Int. J. Nonlinear Anal. Appl.
Volume 12, Special Issue, Winter and Spring 2021, 1561-1572
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2021.5812

DNA barcoding using particle swarm optimization on apache spark SQL case study: DNA of covid-19

Lala Septem Riza[a], Muhammad Ilham Nurfathiya[a], Jajang Kusnendar[a], Khyrina Airin Fariza Abu Samah[b]

[a]*Department of Computer Science Education, Universitas Pendidikan Indonesia, Indonesia*
[b]*Faculty of Computer and Mathematical Sciences, University Teknologi MARA Cawangan Melaka Kampus Jasin, Melaka, Malaysia*

(Communicated by Madjid Eshaghi Gordji)

Abstract

The objective of this research is to design and implement a computational model to determine DNA barcodes by utilizing the Particle Swarm Optimization (PSO) algorithms implemented on Big Data Platforms, namely Apache Hadoop and Apache Spark. The steps are as follows: (i) inputting DNA sequences to Hadoop Distributed File System (HDFS) in Apache Hadoop, (ii) pre-processing data, (iii) implementing PSO by utilizing the User Defined Function (UDF) in Apache Spark, (iv) collecting results and saving to HDFS. After obtaining the computational model, two following simulations have been done: the first scenario is using 4 cores and several worker nodes, meanwhile, the second one consists of a cluster with 2 worker nodes and several cores. In terms of computational time, the results show a significant acceleration between standalone and big data platforms with both experimental scenarios. This study proves that the computational model built on the big data platform shows the development of features and acceleration of previous research.

Keywords: Big data, Algorithm, Particle swarm optimization, Similarity check, Motif discovery, DNA barcoding

1. Introduction

Reading on DNA barcoding [2] is one of the problems of concern in molecular information on the taxonomic aspects of biology. DNA barcoding intends to provide an efficient approach to identifying individuals at the species level, and consequently, greatly contribute to taxonomic investigations [8].

Email addresses: lala.s.riza@upi.edu (Lala Septem Riza), ilhamn33@gmail.com (Muhammad Ilham Nurfathiya), jkusnendar@upi.edu (Jajang Kusnendar), khyrina783@uitm.edu.my (Khyrina Airin Fariza Abu Samah)

Received: August 2021 *Accepted:* November 2021

This DNA barcoding method is easy to popularize and standardize by building a unified database and identification platform [15]. The identification efficiency of the DNA barcoding method will not be affected by experience or environmental factors [14]. Diagnosis on a molecular scale through DNA barcoding provides an alternative for plant identification that is fast, accurate, and unambiguous in the identification process. DNA barcoding can be used for two purposes, which are as a new tool to help taxonomists who are accustomed to working hard on hard-to-identify specimens and as an innovative tool for non-taxonomists and to quickly identify plants [6].

The amount of biological data of living things that exist, both animals and plants that we can collect today is very limitless. This huge amount of data is a challenge for the world of bioinformatics, which is the application of computational system techniques to analyze and manage natural data, namely biological information [3]. There are two challenges faced by the DNA motif detection approach, which are DNA transcription factors and weak DNA motifs. DNA transcription factors where proteins bind to DNA sequences and structural motifs which are usually located upstream of the target gene, usually have a short structure. Weak DNA motifs to conserve due to evolution and mutation are also challenges of this DNA motif detection approach. Therefore, using either a simple string comparison method or an exhaustive search of all combinations of several methods cannot effectively provide accurate identification of transcription factors [26].

The Particle Swarm Optimization Algorithm [12, 25, 11, 7] has undergone many changes since its introduction in 1995. As researchers have learned about this technique, they have created new versions, developed new applications, and published theoretical studies on the effects of various parameters and aspects of the algorithm [16]. Research from [9] on DNA motifs discovery using Particle Swarm Optimization (PSO) and using Expectation Maximization (EM) can be concluded that PSO/EM tends to have a consistent offset from the actual known motif locations. Only in this study, it is stated that even though using environmental topology, particle swarm optimization/Expectation-Maximization stops at a local optimum.

Apache Spark performs memory computations using a Resilient Distributed Dataset (RDD). Apache Spark could let the scientists to process large amounts of data at once in a short amount of time up to 100 times faster than Apache Hadoop in memory and 10 times faster than Apache Hadoop on disk. Apache spark also allows user programs to load data into cluster memory and makes it possible to perform iterative queries, which is great for machine-learning algorithms [5].

