南　華　大　學

資訊管理學系

碩士論文

無線廣播環境下應用 KL 演算法於相關資料排程的配置研究

Using K-L based Algorithm for correlated data allocation in wireless broadcast system

研 究 生：林曉吟

指導教授：蔡德謙　博士

中華民國 九十五 年 六 月

南　華　大　學

資訊管理學系

碩 士 學 位 論 文

無線廣播環境下應用 KL 演算法
於相關資料排程的配置研究

研究生：林曉吟

經考試合格特此證明

口試委員：吳光閔

_____

_____

_____

指導教授：蔡德諫

系主任(所長)：資訊管理學系 系主任吳光閔

口試日期：中華民國 95 年 6 月 29 日

誌　　謝

一年前，看見學長姐們拿著裝訂好的論文，臉上露出令人羨慕的燦爛微笑，如今自己也嘗到了這股喜悅的味道。感謝爸爸、媽媽對我的付出與鼓勵，讓我在充滿愛的無憂環境下成長。謝謝哥哥、大嫂和妹妹平日的照顧，感謝爸爸和哥哥二年來每天載著我來回穿梭二個縣市。還有三個可愛的小姪女，看著妳們純真的笑臉和爆笑的童言童語，不論有什麼不開心，頓時都能消失的無影無蹤。

感謝 蔡德謙老師二年來的指導，老師的謙虛和對事情認真的態度，值得我們學習，謝謝 吳光閔老師及 吳金山老師在百忙中抽空，對我的論文給予指正與寶貴的建議，在此表達對您們的謝意。

感謝乾訓、俊杰、明哲、如卿、美倫、俊男、建磐、元安等學長姐的照顧與經驗的分享。謝謝 325 研究室裡的夥伴們，活字典大師兄嘉明、熱心活潑的小童、搞笑的士颺、唱歌很好聽的昭勳、靦腆的坤錨、香噴噴的淑玲...等，還有特別要感謝學長閔皓，謝謝你總是在我失意難過時安慰我，在課業上給予幫助，默默的支持著我。以簡短的幾句話，表達對大家深深的謝意，謝謝!

無線廣播環境下應用 KL 演算法於相關資料排程的配置研究

學生：林曉吟　　指導教授：蔡德謙 博士

南　華　大　學　資訊管理學系碩士班

摘　　要

無線網路的環境之下頻寬資源是有限的，透過伺服端以廣播的方式，可以有效率的傳送資料、利用頻寬。伺服端決定出一組較佳的廣播序列，能夠滿足用戶端的查詢需求，讓用戶端快速的取得所需資料。

本文探討的無線廣播排程問題中，以廣播資料項之間存在的相依性為主要考量，使用有向無循環圖形(DAG)表示其順序限制，且每一資料項對應於圖形中每一頂點，並應用加入Greedy方式之拓樸排序演算法、積體電路設計中常被使用的KL演算法，以及SA模擬退火演算法等方法於有向無循環圖形的排序問題，試圖降低頂點之間平均路徑長度，以減少用戶端平均查詢讀取時間，並透過模擬實驗結果的顯示分析，比較這幾種方法的優劣。實驗結果顯示，使用我們所提出加入Greedy方式的拓樸排序演算法，以及KL演算法、SA演算法，確實可以縮短資料項之間的相依性長度，降低用戶端在收取所需資料項時花費的平均查詢讀取時間。

關鍵字: 無線廣播、KL 演算法、平均查詢讀取時間

Using K-L based Algorithm for correlated data allocation in wireless broadcast system

Student：Siao Yin, Lin　　Advisors：Dr. Derchian Tsaih.

Department of Information Management
The M.B.A. Program
Nan-Hua University

ABSTRACT

The resource of transmission bandwidth is limited under the wireless network environment. If data is broadcasted from server, bandwidth utilization can be maximized and data can be transmitted more efficiently. To satisfy the need of many different requests from mobile hosts, the broadcast server must schedule the data item such that most mobile host can read data from broadcast channel in short period of time.

Our context focus on the scheduling problem of wireless broadcast, consider dependence between each data item which are broadcasted. The directed acyclic graph was used with its edges as this dependence limitation and its vertex as its data item. Arrange the vertex set by topological sort with greedy strategy, KL algorithm which are used in integrated circuit design problem and Simulated Annealing (SA) algorithm to minimize the average edge weight across all vertices. This broadcasted order of data item will then minimize the average response time for client's queries. The results for applying each algorithm in data schedule problem were shown and compared through extensive simulation.

Keyword: wireless broadcast, KL algorithm, average response time

目　　錄

表 目 錄

圖 目 錄

第一章 緒論

1.1 研究背景與動機

由於無線科技的蓬勃發展，網路的使用者不需受到有線網路環境的侷限，用戶端利用行動裝置即在不受空間限制的環境下，存取所需資料，充份享受無線網路帶來的便利性。

無線網路的環境架構分成二大類，一是沒有固定架構的隨意網路(Ad Hoc Network)，端點與端點間資料的傳送、通訊，都是由各個端點自己負責，每一個行動用戶端都是資料傳送者，也可以是資料接收者的角色;另一個是主從式 Client-Server 的網路架構，由伺服器端將資料項廣播傳送出去，以滿足用戶端的查詢要求。

在無線網路的環境之下，頻寬資源是有限的。由伺服端以廣播的方式，可以有效率的將大量資料傳送出去，透過排程系統決定出一組廣播序列，經由廣播頻道週期性的廣播，使得用戶端能在最短時間內將所需資料項收取完成。

負責廣播的伺服端在進行廣播運作的方式又分為拉式廣播系統 Pull [5][8] [24][27][29]、推式廣播系統 Push [5] [27]、 以及 Hybrid 混合 Push 與 Pull 的廣播方式。拉式廣播系統 Pull 又稱作是請求式的廣播(On demand broadcast)，伺服端接收到用戶端的資料項請求時，利用接到請求作為考

量的依據，透過演算競爭後回覆給用戶端; 推式廣播系統 Push 是由伺服端依照演算法來決定廣播資料項序列，在一段時間內週期性的在頻道上廣播。

拉式廣播系統 Pull 適用於伺服端負載低時，但若出現大量用戶端，拉式廣播系統 Pull 可能無法即時的決定廣播序列，造成一些時間槽(time slot)未被使用而浪費掉的缺點; 推式廣播系統 Push 適用於伺服端負載重時，但如果用戶端要求的是被排入廣播序列的機率較不高的資料項，也就是較冷門的資料項，可能會苦等不到所要的資料，此時就必需藉由 Pull 的方式來向伺服端要求，Hybrid 混合式的廣播方式就是同時利用二種方式的優點，在負載重時，較適合使用 Push 方式進行廣播，在用戶端數量不多的時候，較適合使用 Pull 的方式進行廣播。

伺服端用以將資料項傳送出去的頻道(channel)，也有將資料項放置在多重頻道(multiple channel) [6] [14][25] [28]上進行廣播，以及將資料項放置在單一頻道(single channel)上進行廣播二種。

多重頻道有廣播頻道上的資料項，分成資料項不會重覆的在廣播序列上出現的廣播方式，稱為平坦式(flat)[6][14][13]，以及廣播頻道上的資料項會重覆的在廣播序列上出現的廣播方式，稱為非平坦式(non flat)[3][4][12]，如圖 2 所示;而單一頻道也分成平坦式與非平坦式二類廣

播方式。如圖 1 所示。

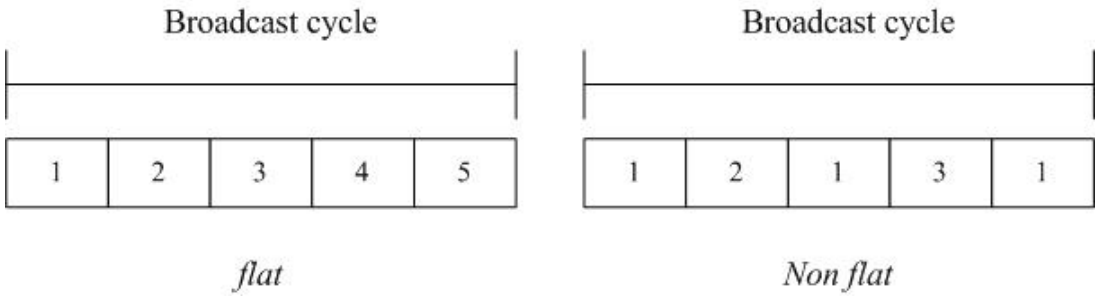


圖 1 單一頻道之平坦與非平坦廣播模式

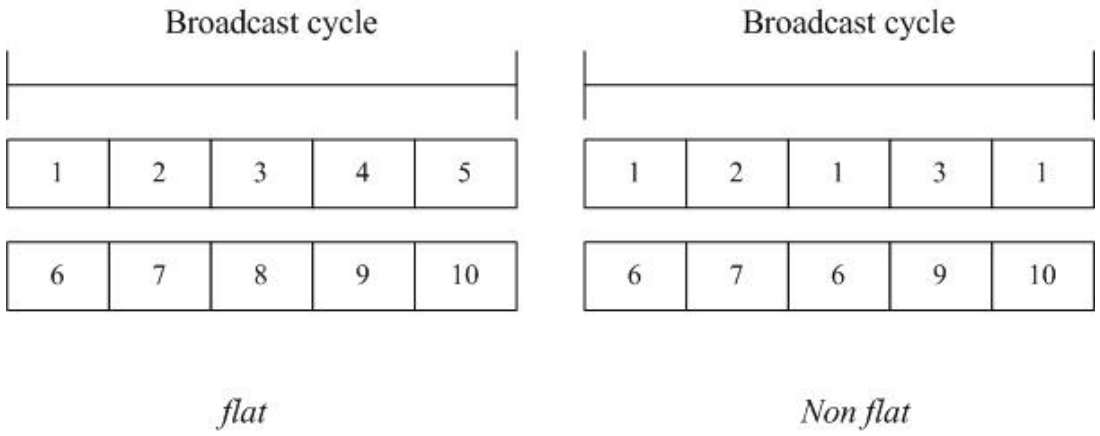


圖 2 多重頻道之平坦與非平坦廣播模式

每個用戶端發出的查詢(query)，有一個查詢只要求一個資料項的單一查詢(single query) [3][4]以及一個查詢要求二個或以上的資料項之多重查詢(multiple query) [22]。

在無線廣播排程問題中，本研究以廣播資料項之間存在的相依性為主要考量，使用有向無循環圖形(DAG)表示其順序限制，應用加入 Greedy 機制之拓樸排序演算法、積體電路設計中常被使用的 KL 演算法，以及

SA 模擬退火演算法等方法於有向無循環圖形的排序問題，試圖降低頂點之間平均路徑長度，以減少用戶端平均查詢讀取時間。

1.2 研究目的與限制

在無線廣播上廣為討論的議題主要有二大部份，一是節省用戶端在收取所需資料所要耗費的能源;另一部份是伺服端要如何決定出一組較佳的廣播序列，在較短的時間內能夠滿足眾多用戶端的查詢需求。

無線環境上的行動用戶端，其行動設備之電力供應完全是仰賴電池裝置，因此電力能源為有限的，節省用戶端在收取所需資料所要耗費的能源是一項重要議題[7][15] [16][17]。使用索引技術 [15] [16] [26][30]是可以減少能源消耗的方法之一，用戶端向伺服端發出要求資料項的請求，伺服端在接收到請求後，回傳一個索引值，告知用戶端它所需的資料項，要經由多少等待時間後會在廣播頻道上被廣播出來，在這段等待播出的時間內，用戶端可以進入較低耗電的休眠模式，等到資料項被播出時，再將資料項接收下來，就可以大量節省電力的消耗。

另一無線廣播上被討論的議題是降低用戶端的查詢時間 [17][22][31]，藉由將較為熱門的資料項之廣播次數增多，或許可以在較短的時間內滿足眾多用戶端的查詢需求，但往往會忽視掉查詢冷門資料項的用戶端，使得這些用戶端久久都存取不到所需資料項，所以在一些研究當中

考量公平性，在有限時間內必定滿足用戶端的查詢需求。

在我們的論文中所要探討的議題，著重在利用演算法來進行相關資料廣播排程，使得無線廣播環境下的眾多用戶端，都能很迅速的存取到其所需要的資料項。用戶端所發出的多資料項查詢中，資料項存在著相依性，要求的資料項有順序上的限制，我們針對這個作為主要方向。經由拓樸排序之後的廣播序列，藉由 KL 演算法與 SA 演算法，透過不斷的移動資料項在廣播序列上的位置，試圖找尋資料項間之較小相依長度，以求能降低用戶端的平均查詢讀取時間。

依照我們所探討的方向，本研究僅適用於部份之無線網路，因此我們研究上的限制為著重在無線廣播排程，將排程之演算法使用於單一頻道廣播系統上，不考慮多頻道資料配置問題。適用於推式廣播系統(Push System)，探討資料項在廣播週期裡是非重覆進行廣播，也就是平坦式的廣播方式。

1.3 研究架構

本論文架構共分成六個章節，第二章的文獻探討裡，主要收集、彙整與本研究的主題相關之文獻，包含圖形結構、拓樸排序演算法與 KL 最佳化演算法，以及 SA 演算法。第三章問題描述則是定義在本研究中所要解決的問題，以及符號的定義與基本假設。第四章研究方法的內容主

要在說明本研究，應用 Greedy 拓樸排序、KL 最佳化演算法，SA 演算法等方法，在進行廣播排程問題解決上的演算流程。第五章為實驗分析，決定各項參數並以 JAVA 程式語言撰寫程式進行模擬，再針對本研究在問題解決上所使用的演算法其效能分析。第六章的結論會根據第五章裡面進行模擬實驗得到的結果分析、比較。

第二章 文獻探討

2.1 圖形結構

一個圖形G是由一個頂點集合V與一個邊集合E來表示，集合V由圖形G的頂點(vertices)所組成，集合E由圖形G中，不同的成對頂點之間的邊(edges)所組成。如果e_{vw}是一個由頂點v與w組成的邊，則e_{vw}連接頂點v與w。如果成對頂點之間是沒有順序之分的，則圖形G為無向圖形(undirected graph);若成對頂點之間是有順序之分的，則圖形G稱為有向圖形。

2.1.1 無向圖形(Undirected Graphs)

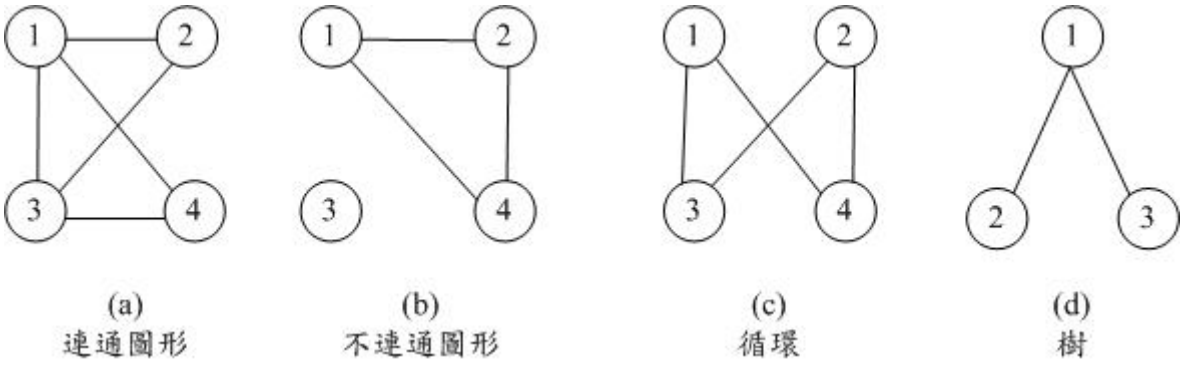


圖 3 各種無向圖形

在圖 3 裡顯示幾種不同種類之無向圖形。在一個無向圖形中二個頂點如果存在一個由此頂點到另一個頂點的邊，我們稱之為二頂點相鄰(adjacent)。

圖 3(a)的無向圖形中，頂點 1 與 2 相鄰，頂點 3 與 4 也相鄰，但 2

與 4 不相鄰。如果從任何一個頂點至少存在一條路徑到達其它的任何一個頂點，則稱此圖形為連通(connected)，圖 3(a)與圖 3(c)皆為連通圖形;在圖 3(b)之中的頂點 1、2、4 無法經由任何路徑到達頂點 3，則圖 3(b)為一個不通連圖形;一個循環(cycle)為一條至少包含三個頂點的路徑，其中路徑的最後一個頂點和第一個頂點相鄰，如圖 3(c)。圖 3(d)顯示一個沒有循環的連通圖形。

2.1.2 有向圖形(directed graph)

所謂有向圖形就是它所擁有的邊皆是有著相同方向的。一條路徑或是一個循環都是順著箭頭指示的方向移動。這樣的一條路徑稱為有向路徑(directed path)，而如果這條有向路徑構成了一個循環，則稱為有向循環(directed cycle)，如圖 4(a)。在圖形裡的任何一個頂點，皆可透過至少一條的有向路徑到達其它的任何一個頂點，則這個有向圖形就稱它為強連通(strongly)，如圖 4(b)。若不考慮邊的方向，以無向圖形的觀點來看，這個圖形是連通的話，稱之為弱連通(weakly connected)，如圖 4(c)。

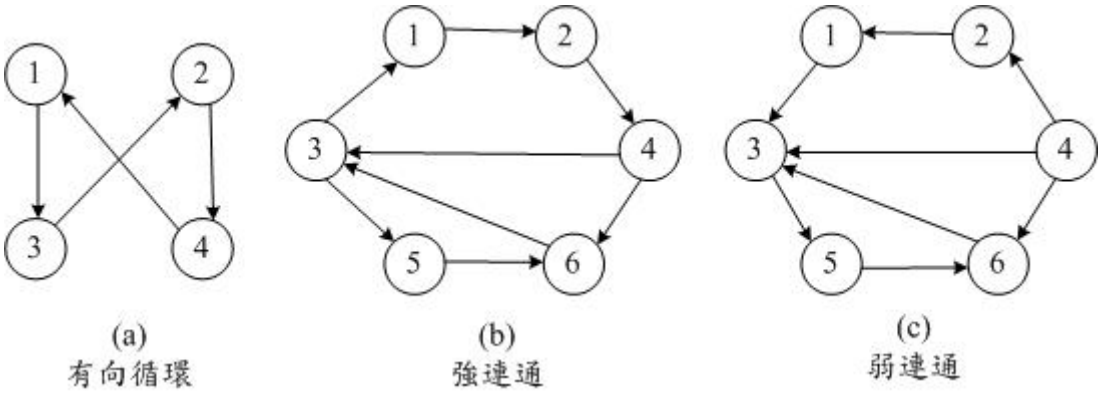


圖 4 各種有向圖形

2.1.3 頂點工作網路(Activity On Vertex Network)

