International Journal of Innovation Engineering and Science Research
(ISSN: 2581-4591)

www.ijiesr.com

Parallel Feature Selection approach for Big Data Processing using Coarse-grained Genetic algorithm

Waad Bouaguel
College of Business, University of Jeddah

ABSTRACT

The Genetic Algorithm (GA) is one of the most popular meta-heuristic algorithms that has been found to be effective for ac- curate feature selection tasks. In recent decades, the number of instances and features hasgrown in size, making most current feature selection algorithms and even GAs, unscalable. To improve the scalability of big data feature selection algorithms and exceptionally GA, we adopt distributed computing strategies such as the Map-Reduce model and Hadoop system. In this paper, we first present recent work dealing with the use of parallel genetic algorithms in large datasets. Then we will propose a new parallel genetic algorithm based on the island model. The performance of the proposed method was theoretically and empirically compared to existing feature selection methods when handling large-scale datasets and the results confirm the effectiveness of our pro- posed method.

Keywords—*Genetic algorithm; featureselection; island model; parallel processing.*

I. INTRODUCTION

Feature selection is a category of dimensionality reduction. In this category, a subset of existing features is selected without a transformation [1]. The idea behind feature selection is to reduce the influence of tricky features in a dataset, where tricky and unnecessary features include irrelevant and redundant features. Removing these features reduces the dimensionality of the search space and speeds up the learning algorithm[2], [3].

The selection of the optimal feature subset is an optimization problem that proved to be NP-hard. Two major approaches are traditionally used to deal with NP-hard problems, exact methods, and meta-heuristics [4]. Exact methods allow the exact solution to be found, but this approach is impractical since it is extremely time-consuming in real-world problems. On the other hand, meta-heuristics are used for solving complex and real-world problems, because they provide sub-optimal solutions in a reasonable time.

GA is an adaptive heuristic search algorithm inspired by the process of natural evolution and one of the most well-known Population-based meta-heuristics. As reported by [5], GA was faster in finding near-optimal features from large datasets compared to other algorithms. Despite the efficiency of GA in feature selection, as most of the existing feature selection algorithms, GAs is also proposed to deal with small data sets under several gigabytes and become impractical in Big data cases [6], [7]. Hence, to overcome this kind of issue feature selection is performed in a distributed manner. This is a simple task for GAs since GA is naturally parallelizable since it searches simultaneously from multiple points, not from a single point. Three levels of parallelization can be exploited: Global parallelization model, Fine-grained parallelization model, and Coarse-grained parallelization or island model [8]–[10]

The global parallelization model uses one master node to apply genetic and selection operators and distributes individuals among the remaining slaves, where the fitness values of the individuals are computed [11]. This model provides lower run time only for very slow objective functions which is a drawback according to [12] and its search mechanism uses a single population. The fine-grain parallel GA uses an overlapped neighborhood system to provide a smooth diffusion of the individuals [13], [14], while the coarse-grained model also known as the island model distributes the population among the processors. Eachnode called an island is given a sub-population to process [13]. The genetic Algorithm is then applied toeach sub-population. The individuals are then migrated to other nodes according to some given criteria in order to synchronize the solution set. This model has two main advantages:

(1) sub-populations could explore various portions of the search space and (2) there is less communication between nodes [15].

In this paper, we present a new Parallel GA based on the island model for feature selection. The performance of the proposed method will be theoretically and empirically compared to existing feature selection methods. This paper is organized as follows: in section 2, we discuss recent works on Parallel GA for large datasets and feature selection. Section 3 describes the proposed method. Section 4 is dedicated to presenting the experimental study and Section 5 gives the results and discussion.

II. RECENT WORKS ON PARALLEL GA FOR LARGE DATASETS AND FEATURE SELECTION.

In recent years, evolutionary algorithms have gained popularity in solving complex problems. Several research works have been done in this context, including a study by [16] that used a weighted Gene Genetic Algorithm for big data feature selection. The proposed method stores a weight for each gene on the population solutions. Then, the stored weights are decreased or increased during the evolution of the genetic algorithm based on the appearance of the tested set of features on the best solutions. The proposed algorithms were applied to five datasets and the results show that the proposed algorithms can effectively enhance the classifying performance against the other feature selection algorithms. In another work, [17] performed feature selection through the implementation of a selection process by a genetic algorithm, using different classification techniques, which allows the generation of a classification model of a set of children's activities based on the environmental sound.