Research on DNA barcoding motif discovery has been done a lot. Following are some studies on DNA barcodes such as in [19] using a spiral dynamic optimization algorithm, also in [27] which used ant colony optimization and expectation-maximization, and [23] which used bacterial foraging optimization. The use of the particle swarm optimization algorithm has also been used in [9] which uses particle swarm optimization and expectation maximization.

This research is aimed on using the PSO algorithm which will then add the concept of big data analysis to the computation which is expected to speed up the time required for computing with large data. Two big data platforms are used in this research as follows: Apache Hadoop and Apache Spark. In the other words, in this case we implement PSO into Apache Spark by utilizing the UDF function, so that it can run parallel along available nodes. Moreover, the DNA sequence datasets need to be saved in Hadoop Distributed File System (HDFS) that is available in Apache Hadoop.

2. Research methodology

The computational model in this study can be seen in Figure 1. The first step is to enter the input data in the form of DNA sequence data which is uploaded first to Google Cloud Storage. Google Cloud Storage itself provides a connector that connects Google Cloud Storage with HDFS

which makes Google Cloud Storage directly accessible without having to use HDFS first. After the file was uploaded, the next step was to import it into Apache Spark with the data type used in the form of a data frame containing RDD which has been partitioned into several parts in each block file in Google Cloud Storage.

