行政院國家科學委員會專題研究計畫 成果報告

循序樣式探勘之研究與擴充
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 97-2221-E-032-046-
執 行 期 間 ：97 年 08 月 01 日至 98 年 07 月 31 日
執 行 單 位 ：淡江大學資訊管理學系

計畫主持人：周清江

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 98 年 10 月 31 日

行政院國家科學委員會補助專題研究計畫

☑ 成 果 報 告
□期中進度報告

循序樣式探勘之研究與擴充

計畫類別：☑ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 97-2221-E-032 -046
執行期間： 2008 年 8 月 1 日至 2009 年 7 月 31 日

計畫主持人：周清江
計畫參與人員： 陳昱辰、蔡宇維、林師晟、陳雅伶、連景華

成果報告類型(依經費核定清單規定繳交)：□精簡報告　☑完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
☑出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：淡江大學資訊管理系

中 華 民 國 98 年 10 月 31 日

目錄：

研究成果發表---------- 3
報告內容---------- 4
計畫成果自評----------27

研究成果發表：

期刊論文：

1. Chichang Jou, Hung-Chang Lee, "Handwritten numeral recognition based on simplified structural classification and fuzzy memberships," Expert Systems with Applications, Vol. 36(9), p 11858-11863, Nov 2009. (SCI, EI)

研討會論文：

1. 周清江、吳鎮成，"PMSPM：夾擊探勘最長循序樣式的演算法"，2008 年兩岸信息與管理學術研討會，廈門，中國，2008 年 7 月 22 日。

碩士論文：

1. 吳鎮成，"PMSPM：夾擊最長循序樣式探勘演算法"，2008 年 6 月。

報告內容

摘要

過去以 Apriori 為基礎的循序樣式探勘方法，皆單純採用由下而上的方式從長度較短的頻繁樣式逐步成長至較長的候選樣式，一直到沒有更長的候選樣式產生為止。在很多應用中，常需探勘“不為其它任一頻繁樣式的子樣式”之最長頻繁樣式。這些應用如依傳統方式，找出所有的頻繁樣式探勘候再刪除頻繁子樣式，會進行許多不必要的運算。本研究提出一個優先探勘最長頻繁樣式之演算法，PMSPM (Pincer Maximal Sequential Patterns Mining)，採取由下而上以及由上而下兩種方向輪流探勘的方式，提早於每次由上而下探勘時，找出該階段的最長頻繁樣式，使得後續由下而上探勘時，不需逐步成長，進而避免大量不需要的候選樣式測試，減少探勘時間。此外，交易的發生時間是決策者考量的重要因素之一，如果能在探勘時加上時間間隔的限制，產生的樣式將會更有意義，更具決策參考價值。本研究擴充項目之間只是相鄰和非相鄰的關係之 PMSPM 循序樣式探勘演算法，提出加上樣式項目之間的相關時間限制因素之 PMSPMTC (Pincer Maximal Sequential Patterns Mining with Time Constraint) 演算法，使得探勘出來的樣式更有意義且有彈性。本研究除了證明以上演算法之正確性與完整性，並以模擬實驗驗證其效能及可行性。

關鍵詞：循序樣式(sequential pattern)、資料探勘(data mining) 、最長循序樣式 (maximal sequential pattern)、時間限制 (time constraint)

1. 前言

循序樣式探勘可運用於顧客交易行為研究、網站日誌探勘、DNA 基因定序等等。以交易資料庫為例，可將交易的發生時間作為前後次序的依據，將顧客的所有交易依序排成一個序列，其目的是計算符合循序樣式要求的交易序列之出現次數，找出所有超過門檻值(即最小支持度)的頻繁循序樣式。以顧客交易資料為例，若發現許多顧客在購買了 A 物品之後，會再購買 B 物品，這就是一種頻繁循序樣式，目前循序樣式探勘的資料型別，主要是以交易型的資料庫為主，而由於電子商務市場的成熟，這種交易型資料量大幅增加，如何利用這些交易資料找出潛藏的資訊，也因此越來越受到重視。

循序樣式探勘的概念，最早是由 Agrawal 和 Srikant[5]提出，以原本運用於從資料庫中找出頻繁項目集合 (frequent itemset) 探勘的 Aprior 演算法[4]為基礎，以項目集合時間前後為順序依據，採取廣度優先且由下而上 (bottom-up) 的方式，由短的循序樣式逐步增加長度，產生較長的候選循序樣式，然後再從交易資料庫中找出符合項目集合順序的樣式，以最小支持度做為門檻值，過濾得到頻繁循序樣式。

一般的循序樣式探勘演算法都是探勘所有的頻繁樣式，因此可能會產生大量的頻繁子樣式，尤其是在探勘長的樣式時，其樣式數量非常龐大。因而便有研究提出探勘封閉式循序樣式(closed sequential pattern)，可以減少許多頻繁子樣式的產生，而探勘出來的樣式具有相同的表達意義。

然而當決策者想要探勘的結果只需為最長頻繁樣式(即不為其它任一頻繁樣式的子樣式)，且不需要詳細記錄其子樣式的支持度時，不論是一般探勘出所有頻繁樣式的方法或者是探勘封閉式循序樣式演算法，皆會產生大量非最長的頻繁循序樣式。本研究提出 PMSPM (Pincer Maximal Sequential Patterns Mining) 演算法，使用由下而上以及由上而下 (top-down) 兩種方向輪流探勘的方式，提早探勘出最長循序樣式，避免此樣式的大量子樣式探勘，省下其支持度之

運算，減少探勘時間。利用這兩種搜尋方式持續向中心點搜尋，最後便會得到所有的最長頻繁樣式。此外，本研究以 PMSPM 為基礎，擴充探勘具時間限制之循序樣式，進一步提升資料探勘在商業領域及科技領域上應用的可行性和實用性。演算法將於第三和第四章詳細說明，最後將以實驗的方式，測試演算法效能和評估實驗結果。

2. 相關研究

2.1 探勘方法

近年來循序樣式探勘演算法，以其探勘方式可分為兩大類：候選樣式產生與測試(candidate pattern generate-and-test) 和樣式成長方法 (pattern growth)：

2.1.1 候選樣式產生與測試

候選樣式產生與測試的方法是延伸 Apriori [4]演算法，由較短的循序樣式產生較長的候選循序樣式，再測試其候選循序樣式支持度是否滿足最小支持度，滿足則為頻繁循序樣式，接著持續不斷的一直產生及檢查越來越長的循序樣式。此方法有兩個缺點：(1)當頻繁樣式長度較長時，探勘過程中的候選樣式會呈指數型的增長，耗費大量的運算；(2)當探勘出的頻繁樣式之最長長度為 n 時，表示探勘時就必須至少掃描資料庫 n 次。

2.1.2 樣式成長方法

樣式成長方法最初來自 Pei 等人的 FP-Growth 演算法[10]，而後續研究為了再加快探勘速度，加入了資料投影 (data projection) 方法[12]，於探勘過程中建立投影資料庫 (projected database) 後，再探勘投影資料庫。投影資料庫資料量會在各回合的探勘中逐漸縮小，將搜尋空間限定在較小的範圍裡，進而提昇探勘效率。其方法最主要成本在於建構投影式資料庫，需耗費額外的運算及儲存空間。

2.2 最長循序樣式相關演算法

此節討論與最長循序樣式相關的探勘議題：探勘最大項目集合**(maximal itemset)**以及探勘封閉式循序樣式。

2.2.1 最大項目集合演算法

一般用來發掘頻繁項目集的演算法都是廣度優先且為由下而上搜尋。然而當頻繁項目集中的項目數量較大時，效能會大幅降低。Lin 及 Kedem 提出 Pincer-Search 演算法[9]，採取由下而上及由上而下雙方向輪流夾擊的搜尋方式。由下而上搜尋是以類似 Apriori 演算法的方式，逐步成長項目集合，而由上而下搜尋是先將所有的項目組成一個最大候選項目集合，然後將由下而上方向中所找出的非頻繁項目集合，依據“非頻繁項目集合的超集合也必定是非頻繁集合”定理，用來更新由上而下方向的最大候選項目集合，以產生新的最大候選項目集合。而當最大候選項目集合中的某一項目集合已經被確認是頻繁，則可依據“頻繁項目集合的子集合，也必定是頻繁項目集合”定理，在後續由下而上方向探勘中，便不再需要處理此最大頻繁項目集合的所有子集合。利用這種搜尋方式持續向中心點搜尋，就像是夾子逐漸夾緊一樣，當所有的最大頻繁項目集合都找出後，搜尋便可以結束，此演算法一個非常重要的特色是不需要去計算每一個頻繁項目集合的支持度。

2.2.2 封閉式循序樣式演算法

一般的循序樣式探勘都是要找出所有的頻繁序列，因此當存在較長的頻繁序列時，會產生數量龐大的頻繁子序列，於是 Yan 等人提出了探勘封閉式循序樣式的演算法 CloSpan[13]，當一個頻繁樣式 I 為封閉時，代表在此資料庫中不存在一個 I 的超樣式且和 I 有著相同的支持度。不再探勘所有的頻繁子序列，探勘封閉式循序樣式可以減少許多頻繁子序列的產生，而探勘出來的樣式也一樣具有相同的表達意義。本研究探討之最長循序樣式為封閉式循序樣式的一種，但封閉式循序樣式不一定是最長循序樣式。

2.2.3 增加時間限制資訊之循序樣式

為了讓循序樣式更能應用於實務上， GSP[6]、cSPADE[14]、Prefix-growth[10] 等演算法均曾考慮過加入相關時間限制，以找出符合時間限制條件之所有頻繁項目集合或循序樣式。

張耕[2] 則提出一個結合 PrefixSpan 演算法[12]與時間機率觀念的演算法 PCTP (PrefixSpan Considering Time Probability)，在原有的最小支持度門檻值以外，新加入間隔時間和最小機率門檻值兩個參數，用來計算與篩選出使用者感興趣的時間機率頻繁樣式。

顏志祐[3] 提出 E-PrefixSpan 演算法，運用樣式成長方法的特性，利用每次已找出的頻繁循序樣式往後增長的同時，加入信賴度的篩選規則，讓每次新產生的樣式與其父樣式間的比值都要大於等於由使用者設定的最小信賴度，通過信賴門檻值才進一步計算項目間所代表的時間間隔關係，以找到符合最小支持度之最小時間間隔。

