以樹狀序列挖掘企業系統效能規則

林豔
中央大學資訊管理研究所
韋俊仲、鄭明松、許秉瑜
中央大學企業管理學系

摘 要

企業系統被視為企業運轉的基石，在企業中扮演著極重要的角色，因此系統的穩定性相當地重要。本文主要是結合資料挖礦技術中的序列關聯規則與有根樹資料結構，發展出適合 Tree-Based 系統效能資料特性的演算法－SPT（Sequence of Performance Trees）。針對樹狀序列資料結構，以及數值屬性資料的處理技巧提出的 SPT 演算法，可針對系統管理者感興趣的階層進行挖掘，找出的關聯規則可以幫助系統管理者從大量的系統效能資料中找出潛藏的訊息，同時可以發現所觀測現象發生的序列變化，這些資訊可以提供系統管理者調校系統、分析和矯正錯誤、增進系統效能，讓企業系統運作更加穩定及可靠。

本文中 SPT 演算法應用在企業系統的效能分析上，可以幫助系統管理員整理出很多有意義的特徵規則或系統效能的型態（patterns）。

關鍵字：資料挖礦、樹狀序列資料、關聯規則、企業系統效能型態。

Mining Enterprise System Monitoring Trees with Sequences

Yen Lin
Department of Information Management, National Central University
Chun-Chung Wei, Ming-Sung Chen, Ping-Yu Hsu
Department of Business Administration, National Central University

Abstract

For enterprises implementing enterprise systems, the systems become the main pipeline where information flows integrated. If the pipelines get clogged, the whole enterprises can grind to a stall. Therefore, maintaining the stability and availability of enterprise systems become one of the most important duties of MIS department. To properly maintain the systems, not only do system administrators need to react to problems but also need to predict the patterns of system performance and act proactively.

The paper proposes to find the patterns with SPT, Sequences of Performance Trees. The patterns and underlying data are represented as sequences of monitoring trees. A tree is a collection of system performance indicators in a particular moment. Trees are organized to sequences to keep system performance in a consecutive time intervals. Since a system administrator may not be interested in the evolution patterns of entire trees, the paper also proposes an interest vector to pick interested subtrees and perform association rule mining on those subtrees. With the addition of the interest vector, the definitions of support and sequence equivalence are all modified. A demo system is also shown in the paper.

Keywords: Data Mining, Tree-based structure, association rules, enterprise system, Performance Pattern.

壹、緒論

近年來，許多企業紛紛採用大型企業系統（Enterprise System，以下簡稱 ES），如企業資源規劃系統（Enterprise Resource Planning，以下簡稱 ERP 系統）來整合企業運作資料，以即時提供企業處理和制定決策之用。由於大量資料集中於系統中，一旦出現問題而停頓的話，將會造成企業莫大的損失，因此系統的穩定性相當重要。系統管理者同時必須能隨時掌握系統的運作，了解資源使用和分配的情形，以確保系統不會因資源短缺而導致效能下降，影響到企業的運作。

導致系統效能下降的因素非常多，系統管理者想要了解系統的效能，可以利用系統提供的監控工具。監控工具可以收集系統效能的資訊，一般以圖表方式呈現予系統管理者，從中可得知系統過去的運作情形。但如想了解系統效能變化是否有一定型態或軌跡可循，則需借重能發掘規則的工具。

有些系統效能問題的型態以人工目視資料並不容易識別。例如，data buffer 不夠會導致 24 小時後 CPU 異常忙碌或 program buffer 不夠用會在三小時候導致並 swap 異常升高，或資料庫的存取速度會週期性的變慢。

而資料挖礦技術則可依據使用者需求自龐大的資料庫中選擇出合適的資料，並加以處理、轉換、挖掘和評估找出有用的法則和知識。本論文研究是利用資料挖礦的技術，對系統效能進行整合與預測，目的是找出效能現象發生的特徵規則。這樣的關聯規則可以幫助系統管理者了解系統發生問題的成因與關係，並有助於系統的監控，更能作為管理者日後調整系統時參考之用。

系統效能狀況會隨時間的變化狀況也隨之不同，一般企業系統都可在特定的時間點擷取系統效能的資料。經過長時間的蒐集，記錄在資料庫中的資料便具時間順序的關係。其中每一筆紀錄都反應了當時整個系統的效能狀況。

在 SAP 之類的 ERP 系統中，每一筆系統效能資料皆可以一個效能樹來表示[21]，效能樹為監測企業系統效能時，各種屬性依其系統架構相關性所形成的多階層概念樹，樹的底層為被監測元件的屬性，例如硬碟的空間、資料讀取速度等，樹的中間與上層各點為被監測的元件，而效能樹的根則反應出整個企業系統的效能狀況，詳細的效能樹結構請參考第二節。

為能發掘相關的效能樹關聯規則，本論文特別設計了 SPT（mining in Sequences of Performance Trees）演算法。此演算法將數值屬性的系統效能資料依系統管理者的知識分類成幾個區間，再將分割後的區間對映到連續的整數，此時效能樹的每個節點都成為一個整數。然後再透過編碼將效能樹轉換成數字串，再以資料挖礦的概念找尋數字串的序列特徵規則或樣式。在大型企業系統中，效能樹常常很大，但並不是所有效能都是管理者有興趣的。此論文也設計了特殊的 mask，幫助系統管理者要求 SPT 只針對其有興趣部份的 SPT 做關聯規則的找尋。

本文以大型企業最常使用的 SAP R/3 之 ERP 系統為例，使用 SAP R/3 中 ABAP

（Advanced Business Application Programming）(Will 1999)和 BAPIs（Business Application Programming Interfaces）(SAP 1999)的方法，取得系統長時期的效能資料，再使用 SPT 演算法找出系統效能現象發生的特徵規則。

本文架構如下：第二節介紹在資料挖礦中運用關聯規則找尋序列樣式的相關研究和各種處理關聯規則的課題，以及資料挖礦技術應用在系統效能評估上之相關研究。第三節說明如何定義所要使用的樹狀資料結構，以及在實際應用時的情況。第四節詳細說明整個 SPT 演算法。第五節是由 SPT 演算法為基礎所實作出的監測系統，並以中央大學 ERP 中心運作中的 SAP R/3 系統所產生的監控資料作為模擬。第六節為結論及未來研究方向。

貳、相關研究

本文的 SPT 演算法是用來處理系統效能資料，以序列關聯規則為基礎。我們以樹狀結構的序列表示系統效能資料，同時為了簡化資料的處理工作，把原數值屬性的資料以區間表示。在本節中，我們針對本文研究範圍相關領域做一探討。

一、挖掘序列關聯規則

在一個顧客交易資料中，有時候我們希望找出一般顧客購買物品的順序型態，例如：有一個顧客可能購買了電視機之後，接著就會再去購買錄放影機。電視和錄放影機並不是在同一筆交易紀錄之中，而是有前後關係的交易紀錄，這種挖掘稱之為挖掘序列樣式（Mining Sequential Patterns）(Agrawal & Srikant 1995)。序列樣式的挖掘可分為 5 個步驟 (Agrawal & Srikant 1995)：Sort Phase、Iitemset Phase、Transformation Phase、Sequence Phase、Maximal Phase。

