基於車輛網路分享的車輛資訊之動態共乘系統

張英超	洪似妮	廖啟盛	林佳陞
國立彰化師範大學	國立彰化師範大學	國立彰化師範大學	國立彰化師範大學
資訊工程學系	資訊工程學系	資訊工程學系	資訊工程學系
icchang@cc.ncue.edu.tw	sunnyhung9943@gmail.com	suboy@hotmail.com.tw	jasonlin2719@gmail.com

摘要

現有的共乘系統中，大多數是由司機在共乘平台上公開自己行程，乘客再根據司機公開的資訊選擇適合自己行程。但此做法不容易選到適合自己的司機或乘客，而雙方的個人偏好也不同，無法獲得良好的配對與路徑規劃。此外，在沒有即時交通路況資訊的狀態下，傳統的共乘系統在時間限制內無法找到最省油錢的行駛路徑以及最佳的共乘配對組合。本論文所研究的共乘系統會基於 VBA*演算法計算出來的結果提出「共乘配對演算法」、「油錢分攤演算法」與「配對範圍初始與擴大演算法」，這樣可以大幅的減少共乘搜尋的範圍。最後，乘客與司機可以找出符合雙方需求的最佳配對結果並即時規劃出共乘路線，達到降低油費與節能省碳之目的。

關鍵詞：車輛無線網路、VBA*路徑規劃演算法、共乘配對演算法、油錢分攤演算法、分享即時路況資訊

1. 系統簡介

近年來節能省碳已經是趨勢，共乘議題的主要設計在於如何根據司機與乘客的位置與條件設定（如吸煙、性別等）去做配對，而得到的共乘順序是否能在司機與所有乘客的需求時間限制（如最晚到達起點、終點的時間）內到達各自目的地，也必須考量是否使司機與乘客分擔的油費低於原本各自單獨開車/坐車的油錢花費，規劃出最省油費的路徑。共乘的問題不只在於共乘配對，「路徑規畫」也是一個重要的問題。近年來車輛網路(Vehicular ad hoc network, VANET)技術逐漸成熟，車輛能與鄰近車輛建立動態的車輛網路分享或者傳遞各種資訊。若共乘規劃時能針對實際的道路行車速度與路況，得出相對的油費支出，調整共乘承載順序與路線，滿足司機與乘客的時間需求限制，將達到共乘油錢花費最低的目標。

本計畫貢獻有以下幾點貢獻：

1. 設計車輛蒐集即時路況的作法，透過車輛網路 VANET 隨時與鄰近車輛交換彼此擁有的道路訊息，並預估每個路段未來的行車速度與行駛於此路段的油費。
2. 利用預估各路段行駛的油費，結合我們過去所提出的 VANET-based A* (VBA*)路徑動態規劃演算法啟發函式[5]，規劃乘客/司機任兩點之間最省油錢的行駛路線。
3. 使用上述 VBA*規劃的結果，設計「共乘配對演算法」與「油錢分攤演算法」，依照乘客及司機的個人化需求找出最佳的共乘配對方式與承載順序，滿足乘客及司機到達起點與終點的時間限制，使得乘客與司機都能比自己單獨開車花費還要少的油費，而且是所有承載順序中油錢最低者。
4. 根據司機的起、終點，決定「初始配對範圍」，減少共乘規劃需比對的乘客人數，降低比對次數。提出有限次數的「擴大配對範圍演算法」，在車輛座位的限制下盡量找到最多的共乘乘客。
5. 提出「起終點順向與路口油費的檢測」，縮小乘客的搜尋範圍，降低演算法複雜度。

2. 文獻探討

本論文主要著重在「即時動態」的車輛網路環境下之規劃最佳路徑的方式並探討共乘系統的規劃與設計。因為一般的共乘論文或者系統提到規劃路徑的議題少之又少，若共乘系統未將現實路況的因素考量進去，即便共乘系統設計的再周全，最終也不一定適用於現實路況中。以下介紹過去相關之共乘系統：

2.1 依照歷史軌跡之共乘論文介紹

歷史軌跡預測主要是依照每個人的行駛路線去做統計，甚至計算出乘客到每一個節點地方的機率為多少，接著再依照這個機率數值加以利用做為共乘配對的規劃條件中。這一類方法的論文雖然有歷史軌跡可以做依據，也可以預測機率值，但是目前共乘系統還尚未普及化以及隱私考量的情況下，乘客的軌跡資訊不

容易事先取得，再加上每個人的行走軌跡變化量難以估算，很容易會造成共乘上的誤差。

- Potential Travel Distance (PTD) [10]

[10]是利用過去的歷史資訊去評估乘客的路線，算出每個載客點會接到客人的機率進行評估去載客。作者提出了一個 Potential Travel Distance(PTD)評估每個候選路徑，並且提出了一個名為 LCP 的路徑建議演算法，它是利用 PTD 函數的單調性。此方法的車輛是在固定的點，不知道事先是否有乘客在等待，而時間需求、路線規劃、即時路況等等並沒有考慮到。

- Parking-lot-assisted carpooling (PASS) [6]

[6]提出了一個基於停車場概念的共乘方法，以停車場作為一個中央數據庫，運用車輛無線隨意網路，讓汽車和司機溝通並且交換資訊。然而這個方法卻只限用於停車場裡面的車輛才可以進行共乘，在行駛中的車輛和路邊的乘客之間並不列為共乘的考慮中。此篇論文運用車輛無線隨意網路的資料傳遞，雖然有提出路由軌跡樹，但並沒有詳細說明歷史軌跡資訊是如何產生的。

- Latent Dirichlet Allocation (LDA) [8]

