基因演算法運用於分散式網路分級資訊管理之研究

黃國禎、林宗良
暨南國際大學資訊管理系

摘要

隨著網路普及率的提昇及上網人口年齡的下降，世界各國對網站內容分級的重視程度也隨著增加。在推網站內容分級的過程中，除了訂定分級標準之外，最重要的研究主題之一，即是分級伺服器的建置與維護策略。由於分級資訊變化快速，而眾多網路用戶在瀏覽網站內容時又必須隨時查詢分級資訊；因此，如何在大量資料存取的需求下，提供高效率的網站內容分級資訊，將是影響成敗的重要因素之一。過去以單一伺服器建立的分級資訊管理方式，在大量資料存取時，將造成網路的擁塞及長時間的等待；因此，本論文中，我們運用基因演算法來解決分散式分級資訊的規劃及配置問題，以達到提供大量且快速分級資訊服務的目的；同時，並實際建置一套分散式分級伺服器，與原有的 proxy 系統進行整合，以提供分級服務。由大量的資料存取測試的結果發現，本論所文提出的分級資訊管理方式，可以讓網路用戶在不需改變使用習慣的前提，即可獲得有效率的分級服務。

關鍵字： 網路內容分級、PICS、資訊管理、基因演算法

A Genetic Approach to the Management of Website Content Rating Information

Gwo-Jen Hwang、Tsung-Liang Lin

Department of Information Management
National Chi Nan University

Abstract

As computer network becomes popular and the average age of the Internet users decreases, website labeling has become an important and challenging issue. In addition to the development of website content rating standard, it is very important to study the maintenance and management of the website labeling information. As the contents of a website may change frequently, and a tremendous number of users need to obtain the information when browsing web pages, the policy of the content-labeling information management will significantly affect the efficiency of the whole website labeling system. In can be seen that installing the entire website labeling data to a single server will lead to network traffic jam and server overloading. To cope with these problems, in this paper, we propose a distributed model for managing website labeling information. To achieve maximum throughput and optimal load balance for the distributed content rating information, we develop a genetic algorithm to allocate website labeling information for each server. Some experiments on a large amount of test data have shown that the genetic approach can efficiently allocate website labeling information and provide desirable results.

Keywords: web content labeling, PICS, information management, genetic algorithms

壹、緒論

隨著網路普及率的提昇及上網人口年齡的下降，世界各國對網站內容分級的重視程度也隨著增加。在分級的標準制定方面，Resnick 及 Miller.（1996）提出的 PICS 協定已受到國際上的認同。依循 PICS 所規範的分級格式，一個位於美國的獨立、非營利性組織 RSAC（The Recreational Software Advisory Council）發展了一套 RSACi 系統，其目的是希望能夠針對所有的網頁內容，提供一個簡單而有效率的評分方法，讓大眾，尤其是父母，能夠經由一個公開、客觀的評分系統所提供的資訊，來選擇他們所想要的電子媒體，如電子遊戲，全球資訊網站等。而隨著分級觀念的普及與龐大的服務需求，RSAC 近年來改組為 ICRA（Internet Content Rating Association），以網路分級的研究及推動為其主要的目標。ICRA 所提出分級標準可以針對各種網頁做簡單而有效率的評分，以期達到保護兒童，並且維護網路創作者的自由發言權。

在台灣，新聞局參考 ICRA 訂立的分級詞彙及現有的電視節目分級處理辦法，於九十年八月的「推動網路分級實施策略計畫」研究報告中提出分級詞彙草案，將網路內容分級詞彙分為聊天室、語言（Language）、性與裸露（Nudity and Sex）、其他議題（Other topics）及暴力（Violence）等類別。在每個類別中，除了將 ICRA 編碼與 RSACi 編碼之外，並對照台灣電視節目分級處理辦法分為限制級、輔導級、保護級、普遍級等四個等級，同時加上解釋及範例以增加詞彙的接受度。同時，電信總局於九十年度網路分級實施辦法計劃中，提出在實施網站分級制度的資訊蒐集架構，因應不同時期的需求，整合數個種蒐集分級資訊的方法，包括第三機構分級、網路使用者合作式分級、網頁作者自行分級、自動化分級及輔助式分級。

在分級資訊蒐集的機制建立之後，即可著手建立分級伺服器端及用戶端程式。目前市面的分級過濾系統，大多將分級資料裝設在用戶端，並且必須另外加裝過濾程式，對使用者而言在操作上及系統效率上都會造成負擔，尤其在分級資料的更新及維護上更是不易，分級設定也容易遭到竄改。因此，建立高效率且集中管理的分級伺服器（Label Bureau）是目前解決這些問題的重要方式之一。

在瀏覽資料時，使用者的瀏覽器會先到分級伺服器查詢該網站的分級資料，並依照預先的設定過濾掉不合適的網站內容。目前這個方式已在國際上被廣泛接受，然而，在長期推廣網站內容分級的過程中，仍有下列幾個問題必須解決：

1. 單一分級伺服器在處理大量資料時將因系統龐大的負擔，而造成時間上的延遲
2. 大量的查詢集中在單一分級伺服器時，將造成網路的擁塞

因此，在本論文中，我們運用基因演算法來解決分散式分級資訊的規劃及配置問題（Yuwono & Lee 1996；Cheung et al. 1996），以達到提供大量且快速分級資訊服務的目的；同時，並實際建置一套分散式分級伺服器，與原有的 proxy 系統進行整合，以提供有效率分級服務。

貳、網路分級資訊配置問題

在網路分級的研究中，以 W3C（World Wide Web Consortium） 所提出的 PICS（Platform for Internet Content Selection）協定最受重視。PICS 協定中詳細地規定網路分級的檢索方式及其標籤語法,其採用的分級檢索方式可分為兩種：一是由網路文件的作者（Content provider）自行定義該網頁內容所屬的級別；另一種方式則是由一特定的伺服器，依據一特定的標準，來對網路上的文件做分級的動作，並且提供分級查詢的服務。用戶在要求連線時，瀏覽器會先至伺服器中比對該網站是否符合預設的級別。此兩種方式詳述如下：

(一) 由網路文件作者自行定義分級資訊：

一般說來，網路文件的作者對於本身所撰寫的文件所屬的性質，應當相當瞭解。因此 PICS 允許作者可在其網頁（目前主要是 HTML 格式）的 meta-data 中，加入符合 PICS 語法的分級資訊。所以在使用者端進行文件的瀏覽時，會一併下載其分級資料，透過分級程式的過濾，便可決定該使用者的網頁瀏覽權。

(二) 由特定的伺服器來提供分級服務：

在 PICS 規劃的此類分級方式中主要又細分出兩種不同的機制。其中一種是由提供 HTML 存取功能的 HTTP 伺服器來附加 PICS 文件分級的服務。在此種方式中，首先須透過各種方式獲得網站內容的分級資訊（Hwang, Chu & Yang 2002），在遠端使用者要求瀏覽網頁內容時，由此一 HTTP 伺服器附加性地傳送出分級資訊。另一種方式則是由一台專門提供分類查詢的伺服器（此類伺服器稱為 label bureau）來處理分類資料；當使用者瀏覽 Internet 上的網頁時，由使用者端的分級程式根據瀏覽網頁的 URL 位置來對 label bureau 提出分級查詢的要求，而由使用者端的分級程式根據伺服器的回應來決定此一網頁的分級類別。

由於在推廣網路分級的概念初期，不易要求所有網頁作者自行分級，因此由特定的伺服器來提供分級服務是較為可行的方式。網站分級資料的建置類似 DNS（Domain Name Server） 的樹狀架構，如圖 1 所示：第一層為 TLD（Top-Level Domain），由國別及網站類別構成，例如 tw、jp、…、zw 等，網站類別則包括 com、net、…、org 等。第二層我們稱為 level 2 domain，可看成是 TLD URL 族群下的小族群；例如在圖 1 的第二層中，可以看到在 TLD 的 tw 族群下又細分為 com.tw 與 edu.tw 兩個族群，依此類推再細分出 level 3 domain 及更細的族群。使用者只要給予網址，即可查詢網路分級資訊。

