The Apriori Property of Sequence Pattern Mining with Wildcard Gaps

Fan Min  
School of Computer Science and Engineering,  
University of Electronic Science and Technology of China, Chengdu 610054, China.  
Email: minfan@uestc.edu.cn.

Youxi Wu  
School of Computer Science,  
Hebei University of Technology,  
Tianjin 300130, China.  
Email: wuc@scse.hebut.edu.cn

Xindong Wu  
Department of Computer Science,  
University of Vermont,  
Burlington, Vermont 05405, USA.  
Email: xwu@cems.uvm.edu

Abstract—In biological sequence analysis, long and frequently occurring patterns tend to be interesting. Data miners designed pattern growth algorithms to obtain frequent patterns with periodical wildcard gaps, where the pattern frequency is defined as the number of pattern occurrences divided by the number of offset sequences. However, the existing definition set does not facilitate further research works. First, some extremely frequent patterns are obviously uninteresting. Second, the Apriori property does not hold; consequently, state-of-the-art algorithms are all Apriori-like and rather complex. In this paper, we propose an alternative definition of the number of offset sequences by adding a number of dummy characters at the tail of sequence. With the new definition, these uninteresting patterns are no longer frequent, and the Apriori property holds, hence our Apriori algorithm can mine all frequent patterns with minimal endeavor. Moreover, the computation of the number of offset sequences becomes straightforward. Experiments with a DNA sequence indicate 1) the pattern frequencies under two definition sets have little difference, therefore it is reasonable to replace the existing one with the new one in practice, and 2) our algorithm runs less rounds than the best case of MMP which is based on the existing definition set.

Index Terms—Sequence pattern mining, wildcard gap, frequency, Apriori.

I. INTRODUCTION

Mining interesting patterns from sequences is fundamental in many areas, particularly those related to bioinformatics [1][2]. The concept sequence, or string, is easy to explain. For example, DNA sequences use an alphabet \{A, C, G, T\} representing the four nitrogenous bases Adenine, Cytosine, Guanine and Thymine [2]. The Homo Sapiens (human) DNA sequence AX829174 [3] starts with TTCCTCCGCGA and contains 10,011 characters.

There are three main tasks in the pattern mining scenario. First, the specification of a pattern should be provided, either by a biologist or by a computer scientist. Generally, a pattern is a subsequence with certain constraints. In its original form, a pattern is simply a subsequence. For the sequence fragment mentioned earlier, pattern TC occurs twice, starting from indices 2 and 5, respectively. We say that pattern TC has 2 matches in the sequence. A wildcard \(\phi\) (also called motif [4] or don’t care [5]) matches any character in the alphabet. \(\text{T}\phi\text{C}\) has 3 matches, starting from indices 1, 2 and 5, respectively. A wildcard gap \([N, M]\) matches any subsequence with \(N\) to \(M\) characters, and \(\text{T}[1, 2]\phi\text{C}\) has 4 matches, 2 of which starting from index 1. Patterns with wildcard gaps favor biologists in that they are adaptive to the phase shift of biological sequences.

Second, potentially useful patterns should be identified. This is often undertaken by efficient data mining algorithms developed by computer scientists. The searching space of patterns is typically exponential with respect to the pattern length and the alphabet size, and enumeration algorithms are infeasible. Heuristic approaches are often employed [2][6].

Third, the functionality of a pattern should be explored. This is often the task of biologists. However, computer scientists could be involved again if some other sequences are investigated for the comparison purpose. It is out of the scope of this paper.

This paper focuses on the first issue, and the second issue is straightforward after the first is clearly addressed. We consider the problem addressed by Zhang et. al [2] and Zhu et. al [6]; that is, mine all patterns with a frequency no less than a user specified threshold \(\rho\) from a sequence. One sequence is considered at a time, and this is the major difference from the problem addressed in some other works (see, e.g., [7][8]). Moreover, patterns contain periodical wildcard gaps, where periodical indicates that wildcard gaps between any two adjacent characters are identical [9]. For example, \(\text{T}[1, 2]\phi\text{A}[1, 2]\phi\text{A}\) is such a pattern while \(\text{T}[1, 2]\phi\text{A}[2, 3]\phi\text{A}\) is not. This requirement reflects the phenomenon of base pair oscillation [10] with a clear biology meaning.