[8] defines a framework for implementing parallel Genetic Algorithms (GAs) on the Hadoop platform by applying the MapReduce model.[18] proposed a parallel GA to derive bi-clusters from the microarray matrix, and [19] proposed a wrapper technique using KNN classifier for supervised feature selection to yield good classification accuracy with minimal features. To improve the performance, they used a master-slave Parallel Genetic algorithm using Hadoop MapReduce and they suggested a wrapper technique using K-NN classifier for supervised feature selection to yield good classification accuracy with minimal features. [20] proposed a MapReduce implementation of a parallel GA to find the minimal rough set redact. The model targeted many components such as constructing a distinction table, GA population evaluation, and GA operations. Several mappers have been used to construct the distinction table and the reduce phase is charged with the genetic operations. The driver is charged with collecting the results to select the best individual. This approach has reduced the execution time without demeaning the solution quality in terms of the redact size. Using parallel also programming, a Fuzzy pre-processing technique was used to reduce the input data for a Multi-objective GA. A Parallel SVM Classifier was used to assess the best-identified gene subsets [21].

III. PROPOSED METHOD

Step I: In order to perform GA, an initial population of features is created by randomly selecting a set of ordered rank lists of features. From empirical studies, over a wide range of problems, a population size between 30 and 100 is usually recommended [22].

Step II: Once the initial population is fixed, we need to select new members for the next generation. To do this, we can use the MapReduce model, which divides the whole population into several separate sub-populations. the whole population is divided into smaller blocks, and then the HDFS files are used to create new populations [23].

Step III: This step is called the map phase. The mapper function processes the input data and creates several small chunks of data.

GA is applied to each data chunk and the resulting map output for each dataset of m features is m pair of key-value < k; v > that refers to the < features; values of features >. In this phase, we apply the genetic operations selection, crossover, and mutation, to each sub-population, which means different sub-populations will take different crossover rates and mutation rates. After every 100 generations, the best individual in each sub-population will be sent to other sub-populations and replace the worst individual. The selected members are then crossed over with the cross-over probability. Therefore, crossover combines the features of two lists to create two similar ranked lists. The mutation operator is used, it acts as a population perturbation operator. Mutation operates by randomly changing one or more elements of any list. Typically, the mutation does not occur frequently, so the mutation is of the order of 0.001 [22].

Step IV: The map phase output contains the features selected by the GA, and the reduce phase aggregates all the relevant features in one file.

IV. EXPERIMENTAL STUDY

A. Dataset description and experimental environment

We used four datasets from the UCI Machine Learning Repository to assess the performance and effectiveness of our proposed method. These datasets were the Madelon dataset, the Semeion Handwritten Digit dataset, the Hill-Valley dataset, and the Breast Cancer dataset.Our approach was implemented using the Hadoop MapReduce implementation with R language on Amazon Web Services Cloud, and the RHadoop project that brings several R packages to work with Hadoop interactively.In this work, we deal with three kinds of implementation: a traditional GA with no parallel processing, a Parallel GA on Hadoop platform, and a fully distributed version on the Amazon cluster.

TABLE 1 Clearly illustrates the experimental environment for each implementation.

- Nonparallel GA: GA was used for the feature selection task with no parallel processing on a single machine with 6 GB of RAM.
- Parallel GA on Hadoop platform: the solutions are represented in the form of a population shared globally and managed by a master processor. The other slave processors take the charge of evaluating the fitness of the global population. the proposed approach was implemented over a Hadoop platform on a single machine with 6 GB of RAM.
- Parallel GA with an Amazon cluster: required a full Hadoop cluster composed of 1 master and 3 slaves, each machine has 2 CPUs and 8 GB of RAM.

TABLE I. EXPERIMENTALENVIRONMENT.

