第肆章 雛形系統實作與數據分析

本研究實作了一個雛形系統，其中包括了整個網頁-路徑規則演算的流程。本章就雛形系統各部份做詳細的說明，並且將討論雛形系統所收集的各種效率數據。

4.1 雛形系統的功能介紹

本雛形系統各子系統可依功能區分為記錄處理子系統(P-A Log System)及資料挖掘子系統(P-A Data Mining System)兩大功能項目。記錄處理子系統的目的就是產生如 表 3-6 的 P-A 序列資料庫，並且完成接下來資料挖掘過程所可能需要的資訊。其中，它包括了使用者記錄留存功能、序列轉換功能、處理 support 總數資料功能。資料挖掘子系統的目的就是有效率的找到藏在序列資料庫的規則。其中，它包括了主要的 PAScanX 功能及 Rule 管理功能。各子系統與功能間的關係如 圖 4-1。

以下幾節茲將各子系統與功能，及其需要完成的任務做進一步的說明。

4.1.1 記錄處理子系統(P-A Log System)

本系統主要任務就是要確定記錄資料已經過妥善的處理，到達可以進行 Mining 的程度。有一點要特別提出來的，由於目前本系統並沒有實際的使用者資料可供分析，所以我們在此雛形系統中，除了建立一般使用者記錄留存的機制

之外，我們還多建立了自動生成記錄的系統，來幫助完成資料的取得。此自動機制在對於實作於實體網站上是不需要的。

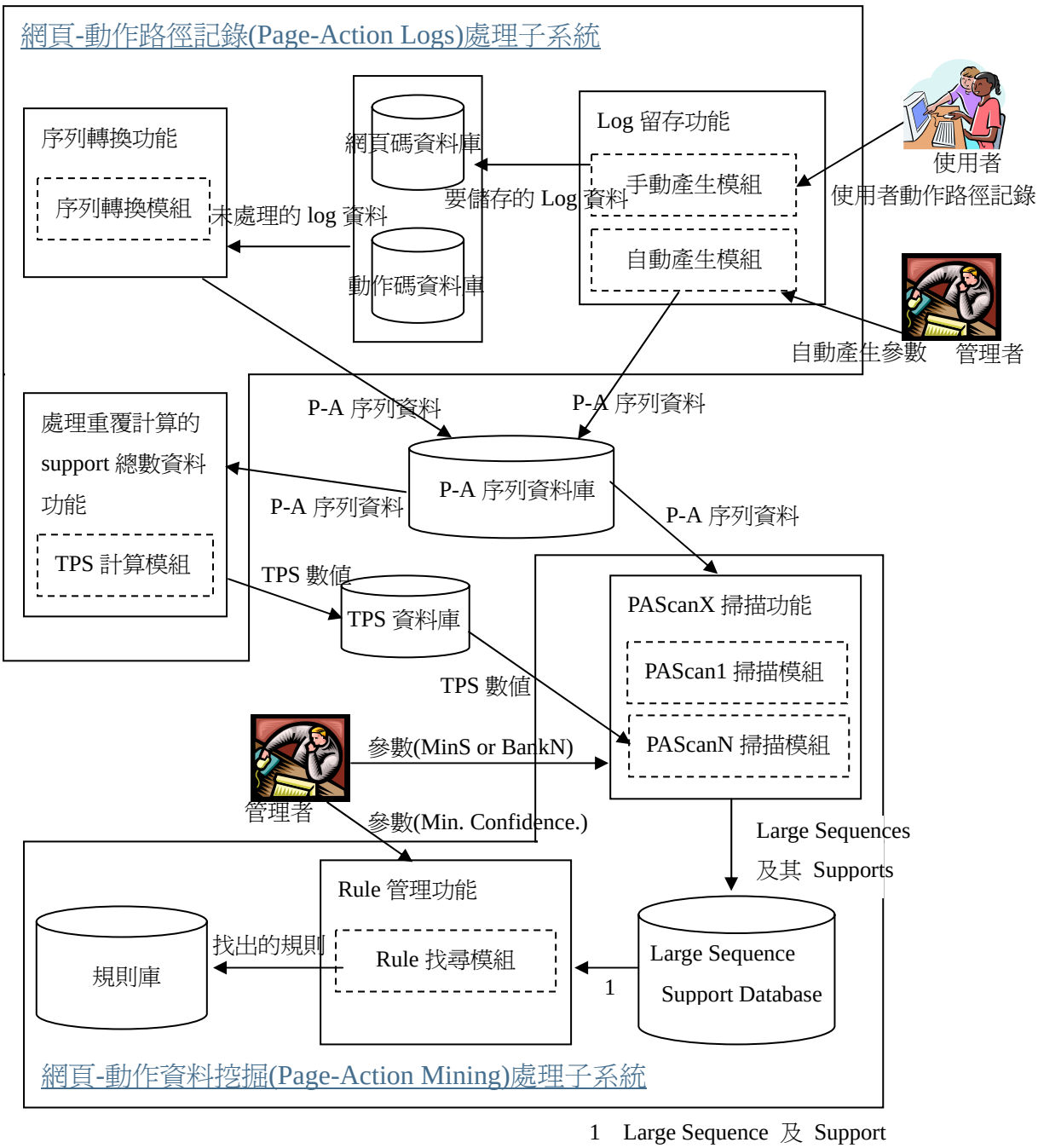

圖 4-1 雛型系統架構圖

以下就記錄處理子系統中的功能做各別的敘述：

一、Log 留存功能：內含二個模組，一個手動記錄處理器，一個自動記錄產生器。手動記錄處理器會忠實記錄使用者轉頁及執行過的功能。為了滿足未來實作的可行性，所以手動記錄器會記錄頁面的執行開始及結束時間，而動作只記錄執行時間。這二類的記錄，都是執行終結就登錄一筆資料，並沒有暫存後再一次寫入的功能。在實作上如果害怕主網站效能容易被 Log 機制所影響，可以採用具暫存機制的方式來留存 Log，這樣就可以減少因為收集 Log 而影響實體經營的可能，但產生不完整資料的可能性，也會隨之提高。手動記錄器的執行頁面如圖 4-2。第二個模組是自動記錄產生器，這個產生器主要是本研究為了實驗效能而製作的，這個模組可能利用亂數功能批次產生符合使用者需要的記錄，這個模組可調整產生序列的平均頁面與動作長度和其最大偏移值，藉由產生大量性質相同或相異的資料。記錄產生的邏輯設計在 4.2 節會有詳細的介紹。自動 Log 產生器的設定畫面如圖 4-3。