Above mentioned works among others have helped biologists mining many interesting patterns [2]. However, there are still some drawbacks of those works. Specifically, according to the existing definition set [2][6], we can construct some extremely frequent patterns, which are obviously uninteresting from the viewpoint of a biologist. This does not comply a widely accepted hypothesis that more frequent patterns tend to be more interesting. Although we can avoid this kind of patterns in practice, in theory we cannot.

The Apriori property is very important in associate analysis [11] and sequence pattern mining [12]. In this context, it can be stated as “any super-pattern of an infrequent pattern should be infrequent”, or more precisely, “let \(P\) be an super-pattern of \(P\), then the frequency of \(P\) should be no more than that
of \( P \). If the pattern frequency is defined as the number of pattern occurrences and wildcard gaps are not considered, this property holds. For example, \( P = T \) appears 3 times in \( S \), then \( P^* = TC \) can never appear more than 3 times in \( S \). This property is the basis of many pattern mining algorithms to ensure the completeness of the result.

Unfortunately, the Apriori property does not hold in the works of Zhang et. al [2] and Zhu et. al [6]. That is, the frequency of a pattern may exceed that of a sub-pattern. Hence they proved some Apriori-like properties and proposed respective algorithms. Moreover, to run these algorithms efficiently, the user should have a rough idea about the length of the longest frequent patterns [2]. This may not be easy to the user. Therefore, these algorithms are not neat in practice.

We neither undertake an incremental work to propose a more efficient algorithm, nor deal with some special cases of the problem. Instead, we point out that these two drawbacks mentioned above are due to the definition of the number of offset sequences. We then propose to revise the definition by adding a number of dummy characters at the tail of sequence. Under the new definition set, these uninteresting patterns are no longer frequent; and the Apriori property holds. This gives us a substantial foundation to our Apriori algorithm, which is validated through experiments in a DNA sequence. Since sets satisfying two definition sets are very close. This claim has the following form:

\[
a_1[N, M]a_2[N, M] \ldots [N, M]a_l,
\]

where \( l \) is the length of the pattern.

By periodic we mean that all gaps are identical. This requirement favors biologists and makes our analysis simpler. However, it is not essential from the theoretical viewpoint. Patterns with independent wildcard gaps are discussed in [13][14] in the context of pattern matching.

**Definition 5:** Given a sequence \( S \) with length \( L \), an offset sequence of length \( l \) is a sequence of integers \( C = \langle c_1, c_2, \ldots, c_l \rangle \) such that \( 1 \leq c_j \leq L \) for all \( j \), and \( N \leq c_{j+1} - c_j - 1 \leq M \) for all \( 1 \leq j \leq l - 1 \).

In other words, an offset sequence is a sequence of positions in \( S \) that satisfies the gap requirements. It is where the pattern is compared to see whether or not there is a match.

**Definition 6:** Given a sequence \( S \), a pattern \( P \), and an offset sequence \( C \), we say that \( P \) matches \( S \) w.r.t. \( C \) if \( s_{c_j} = a_j \) for all \( 1 \leq j \leq l \).

The number of matches of \( P \) in \( S \) is denoted by \( sup(P, S) \). The number of offset sequences that \( P \) might be matched in \( S \) is denoted by \( ofs(P, S) \). According to Definitions 4 and 5, a position could be employed multiple times to produce different offset sequences; and a character in the sequence can be employed multiple times to produce different matches.

**Example 1:** Let \( S = ACTTT \) and \( P_1 = A[1, 3]T \). There are six offset sequences, including \( \langle 1, 3 \rangle, \langle 1, 4 \rangle, \langle 1, 5 \rangle, \langle 2, 4 \rangle, \langle 2, 5 \rangle \) and \( \langle 3, 5 \rangle \), hence \( ofs(P_1, S) = 6 \). Three of these offset sequences (i.e., \( \langle 1, 3 \rangle, \langle 1, 4 \rangle \) and \( \langle 1, 5 \rangle \)) produce matches, hence \( sup(P_1, S) = 3 \). These offset sequences and matches are illustrated in Figure 1.