	Non-Parallel GA	***Parallel GA* Hadoop platform**	***Parallel GA* *Amazon cluster***
Number of machines	One machine	One machine	One master and three slaves
RAM	6 GB	6 GB	8 GB
Framework		Hadoop 2.7.3	Hadoop 2.7.3
Language	R (3.3.1)	R (3.3.1)	R (3.3.1)

B. Parameters setting

In this section, we aim to understand the impact of the GA's parameters on the learning results, and how they affect the trade-off between fast convergence and exploratory power.

1) Setting for Crossover operator and rate

In GAs, the crossover operation is used to create new individuals from the old ones, thus making them more likely to be better than the parents. Some well-known crossover operators are reflected in [24] and summarized in TABLE 2

TABLE II. SUMMARY OF CROSSOVER OPERATORS

Singlepoint; n-point; Discrete; Arithmetical; Simulatedbinary; Unimodalnormaldistribution; Parentcentric; Wright'sheuristic; Dynamicfuzzy; connectivebased simplex; Averagebound; Geometrical; Uniform

We propose to use different crossover operators in each generation of the GA to provide diversity in the population and improve the overall performance of the GA. Hence in each generation, a different crossover operator is chosen from the set of operators presented in TABLE 2

2) Setting of the Mutation operator

In this part we try to apply multi-mutation operators to a GA, so that in each generation a different mutation operator is chosen, thereby increasing diversification and enhancing the GA's performance. Several mutation operators were discussed in [24] and presented in TABLE 3.

3) setting of Crossover and mutation probability rate

Crossover occurs with a probability pc and mutation occurs with a probability pm. The crossover rate and mutation rate control the behavior and performance of GAs.

The higher the crossover rate, the quicker exploitation proceeds. A pc that is too large would disrupt individuals faster than they could be exploited. Small pm values are commonly adopted in GAs. Typicalvalues of pc are in the range 0.5 -1.0, while typical values of pm are in the range 0.001-0.05.

TABLE III. SUMMARY OF MUTATION OPERATORS

Single point; n-point; Discrete; Arithmetical; Simulated binary; Unimodal normal distribution; Parent centric; Wright's heuristic; Dynamic fuzzy; connective based simplex; Average bound; Geometrical; Uniform

To identify suitable values of GA parameters, the performance of the GA will be evaluated for several different sets of these parameters, see Table 4.

TABLE IV. GA CONTROL PARAMETER SETS. CONTROL

Parameter	***Values***
Number of generations	[50;100]
Population size	[10;20;30]

Parameter	***Values***
Probability of Crossover	[0.8;1.0]
Probability of Mutation	[0.01;0.02]

To identify suitable values of GA parameters, 24 different sets of GA control parameters will be tested. The accuracy of the classification results will be improved by using severalgenerations of 100, a population size of 20, a crossover rate of 0.8, and a mutation rate of 0.1, these values will be used in the rest of the experiments. see Table 5 for more details.Each parameter set shown in Table 5 was run 10 times, and each time

with different initial starting search points to eliminate the effect of initial randomness in the selected solutions.

TABLE V. TABLE5.ACCURACYSUMMERFOREACHSEATOFPARAMETERS

(NUMBEROFGENERATION,POPULATIONSIZE,CROSSOVERRATE,MUTATIONRATE)

Index	Parameter set	Accuracy	Index	Parameter set	Accuracy	Index	Parameter set	Accuracy
1	[50;10;0.8;0.01]	0.571	2	[50;10;0.8;0.02]	0.578	3	[50;10;1;0.01]	0.582
4	[50;10;1;0.02]	0.567	5	[50;20;0.8;0.01]	0.560	6	[50;20;0.8;0.02]	0.601
7	[50;20;1;0.01]	0.600	8	[50;20;1;0.02]	0.570	9	[50;30;0.8;0.01]	0.459
10	[50;30;0.8;0.02]	0.462	11	[50;30;1;0.01]	0.502	12	[50;30;1;0.02]	0.494
13	[100;10;0.8;0.01]	0.490	14	[100;10;0.8;0.02]	0.632	15	[100;10;1;0.01]	0.590
16	[100;10;1;0.02]	0.487	**17**	**[100;20;0.8;0.01]**	**0.689**	18	[100;20;0.8;0.02]	0.588
19	[100;20;1;0.01]	0.561	20	[100;20;1;0.02]	0.572	21	[100;30;0.8;0.01]	0.602
22	[100;30;0.8;0.02]	0.569	23	[100;30;1;0.01]	0.604	24	[100;30;1;0.02]	0.490

