國立台灣師範大學
資訊工程研究所碩士論文

指導教授：柯佳伶博士

以 FP-tree 結構為基礎之近似常見項目集探勘法

An Efficient Approach for Mining Approximate Frequent Itemsets based on FP-tree structure

研究生：涂益郎　撰

中華民國 九十八 年 六 月

國 立 臺 灣 師 範 大 學

博碩士論文授權書

本授權書所授權之論文為授權人在國立臺灣師範大學資訊工程研究所

九十八學年度第二學期取得碩士學位之論文。

論文題目：以 FP-tree 結構為基礎之近似常見項目集探勘法

指導教授：柯佳伶 博士

茲同意將授權人擁有著作權之上列論文全文（含摘要），授權本校圖書館及國家圖書館，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

□讀者基於非營利性質之線上檢索、閱覽或下載、列印上列論文，應依著作權法相關規定辦理。

指 導 教 授：柯佳伶 博士

授　權　人：涂益郎

研究生姓名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（請親筆正楷簽名）

中　華　民　國　九十八　年　七　月　十　日

摘要

以 FP-tree 結構為基礎之近似常見項目集探勘法

涂益郎

本論文針對交易資料庫運用 FP-tree 結構可壓縮儲存交易資料的特性,提出以 FP-tree 儲存結構為基礎之近似常見項目集探勘法，稱為 **FP-AFI** 演算法(**FP**-tree **A**pproximate **F**requent **I**temsets mining algorithm)。透過分析容錯包含項目集之交易資料集合間的遞迴關係，擴展 FP-tree 執行投影的方法，可分別找出包含某個項目與不包含某個項目的 conditional FP-tree。FP-AFI 演算法以深先搜尋的方式，從 Header Table 中取出符合核心樣式門檻值的項目產生候選項目集，系統化地建構出其對應的 conditional FP-tree，並透過 conditional FP-tree 根節點所記錄的計數值及 conditional FP-tree 的編碼資訊，快速獲得該項目集的容錯支持度及各項目支持度，以確認是否為一近似常見項目集。在探勘的過程中僅需掃瞄資料庫兩次，可省去大量讀取交易資料所需的時間。由實驗結果顯示，當最小支持度門檻值訂為較小或交易資料的筆數較多時，此方法較之前已提出的近似常見項目集探勘演算法 FT-Apriori 及 AFI 有顯著的執行效率增進。

Abstract

An Efficient Approach for Mining Approximate Frequent Itemsets
based on FP-tree structure

Yi-Lang Tu

In this thesis, an algorithm called **FP-AFI** (**FP**-tree **A**pproximate **F**requent **I**temsets mining) is proposed for mining approximate frequent itemsets. The FP-AFI applies the characteristics of FP-tree structure to compress transaction data. Through analyzing the recursive relationship of the sets of transactions which fault tolerant contain an itemset, the projection operation on FP-tree is extended to obtain the conditional FP-tree which contains a specific item or not. The FP-AFI algorithm applies a depth-first search strategy to generate candidate itemsets by checking the threshold value of a core pattern from the item supports counted in the Header Table. For each candidate itemset, its corresponding fault-tolerant conditional FP-trees are constructed systematically. Accordingly, the approximate support of the candidate and the item supports of each item in the candidate are obtained easily from the counter stored in the root node and the coding vectors of the fault-tolerant conditional FP-trees. Such that, a candidate itemset is confirmed to be an approximate frequent itemset or not efficiently. The experimental results show that, when there are many transactions or the support is small, the performance efficiency of FP-AFI algorithm is better than the FT-Apriori and AFI algorithms proposed previously.

誌謝

論文終於順利完成，最要感謝的是我的指導教授－柯佳伶老師。老師總是在我毫無頭緒的時候，引導我正確的方向，很多想法在跟老師討論後變得豁然開朗；也因為老師耐心的督促與鼓勵，讓我的論文能夠按照既定的進度完成。撰寫論文期間，老師細心的一字一句仔細批改，斟酌調整字句的順序，使論文內容能更加通順。還要感謝陳良弼教授與吳宜鴻教授在百忙之中抽空為我的論文提出建言與指導；也要感謝在這段期間給予我協助的學長們，以及忙著張羅口試大小事項的毓雯與衣菱學姐，還有其他實驗室的學弟們，因為你們的幫助讓我能毫無後顧之憂的專注在論文與口試的準備。

兩年來的研究生活，感謝我的同學蕙君、佩君及哲瑋，陪伴著我相互扶持與成長。在這既緊湊又充實的兩年中，無論在課業或是研究上都受到你們很大的幫助，感謝你們總是在我有困惑的時候，耐心的聽我的問題並為我解惑。

最後要感謝我最摯愛的家人與朋友，感謝你們總在我最需要的時候給予我熱切的鼓勵與支持，因為你們的激勵讓我有無比的信心繼續向前邁進，順利完成研究所的學業。

涂益郎　謹識

於國立台灣師範大學資訊工程研究所

二００九年七月

目錄

附表目錄

附圖目錄

第一章 緒論

1-1 研究背景與研究動機

資料探勘(data mining)[13]的研究主要在探討如何針對大量資料進行分析、歸納，以發掘出隱藏在資料中的資訊。這些資訊可提供決策者在執行上的策略參考方針：例如網站伺服器中記錄的使用者瀏覽資料，可透過資料探勘技術找出使用者瀏覽網站中不同網頁的關聯性，藉此設計網站中網頁間的連結及架構；或是由賣場客戶的交易資料中，找出消費者購買商品的關聯性，以安排商品擺售的動線等。

探勘常見項目集(Frequent Itemset，簡稱 FI)是資料探勘研究領域中經常被探討的主題之一，在眾多提出的方法中，有一類是透過列舉項目集的方式找出常見項目集，如早期的代表方法是論文[1]所提出的 Apriori 演算法，該方法於探勘的過程中必須在每一回合列舉出大量的候選項目集，且在每一回合計數各候選項目集的支持度時都需要重複掃瞄整個資料庫內容一次，將消耗大量資料讀取時間。因此後來又發展出另一類型的探勘方法，將交易資料轉換為樹狀組織表示，並於儲存後進行探勘，如論文[2]所提出的 FP-growth 演算法為主要代表，其利用 FP-tree 結構儲存資料庫中交易記錄所包含的項目集，在探勘常見項目集過程不需列舉大量候選項目集，且最多掃瞄整個資料庫兩次。

根據原始常見項目集的定義，在計數一個項目集的出現次數時，該項目集中的每一個項目都必須被包含在一筆交易資料中才能計數出現一次。但在實際的資

料庫中，由於資料項目的種類繁多，或是因為資料中含有雜訊資料(noise)，探勘出來的常見項目集往往會變得破碎、零散，形成僅由少數項目所構成的常見項目集。例如表 1.1 所示之商店交易資料，若將最小支持度計數門檻值訂為 2，因為{麵包,牛奶}與{牛奶,蘋果}出現在 3 筆交易中的其中 2 筆，因此皆為常見項目集。{蘋果,牛奶,麵包}的支持度只有1次，所以不滿足常見項目集的要求。但實際上{麵包}及{蘋果}都有伴隨{牛奶}出現，又曾經共同出現一次，若放寬交易資料包含項目集的比對方式，有限度的容忍部分項目不被包含在交易資料中，則可算出{蘋果,牛奶,麵包}的容錯支持度為3 ，即滿足最小支持度計數門檻值，因而找出更多項目間的關聯現象。

表 1.1 範例交易資料庫

客戶編號	購買商品
A	麵包,牛奶
B	牛奶,蘋果
C	蘋果,牛奶,麵包

論文[6]中首先針對上述問題，在比對項目集時考慮容錯誤差，將容錯項目個數訂為固定值，找出資料庫中的容錯常見項目集。論文[4]改進容錯項目的個數為一固定比例值ε_r，因此容錯項目個數會隨項目集長度等比例值調整，以反應較長項目集可容忍錯誤的項目應隨之增加的想法。此外，項目集中各項目出現於容錯包含該項目集的交易集合亦必須至少達到比例值$(1-\varepsilon_c)$。其所找出的常見項目集

稱為近似常見項目集(Approximate Frequent Itemset，簡稱 AFI)。

表 1.2 範例交易資料庫

交易編號	Burger	Coke	Diet Coke
T0	1	1	0
T1	1	1	0
T2	1	0	1
T3	1	0	1

論文[3,5]則在探勘近似常見項目集時多加上核心樣式(Core Patterns)的概念。表1.2顯示論文[3]中所列出的一個交易資料庫範例，按照論文[4]所定義的近似常見項目集，若將最小支持度計數門檻值訂為3，兩個誤差容忍參數ε_r與ε_c分別訂為 0.33 及 0.5，則探勘出來的近似常見項目集為{ Burger, Coke, Diet Coke }。但作者認為 Coke 與 Diet Coke 在資料庫中完全沒有被同時購買，因此產生出{ Burger, Coke, Diet Coke }為近似常見項目集是不合理的。於是該論文中作者定義出核心樣式，要求一個近似常見項目集除了其容錯包含的支持度必須大於或等於最小支持度計數門檻值外，該項目集在交易資料集中出現的支持度值必須達到最小支持度計數門檻值*α(其中$0 \le \alpha \le 1$，為使用者所指定)，代表此資料項目集在真實的交易資料集中曾出現且佔有一定的比例，對使用者而言才是一個有意義的近似常見項目集。論文[7]中對各種近似常見項目集方法進行分析比較時，認為使

用核心樣式的概念可以較有彈性地調整探勘近似常見項目集的容錯程度。當資料庫中存在少數雜訊資料或項目種類的數量較少時，可以調高核心樣式參數α的值，使得雜訊資料或支持度計數值較低的項目被過濾掉。當資料庫中雜訊資料或項目種類的數量較多時，會使資料庫中資料的分布較稀疏，透過降低核心樣式參數α的值，仍可過濾掉雜訊資料或支持度計數值較低的項目，在產生候選項目集時不會保留過多的項目，避免耗費大量的時間計數候選項目集的容錯支持度。綜合上述優點，本論文在考慮近似常見項目集探勘時亦將結合核心樣式的過濾條件。

此外，以往所提出探勘近似常見項目集的方法，多數在探勘過程中必須透過列舉的方式組合出候選項目集，可能列舉出一些在資料庫中未出現的項目集，因而增加計數項目集支持度過程的計算成本。因此本論文將探討如何運用論文[2]中提出之 FP-tree 結構壓縮儲存交易資料集，並利用由 FP-tree 可取得與特定項目集一起出現的其他項目及出現次數等資訊，避免產生不必要的候選項目集，發展出以 FP-tree 結構為儲存基礎的近似常見項目集探勘方法。

1-2 相關研究

1-2.1 常見項目集探勘

早期從交易資料庫中探勘常見項目集的代表方法為Apriori演算法[1]，該方法在第一回合先找出常見項目，在接下來的每一回合透過組合的方式由長度k－1的常見項目集組合成長度為k的候選項目集，並掃瞄交易資料庫計數出候選長度為k的項目集之支持度，由支持度是否大於最小支持度計數門檻值來判斷是否為常見項目集。上述步驟會反覆執行，直到無法組合出更長的候選項目集，或已經找不出常見項目集為止。此方法有兩個主要的缺點：（一）每一回合都需組合出新的候選項目集，若項目的種類甚多，則組合出的候選項目集數量亦跟著大量增加，但可能組合出來的項目集根本不存在於資料庫中，卻仍要計數其支持度值，將消耗大量計算時間；（二）每一回合都需要重複掃瞄整個交易資料庫的內容，倘若資料數量龐大，則需耗費相當多的時間在掃瞄及讀取資料。

為提供更有效率的常見項目集探勘方法，近年來陸續有許多論文探討相關研究。論文[2]發表的FP-growth演算法提出利用FP-tree結構，以相同字首共享樹中同一路徑節點的方式，對於資料庫中的交易資料，記錄每筆交易中包含的項目集及其出現在資料庫中的累積次數，壓縮儲存在FP-tree結構中。在其探勘過程會先掃瞄資料庫一次，計數各項目的支持度計數值，保留支持度計數值大於或等於最小支持度計數門檻值的項目，形成長度為1的常見項目集。接著將各交易資料中所包含的項目依其支持度計數值由大至小排序，之後再掃瞄一次資料庫將每筆交

易資料加入 FP-tree 結構中。FP-growth 演算法採取 Divide-and-Conquer 的策略，分別對各單一項目建立 conditional FP-tree，並根據相對應產生的 Header Table 中所統計出的項目出現次數找出常見項目集。繼續再以新產生的常見項目集為基礎，透過遞迴的方式反覆執行，探勘出更長的常見項目集。根據探勘過程可觀察出 FP-growth 演算法僅需掃瞄資料庫兩次，不需以列舉組合的方式產生候選項目集，在資料項目集數量固定，但交易資料筆數龐大的情況下，特別能顯出其所增進的探勘效率。

論文[9]所提出的 TD-FP-Growth 演算法亦採用 FP-tree 結構進行常見項目集探勘，與 FP-growth 演算法不同處在於其利用 FP-tree 結構以遞迴的方式探勘常見項目集時，不需另外建立 Conditional FP-tree ，僅需建立 Conditional Header Table 指向原 FP-tree 中的對應節點，因此與論文[2]提出的 FP-growth 演算法比較起來，TD-FP-Growth 能提供更好的執行效率。

1-2.2 近似常見項目集探勘

如同前述，對於探勘常見項目集的問題，在比對交易資料是否包含一個項目集的過程中考慮容錯誤差以容錯支持度值來判定是否滿足最小支持度門檻的要求，所找出的項目集稱為近似常見項目集或是容錯常見項目集[3,4,6,10,11,12]，其可提供更一般化的探勘結果。

論文[6]最早定義出近似常見項目集的概念，當比對資料庫的交易資料是否包含某項目集時，允許該項目集中固定個數的項目不被包含在一筆交易資料內，以

此容錯包含的方式計算出該項目集在資料庫中的容錯支持度。其中所提出的FT-Apriori 演算法是採用如同 Apriori 演算法產生候選項目集的方式，與 Apriori 演算法不同之處唯有在計數支持度時修改成以檢查容錯包含的方式計數。由於FT-Apriori 演算法在執行上的缺點與 Apriori 相同，若交易資料庫中項目種類眾多，則需要組合出大量的候選項目集，且每個候選項目集都要透過掃瞄整個交易資料庫來計數出其容錯支持度，在執行上將耗費大量的時間。

論文[10]提出 FFT-Mine 演算法，在比對時亦採用固定容錯項目個數的限制條件。其利用出現位元序列表示法來儲存交易資料，並將此表示法擴展成容錯出現位元序列來表示一個容錯項目集在資料庫中出現的情形。FFT-Mine 演算法透過深度優先的搜尋方式產生候選項目集，以遞迴方式定義算出候選項目集容錯出現位元序列之計算函式，由該序列運算結果即可快速判別一個候選項目集是否為一個容錯常見項目集。

論文[12]中則將近似常見項目集稱為 ETIs(Error Torrent Itemsets)，運用一個介於 0 到 1 之間的誤差容忍參數值 ε，允許容錯項目的個數為項目集的長度乘上 ε 值，因此較原本固定容錯項目的做法更有彈性。但僅運用 ε 這個誤差容忍參數，檢查資料庫中容錯包含此項目集的交易資料時，即使一個項目集中的某些項目很少或完全沒有和其他項目一起出現在交易資料內，亦可能因其他項目一起出現的次數夠多而判斷整體為一個容錯常見項目集，但其項目間出現的關聯性不夠強。

為解決上述問題，論文[4]定義出兩個介於0到1的誤差容忍參數ε_r與ε_c，其中ε_r相當於論文[12]中所定義的ε參數，而另一個誤差參數ε_c則用來限定容錯包含一個項目集的那些交易資料中，該項目集中的每個項目曾出現的比例值皆必須大於等於$(1-\varepsilon_c)$值。該論文所提出的AFI演算法，在產生候選項目集的過程中，先掃瞄一次資料庫對每個項目記錄包含該項目的交易之編號所成的集合。後續組成更長候選項目集的做法仍與 Apriori 演算法類似，只是在計數其支持度時可根據所記錄之各項目出現交易編號資訊，因此不需重新掃瞄資料庫。

論文[3]認為僅運用ε_r與ε_c這兩個誤差參數，仍有可能所找出來的近似常見項目集根本從未出現於交易資料中，產生不合理的探勘結果，因此在論文[3]中另提出核心樣式(Core Patterns)的條件，限定所找出近似常見項目集在交易資料的出現比例仍要達到一個最低的比例值α。該論文中提出一個稱為"AC-Close"的演算法，其根據核心樣式的門檻值限制條件，採用由上至下(Top-down)深度優先搜尋，在過程中運用一些可刪減候選項目集的策略縮減搜尋空間，探勘出近似常見封閉項目集(Approximate Closed Itemsets)。

