# 以可重複序列挖掘網路瀏覽規則之研究
# Mining Web Traversal Rules with Sequences

陳仕昇* 許秉瑜** 陳彥良*
**Shih-Sheng Chen** **Ping-Yu Hsu** **Yen-Liang Chen**

(Received Apr. 20, 1999; First Revised Nov. 9, 1999; Second Revised Mar. 29, 2000;
Accepted Apr. 11, 2000)

## 摘要

網頁瀏覽過程中的許多資料，如能適當的整理，應可幫忙了解使用者的瀏覽行為。而了解瀏覽者的行為不僅可幫系統預取所需網頁，並可幫系統業者決定在那些網頁上刊登廣告，也可替 proxy server 制定較佳的資料更新策略，減少使用者在瀏覽網頁的等待時間。在本篇論文中我們利用資料挖礦技術來建立使用者瀏覽關聯規則，以了解使用者的瀏覽行徑，與先前研究不同的是為能了解使用者瀏覽順序，此關聯規則是建立在序列的資料結構上並且將發現的關聯規則分成前溯型及後推型兩種；為了解決序列獨特的重複問題，此論文設計了特別的門檻值計算方法；為了解決門檻值與序列長度成反向成長的問題，此論文設計了 Next Pass Large Threshold 及 Next Pass Large Sequence。

關鍵詞：資料挖掘、樣式、關聯規則、序列式資料

## Abstract

Web traversal patterns and rules are valuable to both Electronic Commerce and System Designers. If business owners know users' traversal behaviors, they can put advertisement banners in proper web pages with proper order. The same information can help systems to pre-fetch web pages and reduce response time. In this article, we propose a new data mining method to find the traversal patterns and associated rules. Traversal patterns are recorded in sequences, which have total orders among their elements. Sequences may have duplicated elements, and hence requires a new threshold computing method. The new method results in thresholds decreasing when sequences expanding. To resolve the issue, we design Next Pass Large Threshold and Next Pass Large Sequences to forecast needed sequences and thresholds. To expand sequences properly, sequence join, instead of traditional set join is employed. Since sequences contain orders, the rules established include forward reasoning and backward reasoning. Forward reasoning asserts rules in the order of event happening. Backward reasoning, on the other hand, asserts the rules in the reversed order. Both rules are valuable to EC and system desig ners.

* 中央大學資訊管理研究所
Department of Information Management, National Central University
** 中央大學企業管理研究所
Department of Business Administration, National Central University

**Keywords**：data mining, pattern, association rule, sequence data

# 壹、簡介

## 一、說明環境及問題敘述

在網路盛行的今日，使用者在使用網路的環境下，在網站業者的資料庫中留下許多資料。以企業體而言，如果能對使用者瀏覽網頁先後順序的行爲能有所了解，就能作更好的目標行銷。例如企業體要在網路上刊登廣告，雖然目前可利用網頁參訪次數的資訊來刊登廣告，但這只是一個點的廣告效果。如果能了解使用者在巡覽某些網頁後接著會到那些網頁，那我們就可在這些網頁上刊登一系列的相關廣告，形成具有劇情的廣告，如同電視劇情式廣告一般，比電視廣告更好的是使用者經由不同的瀏覽路徑，有不同的廣告劇情，這種廣告形成線的宣傳效果，讓使用者對企業體的廣告有整體且連續的印象。另外在使用者巡覽目前這些網頁時，如果能事先得知使用者會從那些網頁巡覽而來，那我們可先在那些網頁上刊登廣告，再對目前這些網頁的廣告作加強、歸納的效果，讓使用者對企業體的廣告有更深的印象，使得廣告形成面的宣傳效果。我們認爲經由上述的兩種方式將可發揮最大的網路行銷效果。欲達到如此，則必須了解使用者最常瀏覽的網頁路徑，以便在適合的網頁刊登適合的廣告，藉以吸引使用者。

就學校而言，現在上網常常花許多時間在等待網頁下載上，如果 Proxy Server 可根據使用者瀏覽路徑得知使用者網頁瀏覽行爲模式，先預測使用者可能到達的網頁，那就可以在使用者還在看目前的網頁時，就將接下來要看的網頁預取到硬碟中，並提供硬碟更好的資料更新策略，這能使用者瀏覽網頁更方便並且省掉等待資料傳送的時間。

爲能記錄瀏覽順序，系統應記錄網頁瀏覽次序，而不只是瀏覽過的網址。而在資料挖礦 (Data Mining) 中利用關聯規則(Association Rule)在網路的環境下找出序列的論文並不多，我們在盡最大的努力後只發現兩篇(Mobasher et al., 1996; Chen et al., 1998)。(Mobasher et al., 1996)是從 Access Log File 找出關聯規則及 sequential patterns，在它計算 support 的方式是看看 sequential pattern 是否在一筆交易 (Transaction) 中，不管 sequential pattern 在一筆交易中出現的次數，這樣的作法，無法反應每個 sequential pattern 存在交易的實際次數，找出每個 sequential pattern 在 Access Log File 真正的比重，對 Access Log File 中的 sequential pattern 作更好的分析。在(Chen et al., 1998 )強調在 Web 上的應用，要找出使用者瀏覽網頁的模式，其所產生的路徑都是不可有迴圈的；也就是一旦使用者回到已瀏覽過的網頁即代表一個新瀏覽序列的開始。我們認爲這樣的假設並不完全符合現實瀏覽模式，因爲在現實狀況中，使用者可能在瀏覽查詢的過程中，因種種原因歷經好幾個相同網頁，但每次都著重不同位置或不同資訊。因此我們認爲以回到相同網頁就斷定一個瀏覽序列已結束似乎不完全合理。

此論文所發展的演算法直接根據使用者從上網到離線的登錄記錄，來分析並自動產生一般使用者最常用的瀏覽路徑模式。在此演算法中，使用者的一個瀏覽序列是啓始於使用者向 server 送出第一個 request，而終結於使用者中斷使用前的最後一個 request。而如何斷定這使用者已中斷使用，則以閒置時間來衡量。當使用者閒置時間超過某一系統設定值時，即認定已中斷使用。

此論文利用序列 join 逐漸加長序列的長度，找出使用者常瀏覽的網頁模式。此法可找出帶有重複網頁的路徑，供決策者作參考。在找出常用路徑後，我們並利用傳統信度定義找出關聯規則，這種利用使用者瀏覽的路徑所找出的關聯規則，我們稱爲瀏覽關聯規則。因爲序列本身帶有先後順序，因此找出的規則也有方向之分。如 AB→CD 即代表在參訪 AB 後，使用者參訪 CD 的機會相當大，而 AB←CD 則代表如果已知使用者參訪過 CD，則其參訪過 AB 的機會相當大。

