從大型資料庫中挖掘加權的關聯規則
Mining Weighted Association Rules from Large Databases

顏秀珍　　邱鼎穎　　李御璽
輔仁大學資訊工程學系　　輔仁大學資訊工程學系　　銘傳大學資訊管理學系
sjyen@csie.fju.edu.tw　　leeys@mcu.edu.tw

摘要

挖掘*關聯規則*(Mining Association Rule)主要是從交易資料庫中找尋物品間的關聯性，而*加權的關聯規則*比一般的關聯規則多考慮了物品在交易中的重要性，因此能比一般的關聯規則找出更多有利的資訊。但過去有關加權關聯規則的研究相當少且執行效能很差，因此我們提出一個有效率的演算法來解決這方面的問題，此演算法不僅可以加快執行的速度，亦可節省相當多的記憶體空間。我們從理論上分析我們的演算法與過去的演算法在執行速度與所需記憶體空間的差異，並以模擬的資料庫進行實驗，證明我們的方法在執行速度上較過去的方法快很多。

關鍵詞：資料探勘，加權的關聯規則，交易資料庫，頻繁項目集

1. 概論

在以往的時候資料的取得是相當困難的。因此若有新的資料，大多會馬上被納入資料庫中妥善的保存。但隨著電腦與網路的日益發達，資料的取得不再是件難事，比較嚴重的問題反而是在於有過多的資料而無法判別哪些是重要的。因此*資料探勘技術*(data mining technique)[3, 13, 15, 16]在資料庫相當龐大的今日漸漸的興起。透過資料探勘技術，使用者可以在大量的資料中找出難以發現的重要資訊。在資料探勘中有許多不同的方法與技術，可以求得不同的資訊與知識。而*關聯規則*(association rule) [1, 2, 5, 6, 7, 8, 10, 11, 12, 14]的找尋是資料探勘中的一個重要項目。

找尋關聯規則必須先找出常常一起被購買的物品，因此，關聯規則的產生取決於物品被購買次數的多寡，這是相當不公平的。舉例來說：對於一個同時販賣電燈泡和電冰箱的家電廠商來說，只計算被購買次數的話，那麼和電燈泡有關的規則會列出一大堆，而和電冰箱有關的資訊可能沒有，但是對於廠商來說，電冰箱的利潤較高，其營業額反而才是他所關心的，但卻無法給予相關的訊息。因此這篇論文定義了*加權的關聯規則*，並提出一個非常有效率的演算法求取加權的關聯規則，以解決上述的問題。

在對問題做說明之前，我們先做一些簡單的定義。首先，一筆交易(transaction)是指一個記錄，記載某一顧客某次所購買的商品項目。而交易資料庫(transaction database)則是由許多交易所組成，因此交易資料庫中所記載的資料包含了交易編號、顧客編號、購買時間及購買項目等。另外，一個項目集(itemset)是由項目(item)所構成的集合，所以交易也算是一個項目集。而項目集的長度(length)是指此項目集所包含的項目個數，若一個項目集長度為k，則我們稱此項目集為k-項目集。假設有A、B兩項目集，且B項目集中所有的項目皆出現在A中，則稱B被A所包含，且B為A的子項目集(sub-itemset)。若一項目集包含於某一筆交易，則稱此交易支持此項目集。在交易資料庫中，所有支持該項目集的交易筆數即為此項目集的*計數*(count)，而所有支持該項目集的交易筆數與資料庫中所有交易筆數的比值即為此項目集的*支持度*(support)。

在找尋關聯規則時，使用者必須給定兩個重要的值，一個是*最小支持度*(minimum support)，另一個是*最小信任度*(minimum confidence)，當某一個項目集的支持度大於或等於使用者所給定的最小支持度時，我們稱該項目集為*頻繁項目集*(frequent itemset)，若一個頻繁項目集長度為 k，則我們稱此項目集為 k-頻繁項目集，所有可能成為頻繁項目集的項目集，我們稱其為候選項目集(candidate itemset)。另外，為了以後討論方便，我們定義*最小計數*(minimum count)為最小支持度乘上資料庫中的總交易筆數後無條件進位的值，當某一個項目集的計數大於或等於最小計數時，該項目集為頻繁項目集。一個*規則*(rule) $X => Y$ 的*信任度*為共同支持 X、Y 項目集的交易筆數與所有支持 X 的交易筆數之比值。用數學式子表示如下：

$$\text{conf}(X => Y) = \frac{Sup(X \cup Y)}{Sup(X)} \qquad (1)$$

其中，*Sup(X)*表示項目集 X 的支持度。若一個規則的信任度大於或等於最小信任度時，我們稱其為關聯規則。

加權關聯規則和一般關聯規則的差異主要在於對項目集的支持度之計算方式。在加權關聯規則方面，每一個項目都會根據該項目在應用上的意義給予其權值，如：要求出佔總利潤 20%以上的加權關聯規則，則每個物品的權值即為該物品的利潤。所以計算某一項目集的支持度，必須考慮此項目集中每一項目的權值[4]，我們舉個例子說明：假設一個項目集 $A=\{a_1,a_2,\ldots,a_n\}$，其中每一項目的權值分別為 $\{w_1,w_2,\ldots,w_n\}$，則此項目集的*加權支持度*用數學式子表示如下：

