Knowledge-Based Systems
Manuscript Draft

Manuscript Number: KNOSYS-D-15-00605

Title: Mining high-utility sequential patterns by using duration and gap constraints

Article Type: Full Length Article

Keywords: utility sequential pattern mining, PrefixSpan algorithm, constraint-based mining, duration constraint, gap constraint.

Mining high-utility sequential patterns by using duration and gap constraints

Ya-Han Hu[a], Cheng-Hsiung Weng[b,*], Chi-Yuan Chang[a]

[a] Department of Information Management, National Chung Cheng University, Taiwan, Republic of China.

[b] Department of Management Information Systems, Central Taiwan University of Science and Technology, Taiwan, Republic of China.

E-mail: chweng@mgt.ncu.edu.tw

Mining high-utility sequential patterns by using duration and gap constraints

Abstract

High-utility sequential pattern mining (high-utility SPM) is a major data mining technique that explores the high-utility purchasing behavior of customers. However, previous high-utility SPM studies have generated many long sequential patterns, which are not meaningful for predicting customers' purchasing behavior. In this study, we considered the duration and gap constraints in high-utility SPM. First, we added a maximum-span-length constraint to discover patterns that occurred within a specific period. Next, the maximum and minimum gap constraints were used to confine the reasonable time-interval between adjacent events. Finally, a novel high-utility SPM algorithm was applied to mine high-utility sequential patterns by integrating duration constraints. The experimental results demonstrated that (1) the proposed approach outperforms conventional utility SPM in runtime, (2) the average utility of each pattern generated by the proposed approach is higher than that of conventional utility SPM, and (3) the predicted value of the patterns generated by the proposed approach is superior to that of conventional utility SPM.

Keywords: utility sequential pattern mining, PrefixSpan algorithm, constraint-based mining, duration constraint, gap constraint.

1. Introduction

Sequential pattern mining (SPM) is one of the most widely used data mining techniques. The main purpose of SPM is to mine relationships among events (i.e., itemsets) (Agrawal & Srikant, 1995; Hu et al., 2013a). The discovered patterns can be used to predict consequent behavior. SPM has been applied in many fields such as market basket analysis (Ngai et al., 2009), customer shopping behavior (Zhang et al, 2004; Hu et al., 2013b; Hu et al., 2015), meteorological and geological exploration (Boulvain et al., 2009), medical analysis (Zheng et al., 2007), gene sequence analysis (Yi et al., 2012), website analysis (Das & Turkoglu, 2009), and stock prediction (Mehzabin Shaikh & Chhajed, 2012).

Although SPM explores pattern frequency and reveals the relationships among different events at different time points, it ignores additional valuable pattern information such as the purchase amount, cost, value, and profit of a pattern (Masseglia et al., 2009). In general, high-profit or luxury items rarely occur in both transactions and sequences; thus, discovering patterns containing luxury items by using conventional methods is difficult. Therefore, mining high-utility patterns from a sequence database, also known as high-utility SPM, has been discussed extensively in recent years (Ahmed et al., 2010; Wang et al., 2007).

In contrast to conventional SPM methods, utility SPM can discover high-utility patterns by using a user-specified minimum utility support. Users can discover valuable patterns according to their needs. However, utility SPM has the same limitation as that of the conventional SPM; that is, both of the two SPM methods generate several long but meaningless sequential patterns when the average length of a sequence is long. Consider the following two utility sequential patterns discovered from a customer sequence database: <(*TV*, *5.1 sound system*)(*washing machine*, *dryer*)> and <(*PC*, *monitor*)(*printer*, *toner*)>. Assume that the average time span of the first pattern is 2 years, whereas that of the second pattern is 1 month. The first pattern is less significant than the second. Without considering the time span between the events in a sequence, several high-utility but meaningful rules may be discovered. Thus, although utility SPM can discover patterns that are more valuable (i.e., higher purchase amount or profit) than conventional SPM can, the long sequence problem must be solved. A feasible solution is to consider useful time constraints in the utility SPM.

This study considered both the duration and gap constraints in utility SPM for solving the long sequence problem. First, we added the maximum-span-length constraint to discover patterns that occurred within a specific period. Next, the maximum and minimum gap constraints were used to confine a reasonable time interval between two adjacent events. Finally, a novel high-utility SPM with duration (HUD) constraint algorithm is proposed for mining high-utility sequential patterns by considering duration constraints.

The remainder of this paper is organized as follows. Section 2 reviews previous studies related to utility pattern mining (UPM), utility SPM, and constraint based-mining. Section 3 formally defines HUD sequential patterns. Section 4 proposes the HUD-PrefixSpan algorithm, which is an extension of the well-known PrefixSpan algorithm. Section 5 details the experimental results. Finally, Section 6 provides conclusions and recommendations for future works.

2. Related work

2.1. Utility pattern mining and utility SPM

UPM, an extension of conventional frequent pattern mining (FPM), aims at discovering potentially valuable patterns from databases (Chan et al., 2003). Instead of using a frequency threshold, UPM considers the profit of each pattern. Given a user-specified minimum utility threshold, a utility pattern is defined as a pattern in which the value is higher than or equal to the minimum utility threshold. The framework of UPM enables discovering patterns that contribute to high business profits, even if the patterns have low frequency.

The main difference between FPM and UPM is that UPM does not satisfy the downward closure property (Chan et al., 2003). In particular, a subset of a utility pattern may not be a utility pattern. Without this property, the number of candidate patterns may be considerably high; thus, more memory space and run time are required for discovering a complete set of utility patterns. To address this concern, Liu et al. (2005) defined the transaction-weighted utility (TWU) architecture for reducing the number of candidates in the mining process, and proposed a two-phase algorithm for discovering high-utility patterns.

Conventional UPM conducts level-wise search, resulting in poor efficiency. Therefore, many scholars have developed various approaches for improving the mining efficiency. Yao and Hamilton (2006) proposed the utility mining (UMining) algorithm for improving the efficiency of the mining process. Erwin et al. (2007b) proposed the CTU-Mine, which does not generate a high number of candidates, based on the pattern growth method. The results showed that, for dense data sets, the CTU-Mine outperformed the two-Phase algorithm. Li et al. (2008) defined the isolated items discarding strategy (IIDS) for discovering high-utility itemsets. Ahmed et al. (2011) proposed an efficient candidate pruning technique for mine high-utility patterns.

In addition, there are many pattern-growth approaches for finding high-utility patterns. Erwin et al. (2007a) proposed the CTU-PRO, which is based on a pattern growth approach. Several efficiency tests have showed its superiority on various data sets. Tseng et al. (2010) proposed the HUC-Prune with an efficient candidate pruning technique based on pattern growth. The results showed that the efficiency surpassed that of the two-phase, FUM, and DCG+ algorithms for both sparse and dense data sets. Ahmed et al. (2009) proposed an efficient tree-based algorithm, called IHUP. The evaluation results showed that IHUP outperformed the level-wise candidate-generation-and-test algorithms. Liu et al. (2012) developed an efficient algorithm, HUI-Miner which effectively reduces the number of candidates by using a utility-list structure.

Several other extensions on UPM have been proposed. Wu (2012) proposed Top-K utility patterns. Ahmed et al. (2009) proposed an efficient tree structure for UPM in incremental databases. Chu et al. (2009) proposed an efficient algorithm for mining high-utility itemsets with negative item values (i.e., items sold with loss or no profit).

