

# Efficient Evolutionary Sequential Pattern Mining Algorithms for Mining DNA Sequences

AHMED SHARAF ELDIN<sup>1</sup> - TAYSIR HASSAN A. SOLIMAN<sup>2</sup> - MARWA MOHAMED M. GHAREEB<sup>3</sup>

<sup>1</sup>Information Systems Department Faculty of Computers and Information, Helwan University, Cairo, Egypt

<sup>2</sup>Information Systems Department Faculty of Computers and Information, Ain Shams University, Cairo, Egypt

<sup>3</sup>Information Systems Department, Modern Academy, Cairo, Egypt

## Abstract

*In the current work, Mutated Rules for Sequential Mining (MRS), an efficient evolutionary sequential pattern mining algorithm, is proposed for mining DNA sequences. MRS applies mutation genetic operator to generate new rules in the population. Pruning redundant rules, MRS efficiently get the new frequent patterns which will be mutated to iteratively generate new rules in the next population.*

*Moreover, MRS does not apply database projection for mining, with high CPU, and memory utilization. However, in order to handle variants length of sequences without alignment, MRS is modified (MMRS) to handle sequences of variants lengths. MMRS outperforms MRS and Apriori algorithms when applied to DNA sequences from GENBANK when testing three main issues: number of bases per sequence, support threshold, and number of sequences in each trial.*

## 1. Introduction

With the completeness of several genome projects, the number of DNA sequences has tremendously enlarged, which requires more complex techniques to be developed.

The biggest excitement currently lies with the availability of complete genome sequences for different organisms.

The GenBank –of National Center for Biotechnology Information (NCBI)–, EMBL –European Molecular Biology Laboratory– and DDBJ databases contain DNA sequences for individual genes that encode protein and RNA products [1]. Current DNA databases provide an excellent opportunity to analyze such data and perform data mining to discover the hidden mysteries within biological data.

Data mining techniques have lead to the discove-

ry of hidden structured patterns in large databases. These patterns vary according to the nature of the data. Several data mining algorithms have been explored in order to be applied in commercial applications and biological applications as well, such as sequential mining. For example, sequential patterns provide very crucial patterns to be mined and have got a lot of attention in the data mining research community. Sequential pattern mining has been explored since its introduction by Agrawal and Srikant [2]. So there is a great diversity of algorithms for sequential pattern mining.

Most of these algorithms are based on the Apriori property proposed in association rule mining. Based on this heuristic, a series of Apriori-like algorithms have been proposed: AprioriAll, AprioriSome, DynamicSome [2], GSP [3]. Another series of data projection based algorithms became popular because of their efficiency, which include FreeSpan [4] and PrefixSpan [5]. Recently, Zaki proposed the SPADE [6], which is a lattice based algorithm based on Apriori property. After that, a fast algorithm, called SPAM [7] has been proposed with a vertical bitmap representation of the data. Also, a memory indexing based approach called MEMISP was proposed [8] to use a memory indexing scheme for reducing the I/O complexity.

Sequential pattern mining has been applied to a wide variety of applications, such as telecommunications, the World Wide Web (WWW) for analysis sequential web logs, and DNA sequences. Sequential pattern mining algorithms applied on biosequences database faces many problems related to the features of biosequences such as:

- **Small alphabet:** A biosequence has a very small alphabet, i.e., 4 for DNA sequences and 20 for

**Correspondence:** Ahmed Sharaf Eldin, Information Systems Department Faculty of Computers and Information, Helwan University, Cairo, Egypt, e-mail: drsharaf@starnet.com.eg

Taysir Hassan A. Soliman, Information Systems Department Faculty of Computers and Information, Ain Shams University, Cairo, Egypt, e-mail: taysir\_soliman@ijicis.net

Marwa Mohamed M. Ghareeb, Information Systems Department, Modern Academy, Cairo, Egypt, e-mail: mortamm@hotmail.com

protein sequences, and many short patterns occur in most sequences [9].

- **Large sequence length:** A biosequence has a very large length. In contrast, a transaction sequence has a typical length from 5 to 15. A long sequence (especially, with a small alphabet) often contains long patterns [9].

- **Gapped patterns over long regions:** A biosequence pattern has the form of  $X_1 * \dots * X_k$  spanning over a long region, where each  $X_i$  is a short region of consecutive items, called a segment, and  $*$  denotes a variable length gap corresponding to a region not conserved in the evolution. The presence of  $*$  implies that pattern matching is more permissible and involves the whole range in a sequence [9].

These features create a different type of explosion of patterns and need for special algorithms to deal with. Evolutionary algorithms are a kind of search methods, which introduce both exploitation of the most promising solutions and exploration of the search space. Evolutionary algorithms operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations are created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation [10]. Evolutionary algorithms span a family of optimization algorithms, differing in details such as the way of selecting the best, how to create new solutions from existing ones, the operators they use to create the next generation, and the data structures used to represent those solutions. However, they correspond to the same basic algorithmic skeleton. These algorithms are genetic algorithms (GAs), Genetic programming (GP), Evolution strategies (ES) and Evolutionary programming (EP) [11].