$$\text{Wsup(A)} = \sum_{i=1}^{n} w_i \times Sup(A) \qquad (2)$$

對於加權的關聯規則方面，一般必須給定*加權最小支持度*(weighted minimum support，wms)與最小信任度。若項目集 A 的加權支持度 Wsup(A)大於或等於加權的最小支持度，則稱 A 為加權的頻繁項目集(weighted frequent itemset)，同樣地，利用式(1)，可以求得加權規則的信任度，若此信任度大於或等於最小信任度，則此規則為加權的關聯規則。在本論文中，我們提出一個求取加權關聯規則的新觀念---加權延遲交易拆解(Weighted Delay Transaction Disassemble，WDTD)，利用此觀念，我們可以大量減少一次所需處理的候選項目集，並快速地找出所有候選項目集的計數，我們所提出的 WDTD 演算法可以找出所有的加權關聯規則，此方法將比過去的方法速度快且更加的節省記憶體空間。

在 1998 年，C.H. Cai 等人[4]提出了求取加權關聯規則的演算法 MINWAL，此演算法是沿用 Apriori 演算法的架構，但因為頻繁項目集的子項目集也必為頻繁項目集的觀念已不適用，故其 k+1-候選項目集是由 k-候選項目集所產生，因此其所產生的候選項目集相當多，且容易產生長度很長的候選項目集，而當一個長度很長的候選項目集被產生時，每一筆交易所會被拆解的數量將會大幅增加。假設最長的候選項目集為 *m*-候選項目集，則當第 *m* 次掃描資料庫要求取 *m*-頻繁項目集時，該交易要被拆解 c_m^n 次，所以一筆交易共要被拆解 $c_1^n + c_2^n + c_3^n + \ldots\ldots + c_m^n = \sum_{i=1}^{m} c_i^n$ 次，其最差執行時間為 $O(n^m)$。所以若是產生長度較長的候選項目集，則執行速度的緩慢可想而知。另外，由於 k+1-候選項目集是由 k-候選項目集所產生，因此候選項目集的數量相當大，而龐大的 k+1-候選項目集所產生的 k+2-候選項目集的數目更加大。龐大的候選項目集會造成計數時的時間大增，這是因為所要搜尋的範圍變大所致。由於 MINWAL 的演算法有上述的缺點，因此我們希望利用延遲交易拆解的概念來快速求出加權的關聯規則。

2. 挖掘加權的關聯規則

在應用層面上，假設一個項目集 $A=\{a_1,a_2,\ldots,a_n\}$，其中每一項目的權值分別為 $\{w_1,w_2,\ldots,w_n\}$，若 n 很大但 w_1~w_n 皆很小，此時用式(2)所求出的支持度會有很大的差異：由於 n 很大，因此項目集 A 的權值 $w_1+w_2+\ldots+w_n$ 也會很大，因此較易形成加權的頻繁項目集。但若將式(2)所求得的加權支持度除以該項目集的項目個數，則會使得平均權值非常低，因此不易形成加權的頻繁項目集。針對這項差異，我們認為當一項目集，雖然其各個項目的權值非常低，但若該項目集的總權值高且被購買的次數也多，使得加權支持度超過我們所給定的加權最小支持度，則該項目集仍有被挖掘出來的需要。

另外，假設總利潤(或總權值)為 500 萬元，若要求出佔總利潤 20%以上的項目集，亦即加權最小支持度為 20%，其意義為若項目集 A 在所有交易中的總利潤超過 100 萬元 (500*0.2=100) ，則其即為所求。但因為式(2)中 Sup(A)所代表的意義為項目集 A 的支持度，亦即為項目集 A 的計數除以資料庫中的總交易筆數，所以式(2)所求得的值為項目集 A 的權值乘上項目集 A 的計數再除以資料庫中的總交易筆數，但除以總交易筆數在應用面上毫無任何意義，因此我們重新定義項目集 A 的加權支持度 Wsup(A)如式(3)所示，而若其要成為加權的頻繁項目集的條件如下：

$$\text{Wsup(A)} = \sum_{i=1}^{n} w_i \times Count(A) \geq \text{mw} \qquad (3)$$

其中 *Count(A)*表示項目集 A 的計數，而 mw 表示資料庫中的總權值乘上加權最小支持度，我們稱為*最小權值*(minimum weight)。而利用式(3)我們可以得到項目集 A 要成為加權頻繁項目集的最小計數如式(4)所示：

$$\text{SC(A)} \geqq \frac{\text{mw}}{\sum_{i_j \in A} w_j} \qquad (4)$$

我們的演算法將根據式(3)與式(4)來運作。而對於加權頻繁項目集，以下我們仍簡稱頻繁項目集。

在概念上我們的演算法需要一個已經排序好的資料庫，如表一所示。在介紹演算法前，我們先介紹資料庫排序的方法：1.首先對於每一筆交易資料的項目，按照其編號由小到大排序。2.對於整體資料庫進行排序：對於兩筆交易，比較第一個交易項目，將項目編號較小的交易排前面，若相等則比較下一個交易項目，一直到分出大小為止。在實作上，我們可以利用一些資料結構，快速完成排序的動作，在後面的章節中，我們會做詳細的說明。

