南　　華　　大　　學

資訊管理學系

碩士論文

植基於頂點對排序的關聯性資料排列法的探討

Solving the Directed Linear Arrangement Problem with successive vertex pair ordering technique

研 究 生：藍弘文

指導教授：蔡德謙

中華民國　2010　年　1　月

南　華　大　學

資訊管理學系

碩 士 學 位 論 文

植基於頂點對排序的關聯性資料排列法的探討

研究生：藍弘文

經考試合格特此證明

口試委員：洪紹鑫

吳光鳴

指導教授：蔡佩詩

系主任(所長)：鍾國貴

口試日期：中華民國 98 年 12 月 31 日

南華大學資訊管理學系碩士論文著作財產權同意書

立書人：藍弘文 之碩士畢業論文

中文題目：植基於頂點對排序的關聯性資料排列法的探討

英文題目：Solving the Directed Linear Arrangement Problem with successive vertex pair ordering technique

指導教授：蔡德謙 博士

學生與指導老師就本篇論文內容及資料其著作財產權歸屬如下：

☑ 共同享有著作權

☐ 共同享有著作權，學生願「拋棄」著作財產權

☐ 學生獨自享有著作財產權

學　　生：藍弘文 （請親自簽名）

指導老師：蔡德謙 （請親自簽名）

中　華　民　國　98　年　12　月　31　月

南華大學碩士班研究生

論文指導教授推薦函

資訊管理 系碩士班 藍弘文 君所提之論文

植基於頂點對排序的關聯性資料排列法的探討

係由本人指導撰述，同意提付審查。

指導教授 蔡德諏

98 年 12 月 31 日

誌　　謝

研究所的兩年半時間，感謝父母親讓學生無壓力的完成學業。感謝南華大學的各位老師，令學生學習許多的知識與人生的道理，以及感謝蔡德謙老師與吳光閔老師在 meeting 時，指導學生在台上的演講。感謝系助理（伊汝姊）的幫忙，使學生在口考前置得以順利，還有感謝各位學長姊（詠生、淳淳、慶豐）與同學（淑樺、佳濬、雲賢、志瑋）以及學弟妹（東迪、成宏、美秀、嘉俊、瀞文）的指導與幫助。

非常感謝蔡德謙老師、吳光閔老師、洪紹鑫老師在口試的指導，給予學生許多寶貴的意見，最後非常感謝蔡德謙老師兩年半的指導，讓學生得以順利完成論文。

植基於頂點對排序的關聯性資料排列法的探討

學生：藍弘文　　指導教授：蔡德謙

南　華　大　學　資訊管理學系碩士班

摘　　要

有向線性排程在於工作排程及無線資料廣播中應用很廣泛且具研究的問題。這著作提出的演算法有不同等級的複雜度來解決此問題，像混合比率切割貪婪排序演算法，其中有數個演算法是根據連續頂點對排序技術及等同邊叢集演算法。混合比率切割貪婪排序演算法是應用雙向貪婪排序以及遞迴二分法，故混合比率切割貪婪排序演算法在全域最佳化中有高效能的區域搜尋。連續頂點對排序技術應用在兩種環境，一種是在於權重邊下的最大延遲縮減技術，另一種是在於優先邊下的搶占邊技術。最大平均延遲縮減比起傳統以切割為基礎的演算法，更可以達成較佳解，如果在權重邊只有優先順序的情形下，搶占邊演算法可以採用且有效。等同邊叢集演算法應用等同邊來取代邊，且使用有向 HAC 結合頂點成為片斷(segment)。透過實驗來確定不同啟發演算法的效率。

關鍵字： 有向線性排程，貪婪搜尋

Solving the Directed Linear Arrangement Problem with successive vertex pair ordering technique

Student：Hong wen Lan Advisors：Dr. Derchian Tsaih .

Department of Information Management
The Graduated Program
Nan-Hua University

ABSTRACT

Directed linear arrangement is a widely adopted and studied problem in task-scheduling and wireless data broadcasting. This work presents algorithms with various levels of complexity to solve this problem, including a hybrid ratio cut greedy sort algorithm, several algorithms based on the successive vertex pair ordering technique, and a equivalent edge clustering algorithm. The hybrid ratio cut greedy sort algorithm, which adopts the two-way greedy sort and recursive bipartition, is a highly efficient local search algorithm with global optimization. The successive vertex pair ordering technique comprises the maximum latency reduction technique for weighted edges, and the preemptive edge technique for priority edge. The algorithm based on maximum average latency reduction could delivers a better solution outcome than a classic cut-based algorithm, and the algorithm based on the priority edge and be adopted when only the edge priority (or preference) is available. The equivalent edge clustering algorithm adopts the equivalent edge technique to replace edges, and adopts the directed HAC process to merge vertices into a segment. Experiments are conducted to check the effectiveness of different heuristic algorithms.

Keywords：Directed linear arrangement, greedy search.

目錄 x

表目錄

圖目錄

符號定義

表 1　符號定義

符號	含意
$F_{in}(u)$	頂點u入射邊的總權重
$F_{out}(u)$	頂點u射出邊的總權重
$F^{+}(u)$	頂點u的 $F_{out}(u)-F_{in}(u)$
p	拓樸排序
p_i	指拓樸排序中第i個頂點
$W(p)$	拓樸排序的總權重值
$W_i, W_i(p)$	拓樸排序中第i個頂點的權重值
$w_{u,v}$	指從頂點u到頂點v的權重
$u \prec v$	頂點u安排在頂點v之前
$u \preceq v$	頂點u安排在頂點v之前或$u=v$
$u \prec\succ v$	頂點u與頂點v都未知先後順序
$u \Rightarrow v$	頂點u到頂點u至少存在一條路徑
$p^{(f)}$	透過往前排序時，將頂點附加到此集合。
$p^{(b)}$	透過往後排序時，將頂點前置加到此集合。
$\Psi^{(f)}$	在於$p^{(f)}$的頂點集合

符號	含意
$\Psi^{(b)}$	在於 $p^{(b)}$ 的頂點集合
$\Psi^{(z)}$	未排序到 $p^{(f)}$ 或 $p^{(b)}$ 的頂點集合，$V=\Psi^{(b)}\bigcup\Psi^{(b)}\bigcup\Psi^{(z)}$
$T[u][v]$	識別 $u\prec v$ 的二元矩陣，若 $u\prec v$ 則 $T[u][v]=1$ 否則 $T[u][v]=0$
$A[u][v]$	令 $u\prec v$，所有平均延遲縮減值
$R(u,v)$	安排 $u\prec v$，所有有影響的頂點對

名詞定義

表 2　名詞定義

英文縮寫	中文定義
DAG	有向無循環圖 (Direct Acyclic Graph)
GS	貪婪拓樸排序 (Greedy Topological Sort)
TGS	雙向貪婪拓樸排序 (Two-way Greedy Sort)
RTGS	遞迴雙向貪婪拓樸排序 (Recursive Two-way Greedy Sort)
RRCP	遞迴比率切割區塊演算法 (Recursive Ratio Cut Partition Algorithm)
HRCGS	混合比率切割貪婪排序演算法 (Hybrid Ratio Cut Greedy Sort Algorithm)
MLRP	最大延遲縮減對演算法 (Maximum Latency Reduction Pair Algorithm)
MALRP	最大平均延遲縮減對演算法 (Maximum Average Latency Reduction Pair Algorithm)
TGS – MALRP	雙向貪婪排序與 MALRP 演算法 (Two-Way Greedy Sort with MALRP Algorithm)

英文縮寫	中文定義
RBFM – MALRP	遞迴型 FM 二分法與 MALRP 演算法 (Recursive FM Bipartitioning with MALRP Algorithm)
PE	搶占邊演算法 (Preemptive Edge Algorithm)
TGS – PE	雙向貪婪排序與搶占邊演算法 (Two-Way Greedy Sort with PE Algorithm)
RBFM – PE	遞迴型 FM 二分法與搶占邊演算法 (Recursive FM Bipartitioning with PE Algorithm)
EEC	等同邊叢集演算法 (Equivalent Edges Clustering Algorithm)

第一章 介紹

第一節　研究背景與動機

近年來，資訊科技成長快速，從過去將資料轉換成數位化的 E 化，到現在的透過各種移動設備隨時隨地處理資料的 M 化，使得無線網路的應用更加廣泛，伺服器可以透過無線網路的廣播將公開的資訊散播給其他的移動用戶，例如交通資訊、股市訊息及氣象等，當移動用戶要存取這類資料可以調整進入到廣播的頻道並傾聽廣播中所請求的資料。

當存取此資料時，在廣播週期的順序中，用戶所請求的資料要如何有組織、有效率的接收，考慮到移動型設備(例如：PDA、手機…等)有著能源的限制，且在接收廣播的資料是較消耗能源，若延遲時間太長，則設備需耗用更多的能源來接受請求的資料；若廣播的頻道數較多或是頻寬較大，則可以提供多個來源或較高速的網路給設備來接收請求的資料，但目前無線網路的寬頻還有成長空間，因此近年來許多的學者嘗試排程廣播資料項來節省頻寬與移動設備的能源消耗。

在無線網路的環境，可以透過兩者來看出用戶請求資料到接收完資料所需耗費的時間：延遲時間(access time)以及調整時間(tuning time)[1]。延遲時間指用戶端送出請求至伺服器後，到接收完請求的資料項所花費的時間，此段時間亦可稱為存取時間(access time)；延

遲時間越短，則用戶端能在越短的時間內接收完所請求的資料。而調整時間是指用戶端調整到廣播頻道中，到開始接收請求資料所花費的時間，若用戶有著較短的延遲時間與調整時間則能確保伺服器的廣播排程有著較好的服務品質(Quality Of Service)。

考慮到如何有組織、有效率的廣播排程，將相依的資料項以較鄰近的方式排程可以使用戶端的延遲時間較短，反之，若將資料項排序的較為分散，則用戶所需要的延遲時間就較長，因此需要耗費更多的能源在接收、傾聽廣播中所請求的資料，因此有著好的廣播排程則可以節省設備的能源，亦可以更有效的使用網路頻寬。

第二節　研究主題

在過去的一些研究，用戶端請求的資料被視為彼此互相獨立[2]，但在現實的應用上，用戶所請求的資料是具有相依性[3 , 4]，例如用戶端請求網頁資料時，該網頁有著圖片、影音檔…等，這些圖片、影音檔將在瀏覽此網頁同時被依序提出請求，因此這些圖片、影音檔是相依於此網頁。在上述例子中，我們可以將請求的資料視為頂點(vertex)，而存取的順序視為有箭頭方向的邊(edge)，而存取資料所需的延遲時間視為邊的權重，因此整個資料廣播可以表示(如圖 1)具有權重的有向無循環圖(directed acyclic graph，DAG)。

將前例套用到圖 1，頂點 A、B 視為網頁，而頂點 C、D、E、F

則視為圖片、影音，或是子網頁，因此本研究透過排程有向無循環圖，期望產生一個有效的有向線性排程(如圖 2)。

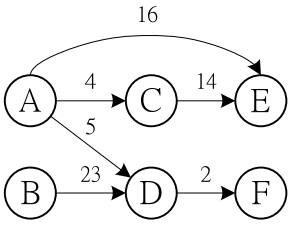


圖 1 具有權重的有向無循環圖

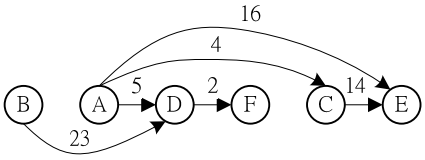


圖 2 有向線性排程

第三節 論文架構

第一章簡要的介紹關於研究動機、背景與主題。第二章是定義問題與符號以及相關的文獻的介紹，第三章敘述關於混合比率切割貪婪排序演算法，第四章討探關於連續頂點對排序技術，第五章描述有向叢集技術以及有向叢集技術與先前演算法結合使用，第六章在不同參數下進行模擬實驗的模擬結果，以及評估結果，最後第七章是本文的結論。

