互動式資料串流高頻封閉項目集探勘

Wan-Chao Lee
Chunghwa Telecom Co., Ltd.
leeonech@cht.com.tw

Tzer-Fu Tu
Department of Information Engineering and Computer Science, Feng Chia University, Taiwan (R.O.C.)
Ph. D. Student
P9945832@mail.fcu.edu.tw

Ming-Yen Lin
Department of Information Engineering and Computer Science, Feng Chia University, Taiwan (R.O.C.)
Associate Professor
linmy@mail.fcu.edu.tw

摘要

資料串流中探勘高頻封閉項目集，具有高複雜度因此非常具有挑戰性。過去的研究方法，大都是使用固定的最小門檻值與無法變更的時間區間。實際應用中，門檻值與時間區間應該配合使用者的需求以及資料特性而改變；並以高頻封閉項目集可還原所有高頻項目集的特性，讓系統僅須記錄高頻封閉項目集，以有效運用系統資源。

本論文中，我們提出可有效處理互動式探勘資料串流高頻封閉項目集的演算法，利用位元向量表示法，且運用扼要結構將過去的交易資料壓縮，並且在需要時擷取出來。擷取的資料可以是特定區間且可隨時改變門檻值。我們針對各種資料庫進行完整的實驗。實驗結果顯示我們所提出的方法具有處理變動支持門檻值與變動時間區間的能力並具高效率，達到減少佔用系統資源。

關鍵詞：資料串流、高頻封閉項目集、位元向量、互動式探勘。

Abstract

Mining frequent closed patterns in a data stream is very challenging for the high complexity of continuous exploration of unbounded data with bounded memory. Most previous approaches assume a fixed minimum support and an unchangeable time interval in mining frequent closed itemsets in a stream. However, the support threshold and the time interval should be flexibly specified to cope with the needs of the users and the characteristics of the incoming data in reality. Since closed frequent itemsets can be used to recover all frequent itemsets, both the space requirement and process time can be reduced significantly.

In this thesis, we propose an efficient algorithm, called VIFCI (Variable Interval Mining of Frequent Closed Itemsets), to mine all frequent closed itemsets over an arbitrary time interval. VIFCI uses an efficient bit-sequence representation of items and a summary structure to approximate past transactions within a flexible interval. However, VIFCI can interactive mining changeable support threshold and variable interval. The experiments over various parameter settings demonstrate that our approach is efficient and scalable for variable time interval mining in data streams.

Keywords: data stream, frequent closed itemsets, bit vector, interactive mining.

1. 緒論

資料串流探勘在現今是一重要研究議題，舉凡電腦、電話以及交通網路上的流量資料等，資料串流以高速連續資料持續流過，沒有流量界限且僅能檢視一次資料的特性，在傳統資料庫管理系統中無法應用在此新的資料類型上，因此傳統方法在資訊查詢和知識發現上如探勘關聯規則等，在現今需要新的機制去處理此快速的及隨時間變動的串流資料。

由於串流資料的特性，在處理串流資料時會有一些固有的挑戰。首先、由於資料串流的資料是連續的、沒有界限的、以及高速一直進來的大量資料，並沒有足夠的時間可以去重新掃瞄整個資料庫，因此每一資料元件(Element)最多僅能被檢查一次。第二、資料串流探勘方法須要在一合理時間內適應各種不同資料分佈。第三、由於資料串流高速的特性需要儘可能快速的處理，是以探勘演算法的處理速度應快於資料進來的速度，並且由近似技術輸出的錯誤應儘可能的縮小。第四、探勘串流的結果

應可立刻回應使用者請求。最後，由於串流的資料量是無止境的，而系統可用資源是受限制的例如主記憶體空間量和 CPU 的速度等，因此如何善加利用它自己本身可用資源的探勘技術是需要的。

資料串流沒有流量界限的本質並不允許將全部串流資料放入主記憶體中，因為資料串流是沒有界限的資料元件序列，即使過去的資料可以被儲存在其他延伸的儲存媒體上但通常會發生高 call-back cost，任何處理串流資料所設計的演算法通常受限於僅能掃瞄資料項目一次，此外大量的串流資料需要一扼要的資料架構來取代全部的資訊，因此為了串流管理的演算法如探勘串流演算法，會因為某些的資料項目不可避免的被丟棄，是以輸出的結果僅是相近似的而非完全正確的結果

探勘頻繁項目集 (frequent itemset mining) [1, 2, 10]是資料探勘(data mining)領域中的一個熱門研究方向。目前已有許多研究致力於探勘靜態資料庫中的頻繁項目集。然而傳統頻繁項目集探勘，往往會產生太多冗餘的頻繁項目集，導致執行效能不佳、佔用大量記憶體空間等問題。而頻繁封閉項目集 (frequent closed itemset)是頻繁項目集的精簡表示法，探勘頻繁封閉項目集不僅能縮短探勘所需的時間、節省記憶體空間，而且不會遺失頻繁項目集的資訊。

我們以圖 1 為例子，在圖左側的資料集有 5 筆交易資料，Tid 欄位為每筆交易唯一識別編號，Transaction 欄位為交易項目集，計有 5 筆交易資料，假設使用者定最小支持度=0.4，即 support count=5*0.4=2，由此參數在資料集中可找出計有 20 個高頻項目集，光是 5 筆交易資料就可找出那麼多高頻項項目集，若交易筆數一多則對記憶體將是一大負擔，但若只要找高頻封閉項目集，則只要找出紅色框所框選的 6 個高頻封閉項目集即可，可縮短執行時間，並降低記憶體空間，並且不會遺失高頻項目集資訊。

資料串流模式[12] 已經被提出來管理此新的資料型態，而在資料串流中探勘頻繁項目集[4, 5, 7-9, 13, 18, 19, 21-23]是在資料串流中找出支持度大於或等於使用者所定的最小支持度門檻值(minimum support threshold)的項目集出來。

