Mining Non-Contiguous Mutation Chain in Biological Sequences based on 3D-structure

Huang Wei

NATIONAL UNIVERSITY OF SINGAPORE
2011

Mining Non-Continguous Mutation Chain in Biological Sequences based on 3D-structure

Huang Wei
(B.COMP, SCU)

**A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF
SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF
SINGAPORE
2011**

Acknowledgment

I am thankful to Prof. Wynne Hsu and Prof. Mong Li Lee for their constant encouragement, guidance and support. I appreciate their vast knowledge in many areas, and their insights and suggestions that have helped to shape my research skills. I am also grateful to Dr Tong Joo Chuan and Dr Feng Mengling from A*STAR. They help me to verify the experiment results on the real world influenza A virus dataset in bioinformatics domain. Finally, I would like to thank Dr. Sheng Chang for providing me the data generator source code.

I offer my regards and blessings to all the students in the database group. I have enjoyed all the discussions we had on various topics, and I have lots of fun being a member of this fantastic group. I would especially like to thank Zhao Gang, Li Xiaohui, Han Zhen, Chen Qi, Patel Dhaval and all the other current members in Database lab 2. They are such good and dedicated friends who are always ready to lend a helping hand to me. Lastly, I thank my family for always being there when I needed them most and for supporting me in all these years.

Summary

Understanding how an infectious agent mutates from one form to another can provide insights into the mechanisms of disease pathogenesis and epidemiology. Existing methods of sequence analysis which focus on identifying regions of similarity may help explain functional or phenotypic variability. However, these approaches do not take into account the spatio-temporal dynamics of virus evolution. Recently, Sheng et. al [42] introduced an approach that incorporated spatio-temporal information to analyze mutation chains in influenza A proteomes. However, this work was restricted to mining contiguous subsequences of mutations, not taking into account the practical 3D-structure of the protein.

In this thesis, we generalize the definition for mutation chain to allow for mining of non-contiguous mutations. We design an efficient algorithm, termed $ptMutationChian - Miner$, to search for non-contiguous mutation chains in influenza A proteomes. This algorithm utilizes three pruning strategies *local hot positions*, *valid Mutation Space* and *increment join* to reduce the search space. Experiments on both synthetic and real world influenza A virus datasets show that the algorithm is effective in discovering non-continuous mutations that occur geographically over time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The influenza A virus is a major human pathogen. In order to infect the host, the pathogen can change its coat proteins from time to time by mutation and spread quickly across geographical regions by air-borne transmission. These factors account for seasonal influenza and occasional pandemic influenza [51].

Understanding how the fast evolving influenza A virus mutates from one form to another can provide insights into the mechanisms of disease pathogenesis and epidemiology, as well as the design of new therapeutic agents. In particular, it is important to know how the geographical spread of the influenza A virus evolving over time, and the trajectories of the said evolution.

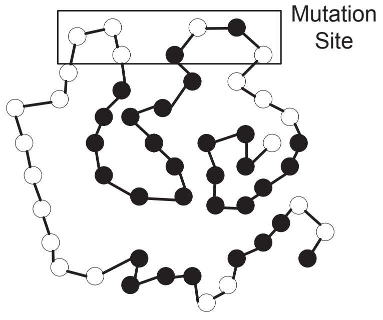


Figure 1.1: Example of non-continuous mutations on a folded protein.

In nature, a protein folds into a particular 3-D structure that allows it to effect a function. Therefore, as graphically demonstrated in Figure 1.1, functional changes of proteins are often caused by non-contiguous mutations. Incorporating space and time information, we develop the definition of the mutation chain whose co-mutations mostly occur in non-contiguous positions.

Table 1.1: An example of influenza A dataset

ID	Year	Country	Host	Aligned Sequences
vs_1	1986	Canada	Human	$ANTCVLEETKPGTQLFNHPD$
vs_2	1988	USA	Avian	$DNTCVLEETKSGYQLFTHPD$
vs_3	1989	Russia	Human	$DNTCVLEETKSGTQLFTHPD$
vs_4	1990	Canada	Swine	$DN-CVLEETKPGYQLF-HPD$
vs_5	1989	Vietnam	Human	$-NTCVLEETKPGTQLF-HPD$
vs_6	1994	Spain	Human	$-NMDVLEETKSGYQLF-HPD$
vs_7	1992	USA	Avian	$ANMDVLEETKSGTQLFNHPD$
vs_8	1994	Mexico	Swine	$DN--VLEETK-GYQLFTHPD$

An example of influenza A dataset is presented in Table 1.1. All virus subsequences are aligned and a representative sequence segment of twenty positions(1 . . . 20) is shown for illustration, including gaps (denoted as ”-”). To understand how a virus mutates from one strain to another, let us first analyze two highly conserved sequences, vs_4 and vs_7, with four amino acid differences between them. These two viruses are isolated in Canada and USA (i.e. countries which share a common border) within a viable period of two years. These factors suggest that vs_7 may have mutated from vs_4 as follows: ”D”,”C”,”P”,”Y” mutate to ”A”,”D”,”S”,”T” at positions 1,4,11,13 in order. Similarly, vs_7 could possibly mutate to vs_8 as there are only three amino acid differences between the two sequences. A closer examination reveals that vs_8 was isolated in Mexico after vs_7 in USA. This implies that the virus could have originated from Canada, spread to USA, and then move on to Mexico. We denote this movement of mutation as $< 1, 13 : DY \rightarrow AT \rightarrow DY >$, where 1 and 13 denote the positions where mutations have occurred. Finding such co-occurrences of the mutations over different time points is computationally expensive as the influenza viruses mutate continuously, resulting in a large number of variants. Existing algorithms are unable to scale up to such high complexity.

1.1 Objectives and Contributions

In this thesis, we define the concept of a non-contiguous mutation chain. To the best of our knowledge, the problem of discovering spatio-temporal patterns of non-contiguous mutation chains in influenza A virus has not been explored in current bioinformatics research. We summarize the contributions of this thesis as follows:

- We define the problem of mining non-contiguous mutation chain and introduce an interesting measurement, $Significance$, to capture the significance of the mutations.

- We present an integrated algorithm to discover non-contiguous subsequences of mutation chain. The algorithm utilizes a data structure, the PointMutation tree, to facilitate the mining process.

- We propose three pruning strategies to improve the mining efficiency. The first strategy prunes off the positions of each sequence that are unlikely to participate in the formation of valid point mutations. The second and third strategies aim to reduce the number of candidates generated by pruning away those sequence chains that are unlikely to support any valid mutation chains.

- We evaluate our algorithm on both synthetic and real world datasets. Experiments on the real world Influenza A virus dataset provide insights into the spread and mutation of the highly pathogenic Avian H5N1 influenza virus and the H3N2 subtype. The discovered mutations have also been validated against the outbreaks of influenza historically.

1.2 Organization

The thesis is organized as follows: Chapter 2 surveys the related work. Chapter 3 introduces some definitions. Chapter 4 describes our algorithm to mine non-contiguous mutation chains. Experimental results are presented in Chapter 5. We conclude this thesis and propose some future work in Chapter 6.

Chapter 2

Related Work

In this chapter we review existing works that are related to this thesis. We first introduce sequential pattern mining in Chapter 2.1 and describe the interestingness measures used in frequent pattern mining in Chapter 2.2. Next, we survey existing algorithms for spatio-temporal sequential patterns mining in Chapter 2.3. In Chapter 2.4, we examine the recent progress in bioinformatics domain.

2.1 Sequential Pattern Mining

Sequential pattern mining aims to discover frequent subsequences as patterns in a sequence database consisting of ordered elements or events. It has many useful applications such as the analysis of customer purchase behaviors, web access patterns, telephone calling patterns, science and engineering processes, medical and disease treatments, natural disasters (e.g., earthquakes), DNA sequences and gene structures, market stocks data, and so on.

Agrawal et. al. introduced the problem of sequential pattern mining problem in [5]. Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items. Items within an element are unordered. Given a user-specified support threshold, sequential pattern mining is to find complete set of the frequent subsequences that occur frequently in the dataset.

Given two sequences $\alpha = < a_1, a_2 \ldots a_n >$ and $\beta = < b_1, b_2 \ldots b_m >$. α is called a subsequence of β or β is a super sequence of α, denoted as $\alpha \subseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < \cdots < j_n \leq m$ such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}$, $\ldots$, $a_n \subseteq b_{jn}$.

Table 2.1: the example of sequence database

SID	sequence
1	<b(*bc*d)(b*d*)e(dg)>
2	<(be)d(cd)(bf)>
3	<(fg)(*bc*)(eg)*d*c>
4	<fh(bg)dcd>

Take the example of the sequence database in Table 2.1, the sequence <b(cd)ed> is a subsequence of <*b*(b*cd*)(bd)*e*(*d*g)>. Suppose the support threshold min_sup = 2, then <(bc)d> is a sequential pattern.

There are two popular approaches to perform sequential pattern mining, namely: Apriori-based approach and pattern-growth-based approach.

2.1.1 Apriori-based Sequential Mining

The Apriori property states that if a sequence S is not frequent, then none of the super-sequences of S is frequent. For example, consider the example in Table 2.1, suppose the support threshold min_sup = 2, if <gb> is infrequent, then <g(bc)e> is also not frequent.

Both GSP [46] and SPADE [54] utilize this property to reduce the search space by pruning the unpromising candidates.

GSP adopts a multiple-pass, candidate-generation-and-test approach. The basic idea is as follows: Initially, every item in the database is a candidate of length 1. For each level (i.e., sequences of length-k), we scan the database to compute support count for each candidate sequence and generate candidate length-(k+1) sequences from length-k frequent sequences. The algorithm terminates when no new sequential pattern is generated.