網路圖形可以用來協助規劃大型工作計劃，通常在計劃一個大型工作時，首先會將繁複的工作細分成多個細項，可將每一個工作細項看成是網路上的一個頂點，由於每一個工作細項之間存有完成的先後順序，有一些可以同時進行，有一些則是有強制性的先後順序關係，無法同時進行的，因此用網路圖來表示其先後完成之順序限制，這種以頂點來代表工作項目的網路稱為頂點工作網路(Activity On Vertex Network)，簡稱 AOV 網路。

以圖形來定義 AOV 網路， 它其實就是一種有向圖形，在這個有向的圖形中，它的每一個頂點代表一項工作或必需執行的動作，而那些有方向性的邊則代表著，工作與工作之間存在的先後順序關係。

以下為幾個圖形內的專有名詞:

(1)前行者: 指的是若頂點 i 的工作必需先完成後，才能進行頂點 j 的工作，則我們就稱頂點 i 為頂點 j 的前行者。

(2)拓樸排序與拓樸序列: 如果在 AOV 網路中，具有部份次序關係，也就是說有某幾個頂點為前行者，而利用拓樸排序可以將這些部份次序關係，轉換成線性次序(Linear Order)的關係。例如:頂點 i 是頂點 j 的前行者，在線性次序當中，頂點 i 會排在頂點 j 之前的位置，具有這種特性的線性次序就稱為拓樸序列(Topological Order)。

2.2 拓樸排序

下列為拓樸排序與拓樸序列的摘要說明:

(1)產生拓樸序列必需存在條件是一個無循環(acyclic)圖形，由於 AOV 網路代表各個細項工作的先後完成順序圖，所以沒有循環的問題，AOV 網路經過拓樸排序後可以產生有線性次序關係之拓樸序列。

(2)在一個 AOV 網路經過拓樸排序之後,所產生的拓樸序列可能有一個以上，也就是拓樸序列並不是唯一解。

進行拓樸排序的步驟如下所示:

步驟一: 找尋圖形中任何一個沒有前行者的頂點。

步驟二: 輸出找尋到的這個頂點，並將這個頂點的所有邊消除。

步驟三: 重複步驟一與步驟二，直到所有頂點皆被選出。

一個有向圖形有兩種不同的拓樸順序，以圖 5 為例，如圖 6、7 所示，一個是深度優先走訪(Depth-First Algorithm)，另一個是廣度優先走訪(Breadth-First Algorithm)。

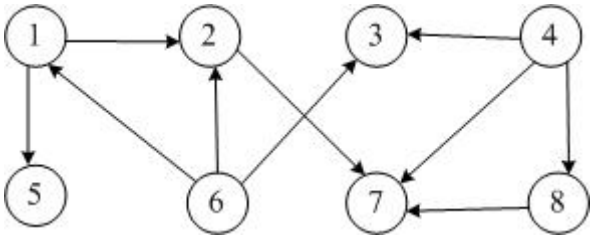


圖 5 有向無循環圖形範例一

2.2.1 深度優先走訪(Depth-First Algorithm):

在一個拓樸順序中，每一個頂點必需在有向圖形中它的後繼者之前出現。在一個深度優先之拓樸順序中，我們一開始找到一個沒有後繼者的頂點，並且將它放到順序中最後一個位置。藉由遞迴之後，我們已將一個頂點它所有的後繼者放入拓樸順序之正確位置，然後我們可以將此頂點放於它的所有後繼者前面。演算法的程式執行中，並未作任何搜尋動作，所以執行時間為 O(n+e)，其中 n 為頂點個數，而 e 代表是圖形中所有邊的數目。

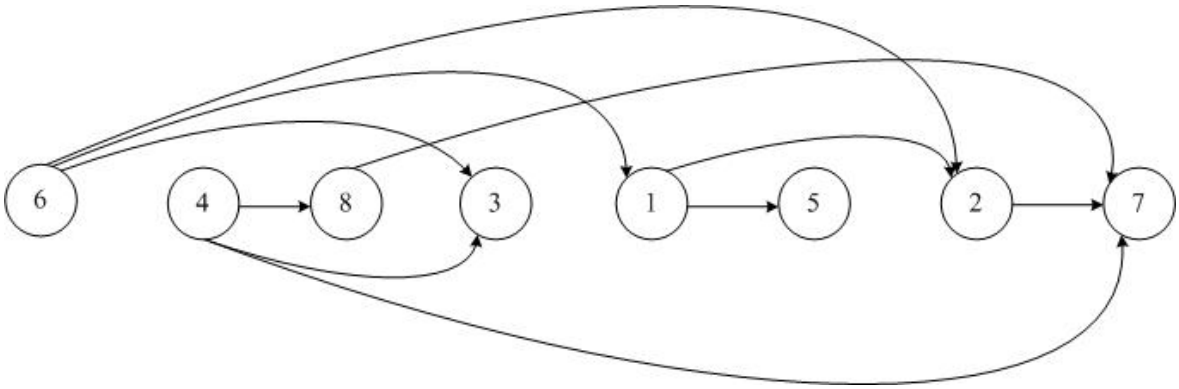


圖 6 以圖 5 為例之深度優先順序

2.2.2 廣度優先走訪(Breadth-First Algorithm):

在一個有向無循環圖形中的廣度優先拓樸順序，首先找尋到在拓樸序列裡在前面的頂點，然後應用每一個頂點都必需出現在它的後繼者前面的事實來排序。拓樸序列中前面的頂點由非任何頂點的後繼者所組成。為了找到這些頂點，我們設定一個 *degree* 矩陣，存放著每一個頂點的前行者個數，*degree* 矩陣內一為 0 的代表這個頂點沒有任何的前行者，用以作為拓樸排選取頂點時的考量。每選出一個排入序列內的頂點，在選出之後必需將相依於它的有向邊去除，重新計算更新每一個還未被選出的頂點其 *degree* 數，重覆執行程序直到所有頂點皆被選出排入序列當中。如同深度優先走訪一樣，廣度優先走訪所需的執行時間同樣為 O(*n*+*e*)其中 *n* 為頂點個數，而 *e* 為此圖形中所有邊數目。

我們知道拓樸排序所輸出的結果並不是唯一解，如果同時有二個或以上的頂點皆沒有前行者，那排序得到的結果就不是唯一解。除此之外，

如果 AOV 網路中的每一個頂點都有前行者，則表示這個網路含有循環，無法進行拓樸排序。

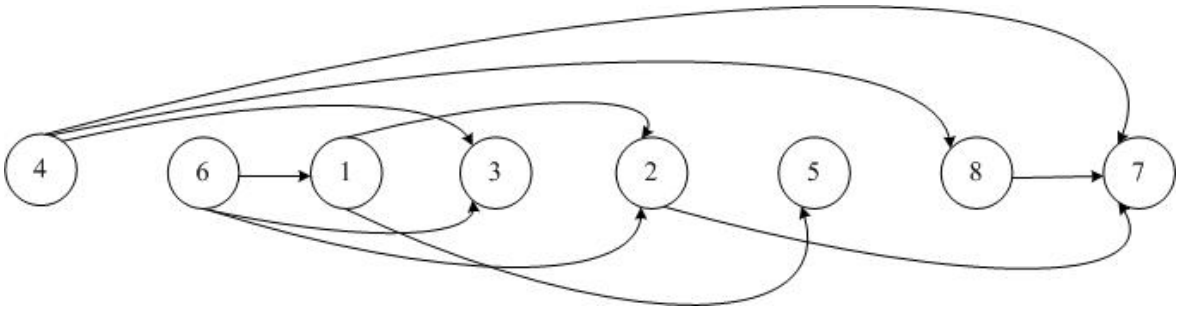


圖 7 以圖 5 為例之廣度優先順序

2.3 KL 最佳化演算法

在超大型積體電路設計(VLSI design)程序之中，電路分割(circuit partitioning)一直扮演非常重要的角色，可使得日漸複雜的電路在不影響原先電路功能的限制之下，依照元件間互連的資訊(interconnection information)把原先電路分割成數個子電路，簡化設計流程以獲得較好之系統效能。通常依據產生的子電路個數，分類為二重(two-way)或多重(multi-way)分割。

一種經常被探討的電路分割問題乃是在不違反各個子電路內可容納元件個數或面積的限制下，期望在分割後，各個子電路間互連的訊號線的個數能夠越少越好，此類問題被稱為有size constraint 的最小切(min-cut)分割。

電路可以利用超圖形(Hyper graph)來描述，所以電路分割又可以視為超圖形分割。在超圖形中的節點(node)代表電路中的閘道元件(gate)，而當不同的元件之間有相連的訊號線(net)時，則以邊(Hyper edge)來表示。給定一個超圖形$G=(V,E)$，其中V為節點之集合、E為邊之集合，則有size constraint的k 重最小切分割就是將V分成k個子集合，同時在滿足每個子集合內頂點個數，皆不低於最少頂點數的條件下，期望連接各個子集合的邊總數(即external net 總數)越小越好。

最小分割問題被認為是NP-hard 的問題[10]，因此從過去至今有許多種演算法不斷被提出，包括有反覆執行改善(iterative improvement)的演算法[20] [9]、模擬退火(simulated annealing)[21]的演算法、光譜式(spectrum)的演算法[11]等。Kernighan-Lin 演算法(簡稱為KL 演算法) [20]其優點為執行時間短，在最佳化問題解決上廣泛的被使用。

給定一個二重的初始分割解，KL 演算法不斷利用交換分屬不同子集合內的節點的技巧，企圖降低external net 的個數，直到無法改善為止；而節點交換的原則是從目前可以被交換的節點中選擇增益(gain，即交換後external net 個數的改變量)最大的做交換。

2.4 SA 模擬退火演算法

模擬退火演算法為一種機率攀登搜尋演算法（Probability

Hill-Climbing Search Algorithms)，其結合了梯度搜尋法（Gradient Search）與隨機過程的方式去搜尋目標函數（Objective Function）的整體最佳解。它採取隨機的方式產生新解，在所有解空間中搜尋，並利用目標函數評估其值，期望它往目標值最佳的狀態移動。但同時也允許產生的新解可以在某些機率條件下，往目標值較差的地方移動，也因為這種機率條件而提供模擬退火法具有避免落入區域最佳解的能力，進而可以進行全域搜尋的最佳化方法，但並不保證可以找到最佳解。

退火演算法是屬於啟發式方法的一種，啟發式方法(heuristic methods)是以直覺上的想法設計出來的求解程序，雖然它不一定可以得到最佳解，但往往可得到不錯的近似解，因此啟發式方法常被應用在工作的排程上。

SA(Simulated Annealing)模擬退火演算法的最初觀念是由Metropolis[1]等學者在 1953 年提出，Jepsen[21]等學者在 1983 年以蒙地卡羅法則(Monte Carlo Method)為概念基礎發展出一種隨機搜尋方法，而蒙地卡羅法則指的是在計算中引用亂數(random number)的一種隨機計算法。隨機演算法有可能會得到較差的結果，但在滿足限制條件下是可以被接受。

SA 主要的概念來自將固體以一固定溫度加熱融解，再重新組合成另

一結晶型態的整個退火過程的模擬。以最佳化問題而言，當找尋到的解是落在區域最佳解之中時，SA 可藉著 Metropolis 法則[1]來判斷是否要接受，因此也讓結果能跳脫區域最佳解，有成為另一更佳解的機會。

Metropolis 法則主要用來判斷是不是要接受一個結果較不好的暫時解，可以使得在搜尋解的過程中，避免落入區域最佳解。當搜尋到的新解成本比現有解之成本大時，有一個非 0 的機率來決定是不是要接受交換。SA 基本上是以 Metropolis 法則為基礎，並配合退火程序，藉由溫度的逐漸降低來決定是否接受成本較差的新解之機率，此機率隨著溫度的降低而降低。

SA 被提出之後，即被廣泛運用在許多領域上，甚至被視為是解決最佳化問題的通用技術，1983年時Kirkpatrick、Gelatt、Vecchi [21]將模擬退火演算法應用在求解最佳化問題上，以及Cerny 等人首先利用SA 的觀念來解決VLSI設計問題。1984年Hinton以模擬退火衍生出Boltzmann machine，用於求解中子輸運的波茲曼(Boltzmann)方程式。

SA 包含以下四個基本要素[2]:

1. 系統狀態(Configuration)

在某一溫度下，系統產生的初始解，並作為目前時間的可行解。

2. 搜尋法則(Move set)

在退火的過程中，由目前的系統狀態經隨機擾動以致產生變化到另一種狀態。

3. 成本函數(Cost function)

衡量某一個系統狀態之下的能量函數，也就是目標函數值。

4. 退火程序(Annealing schedule)

在退火過程中包含了四個參數，分別是初始溫度、溫度下降的時機、冷卻率以及終止溫度。

第三章 問題描述

在此章節中，我們首先定義無線環境裡資料配置的問題。資料的配置乃決定於廣播的先後次序。假設所有資料項的大小皆相同，而且每一個查詢都存取多筆資料項。

3.1 無線廣播排程

整個無線網路的大環境，分成非基礎架構的隨意網路與有基礎架構的主從式 Client-Server 架構。在 Client-Server 基礎架構下的廣播系統運作主要以基地台為主，用戶端透過基地台涵蓋的無線電波範圍與基地台進行通訊的運作，而用戶端與用戶端之間一般是無法直接通訊，是必需透過基地台的轉送資料來進行交換資料的動作。在 Client-Server 架構下廣播資料的方式又分為 Push 與 Pull 兩大類。

Pull 系統則是根據用戶端發出的請求,廣播伺服器依照這個請求裡被要求的資料項，放入廣播頻道裡滿足用戶端的需求，且較熱門的資料項目可能可以較早發送，以節省用戶端的等待時間；用戶端透過無線電波與伺服器端通訊，傳送需求給伺服器；伺服器在收到來自於用戶端的請求後，透過排程演算法與資料項之間的計算，再將資料排入廣播序列裡進行廣播動作。圖 8 為 Pull 廣播系統的架構。

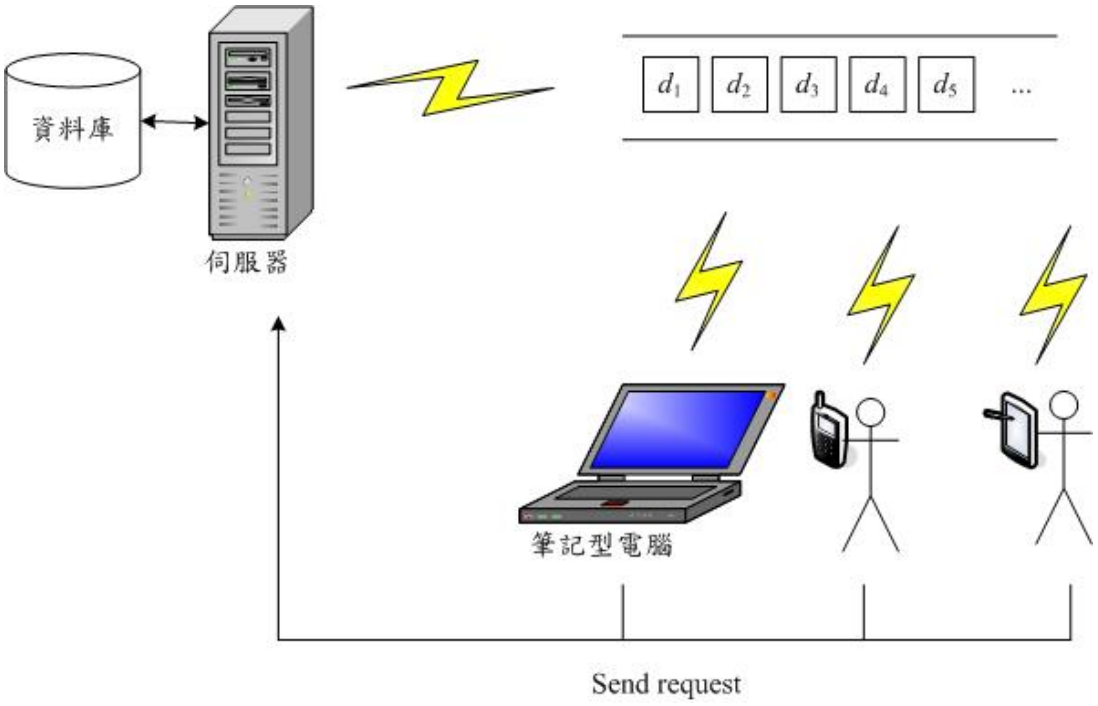


圖 8　Pull 廣播系統架構

Push 系統是由廣播伺服器將其資料庫內部儲存的所有資料項，透過排程演算法決定排入廣播序列的位置，利用週期性的廣播不斷播送資料項；用戶端傾聽到伺服器正在廣播自己所要的資料項時，再將其收取下來。圖 9 為 Push 廣播系統的架構。

