# 國立交通大學

## 運輸科技與管理學系

## 碩 士 論 文

具差異性替選道路演算法之研究

An Algorithm For Finding Dissimilar Alternate Paths

研 究 生：陳冠佑

指導教授：王晉元　副教授

中 華 民 國 九十八 年 七 月

# 具差異性替選道路演算法之研究

# An Algorithm For Finding Dissimilar Alternate Paths


研 究 生：陳冠佑　　Student：Kuan-Yu Chen

指導教授：王晉元　　Advisor：Jin-Yuan Wang


國 立 交 通 大 學

運 輸 科 技 與 管 理 學 系

碩 士 論 文

A Thesis

Submitted to Department of Transportation Technology and Management

College of Management

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Transportation Technology and Management

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

# 具差異性替選道路演算法之研究

研究生：陳冠佑　　指導教授：王晉元 副教授

國立交通大學運輸科技與管理學系（研究所）碩士班

## 摘　　要

替代道路的資訊對於用路人而言是相當重要的參考依據。良好的替代道路可以讓用路人有效地避開擁塞的路段，縮短旅次時間。對於社會而言，能提高公路容量利用率，降低整體社會成本。然而若提供的替代道路數量過少，在車流量極大的狀況下，這些替代道路仍舊可能無法應付這龐大的車流，造成擁塞。因此若能提供足夠數量的替代道路，將能更有效的進行車輛分流，提昇道路的容量利用率。在過去的應用中，通常以 k 條最短路徑(k shortest path problem, ksp)為求解多條替代道路演算法。此演算法雖可計算出成本差異最小的替代道路，但對於用路人而言，過多的重疊會造成此替代道路無替代效果。本研究認為好的替代道路應在經過路段上有一定的差異度，而總成本的差異也必須在可接受範圍內，如此的替代道路方有分散並吸引車流的效果。因此本研究以 k 條最短路徑演算法做為基礎，加入重覆避免與過濾機制，可在有限的時間內透過預先刪除易重複路段，以及加入最大重疊度判斷的權重影響，求得滿足需求數量且具有差異性的最短路徑，同時這些路徑間的成本差距也在一定的範圍之內，足以做為替代道路資訊發佈的求解演算法。

**關鍵字**：替代道路、最短路徑、k 條最短路徑演算法

# An Algorithm For Finding Dissimilar Alternate Paths

Student: Kuan-Yu Chen Advisor: Dr. Jin-Yuan Wang

Department of Transportation Technology and Management
National Chiao Tung University

Abstract

Alternate paths information is very important for road users. Well design alternate paths can help road users avoid crowded roads to shorten their travel time and improve transportation efficiency.

*K shortest paths(KSP) algorithm* is normally used to generate alternate paths. However, these k shortest paths are very similar in term of their geomantic shapes. In many applications, especially transportation routing planning, this similarity characteristic cause serious implementation difficulties for practical application.

This paper introduced an effective algorithm to generate alternative paths. We considered the “good” alternative paths must have enough variance in passed arcs and little different in travel distance. So we defined two indicators, overlapping and detour, to identify the quality of alternative paths. We build an algorithm based on Yen’s ksp algorithm and pre-delete some arcs that may cause high overlap and put overlapping distance into consideration before choose each shortest path.

In order to evaluate this algorithm, we compare it with IPM and CKSP in several scenarios and network types. The result is this algorithm may use more computing time, but it takes great advantage in paths quality and solving ability. The alternative paths that generated from my algorithm are requested to be geographically different and also requested to have little different in travel distance.

This algorithm is useful in real-time path guiding system. It can provide road users several paths to avoid jammed arcs and disperse traffic flow. In the other hand, it can use to solve hazardous material transportation problem, decrease the risk of effected area.

**Keywords**: Alternate Paths, Shortest Path, K Shortest Paths

# 誌謝

這篇論文的完成，首要感謝我的指導老師 王晉元老師。從大學開始跟隨老師做計劃案、研討會、畢業專題、學佛營、以及最後的碩士論文，老師的指導讓學生銘記在心。另外感謝蘇昭銘、李克聰老師在百忙之中撥空參與論文口試，並提供寶貴的修改意見。

在運管系的日子一下子就過去了，感謝在這麼多年中教授我知識的各位老師，還有系辦兩位助理為學生各項活動所提供的協助。另外也感謝實驗室裡的小松、彥佑、思文等多位學長不管是課業上或是生活上的諸多幫忙。同時也感謝各位親愛的同學們，大萩、帥總、友維，綠茵等等，有你們的參與讓這研究生涯不再那麼苦悶。

最後要感謝我的家人，爸爸、媽媽、姐姐、外婆以及維真，你們的陪伴是支持我成長的最大動力。

陳冠佑 謹致

2009 年七月盛夏 於新竹

# 目　錄

# 圖目錄



# 表目錄

# 符號說明

$n$ ：網路中節點數量

$m$ ：網路中節線數量

$d_{uv}$ ：任兩節點 $u, v$ 間之成本

$P(s, t)$ ：任兩節點 $s, t$ 間之最短路徑

$l(s,t)$ ：$P(s, t)$的長度

$K$ ：總需求路徑數

$A^k$ ：第 $k$ 條從起點到終點的最短路徑

$Q_k$ ：第 $k$ 條最短路徑上，從起點(1)到終點($N$)前共經過幾個點

$Q_k^k$ ：終點($N$)的前一點

$A_i^k$ ：從 $A_i^{k-1}$ 路徑上第 $i$ 個點所分岔產生的路徑

$R_i^k$ ：$A^k$ 中從節點(1)到節點($i^k$)的子路徑

$S_i^k$ ：$A^k$ 中從節點($i^k$)到節點($N$)的子路徑

$C_i^k$ ： $A_i^k$ 的成本

# 一、緒論

## 1.1 研究背景與動機

替代道路的資訊對於用路人而言是相當重要的參考依據。良好的替代道路可以讓用路人有效地避開擁塞的路段，縮短旅次時間。對於社會而言，能提高公路容量利用率，降低整體社會成本。目前台灣西部地區雖已建立完善的替代道路路網，包括國道一號、國道三號、以及多條東西向的快速道路皆已陸續通車，但每逢連續假期，流量仍舊停留在幾條主要道路上，未能平均分散至整個路網，達到公路容量最大利用。若能提供良好的替代道路路徑規劃，將能有效舒解瓶頸流量，降低整體旅行時間與成本。

然而若是提供的替代道路數量過少，在車流量極大的狀況下，這些替代道路仍舊可能無法應付這龐大的車流，造成擁塞。因此若能提供足夠數量的替代道路，將能更有效的進行車輛分流，提昇道路的容量利用率。

在過去的應用中，通常以 k 條最短路徑(k shortest path problem, ksp)為求解多條替代道路演算法。k 條最短路徑演算法可將路網中連接該起迄點之所有路徑，依成本大小完整排序，再從中選擇所需之前幾條。但這個演算法常常會遇到路段重疊度過大的問題，以台灣地區基隆到新竹的路徑規劃為例，圖 1.1.a 為台灣地區國道一號與國道三號於基隆至新竹間的國道路網，圖 1.1.b 為基隆至新竹的最短路徑，該路徑沿國道一號，從基隆交流道到新竹交流道。但透過傳統 k 條最短路徑所得的第二條路徑會是從「基隆交流道沿國道一號走到湖口休息區，下去繞一圈再回到國道一號，往前到新竹交流道」，如圖 1.1.c 所示。這條路徑確實是「次短」路徑，但卻不是一條理想的替代道路。對於實際應用而言，所需要的應是具有足夠差異的替代道路，而非大部份路段重疊的路線。根據國道高速公路工程局資料指出，國道一號常見壅塞發生於圓山交流道至泰山收費站、林口至中壢路段，上述之「次短路徑」由於大量的路段重疊造成重複使用這些擁塞路段，無法達到替代道路分散車流、縮短旅行時間的效果。

理想的替代道路應如圖 1.1.d 所示，「從基隆交流道上國道一號，到汐止系統接國道三號，一路到新竹系統接回國道一號到新竹交流道」，這條路徑與最短路徑雖有些許的距離差距，但在經過路段的選擇上卻具有足夠的差異，對於使用者而言，該條替代道路將能有效避開易雍塞路段，為可供參考的替代道路。

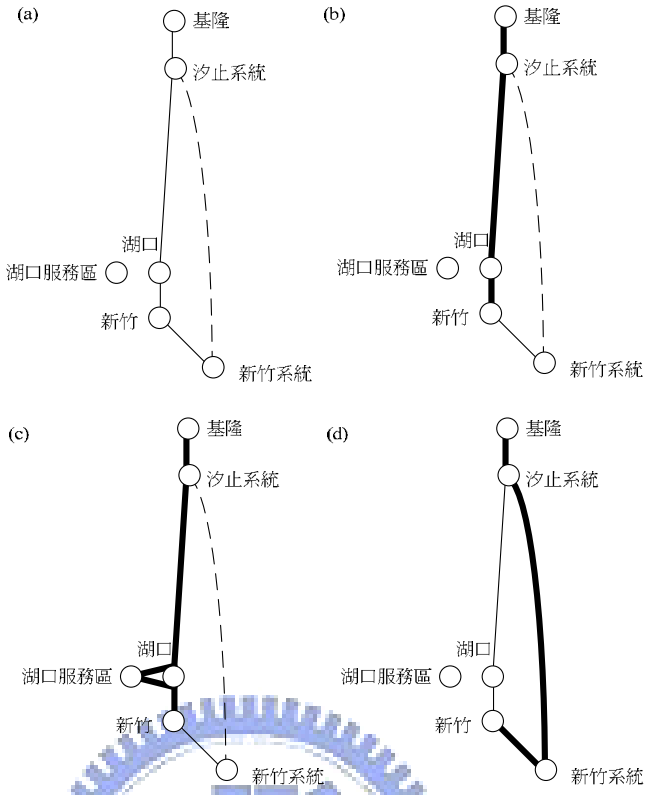



圖 1.1 基隆至新竹替代道路規劃示意圖

由上述說明可知，傳統 k 條最短路徑演算法雖可計算出成本差異最小的替代道路，但對於用路人而言，過多的重疊會造成此替代道路無替代效果。本研究認為好的替代道路應在經過路段上有一定的差異度，而總成本的差異也必須在可接受範圍內，如此的替代道路方有分散並吸引車流的效果。因此本研究希望能夠提供一套演算法，針對一對一的路徑規劃，提供足夠數量的「好的」替代道路，期能有效解決傳統 k 條最短路徑演算法無法提供具實用性替代路徑的問題。對於用路人而言，將擁有更多路徑方案可以選擇，從中選出最符合自己需求的道路，進而達到均勻使用路網資源，降低壅塞的目標。