In the current work, Mutated Rules for Sequential Mining (MRS) algorithm is proposed, which combines both the main concepts of evolutionary algorithms and rule learning for solving the problem of sequential pattern mining. Another algorithm is also proposed, which is Modified MRS (MMRS) in order to handle variant lengths of sequences without alignment. This paper is organized as follows: section 2 illustrates preliminaries to introduce

some basic notations. Section 3 clarifies the proposed algorithm (MRS); section 4 clarifies performance study.

Finally, section 5 concludes the paper, indicating

future work.

## 2. Sequential pattern mining

Sequential pattern mining is trying to find the relationships between occurrences of sequential events, to find if there exist any specific order of the occurrences [12]. A sequence database  $S$  is a set of tuples  $\langle \text{sid}, s \rangle$ , where  $\text{sid}$  is a sequence\_id and  $s$  is a sequence. A sequence  $s = \langle s_1, \dots, s_n \rangle$  is an ordered list of itemsets (elements) where  $s_i$  is the  $i$ th element of  $s$ . The number of elements in a sequence is called the length of the sequence. A sequence

$s$  with length  $K$  is called  $K$ -sequence and is denoted by  $|s|$ .

The support of a sequence  $t$  in a sequence database  $S$ , where the support threshold is called  $\text{min\_sup}$ , is the number of tuples in the database containing  $t$ . Given a positive integer  $\text{min\_sup}$ , a sequence  $s$  is called a (frequent) sequential pattern in sequence database  $S$  if the sequence is contained by at least  $\text{min\_sup}$  tuples in the database, i.e.  $\text{support}(s) \geq \text{min\_sup}$ . Applying sequential mining algorithms to DNA sequences is a very challenging

problem. MRS and MMRS mine DNA sequences using evolutionary sequential mining algorithm. Both of them do not depend on database projection.

## 3. Proposed algorithm: mutated rules for sequential pattern mining (MRS)

Mutated Rules for Sequential Mining (MRS) algorithm is proposed for mining sequential patterns based on the idea of evolutionary algorithm and rule learning. MRS is an iterative algorithm, based on the Apriori property of nonmonotonicity. MRS consists of six main steps: grammatical inference, rule construction, mutation application, evaluation function, frequent patterns generation and rule pruning, and new candidate generation.

### 3.1. Grammatical inference

The aim of grammatical inference is to infer – from set of examples – the set of rules. Therefore, it can be called as Rule learning. Rule learning can be achieved by using a suitable grammar to compose rules. The grammar should specify the structure of a rule, which is of the form “If antecedents then consequent “. The antecedent part is a conjunction of attribute descriptor, “conditions “.

An attribute descriptor characterizes an attribute, which can be described in many ways; thus, there are many different formats of descriptors. There are two broad data types: categorical (nominal) and continuous (real valued) attributes. A descriptor can assign a value to a nominal attribute, a range of values to a continuous attribute, or can be used to compare attribute values. Each of the

conditions in the rule antecedent can be in one of several different forms. The two most common ones are the following:

- (a)  $\text{Attr}_i \text{ Op Val}_k$       (b)  $\text{Attr}_i \text{ Op Attr}_j$

Where  $\text{Attr}_i$  and  $\text{Attr}_j$  denote the  $i^{\text{th}}$  and  $j^{\text{th}}$  attributes in the set of predictor attributes,  $\text{Val}_k$  denotes the  $k^{\text{th}}$  value of the domain of  $\text{Attr}_i$ , and **Op** is a comparison operator usually in  $[ = , \neq , < , > , ? , \leq ]$ . The former kind of condition is referred to it as a propositional condition which can be as (Age > 21) or (sex = male). The latter kind is referred to it as a first-order condition which can be as (Income > Expenditure).

Since DNA sequences Database does not have attributes to be used in conditions, **MRS** will create imitative attributes as two in antecedes part and one in consequent part, so rule will be as the following:

**IF** Anterior condition (**AC**) and succedent condition (**SC**)  
**Then** Candidate pattern (**CP**)

Where:

- AC**: is the first attribute and its value will be one of the list of last frequent pattern. ( $L_{k-1}\text{Set}$ ).
- SC**: is the second attribute & its value will be one of 4 bases ( "a", "c", "g", or "t" )
- CP**: is the resultant pattern which is one of new candidate patterns.

The first attribute "**AC**" will be denoted as  $\text{attr1}$ , the second one "**SC**" will be denoted as  $\text{attr2}$ , and finally the third attribute "**CP**" will be denoted as  $\text{attr3}$ . The symbols  $\text{erc1}$ ,  $\text{erc2}$ , and  $\text{erc3}$  in this grammar are ephemeral random constants (**ERCs**). Each ERC has its own range for instantiation. Table 1 shows the grammar for rule learning in MRS.

**Table 1.** The Grammar for Rule Learning in MRS algorithm.

1: start	→ [if, antes, [then], consq.
2: antes	→ $\text{attr1}$ , [and] $\text{attr2}$ .
3: $\text{attr1}$	→ $\text{attr1\_descriptor}$ .
4: $\text{attr2}$	→ $\text{attr2\_descriptor}$ .
5: $\text{attr1\_descriptor}$	→ [ $\text{attr1} =$ ], $\text{erc1}$ .
6: $\text{attr2\_descriptor}$	→ [ $\text{attr2} =$ ], $\text{erc2}$ .
7: consq	→ $\text{attr3\_descriptor}$ .
8: $\text{attr3\_descriptor}$	→ [ $\text{attr3} =$ ], $\text{erc3}$ .
9: $\text{erc1}$	→ [member (list of last frequent patterns)].
10: $\text{erc2}$	→ [member (A, C, G, T)].
11: $\text{erc3}$	→ $\text{erc1}$ & $\text{erc2}$ .

