***I.J. Information Technology and Computer Science*, 2023, 4, 10-18**
Published Online on August 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2023.04.02

MOTIFSM: Cloudera Motif DNA Finding Algorithm

Tahani M. Allam
Computer and Control Department, Faculty of Engineering, Tanta University, Tanta, Egypt
E-mai: tahany@f-eng.tanta.edu.eg
ORCID iD: https://orcid.org/0000-0001-9624-2984

Received: 11 March 2023; Revised: 03 May 2023; Accepted: 17 June 2023; Published: 08 August 2023

Abstract: Many studying systems of gene function work depend on the DNA motif. DNA motifs finding generate a lot of trails which make it complex. Regulation of gene expression is identified according to Transcription Factor Binding Sites (TFBSs). There are different algorithms explained, over the past decades, to get an accurate motif tool. The major problems for these algorithms are on the execution time and the memory size which depend on the probabilistic approaches. Our previous algorithm, called EIMF, is recently proposed to overcome these problems by rearranging data. Because cloud computing involves many resources, the challenge of mapping jobs to infinite computing resources is an NP-hard optimization problem. In this paper, we proposed an Impala framework for solving a motif finding algorithms in single and multi-user based on cloud computing. Also, the comparison between Cloud motif and previous EIMF algorithms is performed in three different motif group. The results obtained the Cloudera motif was a considerable finding algorithms in the experimental group that decreased the execution time and the Memory size, when compared with the previous EIMF algorithms. The proposed MOTIFSM algorithm based on the cloud computing decrease the execution time by 70% approximately in MOTIFSM than EIMF framework. Memory size also is decreased in MOTIFSM about 75% than EIMF.

Index Terms: Cloud Computing, Motif, DNA, Impala, Cloudera.

1. Introduction

DNA finding algorithms generally produced a high storage and computational requirements [1, 2]. Cloud computing-based services deal with these computational requirements within in an efficient way. Cloud computing offers many service providers such as Google cloud, Microsoft Azure, and AWS [3, 4]. Amazon Web Service provides a powerful computing resources, computing elastic and free data storage space [5, 6]. The computational resources in DNA finding algorithms need to be available to the users [7]. The Impala based cloud framework is an effective way for a dynamic tool to extract an identical string of motifs. It supports a free online storage size for motif datasets. Also, we can use different motif string format in Impala framework to extract an identical string motif. The motif strings can used, uploaded or inserted from user on the cloud directly. Our Impala framework identifies common important motifs and their best matches and provides users with the best matches for each motif for further analysis.

DNA motif is a sequencing technology generates a huge amount of data which present challenges for researchers. One of these challenges appear on a real-time data which increase the space storage. Other challenge is appeared on the motif analysis which need an efficient tool to get a good performance in many computational resources. Impala based cloud computing can supports a DNA finding algorithms which required a high computational resource.

The previous framework called EIMF, used a proposed algorithm which scan the position of the pattern nucleotide sequence, save the position of this pattern in a raw list related to the next nucleotide [7]. We make enhancement on the proposed EIMF to decrease the execution time and increases the sequence size. Also, the enhanced EIMF is tested on Impala based Cloud which increase the efficiency of the algorithm.

The main contributions of this paper are:

- Impala cloud computing can support a DNA finding algorithms which required a high computational resource.
- Impala based Cloud which increase the efficiency of a DNA finding algorithms.
- The average execution time of MOTIFISM framework is decreased and the performance become better than EIMF.
- The good performance of MOTIFISM framework is observed clearly when the data size increased.

The rest of this article is organized as follows: the second section provides a summary of Impala from the user's perspective, highlighting how it differs from typical RDBMSs. Section 3 represent the DNA motif strings. Section 4 depicts the Impala motif finding algorithms' general architecture. In Section 5 Impala's motif finding algorithms performance is evaluated. Section 6 is a conclusion. In the following subsections we discussed the impala cloudera and DNA motifs.

1.1. Impala Cloudera

Impala is a popular engine in Cloudera's Hadoop Distribution (CDH), and it's also available in other Hadoop distributions [8, 9]. Cloudera Impala is a cutting-edge, open-source MPP SQL engine built specifically for Hadoop data processing [8]. Impala enables low latency and high concurrency for read-only BI/analytic queries on Hadoop, which batch frameworks like Apache Hive do not. Impala aims to combine the familiar SQL support and multi-user performance of a typical analytic database with Apache Hadoop's scalability and extensibility, as well as Cloudera Enterprise's production-grade security and administration features. It keeps Hadoop's extensibility by using standard components (HDFS, HBase, Metastore, YARN, Sentry) and can read the majority of widely used data formats (e.g. Parquet, Avro, RCFile). Impala has a distributed design based on daemon processes that are responsible for all elements of query execution and operate on the same computers as the rest of the Hadoop infrastructure to decrease delay, such as that caused by using MapReduce or reading data remotely. Depending on the workload, performance is comparable to or better than that of commercial MPP analytic DBMSs. Impala is the fastest SQL-on-Hadoop solution, especially when multi-user workloads are involved as we see in motif finding algorithm [9].