[8]使用兩個不同的資料庫分析行動資訊，而這篇論文的主要工作在於用戶端可以自動辨識最常拜訪的地點，也可以用資料探勘技術計算常去的地點機率，並在最後將結果運用於共乘系統。此論文運用了歷史資訊去做共乘配對，但是缺乏了確切的共乘需求時間，也缺乏即時路況傳遞以及油費分攤等解法。

2.2 依照配對需求之共乘論文介紹

直接依照乘客條件需求去做共乘配對。第一種配對需求是配對偏好的個人需求，例如：性別、抽菸、寵物等等。第二種配對需求是乘客所要求的起始點的時間以及到達目的地的時間。這部分的配對需求除了要滿足乘客與司機雙方的個人需求之外，也必須要考量到現實的車程時間以及需求時間的考量。

- I-CAP [4]

[4]中提出了詳細的配對演算法過程，並用距離和時間及駕駛者的行為來計算出每一個屬性以及參數值的機率，再依據機率去選擇適合的配對。雖然這篇論文提出的很多項目的參數參考，也有設計更正因子的公式，但僅僅只有說明配對演算法的過程，而這個配對過程也不夠完整，參數值加權的依據沒有詳細的說明，而歷史資訊也沒有詳細說明是如何參考的。若考慮一些共乘現實面的問題，此篇論文並沒有考量到目的地的到達時間，缺乏路徑規劃的設計及研究，也缺少了即時路況的考量。

2.3 轉乘與旅遊規劃之論文介紹

轉乘的定義是乘客與司機共同搭乘一段路徑後，乘客端再藉由其他司機的接送，不斷的轉乘到達目的地。雖然轉乘和共乘的定義和基本條件不太相同，不過需要考量的因素是類似的，一樣必須考量路程規劃以及需求時間。

- DOMARTiC [7]

[7]提出了動態的轉乘服務，此方法是根據乘客的歷史資訊去估算預測路徑，再根據與司機路徑重疊的部分去分配轉乘，且只在固定點才可接送乘客。此篇論文在配對部分缺乏說明，只針對歷史資訊路徑去做規劃是不夠準確的，缺少了時間需求和實際路況的考量。

2.4 考量油費之共乘論文介紹

- TSP [9]

Intelligent Transportation System(ITS)比起傳統的運輸建設增加了資訊和車輛網路通訊技術(VANET)，所以可以預估即時交通的情況且可以提供一些其他的服務，例如：導航、規劃路徑等來改善行車效率。[9]提出 TSP 方法就有運用 ITS 的技術得到道路資訊，並且基於效益提出了一個共乘概念，考慮到配對後行駛的耗油量且可以得到較快的回應時間。此篇論文提出了一個九宮格的概念如圖 1，是用共乘的乘客終點位置與九宮格附近的司機終點位置來做配對，圖 1 為一個 5×5 的一個子區域圖，A 點表示為乘客的起點，而 B 點表示為乘客的目的地。此步驟為取出乘客目的地附近的車子成為候選車輛，而以上圖為例，13、14、15、18、20、23、24、25 為候選車輛。接著再將候選車子來做共乘前後的油耗效能來做篩選，如圖 1。這篇論文雖然有運用車速和油耗的概念，也有提出即時路況塞車問題，但卻沒有考量到司機與乘客的起點位置距離有多遠，也沒有把配對的需求時間問題以及油耗的分攤考量進去，更沒有詳細說明即時路況是如何傳遞和交換資訊的。

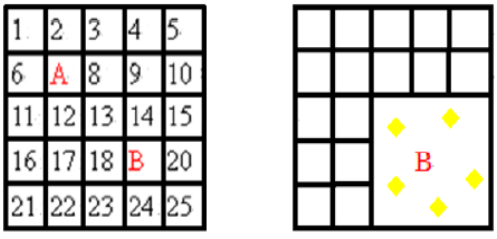


圖 1、TSP 5×5 的一個九宮格概念圖

2.5 目前共乘網站之介紹

共乘王[1]平台需申請會員，但會員有分免費會員跟環保會員兩種，環保會員為付費會員，其刊登之共乘資訊將較易被其他會員看見且其所能收藏之共乘路線將不受數量限制。可欲先設定金錢分攤方法，油錢的分攤方法是由司機、乘客自行商議。

BlueNet[3]共乘 APP 使用 Android 平台且提供搭乘車輛的使用者一個簡單且方便的即時共乘途徑，缺點是沒有明確的金錢分攤方法。總結以上相關工作之缺點：

1. 只參考歷史軌跡預測路線的方法沒有考慮即時路況，不適用於車輛臨時加入共乘。
2. 部分方法沒有考慮司機與乘客的起、終點時間需求，而部分有考慮需求時間的論文多半是以一個時間區段表示，無法確定確切乘載時間，不易安排後面的乘載順序。
3. 大部分方法都缺乏考量即時路況的傳遞，可能發生路段塞車而無法避免的問題。
4. 大部分方法都有提到共乘，大多只能選擇最短路徑，只有少數有提到共乘路線需要降低油費，但並未提出如何尋找最省油費路線的方法。
5. 傳統的共乘方法是依照同一個起終點一起上下車作為共乘，此方法只能乘載兩個人，而少了中間路段的共乘，並沒有達到多人共乘的效益。
6. 有的論文提出油費的考量，但並未詳細的提出油錢分攤的機制，也未討論分攤後的油錢是否低於自己行車的油錢，能否形成成功的共乘配對與承載順序。