第二章 有向線性排程問題

第一節　文獻探討

壹、圖形結構

圖形理論的起源從 1736 年萊昂哈德·歐拉(Leonhard Euler, 1707-1783)解決柯尼斯堡七橋(Seven Bridges of Königsberg)的問題[5]，發現圖形(Graph)可以透過頂點(vertex)及邊(edge)來建構，因此可以以$G=\{V,E\}$來表示，G表示圖形，V表示頂點(vertex)集合，E表示邊(edge)集合。而圖形以邊的方向來決定差異，若邊並無方向性，則稱此圖形為無向圖(Undirected Graphs)(如圖 3)；若邊有著方向性，則稱為有向圖(directed Graphs)(如圖 4、圖 5)，但若邊有著方向性，而且頂點之間形成循環，則稱為有向循環圖(如圖 4)，反之稱為有向無循環圖(如圖 5)。

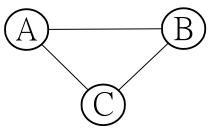


圖 3、無向圖

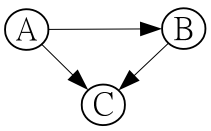


圖 4、有向無循環圖

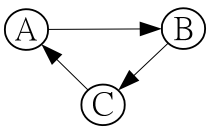


圖 5、有向循環圖

本研究加入w，在有向無循環圖$G=\{V,E,w\}$中，V、E如前文所，而w指的是權重，從父頂點到子頂點的延遲時間(latency)，由二元矩陣組成且此權重不可為負數，並且在有向無

循環圖中是無反向邊的，以圖 1 為例，頂點 A 可以到達頂點 C，但頂點 C 不可到頂點 A，也就是說有一條路徑(u,v)表示是從 u 連到 v，u,v∈V，u≠v，且沒有(v,u)。

貳、相關研究

無向線性排程已被廣泛的使用，而且常被稱為 *edgesum* 問題、線性排列問題[6]，或是線性佈置問題[7]。而且許多的方法已經提出來解決原始的 MinLA(Minimum Linear Arrangement)問題。Saab[7]採用重複節點叢集(或壓縮)來計算線性放置。

在過去有研究在探討在異質系統(heterogeneous system)下排序相依工作[8]，Sakellariou 提出分割演算法，此演算法透過局部群組最佳化以及排序頂點階級值來決定最佳頂點順序。然而此技術在於每個區域群組以及平衡群組的大小中最佳化子網路節點的順序。

聚合(或階層式聚合)叢集已廣泛且成功地應用於不同領域的叢集資料中的[9]。Saab[7]以及 Safro[10]的節點壓縮(node compaction)那是使用叢集技術來決定線性頂點安排的應用。

現有各種的啟發式演算法在線性排程問題的實驗有兩階段流程，首先挑選一個開始的頂點在輸出名單的前端中，且隨著頂

點選擇流程輸出一個局部最小化成本的頂點來放置頂點。

McAllister[11]提出一個正向增加最小(frontal increase minimization)的啟發，根據兩階段流程，最小化在放置與未放置的頂點之間切割值(cut)。

第二節　問題定義

本文主要為有向線性排程研究，有向線性排程為最小化線性排程的有向版本，具有無向線性排程裡原有的限制，有著頂點之間的相依性、頂點之間權重邊的影響。

假設存在一個有向無循環圖 $G=\{V,E,w\}$，具有 n 個頂點以及 m 個無負數的權重邊，我們試圖在有向線性排程中找出一個有著最小化總權重延遲邊的拓樸排序，且一個有效的頂點拓樸排序中頂點是沒有反向邊的。DLA(Directed Linear Arrangement)問題期望找出一個邊的總權重延遲是最小的排列 p，總權重延遲可以透過下列公式得出：

$$W(p)=W(p_1,p_2,\ldots,p_n)=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(j-i)w_{p_i,p_j} \qquad \text{方程式 (1)}$$

第三章 混合比率切割貪婪排序演算法

第一節　貪婪拓樸排序

貪婪拓樸排序是一個頂點排序演算法，在*DAG*中，令來源頂點(source)是沒有射入邊的頂點，而目的地頂點(sink)是沒有射出邊的頂點。貪婪拓樸排序演算法反覆執行以下的程序：選擇來源頂點u，接著附加進p，並從*DAG*中刪除u及u的射出邊。

設$F_{in}(u)=\sum_{(v,u)\in E} w_{v,u}$表示頂點$u$的總入射權重，$F_{out}(u)=\sum_{(u,v)\in E} w_{u,v}$表示頂點$u$的總射出權重，令$F^{+}(u)=F_{out}(u)-F_{in}(u)$為頂點$u$的淨流出量。演算法一開始$W_0=0$，則每個$W_i$可以依下列公式推論出：

$$W_i=\sum_{j\le 1} F^{+}(p_j)=W_{i-1}+F^{+}(p_j) \qquad \text{方程式 (2)}$$

依照貪婪式的推論邏輯，在每個階段選擇的區域最佳解為選擇來源頂點中最小淨流出量的最佳頂點，亦即為有最小的F^{+}的頂點。

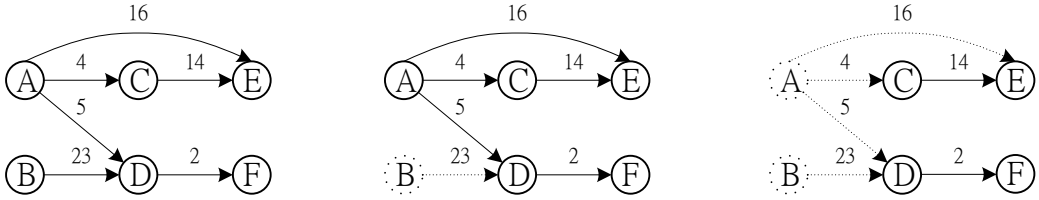


圖 6　貪婪拓樸排序的例子

以圖 6 為例，一開始$W_0=0$，$p=\{\}$，候選的來源頂點有：A,B，

$F^+(A)=(16+4+5)-0=25$，$F^+(B)=23-0=23$，因此首先選擇的頂點為B，$p=\{B,\}$，$W_1=0+23=23$，接著移除頂點B與頂點B射出邊，如圖 6 的第一步。

p加入頂點B之後，接著候選來源頂點有：A,D，但因頂點D的射入邊是有從頂點A射出的，因此排除頂點D，第二個選擇來源頂點便只考慮頂點A，$p=\{B,A\}$，$W_2=23+(16+4+5)=48$，接著移除頂點A與頂點A射出邊，如圖 6 第二步。

可以發現兩個候選來源頂點：C,D，$F^+(C)=14-4=10$，$F^+(D)=2-(23+5)=-26$，因此選擇的來源頂點為 D (-26<10)，$p=\{B,A,D\}$，$W_3=48+(-26)=22$，接著移除D與D的射出邊。

反覆的上述動作，可以得到：$W_4=22+(-2)=20$，$W_5=20+10=30$，$W_6=30+(-30)=0$，$p=\{B,A,D,F,C,E\}$，總權重為 23+48+22+20+30+0=143。

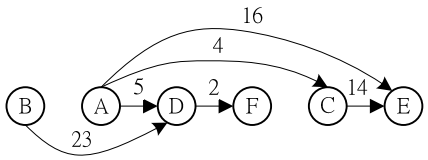


圖 7　貪婪拓樸排序例圖之結果

第二節　雙向貪婪拓樸排序

貪婪拓樸排序從現有的來源頂點中挑出具有最小$F^+(u)$的頂點，並放入p中，然而，當存在兩個F^+相若的頂點但F_{out}差異大的頂點時，在放置有較小F^+的頂點到p時，會使得有較大$F_{out}(u)$的頂點延遲放置，其頂點安排並非最佳。

因此，如果將*GS*視為往前排序(forward sort)，則亦存在一個往後排序(backward sort)的拓樸排序。此往後的拓樸排序是由目地的(sink)頂點開始選擇頂點並且從p的尾端開始放置。開始時，令$W_n = 0$，每個W_i的計算公式依下列公式推論出：

$$W_i = -\sum_{j>i} F^+(p_j) = W_{i+1} - F^+(p_i) \qquad \text{方程式 (3)}$$

如同貪婪拓樸排序的策略，往後排序是從現有的目的地頂點中選擇有著最大$F^+(u)$的頂點，並從p的尾端開始而往前放置；往後排序也存在著類似往前排序的缺點：由一邊來尋找頂點與放置到p，當存在兩個F^+相若的頂點但F_{in}差異大的頂點時，在放置有最大F^+的頂點到p時，會使得有較大$F_{in}(u)$的頂點延遲放置，其頂點安排並非最佳。然而，可以結合往後排序與往前排序，從任一邊來放置頂點到p。

令$p^{(f)}$為透過往前排序所挑選出來頂點所放置的序列(segment)而$p^{(b)}$為透過往後排序所挑選出來頂點所放置的序列(segment)；令u是在所有候選來源頂點中有著最小$F^+(u)$的頂點，而v是在所有候選

目起的頂點中有著最大$F^+(v)$的頂點。假如$F^+(u) < -F^+(v)$，則使用往前排序將所挑選出來的u加入$p^{(f)}$，且從DAG中刪除u以及所有從u射出的邊，反之，則使用往後排序將所挑選出來的v前置加到$p^{(b)}$，且從圖中刪除v以及所有射入v的邊。

除非只有單一來源頂點或是單一目的地頂點存在，否則雙向貪婪拓樸排序會在往前排序與往後排序中選擇最小切割值(cut)增加幅度的排序頂點。假設有多個來源頂點與目的地頂點，雙向貪婪拓樸排序會在現有來源頂點與現有目的地頂點中挑選出有最小切割值增加率的頂點。假如只有一個來源頂點或是目的地頂點存在，則會挑選該頂點以增加下個排序中可選擇的頂點數。

藉由往前排序與往後排序，有些頂點會變成是來源頂點且同時是目的地頂點(如圖 8)，在這情形下，邊的延遲可以透過限制來源或目的地頂點的選擇來進一步縮減切割值。

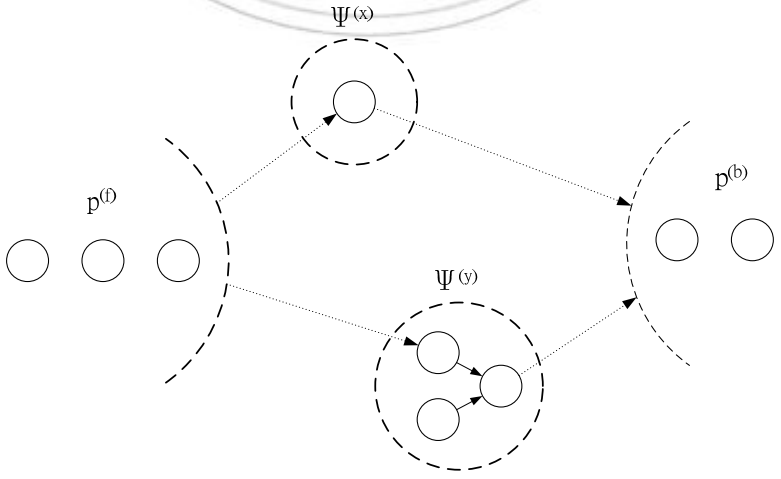


圖 8　雙向貪婪拓樸排序時兩個未連接的頂點集合

在本節，定義兩個定理：

定理 1.雙向貪婪拓樸排序如圖 8 所示，設$\Psi^{(z)}$表示尚未安排到$p^{(b)}$或$p^{(f)}$的頂點集合，若有兩個獨立頂點集合$\Psi^{(x)}$與$\Psi^{(y)}$，且存在著$\Psi^{(x)} \bigcup \Psi^{(y)} = \Psi^{(z)}$，令$W'$表示$\Psi^{(x)}$放置在$\Psi^{(y)}$之前的最小總權重延遲，$W''$表示$\Psi^{(y)}$放置在$\Psi^{(x)}$之前的最小總權重延遲，那$W'$與$W''$兩者總切割值的差異可以依下列公式推論出：

$$W' - W'' = \left|\Psi^{(x)}\right| \sum_{v \in \Psi^{(y)}} F^{+}(v) - \left|\Psi^{(y)}\right| \sum_{v \in \Psi^{(x)}} F^{+}(v) \qquad \text{方程式 (4)}$$

