從時序性資料中發掘非同步部份週期性規則

Mining Asynchronous Partial Periodic Patterns in Time Series Data

顏秀珍	李御璽	陳培炯	劉又誠
銘傳大學資工系	銘傳大學資工系	輔仁大學資工所	國立成功大學資工所
sjyen@mcu.edu.tw	leeys@mcu.edu.tw	Chen-34@yahoo.com.tw	r1750032@ss23.mcu.edu.tw

摘要

從時序性資料中找出週期性規則，在資料探勘中佔有很重要的角色，週期性規則是要找出某些常發生的事件，且週期而規律的發生。然而，在很多的應用上，事件並非理想上那樣連續且循環的發生，因此，允許時序性資料中有某些程度的雜訊是目前各種演算法發展的主要目標。在以往的研究中，關於週期性規則探勘的演算法皆是針對固定週期長度作處理，若要找出各種週期長度的規則，需要對不同的週期長度重新進行探勘程序。此外，對於雜訊的處理，以往的演算法也有可能遺失重要的資訊。本篇論文我們提出了精確而且有效率的方法，可於一次的探勘程序中同時找出各種週期長度的週期性規則。實驗的結果也顯示，我們的演算法不論在空間與時間上都有相當好的效率。。

關鍵詞：資料探勘、時序性資料、非同步部份週期性規則

一、導論

週期性規則探勘與序列規則探勘[3]的不同處在於除了找出事件發生的先後順序外，並且觀察這些事件是否有週期性的發生行為。週期性規則探勘的相關定義如下：一個*事件(Event)*是在某一時間可能發生的行為，所有可能發生的事件所組成的集合稱為*事件集合(Event Set)*。一個*序列*是由事件發生的時間先後順序排列而成。一序列中的某一位置可能存在有*任意事件"*"*，代表在序列中的這個位置的事件可以是任何事件。一個長度為 l 的序列(e_1, e_2, ..., e_l)是由 l 個事件所組成，其中 e_1~e_l 皆為事件集合中的事件或者"*"，且 e_1 即為此序列的*起始事件(Starting Event)*。一個*原始序列*是由我們開始觀察事件的發生，一直到觀察結束，這一連串的事件所形成的序列。

定義 一： 一個長度為 l 的序列 P = (p_1, p_2, ..., p_l)，及一個長度為 m 的原始序列 S = (d_1, d_2, ..., d_m)，對於某個位置j($1 \leqq j \leqq m$)，若 $p_1 = d_j$, $p_2 = d_{j+1}$,…, $p_l = d_{j+l-1}$，其中 p_i 亦可為＊($1 \leqq i \leqq l$)，則稱原始序列 S 中的子序列(d_j, d_{j+1}, …, d_{j+l-1})*支持(Support)*序列 P，或者稱序列 P *吻合(Match)*原始序列 S 中的子序列(d_j, d_{j+1}, …, d_{j+l-1})，我們稱原始序列 S 中的子序列(d_j, d_{j+1}, …, d_{j+l-1})為序列 P 的*吻合區塊(Match Block)*。

對於某一個序列，在原始序列中可能會有多個吻合區塊，而這些吻合區塊可能會發生*位置重疊(Overlap)*的情況。例如在圖一中，序列 P = (d1,d2,d1) 在原始序列 S 中共有三個吻合區塊，分別為 D1、D2 和 D3，其中吻合區塊 D1 和 D2 發生了位置重疊的情況，而這兩個重疊區塊，不能同時支持序列 P，因此在 S 中支持 P 的區塊只有兩個，分別是 D1 和 D3 或 D2 和 D3，其中，D1 和 D3 或 D2 和 D3 則為*非重疊*的吻合區塊。因此，對於某一序列，其在原始序列中，可能會有多組非重疊的吻合區塊。

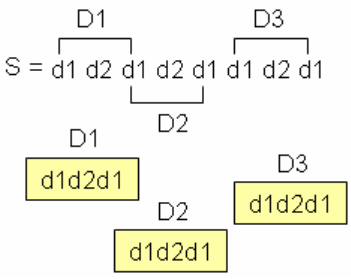


圖1.1 區塊重疊範例

對於某一序列 P，若其在原始序列 S 中有兩個非重疊的吻合區塊 D_i 和 D_j，則 D_i 和 D_j 之間的事件個數，我們稱為 D_i 和 D_j 之間的*距離(Distance)*。例如圖一中，D_2 和 D_3 之間的距離為 0，而 D_1 和 D_3 之間的距離為 2。

定義 二： 若序列 P 在原始序列 S 中存在有非重疊的吻合區塊 D_1, D_2, …, D_k，且 D_i 與

$D_{i+1}(1 \leqq i \leqq k\text{-}1)$之間的距離皆為 0，則我們稱 $D_1, D_2, \ldots, D_k$ 為 P 在 S 中之*連續的吻合區塊(Continuous Matching Block)*。若 D_1 與前一個吻合區塊的距離和 D_k 與下一個吻合區塊之間的距離皆不為 0，則 $D_1, D_2, \ldots, D_k$ 稱為 P 在 S 中的一個*片段(Segment)*，且 P 在此片段的*重複次數(Repetition)*為 *k*。若 P 在 S 中的某一片段中，其重複次數達到*最小重複次數(Minimum Repetition)*，則稱此片段為 P 在 S 中的一個*有效片段(Valid Segment)*。

若序列 P 的長度為 *l*，原始序列 S 的長度為 *m*，且 P 在 S 中的所有有效片段中最大的重複次數為 *r*，則 P 在 S 中的信心度(Confidence)為 *r* 除以 P 在 S 中最多可能的重複次數，如式(1.1)所示。

$$C_{S,P} = \frac{r}{\left\lfloor \frac{m}{\ell} \right\rfloor} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1.1)$$

若序列 P 在原始序列 S 的信心度大於*最小信心度(Minimum Confidence)*，則稱 P 為一個*同步週期性規則(Synchronous Periodic Pattern)*，且序列 P 的長度 *l* 即為規則 P 的*週期長度(Period length)*。

然而，對於序列 P，原始序列 S 中可能存在有某些雜訊，使得吻合區塊之間的距離不為 0，也就是吻合區塊之間可能會有雜訊存在，而導致 P 在 S 中的任一有效片段，其信心度皆無法達到最小信心度。因此，我們允許原始序列 S 有某些雜訊的存在，對於 P 在 S 中的兩相鄰的有效片段 S1 和 S2 之間的*距離(Distance)*為 S1 與 S2 之間的事件個數。若 P 在 S 中的有效片段為 $S_1, S_2, \ldots, S_n$，其中所有相鄰的有效片段 S_i 與 S_{i+1} $(1 \leqq i \leqq n\text{-}1)$之間的距離均沒有大於*最大距離(Max Distance)*，且 S_i 與前一個有效片段以及 S_n 與下一個有效片段之間的距離都大於最大距離，則可將 $S_1, S_2, \ldots, S_n$ 結合在一起，形成一個*連接序列(Connection Sequence)*，而最大距離則為兩相鄰片段之間的雜訊容忍度。

對於一長度為 *l* 的序列 P 及原始序列 S，令 $S_1, S_2, \ldots, S_n$ 為 P 在 S 中之某一連接序列 R 中的有效片段，且 P 在每一片段中的重複次數分別為 $R_1, R_2, \ldots, R_n$，若 S_i 與 S_{i+1} 之間的距離為 d_i $(1 \leqq i < n\text{-}1)$，則 P 在 S 中之連接序列 R 的信心度 $C_{R,P}$ 如式(1.2)所示。

$$C_{R,P} = \frac{R_1 + R_2 + \ldots + R_n}{\left\lfloor \frac{(R_1 + R_2 + \ldots + R_n) * \ell + (d_1 + d_2 + \ldots + d_{n-1})}{\ell} \right\rfloor} \ldots (1.2)$$

若 P 在 S 中之某連接序列的的信心度達到最小信心度，則 P 為 S 中的一個*非同步週期性規則(Asynchronous Periodic Pattern)*。若序列 P 中的事件允許有任意事件＊，則稱 P 為 S 中的一個*非同步部分週期性規則(Asynchronous Partial Periodic Pattern)*。

