第四章 類 FP-tree 出現摘要儲存結構探勘法

上一章節所描述的兩個摘要結構方法，若對每筆交易中所包含的各種資料項子集皆取出儲存，當資料項的種類繁多時，會產生極大量的資料項集，因此所建構之絡結構會極為龐大。所以本論文的想法是採用類似 FP-tree 的方式，對交易中由常見資料項所形成的資料項集樣式組織其出現摘要資訊。然而為了避免建立 FP-tree 時先掃描一次原始資料，計數各資料項出現次數大小的步驟，一開始我們會先依資料項的字母順序來形成資料項集，隨後再調整成 FP-tree 結構。

此外，在資料流環境中，每輸入一筆新交易就要做一次出現摘要資料結構的更新。此時各資料項的支持度計數值可能會有所改變，以致於樹中路徑節點的順序不符合 FP-tree 結構的定義。若為了維持 FP-tree 結構而隨時做一次如 2-2.2 所述 AFPIM 演算法來調整處理，如此一來可能會造成每個時間點皆要做一次調整，而耗費太多處理成本。因此我們會先依先前 FP-tree 中資料項的大小順序將新交易所包含的資料項集樣式加入樹狀結構中，若有原來 FP-tree 結構中不存在的資料項則依字母順序排列，置於該樣式最後位置。每過一段固定時間再依資料項最新的支持度計數值調整結構，並刪除非常見資料項，以維持較精簡的樹狀結構。

4-1 平均時戳探勘法（Average TimeStamp mining method）

平均時戳探勘法（Average TimeStamp mining method），簡稱 ATS 演算法。採用上述之類 FP-tree 結構來儲存資料項集的平均時戳摘要資訊。樹的每個節點

儲存中 3-1 所定義之 (e, t_s, f, sum)資訊，且在 FP-tree 的 Header table 中新增一個開始時間點 t_s 欄位，用來記錄各資料項開始加入 FP-tree 的時間點，以便計算該資料項的最近支持度，用來判斷其是否為最近非常見資料項。

4-1.1 FP-tree 結構調整方法

我們在前面章節 2-2.2 中已說明過 AFPIM 演算法動態調整 FP-tree 結構的方法步驟，在此只需說明如何對節點中的出現摘要資訊做對應處理。

(1) AFPIM 演算法步驟 2.1) 新增節點。

當 X 節點分解成 X 跟 X'兩個節點時，是準備要將新的 X 節點與其子節點 Y 做順序上的調換，因此新的 X 節點要記錄 X.e 與資料項集 Y.e 出現在相同交易中的資訊，而扣除掉和 Y.e 同時出現的 X.e 之出現資訊，則保留在 X'節點中。因此節點 X'的計數值設為 X.f − Y.f ，其時戳總合設為 X.sum − Y.sum ，並將其開始時間點 X'.t_s 設為節點 X 的開始時間點 X.t_s。而 X.f、X.sum 及 X.t_s 皆設為與 Y 相同。

(2) AFPIM 演算法步驟 2.3) 合併節點。

當 Y 節點合併 Z 節點時，表示資料項集 Y.e 與資料項集 Z.e 是相同資料項集，其出現摘要資訊中的出現次數及時戳總合要合併整合存到節點 Y 中。因此除了將節點 Z 的計數值加到節點 Y 的支持度計數值外，還需要將 Z 的時戳總合 Z.sum 加到 Y 的時戳總合 Y.sum，且節點 Y 的開始時間點取 Y.t_s 與 Z.t_s 兩值中的較小值。

4-1.2 ATS 演算法

ATS 演算法在每個時間點 t，會執行維護 FP-tree 結構中出現摘要資訊的處理（步驟 1 及 2），每隔固定期間，再做調整 FP-tree 結構的處理（步驟 3）。

在目前時間為 t 時，維護 FP-tree 結構的步驟如下：

步驟 1) 加入新輸入交易 T_t。

將交易 T_t 所包含的資料項依原 FP-tree 資料項的順序排序成一個樣式，加入 FP-tree 結構中。該樣式所對應的路徑上之每個節點 N，對節點中支持度計數值 $N.f$ 加上 1，時戳總合 $N.sum$ 加上 t。若該樣式對應路徑的節點不存在，則新增對應節點，且將其 f 設為 1，t_s 及 sum 設為 t。