### 3.2. Rule construction

From the proposed grammar for rule learning in MRS, we can initially get rules like the following rules which will serve as the initial first population. Each rule is an individual in the population.

- If  $AC = a$  and  $SC = c$ , then  $CP = ac$
- If  $AC = c$  and  $SC = a$ , then  $CP = ca$
- If  $AC = g$  and  $SC = c$ , then  $CP = gc$
- If  $AC = t$  and  $SC = g$ , then  $CP = tg$

Notice that  $\text{attr1}$  will take its value in the first population from these four bases "a, c, g, & t" which will serve as the list of frequent patterns. In the further coming iterations, the obtained frequent patterns from the last iteration will be used as a list of values for  $\text{attr1}$ .

Example, suppose that "gc" is obtained as a frequent pattern from the first iteration, so we can get rules, as shown in Table 2:

**Table 2.** Rule set obtained for the frequent pattern "gc".

If	$AC = gc$	and	$SC = a$ ,	then	$CP = gca$
If	$AC = gc$	and	$SC = t$ ,	then	$CP = gct$
If	$AC = gc$	and	$SC = c$ ,	then	$CP = gcc$
If	$AC = gc$	and	$SC = g$ ,	then	$CP = gcg$

$\text{Attr2}$  will take a value from these 4 bases (a, c, g, and t) as in the grammar.  $\text{Attr3}$  will take its value as a union or concatenation of  $\text{attr1}$  and  $\text{attr2}$ . Example, if  $\text{attr1} = ta$  and  $\text{attr2} = g$ , hence  $\text{attr3}$  will equal to "tag". The obtained value of  $\text{attr3}$  will serve as a candidate pattern that will be evaluated to deliver frequent patterns.

### 3.3. Mutation application

In **MRS**, the search space is explored by generating new rules using mutation genetic operator. The new value is generated by the some derivation mechanism as in the population creation. Because the offspring have to be valid according to the grammar, a selected attribute can only be mutated to another value with a compatible structure. For example, if the parent is:  
 If  $AC = gt$  and  $SC = c$ , then  $CP = gtc$   
 The underlined attribute is selected for mutation, so that it may be mutated to:  
 If  $AC = gt$  and  $SC = t$ , then  $CP = gtt$   
 Note that  $\text{attr2} -SC-$  can take value from (a, c, g or t).  
 Therefore, we can get three new offspring by mutating  $\text{attr2}$ , which are:

If  $AC = gt$  and  $SC = t$ , then  $CP = gtt$   
 If  $AC = gt$  and  $SC = g$ , then  $CP = gtg$   
 If  $AC = gt$  and  $SC = a$ , then  $CP = gta$

Note that  $attr3 -CP-$  is automatic mutated because its value depends on the values of  $attr1$  &  $attr2$ . Similarly, we can mutate  $attr1$  with all available values of the previous selected frequent pattern list.

### 3.4. Evaluation function

An evaluation (fitness) function is needed to evaluate rules, where the quality of the individuals (candidate solutions) are measured as precisely as possible. The evaluation function contains two pruning techniques: the first is the Candidate Pruning (Rule Removal) and the second is Support Pruning. In Candidate Pruning, MRS will remove rules that have some  $(K-1)$  - subsequences do not exist in  $L_{k-1}$ . Therefore, it supports the antimonotonicity property, which implies that if a sequence does not support a pattern, then it could not support any of its Superpatterns. Therefore, every time search space will be decreased.

In MRS, the Support Pruning is based on the supportconfidence framework [2]. Fitness function will measure support as the coverage of a rule which is the ratio of the number of records covered by the rule to the total number of records. In the Support Pruning, each rule is checked with every record in the database and count the number of records matching the consequent (the 'then' part) as rulesupport "rule-sup". Finally, rule-support is compared with a user-defined minimum threshold (min-sup) and weeds out rules with support less than min-sup.

The fitness function can be defined as:

1- Rule Removal.

2- If rule-sup > min-sup:

I. Then ? add this rule to frequent pattern list.

II. Otherwise ? weeds out this poor rule.

The evaluation process is a scan operation for each rule which can cause a delay and an overhead computation.

Number of scans can be decreased by evaluating all generated rules of the same iteration in one scan which can be called Parallel Rules Scanning (PRS).

MRS differs from conventional GA in that the parents compete with the offspring for places in the new generation where in conventional GA the next generation of population only consists of the offspring's. Therefore, the evaluation process will be applied on parents and offsprings allowing the parents to survive to another generation.

### 3.5. Frequent patterns generation and rule pruning

After evaluating each individual, MRS obtains the best individuals that passed the evaluation as the frequent patterns. Frequent patterns are consequent parts of succeeded Rules which will enter the next generation as the list of values of Anterior condition of new rules which follow the evolution according to Darwin's theory of evolution which states that the best ones survive to create new offspring's.

In conventional GA, rules to be removed are randomly selected among the rules in the population. Rule removal could be improved by choosing the rule to be removed based on some rule-quality criterion rather than at random.