我們在第貳章介紹在資料挖礦中運用關聯規則找尋序列樣式(Pattern)的相關研究，第參章說明如何定義所使用的序列及在實際應用時的情況，第肆章我們用兩個演算法詳細說明門檻值的設計及有兩個演算法的處理流程，並舉例說明處理程序，第伍章說明何謂前溯型及後推型的關聯規則及如何求出這些規則，第陸章則以實例說明兩種演算法所產生序列的差異，最後在第柒章談未來的發展。

## 貳、相關研究

在資料庫的研究中，資料挖礦目前是一個新的領域，它的目的是從資料庫的大量資料中，將潛在有用的資訊及知識挖取出來(Agrawal et al., 1993 ； Chen et al., 1996 )。而在資料挖礦中非常重要的也是經常看到的課題就是如何才能從企業中的銷售交易資料庫(Transaction Database)中，找出項目(Item)間的相互關性，並把這些關聯性用關聯規則的型式表示。

挖掘關聯規則的問題最早是在(Agrawal et al., 1993)中所提出，它的定義如下：交易資料庫中有許多的銷售交易記錄，在每一筆交易中都記錄顧客所購買的項目，而關聯規則的形式如下

X→Y

其中 *X* 和 *Y* 分別代表項目的集合，意思是若購買項目集合 *X* 時，可能會再購買項目集合 *Y*。要找出規則 *X*→*Y*，我們必須先計算項目集合 *X* 及項目集合 *X*∪*Y* 的支持度(Support)，即資料庫有幾筆交易包含它們。接著，我們必須確定 *X*∪*Y* 是否爲大項目集合（Large Item Set）即 *X*∪*Y* 的支持度達到最小支持度（*minimum support*），則爲大項目集合。若是，我們才把 *X*∪*Y* 的支持度除以 X 的支持度，所得的值代表 *X*→*Y* 的信度（*Confidence*）。若信度達到最小信度（Minimum Confidence），則關聯規則 *X*→*Y* 成立。

之前大部分對於交易資料庫的研究均假設每一筆交易只記載顧客所購買的項目，卻沒有記載項目之間的順序關係(Agrawal, 1993; Agrawal and Srikant, 1994; Han et al., 1995; Park et al., 1995; Srikant and Agrawal, 1996)。換句話說，這些的研究假設每一交易可以用一個項目集合（A Set of Items）來代表，因爲在集合中，我們只問元素是否出現，但對於元素間的順序是不在乎的，這樣的做法在一般情形下是合理的，但若所記錄的資料具有順序關係時就不適用了。

早先研究關聯規則中的序列問題的如(Agrawal and Srikant, 1995；Bayardo, 1998; Srikant and Agrawal, 1996; Zaki et al., 1999)假設在一個顧客交易資料庫，要找出一般顧客購買物品的順序型態。每位顧客都是一個序列，序列中項目集(Itemset)的順序則是顧客購買物品的時間先後順序，先設定一個門檻值，找出只有一個品項(item)時的支持度，如果支持度通過門檻值則是 large，再將通過門檻值的品項作 sequential pattern join 產生候選序列，求出候選序列的支持度，找出 large 的候選序列，直到無法產生候選序列爲止。

本篇論文主要是研究使用者在網路上的瀏覽關聯規則。因此，在(Agrawal and Srikant, 1995; Bayardo, 1998; Srikant and Agrawal, 1996; Zaki et al., 1999)中所定義的 sequential pattern，它們定義的序列並不適合分析在 Web 上使用者瀏覽網頁的路徑。因爲在 Web 上，我們要明確知道每一個瀏覽過網頁的先後網頁，以便 prefetch 網頁或在一系列連續的網頁上登一系列的廣告，而非把連續性的網頁瀏覽樣式及非連續性的網頁瀏覽樣式視爲相同，這會造成在分析使用者瀏覽路徑的偏差。例如 sequential pattern<3,4,8>在(Srikant and Agrawal, 1996)中是視爲在一筆序列資料<7,3,9,1,5,4,8>有出現，但當我們在分析網路瀏覽路徑時更需要知道 sequential pattern 中的元素是否和序列資料中的元素一一連續相對應，因此在本篇論文中序列<3,4,8>不算在序列<7,3,9,1,5,4,8>出現，而序列<3,9,1>或序列<1,5,4,8>這種 sequential patterns 和序列<7,3,9,1,5,4,8>某些元素一一連續相對應才算在序列資料中出現過。另外在(Agrawal and Srikant, 1995) 中的 sequential pattern 具有 maximal 的特性，有其不足處，因爲只考慮 sequential pattern 是否達到最小支持度，而在本文中的瀏覽關聯規則亦具有 maximal 的特性，且考慮是否達到最小信度，這可瞭解更多 sequential pattern 的意義。

在網路上使用者的瀏覽行爲隱藏許多有用的知識，在(Perkowitz and Etzioni, 1997)中計設一個 Adaptive Site 對使用者的存取路徑作各種分析，在(Cooley et al., 1997)更針對 Web mining 作定義，將 Web mining 分成 Web Content Mining 和 Web Usage Mining，Web Content Mining 指的是直接自動搜尋線上可獲得的資訊加以分析，Web Usage Mining 指的是從 Web Server 上發現使用者存取路徑。若依(Cooley et al., 1997)所設計的 Web Usage Mining 的架構，本篇論文屬於其架構下的路徑分析。

在網路上要找出一般使用者瀏覽網頁路徑模式的問題，先前的(Mannila et al., 1995)可找到最常出現的序列，但沒有找出關聯規則。在計算 support 方面，先前的作法(Agrawal and Srikant, 1995; Bayardo, 1998; Srikant and Agrawal, 1996; Zaki et al., 1999)都是只看 sequential pattern 有沒有在交易中出現，並沒有考慮 sequential pattern 在交易中出現幾次。如此一來，當我們在分析網路瀏覽路徑時就沒有對更頻繁瀏覽的路徑 (Sequential Pattern)作更精確的分析以便在作 prefetch 網頁時將磁碟中的資料作適當的更新或在網路登廣告時找出使用者更常瀏覽的網頁，因此我們改變 support 的計算方式，考慮 sequential pattern 在每一筆交易中出現的次數。

## 參、資料結構

我們研究的資料組成是以序列來表示，同一個序列中的元素有先後順序，這與 Apriori(Agrawal and Srikant, 1994)演算法是在集合(set)下的關聯規則不同。在集合下的關聯規則，一個項目集中的元素並沒有先後順序的分別，而在序列中的關聯規則，一個序列中的元素有先後順序的分別。

| |
|---|
| vanc01m03-115.bctel.ca-user1 [10/Apr/1997:19:06:44 –07] "GET /SFE/cig-in?VG/dspcnf.cgi?ci=40050&fp=MLIST&query= HTTP/1.0" 200 3927 |
| cnts4p14.uwaterloo.ca-user2 [10/Apr/1997:19:09:43 –0700] "GET ?Waterloo/cgi-bin/UI/UI_welcom.cig HTTP/1.0" 200 1288 |