Chen 和 Hu[8] 提出 CFR-PostfixSpan 演算法，除了傳統的頻繁 (frequency) 度條件，在樣式裡加上交易最近的時間 (recency) 以及樣式的最大時間間隔區間 (compact) 等時間因素，也就是除了最小支持度 (minimum support) 外，利用交易最近時間確認過去常出現的頻繁樣式在最近的時間點是否不再出現，利用樣式的最大時間間隔區間 (max_length) 觀察符合樣式的交易第一個項目和最後一個項目之時間差是否超出最大時間間隔區間。

3. PMSPM 演算法

3.1 背景

以Apriori為基礎的演算法，其在探勘長度較長的樣式時，候選樣式數量皆會大量的增加造成沉重的運算成本。假設在第 k 回合的探勘時，得到 n 個長度為 k 的 k-階頻繁樣式，則在第 k+1 回合時則會產生 n×n+n×(n-1)/2 個長度為 k+1 的 (k+1)-階候選樣式。例如 k 階頻繁樣式有1000個，則 k+1 階候選樣式就有 1499500 個，其數量是呈指數型的成長。而且每一回合皆需掃描原始資料庫來計算支持度。當最後探勘出的最長頻繁樣式的長度為 L 時，這些演算法必需掃描 L 次原始資料庫。有鑒於 Apriori 為基礎的演算法皆有著上述的瓶頸，本研究提出了一個夾擊式探勘、提早找出最長循序樣式的演算法 PMSPM (Pincer Maximal Sequential Patterns Mining)，不需由下而上逐步執行至最後一回合，就可找出所有的最長循序樣式，掃描資料庫的次數也會比較小，因此可避免後續產生大量的候選樣式，以及減少掃描資料庫來測試支持度的成本。

3.2 演算法定義

本研究採用交易資料庫為範例，包含全部顧客的所有交易資料，一個顧客的所有交易依照交易時間先後順序，視為一個交易序列 (sequence)，而每一筆序列至少包含一筆交易 (transaction)，一筆交易包含由至少一個項目 (item) 所組成的一個項目集合 (itemset)。在此先定

義何謂項目、交易、交易序列、交易資料庫、循序樣式(sequential pattern)、支持度(support)、頻繁循序樣式(frequent sequential pattern)、最長頻繁循序樣式(maximal frequent sequential pattern)：

- 項目：代表一個獨立且可被交易的單位個體，以符號 i 表示 。以下假設資料庫中存在 m 個相異的項目，則 $I=\{i_1,i_2,...,i_m\}$ 就是由資料庫中所有項目組成的集合。
- 交易：為 I 的子集合，由一個或多個項目所組成，用來代表顧客在某時間點的交易內容。令交易 T 中有 u 個項目，$T=\{i_{j_1},...,i_{j_u}\}$，則對於所有介於 1 至 u 之整數 r 均滿足以下條件： $1\le j_r\le m$。在一般的演算法中，為求加快比對速度，T 中的多個項目會依照字彙次序由小到大排序。
- 交易序列：由一個顧客所有交易所組成的序列，且序列中交易的先後次序即代表交易時間之順序。以下令交易序列 $S=\langle T_1,T_2,...,T_n\rangle$。
- 交易資料庫：由一筆或多筆交易序列組成。以下令交易資料庫 $D=\{S_1,S_2,...,S_d\}$。
- 循序樣式：由一個或多個有先後次序性的項目集合所組成。令循序樣式 $P=\langle i_1,i_2,...,i_q\rangle$，若有一交易序列 $S=\langle T_1,T_2,...,T_n\rangle$，滿足 $i_1\subseteq T_{P_1}$、$i_2\subseteq T_{P_2}$、…、$i_q\subseteq T_{P_q}$ 且 $1\le P_1<P_2<...<P_q\le n$，則稱此顧客交易序列 S 支持循序樣式 P，請注意在此定義下，循序樣式中相鄰的項目集合在支持該樣式的序列中不一定要相鄰。
- 支持度：循序樣式 P 的支持度即為支持 P 的交易序列數量。
- 頻繁循序樣式：以最小支持度作為頻繁循序樣式之門檻值，所有超過最小支持度的循序樣式即稱為頻繁循序樣式。
- 最長頻繁循序樣式：不為其它任一頻繁循序樣式的子樣式的頻繁循序樣式。

經由上述的定義說明，本演算法的目的即在於從交易資料庫中，找出所有符合最小支持度的最長頻繁循序樣式。

而交易型資料庫的序列組成方式，根據 Chen 等人 [7] 及許俊傑[1] 可分類成下述三種：

1. 單一交易項目集組成(single transaction multiple itemset)
   將交易視為一個序列，而交易的內容包括多個項目；例如序列 S=〈(ABC)〉，序列只有 1 筆交易，有 3 個項目 A、B、C。
2. 多筆交易單一項目組成(multiple transaction single item)
   序列是由多筆交易所組成，但每筆交易的內容只有一個項目；例如序列 S=〈(A)(E)(B)(D)(A)〉，序列有 5 筆交易，每筆交易的項目依序為 A、E、B、D、A。
3. 多筆交易項目集組成(multiple transaction multiple itemset)
   結合上述兩種的組成方式，一個序列包括多筆交易，每一筆交易包括多個項目；例如序列 S=〈(ABC)(AD)(BC)(D)〉，序列有 4 筆交易，第 1 筆交易項目為 A、B、C，第 2 筆交易項目為 A、D，第 3 筆交易項目為 B、C，第 4 筆交易為 D。

本研究著重在於提出一個新的方法來探勘最長頻繁循序樣式，先採用較不複雜的多筆交易單一項目的組成方式進行探勘，後續擴充成多筆交易項目集組成方式並不難達成。

由於本研究採用多筆交易單一項目的組成方式進行探勘，因而將前述之交易及循序樣式重新定義如下：

- 交易：為 I 的子集合，由一個項目所組成，代表顧客在某時間點的交易內容。令交易 T 中有一個項目，$T=\{i_e\}$，則對於整數 e，需滿足以下條件：且 $1\le e\le m$。
- 循序樣式：由一個或多個有先後次序性的項目所組成。令循序樣式 $P=\langle i_1,i_2,...,i_q\rangle$，

若有一交易序列 $S=\langle T_1,T_2,...,T_n\rangle$，滿足 $i_1=T_{P_1}$、$i_2=T_{P_2}$、…、$i_q=T_{P_q}$ 且 $1\le P_1<P_2<...<P_q\le n$，則稱此顧客交易序列 S 支持循序樣式 P。

而為了後續說明方便，往後討論將循序樣式簡稱樣式，交易序列簡稱序列。

為了盡量提早找到最長頻繁樣式，本論文設計了兩個主要的資料結構，候選最長序列集合(Candidate Maximal Sequence Set，以下簡稱 CMS)及候選最長頻繁循序樣式集合(Candidate Maximal Frequent Pattern Set，以下簡稱 CMFS)，其說明如下：

候選最長序列集合：此集合中的序列有可能在往後各回合的探勘，成為最長頻繁樣式。

候選最長頻繁循序樣式集合：此集合中的樣式皆為當下已知的最長頻繁樣式，當演算法結束後，CMFS 即為所有最長循序樣式的集合。

PMSPM 演算法如圖 1 所示：

```
Algorithm PMSPM(D, minSup)
// D 為交易資料庫
// minSup 為最小支持度
// CMS 為候選最長序列集合、 CMFS 為候選最長頻繁樣式集合
// k 為目前所成長的樣式長度
// Ck 為長度為 k 的候選樣式集合
// Fk 為長度為 k 的頻繁樣式集合、Nk 為長度為 k 的非頻繁樣式集合
1   begin
2     k = 1;
3     Gen_1st_Pattern()；      //產生一階頻繁、非頻繁樣式及 CMS
4     while ( CMS != ∅ )       //CMS 若為空集合則結束演算法
5     {
6       k++;     //開始第 k 階之探勘
        //由下而上方向之探勘過程
7       Ck = Gen_Candidate_k_Pattern(Fk-1);     //合併產生 k 階候選樣式
8       Count_Candidate_Sup(Ck);     //計算 k 階候選樣式的支持度
        //由上而下方向之探勘過程
        //以 k 階非頻繁樣式來更新 CMS 中的序列
9       CMS = UpdateCMS(CMS, Nk);
        //計算 CMS 中所有序列的支持度,滿足最小支持度則加入至 CMFS
10      Count_CMS_Sup(CMS);
11      Del_Sub(CMFS);     //刪除 CMFS 中的子樣式
12    }
13    Output CMFS;
14  End
```

圖 1. PMSPM 演算法

PMSPM 演算法包含了下列五個步驟：

1. 掃描資料庫，以產生一階候選樣式，將所有顧客的原始序列組成 CMS，接著將一階頻繁樣式加入 CMFS，此步驟即圖 1 中的 Gen_1st_Pattern 函數。
2. 產生目前所執行第 k 回合的 k-階候選樣式，建立候選樹 (candidate tree) 結構來測試支持度，並將所有 k 階頻繁樣式加入 CMFS，此步驟即圖 1 中的 Gen_Candidate_k_Pattern 及 Count_Candidate_Sup 函數。
3. 利用於步驟 2 中得到的非頻繁樣式來更新 CMS 中的序列，並計算得到的 CMS 各序

列之支持度，如果其支持度大於門檻值，則從 CMS 中刪除並放入 CMFS 中，此步驟即圖 1 中的 UpdateCMS 及 Count_CMS_Sup 函數。

4. 刪除 CMFS 中滿足以下條件之樣式：“為另一 CMFS 樣式的子樣式”。對於不滿足以上條件、且長度大於 k+1 的樣式，則找出其長度為 k+1 的子樣式並加入至 k+1 階(下一階段)頻繁樣式集合，減少下一階段於步驟(2)的支持度計算成本，此步驟即圖 1 中的 Del_Sub 函數。
5. 重覆步驟 2 至步驟 4，成長候選樣式，直到 CMS 為空集合時便終止演算法。CMFS 中的樣式即為所有最長頻繁樣式。

以上步驟將於後續小節詳細說明。

3.3 產生一階頻繁、非頻繁樣式及 CMS

此步驟位於圖 1 中第 3 行，在此掃描資料庫並計算每一項目的支持度，若滿足最小支持度則將此項目加入一階頻繁樣式集合以及 CMFS，若不滿足則加入至一階非頻繁樣式集合。將所有的原始交易序列移除一階非頻繁樣式的項目，然後再逐一加入至 CMS 中。為了方便說明，後續將 k-階頻繁樣式集合簡稱為 F_k、k-階非頻繁樣式集合簡稱為 N_k、k-階候選樣式集合簡稱為 C_k。