圖 4-2 手動記錄器的執行頁面

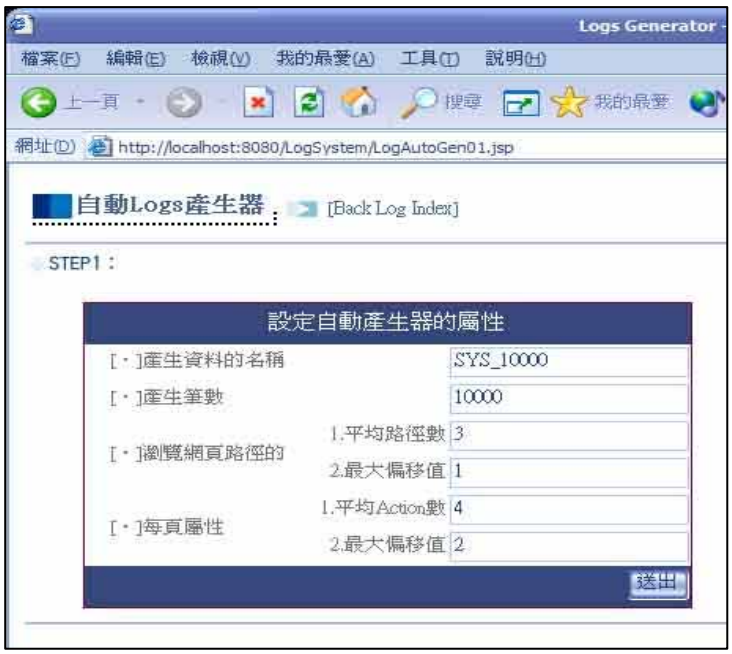

圖 4-3 自動 Log 產生器的設定畫面

二、 序列轉換功能：主要的將手動記錄留存器所留存的 Page 與 Action 記錄轉換成可供 Mining 的 P-A 序列資料。這個轉換器的邏輯做法在 3.1 節中有詳細的說明。轉換完的 P-A 序列資料就可以拿來進行資料挖掘了。當在實際應用及建置系統時，再執行序列轉換功能前可再設計一個資料過濾器。這個過濾器可以將不滿足資料格式、遺失不全或是明顯的錯誤資料先去除，接下來再轉換資料，才能比較確定記錄的正確性。序列轉換功能的執行畫面如圖 4-4。

三、處理重覆計算的 support 總數功能：這個功能主要是為了因應下個階段 PAScanN 掃描所需要的。因為在 PAScanN 演算法中，每一個 Pattern 的可能 Support 總數並不一定，所以每一階的 Pattern 在掃描之後，怎麼確定是否 Large，也無法像 PAScan1 一樣馬上得知，只有得知每一階可能的

support 總數之後，PAScanN 才有可能進行運算。我們稱這些可能的 support 總數為 TPS(Total Possible Supports)，經由計算一個序列資料庫中的所有資料，可以得到每個 Page 長度所有可能的 support 數。對於最大網頁長度為 PV 的序列資料庫來說，他的 TPS 的數量就有 PV 個。圖 4-5 是處理重覆計算的 support 總數功能的執行畫面。

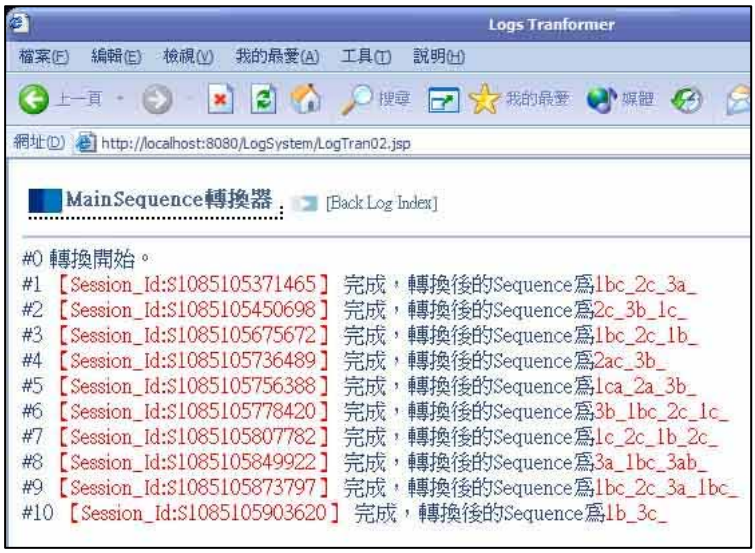

圖 4-4 序列轉換功能的執行畫面

圖 4-5 處理重覆計算的 support 總數功能的執行畫面

4.1.2 網頁-動作資料挖掘子系統(P-A Mining System)

資料挖掘子系統是整個雛形系統的核心。這個子系統的目的是為了找出 P-A 序列資料中所隱藏的規則，其中需先利用 PAScanX 功能找出常出現的序列(Large Sequences)，然後再利用規則尋找功能，對常出現的序列找出之間的規則。

以下就資料挖掘子系統中的功能做各別的敘述：

一、掃描功能：本雛形系統的主要核心，PAScanX 功能包括了完整的 Large Sequences 的計數能力。其中包含本研究所介紹的二種演算法，PAScan1 與 PAScanN 二個模組。PAScan1 模組，可以做到以下六個功能：

1. 初始化第一階 Pattern。

2. 計算各 Pattern 的 Support 數。(一筆資料，不重覆計算 Support)

3. 使用 PA_Join 產生下一階的 Pattern。

4. 可將找到的 Large Sequence 儲存到 Large Sequence 的資料庫中。

5. 可選擇自動掃描到最後一階或是手動一階一階觀察的模式。

6. 自動掃描模式具有記錄執行時間的功能。

圖 4-6、圖 4-7、圖 4-8 是 PA_Scan1 功能的畫面。而 PAScanN 模組，利用 TPS 資料庫的數值則可處理單筆資料出現多次數值的問題，基本上的功能與 PAScan1 類似。PAScanN 的設定畫面如圖 4-9、圖 4-10。