來源：Han and Fu (1995)

圖 1 一個原始登錄檔 (Log File) 的片斷

| |
|---|
| <A HREF="http://www.ncu.edu.tw/~im/" FIRST_VISIT="924425784" LAST_VISIT="924427609" VISIT_COUNT="2">國立中央大學資管系所</A> |
| <A HREF="http://www.ncu.edu.tw/~im/index2.html" FIRST_VISIT="924425788" LAST_VISIT="924427610" VISIT_COUNT="2">國立中央大學資管系所</A> |

圖 2 在瀏覽器上的歷史清單的片斷

此演算法所需的資料來源爲伺服器中的登錄檔(log file)，此檔包含使用者代號、request 時間及所需的 URL，如圖 1。 資料也可從個人瀏覽器中的歷史清單取出如圖 2。以上的資料經過整理，可將使用者代號、URL 及到訪時間萃取出來，排列形成所需的序列。如取於伺服器資料，一個序列中的物件應只含一個使用者所到訪過的 URLs。一個序列起始於使用者第一次到訪本伺服器或中斷超過一定時間後再次到訪，此時間可爲三十分鐘，一小時，或任何系統認爲合理的時間。一個序列止於使用者超過前定時間未再造訪本站，此時間的長短與資料搜尋的演算法並無直接關係，它只影響序列中的資料長短及所產生的規則多寡。在起始與終止間所尋取的 URLs 即爲序列的資料，此資料應依時間先後順序排列於序列中。如取於瀏覽器中歷史資料則只有個人資料而不會與他人資料相混合，因此不須使用者代號。每一個序列的開始起於瀏覽器閒置超過一定時間後被啓動，而終於瀏覽器再次被閒置超過前定的時間。此時間的長短也一樣由系統定義。而中間所到訪過的 URLs 也是依先後順序排列成序列。

如將每筆登錄記錄當中的 URL 當作一個元素(Element)，每次使用者上網瀏覽所經過的 URLs 即成爲一個序列。序列的基本資料結構爲 List，假設 $E$ 爲所有 element 的集合， S 爲所有序列的集合則

1) $\forall e \in E \ <e> \in S$ ，

2)假如 $s1 \in S$ 且 $s2 \in S$ 則 $s1+s2 \in S$, 其中 + 爲 list concatenation 運算元，

3)$S$ 的所有資料只能由上述兩式產生。

例如 $s=<\ldots e_i \ldots e_j \ldots>$，則在 $s$ 中，$e_i$ 比 $e_j$ 早被瀏覽過。對任何一個 $s \in S$，其長度$|s|$ = 內含的 URL 的個數。例如假設 $s=<e_1,e_2,e_3,e_4>$，則$|s|=4$。第肆、伍章中的演算法將爲此資料結構找出所有的序列及其關聯規則。

## 肆、演算法

我們也像 Apriori(Agrawal and Srikant, 1994) 中求關聯規則一樣須求取每個 sequential pattern 在資料庫中的 *support*。*support* 達到門檻值 *(Large Threshold)* 的序列才列入 *Large Sequence(LS)*。求完所有 *large sequence* 才以 Conditional Probability 算出關聯規則。爲了要獲得更精確 candidate sequence 在資料庫中出現的實際次數，以便對使用者瀏覽網頁的樣式有更深入的分析，我們的演算法有兩個特別之處：一是候選序列的比對，二是其門檻值的計算。由於我們改變 candidate sequence 與資料庫中序列資料的比對方式及其門檻值的計算，使得產生的 candidate sequence 產生的數量上有很大的變化，因此在本篇論文中使用兩種演算法求 Large Sequence。

在本章第一節的 Like-GSP 演算法修改(Srikant and Agrawal, 1996)中算 support 的方式，原本在(Srikant and Agrawal, 1996)計算 candidate sequence support 方式是以 candidate sequence 中是否在一筆 transaction 出現爲依據，而我們的做法是將 candidate sequence 在一筆 transaction 中實際出現的次數作加總當作 support，這種作法的好處是對於一些在資料庫中出現較頻繁的 candidate sequence 的計算達到加權的效果但要找出更多的 large sequence 卻無法做到。

爲了改進第一種演算法，我們在本章第四節設計一個具有 look ahead 性質的 $LA^x$ 演算法，它屬於一種探索性的作法，此種作法的考量點在本章第二節有詳細的說明，它主要是考慮 candidate sequence 在作比對時的總次數計算。$LA^x$ 演算法是一種以更客觀、嚴格的方式來求 candidate sequence 及 large sequence，這個演算法的特性是會捨棄一些 large sequence，捨棄的 large sequence 的數量由 look ahead 要先保留幾個階段的 Candidate Sequence 的數目決定，當 look ahead 保留的階段愈多，捨棄的 large sequence 數目愈少，但花的時間成本愈多。

### 一、 **Like-GSP** 演算法

| Path-Id | Path | Path Length |
|---|---|---|
| 001 | ABABCA | 6 |

| | | |
|---|---|---|
| 002 | BABD | 4 |
| 003 | EFGE | 4 |
| 004 | HIJI | 4 |
| 005 | KHK | 3 |

圖 3 使用者的瀏覽路徑

在(Srikant and Agrawal, 1996)中利用 GSP 演算法來找出 Sequential pattern，我們也利用相同的作法來作 join 產生下一回的 Candidate Sequence，但與 GSP 不同的是在計算 support 的方式。我們在算 support 時是計算出 Candidate Sequence 中的序列在實際每一條路徑出現的次數，而非只看 Candidate Sequence 中的序列有無在路徑中出現。以 GSP 演算法計算 candidate sequence 的 support 來看圖 3。假設有 5 條不同的使用者的瀏覽路徑，Threshold 要達到 40%才算是 large sequence，序列<A>在 001、002 有出現，那序列<A>的 support 爲 2，2/5 是 40%，有達到門檻值，因此序列<A>是 large sequence，序列<E>只在 003 出現，那序列<E>的 support 爲 1，1/5 是 20%，未達門檻值。其它<B>、<C>、…、<K>的算法也相同。而我們的作法是在度量 Candidate Sequence 中的序列出現在資料庫中的比例時，以資料庫中的 transaction 數(在本論文中是 Path)爲分母，這與 GSP 演算法相同，但分子是以資料庫中每筆交易和 Candidate Sequence 中每個序列在每筆交易實際出現的次數作累加，而非只是看看每筆交易中是否含有 Candidate Sequence 而已。這個演算法由於會計算 Candidate Sequence 中序列出現的次數，因此最後產生 Large Sequence 的數目將會很大。相同的假設以圖 3 爲例，以 Like-GSP 演算法來說，<A>在 001 出現 3 次，在 002 出現 1 次，因此<A>的 support 是 4，4/5 是 80%，所以<A>是 large sequence。同理可算出<B>、<C>、…、<K>，特別的是<E>、<I>、<K>。雖然<E>只在 003 出現，但算出現的次數是 2，因此 2/5 是 40%，也是達到門檻值的要求，序列<I>、<K>也是分別在 004、005 有相同的情況。

