線性時間座席指派演算法及其應用

An *O*(*N*) Seat Assignment Algorithm

詹景裕	陳泰均	李明哲	駱超民
國立臺北大學	國立臺北大學	銘傳大學	底特律大學
電機工程研究所	電機工程研究所	資訊傳播工程學系	電機與電腦學系
gejan@mail.ntpu.edu.tw	s79782301@webmail.ntpu.edu.tw	leemc@mcu.edu.tw	c2luo@cheetah.vlsi.uwaterloo.ca

摘要

本文主要目的是提出一種快速的座席指派演算法，應用在火車座位管理系統上，本演算法靈感來自 Hashimoto 和 Steven 提出的左端點演算法(Left-Edge Algorithm, LEA) [4]。傳統的左端點演算法可以達到最佳軌道(Track)數，但在資料整理上仍使用搜尋(Search)使得整體時間複雜度需要 $O(N^2)$，而本文的演算法則提出在運用 Buckets 資料結構下，當行(column)數為常數時，則時間複雜度將可降低至 $O(N)$ 並且保持指派後的結果也為最少軌道(座席)數。

關鍵詞：線段繞徑、通道指派、筒狀資料結構、火車座席管理系統

Abstract

This paper presents a fast channel assignment algorithm based on the famous Left-Edge Algorithm (LEA) and the data structure of buckets for printed circuit boards, electronic design automatic, and internet scheduling. Meanwhile, this algorithm can be applied to rails traffic management system with the minimal number of seats. The time and space complexities are both $O(N)$, where N is the number of tickets under the condition that the number of stations is limited.

Keywords: buckets, channel assignment, wire routing, rails traffic management system

一、簡介

通道繞徑(Channel Routing)技術是一門在電子設計自動化中廣被研究的領域，在通道繞徑的問題中，若只注重水平線段上的軌道排列，則稱為通道指派問題(Channel Assignment)，由於通道指派問題只注重水平線段而不注重垂直方向，因此如何把使用的軌道數壓縮到最小才是被關注的重點。而通道指派演算法應用層面非常廣泛，例如:巢狀網路(Cellular Networks) [2, 3, 6, 9]、行程配置(Scheduling) [1, 7, 12]、通道繞徑(Channel Routing) [11, 13]及印刷電路板[4]，在通道指派問題中最典型的四種演算法分別為 LEA 演算法、圖形限制演算法(Constraint Graph Algorithm)、貪婪演算法(Greed Algorithm)跟階層制演算法(Hierarchical Algorithm)，然而 LEA 演算法是由 Hashimoto 和 Steven 於一篇使用通道指派技術於雙層電子印刷版(Printed Circuit Board, PCB)上的文章 [4]中所提出，不過當時他們方法的時間複雜度為 $O(N^2)$，之後雖然有人改用以排序方式來取代以搜尋為基礎的 LEA [10]，但降低的時間複雜度仍然為 $O(N \log N)$。然而本文運用不同的資料結構 Buckets 後，當行數為常數時，其指派的時間複雜度由 $O(N \log N)$降低至 $O(N)$並維持傳統 LEA 演算法可指派出最少軌道數之優點，而此限制條件相當適合運用在火車座席管理系統上。

本文之文章架構說明如下：第二節將問題定義及介紹所需的變數名稱，第三節介紹線性時間通道指派演算法；效能分析及證明則安排在第四節，第五節提出一個實例來幫助說明，最後本文在第六節總結。

二、問題定義及變數介紹

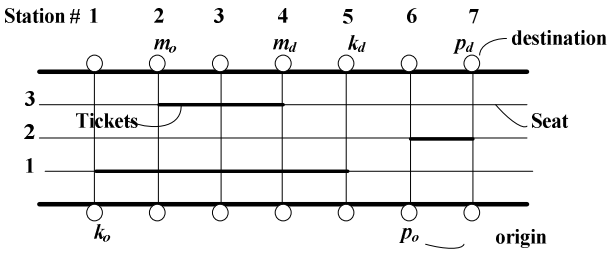
在以往定義通道指派演算法問題時大都會使用一個典型的模型來幫助說明，此模型通常是一個有兩個平行腳位列(Rows of Pins)的長方形空間，腳位(Pin)的位置是固定的並且上下腳位會連成垂直線(Column)以及水平向擺放線段的軌道(Tracks)，讓有著相同標籤(Tag)連結的腳位可以用一條專有的水平線段(Trunk)及兩垂直向線段(Branch)來相通，而通道指派演算法重點只專注在水平線段的指派與本文應用在座席指派上有著異曲同工之妙，只需將 Column 改為