因上述幾點，大部分的相關工作並不適用於動態即時的共乘環境。

3. 研究方法

司機與乘客之間的共乘乘載順序在司機與每一個乘客都有不同的起終點時間限制與每條道路不同時間下不同路況（壅塞情況行駛速度、時間、相對油耗等）的條件下，是個困難的問題。本系統結合 VBA*的作法，路上的車輛在相遇時，互相交換自己蒐集的路段的車速資訊，也透過車輛網路 Vehicular ad hoc network (VANET) Vehicular to Roadside Unit (V2R)的方式，不斷的轉傳道路資訊到附近的 Roadside Unit (RSU)，再接著透過有線網路將路況資訊傳給「配對伺服器」，藉此估計出這些路段的未來行駛速度。

3.1 VBA*導航模式與評估函式設計

由於傳統共乘系統只能根據靜態道路資訊（路段長度、速限等）計算車輛導航行駛的時間，作為共乘的依據，無法掌握即時路況，大幅降低共乘系統的準確度與使用價值。因此本計畫結合「VANET-based A* (VBA*)導航規劃演算法」與「共乘演算法」避免上述問題。

3.1.1 車輛產生即時道路資訊

VBA*首先讓車輛蒐集行駛過的路段的即時道路資訊（此路段在某時段的平均行駛速度等），並分析 Google Maps 所提供的即時路況資訊，加強即時道路資訊的完整與可靠性：車輛產生即時道路資訊流程如圖 2，車輛行駛於道路上時，會不斷透過 GPS 定位及電子地圖資料庫判斷車輛位置，並記錄自身行駛速度，當車輛進入下一條道路時，會將上一條道路所記錄速度資訊，以算術平均得出車輛行駛的平均速度，使用上述格式將其記錄下來，得出該道路的即時資訊（道路編號、紀錄時間、平均速度）。此外，在 Google Map 中提供了即時道路資訊系統，以綠色、黃色、紅色及紅黑間隔所標示的道路，分別代表速度由速度較快至速度較慢，可透過分析 Google Maps 即時路況資訊功能獲得大型道路，如國道、快速道路、省道及部分路段的即時道路資訊。Google Maps 即時路況資訊並不完整，僅能作為輔助。

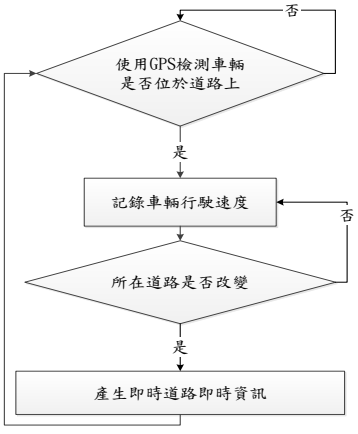


圖 2、車輛產生即時道路資訊流程圖

3.1.2 VBA*推估即時道路資訊

共乘規劃伺服器將搭配車輛分享與 Google Maps 提供的即時路況資訊，使用指數加權移動平均公式計算目前時間點的所有道路的推估車速資訊，流程如圖 3 所示。

首先將即時道路資訊紀錄時間轉換為所在的時間區間，以方便後續計算。配合 Google Maps 所提供的即時路況資訊，將 24 小時以每

15 分鐘劃分成 96 個時間區間。第二步驟將劃分在相同時間區間內的多筆相同道路資訊，使用算術平均速計算出算術平均數，代表每一個時間區間的即時道路資訊。第三步驟將 96 個時間區間，使用指數加權移動平均公式如公式(1)，利用過去歷史的道路資訊計算目前時間點的推估道路資訊，時間較遠的資訊對推估值影響較小，時間較近對推估值影響較大，希望達成更準確的預估資訊。第四步驟中，若 Google Maps 即時路況系統有提供該道路歷史與即時路況資訊，則再將目前時間點的道路資訊與上一步驟所推得的推估資訊進行算術平均，加強即時道路資訊預估的準確與完整性。

$$S_t = \alpha \times Y_{t-1} + (1-\alpha) \times S_{t-1} \quad (1)$$

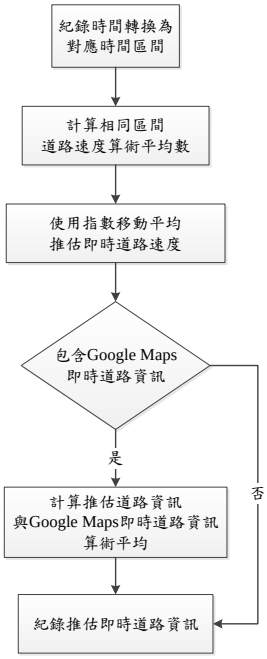


圖 3、VBA*推估即時道路資訊流程圖

3.1.3 VBA*演算法最省油耗啟發函式

使用 VBA*演算法，以「最少油費」為導航路徑規劃目標，求出兩節點（司機與乘客的上下車地點）之間最佳的行駛路徑的距離、時間與油費，作為後續「共乘演算法」計算最佳共乘路徑的依據：

$$F(n) = \sum_{i=0}^{n}(L_i \times O(S_i)) + \frac{H(n)\times O(S_n)}{(2+\cos\theta_n)} \quad (2)$$

公式（2）為改良的 VBA*演算法最省油耗啟發函式。將原始 A*演算法公式結合車輛行駛間交換的即時路段速度資訊，代入速度與油耗關係表，如圖 4，得出油耗數值。接著將原本 A*公式的實際成本部分轉換為使用路段長度 Li 乘上路段預估速度 Si 所對應的油耗值 O(Si)，得出已知路段累積的油耗成本；推估成本部分，將推估路段長度 H(n)乘上目前迭代所到達的路段 n 的推估速度 Sn 所對應的油耗值 O(Si)。然而在現實中，如圖 5，推估路段 H(n)所指出的方向，在實際中並不一定存在此方向的道路，因此透過除以三角函數$\cos\theta_n$，可使得與路段 n 夾角越大的路段 n+1，所計算出的成本越高。因此 VBA*演算法在挑選最佳路段時，會較容易挑選路段 n+1 與終點方向接近的路段。但在計算時，$\cos\theta_n$值介於-1 與 1，造成在計算推估路段時產生負值問題，因此將$\cos\theta_n$值皆加上 2，使得值介於 1 與 3，去除負值問題，並保留角度越大值會越大的特性。最後 VBA*演算法可以得出抵達終點的最佳行駛路徑與所需的最少油耗，乘上單位油耗的費用後，就可以算出起點與終點預估的最小油費。