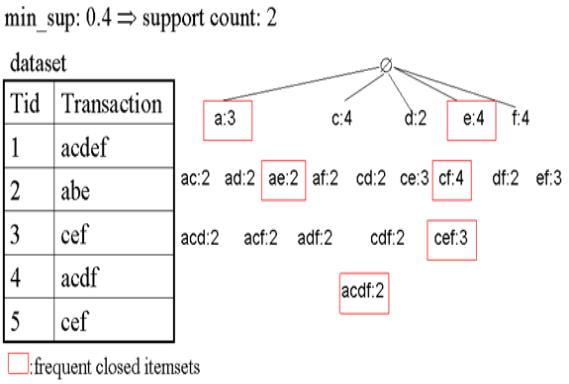

圖 1. 探勘高頻項目集例子

最近提出在資料串流中考慮交易時間區間特性的探勘頻繁項目集[12]，包括有三種模式，第一、地標視窗模式(Landmark Window)，為考慮資料串流從基準點開始到目前為止的全部歷史資料，此方法之缺點為沒有辦法反應項目集的時間特性。第二、滑動視窗模式(Sliding Window)，使用者僅考慮資料串流中在最近視窗中的交易資料，視窗中存在著固定大小的交易量而沒有時間性質限制。第三、時間衰減視窗模式(Time-fading Window)，從某一基準點開始至目前為止的全部資料作探勘有點類似地標模式，但隨著時間流逝，過時的出現次數會逐漸被忽略，即依時間的先後順序給一權重(weight)值，年代愈久遠的權重值愈低，其重要性愈低，權重值表示方式以遞減來表示，強調新進資料的重要性。

在決定探勘資料串流的模式之後，在資料串流中嚴謹的探勘到指定時間間隔的頻繁項目集，然而大多數演算法在資料串流中探勘頻繁項目集僅接受指定一個最小支持度(minimum support)值，且最小支持度值在此方法的探勘期間不能被改變，例如，假如最小支持度變更為較小的值時，這些演算法無法獲得那些被丟棄的交易資料的統計值，所以獲得的結果是不正確的，因此最小支持度門檻值應該是可變的去適合使用者所需要的，此外前面的研究方法在探勘處理期間的時間間隔是不能改變的。

例如，使用者也許想知道從過去 6 個月到最近 3 個月或在最後一年的六月份的資料；這些演算法不能夠滿足此需求，在實際上，使用者也許僅對不同的時間間隔、但相同區間的比較、變動 support 值有興趣等。

在過去的研究議題中大部份的演算法僅針對固定時間間隔來探勘資料串流，而一個新的演算法可以處理由使用者來指定變動時間間

隔，即當使用者變更時間間隔時，在資料串流中探勘變動時間間隔的演算法不可避免的需要裡面具有彈性間隔可近似於過去交易資料的扼要架構，並以高頻封閉項目集可還原所有高頻項目集的特性，讓系統僅須記錄高頻封閉項目集，以滿足資料串流上必須有效運用系統資源的需求。

因此我們提出在資料串流中允許一變動的 support 值並在隨意時間區間中探勘高頻封閉項目集的演算法 VIFCI (Variable Interval mining of Frequent Closed Itemsets)，VIFCI 演算法是為了在串流的交易資料中探勘高頻封閉項目集，VIFCI 演算法使用一種壓縮架構去儲存跨越時間區間的串流交易扼要資訊，所以可以查詢到不同的時間區間的資訊，此扼要架構稱為 Powered Bit Vector 簡稱 PBV，是一有效的位元序列(bit-sequence)表示[3] 具有彈性的間隔可近似於過去的交易資料，廣泛的實驗並調整不同的時間間隔顯示 VIFCI 演算法是一高效率及近線性。

2. 相關研究

在傳統資料串流探勘，Buffer-Trie-SetGen [22]是一種 Lossy Counting based 的演算法，從項目集合的組成串流資料中計算出高頻量，此演算法採用批次處理去探勘串流資料。它將整個串流資料分成許多批次單元並將這些批次單元載入至主記憶中，而每一批次單元的大小由使用者視電腦主記憶體可用的大小來設定。

FP-stream[9]演算法為在資料串流中在任何時間點在有限制大小的視窗範圍內的交易集合中找出全部高頻樣式集，FP-stream 演算法支援在資料串流中維護和計算從新進來的資料中動態的更新所有高頻樣式集，此方法從 FP-growth 架構中延伸來探勘 time-sensitive 的樣式，具有近似的支持度保證，此作者為了每一樣式在多重時間粒度中逐漸維護 tilted-time windows。

Moment 演算法[8]是在資料串流中使用滑動窗模式找出封閉高頻項目集，而其中使用 Closed Enumeration Tree (CET)架構，是一種在記憶體中的資料結構，來從動態的滑動視窗中找出高頻項目集。

CET 架構的重要特性只要視窗大小是合理的大小並且在串流中的分佈傾向並非太戲劇性的，則大部份項目集並不會改變其狀態(從頻繁到非頻繁或從非頻繁到頻繁)，假如項目集一改變狀態，此變動必須從邊界節點更改起，即整個樹架構均須變動是一個重要的限制。

NewMoment 演算法[19]使用交易敏感滑動視窗模式從線上資料串流中找出頻繁封閉項目集，Moment 演算法[8]實驗的結果顯示在 CET 中的邊界是穩定的所以更新的成本是較小的。然而為了頻繁封閉項目集 Moment 必須維護很大量的 CET 節點，有關 CET 節點和頻繁封閉項目集的比例大約為 30 比 1，故若有一大量的頻繁封閉項目集，Moment 在記憶體的使用上將是無效率的。NewMoment 演算法提出一種 bit－sequence 來代表 items 可在滑動視窗時降低時間和記憶體的需求。

VSMDS (Variable Support Mining for Data Streams)演算法提出在[13]，是用來解決在資料串流中變動支持度探勘問題，VSMDS 使用 synopsis vector (縮寫為 SYV)來建構與被丟棄的交易資料相近似的資料及 PFI-tree (Potential Frequent Itemsets tree) 用來記錄可能的高頻項目集的 support counts 值。每次一筆新的交易批次進來 PFI-tree 作更新及將交易批次資料插入 SYV 架構中，直到最後一筆交易批次資料，因此 PFI-tree 用來記錄每一項目支持度大於某一門檻值以上的所有項目集。

VSMTP 演算法 (Variable Support Mining for Time-Sensitive Patterns over Data Stream)提出在[11]，是用來解決在資料串流中時間敏感高頻樣式的變動支持度探勘，為了提供符合於使用者指定時間間隔的樣式，是有關到該時間間隔的意義，因此必須有一新的有效方法去記錄此摘要資訊，VSMTP 採用延伸 VSMDS 的架構在當使用者降低 minimum support 值時具有近似 support 值的保證去探勘時間敏感度樣式，而一個傾斜概要向量表之建構是當發生有需要過去的交易資料時可以經由此傾斜式時間表格來作維護和統計。