AprioriAll 演算法如表 1，在每一個 pass 中使用前一 pass 的 large sequences，再利用常用的 Apriori-gen 函數(Agrawal & Srikant 1994)取得全部 large (k-1) - sequences 的集合作參數產生 candidate sequences，然後計算它們的 support。依照 candidate sequences 的 support 去決定 large sequences。

表 1：AprioriAll 演算法，來源(Agrawal & Srikant 1995)

1)L_1 = {large 1-sequences}； // Result of litemset phase
2)for（k = 2；$L_{k-1}\neq 0$；k++）do
3) begin
4) C_k = New Candidates generated from L_{k-1}
5) foreach customer-sequence c in the database do
6) Increment the count of all candidates in C_k that are contained in c.
7) L_k = Candidates in C_k with minimum support.
8) end
9)Answer = Maximal Sequences in $\cup_k L_k$；

二、挖掘一般化和多階層的關聯規則

Chen et al.(1996)提到資料庫中的資料和物件通常包含概念層級（concept level）中細部的資訊，我們通常希望將這些一大串資料作概念化，將它們表示成一個較高的概念層級。Srikant & Agrawal (1995)提到許多有趣的關聯常發生在較高的概念階層，其所提出的一般化的關聯規則（generalized association rule），便是用來描述資料庫中的資料經過一般化後存在的關聯。

另一類多階層的關聯規則（multiple-level association rule）(Han & Fu 1999)，是用來描述不同概念階層的項目間的關聯，例如：在食物項目中，麵包和牛奶分屬不同的類別，麵包在第一概念階層，巧克力牛奶在第二概念階層，我們希望找出不同概念階層的項目的關聯規則，如“65%的消費者購買麵包時，也會購買巧克力牛奶”。擴展(Yen & Chen 1996)中分析大量交易資料和產生簡單的關聯規則所使用 graph-based 方法，以挖掘一般化和多階層的關聯規則(Yen & Chen 2001)。不同於傳統明確型集合(crisp-set) 挖掘的方法，Hong et al. (2003)使用模糊集合(fuzzy-set)的概念，挖掘出多階層模糊關聯規則。

在挖掘多階層關聯規則時，編碼方式是依據圖 1 各階層、各商品順序編號：在 level 1 中有兩個商品 Milk 和 Bread，依商品順序編號 Milk 為 1，Bread 為 2；level 2 中的商品可分為 Milk 和 Bread 兩類，故 Milk 和 Bread 的概念階層必須獨立編號，其中 2% Milk 是 Milk 中第一個商品，所以 2%的編號為 1，white Bread 是 Bread 概念階層中第一個商品，故用 1 表示 white；level 3 的編碼方式同上。例如：level 3 中的 Dairyland 是 2%概念階層中的第一個商品，2%則是 Milk 概念階層中第一個商品，故 Dairyland 編碼成{111}，同理，Wonder 編碼成{212}，等。假設根據商品概念階層樹，將原始交易資料編碼，產生編碼後的資料庫，如表 4。各階層 Level-1、Level-2、Level-3 的最小支持度分別為 4, 3, 3，首先針對第一層求出 Level-1 large 1-itemsets {1**}和{2**}，然後將非大項目集合的商品從 T[3]（表 2）中刪除，產生表 T[1]（表 3），以減少後續所要掃描的交易筆數；接著便是建構 Level-1 large 1-itemsets 的關聯圖，求出 Level-1 large 2-itemsets {1**, 2**}，圖 2(a)。以同樣的方法找出 Level-2、Level-3 的所有 large itemsets，圖 2 (b) (c)。

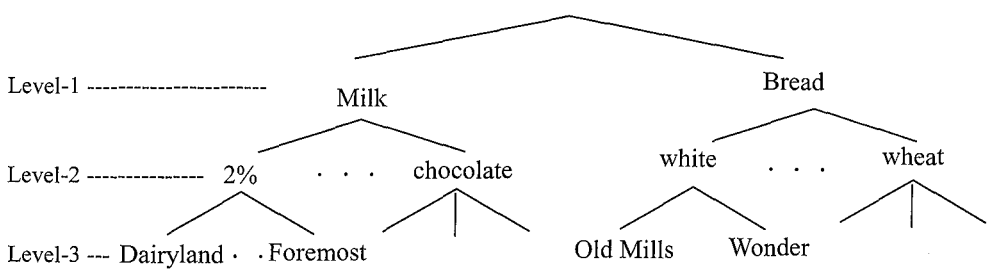


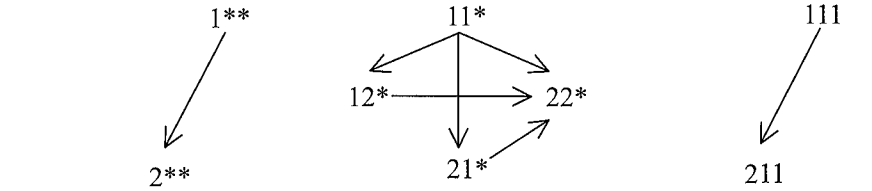
圖 1：概念階層樹，來源(Han & Fu 1999)

表 2：產生編碼後的資料庫 T[3]，來源(Yen & Chen 2001)

TID	Itemset
100	{111, 121, 211, 221}
200	{111, 211, 222, 323}
300	{112, 122, 221, 411}
400	{111, 121}
500	{111, 122, 211, 221, 413}
600	{211, 323, 524}
700	{323, 411, 524, 713}

表 3：刪除不符合的項目後的資料表 T[1]，來源(Yen & Chen 2001)

TID	Itemset
100	{111, 121, 211, 221}
200	{111, 211, 222}
300	{112, 122, 221}
400	{111, 121}
500	{111, 122, 211, 221}
600	{211}



(a) Level-1 association (b) Level-2 association (c) Level-3 association

圖 2：關聯圖，來源(Yen & Chen 2001)

上述這些從 Apriori 改良的演算法中，在同一 level 採用一致的最小支持度(minimum support)，挖掘出 large itemsets，Wang et al. (2003)提出 Adaptive Apriori，在同一 level 中可依照資料的特性，例如(k-1)-itemsets 與 k-itemsets 分別設定不同的最小支持度，以避免因低支持度可遺漏發現有趣的樣式。

三、挖掘數值關聯規則

數值屬性值域內之值極多的這個特性，使得在做資料挖礦時會有大量的計算，又產生出來的規則會過於瑣碎，所以必須先做一番處理。一般而言，數值屬性會被分類成幾個區間再來計算。Srikant & Agrawal (1996)提出如何挖掘數值屬性之關聯規則的方法，對於數值屬性，若該屬性不必分割成區間，則對映到連續的整數；若屬性需先分割，則先把屬性切割成多個細小的區間，將這些區間視為同一屬性裡不同的屬性值，再將分割後的區間對映到連續的整數。將數值屬性轉換成數值後，便可做 Apriori 演算法的處理，得到的數值關聯規則。