Station、Tracks 改成 Seats 和 Trunks 改為 Tickets 即可，架構圖為圖 1(a)所示。本文問題為以最少的座位數容納下所有的車票需求，如圖 1(b)。為了方便索引及參考，本文所使用之變數符號之定義如表 1 所示。

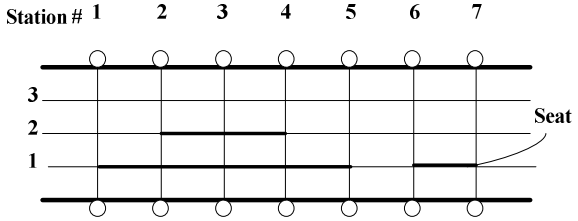
三、座席指派演算法

本演算法是以傳統的 LEA 演算法為基本，利用不同的資料結構 Buckets 降低 Tickets ordering 時間，在行數為常數時，時間複雜度會從 $O(N \log N)$降低至 $O(N)$，在此本文將演算法分為兩個步驟，Step 1 為資料結構變更，Step 2 則是以 LEA 執行通道指派。

Step 1 改進傳統使用搜尋或排序的資料結構，以使用不同的資料結構來改善時間，首先要將原始資料存入一個結合矩陣及連結串列優點的資料結構 Buckets，然後將線段依照其起始站的站名(Station #)以同號一維矩陣連結的鏈結串列串連起來，以此資料結構可快速完成車票的 ordering 放進 Buckets 中相對應之串列的結果，卻只需要花費 $O(N)$的時間。Step 2 則是使用類似 LEA 的概念進行指派，將 Buckets 中 ordering



(a)座席指派問題模型



(b)執行結果

圖1 座席指派問題定義 [8]

表 1 參數與變數表

變數名稱	名詞解釋
S_j	資料結構為 Buckets ，Stations 的集合 $(1 \le j \le M)$。
s_k	資料結構為 Buckets ，Seats 的集合，$(1 \le k \le L)$。
$T_i(o_i, d_i, t_i)$	Tickets 的集合$(1 \le i \le N)$。第 i 張車票會存有有起始點 o_i(origin)跟目的地 d_i(destination)和車票的時間變數 t_i，即訂票的時間。

之車票，由左而右依序地依 LEA 演算法進行指派，本文是以站號最小的一維陣列串連的連結串列當作 LEA 的最左邊，將其指向到座席的連結串列並將其從 Buckets 中刪除，之後再找尋無重疊的線段放入軌道。

Algorithm: Channel Assignment

Begin
Step 1 呼叫 Tickets Ordering Function
Step 2 呼叫 Left-Edge Assignment Function
End

Function 1: Tickets Ordering
Begin
Step 1 將原始資料中所有車票依照其起始站依序插入同值之 Station Bucket($S_{j=}\ o_i$)。
End

Function 2: Left-Edge Assignment
Begin
Step 1 從非空(non-empty)的最小 Station Bucket 中取出第一張車票插入編號最小且未滿的 Seat Bucket。
Step 2 尋找下一個可插入的車票。
  Step 2.1 尋找 Station Bucket 編號大於上一張指派車票 d_i 且最接近的非空 Station Bucket。
  Step 2.2 從 Station Bucket 中取出第一張車票插入目前的 Seat Bucket，回到 Step 2.1。
Step 3 依序將剩下之車票填入所需之座席。
  Step 3.1 若 Station Bucket 為空，則離開演算

法，否則回到 Step 1。

End

四、效能分析

關於快速座席指派演算法的空間複雜度分析，很明顯地該演算法由於一開始需儲存 N 張車票的起點跟終點，所以原始資料的時間複雜度為 N。另外指派所需要的暫存空間需求為一個一維陣列 M 加上資料數 N 即 N 張，最後還需要一個 L 的一維矩陣來存通道指派後的結果，因為 M 為常數且 $L < N$ ，所以本演算法的空間複雜度 (Space Complexity) SC_{Total} 為：

$$SC_{Total} = SC_T + (SC_S + SC_s)$$
$$= N + (M + N + L)$$
$$= O(N)$$

關於快速座席指派演算法的時間複雜度分析，可分為兩步驟來探討，Step 1 中分別將每組車票存入原始資料的時間，最多有 N 張，其時間複雜度 $O(N)$，並把線段資料從原始資料中複製到 Station Buckets，從原始資料連結到 Station Buckets 必須花掉 $O(1)$，最多有 N 張，其時間複雜度 $O(1) \times N$ 即 $O(N)$，因此步驟的時間複雜度 TC_{Step1}：