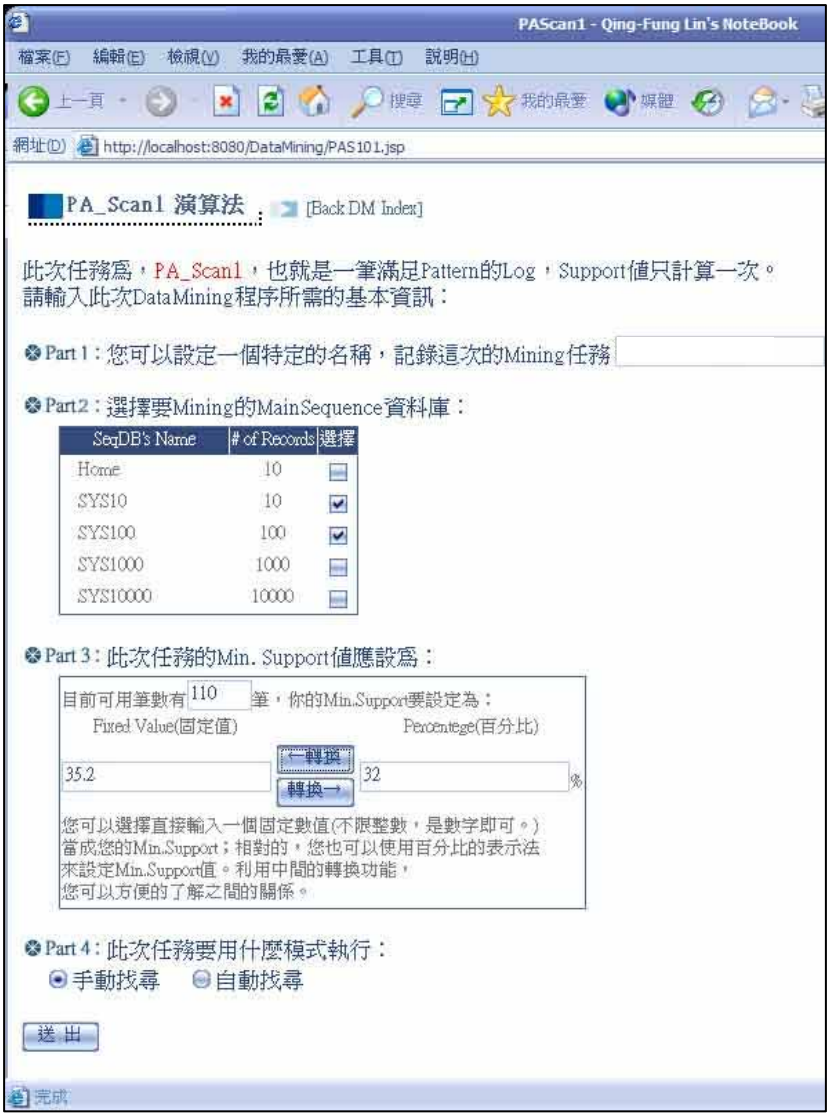

圖 4-6 PAScan1 的設定畫面

圖 4-7 PAScan1 的初始 Scan CS0（手動模式）

圖 4-8 PAScan1 的 Scan CS1（手動模式）

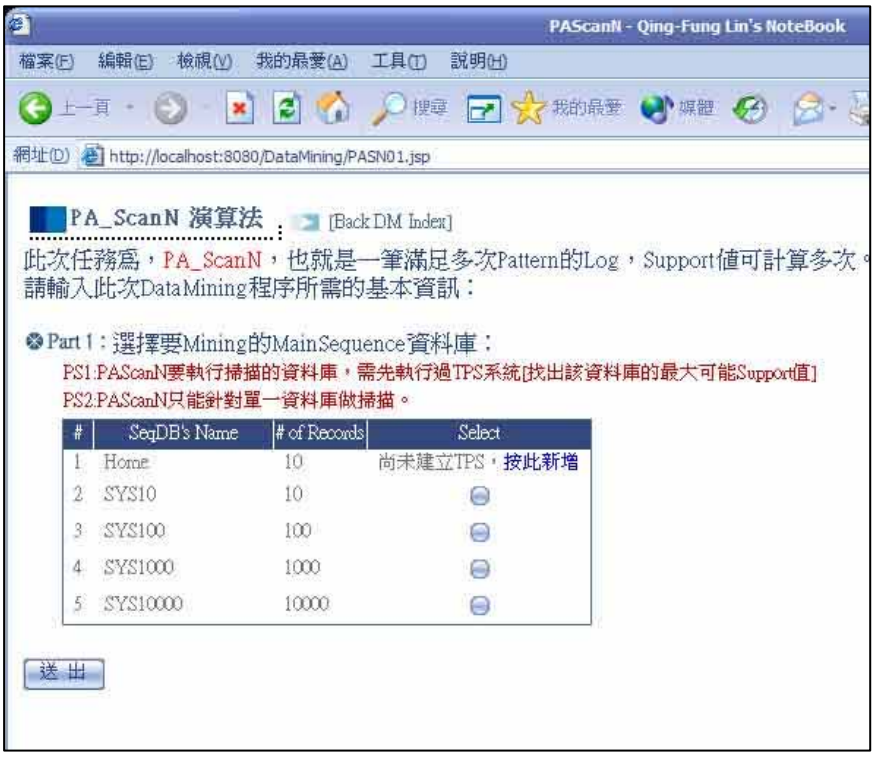

圖 4-9 PAScanN 的設定畫面(一)

