國立臺灣大學工學院土木工程學研究所

碩士論文

Department of Civil Engineering

College of Engineering

National Taiwan University

Master Thesis

依時性 A*路徑演算系統開發

－以國道高速公路路網為例

Development of Time-Dependent A* Route Planning System: A Case Study of Taiwan Highway System

謝伯嘉

Hsieh, Bor-Chia

指導教授：張堂賢　教授

Advisor：Prof. Chang Tang-Hsien

中華民國 102 年 6 月

June, 2013

致　　謝

我要感謝指導教授張堂賢教授兩年來細心指導、耐心栽培，提供許多學習及研究的資源，您的付出幫助我提升至更高的境界。感謝台大土木系交通組的所有老師，你們的每一堂課都為我的研究打下無法撼動的基礎。感謝楊櫟凱、闕嘉宏兩位學長於研究期間能夠不辭辛勞提供協助，對於本篇研究提出許多非常具有建設性的意見，你們幫助我的研究更加豐富完美。感謝楊傑理、劉姿君以及交通組R00 的所有同學，一起奮鬥的時間總是有你們支持著我。感謝洪于翔、樓軒宇、薛雅云學弟妹，你們為我分擔了不少瑣碎的雜事，使我研究之路能夠無後顧之憂。

另外我要感謝我的父母，這些年來，無論我做了什麼決定，你們都會在我背後默默的支持我，是我前進下去的動力。最後我要感謝我自己，研究所兩年來，辛苦了。

摘　　要

「條條大路通羅馬」，交通運輸的快速發展，使得我們要前往目的地，時常都有許多選擇，然而因為有許多不同的選擇，該如何選擇就變成了另一個困擾。市面上有許多路徑導引程式，不論「行前路徑導引」或「途中路徑導引」，多以空間最短路徑為決策目標，然而實際使用的時候，卻時常發現導引程式提供的最佳路徑車多雍塞。

為解決傳統路徑導引程式往往僅依照空間距離做為路徑導引決策的變數，未考慮實際交通路況的問題，本研究使用路段旅行時間做為路徑選擇的成本，建置一適用於國道高速公路路網的依時性 A*路徑演算系統。以 A* Algorithm 做為路徑演算的基礎，將整合車輛偵測器(VD)與電子收費系統(ETC)資料的旅行時間資料庫做為依時性演算法之資料來源，並加入後推式演算法以配合不同使用者的需求。使用者輸入起點、訖點以及預計出發/抵達時間，系統就會根據旅行時間資料庫提供的路段旅行資料，提供建議旅行路徑與總旅行時間。

本研究驗證 A* Algorithm 能夠有效限制路徑搜尋的方向，以較短的時間取得令人滿意的建議路徑，這在路網龐大複雜時，能夠有效縮短運算時間，節省運算資源。本研究同時驗證，後推式演算法雖然需要較長的演算時間，但是演算結果與前推式演算法一致，能夠節省使用者欲於特定時間抵達目的地而反覆求解的時間。

關鍵詞：依時性最短路徑問題、Dijkstra's Algorithm、A* Algorithm、後推式路徑演算法、路徑導引系統

Abstract

"All roads lead to Rome." Because of the developing of transportation, we usually have a lot of way that we can pick to reach a destination. With the choices increasing, which way should we choose become another problem. There are kinds of route guidance program in the market. Whatever "pre-trip route guiding" or "en-route route guiding," most of them use the length of path to make a decision. However we can usually found out that the route it provides always in a traffic jam.

To solve the problem that traditional route guidance programs usually pick out the shortest route but not the best route, this study replaces the spacing cost with traveling-time cost to build a time-dependent A* route planning system which can match highway characteristic. This program based on A* Algorithm and uses VD and ETC data as data sources. It also contain time-dependent backward algorithm to fit different travel planning demands. Users just input origin, destination, and expectation departure/arrival time, system will output recommend route and total traveling time.

This study verify that A* Algorithm can limit route-searching direction to find a satisfactory route within a short time. While the road network coming huge and complex, it can efficaciously saving calculating time. This study also verify that backward algorithm get the same result as forward algorithm, though backward algorithm take more time to solve the problem.

Keywords: the time-dependent shortest path problem, Dijkstra's Algorithm, A* Algorithm, the backward route algorithm, route guidance system

目　　錄

圖　目　錄

表　目　錄

第一章　緒論

1.1 研究背景與動機

「條條大路通羅馬」，交通運輸的快速發展，使得我們要前往目的地，時常都有許多選擇，然而因為有許多不同的選擇，該如何選擇就變成了另一個困擾。市面上有許多路徑導引程式，不論「行前路徑導引」或「途中路徑導引」，多以空間最短路徑為決策目標，然而實際使用的時候，卻時常發現導引程式提供的最佳路徑車多雍塞，花了許多不必要的等待時間，飽受一肚子塞車的怨氣，才好不容易到達目的地。

人在長途移動的時候，對時間的感受性比距離更加敏感，對大部分使用者而言，旅行時間的長短遠比距離的遠近更為重要，因此我認為一套以旅行時間做為路段旅行成本而設計的路徑決策系統是必要的。然而相較於固定不變的旅行距離成本,會隨著時間不斷變動的旅行時間成本則是相當難以預測，隨著旅行時間增加，對於路段旅行時間的預測將可能產生越大的偏差，這將影響路徑決策的準確度，因此需要一個可靠的路段旅行時間資料庫提供準確的路段旅行資料，以支持路徑決策的準確度。

對於上班(上學)或者欲轉乘大眾運輸工具(如飛機、火車、高鐵)等受限於時間壓力或者必須按照時刻班次表的通勤旅客,若以出發時間預估旅行時間，往往需要以不同的出發時間測試數次，才能找到比較合適的出發時間，若提早抵達，將無法有效率的利用等待後續行程的等待時間，更遑論遲到可能減薪或錯過班次造成的損失；若是以預計抵達時間查詢，則可以透過「後推式路徑演算法」提供使用者最晚出發時間，讓駕駛人能夠有效利用行前時間、縮短途中旅行時間及抵達後等待時間。巨觀來看，將能提升整體路網的運輸效率。

1.2 研究目的

Dijkstra's Algorithm 是目前求解一對一最短路徑問題的主要理論基礎[1]，適用於路網節線成本皆為非負的情況。雖然 Dijkstra's Algorithm 能夠不用遍巡所有的節點就有機會找到最短路徑，然而在路網龐大複雜的情況下，Dijkstra's Algorithm 卻有可能浪費許多時間在搜尋明顯不合理的節點。A* Algorithm 可以有效限制搜索範圍，使得搜尋方向能夠往訖點前進。本研究以 A* Algorithm 為基礎,建置一適用於國道高速公路路網之最佳路徑決策系統，期望能節省駕駛人的旅行時間成本，提高國道高速公路整體運輸效率。

1.3 研究範圍

本研究目的在提供國道高速公路旅次最佳路徑選擇，主要適用對象為所有使用此兩條國道的小客車駕駛，包括：通勤者、觀光旅客、返鄉族群、以及其他使用國道的旅客等。本系統以國道一號(中山高速公路)以及國道三號(福爾摩沙高速公路)全線之 ETC 及 VD 資料為資料選擇基礎，可提供使用者於此兩條國道上選擇路徑或轉換路徑之參考。本系統設計具有適應性，若以不同路網做為參考資料，亦可作為其他路網之路線決策系統。

1.4 研究內容與流程

本研究流程如圖 1-1，以下分章敘述：

1. 緒論

   說明研究背景與動機，確立研究目的、內容及流程，完整界定研究範圍以及欲解決問題。

2. 文獻回顧

   回顧國內外有關最短路徑與路徑導引問題相關文獻，分析各演算法之特性、限制條件、優缺點與適用環境，評估其中適合本研究範圍之演算法，修改其演算法架構以符合本研究之問題。

3. 研究方法

   說明本研究方法，主要包括 Dijkstra's Algorithm 與 A* Algorithm。

4. 系統建構

   說明程式運作邏輯、系統運作流程，並針對資料儲存型態、重要子程式與資料轉換方法進行詳盡說明，建構「依時性 A*路徑演算系統」。

5. 路徑規劃實作

   以國道高速公路為實驗研究範圍，利用 JAVA 程式語言架構依時性 A*路徑演算系統，並比較不同種類 A*演算法之優劣。

6. 結論與建議

   對本研究過程作具體結論，並提出後續研究可行方向。

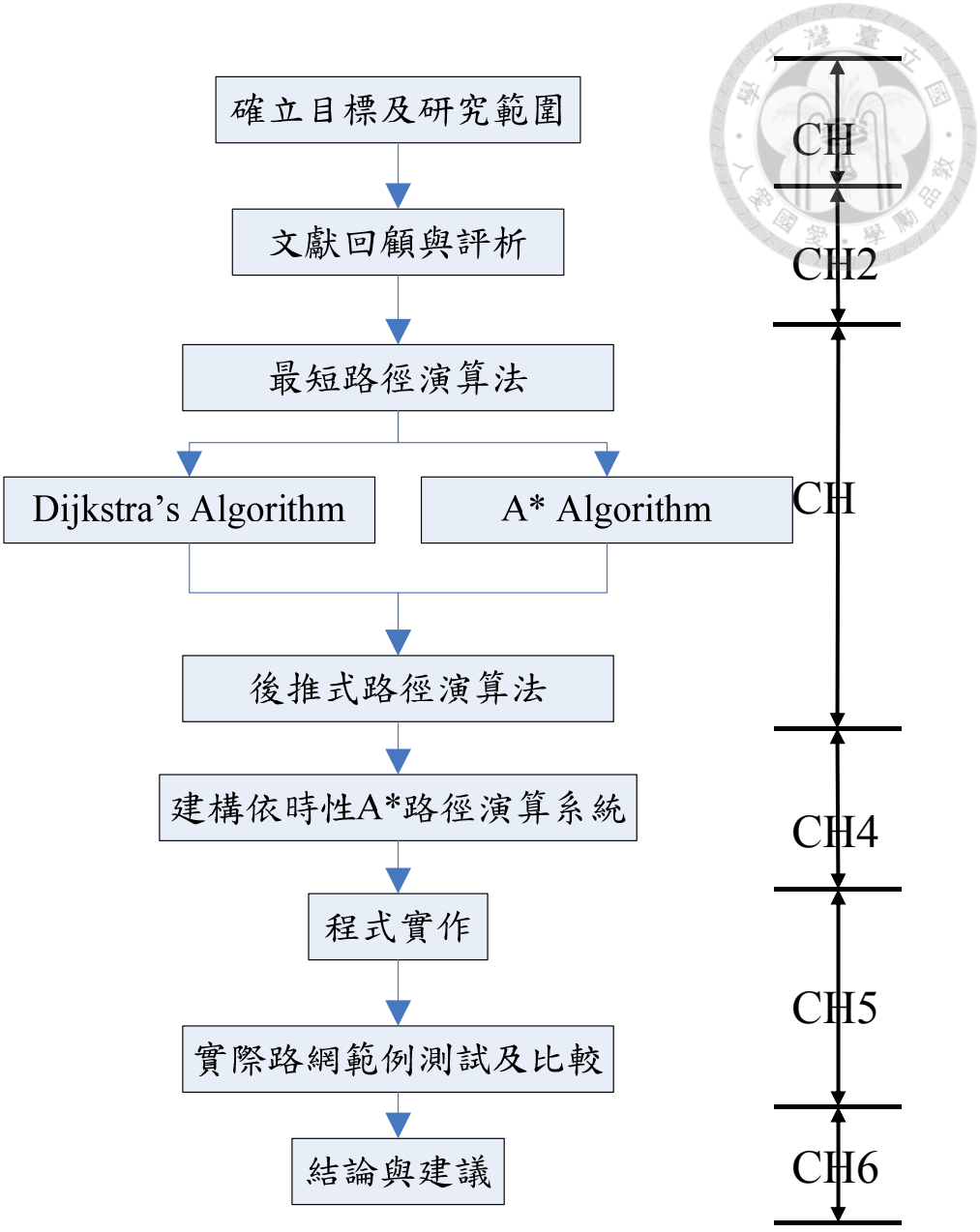

確立目標及研究範圍
文獻回顧與評析
最短路徑演算法
Dijkstra's Algorithm
A* Algorithm
後推式路徑演算法
建構依時性A*路徑演算系統
程式實作
實際路網範例測試及比較
結論與建議
CH
CH2
CH
CH4
CH5
CH6

圖 1-1 研究流程圖

第二章　文獻回顧

本章回顧國內外最短路徑問題及依時性最短路徑問題相關文獻，參考前人做法，建構符合本研究之路徑規劃演算系統。2.1 節簡述智慧運輸系統(ITS, Intelligent Transportation System)及其子系統；2.2 節列出因交通資訊型態而有不同的路徑規劃類型；2.3 節說明各種最短路徑問題及其適用之演算法；2.4 節探討 Dijkstra's Algorithm 與啟發式演算法 A* Algorithm 之優缺與應用。

2.1 智慧型運輸系統(ITS)

智慧型運輸系統(ITS, Intelligent Transportation System)是將先進的信息技術、通訊技術、感測技術、控制技術及計算機技術等，有效率的集成運用於整個交通運輸管理體系，而建立起的一種在大範圍內及全方位發揮作用的，實時、準確及高效率的綜合運輸和管理系統。

ITS 係利用電子、通訊、導航、乘客資訊、電腦、控制等技術，整合人、車、路等各種運輸系統、運具，提供即時正確的交通資訊，增加運輸系統安全、效率、經濟生產力、減少擁擠、汙染，讓交通資源發揮最大效用。ITS 包含七個子系統：

1. 先進交通管理系統(ATMS, Advanced Traffic Management System)
2. 先進旅行者資訊系統(ATIS, Advanced Traveler Information System)
3. 先進大眾運輸系統(APTS, Advanced Public Transportation System)
4. 商車營運系統(CVO, Commercial Vehicle Operation)
5. 緊急救援系統(EMS, Emergency Management System)
6. 先進車輛控制及安全系統(AVCSS, Advanced Vehicle Control and Safety System)
7. 自動公路系統(AHS, Automated Highway System)

2.1.1 先進旅行者資訊系統(ATIS)

ATIS 是和使用者有密切關係的子系統，主要提供使用者藉由先進資訊、通訊及其他相關技術，使其能於車內、家裡、辦公室、車站等地點方便的取得所需之資訊，也是和民眾最直接接觸的部分[2]。

ATIS 可以依照提供對象的不同，分為駕駛者及大眾運輸使用者兩大類。駕駛者交通資訊系統又依照查詢的時程階段分為行前(pre-trip)資訊系統與途中(en-route)資訊系統；大眾運輸使用者交通資訊系統則分為行前、車內與車外資訊系統。行前資訊系統提供旅行者在旅次產生前於家中、辦公室、大型活動中心、購物中心等旅次起點，以手持行動裝置、數據網路或資訊亭(kiosk)，向雲端資訊中心查詢即時路況、大眾運輸班次、轉乘資訊、交通管制情報等交通資訊，以這些情報輔助進行旅次行為決策，如預估出發時間、路徑決策、運具、大眾運輸路線選擇等。

「台灣地區發展智慧型運輸系統系統架構之研究」制定的 ATIS 分為九個項目：廣播是旅行者資訊、互動式旅行者資訊、自主式路徑導引、動態式路徑導引、資訊服務提供者式之路徑導引、整合式運輸管理及路徑導引、資訊查詢服務及預約、動態式共乘車內顯示[3]。

2.1.2 行前資訊系統

行前資訊系統係於旅次產生前，提供使用者所需歷史性、即時性或預測性交通資訊，讓使用者得參考此資訊制定旅程規劃，藉此影響使用者行為；而途中資訊系統則是在旅次行為中，提供使用者即時性或預測性交通資訊，協助使用者於旅次進行中改變路徑決策或旅行行為。

對駕駛者旅次行為而言，行前資訊系統有助於旅次的制定、變更、甚至取消；途中資訊系統則只能在途中進行旅次的修改。顯然行前資訊系統較途中資訊系統對駕駛者的影響更大。若無法提供即時正確的交通導引資訊，ATIS

的效益將會大打折扣。有關行前資訊和途中資訊對於旅次行為的潛在影響，如表 2-1 所示。交通資訊與車輛間關係，如圖 2-1 所示，本研究對象包含所有小汽車駕駛者，無論其是否有搭載車內資訊系統。

表 2-1 行前資訊和途中資訊對旅次行為潛在影響

行為反應/資訊系統	行前資訊(家中)	行前資訊(戶外)	途中資訊(車內/路側)
旅次產生/取消	V		
改變出發時間	V	V	
改變路徑	V	V	V

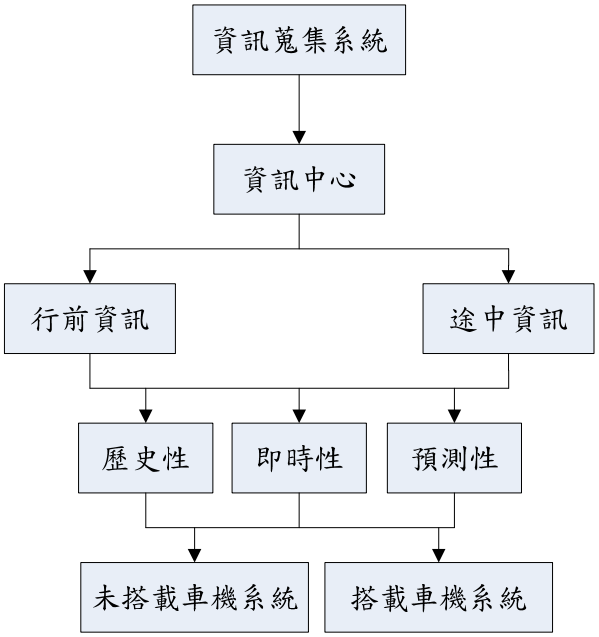


圖 2-1 交通資訊與車輛間之互動關係

2.1.3 旅行者資訊系統的效益

在倫敦，Smith and Russam 比較搭載資訊系統依照最短路徑行駛的車輛與未搭載資訊系統依照經驗路徑行駛的車輛，發現搭載資訊系統的車輛使得整體路網系統減少了 2.5~6%的旅行時間[4]。French and Queree 蒐集每日不同時段的路段旅行時間，搭配 AUTOGUIDE 即時性交通資料進行動態指派測試，

顯示旅行時間節省 20%，其中 5%來自即時性資訊效果，另外 15%來自歷史資料對未來路況的預測[5]。

在美國，ADVANCED 系統利用歷史路段旅行時間進行路徑規劃，在交通路況無異常的情況下，利用實驗路網結合靜態路徑選擇模式，以離線方式估計路段旅行時間與不同時段(上午尖峰、中午、下午尖峰、傍晚及夜晚)每一起訖點的流量。在短期旅行時間預測上，分為靜態和動態兩種型態。靜態預測是利用與預測時段相同的日型態與時段的歷史資料進行預測，因此事先須將交通資料依照不同星期、天候、路段、時段作適當分類；動態預測是利用交通資訊中心提供的旅行時間資訊，改善靜態預測。若無法提供動態預測資訊時，則改採靜態預測資訊。研究結果顯示，靜態預測在週末及離峰時段的預測結果相當良好；但靜態預測在尖峰時段旅行時間的預測能力則非常差，須用動態預測方式改善[6]。

在荷蘭，大多數的計程車透過 RBS 規劃系統輔助進行排程作業。因每天約有十萬至十五萬筆的最短路徑運算要求，為了處理如此龐大的資料量，運算時間勢必要越短越好。BRS 決定計程車的路徑、總旅行時間以及車資，因此必須是最佳路徑。Klunder and Post 認為依照地理位置事先計算路網中任兩點路徑並不可行，因為行駛時段不同，兩點間的最快路徑也會不同；也認為在真實路網相當龐大的情況下，不同的最短路徑演算法將會大大的影響運算時間[7]。

2.2 路徑規劃類型

路徑規畫相關回顧中，Pallottino 和 Scutella 曾對交通領域之典型最短路徑演算法與創新概念進行彙整[8]，將路徑規劃內容分成下列三種類型：

1. 最短路徑搜尋過程類型：包含最短路徑樹求解技術[8][9][10]、Label-settig 演算程序[11]，Lebel-correcting 演算程序[12]、時間複雜度[13]以及最佳路徑更新演算[14]等相關研究；
2. 依時性最短路徑問題：以時變性路徑旅行時間作為成本計算，發展出特定出發時間之最快路徑搜尋模型[15][16]與時空窗最短路徑[17]等研究；
3. 其他目標函數之運輸模型：此類型以特定交通需求目標作為考量，如以懲罰與限制轉向行為下，進行最小成本路徑搜尋[18][19]或多重目標最短路徑搜尋[20]。

然而，於 Klunder and Post (2006)研究中[7]，曾對七種路徑演算法，以不同之變化方式(如: 單雙向搜尋指標和資料結構等) 衍生出 168 種版本進行實測，結果發現具備 bucket of queues 的 A*演算法相當具有效率且彈性的演算模型。

常見的路徑規劃目標多以最短旅行距離、最快旅行時間或最小旅行成本為主。以下將交通路徑規劃依照採用不同的交通資訊型態，區分為最短距離路徑、歷史依時性最快時間路徑、即時性最快時間路徑與預測性依時性最快時間路徑，分別說明如下：

1. 最短距離路徑(LDSP, Least Distance Shortest Path)

   僅考慮空間距離，不考慮交通量與不同時段對路段旅行時間造成的影響，簡單來說，只要攤開地圖就能夠規劃出最短距離路徑。此種路徑規劃適合交通量較低、離峰時段或時間變異性較小的狀況。此種規劃方法最單純，但是空間上距離最短的路徑卻有可能是時間上最久的旅行路徑，尤其在容易受交通量影響的都會區交通路網中，最短距離路徑較不符合最佳路徑選擇。

2. 歷史依時性最快時間路徑(HTDSP, Historical Time Dependent Shortest Path)

歷史依時性交通資料蒐集路網中所有路段於數天或數月前所有時段的路段旅行時間，以平均值提供給駕駛者，利用依時性最快時間路徑演算法作路徑規劃。歷史依時性最快時間路徑規劃對於交通量規律變化的路網有較好的適應性，但是無法反映突發性交通事件，較不具彈性為其缺點。

3. 即時性最快時間路徑(RTSP, Real Time Shortest Path)

即時性最快時間路徑是透過路側偵測器取得路網中的瞬時交通資料，能夠以當下的情況提供最佳的路徑規劃給使用者。即時性最快時間路徑的缺點是，若路網中大部分的使用者都使用相似的路徑規劃，一旦系統發布某處道路交通順暢訊息，該路段會立刻湧進許多駕駛，車輛密度突然增加使得該道路發生阻塞。交通狀況會隨著時間空間而不斷變化，當駕駛者經過一段時間到達該路段時，該路段之旅行時間可能已經與原本提供的即時資訊產生落差，使得最快路徑預估產生偏差，原先提供的路徑資訊並非為最快路徑。這種因為時間落差而產生的預估偏差將導致預估的正確性與可靠性降低，進而使駕駛者不信任系統提供之最佳路徑，降低資訊對其旅運決策的影響。

4. 預測依時性最快路徑(PTDSP, Predictive Time Dependent Shortest Path)

依時性路徑規劃演算法若未考慮未來交通情況，將可能產生時間落差而無法取得最佳解。預測性交通資訊兼具歷史性及即時性規劃的特色，蒐集歷史及瞬時交通資訊，透過卡曼率波、人工智慧、遞迴最小平方法、馬可夫鍊等預測模式，預估未來的路段旅行時間，配合依時性路徑規劃

演算法，降低即時性路徑規劃未考慮歷史時間資訊的缺點。預測依時性路徑規劃的準確程度與選用的預測方法有關。

預測性旅行時間資訊模式提供推估的路段旅行時間，如圖 2-2：從時間 0 開始，每隔五分鐘(ΔT)，推估未來一小時的交通狀況。以圖 2-3 為例，假設時間 0 自起點 A 前往訖點 C，若以即時路徑演算法計算，節線 1 和節線 2 在時間 0 的旅行時間分別是 8 和 3，因此自節點 A 到節點 C 的時間是 11；若以預測依時性演算法計算，時間 0 時節線 1 的路段旅行時間為 8，到達節點 B 時已經進入下一個時階，時間 8 時節線 2 的路段旅行時間為 4，因此自節點 A 到節點 C 的時間是 12。

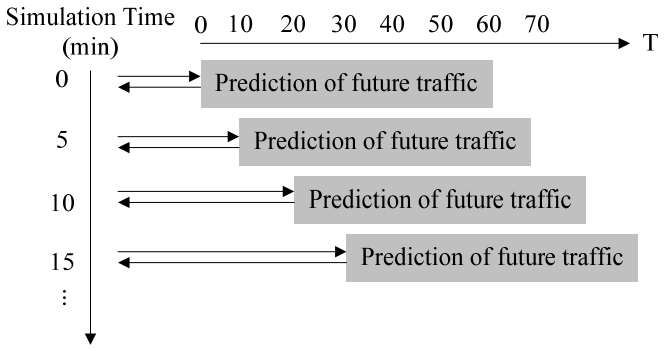
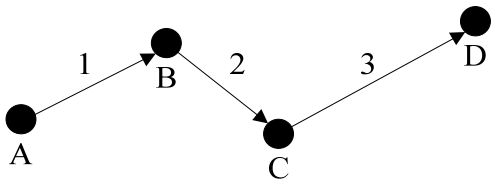


圖 2-2 預測旅行時間資訊模式架構



Min	Link 1	Link 2	Link 3
0~5	8	3	8
5~10	7	4	8
10~15	8	5	10
15~20	10	8	12

圖 2-3 簡單路徑及每路段每五分鐘的平均旅行時間

2.3 最短路徑問題(Shortest Path Problem)

2.3.1 路網結構資料

若要將最短路徑演算法應用在實務上，正確度(Accuracy)與時間複雜度(Time Complexity)都是非常重要的指標。如何在正確度與時間複雜度取得平衡點，路網資料結構是影響演算法成敗的關鍵因素。評估演算法績效有幾項重要指標，可用來評估路徑規劃演算法[21][22]：

(1). 正確性(Accuracy,計算結果的品質):以最佳解與該演算法的解答作比較，以其差距戰最佳解的百分比做為正確性的衡量標準。

(2). 時間複雜度(Complexity，計算實際路網的時間)：演算法執行速度與規畫需求的急迫性和欲求解問題的正確性有關，規劃需求的急迫性和正確性兩者有權衡關係，若規劃需求越急迫，演算法執行速度越重要，相對的正確性就會降低；反之，演算法執行速度會越慢，但是正確性會上升。

(3). 簡易性(Simplicity,執行的容易度):演算法難易程度影響到未來是否容易被廣泛使用。難以理解和編碼繁複的演算法，使用機會將較簡單直觀的演算法低。

(4). 具有彈性(Flexibility,解決其他最短路徑問題的延伸性):好的路徑規劃演算法應該具有高度彈性,能夠適應多種不同型態的路網,以利後續發展，符合不同地區的路網使用。

除了路徑演算法本身，儲存路網節點的資料結構亦會影響最短路徑運算效率。常見的路網結構有下列幾種：

(1). 權數矩陣(weight matrix)：

權數矩陣為路網節點結構中最簡單的一種資料結構。資料以陣列形式儲存，陣列中每一個元素代表每個路段的旅行時間；然而在實際路網

中，大部分的節點只會與少數節點相連接，容易使矩陣形成稀疏矩陣，浪費記憶體空間，使得運算效率不佳。

(2). 相鄰串列(node adjacency-list)：

利用相鄰串列，將路網中與某一點相連的所有節點，利用串列(linked)的方式串接在一起。路網若存在 n 個節點，就會存在 n 個串列，從某節點連接的節線都會放在同一個串列中，因此相鄰串列描述路網是相當便利的。每個節點的連接屬性分成三個主要部分：節點、路段資料及指標，路段資料部分不僅可儲存路段旅行時間，也可儲存其他與此路段相關的屬性資料。

(3). 星結構(star structure)：

星結構主要是利用指標(pointer)建立起訖點關係，在程式設計上只須指標便能代表路網結構；但在路段成本修改和程式設計上較為不便。星結構依儲存方式分為兩種：向前星法及向後星法，兩種結構差別在於起點與訖點位置不同。

A. 向前星法(forward star)：又稱指標法，資料行是由三個陣列組成：指標、起點、訖點路段成本。其中指標行代表起點與訖點相關的指標。以(起點，訖點)=∞表示節點間無路段相連，節省記憶體空間。

B. 向後星法(backward star)：類似向前星法，不同的是資料形式：指標、訖點、起點路段成本。

2.3.2 **路網基本定義**

以 G 表示一方向性路網$G=(N,A)$，N 為節點集合，$|N|=n$為所有節點數；A 為節線集合，$A=\{(u,v)\in N\times N\}$，節線$(u,v)\in A$表示從節點 u 可達到節點 v，$|A|=m$為所有節線數，$1\le n<\infty, 0\le m<\infty$；$C_{uv}$表示節線$(u,v)$的

