關聯法則探勘演算法之研究與分析

蔡賢亮
義守大學
資訊管理學系
助理教授
jim@isu.edu.tw

鄭永雄
義守大學
資訊管理學系
M9322029@stmail.isu.edu.tw

蔡佩君
義守大學
資訊管理學系
m9522005@stmail.isu.edu.tw

周煜書
義守大學
資訊管理學系
m9522021@stmail.isu.edu.tw

摘要

近年來有關資料探勘(Data Mining)的技術已成為相當熱門的研究議題之一，目的是從資料庫中挖掘出有意義的資訊，並提供給管理者做決策。以往的相關研究，不少演算法被提出改善關聯法則的效能；這些演算法各有各的優點，但對於選擇何種探勘關聯法則才會有最大效益所產生困擾，因此必需考慮關聯法則探勘技術的適用性，以期選擇較有效率探勘來擷取有用的資訊。本研究探討五個目前較著名資料探勘演算法效率分析，我們用這五個演算法各別去處理若干具不同特性的資料庫，由實驗數據加以分析與探討，並整理出他們各適用於何種特性的資料庫，最後我們亦將他們歸納出具何種特性之資料庫較適用那一個資料探勘演算法，並且希望能建議使用者能選擇較有效率的資料探勘演算法。

關鍵詞：Apriori 演算法、FP-growth 演算法、DHP 演算法、DIC 演算法、LCM-freq 演算法。

Abstract

In recent years, the techniques of data mining have already become one of the popular research subjects. Its purpose is to mining meaningful information from the database, and provides it to the administrator for decision making. In past relevant research, many algorithms were proposed to improve the effect of association rule currently. These algorithms each have their own advantages, it will be difficult to determine which algorithm provides the best effect, therefore we must consider the applicability of the association rule of data mining algorithm in order to mine data more effectively and obtain useful information. Our research inquires into presently five association rule algorithms, and uses them individually to process several real databases. And then analyze these experiment data to see each algorithm's pros and cons and its applicable type of database characteristics. Finally, we wish to suggest the users use more effective association rules of data mining algorithm.

Keywords ： Apriori algorithm, FP-growth algorithm, DHP algorithm, DIC algorithm, LCM-freq algorithm.

1. 前言

隨著資訊科技與網際網路的蓬勃發展，各種資料的數量也越來越多，企業如何有效地從這些資料得知有用的資訊，以利用這些資訊作為決策參考；因而有資料探勘(Data mining)技術產生[2,3,13,14]。資料探勘結合演算法及各種統計方法分析資料，從大型資料庫中挖掘出有用的資訊與知識，以提供未來決策支援與預測。

目前已有不少的學者相繼投入此研究，使得資料探勘技術已經可處理不同的資料種類，且有許多探勘演算法已被提出。但演算法執行的效率評估，是由當者提出改良的或新的演算法與相關研究作比較，並無現存之演算法做總體的比較。

有鑑於此，本研究對目前現存的關聯法則探勘技術做深入的分析與探討，針對密集(dense)、稀疏(sparse)或具不同長度頻繁項目之資料庫與現存技術之間做效能分析，並以電腦模擬與現實中之資料庫進行實際的資料探勘，且評估各個方法在不同資料庫的效率、數量與時間等之差異並做一比較與討論。我們將依據資料的特性，建議較有效能之關聯法則探勘技術。

2. 關聯法則探勘演算法簡介

本研究主要目的將對目前現存的關聯法則探勘演算法做分析與探討，因此著眼於探討五個資料探勘演算法包含 Apriori 演算法、DHP 演算法、DIC 演算法、FP-growth 演算法和 LCM-freq 演算法。

表 1 四種關聯法則探

演算法	特性	缺點	搜尋次數
Apriori	(1)找出所有高頻項目集合	(1)產生過多的候選項目集合 (2)重覆掃瞄資料庫	需多次，直到無法產生後候選項目集合
DHP	(1)減少C_2的產生 (2)有效地產生高頻項目集。	(1)需花費額外的空間儲存Hash Table (2)分解交易資料，耗費大量記憶體空間、時間	需多次，與Apriori演算法相同
DIC	(1)縮減資料庫搜尋次數	(1)需尋找最佳切割 (2)受資料分佈的影響，在特定之下才有效縮減資料庫掃瞄次數	最差與Apriori演算法相同
FP-growth	(1)不用產生候選項目集合產生 (2)減少在資料庫的搜尋次數 (3)壓縮資料庫容量，轉為FP-tree	(1)FP-tree太大將無法全部置入主記憶體 (2)當自訂的門檻異動時，FP-tree必須重建	需二次
LCM-freq	(1)壓縮資料庫容量，依陣列儲存 (2)使用Backtracking方法，遞迴搜尋頻繁項目集合	(1)1-itemset過多，將增加記憶體需求，執行效能降低	資料庫搜尋為一次

Apriori 演算法：此演算法在 1994 年由 IBM Almaden Research Center 的 Agrawal 等人[1]設計，它有一個很重要的定理：若項目集合 I 不是一個頻繁項目集合，則任何包含項目集合 I 的項目集合絕對不是頻繁項目集合，此定理稱之為 Apriori 定理；而且利用一種逐層搜尋的疊代方法(level-wise search)產生候選項目集合，再去檢測每個候選項目集合；反言之，產生項目長度為 k 的頻繁項目集合後，再由此項目長度為 k 的頻繁項目集合產生項目長度為 k+1 的候選項目集合，直到沒有新的頻繁項目集合產生為止。