$$TC_{Step1} = O(N)$$

在 Step 2 時，依序指派每一組車票，從 Station Buckets 內移動到 Seat Buckets 必須花掉 $O(1)$，最多有 N 張車票，所以為 $O(N)$，之後尋找沒有與已指派座位中車票重疊的線段並放入，因為利用 Buckets 尋找下一張車票只需要 $O(M)$，總時間為 $O(M) \times N$，但 M (Station #)在本應用中一般可視為常數，因此 Step 2 的時間複雜度 TC_{Step2} 為：

$$TC_{Step2} = O(N) + O(M \times N)$$
$$= O(N)$$

綜合 Step 1 跟 2 的時間複雜度可知道該演算法的時間複雜度的最大需求為 $O(N)$，因此該演算法的時間複雜度(Time Complexity)為 $O(N)$，即：

$$TC_{Total} = TC_{Step1} + TC_{Step2}$$
$$= O(N)$$

定理 1:本演算法為最少座席指派(Minimum Seat Assignment)，可保證座位數為最少並且時間複雜度為最快。

證明:

Hashimoto 和 Steven 在其文章[4]中已證明使用 LEA 後，所指派的軌道數會為最少軌道數，該文章[4]核心步驟為:

1. 排序線段的左端點。
2. 選擇有最小左端點線段放入編號最小的軌道中，同本文 Function 2 之 Step 1。
3. 依排序後結果，將有最小左端點並且不會與步驟 2 中已指派線段重疊之線段再放入步驟 2 指派的軌道，同本文 Function 2 之 Step 2 和 3。
4. 當已無符合不重疊或指派軌道已滿，則回到步驟 2，則與本文 Function 2 之 Step 3 相同。

因此除了在 Step 1 本文不是使用排序來整理資料外，其他步驟皆與 LEA 相同，所以就方法上而言，本演算法與 LEA 是相同的，然而 LEA 已經證明其指派後可得到最少軌道數，故間接證明本文之座席指派演算法也可求出最少座位解。

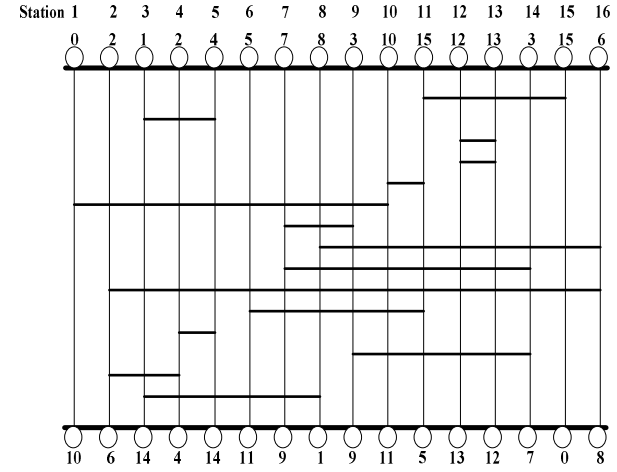
因為本文之演算法 Step 1 是使用 Bucket 來進行資料整理(Ordering)而非使用排序，所以不會受到時間複雜度 $O(N \log N)$的限制，進而將時間複雜度降至 $O(N)$，再加上 M(Station #)一般可視為常數，所以本演算法是一個時間最快且座席數最少的座席指派演算法。

五、演算法執行範例

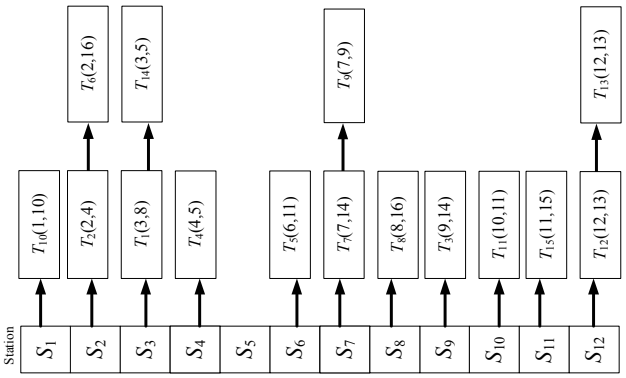
根據上述的演算法本文以下列一個範例來輔助說明整個演算法的過程，讓讀者更清楚了解演算法的內容。