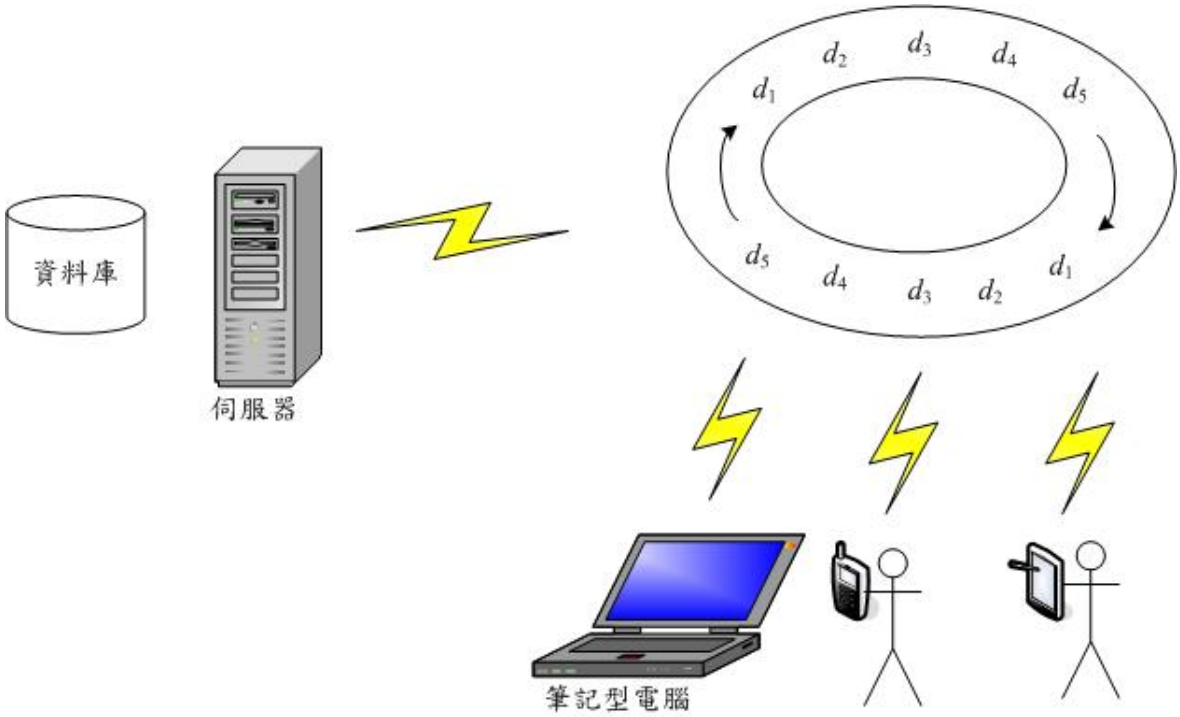


圖 9　Push 廣播系統架構

3.2 平均查詢讀取時間

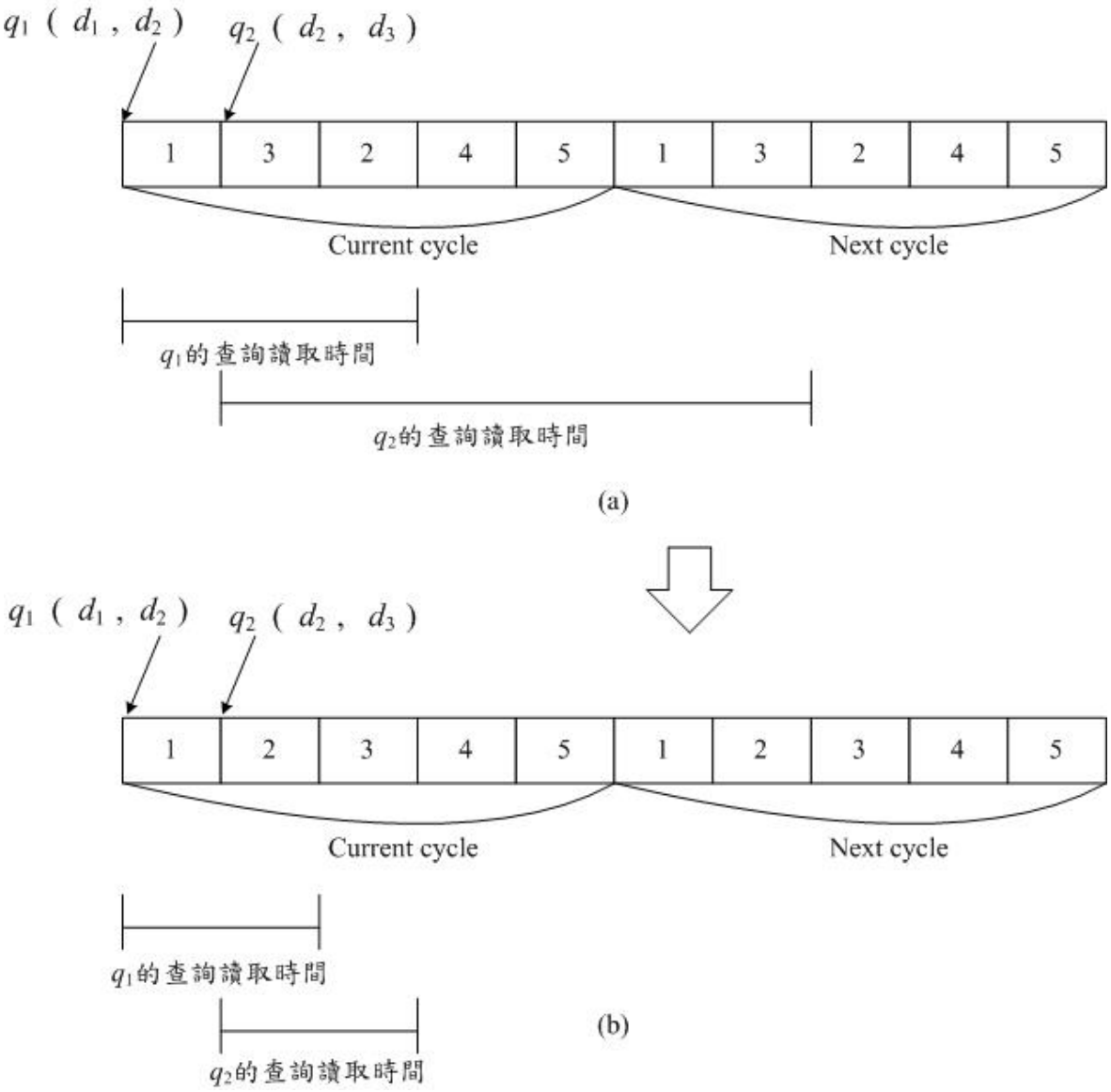


圖 10 平均查詢讀取時間

在我們所要探討的相關資料廣播排程問題上，用戶端對於所要求的資料項是有收取先後順序限制，例如:一個用戶端欲查詢嘉義的天氣狀況，先進到氣象首頁，接著查詢南部天氣，再查詢嘉義詳細的天氣狀況，

他對伺服器的查詢要求有先後的次序。

假設現在有多個用戶端對資料項的查詢q_i，分別為$q_1(d_1,d_2)$以及$q_2(d_2,d_3)$，q_1第一個要求的是d_1，第二個要求的是d_2，q_2第一個要求的是d_2，第二個要求的是d_3，f_{12}為收取完d_1接著要求d_2的存取頻率，f_{23}為收取完d_2接著要求d_3的存取頻率。以圖 10(a)來看，q_1查詢讀取時間為$D_{12}\times W^{1}[1][2]=3f_{12}$，而$q_2$的查詢讀取時間為$D_{23}\times W^{1}[2][3]=6f_{23}$;將相關程度高的資料項排列在較集中的廣播位置，降低相關性資料項之間的相依長度，廣播序列由圖 10(a)轉換變成圖 10(b)所示，q_1的查詢讀取時間為$D_{12}\times W^{1}[1][2]=2f_{12}$，而$q_2$的查詢讀取時間為$D_{23}\times W^{1}[2][3]=2f_{23}$，查詢的平均查詢讀取時間明顯降低。

3.3 最小分割問題

圖(graph)通常用於描述一個系統中各個物體間之關聯性。圖$G(V,E)$包含了一個頂點集合(a set of vertices)V 與邊集合(a set of edges)E，其中每一個頂點表示該系統中的一個物體，而具關聯之兩個頂點之間則以一個邊相連。$G(V,E)$中，V 為所有頂點v_i之集合，記為$V=\{v_1,v_2,v_3,v_4,v_5\ldots\}$，$E$ 為所有邊e_{ij}之集合，記為$E=\{e_{12},e_{23},e_{34},e_{45}\ldots\}$。其中，每一個邊表示其所相連之兩頂點與之間具有關聯，例如e_{12} 表示v_1 與v_2 之間具有關聯。而圖中任兩個頂點之圖距(或稱拓撲距離)為連接此兩頂點最小之邊數目。圖分

割之目的為，以頂點為基本單位，將圖 $G(V,E)$ 分割為數個子圖(sub-graph)，並滿足以下兩條件：

1. 各子圖所擁有之頂點數目相當。

2. 連接不同子圖之邊數目(或稱為邊切數)為最小。

然而圖分割為一NP-complete 問題。目前有一些已發表的圖分割方法，能在有效時間內找出近似的最佳解。圖分割技術被應用於許多科學計算的領域，諸如稀疏矩陣重排[19]、超大型積體電路設計(VLSI design)[18]等領域。

為了將圖分割技術應用於廣播資料排程問題，首先需將有限制的廣播資料拓樸序列轉換為圖之型式，以便進行圖分割的步驟，最後再將已被分割之圖與子圖，還原為廣播序列。以廣播資料項之觀點來看，圖(graph)中每一個頂點均代表一個資料項，而頂點之間的邊代表著二個資料項之間的相關性。

G 是一個無循環之有向圖形，則 G 的一個拓樸順序(Topological order)為一種將 G 中所有頂點順序列出的方式，使得對所有的頂點 i，j 屬於 G，如果有一個邊從 i 到 j，則在此循序串列(sequential listing)之中，i 會排於 j 之前。

此種性質的圖形發生在許多問題中。例如: 將一所大學所開設的課程

視作一個有向圖形的頂點，如果課程 i 是課程 j 的先修課程，則存在一個從課程 i 到課程 j 之有向邊。此時一個拓樸順序就是將所有的課程依序列出，使得每一個課程之所有先修課程都會出現在該課程之前。若依照拓樸排序後的選課順序，就一定不會因為有該修而未修的科目，而發生被擋修的情形。

拓樸排序所輸出的序列結果並不是唯一的，如果同時有二個或以上的頂點皆沒有前行者，那排序得到的結果就不是唯一解。將廣播排程問題轉換到圖形結構上，可將一個資料項視為一個頂點;頂點與頂點之間的有向邊則代表著資料項之間的相依性，有向邊的終止頂點相依起始頂點;以資料項被用戶端存取的頻率為有向邊的權重，而有向邊的權重值則表示著資料項相依的高低程度，若能把相依程度較高的資料項，在決定廣播序列時，排在較近一點的位置，縮短資料項之間的平均長度，以求降低用戶端平均查詢讀取時間。

第四章 研究方法

為了解決相依性資料項廣播排程問題。在這個部份中，將介紹我們所使用的加入 Greedy 方式之拓樸排序、KL 最佳化的演算法、以及 SA 演算法，應用於廣播排程問題解決的演算流程，並使用例子來加以說明。

4.1 廣播查詢轉換有向圖形

下列為演算法中使用到的符號定義說明:

- *V*: 所有頂點集合，一個頂點表示一個資料項，*V*={1,2,3,4,5….}。
- *E*: 連接二個頂點的有向邊(edge)集合，也就是成對資料項之間的相關性集合。
- e_{ij}: 由頂點 i為起始端指向由頂點 j為終止端的有向邊(edge)。代表進行廣播序列排程時，資料項i的廣播位置需在資料項j的廣播位置之前，用戶端對於資料項i與資料項j的收取是有順序限制的，$e_{ij} \in E$。

EX: e_{12}代表存在著一個由頂點 1 為起始端，指向由頂點 2 為終止端的有向邊(edge)。

- A^1[i][j]: 相鄰矩陣元素，$A^1[i][j] = \begin{cases} 1 & ，\text{if } e_{ij} \in E \\ 0 & ，\text{otherwise} \end{cases}$。

相鄰矩陣指的是建立一個路徑矩陣，從這個矩陣中可以判斷

圖形G中的成對頂點 i與j是否相鄰。在相鄰矩陣當中只存在 0 與 1 二種值，若A^1[i][j]=0 表示i與j二個頂點不相鄰，若 A^1[i][j]=1 表示i與j這二個頂點之間有一條長度為 1 之路徑存在，i與j二個頂點是相鄰的。

EX: 利用二個頂點間存在的長度為 1 之路徑，建立相鄰矩陣。

A^1=

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

其中A^1[1][2]=1 表示頂點 1 到頂點 2 有 1 條長度為 1 的路徑，頂點 1 與 2 為相鄰的二個頂點。

- *degree*[j]: 進入頂點 j的edge總數，*degree*[j]= $\sum A^1$[i][j]，i , j$\in V$，i 為起始端，j 代表終止端。

EX: 利用A^1相鄰矩陣計算各個頂點之*degree*:

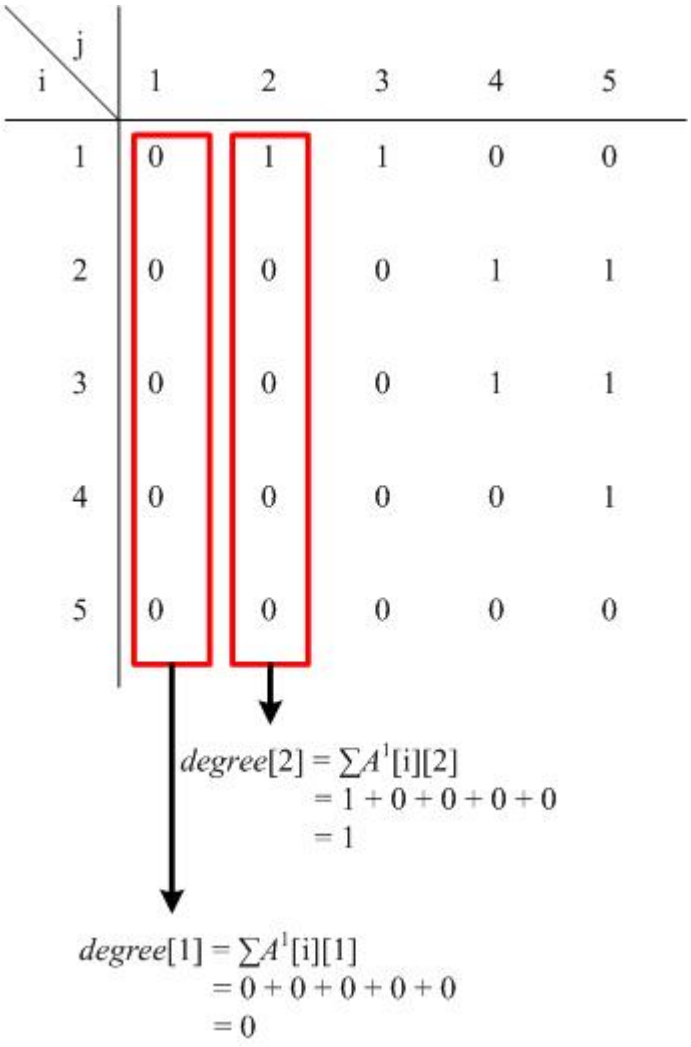


圖 11 相鄰矩陣A^1

- *TC*[i][j]: 遞移封閉性矩陣元素，令 $P = \sum_{i=1}^{|V|} A^i$，則得遞移封閉性矩陣(Transitive Closure)，$TC[i][j] = \begin{cases} 1 & \text{，if } P[i,j]>0 \\ 0 & \text{，otherwise} \end{cases}$

  所謂的遞移封閉性(Transitive Closure)指的是建立一個路徑矩陣(path matrix)*P*。

在路徑矩陣 *P* 當中只存在 0 與 1 二種值，若 *P*[i][j]=0 表示 i 與 j 之間不存在任何一條路徑，若 *P*[i][j]=1 表示 i 與 j 這二個頂點之間有一條路徑存在，且路徑長度大於等於 1。

建立一個圖形其路徑矩陣 *P* 的步驟如下:

---

(1)首先求得A^1，A^2，A^3，A^4.... A^n，其中A^n裡的元素A^n [i][j] 表示圖形中頂點 i與j間存在長度為n的路徑總數。

(2)將矩陣A^1，A^2，A^3，A^4.... A^n全部相加，即令*P*= $A^1+A^2+A^3+A^4$.... $+A^n$。

(3)令 *P* 矩陣中所有元素大於 0 的為 1，可以得到元素值只有 0 與 1 的矩陣 *TC*，也就是這個圖形 *G* 的 Transitive Closure。

EX:

步驟一: 先求出相鄰矩陣A^1

A^1=

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

其中A^1[1][2]=1 表示頂點 1 到頂點 2 長度為 1 的路徑有 1 條。

步驟二: $A^2 = A^1 A^1$

$=$

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

$=$

	1	2	3	4	5
1	0	0	0	2	2
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

其中$A^2[1][4]=2$ 表示頂點 1 到頂點 4 長度為 2 的路徑有 2 條。

即為: $1 \longrightarrow 2 \longrightarrow 4$

$1 \longrightarrow 3 \longrightarrow 4$

步驟三: 有 5 個頂點為 5 維矩陣，所以n=5，依此類推求出$A^3 = A^2 A^1$，$A^4 = A^3 A^1$，$A^5 = A^4 A^1$。

步驟四: $P = \sum_{n=1}^{5} A^n[i][j]$。

$=$

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

$+$

	1	2	3	4	5
1	0	0	0	2	2
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	0
5	0	0	0	0	0

+

	1	2	3	4	5
1	0	0	0	0	2
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

+

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

+

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

=

	1	2	3	4	5
1	0	1	1	2	4
2	0	0	0	1	2
3	0	0	0	1	2
4	0	0	0	0	1
5	0	0	0	0	0

其中 *P*[1][5]=4 表示頂點 1 到頂點 5 長度小於等於 5 之路徑總共有 4 條。

步驟五: 令 *P* 矩陣中所有 *P*[i][j] >0 的項目都為 1。

TC=

	1	2	3	4	5
1	0	1	1	1	1
2	0	0	0	1	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

利用上述步驟即可求得其對應的 Transitive Closure。

- W^1[i][j]: 相鄰權重矩陣元素。相鄰權重矩陣指的是建立一個矩陣，利用這個矩陣可以記錄圖形G中相鄰的頂點 i與j其長度為 1 的路徑之權重值。在相鄰權重矩陣當中存在 0 與大於 0 二種數值，若W^1[i][j]=0 表示i與j二個頂點不相鄰，若W^1[i][j] >0 表示i與j這二個相鄰頂點間長度為 1 的路徑之權重數值。

EX: 以成對頂點之間路徑長度為 1 之路徑，其權重值建立相鄰權重矩陣。

W^1=

	1	2	3	4	5
1	0	5/24	3/24	0	0
2	0	0	0	1/24	3/24
3	0	0	0	2/24	8/24
4	0	0	0	0	2/24
5	0	0	0	0	0

其中W^1[1][2]=5 表示從頂點 1 到達相鄰的頂點 2 長度為 1 的路徑，其權重值為 5/24。

- F_{in}[j]: F_{in}矩陣內記錄進入頂點 j edge weight的總和，i為起始端，j代表終止端。$F_{in}[j] = \sum_{e_{ij} \in E} W^1[i][j]$。