4) Fitness function

The fitness function is one of the most important parameters of GA because it evaluates how good a solution is. In our case, the fitness function was chosen to be RandomForest (RF) [25], and the accuracy of RF helped to decide if a candidate feature subset fit or not. Hence, many sets of randomly selected features are generated from each file. Where, each one of them is represented by an F-bit string in which the value '1' or '0' of any bit means the presence or absence of the corresponding feature, respectively. For each generation, individuals are used to producing a forest of decision trees. Then a fitness score is assigned to each individual based on how well the corresponding tree classifier classified the test dataset using the root-mean-square.

V. RESULTS AND DISCUSSION

A. Evaluation of running time

The given results in Table (6) show that the proposed approach takes less time to choose the most relevant features compared to the non-parallel version and the parallel Hadoop implementation on a single machine.

TABLE VI. RUNNINGTIMERESULTS.

	Runningtime(s)	Selectedfeatures
	Madelon data set	
NonParallel GA	207945.992	261
ParallelGAonHadoopplatform	204530.196	238
ParallelGAwithanAmazoncluster	151949.403	246
	Semeion Handwritten Digit data set	
NonParallel GA	28490.67	81
ParallelGAonHadoopplatform	26954.098	110
ParallelGA with an Amazone cluster	16428.100	114
	Hill-Valley data se	
NonParallel GA	4839.744	37
ParallelGAonHadoopplatform	4092.932	40
ParallelGA with an Amazone cluster	2277.31	44

Table 6 show that for the Madelon dataset our approach success to select features in 151949.403 seconds compared to 207945.992 and 204530.196 second given respectively by the nonparallel and the single machine implementation. We also notice from Table 6 that the new approach with the Amazon cluster obtained also the smallest running time for the Semeion Handwritten Digit dataset. The same results were confirmed with the Hill-Valley dataset where the best set of features was selected in half the time used by the non-parallel GA and the single-machine implementation.

This improvement in running time for both parallel implementations confirm the importance of introducing parallel computing to GA paradigm. The proposed approach for feature selection in GA takes less time compared to the non-parallel version and the parallel Hadoop implementation on a single machine. The best running time was obtained with the Amazon cluster.

B. Evaluation of learning performance

In this part, we evaluate the proposed approach according to the learn- ing results. Accuracy and F-measure are used as classification measures.

From Table (7) and Table (8) we evaluate the effectiveness of our proposed feature selection method using SVM and RF classifiers. The obtained results show the effectiveness of selecting features using the island and the MapReduce models in improving the classification results.

Let's start with the results given by the SVM classifier for the Madelon dataset as shown in Table 7. The accuracy and F-measure of the SVM classifier for the Madelon dataset were 0.4855 and 0.58, respectively, for the non-parallel implement of GA and the parallel implementation on the Haddop platform. While they were about 0.533 and 0.59 for our proposed method for 246 selected features. The obtained results using our approach are higher than those provided by the non-parallel implement of GA and the parallel implementation on Hadoop platform.

From Table 7 we notice that the proposed approach achieves the best rates of classification for the Semeion Handwritten Digit dataset when compared to the baseline model. As shown in Table 7 we achieve the best rates of accuracy (.i.e. 0.992) and F-measure (.i.e. 0.990) and the smallest time for model building. For the Hill-Valley dataset, feature selection improved the classification performance

(accuracy and F-measure) and reduced the model-building time. This finding comes just confirms the importance of feature selection and its effect on improving classification performance.

TABLE VII. CLASSIFICATION RESULTS FOR USING SVM.