![Fig. 1. Six offset sequences and three matches (marked by bolder lines)](image)

In pattern mining, the goal is to determine frequently occurring patterns, therefore we need to define the term frequency.

**Definition 6:** The frequency of pattern \( P \) in sequence \( S \) is

\[
f(P, S) = \frac{sup(P, S)}{ofofs(P, S)}.
\]

Given a sequence \( S \), a gap requirement \([M, N]\), and a user-specified threshold \( \rho \), \( P \) is called a frequent pattern iff
\( f(P, S) \geq \rho \). Our objective is to obtain the set of all frequent patterns. And longer patterns (i.e., with greater \( l \)) tend to be more interesting.

For the discussion of the Apriori property and corresponding pattern growth algorithms, we need to define sub-pattern and super-pattern.

**Definition 7:** Given \( 1 \leq i \leq j \leq l \), \( P' = a_i[N, M]a_{i+1}[N, M] \ldots [N, M]a_j \) is a **sub-pattern** of \( P = a_1[N, M]a_2[N, M] \ldots [N, M]a_l \). Conversely, \( P \) is a **super-pattern** of \( P' \).

## III. The new definition

An ideal definition set should facilitate further analysis. Unfortunately, the one presented in the last section makes problem analysis and algorithm design rather complex. In this section, we discuss the reason and present a new definition.

### A. Drawbacks of the existing definition set

The computation of \( ofs(P, S) \) is rather complex. According to Zhang et. al [2] and Zhu et. al [6], there are three cases to be considered. For the most likely happening in practice, the following formula is obtained:

\[
ofs(P, S) = [L - (l - 1)\left(\frac{M + N}{2} + 1\right)]W^{l-1}.
\]

While for the most complex case, even a recursive formula is needed. The difficulty of computation should not be the reason to deny a definition. It is given here for further comparison.

The first drawback lies in that some extremely frequent patterns are obviously uninteresting. Let us continue Example 1 and consider the following example.

**Example 2:** Let \( P_2 = A[3, 5]T \). We see that \( sup(P_2) = 1 \) (corresponding to the offset sequence \( (1, 5) \)) and \( ofs(P_2, S) = 1 \) (only \( (1, 5) \)), therefore \( f(P_2, S) = \frac{1}{5} = 1 \).

\( P_1 \) and \( P_2 \) have the same flexibility (i.e., \( 3 - 1 + 1 = 5 - 3 + 1 \)). Intuitively, \( P_2 \) only appears once, hence it is less interesting than \( P_1 \). However, due to Definition 4, \( ofs(P_2, S) \) is only 1, making \( P_2 \) more frequent.

To make the point clearer, we consider the following example.

**Example 3:** For any \( S = s_1s_2 \ldots s_L \), let \( P_3 = s_1[L - 2, L - 2]s_L \), from the viewpoint of a biologist, \( P_3 \) is simply uninteresting. However, \( sup(P_3, S) = 1 \) and \( ofs(P_3, S) = 1 \), therefore \( f(P_3, S) = 1 = 100\% \). That is, \( P_3 \) is extremely frequent. Generally, we expect that more frequent patterns are more interesting. Hence the result computed in Examples 2 and 3 are inappropriate.

The second drawback is that the Apriori property does not hold. The following counter example [2] is a continuation of Example 1.

**Example 4:** \( P_4 = A \) is a sub-pattern of \( P_1 \), \( ofs(P_4, S) = 5 \) (including \( (1) \), \( (2) \), \( (3) \), \( (4) \) and \( (5) \)) and \( sup(P_4) = 1 \) (corresponding to the offset sequence \( (1) \)). Hence, the frequency of \( P_1 \) (i.e., \( \frac{3}{5} = \frac{3}{5} \)) exceeds the frequency of its sub-pattern \( P_4 \) (i.e., \( \frac{1}{5} \)).

Therefore, people addressed different Apriori-like properties [2][6] and developed respective algorithms. These properties are, undoubtedly, not as neat as the Apriori property. Consequently, these algorithms are more complex than the Apriori algorithm.