表一：已排序資料庫

交易編號	交易內容
1	{1,2,3,4,5,6,7}
2	{1,3,4,5,6,7}
3	{1,3,5,6,7}
4	{2,3,4,6,7}

另外，我們將表一中每個項目的權值列成表二，如果要求取的關聯規則是與利潤有關的話，則可以將各個權值視為各個項目的利潤。

表二：項目權值表

項目編號	權值(利潤)
1	40
2	30
3	30
4	20
5	20
6	10
7	10

另外，我們稱一筆交易或項目集中的第一個項目為交易第一項目**(the first item of transaction, FIT)**；一筆交易或項目集中的第二個項目為交易第二項目**(the second item of**

transaction, SIT)；一筆交易或項目集中的第三個項目為交易第三項目**(the third item of transaction, TIT)**。例如，表一中第一筆交易的 FIT 為 1；第四筆交易的 FIT 為 2；第一筆交易的 SIT 為 2；第四筆交易的 SIT 為 3；第一筆交易的 TIT 為 3;第四筆交易的 TIT 為 4。

表三：WDTD 的執行步驟

掃描次數	執行的動作
第一次	求出所有的 1-頻繁項目集與 2-頻繁項目集
第二次	對於每一筆交易，執行下列步驟 1. 若有後選項目集存在時，計算候選項目集的計數 2. 產生新的候選項目集 3. 延遲交易的拆解 4. 尋找下一個檢核交易

在實作上 WDTD 為一個需要執行兩次資料庫掃描的演算法，每次掃描資料庫所要執行的動作表三所示。

2.1 產生 k-頻繁項目集(k=1,2)

規則一：求取 1-頻繁項目集與 2-頻繁項目集
For k = 1 to n
(1) 將資料庫中 FIT=k 的交易掃描進來，並執行(2)(3)(4)步驟
(2) 對於每筆交易中的每一項目，將 1-項目集陣列中此項目所對應的計數加一
(3) 對於每筆交易中，去除 FIT 後的每一項目，將 2-項目集陣列中此項目所對應的計數加一
(4) 將每筆交易中，去除其FIT後的子交易，存放到如圖一的暫存區中
(5) 將資料庫中FIT=k的交易掃描完後，至暫存區中檢查，對於所有FIT=k的交易，執行(3)(4)步驟
(6) 前五步驟都完成後，計算2-項目集陣列中各項目集的加權支持度，並決定其是否為 2-頻繁項目集
(7) 做完前六步驟後，將2-項目集陣列所記錄的計數值全清為0
End For
(8) 計算1-項目集陣列中各項目集的加權支持度，並決定其是否為 2-頻繁項目集

由於在概念上 WDTD 使用了已排序的資料庫，如表一所示，所以可以利用一個很特別的方法在第一次掃描資料庫時，同時產生 1-頻繁項目集與 2-頻繁項目集，我們利用 *1-項目集陣列*及 *2-項目集陣列*分別記錄 1-項目集的計數及 2-項目集的計數，當要求取加權的頻繁項目集時，我們將每個項目集乘上該項目集的權值 (也就是計算加權支持度)，若其值大於或等於最小權值，則該項目集為頻繁項目集。

表四：1-項目集陣列

項目編號	計數
1	3
2	1
3	3
4	2
5	3
6	3
7	3

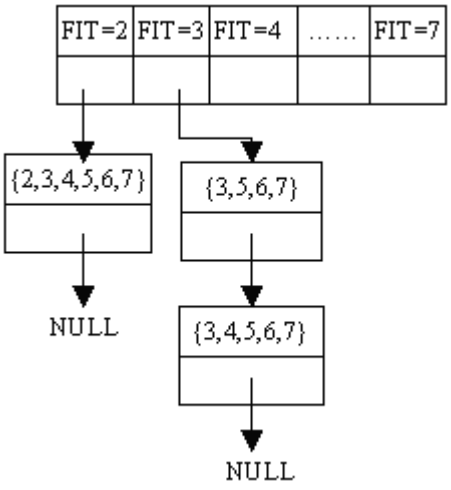
表五：2-項目集陣列

項目編號	計數
1	0
2	1
3	3
4	2
5	3
6	3
7	3

例如表四是掃描表一的前三筆交易後，各項目的計數，由於資料庫已排序，故一開始與項目{1}有關的交易，會先被掃描與處理，例如表一中的前三筆交易。所以在此次資料庫掃描的同時，我們也記錄與項目{1}有關的 2-項目集，例如表五所記錄的是包含項目{1}的 2-項目集之計數。我們利用規則一來產生 k-頻繁項目集(k=1,2)，假設項目總數為 n。

表四、表五及圖一為根據規則一掃描完資

料庫中前三筆交易的結果，此時 FIT=1 的交易全部掃描完畢，因此執行規則一的第六步驟。在表五中項目{3}的計數為 3 乘上權值和 70 得到 210，大於最小權值 200，因此存在 2-頻繁項目集{1,3}。接著我們將表五陣列的計數清為 0，再將 FIT=2 的交易處理完畢，可以發現不存在任何 FIT=2 的 2-頻繁項目集。