SPADE (Sequential PAttern Discovery using Equivalent Class) [54] employs a vertical formatting method with a lattice search technique. A sequence database is mapped to a large set of <SID, EID> in the form of a vertical id-list database format. And we associate each sequence with a list of objects, in which it occurs, along with the time-stamps. Therefore all frequent sequences can be enumerated via simple temporal joins (or intersections) on id-lists. Another lattice-theoretic approach is to decompose the original search space (lattice) into smaller pieces (sub-lattices) which can be processed independently in main-memory. This approach usually requires three database scans, or only a single scan with some pre-processed information.

There are many other studies [9, 14, 16, 29, 31, 36, 45] which have utilized the Apriori property to aid in the efficient mining of sequential patterns or other frequent patterns in time related data. However, these methods all suffer from the limitations of requiring multiple scans of the database and generating a huge set of candidate sequences. As a result, they are not suitable for mining long sequential patterns.

2.1.2 Pattern-Growth-based Approaches

Inspired by Agarwal et al. [2] and J. Han et al. [19], pattern-growth-based approaches have been proposed to mine long sequential patterns. The basic idea is to facilitate sequential pattern mining through projecting the database. There are two typical pattern-growth-based methods: FreeSpan [18] and PrefixSpan [38].

FreeSpan (Frequent pattern projected Sequential pattern mining) uses the frequent items to recursively project sequence databases into a set of smaller projected databases and grow subsequence fragments in each projected database. This process partitions both the data and the set of frequent patterns to be tested, and confines each test being conducted to the corresponding smaller projected database. However, since a subsequence may be generated by any substring combination in a sequence, projection in FreeSpan has to keep the whole sequence in the original database without length reduction. Moreover, since the growth of a subsequence is explored at any split point in candidate sequence, it is costly.

In order to overcome the bottleneck of FreeSpan, J. Pei et al. proposed the PrefixSpan [38] algorithm. Instead of projecting sequence databases by considering all the possible occurrences of frequent subsequences in FreeSpan, the projection of PrefixSpan is based only on frequent prefixes because any frequent subsequence can always be found by growing a frequent prefix. Hence, PrefixSpan examines only the prefix subsequences and project only their corresponding postfix subsequences into the projected databases. In each projected database, sequential patterns are grown by exploring only local frequent patterns which support the short frequent patterns for the mining of longer patterns.

However, these algorithms do not adapt well to the problem of mining mutation chains where the transactions consists of exponential number of mutations and is positional-dependent.

2.2 Interestingness Measures in Association Patterns Mining

The essence of association rule mining is to analyze the relationships among variables and find those interesting association rules [4]. There are many applications of association rules mining, particularly in finding associations among items in customer transactions [6, 17, 20, 21, 32, 37, 41, 1, 47, 53].

To identify the interesting association rules, correlation has been adopted as an interestingness measure. This measure aims to identify groups of variables which are strongly correlated with each other or with a specific target variable. Based on the correlation measure, we are able to capture the dependencies among variables.

Another interestingness measure is the lift measure as proposed by Brin et. al. [10]. However, the lift measure does not satisfy the downward closure property [7]. As a results, several other interestingness measurements have been proposed and extensively studied to capture the interestingness of association patterns [27, 43, 3, 44, 28]. In addition, the works in [34, 48] mention about the criteria for selecting the suitable interestingness measures for different applications.

2.3 Spatio-temporal Sequential Patterns Mining

Spatio-temporal sequential patterns are useful in the investigation of spatio-temporal evolutions of phenomena in many application fields. However, straightforward application of existing sequential pattern mining methods to spatio-temporal data by "transactionization" of spatial and temporal domains may be unnatural due to the continuity of space and time [23]. The main problem is that it is highly possible to miss the spatial, temporal, or spatio-temporal relationships which are across partition/transaction boundaries in a disjoint partitioning; and because of an overlapping partitioning, a relationship may be counted more than once. Recently, Huang et. al [24] proposed a framework for mining sequential patterns from event data. They defined the neighborhood of an event within the space-time dimension and proposed a significance measure that considers the density of event type.

Another type of spatio-temporal data is the trajectory data. A trajectory is a sequence of the locations and timestamps of a moving object. Mamoulis et al. [30, 11, 15] discussed the indexing, querying and mining of trajectory data. Retrieving similar trajectories can reveal the underlying traveling patterns of moving objects in the data. Example applications include homeland security (e.g., border monitoring), law enforcement (e.g., video surveillance), weather forecast, traffic control, location-based service. Mamoulis et. al.

proposed models and algorithms to investigate the trajectories of objects for mining frequent periodic subtrajectory, which consists of a sequence of frequently visited places on trajectories.

2.4 Bioinformatics domain

In the bioinformatics domain, sequential pattern mining techniques have been applied to biological databases to find interesting protein or genome patterns [50, 22]. A biosequence has the following characteristics:

- It has a very small alphabet. For example, 20 for protein sequences and 4 for DNA sequences.
- It has a vary long sequence length of few hundreds, sometime thousands.
- It may contain gaps over long regions.

Because of the above characteristics, it is infeasible to enumerate the entire solution space. The works in [33, 49, 25, 40] make use of heuristics or structural constraints, such as the maximum gaps allowed or the maximum pattern length, to reduce the search space.

Recently, the framework proposed by Huang et. al [24] can discover long, single point mutations (i.e., mutations which occur multiple times at a specific position) across multiple sequences. However, they are unable to find co-mutations involving multiple positions. Other works try to utilize the translation probability matrix to estimate the future composition of amino acids [52, 26], but these works only consider the mutation in one position and cannot analyze how the mutations spread geographically over time.

Sheng et. al [42] proposed a different framework to mine co-mutations across multiple sequences. However, the algorithm does not take into account the 3D-structure of protein and mines only the mutations that occur in k contiguous positions. This restriction to continuous positions may result in missing some biologically meaningful patterns.

Chapter 3

Preliminaries and Definitions

A virus protein sequence dataset $vPSD$ consists of a set of virus protein records, vs_1, vs_2, ..., vs_n, where n is the size of the dataset. Each record has a unique id, virus host, time, location, and the protein sequence. The virus sequences are preprocessed by a multiple sequence alignment so that all sequences have identical number of positions where each position is an amino acid or a gap, denoted as "-" (see Table 1.1).

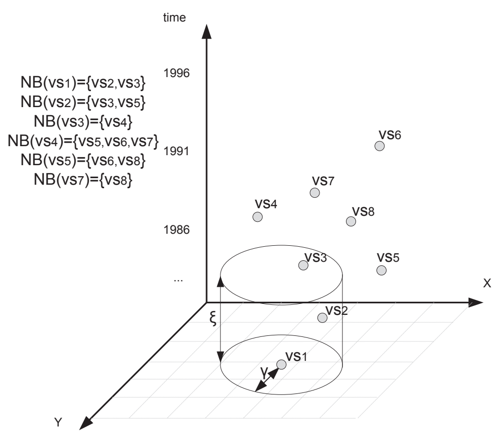


Figure 3.1: Spatio-temporal representation of the viruses in Table 1.1.

Suppose we have two virus sequences vs and vs' that are near in space and time. We say that vs' is in the neighbourhood of vs, denoted by vs'

$\in NB(vs)$. Then vs mutates to vs' if we can find a transformation that maps vs to vs'. Consider the two virus sequences vs_1 and vs_2 in Figure 3.1. We observe that vs_1 and vs_2 are within the same cylinder indicating they are near in space and time. Also, we can transform vs_1 to vs_2 by changing A,P,T,N to D,S,Y,T at positions 1,11,13,17 in order. Hence, we say vs_1 mutates to vs_2.

Definition Let c_i to be the i-th character of sequence vs and c'_i to be the i-th character of sequence vs'. vs is said to **point mutate** or **1-mutate** to vs', if and only if $vs' \in NB(vs)$ and there exists $p \in [1, n]$ such that $c_p \neq c'_p$ but for all $i \neq p$, $c_i = c'_i$. We denote the point mutation at position p as $\langle p, c_p \to c'_p \rangle$. Moreover, the virus sequence pair,(vs, vs'), is said to *support* the point-mutation.

We denote a set of k point mutations as $M = \{< p_1 : c_{p_1} \to c'_{p_1} >, < p_2 : c_{p_2} \to c'_{p_2} > \cdots < p_k : c_{p_k} \to c'_{p_k} >\}$. The set of positions where the point mutations occur is given by $Pos = \{p_1, p_2, \cdots, p_k\}$. A virus sequence pair (vs_i, vs_j) is said to *support* M if $vs_j \in NB(vs_i)$, and $\forall\ p \in Pos$, $c_p \in vs_i$ and $c'_p \in vs_j$.

For example, given a virus sequence $vs = ACDE$ and another sequence $vs' = ARDF$ and $vs' \in NB(vs)$. Suppose $M = \{< 2 : C \to R >, < 4 : E \to F >\}$ with $Pos = \{2, 4\}$. Then (vs, vs') supports M.

Definition Given a set of virus pairs (vs_i, vs_j) that support M, let $VS[i]$ be the set of distinct vs_i and $VS[j]$ be the set of distinct vs_j. Then

$$Support(M) = min(|VS[i]|, |VS[j]|)$$

Definition Let $VPairs_p$ be the set of virus pairs that support the point mutation at position p in M. We define the mutation significance of M as follows:

$$Significance(M) = \frac{Support(M)}{max_{p \in Pos}(|VPairs_p|)}$$

The *Significance* measure indicates the likelihood of M occurring with respect to the individual point mutations. A value close to 1 implies that the likelihood of M occurring is high.

