based parallel ACO-K-means clustering algorithm," *IEEE Access*, vol. 9, pp. 28753-28768, 2021. DOI:10.1109 /ACCESS.2021.3059221.

[14] Z. Jizhao, J. Yantao, X. Jun, Q. Jianzhong, W. Yuanzhuo,C. Xueqi, "SparkCRF: a parallel implementation of CRFs algorithm with spar," *Journal of Computer Research and Development*, vol. 53, no. 8. pp. 1819-1828, 2016. DOI: 10.7544/issn1000-1239.2016.20160197.

[15] C. Deng, X. Tan, X. Dong, and Y. Tan, "A parallel version of differential evolution based on resilient distributed datasets model," in *Proceedings of Bio-Inspired Computing- Theories and Applications*, Springer, Berlin, Heidelberg. vol. 562, pp 84-93, 2015. DOI: 10. 1007/978-3-662-49014-3_8.

[16] D. Teijeiro, X. C. Pardo, P. González, J. R. Banga, and R. Doallo, "Implementing parallel differential evolution on Spark," in *Proceedings of European Conference on the Applications of Evolutionary Computation, Lecture Notes in Computer Science*, vol. 9598, pp. 75-90, 2016. DOI:10.1007/978-3-319-31153-1_6.

[17] B. Liu, S. He, D. He, Y. Zhang, and M. Guizani, "A Spark-based parallel fuzzy c-means segmentation algorithm for agricultural image big data," in *IEEE Access*, vol. 7, pp. 42169-42180, 2019. DOI: 10. 1109/ACCESS.2019.2907573.

[18] R. A. Hasan, R. A. I. Alhayali, N. D. Zaki, and A. H. Ali, "An adaptive clustering and classification algorithm for Twitter data streaming in Apache Spark," *Telkomnika*, 2019, vol. 17, no. 6, pp. 3086-3099, 2019. DOI:10.12928/TELKOMNIKA.v17i6.11711.

[19] C. Zhou, "Fast parallelization of differential evolution algorithm using MapReduce," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 1113-1114, 2010. DOI: 10.1145/1830483.1830689

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

[21] H. Peng, X. Tan, C. Deng and S. Peng, "SparkCUDE: a spark-based differential evolution for large-scale global optimization," *International Journal of High Performance Systems Architecture*, vol. 7, no. 4, pp. 211-222, 2017. DOI: 10.1504/IJHPSA.2017.092390.

[22] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 15-28, 2012. DOI:10.5555/2228 298.2228301.

[23] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: A practical approach to global optimization*, Springer, 2005.

[24] X. W. Wang, Q. Y. Dai, W. C. Jiang, and J. Z. Cao, "Retrieval of design patent images based on MapReduce model," *Journal of Chinese Computer Systems*, vol. 33, no. 3, pp. 626-632, 2012.

[25] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, 1999. DOI:10.1109/4235.771163.

[26] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Information Sciences*, vol. 185, no. 1, pp. 153-177, 2012. DOI: 10.1016/j.ins.2011. 09.001.

[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 International Conference on Neural Networks*, pp. 1942-1948, 1995. DOI: 10.1109/ICNN.1995.488968.

Xujie Tan
received his Ph.D. degree from the School of Computer Information and Communication Engineering of Kunsan National University, Korea and is now a professor in the School of Computer and Big Data Science at Jiujiang University, Jiujiang, China. His research interests include nature-inspired computation and application, pervasive Computing, big data, and cloud computing. He can be contacted at tanxj@kunsan.ac.kr.

Hyun-Ae Lee
received her M. S. degree in transportation planning from Hanyang University and is now a Ph.D. student in the Department of Computer Information and Communication Engineering, Kunsan National University. She is in charge of the Hyundai Autoever Convergence Mobility Business at Hyundai Motor Group. Her research interests include autonomous cooperative driving, advanced traffic information system, smart city, and i-MOD.

Seong-Yoon Shin
received his M.S. and Ph.D. degrees from the Department of Computer Information Engineering of Kunsan National University, Gun- san, Korea, in 1997 and 2003, respectively. From 2006 to the present, he has been a professor in the abovementioned department. His research interests include image processing, computer vision, and virtual reality.