圖一：暫存區的結構

2.2 產生 k-頻繁項目集（k≥3）

由於 WDTD 是一個求取加權關聯規則的嶄新觀念，因此我們必須給定一些與此觀念有關的定義，我們利用表一的資料庫做例子來輔助說明，並假設加權最小支持度為 40%。

定義一：權值檢核交易

假設根據式(4)，我們算出第一筆交易中某一項目集 A 要成為頻繁項目集的最小計數為 n，則第一筆交易後的第 n-1 筆交易 B，我們稱 B 為 A 的*權值檢核交易*。

我們以表一為例，假設加權最小支持度為 40%，因為第一筆交易至第四筆交易中所有項目的權值和分別為 160，130，110，以及 100，因此整個資料庫的權值總和為 500，而加權最小支持度為 40%，所以最小權值為 500*40%＝200，根據式(4)，我們可以計算第一筆交易的某一子項目集{1,2,3,4}之最小計數如下：

$SC(\{1,2,3,4\}) \geq \frac{200}{40+30+30+20}=1.67$，因此此項目集若要成為頻繁項目集，則必須要出現在第二筆交易中，我們稱第二筆交易為第一筆交易中項目集{1,2,3,4}的*權值檢核交易*，因為第二筆交易的前四項項目集並非{1,2,3,4}，因此項目集{1,2,3,4}不為頻繁項目集。利用排序資料庫及上述的觀念，我們可以去掉一些不可能成為頻繁項目集的項目集。

引理一：

在一個已排序的資料庫中，若第一筆交易中的某一k-項目集A與其權值檢核交易的前k項項目相同，則 A 為一個 k-頻繁項目集。若 A 小於權值檢核交易的前 k 項項目所形成之項目集，則 A 不為一個 k-頻繁項目集。

理由：在一個已排序的資料庫中，若 A 與其權值檢核交易的前 k 項相同，則表示從第一筆交易到權值檢核交易間的交易都包含了 A，所以 A 必為頻繁項目集。若 A 小於權值檢核交易的前 k 項項目所形成之項目集，則表示權值檢核交易不包含 A，所以 A 必不滿足其所需的最小計數，因此 A 不為一個 k-頻繁項目集。

例如表一中第一筆交易的某項目集{1,2,3,4}與其權值檢核交易，也就是第二筆交易{1,3,4,5,6,7}，做比較，因為第二筆交易的第二個項目已大於項目集{1,2,3,4}的第二個項目，所以項目集{1,2,3,4}不滿足其所需的最小計數 2，故不為頻繁項目集。

引理二：

對於一筆交易{1,2,…,n}，其所有長度大於或等於三的子項目集可以分為四個部分：

(1)所有包含此交易的FIT、SIT與TIT的項目集，皆為第一部份的項目集

(2)所有包含此交易的FIT與SIT但不包含TIT的項目集，皆為第二部份項目集

(3)所有包含此交易的FIT但不包含SIT的項目集，皆為第三部份的項目集

(4)所有不包含此交易的FIT之項目集，皆為第四部份的項目集並滿足下列兩點：

1．任兩部份交集皆為空集合

2．(1)∪(2)∪(3)∪(4)=交易{1,2,…,n}之所有長度大於三的子集所成之集合

例如表一的第三筆交易{1,3,5,6,7}，根據引理二，其所有長度大於三的子集可以拆解成下列四部分：

(1){1,3,5},{1,3,5,6},{1,3,5,7},{1,3,5,6,7}

(2){1,3,6},{1,3,7},{1,3,6,7}

(3){1,5,6},{1,5,7},{1,6,7},{1,5,6,7}

(4){3,5,6},{3,5,7},{3,6,7},{5,6,7},{3,5,6,7}

定義二：最大集合

對於引理二中的四部份子集，在每一部份中，不被其他在此部份的子集所包含的集合，稱為該部分的*最大集合*。

例如，引理二的例子中，四個部分的集合有被劃底線者，皆為該部分的最大集合。

2.2.1 延遲交易拆解概念

由於 WDTD 演算法在第一次掃描交易資料庫時，即可求出所有的 1-頻繁項目集與 2-頻繁項目集，因此在第二次掃描資料庫時我們只需求取所有長度為 3 以上的頻繁項目集即可。而延遲交易拆解的目的主要是要減少不必要的交易拆解，我們利用引理二將一筆交易 A 拆解成四部份，然後利用引理一來判斷第一部份中的項目集是否不為頻繁項目集，若有可能成為頻繁項目集，則列為候選項目集。對於交易 A 的第二部份、第三部份及第四部份項目集，因目前我們無法利用引理一來判斷其是否為頻繁項目集，因此將延遲對這幾部份的拆解，我們稱此動作為*遞延*。由於延遲交易拆解是主要概念，因此在介紹如何求取長度為 3 以上的頻繁項目集前，我們先介紹如何延遲交易拆解。

