POST SEQUENTIAL PATTERNS MINING
A New Method for Discovering Structural Patterns

Jing Lu\textsuperscript{1}, Osei Adjei\textsuperscript{2}, Weiru Chen\textsuperscript{1} and Jun Liu\textsuperscript{1}
\textsuperscript{1} School of Computer Science and Technology, Shenyang Institute of Chemical Technology, Shenyang 110142, China. Email: Jing.Lu@luton.ac.uk; Tel: 44-1582-743716
\textsuperscript{2} Department of Computing and Information Systems, University of Luton, Park Sq. Luton, LUT 3JU, UK

Abstract: In this paper we present a novel data mining technique, known as Post Sequential Patterns Mining, which can be used to discover Structural Patterns. A Structural Pattern is a new pattern, which is composed of sequential patterns, branch patterns or iterative patterns. Sequential patterns mining plays an essential role in many areas and substantial research has been conducted on their analysis and applications. In our previous work \cite{12}, we used a simple but efficient Sequential Patterns Graph (SPG) to model the sequential patterns. The task to discover hidden Structural Pattern is based on our previous work and sequential patterns mining, conveniently named Post Sequential Patterns Mining. In this paper, in addition to stating this new mining problem, we define patterns such as branch pattern, iterative pattern, structural pattern, and concentrate on finding concurrent branch pattern. Concurrent branch pattern is thus one of the main forms of structural pattern and will play an important role in event-based data modelling.

Key words: Post Sequential Patterns Mining, Sequential Patterns Graph, Structural Pattern, Concurrent Branch Patterns

1. INTRODUCTION

Sequential pattern mining proposed by Agrawal and Srikant \cite{1} is an important data mining task and with broad applications that include the analysis of customer behaviors, web access patterns, process analysis of scientific experiments, prediction of natural disasters, treatments, drug testing and DNA analysis etc. Over the last few years considerable attention has been focused on the achievement of better performance in sequential
pattern mining [1, 5, 10, 14, 15, 16], but there is still the need to do further work in order to improve on results achieved so far. Questions that are usually asked with respect to sequential pattern mining are: What is the inherent relation among sequential patterns? Is there a general representation of sequential patterns? Based on these questions, we proposed a novel framework for sequential patterns called Sequential Pattern Graph (SPG) as a model to represent relations among sequential patterns [12].

From our previous work on SPG and sequential patterns mining, other new patterns such as branch pattern, iterative pattern or structural pattern could be discovered. These patterns were first proposed in [11]. In order to find such patterns, we present a new mining technique known as Post Sequential Patterns Mining. Structural Pattern is the combination of sequential pattern, branch pattern and iterative pattern. Discovering Structural Pattern is the ultimate goal of the Post Sequential Patterns mining task. In this paper, we focus on concurrent branch pattern and its mining algorithms.

The organization of this paper is as follows: We introduce the concept of branch pattern, iterative pattern, and structural pattern and Post Sequential Pattern Mining problem in section 2. In section 3, we present concurrent branch pattern mining. The algorithms used in this approach are also outlined in this section. Section 4 reviews some related work whilst section 5 concludes the study and identifies further work.

1.1 Structural patterns and Post Sequential Patterns Mining Problem Statement

Before introducing the Post Sequential Patterns Mining problem, we formally define some new patterns.

1.2 Structural pattern

All the definitions of terminologies in relation to sequential patterns mining presented in [1] are followed in this paper. Recall that the support of sequence \( s \), denoted by \( s.\text{sup} \), is the number of data sequences containing \( s \) divided by the total number of data sequences in DB. The minsup is the user specified minimum support threshold. A sequence \( s \) is a frequent sequence, or called sequential pattern, if \( s.\text{sup} \geq \text{minsup} \). The aim of sequential pattern mining is to discover all the sequential patterns or maximal sequence.

Based on these definitions, some new concepts are introduced first. This introduction is necessary for the understanding of Post Sequential Patterns Mining, which will be proposed in the next subsection.