為有效且避免重覆產生候選項目集合，候選項目集合產生的程序是由連接(joint)與修剪(prune)二個步驟所組成，連接步驟乃將任二個項目長度為 k 的頻繁項目集合連接成項目長度為 k+1 的新項目集合；而修剪步驟乃根據 Apriori 定理，將連接步驟所產生的新項目集合不符合候選項目集合資格的項目集合予以修剪。

DHP 演算法：J. S. Park et al.於 1995 年提出了 DHP (Direct Hashing Pruning)演算法[5]，其主要的概念是改進 Apriori 演算法在L_1兩兩聯集，產生C_2時，會產生過多的C_2，DHP 演算法利用 Hash Table 刪除不必要的C_2，減少掃瞄資料庫的存取成本，雖然 DHP 在一開始需耗費額外的時間建立 Hash Table，但有助於資料庫掃瞄次數減少，達到效能的提升。利用每個 bucket 中所記錄的數量來篩選出候選項目集，其優點是可以刪除大部分非相關的項目集合，但此方法有時也會使得某些項目集合的實際出現次數被高估，進而使其通過最小支持度的篩選成為候選項目集合。

DIC 演算法：S. Brin、R. Motwani、J. Ullman 與 S. Tsur. 在 1997 年提出 DIC(Dynamic Itemset Counting)演算法[9]，其主要的概念是亦由 Apriori 演算法為基礎加以改善，把 itemsets 存入 lattice 中並分割交易區段來達到儘早決定 large itemsets，其目的減少花費在搜尋資料庫的時間。其作法仍與 Apriori 演算法雷同，不同的地方在於將資料庫分成多個區段，在處理完一個區段後，就會依據 Solid box、Solid circle、Dashed box 與 Dashed circle 四個條件來判斷至此階段為止的所有候選項目集合中是否有滿足最小支持值條件者，就此區段中產生候選項目集合，而不像 Apriori 演

算法判斷一個新的候選項目集合必須從頭去搜尋完整的資料庫。

FP-growth 演算法：Jiawei Han、Jian Pei 及 Yiwen Yin 於 2000 年提出提出一個不需產生候選項目集合的探勘方法[4]，稱之為 Frequent-pattern growth(簡稱 FP-growth)，FP-growth 演算法首度提出 Frequent Pattern tree(簡稱 FP-tree)的資料結構，可有效地壓縮交易資料庫，因此，一般通稱此方法為 FP-tree；FP-growth 演算法最主要的概念，僅需掃瞄資料庫兩次，第一次掃瞄找出長度為 1 符合最小支持度的頻繁項目集合，依出現的次數由大到小排序，建立 Header table。再第二次掃瞄資料庫依據 Header table 刪除資料庫中低於最小支持度的項目集合，依據 Header table 的順序，排序每筆交易的頻繁項目集合，再依據這些頻繁項目集合建構 FP-tree，一一由 FP-tree 上的每條路徑上的節點產生 Conditional pattern base，依據這些 Conditional pattern base 建構出 Conditional FP-tree 後，即 mining Frequent-Pattern tree 找出頻繁項目集合。

LCM-freq演算法：由Takeaki Uno et al.學者於2003提出[10]，在技術上其使用簡單陣列技術壓縮資料庫，不像其他演算法壓縮資料庫亦使用binary tree之方法；這個演算法架構於backtracking方法之下，以及運用有效率的頻繁項目次數計算方法。其backtracking方法是由depth-first變化衍生而來的[4,6,7]，而尋找頻繁項目集合的定義為以樹狀之parent-child關係去定義頻繁項目集合；所以，LCM-freq演算法主要在於遞迴方式搜尋頻繁項目集合，其重覆置入頻繁項目集合P，因而產生尾端P之延伸集合，若延伸集合為頻繁項目集合，將重覆產生延伸集合之遞迴方法。

介紹了以上若干個探勘關聯法則相關的演算法之後，我們從文獻整理關聯法則探勘演算法的特性，讓我們知道對於關聯法則探勘演算法有初步的了解，此階段讓人加強對關聯法則探勘演算法的應用，表1係探討演算法之間的特性。

3.執行方法

本研究進行一系列的實驗以評估前述定理之有效度；實驗的環境為 windows xp professional，硬體為 AMD athlon64 Processor 3000+1.8GHz CPU 及配備 256 MB DDR-RAM，此外程式是以 Visual C++ 6.0 實作 Apriori 演算法、FP-growth 演算法、DIC 演算法、DHP 演算法與 LCM-freq 演算法，產生測試資料程式和結果。

表 2 資料庫之參數定義

參數	參數定義
T	Average size of the transactions
I	Average size of transactions of maximal potentially large itemsets
D	Transaction database
N	Number of items

表 3 資料庫設定

Dataset	T	D	N
T6I4D10	6	10K	959
T10I4D10	10	10K	980
T6I4D100	6	100k	969
T10I4D100	10	100k	969
T10I8D100	10	100k	985
chess	37	3,196	76
connect	43	67,557	130
mushroom	23	8,124	120
pumsb*	50.5	49,046	7,117
pumsb	74	49,046	7,117

