時序性常見資料項集合之有效率探勘方法

房治政 柯佳伶

台灣師範大學 資訊教育系

jlkoh@ice.ntnu.edu.tw

摘要

在此論文中，我們針對如何有效探勘時序性常見資料項集合提出探討。本論文將探勘時序性常見資料項集合過程分成兩個階段：先探勘單一時間點的最大常見資料項集合，再以這些資料項集合為資料單位，探勘出時序性最大常見資料項集合。我們設計出資料項出現計數矩陣，根據計數矩陣中的數值，可加速可能的常見資料項集合之組合過程，並利用資料項索引序列快速計算出各資料項集合的出現次數，這些資料結構並擴展以運用在時序性常見資料項集合之探勘。此外，我們以各最大常見資料項集合的間隔區間中的最大值當作探勘區間大小，可確保找出精簡且完整的時序性常見資料項集合。實驗結果顯示我們所提出之演算法比相關文獻中所提出的方法，在執行時間上更有效率。

關鍵字: 常見時序性資料項集合, 資料探勘

一、緒論

資料探勘是近來相當熱門的研究主題，其主要的目的是從大量資料中尋找有價值的資訊或是知識[3]。以往資料探勘所考慮的資料來源，多是關聯式資料或是交易資料，鮮少對時序性資料庫進行探勘。但在現實生活中有許多資料具有時間屬性，例如台灣歷年進出口金額，或是歷年來國民平均所得等資料。這些時序性資料可經過轉換，表示成記錄某個時間點所發生的事件，例如今年比前一年出口金額增加或國民所得增加等事件。若能探勘出這些事件資料項在時序性資料庫中經常出現的時序性關聯規則，例如當今年出口金額增加，則次年國民所得會增加，則可做為參考知識或用來預測事件。

關聯規則探勘就是要找出不同資料項會伴隨出現的規則性，以往有許多研究以超級市場的交易資料為考慮，探勘單一次交易中購買資料項的關聯性，探勘關聯規則的第一步是要先找出常一起購買的資料項集合，稱為常見資料項集合。論文[1]提出 Apriori 方法來探勘常見資料項集合，先由各單一資料項個別形成一個候選集合，從這些候選集合找出常見資料項集合，再從包含一個資料項的常見資料項集合組合成包含兩個資料項的候選集合，並驗證其是否為常見資料項集合，每次遞迴重複以上類似步驟，以組合出包含多一個資料項集合，直到找不出更大的常出現資料項集合為止。但在探勘時序性常見資料項集合時，集合中不同時間點的資料項必須考慮其時序性及時間間隔，因此其組合情況較非時序性資料複雜許多。

在時序性資料的探勘研究方向中，有一些研究是針對時序性資料中各個資料項是否具有"週期性"進行探討，論文[5]與論文[6]所提出的探勘技術，其目的就是要從一個固定的週期之中找到資料項特有的變化情形。例如在每個星期三上午 9 點王先生固定會閱讀民生報，其中的每個星期就是一個"週期"，而王先生只有在星期三的上午 9 點會閱讀民生報，其他時間則沒有相同的行為，則這個現象便是一個探勘出的結果。論文[5]的作者利用一個"立方體"作為記錄用的資料結構，立方體的長、寬、高分別表示時序性資料庫的週期、時間點與及資料項。再利用這個立方體找出常見資料項集合。而論文[6]的作者則用"樹狀結構"來記錄各個週期發生的資料項，從中找出滿足使用者要求的常見資料項集合。在[5]與[6]這兩篇論文中皆是針對一個固定長度的的週期進行探勘，找出各週期的常見資料項集合，並不適用於週期長度不確定的情況。

論文[9]中將時序性資料庫中的每一筆記錄加入兩個時間屬性，分別儲存了該筆記錄的開始與結束時間，在探勘時序關聯規則時先定一個探勘的時間範圍，利用兩個時間相關屬性判斷有哪些記錄是在這個時間範圍之內，再對這些記錄找出常見因素集合，進而找出關聯性規則。

論文[4]、[7]與[8]對於以往在交易資料庫中只針對單筆交易間(transaction)進行關聯規則探勘的技術稱為"交易內的關聯規則探勘" (intra-transaction association rules)，而對多筆交易間進行探勘的技術稱為"交易間的關聯規則探勘"。將時序性資料庫中特定時間點發生的資料項視為交易資料庫中的一筆交易，則"時序性關聯規則"可視為一種"交易間的關聯規則"。

在論文[2]與[4]中提出的方法可以針對

不同的時間週期長度進行時序性關聯規則的探勘，由使用者先指定自己感興趣的規則形式，例如指定關聯規則中的前項或後項資料，或是對前後項資料間的時間間隔也加以指定。該系統的目的為快速找出滿足指定條件樣式的時序性關聯規則，而非直接由原始資料中找出所有的時序性關聯規則。

論文[7]則是從多維度資料中找出交易間的關聯規則，首先由使用者定出一個要進行探勘的視窗大小，將每個探勘視窗範圍內相同資料項卻屬於不同交易記錄者視為一個探勘視窗內的各別資料項，再利用 Apriori 類似的方法找出這些探勘視窗中的常見資料項集合，最後由常見資料項集合找出交易間的關聯規則。論文[8]的做法也是由使用者指定探勘視窗大小，但其做法是先以 Apriori 的方法找出同一交易內的常見資料項集合，再以常見資料項集合為資料項單位，針對探勘視窗範圍內所出現的常見資料項集合，以 Apriori 的方法找出跨交易間的常見資料項集合。

從以上相關文獻的討論可以發現，以往有關時序性關聯規則的探勘方法，大多採用 Apriori 類似的方法做為探勘時序性常見資料項演算法的主體，並且必須指定一個固定的探勘視窗大小，其缺點是必須要對原始資料進行多次的掃描，而視窗大小決定得是否適當也會影響探勘結果的完整性及執行效率。

本論文中將資料庫中每個資料項皆建立相對應的位元索引序列表示，並將探勘時序性常見資料項集合過程分成兩個階段：先探勘單一時間點的最大常見資料項集合，再以這些資料項集合為資料單位，探勘出時序性最大常見資料項集合。我們設計出資料項出現計數矩陣，根據計數矩陣中的數值，可加速可能的常見資料項集合之組合過程，並利用資料項索引序列快速計算出各資料項集合的出現次數，這些資料結構並擴展以運用在時序性常見資料項集合之探勘。此外，我們以各最大常見資料項集合的間隔區間中的最大值當作探勘區間大小，可確保找出精簡且完整的時序性常見資料項集合。實驗結果顯示我們所提出之演算法比相關文獻中所提出的方法，在執行時間上更有效率。

本論文以下內容簡述如下：第二節將對所研究的問題作詳細的說明並定義相關名詞，在第三節提出探勘同一時間點最大資料項集合的演算法，時序性最大資料項集合的探勘演算法將在第四節介紹，第五節將以實驗結果顯示所提出的演算法和相關演算法的執行效率比較，最後在第六節總結本論文內容，並討論未來的研究方向。

二、問題說明及相關定義

(一) 問題說明

我們以時序性資料庫 TD1 作為說明的範例，在 TD1 中共記錄了 10 個營業日中四種事件資料項發生的情況，四種資料項(A，B，C，D)分別表示台灣股市收盤指數上漲(A)、電子股收盤指數上漲(B)、金融股收盤指數上漲(C)及台幣兌換美元匯率走強(D)，TD1 之中各資料項在各時間點(每個營業日)的出現情形如圖 2.1 所示。