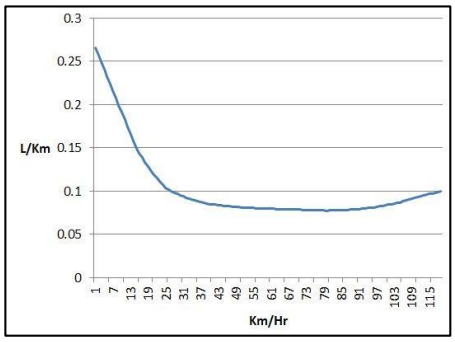


圖 4、速度與油耗關係圖

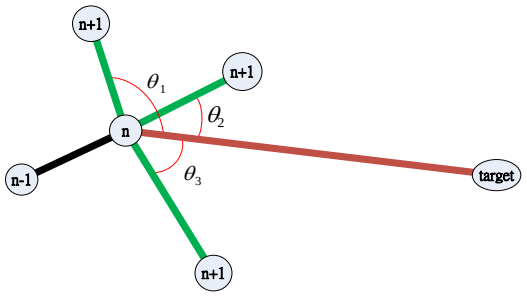


圖 5、路段與推估路段夾角示意圖

3.2 共乘系統架構與演算法設計

由於每條道路在不同時間下，路況（壅塞情況行駛速度、時間、相對油耗等）也不相同，加上司機與每個乘客都有不同的起終點時間限制，如何決定司機與乘客之間最少油錢的共乘乘載順序是個困難的問題。為了估計未來的道路行駛速度，本研究結合 VBA*的作法，路上的車輛在相遇時，互相交換自己蒐集的路段的車速資訊，也透過車輛網路 Vehicular ad hoc network (VANET) Vehicular to Roadside Unit (V2R)的方式，不斷的轉傳車輛蒐集的路況資訊到附近的 Roadside Unit (RSU)，再接著透過有線網路將路況資訊傳給「配對伺服器」，藉此估計出這些路段的未來行駛速度。由於共乘是為了降低司機與乘客的油費，配對伺服器設定「最小油費」為 VBA*的路徑導航目標，求

出任兩路段之間最佳的行駛時間、距離與油費的三個陣列(TimeCostArray[][], DistCostArray [][], OilCostArray [][])。

當車輛與乘客有共乘需求時，同樣透過 V2R 的方式發送這些需求資訊到最近的 RSU，彙整到配對伺服器，然後利用本演算法計算出最佳的乘載順序、共乘路線與分擔油費等。這樣的共乘資料量集中在配對伺服器，可以避免分散式做法下多個司機可能同時計算出接送同一個乘客的競爭現象，還需要透過 VANET 進行額外的協商機制。

本論文使用的符號與定義如表 1 所列：

代表符號	定義
$D_i^+(DX_i^+, DY_i^+)$	第i個的司機起點(x, y 座標)
$D_i^-(DX_i^-, DY_i^-)$	第i個的司機終點(x, y 座標)
DT_i^+	司機的起點需求時間
DT_i^-	司機的終點需求時間
$P_i^+(PX_i^+, PY_i^+)$	第i個乘客的起點(x, y 座標)
$P_i^-(PX_i^-, PY_i^-)$	第i個乘客的終點(x, y 座標)
PT_i^+	第 i個乘客的起點需求時間
PT_i^-	第 i個乘客的終點需求時間
DX_i^{min},DX_i^{max}	司機起終點 x 座標的較小、大值
DY_i^{min},DY_i^{max}	司機起終點 y 座標的較小、大值
E_c	目前範圍擴大次數
E_{MAX}	範圍擴大最大次數限制
P_{MAX}	乘載人數的最大上限
P_c	目前乘載人數
Er	搜尋乘客的擴大範圍大小
Rr	搜尋乘客的軌跡範圍大小

圖 6 為汽車共乘環境示意圖，D1、D2 為司機 1 和司機 2；p1 和 p2 為乘客 1 和乘客 2。圖 7 為整個共乘系統的架構圖，分成三大部分，分別為司機端、乘客端以及配對伺服器。配對伺服器還包含了三個主要演算法：「共乘乘載順序演算法」、「配對範圍擴大演算法」以及「油錢分攤演算法」。

圖 6、汽車共乘環境示意圖

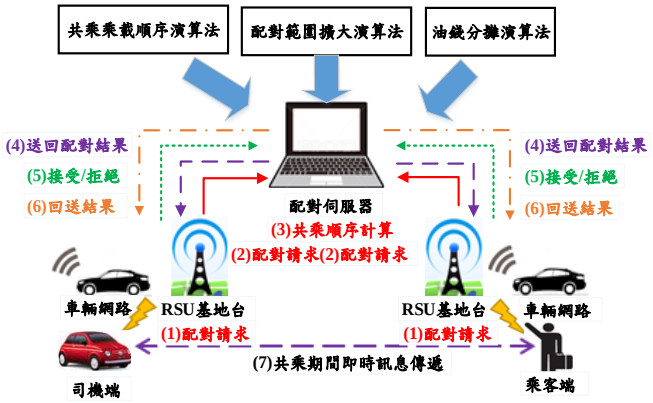


圖 7、共乘系統訊息流程

在此共乘系統當中，設定的規則如下：

規則 1：

【多人共乘，所有乘客先上車後才能下車，盡量坐滿車輛的最多座位(P_{MAX})，為司機等待乘客的作法】此系統考慮所有乘客先上車再下車的情況，納入一個乘客的起點的同時，終點也要同時考慮進去。