在此章節中我們針對 Apriori 演算法、FP-growth 演算法、DHP 演算法、DIC 演算法與 LCM-ferq 演算法進行實驗，並評估各個演算法在探勘資料庫的效率、數量與時間等之差異。首先選擇若干個不同特性之資料庫，其使用的資料庫可分不同特性來分析探討。不同特性之資料庫可區分為：真實資料庫與合成資料庫以及密集資料庫與稀疏資料庫。本研究所使用的資料庫分為合成資料庫與真實資料庫。合成資料庫是由 IBM 提供的 database generator 所產生的[11]；真實資料庫則從 FIMI'03 workshop web site 與 UC-Irvine Machine Learning Database Repository[12]所取得；因此設定相關參數，以產生不同性質的交易資料庫，其資料庫如表 2、3。我們接著進行各個關聯法則演算法探勘這些資料庫，並評估各個演算法在探勘資料庫的效率、數量與時間等之差異並做一比較與討論，其所得的分析結果進

一步的整理。

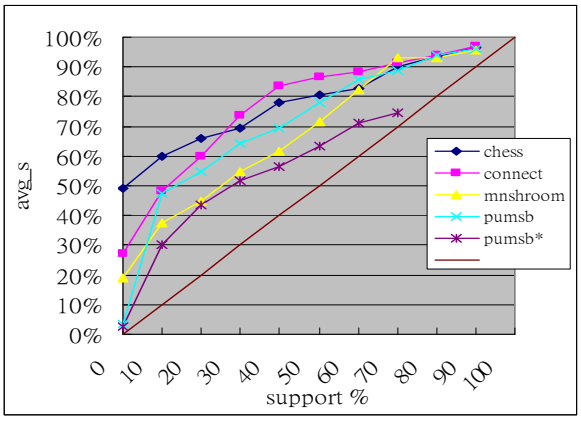


圖 1 密集資料庫之平均支持度

我們所使用的密集的資料庫與稀疏的資料庫是根據[8]之定義去分類，他們使用定性分析來確定兩種不同的資料庫，藉由頻繁項目的平均支持度分類。如圖 1、圖 2 所展示，沿著 X 軸觀看，一個密集 datasets 在支持度門檻值較高的地方，它的頻繁項目之平均支持度將比門檻值高，所呈現的曲線都遠遠超過 Y=X 線。另一方面，稀疏 datasets 只能從門檻植較低的地方觀看，它的頻繁項目之平均支持度只有稍微高於門檻值。因此，密集 datasets 以頻繁項目之平均支持度只有稍微高於門檻值。因此，密集 datasets 以頻繁項目之平均支持度對比所有最小支持度門檻值較高的為特點。所以把不同情況之分類顯示，在[8]的定義其的知圖 1 之中的 chess、musbroom、pumsb、pumsb* 與 connect 這些資料庫屬於密集資料庫，而圖 2 得知合成資料庫都屬於稀疏資料庫。

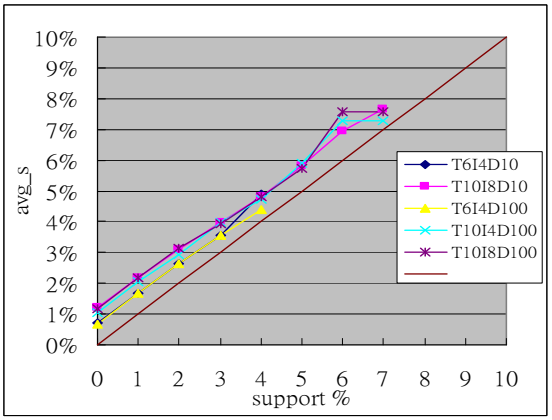


圖 2 稀疏資料庫之平均支持度

3.1 Apriori 演算法效能分析

在此我們依定理 $N^2 \times P \times D$ 來分析 Apriori 演算法，N 代表將要通過 minimum support 之長度為 1 的候選集合項目的個數，P 為每個資料庫之平均交易長度，D 則是代表資料庫的大小；這些會是影響演算法執行時間之因素。我們把目前資料庫套入這個定理，其密集資料庫為 minimum support=75%與稀疏資料庫為 minimum support=1.5%進行分析，詳細記錄這些資料庫執行資料為表 4、表 5。表 4 我們發現密集資料庫 mushroom 的資料計算量最少，pumsb 則耗費最多，由圖 3 顯示出資料計算量多寡使其執行效率亦受影響，另外，在執行 connect、pumsb 兩個資料庫，因執行記憶體耗盡故無法顯示執行效率；其表 5 顯示稀疏資料庫 T6I4D10 的資料計算量在其中為最少，而 T10I8D100 最多，圖 4 也顯示相同結果。本研究發現 Apriori 演算法在資料密集度越稀疏時資料計算量越少，適用於探勘較小與稀疏之資料庫。