例如圖二，對於一個序列 P1 = (d1,d2,d3)及一個原始序列 S，若最大距離為 5，最小重複次數為 1，及最小信心度為 0.7，則兩相鄰片段 S1 與片段 S2 皆為有效片段，且 S1 與 S2 之間的距離為 5 沒有超過最大距離，如此 S1 與 S2 可結合成為一個連接序列，由式(2)可以算出 P 在 S 中之此連接序列的信心度為 $\frac{2+2}{\left\lfloor \frac{(2+2)*3+5}{3} \right\rfloor} = 0.8$，則 P 則為一個非同步週期性規則。而 P2 = (d1,＊, d3) 為 S 中的一個非同步部份週期性規則。

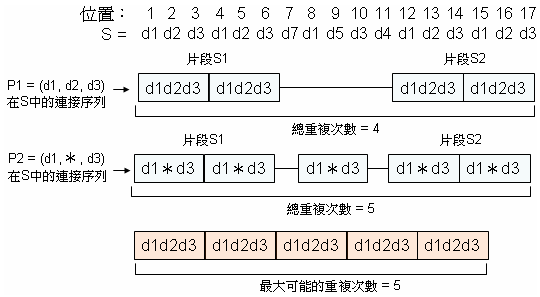


圖1.2 連接序列信心度計算範例

在早期的週期性規則研究中，大多著重於同步週期性規則[1, 2]：週期性規則的發生在原始序列中必定為緊密連接的，若一個原始序列中間參雜一些雜訊，依傳統的方法可能無法找出任何週期性規則。但在許多實際的應用上，也許會有些許的雜訊而造成有些規則延後發生的情形。

由於某一序列 P 在原始序列 S 中可能會有多組非重疊吻合區塊，有效片段會有位置重疊的情形，而導致連接序列也會有位置重疊的情形。因此，我們必須找出 P 在 S 之所有連接序列中，其重複次數最大的連接序列。若 P 在此連接序列的信心度滿足最小信心度，則 P 為 S 中的一個非同步週期性規則。

在部分(非同步)週期性規則探勘(Partial Periodic Pattern)的相關研究中[4, 5, 6, 7]，Jiawei Han 等幾位學者於 1999 年首先提出了這樣的觀念[2]，與傳統的週期性規則探勘主要的不同點就是：在部分週期性規則探勘中，能允許一個週期規則中，有某些事件是可以為任意事件。在探勘的方法上，由使用者設定一個*最小支持度(Minimum Support)*，若一個序列 P 在原始序列 S 上的出現次

數滿足最小支持度即為一個週期性規則。

對於這些論文所提出的方法，都要對於每個頻繁事件的每種長度重新做一次的週期性規則探勘的程序，例如若要找出以 d1 為起始事件且最大週期長度為 5 的所有規則，就必須要對週期長度為 2、週期長度為 3、週期長度為 4 及週期長度為 5 等四種長度分別進行探勘的程序。在一般的情況之下，頻繁事件將會是成千上萬個，而原始序列的長度皆是數以萬計，若最少重複次數設定得較小，其最大週期長度會相對的增長，其探勘過程將會花費相當可觀的處理時間。

有鑑於此，我們提出一個有效率的方法，從原始序列中找出所有的非同步(部份)週期性規則。我們可以先找出長度為 1 的非同步週期性規則，也就是*頻繁事件(Frequent Event)*，及以各個頻繁事件為起始事件的週期性規則可能之最大週期長度之後，再掃描一次原始序列即可同時找出各種長度的非同步(部份)週期性規則。

二、非同步部份週期性規則探勘

在這一章節中，我們詳細說明我們對*非同步週期性規則探勘*所提出的方法，我們利用 2 次的原始序列掃描，就可以找出所有週期長度的非同步週期性規則，在對原始序列進行掃描的過程之中，由原始序列的起始事件開始，依序掃描原始序列中的每個事件，當掃描到一個事件 d 時，此事件即為*目前掃描事件(Current Event)*，而事件 d 在原始序列中的位置則為*目前掃描位置(Current Position)*。

我們的演算法大致可分成下列幾個步驟，這些步驟的細部解說將會在下面的章節說明，步驟如下：

一、 資料的過濾及轉換：
這個步驟主要目的在於使我們演算法能夠應用在所有與時間有關的資料上，將要進行探勘的資料轉換成由時間順序事件排列而成的時序性資料。
例如在傳統的資料表中，將不需要的一些欄位資訊去除，定義事件(Event)並將事件依照時間先後順序排列，即為原始序列。

二、 找出頻繁事件及最大長度：
在這個步驟中，利用一次對於原始序列的掃描，找出頻繁事件，以及以此頻繁事件為起始事件之規則的可能最大週期長度。

三、 找出所有的非同步週期性規則：
此步驟根據使用者所設定的參數：最小重複次數、最大距離和最小信心度，以及前一步驟所得到的結果，找出所有週期長度的非同步週期性規則。

2.1. 資料的過濾及轉換

大多數的資料都是需要做一些轉換的動作，才可以正確的找出我們想要找的規則來。例如：若是我們想要觀測在本地天氣氣象的週期性，在原始的資料中必定會儲存像是地點、氣溫、晴雨、溼度等資訊，而我們想觀察的只是會降雨的情況，這時我們可能依照雨量分級：未下雨為"事件一"、下雨 10mm 以下為"事件二"；以此類推，定出每個事件並將其他不需要的資訊過濾，由此可得一連串依時間順序排列的序列，就可以使用在各個週期性規則探勘的演算法中進行探勘。

另一方面，在傳統的資料表型態的資料表中，以交易資料表為例，可能會記錄的內容包括：交易編號、客戶編號、交易時間、交易的物品及數量...等欄位，這樣的資料並不能直接運用在週期性規則探勘演算法中，因此需要將一些不需要的資料去除，並加以轉換成時間序列資料。

2.2 找出頻繁事件及最大長度

將資料轉換成原始序列後，我們首先要找出在原始序列中有週期特徵的頻繁事件，及以此頻繁事件為起始事件的週期性規則可能的最大長度。

找出頻繁事件的目的即為在往後的探勘程序中，只需要對於以頻繁事件為起始事件的序列進行分析即可。對於一個事件集合中的事件 d 以及一個長度 l ，若在原始序列 S 中事件 d 每 l 個位置發生一次，若其發生次數大於或等於最小重複次數則稱事件 d 為一個*頻繁事件(Frequent Event)*，所有的頻繁事件所組成的集合稱為*頻繁事件集合(Frequent Event Set)*，而長度 l 則為事件 d 的一個*頻繁週期長度(Frequent Periodic Length)*，在事件 d 的所有頻繁週期長度中最大的週期長度即為事件 d 的*最大週期長度(Max Periodic Length)*。

對於一個原始序列中的任一個子序列 P ，若序列 P 的長度沒有大於 P 的起始事件之最大週期長度，則序列 P 有可能成為一個非同步週期性規則，我們稱其為*候選規則(Candidate Pattern)*。

找出頻繁事件可以先刪去一些以不頻繁的事

件為起始事件的規則；而找出頻繁事件的最大週期長度，則可以刪去長度大於事件之最大週期長度的規則，有效降低候選規則的數量。例如：在圖 2.1 中，若最小重複次數為 3 ，則在原始序列 S 中所找出的頻繁事件集合為 {d2, d3}，而事件 d2 的最大週期長度為 3，事件 d3 的週期長度也為 3 。因此，候選規則只包括以事件 d2 和 d3 為起始事件，且長度最多為 3 的序列。另外，因為事件 d1, d4, d5 ,d6, d8 和 d9 均不為頻繁事件，故以這些事件為起始事件的序列均不能成為候選規則。

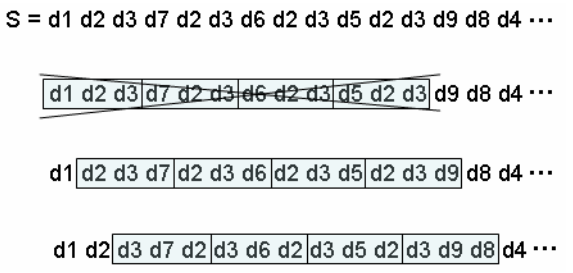


圖2.1 週期性規則的選取

在一個原始序列中，週期性出現的次數能達到最小重複次數的事件並不多，而頻繁事件的週期長度均遠小於原始序列的長度。因此，利用頻繁事件及針對每個頻繁事件找出最大週期長度可以大幅的減少候選規則的數量，同時由於需要處理的候選規則數量減少了，也加快了規則探勘演算法的執行時間。