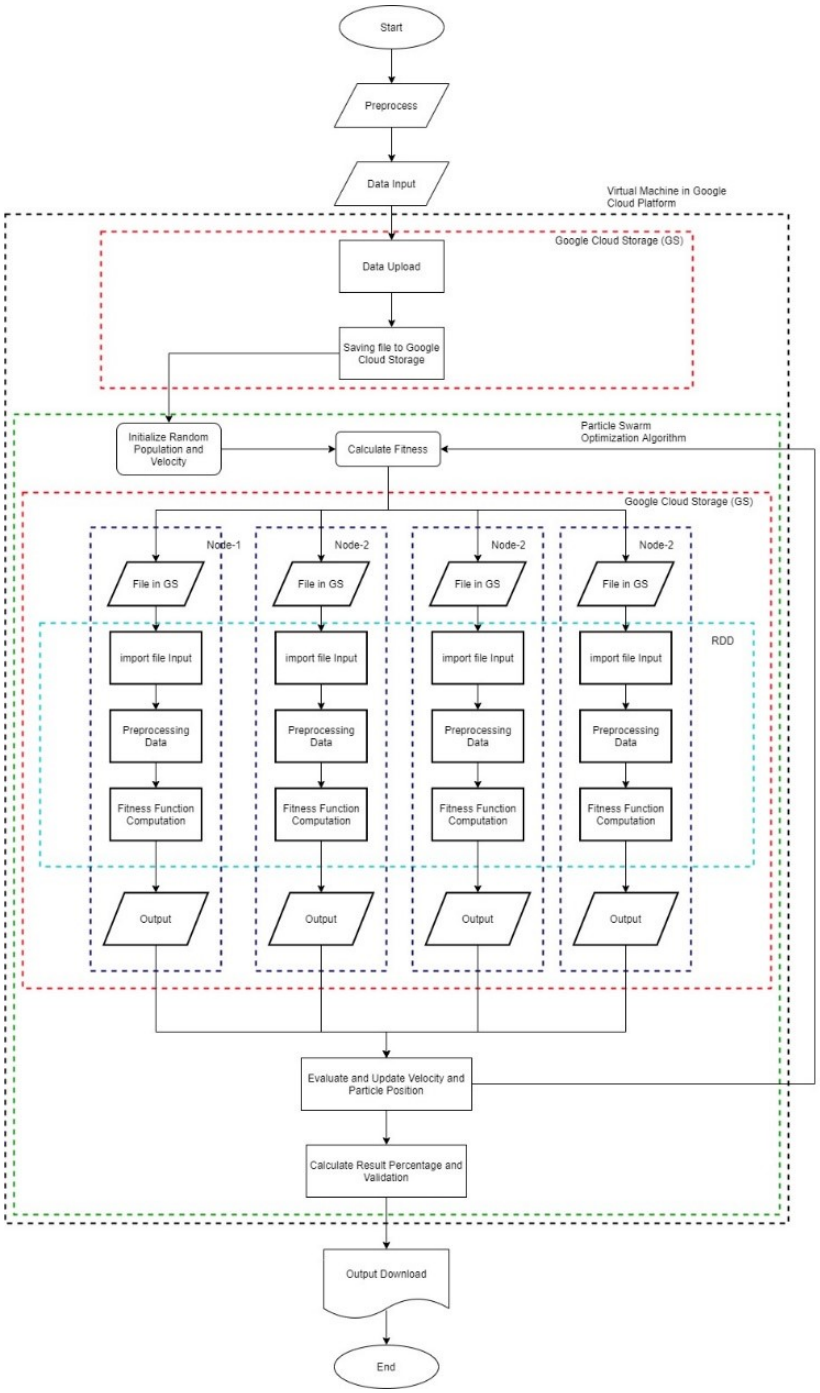

Figure 1: Research Computational Model

The next stage was data preprocessing. This stage was the first stage carried out in the Apache Spark environment. The process carried out at this stage was to separate the sample name and DNA sequence from the sample and also perform sequence trimming on the sequences obtained. After the clean data was obtained, we start to implement the PSO algorithm in Apache Spark as illustrated

in Figure 2.

Input:

- Objective function F(x), with $x = x_1, x_2, x_3, \ldots, x_{dim}$ where dim is numbers of variables/dimensions
- Numbers of particles in population (numPopulation)
- Maximum Numbers of iterations (maxIter)
- Learning Coefficients c_1 and c_2
- Weight inersia w
- Maximum speed Vmax

Output: Best particle Gbest
Process:

1. Generate population $\overrightarrow{X}_i$ ($i = 1, 2, 3, \ldots$,numPopulation)
2. Find best local solution $\overrightarrow{L}_{best}$
3. Update global best $\overrightarrow{G}_{best}$ if $\overrightarrow{L}_{best}$ is better
4. Generate initial speed on each particle V_0
5. while t <maxIter do

for $i = 1$: numPopulation do
Pick random numbers $\overrightarrow{r}_1$ dan $\overrightarrow{r}_2$
Update particles' speed with

$$\overrightarrow{V}_i(t+1) = w * \overrightarrow{V}_i(t) + c_1 * \overrightarrow{r}_1(\overrightarrow{L}_{best} - \overrightarrow{X}_i(t)) + c_g * \overrightarrow{r}_2(\overrightarrow{G}_{best} - \overrightarrow{X}_i(t))$$

Update particles' position with

$$\overrightarrow{X}_i(t+1) = \overrightarrow{X}_i(t) + \overrightarrow{V}_i(t)$$

Check and update particles' position according to
if $F(\overrightarrow{X}_i) < F(\overrightarrow{L}_{best})$ then
Update local best: $\overrightarrow{L}_{best} \leftarrow \overrightarrow{X}_i$
if $F(\overrightarrow{L}_{best}) < F(\overrightarrow{G}_{best})$ then
Update global best: $\overrightarrow{G}_{best} \leftarrow \overrightarrow{L}_{best}$
end if
end if
end for
end while
6. return Gbest

First, we generate a random population index as the initial index of sequence checking. Before generating population, we define an index sequence in numeric as in Figure 2. It can be seen in the DNA sequence DNA-A, we have index 1, 2, and 3 for "G", "A", "C", etc. The same way is done on the DNA sequence to compare: DNA-B, DNA-C. Based on the numerical index sequence, we randomly generate a population as a collection of initial indexes. For example, in Figure 3, on the second row/individual, we have two initial indexes: 2 and 6. It means if we define the length of pattern is 3 then there are two short DNA sequences: "A C A" and "G A C".