1-3 論文方法

為避免在探勘過程列舉不必要產生的候選項目集，本論文運用 FP-tree 結構可壓縮儲存交易資料的特性，提出以 FP-tree 儲存結構為基礎之近似常見項目集探勘法，稱為 FP-AFI 演算法(FP-tree Approximate Frequent Itemsets mining algorithm)。綜合上述討論，本論文針對近似常見項目集探勘的問題，將採用論文[3]中結合誤差容忍參數 ε_r、ε_c 以及核心樣式門檻值的限制條件。本論文擴展 TD-FP-growth 演算法，除了以特定項目對 FP-tree 做投影(projection)得到包含該項目的 conditional FP-tree，亦可對不包含某項目的項目集建立 conditional FP-tree，以系統化的方式從 FP-tree 所儲存資訊計數出一個候選項目集在資料庫中的容錯支持度，因而避免重複掃瞄資料庫所耗費的時間。

在實驗中我們比較本論文提出的 FP-AFI 演算法與其他同樣探勘近似常見項目集的 FT-Apriori 與 AFI 演算法之執行效率，實驗結果顯示當模擬資料集中包含的交易資料筆數眾多或執行時所給定的最小支持度門檻值較小時，FP-AFI 演算法特別能發揮顯著增進的執行效能。將比對項目集容忍誤差比例的誤差參數 ε_r 訂得越大時，項目集內容錯項目的個數隨之變多，而 FP-AFI 演算法的執行時間仍然低於另外兩種方法。另外我們實驗以實際的動物特徵資料進行近似常見項目集探勘，以找出對應到每個類別動物的共同特徵。其結果顯示 FP-AFI 演算法針對動物特徵資料所探勘得到的近似常見項目集，可有效挑選出該類別具有代表性的特徵。

1-4 論文架構

本論文以下章節內容簡介如下：第二章將詳細說明本論文所研究的問題，並對論文中提及的相關名詞加以定義。在第三章中介紹本論文所提出之FP-AFI演算法。第四章將顯示本論文提出的方法與其他相關研究之間的執行效率比較，並驗證本方法對實際資料集分析探勘的效果。最後在第五章做出總結，並探討未來研究方向。

第二章 相關名詞定義及問題定義

2-1 名詞定義

令集合 $I = \{ i_1 , i_2 , i_3 , ... , i_m \}$ 為交易資料庫中可能出現的所有項目(item)所成的集合，一個**項目集**(itemset) P 是由 I 中一個或一個以上的項目所成的集合。一個項目集 P 的**長度**是指該項目集中包含的項目個數，以 $|P|$ 表示。以集合 $P = \{ i_1 , i_3 , i_5 \}$ 為例，其項目集長度為 3。一個長度為 k 的項目集，稱為 **k-項目集**(k-itemset)。

令 TDB 表示一個交易資料庫，其中的每一筆交易資料(transaction) T = (tid , S)，由一個數字編號 tid 及一組 I 的非空子集合 S 所構成。給定一個項目集 P ，則在 TDB 中計數有多少筆交易資料包含 P 所得到的計數值，稱為項目集 P 在 TDB 中的**絕對支持度**，以 sup(P) 表示；而項目集 P 在 TDB 中的**相對支持度**則由 P 的絕對支持度除以 TDB 中的總交易筆數得到。在以下論文的描述中，所稱之支持度即表示絕對支持度。以 $|TDB|$ 表示 TDB 中的交易筆數，當給定系統一個介於 0 到 1 之間的**最小支持度門檻值**(minimal support threshold ，以 min_sup 表示)，則由 $\lceil |TDB| * min\_sup \rceil$ 得到的值稱為**最小支持度計數門檻值**(minimal support count threshold)，以 min_sup_c 表示。若項目集 P 的支持度大於或等於 min_sup_c，則稱 P 為 TDB 中一個**常見項目集**(frequent itemset)，否則稱 P 為 TDB 中一個**非常見項目集**(infrequent itemset)。

2-2 問題定義

在本問題定義中所使用的誤差容忍參數ε_r與ε_c皆參照論文[4]中所定義，而使用到的核心樣式參數α之定義則是參考論文[3]所述。

【定義 2_1】

給一個項目集 P，於交易資料庫中，若交易資料$T_i=(i,S_i)$，其中S_i存在一個子集合S'，$S'\subseteq P$，且S_i不存在另一子集合S''使得$S'\subset S''\subseteq P$，則稱T_i以**比對錯誤值**($|P|-|S'|$)包含 P。給定誤差容忍參數ε_r，以μ_P表示$\lfloor |P|*\varepsilon_r \rfloor$，稱為 P 的**最大比對錯誤值**，若$T_i$包含 P 的比對錯誤值($|P|-|S'|$)$\le \mu_P$，則稱交易資料$T_i$**容錯包含** P。

以上定義表示交易資料T_i中最多可容忍μ_P個項目不包含在 P 中，隨$|P|$的改變μ_P會隨之動態(dynamic)改變。而 TDB 內所有容錯包含 P 的交易資料所形成之交易集合以$T_a^P(\mu_P)$表示，$|T_a^P(\mu_P)|$為 P 在 TDB 中的**容錯支持度**，亦表示為$\sup_a(P)$(approximate support)。而對 P 中每一個項目 x，$T_a^P(\mu_P)$包含 x 的交易筆數稱為 x 在$T_a^P(\mu_P)$中的**項目支持度**，以$\sup_{item}^P(x)$表示。

【範例 2_1】

以表 2.1 所示之範例資料庫 TDB 為例，其中包含 8 筆交易資料。令誤差容忍參數ε_r為 0.2，項目集 P 為{ A, B, C, D, E }，則 TDB 中每一筆交易在比對包含項目集 P 時，可容忍 1 個項目($\lfloor |P|*\varepsilon_r \rfloor=\lfloor 5*0.2 \rfloor=1$)不包含在 P 中，只要在每筆交易資料中比對到 P 中的任 4 個項目($\lceil |P|*(1-\varepsilon_r) \rceil=\lceil 5*(1-0.2) \rceil=4$)即可視為容錯

包含P。因此T3、T4、T7及T8為容錯包含P的交易資料，$T_a^P(1)=\{T3,T4,T7,T8\}$，$sup_a(P)=4$。項目A出現在$T_a^P(1)$中的T3、T4及T8共三筆交易資料，因此統計得到$sup_{item}^P(A)=3$。P中其餘項目於$T_a^P(1)$內獲得的支持度分別為$sup_{item}^P(B)=4$、$sup_{item}^P(C)=4$、$sup_{item}^P(D)=3$及$sup_{item}^P(E)=4$。而項目集P被完全包含在交易資料T4與T8中，因此$sup(P)=2$。

表 2.1 範例資料庫 TDB

交易編號	交易資料項目集
T1	AF
T2	AB
T3	ABCE
T4	ABCDE
T5	ACD
T6	CDE
T7	BCDE
T8	ABCDEG

【定義 2_2】近似常見項目集

給定一個交易資料集TDB，兩個誤差容忍參數ε_r及ε_c、核心樣式參數α及最小支持度門檻值min_sup，其中ε_r、ε_c、α及min_sup皆為介於0到1之間的實數。若一個項目集P在資料庫TDB中符合：

$sup_a(P) \geq min\_sup_c$，(如2-1節所定義$min_sup_c = \lceil |TDB| * min_sup \rceil$)，(條件一)

對於P中每一個項目x，$sup_{item}^P(x) \geq \lceil sup_a(P) * (1-\varepsilon_c) \rceil$， (條件二)

且$sup(P) \geq min_sup_c * \alpha$。 (條件三)

則稱P為TDB中一個**近似常見項目集**。

【範例 2_2】

同樣以表 2.1 所示之範例資料庫 TDB 為例，若給定誤差容忍參數 ε_r 為 0.2，ε_c 為 0.5，min_sup 訂為 0.3、α 訂為 0.5，則 $\min\_sup_c$ 為 $\lceil 8*0.3 \rceil = 3$。如範例 2_1 所示，項目集 P = { A, B, C, D, E } 於 TDB 中之容錯支持度 $sup_a(P) = 4 \geq \min\_sup_c$，滿足條件一。而 P 中每一個項目在 $T_a^P(1)$ = { T3, T4, T7, T8 } 中的項目支持度分別為 $sup_{item}^P(A) = 3$、$sup_{item}^P(B) = 4$、$sup_{item}^P(C) = 4$、$sup_{item}^P(D) = 3$ 及 $sup_{item}^P(E) = 4$，符合 $\lceil sup_a(P) * (1 - \varepsilon_c) \rceil$ (也就是 $\lceil 3*(1-0.5) \rceil = 2$) 的門檻值要求，滿足條件二。且 P 被包含在 T4 與 T8 兩筆交易資料中，$sup(P) = 2 \geq \min\_sup_c * \alpha$ (即 $3*0.5 = 1.5$)，滿足條件三。因此項目集 P = { A, B, C, D, E } 為 TDB 中一個近似常見項目集。

若在相同的參數設定下，項目集 Q = { B, C, D, E, G }，其最大比對錯誤值亦為 1。因此 Q 被 TDB 中的 T4、T7 及 T8 容錯包含，$sup_a(Q) = 3 \geq \min\_sup_c$ 滿足條件一。但分析 Q 中每一個項目出現在 $T_a^Q(1)$ 的項目支持度時，雖然 $sup_{item}^Q(B) = 3$、$sup_{item}^Q(C) = 3$、$sup_{item}^Q(D) = 3$、及 $sup_{item}^Q(E) = 3$ 都符合 $\lceil sup_a(Q) * (1 - \varepsilon_c) \rceil$ (也就是 $\lceil 3*(1-0.5) \rceil = 2$) 的門檻要求，但 $sup_{item}^Q(G) = 1$ 卻無法滿足條件二。此外 $sup(Q) = 1 \leq \min\_sup_c * \alpha$ 亦不滿足條件三。由於無法同時滿足定義 2_2 中三個構成近似常見項目集的條件，因此 Q 不為 TDB 中一個近似常見項目集。

第三章 以 FP-tree 為基礎之近似常見項目集探勘

本章節將詳細說明如何以 FP-tree 為儲存結構探勘近似常見項目集，其中分別介紹 FP-AFI 演算法(FP-tree Approximate Frequent Itemsets mining algorithm)之主要構想及執行方法細節。

3-1 交易資料集儲存結構

本論文提出之 FP-AFI 演算法修改論文[2]提出的 FP-tree(Frequent-Pattern tree)結構，將交易資料集中出現的項目集緊密的壓縮儲存，並累計項目集的支持度。儲存結構建立的過程主要分成三個部分的處理：<1>以核心樣式限制門檻值過濾資料項目，<2>建立 FP-tree 中的樹狀結構，<3>建立 FP-tree 中的 Header Table，我們將在以下三個小節中各別說明。

3-1.1 以核心樣式限制門檻值過濾資料項目

FP-growth 及 TD-FP-growth 演算法在建立 FP-tree 結構前會掃瞄一次交易資料集，計數各項目出現的支持度，以給定之最小支持度門檻值找出資料集中的常見項目。而再一次掃瞄交易資料集時會忽略交易中所包含的非常見項目，使得 FP-tree 上儲存的資料項集都是由常見項目所形成的。這是根據從 Apriori 演算法中所提出之 Apriori property：若有一個長度為$(k-1)$的項目集 P 為非常見項目集，則包含 P 之長度為 k 的項目集也必定為非常見。因此可得知若一個項目 x 為非常見，則所有包含 x 之項目集必為非常見，所以可將 x 從交易資料中忽略不管。近似常

見項目集之間雖然不會滿足Apriori property，但只針對滿足核心樣式限制門檻值條件的項目集間則仍會滿足此特性。

【定理 3_1】

給定項目 x 及項目集 P ，且 x 為 P 中一個項目。若 $sup(x) \le min\_sup_c * \alpha$ ，則 $sup(P) \le min\_sup_c * \alpha$ 。

【證明】

一筆交易資料若包含項目集 P ，則必包含項目 x ，由此可知 $sup(P) \le sup(x)$ ，因此若 $sup(x) \le min\_sup_c * \alpha$ ，sup(P)必定小於或等於 $min\_sup_c * \alpha$ ，由此可得證。

由上述定理，當一個項目 x 之支持度小於核心樣式門檻值 $min\_sup_c * \alpha$ 時，則任何包含項目 x 的項目集 P 必定因不滿足核心樣式的限制條件而無法成為近似常見項目集。因此本論文方法在建立FP-tree時，會先掃瞄一次資料集，計數資料集中各項目的支持度，標記出支持度小於核心樣式門檻值的項目。在建立FP-tree樹狀結構時，忽略這些不符合核心樣式要求的項目。

【範例 3_1】

以表3.1所示的交易資料庫為例，若 min_sup_c 值為 4 ，核心樣式門檻值 α 訂為 0.5 ，第一次掃瞄資料庫後計數出各項目的支持度為 A:6, B:5, C:6, D:5, E:5, F:1, G:1。在 TDB 中支持度大於或等於核心樣式門檻值(即 $4 * 0.5 = 2$)的項目有 A:6, B:5,

C:6, D:5 及 E:5，項目 F 與 G 因支持度小於 2，在下一步驟讀取 TDB 資料集建立 FP-tree 樹狀結構的過程中，會被忽略而不建立對應節點。

表 3.1 範例資料庫 TDB

交易編號	交易資料項目集
T1	AF
T2	AB
T3	ABCE
T4	ABCDE
T5	ACD
T6	CDE
T7	BCDE
T8	ABCDEG

3-1.2 建立 FP-tree 樹狀結構

FP-tree 樹狀結構為一個 prefix tree，也就是交易中包含的項目集若有相同的 prefix 則會共用樹中節點路徑。FP-tree 的根節點不代表資料集中任何項目，而其他節點中皆儲存一個項目資訊及計數值，該節點表示從根節點到該節點所經過路徑上的項目所形成的項目集。原先提出的 FP-tree 結構在建立時，根節點不會記錄計數資訊；但在本論文中，為後續探勘近似常見項目集之用途，在根節點中亦記錄一個計數值，其用途將在以下章節中說明。此外，我們在探勘近似常見項目集的過程是依項目字元順序遞迴加入產生項目集，因此在建立 FP-tree 時交易資料中各項目直接以字元順序排列。

對於交易資料庫中每一筆交易資料 T_i，其加入 FP-tree 的方式如下：<1>去除不符合核心樣式要求的項目。<2>由根節點開始，對根節點計數值加 1，並搜尋第

一層子節點中是否有與交易資料內第一個項目相同的項目。若已存在相同的項目，只需將對應節點的計數值加1；若尚未存在則在根節點下建立一個對應此項目的新子節點，並將節點中的計數值設為1。接著取下一個項目，並由目前所在節點往下一層子節點搜尋，並重複執行步驟<2>。當交易資料中所有項目都處理過即完成該筆交易資料的儲存處理。

【範例 3_2】

延續範例 3_1 的處理，建構完成的 FP-tree 樹狀結構如圖 3.1 所示。

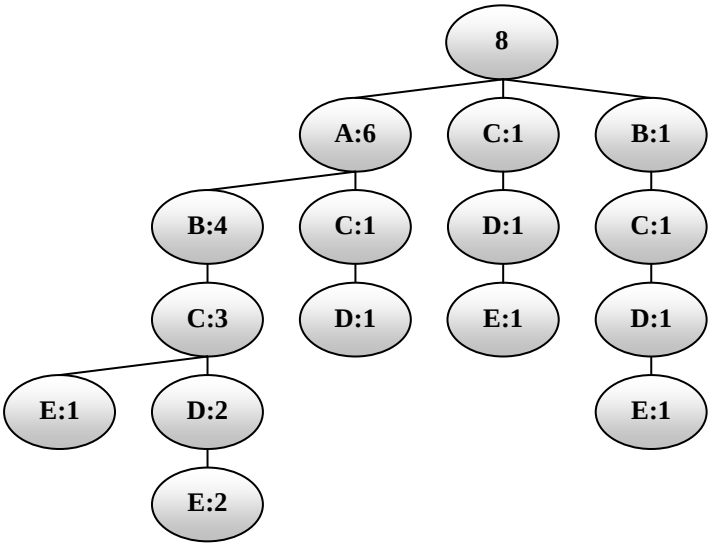


圖 3.1 範例資料庫 TDB 之 FP-tree 樹狀結構

3-1.3 建立 FP-tree 的 Header Table

建構FP-tree時還需建立一個資料項目表稱為 Header Table，將樹狀結構上具相同項目的節點以橫向連結的方式串聯起來，可助於探勘走訪FP-tree時較有效率。Header Table 有以下三個欄位：第一個欄位是項目欄，存放資料庫中的項目；

第二個欄位為計數欄，用來記錄項目欄中所記錄的項目在該FP-tree上出現次數的計數值；第三個欄位為節點連結欄，用來存放一個節點指標(pointer)，指向FP-tree中所有儲存該項目的節點所形成的串列。在本論文以下範例圖中，Header Table的欄位標示簡寫如下，I(item)：項目欄；C(count)：計數欄；L(link)：連結欄。

【範例 3_3】

延續範例 3_2，以表3.1的範例資料庫為例，Header Table 會隨著FP-tree的建立而相應產生，其結果如圖3.2所示，其中虛線表示具相同項目節點間的橫向連結。