在找出頻繁事件的方法上，我們對於任一個事件 di 的每一個可能的週期長度 l 設定一個變數 $C_{d1,\,l}$ ，用來記錄以事件 di 為起始事件之長度 l 的序列連續出現的次數，然後分別依照事件及其週期長度分類存放於如圖 2.3 的樹狀結構中。在圖 2.2 中，事件 d1, d2, …, 和 dn 為出現在原始序列的事件。

首先，我們依序對於原始序列中的每一個事件做一次的掃描，在掃描的過程中動態建構如圖 2.2 之樹狀結構，當於原始序列中掃描到一個事件時，若此事件之前並無發生(也就是在樹狀結構中並無此事件)則在 Root 下新增一子節點用以儲存此事件之所有可能之週期長度的連續發生次數，並記錄此事件在原始序列中的位置。若此事件為已存在結構圖中的事件，則計算此事件與之前所有同樣事件在原始序列中的距離，以求出週期長度，並記錄此事件可能的週期長度之發生次數，以及此事件在原始序列中的位置。掃描完原始序列後，即可找出以每個事件為起始事件的所有可能之週期長度的發生次數，若一事件的某一週期長度的發生次數滿足使用者所設定的最小重複次數，則此事件為頻繁事件。對於某一頻繁事件，其所有滿足最小重複次數的週期長度中最長者即為此頻繁事件的最大週期長度。

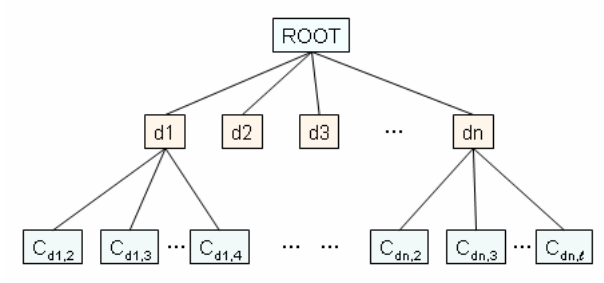


圖2.2 候選頻繁事件及其可能之週期長度的發生次數儲存結構

範例 2.1

在圖 2.3 中，我們依序從原始序列 S 的第一個位置開始掃描。掃描之前，頻繁事件儲存結構只有一個根節點，如圖 2.3 (a)所示。當掃描到第一個位置的事件 d1 時，由於之前並無 d1 存在，所以，在根節點下新增此事件 "d1" 並記錄其位置為 1，如圖 2.3(b)所示。同樣的，掃描到第二個位置時，產生一個新的事件節點 "d2" 並記錄其位置為 2，如圖 2.3(c)所示。在掃描到第三個位置時，由於事件 d1 已在結構中出現，我們計算此位置與 d1 之前出現之位置的距離可以得知其長度為 2，所以於事件 d1 下新增一節點 $C_{d1,2}$、發生次數為 2。依此類推，一直掃描完整個原始序列後，依序對於每個事件找出最大的週期長度，對於 $C_{d1,2}$ 而言，d1 最後出現的位置加上其週期長度再減去 1 為 5+2-1=6，不大於原始序列長度，因此 d1 之最大週期長度為 2 的出現次數仍保持 3 次。而對 $C_{d1,4}$ 而言，因為 5+4-1= 8 大於原始序列長度，因此 d1 之最大週期長度為 4 的出現次數為 $C_{d1,4}$-1 =2-1=1 次。假設我們設定最小重複次數為 3 次，則頻繁事件集合則為{d1}，其最大週期長度為 2。

由於我們所要找的是頻繁事件的最大週期長度，因此對於某一個事件，當其某一週期長度的發生次數已達到最小重複次數，且目前的掃描位置已經大於此事件最後出現位置加上此週期長度，則比週期長度短的週期長度則不需要再計算其發生次數，如此可以減少計算較短之週期長度的發生次數。

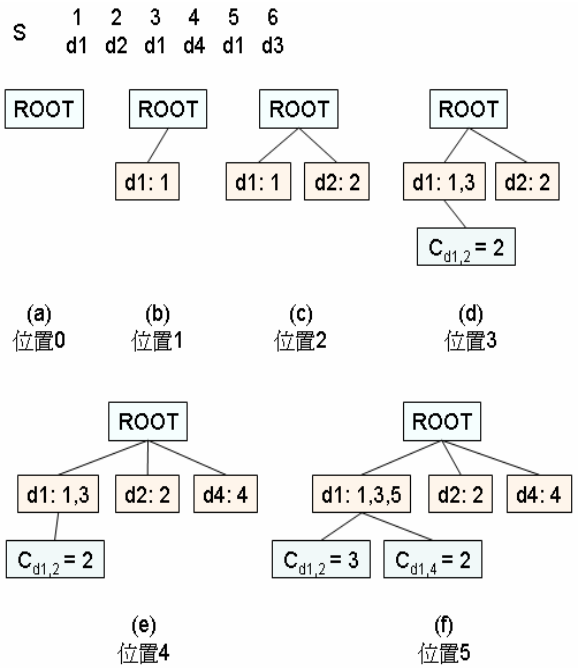


圖2.3 頻繁事件探勘範例

2.3 找出所有部份非同步週期性規則

在此小節中，我們詳細介紹如何利用前一步驟所找出的頻繁事件及各個頻繁事件的最大週期長度，和使用者所設定的參數：最小重複次數、最大距離和最小信心度，找出所有週期長度的非同步(部分)週期性規則。

在我們的方法中，第一次對原始序列的掃描可以找出頻繁事件及以每個頻繁事件為起始事件的週期性規則之最大週期長度；再進行第二次對原始序列的掃描即可找出所有的非同步(部分)週期性規則。對於在原始序列中掃描出的每一個事件 d，處理的程序可分成幾個步驟，當掃描完原始序列之後即可得到所有的非同步(部分)週期性規則，假設目前掃描到的事件 d 在原始序列中的位置為 i，其步驟如下：

一、 *產生暫存候選規則*：
我們利用*序列快取*來產生暫存候選規則。若頻繁事件的最大週期長度之中最大之長度為 Lmax，則序列快取即記錄在原始序列中第 j 個位置到第 i-1 個位置所形成的序列，其中 $\max(1, i - Lmax + 1) \leqq j \leqq i - 1$，將序列快取中的序列與事件 d 連接即可產生長度為 2 到長度為 Lmax 的暫存候選規則。

二、 產生候選規則並計算其重複次數：
對於上一步驟產生的暫存候選規則，若其起始事件為頻繁事件且週期長度不大於其起始事件之最大週期長度，則將此暫存候選規則與已產生之候選規則進行比對，以累計候選規則的重複次數或產生新的候選規則。

三、 連接序列的重疊分析：
由於比對的過程之中沒有判斷暫存候選規則是否與先前在原始序列之中所找到的連接序列或片段發生了位置重疊的情形，所以在此步驟之中判斷暫存候選規則是否可以與候選規則所形成連接序列繼續連接成更長的連接序列。

2.3.1 產生暫存候選規則

在此小節中，我們說明暫存候選規則的產生方式及儲存的結構。在儲存結構方面，我們利用*序列快取*及*暫存候選規則表(Temporal Candidate Table)*，簡稱*候選規則表(Candidate Table)*來完成。在掃描原始序列的過程中每讀進一個事件，我們將序列快取中的序列與此事件進行連接即可產生在此掃描位置的所有暫存候選規則。

由於時序性資料的特性，原始序列的長度都相當長，所以在演算法的設計上，我們希望在掃描一次原始序列的過程中，只向後面單向讀取資料，為了達成這樣的目的，我們利用序列快取儲存先前所讀取到的序列，而序列快取中的序列個數即為所有頻繁事件的最大週期長度中之最大長度 Lmax 減 1。

表2.1 序列快取及候選規則表結構

(a) 序列快取

長度	序列
1	d_{i-1}
2	$d_{i-2}\, d_{i-1}$
…	…
L_{max} - 1	$d_{i-Lmax+1} \ldots d_{i-2}\, d_{i-1}$

(b) 候選規則表

長度	暫存候選規則
1	$d_{i-1}\, d_i$
2	$d_{i-2}\, d_{i-1}\, d_i$
…	…
L_{max} - 1	$d_{i-Lmax+1} \ldots d_{i-1}\, d_i$

對於原始序列 S = d1, … ,di, … dn，在掃描到第 i 個位置的事件 di 前，序列快取中有 min (i, Lmax – 1) 個序列如表 2.1 (a) 所示。當掃描到第 i 個事件 di 時，將序列快取中的每一序列與事件 di 連接，即可產生目前的所有暫存候選規則，如表 2.1 (b) 所示。因此，暫存候選規則表中，最多也只會有 Lmax - 1 個暫存候選規則。在掃描完第 i 個位置的事件 di 後，序列快取中的序列會被更新：將事件 di 放入長度為 1 的序列中，再分別將長度為 2 到長度為 Lmax – 1 的暫存候選規則放入序列快取中。