Unlike typical relational database management systems, where the query processing and the underlying storage engine are tightly connected, it is divorced from the underlying storage engine. Figure 1 illustrates Impala's high-level architecture [9]. Figure 1 covers Impala’s three main services:

- Impalad: This daemon, which is the brains of Impala, is installed on each cluster node. A random node receives a SQL request from a client and then takes over as the coordinating node. Executing the query, gathering the incomplete results, and returning the aggregated results to the client fall under the purview of this node. The impalad daemon is in charge of answering the inquiry.
- Statestore service: This element keeps tabs on the Impala cluster's condition. All of the Impala daemons receive the updated health information. The other daemons will be alerted if a node fails for any reason, and queries will avoid the dead node in the future.
- Catalog service: Any changes to the metadata are distributed to all daemons via this service. Every node receives any database updates, such as the addition of a column to a table. An explanation of a performance profile concludes the chapter. Several tools produce a performance profile that provides a thorough insight of how the query is run and where the bottlenecks are. Once you are familiar with the various services, the profile will make much more sense. By the end of the chapter, you'll be able to spot potential query execution bottlenecks, avoid them, and improve query performance.

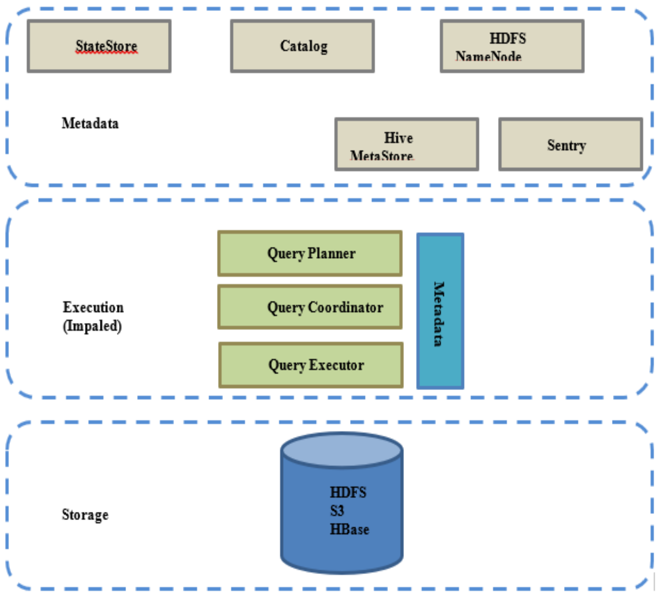


Fig.1. The Impala Architecture

The Impala daemon (impalad) service oversees both receiving client queries and organizing their execution throughout the cluster, as well as executing individual query fragments on behalf of other Impala daemons. When an Impala daemon manages query execution in the first role, it is referred as the query's coordinator. All Impala daemons, on the other hand, are symmetric, and they may play any role. This feature aids with load balance and fault tolerance.

Every system in the cluster that is also running a datanode process - the block server for the underlying HDFS deployment - has one Impala daemon, thus there is generally one Impala daemon on each machine. This enables Impala to take advantage of data locality and read blocks from the file system without the need for a network connection.

Impala's meta-data publish subscribe service (statestored) distributes cluster-wide metadata to all Impala processes. The coordination and synchronization of cluster-wide information is a key problem in the architecture of an MPP database that is meant to run on hundreds of nodes. All nodes in Impala's symmetric-node architecture must be able to accept and execute queries. To schedule queries effectively, all nodes must have up-to-date copies of the system catalogue and a current view of the Impala cluster's membership. We might deal with this issue by creating a separate cluster-management service that includes ground-truth copies of all cluster-wide data. Impala daemons could then query this store in a lazy manner, ensuring that all queries received current results.

Finally, Impala's catalog repository and metadata access gateway is handled by the Catalog daemon (catalogd). Impala daemons can use catalogd to run DDL instructions that are reflected in external catalogue stores like the Hive Metastore. The statestore broadcasts changes to the system catalogue. Impala's catalog service uses the statestore broadcast method to deliver catalog metadata to Impala daemons and performs DDL operations on their behalf. The catalog service gathers data from third-party metadata stores (such as the Hive Metastore or the HDFS Namenode) and organizes it into an Impala-compatible catalog structure.

1.2. DNA Motifs

The binding sites for transcription factors in deoxyribonucleic acid (DNA) is an important task to identify the identical elements [10]. The latest advances in genome sequence availability and high-throughput gene expression analysis techniques allow the development of computational methods for motif discovery. The gene is the basic unit of genetic information in DNA and is defined as a sequence of bases used as a template for a replication process called transcription [11]. The main idea of gene expression is that every gene contains the information to produce protein. Gene expression begins with the combination of multiple protein factors (called transcription factors) with enhancer and promoter sequences. Transcription factors regulate gene expression by activating or repressing transcription mechanisms. In biology, identifying the processes that control gene expression is a big issue. Discovering regulatory elements, particularly transcription factor binding sites in DNA, is a significant job in this issue. One of the most difficult issues in molecular biology and computer science is finding patterns in DNA sequences. The issue may be stated in its most basic form as follows: given a series of sequences, identify an unknown pattern that appears frequently. A simple enumeration of all m-letter patterns that exist in the sequences yields the answer if a pattern of m letters long appears exactly in every series. When working with DNA sequences, however, it is not as straightforward since patterns might contain nucleotide changes, insertions, and deletions [12].