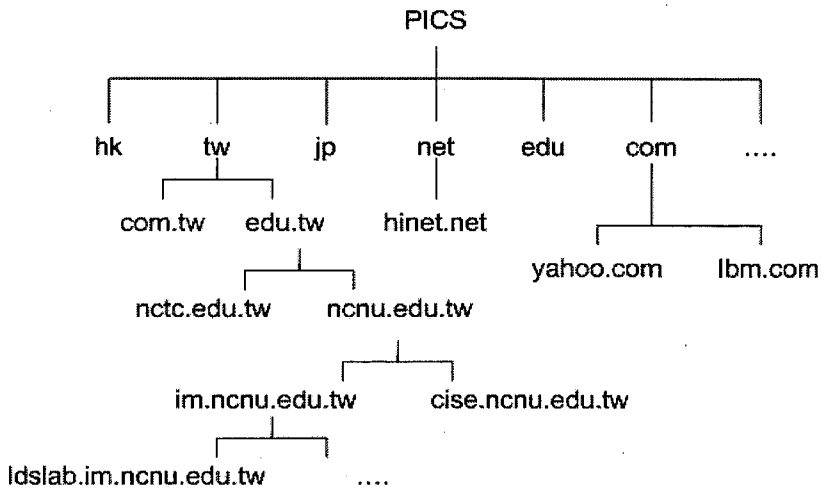

圖 1：分級資料結構樹

然而 ISC（Internet Software Consortium） 在 2002 七月的 Internet Domain Survey 中指出，目前總共有 162,128,493 個 Domain names；因此，面對龐大的分級資料，若以單一伺服器處理分級服務，在負載上將造成問題。因此，以分散式的資料處理方式將是提高伺服器處理效率的考量之一。一般分散式系統主要的設計考量是要獲得最佳的系統處理時間，亦即整體系統的反應時間要最短。在分散式系統的研究架構中，通常必須考慮多個程式及資料檔在多部電腦的配置方式，同時在程式執行時若所需資料不在同一部電腦中則必須透過網路讀取資料。在這些條件下，如何分配程式及資料檔至多部電腦，以達到最佳效能，稱為檔案配置問題（File Allocation Problems – FAP） （Chipperfield & Fleming 1996；Yener & Rose 1997； Zomaya & Wright 2002；Goldberg 1989； Parsaei 1997）。過去已有許多型態的 FAP 被提出，其差異主要是最佳化目標的不同以及問題限制條件的不同（Chen and Akoka 1980; Chang et al. 2001; Chen et al. 1995; Chu 1973; El-Abd 1995; Kurose and Simha 1989; Mahmoud and Riordon 1976; Morgan and Levin 1977）。

為了在未來推動網路分級機制的過程中不影響用戶原先的使用習慣，並避免明顯的影響網路資訊的存取效率，交通部電信總局、中華電信及 SeedNet 等業者基於現況的考量，經過多次的協商之後，獲得以下的決議（曾憲雄、翁秋平、馬宏燦、黃國禎， 民 91 年）：

1. 將台灣分為數個區域，結合現有的網路服務設施進行推廣。各區域的分級過濾機制（Regional Filter Center）獨立運用，各自擁有分級資料庫。而分級資料庫由中央分級資訊管理單位（LDAP Center）統籌進行網站內容分級的審核及資料庫的維護，再定期複製到各區(如圖 2 所示)。分級標準的訂定及相關配套措施由新聞局負責，包括成立分級委員會，負責網站內容分級審核及申訴事宜(曾憲雄、林世華、黃國禎，民 90 年)；而交通部電信總局負責提供協助 ICP（Internet Content Provider）自行分級的軟體，並輔導 ISP（Internet Service Provider）業者建立分級伺服器。

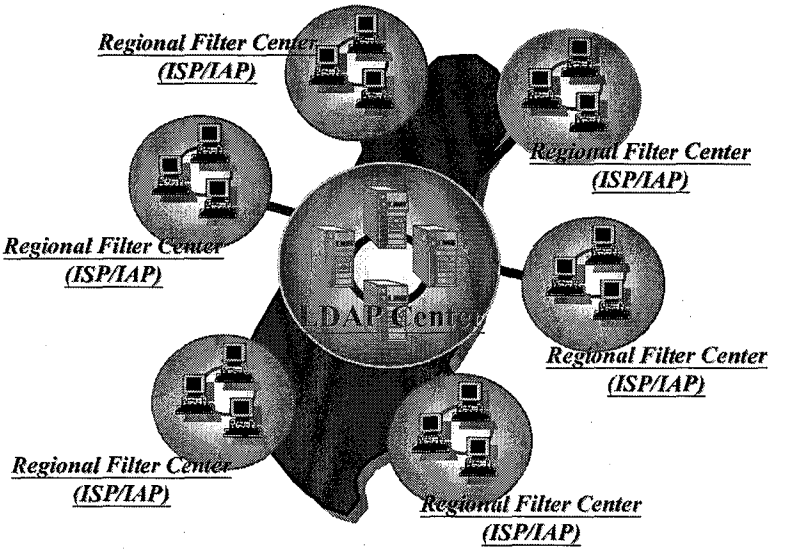

圖 2：網路分級服務的實施架構

2. 各區域的分級伺服器必須與原有的 Proxy 伺服器結合，其分級資料庫應視需要進行分割，以促進執行效率。網站分級的資料應以平均負載的考量，分布到各個主機上，以避免部分主機負擔過重，影響查詢的速度。此外，分級伺服器啟動時，將流量前 1000 大的網站分級資料放在暫存器中，以增進網頁讀取的效率。

基於業界規劃的架構，我們針對各區域的分級資料庫進行規劃。由目前 259 個 TLD 的分配顯示，大部分的網址均集中在少數類別中。表 1 中的 17 個類別即佔了全部 domain names 的 88%，而表 1 中的.net 及.com 類別的 domain names 即佔了 17 個類別的 60%；因此，若直接以結構樹的建置方式來儲存網站分級資料，由於結構樹本身的不平衡，將使得平均查詢效率降低。因此，我們針對結構樹進行適當的分割與重組，將較大的一群如.net、.com 等分割為.net1、.net2、.....、.net9 及.com1、.com2、.....、.com6，另外也將過小的 URL 族群合併，讓結構樹更趨於平衡。

表 1：前 17 大 TLD 比率分配

Domain (TLD)	Hosts	%
net	61945611	36%
com	40555072	24%
jp	9260117	5%
edu	7459219	4%
arpa	6387463	4%
it	3864315	2%
ca	2993982	2%
de	2891407	2%

uk	2583753	1%
au	2564339	1%
nl	2415286	1%
br	2237527	1%
tw	2170233	1%
fr	2157628	1%
mil	1880903	1%
us	1735734	1%
es	1694601	1%
Total	171638297	100%

分級資料分派到不同伺服器的問題定義方式，可以用集合的角度來看分派後的結果，也可以用排序後切割的方式來表示。不論是那一種表達方式，資料筆數及資料存取頻率均為決定分派結果的主要參數，必須在兩者間取捨以達到伺服器的負載平衡。若由排序切割的方式來看，其問題模式與 Grimaldi.（1998）的 Discrete and Combinatorial Mathematics 一書中所提到 combinations with repetition 中的 doughnut problems 問題相似，不同點是 doughnut problems 的目標是切割出不同的字元，而本研究則切割出最佳化的平均負載。分級資料分配問題若模擬 doughnut problems 的求解方式來進行分群，可能解的數目（Solution Space）為 C(n + m - 1, m) = C(n + m - 1, n - 1)。又由於分級資料量 n 是非常龐大的，並且隨著分級資料量的增加，所需使用伺服器 m 亦跟著加大，可能解的數目將跟著指數成長，欲求其最佳解必需花費很長的時間。針對這個問題，我們提出以基因演算法來求取近似最佳解的方式。首先將問題依 Degraeve 及 Vandebroek（1998）的 mixed integer program 模式進行定義：假設有 n 筆依網址排序的網站分級資料，$W_1,W_2,\ldots,W_n$，要分配到 m 部分級伺服器 $S_1,S_2,\ldots,S_m$ 中，其中必須考量到不同時段 $T_1,T_2,\ldots,T_{n2}$ 時，伺服器間的負載是否平均。不同時段可依每小時，或是早中晚等方式來製定，依據個別需求考量，進行合適的分段。經由與業者代表溝通的結果，我們將一天中網路的使用時段分為 3 個時段（但未來可能隨著網路環境的改變調整），所定義的分級資料分配方程式如下：