一般來說，大型資訊系統都會提供監控系統的工具（monitoring tool），這些工具會監控系統，並記錄著被監測系統元件的屬性狀態，而產生的大量統計資料常令系統管理者難以理解其間的關聯性;另一方面，若發生異常狀況，企業系統僅能就目前系統狀態對使用者或管理者提出警告，但卻無法預測系統下一步可能的狀態。雖然以上研究都與本文相關，但都非直接應用於大型企業系統的效能關聯規則之挖掘，因此本研究嘗試以資料挖礦之方法，針對大型企業系統之效能，找出被監控元件間之相關性，並預測系統下一步可能發生之狀態，使管理者或使用者提早預防其發生及因應，甚而提供系統管理者調校系統之參考，使系統更趨穩定及快速。

參、資料結構

一、資料簡介

本研究進行分析的資料來自於企業系統的監控程式，包含所有影響系統運作的各個元件及屬性在某特定時間點的值，例如：2000 年 2 月 1 日 01:00，CPU 的使用率為 30%、CPU 平均 load 為 20、swap 剩餘空間為 23.75mbytes、sap 使用率為 12、SQL 錯誤率為 5，每個因子分別以屬性記載其值，記錄在資料庫中，如表 6 所示。真實資料欄位比表 4 所載多出許多，但為舉例方便，僅列出表 4 中資料。

表 4：原始資料

日期	時間	CPU 使用率	CPU 平均 load	Swap 剩餘空間	Swap 使用率	SQL 錯誤率
20000201	01:00	30	20	23.75	12	5
20000201	02:00	30	20	24	12	5
20000201	03:00	42	30	26	14	7
20000201	04:00	30	10	18	12	5
20000201	05:00	30	14	23	12	7
20000331	23:00	42	20	20	30	6
20000201	06:00	30	14	23.5	12	2
20000201	07:00	30	14	20	16	5
. . .	. . .	. . .	. . .	. . .	. . .	. . .
20000331	23:00	42	20	20	30	6
20000331	00:00	30	13	22	10	8

二、樹狀資料結構

系統效能資料一般以屬性記載其值，儲存在資料庫中。通常影響系統效能的各因子具有可分類的階層關係，例如：Swap 具有使用率和剩餘空間這兩項屬性，CPU 具有平均負載和回應時間等屬性；CPU 和 Swap 都是 ES 的元件，而 SQL 錯誤率則屬於資料庫評估系統效的因子。為了表示屬性間的階層關係，以及把多屬性的資料儲存成單一屬性的資料形式，效能被組成有根樹（rooted tree）的資料結構（圖 3）來呈現。樹中非葉節點的監測值則是依據其項下節點的監測值計算而得。例如在 SAP R/3 中，所有效能樹上的節點都被賦予正常、警告、與嚴重警告三種監視值。葉節點的值是由使用者制定的數值區間決定，其他節點則由其下節點的值依公式決定。效能樹上任二節點 p 與 q, q < p 表示 q 為 p 下的節點，則 p 的監視值為

1. if ∃q , q < p and value(q) = "嚴重警告" then value(p) = "嚴重警告"
2. if ∀q, q < p value(q) != "嚴重警告"and ∃q, q < p, value(q) = "警告" then value(p) = "警告"
3. if ∀q, q < p value(q) = "正常" then value(p) = "正常"
4. if there is no q < p then value(p) is decided by users according to the measured performance.

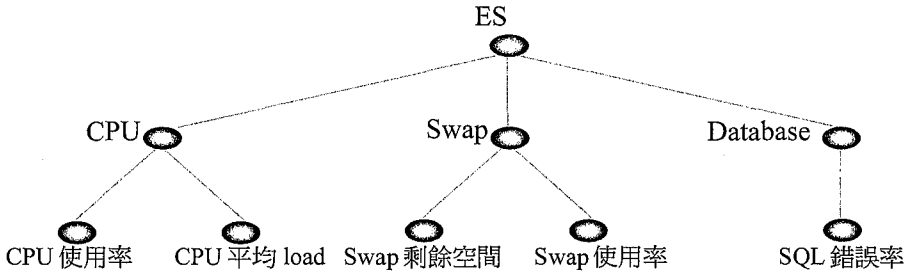


圖 3：系統效能資料以樹狀結構表示

本論文中各節點的值可為任何類型，至於非葉節點如何得到其監測值並非探討重點，本段所舉的 SAP R/3 計算方法僅用以證明效能樹的可行性與在業界的實際應用。

定義一 效能樹為一有根數，其內各節點都含有一個經由測量或計算而得的數值。

利用樹狀結構來表示系統階層關係的方式，目的是讓使用者可以根據其有興趣的階層來進行挖掘。假如系統管理者除了想了解整個系統效能的改變，還想知道 CPU 使用的情形對 Swap 的影響等，就只須要針對 CPU 和 Swap 這兩個子樹（subtrees）進行規則挖掘。在 SPT 中，欲挖掘的子樹不須在同一階層且所挖掘的規則可跨越多棵效能樹。

三、樹狀資料結構的儲存-效能向量

直接將樹狀結構進行分析並不容易，因此一般都會加以編碼，再針對編碼後的資料進行分析。如(Will 1999)中，編碼就以階層加上由左至右的順序為考量。根節點的編碼為 1、其下最左邊的節點編碼為 11、次左點編碼為 12，餘此類推。此編碼方式並未考慮將節點內的值加以儲存，而只是單純的將每個節點視為不同的物件，再找尋其間關係。本文中的編碼則需考慮配合儲存格將節點內資料存入格中，並允許使用者就編碼後節點與儲存內容進行規則挖掘。為儲存節點中的監測值，本文定義效能向量。

定義二 **效能向量為一個數字陣列，陣列中每個儲存格存有一個效能樹節點的監測值。陣列中的每個儲存格都有一個編碼，且編碼為由 0 開始的連續號碼。**

本文中採用編譯器中常用的 postorder 編碼方式來安排節點在效能向量中的位置，亦即，一個節點的編碼一定比其下的節點大，而同一階層的節點則右邊比左邊有較大的編碼。Postorder 編碼演算法如下：

```
Algorithm Postorder
    Input: p – 效能樹上節點
       Code – 可用編碼
    Output: 已用掉編碼
    For each child of p, from left to right ,q, do
      Newcode = postorder(q, newcode + 1)
    End loop
    p.code = newocode+1
    return p.code
```

如果有一個效能樹的根節點為 r, 則此效能樹所有節點的 postorder 可以經由 postorder(r,0)得到。依照 postorder，圖 3 上的各節點都有一個代表數字，如圖 4。然後把節點 0 的值儲存在陣列第 0 個位置中，節點 1 的值儲存在陣列第 1 個位置中，以此類推。