A DNA motif is a nucleic acid sequence pattern with biological relevance, such as DNA binding sites for regulatory proteins, such as transcription factors. The pattern is generally short (5 to 20 bp) and has been seen in many genes or multiple times within a gene [10]. DNA motifs are frequently linked to structural motifs in proteins. Motifs can be found on both the positive and negative strands of DNA. Transcription factors do bind to double-stranded DNA directly. Motifs might appear in zero, one, or several instances in a sequence. The motif identification issue is the task of discovering overrepresented motifs as well as conserved motifs from orthologous sequences that are strong candidates for being transcription factor binding sites given a set of DNA sequences. There have been several algorithms designed to detect DNA motifs. Most of these algorithms are designed to extract motifs from a single genome's regulatory region of multiple coregulated genes. Coexpression of genes is thought to result mostly through transcriptional coregulation. The majority of previous literature divided motif finding algorithms into two groups based on the combinatorial approach used in their development. The first approach is a word-based (string-based) methods that mostly rely on exhaustive enumeration, it is counting and comparing nucleotide frequencies. The second approach is used the maximum-likelihood principle or Bayesian inference to estimate the model parameters in probabilistic sequence models. Our previous algorithm, called EIMF, is worked depending on the string-based methods [6]. The new proposed algorithm changes the method to enhance the computational resources and execute it on Impala framework.

2. Methodology: Proposed Cloud Motifsm Algorithms for Finding Motifes

We divide available motif discovery methods into three categories based on the kind of DNA sequence information used by the algorithm to deduce the motifs: (1) those that utilize promoter sequences from coregulated genes from a single genome, (2) those that use orthologous promoter sequences of a single gene from several species (phylogenetic footprinting), and (3) those that use promoter sequences of coregulated genes as well as phylogenetic footprinting [11].

We present a novel Cloudera motif finding algorithm in this work that uses a previous algorithm presented in paper [7] and a Cloudera Impala framework to overcome the difficulties. The suggested approach, called ClouderaMotif, uses a dynamic programming technique based on cloud to discover identical string motifs in a biological sequence or sequences of all lengths, starting with K= 2 nucleotides and increasing as there are repeated identical patterns with length K=K+1. When all identical string motifs with all conceivable lengths are obtained, or when the maximum necessary length Kmax is reached, the issue is solved [7]. Large datasets are often obtained from publicly available sources for bioinformatics research, which is then used to run experiments utilizing in-house infrastructure [12, 13, 14]. The major issue in the identical motif finding algorithm is a shortage of computer resources to complete experiments in a reasonable amount of time.

CDP Private Cloud now supports Cloudera Machine Learning, Cloudera's machine learning and AI platform [15, 16]. Cloudera Machine Learning combines self-service data science and data engineering into a single, portable service enabling multi-function analytics on data everywhere as part of a business data cloud. Cloudera Machine Learning (CML) is a new cloud-native machine learning solution from Cloudera, designed specifically for CDP. Clusters, also known as ML workspaces, are created by the CML service and operate natively on Kubernetes. Each ML workspace allows data scientists to create, test, train, and deploy machine learning models for creating predictive applications using data from the business data cloud [17,18,19]. Figure 2 shows how a machine learning project's deploy whole lifetime, from research to deployment, using Cloudera Machine Learning.

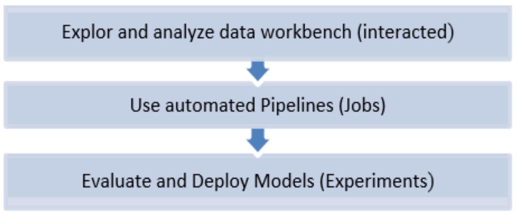

Explor and analyze data workbench (interacted)
Use automated Pipelines (Jobs)
Evaluate and Deploy Models (Experiments)

Fig.2. The Lifetime ML project in CML

2.1. *Explore and Analyze Motif String*

There are several online databases of DNA motifs listed with a short description of each one [10, 15]. We will discuss the algorithm using the following sample sequence data ’ACTCAGCTACCTCAGTACTGACTCAG’. Depending on the string-based methods each nucleotide position in the sample sequence is indicated in Figure 3.

Id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
String	A	C	T	C	A	G	C	T	A	C	C	T	C	A	G	T	A	C	T	G	A	C	T	C	A	G

Fig.3. The sequence index of a sample data

The proposed algorithm depends on the different positions of the target pattern appeared and the next nucleotide after it. We will read a pattern of three (k) nucleotides which includes its position and the next nucleotide after it. In row related table, the positions of the target pattern will be saved with the next nucleotide position for the same nucleotide. For example, as shown in Figure 4, pattern ACT has a position on 0 and 22 related to the next nucleotide C in position 3 so we stored each (0, 22) in row table. For each 3k motif string the bigger position count is saved first. Figure 5 shows different 3k motif string GCT with the same algorithm proposed. There is only one position after GCT followed by nucleotide A in position 5.

