在多重資料串流環境中之跨串流序列樣式探勘

趙景明
東吳大學資訊管理學系 教授
chao@csim.scu.edu.tw

林彥廷
東吳大學資訊管理學系 碩士生
ms9614@csim.scu.edu.tw

摘要

資料串流是一個高速率到達的資料流，探勘結果需要即時性更新，而多重資料串流為多個串流同時到達，因此同時處理之資料量必定更龐大。序列樣式探勘為在資料序列中探勘具有時間順序的頻繁序列樣式。近年來，由於資料量的快速成長，即時性知識需求提高，因此探勘多重資料串流環境已成為重要研究議題之一。先前的研究中，只能處理單一時間單一資料並不足以因應多重資料串流環境的變化，所探勘出來的序列樣式可能會存在於不同的串流中但被視為同一個串流之序列樣式。因此，本論文提出 IAspam 演算法，除了能夠處理單一時間多個資料，並且能夠探勘出跨串流序列樣式。它將串流資料轉換成位元映射方式進行探勘，以減少執行時間與記憶體使用量。

關鍵詞：資料串流、序列樣式、資料探勘、跨串流

1. 前言

在資訊發達的時代，知識儼然成為一個不可或缺的條件。在商業上，知識的獲取使企業不管在客戶關係管理或供應鏈管理上都成為關鍵成功因素；在服務業上，知識的獲取讓提供服務的商家可以針對顧客的個人習性或喜好，使顧客得到最貼切的服務。因此，如何從資料中挖掘出有價值的知識是非常重要的，而資料探勘(Data Mining)的目的就在於協助人們從大量資料中挖掘出有意義的樣式或知識。

序列樣式(Sequential Patterns)探勘的概念是由 Agrawal and Srikant [1]在 1995 年所提出，主要的目的是找出隨著時間或是特定順序而經常發生的序列樣式。例如:在網頁存取上，主要是探勘網頁的使用者瀏覽樣式來預測使用者下一個可能存取的網頁，以便進行預先擷取，加快網頁讀取速度；在醫學方面，運用序列樣式探勘，尋找慢性疾病患者未來可能發生的慢性疾病及其他重大併發症，以預防病人因這些疾病再次住院；手機業者可從客戶流失的前一個月發現，該手機用戶出現較特殊的行為，以預測該客戶是否想轉換到別家電信公司，進而針對這類客戶提供補救方案以留住客戶。

在生活中的各種行為都有可能產生串流資料(Stream Data)，例如: 網路流量、股市交易、網頁瀏覽行為等。而這些資料串流具有以下特性[4, 18]:（1）輸入資料量的大小是沒有限度的（2）記憶體的大小有限（3）輸入資料只被處理一次（4）資料是快速流入的（5）系統無法控制資料量的流入。由於資料串流環境與傳統資料庫特性不同，資料庫探勘演算法無法適用於資料串流環境中，因此，近幾年有許多演算法[10, 14, 15, 21, 23, 28]被提出用來探勘隱藏在資料串流中不同類型的資訊，資料無需儲存於資料庫即可對資料做探勘。如此便可大量降低資料庫空間的使用率與成本，但由於資料串流特性的限制下，探勘過程必定會遇到一些難題，而這些難題正是本研究所要克服的。資料串流可區分為單一資料串流與多重資料串流(Multiple data streams)。在多重資料串流

環境中探勘序列樣式是一個重要的研究議題，同時探勘多個串流資料的情況必定比單一串流資料更加困難，探勘效率也是必須注重的環節。

然而在多重資料串流環境中，隨著現實資料串流環境的變化，資料量將變得龐大並且資料複雜度將會越來越高，但是現有的演算法只能處理單一時間單一資料並不足以因應多重資料串流環境的變化，另一方面，現有的演算法所探勘出來的序列樣式可能會存在於不同的串流中但被視為同一個串流之序列樣式。針對上述問題，本論文提出IAspam (Incremental Across-streams Sequential Patterns Mining)演算法，以因應更複雜的多重資料串流環境。它能夠處理單一時間多個資料，並且能夠探勘出跨串流(Across streams)序列樣式。

所謂跨串流頻繁序列樣式(Frequent Sequential Patterns)意指在一個串流中的頻繁項目集和另一個串流中的頻繁項目集為序列關係，而這個頻繁序列樣式分別跨越於兩個不同串流，此頻繁序列樣式即稱為一個跨串流序列樣式(Across-streams Sequential Patterns)，其主要目的為探勘頻繁序列樣式在串流與串流之間的串流關聯性。

在多重資料串流環境的序列樣式探勘過程中，計算支持度和找到頻繁序列樣式的效率上都必須要顧慮。因此為了提升探勘效率，本論文使用位元映射表示法（Bitmap Representation），將每一個項目（Item）轉換成位元方式，增升計算支持度(Support)的效率以及快速找出頻繁序列樣式，以減少執行時間和記憶體使用量。

2. 文獻探討

本研究主要是探討在多資料串流環境中，使用位元表示法探勘跨串流的序列樣式，因此本章將針對研究的相關文獻進行探討。首先針對序列樣式探勘方法進行探討與分析，再探討位元映射表示法以及資料串流環境探勘的技術發展。

2.1 序列樣式探勘

在探勘序列樣式的方法中，Apriori 演算法是最初被設計用來找出關連規則，而 Agrawal and Srikant [1]所提出的 AprioriAll 演算法是由 Apriori 演算法所改良，其為探勘序列樣式的第一個演算法，方法大致與 Apriori 相同，但因為在探勘序列樣式上其效率不佳，因此該作者在後續研究提出了比 AprioriAll 演算法效率較佳的 GSP (Generalized Sequential Patterns)演算法[33]，其基本精神是使用前一階段所探勘的大型序列來產生下一個階段的大型序列，並在產生候選序列(Candidate Sequences)的步驟上改善了處理的速度。Zaki [37]提出 SPADE (Sequential Pattern Discovery using Equivalence classes)演算法，此演算法使用簡單的序列結合(Sequence Join)操作去找出序列樣式，以垂直資料庫（Vertical Database）資料排序模式取代了前面所描述方法[1, 33]的平行資料庫（Horizontal Database），其探勘效能上更優於 GSP；Han and Pei [17]便針對為了減少候選序列的產生數量，提出了 FreeSpan 演算法，使用投影序列資料庫(Projected Sequence Database)限制搜尋和子序列的成長；雖然 Apriori 可以大大減少項目組合的數量，但其依然會遇到一個問題就是當遇到一個大型序列資料庫或是當序列樣式較多或較長時，其效率亦不彰。因此 Pei et al. [30]提出 PrefixSpan (Prefix-projected Sequential Pattern Mining)演算法，其主要是在序列樣式探勘中探勘出前序投射(Prefix Projection)，序列資料庫被遞迴地投射成較小的投射資料庫(Projected Databases)集合，並且在各個投射資料庫中，序列樣式的增長都只探索

局部的頻繁片段。PrefixSpan 可探勘出完整的樣式並且可大大減少候選序列的產生，前序投射亦減少投射資料庫的大小，這使得探勘變的更有效率。

在交易資料庫中探勘序列樣式的過程中，顧客的交易資料會隨著時間不斷的產生，為了保留已探勘的頻繁序列樣式，在新的交易資料進入時能夠將新的頻繁序列樣式與舊的直接做結合(Join)或是新增(Append)。漸進式探勘(Incremental Mining)的方式可保留舊的探勘資料並加入新的探勘資料，Lin and Lee [27]提出了 FASTUP (Fast Sequential Pattern Update Algorithm)演算法，以漸進式探勘交易資料庫，交易資料是以顧客編號為主鍵，隨著新交易資料的產生，其會依據顧客編號將新資料新增至交易資料庫中，此演算法主要是在交易資料庫中找出頻繁序列樣式，一開始會先找出頻繁的 1-序列(1-Sequences)，再產生候選鍵，以支持度作篩選找出頻繁序列樣式；Chen et al. [12] 提出 IncSPAN (Incremental Mining of Sequential Patterns)演算法，以漸進式探勘序列樣式，在探勘過程中能夠減少搜尋空間，將新資料加入到資料庫中，隨著資料庫不斷更新，探勘序列樣式也能不斷更新，由於新進資料的樣式可能與資料庫中的樣式符合又或是其子序列（Subsequences），依此對於新進資料尋找序列樣式的過程便能減少搜尋時間。