EX: 利用相鄰權重矩陣，計算出進入頂點 j edge weight 的總和，j$\in V$。

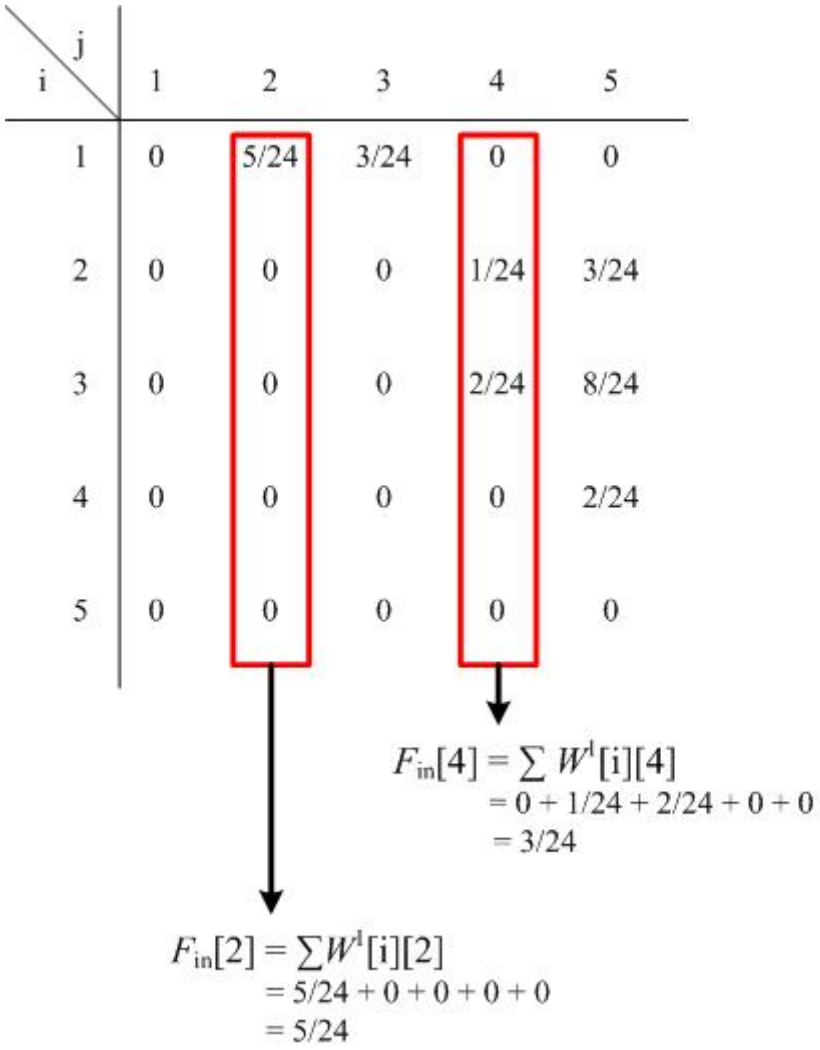


圖 12 計算F_{in}[j]

- F_{out}[i]: F_{out}矩陣內記錄相依於頂點 i edge weight的總和，i為起始端，j代表終止端。$F_{out}[i] = \sum_{e_{ij} \in E} W^1[i][j]$。

EX: 利用相鄰權重矩陣，計算出相依於頂點 edge weight 的總和，i∈V。

i \ j	1	2	3	4	5
1	0	5/24	3/24	0	0
2	0	0	0	1/24	3/24
3	0	0	0	2/24	8/24
4	0	0	0	0	2/24
5	0	0	0	0	0

$F_{out}[1] = \sum W^1[1][j]$
$= 0 + 5/24 + 3/24 + 0 + 0$
$= 8/24$

$F_{out}[3] = \sum W^1[3][j]$
$= 0 + 0 + 0 + 2/24 + 8/24$
$= 10/24$

圖 13 計算F_{out}[i]

- r[i]: 為資料項 i 在廣播序列中的位置。
- D_{ij} : 為收取完所需資料項i與j 所花費的時間，$D_{ij} = r[j] - r[i]$。

EX:

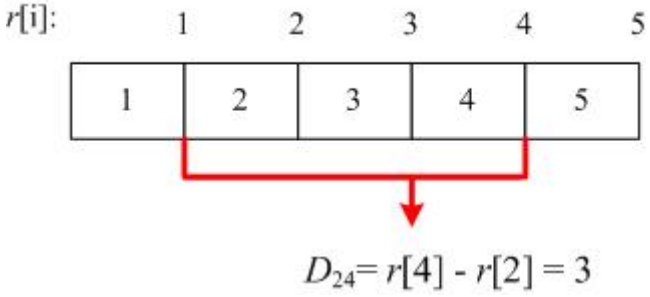


圖 14 資料項廣播序列距離

收取完所需資料項 2 與 4 所花費的時間為三個單位。

- *Cost*[i]: 移動資料項 i 可降低的成本。

將我們所要探討的廣播排程問題，轉換成圖形來處理，舉例說明如下。

EX:

假設有$q_1(d_1, d_2)$，$q_2(d_1, d_3)$，$q_3(d_2, d_4)$，$q_4(d_2, d_5)$，$q_5(d_3, d_4, d_5)$，$q_6(d_5, d_3)$等多個要求多資料項的查詢，其中q_5要求的資料項分別為d_3, d_4, d_5，拆解為d_3, d_4與d_4, d_5，存取頻率分別為f_{12}=5，f_{13}=3，f_{24}=2，f_{25}=3，f_{34}=2，f_{53}=1，f_{45}=8。在$q_k(d_i, d_j)$裡，d_i為查詢k第一個要求的資料項，d_j為查詢k第二個要求的資料項。而f_{ij}為存取完資料項i接著要存取資料項j的頻率，將它記錄在相鄰權重矩陣W^1內，以W^1[i][j]來表示，也就是圖形中的node i與j之間有向邊的權重值。將前述多個查詢作轉換，以一個有向圖形來表示，如圖15(a)所示:

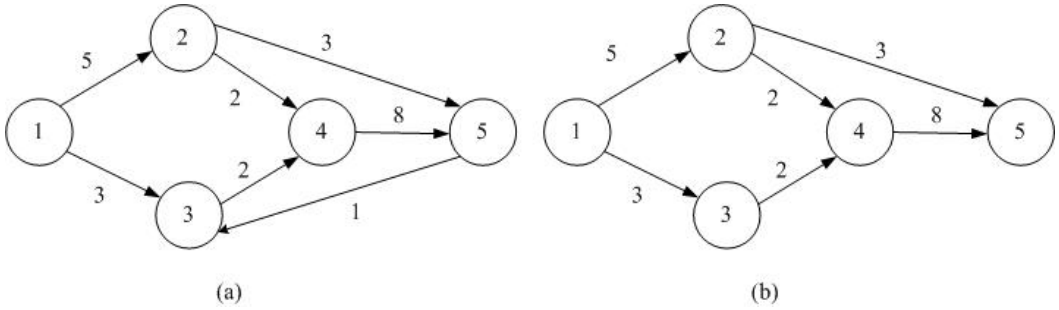


圖 15 有向圖形範例

一個有向圖形裡，可能包含著一些有向循環，在某些應用當中，這樣的有向循環是不良的，例如: 作業系統中，有多個程序循環式的等候系統資源形成死結(deadlock)，多個用於解決死結方法之一，就是以優先權的方式來消除死結，選擇代價最小的行程，作為取消使用資源資格的犧

牲者。藉由消除有向循環，使得原圖形成為一個有向無循環圖形(DAG)。

消除有向循環最簡單的方式，就是丟棄會形成循環的邊來打破循環。反饋弧形集合(Feedback arc set，FAS)是一個邊的集合，存在圖形中的每一個循環，至少有一個邊被FAS所包含，這個集合內的邊從圖形中被移除之後，會使得原圖形成為一個有向無循環圖形。圖 15(a)裡，連接著頂點 3、4、5 的三個有向邊分別為e_{34}、e_{45}、e_{53}，消除權重值最小的e_{53}，以一個有向無循環圖如圖 15(b)所示，表示資料項廣播的先後順序限制。

4.2 Greedy 之拓樸排序演算法

對於一個有向無循環圖(Directed acyclic graph，DAG) $G=(V,E)$而言，拓樸排序(Topological sort)是一個對頂點集合 V 的排序，其中若(u,v)屬於 E 的話，表示在排序中，頂點 u 必須排在 v 之前。常被使用於決定排程。

Ma[23]等學者藉矩陣相乘的方式求得圖形之遞移封閉性矩陣(Transitive Closure)，以其作為設計基礎，發展出拓樸排序演算法;使用拓樸排序及分支度的特性，可以找出一組所有端點的有向邊它的方向皆會由左邊向右邊指之拓撲排列順序，來解決有向之拓樸排序演算法問題。

使用圖形之遞移封閉性矩陣與相鄰權重矩陣來進行演算流程，我們將拓樸排序演算法應用於廣播排程問題解決的演算流程如下:

Begin

1. For each i∈V Set *degree*[i]=0
2. For each $e_{ji} \in E$ Increase *degree*[i] by 1
3. For k=0 to n-1do
4. If there is no i which *degree*[i]=0 then
5. Stop
6. Otherwise
7. Pick i which *degree*[i]=0
8. For each edge e_{ij}, Decrease *degree*[j] by 1

計算所有頂點的 *degree* 值之後;首先挑選出 *degree*[i]為 0 的頂點 i 放入廣播排程序列中，若有 2 個頂點 *degree* 一樣是為 0 的，就任意選擇一個頂點將它排入序列。假設被選出排入序列的頂點為 i，將相依於頂點 i之所有e_{ij}消除，$e_{ij} \in E$，並重新計算*degree*[j]，j∈*V*且頂點j相依於頂點 i。重覆上述步驟直到所有頂點皆被排入廣播序列當中。

EX:

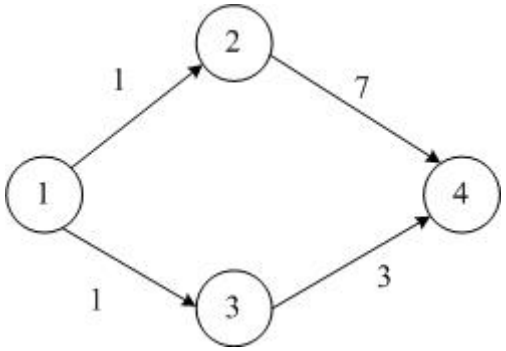


圖 16 有向無循環圖形範例二

以下為使用拓樸排序演算法於圖 16 這個例子當中之排序過程，我們

知道拓樸排序之結果並非是唯一的解，因此經以下拓樸排序步驟後所獲得的結果為多組解其中之一。

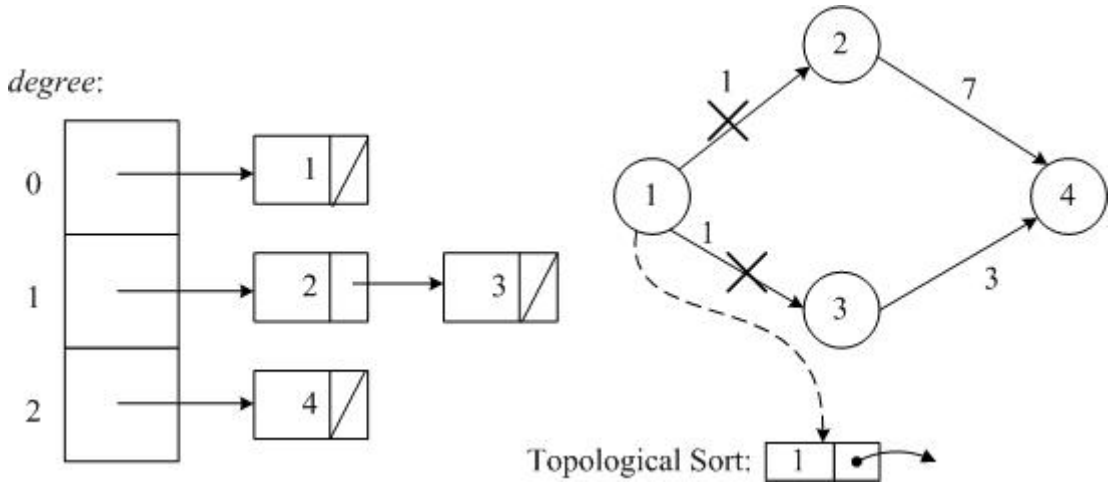


圖 17 應用拓樸排序於圖 16 範例步驟一

步驟一: 取出 *degree* 為 0 的頂點 1 排入廣播序列中，並消除相依於頂點 1 的所有有向邊，而相依於頂點 1 的有頂點 2 與頂點 3，則 *degree*[2]減掉 1 之後為 0，*degree*[3]減掉 1 之後為 0。

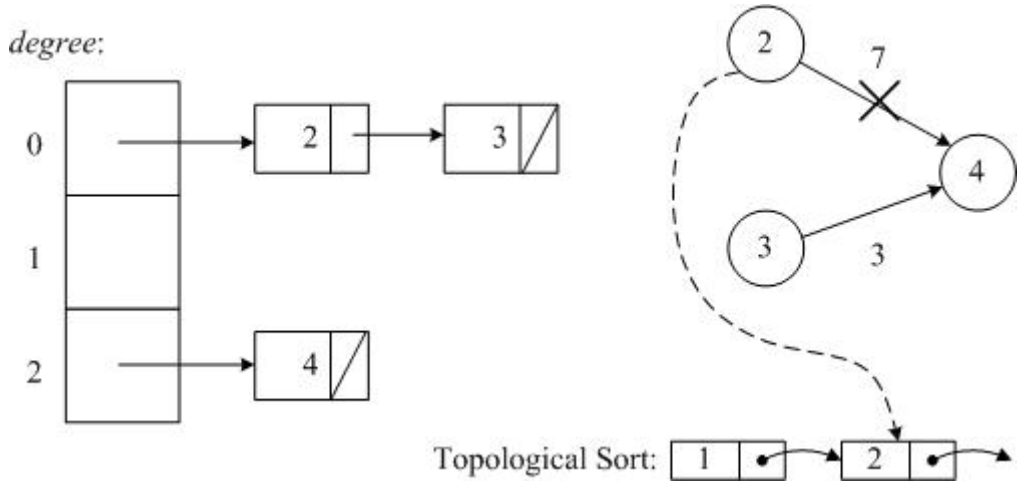


圖 18 應用拓樸排序於圖 16 範例步驟二

步驟二: 取出 *degree* 為 0 的頂點排入廣播序列中,在這裡出現二個 *degree* 同為 0 的頂點，分別是頂點 2 與頂點 3，則任意選擇一個排入廣播序列中，我們選擇取出頂點 2 並消除相依於頂點 2 的所有有向邊，*degree*[4]減掉 1 之後為 1。

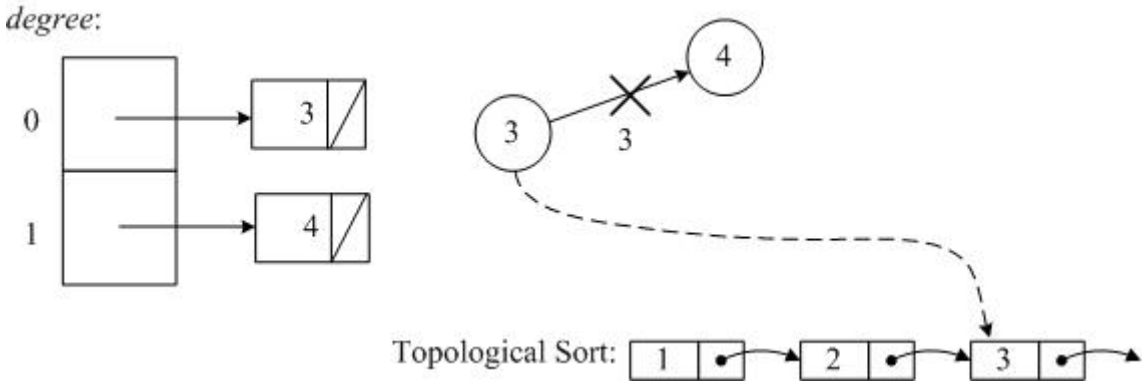


圖 19 應用拓樸排序於圖 16 範例步驟三

步驟三: 取出 *degree* 為 0 的頂點 3 排入廣播序列中，並消除相依於頂點 3 的所有有向邊，*degree*[4]減掉 1 之後為 0。

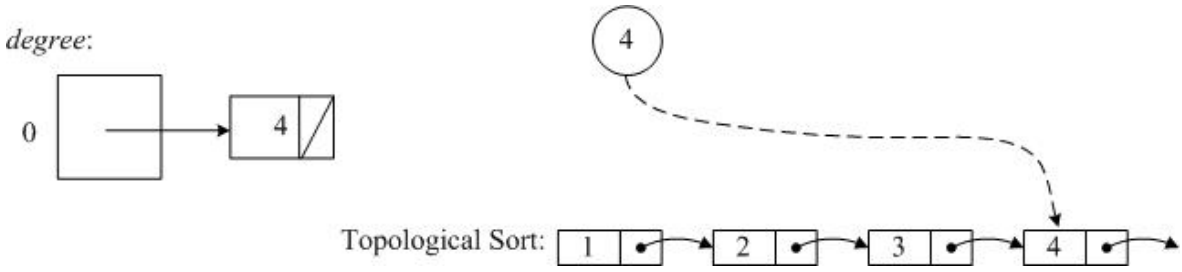


圖 20 應用拓樸排序於圖 16 範例步驟四

步驟四: 取出 *degree* 為 0 的 node 4 排入廣播序列中。

步驟五: 此時所有頂點皆被排入廣播序列中，停止演算步驟流程。得到一組廣播序列如圖 21 所示:

圖 21 應用拓樸排序於圖 16 範例結果

我們知道在同時有 2 個或以上無前行者的頂點時，進行拓樸排序產生的序列並不是唯一的解，以圖 16 為例，利用拓樸排序可得到多組廣播序列，分別為 1-2-3-4 與 1-3-2-4，序列 1-3-2-4 的資料項平均相依長度比序列1-2-3-4低,因此以降低用戶端平均查詢讀取時間為考量,序列1-3-2-4優於序列 1-2-3-4。

本文提出在拓樸排序中加入Greedy機制的排序演算法，在多個*degree*同為 0 的解中，選取F_{in}較大者為優先排入序列的考量，但若存在多個*degree*為 0 且F_{in}相同的解時，就可能無法找尋到最佳的廣播序列，因此我們反轉原圖形中所有頂點之有向邊，建立preference list，經由在排序過程裡，參考preference list這個對偶(dual)的方法可以在之後的排序中使用，找尋到較低的平均資料相依長度，以降低用戶端平均查詢讀取時間。

建立preference list來解決存在多個*degree*為 0 且F_{in}相同的解時，無法找尋到最佳的廣播序列的問題。首先反轉原圖形中所有頂點之有向邊，原圖形中的起始頂點變成是經反轉後圖形中的終止頂點，而原圖形中的終止頂點變成是經反轉後圖形中的起始頂點，使用加入Greedy機制之拓

樸排序演算法，在多個*degree*同為 0 的解中，選取F_{in}較大者優先排入序列中，直到所有頂點皆被排入序列。將得到的序列結果作反轉獲得preference list，以提供進行排序過程中參考用。