Decision variables : x_j 為分級資料的段落切割點。分級資料 1 至 x_1 代表伺服器 S_1 分配之段落；x_1+1 至 x_2 代表伺服器 S_2 分配之段落，依此類推。

Coefficient acc_f_{ik} : 網站 $W_1,W_2,\ldots,W_n$ 在時段 $T_1,T_2,\ldots,T_{n2}$ 的個別存取頻率 (access frequency)。

Objective function:

$$\text{Minimize } \alpha\sum_{j=1}^{m}\sum_{k=1}^{n2}\left|SL_{jk}-ASL_k\right|+\beta\left(\sum_{j=1}^{m}\sum_{k=1}^{n2}SL_{jk}-BTL\right)$$

Subject to

$$x_j \le x_{j2} \quad ,\mathrm{j} < \mathrm{j2} \qquad (1)$$

$$x_j = [0,1,\cdots,\mathrm{n}], \quad j=1,2,\ldots,m. \quad j2=1,2,\ldots,m.$$

where

$$BTL=(\sum_{i=1}^{n}\sum_{k=1}^{n2} acc\_f_{ik})\times(n/m+1)/2,$$

$$\alpha=1/m \ , \ \beta=2(m-1)/m^2,$$

$$SL_{jk}=(\sum_{i=x_{j-1}+1}^{x_j} acc\_f_{ik})\times((x_j-x_{j-1})+1)/2 \ \text{ for } j=1 \text{ to } m \text{ and } k=1 \text{ to } n_2, \text{ and}$$

$$ASL_k=\sum_{j=1}^{m} S_{jk}/m \ \text{ for } k=1 \text{ to } n_2.$$

在此方程式中，限制式 (1) 代表伺服器 $S_1,S_2,\ldots,S_m$ 的切割段落必須相互銜接且不重複；參數 SL_{jk} 代表各個伺服器 $S_1,S_2,\ldots,S_m$ 在不同時段 $T_1,T_2,\ldots,T_{n2}$ 時的負載值；ASL_k 代表在不同時段時，伺服器的平均負載。當求出一組解時，可以透過 $\sum_{j=1}^{m}\sum_{k=1}^{n2}|SL_{jk}-ASL_k|$ 來了解伺服器間負載的差距。*BTL* (basic total load)代表的是一個基本的 NAW 總負載，透過 $\sum_{j=1}^{m}\sum_{k=1}^{n2}S_{jk}-BTL$ 可計算出總負載的好壞，其值越小越好。目標函數前半部乘上 $\alpha=1/m$ 代表平均一個伺服器的負載與平均負載相差的程度；後半部乘上 $\beta=2(m-1)/m^2$ 則代表當 $\sum_{j=1}^{m}\sum_{k=1}^{n2}S_{jk}-BTL$ 負載改進一單位時，在最樂觀的情況下，大致可相當於 $\sum_{j=1}^{m}\sum_{k=1}^{n2}|SL_{jk}-ASL_k|$ 負載差距下降 $2(m-1)/m$ 單位，此觀念可以用以下的例子來說明：

	S1	S2	S3	S4	S5	→	S1'	S2'	S3'	S4'	S5'
伺服器負載	2	0	0	0	0		1	0	0	0	0
網站的數量	1	0	0	0	0		1	0	0	0	0
			(a)			←			(b)		

在(a)情況時總伺服器負載(2×1+0×0+0×0+0×0+0×0+0×0)=2，而個別伺服器的負載與平均負載(2/5)的總差距為 2×(2×(5-1)/5) = 16/5。若由情況(a)變動為情況(b)，則在總伺服器負載(1×1+0×0+0×0+0×0+0×0+0×0)=1，比(a)下降 1 的情況下，(b)個別伺服器的負載與平均負載(1/5)的總差距變為 1×(2×(5-1)/5) = 8/5。故在此樂觀情況下，可求得總伺服器負載下降 1 單位，相當於個別伺服器的負載與平均負載的總差距下降 $2(m-1)/m$ 個單位。

參、基因演算法

基因演算法是 Holland 在 1975 年所提出的，嘗試以嚴密而具體的科學方法來解釋自然界中「物競天擇，適者生存」的演化過程，並將生物界中基因演化的機制於資訊世界以軟體實作模擬。基因演算法的精神，即是「物競天擇，適者生存」概念的延伸，亦即選擇物種中對環境適應力較強的母代（Parents）並隨機性的相互交換彼此的基因資訊，以期能產生較上一母代更優秀的子代（Offspring），經過篩選後留下適應力最佳的物種，再繼續交配、繁衍再篩選，如此重覆終而演化出一對外在環境適應力最強之物種。故其最基本的精神在於演化（Evolution）與篩選（Selection）。基因演算法利用三種基本的運作機制以模仿自然遺傳的過程，包括複製（Reproduction）、交配（Crossover）、突變（Mutation）。透過此三個過程的演化，由母代來產生新的子代，而篩選則是定義一適應函數（Fitness Function）來建構其外在的生存環境，所有的物種皆以其為衡量目標來進行演化。

基因演算法的運作包含四個模組：選擇機制（Selection）、基因運算子（Operators）、基因群族（Population）以及評估方式（Evaluation），茲分述如下。

- 選擇機制：由上一代適應函數所產生的結果，選取其中表現較佳的子代，並淘汰弱勢者。
- 基因運算子：運用基因演算法中的複製、交配與突變機制，衍生新的物種，其中的複製、交配與突變將在之後有較為詳盡的說明。
- 基因群族：分解子代與親代物種以供評估。
- 適應函數：以事先定義好的適應函數，求得每一物種對環境的適應力。

而基因演算法之演化流程如圖3所示，步驟可略述如下：

步驟1：群族初始化：以亂數產生初代族群。

步驟2：解碼與計算適應函數：將族群中各染色體解碼，依據事先所定義之適應函數，計算適合度。

步驟3：複製、交配與突變：以目前所得到親代族群，創造子代族群。

步驟4：取代規則：依適應函數值，來決定物種去留。

步驟5：重覆2~4，直到符合終止條件為止。

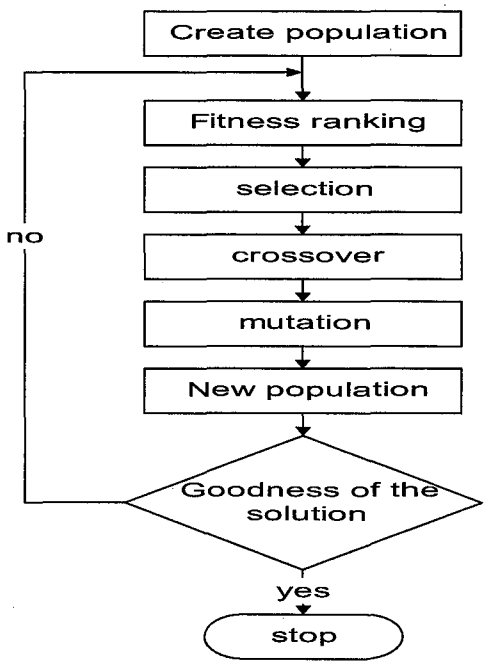

圖 3：基因演算法之演化流程

肆、基因演算法運用於分級資料配置

這個章節描述基因演算法的設計。它包括編碼的描述規劃、初始母體染色體的建立、Fitness 的計算、交換和突變的操作等等。運用基因演算法目的是要找到分級資料配置的策略，以獲得較好的存取效率。