此外，對每個 T_t 中所包含的資料項，其在 Header table 中對應的支持度計數值加 1。

當 $t \geq w$，表示目前交易視窗已有 w 筆交易後，便可繼續進行以下步驟。

步驟 2) 刪除過時節點。

檢查整個 FP-tree 結構中是否有節點 N，其平均時戳（$N.sum/N.f$）小於 $N.t_s + (CTL_t^{first} - N.t_s)/2$，將這些節點從 FP-tree 結構中刪除。

當目前時間點 t 為探勘視窗大小 w 之一半的倍數時，進行以下步驟。

步驟 3) 調整 FP-tree 結構並刪除最近非常見資料項。

根據原本 FP-tree 結構之 Header table 中資料項排列順序，採用泡沫排序法依目前支持度計數值做遞減排序，以 AFPIM 方法調整 FP-tree 結構。調整 FP-tree 結構完後，由 Header table 中計算各資料項 I_i 的最近支持度 $R\sup_t^{DS}(I_i)$。若 $R\sup_t^{DS}(I_i)$ 小於 ε ，則延著資料項 I_i 的橫向連結串列逐一刪除其對應節點。

4-2 出現頻率改變點法（Frequency Changing Point mining method）

出現頻率改變點探勘法(Frequency Changing Point mining method)，簡稱 FCP 演算法 。此方法將資料項集的出現頻率改變點摘要資訊，同樣以上述類 FP-tree 結構來儲存，並動態調整。樹中的每個節點中儲存 3-2 所定義之(*e, f, t_s, t_e, C_d, Rqueue*)資訊，且在 FP-tree 的 Header table 中同樣新增一個開始時間點 t_s 欄位，用來記錄各資料項開始加入 FP-tree 的時間點，用來計算判定要刪除的最近非常見資料項。

4-2.1 FP-tree 結構調整方法

此方法運用 2-2.2 中說明過的 AFPIM 演算法動態調整 FP-tree 結構時，調整節點中的出現摘要資訊處理方式如下：

(1) AFPIM 演算法步驟 2.1) 新增節點。

X 節點分成 X 跟 X'兩個節點，如同上一節中 ATS 演算法中的解釋，此步驟的處理是要將 X.*e* 只和其子節點 Y 中 Y.*e* 一起出現的摘要資訊儲存在新的 X 節點中，而節點 X'則保留扣除掉和 Y.*e* 同時出現的 X.*e* 之出現資訊。因此 X'.*f* 設為 X.f − Y.f ，X'.t_s 設為 X.t_s，X'.t_e 設為 X.t_e。

將 X'.*Rqueue* 設成和 X.*Rqueue* 相同，令在原來 X'.*Rqueue* 中所包含的改變點對表示為<(t_{r1},C_{r1}), (t_{r2},C_{r2}),…, (t_{rk},C_{rk})>，其中 t_{r1}, t_{r2}, …, t_{rk} 為一遞增數列，而落在(Y.t_s, Y.t_e)間的改變點對為<(t_{rm},C_{rm}) , .., (t_{rn},C_{rn})>（即 $t_{r(m-1)} \le Y.t_s < t_{rm}$，且 $t_{rn} < Y.t_e \le t_{r(n+1)}$）。則將 $C_{r(n+1)}$設為 $\sum_{i=m}^{n} C_{ri} - Y.f$ ，再將 C_{ri}（i=m, m+1, …, n）

皆設為 0。

X'.C_d 則設為 X'.*Rqueue* 中所有 C_r 的總合 $\sum_{i=1}^{k} C_{ri}$ 。

最後將 X 的 X.f、X.t_s、X.t_e、X.C_d 及 X.*Rqueue* 皆設為與 Y 相同。

[範例 4.1]