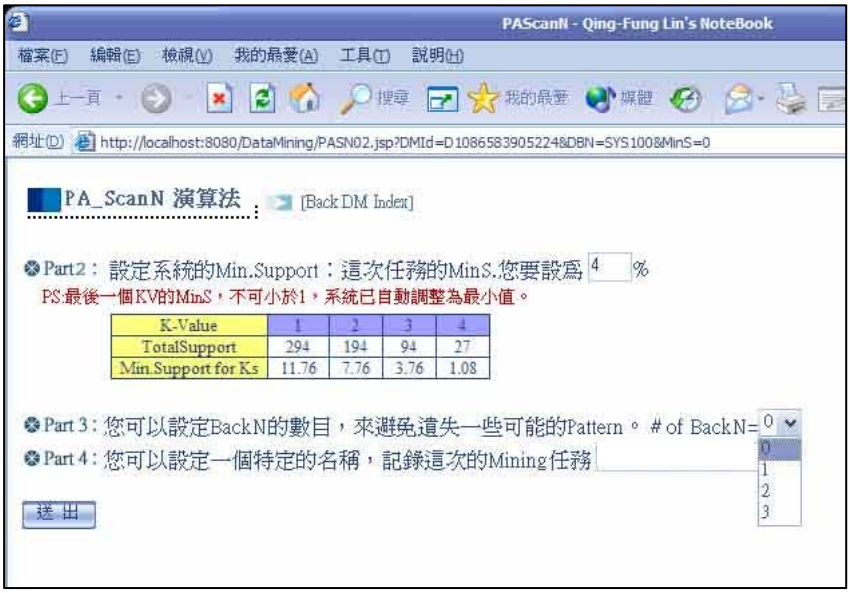

圖 4-10 PAScanN 的設定畫面(二)

二、規則找尋功能：本功能可以依管理者所給與的 Min.Confidence.來找尋所有 Large 的規則並且可將規則做分類。找尋規則的相關畫面如圖 4-11 、圖 4-12 與圖 4-13。

圖 4-11 找尋規則功能的設定畫面

圖 4-12 找尋規則功能的執行畫面

#	LHS	RHS	規則百分比%	Rule Style
1	3a1c	3a1b1c	100.0%	→←中夾型
2	3a1b	3a1b1c	100.0%	→後推型
3	1c2c	1b1c2c	80.0%	←前導型
4	1b2c	1b1c2c	80.0%	→←中夾型
5	1b	1b1c	66.6%	→後推型
6	1b1c	1b1c2c	66.6%	→後推型
7	2c	1b2c	62.5%	←前導型
8	2c	1c2c	62.5%	←前導型
9	1c	1b1c	60.0%	←前導型

圖 4-13 規則功能的呈現畫面

4.2 本雛形系統的執行效能

本雛形系統是採用 Java J2EE 的 MVC 架構建置完成的。進行資料效能測試的機器是採用 Windows XP 的作業系統，CPU 為 AMD Athlon 2400+，主記憶體則是 256MB DDR 133。為了測試這個雛形系統的執行效能，我們還需要再定義如表 4-1 所列的幾個參數，在測試過程中，我們將藉由鎖定某些參數的值來對特定的參數做資料測試。

表 4-1 測試資料有關參數

參數名稱	參數代號	所代表的內涵
資料庫數量	N	資料庫，所擁有資料的總數目。
Min.Support	MinS	本次掃描任務認定如何的資料才算是 Large 的門檻值。
平均網頁路徑長度	AVGPL	平均一筆資料的網頁路徑長度
平均動作路徑長度	AVGAL	平均一個網頁中的動作路徑長度
平均網頁路徑離散值	AVGPVar	資料庫中網頁路徑平均離散值
平均動作路徑離散值	AVGAVar	動作路徑的平均離散值

接下來的模擬與測試過程，我們將針對以下三大項目來討論這個雛形系統的效能：

一、資料庫型式、Support 值相同，但資料量不同，所需的執行時間有什麼不同。

二、資料庫型式、資料量相同，但 Support 值不同，所需的執行時間有什麼不同。

三、Support、資料量相同，但資料庫型式不同，所需的執行時間有什麼不同。

其中，資料型式不同的問題又可以分為，網頁路徑平均長度不同、動作路徑平均長度不同，網頁路徑離散程度不同以及動作路徑離散程度不同四個方向。

為了進行測試，需要產生大量符合各種參數需求的資料。本系統的序列產生器就是基於這個理由所開發出來的。以下是序列產生器的邏輯設計：

```
List    Log Generator(long Mount, int AVGPL, int AVGAL, int MaxPVar, int MaxAVar ) {

//取得五個參數，其中 MaxPVar 是最大網頁路徑偏移量，與 AVGPVar 不同
//                          MaxAVar 是最大動作路徑偏移量，與 AVGAVar 不同
Put {SELECT DISTINCT Page_Id FROM [Page_Code]} into P[];
//將不同的 Page_Code 儲存到 P[]裏。
Long PMax = {SELECT Count(*) FROM [Page_Code]};
Put {SELECT DISTINCT Action_Id FROM [Action_Code]} into A[];
//將不同的 Action_Code 儲存到 A[]裏。
Long AMax = {SELECT Count(*) FROM [Action_Code]};

//Math.random()  會亂數產生一個 0 到 1 之間的小數。服從 0~1 的均一分配。
List Alist;
For (i=1;i<=Mount;i++) { //產生 Mount 筆資料
   String MainSeq = “”;
   Int NowPageLen = AVGPL + int((MaxPVar+1) * Math.random());
   //決定這個序列的網頁路徑長度。
      String LastP = “”;
      For(j=1;j<=NowPageLen;j++){ //產生 NowPageLen 段的網頁路徑
            int PI = int(PMax*Math.random()); //決定要那個頁碼
            If (P[PI]==LastP) { //處理不能重覆二個頁面都是一樣的頁碼
               PI = (PI + (int)(Math.random()*(PageMax-1))+1) % PageMax;
            }
            LastP = P[PI];
            MainSeq = MainSeq + P[PI]; //加入
            Int NowActionLen = AVGAL + int((MaxAVar+1) * Math.random());
            //決定這頁中動作路徑的長度。
            For(k=1;k<= NowActionLen;k++){ //產生 NowActionLen 個的動作
                  MainSeq = MainSeq + A[int(AMax*Math.random())];
                 //決定使用那個動作並加入 MainSeq。
            }
       }
```

```
            Alist.add(MainSeq)
    }
    Return(Alist)
}
```