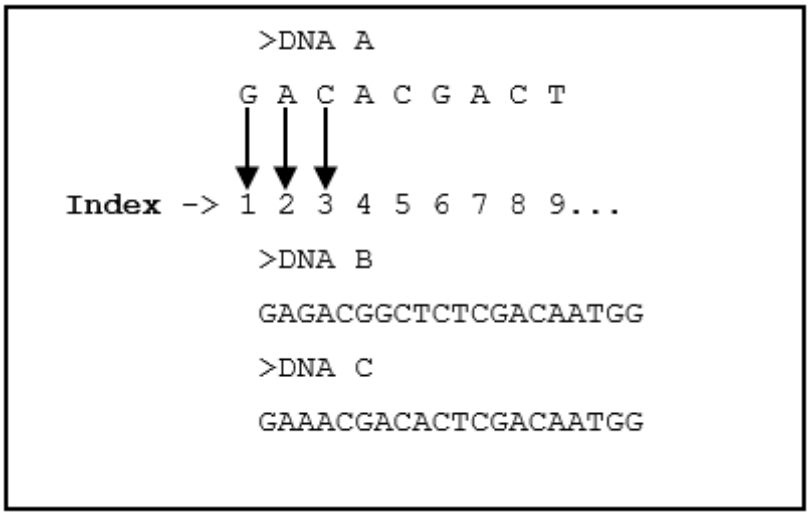

Figure 2: Numerical index Representation on DNA Sequence

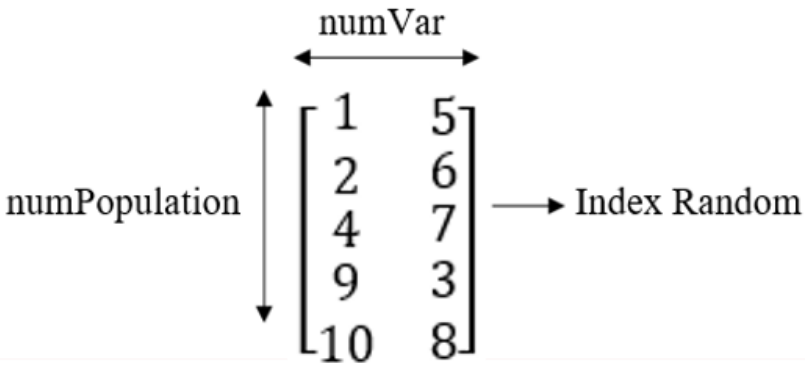

Figure 3: An example of a population generated randomly on the sequence DNA-A

After obtaining a population, we calculate fitness function, which is by using Hamming Distance. Hamming distance is done by taking the first sequence and then comparing it with the following sequences. Then the flow that will be carried out to make comparisons is to compare DNA-A with DNA-B then DNA-A with DNA-C as illustrated in Figure 4. After the value of the fitness function was obtained, the next step was to update the particle position and update the velocity by evaluating the new particle position and velocity which is the learning stage of the particle swarm optimization algorithm as can be seen in Figure 2.

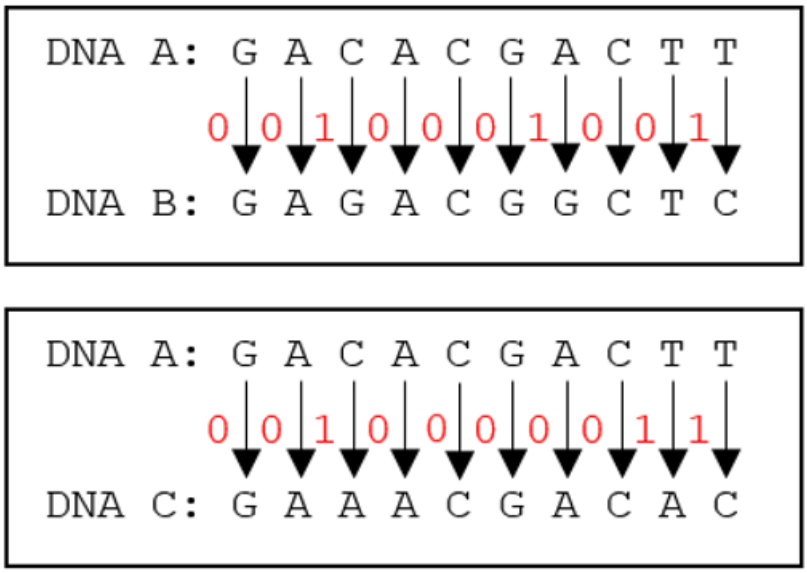

Figure 4: An example of calculation of Hamming Distance on DNA-A, DNA-B, and DNA-C

Since we are working on Big Data Platforms, basically all computational steps explained previously are done in Apache Spark by utilizing the UDF function. In running this program, it is necessary to initiate SparkSession first which is a library to run applications on Apache Spark. SparkSession will use the master set up by the dataproc on the Google Cloud Platform. Some other required spark libraries include types, udf, lit, and length. Meanwhile, the libraries for running the particle swarm optimization algorithm are pandas and numpy, as illustrated in Figure 5.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf, lit, length
from pyspark.sql.types import *

import numpy as np
import pandas as pd
import time

spark = SparkSession \
 .builder \
 .config("yarn") \
 .appName("Python Spark SQL DNA Barcoding") \
 .getOrCreate()
```

Figure 5: Initialization and import library in SparkSession

After importing the software library and running SparkSession, the next step is to generate code for user data input. In addition, we also include information about the cluster such as the number

of masters, the number of nodes, the number of cores, and the number of experiments which will be used for the output file name. For the number of nodes and the number of cores, it is also necessary to repartition data according to the number of cores and the number of existing nodes.

Then, we perform a main computation in PSO, which is to calculate fitness function by using Hamming Distance in Apache Spark. In this Big Data Platform computing model, the calculation of the hamming distance will be made into a UDF so that the function can run on partitions on each existing node. The program code for this stage can be seen in Figure 6. The program code for UDF computing Hamming Distance.