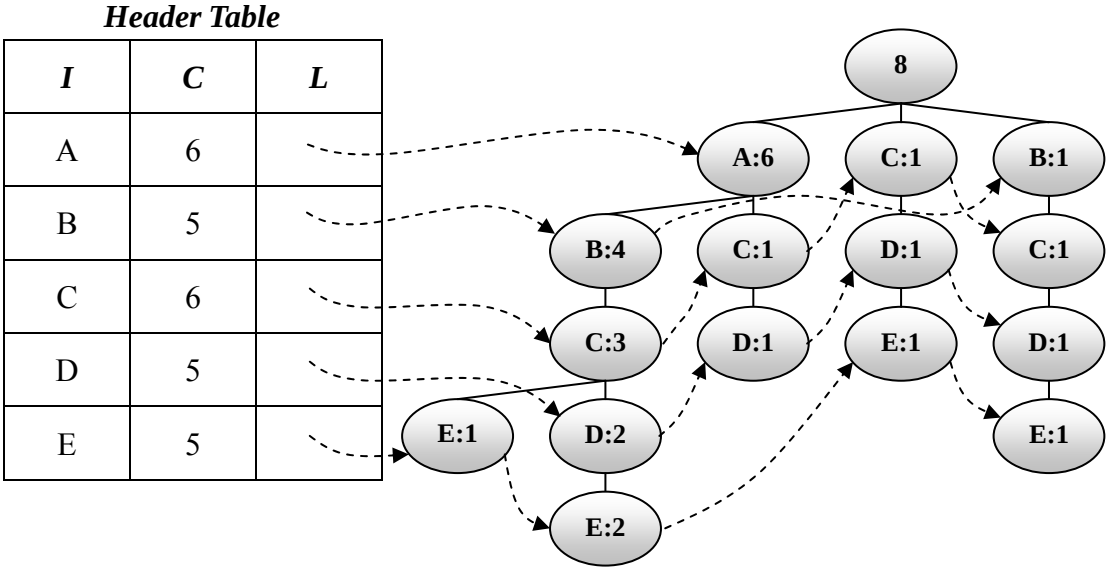


圖 3.2 範例資料庫 TDB 之 FP-tree 結構與 Header Table

3-2 以 FP-tree 結構計算容錯支持度

3-2.1 項目集容錯支持度計算方式

令一項目集 $P=\{p_1,p_2,\ldots,p_n\}$ 由 n 個項目組成，令 P' 表示項目集 P 中前 $n-1$ 個項目所形成的項目集。如前述定義，給定誤差容忍參數 ε_r，$\mu_P=\lfloor |P|*\varepsilon_r \rfloor$ 稱為 P 的最大比對錯誤值，表示在比對 P 時最多可容錯的項目個數。因此比對交易資料是否容錯包含項目集 P 時，其可容錯項目個數會隨 ε_r 值的設定及 P 的長度而決定。

我們以 $T_e^P(i)$ 表示資料庫 TDB 中所有以比對錯誤值 i 包含 P 的交易資料所形成的集合。對 P 而言，資料庫中不同比對錯誤值 i 得到的 $T_e^P(i)$ 彼此間不會有交集 (overlap)。若得知容錯包含 P' 的交易資料集合 $T_e^{P'}(i)$ $(0\le i\le \mu_P)$，可以計算出 P 在資料庫中的容錯支持度 $sup_a(P)$，如以下函數所示：

【計算項目集容錯支持度之函數定義】

若 $\mu_P=0$，$sup_a(P)=\left|T_e^P(0)\right|=sup(P)$ (等式 1)

若 $\mu_P=|P|$，$sup_a(P)=|TDB|$ (等式 2)

否則，$sup_a(P)=\left|T_a^P(\mu_P)\right|=\left|\bigcup_{i=0}^{\mu_P}T_e^P(i)\right|=\sum_{i=0}^{\mu_P}\left|T_e^P(i)\right|$ (等式 3)

其中 $T_e^P(i)$ 的遞迴定義如下：

$$T_e^P(i)=(T_e^{P'}(i)\cap T_e^{\{p_n\}}(0))\cup(T_e^{P'}(i-1)\cap T_e^{\{p_n\}}(1)) \quad \text{(等式 4)}$$

初始值定義如下：

若 $i=0$，$T_e^P(0)=T_e^{P'}(0)\cap T_e^{\{p_n\}}(0)$ (等式 5)

若 $|P|=1$ 且 $i=1$ ， $T_e^P(1)=TDB-T_e^P(0)$ (等式 6)

若 $|P|<i$ ， $T_e^P(i)=\phi$ (等式 7)

圖 3.3 中列舉當項目集 P 長度為 1 到 4，而 $sup_a(P)=\left|T_a^P(\mu_P)\right|$，套用上述函數，列出如何由 $T_e^{P'}(i)$ $(i\le\mu_P)$組合出 $T_a^P(\mu_P)$ ，並從中求得 $sup_a(P)$ 。

項目集 P	ε_r 改變區間	μ_P	容錯支持度
$\{p_1\}$	$0\le\varepsilon_r<1$	0	$sup_a(p_1)=\left\|T_e^{p_1}(0)\right\|$
	$\varepsilon_r=1$	1	$sup_a(p_1)=\left\|TDB\right\|$
$\{p_1,p_2\}$	$0\le\varepsilon_r<\frac{1}{2}$	0	$sup_a(p_1p_2)=\left\|T_e^{p_1p_2}(0)\right\|=\left\|T_e^{p_1}(0)\cap T_e^{p_2}(0)\right\|$
	$\frac{1}{2}\le\varepsilon_r<1$	1	$sup_a(p_1p_2)=\left\|T_a^{p_1p_2}(1)\right\|$ $=\left\|\bigcup_{i=0}^{1}T_e^{p_1p_2}(i)\right\|=\sum_{i=0}^{1}\left\|T_e^{p_1p_2}(i)\right\|=\left\|T_e^{p_1p_2}(0)\right\|+\left\|T_e^{p_1p_2}(1)\right\|$ $=\left\|T_e^{p_1}(0)\cap T_e^{p_2}(0)\right\|+\left\|T_e^{p_1}(1)\cap T_e^{p_2}(0)\right\|$ $+\left\|T_e^{p_1}(0)\cap T_e^{p_2}(1)\right\|$
	$\varepsilon_r=1$	2	$sup_a(p_1p_2)=\left\|TDB\right\|$

圖 3.3 項目集長度為 1 到 4 之容錯支持度運算範例

$\{p_1, p_2, p_3\}$	$0 \le \varepsilon_r < \frac{1}{3}$	0	$\sup_a(p_1p_2p_3) = \left\|T_e^{p_1p_2p_3}(0)\right\|$ $= \left\|T_e^{p_1}(0) \cap T_e^{p_2}(0) \cap T_e^{p_3}(0)\right\|$
	$\frac{1}{3} \le \varepsilon_r < \frac{2}{3}$	1	$\sup_a(p_1p_2p_3) = \left\|T_a^{p_1p_2p_3}(1)\right\|$ $= \left\|\bigcup_{i=0}^{1} T_e^{p_1p_2p_3}(1)\right\| = \sum_{i=0}^{1}\left\|T_e^{p_1p_2p_3}(i)\right\|$ $= \left\|T_e^{p_1p_2p_3}(0)\right\| + \left\|T_e^{p_1p_2p_3}(1)\right\|$ $= \left\|T_e^{p_1p_2}(0) \cap T_e^{p_3}(0)\right\| + \left\|T_e^{p_1p_2}(1) \cap T_e^{p_3}(0)\right\|$ $+ \left\|T_e^{p_1p_2}(0) \cap T_e^{p_3}(1)\right\|$
	$\frac{2}{3} \le \varepsilon_r < 1$	2	$\sup_a(p_1p_2p_3) = \left\|T_a^{p_1p_2p_3}(2)\right\|$ $= \left\|\bigcup_{i=0}^{2} T_e^{p_1p_2p_3}(i)\right\| = \sum_{i=0}^{2}\left\|T_e^{p_1p_2p_3}(i)\right\|$ $= \left\|T_e^{p_1p_2p_3}(0)\right\| + \left\|T_e^{p_1p_2p_3}(1)\right\| + \left\|T_e^{p_1p_2p_3}(2)\right\|$ $= \left\|T_e^{p_1p_2}(0) \cap T_e^{p_3}(0)\right\| + \left\|T_e^{p_1p_2}(1) \cap T_e^{p_3}(0)\right\|$ $+ \left\|T_e^{p_1p_2}(0) \cap T_e^{p_3}(1)\right\| + \left\|T_e^{p_1p_2}(2) \cap T_e^{p_3}(0)\right\|$ $+ \left\|T_e^{p_1p_2}(1) \cap T_e^{p_3}(1)\right\|$
	$\varepsilon_r = 1$	3	$\sup_a(p_1p_2p_3) = \left\|TDB\right\|$

圖 3.3(續) 項目集長度為 1 到 4 之容錯支持度運算範例

$\{p_1, p_2, p_3, p_4\}$	$0 \le \varepsilon_r < \frac{1}{4}$	0	$\sup_a(p_1p_2p_3p_4) = \left\lvert T_e^{p_1p_2p_3p_4}(0)\right\rvert$ $= \left\lvert T_e^{p_1}(0) \cap T_e^{p_2}(0) \cap T_e^{p_3}(0) \cap T_e^{p_4}(0)\right\rvert$
	$\frac{1}{4} \le \varepsilon_r < \frac{1}{2}$	1	$\sup_a(p_1p_2p_3p_4) = \left\lvert T_a^{p_1p_2p_3p_4}(1)\right\rvert$ $= \left\lvert \bigcup_{i=0}^{1} T_e^{p_1p_2p_3p_4}(i)\right\rvert = \sum_{i=0}^{1} \left\lvert T_e^{p_1p_2p_3p_4}(i)\right\rvert$ $= \left\lvert T_e^{p_1p_2p_3p_4}(0)\right\rvert + \left\lvert T_e^{p_1p_2p_3p_4}(1)\right\rvert$ $= \left\lvert T_e^{p_1p_2p_3}(0) \cap T_e^{p_4}(0)\right\rvert + \left\lvert T_e^{p_1p_2p_3}(1) \cap T_e^{p_4}(0)\right\rvert$ $+ \left\lvert T_e^{p_1p_2p_3}(0) \cap T_e^{p_4}(1)\right\rvert$
	$\frac{1}{2} \le \varepsilon_r < \frac{3}{4}$	2	$\sup_a(p_1p_2p_3p_4) = \left\lvert T_a^{p_1p_2p_3p_4}(2)\right\rvert$ $= \left\lvert \bigcup_{i=0}^{2} T_e^{p_1p_2p_3p_4}(i)\right\rvert = \sum_{i=0}^{2} \left\lvert T_e^{p_1p_2p_3p_4}(i)\right\rvert$ $= \left\lvert T_e^{p_1p_2p_3p_4}(0)\right\rvert + \left\lvert T_e^{p_1p_2p_3p_4}(1)\right\rvert + \left\lvert T_e^{p_1p_2p_3p_4}(2)\right\rvert$ $= \left\lvert T_e^{p_1p_2p_3}(0) \cap T_e^{p_4}(0)\right\rvert + \left\lvert T_e^{p_1p_2p_3}(1) \cap T_e^{p_4}(0)\right\rvert$ $+ \left\lvert T_e^{p_1p_2p_3}(0) \cap T_e^{p_4}(1)\right\rvert + \left\lvert T_e^{p_1p_2p_3}(2) \cap T_e^{p_4}(0)\right\rvert$ $+ \left\lvert T_e^{p_1p_2p_3}(1) \cap T_e^{p_4}(1)\right\rvert$

圖 3.3(續) 項目集長度為 1 到 4 之容錯支持度運算範例

$\{p_1, p_2, p_3, p_4\}$	$\frac{3}{4} \le \varepsilon_r < 1$	3	$\sup_a(p_1p_2p_3p_4) = \left\|T_a^{p_1p_2p_3p_4}(3)\right\|$ $= \left\|\bigcup_{i=0}^{3} T_e^{p_1p_2p_3p_4}(i)\right\| = \sum_{i=0}^{3}\left\|T_e^{p_1p_2p_3p_4}(i)\right\|$ $= \left\|T_e^{p_1p_2p_3p_4}(0)\right\| + \left\|T_e^{p_1p_2p_3p_4}(1)\right\| + \left\|T_e^{p_1p_2p_3p_4}(2)\right\|$ $+ \left\|T_e^{p_1p_2p_3p_4}(3)\right\|$ $= \left\|T_e^{p_1p_2p_3}(0) \cap T_e^{p_4}(0)\right\| + \left\|T_e^{p_1p_2p_3}(1) \cap T_e^{p_4}(0)\right\|$ $+ \left\|T_e^{p_1p_2p_3}(0) \cap T_e^{p_4}(1)\right\| + \left\|T_e^{p_1p_2p_3}(2) \cap T_e^{p_4}(0)\right\|$ $+ \left\|T_e^{p_1p_2p_3}(1) \cap T_e^{p_4}(1)\right\| + \left\|T_e^{p_1p_2p_3}(3) \cap T_e^{p_4}(0)\right\|$ $+ \left\|T_e^{p_1p_2p_3}(2) \cap T_e^{p_4}(1)\right\|$
	$\varepsilon_r = 1$	4	$\sup_a(p_1p_2p_3p_4) = \left\|TDB\right\|$

圖 3.3(續) 項目集長度為 1 到 4 之容錯支持度運算範例

【範例 3_4】

以表 3.1 所示的範例資料庫為例，給定誤差容忍參數 $\varepsilon_r = 0.5$，若 $P = \{A, B\}$，則其最大比對錯誤值為1。對照圖 3.3 中所示，$\sup_a(AB) = \left|T_e^A(0) \cap T_e^B(0)\right| + \left|T_e^A(0) \cap T_e^B(1)\right| + \left|T_e^A(1) \cap T_e^B(0)\right|$，其中必須找出三個集合，第一項 $T_e^A(0) \cap T_e^B(0)$ 要找出資料庫中有包含 A 又有 B 的交易資料，因此 $T_e^A(0) \cap T_e^B(0) = \{T2, T3, T4, T8\}$。第二項 $T_e^A(0) \cap T_e^B(1)$ 中又分為兩項，$T_e^B(1)$ 表示在比對 B 時比對錯誤值為1的交易資料所成的集合，也就是找出資料庫中不包含B的交易資料，因此為{ T1, T5, T6 }。$T_e^A(0)$ 表示資料庫中包含A的交易資料{ T1, T2, T3, T4, T5, T8 }，兩者交集得到的結果為{ T1, T5 }，也就是代表在包含A的情況下不包含B的交易資料集。第三項

則表示要找出資料庫中不包含 A 但包含 B 的資料筆數，得到 $\left|T_e^A(1) \cap T_e^B(0)\right| = \left|\{T7\}\right| = 1$。由於此三項中的集合彼此間不會有交集，因此 $\sup_a(AB) = \left|T_e^A(0) \cap T_e^B(0)\right| + \left|T_e^A(0) \cap T_e^B(1)\right| + \left|T_e^A(1) \cap T_e^B(0)\right| = 4 + 2 + 1 = 7$，可得 $\sup_a(AB)$ 為 7。

3-2.2 以 FP-tree 的資訊計算項目集容錯包含支持度

根據圖 3.3 中顯示的計算方式為例，當誤差容忍參數 ε_r 為 0.5 時，$P = \{p_1, p_2\}$ 的最大比對錯誤值為 1，要得知此項目集容錯支持度必須取得 $\left|T_e^{p_1}(0) \cap T_e^{p_2}(0)\right|$、$\left|T_e^{p_1}(0) \cap T_e^{p_2}(1)\right|$ 及 $\left|T_e^{p_1}(1) \cap T_e^{p_2}(0)\right|$ 之值：第一部份 $\left|T_e^{p_1}(0) \cap T_e^{p_2}(0)\right|$ 是指在資料庫中同時包含 p_1 與 p_2 的支持度，以 FP-tree 的儲存結構來看，透過項目 p_1 對 FP-tree 做**投影**(projection)後，可以取出 FP-tree 中與項目 p_1 出現相關的 p_1-conditional FP-tree，再以項目 p_2 對 p_1-conditional FP-tree 做投影可以得到 p_1p_2-conditional FP-tree，藉由樹上根節點儲存的計數值，即可獲得 p_1p_2 在資料庫中的支持度。第二部分 $\left|T_e^{p_1}(0) \cap T_e^{p_2}(1)\right|$ 則表示在包含 p_1 的情形下不包含 p_2 的支持度。為從 FP-tree 的儲存結構中得到此資訊，以項目 p_1 對 FP-tree 投影後，得到 p_1-conditional FP-tree，本論文擴展對 FP-tree 做投影的方法，再以 $\overline{p}_2$ 對 p_1-conditional FP-tree 投影得到 $p_1\overline{p}_2$-conditional FP-tree，同樣由樹上根節點的計數值獲得 $p_1\overline{p}_2$ 在資料庫中的支持度。第三部分 $\left|T_e^{p_1}(1) \cap T_e^{p_2}(0)\right|$ 表示在不包含 p_1 的情形下包含 p_2 的交易資料筆數，則以 $\overline{p}_1$ 對 FP-tree 執行投影，取得 FP-tree 中不包含 p_1 的項目集資料，建立 $\overline{p}_1$-conditional FP-tree。接著以 p_2 對 $\overline{p}_1$-conditional FP-tree 進行投影，得到 $\overline{p}_1p_2$-conditional FP-tree，即可從樹上根節點計數值獲得 $\left|T_e^{p_1}(1) \cap T_e^{p_2}(0)\right|$ 之值。將在以