**Definition 1 (Mono-pattern)** A sequential pattern that contains only one element is called a mono-pattern and it is denoted by \(<\text{itemset}>\) or \(<\text{item}>\).
A mono-pattern is a frequent itemset of the result of association rule mining. Patterns \(<a>, <(a, b)\) and \<(b, c)\> are all examples of mono-patterns.

**Definition 2 (Sub-pattern)** A sub-pattern is part of a pattern that includes some elements and their sequence order relations.

The pattern \(<cd>\) is an example of a sub-pattern of the patterns \(<abcd>, <acbd>, <cabd>, \) and \(<acbde>\).

**Definition 3 (Branch Pattern)** A branch pattern is a combination of some sequential patterns if and only if those patterns have the same prefix and/or postfix, and the support of the combination is greater than or equal to \(\text{minsup}\). Notation \([s_i, s_j]\) denotes that two sub-patterns \(s_i\) and \(s_j\) appear in different branches pattern. Sub-patterns \(s_i, s_j\) are called the branches of a branch pattern.

For example, sequential patterns \(<eab>\) and \(<efcb>\) have the same prefix \(<e>\) and the same postfix \(<cd>\). If the support of those two patterns occur in the same customer sequence is above the \(\text{minsup}\), then they can constitute a new branch pattern and denoted by \(<e[a,f]cd>\).

Note that in branch pattern such as \(<a[b,c]d>\), \(b\) and \(c\) are the branches. The order of those two branches is indefinite, therefore \(<a[b,c]d>\) occurs as \(abcd\) or \(acbd\) in a database.

Branch pattern can be divided into three categories: concurrent, exclusive and trivial.

In the following description, notation \(\text{sup}(a\&b)\) is used to denote the support for two sub-pattern \(a\) and \(b\) which appear in the same customer sequence at the same time.

- **Concurrent Branch Pattern.** For any two given sub-patterns \(s_i\) and \(s_j\), if \(\text{sup}(s_i\&s_j) \geq \text{minsup}\), then they constitute a concurrent branch pattern and denoted by \([s_i \& s_j]\). Concurrent branch pattern mining is discussed in detail in section 3.

- **Exclusive Branch Pattern.** For any two given sub-patterns \(s_i\) and \(s_j\), if \(\text{sup}(s_i) \geq \text{minsup}\), \(\text{sup}(s_j) \geq \text{minsup}\) and \(\text{sup}(s_i\&s_j) \leq \text{maxsup}\), then they constitute an exclusive branch pattern and denoted by \([s_i \& s_j]\). The maximum support \(\text{maxsup}\) is defined as an acceptable maximum probability for some event to occur.

- **Trivial Branch Pattern.** A branch pattern which is neither an exclusive branch pattern nor a concurrent branch pattern is called trivial branch pattern.

**Definition 4 (Iterative Pattern)** A sequential pattern is called an iterative pattern if it is made up of only one sub sequential pattern \(S\), which appears at least \(n\) times \((n \geq 2)\), and at most \(m\) times \((m \geq n)\). The expression \(<\{S\}^n>\) denotes the iterative pattern. If a sub-pattern \(S\) can be repeated at most \(m\) times, the iterative pattern will be denoted by \(<\{S\}^m>\), and if a sub-pattern \(S\) can be repeated at least \(n\) times, the iterative pattern will be
denoted by \( \langle S \rangle \). Hence the expression \( \langle S \rangle \) means \( S \) occurs at least twice (i.e. \( n=2 \)).

As an example, a sequential pattern \( \langle a \, a \rangle \) is an iterative pattern since it is made up of two \( a \)s and denoted by \( \langle \{a\} \rangle \). A sequential pattern \( \langle a,b \rangle (a,b) \) \( \langle a,b \rangle \) is also an iterative pattern made up of 3 \( (a,b) \)s and denoted by \( \langle \{(a,b)\} \rangle \).

**Definition 5 (Structural Pattern)** A structural pattern is a general designation of mono-pattern, sequential pattern, branch pattern, iterative pattern, and their composition, it is made up of some elements and their sequence order relations.

