國 立 交 通 大 學

資訊科學與工程研究所

碩 士 論 文

智慧型交通號誌燈控制研究與分析

Researching and Analyzing Intelligent Traffic Signals Control

研究生：李育緯

指導教授：王協源　教授

中華民國一百年九月

智慧型交通號誌燈控制研究與分析

Researching and Analyzing Intelligent Traffic Signals Control

研究生：李育緯　　Student：Yu-Wei Lee

指導教授：王協源　　Advisor：Shie-Yuan Wang

國立交通大學

資訊科學與資訊工程研究所

碩士論文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

September 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年九月

智慧型交通號誌燈控制研究與分析

學生：李育緯　　指導教授：王協源　教授

國立交通大學　　資訊科學與工程研究所　　碩士班

摘　要

在現今各大城市中，壅塞的交通一直是亟欲解決的問題，而隨著資訊產業的快速發展，智慧型交通運輸的研究漸趨熱門，其中交通號誌燈的控制是智慧型交通運輸中重要的一環。交通號誌燈控制的研究分為兩個種類，一種是固定時間式的交通號誌燈控制，也就是事先幫每個交通號誌燈安排好紅綠燈的循環周期，藉此達到疏通交通阻塞的目的。而另一種是變動時間式的交通號誌燈控制，藉由動態的評估車況來配給紅綠燈的時間，以應付隨時變化的交通狀況。

在本論文的研究中，我們選擇研究變動時間式的交通號誌控制，我們會設計不同的車流拓樸（traffic pattern)、不同的時間間隔、以及不同的車流資訊蒐集程度，來深入的研究變動時間式的交通號誌控制演算法的運作情形，並觀察演算法在不同情境下的運作情形，然後提出改善的建議，透過模擬案例實驗證實，我們的改良確實有效。

Researching and Analyzing Intelligent
Traffic Signals Control

Student：Yu-Wei Lee Advisors：Dr. Shie-Yuan Wang

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

In today's major cities, traffic congestion has been a severe problem. With the rapid development of information industry, intelligent transportation system (ITS) research is becoming more and more popular. Traffic signals control is getting well concerned among ITS. Traffic signals control's study is divided into two categories. One is fixed time-based traffic signals control, that is, arranges the red-green cycle for each traffic signal to solve the traffic congestion. The other is dynamic time-based traffic signals control, which allocates the traffic signals by evaluating traffic flows dynamically to cope with the varied traffic conditions.

In our thesis, we choose a dynamic time-based traffic signals control mechanism to study its behaviors and characteristics under various kind of traffic patterns, time intervals, and collection of car information to get well understanding on the mechanism. We also propose our improvements on this algorithm and verify our improvements using NCTUns network simulator. The results show that our improvements perform better than the original algorithm.

致謝

在這兩年的研究所生活，首先要感謝恩師王協源教授對我的指導與照顧，在王協源教授的實驗室底下做研究，是非常正確的選擇，不僅有豐富的實作經驗，更在每次的開會討論中，激盪出許多研究的靈感。

感謝口試委員林華君教授、吳曉光教授以及黃仁竑教授特地撥冗前來交通大學，聽取我們的論文報告並加以指導，在口試委員們的建議下，修正不足之處，讓這篇論文更加完善。

感謝實驗室的學長學弟們，在平常的討論中讓論文的架構慢慢成形最後完整，在 NSL 實驗室中與大家相處得相當愉快，也一起解決許多研究的難題，在這段日子我充分的感受到團隊合作的重要，有了你們讓我的研究生活更加充實。

最後，要感謝我的爺爺奶奶以及父母親的關懷與照顧，因為有你們的殷殷期盼，才是我度過最艱困的時光的動力，有了你們的包容與支持，才能讓我無後顧之憂的完成碩士學位。

目　　錄

表目錄

圖目錄

一、緒　論

1.1　研究動機

由於工業化與人口的快速增長，全世界每年的汽車產量快速提升，而因為汽車的大量生產、人民的生活水準提高，汽車已經是家家戶戶必備的交通工具，在上下班的顛峰時段，大量的汽車湧上街頭，隨之而來的交通阻塞、空氣汙染，已經是各大都市亟欲解決的問題。

而由於現在資訊產業的發達，以及網路資源越加充裕的環境，造就了智慧型交通運輸的興起，藉由車子定期的更新資訊，或者是路旁的偵測器監控道路環境，都可以讓管理者確切的掌握都市中的車流狀況，而掌握了車流狀況，管理者就可以針對壅塞的路段想出解決方案去舒緩交通，而其中有一種舒緩交通的方式，就是有智慧的調整交通號誌燈，以達到舒緩交通的效果。

智慧型交通號誌燈的控制，通常會分為固定時間式的調控與變動時間式的調控，在本論文中，我們會深入探討變動時間式的智慧型交通號誌燈控制的演算法，還有變動時間式的演算法中，不同的時間間隔對演算法效能的影響，之後我們還會針對變動時間式的智慧型控制演算法中不足的地方加以改良，並且透過模擬的方式驗證我們改良後的演算法在減少車輛行車延遲以及對各車輛的公平性上確實有較佳的表現。

1.2 研究目的與貢獻

1.2.1 研究目的

在本論文的研究中，我們希望對變動時間式的智慧型交通號誌燈控制演算法進行完整的評估，以了解在各種不同的情境下：包含不同的車流拓樸環境、不同的車流密度、不同的時間間隔、以及在不同的車流資訊蒐集完整度的情況下，該演算法對於減少行車延遲的效能，以作為後續分析及改良演算法的研究基礎。

本論文的研究目的是希望研究智慧型交通號誌燈控制演算法，並且分析演算法在各種不同的情境下的效能，但是如果在真實世界上做研究，必須花費大量的時間與成本才能建造出較具規模的環境，並且需要花費可觀的時間才能完成相關的研究實驗。相對的，採用模擬真實世界路網與車流環境的模擬器，除了可以降低真實環境硬體設備架設的經費與複雜度外，更可以提升實驗完成的效率，並且正確的完成各種模擬案例。因此在本論文的研究中，我們計畫採用模擬的方式，使用 NCTUns 這個網路與車流模擬平台進行實驗，並且將智慧型控制演算法移至 NCTUns 上進行模擬以評估演算法效能。

在本論文中，我們除了觀察與研究智慧型交通號誌燈控制演算法的效能之外，經由對演算法模擬結果的分析，我們對原設計的智慧型交通號誌控制演算法進行改良，並且在不同的模擬案例中，比較改良前後的演算法對於行車延遲與公平性的改善效果，最後歸納分析實驗的結果，證明我們的改善確實提升了智慧型交通號誌燈控制演算法的效能，並將智慧型交通號誌控制演算法的模擬結果進行整理與分析討論。

1.2.2 主要貢獻

本論文的主要研究貢獻包含：

1. 設計各種不同的車流拓樸與情境去分析智慧型交通號誌燈控制的演算法效能，並且深入探討不同的時間間隔對動態時間式演算法的效能影響。

2. 在 NCTUns 模擬器中開發了以 Landmark 為目的地的車流模型，使車流路網的模擬更貼近於現實。

3. 經由分析與檢視原設計的演算法，並針對其不足的地方進行改良，以減少原本演算法的行車延遲以及增加其公平性。

4. 分析原始演算法以及改良過的演算法在不同的車流密度下的運作情形，也比較在車流資訊蒐集不足的環境下，兩種演算法的運作效能。

5. 提出了演算法公平性的指標來衡量一個交通號誌控制演算法的優劣，並且以此來評估本論文中所比較的演算法。

二、相關研究

2.1 智慧型交通號誌控制

隨著人口增長、車子數量增加，城市的街口日漸壅塞，智慧型交通運輸越來越受重視，而其中交通號誌的管理是智慧型交通運輸中重要的一環，也是各個國家城市的重要議題，其主要目標是，改善路口的行車安全、最大限度的提升交通流量與減少交通延遲。經過設計的交通號誌管理，可以提高交通網路的效能，增加經濟和環境效益。而智慧型交通號誌的研究一般可區分為固定時間與動態時間規劃。

2.1.1 固定時間規劃

請大家試想一下自己的駕駛經驗，當自己從一個十字路口開車到下一個十字路口時，交通號誌燈剛好轉換為綠燈，這是不是很美好的感覺，而相反的，如果一路上一直遇到紅燈，這感覺是不是相當的糟糕呢？是的，在固定時間的交通號誌研究上，我們會希望每個交通號誌燈遵循某個特定的周期，可以讓汽車在道路上行駛時能夠遇到一路的綠燈，藉此縮短行車的延遲時間。

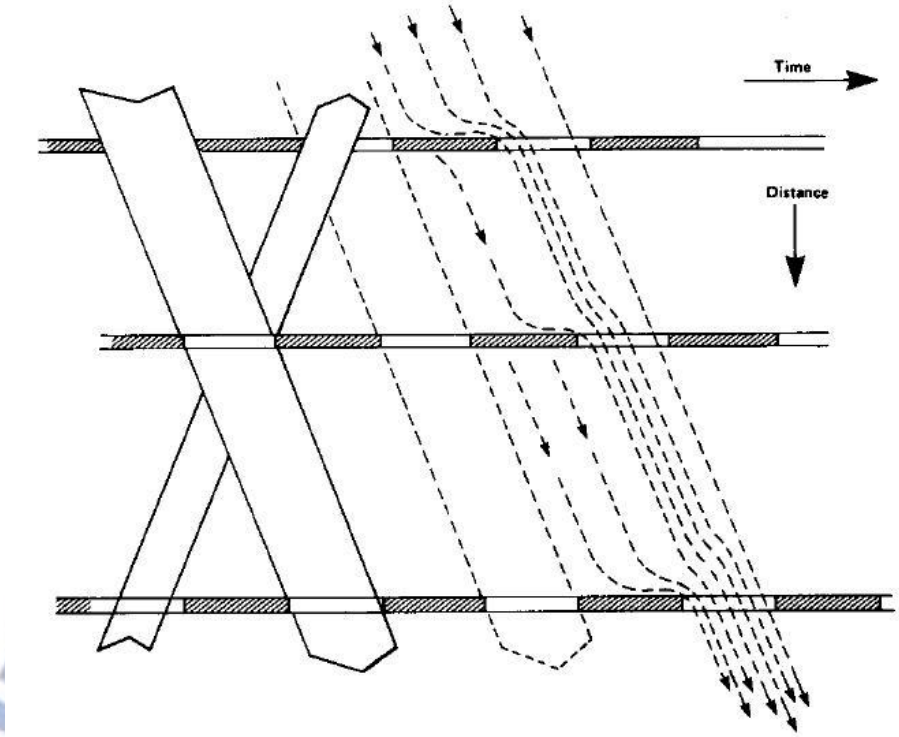


圖 1、Time－Distance 圖

資料來源：[2]

為了達到這個目標，我們通常使用 T-D 圖來做規劃，如圖 1，因為我們知道每個十字路口之間的距離，因此我們可以估算車子從這個十字路口到下一個十字路口所需要的時間，藉此我們就可以讓每個交通號誌燈綠燈的時間有一段時間差，就可以使得車子在離開這個十字路口到下一個十字路口時正巧碰上綠燈。

在圖 1 的右半部，我們可以看到等紅燈的車輛與後來出現的車輛有可能會匯流在一起而形成更長的佇列，過長的佇列會影響到已經排程好的交通號誌燈，因此如何去估算佇列長度以及設計符合的交通號誌燈轉換周期，是一個值得研究的問題，其中相關的研究有[2]、[3]、[4]、[5]。

2.1.2 動態時間規劃

在動態時間規劃的研究上，我們需要一個評量指標，來幫助我們評估交通號誌燈什麼時候該紅燈，而什麼時候該綠燈，在[1]中提出了一套方程式來幫助我們評估每個時間點的狀態。

在交通號誌的控制上我們很重視佇列的長度，也就是在等待紅燈的車輛數，所以在[1]中提出一個評估十字路口交通狀態的方程式。

$$\mathbf{Q_i(n+1) = Q_i(n) + q_i(n) - d_i(n)S_i(n)} \qquad \mathbf{(1)}$$

其中 i = 1,2,…,M 指的是某個十字路口上某條車流的編號；n = 0,1,…,N – 1 指的是一個時間間隔的編號；$Q_i(n)$ 指的是在第 n 個時間間隔的第 i 條車流上的車輛數；$q_i(n)$ 指的是在第 n 個時間間隔要加入到第 i 條車流上的車輛數；$d_i(n)$ 指的是在第 n 個時間間隔要離開第 i 條車流的車輛數；而 $S_i(n)$ 的值有可能是 0 (紅燈不可通行) 或 1 (綠燈可通行) 它意指第 n 個時間間隔的第 i 條車流上的交通號誌燈的狀態。

有了佇列長度，接下來要計算佇列長度對車輛等候時間的影響，在[1]中提出了另外一個方程式可以評估車輛的等候時間。

$$\mathbf{W_i(n+1) = W_i(n) + TQ_i(n) + 1/2\ Tq_i(n) – Td_i(n)S_i(n)} \qquad \mathbf{(2)}$$

其中 $W_i(n)$ 是指在第 n 個時間間隔第 i 條車流上的所有車輛的等候時間；T 是時間間隔的秒數。$W_i(n)$的值累積到最後就是所有車子的全部延遲時間，所以這個指標可以拿來衡量各種演算法的效能。

方程式 (1) 跟 (2) 是[1]提出來最主要的方程式，它可以有效的評估一個十字路口動態環境下的交通狀況。在交通號誌燈控制的研究中，延遲跟車子停下來的次數是最熱門的衡量指標，而這個方程式可以有效的評估延遲時間，在這套方程式中最主要的目標是要找出

$$\min \{ W(n) = \sum_{i=1}^{M} Wi(n) \} \quad (3)$$

由於方程式 (1) 和 (2) 中的數值都是單指每一條車流，所以各車流彼此之間是獨立的，因此要找$\sum_{i=1}^{M} Wi(n)$ 的最小值，我們可以針對每個十字路口去找區域最佳解，每個十字路口的紅綠燈狀態都會影響到四條車流，如圖 2 所示。

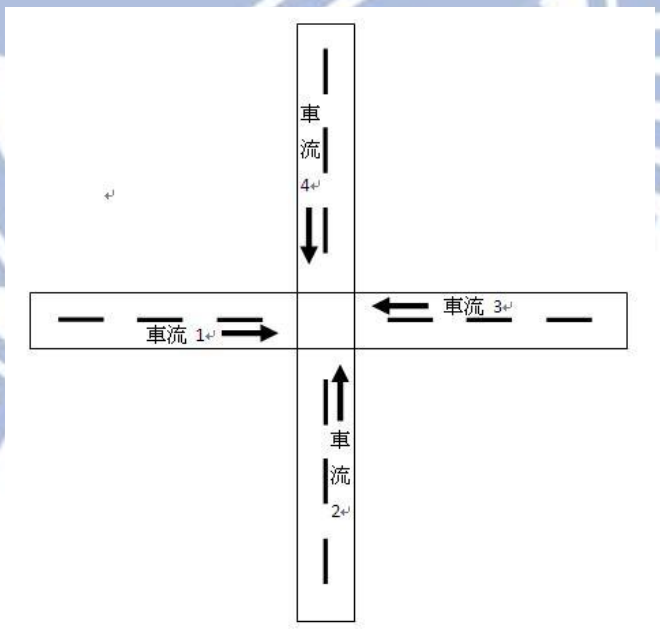


圖 2. 紅綠燈車流圖

在公式 (2) 中，我們要找一個交通號誌燈的配置可以讓這四條車流的總延遲時間最短，在這四條車流中，車流 1 與車流 3 的交通號誌燈會是相同的，而車流 2 與車流 4 的交通號誌燈會是相同的，在公式 (2)

中，$Q_i(n)$、$q_i(n)$與 T 的值不受到燈號的影響，會受到燈號不同而影響到 $W_i(n+1)$的是 $d_i(n)$的值，而 $d_i(n)$的值其實跟佇列的長短有關。

$$\mathbf{d_i(n) = \min\{ q_i(n) + Q_i(n) , d_s \}} \qquad \mathbf{(4)}$$

其中 d_s 指的是在時間間隔中，能夠通過紅綠燈車輛的最大數量，所以為了使四條車流的總延遲時間最少，要找出 $\sum_{i=1}^{4} Wi(n+1)$ 的最小值等於要找出 $\text{Max}\{ d_1(n) + d_3(n) , d_2(n) + d_4(n) \}$。

根據以上的這些資訊，演算法就能夠判斷該給予哪條道路綠燈能帶來最大效益，後來許多研究都是以這套評估車況的演算法為標準繼續延伸，例如使用模糊邏輯控制[6]、[7]，也有自己額外設計車流偵測與紅綠燈控制演算法的[8]，但是綜觀前人所做的研究，大部分研究的環境並不是使用真實世界的路網與逼真的車流模型，而且最後的成效也只以車輛的平均花費時間來做為演算法評量的標準，我們覺得在研究環境上並不足夠真實，而在我們的論文中，我們會提出演算法公平性這個另類的指標來衡量一個演算法的優劣。

2.2 NCTUns 網車路與流模擬器

在本論文中的模擬實驗是採用 NCTUns[9][10][11][12]網路與車流模擬器做為模擬平台，NCTUns 網路與車流模擬器是由國立交通大學所發展，日前 NCTUns 已經廣泛的支援了各種網路的標準與規格，例如：Ethernet、光學網路、GPRS、IEEE 802.11 a/b/e/p、WiMAX (IEEE 802.16 d/e/j)、衛星網路…等…。讓網路議題的研究者能夠使用這個平台，依照其需求自訂所需的網路模擬環境，並設計不同的模擬情境以進行模擬與研究。除了網路模擬之外，NCTUns 的車流路網系統也是其一大特色。

NCTUns 網路與車流模擬器是很優秀的模擬平台，目前廣泛的被全世界許多國家的使用者採用，做為在學術領域或商業領域進行網路或車流行為研究時的工具平台。NCTUns 具備下列的特性與優點：