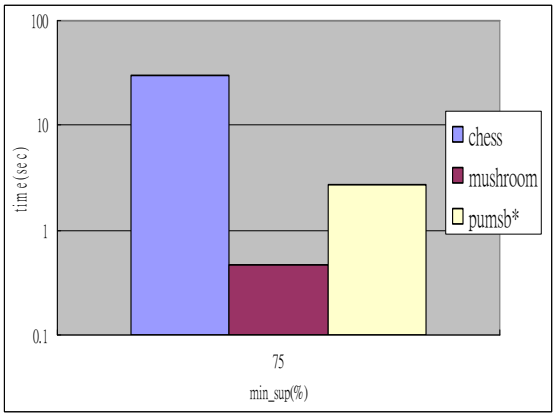


圖 3 密集資料庫執行時間

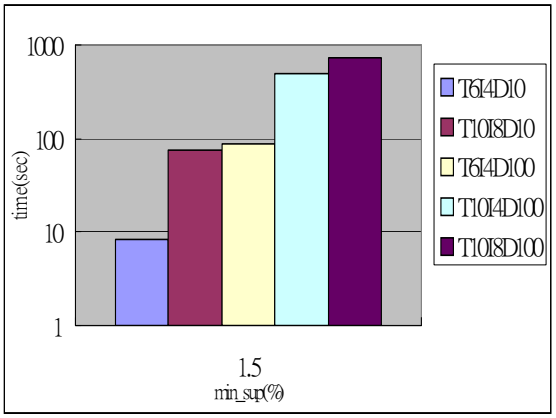


圖 4 稀疏資料庫執行時間

表 4 密集資料庫(min support = 75%)

	N	P	D	N^2*P*D
chess	23	37	3196	≒ 2.7*10^6
connect	30	43	67556	≒ 8.7*10^7
mushroom	5	23	8214	≒ 9.4*10^5
pumsb	27	74	49046	≒ 9.8*10^7
pumsb*	2	50.5	49046	≒ 4.9*10^6

表 5 稀疏資料庫(min support = 1.5%)

	N	P	D	N^2*P*D
T6I4D10	117	6.18	10K	≒ 7.2*10^6
T10I8D10	276	10.53	10K	≒ 7.6*10^9
T6I4D100	119	5.67	100K	≒ 6.7*10^7
T10I4D100	237	9.63	100K	≒ 5.4*10^10
T10I8D100	276	10.46	100K	≒ 7.9*10^10

3.2 FP-growth 演算法效能分析

在此章節我們針對 FP-growth 演算法進行實驗，我們發現 FP-tree mining 亦受到 minimum support 與記憶體容量影響；且 FP-tree 的大小會受到資料的屬性不同而有差異，在密集資料庫越密集則它的 pattern 會越相似，則 FP-tree 的樹狀越緊密；反之，越稀疏的資料庫之 FP-tree 的樹狀越分散。表 6、表 7 記錄的是 connect 與 pumsb 資料庫的 FP-tree 之節點與分支；從這些可以明確知道密集度不同，則產生的 FP-tree 會有差異；connect 資料庫交易量大，但產生的分支與節點比 pumsb 資料庫少，這是因為它的密集度比其它的更密集，因而有效率壓縮資料庫。因此，資料密集度亦影響 FP-tree 之節點與分支緊密或分散。

表 6 connect 資料庫

資料庫	connect		
min_sup	node	path	tree-node
90	21	32	224
80	28	121	655
70	31	207	1081
60	35	991	3453
50	38	1677	5715

表 7 pumsb 資料庫

資料庫	pumsb		
min_sup	node	path	tree-node
90	20	429	2468
80	25	953	5007
70	34	2523	19244
60	39	5409	35590
50	52	10965	108194

此外，由圖 5 顯示 FP-tree 的大小間接影響 mining 的效率。本研究發現 FP-growth 演算法受到資料庫大小影響不大；但建立 FP-tree 的時間還是會因資料量的大小而影響到時間成本。

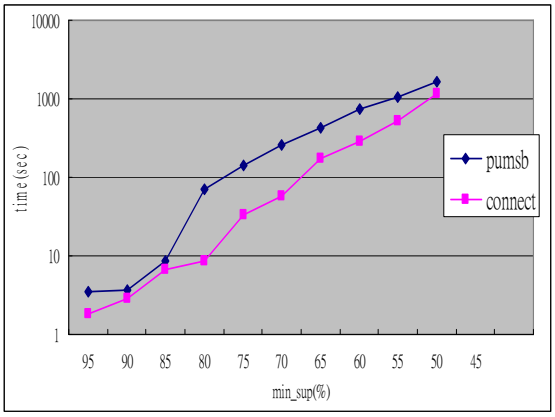


圖 5 FP-growth 演算法執行時間

3.3 DIC 演算法效能分析

文獻探討中，我們了解 DIC 演算法是利用分割資料庫區段以達到儘早決定 large itemsets，以減少掃描資料庫的次數。而 DIC 演算法分割區段 M 的大小必須相同，另外資料庫掃描的次數也隨著分割區段的不同而改變，所本小節將針對 M 的設定進行分析；M 的範圍大小設定為 100 至 10,000。所以我們針對資料庫大小進行 M 的參數設定；若資料庫交易記錄小於 10,000 筆，則參數比較為 M=100、M=500 與 M=1,000；若資料庫交易記錄大於 10,000 筆，則參數比較為 M=100、M=1,000、M=5,000 與 M=10,000。

我們進行資料庫小於 10,000 筆交易紀錄之實驗，在圖 6 可知不同區段的執行效果，且發現執行效率相差甚小，但在 M=500 的效率較佳；因此在小於 10,000 筆交易記錄之設定參數可選擇 M=500。

進行實驗大於 10,000 筆交易紀錄之資料庫，圖 7 顯示 T10I8D100 資料庫執行時間，在實驗中發現 M=100 與 M=10,000 執行時間為較差，較好的參數為 M=1,000 與 M=5,000；