下章節中說明本論文以項目對 FP-tree 執行投影的執行細節與對應儲存資訊。

3-2.3 以項目對 FP-tree 做投影

為節省探勘處理時間,本論文方法中以某個項目對 FP-tree 進行投影時並不會完全重新建構出新的樹狀結構,只會重建 conditional Header Table 及根節點。如同上節內容所述,要獲得項目集在資料庫中的容錯支持度,除了需對某一項目 x 做投影得到包含該項目 x 的 conditional FP-tree 外,還要找出 FP-tree 上不包含該項目(以 $\bar{x}$ 表示)的 conditional FP-tree。因此以下將分別介紹以 x 與 $\bar{x}$ 對 FP-tree 執行投影的處理方式:

(一)以 x 對 FP-tree 做投影:

以 x 對 FP-tree 做投影,即取出 FP-tree 上以 x 為根節點的子樹。由 FP-tree 的 Header Table 中對應到項目 x 的資料,透過其 link 欄儲存的指標所指向的串列,即可快速的取出儲存項目 x 之節點。以項目 x 之節點為根節點,而根節點以下的子樹部分即為原 FP-tree 經項目 x 投影後所得之 x-conditional FP-tree。

以 x 對 FP-tree 做投影時,原 FP-tree 中儲存項目 x 的節點若有一個以上,為避免進行節點合併的處理成本,本方法會直接將這些節點保留成多個根節點串連起來,形成 x-conditional FP-forest。為了不破壞 FP-tree 節點間原本既有的橫向連結,所以在投影時會對每個項目 x 的節點 N_x 建立一個新節點,儲存 N_x 所包含之計數值,並複製 N_x 的子節點連接(child node link)。對於 x-conditional FP-forest 上各節點儲存的項目與計數值,也會建立一個 conditional Header Table。FP-forest 中各

棵FP-tree的節點資訊都會記錄在同一個 Header Table 中，每個項目的 link 欄儲存連結指標，連結 x-conditional FP-forest 中具有相同項目的節點。

【範例 3_5】

根據圖 3.2 中利用表 3.1 範例資料庫 TDB 所建構出的 FP-tree 與 Header Table (如圖 3.4 (a))，若以項目 A 對該 FP-tree 進行投影，得到 A-conditional FP-tree 的結果如圖 3.4 (b)所示。

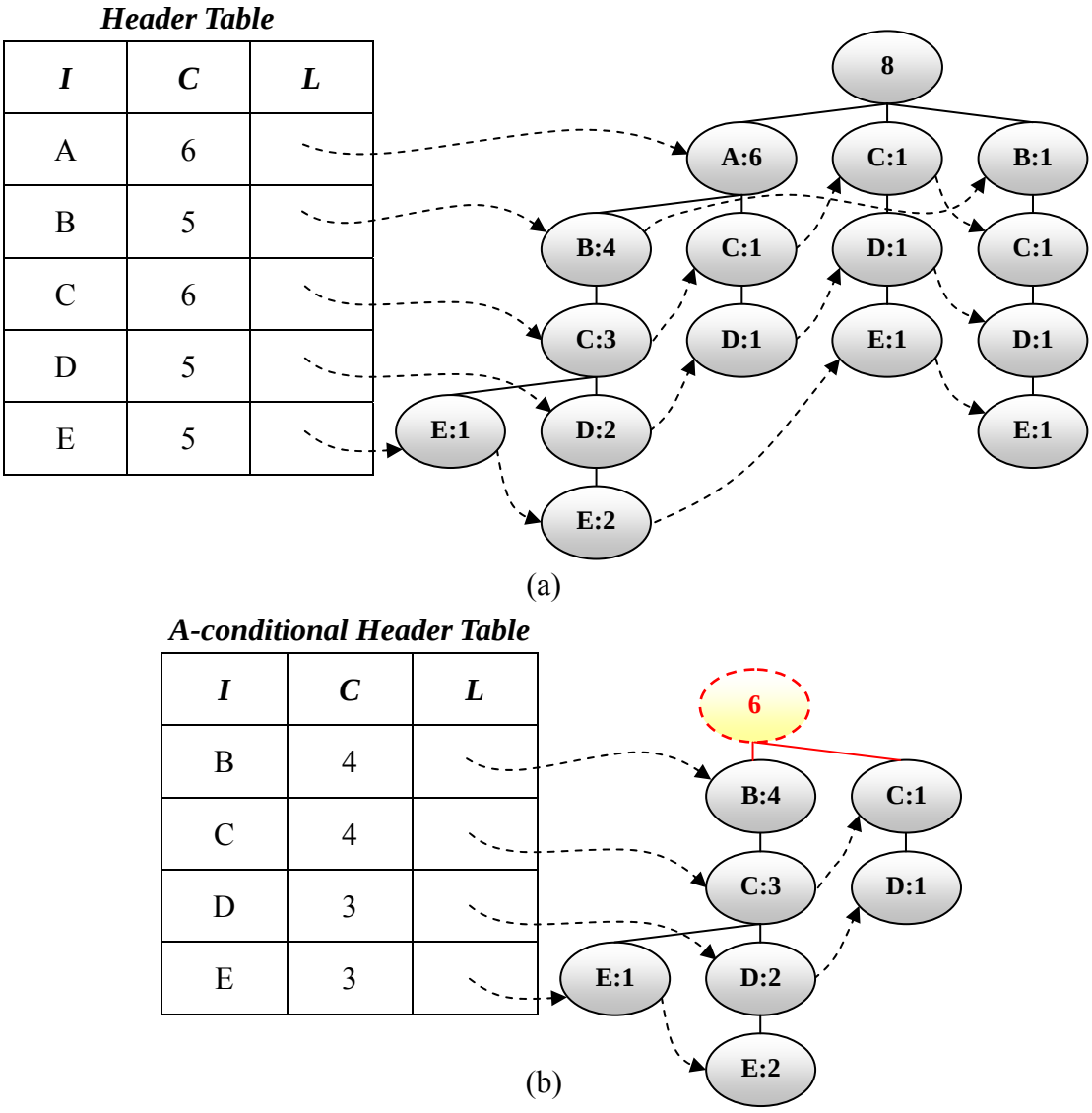


圖 3.4 以項目 A 做投影後所得之 A-conditional FP-tree 與 Header Table

在範例 3_5 中項目 A 對應的節點只有1個，因此建立1個新的節點複製原節點上計數值資訊，並複製原節點之子節點指標指向原節點的子樹，而子樹結構中的節點及連結皆不需重建。

若再繼續對此棵 A-conditional FP-tree 以項目 C 做投影，則取出兩棵以項目 C 為根節點的 AC-conditional FP-tree ，建立兩個複製原項目 C 對應節點，再將兩節點串聯在一起即可得到 AC-conditional FP-forest，其結果如圖 3.5 所示。

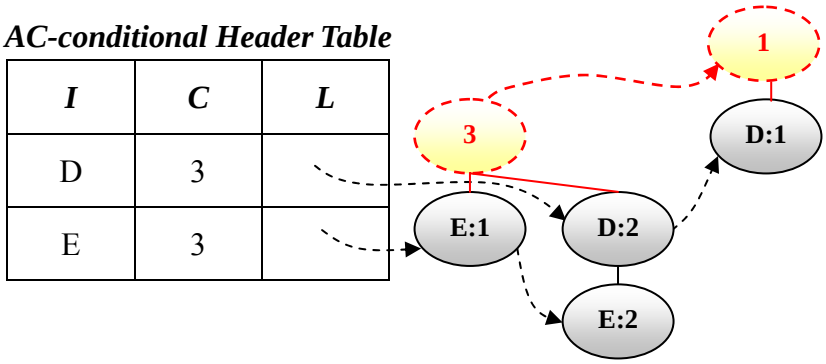


圖 3.5 繼續以項目 C 做投影後所得之 AC-conditional FP-forest 與 Header Table

在此範例中，以項目 C 對 A-conditional FP-tree 做投影後，需額外儲存的資訊只有兩個複製的項目 C 節點及 AC-conditional Header Table 。兩棵 AC-conditional FP-tree 之根節點下的子樹中的節點，則可直接連接到原 FP-tree 中的節點，節省許多重新建立 conditional FP-tree 的時間與空間。

(二)以 $\bar{x}$ 對 FP-tree 做投影：

以 $\bar{x}$ 對 FP-tree 進行投影，目的在於取出 FP-tree 中不包含 x 的子樹部分。在建構 FP-tree 時，每筆交易資料以項目的字元順序排序，因此以項目 $\bar{x}$ 做投影時，所

取出的$\bar{x}$-conditional FP-tree，只包含字元順序在x之後的項目節點。若字元順序在x後的項目有$y_1, y_2, \ldots, y_K$ ($y_1 < y_2 < \ldots < y_K$)，則從Header Table中y_1的橫向連結一一搜尋，檢查每個行經節點N_{y_1}，若節點N_{y_1}之祖先節點(含父節點)中包含x，則表示該項目集的出現伴隨x，不是以$\bar{x}$做投影所要擷取的部分；否則以N_{y_1}為根節點的子樹即為$\bar{x}$-conditional FP-tree的一部份。對於其他項目$y_i (i \geq 2)$，同樣透過Header Table中y_i的橫向連結一一取出節點N_{y_i}，若N_{y_i}的祖先節點中皆不包含x，則以N_{y_i}為根節點的子樹亦為$\bar{x}$-conditional FP-tree的一部份；然而若N_{y_i}的任一祖先節點N_{y_j}包含y_j ($j<i$)，則代表該子樹被包含於節點N_{y_j}之下的子樹中且已被取出，因此不需要重複被取出。

透過上述走訪項目$y_i (1 \leq i \leq k)$對應節點的方式，找到對$\bar{x}$進行投影所要取出子樹之根節點，接著如同對x進行投影的處理，對這些節點各建一個新節點複製原節點上的項目、計數值及子節點連結，彼此間再串連起來。這些節點皆成為$\bar{x}$-conditional FP-tree根節點的子節點。$\bar{x}$-conditional FP-tree的根節點中不記錄任何項目，但會儲存一個計數值表示不包含x的交易資料筆數。若原FP-tree中根節點所記錄的計數值為m，表示此棵FP-tree上共記錄m筆交易資料，從原Header Table中記錄項目x的計數值，可取得在FP-tree包含項目x的交易筆數值n，因此$\bar{x}$-conditional FP-tree中根節點的計數值記錄為$m - n$。

【範例 3_6】

以表 3.1 中範例資料庫 TDB 為例，利用圖 3.4(a)所示之 FP-tree 與 Header Table 資訊，以項目 $\overline{B}$ 對該棵 FP-tree 進行投影，結果如圖 3.6 所示。

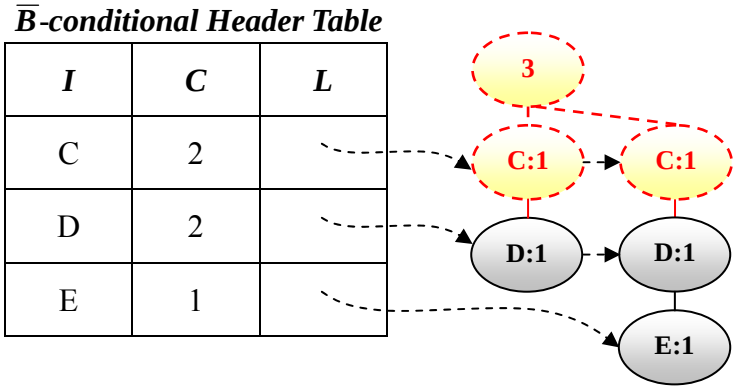


圖 3.6 以項目 $\overline{B}$ 做投影後所得之 $\overline{B}$-conditional FP-tree 與 Header Table

投影過程會搜尋字元順序在 B 之後的項目節點，圖 3.4 (a) 中找到兩個項目 C 節點其祖先節點中皆不包含 B ，其子樹皆屬於 $\overline{B}$-conditional FP-tree 中的子樹，因此建立兩個新節點，複製這兩個項目 C 節點的計數值，設定指標連結指向原節點的子節點上，並設成 $\overline{B}$-conditional FP-tree 根節點的子節點。而 $\overline{B}$-conditional FP-tree 根節點上的計數值則根據原 FP-tree 根節點中的計數值 8 (表示資料庫中有 8 筆交易資料)，以及原 Header Table 上項目 B 的計數值 5，設定為(8－5＝3)，表示有 3 筆交易資料中不包含項目 B 。

本論文中執行 FP-tree 的投影以複製投影後之子樹根節點的方式取代重新建樹，主要原因是觀察到投影的動作只是取出原 FP-tree 的一部份，如果每做一次投影就要重新建立一棵樹，將耗費大量處理時間及儲存記憶體空間。需複製投影後

之子樹根節點的主要考量是避免更動到原始FP-tree的結構。如圖3.4所示，以項目A對原始FP-tree做投影，得到的A-conditional FP-tree上項目為C的節點僅剩兩個，但在範例3_3建構原FP-tree與Header Table時，包含項目C的節點共有4個，並以橫向連結的方式串聯在一起。若建立AC-conditional FP-tree時不另外對項目為C的節點建立具相同資訊的新節點，直接以原節點更改其橫向連結，則原FP-tree中項目C的橫向連結串列會被破壞，一旦原FP-tree結構與Header Table受到破壞，將使得後續以其他項目對FP-tree做投影時得到不正確的結果。因此保留讀取資料庫後所建構的FP-tree，透過複製投影後之子樹根節點的方式進行投影，可利用較少的儲存空間達到投影的效果，並有效增進探勘效率。

3-3 FP-AFI 演算法

FP-AFI 演算法是根據 3.1 節建立的 FP-tree 樹狀結構，以及 3.2 節所述以 FP-tree 結構計算容錯支持度原理來進行近似常見項目集探勘，以下將介紹 FP-AFI 探勘過程的儲存結構及處理流程。

3-3.1 項目集之儲存結構

在探勘過程，FP-AFI 會將用來計算一個候選項目集 P 之容錯支持度所需的 conditional FP-trees 及 conditional Header Tables 記錄下來，以便遞迴探勘更長的項目集。其儲存結構包含 5 個部分：

(1)***I*** (itemsets)：記錄項目集 P ；

(2)***FT*** (FP-tree list)：用來指向對應到 $\bigcup_{i=0}^{\mu_P} T_e^P(i)$ 的各 conditional FP-tree；

(3)***HT*** (Header Table list)：儲存對應到 conditional FP-tree list 中各棵 conditional FP-tree 的 conditional Header Table；

(4)***N*** (FP-tree vector)：以一個長度為 $|P|$ 的二元字串對各 conditional FP-tree 進行編碼，令 $P = \{p_1, p_2, \ldots, p_n\}$，若該 conditional FP-tree 是由對 p_i 做投影產生，則位元 i 設為 1，若是對 $\overline{p}_i$ 做投影產生，則位元 i 設為 0。

(5) ***P*** (parent node)：儲存連結指標指向此節點的父節點，即對應到項目集 $\{p_1, p_2, \ldots, p_{n-1}\}$ 之節點。

此項目集儲存結構的儲存形式，將在後續內容的範例中以圖示說明。

3-3.2 FP-AFI 演算法流程

給定演算法中需設定的參數值，包括ε_r、ε_c、α及$\min_sup$，並根據資料庫大小計算出最小支持度計數門檻值$\min_sup_c$(也就是$\lceil |TDB| * \min_sup \rceil$)。FP-AFI 演算法主要分成以下幾個步驟：

步驟<1>：讀取資料庫，統計各項目之支持度，找出交易資料中不滿足核心樣式門檻值($\min\_sup_c * \alpha$)的項目。

步驟<2>：再次讀取資料庫，忽略支持度小於核心樣式門檻值的項目，建立 FP-tree 樹狀結構與 Header Table，並設$P' = \phi$。

步驟<3>：依次取P'-conditional Header Table 中的項目 x，若該項目之計數值大於或等於$\min_sup_c * \alpha$，$P = P' \cup \{x\}$，表示 P 滿足近似常見項目集條件三，並進行下一步驟，否則繼續取下一項目檢查。若 Header Table 中已沒有下一項目，則回到遞迴呼叫此步驟的上一層執行。

步驟<4>：比較 P 及P'的最大比對錯誤值，若$\mu_P = \mu_{P'}$，則執行步驟<4-1>；若$\mu_P = \mu_{P'} + 1$，表示在項目集增加 1 個項目後，可容錯項目個數發生改變，因此執行步驟<4-2>。

步驟<4-1>：由計算容錯包含支持度之函數定義，$\sup_a(P) = \sum_{i=0}^{\mu_P} |T_e^P(i)|$，其中$T_e^P(i)$即對應到 P 的 conditional FP-trees 中編碼包含 i 個 0 的 FP-trees。由於$\mu_P = \mu_{P'}$，因此對P'來說已建立出對應到$T_e^{P'}(i)$ ($0 \le i \le \mu_{P'}$)之 conditional FP-trees。由$T_e^P(i) =$