For example, in Figure 3.1, we have a set of 2 point mutations

$$M = \{< 1 : A \rightarrow D >, < 11 : P \rightarrow S >\}$$

The set of virus pairs that support M is $\{(vs_1, vs_2), (vs_1, vs_3)\}$. Then $VS[i] = \{vs_1\}$ and $VS[j] = \{vs_2, vs_3\}$. We have

$$\begin{aligned} Support(M) &= min(|VS[i]|, |VS[j]|) \\ &= min(1, 2) = 1 \end{aligned}$$

In order to calculate $Significance(M)$, we first need to compute the sets of virus pairs that support the point mutations at positions 1 and 11 respectively. We have $VPair_1 = \{(vs_1, vs_2), (vs_1, vs_3), (vs_7, vs_8)\}$ and $VPair_{11} = \{(vs_1, vs_2)\ , (vs_1, vs_3)\ , (vs_4, vs_6)\ , (vs_4, vs_7)\ , (vs_5, vs_6)\}$. Then

$$\begin{aligned} Significance(M) &= \frac{Support(M)}{max(|VPair_1|, |VPair_{11}|)} \\ &= \frac{1}{max(3, 5)} \\ &= 0.2 \end{aligned}$$

Definition Suppose we have a set of k point mutations $M = \{< p_1 : c_{p_1} \rightarrow c'_{p_1} >, < p_2 : c_{p_2} \rightarrow c'_{p_2} > \cdots < p_k : c_{p_k} \rightarrow c'_{p_k} >\}$ with $Pos = (p_1, p_2, \ldots, p_k)$. For $\forall\ p_i \in Pos$, if $(c_{p_i}, c'_{p_i}) \in M$, we can get $(c_{p_i}, c'_{p_i}) \in M'$ (another set of k point mutations). Then M is the *sub k point mutations* of M', denoted as $M \sqsubseteq M'$.

For example, a set of 2 point mutations $M = \{< 1 : C \rightarrow R >, < 3 : E \rightarrow F >\}$ is a *sub k point mutations* of a set of 3 point mutations $M' = \{< 1 : C \rightarrow R >, < 3 : E \rightarrow F >, < 6 : G \rightarrow H >\}$.

To capture the sequence of mutations that happen over multiple time points, we define the concept of a mutation chain.

Definition A **mutation chain** MC of length $(T+1)$ is given by $M_1 \rightarrow M_2 \rightarrow \dots M_i \rightarrow \dots\ M_T$, where M_i is the set of k point mutations at the i^{th} time point. The Pos of MC denoted its mutation positions set. $M_1 \dots\ M_i$ and MC, where $i \in [1,T]$, have the same Pos; and for each sequence pair $(vs_j, vs_h) \in$ the set of virus pairs that supports M_i, there must be sequence pair $(vs_h, vs_q) \in$ the set of virus pairs that supports $M_{(i+1)}$, where $j \neq h$, $h \neq q$, $j,h,q \in [1,n]$, $vs_h \in NB(vs_j)$ and $vs_q \in NB(vs_h)$.

A chain of sequences, $vs_1 \rightarrow vs_2 \rightarrow vs_3 \rightarrow \dots\ \rightarrow vs_{(T+1)}$, is said to support the mutation chain MC, if (vs_i, vs_{i+1}) supports the M_i, $i \in [1,T]$.

In Figure 3.1, we can see that $vs_7 \in NB(vs_4)$ and $vs_8 \in NB(vs_7)$. The chain of sequences $vs_4 \rightarrow vs_7 \rightarrow vs_8$ is said to support the mutation chain $MC = M_1 \rightarrow M_2$, where $M_1 = \{<1 : D \rightarrow A>, <13 : Y \rightarrow T>\}$, $M_2 = \{<1 : A \rightarrow D>, <13 : T \rightarrow Y>\}$ (or $MC = <1, 13 : DY \rightarrow AT \rightarrow DY>$ in short).

Definition A mutation chain $MC = M_1 \rightarrow M_2 \rightarrow \cdots \rightarrow M_T$ with Pos, if MC is a *sub mutation chain* of another mutation chain $MC' = M'_1 \rightarrow M'_2 \rightarrow \cdots \rightarrow M'_{T'}$ with Pos', denoted as $MC \sqsubseteq MC'$, if and only if

1) $Pos \subseteq Pos'$; $T \leq T'$.

2) $\forall i \in [1,T]$, $\exists r \in [0, T'-T]$ such that $M_i \sqsubseteq M'_{(i+r)}$.

Specifically, $MC = MC'$ if $MC \sqsubseteq MC'$ and $MC' \sqsubseteq MC$.

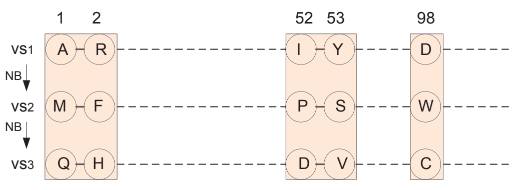
Figure 3.2 shows a mutation chain with $|Pos|$=5, and another mutation chain with $|Pos| = 9$, and the first chain is a sub mutation chain of the second one.

Definition The support of $MC = M_1 \rightarrow M_2 \rightarrow \dots M_i \rightarrow \dots M_T$, is defined as

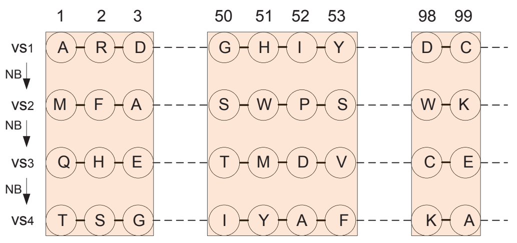
$$Support(MC) = min_{i \in [1,T]}\{Support(M_i)\}$$

Definition The mutation significant of $MC = M_1 \rightarrow M_2 \rightarrow \dots M_i \rightarrow \dots M_T$, is defined as

$$Significance(MC) = min_{i \in [1,T]}\{Significance(M_i)\}$$



(a) One mutation chain



(b) Another mutation chain

Figure 3.2: Examples of mutation chains. The mutation chain in (a) is a sub mutation chain of the mutation chain in (b)

For example, in Figure 3.1, we have a mutation chain $MC = M_1 \rightarrow M_2$, where $M_1 = \{< 1 : D \rightarrow A >, < 13 : Y \rightarrow T >\}$, $M_2 = \{< 1 : A \rightarrow D >, < 13 : T \rightarrow Y >\}$.

$$\begin{aligned} Support(MC) &= min(Support(M_1), Support(M_2)) \\ &= min(1, 2) = 1 \end{aligned}$$

,where we can easily calculate that $Support(M_1)$=1 and $Support(M_2)$=2. In the same reason, we can compute the $Significance(M_1)$ and $Significance(M_2)$, and they are 0.25, 0.4 in order, then

$$\begin{aligned} Significance(MC) &= min(Significance(M_1), Significance(M_2)) \\ &= min(0.25, 0.4) = 0.25 \end{aligned}$$

Both $Support(MC)$ and $Significance(MC)$ satisfy anti-monotone property and the proof about $Significance(MC)$ is as follows: ($Support(MC)$ is obviously satisfiable)

Lemma 3.0.1. ***Anti-monotonicity Property.*** *Given two mutation chains* $MC \sqsubseteq MC'$, $Significance(MC') \leq Significance(MC)$.

Proof: Given a mutation chain $MC= M_1 \rightarrow M_2 \rightarrow \dots M_i \cdots \rightarrow \dots M_T$ with Pos and another mutation chain $MC'= M'_1 \rightarrow M'_2 \rightarrow \dots M'_i \cdots \rightarrow \dots M'_T$ with Pos'. Without loss of generality, $MC \sqsubseteq MC'$, so that 1) $Pos \subseteq Pos'$ 2) $\forall\ i \in [1, T]\ \exists\ r \in [0, T' - T]$ such that $M_i \sqsubseteq M'_{(i+r)}$. By definition of sub mutation chain, if a sequence chain $vs_1 \rightarrow vs_2 \rightarrow vs_3 \rightarrow \dots \rightarrow vs_T$ supports MC', it must also support MC. So $\forall$ $1 \leq i < T$, we have

$$\begin{aligned}
& Significance(M'_{(i+r)}) \\
= \ & \frac{Support(M'_{(r+i)})}{max_{q \in Pos'}(|VPairs_q\ in\ M'_{(i+r)}|)} \\
\leq \ & \frac{Support(M'_{(r+i)})}{max_{q \in Pos}(|VPairs_q\ in\ M'_{(i+r)})|)} \\
\leq \ & \frac{Support(M_i)}{max_{q \in Pos}(|VPairs_q\ in\ M_i|)} \\
= \ & Significance(M_i)
\end{aligned}$$

$$\begin{aligned}
& Significance(MC') \\
= \ & min\{Significance(M'_t), \dots, Significance(M'_{T'})\} \\
\leq \ & min\{Significance(M'_{(1+r)}), \dots, Significance(M'_{(T+r)})\} \\
\leq \ & min\{Significance(M_1), \dots, Significance(M_T)\} \\
= \ & Significance(MC) \quad \blacksquare
\end{aligned}$$

Given a mutation significance threshold $min\_Significance$ and a support threshold $min\_Support$, a mutation chain MC is *valid* if and only if $Support(MC) \geq min\_Support$, and $Significance(MC) \geq min\_Significance$.