路段旅行成本(路段旅行時間、路段長度、道路收費標準等)；$C_{uv}(t)$表示時間t進入節線(u,v)的路段旅行時間，顯示路段旅行時間依車輛進入路段時間點而有所不同。

從起點 O 至訖點 D 的路徑 P 以一連串有順序的路段定義：$P=(v_1,v_2),\dots,(v_i,v_{i+1}),\dots,(v_{k-1},v_k)$，$v_1$代表起點 O，$v_k$代表訖點 D，$(v_i.v_{i+1})\in A\ for\ i=1,\dots,k-1\ and\ v_i\neq v_j\ for\ all\ 1\leq i\leq j\leq k$，加總各路段旅行時間即是路段旅行時間$C(P)=\sum_{(u,v)\in P}C_{uv}$。

2.3.3 最短路徑問題

最短路徑問題(Shortest Path Problem, SPP)是網路研究或圖論(Graph Theory)中極重要的一部分。常見的最短路徑問題包括：物流運輸、汽車導航系統、電腦通訊網路，甚至緊急救難時提供人員疏散等緊急救難決策方案，如：地震、戰爭、核電廠災變等重大災難。不管求解目標為最短距離、最快時間或最少成本等目標或其他最短路徑演算法類型(如：K 條最短路徑演算法或依時性最短路徑演算法)，運算核心階在最短路徑演算法的基礎上。最短路徑演算法有諸多相關研究；運輸模式下也有最短路徑演算法的相關研究。各個最段路徑演算法的搜尋範圍和搜尋方式不同，時間複雜度因此也會有所差異，即使路徑目標皆是最短路徑，但經過路段可能不盡相同。若將最短路徑演算法應用在即時資訊環境中，當交通環境改變，路段旅行時間資料需要更新，以符合依時性路徑規劃要求。

最短路徑問題可以依照起訖點個數以及出發、抵達時間分類。依照起訖個數可以將最短路徑問題分為四類：

(1). 一對一最短路徑問題：特定起點至特定訖點的最短路徑；

(2). 一對多最短路徑問題：特定起點至網路上各點的最短路徑；

(3). 多對一最短路徑問題：網路上各點至特定訖點的最短路徑；

(4). 多對多最短路徑問題：網路上任意兩點的最短路徑。

依出發或抵達時間型態亦可將最短路徑問題分為四類：

(1). 特定出發時間之最短路徑問題；

(2). 特定出發時間之最短路徑問題；

(3). 所有出發時間之最短路徑問題；

(4). 所有抵達時間之最短路徑問題。

路徑規劃問題演算法通常分為兩類：求解最佳解(optimal solution)的最佳演算法(optimal algorithm)；求解近似解(approximate solution)的啟發式演算法(heuristic algorithm)。最佳解演算法雖然能夠找出最佳解，但是時間複雜度會隨著路網的規模指數成長，在實務應用上往往無法忍受過久的演算時間，須重新設計演算法策略，才能應用在即時路徑導引系統上；而啟發式演算法雖然不一定能找到最佳解，但是能夠在有限的時間內求得可以接受的次佳可行解(suboptimal feasible solution)。

最佳演算法基本上是應用動態規畫法不斷遞迴，決策從起點(或從訖點)到訖點(或起點)的最短路徑。假設$L_{(i)}$是從起點到某節點 i 的路徑成本標籤(即節點 i 的標籤(label))；P 為目前搜尋路徑上連接某一節點的前一段路徑，如：$P_{(i)}$表示最短路徑上連接到節點 i 的前一路段。Q 表示合格(scan eligible)的節點集合。以下是假設從起點開始的最短路徑演算法步驟：

Step 1: Initialization:	$Set\ i = O; L_{(i)} = 0; L_{(j)} = \infty\ \forall\ j \neq i; P_{(i)} = NULL$
	$Define\ the\ scan\ eligible\ node\ set\ Q = \{i\}$
Step 2: Node Selection:	$Select\ and\ remove\ a\ node(i)\ from\ Q$
Step 3: Node Expansion:	$Scan\ each\ link\ emanatiog\ form\ node\ i$
	$For\ each\ link\ \ a = (i,j)$
	$if\ L_{(i)} + c_a < L_{(j)}$
	$then\ L_{(j)} = L_{(i)} + c_a; P_{(j)} = a$
	$Insert\ node\ j\ into\ Q$
Step 4: Stopping Rule	$if\ Q = \emptyset\ then\ STOP\ otherwise: goto\ step\ 2$

文獻上求解一對一最短路徑問題的演算法主要有兩類：標記修正法(LCA, Label Correcting Algorithms)與標記設定法(LSA, Label Setting Algorithms)。兩類演算法主要差異在最短路徑演算法步驟中 scan eligible node set 的資料結構格式和節點定義和選取的方式。以下分別詳述標記修正法(LCA)與標記設定法(LSA)。

(1). 標記修正法(LCA)

LCA 相當於動態規劃(dynamic programming)，不侷限在非負節線成本路網，有負值線成本存在時仍適用，前提是路網中不能存在負迴圈(negative cycle)，以 Bellman algorithm 為代表，圖 2-4 為 Label Correcting Algorithms 流程圖。

初始步驟設定一佇列 Q，儲存所有待檢驗節點(包括起點或訖點)。若 Q 為空集合則演算結束；否則從 Q 中取出一節點 i，檢查從 i 分支出去的節線，若滿足$L_{(i)} + w(i,j) < L_{(j)}$，則更新節點 j 標籤$L_{(j)} = L_{(i)} + w(i,j)$。LCA 將所有節點的成本標籤視為暫時標籤(temporary label)，直到 Q 為空集合時，才將所有節點的暫時標籤變成永久標籤(permanent label)。LCA 若要得到最短路徑，一定要遍歷路網中每個節點，才能知道兩點間的最短路徑。LCA 常適用於一對多最短路徑問題(one-to-all shortest path)。以下為 LCA 演算步驟：

起始：

Step 1： 令起點 O 為永久標籤，並移至集合 S，$L_{(O)} = 0$，$P_{(O)} = NULL$，其他非起點 O 的節點 i 皆設為暫時標籤$L_{(i)} = \infty$。

改變標記：

Step 2： 從集合 Q 中選擇最前面的節點，將此節點設為 i，並自集合 Q 移除。

Step 3： 找出與節點 i 相連的所有節點 j，並視為暫時標籤。

Step 4： 計算節點 j 的$L_{(j)} = min\left[L_{(j)}, L_{(i)} + w(i,j)\right]$。

Step 5： 若集合 Q 為空集合，將所有節點移至集合 S；若否，跳回 *Step 2*。

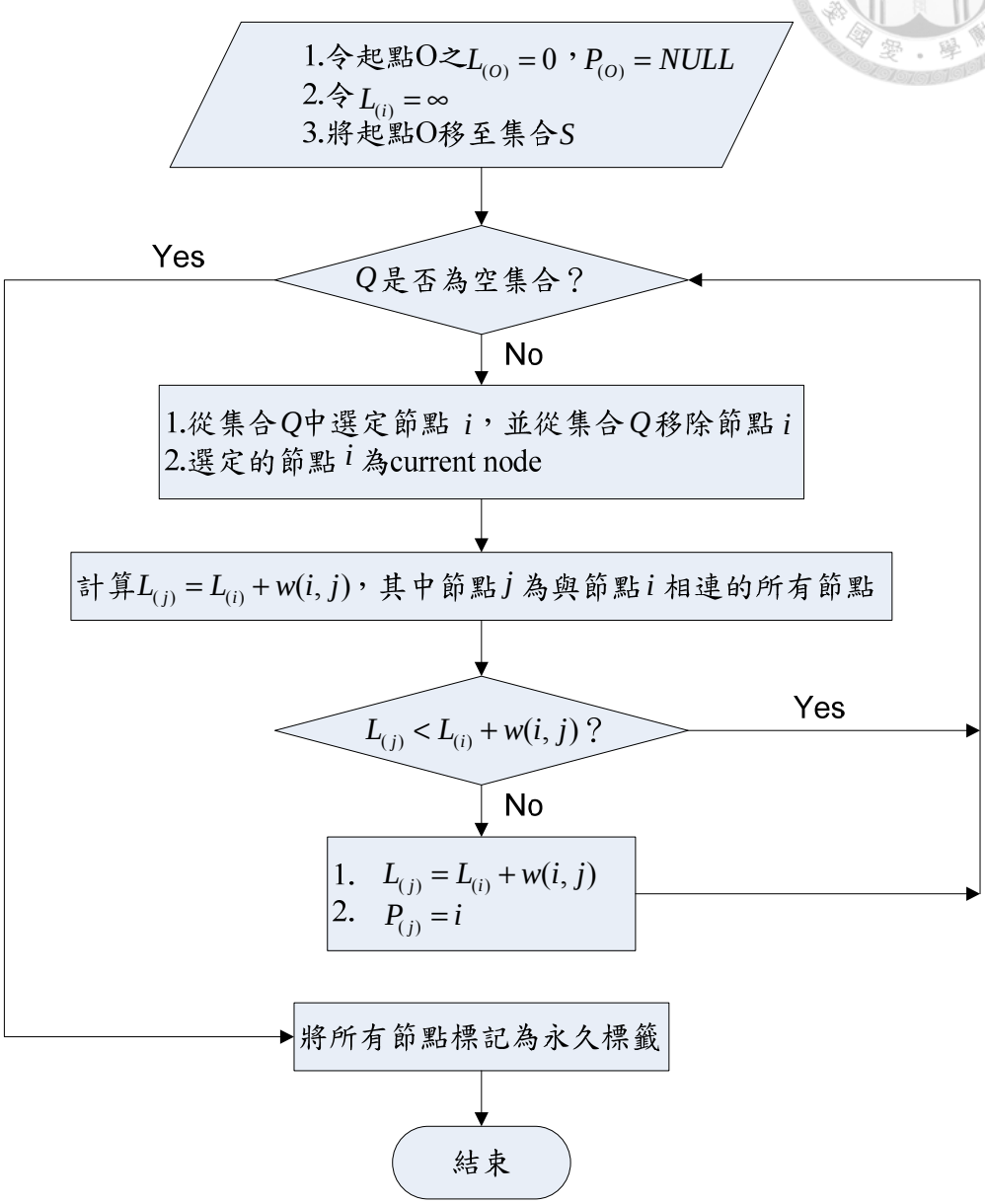


圖 2-4 Label correcting algorithm 流程圖

(2). 標記設定法(LSA)

LSA 必須符合兩個條件：(1)非循環的網狀系統、(2)網路節線成本必須為非負。演算法原理是：除起點的永久標籤設為零外(若為依時性最短路徑問題，則設為出發時間)，首先對每一節點標記為暫時標籤，先設為無限大，目的是

紀錄起點至該節點到目前為止的總旅行時間；一次只考慮一個節點，計算由起點出發至下游各節點的路徑旅行時間。對各個下游節點而言，若抵達時間小於原標記值，則表示此路徑較原先路徑為佳，更新該節點標籤。此法僅適用於節線成本為非負的情況，會比較所有暫時標籤的節點，將標籤值最小者的節點移到永久標籤，表示找到由起點至該節點的最短路徑。利用永久標籤(permanent label)與暫時標籤(temporary label)，反覆將累積旅行時間最小的節點由暫存標籤移至永久標籤，值到暫存標籤為空集合或訖點從暫存標籤移到永久標籤，代表找到一條由起點至訖點總旅行時間最少的路徑。

LSA 中各最短路徑演算法的主要差異在維持 *scan eligible node se*t 內排序方法的資料結構上。LSA 適用於只求起訖點的一條路徑，當訖點成為永久標籤時，即停止運算，為一對一最短路徑問題(one-to-one shortest path)。LSA 各演算法中較具代表性也最廣泛應用的演算法屬 Dijkstra's Algorithm[11]，於 2.4 詳加敘述。

(3). 評析標記標記修正法(LCA)、標記設定法(LSA)

LCA 和 LSA 共同的演算流程是：將起點成本先設為 0，其他節點成本設為無限大，由起點開始搜尋下游節點並將其置入暫存標籤中，接著從暫存標籤中選取節點更新節點成本，直到路網全部節點都處理完畢。LCA 與 LSA 不同處在於暫存標籤中選取節點的方式和何時收斂得到最短路徑。LCA 利用先進先出(FIFO, First in first out)的概念，從暫存標籤中選取最靠近起點的節點進行計算，唯有計算完全部節點時才會知道各節點成本和最短路徑，在此之前皆為暫存值；LSA 在選取節點時，會將暫存標籤中的節點成本值進行比較，選出暫時標籤最小值的節點繼續計算。

在時間複雜度上，LCA 時間複雜度為$O(n^3)$，LSA 時間複雜度為$O(n^2)$；但 LSA 在選取節點時，必須從暫存標籤中選擇節點成本值最小者，這對整個

演算時間影響甚大，若節點數增多，運算效率將會降低。相較之下，LCA 選取節點採 FIFO 方式，選取節點方式簡單很多。

2.4 Dijkstra's Algorithm 與 A* Algorithm

2.4.1 Dijkstra's Algorithm

Dijkstra's Algorithm 是標記設定法(LSA)的一種，目前求解一對一最短路徑問題多以 Dijkstra's Algorithm 為主要理論基礎[1]，時間複雜度為$O(n^2)$，需要在路網節線成本為非負的情況下方能使用。

Dijkstra's Algorithm 的演算過程：把圖形中所有節點分成兩個集合S和Q：集合 S 放「被拜訪過的集合」，初始狀態為空集合；集合 Q 放「尚未被拜訪過的集合 」,初始狀態存放所有節點集合;所有節點 i 的上游節點$P_{(i)} = NULL$;起點 O 的成本標籤$L_{(O)} = 0$；其他節點的暫時標籤先設為無窮大$L_{(i)} = \infty$。每次從集合 Q 中選擇暫時標籤$L_{(i)}$最小的節點 i,把此節點從集合 Q 移到集合 S，並且更新所有與此節點 i 相連的暫時標籤。

每次計算均會有一個節點從暫時標籤變成永久標籤，直到所有節點都從集合 Q 轉移到集合 S，代表每條節線(i,j)皆已拓展。拓展的概念是：如果存在一條從節點 i 到節點 j 的節線，那麼從起點 O 到節點 j 存在一條路徑，該路徑經由起點 O 到節點 i 的最短路徑與節線(i,j)，路徑成本為$L_{(i)} + w(i,j)$，如果小於目前$L_{(j)}$值，表示由起點 O 經由節線(i,j)連至 j 點的路徑優於原本路徑，則以$L_{(i)} + w(i,j)$取代原本$L_{(j)}$。演算法結束時，$L_{(j)}$儲存的便是從起點 O 到節點 j 的最短路徑；如果路徑不存在，$L_{(j)}$為無窮大。每次演算時都會將最靠近起點且尚未被拜訪過的節點更改為永久標籤，並且更新與該節點相連接節點的暫存標籤。

以下為 Dijkstra's Algorithm 的演算步驟：

初始化：

Step 1： 把圖形中所有節點分成兩個集合 S 和 Q：集合 S 放「已被拜訪過的點集合」，初始狀態為空集合；集合 Q 放「尚未被拜訪過的點集合」，初始狀態存放所有節點集合。

改變標記：

Step 2： 變更起點 O 為永久標籤，自集合 Q 中移出至集合 S。設定起點 O 之 $P_{(O)} = NULL$。設定起點 O 的路段旅行成本 $L_{(O)} = 0$，令 $i = O$；其他所有節點 j 的路段旅行成本 $L_{(j)} = \infty$。$P_{(i)}$ 為節點 i 的上游節點，$\Gamma(i)$ 為所有連接節點 i 的集合。

Step 3： 更新 $\Gamma(i)$ 中所有為暫時標籤的節點之 $L_{(j)} = min[L_{(j)}, L_{(i)} + w(i,j)]$，若 $L_{(i)} + w(i,j) < L_{(j)}$，則 $P_{(j)} = i$。

Step 4： 自集合 Q 中選擇 $L_{(j)}$ 最小之節點 j。

Step 5： 令節點 j 為永久標籤，自集合 Q 中移至集合 S 且令 $i = j$。

Step 6： 若 $i = D$，即為起點 O 至訖點 D 之最短路徑，$L_{(D)}$ 為最小旅行成本；若 $i \neq D$，則回到 *Step 2* 繼續演算。

圖 2-5 為 Dijkstra's Algorithm 流程圖：

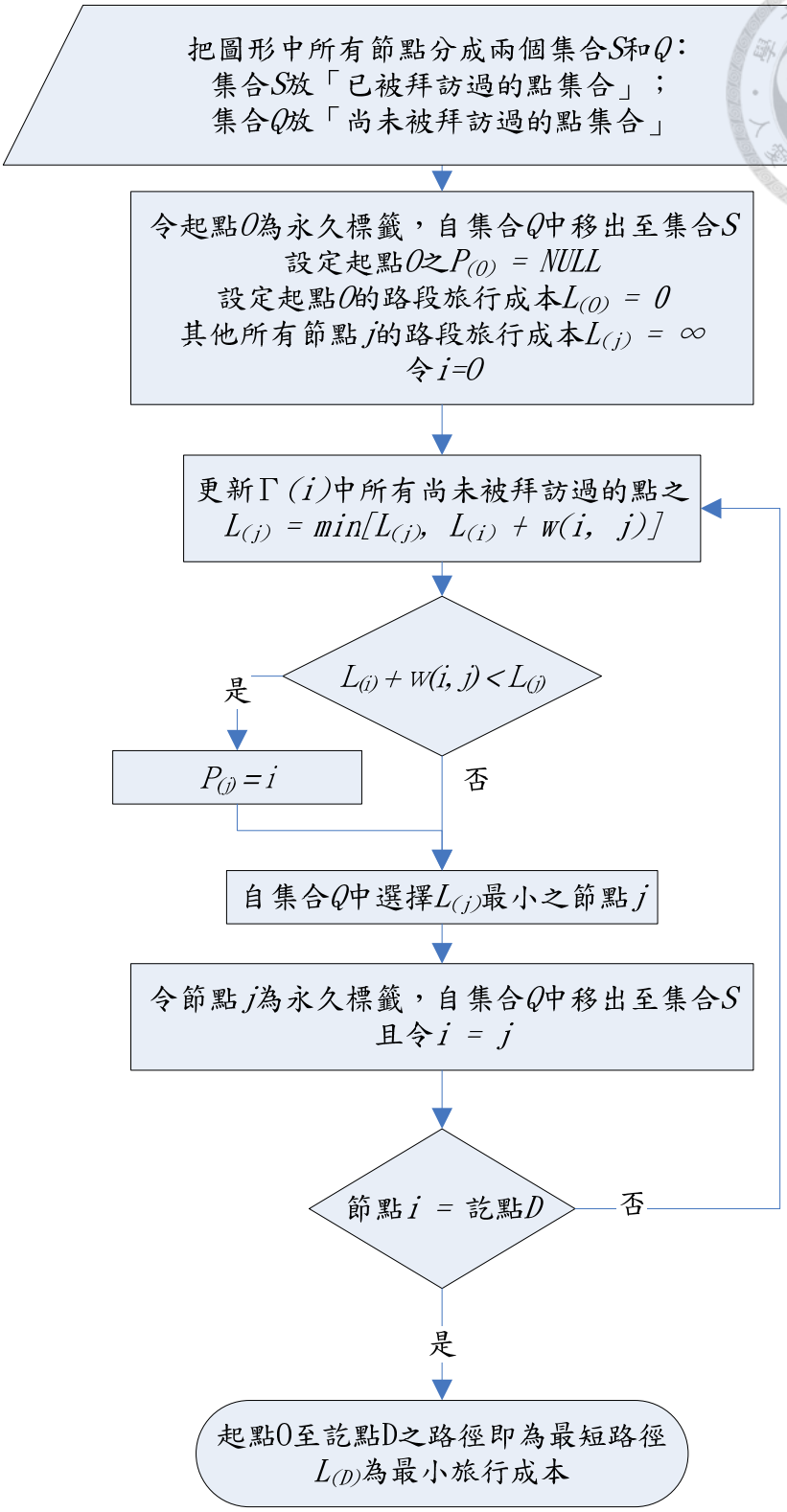
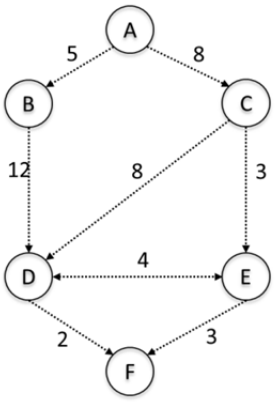


圖 2-5 Dijkstra's Algorithm 流程圖

2.4.2 Dijkstra's Algorithm 範例

假設路網起點為 A，訖點為 F，路段旅行成本如圖所示：

初始化：

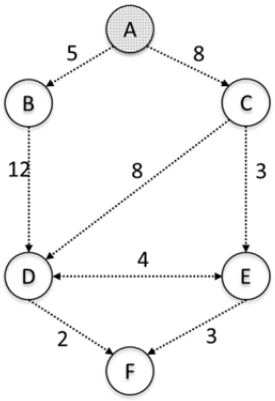


集合Q

節點	旅行成本	上游節點	拜訪與否
A	0	NULL	FALSE
B	∞	NULL	FALSE
C	∞	NULL	FALSE
D	∞	NULL	FALSE
E	∞	NULL	FALSE
F	∞	NULL	FALSE

集合S

節點	旅行成本	上游節點	拜訪與否
--	--	--	--



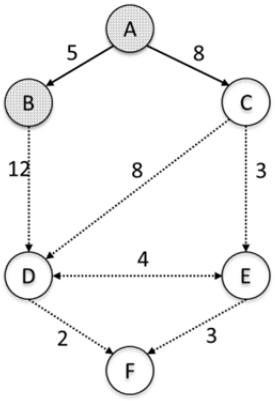
集合Q

節點	旅行成本	上游節點	拜訪與否
B	∞	NULL	FALSE
C	∞	NULL	FALSE
D	∞	NULL	FALSE
E	∞	NULL	FALSE
F	∞	NULL	FALSE

集合S

節點	旅行成本	上游節點	拜訪與否
A	0	NULL	TURE

擴展節線(A, B)



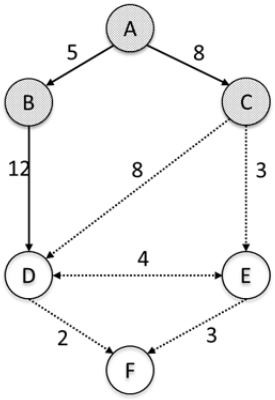
集合Q

節點	旅行成本	上游節點	拜訪與否
C	8	A	FALSE
D	∞	NULL	FALSE
E	∞	NULL	FALSE
F	∞	NULL	FALSE

集合S

節點	旅行成本	上游節點	拜訪與否
A	0	NULL	TURE
B	5	A	TURE

擴展節線(A, C)



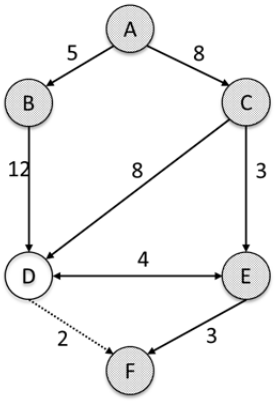
集合Q

節點	旅行成本	上游節點	拜訪與否
D	17	B	FALSE
E	∞	NULL	FALSE
F	∞	NULL	FALSE

集合S

節點	旅行成本	上游節點	拜訪與否
A	0	NULL	TURE
B	5	A	TURE
C	8	A	TRUE

重複以上步驟，值到遍歷所有節點或抵達訖點，搜尋結束，其結果如下：



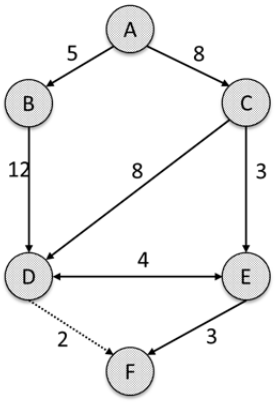
集合Q

節點	旅行成本	上游節點	拜訪與否
D	15	E	FALSE

集合S

節點	旅行成本	上游節點	拜訪與否
A	0	NULL	TURE
B	5	A	TURE
C	8	A	TRUE
E	11	C	TRUE
F	14	E	TRUE

由於已經抵達訖點F，即使節點D仍未被加入集合S，仍可宣告演算結束。若欲遍歷所有節點，則演算結果如下：



集合Q

節點	旅行成本	上游節點	拜訪與否
--	--	--	--

集合S

節點	旅行成本	上游節點	拜訪與否
A	0	NULL	TURE
B	5	A	TURE
C	8	A	TRUE
D	15	E	TRUE
E	11	C	TRUE
F	14	E	TRUE

2.4.3 Dijkstra's Algorithm 評析

Dijkstra's Algorithm 雖能求得最佳解，但是當路網規模龐大時，往往會浪費許多時間在搜尋不必要的方向，例如：訖點位於起點之東南方，Dijkstra's Algorithm 會由起點開始搜尋四面八方的所有可能路徑，包括西北方或其他方向。然而 Dijkstra's Algorithm 的優點是不需要遍歷所有節點便可以搜尋到最短路徑，若最短路徑已經搜尋到訖點，其他通往訖點的路徑旅行時間必然大於此條路徑，而此條最短路徑的子路徑(sub-path)也必然是最短路徑。

2.4.4 A* Algorithm

Klunder and Post 研究 Dijkstra's Algorithm 和 6 種其他 LCA，以不同搜尋方式(A* Algorithm 和 preprocessing "landmarks")、單雙向搜尋指標和資料結構，衍生出 168 種版本。實作結果發現基於 bucket of queues 的資料結構結合雙向搜尋在 A* Algorithm 方式是非常有效率的演算法[7]。A* Algorithm 可有效縮小搜尋範圍，使得搜尋方向能朝訖點前進，如圖 2-6 所示。

雖 A* Algorithm 可有效縮小搜尋範圍，但數學函式要設計得當，否則不易縮小搜尋範圍。除數學函式外，A* Algorithm 多利用 Euclidean 距離算出中

繼點與訖點的相對直線距離，以此作為選擇下一節點的依據，然地理上的直線距離並無法反應現實交通狀況。

圖 2-6 Dijkstra's Algorithm(左)與 A* Algorithm(右)搜尋範圍比較

2.5 依時性最短路徑問題

2.5.1 依時性最短路徑問題

現實路網的交通情況瞬息萬變，前五分鐘還暢行無阻的路段後五分鐘可能已經開始阻塞回堵，路段的旅行時間也會隨著時間推移而不斷改變。因此路段旅行時間並非定值，而是會隨著時間變動的依時性路段旅行時間。依時性最短路徑問題應用在許多領域，如：動態交通指派(DTA, dynamic traffic assignment)、網路控制(network control)、自動駕駛導引系統(automobile driver guidance)船線規劃(ship routing)、航班派遣(airplane dispatching)。

Chabini 依特性區分依時性最短路徑問題[23]：

(1). 最快路徑問題與最少成本(最短路徑問題)；

(2). 最短路徑問題的型態：一對多在特定出發時間或所有出發時間的最短路徑問題、多對一在所有出發時間的最短路徑問題；

(3). FIFO 路網與 non-FIFO 路網(FIFO 條件：$\forall(i,j,t), t + d_{ij}(t) \leq (t+1) + d_{ij}(t+1)$，駛離路段時間為遞增函數；non-FIFO 條件：晚出發者會比早出發者早到或同時到)；

(4). 路徑上是否允許停等；

(5). 離散型與連續型時間

(6). 路段旅行成本為整數或實際值。

由上述第(5)、(6)項時間處理方式，依時性最短路徑問題依時間處理方式分成兩類：離散型時間與連續型時間。離散型動態路網，可利用時空擴張圖(time-space expansion)表示成靜態路網，路段旅行時間以整數表示，稱離散型依時性路段旅行時間，如圖 2-7，橫軸為時間軸 T，縱軸為路段旅行時間t_a，a 為路段；連續型動態路網，路段旅行時間以實際值表示，稱連續型依時性路段旅行時間，如圖 2-8。雖說交通狀況瞬息萬變，但過多的資訊反而顯得繁冗且無用，若以每 5 秒鐘為一個時間間隔，則 2 小時的路段旅行時間資料就有 1440 筆資料，每天同一個路段就會產生 17280 筆資料，對於資料庫是非常大的負擔。因此交通資訊大多是每隔幾分鐘傳輸資料，為離散型路段旅行時間。