Recently, UPM techniques have been extended to the domain of SPM (also known as utility SPM) (Ahmed et al., 2010; Lan et al., 2012; Yin et al., 2012). Utility SPM has demonstrated practicality and flexibility in several studies. Ahmed et al. (2010) proposed a novel approach, called the HUSP algorithm, for mining high-utility sequential patterns in sequence databases. In HUSP, the concept of TWU has been replaced by that of sequence-weight utility (SWU). The first stage of HUSP is to discover all the high sequence-weight utility sequential patterns (HSWU-SPs); that is, patterns with SWU higher than the minimum sequence utility threshold. The second stage rescans the database to calculate the value of all actual HSWU-SP values for identifying all high-utility sequential patterns. In the experimental study, both the UtilitySpan (US) and Utility Level (UL) algorithms were used to compare the efficiency with that of the HUSP algorithm. The UL algorithm conducted level-wise search with multiple database scans, and the US algorithm extended the PrefixSpan algorithm to generate fewer candidate itemsets with fewer database scans. The results showed that the proposed algorithms achieved higher efficiency and scalability than either the UL or US method did. Lan et al. (2012) proposed an improved approach for utility SPM. Yin et al. (2012) proposed an efficient algorithm for mining high-utility sequential patterns.

2.2. Constraint-based mining

Previous studies have shown that pushing user-specified constraints into the mining process can generate patterns that are more meaningful and reduce the runtime (Garofalakis et al., 1999; Lin et al., 2008; Radhakrishna et al., 2013). Table 1 lists previous studies on constraint-based mining, which was first used in the fields of FPM and SPM. Pei et al. (2002) categorized previous constraint-based studies into the following: item, length, super-pattern, aggregate, regular-expression, duration, and gap constraints. These constraints can be roughly divided into two categories: temporal and non-temporal constraints; non-temporal constraints are applied to FPM but temporal constraints can be considered in SPM only.

Table 1 Previous studies on constraint-based pattern mining.

Researcher	**Type**	**Constraints**						
		item	**length**	**super-pattern**	**aggregate**	**regular expression**	**duration**	**gap**
Pei et al., 2007	SPM			✓				
Seno & Karypis, 2002	SPM		✓					
Garofalakis et al., 1999	SPM					✓		
Ji et al., 2007	SPM							✓
Chen & Hu, 2006	SPM						✓	
Wang et al., 2007	UPM				✓			
Sandhu et al., 2010	UPM	✓						
Hu et al., 2010	UPM				✓		✓	
Li et al., 2011	UPM	✓						
Wu et al., 2013	Utility SPM						✓	
This study	Utility SPM						✓	✓

After the utility-based mining methods were proposed, researchers determined that constraints can also be considered for mining utility patterns that are more meaningful. As shown in Table 1, in utility-based mining, constraint-based UPM has been investigated frequently but utility-based SPM has been rarely mentioned. Based on our research, Wu et al. (2013) is the only study that has considered the duration constraint in utility SPM. However, a duration constraint confines only the time span between the first and final events in a sequence, which must be within a user-specified threshold. The minimum or maximum duration between any two adjacent events cannot be specified, thus limiting the ability to determine more meaningful utility SPM. To solve this problem, this study considered both the duration and gap constraints. The maximum- and minimum-gap thresholds were required in the mining process.

3. Problem definition

Let I denote a set of all items in a database. A data sequence α can be represented by $\langle (a_1, t_1, q_1), (a_2, t_2, q_2), ..., (a_n, t_n, q_n) \rangle$, where (a_j, t_j, q_j) indicates that item a_j $(a_j \in I)$ was purchased at time t_j with quantity q_j, $1 \le j \le n$, and $t_{j-1} \le t_j$ for $2 \le j \le n$. If items occur simultaneously in a data sequence, they are ordered alphabetically. Based on this format, we defined the following:

Definition 1. Given a data sequence, $\alpha = \langle (a_1, t_1, q_1), (a_2, t_2, q_2), ..., (a_n, t_n, q_n) \rangle$ and an itemset, $I_p = (i_1, i_2, \ldots, i_m)$, where $I_p \subseteq I$ and $m \le n$. I_p is contained in α if there are m integers $1 \le k_1 < k_2 < \ldots < k_m \le n$, such that $i_1 = a_{k_1}, i_2 = a_{k_2}, ..., i_m = a_{k_m}$ and $t_{k_1} = t_{k_2} =, ..., = t_{k_m}$.

Definition 2. (*Sequence and subsequence*) Let $\beta = \langle I_1, I_2, ..., I_s \rangle$ be a sequence of itemsets, where $I_p \subseteq I$ ($1 \le p \le s$). Sequence β is contained in α or a subsequence of α if the following conditions are satisfied: (1) Each I_p in β is contained in α, and (2) $t_{I_1} < t_{I_2} < ... < t_{I_s}$ where t_{I_p} is the time at which I_p occurs in α.

Example 1. Let a data sequence

$\alpha = \langle (\text{a},35,12),(\text{c},40,8),(\text{e},40,1),(\text{d},50,3),(\text{a},60,6),(\text{e},70,3),(\text{b},75,2),(\text{d},75,2) \rangle$.

Sequence $\beta = \langle (a)(e)(d) \rangle$ is contained in data sequence α because itemsets (*a*), (*e*), and (*d*) are contained in α at times 35, 40, and 50, respectively, and $t_{(d)} < t_{(e)} < t_{(a)}$.

Definition 3. (*d-subsequence*) Let *ms_length*, *maxgap*, and *mingap* be user-specified *maximum-span-length*, *maximum gap*, and *minimum gap* constraints, respectively. Assume that $\beta = \langle I_1, I_2, ..., I_s \rangle$ is a subsequence of α. The variable β is called a d-subsequence (i.e., a subsequence satisfying *ms_length*, *maxgap*, and *mingap* constraints) of α if the following conditions are satisfied: (1) β is a subsequence of α; (2) $t_{I_s} - t_{I_1} \le$ *ms_length*; and (3) *mingap* $\le t_{I_p} - t_{I_{p-1}} \le$ *maxgap* for $2 \le p \le s$.

Example 2. From Example 1, the sequence $\langle (a)(e)(d) \rangle$ is a subsequence of α. Let *ms_length* = 30, *maxgap* = 12, and *mingap* = 3. $\langle (a)(e)(d) \rangle$ is called a d-subsequence of α because the sequence $\langle (a)(e)(d) \rangle$ also satisfies (1) $t_{(d)} - t_{(a)} = 50 - 35 = 15 \le ms\_length = 30$; (2) $t_{(d)} - t_{(e)} = 50 - 40 = 10$, and $t_{(e)} - t_{(a)} = 40 - 35 = 5$, which satisfy both *mingap* and *maxgap* requirements.

Definition 4. A d-subsequence β cyclically occurs *m* times in α if the concatenation of β is also a d-subsequence of α and the repetitions of β in α

are equal to *m*; that is, sequence $<...\beta_1...\beta_2......\beta_m...>$ is a subsequence of α.