因此在大於 10,000 筆交易記錄設定參數可選擇 M=1,000 或 M=5,000。

最後，我們知道 DIC 演算法與區段的大小有關係，區段劃分太小時，將會耗費太多的成本在計算上，因而耗費太多 CPU 時間，相反地，當區段劃分太大時，將無法顯現動態計算支持度的好處，因而耗費太多時間在搜尋資料庫上，因此，區段分的太大或太小都不好，可能都花費額外的 CPU 或 I/O 時間；目前在切割區段的分析中，我們只能由實驗結果去評估在何種區段範圍內有最好效能。

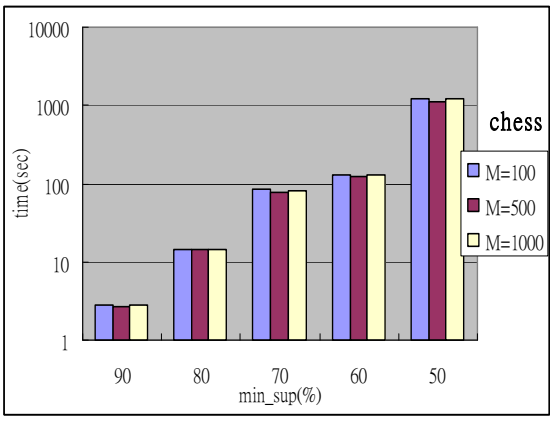


圖 6 DIC 演算法之 chess 執行時間

3.4 DHP 演算法效能分析

DHP 演算法是以 Apriori 演算法為基礎，並且以雜湊表之雜湊技術的架構有效率刪減候選項目集數量。若一個資料庫有 N 個項目，而 N=100，則在資料庫可以產生長度為 2 項目集合有 C_2^{100} =49,950 個的產生，所以設定 bucket 至少要有 2^n 大於接近 499,500，或者小於接近 49,950 之 2^{n-1}；因此我們針對 bucket 設定大小進行分析，設定 bucket 為 2^n、2^{n-1}、2^{n-2}、2^{n-3} 四個參數，進行實驗且分析效率差異。我們進行資料庫實驗；實驗結果顯示如下圖 8，它們效率有受到間接影響；然而圖 8 顯示以 2^{n-2} 的 buckets 參數為佳。觀看圖 9 之資料庫，四個不同 buckets 參數因 minimum support 降低，則顯示效率之差異更為明顯，其中 2^{n-3} 差異較大、較差，然而 2^{n-2} 為較佳。

最後，我們發現到資料庫在不同 buckets 參數，參數之間比較顯示有所差異；我們選擇 buckets 方面，原本應該是 2^n 為有最有效果，因為在刪除不相干候選項目集合為最多，但是實驗顯示為較差的執行結果；其可能因素是在執行演算法的過程，這是因為 DHP 演算法有個兩個階段，第一個階段在於使用 hash function 篩選候選項與縮減資料庫，第二階段與 Apriori 演算法相同；當 hash table 的值小於特定值時，將會跳出第一階段，其後使用第二階段，雖然 2^n 在刪除不相干候選項為最多，但可能提早進入使用第二階段，然而其它的 bucket 可能還在進行 hash function 階段，也因此造成這樣差距；根據我們的實驗結果，以 2^{n-2} 之 buckets 參數的執行效率表現良好。