For example, \( < a \, \{(b, c)\} \, \{d\}_3 \, \{e \, (f, g)\}_4 \, a \, h \> \) is a structural pattern.

**Definition 6 (Pattern Size)** The maximal length of all sub sequential pattern of a structural pattern \( p \) is called Pattern Size, and denoted by \( PSize(p) \).

For example, patterns \( <acd>,<a[a,b]c> \) and \( <[a,b,c,d,e][a,j]> \) are all have the same size of 3.

### 1.3 Post Sequential Pattern Mining Problem Statement

The mining task based on the sequential patterns mining is Post Sequential Patterns Mining. The ultimate goal of this new mining task is to discover the hidden branch pattern, iteration pattern and structural pattern. This new mining task is complex and there are many questions to be asked. For example: (i) How does one find each part of structural patterns? (ii) What is a better way to find them? (iii) What are the actual meanings of these patterns? (iv) Where and how can these patterns be applied? This paper only focuses on tasks with respect to concurrent branch pattern mining.

### 2. CONCURRENT BRANCH PATTERN MINING

One of our contributions is the definition of branch pattern, iterative pattern and structural pattern. The main form of branch pattern is concurrent branch pattern, which indicates that the sequences in different branches may appear in the same customer sequence within a believable probability. In this section, the concurrent branch pattern mining problem is tackled.

#### 2.1 Concurrent Group and Maximal Concurrent Group Set

Before we present a mining algorithm to discover all concurrent branch patterns, we propose another new concept called concurrent group.
Definition 7 (Concurrent Group, CG) Given customer sequences, set of items (or itemset) that have transaction support above \( \text{minsup} \) makes up a concurrent group and it is denoted by CG for brief.

Example 1 Consider the following customer sequences and let \( \text{minsup} \) be 50%: \(<a (a,b,c)(a,c)d(c,f)>\), \(<(a,d) c (b,c) (a,c)>\), \(<(e,f) (a,b) (d,f) c b>\), \(<g (a,f) c b e>\).

Items (or itemset) sets \{a,b,c,d\}, \{(a,b),c,d,f\} and \{(a,c),b,d\} are all examples of concurrent group since the condition in the definition is satisfied. From definition 7 we know that concurrent group is a set and the elements in this set can be an item or an itemset. Consider \{(a,b),c,d,f\} for example, four elements are contained in this concurrent group, one is an itemset \{(a,b)\} and the other three are items \(c,d,\) and \(f\).

Further explanation concerning the Concurrent Group is as follows:

- For any itemset element of a concurrent group CG, items in the itemset can also be considered as an item element of CG when this CG is compared with another set. For example \{(a,b),c,d,f\} can also be considered as \{(a,b),c,d,f\}. This is useful in understanding the rough concurrent branch pattern which will be introduced in section 3.2.
- Any two elements of a CG should not include each other. For example, \{(a,c),a,c,b,d\} is not a concurrent group, for its elements \(a\) and \(c\) are included by another element \(a,c\).

Definition 8 (Maximal Concurrent Group, MCG) A concurrent group is called a maximal concurrent group if any of its superset is not a concurrent group. The set of Maximal Concurrent Group Set is denoted by MCGS for abbreviation.

Example 2 Consider the previous example, among these following three concurrent groups \{(a,b,c,d)\}, \{(a,b),c,d,f\} and \{(a,c),b,d\}. The group \{(a,b),c,d,f\} is a maximal concurrent group but \{(a,b,c,d)\} is not, since its superset \{(a,b),c,d,f\} is a concurrent group.

The set of Maximal Concurrent Group of example 1 is \(\text{MCGS} = \{(a,b),c,d,f\}, \{(a,c),(b,c),d\}, \{a,b,c,e,f\}\). If each customer sequence is considered as a transaction, then discovering concurrent group from customer sequence is identical to the association rules mining from the transaction.