為了回應時間敏感查詢，VSMTP 修正 VSMDS 演算法的 SYV 架構，稱為傾斜式概要向量表，是一種概要架構，設計在不同的時間區段使用一種彈性距離門檻值來與過去交易資料相近似，此距離門檻值決定壓縮概要向量的次方，並僅在當指定新的支持度門檻值小於使用在 PFI-tree 中的門檻值時，再從傾斜式概要向量表中執行掃瞄相關資料作業。

GruFI 演算法(Grouped Synopsis Vectors for Frequent Itemsets Mining) 提出在[24]，是為了在資料串流中探勘變動 support 高頻項目集，GruFI 演算法提出是為了改進 VSMDS 演算法在概要向量的更新與壓縮，GruFI 與概要向量

的架構與 VSMDS 相似。

GruFI 演算法使用一種群組概要向量來將過去交易資料分類成不同的概要向量，在壓縮階段僅動態搜尋可能的概要向量，Length index table 是提供來快速反覆推敲到概要向量以提昇搜尋速度，另一機制是當新的 bucket 進來時概要向量儘可能快速的讓概要向量尺寸變小，提供來快速找出相符交易資料。

VIFI 演算法 (Variable Interval mining of Frequent Itemsets) 提出在[17]，在資料串流中探勘有關任意時間區間變動支持度的高頻項目集，過去的研究方法，從資料串流中找尋高頻項目集，大部分都是使用一個固定不變的最小門檻值與無法變更的時間區間，VIFI 演算法提出是可改變最小門檻值與時間區間來探勘高頻項目集。

VIFI 演算法主要執行有 2 階段，第 1 階段為交易附加階段，使用一個扼要結構將過去的交易資料以近似值的壓縮方式作壓縮，並且僅在需要時再擷取出來使用。第 2 階段為互動式探勘階段，可依據使用者要求的特定區間來擷取資料且可隨時改變門檻值來探勘高頻項目集。

3. 問題定義

設 $\Psi = \{\alpha_1, \alpha_2, ..., \alpha_\gamma\}$ 代表所有項目(items)的集合、其中 α 代表項目，假如有一項目集合(itemset) e 有 k 個項目，用 $(\beta_1, \beta_2, ..., \beta_\kappa)$ 來表示，我們稱之為 k-項目集合(k-itemset)，而此 k-項目集合的長度為 k，用符號 |e| 來表示；為了不要有遺漏發生假設在項目集合(itemset)內的項目(items)是按字母順序來排序的。資料串流 (data stream) 用 DS = $\{t_1, t_2, \ldots, t_c, \ldots\}$ 來表示，是個無止境的、一直不斷進來的交易序列(transaction sequence)，其中每一筆交易 t_i 代表具有唯一交易記錄編號的項目集合，t 為Ψ的子集合($t \subseteq \Psi$)；設 t_c 為目前最後一筆進來的交易，稱為目前交易 (current transaction)，而資料串流的長度則為從某一基準點開始至目前為止交易的數量，某一筆交易 t_i 包含一個項目集合 $e(e \subseteq t_i)$，而此項目集合 e 在某一指定的時間間隔 τ_i 到 τ_j 的支持度(support)用 $\sup(e)\tau_i{:}\tau_j$ 來表示，乃代表在資料串流中從開始時間 τ_i 到結束時間 τ_j 之區間包含項目集合 e 的交易數量。

而在開始資料串流探勘前須先由使用者指定一個最小支持度門檻值(minimum support threshold) ms∈(0,1]，即 ms 的值介於 0 到 1 之間，而在任何時間點使用者可以輸入來改變此最小支持度的門檻值，故此門檻值可形成一最小支持度的序列稱之為支持度序列(support sequence)；另設 ms_c 為在交易 t_c 時的最小支持度稱之為目前最小支持度(current minimum support)；若有一項目集合 e 在時間區間 τ_1 到 τ_2 出現的次數即支持量(support count)大於或等於目前最小支持度(ms_c)乘以在時間區間 τ_1 到 τ_2 的交易數量，可用符號如下表示：$\mathrm{SUP}(e)\tau_1{:}\tau_2 \geq ms_c$，則稱此項目集 e 為高頻項目集(frequent itemsets)，且當一個高頻項目集不存在有任合超項目集(superset)有著與本身相同支持度值時，則稱此項目集 e 為高頻封閉項目集，此目的是在任何指定的時間區間內以任何指定的最小支持度值來找出高頻封閉項目集。

此問題陳述在減輕資料串流中發現高頻封閉樣式共通的限制，使用者可指定任何時間區間來取代資料串流的全部歷史資料，並且可在任意時間改變最小支持度。

4. VIFCI 演算法

在這一段，我們敘述研究的演算法稱之為 VIFCI (Variable Interval mining of Frequent Closed Itemsets)，在資料串流中探勘有關任意時間區間變動支持度的高頻封閉項目集

4.1 VIFCI 演算法概要

VIFCI 演算法是具有變動支持度門檻值及任意時間區間為了在資料串流中探勘高頻封閉項目集，圖 2 是 VIFCI 演算法分為 2 個步驟執行的概要圖：(a) 為每一筆交易資料附加到 PBV (Powered Bit Vector) 資料結構內使用壓縮和附加的方法。(b) 為萃取出使用者要求的所有頻繁封閉項目集。

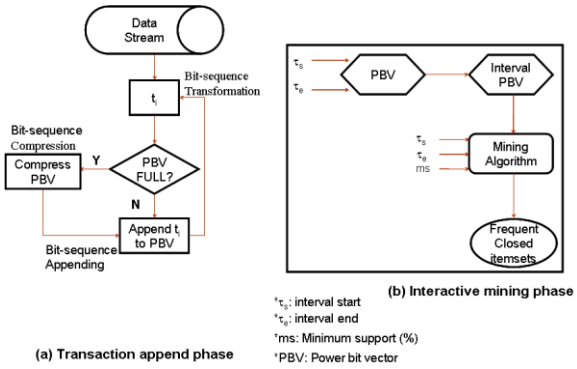

圖 2. VIFCI 演算法架構概要
(a)交易資料附加階段 (b)互動式探勘階段

在一筆新的交易資料 ti 進入時，VIFCI 需要先檢查 PBV 架構內資料是否已滿了，若尚未滿則附加 ti 到 PBV 內，否則須壓縮 PBV；在 PBV 資料結構內使用對數的方法去記錄資料串流中扼要交易資訊，詳述如 4.3 節。