	Time	Accuracy	F-measure
		Madelon data set	
Allfeatures	2.440	0.485	0.58
Non-parallel GA(261 features)	0.408	0.488	0.57
ParallelGAonHadoopplatform (238features)	0.372	0.520	0.581
ParallelGAwithAmazoncluster (246features)	0.361	0.533	0.59
	Semeion Handwritten Digit data set		
Allfeatures	0.592	0.979	0.98
Non-ParallelGA(81features)	0.608	0.979	0.98
ParallelGAonHadoopplatform (110features)	0.348	0.991	0.99
Parallel GA withan Amazon cluster(114features)	0.398	0.992	0.99
	Hill-Valleydataset		
Allfeatures	0.340	0.502	0.260
Non-ParallelGA(37features)	0.048	0.508	0.262
ParallelGAonHadoopplatform (40features)	0.072	0.530	0.27
Parallel GA withan Amazon clus-ter(44features)	0.068	0.532	0.27

Time: Time for building the model in seconds

Let's move to the results given by the RF classifier and summarized in Table 8. The obtained results confirm our assumptions about the importance of introducing parallel computing in feature selection. From the obtained results we conclude on the effectiveness of the proposed approach that obtained not only the best learning performance but alsothe best reduced time for building the model.

TABLE VIII. CLASSIFICATION RESULTS USING RF.

	Time	Accuracy	F-measure
	Madelondataset		
Allfeatures	6.608	0.511	0.56
NonParallelGA(261features)	3.386	0.483	0.59
ParallelGAonHadoopplatform (238features)	3.088	0.515	0.61
Parallel GA withan Amazon cluster(246features)	3.262	0.526	0.61
	Semeion HandwrittenDigitdataset		
Allfeatures	716.608	0.712	0.88
NonparallelGA(81features)	226.004	0.895	0.874
ParallelGAonHadoopplatform (110features)	3.630	0.902	0.98
Parallel GA withan Amazon cluster(114features)	3.570	0.974	0.98
	Hill-Valleydataset		
Allfeatures	183.132	0.5138	0.48
NonParallelGA(37features)	55.64	0.5635	0.58
ParallelGAonHadoopplatform (40features)	0.860	1	1
Parallel GA withan Amazon cluster(44features)	0.872	1	1

Time: Time for building the model in seconds

VI. CONCLUSION

This paper investigates a new feature selection approach based on Genetic Algorithm using parallel processing. It evaluates the efficiency ofthe proposed method on real data sets from the UCI Machine Learning Repository. The obtained results confirm our assumptions about the importance of introducing parallel computing in feature selection. In the future, we intend to study in deep other parallel implementations of a large range of evolutionary feature selection algorithms.

REFERENCES

[1] F. Sahin, “A survey on feature selection methods,” Computers and Electrical Engineering, vol. 40, pp. 16–28, 01 2014.

[2] N. AlNuaimi, M. M. Masud, M. A. Serhani, and N. Zaki, “Streaming feature selection algorithms for big data: A survey,” Applied Computing and Informatics, 2019.

[3] I. Rodriguez, R. Huerta, C. Elkan, and C. S. Cruz, “Quadratic programming feature selection,” Journal of Machine Learning Research. 11, no. 4, pp. 1491–1516, 2010.

[4] W. Bouaguel, G. Mufti, and M. Limam, A New Feature Selection Technique Applied to Credit Scoring Data Using a Rank Aggregation Approach Based on: Optimization, Genetic Algorithm and Similarity, 03 2015, pp. 337–334.

[5] A. Alzubaidi, G. Cosma, D. Brown, and G. Pockley, “Breast cancer diagnosis using a hybrid genetic algorithm for feature selection based on mutual information,” 10 2016.

[6] Y. Li, T. Li, and H. Liu, “Recent advances in feature selection and its applications,” Knowledge and Information Systems, vol. 53, pp. 551–577, 2017.

[7] M. Rong, D. Gong, and X. Gao, “Feature selection and its use in big data: Challenges, methods, and trends,” IEEE Access, vol. PP, pp. 1–1, 01 2019.

[8] F. Ferrucci, P. Salza, T. Kechadi, and F. Sarro, “A parallel genetic algorithms framework based on hadoopmapreduce,” 04 2015, pp. 1664–1667.

[9] A. Natarajan and B. Ramasamy, “A fuzzy parallel island model multi objective genetic algorithm gene feature selection for mi- croarray classification,” vol. 11, pp. 2761–2770, 03 2016.