We argue that these drawbacks are not intrinsic to the problem. Instead, it is due to the problem definition. The definition of \( sup(P, S) \) is appropriate, while the definition of \( ofs(P, S) \) is not.

### B. A new definition of offset sequence

We revise Definition 4 and propose the following definition:

**Definition 8:** Given a sequence \( S \) with length \( l \), an **offset sequence** of length \( l \) is a sequence of integers \( C = \langle c_1, c_2, \ldots, c_l \rangle \) such that \( 1 \leq c_1 \leq L \), and \( N \leq c_j + 1 - c_j - 1 \leq M \) for all \( 1 \leq j \leq l - 1 \).

One can observe that the only difference lies in that in Definition 4, \( c_j \in [1, L] \) is required for all \( j \); while in Definition 8, only \( c_1 \in [1, L] \) is required. We use \( ofs^*(P, S) \) to denote the number of all offset sequences, and \( f^*(P, S) = \frac{sup(P, S)}{ofs^*(P, S)} \) to denote the pattern frequency under the new definition.

This revision is important from the theoretical perspective. There are \( L \) possible positions of \( c_1 \), for each of them, there are \( W \) possible positions of \( c_2 \), and so on. Therefore we have

\[
ofs^*(P, S) = LW^{l-1},
\]

which is much more easier to compute than \( ofs(P, S) \) as indicated by Equation (3).

In other words, we can construct a forest with \( L \) \( W \)-complete trees, all with a depth \( l \). Each path from a root to a leaf corresponds to a valid offset sequence. Let \( l = 3 \), the \( i \)-th tree is depicted in Figure 2.

Consider Example 1 again, now \( ofs^*(P_1, S) = 5 \times 3^1 = 15 \). Figure 3 shows three offset sequences starting from index 5. They do not comply Definition 4, therefore not included in Example 1. Conceptually, we use a dummy character \( X \not\in \Sigma \) to fill blanks after \( S \). In this way, the number of matches remains the same.

Now we continue Examples 2 and 3 to address the first issue. \( f^*(P_2, S) = \frac{sup(P_2, S)}{ofs^*(P_2, S)} = \frac{1}{5} \), and \( f^*(P_3, S) = \frac{sup(P_3, S)}{ofs^*(P_3, S)} = \frac{1}{5} \). The frequencies of \( P_2 \) and \( P_3 \) reflect their interestingness well.

More importantly, the Apriori property holds. This is formally given by the following theorem:
Theorem 1: Let $P'$ be a sub-pattern of $P$,
$$f^*(P, S) \leq f^*(P', S).$$  
\hfill (5)

Proof: We only need to consider the condition where $P'$ contains one less character than $P$. Formally, let $P' = p_1g(N,M)p_2\ldots g(N,M)p_{l-1}$ and $P = p_1g(N,M)p_2\ldots g(N,M)p_{l-1}g(N,M)p_l$. According to Equation (4),
$$ofs*(P, S) = ofs*(P', S)W.$$  
\hfill (6)

On the other hand, each match of $P'$ corresponds to at most $W$ matches of $P$, i.e.,
$$N(P, S) \leq N(P', S)W.$$  
\hfill (7)

Therefore
$$f^*(P, S) = \frac{N(P, S)}{ofs*(P, S)} = \frac{N(P', S)}{ofs*(P', S)} = f^*(P', S).$$  
\hfill (8)

Let us revisit Examples 1, 2 and 4. $f^*(P_1, S) = \frac{3}{5 \times 3} = \frac{1}{5}$, $f^*(P_2, S) = \frac{1}{5 \times 3} = \frac{1}{15}$, and $f^*(P_3, S) = \frac{1}{5}$. While $P_3$ grows, each possible position corresponds with a match of $P_3$, hence $f^*(P_1) = f^*(P_3)$ is natural. $P_3$ is less frequent, therefore less interesting, than $P_1$.