一、Encode Schemes

在基因演算法的使用上，Encoding 方式的選擇，為控制基因演算法成效的一個重要元素。在本篇論文中，基因值以正整數方式來表示，由 1 至 n，其基因串型式如下所示：

$$X=[x_1,x_2,\ldots,x_{m-1}]$$

$$\downarrow$$

[25,55,….,1000]

參數 X 為母體染色體，包括 $m-1$ 個基因 $x_1,x_2,\ldots,x_{m-1}$。並且基因間存在以下幾點特性：

- 不同基因其數值必不相同。
- 基因 xi 必大於基因 xj，當 i 大於 j 的情況下。

二、Seeding 策略

於基因演算法中如果初始母體染色體建立得當，則能夠大大地增進基因演算法的表現，而 Seeding 正是一個建立初始母體染色體的步驟。在本篇論文中，初始母體染色體的數量以參數 S 表示，而初始母體染色體的產生策略則運用平均 Access frequency 和平均網站數量來完成，其方法如下：

- 計算出伺服器所需負載的平均Access frequency，並以Access frequency加總方式求出一組母體染色體的相對應值。例如所有網站的Access frequency加總 $(\sum_{i=1}^{n}\sum_{k=1}^{n2} acc\_f_{ik})$ 為 *total_access_frequency*，則伺服器所需負載的平均 Access frequency 為 *(total_access_frequency/m)*，其中 m 為欲分散的伺服器數量。此組染色體之基因值 $x_1,x_2,\ldots,x_{m-1}$，伺服器所需負載的平均即由第一筆網站的Access frequency $1\times(total\_access\_frequency/1$ 逐漸累加至 $(m-1)\times(total\_access\_frequency/(m-1))$。
- 計算出平均網站數量 n/m，求出第二組母體染色體。此組染色體之基因值 $x_1,x_2,\ldots,x_{m-1}$，分別為 $1\times(n/m),2\times(n/m),\ldots,(m-1)\times(n/m)$。
- 利用問題中基因值 $x_1,x_2,\ldots,x_{m-1}$ 會集中至某一特定範圍的特性，以第一組及第二組母體為基礎，分別運用突變方式針對此兩組母體之基因進行小範圍的變動，各產生一半數量之突變基因串，並以這些基因串為初始母體染色體。

三、Crossover 策略

在本篇論文中 Crossover 使用 one-cut-point method，在所選出的兩染色體字串中，隨機地選取一交配點，並交換兩染色體字串中此交換點後的所有基因位元。如圖 4 所示：

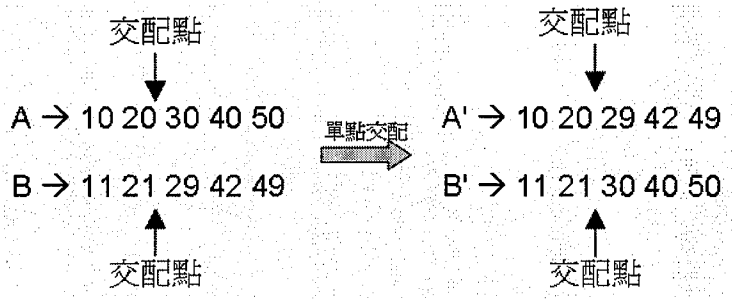

圖 4：One-cut-point method 交配過程示意圖

其中若 Crossover 所得之染色體有不符合 encode 特性者，則隨機產生一數值取代不符合之基因，直到無不符合條件之基因為止。

四、Mutation 策略

基因演算法中突變的意義在於擷取一種不可預測的訊息，以防止物種在一連的複製與交配過程中，侷限在一個區域的最佳化環境中而無法跳出，而找不到全域的最佳解。其過程為隨機地選取一染色體字串，再隨機地選取一突變點，然後改變染色體字串裡的基因位元資料，如圖 5 所示。而突變發生的機率則是由突變率所控制。

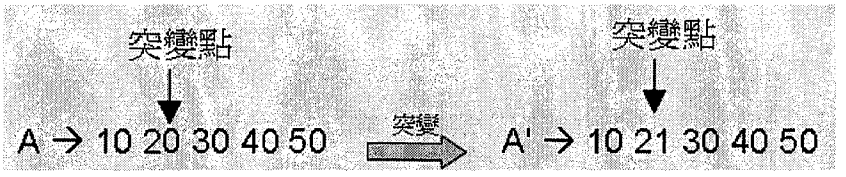

圖 5：突變過程示意圖

除了上述之基本突變型式外，我們在突變過程中加入一名為 *range* 的參數。由於本論文中基因代表意義為切割點，並以負載平衡為訴求而導致基因間互相影響；同時，不同切割區間有不同的突變機率，越靠近原母體之區間其發生機率越高；因此，若突變值與原先基因值差異太大，則該突變的意義並不大，也無法改善結果。因此在特定範圍內的進行突變可讓效果更加顯著，而 *range* 即為此特定範圍之參數。但考慮極端情形，若母體之基因本身偏離特定點，以特定範圍 *range* 之突變則無法適用於此種情形。所以在不捨棄所有可能情況的考量下，本篇論文的突變方式為 90%突變範圍在±*range* 內，其餘 10%則在±min($x_j - x_{j-1}$，$x_{j+1} - x_j$)內。

另外因為各基因間有著相互影響的關係，所以在突變的基因串中，讓其餘基因也有突變的可能性，稱之為互動突變。互動突變點的突變範圍介於原始突變點之正負突變值之間，以 random 方式產生。如圖 6 中所示，若突變點由 20 突變為 22，其突變值為 2，則其餘各點依序以±2 為範圍進行突變。

五、Fitness Function

基因演算法是以適合度函數（fitness function）為回饋的標準，來判斷染色體的優劣，以決定染色體在下一代的存活率。在本論文中，fitness function 的定義如下：

$$\text{Fitness}=\alpha\sum_{j=1}^{m}\sum_{k=1}^{n2}\left|SL_{jk}-ASL_k\right|+\beta\left(\sum_{j=1}^{m}\sum_{k=1}^{n2}SL_{jk}-BTL\right)$$

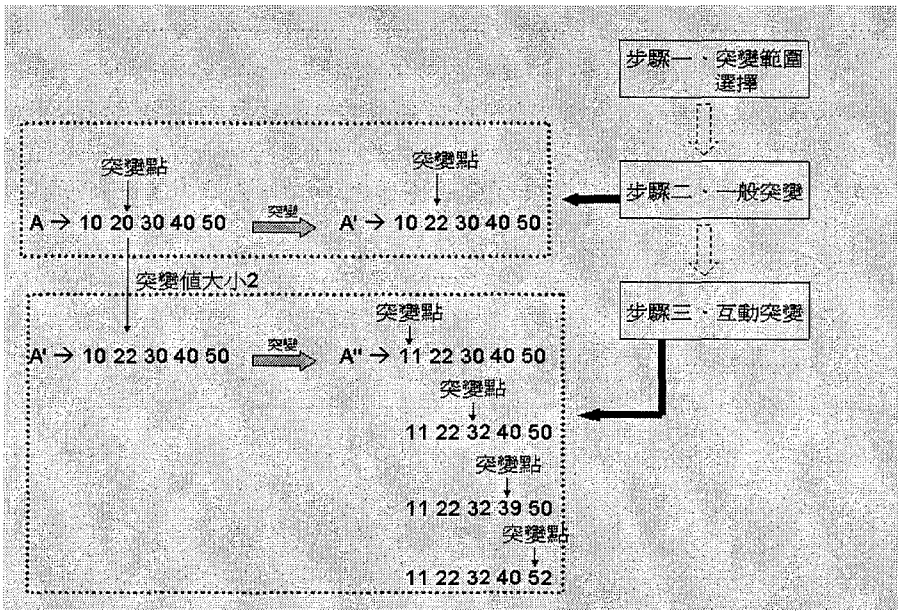

圖 6：互動式突變流程圖

六、Selection 策略

基因演算法中的 Selection 策略是模仿生物界中同形生殖（homogenesis）的行為，即子代與親代間所有的染色體（chromosome）資料完全相同，當然也繼承親代的一切屬性。Selection 的目的在於保留適應力較佳的物種以淘汰不良的染色體。

