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.

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.

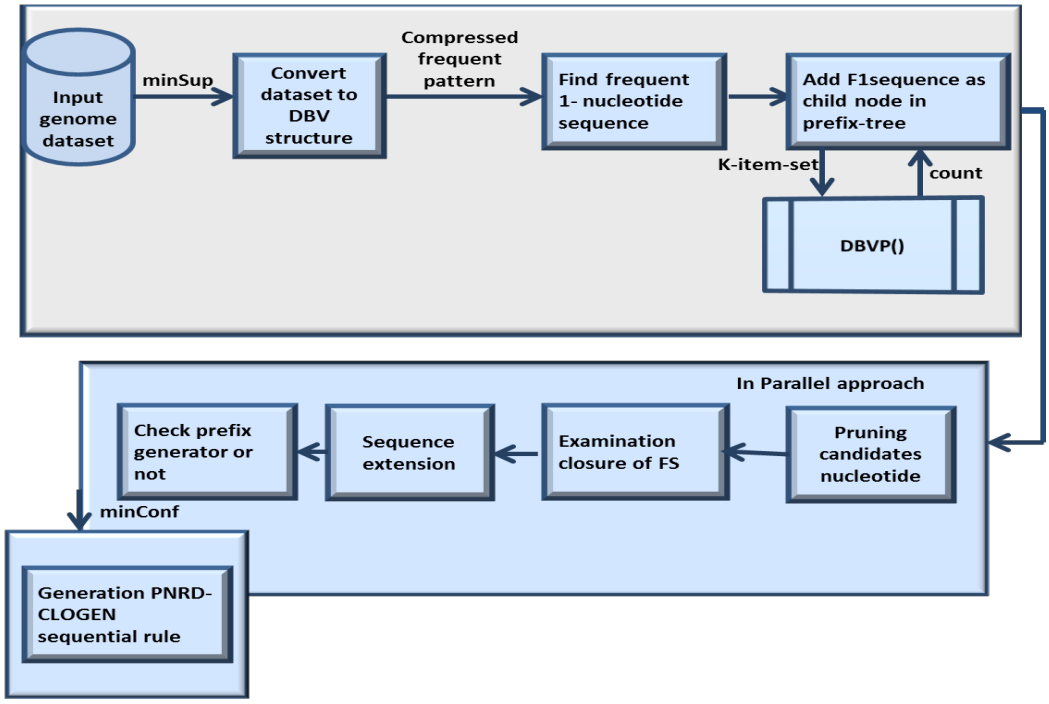

Fig. 2. Proposed PNRD-CLOGEN algorithm framework.

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

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

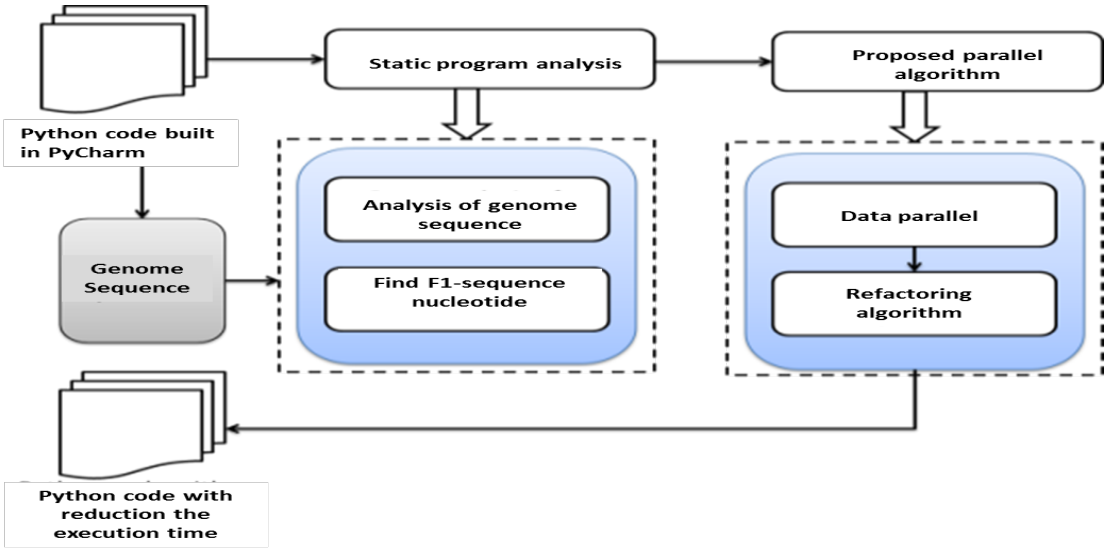

Fig. 3. Parallel method for proposed algorithm.

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.

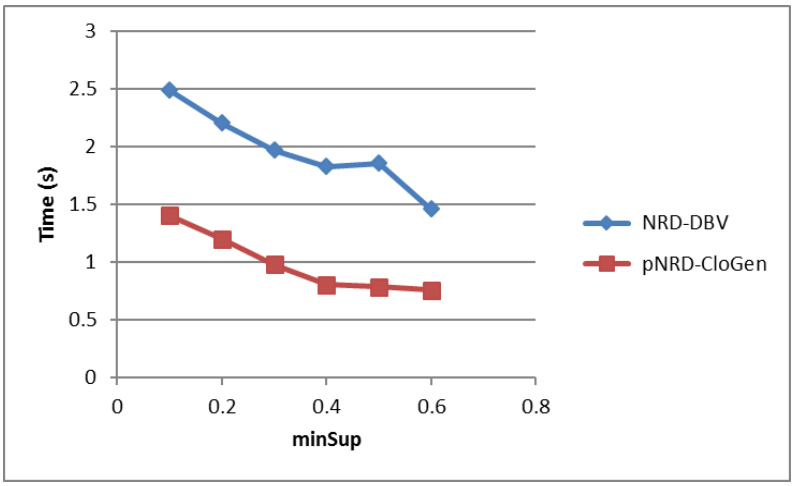

Fig. 5. The runtime of sequence genome dataset with different minSup values (MinConf=0.5)

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.