## 二、門檻值及 **Large Threshold** 及 **Next Pass Large Threshold** 的計算

傳統上一個序列出現在資料庫中的次數須達到門檻值才算 Large。在(Agrawal and Srikant, 1994)中門檻值以資料庫中交易總數的某個比例訂定。因此門檻值爲固定數。但在此演算法中，門檻值雖也是一個固定百分比，但其基準卻非交易總數。而是一個會根據序列長度變化的數目。若計算長度爲 *k* 的序列門檻值時，則其基準值爲資料庫中所有長度爲 *k* 的子序列個數。亦即

*k*-門檻值 = $|\{ ss \mid |ss| = k \text{ and } ss \in s \text{ and } s \in D\}| * minsup\text{-}ratio$，其中 *D* 爲資料庫，*s* 爲其中的序列，*ss* 爲子序列，而 *minsup-ratio* 爲某一最低門檻比例。

做此變革是因一個序列可能含有多個相同子序列，如 $<e_1,e_2,e_3,e_1,e_2>$ 中含有兩個 $<e_1,e_2>$ 子序列。在計算此子序列出現於資料庫中個數時，我們認定此子序列在此序列中出現兩次，而非傳統集合式資料庫中的一

次。因此某子序列在資料庫中出現的個數是以此子序列在所有序列中出現個數的總合爲計算。由此推理，我們認爲門檻值的基準數應爲資料庫中所有相同長度子序列數目的總和。因此，當選候選序列的長度增長時，門檻值基準值會隨著變化。

公式一

當資料庫中有 *n* 筆序列時，長度爲 *k* 的子序列門檻基準值爲

$$total_k = \sum_{i=1}^{n} max(|s_i| - (k-1), 0)$$

證明

對任一個長度達到 *k* 的序列 *s* 而言，其所含有的長度爲 *k* 的子序列個數爲|*s*| -(*k*-1)。

此現象可以數學歸納法證明：

當 *k*=1 時：子序列個數爲|*s*|。

當 *k*=*m*-1 時：假設個數爲|*s*|-(*m*-2)。

當 *k*=*m* 時：此序列所能擁有的長度爲 m 的子序列個數較長度爲 *m*-1 的子序列個數少一個。因此，長度爲 m 的子序列個數有|*s*|-(*m*-2) – 1 = |*s*| -(*m*-1)。此值必須大於等於 0，故得證長度爲 k 的子序列門檻基準值爲

$$\sum_{i=1}^{n} max(|s_i| - (k-1), 0)。$$

因此在比例固定下，隨著子序列長度的增加，門檻基準值作會逐漸減少，而造成了門檻基準數隨著序列的增長而遞減。這與傳統的資料挖礦情形有很大的不同。在傳統的方法中，門檻值是不變的。因此若某一個項目的集合(Itemset)在資料庫中的 *support* 不到門檻值，則由此集合與其他任何項目集合編成的新項目集合亦不可能在資料庫中有達到門檻值的 *support*。但在此論文所發展的方法中，因門檻值可能降低，而使前述情形變可能。亦即一個子序列的 *support* 可能未達門檻值，但由其所組合成的序列卻可能有足夠的 *support*。此問題若無法解決，會造成演算過程中須考慮到相當多的序列。因爲所有序列都應予保留到產生新的序列後，無論其是否是 *large sequence*，因爲它們可能組合出有足夠 *support* 的新序列。

爲解決此一問題，我們設計一個特別的門檻值稱爲 *Next Pass Large Threshold*，以便此演算法會向前多看一步或多步，視使用者的需求而定，由於此演算法有向前看(look ahead)的性質，因此稱爲 $LA^x$ 演算法，上標 x 指的是向前看 x 個 Pass 即向前看 x 步。以向前多看一步而言即 $LA^1$ 演算法，也就是說當判斷某一個序列爲 *large sequence* 的同時，此演算法亦同時預估其是否在長度增加 1 的新序列中仍有可能爲新 *large sequence* 的一部份。如果是則保留此序列，反之則放棄此序列。進行預估實際上相當容易，因爲我們已知如何估算每個長度的序列所需的門檻值，而且門檻值是成遞減狀態。也就是若此序列爲 *large sequence*，則應將其保留。若非 *large sequence*，則此演算法只需檢查此序列的 *support* 是否達到下一步驟的門檻值。如果是的話，亦將其保留。例如我們只向前多看一

步時，在圖 5 的 Pass 1 的 *Next Pass Large Threshold* 是 9，因此在圖 6 的 $NPLS^1_1$(在第肆章第三節有正式的定義)只有<A>、<B>、<C>、<D>、<F>。特別是如果向前多看二時，Pass 1 的 *Next Pass Large Threshold* 是 8，那在圖 6 的 $NPLS^1_1$ 還要加入<E>。下面的範例是以向前多看一步爲假設，計算出 *Next Pass Large Threshold* 及 *Large Threshold* 的序列個數。

| Path-Id | Path | Path Length |
|---|---|---|
| 001 | ABEABGABFDCG | 12 |
| 002 | BDFDEABFECABFDCDF | 17 |
| 003 | ABABEGCBABFDG | 13 |
| 004 | AEFEBABGABFD | 12 |
| 005 | DFABFD | 6 |
| 006 | ABFDCFDF | 8 |
| 007 | CFDABDFDGDF | 11 |
| 008 | BEABFDCGDFDC | 12 |
| 009 | ADFD | 4 |
| 010 | ACEFF | 5 |

圖 4 不同使用者的瀏覽路徑

| minsup-ratio 10% | | |
|---|---|---|
| Pass | Next Pass Large Threshold | Large Threshold |
| 1 | 9 | 10 |
| 2 | 8 | 9 |
| 3 | 7 | 8 |
| 4 | 6 | 7 |

圖 5 4 階的 Next Pass Large Threshold 及 Large Threshold

範例 ***1*** 在圖 *4* 中有十條路徑共有 *100* 個元素，我們假定 *minsup-ratio* 是 *10%*，現在要計算序列長度爲 *1* 的門檻值，因爲要比對的序列有 *100* 個，所以 *Large*

$CS^1_1$

| Candidate Sequence | Support |
|---|---|
| A | 17 |
| B | 18 |
| C | 9 |
| D | 20 |
| E | 8 |
| F | 21 |
| G | 7 |

$NPLS^1_1$

| Next Pass Large | Support |
|---|---|
| A | 17 |
| B | 18 |
| C | 9 |
| D | 20 |
| E | 8 |
| F | 21 |