```
def calcFitness(popu, dimension, barcodeLength):
    start_time = time.time()
    column_names = []

    firstSequence = df_sequence.first()

    df = df_sequence.filter(df_sequence.sequence != firstSequence.sequence)
    df = df.withColumn("fitness",udf_fitness(firstSequence.sequence, popu, barcodeLength)(df_sequence.sequence))
    for i in range(popu.shape[0]):
        column_names.append(str(i))
    df = df.rdd.map(lambda x:[int(y) for y in x['fitness']]).toDF(column_names)

    finalFitness = df.groupBy().sum().rdd.map(list).collect()[0]
    df.unpersist()

    time_elapsed = time.time() - start_time
    print("time Elapsed on calcFitness: ", time_elapsed)
    return finalFitness
```

Figure 6: Calculation of Fitness Function involving Hamming Distance by using UDF in Apache Spark

After the results were obtained, the next step was to create a barcode from the index obtained for the sequence data that is on the RDD in the data frame with the default function of the data frame itself. After the barcode results were obtained in the form of the data frame, the results were combined first using coalesce because the data was still divided into several partitions. The result file was saved directly into Google Cloud Storage and in this environment the data is not split into several parts.

3. Experimental study

There is a process in DNA barcoding that still uses conventional methods such as similarity check or motif discovery. The process currently used is by looking at the DNA sequence and then marking areas that are potential used as barcodes, which results in a low level of accuracy and requires a long time [18] coupled with the currently very large amount of biological data [3]. In bioinformatics, motifs discovery is very important because it represents conserved sequences that can be biologically meaningful. Motif Discovery is an important step towards understanding gene regulatory mechanisms. Motifs can represent patterns that activate or inhibit the transcriptional process and are responsible for regulating gene expression [17].

DNA Barcoding assists taxonomists in the identification, discovery, and genetic study of specimens to achieve certain goals, namely knowledge of species diversity and the degree of variation among species [10]. DNA Barcoding translates taxonomist knowledge of morphological diagnostic characters into a widely accessible format, DNA sequences, enabling more people to identify specimens. In addition to assigning specimens to known species, DNA barcoding can speed up the discovery of new species, as large sequence differences in animal DNA generally signify species status [13, 4].

3.1. Data collection

The data used in this study is a collection of RNA sequence data from Coronavirus-19 (SARS-CoV-2) obtained from online databases such as GISAID (https://www.gisaid.org/). In data collection, filtering was carried out only for sequence data that had more than 29000 base pairs with N entries less than or equal to 5%. The collected data were also grouped based on several continents for DNA Barcode searches, which are Asia, America, Europe, and Oceania. As of May 9, 2020, 15.929 RNA samples from the SARS-CoV-2 virus had been obtained. The data obtained can be seen in Table 1.

Table 1: SARS-CoV-2 Virus RNA Data

File Name	Number of Base Pairs	Number of Sequences	File Size
asia_hcov-19_2020_05_09.fasta	$\leq$ 30.643	1.150	43.627
europe_hcov-19_2020_05_09.fasta	$\leq$ 29.977	8.773	332.845
america_hcov-19_2020_05_09.fasta	$\leq$ 29.977	4.598	174.458
oceania_hcov-19_2020_05_09.fasta	$\leq$ 30.442	1.285	48.753
Total		15.806	599.683

3.2. Experiment scenario