## 1.2 研究範圍與目的

本研究的主要目的為在具有方向性的網路中，提出一套替代道路的路徑演算法，針對一對一的路徑規劃，提供足夠數量的替選方案，並在經過路段上有一定的差異度，同時其成本差距也在可接受的範圍內，此解具有低成本差異與低路段共用的特性，將能有效的提供用路人決策輔助。而由於本研究的演算法將以適用於交通運輸路網為目標，因此產生的路徑中不可有迴圈。

為降低演算的複雜度，本研究在考慮路段重疊成本時，並未考慮各路段屬性的差異，路段的重疊成本僅與該路段的長度有關。同時演算過程中，並未考慮路口的轉向限

制、單行道限制、或是車輛高度等等的路段屬性限制。

本研究提出之演算法將以交通部運輸研究所發行的新世紀台灣地區交通路網數值地圖 1.0 版做為實證對象，並與傳統的 k 條最短路徑演算法進行比較與數值分析，進行相關的評估，並對實際應用提出建議。

## 1.3 研究內容與步驟

本研究的主要內容如下：

1. 現有 k 條最短路徑演算法的回顧

   將各個 k 條最短路徑演算法做一回顧，分析其演算法的基本原理與優缺點，藉以做為新演算法的參考。

2. 新演算法的演算流程設計

   發展新演算法的演算流程，能有效的解決前述議題。

3. 與傳統演算法的數值分析比較

   針對先前回顧的 k 條最短路徑演算法，與新演算法進行數值分析比較。並輔以路網範例進行評比。

根據上述的研究內容，可定出研究步驟如圖 1.2 所示：

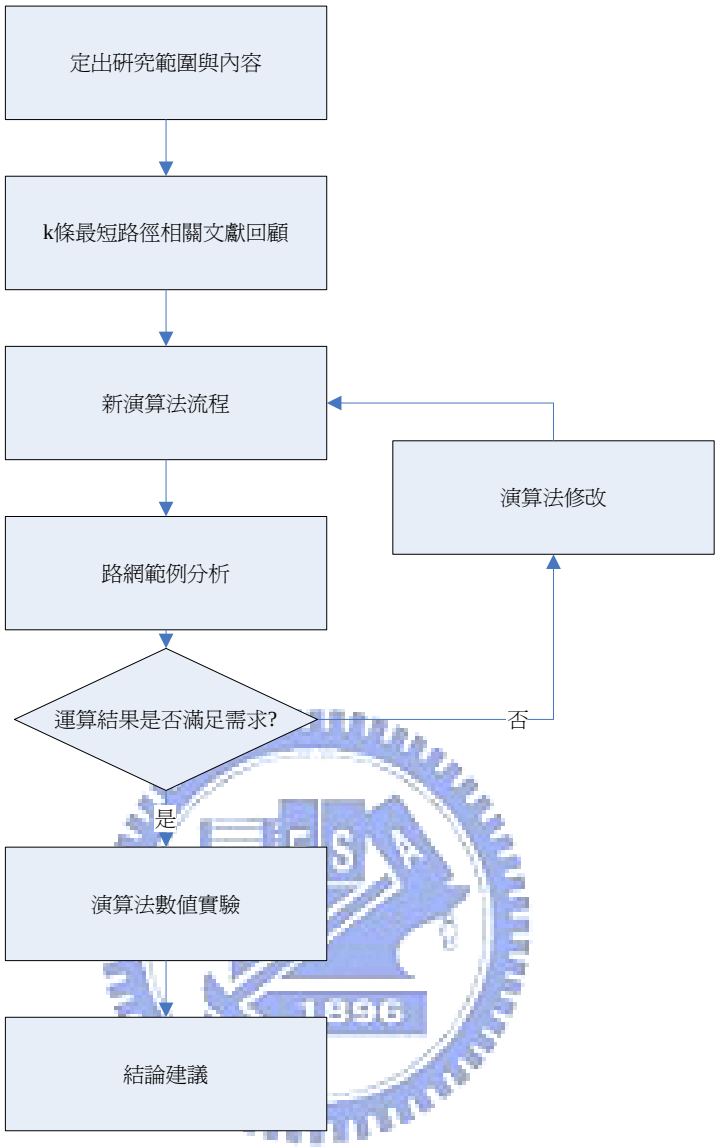
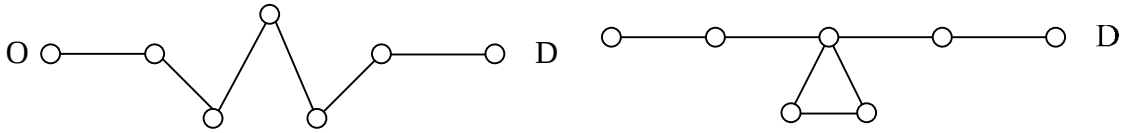

定出研究範圍與內容
k條最短路徑相關文獻回顧
新演算法流程
演算法修改
路網範例分析
運算結果是否滿足需求?
否
是
演算法數值實驗
結論建議

圖 1.2 研究流程圖

# 二、文獻回顧

## 2.1 k 條最短路徑相關文獻回顧

k 條最短路徑問題在條件的限制上可分為兩類：(1)不允許迴圈(Simple Path) 與(2)允許迴圈(Looping Path)。如圖 2.1 所示：




(a) 不允許迴圈路徑　　(b) 允許迴圈路徑

圖 2.1 允許迴圈與不允許迴圈路徑

不允許迴圈路徑表示該路徑上，同一個節點只能被訪問一次，以 Yen(1971)的解法為代表，之後的 Lawler(1976) , Katoh(1982), Hadjiconstantinou and Christofides (1999), Pascoal (2005) 皆沿續其想法進行研究。允許迴圈路徑則是允許重複訪問同一個點。Martins (1984), Azevedo(1993), Eppstein(1998)的研究皆屬於這個範圍。無論是否允許迴圈產生，Azevedo (1993)將 k 條最短路徑的相關研究，根據其演算核心分為三類。(1)最佳原則產生法(Principle of Optimality Method)、(2)循序產生法(Iterative Method)、(3)網路擴大法(Network Enlargement Method)，以下分別加以介紹：

### 2.1.1 最佳原則產生法

由 Yen(1971)提出，核心觀念為第 *k* 條路徑必由第 *j* 條路徑分叉出來( *j* = *1..k-1* )。其演算流程概述如下：首先產生第一條最短路徑，沿此路徑上每一個點都分別產生一條新的最短路徑連到終點，與分岔點以前的路徑相連結即可獲得一條新的路徑，將這所有的路徑放入候選區，選出其中最小成本的即為次短路徑。再仿照此方法將第二條最短路徑的所有節點再分叉出一條路徑，與第一條最短路徑所分叉出的路徑混合比較，取出其中最短的做為第三條路徑。如此執行 *k* 次即可得到 *k* 條最短路徑。

Lawler(1976)的作法與 Yen 類似，他將連接該起迄點的所有路徑視為一個集合，第一次運算時利用最短路徑將整個最短路徑集合分割成數個子集合，切割方式乃是以該路徑上的每一個點做為切割點，所有由該路徑所分岔產生的路徑，皆視為同一個子集合，這些子集合內最短的路徑即為次短路徑。下一次再利用該次短路徑將產出該路徑的子集合再分割為多個子集合，此時除了已選出的路徑外，所有的子