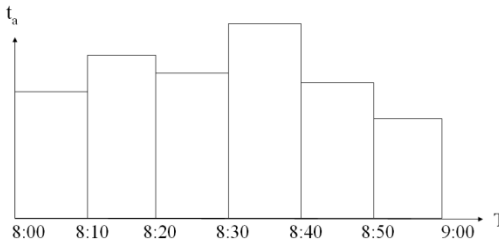


圖 2-7 離散型路段旅行時間

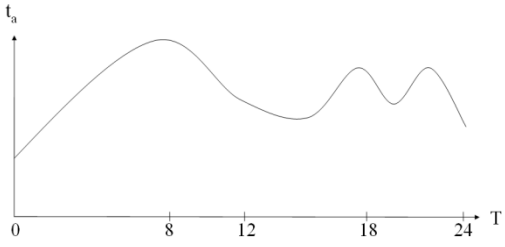


圖 2-8 連續型路段旅行時間

相對於靜態最短路徑問題的研究，動態最短路徑問題的研究較少，主要原因是靜態路徑演算法可在不多花時間的情況下解決動態路網一對多最快路徑問題，而路段旅行時間須滿足 FIFO 條件。其他原因如下[24]：

1. 最快路徑問題的演算法不適用最少成本路徑問題。以油耗舉例，最快到達路徑不一定就是最省油的路徑；
2. 交通應用上，通常是多點到某特定幾個訖點，求解時會利用靜態最短路徑演算法計算多點至多點的最快路徑，從中擷取答案。訖點個數通

常僅占所有節點的一小部分，因此靜態最短路徑演算法有可能不會得到最佳解；

3. 路段旅行時間不一定都符合 FIFO 條件；
4. 靜態最短路徑演算法僅計算特定時間區間內出發的最短路徑；然而一般情況是要考慮所有可能出發區間的最短路徑。

2.5.2 依時性最短路徑演算法

依時性最短路徑演算法最早由 Cooke and Halsey 研究[25]，從各離散出發時段$\{t_0, t_0+\delta, t_0+2\delta, \dots, t_0+M\delta\}$，以 Bellman's principle of optimality[26]發展出遞迴函數，計算路網中每一節點至訖點的依時性最短路徑。若任一節點至訖點的抵達時間超過$t_0+M\delta$，則假設$t > t_0+M\delta$時段的旅行時間為無限大。此演算法以式(2-1)遞迴至無法改善旅行時間為止[16]。

$$f_D(t) = 0$$

$$f_i(t) = \min_{j \neq i} \left(f_i(t), \bar{g}_{ij}(t) \right) + f_j\left[t + \bar{g}_{ij}(t)\right] \; for \; i = 1, 2, \dots, n-1 \qquad (2\text{-}1)$$

其中，$g_{ij}(t)$為路段(i,j)於時間 t 的旅行時間。

$$\bar{g}_{ij}(t) = \begin{cases} g_{ij}(t), if \; t + g_{ij}(t) \leq t_0 + M\delta \\ \infty, if \; t + g_{ij}(t) > t_0 + M\delta \end{cases}$$

$f_i(t)$：時間 t 由 i 訖點 D 路徑的總旅行時間；無法到達訖點 D 時，為無限大。

2.5.3 依時性前推式最短路徑演算法

f_j表示從起點O在時間區間0出發到節點j的最少路徑旅行時間。路段(i,j)的路段旅行時間取決於到達節點 i 的時間f_j，必須選擇$t \geq f_i$的時間區間，才會是抵達節點 j 的最佳路徑旅行時間f_j。$B(j)$表所有連接到節點 j 路段的節點集合。最少路徑旅行時間定義如下：

$$f_j = \begin{cases} \min\limits_{i \in B(j)} \min\limits_{t \geq f_i} \left(t + d_{ij}(t)\right); j \neq O \\ 0 \qquad\qquad\qquad\qquad\quad ; j = 0 \end{cases}$$

若 FIFO 條件滿足，求解最快路徑問題在動態路網下的時間複雜度與求解相關靜態最短路徑問題型態的時間複雜度相同。而最快路徑問題等同於以下函式：

$$f_j = \begin{cases} \min\limits_{i \in B(j)} \left(t + d_{ij}(t)\right); j \neq O \\ 0 \qquad\qquad\qquad\quad ; j = 0 \end{cases}$$

2.6 小結

空間上最短距離路徑不代表時間上最快路徑，尤其都會區通勤旅次時間彈性小，時間常是駕駛者選擇路徑的主要依據，在路徑規劃上應以旅行時間為考量。行前即時路徑導引時，要避免提供過時資訊，又要提供最佳路徑，設計具效率且考量動態資訊的依時性最短路徑演算法進行求解是必要的。最短路徑問題多為定值，然而車流動態特性會使不同時間點有不同路段旅行時間，應透過路側設備蒐集即時交通資訊，讓路段旅行時間為依時性，應用上更具合理性。由於網路傳輸速度和實際資料取得，路段旅行時間都多為離散型資料。

在路徑規劃演算法上，已知出發時間，預測可行的抵達時間和建議路徑，屬於「依時性前推式(forward)最短路徑問題」類型；然而在某些情況下，已知抵達時間，預測可行的出發時間和路徑規劃較符合實際需要，屬於「依時性後推式(backward)最短路徑問題」類型。

由於過往交通資料蒐集方式，依時性前推式最短路徑問題或最短路徑問題的研究，皆較依時性後推式最短路徑問題研究探討得多，尤其在尖峰、擁擠時間情況下，依時性前推式最短路徑問題較易從各個路段旅行時間曲線找到相對應的路段旅行時間，因此鮮少文獻探討依時性後推式最短路徑問題，

但依時性後推式最短路徑問題，如：上班或搭乘固定班次的大眾運輸工具等抵達時間確定的情況下，比較前推式與後推式最短路徑演算法，後推式最短路徑旅行時間可能會比前推式路徑旅行時間的等待時間損失來得小，使駕駛者能在預計抵達時間抵達目的地，因此有研究必要性，搭配上先進交通資料蒐集到的即時資訊，應能讓小汽車駕駛者獲得更多效益。

有關依時性最短路徑問題大多以 LCA 和 LSA 作為路徑規劃基礎，依不同目的加以修改。本研究目的即是在抵達時間已知下，修正 Dijkstra's algorithm，建構「依時性後推式最快路徑演算法」，其演算法是自訖點往起點後推得建議路徑及預測可行的出發時間。

第三章　研究方法

本研究所開發之實作系統中，係以我國高速公路路網作為研究對象，並透過真實車輛偵測器之時空資料，打造出依時性 A*路徑演算系統平台。為達成此目標，3.1 節將介紹本研究所構建之時間成本資料庫，透過解析國道一號與三號車輛偵測器之點速率資料，搭配電子收費系統之探偵車隊空間速率資料交互融合，提供路徑演算過程所需之各路徑時間成本；3.2 節將說明在獲得時間成本資料下，系統採用 A*路徑演算法之演算程序，該演算法具備獨特的啟發式向量成本特性，能使演算過程快速獲得區域次佳解。有別於傳統求解目標之全域最佳解中，A*演算法的操作特性能更加貼近即時用路人之系統需求；而 3.3 節再繼承 3.1 節與 3.2 節開發成果，說明本研究如何打造出「依時性前推式路徑演算」以及「後推式路徑演算」2 種功能，讓使用者可依據個別旅次需求，自行查詢預計出發時間或預期抵達時間；然而，依時性後推式路徑演算功能必須構建於連續時間成本資料下方能進行，因此 3.4 節將說明本研究如何將國道車輛偵測器之離散資料轉換成連續函數，並克服實務上前推式演算與後推式演算的逆轉換困境，其詳細說明內容如下所述。

3.1 依時性資料庫與資料融合

本研究係利用路段旅行時間取代空間上的路段旅行距離作為路段成本進行路徑選擇，因此需要從旅行時間資料庫取得路段旅行時間資料作為路段旅行成本。本研究的路網範圍為國道高速公路一號與國道高速公路三號，旅行時間資料庫的資料來源為國道高速公路上之車輛偵測器(Vehicle Detector, VD)提供的車輛速率資料與電子收費系統(Electrical Toll Collection, ETC)的扣款資料。VD 的優點是偵測器遍布整條國道，可以提供大量且密集的資料，然而 VD 的故障率偏高，回傳的資料經常為異常值或資料漏失；ETC 提供的資料準

確度較高，但是 ETC 目前只裝設在各個收費站的 ETC 收費車道，資料相對較少，且收費站與收費站間距離較長，往往橫跨數個交流道，使得 ETC 提供的皆為長程旅行資料，不利以交流道作為路段節點的本系統。

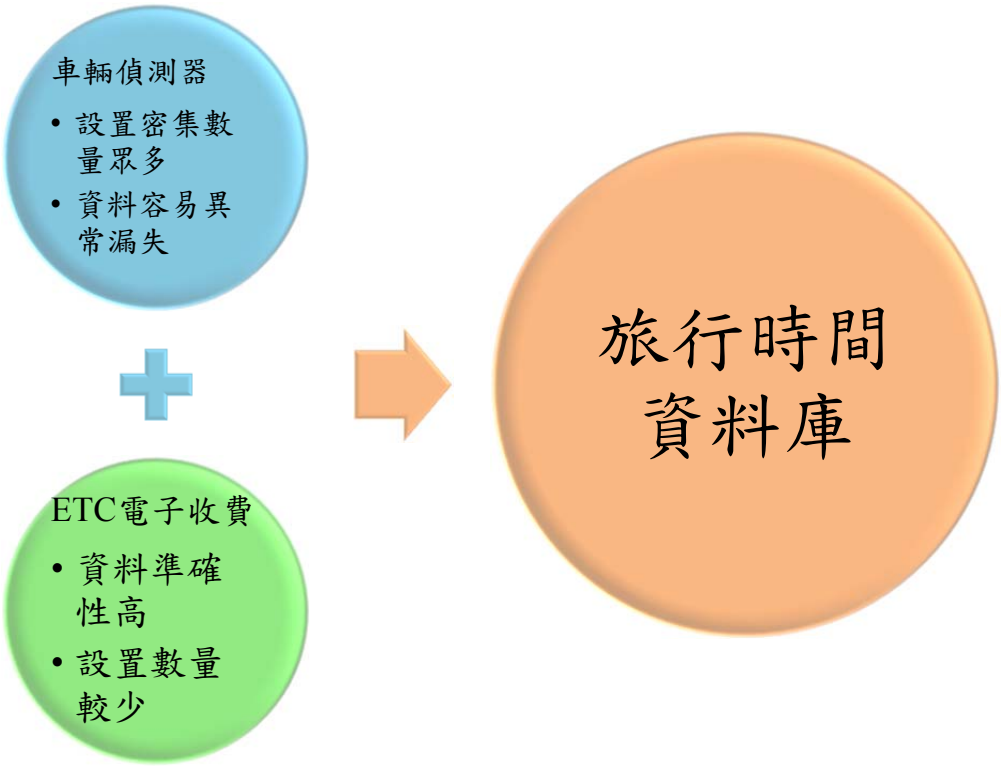


圖 3-1 旅行時間資料庫資料來源

另外，兩種資料來源提供的資料形式亦有差異，VD 測得的資料為車輛通過偵測器時的瞬時速率，是為點資料(Spot data)；而 ETC 所回傳的則是以同一輛車通過兩個收費站的時間所推算的路段平均速率，為空間資料(Space data)。為了彌補兩者資料的不足，需要進行資料插補與資料融合，整合兩種資料的優點：ETC 資料在長距離的旅行時間計算十分可靠，VD 資料則適合用來將國道區分為容易進行演算的較小區段。

3.2 A* Algorithm

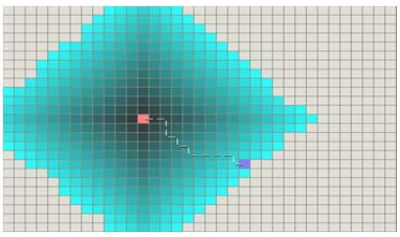
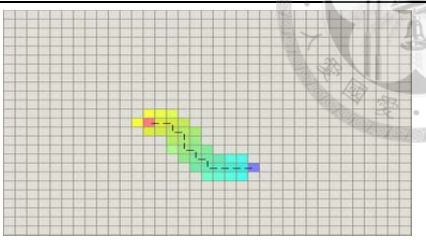
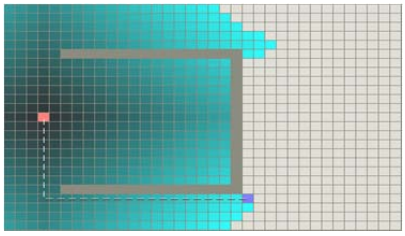
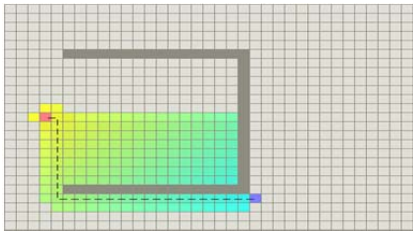
3.2.1 A* Algorithm 運作原理

基於標記修正法的路徑演算法，在結束演算以前，所有節點皆仍為暫時標籤，往往為了遍尋所有節點而耗費了許多時間。基於標記設定法的 Dijkstra Algorithm 優點是一旦搜尋到訖點，即可確認此路徑為到達訖點之最短路徑。

然而 Dijkstra's Algorithm 在搜尋下一節點並無特定方向性，只會從永久標籤中漫無目的的搜尋。當路網龐大時，Dijkstra's Algorithm 更容易浪費許多時間搜尋可能性較小的方向。

具備啟發式設計潛力的 A* Algorithm 是目前遊戲軟體中處理最短路徑問題的主流技術。A* Algorithm 透過一套特殊的啟發式評價(Heuristic Estimate)公式將許多明顯為壞的路徑排除考慮，目的在於限制節點搜尋的方向，以減少搜尋不必要節點的數量，進而減少總搜尋時間。相較於作業研究與交通工程中知名的 Dijkstra algorithm 中，A* algorithm 的搜尋深度並不以全域搜尋的最佳解作為輸出，而是採用適當的成本函數設計來尋求更具效率的次佳解，學理上 A* algorithm 屬於 Dijkstra algorithm 的延伸與改良型。其搜尋範圍在不同的操作環境狀況可透過表 3-1 進行觀察：

表 3-1 Dijkstra's Algorithm 與 A* Algorithm 搜尋狀態比較

	Dijkstra's Algorithm	A* Algorithm
開放空間		
障礙空間		

從表 3-1 可以發現，A* Algorithm 的搜尋深度並不如 Dijkstra Algorithm 達到全域搜尋，然而 A* Algorithm 的搜尋速度明顯較具效率且搜尋結果也令人滿意。縱然 A* Algorithm 在少數機會下有可能無法找到全域最佳解，但是能夠在相對較短的時間內找到合理的次佳解，當路網節點數量龐大時，A*演算法將能夠更高效率的提供搜尋結果。

A* Algorithm 的評價公式如式(3-1)：

$$f(n) = g(n) + h(n) \tag{3-1}$$

$g(n)$：從起點到目前節點 n 的距離

$h(n)$：預測目前節點 n 到訖點的距離

$f(n)$：目前節點 n 的評價分數

舉例一路網如圖 3-2，起點為 O 訖點為 D。若以 Dijkstra's Algorithm 運算，從起點 O 到節點 A 的路段成本為 3，小於從起點 O 到節點 B 的路段成本 5，因此在搜尋 O 點的下游節點時，Dijkstra's Algorithm 會優先加入節點 A；A* Algorithm 不只考慮路段成本，也參考所有候選節點與訖點的距離作為搜尋下

游節點的考量，以確保搜尋方向會指向訖點。因此雖然 B 點的路段成本高於 A 點，但是考量到與訖點 D 的距離以後，B 點的 A*評價分數優於 A 點，因此 A* Algorithm 會優先將節點 B 加入下游節點，如表 3-2。

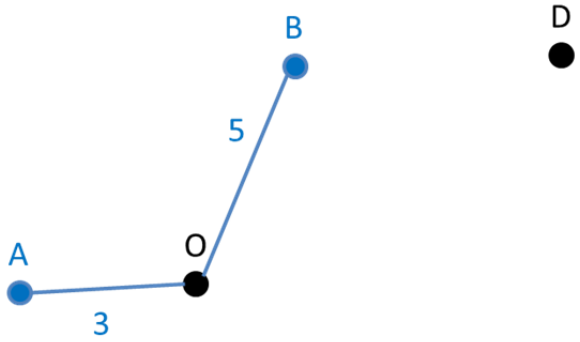
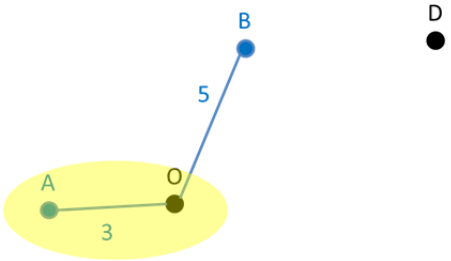
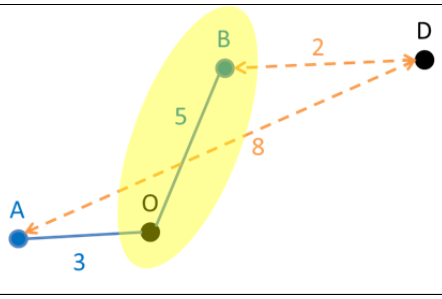


圖 3-2 A* Algorithm 範例路網

表 3-2 Dijkstra's Algorithm 與 A* Algorithm 路徑選擇比較範例

Dijkstra's Algorithm	A* Algorithm
	

其中，*h(n)*主導著 A* Algorithm 的表現方式，有以下幾種情形：

1. $h(n) = 0$

   等同於 Dijkstra's Algorithm，並以最佳解為搜尋目標。

2. $h(n) <$ 目前節點到訖點的距離

   A* Algorithm 以最佳解為搜尋目標，*h(n)*越小，搜尋深度越深。

3. $h(n) =$ 目前節點到訖點的距離

   A* Algorithm 尋找次佳解，並且能快速找到結果。

4. $h(n) >$ 目前節點到訖點的距離

   不保證能找到最短路徑，但搜尋速度快。

3.2.2 A* Algorithm 參數設計

本研究為建置一依時性路徑演算系統，為貼近使用者需求，本研究參考上述 A*評價公式，設計幾項依時性交通參數，使演算法跳脫以往較常使用之空間距離為路段成本，改採路段旅行時間作為成本函數，讓演算法符合不同時間之變化趨勢。

依照 Dijkstra's Algorithm 的演算邏輯，演算法會自起點開始，不斷加入尚為暫時標籤且累計旅行成本最小的節點，直到找到訖點為止。然而如上所述，Dijkstra's Algorithm 的搜尋方式並無方向性，在搜尋速度上較無效率。本研究參考 A* Algorithm，在考慮路段旅行成本之外，引入下游節點與訖點之直線距離作為參數，相當於以起訖點直線距離作為直徑，起訖點之中點為圓心，優先搜尋圓內的節點，距離圓心越遠的節點，搜尋優先度越低，如圖 3-3。在搜尋下游節點時，能夠確保優先訖點方向搜尋，又不至於完全忽略其他方向節點，將可在較短的搜尋時間內找到合理的最短路徑。相關參數設計如下：

1. $g'(n)$

   以下游節點之累計旅行時間取代原本旅行距離之$g(n)$。

2. $h'(n)$

   交通路網依據 A* Algorithm 中「Euclidean distance」距離[27]作為評估類型，其啟發是距離計算公式如下：

$$h(n) = G \times \sqrt{[(node.x - goal.x)^2 + (node.y - goal.y)^2]} \qquad (3\text{-}2)$$

其中，

G：下游節點的成本項目(距離)。

node.x：下游節點的座標 X 值。

node.y：下游節點的座標 Y 值。

goal.x：訖點的座標 X 值。

goal.y：訖點的座標 Y 值。

由於所得出之啟發式距離 *h(n)*仍為空間成本項，因此將 *h(n)*除以路段行駛速度上限，產出啟發式時間成本項目$h'(n) = h(n)/\bar{V}$。

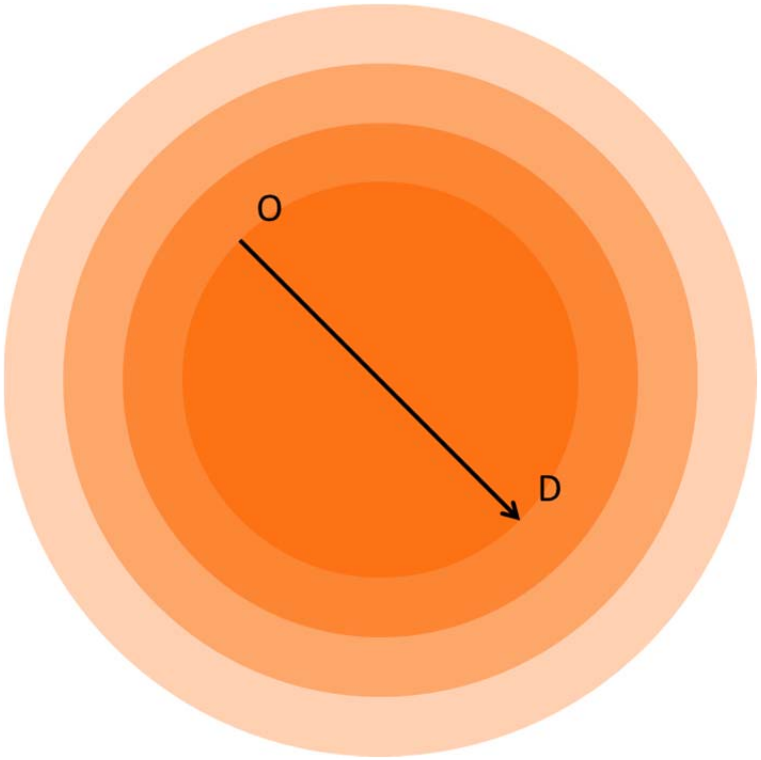


圖 3-3 A* Algorithm 搜尋優先度示意圖

3.3 前(後)推式路徑演算法原理及證明過程

在文獻回顧中，有關最短路徑問題的研究多屬於前推式最短路徑演算法，後推事最短路徑問題及後推式路徑演算法的相關研究較少，僅 Daganzo 證明在FIFO路網下，任何前推式演算法只要能求解某類型的前推式最短路徑問題，即能求解該類型的後推式最短路徑問題，反之亦然[28]。證明過程如下：

式(3-3)~(3-5)表示依時性前推式最短路徑問題：從時間$t_0(t_0 = 0)$開始，自起點 O 至訖點 D，尋找一路徑$\{D, i_n, i_{n-1}, \dots, i_2, i_1, O\}$，令式(3-3)最小

$$min\left\{E_{D,i_n}\left(E_{i_n,i_{n-1}}\left(\dots E_{i_2,i_1}\left(E_{i_1,O}(0)\right)\right)\dots\right)\right\} \quad (3\text{-}3)$$

$$for\ all\ sequence\ \{D,i_n,i_{n-1},\dots,i_2,i_1,O\}\ such\ that\ \delta_{ij,ij-1}=1 \quad (3\text{-}4)$$

$$FIFO假設條件為：E_{ij}(t)>t\ and\ E_{ij}(t)increases \quad (3\text{-}5)$$

$E_{ij}(t)$為離開時間函數，表示自時間 t 離開上游節點 j 並於時間$E_{ij}(t)$抵達下游節點 i。式(須在合理的範圍內求解，求解範圍以式(3-4)、式(3-5)規範。$\delta_{r,s}$以二元形式表示路段的連通性，$\delta_{r,s}=1$表示從節點 s 到節點 r 有路段連接。

式(3-6)~(3-8)表示依時性後推式最短路徑問題：自起點 O 出發，要在時間 $t_D(t_D=0)$抵達訖點 D，尋找一路徑$\{O,i_1,i_2,\dots,i_{n-1},i_n,D\}$，令式(3-6)最小

$$\max\left\{I_{O,i_1}\left(I_{i_1,i_2}\left(\dots I_{i_{n-1},i_n}\left(I_{i_n,D}(0)\right)\dots\right)\right)\right\} \quad (3\text{-}6)$$

$$for\ all\ sequence\ \{O,i_1,i_2,\dots,i_{n-1},i_n,D\}\ such\ that\ \Delta_{ij-1,ij}=1 \quad (3\text{-}7)$$

$$FIFO假設條件為：I_{ij}(t)<t\ and\ I_{ij}(t)\ increases \quad (3\text{-}8)$$

$I_{ij}(t)$為進入時間函數，表示自時間$I_{ij}(t)$進入上游節點 i，在時間 t 離開下游節點 j。式(3-6)須在合理的範圍內求解，求解範圍以式(3-7)、式(3-8)規範。$\Delta_{r,s}$以二元形式表示路段的連通性，$\Delta_{r,s}=1$表示從節點 r 到節點 s 有路段連接。

逆運算(reversibility)：式(3-3)~(3-5)能轉換到式(3-6)~(3-8)，反之亦然。逆運算證明如下：把所有節線轉向(即 δ變為 δ'、Δ變為Δ')，而時間的方向性亦跟著轉變(即 t 變為-t')。上述依時性最短路徑問題皆以物理方程式表示，與轉換過的時空座標圖無差別，下述為證明此論點，重新定義的離開時間函數集合為：

$$E'_{ij}(t)=-I_{ij}(-t) \quad (3\text{-}9)$$

代換(3-8)的 t 為 t'，式(3-9)代入式(3-8)，並以 E'表示：

$$E'_{ij}(t') > t' and\ E'_{ij}(t')\ increases \tag{3-10}$$

*E'*為時間離開函數，如式(3-3)。式(3-6)可以表示如下：

$$\max\{-[-I_{O,i_1}(-[-I_{i_1,i_2}(\dots - I_{i_{n-1},n}(-[-I_{i_n,D}(-0)])\dots)])]\}$$

利用式(3-9)代入，得：

$$\max\left\{-E'_{O,i_1}\left(E'_{i_1,i_2}\left(\dots E'_{i_{n-1},i_n}\left(E'_{i_n,D}(0)\right)\dots\right)\right)\right\}$$

$$= -\min\{E'_{O,i_1}\left(E'_{i_1,i_2}\left(\dots E'_{i_{n-1},i_n}\left(E'_{i_n,D}(0)\right)\dots\right)\right)\} \tag{3-11}$$

$$for\ all\ sequence\ \{O, i_1, i_2, \dots, i_{n-1}, i_n, D\}\ such\ that\ \delta'_{ij-1,ij} = 1 \tag{3-12}$$

式(3-10)~(3-12)等同於式(3-3)~(3-5)，其為前推式最短路徑：時間自 0 起點 D 出發至訖點 O 的一條路徑，得證一演算法如能求解式(3-3)~(3-5)，亦能求解式(3-10)~(3-12)，反之亦然。Daganzo 強調在依時性路網問題中，擁擠現象是不可逆的(reversible)。

3.4 時間連續性與時間間隙問題

在 3.1 節提到本研究的資料來源為國道高速公路 VD 與 ETC 資料整合所得的路段旅行時間。路段旅行時間是以單位時間內的平均速率來呈現，為離散型時間資料，每五分鐘一筆資料。路段旅行時間的精確度與資料來源的單位時間間隔(time interval)的大小有關，單位時間間隔越小，則路段旅行時間的精確度越高，兩者呈反向關係，若欲求得準確度較高的路段旅行時間，則單位時間間隔要越小，例如本來十五分鐘取得一筆路段旅行時間資料，改為五分鐘或一分鐘取得一筆資料。然而時間間隔切割越細，雖然能提升路段旅行時間的精確度，卻會使得資料庫儲存量變大，增加資料處理時間，若五分鐘記錄一次，一天一個偵測器會產生 288 筆資料；若縮短為 30 秒記錄一次，一天一個偵測器就會產生 2880 筆資料。

離散型時間資料的另一個缺點是進行後推式演算時會產生時間間隙問題。離散型後推式路徑演算法的時間間隙問題分為兩種情況：1. 路段旅行時間≦單位時間間隔；2. 路段旅行時間＞單位時間間隔，分別舉例如後述。

1. 路段旅行時間≦單位時間間隔

假設一路段 ij 的旅行時間資料如圖 3-4，預計於 8:33 離開下游節點 j，希望尋找最晚進入上游節點的時間。若於 8:30 進入上游節點 i，8:30 屬於 8:30~8:35 的範圍，路段旅行時間為 5 分鐘，則離開下游節點 j 的時間為 8:35，已經超過原先離開下游節點 j 的時間點 8:33(如圖 3-5)；若提早 1 分鐘於 8:29 進入上游節點 i，8:29 屬於 8:25~8:30 的範圍，路段旅行時間為 3 分鐘，則離開下游節點 j 的時間為 8:32，與原先預計離開下游節點 j 的時間點 8:33 存在一長度為 1 分鐘的時間間隙(time gap)(如圖 3-6)。在這段時間內，並無法找到能符合預計離開節點 j 的出發時間，只能找到使得離開路段時間最接近且尚未超過的出發時間 8:29。