1. 使用真實世界中的傳輸層與網路層的協定堆疊 (Protocol Stack)，能產生精確的模擬結果。

2. 真實世界的應用程式可以不經修改直接運行於此模擬器上。

3. 可以使用真實世界的路網環境來進行模擬，讓模擬的結果更貼近於真實世界。

4. 具有高度整合性的使用者圖形化介面 (Graphical User Interface)，提供便捷的操作方式進行模擬情境的設計、執行與模擬結果檢視。

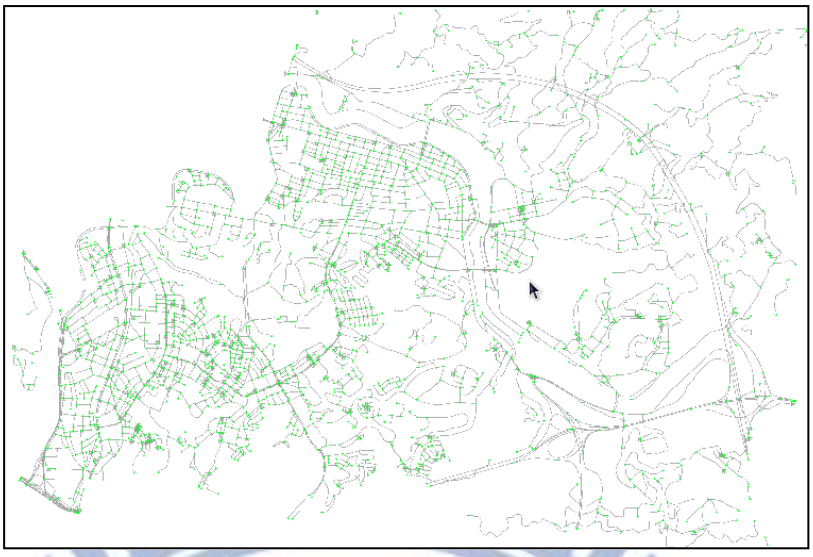


圖 3. 台北市文山區路網分布圖

NCTUns 的車流模擬系統擁有相當多的功能以及設計上的彈性，其中最具特色的是可以引進真實世界的路網，增加模擬的真實度，如圖 3 就是台北市文山區的道路分佈圖。除了引進真實世界的地圖外，在路網上他可以提供單線道與多線道的道路。交通號誌燈上，他提供兩種不同的機制讓使用者選擇，一種是中央統一控制的交通號誌燈，另一種是由 AGENT 來控制的交通號誌燈，AGENT 也是使用者可以在上面設計自己的演算法的應用程式，然後透過 NCTUns 所提供的 API 與模擬引擎溝通。在車子方面 NCTUns 讓使用者可以針對不同的車子配置不同的車子性能，例如最大加速度、最高速度、最大減速度...等，另外特別是 NCTUns 有所謂的車隊，可以把附近的車子設為同一個車隊，如此這些車子在行駛的時候就會尾隨著前方帶頭的車輛前進。

NCTUns 的 GUI 也帶給研究者非常大的方便，在模擬結束之後，NCTUns 在 Play Back 模式下可以從模擬開始到結束以動畫的方式呈現所有車子移動以及交通號誌燈的變換情形，所以可以幫助我們觀察每個時間點所有交通號誌燈的運作情形，評估是否還有可以改善的空間。

三、Landmark Based 車流模型的開發

3.1 開發動機

想要做紅綠燈的智慧型控制，必須要依據及時的路況來給予適當的紅綠燈轉換，為了使研究的環境更逼近於現實，我們必須要有更好的車流模型，而因為 NCTUns 所使用車流模型是隨機轉彎，也就是汽車到達十字路口時，會隨機選擇下一條路線前進，如此的行進方式不符合現實的狀況，所以我們必須實做出一個更符合真實的車子行進模型。

3.2 系統設計與架構說明

根據我們的觀察，車子在路上的行駛並非漫無目的，而是朝一個特定的目的地前進，而且通常是選擇一條最短路徑。因為我們決定設計一個特定標的“Landmark”。這個 Landmark 代表的意涵可能是一個熱門的景點，類似 101 大樓、博物館或者是哪一個熱門的風景區，因為我們相信車子的駕駛是為了要去這些景點而將車子開上道路。

我們實作的方法是在模擬器裡面新增加一個物件，稱為 Landmark，而為了讓車流的模型更多樣化，我們將 Landmark 設定的熱門等級，分為最高、中等以及最低，最高的等級意味著最熱門的區域，可能是西門町或哪個鬧區，而中等等級可能就是次一級的景點，例如電影院，最低等級可能只是意味著一間餐廳。而當我們把 Landmark 佈署上路網的時候，模擬器會把所有 Landmark 的位置記錄下來，接著我們可以讓車子選擇特定的 Landmark 為他行駛的目的地，如果不特定指派車子與

Landmark 的對應，則系統會自動根據 Landmark 的熱門程度有權重的指派車子與他們的目的地 Landmark。

車子的行走路線我們使用道路指引的方式，我們會算出每個十字路口到每個 Landmark 的最短路徑，因此只要汽車到了十字路口，再去察看自己所要前往的 Landmark 該往哪個方向走，一路跟隨著道路的指引，最後車子就會到達自己的目的地，而為了維持模擬的持續運行，我們在車子到達目的地後，會依據 Landmark 的熱門程度再次幫車子決定他下一個目的地，如此一來車子就可以在模擬進行中，一直朝著特定目標前進，完成我們當初開發的目的。

3.3 資料結構與實作

在開發 Landmark 之前我們必須先了解 NCTUns 裡面路網的架構，在 NCTUns 裡，一條道路是由很多區塊組成，我們把資料整理成表 1。

表 1. 路網組成元件

路網組成元件	說明
Road Block	路網組成的最小元件，由四個點座標形成的一個四方形。
Lane Block	許多的 Road Block 會形成一條 Lane，Lane 裡面會記錄它是由哪些 Road Block 所組成。
Edge Block	由許多的 Lane Block 所組成，例如：一條雙線道是由兩條不同方向的 Lane Block 所組成。

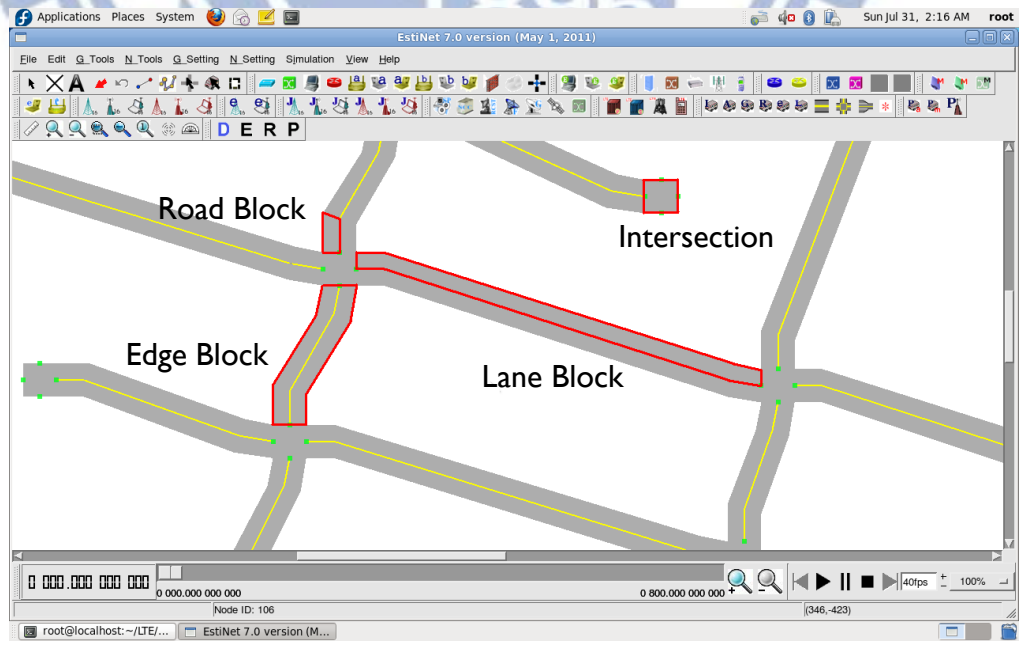


圖 4. NCTUns 道路網路

如圖 4 所示，Road Block 是路網最小塊的區塊，許多的 Road Block 會組成一條 Lane Block，而數條 Lane Block 就會組成 Edge Block。而每個十字路口我們稱為 Intersection。接下來我們著重觀察 Intersection 與 Edge Block 的資料結構。

Intersection
int Nid;
int NumofEdges;
int *EID;
double *direction;

表 2. Intersection 資料結構

表 2 是 Intersection 的資料結構，Nid 指的是這個 Intersection 的 ID，NumofEdges 指的是與這個 Intersection 相連的 Edge 數，*EID 指向一個陣列，陣列長度是 NumofEdges 的值，而陣列內容是 Edge 的 ID，*direction 跟*EID 相同，陣列內容指的是每條 Edge 的方向。

Edge Block
int Eid
int NumofLane
int NumofNode;
int *NID;
int L_Number;
int R_Number;
double length;

表 3. Edge Block 資料結構

表 3 是 Edge Block 的資料結構，Eid 是這條 Edge 的 ID，NumofLane 是這條 Edge 的 Lane 數，NumofNode 是指這條道路與幾個十字路口交接，NID 指向一個陣列，大小是 NumofNode 的值，L_Number 與 R_Number 是指左右線道的個數，length 是指這條 Edge 的長度。

由 Intersection 與 Edge 這兩個資料結構中，Intersection 知道有哪些 Edge 與它相連，而 Edge 也知道與它相連的 Intersection，而且我們也知道 Edge 的長度，我們可以將這些資訊轉成圖論，然後使用 Dijktra 來計算每個 Intersection 到 Landmark 的最短距離。由於 Dijktra 是計算點到點的最短距離，但 Landmark 是擺在 Edge 上，所以我們會在每個 Landmark 上紀錄離他最近的 Intersection 的 ID，然後我們以到達這個 Intersection 當作到達 Landmark。

3.4 開發成果

在開發成果這個章節我們會比較有擺放Landmark與沒有擺放Landmark的情況。首先我們讀取台北市文山區的一小段路網，並且在上面隨機佈置100台車輛如

圖5，但此時我們還沒有將Landmark擺放上去，我們進行600秒的模擬，觀察車子的散布情形，之後再與我們有擺設Landmark的兩個案例做比較。

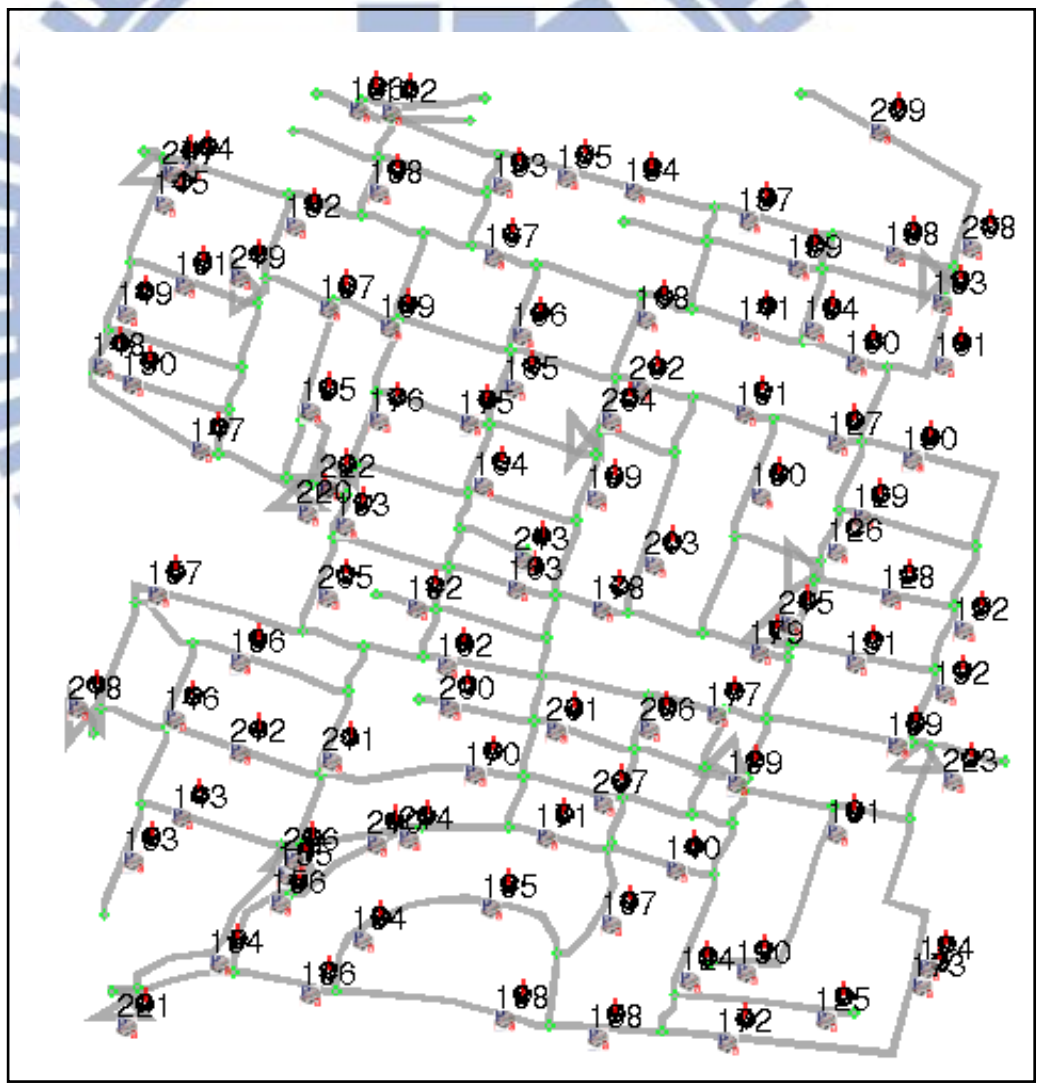


圖5.100台車子初始佈署狀態

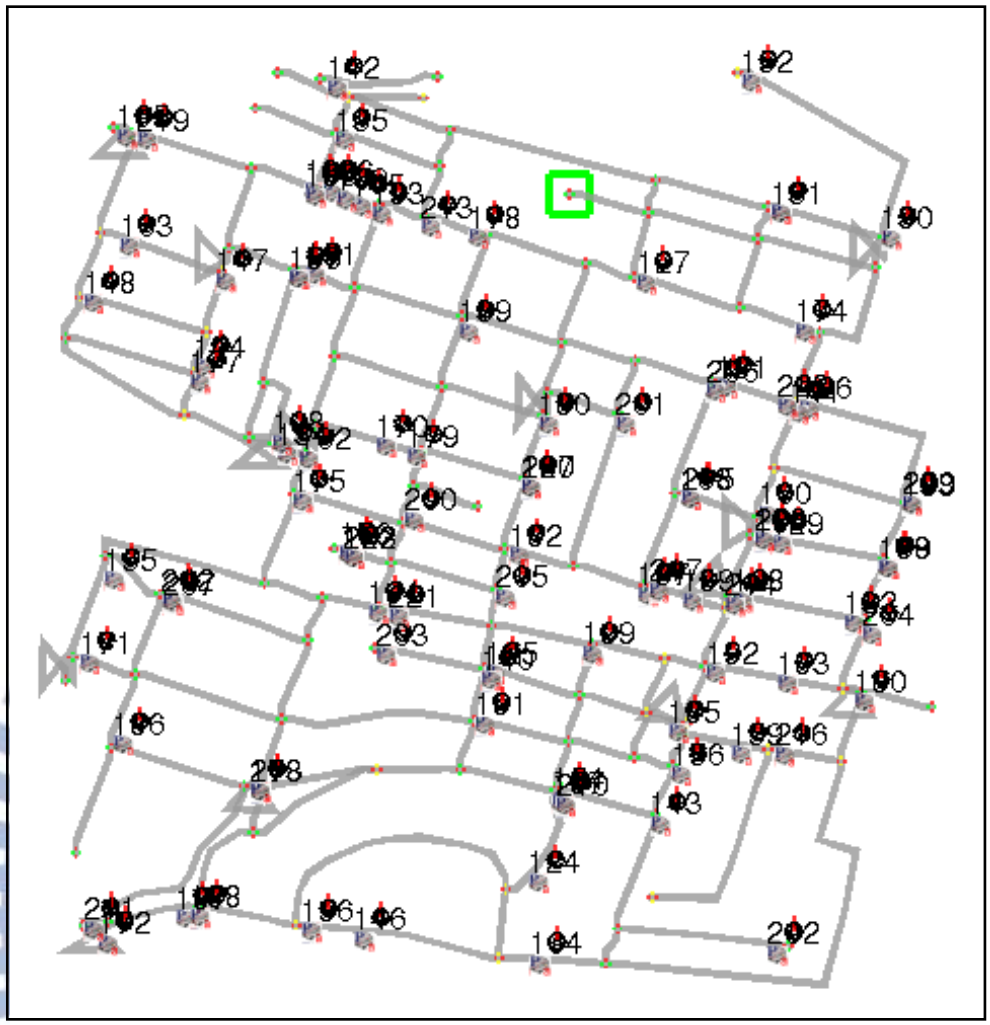


圖 6.100 台車子模擬結果

當沒有佈署 Landmark 時，車子會隨機轉彎，而圖 5 是模擬前 100 台車輛的佈署狀況，而圖 6 是模擬了 600 秒後，車輛的所在位置，因為車子是隨機轉彎的，所以車子在 600 秒模擬後依然散佈在地圖上，並沒有特別聚攏的情形，在這種狀況下我們很難模擬塞車的情形以及城市熱門區域的交通環境。