範例 2.2

假設有一原始序列 S = d1, d2, d3, d4, d5,…，且所有頻繁事件之最大週期長度中的最大長度為 5，所以序列快取與候選規則表的大小為 4 筆，當讀取到原始序列 S 的第 4 個位置 d4 時，序列快取的內容如表 2.2 (a)所示。當讀到第 5 個位置 d5 時，將序列快取中的每一個序列與 d5 進行連接即可得到表 2.2 (b) 的候選規則表。之後，再將 d5 放入序列中快取長度為 1 的位置，候選規則表中長度為 2 至長度為 4 的序列則放入序列快取長度為 2 到長度為 4 的序列中，即完成更新序列快取的動作，如表 2.2 (c) 所示。

表2.2 序列快取及候選規則表產生範例

序列快取(位置4)

長度	序列
1	d4
2	d3d4
3	d2d3d4
4	d1d2d3d4

(a)

候選規則表(位置5)

長度	暫存候選規則
2	d4d5
3	d3d4d5
4	d2d3d4d5
5	d1d2d3d4d5

(b)

序列快取(位置5)

長度	序列
1	d5
2	d4d5
3	d3d4d5
4	d2d3d4d5

(c)

2.3.2 產生候選規則與連接序列

當我們掃描到原始序列中的某一個事件時，會產生目前的暫存候選規則，對於每一個暫存候選規則，我們先檢查其起始事件是否為頻繁事件，若其起始事件不為頻繁事件則此暫存候選規則不可能成為週期性規則，故可忽略不需處理。若其起始事件為頻繁事件則檢查此暫存候選規則的長度是否大於其起始事件的最大週期長度，若其長度大於起始事件的最大週期長度則此暫存候選規則也不可能成為一個週期性規則，故可忽略不需處理。若此暫存候選規則的起始事件為頻繁事件，且其長度沒有大於其起始事件的最大週期長度，則我們分析並判斷此暫存候選規則是否可成為某一連接序列中的候選規則。

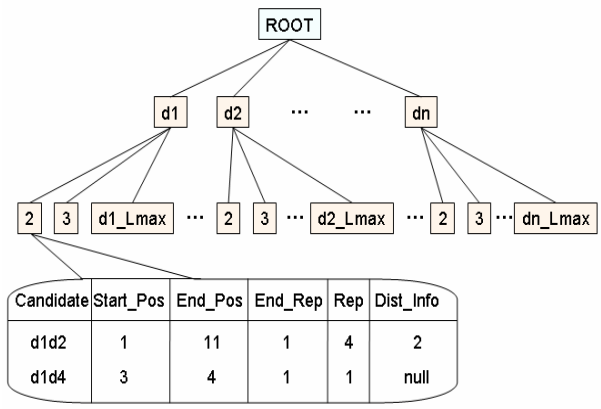


圖2.4 候選規則儲存結構

對於候選規則的儲存，我們根據候選規則的起始事件與其長度，將所有的候選規則儲存於一個如圖 2.4 的樹狀結構中，此樹狀結構與頻繁事件的儲存結構類似，根節點的每一子節點為頻繁事件，而以某一頻繁事件為起始事件的不同長度之候選規則，則儲存於此頻繁事件的子節點中。因此，每一葉部節點儲存以某一頻繁事件為起始事件的某一長度的所有候選規則，而每一候選規則所形成的連接序列之資訊則儲存於*候選規則儲存區*，每個候選規則暫存區中需要記載的資訊包含：候選規則本身、此候選規則在原始序列中目前所在之連接序列的第 1 個吻合區塊的*起始位置(Start_Pos)*、此候選規則在原始序列中最後一個吻合區塊的*結束位置(End_Pos)*、此候選規則在原始序列中在目前的片段中的*最後重複次數(End_Rep)*、此候選規則在原始序列中目前所在之連接序列的*重複次數(Rep)*、此候選規則在原始序列中片段與片段之間的*距離分布情形(Dist_Info)*。

如圖 2.5 中，若候選規則為(d1, d2, d3)且目前掃描到原始序列的第 14 個位置，使用者設定的最大距離為 3 ，則對候選規則(d1, d2, d3)而言，目前的連接序列是原始序列的第 1 個位置到第 14 個位置，因此，候選規則(d1, d2, d3)的 Start_Pos = 1, End_Pos = 14, End_Rep = 1, Rep = 4 和 Dist_Info = 2。

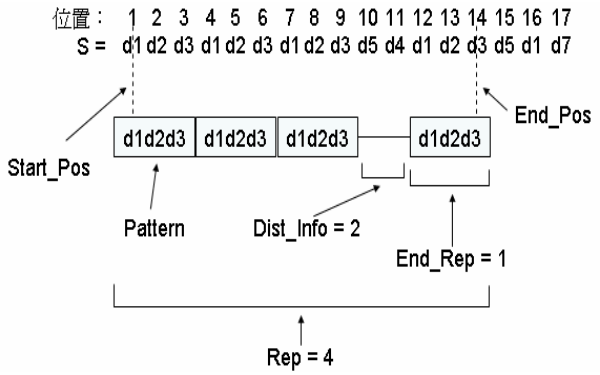


圖2.5 候選規則儲存區儲存之資訊

對於一個暫存候選規則 TC = $(d_1,d_2,\ldots,d_p)$ 和一個候選規則 C = $(e_1, e_2, \ldots, e_g)$，若 TC 與 C 的長度相同 (p = g)，且 $d_i = e_i$ $(1 \leqq i \leqq p)$ 或 e_i = ＊ $(2 \leqq i \leqq p)$，則我們稱 C 為與 TC 相同序列的*候選規則*。

Dist(TC, C)=P–TC 的長度–C.End_Pos … (2.1)

當我們掃描到原始序列中某一位置 P 的事件，產生暫存候選規則時，我們就要根據每一暫存候選規則 TC 的起始事件與其長度，從候選規則的儲存結構中，找出與其序列相同的候選規則，若無法找到相同序列的候選規則，則此暫存候選規則就成為一個新的候選規則，並將其相關資訊存入候選規則儲存區中，若有找到相同序列的候選規則 C，則利用式 (2.1) 計算出此暫存候選規則與其序列相同的候選規則在原始序列中的距離，若此距離大於使用者所設定的最大距離，表示此暫存候選規則在原始序列中無法與候選規則 C 目前所形成的連接序列繼續連接，也就是候選規則 C 目前所形成的連接序列已是一個完整的連接序列，因此，我們可以利用式 (1.1) 計算候選規則 C 在此連接序列的信心度，若其信心度小於使用者所設定的最小信心度，則此候選規則 C 不為非同步週期性規則，可將之從候選規則儲存區中去除，否則此候選規則為一非同步週期性規則。在找尋序列相同之候選規則的同時，暫存候選規則 TC 需與候選規則 C 的每一事件一一進行比對，若 TC 與 C 中某些位置的事件不相同，則會產生一候選規則，其中 TC 與 C 中事件不相同的位置為 ＊，其他位置則為 TC 與 C 共同的事件。例如：暫存候選規則 TC = (d1, d2, d3)，在儲存區之中分別有兩個候選規則：C1 = (d1, d2, d4)、C2 = (d1, d2, d3)，則 TC 與 C1 的比對結果為(d1, d2, ＊)，TC 與 C2 的比對結果為(d1, d2, d3)。

由於我們並非產生出所有的候選規則之後再進行原始序列的掃描，所以在候選規則的數量上將比先產生所有候選規則的方式少上許多。進一步分析候選規則的數量，在我們的方法中，根據找出來的候選規則最大長度跟使用者設定的最大距離即可求得於候選規則暫存區內的候選規則數量最大值。

在候選規則數量的分析上，每當產生一個暫存候選規則時，須與候選規則暫存區中的所有候選規則進行比對，比對的結果會有完全吻合及部分吻合的情形。以長度 2 為例，假設我們組合出的候選規則為 (d1, d2)，則與暫存區比對的結果可能會有❶ (d1, d2)(完全吻合)、❷ (d1, ＊)(部分吻合)兩種，依此類推可得到長度 3、長度 4、…、最大長度 Lmax，各個長度的候選規則所產生之比對的結果數量之最大值為：$1 + (2^{(\text{長度}-1)} - 1) = 2^{(\text{長度}-1)}$ 。於表 2.3 中分別以長度 2：d1d2、長度 3：d1d2d3、長度 4：d1d2d3d4 做為範例說明這三個規則長度所可能產生的所有比對結果。