探勘頻繁序列樣式中，為了減少記憶體的使用率，避免產生大量的候選序列，便會找出最大序列樣式或是封閉式序列樣式，並且刪除多餘的頻繁樣式，Yan et al. [36]提出了 CloSPAN (Mining Closed Sequential Patterns)演算法，其引用 PrefixSPAN 的概念來探勘封閉式的序列樣式，並且提出搜尋空間刪減(Search Space Pruning)的方法以減少多餘重覆的序列樣式，使用雜湊表(Hash Table)來儲存樹節點索引以增快其搜尋樹的速度。封閉式探勘序列樣式的觀念提出之後，陸續有研究針對封閉式做更有效率的探勘，Wang and Han [34]提出 BIDE (BI-Directional Extension)演算法，其亦是封閉式序列樣式探勘演算法，但其在探勘過程中，針對序列樣式的刪減提出了不同的作法，而 BIDE 演算法最主要是能夠在探勘封閉式序列樣式時不會保留住非封閉式序列，其序列延伸方法可分為前向延伸(Forward Directional Extension)和後向延伸(Backward Directional Extension)，前向延伸是用來生長 prefix 樣式並且檢查 prefix 樣式的結束點(Closure)，後向延伸是同時用來檢查 prefix 樣式的結束點和刪減搜尋空間，在搜尋空間上提出了 BackScan 搜尋空間刪減方法，有效節省搜尋空間，不保留候選序列雖然可以大大節省記憶體空間，但是在探勘時間的效率來說，卻不一定優於 CloSPAN。

在 BIDE 演算法的提出後，Chang et al. [6]提出 IMCS (Incremental Mining of Closed Sequential Patterns)演算法，除了引用 BIDE 的節省搜尋空間的概念之外，也加入了漸進式的概念，以漸進式探勘封閉式序列樣式，探勘過程中其是由他們所設計的封閉式序列樹(Closed Sequence Tree)來保留封閉式序列樣式，以不同類別的節點儲存不同狀態的序列樣式，在探勘時間上優於上述之演算法，而在記憶體的使用量卻較大。BIDE 與 IMCS 分別在時間與空間上做些取捨，但欲同時兼顧時間與空間的效率是一件困難的事。搜尋空間的刪減策略上，Xu et al. [35]提出 $SPAM_{SEPIEP}$ 演算法，其中包含兩個新的刪減策略 SEP (Sequence Extension Pruning)和 IEP (Item Extension Pruning)，分別使用 S-list 和 I-list 已儲存 SEP 和 IEP 所產生的延伸候選集合，而 $SPAM_{SEPIEP}$ 演算法即是使用這兩個刪減策略將 SPAM 演算法改善的更有效率。

2.2 使用位元表示法探勘序列樣式

前述之演算法的演進雖然在效率上提升不少，不過面對龐大資料量或是大量候選序列的處理仍需不斷改善其探勘效率，在探勘過程的資料處理上，將每個項目(Item)轉換成以位元(Bit)方式表示，可以增快其在探勘大量項目集合和計算支持度的效率，Ayres et al. [3]提出了 SPAM (Sequential Pattern Mining)演算法，其產生候選項目集所需兩個步驟：項目延伸步驟(Itemset-extension Step, I-step)和序列延伸步驟(Sequence-extension Step, S-step)，這兩個步驟使每個項目集合使用位元映射表示法，將交易資料庫中的項目集合轉換成 1 和 0，並直接以邏輯運算 AND 的方式快速執行 I-step 和 S-step，依此方法可增進候選項目集產生的效率。

將位元映射的概念應用在探勘序列樣式的方法提出後，陸續有學者引用此方法提出了不同的演算法，Zhenglu and Masaru [38]提出 LAPIN-SPAM (LAst Position INduction Sequential PAttern Mining)演算法，其可以從一個大型資料庫取得所有的頻繁序列樣式，與前述之演算法不同之處在於 PrefixSpan 演算法[30]是用 S-Matrix 來掃描投射資料庫(Projected Database)、SPADE 使用結合來計算支持度、SPAM 使用 ANDing 來產生候選項目，LAPIN-SPAM 則是以最後位置(Last Position)的位元是否小於目前 prefix 位置來判斷項目使否會出現在目前的 prefix 中，依此方法可以在探勘過程中減少搜尋空間；Aseervatham et al.[2]提出 bitSPADE 演算法，此演算法結合位元映射以及 SPADE 演算法的概念，將垂直資料庫的資料轉換成位元表示，其探勘時間以及記憶體的使用率亦優於 SAPDE 演算法；Ho et al. [18]所提出的 IncSPAM 演算法，其與 IncSPAN 的相同點在於皆是使用漸進式概念，不同點在於 IncSPAM 是以位元映射表示法做探勘，其是將 SPAM 演算法加入漸進式的概念，以交易資料為主鍵並將資料轉換成位元矩陣，在探勘過程中，可以迅速的計算其支持度並且以位元的映射方式快速找到序列樣式，探勘序列樣式趨向以漸進式的方式探勘，IncSPAM 依此特性並結合位元映射的特性，使其在探勘效能上能優於 IncSPAN。由上述可知，以位元映射表示法探勘序列樣式可以增快其探勘效率。

2.3 資料串流環境探勘序列樣式

前一節所提到的序列樣式探勘其資料來源皆是從一個大型資料庫中探勘資料，當大型資料庫的探勘技術仍在發展的階段時，Oates and Cohen [29]對多資料串流架構提出了 MSDD (Multi-Stream Dependency Detection)演算法，搜尋多串流資料的資料結構，Golab and Ozsu [16]提出了資料串流的特性與資料串流的管理，因此便有研究趨向於在資料串流環境中探勘序列樣式，資料串流的探勘技術尚未成熟，這方面的研究也陸續有更完善的演算法被提出，例如：在資料串流環境探勘頻繁項目集、探勘頻繁序列樣式、最大序列樣式或是封閉式序列樣式等等，適用在資料庫探勘的演算法也被改良成適用於資料串流環境的演算法，而單資料串流環境與多資料串流環境之探勘技術皆有演算法被提出，其中多資料串流環境探勘仍少有學者研究。

Chang and Lee [7]提出在資料串流環境中探勘頻繁項目集的遞減機制(Decay Mechanism)，依資料串流環境的特性，舊的探勘資料長久未在資料串流中出現，因此必須遞減其支持度以確保探勘資料的正確性。一般的刪減策略會在滑動視窗往後移時會刪減掉最早的交易資料，Chang and Lee [9]所提出的 estDec 演算法是藉著遞減舊交易資料的權重，在線上資料串流中找到最近的頻繁項目集，在探勘資料串流中的交易資料過程中，資料的權重是會逐漸的隨著時間而減少，亦也利用此刪減策略來提出能夠提高記憶體使用效率與時