本論文中採用輪盤法（Roulette Wheel）來決定 Selection 的比例。首先將所有的染色體按照其評估函數值來分配其在輪盤上所佔面積的大小，而每個染色體所佔的面積比例，即代表被挑選至交配池中的機率。然後隨機地選取輪盤上的一點，其所對應的染色體即被選取至交配池中。由此觀點思考，評估函數值愈佳者，被選中的機率愈高，其精神符合適者生存的原則。以下為本論文之 Selection 執行步驟：

- 計算所有染色體之 fitness value *eval(s)*：

$$eval(s_k) \quad k=1,2,\ldots,pop\_size + offspring\_size$$

- 計算所有母體染色體之 fitness 平均值：

$$F = \sum_{k=1}^{pop\_size} eval(S_k)$$

$$AF = \left(F \Big/ pop\_size + offspring\_size \right)$$

- 計算所有染色體之 Selection value：

$$Selection(s_k) = AF \times R - eval(S_k)$$

$$\text{if}(Selection(s_k) > 0)$$

$Selection(s_k)=0$

此處之 R 為一動態參數，隨著執行子代數增加而縮小其值，在本論文中，R 值之設定如表 1 所示。扣除母體染色體之 fitness 平均值的目的是為了刪除 fitness value 較差之值，以加快收斂的速度。為避免侷限在區域最佳解的情況，我們運用 R 值來適應不同子代所需扣除之 fitness value。如表 2 中所示，R 值隨著代數增加而變小；因為隨著代數的增加，基因串 Fitness 值收斂，故基因串間 Fitness 值差異越來越小，所以 R 值的控制必須越嚴格。

表 2：R 值變動說明表

1~20 代	21~5 代	51~10 代	101~12 代	121~14 代	141~16 代	161~200 代
1.5	1.1	1.01	1.001	1.0001	1.00001	1.000001

- 計算所有染色體之 Selection value 加總：

$$SF = \sum_{k=1}^{pop\_size+offspring\_size} Selection\ (s_k)$$

- 計算所有染色體的 Selection 機率 Pk：

$$p_k = Selection(s_k) / SF \qquad k=1,2,...,pop\_size + offspring\_size$$

- 計算所有染色體的累積機率 qk：

$$q_k = \sum_{j=1}^{k} p_j \qquad k=1,2,...,pop\_size + offspring\_size$$

- 隨機產生 0,1 間數值，選取下一子代的母體

伍、實驗與評估

為測試分散式分級伺服器的效率，我們採用由 Hinet 提供的 12 萬筆真實網站資料，透過模擬方式，將分級資料量擴充至 100 萬筆。每筆資料除了分級資訊外，並包括網站的 Access frequency。由於不同網站的 Access frequency 有相當大的差異，尤其熱門網站的使用率明顯的比一般網站多了數倍，甚至幾萬倍；因此，將 Access frequency 的範圍訂定為 1/hour 至 10000/hour 之間，讓網站間的 Access frequency 可以有明顯的差別。另外，為考量熱門網站數量與總網站數量間的比例關係，Access frequency 越高的網站採樣數量越少。最後再將 Access frequency 除以 3600 求得平均一秒的 Access frequency，來判斷伺服器的瞬間負載。由於網站在不同時段的個別 Access frequency 亦不盡相同，故每筆分級資料包括不同時段 $T_1, T_2, T_3, ..., T_{n2}$ 的 Access frequency。

我們以 PIII 800MHz CPU、256MB RAM 的設備進行兩項模擬實驗。實驗一：分級資料量 n 分別為 100、200、400、1000、2000、10000、100000 及 1000000；伺服器數量 m=5。實驗二:分級資料量分別是 20、30、40、100、1000、10000、100000 及 1000000；伺服器數量 m=10。表 3 顯示不同的資料量時，資料由硬碟匯入每筆分級資料所需花費的時間。

表 3：資料由硬碟匯入總共花費的時間

n	20	30	40	100	200	400	1000	2000	10000	100000	1000000
Time/m	0.0004	0.0006	0.0008	0.0026	0.0032	0.0043	0.0077	0.0120	0.0499	0.4500	4.4399

我們以 AAF、NAW、最佳解演算法與本篇論文中提出的基因演算法進行比較，其中 AAF 演算法為以平均 Access frequency 為基準來分配分級資料；NAW 則以網站的數量為基準配置分級資料；最佳解演算法則根據所有可能配置分級資料的組合求出最佳的配置方式。實驗參數如表 4、5 所示：表 4 為實驗一的參數;表 5 則為實驗二的參數。我們以 20 串母體及 200 代的運算來執行基因演算法。另外由於在分級資訊的特性，使得單一基因變動範圍較大，因此採取機率較高的突變方式來執行運算。

表 4：實驗一的參數

Parameters					
Number of 網站 s		100,200,400,1000,2000,10000,100000,1000000			
Number of rating hosts		5			
Population size		20			
GA iterations		200			
Crossover probability		0.2			
Modified mutation rate		0.5			
Range	n	100	200	400	1000
	size	4	4	4	4
	n	2000	10000	100000	1000000
	size	4	8	16	32

表 5：實驗二的參數

Parameters					
Number of 網站 s		20,30,40,100,1000,10000,100000,1000000			
Number of rating hosts		10			
Population size		20			
GA iterations		200			
Crossover probability		0.2			
Modified mutation rate		0.25			
Range	n	20	30	40	100
	size	2	2	2	2
	n	1000	10000	100000	1000000
	size	4	8	16	32

進行了實驗一及實驗二之後，可由表 6 及表 7 了解基因演算法在分級資料配置上的效果。其中表 6 為實驗一的數據，表 7 則為實驗二的數據。在表 6 中可以看到網站數量 100、200、400 及 1000 的情況下本論文的基因演算法皆可在 200 代內求得最佳解，並且可以在極短的時間內完成。而當網站數量加大時，由表 6 中網站數量 2000、10000、100000 及 1000000 的實驗數據可以看出，若僅以平均 access frequency 的方式（AAF）或是平均網站數量的方式（NAW）來進行分配，伺服器間的負載將會很不平均；而運用基因演算法可使伺服器間負載差異的情形得到很好的改善。

在表 7 中可以看到當網站的數量超過 100 時，求取最佳解將會花費相當長的時間，約需要 10^5 分鐘以上。因為受到伺服器數量 m 的影響，原本在實驗一中伺服器數量 m 為 5 的情況下，還可以大致求取網站數量到 1000 的最佳解，但在實驗 m 為 10 的情況下，在求取 n 等於 40 時即需花費 3054.7249 分鐘；而在實際應用時，m 的值將有可能更大，因此求取最佳解將是不可行的方式。若使用基因演算法，不但可以求到近似最佳解，由表 6 及表 7 中網站數量 1000~1000000 萬的實驗數據中，可以看到表 7 中所花費的時間與表 6 無太大差異，亦即基因演算法執行分級資料配置的時間將不會受到 m 的大小影響，因此在實際應用中將不會有任何困難。