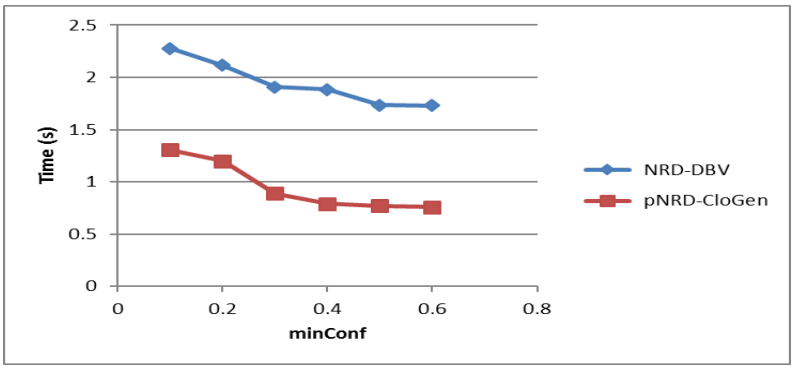

Fig. 6. The runtime of sequence genome dataset with different minConf values (MinSup=0.5)

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.

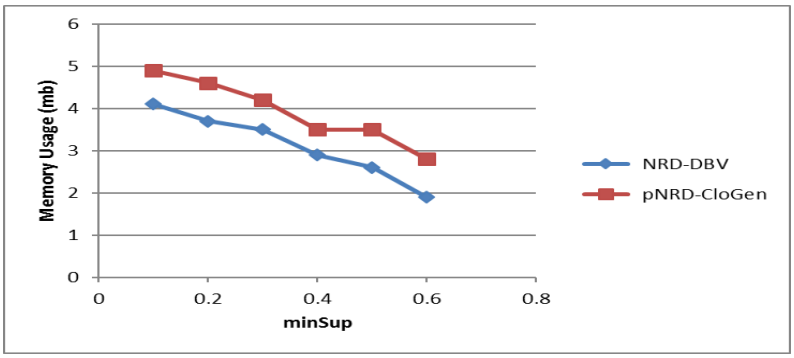

Fig. 7. The memory usage of sequence genome dataset with different minSup values (MinConf=0.5)

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.

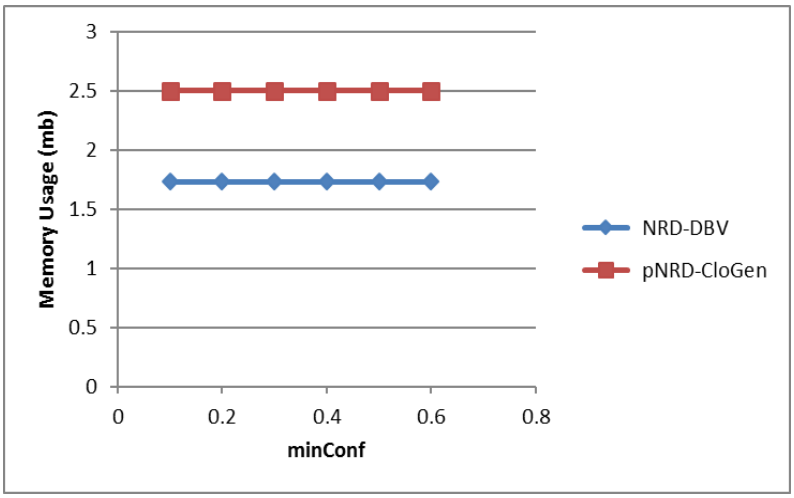

Fig. 8. The memory usage of sequence genome dataset with different minConf values (MinSup=0.5)

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.

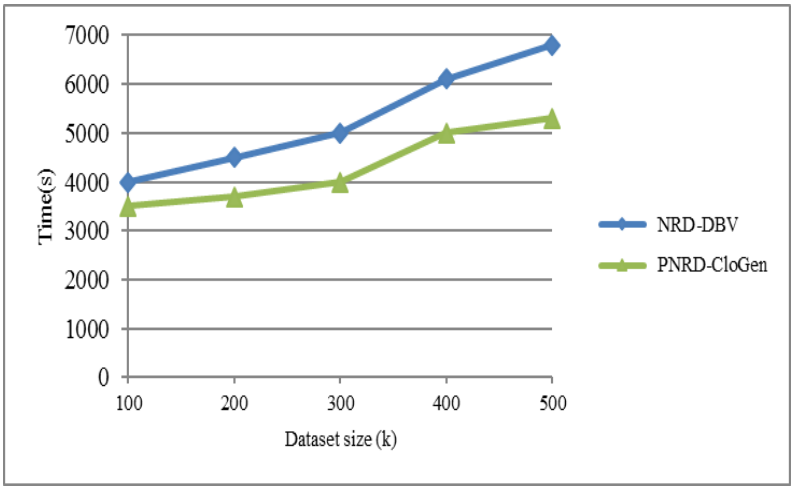

Fig. 9. Scalability of compared algorithms for various dataset sizes with minSup=0.04 on (MT745584) dataset

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).