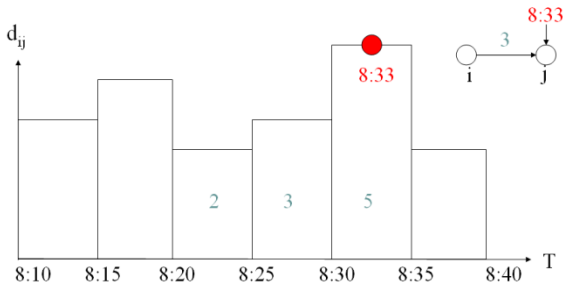


圖 3-4 路段旅行時間≦單位時間間隔之離散型路段旅行時間資料

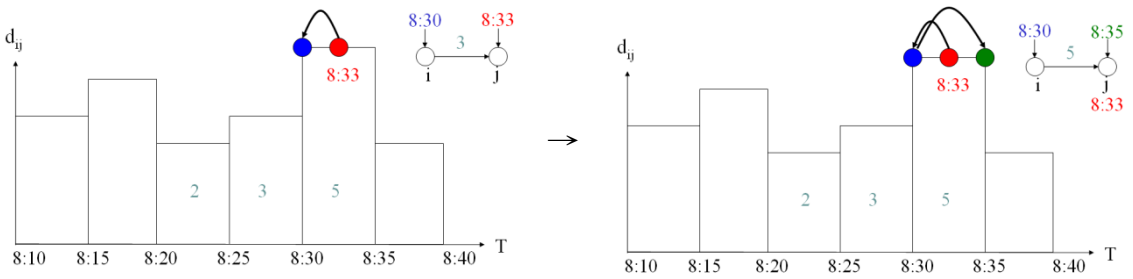


圖 3-5 路段旅行時間≦單位時間間隔之時間間隙問題

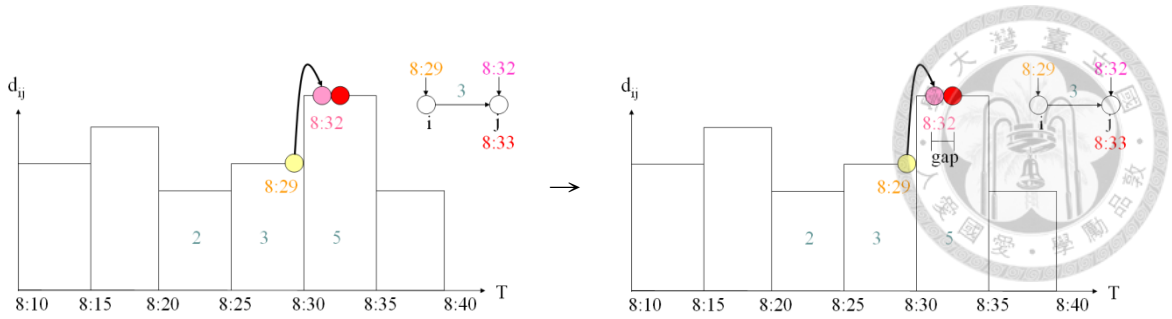


圖 3-6 路段旅行時間≦單位時間間隔之時間間隙問題

2. 路段旅行時間＞單位時間間隔

假設一路段 ij 的旅行時間資料如圖 3-7，預計於 8:33 離開下游節點 j，希望尋找最晚進入上游節點的時間。若於 8:27 進入上游節點 i，8:27 屬於 8:25~8:30 的範圍，路段旅行時間為 9 分鐘，則離開下游節點 j 的時間為 8:36，已經超過原先離開下游節點 j 的時間點 8:33(如圖 3-8)；若於上一個時階最晚的 8:24 進入上游節點 i，8:24 屬於 8:20~8:25 的範圍，路段旅行時間為 6 分鐘，則離開下游節點 j 的時間為 8:30，與原先預計離開下游節點 j 的時間點 8:33 存在一長度為 3 分鐘的時間間隙(time gap)(如圖 3-9)。在這段時間內，無論如何都無法找到能符合預計離開節點 j 的出發時間，只能找到使得離開路段時間最接近且尚未超過的出發時間 8:27。

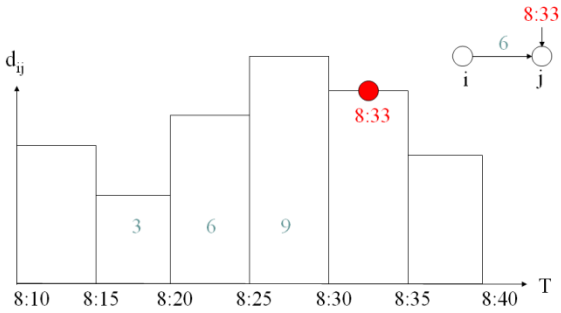


圖 3-7 路段旅行時間＞單位時間間隔之離散型路段旅行時間資料

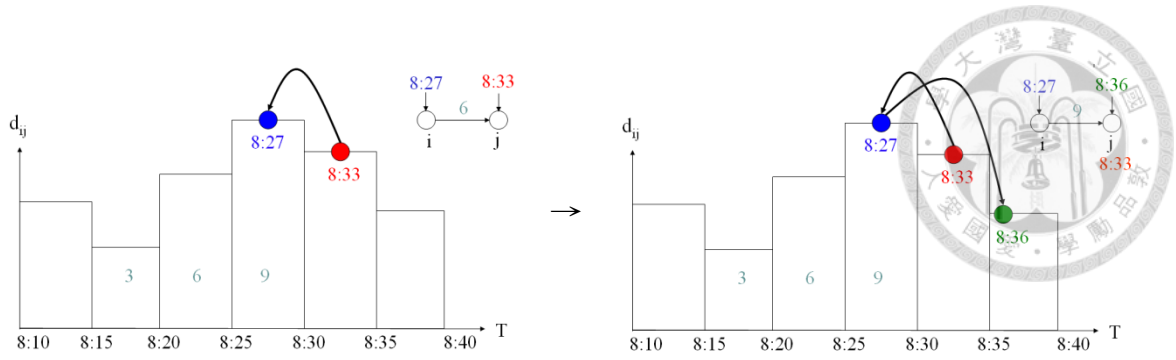


圖 3-8 路段旅行時間＞單位時間間隔之時間間隙問題

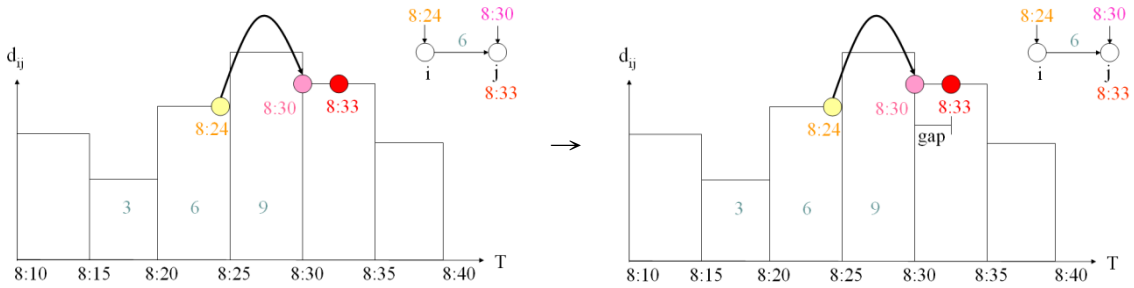


圖 3-9 路段旅行時間＞單位時間間隔之時間間隙問題

因為時間間隙關係，預測的可行出發時間可能會有偏估的情形，駕駛者會過早自起點出發，與原先期望的路徑規劃不符，如圖 3-10。時間間隙過多也可能造成抵達路段時的交通狀況與預測的有差異，使得路徑規劃失準。

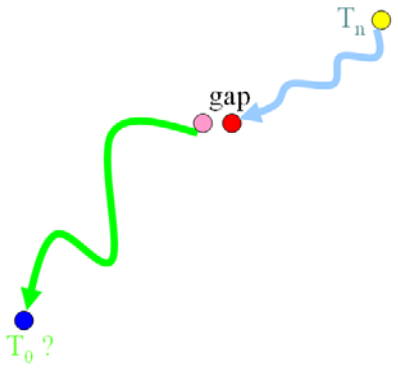


圖 3-10 路徑規劃中的時間間隙問題

離散型時間資料有上述兩項缺點(資料量龐大影響運算效率、時間間隙問題)，本研究使用離散時間傅立葉轉換法(DTFT, Discrete-time Fourier Transform)將原本的離散型時間資料轉換為連續型時間資料，將可有效解決離散型時間資料的缺點。離散時間傅立葉轉換法是傅立葉轉換的一種，以不同頻率的餘弦波函數不斷疊加，轉化為一條連續型的頻率域曲線，擬和(fit)原始離散型的

時間域資料。為避免轉換後的曲線與原始資料誤差過大，在擬和原始資料時會設定一個誤差限度，傅立葉轉換程序會不斷疊加擬和，直到轉換後的曲線與原始資料的誤差小於此限度後才會停止。從純粹的數學意義上看，傅立葉轉換是將一個函數轉換為一系列的週期函數累加來處理;而從物理意義說明，則是將信號由時間域(Time domain)轉換到頻率域(Frequency domain)，或是利用其逆轉換，將信號由頻率域轉換回時間域。

路段旅行時間資料庫原本每 5 分鐘儲存一筆旅行時間資料，每天有 288 筆旅行時間資料。經過傅立葉轉換後，每天的路段旅行時間皆能以一條傅立葉曲線表示，而且每個時間點皆能對應到一個路段旅行時間，如圖 3-11。連續型後推式路徑演算法計算路段旅行時間時，可能會發生不同時間點進入上游節點卻同時離開下游節點的情況，如圖 3-12，由於目標是找到最快路徑，在抵達時間相同的情況下，越晚進入路段代表旅行時間越短，因此當有數個可行的出發時間可供選擇時，應該選擇最晚的一個為最佳選擇。

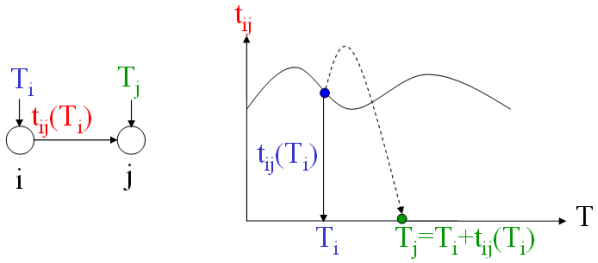


圖 3-11 連續型路段旅行時間資料

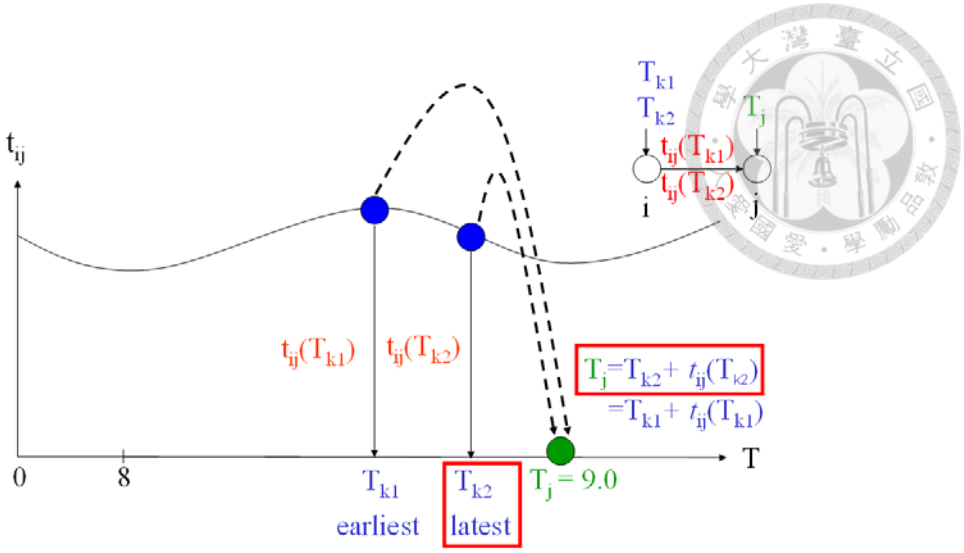


圖 3-12 多個出發時間對應同一個抵達時間

3.5 小結

本研究使用依時性 A*路徑演算法，建置一符合國道高速公路路徑選擇的依時性 A*路徑演算系統，提供使用者自旅次起點至旅次訖點的建議旅行路徑，以最短總旅行時間為目標。為達到依時性需求，融合 VD 與 ETC 資料成為路徑決策根據的旅行時間資料庫。為避免離散型後推式路徑演算所產生的時間間隙問題，採用離散時間傅立葉轉換，將原本的離散時間資料轉換為連續時間資料。最後根據使用者的需求，使用依時性前推式 A*路徑演算或依時性後推式路徑演算，提供使用者最快速的建議旅行路徑。

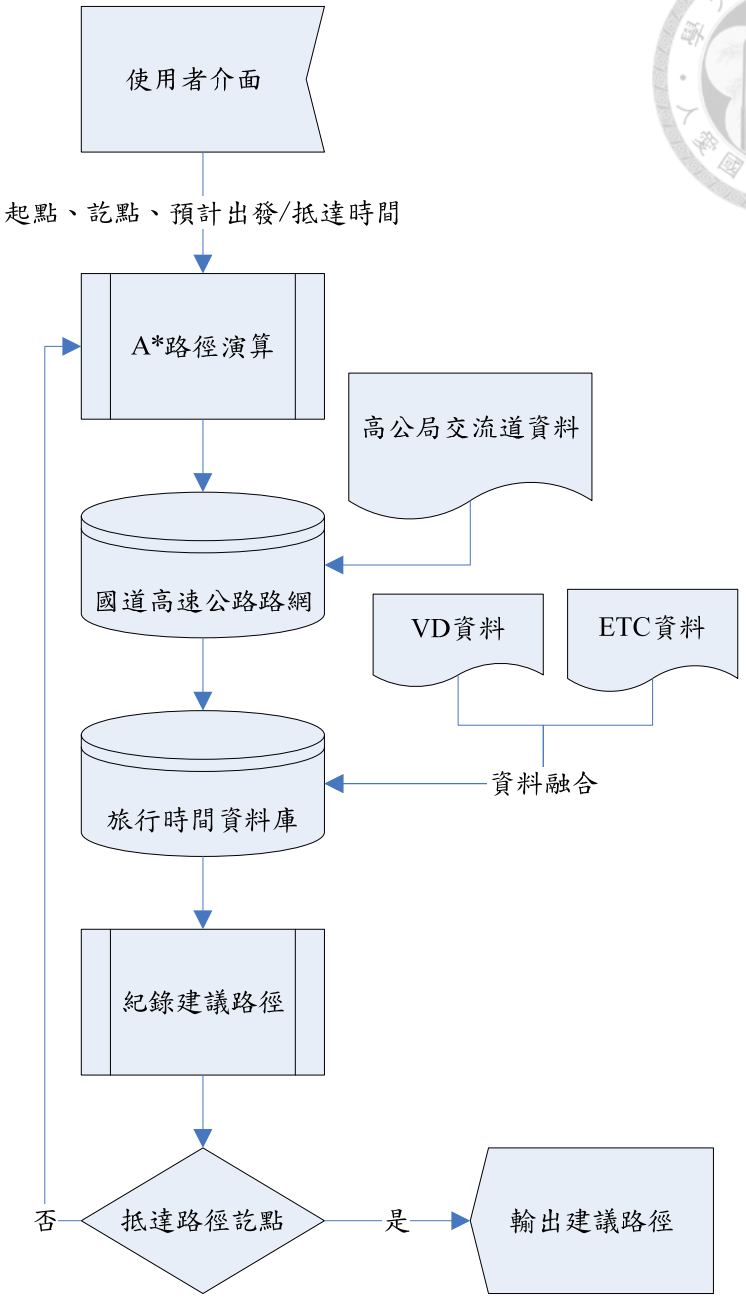

使用者介面
起點、訖點、預計出發/抵達時間
A*路徑演算
高公局交流道資料
國道高速公路路網
VD資料
ETC資料
資料融合
旅行時間資料庫
紀錄建議路徑
否
抵達路徑訖點
是
輸出建議路徑

圖 3-13 研究架構

第四章　系統建構

本章節將會介紹依時性 A*路徑演算系統之系統架構與運作邏輯，並針對配合本系統而設計的資料型態以及各項重要子程式進行詳細介紹。

4.1 系統架構與運作邏輯

本系統之基本架構可分為使用者介面、路徑演算模組、旅行時間資料庫三個部分。使用者先依照需求輸入“起點”、“訖點”、“預計出發/抵達時間”，使用者介面會將使用者輸入的資料轉換為標準格式(起點、訖點轉換為里程數，預計出發/抵達時間轉換為標準時間)，並以此為依據進行依時性最短路徑演算。

路徑演算模組接收到查詢條件後開始進行依時性最短路徑演算，由於路段旅行時間會隨著時間不同而變化，演算進行中系統會持續向旅行時間資料庫查詢路段旅行時間作為路徑成本，每加入一個下游節點即記錄旅行路徑及總旅行時間，直到抵達訖點為止。隨後，輸出已紀錄之建議旅行路徑以及總旅行時間供使用者參考。系統運作流程圖如圖 4-1。

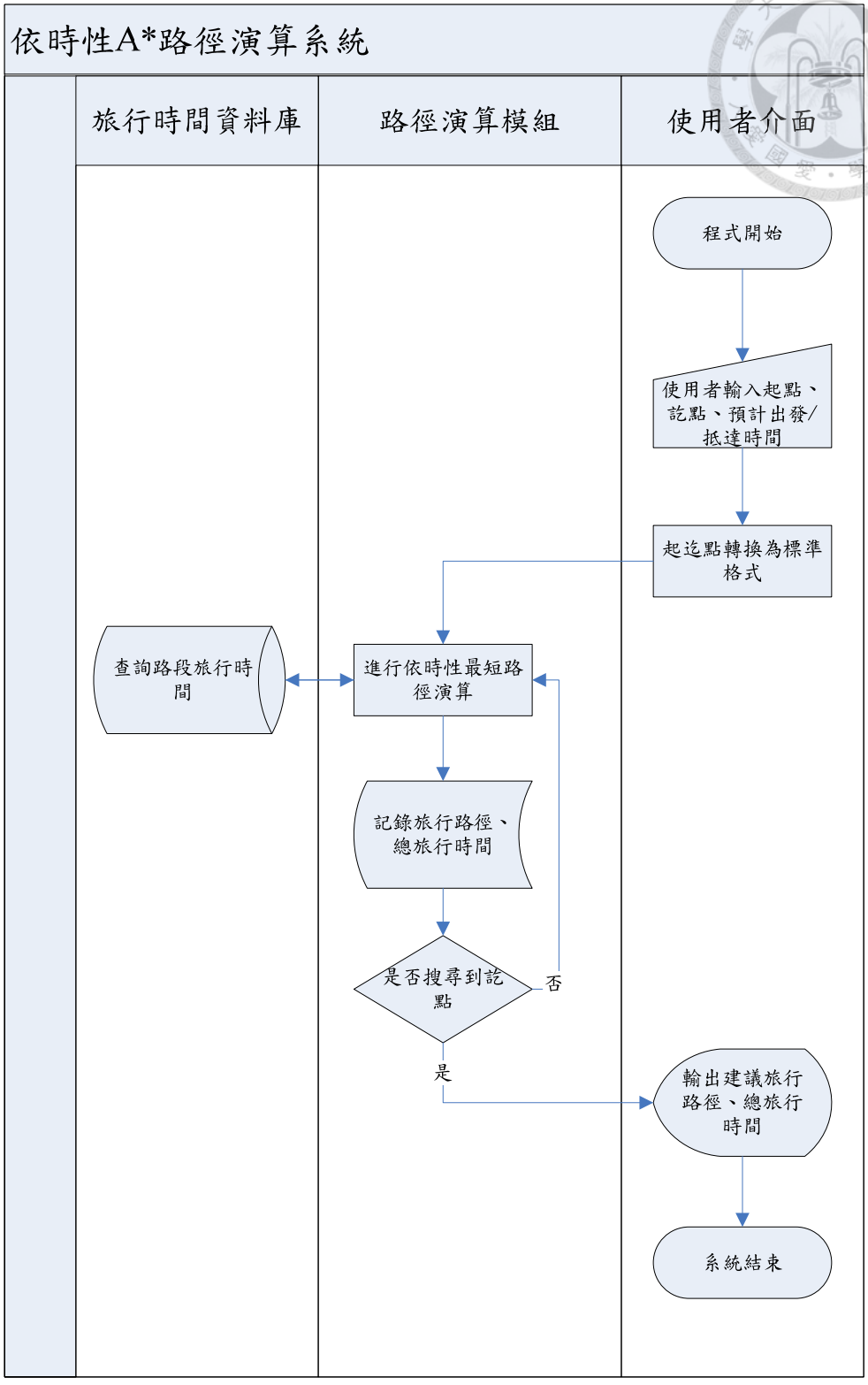


圖 4-1 依時性 A*路徑演算系統流程圖

4.2 系統建置

4.2.1 使用者介面

本系統系統介面如圖 4-2，使用者可依照需求選擇起訖點、前推式或後推式、以及預計出發時間(前推式)／預計抵達時間(後推式)。起訖點可選擇國道一號或國道三號上所有交流道、收費站或服務區。演算結果將會輸出於下放空白處。

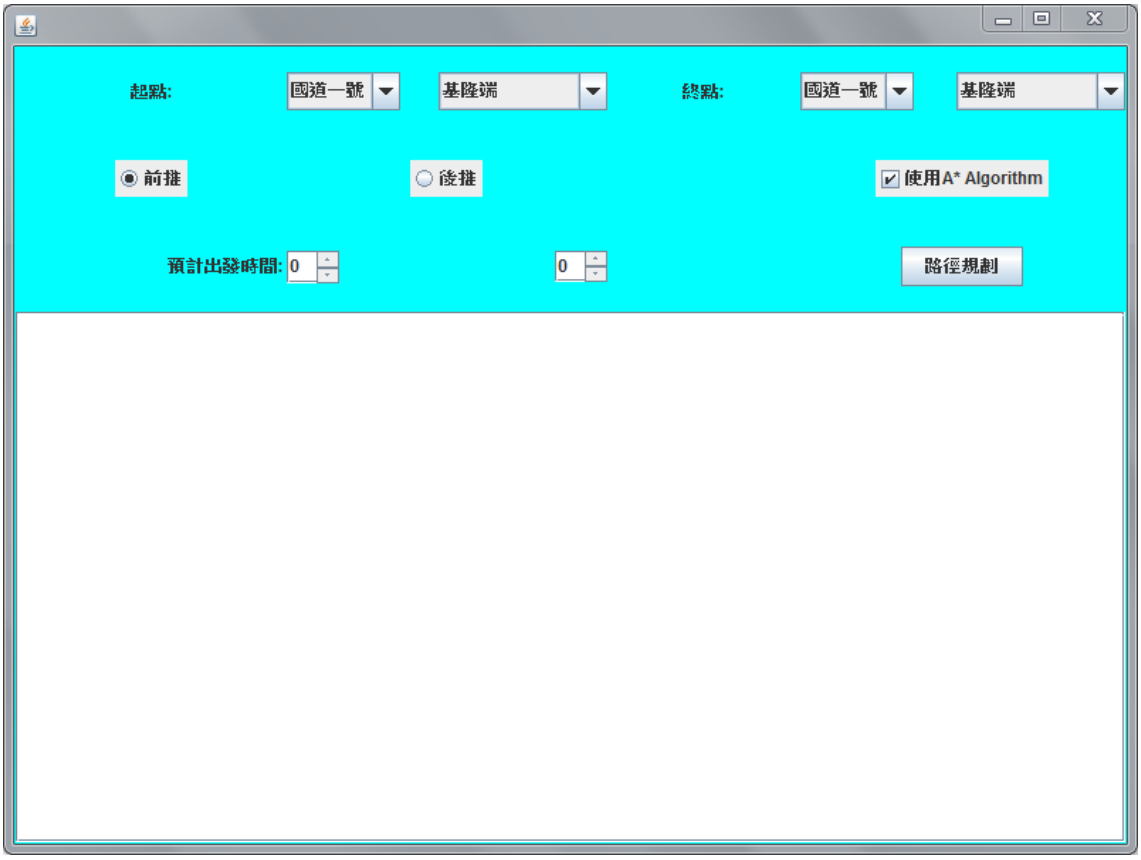


圖 4-2 依時性 A*路徑演算系統使用者介面

4.2.2 建立路網節點資料

本研究之研究範圍為國道一號與國道三號全線，路網節點設定為國道一號與國道三號上所有交流道、收費站與服務區。路網節點的參數設定為：國道編號、里程、名稱、經緯度、連結關係。路網節點之名稱與里程數資料取

自交通部台灣區國道高速公路局網頁，然高公局網頁並沒有提供節點之經緯度位置，節點之經緯度位置為手動自 Google 地圖上點取查詢。

單有節點資料並無法形成路網，需加入節點間連結關係才能成為路網。權數矩陣(weight matrix)為最簡單描述路網的資料結構；然而國道高速公路的特性使得大部分節點幾乎都只連接上游與下游節點，若以權數矩陣描述，將會形成稀疏矩陣，浪費記憶體空間並且降低運算效率。因此本系統參考超連結(Hyperlink)的方式，將節點的參數增加「連結關係」一項，負責記錄所有與此節點連結的其他節點，不僅可以節省記憶體空間，每當搜尋到一個節點時也可以立即得知此節點與其他節點的連結關係。

4.2.3 資料儲存結構

本系統為了配合程式運作與資料儲存，自創了一個名為 VDMenu 的類別(class)，VDMenu 內包含 11 個屬性(field)與 22 個方法(method)，如表 4-1、表 4-2。系統初始化時，自事先建立為 Excel 檔的讀入節點資料，每個節點儲存為一個 VDMenu。根據節點所在國道編號、節點里程、節點名稱、經度、緯度、路網節點關係，儲存為 VDMenu 中的節點資訊，節點變數填入初始值：timeWhenPass = ∞、travelTime = ∞、parent = null、hasVist = false。

表 4-1 VDMenu 屬性

		屬性 (field)	描述
VDMenu	節點資訊（一經輸入即不改變）	highwayNO (Integer)	節點所在國道編號
		millage (Integer)	節點里程
		name (String)	節點名稱
		longitude (Float)	節點經度
		latitude (Float)	節點緯度
		relationship (ArrayList)	路網連結關係
		distanceToGoal (Integer)	與訖點距離
	節點變數（隨著演算改變）	timeWhenPass (Double)	到達此節點之時間
		travelTime (Double)	累計旅行時間
		parent (VDMenu)	上游節點
		hasVist (Boolean)	是否已經被搜尋到

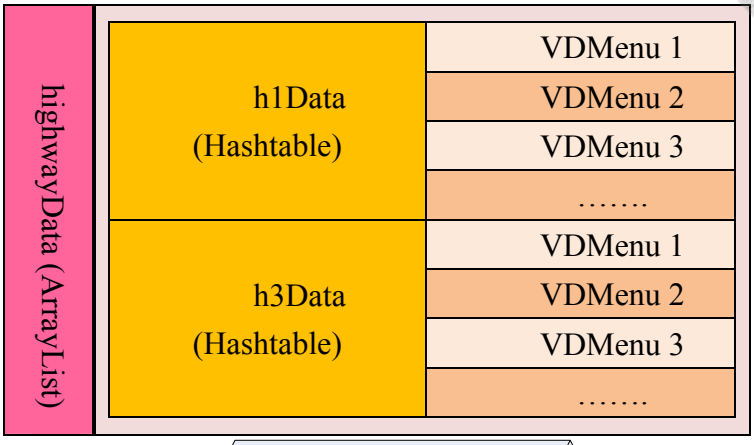
表 4-2 VDMenu 方法

	方法(method)	
VDMenu	寫入方法	讀取方法
	setHighwayNO	getHighwayNO
	setMilliage	getMilliage
	setNaame	getNaame
	setLongitude	getLongitude
	setLatitude	getLatitude
	setRelationship	getRelationship
	setDistanceToGo	getDistanceToGo
	setTimeWhenPass	getTimeWhenPass
	setTravelTime	getTravelTime
	setParent	getParent
	setHasVist	getHasVist

將所有節點儲存為 VDMenu 以後，分別依照各自所處於不同國道存入不同的 Hashtable，以每個節點的里程數作為索引值(key)，最後將所有 Hashtable 統一存入一個命名為 highwayData 的 ArrayList(表 4-3、圖 4-3)。如此一來，