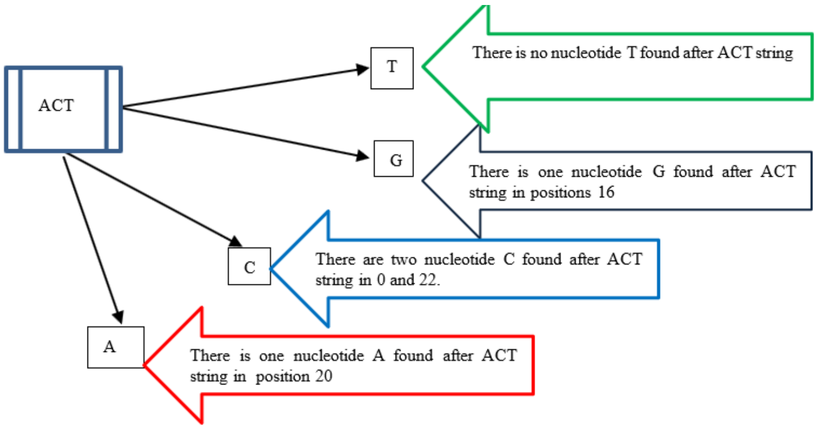

ACT
T
There is no nucleotide T found after ACT string
G
There is one nucleotide G found after ACT string in positions 16
C
There are two nucleotide C found after ACT string in 0 and 22.
A
There is one nucleotide A found after ACT string in position 20

Fig.4. Sample of the 3k subsequences ACT and their specified positions and next position

2.2. *Use Automated Pipelines*

This section explains how to automate analytics workloads using an integrated task and pipeline scheduling system with real-time monitoring, job history, and email alerts [14]. In one batch process, a job automates the actions of starting an engine, running a script, and tracking the outcomes. You may wish to divide up motif data science initiatives into numerous phases as they progress beyond ad hoc scripts. For such projects, Cloudera Machine Learning lets you run many tasks in a pipeline, where each job is dependent on the result of the one before it.

The pipeline created according to the previous section, the first job will create a pattern array that has all 3 k patterns of the four nucleotides A, C, G, and T, as well as sixty-four rows, as shown in Table 1. Each row will be associated with a pattern from the pattern array with the same index. Scanning the sequence data from start to finish, reading it as a pattern of three (k) nucleotides at a time, and recording the position of each nucleotide.

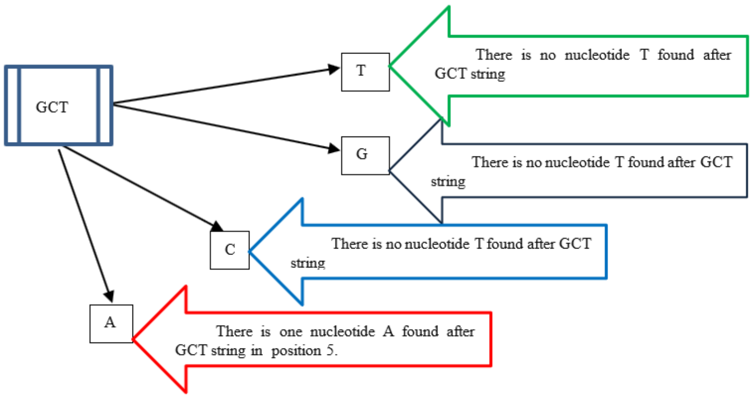

GCT
T
G
C
A
There is no nucleotide T found after GCT string
There is no nucleotide T found after GCT string
There is no nucleotide T found after GCT string
There is one nucleotide A found after GCT string in position 5.

Fig.5. Sample of the 3k subsequences GCT and their specified positions and next position

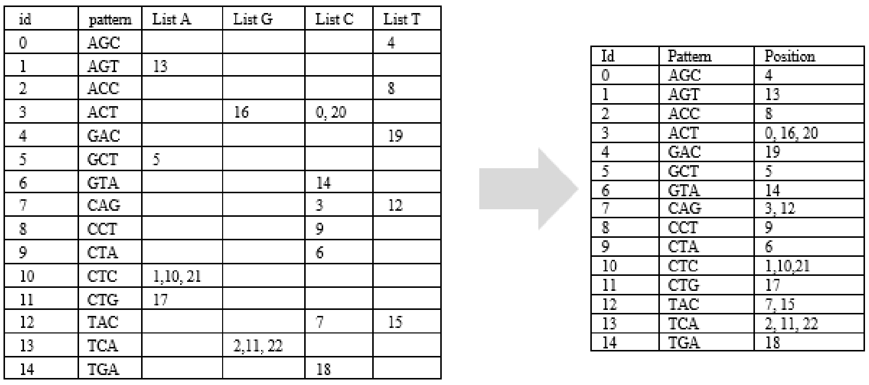
Table 1. First job "positions for the 3 bases subsequences"