表2.3 候選規則數量分析

長度2	長度3	長度4
d1d2	d1d2d3	d1d2d3d4
d1＊	d1＊d3	d1＊d3d4
	d1d2＊	d1d2＊d4
	d1＊＊	d1d2d3＊
		d1＊＊d4
		d1d2＊＊
		d1＊d3＊
		d1＊＊＊

在比對的程序上，將候選規則表中的每一筆候選規則逐一取出，對於一起始事件為 d1 且長度為 *l* 的暫存候選規則 C1，若 *l* 大於事件 d1 的最大週期長度 $d1_{Lmax}$，則此候選規則必無法成為週期性規則不需做處理；若是 *l* 小於或等於事件 d1 的最大週期長度 $d1_{Lmax}$，則於候選規則儲存區中找出所有同樣為以 d1 為起始事件且長度為 *l* 的後選序列，一一與 C1 進行比對。比對的方式上，由於第一個事件(起始事件)已經確定，所以只需針對第二個事件開始直到最後的事件進行比對，判斷是否為相同事件，若為相同的事件則比對的結果即為此事件，若為不相同的事件則比對的結果為＊。例如：在候選規則表中欲進行比對的一筆暫存候選規則為 C = (d1, d2, d3, d4)，候選規則儲存區中以 d1 為起始事件且週期長度為 4 的候選規則有 P1 = (d1, d4, d3, d5)、P2 = (d1, d2, d3, d4)，則比對的結果分別為 R1 = (d1,＊,d3,＊) 及 R2 = (d1, d2, d3, d4)兩種，我們稱 R1 和 R2 為*比對結果(Match Results)*，

將所有的比對結果放置於一個集合之中，稱為*比對結果集合(Match Result Set)*，依照比對結果中的序列，將相同的比對結果放置於比對結果集合之同一串列中，而每個比對結果所需儲存的資料為與其比對的候選規則在儲存區中的所有資訊(包括起始位置、結束位置，重複次數、最後重複次數等)，在上例中比對的結果 R1 需要儲存 P1 在儲存區中所儲存的資訊。

例如在圖 2.6 之中，有一個候選規則 Cand1，在暫存區中與 Cand1 相同起始事件及同樣長度的規則共有九筆，經過比對之後產生三種比對結果分別為比對結果一、比對結果二、比對結果三，則分別儲存於三個串列資料之中。對於每一個類型的比對結果，每一列的資料筆數即為此種規則類型的*吻合數(Match Count)*，例如圖 2.6 中的比對結果一的吻合數為 3、比對結果二的吻合數為 2、比對結果三的吻合數為 4。

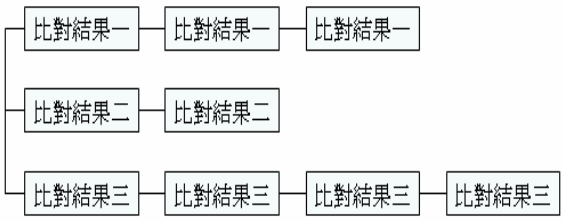
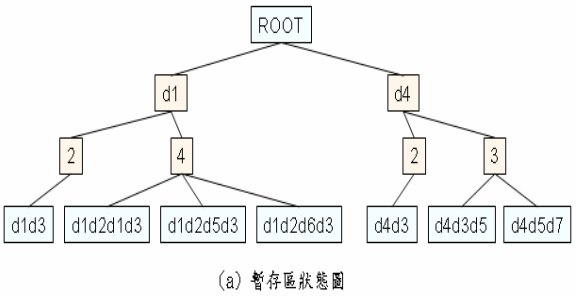


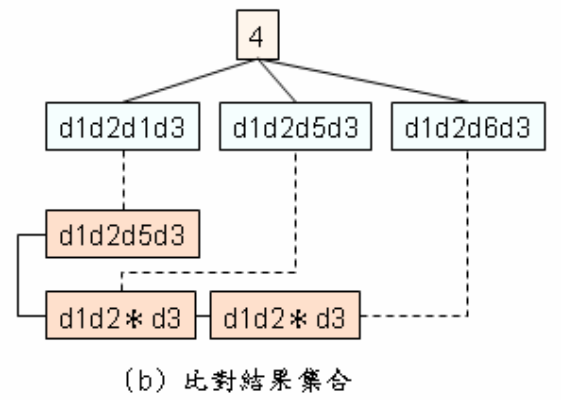
圖2.6 候選規則比對結果集合

範例 2.3

假設有一暫存候選規則 C = (d1, d2 ,d5, d3)，而候選規則儲存結構的狀態如圖 2.7(a)所示，則與 C 有相同起始事件及長度的共有 3 筆資料分別為：(d1,d2,d1,d3)、(d1,d2,d5,d3)、(d1,d2,d6,d3)，取此 3 筆資料進行比對後可以得到兩種比對結果分別為"d1d2d5d3"(吻合數為 1)及"d1d2＊d3"(吻合數為 2)，因此將這兩種共三筆的比對結果放入比對結果集合中如圖 2.7 (b)，即為最後比對結果。



(a) 暫存區狀態圖



(b) 比對結果集合

圖2.7 候選規則比對範例

2.3.3 連接序列的重疊分析

當一個暫存候選規則完成比對的分析後，接著對相對應的候選規則計算重複次數，將比對分析的程序所得到的結果集合利用重疊及連接分析程序判斷此暫存候選規則能否連接於之前掃描過程之中所發現同樣的候選規則上，且若同時有多筆候選規則可連接的情況之下，則找出所連接之後重複次數為最多的候選規則(即找出最大的連接序列)。

若一個以事件 d1 為起始事件，長度為 l 的候選規則 C，若比對結果集合為空集合，即在之前的原始序列，掃描過程中尚未發生此候選規則的序列，或者之前發生的序列已經超過最大距離，則只需要將候選規則 C 存入儲存區中不需進行重疊及連接的分析。

對於比對分析的結果集合，則需要分別對每種不同的規則進行重疊及連接的分析，如圖 2.7 (b)中有兩種比對結果 (d1, d2, d5, d3) 和 (d1, d2, ＊, d3)，依序處理(d1, d2, d5, d3) 的重疊及連接分析之後，再對 (d1, d2,＊,d3) 的兩個比對結果分析。

依照暫存候選規則及欲進行重疊及連接分析的候選規則所形成之連接序列之間的距離，可分為下列幾種情形：

1. 暫存候選規則的起始位置在候選規則所形成之連接序列的結束位置之前(包含)，則為位置重疊 (Overlap)。

2. 若暫存候選規則與候選規則所形成之連接序列之間的距離為 0，即為直接連接的狀態，則不需作任何判斷，直接進行連接，並記錄目前片段中候選規則的重覆發生次數。

3. 暫存候選規則與候選規則所形成之連接序列之間的距離大於 0 且小於最大距離時，則須加以判斷是否可與此連接序列連接：若連接序列的最後重複次數有超過最小重複次數，則進行連接並紀錄距離情形；若沒有超過最小重複次數，表示此片段不為有效片段，則不可連接。若此連接序列扣除最後一個無效片段之後與暫存候選規則之間的距離沒有大於最大距離，則可連接並紀錄距離情形；若暫存候選規則無法與此連接序列繼續連接下去，則計算此連接序列的信心度，判斷此候選規則是否為非同步(部份)週期性規則，若為週期性規則，則再判斷此連接序列是否為所有相同之候選規則中，重覆次數最多的連接序列。

4. 暫存候選規則與候選規則所形成之連接序列之間的距離大於最大距離，則計算此連接序列的信心度，判斷此候選規則是否為非同步(部份)週期性規則，若為週期性規則，則再判斷此連接序列是否為所有相同之候選規則中，重覆次數最多的連接序列。

在重疊分析方面，利用候選規則所形成之連接序列的結束位置與暫存候選規則的起始位置即可判斷出是否此連接序列與暫存候選規則發生重疊情形。若是暫存候選規則的結束位置(即 Cur_pos)減掉規則的長度加 1 (即為暫存候選規則的起始位置)小於或等於候選規則的結束位置，即可判斷出發生了重疊。例如：在圖 2.8 中，片段 D 是候選規則 (d1, d2, d1) 於原始序列中的一個片段，其結束位置為 9，當掃描到位置 11 時可以找到暫存候選規則 C，結束位置為目前位置 (11)，因此可以推算出 C 的起始位置為 11 – 3 + 1 = 9，於是暫存候選規則 C 與片段 D 即為重疊的狀況。