3.4 由下而上方向的探勘

在此階段主要是使用類似 GSP 演算法[6] 成長候選樣式的方法，利用兩個第 k-1 階的頻繁樣式來合併產生第 k 階的候選樣式，並將所有候選樣式建成候選樹的結構，以完成支持度的計算。當完成此階段以及後續第 3.5 – 3.7 節中的處理後，如果尚未找出全部的最長頻繁樣式，則會再次從此階段將 k 值加 1，持續進行探勘。

3.4.1 產生 k 階候選樣式

透過合併產生候選樣式之方法如下：在 F_{k-1} 中取出兩個樣式，假設分別為 S_1=〈X_1 X_2…X_{k-2} X_{k-1}〉以及 S_2=〈Y_1 Y_2…Y_{k-2} Y_{k-1}〉。當 k=2 時，直接合併產生長度為 2 的〈X_1Y_1〉並加入 C_k；但是當 k≥3 時，則需滿足 $X_2=Y_1, X_3=Y_2, \ldots, X_{k-1}=Y_{k-2}$，才允許合併產生長度為 k 的〈$X_1$ X_2…X_{k-1} Y_{k-1}〉的候選樣式並加入 C_k。例如，當 k=3，S_1 為〈A<u>BD</u>〉而 S_2 為〈<u>BD</u>A〉時，則可合併成 4 階候選樣式〈A<u>BD</u>A〉。

3.4.2 計算 k 階候選樣式支持度

計算 k 階候選樣式支持度的詳細做法如圖 2 所示，會將上節所產生的候選樣式逐一加入至候選樹中；候選樹上的每個節點都有一計數器以及所代表的樣式，當每讀入一筆交易序列走訪(traverse) 候選樹時，會將所走訪過的節點計數器加 1，直到將資料庫中所有的交易序列都讀入過後便完成支持度的計算。在將 k 階候選樣式加入候選樹之前，會先判斷此樣式是否已存在於 k 階頻繁樣式集合或 k 階非頻繁樣式集合中，如果已存在，則不需再將此候選樣式加入至候選樹以避免不必要的走訪。當走訪完畢後，若在候選樹上的葉節點之計數器的值滿足最小支持度，則將此葉節點上代表的樣式加入 k 階頻繁樣式集合以及 CMFS，不滿足則加入至 k 階非頻繁樣式集合。

```
Function Count_Candidate_Sup(Ck)
1   begin
    // CTree 為候選樹
2     CTree = ∅;      //重新建立一個空的候選樹
3     for ( each S∈Ck ) {
        //若候選樣式 S 尚未確定為頻繁或非頻繁，則加入候選樹
4       if ( S∉Fk && S∉Nk )   CTree.add(S);
5     }
      //讀入交易序列走訪候選樹計算支持度
6     Traverse(D, CTree);
7     for ( each leaf node O in CTree){
8       if ( O.counter ≥ minSup){
9         Fk.add(O.pattern);
10        CMFS.add(O.pattern);
11      }else{
12        Nk.add(O.pattern);
13      }
14    }
15  end
```

圖 2. 計算候選樣式支持度函數

3.5 由上而下方向的探勘

在此階段主要是利用在第 3.4.2 節所探勘出的 N_k 中的樣式，裂解 CMS 中的序列，以更新 CMS，提早找出長度大於 k 的頻繁樣式。

3.5.1 更新 CMS

CMS 為候選最長序列集合，其中的序列皆有可能在每一次更新後成為最長頻繁樣式，因此在此階段是利用已知的 N_k 中的樣式，依據“非頻繁樣式的超樣式也必定是非頻繁樣式”定理，裂解 CMS 中的序列，目的是將 CMS 中的序列更新成非任何「已知」非頻繁樣式的超序列，使其序列更有可能成為最長頻繁樣式，而每一次裂解皆會使原本的序列長度減一。

此步驟詳細做法如圖 3 所示。對於 N_k 中的每一個樣式 S_N＝〈T_1…T_k〉，若 CMS 中存在一長度為 m 的 S_N 之超序列 S_C，且存在整數 I_1…I_k 分別表示 T_1…T_k 在 S_C 中的索引位置。此處之 m 必須大於 k+1，因為若 m 小於等於 k+1，則裂解後的序列長度必小於等於 k，皆為已知樣式，因而不須要再更新。根據“非頻繁樣式的超樣式也必定是非頻繁樣式”定理，S_C 必定不是頻繁樣式，必須將 S_C 更新成有可能是最長頻繁樣式的序列，同時將 S_C 從 CMS 中移除。由於 S_N 是由 k-1 階頻繁樣式〈T_1…T_{k-1}〉跟〈T_2…T_k〉合併而成，且〈T_1…T_{j-1} T_{j+1}…,T_k〉($2 \le j \le k-1$) 有可能是頻繁樣式，因此必須將原本 S_C 中的 k 階非頻繁樣式 S_N 的部分，更新成下述三種 k-1 階頻繁樣式後成為新的 S_C，再加入 CMS。依照移除交易項目位置的不同，更新規則分為下列三種情形：

(1) 移除 S_C 中索引位置在 I_2 前的交易 T_1，如圖 3 中第 6 行。
(2) 移除 S_C 中索引位置在 I_{k-1} 後的交易 T_k，如圖 3 中第 11 行。
(3) 當〈T_1…T_{j-1} T_{j+1}…T_k〉($2 \le j \le k-1$)是 k-1 階頻繁樣式時(若為非頻繁樣式，則更新後亦必定非頻繁樣式，因此不需進行更新)，移除 S_C 中索引位置在 I_{j-1} 及 I_{j+1} 中間的交易 T_j，如圖 3 中第 7 至 10 行。

若無完整的考慮此三種情形的更新，將會導致演算法結束時 CMFS 遺失部分的最長頻繁樣式。

```
Function UpdateCMS(CMS, N_k)
begin
  for ( each S_N ∈ N_k ){
  // S_N = 〈T_1…T_{k-1} T_k〉
    for ( each S_C ∈ CMS ){
      if ( S_N.legnth > k+1 && S_N is a subsequence of S_C ) {
      //exist integers I_1…I_{k-1} I_k to represent index of T_1…T_{k-1} T_k in S_C
        Remove S_C From CMS;
        CMS.add(Remove all T_1 located the index before I_2 in S_C);
        for ( j = 2 to k-1){
          if ( 〈T_1…T_{j-1} T_{j+1}…T_k〉 ∈ F_{k-1})
            CMS.add(Remove all T_j located between index I_{j-1} to I_{j+1} in S_C);
        }
        CMS.add(Remove all T_k located the index after I_{k-1} in S_C);
      }
    }
  }
  Return CMS;
end
```

圖 3. 更新 CMS 序列函數

3.6 計算 CMS 中序列的支持度

計算 CMS 中序列的支持度的詳細做法如圖 4。目前在 CMS 中的序列的長度皆大於 k，且有可能在第 3.5.1 節已更新成最大頻繁樣式，因而在此需先判斷序列 S(長度為 L)是否存在於 CMFS 中，若不存在，再開始以第 3.4.2 節建立候選樹的方式，計算其支持度，以避免不必要的運算；若滿足最小支持度，則從 CMS 中移除 S，並將此 S 加入 L 階頻繁樣式集合，以減少往後在第 L 階時於由下而上方向計算支持度，同時亦將 S 加入至 CMFS；若不滿足最小支持度，且 L 等於 k+1 的話，則將 S 從 CMS 中移除，並加入 L 階非頻繁樣式集合，原因是此 S 以及其所有子樣式皆確定為非頻繁樣式，不需在 CMS 中等待後續的更新。

```
Function Count_CMS_sup(CMS)
begin
  // CTree 為候選樹
  CTree = ∅;          //重新建立一個空的候選樹
  for ( each S ∈ CMS ) {
    if ( S ∉ CMFS )   CTree_CMS.add(S);      //將 S 加入候選樹
  }
  Traverse(D, CTree);      //讀入交易序列走訪候選樹計算支持度
  for ( each node O in CTree){
    if ( O.counter ≥ minSup ){
      Remove O.pattern From CMS;
      CMFS.add(O.pattern);
      F_pattern.length.add(O.pattern);
    }else if (O.pattern.length == k+1 ){
      Remove O.pattern From CMS;
      N_pattern.length.add(O.pattern);
    }
  }
end
```

圖 4. 計算 CMS 中序列支持度函數

3.7 刪減 CMFS 中的子樣式並產生未被刪除樣式的長度 k+1 子頻繁樣式

PMSPM 演算法的目的是在求最長頻繁樣式，因而必須刪除 CMFS 中滿足以下條件的樣式：“為另一 CMFS 樣式之子樣式”。詳細做法如圖 5，說明如下：對於 CMFS 中的每一個序列 S_1，若在 CMFS 中亦存在一序列 S_2，且 S_1 為 S_2 的子序列，則將 S_1 從 CMFS 中移除；而未被刪除的樣式即為候選最長頻繁樣式，接著使用 Generate_sub 函數來產生長度大於 k+1 的最長頻繁樣式其長度為 k+1 的所有子樣式，並將這些子樣式加入至 k+1 階頻繁樣式集合，這樣在第 k+1 階測試候選樣式支持度時，就不需再去測試已知的樣式，減少測試的成本。

```
Function Del_Sub(CMFS)
begin
  for ( each S1∈CMFS ) {
    if ( ∃S2∈CMFS ∋ S1 is a subsequence of S2 ){
      Remove S1 From CMFS;
    }else if ( S1.length＞k+1 ){
      SubS = Generate_sub(S1, k+1);
      for ( each s∈SubS )   Fk+1.add(s);
    }
  }
end
```

圖 5. 刪除 CMFS 的子樣式函數

3.8 結束演算法

當 CMS 為空不存在任何序列時，則表示已無任何候選最長序列在等待更新，此時 CMFS 即為所有的最長頻繁樣式。

4. PMSPMTC 演算法

4.1 背景

在循序樣式探勘的實務上，樣式中相鄰的兩項目之時間間隔可能早就超過行銷的時效性，對於了解顧客購物行為的重要性，反而不如次序上不相鄰但時間間隔較短的兩項目。因此，我們提出一個 PMSPMTC (Pincer Maximal Sequential Patterns Mining with Time Constraint) 演算法，擴充 PMSPM 演算法，加上時間限制因素現，使得項目之間不再只是相鄰和非相鄰的關係，而是有具體的時間限制。在作法上我們參考 Chen and Hu [7]，除了原有之最小支持度作為頻繁循序樣式之門檻值，另外加上樣式最前與最後項目的最大時間間隔限制、交易時間不能過久的條件，使得探勘出來的樣式更具意義。例如可找出：「顧客在某一天買項目A，二天後買項目B，再三天後買項目C」的樣式。