接下來的兩個案例是我們把 Landmark 都佈置到地圖上，模擬開始後車子就會依據一定的比例朝各自的 Landmark 前進。

在案例一中，我們佈署 4 個 Landmark 到地圖上如圖 7 所示。

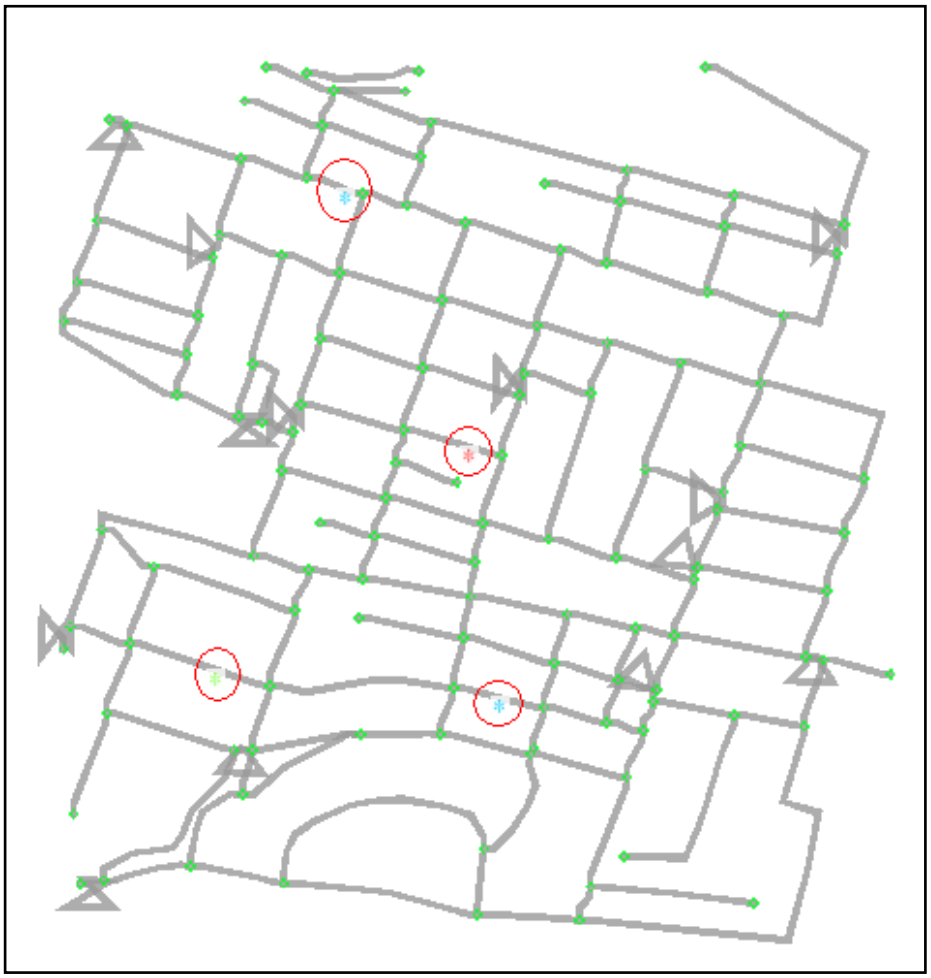


圖 7. 案例一的 Landmark 配置狀況

圖 8. 案例一的模擬結果

圖 8 是案例一模擬 600 秒後的結果，我們可以清楚的看見所有的車輛都往有配置 Landmark 的位置聚攏，因為只有布置四個集中的 Landmark，所以其他地方會沒有車輛行進，這個案例是為了證實所有的車輛都會朝著 Landmark 前進，在這情況下，我們可以營造出都市中的熱門景點，而熱門景點的附近會伴隨的塞車情形也會真實的呈現出來。

在案例二中，我們嘗試佈署更多的 Landmark，如圖 9 就是佈置了 10 個 Landmark 的情形。

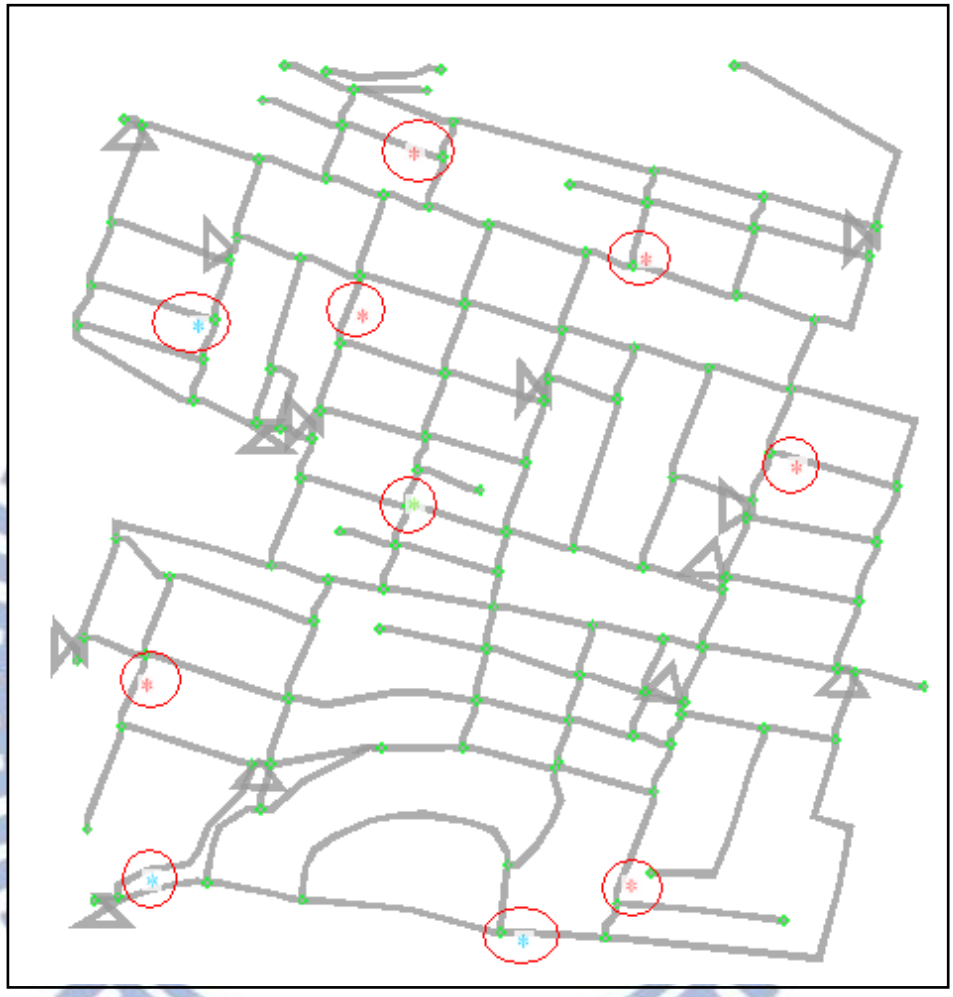


圖 9. 案例二的 Landmark 配置狀況

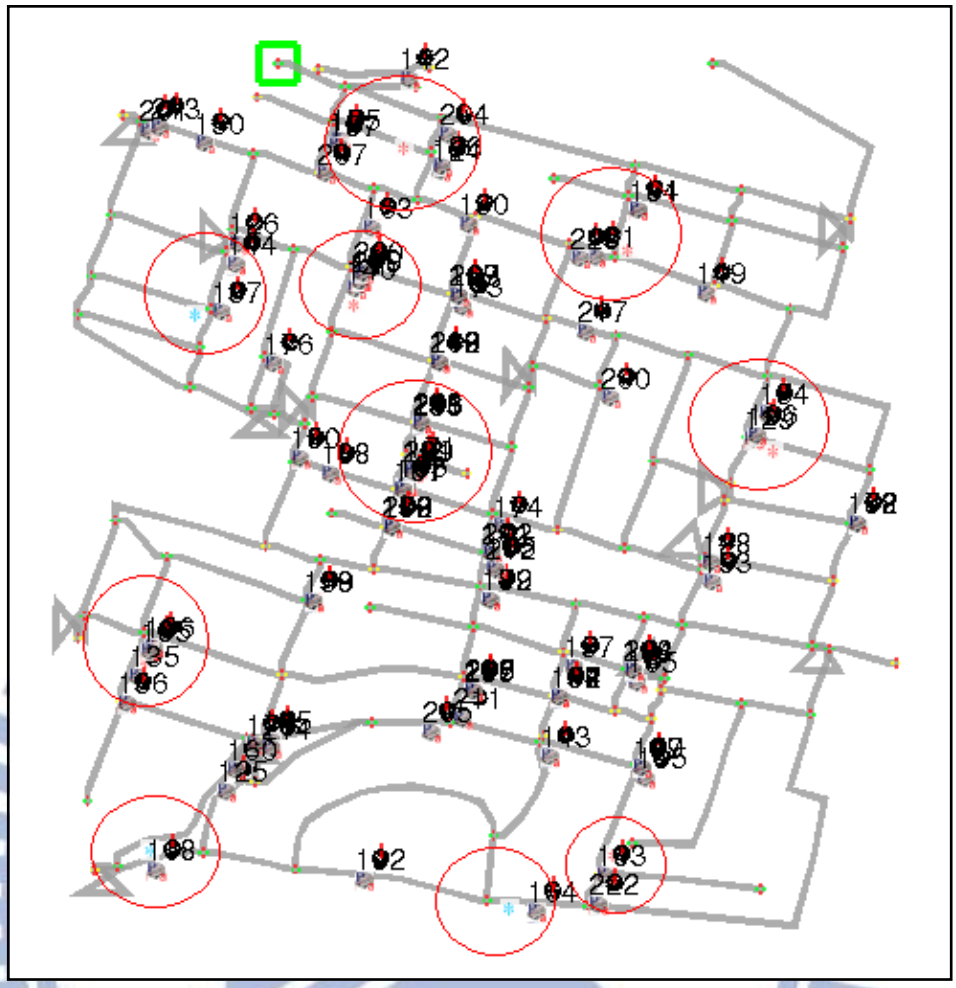
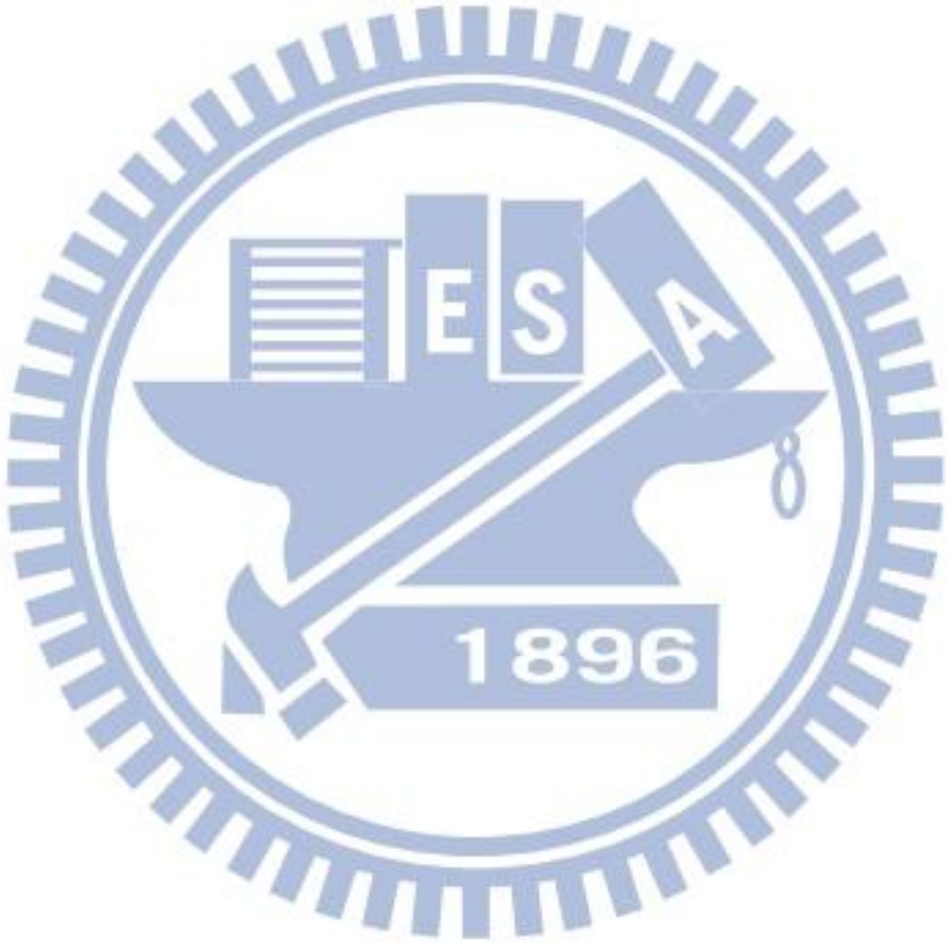


圖 10. 案例二的模擬結果

圖 10 是案例二模擬 600 秒後的結果，我們可以看到，每個 Landmark 附近都有車流匯聚，而中間會有些道路是 Landmark 間來往必經之路，這些路段會形成所謂的主要幹道，容易匯集車流形成塞車。在案例二中，由於我們將 Landmark 佈置得更多更分散，所以可以看到車流會分散在地圖上，但並不是均勻的分散，而是在各地有 Landmark 的附近形成車流，如此的情況比起原本隨機轉彎的車流模型更貼近現實。

以上兩個不同案例的 Landmark 佈置會導致兩種不同型態的車流網路，善用不同熱門程度的 Landmark 以及將這些 Landmark 佈署在不同的地方就可以形成各式各樣的車流網路系統，這樣的系統對於要做車流模擬的研究者來說，是非常方便的功能。同時這樣的系統對於我們做智慧型交通號誌控制的研究也有著至關重要的影響。



四、智慧型交通號誌的改良

我們將智慧型交通號誌燈的演算法實做到 NCTUns 模擬器，並且研究各種拓樸的模擬數據後，我們發現原有的智慧型交通號誌控制演算法有幾個部分還可以繼續改良。分別說明如下：

4.1 固定時間的探查車況改成動態時間

在原本的演算法中會每隔一個固定的時間間隔去探查現在道路的車況如何，然後視車況配置紅燈與綠燈，因為是固定的時間間隔，所以每次配置完紅綠燈後，必須等待下一次的車況探查才有可能改變紅綠燈，但車流的狀況並不對等，並且差異很大，如果某條路上只有少數一兩台車要通過，卻配置給他 20~30 秒的綠燈時間，此時如果另外一條道路上有車輛到來，會增加額外的等候時間。

所以在我們改良的演算法中，我們並不把探查車況的時間固定住，我們會先評估道路上是否有衝突的車流在競爭紅綠燈，如果只有單向的車流，那我們可以給予一段比較長的綠燈時間，然後等候下一次探查車況的時間，如果有不同方向的車流要競爭紅綠燈，那我們會觀察車流的數量是否差異很大，如果一邊只有少數一兩台車，那我們會給予他短時間的綠燈，然後馬上把綠燈還給較多車流的道路。

固定時間探查車況還有另外一個缺點，探查時間如果過長，會導致上述我們說的問題，配置大量時間給少數車輛，造成其他路的車輛不必要的等候時間。但如果探查時間過短，會有另外一個問題是，當兩條道

路的車流差不多壅塞，頻繁的探查車況有可能會導致頻繁的紅綠燈變換，然而每次紅綠燈的轉換，都有會有 5 秒的黃燈時間，這個時間雙向的車都無法通行，所以頻繁的變換紅綠燈，對雙向的車輛而言都是額外的負擔。

因此在我們改良的演算法中，遇到雙向車流都壅擠的時候，我們會配置一個較長的綠燈時間，之後再換另一條道路有較長的綠燈時間，這樣可以避免因為頻繁變換紅綠燈所帶來的負面影響。

4.2 使用老化技術取代最長紅燈時間

在未改良的智慧型交通號誌控制演算法中，我們會評估車流狀況，給予較多車流的道路綠燈以舒緩交通，但如果兩條車流的流量差異過大，例如主要幹道與鄉間小路的交接，有可能造成演算法每次都給予主要幹道綠燈，而導致鄉間小路上的車流無限期的等待，因此在舊有的演算法中會使用最長的紅燈時間，以避免少數車輛被無限期的紅燈困住，而我們實作的演算法中將最長紅燈時間制訂為 90 秒。

而在我們改良的演算法中，我們要引進老化技術來取代最長紅燈時間，這邊可以分兩個部分來說，第一是確保車子不會被無限期的紅燈困住，第二是盡快恢復主要幹道的通暢。

為了達到第一點的目標，確保車子不會被無限期的紅燈困住，我們將使用老化技術，也就是說，當演算法第一次將綠燈給予主要幹道的車流時，遇到紅燈的車流會增加自身的權重，以便在下一次演算法的判斷中，比較容易搶得綠燈，但如果還是被判定為紅燈，則再次加重權重，

最多連拿兩次紅燈，第三次就一定會搶到綠燈，在這個機制下，車子不會被無限期的紅燈困住，也不用等到 90 秒這個最長的紅燈時間，而且主要幹道的車在沒有車流競爭的道路上，也不用因為到了 90 秒最長的紅燈時間就被迫將綠燈換給沒有車流的道路，因而增加車子的延遲。

而為了達到第二點的目標，當我們因為老化技術而要把綠燈讓給車流量較小的道路時，我們會觀察此時道路上的車輛數，然後針對車輛數給予相對應的綠燈時間，讓這一批車通過十字路口後，馬上把綠燈還給主要幹道的大批車流，如此一來可以在顧及公平性的同時又不至於拖慢多數車輛的行車時間。

五、模擬環境與模擬結果

5.1 研究與模擬方式

在本論文的研究中，我們想知道智慧型交通號誌燈的演算法在各個不同參數下，對於降低行車延遲有多大的效益，並比較原本的演算法與我們改良過的演算法對於降低行車延遲上有多大的改良效果。

我們首先對於不同的模擬情境參數進行分類如表 4. 模擬情境分類表，將各種不同的拓樸環境與車流密度搭配我們想要研究的模擬設計，定義出各種模擬案例，之後針對各個案例的模擬結果進行整理、分析與歸納。

表 4. 模擬情境分類表

情境參數	說明
車流拓樸	我們會使用不同的 Landmark 佈置方式，搭配不同的車流密度，結此組合出不同車流拓樸。
演算法時間間隔	我們想觀察不同的時間間隔對原本演算法效能的影響，然後與我們改良過的演算法做比較。
車流資訊蒐集	我們想知道在車流資訊蒐集齊全的程度對於演算法會有什麼樣的影響。

5.2 模擬情境參數

在定義本論文所要探討的模擬案例前，我們先將各種不同的模擬情境參數進行分類，在設定模擬案例時，將依所要探討之各個議題的模擬設計搭配各種模擬情境參數，定義出不同的模擬案例。在本節中，我們將介紹我們使用的車流拓樸型式，然後對車流密度、演算法時間間隔、車流資訊蒐集完整程度進行說明。

5.2.1 拓樸型式

在本論文中，我們預計採用三種不同的 Landmark 數量與擺設方式，藉以呈現三種不同的車流拓樸型式，地圖是選用台北市文山區中的一部分，大小是 700 公尺乘 700 公尺。以下將分別說明我們在模擬過程中所使用的拓樸：

1. 拓樸 A

拓樸 A 的型式如圖 11，是由 5 個 Landmark 所以組成的車流拓樸，在此拓樸中，由於拓樸都佈置於路網中間區域，所以車流會集中於地圖中央的位置造成塞車。

圖 11. 拓樸 A 的 Landmark 佈署情形

2. 拓樸 B

拓樸 B 的型式如圖 12，是由 10 個 Landmark 所組成的車流拓樸，在此拓樸中，因為 Landmark1 佈置的比拓樸 A 更多且更分散，所以車流的走向會比拓樸 A 更多樣化，同樣的車流也會分散到地圖的各處。