利用 4.1 節裡所描述的方式，求出有向無循環範例圖 16 的遞移矩陣*TC*，利用遞移矩陣*TC*計算出所有頂點之*degree*。進行Greedy方式的拓樸排序首先挑選取出*degree*為 0 的頂點，在選擇的過程中，若有多個頂點其*degree*之值均為 0，選擇出F_{in}最大的頂點，將之排入廣播序列中。在有多個*degree*均為 0 且F_{in}相等的解時，就可能無法找尋到最佳的廣播序列，因此我們參考反轉原圖形中所有頂點之有向邊，所建立的preference list，以求得較佳的廣播序列。

每選出一個頂點排入序列中，就必需重新執行計算，更新所有相依於它的頂點之 *degree*。假設被選出排入序列的頂點為 i，將相依於頂點 i之所有e_{ij}消除，$e_{ij} \in E$，並重新計算*degree*[j]，$j \in V$且頂點j相依於頂點 i。重覆上述步驟直到所有頂點皆被排入廣播序列當中。重新整理演算法之演算流程如下:

Begin

1. For each i$\in V$ Set *degree*[i]=0
2. For each $e_{ji} \in E$ Increase *degree*[i] by 1
3. For k=0 to n-1do
4. If there is no i from *degree*[i]=0 then
5. Stop
6. Otherwise
7. If exist only one vertex i with largest F_{in}[i] from *degree*[i]=0
8. output i
9. Else (more than one vextex i, m ,…with same largest F_{in}[i]=F_{in}[m]=…)
10. output vextex according from preference list
11. For each edge e_{ij}, Decrease *degree*[j] by 1

加入 Greedy 方式之拓樸排序演算法，演算流程圖如下:

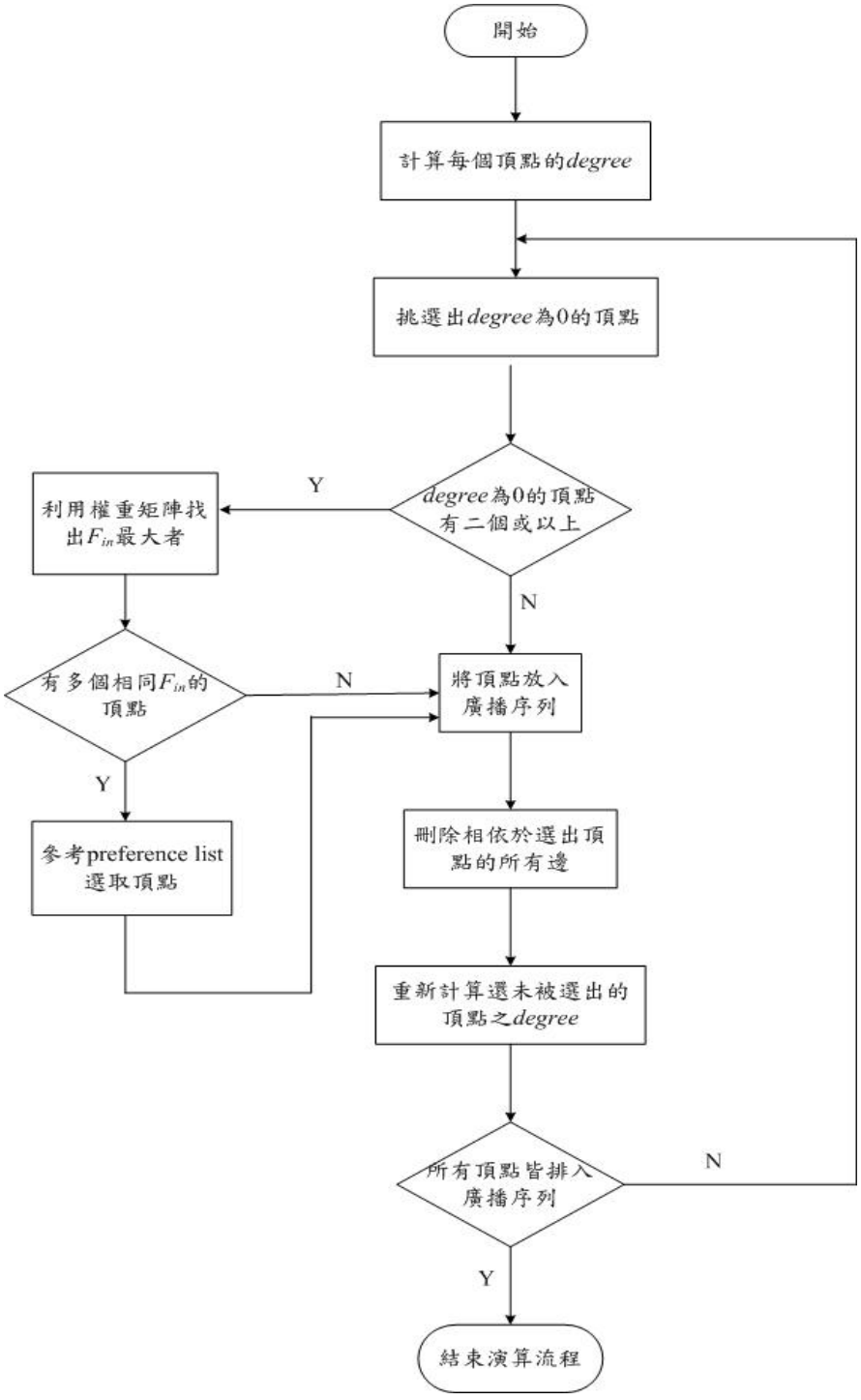


圖 22 加入 Greedy 方式於拓樸排序之演算流程圖

EX:

以圖 16 為例，用以說明加入 Greedy 方式的拓樸排序演算法，應用於廣播排程問題解決的演算流程。首先反轉原圖形中所有頂點之有向邊，以加入 Greedy 方式的拓樸排序演算法，建立 preference list:

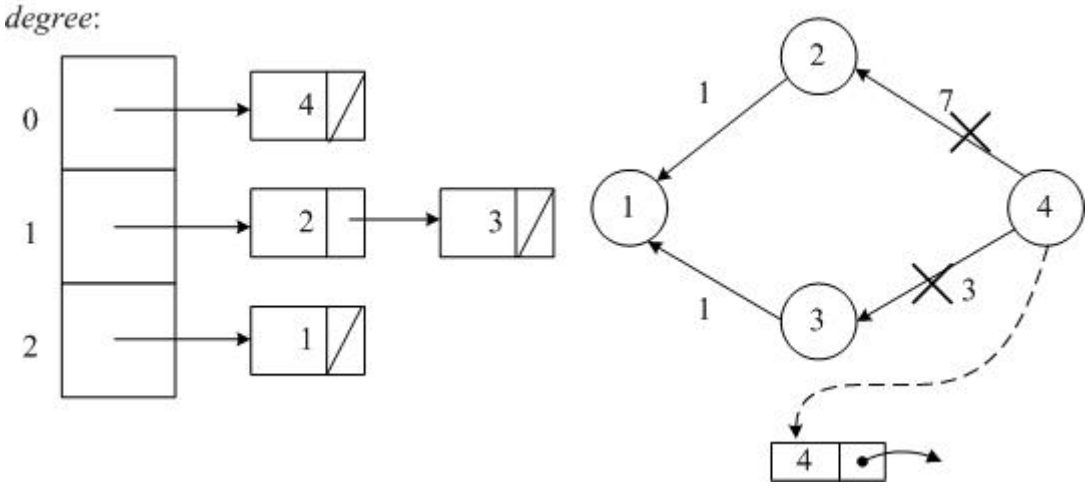


圖 23 建立圖 16 範例之 preference list 步驟一

步驟一: 反轉原圖形中所有頂點之有向邊，取出 *degree* 為 0 的頂點 4 排入序列中，並消除相依於頂點 4 的所有有向邊，而相依於頂點 4 的有頂點 2 與頂點 3，則 *degree*[2]減掉 1 之後為 0，*degree*[3]減掉 1 之後為 0。

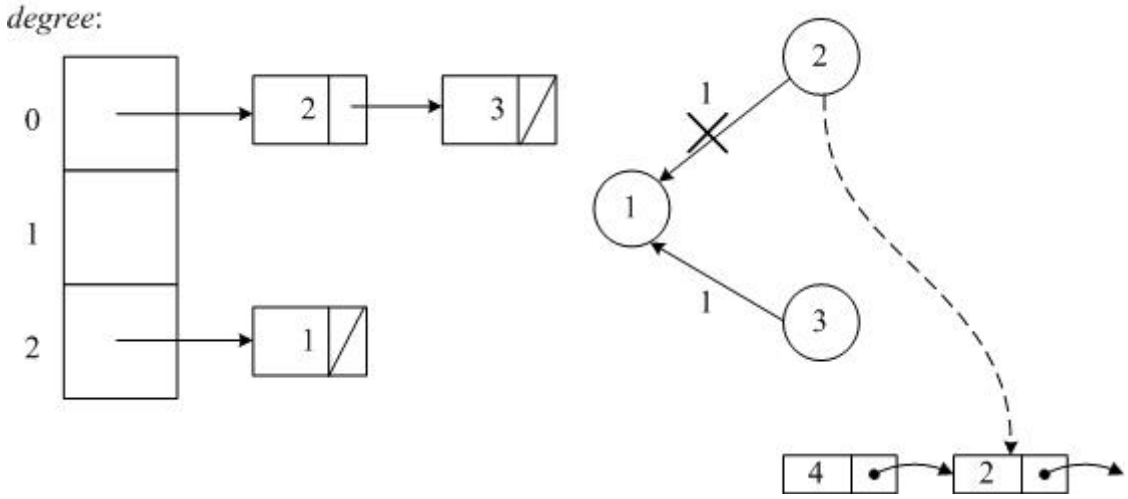


圖 24 建立圖 16 範例之 preference list 步驟二

步驟二: 取出*degree*為 0 的頂點排入序列中，在這裡出現二個*degree*同為 0 的頂點，分別是頂點 2 與頂點 3，則選擇F_{in}最大者排入 preference list中，F_{in}[2]=7，F_{in}[3]=3，因此我們選擇取出頂點 2 並消除相依於頂點 2 的所有有向邊，*degree*[1]減掉 1 之後為 1。

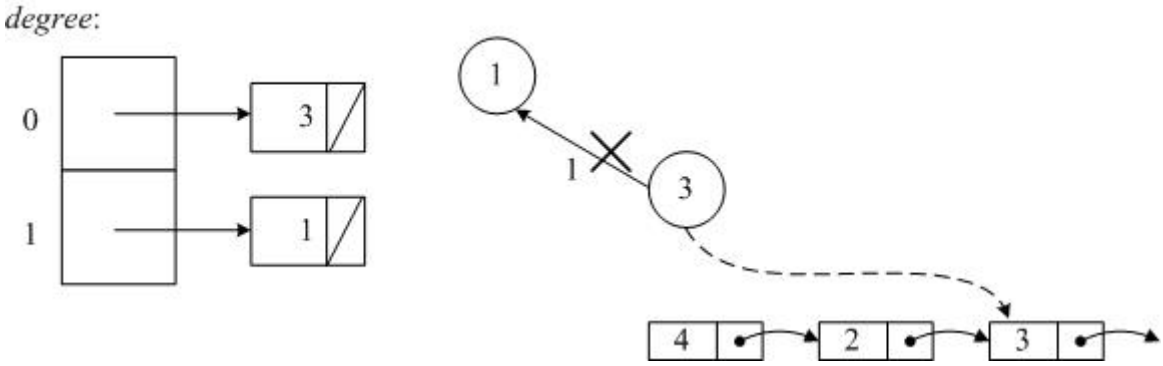


圖 25 建立圖 16 範例之 preference list 步驟三

步驟三: 取出 *degree* 為 0 的頂點 3 排入序列中，並消除相依於頂點 3 的所有有向邊，而相依於頂點 3 的有頂點 1，則 *degree*[1]減掉 1 之後為 0。

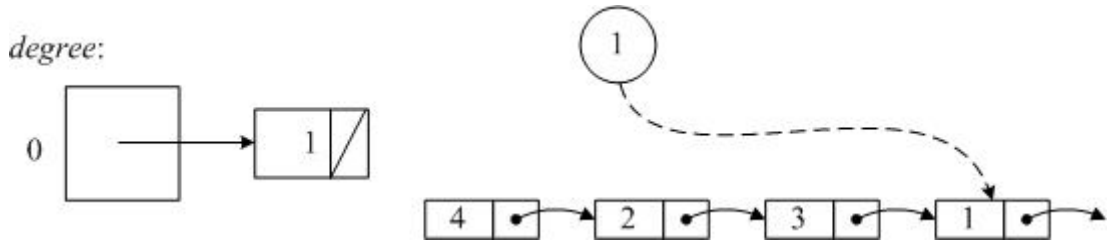


圖 26 建立圖 16 範例之 preference list 步驟四

步驟四: 取出 *degree* 為 0 的頂點 1 排入序列，此時所有頂點皆排入序列中。

使用上述步驟求出序列後反轉，得到 preference list 如下圖 27 所示，提供應用加入 Greedy 方式之拓樸排序於廣播排程問題排序過程參考。

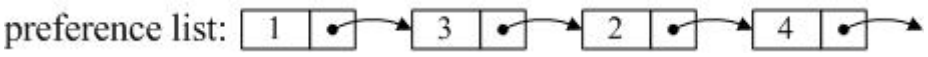


圖 27 圖 16 範例之 preference list

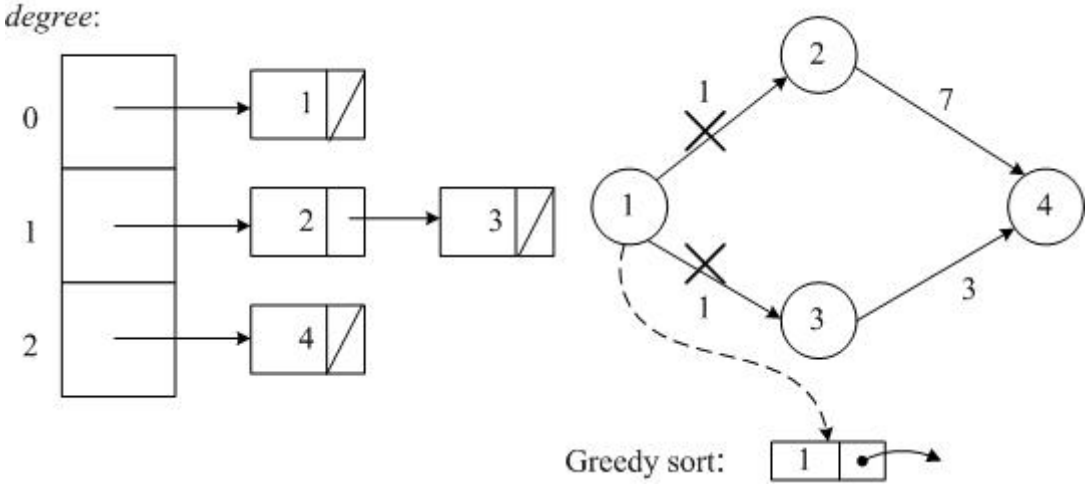


圖 28 應用 Greedy 拓樸排序於圖 16 範例步驟一

步驟一: 取出 *degree* 為 0 的頂點 1 排入廣播序列中，並消除相依於頂點 1 的所有有向邊，而相依於頂點 1 的有頂點 2 與頂點 3，則 *degree*[2] 減掉 1 之後為 0，*degree*[3]減掉 1 之後為 0。

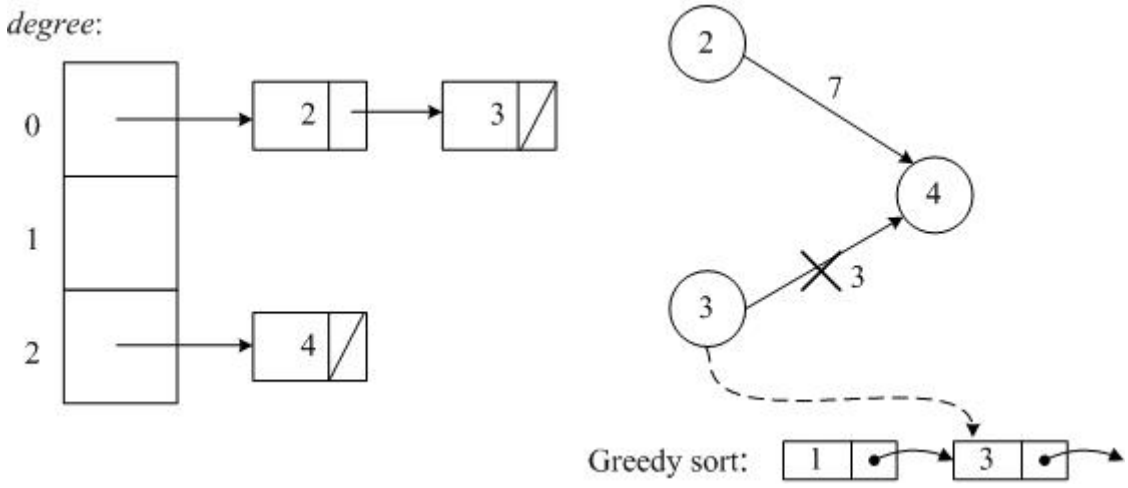


圖 29 應用 Greedy 拓樸排序於圖 16 範例步驟二

步驟二: 取出*degree*為 0 的頂點排入廣播序列中，在這裡出現二個*degree*

同為 0 的頂點，分別是頂點 2 與頂點 3，則選擇F_{in}最大者排入廣播序列中，但F_{in}[2]=1，F_{in}[3]=1，二者F_{in}皆相同，因此我們參考preference list，頂點 1 之後是接著頂點 3，所以選擇取出頂點 3，並消除相依於頂點 3 的所有有向邊，*degree*[4]減掉 1 之後為 1。