4.2 演算法定義

在 PMSPMTC 演算法中，除了原本採用的交易資料庫，本研究同時也建立一個對應交易序列的交易時間資料庫，包含全部顧客的所有交易時間，因此新增了交易時間資料庫，並修改頻繁循序樣式的定義:

- 交易時間資料庫：由一筆或多筆交易時間序列組成。以下令交易時間資料庫

$T = \{ St_1, St_2, ..., St_d \}$ 。

- 頻繁循序樣式：以頻繁樣式最小支持度 (minimum support) 和時間限制最小支持度 (compact minimum support) 作為頻繁循序樣式之門檻值，頻繁循序樣式必須超過最小支持度和時間限制最小支持度。

PMSPMTC 演算法也是如圖 1，只是增加 T 為交易時間資料庫、C_minSup 為時間限制的最小支持度這兩個輸入資料，以下將指出 PMSPMTC 與 PMSPM 不同之處。

PMSPMTC 演算法的第一個步驟需修改為：

1. 掃描交易資料庫以及交易時間資料庫，增加時間限制的最小支持度 (C_minSup) 之判斷，以產生一階候選樣式，將所有顧客的原始序列組成 CMS，接著將一階頻繁樣式加入 CMFS。

PMSPMTC 演算法在計算候選樣式支持度和 CMS 中序列支持度的 Count_Candidate_Sup 方法中，當每讀入一筆交易序列走訪候選樹時，會再判斷時間交易序列的交易最近時間 (recency) 和項目時間的最大間隔限制 (max_length) 是否皆符合門檻值，才將所走訪過的節點計數器加 1，直到將資料庫中所有的交易序列都走訪過後便完成支持度的計算；此步驟位於圖 6 中第 6 行和圖 7 中第 7 行。

PMSPMTC 演算法當走訪完候選樹，如果在候選樹上的葉節點之計數器的值滿足頻繁樣式最小支持度以及時間限制最小支持度，則將此葉節點上代表的樣式加入 k 階頻繁樣式集合以及 CMFS，不滿足則加入至 k 階非頻繁樣式集合。此步驟位於圖 6 中第 8 行和圖 7 中第 9 行。

```
Function Count_Candidate_Sup(Ck)
1   begin
    // CTree 為候選樹
2     CTree = ∅;      //重新建立一個空的候選樹
3     for ( each S ∈ Ck ) {
        //若候選樣式 S 尚未確定為頻繁或非頻繁，則加入候選樹
4       if ( S ∉ Fk && S ∉ Nk )   CTree.add(S);
5     }
      //讀入交易序列走訪候選樹計算支持度
6     Traverse(D, T, CTree);
7     for ( each leaf node O in CTree){
8       if ( (O.counter ≥ minSup)&&( O.counter ≥ C_minSup)){
9         Fk.add(O.pattern);
10        CMFS.add(O.pattern);
11      }else{
12        Nk.add(O.pattern);
13      }
14    }
15  end
```

圖 6. PMSPMTC 計算候選樣式支持度函數

```
Function Count_CMS_sup(CMS)
begin
  // CTree 為候選樹
  CTree = ∅;          //重新建立一個空的候選樹
  for ( each S∈CMS ) {
    if ( S∉CMFS )   CTree_CMS.add(S);      //將 S 加入候選樹
  }
  Traverse(D, T, CTree);      //讀入交易序列和交易時間序列走訪候選樹計算支持度
  for ( each node O in CTree){
    if ( (O.counter ≥ minSup)&&( O.counter ≥ C_minSup)){
      Remove O.pattern From CMS;
      CMFS.add(O.pattern);
      F_pattern.length.add(O.pattern);
    }else if (O.pattern.length == k+1 ){
      Remove O.pattern From CMS;
      N_pattern.length.add(O.pattern);
    }
  }
End
```

圖 7. PMSPMTC 計算 CMS 中序列支持度函數

5. PMSPM 系統實作

5.1 實驗環境和模擬資料庫產生

本節敘述我們進行的 PMSPM 演算法實驗，在不同資料庫參數下，取得本演算法的執行時間長短，並與 Apriori-based 演算法做比較。實驗機器是使用 Intel Core 2 Duo T2300e 處理器，2048MB DDRII 667 主記憶體，實驗環境是在 Windows XP 作業系統下進行，使用 Java (JDK 1.6.0) 程式語言開發。

我們實驗使用的資料集，是與 Agrawal 等人[4]相同的產生方式，

表 1 為參數說明。由於本研究是對於頻繁樣式長度較長的資料做探勘，根據我們觀察產生的資料集，模擬資料集中產生的序列雖然較長，但序列中項目之間的次序過於混亂無一定順序性，無法形成較長的頻繁樣式，因此需將序列間的信賴度參數設為 1，使得序列中的項目之間較具有一定或相似的順序性。接著設定最長樣式種類數(S)，此參數簡述如下，假使設定最長樣式種類數為 100，交易序列數量為 25000，則模擬資料庫會產生約 100 種左右相異的序列，且其每一種的數量平均約 250 筆序列左右，而這 250 筆之間彼此的相似度較高。因此，我們利用設定序列間的信賴度以及設定最長樣式種類數這兩種參數的方式，使得模擬資料庫能存在較長的頻繁樣式。

表 1. PMSPM 實驗參數說明

參數	參數說明
T	交易序列平均交易數量
D	交易序列數量
S	最長樣式種類數量
N	項目數量

PMSPM 實驗將分為下述四個部分：

第一部分，實驗在交易序列的平均交易數量(T)不同的情形下，比較 PMSPM 與只使用由下而上方向探勘的 Apriori-based 演算法其執行效能，目的在於證明本演算法在交易序列的平均交易數量較長時，其執行效能會比 Apriori-based 演算法的效能來的好，模擬資料庫參數如表 2 所示。

表 2. PMSPM 實驗之第一部分參數

實驗	資料庫參數	最小支持度(%)
第一部分 (T)	D50k N1000 **T7** S100	0.05
	D50k N1000 **T8** S100	0.05
	D50k N1000 **T9** S100	0.05
	D50k N1000 **T10** S100	0.05

第二部分，實驗在不同的交易序列數量(D)下，PMSPM 的執行時間是否有良好的加速比，模擬資料庫參數如表 3 所示。

表 3. PMSPM 實驗之第二部分參數

實驗	資料庫參數	最小支持度(%)
第二部分 (D)	**D10k** N1000 T10 S100	0.05, 0.1, 0.15
	D25k N1000 T10 S100	0.05, 0.1, 0.15
	D50k N1000 T10 S100	0.05, 0.1, 0.15

第三部分，實驗在最長樣式種類(S)的數量多寡下，PMSPM 執行時間的變化，模擬資料庫參數如表 4 所示。

表 4. PMSPM 實驗之第三部分參數

實驗	資料庫參數	最小支持度(%)
第三部分 (S)	D50k N1000 T10 **S50**	0.05, 0.1, 0.15
	D50k N1000 T10 **S100**	0.05, 0.1, 0.15
	D50k N1000 T10 **S150**	0.05, 0.1, 0.15

第四部分，比較在不同的項目個數下(N)，所花費的執行時間，模擬資料庫參數如表 5 所示。

表 5. PMSPM 實驗之第四部分參數

實驗	資料庫參數	最小支持度(%)
第四部分 (N)	D50k **N500** T10 S100	0.05, 0.1, 0.15
	D50k **N1000** T10 S100	0.05, 0.1, 0.15
	D50k **N1500** T10 S100	0.05, 0.1, 0.15

5.2 實驗結果與討論

第一部分，於圖 8 中可發現一開始在長度較短時，由於頻繁樣式的平均長度較短，Apriori-based 的效率比 PMSPM 較佳，但是隨著平均交易數量的增加，在圖 9 中可明顯看出頻繁樣式的平均長度也相對增長，導致在由下而上方向中的候選樣式數量開始呈指數型增加，使用 Apriori-based 方法則需耗費大量的運算，而相對於 PMSPM 雖然在初期時需比 Apriori-based 額外耗費更新最長候選序列集的成本，但是之後卻能早一步的將最長頻繁樣式找出，進而優先完成探勘動作，節省中間大量的候選樣式檢查與測試。

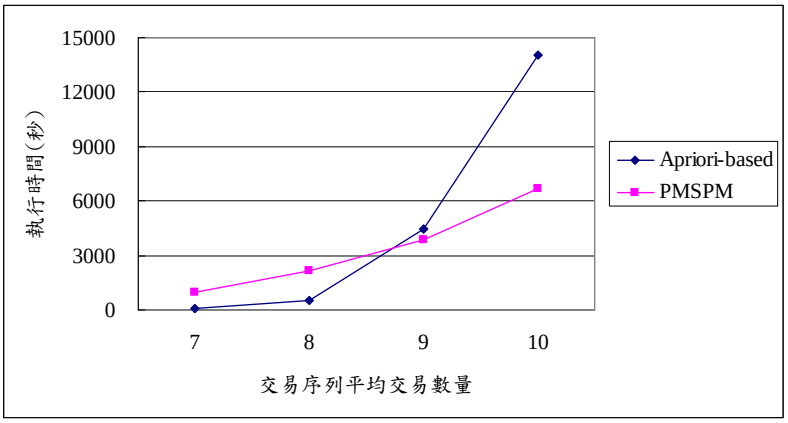

圖 8. PMSPM 交易序列平均交易數量在不同方法下執行時間的比較

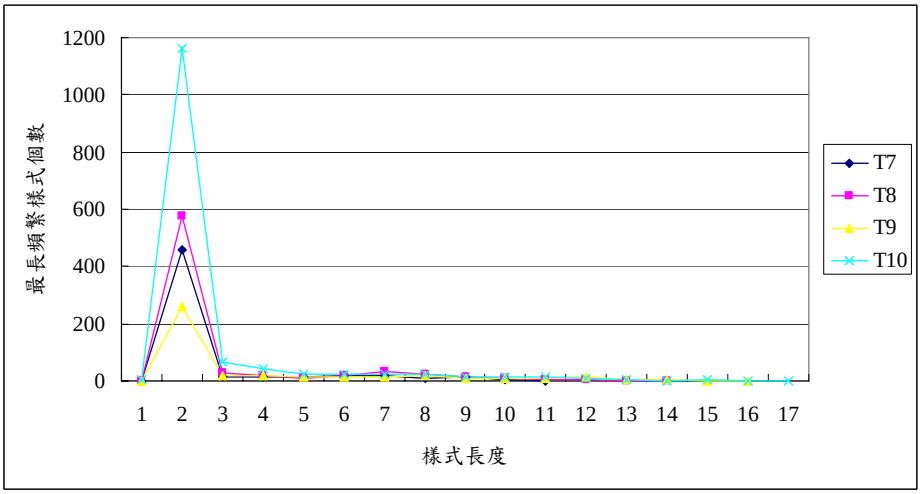

圖 9. PMSPM 各長度的最長樣式個數在不同交易序列平均交易數量下的分佈

第二部分，於圖 10 中可以看出 PMSPM 演算法在不同的支持度下，隨著交易序列數的增加，在執行時間上並沒有大幅地增加的情形，具有良好的加速比；並由圖 11、12、13 中最長樣式個數的分佈情形也發現，當交易序列數增加的同時，由於較長的交易序列數量也隨之增多，因而最長頻繁樣式主要的分佈長度也會越長，例如在圖 12 中，當 D10k 時最主要分佈在長度為 6，而 D25k 跟 D50k 時主要集中在交易序列平均交易數附近的長度 9，且 D50k 分佈在大於長度 9 的最長頻繁樣式數量又比 D25k 來得更多。

