國 立 成 功 大 學
資 訊 管 理 研 究 所

碩 士 論 文

探勘多期資料下樣式變化之研究

Mining changes of patterns from multi-period datasets

研 究 生：戴 群 達
指 導 教 授：王 逸 琳 博士

中 華 民 國 九 十 六 年 六 月

摘要

資料探勘可以找出隱藏在資料下的知識，其中包括樣式發現；一般而言，樣式被認為是一類很重要的知識，因為它代表在大量資料中某些有意義的事件，同時也可以用來產生關聯法則。因此，瞭解樣式的變化是一個重要的研究議題。變化探勘即是為了瞭解資料的變化情況，期望更進一步地協助探勘者制訂決策；然而過去關於樣式變化的研究，皆是採用探勘頻繁樣式的處理程序，也就是在第一階段先刪除不夠頻繁的樣式，只保留符合使用者所設定門檻值下的樣式。如此做法不但會遺失部份資訊，亦不甚合理，因為變化探勘的目標是尋找有「變化」的樣式，而不只是變化的「頻繁」樣式；另外，大部份的研究皆只是比對兩期的資料，但若能觀察愈多期的資料，則所得到的結論也將會愈可靠；若能處理多期的資料，也就可以進行時間性資料探勘，譬如分析每日、每月或每季資料的變化情況等以提供管理者有用的資料變化資訊，方便其掌握市場的變動趨勢或開拓新的客源。由於相關議題在文獻中並無太多著墨，因此我們將以探勘多期時間下樣式的變化為主要的研究議題，期能提供不同於探勘頻繁樣式之外的另一個研究方向。

本研究提出一個新的演算法來探勘所有樣式；並直接採用以樣式的成長幅度作為篩選的標準，以篩選出所有符合變化趨勢的樣式。為了改善探勘效率，本研究發展一個候選樣式森林的特殊資料結構及演算法以探勘多期資料的變化；並為該特殊的資料結構設計一套彈性的的機制以增加發現樣式變化的可能性，以及可以避免太多不具參考價值的樣式。為了測試本研究所提出之演算法效率，我們另外設計並實作了一套以探勘頻繁樣式為基礎之演算法來探勘多期時間下樣式的變化。進行一連串的測試結果之後，我們發現本研究所提出之資料結構與演算法的確可以非常地有效率探勘多期時間下樣式的變化，而且探勘愈多期的資料集合

愈能突顯本研究所提出的演算法之優越性。此外，我們亦針對各期資料集合間的相似度，以及樣式的變化率門檻值對不同的變化探勘演算法所造成的影響加以探討。

關鍵字:樣式發現；變化探勘；候選樣式森林；時間性資料探勘；演算法；資料結構

Abstract

Pattern discovery is a common task in data mining. Given the transaction datasets of multi periods, we are concerned with a temporal data mining problem that detects any pattern of interested changes that have been consistent from some period to the last period. Discovering such changes from the transaction database of multi periods will help the managers to detect the tendency of customer needs so that potential customers may be identified. To the best of our knowledge, previous studies in change mining only focus on datasets of two datasets, although the tendency of changes are more meaningful for datasets of multi periods in real-world applications. Conventional data mining techniques that seek frequent patterns could be modified for mining changes from datasets of multi periods, but such approaches would require many pairwise comparisons between datasets of consecutive periods and thus not so efficient.

In this thesis, we propose an algorithm called MCP for mining changes from multi-period datasets. MCP is based on a novel data structure modified from the popular frequent-pattern tree (FP-tree), and seeks the target patterns in a very efficient way. In particular, starting from the last two periods, our algorithm first constructs a candidate-pattern forest (CP-forest) to store those patterns of qualified changes, and then iteratively updates the CP-forest using the dataset of each period. The CP-forest is carefully designed such that useless information will not be stored and qualified patterns can be easily identified by tree traversals.

Computational experiments have been conducted to compare MCP and another algorithm called modiFP which is modified from the popular FP-growth algorithm for

mining the changes of patterns from multi-period datasets. Several parameters have be used to evaluate the performance of MCP and modiFP, and the results show that MCP is much more efficient than modiFP, especially when the number of periods increases or when the datasets of consecutive periods share more similarities.

keywords: pattern discovery, change mining, candidate-pattern forest, temporal data mining, algorithm, data structure

誌謝

本論文能夠完成，需要感謝的人太多了。首先，要感謝我的指導教授王逸琳老師在研究所求學過程中的悉心教導。在老師的指導下，不但讓本論文更加地順利進行，而且也讓我學習到該用更高的標準去要求自己，相信這對我在日後做人及做事上都有很好的幫助。除此之外，也要感謝所上的老師在修課中的教學，使我能夠更加充實我的知識。同時，也要感謝論文口試的李宇欣老師以及王維聰老師所提供的寶貴意見，讓本論文能夠更加完整。

在這二年中，我也交到一群好朋友；我們在繁重的課業壓力下一起學習一起玩。師門中的修杰、惠娥、正翰、橙坤、姿君、正楠、建傑、姿儀與家宜；我們曾經在同一個屋簷下朝夕相處，這是我難忘的美好回憶。謝謝我的同窗：暉堡、冠良、盈安、岱穎、永耀、麗娟、光毅、武翰、欣洋、明哲、福吉、張凱與強棋等；我們在修課時一起討論，還有熬夜趕程式、做報告以及所有的娛樂活動都使我這二年過的很充實。當然，我也要感謝不在台南的諸多好友們：雅惠、一博、爵安與安柏等，雖然我們沒辦法時常在一起，但是我知道你們對我的關心始終沒有少過。謝謝你們，我所有的好友。

最後，我要把完成這份學業的榮耀獻給我最愛的父母親以及家人，沒有你們的支持，我肯定無法一帆風順地完成各階段的學業。謝謝你們的栽培與鼓勵，我將以更積極的態度來面對往後的人生。

目 錄

摘要 i

Abstract iii

誌謝 v

表目錄 viii

圖目錄 x

第一章 緒論 1

1.1 研究動機 2

1.2 研究目的 3

1.3 論文架構 3

第二章 文獻探討 5

2.1 探勘頻繁樣式與關聯法則 5

2.1.1 探勘頻繁樣式與關聯法則相關符號表示 5

2.1.2 探勘頻繁樣式與關聯法則演算法 6

2.2 時間性資料探勘 8

2.3 變化探勘 9

2.3.1 Change detection 9

2.3.2 Emerging pattern 11

2.3.3 Exception rules 13

2.3.4 Same attribute and same cut 13

2.4 小結 16

第三章 探勘樣式的變化 **18**

3.1 問題定義 18

3.2 相關議題 21

3.3 MCP 演算法 23

3.3.1 建立原始搜尋樹架構 24

3.3.2 挑選候選樣式及建立樣式森林 28

3.3.3 探勘多期變化樣式 32

3.4 時間容忍 34

3.5 範例說明 36

3.6 小結 44

第四章 資料測試與分析 **45**

4.1 資料產生器說明 45

4.2 modiFP 演算法 46

4.3 測試結果 47

4.3.1 驗證期數的影響 47

4.3.2 驗證相似度的影響 49

4.3.3 驗證支持度的影響 51

4.3.4 驗證變化率的影響 53

4.3.5 驗證大資料檔 56

4.4 小結 59

第五章 結論與建議 **60**

5.1 結論 60

5.2 後續研究建議 61

參 考 文 獻 **63**

表 目 錄

2.1 border 參數設定 12

3.1 第 3 期的資料 38
3.2 第 2 期的資料 39
3.3 排序後第 3 期的資料 39
3.4 排序後第 2 期的資料 40
3.5 排序後第 1 期的資料 41
3.6 全部的變化樣式 44

4.1 參數與預設值 46
4.2 期數對執行時間的結果表 (時間單位 : 秒) 49
4.3 相似度對執行時間的結果表 (時間單位 : 秒) 50
4.4 支持度對執行時間的結果表 (時間單位 : 秒) 53
4.5 變化率對執行時間的結果表 (時間單位 : 秒) 56
4.6 大範圍資料中相似度對於 MCP 演算法的執行時間結果表 (時間單位 : 秒) 57

圖 目 錄

2.1 FP-Growth:FP-tree 範例 7

2.2 兩個屬性且各有兩種可能值的搜尋樹 11

2.3 第 t 期的決策樹 14

2.4 第 $t+1$ 期的決策樹 15

2.5 依 RMT 判斷不同型態的變化 16

3.1 樣式的變化 19

3.2 變化樣式與頻繁樣式的概念圖 19

3.3 樣式的三種變化 21

3.4 兩期資料的處理 22

3.5 只有第 n 期資料的搜尋樹 25

3.6 更新後的搜尋樹 25

3.7 具有時間容忍值的圖形 1 35

3.8 具有時間容忍值的圖形 2 36

3.9 時間容忍值單位為 3 的圖形 36

3.10 排序 38

3.11 加入第一筆資料的原始搜尋樹 39

3.12 第 3 期資料的原始搜尋樹 40

3.13 加入第 2 期資料的原始搜尋樹 41

3.14 挑選項目 b 候選樣式的處理程序 42

3.15 所有的樣式樹 43

3.16 更新第一期資料後的樣式樹 43

3.17 修剪過後的樣式樹 43

4.1 modiFP 探勘流程圖 48

4.2 驗證期數 49
4.3 驗證相似度 50
4.4 緊實與鬆散的搜尋樹 51
4.5 MCP 演算法在不同相似度的樣式比對次數圖 52
4.6 驗證支持度 54
4.7 MCP 演算法與 modiFP 演算法在支持度為 0% 與 1% 的執行時間比較 . 54
4.8 MCP 演算法與 modiFP 演算法在支持度為 0% 與 5% 的執行時間比較 . 55
4.9 驗證變化率 56
4.10 驗證大範圍資料中相似度對 MCP 演算法的影響 58
4.11 驗證大範圍資料中支持度對 MCP 演算法的影響 58

第一章

緒論

現今所處的環境中，每天皆產生無數的資料，而如何管理並利用這些資料便成了一個很重要的問題。資料探勘(data mining)即是找尋隱藏在大量資料下具有意義而且讓使用者感興趣的知識。資料探勘包含了預測及解釋的能力，透過資料探勘的各種技術可以用來處理分類、分群及發現資料間的關聯性等問題。近年來資料探勘已被廣泛的應用在醫學、化學及商業等領域中，而資料探勘本身更是一個熱門的研究領域。

發現資料集合中的變化是資料探勘的主要任務之一(Kantardzic, 2003)，而這些資料庫(database)可能是顧客的交易資料(transactions)、網路的瀏覽日誌檔(log files)或是生物上的基因序列資料(DNA sequence)。若能從資料中探勘出頻繁樣式(frequent pattern)就可以發現這群資料中較具意義的特性，進而排除大量的零散資料。舉例來說，探勘交易資料庫可以協助管理者瞭解顧客的購買偏好及購買行為，進而依照探勘所得之資訊將這些經常被購買的商品擺放在同一個架位，或者是搭配銷售，如此不但可以增加顧客滿意度而且也能提升銷售量。然而，若想進一步瞭解購買偏好的改變，或者是哪些商品的銷售是否有顯著上升或下降的趨勢等議題，就要透過變化探勘(change mining)以找出資料的變化趨勢。另外，由於傳統的資料探勘技術通常會將所有的資料視為一個大集合，而忽略時間性因素(Antunes and Oliveira, 2001)。因此，探討資料隨時間變化的時間樣式(temporal pattern)探勘亦為變化探勘研究領域當中的重要議題。

1.1 研究動機

Liu et al.(2000)指出，長期以來資料探勘的研究大都專注在建立一個精確的預測模型或是有效率的探勘方法，而很少探討資料的變化情形；然而，我們可藉由發現資料的變化及其原因來藉此調整原本的決策並進行更深入的分析。因此，探勘資料的變化趨勢為一重要的研究課題。目前探討有關樣式變化的研究仍然不多，Song et al.(2001)使用Apriori演算法求出關聯法則(association rule)，再透過法則比對(rule matching)以找尋法則的變化，這種方法雖然容易理解，但其效率不彰；Kim et al.(2005)為了在動態的環境中偵測資料特性的改變，採用決策樹(decision tree)比對的方法，卻又受到決策樹本身結構的限制，只能比對兩期資料；同樣地，Dong et al.(2004)所提出的演算法亦只能探勘兩期資料的變化情形；因此，針對兩期以上資料變化的相關議題，在文獻中並無著墨。根據大數法則(Law of Large Number)的主旨，當對現象的觀察次數愈多，所獲得的結論也將愈可靠；所以在現實的環境中，若能連續觀察到兩期以上的資料變化，即可就該變化現象加以探討深究，提出相關之對應決策加以處理，並再探勘更新一期的資料，由其變化趨勢以驗證決策的有效性。

探勘變化樣式必須進行樣式的比對，由於文獻中沒有相關議題之探討，若以直覺的比對方式加以處理，則幾乎必須針對每個樣式的每個屬性逐一比對。舉例而言，在兩期資料(t=2)下，假設每期資料集合各有p個樣式，而每個樣式包含m個項目，那麼就要進行p^2m^2次比對；同理可知，k期資料下則需要$(k-1)(p^2m^2)$次比對。因此在樣式與項目的數目皆很大的情況下，這種兩期間的逐一比對方式將耗費非常龐大的運算時間。

當資料庫中的某項目集合(itemset)其支持度(support)超過使用者所設定之最小支持度門檻值(minimum support thresholds)時，則此樣式即為頻繁樣式(frequent patterns)。資料探勘中關於樣式發現(pattern discovery)的研究，幾乎都集中在探勘頻繁樣式(mining frequent pattern)方面，而很少用於探勘其他類型的樣式。頻繁樣式代表著在資料庫中出現頻率很高的項目集合，一般在探勘頻繁樣式時，低頻率的項目集合普遍被認為沒有參考價值且數目眾多，因此通常都會被

排除。然而，探勘樣式變化與探勘頻繁樣式有著不同的目標，以支持度為標準而先行過濾樣式的處理方式並不適用於探勘樣式變化，因為支持度本身是以發生次數的多寡作為評估依據，若先以支持度作為篩選標準，很可能會忽視某些已發生變化但卻不夠顯著的樣式。舉例來說，若有一項目集合在第一至第五期間內，其支持度分別為：1%、2%、4%、8%、16%。當最小支持度門檻值被設為5%時，此樣式只能在第四及第五期因為足夠頻繁而被發現。然而，事實上此樣式每一期皆有兩倍的成長，且其成長趨勢理應自第一期即開始計算，而非自第四期。為了避免某些不夠頻繁的樣式能在探勘過程中不被忽略，本研究將提出有效率的新演算法以探勘所有滿足變化幅度的樣式。

1.2 研究目的

本研究主要探討如何有效地自多期資料中發現樣式的變化情形，並將這些樣式依其發生之時間先後依序找出。我們欲觀察這些樣式是否隨著時間而持續發生變化；如果能發現這些變化，就能在最短時間內針對這些現象作出反應。因此本研究主要目的為：發展一套能夠處理多期資料且有效率地探勘變化的樣式發現演算法，能夠避免採用先探勘「頻繁的程度」後計算「變化的幅度」的方式所造成遺失資訊的問題，以發現那些直至最後一期仍一直滿足變化率的所有樣式，提供給探勘者參考。由於文獻中所提出的方法僅將兩期資料交相比對，而無法探勘多期資料的變化，因此本研究亦將針對過去處理多期資料之相關研究的限制及缺點一一討論。

在研究限制方面，本研究僅處理離散型態 (discrete) 的資料，不討論資料屬性為連續型態 (continuous)。

1.3 論文架構

本論文的主要架構如下：第一章為緒論，說明本研究的背景及研究的動機，進而提出本研究所欲達成之目的；第二章回顧相關文獻，主要包括樣式發現、時間性資料探勘及變化探勘等相關研究成果；第三章將詳細說明本研究欲解

決之問題及所提出的解決方法，並以一個簡單的小例子輔助說明；第四章實作本研究提出的探勘演算法，並進行數值測試，以檢驗其實作效率；最後，第五章總結本研究的成果與貢獻並提出後續研究的建議。

第二章

文獻探討

資料探勘的研究通常包含預測(prediction)趨勢與解釋(description)現象兩大類。其中解釋現象乃針對資料的特性提供有價值的、有意義的資訊，使人們容易瞭解資料隱含的訊息，而本章所回顧的樣式發現(pattern discovery)即屬於解釋現象這一大類。由於樣式發現可被用來產生法則(rule)，因此被認為是項重要的任務。過去對於樣式發現的研究主要集中於探勘出頻繁樣式，而本研究的重點在於探勘出考量時間因素的樣式變化。

2.1 探勘頻繁樣式與關聯法則

在探勘出資料集合中的頻繁樣式之後，我們可由資料集合中尋找構成某些事件的相關項目，進而由這些項目的關係找出事件間可能的關聯法則(association rule)(Agrawal and Srikant, 1994)。由於關聯法則代表某些事件發生之後可能連帶造成另一相關事件也跟著發生，因此經常被應用於購物籃分析(market basket analysis)，藉由分析購物之交易資料，商家可發現顧客的消費行為；如知名的「啤酒與尿布法則」(Berry and Linoff, 1997)，原本很難想像這兩件商品竟會被同時購買，但經由關聯法則可得知此一資訊，便可幫助經理人制訂行銷策略。

2.1.1 探勘頻繁樣式與關聯法則相關符號表示

已知$I = \{i_1, i_2, ..., i_m\}$為全部項目所組成的集合，項目在不同應用領域有不同的代表意義，例如在購物籃分析時，一個商品即是一個項目。若將資料庫表示為D，則$I \in D$。項目集合(itemset)為數個項目所組成的集合，通常以*k-itemset*表

示由k個項目所組成的項目集合，$1 \leq k \leq m$，m為項目的總數；一般情況下，k遠小於m。已知現有兩個項目集合：X與Y，則$X, Y \subseteq I$且$X \cap Y = \emptyset$。衡量項目集合在D中的重要性指標為支持度(*support*)，$support(X)$即表示X在D中所佔的比例。另一個指標為信心度(*confidence*)，其用來衡量一條關聯法則的強度，即$confidence(X, Y)=\frac{support(X,Y)}{support(X)}$，$support(X, Y)$為$X$與$Y$同時在$D$中出現的機率。使用關聯法則時，使用者將設定兩個門檻值，即最小支持度門檻值(*minimum support threshold*)與最小信心度門檻值(*minimum confidence threshold*)，其意義為使用者所能接受最低程度的條件。當某一法則的支持度與信心度皆符合門檻值時，例如$support(X, Y)$>*minimum support threshold*與$confidence(X, Y)$>*minimum confidence threshold*，則此法則被認為是一條關聯法則，表示方式為$X \Rightarrow Y$。通常$\Rightarrow$的左手邊稱為條件部份(conditional part)或稱為LHS(left-hand-side)，在右邊的為結果部份(consequent part)或稱為RHS(right-hand-sede)，這表示若發生LHS，那麼RHS也很有可能會發生。

2.1.2 探勘頻繁樣式與關聯法則演算法

近年來，已有許多學者提出各式各樣的關聯法則演算法，根據Ceglar and Roddick(2006)指出，關聯法則可分為兩大典型，第一種為Candidate Generation Algorithms；這類的演算法會先產生大量的候選項目集合，之後再逐一驗證所產生的項目集合是否符合要求，若不符合要求即予以刪除，並繼續下一次的疊代直到終止條件成立為止。最具代表性的演算法為Apriori(Agrawal and Srikant, 1994)，此演算法首先掃描資料庫建立項目長度為1的候選項目集合並計算各自的支持度，其後將合乎最小支持度門檻值的的項目合併，使得項目長度為2，再進行下一次的掃描計算支持度；如此疊代直到沒有候選項目集合為止。由此可知Apriori演算法必須多次掃描資料庫以計算支持度，並且需要產生大量的候選項目集合，使得此演算法的效率降低，因此後續的研究如DIC(Brin et al., 1997)透過分割資料集合及使用查核點(checkpoints)的方法，降低掃描資料庫的次數，DHP(Park et al., 1997)提供了減少大量產生候選項目集合的方式；還有Partition Algorithm(Savasere et al., 1995)等，皆是為了改善Apriori演算法的效率。

另一種典型為Pattern Growth Algorithm，這類的演算法透過由item list及pattern frame所構成的特殊資料結構的操作，排除了產生候選項目集合的需要。此概念最先是由Han et al.(2004)所提出的，他們方法稱為FP-Growth演算法，使用樹狀結構作為pattern frame用來儲存項目，圖2.1是此結構的範例，其中包括