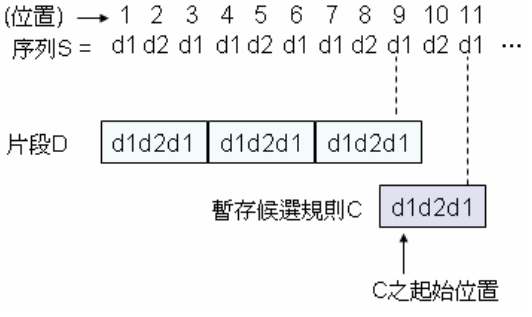


圖2.8 片段重疊範例

當發生重疊時，由於尚未掃描完整個原始序列，無法預知往後的序列情形，所以每當遇到重疊的情況時就先將重疊的暫存候選規則放入候選規則儲存區中，當往後的掃描過程之中再發現到相同的暫存候選規則時，再加以判斷是否可進行連接或者是又再度與所有連接序列發生重疊的情形。例如在圖 2.11 中的片段 D 與候選規則 C 發生重疊，所以在儲存區內新增候選規則 C (d1,d2,d1)、起始位置為 9、結束位置為 1、重複次數為 1、最後重複次數為 1，由於沒有進行連接的動作，所以其距離分布情形為 0。

在連接分析方面，如圖 2.9，假設連接序列 D1 為儲存區內存放的一個候選規則 (End_Pos = 9)，且使用者設定的最大距離為 3，最小重複次數為 1。我們利用三種不同候選規則的位置來分析連接時的三種狀況：對於暫存候選規則 C1 (End_Pos = 12)，可經由推算得知 C1 的起始位置為 10 ，所以可直接與 D1 連接；暫存候選規則 C2 (End_Pos = 13) 的起始位置為 11，與 D1 的距離為 1，沒有超過最大距離，所以可進行連接；同樣的可推算出暫存候選規則 C3 (End_Pos = 16)的起始位置為 14，與 D1 之間的距離為 4，超過了最大距離，所以不可進行連接，然後判斷形成 D1 的候選規則是否為週期性規則，並將此候選規則從儲存區中移除，再新增此暫存候選規則至儲存區之中。

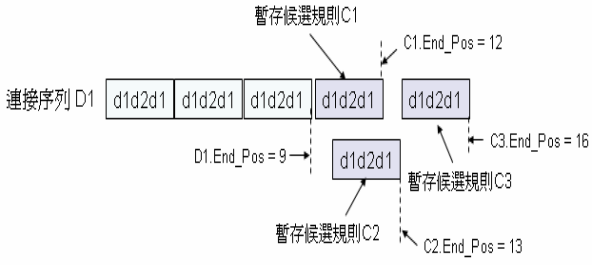


圖2.9 候選規則連接分析

上述的連接方法之中，由於設定最小重複次數為 1 次，不需對於未超過最小重複次數的部份進行處理，若是片段的最小重複次數設定為超過 1 次，直接連接的情形不受影響仍可直接連接，且超過最大距離的情形仍不可連接，若連接序列與暫存候選規則間的距離大於 0 而需進行連接時，就必須判斷此連接序列的最後片段有無超過最小重複次數(即為有效片段)，若此連接序列的最後重複次數沒有超過最小重複次數，則需將此最後的片段扣除，再判斷能否進行連接，若此片段的最後重複次數超過了最小重複次數則可進行連接，最後的重複次數則更新為 1 次。

週期性規則的儲存方式，與候選規則的儲存結構類似，根據每個起始事件分類，再分別對於每個週期長度加以細分，則可得到與圖 2.4 相同的樹狀結構。而結構中每個週期性規則所儲存的資料有規則本身、此週期性規則的重複次數(Rep)、此週期性規則所形成之連接序列的信心度(Confidence)。

在週期性規則結構中所儲存的各個週期性規則，皆為此規則在原始序列之中目前重複次數最多的連接序列，若是往後掃描的過程中發現到相同的週期性規則且其重複次數更多，則更新為重複次數多的連接序列，重複次數較少的就捨棄不用，如此當掃描完整個原始序列，就可以得到每個週期性規則的最大連接序列。

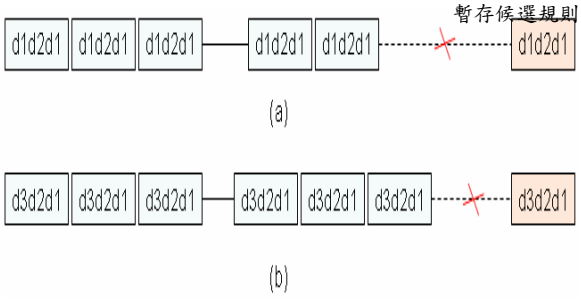


圖2.10 連接範例

範例 2.4

在圖 2.10 中，假設我們設定的最小重複次數為 3，其中 2.10(a) 與 2.10(b) 兩個連接序列與暫存候選規則之間的距離都大於最大距離，兩者都不可進行連接。由於最小重複次數為 3，所以 2.10(a) 中的連接序列，其最後的片段為無效片段，因此將此無效片段刪除，並計算此連接序列的信心度，若有滿足最小信心度，則輸出到週期性規則結構中、重複次數為 3；在 2.10(b) 中的連接序列，由於最後重複次數有超過最小重複次數，整個連接序列予以保留，並計算此連接序列的信心度，若有滿足最小信心度，則輸出到週期性規則結構中、重複次數為 6。

當有多個同樣序列的候選規則在儲存區中，暫存候選規則在進行比對程序時，我們會將儲存區中所有比對到的候選規則找出，並列於結果集合之中，再與暫存候選規則進行連接分析，在連接分析時我們檢查候選規則所形成的連接序列是否與暫存候選規則重疊或可進行連接。連接序列與暫存候選規則連接時，會有下列幾種情況：

1. 若暫存候選規則與多個連接序列皆可連接，且有一個最大重複次數的連接序列：由於只需找出一個最大重複次數的連接序列，所以只需連接上這個最大重複次數的連接序列，其餘連接序列皆可刪除。

2. 若暫存候選規則與多個連接序列皆可連接，且所有連接序列的重複次數皆相同：對於重複次數相同的連接序列，在最小重複次數及最大距離的條件之下是屬於相同的連接序列，差異只是在發的位置稍有不同，所以取一個距離暫存候選規則較接近的連接序列進行連接。

3. 有連接序列與此暫存候選規則的距離超過最大距離，而無法進行連接，但其重複次數是最多的：此時必須判斷能否將此連接序列與其他連接序列的片段進行剪接的動作，而產生長度最大的連接序列。

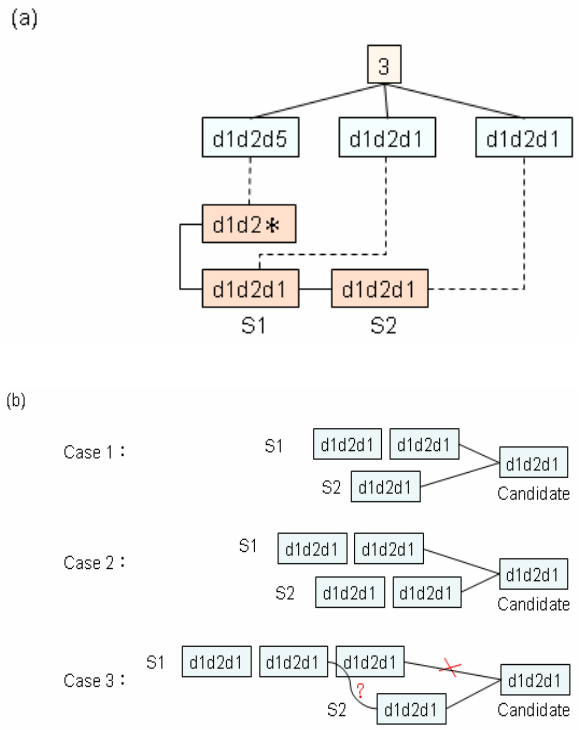


圖2.11 重疊分析範例

例如在圖 2.11 中，假設在候選規則表中的一個長度為 3 的暫存候選規則為 "d1d2d1"，而進行比對後的結果集合如圖 2.11 (a)所示，與此候選規則完全吻合的有兩筆資料 S1 及 S2，假設我們所設定的最小重複次數為 1，也就是當一個規則在序列中有出現了一次，就是可以連接的有效片段。在進行重疊及連接的分析時，共會有以下幾種情形：第一，如圖 2.11 (b) Case 1 所示： S1 與 S2 都可以進行連接，則取重複次數較多的 S1 與候選規則連接；第二，如圖 2.11 (b) Case 2 所示： S1 與 S2 皆可進行連接，且 S1 與 S2 重複次數相同，則取較接近的連接序列 S2 與暫存候選規則進行連接；第三，如圖 2.11 (b) Case 3 所示： S1 與暫存候選規則之間的距離超過了最大距離，不可進行連接，而 S2 與暫存候選規則的距離則在最大距