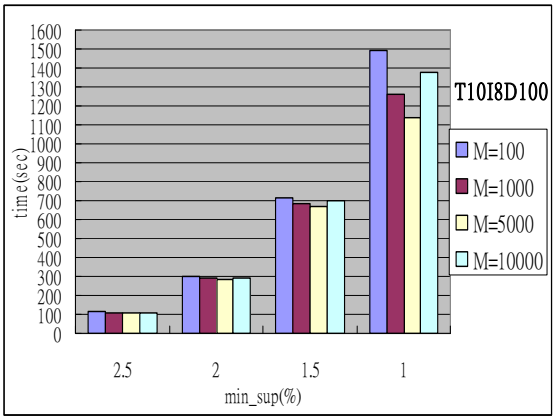


圖 7 DIC 演算法之 T10I8D100 執行時間

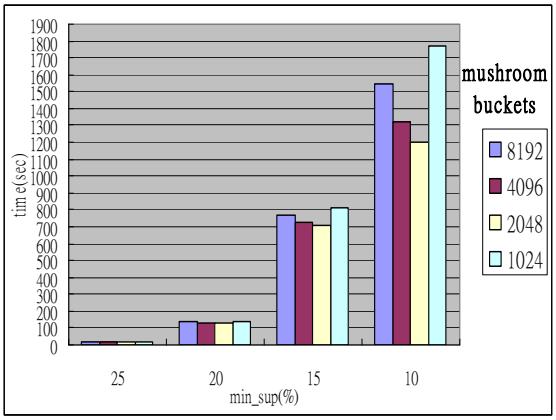


圖 8 DHP 演算法之 mushroom 資料庫

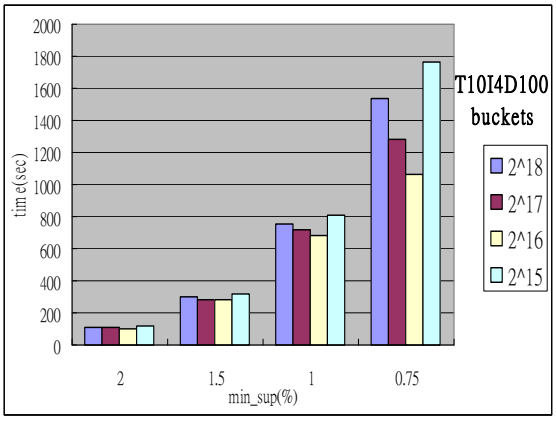


圖 9 DHP 演算法之 T10I4D100 資料庫

3.5 LCM-freq 演算法效能分析

LCM-freq 演算法是以 backtracking 演算法為基礎，使用遞迴方式找尋頻繁項目，換句話說兩項目集合做交集的，若新的集合為頻繁的，則將與下一個結合，直到非頻繁的項目集合或者到最後一個返回，這樣搜尋次數可能 2^N ，N 為頻繁項目的個數；舉例說，若有一個交易資料庫，有 5 個頻繁項目，則搜尋次數可能要 $2^5 = 32$ 次，因此我們針對 LCM-freq 演算法空間效益探討。在此進行 LCM-freq 演算法對資料庫實驗，並把詳細資料整理為表 8 及執行效率如下圖 10。

表 8 LCM-freq 演算法之密集資料庫

pumsb			Pumsb*		
min_sup	N	搜尋次數	min_sup	N	搜尋次數
95	13	547	70	9	139
90	20	9813	60	14	1154
85	24	80096	50	27	8684
80	25	597653	40	46	583678
75	27	936234	35	55	1086476

在表 8 發現 minimum support 漸小，N 的個數跟隨著增加，此搜尋頻率也增加成長，且是以倍數成長。則圖 10 顯示時間以 pumsb 為高，這是因為它的資料庫密集度高，頻繁的項目將較為集中，然而可發現長度較長頻繁的項目，使他的搜尋次數，時間也跟隨增長；最後，我們知道進行探勘密集資料庫，也可獲得得到頻繁項目集合，但是需要較大空間容量來支援搜尋頻繁項目。

我們針對稀疏資料庫進行實驗，並把詳細資料整理為表 9 以及執行效率如下圖 11。從表 9 也發現到 minimum support 漸小，同樣與表 8 相同結果；且在同樣資料量但不同的資料分佈，其顯示著搜尋頻繁項目集合的次數隨著 minimum support 減少而有極大差異。最後，我們知道 LCM-freq 演算法探勘稀疏資料庫，也可較快搜尋頻繁項目集合，但在 minimum supprot 漸小，卻需要較大空間換取執行效率。

4.研究結果

首先對密集資料庫進行實驗，並顯示各個演算法執行時間如下圖12。從圖121發現到在我們有限的資料庫中，各個實驗結果皆顯示出FP-growth演算法與LCM-freq演算法有不錯執行效果，尤其當mining support逐漸降低時，比其他的演算法有較好執行效果。FP-growth演算法與LCM-freq演算法在各自分析中，他們以壓縮資料庫方式降低資料搜尋量，而且他們遞迴方法有效率的探勘頻繁項目集合，因此我們可以選擇此演算法去進行探勘關聯法則。然而Apriori演算法在mining support較高時，才有比其他演算法稍微較好的表現；此外，我們發現DIC演算法的執行效果與Apriori演算法相差不大，在各自實驗分析得知，此演算法受到切割區段大小影響，並且與資料分佈有相關的影響，因此導致在實驗中有這樣的結果；

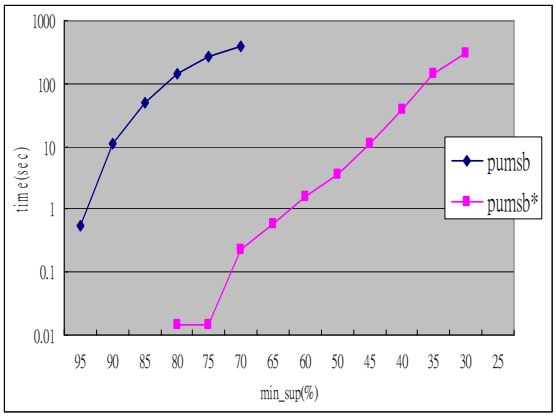


圖 10 密集資料庫執行時間

表 9 LCM-freq 演算法之稀疏資料庫

	T6I4D100		T10I8D100	
min_sup	N	搜尋次數	N	搜尋次數
2	47	1275	185	16110
1.5	119	6903	276	38226
1	235	27730	413	81406
0.75	317	52975	518	132870
0.5	465	109278	648	206167

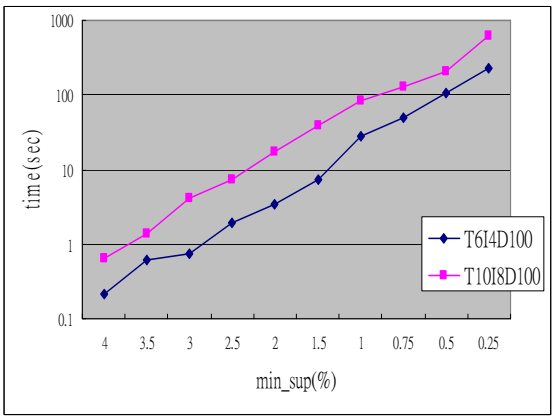


圖 11 稀疏資料庫執行時間

DHP演算法在實驗分析得知，此演算法受到hash table之buckets的影響，一開始在執行探勘

皆需進行長度為2項目集合之hash function，所以不管mining support高低都要耗費一些成本，但也減少資料庫量與候選項產生，使其mining support較低時些微差異。