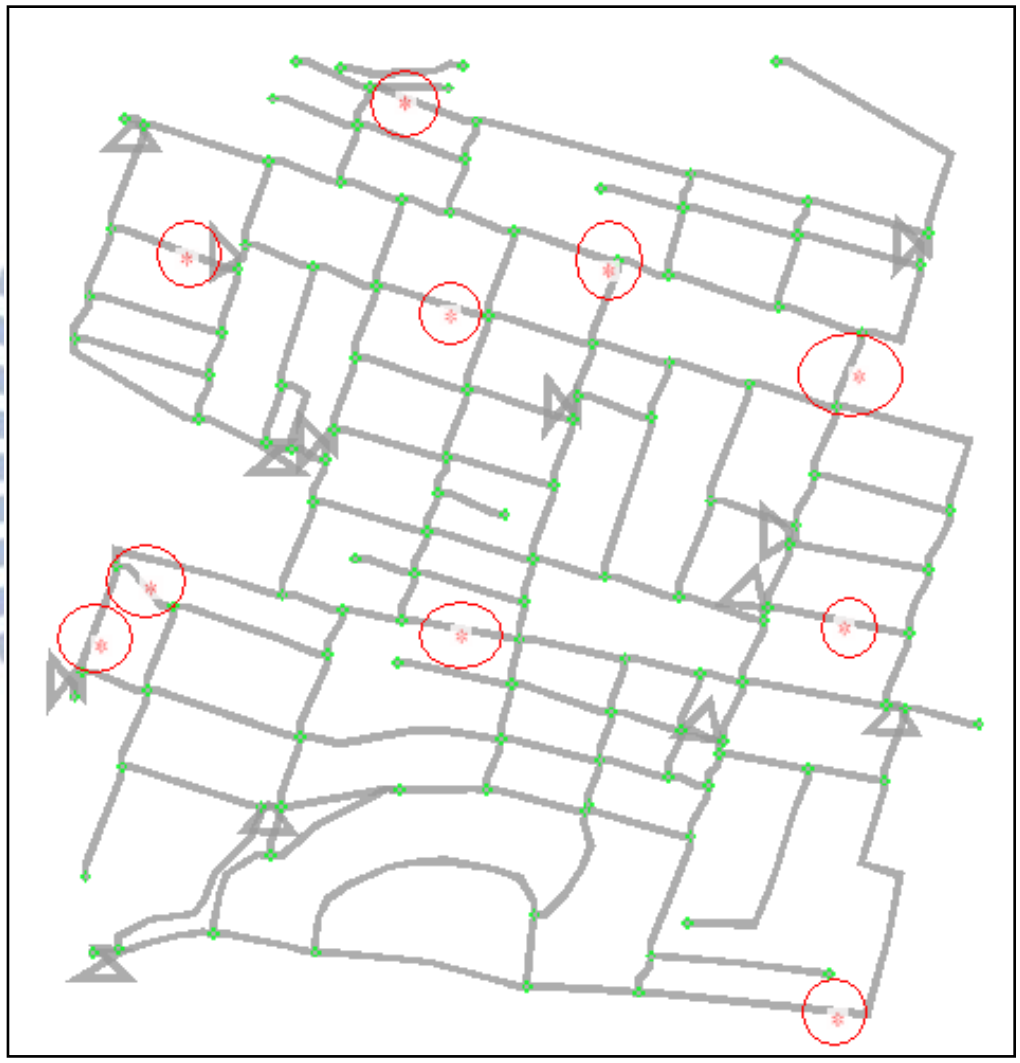


圖 12. 拓樸 B 的 Landmark 佈署情形

3. 拓樸 C

拓樸 C 的型式如圖 13，是由 20 個 Landmark 所組成的車流拓樸，在此拓樸中，Landmark 佈置得比拓樸 A 與拓樸 B 更多且更分散，所以車流走向會比拓樸 A 與拓樸 B 更複雜，同時也更均勻散佈在地圖上。

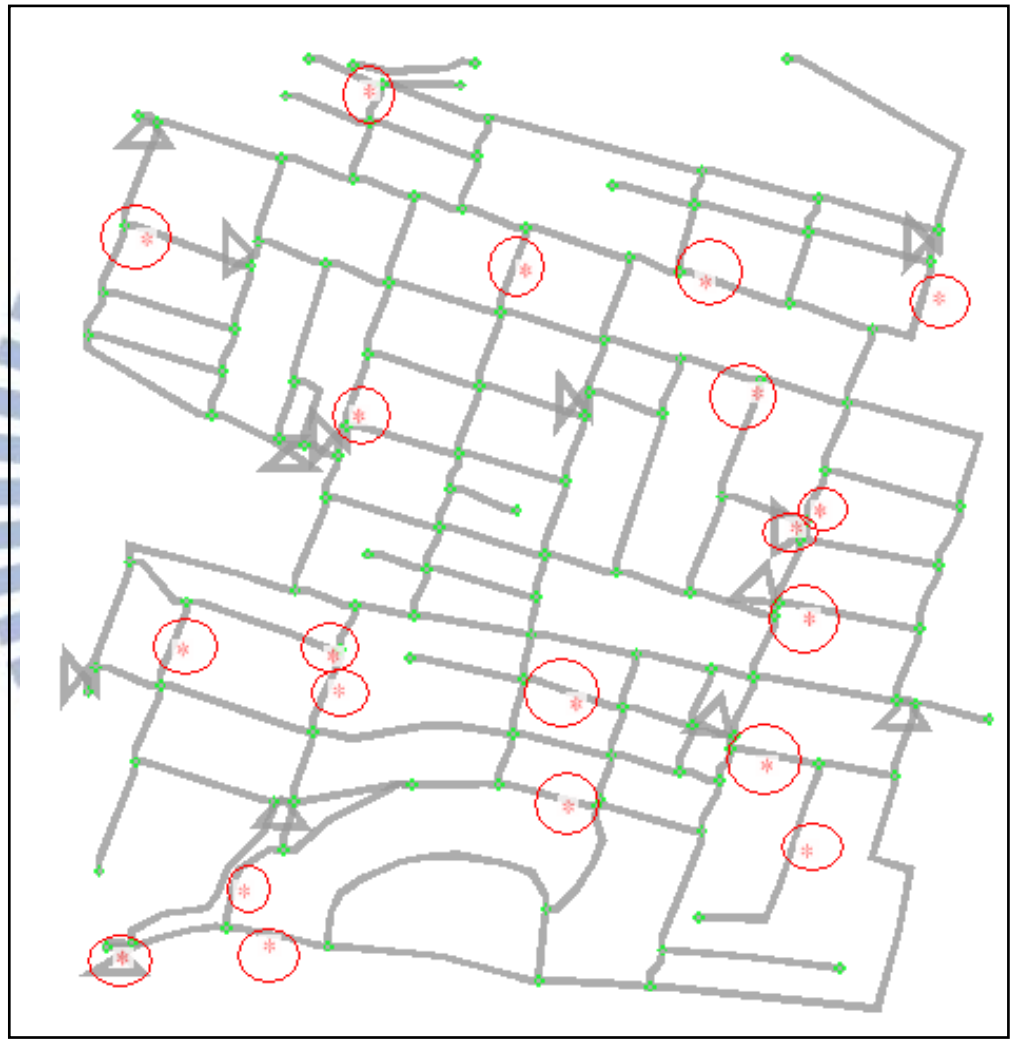


圖 13. 拓樸 C 的 Landmark 佈署情形

5.2.2 車流密度

在本論文研究的車流拓樸中，除了 Landmark 的個數與佈置位置有差異外，我們也會探討不同的車流密度對演算法效能的影響，我們前面三種拓樸 A, B, C 均會與三種車流密度做搭配，三種車流密度分別是 70 台車、140 台車與 210 台車，如圖 14、圖 15、圖 16 所示，這些車輛會均勻的散佈在地圖上，然後搭配三種拓樸形成九種不同的車流模型。

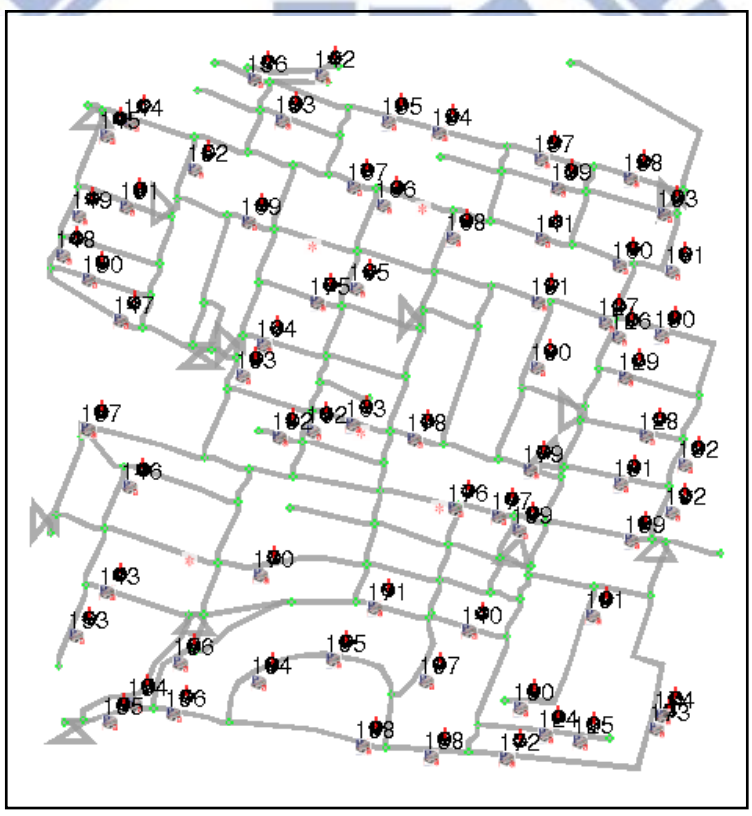


圖 14. 70 台車散佈在地圖上

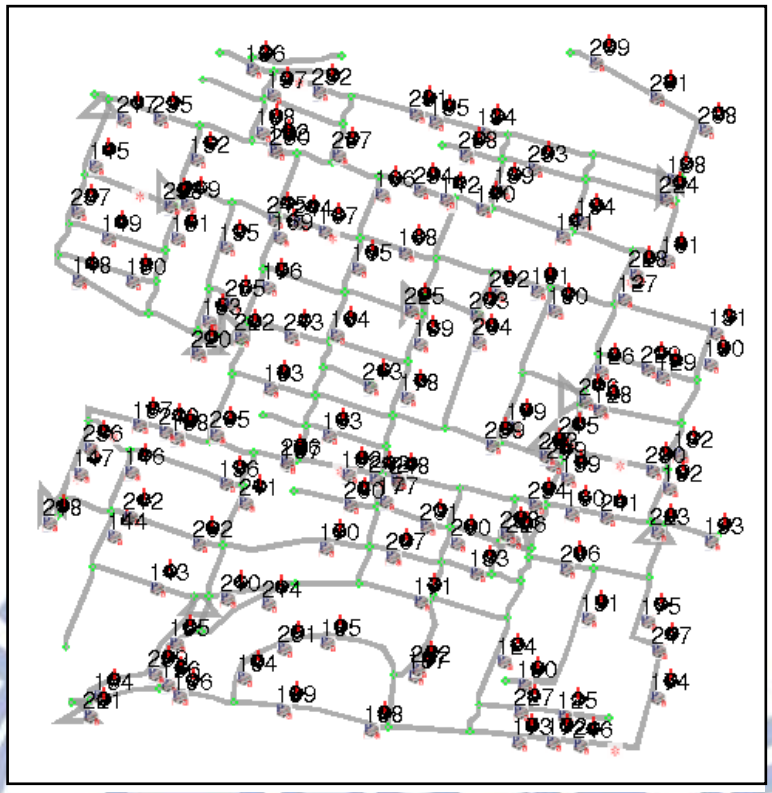


圖 15.140 台車散佈在地圖上

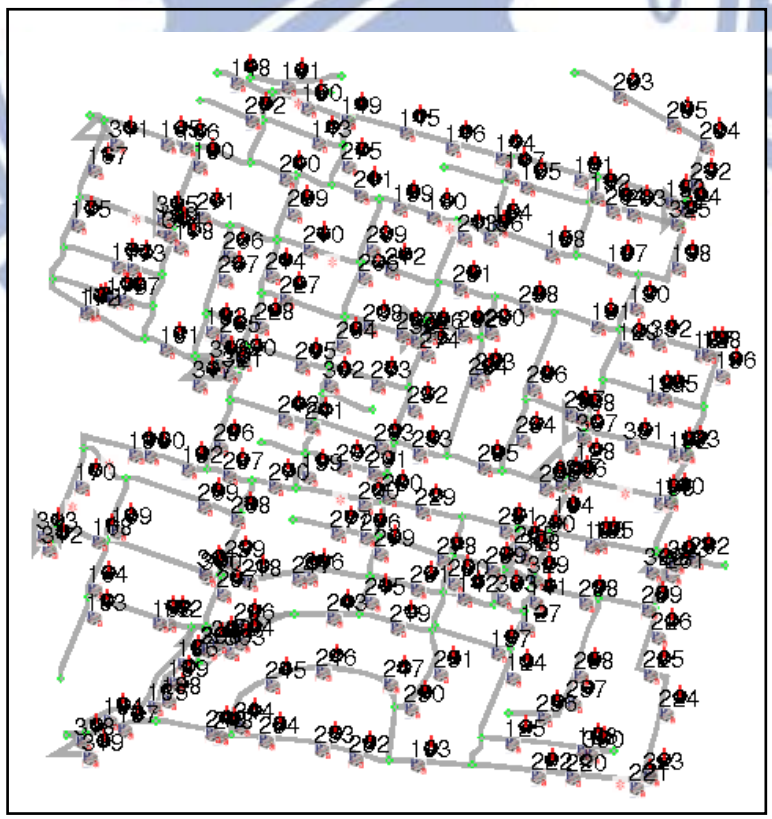


圖 16.210 台車散佈在地圖上

5.2.3 演算法時間間隔

在本論文中提到的智慧型交通號誌控制演算法，是動態去觀察每個時間間隔內紅綠燈的變化對行車延遲的影響，所以我們也把時間間隔的長短當成是一個研究的指標，由於演算法會隔一段固定時間去評估一次車況並決定現在哪個方向的交通號誌燈該變成綠燈，而要改變燈號最快也要等到下一次評估車況的時候，所以這個時間間隔同時也代表著最短的綠燈秒數，我們會把演算法的時間間隔區分為以下數種，如表 5 所示。

表 5. 演算法時間間隔秒數

演算法去評估車況的時間間隔秒數	說明
5 秒	最短綠燈秒數：5 秒　黃燈秒數：5 秒
10 秒	最短綠燈秒數：10 秒　黃燈秒數：5 秒
15 秒	最短綠燈秒數：15 秒　黃燈秒數：5 秒
20 秒	最短綠燈秒數：20 秒　黃燈秒數：5 秒
25 秒	最短綠燈秒數：25 秒　黃燈秒數：5 秒
30 秒	最短綠燈秒數：30 秒　黃燈秒數：5 秒
35 秒	最短綠燈秒數：35 秒　黃燈秒數：5 秒
40 秒	最短綠燈秒數：40 秒　黃燈秒數：5 秒

5.2.4 車流資訊蒐集

在本論文中，智慧型交通號誌控制演算法中，評估車況是假設我們擁有所有車流資訊的前提下做計算，我們假設在未來的智慧型交通運輸上，會有一套完整的車流資訊蒐集的系統，知道所有汽車的位置與行車速度，藉此來評估車況以調變交通號誌燈達到減少延遲的效果。

但是在本論文中我們也希望研究智慧型交通號誌控制的演算法，如果在車流資訊蒐集不齊全的情況下，運作的效能如何，我們模擬的情境如表 6 所示。

表 6. 車流資訊蒐集程度

車流資訊蒐集程度	說明
100 %	系統能掌握所有的車輛位置與行車速度
80 %	系統能掌握 80 %的車輛位置與行車速度
60 %	系統能掌握 60 %的車輛位置與行車速度
40 %	系統能掌握 40 %的車輛位置與行車速度
20 %	系統能掌握 20 %的車輛位置與行車速度
0 %	系統能掌握 0 %的車輛位置與行車速度

5.3 模擬設計

在本論文的規劃中，我們將幾個想要探討的主題分別設計了不同的模擬案例。各模擬案例的設計目的、說明、以及在該設計中使用的演算法種類分別說明如下：

1. 模擬設計 A

模擬設計 A 是為了比較原設計的智慧型交通號誌控制演算法與沒有使用任何智慧型交通控制的交通號誌變換，能減少多少的行車延遲，關於模擬設計 A 的設計目的與說明請見表 7 與表 8：

表 7. 模擬設計 A 之設計目的與說明

模擬設計編號	A
設計目的	比較使用了原設計的智慧型交通號誌控制與沒有使用智慧型交通號誌控制的交通號誌變換，能減少多少的行車延遲。
模擬設計說明	為了知道智慧型交通號誌控制是否真的能確實減少行車延遲，以及這樣的智慧型控制是否增加了部分車輛的行車時間以換取減少其他車輛的行車時間這種不公平的效益。

表 8. 模擬設計 A 使用的模擬情境參數

組別	**演算法**	**時間間隔 (最短綠燈時間)**	**車流資訊蒐集**	**車流拓樸**
0	不使用智慧型演算法控制	每 40 秒轉換一次燈號	100 %	拓樸 A,B,C
1	原智慧型控制演算法	每 15 秒評估一次車況	100 %	拓樸 A,B,C

2. 模擬設計 B

模擬設計 B 是為了分析原設計的演算法在不同的時間間隔下的效能，時間間隔越短，代表著智慧型交通號誌燈的靈敏度越高，關於模擬設計 B 的設計目的與說明請見表 9 與表 10：

表 9. 模擬設計 B 之設計目的與說明

模擬設計編號	B
設計目的	為了分析原設計的演算法在不同的時間間隔下的效能。
模擬設計說明	越短的時間間隔，代表著越高的靈敏度，但是過高的靈敏度可能會導致交通號誌燈的頻繁變換，造成負面的效果，這個模擬設計是希望找到一個最合適的演算法時間間隔。

表 10. 模擬設計 B 使用的模擬情境參數

組別	**演算法**	**時間間隔 (最短綠燈時間)**	**車流資訊蒐集**	**車流拓樸**
0	原演算法	每 5 秒評估一次車況	100 %	拓樸 B
1	原演算法	每 10 秒評估一次車況	100 %	拓樸 B
2	原演算法	每 15 秒評估一次車況	100 %	拓樸 B
3	原演算法	每 20 秒評估一次車況	100 %	拓樸 B
4	原演算法	每 25 秒評估一次車況	100 %	拓樸 B
5	原演算法	每 30 秒評估一次車況	100 %	拓樸 B
6	原演算法	每 35 秒評估一次車況	100 %	拓樸 B
7	原演算法	每 40 秒評估一次車況	100 %	拓樸 B