當使用者指定項目集 e 從時間區間τ_s到τ_e的 support 值時，則 $sup(e)_{\tau s.\tau e}$ 在時間區間τ_s和τ_e對應到 PBV 內的時間間隔，此探勘演算法使用 bit-vector “and”操作來計算 weighted support 值，因此可以找出在任何時間區間任何 support 值的高頻封閉項目集。

4.2 使用 Bit-Vector 來表示項目

在 VIFCI 演算法中，為了建構每一項目 X 出現在第幾筆交易中而採用 bit-sequence 表示法為 Bit(X)，最初始 Bit(X)設為空集合，假如項目 X 出現在第 i 筆的交易中，則第 i 筆的 Bit(X)設為 1，否則設為 0。

例如，在圖 3 中，假設視窗(window)大小為 6bit，並且僅考慮 6 筆交易資料：<T1, (abc)>, <T2, (bcd)>, <T3, (ad)> ,<T4, (bd)> , <T5, (abcd)>及<T6, (abc)>因為項目 a 出現在 t1、t3、t5、t6 筆交易中，項目 a 的位元-向量(bit-vector)表示 Bit(a) =101011，同樣道理 Bit(c)=110011, 及 Bit(d)=011110。

Tid	Transaction
t1	abc
t2	bcd
t3	ad
t4	bd
t5	abcd
t6	abc

Bit(a)	Bit(b)	Bit(c)	Bit(d)
1	1	1	0
0	1	1	1
1	0	0	1
0	1	0	1
1	1	1	1
1	1	1	0

圖 3. 使用 6 筆交易資料串流的例子

4.3 Powered Bit Vector：傾斜式表格表示近似交易資料

為了表示項目而採用的位元-向量是一有效的表示法，然而此方法會快速耗盡可用的記憶體空間，所以新的架構應限制空間的配置來處理資料串流沒有界限的交易資料特性，而為了維持類似每一近似交易資料因此提出 powered bit vector 簡稱 PBV 的傾斜-時間表格(tilted-time table)，PBV 是由包括 Bit(α_1), Bit(α_2), …, Bit(α_n)等所有項目的位元-向量所組成，其處理過程有 2 步驟：壓縮原有交易資料及附加新進入的交易資料。

4.3.1 壓縮舊的交易資料

在此步驟為壓縮舊的交易資料，而壓縮的方法有很多種方式，本研究採用的方式為 2 選 1 的方式，用來將項目的位元-向量表內若有 w 位元量壓縮成為 w/2 位元量，而其 2 選 1 的壓縮方式又可分為下列 4 種方法：(1) 假如第 1 個偶數位元 tn 是 1，則設為 1，否則設為 0；然後接著檢查下一個偶數位元直到所有偶數位元均檢查過為止。(2) 假如第 1 個奇數位元 tn-1 是 1，則設為 1，否則設為 0；然後接著檢查下一個奇數位元直到所有奇數位元均檢查過為止。(3) 將第一位元 tn-1 與第二位元 tn 作 AND 邏輯運算作為結果。(4) 將第一位元 tn-1 與第二位元 tn 作 OR 邏輯運算作為結果。

本論文採用的壓縮方式為第(1)種壓縮方法，例如，假設視窗大小 w 為 4，在圖 4 中項目 a 的位元-向量 Bit(a)=1010，經壓縮後 Bit(a)=0000，因為第 1 個偶數位元與第 2 個偶數位元均為 0 ，同理 Bit(b)=1100，Bit(c)=1000, 及 Bit(d)=1100，壓縮後所空出的 w/2 是預留作為下一步驟轉換用的，並定義一壓縮次數(縮寫為 cp)來記錄壓縮次數。

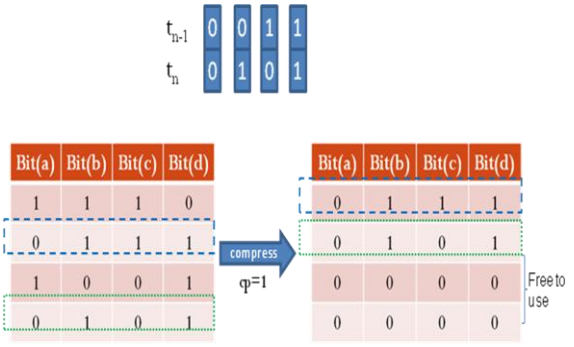

圖 4. 壓縮舊交易資料例子

4.3.2 附加新進入的交易資料

在壓縮舊交易資料後，進行第 w/2+1 筆位元由檢查新進來交易 ti 項目的位元-向量的位元，假如 ti 包含 X 則設第 w/2+1 位元為 1，否則設為 0。

例如，在圖 5 的 Power Bit Vector(簡稱 PBV)表中項目 a 的位元-向量在壓縮舊的交易資料後 Bit(a)=0000，因為進來的交易 t5 包含項目 a

所以設 w/2+1 位元的 Bit(a)為 1，因此 Bit(a)從 0000 變更為 0010，同理 Bit(b)從 1100 變更為 1110，Bit(c)從 1000 變更為 1010，Bit(d)從 0100 變更為 0110。

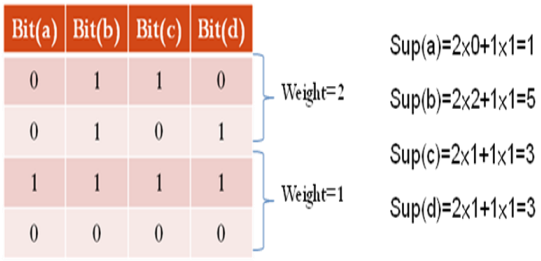

圖 5. 附加新進入交易資料例子

4.4 使用 Bit-Vector 計算 Weighted Support

計算單一項目的權重支持度(weighted support)方法如下，在圖 5 的 Bit(a)在 PBV 內是 0010，即在 t5 內包含項目(a)，而 Bit(a)的權重支持度=2(weight)x0+1(weight)x1=1，Bit(c)的權重支持度=2(weight)x1+1(weight)x1=3；同理項目的位元-向量的概念可以被延伸至項目集(itemsets)，例如，在圖 5 的 2-itemset bc 的位元-向量，Bit(bc)在 PBV 內是 1010，意即在 PBV 內的交易 t1、t2 及 t5 內包含項目集(bc)。