如果$\Psi^{(x)} = \{u\}$，則u附加到$p^{(f)}$與前置加到$p^{(b)}$之間的切割值差異為$\left|\Psi^{(x)}\right| F^{+}(u) - \sum_{v \in \Psi^{(y)}} F^{+}(v)$，因此，如果$u$是來源頂點同時也是目的地頂點，且$F^{+}(u) < \frac{1}{\left|\Psi^{(z)}\right|} \sum_{v \in \Psi^{(z)}} F^{+}(v)$，則$u$就只該當成來源頂點，否則只該當成目的地頂點。

當雙向貪婪排程開始時，$p^{(f)}$及$p^{(b)}$初始化成兩個空序列，而後反覆執行(1)往前排序將來源頂點附加到$p^{(f)}$，或(2)往後排序將目的地頂點前置加到$p^{(b)}$。最後則透過串接$p^{(f)}$與$p^{(b)}$來排列出完整的排序p。

透過調和這兩個方法，可以減輕在往前排序中$F_{out}(u)$較大的頂點以及在往後排序中$F_{in}(u)$較大的頂點，會延遲排序的弱點。藉由搜尋來源頂點與目的地頂點，雙向貪婪排序比起單向貪婪排序更不容易受到輸入的圖形變化的影響，因此可以更容易地選擇最佳頂點。

令$W^+(p_i, p_{i+1}, \ldots, p_j)$表示序列$p_i, p_{i+1}, \ldots, p_j$的成本函數，可以依下列公式推論出：

$$W^+(p_i, p_{i+1}, \ldots, p_j) = \sum_{k=i}^{j} (j-k+1)F^+(p_k) \qquad \text{方程式 (5)}$$

定理 2.設$S_x(p_i, p_{i+1}, \ldots, p_j)$表示任意拓樸排序的序列$p$的一個片斷，且$S_y$表示再排序$S_x$後的序列。且令$p'$表示$S_y$取代$S_x$後的$p$，在此情形下，假如$W^+(S_y) < W^+(S_x)$，則$W^+(p') < W^+(p)$。

因此，如方程式 (5)所示，在有向無循環圖中應用雙向貪婪排序之後，亦可以藉由應用雙向貪婪排序在其中一段序列，以進一步精煉p。

第三節　遞迴雙向貪婪拓樸排序

在往前排序中選擇具有最小$F^+(u)$的來源頂點。往後排序中選擇具有最大$F^+(v)$的目的地頂點時會形成有著較大的權重邊的開始與末端頂點容易被同一排序方法所選擇(往前或往後排序)，所以會自然的形成分割。在不移動任何頂點到另一個分割區的情形下，$p^{(f)}$與$p^{(b)}$可以再進一步各自以遞迴 TGS 的方式精煉。

遞迴雙向貪婪拓樸排序是以遞迴的方式運行雙向貪婪拓樸排序，開始時$i=1$，$j=n$以及$p_k = k, \forall k$，每次遞迴雙向貪婪拓樸排序過程是建立從頂點p_i到p_j的有向圖，且使用雙向貪婪拓樸排序找出

$p^{(f)}$與$p^{(b)}$兩個序列，接著使用兩個獨立的雙向貪婪拓樸排序來再次排序$p^{(f)}$與$p^{(b)}$。

第四節　遞迴比率切割區塊演算法

divide-and-conquer 透過解決數個小問題來解決一個大型的問題，同理，透過移除頂點之間所連結的邊，可以將一個大型的圖形切割為兩個分離的子圖形，因此可以在一個大型的圖形以遞迴的方式使用二分法(bipartition)來解決圖形過大而不易處理問題。

Fiduciary and Mattheyses 所提出循環分割是在分割的區塊之間透過一次移動一個頂點，以減少穿過分割的總切割。他們的方法是個典型利用 divide-and-conquer 的方法，且已被証實有很好的應用成效。然而，假如分割的群組不適當，則分割結果可能會惡化為 k-way 分割或是循環二分法問題(recursive bipartition problem)，而產生較不佳的解。

為了解決群組的問題，Cheng and Wei[12] 提出對於 FM 演算法的切割值比率測量(ratio cut measure)，已廣泛的應用到許多分割演算法[13]。

切割值比率的構想允許在分割時，在切割值與頂點之間平衡取捨。我們在此研究中使用下列最小切割值比率作為二分法的成本函數。

$$Rcut(\Psi^{(f)},\Psi^{(b)}) = (\frac{1}{\left|\Psi^{(f)}\right|} + \frac{1}{\left|\Psi^{(b)}\right|}) \sum_{u\in\Psi^{(f)},v\in\Psi^{(b)}} w_{u,v} \qquad \text{方程式 (6)}$$

RRCP 演算法是一個標準的遞迴式二分演算法，在每個二分流程包含兩個各自的兩個子流程。在往前排序流程，假如已有 i 個頂點在 $p^{(f)}$ 中，為了挑選具有最小切割值比率成本的頂點，可以透過方程式(7)來挑取頂點 u 。

$$Rcut^{(f)} = (W_i + (F^+(u))(\frac{1}{i+1} + \frac{1}{n-i-1}), \forall 0 \le i < n-1 \qquad \text{方程式 (7)}$$

在往後排序流程，假如已有 j 個頂點在 $p^{(b)}$ ，為了挑選具最小切割值比率成本的頂點，可以透過方程式 (8)來挑取頂點 u 。

$$Rcut^{(b)} = (W_{n-j} - (F^+(u))(\frac{1}{j+1} + \frac{1}{n-j-1}), \forall 0 \le j < n-1 \qquad \text{方程式 (8)}$$

令設 $\Psi^{(f)}$ 與 $\Psi^{(b)}$ 分別為 $p^{(f)}$ 與 $p^{(b)}$ 中的頂點集合。開始時令 $\Psi^{(f)} = \Psi^{(b)} = \varnothing$ 且 $\Psi^{(z)} = V$ ，在雙向排序流程中，使用往前排序在 $\Psi^{(f)}$ 與 $\Psi^{(z)} \bigcup \Psi^{(b)}$ 之間挑選有最小化切割值比率成本的頂點，以及使用往後排序在 $\Psi^{(f)} \bigcup \Psi^{(z)}$ 與 $\Psi^{(b)}$ 之間挑選有最小化切割值比率成本的頂點。

RRCP 二分法的過程類似 *TGS* 的方法，透過往前排序選擇來源頂點與往後排序選擇目的地頂點。然而，*TGS* 的目標是局部最佳解，且選擇的頂點是從 u 與 v 中選取有最小化切割值的頂點(最佳解)。而 *RRCP* 二分法的目標是廣域最佳解，選擇的頂點是 u 與 v 之中有著最大比率切割成本的頂點(次佳解)，其目的則是為了在下個排序中，尋找可能較佳二分的點。

一旦所有在 $\Psi^{(z)}$ 的頂點全都移動到 $\Psi^{(f)}$ 或 $\Psi^{(b)}$，則我們利用同樣的二分流程在 $\Psi^{(f)}$ 與 $\Psi^{(b)}$ 分別遞迴執行此二分法。

第五節　混合比率切割貪婪排序演算法

RTGS 是一個具有部份二分法能力的有效區域搜尋演算法，而 RRCP 分割演算法則使用全域優化方法應用在每個二分法。由於這兩個方法均是線性且是遞迴式運算，故可以保證迅速的收斂，因此此兩個方法可以結合成為一個新的有效混合演算法。

HRCGS 透過 RRCP 將有向無循環圖分割為二個子圖，且遞迴地各應用一個 HRCGS 於各個子圖中，來發展下個層級的 HRCGS 的解答，並且與 RTGS 的頂點序列相互比較以達到改善解答序列的目的。透過結合這兩個單獨解答的好處，開發出來的解答可以兼具兩者的優點。

圖 9 顯示 HRCGS 的虛擬碼。演算法初始先計算所輸入圖形的頂點並存入 p 中，接著使用 *HybirdSort* 的方法，遞迴式地排列頂點，副程式 *HybirdSort* 的主要功能是從 $(p_h,...,p_t)$ 中建立 *DAG*，並且回傳排序後的頂點序列。演算法首先，以 *Bisect* 程序執行 *RRCP* 演算法，將頂點分為二部份並且放置到 $(p_h,...,p_i),(p_{i+1},...,p_t)$ 這兩個序列中，再使用 *HybirdSort* 程序分別應用到兩個序列，並將此兩排序完的序列串接後之結果存入 S_x (lines#3)，另外運用 *RTGS* 演算法並將結果存入 S_y (lines#4)，再將 S_x 與 S_y 兩者相比較後傳回較佳的序列(lines#5-8)。

Procedure *HybridSort*(int h, int t)

1. if(t- h) ≤ 2 return RTGS(h,t)
2. i $\leftarrow$ Bisect(h, t)
3. $S_x \leftarrow$ HybridSort(h, i) || HybridSort(i+1, t)
4. $S_y \leftarrow RTGS$(h, t)
5. if $W^+(S_x) < W^+(S_y)$ then
6. return S_x
7. else
8. return S_y

Algorithm Hybrid-ratio-Cut-Greedy-Sort

9. *Global* : p
10. for i=1 to n
11. $p_i \leftarrow i$
12. $(p_1, p_2, ..., p_n) \leftarrow HybridSort(1, n)$

圖 9 HRCGS 的虛擬碼

第六節　複雜度分析

TGS 是拓樸排序演算法的另一種變異，計算所有來源(*source*)與目的地(sink)的 $F^+(u)$ 及頂點堆積(*heap*)建立，所需的時間為 $O(m+n\log n)$。而 *RTGS* 是一個遞迴執行 *TGS* 的演算法，其複雜度為 $O(m\log n+n\log^2 n)$。

假設建立來源(*source*)的 $F^+(u)$ 頂點與目的地(sink)的 $F^+(u)$ 使用相同的堆積(*heap*)結構，則 *RRCP* 二分過程的複雜度為 $O(m+n\log n)$，因此，*RRCP* 的時間複雜度為 $O(m\log n+n\log^2 n)$。

HRCGS 是混合 *RRCP* 與 *RTGS* 的演算法，使用全域優化的 *RRCP* 遞迴地二分 *DAG*，並在每個子圖中使用區域最佳 *RTGS*，以改善解答，因此複雜度為 $O(m\log^2 n+n\log^3 n)$。