規則 2：

【設定初始配對範圍與擴大範圍的最大次數(E_{MAX})，減少需要判斷的乘客人數】為了減少共乘配對比對的乘客數目，需要根據司機的起點與終點位置設定「初始配對範圍」。此外，由於乘客可以半途上下車，所以共乘車輛可能會常常有尚未達到最大乘載人數的情況，所以設計了「配對範圍擴大演算法」，在乘客和司機的需求時間內，可以有限度的去擴大範圍，搜尋最多符合的共乘乘客，降低分擔的油費。

規則 3：

【滿足司機與每位共乘乘客的上下車時間需求】

規則 3.1：

【乘客 i 的(終點的需求時間(PT_i^-)−起點的需求時間(PT_i^+))＞自己直接行駛(沒有共乘的行程時間)+門檻值θ】此規則表示乘客接受的共乘行車時間必須大於自己直接從起點到終點的行程時間加上一個

門檻值，門檻值θ為一個固定值。

規則 3.2：

【司機與每位乘客共乘後到達終點的實際時間≤司機與每位乘客各自的終點時間】

規則 4：

【根據油錢分擔演算法算出的司機與所有共乘乘客在此共乘乘載順序所分攤的油費≤自己直接行駛的油費】

目標式：

【如何在配對範圍與司機/每個乘客的時間限制內，使用 VBA*最低油費的路徑規劃模式，算出司機與乘客分擔油費最低的共乘承載順序、分擔油費與最佳路徑】

3.2.1 初始配對範圍演算法設計

Input：

司機端的起點經緯度(DX_i^+, DY_i^+)、司機端的終點經緯度(DX_i^-, DY_i^-)、所有乘客端的起點經緯度(PX_i^+, PY_i^+)、所有乘客端的終點經緯度(PX_i^-, PY_i^-)。

Output:

此司機所考量接送乘客的矩形範圍大小 Range[2][2]、範圍內的共乘乘客、範圍內的共乘乘客的起點經緯度(PX_i^+, PY_i^+) 、範圍內的共乘乘客的終點經緯度(PX_i^-, PY_i^-)。如圖 6 中司機 D1 的初始配對範圍為藍色矩形範圍，而司機 D2 的初始配對範圍為紅色矩形範圍。

STEP 1：

如圖 8，若司機的起終點連線不是水平或者垂直線，司機取的乘載範圍大小為司機的起、終點對角線連接所規劃出來的矩形範圍的左下角為 (DX_i^{min}, DY_i^{min})，右上角為(DX_i^{max}, DY_i^{max})。若司機的起終點連線為水平(如圖 9)或垂直線(如圖 10)，就將司機的起終點朝著垂直或者水平向兩端延伸司機起終點連線總長度 L 的 1/2r，並以規劃出的矩形範圍，如圖 9 起終點連線為水平的左下角為 $(DX_i^{min}, DY_i^- - L/2r)$，右上角為$(DX_i^{max}, DY_i^- + L/2r)$，以及如圖 10 起終點連線為垂直的左下角為$(DX_i^- - L/2r, DY_i^{max})$，右上角為$(DX_i^- + L/2r, DY_i^{min})$作為乘載範圍大小。

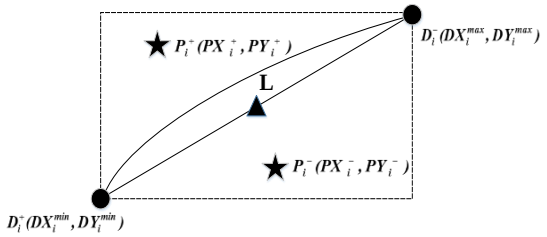


圖 8、當司機的起終點對角線為矩形時，所規劃出來的乘載矩形範圍大小

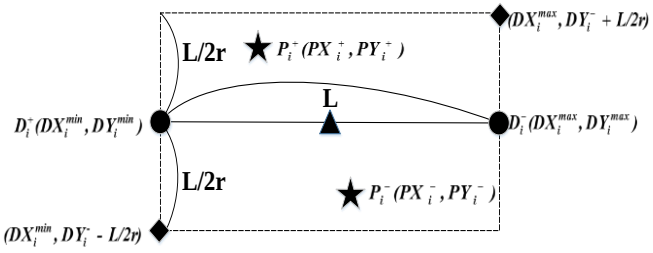


圖 9、當司機的起終點連線為水平線時，向垂直兩端各延伸連線長度 L 的1/2r規劃出乘載矩形範圍大小

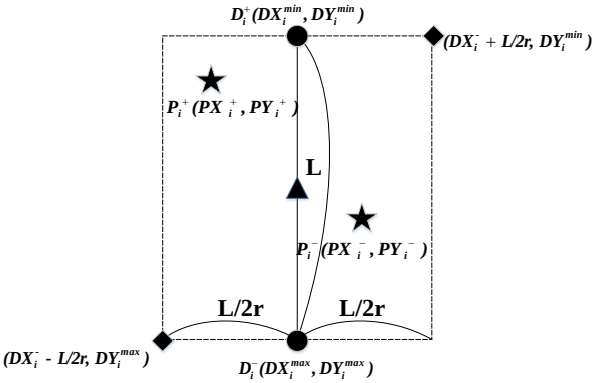


圖 10、當司機的起終點連線為垂直線時，向水平兩端延伸L/2r規劃出乘載範圍大小

STEP 2：

若所規劃出來的矩形範圍內沒有搜尋到任何的乘客需要接送，就將此矩形範圍依照司機起終點連線總長度L′的 1/2r，直接以此對角線向外延伸，擴大矩形乘載範圍如圖 11。若還是沒有搜尋到任何一位乘客需要接送，就不斷擴大直到E_{Max}次為止。