id	pattern	List A	List G	List C	List T	Id	pattern	List A	List G	List C	List T
0	AAA					16	GAA				
1	AAC					17	GAG				
2	AAG					18	GAC				19
3	AAT					19	GAT				
4	AGA					20	GCA				
5	AGG					21	GCC				
6	AGC				4	22	GCG				
7	AGT	13				23	GCT	5			
8	ACA					24	GGA				
9	ACG					25	GGC				
10	ACC				8	26	GGG				
11	ACT		16	0, 20		27	GGT				
12	ATA					28	GTA			14	
13	ATG					29	GTC				
14	ATC					30	GTG				
15	ATT					31	GTT				
id	pattern	List A	List G	List C	List T	id	pattern	List A	List G	List C	List T
32	CAA					48	TAA				
33	CAC					49	TAC			7	15
34	CAG			3	12	50	TAG				
35	CAT					51	TAT				
36	CCA					52	TCA		2, 11, 22		
37	CCC					53	TCC				
38	CCG					54	TCG				
39	CCT			9		55	TCT				
40	CGA					56	TGA			18	
41	CGC					57	TGC				
42	CGG					58	TGG				
43	CGT					59	TGT				
44	CTA			6		60	TTA				
45	CTC	1,10, 21				61	TTC				
46	CTG	17				62	TTG				
47	CTT					63	TTT				

The second job save the scanned position in the list which related to the next nucleotide that we scanned in the first job in the raw equivalent to the pattern index. Table 2 shows the sequence data scanned as an example.

Third job focuses on the pattern array of each row, it checks if there is any equivalent's row in List A, List C, List G and List T. For example, if the count of positions stored in a List A is greater than the minimum number of required motifs in the problem conditions; concatenate the nucleotide A with the current pattern to get the next k motifs (in this case the 4k motif). Rescan the sequence and read k+1 nucleotide at the positions of founded k motifs that founded in the previous step and repeat the steps to get the required maximum motif length or reach the higher motif length found in the sequence. Table 3 shows the 4 bases motifs and their positions. We neglect any pattern which have less than two positions in the next nucleotide. We apply the same three jobs in the 5k and 6k motifs in tables 4 and 5.

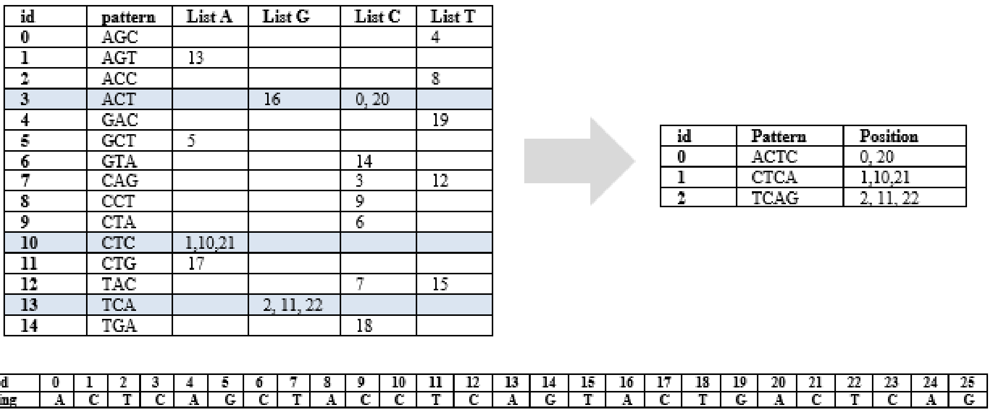
Table 2. Second Job "The 3 bases motifs and their positions"



id	pattern	List A	List G	List C	List T
0	AGC				4
1	AGT	13			
2	ACC				8
3	ACT		16	0, 20	
4	GAC				19
5	GCT	5			
6	GTA			14	
7	CAG			3	12
8	CCT			9	
9	CTA			6	
10	CTC	1,10, 21			
11	CTG	17			
12	TAC			7	15
13	TCA		2,11, 22		
14	TGA			18	

Id	Pattern	Position
0	AGC	4
1	AGT	13
2	ACC	8
3	ACT	0, 16, 20
4	GAC	19
5	GCT	5
6	GTA	14
7	CAG	3, 12
8	CCT	9
9	CTA	6
10	CTC	1,10,21
11	CTG	17
12	TAC	7, 15
13	TCA	2, 11, 22
14	TGA	18

Table 3. Third job " the 4 bases motifs and their positions"

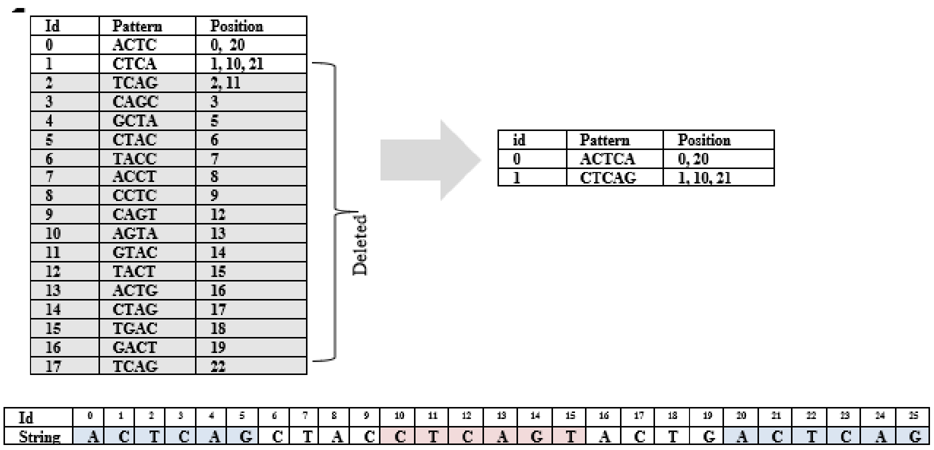


id	pattern	List A	List G	List C	List T
0	AGC				4
1	AGT	13			
2	ACC				8
3	ACT		16	0, 20	
4	GAC				19
5	GCT	5			
6	GTA			14	
7	CAG			3	12
8	CCT			9	
9	CTA			6	
10	CTC	1,10,21			
11	CTG	17			
12	TAC			7	15
13	TCA		2, 11, 22		
14	TGA			18	