We need to compare results computed according to those different definition sets. According to Definition 8, while $l = 1$, no dummy position is needed; while $l > 1$, they are needed to form more offset sequences. Hence we have
$$\begin{cases} 
ofs^*(P, S) = ofs(P, S), & l = 1; \\
\end{cases}$$  
\hfill (9)

Consequently,
$$\begin{cases} 
f^*(P, S) = f(P, S), & l = 1; \\
\end{cases}$$  
\hfill (10)

In practice, however, results computed according to Definitions 4 and 8 do not vary too much. Since $L \gg l \times N$, the two counting mechanisms are only different at the tail of the sequence, as depicted in Figure 3. The number of additional offset sequences, i.e., $ofs^*(P, S) - ofs(P, S)$ is small compared with $ofs^*(P, S)$. Therefore $f^*(P, S)$ and $f(P, S)$ are very close. This issue will be discussed through an example in Section V-B.

From Equation (10), the following theorem is straightforward.

Theorem 2: Given a gap requirement $g = g[N,M]$ and a frequency threshold $\rho$, let $PT^*(g, \rho, S) = \{P | f^*(P, S) \geq \rho$ and gaps in $P$ are all $g\}$, and $PT^*(g, \rho, S) = \{P | f^*(P, S) \geq \rho$ and gaps in $P$, we have
$$PT^*(g, \rho, S) \subseteq PT^*(g, \rho, S).$$  
\hfill (11)

This theorem shows that given a threshold, we may mine more patterns under the new definition. This issue will be discussed in Section V-C.

ไThe Apriori Algorithm

Let $A_1 \subseteq \{s_0 \ldots s_{l-1}\}$ and $\rho = 1$. The algorithm is straightforward, as depicted in Algorithm 1. Lines 1 through 5 deal with patterns with only one character, lines 6 through 7 deal with longer patterns. Since each pattern constructed in the algorithm is tested to see whether or not it contains a tested pattern. We call it a tested pattern.

Since the algorithm is based on the Apriori property, it is both complete and optimal.

Property 1: The output of Algorithm 1 is complete.

Proof: Suppose that there exists a pattern $P = P_1 = a_1[N,M]a_2[N,M] \ldots a_l$, $f(P) \geq \rho$ and $P \not\in \Pi_l$. Then $P$ should not be tested in Algorithm 1. According to Lines 11 through 18, $P' = P_{l-1} = a_1[N,M]a_2[N,M] \ldots [N,M]a_{l-1}$ should not be tested and $P_{l-1} \not\in \Pi_{l-1}$. For the same reason,
biological sequences, PAIG is more efficient when while using GCS [6]. However, according to experiments in Table II. The number of tested patterns and the number of frequent patterns are listed in Table II. Here MMP (best case) is the algorithm proposed in [2], and the result corresponds with the best case. MPPm is an enhanced version of MPP. We observe that in our algorithm, the pattern stops growing while \( l = 11 \) since the number of frequent patterns is 0. While MPPm and MPP (best case) continues this process until \( l = 14 \).

C. Performance comparison

According to discussions in the last subsection, it is also reasonable to compare the performance between existing algorithms computing \( ofs(P, S) \) with our algorithm.

Let \( \rho = 3 \times 10^{-5} \) [2]. The number of tested patterns and the number of frequent patterns are listed in Table II. Here MMP (best case) is the algorithm proposed in [2], and the result corresponds with the best case. MPPm is an enhanced version of MPP. We observe that in our algorithm, the pattern stops growing while \( l = 11 \) since the number of frequent patterns is 0. While MPPm and MPP (best case) continues this process until \( l = 14 \).

\[
|\Sigma| + |P| |\Sigma|, \quad (12)
\]

where \(| \cdot |\) denotes the cardinality of a set. Those patterns are exactly what we test in the algorithm. Therefore the algorithm is optimal.

According to Equation (4), the complexity of computing \( ofs(P, S) \) is only \( O(1) \), hence the complexity of computing \( f(P, S) \) is determined by that of \( sup(P, S) \). It is \( O(L^2W^2 \log(|W|)) \) while using PAIGS [13] and \( O(LW) \) while using GCS [6]. However, according to experiments in biological sequences, PAIG is more efficient when \( W \) is not big (typically less than 4) [14]. Either way the computation time is polynomial.

Since frequent patterns are relevant, we do not have to compute the frequency of each pattern independently. Instead, we can build a tree for this purpose. However, the focus of this paper is the Definition 8 and the Apriori property, and we will not discuss this issue further.