規則二：延遲交易拆解之基本規則

根據引理二，我們可將資料庫中的一筆交易拆解成四個部分，並依循下列規則：

1. 將第一部份的所有項目集列為候選項目集。
2. 將第二、三、四部分的最大集合各與其權值檢核交易做比較，若
   (a) 兩者的 FIT 或 SIT 不相等，則第二部分的最大集合不遞延；若需遞延，則將第二部份最大集合中，除了其 FIT 與 SIT 外，小於其權值檢核交易之 TIT 的所有項目皆刪除後再遞延。
   (b) 若兩者的 FIT 不相等，則第三部分的最大集合不遞延。若需遞延，則將第三部份最大集合中，除了 FIT 外，小於權值檢核交易之 SIT 的所有項目皆刪除後再遞延。
   (c) 將第四部分最大集合中項目編號小於其權值檢核交易之 FIT 的所有項目刪掉後，再將剩下的部分遞延。

對於規則二之 2(a)，我們舉例如下：在表一的資料庫中，因為第一筆交易的第二部分最大集合{1,2,4,5,6,7}與其權值檢核交易，也就是第二筆交易，比較後，兩者的 SIT 不相等，因此第一筆交易中所有 FIT=1 且 SIT=2 的項目集(即第一筆交易的第二部份項目集)，皆不可能成為頻繁項目集，因此可以不必遞延第二部分最大集合。

對於規則二之 2(b)，我們舉例如下：在表一的資料庫，因為第一筆交易的第三部分最大集合{1,3,4,5,6,7}與其權值檢核交易，也就是第二筆交易，比較後，兩者的 FIT 相等，且第三部份最大集合的 SIT 不小於其權值檢核

交易的 SIT，所以此部份中可能有項目集會成為頻繁項目集，因此須遞延此部份的最大集合。

對於規則二之 2(c)，我們舉例如下：在表一的資料庫，因為第一筆交易的第四部分最大集合{2,3,4,5,6,7}與其權值檢核交易，也就是第二筆交易，比較後其項目編號皆大於第二筆交易的 FIT，因此此部份中可能有項目集會成為頻繁項目集，所以須遞延此部份的最大集合。

規則二可以實行的原因是因為我們發現在掃描資料庫時，只要每筆交易的所有子項目集都有被處理到即可。而對於一筆交易，其所有長度大於或等於三的子項目集中，第一部份的項目集被用來產生候選項目集，而第二部分的最大集合之所有子集中，其 FIT 與 SIT 和此最大集合之 FIT 與 SIT 相同的所有子集，即為第二部分的所有項目集。同理，第三部分的最大集合之所有子集中，其 FIT 和此最大集合之 FIT 相同的所有子集，即為第三部分的所有項目集，而第四部分的最大集合之所有子集，即為第四部分的所有項目集，因此，我們如要確保每筆交易的所有子項目集都有被處理到，只要將每一部分的最大集合遞延即可。根據以上的討論，我們可以得到規則三。

規則三：延遲交易拆解之最大集合拆解規則

當我們掃描到一筆被遞延的項目集時，我們將此項目集視為一筆交易，因此對其拆解除了滿足規則二外，另須符合以下規則：

(1) 若為第二部分遞延過來的交易，則以後對其拆解出的所有子集皆須含有此交易的 FIT 與 SIT，因此只須拆解出其第一部分與第二部分最大集合。

(2) 若為第三部分遞延過來的交易，則以後對其拆解出的所有子集皆須含有此交易的 FIT，因此，只須拆解出其第一部分、第二部分與第三部分最大集合。

(3) 若為第四部分遞延過來的交易，則仍須拆解出四個部分的最大集合。

2.2.2 候選項目集的產生

引理三：

對於一項目集 A，其成為加權頻繁項目集所需要的最小計數，必小於或等於其子項目集成為加權頻繁項目集所需要的最小計數。

理由：根據式(4)，因為項目集 A 的最小計數為最小權值除以項目集 A 的權值，又項目集 A 的權值必大於或等於其子項目集的權值，因此項目集 A 所需的最小計數比其子項目集所需的最小計數少。

引理四：

對於一項目集 A=$\{a_1,a_2,\ldots,a_n\}$，其權值檢核交易 B=$\{b_1,b_2,\ldots,b_n\}$，若兩者的前 k-1 項相同，但第 k 項項目不同，則項目集 A 中包含其前 k 項的子項目集皆不為頻繁項目集。

理由：因為項目集 A 與其權值檢核交易兩者的第 k 項項目不相同，根據引理一，項目集 A 不為頻繁項目集，由於在資料庫中資料是經過排序的，所以項目集 A 中包含其前 k 項的子項目集其計數必小於或等於項目集 A 的最小計數。根據引理三，項目集 A 中包含其前 k 項的子項目集要成為頻繁項目集所需的最小計數必須大於或等於 A 的最小計數。因此，項目集 A 中包含其前 k 項的子項目集必不為頻繁項目集。

引理五：

對於一項目集 A=$\{a_1,a_2,\ldots,a_n\}$，其權值檢核交易 B=$\{b_1,b_2,\ldots,b_n\}$，若兩者的前 k-1 項相同，但第 k 項項目不同 ($k < 4$)，則項目集 A 的所有第一部份項目集，皆不為頻繁項目集。