Chapter 4

Mining Non-Contiguous Mutation Chains

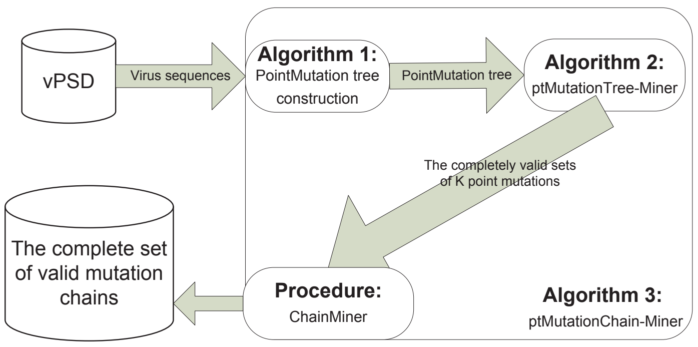


Figure 4.1: The mutation chains mining framework.

Figure 4.1 shows the proposed framework for mining non-contiguous mutation chains. Given the virus protein sequence dataset $vPSD$, we first construct the PointMutation tree which keeps track of the complete sets of k point mutations. To obtain the valid sets of k point mutations, we traverse the constructed PointMutation tree recursively, generating the sets of k point mutations that are both frequent and significant by concatenating the suffix. Having obtained the valid sets of k point mutations, we initiate procedure

ChainMiner to generate the complete set of valid mutation chains by linking the mutations across different time points.

4.1 Mining k point mutations

Given a virus protein sequence dataset $vPSD$, we first generate the set of single point mutations. We then extend this set of single mutation to k point mutations by constructing the PointMutation tree. Based on the constructed PointMutation tree, we design a recursive algorithm to mine the valid sets of k point mutations.

Finding the set of k point mutations is computationally expensive, especially when the length of the virus sequence is long. In order to reduce the complexity, we introduce the notion of *local hot positions* to identify positions that have a high probability of mutation. We use the entropy measure to determine the likelihood of mutation occurring at a position. This measure is defined as follows:

Definition Given a virus vs, let $V = NB(vs) \bigcup \{vs\}$ and Freq(c,vs,p) be the number of times the character c appears at position p in the virus sequences in V. We have

$$\begin{aligned} Entropy(vs,p) &= -\sum_{c \in A} Prob(c,vs,p) \log_2 Prob(c,vs,p) \\ where\ \ Prob(c,vs,p) &= \frac{Freq(c,vs,p)}{|V|} \end{aligned}$$

and A is the set of 20 standard amino acids characters and the gap character '-'.

If the entropy value exceeds a certain user-given threshold, we say that the position is a local hot position.

Consider vs_1 in Figure 3.1. We have $NB(vs_1) = \{vs_2, vs_3\}$ and $V = \{vs_1\} \bigcup NB(vs_1) = \{vs_1, vs_2, vs_3\}$. The characters that occur in position 1 of the virus sequences in V are A,D.

$$\begin{aligned} Prob(A, vs_1, 1) &= \frac{freq(A, vs_1, 1)}{|V|} = 1/3 \\ Prob(D, vs_1, 1) &= \frac{freq(D, vs_1, 1)}{|V|} = 2/3 \end{aligned}$$

The entropy value for vs_1 at the first position,

$$Entropy(vs_1, 1) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} \approx 0.918295.$$

Suppose the threshold is 0.5, we conclude that position 1 of vs_1 is a local hot position. Repeating the same analysis, the local hot positions for vs_1, vs_2, vs_3, vs_4, vs_5, vs_7 are (1,11,13,17), (1,11,13,17), (1,3,11,13,17), (1,3,4,11,13,17), (1,3,4,11,13,17), (1,3,4,11,13,17) in order, and because NB(vs_6) and NB(vs_8) is null, then all the positions of vs_6 and vs_8 are hot positions in default.

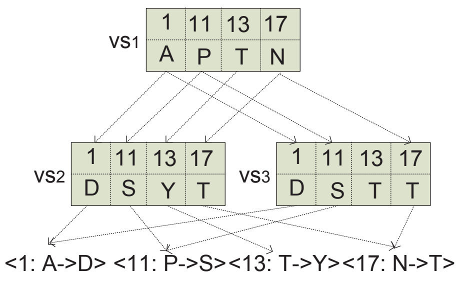


Figure 4.2: Example to show the generation of sets of k point mutations

After finding all the local hot positions in the virus sequences in $vPSD$, we generate the sets of k point mutations by comparing the common local hot positions between the virus and its neighborhood without regard to the gaps. For example, consider virus vs_1 in Figure 4.2 where $NB(vs_1) = \{vs_2, vs_3\}$. For the virus pair (vs_1, vs_2), their common local hot positions are $1, 11, 13$ and 17. From them, we generate a set of 4 point mutations $\{< 1 : A \rightarrow D >$, $< 11 : P \rightarrow S >$,$< 13 : T \rightarrow Y >$,$< 17 : N \rightarrow T >\}$. Next, we consider the virus pair (vs_1, vs_3). Their common local hot positions are $1, 11, 13, 17$. We observe that the characters at position 13 in both vs_1 and vs_3 are the same T, hence we have a set of 3 point mutations $\{< 1 : A \rightarrow D >, < 11 : P \rightarrow S >, < 17 : N \rightarrow T >\}$. In the same reason, based on all the virus pairs in Figure 3.1, we can generate the *mutation base* (Table 4.1) of our example, which is composed by virus pairs and their corresponding sets of k point mutations.

As every set of k point mutations M is generated, we need to evaluate the $Support(M)$ and $Significance(M)$ values to determine whether M is valid. However, this evaluation is computationally expensive as it involves finding the supporting virus pairs for all possible subsets of M which, in the worst case, is exponential to the length of the virus sequences.

Table 4.1: Mutation base: Virus pairs and their corresponding sets of k point mutations

virus pair	k point mutations
(vs_1, vs_2)	$< 1 : A \rightarrow D >, < 11 : P \rightarrow S >, < 13 : T \rightarrow Y >, < 17 : N \rightarrow T >$
(vs_1, vs_3)	$< 1 : A \rightarrow D >, < 11 : P \rightarrow S >, < 17 : N \rightarrow T >$
(vs_2, vs_3)	$< 13 : Y \rightarrow T >$
(vs_2, vs_5)	$< 11 : S \rightarrow P >, < 13 : Y \rightarrow T >$
(vs_3, vs_4)	$< 11 : S \rightarrow P >, < 13 : T \rightarrow Y >$
(vs_4, vs_5)	$< 13 : Y \rightarrow T >$
(vs_4, vs_6)	$< 4 : C \rightarrow D >, < 11 : P \rightarrow S >$
(vs_4, vs_7)	$< 1 : D \rightarrow A >, < 4 : C \rightarrow D >, < 11 : P \rightarrow S >, < 13 : Y \rightarrow T >$
(vs_5, vs_6)	$< 3 : T \rightarrow M >, < 4 : C \rightarrow D >, < 11 : P \rightarrow S >, < 13 : T \rightarrow Y >$
(vs_5, vs_8)	$< 13 : T \rightarrow Y >$
(vs_7, vs_8)	$< 1 : A \rightarrow D >, < 13 : T \rightarrow Y >, < 17 : N \rightarrow T >$

We transform this problem to the problem of frequent itemset mining [5]: a point mutation corresponds to an item in the frequent itemset mining problem. The sets of k point mutations(mutation base) correspond to the transaction dataset. Finding the sets of valid k point mutations is equivalent to finding the set of frequent k itemsets. For each single point mutations that can be found in the mutation base (Table 4.1), we generate a *statistic table* consisting of the support and significance values (see Table 4.2). The point

Table 4.2: Statistic table: Point mutations and their supporting virus pairs. (min_Support=2 and min_Significance=0.4)

M	$\{(vs_i, vs_j)\}$	$VS[i]$	$VS[j]$	$Sup(M)$	$Signi(M)$
$< 1 : A \to D >$	$\{(vs_1, vs_2),(vs_1, vs_3),(vs_7, vs_8)\}$	$\{vs_1, vs_7\}$	$\{vs_2, vs_3, vs_8\}$	2	0.67
$< 1 : D \to A >$	$\{(vs_4, vs_7)\}$	$\{vs_4\}$	$\{vs_7\}$	1	1
$< 3 : T \to M >$	$\{(vs_5, vs_6)\}$	$\{vs_5\}$	$\{vs_6\}$	1	1
$< 4 : C \to D >$	$\{(vs_5, vs_6),(vs_4, vs_6),(vs_4, vs_7)\}$	$\{vs_4, vs_5\}$	$\{vs_6, vs_7\}$	2	0.67
$< 11 : P \to S >$	$\{(vs_1, vs_2),(vs_1, vs_3),(vs_4, vs_6),$ $(vs_4, vs_7),(vs_5, vs_6)\}$	$\{vs_1, vs_4, vs_5\}$	$\{vs_2, vs_3,$ $vs_6, vs_7\}$	3	0.6
$< 11 : S \to P >$	$\{(vs_3, vs_4),(vs_2, vs_5)\}$	$\{vs_2, vs_3\}$	$\{vs_4, vs_5\}$	2	1
$< 13 : T \to Y >$	$\{(vs_1, vs_2),(vs_3, vs_4),(vs_5, vs_6),$ $(vs_7, vs_8),(vs_5, vs_8)\}$	$\{vs_1, vs_3,$ $vs_5, vs_7\}$	$\{vs_2, vs_4,$ $vs_6, vs_8\}$	4	0.8
$< 13 : Y \to T >$	$\{(vs_2, vs_3),(vs_2, vs_5),(vs_4, vs_5),$ $(vs_4, vs_7)\}$	$\{vs_2, vs_4\}$	$\{vs_3, vs_5, vs_7\}$	2	0.5
$< 17 : N \to T >$	$\{(vs_1, vs_2),(vs_1, vs_3),(vs_7, vs_8)\}$	$\{vs_1, vs_7\}$	$\{vs_2, vs_3, vs_8\}$	2	0.67

mutations whose support and significance values fall below the thresholds are invalid and will not participate in the generation of the valid sets of k point mutations, $k > 1$.