Temporal Database TD1

時間點 t	1	2	3	4	5
出現資料項	AB	CD	BCD	ABCD	C
時間點 t	6	7	8	9	10
出現資料項	BCD	ABCD	AB	CD	ABCD

圖 2.1

在此資料庫 TD1 中，我們發現當 A,B 事件同時發生後一個時間點後 C 事件會發生，再下一個時間點 B 和 C 事件也會接著發生，此具有時序性關係的資料項集合，在 TD1 共出現了兩次，因此很可能代表這些資料項之間有時序上的關連，此種時序性資料項集合便稱為時序性常見資料項集合。時序性常見資料項集合可用來分析資料項在時序性上的關連規則，進一步可用於事件預測，因此本論文的目的便是從記錄各時間點所發生資料項的時序性資料庫中，探勘出所有最精簡表示的時序性常見資料項集合。

(二) 相關名詞定義

給定一個時序性資料庫，是以固定的時間軸單位作為記錄依據，並依特定的時間點記錄在該時間點所發生的資料項。

[定義 2.1]一個時序性資料庫的 DB_T 由(T,X)所組成，X 表示資料庫中所有資料項所成的集合，T 表示時間軸及各個時間點所發生資料項的對應關係，時間軸上的特定時間點以 $t_1t_2\ldots t_n$ 表示，此外 t_i.items 表示在時間點 t_i 時所發生的資料項集合。

[定義 2.2]資料項的集合 Y 稱為時序性資料庫 $DB_T(T,X)$中***單一時間點的一個常見資料項集合***，若 $Y \subseteq X$ 且 Y.support $\geq$ min-support。

其中 $Y.support=\sum_{i=1}^{n} Appear(Y,i)$ ，

$$Appear(Y,i)=\begin{cases}1 & if \quad Y \subseteq t_i.items \\ 0 & otherwise\end{cases}$$

Y.support 代表 Y 在 $DB_T(T,X)$出現的總次數，min-support 稱為最小常見資料項集合出現次數，是由使用者決定常見資料項集合至少必須出現於資料庫中的次數，在以下探勘的過

程中，min-support 定為 2。此外，我們以 $FIS(DB_T)$表示 DB_T 中所有單一時間點的常見資料項集合所成的集合。

[定義 2.3] Y 稱為 $FIS(DB_T)$中一個***最大常見資料項集合***，若 Y 符合下列條件：

(1) $Y \in FIS(DB_T)$，且

(2) 不存在$Y' \in FIS(DB_T)$，使得 Y 為Y'的子集，且Y'.support=Y.support。

我們以 $MFIS(DB_T)$表示 DB_T 中所有最大常見資料項集合所成的集合。

[範例 2.1]在圖 2.1 範例中的時序性資料庫 TD1，可以發現同一時間點發生{A,B,C,D}在資料庫中共出現 3 次、{D}發生 9 次，{C,D}發生 9 次，這三種資料項集合都是單一時間點的常見資料項集合，其中{D}的出現次數與{C,D}的出現次數相同且{D}被{C,D}包含，因此{D}不是一個最大常見資料項集合，所以{A,B,C,D}與{C,D}為此資料庫中的最大常見資料項集合。

[定義 2.4]對於 $DB_T\{T,X\}$每個最大常見資料項集合 Y，令 $T_{appear}(Y)$代表 T 中有出現 Y 的各個時間點，若 Y 在 $DB_T\{T, X\}$的 support 為 s 則 $T_{appear}(Y)$表示為$\{t_{y1}t_{y2}\ldots t_{ys}\}$($t_{yi} \le t_{y(i+1)}$ for i=1,…,s-1)，Y 的***間隔區間***(以 SI 表示)定義為 $SI(Y)=\{\max(t)|t= t_{yi}-t_{y(i-1)}$ ，i=2,…,s}。

[範例 2.2]對於圖 2.1 範例中的時序性資料庫 TD1，可以找到最大的常見資料項集合 MFIS(TD1)={{C,D},{B},{B,C,D},{A,B,C,D}}，出現次數分別是 9、7、6 及 3 次。而各個資料項集合的間隔區間大小分別是 2、2、2 與 3 個時間點。其中{A,B,C,D}之間隔區間大小為 3，表示每次出現{A,B,C,D}後最多 3 個時間點時間點內會再發生{A,B,C,D}，代表{A,B,C,D}出現的最大週期。

[定義 2.5] 由時序性資料庫 $DB_T(T,X)$中 k 個不同的最大常見資料項集合($k \ge 2$)，可組合成一個***時序性資料項集合***

$Y_T=\{X_1(I_2)X_2(I_3)\ldots X_{k-1}(I_k)X_k\}$，

此時序性資料項集合的意義為:出現 X_1 後 I_2 個時間點後會出現 X_2，I_3 個時間點後會出現 $X_3,\ldots,I_k$ 個時間點後會出現 X_k，且其中 I_k 必須小於 X_1 的間隔區間。

[範例 2.3]對於圖 2.1 範例中的時序性資料庫 TD1，可以找到一個時序,性資料項集合{{C,D}(1){B,C,D}(2){C,D}}，其意義為當{C,D}發生一個時間點後，發生{B,C,D}，後兩個時間點發生{C,D}。

[定義 2.6]對於時序性資料庫 DB_T 中任一個時序性資料項集合 Y_T $Y_T=\{Y_1(I_2)Y_2(I_3)\ldots Y_{m-1}(I_m)Y_m\}$，與另一個時序性資料項集合 X_T，$X_T=\{X_1(J_2)X_2(J_3)\ldots X_{n-1}(J_n)X_n\}$，若 Y_T 與 X_T 符合下列條件，則稱 Y_T 為 X_T 的***時序性子集合***，

1. $m \le n$，且
2. $\exists\, 1 \le j < n$，$\forall\, 2 \le k \le m$ $I_k = J_{j+k-1}$，且 $\forall\, 1 \le k \le m$，Y_k 為 X_{j+k-1} 的子集。

[定義 2.7]將時序性資料庫 DB_T 中的一個時序性資料項集合 Y_T，$Y_T=\{X_1(I_2)X_2(I_3)\ldots X_{k-1}(I_k)X_k\}$為 DB_T 中的一個***時序性常見資料項集合***，若 Y_T 滿足 Y_T.support $\ge$ min-support。

其中 $Y_T.support = \sum_{i=1}^{n} Appear(Y_T, i)$，

$Appear(Y_T, i) =$

$$\begin{cases} 1 & if \;\; X_1 \subseteq t_i.items \;\; and \;\; for \;\; j=2,\ldots k \;\; X_j \subseteq (t_i + I_j).ite \\ 0 & otherwise \end{cases}$$

Y_T.support 代表 Y_T 在 $DB_T\{T,X\}$出現的總次數。

[定義 2.8] 我們以 $TFIS(DB_T)$表示時序性資料庫 DB_T 中所有時序性常見資料項集合所成的集合。時序性資料項集合Y_T稱為 DB_T 中一個***最大時序性常見因素集合***，若Y_T符合下列條件：

(1) $Y_T \in$ $TFIS(DB_T)$且

(2) 不存在$Y_T' \in$ $TFIS(DB_T)$，使得Y_T為Y_T'的一個時序性子集合且 Y_T .support= Y_T'.support。

[範例 2.2]對於圖 2.1 範例中的時序性資料庫 TD1，可以找到時序性常見資料項集合 TFIS(TD1)={{C}(1){B,D}(1){C}}，{{D}(1){C}}，因為後者為前者的時序性子集合，{{D}(1){C}}這個時序性常見資料項集合在 TD1 中不是一個最大時序性常見資料項集合。