$(T_e^{P'}(i) \cap T_e^{\{p_n\}}(0)) \cup (T_e^{P'}(i-1) \cap T_e^{\{p_n\}}(1))$ 的關係式，$T_e^{P}(i)$ 之 conditional FP-tree 是以 p_n 對 P′-conditional FP-trees 中編碼包含 i 個 0 的 trees 進行投影，以及以 $\overline{p}_n$ 對 P′-conditional FP-trees 中編碼包含$(i-1)$個 0 的 trees 進行投影而得。

步驟<4-2>：當 $\mu_P = \mu_{P'} + 1$，表示 $T_e^{P'}(\mu_P)$ 對應的 FP-tree 尚未被建立，根據遞迴函數定義，回到 P′之父節點 P″建立 $T_e^{P''}(\mu_P)$，直到回到其祖先節點 P^k 可建出 $T_e^{P^k}(\mu_P)$，再逐層往下建立比對錯誤值為 μ_P 的 conditional FP-tree，直到建立出 $T_e^{P'}(\mu_P)$ 之 conditional FP-tree 後，即可採取與步驟<4-1>相同的做法，建立出對應到 $T_e^{P}(i)$ $(i = 0,...,\mu_P)$ 之 conditional FP-trees。

步驟<5>：利用步驟<4>得到 $T_e^{P}(i)$ $(i = 0,...,\mu_P)$ 之 conditional FP-trees，根據 FP-tree 根節點上儲存的計數值加總後即可計算出 P 的容錯支持度，檢查是否滿足近似常見項目集條件一。若不滿足條件一則跳至步驟<7>，否則繼續執行步驟<6>。

步驟<6>：檢查項目集 P 中每個項目 p_i 之項目支持度是否滿足誤差容忍參數 ε_c 的限制。步驟<4>得到的 conditional FP-tree，若其根節點計數值為 c，其編碼為 $b_1, b_2, ..., b_n$，則項目 p_i 在此 tree 中出現的次數為 $b_i * c$。加總項目 p_i 在各 conditional FP-tree 的出現次數，即可計算出項目 p_i 在容錯包含 P 的交易資料中之項目支持度，並檢查是否滿足近似常見項目集條件

二。若條件二亦滿足，則P為一個近似常見項目集。

步驟<7>：將P′設為P ，並遞迴執行步驟<3>。

【範例 3_8】

本範例繼續採用前章節所列之範例資料庫 TDB，如表 3.2 所示，來說明整個 FP-AFI 演算法在探勘近似常見項目集的處理過程。假設各參數訂為 $\varepsilon_r = 0.4$ 、$\varepsilon_c = 0.5$ 、$\alpha = 0.5$ 、$\min\_sup = 0.5$ ，並計算出 $\min\_sup_c = \lceil 8 * 0.5 \rceil = 4$ ，核心樣式門檻值為 $\min\_sup_c * \alpha = 2$ 。

表 3.2 範例資料庫 TDB

交易編號	交易資料項目集
T1	AF
T2	AB
T3	ABCE
T4	ABCDE
T5	ACD
T6	CDE
T7	BCDE
T8	ABCDEG

步驟<1>：掃瞄資料庫一次，統計資料庫中每個項目的支持度，項目 F 與 G 項目支持度為 1，小於核心樣式門檻值。

步驟<2>：建立 FP-tree 與 Header Table 時忽略項目支持度小於核心樣式門檻值的項目 F 和 G，所建構資料結構如圖 3.7 所示。

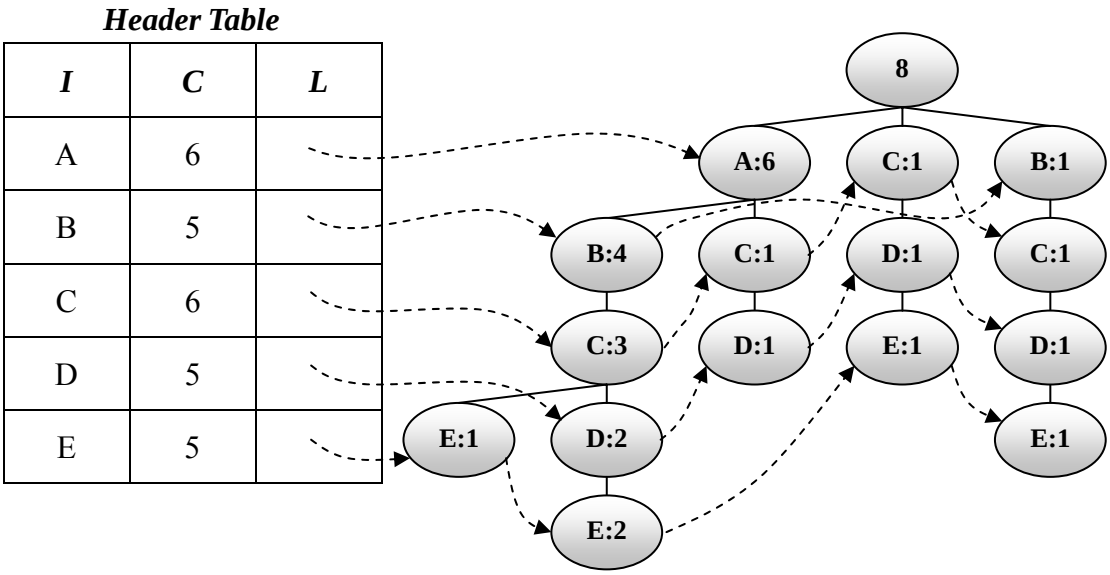
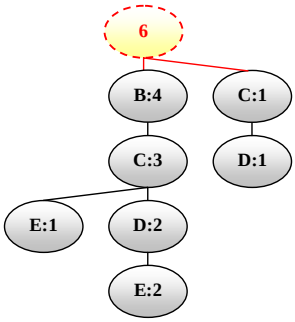


圖 3.7 範例資料庫 TDB 之 FP-tree 結構與 Header Table

I	A
FT	[0] 
HT	[0]
N	1
P	Null

I	*C*	*L*
B	4	
C	4	
D	3	
E	3	

圖 3.8 以項目 A 做投影後所得之 A-conditional FP-tree 與 Header Table

從 Header Table 中取出第一個支持度大於 $min\_sup_c * \alpha$ 的項目 A，以項目 A 組成長度 1 的項目集。由於 $\mu_A = \lfloor 1 * 0.4 \rfloor = 0$，且 $sup_a^A(0) = \left|T_a^A(0)\right| = \left|T_e^A(0)\right|$，因此以項目 A 對 FP-tree 做投影得到 A-conditional FP-tree 與 Header Table 如圖 3.8 所示，其編碼為 1。從根節點中之計數值可得 $sup_a^A(0) = \left|T_e^A(0)\right| = 6$，大於 min_sup_c，因此 A 為一個近似常見項目集。接下來遞迴執行步驟<3>，由 A 的 conditional Header Table，取出第一個計數值大於 $min\_sup_c * \alpha$ 的項目 B，其支持度為 4，代表項目集 AB 在資料庫中的支持度 sup(AB)=4，滿足核心樣式門檻值(條件三)，形成長度 2 的候選項目集。此時 $\mu_{AB} = \lfloor 2 * 0.4 \rfloor = 0$，$sup_a^{AB}(0) = \left|T_a^{AB}(0)\right| = \left|T_e^{AB}(0)\right| = \left|T_e^A(0) \cap T_e^B(0)\right|$，因此藉由項目 B 對 A 的 conditional FP-tree 做投影，得到 AB-conditional FP-tree，代表交易資料中同時包含項目 AB 的項目集形成之樹狀結構。該棵 FP-tree 記錄同時出現項目 A 與 B 的交易資料，因此將其編碼為 11，如圖 3.9 所示，其對應儲存結構如圖 3.10 所示。由其根節點之計數值可得 $sup_a^{AB}(0) = \left|T_e^{AB}(0)\right| = 4 \geq min\_sup_c$，滿足近似常見項目集的第一個條件要求。接下來檢查項目集 AB 中各項目的項目支持度是否滿足 ε_c 的容錯限制，由於 $T_a^{AB}(0) = T_e^{AB}(0)$，因此以儲存結構中編號 11 之 FP-tree 與其對應的計數值 4，可分別計算出項目支持度 $sup_{item}^{AB}(A) = 4 * 1 = 4$ 及 $sup_{item}^{AB}(B) = 4 * 1 = 4$，皆大於 2 ($\lceil 4 * (1 - 0.5) \rceil = 2$)，則滿足前述定義的三個近似常見項目集之條件，因此 AB 為一個近似常見項目集。

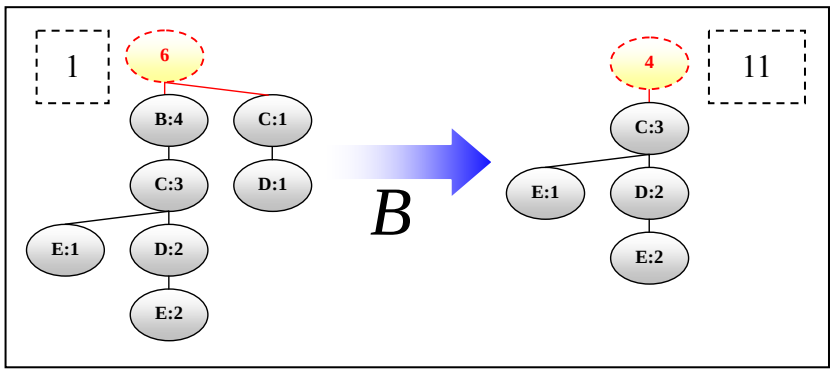
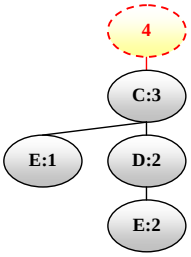


圖 3.9 以項目 B 對 FP-tree 做投影所得之 conditional FP-tree

I	AB
FT	[0] 
HT	[0]
N	11
P	A

I	***C***	***L***
C	3	
D	2	
E	3	

圖 3.10 項目集 AB 之資料儲存結構範例

繼續遞迴執行步驟<3>，由 AB 產生更長的項目集進行探勘。從 AB-conditional Header Table 取出第一個計數值大於 $\min\_sup_c * \alpha$ 的項目 C ，形成候選項目集 ABC。由於此範例之誤差容忍參數 ε_r 訂為 0.4，當候選項目集長度為 3 時，最大比對錯誤值為 $\mu_{ABC} = \lfloor 3 * 0.4 \rfloor = 1$。然而 $\mu_{ABC} = \mu_{AB} + 1$，最大比對錯誤值發生改變，與上述情況不同，需執行 FP-AFI 演算法的步驟<4-2>。由 3-2 節之等式 4，欲得到

$T_e^{ABC}(1)$ 必須先有 $T_e^{AB}(1)$ 的資訊，但 $T_e^{AB}(1)$ 對應的 FP-trees 尚未被建立，根據遞迴函數定義必須回到項目集 AB 之父節點項目集 A，獲得 $T_e^{A}(1)$ 資訊。由 3-2 節之等式 6，當項目集長度為 1 且最大比對錯誤值為 1，則 $T_e^{A}(1) = TDB - T_e^{A}(0)$，表示找出資料庫中不包含 A 的交易資料集。以項目 $\overline{A}$ 對原 TDB 之 FP-tree 投影，即可得到資料庫中不包含項目 A 之交易資料集所構成的 $\overline{A}$-conditional FP-tree，如圖 3.11 所示。而項目集 A 的儲存結構中原本只儲存編碼為 1，為包含項目 A 的 conditional FP-tree，$\overline{A}$-conditional FP-tree 亦加入此結構中，其編碼為 0，因此項目集 A 儲存內容更新為圖 3.12 所示(為避免圖形過於複雜，在以下呈現儲存結構內容時皆省略 Header Table 的部分)。

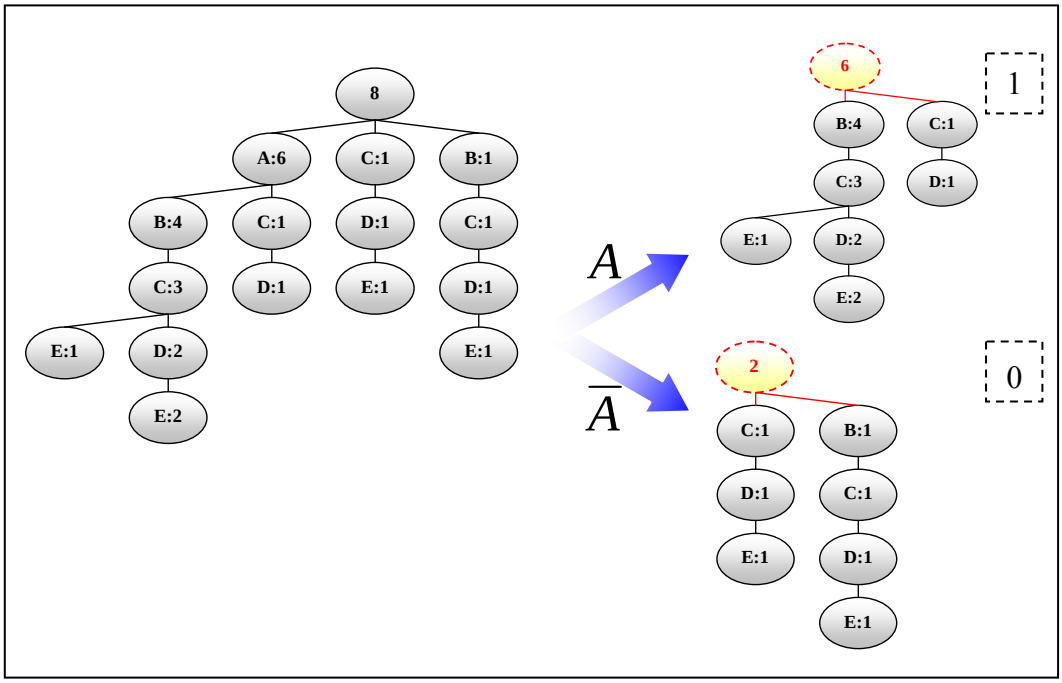


圖 3.11 以項目 A 及 $\overline{A}$ 對 FP-tree 做投影結果

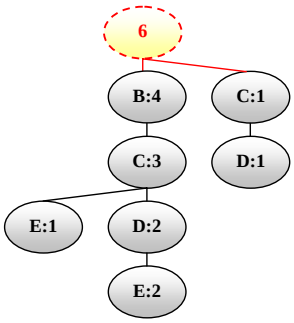
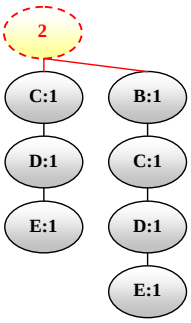
I	A	
FT	[0] 	[1] 
N	1	0
P	Null	Null

圖 3.12 項目集 A 之儲存結構

在建立 $T_e^A(1)$ 對應的 FP-tree 後，即可根據 $T_e^{AB}(1) = (T_e^A(1) \cap T_e^B(0)) \cup (T_e^A(0) \cap T_e^B(1))$ 的關係式建立 $T_e^{AB}(1)$ 所對應的 FP-trees。如圖 3.13 所示，也就是對項目集 A 的 conditional FP-tree 中編碼包含 1 個 0 的樹，以 B 對該 FP-tree 做投影；並對編碼中沒有包含 0 的 conditional FP-tree 以 $\overline{B}$ 對該 FP-tree 進行投影。因此如圖 3.14 所示，項目集 AB 的儲存結構中，新增兩棵編碼分別為 10 與 01 的 FP-tree，即對應到 $T_e^{AB}(1)$。

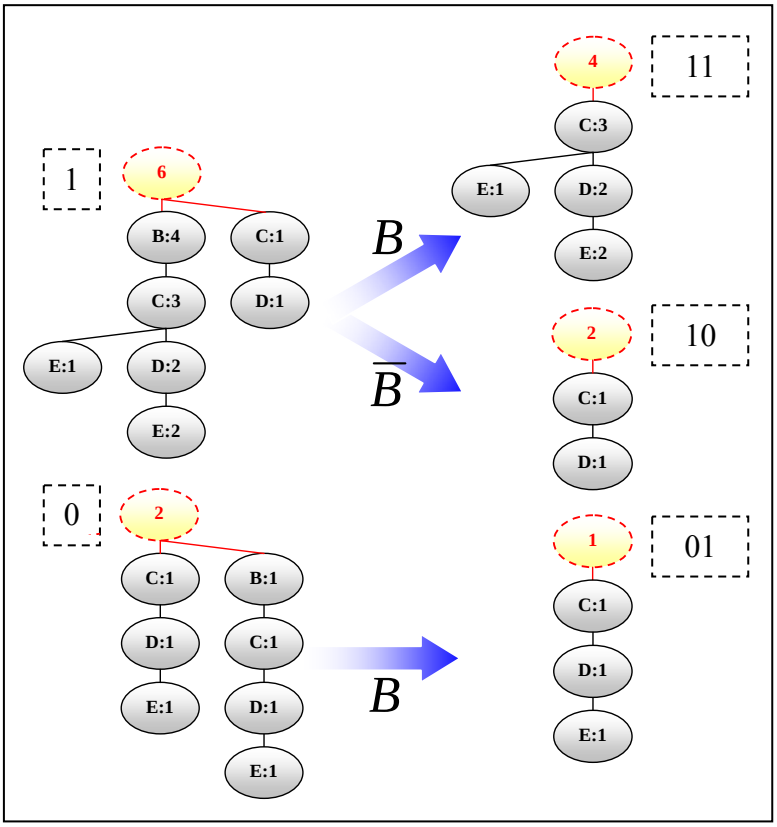


圖 3.13 以項目 B 及 $\overline{B}$ 對 FP-tree 做投影結果

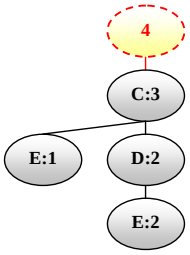
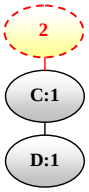
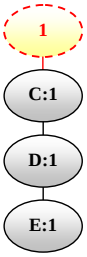
I	AB		
FT	[0]	[1]	[2]
			