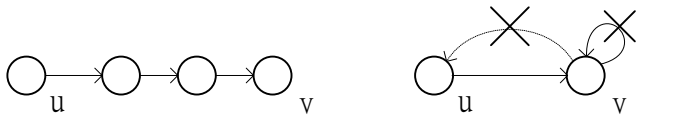
第四章 連續頂點對排序技術

(Successive Pair Ordering Technique)

連續頂點對排序列技術選擇尚未分出順序的頂點對並加以定出順序，並且計算已知頂點對的順序來形成一個有效的拓樸頂點排序，進一步地達到最小化邊延遲的目的。

壹、初始已排序頂點對集合

設$u \Rightarrow v$表示在頂點集合 V 裡存在一條從u到v的二元關係路徑，且此路徑關係有兩種特性：遞移性(Transitive)與非對稱性(Asymmetric)，如圖 10 所示，此路徑關係為部份排序，且此排序的頂點對$(V,\Rightarrow)$稱為部份有序集(partially ordered set, poset)。



遞移性(Transitive)　　非對稱性(Asymmetric)

圖 10　遞移性與非對稱性示意圖

設$u \prec v$表示現有的排序中u是放置在v之前，如圖 10 所示，關係$\prec$亦有遞移性(Transitive)與非對稱性(Asymmetric)，此關係也為部份排序，且此頂點對亦是部份有序集。

對於任意有效的拓樸排序，假設$u \Rightarrow v$，則u必然在v前，因此這部份有序集$(V,\Rightarrow)$必定為$(V,\prec)$的子集，表示如下。

$$(V,\Rightarrow) \subseteq (V,\prec)$$

因此，對於已識別的線性範圍$(V,\prec)$來說，$(V,\Rightarrow)$是初始的解集合；此外設T為一個二元矩陣，用來表示$\prec$關係，在此矩陣中所有元素，根據對應於$(V,\prec)$裡已排列的配對，分別將其值設為 0 或 1。在矩陣T裡每個元素的初始值是根據在對應的順序配對之間所存在的路徑。因此，矩陣T裡所有的元素的初始值如下。

$$T[u][v]=\begin{cases}1 & \text{if } u \Rightarrow v \\ 0 & \text{otherwise}\end{cases} \qquad \text{方程式 (9)}$$

使用已存在的頂點對路徑建立矩陣T之後，我們再一步接一步地新增每個附加的排序頂點配對來建立完全的排序集合。

貳、頂點對排序過程

假設$u \prec v$或$v \prec u$($T[u][v]=1$ or $T[v][u]=1$)，則u與v可以稱做已配對(comparable)頂點對，反之稱為可配對(incomparable)頂點對($T[u][v]=0$ and $T[v][u]=0$)。

設$u \prec\succ v$表示為即非$u \prec v$亦非$v \prec u$的二元關係；並且將放置在頂點 u 之前的頂點稱為祖先頂點(predecessor vertex)，在頂點u之後的稱為繼承頂點(successor vertex)。因此，設$\Delta_a(u)$表示所有包含頂點u以及所有頂點u的祖先頂點，以及$\Delta_p(u)$表示所有包含頂點u以及所有頂點u的繼承頂點。$\Delta_a(u)$與$\Delta_p(u)$表示如下。

$$\Delta_a(u)=\{u\}\bigcup\{z\,|\,T[z][u]=1\}$$
$$\Delta_p(u)=\{u\}\bigcup\{z\,|\,T[u][z]=1\}$$

在每次頂點對排序過程，假設可配對的頂點對(u,v)安排成$u \prec v$，則透過遞移性(Transitive)原理可以得知，所有在$\Delta_a(u)$的頂點亦安排所有在$\Delta_p(v)$的頂點之前，如下所示。

$$(x \prec y) \wedge (y \prec z) = x \prec z$$

因此，每次頂點排序過程可以觸發更多頂點對排序過程，換言之，安排u在v之前，則會安排所有$i \prec j,\ (i,j) \in R(u,v)$的頂點對，$R(u,v)$表示$u$安排在$v$之前的排序頂點對之集合，定義如下。

$$R(u,v) = \{(i,j) \mid i \prec\succ j, i \in \Delta_a(u), j \in \Delta_p(v)\}$$

此研究根據最大延遲縮減或是搶占邊方法，提出數個不同啟發的方法來安排可配對的頂點。

第一節　最大延遲縮減技術

在可配對頂點對(u,v)($T[u][v]=0$且$T[v][u]=0$)的情況下，假設u放於v前，則u到v的邊延遲會增加$F_{out}(u)+F_{in}(v)$，若u放在v之後，則邊延遲會增加$F_{out}(v)+F_{in}(u)$，因此，在$F^{+}(u)<F^{+}(v)$時，u放在v之前為最佳選擇，反之，則v在u之前為最佳選擇。

在每個頂點對排序時，若挑選有著最大$F^{+}(v)-F^{+}(u)$的可配對頂點對(u,v)，並且安排成$u\prec v$，則相對於$v\prec u$來說，此選擇有較大的邊延遲縮減。

壹、最大延遲縮減對演算法

令一開始時，已存在排序的頂點對集合$(V,\Rightarrow)$，我們的*MLRP*演算法使用反覆頂點對排序過程，來達成$(V,\prec)$。在每個頂點排序時，選擇有著最大$F^{+}(v)-F^{+}(u)$的可配對頂點對(u,v)，並且安排$i\prec j,(i,j)\in R(u,v)$的頂點對$(T[i][j]=1)$。

*MLRP*的時間複雜度為$O(n^4)$。圖 11 為基本的*MLRP*演算法虛擬碼。演算法開始時先初始化T作為完全部分有序集(line#1)，接著反覆頂點對排序過程(line#2-5)，最後透過識別T來完成所有頂點的順序p。

Algorithm Maximum-Latency-Reduction-Pair

1. build T with (V, $\Rightarrow$)
2. while T is not finished
3. find (u,v) pair with largest $(F^{+}(v) - F^{+}(u))$
   from both T[u][v]=0 and T[v][u]=0
4. for each (i,j) $\in$ R(u,v)
5. T[i][j] $\leftarrow$ 1
6. construct p from T

圖 11 *MLRP* 演算法的虛擬碼

貳、最大平均延遲縮減對演算法

MALRP 的方法在每一安排每個可配對頂點對 (u,v) 為 $u \prec v$ 的過程中，以批次的方式排序在 $R(u,v)$ 中所有頂點對。且在每次頂點對排序過程時，選擇在 $R(u,v)$ 中具有達到最大平均邊延遲縮減的最佳可配對頂點對 (u,v)。令矩陣 A 的所有元素為排序頂點對的平均邊延遲縮減的值，每個元素定義如下。

$$A[u][v] = \begin{cases} avg(\{F^{+}(j) - F^{+}(i) \mid (i,j) \in R(u,v)\}) & if \ \text{u} \prec\succ \text{v} \\ -\infty & otherwise \end{cases}$$

在此 $avg(\cdot)$ 表示每個元素的平均值，*MALRP* 演算法如同 *MLRP* 演算法，而每次在矩陣 A 中選擇最大元素的頂點對以找出應排序的頂點對。*MALRP* 頂點對排序過程包含頂點對的選擇，批次排序頂點對及更新矩陣 A。頂點對選擇階段，在矩陣 A 中選擇有著最大元素的頂點對 (u,v)；批次排序頂點對階段，安排每個頂點對，並使 $i \prec j, (i,j) \in R(u,v)$ $(T[i][j]=1)$；最後更新矩陣 A。

圖 12 顯示基本的*MALRP*演算法虛擬碼。因為建構A需要$O(n^4)$且A均需隨著每次頂點對排序過程而再次更新，因此*MALRP*演算法的時間複雜度為$O(n^6)$。*MALRP*演算法且有高實現的複雜度，因此，它的解接近最佳解[14]。

```
Algorithm Maximum-Average-Latency-Reduction-Pair
1. build T with (V,⇒)
2. build A with T
3. whild T is not finished
4.     find (u,v) pair with largest A[u][v]
5.     for each (i, j) ∈ R(u,v)
6.         T[i][j] ← 1
7.     update A
8. construct p form T
```

圖 12　*MALRP*演算法的虛擬碼

參、雙向貪婪排序與*MALRP*演算法

*TGS*演算法根據最小的下個切割值來決定選擇的頂點，而會導致較不佳的全域最佳解，*TGS – MALRP*則是一個雙向貪婪排序，每次均根據最大平均延遲縮減對的啟發來選擇最好的來源頂點與目的地頂點。

若u附加到$p^{(f)}$，則來源頂點u排列在所有它的可配對頂點之前，相同地，若u前置加到$p^{(b)}$，則目的地頂點u排列在所有它的可配對頂點之後，因此，附加u到$p^{(f)}$或前置加u到$p^{(b)}$頂均會決定頂點u與頂點u的可配對頂點之間的順序。令$AH(u)$表示頂點u附加到$p^{(f)}$

時的平均延遲縮減，且令 $AT(u)$ 表示頂點 u 前置加到 $p^{(b)}$ 時的平均延遲縮減。因此，$AH(u)$ 與 $AT(u)$ 是來源頂點與目的地頂點的價值函數，此價值函數可定義如下。

$$AH(u) = avg(F^{+}(v) - F^{+}(u) \mid v \prec\succ u)$$
$$AT(u) = avg(F^{+}(u) - F^{+}(v) \mid v \prec\succ u)$$

在 $TGS - MALRP$ 的頂點選擇階段，從所有的來源頂點中找出最大的 $AH(u)$ 以及從所有目的地頂點中找出最大的 $AT(v)$，並且比較 $AH(u)$ 與 $AT(v)$，若 $AH(u) > AT(v)$，則利用往前排序過程中，將 u 附加到 $p^{(f)}$，否則，利用往後排序過程中，將 v 前置加到 $p^{(b)}$。只有當 u 為來源頂點時才需計算 $AH(u)$，且當 u 為目的地頂點時才需計算 $AT(u)$，而且 $AH(u)$ 與 $AT(u)$ 均會隨著每次的往前排序或往後排序更新，直到頂點 u 附加至 $p^{(f)}$ 或前置加到 $p^{(b)}$。

由於 $AH(u)$ 與 $AT(u)$ 分成兩個部份儲存，其中包含總延遲縮減與可配對頂點數，更新單一個 $AH(u)$ 或 $AT(u)$ 的複雜度為 $O(1)$。因為建構矩陣 T 以及更新矩陣 T 的複雜度皆為 $O(nm)$，所有的 $AH(u)$ 與 $AT(u)$ 的建構與更新的複雜度為 $O(n^2)$，因此總體來說 $TGS - MALRP$ 演算法的複雜度為 $O(n(n+m))$。

肆、遞迴型 FM 二分法與 MALRP 演算法

傳統 FM 演算法移動頂點到不同的分割，藉此達到最小化不同分割之間的總切割值。[15]提出根據最大總延遲縮減來移動頂點，但此

方法可能會像使用 *TGS* 的方法而使得結果落入區域最小解。而 *RBFM – MALRP* 演算法則根據最大平均延遲縮減對的方法來移動頂點，以改善根據最大總延遲縮減來移動頂點的遞迴的 *FM* 分割演算法。