另一方面，由圖 11、12、13 可明顯得知，最長頻繁樣式的最長長度會隨著門檻值的增加而呈反向減少，符合我們預期的結果。

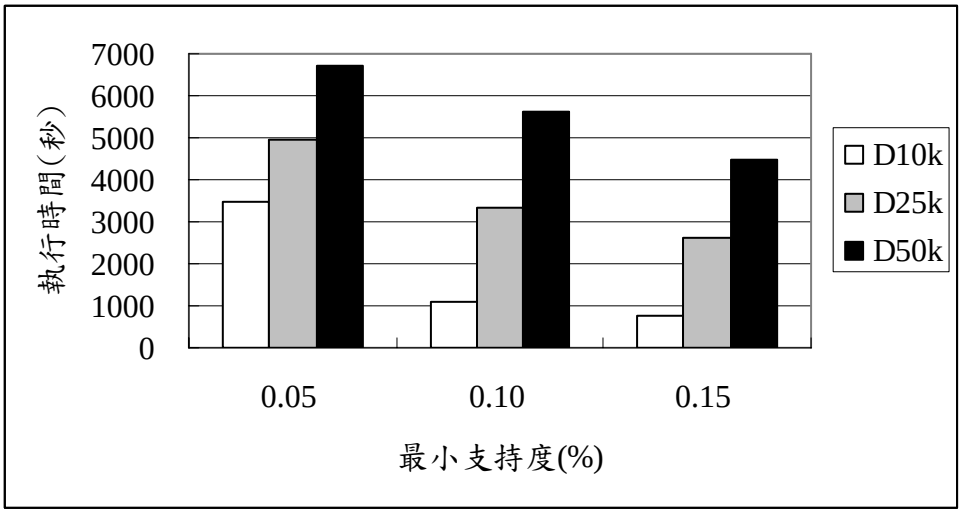

圖 10. PMSPM 交易序列數量和執行時間的比較

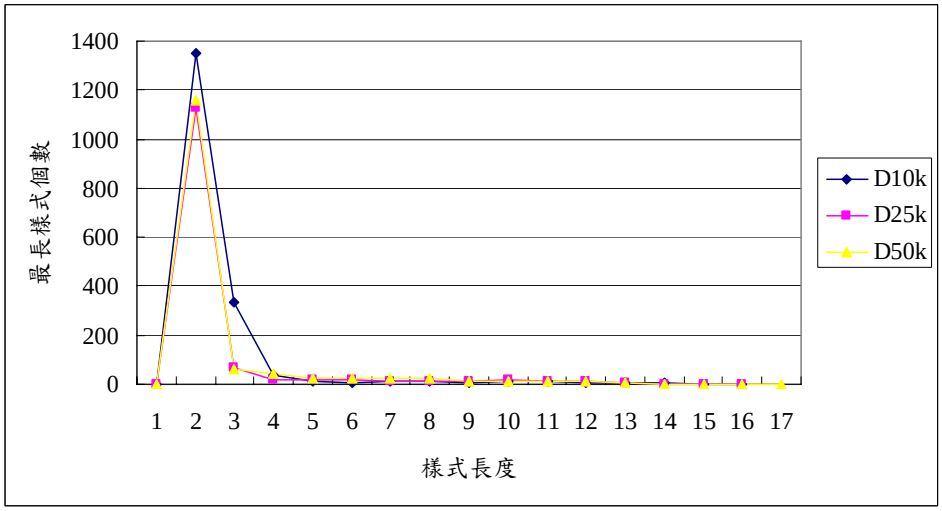

圖 11. PMSPM 各長度的最長樣式個數在支持度為 0.05%下的分佈

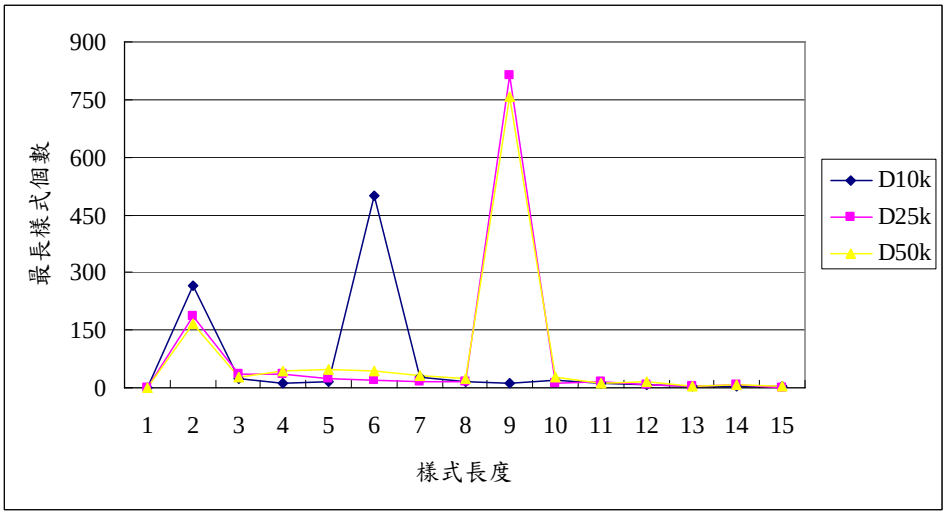

圖 12. PMSPM 各長度的最長樣式個數在支持度為 0.10%下的分佈

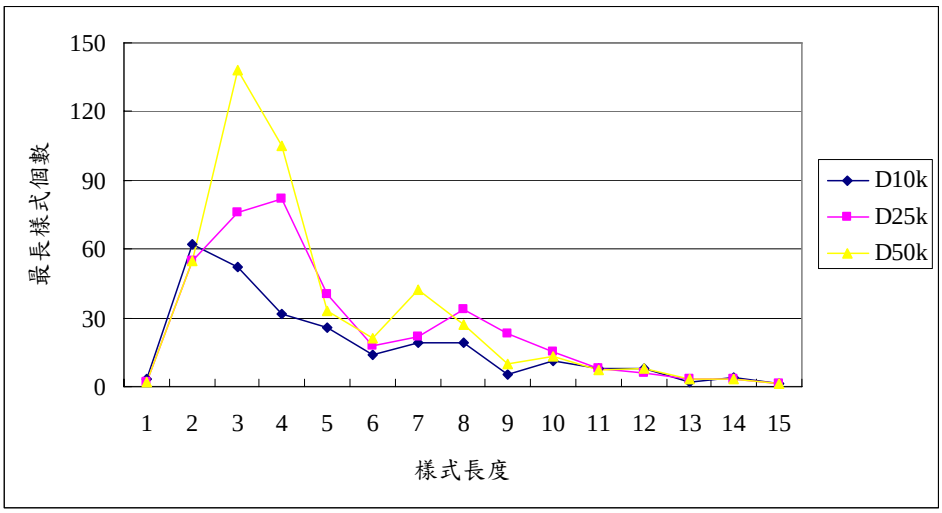

圖 13. PMSPM 各長度的最長樣式個數在支持度為 0.15%下的分佈

第三部分，在此預期增加最長樣式種類數，將會造成執行時間增加的結果。圖 14 中顯示當最長樣式種類數增加時，PMSPM 的執行時間會相對的提高，原因是種類增多的同時，顧客交易序列符合門檻值的機率也相對降低，造成最長頻繁樣式的平均長度減少，因而需要耗費較多的更新成本；反之當最長樣式種類數較少時，交易序列為頻繁樣式的機率會大大的提高，同時資料庫中會存在著長度較長的頻繁樣式，如圖 15、16、17 所示，在不同最長樣式種類數下，當最小支持度相同時，S50 通常會比 S100 及 S150 有著較多、較長的最長頻繁樣式；因而本方法在由上而下的階段中，候選最長序列不需做太多次的更新便可得到最長頻繁樣式，所以能比 Apriori-based 方法優先找出所有最長頻繁樣式以結束探勘，但若當 S 太大時亦會導致運算時間減少，其原因是頻繁的樣式過少，而樣式長度也相對減少，導致在探勘初期時於由下而上方向成長就可完成探勘，但此種情形較為極端，故於此不加以實驗討論。