集合內的最短路徑，即為第三條最短路徑。其演算法示意圖如圖 2.2：

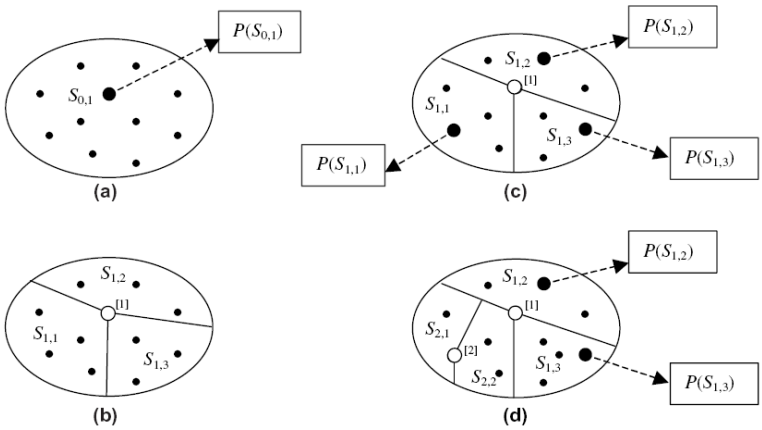



圖 2.2 Lawler 演算法示意圖

圖 2.1.a 中，$S_{0,1}$ 是所有的路徑，$P(S_{0,1})$則是整個集合內的最短路徑，假設該路徑上共有四個點，由於終點無法產生分岔路徑，因此 $P(S_{0,1})$將整個集合切成 $S_{1,1}, S_{1,2}, S_{1,3}$ 三個集合，如圖 2.1.b 所示，切割方式與 Yen 的做法相同，從 $P(S_{0,1})$上第一個點分叉得來的所有路徑，屬於 $S_{1,1}$，第二個點分叉得來的所有路徑，屬於 $S_{1,2}$，第三個點分叉得來的所有路徑，屬於 $S_{1,3}$。這三個集合內的最短路徑分別為 $P(S_{1,1})$, $P(S_{1,2})$, $P(S_{1,3})$，而其中最短的路徑就是第二條最短路徑。此分法與 Yen 的方法類似，只是解釋的觀點不同。

Katoh(1982)針對 Yen 的演算法進行改進，在 Yen 的方法中，每一次的運算 $k$ 產生的候選路徑數量，等於第 $k$-1 運算所得的路徑的總節點數。因此 Katoh 演算法的核心概念在於減少可行解的量。於每次的運算中將該次應產生的候選路徑利用一個虛擬點 $V_{d(Pk)}$分為三個集合，在 $V_{d(Pk)}$點前所分叉的路徑集合、在 $V_{d(Pk)}$點分叉的路徑、以及在 $V_{d(Pk)}$點後分叉的路徑集合，從中選擇最成本最小的做為候選路徑。透過這個想法，理論上每次運算只會產生三條候選路徑，但實際上每一個集合內的路徑選擇也同樣必須花費大量運算時間，只是節省了理論上的計算複雜度。

### 2.1.2 循序產生法

Shier(1976)提出不少類似的矩陣運算方法求解一對多的 k 條最短路徑，在此僅就常被引用的雙向掃描法(Double-Sweep Method)做說明。此法先產生所需要的 k 條路徑做為起始值的向量元素個數，再以特別定義的矩陣運算求解，如此可得所需的 k 條最短路徑。其演算步驟如下：

1. 首先定義一個距離矩陣 $D$，此矩陣中的每一個值 $D_{ij}$ 表示節點 $i$ 到節點 $j$ 的最短距離。定義兩個矩陣 $L$ 與 $U$，這兩個矩陣將會在每次的運算(sweeps)中被使用。

   $D=[\ D_{ij}\ ]$

$L=[\ L_{ij}\ ]$, *where* $L_{ij} = D_{ij}$ *for* $i>j$; $L_{ij} = \infty$ *for* $i \leqq j$

$U=[\ U_{ij}\ ]$, *where* $U_{ij} = D_{ij}$ *for* $i<j$; $U_{ij} = \infty$ *for* $i \geqq j$

2. 建立一個向量集合 *S(K)*，其中每一個元素皆為一個 K 維度的向量，用以儲存求解結果。

3. 在第一次的運算中，產生 $E(0) = E[E_{01}, E_{02}, E_{03}, \ldots, E_{0n}] \in S(K)$ *where* $E_{01}, E_{02}, E_{03}, \ldots, E_{0n} = 0$，為起點到各點的期望長度

4. 在第 *k* 次的運算中，產生 $E(k) = E[E_{k1}, E_{k2}, E_{k3}, \ldots, E_{kn}]$，而其中各值將由以下的遞迴關係式所產生，其中 $r = 0,1,2,\ldots$：

   $E(2r + 1) = E(2r) \oplus E(2r+1) \otimes L$ Backward Sweep

   $E(2r + 2) = E(2r+1) \oplus E(2r + 2) \otimes U$ Forward Sweep

   其中⊕、⊗為特別定義的矩陣運算子，⊕表示於兩矩陣元素合併取前 K 小元素，例如 *k*=3，[1,4,8]⊕[2,3,6] = [1,2,3]。⊗表示於兩矩陣的加總取最前 K 小值，例如 *k*=3，[1,4,8]⊗[2,3,6] 為在 [1+2,1+3,1+6,4+2,4+3,4+6,8+1,8+3,8+6]中取出前 3 小的元素，即為 [3,4,6]。另外在 Backward Sweep 中，計算順序是由 $E_{kn}, E_{kn-1}, E_{kn-2}, \ldots, E_{k1}$，但在 Forward Sweep 中，計算順序為 $E_{k1}, E_{k2}, E_{k3}, \ldots, E_{kn}$。

5. 當 *E(2r + 1)*的值等於 *E(2r + 2)*的值，表示找到答案，但此時矩陣中所存的只是各條路徑的成本，需再透過一個回溯過程取得所有路徑。

### 2.1.3 網路擴大法

Martin(1984)於提出「刪除法」(Deletion Algorithm)，用以在有向圖中求解允許迴圈的 k 條最短路徑問題，其核心概念是在已有的最短路徑上刪除某條關鍵路段，並重新求得替代的路段，藉以產生新的路徑。但演算法中必須透過在路網上加入虛擬的節點與節線來實現，因此通常稱為網路擴大法。以圖 2.3 為例，其演算步驟如下：

1. 當 *k* 等於 1 時，求解最短路徑 *p*。在圖 2.2.a 的範例中， *p*=(1,2,3,4,7)

2. 當 *k* 等於 2 時，從 *k-1* 最短路徑中的所有節點，選出第一個於路網中入度(in degree)大於 1，且尚未於路網中加入虛擬對應點的節點，設定為 *n*。(設定 *n*=3)

3. 增加一個節點 *n'*到路網中，並將所有 *n* 在路網中的前置點連結到 *n'*，但扣除 *n* 在 *p* 中的前置點。(加入 *n'*=3'，並連接 5 到 3')

4. 將 *p* 中 *n* 之後的所有點 $n_i$，加入一虛擬對應點 $n_i$'於路網中，並仿照步驟 3

中，將所有 $n_i$ 在路網中的前置點連接至 $n_i$'。此時若 $n_i$ 的前一點 $n_{(i-1)}$同樣擁有虛擬對應點時，也需連結 $n_{(i-1)}$'至 $n_i$'。(加入 4'、7'，並連接 3'至 4'、6 至 4'、4'至 7')

5. 求解起點到終點虛擬對應點的最短路徑，即為第 2 條最短路徑。(求解 1 到 7'的最短路徑，可得 1,2,3,6,4',7'，第二條最短路徑即為 1,2,3,6,4,7，如圖 2.2.b)

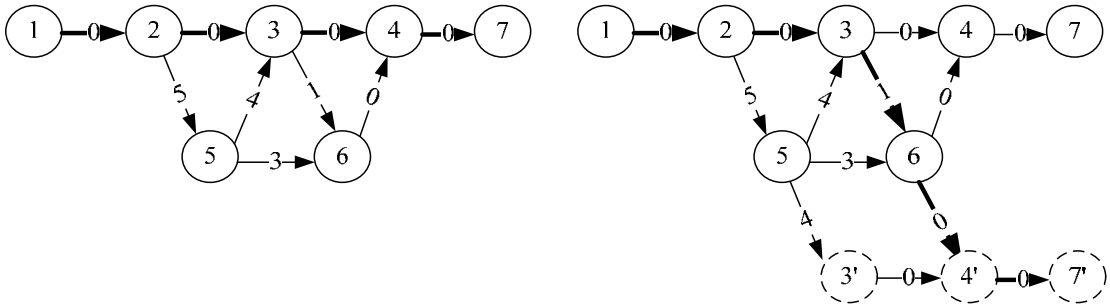



圖 2.3 網路擴大法示意圖 (a) 原始的網路圖(b)下一迴圈的網路圖

於先前回顧的文獻中，主要都是集中在 k 條最短路徑演算法的探討上，重點在於減少運算時間、記憶體空間等。但由於這些演算法的運算時間皆為多項式時間，即使網路擴大或是需要求解的 k 值數量增加，皆可在合理時間內求得結果。就實務上而言，隨著硬體等級、運算能力的提昇，上述演算法的運算時間差距並不顯著。

然而在實務應用上，由於上述的演算法得到的結果皆為 k 條最短路徑的最佳解，也就是所有路徑中，路徑長度為前 k 短的路徑。若要以提供替代道路為目的，將遭遇替代道路的重疊度過大，易造成用路人參考時的困擾。因此在下一節中將就 k 條差異路徑的相關文獻進行回顧。

## 2.2 k 條差異路徑相關文獻回顧

### 2.2.1 懲罰法(Iterative Penalty Method, IPM)

此方法為最直覺的方法，透過執行 k 次最短路徑，在每次的運算後將已走過的路段加上一個懲罰值(Penalty)，藉由此懲罰值的作用，即可在下次的運算中盡量避開已走過的路段，進而得到 k 條不同的路徑。此法最早由 Johnson (1992)提出，用於求解危險物品運送的路徑。懲罰法最大的優點是運算速度快，僅需執行 k 次最短路徑演算法即可得到所需的 k 條最短路徑，而路線的重複程度也透過懲罰值的增加而降低。但其缺點就是距離成本差異難以掌握。

### 2.2.2 最小最大法(Minimax Method)

這方法由 Kuby(1997)提出，首先運用 k 條最短路徑演算法產生足夠數量的候選路徑，再透過簡單的整數規劃，以最大化差異度(最小化重疊度)、最小化總成本為目標從中選擇需要的路徑。