將 G 以二分法分化成 (V_1,V_2)，如果頂點 u 從 V_2 移動到 V_1，則在 V_2 中頂點 u 的所有可配對頂點對會排列在頂點 u 之後，同樣的，如果頂點 u 從 V_1 移動到 V_2，則在 V_1 中頂點 u 的所有可配對頂點對會排列在頂點 u 之前。我們選擇移動頂點到不同分割以最大化價值函數 $AS(V_1,V_2)$ 為依據，$AS(V_1,V_2)$ 為在 V_1 與 V_2 之間的平均延遲縮減且定義如下。

$$AS(V_1,V_2)=avg(F^+(v)-F^+(u)\,|\,u\in V_1,v\in V_2,u\prec\succ v)$$

令 $AN^{(1)}(u)$ 表示頂點 u 在 V_1 中的可配對頂點數，$AW^{(1)}(u)$ 表示 u 與 V_1 的頂點之間所有可配對頂點的總延遲縮減，而 $AN^{(2)}(u)$ 與 $AW^{(2)}(u)$ 各自表示頂點 u 在 V_2 中所有可配對的頂點數以及 u 與 V_2 的頂點之間可配對頂點的總延遲縮減。$AS(V_1,V_2)$ 可以表示為 $\frac{ASW}{ASN}$，ASW 表示每個在於 V_1 的頂點都領先於每個在於 V_2 的頂點之前的總延遲縮減，定義如下。

$$ASW=\sum_{u\in V_1}AW^{(2)}(u)=\sum_{u\in V_2}AW^{(1)}(u)$$

ASN 表示在於 V_1 與 V_2 之間可配對的頂點對總數，定義如下。

$$ASN=\sum_{u\in V_1}AN^{(2)}(u)=\sum_{u\in V_2}AN^{(1)}(u)$$

若尚未移動任何頂點到另一個分割之前，平均延遲縮減為$\frac{ASW}{ASN}$，若頂點$u \in V_1$且沒有繼承的頂點位於V_1，則頂點u從V_1移至V_2的平均延遲縮減改變為。

$$\frac{ASW + AW^{(1)}(u) - AW^{(2)}(u)}{ASN + AN^{(1)}(u) - AN^{(2)}(u)}$$

但假設頂點$u \in V_2$且沒有領先的頂點於V_2，則頂點u從V_2移至V_1的平均延遲縮減為。

$$\frac{ASW + AW^{(2)}(u) - AW^{(1)}(u)}{ASN + AN^{(2)}(u) - AN^{(1)}(u)}$$

$RBFM - MALRP$藉由執行 FM 的方法，反覆移動有著最大價值的頂點到另一分割，並鎖住u，且對每一$v \lessdot\gtrdot u$更新其$AN^{(1)}(v)$、$AW^{(1)}(v)$、$AN^{(2)}(v)$及$AW^{(2)}(v)$。每個重複動作的時間複雜度為$O(n)$，而單次二分法的時間複雜度則為$O(n^2)$。使用 FM 演算法遞迴地將圖形持續二分以形成完全排序的頂點順序，這樣時間複雜度為$O(n^2 \log n)$。實驗結果顯示利用$RBFM - MALRP$演算法能產生比使用分割比率的傳統FM二分法較佳的結果。

第二節　搶占邊技術

若建構的*DAG*只有邊延遲的優先順序，則只需最小化較高層階級的邊延遲。因此，每個可配對的頂點對之間的順序，可基於邊的優先順序關係透過重複的頂點對排序過程來完成。

在演算法使用搶占邊技術之前，所有邊依其優先順序來遞減排序，並分配從m(最高階級)到 1(最低階級)的優先值。

設(s_i,t_i)表示在第i階級的邊(邊i)中，s_i為開始頂點與t_i為結束頂點，並且令$Z_i(u)$表示為u是否位於邊i的排序階級中，當u為s_i、t_i、或在s_i與t_i之間的頂點時，$u=1$，否則為 0，$Z_i(u)$可定義如下。

$$Z_i(u)=\begin{cases}1 & \text{if } s_i \preceq u \preceq t_i \\ 0 & \text{otherwise}\end{cases} \qquad \text{方程式 (10)}$$

此處，二元關係$\preceq$表示等同於$\prec$，但具有反射性(reflexivity)，如圖 13 所示。

反射性(reflexivity)

圖 13　反射性

藉由安排u在v之前，增加邊i的延遲有以下條件。

1. u與v為尚未決定順序的頂點對，並$u \neq v$。
2. 至少有一頂點u或v是不在邊i階級的排序之中。

3. u 必須是位在 s_i 之後或 $u = s_i$，且 v 必須位在 t_i 或 $v = t_i$。

圖 14 顯示增加邊延遲的例子，在此例，安排 $u \prec t_i$ 或 $s_i \prec v$ 均會增加邊 i 的延遲。

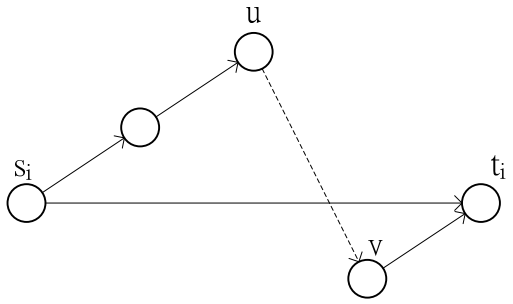


圖 14 安排 u 在 v 之前，透過增加延遲邊 i。

設 $S(u,v)$ 表示將 u 安排在 v 之前，所有會使邊延遲增加的頂點對集合。從上述的三個條件，$S(u,v)$ 可以由下列公式求出。

$$S(u,v) = \{i \mid u \prec\succ v,\ z_i(u) + z_i(v) < 2,\ v \preceq t_i,\ s_i \preceq u\}$$

安排 u 在 v 之前會影響在 $S(u,v)$ 中多個邊元素，因此，本研究試圖找出最小化最高階級的邊延遲。設矩陣 B 為一整數矩陣，且在矩陣 B 中的每個元素，皆為 S 中最高階級邊的值，矩陣 B 的每個元素定義如下。

$$B[u][v] = \begin{cases} \infty & \text{for v} \preceq u \\ \max(S(u,v)) & \text{for u} \prec\succ v, S(u,v) \neq \varnothing \\ 0 & \text{for u} \prec\succ v, S(u,v) = \varnothing \\ -\infty & \text{for u} \prec v \end{cases}$$

此處 $\max(\cdot)$ 表示最大的元素，假設 $u = v$ 或 v 安排在 u 之前，則 $B[u][v] = \infty$；若 $S(u,v)$ 不為空集合，則 $B[u][v]$ 受到設定 $u \prec v$ 之中最高

階級邊的影響，因此，$B[u][v]$ 為 $S(u,v)$ 中最大元素；若 $u \prec\succ v$ 且 $S(u,v)$ 為空集合，則 $B[u][v]=0$；最後如果 $u \prec v$ 則 $B[u][v]=-\infty$。

每個在矩陣 B 的元素代表著各個頂點對中所影響的最高階級的邊；對任意可配對頂點對 (u,v)，假設 $B[u][v] < B[v][u]$，則 u 安排在 v 之前 $(u \prec v)$，否則，v 安排在 u 之前 $(v \prec u)$。透過最大非無限的 $B[u][v]$ 可以找出在每次頂點對排序過程中最壞的頂點對排序，並且及安排 $v \prec u$ 來排除最壞可能的頂點對。

壹、搶占邊演算法

PE 演算法是一個反覆的依階級邊進行頂點對排序過程的簡易實作 *PE* 技術的方法。*PE* 演算法應用類似 *MALRP* 演算法的技術，在 *PE* 頂點對排序過程含有：頂點對選擇、批次排序頂點對以及更新矩陣 B。在頂點對選擇階段，選擇符合在矩陣 B 中最大非無限元素的可配對頂點對 (u,v)；在批次排序頂點對階段，設定 $i \prec j$，$(i,j) \in R(v,u)(T[i][j]=1)$；隨著批次排序之後，在每次安排 $i \prec j$ 時，每一 $k \in V$ 的 $B[k][i]$，$B[k][j]$，$B[i][k]$，$B[j][k]$ 均需更新。

初始化矩陣 B 的時間複雜度為 $O(n^2m)$，每次頂點對排序過程的複雜度受到 2 者控制，一為矩陣 B 中搜尋最大元素 $(O(n^2))$，另一為更新矩陣 $B(O(nm))$，由於頂點對排序過程最大為 n^2，因此整體複雜度為 $O(n^3m)$，為了改善效率，本篇研究提出一個更有效率的 PE 演算

法，此演算法藉由限制矩陣B中的搜尋與更新，以提高效率。

在本節，要定義 2 個定理

定理 3.對任意頂點$u \in v$且$(s_i, t_i) \in E$：

(i) 假設$u \prec\succ s_i$，則$S(u, s_i) \supseteq S(u, t_i)$。

(ii) 假設$u \prec\succ t_i$，則$S(t_i, u) \supseteq S(s_i, u)$。

證明(i) 假設$u \prec\succ s_i$且$u \preceq t_i$，則$S(u, t_i) = \varnothing$，故$S(u, s_i) \supseteq S(u, t_i) = \varnothing$，

在$u \prec\succ t_i$的情形下，設定$u \prec t_i$，所增加的延遲亦會因設定

$u \prec s_i$而增加，因此$S(u, s_i) \supseteq S(u, t_i)$。

(ii) 以相同的方法可以證明(ii)。

定理 4.對任意頂點$u, x, y \in V$且$u \prec\succ x$，$u \prec\succ y$，並且$u \Rightarrow y$，則：

(i) $S(u, v) \supseteq S(u, y)$且$S(y, u) \supseteq S(x, u)$

(ii) $B[u][x] \geq B[u][y]$且$B[y][u] \leq B[x][u]$

設$J^{(f)}(u)$表示頂點u的最小化的可配對來源頂點集合，例如：若$u \prec\succ v$，則$v \in J^{(f)}(u)$或者存在著一條從某一在$J^{(f)}(u)$中的頂點開始並且結束於v的路徑，此外，設$J^{(b)}(u)$表示頂點u的最小化可配對目的地頂點集合，例如：若$u \prec\succ v$，則$v \in J^{(b)}(u)$，或者存在一條從v開始並結束於某個$J^{(b)}(u)$中的頂點的路徑。

假設$x \in J^{(f)}(u)$，則對於u來講x為前面可配對頂點

(front-incomparable)，而 $x \in J^{(b)}(u)$ ，則對 u 來說 x 為後面可配對頂點(back-incomparable)， $J^{(f)}(u)$ 與 $J^{(b)}(u)$ 可定義如下。

$$J^{(f)}(u) = \{i \mid i \prec\!\succ u, (\nexists j)(j \Rightarrow i, j \prec\!\succ u)$$
$$J^{(b)}(u) = \{i \mid i \prec\!\succ u, (\nexists j)(i \Rightarrow j, j \prec\!\succ u)$$

方程式 (11)

從圖形中排除掉每個 $u \preceq x$ 的頂點 x 後，我們可以透過橫向優先搜尋(breadth first search)從每個來源頂點開始並結束於每個可配對頂點，以建立 $J^{(f)}(u)$ 。從圖形中排除掉每個 $x \preceq u$ 的頂點後，我們可以透過反向橫向優先搜尋(reverse breadth first search)從每個目的地頂點開始並結束於 u 的每個可配對頂點以建立 $J^{(b)}(u)$ 。如果 $u \in J^{(b)}(v)$ 而 $v \in J^{(f)}(u)$ 則 (u,v) 可以稱為前後可配對頂點對(back-incomparable and front-incomparable pair)，從定理 4(i)可知，(u,v) 是個前後可配對頂點對時，則 $R(v,u) = \{(v,u)\}$ 。從定理 4(ii)發現，若 (x,y) 不是一個前後可配對頂點對時，則至少存在著一個前後可配對頂點對 (u,v) 會使得 $x \Rightarrow u$(*or* x=u)， $v \Rightarrow y$(*or* y=v)，並且 $B[u][v] \geq B[x][y]$ ，因此，若需從矩陣 B 中找出最大非無限的元素與更新矩陣 B，可將其限制在矩陣 B 中的前後可配對頂點對，此外，若選擇的 (u,v) 為前後可配對頂點對，則在批次排序頂點對時，會縮減為單一頂點對排序，並且不受遞移性影響。