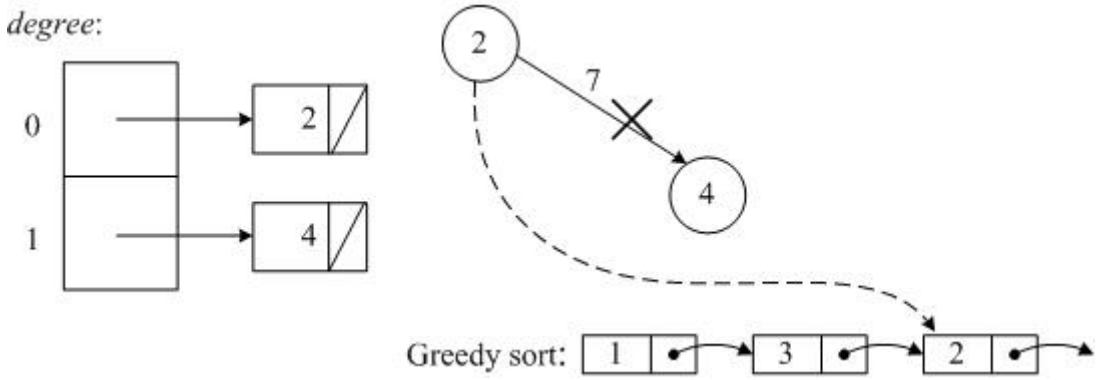


圖 30 應用 Greedy 拓樸排序於圖 16 範例步驟三

步驟三: 取出 *degree* 為 0 的頂點 2 排入廣播序列中，並消除相依於頂點 2 的所有有向邊，*degree*[4]減掉 1 之後為 0。

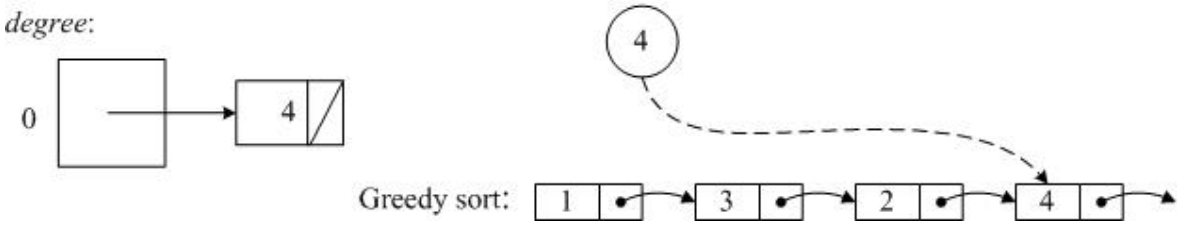


圖 31 應用 Greedy 拓樸排序於圖 16 範例步驟四

步驟四: 取出 *degree* 為 0 的頂點 4 排入廣播序列中。

步驟五: 此時所有頂點皆被排入廣播序列中，停止演算步驟流程。得到一組廣播序列如圖 32 所示:

圖 32 應用 Greedy 拓樸排序於圖 16 範例結果

4.3 KL 最佳化演算法

先由根據限制條件的一組資料項序列開始，應用 KL 演算法來找尋較低的平均資料項相依長度，以求降低用戶端平均查詢讀取時間。

KL 演算法[20]是一種圖形分割之最佳化演算法，它的基本原理試圖在分割的二個子部份中，同時滿足每個子部份的資料項數量皆不少於一個最少資料項數的條件之下，重覆進行資料項的廣播位置交換動作，期望二個子部份資料項之間的相依性越小越好，而使用 KL 演算法於問題上的解決，只會移動能使成本減少最多的資料項，藉由逐步分割來修飾結果。

首先根據拓樸排序演算法求得一組資料項廣播序列，將其分割成為集合 L 與集合 R 二個子圖;其中 $L \cap R = \emptyset$ and $L \cup R = V$。所有資料項個數 $|V| = l$。以第 $\left\lfloor \frac{l}{2} \right\rfloor$+1 個 node 為分割二個子圖之中心點;集合 L 的資料項個數為 $|L| = \left\lfloor \frac{l}{2} \right\rfloor$+1，集合 R 的資料項個數為 l - $|L|$。

從集合 L 與集合 R 中參考遞移封閉性矩陣(Transitive Closure)，在每

一個集合的資料項數量，皆不少於一個最少資料項數的條件下，選擇出模擬移動廣播位置的資料項。移動廣播位置指的是，屬於集合 *L* 的資料項由集合 *L* 移動至集合 *R*，屬於集合 *R* 的資料項由集合 *R* 移動至集合 *L*。在模擬移動的範圍中，透過在 *L* 與 *R* 二個集合之間資料項的移動，可使得二集合間資料項的相依性降低。

其中最少資料項數是由每一集合資料項個數乘以 *prop* 所獲得，其中 *prop* 為一自訂值，用以確保不會因為移動資料項的關係，使得二個集合之間的資料項個數失衡。

每移動一次資料項，就要更新|*L*|數值。若資料項是從集合 *L* 移動到集合 *R*，則|*L*| = |*L*| - 1，反之資料項是從集合 *R* 移動到集合 *L*，則|*L*| = |*L*| + 1。

資料項在被移動後可減少的成本以*Cost*表示，若資料項i屬於*L*集合，則*Cost*[i] = F_{out}[i]- F_{in}[i]。若資料項i屬於*R*集合，則*Cost*[i] = F_{in}[i]- F_{out}[i]。

將被選擇模擬移動廣播位置的資料項，和資料項在真正移動之後屬於集合 *L* 或集合 *R*，以及移動後可減少的成本 *Cost*，等資訊儲存起來。並 *Lock* 被真正移動過的資料項。執行遞迴找尋的動作，直到可減少的成本小於等於 0 為止。

依照模擬移動廣播位置的範圍，移動可減少的成本最大的資料項

opt，其 *Cost* 為:

$$Cost_{opt} = Max \sum_{i=1}^{opt} Cost[i] \qquad \text{(公式 1)}$$

進行 *L* 與 *R* 二集合間資料項真正的廣播位置移動。對集合 *L* 及集合 *R* 中的元素遞迴重覆執行 KL 演算步驟，直到交換二集合的資料項和再分割都無法使得成本減少為止。

當分割集合使得內部資料項個數小於或等於 2 時，呼叫 *Change* 副程式，進行成對資料項之間的廣播位置交換，直到再也無法使得成本減少為止，停止演算流程輸出序列結果。使用 KL 演算法來進行廣播資料排程步驟表示如下:

Kernighan-Lin Algorithm

輸入: 由圖形 $G=(V,E)$經拓樸排序產生之序列，序列長度$|V|=l$，序列中的第一個 node 為 *S*，序列的最後一個 node 為 *T*。

輸出: *L* 與 *R* 二集合間資料項的相依性最小化

Class KL(l , *S*, *T*)

1. Begin
   If $l>2$
2. Partition *V* into *L* and *R*，Such that $|L| = |R|$，$L \cap R =$ null and $L \cup R = V$
3. Repeat
4. While there are unlocked vertices
5. If $|L|$ or $|R|$ are not < $(\left\lfloor \frac{l}{2} \right\rfloor + 1) * prop$，find unlocked vertex $V_i \in L$

or $V_i \in R$，maximizing *Cost*[i]

6. Update |*L*| value，if find vertex $V_i \in L$ than |*L*| = |*L*| - 1，if find vertex $V_i \in R$ than |*L*| = |*L*| + 1
7. Store V_i and *Cost*[i]
8. Lock V_i
9. All vertices unlock，as though V_i had been move
10. End while
11. Pick *opt* such that *Cost* is maximized， $Cost_{opt} = Max \sum_{i=1}^{opt} Cost[i]$
12. If $Cost_{opt} > 0$ then Call KL(*l* , *S*, *T*)
13. Until $Cost_{opt} <= 0$
14. ELSE if *l*==2
15. *Change* (*S*, *T*)
16. End

Class *Change*(*S*, *T*)

1. If *Cost*[*S*] >*Cost*[*T*] && V_T are not V_S**'s** successor than
   a = *r*[*S*], *r*[*S*] = *r*[*T*], *r*[*T*] = a
2. End

EX:

以下列舉一個例子，用以說明 KL 演算法應用於廣播排程問題解決的演算流程。某一拓樸排序後的串列如下:

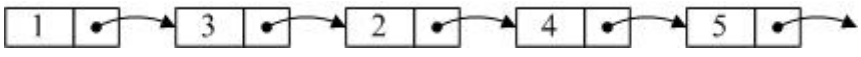


圖 33 拓樸序列範例一

其遞移矩陣*TC*與權重矩陣W^1如下所示:

TC=

	1	2	3	4	5
1	0	1	1	1	1
2	0	0	0	1	1
3	0	0	0	1	1
4	0	0	0	0	1
5	0	0	0	0	0

W^1=

i \ j	1	2	3	4	5
1	0	5/23	3/23	0	0
2	0	0	0	1/23	2/23
3	0	0	0	2/23	8/23
4	0	0	0	0	2/23
5	0	0	0	0	0

令集合 L 資料項個數$|L| = \left\lfloor \frac{l}{2} \right\rfloor$+1=2+1=3，集合 R 資料項個數$|R| = l - |L|$ =5-3=2

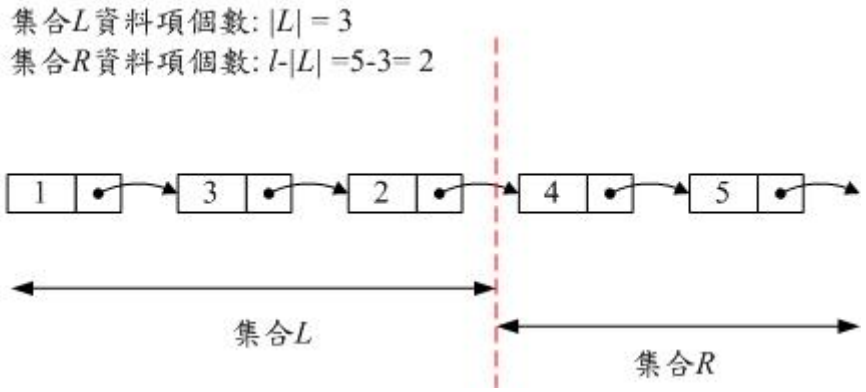


圖 34 應用 KL 演算法於圖 33 範例步驟一

步驟一: 將序列分割成為集合L與集合R二個子集合如圖 34 所示，集合L之資料項個數$|L| = \left\lfloor \frac{l}{2} \right\rfloor$+1=3，集合$R$之資料項個數$|R| = l - |L|$=2，設定最少資料項個數為 1。由集合$L$中選擇出第一個資料項$d_1$，從遞移封閉性矩陣中可看出，在廣播序列中必需排在$d_1$之後的$d_2$、$d_3$、$d_4$、$d_5$，其中$d_2$、$d_3$與$d_1$一樣同在集合$L$內，故此時它是不能被模擬從集合$L$移動到$R$，故不移動它並循序往後選擇其它

可行資料項。

步驟二: 循序往後選擇集合L的第二個資料項d_3，從遞移封閉性矩陣看出，必需排在d_3之後的d_4、d_5，並沒有與d_3同在L集合內，接著計算d_3的*Cost*，*Cost* [3] = F_{out}[3]- F_{in}[3] = (2/24+8/24)-(3/24) = 7/24>0，若移動資料項 3，$|L|=|L|-1=3-1=2$，沒有小於最少資料項數 1，將d_3加入試圖移動的選擇中。

步驟三: 循序往後選擇集合L的第三個資料項d_2，從遞移封閉性矩陣可知，必需排在d_2之後的d_4、d_5，並沒有與d_2同在L集合內，接著計算d_2的*Cost*，*Cost* [2] = F_{out}[2]- F_{in}[2] = (1/24+2/24)-(5/24) = -2/24< 0，若移動d_2則$|L|=|L|-1=2-1=1$，沒有小於最少資料項數 1，將d_2加入試圖搬動的選擇中。

步驟四: 選擇集合R的第一個資料項d_4，從遞移封閉性矩陣得知，必需排在d_4之前的d_1、d_2、d_3，並沒有與d_4同在R集合內，接著計算d_4的*Cost*，*Cost* [4] = F_{in}[4]- F_{out}[4] = (1/24+2/24)-(2/24) = 1/24> 0，若移動d_4則$|L|=|L|+1=1+1=2$，沒有小於最少資料項數 1，將d_4加入試圖搬動的選擇中。

步驟五: 選擇集合R的第二個資料項d_5，從遞移封閉性矩陣得知，必需排在d_5之前的d_4與它同在集合R內，因此d_5是不可以被模擬從集合R

移動到L。故不移動它並循序往後選擇其它可行資料項。

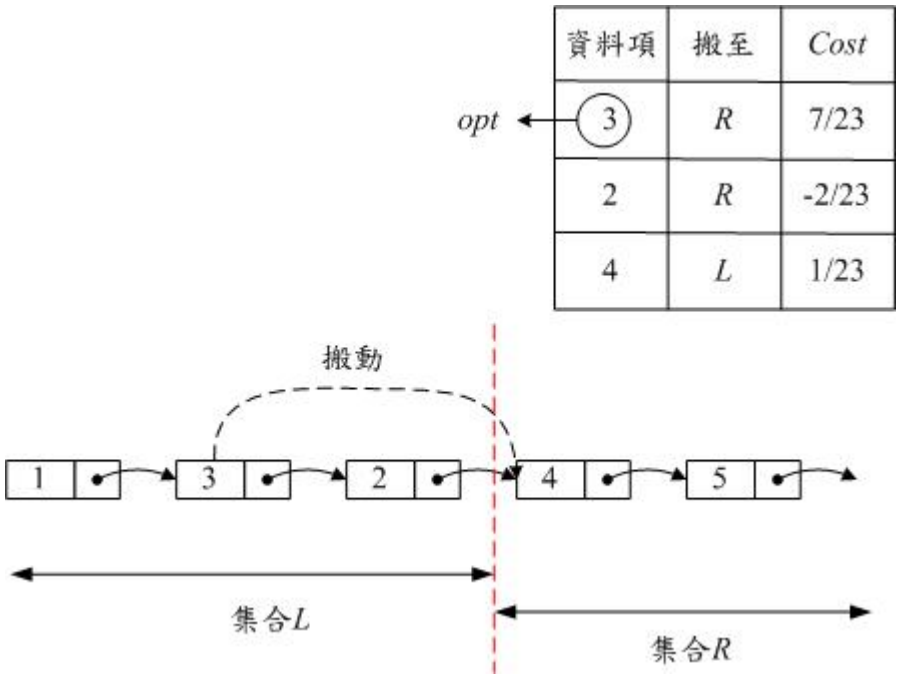


圖 35 應用 KL 演算法於圖 33 範例步驟六

步驟六: 所有的資料項都檢查完畢，由公式 1 可看出陣列內移動後可減少成本最大的 *opt* 為 1，也就是資料項 3，因此我們只將原序列中的資料項 3 作移動。因為資料項 3 是從集合 L 移動到集合 R，所以更新集合 L 的資料項個數為$|L|= |L|-1=3 - 1= 2$，集合 R 的資料項個數為$(l - |L|) = 5-2 =3$。得到如圖 36 所示新的廣播序列。

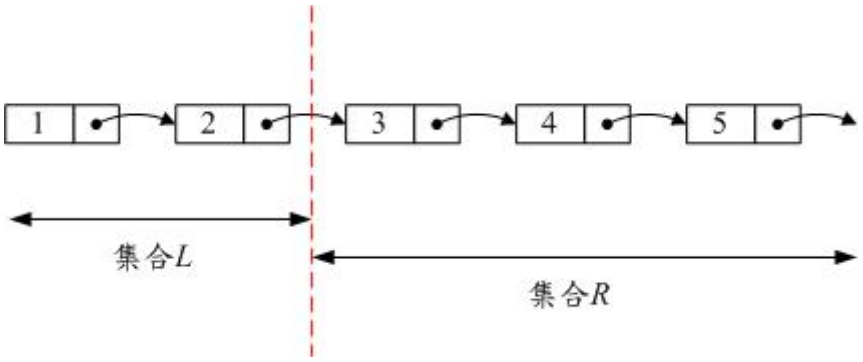


圖 36 應用 KL 演算法於圖 33 範例步驟六搬動結果

步驟七: 將集合L分割成L_1與R_1，集合R分割成L_2與R_2，如圖 37 所示，重覆執行KL模擬資料項移動過程，用以求得下一個最佳的可搬動資料項。

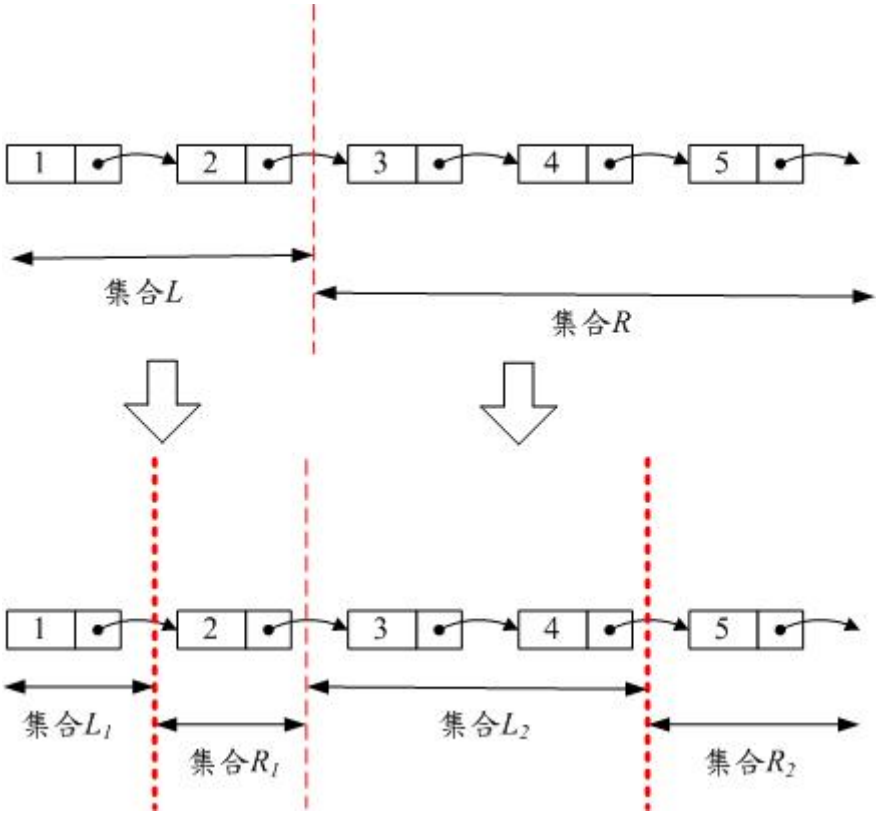


圖 37 KL 演算法之集合分割

步驟八: 當每個子集合裡的資料項個數皆小於等於 2 時，停止 KL 演算執行，呼叫 *Change* 副程式，進行成對資料項廣播位置交換之後，停止演算執行。

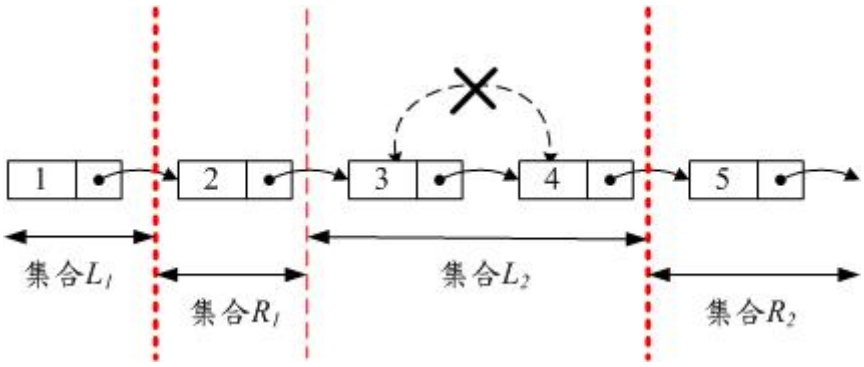


圖 38 應用 KL 演算法於圖 33 範例步驟九

步驟九: 從遞移封閉性矩陣得知，d_3是d_4的前行者，所以不能進行互換，則放棄d_3與d_4的廣播位置交換，結束演算流程並輸出序列結果。

4.4 模擬退火(SA)演算法

SA(Simulated Annealing)模擬退火演算法[21]是一種模擬金屬物體在加熱至一定溫度後，隨時間增加慢慢冷卻，其所有的分子逐漸停留在不同狀態分佈之最佳化演算法，跟前面我們介紹的 KL 演算法類似，應用在我們的廣播排程問題上，它也是評估可行性之後，嘗試進行資料項廣播位置搬動變換，使得廣播序列結果可達到最佳化的一種演算法。