離之內，由於 S1 的重複次數大於 S2，所以須進行分析判斷能否將 S1 與 S2 連接成一個較長的連接序列。

當兩連接序列發生重疊時，我們可以利用在候選規則儲存區中所記錄的距離分布情形，來判斷是否可剪接成較長的連接序列，所以當此連接序列沒有與其它連接序列發生重疊的片段時(即在儲存區中此序列的候選規則只有一筆)，距離分布情形(Dist_Info)欄位只需儲存最後的兩個片段之間的距離即可，而往後再連接上一個片段時，之前所儲存的距離分布情形只需更新為連接之後的最後兩個片段間的距離。而當連接序列之間發生重疊時，於重疊的位置開始至此連接序列的最後，對於每一片段的重複次數及距離分布情形都需要作儲存。當進行剪接分析時必須利用重疊之後的距離分布情形來判斷是否可進行剪接的動作。

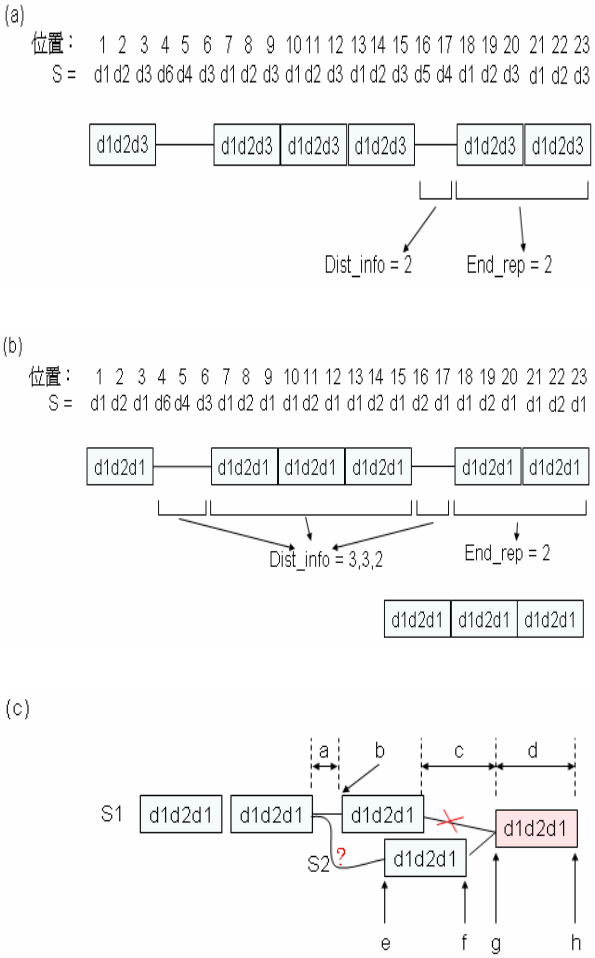


圖2.12 距離分布情形(Dist_Info)之紀錄

例如在圖 2.12 (a) 中，由於連接序列 S 並無與其他的子序列發生位置重疊的情況，所以我們在 Dist_Info 的欄位只須記錄最後兩個片段間的距離。當發生位置重疊時，則從位置重疊的位置開始，對於片段的重複次數及片段間的距離都需要記錄，如圖 2.12 (b)所示，由重疊的位置開始，其距離的分布情形依序紀錄片段的重複次數與片段間的距離。

圖 2.12 (c)中說明當由 S1 所組成的子序列與暫存候選規則進行連接分析時，由於超過最大距離不可連接，而 S2 的重複次數較少但可以與目前的暫存候選規則連接，則我們必須經由距離分布情形判斷是否能將 S1 及 S2 各擷取一部份進行連接。若我們要將暫存候選規則連接到 S2 最後一個區塊，再將 S1 的最後一個區塊移除後進行連接，則必須計算圖 2.13 (c)中第一個片段與 S2 最後一個區塊之間的距離，利用 S2 的最後一個區塊的起始位置(e) 與 S1 的結束位置扣去最後一個區塊(b)及最後兩片段之間的距離(a)即可求得。

在進行重疊與連接分析時，當有一個連接序列 CS，其重複次數超過所有同一候選規則所形成之連接序列的最大重複次數，但是其與暫存候選規則的距離大於最大距離數，則必須加以分析能否利用其他連接序列的片段與此最大重複次數的連接序列接成更長的連接序列。在剪接的方法上，由可與暫存候選規則連接的連接序列之最後一個片段開始，在維持最小重複次數的原則下，對於重疊的片段扣除一個區塊，每減去一個區塊即判斷能否與連接序列 CS 連接，若可進行剪接則將兩個連接序列合而為一，成為重複次數較多的連接序列。

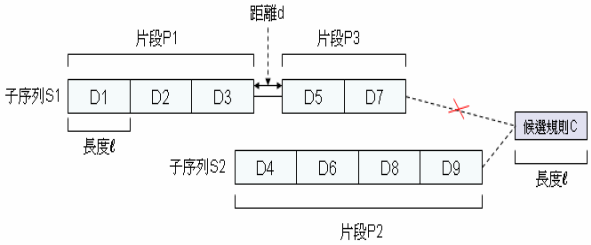


圖2.13 連接序列剪接範例

範例 2.5

在圖 2.13 中，若最小重複次數為 2 次，對於一暫存候選規則 C，於儲存區內分別有兩個連接序列 S1 及 S2 與暫存候選規則 C 完全吻合，則比對結果集合之中此種規則序列的吻合數為 2，在進行連接的程序時，發現連接序列 S1 與暫存候選規則 C 的距離超過最大距離而不可進行連接，而其重複次數大於另一連接序列 S2 的重複次數，所以須加以判斷是否能經由 S1 與 S2 有效片段間的剪接而連接上暫存候選規則 C，由 S2 的最後一

個片段的第一個區塊開始，也就是片段 P2 中的 D4，由於 D4 與 D3 重疊，不可連接，故刪去 D4，繼續檢查下一個區塊 D6 是否可與 D3 連接，因 D3 與 D6 的距離不大於最大距離，故可以將 D2 連接到 D4，形成一個更長的連接序列。

三、實驗報告

在這個章節之中，我們對此片論文所提出的演算法做執行效能以及候選規則數量(空間使用度)的評估，由於真實資料難以取得，所以我們的評估所使用的資料採用由程式所產生出的人工資料，利用程式參數的設定，模擬真實的環境，其人工數據的各項產生參數及產生的方式以及實驗之結果詳述如下。

3.1 人工資料的產生方式

由於演算法所處理的資料為時序性資料，我們產生的人工資料是由 1,024 個不同的事件組成，D = {d1, d2, …, d1024}，其總長度為 20M 的序列資料。為了模擬現實上各種不同的狀況，我們分別產生了四個不同的時序性資料，人工產生的序列資料分別利用利用下列幾個參數來模擬真實的狀況：

1. 由序列的起始位置開始，下一個規則的週期長度 l 的選擇是利用 *幾何分配(Geometric Distribution)* 且產生的平均長度為 PL 的方式產生。其中，規則中所出現的事件數目，數目由 1 至 l 利用隨機的方式從事件集合 D 中挑選出規則中的事件。

2. 每個規則的有效片段(Valid Segment)的數目，利用幾何分配且平均數目為 PS 產生。

3. 每個有效片段中的重複次數(Repetition)利用幾何分配且平均重複次數為 SR 次。

4. 在一個有效片段之後的距離(Distance)，利用幾何分配且平均數目為 SG 產生。

重複利用上面的序列產生方式產生原始序列，直到原始序列的長度為 20M 為止。我們產生的四個人工序列資料其使用的各個參數分別列於表 3.1 中。

3.2 實驗結果分析

在實驗之中，我們對於每個頻繁事件，均設定最大週期長度為 1000，且對於每一個週期長度 l ，設定最大距離數 Max_dist 為 $\frac{\min\_rep*l}{4}$ 。