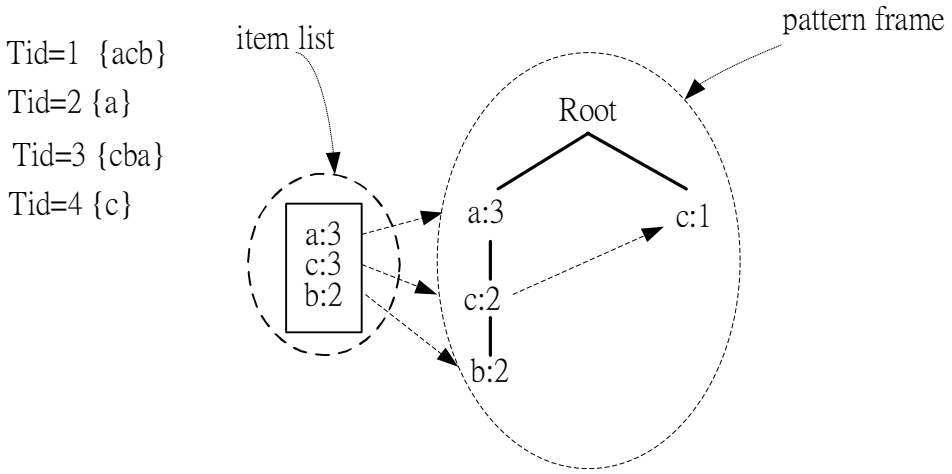


圖 2.1: FP-Growth:FP-tree範例

四個項目集合:{a,c,b}、{a}、{c,b,a}及{c}；按照項目的出現次數排序，並將結果儲存在item list中；再將四個項目集合依照item list的順序，逐一地插入至圖2.1右邊的樹狀結構中，以記錄項目集合與其出現次數。FP-Growth演算法首先掃描資料庫一次，目的是找出所有項目及其各自的出現次數，剔除次數不夠頻繁的項目之後，依照出現次數排序可以得到一個序列；之後當我們掃描資料庫的每一筆交易時，即可根據此序列來建構Frequent-Pattern Tree(FP-tree)。因此該演算法總共只需要掃描資料庫2次即可。完成FP-tree的建構之後，依循item list可以得到個別項目的分支；遞迴地處理與該項目有關的樣式p，若此樣式的支持度大於最小支持度門檻值時，則該樣式即為頻繁樣式。FP-Growth演算法已經被證明是非常有效的方法，尤其適合處理資料集合為密集(dense)的型態。但是當資料集合所含的不同項目太多時，該演算法會因為樹的節點(node)增加，而降低其在節點間走訪的效率。此時可用FP-array這個資料結構來解決這個問題，同時FP-array對於稀疏(sparse)的資料集合也有很好的效果(Grahne and Zhu, 2005)。

2.2 時間性資料探勘

傳統資料探勘是把所有資料當成一個大集合，然而很多類型的資料都是有時間性的，以最普遍的交易資料來說，可以分成每日、每週、每月甚至每年，也就是考慮了時間維度而不再是把資料當成純粹的靜態資料，處理這種包含時間維度的資料探勘即稱為時間性資料探勘(temporal data mining)。隨著時間點的變動，時間性資料探勘的結果可以協助探勘者推論事件的來龍去脈。因此時間性資料探勘不但可幫助我們瞭解在一段時間內的資料變化，甚至可以幫助我們推測出資料變化的原因。關於時間性資料探勘的研究範圍相當廣泛，本節將只針對發現樣式的部份加以闡述。

雖然時間性資料探勘處理的是考量時間因素的資料，但是知識的發現並不一定要使用完全的時間性資料庫(Temporal Database)，在一般靜態資料庫中依然可以進行時間性資料探勘(Roddick and Spiliopoulou, 2002)。為了將資料更明確的劃分，學者Roddick and Spiliopoulou(2002)將時間型態分成以下四類：

1. 靜態(Static):靜態型的資料不包括任何與時間有關的因素，故無法推測事件發生的先後關係。

2. 序列(Sequence):當資料隨著時間先後的次序被儲存在序列資料庫中即稱為序列資料，但是這些先後順序有時並不存在特定的持續時間，即每個事件的間隔時間未必相等。

3. 時間戳記(Timestamped):將靜態資料依照特定的時間區間分成數個特定區段即可形成時間性的資料。

4. 完全時間性資料(Fully temporal):只要是和時間因素有關，會隨著時間變化而波動的數據或觀察結果，如股價或氣溫等；甚至具有多個時間維度的如時、分與秒等。

樣式發現在時間性資料探勘的研究大致上可以分成以下幾個方向。第一個為序列樣式(sequential pattern)探勘:在某些領域中，資料是具有先後順序的，例

如藥品的交互作用、瀏覽網頁的順序等；因此專門處理這種儲存在序列資料庫中，資料元素間具有順序關係的研究即稱為序列樣式探勘。順序是依照時間的先後發生而決定，因此序列樣式探勘也被視為一種時間性資料探勘。序列樣式的概念最早是由Agrawal and Srikant(1995)所提出。相關應用如商品A被購買後，在多久之後也會有興趣購買B。探勘序列樣式包括最早的AprioriAll演算法(Agrawal and Srikant, 1995)及一個更有效率的PrefixSpan演算法(Pei et al., 2001)。第二類為與頻繁樣式有關的時間性資料探勘，包括發現某些週期性出現的樣式(Ozden et al., 1998)及時間曆(calendar)探勘(Li et al., 2002)，其目的是為了發現在使用者所設定的特定日期內出現的樣式；另外還有研究為了避免遺失那些具有高的信心度、低支持度的樣式，因此將探勘的時間限定在特定的項目存續時期內，以探勘出被整體資料量給埋沒的關聯法則(Ale and Rossi, 2000)；而探勘跨交易間樣式的關聯性(Tung et al., 2003)亦屬於此類的時間性資料探勘。第三類則是與趨勢分析相關的研究:此類研究大都處理時間序列資料，因此需先透過離散化及維度精簡的程序，再進行相似度的計算。若相似度很高或者透過視覺化(visualization)技術所呈現的圖形很類似時，則被預期具有相同的趨勢(Keogh et al., 2001)。

2.3 變化探勘

研究動態環境變化相關的議題一直是資料探勘及機器學習(machine learning)的任務之一，為了能根據變化進而調整原本建立的模型，以確保學習能力的品質，而這種能因應變化迅速調整的問題，在機器學習的觀點稱之為concept drift(Webb et al., 2001)。但是機器學習研究的重點在於希望預測的精確度能不受環境變動的影響，而不是在探索究竟發生了什麼變化(Liu et al., 2000)。資料探勘對於發現或偵測動態環境變化的主要處理方式在於比較不同的資料集合或是法則，期望在分析比較過其中的差異之後，能從中瞭解事件的演化趨勢。

2.3.1 Change detection

在資料探勘中發展的模型都是有個特定的目標，如分群(cluster)的模型，其目的是將特性最接近的資料當作一個群體。Ganti et al.(2002)指出根據兩個不同

的資料集合，衡量這些模型所產生結果的差異是一個重要的議題，因此發展了一個架構(framework)衡量模型間的變化。此架構能處理的模型包含資料探勘中最普遍的三類：探勘頻繁項目集合、決策樹分類器及分群；並能獲得標準的量化指標包括misclassification rate和chi-squared metric。該架構由structural component及measure component所組成，structural component代表的是region set的部份。不同的模型中，region有著不同的意義，在分類器中，一個類別是一個region；而在分群中，一群為一個region。為了衡量兩個不同資料集合的structural component，需要透過greatest common refinement(GCR)的處理；measure component包括計算個別region差異的difference function及將個別差異組合成全部差異的aggregate function。故最後可以得到一個量化數值，若此數值愈大，代表兩個模型的差異愈大，反之亦然。除了可以計算整體資料之外，也可以只計算某些特定部份，如某個類別值的差異等。

發現群組的差異也是個很重要的議題，如以職業來說，想要瞭解不同的群組如教師、醫生、工程師和設計師等，他們對於保險的偏好有何不同。為了解決這個問題，Bay and Pazzani(2001)提出對比集(contrast sets)的概念，試圖瞭解不同群組之間的差別。若以G代表群組，則對群組$G_1, G_2, ..., G_n$，需符合$G_i \cap G_j = \emptyset, \forall i \neq j$；若$A_1, A_2, ..., A_k$表示$k$個屬性，則在不同$G$下的每個$A_i$與其可能值$\{V_{i1}, V_{i2}, ..., V_{im}\}$所構成的聯集即稱為對比集。易言之，對比集為在某個特定群組下包含一組至多組屬性-值的集合，譬如：(學歷＝博士)∧(性別＝男)即為一個具有兩組屬性-值的對比集。而對比集必須符合下列兩個條件：

$$\exists ij \;\; P(contrast\ sets = True|G_i) \neq P(contrast\ sets = True|G_j)$$
$$max_{ij}|support(contrast\ sets, G_i) - support(contrast\ sets, G_j)| \geq \delta$$

第一個條件是為了確保利用統計所檢定為顯著的對比集，在不同的G下是真的不同的；另一個為任兩個群組間的的最大差異要符合使用者設定的最小差異門檻值δ，也就是必須具有某種程度上的差異。為了找出所有的對比集，需先建構由屬性-值所組成的搜尋樹如圖2.2，每個節點代表一個對比集，從根節點往下加入，在第一層只有一個屬性，往下一層即增加一個屬性，且每個節點的屬性-值不能

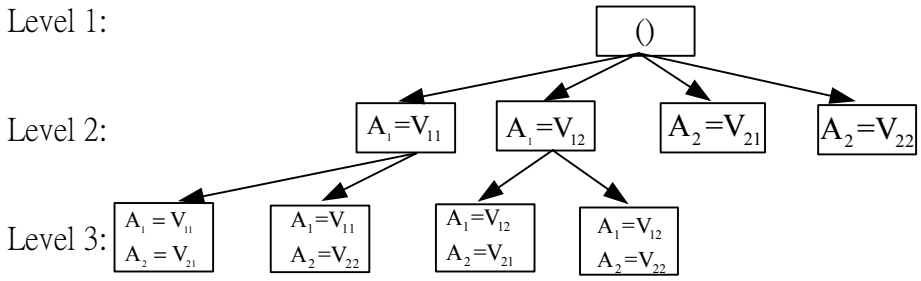


圖 2.2: 兩個屬性且各有兩種可能值的搜尋樹

重複。最後可以建立出候選項目集合，再針對不同的群組進行卡方獨立性檢定 (chi square independence test)，檢定是否有顯著相關；同時也提出一套有效的修剪方法 (pruning)，以減少搜尋空間及增加運算的效率。

2.3.2 Emerging pattern

在不同的資料集合間，若樣式有明顯的改變或者差異，則該樣式可稱為 emerging patterns(EPs)，最早是由 Dong and Li(1999) 所提出的，而 EPs 可定義為某項目集合的支持度在兩個不同的資料集合 (D_i 與 D_{i+1}) 間有顯著的增加；其增加的比例稱為 *Growth Rate*。此研究也對此提出一個衡量的準則，假設 X 為一個項目集合，而 $support_i(X)$ 代表 X 在 D_i 下的支持度，衡量準則 *Growth Rate(X)* 表示如下：

$$Growth\ Rate = (X) \begin{cases} 0, & \text{if } support_i(X) = 0 \text{ and } support_{i+1}(X) = 0 \\ \infty, & \text{if } support_i(X) = 0 \text{ and } support_{i+1}(X) \neq 0 \\ \frac{support_{i+1}(X)}{support_i(X)}, & \text{otherwise} \end{cases}$$

當 *Growth Rate(X)* 大於由使用者自己設定的門檻值時，則 X 即為一個 EPs；若 *Growth Rate(X)* 等於 ∞，則 X 被稱為 jumping emerging patterns(JEPs)。為了探勘出 EPs，Dong and Li(2004) 提出了 border-differential 演算法，採用 border 可以簡捷地表示項目集合，border=$\langle L, R\rangle$，其中 L 是 *left bound*，代表的是最小集合，而 R 是 *right bound*，代表最大集合，L 中的所有元素一定是 R 中某些元素的子集合 (subset)，R 中所有元素一定是部份 L 元素的超集合 (superset)，舉例來說，border$\langle L, R\rangle$= $\langle\{1, 23\}, \{123, 234\}\rangle$ 就可以用來表示 6 個項目集合，即

$\langle\{1, 12, 13, 123, 23, 234\}\rangle$，而用 $\langle\{12\}, \{12345, 12456\}\rangle$ 就可以表示 12 個項目集合；由以上可知，利用 border 就可以表示所有的項目集合。

透過 border 的處理可以發現兩筆資料集合的不同，這個處理程序稱為 border differential。若已知兩個 border 為 $\langle\{\emptyset\}, R_1\rangle$ 及 $\langle\{\emptyset\}, R_2\rangle$，則透過 border differential $\langle\{\emptyset\}, R_1\rangle - \langle\{\emptyset\}, R_2\rangle$ 所得到的 $[L, R]$ 即為兩筆資料集合的差異；必須注意的是 L 必須為 $\emptyset$，如此一定可以滿足 L 的特性，並且不需再花費時間尋找這個部分，只需要探勘 R 即可，此處可以利用 Max-miner 演算法 (Bayardo, 1998) 找出符合門檻值的 *long pattern* 就可以發現 R。利用 border-differential 演算法發現變化樣式的參數設定如表 (2.1) 所示；因此我們可以隨著使用者本身的需求設定不同參數就可以探勘出不同的變化樣式；而後續對於 EPs 的研究包括改善探勘 EPs 效率的演算法 (Bailey et al., 2002)，以及利用 EPs 的特性建立成分類器 (classifier)(Dong et al., 1999) 等。

表 2.1: border 參數設定

JEPs	EPs
$supp_i = 0$	$\theta \le supp_i$
$supp_{i+1} \ge 0$	$\delta \times \theta \le supp_{i+1} \le 1$

θ 爲 $supp_i$ 的 *minsupp*

δ 爲設定的 *Growth Rate*

利用 border-differential 演算法也可以用來探勘出時間趨勢，假設擁有 1999 年至 2002 年共計四年份的資料集合 D_i，若以 E_i 表示 i 年的 EPs，則可探勘出此四年的 EPs= $\langle E_{99,00}, E_{00,01}, E_{01,02}\rangle$，若某 EP 在這三個集合都曾出現，則可以視為趨勢 (Dong and Li, 2004)。然而此方法只會探勘出符合門檻值的樣式，若不符合此條件，將會被認為不存在；而且此方法在不同資料集合會使用不同程度的支持度作為門檻值，如此處理並不公平；另外在探勘多期資料時，此方法只是比對前後兩期的資料。因此利用 border-differential 演算法探勘多期資料集合時很可能會遺失某些樣式，使得最後得到的資訊不甚完整。

2.3.3 Exception rules

大部份的探勘法則研究，多是致力於發現特定領域有意義的知識，而這當中也包括了被大家認為理所當然或者是既有印象的知識。舉例而言，商家在週末的生意通常較好，但是對於學區而言，學生在週末大都回家了，所以生意反而較差，這就是一種例外的事件；而在資料探勘中處理這問題的即是探勘例外法則(exception rules)。

例外是一種相對於一般認知的事件，而這種事件常常被忽略，探勘例外法則可以挑戰現存的知識之外，同時也發現了一個新的方向，若能察覺這種差異，便可以進一步改善決策品質(Suzuki and Zytkow, 2005)。探勘例外法則可以分成直接式(directed)與間接式(undirected)，直接式的方法必須先存在某些認知(譬如專家的領域知識)；而這類的方法即會根據這些認知以找尋違背認知的案例，這種與預期不符合的事件即可當作例外。Padmanabhan and Tuzhilin(1999)提出使用者須事先定義認知的方法，令AX為$A \wedge X$，假設定義一個認知為$X \rightarrow Y$，若經過支持度及信心度的計算求得一個法則為$A, X \Rightarrow B$。現假設B是Y的反義，故可推論例外法則$A \rightarrow B$存在。舉例來說，認知為*Doctor→shopping in weekend*，然後發現法則為*Doctor∧ February→shooping in weekday*，就可以推論*February→shopping in weekday*的存在。其它相關研究還包括使用模糊比對(fuzzy matching)並且以有趣度(interestingness)為衡量，最後產生相似度不夠高的法則即當作例外(Liu et al., 1999)。

相對而言，間接式的方法即不需事先知道某些認知的存在，且在處理的過程中，間接式的方法會同時發現預期與例外的法則，因此這兩個法則是成對出現的，其稱為rule pair，而rule pair中出現機率較高的法則將被認為是預期法則，另一個則為例外法則(Suzuki and Zytkow, 2005)。

2.3.4 Same attribute and same cut

Liu et al.(2000)使用決策樹(decesion tree)(Quinlan, 1992)的方法探勘資料集合的變化。決策樹透過Gain ratio的計算從根(root)節點向下分支，因此愈往下，

資料愈趨近一致，故利用決策樹探勘能發現資料在分割(partition)及error rate的變化情形。Liu et al.(2000)所採用的方法稱為same attribute and same cut，此作法的概念是將兩個不同的資料集合透過相同結構的決策樹以找出其中的差異。假設，現有兩期資料集合*t*及*t+1*，首先利用第*t*期的資料建構成一棵決策樹，如圖2.3所示，再將第*t+1*期的資料放進以第*t*期為基礎所建立成的決策樹結構(如圖2.3)形成圖2.4的決策樹，我們可以發現圖2.3及圖2.4是一個擁有相同結構，但是不同資料量的決策樹。之後再以第*t+1*期的資料為基礎建成另一棵決策樹，再加入第*t*期的資料。這種作法就是各自建立自己的決策樹，然後加入另一期的資料集合，如此就可以利用相同的結構，但是不同的資料特性，以計算出其中的差異。

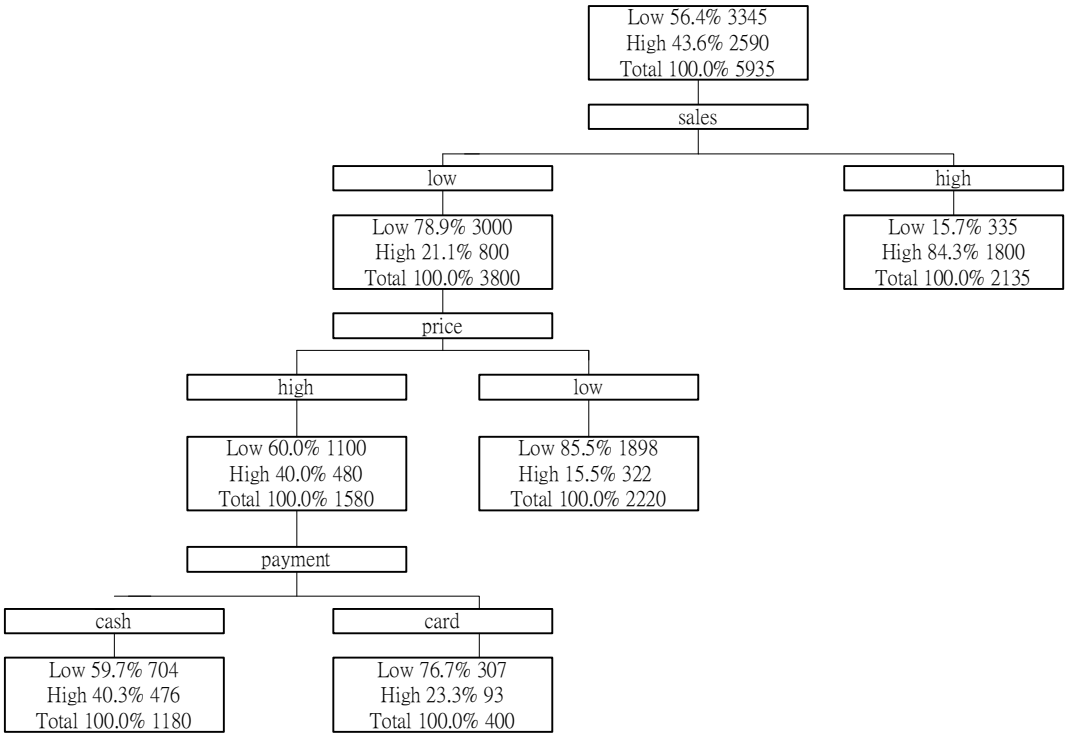


圖 2.3: 第*t*期的決策樹

為了探測出更具體的變化情況，Kim et al.(2005)也採用了same attribute and same cut的方法，因為此方法被證明能探勘出完全的變化，另外為了衡量變化的程度，他們也提出了兩種量化的方法，可以幫助探勘者瞭解變化的強度，為了要