此方法若起始的 k 值夠大，將可求得路徑差異度、成本差異都不錯的解，但最大的缺點就是運算時間過長(Akgun, 2005)，因此方法在應用上，最重要的課題就是決定起始解的數量。一個好的起始解將能於可接受的時間內，求得所需的替代道路。

### 2.2.3 限制 k 條最短路徑(Contrained k shortest path, CKSP)

此方法由 Zijpp(2005)提出，以 Lawler 的演算法為基礎，加入繞道(detour)與重疊(Overlapping)的判斷。其核心觀念相當簡單，基於 Lawler 與 Yen 的「最佳原則產生法」，每一條新的路徑都是由前一條分岔出來的。因此該方法在將路徑分岔之時即先行檢驗，判斷此路徑是否滿足重疊度與成本差異的條件，若未滿足限制，即直接刪除，不再從此路徑進行候選路徑的產生。

相較於最小最大法，此方法大量降低了不可行解的產生，但由於其限制條件的檢查只與「最短路徑」相比較，產生的解中可能會發生除了最短路徑以外，其他的路徑彼此之間的重疊度都很大的問題發生。

根據上述的文獻回顧，過去的研究主要是求解 k 條最短路徑的問題，但是對於實務上的需求，仍有改善的空間。傳統的 k 條最短路徑演算法，求得的路徑重疊度過高，無法做為用路人參考；在 k 條差異路徑部分，懲罰法雖簡單易用，但其成本差異卻難以掌控，若替代道路成本差異過大，用路人將不會採用。最小最大法在足夠的起始解支持下，可以提供不錯的解，但卻會耗費龐大的運算時間，難以進行即時路徑規劃。限制 k 條最短路徑能有效的降低不可行解的產生，但某些較好的解卻可能被提早刪除，降低了解的

品質。

本研究將以 Yen 之演算法為基礎，發展相關的解法，融合限制 k 條最短路徑的門檻概念，於 Yen 的演算法中預先刪除可以造成大量重疊的路段，並於計算成本時加入重疊成本的權重，並在路徑未產生時即刪除未滿足重疊與成本差異的路徑，減少不可行解的產生。雖然失去 k 條最短路徑演算法的最佳性，卻能在有限時間內，獲得重疊度與成本差異皆低的可行解，以供用路人在路徑選擇時參考。

# 三、具差異性的 k 條最短路徑演算法

本研究提出一個可求解具差異性的 k 條最短路徑演算法，以 Yen 的演算法為基礎，融合限制 $k$ 條最短路徑的門檻概念，於 Yen 的演算法中預先刪除可以造成大量重疊的路段，並於計算成本時加入重疊成本的權重。雖然會失去 $k$ 條最短路徑演算法的最佳性，卻能在有限時間內，獲得重疊度與成本差異皆低的可行解，在下面的各節中，將對新的演算法進行詳細的描述，並提出本研究的演算法說明與演算範例。

## 3.1 符號說明

本研究定義符號如下，$G = (V, A)$為一網路圖形，其中 $V$ 為 $n$ 個節點的集合，$A$ 為 $m$ 條節線的集合。$d_{uv}$ 表示任兩節點 $u, v$ 間之成本，$u, v \in V$。$P(s, t)$為任兩節點 $s, t$ 間之最短路徑，$l(s,t)$為 $P(s, t)$的長度 $s, t \in V$，起點為 1，終點為 $N$。

$K$ 為總需求路徑數，$A^k = (1) \text{-}> (2^k) \text{-}> (3^k) \text{-}> \ldots \text{-}> (Q_k^k) \text{-}> N$ 為第 $k$ 條從起點到終點的最短路徑，$k=1..K$。$(1), (2^k), (3^k), \ldots, (Q_k^k)$則是第 $k$ 條最短路徑上的第 $1,2 \ldots Q_k$ 個節點。$Q_k$ 表示在第 $k$ 條最短路徑上，從起點(1)到終點$(N)$前共經過幾個點，$Q_k^k$ 即表示終點$(N)$的前一點。(例：$A^k = 1\text{-}>4\text{-}>3\text{-}>7\text{-}>N$, $Q_k = 4$, $Q_k^k = 7$)。$C^k$ 為第 $k$ 條路徑的成本。

$A_i^k$, $i=1,2 \ldots Q_k$ 表示從 $A_i^{k-1}$ 路徑上第 $i$ 個點所分岔產生的路徑，$R_i^k$ 為 $A^k$ 中從節點$(1)$到節點$(i^k)$的子路徑，稱為節點$(i^k)$的前置路徑。$S_i^k$ 為 $A^k$ 中從節點$(i^k)$到節點$(N)$的子路徑，稱為點$(i^k)$的後置路徑。$C_i^k$ 為該條路徑 $A_i^k$ 的成本。如圖 3.1 所示：

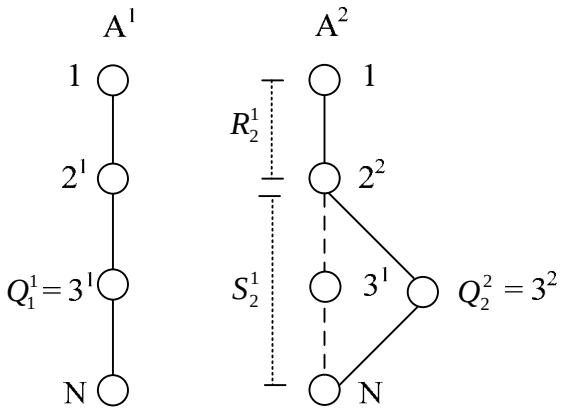


圖 3.1 演算法符號說明

在演算法中，本研究為避免路段大量重複，擴大分岔點後所刪除的路段，利用 $\alpha$ 、$\beta$ 防止路徑重複參數控制其範圍。最後在評選路徑時，透過 $\omega$ 為重複成本權重將路徑重疊度納入考慮，藉以產生具差異性的 k 條最短路徑。

## 3.2 路段重疊度定義

為求得具差異性的替代道路方案，本研究定義路段重疊度 *S(x,y)* 為路徑 y 對路徑 x 的重疊度，其計算方式以 y 路徑成本為分母，分子為所有 x, y 路徑上共用路段成本的總合，以圖 3.2 為例，y 路徑對於 x 路徑的重疊度為 $\frac{x,y\text{重疊路段成本}}{y\text{路徑成本}} = \frac{2}{4} = 0.5$

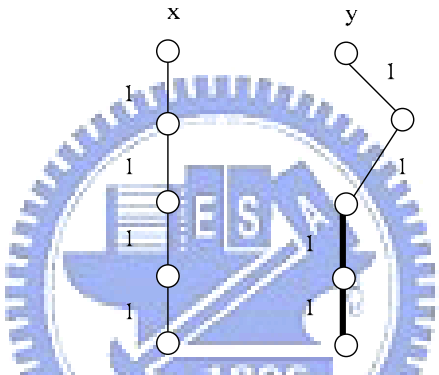


圖 3.2 重疊度計算示意圖

每一條候選路徑的重疊成本定義為 $O_i^k$ ，$O_i^k = MAX( S( A^j, A_i^k )) , j = 1..k-1$，表示該條路徑與所有已產生的最短路徑中的最大重疊度。如圖 3.3 所示，若目前已產生 3 條最短路徑 $A^1 \sim A^3$，目前產生的候選路徑為 $A_i^k$ ，則 $A_i^k$ 與目前產生的所有最短路徑的最大重疊度即為 $O_i^k = MAX( 0.5, 0.25, 0.25) = 0.5$。

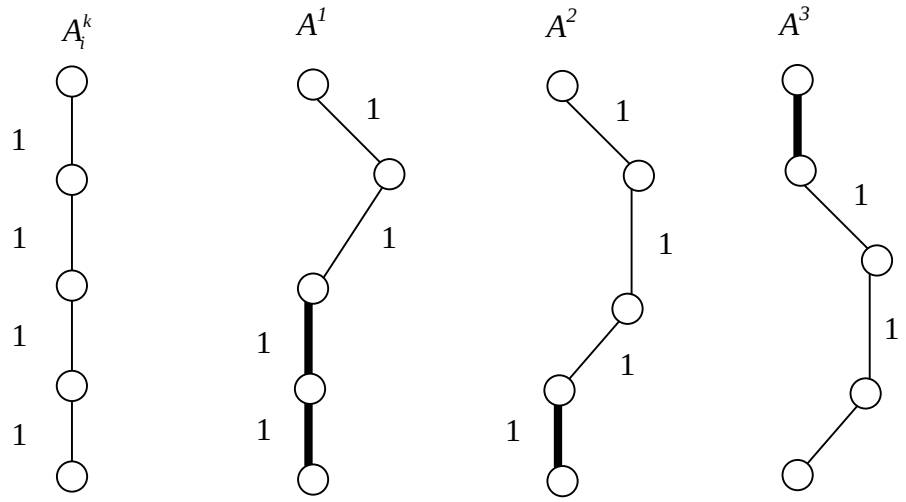



圖 3.3 各候選路徑最大重疊度計算示意圖

## 3.3 限制條件定義

為使本研究的求解路徑更符合需求，同時也可降低運算時間，本研究定義兩個限制條件：

1. 最大重疊度限制 $\Delta^{max}$，表示輸出的 k 條最短路徑與最短路徑的重疊度不可大於此限制。
2. 最大繞道度限制 $\Phi^{max}$，表示輸出的 k 條最短路徑其距離成本不可超過最短路徑的 $\Phi^{max}$ 倍。