3. 模擬設計 C

模擬設計 C 是為了比較我們改良過後的演算法與原設計演算法在效能上的差異，關於模擬設計 C 的設計目的與說明請見表 11 與表 12：

表 11. 模擬設計 C 之設計目的與說明

模擬設計編號	C
設計目的	為了比較我們改良過後的演算法與原設計演算法在效能上的差異。
模擬設計說明	我們改良的演算法是使用動態時間間隔，與原本設計的固定時間間隔的演算法有設計上的差異，我們想知道這兩種演算法對於行車延遲的改善有多少的效果。

表 12. 模擬設計 C 使用的模擬情境參數

組別	**演算法**	**時間間隔 (最短綠燈時間)**	**車流資訊蒐集**	**車流拓樸**
0	不使用智慧型控制演算法	每 40 秒變換一次燈號	100 %	拓樸 A,B,C
1	原智慧型控制演算法	每 15 秒評估一次車況	100 %	拓樸 A,B,C
2	改良後智慧型控制演算法	動態時間評估車況	100 %	拓樸 A,B,C

4. 模擬設計 D

模擬設計 D 是想研究，在車流資訊蒐集不完整的情況下，依照車況來變動交通號誌燈的演算法是否能正常運作，以及運作的效能如何，關於模擬設計 D 的設計說明與目的請見表 13 與表 14：

表 13. 模擬設計 D 的設計目的與說明

模擬設計編號	D
設計目的	研究在車流資訊蒐集不完整的情況下，依照車況來變動交通號誌燈的演算法是否能正常運作，以及運作的效能如何。
模擬設計說明	假設我們擁有所有車輛的位置與行車速度，我們可以很適當的配置交通號誌燈的燈號，但現實狀況不見得所有的車流資訊我們都能夠掌握，所以我們想知道在車流資訊掌握不充足的情況下，演算法的運作是否正常，我們也想知道當我們掌握車流資訊到哪個程度就能夠準確的使用智慧型交通號誌。

表 14. 模擬設計 D 使用的模擬情境參數

組別	**演算法**	**時間間隔 (最短綠燈時間)**	**車流資訊蒐集**	**車流拓樸**
0	不使用智慧型控制演算法	每 40 秒變換一次燈號	0%~100%	拓樸 A,B,C
1	原智慧型控制演算法	每 15 秒評估一次車況	0%~100%	拓樸 A,B,C
2	改良後智慧型控制演算法	動態時間評估車況	0%~100%	拓樸 A,B,C

5.4 模擬結果與分析

在本節中，我們將呈現之前介紹的模擬設計的模擬結果，並且將分析模擬結果來了解智慧型交通號誌控制的演算法在各個層面所帶來的成效。

5.4.1 模擬設計 A 之結果

模擬設計 A 的設計目的是比較沒有使用智慧型號誌控制演算法與原設計演算法的效能差異，我們使用了拓樸 A、B、C 並搭配三種車流密度，這九種車流拓樸中，都顯示使用了智慧型號誌燈控制演算法比沒有使用智慧型交通號誌控制演算法能減少 20% ~ 40%不等的行車延遲時間。

請看圖 18、圖 20、圖 22，圖中的橫軸表示車輛使用智慧型控制演算法與沒有使用智慧型控制演算法，到達目的地的時間差，如果是正值，代表使用智慧型控制演算法後所減少的行車延遲，而負值就是增加的行車延遲。圖中我們能發現，雖然有很小一部分的車子行車時間有增加，但絕大部分的車子都大大的縮減了行車時間，這是因為智慧型交通號誌燈是使用了顧全大局的演算法，所以為了整體行車時間的縮短，有可能會造成少部分的車輛行車時間的增加。

值得一提的是，這種智慧型交通號誌燈控制的演算法，在車流密度不高時，只有一小部分的車子增加了一些等待時間，但卻能讓大部分的車子縮減很多行車時間，這是因為在車流未飽和的情況下，車子到十字路口時，智慧型交通號誌演算法就能給予車子綠燈通行。但是相對的，在車流密度比較高的環境下，就會有車輛被犧牲而增加較長的行車時間，所以在圖 18、圖 20 圖 22 中顯示，有車輛的行車延遲甚至增加了 300 秒之多，這是因為演算法為了顧全大局所造成的不公平現象，雖然如此，

整體的行車時間，還是有很大幅度的縮減，這是智慧型交通號誌燈控制演算法帶來的利益。

我們將模擬設計 A 的模擬結果整理成圖表，請參見圖 17 圖 22。

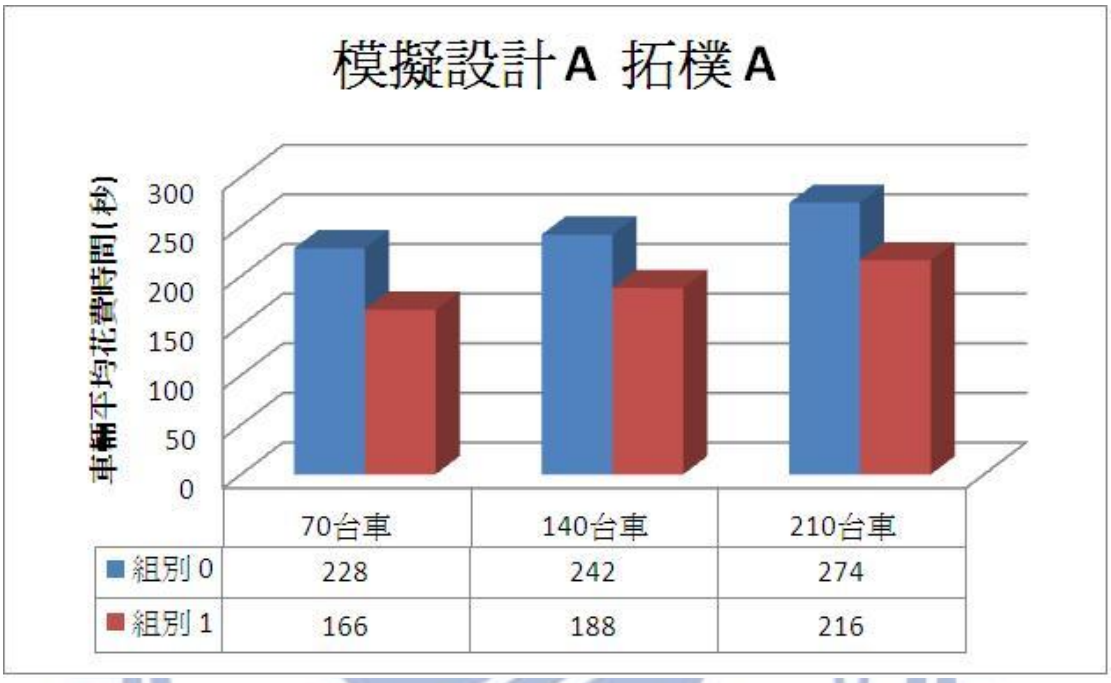


圖 17. 模擬設計 A 拓樸 A 之模擬結果圖

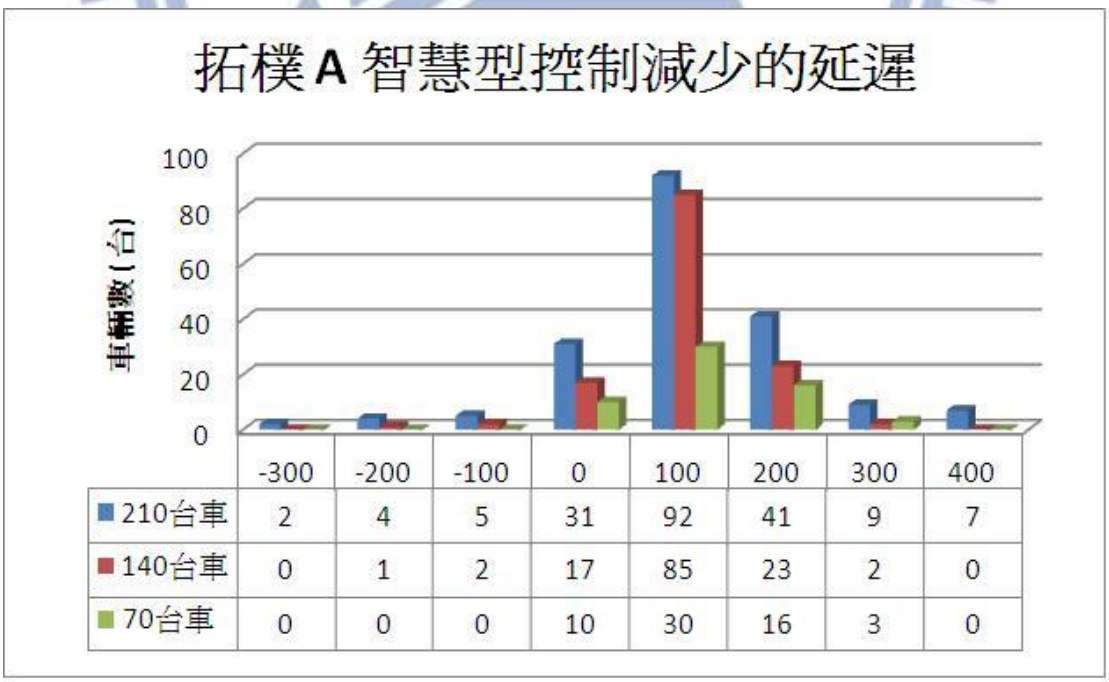


圖 18. 拓樸 A 智慧型控制減少的時間延遲

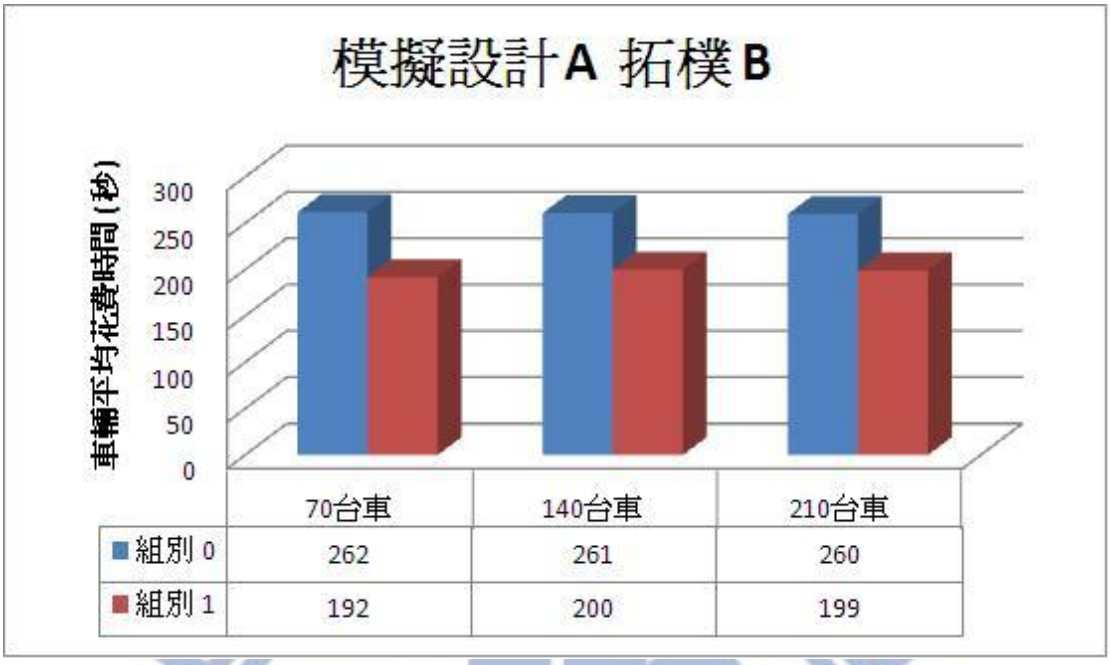


圖 19.模擬設計 A 拓樸 B 之模擬結果圖

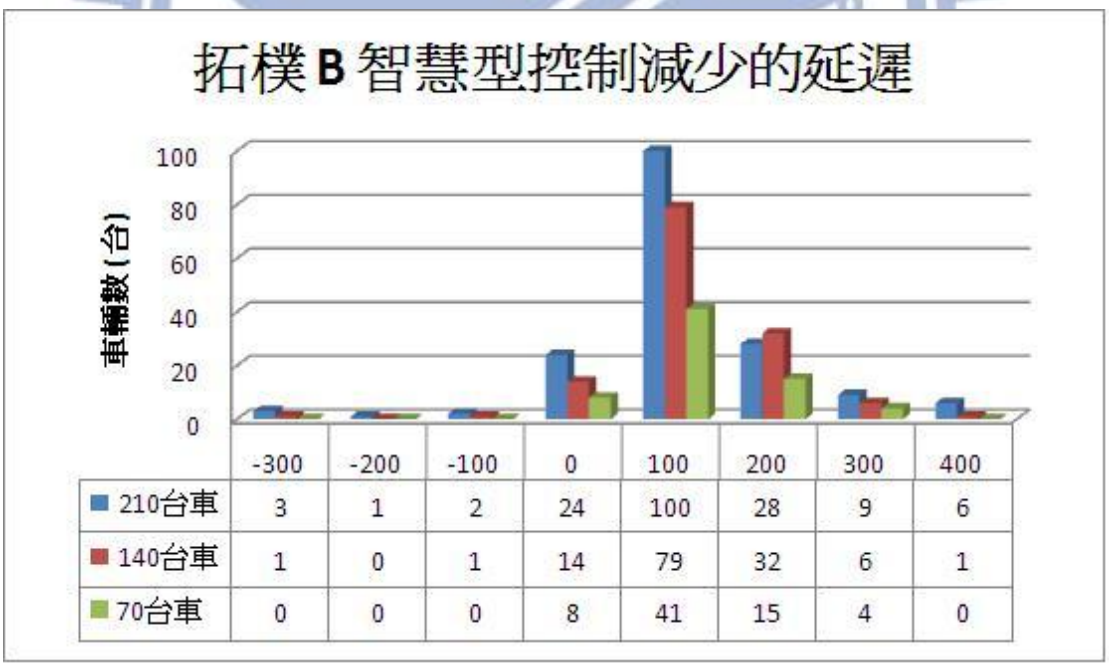


圖 20.拓樸 B 智慧型控制減少的時間延遲

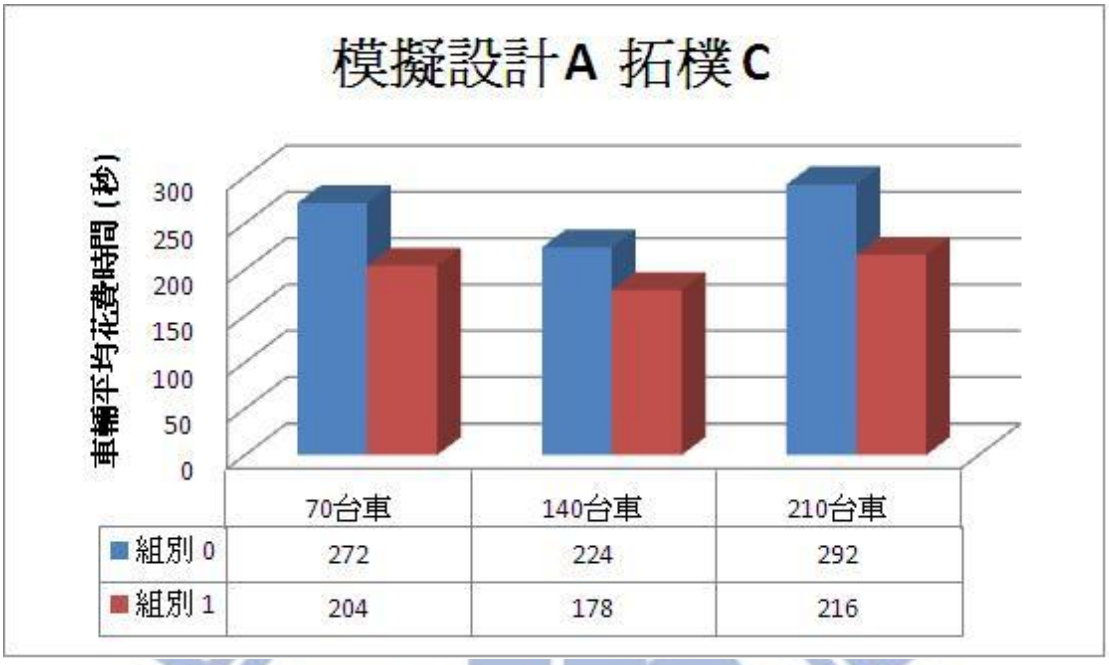


圖 21.模擬設計 A 拓樸 C 之模擬結果圖

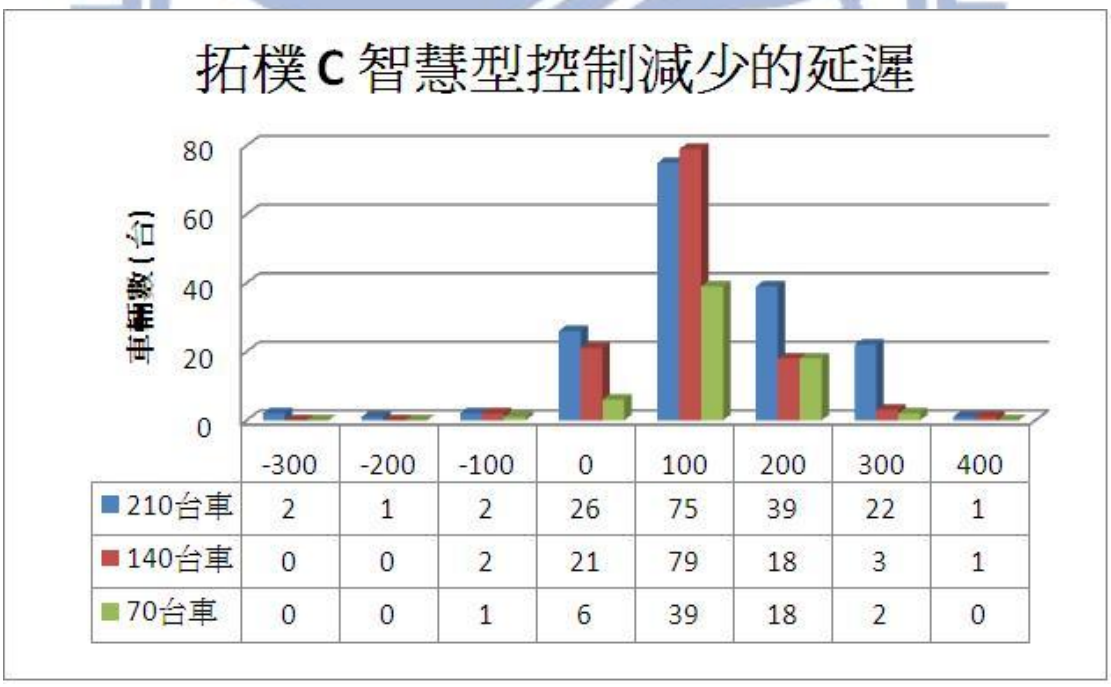


圖 22.拓樸 C 智慧型控制減少的時間延遲

5.4.2 模擬設計 B 之結果

模擬設計 B 是希望觀察原設計的演算法在不同的時間間隔下，其效能表現如何，因為智慧型交通號誌燈控制演算法是每隔一段固定時間就使用演算法去評估車況，並對當下的車況選擇一個最佳解來分配紅綠燈，所以這個評估車況的時間間隔如果越短，代表智慧型交通號誌燈評估車況的頻率越頻繁，對車況改變的靈敏度較高，也代表著最短的綠燈時間也越短暫，而評估車況的時間間隔如果越長，代表智慧型交通號誌燈評估車況的頻率越不頻繁，對車況改變的靈敏度越低，也代表著最短的綠燈時間越冗長。