Algorithm Preemptive-Edge

1. build T with $(V, \Rightarrow)$
2. sort all edges into (s_i, t_i), $1 \le i \le m$
3. for each $u \in V$
4. build $J^{(f)}(u), J^{(b)}(u)$
5. $j \leftarrow \{(u,v) | u \in j^{(b)}(v), v \in J^{(f)}(u)\}$
6. compute B[u][v], $\forall (u,v) \in J$
7. while T is not finished
8. find (u,v) vertex pair with largest B[u][v] from J
9. T[v][u] $\leftarrow 1$
10. update $J^{(f)}(u), J^{(f)}(v), J^{(b)}(u), J^{(b)}(v)$ and J
11. recompute B[u][x], $\forall (u,x) \in J$
12. recompute B[x][u], $\forall (x,u) \in J$
13. recompute B[v][x], $\forall (v,x) \in J$
14. recompute B[x][v], $\forall (x,v) \in J$
15. construct p from T

圖 15 PE 演算法的虛擬碼

圖 15 為搶占邊演算法的虛擬碼，此演算法(line#1-6)為初始化階段，(line#7-14)反覆進行頂點配對。在初始化階段含有：一、依邊的階級，以遞減的方式排序所有的邊，以及建構每個頂點 u 的 $J^{(f)}(u)$ 與 $J^{(b)}(u)$；二、建立集合 J 來記錄所有前後可配對頂點對，以及從 $(u,v) \in J$ 中比較每個 $B[u][v]$。在每個頂點對排序過程，從 J 中選擇有最大非無限 $B[u][v]$ 的頂點對 (u,v)，並將 v 排在 u 之前 $(T[v][u]=1)$。$J^{(f)}(u)$、$J^{(b)}(v)$、$J^{(b)}(u)$、$J^{(f)}(v)$ 會隨著 $v \prec u$ 新建立的排序所影響如下。

i. 從 $J^{(f)}(u)$ 中移除 v，並對 v 的繼承頂點做橫向優先搜尋來更新 $J^{(f)}(u)$。

ii. 從 $J^{(b)}(v)$ 中移除 u，並對 u 所有的祖先頂點做反向橫向優先搜尋來更新 $J^{(b)}(v)$。

iii. 因為 v 會被安排在 u 的所有繼承頂點之前，故若 $u \in J^{(f)}(v)$，則可將 u 從 $J^{(f)}(v)$ 中直接移除，並且不需使用橫向優先搜尋來更新 $J^{(f)}(v)$。

iv. 因所有 v 的祖先頂點亦會被安排在 u 之前，故若 $v \in J^{(b)}(u)$，則可將 v 從 $J^{(b)}(u)$ 中直接移除，且不需使用反向橫向優先搜尋來更新 $J^{(b)}(u)$。

頂點對集合 J 會根據新的 $J^{(f)}(u)$、$J^{(b)}(v)$、$J^{(b)}(u)$ 以及 $J^{(f)}(v)$ 作更新，接著對每個 $(u,x) \in J$ 的 $B[u][x]$、$(x,u) \in J$ 的 $B[x][u]$、$(v,x) \in J$ 的 $B[v][x]$、以及 $(x,v) \in J$ 的 $B[x][v]$ 重新計算。

對於頂點 u 的橫向優先搜尋 $J^{(f)}(u)$ 與反向橫向優先搜尋 $J^{(b)}(u)$ 的時間複雜度皆為 $O(n+m)$，總體來說，在每個 $J^{(b)}(u)$ 與 $J^{(f)}(u)$ 運作的時間複雜度為 $O(n(n+m))$。由於搜尋與更新矩陣 B 只需限於前後可配對頂點對，故在頂點對排序過程之前矩陣 B 不需全部建構，運算此種方法可大幅降低在矩陣 B 中搜尋及更新的計算時間。

貳、雙向貪婪排序與搶占邊演算法

$TGS-PE$ 演算法是一個雙向貪婪排序演算法，此演算法根據邊的階級來排序每個頂點並利用類似 $TGS-MALRP$ 的雙向貪婪排序技術。

設 $BH_i(u)$ 表示當來源頂點 u 附加到 $p^{(f)}$ 可否避免邊 i 延遲的增加。若 $u \neq s_i$, $u \preceq t_i$ ，且 s_i 在 $p^{(f)}$ 或為目的地頂點時， $BH_i(u)=1$ ，否則， $BH_i(u)=0$ 。設 $BT_i(u)$ 表示當目的地頂點 u 前置加到 $p^{(b)}$ 可否避免邊 i 延遲的增加。若 $u \neq t_i$, $s_i \preceq u$ ，且 t_i 在 $p^{(b)}$ 或為來源頂點時， $BT_i(u)=1$ ，否則， $BT_i(u)=0$ 。往前排序過程中，利用每個 $BH_i(u)$ 來選擇最佳的來源頂點，以避免階級邊 i 的延遲增加，而往後排序過程中，則利用每個 $BT_i(u)$ 來選擇最佳目的地頂點，來避免階級邊 i 的延遲增加。

我們使用 $BH(u)$ 作為評估附加來源頂點 u 到 $p^{(f)}$ 的價值函數，此價值函數包括 m 個從最重要的元素 $BH_m(u)$ 到最不重要的元素 $BH_1(u)$ 。另外，使用 $BT(u)$ ，作為評估前置加目的地頂點到 $p^{(b)}$ 的價值函數，此價值函數包括 m 個從最重要的元素 $BT_m(u)$ 到最不重要的元素 $BT_1(u)$ ，透過從 m 到 1 的依次成對元素比較，若 $L=<L_m,L_{m-1},...,L_1>$, $L'=<L'_m,L'_{m-1},...,L'_1>$ ，且存在某一 i 使得 $L_i > L'_i$ 與 $L_j = L'_j$ ， $\forall j > i$ ，則必然 $L > L'$ 。

在雙向排序過程，從所有來源頂點中找出最大的$BH(u)$，以及從所有的目的地頂點中找出最大的$BT(v)$，兩者相比較，若$BH(u) > BT(v)$，則利用往前排序過程，將u附加進$p^{(f)}$，反之，則利用往後排序過程，將v前置加到$p^{(b)}$。

$TGS-PE$的時間複雜度為$O(n^2m)$，且此演算法藉由使用向前有效邊列表(forward active edge list)比較每個$BH(u)$，以及使用向後有效邊列表(backward active edge list)比較每個$BT(u)$，來進一步改善演算法。向前有效邊列表是指往前排序過程中可增加延遲的邊，而向後有效邊列表是指在往後排序中可增加延遲的邊，在往前排序過程，若s_i變為來源頂點，則邊i便加入向前有效邊列表，但如果t_i前置加到$p^{(f)}$則從此列表中移除邊i，而在往後排序，若t_i變為目的地頂點，則邊i便加入向後有效邊列表，但如果s_i附加到$p^{(b)}$則從此列表中移除邊i。在往前排序過程，比較所有$BH(u)$可以簡化為在向前有效邊列表中從最高階級邊到最低階級邊的順序來比較所有$BH_i(u)$，而在往後排序過程，比較所有$BT(u)$可以簡化為，在向後有效邊列表中，最高階級邊到最低階級邊的順序來比較所有$BT_i(u)$。

參、遞迴型 FM 二分法與搶占邊演算法

RBFM − PE 演算法是以遞迴的方式運行 *FM* 演算法，此演算法使用類似 *RBFM − MALRP* 演算法，反覆進行二分法過程來完成頂點的排序。設 $BS_i(V_1,V_2)$ 表示邊 i 透過二分法建立 (V_1,V_2) 分割所縮減的延遲，定義如下。

$$BS_i(V_1,V_2)=\sum_{u\in V_1,v\in V_2,B[v][u]=i}1-\sum_{u\in V_1,v\in V_2,B[u][v]=i}1 \qquad \text{方程式 (12)}$$

使用 $BS(V_1,V_2)$ 來作為頂點集合 V_1 與 V_2 之間分割的價值函數。$BS(V_1,V_2)$ 可以以 m 個正整數表示，從最重要的 $BS_m(V_1,V_2)$ 元素開始到最不重要的 $BS_1(V_1,V_2)$ 元素。我們的方法根據最大化 $BS(V_1,V_2)$ 的方法來選擇移動到不同分割的頂點，以將 G 二分為 (V_1,V_2)。

設 $BW^{(1)}$ 與 $BW^{(2)}$ 為各自地代表頂點 u 放置到 V_2 與 V_1 的兩個價值函數。$BW^{(1)}$ 是一個包含從 $BW_m^{(1)}(u)$ 到 $BW_1^{(1)}(u)$ 的 m 個正整數，且 $BW_i^{(1)}(u)=\sum_{v\in V_1,B[u][v]=i}1$，$BW^{(2)}(u)$ 是一個包含從 $BW_m^{(2)}(u)$ 到 $BW_1^{(2)}(u)$ 的 m 個正整數，且 $BW_i^{(2)}(u)=\sum_{v\in V_1,B[v][u]=i}1$，若頂點 $u\in V_1$，且頂點 u 沒有任何繼承頂點在 V_1，則頂點 u 移動到 V_2 的價值函數為 $BW^{(1)}(u)-BW^{(2)}(u)$，另外，若頂點 $u\in V_2$，且頂點 u 沒有任何祖先頂點在 V_2，則 u 移動到 V_1 的價值函數為 $BW^{(2)}(u)-BW^{(1)}(u)$。

有著最大價值的頂點 u 移動後，對每個 $v\lessgtr u$ 均需更新其 $BW^{(1)}(v)$ 與 $BW^{(2)}(v)$。更明確的說，若頂點 u 從 V_1 移到 V_2 且 $u\lessgtr v$，

$B[u][v]=i$，則 $BW_m^{(1)}(v)$ 減一，且 $BW_m^{(2)}(v)$ 加一。若頂點 u 從 V_2 移到 V_1 且 $u \lessdot\gtr v$ ， $B[v][u]=i$ ， 則 $BW_m^{(1)}(v)$ 加一 ， 且 $BW_m^{(2)}(v)$ 減一 。 $RBFM-PE$ 在每個二分法過程中重複地移動最大價值的頂點 u 到另一個分割，並鎖定 u 且對每個 $v \lessdot\gtr u$ 更新其 $BW^{(1)}(v)$ 與 $BW^{(2)}(v)$ 。

$RBFM-PE$ 每次重複的時間複雜度為 $O(nm)$ ，而單個二分法的複雜度為 $O\left(n^2m\right)$ ，因為 $RBFM-MALRP$ 使用遞迴 2 分的 FM 演算法以完成整個頂點列表，故整體此時間複雜度為 $O(n^2m\log m)$ 。

第五章 有向叢集技術

(Directed Clustering Technique)

許多研究講到叢集可以增加頂點的平均分支度(或圖形密度)並且可以改善像是反覆分割啟發(iterative partition heuristic)[16]與模擬退火放置(Simulated annealing placement)[17]的效能，因為頂點對排序技術是從分割演算法轉化而來，故亦可以透過使用叢集來預先處理，以對頂點對排序技術所提出的解答，進一步的改善其解答品質。