SA 演算法與 KL 演算法相同，是透過搬動資料項的廣播位置，試圖

降低用戶端的平均查詢讀取時間;不同的是 KL 演算法會選擇能使得用戶端平均查詢讀取時間，降低最多的部份來當作變動交換的可能性，而 SA 演算法在一個機率之下，可容許挑選到的資料項在變動交換之後，用戶端平均查詢讀取時間並不一定會降低的情況，主要是採用機率來決定接受與否，利用機率去探索所有可能的答案，當機率大於隨機亂數時，就會繼續向下搜尋問題的新解;藉由 SA 演算法來解決廣播排程的問題，將挑選出的資料項其廣播位置作變換，有可能會使得結果較為略差一些，但有時反而能把結果帶入另一區域的最佳解，或是整體的最佳解。SA 可避免侷限在區域的最佳解中，但其缺點為執行時間會較長些。

在開始使用SA演算法我們將採用較大的T_0值，讓問題的解其活動範圍較廣一些，不易落入局部最佳解當中，隨著資料項廣播位置交換次數增加，將逐漸的降低T_0值，讓結果逐步收斂。

輸入伺服端資料庫內欲廣播的資料項集合，經由拓樸排序之後的廣播序列。設定演算法參數: 起始溫度T_0，終止溫度T_N，與執行次數N。隨機產生一個初始解，也就是任意選取二個要進行交換的資料項，且參考遞移矩陣後，是可交換的資料項i與j當作廣播位置交換之可行解，在序列中資料項i的廣播位置在資料項j之前，分別計算這二個資料項的搬動成本 $Cost[i] = F_{out}[i] - F_{in}[i]$，$Cost[j] = F_{out}[j] - F_{in}[j]$。

若交換資料項 i 與 j 的廣播位置可使成本降低且是可交換的，也就是 (*Cost*[i] – *Cost*[j]) > 0 且j不相依於i，i與j在廣播序列中，也就是二者之間相隔的資料項，不是資料項i的後繼者，也不是資料項j的前行者，則將這二個資料項的廣播位置交換。如果資料項i與j的廣播位置是可以交換的，但並不能使得成本降低，也就是(*Cost*[i] – *Cost*[j]) < 0，使用一個機率值判定是否接受較不佳的新解，產生一個 0~1 之間的亂數*X*，計算接受機率 $Y = \exp((\mathit{Cost}[i] - \mathit{Cost}[j]) / T_k)$，*Y* > *X*則將i與j這二個資料項的廣播位置交換，反之若*Y* < *X*就不接受交換。

每執行交換一次，就將起始溫度T_0 乘以冷卻值$Z = (T_N/T_0)^{k/N}$用以降低溫度，並計算時間k的溫度T_k。$T_k = T_0 * Z$。

判斷現在的溫度T_k與終止溫度的關係，若大於終止溫度就另外產生一組新解繼續執行。反之T_k小於終止溫度則停止執行程式。

以下說明本研究對於 SA 模擬退火演算法的設計以及演算過程:

Simulated Annealing Algorithm

Input:

Graph *G*=(*V*,*E*)

Cost function: $\mathit{Cost}[i] = F_{out}[i] - F_{in}[i], i, \in V$

Starting temperature T_0, final temperature T_N,

And the number of iterations N.

Algorithm:

1.Generate a random feasible solution i，j such that |i|=|j|

2. For k=1 to N
3. If $T_k < T_N$ than Stop
4. If ($Cost$[i] – $Cost$[j]) > 0 than
   a = r[i], r[i] = r[j], r[j] = a ;
5. Else
6. $Z=(T_N/T_0)^{k/N}$
7. $T_k=T_0 * Z$
8. X= a random number，0< X <1
9. Y = exp(($Cost$[i] – $Cost$[j]) / T_k)
10. If(X<Y) than a = r[i], r[i] = r[j], r[j] = a ;
11. k++

SA 演算法之演算流程圖如下:

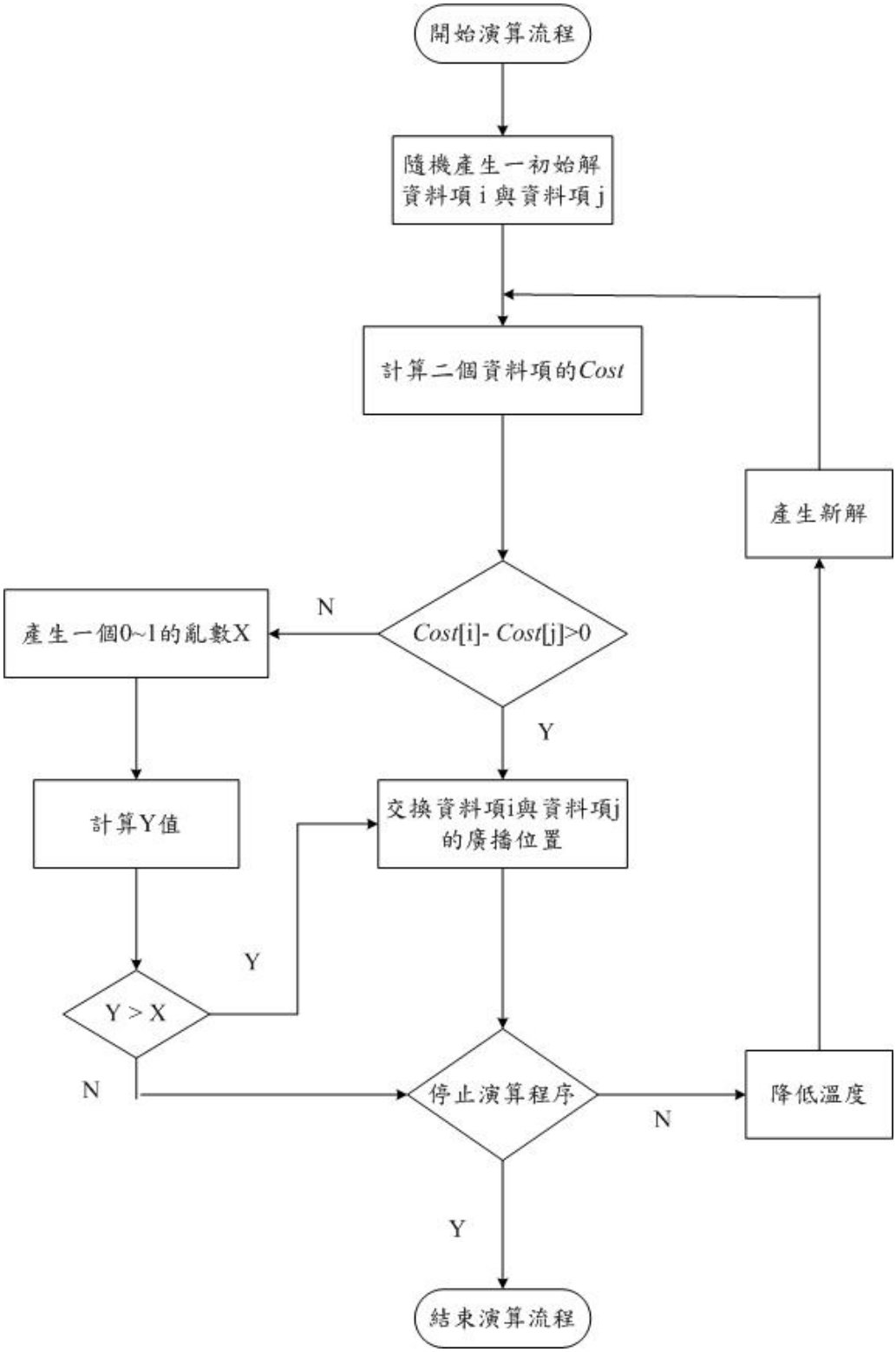


圖 39 SA 演算流程圖

EX:

以下列舉一個例子，用以說明 SA 演算法應用於廣播排程問題解決的演算流程。

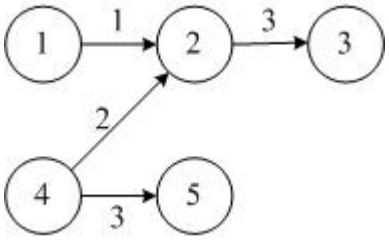


圖 40 有向無循環圖形範例四

有一有向無循環圖形如圖 40，利用 4.1 節描述方式求得遞移封閉性矩陣*TC*與相鄰權重矩陣W^1如下:

***TC*:**

i \ j	1	2	3	4	5
1	0	1	1	0	0
2	0	0	1	0	0
3	0	0	0	0	0
4	0	1	1	0	1
5	0	0	0	0	0

W^1

i \ j	1	2	3	4	5
1	0	1	0	0	0
2	0	0	3	0	0
3	0	0	0	0	0
4	0	2	0	0	3
5	0	0	0	0	0

設定演算法參數: 起始溫度T_0=10，終止溫度T_N=0.2，N=5，某一亂數序列random[1]=0.1，random[2]=0.2…，使用一指標*rp*，在一開始指向亂數序列random[1]，每使用一次亂數序列中的值，指標*rp*往下移動。

應用 SA 演算法於下列拓樸序列:

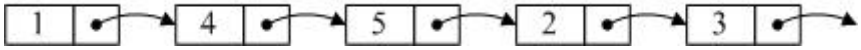


圖 41 拓樸序列範例二

參考遞移權重矩陣W^1內容計算每個資料項之*Cost*如:

Cost[1] = F_{out}[1]- F_{in}[1] = (1/9)- 0 = 1/9
Cost[2] = F_{out}[2]- F_{in}[2] = (3/9)-(1/9+2/9) = 0
Cost[3] = F_{out}[3]- F_{in}[3] = 0 -(3/9) = -3/9
Cost[4] = F_{out}[4]- F_{in}[4] = (2/9+3/9)- 0 = 5/9
Cost[5] = F_{out}[5]- F_{in}[5] = 0 -(3/9) = -3/9

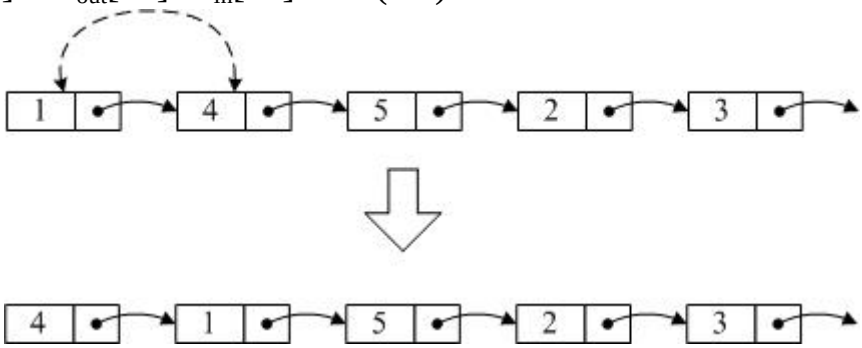


圖 42 應用 SA 演算法於圖 41 範例步驟一

步驟一: 現在的溫度T_1= T_0 *$(T_N/T_0)^{k/N}$=10*$(0.2/10)^{1/5}$=4.57 大於終止溫度T_N=0.2，則隨機選擇二個資料項 1 與 4 進行廣播位置交換，從遞移封閉性矩陣得知，資料項 1 非資料項 4 的前行者，而資料項 4 非資料項 1 的後繼者，計算交換後的*Cost*為*Cost*[1]- *Cost*[4] = 1/9-5/9 = -4/9<0。使用一個機率值判定是否接受這個較不佳的解，亂數序列中的每一個值為一介於 0~1 之間的亂數，以*X*表示，計算接受機率$Y = \exp((\ Cost[1] - Cost[4]\ ) / T_1) = \exp(0.444 /$

4.57)= exp (0.097)= 1.102，$Y > X$則將i與j這二個資料項的廣播位置交換，反之若$Y < X$就不接受交換。在此X = random[1]=0.1，1.102>0.1，$Y > X$，則將 1 與 4 這二個資料項的廣播位置交換。

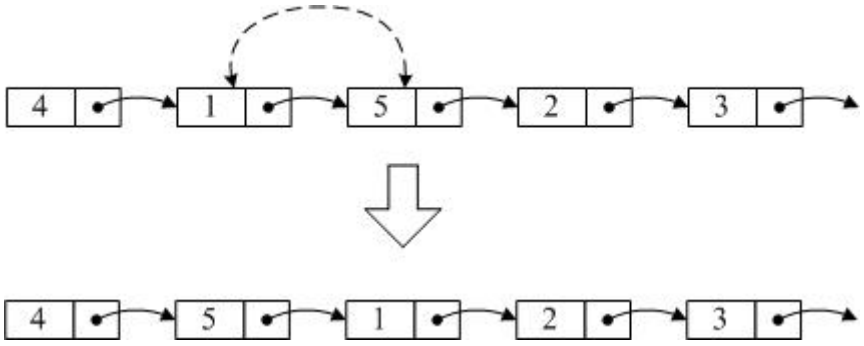


圖 43 應用 SA 演算法於圖 41 範例步驟二

步驟二: 現在的溫度$T_2 = T_0 * (T_N/T_0)^{k/N} = 10*(0.2/10)^{2/5} = 2.09$ 大於終止溫度T_N=0.2，則隨機選擇二個資料項 1 與 5 進行廣播位置交換，從遞移封閉性矩陣得知，資料項 1 非資料項 5 的前行者，而資料項 5 非資料項 1 的後繼者，計算交換後的*Cost*為，*Cost*[1]- *Cost*[5] = 1/9-(-3/9) = 4/9>0，將 1 與 5 這二個資料項的廣播位置交換。

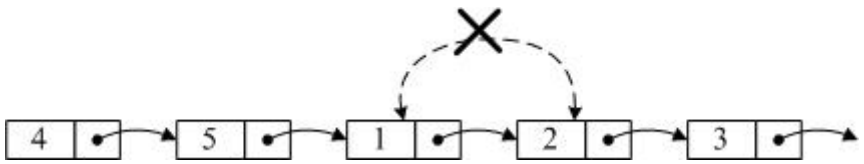


圖 44 應用 SA 演算法於圖 41 範例步驟三

步驟三: 現在的溫度$T_3 = T_0 * (T_N/T_0)^{k/N} = 10*(0.2/10)^{3/5} = 0.96$ 大於終止溫度

T_N=0.2，隨機選擇二個資料項 1 與 2 進行廣播位置交換，從遞移封閉性矩陣得知，資料項 1 為資料項 2 的前行者，則 1 與 2 這二個資料項無法交換廣播位置。

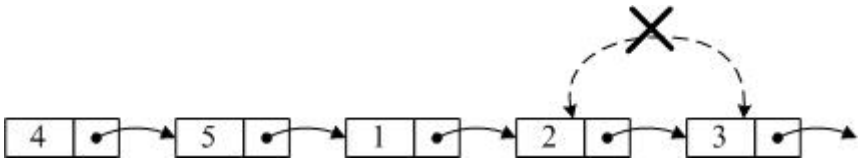
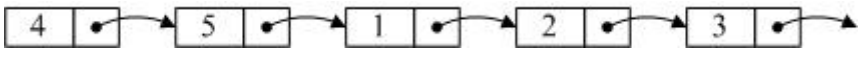


圖 45 應用 SA 演算法於圖 41 範例步驟四

步驟四: 現在的溫度$T_4 = T_0 * (T_N/T_0)^{k/N} = 10*(0.2/10)^{4/5} = 0.44$ 大於終止溫度 T_N=0.2，隨機選擇二個資料項 2 與 3 進行廣播位置交換，從遞移封閉性矩陣得知，資料項 2 為資料項 3 的前行者，則 2 與 3 這二個資料項無法交換廣播位置。

步驟五: 現在的溫度$T_5 = T_0 * (T_N/T_0)^{k/N} = 10*(0.2/10)^{5/5} = 0.2$ 等於終止溫度 T_N=0.2，停止演算流程。

經過 SA 演算法交換資料項在廣播序列內的位置後，得到一組新的廣播序列如下所示:



第五章 模擬實驗

為了評估使用拓樸排序演算法[23]、加入 Greedy 方式的拓樸排序演算法，以及 KL 演算法[20]和 SA 演算法[21]，在解決廣播排程問題的效能，我們執行模擬實驗來進行分析。5.1 節為模擬環境的介紹，5.2 節為實驗資料檔的產生與介紹，5.3 節為實驗結果分析。

5.1 模擬環境

在進行模擬實驗中，我們使用作業系統為 Microsoft Windows XP，利用 JAVA 程式語言來開發模擬程式。在模擬環境中，我們假設用戶端查詢的資料項大小固定、長度相等，且每一個用戶端查詢所要求的資料項為兩個，也就是「多重資料項的查詢」的廣播模式。負責廣播資料項的伺服端，其處理用戶端查詢的能力為非即時的，它會在一段時間內收集用戶端所要求之資料項之後，將資料項根據演算法進行排程，產生出一組較佳的廣播序列，透過廣播頻道將資料項傳送出去滿足用戶端需求，並降低用戶端的平均查詢讀取時間。

5.2 實驗資料檔的產生與介紹

經由造檔程式產生出進行模擬實驗所需的資料檔案，用以模擬用戶端對於伺服端提出資料項查詢的情形。在本文的實驗資料檔中，每一個

查詢的存取頻率使用不同的 Skew 參數在 Zipf 分配下執行實驗。

本文採用 Zipf 分配(Zip f distribution)[32]來假設用戶端讀取資料項的機率。θ為偏斜因素(skew factor)，$0 \le \theta \le 1$，藉由θ的增加偏斜程度會提高，當θ等於 0 的時候為均勻分配，每一個資料項被用戶端存取的頻率越相似。θ往上提高等於 1 時，就表示偏斜程度很高，各個資料項被用戶端存取的頻率越不平均。