N	11	10	01
P	A	A	A

圖 3.14 項目集 AB 之儲存結構

得到 $T_e^{AB}(1)$ 的資訊後，利用編碼為 11、10 及 01 的 FP-trees，如圖 3.15 分別建構出對應到 $T_e^{ABC}(0)$ (對應編碼為 111) 及 $T_e^{ABC}(1)$ (對應到編碼 110、101 及 011) 的 FP-trees，更新後的項目集 ABC 儲存內容如圖 3.16 所示。

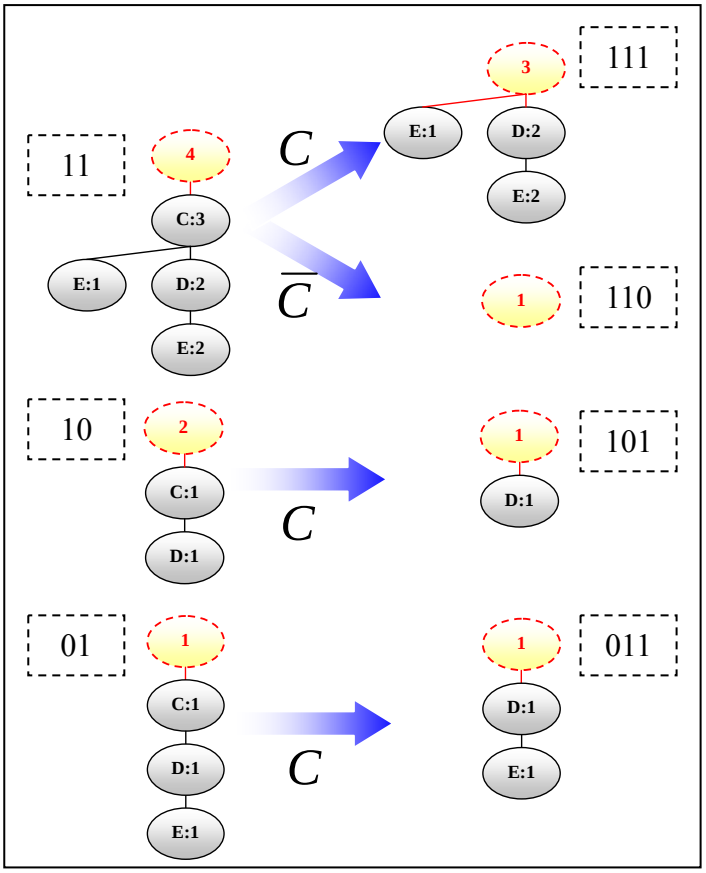


圖 3.15 以項目 C 及 $\overline{C}$ 對 FP-tree 做投影所得之 conditional FP-tree

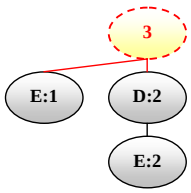
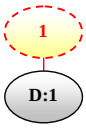
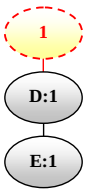
I	ABC			
FT	[0] 	[1] 	[2] 	[3] 
N	111	110	101	011
P	AB	AB	AB	AB

圖 3.16 項目集 ABC 之儲存結構

由圖 3.16 所示對應到 $T_e^{ABC}(0)$ 與 $T_e^{ABC}(1)$ 之 conditional FP-trees，加總各樹上根節點的計數值，即可得到項目集 ABC 在資料庫中的容錯支持度 $\sup_a(ABC)=3+1+1+1=6\geq \min\_\sup_c$，因此可判斷滿足近似常見項目集條件一。

再檢查項目集 ABC 中每個項目之項目支持度是否滿足誤差容忍參數 ε_c 的限制，以項目 A 為例，對應到 FP-tree 編碼的第一個位元值，乘上各 FP-tree 根節點的計數值，經過加總後 $\sup_{item}^{ABC}(A) = (1*3) + (1*1) + (1*1) + (0*1) = 5$。依同樣方式可計算出其他項目之項目支持度分別為 $\sup_{item}^{ABC}(B) = 5$ 及 $\sup_{item}^{ABC}(C) = 5$，皆大於 $\lceil \sup_a(ABC) * (1 - \varepsilon_c) \rceil$(也就是 $\lceil 6 * (1 - 0.5) \rceil = 3$)，因此項目集 ABC 亦滿足近似常見項目集條件二，可判斷出 ABC 為一個近似常見項目集。

執行完上述步驟後，將 P′ 設為 P = { A, B, C }，回到步驟<3>繼續遞迴執行。$\sup(ABC) > \min\_\sup_c * \alpha$，因此考慮由 ABC 產生更長的項目集進行探勘，從項目集 ABC 儲存的 conditional Header Table 中取出第一個計數值等於 $\min\_\sup_c * \alpha$ 的項目 D，將項目 D 加入 ABC 中，形成候選項目集 ABCD。容錯項目個數在項目集長度為 4 時，$\mu_{ABCD} = \lfloor 4 * 0.4 \rfloor = 1$(即 $\mu_{ABCD} = \mu_{ABC}$)。由於最大比對錯誤值沒有發生變化，由 3-2 節之等式 4，欲得到 $T_e^{ABCD}(1)$ 必須先有 $T_e^{ABC}(1)$，由上述的執行已知 $T_e^{ABC}(1)$，因此直接利用圖 3.16 中項目集 ABC 儲存編碼分別為 111、110、101 及 011 的 FP-trees，如圖 3.17 所示建構出對應到 $T_e^{ABCD}(0)$ (對應編碼為 1111)及 $T_e^{ABCD}(1)$ (對應到編碼 1110、1011 及 0111)的 FP-trees，項目集 ABCD 之儲存內容如圖 3.18 所示。

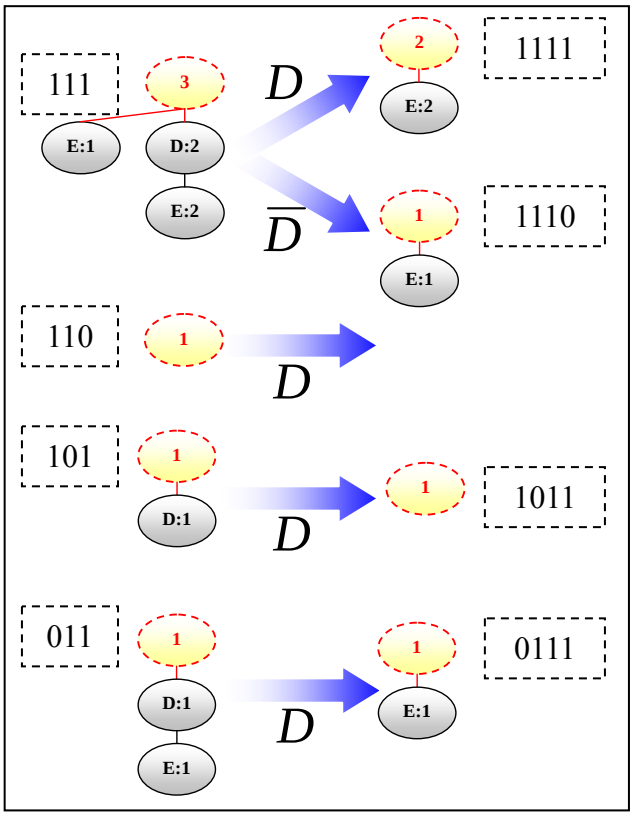


圖 3.17 以項目 D 及 $\overline{D}$ 對 FP-tree 做投影所得之 conditional FP-tree

I	ABCD			
FT	[0]	[1]	[2]	[3]
	2 E:2	1 E:1	1	1 E:1
N	1111	1110	1011	0111
P	ABC	ABC	ABC	ABC

圖 3.18 項目集 ABCD 之儲存結構

由圖 3.18 所示對應到 $T_e^{ABCD}(0)$ 與 $T_e^{ABCD}(1)$ 之 conditional FP-trees，加總樹上根節點的計數值可得到項目集 ABCD 在資料庫中的容錯支持度 $sup_a(ABCD)=5$，大於 min_sup_c，因此可判斷滿足近似常見項目集條件一。按照前述相同的步驟計算項目集 ABCD 中每個項目之項目支持度，得到 $sup_{item}^{ABCD}(A)=4$ 、 $sup_{item}^{ABCD}(B)=4$ 、

$\sup_{item}^{ABCD}(C)=5$ 及 $\sup_{item}^{ABCD}(D)=4$ 皆滿足 ε_c 的限制，因此項目集 ABCD 亦滿足近似常見項目集條件二，可判斷 ABCD 為一個近似常見項目集。

執行完上述步驟後，將 P′ 設為 ABCD ，回到步驟<3>繼續遞迴執行。又可從 ABCD-conditional Header Table 中取出第一個計數值滿足核心樣式門檻值(條件三)的項目 E ，形成候選項目集 ABCDE 。候選項目集長度變為 5 ，最大比對錯誤值 $\mu_{ABCDE}=\lfloor 5*0.4 \rfloor=2$ ，且 $\mu_{ABCDE}=\mu_{ABCD}+1$ 。根據 3-2 節之等式 3 ，該項目集的容錯支持度 $\sup_a(ABCDE)=\left|T_e^{ABCDE}(0)\right|+\left|T_e^{ABCDE}(1)\right|+\left|T_e^{ABCDE}(2)\right|$ ，其中 $T_e^{ABCDE}(0)$ 與 $T_e^{ABCDE}(1)$ 都可由 $T_e^{ABCD}(0)$ 及 $T_e^{ABCD}(1)$ 對應的 FP-trees 得到。但欲得到 $T_e^{ABCDE}(2)$ 必須先有 $T_e^{ABCD}(2)$ 的資訊，然而 $T_e^{ABCD}(2)$ 對應的 FP-tree 尚未被建立，必須根據遞迴函數定義直到回到項目集 ABC 的父節點 AB 找出對應到 $T_e^{AB}(2)$ 的 FP-tree。以項目 $\overline{B}$ 對 $\overline{A}$-conditional FP-tree 進行投影，如圖 3.19 所示。因此在項目集 AB 中加入 $\overline{AB}$-conditional FP-tree，其編碼為 00，則項目集 AB 儲存內容更新如圖 3.20 所示。

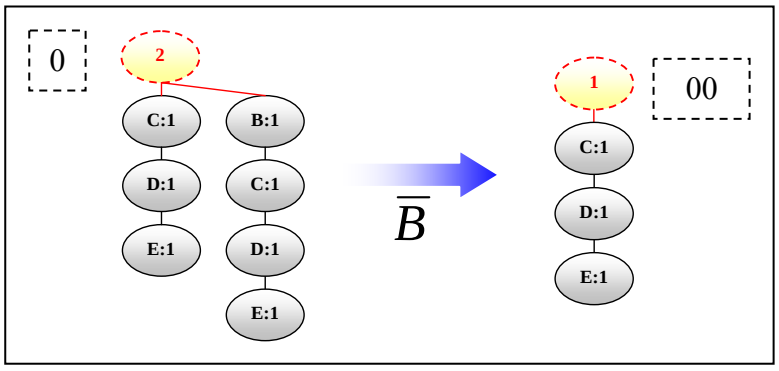


圖 3.19 以項目 $\overline{B}$ 對 FP-tree 做投影所得之 conditional FP-tree

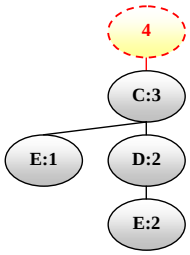
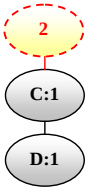
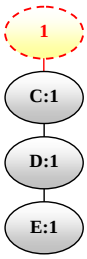
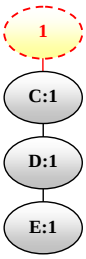
I	AB			
FT	[0]	[1]	[2]	[3]
				
N	11	10	01	00
P	A	A	A	A

圖 3.20 項目集 AB 之儲存結構

在建立 $T_e^{AB}(2)$ 後，即可透過同樣的方式產生 $T_e^{ABC}(2)$ 對應的 FP-trees，如圖 3.21 所示，得到兩棵編碼分別為100及001的 FP-trees。項目集 ABC 的儲存結構則更新如圖 3.22 所示。

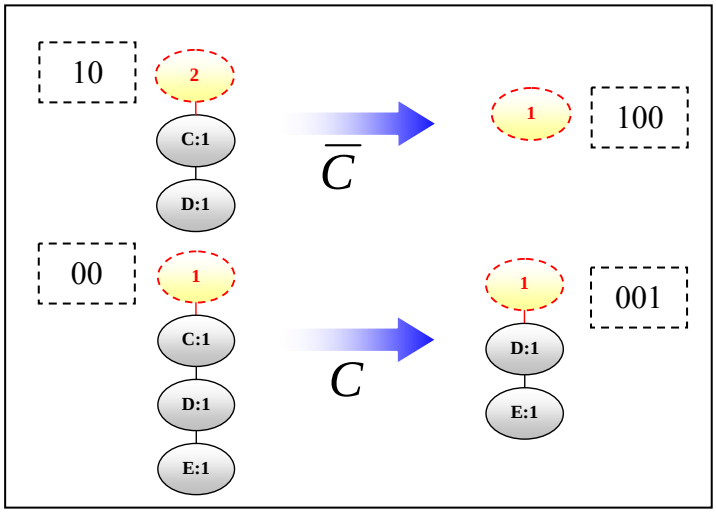


圖 3.21 以項目 C 與 $\overline{C}$ 對 FP-tree 做投影所得之 conditional FP-tree

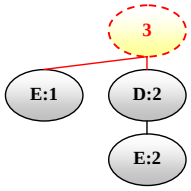
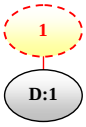
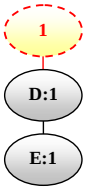
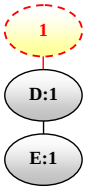
I	ABC					
FT	[0]	[1]	[2]	[3]	[4]	[5]
						
N	111	110	101	011	100	001
P	AB	AB	AB	AB	AB	AB

圖 3.22 項目集 ABC 之儲存結構

重覆上述執行方式，可建構出對應到 $T_e^{ABCD}(2)$ 的 FP-trees (對應編碼為 1100 及 0011)，如圖 3.23 所示。因此項目集 ABCD 的儲存內容更新如圖 3.24 所示。

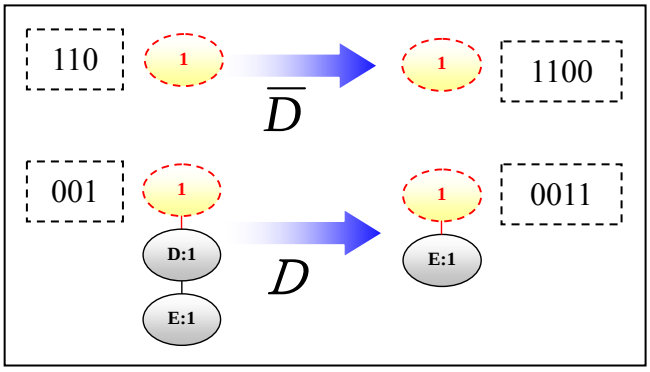


圖 3.23 以項目 $\overline{D}$ 對 FP-tree 做投影所得之 conditional FP-tree

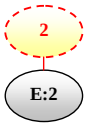
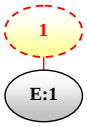
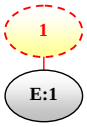
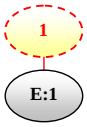
I	ABCD					
FT	[0]	[1]	[2]	[3]	[4]	[5]
						
N	1111	1110	1011	0111	1100	0011
P	ABC	ABC	ABC	ABC	ABC	ABC

圖 3.24 項目集 ABCD 之儲存結構

由上述步驟得到 $T_e^{ABCD}(2)$ 後，即可利用圖 3.24 所列之 FP-tree 資訊分別建構出對應到 $T_e^{ABCDE}(0)$（對應編碼 11111）、$T_e^{ABCDE}(1)$（對應編碼 11101、01111）及 $T_e^{ABCDE}(2)$（對應編碼10110、00111）的 conditional FP-trees，如圖 3.25 所示。項目集 ABCDE 的儲存內容如圖 3.26 所示。