In this experiment, we perform a scenario on a cluster with several worker nodes, each of which has 4 CPU cores. Worker nodes that will be used are 1, 2, 5, 10, 15, and 20 worker nodes. This scenario will be run on Dataproc Google Cloud Platform which is one of the cloud computing providers [24]. In the Dataproc, we could create several clusters with various specifications that we can use to perform computing via the cloud.

In running this scenario, the following parameters are needed:

- barcodeLength = 400. The barcodeLength parameter serves as an input parameter for the number of barcodes to be searched.
- maxIter = 100. This parameter serves to determine the maximum number of iterations to be performed by the particle swarm optimization algorithm.
- numPopulation = 50. This parameter serves to determine the number of populations to be generated as a matrix containing a random population index.
- numVar = 2. This parameter is used to determine the number of barcode results sought. This parameter will also determine the dimensions of the matrix containing the random population index.
- trim = 360. This parameter serves to determine how many base pairs will be cut at the beginning and end of the sequence or can be referred to as sequence trimming.

The data used in this study used all the files that previously mentioned with a total of 4 files that have a total size of 599,683 KB with the standard for finding barcodes using RNA sequences from severe acute respiratory syndrome-related coronavirus which has been aligned to adjust RNA from SARS-CoV-2.

4. Result and discussion

Based on the previously designed scenario, Table 2 shows the results of the experiments that have been carried out. There are 4 columns, which are file, file size, number of worker nodes used, and time required to complete the computation. It shows the results of the summary result which contains 4 columns, which are the initial index of the barcode found, the barcode similarity percentage, the status of the founded barcode, and the time required for the computation. After that, the change in the efficiency of each experiment in this scenario was compared.

Table 2: Computational cost as experimental Result

No	File	File Size (KB)	Number of Worker Nodes	Time (second)
1	asia_hcov-19_2020_05_09.fasta	43.627	1	892.92
			2	459.14
			5	306.18
			10	315.07
			15	328.84
			20	378.94
2	europe_hcov-19_2020_05_09.fasta	332.845	1	4,217.23
			2	2,776.63
			5	2,275.57
			10	491.96
			15	344.92
			20	285.55
3	america_hcov-19_2020_05_09.fasta	174.458	1	2,366.65
			2	1,468.71
			5	903.52
			10	289.21
			15	216.96
			20	188.07
4	oceania_hcov-19_2020_05_09.fasta	48.753	1	945.24
			2	449.57
			5	216.71
			10	128.33
			15	113.14
			20	106.44

From Table 2, it can be seen that as the number of cores used in the cluster increases, the computing process itself will also accelerate. This shows that the number of cores will affect the length of time required for computing. The total time required for each file of each experiment is shown in Figure 7.

From the picture above, it can be seen that with the increase in the number of cores used, the increase in speed also varies. It can be seen that the line of increasing speed of the number of cores used 12, 16, and 32 for each worker node are smoother than the lines of increasing the number of cores used of 2, 4, and 8 for each worker node. It can be proven by looking at the efficiency value of each experiment carried out using the previously described formula. The calculation results can be seen in Table 3.

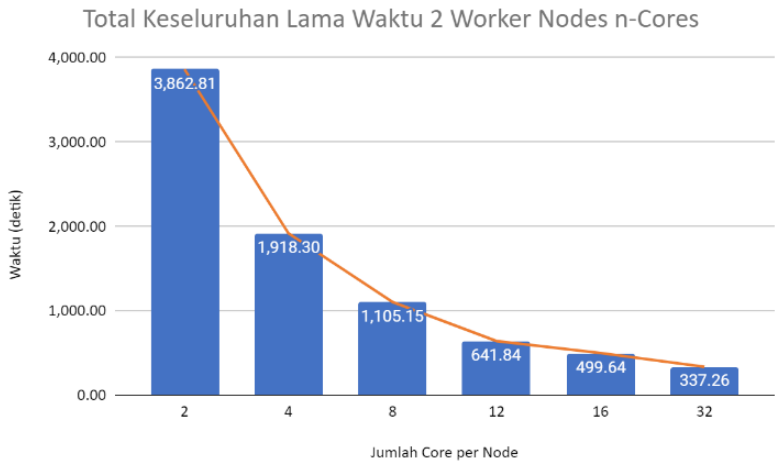

Figure 7: Comparison of the Speed of the Number of Cores against Time

Table 3: Speedup and Efficiency Calculation Results per Worker Nodes

No	Number of Worker Nodes	Time (second)	Speedup	Efficiency
1	1	8.472.48	1.00	1.00
2	2	5,154.08	1.64	0.82
3	5	3,760.19	2.25	0.45
4	10	1,276.07	6.64	0.66
5	15	1,039.73	8.15	0.54
6	20	950.48	8.91	0.45