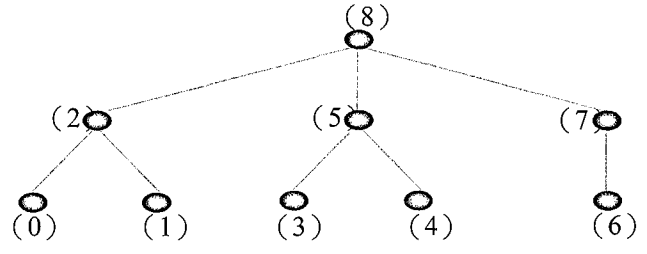


圖 4：經過編號的系統效能資料

因此，監測資料庫可定義為一個序列資料庫，其中每一筆資料有相同的欄位。而當比較兩筆記錄的欄位是否一樣時，就比較除了序列 id 外的所有欄位值。

序列與序列資料庫的正式定義如下：

定義三

1. 一個元素是一筆多屬性的記錄。
2. 一個序列是由一連串同屬性的元素組成。 在本文中，有時以序列資料庫稱之。
3. 一個序列 t 的第 i 個元素為 t[i]。
4. $t_1[i] \subseteq t_2[j]$ 如果 t_1, t_2 為序列且∀a a 為 t_1 屬性->if $t_1[i]$.a != 0 then $t_1[i]$.a = $t_2[j]$.a
5. $t_1[i] = t_2[j]$ 若且為若 t_1, t_2 為序列且 $t_1[i] \subseteq t_2[j]$ 且 $t_1[i] \supseteq t_2[j]$
6. $t_1 = t_2$ 若且為若 t_1, t_2 為序列且$|t_1| = |t_2|$而且∀i $t_1[i] = t_2[i]$
7. $t_1 \in t_2$ 如果 t_1, t_2 為序列且∃k∀i $t_1[i] = t_2[k+i]$

四、將數值資料轉換成區間資料

因系統記錄值極多，若直接做資料挖掘處理，會有大量的計算，產生出來的規則也會過於瑣碎，所以對於數值屬性的處理方法是把數值屬性分類成幾個區間再來計算，分類準則依領域相關知識，由系統管理者定義，例如：CPU 的負載到達 60%以上則系統會嚴重警告，系統以紅燈警示；40% ~ 60%則系統會提出警告，系統以黃燈警示；負載在 40%以下則系統以綠燈顯示，這種情形基本上數值屬性被視為類別屬性。將連續的數值分別對映到各個區間裡，再將分割後的區間對映到連續的整數，這樣一來便可以大大減少所要處理的複雜度。

假設根據表 4 第一筆資料所獲得的效能樹中，陣列第 0 個元素 30 代表 CPU 使用率為 30%，第 1 個元素 20 代表 CPU 平均 load 為 20，餘類推。假設相對表 4 的效能樹中非葉節點的計算公式如下:

CPU	CPU 使用率	CPU 平均 load
3(綠燈)	< 40%	< 15
2(黃燈)	< 60%	< 40
1(紅燈)	o/w (otherwise)	o/w

SWAP	SWAP 剩餘空間	SWAP 使用率
3(綠燈)	<20	< 12
2(黃燈)	<50	< 18
1(紅燈)	o/w	o/w

Database	SQL 錯誤率
3(綠燈)	< 6
2(黃燈)	< 10
1(紅燈)	o/w

父節點值=min(各子節點)
根節點 ES = min(CPU, SWAP, Database)

則表 4 中每筆資料都可轉為如表 5 的效能向量。表 6 中的日期與時間欄位也依先後順序在表 5 中轉成序列 id。

表 5：原始資料轉換成效能向量

序列 id	系統效能狀況
1	[3, 2, 2, 2, 2, 2, 3, 3, 2]
2	[3, 2, 2, 2, 2, 2, 3, 3, 2]
3	[2, 2, 2, 2, 2, 2, 2, 2, 2]
4	[3, 3, 3, 3, 2, 2, 3, 3, 2]
5	[3, 3, 3, 2, 2, 2, 2, 2, 2]
6	[3, 3, 3, 2, 2, 2, 3, 3, 2]
7	[3, 3, 3, 2, 2, 2, 3, 3, 2]
. . .	. . .
..	[2, 2, 2, 2, 1, 1, 2, 2, 1]
..	[3, 3, 3, 2, 3, 2, 2, 2, 2]

肆、演算法

有些時候，系統管理者可能只針對某些幾個影響系統效能的因素，想要了解它們之間的關聯，為了能讓系統管理者可對感興趣的階層進行挖掘，我們發展的 SPT 演算法可針對樹狀結構中的某些子樹（subtrees）或任意一個節點進行挖掘。

一、興趣度向量

一個完整的企業系統所要追蹤的效能節點可多達上百個，在找尋關聯規則時，可能大部分都非管理者所關心的，因此為了能讓系統管理者可針對感興趣的階層或節點進行挖掘，我們以興趣度向量來表示系統管理者感興趣的階層或節點。興趣度向量[b_1 b_2 · · · b_n]，其中 b_1 代表樹狀結構的 ES 效能資料中編號為 0 的節點，b_2 代表編號為 1 的節點，如此類推，影響 ES 效能的因子共有 n 個。如果這個節點對系統管理者而言是有趣的，則向量中表示這節點位置的值為 1，否則為 0。假設圖 5 為 ES 效能資料，例如系統管理者有興趣的是 CPU 和 Memory 兩個子樹之間的關聯（圖 5 陰影部份），興趣度向量為[1 1 1 1 1 1 0 0 0]；又如果系統管理有興趣的只是 CPU 和 Memory 這兩個節點，則興趣度向量為[0 0 1 0 0 1 0 0 0]。根據興趣向量，系統管理者可表達對任意一個或多個節點的興趣。

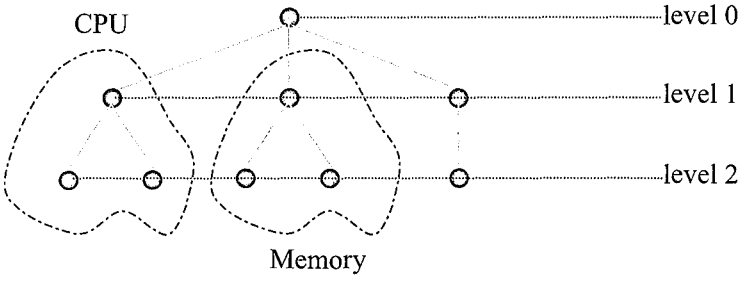


圖 5：使用者感興趣的階層

定義四 若 B 為一個效能向量,A,的興趣向量則

1) $|A| = |B|$
2) $\forall$ b in B, b = 0 or b =1

定義五 給定 n+1 維效能向量,B, 與興趣向量, A, 其內積, *,

定義 $B * A = [b_0 * a_0..b_n * a_n]$。

此 B * A 即為用已從事關聯規則挖掘的資料。

二、SPT 演算法

SPT 演算法共分為三個主要的步驟。「Step 1」針對系統管理者有興趣的階層，利用興趣向量過濾掉不重要的因子;然後將值相同的紀錄進行合併,並計算其支持度(support);依據使用者所設定的門檻值，刪除掉未通過門檻值的紀錄,「Step 2」以 Apriori 演算法為基礎，找出最長的序列樣式,「Step 3」利用上一步驟的找到的序列樣式，產生關聯規則。以提供系統管理者作參考。整個 SPT 演算法如表 6。