Next, we extend the valid single point mutations to find valid sets of k point mutations by constructing a PointMutation tree. The tree has a root labeled as *null* at level 0 and a set of nodes labeled with a point mutation. A path from the root to a level k node corresponds to a set of k point mutations. It is similar to the FP-tree [19] but with one subtle difference. Due to the interesting measurements used in this application, simply summing the number of occurrences of supported virus pairs is insufficient. Consider two k point mutations in the Figure 4.3: $M = \{ < 13 : T \to Y >, < 11 : P \to S >, < 4 : C \to D > \}$ and it's supported virus pair is (vs_5, vs_6); another one is $M' = \{ < 13 : T \to Y > \}$ and it's supported virus pair is (vs_5, vs_8). We observe that $M'' = \{ < 13 : T \to Y > \}$ is a common sub k point mutations of both M and M'. However, the support of M'' is not support of M + support of M'. Instead, can only be calculated based on its supported virus pairs(vs_5, vs_6) and (vs_5, vs_8). To overcome this, we store the set of the supported sequence pairs instead of just one count value.

Now, based on Table 4.2, we remove the non valid point mutations $< 1 : D \to A >$ and $< 3 : T \to M >$ in mutation base and reorder each virus pair's valid point mutations in the support-descending order. Then, we

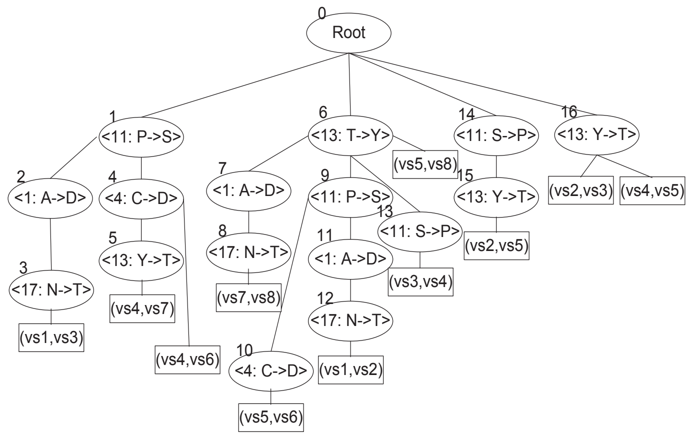


Figure 4.3: PointMutation tree.

can generate the PointMutation tree of our example as shown in Figure 4.3. The leftmost branch in the tree indicates a set of 3 point mutations $\{< 11 : P \rightarrow S >, < 1 : A \rightarrow D >, < 17 : N \rightarrow T >\}$. We associate each path with its supporting virus pairs. For example, the set of 3 point mutations $\{< 11 : P \rightarrow S >, < 1 : A \rightarrow D >, < 17 : N \rightarrow T >\}$ is supported by the virus pair (vs_1, vs_3). Clearly, if a virus pair supports a length k path of the PointMutation tree, it will support all its prefix paths. Hence, a bottom-up recursive algorithm is utilized to discover all valid sets of k point mutations.

Algorithm 1 gives the details of the PointMutation tree construction process. Based on the neighborhood relationships, lines 9-13 generate the set of k point mutations for each virus pair in $vPSD$ and store them in $MBase$. Given the $min\_Support$ and $min\_Significance$, line 14 determines whether the single point mutations in $Mbase$ are valid. The invalid single point mutations are removed from further consideration in lines 15-19. Line 20 initializes the PointMutation tree. Lines 21-29 construct the PointMutation tree by inserting each set of k point mutations from $MBase$ into the tree. Lines 31-34 give the insertTree, which handles every point mutation in all sets of k point mutations. Its main task is to determine whether this point mutation is equal to some existing tree node and whether can combine them

Algorithm 1: PointMutation tree construction

```
1: input:
2: $vPSD$: influenza A virus protein sequence database;
3: $Localhot$: the threshold value for local hot positions;
4: $min\_Support$: the minimal support;
5: $min\_Significance$: the minimal mutation significance;
6: output:
7: PointMutation tree, the PointMutation tree of vPSD;
8:
9: perform local hot position pruning strategy;
10: for virus pair $(vs_i, vs_j)$ that satisfies the neighborhood constraint do
11:     $M$ = {point mutations corresponding to the hot positions of both $vs_i$
        and $vs_j$) };
12:     $MBase = MBase \cup \{M\}$
13: end for
14: Generate the statistic table for all the single point mutations found in
    $MBase$
15: for M $\in MBase$ do
16:     if $\exists$ a single point mutations in $M$ that is invalid then
17:         $MBase = MBase$ - $\{M\}$
18:     end if
19: end for
20: initialize the node $root$ of PointMutation tree, $T$, and label it as "null";
21: for M $\in MBase$ do
22:     Let $M_1$ be the first single point mutation in $M$ and $M'$ be the
        remaining set of point mutations.
23:     insertTree($M_1, M'$,$T$).
24:     while M' is not empty do
25:         Let $M_1$ be the first single point mutation in $M'$ and $M'$ be the
            remaining set of point mutations.
26:         insertTree($M_1, M'$,$T$).
27:     end while
28:     update the corresponding supporting virus pairs.
29: end for
30:
31: Procedure insertTree($M$, $M'$,$T$)
32: if $T$ doesn't have a child $P$ and $P$.(point mutation) = $M$.(point
    mutation) then
33:     create a new node $P$, with its parent-link linked to $T$
34: end if
```

or not.

Lemma 4.1.1. *The PointMutation tree contains the completely candidate sets of k point mutations.*

Proof: We prove this by induction. When $|MBase| = 1$, the set of k point mutations corresponds to the single path from the root node. Suppose we do not miss any candidate set of k point mutations when $|MBase = \{M_1, M_2, \cdots, M_n\}| = n$. Now, for $Mbase = \{M_1, M_2, \cdots, M_n, M_{n+1}\}$, we have the following three cases:

1. M_{n+1} is the same as one of the M_i, $1 \geq i \geq n$. In this case, no additional nodes are created in the PointMutation tree, we only need to update the supporting virus pairs. The set of k point mutations remains the same.

2. There exists a continuous sub k point mutations M' between M_{n+1} and one of the M_i, $1 \geq i \geq n$. If M' corresponded to a direct child of the root node, we follow the path of this direct child until we come upon the first node that deviates from M'. From this node, we create a new branching path for the remaining mutations of M'. If M' does not correspond to any direct child of the root node, we simply create a separate path corresponding to M_{n+1} and insert it as a direct child of root node. In this manner, all possible mutations from M_{n+1} are incorporated into the PointMutation tree.

Hence, no candidate mutations will be missed. ■

Continuing with our example, the valid point mutation $< 17 : N \rightarrow T >$ has the lowest support value. We observe that $< 17 : N \rightarrow T >$ occurs three times in the PointMutation tree (see Figure 4.3). The paths corresponding to these occurrences are:($< 11 : P \rightarrow S >$-$< 1 : A \rightarrow D >$-$< 17 : N \rightarrow T >$, $< 13 : T \rightarrow Y >$-$< 1 : A \rightarrow D >$-$< 17 : N \rightarrow T >$,$< 13 : T \rightarrow Y >$-$< 11 : P \rightarrow S >$-$< 1 : A \rightarrow D >$-$< 17 : N \rightarrow T >$. We extract the prefix of these three paths to form the conditional mutation base(Table 4.3) for $< 17 : N \to T >$. For each path, its supporting virus pairs are equal to those of corresponding $< 17 : N \to T >$ in the PointMutation tree.

Table 4.3: The $< 17 : N \to T >$'s conditional mutation base

virus pair	k point mutations
(vs_1, vs_2)	$< 13 : T \to Y >$, $< 11 : P \to S >$, $< 1 : A \to D >$
(vs_1, vs_3)	$< 11 : P \to S >$, $< 1 : A \to D >$
(vs_7, vs_8)	$< 13 : T \to Y >$, $< 1 : A \to D >$

With the conditional mutation base, we compute support and significance values of these different point mutations in it to form the conditional statistic table(Table 4.4). These point mutations are $< 1 : A \to D >$, $< 11 : P \to S >$, $< 13 : T \to Y >$, whose suffix sets of k point mutations are all $\{< 17 : N \to T >\}$ and support values are 2, 1, 2 in order. Now, we can calculate the significance values of those point mutations. For example $Significance(< 1 : A \to D >) = \frac{Support(<1:A \to D>)}{max\{|VPair_1(<1:A \to D>)|,|VPair_{17}(<17:N \to T>)|\}} = \frac{2}{max\{3,3\}} \approx 0.67$, the other significance values are 0.2, 0.4 in order.

Table 4.4: The $< 17 : N \to T >$'s conditional statistic table. (min_Support=2 and min_Significance=0.4)

M	$\{(vs_i, vs_j)\}$	$VS[i]$	$VS[j]$	$Support(M)$	$Significance(M)$
$< 1 : A \to D >$	$\{(vs_1, vs_2),(vs_1, vs_3),(vs_7, vs_8)\}$	$\{vs_1, vs_7\}$	$\{vs_2, vs_3, vs_8\}$	2	0.67
$< 11 : P \to S >$	$\{(vs_1, vs_2),(vs_1, vs_3)\}$	$\{vs_1\}$	$\{vs_2, vs_3\}$	1	0.2
$< 13 : T \to Y >$	$\{(vs_1, vs_2),(vs_7, vs_8)\}$	$\{vs_1, vs_7\}$	$\{vs_2, vs_8\}$	2	0.4

Based on $< 17 : N \to T >$'s conditional statistic table, we can remove the invalid point mutation $< 11 : P \to S >$ in the $< 17 : N \to T >$'s conditional mutation base, resulting in three sets of k point mutations: ($< 1 : A \to D >$, $< 1 : A \to D >$-$< 13 : T \to Y >$, $< 1 : A \to D >$-$< 13 : T \to Y >$). The $< 17 : N \to T >$'s conditional PointMutation tree is shown in Figure 4.4. This tree is then mined recursively and the whole process repeats until no new valid mutations are found.