計算一項目集的權重支持度(weighted support)過程如下：假如有 2 個 k-itemsets X 和 Y，它們分別對應的位元-向量為 Bit(X)和 Bit(Y)，另(k+1)-itemset 的位元-向量 Z=X∪Y 可以由 Bit(X)和 Bit(Y)作 AND 邏輯運算並且 Bit(Z)可以得到適當的權重。在圖 5 中 2-itemset bc 在 PBV 中的位元-向量可以由項目 b 和 c 的位元-向量作 AND 邏輯運算，即 Bit(b)=1110 和 Bit(c)=1010 作 AND 邏輯運算後可獲得 Bit(bc)=1010；另權重支持度的計算可以由 2(weight)x1+1(weight)x1=3 獲得。

4.5 VIFCI 演算法

VIFCI 演算法在 Output_on_demand 由 3 個部份所組成：(1) 由計算 weighted support 來產生所有 1-itemsets。(2) 由使用 AND 操作按字母順序以 depth-first 來產生 large itemsets。(3) 當 support count 大於或等於 ms × (τ_e-τ_s)，則為高頻項目集，且當一個高頻項目集不存在有任合超項目集有著與本身相同 support count 值時則輸出所有高頻封閉項目集。

本段詳細說明 VIFCI 演算法探勘過程，詳如圖 6，當有一筆交易 ti 進來時，會先由計算 w+cp*w/2 來檢查 PBV 內是否滿了，若尚未滿則將交易 ti 附加進入到 PBV 內；若 PBV 內已滿則壓縮 PBV，以騰空部分 PBV 空間給後來的交易可附加進入用。

接著說明 Append 子程式的過程，詳如圖 7，從某一筆交易 t_i 中可得到它的所有項目資料，並更新它對應的位元值進入到 PBV 內，但若要將所有資料串流的資料全部放入電腦的主記憶體內則會需要非常大的空間，是以當 PBV 內空間滿了時我們必須採用壓縮方式將空間壓縮，以騰空 1/2 的 PBV 空間來容納新進來的交易資料。

而 Output_on_demand 子程式(圖 8)執行位元 AND 運算及權重操作來產生頻繁項目集，並檢查每一子節點(n_i')的 support 值與母節點(n_i)的 support 值若相等，則記住 n_i 是頻繁封閉項目集並加入至 hash table 內，最後輸出。

Input：t_i: a new incoming transaction in the data stream DS; ms: minimum support; t_s: interval start; t_e: interval end
Output：Frequent closed itemset FCI
Let PBV be empty and cp=0 /* PBV : power bit vector; cp: compression count */
while t_i comes
  If (i <w+cp*w/2) /* w: window size */
    Append(PBV, t_i); /* append t_i into PBV */
  Else /* PBV Full */
    for(n=1;n<w;n+=2) /*compress PBV*/
      test PBV(bit(n)) and reset bit position
      end-for
      cp++; /* cp: compression count */
    Append(PBV, t_i); /* append t_i into PBV */
end-while
Output_on_demand(PBV, ms, τ_s, τ_e)/*output the FCI*/

圖 6. VIFCI 演算法

Subroutine Append(PBV, t_i)
Input: PBV: powered bit vector; t_i: a new incoming transaction in the data stream DS
Output: PBV_i
for each item in PBV do
  If ($t_i \subseteq$ item) set bit
end- for
return PBV

圖 7. Append 子程式

Subroutine Output_on_demand(PBV, ms, τ_s, τ_e)
Input: PBV: powered bit vector; τ_s: interval

start; τ_e: interval end
Output: FCI: Frequent closed itemsets
L_1={ bitmaps of large 1-itemset generate by PBV}
for{k=2;$L_{k-1}\neq\varnothing$;k++} do
Remove those elements in L_1 which are not included in any itemset of L_{k-1}
C_k=bitwise AND Bit(L_{k-1}) and Bit(L_1)
L_k={c ∈ C_k | bitmap weighted count of c ≥ ms ×(τ_e-τ_s)};
If a child $n_{I'}$ of n_I such that
support($n_{I'}$)=support(n_I) then
mark $n_{I'}$ an intermediate node;
else
mark n_I a closed node;
insert n_I into the FCI
end-for
Output = FCI

圖 8. Output_on_demand 子程式

5. 實驗結果

在實驗上我們使用的硬體設備為 Intel Core 2 Duo processor 2.0GH，2GB DDR2 RAM，而軟體為在 Microsoft Windows XP 作業系統環境上測試，本論文提出的 VIFCI 演算法是執行在 C++ STL，編譯是採用 Visual C++ .NET 編譯器。此外為了研究壓縮的效果，我們使用 2 個測量指標值 precision 及 recall 來評量演算法的準確性；其中設正確的高頻項目集為 T，而由演算法所輸出的高頻項目集為 O，recall 為相關成功擷取查詢的分數比率 $= \frac{|T \cap O|}{|T|}$，而 precision 為依據使用者所需資訊所擷取出來的分數比率 $= \frac{|T \cap O|}{|O|}$ 詳如圖 9，另 F-Measure 為 precision 和 recall 的權重調和 $= \frac{2(P*R)}{P+R}$。測試資料使用 IBM Data Generator[1] 資料庫產生器來產生資料庫，該產生器在產生資料時有數個參數可供設定，如表 1 所示。其中資料集 T10I4D100K 的意義如下：D100K 為資料串流有 100,000 筆交易，T10 為每筆交易的平均長度為 10，I4 為潛在最大頻繁項目集平均長度為 4。

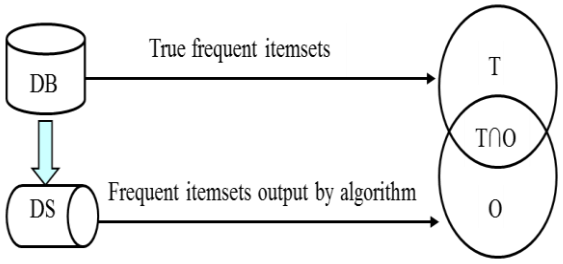

圖 9. Precision 和 Recall

表 1 資料庫的參數設定

參數	描述	參數值
|D|	在資料庫中交易的數量	100k~5000k
|T|	每筆交易平均長度	10, 12, 15
|I|	潛在最大頻繁項目集平均長度	4, 5, 6
|N|	項目個數	1000