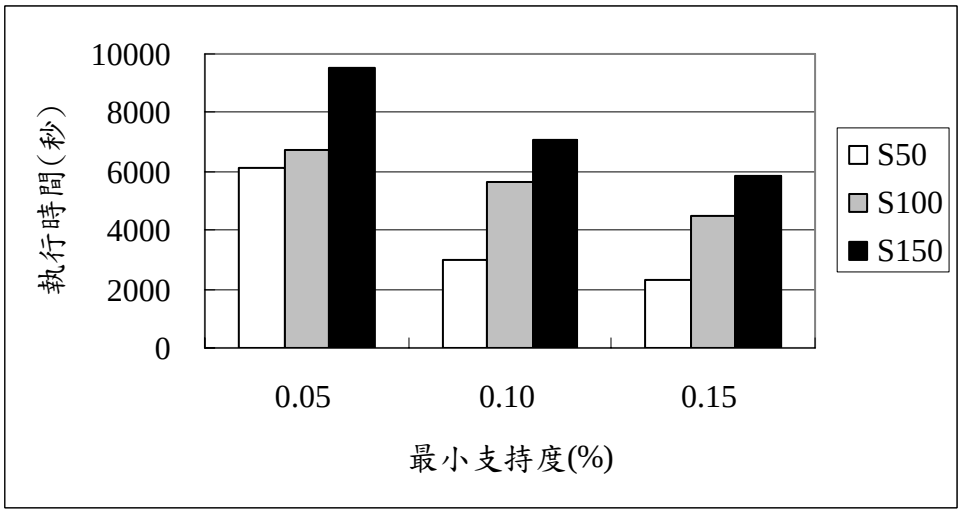

圖 14. PMSPM 最長樣式種類數和執行時間的比較

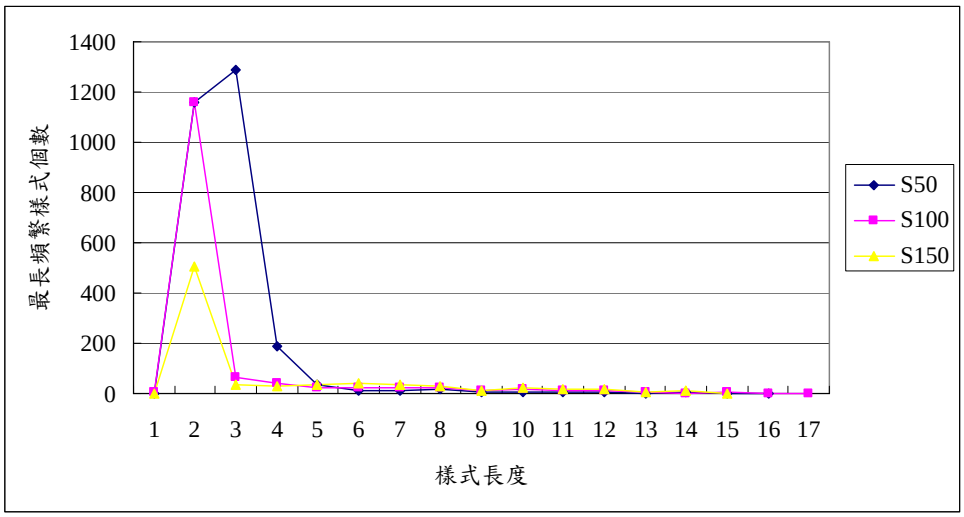

圖 15. PMSPM 各長度的最長樣式個數在支持度為 0.05%下的分佈

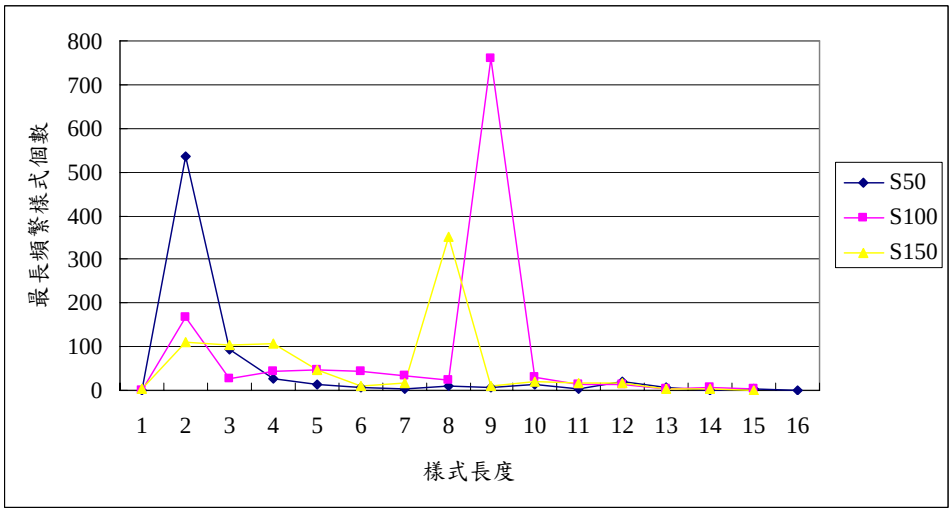

圖 16. PMSPM 各長度的最長樣式個數在支持度為 0.10%下的分佈

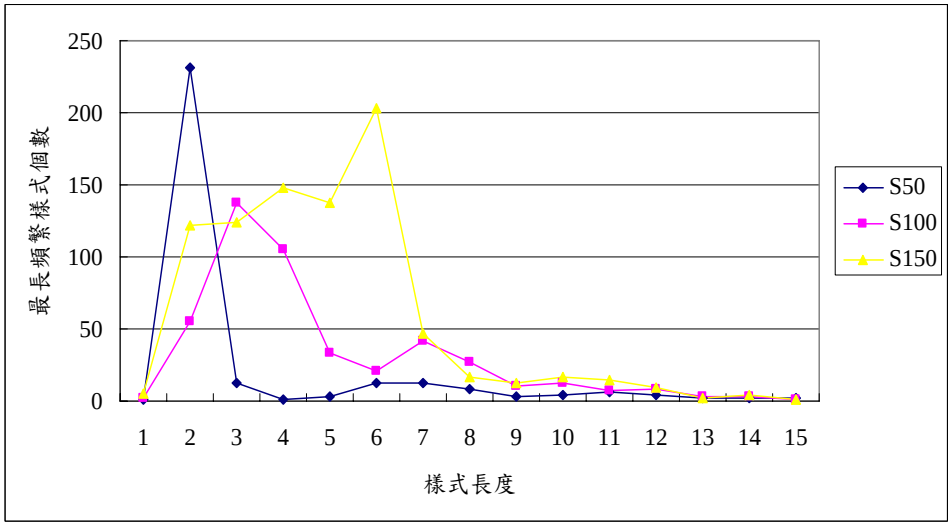

圖 17. PMSPM 各長度的最長樣式個數在支持度為 0.15%下的分佈

第四部分，於圖 18 可發現當項目個數較少的時候，所耗費的執行時間會比較少，原因在

於受到最長樣式種類數參數的影響，因為當項目個數減少時，模擬資料庫產生的交易序列會以較少的項目個數來組成樣式，其頻繁的可能性也隨之增高，因此使得在由上而下方向的處理不需太多更新的次數即可完成探勘。

另外，由圖 19、20、21 可發現當項目個數越大時，最長頻繁樣式的最長長度也有著隨之增加的趨勢，對於此現象的解釋，在觀察模擬資料庫後發現，當項目個數參數設定較大時，所產生的模擬資料庫中交易序列平均長度同時也隨著增加，因而造成有長度較長的最長頻繁樣式產生。