最後，我們分析整理把結果呈現如下表10，顯示實驗結果分析之後獲得的資訊，說明各個演算法在何種特性資料庫之下可獲較好的執行效果，希望能提供給使用者探勘相關的資料，提供研究結果讓使用者知道各演算法探勘對於不同特性的資料庫之效能，進而提供使用者選擇關聯演算法探勘的建議。

表10 演算法適用資料庫

演算法	平均長度	資料庫密集度	資料庫大小
Apriori	短	稀疏	小
DIC	短	稀疏	區段設定影響
DHP	短	稀疏	小
FP-growth	無影響	密集	無影響
LCM-freq	無影響	稀疏	無影響

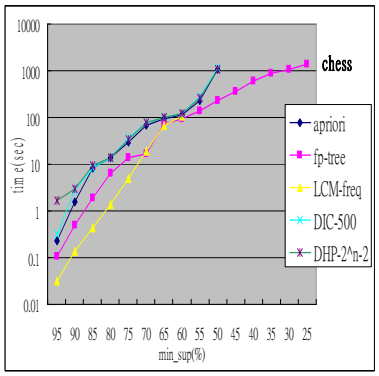
5.結論與未來研究

對於關聯法則探勘技術持續改良而言，這些主要目的皆是為了減少成本支出以及快速獲取資訊，而且各種演算法都有適用的資料庫，所以本研究進行實驗並且分析與探討，從這些資訊知道各個演算法在何種特性資料庫有較好的表現；我們從獲得的資訊知道，以Apriori演算法為基礎之改良方法亦受到minimum support的影響，當minimum support小的時候，執行效率亦較差，反之，造成時間無法控制；其次FP-growth演算法對於緊密的資料庫(dense database)有相當好的壓縮效果，可大幅節省儲存交易記錄的空間需求，但對於稀疏的資料庫(sparse database)可能的增加大量儲存交易記錄的空間需求；此外，Non Apriori-like的方法不用產生候選項目集合，而且避免多次的高成本的資料庫掃瞄，節省了大量I/O的時間，因此整體的效率表現相當不錯。

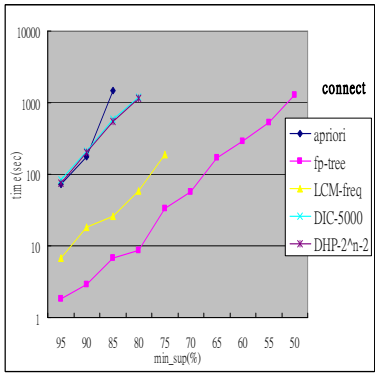
目前本研究只針對 Apriori 演算法、FP-growth 演算法、DIC 演算法、DHP 演算法與 LCM-freq 演算法進行探討，但是還有不少相關研究未進行分析，而且實驗資料庫目前為10 個，也有不足夠的地方；因此，我們希望持續增加探討分析有關探勘關聯演算法的特性；在實驗方面也希望能增加資料庫實驗，彙整相關因素，並且希望可以推導出一個模式，使能由資料庫大小、交易記錄的長短、資料分佈等各種相關資訊中得到一些規則與資訊，使其資訊充足以建立關聯法則知識庫，並提供給使用者運用。

6.致謝

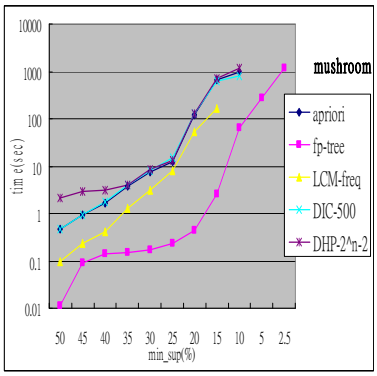
本論文承蒙九十五年度國科會研究計畫NSC95-2221-E-214-065-經費補助，特此致謝。