表 6：實驗一的數據

m=5	GA		AAF		NAW		OPT	
n	Time/m	Fitness	Time/m	Fitness	Time/m	Fitness	Time/m	Fitness
100	0.0023	32.66	0	60.92	0	49.02	0.7103	32.66
200	0.0051	95.34	0.0001	193.70	0.0001	189.34	18.2833	95.34
400	0.0113	273.65	0.0001	596.98	0.0001	500.02	570.7787	273.65
1000	0.0170	562.34	0.0001	1969.78	0.0001	1970.26	29109.7145	562.34
2000	0.0404	2687.87	0.0001	4062.58	0.0001	3702.21	$>10^5$	N/A
10000	0.1280	37603.26	0.0003	56423.59	0.0003	55882.92		N/A
100000	2.2188	3987323.41	0.0007	5405494.28	0.0007	5540400.33		N/A
1000000	26.1606	65896207.67	0.009	90815223.64	0.009	84477786.83		N/A

表 7：實驗二的數據

m=10	GA		AAF		NAW		OPT	
n	Time/m	Fitness	Time/m	Fitness	Time/m	Fitness	Time/m	Fitness
20	0.0006	1.90	0	6.54	0	2.45	0.9381	1.90
30	0.0010	3.41	0	7.90	0	4.52	22.3828	3.41
40	0.0013	4.31	0	8.77	0	5.13	3054.7249	4.31
100	0.0025	11.26	0	30.59	0	24.05	$>10^5$	N/A
1000	0.0177	262.62	0.0001	810.53	0.0001	780.19		N/A
10000	0.1031	13389.21	0.0003	19729.56	0.0003	20390.25		N/A
100000	2.6616	1098997.85	0.0007	1421644.79	0.0007	1525763.11		N/A
1000000	26.9824	31462308.35	0.009	40230287.52	0.009	42033912.27		N/A

在圖 7 及圖 8 中，將基因演算法所花費的時間與求最佳解及資料匯入的時間進行比較。由圖中可以明顯的看出求取最佳解的時間相當的耗時，而運用基因演算法只需花費接近資料匯入的時間；即使網站數量到達 1000000 萬筆時，基因演算法只需花費不到一小時的時間。因此，對於每個月重新檢討分級資料配置的管理方式而言，基因演算法是相當實用而有效率的方式。

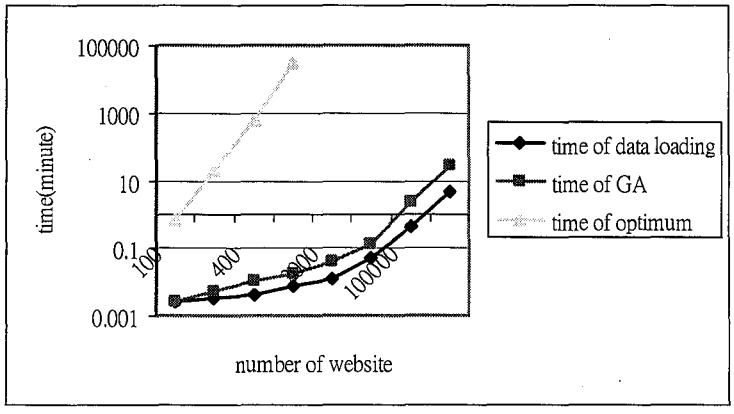

圖 7：實驗一的演算法處理時間比較

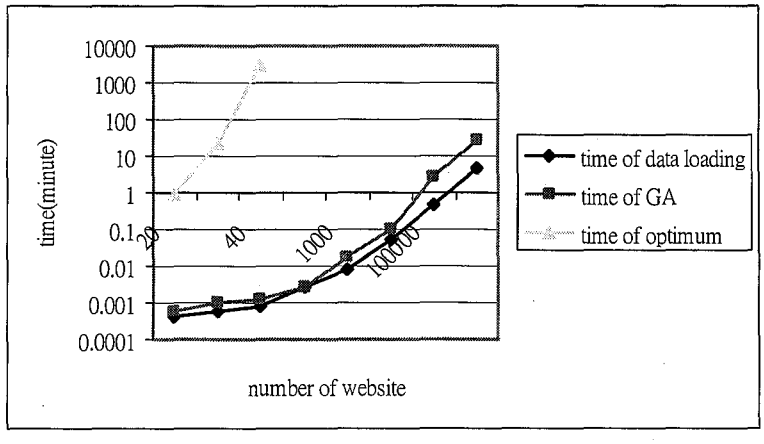

圖 8：實驗二的演算法處理時間比較

在圖 9 至 16 中，我們以直條圖呈現 GA、AAF、NAW、OPT 的適合度（Fitness）值來比較其差異。由圖 9 中可看到，在最佳解可求得的情況下，基因演算法提供的配置方式與最佳解非常接近，而 AAF、NAW 與最佳解差距則明顯得大很多。而在最佳解無法求得的情況下，基因演算法與 AAF、NAW 的適合度皆有明顯的差距。由此可知，基因演算法提供的分級資料配置方式，確實可以使伺服器間的負載差距減少，因此整體系統效能可大幅的提昇。