間效能。Chen et al. [11]提出 MILE 演算法，其是建立在 PrefixSPAN 演算法上，以 Prefix 的概念在多資料串流環境中探勘序列樣式，主要是避免重複資料的搜尋以及快速探勘新的序列樣式，由於資料串流的特性限制，探勘過程中的記憶體使用率是一個需要克服的難題，而 MILE 達到了記憶體使用和時間效率的平衡。在前一節所提及的 IncSPAM 演算法[18]亦是在資料串流環境中使用滑動視窗來擷取串流資料，再轉換成以顧客位元向量列陣(Customer Bit-vector Array with Sliding Window, CBSW)探勘頻繁序列樣式。因為是在資料串流漸進式探勘，而久未被更新的頻繁序列樣式需要加入權重值(Weight Value)的概念以遞減支持度，此概念引用於[7]所提出的遞減機制，以方程式(1)來遞減項目集合支持度，其中 d 為遞減率(Decay Rate)，b 為 Decay-base，h 為 Decay-base-life。在資料串流環境探勘最大序列樣式(Maximal Sequential Patterns)中 Raissi et al. [31]提出了 SPEED (Sequential Patterns Efficient Extraction in Data Streams)演算法，以快速的刪減策略(Deleting Strategy)在新的資料架構保留頻繁序列樣式並找出最大化頻繁序列樣式；Li and Chen [25]所提出新的刪減策略，可以減少其運算成本，在框架移動時，利用項目集合辨識度去辨識出新的項目集合與舊的項目集合使否大於使用者所設的最低門檻值，進而執行刪減策略。

$$d=b^{-(1/h)} \quad (b>1, h\geqq 1, b^{-1}\leqq d<1) \qquad (1)$$

探勘頻繁序列樣式的過程中必定會先找到頻繁項目集，因為序列樣式是由項目集所延伸出來的，因此這兩者是息息相關的。而在探勘頻繁項目集的研究上，亦可分為最大頻繁項目集(Maximal Frequent Itemsets) [5, 22]和封閉式頻繁項目集(Closed Frequent Itemsets)，Li et al. [26]則是提出一個新的一次性通過(Single-pass)演算法，稱為 DSM-FI (Data Stream Mining for Frequent Itemsets)演算法，其是在資料串流中探勘頻繁項目集，使用 CFI-tree (Candidate Frequent Itemset Trees of Item-suffixes)以 suffix 的方式找出頻繁項目集; Chang and Lee [8]則是著重在使用滑動視窗方法依照交易先後順序擷取串流資料，找出頻繁項目集；Chi et al. [13]提出 MOMENT 演算法，亦是使用滑動視窗擷取串流資料，再以所設計的 CET(Closed Enumeration Tree)儲存頻繁項目集，並找出封閉式序列樣式，由於其在 CET 中留住太多的頻繁項目集，造成記憶體的浪費，因此 Ho et al. [19]提出改良的 NewMOMENT 演算法，只有在 NewCET 保留住封閉式頻繁項目集，因此改善了記憶體的使用率。Ren and Li [32]所提出的 FRFI 演算法也是利用滑動視窗探勘最近的頻繁項目，此演算法使用頻繁項目表(FIT)儲存頻繁項目，再將此頻繁項目加入至頻繁項目雜湊表(FIH)中，其以樹狀結構表示，並在此雜湊表中做刪減和更新，以保留住頻繁項目。此外亦有利用圖形來探勘序列樣式，Li and Chen [24]所提出的 GraSeq 演算法亦是利用方向性加權圖形(Directed Weighted Graph)架構來探勘序列樣式，以方向性表示序列的先後順序，其亦以遞減率來刪除長時間未被更新的序列樣式。Ezeife and Monwar [20]所提出的 SSM 演算法則是使用 D-list、PLWAP tree 和 FSP-tree 三種類型資料結構來處理在資料串流中探勘序列樣式的複雜性，D-list 為儲存項目的頻繁次數、建置 PLWAP tree 去批次探勘序列樣式並漸進式儲存至 FSP-tree 中並維護頻繁序列樣式。

3. 研究方法

在多重資料串流環境中，隨著現實資料串流環境的變化，其資料量將變得龐大並且資料複雜度將會越來越高，但是現有的演算法只能

處理單一時間單一資料並不足以因應多重資料串流環境的變化，然而基於商業因素考量，業者可能需要找到每個地點所產生串流之間的關聯性來對使用者提供更適當的服務。因此，本研究提出ASPAMDAS (Across-streams Sequential Pattern Mining in Multiple Data Streams)方法來解決這幾個問題。ASPAMDAS主要分為資料取樣與漸進式探勘兩個階段。首先，在資料取樣階段中，利用交易導向移動框架(Transaction-sensitive Sliding Window)模式對串流資料取樣，取樣資料以顧客交易資料為切割依據。然後，在漸進式探勘階段中，將移動框架中的顧客交易資料轉換為位元映射來尋找跨串流序列樣式，以提升探勘效率。接著再針對新舊跨串流序列樣式進行更新或刪除以產生精確的跨串流序列樣式探勘結果。圖 1 為ASPAMDAS的整體流程。

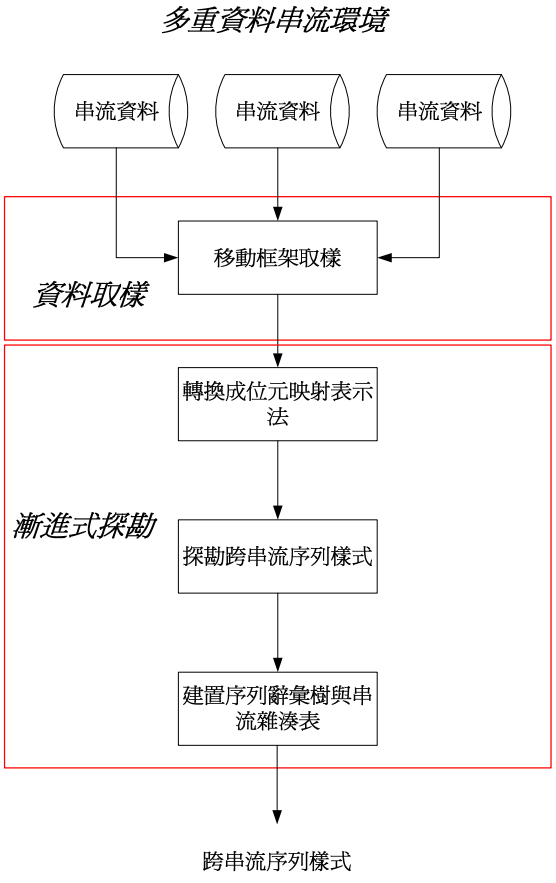

圖 1 ASPAMDAS 的整體流程

3.1 多重資料串流環境

ASPAMDAS 所採用的多重資料串流環境描述如下：

- 每一個資料串流在一個時間點會產生一筆顧客交易資料。
- 資料串流中的顧客交易資料包括顧客序號與交易資料。
- 顧客交易資料的先後順序以資料進入時間點區分。
- 每個時間點的各串流資料會被同步處理。
- 由於串流數目的增多，資料量會比單一資料串流更為龐大。
- 一個串流集合表示為 $S=\{s^1, s^2, ..., s^j, ..., s^m\}$，這裡 m 表示串流的數量，$s^j$ $(1 \le j \le m)$代表一個資料串流。

3.2 資料取樣

在資料串流環境中，由於即時性知識需求提高和資料串流特性的限制，相對地，越新的資料便顯得越重要，因此我們使用移動框架來擷取最新幾筆串流資料。移動框架只會留住最近 N 筆交易資料，N 即是框架的大小，如圖 2 所示，資料取樣是採用交易導向的移動框架模式，意即在每次處理完框架內的交易資料後，框架會往後移動一筆交易資料的方式進行取樣。