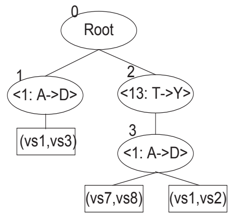


Figure 4.4: $< 17 : N \rightarrow T >$'s conditional PointMutation tree

Algorithm 2 gives the details of the recursive mining process.Line 8 calls the recursive procedure ptMutationTree-Miner. Line 11 starts the loop with mutation M_i that has the lowest support value. Line 12 constructs M_i's conditional mutation base. Given $min\_Support$ and $min\_Significance$, Line 13 computes M_i's conditional statistic table. In Line 14, the invalid point mutations are removed. Line 15 constructs the PointMutation tree corresponding to this mutation base by calling $PointMutation\ tree_{M_i}$. Line 16 determines whether $PointMutation\ tree_{M_i}$ is null or not. If it is not null, then, line 17 links this point mutation M_i with its suffix set of k point mutations M to form new suffix set of k point mutations M' for $PointMutation\ tree_{M_i}$. Line 18 calls the procedure ptMutationTree-Miner to recursively increase the k value of the valid sets of k point mutations. Line 21 sums up the complete sets of *valid k point mutations*.

4.2 Mining the mutation Chain

With the valid sets of k point mutations discovered, the next step is to extend them to form valid mutation chains. We observe that certain sequence pairs cannot form valid mutation chains if they do not form any valid set of k point mutations in the previous step. Hence, we introduce another pruning strategy *valid Mutation Space*: for each sequence pair (vs,vs') in $vPSD$, where $vs' \in NB(vs)$, if the pair does not support any valid set of k point mutations. This means that there is no probability that vs could mutate to vs'. Thus, we can remove vs' from the $NB(vs)$. This reduces the search

Algorithm 2: ptMutationTree-Miner

```
1: input:
2: $PointMutation\ tree$: the PointMutation tree of vPSD;
3: $min\_Support$: the minimal support;
4: $min\_Significance$: the minimal mutation significance;
5: output:
6: The completely valid sets of k point mutation
7: method:
8: call ptMutationTree-Miner(PointMutation tree,null);
9:
10: Procedure ptMutationTree-Miner($Tree$,$M$(suffix set of k point
    mutations))
11: for all $M_i \in Tree$ (start from $M_i$ that is the point mutation with
    lowest support value in the $Tree$) do
12:   construct $M_i$'s conditional mutation base;
13:   construct $M_i$'s conditional statistic table based on the $min\_Support$
      and $min\_Significance$;
14:   remove the invalid point mutations in $M_i$'s conditional mutation base;
15:   construct $M_i$'s conditional $PointMutation\ tree_{M_i}$;
16:   if $PointMutation\ tree_{M_i} \neq \emptyset$ then
17:     $M' = M_i \bigcup M$;
18:     call ptMutationTree-Miner ($PointMutation\ tree_{M_i}$,$M'$);
19:   end if
20: end for
21: return($the\ completely\ valid\ sets\ of\ k\ point\ mutations$);
```

space for the generation of mutation chain.

A sequence chain $vs_1 \rightarrow \ldots \rightarrow vs_{T-1}$ will join with a sequence vs' to support a new mutation chain of length T if and only if

1. $vs_1 \rightarrow \ldots \rightarrow vs_{T-1}$ supports a valid mutation chain $M_1 \rightarrow \ldots M_{T-2}$ with length $T-1$;
2. (vs_{T-1}, vs') supports a valid set of k point mutations M', $vs' \in NB(vs_{T-1})$;
3. M_{T-2} and M' share a common subset of local hot positions Pos where $|Pos| \geq min\_k$.

Therefore, there is one important operation as a new pruning strategy: *increment join*, denoted as $\bowtie_{min\_k}$:

$$\mathcal{SF}^T = \mathcal{SF}^{T-1} \bowtie_{min\_k} vs'$$

,where $\mathcal{SF}^{T-1}$ is the set of sequence chains with length $T-1$, $T \geq 3$.

Algorithm 3 shows the ptMutationChain-Miner framework to mine the complete set of valid mutation chains. Line 11 generates the sequence pairs. Line 12 finds the local hot positions for each sequence. Line 13 generates the completely valid sets of k point mutations through the ptMutationTree-miner algorithm. Line 14 performs the valid mutation space pruning strategy and resets the neighborhood relationships in the $vPSD$, $\mathcal{SF}^2$. Line 15-17 find all the valid mutation chains. Line 18 returns the complete set of valid mutation chains of $vPSD$.

Algorithm 3: ptMutationChain-Miner

```
input:
$vPSD$: influenza A virus protein sequence dataset;
$Localhot$: the threshold value for local hot positions;
$min\_Support$: the minimal support;
$min\_Significance$: the minimal mutation significance;
$min\_k$: the minimal $|Pos|$;
$min\_L$: the minimal mutation chain length;
output:
the complete set of valid mutation chains;

$\mathcal{SF}^2$: the set of sequence pairs ($vs_i$,$vs_j$) that satisfies $vs_j \in NB(vs_i)$;
perform local hot position pruning strategy;
generate the completely valid sets of k point mutations;
perform valid mutation space pruning strategy and reset $\mathcal{SF}^2$;
for all $\mathcal{SF}_i^2 \in \mathcal{SF}^2$ do
  call ChainMiner($\mathcal{SF}_i^2$);
end for
return($the\ complete\ set\ of\ valid\ mutation\ chains$);

Procedure ChainMiner($\mathcal{SF}^{T-1}$)
$vs_i$ is the last sequence of $\mathcal{SF}^{T-1}$;
for all $vs_j \in$ NB($vs_i$) do
  $\mathcal{SF}^T = \mathcal{SF}^{T-1} \bowtie_{min\_k} vs_j$;
  if $\mathcal{SF}^T \neq \emptyset$ then
    call ChainMiner($\mathcal{SF}^T$);
  end if
end for
```

Chapter 5

Performance Study

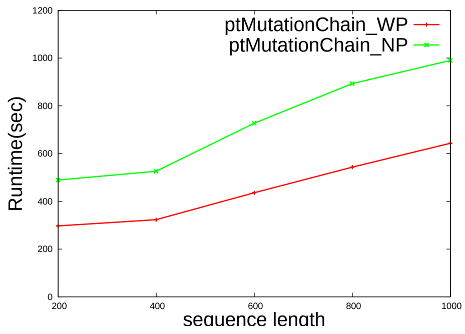
In this section, we report the results of our mining algorithms on both synthetic and real world datasets. All the algorithms are implemented in C++ and the experiments are carried out on a server with Quad Intel 2.83GHZ processors and 3GB memory, running Windows XP.

5.1 Experiments on Synthetic Datasets

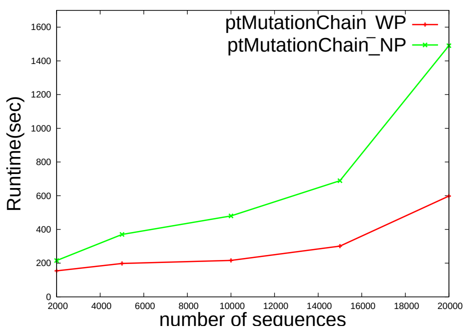
We modify the data generator in [42] to generate the synthetic datasets composed of those sequences with two parameters (location and time): K is the length of virus sequences, D is the total sequence number in the Synthetic Datasets and L is the length of the mutation chain. The spatial-temporal dimensions are set to $1000 \times 1000 \times 1200$, the alphabet size $|\Sigma|$ is set to 21.

In order to test the effectiveness of the pruning strategies (local hot positions, valid Mutation Space and increament join) and show the scalabilty of algorithm ptMutationChain-Miner. We introduce two variants: $ptMutationChain\_WP$, ptMutationChain-Miner with pruning strategies; and $ptMutationChain\_NP$, ptMutationChain-Miner without any pruning strategy. The second one needs to take into account all the positions in any sequence and join all instances to obtain instance chains. Then, we vary the virus sequence length K, sequence number D and mutation chain length L

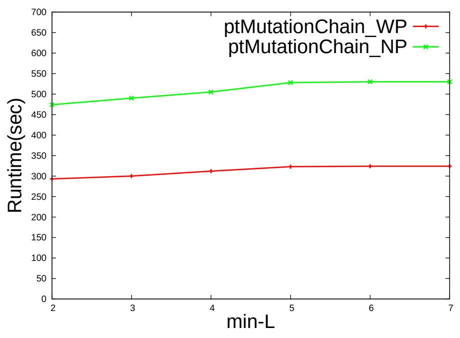
respectively to finish three comparative studies.



(a) Effect of sequence length



(b) Effect of database size



(c) Effect of mutation chain length min_L

Figure 5.1: Comparative study on effect of pruning techniques

To fix the sequence length $K = 100$, set min_k=4, and integrate three mutation chains of length L (4≤L≤14 and 2≤ $|Pos|$ ≤5) into the testing datasets, we can get the first result in Figure 5.1(a) by varying sequence length $|K|$ from 200 to 1000 and fixing $|D| = 5$k, and the second result in Figure 5.1(b) by varying $|D|$ from 2k to 20k and fixing $|K| = 100$. Both the two results show that $ptMutationChain\_WP$ is much faster than $ptMutationChain\_NP$, because our pruning strategies can prune more and more positions of each sequence without the probability to form any valid set of k point mutations and reduce more and more sequence chains without the probability to support any valid mutation chain.