表3.1 人工資料產生參數

原始序列	PL	PS	SR	SG
Seq1	5	5	50	50
Seq2	5	5	1000	1000
Seq3	100	1000	50	50
Seq4	100	1000	1000	1000

首先，我們觀察經由頻繁事件掃瞄之後可以刪除的暫存候選規則數量，在沒有進行頻繁事件及最大週期的程序之下，於原始序列之中所發現的任何暫存候選規則皆必須進行後續的比對及重複次數計數的動作，所以暫存候選規則的數量將可能達到 *CandMax* = (所有事件各數 * 最大週期長度) = 1024 * 1000 之多，而利用頻繁事件及各個頻繁事件的最大週期長度可有效降低不必要的探勘程序。如圖 3.1 所示，我們計算符合頻繁事件及在此頻繁事件的最大週期長度之內的暫存候選規則數量 *CandTemp*，利用 $\frac{CandMax - CandTemp}{CandMax}$ 可計算出由頻繁事件掃描的程序所刪減掉的暫存候選規則比率。

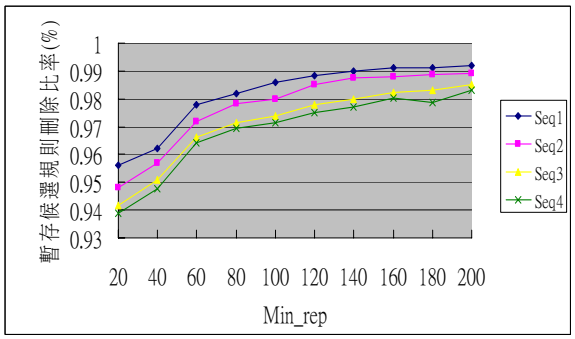


圖3.1 頻繁事件探勘之效果分析

由圖 3.1 之中可以發現，當最小重複次數漸漸增大時，經由頻繁事件掃描所刪除的暫存候選規則比率也逐漸增高，且在所設定的最小重複次數之中，刪減掉的暫存候選規則數量均可維持在 90% 以上的水準，對於演算法的效能及候選規則儲存區

的效能上有顯著的助益。由於在 Seq1 之中所設定的每個頻繁事件最大的週期長度為 5，所以在暫存候選規則刪除的數量上表現為最好，但由於重複次數的設定最大值為 50 ，相對的頻繁事件也較其他原始序列為多，所以並無法如理想上那樣能比其他的原始序列大幅刪除不必要的暫存候選規則，維持約 0.05% 的差距。在此實驗結果之中，可以發現在 Seq4 所刪除掉的暫存候選規則比率表現為最差，雖然重複次數設定的多(相對的頻繁事件較少)，但是由於最大規則長度較大且片段之間的距離也增大，因此會多產生出一些不必要的暫存候選規則。

在空間的利用度上，我們觀察在演算法之中所需保留在候選規則儲存區裡，候選規則數量的峰值，候選規則的最大值即為在圖 3.1 之中暫存候選規則的總數 *(最大週期長度 + 最大距離)。

由於此數值是理論上候選規則數量最大值，在現實執行時，並不會發生所有的部份週期規則均出現的情況，所以在候選規則峰值的數量上比理論上的最大值少上許多。此實驗中，均設定最小重複次數為 20，最大距離為 $\frac{5*\text{規則長度}}{4}$ ，如表 4.2 所示，在候選規則的數量上比理論上之最大值少上許多，在 Seq4 中，由於重複次數設定較大，所以頻繁事件數量較小，所以儘管在片段之間的雜訊數量比 Seq3 多，其候選規則數量的峰值比 Seq3 所產生的候選規則數量峰值較少。

表3.2 候選規則數量分析

原始序列	理論最大值	實際數量峰值
Seq1	1351680	311
Seq2	1351680	246
Seq3	36249600	53145
Seq4	36249600	42672

在演算法的執行效能上，利用我們的演算法分別對於四種人工資料進行探勘，並計算探勘過程耗費時間，結果如圖 3.2 所示，可以看出各種資料的變化上在我們的演算法之中影響並不是很大，在變動性大的資料(如 Seq3)上之執行時間並無大量增加，表示此演算法有相當的穩定性。由於我們演算法的執行效能與設定的最小重複次數及最大距離有關，所以當 Min_rep 設定的較大時可獲得更好的執行效能，執行時間隨著最小重複次數增大而遞減。

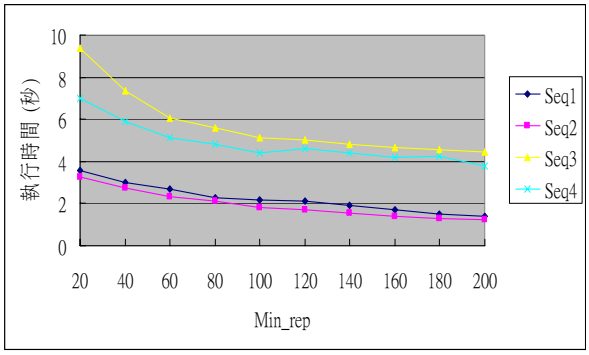


圖3.2 執行效能分析

四、結論與未來工作

在此篇論文之中，我們對於非同步部分週期性規則探勘提出了一個有效的探勘方式，同時由於在找尋最大連接序列時採用序列完全比對，並不會有遺漏規則的情形，且重疊的片段之數量也比只對於起始事件比對之方法少了許多，在重疊分析時就可省去很多不必要的運算時間。因此，除了能準確的找出非同步部分週期性規則外，對於探勘的效率上也能快速的將所有週期長度的規則找出，且無論原始序列資料上的變動多大，或者是探勘的序列長度增加，演算法均能維持一定的效能，從實驗的數據可以發現，隨著資料序列長度的增加，程式的時間也隨之緩慢的增加，呈現線性的成長，不會因資料量增加而執行時間大增，也顯示出此演算法的穩定性。

在未來的工作中，由於在時序性的資料序列當中，很多的應用上是不允許回朔的，也就是原始序列只能進行一次的掃描。在我們的演算法之中，若能將頻繁事件及最大長度的掃描程序加以改進，於掃描的同時決定最大長度並且找出週期性規則，則可將演算法改進成在一次的原始序列掃描之中完成所有規則的探勘。

另一方面，由於對於規則的衡量標準採用較為一般化的最小重複次數、最大距離及最小信心度，並非完全適合所有的應用上使用，因此若能找出更能符合週期性規則的衡量標準，將可使探勘出的規則更加的具有價值。

在週期性規則方面，由於演算法所找出的規則只根據事件發生的先後關係，若能更進一步找出每

個事件之間的確切時間，且最小距離也不只是相隔了多少事件，可以更精確的設定間隔的時間，提供更為精準的週期性規則。

五、誌謝

這篇論文的研究成果是國科會計劃（NSC 94-2213-E-130-004 和 NSC 94-2622-E-130-001-CC3）的一部份。我們在此感謝國科會經費支持這個計劃的研究。

六、參考文獻

[1] V. Guralnik and J. Srivastava, “Event Detection from Time Series Data”, Proc. ACM SIGKDD, pp. 33-42, 1999.

[2] Jiawei Han, Guozhu Dong, Yiwen Yin: “Efficient Mining of Partial Periodic Patterns in Time Series Database”, IEEE ICDE Proceedings of the 15th International Conference on Data Engineering, pp. 106-115, 1999.

[3] Jian Pei et al. “Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach” IEEE Transactions on Knowledge and Data Engineering, Vol.16, No.10, October 2004

[4] Jiong Yang, Wei Wang, and Philip S. Yu: “Mining Asynchronous Periodic Patterns in Time Series Data”, Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 275-279, 2000.

[5] Jiong Yang, Wei Wang, and Philip S. Yu: “Infominer: Mining Surprising Periodic Patterns”, KDD’01 San Francisco CA USA ACM, pp. 395-400, 2001.

[6] Jiong Yang, Wei Wang, and Philip S. Yu: “InfoMiner+: Mining Surprising Periodic Patterns with Gap Penalties”, Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM), pp. 725-728, 2002.

[7] Jiong Yang, Wei Wang, and Philip S. Yu: “Efficient Mining of Partial Periodic Patterns in Time Series Database”, IEEE Transactions on Knowledge and Data Engineering, Vol.15, No.3, pp.623-628, May/June 2003.