$LS^1_1$

| Large Sequence | Support |
|---|---|
| A | 17 |
| B | 18 |
| D | 20 |
| F | 21 |

圖 6 $NPLS^1_1$ 及 $LS^1_1$ 的產生

*Threshold* 爲 *10* *( 100 * 10% )*，而 *Next Pass Large Threshold* 則根據序列長度爲 *2* 要比對檔案 *D* 的次數決定，例如候選序列長度爲 *2* 時，在 *Path-Id 001* 要比對 *11* 次，*Path-Id 002* 次要比對 *16*，…，*Path-Id 010* 要比對 *4* 次，全部共 *90* 次，所以 *Next ass Large Threshold* 爲 *9 ( 90 * 10%)*。我們看候選序列<*A*>的長度是 *1*，在 *Path-ID* 從 *001* 到 *010* 中，<*A*>出現 *17* 次，因此<*A*>的支持度是 *17*。因 *pass 1* 的 *Large Threshold* 是 *10*，可知<*A*>達到 *Large Threshold* 及 *Next Pass Large Threshold*。再看候選序列<*BA*>的長度是 *2*，只在 *003* 出現 *2* 次、在 *004* 出現 *1* 次，可得知<*BA*>的支持度是 *3*，而在 *pass 2* 中算出 *Next Pass Large Threshold* 爲 *8(80 * 10% )*，*Large Threshold* 爲 *9(90*10%)*，所以<*BA*>未達到長度爲 *2* 的 *Large Threshold* 和 *Next Pass Large Threshold*。值得一提的是在圖 *4* 中序列

<*BF*>的支持度爲 *8*，此數達到 *Next Pass Large Threshold* 但未達到 *Large Threshold*。依照傳統研究對 *Threshold* 的處理，其 *Threshold* 爲 *9* 在第 *2* 回就會把<*BF*>去掉，那在第 *3* 回中就找不到<*ABF*>。圖 *4* 範例中的前 *4* 個 *Large Threshold* 及 *Next Path Large Threshold* 的值列於圖 *5*。

## 三、候選序列的產生

有了 *next pass large sequence*，即可產生 *candidate sequence*。若 *s* 的支持度達到一個 *Large Threshold*，稱 *s* 爲一個大序列*(large sequence)*，而由 *s* 所組成的集合稱爲 *LS*。而 $LS^x_k$ 表示 $LA^x$ 演算法向前看 x 步時，每個 *s* 的長度爲 *k* 的序列集合，例如圖 6 中的 $LS^1_1$、$LS^1_2$ 等。*Next Pass Large Sequence (NPLS)* 爲要產生下一回 *Candidate Sequence (CS)* 作 join 的序列集合，類似 Apriori 的 next pass candidate itemset 是由 $L_{k-1}$ join $L_{k-1}$ 產生 $C_k$，現在是由 $NPLS^x_{k-1}$ join $NPLS^x_{k-1}$ 產生 $CS^x_k$。$NPLS^x_k$ 表示 $LA^x$ 演算法向前看 *x* 步時，此集合中序列的長度是 k，$CS^x_k$ 中的 *x* 及 *k* 的意義亦同。要產生下一回的候選序列 $CS^x_k$ 先由這回的 $NPLS^x_{k-1}$ join $NPLS^x_{k-1}$，而 join 的條件是這兩個序列須是可相銜接的序列。亦即，把兩序列中一個序列去掉第一個元素後的子序列等於另一個序列去掉最後一個元素後的子序列，這與(Chen et al. 1998 ; Srikant and Agrawal 1996)使用相似的 join 方法。

我們產生 $NPLS^x_{k-1}$ join $NPLS^x_{k-1}$ 候選序列的方式如下

insert into $CS^x_k$

select $p.e_1,\ldots,p.e_k, q.e_1,\ldots,q.e_k$

from $NPLS^x_{k-1}$ *p* , $NPLS^x_{k-1}$ *q*

where $p.e_2 = q.e_1, \ldots, p.e_k = q.e_{k-1}$

範例 ***2*** 圖 *6*、圖 *7*、圖 *8* 解釋 $LK^1$ 演算法由圖 *4* 中例子計算出 $NPLS^1_1$ 至 $NPLS^1_4$ 及 $LS^1_1$ 至 $LS^1_4$ 的過程。在 *2-Next-Pass-Large-Seqence* 中<*AB*>，<*BF*>，經由 *join* 會產生 *3-Candiate-Seqence* 的<*ABF*>，而在圖 *8* 中的 *3-Next-Pass-Large-Seqence* 中 <*ABF*>，<*BFD*>，經由 *join* 會產生 *4-Candiate-Seqence* 的<*ABFD*>。

## 四、$LA^x$ 演算法

$LA^x$ 演算法是在向前看 x 步的條件下求 candidate sequence，在計算 candidate sequenced support 的過程中我們先利用如 Boyer-Moore(Cormen et al. 1992)演算法計算每個序列在整個資料庫 D 出現的次數，如果其達到 *Large Threshold*，就加入 *LS* 中。如果達到 *Next Pass Large Threshold* 就成爲 *next pass large sequence*。

特別注意是當我們向前多看幾步時，*Next Pass Large Threshold* 會比較低，那會產生比較多的 *next pass large sequence*。接著將所有在 *NPLS* 中序列做 join，產生下一回的候選序列*(Candidate Sequence)*的集合，再求各候選序列的出現次數，找出下階段 *LS* 及 *NPLS* 不斷重複此步驟，直到無法再產生候選

序列爲止，整個演算法如圖 9。

$CS^1_2$

| Candidate Sequence | Support |
|---|---|
| AA | 0 |
| AB | 14 |
| AC | 1 |
| AD | 1 |
| AF | 0 |
| BA | 3 |
| BB | 0 |
| BC | 0 |
| BD | 2 |
| BF | 8 |
| CA | 1 |
| CB | 1 |
| CC | 0 |
| CD | 1 |
| CF | 2 |
| DA | 1 |
| DB | 0 |
| DC | 5 |
| DD | 0 |
| DF | 8 |
| FA | 1 |
| FB | 0 |
| FC | 0 |
| FD | 13 |
| FF | 1 |

$NPLS^1_2$

| Next Pass Large Sequence | Support |
|---|---|
| AB | 14 |
| BF | 8 |
| DF | 8 |
| FD | 13 |

$LS^1_2$

| Large Sequence | Support |
|---|---|
| AB | 14 |
| FD | 13 |

圖 7 $NPLS^1_2$ 及 $LS^1_2$ 的產生

$CS^1_3$

| Candidate Sequence | Support |
|---|---|
| ABF | 8 |
| BFD | 7 |
| DFD | 4 |
| FDF | 1 |