From the table above, it can be seen that the number of worker nodes used will affect the length of time required for computing itself. The fastest time required to perform this computation is 950.48 seconds which is equivalent to 15.48 minutes which shows the speedup value or speed increase of 8.91 times faster than the first experiment that using only one worker node.

However, what is interesting is that increasing the number of worker nodes does not guarantee computational efficiency. This can be seen in the 'efficiency' column where the efficiency of 1 worker nodes to 2 worker nodes decreases from 1 to 0.82 and the efficiency of 2 worker nodes to 5 worker nodes also decreases from 0.82 to 0.45. Although the efficiency value has shown an increase as seen from the change in efficiency of 5 worker nodes to 10 worker nodes which increased from 0.45 to 0.66, the change in efficiency value decreased after that, as in the number of 10 worker nodes to 15 worker nodes and also to 20 worker nodes, which is from 0.66 to 0.54 and then to 0.45.

Based on the explanation above, it can be concluded that with the increase in the number of worker nodes used, the computational process will experience a significant acceleration. However, increasing the number of worker nodes used does not guarantee an increase in the efficiency of using these worker nodes. As shown in Figure 8 above, the acceleration of the time required for the computational process does not move linearly with the increase in the number of worker nodes used.

5. Conclusion

After researching DNA barcodes with particle swarm optimization algorithms using Apache Spark SQL, several conclusions were obtained that showed the main contributions of this research. Firstly, this study succeeded in designing a computational model on standalone using Python language to

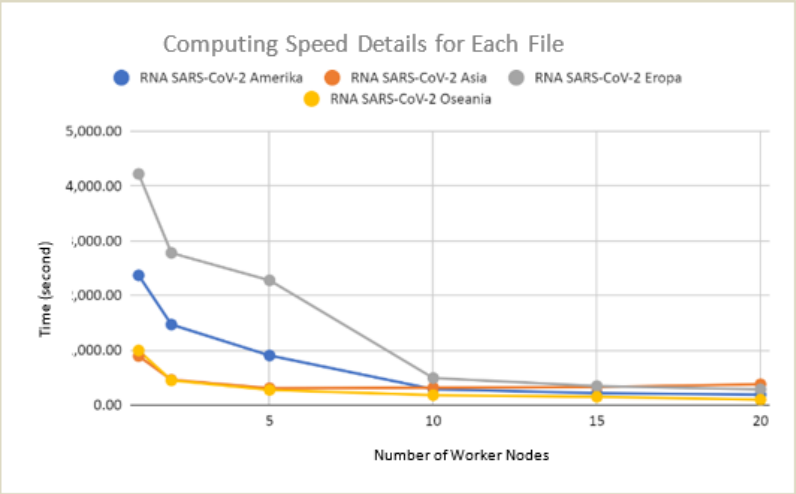

Figure 8: Computing Speed Details for Each File

search for DNA barcodes using particle swarm optimization algorithm using Apache Spark SQL with several stages, which are preprocessing, particle swarm optimization algorithm, hamming distance calculation, and result validation. Moreover, it succeeded in modifying the computing model on standalone using the Python language so that it can be run on a cluster on a big data platform using Apache Hadoop and Apache Spark SQL which involves several environments, which are the user's computer, Google Cloud Platform, Google Cloud Storage, and Apache Spark. This research has conducted 12 experiments that were divided into 2 scenarios, which are experiments with 4 cores with different number of worker nodes and experiments with 2 worker nodes with different number of cores. From the experimental results, it can be concluded that the more worker nodes or cores used will not guarantee the increase in efficiency from the use of resources in the form of worker nodes and cores in computing.

There are several suggestions that researchers can convey to be carried out in the future. Future research could result in much better programs in terms of speed by using the following methods: Knuth-Morris-Pratt algorithm [20] and machine learning methods [1, 21, 22].

References

[1] H. Alasker, S. Alharkan, W. Alharkan, A. Zaki, and L.S. Riza, *Detection of kidney disease using various intelligent classifiers*, IEEE 3rd International Conference on Science in Information Technology (ICSITech). 2017, pp.681–684.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts and P. Walter, *Molecular Biology of the Cell, 4th editio*, Garland Science, New York, 2002.

[3] T.K. Attwood, D.J. Parry-Smith and S. Phukan, *Introduction to Bioinformatics*, Benjamin-Cummings Publishing Company, 2001.

[4] S. Bandyopadhyaya, R. Vankayalapati, L. Rajanna, and S. Kulkarni, *DNA barcoding and its applications - A critical review*, C. J. Res. Dev. 1 (2013) 77–81.

[5] A. Bhattacharya and S. Bhatnagar, *Big data and apache spark: a review*, Int. J. Eng. Res. Sci. (2) (2016) 206–210.