(三) 主要探勘步驟

綜合上述討論，由於不同時間點的資料項組合情況非常複雜，因此本論文在探勘時序性常見資料項集合之前，先找出單一時間點中的常見資料項集合。採取這種做法的原因在於各資料項集合必須是單一時間點上的常見資料項集合，才有可能組合成跨越時間點的時序

性常見資料項集合。此外，亦能以更精簡的方式來表達時序性常見資料項集合。

在進行時序性常見資料項集合的探勘過程中，另一個要考慮的重點是:要定多長的時間範圍為探勘視窗，如果時間範圍定得太長，可能找出大於資料項出現週期的關連關係，不但增加探勘成本，亦非最精簡的探勘結果，而時間範圍定得太短，若資料項發生時序性關連的區間超出探勘視窗，則無法找出這些時序性資料項集合。為了解決這個問題，在本篇論文中對每個單一時間點上的常見資料項集合，找出其每兩次相鄰出現時間點間最大的時間間隔，稱為其間隔區間大小(請參考定義 2.4)，並以最大的間隔區間大小作為探勘視窗大小。如此將能確保找出所有最精簡的常見資料項集合，亦能避免探勘視窗過大而加重計算成本。

因此，本論文中時序性常見資料項集合的探勘過程分成下面幾個主要處理步驟：

[步驟 1].找出單一時間點中最大常見資料項集合：我們設計出資料項出現計數矩陣，根據計數矩陣中的數值，可加速可能的常見資料項集合之組合過程，並利用資料項索引串列快速計算出各資料項集合的出現次數。所找出的常見資料項集合再檢查其彼此間是否存在子集合關係，而留下最大常見資料項集合。

[步驟 2].決定探勘區間大小:對於步驟 1 找出的最大常見資料項集合，計算出其間隔區間大小，找出最大的間隔區間大小作為時序性常見資料項集合的探勘視窗大小。

[步驟 3].探勘跨時間點的最大時序性常見資料項集合:對各個最大常見資料項集合分別建立在探勘視窗大小內的資料項出現計數矩陣，以步驟 1 的方法擴展至驗證時序性常見資料項集合，主要不同是在資料項組合時必須考慮其相對時序性間隔，最後檢查是否存在時序性子集合關係，留下最大時序性常見資料項集合。

三、單一時間點常見資料項集合探勘

在這一節中，我們將介紹 MFIS 的演算法，用來探勘單一時間點上的常見資料項集合。在此演算法中運算過程中，我們以位元索引表(Bit Index Table)此種資料結構來儲存時序性資料庫，以減少探勘過程中儲存空間的使用，並可以位元運算的方式快速計算出資料項集合在資料庫中的出現次數，而不需要重複檢查原始資料。此外我們提出出現計數矩陣(Appearing Counting Matrix)，利用其中的資訊來過濾一些確定不是常見資料項集合的可能性，以減少候選常見資料項集合的個數，進而加速單一時間上常見資料項集合的探勘過程。

(一) 資料項位元索引表

資料項位元索引表是以二元位元表示法為基礎，對於時序性資料庫 $DB_T(T,X)$中的每個資料項 I 建立一個位元索引序列。位元索引序列的長度是時序性資料庫中時間軸的全部時間點個數，位元索引序列的低位元到高位元表示(位元編號由 1 開始)由時間軸的開始到結束的每個時間點該種資料項是否發生的情形。如果資料項 I 在時間點 t_j 的位置出現，則將 I 的位元索引序列在位元 j 設定為 1，否則設定為 0。在以下，我們以 I.bit-string 表示一個資料項 I 的位元索引序列。

延續前一節圖 2.1 的範例，在 TD1 中，共有 4 種資料項，分別是(A,B,C,D)，時序性資料庫共有 10 個時間點(1~10)，所以各個資料項的位元索引序列長度等於 10，A 在第 1、第 4、第 7 及第 10 個時間點出現，所以 A 的位元索引序列是(1.0.0.1.0.0.1.1.0.1)。TD1 的四個資料項的位元索引序列如圖 3.1 所示：

資料項	資料項位元索引序列
A	(1.0.0.1.0.0.1.1.0.1)
B	(1.0.1.1.0.1.1.1.0.1)
C	(0.1.1.1.1.1.1.0.1.1)
D	(0.1.1.1.0.1.1.0.1.1)

圖 3.1　TD1 的位元索引表

由上表可以看出，在時間點 1 及時間點 8 與時出現 A 與 B 兩個資料項，而時間點 2 及時間點 9 出現 C 與 D 兩個資料項。

利用建構好的位元索引表，我們可以很容易地計算出某種資料項集合在時序性資料庫中的出現次數，例如要計算 A 與 B 同時出現的次數，則將 A.bit-string 與 B.bit-string 進行 and 運算後，計算在所得到的位元序列中位元值為 1 的個數。當資料項集合的出現次數大於或等於 min-support，則判定為一個常見資料項集合。

資料項位元索引序列亦可用於時序性資料項集合的支持度計算上，例如要計算 A 之後一個時間點 B 出現的次數，則將 B.bit-string 左移一個位元再與 A.bit-string 進行 and 運算，運算的結果即為時序性資料項集合{A}(1){B}的位元索引序列，序列中位元為 1 的個數就表示{A}(1){B}.support，此方法將在下一章探勘時序性常見資料項集合時用到。

(二) 出現計數矩陣

當我們運用前一小節的方法進行驗證常見資料項集合探勘之前，必須先產生候選常見資料項集合。一種最常見的方法是循序地對資料庫中的所有資料項進行組合，再檢查組合產

生之資料項集合的出現次數，但因組合情況很多而極為耗時。因此我們設計"出現計數矩陣"，用來記錄每兩個資料項會同時出現的次數，以減少候選資料項集合產生的數量。

出現計數矩陣是一個二維陣列，若時序性資料庫 $DB_T(T,X)$，X 中的資料項個數為 n，則二維陣列的大小為 n× n，將 X 中的資料項依某選定的方法排序，依此順序對應到陣列中單一維度的索引位置。在出現計數矩陣的位置[I][J]記錄了在資料庫中當資料項 I 發生時，資料項 J 在同一時間點的發生次數。出現計數矩陣可與上一節的位元索引表同時建立，因此在探勘常見資料項集合過程中，只需對時序性資料庫進行一次掃描即可建立這兩個資料結構。

範例:時序性資料庫 TD1 的出現計數矩陣

	A	B	C	D
A	5	5	3	3
B		7	5	5
C			8	7
D				7

圖 3.2 出現計數矩陣範例(ACM)

在出現計數矩陣中，ACM 第 I 列顯示其他的資料項 J 與 I 一起出現的次數，因此 ACM[I][J]一定小於或等於 ACM[I][I]，如果 ACM[I][J]等於 ACM[I][I]，代表當資料項 I 出現時，資料項 J 皆同時出現，且(I,J).bit-string 即等於 I.bit-string。

(三)常見資料項集合探勘

我們運用所提出的位元索引表與出現計數矩陣來探勘單一時間點上的常見資料項集合。由於常見資料項集合之間會存在著一些資訊重複的現象，因此我們探勘的結果只留下最大的常見資料項集合。演算法主要分成探勘常見資料項集合與檢驗最大常見資料項集合兩部分。