5.1 在執行時間的效能和記憶體使用分析

在此實驗上，我們研究在 T10.I4.D100K 資料集的總執行時間、記憶體使用量、precision 和 recall。此探勘的時間間隔為資料串流的全部歷史資料，視窗大小 w=32768，其中最小支持度門檻值(minimum support threshold)變化從 1.0%到 0.1%。在圖 10 中 VIFCI 演算法的執行時間比 Non compress 慢，那是因為 VIFCI 執行增加了額外的壓縮時間；在圖 11 中顯示記憶體使用以 MB 為單位，VIFCI 記憶體使用約 7MB，但 Non compress 使用了超過 15 MB，因為 VIFCI 使用固定大小的記憶體(32kbit×1000items=4MB)來記錄 Powered Bit Vector (VIFCI-PBV)及小部分為探勘所需的記憶體。在表格 2 中 VIFCI 有高的正確率及 precision 和 recall 的平均值大於 87%以上，此外，當最小支持度門檻值降到 0.1%時，precision 和 recall 降到 53.04%和 62.14%，那是因為頻繁封閉項目集數量有 14640，然而從表格 3 中，我們變化不同的 PBV 大小(16K, 32K, 及 64K)來檢查 precision 和 recall 的效果，當 PBV 大小增加時，即當使用者有足夠的記憶體空間時，precision 和 recall 也增加，是以 VIFCI 此特性可有效地改進正確率。

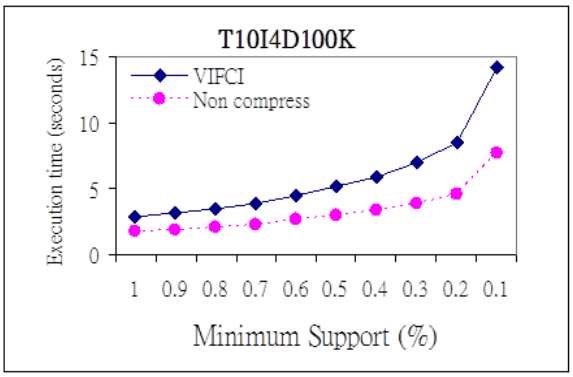

圖 10. 探勘 T10I4D100K 的總執行時間

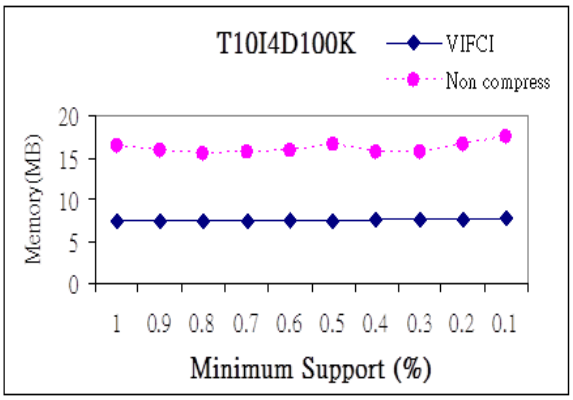

圖 11. 探勘 T10I4D100K 記憶體的使用

表 2. 探勘 T10I4D100K 的 precision 和 recall

Minsup(%)	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	Average
Precision(%)	95.65	95.56	95.43	97.31	96.67	96.31	95.39	87.34	64.13	53.04	87.683
Recall(%)	94.62	95.79	95.22	95.73	96.13	96.31	95.39	93.99	84.49	62.14	90.981
F-Measure(%)	95.14	95.67	95.32	96.52	96.4	96.31	95.39	90.54	72.91	57.23	89.143

表 3. 變化 WB 大小的 Precision 和 Recall 值

Windows size	32KB	64KB	128KB
Precision (%)	87.68	94.83	100
Recall (%)	90.98	96.07	100

接著改變資料串流的資料集，平均每筆交易的大小增加從 10、12 到 15，潛在最大頻繁項目集平均長度從 4 到 6，一般資料集此參數一改變，在最小支持度門檻值同前面狀況下，則將會產生更多及更長的頻繁項目集。圖 12 及圖 13 顯示在資料集 T12I6D100K 的執行時間和記憶體的使用，圖 14 及圖 15 顯示在資料集 T15I4D100K 的執行時間和記憶體的使用。在前面的效能研究中，我們可以歸納出 VIFCI 演算法執行在所有合成的資料集中有好的效能及穩定的記憶體使用，然而當記憶體相當足夠的情形下，使用者可以調高 PBV 的大小以獲得較高的正確率。