我們只須要有該節點的國道編號與里程數，即可查到該節點的所有資訊，並且可以對該節點的所有參數進行編輯。

表 4-3 路網節點儲存結構



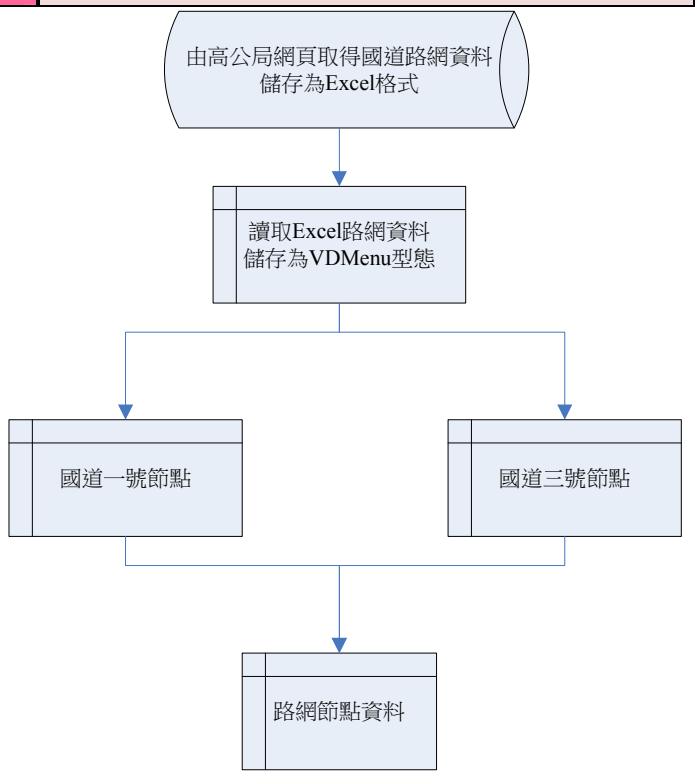


圖 4-3 路網節點資料產生流程

4.3 路徑演算模組

路徑演算模組分為前推式與後推式兩部分，前推式的輸入資料為：起點、訖點、預計出發時間，輸出資料為：建議旅行路徑、抵達時間、總旅行時間；後推式的輸入資料為：起點、訖點、預計抵達時間，輸出資料為建議旅行路

徑、出發時間、總旅行時間。以下詳細介紹前推式與後推式路徑演算之演算流程。

4.3.1 前推式路徑演算流程

前推式路徑演算的輸入資料為：起點、訖點、預計出發時間。路徑演算法開始時，將起點變更為永久標籤。搜尋所有永久標籤向外相連的節點，查詢路段旅行時間，更新其暫時標籤。從所有暫時標籤中，尋找 A*評價分數最優的標籤，將其變更為永久標籤。重複上述步驟，直到訖點成為永久標籤演算即結束。演算流程如圖 4-4。

前推式路徑演算演算步驟如下：

Step 1： 更新起點的參數：timeWhenPass = 出發時間、travelTime = 0、parent = null、hasVist = true。並將起點加入已經被拜訪過的節點集合 *S*。

Step 2： 針對最新加入集合 S 的節點 *i*，尋找與其連接且 hasVist = false 的節點 *j*。以 *i* 為路段起點，*j* 為路段終點，*i* 的 timeWhenPass 為查詢時間，向旅行時間資料庫查詢路段 *ij* 的旅行時間。若 *i* 的 timeWhenPass 加上路段 *ij* 的旅行時間小於 *j* 的 timeWhenPass，則以 *i* 作為 *j* 的上游節點，更新 *j* 的參數：timeWhenPass = *i* 的 timeWhenPass 加上 *ij* 的路段旅行時間、travelTime = *ij* 的路段旅行時間、parent = *i*。

Step 3： 將 *j* 加入備選集合 *Q*。

Step 4： 自集合 *Q* 中，選取 A*評價分數最優的節點 *k*，將 *k* 自集合 *Q* 中移除，加入集合 *S*。

Step 5： 確認訖點是否已經被加入集合 *S*。若已被加入集合 *S*，代表演算法已經搜尋到訖點，演算結束；若尚未被加入集合 *S*，則回到 *Step 2* 繼續演算。

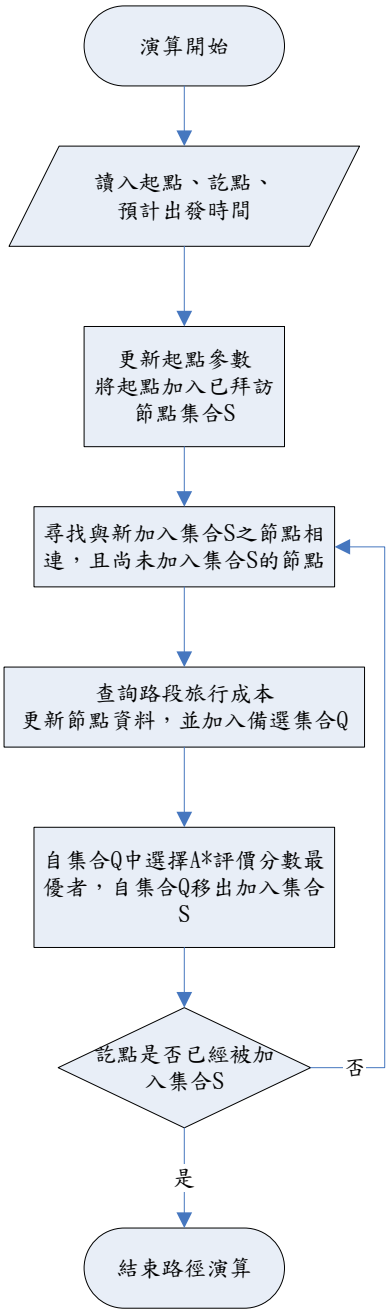


圖 4-4 前推式依時性 A*路徑演算流程圖

4.3.2 後推式路徑演算流程

後推式路徑演算流程類似於前推式路徑演算流程，後推式路徑演算的輸入資料為：起點、訖點、預計抵達時間。運算時只需將使用者輸入之起訖點

對調後，按照前推式路徑演算法流程運算即可。惟查詢路段旅行成本時與前推式路徑演算流程有所差異，以下將舉例詳述查詢路段旅行成本的方法。

設一路段，路段起點為 O，路段訖點為 D。旅行時間資料庫的查詢方法是輸入路段起點 O、路段訖點 D 以及進入路段起點 D 的時間t_O，資料庫會回傳自t_O出發，O→D 所需要的旅行時間$E_{OD}(t)$。後推式演算法進行演算時，是將使用者輸入的起訖點對調進行演算，自通過 D 的時間t_D查詢 D→O 的旅行時間。然而使用者最終的旅行方向為 O→D，即使在同一路段，順向與反向的旅行時間並不相同(圖 4-5)，因此仍需求得 O→D 的路段旅行時間作為路段旅行成本。

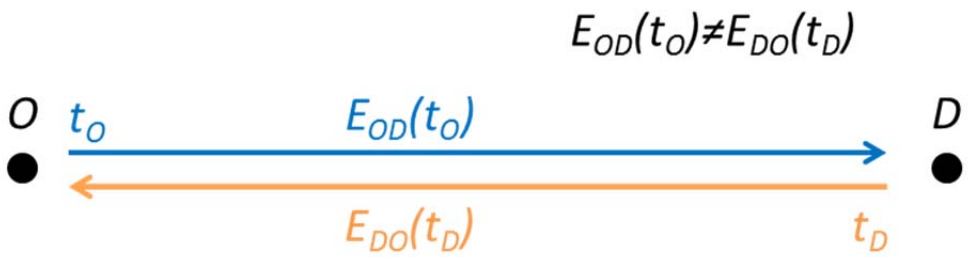


圖 4-5 後推式路段旅行時間查詢

為從 D 點的抵達時間求得 O 點的出發時間，我先假設短時間中，路段旅行時間並不會相差太多，以 O 為起點，t_D為出發時間，得路段旅行時間為$E_{OD}(t_D)$，若路段旅行時間在這段時間中並沒有改變的話，即可推得實際出發時間為$t'_O = t_D - E_{OD}(t_D)$。接著需要測試此出發時間是否能夠符合需求，再次查詢旅行時間資料庫，起點為 O，出發時間為t'_O，抵達 D 點的時間為$E_{OD}(t'_O)$。計算$\Delta t = E_{OD}(t'_O) - t_D$代表實際抵達時間與預計抵達時間的差距，$\Delta t > 0$代表實際抵達時間晚於預計抵達時間，$\Delta t < 0$代表實際抵達時間早於預計抵達時間。若實際抵達時間與預計抵達時間存在差異，則調整出發時間$t''_O = t'_O - \Delta t$再次

測試，直到實際抵達時間與預計抵達時間的差距小到可以接受為止，如圖 4-6。

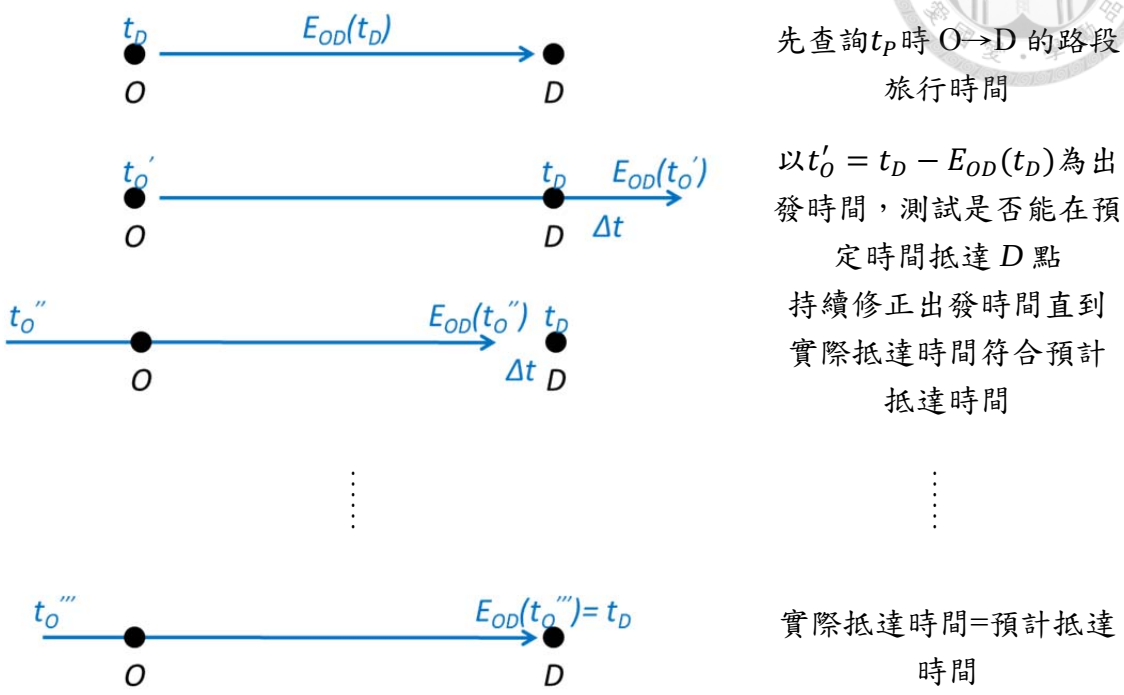


圖 4-6 後推式路段旅行時間查詢範例

4.3.3 建議路徑輸出方式

路徑演算完成後，所有被搜尋過的節點參數皆紀錄了上游節點的資料，由於所有節點最多只可能有一個上游節點，因此只要從訖點開始，層層往上游節點尋找，即可求得建議旅行路徑。後推式演算法的情況下，將起點與訖點對調後，按照相同方法從起點開始往下游節點尋找即可。輸出流程如圖 4-7。實際操作步驟如下：

Step 1： 將訖點加入一串列(List)

Step 2： 針對串列中最新的節點，搜尋上游節點，並加入串列

Step 3： 確認起點是否已加入串列。若否，回到 *Step 2*。

Step 4： 將整個串列反序排列。

Step 5： 依序輸出串列中的節點，即為從起點到訖點之建議旅行路徑。

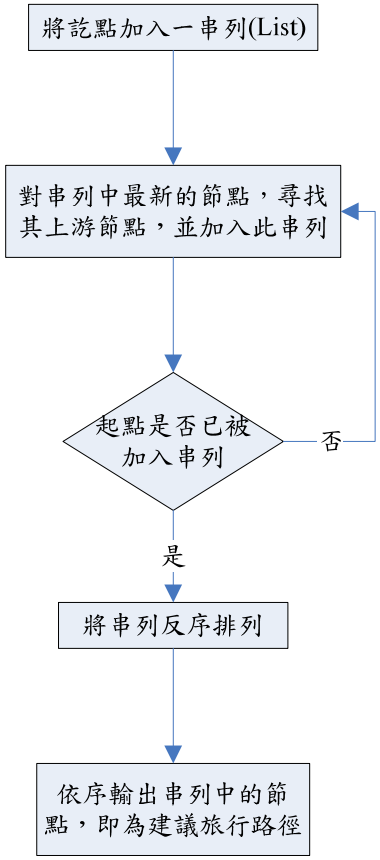


圖 4-7 建議路徑輸出流程圖

4.4 相關子程式

4.4.1 標準時間轉換

使用者介面輸入的時間為「時」、「分」制，然而無論對演算系統或資料庫來說，單純只有「時」與「分」並不夠精確，而且還需要儲存日期等變數，相當占用記憶體空間，也降低演算法運算效率。

比較常見的作法是，依照輸入的日期時間，轉換為對某個特定標準時間的差值，如此就可以以一個變數表示任意時間。JAVA 中的標準時間是 1970 年 1 月 1 日 00:00:00 GMT，可以將任何時間表是為與此標準時間的差值，精確到毫秒(ms)。

因此，當使用者於使用者介面輸入「時」、「分」時，系統會先行將「時」、「分」轉換為標準時間再儲存，之後所有的演算、查詢與儲存皆以標準時間

處理，待所有運算完成即將輸出旅行時間資料，最後再將標準時間轉換為使用者容易辨識的「時」、「分」。

4.4.2 候選路段集合

路徑演算法開始後，會自起點開始逐漸將 A*評價分數最優的節點，依序加入已被拜訪的集合 *S*，並且更新周遭其他節點的參數。然而在搜尋下一個加入集合 *S* 的節點時，並非是所有的節點都有機會被搜尋到，唯有與集合 *S* 中任一節點有相連且不在集合 *S* 中的節點，才有可能成為下一個被加入集合 *S* 的節點。因此在搜尋 A*評價分數最優的節點時，不用搜尋整個路網上的節點，只需要比較這些與集合 *S* 相連的節點即可，如此可以大幅縮短演算時間。這些節點可以另外設定為一個集合，稱為「候選路段」。

所有候選路段的起點必定為已經加入集合 *S* 的節點，訖點則必定為尚未加入集合 *S* 的節點。每當有一個新的候選路段被選為下一個加入集合 *S* 的節點，代表其他以此節點為訖點的候選路段已經不符合條件，需要從候選路段中移除；並且可能有其他以此節點為起點的路段，將會作為新的候選路段被加入「候選路段」集合。

第五章　路徑規劃實作

本研究實作國道高速公路路網，5.1 節為實驗範圍，5.2 節為路網節點相關資料；5.3 節為實驗相關假設與限制；5.4 節進行案例測試。

5.1 實驗範圍

本研究以國道高速公路一號與國道高速公路三號全線做為系統實作範圍，國道一號北起基隆端，南至高雄端 372.7 公里；國道三號北起基金交流道，南至大鵬灣端，全長 431.5 公里，如圖 5-1。

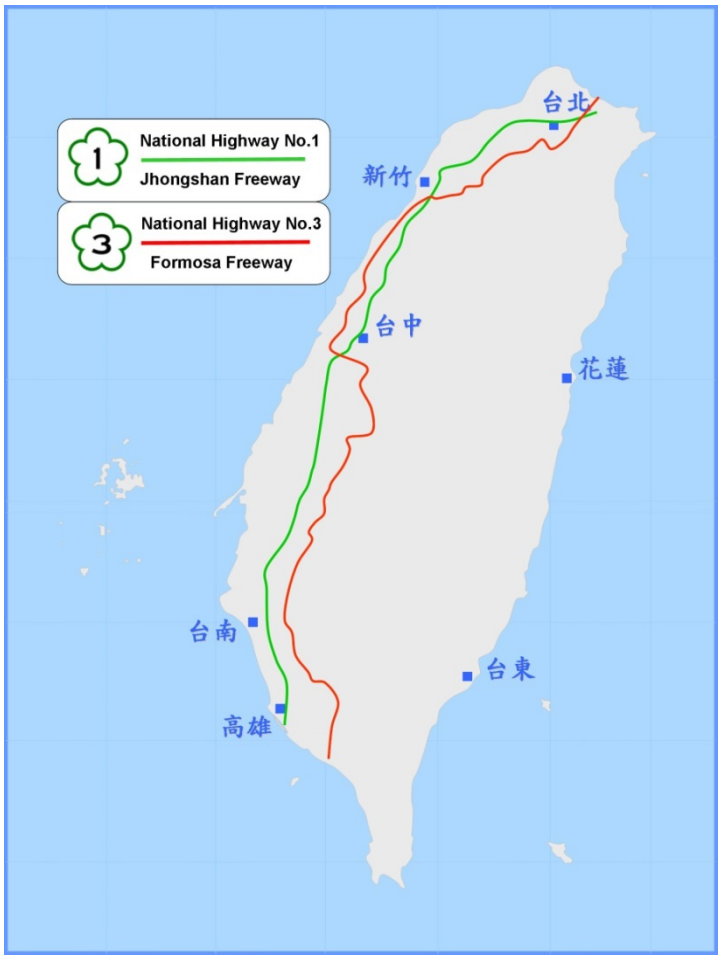


圖 5-1 國道高速公路示意圖

5.2 路網節點相關資料

實驗路網節點包含國道一號與國道三號主線全線之交流道、收費站與服務區。國道一號有 89 個節點，國道三號有 76 個節點，其中有 3 個為國道一號與國道三號互相轉換的系統交流道，共 162 個節點，詳細節點列表如表 5-1、表 5-2、表 5-3、表 5-4、表 5-5，節點分布圖如圖 5-2、圖 5-3、圖 5-4 的國道一號與三號部分。

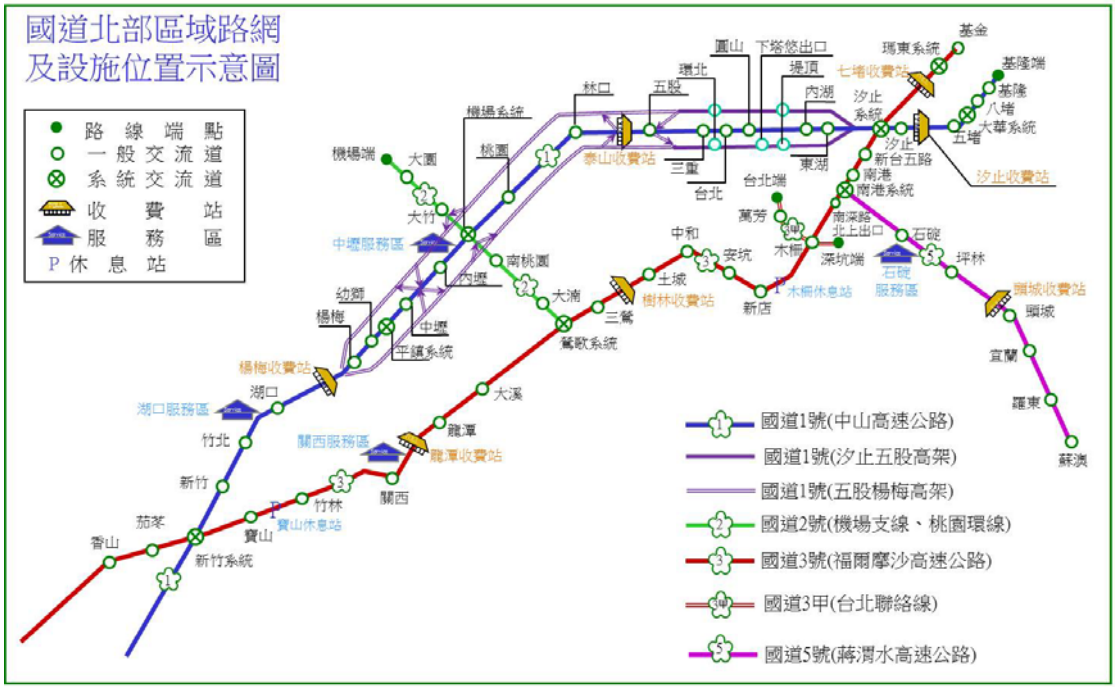


圖 5-2 國道北部區域路網及設施位置示意圖

圖 5-3 國道中部區域路網及設施位置示意圖

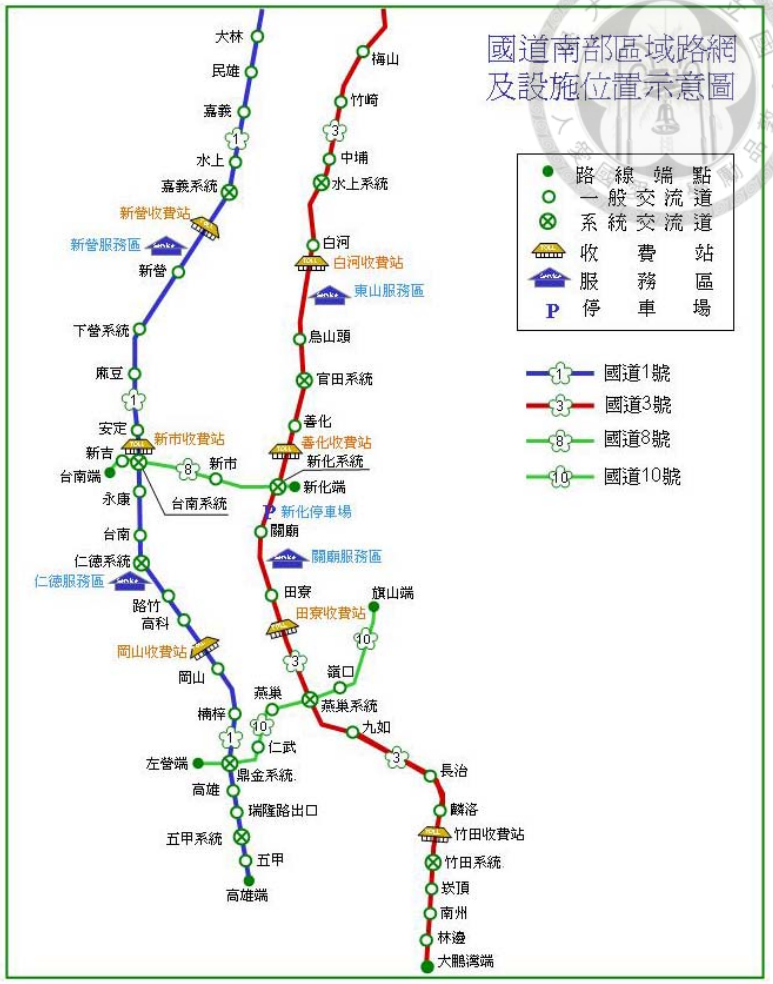


圖 5-4 國道南部區域路網及設施位置示意圖

表 5-1 國道一號設施列表(1)

國道編號	設施里程	設施名稱	經度	緯度
1	0	基隆端	121.736615	25.124276
1	1	基隆交流道	121.728106	25.114688
1	2	八堵交流道	121.720809	25.103749
1	5	大華系統交流道	121.698806	25.096987
1	6	五堵交流道	121.687094	25.08676
1	9	汐止收費站	121.66404	25.078816
1	10	汐止交流道	121.654124	25.074235
1	11	汐止系統交流道	121.645689	25.072791
1	12	汐五高架汐止端	121.627944	25.066961
1	15	東湖交流道	121.608328	25.064988
1	16	內湖交流道	121.574733	25.070454
1	23	圓山交流道	121.53333	25.072655
1	25	台北交流道	121.513917	25.077927
1	27	三重交流道	121.494379	25.075736
1	32	五股轉接道	121.439336	25.06905
1	33	五股交流道	121.437533	25.068792
1	35	泰山收費站	121.415297	25.061426
1	36	泰山轉接道	121.414676	25.060964
1	41	林口交流道	121.362446	25.06491
1	49	桃園交流道	121.297629	25.037705

國道編號	設施里程	設施名稱	經度	緯度
1	52	機場系統交流道	121.273535	25.018814
1	55	中壢服務區	121.252182	25.002164
1	57	內壢交流道	121.240675	24.991732
1	59	中壢轉接道	121.213566	24.967516
1	62	中壢交流道	121.199715	24.957075
1	65	平鎮系統交流道	121.189592	24.935871
1	67	幼獅交流道	121.170911	24.92468
1	69	楊梅交流道	121.164506	24.90916
1	70	五楊高架楊梅端	121.15963	24.903197
1	71	楊梅收費站	121.146878	24.899361
1	83	湖口交流道	121.029537	24.875828
1	86	湖口服務區	121.011345	24.862073
1	91	竹北交流道	121.017188	24.823558
1	95	新竹交流道	121.008398	24.79776
1	99	新竹系統交流道	120.987673	24.757422
1	110	頭份交流道	120.918351	24.691398
1	117	造橋收費站	120.873295	24.644443
1	132	苗栗交流道	120.820303	24.524104
1	140	銅鑼交流道	120.783828	24.479247
1	150	三義交流道	120.758143	24.392641

表 5-2 國道一號設施列表(2)

國道編號	設施里程	設施名稱	經度	緯度
1	159	泰安服務區	120.713493	24.32824
1	160	后里交流道	120.701551	24.312358
1	162	后里收費站	120.69623	24.296405
1	165	台中系統交流道	120.69284	24.270753
1	168	豐原交流道	120.690286	24.250545
1	174	大雅交流道	120.656619	24.205627
1	178	台中交流道	120.631063	24.17521
1	181	南屯交流道	120.619926	24.152862
1	189	王田交流道	120.583127	24.11881
1	192	彰化系統交流道	120.561436	24.105638
1	198	彰化交流道	120.522766	24.065098
1	207	埔鹽系統交流道	120.50752	23.983743
1	211	員林交流道	120.50634	23.95346
1	218	員林收費站	120.493578	23.88995
1	220	北斗交流道	120.490363	23.875243
1	229	西螺服務區	120.478392	23.788624
1	230	西螺交流道	120.475312	23.781545
1	235	虎尾交流道	120.473968	23.725974
1	240	斗南交流道	120.471525	23.69312
1	243	雲林系統交流道	120.456416	23.668043
1	246	斗南收費站	120.444476	23.643502
1	250	大林交流道	120.437109	23.612077
1	257	民雄交流道	120.417473	23.558037
1	264	嘉義交流道	120.391583	23.494815
1	270	水上交流道	120.366967	23.444477
1	272	嘉義系統交流道	120.356688	23.424582
1	280	新營收費站	120.338841	23.358513
1	284	新營服務區	120.312122	23.339872
1	288	新營交流道	120.290861	23.307321
1	299	下營系統交流道	120.239106	23.219193
1	303	麻豆交流道	120.235564	23.182361
1	311	安定交流道	120.243579	23.117608
1	313	新市收費站	120.249693	23.096076
1	315	台南系統交流道	120.252216	23.079179
1	319	永康交流道	120.251474	23.042003
1	327	台南交流道	120.249433	22.972196
1	330	仁德系統交流道	120.24969	22.942665
1	335	仁德服務區	120.265714	22.906044
1	338	路竹交流道	120.27924	22.879576
1	342	高科交流道	120.295378	22.845795

表 5-3 國道一號設施列表(3)

國道編號	設施里程	設施名稱	經度	緯度
1	346	岡山收費站	120.314668	22.810968
1	349	岡山交流道	120.320623	22.787884
1	356	楠梓交流道	120.334452	22.737052
1	362	鼎金系統交流道	120.328957	22.675204
1	367	高雄交流道	120.336372	22.63051
1	369	瑞隆路出口	120.341286	22.606236
1	370	五甲系統交流道	120.341207	22.598664
1	371	五甲交流道	120.33899	22.59545
1	372	高雄端	120.326057	22.584043

表 5-4 國道三號設施列表(1)

國道編號	設施里程	設施名稱	經度	緯度
3	0	基金交流道	121.714887	25.139787
3	2	瑪東系統交流道	121.697679	25.126743
3	4	七堵收費站	121.684524	25.108528
3	10	汐止系統交流道	121.645683	25.072806
3	12	新台五路交流道	121.637958	25.055818
3	15	南港交流道	121.62466	25.042546
3	16	南港系統交流道	121.620862	25.03596
3	20	木柵交流道	121.595037	25.001952
3	26	新店交流道	121.553112	24.972664
3	31	安坑交流道	121.524554	24.969355
3	35	中和交流道	121.484412	24.989228
3	43	土城交流道	121.432216	24.960178
3	46	樹林收費站	121.395351	24.951307
3	50	三鶯交流道	121.35999	24.938782
3	54	鶯歌系統交流道	121.325013	24.939302
3	62	大溪交流道	121.264449	24.891896
3	68	龍潭交流道	121.223176	24.861684
3	72	龍潭收費站	121.199374	24.837248
3	79	關西交流道	121.17129	24.801568
3	90	竹林交流道	121.081039	24.766405