id	Pattern	Position
0	ACTC	0, 20
1	CTCA	1,10,21
2	TCAG	2, 11, 22

Id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
String	A	C	T	C	A	G	C	T	A	C	C	T	C	A	G	T	A	C	T	G	A	C	T	C	A	G

Table 4. The 5 bases motifs and their positions related to next nucleotide"



Id	Pattern	Position
0	ACTC	0, 20
1	CTCA	1, 10, 21
2	TCAG	2, 11
3	CAGC	3
4	GCTA	5
5	CTAC	6
6	TACC	7
7	ACCT	8
8	CCTC	9
9	CAGT	12
10	AGTA	13
11	GTAC	14
12	TACT	15
13	ACTG	16
14	CTAG	17
15	TGAC	18
16	GACT	19
17	TCAG	22

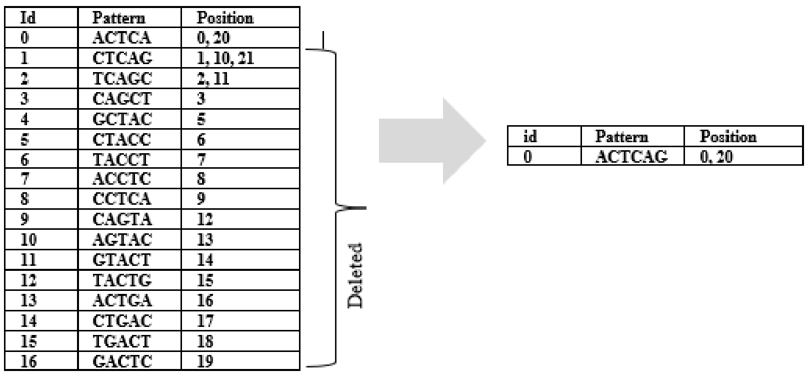
id	Pattern	Position
0	ACTCA	0, 20
1	CTCAG	1, 10, 21

Id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
String	A	C	T	C	A	G	C	T	A	C	C	T	C	A	G	T	A	C	T	G	A	C	T	C	A	G

3. Experimental Results

We assessed the performances of Impala model using the same criteria used in the baseline studies as described. We used the same data set that we used in the previous paper [6]. The new algorithm is implemented in Cloudera platform MOTIFISM, MOTIFSIM is a cloud-based software that runs on Amazon Web Services (AWS). This cloud-based version also enables researchers to use AWS computational resources to find similarities in different large-scale DNA motif data sets generated by next-generation sequencing technologies. The used data can be downloaded from http://bio.cs.washington.edu/assessment/download.html. We run the new tool and compared the average execution time in ClouderaMotif MOTIFISIM framework with the previous modified EIMF algorithm to find all identical string motifs with size up to 40 nucleotides.

Table 5. The 6 bases motifs and their positions related to next nucleotide



Id	Pattern	Position
0	ACTCA	0, 20
1	CTCAG	1, 10, 21
2	TCAGC	2, 11
3	CAGCT	3
4	GCTAC	5
5	CTACC	6
6	TACCT	7
7	ACCTC	8
8	CCTCA	9
9	CAGTA	12
10	AGTAC	13
11	GTACT	14
12	TACTG	15
13	ACTGA	16
14	CTGAC	17
15	TGACT	18
16	GACTC	19

id	Pattern	Position
0	ACTCAG	0, 20

In EIMF algorithm the average execution time is evaluated using windows 10 and has Intel core i7 processor 2.67 GHz and 16 GB Ram. The data set is divided into three groups: the first comprises data files of modest size, the second contains data files of intermediate size, and the third has a bigger data set. Table 6 shows the average execution time in MOTIFISM frameworks and EIMF in group 1 of data sets to find identical string motifs of length up to 40 nucleotides. Also, we show the percentage of improvements between EIMF and MOTIFSM frameworks which it decreased by 70% approximately in MOTIFSM than EIMF framework.

Table 6. Average execution time in seconds for the MOTIFISM framework and EIMF in the Group 1 dataset

Sequence	Group 1 (N0. Of Nucleotide)	EIMF	MOTIFISIM	Improved Percentage %
mus06r	1,500	0.010	0.007	**70**
dm06r	3,000	0.044	0.042	**95.45**
yst04r	7,000	0.013	0.010	**76.9**
hm26r	9,000	0.014	0.010	**71.42**
yst09r	16,000	0.013	0.010	**76.9**
hm01r	36,000	0.025	0.017	**68**
hm20r	70,000	0.036	0.022	**61.1**

Table 7 shows the average execution time in seconds of MOTIFISM framework and EIMF in intermediate-sized dataset in group 2 to find identical string motifs up to 40 nucleotides in length.