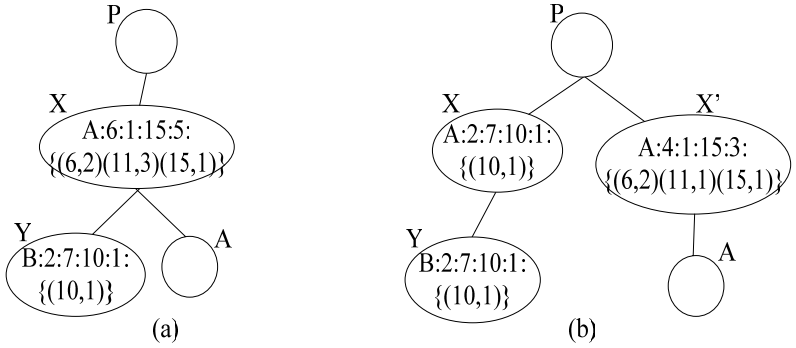

圖 4.2.1 新增節點

以圖 4.2.1(a)所示之例，X.*Rqueue* 中有三個改變對(6,2)、(11,3)、(15,1)，其中 t_r 介於 Y.t_s（即 7）和 Y.t_e（即 10）之間的改變點對為空集合，大於 Y.t_e 的最小改變點為(11,3)，因此需扣掉 2（Y 的計數值 Y.f），其他兩個改變點對維持不變加進 X'.*Rqueue* 中。新增節點後的處理結果如圖 4.1.2(b)所示。

上述調整 X'中 *Rqueue* 的方式，是因為落在(Y.t_s, Y.t_e)間的改變點對，必須扣除有出現 Y.e 之出現次數。然而無法精確估算出這些次數（C_r 應扣除的值），因此將這些改變點對的 C_r 值設為 0。則在調整 X'的開始時間點及支持度計數

值時（將 X'.*f* 扣除 C_r 值），可保證調整後之 X'.*f* 不會比實際出現值小。但此誤差並不會繼續累積變大，因為 $C_{r(n+1)}$ 中之值為精確值，因此當 X'.t_s 調整到 t_{rn+1}，調整後之 X'.*f* 便恢復為其實際出現值。

(2) AFPIM 演算法步驟 2.3) 合併節點。

Y 節點合併 Z 節點時，將節點 Z 的計數值加到節點 Y 的支持度計數值，Y 的開始時間點取 Y.t_s 與 Z.t_s 兩值中的較小值，最近時間點取 Y.t_e 與 Z.t_e 中的較大值；過時計數值 Z.C_d 加進 Y.C_d；改變點對佇列 Z.*Rqueue* 合併到 Y.*Rqueue*，即將 Z.*Rqueue* 的改變點對依改變點 t_r 的大小插入 Y.*Rqueue* 中。

[範例 4.2]

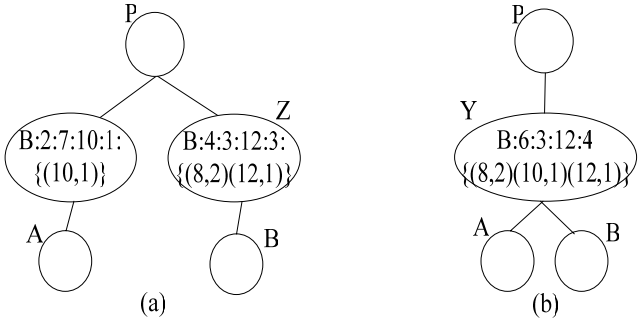

圖 4.2.2 合併節點

以圖 4.2.2(a)所示之例，將 Z.*Rqueue* 的改變對(8,2)及(12,1)依改變點大小加進 Y.*Rqueue* 中，最後合併結果如圖 4.2.2(b)所示。