在圖 2(a) 上面有 16 張車站扣除無人買票的車站 0 後，則有 15 張車票須要進行指派。然後利用快速座席演算法進行指派後，很明顯的將車票依照其起始站及終點站進行連接，在還沒有指派之前，各個車票都佔用一個座位，因此剛開始佔用了 15 個座位數。而本文則是要將這些車票進行

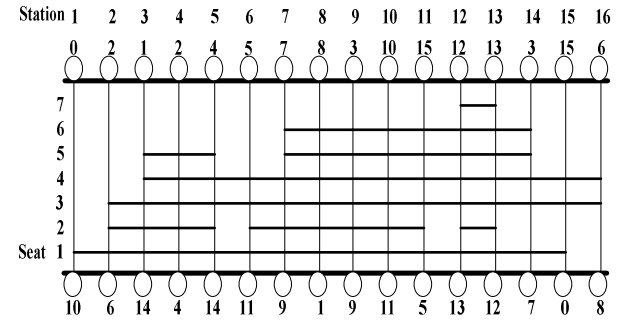
指派以期達到座位數減少的效果。將座位數量從原本的 15 個大幅降低至 7 個，而 7 個則為最小座位數，結果如圖 2 (c) 所示。因此成功的完成軌道數減少的目的，但本文之時間複雜度卻降低至 $O(N)$。



(a) 所有的車票



(b) 存入 Station Buckets



(c) 指派後結果

圖 2 演算法範例

當車票資料已經放入 Station Buckets 處理過後的情形則在圖 2(b)表示，為了方便閱讀，故將其資料概念轉以實體圖形來顯示，S_1 表示第一個 Station 其後面鏈結的是所有起始站在 Station 1 的車票資料。

六、結論與未來展望

本文提出了一個快速的座席指派演算法，此演算法是利用新的資料分類方式來取代原本傳統 LEA 使用搜尋或排序的方式，也因為這個原因，在行數為常數時，其時間複雜度將從 $O(N^2)$ 以及 $O(N \log N)$降為至 $O(N)$並且仍然可以維持指派後為最少座位數。未來企圖加上時間變數 t_i 使本演算法成為動態性或加入團體購票座席相鄰之功能。

參考文獻

1. D. Bozda, F. Ozgner, and U.V. Catalyurek, “Compaction of Schedules and a Two-Stage Approach for Duplication-Based DAG Scheduling,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 6, 2009, pp. 857-871.
2. T. Chiueh, K. Gopalan, and A. Raniwala, “Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks,” *ACM SIGMOBILE Mobile Computing and Communications Review archive*, vol. 8, no. 2, 2004, pp. 50-65.
3. A. J. Goldsmith, and S.B. Wicker, “Design challenges for energy-constrained ad hoc wireless networks,” *IEEE wireless communications*, vol. 9, no. 4, 2002, pp. 8-27.
4. A. Hashimoto, and J. Stevens, “Wire routing by optimizing channel assignment within large apertures,” *Annual ACM IEEE Design Automation Conference, Proceedings of the 8th Design Automation Workshop*, 1971, pp. 155-169.
5. S.E. Lass, “Automated printed circuit routing with a stepping aperture,” Communications of the ACM, vol. 12,no 5, 1969, pp. 262 - 265.

6. Y. Lee, K. Kim, and Y. Choi, “Optimization of AP placement and channel assignment in wireless LANs,” Annual Proceedings of 27th IEEE Conference on Local Computer Networks, 2002, pp. 831-836.
7. M.P. McGarry, M. Reisslein, C.J. Colbourn, M. Maier, F. Aurzada, and M. Scheutzow, “Just-in-time scheduling for multichannel EPONs,” *Journal of Lightwave Technology*, vol. 26, no. 10, 2008, pp. 1204-1216.
8. S.M. Sait, and H. Youssef, *VLSI physical design automation: theory and practice*, World Scientific Publishing Company; 1st edition, 1999, pp. 327-375.
9. C.A. Schemes, “Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey,” *IEEE Personal Communications*, 1996, pp. 11-31.
10. N.A. Sherwani, *Algorithms for VLSI physical design automation*, Kluwer Academic Publishers, Norwell, MA, USA, 1995.
11. A. Somogyi, “Equivalence of the left-edge and column by column channel routing algorithms, ” *Electronics Letters*, vol. 20, no. 14, 1984, pp. 606-607.
12. C. Wang, H.H. Ghenniwa, and W. Shen, “A winner determination algorithm for auction based decentralized scheduling,” *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* vol. 39*,* no. 5. 2009, pp. 609-618.
13. T. Yoshimura, and E.S. Kuh, “Efficient algorithms for channel routing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, no. 1, 1982, pp. 25-35.