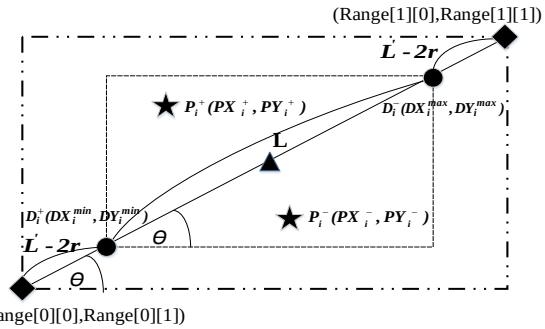


圖 11、當司機的起終點連線為對角線，再次擴大範圍時，對角線向兩端延伸$L'/2r$，規劃出乘載矩形範圍大小

3.2.2 共乘乘載順序演算法設計

執行「共乘乘載範圍演算法」後，初步的求出司機的乘載範圍，雖然共乘乘載範圍演算法大幅的縮小了整體配對乘客的數量，但若是所有搜尋範圍內的乘客（數目為 n）都要判斷能否滿足共乘的時間與油費的條件，所有可能的排列組合數目可能太大：假設某車輛最多的共乘座位數是P_{MAX}，所有共乘承載順序中包含 1~P_{MAX}位共乘乘客的所有排列順序。1 位共乘乘客的所有排列順序（$D^+ \rightarrow Pi^+ \rightarrow Pi^- \rightarrow D^-$）有$\binom{n}{1}$種，而$P_{MAX}$位共乘乘客的所有排列順序（$D^+ \rightarrow (P_1^+ \rightarrow P_2^+ \ldots \rightarrow P_{P_{MAX}}^+) \rightarrow (P_i^- \rightarrow P_j^- \ldots \rightarrow P_k^-) \rightarrow D^-$）有$\binom{n}{P_{MAX}}$種共乘乘客選法與各（$P_{MAX}!$）種上車與下車順序，共$\binom{n}{P_{MAX}} \times (P_{MAX}!)^2$種排列順序。因此全部要比對的共乘順序有$\sum_{i=1}^{P_{MAX}} \binom{n}{i} \times (i!)^2$種，當 n 大時，這是相當大的次數。於是本計畫提出了進一步的縮小比對範圍的方法--「軌跡範圍搜尋乘客」，減少需要比對能否共乘的乘客數目 n，其主要概念是要避免比對「偏離目前的共乘路線太遠的乘客」（如下圖 12 乘客 P1），避免司機繞路太遠去接送乘客，不僅可以降低分擔的油費，也可以降低共乘演算法的計算量。每條共乘路線紀錄經過的每一個路口，紀錄方式如$Di^+ \rightarrow Di^-$=[$I_1, I_2, I_3, \ldots \ldots$]。在目前的共乘路線下，若司機能由某個行經路口繞路去一位新的乘客的起點與終點所額外花費的油錢都小於一個給定的油錢門檻值O_{TH}，則可以將這位新的乘客加入後續共乘的時間與油費判斷。反之，這位乘客則會被刪除。為了快速找出需要判斷的路口，本計畫提出以下方法：

1. 以某位新乘客的起點或終點作為中心，向外以 l 為對角線的長度生成一個矩形。
2. 判斷是否有至少一個共乘路線上的路口I_i的（x, y）座標位於此矩形範圍內，且I_i與新乘客的起點或終點的油費都小於O_{TH}。

如圖 12，黑色路線為司機Di共乘前原本直接行駛到Di^-的路線，兩個藍色矩形範圍分別為P_2^+與P_2^-生成的矩形，只要判斷 I_1, I_2, I_3這三個位於P_2^+生成矩形內的路口到P_2^+的油費O_1, O_2, O_3，是否小於O_{TH}，即可以知道P_2^+是否可以繼續後續共乘的時間與油費比對，不需要判斷更多的路口，大幅減少計算量，而P_1^+與P_1^-生成矩形都沒有涵蓋到Di到Di^-的路口，因此P_1 直接刪除。

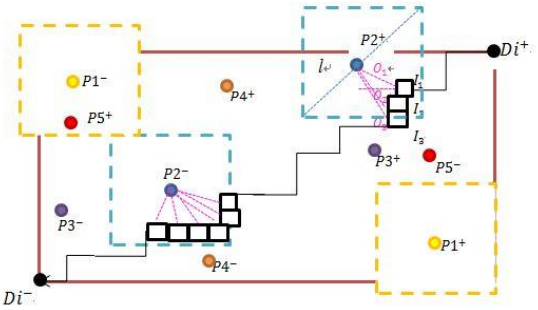


圖 12、判斷乘客是否能加入目前的共乘路線（$Di^+ \rightarrow Di^-$）的示意圖

Input：（事先去掉不符合規則 3.1 與 3.2 的司機與乘客）

司機端（出發位置（$D_i{}^+$）、終點位置（$D_i{}^-$）、出發時間（$DT_i{}^+$）、最晚到達時間（$DT_i{}^-$））、乘客端（出發位置（$P_i{}^+$）、終點位置（$P_i{}^-$）、出發時間（$PT_i{}^+$）、最晚到達時間（$PT_i{}^-$））、每個乘客與司機由 VBA*最低油費導航模式所得到的 DistCostArray[ ]、TimeCostArray[ ]、OilCostArray[ ]。