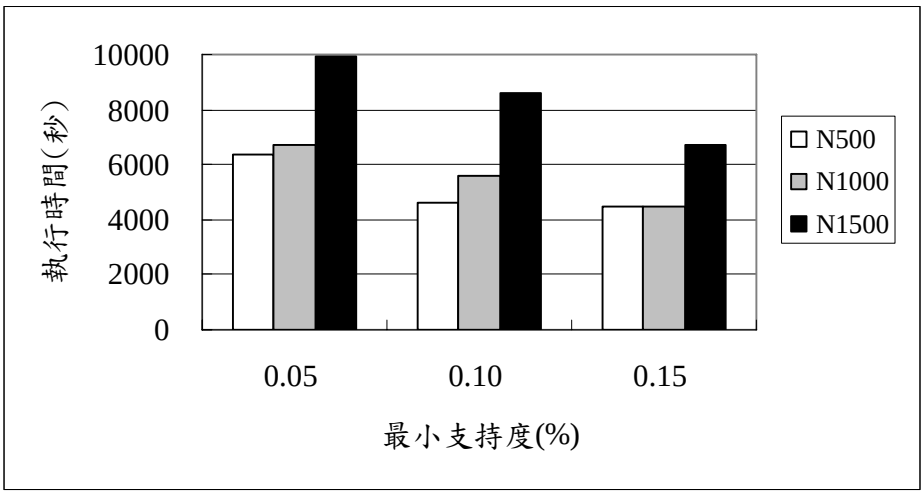

圖 18. PMSPM 項目數量和執行時間的比較

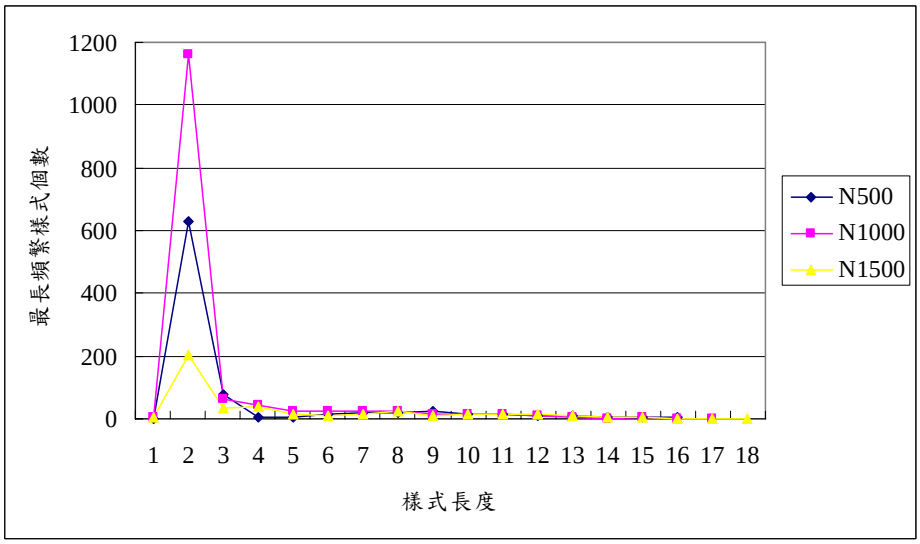

圖 19. PMSPM 各長度的最長樣式個數在支持度為 0.05%下的分佈

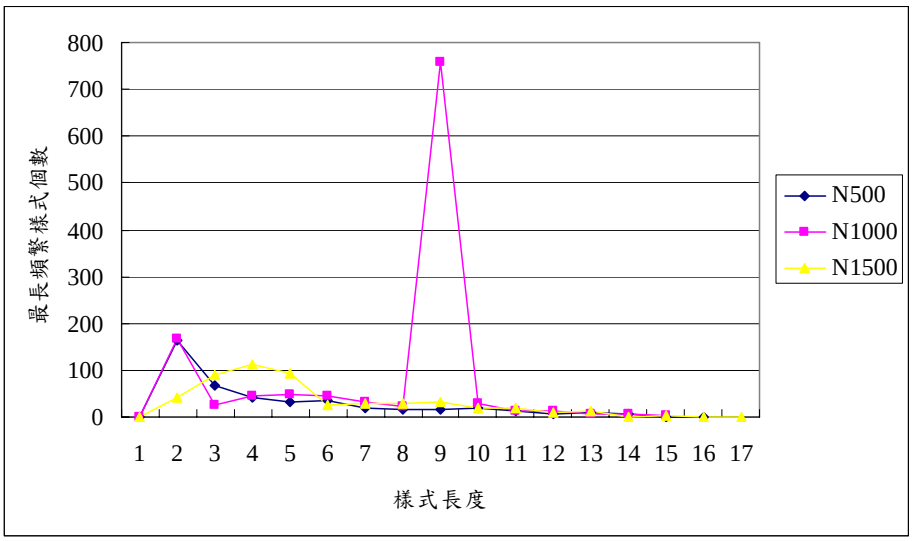

圖 20. PMSPM 各長度的最長樣式個數在支持度為 0.10%下的分佈

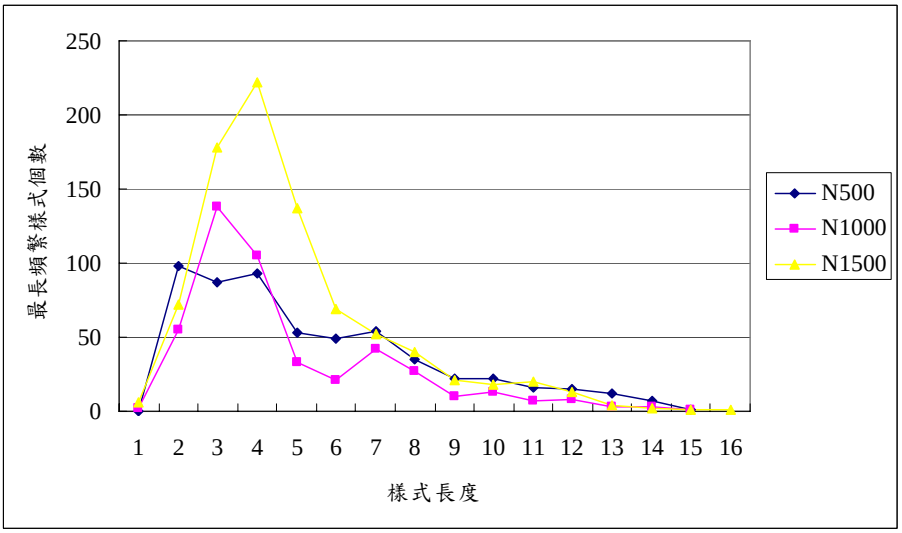

圖 21. PMSPM 各長度的最長樣式個數在支持度為 0.15%下的分佈

在上述四個部分的實驗結果中，可發現在不同支持度下的最長頻繁樣式長度的分佈情形通常會有下述的趨勢：

當最小支持度為 0.05%時，由於門檻值較低，同時造成支持度較低的低階樣式也較容易變得符合門檻值，然而當這些原本支持度較低的樣式繼續成長時，便無法再符合門檻值，因而造成在長度較短的最長頻繁樣式數量變得非常的多。

當最小支持度為 0.1%時，在較低階的最長頻繁樣式數量比支持度為 0.05%時來少得多，反而主要分佈在長度較長的部分，尤其是在交易序列平均交易數此參數值左右。

當最小支持度為 0.15%時，由於門檻值較高，因此造成多數的最長頻繁樣式比最小支持度為 0.1%時主要所分佈的長度較低，且最長頻繁樣式的數量跟長度也都會是最低的。

5.3 實驗結論

在第一部分的實驗中，可得知 PMSPM 演算法在交易序列的平均交易數量較長時，其效能

確實比 Apriori-based 演算法要來的好。

而在第三部分及第四部分的實驗中，我們也發現當最長樣式種類數較多，或是項目個數增加時，會使得所產生的模擬資料庫所產生的序列較混雜不容易產生較長頻繁樣式，因此便需要較多由上而下的更新成本來更新候選最長序列集合，使得執行的效能變差，同時其產生的最長頻繁樣式的長度也會相對減少。

因此我們可以得知 PMSPM 演算法較不適用於當序列過於混雜時，造成最長樣式長度較短的資料庫；反之，PMSPM 演算法則非常適合處理資料庫中存在著較長的最長樣式且序列較不混雜時的情形。

6. PMSPMTC 系統實作

6.1 實驗環境

本節敘述我們進行的 PMSPMTC 演算法實驗，在不同資料庫參數下，本演算法的執行時間長短，並與 PMSPM 演算法做比較。實驗機器是使用 Intel Core 2 Quad Q9550 處理器，2048MB DDRII 800 主記憶體，實驗環境是在 Windows XP 作業系統下進行，使用 Java (JDK 1.6.0) 程式語言開發。交易資料庫產生方式與 第 5.1 節相同，而時間資料則假設各交易之相鄰項目之間隔時間符合 卜瓦松分佈 (Poisson Distribution)，我們假設單位時間平均出現次數（λ）為 3。

6.2 實驗結果與討論

第一部分實驗在相同交易序列的平均交易數量(T)與三組不同支持度的情形下，以 10000 筆交易序列來比較 PMSPM 與加上時間限制的 PMSPMTC 演算法以顯示其 CMFS 個數和探勘總時間之差異，模擬資料庫參數如表 6 和表 7 所示。實驗結果可以由圖 22 發現以 10000 筆交易序列在 PMSPM 以及 PMSPMTC 演算法中的探勘時間其實是相近的，並未因為加上時間條件而有太大的差距。

在表 6 和表 7 中，我們發現以 PMSPMTC 演算法實驗所得到的 CMFS 個數比以 PMSPM 演算法得的 CMFS 個數多出幾倍數量。一般直覺會認為在加入時間限制條件後，條件較多，CMFS 個數應該會比原本未加入時間條件時來得少。我們在詳細檢視其探勘過程後，發現在 PMSPM 演算法中，因為沒有時間條件的限制，因此能產生較長的頻繁樣式，在每一階的過程中，藉由更新 CMFS 的動作，會刪除掉很多 CMFS 中的子序列，當最長頻繁樣式長度越大，就會刪除越多的子序列。相對來說 PMSPMTC 演算法，由於有時間條件的限制所獲得的最長頻繁樣式會比 PMSPM 演算法中來的短，因此刪除的子序列較少，所以留下的 CMFS 個數也較多，但這些子序列在有時間限制的 PMSPMTC 演算法中是相對重要的樣式，因此 PMSPMTC 保留了更多子序列使得探勘出來的資訊更顯得有意義且有彈性。

表 6. PMSPMTC 實驗之第一部份參數

資料集名稱	CMFS 個數	探勘總時間(分)
D10I1T10S10_ms5	4457	68.70722
D10I1T10S100_ms10	960	26.81907
D10I1T10S100_ms15	563	18.94326

表 7. PMSPM 實驗之第一部份參數

資料集名稱	CMFS 個數	探勘總時間(分)
D10I1T10S100_ms5	1823	70.14018
D10I1T10S100_ms10	419	23.30625
D10I1T10S100_ms15	266	15.08789

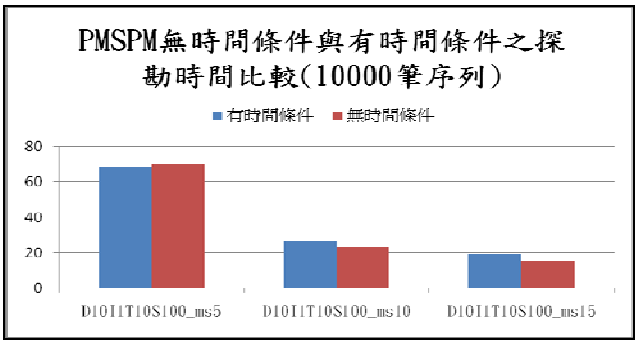

圖 22. PMSPM 與 PMSPMTC 之探勘時間比較(10000 筆序列)

第二部分實驗在交易序列的平均交易數量(T)不同的情形下，以 50000 筆交易序列和支持度為 25 來比較 PMSPM 與加上時間限制的 PMSPMTC 演算法以顯示其 CMFS 個數和探勘總時間，模擬資料庫參數如表 8 和表 9 所示，實驗結果可由圖 23 觀察到，以 50000 筆交易序列在 PMSPMTC 演算法的探勘時間較 PMSPM 演算法是大幅的增加，造成此種結果是因為在 PMSPMTC 演算法中的 CMS 個數所執行的時間比在 PMSPM 演算法中來得多。

以表 8 的 D50I1T7S100 資料集為例，由表 10 可以觀察出以 PMSPM 演算法計算出的 2 階非頻繁樣式個數為 35488，比較起 PMSPMTC 演算法中計算出的 15739 個 2 階非頻繁樣式多一倍以上，當非頻繁樣式數目較多，在更新 CMS 序列時，可以刪除掉較多不滿足支持度的樣式，CMS 個數相對比較少，所以造成在 PMSPMTC 演算法中 2 階 CMS 個數比 PMSPM 演算法還要多的結果，因此 PMSPMTC 的 CMS_updatetime 約 54 分鐘，比在 PMSPM 演算法中的 CMS_updatetime 多了 5 倍的時間，如表 11 所示。

表 8. PMSPMTC 實驗第二部份參數

資料集名稱	CMFS 個數	探勘總時間(分)
D50I1T7S100	611	56.7908
D50I1T8S100	1896	159.6413
D50I1T9S100	1037	149.5508
D50I1T10S100	2199	315.1942

表 9. PMSPM 實驗之第二部份參數

資料集名稱	CMFS 個數	探勘總時間(分)
D50I1T7S100_ms25	564	9.803577
D50I1T8S100_ms25	751	22.47079
D50I1T9S100_ms25	410	41.34904
D50I1T10S100_ms25	1428	101.7499

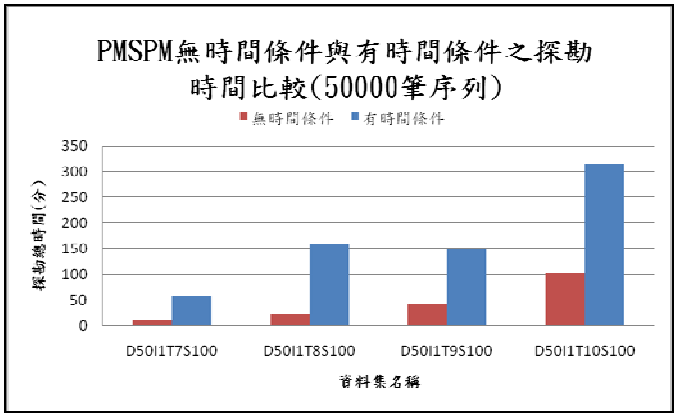

圖 23. PMSPM 與 PMSPMTC 之探勘時間比較(50000 筆序列)