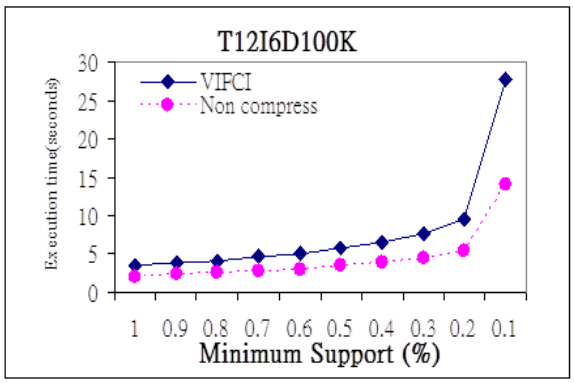

圖 12. 探勘 T12I6D100K 的總執行時間

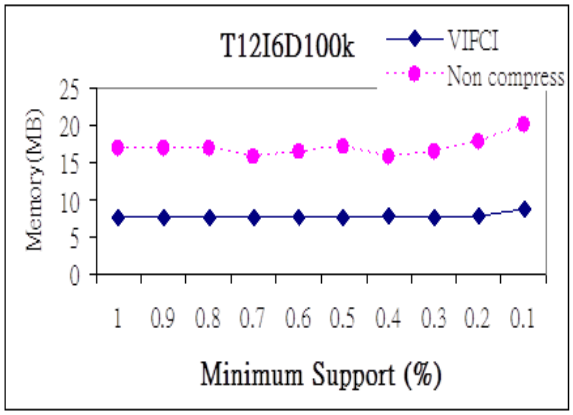

圖 13. 探勘 T12I6D100K 記憶體的使用

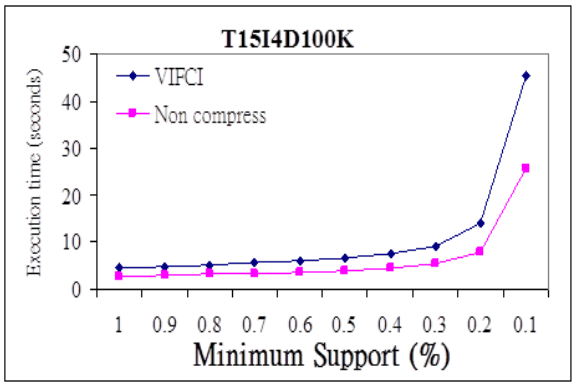

圖 14. 探勘 T15I4D100K 的總執行時間

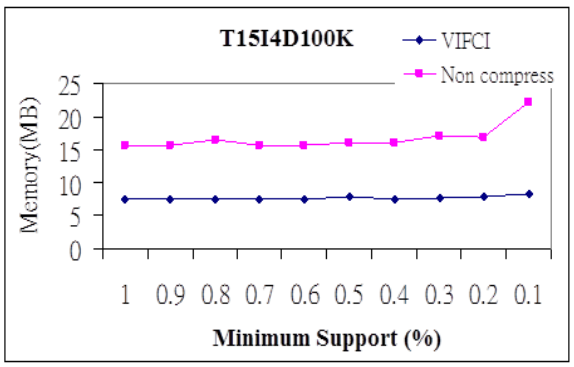

圖 15.探勘 T15I4D100K 記憶體的使用

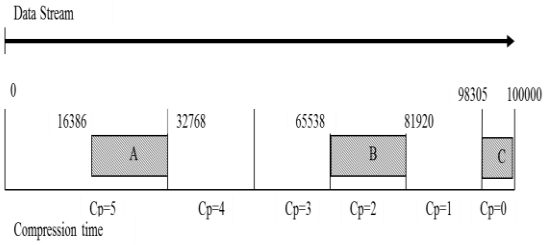

圖 16. 探勘 T10I4D100K 的不同區間

5.2 變動時間間隔分析

在此實驗中，我們研究在 T10I4D100K 資料集上，變化時間間隔及最小支持度門檻值來分析 PBV 壓縮技術的效能分析，在圖 16 中顯示區間 A 是從 16386 到 32768 均為壓縮 5 次，區間 B 是從 65538 到 81920 並全部為壓縮 2 次，區間 C 是從 98305 到 100000 均無壓縮。在 3 個區間提供實驗數據變化，其中區間 A 與區間 B 的區間大小均相同，詳表格 4 中顯示總執行時間、記憶體使用量、precision、 recall、pattern 數量(#pattern)及 pattern 的最大長度(MaxLen)等，其中執行時間和記憶體使用量在 3 個區間均相接近，當壓縮次數增加時則 precision 和 recall 降低，因為我們的 VIFCI 演算法採用的是從近似過去的交易資料中探勘頻繁封閉項目集，其中資料集為 T10I4D100K，預設最小支持度門檻值為 0.4%，及最大 pattern 的長度為 10，此外變動最小支持度結果相類似，我們可以產生具有良好準確率的近似值回應及使用者可以在任何時間改變最小支持度門檻值或時間間隔。

表 4. 不同區間的實驗

Interval	Total time (sec.)	Memory (KB)	Precision (%)	Recall (%)	#Pattern	MaxLen
A	4.719	7640	57.46	80.68	945	10
B	5.828	7556	91.42	94.95	699	10
C	6.516	8112	100	100	1192	10

5.3 Scalability 擴展性

在此實驗中，我們研究在 VIFCI 演算法的擴展性分析，在圖 17 顯示 VIFCI 演算法在資料集從 T10I4D100K 到 T10I4D1000K 及最小支持門檻值為 0.6%的條件下的 scale-up，其中實驗在資料集為 T10I4D100K 時 PBV 設為 32K，當資料集為 T10I4D200K 時 PBV 設為 64K，當資料集從 T10I4D300K 到 T10I4D600K 時 PBV 設為 128K，當資料集從 T10I4D700K 到 T10I4D1000K 時 PBV 設為 256K，此乃由自動調整 PBV 大小來改進正確性；另如圖 18 顯示 VIFCI 演算法在資料集從 T10I4D1000K 到 T10I4D5000K 及最小支持度定為 0.6%條件下的擴展性，此實驗可看出 VIFCI 演算法的擴展性在執行時間上是近似線性的。

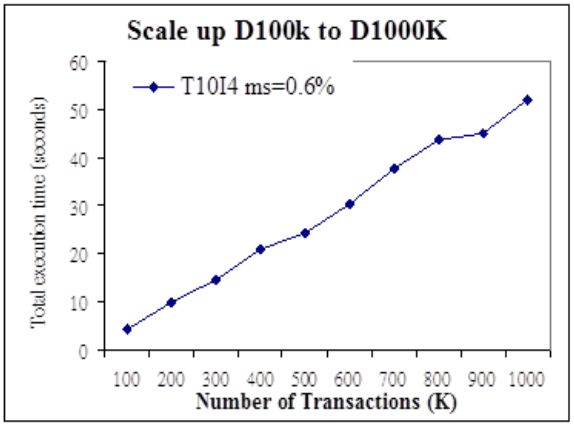

圖 17 .VIFCI 從 T10I4D100K 到 T10I4D1000K 的 scale-up

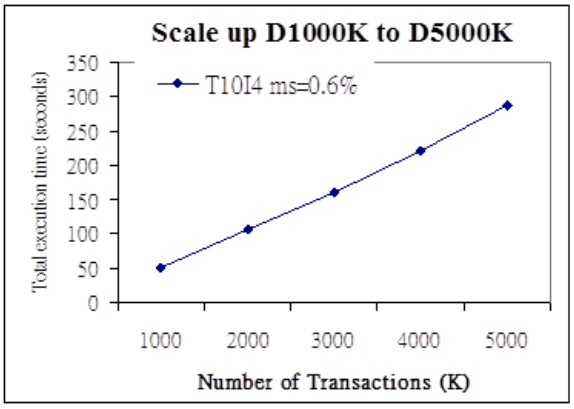

圖 18 VIFCI 從 T10I4D1000K 到 T10I4D5000K 的 scale-up

6. 結論

近年來由於資訊發達與科技進步，企業面臨著資訊爆炸所帶來的壓力與衝擊。面對資料量不斷的成長且需要快速回應使用者的環境下，傳統的資料庫探勘已經無法應用在此環境下，因此利用資料串流的概念作資料探勘的問題被日益重視。