$NPLS^1_3$

| Next Pass Large Sequence | Support |
|---|---|
| ABF | 8 |
| BFD | 7 |

$LS^1_3$

| Large Sequence | Support |
|---|---|
| ABF | 8 |

$CS^1_4$

| Candidate Sequence | Support |
|---|---|
| ABFD | 7 |

$NPLS^1_4$

| Next Pass Large Sequence | Support |
|---|---|
| ABFD | 7 |

$LS^1_4$

| Large Sequence | Support |
|---|---|
| ABFD | 7 |

圖 8 $NPLS^1_3$，$LS^1_3$ ，$NPLS^1_4$ 及 $LS^1_4$ 的產生

1) $LS^x_1$=<large 1-sequence>;
2) $NPLS^x_1$=<符合條件的 1-sequences>
3) For ( k = 2 ; $NPLS^x_{k-1} \neq 0$ ; k++ ) do
4) Begin
5) $CS^x_k = NPLS^x_{k-1}$ join $NPLS^x_{k-1}$；
6) ForAll $cs_k \in CS^x_k$ Do
7) ForAll s∈ D Do
8) 利用利用如 Boyer-Moore(Cormen et al.1992)演 算法，
9) 累計 $cs_k$ 在所有 s 出現的次數
10) End
11) End
12) $NPLS^x_k = \{cs_k | cs_k \in CS^x_k$ 且 $cs_k$ 的 support 達到對應的 Next Path Large Threshold}
13) $LS^x_k = \{cs_k | cs_k \in CS^x_k$ 且 $cs_k$ 的 support 達到對應的 Large Threshold}
14) End

圖 9 整個處理的演算法

範例 ***3*** 藉由圖 *6-7* 解釋 $LK^1$ 演算法，一開始先算每一個元素出現的次數 ，即算出 $CS^1_1$ 的支持度，由圖 *4* 算出如 *A* 的支持度 *17*，*E* 的支持度 *8*，由要在 *D* 比對的總次數 *100*

乘以門檻值 *10%*得到 *Large Threshold* 爲 *10*，由第二回序列長度爲 *2* 算出在 *D* 要比對的總次數，求出 *Next Pass Large Threshold* 爲 *9*，所以在 $NPLS^1_1$ 有<*A*>、<*B*>、<*C*>、<*D*>、<*F*>，在 $LS^1_1$ 有<*A*>、<*B*>、<*D*>、<*F*>。將 $NPLS^1_1$ 中的序列作 *join*，可得 $CS^1_2$，$CS^1_2$ 中的序列再到 *D* 中作比對，求出 $CS^1_2$ 中每個序列的支持度，如<*AB*>支持度爲 *14*，<*BF*>支持度爲 *8*，由第三回序列長度爲 *3* 算出在 *D* 要比對的總次數 *80*，求出 *Next Pass Large Threshold* 爲 *8*。所以在 $NPLS^1_2$ 有<*AB*>、<*BF*>、<*DF*>、<*FD*>，在 $LS^1_2$ 有<*AB*>、<*FD*>，$LS^1_3$ 和 $LS^1_4$ 的產生也是一樣。最後，$NPLS^1_4$ 中的 *ABFD* 無法再產生候選序列，所以終止整個演算法的處理。在 $LS^1_4$ 中，我們找到的序列是<*ABFD*>，在 $LS^1_3$ 中找到的序列是<*ABF*>，在 $LS^1_2$ 中找到的是<*AB*>、<*FD*>。如果我們只要最長的序列，則只有<*ABFD*>，其它的<*AB*><*FD*><*ABF*>都是<*ABFD*>的子序列。

值得注意的是在範例 3，是以向前多看一步爲前提，以相對向前多看二步而言，在圖 6NPLS$^1_1$ 就少了<E>，因此在 CS$^1_2$ 就少與<E>作 join 的 candidate sequence，這最後可能導致捨棄一些 large sequence，這是 LA$^x$ 的本身的特質，因爲它不具有 Downward closure 的性質。如果要減少捨棄 large sequence 的數量，可向前多看幾步，但這會產生更多的 candidate sequence，但會得到更多的 large sequence。就整個演算法來說，如果全部使用者的瀏覽路徑最長爲 M 則可向前看 M-1 步。

公式二

如果全部使用者的瀏覽路徑最長爲 M 則可向前看 M-1 步。

證明：

當全部使用者的瀏覽路徑最長爲 1 時，顯然無法向前看。

當全部使用者的瀏覽路徑最長爲 2 時，只可向前多看 1 步。

當全部使用者的瀏覽路徑最長爲 n-1 時，可向前看 n-2 步，

則當全部使用者的瀏覽路徑最長爲 n 時，可向前看 n-1 步

因全部使用者的瀏覽路徑最長爲(n-1)+1，可向前看(n-2)+1 步。

## 五、LA$^x$ 演算法正確性評估

定理一可證明 LA$^x$ 演算法中 $LS^x_k \subseteq CS^x_k$ 的正確性。即在 look ahead x pass 的條件下，長度爲 *k* 的 *candidate sequence* 應包含所有長度爲 *k* 的 *large sequence*。

定理一

$\forall k$， $LS^x_k \subseteq CS^x_k$。

證明

當 $k=1$，$CS^x_1$ 爲所有出現於資料庫中元件的集合，因此假設顯然成立。

假設 當 $k=m-1$ 時，此式成立。

當 $k= m$ 時，

假設存在一個序列$<e_1,..,e_m> \in NPLS^x_m$ 且不在 $CS^x_m$。

因此序列的任一子序列的 *support* 應

達到相對應 look ahead *x* pass 的 *Next Pass Large Threshold*，所以 $<e_1,\ldots,e_{m-1}>$ 及 $<e_2,\ldots,e_m>$也是在 $NPLS^x_{m-1}$ 中，

又 $NPLS^x_{m-1}$ join $NPLS^x_{m-1} = CS^x_m$ ，因此$<e_1,\ldots,e_m> \in CS^x_m$。此與假設矛盾

∴ 不存一個序列 $s \in LS^x_k$ 且不在 $CS^x_k$ 中。

## 伍、關聯規則

從我們所找到的樣式，我們可以找出關聯規則，我們的規則有兩個特點：不但可以有傳統的後推規則如 $X \rightarrow Y$，我們也可以有前溯規則如 $X \leftarrow Y$。後推型的規則可以告訴我們當使用者巡訪某一網點後，接著會巡訪哪些網點，可用作 prefetch 網頁，而前溯型的規則卻可以告訴我們巡訪某一網點的顧客是從哪些網點過來的，可找適當的網頁刊登廣告。

(Ⅰ) 後推型關聯規則 $X \rightarrow Y$ 成立的條件有兩個：(一) 樣式 $Z=X+Y$ 的支持度必須達到 *Large Threshold*；(二) 規則的信度必須達到使用者所設定的最小信度 (*Minimum Confidence*)。在此信度的定義是