[10] Shyh-Chang Lin, W. F. Punch, and E. D. Goodman, “Coarse-grain parallel genetic algorithms: categorization and new approach,” in Proceedings of 1994 6th IEEE Symposium on Parallel and Distributed Processing, 1994, pp. 28–37.

[11] F. De Toro, J. Ortega, J. Fern´andez, and A. D´ıaz, “Psfga: a parallel genetic algorithm for multiobjective optimization,” in Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on. IEEE, 2002, pp. 384–391.

[12] E. Alba and J. Troya, “Troya, j.m.: Improving flexibility and effi- ciancy by adding parallelism to genetic algorithms. statistics and computing 12(2), 91-114,” Statistics and Computing, vol. 12, pp. 91–114, 04 2002.

[13] N. H., S.Mahajan, and S. Omkar., “Big data clustering using ge- netic algorithm on hadoop mapreduce,” International Journal Of Scientific and Technology Research, vol. 4, April 2015.

[14] L. Di Geronimo, F. Ferrucci, A. Murolo, and F. Sarro, “A parallel genetic algorithm based on hadoop mapreduce for the automatic generation of junit test suites,” in Software Testing, Verification

[15] and Validation (ICST), 2012 IEEE Fifth International Conference on. IEEE, 2012, pp. 785–793.

[16] J. Yazidi, W. Bouaguel, and N. Essoussi, “A parallel imple- mentation of relief algorithm using mapreduce paradigm,” in In- ternational Conference on Computational Collective Intelligence. Springer, 2016, pp. 418–425.

[17] T. Abed Mohammed, O. Bayat, O. Ucan, and S. Alhyali, “Hybrid efficient genetic algorithm for big data feature selection problems,” Foundations of Science, 03 2019.

[18] A. Garcia-Dominguez, C. E. Galvan-Tejada, L. A. Zanella- Calzada, H. Gamboa-Rosales, J. I. Galvan-Tejada, and J. M. Celaya, “Feature selection using genetic algorithms for the gen- eration of a recognition and classification of children activities model using environmental sound,” Mobile information systems, vol. 2020, 2 2020.

[19] A. Laishram and S. Vipsita, “Bi-clustering of gene expression mi- croarray using coarse grained parallel genetic algorithm (cgpga) with migration,” in India Conference (INDICON), 2015 Annual IEEE. IEEE, 2015, pp. 1–6.

[20] G. T. Hilda and R. Rajalaxmi, “Effective feature selection for supervised learning using genetic algorithm,” in Electronics and Communication Systems (ICECS), 2015 2nd International Con- ference on. IEEE, 2015, pp. 909–914.

[21] M. A. Alshammari and E.-S. M. El-Alfy, “Mapreduce implemen- tation for minimum reduct using parallel genetic algorithm,” in Information and Communication Systems (ICICS), 2015 6th In- ternational Conference on. IEEE, 2015, pp. 13–18.

[22] A. Natarajan and R. Balasubramanian, “A fuzzy parallel island model multi objective genetic algorithm gene feature selection for microarray classification.” International Journal of Applied Engineering Research, vol. 11, no. 4, pp. 2761–2770, 2016.

[23] V. Pihur, S. Datta, and S. Datta, “RankAggreg, an R package for weighted rank aggregation,” BMC Bioinformatics, vol. 10, no. 1, pp. 62–72, 2009.

[24] Q. He, X. Cheng, F. Zhuang, and Z. Shi, “Parallel feature selection using positive approximation based on mapreduce,” in Fuzzy Sys- tems and Knowledge Discovery (FSKD), 2014 11th International Conference on. IEEE, 2014, pp. 397–402.

[25] S. L. Siew Mooi, A. B. Md Sultan, M. Sulaiman, A. Mustapha, and K. Y. Leong, “Crossover and mutation operators of genetic algorithms,” International Journal of Machine Learning and Com- puting, vol. 7, pp. 9–12, 02 2017.

[26] G. Biau, F. C´erou, and A. Guyader, “On the rate of convergence of the bagged nearest neighbor estimate,” Journal of Machine Learn- ing Research, vol. 11, no. Feb, pp. 687–712, 2010.

[27] .