IJCI V9-01 (2022) 29–45
International Journal of Computers and Information (IJCI)
Available online at *https://ijci.journals.ekb.eg/*

Detecting the Behaviour of COVID-19 Based On Parallel Approach of Sequential Rule Mining Algorithm

Nesma Youssef [a, c*], Hatem Abdulkader[c], Amira Abdelwahab[b,c]

[a] *Department of Information System, Sadat Academy for Management Science, Cairo, Egypt,*
[b] *Department of Information Systems, College of Computer Science and Information Technology, King Faisal University, P.O. Box 400, Al-Ahsa 31982, Saudi Arabia*
[c] *Department of Information Systems, Faculty of Computers and Information, Menoufia University, Shibin Al Kawm 32511, Menoufia, Egypt*

Abstract

The COVID-19 (Coronavirus) is a catastrophic disease, as it causes a global health crisis. Due to the nature of COVID-19, it spreads quickly among humans and infects millions of people within a few periods in the world. It is critical to detect the behaviour of COVID-19 and the speed of its mutating rapidly for better improvement of medications and assists patients in preventing the progression of the disease. This paper examines the discovery of additional information and interest patterns in COVID-19 genome sequences. An enhanced non-redundant sequential rule algorithm is mined from frequent closed dynamic bit vector and sequential generator patterns simultaneously. It speedily discovers nucleotide rules and predicts the next one after eliminating un-candidates' sequential patterns early. Almost all genotyping tests are partial, time-consuming, and involve multi-step processes. So, an efficient parallel approach is implemented by utilizing multicore processor architecture to produce the sequential rules in less time required. The experimental results show that; the proposed Parallel Non-Redundant Dynamic closed generator (PNRD-CloGen) algorithm performs well in terms of execution time, computational cost, and scalability. It has better performance, especially for large datasets and low minimum support values, as it takes around half the time as the competing algorithm. So, it helps to monitor the strain progression of COVID-19 sequentially and enhance clinical management.

Keywords: : Sequential rule mining; Non redundant sequential rule; closed sequential patterns; sequential Generator patterns COVID-19.Main text

1. Introduction

The COVID-19 (coronavirus) has arisen as a cosmopolitan pandemic obstacle as it transmission rapidly among people. The Ministry of Health announced that COVID-19 is a universal epidemic on MAR 12, 2020. Once the Corona pandemic appeared, the trend quickly took preventive measures and isolated the infected to prevent the virus from spreading and reduce its reproduction to avoid entering the so-called second wave [1]. Health authorities have tended to develop treatment protocols, research and develop them, and find effective vaccines to provide humans with long-term immunity.

One of the most significant factors of the COVID-19's fast spread is the lack of specificity in clinical diagnostic procedures. Molecular approaches such as quantitative real-time reverse transcription-polymerase chain reaction, as well as serologic assays and viral throat swab testing, are required and widely utilized to identify COVID-19. Computed tomography (CT) scans and X-ray examinations might be used as the primary detection technique to assess COVID-19 severity, monitor infected patients' emergency cases, and forecast COVID-19 development [2]. However, time is often limited in emergencies and doesn't allow the experiments to be performed using the conventional manual diagnosis.

It was essential to understand the virus receptor recognition mechanism that regulates the spread and the incidence of infection. Many researchers study the genetic material (genome) to arrive at the evolutionary relationship among other viruses. The concept of the genome is an organism's genetic material [3]. Scientists were able to find four specific compounds of nitrogenous bases (nucleotides) that all living organisms shared in representing their existence and stored as a coded sequence. Nucleic acids are composed of (Adenine-A, Guanine- G, Cytosine-C, and Thymine-T). The COVID-19 genome sequence consists of a single nucleotide chain called ribonucleic acid (RNA). Also, the genome sequence has revealed the existence of multiple strains of the COVID-19 virus. Discovering the genome features is critical in helping biomedicine derive hypotheses about the feature's effect on disease development. This process suffers from a slow completion process and requires many resources and experience in the field. As the sequence of the COVID-19 genome is unrevealed early, the behaviour of COVID-19 becomes unclear in time.

Data mining techniques were crucial in accelerating the discovery of sequential patterns in the genome sequence by speeding up the discovery of possible solutions in genomic data [4]. An important field of data mining techniques is pattern mining. It is divided into two parts: itemset mining and sequential pattern mining (SPM). Implementing the SPM on sequential genomic data can produce new information on viral mutations, pathogenicity, and clinical symptoms. Furthermore, utilizing SPM to uncover essential hidden information in genomes might assist speed up the process of biological research. SPM is the extraction of frequent sequential patterns in large datasets [5]. It has just one measure called; the minimum support threshold (MinSup). The minSup parameter is the existence of items in the sequence database. It may be misleading and not sufficient for making a prediction. Sequential Rule Mining (SRM) presents a solution for this problem by considering another parameter called minimum confidence (MinConf) that measures the probability of the following pattern [6]. SRM proves its efficiency in predicting mutations in the coronavirus by predicting subsequent nucleotide rules. It led to improved clinical management and the development of the vaccine and effective protocols.

The mining process consists of two phases: first, mining the frequent sequential patterns. The second phase is mining the sequential rules depending on the first phase. As a result, academics are concentrating their efforts on improving the performance of mining the sequential patterns. Sequential patterns are found in three different concise representations [7]. The first is called Closed Sequential Patterns (CSP) mining. It is more efficient than mining the entire collection of sequential patterns. It retains all of the extracted data without altering the findings. CSP with a vertical data format is more efficient than CSP with a horizontal data format [8]. In the vertical data format, it has a substantial benefit because the dataset is scanned only once. As a result, it estimates each sequence's support count quickly. The second is Maximal Sequential Patterns (MSP) which have sequences that aren't contained in other closed sequential patterns. When compared it with closed sequential patterns, it produces a lesser number of sequential patterns. Many applications exist, like finding frequent sequences in texts, studying DNA progressions, and weblog mining [9]. The last is Generator Sequential Patterns (GSP). Compared with closed sequential patterns, sequential generator patterns (SGP) are more compact. Many SGP algorithms have been suggested include GenMiner, FEAT, and MSGPs [10]. GenMiner is a three-phase algorithm: compact, generate, and filter. In classification, generator patterns are a prioritized representation of the classes as well as model selection.

In big sequence datasets, the number of sequential patterns continues to rise. As a result, it has an impact on the efficiency of sequential rule mining. Many studies have attempted to eliminate sequential patterns that have no bearing on the ultimate mining result. They created a compressed sequence dataset by combining several concise representations of sequential patterns with efficient methods.

Most algorithms are depending on complicated data structure and repetitive tasks for mined sequential patterns. It causes time-consuming and CPU idle time. An enhanced algorithm is needed for quickly generating sequential rules, especially for large sequence datasets. With the rising number of generated candidates, it demands performing the same tasks on each sequence. It takes a lot of time.