傳統方法計算某一序列支持度是以此序列在資料庫中有多少筆一樣的序列為計算準則。但在 SPT 中經過興趣向量轉換後，某些屬性質可能為 0，因此支持度應計算資料庫中有多少序列含有此經過轉換的向量。

定義六 序列 t_1 在序列資料庫, T, 中的 SPT 支持度 = $|\{t \mid t_1 \subseteq t \text{ and } t \in T\}|$

表 6：SPT 演算法

Algorithm SPT
Input:
T：紀錄資料的集合
IV：興趣度向量
Threshold：使用者定義的支持度門檻值
Minconf：使用者定義的信度門檻值
Output:
關聯規則
// Step 1
CSS1 = {}

```
For Each t ∈ T and |t| = 1
{
   css1 += t * IV    //針對感興趣的階層或節點進行挖掘
}
將值相同的進行合併，並累計其 support
LSS1 = { css1 | css1 in CSS1 and support of css1 >= Threshold }
// Step 2
i = 2
Terminate = FALSE
 While (Terminate = FALSE)
{
   CSSi=LSSi-1 join LSSi-1
   if (CSSi == 空集合)
      Terminate = TRUE
}

LSSi = { cssi | cssi in CSSi and support >= Threshold }
if (LSSi == 空集合)
   Terminate = TRUE
// Step 3
For all lssi-1 In LSSi-1
{
   Conf=(support of lssi-1) / (support of lssi-2)
   if (Conf >= Minconf)
      產生關聯規則
}
```

Step 1 過濾掉不重要的因子

以表 7 的資料為例說明，系統效能狀況欄中的數值向量為特定時間點系統效能的狀況，假設系統管理者有興趣的階層為 level 1 中的 CPU 和 Swap 這兩個子樹（圖 6 中之陰影部份），興趣度向量 IV=[1 1 1 1 1 1 0 0 0]。首先，取得表示系統效能狀況的數值向量[2 1 2 3 2 3 2 2 3]，針對系統管理者感興趣的階層把興趣度向量與數值向量相乘（[2 1 2 3 2 3 2 2 3] * [1 1 1 1 1 1 0 0 0] = [2 1 2 3 2 3 0 0 0]）過濾掉不重要的因子（Database 子樹及表示整個系統的根節點），當資料庫中每筆紀錄經過過濾後，可以值相同的紀錄合併為一個，累計其支持度（support），例如：資料庫中[2 1 2 3 2 3 2 2 3]和[2 1 2 3 2 3 1 2 2]，經過濾不重要因子後數值向量皆為[2 1 2 3 2 3 0 0 0]，有兩筆相同的紀錄存在，可合併之並累計其支持度，最終的計算結果如表 8；從表 9 我們可得到 CSS_1(Candidate Set of Sequence 1）長度為一的候選序列集合，為了方便後面的敘述，我們在每一筆資料前面加上英文大寫字母以區別不同值的資料，假設使用者設定的門檻值為 10，LSS_1 是由 CSS_1 中通過門檻值的序列集合所構成，LSS_1 的結果列在表 10。

表 7：ES 系統效能經過前置處理後的資料內容

RecordID	系統效能狀況
001	[2 1 2 3 2 3 2 2 3]
002	[3 3 3 2 3 3 5 5 5]
003	[4 2 4 4 2 5 3 4 4]
. . .	. . .
015	[1 3 3 1 4 4 1 2 4]
016	[2 1 2 3 2 3 1 2 2]
017	[3 3 3 4 3 4 5 5 5]
018	[4 2 4 4 2 5 3 4 4]
. . .	. . .
026	[4 2 4 3 2 3 3 4 4]
027	[3 3 3 4 3 4 5 5 5]
028	[3 3 4 4 3 4 2 3 5]
029	[3 3 4 4 3 4 2 3 5]
. . .	. . .

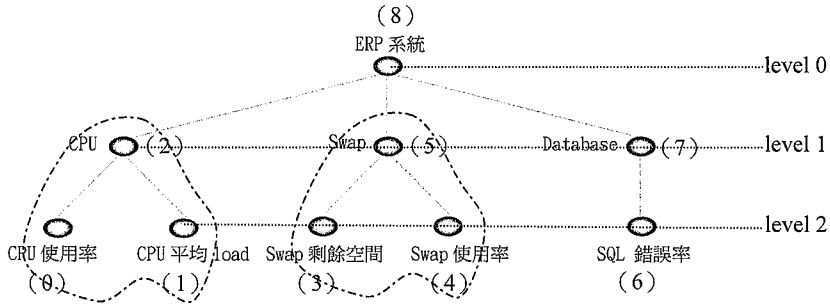


圖 6：表 7 中系統效能狀況的樹狀結構

表 8：過濾不重要因子，合併相同值的支持度

系統效能狀況	支持度
[2 1 2 3 2 3 0 0 0]	213
[3 3 3 2 3 3 0 0 0]	120
[4 2 4 4 2 5 0 0 0]	60
. . .	. . .
[1 3 3 1 4 4 0 0 0]	40
[1 1 1 4 2 4 0 0 0]	40
[2 1 2 3 2 5 0 0 0]	6
[5 1 5 4 2 4 0 0 0]	9
[1 2 3 2 2 5 0 0 0]	35
. . .	. . .

表 9：CSS_1

系統效能狀況	支持度
A [2 1 2 3 2 3 0 0 0]	213
B [3 3 3 2 3 3 0 0 0]	120
C [4 2 4 4 2 5 0 0 0]	60
D [1 3 3 1 4 4 0 0 0]	40
E [1 1 1 4 2 4 0 0 0]	40
F [2 1 2 3 2 5 0 0 0]	6
G [5 1 5 4 2 4 0 0 0]	9
H [1 2 3 2 2 5 0 0 0]	35
. . .	. . .

表 10：LSS_1

系統效能狀況	支持度
A [2 1 2 3 2 3 0 0 0]	213
B [3 3 3 2 3 3 0 0 0]	120
C [4 2 4 4 2 5 0 0 0]	60
D [1 3 3 1 4 4 0 0 0]	40
E [1 1 1 4 2 4 0 0 0]	40
H [1 2 3 2 2 5 0 0 0]	35
. . .	. . .