我們模擬的結果發現，在一般的情況下，越頻繁的評估車況然後調整紅綠燈，會有比較好的效果，如圖 23 圖 24 所示，我們可以看到隨著評估車況的時間間隔越長，車輛平均的花費時間也隨著提高，這是因為在主要幹道上，有很大量的車流，而智慧型交通號誌控制的演算法會讓主要幹道的車流一直拿到綠燈，但是在智慧型交通號誌控制的演算法中，有最長的紅燈時間的限制，所以在 90 秒之後，交通號誌燈會被強制轉換，而轉換後必須等到下一次評估車況時，才能將綠燈換給主要幹道，因此評估車況的演算法時間間隔越短暫，就能越早把交通號誌燈轉換回來，讓主要幹道的車流繼續通行。這也是為什麼在圖中會隨著時間間隔越長，車輛平均花費的時間越多的原因。

一般來說，頻繁的評估車況，並就當時的最佳解去配置交通號誌燈，會有比較好的效能，但是有一個狀況是大家不能忽略的，就是在紅綠燈的轉換期間，會有一個 5 秒的黃燈時間，在我們的程式中，遇到黃燈時車輛也會停下來，所以這 5 秒的黃燈時間是雙向都不能通行的，所以我

們可以預見，如果太過頻繁的變換燈號，就會有更多的黃燈時間，導致整體行車的時間被拉長，因此我們看圖 25 圖 26 這兩個高密度的拓樸，在比較壅塞的交通環境，太過頻繁的評估車況，有可能會對兩條同樣壅塞的車流造成乒乓效應，導致交通號誌燈頻繁變換，而導致行車時間的增加，所以會造成如圖 25 圖 26 中，以十秒為評估車況的時間間隔效能會比以五秒為評估車況的時間間隔的效能要好。

我們將模擬設計 B 的模擬結果整理成圖表，請參見圖 23 至圖 26。

圖 23.模擬設計 B 拓樸 B 70 台車的模擬結果

圖 24.模擬設計 B 拓樸 B 140 台車的模擬結果

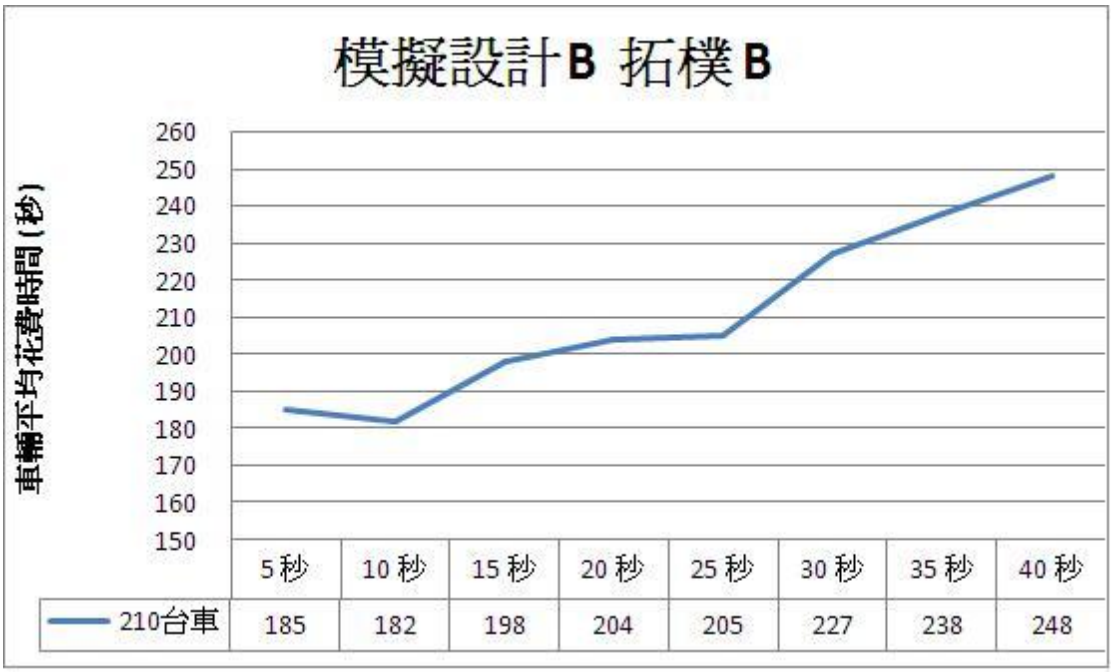


圖 25.模擬設計 B 拓樸 B 210 台車的模擬結果

圖 26.模擬設計 B 拓樸 C 210 台車的模擬結果

5.4.3 模擬設計 C 之結果

在模擬設計 C 中，我們比較改良後的演算法與原設計的演算法在效能上的差異，在效能的評估上，我們針對兩個方面去探討，第一點是車輛到達目的地的平均花費時間，如果花費的時間越少那代表這演算法對降低整體行車延遲有比較好的效能。第二點是針對公平性來看，因為在交通號誌的控制上，有一邊是綠燈，另一邊就會是紅燈，所以減少了整體的行車延遲時間，有可能是犧牲少數車輛增益大多數車輛換來的結果，因此演算法的公平性也是我們希望列入考量的。

首先我們先看在車流密度比較小的拓樸中，原本演算法與我們改良過後的演算法的效能比較。在圖 27 中我們可以看到組別 2 也就是我們改良過的演算法，所花費的時間，比組別 1 原始演算法花費的時間少，而且在圖 29 圖 30 圖 31 圖 32 中可以看到，改良後的演算法能減少犧牲車輛的車輛數，也減少犧牲車輛延遲時間的總和，所以在車輛密度不高的情況下，我們改良過後的演算法在行車延遲以及公平性的效能上都比原本的演算法更好。

接下來我們看車流密度高的情況，在圖 28 中我們可以看到，改良後的演算法跟原始的演算法對於減少行車延遲的效能表現差不多，這是因為在車流密度高的環境下，兩邊的道路都飽和，此時不管給予哪條道路綠燈，總會有另外一群車輛在等候，所以對減少整體行車延遲，已經很難再有所提升，但是對於演算法的公平性而言，還是有提升的空間，請看圖 33 圖 34 圖 35 圖 36，我們改良過的演算法對於被犧牲車輛的車輛數以及整體犧牲的延遲時間，都能有所改善，由模擬結果可知，在車