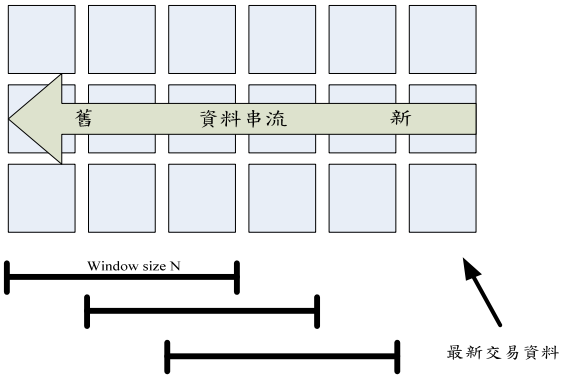

圖 2 資料串流環境之移動框架位移

在多重資料串流環境中，串流與串流之間的資料可能存在序列關係，因此會有序列樣式之項目來自於不同串流資料的跨串流序列樣式。本研究提出的 IAspam 演算法目的是在多重資料串流環境中探勘跨串流序列樣式。如圖 3 所示，同一時間點會產生三個串流資料，每個串流資料會有一個顧客的交易資料，串流 S 亦可視為顧客交易的地點或位置，Wi $(1 \le i \le n)$代表移動框架的第 i 個框架。圖 3 顯示框架大小 N=3，在處理完 W1 的交易資料後，會處理 W2 框架內的交易資料，並依此移動框架的方式，將可以漸進的方式探勘串流資料。

3.3 漸進式探勘

漸進式探勘(Incremental Mining)在串流資料不停的流進流出的情況下，所探堪出的頻繁序列樣式會隨著新串流資料的產生而不斷的更新頻繁序列樣式，更新狀態如以下情況：

- 新資料帶來新項目且為頻繁序列樣式
- 頻繁序列樣式更新之後依然是頻繁序列樣式
- 頻繁序列樣式有可能變成非頻繁序列樣式

了解多資料串流環境特性之後，我們必須針對其探勘方法做初始假設與定義:

- 假設 I = {$i_1, i_2, \ldots, i_k$}是一個所有項目的集合，I 的子集合(Subset)便稱為一個項目集合(Itemsets)。
- 序列(Sequences)，s = <$t_1, t_2, \ldots, t_m$> (ti ⊆ I) 是一個有順序性的序列，假設 α = <$a_1, a_2, \ldots, a_m$>是另一個序列 β = <$b_1, b_2, \ldots, b_n$>的子序列(Sub-sequence)，則可稱為 α ⊆β，如果 α≠β 則為 α⊂β。
- 顧客序列表(Customer Sequence)，CS = {$s_1, s_2, \ldots, s_n$}, 為一個暫時儲存序列的資料庫，將每一個顧客的所有交易根據交易時間排序所得到的的結果。α 的支持度就是在 CS 中包含 α 的序列個數，support(α) = {s|s ∈ CS，α⊆s}。
- 最小支持度度門檻值(Minimum Support Threshold)簡稱為 min_sup，為使用者自行定義的門檻值，用來辨別序列樣式是否頻繁，例如:sup(a)≧min_sup，即表示項目 a 是頻繁項目。
- 頻繁序列樣式(Frequent Sequential Patterns)；如果一個序列樣式大於或等於最小支持度門檻值，即為頻繁序列樣式，反之，即為非頻繁(Infrequent)序列樣式。

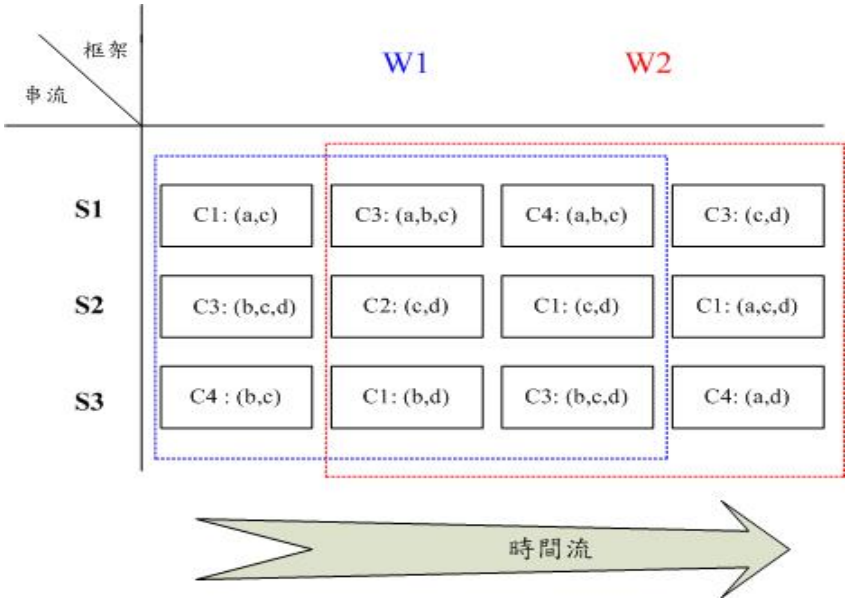

圖 3 多資料串流環境範例

3.3.1 位元映射表示法

為了降低探勘的時間與空間耗費，IAspam 採用位元映射表示法，將移動框架擷取的多筆交易資料轉換成以顧客編號(CID)為主鍵的顧客序列表，如表 1 所示。顧客序列表是一個暫存資料，主要目的為儲存移動框架所擷取的資料以作為探勘序列樣式的資料來源，其資料量的儲存大小，會依照移動框架所擷取的資料量而定。表 1 為圖 3 的框架 W1 轉換而成之顧客序列表。因為移動框架大小設為 3，因此在第一次擷取資料後會將三個串流中的前三筆項目集合暫時存入顧客序列表中，以<S_i,α>表示，其中 S_i 為串流資料編號，α 為項目集，例如：<S1,(a,c)>即表示(a,c)有出現在 S1 串流中。在 CID2 的第一欄以及第三欄表示為空集合即表示此顧客分別在這兩個時間點沒有交易動作的發生。

表 1　W1 之顧客序列表範例

CID	顧客交易序列		
1	**<S1, (a,c)>**	**<S3, (b,d)>**	**<S2, (c,d)>**
2	<∅>	**<S2, (c,d)>**	<∅>
3	**<S2, (b,c,d)>**	**<S1, (a,c,d)>**	**<S3, (b,c,d)>**
4	**<S3, (b,c)>**	<∅>	**<S1, (a,b,c)>**

在探勘完 W1 的資料之後，移動框架便會往後移動一筆項目集合，此時 W2 框架中便可擷取到一筆新的資料，如表 2 所示。在 CID 為 1、2、4 中可看到分別有新的項目集<S2,(a,c,d)>、<S1,(c,d)>、<S3,(a,d)>出現，這三個項目集即為移動框架位移後(W2)所擷取的新資料，而 CID3 未出現新的項目集資料，即表示在第二個框架擷取資料時，這位顧客尚未有新的交易出現在圖 3 範例中。

表 2　W2 之序列資料庫範例

CID	顧客交易序列		
1	**<S3, (b,d)>**	**<S2, (c,d)>**	**<S2, (a,c,d)>**
2	**<S2, (c,d)>**	<∅>	<∅>
3	**<S1, (a,c,d)>**	**<S3, (b,c,d)>**	**<S1, (c,d)>**
4	<∅>	**<S1, (a,b,c)>**	**<S3, (a,d)>**