Step 2 找出符合門檻值的最長集合序列

在 Step 1 最後所產生的表 10，已將使用者不感興趣的屬性去掉，且其支持度也符合給定的門檻值，這時我們將產生下一回的候選序列。要產生下一回的候選序列 CSS_k 先由這回的 LSS_{k-1} join LSS_{k-1}，而 join 的條件是這兩個序列須是可相銜接的序列。亦即，把

兩序列中一個序列去掉第一個元素後的子序列等於另一個序列去掉最後一個元素的子序列，這與 Agrawal & Srikant (1995)使用相同的 join 方法。定義如下：

insert into CSS_k
select p[1] p[2]..p[k-1]q[k]
from LSS_{k-1} p，LSS_{k-1} q
where *p[2]* = q[1]..*p[k]* = *q[k-1]*

以表 10 的 A 為例，它可以和 A 或下一個集合序列 B 進行 join，其結果就如表 11 的第一筆 AA 和 AB 兩個序列集合，而它們的支持度是檢視資料庫中符合這樣屬性值的筆數所累加的結果。當產生 CSS_2產生之後，接下來是將不符合門檻值的集合序列去除，進而產生 LSS_2。其結果列在表 12。CSS_3是 LSS_2 join LSS_2後產生，以表 12 的 AB 和 BH 為例，可以產生 ABH 的候選序列集合，其結果列在表 13。依此方法，演算法一直進行，以尋找最長的集合序列，直至無法產生候選集合序列或大集合序列（Large Set of Sequence），整個 Step 2 才告終止。

表 11：CSS_2

系統效能狀況	支持度
A [2 1 2 3 2 3 0 0 0]	40
A [2 1 2 3 2 3 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	20
B [3 3 3 2 3 3 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	4
C [4 2 4 4 2 5 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	6
D [1 3 3 1 4 4 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	60
B [3 3 3 2 3 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	30
H [1 2 3 2 2 5 0 0 0]	
. . .	. . .

表 12：LSS_2

系統效能狀況	支持度
A [2 1 2 3 2 3 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	40
A [2 1 2 3 2 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	22
B [3 3 3 2 3 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	60
B [3 3 3 2 3 3 0 0 0]	
H [1 2 3 2 2 5 0 0 0]	25
. . .	. . .

表 13：CSS_3

系統效能狀況	支持度
A [2 1 2 3 2 3 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	40
A [2 1 2 3 2 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	7
B [3 3 3 2 3 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	21
B [3 3 3 2 3 3 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	16
H [1 2 3 2 2 5 0 0 0]	
. . .	. . .

表 14：LSS_3

系統效能狀況	支持度
A [2 1 2 3 2 3 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	
A [2 1 2 3 2 3 0 0 0]	40

B [3 3 3 2 3 3 0 0 0]	21
B [3 3 3 2 3 3 0 0 0]	
B [3 3 3 2 3 3 0 0 0]	

A [2 1 2 3 2 3 0 0 0]	16
B [3 3 3 2 3 3 0 0 0]	
H [1 2 3 2 2 5 0 0 0]	

Step 3 **產生關聯規則**

關聯規則可以從 Step 2 所找出的 Large Set of Sequence 取得。其規則型式為 X ⇒ Y，X 和 Y 分別是序列集合，但 X 與 Y 間具有序列關係。而規則成立的條件必須為（1）X+Y 需達到門檻值的標準，（2）規則本身的信度需達到最小信度（minimum confidence）。

我們以表 14 為例，假設表中的 LSS_3 是通過門檻值的 CSS_3 所得到長度為 3 的集合序列，例如集合序列 ABH，其可產生的規則為 AB ⇒ H，所以 Confidence（AB ⇒ H）= SUPPORT（ABH）/ SUPPORT（AB）中 SUPPORT（ABH）的計算可以從表 12 中 AB 的支持度取得，因我們可算得規則 AB ⇒ H 的信度為（16 / 22）= 72.3 %。假設使用者設定的最小信度為 55%，那麼我們可以這樣解釋：當 CPU 和 Memory 這兩個影響系統效能的因子，其狀況連續出現 AB 的情形時，它接著出現 H 的情況的機率為 72.3%，也就是說當 CPU 和 Memory 的情況連續出現下面這種情形（時間順序由先至後為 A, B）：

A [2 1 2 3 2 3 0 0 0]
B [3 3 3 2 3 3 0 0 0]
它會發生下面情況的機率為 72.3%
H [1 2 3 2 2 5 0 0 0]

運用興趣度向量，可以產生不同型式的規則，這些規則可以是描述整個系統效能變化的狀況，也可以是某個子系統或個別的因子，視乎系統管理者感興趣的程度。藉著效能的變化情形用來預測其未來可能會發生的情況。

伍、系統實作

發展出 STP 演算法之後，為了驗證 STP 演算法的可行性，特以 ERP 系統領導廠商 SAP R/3 的系統效能資料為例作為模擬。

首先，撰寫 SAP R/3 的 ABAP 程式，持續收集 SAP R/3 在一個月每小時的效能監控資料，再把 R/3 系統的效能監控資料儲存到外部資料庫中。在應用 STP 演算法挖掘前，資料需轉換成適當的格式，因此，系統一開始便要求管理者根據系統中既定的準則，設定警報訊號的門檻值，進行前置處理的工作；然後再針對系統管理者有興趣的階層，過濾掉不重要的因子；接著找出長度為 1 的大集合序列,再找出長度最長的大集合序列，最後產生關聯規則。

圖 7 以樹狀結構呈現出 R/3 系統環境中被監控的所有元件的整體架構，系統管理者以點選的方式，設定各個元件觸發警報訊號的門檻值，當所有的門檻值都設定完畢，即可進行前置處理的工作。

前置處理的程序會把 R/3 系統中具有多屬性特徵的原始資料（圖 8），轉換成單屬性資料，目前 R/3 系統的警報訊號有紅、黃、綠三個類別值，我們分別以 1 代表紅燈；2 代表黃燈；3 代表綠燈來進行資料的轉換。前置處理後，原始資料轉換成圖 9 的形式。

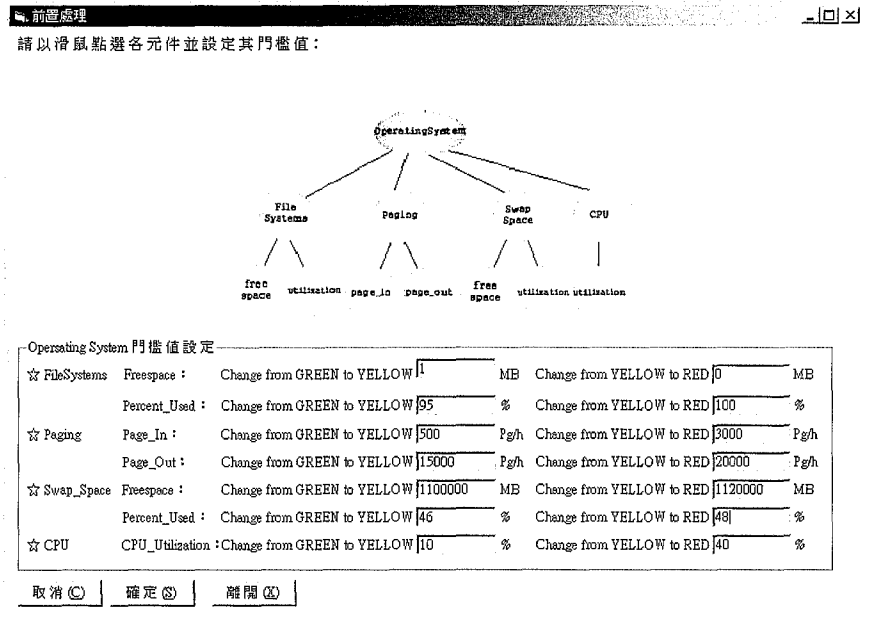
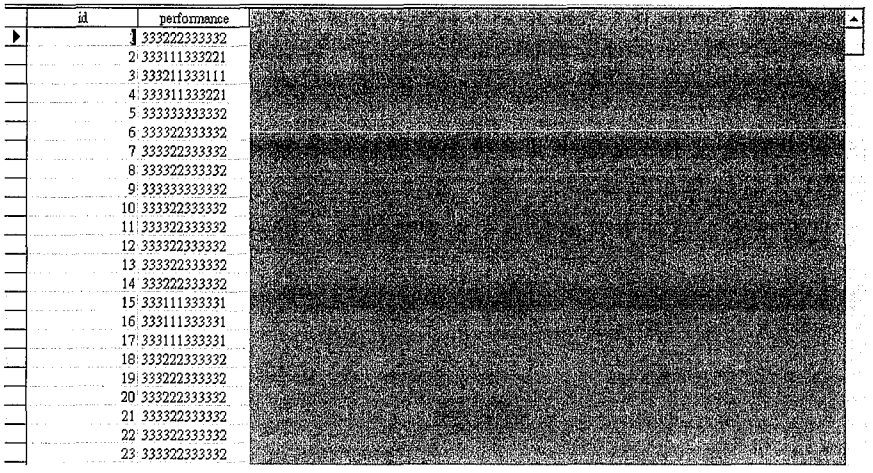