Example 3. From Example 2, in data sequence $\alpha = \langle (\text{a},35,12),(\text{c},40,8),(\text{e},40,1),(\text{d},50,3),(\text{a},60,6),(\text{e},70,3),(\text{b},75,2),(\text{d},75,2) \rangle$, the d-subsequence $\langle (a)(e)(d) \rangle$ cyclically occurs twice in α (the first occurrence is during time interval 35–50 and the second is during time interval 60–75).

Definition 5. (*Sequence utility value*) Let $P(a_i)$ denote the unit profit of item a_i. According to Definition 4, a d-subsequence β cyclically occurs *m* times in α. Let B_r denote the *r*th repetition of β in α, the sequence utility value of *r*th repetition of β, denoted as $su_r(\beta,\alpha)$, is defined as follows.

$$su_r(\beta,\alpha) = \sum_{a_i \in \beta_r} p(a_i) \times q_{a_i}$$

Moreover, the sequence utility value of β in α, denoted as $su(\beta,\alpha)$, is defined as the sum of sequence utility values of all β repetitions.

$$su(\beta,\alpha) = \sum_{r=1}^{m} su_r(\beta,\alpha)$$

Example 4. Consider the list of item unit prices shown in **Error! Reference source not found.**. From Example 3, the d-subsequence $\langle (a)(e)(d) \rangle$ cyclically occurs twice in α. The sequence utility value of the first repetition is $su(\beta,\alpha) = 12\times10 + 1\times75 + 3\times50 = 345$, and that of the second repetition is $su(\beta,\alpha) = 6\times10 + 3\times75 + 2\times50 = 385$. Therefore, $su(\beta,\alpha) = 345 + 385 = 730$.

Item	**Price**
a	10
b	100
c	35
d	50
e	75

Fig. 1 A list of item unit prices

Definition 6. (*Total sequence utility value*) Given a sequence database *SDB* and a d-subsequence β. The total sequence utility value of β, denoted as $su_{SDB}(\beta)$, is defined as the sum of the sequence utility values of all data sequences containing β in *SDB*.

$$su_{SDB}(\beta) = \sum_{\alpha \in SDB} su(\beta, \alpha)$$

Definition 7. Given a user-specified minimum sequence utility threshold, *minSeqUtil,* a sequence β is defined as a HUD-sequential pattern (HUD-SP) if $su_{SDB}(\beta) \geq minSeqUtil$.

Example 5. Consider the sequence database *SDB* shown in Fig. 2 and the list of item unit prices shown in **Error! Reference source not found.**. Assume that *ms_length* = 30, *maxgap* = 12, and *mingap* = 3. Given a d-subsequence $\beta = \langle (a)(e)(d) \rangle$, β occurs once in sid_{10} and twice in sid_{30}. For sid_{10} and sid_{30}, $su\ (\beta, sid_{10}) = 3 \times 10 + 5 \times 75 + 1 \times 50 = 455$ and $su(\beta, sid_{30}) = 730$. Finally, $su_{SDB}(\beta) = 455 + 730 = 1185$. Moreover, given *minSeqUtil* = 1000, a d-subsequence β is a HUD-SP because $su_{SDB}(\beta) = 1305 \geq 1000$.

Sid	***Sequence***
10	<(*a*,10,6),(*c*,15,4),(*d*,18,8),(*a*,20,3),(*e*,25,5),(*a*,30,2),(*d*,35,1),(*c*,70,3)>
20	<(*d*,30,2),(*a*,45,3),(*b*,50,4),(*e*,60,4),(*d*,80,6)>
30	<(*a*,35,12),(*c*,40,8),(*e*,40,1),(*d*,50,3),(*a*,60,6),(*b*,65,2),(*e*,65,3), (*d*,75,2)>
40	<(*c*,20,2),(*b*,40,1),(*c*,40,12),(*e*,70,5),(*a*,90,25),(*d*,100,10)>
50	<(*b*,70,3),(*c*,70,10),(*d*,80,10),(*a*,90,15),(*c*,100,10)>

Fig. 2 A sequence database

In summary, given a user-specified *minSeqUtil* threshold, the aim of this study was to discover a complete set of HUD-SPs. However, the mining of HUD-SPs was not difficult because HUD-SPs do not support the downward closure property. In other words, a super pattern of a non-HUD-SP may be a HUD-SP. For example, the *d-subsequence* $\langle (a)(e) \rangle$ is not a HUD-SP because $su_{SDB}(\langle (a)(e) \rangle) = 405 + 480 = 885 < 1000$, but its super sequence $\langle (a)(e)(d) \rangle$ is a HUD-SP, as mentioned in Example 4. Without the downward closure property, the search space for discovering a complete set of HUD-SPs increases considerably. To solve this problem in utility SPM, this study defined the HSWU-SP.

Definition 8. (*Sequence-weighted utility*) Given sequence α, let β be a d-subsequence of α. The sequence-weighted utility value of the d-subsequence β in α, denoted as $swu(\beta,\alpha)$, is equal to the total amount of utility gained from α. Moreover, the total sequence-weighted utility of β in *SDB*, denoted as $swu_{SDB}(\beta)$, is defined as the sum of all sequence-weighted utility values of β in *SDB*.

$$swu_{SDB}(\beta) = \sum_{\alpha \in SDB} swu(\beta,\alpha)$$

Definition 9. (HSWU-SP) β is called an HSWU-SP if $swu_{SDB}(\beta) \geq minSeqUtil$.

The HSWU-SP satisfies the downward closure property, and the set of HSWU-SP is a superset of that of HUD-SPs because of the following reason. First, assume that β and β' are two sequences and $\beta' \subset \beta$. Let S_β and $S_{\beta'}$ be the set of sequences containing β and β', respectively. Based on Definition 8, $swu_{SDB}(\beta) = \sum_{\alpha \in S_\beta} swu(\beta,\alpha)$ and $swu_{SDB}(\beta') = \sum_{\alpha \in S_\beta'} swu(\beta',\alpha)$. Because $\beta' \subset \beta$, $S_{\beta'}$ must be the superset of S_β. Therefore, $\sum_{\alpha \in S_\beta'} swu(\beta',\alpha) \geq \sum_{\alpha \in S_\beta} swu(\beta,\alpha)$ (i.e., $swu_{SDB}(\beta') \geq swu_{SDB}(\beta)$. If β' is not an HSWU-SP (i.e., $swu_{SDB}(\beta') < minSeqUtil$), then β must not be an HSWU-SP.

According to Definitions 5 and 8, given sequence α, let β be a d-subsequence of α. The sequence utility value of β considers only the utility values of repetitions of β in α and the sequence-weight utility value of β considers the total amount of utility values gained from α. Therefore, $swu_{SDB}(\beta) \geq su_{SDB}(\beta)$. If β is a HUD-SP (i.e., $su_{SDB}(\beta) \geq minSeqUtil$), then β is an HSWU-SP because $swu_{SDB}(\beta) \geq su_{SDB}(\beta) \geq minSeqUtil$. In other words, the set of HSWU-SP is a superset of that of HUD-SPs.

4. HUD-PrefixSpan algorithm

This section introduces the HUD-PrefixSpan algorithm, which is an extension of the well-known PrefixSpan algorithm, for mining a complete set of HUD-SPs. HUD-PrefixSpan recursively partitions a sequence database into a number of projected databases and retrieves the HUD-SPs by exploring only the local utility patterns in each projected database. Before illustrating the HUD-PrefixSpan algorithm, we first define d-subsequence-prefix (*d-prefix*), d-subsequence-projection (*d-projection*).