## 3.4 演算步驟

本研究所提出的演算法，主要以 Yen 的演算法為基礎，加入防止路線重覆的機制，並保持成本的差異度在一定的範圍內。在 Yen 的演算法中，為求出不同的路徑，會把每一個分岔點的下一路段刪除，本研究為避免路段大量重複，擴大分岔點後所刪除的路段，並利用 $\alpha$、$\beta$ 參數控制其範圍，最後在評選路徑時，透過 $\omega$ 參數將路徑重疊度納入考慮，藉以產生具差異性的 k 條最短路徑。詳細演算步驟如下：

Iteration 1: 產生最短路徑 $A^1$

Iteration $k$, $k=2$ *to* $K$: For $i = 1, 2, 3 \ldots Q_k$

A. 分別檢查 $A^j$ 的$(1), (2^j), (3^j), \ldots, (i^j)$ $(j = 1..k-1)$是否與$(1), (2^{k-1}), (3^{k-1}), \ldots, (i^{k-1})$一致。若一致，將$(i^j)$到$(i+1^j)$的成本設為無限大。

本演算法為了產生具有差異的替代路徑，逐一與已產生的前 *k-1* 條最短路徑比對，若該點之前的所有路徑皆相同( $R_i^k = R_i^j$ , j = 1…*k-1*)，即刪除該路徑的下一個路段( $d_{(i^j),(i+1^j)} = \infty$ )，強迫產生不同的路徑。如圖 3.4 所示：

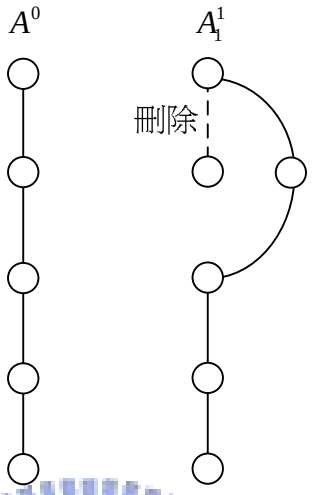
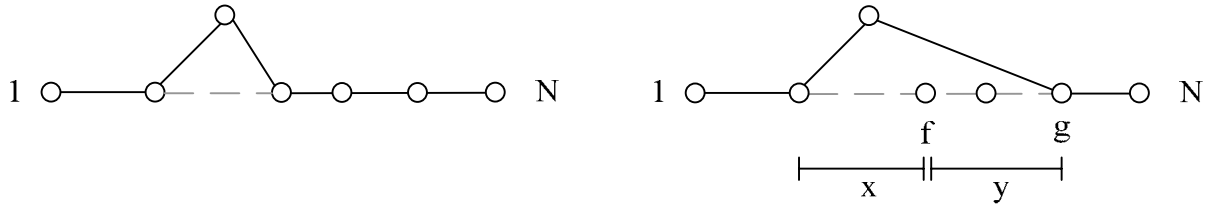



圖 3.4 刪除關鍵路段，產生差異路徑示意圖

B. 檢查 $d_{(1),(2^j)} + d_{(2^j),(3^j)} + \ldots d_{(l^j),(l+1)^j}$ 是否大於 $C^k * \alpha$ ，$o \in i \ldots Q_k$ ，若否，檢查 $d_{(1),(2^j)} + d_{(2^j),(3^j)} + \ldots d_{(l+1^j),(l+2)^j}$ 是否大於 $C^k * \alpha$，直到 *o* 等於某 $f \in i \ldots Q_k$，使得 $d_{(1),(2^j)} + d_{(2^j),(3^j)} + \ldots d_{(f^j),(f+1)^j} > C^k * \alpha$

C. 檢查 $d_{(1),(2^j)} + d_{(2^j),(3^j)} + \ldots d_{(l^j),(l+1)^j}$ 是否大於 $C^k * \beta$， $o \in f \ldots Q_k$，若否，將 $(f^j)$到$(f+1)^j$ 的成本設為無限大，直到 *o* 等於某 $g \in f \ldots Q_k$，使得 $d_{(1),(2^j)} + d_{(2^j),(3^j)} + \ldots d_{(g^j),(g+1)^j} > C^k * \beta$

為防止該路徑在經過些微的繞道即又轉回前一路徑，造成過大的重疊，本研究透過兩個參數 $\alpha$ 、 $\beta$ 控制，首先計算出兩點 f、g，其中 1~*f* 點的成本總和大於或等於該路徑成本的 $\alpha$ 倍，且 1~*g* 點的成本總和大於或等於該路徑成本的 $\beta$ 倍，此時將該路徑上 *f* 至 *g* 的路段刪除，於下一步計算新的最短路徑時，即可強迫新路徑形成較大差異，如圖 3.5 所示：




(a) 傳統演算法，僅刪除下一路段，路線很快就會回到同一路徑上

(b)新演算法，多刪除一些路段，路線較具差異性，x 為傳統演算法刪除的路段，y 為新演算法額外刪除的路段

圖 3.5 演算法比較圖

D. 計算節點($i^k$)到 $N$ 的最短路徑為 $S_i^k$，$R_i^k$ =(1) -> ($2^k$) -> ($3^k$) ->…-> ($i^k$)。將 $S_i^k$ 與 $R_i^k$ 相連接，可得 $A_i^k$ =(1)-> ($2^k$) -> ($3^k$) ->…-> ($i^k$) -> ($i+1^k$) ->…-> ($Q_k^k$) -> $N$。

利用新的路網成本計算分岔點($S^k$)到終點的最短路徑，與該分岔點的前置路徑相連接，即可得到一條替代路徑 $A_i^k$。

E. 計算 $O_i^k = MAX(S(A^j, A_i^k)), j = 1..k-1$。

將步驟 D 中所取得的路徑，與已產生的前 $k-1$ 條最短路徑相比較，計算重疊度，取其中最大值為此條路徑的重疊度。

F. 計算 $A_i^k$ 與最短路徑的成本差異與重疊度，若小於 $\Delta^{max}$ 與 $\Phi^{max}$，則將 $A_i^k$ 加入候選路徑集合 B 中，$i = i + 1$，回到步驟 A。

$A_i^k$ 為透過 $A^{k-1}$ 上的第 i 點所分岔出來的替代道路，若該路徑滿足限制，則將其加入候選路徑集合，前往下一點($i+1^k$)，繼續產生分岔的候選路徑，直到所有的分岔點皆已運算完畢($i = Q_k$)。

G. 從 $B$ 中的所有候選路徑中選出一條總成本最短的路徑 $t = A_x^y$ (總成本為 $C_i^k + \omega * O_i^k$)，若 $y < k$，重新計算其與 $A^j(j = y..k-1)$的重複度，並重新加入 $B$ 中，再重回(G)，若 $B$ 已空，表示無法找到足夠數量的路徑，停止運算。

H. 若 $y=k$， $A^k = t$。若 $k = K$，$A^1, A^2, \ldots, A^K$ 即為所求的 K 條具差異性替代道路。

$B$ 集合為所有候選路徑的集合，在第 $k$ 次運算結束之前，$B$ 集合中包含所有 $A_x^y$，$y = 1 \ldots k, x = 1 \ldots Q_y$，本研究於每次產生路徑時與先前已產生的前 $k$-1 條最短路徑比較重疊度，之後從所有的候選路徑集合中取出總成本(重疊度加權成本+路徑距離成本)最短的路徑，若此路徑是在第 $j$ 次($j<k$)所產生出來的路徑，表示它僅與 $A^1 \sim A^j$ 做過重疊判斷，因此將其重與 $A^j \sim A^{k-1}$ 進行重疊比對，再放入 $B$ 中，重新取出最高優先的路徑。如此即可得到 K 條具差異性的替代道路。

由於演算法是透過 Yen 的最佳性原則求解替代道路，雖已透過繞道限制控制路徑的成本差異，但無可避免的在某些路段可能會產生對實務上沒有意義的繞道。因此在路徑輸出發佈時，應再以人工進行截彎取直作業，藉以取得更適應用之路徑。

## 3.5 演算法範例

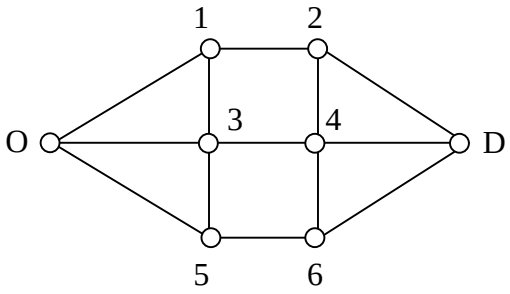



圖 3.6 範例路網圖

本段將以圖 3.6 之路網進行範例演練，此路網中所有路段成本皆為 1，並設定 $\alpha$=0.3、$\beta$=0.6、$\omega$=1、$K$=3、$\Delta^{max}$=0.5、$\Phi^{max}$=0.5

Iteration 1: 產生最短路徑 $A^1 = (O, 3, 4, D)$，成本為 3

Iteration 2: 針對 $A^1$ 上的每一個點檢查

1、 $i$=1，針對第一個點 $O$，檢查 $1^1$ 是否與$(1^1)$相同，由於此點與最短路徑的第一個點相同皆為 $O$，將下一路段$(O,3)$成本設定為無限大。(步驟 A)

2、 檢查是否需預先刪除下一路段，防止重複：$d_{O,3} / C^1 = 0.33$，大於 $\alpha$，$f = O$。(步驟 B)