2.2 Rough Concurrent Branch Pattern

Following the definition of the maximal concurrent group in the previous section, we investigate the relation between the Maximal Sequence Set (MSS) discovered in sequential patterns mining and the maximal concurrent group proposed.
**Definition 9 (Rough Concurrent Branch Pattern, RCBP)** Let \( C \) be a maximal concurrent group in MCGS. *Concurrent sequences* can be obtained by the *sequential intersection operation* of \( C \) and each element in \( MSS \) respectively. These concurrent sequences constitute a **rough concurrent branch pattern (RCBP)**.

*Sequential intersection operation* can be treated as a normal intersection, and the sequence relations among elements after this operation will be consistent with that in the original sequence pattern. The notation of sequential intersection is

*Sequential pattern or Sequential pattern set \( \cap \) Concurrent Group*

The main goal of our work is to discover concurrent branch pattern. We start by finding the rough concurrent branch pattern and then refine it as presented in section 3.3.

**Algorithm 1 (Getting a RCBP)**

Input: Maximal Concurrent Group \( C \) and Maximal Sequence Set \( MSS \).

Output: Rough Concurrent Branch Patterns \( RCBP(C) \).

Method: Finding rough concurrent branch patterns in the following steps.

1. Let rough concurrent branch pattern for \( C \), \( RCBP(C) \), be empty.
2. For each element \( ms \) in \( MSS \)
   - Add \( ms \) to \( RCBP(C) \);
     - For each element (item or itemset) \( i \) in \( ms \), test if \( i \) is an element of \( C \) or \( i \) is included in one element of \( C \);
     - If neither condition is satisfied, then delete \( i \) from \( ms \).
3. Delete the element in \( RCBP(C) \) which contained by another pattern in the \( RCBP(C) \).
4. The result is \( RCBP(C) \).

**Example 3** Given \( MSS=\{<acb>, <efcb>, <a(b,c)a>, <a(b)d>, <fbc>, <a(b)f>, <abc>, <dbc>, <abc>, <acc>, <a(c)>\} \). Let us find the rough concurrent branch pattern for the maximal concurrent group in example 2.

The *sequential intersection* of maximal concurrent group \( \{(a,b),c,d,f\} \) and each element in \( MSS \) is \( MSS \cap \{(a,b),c,d,f\}=\{<acb>, <fcb>, <aa>, <a(b)d>, <fbc>, <a(b)f>, <abc>, <abc>, <acc>\} \). This is the rough concurrent branch pattern \( RCBP(1) \). Similarly, \( MSS \cap \{(a,c),(b,c),d\}=\{<acb>, <a(b,c)a>, <dbc>, <abc>, <acc>, <a(c)>\}=RCBP(2) \); \( MSS \cap \{(a,b,c,e,f)\}=\{<acbf>, <efcb>, <aa>, <fbc>, <erec>, <abc>, <acc>\}=RCBP(3) \). There are three rough concurrent branch patterns in this example.

The following theorem can ensure the correctness of algorithm 1.

**Theorem 1** All concurrent sequences obtained by sequential intersection operation in a RCBP constitute a concurrent branch pattern.
Proof. Concurrent sequences in a RCBP are the sequential intersection of a maximal concurrent group C and each element in MSS respectively. Since we know that:

- All concurrent sequences are sequence patterns, for they are all from MSS;
- All elements in any one sequence of RCBP are concurrent, for they are all from C, a maximal concurrent group in MCGS.

Therefore theorem 1 is true.

2.3 Refining of Rough Concurrent Branch Pattern

RCBPs are only rough concurrent branch patterns, which should be refined for getting the most accurate concurrent branch patterns. What does an accurate concurrent branch pattern mean? Consider \([abc] \otimes [afg]\) and \([a[bc] \otimes [fg]]\) for example. \([a[bc] \otimes [fg]]\) is obtained by combining the common prefix a in \([abc] \otimes [afg]\). The latter is considered more accurate than the former.