國道編號	設施里程	設施名稱	經度	緯度
3	98	寶山交流道	121.0061	24.75717
3	100	新竹系統交流道	120.9877	24.75744
3	103	茄苳交流道	120.9568	24.75685
3	109	香山交流道	120.9097	24.73519
3	115	西濱交流道	120.8663	24.70127
3	119	竹南交流道	120.8474	24.6812
3	122	後龍收費站	120.8319	24.6498
3	124	大山交流道	120.818	24.63645
3	130	後龍交流道	120.7859	24.59794
3	144	通霄交流道	120.7053	24.49715
3	156	苑裡交流道	120.6693	24.39666
3	158	大甲收費站	120.6661	24.37979
3	164	大甲交流道	120.6433	24.33716
3	169	中港系統交流道	120.6135	24.30746
3	176	沙鹿交流道	120.5904	24.25006
3	182	龍井交流道	120.5721	24.19506
3	191	和美交流道	120.5209	24.14018
3	196	彰化系統交流道	120.5614	24.10564
3	202	快官交流道	120.604	24.09184
3	207	烏日交流道	120.6461	24.0683

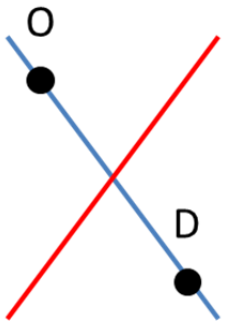
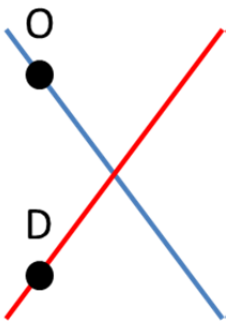
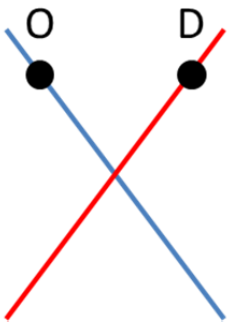
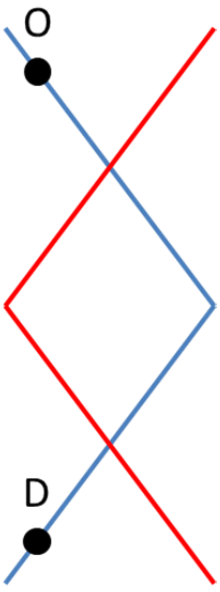
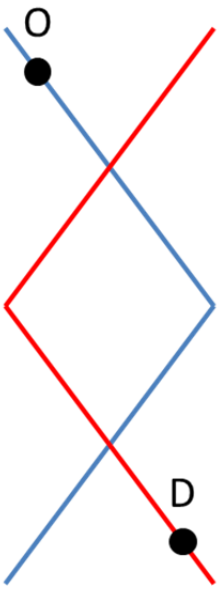
表 5-5 國道三號設施列表(2)

國道編號	設施里程	設施名稱	經度	緯度
3	209	中投交流道	120.6628	24.06532
3	211	霧峰交流道	120.6789	24.04952
3	214	霧峰系統交流道	120.6689	24.02747
3	217	草屯交流道	120.6524	24.00559
3	222	中興系統交流道	120.6553	23.95824
3	224	中興交流道	120.669	23.94872
3	234	名間收費站	120.7069	23.87512
3	236	名間交流道	120.7025	23.85232
3	243	竹山交流道	120.7019	23.79592
3	260	斗六交流道	120.5953	23.73296
3	269	古坑系統交流道	120.5741	23.66521
3	273	古坑收費站	120.5661	23.62622
3	279	梅山交流道	120.5228	23.59556
3	290	竹崎交流道	120.4847	23.52004
3	297	中埔交流道	120.4848	23.4507
3	300	水上系統交流道	120.4686	23.4293
3	311	白河交流道	120.4395	23.34728
3	313	白河收費站	120.4325	23.33012
3	329	烏山頭交流道	120.356	23.21265
3	334	官田系統交流道	120.3521	23.16649
3	340	善化交流道	120.3317	23.12368
3	342	善化收費站	120.3281	23.10667
3	346	新化系統交流道	120.3239	23.06436
3	357	關廟交流道	120.3244	22.97698
3	369	田寮交流道	120.3605	22.88047
3	373	田寮收費站	120.3735	22.85889
3	383	燕巢系統交流道	120.4262	22.7848
3	391	九如交流道	120.4916	22.7535
3	400	長治交流道	120.5555	22.69636
3	407	麟洛交流道	120.5409	22.644
3	411	竹田收費站	120.5291	22.6056
3	415	竹田系統交流道	120.5281	22.57145
3	421	崁頂交流道	120.5248	22.51328
3	424	南州交流道	120.5312	22.48536
3	430	林邊交流道	120.5059	22.44537
3	431	大鵬灣端	120.4992	22.4415

5.3 系統實作案例測試

由於高速公路路網只有三個系統交流道，整個旅行路徑須至少跨越兩個系統交流道才會有路徑選擇問題；如果旅行路徑間只跨越一個以下的系統交流道，此系統也能提供旅行時間預測。為了體現 Dijkstra's Algorithm 與 A* Algorithm 的差異，本實驗將實作旅行路徑跨越一個系統交流道與旅行路徑跨越兩個以上系統交流道的起訖點組合，兩種組合又分別設計三種與兩種不同的情境，如表 5-6。分別測試 Dijkstra's Algorithm 與 A* Algorithm 的演算速度與搜尋結果，比較兩種演算法的優缺。

表 5-6 系統測試情境

	起訖點於相同國道	起訖點於不同國道	經系統交流道後反行
經過一個系統交流道			
經過兩個系統交流道			

5.3.1 旅行路徑經過一個以下系統交流道

若旅行路徑經過一個以下的系統交流道，則無路徑選擇的必要。系統相當於預測旅行時間，提供使用者旅次安排參考。旅行路徑經過一個系統交流道的情況又可以分為三種不同情境：起點與訖點位於相同國道、起點與訖點位於不同國道，起點與訖點位於不同國道且皆在系統交流道的北側或南側。起訖點位於不同國道的情況，旅行路徑一定會經過系統交流道轉換國道，此轉換時間在此假設為 0。以下將按照不同情境，以旅行路徑跨越彰化系統交流道之起訖點組合為例，分別實作路徑演算系統。

(1). 起點與訖點位於相同國道

- 起點：國道一號 造橋收費站
- 訖點：國道一號 斗南收費站
- 預計出發時間：自 07:00 開始，每 3 個小時演算一次，直到 19:00

表 5-7 旅行路徑經過一個系統交流道-起點與訖點位於相同國道

Dijkstra's Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
造橋收費站	斗南收費站	07:00	08:20	80	483	79
造橋收費站	斗南收費站	10:00	11:22	82	497	82
造橋收費站	斗南收費站	13:00	14:20	80	489	82
造橋收費站	斗南收費站	16:00	18:23	143	690	115
造橋收費站	斗南收費站	19:00	20:19	79	488	82
A* Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
造橋收費站	斗南收費站	07:00	08:20	80	168	26
造橋收費站	斗南收費站	10:00	11:22	82	168	26
造橋收費站	斗南收費站	13:00	14:20	80	167	26
造橋收費站	斗南收費站	16:00	18:23	143	464	75
造橋收費站	斗南收費站	19:00	20:19	79	169	26

(2). 起點與訖點位於不同國道

- 起點：國道一號 造橋收費站
- 訖點：國道三號 古坑收費站
- 預計出發時間：自 07:00 開始，每 3 個小時演算一次，直到 19:00

表 5-8 旅行路徑經過一個系統交流道-起點與訖點位於不同國道

Dijkstra's Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
造橋收費站	古坑收費站	07:00	08:32	92	541	88
造橋收費站	古坑收費站	10:00	11:35	95	623	100
造橋收費站	古坑收費站	13:00	14:32	92	623	101
造橋收費站	古坑收費站	16:00	17:34	94	454	74
造橋收費站	古坑收費站	19:00	20:33	93	625	101
A* Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
造橋收費站	古坑收費站	07:00	08:32	92	289	44
造橋收費站	古坑收費站	10:00	11:35	95	290	46
造橋收費站	古坑收費站	13:00	14:32	92	277	44
造橋收費站	古坑收費站	16:00	17:34	94	231	35
造橋收費站	古坑收費站	19:00	20:33	93	283	44

(3). 起點與訖點位於不同國道且皆在系統交流道的北側或南側

- 起點：國道一號 造橋收費站
- 訖點：國道三號 大甲收費站
- 預計出發時間：自 07:00 開始，每 3 個小時演算一次，直到 19:00

表 5-9 旅行路徑經過一個系統交流道-起點與訖點位於不同國道且皆在系統交流道的北側或南側

Dijkstra's Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
造橋收費站	大甲收費站	07:00	08:13	73	434	70
造橋收費站	大甲收費站	10:00	11:16	76	466	76
造橋收費站	大甲收費站	13:00	14:14	74	477	76
造橋收費站	大甲收費站	16:00	17:13	73	388	62
造橋收費站	大甲收費站	19:00	20:14	74	470	75
A* Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
造橋收費站	大甲收費站	07:00	08:13	73	257	40
造橋收費站	大甲收費站	10:00	11:16	76	264	41
造橋收費站	大甲收費站	13:00	14:14	74	260	40
造橋收費站	大甲收費站	16:00	17:13	73	251	39
造橋收費站	大甲收費站	19:00	20:14	74	263	41

5.3.2 旅行路徑經過兩個以上系統交流道

若旅行路徑行經兩個以上的系統交流道，表示在兩個系統交流道之間的路徑是有選擇的餘地，系統會自兩條可能路徑中，選擇旅行時間較短的一條做為建議路徑輸出。旅行路徑經過兩個以上系統交流道的情況又可以分為兩種不同情境：起點與訖點位於相同國道、起點與訖點位於不同國道。起點與訖點位於相同國道的情況，可能會經過 0 次或 2 次系統交流道轉換國道；起點與訖點位於不同國道的情況，則一定會經過 1 次系統交流道轉換國道，此轉換時間在此假設為 0。以下將按照不同情境，以起訖點跨越新竹系統交流道與彰化系統交流道為例，分別實作路徑演算系統。

(1). 起訖點位於相同國道

- 起點：國道一號 楊梅收費站
- 訖點：國道一號 斗南收費站
- 預計出發時間：自 07:00 開始，每 3 個小時演算一次，直到 19:00

表 5-10 旅行路徑經過兩個以上系統交流道-起訖點位於相同國道

Dijkstra's Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
楊梅收費站	斗南收費站	07:00	08:54	114	612	102
楊梅收費站	斗南收費站	10:00	11:48	108	613	102
楊梅收費站	斗南收費站	13:00	14:50	110	605	102
楊梅收費站	斗南收費站	16:00	18:19	139	655	109
楊梅收費站	斗南收費站	19:00	20:51	111	611	102
A* Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
楊梅收費站	斗南收費站	07:00	08:54	114	298	47
楊梅收費站	斗南收費站	10:00	11:48	108	283	42
楊梅收費站	斗南收費站	13:00	14:50	110	269	43
楊梅收費站	斗南收費站	16:00	18:19	139	442	72
楊梅收費站	斗南收費站	19:00	20:51	111	269	43

(2). 起訖點位於不同國道

- 起點：國道一號 楊梅收費站
- 訖點：國道三號 古坑收費站
- 預計出發時間：自 07:00 開始，每 3 個小時演算一次，直到 19:00

表 5-11 旅行路徑經過兩個以上系統交流道-起訖點位於不同國道

Dijkstra's Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
楊梅收費站	古坑收費站	07:00	09:06	126	634	107
楊梅收費站	古坑收費站	10:00	11:59	119	624	107
楊梅收費站	古坑收費站	13:00	15:01	121	660	107
楊梅收費站	古坑收費站	16:00	18:01	121	588	99
楊梅收費站	古坑收費站	19:00	21:03	123	637	107
A* Algorithm						
起點	訖點	出發時間	抵達時間	累計旅行時間	程式運行時間	已搜尋的節點
楊梅收費站	古坑收費站	07:00	09:06	126	439	70
楊梅收費站	古坑收費站	10:00	11:59	119	467	67
楊梅收費站	古坑收費站	13:00	15:01	121	419	69
楊梅收費站	古坑收費站	16:00	18:01	121	322	54
楊梅收費站	古坑收費站	19:00	21:03	123	411	68

綜觀以上五種情境實作結果可以發現，A* Algorithm 的預測結果與 Dijkstra's Algorithm 相同，但是運算時間與搜尋的節點數量卻能夠減少到 Dijkstra's Algorithm 的 30%~65%，當路網越趨龐大時，能夠減少更多的運算時間提升運算效率。

另外，由於高速公路路網呈現南北非常狹長的情況，當起訖點呈現南北向(情境 1-(1)、情境 1-(2))的狀況，A* Algorithm 的演算效率會比當起訖點呈現東西向(情境 1-(3))的狀況好。

5.4 後推式演算法實作測試

為驗證後推式演算法能找到與前推式演算法相同的建議路徑，本節設計兩個起訖點組合：旅行路徑經過一個系統交流道、旅行路徑經過兩個系統交流道。先輸入預計抵達時間，以後推式路徑演算求得出發時間，再從此出發時間以前推式路徑演算驗證後推式路徑演算。

5.4.1 旅行路徑經過一個系統交流道

- 起點：國道三號 大甲收費站
- 訖點：國道一號 斗南收費站
- 預計抵達時間：20:00

表 5-12 前推式演算與後推式演算比較(1)-Dijkstra's Algorithm

Dijkstra's Algorithm					
前推式演算			後推式演算		
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
3	大甲收費站	18:58	3	大甲收費站	18:58
3	大甲交流道	19:03	3	大甲交流道	19:04
3	中港系統交流道	19:07	3	中港系統交流道	19:08
3	沙鹿交流道	19:12	3	沙鹿交流道	19:13
3	龍井交流道	19:16	3	龍井交流道	19:17
3	和美交流道	19:21	3	和美交流道	19:22
3	彰化系統交流道	19:24	3	彰化系統交流道	19:25
1	彰化系統交流道	19:24	1	彰化系統交流道	19:25
1	彰化交流道	19:28	1	彰化交流道	19:29
1	埔鹽系統交流道	19:34	1	埔鹽系統交流道	19:35
1	員林交流道	19:36	1	員林交流道	19:37
1	員林收費站	19:40	1	員林收費站	19:41
1	北斗交流道	19:42	1	北斗交流道	19:43
1	西螺服務區	19:47	1	西螺服務區	19:48
1	西螺交流道	19:48	1	西螺交流道	19:49
1	虎尾交流道	19:52	1	虎尾交流道	19:53
1	斗南交流道	19:55	1	斗南交流道	19:56
1	雲林系統交流道	19:57	1	雲林系統交流道	19:58
1	斗南收費站	19:59	1	斗南收費站	20:00
累計旅行時間：		61 分鐘	累計旅行時間：		61 分鐘
系統演算時間：		258 ms	系統演算時間：		693 ms
已搜尋的節點數量：		42 個	已搜尋的節點數量：		54 個

表 5-13 前推式演算與後推式演算比較(2)-A* Algorithm

A* Algorithm					
前推式演算			後推式演算		
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
3	大甲收費站	18:58	3	大甲收費站	18:58
3	大甲交流道	19:03	3	大甲交流道	19:04
3	中港系統交流道	19:07	3	中港系統交流道	19:08
3	沙鹿交流道	19:12	3	沙鹿交流道	19:13
3	龍井交流道	19:16	3	龍井交流道	19:17
3	和美交流道	19:21	3	和美交流道	19:22
3	彰化系統交流道	19:24	3	彰化系統交流道	19:25
1	彰化系統交流道	19:24	1	彰化系統交流道	19:25
1	彰化交流道	19:28	1	彰化交流道	19:29
1	埔鹽系統交流道	19:34	1	埔鹽系統交流道	19:35
1	員林交流道	19:36	1	員林交流道	19:37
1	員林收費站	19:40	1	員林收費站	19:41
1	北斗交流道	19:42	1	北斗交流道	19:43
1	西螺服務區	19:47	1	西螺服務區	19:48
1	西螺交流道	19:48	1	西螺交流道	19:49
1	虎尾交流道	19:52	1	虎尾交流道	19:53
1	斗南交流道	19:55	1	斗南交流道	19:56
1	雲林系統交流道	19:57	1	雲林系統交流道	19:58
1	斗南收費站	19:59	1	斗南收費站	20:00
累計旅行時間：		61 分鐘	累計旅行時間：		61 分鐘
系統演算時間：		134 ms	系統演算時間：		344 ms
已搜尋的節點數量：		20 個	已搜尋的節點數量：		25 個

5.4.2 旅行路徑經過兩個系統交流道

- 起點：國道三號 龍潭收費站
- 訖點：國道一號 員林收費站
- 預計抵達時間：20:00

表 5-14 前推式演算與後推式演算比較(3)-Dijkstra's Algorithm

Dijkstra's Algorithm					
前推式演算			後推式演算		
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
3	龍潭收費站	18:17	3	龍潭收費站	18:17
3	關西交流道	18:24	3	關西交流道	18:25
3	竹林交流道	18:36	3	竹林交流道	18:37
3	寶山交流道	18:45	3	寶山交流道	18:45
3	新竹系統交流道	18:47	3	新竹系統交流道	18:48
1	新竹系統交流道	18:47	1	新竹系統交流道	18:48
1	頭份交流道	18:54	1	頭份交流道	18:54
1	造橋收費站	18:58	1	造橋收費站	18:58
1	苗栗交流道	19:07	1	苗栗交流道	19:07
1	銅鑼交流道	19:12	1	銅鑼交流道	19:12
1	三義交流道	19:18	1	三義交流道	19:18
1	泰安服務區	19:24	1	泰安服務區	19:24
1	后里交流道	19:25	1	后里交流道	19:25
1	后里收費站	19:26	1	后里收費站	19:26
1	台中系統交流道	19:28	1	台中系統交流道	19:28
1	豐原交流道	19:30	1	豐原交流道	19:30
1	大雅交流道	19:34	1	大雅交流道	19:34
1	台中交流道	19:36	1	台中交流道	19:36
1	南屯交流道	19:38	1	南屯交流道	19:38
1	王田交流道	19:42	1	王田交流道	19:42
1	彰化系統交流道	19:44	1	彰化系統交流道	19:44
1	彰化交流道	19:47	1	彰化交流道	19:47
1	埔鹽系統交流道	19:53	1	埔鹽系統交流道	19:53
1	員林交流道	19:55	1	員林交流道	19:55
1	員林收費站	19:59	1	員林收費站	20:00
累計旅行時間：		102 分鐘	累計旅行時間：		102 分鐘
系統演算時間：		397 ms	系統演算時間：		1422 ms
已搜尋的節點數量：		66 個	已搜尋的節點數量：		97 個

表 5-15 前推式演算與後推式演算比較(4)-A* Algorithm

A* Algorithm					
前推式演算			後推式演算		
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
3	龍潭收費站	18:17	3	龍潭收費站	18:17
3	關西交流道	18:24	3	關西交流道	18:25
3	竹林交流道	18:36	3	竹林交流道	18:37
3	寶山交流道	18:45	3	寶山交流道	18:45
3	新竹系統交流道	18:47	3	新竹系統交流道	18:48
1	新竹系統交流道	18:47	1	新竹系統交流道	18:48
1	頭份交流道	18:54	1	頭份交流道	18:54
1	造橋收費站	18:58	1	造橋收費站	18:58
1	苗栗交流道	19:07	1	苗栗交流道	19:07
1	銅鑼交流道	19:12	1	銅鑼交流道	19:12
1	三義交流道	19:18	1	三義交流道	19:18
1	泰安服務區	19:24	1	泰安服務區	19:24
1	后里交流道	19:25	1	后里交流道	19:25
1	后里收費站	19:26	1	后里收費站	19:26
1	台中系統交流道	19:28	1	台中系統交流道	19:28
1	豐原交流道	19:30	1	豐原交流道	19:30
1	大雅交流道	19:34	1	大雅交流道	19:34
1	台中交流道	19:36	1	台中交流道	19:36
1	南屯交流道	19:38	1	南屯交流道	19:38
1	王田交流道	19:42	1	王田交流道	19:42
1	彰化系統交流道	19:44	1	彰化系統交流道	19:44
1	彰化交流道	19:47	1	彰化交流道	19:47
1	埔鹽系統交流道	19:53	1	埔鹽系統交流道	19:53
1	員林交流道	19:55	1	員林交流道	19:55
1	員林收費站	19:59	1	員林收費站	20:00
累計旅行時間：		102 分鐘	累計旅行時間：		102 分鐘
系統演算時間：		197 ms	系統演算時間：		786 ms
已搜尋的節點數量：		30 個	已搜尋的節點數量：		53 個

根據上述兩種情境的前推式演算與後推式演算結果可以發現後推式演算與前推式演算的結果一致，而且無論是 Dijkstra's Algorithm 還是 A* Algorithm，後推式演算法的搜尋數量與演算時間都大於前推式演算法。雖然最佳路徑只有一條，但是前推式路徑演算與後推式路徑演算搜尋節點的方向並不相同(前推式演算由起點開始延伸，後推式演算由訖點開始延伸)，因此雖然兩者在演算結束時搜尋到的節點數量與節點內容不一定一樣，但是建議路徑卻會相同。

另外，比較使用 A* Algorithm 的後推式演算與使用 Dijkstra's Algorithm 的前推式演算，雖然演算結束時前者搜尋到的節點數量較後者少，但是前者的演算時間仍然比後者長。其原因乃是在查詢一路段之路段旅行時間時，前推式演算法能夠直接以進入路段的時間查得路段旅行時間；後推式演算法卻需要反覆測試進入路段的旅行時間，以求得能夠在期望時間離開路段的進入時間。對同一個路段，前推式路徑演算只需要查詢一次即可得到路段旅行時間，後推式路徑演算卻需反覆查詢數次才能得到路段旅行時間，因此使得搜尋到同樣數量節點的情況下，後推式演算法需要比前推式演算法更長的演算時間。

第六章　結論與建議

本研究為依時性 A*路徑演算系統開發，成功將依時性路徑搜尋概念導入 A*路徑演算中，結合依時性路段旅行時間資料庫，建置了一套適用於國道高速公路的依時性最佳路徑演算系統，包含前推式與後推式路徑演算，以符合使用者不同的需求。並且以國道高速公路路網為例，實測 Dijkstra's Algorithm 與 A* Algorithm 在不同情境下的運作效率，證實 A* Algorithm 可以在較短的時間內求得令人滿意的建議路徑。

6.1 結論

1. 本研究旨在建構一適用於國道高速公路路網之「依時性 A*路徑演算系統」，並以物件導向程式語言 JAVA 實作。使用者於行前輸入起訖點、預計出發/抵達時間，透過依時性前推/後推路徑演算法，提供使用者建議路徑以及預測旅行時間，作為旅行規劃的參考。
2. 路徑演算系統同時包含前推式路徑演算與後推式路徑演算。使用者可依照旅次目的不同，選擇不同的演算法做為旅次規劃的參考。
3. 本系統沒有以傳統路徑規劃常用的空間距離做為旅行成本，而改採對旅行者來說感受更深刻的旅行時間做為路段旅行成本。透過引入旅行時間資料庫，將依時性概念整合到路徑演算法中，讓演算法能夠反應隨著時間不斷變化的交通狀況，提供更準確的旅行時間預測與路徑導引。
4. 本研究比較 Dijkstra's Algorithm 與從其中衍伸的 A* Algorithm，發現 A* Algorithm 能夠限制路徑搜尋方向，有效減少路徑搜尋時間增加演算效率，當路網變得複雜龐大時，A* Algorithm 的效果會變得更加顯著。A* Algorithm 的搜尋範圍可能不如 Dijkstra's Algorithm，但是能夠在較

短的時間內求得令人滿意的旅行路徑。經範例路網實作證實，A* Algorithm 只需要搜尋 30%~65%的節點數量即可找到與 Dijkstra's Algorithm 相同的最佳旅行路徑。

5. 對於想要在特定時間抵達訖點的使用者，查詢條件為起訖點及預計抵達時間的後推式路徑演算法將更能符合需求，節省以不同出發時間重複查找的時間。後推式演算法以連續型時間資料解決時間間隙的問題，避免因為時間間隙而使得路徑演算出現偏估。
6. 後推式演算法能夠找到與前推式演算法同的最佳旅行路徑，但是後推式演算法因為需要反覆查詢進入路段時間以得到合理的路段旅行時間，所以在搜尋節點數量相同的情況下，後推式演算法需要的運算時間較前推式演算法長。

6.2 建議

1. 東西向國道高速公路持續建置與完備，實驗路網範圍加入國道二號、國道四號、國道六號、國道八號、國道十號等東西向高速公路，使得路徑規劃的選擇更加多元，服務範圍也更加廣闊。
2. 目前旅行時間資料庫的來源為 VD 與 ETC 進行資料插補與資料融合後的歷史資料，雖然歷史資料代表一定程度的趨勢，但仍然缺乏即時性。期望未來國道全線計程收費實施後，增加為數可觀的 ETC 電子收費系統，能夠提供更迅速更準確的即時旅行時間資料，甚至透過傅立葉轉換、卡曼濾波(Kalman filter)等方式，提供預測性旅行時間資料。
3. 國道路網的特色是所有國道均依靠系統交流道連接。在不考慮以平面道路轉換國道的情況下，可以將國道路網視為被系統交流道分為許多區段，區段與區段間依靠系統交流道連接，區段內並無路徑選擇的需求。以區段做為路徑選擇的單位，將可大幅簡化路徑演算的複雜度。

或者可以區段做為 A* Algorithm 的評價指標，改良 A* Algorithm，令演算法優先搜尋起訖點所在區段內的節點。

4. 路徑演算所依據的路段成本不必侷限於空間距離或者旅行時間，可以考慮加入綠色安全指標(GSI)之類不同的路段成本做為路徑演算的依據，甚至引入差別訂價的概念,不只能提供使用者更多元的路徑決策模式，更有可能協助管理者進行動態交通指派應用。

5. 跨平台發展，配合手持行動裝置潮流，開發能夠適用於智慧型手機、平板電腦、車載機……等環境之系統。並將旅行時間資料庫建置為雲端資料庫，利用 Wi-Fi 無線網路或近場資訊系統(Near Field Informatics, N-Fi)可即時查詢路段旅行時間資料。讓使用環境不必限制於室內，在戶外也能隨時滿足路徑規劃的需求。

參考文獻

[1] Cherkassky, B.V., A.V. Goldberg, and T. Radzik. (1996). Shortest paths algorithms: Theory and experimental evaluation. Mathematical Programming, 129-174.

[2] 任維廉, 涂榮庭, 呂明穎, 呂堂榮. (2009). 探討先進旅行者資訊系統相關商品的知覺風險. 中華民國運輸學會.

[3] 交通部運輸研究所. (2001). 台灣地區發展智慧型運輸系統 (ITS) 系統架構之研究（Ⅰ). 交通部運輸研究所.

[4] Smith, J.C. and K. Russam. (1989). Some possible effects of AUTOGUIDE on traffic in London. IEEE International Conference on Vehicle Navigation and Information Systems. Toronto, Canada.

[5] French, B. and C. Queree. (1991). The intelligent highway. RTI/IVHS News. London.

[6] Boyce, D.E., A.M. Kirson, and J.L. Schofer. (1994). Design and implementation of ADVANCE：The Illinois dynamic navigation and route guidance demonstration program. IEEE International Conference on Vehicle Navigation and Information Systems. Ottawa, Canada.

[7] Klunder, G.A. and H.N. Post. (2006). The shortest path problem on large-scale real-road networks. NETWORKS, 182-194.

[8] Pallottino S., and Scutella M. G. (1998). Shortest path algorithms in transportation models: classical and innovative aspects, Equilibrium and Advanced Transportation Modeling.

[9] Lawler, E. L. (1976), Combinatorial Optimization: Networks and Matroids. Holt, New York : Rinehart and Winston.Lewis, C. D. (1982), Industrial and Business Forecasting Method, Southampton: The Camelot Press Ltd.

[10] Gallo, G., and Pallottino, S. (1986), "Shortest Path Methods: An Unifying Approach", Mathematical Programming Study, Vol. 26, pp. 38-64.

[11] Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 269-271.

[12] Bellman, R. (1958). On a routing problem. Quart. Appl. Mathematics, 87-90.

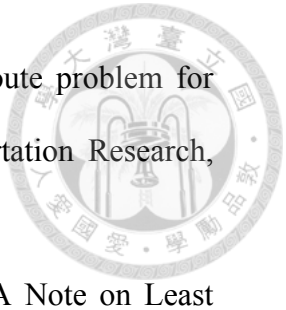
[13] Kershenbaum, A. (1981), "A note on finding shortest path trees," Networks, Vol. 11, No.1, pp. 399-400.

[14] Gallo, G. (1980), "Reoptimization procedures in shortest path problems", Rivista di Matematica per le Scienze Economiche e Sociali, Vol. 3, No.1, pp. 3-13.

[15] Dreyfus, S.E. (1969). An appraisal of some shortest path algorithms. Operations Research, 395-412.

[16] Ziliaskopoulos, A. K., and Mahmassani, H. S. (1993), "Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications," Transportation Research Record, No. 1408, pp. 94-100.

[17] Desrochers, M., and Soumis, F. (1988), "A generalized permanent labelling algorithm for the shortest path problem with time windows", INFOR, Vol. 26, No. 3, pp. 191-212.