3、 檢查下一個節點 4：$(d_{O,3} + d_{3,4}) / C^1 = 0.66$ ，已大於 $\beta$，$g = 4$，令 $O$ 到 4 之間的路段成本為無限大，共刪除$(O,3)$、$(3,4)$兩個路段。(步驟 C)

4、 計算從 $O$ 到 $D$ 的最短路徑為 $A_1^1$=$(O,1,2,D)$，成本為 3。(步驟 D)

5、 計算與 $A^0$ 的相似度為 $O_1^1 = MAX( S(A^0, A_1^1)) = 0$。(步驟 E)

6、 檢查此路徑與最短路徑的成本差距為 0，重疊度為 0，滿足限制，將此路徑加入候選路徑集合 $B$ 中，結束 $i$=1 的運算。(步驟 F)

7、 $i$=2，針對第二個點 3，仿照步驟(A)到步驟(E)，令路段(3,4)的成本為無限大，計算 3 到 $D$ 的最短路徑，加上$(O,3,4)$為 $A_2^1$=$(O,3,4,2,D)$，成本為 4，與 $A^1$ 的相似度為 $O_2^1$=1，與最短路徑的成本差距為 0.33，重疊度為 0.33，滿足限制加入候選路徑集合 $B$ 中。(步驟 A~F)

8、 $i$=3，針對第三個點 4，仿照步驟(A)到步驟(E)，令路段$(4,D)$的成本為無限大，計算 4 到 $D$ 的最短路徑，加上$(O,3)$為 $A_3^1$ = $(O,3,1,2,D)$，成本為 4，與 $A^1$ 的相似度為 $O_3^1$=2，與最短路徑的成本差距為 0.33，重疊度為 0.66，未

滿足限制，刪除此路徑。(步驟 A~F)

9、 此路徑上所有可分岔的點皆已完成，從 $B$ 中的所有候選路徑裡，選出路徑 $A_1^1 = (O,1,2,D)$，使得(成本+ω*相似度) = 3，與其他路徑相比最小，為第二條路徑：$A^2$。(步驟 G)

Iteration 3: 針對 $A^2$ 上的每一個點檢查

1、 針對第一個點 $O$，根據 $A^1$， 將(O,3)、(3,4)路段成本設為無限大；跟據 $A^2$，將(O,1)、(1,2)路段成本設為無限大，計算 $O$ 到 $D$ 的最短路徑為 $A_1^2$ =(O,5,6,D)，成本為 3，與 $A^1$、$A^2$ 的最大相似度為 $O_1^2$ =0，與最短路徑的成本差距為 0，重疊度為 0，滿足限制，加入候選路徑集合 $B$ 中。(步驟 A~F)

2、 針對第二個點 1，根據 $A^2$，將(1,2)路段，計算 1 到 $D$ 的最短路徑加上(O,1)為 $A_2^2$ =(O,1,3,4,D)，成本為 4，與 $A^1$、$A^2$ 的最大相似度為 $O_2^2$ =2，與最短路徑的成本差距為 0.33，重疊度為 0.66，未滿足限制，刪除路徑。(步驟 A~F)

3、 針對第三個點 2，根據 $A^2$，將(2,D)路段成本設為無限大，計算 2 到 $D$ 的最短路徑加上(O,1,2)為 $A_3^2$ =(O,1,2,4,D)，成本為 4，與 $A^1$、$A^2$ 的最大相似度為 $O_3^2$ =2，與最短路徑的成本差距為 0.33，重疊度為 0.66，未滿足限制，刪除路徑。(步驟 A~F)

4、 從 $B$ 中的所有候選路徑裡，選出路徑 $A_1^2 = (O,5,6,D)$，使得(路徑成本+ω*相似度) = 3，與其他路徑相比最小，為第三條路徑：$A^3$。(步驟 G)

5、 演算結束，取得三條路徑 $A^0$ =(O,3,4,D)、$A^1$ = (O,1,2,D)、$A^2$ = (O,5,6,D)，平均路徑長度為 3，平均最大重疊度為 0。(步驟 H)

# 四、數值實驗

本研究所提出的具差異性k條替代道路演算法已進行一系列的數值實驗證明其可用性，實驗內容包括：

1. 新演算法的參數敏感度分析
2. 新的演算法所花費的 CPU 時間與已知演算法相比較
3. 新的演算法與已知演算法產生的路徑品質相比較
4. 對於演算法進行繞道、重疊限制與網路規模的敏感度分析

實驗的對象包括限制 k 條最短路徑法以及懲罰法。實驗流程圖如圖 4.1：


圖 4.1 數值實驗流程圖

數值實驗中首先將進行新演算法的參數敏感度分析，藉以求得適用之參數以供後續實驗中使用。第二步驟中將以新竹市路網為例，比較各演算法在不同測試情境下的求解速度與品質表現。最後則將實驗拓展到不同的路網規模中，比較演算法的求解能力。

由於懲罰法無法對限制條件進行處理，因此在演算法的比較中，運用出局法則(Strike Out Rule, Zijpp, 2005)來執行懲罰法。出局法則乃是在懲罰法產生第 $r$ 條路徑時，檢查其是否滿足限制條件。若此路徑滿足限制則加入可行解集合，直到可行解的數量等於 k，或是運算時間超過預設的限制即停止演算。詳細演算流程如圖 4.2 所示：

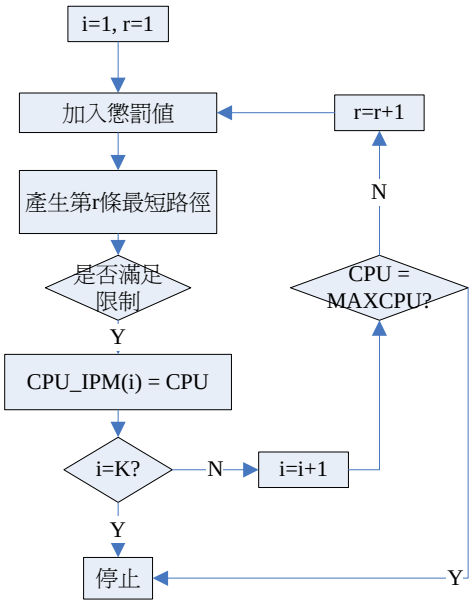



圖 4.2 出局法則執行流程圖

## 4.1 路徑品質比較指標

本研究的數值實驗相較於 Zijpp(2005)只比較了運算時間，更加入了求解品質的比較。為進行路徑品質的評比，本研究定義兩個品質指標：

Φ：平均路徑成本差異度，為第 2 到第 K 條路徑與最短路徑的成本差異比平均值。假設現有三條路徑，最短路徑成本為 10，第 2、第 3 條成本為 12、14。

$$\Phi = \left( \frac{12-10}{10} + \frac{14-10}{10} \right) \Big/ 2 = 30 \ (\%)$$

Δ：平均路徑最大重疊度，為第 2 到第 K 條路徑最大重疊度的平均。假設現有三條路徑，第 2 條路徑成本為 12，最大重疊度為 4、第 3 條成本為 15，最大重疊度為 5。

$$\Delta = \left( \frac{4}{12} + \frac{5}{15} \right) \Big/ 2 = 33 \ (\%)$$

## 4.2 防止路徑重複參數敏感度分析

本研究演算法中透過 $\alpha$、$\beta$ 防止路徑重複參數刪除路徑中段之可能重覆路段，為瞭解此參數對演算法之影響，本階段實驗調整參數範圍進行敏感度分析。本階段實驗採用新竹地區路網，隨機選擇十組起迄對，產生五條路徑。平均求解品質如表 4.1：

表 4.1 敏感度分析表

| $\alpha$ | 0.1 | | 0.2 | | 0.3 | | 0.4 | | 0.5 | | 0.6 | | 0.7 | | 0.8 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\beta$ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ |
| 0.2 | 12 | 80 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 0.3 | 33 | 40 | 4 | 27 | - | - | - | - | - | - | - | - | - | - | - | - |
| 0.4 | 4 | 28 | 4 | 27 | 21 | 25 | - | - | - | - | - | - | - | - | - | - |
| 0.5 | 4 | 27 | 4 | 26 | 21 | 25 | 4 | 22 | - | - | - | - | - | - | - | - |
| 0.6 | 6 | 23 | 6 | 22 | 20 | 24 | 4 | 22 | 4 | 25 | - | - | - | - | - | - |
| 0.7 | 12 | 15 | 9 | 16 | 10 | 14 | 4 | 22 | 8 | 21 | 3 | 30 | - | - | - | - |
| 0.8 | 12 | 13 | 10 | 15 | 10 | 14 | 4 | 22 | 4 | 23 | 3 | 24 | 3 | 30 | - | - |
| 0.9 | 12 | 13 | 5 | 18 | 5 | 18 | 4 | 21 | 4 | 22 | 4 | 21 | 4 | 21 | 3 | 28 |

表 4.1 中，橫軸、縱軸分別為防止路段重複參數範圍之起迄值。Φ、Δ欄位表示在此$\alpha$、$\beta$的範圍下，所求路徑的平均成本差異度與平均最大重疊度。由表中可知，$\alpha$、$\beta$間的差距越大，重疊度越低，但成本差距越高。反之$\alpha$、$\beta$間的差距越小，重疊度越高，成本差距越低。而以刪除路徑中間路段的參數表現最為平均，成本差距與路徑重疊度較為接近。因此本研究後續的數值實驗將以$\alpha$=0.3、$\beta$=0.7 進行。