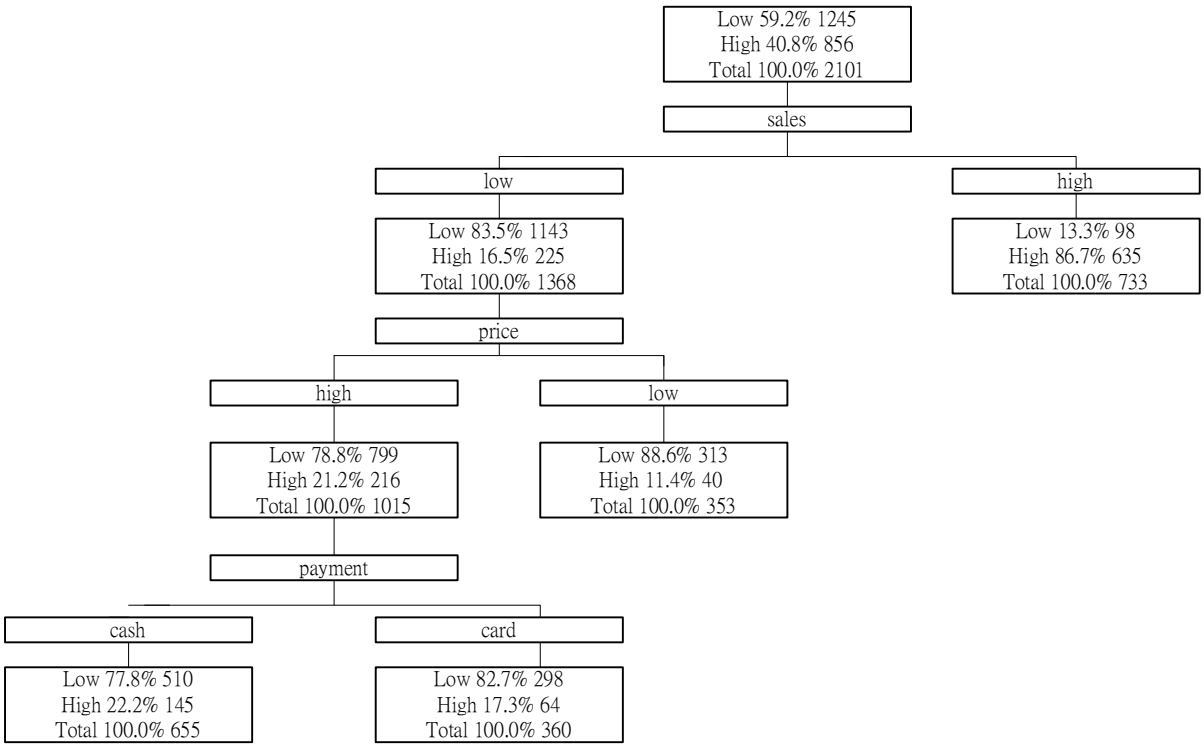


圖 2.4: 第 *t+1* 期的決策樹

進一步地確認變化的類型，該研究除了整合了先前探討變化的型態之外，又另外提出兩種新的變化，總共包含了五種變化型態，其說明如下：

1. emerging patterns: 法則的 LHS 及 RHS 具有相同的項目集合，但是在兩個不同的資料集合中，其支持度明顯不同。

2. unexpected consequent：在兩個不同的資料集合中，法則的 RHS 相同，但 LHS 不同。

3. unexpected condition：在兩個不同的資料集合中，法則的 LHS 相同，但 RHS 不同。

4. perished rules：前一個資料集合中曾出現過的法則在後一個資料集合中卻不存在。

5. added rules：後一個資料集合中出現的法則在前一個資料集合中卻不存在。

此研究偵測變化的程序為：第一步時輸入資料集合D_i, D_{i+1}經過運算之後可以得到決策法則R_i, R_{i+1}；第二步驟輸入R_i, R_{i+1}透過*similarity measure*及*difference measure*計算相似度與相異度，這兩個衡量方法皆是採用逐一比對每個屬性及其值，經計算之後可以求得一個具體的數值，再以此數值與由使用者自訂的門檻值RMT(rule matching threshold)比較後，即可以得到五種型態的變化，其關係如圖2.5(其中的Unexpected Change包括unexpected consequent與unexpected condition)所示，最後根據相對支持度的大小排序出變化的強度。

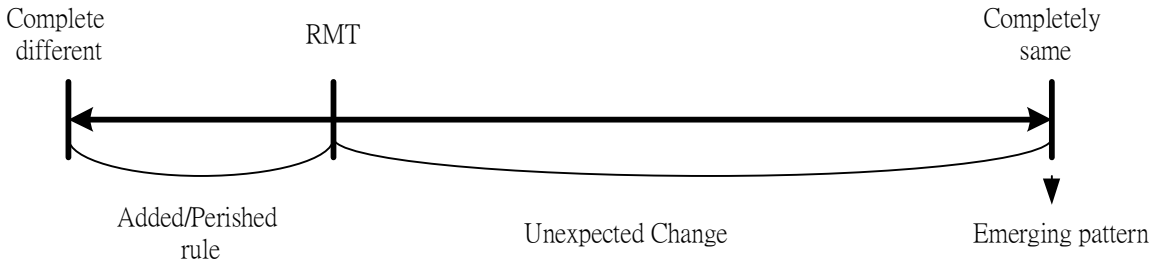


圖 2.5: 依RMT判斷不同型態的變化

此研究最大的貢獻在於不但可以發現變化的型態，而且可以計算出變化的程度，這都可以幫助探勘者更瞭解事件的變化；然而此研究仍存在兩個限制，第一：每個資料集合會依其特性的不同長成不同的決策樹(這裡的不同指的是每個節點包含多少個子節點及何時該分支)，而same attribute and same cut強迫不同的資料集合要在相同的決策樹下比較，而這個條件意指不同的資料集合必需用相同的結構，然而，若我們具有多個資料集合，則如何選擇相同結構的決策樹將成為一個問題。因此，這個方法的限制是不知如何探勘多期的資料集合；第二：採取屬性逐一比對的方式將缺乏效率，尤其當資料期數愈多時，這個情況會更明顯。

2.4 小結

本章主要是回顧過去資料探勘中的相關研究，主要包括三個部份：首先，2.1節是探勘頻繁樣式的部份，說明了探勘頻繁樣式的目的及相關應用，另外也整理出目前較具代表性的演算法，其中由Han et al. (2004)提出的FP-growth演算

法在本研究中特別重要。2.2 節則是時間性資料探勘，在傳統資料探勘上加入了時間性因素的討論，包括具有時間性的資料型態及一些時間性資料勘的研究方向。最後，2.3 節中回顧了探勘資料間之差異與變化的變化探勘研究，從這部份的文獻回顧，我們發現目前為止的相關研究大都是探討兩期資料的變化，對於多期資料則少有著墨。

第三章

探勘樣式的變化

本章首先在3.1節中定義本研究欲尋找的樣式變化。3.2節則是描述處理兩期資料與多期資料的不同點，及探勘多期資料的相關問題，並解釋先前的研究無法處理本問題的原因。另外，我們也將詳細說明本研究採用以「變化的幅度」決定樣式取代過去先考慮樣式是否頻繁再計算變化程度的兩階段模式之理由；接著在3.3節介紹一個在搜尋樹架構下連續探勘多期樣式變化的MCP(mining the changes of patterns)演算法及其計算變化程度的衡量準則。MCP演算法包含三個主要階段，我們將於3.3.1節至3.3.3節中作詳細的說明。為了增加探勘樣式變化的彈性，將於3.4節中介紹一個作法以達成此目的。3.5節則會以一個簡單的例子實際說明演算的程序。

3.1 問題定義

若$I = \{i_1, i_2, ..., i_m\}$是一個由項目$i$所構成的集合；而交易資料庫$TD = \{T_1, T_2, ..., T_n\}$，則任一個交易$T_j \subseteq I$；當存在$P \subseteq I$，且$P \subseteq T$時，則$P$被視為一個樣式或項目集合。本研究的目的在於探勘樣式的長期變化情況，而樣式的變化計算方式如下：

$$\delta_i^t = \begin{cases} \frac{occurrence(P_i^t)}{occurrence(P_i^{t-1})}, & \text{if } occurrence(P_i^{t-1}) > 0 \\ M, & \text{if } occurrence(P_i^{t-1}) = 0 \end{cases}$$

假設樣式P_i在第*t-1*期曾經出現過(即$occurrence(P_i^{t-1}) > 0$)，而$\delta_i^t$代表$P_i$的出現次數在第*t-1*期到第$t$期間相除所得到的比率，並稱之為變化率；當$P_i$沒有出現在

第 *t-1* 期，卻有出現在第 *t* 期時，我們以一個大數 M 代表其變化率。從變化率程度的高低即可觀察出樣式之出現次數的變化情況。

定義 1. 若給定 n 期資料，當某樣式從任何一期直到第 n 期，其變化率皆符合使用者設定的門檻值，即認為此樣式發生變化。

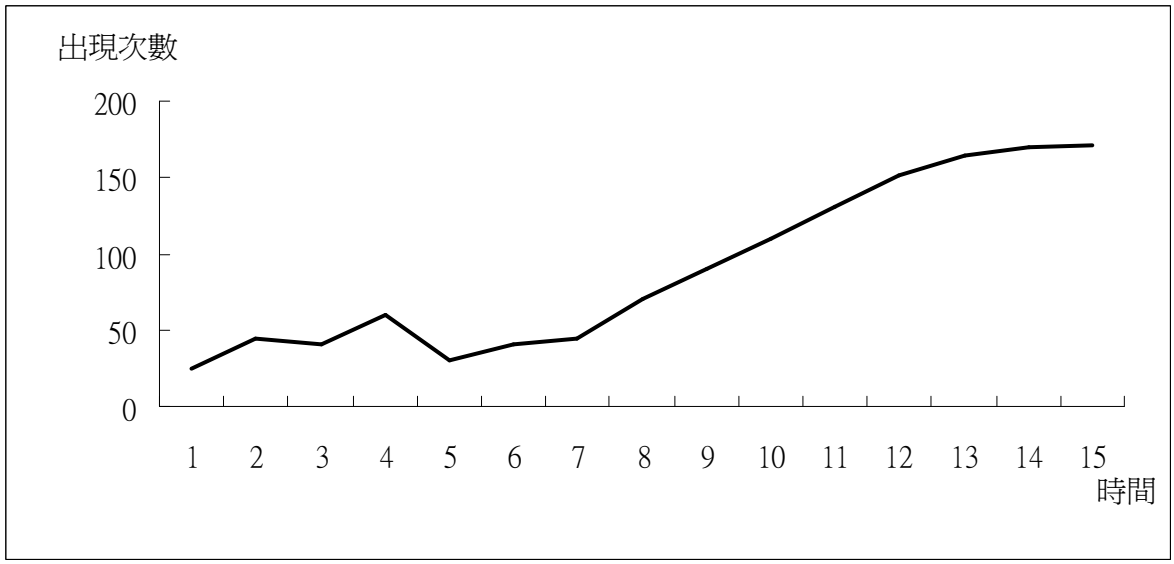


圖 3.1: 樣式的變化

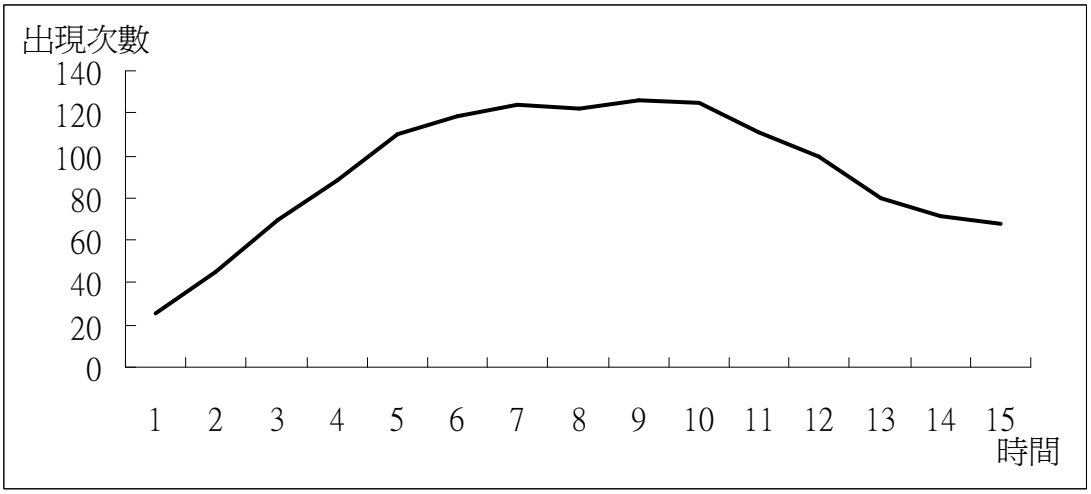


圖 3.2: 變化樣式與頻繁樣式的概念圖

由於我們想探勘那些直至目前仍持續發生變化的樣式，因此乃從後面的時間點往前面的時間點進行探勘。如此即可探勘出所有由任一期開始直到目前時間

點仍存在變化的樣式。探勘的順序由後往前開始的另一個理由是某些樣式很可能只在前面數個時期有顯著的變化，之後此變化就消失了；像這類的情況很可能只是偶然的變化，而且長期下來這種情況將會非常多，故較缺乏參考價值。如圖3.1中可發現樣式的出現次數在時間點1到2是成長的，但是時間點2到3卻是下降，而之後一段時間也是忽而上升忽而下降。因此，我們無法從前面7期的時間觀察出此樣式的變化趨勢。然而，此樣式從時間點7一直到最後的11卻一直呈現上升的情形，這說明了此樣式的發生頻率愈來愈高，而重要性也明顯增加；這類的變化相當具有參考價值，也是我們期望找出的變化。我們想要探勘的資料是接近目前時間點且符合我們期望的變化資訊。因此，若某些樣式在後面的時間點已不符合需求，則該樣式在前面時間點的變化即不需再費時探勘，如此即可省下許多時間，並可提升執行的速度。因此，為了避免浪費時間去探勘偶然或不規則變化的區段，我們將由後面的時間點往前探勘。

為了進一步了解本研究所想要探勘出的樣式，我們將以圖3.2說明之。一個樣式可以代表一個產品組合或是網頁的瀏覽順序等；他們就像產品生命週期般，會有所謂的成長期、成熟期及衰退期等波動。本研究主要目的即在於探勘出如圖3.2中的第一到第五期的樣式變化。相反地，探勘圖3.2中y軸出現次數在某種程度以上的樣式(也就是第六到第十期的部份)即為頻繁樣式的探勘。一般來說，不頻繁樣式在探勘頻繁樣式時會被忽略，然而，本研究會將那些長期下來會呈現某種變化趨勢的樣式全部找出來。因此，當時間因素被列入考慮之後，即使樣式不夠頻繁，其變化的程度亦可被本研究所提出的方法挖掘出來。

雖然本研究主要探勘的是具有長期成長趨勢的樣式變化，但根據使用者的不同需求，我們對於變化的認定是相當具有彈性的。廣意來說，變化可包含下列三種類型：

1. 成長：將變化率的門檻值設定大於1，也就是樣式的後一期出現次數都大於前一期；代表此樣式在資料庫所佔的比例愈來愈高，呈現的是成長趨勢，如圖3.3(a)所示。

2. 衰退：將變化率的門檻值設定小於1則可探勘逐漸衰退的樣式，也就是長期下來找出的樣式其出現次數有愈來愈少的趨勢，如圖3.3(b)所示。

3. 不規則變化：若是不設定門檻值，也就是將變化率設為0，則可以發現樣式的所有變動情況，其中包括呈現不規則變化的樣式如3.3(c)所示；但是這類的樣式因為較難預期與評估其日後的變化情況，故參考價值較低且所需花費探勘的時間也最多。

由圖3.3可以發現我們所提出的方法是相當具有彈性的，能夠根據使用者的需要，得以探勘出不同類型的變化，如成長或衰退，甚至可以根據斜率的大小找出不同趨勢的變化；但是必須注意的是，本研究無法同時找出三種類型的變化。

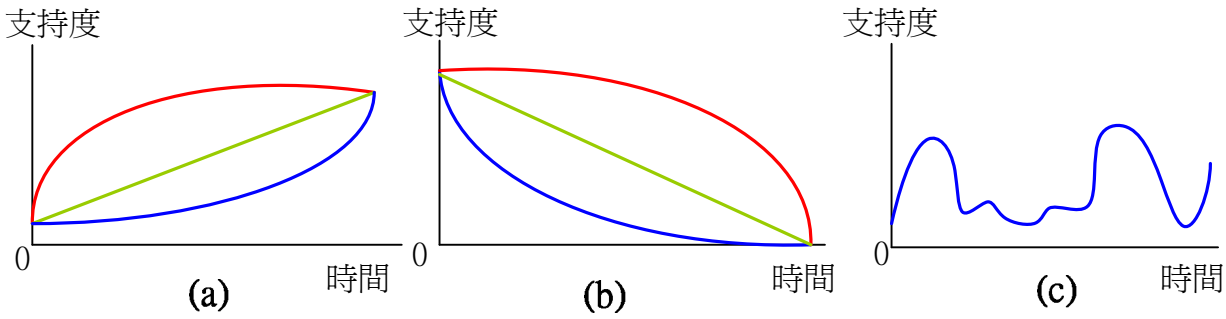


圖 3.3: 樣式的三種變化

3.2 相關議題

在過去探勘兩期資料的相關研究中探勘「變化」的意義在於了解其資料差異程度，也就是以兩個資料集合所產生的樣式或法則做比對，經計算後所產生的比率即是差異的幅度，其概念如圖3.4所示：

在圖3.4中，P代表樣式的集合，i代表第i期的資料；在圖3.4的第二層代表經過比對後，所存在的變化樣式。由此可知，處理兩期資料僅需將該筆資料集合加以比對即可。需而，此種比對方式，在處理多期資料時將變得很複雜。雖然相關研究並沒有提到如何處理多期資料的方式，我們仍試著依其處理兩期資料的方式延伸至多期，並探討其可行性。same attribute same cut(Kim et al., 2005) 探勘

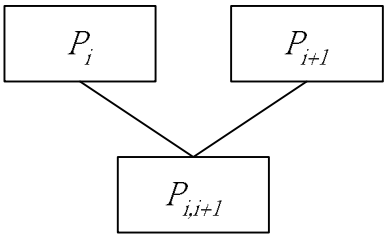


圖 3.4: 兩期資料的處理

資料集合時，要分別以一期的資料為基礎建立決策樹，再加入另一期的資料進行比對。以兩期而言，先建立第i期的決策樹D_i，再加入第$i+1$期的資料進行比對，之後再以第$i+1$期的資料為基礎建立D_{i+1}並加入第i期的資料進行另一次的比對。在探勘多期資料時(以三期為例)，在處理完先前的兩期(第i期及第$i+1$)期資料後，需把第$i+2$期的資料分別加入至D_i及D_{i+1}，如此會產生一個決策法則的集合，而這些法則需與之前兩期比對後的法則再進行一次比對(兩期需兩次，三期需三次，依此類推)。除此之外，還需以第$i+2$期為基礎建立新的決策樹D_{i+2}，而之前的第i與第$i+1$期的資料還需加入到D_{i+2}的結構裡，如此會產生三期的決策法則集合，且又需再進行另一次的比對。因此每多探勘一期資料集合，比對次數會比上一期的比對次數再多一次。而加入資料到不同決策樹的次數，為每多一期即需多增加兩次。因此，我們可以發現當探勘的期數愈多時，所需比對的次數及加入資料至不同的決策樹所需的次數與儲存空間也將愈多(n期即需要儲存n個決策樹)。

border-differential algorithm(Dong and Li, 2004)在不同期的資料集合下需採用不一樣的最小支持度門檻值以過濾樣式，然而這種作法會導致在不公平的條件下探勘樣式；舉例來說，假設將成長率設為兩倍，則我們應該找尋滿足成長率至少兩倍之所有可能樣式；但是若將border-differential方法的第一期的支持度設為0%，則會造成0乘上任何數都仍然是0的情況，故只能將第一期的支持度設定為一個極小的數，像是1%；所以第二期會在支持度為2%下搜尋，至於2%以下的樣式則完全被忽略，第三期則是在4%下搜尋樣式；我們可以發現每一期的資料都是在不同的門檻值下進行探勘，這將會造成資訊的遺失。舉例來說，第二期中

被忽略的樣式有可能會在第三期中顯著成長，然而該樣式卻可能在探勘完第二期時會因為不夠頻繁而被刪除。如此一來，該變化樣式將無法在第三期被發現。這種情況尤其可能當資料期數一增加時，將變得愈加嚴重。以border-differential algorithm探勘多期資料的變化有兩種可能會造成資訊遺失的情況：首先，此法仍然採用探勘頻繁樣式的做法，先將不同資料集合在設定的支持度下探勘，之後再找出變化的樣式。再者，由於此法本身即不是一個探勘多期的演算法，它只考慮到兩期的資料，無法提供某個樣式在全部資料期間的變化情形。

若要以border-differential algorithm探勘長期趨勢，除了第一期外，其他期的資料都須在不同支持度下探勘多次；若第一期的資料在$a\%$下探勘，而成長率設為g的話，則第k期的資料就必須在$ag^{k-1}\%$下探勘，才能找到那些自第一期開始保持每期成長率至少為g的資料。然而，為了也要探勘出自第k期(k=2,3,....,i-1)以來的那些持續保持成長率為g的所有資料，我們針對第二期的資料就必須重複探勘了i-1次(亦即，以成長率為分別為$ag^{k-1}\%$，k=1,2,...,*i-1*來探勘)。由於Dong et al.(2004)並沒有說明該如何處理兩期以上的資料，若以處理兩期資料的作法為基礎，需要逐層比對每兩期為一個單位所產生的樣式以探勘多期資料的變化樣式。如此一來，每一期的資料皆必須被重複探勘與比對；由此可知，這樣的做法是相當地缺乏效率。

3.3 MCP演算法

從本節開始，我們將說明本研究所提出的MCP演算法。MCP演算法包括三個階段：(1)建立原始搜尋樹(Constructing Original Search Tree, COST)：以作為探勘樣式基礎的儲存結構、(2)探勘候選樣式及建立樣式樹(Mining Candidate Patterns and Constructing Pattern Trees, MCPCPT)：以尋找樣式，(3)更新資料與調整樣式樹(Updating Data and Adjusting Pattern Trees, UDAPT)。我們也會介紹用來作為多期探勘基礎的樹狀資料結構:樣式森林。由於MCP演算法是在樹狀結構上發現樣式變化的方法，故我們需要多次進行樹的走訪以發現樣式的變化。綜合上述，MCP演算法所包括的程序顯示在Algorithm MCP中。