**Definition 10 (Common Item/Itemset Set, CIS)** Given any two patterns \(s_i\) and \(s_j\), item/itemset \(i\) is called a common item/itemset of \(s_i\) and \(s_j\) if \(i\) is contained in both patterns. All common item/itemset for a group of patterns constitute a **common item/itemset set (CIS)**.

For example, for patterns \(<acc>\) and \(<(a, c)>\), their Common Item/Itemset Set is CIS = \{a, c\}. For \(<eacb>\) and \(<efcb>\), CIS = \{e, c, b\}.

**Definition 11 (Common Pattern Pair, CPP)** A pair of patterns \(a\) and \(b\) are called **common pattern pair** if they share a common item (or itemset).

There maybe several common pattern pairs for each ci \(\in\) CIS. The notation CPP(ci) is used to denote the set of common pattern pairs of ci.

**Example 4** Consider \(<eacb>\) and \(<efcb>\), CIS = \{e, c, b\}, if pattern size is set to 2, i.e. PSize(CPP) = 2, we have CPP(e) = \{<e[a,f]>, <e[a,c]>, <e[a,b]>, <e[f,c]>, <e[f, b]>, <e[b,c]>\}.

**Definition 12 (Accurate Concurrent Branch Pattern, ACBP)** Concurrent sequences in RCBP are accurate if there are no common pattern to be taken out to reconstruct a new concurrent branch pattern. Accurate concurrent branch pattern is the refined result of rough concurrent branch pattern.

**Example 5** Consider concurrent sequences \([abc] \otimes [cdf] \otimes [xy]\) for example, no common pattern can be founded. Therefore, it is an accurate concurrent branch pattern.

**Definition 13 (Concurrent Structural Pattern, CStruP)** A pattern which is made up of sequential patterns and concurrent branch patterns is called concurrent structural pattern (CStruP for brief). RCBP is a special form of concurrent structural pattern. From definition 12, we can define that
a concurrent structural pattern is accurate if all the concurrent branch patterns are accurate. \textit{ACStruP} is used to denote an accurate concurrent structural pattern. An accurate concurrent branch pattern defined in definition 12 is a special form of \textit{ACStruP}.

**Example 6** \([<[a\oplus b]c>\oplus <xy> \ominus <c [d\oplus b]>]\) is a concurrent structural pattern since it is composed of sequential patterns \(<xy>\) and two concurrent branch patterns \(<[a\oplus b]c>\) and \(<[c [d\oplus b]>\). This concurrent structural pattern is accurate since these two concurrent branch patterns are accurate.

Next, the notation \(\text{MSS}(\text{CStruP})\) is used to represent the Maximal Sequential Set for concurrent structural pattern. The \(\text{MSS}(\text{CStruP})\) can be considered as one operation. All sequential patterns of \(\text{CStruP}\) and branches in branch patterns of \(\text{CStruP}\) are included in \(\text{MSS}(\text{CStruP})\).

**Theorem 2** For a given pattern set \(P = \{p_1, p_2, ..., p_n\}\), \(\text{MSS}(P) = \text{MSS}(\text{MSS}((p_1)) \cup \text{MSS}((p_2)) \cup ... \cup \text{MSS}((p_n)))\).

**Theorem 3** For three given patterns set \(P\), \(P_1\) and \(P_2\) where \(P_1 = \{p_{11}, p_{12}, ..., p_{1m}\}\), \(P_2 = \{p_{21}, p_{22}, ..., p_{2n}\}\), and if the condition \(P = P_1 \cup P_2\) holds, then it is concluded that \(\text{MSS}(P) = \text{MSS}((\text{MSS}(P_1) \cup \text{MSS}(P_2))\).

The above two theorems form the theoretical foundations for concurrent patterns refining. The definition of concurrent patterns refining is given as follows:

**Definition 14 (Concurrent Patterns Refining, CPR)** The process that refines Concurrent Structural Pattern \(\text{CStruP}\) is called concurrent patterns refining. This process can be denoted by \(\text{CPR}(\text{CStruP})\). The refined result is that \(\text{MSS}(\text{ACStruP})\) is equal to \(\text{MSS}(\cup \text{CStruPs})\).