我們將第一個框架 W1 所擷取的交易項目集合轉換成位元映射(Bitmap)，即將有存在的項目設為 1，沒有存在的項目設為 0，如圖 4 所示，a、b、c、d 代表項目，在 CID 1 中，項目 a 欄位所示之(1,0,0)表示為在表 1 中 CID1 的資料序列中只在第一個時間點存在 a 項目，轉換成位元時以(1,0,0)表示。第一個 1 代表第一個時間點存在項目 a 的交易資料，第二個 0 和第三個 0 則代表第二、三個時間點沒有出現項目 a 的交易資料。

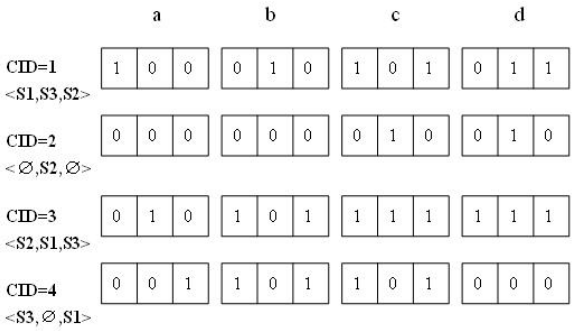

圖 4　W1 顧客位元映射矩陣範例

為了要探勘跨串流序列樣式，首先在轉換成位元矩陣時，必須紀錄串流編號。其目的在於探勘跨串流序列樣式時，必須以串流為主，分別計算各串流之序列支持度，以便能辨別出每個項目分別是出現在哪一個串流中，找出頻繁序列樣式屬於哪些串流所組成的。如圖 4 所示，位於 CID1 下方的串流記錄<S1,S3,S2>為 W1 的交易資料中，第一個時間點出現在 S1、

第二個時間點出現在 S3、第三個時間點出現在 S2，即表示 a 項目的第一個位元是在 S1 出現；第二個位元則是在 S3 中出現；第三個位元則是在 S2 中出現。

將顧客序列表轉換為位元表示之後，必須計算其支持度來辨別序列樣式是否為頻繁。支持度為某一個項目在顧客序列表中所佔的比例，以 sup 表示之。由於本研究為探勘跨串流序列樣式，因此計算支持度的方法和傳統方式不同。在計算支持度時必須考量項目是否存在於同一個串流中。例如：在 min_sup=2 的情況下，圖 4 中 c 項目的支持度在 S1 中為 3、在 S2 中為 3、在 S3 中為 2，皆大於等於 min_sup，因此 c 項目在三個串流中皆為頻繁項目。圖 5 為 1-目的索引集合，其分別在各個串流中記錄了 1-項目(1-item) 的支持度。

<a>	<b>	<c>	<d>
[S1:3,S2:0,S3:0]	[S1:1,S2:1,S3:3]	[S1:3,S2:3,S3:2]	[S1:1,S2:3,S3:2]

圖 5　1-項目支持度範例

3.3.2 探勘跨串流序列樣式

此章節可分為兩個步驟，第一步驟為利用序列延伸以產生候選序列集，第二步驟為將候選序列對映至位元映射矩陣以尋找頻繁跨串流序列樣式和串流序列。

序列延伸會根據圖 5 之 1-項目來產生候選序列，可分為兩個步驟：(1)項目集延伸步驟(I-step)；(2)序列延伸步驟(S-step)。這兩個步驟的延伸順序為先執行 I-step 後再執行 S-step。I-step 為新增一個項目在同一個項目集裡面，例如：<(a,b)>與<(c)>進行 I-step 之後會變成<(a,b,c)>，其可表示為<(a,b)> $\diamondsuit_I$ <(c)>=<(a,b,c)>；S-step 則是在一個項目集後新增一個項目，例如：<(a,b)>與<(c)>進行 S-step 之後會變成<(a,b)(c)>，其可表示為<(a,b)> $\diamondsuit_S$ <(c)>=<(a,b)(c)>。根據 Apriori 定理，非頻繁的子項目集(Sub Itemsets)其超項目集(Super Itemsets)必定為非頻繁，因此非頻繁項目集和非頻繁序列樣式則無需進行 I-step 和 S-step，也就不會產生多餘的候選序列。

接下來便會將 I-step 與 S-step 所產生的候選序列對映至位元映射矩陣。位元映射方法依不同的延伸方法有不同的映射方法， I-step 所產生的候選項目集為對映至位元映射矩陣中同一顧客同一時間點的位元 1，並且該位元 1 必須出現在同一串流中；S-step 則是對映至位元映射矩陣中同一顧客不同時間點的位元 1，而該候選序列之同一時間點的子序列必須出現在同一個串流中，其後再計算候選序列的支持度。如圖 6 所示，將 I-step 與 S-step 所產生的候選序列直接映射至位元映射矩陣並計算支持度，映射方法如下所述：

- I-step：在此步驟中僅需針對在同一串流中的頻繁項目做延伸，如圖 5 所示，項目 a 在 S1 中的支持度為頻繁，因次便針對在 S1 中的項目 a 做項目延伸，以同位於 S1 串流中的<(a,c)>為例。對映至位元映射矩陣，如圖 6 所示，欲計算<(a,c)>支持度可從 a 與 c 的欄位中看到，CID 1、CID 3 和 CID 4（圖 6 的粗體 1)都各在同一個時間點出現位元 1，並且皆是出現在 S1 中，因此可計算其支持度為 3;而在 S2 和 S3 中的項目 a，因其為非頻繁，執行 I-step 後必定不為頻繁項目集，因此在該移動框架中並不需要去處理，以減少多餘候選序列的產生。
- S-step：此步驟可根據 I-step 產生的頻繁項目集做序列延伸，亦可針對 1-項目做序列延伸，例如：從上述 I-step 中可得到<(a,c)>，而此例將針對<(a,c)>做 S-step。<(a,c)(d)>為<(a,c)>所延伸之後選序列，支持度可從圖 6 之 a、c、d 的欄位中看到，因為序列樣式是有時間順序關係，所以<(d)>必須在<(a,c)>之後的時間點出現。在

CID 1 中 S1 之<(a,c)>的第一個時間點和 S3 之<(d)>的第二個時間點以及在 CID 3 中 S1 之<(a,c)>的第二個時間點和 S3 之<(d)>的第三個時間點(圖 6 底線 1)都同時出現位元 1，因此便可計算<(a,c)(d)>的支持度為 2，串流序列為<(S1,S3)>。

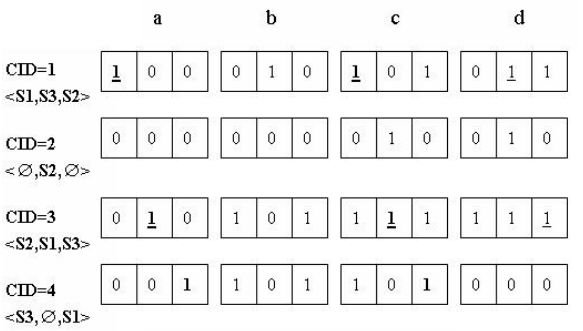

圖 6　W1 顧客位元映射矩陣範例

執行完 I-step 與 S-step 之位元映射後，會將找到的頻繁項目集和頻繁跨串流序列樣式儲存至序列辭彙樹中，接著移動框架便會往後移一筆交易。當 W2 擷取到串流資料後，圖 4 中的每個位元會往左移一個位元，如圖 7 所示。W2 的新進資料為<S2,(a,c,d)>、<S1,(c,d)>、<S3,(a,d)>，即為表 2 的第三個時間點項目集，將所有位元左移一個時間點後，便會將新進項目集轉換成位元並且新增至位元映射矩陣的第三個時間點。