[6] R. Desalle, M.G. Egan and M. Siddall, *The unholy trinity: taxonomy, species delimitation and DNA barcoding*, Philos Trans R Soc Lond B Biol Sci. 360(1462) (2005) 1905–1916.

[7] R.C. Eberhart and Y. Shi, *Particle swarm optimization: developments, applications and resources*, IEEE. Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), Korea, 2001.

[8] K. Ghulam, M. Husain, S. Salman, M. Tufail, S. Sukirno and A.S. Aldawood, *DNA barcoding of the fire ant genus solenopsis westwood (hymenoptera: formicidae) from the Riyadh region, the kingdom of saudi arabia*, Saudi J. Biol. Sci. 27 (1) (2020) 184–188.

[9] C.T. Hardin and E.C. Rouchka, *DNA motif detection using particle swarm optimization and expectation-maximization*, Proc. - 2005 IEEE Swarm Intell. Symp. USA, 2005.
[10] A. Imtiaz, S.A. Mohd Nor and D.Md. Naim, *Progress and potential of DNA barcoding for species identification of fish species*, Biodivers. J. 18(4) (2017) 1394–1405.
[11] M. Imran, R. Hashim and N.E.A. Khalid, *An overview of particle swarm optimization variants*, Procedia Eng. 53 (2013) 491–496.
[12] J. Kennedy and R. Eberhart, *Particle swarm optimization*, IEEE International Conference on Neural Networks, Australia. 1995, pp. 1942–1948.
[13] K.C.R. Kerr, M.Y. Stoeckle, C.J. Dove, L.A. Weigt, C.M. Francis and P.D.N. Hebert, *Comprehensive DNA barcode coverage of North American birds*, Mol. Ecol. Notes. 7 (2007) 535–543.
[14] X. Li, F. Yang, R. Henry, M. Rossetto, Y. Wang and S. Chen, *Plant DNA barcoding: From gene to genome*, Biol. Rev. Camb. Philos. Soc. 90 (2014).
[15] L.I.U. Miao, L.I. Xi-wen, L. Bao-sheng, L.U.O. Lu and R.E.N. Yue-ying, *Species identification of poisonous medicinal plant using DNA barcoding*, Chin. J. Nat. Med. 17 (2019) 585–590.
[16] R. Poli, J. Kennedy and T. Blackwell, *Particle swarm optimization: An overview*, Swarm Intell. 1 (2007) 33–57.
[17] N. Qader and H.K. Al-Khafaji, *Motif discovery and data mining in bioinformatics*, Int. J. Comput. Technol. 13 (2014) 4082–4095.
[18] L.S. Riza, F.D. Pratama, E. Piantari and M. Fashi, *Genomic repeats detection using boyer-moore algorithm on apache spark streaming*, Telkomnika. 18 (2020) 783–791.
[19] L.S. Riza, M.B.A. Prabowo, E. Junaeti, A.G. Abdullah and K.A. Fariza, *Development and experimentation of R package metaheuristicOpt on continuous optimization*, Int. J. Eng. Sci. Technol. 16 (2021) 1006–1018.
[20] L.S. Riza, A.B. Rachmat, Munir, T. Hidayat and S. Nazir, *Genomic repeat detection using the knuth-morris-pratt algorithm on R high-performance-computing Package*, Int. J. Comput. Appl. 11 (2019) 94–111.
[21] L.S. Riza, A. Janusz, C. Bergmeir, C. Cornelis, F. Herrera, D. Ślezak, and J.M. Benítez, *Implementing algorithms of rough set theory and fuzzy rough set theory in the R package roughSets*, J. Inf. Sci. 287 (2014) 68–89.
[22] L.S. Riza, F.S. Anwar, E.F. Rahman, C.U. Abdullah and S. Nazir, *Natural language processing and levenshtein distance for generating error identification typed questions on TOEFL*, J. Comput. Soc. 1 (2020) 1–23.
[23] L. Shao and Y. Chen, *Bacterial foraging optimization algorithm integrating tabu search for motif discovery*, IEEE Int. Conf. Bioinf. Biomed. USA. (2009) 415–418.
[24] P. Srivastava and R. Khan, *A review paper on cloud computing*, Int. J. Adv. Res. Comput. Sci. Softw. Eng. 8 (2018).
[25] D. Wang, D. Tan, and L. Liu, *Particle swarm optimization algorithm: an overview*, Soft Comput. 22 (2018) 387–408.
[26] D. Yang and J. Wang, *UPNT: uniform projection and neighbourhood thresholding method for motif discovery*, Int. J. Bioinforma. Res. Appl. 4 (2008) 96–106.
[27] C.H. Yang, Y.T. Liu and L.Y. Chuang, *DNA motif discovery based on ant colony optimization and expectation maximization*, Int. Multi Conf. Eng. Comput. Sci. 2011. 1 (2011) 169–174.