Finally, we get the third result in Figure 5.1(c) by varying the minimal mutation chain length *min_L*. In this experiment, the size of dataset is 15k and the length of each sequence is 100, and we integrate three mutation chains (2≤L≤5 and 5≤ $|Pos|$ ≤7). We see that *ptMutationChain_WP* is still faster than *ptMutationChain_NP* and both them are slowly increasing, because constructing longer mutation chain needs more time. When larger than length 5, like 6 and 7, their runtime are all the same, because the maximal mutation chain length in the testing dataset is 5 and program will be terminated after the length of chain reached 5.

5.2 Experiments on Influenza A Virus Dataset

Two common mechanisms the influenza virus uses to escape detection by the host immune system is changing antigens by antigenic shift or drift. Antigenic shift is the process by which two or more different strains of the virus combine to form a new subtype having a mixture of the surface antigens of the parent strains, while antigenic drift refers to incremental accumulation of mutations on the viral proteins over time resulting in changes in their antigenic makeup. We next apply our algorithm ptMutationChain-Miner to our dataset [8] of 40326 influenza A virus sequences to detect mutations that may be indicative of antigenic drift and shift events.

The influenza A virus protein dataset is composed of 11 influenza A virus proteins; and for each virus sequence record, there are all relative informations, like subtype (e.g., H1N1,H3N2,H5N1), host (e.g., swine, avian,human), country and year of isolation(e.g., Table 1.1). Next, we use MUSCLE 3.6 [13] to finish the multiple sequence alignments of such 11 proteins. According to the suggestions by our cooperators in bioinformatics, because that the viruses spread and mutate gradually, instead of sudden changes and promulgation, one protein sequence vs is likely to mutate to another sequence vs' if vs' occurs within two years after the occurrence vs, and the geographical distance between them is less than 1,000 kilometers. In addition, those cooperators also help us to set the reasonable values for *Localhot*, *min_Support* and

$min\_Significance$ are 0.5, 2 and 0.01 respectively.

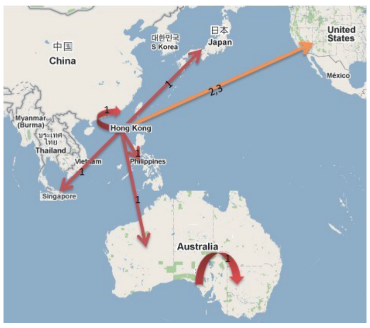
Based those experiments on Influenza A Virus Dataset, we successfully find some interesting mutation patterns which can reflect those famous pandemic influenzas in human history.

Hong Kong flu (H3N2) outbreak (1968-69)

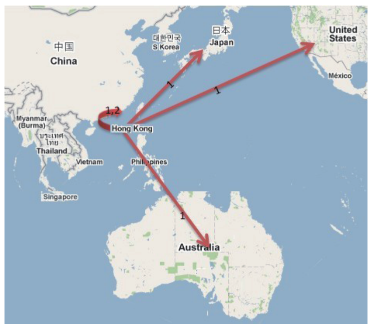
The hemagglutinin (HA) and neuraminidase (NA) glycoproteins of influenza A viruses comprise the major surface proteins and the main immunizing antigens of the virus. HA is responsible for virion entry into host epithelial cells while NA assists in the elution of virion progeny from the infected cell. Neutralization of the virus is mediated through the HA, which is hence subject to strong selective pressure by the host immune system as new strains emerge to produce new epidemics [12]. We examined the spatio-temporal spread patterns of the Pandemic Hong Kong flu (H3N2) between 1968 and 1969. The first incidence of disease was reported in Hong Kong in 1968, which subsequently spread worldwide in the following two winters. Two sets of 2 point mutations, i) $\{< 136 : P \rightarrow H >, < 57 : N \rightarrow S >\}$ in NA protein and ii) $\{< 250 : W \rightarrow G >, < 542 : N \rightarrow T >\}$ in HA protein, were identified that could reflect the overall transmission route of the Pandemic Hong Kong flu (H3N2) between 1968 and 1969, including the virus entry into the distant California from returning Vietnam War troops (Figure 5.2). During the period of 1968, we also found evidence of the virus evolving within Hong Kong and Australia.

H5N1 pandemic (2003)

We next applied our algorithm to analyze the spatio-temporal spread patterns of the pandemic influenza (H5N1) in 2003. Previous studies had demonstrated that the virulence of a highly pathogenic H5N1 virus might correlate with polymerase activities [39], and play a central role in adaptive mutations and potential reassortment [35]. For the 2003 H5N1 pandemic, two mutation events were found in the polymerase proteins, which could biologically reflect

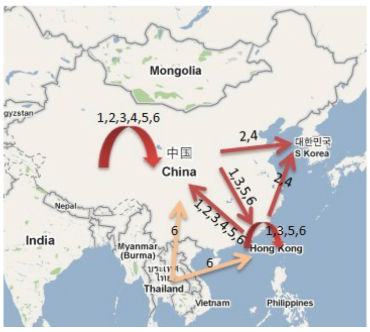


(a) NA: $\{< 136 : P \rightarrow H >, < 57 : N \rightarrow S >\}$

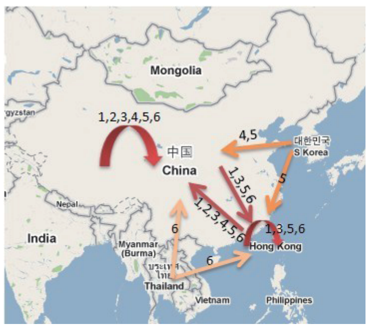


(b) HA: $\{< 250 : W \rightarrow G >, < 542 : N \rightarrow T >\}$

Figure 5.2: Proposed geographical spread of the Pandemic Hong Kong flu (H3N2) between 1968 and 1969 (1: 1968; 2: 1968-69; 3: 1969)

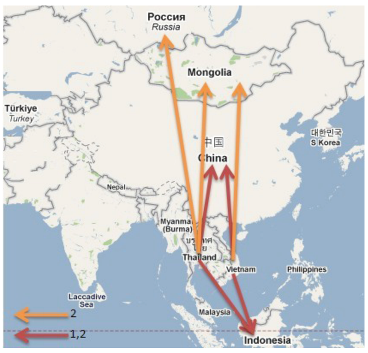


(a) PA: $\{< 128 : T \rightarrow I >, < 203 : K \rightarrow R >\}$

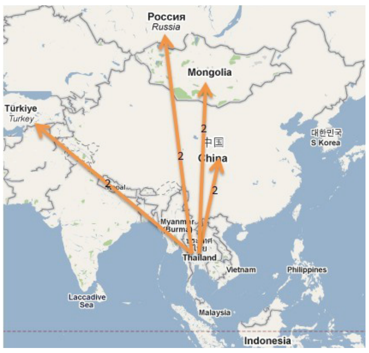


(b) PB1: $\{< 385 : K \rightarrow R >, < 383 : L \rightarrow S >, < 13 : V \rightarrow A >\}$

Figure 5.3: Proposed geographical spread of the Pandemic influenza (H5N1) in 2003 (1: 2002; 2: 2002-03; 3: 2002-04; 4: 2003; 5: 2003-04; 6: 2004)



(a) M2: $\{<30: N \rightarrow S>, <25: I \rightarrow L>, <63: A \rightarrow S>\}$



(b) HA: $\{<225: K \rightarrow R>, <114: P \rightarrow I>\}$

Figure 5.4: Proposed geographical spread of the Pandemic influenza (H5N1) in 2005 (1: 2004; 2: 2005)

the transmission route of 2003 H5N1 pandemic: i) a set of 2 point mutations pattern $\{<128: T \rightarrow I>, <203: K \rightarrow R>\}$ in the polymerase acidic (PA) protein, and ii) a set of 3 point mutations pattern $\{<385: K \rightarrow R>, <383: L \rightarrow S>, <13: V \rightarrow A>\}$ in the polymerase basic 1 (PB1) gene segment. The spatio-temporal spread patterns of the two mutation events (Figure 5.3) revolved around four countries in Asia: China, Hong Kong, Thailand and Korea. The sets of k point mutations $\{<128: T \rightarrow I>, <203: K \rightarrow R>\}$ and $\{<385: K \rightarrow R>, <383: L \rightarrow S>, <13: V \rightarrow A>\}$ first reported in China and Hong Kong respectively, undergone mutations within the respective countries, and spread outwards to neighboring countries including South Korea and Thailand.

H5N1 pandemic (2005)

Two mutation events were identified (Figure 5.4), which could possibly reflect the transmission route of H5N1 pandemic in 2005. They include i) a set of 3 point mutations pattern $\{<30: N \rightarrow S>, <25: I \rightarrow L>, <63: A \rightarrow S>\}$ in the matrix 2 (M2) protein and ii) a set of 2 point mutations pattern $\{<225: K \rightarrow R>, <114: P \rightarrow I>\}$ in HA. Based on these mutation patterns, it could be seen that the H5N1 pandemic influenza mainly occurred in Asia and Europe during 2005. Thailand and Vietnam were the primary source of the new strain, which spread rapidly to the surrounding countries of China and Indonesia, and subsequently to Turkey, Mongolia and Russia.