[18] Kirby, R. F., and Potts, R. B. (1969), "The minimum route problem for networks with turn penalties and prohibitions," Transportation Research, Vol. 3, pp. 397-408.

[19] Ziliaskopoulos, A. K., and Mahmassani, H. S. (1996), "A Note on Least Time Path Computation Considering Delays and Prohibitions for Intersection Movements", Transportation Research Part B, Vol. 30, No. 5, pp. 359-367.

[20] Hansen, P. (1980), Bicriterion path problems, In Fandel, G., and Gal, T. (Eds.), Multicriteria decision making: theory and applications, Heidelberg :Springer, pp. 109-127.

[21] Jacob, R., M.V. Marathe, and K. Nagel. (1998). A computational study of routing algorithms for realistic transportation networks. Proceedings WAE'98 Saarbrucke, 167-178. Germany.

[22] Cordeau, J.F., et al. (2002). A guide to vehicle routing heuristics. The Journal of the Operational Research Society, 512-522.

[23] Chabini, I. (1997). A new short paths algorithm for discrete dynamic networks. Proceedings of 8th IFAC Symposium on Transportation Systems. Chania, Greece.

[24] Chabini, I. (1998). Discrete dynamic shortest path problems in transportation applications: complexity and algorithms with optimal run time. Transportation Research Records, 170-175.

[25] Cooke, K.L. and E. Halsey. (1966). The shortest route through a network with time dependent intermodal transit time. Journal of Math. Anal. Appl., 492-498.

[26] Bellman, R., On a routing problem. Quart. Appl. Mathematics, 1958. 16: p. 87-90.

[27] Sturtevant, N. and Buro, M. (2005). Partial pathfinding using map abstraction and refinement. ceedings of AAAI, 47-52.

[28] Daganzo, C.F. (2002). Reversibility of the time-dependent shortest path problem. Transportation Research Part B, 665-668.

[29] Ahuja, R.K., T.L. Magnanti, and J.B. Orlin. (1993). Network flows: Theory, algorithms, and applications. Englewood Cliffs, NY.: Prentice-Hall.

[30] Bertsekas, D.P. (1991). Linear network optimization: Algorithms and codes. Cambridge, MA: The MIT Press.

[31] Denardo, E.V. and B.L. Fox. (1979). Shortest-route methods: 1. Reaching, pruning, and buckets. Operations Research, 161-186.

[32] Denardo, E.V. and B.L. Fox. (1979). Shortest-route methods: 2. Group knapsacks, expanded networks, and branch-and-bound. Operations Research, 548-566.

[33] Deo, N. and C. Pang. (1984). Shortest path algorithms: Taxonomy and annotation. Networks, 275-323.

[34] Gallo, G. and S. Pallottino. (1984). Shortest path methods in transportation models. Transportation Planning Models (M. Florian, ed.), 227-256.

[35] Gallo, G. and S. Pallottino. (1988). Shortest path algorithms. Annals of Operational Research, 3-79.

[36] Halpern, J. and I. Priess. (1974). Shortest path with time constraints on movement and parking. Networks, 241-253.

[37] Johnson, D.B. (1973). Algorithms for shortest paths. NY: Cornell Univ.: Ithaca.

[38] Orda, A. and R. Rom. (1990). Shortest path and minimum-delay algorithms in network with time-dependent edge length. Journal of the ACM, 607-625.

[39] Shier, D.R. and C. Witzgall. (1981). Properties of labeling methods for determining shortest path trees. Journal of Research of the National Bureau of Standards, 317-330.

[40] SteenbrinkP.A. (1974). Optimization of transport networks. London: John Wiley & Sons.

[41] Van Vliet, D. (1978). Improved shortest path algorithms for transport networks. Transportation Research, 7-20.

[42] 張堂賢、闕嘉宏. (2012). 依時性後推式路徑演算系統開發. 運輸學刊.

[43] 葉羿稚. (2007). 行前即時路徑規劃演算法之研究. 台灣大學土木工程學系碩士論文.

附錄

(一)、旅行路徑經過一個以下系統交流道：起點與訖點位於相同國道 詳細路徑演算結果

起點：	造橋收費站		時間：	07:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	07:00	1	彰化系統交流道	07:46
1	苗栗交流道	07:08	1	彰化交流道	07:49
1	銅鑼交流道	07:14	1	埔鹽系統交流道	07:54
1	三義交流道	07:20	1	員林交流道	07:57
1	泰安服務區	07:26	1	員林收費站	08:01
1	后里交流道	07:27	1	北斗交流道	08:02
1	后里收費站	07:28	1	西螺服務區	08:08
1	台中系統交流道	07:30	1	西螺交流道	08:09
1	豐原交流道	07:32	1	虎尾交流道	08:13
1	大雅交流道	07:35	1	斗南交流道	08:16
1	台中交流道	07:38	1	雲林系統交流道	08:18
1	南屯交流道	07:39	1	斗南收費站	08:20
1	王田交流道	07:44			
累計旅行時間：			80		分鐘
系統演算時間：			483		ms
已搜尋的節點數量：			79		個

起點：	造橋收費站		時間：	07:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	07:00	1	彰化系統交流道	07:46
1	苗栗交流道	07:08	1	彰化交流道	07:49
1	銅鑼交流道	07:14	1	埔鹽系統交流道	07:54
1	三義交流道	07:20	1	員林交流道	07:57
1	泰安服務區	07:26	1	員林收費站	08:01
1	后里交流道	07:27	1	北斗交流道	08:02
1	后里收費站	07:28	1	西螺服務區	08:08
1	台中系統交流道	07:30	1	西螺交流道	08:09
1	豐原交流道	07:32	1	虎尾交流道	08:13
1	大雅交流道	07:35	1	斗南交流道	08:16
1	台中交流道	07:38	1	雲林系統交流道	08:18
1	南屯交流道	07:39	1	斗南收費站	08:20
1	王田交流道	07:44			
累計旅行時間：			80		分鐘
系統演算時間：			483		ms
已搜尋的節點數量：			79		個

起點：	造橋收費站		時間：	10:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	10:00	1	彰化系統交流道	10:48
1	苗栗交流道	10:08	1	彰化交流道	10:52
1	銅鑼交流道	10:15	1	埔鹽系統交流道	10:57
1	三義交流道	10:21	1	員林交流道	10:59
1	泰安服務區	10:28	1	員林收費站	11:03
1	后里交流道	10:28	1	北斗交流道	11:05
1	后里收費站	10:29	1	西螺服務區	11:10
1	台中系統交流道	10:31	1	西螺交流道	11:12
1	豐原交流道	10:33	1	虎尾交流道	11:15
1	大雅交流道	10:38	1	斗南交流道	11:18
1	台中交流道	10:40	1	雲林系統交流道	11:20
1	南屯交流道	10:42	1	斗南收費站	11:22
1	王田交流道	10:47			
累計旅行時間：			82		分鐘
系統演算時間：			497		ms
已搜尋的節點數量：			82		個

起點：	造橋收費站		時間：	10:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	10:00	1	彰化系統交流道	10:48
1	苗栗交流道	10:08	1	彰化交流道	10:52
1	銅鑼交流道	10:15	1	埔鹽系統交流道	10:57
1	三義交流道	10:21	1	員林交流道	10:59
1	泰安服務區	10:28	1	員林收費站	11:03
1	后里交流道	10:28	1	北斗交流道	11:05
1	后里收費站	10:29	1	西螺服務區	11:10
1	台中系統交流道	10:31	1	西螺交流道	11:12
1	豐原交流道	10:33	1	虎尾交流道	11:15
1	大雅交流道	10:38	1	斗南交流道	11:18
1	台中交流道	10:40	1	雲林系統交流道	11:20
1	南屯交流道	10:42	1	斗南收費站	11:22
1	王田交流道	10:47			
累計旅行時間：			82		分鐘
系統演算時間：			497		ms
已搜尋的節點數量：			82		個

起點：	造橋收費站		時間：	13:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	13:00	1	彰化系統交流道	13:46
1	苗栗交流道	13:08	1	彰化交流道	13:50
1	銅鑼交流道	13:13	1	埔鹽系統交流道	13:55
1	三義交流道	13:19	1	員林交流道	13:57
1	泰安服務區	13:25	1	員林收費站	14:01
1	后里交流道	13:26	1	北斗交流道	14:03
1	后里收費站	13:27	1	西螺服務區	14:08
1	台中系統交流道	13:29	1	西螺交流道	14:09
1	豐原交流道	13:31	1	虎尾交流道	14:13
1	大雅交流道	13:35	1	斗南交流道	14:16
1	台中交流道	13:38	1	雲林系統交流道	14:18
1	南屯交流道	13:40	1	斗南收費站	14:20
1	王田交流道	13:44			
累計旅行時間：			80		分鐘
系統演算時間：			489		ms
已搜尋的節點數量：			82		個

起點：	造橋收費站		時間：	13:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	13:00	1	彰化系統交流道	13:46
1	苗栗交流道	13:08	1	彰化交流道	13:50
1	銅鑼交流道	13:13	1	埔鹽系統交流道	13:55
1	三義交流道	13:19	1	員林交流道	13:57
1	泰安服務區	13:25	1	員林收費站	14:01
1	后里交流道	13:26	1	北斗交流道	14:03
1	后里收費站	13:27	1	西螺服務區	14:08
1	台中系統交流道	13:29	1	西螺交流道	14:09
1	豐原交流道	13:31	1	虎尾交流道	14:13
1	大雅交流道	13:35	1	斗南交流道	14:16
1	台中交流道	13:38	1	雲林系統交流道	14:18
1	南屯交流道	13:40	1	斗南收費站	14:20
1	王田交流道	13:44			
累計旅行時間：			80		分鐘
系統演算時間：			489		ms
已搜尋的節點數量：			82		個

起點：	造橋收費站		時間：	16:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	16:00	1	彰化系統交流道	16:45
1	苗栗交流道	16:08	1	彰化交流道	16:53
1	銅鑼交流道	16:13	1	埔鹽系統交流道	17:57
1	三義交流道	16:19	1	員林交流道	18:00
1	泰安服務區	16:25	1	員林收費站	18:04
1	后里交流道	16:26	1	北斗交流道	18:05
1	后里收費站	16:27	1	西螺服務區	18:11
1	台中系統交流道	16:29	1	西螺交流道	18:13
1	豐原交流道	16:31	1	虎尾交流道	18:16
1	大雅交流道	16:34	1	斗南交流道	18:20
1	台中交流道	16:37	1	雲林系統交流道	18:22
1	南屯交流道	16:39	1	斗南收費站	18:23
1	王田交流道	16:44			
累計旅行時間：			143		分鐘
系統演算時間：			690		ms
已搜尋的節點數量：			115		個

起點：	造橋收費站		時間：	16:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	16:00	1	彰化系統交流道	16:45
1	苗栗交流道	16:08	1	彰化交流道	16:53
1	銅鑼交流道	16:13	1	埔鹽系統交流道	17:57
1	三義交流道	16:19	1	員林交流道	18:00
1	泰安服務區	16:25	1	員林收費站	18:04
1	后里交流道	16:26	1	北斗交流道	18:05
1	后里收費站	16:27	1	西螺服務區	18:11
1	台中系統交流道	16:29	1	西螺交流道	18:13
1	豐原交流道	16:31	1	虎尾交流道	18:16
1	大雅交流道	16:34	1	斗南交流道	18:20
1	台中交流道	16:37	1	雲林系統交流道	18:22
1	南屯交流道	16:39	1	斗南收費站	18:23
1	王田交流道	16:44			
累計旅行時間：			143		分鐘
系統演算時間：			690		ms
已搜尋的節點數量：			115		個

起點：	造橋收費站		時間：	19:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	19:00	1	彰化系統交流道	19:45
1	苗栗交流道	19:08	1	彰化交流道	19:48
1	銅鑼交流道	19:13	1	埔鹽系統交流道	19:54
1	三義交流道	19:19	1	員林交流道	19:57
1	泰安服務區	19:25	1	員林收費站	20:01
1	后里交流道	19:26	1	北斗交流道	20:02
1	后里收費站	19:27	1	西螺服務區	20:08
1	台中系統交流道	19:29	1	西螺交流道	20:09
1	豐原交流道	19:31	1	虎尾交流道	20:12
1	大雅交流道	19:35	1	斗南交流道	20:16
1	台中交流道	19:38	1	雲林系統交流道	20:18
1	南屯交流道	19:40	1	斗南收費站	20:19
1	王田交流道	19:43			
累計旅行時間：			79		分鐘
系統演算時間：			488		ms
已搜尋的節點數量：			82		個

起點：	造橋收費站		時間：	19:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	19:00	1	彰化系統交流道	19:45
1	苗栗交流道	19:08	1	彰化交流道	19:48
1	銅鑼交流道	19:13	1	埔鹽系統交流道	19:54
1	三義交流道	19:19	1	員林交流道	19:57
1	泰安服務區	19:25	1	員林收費站	20:01
1	后里交流道	19:26	1	北斗交流道	20:02
1	后里收費站	19:27	1	西螺服務區	20:08
1	台中系統交流道	19:29	1	西螺交流道	20:09
1	豐原交流道	19:31	1	虎尾交流道	20:12
1	大雅交流道	19:35	1	斗南交流道	20:16
1	台中交流道	19:38	1	雲林系統交流道	20:18
1	南屯交流道	19:40	1	斗南收費站	20:19
1	王田交流道	19:43			
累計旅行時間：			79		分鐘
系統演算時間：			488		ms
已搜尋的節點數量：			82		個

(二)、 旅行路徑經過一個以下系統交流道：起點與訖點位於不同國道 詳細路徑演算結果

起點：	造橋收費站		時間：	07:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	07:00	3	快官交流道	07:49
1	苗栗交流道	07:08	3	烏日交流道	07:52
1	銅鑼交流道	07:14	3	中投交流道	07:53
1	三義交流道	07:20	3	霧峰交流道	07:54
1	泰安服務區	07:26	3	霧峰系統交流道	07:56
1	后里交流道	07:27	3	草屯交流道	07:58
1	后里收費站	07:28	3	中興系統交流道	08:01
1	台中系統交流道	07:30	3	中興交流道	08:02
1	豐原交流道	07:32	3	名間收費站	08:08
1	大雅交流道	07:35	3	名間交流道	08:09
1	台中交流道	07:38	3	竹山交流道	08:14
1	南屯交流道	07:39	3	斗六交流道	08:23
1	王田交流道	07:44	3	古坑系統交流道	08:29
1	彰化系統交流道	07:46	3	古坑收費站	08:32
3	彰化系統交流道	07:46			
累計旅行時間：			92		分鐘
系統演算時間：			541		ms
已搜尋的節點數量：			88		個

起點：	造橋收費站		時間：	07:00	
訖點：	古坑收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	07:00	3	快官交流道	07:49
1	苗栗交流道	07:08	3	烏日交流道	07:52
1	銅鑼交流道	07:14	3	中投交流道	07:53
1	三義交流道	07:20	3	霧峰交流道	07:54
1	泰安服務區	07:26	3	霧峰系統交流道	07:56
1	后里交流道	07:27	3	草屯交流道	07:58
1	后里收費站	07:28	3	中興系統交流道	08:01
1	台中系統交流道	07:30	3	中興交流道	08:02
1	豐原交流道	07:32	3	名間收費站	08:08
1	大雅交流道	07:35	3	名間交流道	08:09
1	台中交流道	07:38	3	竹山交流道	08:14
1	南屯交流道	07:39	3	斗六交流道	08:23
1	王田交流道	07:44	3	古坑系統交流道	08:29
1	彰化系統交流道	07:46	3	古坑收費站	08:32
3	彰化系統交流道	07:46			
累計旅行時間：			92		分鐘
系統演算時間：			289		ms
已搜尋的節點數量：			44		個

起點：	造橋收費站		時間：	10:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	10:00	3	快官交流道	10:52
1	苗栗交流道	10:08	3	烏日交流道	10:55
1	銅鑼交流道	10:15	3	中投交流道	10:56
1	三義交流道	10:21	3	霧峰交流道	10:57
1	泰安服務區	10:28	3	霧峰系統交流道	10:59
1	后里交流道	10:28	3	草屯交流道	11:01
1	后里收費站	10:29	3	中興系統交流道	11:04
1	台中系統交流道	10:31	3	中興交流道	11:05
1	豐原交流道	10:33	3	名間收費站	11:12
1	大雅交流道	10:38	3	名間交流道	11:13
1	台中交流道	10:40	3	竹山交流道	11:17
1	南屯交流道	10:42	3	斗六交流道	11:27
1	王田交流道	10:47	3	古坑系統交流道	11:32
1	彰化系統交流道	10:48	3	古坑收費站	11:35
3	彰化系統交流道	10:48			
累計旅行時間：			95	分鐘	
系統演算時間：			623	ms	
已搜尋的節點數量：			100	個	

起點：	造橋收費站		時間：	10:00	
訖點：	古坑收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	10:00	3	快官交流道	10:52
1	苗栗交流道	10:08	3	烏日交流道	10:55
1	銅鑼交流道	10:15	3	中投交流道	10:56
1	三義交流道	10:21	3	霧峰交流道	10:57
1	泰安服務區	10:28	3	霧峰系統交流道	10:59
1	后里交流道	10:28	3	草屯交流道	11:01
1	后里收費站	10:29	3	中興系統交流道	11:04
1	台中系統交流道	10:31	3	中興交流道	11:05
1	豐原交流道	10:33	3	名間收費站	11:12
1	大雅交流道	10:38	3	名間交流道	11:13
1	台中交流道	10:40	3	竹山交流道	11:17
1	南屯交流道	10:42	3	斗六交流道	11:27
1	王田交流道	10:47	3	古坑系統交流道	11:32
1	彰化系統交流道	10:48	3	古坑收費站	11:35
3	彰化系統交流道	10:48			
累計旅行時間：			95	分鐘	
系統演算時間：			290	ms	
已搜尋的節點數量：			46	個	

起點：	造橋收費站		時間：	13:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	13:00	3	快官交流道	13:50
1	苗栗交流道	13:08	3	烏日交流道	13:53
1	銅鑼交流道	13:13	3	中投交流道	13:54
1	三義交流道	13:19	3	霧峰交流道	13:55
1	泰安服務區	13:25	3	霧峰系統交流道	13:57
1	后里交流道	13:26	3	草屯交流道	13:59
1	后里收費站	13:27	3	中興系統交流道	14:02
1	台中系統交流道	13:29	3	中興交流道	14:03
1	豐原交流道	13:31	3	名間收費站	14:09
1	大雅交流道	13:35	3	名間交流道	14:10
1	台中交流道	13:38	3	竹山交流道	14:15
1	南屯交流道	13:40	3	斗六交流道	14:24
1	王田交流道	13:44	3	古坑系統交流道	14:30
1	彰化系統交流道	13:46	3	古坑收費站	14:32
3	彰化系統交流道	13:46			
累計旅行時間：			92		分鐘
系統演算時間：			623		ms
已搜尋的節點數量：			101		個

起點：	造橋收費站		時間：	13:00	
訖點：	古坑收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	13:00	3	快官交流道	13:50
1	苗栗交流道	13:08	3	烏日交流道	13:53
1	銅鑼交流道	13:13	3	中投交流道	13:54
1	三義交流道	13:19	3	霧峰交流道	13:56
1	泰安服務區	13:25	3	霧峰系統交流道	13:57
1	后里交流道	13:26	3	草屯交流道	13:59
1	后里收費站	13:27	3	中興系統交流道	14:02
1	台中系統交流道	13:29	3	中興交流道	14:03
1	豐原交流道	13:31	3	名間收費站	14:09
1	大雅交流道	13:35	3	名間交流道	14:10
1	台中交流道	13:38	3	竹山交流道	14:15
1	南屯交流道	13:40	3	斗六交流道	14:24
1	王田交流道	13:44	3	古坑系統交流道	14:30
1	彰化系統交流道	13:46	3	古坑收費站	14:32
3	彰化系統交流道	13:46			
累計旅行時間：			92		分鐘
系統演算時間：			277		ms
已搜尋的節點數量：			44		個

起點：	造橋收費站		時間：	16:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	16:00	3	快官交流道	16:49
1	苗栗交流道	16:08	3	烏日交流道	16:52
1	銅鑼交流道	16:13	3	中投交流道	16:53
1	三義交流道	16:19	3	霧峰交流道	16:55
1	泰安服務區	16:25	3	霧峰系統交流道	16:57
1	后里交流道	16:26	3	草屯交流道	16:59
1	后里收費站	16:27	3	中興系統交流道	17:02
1	台中系統交流道	16:29	3	中興交流道	17:03
1	豐原交流道	16:31	3	名間收費站	17:09
1	大雅交流道	16:34	3	名間交流道	17:11
1	台中交流道	16:37	3	竹山交流道	17:16
1	南屯交流道	16:39	3	斗六交流道	17:26
1	王田交流道	16:44	3	古坑系統交流道	17:31
1	彰化系統交流道	16:45	3	古坑收費站	17:34
3	彰化系統交流道	16:45			
累計旅行時間：			94	分鐘	
系統演算時間：			454	ms	
已搜尋的節點數量：			74	個	

起點：	造橋收費站		時間：	16:00	
訖點：	古坑收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	16:00	3	快官交流道	16:49
1	苗栗交流道	16:08	3	烏日交流道	16:52
1	銅鑼交流道	16:13	3	中投交流道	16:53
1	三義交流道	16:19	3	霧峰交流道	16:55
1	泰安服務區	16:25	3	霧峰系統交流道	16:57
1	后里交流道	16:26	3	草屯交流道	16:59
1	后里收費站	16:27	3	中興系統交流道	17:02
1	台中系統交流道	16:29	3	中興交流道	17:03
1	豐原交流道	16:31	3	名間收費站	17:09
1	大雅交流道	16:34	3	名間交流道	17:11
1	台中交流道	16:37	3	竹山交流道	17:16
1	南屯交流道	16:39	3	斗六交流道	17:26
1	王田交流道	16:44	3	古坑系統交流道	17:31
1	彰化系統交流道	16:45	3	古坑收費站	17:34
3	彰化系統交流道	16:45			
累計旅行時間：			94	分鐘	
系統演算時間：			231	ms	
已搜尋的節點數量：			35	個	

起點：	造橋收費站		時間：	19:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	19:00	3	快官交流道	19:49
1	苗栗交流道	19:08	3	烏日交流道	19:52
1	銅鑼交流道	19:13	3	中投交流道	19:53
1	三義交流道	19:19	3	霧峰交流道	19:55
1	泰安服務區	19:25	3	霧峰系統交流道	19:56
1	后里交流道	19:26	3	草屯交流道	19:58
1	后里收費站	19:27	3	中興系統交流道	20:01
1	台中系統交流道	19:29	3	中興交流道	20:02
1	豐原交流道	19:31	3	名間收費站	20:09
1	大雅交流道	19:35	3	名間交流道	20:10
1	台中交流道	19:38	3	竹山交流道	20:15
1	南屯交流道	19:40	3	斗六交流道	20:25
1	王田交流道	19:43	3	古坑系統交流道	20:31
1	彰化系統交流道	19:45	3	古坑收費站	20:33
3	彰化系統交流道	19:45			
累計旅行時間：			93	分鐘	
系統演算時間：			625	ms	
已搜尋的節點數量：			101	個	

起點：	造橋收費站		時間：	19:00	
訖點：	古坑收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	19:00	3	快官交流道	19:49
1	苗栗交流道	19:08	3	烏日交流道	19:52
1	銅鑼交流道	19:13	3	中投交流道	19:53
1	三義交流道	19:19	3	霧峰交流道	19:55
1	泰安服務區	19:25	3	霧峰系統交流道	19:56
1	后里交流道	19:26	3	草屯交流道	19:58
1	后里收費站	19:27	3	中興系統交流道	20:01
1	台中系統交流道	19:29	3	中興交流道	20:02
1	豐原交流道	19:31	3	名間收費站	20:09
1	大雅交流道	19:35	3	名間交流道	20:10
1	台中交流道	19:38	3	竹山交流道	20:15
1	南屯交流道	19:40	3	斗六交流道	20:25
1	王田交流道	19:43	3	古坑系統交流道	20:31
1	彰化系統交流道	19:45	3	古坑收費站	20:33
3	彰化系統交流道	19:45			
累計旅行時間：			93	分鐘	
系統演算時間：			283	ms	
已搜尋的節點數量：			44	個	

(三)、 旅行路徑經過一個以下系統交流道：起點與訖點位於不同國道且皆在系統交流道的北側或南側 詳細路徑演算結果

起點：	造橋收費站		時間：	07:00	
訖點：	大甲收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	07:00	1	南屯交流道	07:39
1	苗栗交流道	07:08	1	王田交流道	07:44
1	銅鑼交流道	07:14	1	彰化系統交流道	07:46
1	三義交流道	07:20	3	彰化系統交流道	07:46
1	泰安服務區	07:26	3	和美交流道	07:49
1	后里交流道	07:27	3	龍井交流道	07:52
1	后里收費站	07:28	3	沙鹿交流道	07:59
1	台中系統交流道	07:30	3	中港系統交流道	08:06
1	豐原交流道	07:32	3	大甲交流道	08:10
1	大雅交流道	07:35	3	大甲收費站	08:13
1	台中交流道	07:38			
累計旅行時間：			73		分鐘
系統演算時間：			434		ms
已搜尋的節點數量：			70		個

起點：	造橋收費站		時間：	07:00	
訖點：	大甲收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	07:00	1	南屯交流道	07:39
1	苗栗交流道	07:08	1	王田交流道	07:44
1	銅鑼交流道	07:14	1	彰化系統交流道	07:46
1	三義交流道	07:20	3	彰化系統交流道	07:46
1	泰安服務區	07:26	3	和美交流道	07:49
1	后里交流道	07:27	3	龍井交流道	07:52
1	后里收費站	07:28	3	沙鹿交流道	07:59
1	台中系統交流道	07:30	3	中港系統交流道	08:06
1	豐原交流道	07:32	3	大甲交流道	08:10
1	大雅交流道	07:35	3	大甲收費站	08:13
1	台中交流道	07:38			
累計旅行時間：			73		分鐘
系統演算時間：			257		ms
已搜尋的節點數量：			40		個

起點：	造橋收費站		時間：	10:00	
訖點：	大甲收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	10:00	1	南屯交流道	10:42
1	苗栗交流道	10:08	1	王田交流道	10:47
1	銅鑼交流道	10:15	1	彰化系統交流道	10:48
1	三義交流道	10:21	3	彰化系統交流道	10:48
1	泰安服務區	10:28	3	和美交流道	10:52
1	后里交流道	10:28	3	龍井交流道	10:55
1	后里收費站	10:29	3	沙鹿交流道	11:01
1	台中系統交流道	10:31	3	中港系統交流道	11:08
1	豐原交流道	10:33	3	大甲交流道	11:13
1	大雅交流道	10:38	3	大甲收費站	11:16
1	台中交流道	10:40			
累計旅行時間：			76		分鐘
系統演算時間：			466		ms
已搜尋的節點數量：			76		個

起點：	造橋收費站		時間：	10:00	
訖點：	大甲收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	10:00	1	南屯交流道	10:42
1	苗栗交流道	10:08	1	王田交流道	10:47
1	銅鑼交流道	10:15	1	彰化系統交流道	10:48
1	三義交流道	10:21	3	彰化系統交流道	10:48
1	泰安服務區	10:28	3	和美交流道	10:52
1	后里交流道	10:28	3	龍井交流道	10:55
1	后里收費站	10:29	3	沙鹿交流道	11:01
1	台中系統交流道	10:31	3	中港系統交流道	11:08
1	豐原交流道	10:33	3	大甲交流道	11:13
1	大雅交流道	10:38	3	大甲收費站	11:16
1	台中交流道	10:40			
累計旅行時間：			76		分鐘
系統演算時間：			264		ms
已搜尋的節點數量：			41		個

起點：	造橋收費站		時間：	13:00	
訖點：	大甲收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	13:00	1	南屯交流道	13:40
1	苗栗交流道	13:08	1	王田交流道	13:44
1	銅鑼交流道	13:13	1	彰化系統交流道	13:46
1	三義交流道	13:19	3	彰化系統交流道	13:46
1	泰安服務區	13:25	3	和美交流道	13:50
1	后里交流道	13:26	3	龍井交流道	13:53
1	后里收費站	13:27	3	沙鹿交流道	13:59
1	台中系統交流道	13:29	3	中港系統交流道	14:06
1	豐原交流道	13:31	3	大甲交流道	14:11
1	大雅交流道	13:35	3	大甲收費站	14:14
1	台中交流道	13:38			
累計旅行時間：			74		分鐘
系統演算時間：			477		ms
已搜尋的節點數量：			76		個