輛密度高的情況下，我們無法改善行車延遲，但卻能改善演算法的公平性。

我們將模擬設計 C 的模擬結果整理成圖表，請參見圖 27 至圖 36。

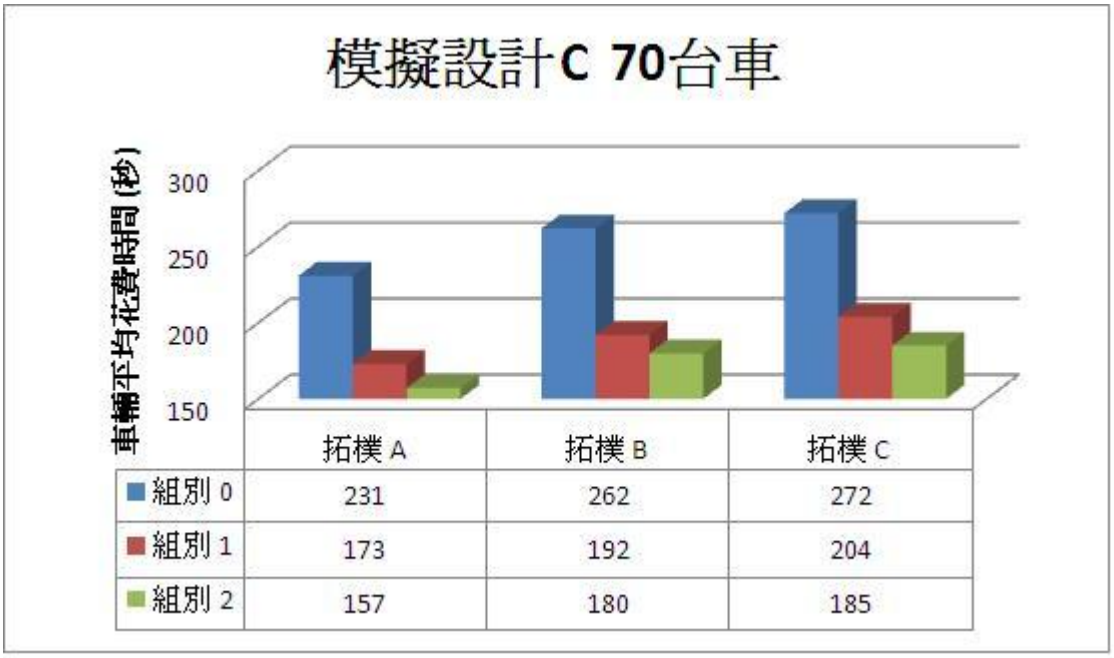


圖 27. 模擬設計 C 70 台車模擬結果

圖 28. 模擬設計 C 210 台車模擬結果

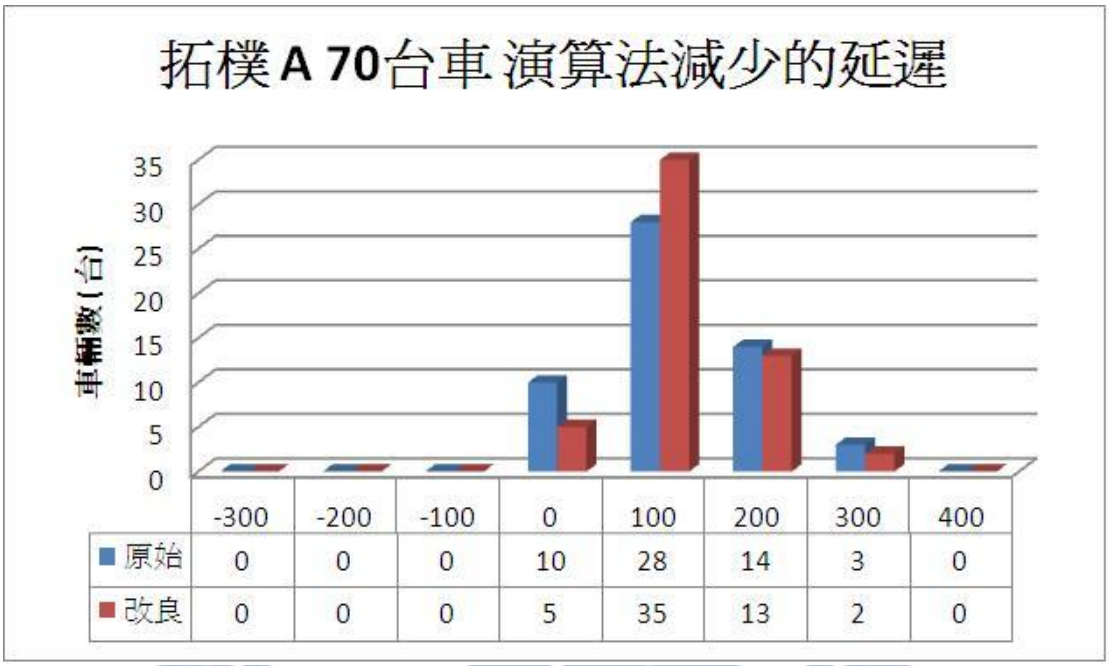


圖 29.拓樸 A 70 台車 演算法減少的延遲

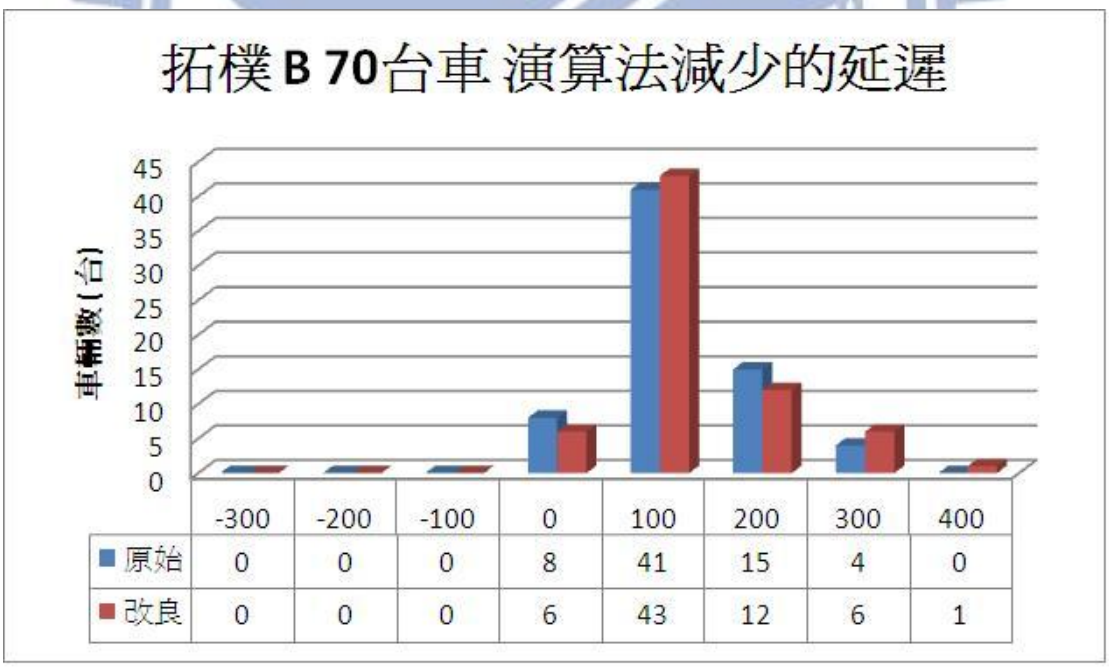


圖 30.拓樸 B 70 台車 演算法減少的延遲

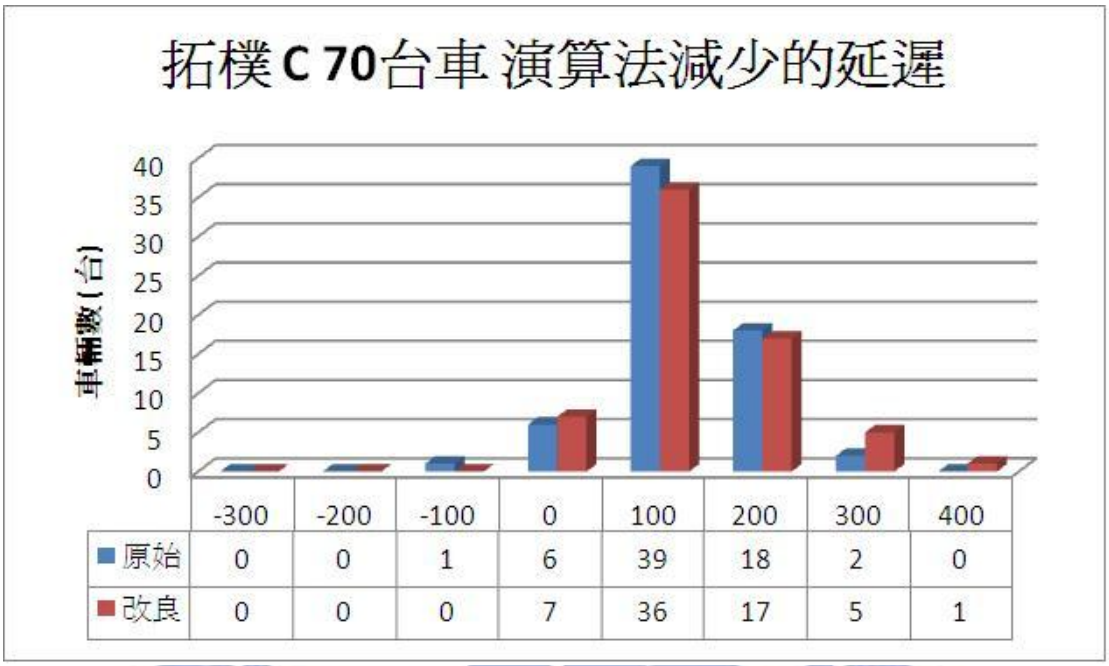


圖 31. 拓樸 C 70 台車 演算法減少的延遲

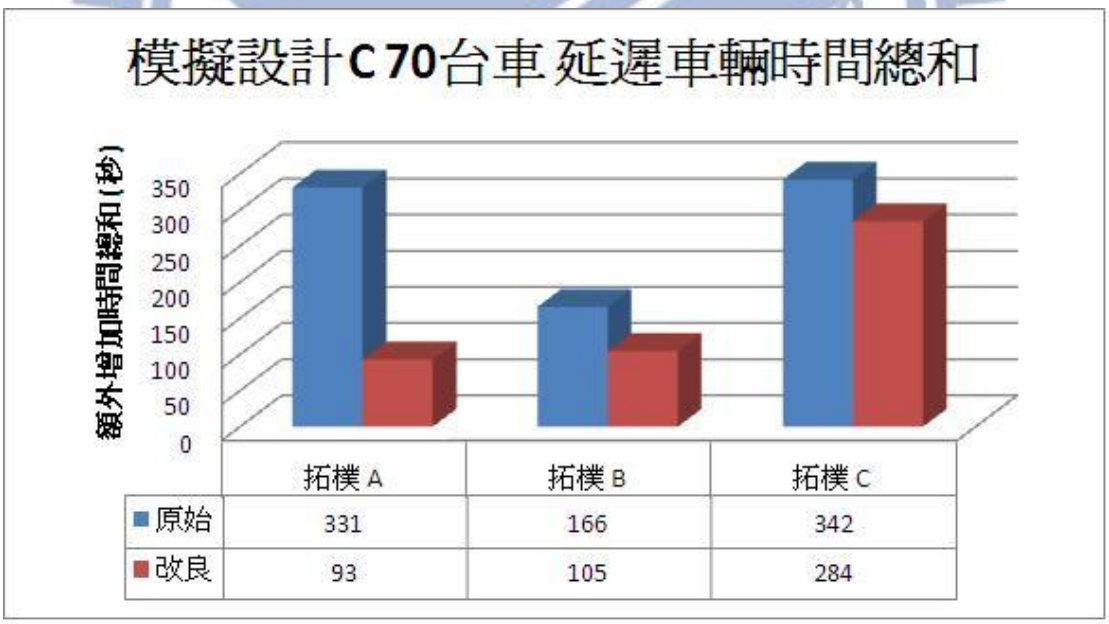


圖 32. 模擬設計 C 70 台車延遲車輛時間總和

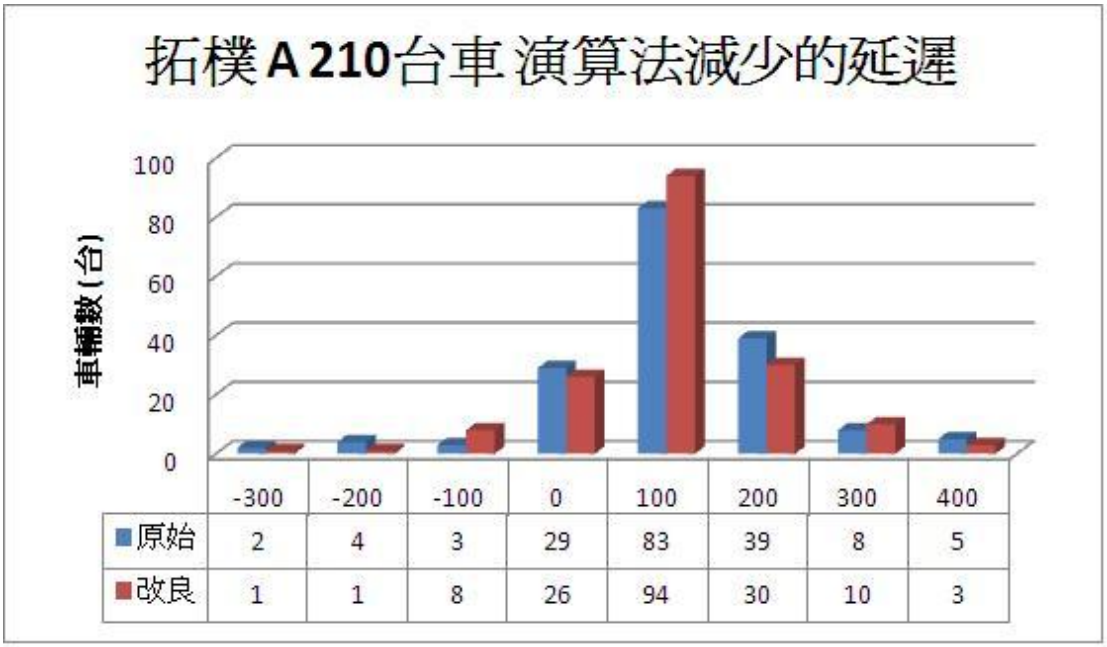


圖 33.拓樸 A 210 台車 演算法減少的延遲

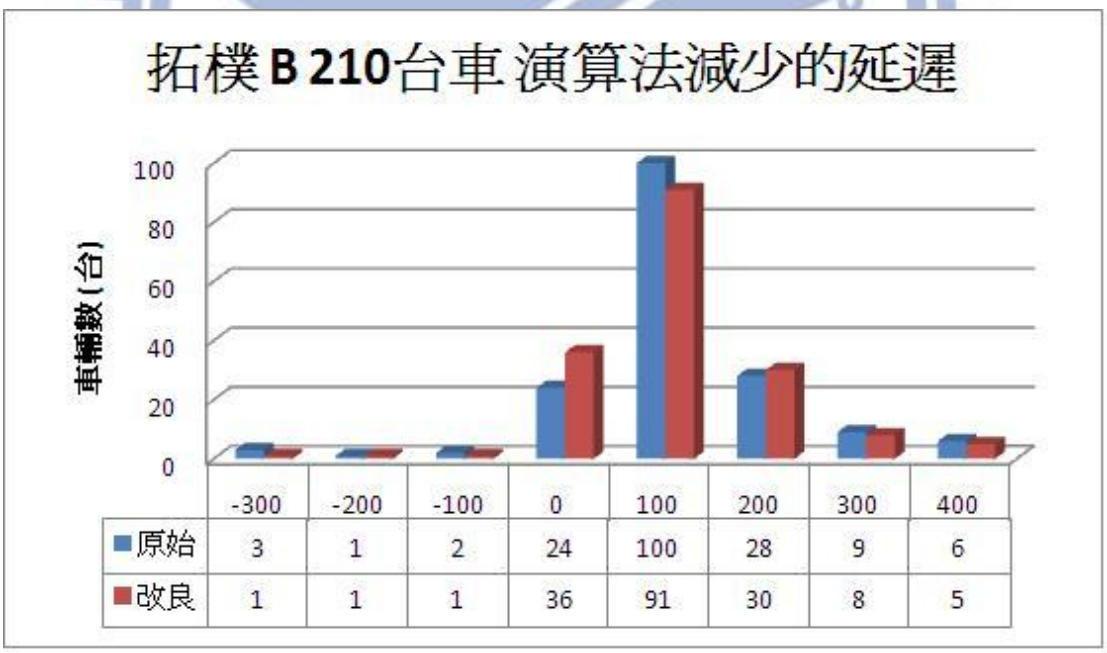


圖 34.拓樸 B 210 台車 演算法減少的延遲

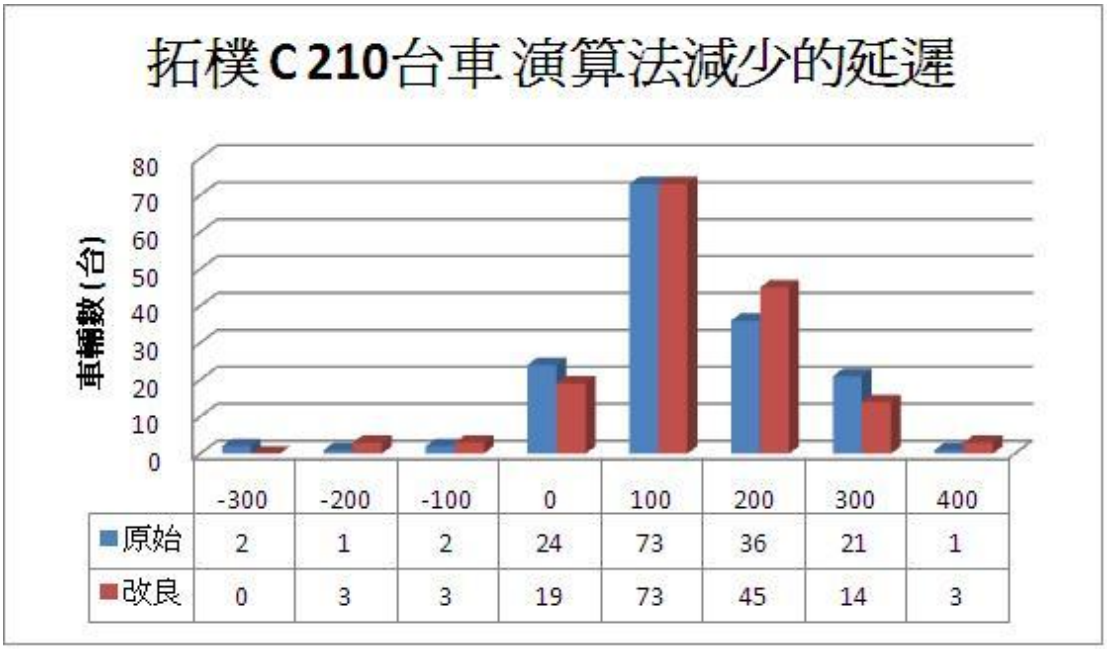


圖 35.拓樸 C 210 台車 演算法減少的延遲

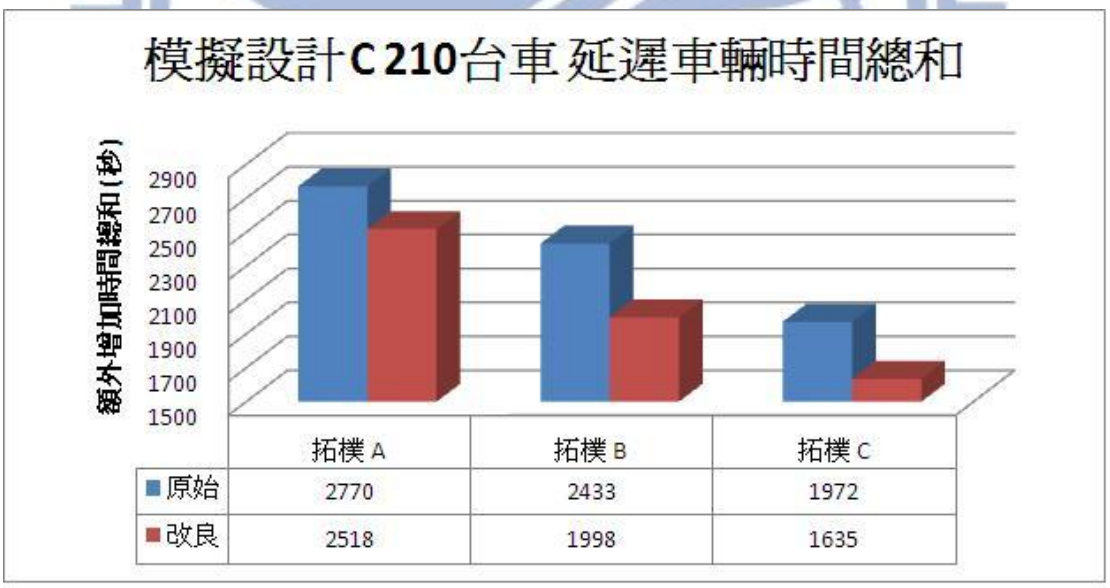


圖 36.模擬設計 C 210 台車 延遲車輛時間總和

5.4.4 模擬設計 D 之結果

在模擬設計 D 中，我們想要探討在車流資訊蒐集不夠完整的情況下，原始演算法與我們改良過後的演算法運作的效能如何。

首先我們先分析原始演算法的效能，在圖 37 與圖 38 中我們可以發現，原始演算法在車流資訊蒐集越完整的情況下，能有越好的效能表現，而且在資訊蒐集程度有達到 40%以上的話，演算法都還能維持不錯的效能，但隨著資訊蒐集程度減少到了 20%或以下，演算法的效能可能就會低於不使用演算法的狀況，這是因為，資訊蒐集程度太差的話會造成演算法嚴重的誤判，給予車流少的道路優先通行，或著有車流到了路口，交通號誌燈卻渾然不知的情況。

在圖 37 與圖 38 中我們也能發現，演算法在車流密度較高的環境中，車流資訊蒐集不完整的影響會比在車流密度低的環境中還小，這是因為在車流密度低的環境下，每次到達路口的車輛都很少，此時如果只有 20%的機會蒐集到車輛資訊，那這些車輛被忽略的機會很高，會導致交通號誌燈沒有發現這些少數車輛，以至於沒有把紅綠燈變換過去讓車輛通行，車子就會一直被延遲到第二次或第三次的車況評估才有可能拿到綠燈通行。但是在高車流密度的環境下，路口的車輛很多，雖然只有 20%的車輛被搜集到，但車流較多的道路還是會有比較多的車輛被搜集到，因此大車流的道路還是容易取得綠燈通行，因此在資訊蒐集不完整的情況下，效能降低的幅度才不至於這麼大。

接下來我們要分析改良過後的演算法，由於我們改良過的演算法是根據道路的車輛數去配置綠燈秒數，所以沒有最大的紅燈時間，因此 0%的車流資訊會導致紅綠燈永遠不會變換，因此我們使用 5%的車流資訊來代替，在圖 39 與圖 40 中我們可以發現改良後的演算法在車流資訊蒐集不足的環境下比較不受到影響，這是因為我們改良後的演算法是根據車輛數配給綠燈秒數，如果只蒐集到 40%的資訊的話，那我們配給的秒數會比較少，然後很快的進入下一次的車況判定，因此對改良過的演算法來說，會更頻繁的去評估車況造成演算法的靈敏度增加，所以演算法如果誤判之後很快就能把錯誤修正回來，所以在車流資訊蒐集比較不足的環境下，改良過的演算法還是能維持不錯的效能。