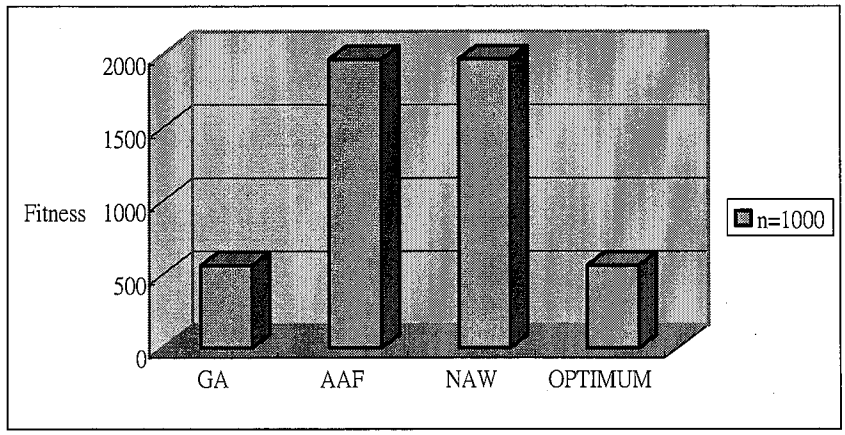

圖 9：實驗一在網站數量 1000 時的適合度比較

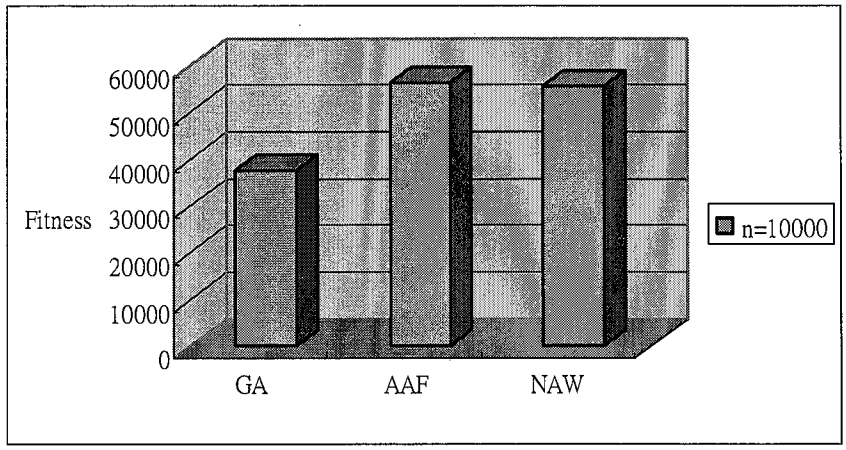

圖 10：實驗一在網站數量 10000 時的適合度比較

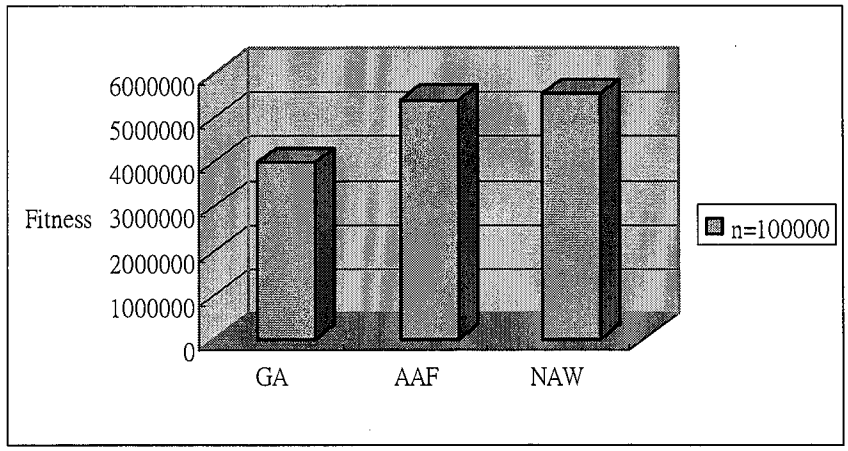

圖 11：實驗一在網站數量 100000 時的適合度比較

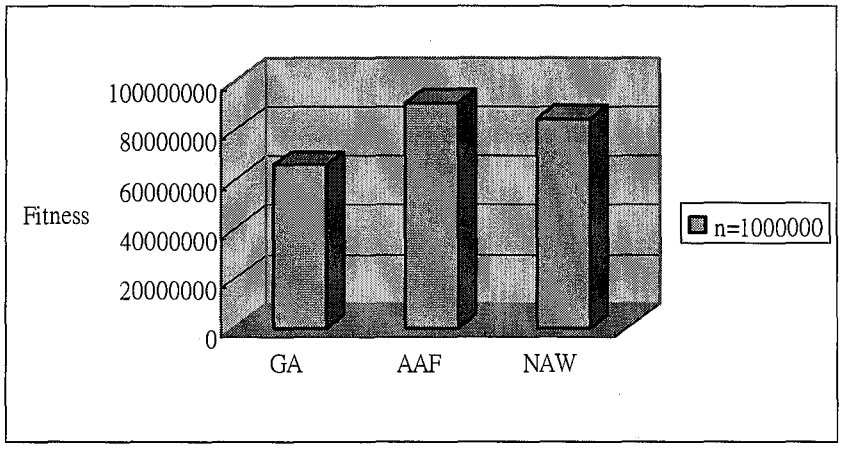

圖 12：實驗一在網站數量 100000 時的適合度比較

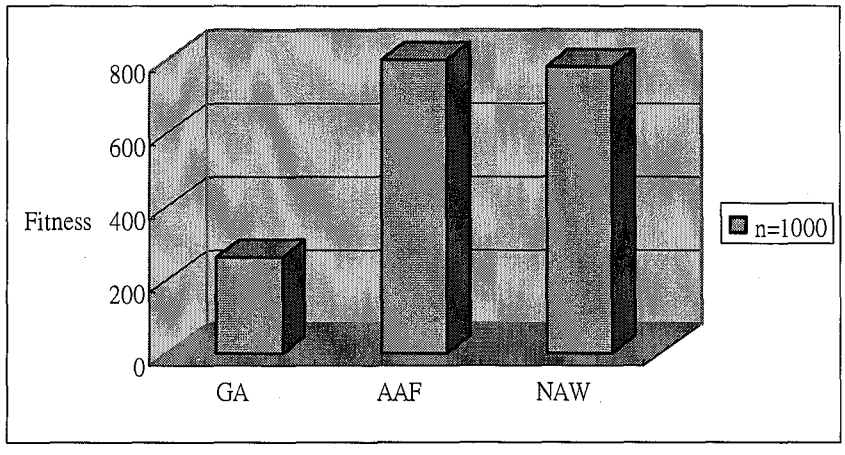

圖 13：實驗二在網站數量 1000 時的適合度比較

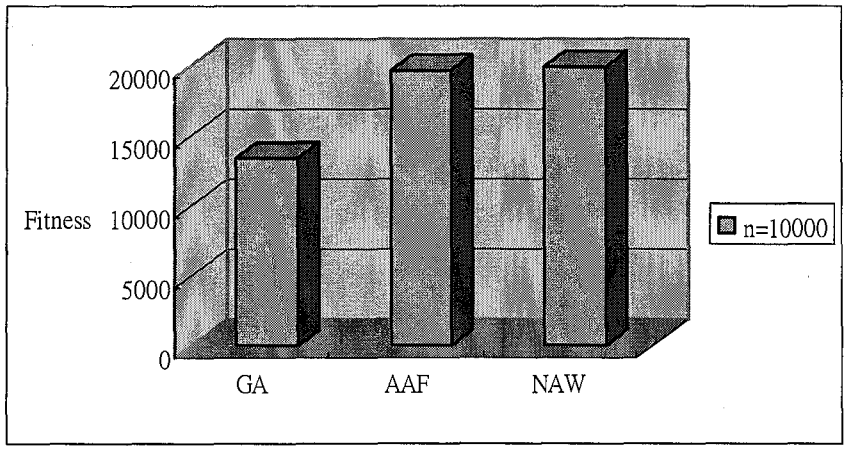

圖 14：實驗二在網站數量 10000 時的適合度比較

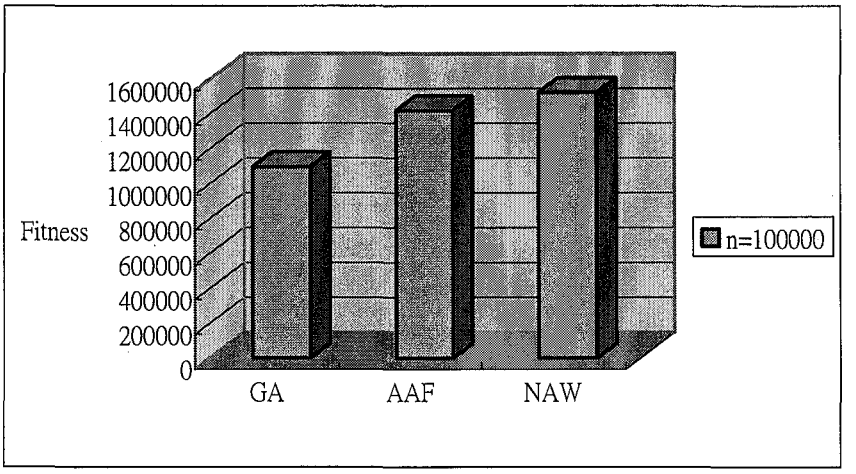

圖 15：實驗二在網站數量 100000 時的適合度比較

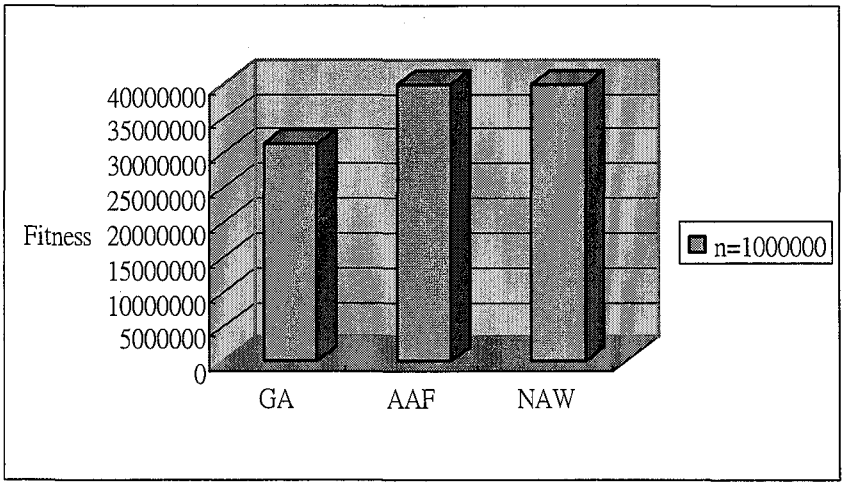

圖 16：實驗二在網站數量 1000000 時的適合度比較

在表 8 中我們列出 AAF 的適合度值減掉 GA 適合度值及 NAW 適合度值減掉 GA 適合度值的結果，由此可以了解基因演算法在分級資料分配上，與 AAF 及 NAW 的差異到底有多大，由表中我們可以看到隨著 n 的加大，差異也跟著加大，由此可知基因演算法並不會因為 n 的加大而使其效果變差，故在 n 很大的情況下，基因演算法是相當適用的。由此可知在使用基因演算法分配分級資料，可以使伺服器間負載差異降低許多，也較能達到我們均衡伺服器負載的目的。