MRS removes rules that fail in the evolution process and so it prevents worst-suited rules to survive to the next generation and so it apply the rule removal operator of GA and follows pruning techniques of sequential patterns mining.

### 3.6. New candidate generation

MRS will loop to the Rule generation step and use the frequent patterns list obtained from the previous step to construct new rules by following steps of Rule learning from MRS's grammar.

MRS will stop generating new rules when stopping criterion is reached which is here the length of sequence.

This means that MRS will generate number of population equals to the length of sequence in DNA sequences database.

**Stopping Criterion is:**

Number of iteration = Number of items in sequence "length".

### 3.7. MMRS

The proposed model, MRS, is customized to fit with sequences file that is not aligned where the number of bases per sequence is variant from sequence to another.

This derives a new model called Modified MRS, MMRS.

MMRS contains the same six stages as MRS, which are grammatical inference, rules constructing, applying mutation, evaluation function, frequent patterns and rule removal, and finally new generation stage. However, the new generation stage of MRS is modified here where the stopping criterion is the number of bases per sequence, which is a constant value with all the sequences. The stopping criterion will be the maximum number of bases a sequence can have. When sequences file is scanned, the

model will check the length of each sequence and get the maximum number of bases per sequence

in the sequences file under processing then uses it as the stopping criterion of MMRS model.

MMRS will stop generating new rules when stopping criterion is reached. This means that MMRS will generate number of population equals to the maximum length of sequence in DNA sequences database. We get two advantages of this modification. The first is that MMRS

can works with variant length sequences without having to align it. The second is that the search space is decreased with each population; this is because there are sequences with small lengths will be eliminated after each population where their length is smaller than the new candidate patterns generated from the next population.

Figure 1 depicts the flowchart of MRS and MMRS, which illustrate its skeleton with the sex stages:

**Example:** Consider the set of sequences D Shown in figure 2. Let the minimum support threshold be 8. Figure 3 shows th e complete set of sequential patterns F.

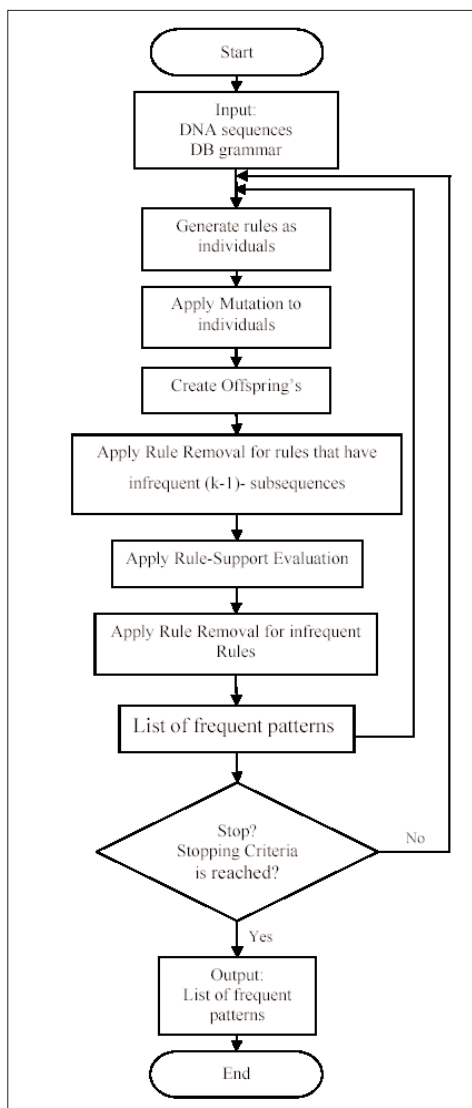
Figure 2. Sequences Data set D

1	t a a c c c t t t c
11	c c t a a g g t t a
21	a c c c t t a c c c
31	t a a c c c t t a c
41	c c t a a c c c t a
51	a c c c t a a c c c
61	t a a c c t t t g c
71	c c t a a g g c t t
81	a c c c t a a c c c
91	t a a c g g t t a c
101	c c t a a c c c a a
111	g g c t g g c c c t
121	a g g c t t a a c c

Figure 3. The complete set of sequential patterns F

F	count
aa	11
ac	10
cc	12
ct	12
ta	12
aac	9
acc	9
cct	11
taa	11
taac	9

Figure 1. Flowchart of MRS



### First Iteration:

The values of AC = {a, c, g, t }

The mutated rules are:

If	AC = a and	SC = a, then	CP = aa
If	AC = a and	SC = c, then	CP = ac
If	AC = a and	SC = g, then	CP = ag
If	AC = a and	SC = t, then	CP = at

If	AC = c and	SC = a, then	CP = ca
If	AC = c and	SC = c, then	CP = cc
If	AC = c and	SC = g, then	CP = cg
If	AC = c and	SC = t, then	CP = ct

If	AC = g and	SC = a, then	CP = ga
If	AC = g and	SC = c, then	CP = gc
If	AC = g and	SC = g, then	CP = gg
If	AC = g and	SC = t, then	CP = gt

If	AC = t and	SC = a, then	CP = ta
If	AC = t and	SC = c, then	CP = tc
If	AC = t and	SC = g, then	CP = tg
If	AC = t and	SC = t, then	CP = tt

The list of obtained frequent patterns from the second iteration are: F = { aa, ac, cc, ct, ta }