Definition 9. (*d-prefix*) Given a data sequence $\alpha = \langle (a_1, t_1, q_1), (a_2, t_2, q_2), ..., (a_n, t_n, q_n) \rangle$ and a sequence $\beta = \langle I_1, I_2, ..., I_s \rangle$, β is called a d-prefix of α if and only if (1) β is a d-subsequence of α, and (2) $a_1 \in I_1$.

Example 6. From Example 2, given a data sequence $sid_{30} = \langle (a,35,12), (c,40,8), (e,40,1), (d,50,3), (a,60,6), (b,65,2), (e,65,3), (d,75,2) \rangle$, let *ms_length* = 30, *maxgap* = 12, and *mingap* = 3. Sequence $\langle (a)(e)(d) \rangle$ is a *d-prefix* of sid_{30} because (1) $\langle (a)(e)(d) \rangle$ is a d-subsequence of α, and (2) item (a) is the prefix item of α.

Definition 10. (*d-projection*) Given a data sequence

$\alpha = \langle (a_1, t_1, q_1), (a_2, t_2, q_2), ..., (a_n, t_n, q_n) \rangle$, let β be a d-subsequence of α. A sequence $\alpha' = \langle (a_1', t_1', q_1'), (a_2', t_2', q_2'), ..., (a_p', t_p', q_p') \rangle$ of sequence α is called a d-projection of α with respect to d-prefix β if and only if (1) α' has the d-prefix β, (2) α' is a d-subsequence of α, and (3) there exists no super sequence α'' of α' such that α'' is a d-subsequence of α and has the d-prefix β.

Example 7. Referring to Example 6 and assuming that sid_{30} is projected with *d*-prefix <(*a*)>, we obtained the following two d-projections: $\langle (a,35,12),(c,40,8),(e,40,1),(d,50,3),(a,60,6),(b,65,2),(e,65,3) \rangle$ and $\langle (a,60,6),(b,65,2),(e,65,3),(d,75,2) \rangle$. Each of the two d-projections satisfies the following conditions: (1) It has d-prefix <(*a*)>, (2) it is a d-subsequence of sid_{30} (i.e., satisfying the *ms_length*, *maxgap*, and *mingap* constraints), and (3) appending additional items after the projection violates any of the three constraints.

According to Example 7, the d-projections generated from the same data sequence may overlap (i.e., the same items may occur multiple times in different d-projections), which leads to over counting. In particular, sequences $\langle (a,60,6),(b,65,2),(e,65,3) \rangle$ occur in both d-projections in Example 7. When cumulating the total sequence utility value of a pattern in the d-projections, the utility value of the same items that existed in different d-projections can be counted only once. In the proposed algorithm, the time stamp of each occurrence in the d-projections was recorded during the counting process. While proceeding to the next d-projection of the same data sequence, the algorithm analyzed the timestamps of each item with those recorded in previous d-projection.

The HUD-PrefixSpan algorithm is shown in Fig. 3. The main procedure is briefly stated as follows. Initially, we set $\beta = null$. The algorithm first calls $HUD-PrefixSpan(null, 0, SDB)$ and appends each item to α to form α-projected database $SDB|_{\beta}$. It then calculates $swu_{SDB}(\beta)$ and $su_{SDB}(\beta)$ for each β to find complete sets of 1-HSWU-SPs (i.e., HSWU-SP with one item only) and 1-HUD-SPs in $SDB|_{\beta}$. The super sequences of an HSWU-SP can be HUD-SPs. Therefore, for each HSWU-SP in *α*, we constructed the projected database *SDB*|*α*' and called the procedure $HUD-PrefixSpan(\beta', 1, SDB|_{\beta'})$ for finding 2-HSWU-SPs and 2-HUD-SPs. Recursively doing so yields the complete sets of HSWU-SPs and HUD-SPs. The following example illustrates the major steps of HUD-PrefixSpan in detail.

```
Input: A sequence database $SDB$, $ms\_length$, $mingap$, $maxgap$ and $minSeqUtil$
Subroutine: $HUD-PrefixSpan(\beta, 0, SDB|_{\beta})$
Parameters:
   $\beta$ is a set of HSWU-SPs
   $l$ is the length of $\beta$
   $SDB|_{\beta}$ is the $\beta$-projected database with d-projection
Output: The complete set of HUD-SPs
Method:
   Each item $\beta$ in $SDB|_{\beta}$ is appended after $\beta$ as $\beta'$ or is added into the first itemset of $\beta$ as $\beta'$
   Scan the database $SDB|_{\beta}$ once
   If $\alpha.time() - \beta.begintime() \le ms\_length$, $mingap \le \alpha.time() - \beta.endtime() \le maxgap$ then calculate
        $swu_{SDB}(\beta')$ for each $\beta'$
   For each $\beta'$
      If $swu_{SDB}(\beta') \ge minSeqUtil$
           Output $\beta'$ as HSWU-SP
           If $su_{SDB}(\beta') \ge minSeqUtil$ then
                Output $\beta'$ as HUD-SP
      Construct $\beta'$-projected database $SDB|_{\beta}$, and call $HUD-PrefixSpan(\beta', l+1, SDB|_{\beta'})$
```

Fig. 3 The HUD-PrefixSpan algorithm

The *SDB* was scanned once for constructing projected databases for each item. After an item's projected database was built, we could easily obtain the item's total sequence utility and sequence-weighted utility and identify whether that item is a

1-HSWU-SP or a 1-HUD-SP. For example, consider the sequence database *SDB* in Fig. 2 and let *ms_length* = 30, *maxgap* = 12, *mingap* = 3, and *minSeqUtil* = 1200. The projected database of item *a* is shown in Fig. 4. For sid_{10}, item *a* occurred at intervals 10, 20, and 40, and we obtained the following d-projections, [sid_{10}:60:10:10]:(*c*,15,4),(*d*,18,8),(*a*,20,3),(*e*,25,5),(*a*,40,2), [sid_{10}:30:20:20]:(*e*,25,5),(*a*,40,2),(*b*,50,1), and [sid_{10}:20:40:40]:(*b*,50,1),(*c*,70,3), where the notation "[*sid*:su_{prefix}:*begin-time*:*end-time*]:<*projection*>" represents the d-projection and its sequence ID, sequence utility value, begin time, and end time of the prefix. The item *a* projections for other data sequences was built similarly. After the projected database of item *a* was constructed, we calculated $swu_{SDB}(\langle(a)\rangle) = 6885$ and $su_{SDB}(\langle(a)\rangle) = 720$. Because $swu_{SDB}(\langle(a)\rangle)$ satisfied *minSeqUtil* but $su_{SDB}(\langle(a)\rangle)$ did not, item *a* was appended to the set of 1-HSWU-SPs and not included in 1-HUP-SPs.