探勘常見資料項集合的過程是對 *X* 中每一個資料項 I(I=1,..,n)，依據出現計數矩陣中的第 I 列中的資訊，判斷 I 是否為一個常見資料項集合。若 I 的出現次數大於 min-support，則找出包含資料項 I 且元素個數為 2 的常見資料項集合(以 $L_2(I)$表示)。再從 $L_2(I)$組合成元素個數為 3 的候選集合(以 $C_3(I)$表示)，再從中檢查是否有集合的出現次數滿足 min-support 的限制，找出 $L_3(I)$。重複相同方式，從 $L_3(I)$中的資料項集合組合出 $C_4(I)$，再找出 $L_4(I)$，直到找不出更大且包含 I 的常見資料項集合為止。在進行常見資料項集合探勘時，集合中並不需考慮資料項彼此間的順序差異，為避免產生重複結果，在探勘包含資料項 I 的常見資料項集合之過程，候選集合只需考慮比資料項 I 編號順序更大的資料項組合即可。針對每個資料項皆以上述過程找出包含 I 的常見資料項集合後，將結果聯集起來即為時序性資料庫中全部的常見資料項集合。

由於我們只要保留最大常見資料項集合(如定義[2.3])，必須檢查兩個常見資料項集合間的子集合關係。為減少此部份的計算成本，在探勘包含資料項 I 的常見資料項集合過程中，會先檢查其間之子集合關係，過濾掉一些確定不是最大常見資料項集合，留下的集合稱為局部最大常見資料項集合，最後再從中挑出最大常見資料項集合。

在探勘過程中，我們以位元索引序列來加速資料項集合出現次數的計算。此外，我們應用一個技巧來加速探勘過程：在對資料項 I 進行探勘 $L_n(I)(n\geq 2)$時，先檢查其他資料項 J 和 I 同時的出現次數(由 ACM[I][J]得知)是否與 I 出現次數相同(由 ACM[I][I]得知)相同。如果相同，表示有 I 出現時 J 皆會出現，若資料項集合{I,K}為一個常見資料項集合，{I,J,K}一定會是一個常見資料項集合，且{I,K}確定不是一個最大常見資料項集合，因此在探勘 $L_n(I)(n\geq 2)$的過程中，資料項 J 就不列入組合候選集合的考慮，並將{I,J}另外存在 Freq(I)中。在找出 I 和其他資料項所組成的每個常見資料項集合，再聯集 Freq(I)，成為一個可能的最大常見資料項集合，並檢查是否可以加入局部最大常見資料項集合 L-FIS(I)。

我們以時序性資料庫 TD1 的出現計數矩陣與資料項位元索引序列說明探勘過程。從出現計數矩陣的 A 列中，A 的出現次數大於 min-support，因此首先找與 A.support 次數相同的資料項，發現 B 的出現次數與 A.support 相同，將{A,B}存入 Freq(A)中。而 A 列中，另兩個資料項 C,D 的出現次數雖不等於 A.support，但也大於 min-support，所以將{A,C}、{A,D}加入 temp 集合中，用作組合 $L_3(A)$的依據，並存入 A 的局部最大常見資料項集合中(L-FIS(A))，此時 A 列中已沒有其他資料項的出現次數大於 min-support。又因為 temp 中有{A,C}與{A,D}集合存在，將{A,C}與{A,D}組合而成{A,C,D}，出現次數可由 3 個資料項的位元索引序列進行 and 運算計算 1 的數目而得到，經過計算 1 的個數為 5 次，大於 min-support 且無法再組成更大的資料項集合，所以與 Freq(A)聯集，再與 A 的局部最大常見資料項集合進行比對，將 L-FIS(A)中原有的{A,C}與{A,D}刪除，將{A,B,C,D}加入 A 的局部最大常見資料項集合中。最後將 Freq(A)加入 L-FIS(A)中，此時 A 的局部最大常見資料項集合中有{A,B,C,D}與{A,B}出現次數分別是 3 次與 5 次。

再對 B 列進行相同的步驟，因為 B 的資料項編號等於 2，由 B 列的第 2 項開始進行組合。在 B 列中，沒有其他資料項的次數與 B 的出現次數相同，另外只有 C 與 D 的出現次數大於 min-support，所以{B,C}與{B,D}會成為包含 B 且個數為 2 的常見資料項集合 L_2(B)，並存入 temp 集合中，用作組合 L_3(B)的依據也將{B,C}與{B,D}存入 B 的局部最大常見資料項集合 L-FIS(B)中。將 temp 中的{B,C}與{B,D}加以組合成包含 B 且個數為 3 的常見資料項集合，結果為{B,C,D}，利用 B.bit-string 與 C.bit-string 與 D.bit-string 進行 and 運算，運算後的結果中 1 的數目為 5，大於 min-support，所以{B,C,D}也是一個常見資料項集合，而且無法再組成更大的資料項集合，將{B,C,D}與 B 進行聯集，仍然是{B,C,D}，再與 L-FIS(B)中所有元素進行比對，因{B,C}與{B,D}出現次數為 5 次且為{B,C,D}的子集合，所以將{B,D}與{B,D}於 B 的局部性最大常見資料項集合中刪除。以同樣方式處理 C 列，則可以得到 L-FIS(C)={{C},{C,D}}與 L-FIS(D)={{D}}，最後整合各個資料項的 L-FIS，並檢查其中資料項集合是否符合"最大"的定義，可以得到時序性資料庫 TD1 單一時間點上的的最大常見資料項集合為{{C},{B},{C,D},{A,B},{B,C,D}, {A,B,C,D}}，次數分別為 8、7、7、5、5 與 3 次。

探勘單一時間點上的最大常見資料項集合之演算法虛擬主式碼如圖 3.6 所示

程式 3.1 MFIS()

```
/*輸入:時序性資料庫的資料項位元索引表與出現
      計數矩陣 ACM
/*輸出:單一時間點的最大常見資料項集合 MFIS */
0: begin
1:     for each item I in X
2:     begin
4:       if ACM[I][I] ≥ min-support
5:           L-FIS(I):=FIS_generate(I)
         /*針對 I 資料項找常見資料項集合*/
6:       for each set LF_i in L-FIS(I)
7:           check (MFIS, LF_i)
8:     end
9:end
```

圖 3.3 單一時間點的最大常見資料項集合之演算法

而產生常見資料項集合的演算法 FIS_generate(I)的基本概念則與 Apriori 的方法類似，但因為在出現計數矩陣中只要出現次數大於 min-support 者，一定是長度為 2 的常見資料項集合，所以只需要由長度為 3 的常見資料項集合開始找起即可，而且每次呼叫 FIS_generate()所探勘的常見資料項集合都是包含資料項 I 的集合。程式虛擬碼如程式 3.4 所示。

程式 3.2 FIS_generate(I)

```
/*輸入：資料項 I*/
/*輸出：包含 I 且順序在 I 之後的資料項之最大常
見資料項集合*/
0: begin
1:   Freq(I):={I}
2:   for each data item I
3:       for each data item J＞I
4:           if ACM[I][J]= ACM[I][I]
5:              begin
6:                Freq(I):= Freq(I)∪{J}
7:                check(L-FIS(I), {I,J})
8:              end
9:           else if ACM[I][J] ≥ min-support
10:             begin
11:               temp:=temp∪{{I,J}}
12:               check(L-FIS(I), {I,J})
13:             end
14:  while temp !=ϕ
15:          begin
16:            C-FIS:=ϕ
17:            for each item set X in temp
18:                for each item set Y in temp
19:                   if X-Y contain one item
20:                     begin
21:                       M_i:=X∪Y,
22:                        C-FIS:=C-FIS∪{M_i}
23:                     end
24:            temp:=ϕ
25:            for each set M_i in C-FIS
26:              if M_i.bit-string.count ≥ min-supoport
27:                begin
28:                  temp:=temp∪{M_i}
29:                  check(L-FIS(I),Freq(I)∪{ M_i })
30:                end
31:    end
32:    return L-FIS(I)
33:end
```