The general objective of this paper is to detect the behaviour of the COVID-19 genome using the SRM algorithm through generating rules depending on mining two concise representations of patterns; closed dynamic bit vector and generator patterns. It helps in finding additional information from candidate patterns in the genome sequence, which is the first phase to find the rules of nucleotides. Additionally, predicting the following nucleotide rules in the genome sequences speedily. It will support the clinicians' understanding of the evolution

and mutation of the coronavirus. It assists in preventing the spread of the coronavirus and provides means of prevention.

An enhanced algorithm called Parallel Non-Redundant Dynamic closed Generator (PNRD-CLOGEN) is proposed for mining non-redundant sequential nucleotide rules in a parallel approach to reduce the execution time for generating the sequential rules. The genome dataset is examined in a reasonable amount of time using a mining parallel approach. A multi-core processor is one way to implement this strategy. On a single chip, it has several processor cores. It can boost productivity by performing multiple tasks at once [11]. In addition to this, the original algorithm is improved by using the dynamic bit vector structure directly to minimize the required time and generate rules with actual positions.

2. Related work

Data mining helps in knowledge discovery for making a prediction. It utilizes in many fields like quality control, manufacturing simulation, embedded systems, and biomedical applications such as genomic analysis, disease-gene analysis, death cases prediction, medication discovery, and prediction of adverse drug events. Sequential rule mining (SRM) is a data mining technique that aims to discover rules in large sequence databases. The main difference between association rule mining, sequential pattern mining, and sequential rule mining is that; association rule mining only discovers relations between items in large databases. Sequential pattern mining (SPM) finds the relationship between two frequent patterns [12]. It utilizes only one measure named support. The support measure is misleading for users as there is no assessment of the probability for the following sequence pattern. The sequential rule mining provides a solution for this problem by counting the confidence measure that calculates the subsequent pattern probability. It proves its efficiency in making a prediction [13]. It is based on the mining of frequent sequences to enable generating sequential rules. Agrawal and Srikant [14] propose the mining of sequential patterns through the use of AprioriAll.

Table 1. *Summary of the most relevant proposed algorithms of data mining in clinical area*

Ref	Title	Description	Algorithm	Benefits	Limitations
15 (2018)	Association Rule Mining of Polypharmacy Drug Utilization Patterns in Health Care Administrative Data Using SAS	This paper demonstrates how to utilize SAS Enterprise Miner's Association Node.	Association rule mining (ARM)	Provide extra insight into the data and discover patterns that standard statistical approaches are unable to detect.	_____
17 (2020)	Transitive Sequencing Medical Records for Mining Predictive and Interpretable Temporal Representations	In this paper, the sequential patterns revealed throughout the dimensionality reduction process were evaluated using an interactive graphical dashboard.	A transitive sequential pattern mining (TSPM) approach	Large-scale clinical databases can generate interpretable findings by including time and understanding the complexity inherent in the healthcare process.	Sequences identified were not further evaluated based on their accuracy.

Many researchers have been suggested enhanced algorithms for mining association relations, sequence patterns, and sequential rules in the healthcare area, as shown in Table 1. Dingwei Dai and Chris Feudtner [15] have discovered the co-utilization associations with drugs and detected patient characteristics with many poly-pharmacy patterns. Another enhanced algorithm proposed to conclude the importance of the association between patient's diagnosis and diagnostic exams to improve the decision-making of resources utilization in the emergency department [16]. For the SPM, there are many algorithms suggested in clinical management. One of the most recent methods is construction establishing temporal representation from EHR observation by utilizing clinical data from cohort patients [16]. Many enhanced algorithms have been proposed for mining

SRM, like an algorithm to discover associations for addressing the validity of the gateway hypothesis [17]. Also, KILGORE and Phillip CSR [18] proposed a sequential rule engine approach to detect insurance claims with fraud in the healthcare system. The focus of previous studies related to COVID-19 has been on predicting case numbers and categorizing COVID-19 patients from real-world X-ray databases. It requires sophisticated methods of deep neural networks that help in accurate diagnosis and prediction of severity of patients' cases [19,20], as seen in Table 2. Artificial intelligence has significant applications that limit disease spread efficiently [21-23]. Various researchers emerged to determine the risk of contagious disease and explore the mutation rate in patients' genome sequences [24,25]. Other researchers concentrate on evaluating the severity of cases to perform better clinician decision management [26,27]. The utilization of genomic sequence data particularly proves its importance in guidance the pandemic on time [4]. It's able to identify nucleotide patterns and understand the meaningful relations between them. It also reveals the genomes mutations that can impact virulence and infectivity [28-29]. However, the identification of genome characteristics is time-consuming and requires a lot of phases. All previous studies have concentrated on analyzing genomic data with complicated approaches that require high costs. It also suffers from slow processes to accomplish its goal [30,31].

Table 2. *Summary of the most recent sophisticated algorithms*

Ref	Title	Description	Algorithm	Benefits	Limitations
19 (2020)	Artificial Intelligence and COVID-19: Deep Learning Approaches for Diagnosis and Treatment	This study proposes an Artificial Intelligence-based solution to tackle infection (AI).	Deep Learning, Extreme Learning Machine (ELM)	Succeed in the fight against COVID-19 and its eventual extinction by building an armory of platforms, methods, tactics, and tools.	Need sophisticated approach that requires high costs.
20 (2020)	Exploring the Potential of Artificial Intelligence and Machine Learning to Combat COVID-19 and Existing Opportunities for LMIC: A Scoping Review	This paper discusses relieving pressure on private healthcare systems in low- and middle-income countries.	AI-driven tools	- COVID-19 detection, screening, and diagnosis were cost-effective and speedier - Reducing the load on healthcare systems.	- Caused time-consuming due to repeated operations - Needed to adjust the visualizations/tables
21 (2021)	Machine and Deep Learning towards COVID-19 Diagnosis and Treatment: Survey, Challenges, and Future Directions	Provide a survey with an overview of the existing state-of-the-art methodologies for ML and DL researchers.	Artificial Intelligence (AI)-based ML and DL methods	Discuss the challenges and possible guidance for avoiding the disruption of COVID-19.	_____
22 (2020)	Approaches Based on Artificial Intelligence and the Internet of Intelligent Things to Prevent the Spread of COVID-19: Scoping Review.	The goals of this study were to examine the present literature and discuss the usefulness of reported AI ideas for preventing an COVID-	AI-based approaches	The ten most recent AI approaches have been presented to provide suitable solutions for maximizing safety.	High investment is required to respond quickly to the disease's threats.

The previous researches proved the ability to detect the behaviour of COVID-19, but most of them take a long time in processing. Other researches rely on complicated approaches like deep learning and neural network that require high costs to accomplish the goal. Fewer researchers used the SPM algorithms, but anyone applied SRM algorithms in the field of detecting the behaviour of genome sequences, as shown in Table 3.