在本論文中我們提出 VIFCI 演算法，具有變動支持度和時間區間在資料串流中探勘頻繁封閉項目集。VIFCI 為了可以快速分配可用記憶體而使用 bit-sequence 來表示 item；另為了近似於每一筆交易資料而採用 power bit vector (簡稱 PBV)來維護 tilted-time table，因此新的結構可以限制配置空間來處理資料串流

具有無止境特性的交易資料，並利用頻繁封閉項目集可以表示所有頻繁項目集的優點使系統僅需要記錄頻繁封閉項目集，減少系統佔用資源空間，廣泛的實驗證實 VIFCI 具有變動支持度和時間區間可有效率的探勘頻繁封閉項目集及具有良好的線性。

本論文的壓縮方式為水平式壓縮(即水平列的壓縮)，係針對交易的筆數來做壓縮，在後續的研究方面，可採用垂直式的壓縮方式，針對項目(item) 來做壓縮，可產生使用者有興趣的項目集。

7. 致謝

感謝逢甲大學專案「量化高效益項目集探勘之研究」支援本研究。

參考文獻

[1] Agrawal, R., Imielinski, T., and Swami, A., "Mining association rules between sets of items in large databases," ***In Proceedings of the ACM SIGMOD International Conference on Management of Data***, pages 207-216, May 1993.

[2] Agrawal, R. and Srikant, R., "Fast algorithm for mining association rules," ***In Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)***, pages 487-499, 1994.

[3] Ayres, J., Gehrke, J., Yiu, T., and Flannick, J., "Sequential pattern mining using a bitmap rRepresentation," ***In Proc. of the 8th ACM International Conference on Knowledge Discovery and Data Mining (KDD)***, pages 429-435, July 2002.

[4] Chang, J. H. and Lee, W. S., "Decaying obsolete information in finding recent frequent itemsets over data stream," ***IEICE Transaction on Information and Systems***, Volume E87-D, No. 6, June 2004.

[5] Chang, J. H. and Lee, W. S., "estWin: Online data stream mining of recent frequent itemsets by sliding window method," ***Journal of Information Science***, Volume 31, Issue 2, pages 76-90, April 2005.

[6] Charikar, M., Chen, K., Farach-Colton, M., "Finding frequent items in data streams," ***Theoretical Computer Science***, Volume 312, Number 1, pages 3-16, January 2004.

[7] Cheng, J., Ke, Y., and Ng, W., "Maintaining frequent itemsets over high-speed data streams," ***In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2006)***, pages 462-467, April 2006.

[8] Chi, Y. and Wang, H., "Moment: Maintaining closed fFrequent itemsets over a stream sliding window," ***In Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)***, pages 59-66, Brighton, United Kingdom, 01-04 November 2004.

[9] Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P. S., "Mining frequent patterns in data streams at multiple time granularities," ***In Proceedings of the NSF Workshop on Next Generation Data Mining***, 2002.

[10] Han, J., Pei, J., and Yin, Y., "Mining Frequent patterns without candidate generation," ***In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data***, Volume 9, Issue 2, pages 1-12, 1999.

[11] Hwang S. K., "***Efficient Mining of frequent patterns in data streams with variable support thresholds***," Master Thesis, Feng Chia University, 2006.

[12] Jiang, N. and Gruenwald, L., "Research issues in data stream aAssociation rule mining," ***ACM SIGMOD Record***, Volume 35, No. 1, March 2006.

[13] Jia, L., Wang, Z., Lu, N., Xu, X., Zhou, D., Wang, Y., "RFIMiner: A regression-based algorithm for recently frequent patterns in multiple time granularity data streams," ***Applied Mathematics and Computation***, Volume 185, Number 2, pages 769-783, February 2007.

[14] Jiang, N., Gruenwald, L., "CFI-Stream: mining closed frequent itemsets in data streams," ***In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining***, pages 592-597, August 2006.

[15] Koh, J. L. and Shin, S. N., "An approximate approach for mining recently frequent itemsets from data streams," ***In Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery***, DaWaK 2006, pages 352-362, Krakow, Poland, September 2006

[16] Leung, K. S., Khan, I., "DSTree: A tree structure for the mining of frequent sets from data streams," I***n Proceedings of the 6th***

IEEE International Conference on Data Mining (ICDM 2006), Hong Kong, pages 928-932, December 2006.

[17] Liu, G. P., "***Interactive mining of frequent itemsets in a data stream***," Master Thesis, Feng Chia University, 2008.

[18] Li, H. F., Lee, S. Y., and Shan, M. K., "An efficient algorithm for mining frequent itemsets over the entire history of data streams," ***In Proceedings of the First International Workshop on Knowledge Discovery in Data Streams***, pages 20-24, Pisa, Italy, September 2004.

[19] Li, H. F., Ho, C. C., Kuo, F. F., and Lee, S. Y., "A new algorithm for maintaining closed frequent itemsets in data streams by incremental updates," ***In Proceedings of IEEE International Workshop on Mining Evolving and Streaming Data (IWMESD 2006)***, Hong Kong, December 2006.

[20] Lin, M. Y. and Lee, S. Y., "Interactive sequence discovery by incremental mining," ***Information Sciences: An International Journal***, Volume 165, Issue 3-4, pages 187-205, 2004.

[21] Lin, M. Y., Hsueh, S. C., and Hwang, S. K., "Variable support mining of frequent itemsets over data stream using synopsis vectors," ***In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2006)***, pages 724-728, April 2006.

[22] Manku, G. S., Motwani, R., "Approximate frequency counts over data streams," ***In Proceedings of the 28th VLDB Conference***, pages 346-357, Hong Kong, China, August 2002.

[23] Raïssi, C., Poncelet, P., Teisseire, M., "Towards a new approach for mining frequent itemsets on data stream," J***ournal of Intelligent Information Systems stream***, Volume 28, Issue 1, pages 23-36, February 2007.

[24] Wang, C. Y., "***Enhanced variable support mining for frequent itemsets over data streams,***" Master Thesis, Feng Chia University, 2007.

[25] Wong, C. W. and Fu, W. C., "Mining top-K frequent itemsets from data streams," ***Data Mining and Knowledge Discovery***, Volume 13, Number 2, pages 193-217, September 2006.

[26] Yu, X., Chong, Z., Lu, H., Zhang, Z., and Zhou A., "A false negative approach to mining frequent itemsets from high speed transactional data streams," ***Information Sciences***, Volume 176, Number 14, pages 1984-2015, July 2006.