Table 7. Average execution time in seconds for the MOTIFISM framework and EIMF in intermediate-sized dataset

NCBI RefSeq	Size (MB)	EIMF	MOTIFISIM
NC 006998.1	0.19	0.22	0.19
NC 004432.1	1.36	1.903	1.805
NC 006814.3	1.99	2.624	2.064
NC 013665.1	2.96	4.392	3.994
NC 015186.1	3.75	6.034	5.897
NC 000962.3	4.41	5.933	5.023
NC 013421.1	5.06	7.306	6.675
NC 015675.1	6.88	10.602	9.665
NC 009142.1	8.21	13.457	12.865
NC 003281.10	13.78	27.489	25.005
NC 003280.10	15.28	30.588	28.776
NC 003283.11	20.92	45.853	42.874

Table 8 shows the average execution time in minutes of MOTIFISM framework and EIMF to find all identical string motifs of length up to 40 nucleotides in big biological sequences.

Table 8. Average execution time in minutes for the MOTIFISM framework and EIMF in big-sized dataset

NCBI RefSeq	Size (MB)	EIMF	MOTIFISIM
NC 000085.6	61.43	2.14	1.97
NC 000083.6	94.99	3.561	2.04
NC 000077.6	122.08	4.342	2.99
NC 024468.2	150.98	10.267	6.98
NC 000069.6	160.04	6.443	4.08
NC 000067.6	195.47	7.957	5.86
NC 024461.2	235.67	17.034	11.84
NC 024459.2	307.04	21.589	18.54

In table 9, we show the memory size in different nucleotide groups. It shows a percentage improvement about 75% between Cloudera and EMIF frameworks.

Table 9. Memory size (mb) in emif and motifisim frameworks

Dataset Group	No. of nucleotide	Size (MB) in EMIF	Size (MB) in MOTIFISIM
Group 1	142,500	20	15
Group 2	568,900	84.79	66.65
Group 3	15,250,000	1327	120.9

4. Result Explanation

As tables 7, 8 and 9 show, we can observe the MOTIFISM framework has a less execution time than previous EIMF framework in three groups of datasets. The most reason of the good performance is related to the mechanism of Cloudera which works depends on a dynamic programming technique based on cloud to discover identical string motifs in a biological sequence or sequences of all lengths. In group 1, the time is decreased about 38% to 30 % compared with the previous EIMF algorithm. In the intermediate size the performance of MOTIFISM framework is better than EIMF is decreased about 6% to 13% only in all datasets. The third group contains a larger data set, the EIMF tool works only when we increase the machine memory to 16 GB Ram. The average execution time of MOTIFISM framework is decreased by about 8% to 32% and the performance become better than EIMF and, the sequence NC 024459.2 has a size 307.04 M executed in only 18 minutes by MOTIFISM framework. We found that the good performance of MOTIFISM framework is observed clearly when the data size increased.

In the MOTIFISM framework, reliability is high because it measures how well the findings can be repeated when the study is conducted under same circumstances. We discover that the MOTIFISM framework has excellent accuracy and reliability by examining the consistency of outcomes across time, among various observers, and among test components. MOTIFISM Since the results produced by validity are very valid, they accurately reflect the actual attributes, traits, and variances found in the physical data sets. One sign of a valid framework is high reliability.

However, the MOTIFISM paradigm helps to clarify several complex and perplexing sequence analysis problems and warns computer science students away from the most typical mistakes. This debate is heavily influenced by our own, hands-on experience with DNA motif analysis. We make an effort to go through the useful techniques for sequence and structural analysis that are essential to comparative genomics. After reading this article, we hope that interested readers will be motivated to continue learning about sequence analysis techniques by consulting more specialized publications, such as those in the reference list.

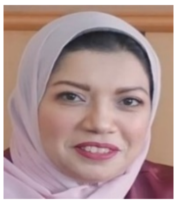
5. Conclusions

Notwithstanding exhaustive efforts make in diverse subject areas by theorists and experimentalists, the sequence motif, a short DNA pattern consisting of a smaller motif repeated multiple times, remains a difficult challenge for biologists and computer science researchers. MOTIFISM toolkits are designed supported algorithms that are numerous and complicated, associate our incomplete understanding of biological regulation doesn't forever give an adequate analysis of underlying algorithms. In this paper, we presented the proposed algorithm which is called Cloudera MOTIFISM that discovers all the identical string motifs dynamically. The proposed algorithm is designed to support the big data size sequences and we test it with sequence data up to 300 millions nucleotides in 20 minutes of execution time. The execution time depends on the size and number of existing identical string motifs in the sequence. We prove that the proposed MOTISM framework can solving a motif finding algorithms in single and multi-user based on cloud computing. Also, we used our proposed algorithm based on the cloud computing which decrease the execution time, memory size and increases the sequence size.