*confidence*$(X \rightarrow Y)$=*support*$(Z)$/*support*$(X)$

例如若規則是 AB→CD，則 $Z$=ABCD 且

*confidence*(X→Y)=*support*(ABCD)/*support*(AB)。

(Ⅱ) 同理，前溯型關聯規則 $X \leftarrow Y$ 成立的條件有兩個：(一) 樣式 $Z=X+Y$ 的支持度必須達到 *Large Threshold*；(二) 規則的信度必須達到使用者所設定的最小信度 (*minimum confidence*)。在此信度的定義是

*confidence*$(X \leftarrow Y)$=*support*$(Z)$/*support*$(Y)$

例如若規則是 AB←CD，則 Z=ABCD 且

*confidence*(X←Y)=*support*(ABCD)/*support*(CD)。

根據以上定義，我們可以從 large sequence 中產生這二類類型的規則，底下的範例解釋此一過程。

範例 ***4*** 我們假設 *minsup-ratio=10%*, *minconf=50%*，在圖 *7* 中 *AB* 的支持度是 *14*，圖 *8* 中 *ABFD* 的支持度是 *7*，則 *AB→FD* 的信度是 *50%*，此即(Ⅰ) 後推型關聯規則。在圖 *8* 中 *ABF* 的支持度是 *8*，在圖 *7* 中 *BF* 的支持度是 *8*，則 *A←BF* 的信度是 *100%*，此即*(Ⅱ)* 前溯型關聯規則。

## 陸、實際資料的實驗

我們利用中央大學圖書館的 Web server 上的 Access Log file 來作實驗的資料，在 Access Log file 的內容有使用者的 IP、瀏覽網頁的時間、瀏覽的網頁等。我們是以一個月的資料量爲分析對象，將原來 Access Log file 中每筆資料的形式是一個 IP 位址和一個讀取網頁動作的格式，轉成每一個 IP 在一段時間連續讀取的 HTML 檔當作一筆記錄加以編碼，變成像圖 3 所表示的資料結構。在檔案內總共有 22471 筆記錄。我們的實驗是在 Pentium-Ⅱ 266 的電腦及 288MB 的主記憶體

上執行程式，資料是放在 NTFS 的檔案系統上。

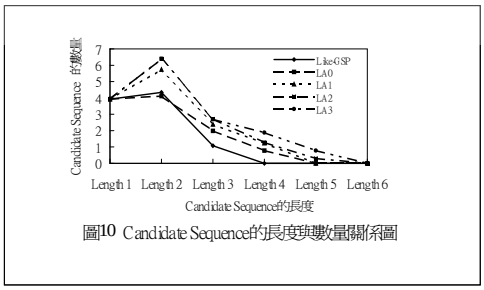


圖10 Candidate Sequence的長度與數量關係圖

圖 10 說明在 *minsup-ratio* 爲 0.00066 時，Like-GSP 演算法及 $LA^k$ 演算法於不同的 Candidate Sequence 長度下所產生的 Candidate Sequence 數量。此圖中之 Candidate Sequence 的數量是以對數表示。Candidate Sequence 長度爲 2 時，Look ahead 2 pass 和 Look ahead 3 pass 的 Candidate Sequence 數量相同，因此在圖 10 中 Length 2 看起來才只有 4 條線。由此圖可得知當 Look ahead pass 愈多時，產生的 Candidate sequence 的數量愈大，要得到 Large sequence 花的時間愈長。

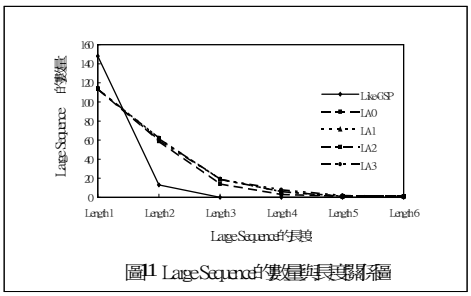


圖11 Large Sequence的數量與長度關係圖

圖 11 一說明在相同條件下，Like-GSP 演算法及 $LA^x$ 演算法在不同 Large Sequence 長度下所產生的 Large Sequence 數量，$LA^x$ 演算法在 Look ahead 0 pass、Look ahead 1 pass、Look ahead 2 pass 和 Look ahead 3 pass 產生 Large sequence 的數量差異很小，例如在序列長度爲 2 時，Look ahead 1 pass 和 Look ahead 2 pass 的 Large sequence 數量相同，在序列長度爲 3 時，Look ahead 1 pass、Look ahead 2 pass 和 Look ahead 3 pass 的 Large sequence 數量相同，因此畫出的曲線圖有許多重疊。在圖 10 中 Candidate sequence 長度爲 2 時，$LA^x$ 演算法向前看的回數愈多時，產生的 Candidate sequence 的數量可能會劇增，但圖 11 一可看出向前看的回數多時，它的 large sequence 的數量不會隨著序列的長度增長後，像 Candidate sequence 的數量一樣有劇增的情況。另外，圖 11 值得一提的是雖然 Like-GSP 演算法具有 downward closure 的性質，有較佳的執行效率，卻只能找出較少的結果。在執行時間上，Like-GSP 演算法花了 4 分鐘，$LA^0$ 演算法花了 8 分鐘，$LA^1$ 演算法花了 53 分鐘，$LA^2$ 演算法花了 3 小時 20 分鐘，$LA^3$ 演算法花了 3 小時 41 分鐘。由圖 10、圖 11 的結果，建議用 $LA^0$ 演算法，因爲相對 Like-GPG 來說，可以得到的 Large sequence 數量多很多，相對於 Look ahead 1 pass 到 Look ahead 3 pass 來說，產生的 Candidate sequence 數量少很多。在學校的實務應用上，我們可以根據得到 Large Sequence 依系統的需求，在 cache 資料作更新時作更好的管理，讓使用者在巡覽網頁時能快速。

## 柒、結論

我們這篇論文主要探討在網路環境下，如何利用伺服器端的登錄檔(log file)分析眾多使用者瀏覽網頁的行爲模式，以決定在作網路行銷時，廣告要放在那裡。或利用個別使用者瀏覽網頁的習慣，在學校 proxy server 的 cache 中的歷史資料中找出使用者瀏覽規則，cache 制定一個好的資料更新策略，幫使用者預取(pre-fetching)要繼續瀏覽的網頁，節省使用者要看網頁時，網頁下載的時間。

先前在類似課題上的研究都有所限制，譬如限制 sequence 內容不得有重複的項目，或研究成果只能用以預測在給定的一個 sequence 下，下一個可能的項目爲何。另外傳統在關連規則下的 Sequential Pattern 並未針對連續性的樣式及非連續性的樣式作區分，本論文則加以區分，以推算出連續性樣式的關聯規則爲主，將傳統的 Sequential Pattern 作更精確的分析，而且對樣式中的項目沒有任何限制，以提供決策者更正確的訊息。