Algorithm MCP
Input: n-period datasets
Output: the changes of patterns
1 constructing Original Search Tree (COST)
2 mining candidate patterns and constructing Pattern Trees (MCPCPT)
3 traversing Pattern Trees to find the changes of patterns (Traversal)
4 **for** each period n-2→ 1 **do**
4 updating data and adjusting Pattern Trees (UDAPT)
5 traversing Pattern Trees to find the changes of patterns (Traversal)

3.3.1 建立原始搜尋樹架構

演算法首先須建立一個搜尋樹狀結構，用來儲存資料集合中的所有項目；本研究所採用的結構是修改自Han et al.(2004)所提出的FP-tree，它已被認為是能有效減少儲存空間的一種結構(Grahne and Zhu, 2005)。由於本研究目的是找出所有從任一期直到最後一期皆呈現顯著變化的樣式，所以演算法的探勘順序會從最後一期往前進行，以省下探勘那些中途即無法顯著變化的樣式所花費的時間。同時，在資料的存取上，我們改良了原本僅用來儲存單一資料集合的FP-tree，以建立可儲存連續兩期資料集合之搜尋樹作為探勘候選變化樣式的資料來源及避免對資料庫進行重複的存取。首先，我們先用建構FP-tree的方式來建立以第n期資料集合為基礎的搜尋樹，其中包含項目名稱及其出現次數，而且相同項目的節點會相連以增加走訪的速度，如圖3.5所示，我們以具有箭頭的虛線表示連結的關係。針對第*n-1*期的資料，我們不須特別為其建立另一棵新的搜尋樹，而是直接將之加入由第n期為基礎所建成的搜尋樹裡，每個項目節點增加另一個欄位以儲存第*n-1*期資料的項目出現次數，如圖3.6所示(節點的左右欄位分別儲存了第n期、第*n-1*期資料的出現次數)。從圖3.6中可以觀察出更新後的搜尋樹一開始先保留了原有的分支，包括項目名稱及其出現次數；之後，當逐一加入第*n-1*期資料中的交易時，若發現原始搜尋樹中存在與交易相同的分支(亦即具有相同的項目集合)，則需更新每個對應的項目節點中儲存第*n-1*期欄位的出現次數；而當原始搜尋樹不存在相對應於交易的分支時，則需產生新的分支以儲存此交易，如圖3.6中的{c,e}及{a,d,b}，在第n期並未曾出現，但在第*n-1*期曾出現，則這些相關節點在第n期的出現次數將會被設為零；同樣地，若第n期的資料集合中具有

的分支，在第 *n-1*期卻沒有出現，則該節點的出現次數亦將被設為零，如圖 3.6的{d,b}。

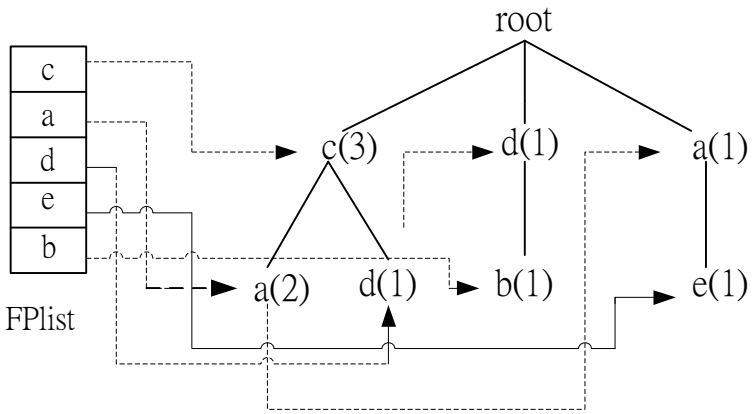


圖 3.5: 只有第 *n*期資料的搜尋樹

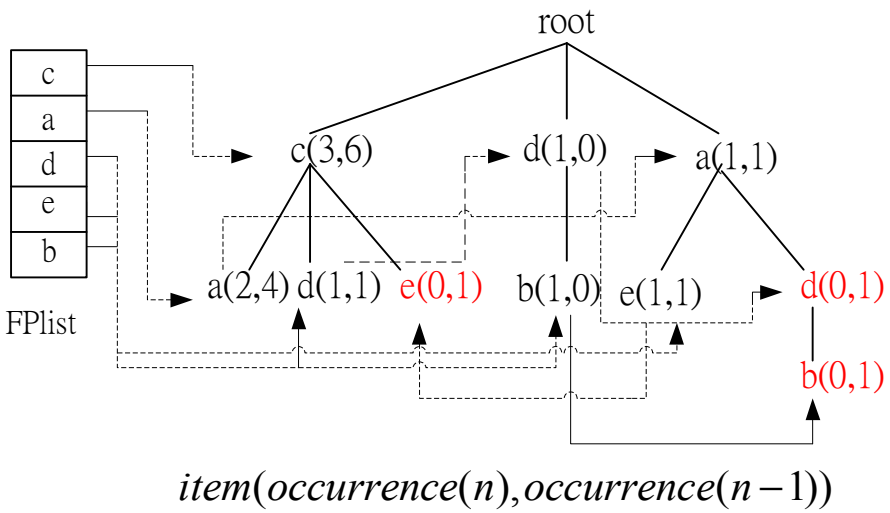


圖 3.6: 更新後的搜尋樹

建立原始搜尋樹的程序顯示在Procedure *COST*中，先輸入第 *n*期資料集合，再輸入第 *n-1*期資料集合；在第 2 行的部份判斷是否為第 *n*期的資料集合，若是的話，則於第 3 行進行第一次掃描資料集合 *n*，以記錄所有項目的出現次數。第 4 行依照每個項目的出現次數由多到少進行排序，最後將結果儲存在FPlist 中；故資料加入時便能夠按照項目出現次數的多寡，將出現愈多次的項目將被放在愈上層的節點，如此便可將相同的項目集合加入到同一個分支裡，以有效地減少儲存空

```
Procedure COST
Task: constructing Original Search Tree
1     for each Dataset,DS in {n,n-1} do
2         if DS is n
3             scan Dataset,save item name and count into FPlist
4             sort the item of FPlist in descending order of count
5             create the root of an original search tree,T
6             call InsertingData
7         else
8             call InsertingData
```

```
Sub Procedure InsertingData
Task: inserting data into Original Search Tree
1 for each transaction in DS do
2     sort each item in the transation according to the order of the FPlist
3     if there exist no branch corresponding to transaction
4         create node
5         node.name=item name
6         node.parent=parent
7         parent=node
8         if DS is n
9             node.Lcount=1 //Lcount save the occurrence of item of Dataset n
10        else
11            node.Rcount=1 //Rcount save the occurrence of item of Dataset
                                                                    n-1
12        if FPlist[item]→next is not NULL
13            node→next=FPlist[item]→next
14        node=FPlist[item]→next
15    else
16        if DS is n
17            node.Lcount++
18        else
19            node.Rcount++
21 return
```

間的使用。值得注意的是，第*n-1*期資料集合不會另外建立一棵新的搜尋樹，而是直接將資料內容加入到原來的那一棵搜尋樹裡，故不需掃描第*n-1*期資料集合以記錄各項目的出現次數，而是依照原本第*n-1*期資料集合的項目之頻繁順序加入至搜尋樹中。因此，第*n-1*期資料集合不用再次被掃描及排序。第5行則是建立原始搜尋樹的初始化動作。第6行呼叫InsertData，目的是將兩期的資料集合都加入到同一棵原始搜尋樹裡。Sub Procedure *InsertingData*的第1行將逐一掃描第*n*期資料集合中的每一筆交易；第2行則將該筆資料中的項目按照出現次數的多

寡依序加入搜尋樹；第3行會先判斷在搜尋樹中的任何一個分支(branch)是否已存在與該筆交易相符合的項目。要注意的是，此處指的相同項目指的是該筆交易的項目順序與分支的順序相同的那些項目；倘若項目順序不符合的話，第4至6行的部份將產生新的節點，並且設定其相關資訊。第7行須判斷目前為哪一個資料集合，若是第n期則將該節點出現次數的左欄位設為1，反之(即第*n-1*期)則將右欄位設為1。第12到15行乃為了連結搜尋樹中所有相同項目的節點，如此能加速搜尋樹的走訪。故第12行會先找到FPlist中的項目，並且在第13至15行找到此項目在搜尋樹中相連的第一個節點的位址。若此項目不存在於搜尋樹中，則演算法將為此項目產生一個新的節點，並直接連結此新節點與FPlist中的相同項目；反之則將原本與FPlist相連的節點與新節點相連，同時再連接新節點與FPlist的項目。故FPlist所連結的節點一定都是最新產生的節點。第16行即是代表搜尋樹的分支中已經存在與交易相同的項目，這時只要將該項目的出現次數加一即可。當呼叫完兩次Sub Procedure *InsertingData*之後，即可建立原始搜尋樹，也就是如圖3.6所示。

分析**1.** 一般建立FP-tree需掃描資料集合兩次，第一次是為了計算各項目在資料集合中的出現次數，第二次則是讀取各筆交易並將之加入至搜尋樹中。在本研究中，建立原始搜尋樹也需要掃描兩次第n期的資料集合，但只需掃描一次第*n-1*期的資料集合。這是因為我們只建立一棵搜尋樹，而第*n-1*期的資料集合加入至原始搜尋樹只需按照原本第n期資料集合的項目次序即可，所以只需要掃描第*n-1*期資料集合中的每一筆交易。因此，將兩個不同期間的資料集合以同一棵搜尋樹表示，將可以少做一次掃描，並且省下一次將項目依其出現次數加以排序所花費的時間。若m為某筆交易中的項目數，而k為搜尋樹中每一層可能的節點個數則插入一筆交易至原始搜尋樹需要費時$O(m\times k)$。

若假設$freq(m)$為交易中的頻繁項目個數，則Han et al.(2004)指出加入交易到搜尋樹的時間成本為$O(freq(m))$。然而，這個時間成本似乎沒有考量需要花費時間去尋找搜尋樹中每一層的項目。因此，若搜尋樹的每一層有k個節點，則加入一個交易到搜尋樹的時間成本應為$O(freq(m)\times k)$。

3.3.2 挑選候選樣式及建立樣式森林

探勘樣式的變化首先必須找出資料集合中所有可能的候選樣式。為了避免挑選出太多重複的候選樣式及減少儲存空間，當某項目集合的出現次數與其子集合相同時，該子集合之出現次數即可從其原母集合推導而得，因此將不被視為候選樣式。舉例來說，假設{a,b,c,d,e}、{a,b,c}及{b,d,e}皆為出現次數為10次之項目集合，則我們僅將{a,b,c,d,e}視為候選樣式。當候選樣式{a,b,c,d,e}的變化率符合門檻值時，即可稱為變化樣式。

演算程序首先將原始搜尋樹中具有某一項目i的相關分支稱為項目之原始集合(original set, OS)。挑選項目i之候選樣式的程序必須從原始搜尋樹中依照項目出現次數由少到多的順序，也就是由最底層的葉部節點開始逐一走訪所有具該項目的分支。如圖3.6中所有和項目b有關的分支為{b,d}及{b,d,a}，這些項目集合即為原始集合。但是資料裡包含的項目集合卻不一定只包括原始集合，而是可能也包括某項目集合的子集合，或是原先沒發現的項目集合等狀況。舉例來說，假設有{a,b,c,g,h}與{g,h,i,j}兩個原始集合，則這兩個原始集合可以發現另一可能的候選樣式{g,h}，因為用這種方法所產生的候選樣式，其出現次數至少大於或等於其原來的原始集合。而我們將這類透過原始集合的交集所發現的項目集合稱之為衍生集合(derived set, DS)。根據定理1，我們可確認只要找出原始集合即可以探勘出所有可能成為候選樣式的項目集合。

定理 1. For a given original set $OS = \{s_1, s_2, ..., s_n\}$, where s_i represents the ith itemset, let $DS = \cup_{i,j}(s_i \cap s_j)$ represent the derived set. Let $CS = \{s : s \subseteq s_i$, support($s$)≥support($s_i$) for each $i = 1, ..., n\}$ represent the candidate set obtained by our algorithm. Then, (a) $OS \cup DS \subseteq CS$ and (b) $CS \subseteq OS \cup DS$, or equivalently speaking, $CS = OS \cup DS$.

證明： (a)For each itemset $\widehat{s} \in OS$, we know $\widehat{s} = s_k$ for some k and *support($\widehat{s}$)≥support(s_i)* for each $i = 1, ..., n$. Thus $\widehat{s} \in CS$. Similarly, for each itemset $\widehat{s} \in DS$ satisfying $\widehat{s} \subseteq s_i$, *support($\widehat{s}$)≥support($s_i$)* for each $i = 1, ..., n$, and thus $\widehat{s} \in CS$. Therefore,

$OS \cup DS \subseteq CS$.

(b) For each itemset $\widehat{s} \in CS$, there are two cases:

Case 1: if $\widehat{s} = s_k$ for some k, then it is trivial that $\widehat{s} \in OS$.

Case 2: if $\widehat{s} \subset s_k$ for some k, we know by definition that *support($\widehat{s}$)>support(s_k)*.

(i) if $s_k \cap s_i = \emptyset$ for each $i \neq k$, then *support($\widehat{s}$)=support(s_k)*, and we have a contradiction.

(ii) if $s_k \cap s_i \neq \emptyset$ for some i, there are two cases:

(ii.1) if $\widehat{s} \not\subseteq s_k \cap s_i$ for any of such i, then *support($\widehat{s}$)=support(s_k)*, and we have a contradiction.

(ii.2) if $\widehat{s} \subseteq s_k \cap s_i$ for some i, then $\widehat{s} \subseteq DS$.

Thus each itemset $\widehat{s} \in CS$ implies $\widehat{s} \in OS$ or $\widehat{s} \subseteq DS$, which means $\widehat{s} \subseteq OS \cup DS$, and thus $CS \subseteq OS \cup DS$ □

根據定理1，本研究提出一個可以發現所有候選樣式的方法，顯示在Procedure *MCPCPT*中。從原始搜尋樹中探勘出所有與某個特定項目有關的所有項目集合之後，將之儲存成一個序列並同時建立此特定項目的樣式樹。此後依照存進序列的順序將項目集合逐一加入至樣式樹中，同時此項目集合需與之前的每一個項目集合進行兩兩交集比對以確認是否有衍生集合的產生，以上步驟均列於Procedure *MCPCPT*中的2到6行，第7及第8行將衍生集合逐一加入至樣式樹中，而第9行到第15行則判斷衍生集合加入分支時該具有的出現次數。

當樣式樹已經存在與衍生集合相同的分支時，只要更新該分支的出現次數即可。然而，更新支持度是有條件的，若此分支是由目前處理的項目集合所建立，則代表是一個新的分支，之前的項目集合並沒有被考慮，故由此項目集合所產生的所有相關衍生集合的出現次數將等於被比較項目集合的出現次數，且都要去更新此分支的出現次數。當衍生集合所加入的分支是由之前項目集合所產生時，代表此分支之前已被更新過了，故此分支在這次的比對中，就算有多個項目集合與之相同，仍然只能被更新一次；而衍生集合的出現次數應等於目前所處理的項目集合之出現次數。以上所討論的皆是項目集合為原始集合的處理方式。另

一個情況是樣式樹中不存在與衍生集合相同的分支，則表示此衍生集合所代表的項目集合在之前並沒有被考慮，故要為此項目集合建立分支，並加入至序列中與所有的項目集合進行交集比對。新產生出來的衍生集合的出現次數皆等於被比較項目集合的出現次數，因為此項目集合是由透過原始集合所產生的，故本身並沒有記錄出現次數。

例子**1.** 序列L儲存由原始搜尋樹中探勘出的四個項目集合與兩期的出現次數，而此四個項目集合皆是與項目a相關，以{項目集合}:(後一期出現次數，前一期出現次數)的方式表示，其分別是：1.{a,b,c,d,e,f}:(4,2)、2.{a,b,c,d}:(3,1)、3.{a,c,e,f}:(2,2)與4.{a,d,f}:(3,5)。首先建立項目a的樣式樹且把a設為根節點，並長出由第一個項目集合{b,c,d,e,f}所構成的分支，而第一筆資料不需進行交集的比對。加入第二個項目集合{b,c,d}之後，發現樣式樹中已存在相同的分支{b,c,d}了，故只需更新出現次數即可。這時第二筆項目集合需與之前的項目集合進行交集比對，得到衍生集合{b,c,d}，並且將出現次數設為(4,2)。而將此衍生集合加入到樣式樹時，發現已存在相同的分支，並且此分支是由第一個項目集合所產生的舊分支，同時此分支的出現次數已經由第二個項目集合所更新，故衍生集合不需再重複更新此分支。第三個項目集合{c,e,f}會產生一個新的分支，進行交集比對後，發現與第一個項目集合產生一衍生集合{c,e,f}其出現次數為(4,2)，與第二個項目集合的衍生集合為{c}其出現次數為(3,1)。而這兩個衍生集合加入至樣式樹後，發現對應的分支為新產生的，故都該更新此分支。為第四個項目集合{d,f}建立分支及進行交集計算後，將得到衍生集合{d,f}與{f}。將{f}加入到樣式樹後，可知樣式樹中不存在對應的分支，故應為此衍生集合產生新的分支，同時把它加入到L裡。{f}與前面的項目集合交集計算後，得知與第一及第三個項目集合有交集，故產生兩個衍生集合分別為{f}:(4,2)與{f}:(2,2)，其出現次數皆來自於被比較的項目集合。最後再將這兩個衍生集合加入至樣式樹中，即可建立完成項目a的樣式樹及找出相關的候選樣式。

```
Procedure MCPCPT
Task: mining candidate patterns and constructing Pattern Trees
1 for each set i in the OS
2       insert i-set into pattern tree
3       for (j=1 ; j<i ; j++)
4           intersection-set= i-set ∩ j-set
5           if intersection-set ≠ ∅
6               insert intersection-set into DS
7       for each set i in the DS
8           insert i-set into pattern tree
9           if pattern tree has already contained the same branch
10              if the branch is new
11                  increase occurrence
12              else
13                  occurrence just can be updated once
14          else
15              insert i-set into OS
16 end
```

```
Procedure Traversal
Task: mining changes of patterns
1 for each item, i in Pattern-Tree PT
2       while traversing every node linked with i
3           if node.rate≥threshold
4               the node and all its ancestors are changed patterns
5           else
6               if node is not a leaf node
7                   mark the node to be ignored
8               else
9                   delete node
```

當原始搜尋樹中所有項目皆已完成Procedure *MCPCPT*之後，則我們已為所有項目建立好其各自的樣式樹了。因此，就儲存結構而言，會形成一個樣式森林(pattern forest)。

更進一步地，我們可以根據樣式樹探勘出所有的變化樣式，同時對樣式樹進行調整。Procedure *Traversal*是探勘樣式變化的作法，其第1行說明走訪樣式樹的次數等於樣式樹中所具有的項目數；第2行將走訪所有與該項目相連的節點，這可透過相同項目節點間所建立的連結來達成；第3行將檢查所在分支的候選樣式是否符合門檻值。我們將會從最底部的節點開始走訪，當某節點的變化率符合使用者所設定的門檻值時，則代表該樣式是一個變化樣式，而該節點以上的所

有父節點皆為符合需求之樣式，如b樣式樹中的a節點以上存在分支為b→d→a。當樣式的變化率不足時，即代表該樣式不再是我們感興趣的樣式了，故無法作為候選樣式；這時第6行會先判斷該節點是否有子節點(是否為葉部節點)；若有的話，則不能直接刪除，因為若刪了此節點的話，則該分支會喪失連結。舉例來說，在b→ a →c這個分支中存在著樣式{b,a}及{b,a,c}，雖然{b,a}不是變化樣式，然而刪除節點a會遺失變化樣式{b,a,c}的連結，故只要將節點a作記號以表示{b,a}已非候選變化樣式，且之後的探勘亦不再考慮此樣式；反之若該節點不含子節點的話(也就是葉部節點)，則第9行可將該節點逕行刪除。

3.3.3 探勘多期變化樣式

上一節的方法雖然可以探勘出樣式的變化，但仍只限於兩期；在探勘多期變化樣式之前，須先加入其它期的資料，且不須再建立新的搜尋樹。Procedure *UDAPT*會將資料加入至各個樣式樹中。Procedure *UDAPT*的第1行將會處理資料集合中的每一筆交易，第2行會將該筆交易依照項目出現次數由少至多的順序做排序，這是為了配合樣式樹的節點順序；第3行迴圈的次數為該筆交易所具有的項目數再減一，這是因為要將資料逐一加入到樣式樹中，如交易為{b,d,a}，則該交易將被加到b樣式樹；由於{b,d,a}也包含{d,a}，故也要將其加入d樣式樹中；因單一項目不被視為樣式，故a不用被加入至a的樣式樹中。因此，就此交易而言，具有三個項目，故要尋找兩個樣式樹；第4行即是找到該項目所屬的樣式樹；找到樣式樹之後，將會把交易內所具有的項目與樣式樹的分支進行比對，第5行顯示比對的次數為該交易所具有的項目數減掉目前處理過的項目數；舉例來說，假設交易為{b,d,a}，若我們現在所處理的是d樣式樹，那麼比對次數為兩次；第6行判斷樣式樹中是否有分支與交易是相同的，也就是說樣式樹中是否存在相同的樣式，若存在的話，第7行就會更新樣式樹中每個項目節點的出現次數；而8到12行的目的是為了檢查樣式樹中存在某些樣式與該樣式的超集合，當此情形發生時，會先找到超集合樣式的分支，同時更新其出現次數；為避免其它子集合的樣式被忽略計算，故應一併更新。第8行的node→next即是檢查在樣式樹中是否存在其它分支有著相同的項目節點；第10行是為了判斷該樣式