References

[1] Koppad, S., Annappa, B., Gkoutos, GV.: Cloud Computing Enabled Big Multi-Omics Data Analytics, BIOINFORMATICS AND BIOLOGY INSIGHTS, SAGE PUBLICATIONS LTD, September, 2021.
[2] Dai, L., Gao, X., Guo, Y. et al.: Bioinformatics clouds for big data manipulation. Biology Direct 7 (1): 43, 2012.
[3] De Oliveira Veras, Adonney A.a, De Sá, Pablo H.C.G.a. et al: Computational techniques in data integration and big data handling in omics, Omics Technologies and Bio-engineering: Towards Improving Quality of LifeVolume 1, Pages 209 – 222, 2018.
[4] Zhang, L., Gu, S., Liu, Y. et al.: Gene set analysis in the cloud. Bioinformatics 28 (2): 294–295, 2011.
[5] Oh, M., Park, S., Kim, S. et al.: Machine learning-based analysis of multi-omics data on the cloud for investigating gene regulations, Briefings in Bioinformatics, OXFORD UNIV PRESS, May, 11, 2021.
[6] Li, H., Weng, S., Tong, J. et al.: Composition of Resource-Service Chain Based on Evolutionary Algorithm in Distributed Cloud Manufacturing Systems, IEEE Access, Open Access, Volume 8, Pages 19911 – 19920, 2020.
[7] Abdelmenem S. Elgabry, Tahani M. Allam, and Mahmoud M. Fahmy.: An Identical Motif Finding Algorithm through Dynamic Programming, International Journal of Online and Biomedical Engineering, 2021.
[8] Google Cloud, https://cloud.google.com/. Accessed March , 2023.
[9] Cloudera Bringing Impala to AWS Cloud, https://www.datanami.com/2017/11/28/cloudera-bringing-impala-aws-cloud/, March 2, 2023.
[10] Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., et al.: Impala: A Modern, Open-Source SQL Engine for Hadoop. In Cidr 2015 Jan 4, Vol. 1, p. 9.
[11] Das, Modan K., Dai, H., A survey of DNA motif finding algorithms, *BMC Bioinformatics Open Access* Volume 8, Issue SUPPL. 71 November 2007.
[12] Karczewski, K.J., Fernald, G.H., Martin, A.R. et al. STORMSeq: an open-source, User-friendly pipeline for processing personal genomics data in the cloud. PLoS One 9 (1): 2014.
[13] Garbelini, JMC., Sanches, DS., Pozo, ATR.: Expectation Maximization based algorithm applied to DNA sequence motif finder, IEEE Congress on Evolutionary Computation, Proceedings Paper, Accessed in October 15, 2022.
[14] Petit, Robert A., Read, Timothy D.: Bactopia: A flexible pipeline for complete analysis of bacterial genomes, my Systems, Open Access, Volume 5, Issue 4 August 2020.
[15] Managing Jobs and Pipelines in Cloudera Machine Learning. Date published, CLOUDERA,: https://docs.cloudera.com/cdsw/1.9.2/jobs-pipelines/topics/cdsw-jobs-pipelines.html, 2020-07-16.
[16] Sparks, ER., Venkataraman, S., Kaftan, T., Franklin, MJ., Recht, B.: KeystoneML: Optimizing Pipelines for Large-Scale Advanced Analytics, INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE 2017), DOI 10.1109/ICDE.2017.109, IEEE 2017.
[17] Goudarzi, M., Palaniswami, Marimuthu S., Buyya, R., A Distributed Deep Reinforcement Learning Technique for Application Placement in Edge and Fog Computing Environments, IEEE Transactions on Mobile Computing, Open Access, 2021.
[18] Microsoft Azure, https://azure.microsoft.com/en-us/. Accessed March 2023.
[19] Hussein, E., Sadiki, R., Jafta, Y., Sungay, MM., Ajayi, O., Bagula, A.: Big Data Processing Using Hadoop and Spark: The Case of Meteorology Data, E-INFRASTRUCTURE AND E-SERVICES FOR DEVELOPING COUNTRIES (AFRICOMM 2019), DOI 10.1007/978-3-030-41593-8_13, Published 2020.

Authors' Profiles



Dr. Tahani M. Allam, a Lecturer at the Computers and Control Engineering Department, Faculty of Engineering, Tanta University, Egypt. Tahani was born in Kuwait, and her B.Sc., M.Sc., and Ph.D. degrees have taken from the Computers and Control Engineering Department, Faculty of Engineering, Tanta University in 2002, 2010, and 2018, respectively. Tahani works as a Consultant Engineer on Management Information Systems (MIS) Project, Tanta University, Egypt, from 2010 until now. Also, I am currently working as Director of the Functional Performance Evaluation Unit at Tanta University. Her research interests include Cloud Computing, Artificial Intelligence, Machine Learning, Security, and the Internet of Things. Tahani has published more than 6 articles in various refereed international journals and conferences. (email: Tahany@f-eng.tanta.edu.eg)
https://www.scopus.com/search/form.uri?display=basic#basic
https://www.webofscience.com/wos/woscc/basic-search
https://orcid.org/0000-0001-9624-2984
https://www.researchgate.net/profile/Tahani-Allam

How to cite this paper: Tahani M. Allam, "MOTIFSM: Cloudera Motif DNA Finding Algorithm", International Journal of Information Technology and Computer Science(IJITCS), Vol.15, No.4, pp.10-18, 2023. DOI:10.5815/ijitcs.2023.04.02