圖 7：設定系統各元件的門檻值之畫面

ID	日期	時間	file_free	file_used	page_in	page_out	swap_free	swap_used	cpu
1	20010610	0	245	26	777	18903	1133720	45.9396052521153	2
2	20010610	1	245	26	7138	21614	1133717	45.9397483043541	18
3	20010610	2	245	26	1234	25462	1133699	45.9406066177873	53
4	20010610	3	245	26	32	20051	1133721	45.9395575680356	39
5	20010610	4	245	26	152	1888	1133728	45.9392237794783	2
6	20010610	5	245	26	272	15619	1133728	45.9392237794783	2
7	20010610	6	245	26	249	18922	1133728	45.9392237794783	2
8	20010610	7	245	26	251	18654	1133728	45.9392237794783	2
9	20010610	8	245	26	136	1883	1133728	45.9392237794783	2
10	20010610	9	245	26	139	17686	1133728	45.9392237794783	2
11	20010610	10	245	26	118	17334	1133728	45.9392237794783	2
12	20010610	11	245	26	134	15818	1133728	45.9392237794783	2
13	20010610	12	245	26	101	15431	1133728	45.9392237794783	2
14	20010610	13	245	26	1869	16404	1133728	45.9392237794783	2
15	20010610	14	245	26	3758	20218	1133723	45.9394621998764	2
16	20010610	15	245	26	3897	21939	1133720	45.9396052521153	2
17	20010610	16	245	26	22672	22094	1133655	45.9427047172906	3
18	20010610	17	245	26	1583	18202	1133600	45.9453273416698	2
19	20010610	18	245	26	1462	17517	1133603	45.9451842894309	2
20	20010610	19	245	26	2864	17971	1133599	45.9453750257494	2
21	20010610	20	245	26	411	15601	1133592	45.9457088143068	2
22	20010610	21	245	26	75	19245	1133592	45.9457088143068	2
23	20010610	22	245	26	172	17599	1133592	45.9457088143068	2
24	20010610	23	245	26	173	17587	1133592	45.9457088143068	2
25	20010611	0	245	26	791	20324	1132119	46.0159474635884	2

圖 8：R/3 系統原始資料



id	performance
1	333222333332
2	333111333221
3	333211333111
4	333311333221
5	333333333332
6	333322333332
7	333322333332
8	333322333332
9	333333333332
10	333322333332
11	333322333332
12	333322333332
13	333322333332
14	333222333332
15	333111333331
16	333111333331
17	333111333331
18	333222333332
19	333222333332
20	333222333332
21	333322333332
22	333322333332
23	333322333332

圖 9：前置處理後之 R/3 系統效能資料

前置處理完畢便開始進行資料挖掘的程序，可針對管理者感興趣的節點進行挖掘，例如管理者想要針對 CPU 這個節點和整個 paging 子節點進行挖掘（圖 10）。另外，可以由管理者自行輸入門檻值和最小信度（圖 10），系統則依據輸入的門檻值及最小信度產生出序列樣式及關聯規則。

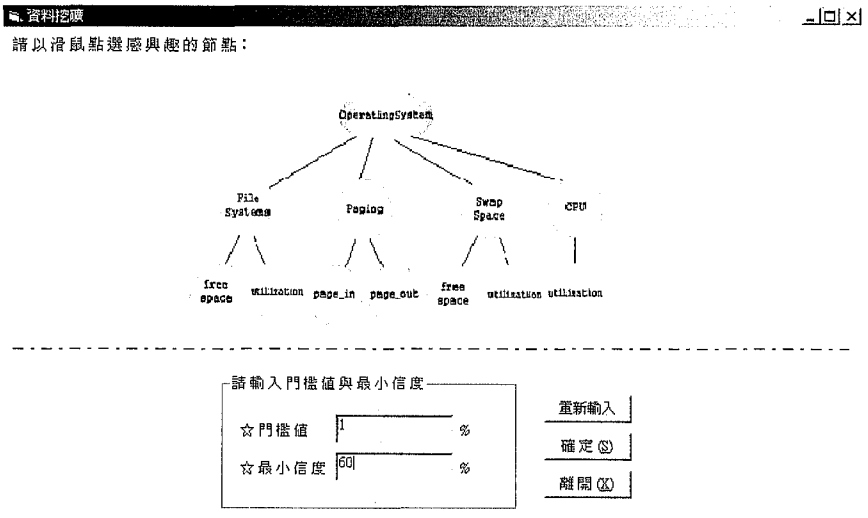


圖 10：資料挖掘前的設定之畫面

圖 11 顯示出找出的序列樣式，在此找出長度為 4 的大集合序列，而其支持度為 6、信度為 87%，符合使用者所設定的門檻值（1%），產生符合管理者設定的信度的關聯規則。

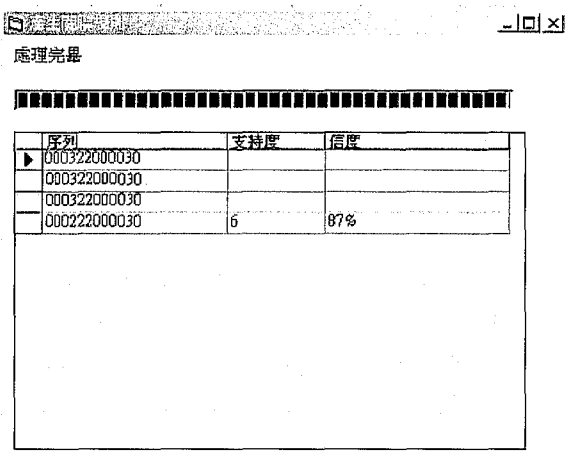


圖 11：產生關聯規則之畫面

從圖 11 我們可以得知產生一關聯規則，其規則為連續三小時發生的 Page In 警報訊號為綠色、Page Out 警報訊號為黃色和 CPU 警報訊號為綠色的情況時，會有 87%的機率在第四小時 Page In 會被觸發成黃色的警報訊號。以上例而言，系統管理者可推論 Page In 之黃色警訊可能來自 Page Out 之黃色警訊，如果管理者能夠解除 Page Out 之黃色警訊，則能解除 Page In 之黃色警訊，因此，管理者得此關聯規則，將不再視系統元件之狀態異常為單一事件之發生，而更能綜觀系統元件間相互之影響。