Output：

規畫出來的符合司機與乘客時間需求且司機分擔的油費最低的共乘乘載順序，並且列出共乘乘客的上下車時間、位置與分攤的油錢。

Step0：（初始化）

目前的共乘序列 SeqP[ ]={ $D_i{}^+$}；下一位新乘客節點的集合 NextP[ ]={所有初始範圍內的乘客}；所有可共乘的乘載順序 MatchP[ ]={ $D_i{}^+, D_i{}^-$}；目前的共乘乘客人數 P_c= Seq[ ]集合的乘客人數-1=0。

Step1：（判斷是否為順向）

運用此判斷可確保新加入的乘客與目前暫存的共乘序列的行駛方向相同，可確保乘客的起點在乘客終點前，避免走回頭路的情況發生。

Step2：（取軌跡的矩形範圍）

將 Step1 所得到 NextP[ ]中的每位乘客以該節點（起點／終點）為中心，向外以 l 為對角線的長度生成一個矩形範圍(如圖 12 中$P2^+$所取得的藍色矩形範圍)。

Step3：（Oi路口取法）

檢查每位新乘客的起點和終點的矩形範圍是否有覆蓋到每一個共乘的乘載順序 MatchP[ ]從 SeqP[ ]最後一個節點開始，經過所有確定的共乘節點到$D_i{}^-$的共乘路線。若起點和終點的矩形範圍都有覆蓋到，繼續 Step4。否則將這乘客從 NextP[ ]中刪除，不列為這次的共乘考量。

例如以圖 12 中$Di^+ \rightarrow Di^-$的行駛軌跡為例，

$P2^+$和$P2^-$這兩個節點的矩形範圍(圖 12 中藍色矩形範圍)都有涵蓋到$Di^+ \rightarrow Di^-$路線。反之，$P1^+$和$P1^-$這兩個節點的矩形範圍(圖 12 中黃色矩形範圍)都沒有涵蓋到$Di^+ \rightarrow Di^-$路線，所以將 P1 從 NextP[ ]中刪除，此時 NextP[ ]={ $P2, P3, P4$}。

Step4：

若【可共乘的下一位乘客 NextP[ ]＋目前共乘序列 Seq[ ]的人數）-1（扣除司機）】小於P_{MAX}且目前擴大次數E_c小於最大擴大次數E_{MAX}，則須結束這次共乘演算法，並執行「擴大區域演算法」，再重新呼叫回到共乘演算法，從 Step0 開始，在擴大後的範圍內，對於更多的乘客重新進行共乘配對，盡量找到P_{MAX}的共乘乘客數；反之，則繼續下一個步驟。

Step5：（產生所有可能的乘載順序）

將P_c加 1。將 Step4 所得到的乘客集合 NextP[ ]中的每一位乘客，加到從共乘序列 SeqP[ ]複製出來的每一個序列的最後，並產生所有可能的完整乘載順序，乘載順序中若有部分順序屬於 UnS[ ]，則此乘載順序將刪除，不用再進行後續判斷。

例如：由於$P1$ 以及$P5$在前面步驟已經刪除了，所以 NextP[ ]中剩下的乘客為$P2$ 、$P3$ 、$P4$。將$P2$ 、$P3$ 、$P4$分別加到共乘序列 SeqP[ ]={ $D_i{}^+$}複製出來的$D_i{}^+$之後，產生：

$$Di^+ \rightarrow P2^+ \rightarrow P2^- \rightarrow Di^-$$
$$Di^+ \rightarrow P3^+ \rightarrow P3^- \rightarrow Di^-$$
$$Di^+ \rightarrow P4^+ \rightarrow P4^- \rightarrow Di^-$$

Step6：（時間需求篩選）

將 Step5 所得出來的所有乘載順序執行司機與乘客的時間需求判斷，判斷在目前的共乘順序下，司機是否可以加入這位乘客共乘。

條件：

【此節點需求時間 ≥ 前一個節點的需求時間 ＋ 前一個節點到此節點的預計行駛時間 ＋ 門檻值 β】 (3)

節點的定義為D^+ 、D^- 、$P_i{}^+$ 、$P_i{}^-$。從 TimeCostArray[ ]中取出前後兩節點之間，依照 VBA*最低油費導航目標所求得的預計行駛時間，判斷司機是否可以在此乘客時間要求內趕去接送，門檻值β為一個從歷史軌跡分析得到的變異值，用來容忍共乘路線上塞車或者意外的一個時間差值，通常這個變異值會與距離成正比。必須同時滿足一個乘載順序前後兩節點間每一個時間需求，司機才可以去接送此乘客P_i，否則P_i不列為此次共乘乘載順序的考慮內，並從 NextP[ ]中刪除。不符合條件的順序要記錄在 UnS[ ]內，避免後面步驟重複計算，降低計算量。

$$DT^- \geq DT^+ + (D^+ \rightarrow P_i{}^+) + (P_i{}^+ \rightarrow P_i{}^-) + (P_i{}^- \rightarrow D^-) + \beta1+\beta2+\beta3 \quad (4)$$

此β1+β2+β3 為此次Di^+到Di^-共乘路線的門檻值。

Step7：（油費篩選）

將符合 Step6 時間需求篩選的所有乘載順序，從 OilCostArray[ ]中取出前後兩節點之間，依照 VBA*最低油費導航目標所求得的預計行駛油費，再依照本論文所提出的「油費分攤演算法」來分攤油費，記錄在 OilMoney[ ]中。若司機以及每位乘客共乘後的所分攤的油費都不高於自己直接行駛的油費，此乘載順序就符合共乘後降低分擔油費的目標，此時就將此乘載順序加入 MatchP[ ]中，並且把此P_i加入暫存序列 SeqP[ ]的最後。反之，不符合油費條件的順序也要記錄在 UnS[ ]中，避免後面步驟重複計算，降低計算量。