### Second Iteration:

The values of AC = {aa, ac, cc, ct, ta }

The mutated rules are:

If	AC = aa and	SC = a, then	CP = aaa
If	AC = aa and	SC = c, then	CP = aac
If	AC = aa and	SC = g, then	CP = aag
If	AC = aa and	SC = t, then	CP = aat

If	AC = ac and	SC = a, then	CP = aca
If	AC = ac and	SC = c, then	CP = acc
If	AC = ac and	SC = g, then	CP = acg
If	AC = ac and	SC = t, then	CP = act

If	AC = cc and	SC = a, then	CP = cca
If	AC = cc and	SC = c, then	CP = ccc
If	AC = cc and	SC = g, then	CP = ccg
If	AC = cc and	SC = t, then	CP = cct
If	AC = ct and	SC = a, then	CP = cta
If	AC = ct and	SC = c, then	CP = ctc
If	AC = ct and	SC = g, then	CP = ctg
If	AC = ct and	SC = t, then	CP = ctt
If	AC = ta and	SC = a, then	CP = taa
If	AC = ta and	SC = c, then	CP = tac
If	AC = ta and	SC = g, then	CP = tag
If	AC = ta and	SC = t, then	CP = tat

The list of obtained frequent patterns from the second iteration are: F = {aac, acc, cct, taa }

### Third Iteration:

The values of AC = {aac, acc, cct, taa }

The mutated rules are:

If	AC = aac and	SC = a, then	CP = aaca
If	AC = aac and	SC = c, then	CP = aacc
If	AC = aac and	SC = g, then	CP = aacg
If	AC = aac and	SC = t, then	CP = aact
If	AC = acc and	SC = a, then	CP = acca
If	AC = acc and	SC = c, then	CP = accc
If	AC = acc and	SC = g, then	CP = accg
If	AC = acc and	SC = t, then	CP = acct
If	AC = cct and	SC = a, then	CP = ccta
If	AC = cct and	SC = c, then	CP = cctc
If	AC = cct and	SC = g, then	CP = cctg
If	AC = cct and	SC = t, then	CP = cctt
If	AC = taa and	SC = a, then	CP = taaa
If	AC = taa and	SC = c, then	CP = taac
If	AC = taa and	SC = g, then	CP = taag
If	AC = taa and	SC = t, then	CP = taat

The list of obtained frequent patterns from the second iteration are: F = {taac }

## 4. Performance study

### 4.1. Complexity analysis of MRS and MMRS

#### Space Complexity:

For a sequence file with  $n$  as the number of sequences, and  $m$  as the number of bases per sequence and number of items equals to 4 which is the 4 DNA bases, the space complexity of MRS and MMRS is  $O(4^m + n)$ .

#### Time Complexity:

The complexity of generating all the candidate patterns is  $O(4m)$ . The complexity of get frequencies for the candidate patterns is  $O(n \cdot 4^m)$ . The complexity of discovering the frequent patterns is  $O(4^m)$ . The complexity of main function that calls the other functions with  $m$  number equals to the number of bases per sequence is  $O(m)$ .

The total time complexity for MRS and MMRS algorithms are:  $O(mn \cdot 4^m)$

### 4.2. Performance

To evaluate the efficiency of MRS and MMRS, we performed a study of Apriori algorithm on many samples of data sets with different sizes and lengths.

The implementation of the three algorithms was done at the same platform and at the same environment. The computer used to run the experiments was a Pentium4 1.70GHz with 128MB of RAM.

The operating system used was Windows XP.

To perform the study over a large range of different characteristics, we used the NCBI-GenBank Flat File Release 148.0, which is DNA sequences database published at Jun, 2005 obtained from the National Center for Biotechnology Information

(NCBI) as mentioned.

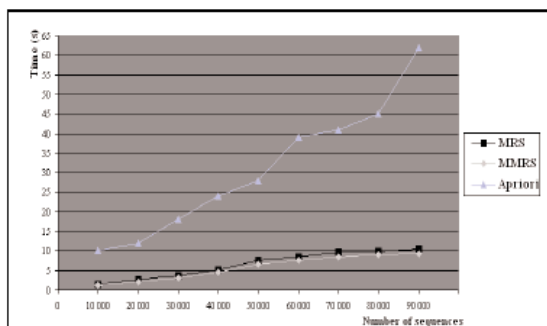
These data are in different sizes and different sequence lengths to cover all the cases in this evaluation. The evaluation of the efficiency of MRS and MMRS concerns three issues which are the changes of number of sequences in each trial, the changes of support threshold values, and the changes of number of bases per sequence. The result yielded from the experiment was as follows.

#### 4.2.1. Number of sequences

This performance study measures the performance of both MRS and MMRS versus Apriori according to the change in number of sequences. This issue is applied on data set with 23 files range from 10 000 sequences to 5000 000 sequences with fixed values of Support threshold. Each file represents a trial in the experiment. The execution time is measured in each trail. The result of this test shows the relationship between the numbers of transactions and the processing time in seconds or minutes according to the three algorithms.

As can be observed from figure 4(a,b,c), there is a direct proportional between number of sequences and the execution time where the higher the number of sequences, the higher consumed time. Also it is clear that MRS and MMRS outperform Apriori where time taken by them is much smaller than time taken by Apriori. From figure 4(c), it is clear that the performance of Apriori is unacceptable with high volume of data where it takes hours to processing data such as with 5000 000 sequences, MRS and MMRS take almost 10 minutes where Apriori takes 6 hours and half. Also, the performance result of MRS is very close to the performance result of MMRS but MMRS outperforms MRS. This is because the search space is decreased with each iteration of MMRS.