Table 3. *Summary of the proposed pattern mining algorithms*

Ref	Title	Description	Algorithm	Benefits	Limitations
25 (2020)	Intrusion detection and performance simulation based on improved sequential pattern mining algorithm	An improved sequential pattern mining approach is used to investigate intrusion detection and performance simulation.	Pattern mining algorithm	Combine data mining methods and the simulation outcome to accomplish the IDS that shows the methodology's effectiveness.	Time- consumed
26 (2020)	Time series prediction of COVID-19 by mutation rate analysis using recurrent neural network-based LSTM model	This paper discusses the mutation rate of the entire genome sequence acquired from patient datasets from various nations is investigated.	neural network-based Long Short Term Memory (LSTM) model	This model can be used to forecast daily mutation rates.	Suffer from accomplishing several processes that require extra time to reach the goal.
28 (2021)	Initial whole-genome sequencing and analysis of the host genetic contribution to COVID-19 severity and susceptibility	Present the first host genetic research in the Chinese population, based on deep sequencing and analysis of COVID-19 patients from the Shenzhen Third People's Hospital.	Genome Analysis Toolkit (GATK version 4.1.2)	Provide the genomic architecture of host-pathogen interaction For COVID-19 and other infectious and complicated illnesses.	Most complicated terms are used that led to ambiguity in understanding the utilized processes.
29 (2021)	Discovering symptom patterns of COVID-19 patients using association rule mining	The goal of this study was to find symptom patterns and general symptom rules, as well as disaggregated rules.	rule-based machine learning technique	Identify common symptoms and patterns in the identified rules.	Rules can be inferred by other rules and caused time consuming.

In this paper, an enhanced parallel algorithm named PNRD-CloGen is proposed to generate nucleotides sequential rules. There aren't any algorithms used the same methodology of mining nucleotide rules from two concise representations of sequential patterns through applying a parallel approach. It provides additional information to understand the virus behaviour from its origin passes to mutate phases. Additionally, a parallel approach (multicore) is implemented to minimize the execution time, which can help clinical decisions for better management of COVID-19 patients depending on the severity of the cases

3. Data structure of the proposed algorithm

SRM discovers the frequent patterns in the sequence database. A sequence is consists of ordered itemsets, as shown in **Table 4**. Since the mining procedure is composed of two phases, the first phase is mining frequent sequential patterns. The second is generating sequential rules based on the first phase. So, many researchers concentrate on the first phase to enhance the efficiency of the mining procedure performance. For the first phase, the genome sequence dataset appears in the form of detached sequences. Thus the SPM techniques are established to analyze sequences by counting each one, like a transaction, to perform them efficiently.

Table 4. *Sample sequence database*

SID	Sequence
S1	<AB(BD)E(EC)>
S2	<A(BD)G>
S3	<AB(FD)(BD)>
S4	<AD(DF)(AF)>
S5	<AD(AD)E>

3.1. *Blending closed with generator patterns*

The proposed approach for analyzing genome sequences of Covid-19 is based on mining two concise representations of patterns; the first is mining closed sequential pattern with a dynamic bit vector. It can compact the dataset without affecting the final result by applying the dynamic bit vector (DBV) structure in a vertical format. It refers to there being no super-sequence with the same support. It considers itemsets (sequences of a nucleotide of COVID-19) as several transactions and expresses the appearance of nucleotide with '1' and the absence with '0, as shown in Table 5. It achieved compaction by eliminating '0's from the beginning and the ending of the sequences. The (DBV) structure can calculate the support easily and combine the position or two items concerning the order by adding operation. For instance, when extending item <D> by implementing the ANDING operation to get <DB>, the result with DBV only is (111). When the DBV structure concern a position, it is clear that <DB> does not exist, and the correct result is (00000). The accurate result of the ANDING operation is <BD>, which has (111). For example, item <E> in sequence database, as shown in Table 4 appeared in sequences 1 and 5. The bit vector for <E> is (1,0,0,0,1) that the first appearance of item <E> in sequence 1 in the third and fourth position denoted as 3:{3,4}. Table 6 represents the DBV structure of item <E>.

Table 5. *Example of bit vector*

0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	0

The second concise representation is mining generator sequential patterns. Mining GSP means no subsequence with the same support. It reduces the time required by generating a smaller size of patterns than required in mining closed patterns. It starts with producing the frequent sequential patterns, then discovering sets of sequential generators that haven't subsequence equivalent to its support value. The GSP has a better pruning technique with more safety without time-consuming and highest cost by applying backward and forward pruning. It only checks the FSP to remove the non-generators from the candidate set. Combining generator patterns with closed patterns helps acquire additional information and generate sequential rules without any redundancy. The sequential rules are performed by using a generator pattern in antecedent and closed patterns in consequence.

Table 6. *DBV structure for item <E>*

Item	<E>	
Start bit	1	
Index	1	5
List of position	3:{3, 4}	3:{3}

It avoids the limitations of producing un-candidates that maximize the search space, particularly with the minSup lower values. It assists users in having better knowledge about the frequent sequences in large datasets. Thus, it can be performed efficiently in terms of runtime.

3.2. Multicore processor architecture

A parallel mining method is used to search large datasets in a short amount of time. One approach to accomplish this strategy is to use a multi-core CPU. It features many CPU cores on a single chip. It can boost productivity by doing multiple tasks at once. The multi-core processor has two or more cores are placed in the same physical package [32-33]. Each CPU core has its memory and shares the main memory, as seen in Figure 1. It is linked to external memory and has link controllers for interacting with other system components.

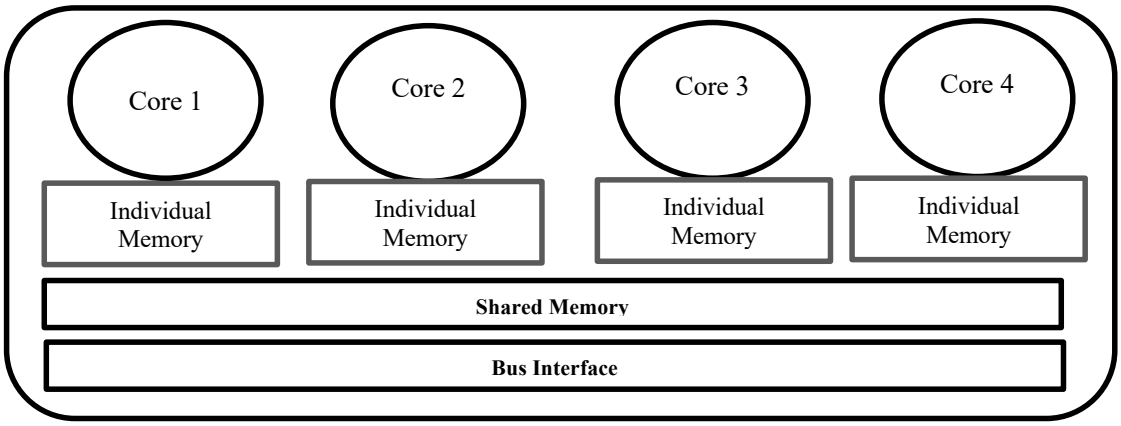

Fig. 1. Multicore (Quad-core processor) System

Each core has its L1 and L2 cache, with a shared cache L3. It maximizes resource consumption by sharing additional resources among these cores. These cores are read and execute program instructions independently, creating the impression that the computer system has many processors, but they are cores, not processors. The calculation, data transmission, branching, and other types of instructions are all possible. At the same time, the processor can run instructions on several cores. It improves the system's overall program execution performance. Data duplication will also reduce due to the performance of the shared cache so communication can be more effective. As a result, the heat created by the CPU is reduced, and the total speed of execution is increased. In comparison to a multiprocessor system, multi-core CPUs have many advantages [34]. It has one CPU, so it has less expensive than a multiprocessor. It has less traffic due to the integration of cores into a single chip, which takes less time. It isn't necessary to configure it. It reduces the amount of heat generated by the CPU. It is speedier, especially when only running one program.