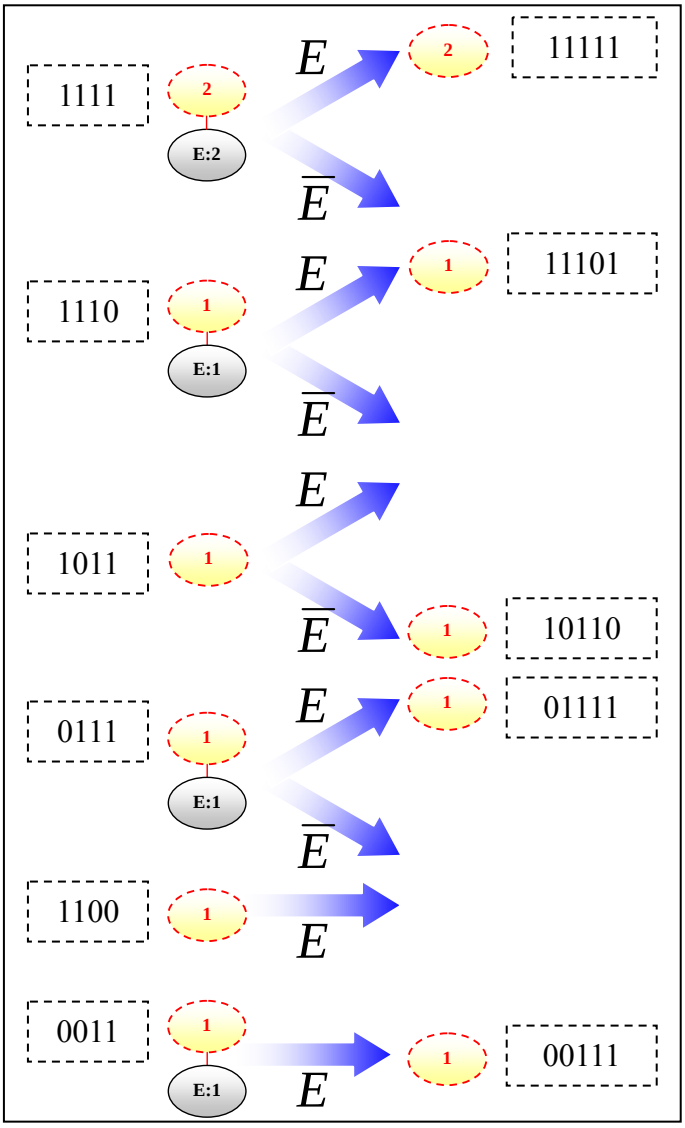


圖 3.25 以項目 E 與 $\overline{E}$ 對 FP-tree 做投影所得之 conditional FP-tree

I	ABCDE				
FT	[0] 2	[1] 1	[2] 1	[3] 1	[4] 1
N	11111	11101	01111	10110	00111
P	ABCD	ABCD	ABCD	ABCD	ABCD

圖 3.26 項目集 ABCDE 之儲存結構

由圖 3.25 中，加總各樹上根節點的計數值，即可得到項目集 ABCDE 在資料庫中的容錯支持度 $\sup_a(\text{ABCDE}) = 6 \geq \min\_\sup_c$，因此可判斷滿足近似常見項目集條件一。

再檢查項目集 ABCDE 中每個項目之項目支持度，經計算後 $\sup_{\text{item}}^{\text{ABCDE}}(\text{A}) = 4$、$\sup_{\text{item}}^{\text{ABCDE}}(\text{B}) = 4$、$\sup_{\text{item}}^{\text{ABCDE}}(\text{C}) = 6$、$\sup_{\text{item}}^{\text{ABCDE}}(\text{D}) = 5$ 及 $\sup_{\text{item}}^{\text{ABCDE}}(\text{E}) = 5$，皆大於 $\lceil \sup_a(\text{ABCDE}) * (1 - \varepsilon_c) \rceil$(也就是 $\lceil 6 * (1 - 0.5) \rceil = 3$)，因此項目集 ABCDE 滿足近似常見項目集條件二，可判斷出 ABCDE 為一個近似常見項目集。

執行完上述步驟後，將 P′ 設為 ABCDE，回到步驟<3>繼續遞迴執行。但由圖 3.26 可觀察出已不存在其他項目可與 ABCDE 組成長度 6 的候選項目集，因此回到遞迴呼叫檢查 $\text{P} = \{\text{A}, \text{B}, \text{C}, \text{D}, \text{E}\}$ 的上一層即項目集 ABCD，找出是否有其他項目可加入成為新的候選項目集，並繼續以深先搜尋的方法遞迴執行探勘近似常見項目集。

在探勘近似常見項目集的過程中，除非在挑選項目組合出候選項目集時，該項目集連核心樣式的限制條件 $min\_sup_c * \alpha$ 都無法滿足，否則儘管該項目集之容錯支持度沒有超過 min_sup_c，仍可被保留下來，因為組成長度更長的候選項目集後，因最大比對錯誤值的改變反而有可能成為近似常見項目集。

第四章 實驗結果

在此以實做程式的方式來評估本論文提出的FP-AFI演算法之執行效率，並與其他探勘近似常見項目集的FT-Apriori [6]與AFI [4]演算法之執行效率進行比較。用以實做的程式語言為Microsoft Visual C++ 6.0，實驗環境的作業系統為Microsoft Windows XP Professional，系統配備採用 Pentium 4 3.4GHz 的中央處理器，並搭載 2.0GB 的記憶體。

4-1 模擬資料集

4-1.1 模擬交易資料集產生方式

在實驗中使用的模擬資料是用 IBM Data Generator[14]模擬產生的交易資料，其中在描述交易資料集內資料特性的參數意義如表 4.1 所示。

表 4.1 實驗資料集參數說明

參數	參數說明
|T|	每筆交易資料的平均項目個數
|I|	項目的種類數量
|D|	交易資料的筆數

在以下實驗說明中將以TmInDs來表示執行程式時所採用的資料集之參數，其所代表的意義為|T|=m、|I|=n 及|D|=sK。

4-1.2 實驗評估方式

在進行近似常見項目集探勘時，有許多因素會影響到探勘的結果與執行時間，以下實驗將針對最小支持度門檻值、核心樣式參數值α、誤差容忍參數值ε_r、ε_c及交易資料筆數與項目種類的數量等各個不同的因素，透過控制參數的方式，觀察參數值的變化對不同演算法執行時間的影響。

原本 AFI 及 FT-Apriori 演算法中在探勘近似常見項目集時並沒有考慮核心樣式的限制條件，為使兩個演算法與本論文所提出之FP-AFI演算法能進行公平的效能比較，本實驗在實做 AFI 及 FT-Apriori 演算法的過程中，亦加入核心樣式的限制條件，使這三種探勘近似常見項目集演算法得到的結果能夠相符。

【實驗一】改變最小支持度門檻值

使用資料集 T10I50D20K，探討改變最小支持度門檻值時各演算法在執行時間上的變化。固定各執行參數值 $\alpha = 0.8$、$\varepsilon_r = 0.2$ 及 $\varepsilon_c = 0.5$，改變 min_sup 值由1%開始逐次增加1%，執行結果如圖 4.1 所示。

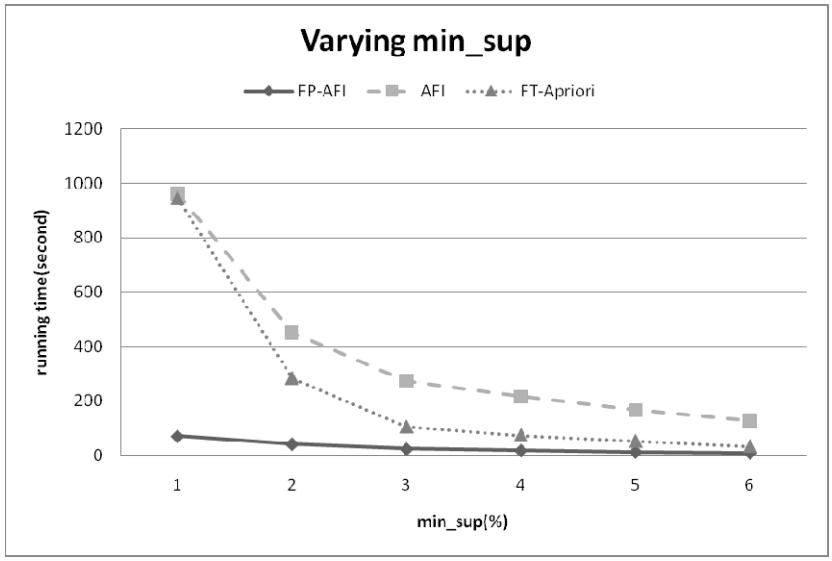


圖 4.1 改變最小支持度門檻值的執行時間比較

將 min_sup 訂的越小，則探勘過程需檢查之候選項目集數量越多，因此實驗結果顯示三種演算法在 min_sup 設定較小時，需耗費較多的執行時間。但 FP-AFI 執行時間的成長趨勢較另兩個演算法來得緩和。在最小支持度門檻值為1% 時，相較於 AFI 與 FT-Apriori 演算法所需的執行時間，本論文提出的 FP-AFI 演算法，在執行時間約只需另兩者執行時間的十三分之一，執行效率明顯優於其他兩種方法。此實驗結果顯示在 min_sup 訂為較小時，以 Apriori 演算法為基礎的另兩種演算法需大量列舉候選項目集，因此特別能凸顯運用 FP-tree 結構進行近似常見項目集探勘的執行效率。

【實驗二】改變核心樣式門檻值α

使用資料集 T10I50D20K，探討當核心樣式門檻值α改變時對執行時間造成的影響。核心樣式門檻值α由0.3開始逐次增加0.1，其餘參數皆設為固定值：$\min\_sup = 2\%$、$\varepsilon_r = 0.2$及$\varepsilon_c = 0.5$，執行結果如圖 4.2 所示。

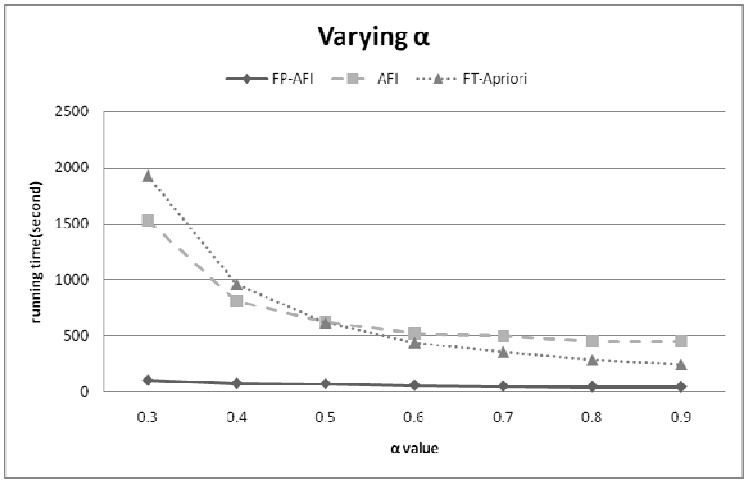


圖 4.2 改變核心樣式門檻值α的執行時間比較

改變核心樣式門檻值α最主要的影響在於挑選項目組成候選項目集的過程，若將α值訂的越低，則探勘過程滿足核心樣式門檻值的候選項目集數量越多，增加執行所需時間，但相對於另兩種方法，α值降低對FP-AFI執行時間顯得沒有顯著影響。具有 Apriori 特性的 AFI 及 FT-Apriori 演算法，隨候選項目集的長度逐漸增加，在每回合列舉出候選項目集後，會檢查核心樣式門檻值條件，找出符合核心樣式限制條件的候選項目集。當α值愈大，會符合此限制條件的項目集數量也愈少。當α值到達0.5以上時，已經過濾掉大部分支持度小於核心樣式門檻值的候選項目集，結合 Apriori property 的砍除策略，大量減少組合產生的候選項目集，因此 FT-Apriori 演算法的執行效率反而變成優於 AFI 演算法。

【實驗三】改變誤差容忍參數ε_r

以資料集 T10I50D20K 進行實驗，探討改變誤差容忍參數ε_r時各演算法在執行時間上的差異。其中固定各參數為：$\min\_sup = 2\%$、$\alpha = 0.8$及$\varepsilon_c = 0.5$。ε_r的數值則由 0 開始逐次增加 0.1，執行結果如圖 4.3 所示。

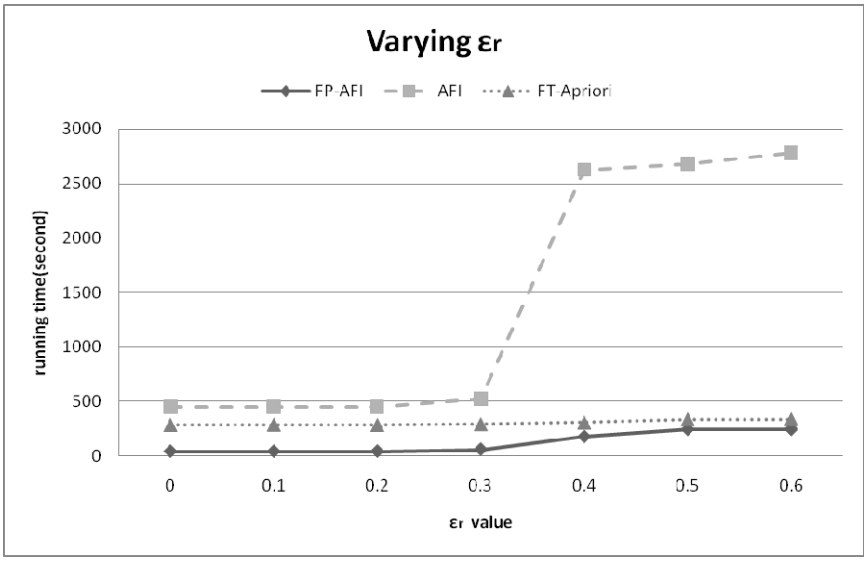


圖 4.3 改變誤差容忍參數ε_r的執行時間比較

改變誤差容忍參數ε_r的意義在於當ε_r值愈大，代表比對項目集出現的容錯條件放寬，比對項目集時，容錯項目的個數也會受到ε_r值的影響。當ε_r值超過0.3後，AFI與FP-AFI演算法的執行時間都有上升的趨勢，FP-AFI演算法採取的策略是當容錯項目的個數發生改變時，會回到先前的項目集之儲存結構中建立比對錯誤值增加後對應的 conditional FP-tree，隨著ε_r值變大容錯項目個數發生改變的頻率也變高，造成FP-AFI執行時間會微幅增加，但與AFI演算法的執行時間相較之下，成長趨勢是較緩和的。FT-Apriori 的執行時間變化呈現近於水平(當ε_r增加到 0.4 時，執行時間有微幅的上升只是不容易從圖4.3中觀察出來)，主要的原因

在於不管ε_r值變為多少，每次產生候選項目集後，針對每個候選項目集都必須掃瞄交易資料集一次以計算容錯支持度，ε_r值的改變僅影響到比對時的容錯放寬程度，因此在執行時間上沒有太大的變化，呈現近似於水平的曲線。

【實驗四】改變誤差容忍參數ε_c

使用 T10I50D20K 的資料集，探究當誤差容忍參數ε_c發生改變時對執行時間造成的影響。調整誤差容忍參數ε_c，由 0 開始逐次增加 0.1，固定其他參數值：$\min\_sup = 2\%$、$\alpha = 0.8$及$\varepsilon_r = 0.2$，執行結果如圖 4.4。

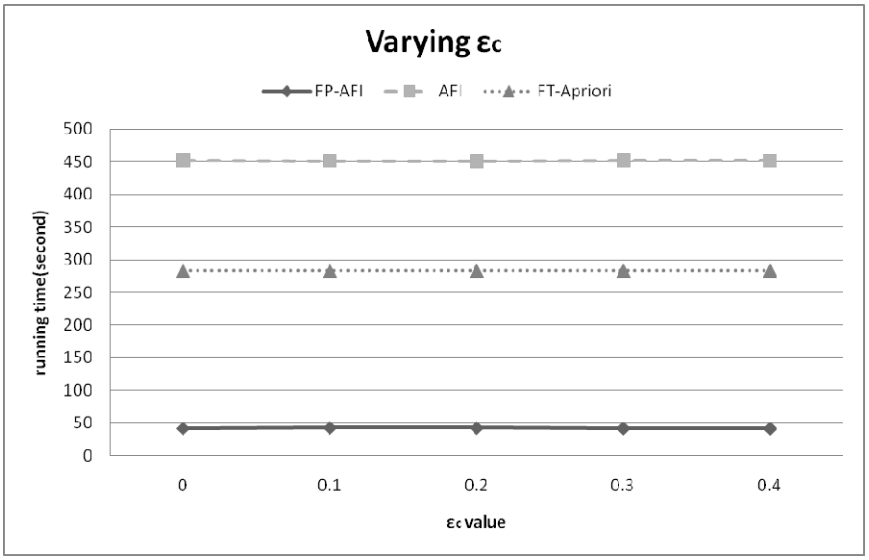


圖 4.4 改變誤差容忍參數ε_c的執行時間比較

改變誤差容忍參數ε_c對三種演算法最主要的影響在於檢驗容錯包含某候選項目集的交易資料集中，該候選項目集內每一個項目的支持度是否都達到一定比例之上。從圖 4.4 的結果看來ε_c值的改變並不會影響各演算法所需的執行時間，隨著ε_c值的改變，三種探勘近似常見項目集演算法的執行時間都趨於固定。