大型企業系統產生大量的系統監控資料，常令系統管理者無所是從，更無法窺視出各元件間之關聯，本研究嘗試以資料挖掘的方法，從大量系統效能資料中，發現系統效能轉變時各元件的關聯，找出企業系統在下一時間點極可能發生之狀態，此為一對未來狀態之預測，系統管理者可依其發現之關聯規則得知系統瓶頸可能發生之處而加以調校，事先防範於未然，使其企業系統更加穩定。

陸、結論及未來研究方向

對企業系統而言，效能的維護是非常重要的，因此很多系統都提供監控功能，但系統管理者卻常被淹沒在眾多零亂的資訊中而理不出頭緒。本研究使用資料挖礦法則來協助 MIS 人員發現系統效能變化的規則。

本文主要是結合資料挖礦技術中的序列關聯規則與有根樹資料結構，發展出適合 Tree-Based 系統效能資料特性的演算法－SPT（Sequences of Performance Trees）。針對樹狀序列資料結構，以及數值屬性資料的處理技巧提出的演算法，可針對系統管理者感興趣的階層進行挖掘，找出的關聯規則以幫助系統管理者從大量詳細的系統效能資料中找出潛藏的訊息，同時可以發現觀測現象發生的序列變化，這些資訊可以供系統管理者維持系統的運作、分析和矯正錯誤、以增進系統效能。

由於使用興趣向量，使得論文中計算 support 的方法異於一般方法，而強調可計算兩元素間的子集合關係。亦即，t_2 包含 t_1 如果 t_1 的所有非 0 屬性值皆出現於 t_2 中。而序列資料庫中每一個包涵 t_1 的子序列皆貢獻 1 個 support 值。

未來的研究方向可分為：

1. TTS 演算法需多次掃描資料庫，花費較多時間，未來應該思考如何改進這方面的問題，以提高效率。
2. 將連續數值屬性資料轉換為區間資料時，可應用模糊集合理論，以降低因接近區間的分界值資料而產生的影響。
3. 在挖掘同一 level 的關聯規則時，從二個 frequent (k-1)-itemsets 產生 k-itemsets candidate 時，若可依照樹狀資料的特性，設定不同的支持度限制條件，應可更符合現實的狀態。

影響系統效能的因素非常多，在本研究當中是以樹狀結構來表示，實際資料值儲存在陣列中，為了減少挖掘時對資料處理的速度和複雜度，運用了數值屬性資料的處理技術，雖然可以有效地改善計算支持度時的效率，但若能加以壓縮編碼，相信效率會更好。另外，不是每個因素都有絕對的影響，因此我們可以選擇最重要的因素作為評估。

參考文獻

1. 陳仕昇, 陳彥良, 許秉瑜 “在序列式資料中挖掘序列規則”, 資訊管理學報第六卷第二期,2001
2. 陳仕昇, 許秉瑜, 陳彥良 “以可重覆序列挖掘網路瀏覽規則之研究”, 資管評論,1997
3. 陳智宗, 陳振明, 許秉瑜 “以資料挖礦法挖掘多屬性序列式資料規則之研究,” 中大資管所碩士論文, 2000
4. 盧靜婷, 陳彥良, “在 DAG 中挖掘家族特徵規則,” 中大資管所碩士論文, 2000
5. Agrawal, R., Imielinski, T. and Swami, A.N. “Mining association rules between sets of items in large database,” Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993
6. Agrawal, R. and Srikant, R. “Fast Algorithms for Mining Association Rules,” Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile
7. Agrawal, R. and Srikant, R. “Mining Sequential Patterns,” Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan
8. Chen, M.S., Han, J. and Yu, P.S. “An Effective Hash Based Algorithm for Mining Association Rules,” Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995
9. Chen, M.S., Han, J. and Yu, P.S. “Data Mining：An Overview from a Database

Perspective," IEEE Transactions on Knowledge and Data engineering, 8(6), pp. 866-883, 1996.

10. Chen, M.S., Han, J. and Yu, P.S. "Efficient Data Mining for Path Traversal Patterns," IEEE Transactions on Knowledge and Data Engineering, Vol. 0, No. 2, pp. 209-221, 1998
11. Han, J. and Fu, Y. "Discovery of Multiple-Level Association Rules from large databases," IEEE Transactions on knowledge and Data Engineering, 11(5), 1999
12. Han, J. and Kamber, M. Simon Fraser University, Data Mining Concepts and Techniques, 2001
13. Hong, T-P., Lin, K-Y. and Chien, B-C. "Mining Fuzzy Multiple-Level Association Rules from Quantitative Data, "Applied Intelligence, Volume 18, Issue 1, 2003
14. Jain, R. The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling, 1991
15. Larocca, D. Sams Teach Yourself SAP R/3 in 24 Hours, Indianapolis, Ind.: Sams, June 1999
16. Lin, X., Liu, C., Zhang, Y. and Zhou, X. "Efficiently Computing Frequent Tree-Like Topology Patterns in a Web Environment," Technology of Object-Oriented Languages and Systems, 1999. TOOLS 31. Proceedings, 1999, pp:440-447
17. Ling, C.X., Gao, J., Zang, H., Qian, W., and Zhang, H. "Mining Generalized Query Patterns from Web logs," Proceeding of the Thirty-Forth Hawaii International Conference on System Science, HICSS-34, 2001
18. SAP AG, R/3 library - an online library of the entire R/3 documentation, 1999
19. SAP AG, System R/3 Technicale Consultant Training 1 - Administration, Release 4.0A 12-3, December 1998
20. Srikant, R. and Agrawal, R. "Mining Generalization Association Rules,"Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland 1995.
21. Srikant, R. and Agrawal, R. "Mining Quantitative Association Rules in Large Relational Tables,"Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996
22. Verykos V.S., Houstis E.N., and Rice, J.R., "Mining the performance of complex systems," Information Intelligence and Systems,1999. Proceedings. 1999 International Conference on , 1999, pp:606 -612
23. Wang, K., He, Y. and Han, J. "Pushing Support Constraints into Association Rules Mining," IEEE Transactions on knowledge and Data Engineering, 2003,pp:642-657
24. Will, L. SAP R/3 System Administration : The Official SAP Guide, San Francisco : Sybex, 1999
25. Yen, S.J. and Chen, A.L. "An Efficient Approach to Discovery Knowledge from Large Databases,"Proceedings of the IEEE/ACM International Conference on Parallel and

Distributed Information Systems, 1996, pp:8-18

26. Yen, S.J. and Chen, A.L. "A Graph-Based Approach for Discovering Various Types of Association Rules,"IEEE Transactions on knowledge and Data Engineering, 2001