Prefix	su_{SDB}	***sid***	su_{prefix}	***begin time***	***end time***	**Projected (prefix) database**	swu_{sid}
<(*a*)>	720	10	60	10	10	(*c*,15,4),(*d*,18,8),(*a*,20,3)(*e*,25,5),(*a*,30,2)	1180
			30	20	20	(*e*,25,5),(*a*,30,2),(*d*,35,1)	
			20	30	30	(*d*,35,1),(*c*,70,3)	
		20	30	45	45	(*b*,50,4),(*e*,60,4)	1130
		30	120	35	35	(*c*,40,8),(*e*,40,1),(*d*,50,3),(*a*,60,6), (*b*,65,2),(*e*,65,3)	1210
			60	60	60	(*b*,65,2),(*e*,65,3),(*d*,75,2)	
		40	250	90	90	(*d*,100,10)	1715
		50	150	90	90	(*c*,100,10)	1650

Fig. 4 The $\langle(a)\rangle$-projected databases

Next, the algorithm started searching for 2-HSWU-SPs and 2-HUD-SPs with prefix $\langle(a)\rangle$ by calling the algorithm $HUD-PrefixSpan(\langle(a)\rangle, 2, SDB|_{\langle(a)\rangle})$. As shown in Fig. 4, the $\langle(a)\rangle$-projected database consists of eight d-projections. All the information regarding length-2 sequence with prefix $\langle(a)\rangle$ was obtained. In other words, the total sequence utility and sequence-weight utility of length-2 sequences, such as $\langle(a)(a)\rangle$, $\langle(a)(b)\rangle$, and $\langle(a)(c)\rangle$, were calculated. For the first *d*-projection [sid_{10}:60:10:10]:(*c*,15,4),(*d*,18,8),(*a*,20,3)(*e*,25,5),(*a*,30,2), the *su* and *swu* of the three patterns, $\langle(a)(a)\rangle$, $\langle(a)(c)\rangle$, and $\langle(a)(d)\rangle$ were cumulated. The item *e* and the final item *a* (i.e., (*a*,30,2)) were not considered because they did not satisfy the *maxgap* constraint in this d-projection. Simultaneously, the *su* and *swu* values of the three patterns were calculated as follows: $su(\langle(a)(a)\rangle) = su_{prefix} + 3\times10 = 90$, $su(\langle<(a)(c)>\rangle) = 60 + 4\times35 = 200$, $su(\langle(a)(d)\rangle) = 500$, $swu(\langle(a)(a)\rangle) = swu(\langle(a)(a)\rangle) = swu(\langle(a)(a)\rangle) = 1180$. For the second d-projection [sid_{10}:30:20:20]:(*e*,25,5),(*a*,30,2),(*d*,35,1), the items a and d were ignored and only the *su* and *swu* values of pattern <(*a*)(*e*)> were cumulated (i.e., $su(\langle(a)(e)\rangle) = 405$). This is because the utility values of the prefix a in the second d-projection were calculated in the first d-projection; the item d did not satisfy the *maxgap* constraint. This step was followed until all the *su* and *swu* values of length-2 sequences were obtained in $\langle(a)\rangle$-projected databases. Based on the information shown in Table 2, all 2-HSWU-SPs and 2-HUD-SPs with prefix $\langle(a)\rangle$ were determined. In other words, the patterns $\langle(a)(b)\rangle$, $\langle(a)(c)\rangle$, $\langle(a)(d)\rangle$, and $\langle(a)(e)\rangle$ were output as 2-HSWU-SPs; the patterns $\langle(a)(c)\rangle$ and $\langle(a)(d)\rangle$ were output as 2-HUD-SPs.

Table 2 The *su* and *swu* values of all length-2 patterns in <(*a*)>-projected databases

Pattern	$su_{SDB}(pattern)$	$swu_{SDB}(pattern)$
<(*a*)(*a*)>	90	1180
<(*a*)(*b*)>	705	2340
<(*a*)(*c*)>	1100	4040
<(*a*)(*d*)>	1280	2895
<(*a*)(*e*)>	800	2390

According to Definition 9, the mining of HSWU-SPs satisfied the downward closure property and therefore the algorithm considered all 2-HSWU-SPs for further discovering 3-HSWU-SPs and 3-HUD-SPs. In other words, the algorithm proceeded to build the $\langle(a)(b)\rangle$-, $\langle(a)(c)\rangle$-, $\langle(a)(d)\rangle$-, and $\langle(a)(e)\rangle$-projected databases and discovered 3-HSWU-SPs and 3-HUD-SPs with prefixes $\langle(a)(b)\rangle$, $\langle(a)(c)\rangle$, $\langle(a)(d)\rangle$, and $\langle(a)(e)\rangle$, respectively. For example, the algorithm calls $HUD-PrefixSpan(\langle(a)(b)\rangle, 2, SDB|_{\langle(a)(b)\rangle})$ to build $\langle(a)(b)\rangle$-projected databases (as shown in Fig. 5). The complete set of HSWU-SPs and HUD-SPs in *SDB* were then obtained**.**

Prefix	su_{SDB}	***sid***	su_{prefix}	***begin time***	***end time***	**Projected (prefix) database**	swu_{sid}
<(*a*)(*b*)>	705	20	445	45	50	(*e*,60,4)	1130
		30	260	60	65	(*e*,65,3),(*d*,65,2)	1210

Fig. 5 The $\langle(a)(b)\rangle$-projected databases

5. Experimental results

Four tests were conducted to evaluate the proposed algorithm. Moreover, we compared the proposed algorithm with the utility SPM. All the investigated algorithms were implemented in Java by using JDK7.0 and tested on an Intel core i5-4570 (3.2GHz) Windows7 system with 4 gigabytes of main memory.

A real data set called SC-POS was used in our experiments. This data set contained all the sales data of a supermarket chain in Taiwan. The sales data, called SC-POS, recorded all transactions from 18 branches between 12/27/2001 and 11/27/2002. A series of data preprocessing tasks were performed, such as deleting transactions without a member ID. Each transaction in SC-POS is a customer's shopping list, which records the purchased items, prices, and quantities.

First, we tested the difference in performance (runtime) of the proposed HUD-PrefixSpan and the conventional utility SPM. The minimum utility (*minSeqUtil*) varied from 80,000 to 120,000 for the utility SPM and the proposed HUD-PrefixSpan. In addition, the other three parameters for HUD-PrefixSpan were set as follows: (1) *ms_length* = 30, *mingap* = 7, and *maxgap* = 10; and (2) *ms_length* = 300, *mingap* = 1, and *maxgap* = 30. Fig. 6 shows that the proposed HUD-PrefixSpan outperformed the conventional utility SPM under both parameter settings, because the proposed HUD-PrefixSpan considered more constraints (i.e., additional thresholds) than did the conventional utility SPM in the mining process.