為了配合以下的模擬測試，我們利用序列產生器產生了如 表 4-2 所列的幾個資料庫來測試各種不同的狀況。

表 4-2 測試資料列表

編號	資料庫代號	N	AVGPL	AVGAL	AVGPVar	AVGAVar
1	SYS01001	10	3	4	0	0
2	SYS01002	100	3	4	0	0
3	SYS01003	1,000	3	4	0	0
4	SYS01004	5,000	3	4	0	0
5	SYS01005	10,000	3	4	0	0
6	SYS01006	30,000	3	4	0	0
7	SYS01007	50,000	3	4	0	0
8	SYS01008	70,000	3	4	0	0
9	SYS01009	90,000	3	4	0	0
10	SYS01010	10,0000	3	4	0	0
11	SYS02001	1,000	3	4	0.69	2
12	SYS03A01	1,000	2	4	0	0
13	SYS03A02	1,000	3	4	0	0
14	SYS03A03	1,000	4	4	0	0
15	SYS03B01	1,000	3	2	0	0
16	SYS03B02	1,000	3	3	0	0
17	SYS03B03	1,000	3	4	0	0
18	SYS03C01	1,000	4	4	0.65	0
19	SYS03C02	1,000	4	4	2	0
20	SYS03C03	1,000	4	4	4	0
21	SYS03D01	1,000	3	4	0	0.68
22	SYS03D02	1,000	3	4	0	2
23	SYS03D03	1,000	3	4	0	4

以下三節我們將分別就這三大項目來討論雛形系統與網頁-動作路徑演算法的效能。在進行模擬測試前，需要再提出來的是，這裏所測量的時間數值，並未包含原始 Log 處理的時間。這些時間數值取得的方式是由使用者觸發進行 Scan 時開始計算，並於系統完成所有回應後停止計算。所有的計時工作，皆是由系統完成。

4.2.1 資料量不同對於網頁-動作路徑演算法的影響

在第一個模擬中，我們使用了編號 1 到 10 號的資料庫，Min. Support 設為 30%，其餘四項資料庫參數，皆為 AVGPL=3、AVGAL=4、AVGPVar=0、AVGAVar=0。圖 4-14 是使用 PAScan_1 演算法所呈現出的效能狀況。我們可以發現，PAScan1 的執行時間與資料總數具有一個線型關係。

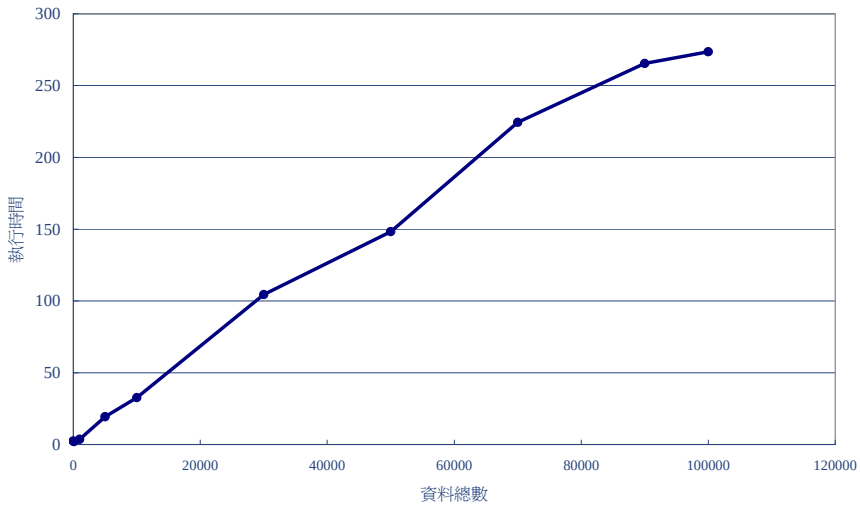

圖 4-14 PAScan1 在不同資料量所需的執行時間

PAScan_N 演算法因其 Next Pass 數目的不同所呈現出的效能狀況也有不同，當我們設定 MinS 為 25%時，掃描編號 1 到 10 的資料庫，我們得到 圖 4-15

的數據，其代表 PAScan_N 在 Next Pass = 0 與 Next Pass = 1 兩種狀況所呈現出的效能狀況。PAScan_N 在不同數量的資料庫中所執行出的效能看來也是呈線性關係的。而「不向後看」，與「向後看一步」這兩種演算方式在這 10 個資料庫的執行效能則是相差四倍左右。

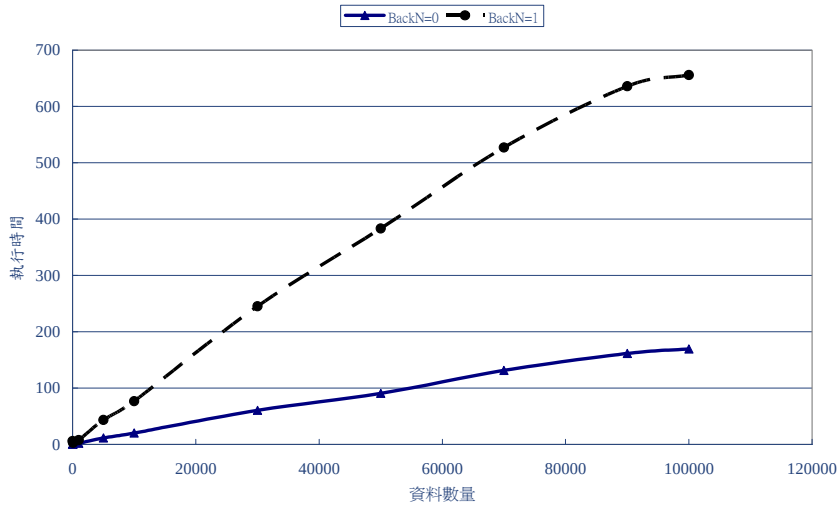

圖 4-15 PAScanN 二種 Next Pass 參數在不同資料量所需的執行時間

4.2.2 Support 數不同對於網頁-動作路徑演算法的影響

針對同一個資料庫，MinS 數值越小，系統執行的效能應該會越慢，執行所需的時間就越多。為了驗証及觀察這個現象，我們在這次的模擬中使用了編號 11 的資料庫，資料量有 1000 筆，平均網頁路徑為 3，離散值 1，平均動作路徑為 4，離散值為 2。執行出來的數據如 圖 4-16 與 圖 4-17。