3.3. *Proposed algorithm*

This section represents the proposed PNRD-CloGen algorithm by using multi-core processor architecture. In a parallel mining approach, tasks are distributed as child nodes in the prefix tree, and each node acts as a separate task. The computation performs on all nodes in parallel. It's distributed between the available processor cores that able to be handled independently and generate nucleotide rules. The PNRD-CloGen algorithm composes of five steps. First, the genome sequence dataset is converted to a DBV structure that helps in compact sequences with their position in the prefix tree. Second, the first frequent nucleotides are discovered with a predefined minSup threshold and assigned to a prefix tree. Third, all nodes are pruned with the examination of closing and sequence extension in a parallel approach. It deals with each branch of the prefix tree as a separate task and implements a task-parallel method to compute each node concurrently. Each node at the first level is increased frequently by appending an item every time to form the next level. The extension of each node is achieved by the sequence extension method. It produces a new itemset by adding an item to the sequential patterns. The minimum support threshold is satisfied for each node to obtain all frequent sequences. The pruning procedure is implemented to eliminate uninteresting candidates early. Then the sequential patterns closeness is examined by discarding items that have similar support in other sequence patterns. Fourth, all nodes are checked, whether generator or closed patterns, to produce sequential rules. The generator patterns haven't subsequences equivalent to its support value and perform as a prefix rule, while the closed sequential patterns haven't supersequence equivalent to their support value and act as the postfix rule. This process is also performed in a parallel approach until generating all possible nucleotide rules. Fifth, meaningful nucleotide rules are generated, as shown in Figure 2.

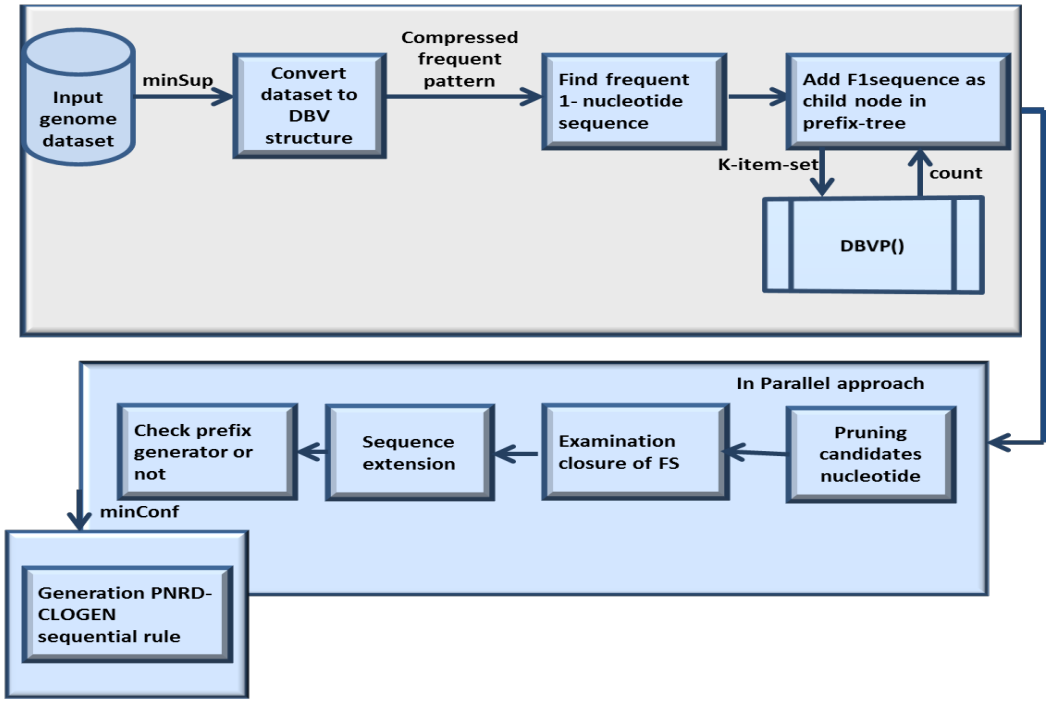

Fig. 2. Proposed PNRD-CLOGEN algorithm framework.

As shown in Algorithm 1, the database is first scanned to create the list of nucleotides. Next, all frequent first sequences (F1-S) that achieve the minSup threshold are discovered. In F1-S, genome sequences are stored as a child node in the prefix tree. Then, PNRD-CloGen is formed new tasks for each node of the root. Each task implements the method CloGen-extension. All tasks are executed in a parallel approach to generate frequent

CloGen sequential patterns. Finally, significant nucleotide rules that achieve the minConf threshold are generated.

Algorithm 1

Input: SDB, minSup, minConf

Output: nucleotide-sequential rules (SRs)

Steps:

```
1 Scan DB to create an ID list of the genome sequences.
2 Calculate the length of DB & set the minSup threshold
3 Find Frequent First Sequence >= minSup
4 Create DBV structure in a dictionary to get all nucleotide's positions
5   Add F1-S as a child node, where node=null
6     For each node do
7        Create a new task, then call PCloGen-extension.
8   Set minConf & generate nucleotide-SRs
```

The method that extends the F1-S is clarified in Algorithm 2, named CloGen pattern-Extension. The F1-S are set as a child node in the prefix tree. The CloGen pattern extension procedure extends patterns parallelly by rendering sequence extension to generate new sequential nucleotide patterns. It produces new itemsets by combining an item in the last position of the sequential patterns. The achievement of the minimum support threshold is checked for each node after implementing this method to obtain all frequent sequences.

The pruning technique is implemented in a parallel approach to exclude candidates that can't extend the frequent pattern. For example, consider the DB in Table4. There is no need to expand the prefix 'B' because an item 'A' has (start position = 1) that usually occurs before an item 'B' has (first locate= 2). The prefix 'A' in his extension already has 'B' with the same support. So, 'B' will be absorbed.