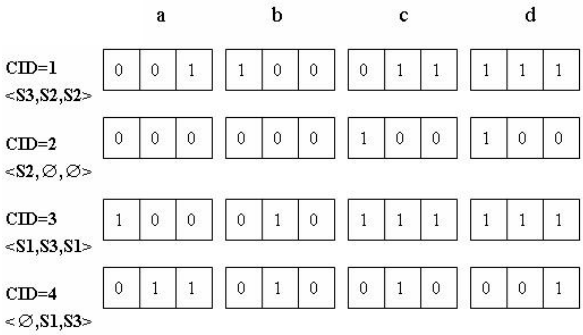

圖 7　W2 顧客位元映射矩陣範例

3.3.3 建置序列辭彙樹與串流雜湊表

建置序列辭彙樹的目的主要是為了保留頻繁序列樣式，根節點為一個空集合，樹的第一層為 1-項目集合，依序往下建立，在建置完辭彙樹之後會針對移動框架的移動而動態的新增、修改或刪除樹節點和更新支持度，讓樹節點僅保留頻繁序列樣式。

在這些葉節點中的項目集中存在著辭彙順序(Lexicographical Order)，辭彙樹的建置是以由上而下、由左而右的順序建置，而辭彙的順序大小則是階層越大的節點其項目集越大，越左邊的節點其項目集越大，根據辭彙順序的特性，其序列辭彙順序會有以下情況[36]：

- 如果 $s' = s \diamond p$ 則 $s < s'$，例如:<(ab)><<(ab)(b)>。
- 如果 $s = \alpha \diamond_I p$，$s' = \alpha \diamond_I p'$並且 $p < p'$ 則 $s < s'$，例如:<(abc)><<(abd)>。
- 如果 $s = \alpha \diamond_S p$，$s' = \alpha \diamond_S p'$並且 $p < p'$ 則 $s < s'$例如:<(ab)(c)><<(ab)(d)>。
- 如果 $s = \alpha \diamond_I p$ and $s' = \alpha \diamond_S p'$, 無論 p 與 p’ 之間的順序如何，只要 $s < s'$，例如:<(abc)><<(ab)(a)>。

依前述的辭彙樹架構規則，將建置一個辭彙樹實例，如圖 8 所示，其為第一個移動框架 W1 所建置出的序列辭彙樹，每個節點都會紀錄其串流序列和串流支持度，以辨識該序列樣式是來自哪個序列，並且以支持度來更新或刪除節點。隨著框架的移動，序列辭彙樹會不斷的做變動，以保留住頻繁序列樣式，新的資料進來之後，辭彙樹的節點會依據下一個框架的探勘結果，新增新的節點或是更新現有節點的支持度。在這個步驟中，有可能會發生一個頻繁樣式存在於辭彙樹節點中，卻因為新資料的進入不包含該序列樣式，因此導致長時間未被更新，而該頻繁序列樣式便會有過期的問題存在，因此必須加入權重(Weight)，以遞減其支持度，將長久未被更新的頻繁序列樣式刪除，

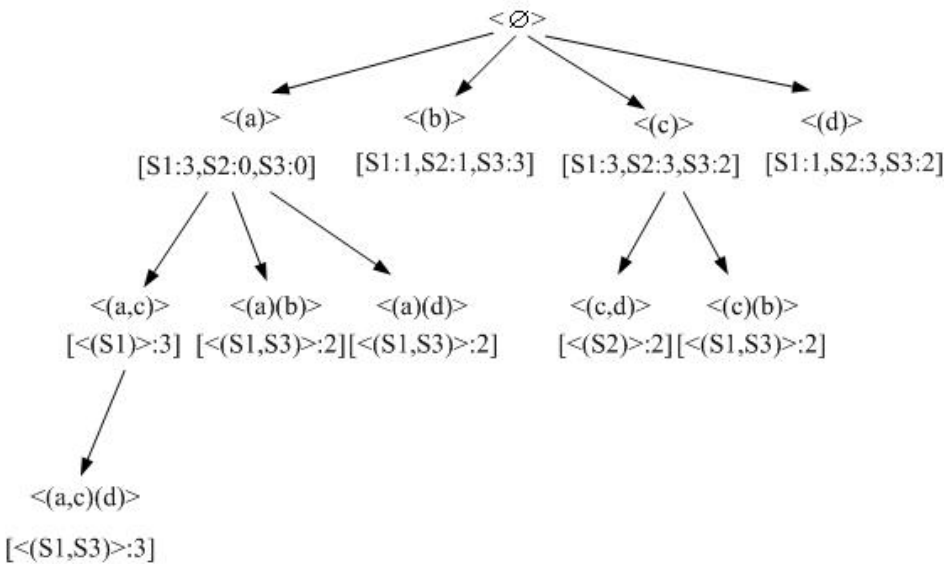

圖 8　W1 之頻繁序列辭彙樹範例

當支持度小於最低加權支持度門檻值(Minimum Weighted Support Threshold)便刪除節點，以此方法來解決這個問題。權重方程式如下：

$$\text{Weight}=d^{p}\ ,\ 0\leqq\text{weight}\leqq 1 \qquad (2)$$

$$p=w-u \qquad (3)$$

在權重方程式(2)中，遞減率 d (Decay-rate)為使用者所定義，依 d 的大小來控制序列樣式遞減的速度。如方程式(3)所示，p 是序列樣式的遞減週期(Decay-period)，w 為框架移動的次數，u 為辭彙樹中樹節點被更新的次數，初始值為 1，遞減週期的計算方式則是利用框架移動次數減去樹節點被更新的次數，由此方程式可看出越久未被更新的樹節點其 p 值會越大，也就表示權重會越低。如(4)之加權支持度(Weighted Support, w_sup)方程式所示，當權重值乘上支持度小於使用所定義之加權支持度的最低門檻，該節點便會被刪除。

$$w\_sup=weight*sup \qquad (4)$$

依漸進式概念，框架位移後的第二個框架 W2 如圖 9 所示。圓圈標示部分為經過權重遞減的支持度，遞減率設定為 0.9 的情況下，其由支持度由 2 遞減為 1.8。由於遞減後小於最小支持度，因此該節點將會被刪除，樹枝以虛線表示。方框標示部分為更新後之支持度，表示該節點在 W2 亦為頻繁，因此會更新其支持度。<(b,d)>節點為 W2 中的新頻繁項目集，因此會將其新增至序列辭彙樹中。依此方式維護序列辭彙樹，將可保留頻繁序列樣式。

串流雜湊表(Stream Hash Table)主要是以串流為主鍵的雜湊表，每個主鍵會有各自的鏈結串列(Linked List)，其各個主鍵會有數個節點(Nodes)被鏈結串列所串連，而每個雜湊表的節點都會與辭彙樹的樹節點關連，如圖 10 所示，雜湊表儲存串流序列，每個欄位皆會有一個鏈結串列，鏈結串列中的每個鏈結會指向辭彙樹中的節點，利用序列雜湊表的特性，在搜尋頻繁序列樣式上可以直接透過序列雜湊表搜尋到串流序列後，再依其鏈結串列直接找到頻繁序列樣式，其與直接從辭彙樹根往下搜尋相較之下，使用雜湊表的搜尋時間會更加快速。

最大序列樣式(Maximal Sequential Patterns)即是在不論支持度大小的情況下，不存在比該序列更大的超序列，即可謂該序列樣式為最大序列樣式。在串流雜湊表的鏈結串列中，為了