理由：因為項目集 A 與其權值檢核交易兩者的第 k 項項目不同，且 k 小於 4，根據引裡四，項目集 A 中包含其前 k 項的子項目集皆不為頻繁項目集。因為項目集 A 的所有第一部份項目集其前三項皆與 A 相同，所以項目集 A 的所有第一部份項目集皆不為頻繁項目集。

根據規則二，我們須將所掃描到的交易之第一部份的所有項目集列為候選項目集，但所產生的候選項目集，可能會很多，因此，我們利用規則四來漸進式地產生及刪減候選項目集。

規則四：產生及刪除候選項目集規則

對於一筆交易 A={a_1,a_2,…,a_n}，我們利用下列演算法來產生及刪除候選項目集：

```
For ( i=n; i>=3; i-- ) {
  將 A 的第一部份長度為 i 的每一個項目集 B 與其檢核交易比較
  1. 利用定理一判斷項目集 B 是否為頻繁項目集
  2. 若不為頻繁項目集，利用引理四或引理五來減少產生候選項目集，設項目集 B 與其檢核交易的前 j-1 項皆相同，但第 j 項不相同，則其長度大於或等於 j 的子項目集可利用引理四刪減，長度小於 j 的子項目集可利用下列演算法來判斷
  k=j-1;
  go to 4.
  3. 若為頻繁項目集，因未知其計數，故將其列為候選項目集，並利用下面的演算法來判斷項目集 B 中，哪些子項目集可能為頻繁項目集。
  k=i-1;
  go to 4
  4. while ( k >= 3 ) {
    if (項目集 B 中長度為 k 的子項目集沒被判斷過) {
      if (利用定理一判斷項目集 B 中長度為 k 的子項目集為頻繁項目集) {
        將其列為候選項目集;
        k--;
      }
      else break;
    }
    else break;
  }
}
```

在此我們先以一個例子來說明如何產生候選項目集，以及如何拆解與遞延交易，但在此處我們先避開所有資料結構上的問題，我們把資料結構的問題放第 3 節來討論。

由於在第一次掃描資料庫時，我們已經產生了所有的 1-頻繁項目集與 2-頻繁項目集。所以第二次掃描資料庫時，我們只要產生長度 3 以上的頻繁項目集即可。我們使用表一的資料庫及表二的權值表，加權最小支持度為 40%為例子來做說明。

根據規則四，先將此交易的第一部份之長度為 7 的項目集與其檢核交易做比較。因為第一部份之長度為 7 的項目集為{1,2,3,4,5,6,7}，根據式(4)我們可以得到其要成為頻繁項目集所須的最小計數為 2，所以項目集{1,2,3,4,5,6,7}的權值檢核交易為第二筆交易。因為第二筆交易與項目集{1,2,3,4,5,6,7}只有第一項項目相同，因此根據引理五可知項目集{1,2, 3,4,5,6,7}之第一部份的所有項目集皆不可能成為頻繁項目集，因此掃描完第一筆交易後不須產生任何候選項目集。

第一筆交易的第二部分之最大集合為{1,2,4,5,6,7}，根據式(4)我們可以得到其要成為頻繁項目集所須的最小計數為 2，所以項目集{1,2,4,5,6,7}的權值檢核交易為第二筆交易。根據規則二之 2(a)，因為兩者的 SIT 不相同，因此第二部分最大集合不必遞延。

第一筆交易的第三部分之最大集合為{1,3,4,5,6,7}，根據式(4)我們可以得到其要成為頻繁項目集所須的最小計數為 2，所以此項目集{1,3,4,5,6,7}的權值檢核交易為第二筆交易。根據規則二之 2(b)，因為 FIT 相同，因此第三部份最大集合{1,3,4,5,6,7}必須遞延。另外，根據規則三，為了註明該最大集合以後拆解時皆須保留其 FIT，因此我們遞延此項目集的同時附加一個標記“*”。

第一筆交易的第四部分之最大集合為{2,3,4,5,6,7}，根據式(4)我們可以得到其要成為頻繁項目集所須的最小計數為 2，所以項目集{2,3,4,5,6,7}的權值檢核交易為第二筆交易。根據規則二之 2(c)，因為第二筆交易的 FIT 小於第四部分最大集合中的所有項目，因此第四部分最大集合{2,3,4,5,6,7}也必須遞延。當處理完第一筆交易，我們可以將其自資料庫中刪除，並將後面的交易遞補上來，因此我們可以得表六。

表六：處理完第一筆交易之資料庫內容

交易編號	交易內容	是否剛被遞延
1	{1,3,4,5,6,7}	
2	*{1,3,4,5,6,7}	Yes
3	{1,3,5,6,7}	
4	{2,3,4,5,6,7}	Yes
5	{2,3,4,6,7}	

當掃描一筆交易時，我們同時比較與計數目前所產生的所有候選項目集，我們使用如圖二的資料結構記錄候選項目集的計數，其中*項目集陣列*的每一元素儲存此候選項目集的每一個項目，而*比較索引*則指示目前與交易比較至項目集陣列中的第幾個項目：