**Figure 4(a).** MRS and MMRS vs. Apriori under data ranges from 10 000 to 90 000 sequences.



#### 4.2.2. Number of bases

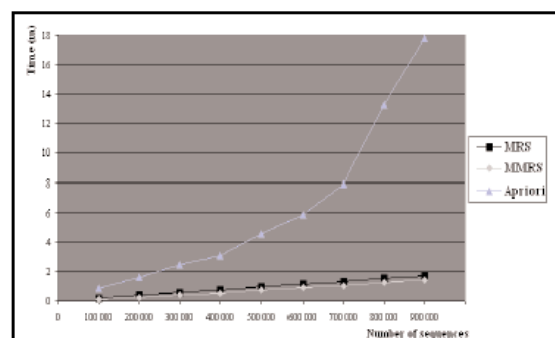
This issue measures the performance of MRS and MMRS versus Apriori according to the change in

number of bases per sequence with fixed values of support threshold and number of sequences. This issue is applied on data set containing 12 files with variant values of number of bases per sequence range from 50 bases per sequence (bps) to 600 bps.

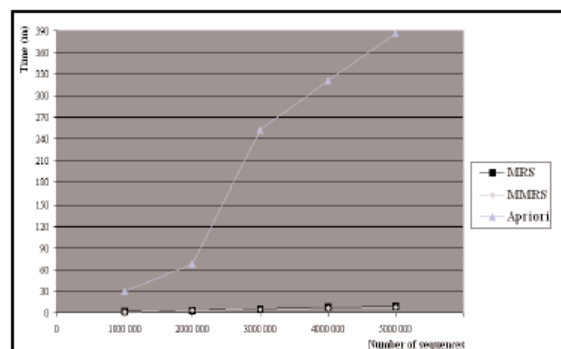
Regarding to the files used to measure performance of MMRS, ChromasPro program is used to prepare files with sequences have variant lengths where the maximum length of any sequence does not exceed the number of bases under testing. The result of this issue shows the relationship between the value of number of bases per sequence and the processing time in seconds according to the three algorithms.

As it is revealed from figure 5, there is a direct proportional between the number of bases per sequence and the execution time where the higher the value of number of bases per sequence, the higher the consumed time. Also it is clear that MRS and MMRS outperform Apriori in large number of bases per sequence.

**Figure 4(b).** MRS and MMRS vs. Apriori under data ranges from 100 000 to 900 000 sequences.



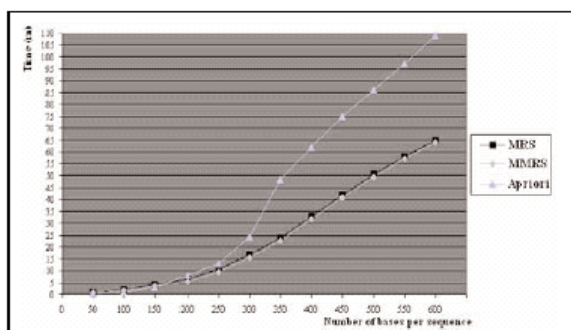
**Figure 4(c).** MRS and MMRS vs. Apriori under data ranges from 1000 000 to 5000 000 sequences.



#### 4.2.3. Support threshold

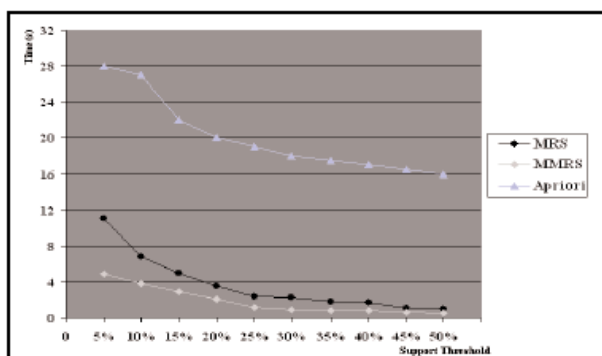
In the current performance study, the performance of MRS and MMRS versus Apriori is measured according to the changes in support threshold under two aspects which are the variant number of sequences and the variant number of bases per sequence.