Algorithm 2

Input: root in prefix tree and minSup

Output: set of frequent CloGen sequences

Steps:

```
1   set a list node as child nodes
2 For each SP in list node do
3    If SP is pruned then
4       set SP as SP node
5       set Spa as SP child node
6          If (sup (Spa)≥minsup)then
7             Add Spa as a child node of SP
8         check and set the attribute of SP (closed or prefixed generator)
```

The parallel approach is implemented by using the multicore architecture. It can accomplish multiple tasks simultaneously depending on CPU cores. It utilizes an object to do various tasks and transfers them to a proper function that takes multiple arguments. So, the user can succeed many assumptions of an individual item, as seen in Algorithm 3.

Algorithm 3

Input: nodes, F1S, MinSup

Output: all child nodes ready to generate rules

Steps:
1 Set process equal to CPUs as pool
2 Set parameters of list node as Childs
3 Get child node by star-map () in pool

After discovering all frequent CloGen sequences, the algorithm produces all significant nucleotide rules, as shown in Algorithm 4. The algorithm generates nucleotide rules in a sub-tree by setting the probability each child node may be a prefix or closed. The prefix generator is pre and closed as post of the nucleotide rule. It performs recursively in parallel until the child nodes don't achieve the minConf value.

The framework is achieved through coding the algorithms in python and implementing the third and fourth steps in a parallel approach to achieve the highest efficiency. They included numerous child nodes that make many same tasks. The data-parallel is performed to save the required time for generating the nucleotide rules. Figure 3 shows the applied parallel method for the proposed algorithm.

Algorithm 4

Input: root, minConf

Output: Nucleotide-CloGen Rules

Steps:
1 Set pre as a sequence of the root
2 Set sub-node as a child node
3 For SP node in the root
4 Search prefix (SP node, rules, minConf)
5 If a node is a prefix
6 Search closed (value of node, node, rules, minConf)
7 For a child of children node
8 search prefix (child, rules, minConf)
9 If a sequence is closed
10 Conf= support of seq (support)/pre (support)
11 If Conf ≥ minConf
12 Set rules = pre⟶ post
13 Append the rule
14 Else stop generating rules
15 End if
16 End for
17 Call generate Nucleotide-CloGen Rule
18 End for

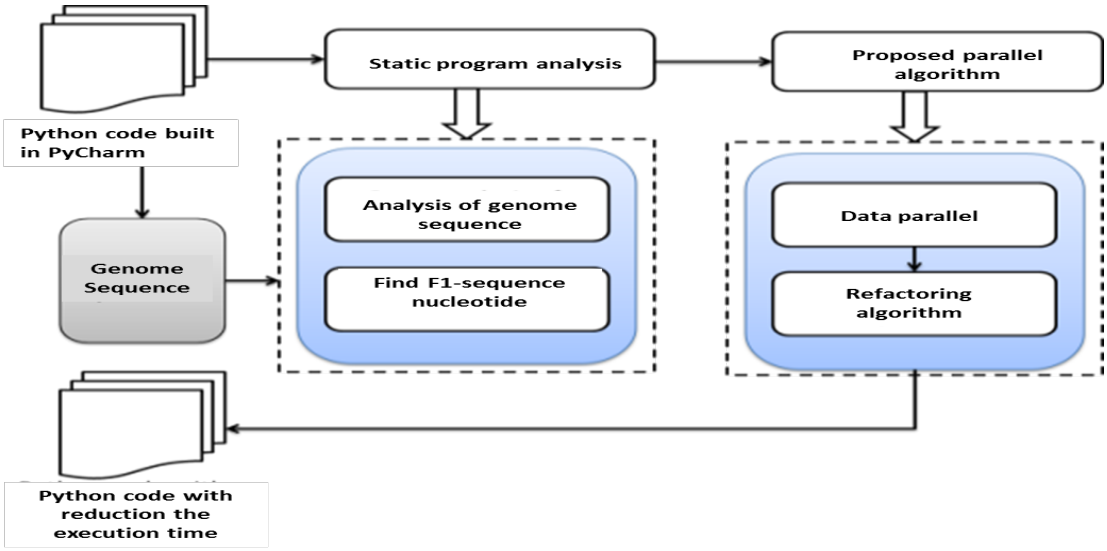

Fig. 3. Parallel method for proposed algorithm.

3.4. Technical Challenges

The implementation of a parallel approach on a whole algorithm caused recursion and consumed more time. The better utilization of the proposed algorithm is to apply only in two procedures; first, testing the child nodes that attained the minSup. Second, check each pattern, whether generator or closed pattern, to produce non-redundant sequential nucleotide rules. There's also no need to use the DBV then DBV structure that the (Non-redundant dynamic bit vector) NRD-DBV algorithm employs, that it caused additional time. Utilizing only the DBV structure is sufficient and more accurate for producing sequential rules with actual item positions. The utilization of the parallel approach helps enhance the performance without effect on other parameters such as power consumption. The combination of concise representation of frequent sequences provided a more compact sequence database than mining the whole set of sequential patterns with preserving all information. It will minimize the search area and help users to analyze the frequent sequences in large datasets efficiently.

4. **Experimental results**

4.1. Experiments on sample data

The proposed algorithm is coded in PyCharm and the validity of the code is assessed by using sample data. A sample of data contains thirty patients with a different order of their symptoms. Each patient is considered a single transaction, and there are only five symptoms (fever, cough, tiredness, difficulty breathing, and a rash on the skin). The phases are performed, and the result is calculated manually to test the correctness of the code. It proved its validity by producing the same symptoms rules. Figure 4 shows the detailed results of the mining process with minSup and minConf equal to 0.5. It produces only one symptom rule. The antecedent (x) = fever, and consequent (y) = difficulty breathing. It clarifies the number of patients, the number of symptoms that the patients suffered from, the count of the F1 sequence. It also provides the symptoms sequence tree after applying the CloGen method and the symptom rules that help determine the severity of cases. Finally, it shows the memory usage and the required time for each process to assess the proposed algorithm. The multicore approach is applied, and the results are compared with the original algorithm. The algorithm is implemented with different thresholds, and the results are measured. For example, when changing the minSup equal to 0.25 and the minConf to 0.40, the generated symptoms rules became six symptoms rules, as shown in table 7**.**

```
C:\Users\admin\AppData\Local\Programs\Python
\Python38-32\python.exe D:/seq_memory/seq-
rule/main.py
sample-1.1.txt
** frequent 1-sequences count: 4
** closed sequences:
 None
# [1], sup:21.0000
## [1, 4], sup:15.0000
# [3], sup:20.0000
# [4], sup:19.0000
# [2], sup:19.0000
** rules:
[1] -> [4], conf:0.71
# memory peak: 34085 Byte
```

Fig. 4. Sample execution of the proposed algorithm

Table 7. *Symptom rules generated by proposed algorithm in PyCharm program*

Rules			support	Confidence
Antecedent (x)	→	Consequent (y)		
Fever	→	tiredness	0.12	0.57
Fever	→	difficulty breathing	0.15	0.71
Fever	→	cough	0.10	0.48
Tiredness	→	difficulty breathing	0.10	0.50
Cough	→	tiredness	0.12	0.64
Cough	→	difficulty breathing	0.9	0.47