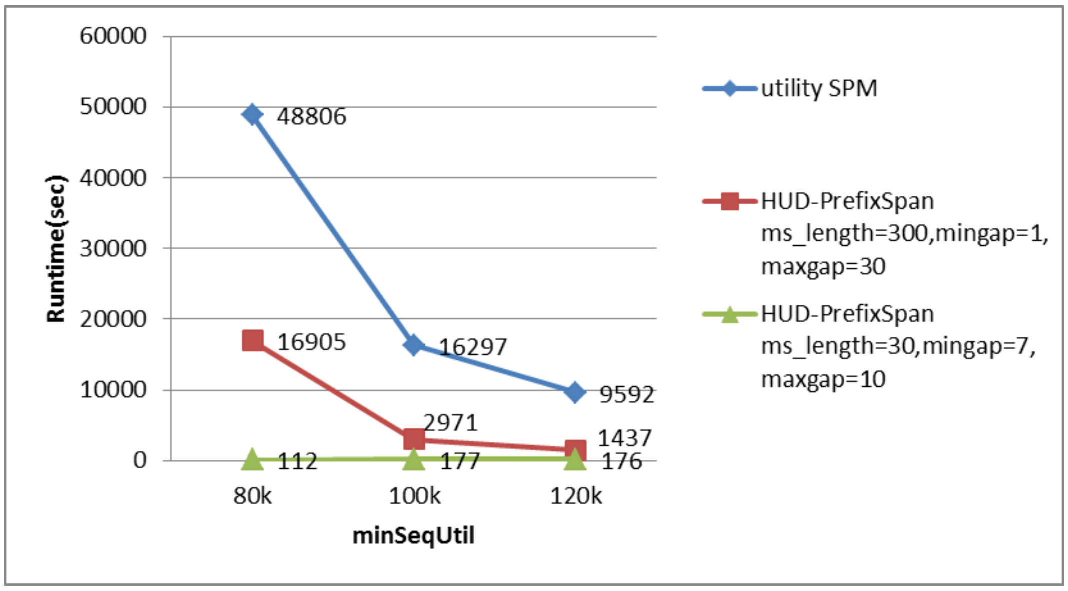

Fig. 6 Runtime vs. *minSeqUtil*

The second test investigated the difference between the accumulative numbers of patterns of the two algorithms. Fig. 7 shows that the proposed HUD-PrefixSpan generated fewer patterns than the conventional utility SPM did. This is because although more constraints (i.e., *ms_length*, *mingap*, and *maxgap* constraints) were added into the proposed HUD-PrefixSpan, only a few interesting patterns were discovered. Thus, the proposed HUD-PrefixSpan outperformed the conventional utility SPM in runtime.

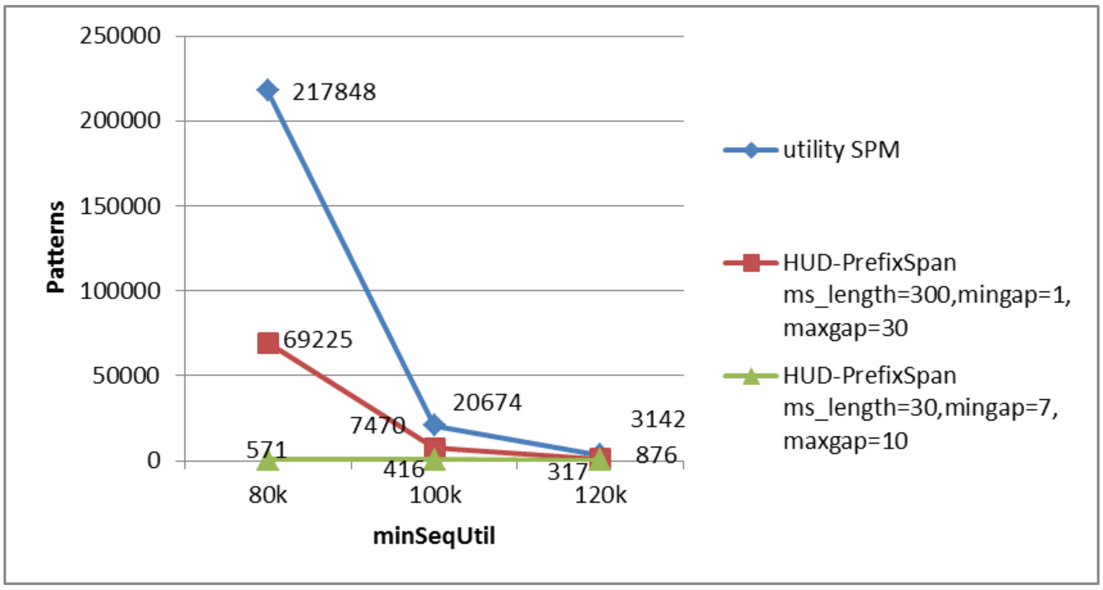

Fig. 7 Number of patterns versus *minSeqUtil*

The third test investigated the average values of patterns between two algorithms by varying *minSeqUtil* (from 80,000 to 120,000). The pattern value was calculated in New Taiwan dollars (NTD). Fig. 8 shows that the patterns obtained using HUD-PrefixSpan have higher average values than those obtained using the conventional method. The reason is that although more constraints were added into the proposed HUD-PrefixSpan, only patterns with higher values were discovered. By contrast, the conventional utility SPM discovered all patterns satisfying the *minSeqUtil* threshold, indicating that both high- and low-value patterns were retained in the mining process.

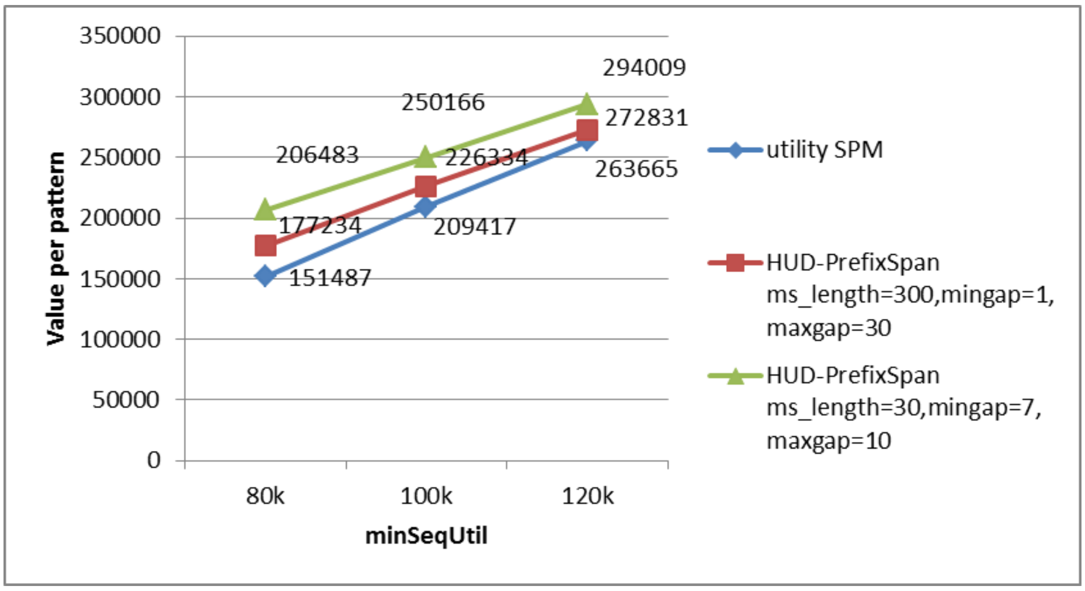

Fig. 8 Value per pattern versus *minSeqUtil*

Table 3 shows the experimental results of the two algorithms with different parameter settings. When the *minSeqUtil* threshold was set at 80,000, the HUD-PrefixSpan performed the highest with parameter settings *ms_length* = 60, *mingap* = 1, and *maxgap* = 30. Only 590 patterns were discovered and the average value per pattern was 208,120 NTD. This result was superior to that of the conventional utility SPM (217,848 patterns and 151,487 NTD per pattern). Similar results were obtained when the *minSeqUtil* threshold was set at 100,000. The conventional method discovered 20,674 patterns, and the average value per pattern was 209,417 NTD. The HUD-PrefixSpan showed superior performance under all parameter settings. Among them, the most favorable result had only 633 patterns (i.e., 3.06% of conventional patterns) and the value per pattern was 260,846 NTD. (i.e., 24.56% higher than that of the conventional pattern). When the *minSeqUtil* threshold was set at 120,000, the proposed HUD-PrefixSpan exhibited the most favorable case under the condition *ms_length* = 180, *mingap* = 1, and *maxgap* = 30. Compared with the conventional method (3,142 patterns; 263,665 NTD per pattern), only 423 patterns were discovered in HUD-PrefixSpan, and the value per pattern was 310,818 NTD.