**Figure 5.** MRS and MMRS vs. Apriori under variant number of bases.

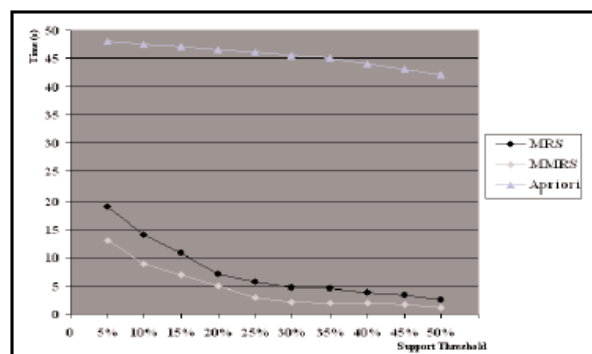


This first aspect is applied on the first data set with variant number of sequences and variant values of support threshold. Here, we do four tests. Starting support threshold 5% and varying support threshold with 5%, four tests are performed, with sequence file 30,000, 50,000, 75,000, and 100,000 sequences. The execution time is measured in each trail. The result of these tests shows the relationship between the value of the support threshold and the processing time in seconds according to the three algorithms. As it is revealed from figure 6, there is an inversely proportional between support threshold and the execution time where the higher the value of support threshold, the lower consumed time. Also it is clear that MRS and MMRS outperform Apriori where time taken by MRS and MMRS is much smaller than time taken by Apriori. It is interesting to note that the execution times with lower values of support threshold in MRS and MMRS are about three times faster than execution times with lower values of support threshold in Apriori and the execution times with higher values of support threshold in MRS and MMRS are about fifteen times faster than execution times with higher values of support threshold in Apriori. These results show that a great part of the efficiency of MRS is due to its efficiency of memory consumption. Also, the performance result of MRS is very close to the performance result of MMRS but MMRS outperforms MRS. This is because the search space is decreased with each iteration of MMRS as mentioned before.

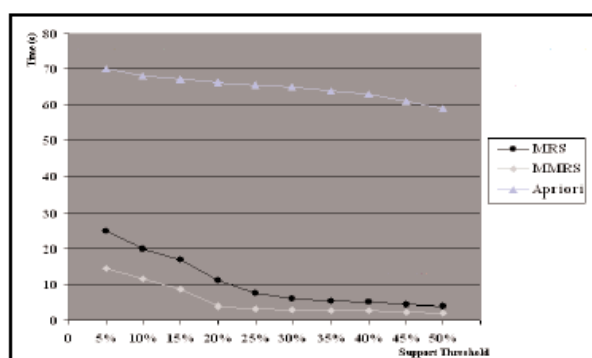
**Figure 6(a)**



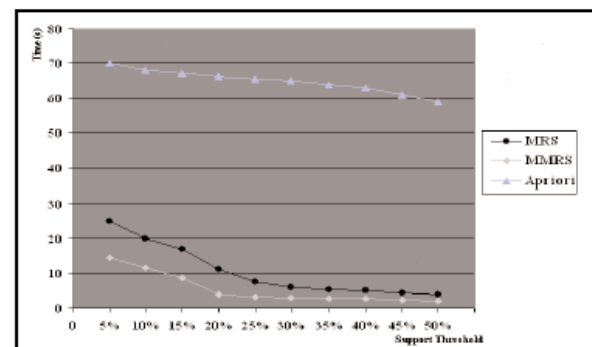
**Figure 6(b)**



**Figure 6(c)**



**Figure 6(d)**  
**Figure 6.** MRS and MMRS vs. Apriori under different support threshold and variant number of sequences.



This second aspect is applied on the data sets with variant number of bases and variant values of support threshold. Here, we do four tests. The first test is on the sequence file with 25 bases per sequence and variant support threshold. The second test is on the sequence file with 50 bases per sequence and variant support threshold. The third test is on the sequence file with 75 bases per sequence and variant support threshold. The fourth test is on the sequence file with 100 bases per sequence and variant support threshold. Each trial in each test of the experiment is represented by adding 5% to the support threshold value of

Figure 7(a). The first test with 25 bps

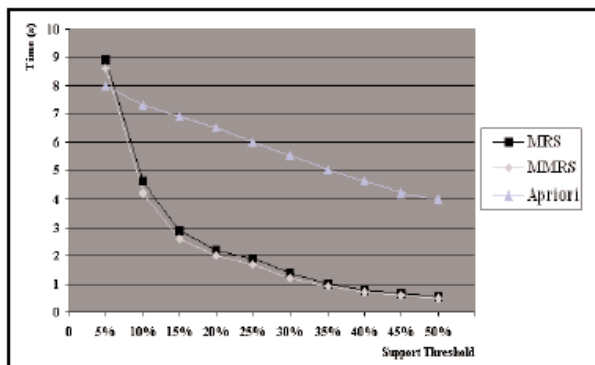


Figure 7(b). The second test with 50 bps

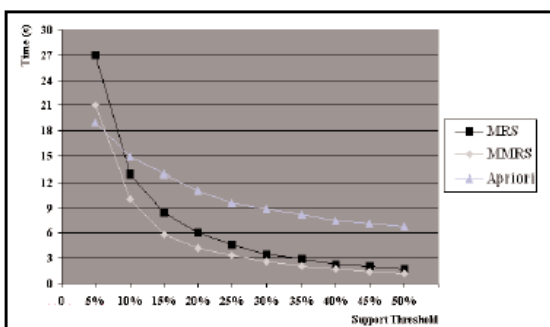


Figure 7(c). The third test with 75 bps

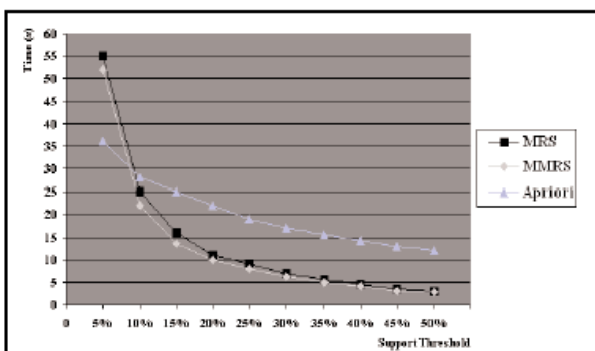
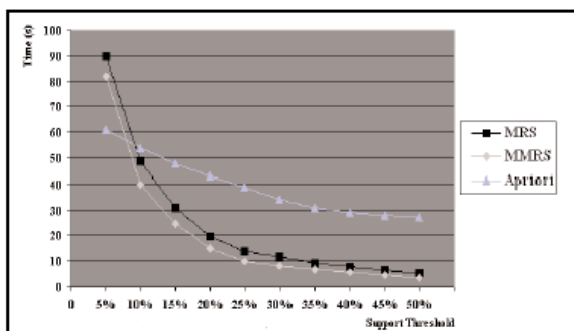