上述合併 Y.*Rqueue* 及 Z.*Rqueue* 後，對於某些改變點對(t_r', C_r')，t_r'到上一次

改變點間 Y.e 的出現次數可能會增加，但無資訊可得知對計數值C_r'精確的增加值，因此我們保留原來的C_r'值可能比實際值小，但可保證在調整 Y 至t_r'時，調整後之 Y.f 不會比實際出現值小。此誤差同樣不會繼續累積變大，因為若C_r'值比實際值少 n 次，則 Y.C_d所累計值也會比實際值少 n 次，下一次發生頻率改變點t_r''時，其計數值C_r''由$(Y.f-1)-Y.C_d$算出，會比實際值多 n 次。因此當以t_r'重設 Y.t_s時，$Y.f = Y.f - C_r$'會比實際值多 n 次，但當遇到以t_r''重設 Y.t_s時，$Y.f = Y.f - C_r$''又會將多出來的 n 次扣除掉，恢復為實際出現次數。

4-2.2 FCP 演算法

在目前時間點為 t 時，FCP 演算法維護 FP-tree 的處理如以下步驟 1 及步驟 2。

步驟 1) 加入新輸入交易 T_t。

將交易 T_t所包含的資料項依原 FP-tree 資料項的順序排序成一個樣式，加入 FP-tree 結構中。該樣式所對應路徑上的每個節點 N，其支持度計數值 $N.f$ 加上 1，此外並計算$t - N.t_e$是否大於$(1/S_{min})$，若是則執行步驟 1.1 記錄出現頻率改變點。最後將 $N.t_e$設為 t。

步驟 1.1) 記錄出現頻率改變點。

節點 N 對應的資料項集 $N.e$ 在時間點 t 時出現一個頻率改變點，將頻率改變點 t_r 設為 t，t_r 到上一個改變點間的計數值（不包含 t_r 這一次計數值）C_r 設為$(N.f-1)-N.C_d$，將頻率改變點對(t_r, C_r)加入

N.Rqueue，最後將 $N.C_d$ 設為 $N.f-1$。

若該樣式對應的路徑節點不存在，則新增對節點，並將其 f 設為 1，t_s 及 t_e 設為 t，C_d 設為 0，且 *Rqueue* 設為空佇列。

最後，對每個 T_t 所包含的資料項，其在 Header table 中對應的支持度計數值加 1。

當 $t \geq w$，表示目前交易視窗已有 w 筆交易後，便可繼續進行以下步驟。

步驟 2) 調整開始時間並刪除過時節點。

探訪 FP-tree 中每個節點，對正在探訪的節點 *N* 依序執行步驟 2.1 及步驟 2.2。

步驟 2.1) 調整開始時間。

若節點 *N* 開始時間點 $N.t_s$ 小於 CTL_t^{first} 且 *N.Rqueue* 不為空佇列，則檢查 *N.Rqueue* 中第一個頻率改變點對(t_r, C_r)是否符合 3-1 節步驟 2)所述的三種調整開始時間點情形之一，若是則執行步驟 2.1.1。

步驟 2.1.1) 調整開始時間點及支持度計數值。

調整節點 *N* 中資料項集 *N.e* 的開始時間點 $N.t_s$ 的方式是，將支持度計數值 *N.f* 扣掉過時計數值 C_r，將開始時間 $N.t_s$ 設為此頻率改變點 t_r，並將累計過時計數值 $N.C_d$ 減掉 C_r。調整完後，將此頻率改變對從 *N.Rqueue* 中移除。

步驟 2.2) 刪除過時節點。

若節點 N 支持度計數值 $N.f$ 為 0 或最近一次出現的時間點 $N.t_e$ 小於 CTL_t^{first}，這些情況皆表示此節點中的資料項集 $N.e$ 沒有出現在目前交易視窗 CTL_t 中，因此將此節點從 FP-tree 結構中刪除。

當目前時間點 t 為探勘視窗大小 w 之一半的倍數時，進行以下步驟 3。

步驟 3) 調整 FP-tree 結構並刪除最近非常見資料項。

根據原 FP-tree 結構之 Header table 中資料項排列順序，採用泡沫排序法依目前支持度計數值做遞減排序，以 AFPIM 方法調整 FP-tree 結構。

調整 FP-tree 結構完後，由 Header table 中計算各資料項 I_i 的最近支持度 $R\sup_t^{DS}(I_i)$。若 $R\sup_t^{DS}(I_i)$ 小於 ε，則延著資料項 I_i 的橫向連結串列逐一刪除其對應節點。