圖 3.4 產生最大常見資料項集合之演算法

是否可加入最大常見資料項集合，以其合併局部最大為整體最大的最大常見資料項集合，皆是呼叫函式 check(TFIS, LF_i)，只是傳入的變數有所不同。欲加入之常見資料項集合中 LF_i 與 TFIS 中已有的資料項集合檢驗出現次數相同及是否有子集合關係，以決定 LF_i 可否加入 TFIS 中，TFIS 的存放方式是以出現次數由大至小排序，因此可先以次數比較結果決定是否需要比較子集合關係，並能決定是否需要繼續比較 MFIS 中的其他集合，以增進執行效率。

四、時序性常見資料項集合之探勘

這一節中將說明如何對時序性常見資料項集合進行探勘。首先我們對最大常見資料項

集合建立各自對應的位元索引序列，並擴展在第三節所介紹之出現計數矩陣，建立最大常見資料項集合的時序性出現計數矩陣，再擴展MFIS()演算法，以時序性出現計數矩陣加快候選集合選取的方法，並提出探勘時序性常見資料項集合的演算法。

(一) 最大常見資料項位元索引表

在經過第三章所述 MFIS()演算法的處理，我們可以找出時序性資料庫中最大常見資料項集合所成的集合(MFIS)。若 MFIS 中有 n 個資料項集合，我們以 f_i $(1 \le i \le n)$表示一個最大常見資料項集合，並稱為一個時序性資料項。對每個時序性資料項 f_i，將 f_i 中所包含的資料項之位元索引序列進行 and 運算，得到 f_i 的位元索引序列。因此，時序性資料項所形成的位元索引表即表示刪除掉非常見資料項集合後，時序性資料庫中的內容。

同樣以時序性資料庫 TD1 為範例加以說明，TD1 的時序性資料項 MFIS $=\{f_1,f_2,f_3,f_4,f_5,f_6\}$，其中 $f_1=\{C\}$，$f_2=\{B\}$，f_3 為 {C,D}，$f_4=\{A,B\}$，$f_5=\{B,C,D\}$，而 f_6 表示 {A,B,C,D}。時序性資料項位元索引表如圖 4.1 所示。

時序性資料項	索引序列
f_1{C}	(0.1.1.1.1.1.1.0.1.1)
f_2{B}	(1.0.1.1.0.1.1.1.0.1)
f_3{C,D}	(0.1.1.1.0.1.1.0.1.1)
f_4{A,B}	(1.0.0.1.0.0.1.1.0.1)
f_5{B,C,D}	(0.0.1.1.0.1.1.0.0.1)
f_6{A,B,C,D}	(0.0.0.1.0.0.1.0.0.1)

圖 4.1 時序性資料項之位元索引表

此外，透過時序性資料項之位元索引表，我們可以利用位元索引序列運算快速計算出一個時序性資料項的出現次數，例如計算 f_1 後一個時間點後出現 f_2 的次數，即將 f_2 的位元索引序列左移一個位元之後，與 f_1 的位元索引序列進行 and 運算，再計算結果中位元值為 1 的個數，所以$\{f_1(1)f_2\}$的出現次數為 6。

在各個時序性資料項(f_i)的位元索引序列中，計算兩個相鄰近位元值為 1 的最大間隔，即是 f_i 的間隔區間，對於範例的時序性資料庫 TD1，MFIS 中 f_1,f_2,f_3,f_4,f_5,f_6 間隔區間分別是 2、2、2、3、3 與 3 個時間點。在此論文中，我們將探勘視窗大小(SI)定為時序性資料項間隔區間中的最大值，該值為可確保當任一個時序性資料項 f_i 出現後，至少再出現一次 f_i 的最小時間範圍。

(二) 時序性出現計數矩陣

為了快速過濾時序性常見資料項集合的後選集合，我們擴展出現計數矩陣的做法，建立時序性資料項的時序性出現計數矩陣。時序性出現計數矩陣與出現計數矩陣不同之處是其多了一個時間點維度，時間點維度大小為探勘視窗大小減 1。若時序性資料項個數為 m，則此矩陣大小為 m× m× (SI-1)。陣列[i][j][k]中記錄當 f_i 發生後 k 個時間點時發生 f_j 的次數，也就是$\{f_i$ (k) $f_j\}$.support。

時序性出現計數矩陣的建構方法如下：將 MFIS 中的各個元素 f_i 對 MFIS 中的其他每個元素 f_j，計算$\{f_i(k)f_j\}$.support (k=1,…,SI-1)，$\{f_i(k)f_j\}$.support 可以將 f_j.bit-string 左移 k 個位元，再和 f_i.bit-string 作 and 運算中位元值為 1 的個數求得。

時序性出現計數矩陣 TACM[1][*][*] 及 TACM[2][*][*]

	1	2
f_1{C}	6	5
f_2{B}	6	4
f_3{C,D}	5	4
f_4{A,B}	5	3
f_5{B,C,D}	4	3
f_6{A,B,C,D}	3	2

f_1.support=8

	1	2
f_1{C}	5	5
f_2{B}	3	4
f_3{C,D}	4	4
f_4{A,B}	3	2
f_5{B,C,D}	2	3
f_6{A,B,C,D}	2	1

f_2.support=7

時序性出現計數矩陣 TACM [3][*][*] 及 TACM [4][*][*]

	1	2
f_1{C}	5	4
f_2{B}	5	3
f_3{C,D}	4	3
f_4{A,B}	4	2
f_5{B,C,D}	4	2
f_6{A,B,C,D}	2	1

f_3.support=7

	1	2
f_1{C}	3	4
f_2{B}	1	3
f_3{C,D}	2	4
f_4{A,B}	1	1
f_5{B,C,D}	0	3
f_6{A,B,C,D}	0	1

f_4.support=5

時序性出現計數矩陣 TACM [5][*][*] 及 TACM [6][*][*]

	1	2
f_1{C}	3	3
f_2{B}	3	2
f_3{C,D}	2	2
f_4{A,B}	3	1
f_5{B,C,D}	2	1
f_6{A,B,C,D}	2	0

f_5.support=5

	1	2
f_1{C}	1	2
f_2{B}	1	1
f_3{C,D}	0	2
f_4{A,B}	1	0
f_5{B,C,D}	0	1
f_6{A,B,C,D}	0	0

f_6.support=3

圖 4.2

以時序性資料庫 TD1 為範例，說明其為時序性資料項所建立的時序性出現計數矩陣。其中 MFIS={f_1,f_2,f_3,f_4,f_5,f_6}，而 SI=3。因此建立一個 6× 6× 2 的陣列，為方便顯示，我們以六個二維陣列表示如圖 4.2。f_i 的出現次數則以 f_i.support 來表示。