**Theorem 4** For a given \(\text{RCBP}\), suppose \(\text{RCBP} = \{P\} \cup \text{RCBP'}\), where \(P\) is an element of \(\text{RCBP}\), \(\text{RCBP'}\) is a sub set of \(\text{RCBP}\) when \(P\) is deleted from \(\text{RCBP}\), then \(\text{CPR}(\text{RCBP}) = \text{CPR}((P) \cup \text{CPR}(\text{RCBP'}))\).

Theorem 4 is the foundation of the CPR algorithm to be discussed in the next section. Proofs of theorem 2,3 and 4 are omitted due to the restriction on the paper length.

### 2.4 Concurrent Branch Patterns Mining Algorithm

The following algorithm is used to find the concurrent branch patterns.

**Algorithm 2 (Finding Concurrent Branch Pattern)**

**Input:** A transaction database \(DB\) and a minimum support \(\text{minsup}\).

**Output:** Concurrent branch patterns \(\text{CBP}\).

**Method:** Finding concurrent branch patterns in the following steps:

1. Find Maximal Concurrent Group Set \(\{MCGS\}\) from customer sequences in \(DB\) using traditional algorithm such as association rules mining.
2. Find MSS from customer sequences using traditional sequential patterns mining algorithm.

3. Calculate Rough Concurrent Branch Patterns RCBPs using Algorithm 1 based on the MCGS and MSS.

4. Find the Accurate Concurrent Structural Pattern ACSruP for each RCBP by calling Algorithm 3.

5. The Union of all the ACSruPs is the concurrent branch patterns set CBP.

Algorithm 3 (Finding the ACSruP for a RCBP)
Input: Rough Concurrent Branch Pattern r.
Output: Accurate Current Structural Pattern ACSruP
Method: Call CPR(r)
Procedure CPR(r)
{
    if r contains only one sequential pattern, then return r. /*r is the result*/
    Decompose r into two parts such that r = {p} ∪ r1, where p is the first element of r (also a sequential pattern), and r1 is the set of the left patterns by removing p from r.
    return the result by calling CPR_2P( p, CPR(r1) ) /* CPR_2P is shown in algorithm 4*/
}

The definition of Common Item/Itemset Set and Pattern Size introduced previously will be useful in the understanding of the following algorithm.

Algorithm 4 (Finding accurate concurrent structural pattern for two patterns)
Input: Patterns p1 and p2, where p1 must be a sequential pattern, p2 may be an ACSruP.
Output: Accurate Current Structural Pattern P
Method: Call CPR_2P(p1,p2)
Procedure CPR_2P(p1,p2)
{
    Let P={p1} ∪ {p2}. /* P will be used as the result of CPR_2P */
    Calculate the common item/itemset set CIS for p1 and p2.
    Let pattern size variable PSize=2.
    Do while CIS is not empty {
        For each element ci∈CIS, find CPP(ci). CPP(ci) must satisfy the conditions: CPP(ci) should not be sub-patterns of any pattern in P, and PSize(CPP(ci)) is PSize.
        For each possible pattern in CPP(ci), test if its support is above minsup.
        If it satisfies the condition then let it be included into P.
        For any element ci in CIS, if the support of every element in CPP(ci) is all below minsup, then delete ci from CIS.
Clear up \( P \) by deleting patterns, which have any super pattern in \( P \).

Let \( P_{\text{Size}} = P_{\text{Size}} + 1 \).

return \( P \).