表 10. PMSPM 與 PMSPMTC 之探勘結果比較

	PMSPMTC	PMSPM
1 階頻繁樣式個數	128	194
1 階非頻繁樣式個數	72	6
2 階頻繁樣式個數	645	2148
2 階非頻繁樣式個數	15739	35488
2 階 CMS 個數	21222	2725

表 11. PMSPM 與 PMSPMTC 演算法各步驟執行時間比較(單位:分)

	PMSPMTC	PMSPM
bottomup_time	0.131778101	0.129504
CMS_updatetime	53.88708941	9.141991
CMS_countingTime	0.176819807	0.087782
CMFS_delsubtime	0.000396189	0.000699
gen_subTime	6.29087E-06	0.000245

第三部份，實驗在交易序列的平均交易數量(T)不同但支持度相同的情形下，以 10000 筆、25000 筆以及 50000 筆交易序列在 PMSPMTC 演算法顯示其 CMFS 個數和探勘總時間，模擬資料庫參數 10000 筆交易序列如表 6 所示、模擬資料庫參數 25000 筆交易序列如表 12 所示、模擬資料庫參數 50000 筆交易序列如表 13 所示。比較結果顯示在圖 24，可以看出 PMSPMTC 演算法在相同的支持度下，以支持度為 0.15%和 0.1%下顯示隨著交易序列數量的增加，探勘時間只有些微的提升，但在支持度為 0.05%下顯示交易序列數量到達 50000 筆時探勘時間會大幅增加。

表 12. PMSPMTC 實驗之第三部份參數 (1)

資料集名稱	CMFS 個數	探勘總時間(分)
D25I1T10S100test2.txt_ms13.txt	3323	188.9238
D25I1T10S100test2.txt_ms25.txt	933	92.75805
D25I1T10S100test2.txt_ms38.txt	548	63.90104

表 13. PMSPMTC 實驗之第三部份參數 (2)

資料集名稱	CMFS 個數	探勘總時間(分)
D50I1T10S100test2.txt_ms25.txt	3707	578.0044
D50I1T10S100test2.txt_ms50.txt	1110	282.5549
D50I1T10S100test2.txt_ms75.txt	686	205.3713

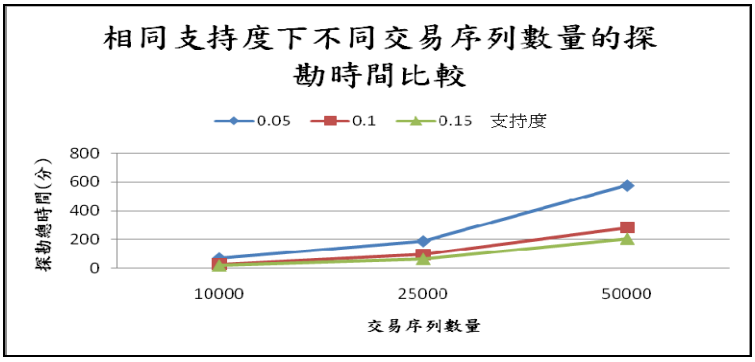

圖 24. PMSPMTC 演算法中相同支持度不同交易序列數量的探勘時間比較

7. 結論與未來發展

有鑒於以 Apriori 為基礎的方法在探勘頻繁樣式長度較長的資料時，會產生巨大的候選樣式產生與測試成本，導致若需探勘最長循序頻繁樣式時，需耗費大量的運算，因此本研究改良 Apriori 並結合 Pincer-Search[9]的概念，提出了一個探勘最長頻繁循序樣式的方法 PMSPM，運用如同夾子一般，以兩個方向夾擊的方式找出所有最長頻繁樣式進而早一步完成探勘。本方法的特點在於不需確切的檢查每一個樣式，以免不必要的運算。並且由前述的實驗中可以證明本方法當頻繁樣式平均長度較長時，其執行效能比 Apriori-based 來的好，也就是在由上而下階段中若能以較少的更新次數來得到較長的頻繁樣式，就可大量節省在由下而上中逐步探勘的時間。另外，我們也發現當項目數量較低時，所耗費執行的時間也比在其它條件相同下項目數量較多的情形來的少，因而可知本研究可應用於基因定序等等項目數較少，但是序列長度皆較長的實際情形下。

我們發現在由上而下方向探勘時，更新最大候選序列集合的運算，為最主要的成本之一，其在探勘初期所耗費的運算也是相當可觀的，所以未來研究可以分為下列幾項：

1. 進一步嘗試改進設計成更好的資料結構，降低運算成本，例如可嘗試將由上而下方向的探勘中所計算出的非最長頻繁樣式支持度所保留下來，在後續的探勘中若有相同的頻繁樣式則可不用計算其支持度或是可用於其它的方式來減少運算的成本。
2. 目前所處理的資料為單一項目的序列組成方式，將進一步改良成可處理多筆交易項目集，使得可應用的範圍更為廣泛。

由於 PMSPM 演算法所探勘的循序樣式中，項目間只有以前後順序來代表其時間關係，但在實務上，相鄰的兩個項目的時間間隔可能超過行銷的時效性，因此本研究提出一個 PMSPMTC 演算法，擴充 PMSPM 演算法，加入時間限制因子，使得探勘出的樣式更具意義性及解釋能力。由前述的實驗中也針對 PMSPM 演算法和 PMSPMTC 演算法對探勘時間做比較，且證實雖然交易序列數量越大在運算上耗費越多的時間，但 PMSPMTC 演算法卻能探勘出比 PMSPM 更有價值的 CMFS。這部份後續研究方向可分為下列幾項：

1. 進一步改進演算法的執行效率，使探勘時間減少。
2. 針對醫學上或商業上的資料集，使用 PMSPMTC 演算法進行實驗與測試，以對演算法做進一步的修正與改進。

參考文獻

[1] 許俊傑，MIHSPM：一個多項目集的混合循序樣式探勘演算法，*淡江大學資訊管理研究所碩士論文*，2007 年。

[2] 張耕，考量時間機率之循序樣式探勘方法，*淡江大學資訊管理研究所碩士論文*，2008 年。

[3] 顏志祐，一個能發掘更具意義循序樣式的探勘流程，*淡江大學資訊管理研究所碩士論文*，2008 年。

[4] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules", *In proceeding of the 20th international conference on VLDB*, Santiago, 1994, 487-499.

[5] R. Agrawal and R. Srikant, "Mining Sequential Patterns", *In proceedings of the 17th international conference on Data Engineering*, Taipei, Taiwan, 1995, 3-14.

[6] R. Agrawal and R. Srikant, "Mining Sequential Patterns: Generalizations and Performance Improvements", *Proc. of the Fifth International Conference on Extending Database Technology*, 1995.

[7] Y. L. Chen, S. S. Chen, Y. H. Hu, "Mining hybrid sequential patterns and sequential rules", Information Systems, Vol. 27, No. 5, 345-362.

[8] Y. L. Chen, Y. H. Hu, "Constraint-based sequential pattern mining: The consideration of recency and compactness", *Decision Support Systems*, Vol. 42, No. 2, pp. 1203-1215, 2006.

[9] D. Lin and Zvi M. Kedem, "Pincer-Search: An Efficient Algorithm for Discovering the Maximum Frequent Set", *IEEE Transactions on Knowledge and Data Engineering*, MAY 2002, Vol. 14 Issue 3, pp553-566.

[10] J. Pei, J. Han, and W. Wang, "Mining sequential patterns with constraints in large databases," *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, 2002, 18-25.

[11] J. Pei, J. Han and Y. Yin, "Mining frequent patterns without candidate generation", *In proceeding of the 2000 ACM SIGMOD international conference on Management of Data*, New York: ACM Press, 2000, 1-12.

[12] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M.C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix Projected Pattern Growth", *Proceedings of the 17th International Conference on Data Engineering*, Heidelberg, Germany, 2001, pp. 106-115.

[13] X. Yan, J. Han and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Datasets", *In proceeding of 2003 SDM conference*, San Franciso, CA, May 2003.

[14] M.J. Zaki, "Sequence Mining in Categorical Domains: Incorporating Constraints," *Proceedings of the 9th International Conference on Information and Knowledge Management*, pp. 422-429, Nov. 2000.

計畫成果自評：

自評項目	自評結果
研究內容與原計畫相符程度	100%
達成預期目標情況	100%
研究成果之學術或應用價值	1. 學術價值：開創循序樣式探勘新議題及做法 2. 應用價值：本研究可應用於基因定序等等項目數較少，但是序列長度皆較長的實際情形。
是否適合在學術期刊發表或申請專利	是
主要發現或其他有關價值	針對一般公認較不易處理之頻繁樣式平均長度較長情形，提出執行效能比較好的演算法。

RDT08

行政院國家科學委員會補助國內專家學者出席國際學術會議報告

2009 年 5 月 27 日

報告人姓名	周清江	服務機關及職稱	資訊管理系 副教授
會議時間 會議地點	2009年 5月6日至10日 義大利米蘭	本會核定補助文號	計畫編號： NSC 97-2221-E-032 -046
會議名稱	（中文） 第 11 屆企業資訊系統國際研討會 （英文）The 11th International Conference on Enterprise Information Systems		
發表論文題目	（中文） 一個以投影為基礎之混合式循序樣式探勘演算法 （英文）A Projection-Based Hybrid Sequential Patterns Mining Algorithm		

報告內容應包括下列各項：

一、參加會議經過

這次會議是在義大利米蘭舉行，全球五大洲共 70 餘國學者專家投稿，共發表 252 篇論文。此會議涵蓋 資料庫與資訊系統整合、人工智慧與決策支援系統、資訊系統分析與規格擬定、軟體代理人與網路運算、人機介面 等五個領域，並同時舉辦 13 個研究講習會(workshop)以及 1 個博士生討論會(consortium)，議程委員的國籍涵蓋 45 個國家或地區，規模隆重盛大，是每年舉行的資訊管理學界及產業界交流盛會。個人最大的收獲是參與座談討論會，了解服務運算、語意網路、調適性網路服務設計的最新研究方向。美國 Yahoo 實驗室 Mao 博士 “機器學習在線上廣告的應用” 演講對我也相當具有啟發性。此外與波蘭 Hryniewicz 教授、日本工業大學大木幹雄教授建立初步友誼。

二、與會心得

本人此次擔任人工智慧與決策支援系統之議程主席(session chair)，更了解國際會議有關論文評比的運作。

三、考察參觀活動(無是項活動者省略)

無

四、建議

建議增加補助金額，並多鼓勵老師擔任議程主席。

五、攜回資料名稱及內容

論文摘要集、論文光碟、論文發表證明及議程主席證明。

六、其他