(三) 時序性常見資料項集合之探勘

在探勘時序性常見資料項集合的過程中，不只要考慮如何由包含資料項較少的時序性常見資料項集合組合成包含資料項更多的時序性常見資料項集合，更要考慮各個資料項集合之間時間點先後的關係。同樣為避免資訊重複的現象，我們探勘的結果將只留下最大的時序性常見資料項集合，至於演算法將分成探勘時序性常見資料項集合與檢驗最大時序性常見資料項集合兩部分。

探勘時序性常見資料項集合的過程和前一章所述 MFIS 演算法類似，若有 n 個時序性資料項，則分別探勘以 f_i (i=1,..,n)為起點的最大時序性常見資料項集合。由 f_i 為起點，且由 m 個資料項集合組成的時序性常見資料項集合，我們以 $TL_m(f_i)$表示。從出現計數矩陣中[i][*][*]中，若[i][j][t]值大於 min-support，即表示[i][j][t]({$f_i(t)f_j$}) 為 $TL_2(f_i)$其中之一。由 $TL_m(f_i)$ (m≥2)再組合成包含 m+1 個資料項集合的候選時序性資料項集合(以 $TC_{m+1}(f_i)$表示)。如果候選集合的出現次數大於 min-support，則為 $TL_{m+1}(f_i)$之一。重複上述步驟直到找不出包含更多資料項集合的時序性資料項集合為止。

至於找出最大時序性常見資料項集合的做法，是先找出以 f_i 為起始資料項的局部最大時序性常見資料項集合，再由各局部最大時序性常見資料項集合中找出全體的最大時序性常見資料項集合

在產生候選時序性資料項集合時，必須要考慮到時序性資料項集合間時間點先後的關係，兩個集合必須要滿足下列條件才允許產生候選集合。

1.集合內的時序性資料項個數相同，且只有一個資料項不同。

2.集合中的時序性料項對起始資料項 f_i 的相對時間點，只能有一個時間點不同。

此外，為了避免重複產生候選的時序性資料項集合，在兩個常見時序性集合進行組合時，要求上述條件 2 中差異的時間點記錄必須比原集合中最後一個資料項的出現時間點更大，才允許進行組合。

為了加速組合兩個時序性常見資料項集合時的檢驗，我們提出以下的資料結構(TIS-structure)，用來儲存探勘過程找出的一個時序性資料項集合，如圖 4.3。

```
struct TIS
1：  {
2：  bitset<n× (SI)> items;
       /* n 為時序性資料庫 DB_T(T,X)X 中資料項目的個數，利用一個長度為 n 的位元序列記錄，此時序性資料項集合所包含的時序性資料項*/
3：  bitset<m×(SI-1)> freq-item-set ;
       /*依照集合中的時序性資料項對應在 TACM[i][*][*]中的位置*/
       /*依照行優先的順序表示，m 為時序性資料項的個數*/
4：  bitset<SI-1> order ;
       /*表示集合中的時序性資料項在起點資料項發生後(SI-1)範圍內的那些時間點發生
5：  bitset<time-no>  bitsequence;
       /*記錄該時序性資料項集合的位元索引序列*/
6：  int count;
       /*記錄該時序性資料項集合在資料庫的出現次數*/
7:   int level;
       /*記錄最後一個資料項和起始資料項的時間點間隔*/
8:     } ;
```

圖 4.3 TIS-structure

例如在探勘以 f_5 為起點的 TL_m(m≥2)，會檢查時序性出現矩陣中的[5][*][*]。在 TACM[5][*][*]中可以發現 TACM[5][1][1]= 3，表示當 f_5 組合在它出現後一個時間點的 f_1(表示{$f_5(1)f_1$})，其出現次數為 3 次，大於 min-support，所以是一個時序性常見資料項集合。TIS 結構中 items 的作用是要表示此時序性常見資料項集合中的資料項內容及其時序關係。因原本時序性資料庫中共有 4 種資料項，所以每一個時間點以一個長度為 4 的位元序列表示，依序表示各個資料項的出現情形，且探勘視窗大小為 3，所以會將 3 串長度為 4 的位元序列(分別表示開始時間點及之後的時間點 1 與時間點 2 之內容)合成一個長度為 12 的位元序列。TIS 結構中的 freq-item-set 則表示集合是由那些時序性資料項(f_i)組合而成，同樣以位元序列方法表示，並將對應到各時間點的序列依時間點順序合併成一個序列表示。因此{$f_5(1)f_1$}的 TIS 內容如圖 4.4 所示。

如果要對以 f_5 為開頭中的兩個時序性常見集合進行組合，則必須符合之前所列的兩種條件，例如在 TACM[5][*][*]中的[5][2][2]之值大於 min-support，因此{$f_5(2)f_2$}亦為一個時序性長間資料項集合，利用 TIS 此種資料結構儲存的結果如圖 4.5 所示。

各個欄位	儲存內容(位元序列、數值)	說明
{$f_5(1)f_1$}.items	(0.1.1.1,1.0.0.0,0.0.0.0)	第一組 4 個位元表示開始時間點出現 f_5={B,C,D}，第二組 4 個位元表示下一個時間點出現 f_1={A}，第三組 4 個位元表示沒有資料項包含在集合內
{$f_5(1)f_1$}.freq-item-set	(1.0.0.0.0.0,0.0.0.0.0.0)	第一組 6 個位元中，第 1 個位元為 1，表示 f_5 出現後第 1 個時間點有 f_1 包含在集合中
{$f_5(1)f_1$}.order	(1.0)	表示在起始資料項後，第 1 個時間點有資料項包含在集合內
{$f_5(1)f_1$}.bitsequence	(0.0.1.1.0.1.0.0.0.0)	利用 f_1 的位元序列左移一個位元後與的 f_5 的位元進行 and 運算
{$f_5(1)f_1$}.count	3	計算 {$f_5(1)f_1$}.bitsequence 中 1 的數目
{$f_5(1)f_1$}.level	1	說明{$f_5(1)f_1$}最後一個資料項與起始資料項相差的時間點

圖 4.4 {$f_5(1)f_1$}以 TIS 結構儲存之結果

各個欄位	儲存內容(位元序列、數值)	說明
{$f_5(2)f_2$}.item-set	(0.1.1.1,0.0.0.0,0.1.0.0)	第一組 4 個位元表示開始時間點出現 f_5={B,C,D}，第三組 4 個位元表示之後 2 個時間點出現資料項{B}
{$f_5(2)f_2$}.freq-item	(0.0.0.0.0.0,0.1.0.0.0.0)	第二組 6 個位元中，第 2 個位元為 1，表示第 2 個時間點出現 f_2
{$f_5(2)f_2$}.order	(0.1)	表示在第 2 個時間點有資料項包含在集合內
{$f_5(2)f_2$}.bitsequence	(0.0.1.0.0.1.0.0.0.0.)	
{$f_5(2)f_2$}.count	2	
{$f_5(2)f_2$}.level	2	

圖 4.5 {$f_5(1)f_2$}以 TIS 結構儲存之結果

如果要組合{$f_5(1)f_1$}與{$f_5(2)f_2$}，必須檢查兩個集合是否合乎組合的兩個條件，因為 $TL_m(f_i)$中的任兩個時序性常見資料項集合所包含的時序性資料項個數一定是 m 個，所以只要確定兩者所包含的時序性資料項是否只有一項不同即可。利用 TIS 結構中的 freq-item-set 此一欄位，進行 or 位元運算，運算的結果如果位元為 1 的個數較原本集合中的個數只增加 1，則表示兩個集合所包含的時序性資料項只有一項不同。