起點：	造橋收費站		時間：	13:00	
訖點：	大甲收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	13:00	1	南屯交流道	13:40
1	苗栗交流道	13:08	1	王田交流道	13:44
1	銅鑼交流道	13:13	1	彰化系統交流道	13:46
1	三義交流道	13:19	3	彰化系統交流道	13:46
1	泰安服務區	13:25	3	和美交流道	13:50
1	后里交流道	13:26	3	龍井交流道	13:53
1	后里收費站	13:27	3	沙鹿交流道	13:59
1	台中系統交流道	13:29	3	中港系統交流道	14:06
1	豐原交流道	13:31	3	大甲交流道	14:11
1	大雅交流道	13:35	3	大甲收費站	14:14
1	台中交流道	13:38			
累計旅行時間：			74		分鐘
系統演算時間：			260		ms
已搜尋的節點數量：			40		個

起點：	造橋收費站		時間：	16:00	
訖點：	大甲收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	16:00	1	南屯交流道	16:39
1	苗栗交流道	16:08	1	王田交流道	16:44
1	銅鑼交流道	16:13	1	彰化系統交流道	16:45
1	三義交流道	16:19	3	彰化系統交流道	16:45
1	泰安服務區	16:25	3	和美交流道	16:49
1	后里交流道	16:26	3	龍井交流道	16:52
1	后里收費站	16:27	3	沙鹿交流道	16:59
1	台中系統交流道	16:29	3	中港系統交流道	17:05
1	豐原交流道	16:31	3	大甲交流道	17:10
1	大雅交流道	16:34	3	大甲收費站	17:13
1	台中交流道	16:37			
累計旅行時間：			73		分鐘
系統演算時間：			388		ms
已搜尋的節點數量：			62		個

起點：	造橋收費站		時間：	16:00	
訖點：	大甲收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	16:00	1	南屯交流道	16:39
1	苗栗交流道	16:08	1	王田交流道	16:44
1	銅鑼交流道	16:13	1	彰化系統交流道	16:45
1	三義交流道	16:19	3	彰化系統交流道	16:45
1	泰安服務區	16:25	3	和美交流道	16:49
1	后里交流道	16:26	3	龍井交流道	16:52
1	后里收費站	16:27	3	沙鹿交流道	16:59
1	台中系統交流道	16:29	3	中港系統交流道	17:05
1	豐原交流道	16:31	3	大甲交流道	17:10
1	大雅交流道	16:34	3	大甲收費站	17:13
1	台中交流道	16:37			
累計旅行時間：			73		分鐘
系統演算時間：			251		ms
已搜尋的節點數量：			39		個

起點：	造橋收費站		時間：	19:00	
訖點：	大甲收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	19:00	1	南屯交流道	19:40
1	苗栗交流道	19:08	1	王田交流道	19:43
1	銅鑼交流道	19:13	1	彰化系統交流道	19:45
1	三義交流道	19:19	3	彰化系統交流道	19:45
1	泰安服務區	19:25	3	和美交流道	19:49
1	后里交流道	19:26	3	龍井交流道	19:52
1	后里收費站	19:27	3	沙鹿交流道	19:59
1	台中系統交流道	19:29	3	中港系統交流道	20:06
1	豐原交流道	19:31	3	大甲交流道	20:12
1	大雅交流道	19:35	3	大甲收費站	20:14
1	台中交流道	19:38			
累計旅行時間：			74		分鐘
系統演算時間：			470		ms
已搜尋的節點數量：			75		個

起點：	造橋收費站		時間：	19:00	
訖點：	大甲收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	造橋收費站	19:00	1	南屯交流道	19:40
1	苗栗交流道	19:08	1	王田交流道	19:43
1	銅鑼交流道	19:13	1	彰化系統交流道	19:45
1	三義交流道	19:19	3	彰化系統交流道	19:45
1	泰安服務區	19:25	3	和美交流道	19:49
1	后里交流道	19:26	3	龍井交流道	19:52
1	后里收費站	19:27	3	沙鹿交流道	19:59
1	台中系統交流道	19:29	3	中港系統交流道	20:06
1	豐原交流道	19:31	3	大甲交流道	20:12
1	大雅交流道	19:35	3	大甲收費站	20:14
1	台中交流道	19:38			
累計旅行時間：			74		分鐘
系統演算時間：			263		ms
已搜尋的節點數量：			41		個

(四)、 旅行路徑經過兩個以上系統交流道：起訖點位於相同國道 詳細路徑演算結果

起點：	楊梅收費站		時間：	07:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	07:00	1	大雅交流道	08:10
1	湖口交流道	07:06	1	台中交流道	08:12
1	湖口服務區	07:08	1	南屯交流道	08:14
1	竹北交流道	07:11	1	王田交流道	08:18
1	新竹交流道	07:14	1	彰化系統交流道	08:20
1	新竹系統交流道	07:16	1	彰化交流道	08:23
1	頭份交流道	07:23	1	埔鹽系統交流道	08:28
1	造橋收費站	07:27	1	員林交流道	08:31
1	苗栗交流道	07:35	1	員林收費站	08:35
1	銅鑼交流道	07:48	1	北斗交流道	08:36
1	三義交流道	07:54	1	西螺服務區	08:42
1	泰安服務區	08:00	1	西螺交流道	08:43
1	后里交流道	08:01	1	虎尾交流道	08:47
1	后里收費站	08:02	1	斗南交流道	08:50
1	台中系統交流道	08:04	1	雲林系統交流道	08:52
1	豐原交流道	08:06	1	斗南收費站	08:54
累計旅行時間：				114	分鐘
系統演算時間：				612	ms
已搜尋的節點數量：				102	個

起點：	楊梅收費站		時間：	07:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	07:00	1	大雅交流道	08:10
1	湖口交流道	07:06	1	台中交流道	08:12
1	湖口服務區	07:08	1	南屯交流道	08:14
1	竹北交流道	07:11	1	王田交流道	08:18
1	新竹交流道	07:14	1	彰化系統交流道	08:20
1	新竹系統交流道	07:16	1	彰化交流道	08:23
1	頭份交流道	07:23	1	埔鹽系統交流道	08:28
1	造橋收費站	07:27	1	員林交流道	08:31
1	苗栗交流道	07:36	1	員林收費站	08:35
1	銅鑼交流道	07:48	1	北斗交流道	08:36
1	三義交流道	07:54	1	西螺服務區	08:42
1	泰安服務區	08:00	1	西螺交流道	08:43
1	后里交流道	08:01	1	虎尾交流道	08:47
1	后里收費站	08:02	1	斗南交流道	08:50
1	台中系統交流道	08:04	1	雲林系統交流道	08:52
1	豐原交流道	08:06	1	斗南收費站	08:54
累計旅行時間：				114	分鐘
系統演算時間：				298	ms
已搜尋的節點數量：				47	個

起點：	楊梅收費站		時間：	10:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	10:00	1	大雅交流道	11:03
1	湖口交流道	10:06	1	台中交流道	11:05
1	湖口服務區	10:08	1	南屯交流道	11:07
1	竹北交流道	10:11	1	王田交流道	11:12
1	新竹交流道	10:14	1	彰化系統交流道	11:14
1	新竹系統交流道	10:16	1	彰化交流道	11:17
1	頭份交流道	10:23	1	埔鹽系統交流道	11:22
1	造橋收費站	10:27	1	員林交流道	11:25
1	苗栗交流道	10:36	1	員林收費站	11:29
1	銅鑼交流道	10:41	1	北斗交流道	11:30
1	三義交流道	10:47	1	西螺服務區	11:36
1	泰安服務區	10:53	1	西螺交流道	11:37
1	后里交流道	10:54	1	虎尾交流道	11:41
1	后里收費站	10:55	1	斗南交流道	11:44
1	台中系統交流道	10:56	1	雲林系統交流道	11:46
1	豐原交流道	10:58	1	斗南收費站	11:48
累計旅行時間：			108	分鐘	
系統演算時間：			613	ms	
已搜尋的節點數量：			102	個	

起點：	楊梅收費站		時間：	10:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	10:00	1	大雅交流道	11:03
1	湖口交流道	10:06	1	台中交流道	11:05
1	湖口服務區	10:08	1	南屯交流道	11:07
1	竹北交流道	10:11	1	王田交流道	11:12
1	新竹交流道	10:14	1	彰化系統交流道	11:14
1	新竹系統交流道	10:16	1	彰化交流道	11:17
1	頭份交流道	10:23	1	埔鹽系統交流道	11:22
1	造橋收費站	10:27	1	員林交流道	11:25
1	苗栗交流道	10:36	1	員林收費站	11:29
1	銅鑼交流道	10:41	1	北斗交流道	11:30
1	三義交流道	10:47	1	西螺服務區	11:36
1	泰安服務區	10:53	1	西螺交流道	11:37
1	后里交流道	10:54	1	虎尾交流道	11:41
1	后里收費站	10:55	1	斗南交流道	11:44
1	台中系統交流道	10:56	1	雲林系統交流道	11:46
1	豐原交流道	10:58	1	斗南收費站	11:48
累計旅行時間：			108	分鐘	
系統演算時間：			283	ms	
已搜尋的節點數量：			42	個	

起點：	楊梅收費站		時間：	13:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	13:00	1	大雅交流道	14:04
1	湖口交流道	13:06	1	台中交流道	14:06
1	湖口服務區	13:08	1	南屯交流道	14:08
1	竹北交流道	13:11	1	王田交流道	14:13
1	新竹交流道	13:14	1	彰化系統交流道	14:15
1	新竹系統交流道	13:16	1	彰化交流道	14:18
1	頭份交流道	13:23	1	埔鹽系統交流道	14:25
1	造橋收費站	13:27	1	員林交流道	14:27
1	苗栗交流道	13:36	1	員林收費站	14:31
1	銅鑼交流道	13:41	1	北斗交流道	14:33
1	三義交流道	13:47	1	西螺服務區	14:38
1	泰安服務區	13:53	1	西螺交流道	14:39
1	后里交流道	13:54	1	虎尾交流道	14:43
1	后里收費站	13:55	1	斗南交流道	14:46
1	台中系統交流道	13:57	1	雲林系統交流道	14:48
1	豐原交流道	13:59	1	斗南收費站	14:50
累計旅行時間：			110	分鐘	
系統演算時間：			605	ms	
已搜尋的節點數量：			102	個	

起點：	楊梅收費站		時間：	13:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	13:00	1	大雅交流道	14:04
1	湖口交流道	13:06	1	台中交流道	14:06
1	湖口服務區	13:08	1	南屯交流道	14:08
1	竹北交流道	13:11	1	王田交流道	14:13
1	新竹交流道	13:14	1	彰化系統交流道	14:15
1	新竹系統交流道	13:16	1	彰化交流道	14:18
1	頭份交流道	13:23	1	埔鹽系統交流道	14:25
1	造橋收費站	13:27	1	員林交流道	14:27
1	苗栗交流道	13:36	1	員林收費站	14:31
1	銅鑼交流道	13:41	1	北斗交流道	14:33
1	三義交流道	13:47	1	西螺服務區	14:38
1	泰安服務區	13:53	1	西螺交流道	14:39
1	后里交流道	13:54	1	虎尾交流道	14:43
1	后里收費站	13:55	1	斗南交流道	14:46
1	台中系統交流道	13:57	1	雲林系統交流道	14:48
1	豐原交流道	13:59	1	斗南收費站	14:50
累計旅行時間：			110	分鐘	
系統演算時間：			269	ms	
已搜尋的節點數量：			43	個	

起點：	楊梅收費站		時間：	16:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	16:00	1	大雅交流道	17:02
1	湖口交流道	16:06	1	台中交流道	17:05
1	湖口服務區	16:08	1	南屯交流道	17:07
1	竹北交流道	16:11	1	王田交流道	17:12
1	新竹交流道	16:14	1	彰化系統交流道	17:13
1	新竹系統交流道	16:16	1	彰化交流道	17:43
1	頭份交流道	16:23	1	埔鹽系統交流道	17:53
1	造橋收費站	16:27	1	員林交流道	17:56
1	苗栗交流道	16:36	1	員林收費站	18:00
1	銅鑼交流道	16:41	1	北斗交流道	18:01
1	三義交流道	16:47	1	西螺服務區	18:07
1	泰安服務區	16:53	1	西螺交流道	18:08
1	后里交流道	16:53	1	虎尾交流道	18:12
1	后里收費站	16:55	1	斗南交流道	18:15
1	台中系統交流道	16:56	1	雲林系統交流道	18:17
1	豐原交流道	16:59	1	斗南收費站	18:19
累計旅行時間：			139		分鐘
系統演算時間：			655		ms
已搜尋的節點數量：			109		個

起點：	楊梅收費站		時間：	16:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	16:00	1	大雅交流道	17:02
1	湖口交流道	16:06	1	台中交流道	17:05
1	湖口服務區	16:08	1	南屯交流道	17:07
1	竹北交流道	16:11	1	王田交流道	17:12
1	新竹交流道	16:14	1	彰化系統交流道	17:13
1	新竹系統交流道	16:16	1	彰化交流道	17:43
1	頭份交流道	16:23	1	埔鹽系統交流道	17:53
1	造橋收費站	16:27	1	員林交流道	17:56
1	苗栗交流道	16:36	1	員林收費站	18:00
1	銅鑼交流道	16:41	1	北斗交流道	18:01
1	三義交流道	16:47	1	西螺服務區	18:07
1	泰安服務區	16:53	1	西螺交流道	18:08
1	后里交流道	16:53	1	虎尾交流道	18:12
1	后里收費站	16:55	1	斗南交流道	18:15
1	台中系統交流道	16:56	1	雲林系統交流道	18:17
1	豐原交流道	16:59	1	斗南收費站	18:19
累計旅行時間：			139		分鐘
系統演算時間：			442		ms
已搜尋的節點數量：			72		個

起點：	楊梅收費站		時間：	19:00	
訖點：	斗南收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	19:00	1	大雅交流道	20:05
1	湖口交流道	19:06	1	台中交流道	20:08
1	湖口服務區	19:08	1	南屯交流道	20:10
1	竹北交流道	19:11	1	王田交流道	20:15
1	新竹交流道	19:14	1	彰化系統交流道	20:16
1	新竹系統交流道	19:19	1	彰化交流道	20:20
1	頭份交流道	19:26	1	埔鹽系統交流道	20:26
1	造橋收費站	19:30	1	員林交流道	20:28
1	苗栗交流道	19:38	1	員林收費站	20:32
1	銅鑼交流道	19:43	1	北斗交流道	20:34
1	三義交流道	19:50	1	西螺服務區	20:39
1	泰安服務區	19:56	1	西螺交流道	20:40
1	后里交流道	19:56	1	虎尾交流道	20:44
1	后里收費站	19:58	1	斗南交流道	20:47
1	台中系統交流道	19:59	1	雲林系統交流道	20:49
1	豐原交流道	20:01	1	斗南收費站	20:51
累計旅行時間：				111	分鐘
系統演算時間：				611	ms
已搜尋的節點數量：				102	個

起點：	楊梅收費站		時間：	19:00	
訖點：	斗南收費站		演算法：	A* Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	19:00	1	大雅交流道	20:05
1	湖口交流道	19:06	1	台中交流道	20:08
1	湖口服務區	19:08	1	南屯交流道	20:10
1	竹北交流道	19:11	1	王田交流道	20:15
1	新竹交流道	19:14	1	彰化系統交流道	20:16
1	新竹系統交流道	19:19	1	彰化交流道	20:20
1	頭份交流道	19:26	1	埔鹽系統交流道	20:26
1	造橋收費站	19:30	1	員林交流道	20:28
1	苗栗交流道	19:38	1	員林收費站	20:32
1	銅鑼交流道	19:43	1	北斗交流道	20:34
1	三義交流道	19:50	1	西螺服務區	20:39
1	泰安服務區	19:56	1	西螺交流道	20:40
1	后里交流道	19:56	1	虎尾交流道	20:44
1	后里收費站	19:58	1	斗南交流道	20:47
1	台中系統交流道	19:59	1	雲林系統交流道	20:49
1	豐原交流道	20:01	1	斗南收費站	20:51
累計旅行時間：				111	分鐘
系統演算時間：				269	ms
已搜尋的節點數量：				43	個

(五)、旅行路徑經過兩個以上系統交流道：起訖點位於不同國道 詳細路徑演算結果

起點：	楊梅收費站		時間：	07:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	07:00	1	南屯交流道	08:14
1	湖口交流道	07:06	1	王田交流道	08:18
1	湖口服務區	07:08	1	彰化系統交流道	08:20
1	竹北交流道	07:11	3	彰化系統交流道	08:20
1	新竹交流道	07:14	3	快官交流道	08:23
1	新竹系統交流道	07:16	3	烏日交流道	08:26
1	頭份交流道	07:23	3	中投交流道	08:28
1	造橋收費站	07:27	3	霧峰交流道	08:29
1	苗栗交流道	07:36	3	霧峰系統交流道	08:30
1	銅鑼交流道	07:48	3	草屯交流道	08:32
1	三義交流道	07:54	3	中興系統交流道	08:35
1	泰安服務區	08:00	3	中興交流道	08:36
1	后里交流道	08:01	3	名間收費站	08:42
1	后里收費站	08:02	3	名間交流道	08:44
1	台中系統交流道	08:04	3	竹山交流道	08:48
1	豐原交流道	08:06	3	斗六交流道	08:58
1	大雅交流道	08:10	3	古坑系統交流道	09:03
1	台中交流道	08:12	3	古坑收費站	09:06
累計旅行時間：			126	分鐘	
系統演算時間：			634	ms	
已搜尋的節點數量：			107	個	

起點：	楊梅收費站		時間：	07:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	07:00	1	南屯交流道	08:14
1	湖口交流道	07:06	1	王田交流道	08:18
1	湖口服務區	07:08	1	彰化系統交流道	08:20
1	竹北交流道	07:11	3	彰化系統交流道	08:20
1	新竹交流道	07:14	3	快官交流道	08:23
1	新竹系統交流道	07:16	3	烏日交流道	08:26
1	頭份交流道	07:23	3	中投交流道	08:28
1	造橋收費站	07:27	3	霧峰交流道	08:29
1	苗栗交流道	07:35	3	霧峰系統交流道	08:30
1	銅鑼交流道	07:48	3	草屯交流道	08:32
1	三義交流道	07:54	3	中興系統交流道	08:35
1	泰安服務區	08:00	3	中興交流道	08:36
1	后里交流道	08:01	3	名間收費站	08:42
1	后里收費站	08:02	3	名間交流道	08:44
1	台中系統交流道	08:04	3	竹山交流道	08:48
1	豐原交流道	08:06	3	斗六交流道	08:58
1	大雅交流道	08:10	3	古坑系統交流道	09:03
1	台中交流道	08:12	3	古坑收費站	09:06
累計旅行時間：			126	分鐘	
系統演算時間：			439	ms	
已搜尋的節點數量：			70	個	

起點：	楊梅收費站		時間：	10:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	10:00	1	南屯交流道	11:07
1	湖口交流道	10:06	1	王田交流道	11:12
1	湖口服務區	10:08	1	彰化系統交流道	11:14
1	竹北交流道	10:11	3	彰化系統交流道	11:14
1	新竹交流道	10:14	3	快官交流道	11:17
1	新竹系統交流道	10:16	3	烏日交流道	11:20
1	頭份交流道	10:23	3	中投交流道	11:21
1	造橋收費站	10:27	3	霧峰交流道	11:23
1	苗栗交流道	10:36	3	霧峰系統交流道	11:24
1	銅鑼交流道	10:41	3	草屯交流道	11:26
1	三義交流道	10:47	3	中興系統交流道	11:29
1	泰安服務區	10:53	3	中興交流道	11:30
1	后里交流道	10:54	3	名間收費站	11:37
1	后里收費站	10:55	3	名間交流道	11:38
1	台中系統交流道	10:56	3	竹山交流道	11:42
1	豐原交流道	10:58	3	斗六交流道	11:51
1	大雅交流道	11:03	3	古坑系統交流道	11:57
1	台中交流道	11:05	3	古坑收費站	11:59
累計旅行時間：				119	分鐘
系統演算時間：				624	ms
已搜尋的節點數量：				107	個

起點：	楊梅收費站		時間：	10:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	10:00	1	南屯交流道	11:07
1	湖口交流道	10:06	1	王田交流道	11:12
1	湖口服務區	10:08	1	彰化系統交流道	11:14
1	竹北交流道	10:11	3	彰化系統交流道	11:14
1	新竹交流道	10:14	3	快官交流道	11:17
1	新竹系統交流道	10:16	3	烏日交流道	11:20
1	頭份交流道	10:23	3	中投交流道	11:21
1	造橋收費站	10:27	3	霧峰交流道	11:23
1	苗栗交流道	10:36	3	霧峰系統交流道	11:24
1	銅鑼交流道	10:41	3	草屯交流道	11:26
1	三義交流道	10:47	3	中興系統交流道	11:29
1	泰安服務區	10:53	3	中興交流道	11:30
1	后里交流道	10:54	3	名間收費站	11:37
1	后里收費站	10:55	3	名間交流道	11:38
1	台中系統交流道	10:56	3	竹山交流道	11:42
1	豐原交流道	10:58	3	斗六交流道	11:51
1	大雅交流道	11:03	3	古坑系統交流道	11:57
1	台中交流道	11:05	3	古坑收費站	11:59
累計旅行時間：				119	分鐘
系統演算時間：				467	ms
已搜尋的節點數量：				67	個

起點：	楊梅收費站		時間：	13:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	13:00	1	南屯交流道	14:08
1	湖口交流道	13:06	1	王田交流道	14:13
1	湖口服務區	13:08	1	彰化系統交流道	14:15
1	竹北交流道	13:11	3	彰化系統交流道	14:15
1	新竹交流道	13:14	3	快官交流道	14:19
1	新竹系統交流道	13:16	3	烏日交流道	14:22
1	頭份交流道	13:23	3	中投交流道	14:23
1	造橋收費站	13:27	3	霧峰交流道	14:25
1	苗栗交流道	13:36	3	霧峰系統交流道	14:26
1	銅鑼交流道	13:41	3	草屯交流道	14:28
1	三義交流道	13:47	3	中興系統交流道	14:31
1	泰安服務區	13:53	3	中興交流道	14:32
1	后里交流道	13:54	3	名間收費站	14:38
1	后里收費站	13:55	3	名間交流道	14:39
1	台中系統交流道	13:57	3	竹山交流道	14:44
1	豐原交流道	13:59	3	斗六交流道	14:53
1	大雅交流道	14:04	3	古坑系統交流道	14:59
1	台中交流道	14:06	3	古坑收費站	15:01
累計旅行時間：			121		分鐘
系統演算時間：			660		ms
已搜尋的節點數量：			107		個

起點：	楊梅收費站		時間：	13:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	13:00	1	南屯交流道	14:08
1	湖口交流道	13:06	1	王田交流道	14:13
1	湖口服務區	13:08	1	彰化系統交流道	14:15
1	竹北交流道	13:11	3	彰化系統交流道	14:15
1	新竹交流道	13:14	3	快官交流道	14:19
1	新竹系統交流道	13:16	3	烏日交流道	14:22
1	頭份交流道	13:23	3	中投交流道	14:23
1	造橋收費站	13:27	3	霧峰交流道	14:25
1	苗栗交流道	13:36	3	霧峰系統交流道	14:26
1	銅鑼交流道	13:41	3	草屯交流道	14:28
1	三義交流道	13:47	3	中興系統交流道	14:31
1	泰安服務區	13:53	3	中興交流道	14:32
1	后里交流道	13:54	3	名間收費站	14:38
1	后里收費站	13:55	3	名間交流道	14:39
1	台中系統交流道	13:57	3	竹山交流道	14:44
1	豐原交流道	13:59	3	斗六交流道	14:53
1	大雅交流道	14:04	3	古坑系統交流道	14:59
1	台中交流道	14:06	3	古坑收費站	15:01
累計旅行時間：			121		分鐘
系統演算時間：			419		ms
已搜尋的節點數量：			69		個

起點：	楊梅收費站		時間：	16:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	16:00	1	南屯交流道	17:07
1	湖口交流道	16:06	1	王田交流道	17:12
1	湖口服務區	16:08	1	彰化系統交流道	17:13
1	竹北交流道	16:11	3	彰化系統交流道	17:13
1	新竹交流道	16:14	3	快官交流道	17:17
1	新竹系統交流道	16:16	3	烏日交流道	17:20
1	頭份交流道	16:23	3	中投交流道	17:21
1	造橋收費站	16:27	3	霧峰交流道	17:23
1	苗栗交流道	16:36	3	霧峰系統交流道	17:25
1	銅鑼交流道	16:41	3	草屯交流道	17:27
1	三義交流道	16:47	3	中興系統交流道	17:30
1	泰安服務區	16:53	3	中興交流道	17:31
1	后里交流道	16:53	3	名間收費站	17:37
1	后里收費站	16:55	3	名間交流道	17:39
1	台中系統交流道	16:56	3	竹山交流道	17:44
1	豐原交流道	16:59	3	斗六交流道	17:54
1	大雅交流道	17:02	3	古坑系統交流道	17:59
1	台中交流道	17:05	3	古坑收費站	18:01
累計旅行時間：				121	分鐘
系統演算時間：				588	ms
已搜尋的節點數量：				99	個

起點：	楊梅收費站		時間：	16:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	16:00	1	南屯交流道	17:07
1	湖口交流道	16:06	1	王田交流道	17:12
1	湖口服務區	16:08	1	彰化系統交流道	17:13
1	竹北交流道	16:11	3	彰化系統交流道	17:13
1	新竹交流道	16:14	3	快官交流道	17:17
1	新竹系統交流道	16:16	3	烏日交流道	17:20
1	頭份交流道	16:23	3	中投交流道	17:21
1	造橋收費站	16:27	3	霧峰交流道	17:23
1	苗栗交流道	16:36	3	霧峰系統交流道	17:25
1	銅鑼交流道	16:41	3	草屯交流道	17:27
1	三義交流道	16:47	3	中興系統交流道	17:30
1	泰安服務區	16:53	3	中興交流道	17:31
1	后里交流道	16:53	3	名間收費站	17:37
1	后里收費站	16:55	3	名間交流道	17:39
1	台中系統交流道	16:56	3	竹山交流道	17:44
1	豐原交流道	16:59	3	斗六交流道	17:54
1	大雅交流道	17:02	3	古坑系統交流道	17:59
1	台中交流道	17:05	3	古坑收費站	18:01
累計旅行時間：				121	分鐘
系統演算時間：				322	ms
已搜尋的節點數量：				54	個

起點：	楊梅收費站		時間：	19:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	19:00	1	南屯交流道	20:10
1	湖口交流道	19:06	1	王田交流道	20:15
1	湖口服務區	19:08	1	彰化系統交流道	20:16
1	竹北交流道	19:11	3	彰化系統交流道	20:16
1	新竹交流道	19:14	3	快官交流道	20:20
1	新竹系統交流道	19:19	3	烏日交流道	20:23
1	頭份交流道	19:26	3	中投交流道	20:24
1	造橋收費站	19:30	3	霧峰交流道	20:26
1	苗栗交流道	19:38	3	霧峰系統交流道	20:28
1	銅鑼交流道	19:43	3	草屯交流道	20:29
1	三義交流道	19:50	3	中興系統交流道	20:32
1	泰安服務區	19:56	3	中興交流道	20:33
1	后里交流道	19:56	3	名間收費站	20:40
1	后里收費站	19:58	3	名間交流道	20:41
1	台中系統交流道	19:59	3	竹山交流道	20:46
1	豐原交流道	20:01	3	斗六交流道	20:56
1	大雅交流道	20:05	3	古坑系統交流道	21:01
1	台中交流道	20:08	3	古坑收費站	21:03
累計旅行時間：			123		分鐘
系統演算時間：			637		ms
已搜尋的節點數量：			107		個

起點：	楊梅收費站		時間：	19:00	
訖點：	古坑收費站		演算法：	Dijkstra's Algorithm	
國道編號	交流道名稱	抵達時間	國道編號	交流道名稱	抵達時間
1	楊梅收費站	19:00	1	南屯交流道	20:10
1	湖口交流道	19:06	1	王田交流道	20:15
1	湖口服務區	19:08	1	彰化系統交流道	20:16
1	竹北交流道	19:11	3	彰化系統交流道	20:16
1	新竹交流道	19:14	3	快官交流道	20:20
1	新竹系統交流道	19:19	3	烏日交流道	20:23
1	頭份交流道	19:25	3	中投交流道	20:24
1	造橋收費站	19:30	3	霧峰交流道	20:26
1	苗栗交流道	19:38	3	霧峰系統交流道	20:28
1	銅鑼交流道	19:43	3	草屯交流道	20:29
1	三義交流道	19:50	3	中興系統交流道	20:32
1	泰安服務區	19:56	3	中興交流道	20:33
1	后里交流道	19:56	3	名間收費站	20:40
1	后里收費站	19:58	3	名間交流道	20:41
1	台中系統交流道	19:59	3	竹山交流道	20:46
1	豐原交流道	20:01	3	斗六交流道	20:56
1	大雅交流道	20:05	3	古坑系統交流道	21:01
1	台中交流道	20:08	3	古坑收費站	21:03
累計旅行時間：			123		分鐘
系統演算時間：			411		ms
已搜尋的節點數量：			68		個