V. EXPERIMENTS

In this section we show our experimental results. Three experiments are conducted. The first is to show the organization of frequent patterns as a tree. The second is to show that frequencies computed according two definition sets are very close. The third is to compare the performance of our algorithm with MMP.

We use the Homo Sapiens (human) DNA sequence AX829174 downloaded from the National Center for Biotechnology Information website [3]. This sequence was used in [2][13]. The sequence consists of 10,011 characters. To enable results comparable with relative works, we also use the same settings as that of [2], that is \( N = 9, M = 12 \).

A. The frequent pattern tree

Let \( \rho = 3 \times 10^{-2} \), and frequent patterns be organized as a tree, as depicted in Figure 4. All branches except from the root to its sons are marked by the gap requirement. Each path from the root to a node constitutes a frequent pattern. There are \( 4 + 16 + 2 = 22 \) frequent patterns, and the longest patterns are \( A[9, 12]A[9, 12]A \) and \( T[9, 12]T[9, 12]T \).

B. Results comparison of two definition sets

From the theoretical viewpoint, we are essentially dealing with a different problem from that of [2], [6]. However, in practice, the difference incurred by the two definitions is trivial.

We consider patterns only contain Ts. For example, while \( l = 4 \), the pattern is \( T[9, 12]T[9, 12]T[9, 12]T \). Results are listed in Table I.

We observe that \( ofs^*(P, S) > ofs(P, S) \) and \( f^*(P, S) < f(P, S) \) while \( l > 1 \). However, the difference between \( f^*(P, S) \) and \( f(P, S) \) is trivial. As listed in the table, \( \frac{f^*(P, S)}{f(P, S)} - 1 \% < 1.5\% \) while \( l \leq 14 \). This phenomenon shows that it is reasonable to replace the traditional pattern mining problem with ours. That is, to compute \( ofs^*(P, S) \) instead of \( ofs(P, S) \).
Note that while \( l \) is 8 or 9, patterns tested by MPP are less than that of our algorithm. As mentioned earlier, our algorithm is the most efficient one to ensure that all frequent patterns to be mined. Hence the reason lies in that we may find more patterns with the new definition set, as indicated by Theorem 2. To obtain them, more tests are natural.

VI. CONCLUSIONS AND FURTHER WORKS

The definition of a problem in computer science is critical to further research efforts. In this paper, we point out that the existing definition of offset sequences is inappropriate through Examples 2 and 3. Then we propose a new definition, under which the computation of \( ofs(P, S) \) is simple. The most important aspect is that the Apriori property holds, which guarantees the completeness (Property 1) and the efficiency of the Apriori algorithm (Property 2).

However, this work is the very beginning, rather than the end of our research since there is still some important issues. First, the computation of pattern frequency is undertaken by an existing pattern matching algorithm, and the result of a pattern is not employed to compute that of its super-patterns. This issue might be relatively easy to address using the pattern growth tree and either depth-first or width-first approaches.

Second, the gap requirement \([N, M]\) is specified by the expert. In some applications [10], experts have a good idea of this setting; while in others the expert have to try many different settings. It is more convenient for the expert to specify the flexibility \((W = M - N + 1)\), or not to specify this requirement at all.

Third, the frequency threshold \( \rho \) is also specified by the expert. This parameter is hard to decide. If \( \rho \) is relatively large, only short patterns can be mined, as indicated by Figure 4. On the other hand, if \( \rho \) is relatively small, almost all short patterns are frequent, and the searching space is exponential. For example, as indicated in Table II, all patterns with length no more than 6 are frequent, and there is only 1 frequent pattern with length 10. To make the matter worse, even if the expert can specify a rational \( \rho \), the number of frequent patterns is still overwhelming to the expert. We can consider another problem to mine top-\( K \) frequent patterns with a given length \( l \). \( K \) and \( l \) would be easy to specify, and the result might be more interesting to the expert.

VII. ACKNOWLEDGEMENTS

This research is supported in part by the National Natural Science Foundation of China (NSFC) under grant No. 60828005.

This paper has benefited from discussions with Xuhua Gao, Jian Zhou and Zhenyu Pan.

REFERENCES