在圖 41 與圖 42 的綜合比較中，我們可以發現改良過的演算法在車輛少的時候，運作效能都比原始演算法好，在車輛密度高的時候車流資

訊蒐集不足的情況下，改良過的演算法還是能提供更好的效能。

我們將模擬設計 C 的模擬結果整理成圖表，請參見圖 37 至圖 42。

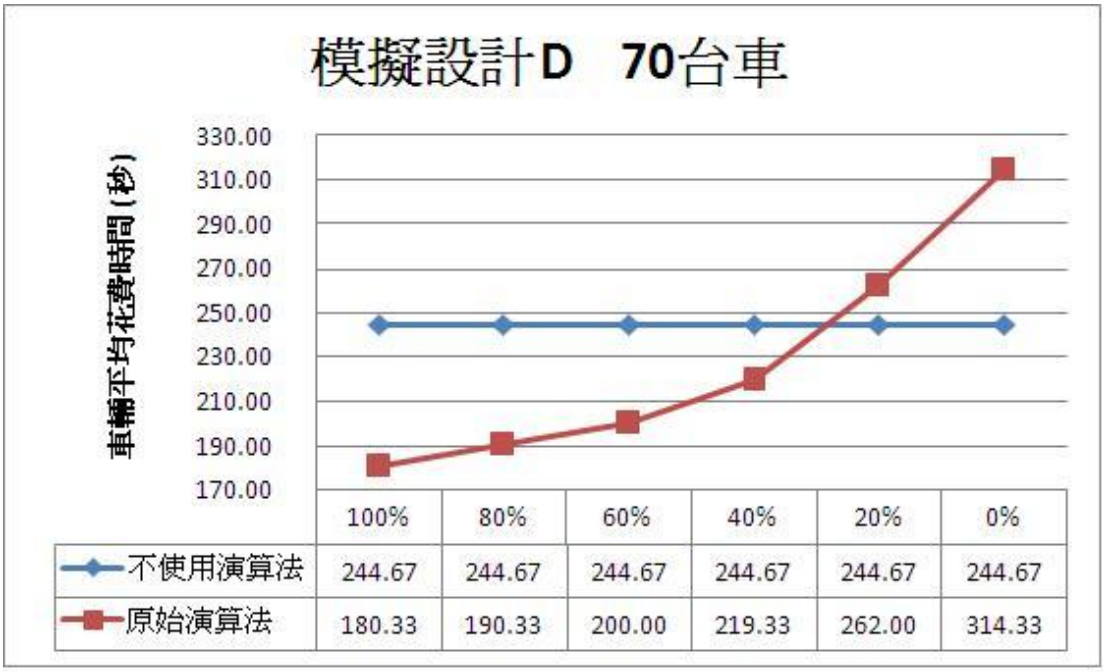


圖 37. 模擬設計 D 原始演算法 70 台車模擬結果

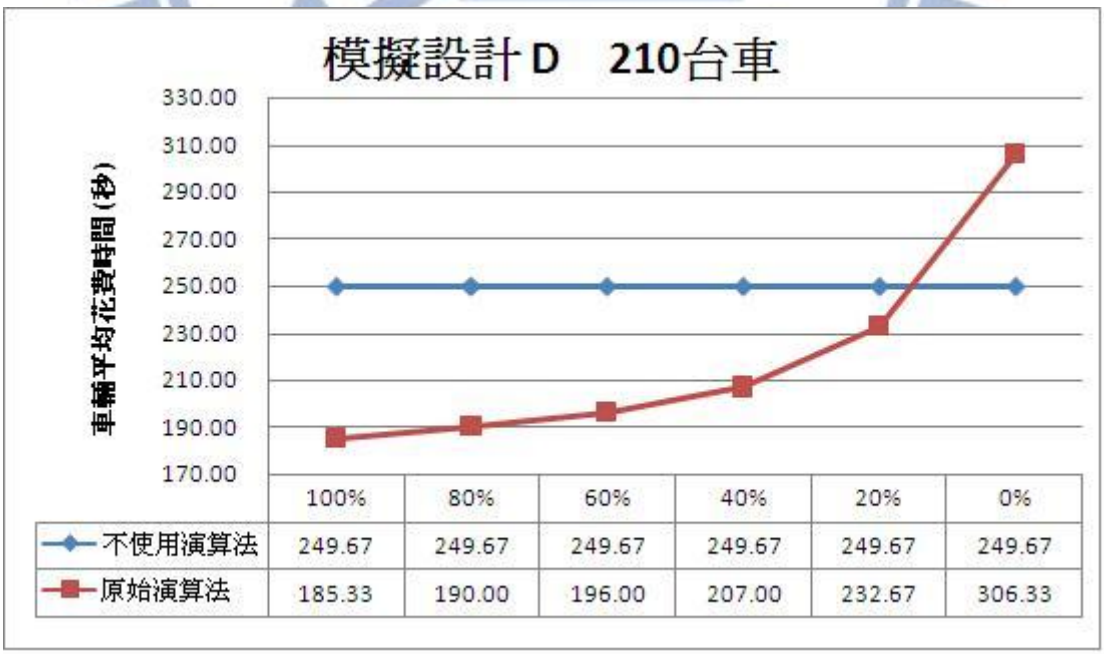
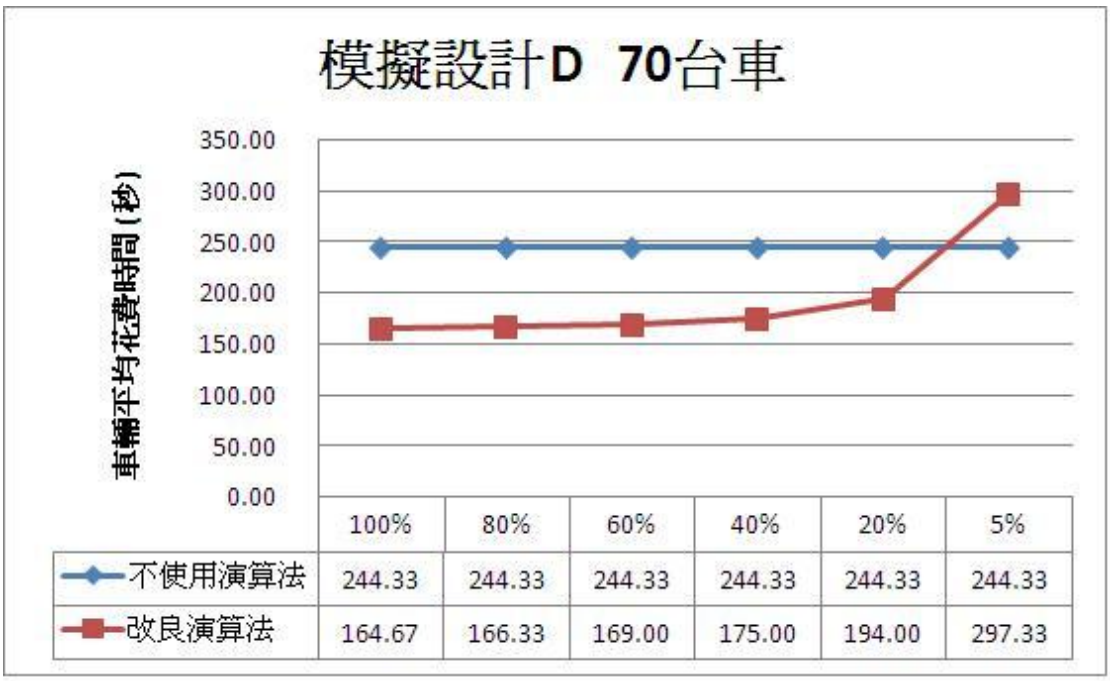
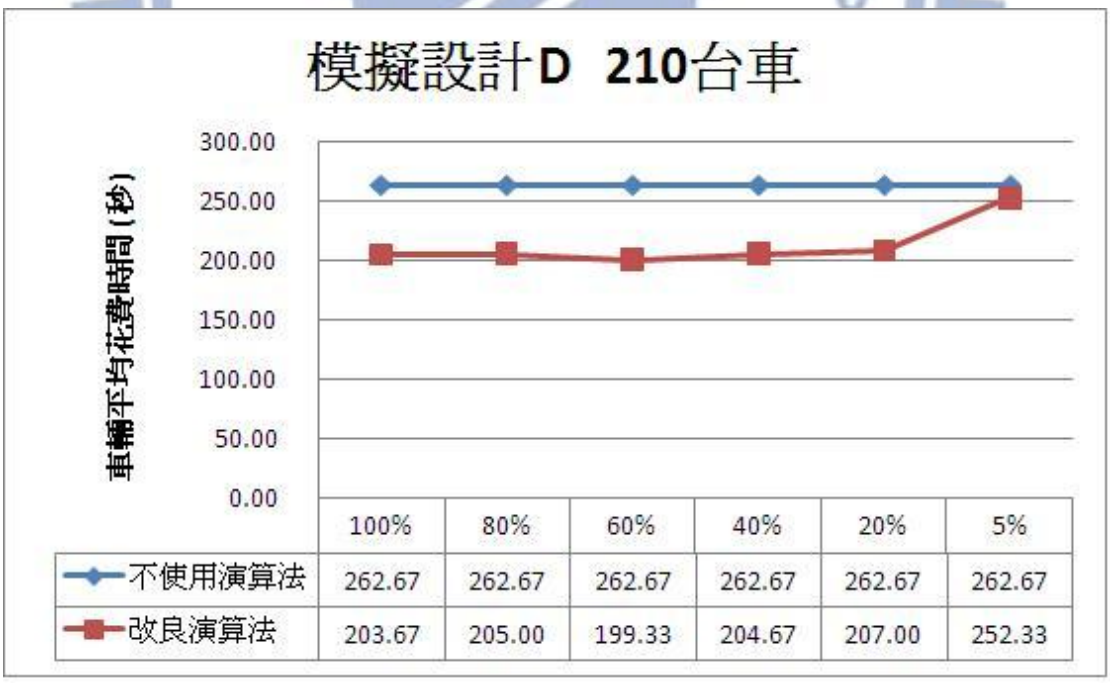


圖 38. 模擬設計 D 原始演算法 210 台車模擬結果



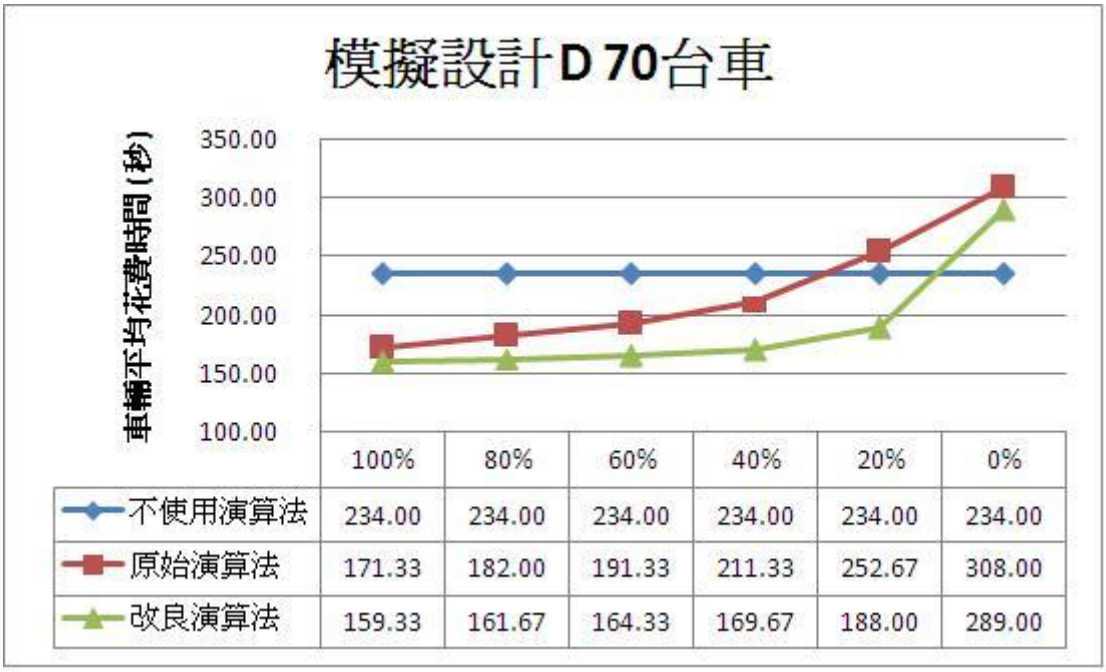
	100%	80%	60%	40%	20%	5%
不使用演算法	244.33	244.33	244.33	244.33	244.33	244.33
改良演算法	164.67	166.33	169.00	175.00	194.00	297.33

圖 39. 模擬設計 D 改良演算法 70 台車模擬結果



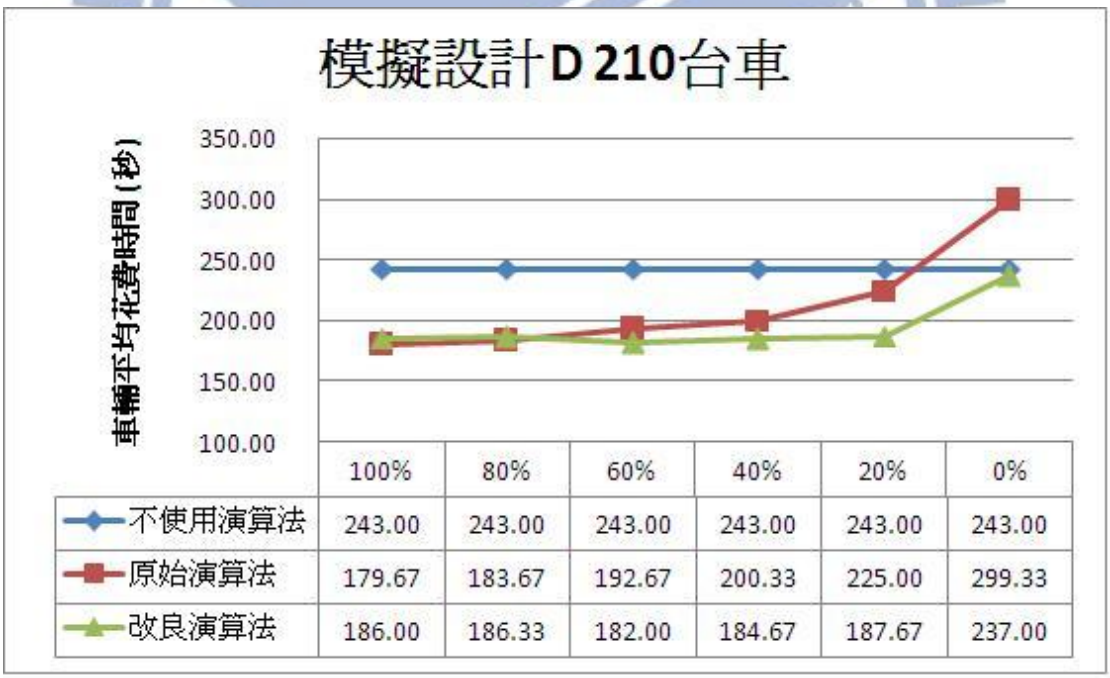
	100%	80%	60%	40%	20%	5%
不使用演算法	262.67	262.67	262.67	262.67	262.67	262.67
改良演算法	203.67	205.00	199.33	204.67	207.00	252.33

圖 40. 模擬設計 D 改良演算法 210 台車模擬結果



	100%	80%	60%	40%	20%	0%
不使用演算法	234.00	234.00	234.00	234.00	234.00	234.00
原始演算法	171.33	182.00	191.33	211.33	252.67	308.00
改良演算法	159.33	161.67	164.33	169.67	188.00	289.00

圖 41.模擬設計 D 70 台車綜合比較模擬結果



	100%	80%	60%	40%	20%	0%
不使用演算法	243.00	243.00	243.00	243.00	243.00	243.00
原始演算法	179.67	183.67	192.67	200.33	225.00	299.33
改良演算法	186.00	186.33	182.00	184.67	187.67	237.00

圖 42.模擬設計 D 210 台車綜合比較模擬結果

六、結　論

在這個資訊產業迅速發展的時代，智慧型交通運輸系統也迅速發展，智慧型的交通號誌燈控制也是相當重要的一環，交通號誌控制又分為固定時間式的交通號誌控制與動態時間式的交通號誌控制。

在本論文中，我們選擇動態時間式的交通號誌控制進行深入的研究與分析，透過不同的車流拓樸、時間間隔以及車流資訊蒐集完整程度，對動態時間式的交通號誌控制演算法進行觀察與分析，藉此了解演算法的運作效能，並觀察演算法在不同環境下的模擬結果，提出改良的建議，然後透過模擬案例證實，我們的改良確實有效。

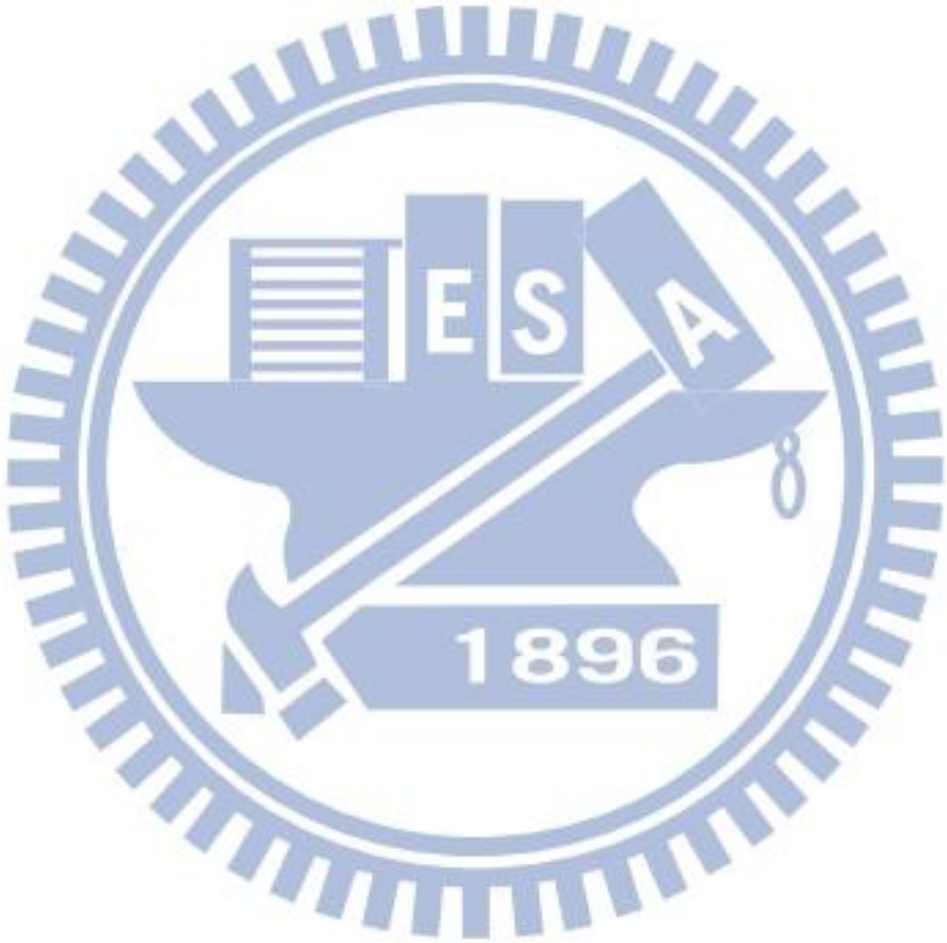
我們將本論文中，經由觀察、模擬以及改良原本演算法後所得到的結果整理成以下的結論：

1. 在原本的演算法運作下，比起沒有使用演算法的交通號誌燈，確實能有效的減少車輛到達目的地的平均花費時間，而我們改良過後的演算法，比起原始的演算法，在車流密度低的時候，能更有效的減少車輛到達目的地的平均花費時間，而在車流密度高的時候，也能維持跟原本演算法差不多的效能。

2. 在原本的演算法運作下，會有少數車輛增加行車延遲時間，而在我們改良後的演算法中，能確實減少這些被犧牲車輛的延遲時間，提升整體的公平性。

3. 在我們使用不同時間間隔的分析下，發現原本動態時間評估車況的演算法，評估的時間越短越好，但是太短暫又會有頻繁變換交通號誌的問題，所以以十秒左右為評估車況的時間間隔，能帶來比較大的利益。

4. 在車流資訊蒐集不足的環境中，原本演算法的延遲時間會隨著資訊蒐集程度的減少而快速增加，而我們改良過後的演算法在車流資訊蒐集減少到 20%以前都還能維持不錯的效能表現，所以我們改良過的演算法比起原本的演算法更能承受資訊的遺失。



七、未來工作

在本論文中，我們由不同的車流拓樸、不同的時間間隔以及不同的資訊蒐集程度去了解動態交通號誌燈控制演算法的特性與效能，但是還有一些議題是論文中還可以繼續延伸研究的，在以下詳述之：

1. 在資訊蒐集的部分，因為在我們論文中是假設汽車有會定期向路旁的 RSU 更新自己的位置與行車速度，所以每個交通號誌燈都可以確實掌握車流資訊，這部分未來可以使用真實的網路傳輸讓汽車定期更新資訊，這樣掌握的車流資訊會更貼近於真實現況。

2. 我們研究與改進的動態交通號誌控制演算法是屬於尋找區域最佳解，每個交通號誌燈只根據自己附近的車流判斷該如何配給紅綠燈，但如果每台車都會更新資訊的話，應該也可以把目的地跟未來移動路線一起更新，如此一來交通號誌燈能掌握底下車流未來會前往哪個交通號誌燈，應該可以做到更全面化的交通號誌控制優化。

參考文獻

[1] LI Jinyuan, PAN Xin, WANG Xiqin, “State-Space Equations and the First-Phase Algorithm for Signal Control of Single Intersections”, Tsinghua Science & Technology Volume 12, Issue 2, April 2007, Pages 231-235.

[2] Dennis I. Robertson, R. David Bretherton, “Optimizing Networks of Traffic Signals in Real Time – The SCOOT Method”, IEEE Transactions on Vehicular Technology, vol. 40, no. 1, February 1991.

[3] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, “Optimizing traffic lights in a cellular automaton model for city traffic, *Phys. Rev. E*, vol. 64, no. 5, p. 056132, Oct. 2001.

[4] D.-W. Huang and W.-N. Huang, “Traffic signal synchronization,” Phys. Rev. E, vol. 67, no. 5, p. 056124, 2003.

[5] C. Gershenson, “Self-organizing traffic lights,” Complex Syst., vol. 16, no. 1, pp. 29–53, 2005.

[6] Ehsan Azimirad, Naser Pariz, M. Bagher Naghibi Sistani, “A Novel Fuzzy Model and Control of Single Intersection at Urban Traffic Network” IEEE Systems Journal, vol. 4 no. 1 March 2010.

[7] Lai Guan Rhung, Azura Che Soh, Ribhan Zafira Abdul Rahman, Mohd Khair Hassan, “Fuzzy Traffic Light Controller Using Sugeno Method for Isolated Intersection”, Proceedings of 2009 IEEE Student Conference on Research and Development (SCORed 2009), 16-18 Nov’09, UPM Serdang, Malaysia.

[8] D.T Dissanayake, S.M.R Senanayake, H.K.D.W.M.M.R Divarathne, B.G.L.T. Samaranayake, “Real-Time Dynamic Traffic Light Timing Adaptation Algorithm and Simulation Software”, Fourth International Conference on Industrial and

Information Systems, ICIIS 2009, 28 – 31 December 2009, Sri LanKa.

[9] NCTUns Network Simulator and Emulator is available at http://nsl10.cs.nctu.edu.tw/

[10] S.Y. Wang, C.L. Chou, C.H. Huang, C.C. Hwang, Z.M. Yang, C.C. Chiou, and C.C. Lin, "The Design and Implementation of the NCTUns 1.0 Network Simulator," Computer Network, vol. 42, no. 2, pp. 175-197, June 2003.

[11] S.Y. Wang, C.L. Chou, and C.C. Lin, "The GUI User Manual for the NCTUns 6.0 Network Simulator and Emulator" is available at http://nsl10.csie.nctu.edu.tw/support/documentation/GUIManual.pdf

[12] S.Y. Wang, C.L. Chou, C.C. Lin, and C.H. Huang, "The Protocol Developer Manual for the NCTUns 6.0 Network Simulator and Emulator" is available at http://nsl10.csie.nctu.edu.tw/support/documentation/DeveloperManual.pdf