According to the aforementioned results, the HUD-PrefixSpan can discover relatively fewer patterns but with higher values than those discovered using conventional utility SPM.

Table 3 Comparisons between HUD-PrefixSpan and conventional utility SPM

minSeqUtil	**Pattern type**	**Parameter**	**number of patterns**	**Value per pattern**
80k	utility SPM	Null	217848	151487
	HUD PrefixSpan	ms_length=60,mingap=1, maxgap=30	590	208120
		ms_length=120,mingap=1, maxgap=30	1178	197441
		ms_length=180,mingap=1, maxgap=30	9390	165315
		ms_length=240,mingap=1, maxgap=30	39039	154509
		ms_length=300,mingap=1, maxgap=30	69225	177234
		ms_length=300,mingap=3, maxgap=30	11751	161398
		ms_length=300,mingap=5, maxgap=30	1929	165732
		ms_length=300,mingap=7, maxgap=30	862	187091
		ms_length=300,mingap=9, maxgap=30	600	205929
		ms_length=300,mingap=1, maxgap=10	618	206262
		ms_length=300,mingap=1, maxgap=15	3687	195387
		ms_length=300,mingap=1, maxgap=20	12370	184936
		ms_length=300,mingap=1, maxgap=25	41702	182071
		ms_length=300,mingap=1, maxgap=30	69225	177234
100k	utility SPM	Null	20674	209417
	HUD PrefixSpan	ms_length=60,mingap=1, maxgap=30	423	250947
		ms_length=120,mingap=1, maxgap=30	478	259043
		ms_length=180,mingap=1, maxgap=30	1181	227159
		ms_length=240,mingap=1, maxgap=30	4670	213030
		ms_length=300,mingap=1, maxgap=30	7470	226334
		ms_length=300,mingap=3, maxgap=30	1711	216850
		ms_length=300,mingap=5, maxgap=30	560	234476
		ms_length=300,mingap=7, maxgap=30	442	251471
		ms_length=300,mingap=9, maxgap=30	436	249273
		ms_length=300,mingap=1, maxgap=10	461	245371
		ms_length=300,mingap=1, maxgap=15	633	260846
		ms_length=300,mingap=1, maxgap=20	1787	233198
		ms_length=300,mingap=1, maxgap=25	4377	232895
		ms_length=300,mingap=1, maxgap=30	7470	226334
120k	utility SPM	Null	3142	263665
	HUD PrefixSpan	ms_length=60,mingap=1, maxgap=30	323	293566
		ms_length=120,mingap=1, maxgap=30	339	305822
		ms_length=180,mingap=1, maxgap=30	423	310818
		ms_length=240,mingap=1, maxgap=30	726	280580
		ms_length=300,mingap=1, maxgap=30	876	272831
		ms_length=300,mingap=3, maxgap=30	395	295065
		ms_length=300,mingap=5, maxgap=30	349	295315
		ms_length=300,mingap=7, maxgap=30	341	292618
		ms_length=300,mingap=9, maxgap=30	335	290764
		ms_length=300,mingap=1, maxgap=10	359	283714
		ms_length=300,mingap=1, maxgap=15	381	309924
		ms_length=300,mingap=1, maxgap=20	434	310804
		ms_length=300,mingap=1, maxgap=25	708	271090
		ms_length=300,mingap=1, maxgap=30	876	272831

In the fourth test, the prediction performance of the two algorithms was investigated. We first separated the SC-POS data set into training and test data sets. In particular, the training data set involved transactions from 12/28/2001 to 07/19/2002; the test data set involved transactions from 07/20/2002 to 10/15/2002. Three metrics were used in this test: *precision*, *recall*, and F_1 measure. Let A and B be the sets of patterns generated from training and test data set, respectively. The three metrics can be defined as follows:

$$precision = \frac{A \cap B}{A}$$

$$recall = \frac{A \cap B}{B}$$

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Table 4 shows that the HUD-PrefixSpan considerably outperformed the conventional utility SPM in all measures, particularly in the *Precision* measure. The results showed that, compared with HUD-PrefixSpan, the conventional utility SPM generated an excessive number of meaningless patterns in the training data set, and most of these patterns never occurred in the following period (i.e., test data set). By contrast, with the consideration of both duration and gap constraints, the HUD-PrefixSpan filtered meaningless patterns and retained valuable ones.

Table 4 The prediction performance

	minsequtil = 50k		*minsequtil* = 55k	
	utility SPM	HUD-PrefixSpan	utility SPM	HUD-PrefixSpan
Precision	0.01	0.45	0.06	0.48
Recall	0.30	0.36	0.34	0.59
F_1	0.02	0.40	0.10	0.53

6. Conclusions

High-utility SPM is a useful method for discovering a customer’s purchasing behavior from large sequence databases. In this study, we defined a new type of sequential pattern, called a HUD-SP, for proposing a novel high-utility SPM technique, called HUD-PrefixSpan, which considers maximum-span-length, minimum gap, and maximum gap constraints, to discover a complete set of HUD-SPs from a sequence database.

The experimental results show that (1) the proposed HUD-PrefixSpan outperformed the conventional utility SPM in runtime, (2) the proposed HUD-PrefixSpan generated fewer patterns than the conventional utility SPM did, (3) the value of a pattern discovered by HUD-PrefixSpan is higher than that of the pattern discovered by the conventional utility SPM, and (4) the proposed HUD-PrefixSpan outperformed the conventional utility SPM in *Precision*, *Recall*, and F_1 measures.

Several concerns must still be addressed in the future. First, we considered only three constraints (i.e., maximum-span-length, minimum gap, and maximum gap constraints) in this study. Considering other constraints for high-utility SPM to discover other meaningful patterns is a worthy research direction. Moreover, we suggest that the proposed approaches be refined for discovering patterns more efficiently.

References

Agrawal, R., & Srikant, R. (1995, March). Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*(pp. 3-14). IEEE.

Ahmed, C. F., Tanbeer, S. K., & Jeong, B. S. (2010). A novel approach for mining high-utility sequential patterns in sequence databases. *ETRI journal*, *32*(5), 676-686.

Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Lee, Y. K. (2009). Efficient tree structures for high utility pattern mining in incremental databases. *Knowledge and Data Engineering, IEEE Transactions on*, *21*(12), 1708-1721.

Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Lee, Y. K. (2011). HUC-Prune: an efficient candidate pruning technique to mine high utility patterns. *Applied Intelligence*, *34*(2), 181-198.

Boulvain, F., Mabille, C., Poulain, G., & Da Silva, A. C. (2009). Towards a palaeogeographical and sequential framework for the Givetian of Belgium. *Geologica Belgica*, *12*, 161-178.

Chan, R., Yang, Q., & Shen, Y. D. (2003, November). Mining high utility itemsets. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on* (pp. 19-26). IEEE.

Chen, Y. L., & Hu, Y. H. (2006). Constraint-based sequential pattern mining: The consideration of recency and compactness. *Decision Support Systems*,*42*(2), 1203-1215.

Chu, C. J., Tseng, V. S., & Liang, T. (2009). An efficient algorithm for mining high utility itemsets with negative item values in large databases. *Applied Mathematics and Computation*, *215*(2), 767-778.

Das, R., & Turkoglu, I. (2009). Creating meaningful data from web logs for improving the impressiveness of a website by using path analysis method. *Expert Systems with Applications*, *36*(3), 6635-6644.

Erwin, A., Gopalan, R. P., & Achuthan, N. R. (2007a, December). A bottom-up projection based algorithm for mining high utility itemsets. In *Proceedings of the 2nd international workshop on Integrating artificial intelligence and data mining-Volume 84* (pp. 3-11). Australian Computer Society, Inc.

Erwin, A., Gopalan, R. P., & Achuthan, N. R. (2007b, October). CTU-Mine: an efficient high utility itemset mining algorithm using the pattern growth approach. In *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on* (pp. 71-76). IEEE.

Garofalakis, M. N., Rastogi, R., & Shim, K. (1999, September). SPIRIT: Sequential pattern mining with regular expression constraints. In *VLDB* (Vol. 99, pp. 7-10).

Hu, Y. H., Huang, T. C. K., & Kao, Y. H. (2013a). Knowledge discovery of weighted RFM sequential patterns from customer sequence databases. *Journal of Systems and Software, 86*(3), 779-788.

Hu, Y. H., Tsai, C. F., Tai, C. T., & Chiang, I. C. (2015). A novel approach for mining cyclically repeated patterns with multiple minimum supports. *Applied Soft Computing*, *28,* 90-99.

Hu, Y. H., Wu, F., & Liao, Y. J. (2013b). An efficient tree-based algorithm for mining sequential patterns with multiple minimum supports. *Journal of Systems and Software, 86*(5), 1224-1238.

Hu, Y. H., Wu, F., & Yen, T. W. (2010, June). Considering RFM-values of frequent patterns in transactional databases. In *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on* (pp. 422-427). IEEE.

Ji, X., Bailey, J., & Dong, G. (2007). Mining minimal distinguishing subsequence patterns with gap constraints. *Knowledge and Information Systems*, *11*(3), 259-286.

Lan, G. C., Hong, T. P., Tseng, V. S., & Wang, S. L. (2012, August). An improved approach for sequential utility pattern mining. In *Granular Computing (GrC), 2012 IEEE International Conference on* (pp. 226-230). IEEE.

Li, H. F., Huang, H. Y., & Lee, S. Y. (2011). Fast and memory efficient mining of high-utility itemsets from data streams: with and without negative item profits. *Knowledge and information systems*, *28*(3), 495-522.

Li, Y. C., Yeh, J. S., & Chang, C. C. (2008). Isolated items discarding strategy for discovering high utility itemsets. *Data & Knowledge Engineering*, *64*(1), 198-217.

Lin, M. Y., Hsueh, S. C., & Chang, C. W. (2008). Mining closed sequential patterns with time constraints. *Journal of information Science and Engineering*,*24*(1), 33.

Liu, J., Wang, K., & Fung, B. (2012, December). Direct discovery of high utility itemsets without candidate generation. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining* (pp. 984-989). IEEE Computer Society.

Liu, Y., Liao, W. K., & Choudhary, A. (2005). A two-phase algorithm for fast discovery of high utility itemsets. In *Advances in Knowledge Discovery and Data Mining* (pp. 689-695). Springer Berlin Heidelberg.

Masseglia, F., Teisseire, M., & Poncelet, P. (2009). Sequential Pattern Mining.

Mehzabin Shaikh, P., & Chhajed, M. G. J. (2012). Review on Financial Forecasting using Neural Network and Data Mining Technique. *Global Journal of Computer Science and Technology*, *12*(11-D).

Ngai, E. W., Xiu, L., & Chau, D. C. (2009). Application of data mining techniques in customer relationship management: A literature review and classification. *Expert systems with applications*, *36*(2), 2592-2602.

Pei, J., Han, J., & Wang, W. (2002, November). Mining sequential patterns with constraints in large databases. In *Proceedings of the eleventh international conference on Information and knowledge management* (pp. 18-25). ACM.

Pei, J., Han, J., & Wang, W. (2007). Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems,28*(2), 133-160.

Radhakrishna, V., Srinivas, C., & Rao, C. G. (2013). Constraint based Sequential Pattern Mining in Time Series Databases-A Two Way Approach. *AASRI Procedia*, *4*, 313-318.

Sandhu, P. S., Dhaliwal, D. S., Panda, S. N., & Bisht, A. (2010, April). An improvement in apriori algorithm using profit and quantity. In *Computer and Network Technology (ICCNT), 2010 Second International Conference on* (pp. 3-7). IEEE.

Seno, M., & Karypis, G. (2002). Slpminer: An algorithm for finding frequent sequential patterns using length-decreasing support constraint. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on* (pp. 418-425). IEEE.

Tseng, V. S., Wu, C. W., Shie, B. E., & Yu, P. S. (2010, July). UP-Growth: an efficient algorithm for high utility itemset mining. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 253-262). ACM.

Wang, J., Liu, Y., Zhou, L., Shi, Y., & Zhu, X. (2007). Pushing frequency constraint to utility mining model. In *Computational Science–ICCS 2007* (pp. 685-692). Springer Berlin Heidelberg.

Wu, C. W., Lin, Y. F., Yu, P. S., & Tseng, V. S. (2013, August). Mining high utility episodes in complex event sequences. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 536-544). ACM.

Wu, C. W., Shie, B. E., Tseng, V. S., & Yu, P. S. (2012, August). Mining top-K high utility itemsets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 78-86). ACM.

Yao, H., & Hamilton, H. J. (2006). Mining itemset utilities from transaction databases. *Data & Knowledge Engineering*, *59*(3), 603-626.

Yi, S., Zhang, Y., Zhao, T., Ma, S., Yin, J., Sun, H., & Chen, X. (2012). Efficient Sequential Generator Discovery Over Stream Sliding Windows. *Advanced Science Letters*, *11*(1), 437-442.

Yin, J., Zheng, Z., & Cao, L. (2012, August). USpan: an efficient algorithm for mining high utility sequential patterns. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 660-668). ACM.

Zhang, X., Gong, W., & Kawamura, Y. (2004). Customer behavior pattern discovering with web mining. In *Advanced Web Technologies and Applications* (pp. 844-853).

Springer Berlin Heidelberg.

Zheng, K., Padman, R., & Johnson, M. P. (2007). User interface optimization for an electronic medical record system.