是否為子集合樣式，因為唯有該節點的父節點被更新的情況下，才能更新該節點；反之若父節點沒被更新，即代表此樣式不為子集合樣式。舉例來說，假設樣式樹共存在三個分支，分別為c→d→a→b、c→d→b及c→a→b。現有一筆交易為{c,d,a,b}，則分支c→d→a→b毫無疑問地會被更新。其更新流程如下：先找到樣式樹c並更新節點d的支持度；再尋找樣式樹中其它分支是否也存在節點d，發現分支c→d→b有d，且其父節點c也已被更新，故更新d；再繼續更新下一個節點a，發現樣式樹中存在另一個分支存在節點a，即分支c→a→b中的a，而a的母節點c已被更新，故節點a的支持度亦會被更新；同樣地，在更新節點b時，c→a→b的節點b也會被更新。故第一筆交易會更新超集合樣式{c,d,a,b}及其子集合樣式{c,d,b}和{c,a,b}。而第二個交易為{ c,d,b,e,f}，更新的第一個項目為c，則這三個分支都會被更新。在更新第二個項目d時，c→d→a→b及c→ d→b的d都會被更新。但是更新第三個項目b時，只有分支c→ d→b的b會被更新，而另兩個分支c→d→a→b與c→ a→b的b都不會被更新，因為這兩個分支的父節點都沒被更新。由這兩個例子可發現當讀入一筆交易時，其實只會尋找唯一的樣式進行更新，然而還可能存在其它樣式也需更新，這時只要透過上述說明的處理方式，便可一併更新。倘若需要更新不只一個樣式的話，那一定是因為存在著子集合樣式。當Procedure *UDAPT*執行完畢時，便是將另一期的資料全部加入樣式樹了。要注意的是，只有樣式樹所具有的樣式才會被更新，也就是說當資料集合中的交易沒有和任一樣式樹的樣式相同時，該交易就會被忽略，因為該交易並不具有變化樣式所需的資料。

分析**2.** 更新時會尋找交易中每一個項目所對應的樣式樹，如交易有*m*個項目，則會尋找*m-1*個樣式樹。找到樣式樹之後則更新與交易相對應的分支。此時所需的成本與分析1的插入交易至搜尋樹的成本是相同的，即$O(m\times k)$，在此k為搜尋樹中每一層可能的節點個數。另外，找到樣式樹中的每一個節點時，另需花費時間尋找是否有相連的節點需連帶更新。連帶更新的節點數將介於0與該筆資料集合所包含的交易數*Tran*之間，即每一筆交易都包括該項目。因此更新一筆交易至樣式森林時，所需的成本為$O((m\text{-}1)\times\ m\times\ k\times\ Tran)$。

```
Procedure UDAPT
1 Task: updating data and adjusting Pattern Trees (PT)
1 for each transaction T_h in dataset
2       sort each item i∈ T_h according to the order of the FPlist
3       for each i∈ T_h
4             find i-PT
5             for each item j from i to the last, j∈ T_h
6                   if j matches the branch, B of i-PT
7                         node.Rcount++
8                         while node→next is not NULL
9                               current=node→next
10                              if current→parent is updated
11                                    current.Rcount++
12                              node=current
```

更新資料後，才能探勘變化樣式；由於我們已經建立了樣式樹，再來只要進行各個樣式樹的走訪，也就是透過Procedure *Traversal*即可探勘出另一期的變化樣式。因此，只要重複地操作Procedure *UDAPT*及Procedure *Traversal*即可探勘多期的樣式變化。

3.4 時間容忍

我們原本的目的是想要發現樣式穩定地呈現某種變化趨勢，譬如成長或是下降。但是當探勘的期數愈多時，我們可以瞭解能夠一直呈現某種變化趨勢將會非常困難。當樣式的變化率不符合門檻值時，我們將這種現象稱之為「反常」。按照原先的探勘流程，當發生反常時，我們將會停止對該樣式的探勘。然而，若樣式在整個探勘過程中只發生少數的反常，大體而言，該樣式仍然呈現特定的變化趨勢。因此，為了增加探勘樣式變化的彈性，本研究亦提出了一個時間容忍的概念。時間容忍是一個允許在探勘過程中能夠接受資料集合其變化率不符合變化率門檻值的一個機制。假設我們想要探勘十期的資料集合，並且將時間容忍值設為三，其代表我們可以接受十期中任三期的變化率不符合門檻值，也就是允許三次的反常，如圖3.9所示。舉例而言，圖3.7中的樣式只有在第七與第八期是下降的，但是長期看來還是呈現成長的趨勢。若依照我們原本的探勘程序，則圖3.7中的樣式在第七期不符合門檻值，故將不會繼續探勘一到六期的資料；然

而，如果我們能夠忽略這次反常的話，就能夠發現此樣式在十期中有九期皆是成長的。故為了探勘出那些在大部份時間點其變化趨勢仍符合需求之樣式，我們提供使用者一個彈性機制，讓使用者可以設定。

由於變化率的計算是以樣式的前一期(t_k)的支持度除以後一期(t_{k+1})的支持度所得到的。因此，在加入時間容忍值之後，我們需要討論發生反常時，該選擇第t_k期或第t_{k+1}期的資料集合為基礎較佳。反常代表樣式的變化違反了趨勢，若我們想要探勘的是成長的趨勢，即發生了該成長卻變成了下降的現象；更詳細的說明，即樣式在第t_k期支持度原本應該小於第t_{k+1}期的支持度，然而資料中卻反而是第t_k期的支持度比較大。若我們要繼續探勘前一個時間點第t_{k-1}期，則在計算變化率時，自然是選擇支持度較大的時間點與t_{k-1}比較，其發生反常的機率才會降低。也就是說為了增加繼續探勘的可能性，應該選擇在原本反常中的前一期資料集合為基礎是比較理想的。舉例來說，在圖3.7中，若要繼續探勘第六期，則選擇第七期比第八期為佳。若以第七期為基礎的話，則能夠探勘圖3.7及圖3.8的趨勢；以第八期為基礎，則只能找出圖3.8的趨勢。

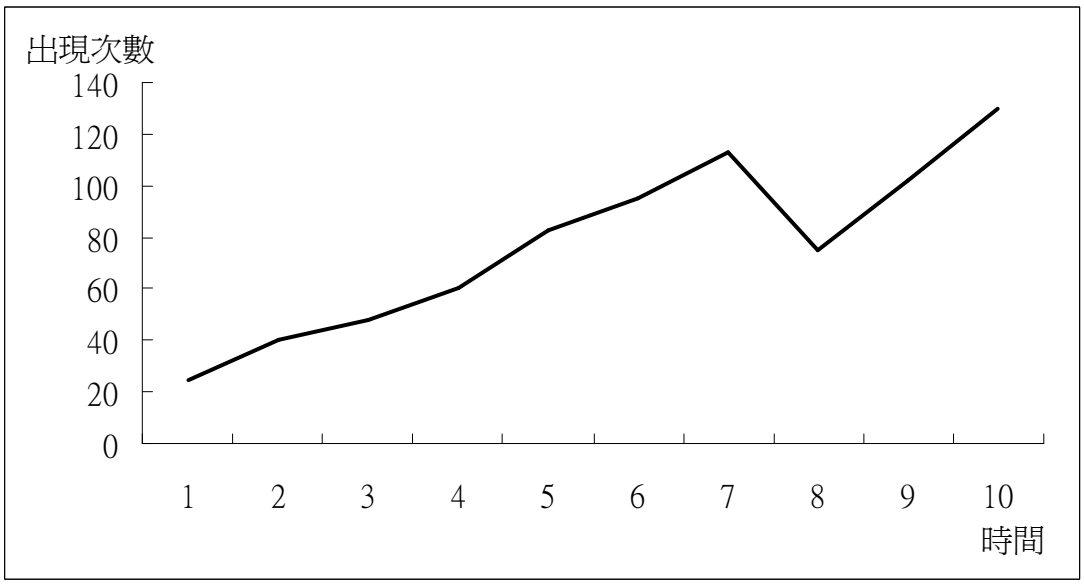


圖 3.7: 具有時間容忍值的圖形1

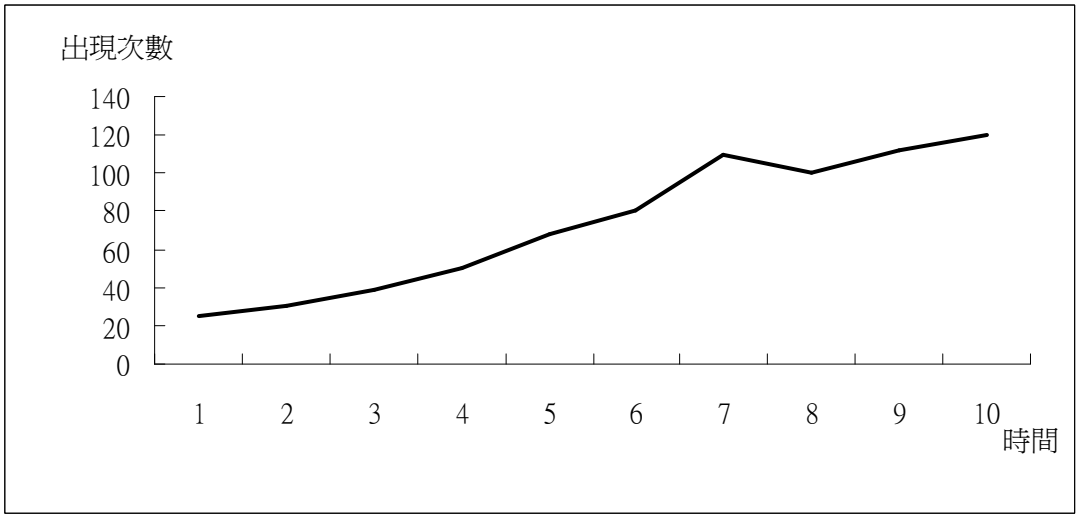


圖 3.8: 具有時間容忍值的圖形2

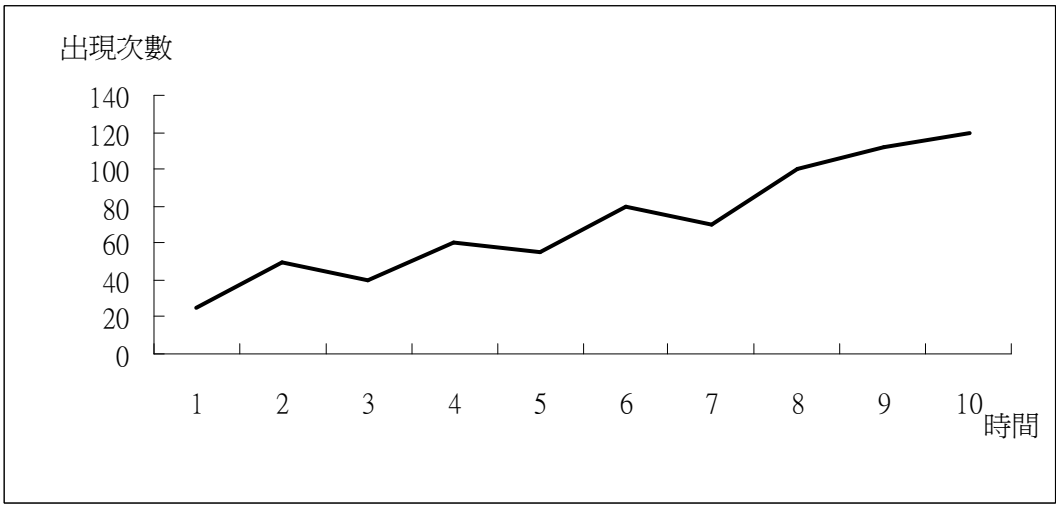


圖 3.9: 時間容忍值單位為3的圖形

3.5 範例說明

在這個小節中，我們將以一個簡單的例子說明演算法是如何進行並且能得到所要求的結果；在這裡將使用三期的資料，因為演算法在處理三期與更多期資料的方式都一樣，故為了方便說明，在此只用三期資料作為說明之範例。表3.1及表3.2中，TID代表交易序號，而Item指該筆交易所包括的項目；首先依照表3.1中項目的出現次數進行排序，依序為{c,a,d,e,b}，如圖3.10所示。建構原始搜尋樹的同時，也需建立FPlist以儲存排序後的項目順序；FPlist尚包含一個指

標指向每個項目在原始搜尋樹中的位址，其目的是方便進行樹的走訪。原本的每一筆交易要依排序後的順序調整項目的位置再加入至原始搜尋樹中，調整後的結果將如表3.3。需注意的是，每一筆交易是在被加入原始搜尋樹時才依照FPlist的次序調整其項目位置。加入第一筆交易後的原始搜尋樹為圖3.11。當加入完所有第3期的資料後，搜尋樹將如圖3.12所示。緊接著，我們需加入第二期的資料至原始搜尋樹中。按照FPlist中項目的次序，將第二期資料中每筆交易的項目重新排序後的結果將如表3.4所示。依照同樣的程序，將第二期的每一筆交易逐筆加入至原始搜尋樹之後的結果如圖3.13所示；其中分支c→a→d→e的e與c→d→ b的b皆是根據第二期資料所長出的節點。建立完成原始搜尋樹之後，便要開始探勘目標候選樣式及建立各個項目的樣式樹，建立的順序是由項目b到c。在這一節的例子中，我們將只示範項目b的部份，整個程序顯示於圖3.14中。在圖3.14左邊的項目集合是從原始搜尋樹中探勘出的所有與項目b相關的分支共計四條；我們將這四條稱之為原始集合並以一個序列儲存。由我們的方法，可以從原始項目集合推導出所有的候選樣式。探勘的程序為將項目集合逐一加入至樣式樹中，再依序比對序列中先前的所有項目集合。第一個原始集合為{e, a, c}，其會被加入至樣式樹中以產生第一個分支，顯示在圖3.14中的建立樣式樹部份的第一個樹。圖3.14中左邊部份的項目集合次序會對應至圖3.14的交集比對及建立樣式樹部份的次序。因為{e, a, c}之前沒有其他項目集合，故不需比對。第二筆為{d}，同樣地需要被加入至樣式樹中，接著再與先前的項目集合(第一筆)比對時，我們可以發現沒有交集，故繼續處理第三筆。第三筆可以發現與第一筆{e,a,c}產生交集，也就是衍生集合{e, a}，然而樣式樹中已存在此分支，故只要直接更新出現次數即可。第四筆的話，仍然有衍生集合{c}與{d}，而衍生集合{c}無法找到樣式樹中相對應的分支，故會被加入至序列中成為第五筆項目集合及進行交集比對。當序列中的項目集合皆處理完畢時，就完成項目b的樣式樹了，其顯示在圖3.14建立樣式樹中的第五棵樹。

當所有項目的樣式樹皆已建立完成時，就可以直接從樹中探勘出所有的樣式變化。假設變化率門檻值設為1，那麼只要節點的變化率符合門檻值時，在節

點以上的分支即是一個變化樣式。以項目b樣式樹為例，包括的樣式為{b,e,a,c}變化率為1、{b,e,a}變化率為1、{b,e}變化率為1、{b,d,c}變化率為0、{b,d}變化率為0.5及{b,c}變化率為0.5。而符合門檻值的只有{b,e,a,c}、{b,e,a}和{b,e}，因此其他不符合的節點將會被修剪。修剪過後的b樣式樹顯示在圖3.14。其他項目的樣式樹之處理方式相同，圖3.15顯示所有的樣式樹；項目d與a因為不存在任何變化樣式，故沒有樣式樹。另外，e樣式樹上的節點a上之*代表{e,a}已不是變化樣式(變化率為0.75不符合門檻值)，因此節點a原本應該被修剪，但是{e,a,c}卻是一個變化樣式，刪了節點a之後便會喪失連結，故不能刪除，但是之後也不會再考慮樣式{e,a}。

表 3.1: 第3期的資料

TID	Item
1	a,c,d
2	a,b,c,e
3	c,d,e
4	b,d
5	a,b,e
6	a,c,e
7	c,d

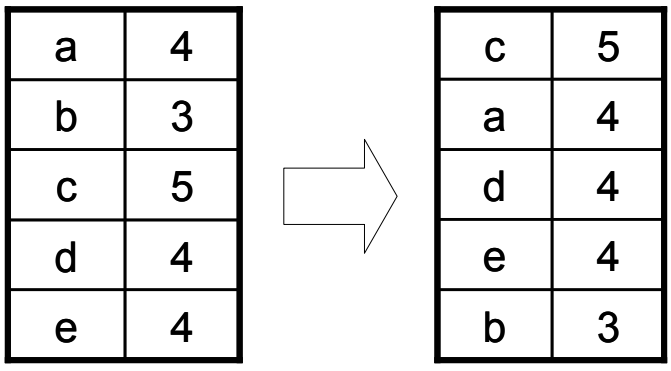


圖 3.10: 排序

表 3.2: 第 2 期的資料

TID	Item
1	b,d,c
2	a,c,d,e
3	c,d,e
4	a,b,c,e
5	a,b,e
6	a,c,d
7	a,e
8	b,d

表 3.3: 排序後第 3 期的資料

TID	Item
1	c,a,d
2	c,a,d,e
3	c,d,e
4	d,b
5	a,e,b
6	c,a,e
7	c,d

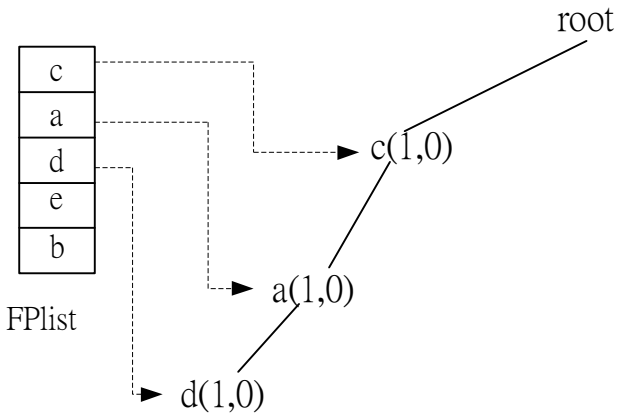


圖 3.11: 加入第一筆資料的原始搜尋樹

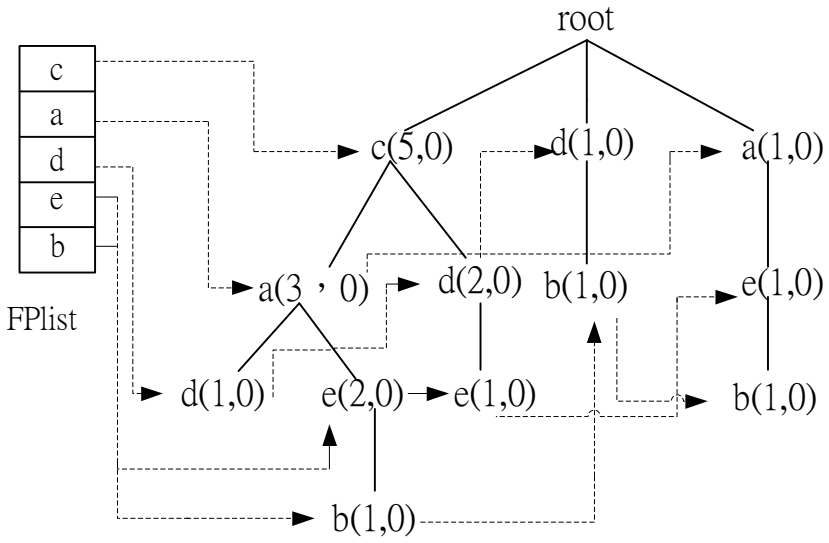


圖 3.12: 第3期資料的原始搜尋樹

表 3.4: 排序後第2期的資料

TID	Item
1	c,d,b
2	c,a,d,e
3	c,d,e
4	c,a,e,b
5	a,e,b
6	c,a,d
7	a,e
8	d,b

現在繼續探勘第一期的變化樣式，第一期資料只要加入至樣式樹即可，每筆交易同樣依照FPlist中項目的次序進行排序，結果如表3.5所示；每一筆交易將被加入至圖3.15中的樣式樹；加入第一筆交易{e,d,c}時，首先尋找e的樣式樹並更新分支e→c的出現次數。而項目d和c都沒有樣式樹，故結束第一筆交易；第二筆交易也是相同情形，即不存在d樣式樹；第三筆交易找到b的樣式樹，並更新分支b→e；接著繼續尋找下個項目e，故e樣式樹的分支e→c也會被更新，其它交易的處理方式相同。當第一期的資料全部更新後，便如圖3.16所示。最後進