## 4.3 測試情境定義

為了瞭解各演算法在不同的應用條件下的求解品質，本研究透過變動兩個限制條件定義出五個測試情境( Zijpp, 2005)，並設定參數$\Delta^{max}$=0.5，$\Phi^{max}$=0.5 為中心值，進行參數調整，如表 4.2。

表 4.2 測試情境參數設置

| **run** | $\Phi^{max}$ | $\Delta^{max}$ | **限制類型** |
|---|---|---|---|
| 1 | 0.5 (100%) | 0.5 (100%) | 預設情況 |
| 2 | 0.3 (80%) | 0.5 (100%) | 限制較嚴格，較多的路徑會被判定為過長 |
| 3 | 0.7 (120%) | 0.5 (100%) | 限制較鬆，較少的路徑會被判定為過長 |
| 4 | 0.5 (100%) | 0.3 (80%) | 限制較嚴格，較多的路徑會被判定為重疊 |
| 5 | 0.5 (100%) | 0.7 (120%) | 限制較鬆，較少的路徑會被判定為重疊 |

## 4.4 不同限制條件情境測試結果

本階段實驗以新竹市市區路網做為測試範例，分別對五個測試情境進行測試：

表 4.3 新竹市路網 CPU 運算時間結果(單位 0.001 秒)

| | 本研究演算法 | | | 限制 k 條最短路徑 | | | 懲罰法 | | |
|---|---|---|---|---|---|---|---|---|---|
| #\K | 5 | 10 | 15 | 5 | 10 | 15 | 5 | 10 | 15 |
| 1 | 2594 | 6484 | 14547 | 281 | 625 | 1015 | 47 | 94 | ∞ |
| 2 | 2594 | 6484 | 17593 | ∞ | ∞ | ∞ | 47 | 110 | ∞ |
| 3 | 2594 | 6484 | 14547 | 281 | 594 | 1094 | 47 | 94 | ∞ |
| 4 | 2594 | 6484 | 14547 | 281 | 703 | 984 | 47 | ∞ | ∞ |
| 5 | 2594 | 6484 | 14547 | 297 | 625 | 1000 | 47 | 94 | ∞ |

註：演算時間為∞表示無法求得所需數量的路徑

表 4.4 新竹市路網求解品質 (%)

| | 本研究演算法 | | | | | | 限制 K 條最短路徑 | | | | | | 懲罰法 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | 5 | | 10 | | 15 | | 5 | | 10 | | 15 | | 5 | | 10 | | 15 | |
| \ # | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ |
| 1 | 14 | 17 | 9 | 30 | 13 | 34 | 31 | 50 | 17 | 68 | 14 | 75 | 36 | 35 | 36 | 52 | ∞ | ∞ |
| 2 | 14 | 17 | 35 | 20 | 33 | 26 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 38 | 47 | 39 | 49 | ∞ | ∞ |
| 3 | 14 | 17 | 9 | 30 | 13 | 34 | 27 | 70 | 14 | 74 | 11 | 73 | 36 | 35 | 36 | 52 | ∞ | ∞ |
| 4 | 14 | 17 | 9 | 30 | 13 | 34 | 31 | 50 | 17 | 68 | 14 | 75 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| 5 | 14 | 17 | 9 | 30 | 13 | 34 | 31 | 50 | 17 | 68 | 14 | 75 | 36 | 35 | 36 | 52 | ∞ | ∞ |

註：求解品質為∞表示無法求得所需數量的路徑

表 4.3 為三種演算法在五種測試情境下的求解速度比較，以千分之一秒為單位，而演算時間為∞表示在達到停止條件時仍無法求得所需數量的路徑，根據此表可知，在可求解的情況下，懲罰法的求解速度最為快速，每次運算皆可在小於 1 秒內完成，次之為限制 k 條最短路徑，運算時間約在 1 秒，最後是本研究所提出的演算法，約在 10 秒完

成運算。但懲罰法在求解路徑增加，或是限制條件較為嚴苛之時將無法求解，而限制 k 條最短路徑也是相同的情形，在測試情境 2 下，路徑重疊限制較多，限制 k 條最短路徑無法求解。

表 4.4 為三種演算法在五種測試情境下的求解品質比較，比較的指標為平均成本差異度與平均最大重疊度。由表中可知，在可求解的情況下，本演算法之路徑品質大多優於限制 K 條最短路徑演算法、以及懲罰法。(兩項指標數值均較低。)

## 4.5 各型態路網測試結果

在 4.4 節實驗中可知，懲罰法在路徑成本限制較嚴時表現較差，限制 k 條最短路徑演算法則是在重疊限制較嚴時表現較差，因此本階段的實驗，將限制條件固定為 $\Delta^{max}=0.5$，$\Phi^{max}=0.5$，對三方演算法皆無特定之優劣的測試情境下，針對不同型態的路網進行實驗，藉以瞭解各演算法在路網規模改變時求解能力的變化。實驗的路網包括：

1. 大型都市路網，以台北市路網為測試對象
2. 中型都市路網，以新竹市路網為測試對象
3. 郊區路網，以新竹縣路網為測試對象
4. 國省道路網，僅擷取國道與省道進行運算

表 4.5 各型態路網測試結果

| # | 本研究演算法 | | | | | | 限制 K 條最短路徑 | | | | | | 懲罰法 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | 5 | | 10 | | 15 | | 5 | | 10 | | 15 | | 5 | | 10 | | 15 | |
| | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ | Φ | Δ |
| 台北 | 16 | 21 | 18 | 32 | 22 | 35 | 33 | 76 | 20 | 88 | 16 | 89 | 48 | 38 | (6) | (6) | (6) | (6) |
| 新竹 | 14 | 17 | 9 | 30 | 13 | 34 | 31 | 50 | 17 | 68 | 14 | 75 | 36 | 35 | 36 | 52 | (11) | (11) |
| 竹北 | 6 | 38 | 15 | 40 | 22 | 43 | 27 | 62 | 13 | 72 | 11 | 76 | 46 | 58 | (6) | (6) | (6) | (6) |
| 國道 | 6 | 28 | 9 | 36 | 11 | 42 | (3) | (3) | (3) | (3) | (3) | (3) | 36 | 42 | (8) | (8) | (8) | (8) |

註：(N) 表示僅能求得 N 條路徑

由表 4.5 中可見，懲罰法在節點數較大的台北市，以及節點數較少的郊區、國省道路網求解能力較差，限制 K 條最短路徑演算法則是在節點數最少的國省道路網上表現最

差，無法求得最低標準的 5 條最短路徑。由圖 4.3 至圖 4.5 中可明顯看出本研究演算法在四種路網下皆能求得答案，同時求解品質也優於限制 K 條最短路徑演算法與懲罰法。

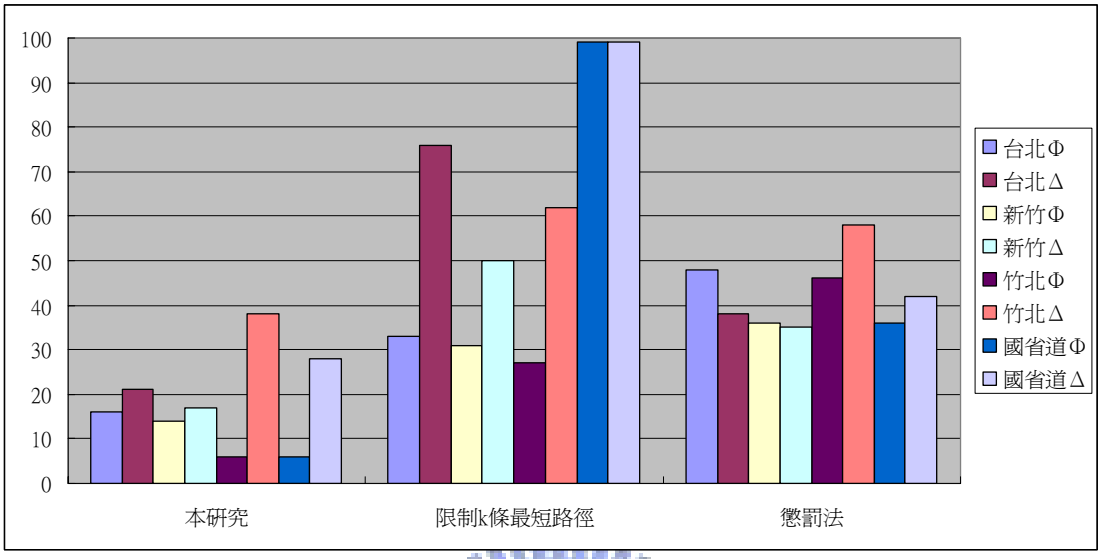



圖 4.3 各型態路網測試圖(k=5)