項目集計數	比較索引	項目集陣列

圖二：候選項目集資料結構

對候選項目集的計數方式如下：由左而右掃描交易中的每一項目，並同時比較所有的候選項目集，若某一候選項目集中的項目與交易中的項目相同，則其比較索引加 1，若比較索引所指示的項目，已是此候選項目集的結尾，則表示此候選項目集被此交易所包含。

3. 儲存交易資料的資料結構

由於要常對交易資料做新增的動作，而高度平衡樹有著相當良好的新增效能，因此我們儲存交易資料的資料結構是使用易於新增處理的高度平衡樹。若樹有 N 個節點，則其能在 O(log N)的時間內完成新增的工作。因此我們現在只需要解決如何快速尋找檢核交易的問題即可。

3.1 快速尋找權值檢核交易的方法

由於在找尋加權頻繁項目集之過程中，各項目集之權值檢核交易所在的位置都不同，我們可以利用式 (4)來快速求出權值檢核交易為交易資料庫中的第幾筆交易，但是我們如何從資料結構中，快速的取出此筆權值檢核交易是一大問題。由於我們無法從高度平衡樹中直接取出第 k 筆交易，也就是第 k 小的交易。因此，我們提出了一個資料結構---高度平衡索引樹(AVL-index tree)來解決這個問題。

高度平衡索引樹和高度平衡樹的差異在於，每個節點多記錄了其左子樹的節點數目。高度平衡索引樹的優點，即是可以快速找到檢核交易的所在，因為它可以很容易找到在所有資料中第 k 小的資料。從整個樹中找出第 k 小的資料節點的方法如下：我們利用變數 sortnum 來記錄有多少節點比目前所在節點為小。從根節點開始，對於每個節點判斷第 k 小的資料節點在其左邊或右邊，若 sortnum 加上目前所在節點所紀錄的左子樹的節點數目後比 k 小，則表示要尋找的節點在右邊，須向其右子樹的節點移動，並將目前節點的左子樹數

目加一後，累加到 sortnum，就可以知道有多少節點比移動後所在節點小。若 sortnum 加上目前所在節點所紀錄的左子樹的節點數目後比 k 大，則表示要尋找的節點在左邊，須向其左子樹的節點移動，此時 sortnum 不變。

在高度平衡索引樹中尋找第 k 小的資料其所需時間為 O(log N)，因此整個高度平衡索引樹的新增、搜尋與索引等動作，皆可於 O(log N) 的時間內完成。

另外，當我們在實作 WDTD 演算法時，只將資料庫中的交易依照 FIT 的大小做排序即可，例如：將所有 FIT=1 的交易擺在資料庫中的最前面，接著放 FIT=2 的所有交易，其後放 FIT=3 的所有交易……，以此類推。而 WDTD 每次只讀入相同 FIT 的交易作處理，而當具有相同 FIT 的交易被讀入高度平衡索引樹時，就會被排序。這樣的做法，一次只處理資料庫中某一部份的交易資料，因此可以節省許多的記憶體。

4. 實驗結果

我們在第 1 節中分析了 MINWAL 演算法對於每筆交易大約須拆解 $\sum_{i=1}^{m} c_i^n$ 次，其中 m 為最長的候選項目集長度，n 為平均交易長度，但由於 MINWAL 不能很有效的減少候選項目集的產生，因此很容易產生和交易一樣長的候選項目集，若存在與交易一樣長的候選項目集，則對於每筆交易的拆解次數大約為 $\sum_{i=1}^{n} c_i^n = 2^n$ 次，因此我們在以實驗驗證時，我們認為變動平均交易長度最能顯示出我們的延遲交易拆解演算法與 MINWAL 演算法在效率上的差異，所以我們固定了產生模擬資料庫的其他變數，只變動參數 n，所產生之模擬資料庫的參數如表七所示：

表七：資料庫參數表

參數	意義	值
K	交易筆數	100000
n	平均交易長度	10,11,12,13,14
I	項目總類	1000

我們總共使用 5 個資料庫 mad10-10, mad10-11, mad10-12, mad10-13 和 mad10-14，其參數 n 分別為 10, 11, 12, 13 和 14。我們將 MINWAL 的執行時間，與 WDTD 的執行時間列於表八，其中執行時間的單位為毫秒：

表八：執行結果

資料庫名稱	MINWAL 執行時間	WDTD 執行時間	速度比
mad10-10	62630	19077	3.28
mad10-11	151477	22492	6.73
mad10-12	418722	27769	15.08
mad10-13	1160899	33978	34.17
mad10-14	3593356	33918	105.94

我們從實驗中可以看出 MINWAL 演算法在交易平均長度變大時，其所花費的時間也會跟著大量的成長，這是因為其對於每一筆交易需要將近 2^n 次的拆解，所以會隨著平均交易長度的增加而花費更大量的時間，但是 WDTD 並不會受到平均交易長度太大的影響。WDTD 所花費的時間反而比較受到候選項目集多寡的影響，在 mad10-10 到 mad10-13 的資料庫中，候選項目集跟隨著交易平均長度的增加而變多，所以 WDTD 所花費的時間有些微的增加，但 mad10-14 的資料庫其候選項目集個數與 mad10-13 差不多，所以所花費的時間也差不多。因此從實驗中我們可以得知 WDTD 的效能要比 MINWAL 好得多。