4.2. Experiments on COVID-19 genome sequence dataset (MT745584)

The proposed algorithm is appraised by comparing it with the NRD-DBV algorithm. These algorithms have been executed on a laptop with an Intel Core i5 2.33GHz and 6.58 GB free RAM running windows 7. The proposed algorithm has been applied on a COVID-19 genome sequences dataset. It is analyzed using the SPMF data mining library [35]. SPMF is a pattern mining framework that is open-source and cross-platform. It has about 180 data mining methods implemented in it. This dataset is composed of sequences of nucleotides of the MT745584 progeny of the coronavirus. It was captured on 2020-07-13 in Bahrain. M. Saqib Nawaz acquired the dataset from a public database and modified it to the SPMF format. It consists of 1493 sequences with four items and an average length equal to 16. The applied experiment compares the execution time of the two algorithms. Various minSup values are assigned while fixing a minConf value and vice versa. Many initial experiments were performed to determine the value of parameters to acquire the highest performance.

The CloGen SPM algorithm is applied to discover significant relationships between nucleotides. It requires setting a minSup threshold to find frequent sequential patterns. Count of generated pattern variants with different values of minSup. Only ten sequence patterns appear for a minSup equal to 0.5 and 37 sequence patterns for a minSup equal to 0.1, as shown in Tables 8 & 9. The performance of the mining patterns process was pretty fast. The decreasing of the minSup caused rising into frequent patterns generated, the execution time, and the memory usage. There aren't any nucleotides rules through the mining process when fixing the minConf value and setting the minSup with a value greater than 0.6. The values are determined after trying multiple values of minSup to obtain better performance. In general, the runtime increases with low minSup values that generate an increased number of generated nucleotides rules.

Table 8. *Frequent nucleotides extracted by CloGen*

Pattern	Support	Pattern	Support	Pattern	Support	minSup
A	1491	[G,A,C]	286	[T,G,C]	387	11%
[A,G]	871	[G,T]	540	[T,C]	901	11%
[A,G,T]	219	[G,T,A]	189	[T,C,A]	245	11%
[A,G,C]	303	[G,T,C]	239	[T,C,G]	259	11%
[A,T]	670	[G,C]	807	[C]	1466	11%
[A,T,G]	312	[G,C,A]	217	[C,A]	573	11%
[A,T,C]	263	[G,C,T]	180	[C,A,G]	261	11%
[A,C]	891	[T]	1489	[C,A,T]	162	11%
[A,C,G]	291	[T,A]	817	[C,G]	636	11%
[A,C,T]	236	[T,A,G]	338	[C,T]	561	11%
[G]	1470	[T,A,C]	392	[C,T,A]	162	11%
[G,A]	597	[T,G]	926	[C,T,G]	265	11%
		[T,G,A]	275			11%

Table 9. *Frequent nucleotides extracted by CloGen*

Pattern	Support	MinSup
[A]	1491	50%
[A, G]	871	50%
[A, C]	891	50%
[G]	1470	50%
[G, C]	807	50%
[T]	1489	50%
[T, A]	817	50%
[T, G]	926	50%
[T, C]	901	50%
[C]	1466	50%

It's clear from the experiment that PNRD-CloGen accomplished much less time than the NRD-DVB algorithm. It has confirmed its efficiency when setting the minSup with low values, as the time to discover the nucleotides rules is roughly half the time that the NRD-DVB achieves, as shown in Figure 5.

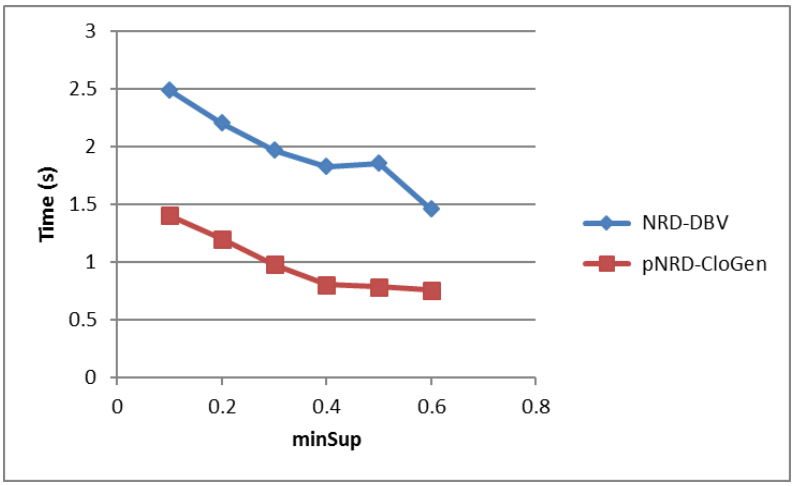

Fig. 5. The runtime of sequence genome dataset with different minSup values (MinConf=0.5)

The execution time is shown in Figure 6. It examined with various minConf values (from 0.1 to 0.6), while minSup is 0.5 for all cases. The runtime increased with decreasing minConf values due to the increasing number of generated nucleotide rules. The experiments show that PNRD-DVB performs faster than NRD-DVB.

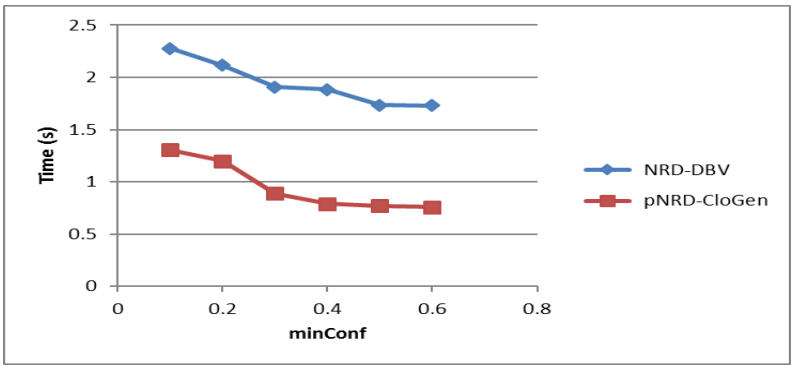

Fig. 6. The runtime of sequence genome dataset with different minConf values (MinSup=0.5)

For memory usage, various values of minSup and minConf are implemented on the sequences genome dataset. The NRD-DBV is achieved less memory than PNRD-CloGen with different values of minSup and fixed the minConf, as shown in Figure 7. However, both algorithms utilized the same approach for mining sequence patterns; the PNRD-CloGen assigns multiple tasks into individual tasks and processes each task independently. So require more memory to save the results. It turns out that the increasing value of minSup produces a fewer number of generated nucleotide rules. So the memory usage and the runtime decreased.