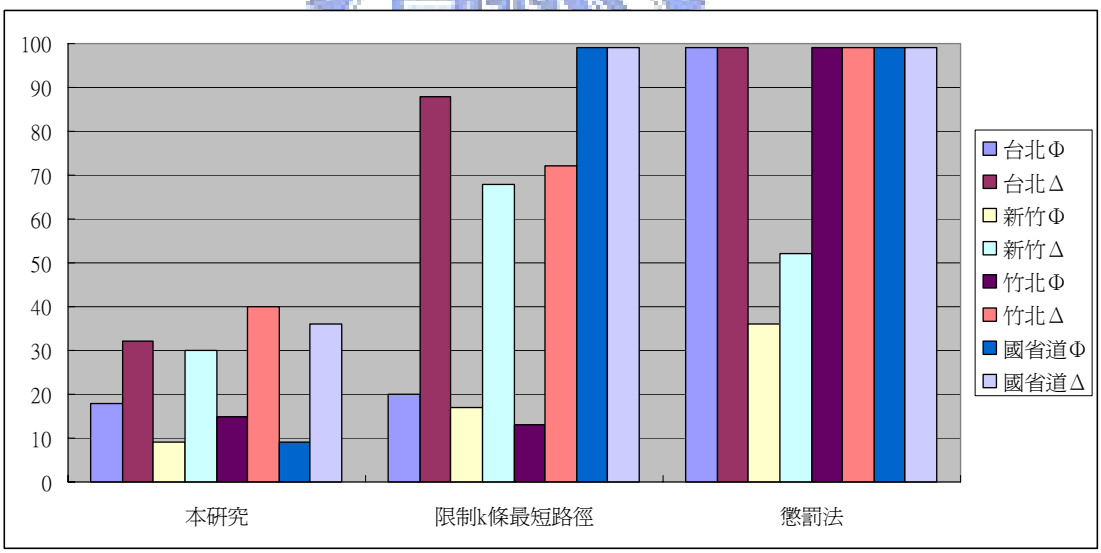



圖 4.4 各型態路網測試圖(k=10)

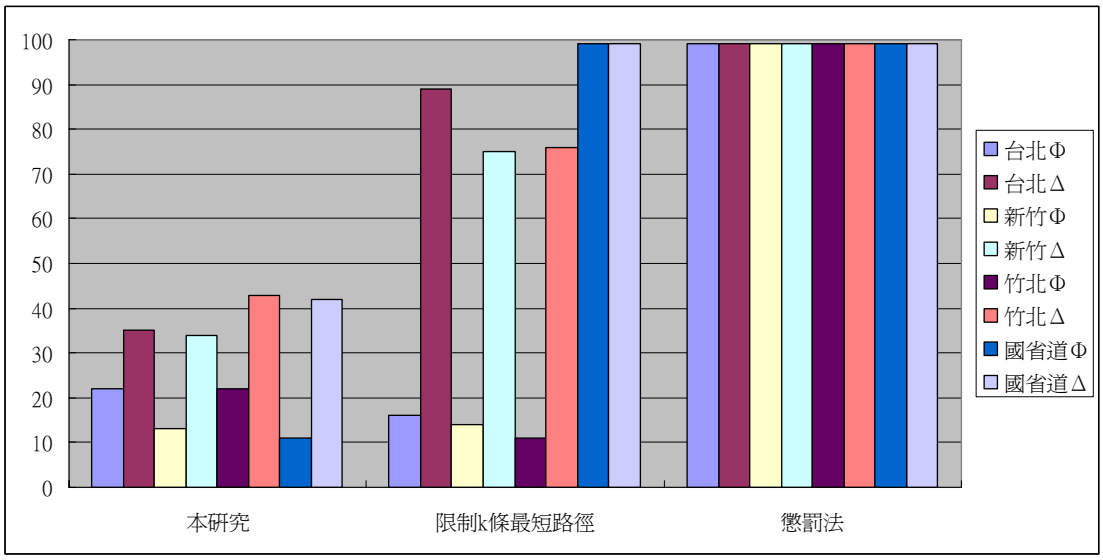



圖 4.5 各型態路網測試圖(k=15)

# 五、結論與建議

## 5.1 研究結論

在限制條件變動下，就求解品質而言，懲罰法在懲罰值較低時可求得較多的解，但會付出重疊度提高的代價，限制 k 條最短路徑也同樣遇到路徑重疊度過高的問題。而本研究的演算法在路徑差異度與路徑重疊度上，皆比懲罰法與限制 k 條最短路徑演算法得到更好的品質。而針對不同的路網求解時，本研究的演算法可適用於大型、中型都市、郊區以及國省道路網，相較於限制 k 條最短路徑演算法與懲罰法將更具備使用彈性。

就實際應用的角度而言，用路人可透過自行設定防止路段重複參數而彈性的使用本演算法，加大該參數的範圍將能獲得較大差異之替代道路。以國道春節假期而言，由於主要道路上雍塞情形較為嚴重，可使用較大的參數範圍；反之一般假日時，可選擇較小的參數範圍，保持使用路段為主要道路。本研究透過敏感度參數測試建議一般應用時可採取 $\alpha$=0.3、$\beta$=0.7 進行，將可獲得較為平均的重疊與繞道比例。

本研究的演算法可在有限的時間內透過預先刪除易重複路段，以及加入最大重疊度判斷的權重影響，求得滿足需求數量且具有差異性的最短路徑，同時這些路徑間的成本差距也在一定的範圍之內，足以做為替代道路資訊發佈的求解演算法。

## 5.2 後續研究建議

1. 路段屬性影響

本研究為簡化演算複雜度，未考慮路段屬性的差異。但實際上用路人對於高快速道路、以及一般道路的選擇偏好是不同的。因此本研究建議如需考慮路段屬性，可在不更動演算法的前提下，透過下列兩種方法執行：

a. 將目前採用之距離成本轉換為旅行時間成本。用路人在選擇路段時，主要是透過時間成本進行考慮，例如從新竹到竹北時，即使經省道與國道的距離成本相差無幾，但道路的屬性影響了車行速度，進而影響旅行時間，因此可透過輸入成本的轉換，達到此目的。

b. 將目前採用之距離成本乘上路段屬性權重值。此為較直觀之解決方案，同樣只需將目前做為路段成本之距離成本乘上一用路人的偏好權重，即可達到目的。但由於用路人對道路使用之偏好差異較大，即使是同種路段也可能造成不同之選擇偏好。因此此數值建議可透過一詳細的質化道路偏好研究取得。

2. 路口轉向限制與路段行進限制

於本研究之演算法中，僅考慮點與線之連接關係，並未考慮額外的屬性限制。若於實際應用中有此需求，建議可透過新增節點、節線的方式表示各個轉向路口之轉向限制，如此仍可操作本演算法求得答案。

3. 無謂繞道的消除

由於本演算法是透過 Yen 的最佳性原則求解替代道路，在刪除關鍵路段，產生差異路徑此步驟上，必會產生一定程度的繞道。在本研究中雖已對整體路徑之繞道成本進行控制，但仍有可能在某路段發生一段無謂繞道。本研究為解決該問題，曾提出一截彎取直演算法雛型：「檢查路徑上任兩點之距離，是否大於最短距離之 $\gamma$ 倍，若是則將路徑上該兩點改由最短路徑連接」。但由於參數取得困難，同時也會造成演算複雜度過高，因此無法使用。建議後續研究可針對此議題進行一系列之數值實驗，提供不同狀況下之建議參數，藉以提高所求替代道路之合理性。

# 參考文獻


1. J. A. Azevedo, M. E. O. Santos Costa, J. J. E. R. Silvestre Madeira, and E. Q. V. Martins, An algorithm for the ranking of shortest paths, Europe Journal of Operation Research, 69(1993), pp. 97-106.
2. G.-H. Chen and Y.-C. Hung, Algorithms for the constrained quickest path problem and the enumeration of quickest paths, Computers & Operation Research, 21 (1994), pp. 113-118.
3. Y. L. Chen, An algorithm for Finding the k quickest paths in a network, Computers & Operation Research, 20 (1993), pp. 59-65.
4. Y. L. Chen, Finding the k quickest simple paths in a network, Inform. Process. Letter, 50 (1994), pp. 89-92.
5. B. L. Fox, k-th shortest paths and applications to the probabilistic networks, in ORSA/TIMS Joint National Mtg., Vol. 23, 1975, p. B263.
6. N. Katoh, T. Ibaraki, and H. Mine, An efficient algorithm for K shortest simple paths, Networks, 12 (1982), pp. 411-427.
7. E. L. Lawler, A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem, Management Science, 18 (1972), pp. 401- 405.
8. E. Q. V. Martins, An algorithm for ranking paths that may contain cycles, Europe Journal of Operation Research, 18 (1984), pp. 123-130.
9. J. B. Rosen, S.-Z. Sun, and G.-L. Xue, Algorithms for the quickest path problem and the enumeration of quickest paths, Computers & Operation Research, 18 (1991), pp. 579-584.
10. D. R. Shier, Iterative methods for determining the k shortest paths in a network, Networks, 6 (1976), pp. 205-229.
11. D. R. Shier, On algorithms for finding the k shortest paths in a network, Networks, 9 (1979), pp. 195-214.
12. J. Y. Yen, Finding the K shortest loopless paths in a network, Management Science, 17 (1971), pp. 712-716.
13. Vedat Akgun, 2000, On finding dissimilar path, European Journal of Operational Research, 121(2000), pp. 232-246

14. N.J. van der Zijpp, Path enumeration by finding the constrained K-shortest paths, Transportation Research Part B, 39 (2005), pp. 545-563
15. Marta M.M. Pascoal, An algorithm for ranking quickest simple paths, Computer & Operation Research, 32(2005), pp. 509-520
16. David Eppstein, Finding the k shortest paths, Society for Industrial and Applied Mathematics, 28(1998), pp. 652-673
17. Yongtaek LIM , A SHORTEST PATH ALGORITHM FOR REAL ROAD NETWORK BASED ON PATH OVERLAP ,Journal of the Eastern Asia Society for Transportation Studies, 6 (2005), pp. 1426 – 1438

# 簡　歷

姓名：陳冠佑

出生年月日：1984年8月3日

出生地：台南縣

學歷：

2009年6月　國立交通大學運輸科技與管理學系碩士班
2006年6月　國立交通大學運輸科技與管理學系
2002年6月　台北市立建國高級中學

聯絡方式：foryuchen@gmail.com