Chapter 6

Conclusion and Future Work

In this thesis, we have proposed a framework for discovering mutation chains, which are mostly non-continuous and take into account the 3D-structure of the virus protein. We introduced the neighborhood of each sequence to capture its mutation likelihood. We proposed an integrated algorithm ptMutationChain-miner to mine mutation chains utilizes pruning strategies to reduce the search space. Experiments on synthetic datasets showed that our pruning strategies are effective. Experiments on the real world Influenza A virus dataset revealed meaningful mutation patterns that correspond to some episodes of influenza outbreak in human history. Our method is expected to provide an generally effective tool in the fight against emerging and re-emerging infectious diseases with rapid mutations and transmissions.

In our future work, we plan to extend the mutation chains to find positions that always co-mutate for each virus subtype taking into account the spatial and temporal variations. Such positions are often a strong indication of the function sites. This will allow us to predict the function sites of virus subtype.

Bibliography

[1] E. Omiecinski A. Savasere and S. Navathe. Mining for strong negative associations in a large database of customer transactions. *IEEE Data Eng. Conf.*, Feb,1998.

[2] R. Agarwal, C. Aggarwal, and V. Prasad. A tree projection algorithm for generation of frequent itemsets. *Parrallel and Distributed Computing (Special Issue on High Performance Data Mining)*, 2000.

[3] C. Aggarwal and P. Yu. A new framework for itemset generation. *the 17th Symposium on Principles of Database Systems*, pages 18–24, June,1998.

[4] R. Agrawal, T. Imielinski, and A. Swami. Mining association ruls between sets of items in large database. *1993 ACM-SIGMOD Int. Conf. on Management of Data*, pages 207–216, May,1993.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, page 3, Los Alamitos, CA, USA, 1995. IEEE Computer Society.

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. *20th Intl Conf. Very Large Data Bases*, Aug,1994.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. pages 487–499, 1994.

[8] Y. Bao, P. Bolotov, D. Dernovoy, B. Kiryutin, L. Zaslavsky, T. Tatusova, J. Ostell, and D. Lipman. The influenza virus resource at the national center for biotechnology information. *J. Virol.*, 82(2):596–601, 2008.

[9] C. Bettini, X.S. Wang, and S. Jajodia. Mining temporal relationships with multiple granularities in time sequences. *Data Engineering Bulletin*, 21:32–38, 1998.

[10] S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. *SIGMOD*, May,1997.

[11] H. Cao, D.W. Cheung, and N. Mamoulis. Discovering partial periodic patterns in discrete data sequences. *Eighth PacificAsia Conf. Knowledge Discovery and DataMining (PAKDD04)*, 2004.

[12] Kilbourne ED, Johansson BE, and Grajower B. Proc natl acad sci usa 1990. 87(786-790).

[13] R. C. Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 32(5):1792–1797, 2004.

[14] M. Garofalakis, R. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. *1999 Int. Conf. Very Large Data Bases(VLDB99)*, pages 223–234, Sept.1999.

[15] N. Mamoulis H. Cao and D.W. Cheung. Mining frequent spatio-temporal sequential patterns. *Fifth IEEE Intl Conf. Data Mining (ICDM05)*, pages 82–89, 2005.

[16] J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. *1999 Int. Conf. Data Engineering (ICDE99)*, pages 106–115, Apr.1999.

[17] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. *Very Large Databases Conf.*, pages 420–431, Sept,1995.

[18] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. *2000 Int. Conf. Knowledge Discovery and Data Mining (KDD00)*, pages 355–359, Aug. 2000.

[19] JIAWEI HAN, JIAN PEI, and YIWEN YIN. Mining frequent patterns without candidate generation. *the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, 2000.

[20] C. Hidber. Online association rule mining. *ACM-SIGMOD Conf. Management of Data*, pages 145–156, June,1999.

[21] M. Houtsma and A. Swami. Set-oriented mining of association rules. *Intl Conf. Data Eng.*, Mar,1995.

[22] Meng Hu, Jiong Yang, and Wei Su. Permu-pattern: discovery of mutable permutation patterns with proximity constraint. In *KDD '08*, pages 318–326, New York, NY, USA, 2008. ACM.

[23] Y. Huang, S. Shekhar, and H. Xiong. Discovering colocation patterns from spatial datasets: A general approach. *IEEE Trans. Knowledge and Data Eng.*, 16(12), Dec,2004.

[24] Yan Huang, Liqin Zhang, and Pusheng Zhang. A framework for mining sequential patterns from spatio-temporal event data sets. *IEEE Trans. on Knowl. and Data Eng.*, 20(4):433–448, 2008.

[25] I. Jonassen, J.F. Collins, and D.G. Higgins. Finding flexible patterns in unaligned protein sequences. *Protein Sci.*, 4:1587–1595, 1995.

[26] AK. Kashyap, J. Steel, AF. Oner, and MA. Dillon. Combinatorial antibody libraries from survivors of the turkish h5n1 avian influenza outbreak reveal virus neutralization strategies. *Proc Natl Acad Sci U S A*, 105(598), 2008.

[27] M. Klemettinen, H. Mannila, P. Ronkainen, T. Toivonen, and A. Verkamo. Fingding interesting rules from large sets of discovered association rules. *the 3rd int'lConf. on Information and Knowledge management(CIKM'94)*, pages 401–407, Nov,1994.

[28] B. Liu, W. Hus, and Y. Ma. Pruning and summarizing the discovered associations. *the Fifth Int'l Conference on Knowledge Discovery and Data Mining*, pages 125–134, 1999.

[29] H. Lu, J. Han, and L. Feng. Stock movement and ndimensional inter-transaction association rules. *1998 SIGMOD Workshop Research Issues on Data Mining and Knowledge Discovery (DMKD98)*, pages 12:1–12:7, June 1998.

[30] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D.W.L. Cheung. Mining, indexing, and querying historical spatiotemporal data. *10th ACM SIGKDD*, 2004.

[31] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[32] H. Mannila, H. Toivonen, and A.I. Verkamo. Efficient algorithms for discovering association rules. *Knowledge Discovery and Data Mining 94: AAAI Workshop Knowledge Discovery in Databases*, pages 181–192, July,1994.

[33] A.F. Neuwal and P. Green. Detecting patterns in protein sequences. *J. Mol. Biol.*, 239:698–712, 1994.

[34] Edward R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 15(1), 2003.

[35] Li OT, Chan MC, Leung CS, Chan RW, Guan Y, Nicholls JM, and Poon LL. Full factorial analysis of mammalian and avian influenza polymerase subunits suggests a role of an efficient polymerase for virus adaptation. *PLoS One*, 4(5)(e5658), 2009 May 21.

[36] B. Özden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. *1998 Int. Conf. Data Engineering(ICDE98)*, pages 412–421, Feb. 1998.

[37] J.S. Park, M-S. Chen, and P.S. Yu. An effective hash based algorithm for mining association rules. *ACM-SIGMOD Conf. Management of Data*, pages 229–248, May,1995.

[38] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. *2001 Int. Conf. Data Engineering(ICDE'01)*, pages 215–224, April. 2001.

[39] Salomon R, Franks J, Govorkova EA, Ilyushina NA, and Yen HL. The polymerase complex genes contribute to the high virulence of the human h5n1 influenza virus isolate a/vietnam/1203/04. *J Exp Med*, 203(689-697), 2006.

[40] M.F. Sagot and A. Viari. A double combinatorial approach to discovering patterns in biological sequences. *Symposium on Combinatorial Pattern Matching*, pages 186–208, 1996.

[41] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules. *Very Large Databases Conf.*, pages 432–444, Sept,1995.

[42] C. Sheng, W. Hsu, M.-L. Lee, J. C. Tong, and S.-K. Ng. Mining mutation chains in biological sequences. *Proceedings of the 26th International Conference on Data Engineering.*, pages 473–484, 2010.

[43] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery system. *IEEE Transctions on Knowledge and Data Eng.*, pages 8(6):970–974, 1996.

[44] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, pages 2(1):39–68, 1998.

[45] R. Srikant and R. Agrawal. Mining sequential patterns:generalizations and performance improvements. *5th Int. Conf. Extending Database Technology (EDBT96)*, pages 3–17, Mar. 1996.

[46] R. Srikant and R. Agrawal. Mining sequential patterns:generalizations and performance improvements. In *EDBT*, page 3C17, Avignon, France, Mar.1996. 5th Int. Conf. Extending Database Technology.

[47] R. Srikant and R. Agrawal. Mining generalized association rules. *Very Large Databases Conf.*, pages 407–419, Sept,1995.

[48] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. *SIGKDD 02 Edmonton*, 2002.

[49] J.T.L. Wang, G.W.chirn, T.G. Marr, B. Shapiro, D.Shasha, and K. Zhang. Combinatorial pattern discovery for scientic data: some preliminary results. *SIGMOD*, 1994.

[50] Ke Wang, Yabo Xu, and Jeffrey Xu Yu. Scalable sequential pattern mining for biological sequences. In *CIKM*, pages 178–187, New York, NY, USA, 2004. ACM.

[51] R. G. Webster, W. J. Bean, O. T. Gorman, T. M. Chambers, and Y. Kawaoka. Evolution and ecology of influenza a viruses. *Microbiological reviews.*, pages 152–179, 1992.

[52] Guang Wu. Prediction of mutations in h5n1 hemagglutinins from influenza a virus. *Protein and Peptide Letters*, 13:971–976(6), October 2006.

[53] M. Zaki. Generating non-redundant association rules. *2000 ACM Knowledge Discovery and Data Mining Conf.*, pages 34–43, 2000.

[54] Mohammed J. Zaki. Spade: an efficient algorithm for mining frequent sequences. In *Machine Learning Journal, special issue on Unsupervised Learning*, pages 31–60, 2001.