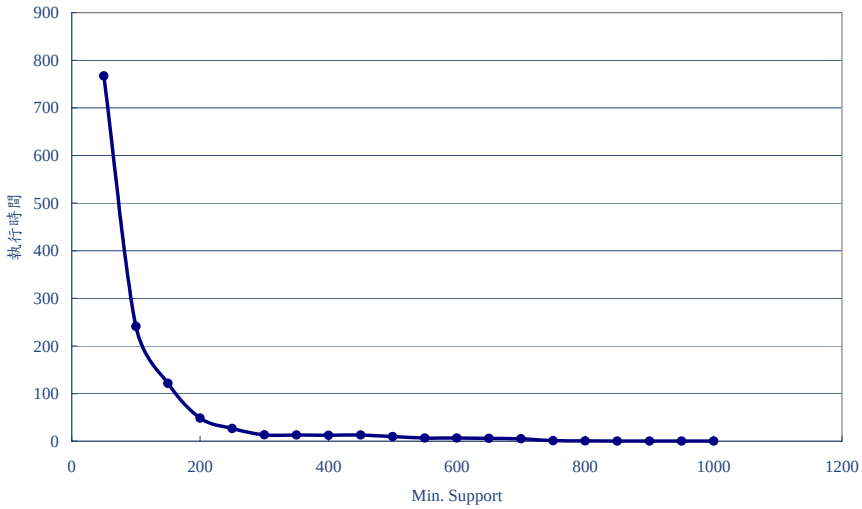

圖 4-16 PAScan1 對同一資料庫在不同 MinS 所需的執行時間

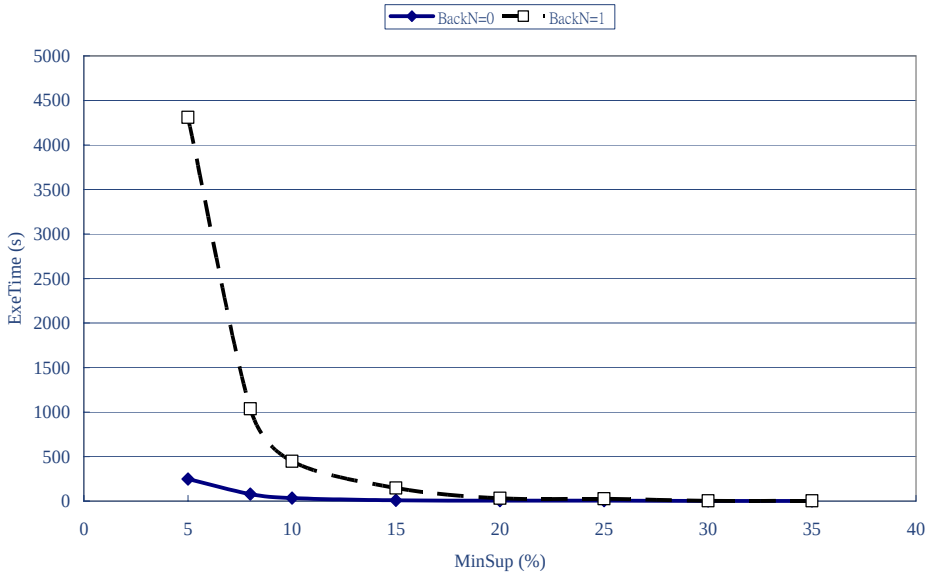

圖 4-17 PAScanN 對同一資料庫在不同 MinS 所需的執行時間

由上面二個圖看起來，不論是 PAScanN 或 PAScan1 系統的效能與 MinS 的大小成反比，而且皆是二次曲線的關係。

4.2.3 資料型態不同對於網頁-動作路徑演算法的影響

一個網頁-動作序列的資料型態可以用四個參數來測量。以下我們將分別對平均網頁路徑、平均動作路徑、平均網頁路徑離散度對平均動作路徑離散度做不同的模擬測試。

首先，我們使用編號第 12、13、14 三個資料庫，進行針對不同平均網頁路徑長度的資料庫做效能測試，這三個資料庫，平均動作路徑皆為 4，二項變異值皆為 0，在 PAScan1 演算法中，我們設定 MinS 為 30%，而 PAScanN 演算法中，我們設定 MinS 為 20%。圖 4-18 與圖 4-19 是測試的結果。我們可以發現，網頁路徑越長，所需要的執行時間就越長。二者之間，有正比的關係。

如果當我們使用編號第 15、16、17 三個資料庫，進行針對不同平均動作路徑長度的資料庫做效能測試，這三個資料庫，平均網頁路徑為 3，二項變異值皆為 0，在 PAScan1 演算法中，我們設定 MinS 為 20%。而 PAScanN 演算法中，MinS 則設為 10%。我們可以得到 圖 4-20 與 圖 4-21 的測試結果。