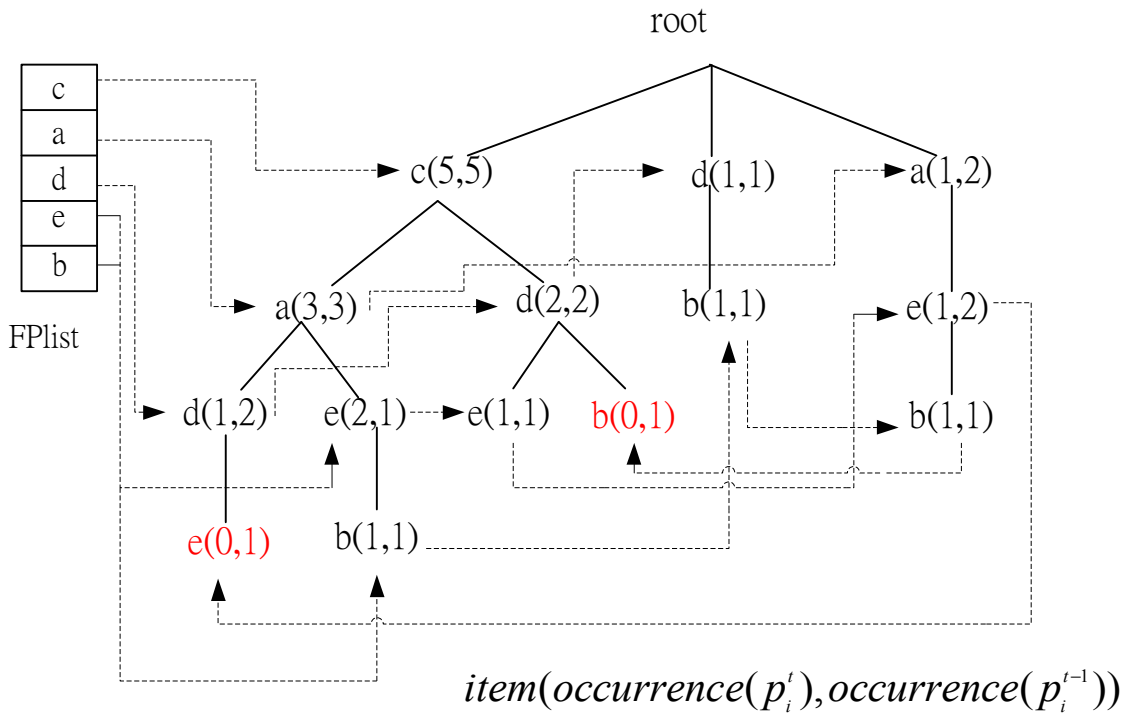


圖 3.13: 加入第 2 期資料的原始搜尋樹

表 3.5: 排序後第 1 期的資料

TID	item
1	e,d,c
2	d,a
3	b,e,c
4	b,e,a,c
5	e,d,a
6	b,a,c
7	e,d,c

行樣式樹的走訪及計算變化率以判斷樣式是否發生變化。我們可以發現只有樣式{e,c}的變化率小於門檻值，刪除此節點之後，最後的樣式樹如圖3.17所示。另外，我們將這三期中的變化樣式及其變化率整理在表3.6中。

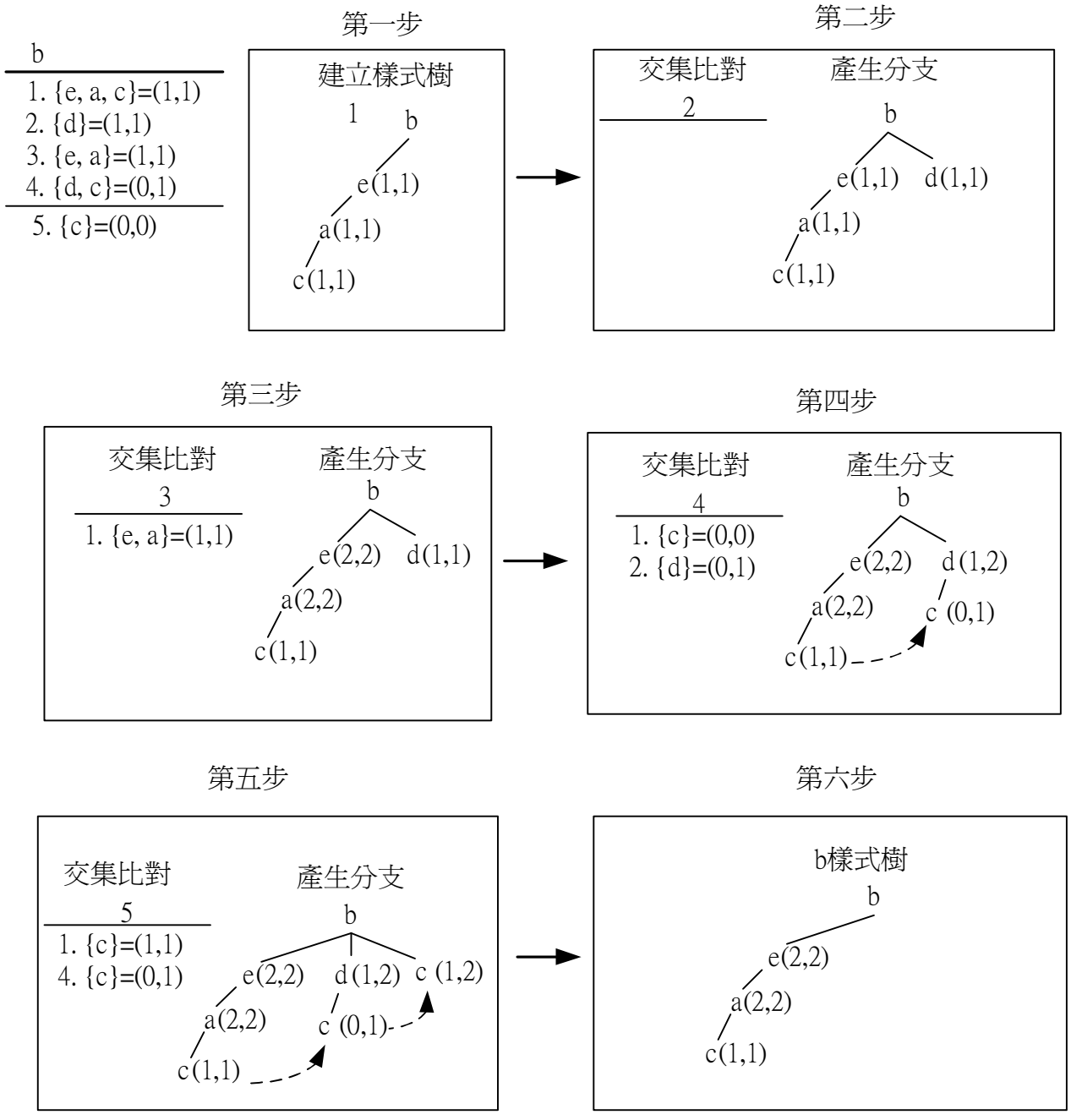


圖 3.14: 挑選項目 b 候選樣式的處理程序

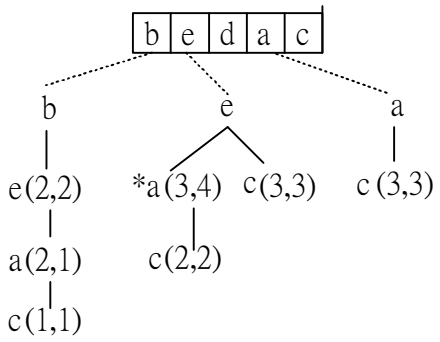


圖 3.15: 所有的樣式樹

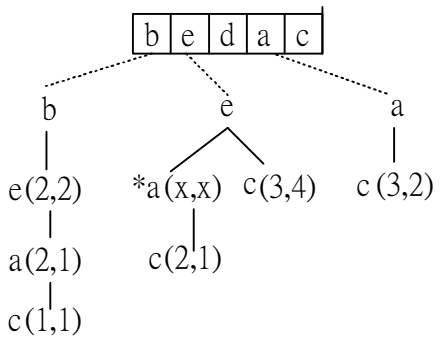


圖 3.16: 更新第一期資料後的樣式樹

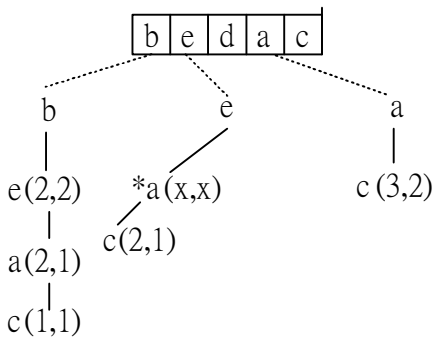


圖 3.17: 修剪過後的樣式樹

表 3.6: 全部的變化樣式

變化樣式	變化率(1~2 期)	變化率(2~3 期)
b,e,a	200%	100%
b,e,a,c	100%	100%
e,a,c	200%	100%
e,c	-	100%
a,c	150%	150%

3.6 小結

本章解釋了本研究欲處理的問題，包括變化率的設定方式及欲的探勘變化類型。為了解決在多期資料中探勘樣式變化的問題，本研究提出了MCP 演算法，主要有三個階段，分別是：(1) 建立原始搜尋樹：以作為探勘樣式基礎的儲存結構、(2) 探勘候選樣式及建立樣式樹：以尋找樣式，(3) 更新資料與調整樣式樹的程序。同時，本研究也提出了一個能夠有效存取資料的結構，名為樣式森林；我們會為每一個項目建立獨立的樣式樹，而樣式森林則是樣式樹的聚合體。藉由本研究所使用的樣式森林，可以有效地探勘多期資料中樣式的變化。另外，為了增加MCP 演算法探勘樣式變化的彈性，我們加入時間容忍的概念，能夠更寬鬆的認定變化之發生。為了幫助理解MCP 演算法的作法，我們也在3.5 節中以一個簡單的例子輔助說明。

第四章

資料測試與分析

這個章節將會實作本研究的演算法，並且測試其效能。由於文獻中並無相關方法可以直接處理我們的問題，然而為了能夠比較及顯示本研究所提出的MCP演算法之效率，我們將FP-growth演算法(Han et al., 2004)修改成一個能夠探勘多期樣式變化的演算法，並將之命名為modiFP演算法。同時我們也設計了一個資料產生器以產生多期的測試資料。本研究的實驗皆是在處理器為3GHZ且具有2GB主記憶體下的電腦執行，作業系統為Windowns XP，測試平台為cygwin，所有的程式皆以C++撰寫。

4.1 資料產生器說明

本研究所使用的資料產生器乃根據Agrawal and Srikant(1994)所使用的交易資料產生器修改而成。所有關於原本交易資料產生器的相關資訊可以參考Agrawal and Srikant(1994)。為了可以產生多期資料，除了該資料產生器原本的參數之外，我們又加入了期數(T)與相似度(S)這兩個參數。其中，期數(T)或可稱為時間點，根據使用者給定之T，資料產生器能夠連續產生T期的資料以供探勘。例如我們以月為單位，希望探勘一年份的資料量，則可將期數設為12，便能夠產生連續12個月份的資料。另外，在探勘考量時間性資料的樣式時，Li et al.(2006)認為不同時間點的資料彼此應該存在著一定程度的關係，因此加入了相似度這個參數，以使測試資料能夠更接近真實環境。同樣地，本研究亦將在交易資料產生器裡加入相似度。在我們的設定中，假設我們每期欲產生D筆交易資料，那麼根據相似度S的設定，每一期會產生新的(1-S)×D筆資料，另外再從上

一期隨機抽取$S \times D$筆資料。因此，當S設為0.4時，每一期產生10000筆資料中將有4000筆是從上一期的10000筆中隨機挑選出來的，而另外的6000筆資料則由產生器重新產生。同樣地，當我們把相似度設定為1時，代表每一期的資料皆完全相同；而設定為0時則表示每一期的資料毫無關連，彼此皆是獨立的。由於Li et al.(2006)把每一期的相似度設定在0.4到0.8之間，因此本研究的實驗也將相似度設定在這個範圍之間。表4.1列出所有實驗中所包含的參數及預設值。

由於modiFP演算法需要大量儲存空間，因此在比對modiFP與MCP演算法的相關測試中，我們只能使用較小的資料量。因此，我們將資料產生器的參數L設為5，P設為20所產生的資料作為本實驗的測試資料，而每一期所產生的資料約佔1.8MB。

表4.1: 參數與預設值

參數	意義	預設值
D	每期的交易數	10000
I	項目數	1000
L	每筆交易平均的項目數	5
P	樣式個數	20
T	期數	9
S	相似度	0.4~0.8

4.2 modiFP演算法

modiFP演算法將會與本研究提出的MCP演算法一起進行測試，並且比較其執行效率。首先我們先說明modiFP演算法的概念，其探勘流程顯示在圖4.1中。modiFP演算法是一種階層式的方法，也就是先探勘出每一期資料集合中的所有樣式再與候選樣式進行比對以篩選出符合變化的樣式，並在連續的兩期資料間重複進行此樣式比對之程序。在探勘樣式的部份，modiFP演算法與FP-growth演算法的處理方式完全一樣。圖4.1中程序一的部份將分別探勘出最後兩期資料集合的樣式，再使用兩個序列儲存所有樣式以便下一階段的樣式比對。由於我們需要用序列儲存資料集合中的所有樣式，因此需要很大的儲存空間，這也是modiFP演算法無法進行大量資料探勘的理由。尋找候選樣式時，即需要三個序列儲存樣

式(圖4.1中的樣式序列1、樣式序列2及目標候選樣式)。而在探勘多期變化時，每一期都需儲存新的一期資料集合所探勘的樣式與上一次的樣式比對後的候選樣式序列及比對後的候選樣式序列，因此主記憶體仍是需要儲存三個序列。由此可知，modiFP演算法是相當耗費記憶體空間的。圖4.1中替程序二主要在處理樣式的比對過程。首先，它將逐一循序比對以找出兩期資料集合中完全相同的樣式，當找到之後立即進行變化率的計算，若符合門檻值則此樣式即被儲存在目標候選樣式序列裡。此序列之後還會不斷與每一期資料所探勘出來的樣式再進行比對並更新目標候選樣式序列；當全部資料皆比對完畢之後仍留在此序列的樣式即為在所有時間裡皆持續變化的樣式。

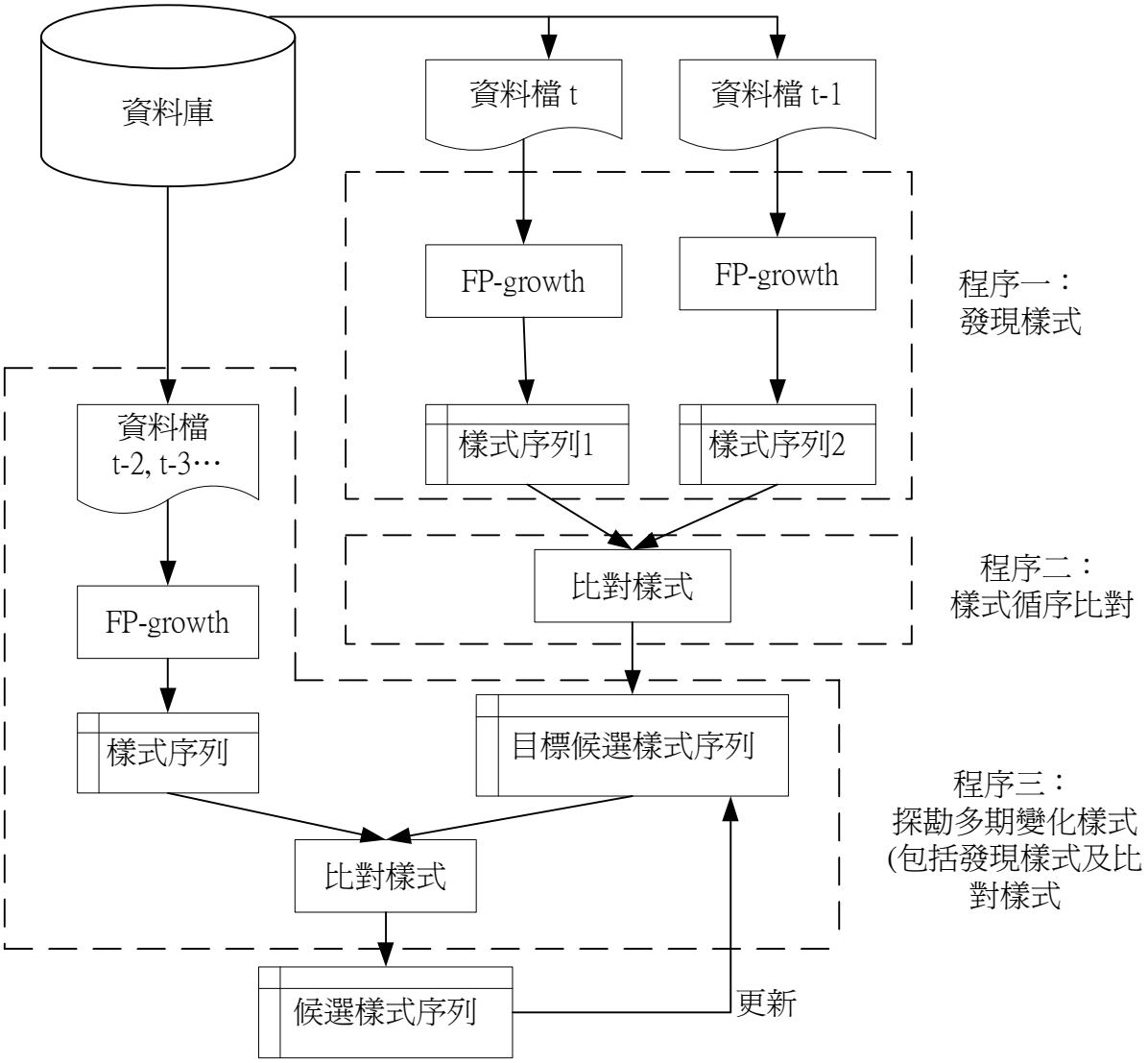
4.3 測試結果

在本小節中將會顯示演算法的實驗結果及分析說明。

4.3.1 驗證期數的影響

實驗的目的為比較modiFP與MCP兩個演算法在不同期數下的效率表現。因此我們連續探勘九期的資料檔以比較其效率。每期資料檔包含一萬筆交易，而連續兩期資料檔間的相似度設定為0.6。實驗結果的實際數字顯示在表4.2及圖4.2中。從結果可以觀察出modiFP演算法對每一期的資料檔都需要花費比MCP演算法還多的時間。這是因為不管探勘多少資料檔，modiFP演算法仍需要花費時間來尋找每個資料檔中的所有樣式，故它花在探勘樣式的時間是無法減少的。modiFP演算法能夠減少的只有在比對樣式的部份。因為隨著期數愈多，能夠符合要求的樣式必然會減少，所以從圖4.2可觀察出曲線漸趨水平。然而，從MCP演算法的執行結果來看，我們可以發現探勘期數的多寡對於MCP最演算法後的執行時間並沒有很大的影響。表4.2顯示出MCP演算法在處理多期資料時皆只需要一秒左右的時間，這是因為MCP演算法探勘出候選樣式時即已建立了樣式樹，而再探勘另一個資料檔時，只要更新樣式樹的資料即可以探勘樣式的變化，不需要再重新尋找資料檔中的樣式。易言之，MCP演算法在多期探勘時，只要

圖 4.1: modiFP 探勘流程圖

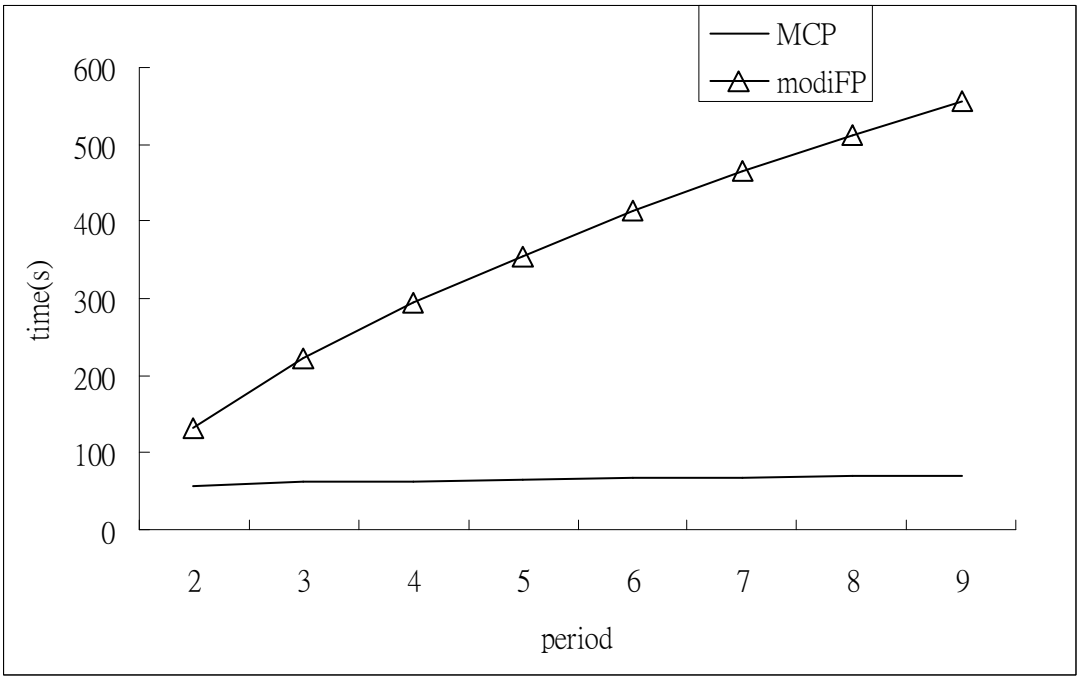


花一次的時間探勘所有樣式，同時建立每一樣式的儲存結構，因此在這個部份將花費最多時間，之後只需進行更新資料及樣式樹的走訪即可很快地找出變化樣式。圖 4.2的實驗結果顯示MCP 演算法的執行時間不會隨著探勘的期數增加而大幅增加，因此MCP 演算法是相當適合進行多期資料探勘的演算法。