A concrete example of the algorithm 4 for readers' understanding is given follows:

Example 7 Consider the customer sequences in example 1 and suppose \( \text{minsup} = 50\% \): \(<a (a,b,c) (a,c) d (c,f)>, <(a,d) c (b,c) (a,c)>, <(e,f) (a,b) (d,f) c b>, <e g (a,f) c b c>\). Let patterns \(<\text{eacb}> \) and \(<\text{efcb}> \) be the input patterns of CPR_2P. While calling CPR_2P:

1. \( P = \{<\text{eacb}>, <\text{efcb}>\}, \text{CIS} = \{e, c, b\}, P_{\text{Size}} = 2 \).
2. For the first element of CIS e, \( \text{CPP} = \{<e[a,f]>\} \).
3. Since the support of \(<e[a,f]>\) is above \( \text{minsup} \), add it into \( P \). Thus, \( P = \{<\text{eacb}>, <\text{efcb}>, <e[a@ f]>\} \). Note that \(<e[a,c]>, <e[a,b]>, <e[f,c]>, <e[f, b]> \) are not in CPP, for they are sub-patterns of some elements of \( P \).
4. Similarly, for element of CIS c and b, the result is \( P = \{<\text{eacb}>, <\text{efcb}>, <e[a@ f]>, <[a@ f]c>, <[a@ f]b>\} \).
5. When \( P_{\text{Size}} = 4 \), \( P = \{<\text{eacb}>, <\text{efcb}>, <e[a@ f]c>b>\} \).
6. Delete \(<\text{eacb}> \) and \(<\text{efcb}> \) from \( P \) for they are sub-patterns of \(<e[a@ f]c>b>\).
7. The final result is \( P = \{<e[a@ f]c>b>\} \).

3. RELATED WORK

Discovering Structural Patterns seems to be new to data mining. There are, of course, several topics in which related issues have been considered. A theoretical framework and practical methods are described in [13] for finding event sequence decompositions. These methods used the probabilistic modelling of the event generating process. Recently, a simple and efficient data mining algorithm presented in [6] to discover all fragments that satisfy certain conditions. A fragment is an ordering of a subset of variables in a 0-1 dataset. They describe the criteria, frequency and violation fraction, to be used for finding potentially interesting fragments of order. However, the emphasis of our work is on finding structural patterns (including sequential pattern, branch pattern or iterative pattern) based on the result of sequential patterns mining.

We mentioned that we can apply the post sequential pattern mining into event-based modelling. In traditional workflow modelling process, a designer has to construct a detailed workflow model accurately describing the routing of the work. It is time consuming and the initial design is often
incomplete and subjective. Since WfMSs (Workflow Management Systems) log many events that occur during process executions, so a technique called workflow mining (also referred to as process mining) was proposed to solve the problem in workflow model design. Workflow mining is an approach to reverse the process and the information collected at run-time can be used to derive a model. The process mining is not new [2,3,4,7,8,9], however most process mining results are limited to sequential behavior. Not much work has been done to find concurrent or iterative behavior. Our approach in workflow modelling is one of the key directions for further research.

4. CONCLUSIONS AND FUTURE WORK

One of the future research directions of data mining is to propose new mining task or discover various patterns. Post sequential patterns mining is just for this purpose. This novel mining task is based on sequential patterns mining. In this paper we first reviewed the Sequential Patterns Graph (SPG) as proposed in [12]. It is clear from the previous work that SPG is a bridge from discrete sequences set to structural knowledge and it is also the foundation of post sequential patterns mining.

The main purpose of post sequential patterns mining is to discover the hidden structural patterns in event-based data. Before addressing the problem of post sequential patterns mining, we defined formally some new patterns including branch pattern, iterative branch and structural pattern. Concurrent branch pattern is an important pattern, which occurs in many event-based data. Thus, we concentrated on concurrent branch pattern mining in this paper.

An important phase for our work is to perform more experiments to support our theories. In our previous work, we implemented the algorithm for constructing SPG and analyzed the efficiency of that approach. In our existing research work, we anticipate that more experiments are needed to demonstrate the affective nature and efficiency of concurrent branch patterns mining algorithms. This paper has been theoretical, experimentation is on going to establish the validity of our algorithms. In addition to the above, we intend to extend the method to cover concurrent branch patterns to exclusive branch patterns mining or iterative patterns mining. This, we envisage will be our ultimate goal.
REFERENCES