對於另外一個條件:確定兩個集合中各個資料項與 f_i 的相對時間點也只能有一個時間點不同，則可利用 TIS 結構中的 order 欄位，進行 or 位元運算，運算的結果中位元為 1 的個數若較原本集合中的個數只增加 1，表示兩個集合中所包含的時序性資料項的時間點只有一個不同。

此外，至於為了避免重複產生候選的時序性資料項集合，可以在選擇組合對象時，只挑選 level 更大者進行組合。

所以以{$f_5(1)f_1$}與{$f_5(2)f_2$}為例，可以發現，{$f_5(1)f_1$}.freq-item-set 與{$f_5(2)f_2$}.freq-item-set 進行 or 運算後的位元序列為(1.0.0.0.0.0.0.1.0.0.0.0)，位元值為 1 的個數比原本集合中 freq-item-set 欄位中 1 的個數多 1 個，而兩者的 order 欄位進行 or 運算後結果是(1.1)，也比原本 order 欄位中位元值為 1 的個數多 1 個，再加上{$f_5(2)f_2$}.level=2 比{$f_5(1)f_1$}.level 為大，所以{$f_5(1)f_1$}與{$f_5(2)f_2$} 可以進行組合，產生{$f_5(1)f_1(2)f_2$}。

探勘 $TL_m(f_i)$的過程，先針對 TACM[i][*][*]矩陣的每一行(時間點)進行檢查，只保留出現次數大於 min-support 者成為 $TL_2(f_i)$，在掃描的同時，對於出現次數相同的資料項集合，去掉為其他集合之子集合者，並不存入 $TL_2(f_i)$中。

例如在 TACM[5][*][*]的第一行，可以發現[5][1][1]、[5][2][1]與[5][4][1]的值皆為 3 次，表示{$f_5(1)f_1$}、{$f_5(1)f_2$}及{$f_5(1)f_4$}為常出現的時序性常見資料項集合，其中 f_2({B})為 f_4({B,C,D})的子集合，所以並不將[5][4][1]存入 $TL_2(f_5)$中，而[5][3][1]、[5][5][1]與[5][6][1]的次數皆為 2 次，但是 f_3 與 f_5 為 f_6 的子集合，所以只存{$f_5(1)f_6$}於 $TL_2(f_5)$中，但在 TACM[5][*][*]的第二行，雖然[5][2][2]與[5][3][2]的次數相同，但彼此間沒有子集合的

關係，所以{$f_5(2)f_1$}、{$f_5(2)f_2$}與{$f_5(2)f_3$}皆加入 $TL_2(f_5)$中。並將 $TL_2(f_5)$的每個資料項集合存入 f_5 的局部最大時序性常見資料項集合中。

接著在 $TL_2(f_5)$中每次取出兩個時序性常見資料項集合，如果符合組合的條件則進行組合，可以找到 9 種組合，但沒有一種組合的出現次數大於 min-supoport。最後再對 f_5 的局部最大時序性常見資料項集合進行檢查，以確定是否為全體的最大時序性常見資料項集合。

因論文篇幅所限，探勘最大時序性常見資料項集合的演算法 TMFIS 之程式虛擬碼請參考[10]。

五、效率評估

在這一節中，我們以實際實驗數據來評估所提出 MTFIS 演算法的執行效率，並與同樣探勘時序性常見資料項集合的 FITI 演算法[8]做比較。

實驗是以 Borland C++ Builder 4.0 實作演算法程式，並利用 C++所提供的 bitset class 作為 bit-string 實作的依據，以個人電腦作為實驗環境，該電腦配備 Pentium II 266 的中央處理器，搭配 128 MB 主記憶體，採用 Windows 2000 Professional 作業系統。

測試資料是在控制參數設定的情況之下由電腦隨機產生，其中控制的參數有資料項種類個數，各資料項出現的最大時間間隔，以及時間軸上的時間點數目。此外，對於每一個實驗皆以 20 組測試資料執行演算法，再以平均值作為實驗的數據。實驗一到實驗三，將考慮不同控制參數設定，分別針對我們的演算法所需要的執行時間與所需要的記憶體空間進行評估比較。

實驗一的目的是比較時間軸的時間點數目對 MTFIS 與 FITI 兩種演算法在執行時間上的影響，資料項種類設為 5 種、資料項出現的最大探勘區間則設為 4 個時間點，實驗結果如圖 5.1 所示。而圖 5.2 顯示將資料項出現的最大探勘區間設定為 5 個時間點的測試結果。

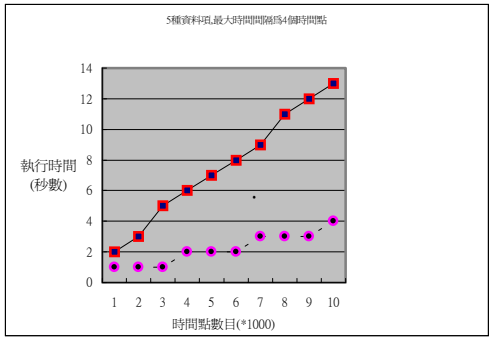

圖 5.1

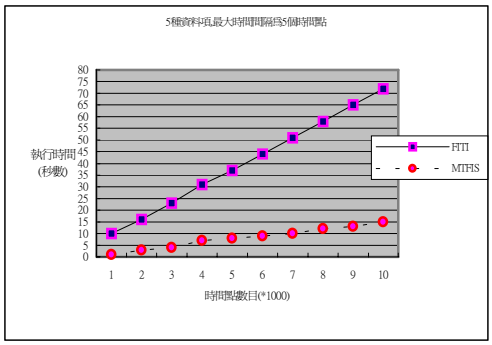

圖 5.2

由圖 5.1 及圖 5.2 可以看出，對相同參數設定條件之下產生的測試資料，MTFIS 演算法比[8]所提出的 FITI 有更佳的執行效率。且當時間點數目增加時，兩者的執行時間皆近似線性成長。

既然在相同的參數設定條件之下，TMFIS 比 FITI 有更佳的執行效率，我們希望能了解，那一種資料參數變因對執行效率的影響最大，所以接下來針對不同數目的資料種類進行實驗二，將時間點數目設定為 5000，最大時間間隔為 4 個時間點，以下圖 5.3 顯示實驗二的結果。

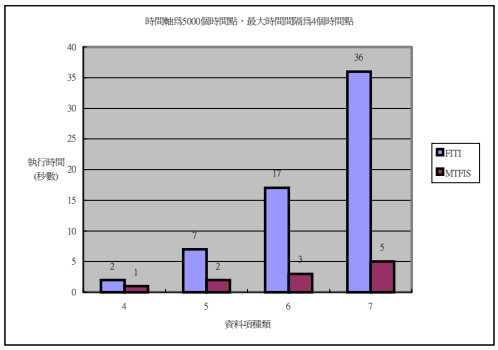

圖 5.3

實驗三則顯示資料出現的最大時間間隔對演算法執行時間的影響，(如圖 5.4)在此實驗，資料項種類設為 5 項，時間軸大小控制為 5000 個時間點。

由實驗二與實驗三的結果，我們可以發現兩種演算法的執行時間都會受到資料項種類的多寡與最大時間間隔大小的影響，但在這兩種資料參數中，又以最大時間間隔的大小對演算法執行效率的影響最為顯著。這是因為資料出現的最大時間間隔越長，探勘視窗大小也會越大，則探勘出的時序性常見資料項集合組合情況也將更為複雜。至於資料項種類的多寡，雖然也對執行效率產生一定的影響，但相較之下影響程度就並不那麼顯著。