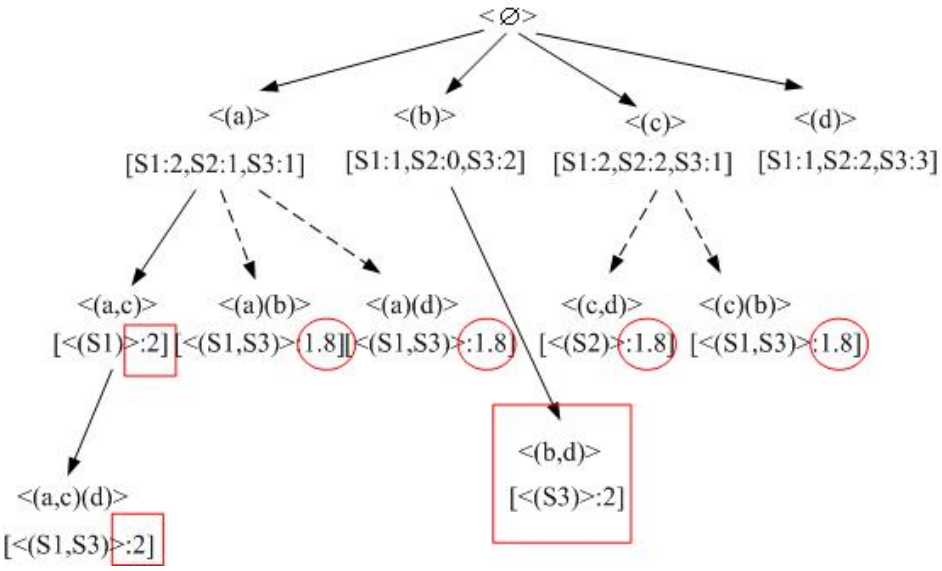

圖 9　W2 之頻繁序列辭彙樹範例

節省空間的使用，所以僅保留住最大頻繁序列樣式，例如：<(a)(c)>與<(ab)(c)>存在時，其會刪除<(a)(c)>的連結，並保留著<(ab)(c)>。因此，串流雜湊表的鏈結串列亦會隨著框架的移動和序列辭彙樹的更新而更新雜湊表和鏈結。

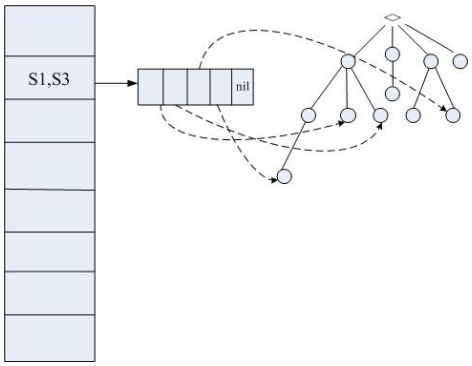

圖 10　串流雜湊表範例

3.3.4 漸進式跨串流序列樣式探勘演算法

圖 11 為本研究之漸進式跨串流序列樣式探勘演算法(稱之為 IAspam 演算法)，演算法的參數定義如下：

- SW_i 為第 i 個移動框架。
- min_sup 為最小支持度。
- CID 為顧客編號。
- S 為串流編號。
- CI 為候選項目集、CP 為候選序列。
- k-item 為項目長度 k 之項目。
- CS 為顧客序列表，以暫時儲存顧客資料。
- L-tree 為辭彙樹。
- FASP 為頻繁序列樣式。
- Stp 為串流序列樣式。

IAspam 演算法的步驟分為三個部份。第一部份是轉換成位元映射表示法(圖 11 的 Step 2)。這個部份是將顧客交易資料轉換成位元映射矩陣。第二部份是漸進式探勘跨串流序列樣式(圖 11 的 Step 3-5)，將候選序列對映至位元矩陣，找出頻繁跨串流序列樣式。第三部份則是建置序列辭彙樹與串流雜湊表並且遞減支持度(圖 11 的 Step 6-7)，將所找到的頻繁序列樣式儲存至序列辭彙樹中，並且使用串流雜湊表鏈結至辭彙樹節點，以快速搜尋跨串流序列樣式。以下是 IAspam 演算法詳細步驟說明：

Step 1：將框架內的串流資料以轉換成顧客序列表，資料格式為(Si,I_{si})，Si 代表串流，I_{si} 代表 Si 串流所屬之項目集。

Algorithm: IAspam

Input：SW_i , min_sup, CID, S, I, k-item, CS, L-tree

Output：FASP, StP

1. transform the transation data of SW_i into CS **;**
   For each I **do** store it into CS order by CID;
   For each CID **do** store customer information as (Si, I_{si});
2. **For** CS **do** turn it into bit-vector with three column for each k-item;
   If the I_{si} exist in k-item **then** set bit 1;
   else set bit 0;
   record (S_{C1} ,S_{C2}, S_{C3}) and store into bit-vector matrix;
3. **For** each 1-item **do** count support with the same Si and store into L-tree with [Si:sup(I_{si})];
4. **For each** node of L-tree
   If the stream of 1-item is frequent **then do** I-step then S-step and count support;
5. **If** sup(CI) and sup(CP) ≧ min_sup **then** add the FASP and StP into the L-tree;
   else eliminate it;
6. **For each** StP of the L-tree **do**
   insert it into stream hash table;
   construct the linked list to link to node of L-tree;
   maintain the maximal frequent patterns;
7. **For each** FASP **do** insert new FASP into L-tree or update support of node or decay the weight of FASP in L-tree;
   return;

圖 11 漸進式跨串流序列樣式探勘演算法

Step 2：將顧客序列表轉換成位元映射表示法，有存在的項目設為 1，否則設為 0，並將所轉換的位元資料儲存至顧客位元映射矩陣，並且記錄(S_{C1},S_{C2},S_{C3})，各三個時間欄位之 SID。

Step 3：計算同一個串流中的 1-項目集支持度，以[Si:sup(I_{si})]的資料格式儲存至序列辭彙樹 L-tree 中之第一層樹節點。

Step 4：針對各串流頻繁的 1-項目進行 I-step 與 S-step 來產生候選鍵。

Step 5： 將每一個延伸項目和延伸序列對映至顧客位元映射矩陣，檢查其是否存在，如果該項目存在，則對延伸項目尋找在不同個項目而在同一個時間點並且是同一串流的位元 1 計算其支持度；對序列項目尋找在不同項目而在不同個時間點並且前序(prefix)是同一串流的位元 1，後序(postfix)是同一串流的位元，並計算其支持度，支持度大於 min_sup 則新增頻繁跨串流序列樣式(FASP)和串流序列(StP)至序列辭彙樹中。

Step 6：產生串流雜湊表，將 StP 儲存至串流雜湊表，並且產生鏈結串列鏈結至序列辭彙樹中的節點，僅保留最大序列樣式之鏈結。

Step 7：進行序列辭彙樹更新，辨識後續框架的探勘結果，如果為新序列則新增節點，如為已存在序列則更新節點，如果節點未被更新則以權重值遞減支持度。

4. 結論

本研究提出的 IAspam 演算法，與一般的多重資料串流序列樣式探勘演算法的差異在於其是探勘介於串流與串流之間的跨串流序列樣式。此研究著重於探勘新類型的序列樣式，不僅找出交易資料的序列樣式也找出串流

之間的序列樣式。此外，IAspam 演算法使用位元映射方式使得在效能方面能降低其時間與空間的耗費。

參考文獻

[1] Agrawal, R. and Srikant, R., "Mining Sequential Pattems," ***the 11th International Conference on Data Engineering***, pp. 3-14, Taipei, Taiwan, 1995.

[2] Aseervatham, S., Osmani, A. and Viennet, E., "bitSPADE: A Lattice-based Sequential Pattern Mining Algorithm Using Bitmap Representation," ***the 6th International Conference on Data Mining***, pp. 792-797, 2006.

[3] Ayres, J., Gehrke, J., Yiu T. and Flannick, J., "Sequential Pattern Mining using A Bitmap Representation," ***the 8th ACM International Conference on Knowledge Discovery and Data Mining (KDD)***, pp. 429-435, 2002.