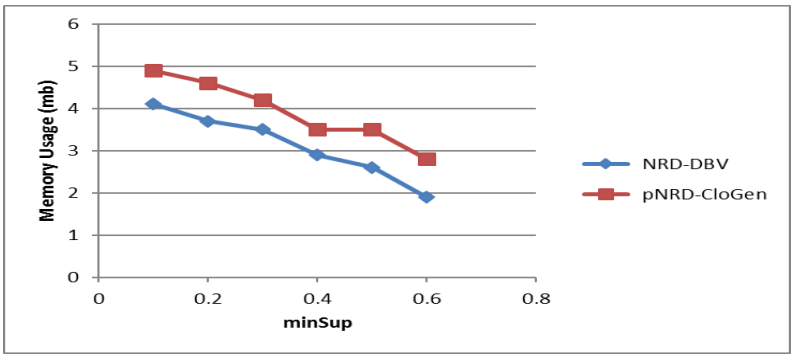

Fig. 7. The memory usage of sequence genome dataset with different minSup values (MinConf=0.5)

While utilizing different values of the minConf with fixing the minSup value, the memory consumption remains constant for both algorithms, and the NRD-DVB algorithm still consumes less memory, as shown in Figure 8. However, the proposed PNRD-CloGen algorithm improves the CPU idle time in that it makes a parallel approach only on two procedures; first, find F1-Sequence and add them as child nodes in a prefix tree. Second, check all nodes prefix or closed for generating the sequential nucleotide rules.

4.3. Computational cost analysis

The computing cost is calculated to assess the proposed algorithm's performance. The complexity of creating non-redundant nucleotide rules is O(n*c), where n is the number of nodes and c is the average number of child nodes. The (n-1) operations are utilized to verify and generate sequential rules like k<<n for each sequence. As a result, the complexity of PNRD-CloGen is O(n). It means that the algorithm's maximum running duration is proportional to the size of the input. The suggested technique confirms its efficiency because it is the second-best state of big O notation after the constant state.

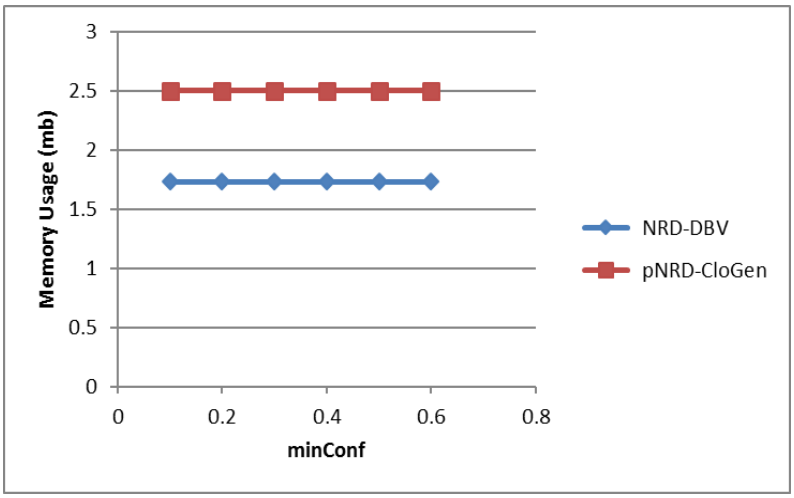

Fig. 8. The memory usage of sequence genome dataset with different minConf values (MinSup=0.5)

4.4 Scalability measurement

Scalability refers to a system's ability to increase or decrease its performance and cost in response to variations in processing demands. The system's scalability is evaluated by appraising how its performance changes as the input size increases. On the **MT745584** dataset, the scalability of the two contrasted algorithms are tested for various data sizes and a fixed value of minSup equal to 0.04. The results demonstrate the impact of developing rules on runtime. As indicated in the results, the PNRD-CloGen method outperformed the NRD-DBV algorithm in terms of scalability, as shown in Figure 9.

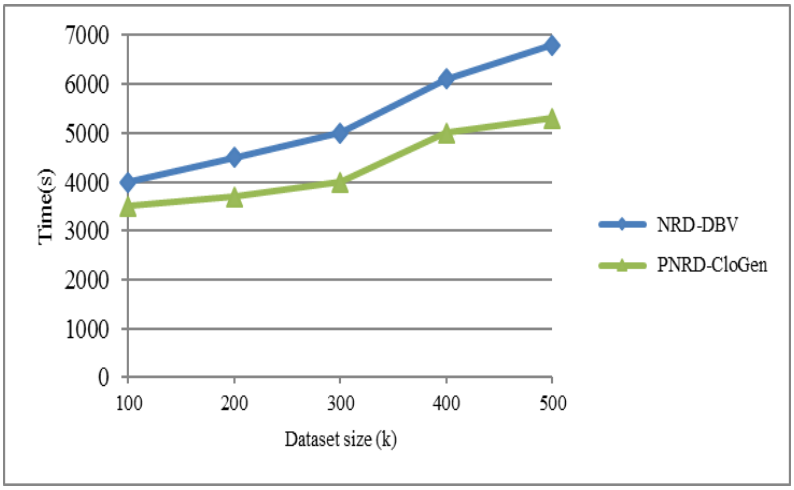

Fig. 9. Scalability of compared algorithms for various dataset sizes with minSup=0.04 on (MT745584) dataset

5. Conclusion

The process of detecting nucleotide rules suffers from a slow completion process and requires costly sophisticated techniques. As the sequence of the COVID-19 genome is unrevealed early, the behavior of COVID-19 becomes unclear in time. This paper proposed a sequential rule mining algorithm called PNR-CloGen to detect mutations in COVID-19 genome sequences and generate nucleotide rules. The PNRD-CloGen algorithm is used to find frequent nucleotide rules in the sequence patterns. It utilizes the DBV structure and prefix tree in a parallel approach. The prefix tree helps to quickly generate the nucleotide rules and early stop

generating un-candidate's rules. Two methods of the algorithm have been implemented in a parallel approach. The first method examines all F1-S (child nodes in the prefix tree) that achieve the minSup value to extend nodes. The second procedure prunes all child nodes and checks whether they are prefixes generators or closed to generate sequential nucleotide rules. It performs many same tasks of child nodes simultaneously and obtains the overall result in less time. The experiment results denoted that; the proposed algorithm outperformed the NRD-DBV algorithm in terms of execution runtime and scalability, specifically when increasing the number of cores and decreasing the minSup value. It helps to detect the behavior of COVID-19 and the extent of the coronavirus mutating efficiently, so it supports better clinical management.

6. Recommendation and future work

The best utilization of the parallel method is started from testing the child nodes of the prefix tree. Then check each pattern generator or closed pattern for producing the nucleotide rules. Using only the DBV structure without starting with DBV is sufficient and more accurate to acquire nucleotide rules with actual valid positions of items. The use of processes instead of threads provides better memory utilization when memory goes down. The multi-core processing approach prevents data corruption and deadlocks that may appear with a multithreading approach.