【實驗五】改變資料集的交易資料筆數

在實驗五中探討當交易資料的筆數發生變化時對執行時間造成的影響。使用的交易資料筆數分別為 1000 筆、5000 筆、10000 筆、15000 筆、20000 筆及 25000 筆，在這些交易資料中固定項目種類的數量為 50 種、每筆交易的平均項目個數為 10 個。其他參數固定為：$\min\_sup = 2\%$、$\varepsilon_r = 0.2$、$\varepsilon_c = 0.5$ 及 $\alpha = 0.8$，執行結果如圖 4.5 所示。

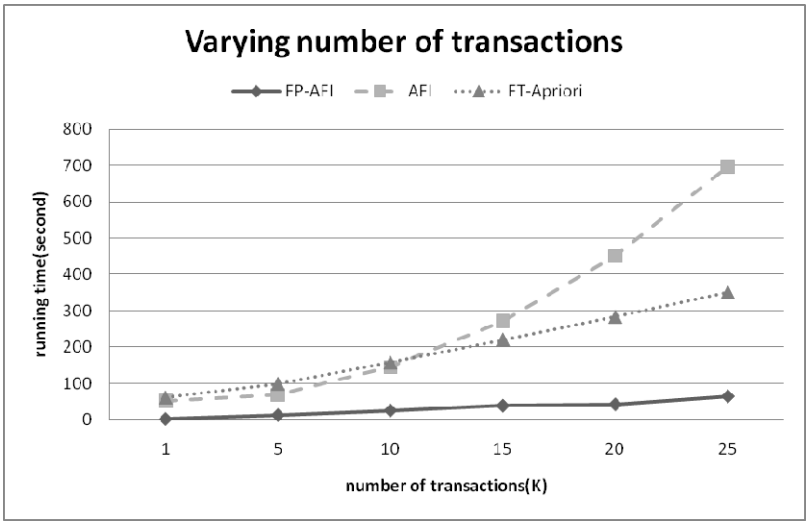


圖 4.5 改變交易資料筆數的執行時間比較

當交易資料筆數較少時，三種演算法在執行時間上沒有太大的差距，但隨交易資料筆數的逐漸增加，三種演算法的執行時間都有成長的趨勢，其中具有 Apriori 特性的 AFI 與 FT-Apriori 演算法之執行時間增加的幅度較 FP-AFI 明顯。每當 FT-Apriori 演算法在產生候選項目集後，針對每一個候選項目集必須掃瞄所有交易資料集一次，找出資料集中容錯包含該項目集的容錯支持度，因此交易資料的筆數越多時，就必須花費較多的時間來進行比對。當交易資料筆數高於一萬筆

後，AFI 演算法的執行時間逐漸高於 FT-Apriori 演算法，AFI 雖然不用在每次產生候選項目集後就掃瞄交易資料集一次以計算該項目集的容錯支持度，而是在執行之初就已經對每個項目記錄一個包含該項目的交易編號所成的集合，隨交易筆數增加，集合中必須記錄的交易編號也會隨之增加，當以此集合進行運算找出容錯支持度時，就必須耗費較多的計算成本，造成執行時間多於 FT-Apriori 演算法的情況。

而 FP-AFI 演算法利用 FP-tree 的結構，將交易資料壓縮成樹狀結構，在執行的過程中只須掃瞄所有的交易資料兩次，不需反覆掃瞄比對資料內容，因此在交易資料的數量越多時，越可凸顯本論文提出的 FP-AFI 演算法之執行效率。

【實驗六】改變資料集中項目種類的數量

在實驗六中探討當改變交易資料內項目種類的數量時對執行時間產生的影響。在實驗中改變項目種類的數量依序為 10 種、50 種、100 種、150 種及 200 種，資料集內平均的項目個數為10個、交易資料的筆數為 20K。其餘會使用到的參數也都予以固定，包括 $\min\_sup = 2\%$ 、$\varepsilon_r = 0.2$ 、$\varepsilon_c = 0.5$ 及 $\alpha = 0.8$ ，執行結果如圖 4.6 所示。

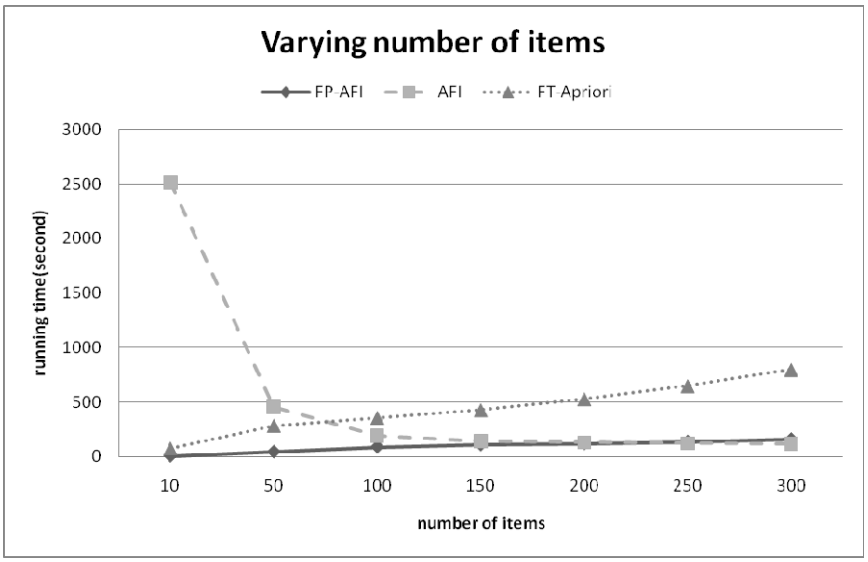


圖 4.6 改變資料集中項目種類的數量之執行時間比較

根據圖 4.6 的結果，FP-AFI 演算法隨資料集內項目種類的增加，其執行時間有微幅上升的趨勢。分析主要的原因是利用 FP-tree 結構進行探勘時，當項目種類較多，造成建構出來的 FP-tree 尺寸變得比較龐大。以致於透過 FP-tree 進行近似常見項目集的探勘時，受到樹架構大小的影響，而些微增加執行時間。但資料集內項目種類之數量增加也會造成各項目在交易資料集內的分佈較稀疏，因此在 FP-AFI 演算法的執行流程中，會先掃瞄交易資料集一次統計出每個項目的支持

度，可先過濾某些支持度不滿足核心樣式門檻值的項目，當第二次掃描建立FP-tree 時便不會將這些支持度過低的項目加入FP-tree 中，減低 FP-AFI 演算法受項目種類數量變多而增加執行時間的影響。

經實驗結果發現，當項目種類的數量為 200 時，利用核心樣式限制條件過濾後，滿足核心樣式門檻值的項目仍有107 個，因此利用 FT-Apriori 演算法在組合候選項目集的過程需要耗費較多的計算成本，並一一掃瞄交易資料集得到容錯支持度，增加許多執行的時間。而同為具有 Apriori 特性的 AFI 演算法，因為項目種類的數量變多，每一個項目記錄包含該項目的交易資料編號之集合較小，因此在計算候選項目集的容錯支持度時，可以展現較好的執行效率，使得執行時間在項目種類的數量超過 50 種後便逐漸低於 FT-Apriori 演算法。整體而言，雖然 FP-AFI 演算法會受到項目種類增加的關係而造成執行時間些微增加，但增加的幅度相較於 FT-Apriori 演算法是呈現較平緩的趨勢。

4-2 實際資料集

4-2.1 資料內容

實驗中所使用的實際資料是一份描述不同類型動物之間的特徵資訊，下載自UCI machine learning repository[8]。資料集中包含101種動物的特徵資料，每一種動物都以18個不同的屬性做描述，第一個屬性代表動物名稱，而其中15個屬性利用布林代數做表示，以0或1表示動物是否具有某個屬性代表的特徵。這15個布林代數的屬性分別為：hair、feathers、eggs、milk、airborne、aquatic、predator、toothed、backbone、breathes、venomous、fins、tail、domestic、catsize。以及另外兩個數值化的屬性為：腳的數量與對應的動物類別編碼，資料集中包含的101種動物涵蓋哺乳類、鳥類、魚類、昆蟲等七種類別，最後一個屬性表示的數值代號就是動物所屬的類別編碼。

在論文[4]中提到使用這種動物資料集找近似常見項目集的主要目的在於若將動物的各種特徵當成各自獨立的項目，在組合成項目集後，可用項目集內包含的動物特徵做為標示同種類動物的代表。在同一種類型的動物中，舉例來說哺乳類動物一般都認定可進行哺乳、有毛髮覆蓋或具有牙齒的咀嚼功能等特徵，但某些屬於哺乳類的動物例如海豚就沒有毛髮覆蓋，而鴨嘴獸沒有牙齒進行咀嚼動作，並非同一種類型的動物都具有相同的特徵。在探勘代表某類型動物具有共同特徵所組成的常見項目集時，若不允許容錯部分特徵，則找出來的常見項目集可能只包含少數特徵，不夠具有代表性，無法完全涵蓋同一種類型的動物或是其他

類型的動物可能也具有少數相同的特徵，而被錯誤的包含進來。因此考慮探勘近似常見項目集，允許容錯包含部分動物特徵，將會得到更具有代表性的探勘結果。

4-2.2 實驗評估

在實驗中只有採用15個以布林代數表示的特徵進行探勘，其中特別針對哺乳類、鳥類及魚類動物這三種動物數量較多的類別進行評估，但觀察鳥類與魚類動物間具有的特徵較為相似，為明顯看出近似常見項目集的探勘結果，在本實驗中僅對哺乳類動物進行分析。找出代表哺乳類動物的近似常見項目集後，觀察容錯包含該項目集的動物資料是否對應到正確的類別或是出現分錯類別的情況。根據動物資料集內標示類別欄位的統計，在101種動物中屬於哺乳類的有41種，以FP-AFI演算法進行探勘得到的結果如下：

哺乳類：設定所使用的參數中，$\min\_\sup = 0.4$、$\alpha = 0.5$、$\varepsilon_r = 0.2$、$\varepsilon_c = 0.2$。探勘得到的近似常見項目集為{ hair, milk, toothed, backbone, breathes, tail }，此近似常見項目集於動物資料集內支持度為41，即資料集內101筆動物資料中有41筆容錯包含此項目集，且這41筆資料代表的動物皆為哺乳類動物。以此項目集在動物資料集中進行比對，沒有其他類型的動物會被歸於哺乳類。

表 4.2　FP-AFI 與 AFI 演算法探勘哺乳類動物之結果比較

	FP-AFI 演算法	AFI 演算法
探勘結果項目集長度	6	7
項目集包含動物種類數量	41	40
召回率	$\frac{41}{41}=100\%$	$\frac{40}{41}=97.56\%$
正確率	$\frac{41}{41}=100\%$	$\frac{40}{40}=100\%$

表 4.2 比較 FP-AFI 與 AFI 演算法在探勘哺乳類動物的近似常見項目集時，取用的特徵項目數量，並觀察此動物資料集包含的 41 種哺乳類動物中，有多少種哺乳類動物可以被探勘結果的近似常見項目集涵蓋進來，即可計算探勘結果的召回率(recall)。而根據探勘結果計算容錯包含該項目集的動物資料中哺乳類動物所佔的比率即可得到探勘結果的正確率(precision)。實驗結果顯示，相較於 AFI 演算法必須挑選 7 種動物特徵，才能涵蓋 40 種哺乳類動物，而 FP-AFI 演算法只須挑選 6 種動物的特徵，就可以包含所有的哺乳類動物。以分類的觀點來看，FP-AFI 演算法能透過挑選較少量的動物特徵，形成較簡易的分類規即可將所有屬於哺乳類的動物包含進來。

透過以上實驗結果顯示，FP-AFI 演算法應用在實際資料中確實能夠有效的探勘出近似常見項目集。藉由哺乳類動物所得到的項目集結果，檢驗項目集在動物資料集中所獲得的支持度，皆沒有其他種類的動物會被錯誤包含進來。

4-3 實驗結果總結

根據以上的實驗結果顯示本論文提出的FP-AFI演算法，在min_sup或α值越小時，候選項目集產生的數量將直接受到影響，透過FP-tree的儲存結構確實能大幅降低探勘近似常見項目集所需的時間。若改變交易資料的特性，當交易資料的筆數逐漸增加時，AFI 與 FT-Apriori 演算法會隨著資料量的增加造成執行時間明顯遽增，而FP-AFI演算法利用 FP-tree 結構具有壓縮交易資料集的特性，比起另兩種演算法較不易受交易資料筆數增加的影響，因此執行時間僅微幅的上升。

應用到實際的動物資料上，透過挑選動物特徵形成的集合，可正確的探勘出具有代表性的近似常見項目集，掌握不同類型的動物之關鍵特徵，容忍某部分的特徵可以不被包含，凸顯同一種類的動物並不會具有完全相同特徵的性質，得到比純粹探勘常見項目集更具一般化的結果。

第五章 結論與未來研究方向

本論文提出以 FP-tree 為儲存結構，擴展成可從中探勘出近似常見項目集的 FP-AFI 演算法。論文中提出對 FP-tree 依包含某項目及不包含某項目的投影方法，並分析容錯包含項目集之交易資料集合間的遞迴關係，因此 FP-AFI 演算法可以系統性方式建構出對應的 conditional FP-tree ，並依所建構之 conditional FP-tree 根節點上儲存的計數值資訊，獲得項目集在資料庫中的容錯支持度。並運用 conditional FP-tree 的編碼，可快速計算出候選項目集內各個項目在容錯包含該項目集之交易資料中的項目支持度。根據實驗結果顯示，FP-AFI 演算法在最小支持度門檻值設為較小或交易資料筆數較多時，執行效率皆顯著優於 AFI 及 FT-Apriori 演算法。

FP-AFI 演算法在探勘近似常見項目集時，隨容錯項目個數的改變會在儲存結構中存放多棵的 FP-tree 與 Header Table。隨容錯項目個數變多或資料集中項目種類的數量增加，儲存空間需求亦漸增，因此未來可深入探討如何在資料儲存的結構上更節省空間。此外，可進一步探討將此方法應用在資料流環境的可行性，透過探勘資料流中近似常見項目集，觀察資料變動的走向或改變趨勢。

參考文獻

[1] R. Agrawal and R. Srikant, “Fast Algorithm for Mining Association Rule in Large Databases,” in Proc. of the 20th International Conference on Very Large Data Bases, 1994.

[2] J. Han, J. Pei and Y. Yin, “Mining Frequent Patterns without Candidate Generation, ” in Proc. of the 2000 ACM SIGMOD International Conference on Management of Data, 2000.

[3] H. Cheng, P. S. Yu and J. Han, “AC-Close: Efficiently Mining Approximate Closed Itemsets by Core Pattern Recovery,” in Proc. of the 6th IEEE International Conference on Data Mining (ICDM 2006), 2006.

[4] J. Liu, S. Paulsen, X. Sun, W. Wang, A. Nobel and J. Prins, “Mining approximate frequent itemsets in the presence of noise: Algorithm and analysis,” in Proc. of the 6th SIAM International Conference on Data Mining (SDM 2006), 2006.

[5] H. Cheng, P. S. Yu and J. Han, “Approximate Frequent Itemset Mining In the Presence of Random Noise,” Soft Computing for Knowledge Discovery and Data Mining. Oded Maimon and Lior Rokach. Springer, pages 363-389, 2008.

[6] J. Pei, A. K. H. Tung and J. Han, “Fault-Tolerant Frequent Pattern Mining: Problems and Challenges,” in Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD 2001), 2001.

[7] R. Gupta, G. Fang, B. Field, M. Steinbach and V. Kumar, “Quantitative evaluation of approximate frequent pattern mining algorithms,” in Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), 2008.

[8] UCI machine learning repository. (http://www.ics.uci.edu/mlearn/MLSummary.html)

[9] K. Wang, L. Tang, J. Han and J. Liu, “Top Down FP-Growth for Association Rule Mining,” in Proc. of Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference (PAKDD 2002), 2002.

[10] J. L. Koh and P. W. Yo, “An Efficient Approach for Mining Fault-Tolerant Frequent Patterns Based on Bit Vector Representations,” in Proc. of the 10th International Conference, Database Systems for Advanced Applications, (DASFAA 2005), 2005.

[11] J. K. Seppänen and H. Mannila, “Dense itemsets,” in Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004), 2004.

[12] C. Yang, U. M. Fayyad and P. S. Bradley, “Efficient discovery of error-tolerant frequent itemsets in high dimensions,” in Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001), 2001.

[13] M. S. Chen, J. Han and P. S. Yu, “Data Mining: An Overview from a Database Perspective,” in Proc. of the IEEE Transactions on Knowledge and Data Engineering, Volume 8(6): pages 866-883, 1996.

[14] IBM inc. http://www.almaden.ibm.com/cs/quest/syndata.html.