Figure 7(d). The fourth test with 100 bps  
 Figure 7. MRS and MMRS vs. Apriori under different support threshold and variant number of bases.



the previous trail. Thus the first trial with support threshold value equals to 5% and the last one with

support threshold value equals to 50%. The execution time is measured in each trail. The result of these tests shows the relationship between the value of the support threshold and the processing time in seconds according to the three algorithms.

It is clear from figure 7 that the performance result of MRS is very close to the performance result of MMRS but MMRS outperforms MRS. Also it is revealed from figure 7 that the performance of Apriori is better than MRS and MMRS in processing sequences with a small number of bases and a small support threshold values but with large support threshold values, MRS and MMRS outperform Apriori.

## 5. Conclusion and future work

### 5.1. Conclusion

In the current work, we have proposed a simple and novel model, MRS, for efficient mining of sequential patterns in the DNA sequences databases by employing the evolutionary algorithms. This model is applied on DNA sequences database to extract and identify frequent patterns, which benefit in many applications and have many significant biological implications. We made a modification to MRS model to can fit with DNA sequences file that contains sequences with variant length - MMRS. Most of the previous works deal with classification and clustering. An advantage of MRS relies in no database projection, and high CPU and memory utilization. MRS scans DNA sequences into memory in one pass. MRS discovers patterns directly from sequences in-memory which saves time.

In order to assess the performance of MRS and MMRS, the performance of MRS, MMRS, and Apriori algorithms are tested. The evaluation of the efficiency of MRS and MMRS concerns three issues which are: the changes of number of sequences in each trial, the changes of support threshold values, and the changes of number of bases per sequence. The conducted experiments show that the performance result of MRS is very close to the performance result of MMRS but MMRS outperforms MRS. This is because the search space is decreased with each iteration of MMRS as mentioned before.

Furthermore, MRS and MMRS run faster than Apriori algorithm even with the database with long sequence length and high number of sequences. The performance of Apriori is better than MRS and MMRS in processing sequences with a small number of bases and a small support threshold values. however, with large support threshold values and large number of bases per sequence, MRS and MMRS outperform Apriori. The performance of MRS and MMRS is very stable even when minimum support threshold is very low for large databases.

## 5.2. Future work

There are still many interesting issues related to both MRS and to MMRS. These extensions can be in several directions. For example, the usage of incremental mining of sequential patterns when new sequences are added to the original file can be more extensively studied. The second is to add the locations of each occurrence of sequential patterns in extended result file. The third is the use of regular expression constraints to form the grammar that initialize the rules. Finally, both MRS and MMRS can be customized to work on the regular transaction databases.

## References

- [1] NM Luscombe, D Greenbaum and M Gerstein. What is bioinformatics? An introduction and overview. 2001 International Medical Informatics Association Yearbook: 83-100, 2001.
- [2] R. Agrawal and R. Srikant. Mining sequential patterns. In Proc. Int. Conf. Data Engineering (ICDE'95), pages 3-14, Taipei, Taiwan, Mar. 1995.
- [3] M. Zhang, B. Kao, C.L. Yip, and D.W. Cheung. A gsb-based efficient algorithm for mining frequent sequences. In Proc. 2001 International Conference on Artificial Intelligence (IC-AI 2001), Las Vegas, Nevada, USA, June 2001.
- [4] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00), pages 355-359, Boston, MA, Aug. 2000.
- [5] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. Proc. 2001 Int. Conf. Data Engineering (ICDE'01), pages 215-224, Heidelberg, Germany, April 2001.
- [6] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. In Proc. Of Machine Learning Journal, Special issue on Unsupervised Learning (Doug Fisher, ed.), vol. 42 Nos. 1-2, pages 31-60 Jan/Feb 2001.
- [7] J. Ayres, J. Gehrke, T. Yiu, Discovery and Data Mining. Edmonton, Alberta, Canada, July 2002.
- [8] L. Ming-Yen and L. Suh-Yin. Fast discovery of sequential patterns by memory indexing. In Proc. of DaWaK2002, pages 150-160, France, September 4-6, 2002.
- [9] K. Wang, Y. Xu, J. Yu, Scalable Sequential Pattern Mining for Biological Sequences. In Proc. of CIKM 2004, Nov, 2004.
- [10] M. Leung Wong and K. Sak Leung. Data Mining Using Grammar Based Genetic Programming. Kluwer Academic Publishers; January 1, 2000.
- [11] A. Alex Freitas. Data Mining and knowledge Discovery with Evolutionary Algorithms. Springer-Verlag; March 2002.
- [12] J. Han and M. Kamber. Data Mining: Concepts and Techniques. New York: Morgan Kaufman 2001.