當結合(merge)數個頂點而成為一個大頂點時，我們以片斷(segment)圖形來取來頂點的圖形，片斷圖形為一個標準的圖形，而每個頂點皆為線性片斷。根據有向邊的權重以及 2 個要結合的斷片大小[18]，每個開始/結束片斷對(Star/end segment pair)之間的相依性以判斷函數(Objective function)來評估。S_i 到 S_j 的群組相依判斷函數可定義如下。

$$W_d(S_i,S_j)=\frac{1}{|S_i||S_j|}\sum_{u\in S_i,v\in S_j} w_{u,v}$$

設 *SDM* 表示為一 n x n 片斷相依矩陣，且每個元素定義如下。

$$SDM[i][j]=W_d(S_i,S_j)$$

在每個叢集過程中，我們選擇 *SDM* 中符合最大元素的 S_i / S_j 片斷對來有向結合(directed merge)片斷對。隨著有向結合片斷對之後，移除片斷 S_j ，並以新結合成的片斷 $S_i \| S_j$ 取代原來的 S_i ，且在 SDM 中第 i 列與第 i 欄均需更新如下。

$$SDM[x][i] = \frac{|S_j| SDM[x][j]}{|S_i| + |S_j|} + \frac{|S_i| SDM[x][i]}{|S_i| + |S_j|} \quad \forall x \neq i, j$$

$$SDM[i][x] = \frac{|S_j| SDM[j][x]}{|S_i| + |S_j|} + \frac{|S_i| SDM[i][x]}{|S_i| + |S_j|} \quad \forall x \neq i, j$$

此外，在 *SDM* 中每個第 i 列與第 j 欄均清除為零，$SDM[x][j] = 0, SDM[j][x] = 0$ 。

有向階層式聚合群集法 (Hierarchical agglomerative clustering,HAC)以聚合群集的方式合併有向無循環片斷中的頂點，在每個有向片斷結合時，選擇一個結束片斷 S_j 附加到開始片斷 S_i ，以將所有在 S_i 與 S_j 的頂點形成一個新的片斷 S_i 。

為了確定開始/結束片斷對可以有向結合，我們用拓樸排序來確認此圖形並不會因為結合此開始/結束片斷而形成有向循環圖。如果開始/結束片斷是沒有反向邊，且在測試結合時不為有向循環，則可以稱開始/結束片斷對為具備有向結合性(directed-mergeable)。令 *SNM* 表示為 n x n 的可結合片斷矩陣，每個元素定義如下。

$$SNM[i][j] = \begin{cases} 1 & \text{if } S_i / S_j \text{ is directed-mergeable} \\ 0 & \textit{otherwise} \end{cases}$$

在每次的叢集過程中，只有符合在*SNM* 中非零元素的片斷對才可以有向結合。在S_i / S_j片斷對結合之後，在*SDM* 中每個元素的第*i* 列，第*i* 欄與第 *j* 列，第 *j* 欄均需隨之更新。

因為有最大群組相依性的開始/結束片斷對並非一定具備有向結合性，[14]提出有向叢集演算法(directed clusting algorithm)，此演算法反覆的結合有最大群組相依性且具有向結合性的片斷對。然而有著最大群組相依性的片斷對並不一定具備有向結合性，因此執行此有向叢集演算法可能會令有些片斷對太早結合，導致落入次佳解。

第一節　等同邊叢集演算法

因為受限於有效的拓樸排序需求，安排有向叢集的*DAG*的頂點順序並不會如同在無向圖中安排頂點般地有效率的。假設在結合過程中有著最大群組相依性的開始/結束片斷對，不具有有向結合性，則此組群相依可以移除並轉移到它等同邊對(equivalent edge pair)。若有向無循環片斷圖 $G(V,E)$ ，且 $(S_i,S_j)\in E$ ，並且存在著 $S_i \Rightarrow S_x, S_x \Rightarrow S_j$ ，則在任何的拓樸排序中，S_x 必放在 S_i 與 S_j 之間。

定義 1. 設 $(S_i,S_j)\in E$ ， S_x 為頂點且 $S_i \Rightarrow S_x \Rightarrow S_j$ ，則 $(S_i,S_x),(S_x,S_j)$ 則為 (S_i,S_j) 的一個等同邊對。

假設最大群組相依片斷對 S_i / S_j 能有向結合，則有向結合 S_i / S_j ，否則 S_i 到 S_j 之間的邊權重轉移到任一等同邊對 $(S_i,S_x),(S_x,S_j)$ ，同時設 $SDM[i][j]=0$ ，且 $SDM[i][x]$ 與 $SDM[x][i]$ 更新如下。

$$SDM[i][x] += \frac{|S_j|}{|S_x|} SDM[i][j]$$

$$SDM[x][j] += \frac{|S_i|}{|S_x|} SDM[i][j]$$

經過此邊權重的轉移後，S_i / S_j 已不是一個有著最大群組相依的開始/結束片斷對，則叢集過程可以重新尋找下一個有著最大群組相依性的開始/結束片斷對。

Algorithm Equivalent Edges Clustering

1. Star each vertex as a singleton segment with $S_1, S_2, ...S_n$
2. $U \leftarrow \{S_1, S_2, ...S_n\}$, Initalize SDM, SMM
3. While $|U| > 1$
4. Find S_i, S_j with largest SDM[i][j] from elements in $S_i, S_j \in U$
5. If SMM[i][j]=1 then
6. $S_i \leftarrow S_i \| S_j, U \leftarrow U \setminus \{S_j\}$
7. Update SDM and SMM
8. Else
9. Find any S_x with $S_i \Rightarrow S_x \Rightarrow S_j$
10. SDM[i][x] $\leftarrow$ SDM[i][x]+$\frac{|S_j|}{|S_x|} SDM[i][j]$
11. SDM[x][j] $\leftarrow$ SDM[x][j]+$\frac{|S_i|}{|S_x|} SDM[i][j]$
12. SDM[i][j] $\leftarrow 0$
13. Sequentially put each vertex u in last segment into p

圖 16 EEC 演算法的虛擬碼

因為受限於需要以n^2的拓樸排序去建立*SMM*，所以等同邊叢集演算法的時間複雜度為$O(n^2m)$。

圖 16 顯示*EEC*演算法的虛擬碼，演算法一開始，每個頂點皆為單獨的片斷且初始化*SDM* 與*SMM* (line #1-2)，假設在每次叢集過程之前找到能有向結合的最大*SDM*[i][j]的開始/結束片斷對，則選擇此片斷對來有向結合並形成新的片斷。否則，在此片斷之間的邊權重會轉移到所選擇等同邊對(line #10-12)並且繼續尋找下個開始/結束片斷對。

第二節　使用有向叢集來適應演算法的預處理。

有向叢集可以結合其他演算法，其中將原始的輸入圖先結合數個頂點後轉化成具備有向叢集的片斷圖，再使用其他演算法加以求出解答，例如雙向拓樸排序技術(two-way topological sort technique)或連續頂點對排序技術(successive pair ordering technique)。我們先前提出的演算法稍作修改後可以很容易的應用在串接片斷圖。

壹、雙向拓樸排序演算法與串接頂點

在原始的演算法，往前排序是選擇最小的$F^+(u)$以及在往後排序中，選擇最大的$F^+(v)$，並且兩者相比較以求最佳解。如果應用在片斷圖中，則在往前排序中選擇最小的$\frac{|F^+(u)|}{|S_u|}$以及在往後排序中，選擇最大的$\frac{|F^+(v)|}{|S_v|}$，並且兩者相比較，以求最佳解。

在原始的演算法，頂點u同時為來源與目的地頂點時，若$F^+(u) < \frac{1}{|\Psi^{(z)}|}\sum_{v\in\Psi^{(z)}} F^+(v)$則視頂點$u$為來源頂點，反之則視頂點$u$為目的地頂點，如果應用在片斷圖中，若$\frac{F^+(u)}{|S_u|} < \frac{\sum_{v\in\Psi^{(z)}} F^+(v)}{\sum_{v\in\Psi^{(z)}}|S_v|}$則視頂點$u$為來源頂點，反之則視頂點$u$為目的地頂點。

貳、最大平均延遲縮減對演算法與串接頂點

在原始的演算法是根據最大的$F^+(v)-F^+(u)$來選擇頂點，如果應用在片斷圖中，則以最大的$\frac{F^+(v)}{|S_v|}-\frac{F^+(u)}{|S_u|}$來選擇頂點。

參、搶占邊演算法與串接頂點

在原始的演算法是以$w_{u,v}$來排序階級邊，如果應用在片斷圖中，我們則以$\frac{w_{u,v}}{|S_u|+|S_v|}$來排序階級邊。

第六章 實驗結果

在實驗中我們以隨機產生的多來源與多目的地的 DAG 來評估所有提出的演算法的效能。每個產生 DAG 的邊權重是依循著類齊夫分佈(zipf-like distribution)，設 $w^{(i)}$ 表示在已排序邊列表中，第 i 邊的邊階級權重，應用廣義的齊夫分佈(zipf distribution)，則每個 $w^{(i)}$ 有如下的特性。

$$w^{(i)} \propto \frac{1}{i^s} \qquad 1 \le i \le m$$

此處 s 為偏態系數(skew coefficient)，若 $s = 0$，齊夫分佈縮減為均衡分佈(uniform distribution) $w^{(i)} = \frac{1}{m}$，反之，s 的值越大，偏態分佈越高，每個模擬實驗的資料集包含 20 個隨機產生的 *DAG*。而對此些 *DAG* 我們評估在 $n = 50$ 時，$100 < m < 250$ 與 $0 < s < 2.0$ 的各個不同環境。

表 3 顯示出 16 個方法的績效比較，演算法的結果與從 Tsaih[14] 中提出的最佳排程解相比較。其中第一個方法是以拓樸排序隨機產生的初始解，在方法 2-7 依序為貪婪拓樸排序(Greedy Topological Sort, GS)、雙向貪婪拓樸排序(Two-way Greedy Sort, TGS)、遞迴雙向貪婪拓樸排序(Recursive Two-way Greedy Sort, RTGS)、遞迴比率切割區塊演算法(Recursive Ratio Cut Partition Algorithm, RRCP)、混合比率切割貪婪排序演算法(Hybrid Ratio Cut Greedy Sort Algorithm, HRCGS)以

及一個使用具有比率切割的遞迴 FM 二分法(Recursive FM Bipartitioning，RBFM)，在方法 8-14 依序為最大延遲縮減對演算法(Maximum Latency Reduction Pair Algorithm, MLRP)、最大平均延遲縮減對演算法(Maximum Average Latency Reduction Pair Algorithm, MALRP)、雙向貪婪排序與 MALRP 演算法(Two-Way Greedy Sort with MALRP Algorithm, TGS-MALRP)、遞迴型 FM 二分法與 MALRP 演算法 (Recursive FM Bipartitioning with MALRP Algorithm, RBFM-MALRP)、搶占邊演算法(Preemptive Edge Algorithm, PE)、雙向貪婪排序與搶占邊演算法(Two-Way Greedy Sort with PE Algorithm, TGS-PE)、遞迴型 FM 二分法與搶占邊演算法(Recursive FM Bipartitioning with PE Algorithm, RBFM-PE)，最後方法 15 為有向叢集(directed clustering, DC)是由[14]提出，以且方法 16 為等同邊叢集(equivalent edges clustering, EEC)。

GS、TGS、RTGS 與 HRCGS 的比較：HRCGS 優於 RTGS，RTGS 優於 TGS，TGS 優於 GS，但在均衡分佈邊權重($s=0$)的情形中，TGS 並非始終勝於過 GS。

RRCP 與 RTGS 的比較：RRCP 與 RTGS 均是從兩端放置頂點到 p，並遞迴地最佳化在 p 中每個二分割內的頂點。RTGS 執行在較均衡邊權重圖時較佳，而 RRCP 執行在較偏斜邊權重圖時較佳。