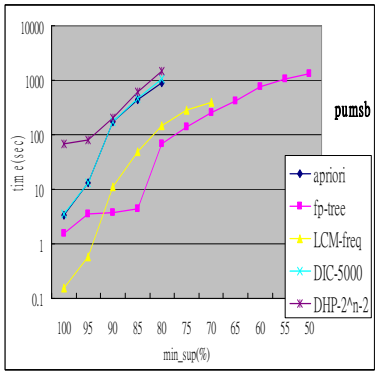
(1)chess



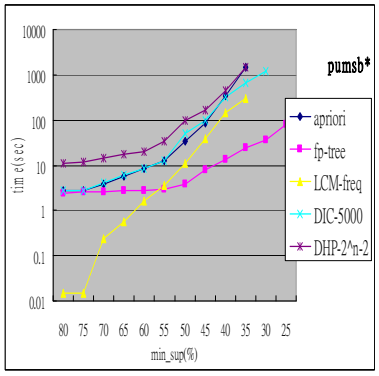
(2)connect



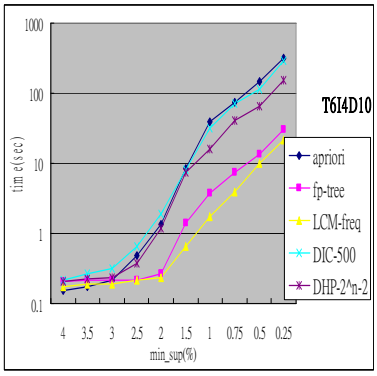
(3)mushroom



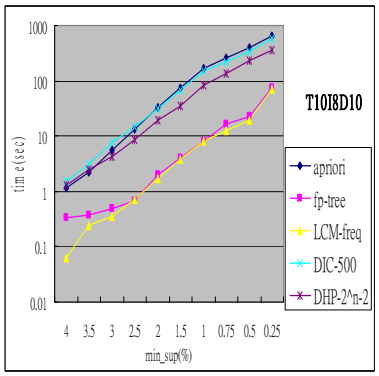
(4)pumsb



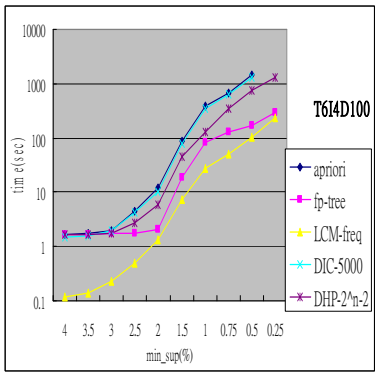
(5)pumsb*



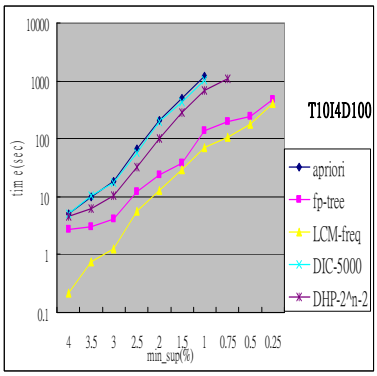
(6)T10I4D10



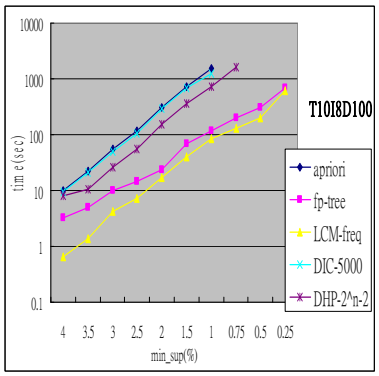
(7)T10I8D10



(8)T6I4D100



(9)T10I4D100



(10)T10I8D100

圖 12 演算法於各個資料庫之執行時間(續)

參考文獻

[1] Agrawal, R. and Srikant, R., "Fast Algorithm for Mining ssociation Rules," ***In Proceedings of the 20th International onference on Very Large Databases***, pp. 487-499, 1994.

[2] Brin, S., Motwani R. and Silverstein, C., "Beyond market baskets：Generalizing association rules to correlations," ***In Proceedings of ACM SIGMOD Conference on Management of Data***, pp. 265-276, 1997.

[3] Chen, M.S., Han J. and Yu, P.S., "Data Mining: An Overview from Database Perspective," ***IEEE Transactions on Knowledge and Data Engineering, Volume 8***, Number 6, pp. 866-883, 1996.

[4] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation, " ***Proceedings of the ACM SIGMOD International Conference on Management of Data, Dallas***, USA, pp. 241-250, 2000.

[5] J. S. Park, M. S. Chen, and P. S. Yu, "An effective hash based algorithm for mining association rules," ***Proceedings of the ACM SIGMOD International Conference on Management of Data***, San Jose, USA, pp. 175-186, 1995.

[6] M. J. Zaki and C.-J. Hsiao. "CHARM: An efficient algorithm for closed association rule mining." ***Technical Report 99-10, Computer Science Dept., Rensselaer Polytechnic Institute***, October 1999.

[7] M.J. Zaki and C.J. Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining," ***Proc. 2002 SIAM Int'l Conf. Data Mining (SDM '02)***, pp. 457-473, 2002.

[8] P. Palmernin, S. Orlando, R. Perego, "Statistical Properties of Transactional Databases" , ***In , Proc. of the ACM Symposium on Applied Computing*** , pp. 515-519,2004.

[9] S. Brin, R. Motwai, J.D. Ullman, and S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data," ***ACM SIGMOD Conference on Management of Data***, pp. 265-276, 1997.

[10] Takeaki Uno, Taisuya Asai, Yuzo Uchida, Hiroki Arimura"LCM：AN Efficient Algorithm for Enumerating Frequent Closed Item Sets", ***In Proc. IEEE ICDM99 Workshop*** *FIMI'03*,2003.

[11] IBM Almaden Research Center "Quest Synthetic Data Generation Code, "http://www.almaden.ibm.com/cs/quest/syndata.html."

[12] UC-Irvine Machine Learning Database Repository "http://mlearn.ics.uci.edu/databases/"

[13] 李金鳳，「資料探勘面面觀」，***資訊與教育雜誌***，台北(2001).

[14] Jiawei H. & Micheline K.(2000), Data Mining, Concepts and Techniques, Morgan KaufmannPublishers，曾龍 譯，資料探勘—概念與技術，***維科圖書***，台北縣中和市(2003)