表 4.2: 期數對執行時間的結果表 (時間單位 : 秒)

期數	MCP	modiFP
2	58.015	130.828
3	61.828	221.816
4	63.312	293.767
5	64.812	353.922
6	66.313	413.486
7	67.813	464.265
8	69.313	511.39
9	70.828	555.484

圖 4.2: 驗證期數



4.3.2 驗證相似度的影響

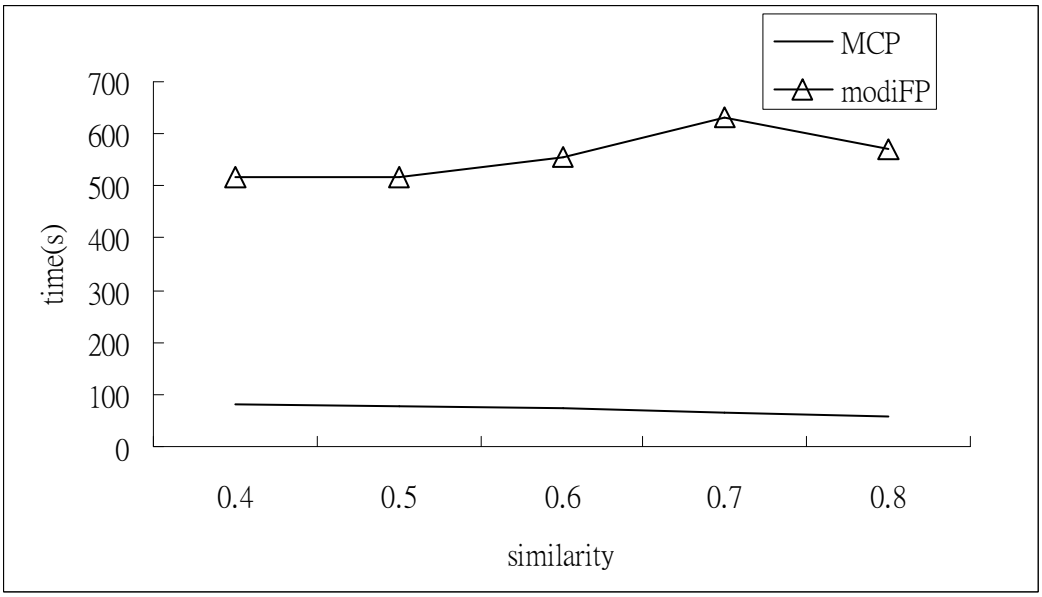
第二個實驗為驗證不同資料檔間相似程度的高低對演算法效率上所造成的影響，實驗的結果顯示在表 4.3及圖 4.3中。我們可以觀察出相似度對於 modiFP 演算法而言並沒有很顯著的影響。modiFP 演算法在樣式探勘時，其處理方式與 FP-

growth是相同的。然而，FP-growth是探勘單一資料檔的演算法，故以modiFP演算法探勘多期資料可視為一次又一次地執行FP-growth於不同的資料檔。而這些不同資料檔之間的操作，彼此是不相關的。因此，對於modiFP演算法而言，每個資料檔在探勘樣式時皆被獨立處理，所以相似度對modiFP演算法的影響並不大。

表 4.3: 相似度對執行時間的結果表 (時間單位 : 秒)

相似度	MCP	modiFP
0.4	80.796	515.11
0.5	76.797	516.56
0.6	70.828	555.484
0.7	64.421	630.814
0.8	64.421	569.654

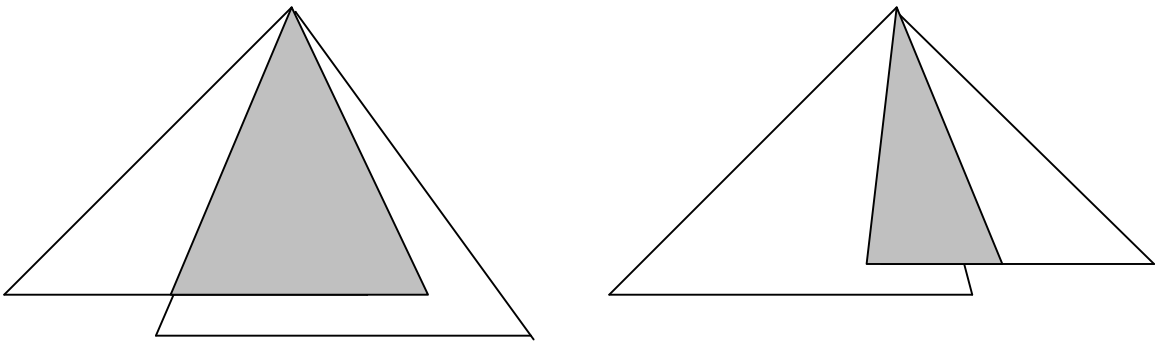
圖 4.3: 驗證相似度



然而，測試結果卻顯示出相似度與MCP演算法的效率有很顯著的關聯，亦即相似度愈高時，MCP演算法的執行效率就愈佳。這是因為每一期資料檔間的相似度愈高，代表資料檔的特性愈接近，而產生相同項目集合的機會也較高，因

此所建構出原始搜尋樹可以用較少的分支去涵蓋更多的項目集合，使得搜尋樹呈現較為緊實的形狀，如圖4.4左邊左圖所示。在圖4.4中，三角形代表是搜尋樹，從圖中可以發現左邊兩個三角形重疊的部份較多，代表兩期的資料檔中有很多項目集合是重複的；反之當資料檔的相似度愈低，則將產生更多不同的項目集合，這將使得搜尋樹長出更多的分支；也就是搜尋樹會呈現較為鬆散的狀態，兩個三角形重疊的部份也將會較少，如圖4.4右邊子圖所示。由於緊實的搜尋樹其分支會較少，因此需要進行比對的次數也會較少，而比對的次數正是影響MCP演算法效率最直接的因素。因此MCP演算法在探勘候選樣式時所需的時間會隨著搜尋樹的緊實(也就是資料檔具有較高的相似度)而減少。圖4.5顯示MCP演算法在不同相似度之下尋找候選樣式時所進行的樣式比對次數，由其結果可知若連續兩期的資料間之相似度愈高則MCP演算法所需進行的樣式比對次數將為之下降，同時也造成花費的執行時間因而減少。以另一個角度來看，在探勘多期資料時，MCP演算法會先將不同期的資料檔先輸入至樣式樹中，而這部份所花的時間佔總執行的時間的比例相當低。也就是說，對MCP演算法而言，影響效率的關鍵程序為探勘候選樣式，而相似度愈高愈可減少這部份的時間，因而導致MCP演算法的效率隨之提升。

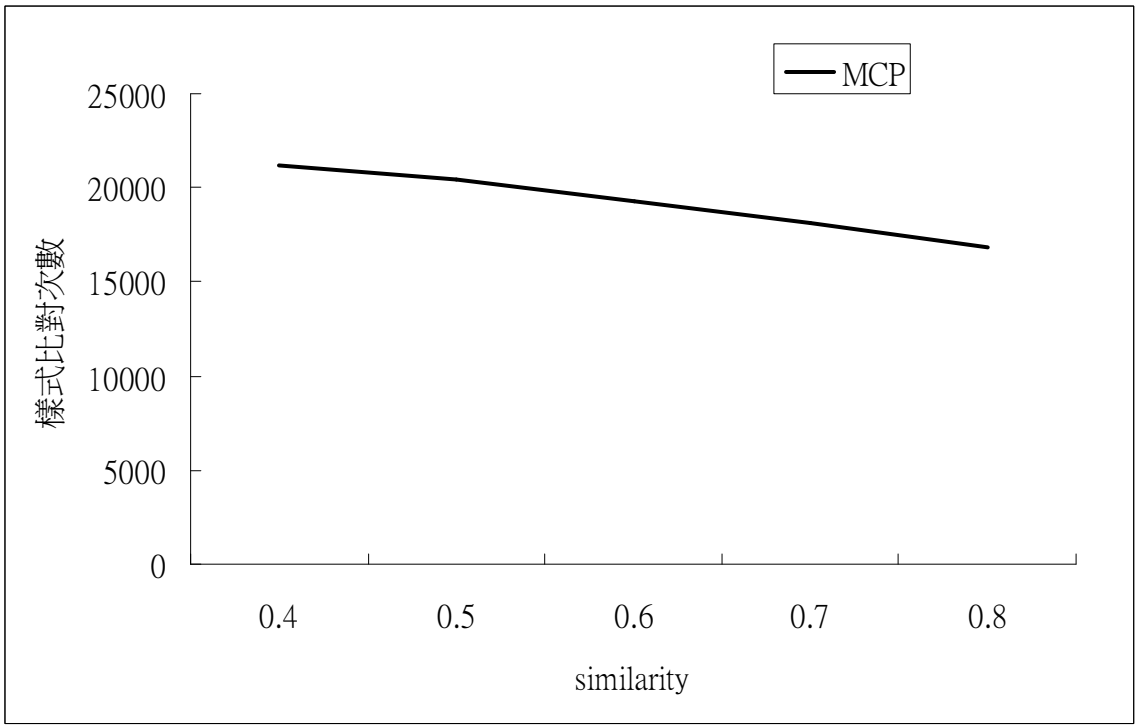
圖 4.4: 緊實與鬆散的搜尋樹



4.3.3 驗證支持度的影響

當使用者想探勘出哪些符合變化需求且又足夠頻繁的樣式時，則演算法必須在探勘樣式變化之外，同時考量探勘頻繁樣式的最小支持度門檻值。一般而

圖 4.5: MCP 演算法在不同相似度的樣式比對次數圖



言，最小支持度門檻值通常是影響樣式發現相關方法的執行效率之重要因素。由於交易資料庫中大多數的項目不夠頻繁，因此愈高的最小支持度門檻值，將會過濾掉愈多的樣式，而大大地減少了可供探勘的資料量。我們測試了五種不同數值下的最小支持度門檻值，其實驗結果顯示於圖 4.6及表 4.4中。從 modiFP 演算法的執行結果可以發現當最小支持度門檻值被設在 1% 至 5% 之間時，其執行時間並沒有明顯的變動。這主要是因為符合這五種支持度的樣式並不多。我們進一步比較有無設定最小支持度門檻值在多期探勘下對 modiFP 演算法的影響，由圖 4.7與圖 4.8所顯示之結果，我們可以發現 modiFP 演算法在支持度為 1% 下探勘兩期樣式所需的時間比支持度為 0% 所需的時間大幅減少，甚至在期數較少 (譬如期數為 2) 時還可能比 MCP 演算法所需的時間還少；然而隨著探勘的期數增加，其所需的時間能將跟著提高，這是因為其他的資料檔仍是需要在支持度為 0 的情況下探勘。也就是說，雖然設置了最小支持度門檻值，但因為我們無法得知前一期的支持度應為多少才能符合變化率門檻值，故除了最後一期資料檔適用該最小支持度門檻值之外，其它時間點資料檔的支持度仍應設為 0。從圖 4.7與圖 4.8中，我們可以發

現modiFP演算法在有無設定最小支持度門檻值(1%與5%)之間，其執行時間相差了近二百秒。這是因為最小支持度門檻值過濾了大部份的樣式，同時也使得需要進行多期比對的樣式大幅減少，進而導致執行時間為之減少。

最小支持度門檻值對於MCP演算法的影響，主要在於刪除交易資料庫中不符合的項目。然而，MCP演算法在走訪樣式樹時才會檢查支持度是否符合門檻值，也就是說在探勘出候選樣式及建立樣式樹之後，MCP演算法才會過濾不符合的樣式。故最小支持度對MCP演算法的影響只在於刪除愈多的項目，會使得項目集合的數量也跟著減少，連帶地降低比對的次數。從圖4.7中可發現最小支持度門檻值為0%及1%對於MCP演算法的影響不大，此乃因為小於此門檻值的項目不多。然而，當最小支持度門檻值設為5%時，從圖4.8中卻可以發現MCP演算法的執行時間將顯著下降，此乃因為符合5%門檻值的項目大幅減少所以使得候選樣式大幅減少，故比對次數也跟著大幅下降。

另外，由實驗結果可得知最小支持度門檻值對於MCP演算法的影響不如對modiFP演算法的影響來得明顯。這是因為MCP演算法要等探勘完所有樣式之後才會檢查是否符合門檻值；而此種作法主要是我們為了避免MCP演算法遺失資訊，因此才在最小支持度門檻值為0的情況下，探勘資料檔中所有的樣式。

表4.4: 支持度對執行時間的結果表(時間單位:秒)

支持度	MCP	modiFP
1%	69.609	350.689
2%	67.501	347.688
3%	64.484	347.172
4%	58.797	347.19
5%	51.687	346.623

4.3.4 驗證變化率的影響

變化率代表使用者想探勘之樣式變化程度，如變化率大於1則代表樣式的支持度在每一期都要不斷的成長，而變化率設的愈高，則所符合的樣式也將愈

圖 4.6:驗證支持度

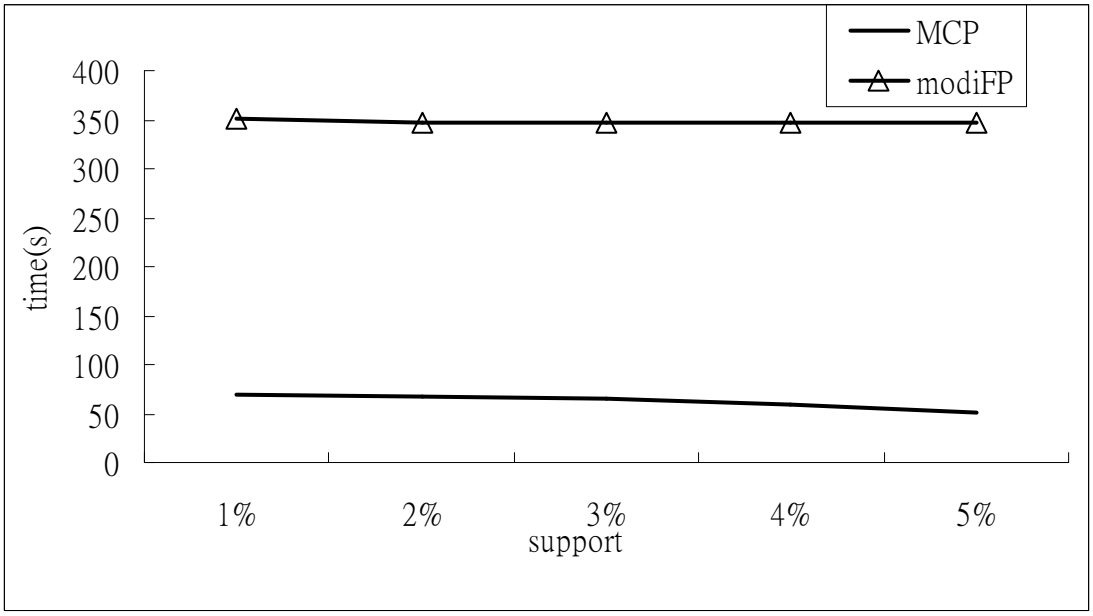
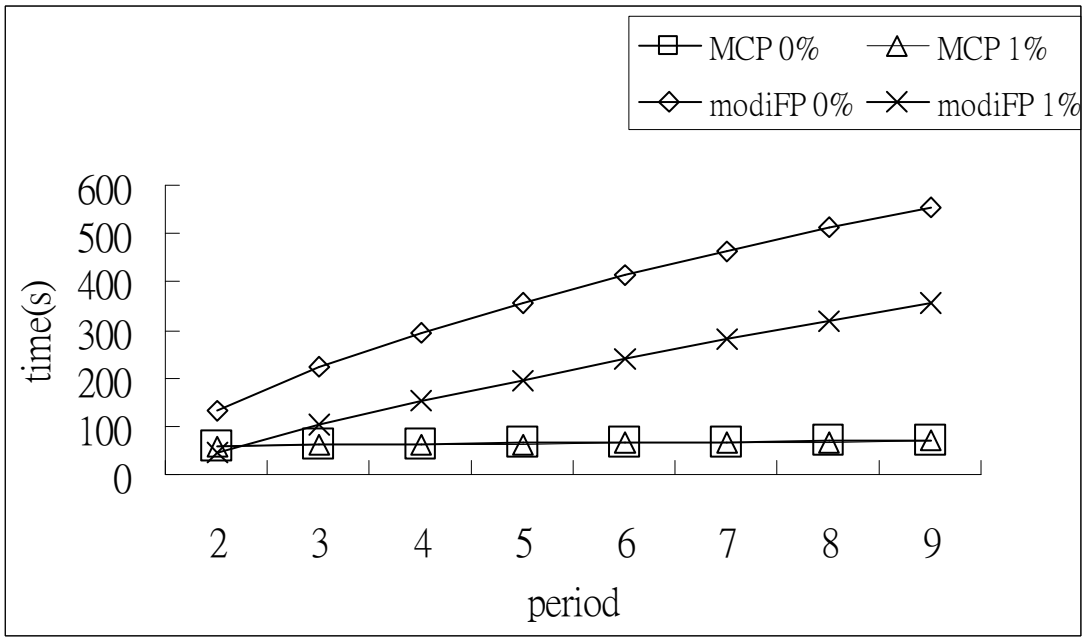
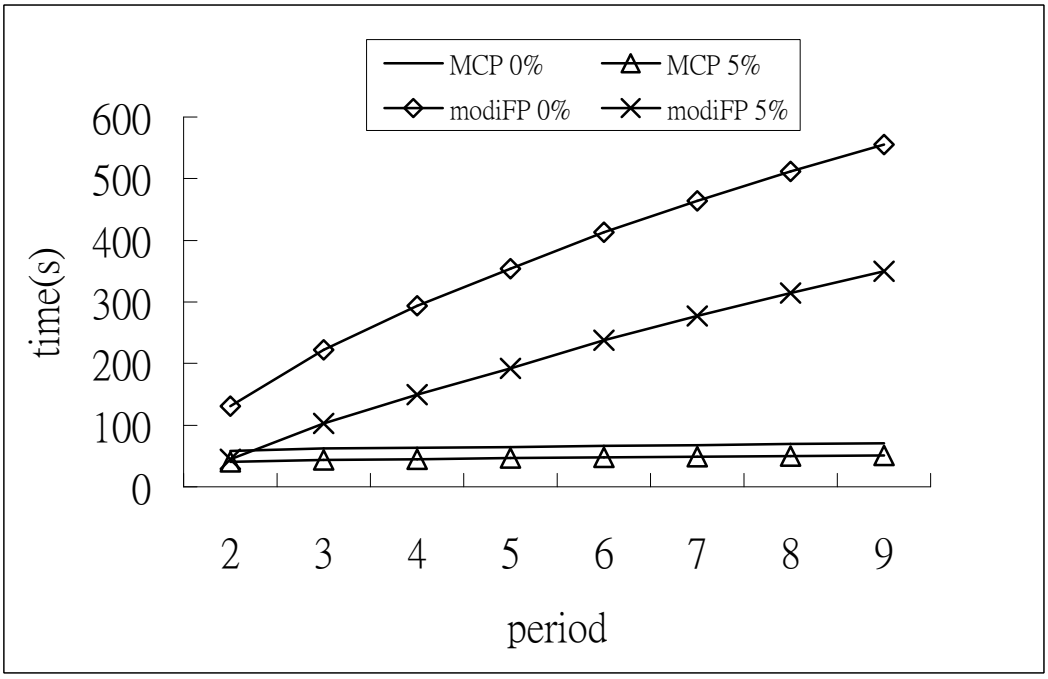


圖 4.7: MCP 演算法與 modiFP 演算法在支持度為 0% 與 1% 的執行時間比較



少；同樣的，若將變化率設的極小(譬如小於1)，代表樣式的支持度必須持續地下降，而符合的樣式也不會太多。直覺上來說，當我們將變化率設的較為極端(譬如極大或極小)時，符合變化的樣式將會減少，而所需的執行時間應該也會下