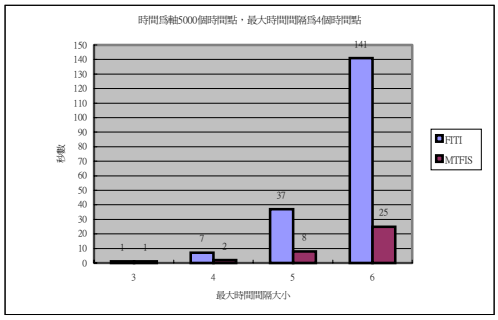

圖 5.4

由於 MTFIS 演算法採用一些額外的資料結構以加速探勘過程，因此我們統計在前 3 個實驗所設定資料參數條件下執行所需記憶體空間，如圖 5.5 至圖 5.8 所示。而由圖 5.5 及 5.6 顯示 MTFIS 演算法所需之記憶體儲存空間呈線性成長，且儲存空間大小隨時間點數目增加而呈現線性成長。

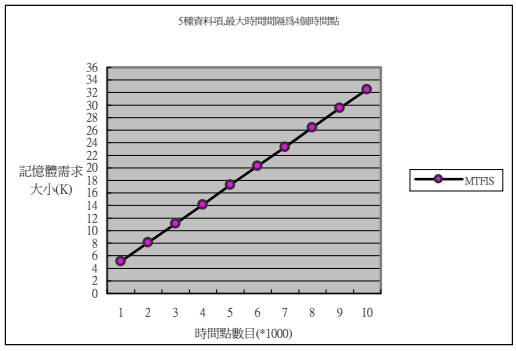

圖 5.5

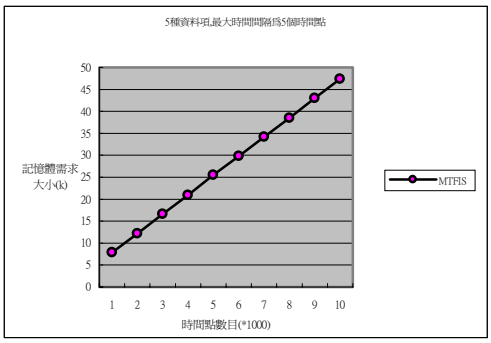

圖 5.6

另一方面，由圖 5.7 顯示，在資料項出現的最大間隔區間加大的情況之下，我們所提出的演算法對記憶體的需求並沒有很明顯的改變，反而是資料種類的多寡對記憶體的需求產生一定的影響(由圖 5.8 顯示)。主要的原因是，MTFIS 演算法中的時序性出現計數矩陣大小會因探勘區間加大而改變，因而需要額外的記憶體，但是增加的幅度非常小，甚至可以忽略。

綜合以上實驗結果，在進行時序性常見資料項集合的探勘過程，資料項種類個數、各資料項出現的最大間隔，以及時間軸上的時間點數目皆會影響執行效率，其中又以最大時間間隔的影響最為顯著。此外，在相同資料參數設定的條件之下，MTFIS 演算法的執行效率較 FITI 演算法有更好的表現。而影響 MTFIS 演算法記憶體使用量的主要因素則是資料項種類的多寡及時間點的多寡。

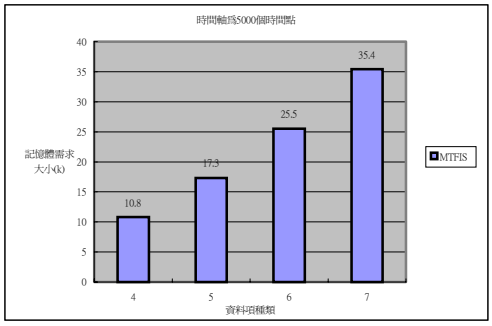

圖 5.7

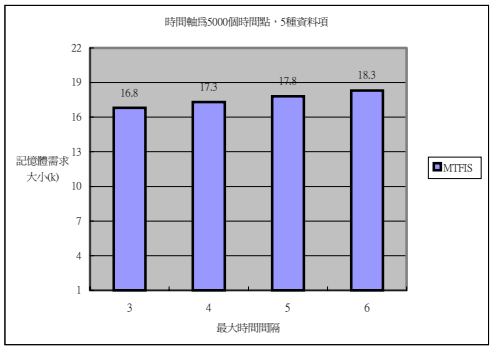

圖 5.8

六、結論與未來方向

在本論文中，為了能有效率地探勘出所有最大時序性常見資料項集合，我們提出 MTFIS 演算法，先探勘單一時間點上的最大常見資料項集合，為此，我們設計了出現計數矩陣，運用其中的資訊來減少候選常見資料項集合的個數，並利用位元索引序列快速計算出候選集合的出現次數。在第二階段的處理，我們以最大常見資料項集合為單位，以其中之間隔區間最大者為探勘視窗大小，並擴展出現計數矩陣成為考慮探勘視窗大小內時間點資訊之時序性出現計數矩陣，同時結合位元索引序列及 TIS 資料結構，可快速檢查時序性資料項集合組合的條件、其出現次數，及彼此間的時序性子集合關係，有效率地探勘出最大時序性常見料項集合。由實驗結果顯示，MTFIS 演算法比其他相關研究所提出的 FITI 演算法有更佳的執行效率。

應用本論文的研究成果，可運用在對時序性事件進行預測或是建立參考知識等決策系統之中。更進一步考慮，要如何對數值型態的時序性資料進行時序性的關聯探勘，並進而預測資料數值。

七、參考文獻

[1] R. Agrawal and R.Srikant, ”Fast Algorithm for Mining Association Rules in Large Databases,” in Proc. of The 20th International Conference on Very Large DataBases, 1994.

[2] C. Bettiniand and X.S. Wang, ”Discovering Frequent Event Patterns with Multiple Granularity in Time Sequences,” IEEE Transactions on Knowledge and Data Engineering , Vol. 10, No. 2, 1998.

[3] M.-S. Chen, J. Han, and P.S. Yu, ”Data Mining :An Overview from Database Perspective,” IEEE Transaction on Knowledge and Data Engineering, Vol. 8, No. 6,1996.

[4] L. Feng, H. Lu and J.X. Yu, “Mining Inter-Transaction Associations with Templates,” Proc. Of ACM International Conference on Information and Knowledge Management (CIKM99), 1999.

[5] J. Han, W. Gong and Y. Yin, “Mining Segment–Wise Periodic Patterns in Time-Related Databases”, in Proc. of 1998 International Conference on Knowledge Discovery and Data Mining(KDD’98), 1998.

[6] J. Han, G. Dong and Y. Yin, “Efficient Mining of Partial Periodic Patterns in Time Series Database ”, in Proc. of the 15th International Conference on Data Engineering, 1999.

[7] H. Lu, J. Han and L. Feng “Stock Movement Predication And N-Dimensional Inter-Transaction Association Rules,” ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 1998.

[8] A.K.H. Tung, H. Lu, and J. Han, “Breaking the Barrier of Transactions: Mining Inter-transaction Association Rules,” in Proc. Of Knowledge Discovery and Data Mining (KDD’99), 1999.

[9] X. Ye and J. A. Keane, “Mining Association Rules in Temporal Databases”, in Proc. Of 16th International Conference on Data Engineering, 1998.

[10] 房治政, “有效率探勘時序性常見資料項集合之方法研究,”碩士論文, 國立台灣師範大學資訊教育研究所, 2001.