我們也可以發現，動作路徑平均長度越長，所需要的執行時間也就越長，二者之間，有正比的關係。

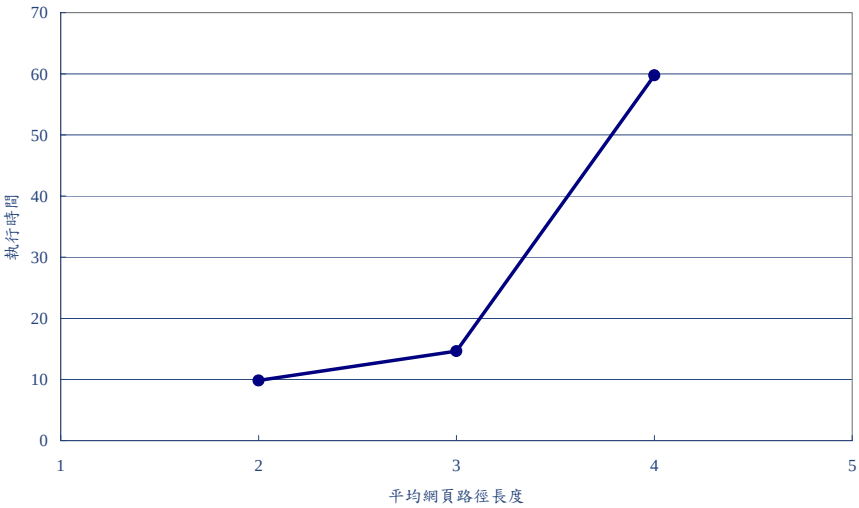

圖 4-18 PAScan1 對不同平均網頁路徑長度所呈現的效能

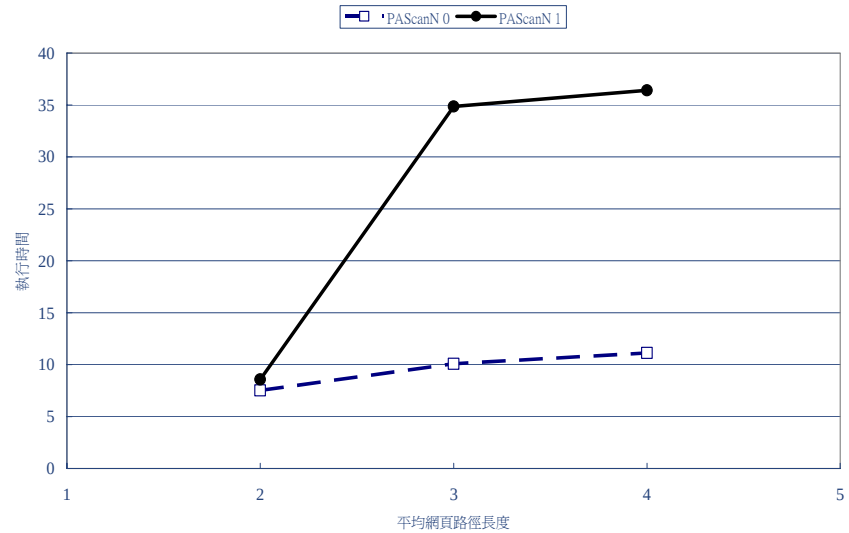

圖 4-19 PAScanN 對不同平均網頁路徑長度所呈現的效能

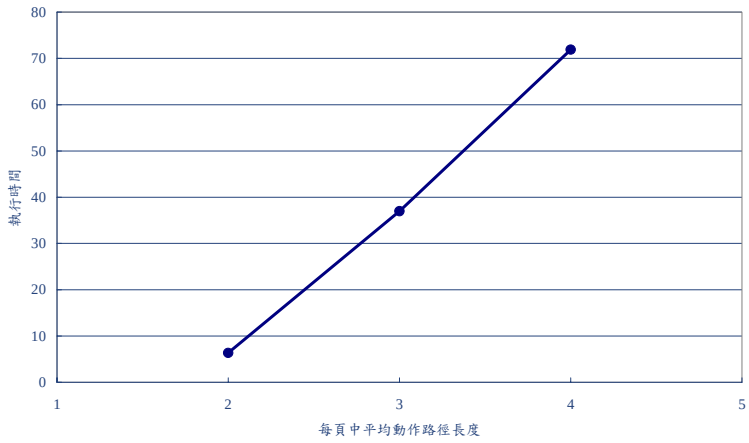

圖 4-20 PAScan1 對不同平均動作路徑長度所呈現的效能

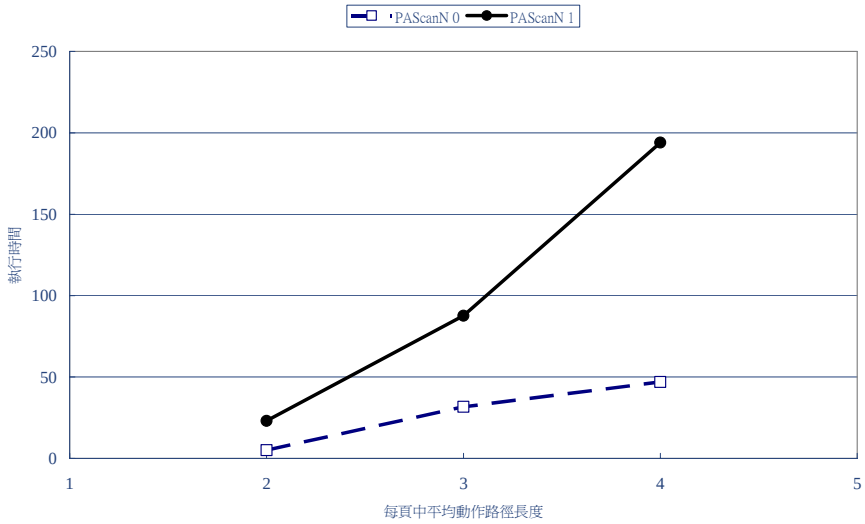

圖 4-21 PAScanN 對不同平均動作路徑長度所呈現的效能

如果針對兩個長度變異數值來做系統測試，我們得使用資料編號 18、19、20 與 21、22、23 兩組的資料庫。針對網頁長度路徑的離散數值所做的測試是在 PAScan1 的 MinS 設為 30%，PAScanN 的 MinS 設為 15%下測試的，我們可以發現如圖 4-22 與圖 4-23 所示不管離散程度如何變化，對於效能並沒有直接的影響。