For future work, the maximal patterns are utilized instead of closed generator patterns and study their impact on produced sequential rules in terms of the runtime and the accuracy of the generated rules. Additionally, increasing the number of cores, especially for large databases, and voiding the increase of power consumption. It can be accomplished by minimizing the clock frequency that leads to avoiding overheating.

References

[1] Smith, David R. "A short guide to genetic data mining." EMBO reports 22.1 (2021): e51845.
[2] Feng, Wei, et al. "Molecular diagnosis of COVID-19: challenges and research needs." Analytical chemistry 92.15 (2020).10196-10209.
[3] Stencel, Agnieszka, and Bernard Crespi. "What is a genome?.", *Institute of Environmental Sciences, Jagiellonian University, Gronostajowa 7, 30-387 Krakow, Poland; Department of Biological Sciences, Simon Fraser University, Burnaby, British Columbia, V5A 1S6 Canada (2013).3437-3443.
[4] Nawaz, M. Saqib, et al. "Using artificial intelligence techniques for COVID-19 genome analysis." Applied Intelligence 51.5 (2021). 3086-3103.
[5] Fournier-Viger, Philippe, et al. "A survey of sequential pattern mining." Data Science and Pattern Recognition 1.1 (2017). 54-77.
[6] Gao, Chuancong, et al. "Efficient mining of frequent sequence generators." Proceedings of the 17th international conference on World Wide Web. (2008). (pp. 1051-1052).
[7] Gan, Wensheng, et al. "A survey of parallel sequential pattern mining." ACM Transactions on Knowledge Discovery from Data (TKDD) 13.3 (2019). 1-34.
[8] Pham, Thi-Thiet, Jiawei Luo, and Bay Vo. "An effective algorithm for mining closed sequential patterns and their minimal generators based on prefix trees." International Journal of Intelligent Information and Database Systems 7.4 (2013). 324-339.
[9] Fournier-Viger, Philippe, et al. "VMSP: Efficient vertical mining of maximal sequential patterns." Canadian conference on artificial intelligence. Springer, Cham, (2014). p. 83-94.
[10] Martinez, Ricardo, Claude Pasquier, and Nicolas Pasquier. "GenMiner: mining informative association rules from genomic data." IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2007). (pp. 15-22).
[11] Attia, Khaled M., Mostafa A. El-Hosseini, and Hesham A. Ali. "Dynamic power management techniques in multi-core architectures: A survey study." Ain Shams Engineering Journal 8.3 (2017). 445-456.
[12] Mooney, Carl H., and John F. Roddick. "Sequential pattern mining--approaches and algorithms." ACM Computing Surveys (CSUR) 45.2 (2013). 1-39.
[13] Fournier-Viger, Philippe, et al. "CMRules: Mining sequential rules common to several sequences." Knowledge-Based Systems 25.1 (2012). 63-76.
[14] Han, Jiawei, et al. "Frequent pattern mining: current status and future directions." Data mining and knowledge discovery 15.1 (2007). 55-86.
[15] Dai, Chris Feudtner. Association Rule Mining of Polypharmacy Drug Utilization Patterns in Health Care Administrative Data Using SAS. Enterprise Miner, (2018). pp. 2470.
[16] Sarıyer, Görkem, and Ceren Öcal Taşar. "Highlighting the rules between diagnosis types and laboratory diagnostic tests for patients of an emergency department: Use of association rule mining." Health informatics journal 26.2 (2020). 1177-1193.
[17] Estiri, Hossein, et al. "Transitive sequencing medical records for mining predictive and interpretable temporal representations." Patterns 1.4 (2020). 100051.

[18] Kilgore, Phillip CSR, et al. "GatewayNet: a form of sequential rule mining." BMC medical informatics and decision making 19.1 (2019). 1-13.

[19] Jamshidi, Mohammad, et al. "Artificial intelligence and COVID-19: deep learning approaches for diagnosis and treatment." Ieee Access 8 (2020). 109581-109595.

[20] Naseem, Maleeha, et al. "Exploring the potential of artificial intelligence and machine learning to combat COVID-19 and existing opportunities for LMIC: a Scoping review." Journal of Primary Care & Community Health 11 (2020). 2150132720963634.

[21] Alafif, Tarik, et al. "Machine and deep learning towards COVID-19 diagnosis and treatment: survey, challenges, and future directions." International Journal of Environmental Research and Public Health 18.3 (2021). 1117.

[22] Adly, Aya Sedky, Afnan Sedky Adly, and Mahmoud Sedky Adly. "Approaches based on artificial intelligence and the internet of intelligent things to prevent the spread of COVID-19: scoping review." Journal of medical Internet research 22.8 (2020). e19104.

[23] Naudé, Wim. "Artificial Intelligence against COVID-19." Institute of Labor Economics (IZA) Discussion Papers, No. 13110, (2020).

[24] Louis, Christine, and Isabelle Nepomuceno. "WestJEM Volume 21, Issue 5." Western Journal of Emergency Medicine: Integrating Emergency Care with Population Health 21.5 (2020).

[25] Wang, Yazi, et al. "Intrusion detection and performance simulation based on improved sequential pattern mining algorithm." Cluster Computing 23 (2020). 1927-1936.

[26] Pathan, Refat Khan, Munmun Biswas, and Mayeen Uddin Khandaker. "Time series prediction of COVID-19 by mutation rate analysis using recurrent neural network-based LSTM model." Chaos, Solitons & Fractals 138 (2020). 110018.

[27] World Health Organization. "Genomic sequencing of SARS-CoV-2: a guide to implementation for maximum impact on public health", 8 January (2021).

[28] Wang, Fang, et al. "Initial whole-genome sequencing and analysis of the host genetic contribution to COVID-19 severity and susceptibility." Cell discovery 6.1 (2020).1-16.

[29] Tandan, Meera, et al. "Discovering symptom patterns of COVID-19 patients using association rule mining." Computers in biology and medicine 131 (2021). 104249.

[30] Yang, Hsin-Chou, et al. "Analysis of genomic distributions of SARS-CoV-2 reveals a dominant strain type with strong allelic associations." Proceedings of the National Academy of Sciences 117.48 (2020). 30679-30686.

[31] Marquez, Sully, et al. "Genome sequencing of the first SARS-CoV-2 reported from patients with COVID-19 in Ecuador." medRxiv (2020).

[32] Upadhyay, Pragati, M. K. Pandey, and Narendra Kohli. "A comprehensive survey of pattern mining: challenges and opportunities." International Journal of Computer Applications 975 (2018). 8887.

[33] Czarnul, Paweł, Jerzy Proficz, and Krzysztof Drypczewski. "Survey of methodologies, approaches, and challenges in parallel programming using high-performance computing systems." Scientific Programming (2020).

[34] Venu, Balaji. "Multi-core processors-an overview." Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, UK (2011) arXiv:1110.3535 (2011).

[35] Fournier-Viger, Philippe, et al. "The SPMF open-source data mining library version 2." Joint European conference on machine learning and knowledge discovery in databases. Springer, Cham, (2016).