圖 4.8: MCP 演算法與 modiFP 演算法在支持度為 0% 與 5% 的執行時間比較



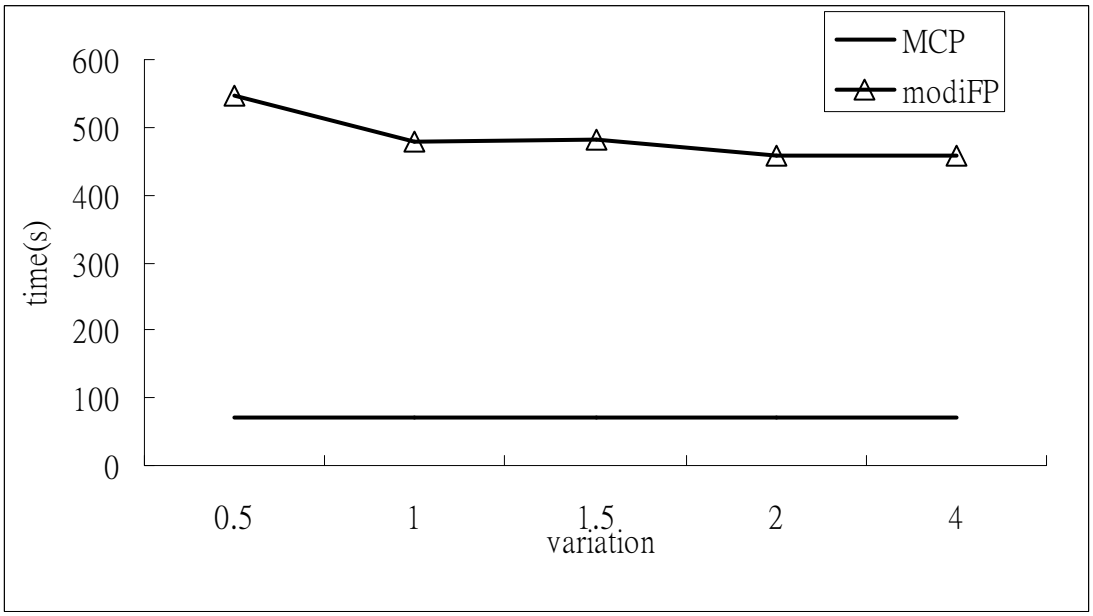
降。但是從實驗的結果表 4.5及圖 4.9中可發現變化率的大小變化雖然對 modiFP 演算法有很大的影響，然而對 MCP 演算法的影響卻不是那麼明顯。其實，這個結果蠻符合 MCP 演算法的設計概念，因為 MCP 演算法最耗費時間的程序在於篩選目標候選樣式與建立樣式樹，在建立樣式樹之後才開始驗證變化率是否符合門檻值，因此變化率對決定 MCP 演算法效率關鍵的探勘候選樣式及建立樣式樹的部份是完全沒有影響的。然而，由於不符合變化率的樣式將會從樣式樹中被刪除，故當探勘愈多期資料檔時，樣式樹也會修剪掉更多的節點，因而造成更新資料及探勘變化樣式的時間仍會隨之減少。從表 4.2中可知，由於 MCP 演算法在多期探勘所花的時間極少 (約一秒)，因此變化率對 MCP 演算法的影響不會太明顯。

對於 modiFP 演算法而言，由於變化率的設定大幅減少了符合條件的樣式，因此花費在樣式比對的時間也會跟著下降，進而造成 modiFP 演算法的執行時間下降幅度大於 MCP 演算法。

表 4.5: 變化率對執行時間的結果表 (時間單位:秒)

變化率	MCP	modiFP
0.5	69.938	547.609
1	69.875	477.45
1.5	69.968	480.72
2	69.845	459.001
4	69.891	457.782

圖 4.9: 驗證變化率



4.3.5 驗證大資料檔

由於受限於儲存空間，使得 modiFP 演算法無法測試較大範圍的資料，因此在本節中我們將單獨測試 MCP 演算法在處理較大型資料檔的執行效率。我們採用資料檔的參數除了將 P 改為 10000 及 L 設定為 10 之外，其餘參數仍如表 4.1所示；產生之後的每個資料檔約為 3.8MB，而總資料量約為 30MB。我們首先測試不同相似度對 MCP 演算法的影響，實驗結果顯示在表 4.6及圖 4.10中。從這個結果我們可以發現在資料量更大的情況下，相似度對於 MCP 演算法影響程度大致與資料量較小的結果類似。由於在小範圍資料的測試中，我們發現最小支持度門

檻值對於MCP演算法的確有影響，但因MCP演算法的執行時間不夠長，故不易觀察出此影響。在此我們還更進一步測試最小支持度門檻值在0%至5%對於大範圍資料的影響，其實驗結果顯示於圖4.11中。在大範圍資料的測試中，就可以發現最小支持度門檻值對於MCP演算法的效率有很明顯的影響。在門檻值為0%及1%時，執行結果大致符合先前的分析，即探勘候選樣式時花了大部份的時間，但是在多期探勘所增加的時間不多。比較值得討論的是在門檻值為4%及5%的情況時，探勘候選樣式所需的時間非常少，甚至還比在多期探勘(即3到9期)時對每一期所花的時間還少。另外，我們也可以發現MCP演算法在最小支持度門檻值設為5%時，在大資料檔的測試中所需的時間居然比小資料檔的測試(圖4.6)所花費的時間還要少，這是因為在大資料檔中符合門檻值的項目非常少，以致於很多項目都被略去了，所以能夠很快速地探勘出候選樣式及建立樣式樹；同時也可以發現大資料檔因為每筆交易所含的項目較多，也使得更新樣式樹的時間因而增加。

表 4.6: 大範圍資料中相似度對於MCP演算法的執行時間結果表(時間單位:秒)

相似度	MCP
0.4	502.391
0.5	453.891
0.6	398.016
0.7	340.672
0.8	294.578

圖 4.10:驗證大範圍資料中相似度對MCP演算法的影響

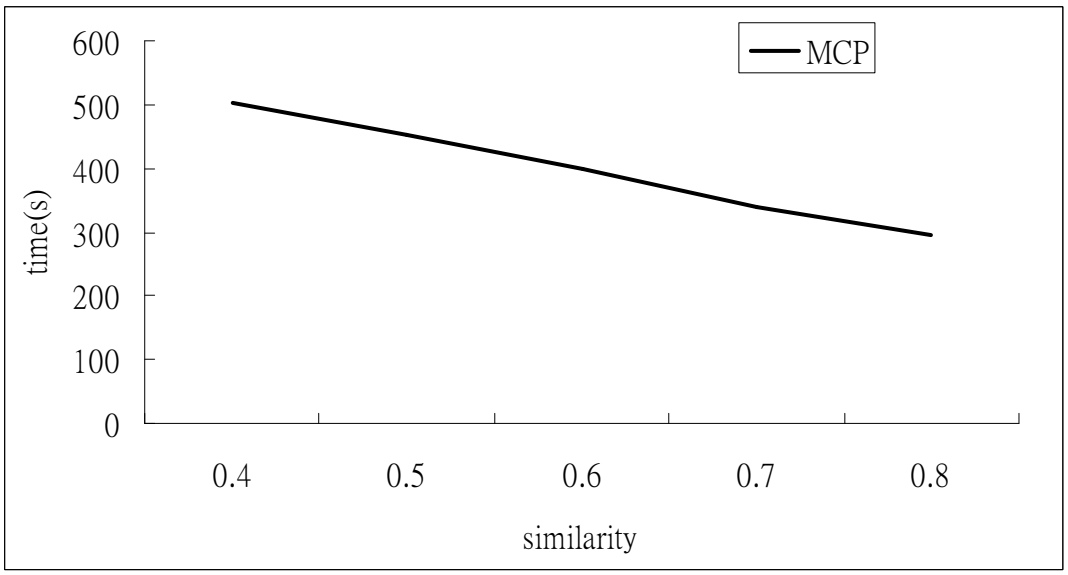
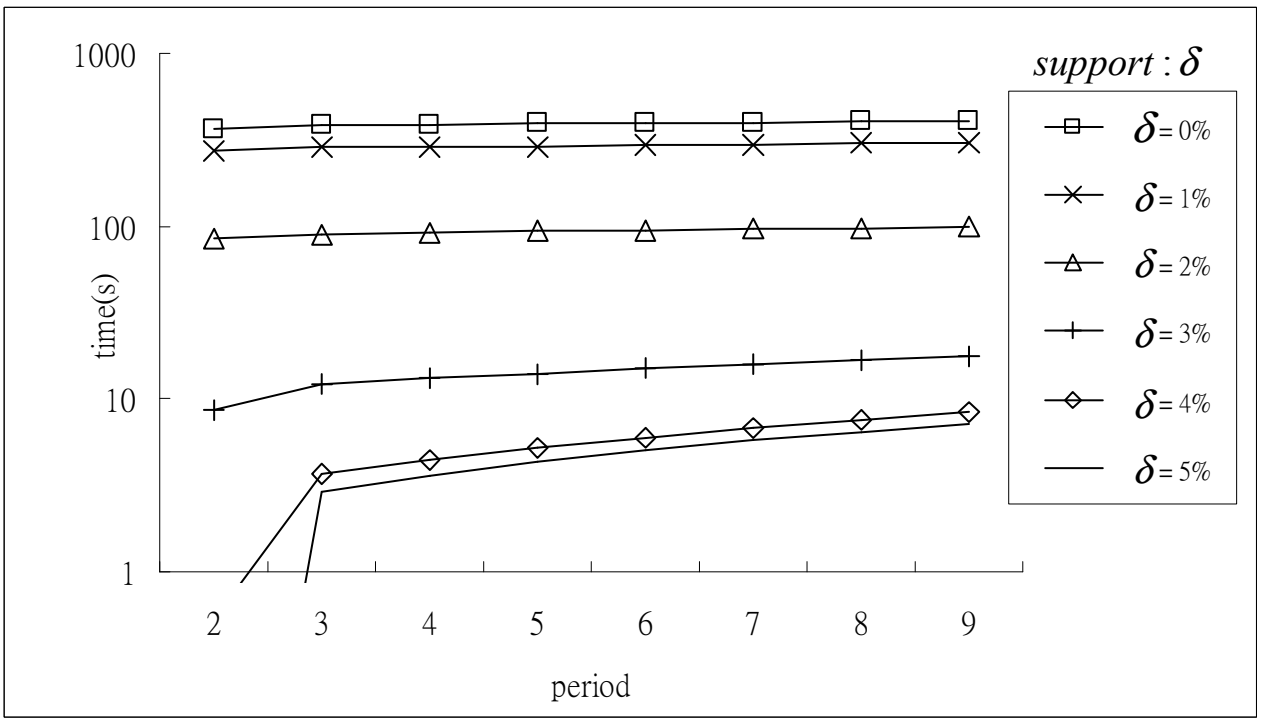


圖 4.11:驗證大範圍資料中支持度對MCP演算法的影響



4.4 小結

本章實作MCP演算法，同時為了比較執行的效率，本研究亦將Han et al.(2004)的FP-growth演算法修改成為一個可以探勘樣式在多期資料集合之變化的新演算法modiFP。我們利用虛擬的資料產生器提供測試的資料，為了使得測試資料能夠更貼近真實資料，我們加入了相似度(Li et al., 2006)的考量。為了測試演算法，我們進行了期數、相似度、最小支持度門檻值與變化率門檻值的驗證。根據測試及分析的結果，我們相信MCP演算法相當適合探勘多期的樣式變化。

第五章

結論與建議

5.1 結論

過去在資料探勘中關於樣式發現的研究多注重在發現頻繁樣式，以提供探勘者在資料庫中某些出現頻率很高的事件。然而頻繁樣式探勘的演算法皆是處理單一資料集合，無法觀察探勘出的樣式在不同時間點的變化情形。也就是說，探勘單一資料集合的演算法並沒有考慮到某些資料是具有時間因素的，因此當探勘者在得知上一期資料檔中所發現的樣式之後，即使做了些決策上的調整，仍無法進一步瞭解決策調整的成效；此時即需要連續地探勘多期資料檔，以觀察出事件的變化。假使我們能夠獲得連續多個時間單位的資訊，也就能從中發現一些趨勢的變化。

換個觀點來說，我們也期望能由那些頻繁樣式挖掘出有用的相關資訊。舉例來說，在單一資料集合的探勘中，不頻繁樣式的價值很低，甚至被視為雜訊；但是當某些不頻繁樣式連續地在數個期間的資料集合中都出現，則那些樣式不應再被視為毫無意義。以商業交易的觀點而言，那些不頻繁樣式有可能代表著某些小眾市場，或是某些消費偏好特殊但再購率很高的顧客。若能找出這些顧客，就可能進一步開發新市場，創造新的利潤來源。

文獻中有關變化探勘的方法，大多只是交叉比對兩個資料集合的差異，至於多個資料集合的差異則可能要重複進行兩期資料間的兩兩比對。本研究討論處理多個資料集合的變化探勘問題，為了避免遺失資訊及花費不必要的時間，我們說明了探勘的程序應該由後面的時間點往前進行；另外，為了在不遺失資訊的情

況下可以找出所有的變化樣式，應該在支持度為零的情況下進行探勘，但是如此一來將花費大量的執行時間，因此有必要發展出有效率的演算法。

為了提供探勘者不同於傳統頻繁樣式的資訊，我們發展出一套有效率且能同時考量時間因素與變化的樣式發現演算法，並將之命名為MCP演算法。MCP演算法包含三個部份，首先，演算法將建立原始搜尋樹以儲存最後兩個時間點的資料集合，並以該原始搜尋樹做為探勘樣式的基礎；接著，演算法將從原始搜尋樹中擷取出項目集合並經過諸如項目比對取其交集等等特殊程序的處理以找出所有的候選樣式，並同時為所有的候選樣式建立一個候選樣式森林(Candidate Pattern Forest，CP-forest)，走訪CP-forest的每棵樣式樹即可發現每一個項目的所有變化樣式；最後，只要針對其它期的資料檔逐一地進行更新各棵樣式樹及修剪各棵樣式樹，並再一次走訪樣式樹即可探勘出另一期的變化樣式。

本研究也實作了MCP演算法，同時為了比較演算法的效率，我們另外修改著名的探勘頻繁樣式演算法FP-growth而設計了modiFP演算法，以探勘多期的變化樣式。為了提供測試之用的資料，我們也修改了原本只能產生單一資料檔的交易資料產生器。新的交易資料產生器不但可以產生多期的資料檔，還可以根據探勘環境的需求，為連續兩期的資料檔設定某種程度的相似度。

從實驗結果可知，MCP演算法相當適合進行多期的資料探勘。而當不同期間的資料檔具有愈高的相似度時，MCP演算法的執行效率愈好。雖然最小支度門檻值是影響樣式發現演算法效率的關鍵，但是因為MCP演算法的設計方式比較特殊，使得此門檻值的影響成效沒有如modiFP般顯著。而樣式變化率的高低僅對MCP演算法在更新樣式樹時的部份有所影響，對於探勘候選樣式及建立樣式樹則沒有影響。從各種實驗結果可知，MCP演算法在多期探勘時的效率遠比modiFP為佳，特別在探勘的期數愈多時此優勢將愈明顯。為了增加探勘變化樣式的彈性，MCP演算法可以由使用者定義不同的變化程度。

5.2 後續研究建議

在實驗的過程中，我們發現MCP演算法中最耗費時間的程序為開始的探勘候選變化樣式與建立樣式樹。雖然最小支持度門檻值愈高與變化率的設定程度愈

極端應會導致MCP演算法的整體執行時間縮短，但是實驗結果卻顯示MCP演算法受這兩個門檻值的影響不若modiFP明顯。因此，若欲進一步地提升MCP演算法效率的話，可能必須改善探勘候選樣式的方法。MCP演算法能快速地探勘多期資料集合，是因為其建構了樣式森林。然而，當項目集合及其每個子集合都是候選樣式時，MCP演算法仍會耗費許多儲存空間且其執行效率也會變得很差。

雖然MCP演算法一開始是以探勘多期資料集合為目的而被設計出來的，但是我們發現MCP演算法其實也是可以被用來探勘單一資料集合的頻繁樣式，其作法如下：在建立原始搜尋樹時，僅需加入一期的資料即可；之後的探勘候選樣式及建立樣式樹作法與MCP演算法的原作法相同，而最後只要進行樣式樹的走訪，即可探勘出頻繁樣式。這種以MCP演算法來探勘頻繁樣式的作法由於需要建立樣式樹，故將會佔用較多的記憶體空間。

由另一個觀點來看，由於MCP演算法的目的在於處理多個資料集合，亦即將原本大資料檔按照時間切割成多個較小的資料檔，因此其所處理的各個期間的資料檔其實都不大。所以MCP演算法能快速地連續探勘多個資料檔。然而，由於探勘頻繁樣式的目的在於從大量的資料中發現出現次數較高且較具代表性的樣式；因此探勘頻繁樣式所使用的資料檔通常都不小，而此時若用MCP演算法來探勘頻繁樣式，將較難突顯其優勢。由此可知，MCP演算法雖然可以被用來探勘頻繁樣式，但卻不見得比傳統探勘頻繁樣式的演算法來得有效率。

參 考 文 獻

Agrawal, R. and Srikant, R. Fast algorithm for mining association rules in large databases. *In Proceedings of 20th International Conference on Very Large Data Bases*, 487-499, 1994.

Agrawal, R. and Srikant, R. Mining sequential patterns. *Proceedings of the Eleventh International Conference on Data Engineering*, 3-14, 1995.

Ale, J. K. and Rossi, G. H. An approach to discovering temporal association rules. *In Proceedings of the 2000 ACM Symposium on Applied Computing(SAC'00)*, 294-300, 2000.

Antunes, C. and Oliveira, A. Temporal data mining: an overview. *Workshop on Temporal Data Mining(KDD2001)*, 2001.

Bailey, J., Manoukian, T. and Ramamohanarao, K. Fast algorithms for mining emerging patterns. *Lecture Notes in Computer Science*, **2431**, 39-50, 2002.

Bay, S. D. and Pazzani, M. J. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, **5**, 213-246, 2001.

Bayardo, R. J. Efficiently mining long patterns from databases. *In Proceedings of the ACM-SIGMOD international conference on management of data*, 85-93, 1998.

Berry, M. J. and Linoff, G. *Data mining techniques: For marketing, sales, and customer support.* John Wiley and Sons, 1997.

Brin, S., Motwani, R., Ullman, J. D. and Tsur, S. Dynamic itemset counting and implication rules for market basket data. *SIGMOD Record*, **26**(2), 255-276, 1997.

Ceglar, A. and Roddick, J. F. Association mining. *ACM Computing Surveys*, **38**(2), 1-42, 2006.

Dong, G. and Li, J. Mining border descriptions of emerging patterns from dataset paris. *Knowledge and Information Systems*, **8**, 178-202, 2004.

Dong, G., Zhang, X., Wong, L. and Li, J. *Caep : Classification by aggregating emerging patterns.* Tokyo, Japan: In:Proceedings of teh 2nd international conference on discovery science, 1999.

Grahne, G. and Zhu, J. Fast algorithms for frequent itemset mining using fp-trees. *IEEE Transactions on Knowledge and Data Engineering*, **17**(10), 1347-1362, 2005.

Han, J., Pei, Y., Yin, Y. and Mao, R. Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, **8**(2004), 2004.

Kantardzic, M. *Data mining:concepts, models, methods, and algorithms.* WILEY-INTERSCIENCE, 2003.

Keogh, E., Chakrabarti, K., Pazzani, M. and S., M. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, **3**(3), 263-286, 2001.

Kim, J. K., Song, H. S., Kim, T. S. and Kim, H. K. Detecting the change of customer behavior based on decision tree analysis. *Expert Systems*, **22**(4), 193-205, 2005.

Li, T., Zhu, S., Wang, X. S. and Jajodia, S. Looking into the seeds of time: Discovering temporal patterns in large transation sets. *Information Sciences*, **176**, 1003-1031, 2006.

Li, Y., Ning, P., Wang, X. S. and Jajodia, S. Discovering calendar-based temporal association rules. *Data and knowledge Engineering*, **44**(2), 193-218, 2002.

Liu, B., Hsu, W., Han, H. S. and Xia, Y. Mining changes for real-life applications. *Second International Conference on Data Warehousing and Knowledge Discovery*, 337–346, 2000.

Liu, B., Hsu, W., Mun, L. and Lee, H. Y. Finding interesting patterns using user expectations. *IEEE Transactions on Knowledge and Data Engineering*, **11**, 817-832, 1999.

Ozden, S., Ramaswamy, S. and A., S. Cyclic association rules. *In Proceedings of the 14th International Conference on Data Engineering(ICDE'98)*, 412-421, 1998.

Padmanabhan, B. and Tuzhilin, A. Unexpectedness as a measure on interstingness in knowledge discovery. *Decision Support Systems*, **27**, 303-318, 1999.

Park, J. S., Chen, M. S. and Yu, P. S. Using a hash-based method with transaction trimming and database scan reduction for mining association rules. *IEEE Transactions on Knowledge and Data Engineering*, **9**(5), 813-825, 1997.

Pei, J., Han, J., Chen, Q., Dayal, U. and Hsu, M. C. Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. *Proceedings of the 17th International Conference on Data Enigneering*, 215-224, 2001.

Quinlan, R. C4.5: program for machine learning. *Morgan Kaufmann*, 1992.

Roddick, J. F. and Spiliopoulou, M. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, **14**(4), 750-767, 2002.

Savasere, A., Omiecinski, E. and Navathe, S. *An efficient algorithm for mining association rules in large databases.* Zurich, Switzerland: In proceedings of the 21st International Conference on Very Large Data Bases, 1995.

Song, H. S., Kim, J. K. and Kim, S. H. Mining the change of customer behavior in an internet shopping mall. *Expert Systems with Application*, **21**(3), 157-168, 2001.

Suzuki, E. and Zytkow, J. M. Unified algorithm for undirected discovery of execption rules. *International Journal of Intelligent Systems*, **20**, 673-691, 2005.

Tung, A., Lu, H., Han, J. and Feng, L. Efficient mining of intertransaction association rules. *IEEE Transactions on Knowldege and Data Engineering*, **15**(1), 43-56, 2003.

Webb, G. I., Pazzani, M. and Billsus, D. Machine learning for user modeling. *User modeling and User-Adapted Interaction*, **11**, 19-29, 2001.