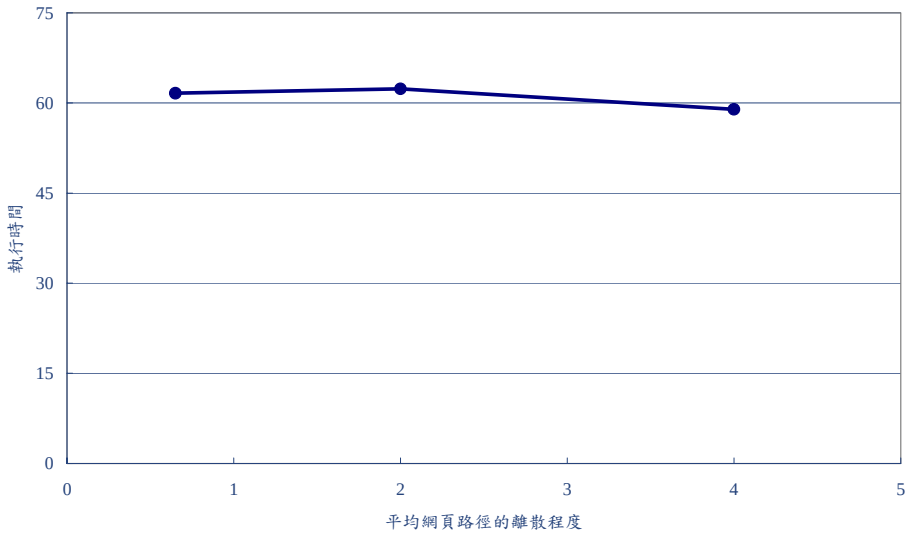

圖 4-22 PAScan1 對不同網頁路徑的離散程度的效能呈現

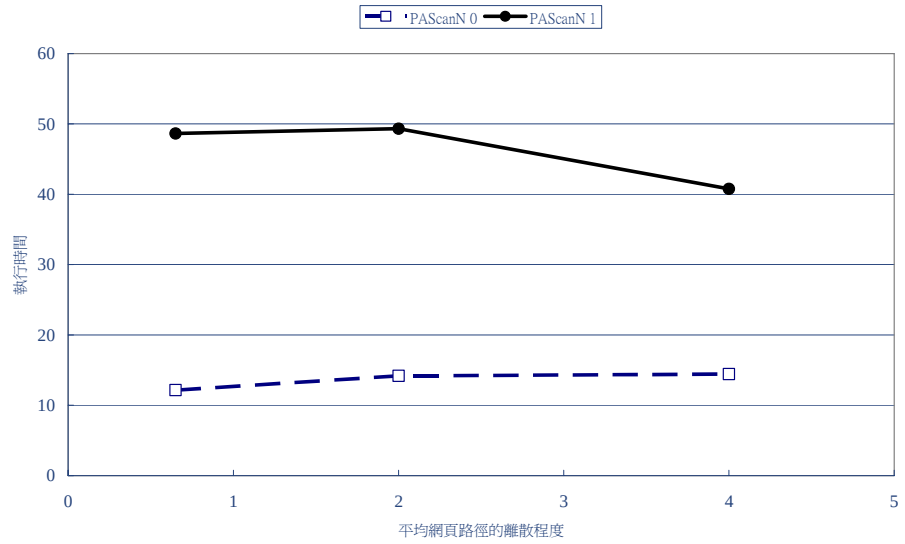

圖 4-23 PAScanN 對不同網頁路徑的離散程度的效能呈現

我們在針對動作長度路徑離散數值所做的模擬測試是將 PAScan1 演算法中的 MinS 定為 30%，PAScanN 的 MinS 定為 15%，我們可以由圖 4-24、圖 4-25 發現結果與上一個模擬的結果類似，動作路徑的離散程度並不明顯影響效能。

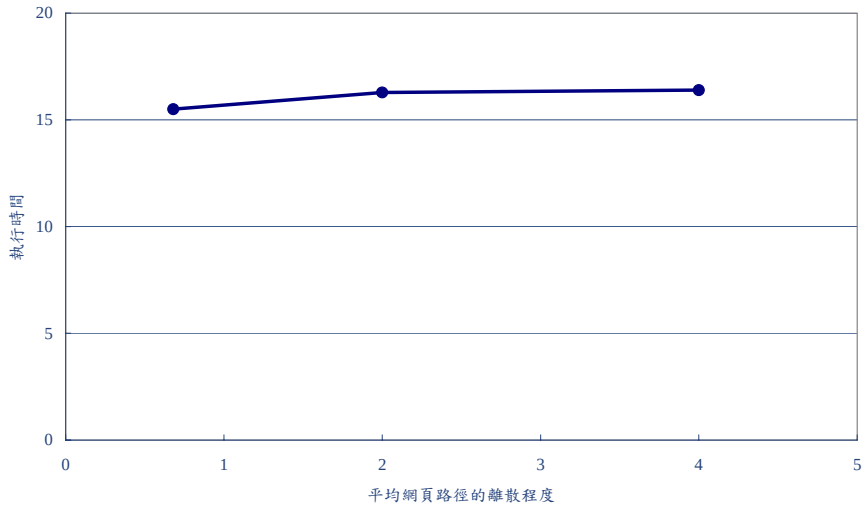

圖 4-24 PAScan1 對不同動作路徑的離散程度的效能呈現

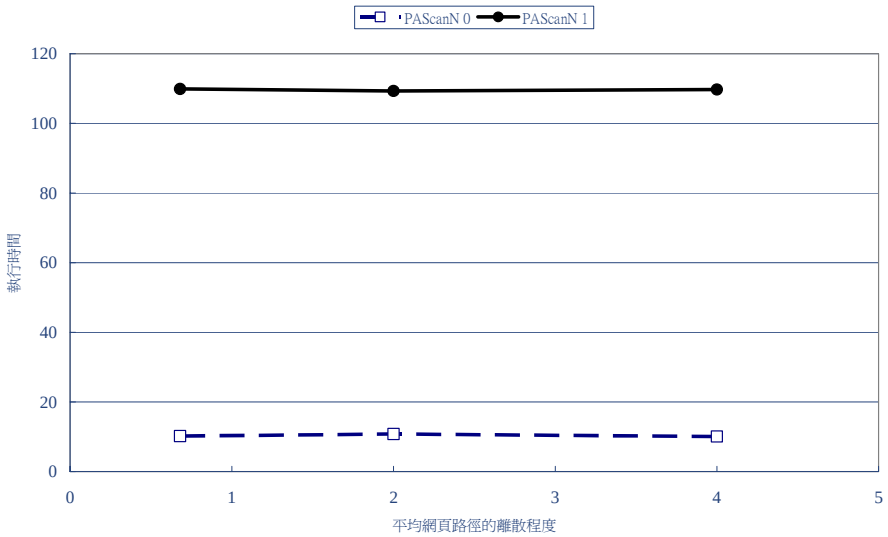

圖 4-25 PAScanN 對不同動作路徑的離散程度的效能呈現