表 8：GA 的解與 AAF/NAW 的差距

n	m=5		n	m=10	
	AAF-GA	NAW-GA		AAF-GA	NAW-GA
100	28.26	16.36	20	4.64	0.55
200	98.36	94.00	30	4.49	1.11
400	323.33	226.37	40	4.46	0.82
1000	1407.44	1407.92	100	19.33	12.79
2000	1374.71	1014.34	1000	547.91	517.57
10000	18820.33	18279.66	10000	6340.35	7001.04
100000	1418171.00	1553077.00	100000	322646.90	426765.30
1000000	24919016.00	18581579.00	1000000	8767979.00	10571604.00

柒、結論

目前台灣有許多單位或團體對網路內容的管理十分重視，並已投入相當的人力及物力資源，進行網路分級、網路犯罪預防以及青少年保護工作。而網站內容分級的推動，不僅需要觀念的宣導、政策的配合，在技術面也必須有所突破。尤其網路人口不斷暴增，頻寬經常不敷使用的情況下，缺乏效率的分級系統將是推動網站內容分級的最大阻礙。在本論文中，我們運用基因演算法來解決分散式分級資訊的規劃及配置問題，以達到提供大量且快速分級資訊服務的目的。由實驗的結果得知，運用基因演算法來執行分級資料的配置，效果相當的顯著，不僅可滿足使用者在時間上的要求，也兼顧到分級資訊的正確性。

除了本論文中考量之分級資料分配方式，不同的 ISP 業者基於實際管理面的考量，亦可以依據分級資料本身的特性來進行分配。例如依網站級別來分配，可將伺服器分為限制級、輔導級、保護級、普遍級的用途。此外亦可依網站類型分配，例如分為 com、edu、net、org 等等。由於不同分配或儲存方式在管理上及效率上各有其優缺點，未來在實際推動網路分級時，基於不同的實務考量，對分級資料的配置問題將有進一步探討的必要。而以長遠的經營網頁分級管理的角度來看，由於網頁資料眾多且內容變化快速，最終的理想還是期望網頁作者自行利用輔助工具，在自己的網頁上加上標示，畢竟集中式的管理只是過渡時期的作法，自律機制的推動才是長遠的經營之道。目前台灣網際網路協會正在推動 ISP 及 ICP 業者的自律公約，我們也期許業界的自律機制能在未來的 3 至 5 年順利啟動。

致謝

本研究由中華民國交通部電信總局補助，計劃編號: JHTB002-910328

參考文獻

1. 曾憲雄、林世華、黃國禎，「推動網路分級實施策略計畫」，行政院新聞局委託研究計劃報告，民 90 年。
2. 曾憲雄、翁秋平、馬宏燦、黃國禎，交通部電信總局「網站內容分級伺服器、用戶端介面及分級輔助系統之規劃建置」研究報告，民 91 年。
3. P. Y. Chang, D. J. Chen and K.N. Kavi, "File allocation algorithms to minimize data transmission time in distributed computing systems", *Journal of Information Science and Engineering*, 2001, Vol. 17, No. 4, pp. 633-646.
4. P. S. Chen and J. Akoka, "Optimal design of distributed information systems," *IEEE Transactions on Computers*, Vol. C-29, No.12, 1980, pp. 1068-1080.
5. R. S. Chen, D. J. Chen, and Y. S. Yeh, "Reliability optimization of distributed computing systems subject to capability constraints," *International Journal of Computer & Mathematic with Applications*, Vol. 29, No. 4, 1995, pp. 93-99.
6. M. S. Chen, J. Han, and P. S. Yu. (1996). "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp. 866-883.
7. D. W. Cheung, V. T. Ng, A. W. Fu, and Y. Fu. (1996). "Efficient Mining of association rules in distributed databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp. 911-922.
8. A. J. Chipperfield and P. Fleming. (1996). "Parallel Genetic Algorithms," *Parallel and Distributed Computing Handbook*, A. Zomaya, ed., pp. 1118-1193.
9. W. Chu, "Optimal file allocation in a computer network," *Computer-Communication Systems*, Prentice-Hall, Englewood Cliffs, N. J, 1973, pp. 82-94.
10. Z. Degraeve and M. Vandebroek (1998), "A mixed integer programming model for solving a layout problem in the fashion industry", *Management Science*, Vol. 44, No. 3, pp. 301-310.
11. A. E. El-Abd, "Modeling resources allocation and performance measures in distributed computer networks," *IEEE Singapore International on Networks/ Conference on Information Engineering*, 1995, pp. 581-586.
12. D. E. Goldberg (1989). "Sizing Populations for Serial and Parallel Genetic Algorithms,"

Proceedings of the Third International Conference on Genetic Algorithms, pp. 70-79.

13.R. P. Grimaldi. (1998). Discrete and Combinatorial Mathematics, Addison Wesley Higher Education, New York.

14.F. S. Hillier and G.. J. Lieberman (2001), *Introduction to Operations Research*, 7th Ed., McGraw-Hill, New York.

15.J. H. Holland. (1975). "Adaption in Natural and Artificial Systems", University of Michingan Press, Ann Arbor.

16.G. J. Hwang, H. C. Chu and Y. S. Yang (2002), "Development of an Integrated Environment for Website Content Labeling", *Chi Nan University Journal*, Vol. 6, No. 2.

17.M. F. Jiang, S. S. Tseng, and Y. T. Lin. (1999). "Collaborative Rating System for Web Page Labeling," *World Conference of the WWW and Internet*, Honolulu, Hawaii.

18.J. F. Kurose and R. Simha, "A microeconomic approach to optimal resource allocation in distributed computer systems," *IEEE Transactions on Computers*, Vol. C-38, No. 5, 1989, pp. 705-717.

19.Y. T. Lin, S. S. Tseng, and M. F. Jiang. (2000). "Voting based Collaborative Platform for Internet Content Selection," GCCCE 2000, *The Fourth Global Chinese Conference on Computing in Education*, Singapore.

20.J. T. Linderoth and M.W.P. Savelsbergh (1999). "A computational study of search strategies for mixed integer programming", *INFORMS Journal on Computing*, Vol. 11, No. 2, pp. 173-187.

21.S. Mahmoud and J. S. Riordon, "Optimal allocation of resources in distributed information networks," *ACM Transactions on Database Systems*, Vol. 1, No. 1, 1976, pp. 66-78.

22.L. Morgan and K. D. Levin, "Optimal program and data locations in computer networks," *Communication of ACM*, Vol. 20, No. 5, 1977, pp. 315-322.

23.P. Resnick and H. R. Varian. (1997). "Recommender systems," *Communications of ACM*, vol. 40, no. 3, pp. 56-58.

24.H. R. Parsaei. (1997), "Genetic Algorithms and Engineering Design", John Wiley & Sons, Canada.

25.P. Resnick and J. Miller. (1996). "PICS: Internet Access Controls Without Censorship," *Communications of the ACM*, vol.39 (10), pp. 87-93.

26.A. Yener and C. Rose. (1997). "Genetic Algorithms Applied to Cellular Call Admission Problem: Local Policies," *IEEE Transactions on Vehicular Technology*, Vol. 46, No. 1, pp. 72-79.

27.B. Yuwono and D. L. Lee. (1996). "Wise: A World Wide Web Resource Database System," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, NO. 4, pp. 548-554.

28.A. Y. Zomaya and M. Wright. (2002). "Observations on Using Genetic-Algorithms for Channel Allocation in Mobile Computing" *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 9, pp. 948-962.