5. 結論

我們對於加權的關聯規則提出了相當快速的求解方法，我們利用延遲交易拆解的觀念來改進 MINWAL 需大量拆解交易的問題，並利用高度平衡索引樹來加快演算速度並節省更多的記憶體空間，另外，我們還提出刪減候選項目集與避免候選項目集重複的方法，加上延遲交易拆解演算法在執行中是分段處理資料庫的，所以可以節省候選項目集的比對時間。除此之外，我們尚有許多方向需要努力，將其列於後：

1. 負權值的關聯規則：目前對於加權的關聯規則，範圍僅在探討權值為正值時的情形。可是在某些應用上，例如對於贈品的計算，由於贈品對公司來說是負利潤的，但卻也是吸引顧客的手段，如何挖掘出贈品與其他商品間的關係進而吸引更多的客戶是相當重要的。
2. 瀏覽路徑(Traversal Pattern)的探討：近年來電子商務(Electronic Commerce)提供了另一種形式的銷售方式。而對於一個網路商店的經營者而言，如何提供客戶所需要的資訊與服務，以加強其購買商品的意願便是相當重要的事情。而瀏覽路徑的挖掘，即是要找出大多數客戶的瀏覽行為，以加快客戶搜尋商品的速度。但過去的研究 [9]只計算網頁被點選次數的多寡來決定其瀏覽行為，這種做法並不適當，因為索引網頁被點選的比例比其他網頁高很多，但推薦索引網頁給客戶瀏覽，並不能帶來實質上的好處。較好的做法是針對不同的網頁，給予不同的權值，如此才能求出真正對公司營運有所幫助的資訊。

我們希望能將我們現在所研究的方法，推廣到這幾個部份，使得能從龐大的資料中挖掘出更有用的資訊。

誌謝

這篇論文的研究成果是國科會計劃（NSC-90-2213-E-030-003）的一部份。我們在此感謝國科會經費支持這個計劃的研究。

參考文獻

[1] R. Agrawal and et al.: Fast Algorithm for Mining Association Rules. In *Proceedings of International Conference on Very Large Data Bases*, pages 487-499, 1994.

[2] R. Agrawal and et al.: Database Mining: A Performance Perspective. In *IEEE Transactions on Knowledge and Data Engineering*, pages 914-925,1993.

[3] Rakesh Agrawal, and et al.: Mining Sequential Patterns. In *Proceedings of International Conference on Data Engineering*, pages 3-14,1995.

[4] C.H. Cai, Ada W.C. Fu, C.H. Cheng and W.W. Kwong: Mining Association Rules with Weighted Items. International Database Engineering and Applications Symposium (IDEAS), pages 68-77,1998

[5] J iawei Han, Jian Pei, Yiwen Yin: Mining Frequent Patterns without Candidate Generation. *ACM SIGMOD* Conference, pages 1-12, 2000.

[6] Ramakrishnan Srikant and Rakesh Agrawal: Mining Generalized Association Rules. *In Proc. of the 21st Int'l Conference on Very Large Databases*, Zurich, Switzerland, September, pages 407-419, 1995.

[7] J.S. Park, M.S. Chen, and P.S. Yu: An Effective Hash-Based Algorithm for Mining Association Rules. In *Proceedings of ACM SIGMOD*, 24(2):175-186, 1995

[8] Ashok Savasere, Edward Omiecinski, Shamkant Navathe: An Efficient Algorithm

for Mining Association Rules in Large Database. *VLDB*, pages 432-444, 1995.

[9] Ming-Syan Chen, Jong Soo Park, Philip S. Yu:Efficient Data Mining for Path Traversal Patterns. *Transactions on Knowledge and Data Engineering. IEEE*, pages 209-221, 1998.

[10] Wei. Wang, Jiong Yang, Philip. S. Yu: Efficient Mining of Weighted Association Rules (WAR). *KDD*, Pages 270-274, 2000.

[11] S.J. Yen and Arbee L.P. Chen: An Efficient Approach to Discovering Knowledge from Large Databases. In *Proceedings on Parallel and Distributed Information Systems*, pages 8-18,1996.

[12] S.J. Yen and Arbee L.P. Chen: An Efficient Data Mining Technique for Discovering Interesting Association Rules. *DEXA Workshop*, pages 664-669, 1997.

[13] S.J. Yen and C.W. Cho. "An Efficient Approach to Discovering Sequential Patterns from Large Databases," *Lecture Notes in Artificial Intelligence: Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 685-690, 2000.

[14] S.J. Yen "Mining Generalized Multiple-Level Association Rules," *Lecture Notes in Artificial Intelligence: Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 679-684, 2000.

[15] S.J. Yen and A.L.P. Chen. "A Graph-Based Approach for Discovering Various Types of Association Rules," *IEEE Transactions on Knowledge and Data Engineering*, to appear, 2001.

[16] S.J. Yen and C.W. Cho. "An Efficient Approach for Updating Sequential Patterns Using Database Segmentation," *International Journal of Fuzzy Systems Special Issue on Soft Computing and Data Mining*, pages 422-431, 2001.