[4] Burdick, D. , Calimlim, M., Flannick, J., Gehrke, J. and Yiu, T., "MAFIA：A Maximal Frequent Itemset Algorithm," ***IEEE Transactions on Knowledge and Data Engineering*** , pp. 1490-1504, 2005.

[5] Burdick, D. , Calimlim, M., Flannick, J., Gehrke, J. and Yiu, T., "MAFIA：A Maximal Frequent Itemset Algorithm," ***IEEE Transactions on Knowledge and Data Engineering*** , pp. 1490-1504, 2005.

[6] Chang, L., Yang, D., Wang, T., and Tang, S., "IMCS: Incremental Mining of Closed Sequential Patterns," ***G. Dong et al.: APWeb/WAIM 2007, LNCS 4505***, pp. 50–61, 2007.

[7] Chang, J. and Lee, W., "Decaying Obsolete Information in Finding Recent Frequent Itemsets over Data Stream," ***IEICE Transaction on Information and Systems***, Vol. E87-D, No. 6, June, 2004.

[8] Chang, J. and Lee, W., "A Sliding Window Method for Finding Recently Frequent Itemsets over Online Data Streams," ***J. Inf. Sci. Eng. 20(4)***, pp. 753-762 , 2004.

[9] Chang, J. H. and Lee, W.S., "Finding recent frequent itemsets adaptively over online data streams," ***the 9th ACM SIGKDD international conference on Knowledge discovery and data mining***, pp. 487-492, 2003.

[10] Charikar, M., Chen, K. and Farach-Colton, M., "Finding Frequent Items in Data Streams," ***the 29th International Colloquium on Automata, Language and Programming***, pp. 251-258, 2002.

[11] Chen, G., Wu, X. and Zhu, X., "Sequential Pattern Mining in Multiple Streams," ***the 5th IEEE International Conference on Data Mining***, pp. 27-30, 2005.

[12] Cheng, H., Yan, X. and Han, J., "IncSpan: Incremental Mining of Sequential Patterns in Large Da tabase," ***the 10th ACM International Conference on Knowledge Discovery and Data Mining***, pp. 97-121, 2004.

[13] Chi, Y., Wang, H., Yu, P. and Muntz, R., "MOMENT: Maintaining Closed Frequent Itemsets over a Stream Sliding Window," ***the 4th IEEE International Conference on Data Mining***, pp. 59-66, 2004.

[14] Datar, M., Gionis, A., Indyk, P. and Motwani, R., "Maintaining stream statistics over sliding windows," ***the 13th annual ACM-SIAM symposium on Discrete algorithms***, pp. 635-664, 2002.

[15] Domingos, P. and Hulten, G., "Mining

high-speed data streams," ***the 6th ACM SIGKDD international conference on Knowledge discovery and data mining***, pp. 71-80, 2000.

[16] Golab, L. and Ozsu, M.T., "Issues in Data Stream Management," ***ACM SIGMOD Record Volume 32 , Issue 2***, pp. 5-14, 2003.

[17] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. and Hsu, M.-C., "Freespan: Frequent pattern-projected sequential pattern mining," ***the 6th ACM SIGKDD international conference on Knowledge discovery and data mining,*** pp. 355–359, 2000.

[18] Ho, C.C., Li H.F., Kuo, F.F. and Lee, S.Y., "Incremental Mining of Sequential Patterns over a Stream Sliding Window," ***the 6th IEEE International Conference on Data Mining - Workshops***, pp. 677-681, 2006.

[19] Ho, C.C., Li, H.F., Kuo, F.F. and Lee, S.Y., "A New Algorithm for Maintaining Closed Frequent Itemsets in Data Streams by Incremental Updates," ***the 6th IEEE International Conference on Data Mining***, pp. 672-676, 2006.

[20] Ezeife, C.I. and Monwar, M., "SSM：A frequent Sequential Data Stream Patterns Miner," ***IEEE Symposium on Computational Intelligence and Data Mining***, pp. 120-126, 2007.

[21] Hulten, G., Spencer, L. and Domingos, P., "Mining time-changing data streams," ***the 7th ACM SIGKDD international conference on Knowledge discovery and data mining***, pp. 97-106, 2001.

[22] Ju, S. and Chen, C., "MMFI: An Effective Algorithm for Mining Maximal Frequent Itemsets," ***International Symposiums on Information Processing***, pp. 144-148, 2008.

[23] Lambert, D. and Pinheiro, J.C., "Mining a stream of transactions for customer patterns," ***the 7th ACM SIGKDD international conference on Knowledge discovery and data mining***, pp. 305-310, 2001.

[24] Li, H. and Chen, H., "*GraSeq* : A Novel Approximate Mining Approach of Sequential Patterns over Data Stream,"***ADMA 2007, LNAI 4632***, pp. 401-411, 2007.

[25] Li, H. and Chen, H., "DSOSW: A Deleting Strategy in Mining Frequent Itemsets over Sliding Window of Stream," ***2008 International Symposiums on Information Processing ,*** pp. 135-138, 2008.

[26] Li, H.F., Lee, S.Y. and Shan, M.K., "An Efficient Algorithm for Mining Frequent Itemsets over the Entire History of Data Streams," ***the 5th International Workshop on Knowledge Discovery in Data Streams***, 2004.

[27] Lin, M.Y. and Lee, S.Y., "Incremental update on sequential patterns in large databases," ***the 10th IEEE International Conference***, pp. 24-31, 1998.

[28] Manku, G. S. and Motwani, R., "Approximate frequency counts over data streams," ***the 28th international conference on Very Large Data Bases***, pp. 346-357, 2002.

[29] Oates, T. and Cohen. P.R., "Searching for structure in multiple streams of data," ***the 13th International Conference on Machine Learning***, pp. 346–354, 1996.

[30] Pei J., Han, J., Mortazavi-Asl, B. and Pinto, H., "PrefixSpan: Mining Sequential Patterns

Efficiently by Prefix-Projected Pattern Growth," ***the 17th Internatioanl Conference on Data Engineering***, pp. 215-224, 2001.

[31] Raissi, C., Poncelet, P. and Teisseire, M.," SPEED : Mining Maxirnal Sequential Patterns over Data Strearns," ***the 3rd International IEEE Conference Intelligent Systems***, pp. 546-552, 2006.

[32] Ren, J. and Li, K., "Find Recent Frequent Items with Sliding Windows in Data Streams," ***the 3rd International Conference on International Information Hiding and Multimedia Signal Processing - Volume 02*** , pp. 625-628, 2007.

[33] Srikant, R. and Agrawal, R., "Mining Sequential Patterns: Generalizations and Performance Improvements," ***the 5th International Conference on Extending Database Technology: Advances in Database Technology***, pp. 3-17, 1996.

[34] Wang, J. and Han, J., "BIDE: Efficient mining of frequent closed sequences," ***the 20th International Conference on Data Engineering*** , pp. 79-90, 2004.

[35] Yan, X.,Han, J., and Afshar, R., "CloSpan: Mining Closed Sequential Patterns in Large Datasets," ***2003 SIAM Int'l Conf. on Data Mining***, pp. 438-457, 2003.

[36] Yan, X.,Han, J., and Afshar, R., "CloSpan: Mining Closed Sequential Patterns in Large Datasets," ***2003 SIAM Int'l Conf. on Data Mining***, pp. 438-457, 2003.

[37] Zaki, M.J., "Efficient enumeration of frequent sequences," ***the 7th international conference on Information and knowledge management***, pp. 68-75, 1998.

[38] Zhenglu, Y. and Masaru, K., "LAPIN-SPAM: An Improved Algorithm for Mining Sequential Pattern," ***the 21st International Conference on Data Engineering***, pp. 1222, 2005.