在論文中演算法容許項目重複出現，因此對資料庫中所有 sequence 個數的算法與傳統的演算不同，產生兩個演算法。在第一個演算法，雖然產生的 Large sequence 具有 Downward closure 的性質，但產生的 Large sequence 較少，因此在第二個演算法我們改變門檻值的計算方式，當計算長度爲 *k* 的序列的門檻值時，此演算法以資料庫中所有相同長度的子序列數目爲基準值。因基準值隨著長度而變化，因此在公式一中我們提供了計算此值的公式。此基準值實際上因序列長度的增長而逐漸縮小，因此長度較長的序列只需較少的 support 即可達到門檻值。此特性造成 *candidate sequence* 可能由不在 *large sequence* 中的序列所組成。爲適當解決此問題，我們設計出 *Next Pass Large Sequence* 及 *Next Pass Large Threshold*，由使用者決定要往前看幾步，往前看的步數愈多，花的時間及空間成本愈多。序列 *support* 只要達到 *Next Pass Large Threshold* 即應被列入 *Next Pass Large Sequence*。由 *Next Pass Large Sequence* 中的序列互相 join 即可產生所有下一回的 *candidate sequence*。這個演算法由於序列本身的特性，因此找出的 large sequence 可能不具有 Downward closure 的性質，會捨掉一些 large sequence，但相對第一個演算法來說，卻找出更多的 large Sequence。

Join 運算元的含意也與傳統的集合運算中的 join 有所不同。在此論文中，兩個序列 join 的要素爲兩個序列在一個去頭而另一個去尾的情況下應該相等。此兩序列 join 後所產生的新序列則爲共同部份加上去兩序列獨特的部份。

最後將關聯規則分成前溯型及後推型兩種。並提供計算此兩類規則的方法。決策者可針對不同的用途及需要，設定最小的支持度及最小的信度並選擇所要的關聯規則類型。

我們的論文未來的發展方向主要包括三

方面，(1)產生一些具有指定的 meta-forms 的規則，這可以幫助去除掉很多與我們的應用不相關的規則，讓我們更看清楚規則間的關係。(2)將序列中的元素加上數量的考慮，這可以獲得更多精確的資訊。若以顧客購買物品爲例，當我們能預測顧客要購買物品的時間順序時，能進一步去推估那些物品的數量時，就能對物品的存貨作更好的控制。(3)這兩個演算法都需要對整個檔案作多次的掃瞄，需費較多的時間，因此未來要發展更節省時間的演算法，提高效率。

## 參考文獻


1. Agrawal, R., Imielinski, T., and Swami, A. "Mining association rules between sets of items in large database," *SIGMOD 93*, pp. 207-216.
2. Agrawal, R., and Srikant, R. "Fast Algorithms for Mining Association Rules," *Proc. Of the 20th Int'l Conference on Very Large Databases*, Santiago, Chile, Sept. 1994.
3. Agrawal, R., and Srikant, R. "Mining Sequential Patterns," *Proc. Of the Int'l Conference on Data Engineering (ICDE)*, Taipei, Taiwan, March 1995.
4. Bayardo, R. J. Jr. "Efficiently Mining Long Patterns from Databases," *Proc. Of the 1998. ACM-SIGMOD Int'l Conf. on Management of Data*, 1998, pp. 85-93.
5. Chen, M.-S., Han, J., and Yu, P.S. "Data Mining：An Overview from a Database Perspective," *IEEE Transactions on Knowledge and Data Engineering*, 8 , 1996, pp. 866-883.
6. Chen, M.-S., Park, J.S., and Yu, P.S. "Efficient Data Mining for Path Traversal Patterns," *IEEE Transactions on Knowledge and Data Engineering*, 1998 v.10 n.2 pp. 209-221.
7. Cooley, R., Mobasher,B., and Srivastava, J. "Web Mining: Information and Pattern Discovery on the World Wide Web," *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.
8. Han, J. and Fu, Y. "Discovery of Multiple-Level Association Rules from Large Databases," *VLDB* 1995, pp. 420-431.
9. Mannila, H., Toivonen, H., and Verkamo, A. I. "Discovering frequent episodes in sequences," *Proc. Of the First Int'l conference on Knowledge Discovery and Data Mining*, Montreal, Quebec, 1995, pp. 210-215.
10. Mobasher, B., Jain N., Han, E. and Srivastava, J. "Web Mining: Pattern

Discovery from World Wide Web Transactions," *Technical Report TR96-050, Department of Computer Science, University of Minnesota*, 1996.

11. Park, J.-S., Chen, M.-S., and Yu, P.S. "An Effective Hash-Based Algorithm for Mining Association Rules," *SIGMOD* 1995, pp. 175-186.
12. Perkowitz, M. and Etzioni, O. "Adaptive Web Sites: Automatically Learning from User Access Patterns," *Proceedings of WWW6.* 1997.
13. Srikant, R., and Agrawal, R. "Mining Generalized Association Rules," *Proc. Of the 21th Int'l Conference on Very Large Databases*, Santiago, pp. 407-419, Zurich, Swizerland, Sep. 1995.
14. Srikant, R., and Agrawal, R. "Mining Quantitative Association Rules in Large Relational Tables," *SIGMOD* 1996, pp. 1-12.
15. Srikant,R., and Agrawal, R. "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proc. Of the Fifth Int'l Conference on Extending Database Technology (EDBT)*, Avignon, France, March 1996.
16. Cormen, T.H., Leiserson,C.E., and Rivest, R.L. "Introduction to Algorithms," *The MIT Electrical Engineering and Computer Science Series*, 1992.
17. Zaiane, O. R., Xin, M.., and Han, J. "Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs," *Proc. Advances in Digital Libraries Conf. (ADL'98)*, Santa Barbara, CA, April 1998, pp. 19-29.
18. Zaki, M. J., Lesh,N., and Ogihara, M. "PlanMine: Predicting Plan Failures using Sequence Mining," *Artificial Intelligence Review*, 1999.

## 作者簡介

陳仕昇

1999 年中央大學資訊管理碩士，研究領域有資料挖礦，現職中央大學資管系博士班研究生。

許秉瑜

1995 年 UCLA 資訊工程博士，研究領域有 WWW 資料庫、資料挖礦、資料倉儲、ERP 等，多篇相關論文發表於國內外期刊上，現任教於中央大學企管系，並擔任中央大學管理學院電算中心主任及中央大學 ERP 中心主任。

陳彥良

1989 年清華大學資訊科學博士，1985 年清華大學工業工程碩士，1983 年成功大學工業管理系學士，研究領域有演算法、資料庫、資料倉儲、資料挖礦等，曾發表多篇文章於 OR、EJOR、Computers &OR、IPL、Trans Res B 等國際雜誌，並曾發表文章於管理學報、資訊管理學報、資訊管理研究等國內期刊，現任教於中央大學資管系。