MALRP 的效能分析：由於它的高複雜度且 MALRP 演算法的結果是接近最佳演算法的結果，*MALRP* 始終優於其他演算法。

RBFM 與 RBFM-MALRP 的比較：因為 RBFM-MALRP 是使用 MALRP 作為價值函數，而 RBFM 是以比率切割作為價值函數，所以在相同時間複雜度之下，RBFM-MALRP 優於 RBFM。

TGS 與 TGS-MALRP 的比較：TGS 與 TGS-MALRP 均為區域最佳演算法，而兩者均在 p 的兩端放置頂點；因為 TGS-MALRP 演算法是使用 MALRP 作為價值函數，而 TGS 是以 $F^{+}(u)$ 作為價值函數，故 TGS-MALRP 優於 TGS。

PE 的效能分析：在 *PE* 演算法中只考量每個邊的優先順序，故 *PE* 演算法在較偏斜權重圖時執行效能較好。

TGS-PE 與 RBFM-PE 的比較：TGS-PE 與 RBFM-PE 皆實施搶占邊演算法。邊的數量較少時，TGS-PE 優於 RBFM-PE，而在邊的數量較多時 RBFM-PE 優於 TGS-PE。

DC 與 EEC 的比較：藉由使用等同邊取代原始邊，EEC 演算法是一種改良的 DC 演算法，在不同的偏斜值之下，EEC 演算法是均優於 DC 演算法。

第七章 結論

我們提出數個啟發式的方法來解決有向線性排程問題。其中我們可證明混合式比率切割貪婪排序演算法在較偏斜的邊權重圖中提出的解是個高品質的近似解。我們亦應用 MALRP 提出一個新的價值函數來導引最佳頂點順序的搜尋，且每個使用 MALRP 函數的演算法均優於傳統以切割值為基礎的演算法。搶占邊演算法可應用在只有邊階高低資訊的環境中而以邊階級優先為根據來決定最佳頂點順序，搶占邊演算法在較偏斜的邊權重圖中能提供高品質的解答。此外，等同邊叢集演算法結合了最大等同群組相依的片斷對，能在有向圖的環境中大幅地改善單獨使用一般叢集的方法。

參考文獻

1. Kun-Feng, L. and L. Chuan-Ming. *Schedules with minimized access latency for disseminating dependent information on multiple channels*. in *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*. 2006.
2. Sih, G.C. and E.A. Lee, *A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures.* Parallel and Distributed Systems, IEEE Transactions on, 1993. **4**(2): p. 175-187.
3. Cong, J. and L. Sung Kyu. *Performance driven multiway partitioning*. in *Design Automation Conference, 2000. Proceedings of the ASP-DAC 2000. Asia and South Pacific*. 2000.
4. Bar-Noy, A., J. Naor, and B. Schieber, *Pushing dependent data in clients-providers-servers systems.* Wirel. Netw., 2003. **9**(5): p. 421-430.
5. Wikipedia. *Leonhard Euler*. 2001; Available from: http://en.wikipedia.org/wiki/Leonhard_Euler.
6. Sungho, K. *Linear Ordering and Application to Placement*. in *Design Automation, 1983. 20th Conference on*. 1983.
7. Saab, Y.G., *An improved linear placement algorithm using node compaction.* Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 1996. **15**(8): p. 952-958.
8. Sakellariou, R. and H. Zhao. *A hybrid heuristic for DAG scheduling on heterogeneous systems*. in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*. 2004.
9. Jain, A.K., M.N. Murty, and P.J. Flynn, *Data clustering: a review.* ACM Comput. Surv., 1999. **31**(3): p. 264-323.
10. Safro, I., D. Ron, and A. Brandt, *Graph minimum linear arrangement by multilevel weighted edge contractions.* J. Algorithms, 2006. **60**(1): p. 24-41.
11. Mcallister, A.J., *A New Heuristic Algorithm For The Linear Arrangement Problem.* Faculty of Computer Science; University of New Brunswick, 1999.

12. Cheng, C.K. and Y.C.A. Wei, *An improved two-way partitioning algorithm with stable performance [VLSI]*. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 1991. **10**(12): p. 1502-1511.
13. Patkar, S.B. and H. Narayanan. *An efficient practical heuristic for good ratio-cut partitioning*. in *VLSI Design, 2003. Proceedings. 16th International Conference on*. 2003.
14. Tsaih, D., *Optimal Solution to the Directed Linear Arrangement problem with a Hybrid Heuristic*. submitted for review,anonymous available at ftp://www1.ec.nhu.edu.tw.
15. Fiduccia, C.M. and R.M. Mattheyses. *A Linear-Time Heuristic for Improving Network Partitions*. in *Design Automation, 1982. 19th Conference on*. 1982.
16. Saab, Y., *A Contraction-based Ratio-cut Partitioning Algorithm.* VLSI Design, 2002. **15**(2): p. 485-489.
17. Wern-Jieh, S. and C. Sechen, *Efficient and effective placement for very large circuits.* Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 1995. **14**(3): p. 349-359.
18. Karger, D.R., *Global min-cuts in RNC, and other ramifications of a simple min-out algorithm*, in *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*. 1993, Society for Industrial and Applied Mathematics: Austin, Texas, United States. p. 21-30.

表 3　各演算法的實驗結果比較

	|E|=100 Opt=9.64	|E|=150 Opt=11.81	|E|=200 Opt=13.24	|E|=250 Opt=14.06
Random	11.62 (20.56%)	12.61 (6.82%)	13.66 (3.22%)	14.30 (1.68%)
GS	10.76 (11.62%)	12.18 (3.17%)	13.39 (1.13%)	14.14 (0.55%)
TGS	10.73 (11.28%)	12.26 (3.82%)	13.36 (0.95%)	14.11 (0.36%)
RTGS	10.46 (8.48%)	12.02 (1.81%)	13.33 (0.69%)	14.08 (0.12%)
RRCP	10.49 (8.83%)	12.03 (1.84%)	13.26 (0.18%)	14.11 (0.35%)
HRCGS	10.21 (5.89%)	11.92 (0.94%)	13.29 (0.36%)	14.07 (0.08%)
RBFM	10.15 (5.29%)	11.94 (1.11%)	13.31 (0.52%)	14.08 (0.10%)
MLRP	9.79 (1.56%)	11.88 (0.63%)	13.27 (0.25%)	14.07 (0.07%)
MALRP	9.65 (0.15%)	11.81 (0.00%)	13.24 (0.00%)	14.06 (0.00%)
TGS-MALRP	10.35 (7.31%)	11.96 (1.28%)	13.31 (0.54%)	14.09 (0.17%)
RBFM-MALRP	9.80 (1.64%)	11.86 (0.44%)	13.25 (0.11%)	14.07 (0.07%)
PE	10.90 (13.10%)	12.36 (4.65%)	13.44 (1.54%)	14.17 (0.76%)
TGS-PE	11.27 (16.87%)	12.38 (4.84%)	13.45 (1.56%)	14.17 (0.81%)
RBFM-PE	10.91 (13.14%)	12.44 (5.34%)	13.48 (1.81%)	14.18 (0.82%)
DC	10.73 (11.29%)	12.21 (3.44%)	13.55 (2.36%)	14.24 (1.24%)
EEC	10.14 (5.18%)	12.14 (2.83%)	13.49 (1.87%)	14.20 (1.01%)
s=0 , |V|=50				

各演算法的實驗結果比較(續)

	|E|=100 Opt=7.60	|E|=150 Opt=9.81	|E|=200 Opt=10.70	|E|=250 Opt=11.60
Random	10.94 (43.90%)	11.72 (19.56%)	11.84 (10.64%)	12.35 (6.52%)
GS	8.89 (16.97%)	10.38 (5.90%)	10.96 (2.43%)	11.72 (1.04%)
TGS	8.78 (15.54%)	10.27 (4.78%)	10.85 (1.37%)	11.67 (0.60%)
RTGS	8.36 (9.95%)	9.95 (1.49%)	10.72 (0.18%)	11.61 (0.09%)
RRCP	8.22 (8.21%)	10.07 (2.70%)	10.80 (0.91%)	11.66 (0.57%)
HRCGS	8.00 (5.26%)	9.93 (1.30%)	10.73 (0.21%)	11.60 (0.06%)
RBFM	7.94 (4.55%)	9.95 (1.50%)	10.77 (0.58%)	11.62 (0.21%)
MLRP	7.92 (4.24%)	9.99 (1.90%)	10.73 (0.26%)	11.61 (0.12%)
MALRP	7.61 (0.12%)	9.81 (0.00%)	10.70 (0.00%)	11.60 (0.00%)
TGS-MALRP	8.34 (9.74%)	10.12 (3.22%)	10.78 (0.73%)	11.65 (0.46%)
RBFM-MALRP	7.77 (2.29%)	9.85 (0.42%)	10.74 (0.32%)	11.60 (0.01%)
PE	8.09 (6.51%)	10.17 (3.73%)	10.78 (0.75%)	11.65 (0.46%)
TGS-PE	8.40 (10.64%)	10.23 (4.34%)	10.80 (0.89%)	11.65 (0.43%)
RBFM-PE	8.32 (9.48%)	10.33 (5.33%)	10.91 (1.89%)	11.74 (1.23%)
DC	9.02 (18.72%)	10.86 (10.77%)	11.59 (8.31%)	12.17 (4.95%)
EEC	7.99 (5.23%)	10.14 (3.39%)	10.88 (1.61%)	11.79 (1.73%)
s=1.0 , |V|=50				

各演算法的實驗結果比較(續 2)

	|E|=100 Opt=5.40		|E|=150 Opt=7.66		|E|=200 Opt=8.53		|E|=250 Opt=8.98	
Random	10.37	(92.10%)	11.10	(45.02%)	10.32	(21.00%)	10.20	(13.63%)
GS	6.59	(21.98%)	8.08	(5.54%)	8.88	(4.11%)	9.12	(1.54%)
TGS	6.04	(11.89%)	7.91	(3.21%)	8.60	(0.74%)	9.09	(1.17%)
RTGS	5.78	(7.06%)	7.90	(3.12%)	8.54	(0.05%)	9.12	(1.49%)
RRCP	5.52	(2.27%)	7.73	(1.00%)	8.56	(0.33%)	8.99	(0.04%)
HRCGS	5.45	(0.83%)	7.69	(0.49%)	8.54	(0.09%)	8.98	(0.01%)
RBFM	5.64	(4.40%)	7.72	(0.86%)	8.54	(0.07%)	9.00	(0.14%)
MLRP	5.47	(1.28%)	7.70	(0.52%)	8.54	(0.10%)	8.98	(0.01%)
MALRP	5.40	(0.02%)	7.66	(0.00%)	8.53	(0.00%)	8.98	(0.00%)
TGS-MALRP	5.68	(5.23%)	7.75	(1.20%)	8.57	(0.47%)	9.01	(0.28%)
RBFM-MALRP	5.52	(2.26%)	7.71	(0.69%)	8.56	(0.27%)	8.99	(0.07%)
PE	5.47	(1.32%)	7.72	(0.86%)	8.54	(0.06%)	8.98	(0.04%)
TGS-PE	5.62	(4.00%)	7.76	(1.34%)	8.54	(0.16%)	8.99	(0.07%)
RBFM-PE	5.59	(3.53%)	7.85	(2.48%)	8.65	(1.33%)	9.05	(0.74%)
DC	8.44	(56.29%)	9.49	(24.00%)	9.98	(17.02%)	10.12	(12.61%)
EEC	5.50	(1.89%)	7.77	(1.43%)	8.62	(1.06%)	9.06	(0.92%)
s=2.0 , |V|=50								