表 1 檔名符號定義表

檔名符號	符號說明
zd	偏態分配資料檔
q	用戶端發出的查詢個數
s	選擇率(%)
k	查詢存取的偏斜程度

例如: 一個檔名為 *zd*50*q*600*s*004*k*01 的資料檔，表示廣播資料項個數為 50 筆，在一段時間內用戶端發送出 600 個查詢，選擇率為 4 % 表示每一個查詢所取的資料項為 2 筆(廣播資料項個數乘以選擇率 4 %)，查詢存取偏斜程度為 0.1。

將產生出的資料檔案利用原始的拓樸排序演算法、加入 Greedy 方式的拓樸排序演算法，以及在積體電路設計常被使用的 KL 演算法，與使用

SA 模擬退火演算法等方法，獲得的廣播排程結果相比較，試著找尋到較低的平均資料相依長度，以降低用戶端平均查詢讀取時間。

5.3 實驗結果分析

在這個部份裡，我們將透過實驗來評估在問題解決上所使用的各個演算法效能的差異。以用戶端收取資料項所花費的平均查詢讀取時間為標準，分別以不同的用戶端查詢數、不同的查詢偏斜程度以及廣播資料項總數來進行模擬實驗。計算用戶端平均查詢讀取時間為:

$\sum(D_{ij} \times W^{1}[i][j])$。

首先以廣播資料項個數 100 筆，偏斜程度為 0.4，選擇率為 2 % 也就是每一個查詢所取的資料項為 2 筆(廣播資料項個數乘以選擇率 2 %)，在不同查詢數量分別為 200、400、600、800 之下做演算法效能的比較。表 2 為實驗後數據。

表 2 在不同查詢數量下，演算法的平均查詢讀取時間

查詢數量 / 演算法	200	400	600	800
Topological	29.80031	36.35384	38.77608	39.89751
Greedy	23.79344	32.13412	34.42163	36.15738
KL	23.58036	31.68817	34.07248	35.8329
SA	25.56358	31.95355	34.67673	36.59469

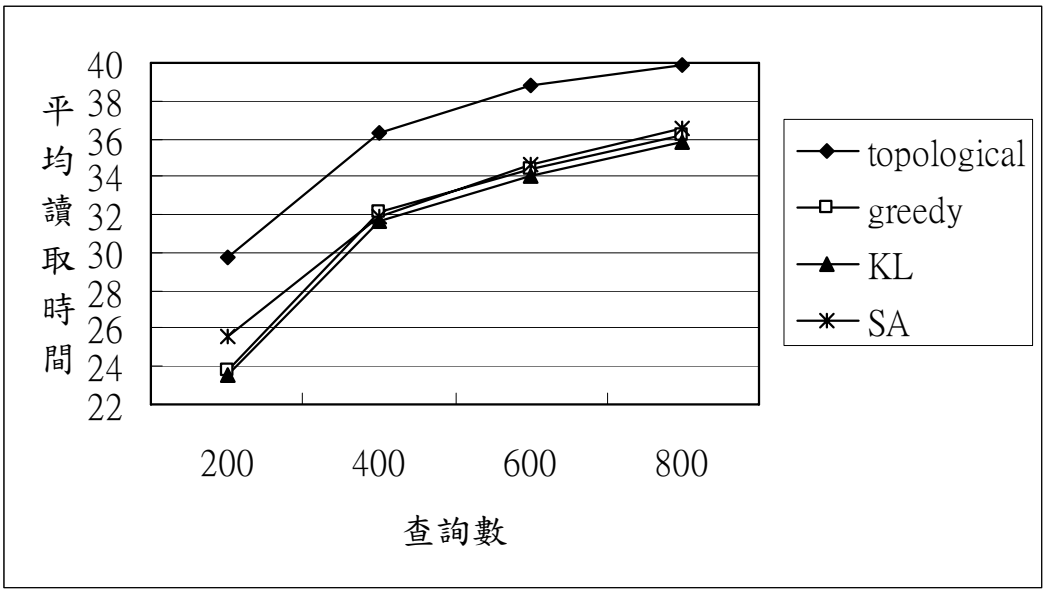


圖 46 依照不同 query 下用戶端的平均查詢讀取時間

圖 46 中顯示出不同查詢數量下用戶端花費的平均查詢讀取時間之變化，結果顯示出平均查詢讀取時間會隨著查詢數增加相對的提高，我們所提出的加入 Greedy 方式的 Topological sort 演算法比原始 Topological sort 演算法，得到的廣播序列其平均查詢讀取時間來得低，再經由應用 KL 最佳化演算法，將資料項之間的相依長度縮短，可以使得平均查詢讀取時間降的更低一些。使用 SA 演算法也是可以獲得不錯的效能。

在第二個實驗當中，我們以用戶端查詢數量 600 筆，和資料項總數為 50，選擇率為 4％，每一個查詢所取的資料項為 2 筆(廣播資料項個數乘以選擇率 4％)，在不同偏斜程度分別為 1、1.5、2、3 之下，呈現用戶端收取所需資料項花費的平均查詢讀取時間變化的情形。表 3 為實驗後

數據。

表 3 在不同偏斜程度下，演算法的平均查詢讀取時間

偏斜程度 / 演算法	1	1.5	2	3
Topological	17.81052	12.88289	7.619567	6.440066
Greedy	15.20578	10.57637	6.667007	3.78407
KL	14.994	10.04102	6.380601	3.743465
SA	15.1192	10.59392	6.638365	3.295766

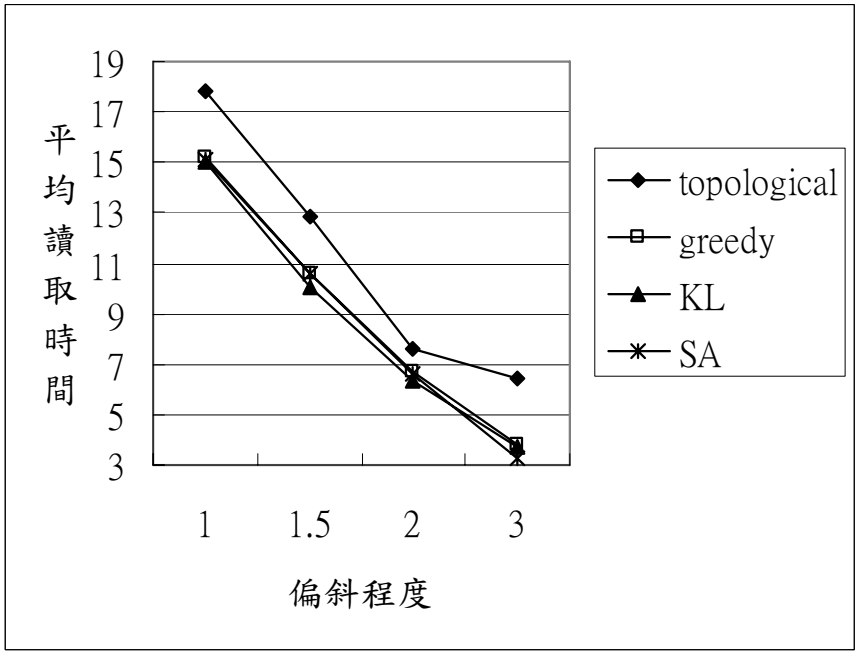


圖 47 依照不同偏斜程度下用戶端的平均查詢讀取時間

圖 47 中顯示出使用加入 Greedy 方式之 Topological sort 演算法，以及應用 KL 演算法、SA 演算法所得到的相關性資料項廣播順序，都比使用原始 Topological sort 演算法的廣播序列可獲得較短的用戶端平均查詢

讀取時間。

第三個實驗中，我們以用戶端查詢數量 600 筆，在偏斜程度為 2.0 時，不同資料項總數分別為 100、150、200、250 之下，觀察四個演算法的用戶端平均查詢讀取時間變化情形。表 4 為實驗後數據。

表 4 在不同資料項個數下，演算法的平均查詢讀取時間

資料個數 / 演算法	100	150	200	250
Topological	51.43925	82.43322	111.4631	124.5786
Greedy	46.65121	73.47091	87.39299	106.9133
KL	45.87139	72.9044	87.24771	106.5618
SA	43.13679	73.20895	81.58394	107.0785

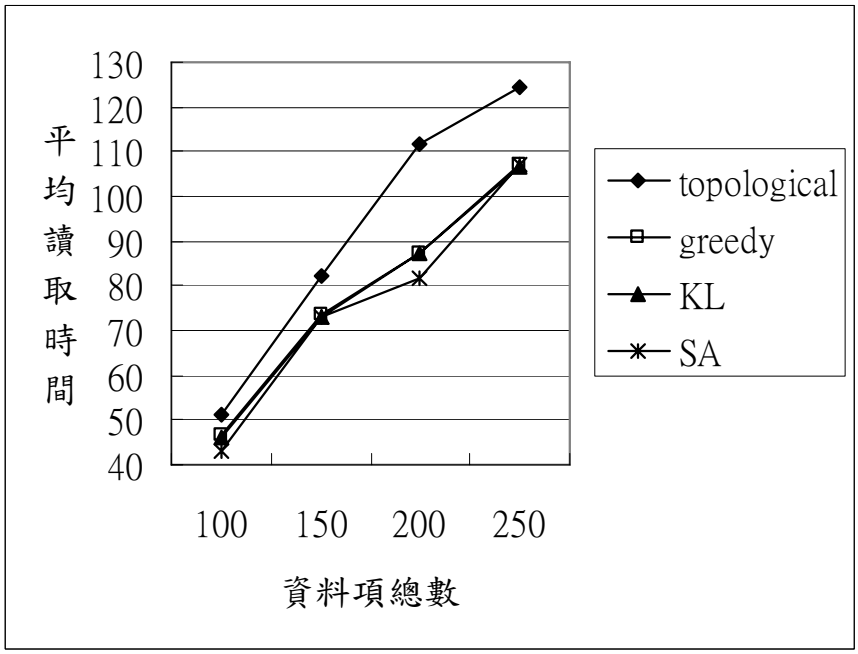


圖 48 依照不同資料項總數用戶端的平均查詢讀取時間

於圖 48 之中的實驗結果顯示出用戶端平均查詢讀取時間會隨著廣播資料項總數的增加而提高，使用加入 Greedy 方式的 Topological sort 演算法，以及應用 KL 演算法、SA 演算法所得到的廣播序列，皆可獲得較使用原始 Topological sort 演算法得到的廣播順序較短的用戶端平均查詢讀取時間。經由 SA 演算法排序後得到的廣播序列，其用戶端的平均查詢讀取時間，有時是多個演算法中最低，因為 SA 演算法在選擇移動資料項時，可接受任何的可能性，反能將結果帶入另一個區域的最佳解或是整體的最佳解。

第六章 結論

本研究所探討的無線廣播排程問題中，以廣播資料項之間存在的相依性為主要考量，使用一個有向無循環圖形(DAG)表示其順序限制，且每一資料項對應於圖形中每一頂點，並應用Greedy 拓樸排序演算法、積體電路設計中常被使用的KL演算法，以及SA模擬退火演算法等方法於DAG的排序問題，試圖藉由降低頂點之間平均路徑長度，用以減少用戶端平均的查詢讀取時間。

經過拓樸排序演算法排序之後，有向無循環圖形轉換成有線性次序關係之拓樸序列，應用加入Greedy 機制之拓樸排序演算法，決定出一組較佳之廣播序列，以及使用KL最佳化演算法與SA演算法，透過移動資料項的廣播序列位置，試圖降低資料項之間的相依長度。KL最佳化演算法執行時間較短，而使用SA演算法得到的序列，會受執行次數與執行時間所影響，執行次數較多、執行時間較長時，應用SA演算法可獲得較佳的廣播序列。

經由實驗結果顯示，使用我們所提出Greedy方式所改進的拓樸排序演算法，以及KL最佳化演算法、SA演算法，確實可以縮短資料項之間的相依性長度，使得用戶端在收取所需資料項時，花費的平均查詢讀取時間降低。

參考文獻

[1] E. H. L. Aarts, F. M. J. De Bont, E. H. A. Haberts and P. J. M. Van Laarhoven,“Statistical Cooling: A General Approach to Combinatorial Optimizations”, Philips Journal of Research, Vol. 4,pp.193-226, 1985.

[2] N. Abbound, M. Sakawa and M. Inuiguchi, “Statistical Cooling: A General Approach to Combinatorial Optimizations”, Philips Journal of Research, Vol. 29, pp.593-611, 1998.

[3] S. Acharya, R. Alonso, M. Franklin and S. Zdonik, “Broadcast disks: Data mangement for asymmetric communication environments,” Proceedings of ACM SIGMOD Conference, pp. 199-210, 1995.

[4] S. Acharya, M. Franklin and S. Zdonik, “Disseminating updates on broadcast disks,” in proceedings of Very Large Data Bases Conference, pp. 354-365, 1996.

[5] D. Aksoy and M. S. F. Leung, “Pull vs Push: A Quantitative Comparison for Data Broadcast,” Global Telecommunications Conference, Vol. 3 ,pp.1464-1468, 2004.

[6] A. A. Bertossi, M. C. Pinotti and S. Ramaprasad, “Optimal multi-channel data allocation with flat broadcast per channel,” Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004.

[7] O. E. Demir and D. Aksoy, “Energy-Efficient Broadcast-based Event Update Dissemination,” Pertormance, Computing, and Communications, 2004 IEEE International Conference ,pp. 477-482, 2004.

[8] Q. Fang, S. V. Vrbsky, Y. Dang and W. Ni, “A Pull-Based Broadcast Algorithm that Considers Timing Constraints,” Proceedings of the International Conference on Parallel Processing Workshops, 2004.

[9] C. M. Fiduccia and R. M. Mattheyses, ”A Linear-time Heuristic for Improving Network Partitions”, Proc. Design Automation Conf., pp. 175-181, 1982.

[10] M. Garey and S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, 1979.

[11] L. Hagen and A. B. Kahng, “Fast Spectral Methods for Ratio Cut Partitioning and Clustering”, Proc. International Conf. on Computer-Aided Design, pp. 10-13, 1991.

[12] J. L. Huang and M. S. Chen, “Broadcast Program Generation for Unordered Queries with Data Replication,” Proceedings of the ACM Symposium on Applied Computing, pp. 866-870, 2003.

[13] J. L. Huang and M. S. Chen, “Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Environment,” IEEE Transaction on Knowledge and Data Engineering ,Vol.16, No.9, pp. 1143-1156, 2004.

[14] J. L. Huang, M. S. Chen and W. C. Peng , “ Broadcasting Dependent Data for Order Queries Without Replication in a Multi-Channel Mobile Environ ment,” Proceedings of the 19th International Conference on Data Engineering, 2003.

[15] Y. Huang and Y. H. Lee, “An Efficient Indexing Method For Wireless Data Broadcast With Dynamic Updates,” Communications, Circuits and Systems and West sino Expositions, IEEE 2002 International Conference, Vol.29, pp. 358-362, 2002.

[16] J. J. Huang and Y. Lee, “Efficient Index Caching Schemes for Data Broadcasting in Mobile Computing Environments,” Proceedings of the 14th International Workshop on Database and Expert Systems Applications, 2003.

[17] K. F. Jea and M. H. Chen, “A Data Broadcast Scheme Based on Prediction for The Wireless Environment,” Proceedings of the Ninth international Conference on Parallel and distributed Systems, 2002.

[18] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, ”Multilevel Hypergraph Partitioning:Applications in VLSI Domain,” Department of Computer Science, University of Minnesota, Minneapolis, MN, USA, 1997.

[19] G. Karypis and V. Kumar, ”METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering-Version 2.0,” Department of Computer Science,University of Minnesota, Minneapolis, MN, USA, 1995.

[20] B. W. Kernighan and S. Lin, ”An Efficient Heuristic for Partitioning Graphs,” The Bell System Technical Journal, 49, pp. 291-308, 1970.

[21] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, “Optimization by Simulated Annealing”, Science, pp. 671-680, 1983.

[22] G. Lee, M. S. Yeh, S. C. Lo and A. L. Chen, “A Strategy for Efficient Access of Multiple Data Items in Mobile Environments,” Proceedings of the Third International Conference on Mobile Data Management , 2002.

[23] J. Ma, K. Iwama, T. Takaoka and Q. P. Gu, “Efficient Parallel and Distributed Topological Sort Algorithms,” Parallel Algorithms/Architecture Synthesis, Proceedings, Second Aizu International Symposium, pp. 378-383, 1997.

[24] W. Ni , Q. Fang and S. V. Vrbsky , “A Lazy Data Request Approach for On-demand Data Broadcasting,” Proceedings of the 23rd International Conference on Distributed Systems Workshops , 2003.

[25] W. C. Peng, J. L. Huang and M. S. Chen, “Dynamic Leveling: Adaptive Data Broadcasting in a Mobile Computing Environment,” Mobile Networks and Applications 8, pp. 355-364, 2003.

[26] OBV Ramanaiah and H. Mohanty, “NICD: A Novel Indexless Wireless On-Demand Data Broadcast Algorithm,” Proceedings of the Internation Conference on Information Technology: Coding and Computing , 2004 .

[27] N. Saxena, K. Basu and S. K. Das, “Design and Performance Analysis of a Dynamic Hybrid Scheduling Algorithm for Heterogeneous Asymmetric Environments,” Proceedings of the 18th International Parallel and distributed Processing Symposium , 2004.

[28] S. Wang and H. L. Chen, “Near-Optimal Data Allocation Over Multiple Broadcast Channels,” Networks, Proceedings. 12th IEEE International Conference on Vol 1, pp. 207-211, 2004.

[29] X. Wu and V. C. S. Lee, “Preemptive Maximum Stretch Optimization Scheduling for Wireless On-Demand Data Broadcast,” Proceeding of the International Database Engineering and Applications Symposium , 2004.

[30] J. Xu, B. Zheng, W. C. Lee and D. L. Lee, “Energy Efficient Index for Querying Location-Dependent Data in Mobile Broadcast Environments,” Proceedings of the International Conference on Data Engineering , 2003.

[31] J. Zhang and L. Gruenwald , “Optimizing Data Placement Over Wireless Broadcast Channel For Multi-Dimensional Range Query Processing,” Proceedings of the International Conference on Mobile Data Management, 2004.

[32] G. K. Zipf. Human Behavior and the Principle of Least Effort. Addison-Wesley, Massachusetts, 1949.