條件：

【司機與乘客共乘乘載順序所得的分攤油費 (OilMoney[ ])≤自己直接行駛的油費】 (5)

Step8：

假如$P_c < P_{MAX}$，表示共乘乘客數目尚未到達P_{MAX}，接著將 NextP[ ]設定為{所有範圍內的乘客-已經共乘的乘客}，重複 Step1～Step7，將共乘乘客數目 1~P_{MAX}所有可能的共乘順序都檢驗一次，找出所有符合條件的共乘承載序列 MatchP[ ]與油錢分擔的陣列 OilMoney[ ]。

Step9：

接著選出所有暫存的共乘乘載順序 MatchP[ ]中哪一個乘載順序的司機油費 OilMoney[ ]為最少，而最少的那一個乘載順序為最後確定的共乘乘載順序，存放在 CarP[ ]中。此時就得出所有共乘乘客列表、共乘的承載順序 CarP[ ]、各乘載點的時間與距離、司機與乘客分攤的油費 OilMoney[ ]等資訊。後續共乘配對伺服器可經由車載通訊網路通知司機與各共乘的乘客。

透過上述「初始軌跡區域的設定」、「起終點順向與否」大幅的縮小共乘乘客的搜尋範圍，再經由「時間和油費篩選」，將不符合條件的順序記錄在 UnS[ ]，將後續判斷更多乘客共乘時，刪去已經檢查過不可行的乘載順序，這樣可以大幅的減少共乘演算法所需要的計算。

3.2.3 油錢分攤演算法設計

針對共乘油費的分攤，我們設計出公平的機制：依據共乘路徑規劃演算法得出最適合的乘載順序，以每個路段的總油費除以共乘人數，計算出司機與每位共乘乘客在每個共乘的路段需要分攤的油費，最後統計出司機與每位共乘乘客的總油費。

Input：

司機與乘客的乘載順序 CarP[ ] = $[D^+, P_1{}^+, P_3{}^+, P_3{}^-, P_2{}^+, P_2{}^-, P_1{}^-, D^-]$;

Output：

司機與乘客個別分攤的油費金額 OilMoney[ ] 與每個路段的油費金額 RoadMoney[ ];

STEP 1：

從 CarP[0]取得司機D^+。RoadMoney[0]= OilMoney[0]= D^+; NP=1;

STEP 2：

取得 CarP[1]（載到第一個乘客$P_1{}^+$），將 RoadMoney[1]紀錄司機到$P_1{}^+$之前的總金額（由 OilCostArray[ ]取得 30 元），並將$P_1{}^+$加入集合 RoadMoney[2]。如表 2，此時記錄為 RoadMoney[$D^+, 30, P_1{}^+$]，OilMoney[$D^+, 30$]，

此路段司機需花費全部金額 OilMoney[1]=30。NP＋1=2。

STEP 3：

繼續取得 CarP[]的下一個乘客（$P_3{}^+$）之前（$P_1{}^+ \rightarrow P_3{}^+$）需花費的金錢（20 元），記錄 RoadMoney[$D^+, 30, P_1{}^+, 20$]。

STEP 4：

當司機到下一個乘載點（$P_3{}^+$）後，將 STEP 3 該路段司機與每個乘客須分攤金額為路段總金錢/ 這路段共乘的人數個數(NP)後，加入到集合 OilMoney[ ]中所有人需付的金額，進行更新 OilMoney[ ]。此時記錄為 RoadMoney[$D^+, 30, P_1{}^+, 20$]，OilMoney[$D^+, 40, P_1{}^+, 10$]。

STEP 5：

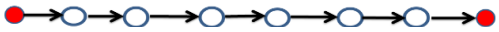
檢查下一個乘載點是否有人下車，若有，則 NP 減 1，該下車的人（P3）需付金額即為目前集合 OilMoney[ ]中對應乘客 ID 的金額。此時記錄為 RoadMoney[$D^+, 30, P_1{}^+, 20, P_3{}^+, 30$]，OilMoney[$D^+, 50, P_1{}^+, 20, P_3{}^+, 10$]。

STEP 6：

檢查司機下一個乘載點是否存在或者已抵達共乘行程的終點。若未到達終點則重複 STEP 4~STEP 5。若已抵達共乘行程的終點（D^-），則最終油錢為最後記錄的 OilMoney[$D^+, 130, P_1{}^+, 60, P_2{}^+, 10, P_3{}^+, 10$]，RoadMoney[$D^+, 30, P_1{}^+, 20, P_3{}^+, 30, P_3{}^-, 40, P_2{}^+, 30, P_2{}^-, 20, D^-, 40$]。

表 2、司機及各乘客在每個路段所需支付油錢表

D^+ $P1^+$ $P3^+$ $P3^-$ $P2^+$ $P2^-$ $P1^-$ D^-



RoadMoney[ ] 路段/金額 司機/乘客	30	20	30	40	30	20	40	OilMoney[ ] 總計
D	30	10	10	20	10	10	40	130
P1		10	10	20	10	10		60
P2					10			10
P3			10					10

4. 模擬與實驗

4.1 實驗比較演算法

在實驗效能部分，本論文分析傳統共乘論文與網站系統，整理並歸納出這些方法的特性與應用於動態共乘的合適性，做為之後模擬實驗時效能比較的依據。我們將最佳共乘配對法(OPT)、TSP、本論文提出的 CSVI 法，與兩種路徑規劃演算法(Dijkstra 與 VBA*)組合，比較三種方法（OPT＋Dij, TSP+Dij 與 CSVI+VBA*）的效能。

4.1.1 路徑規劃法實驗設定

1. Dijkstra 路徑規劃法模擬環境為根據交通部運輸研究所[2]提供的台北市交通路網數值地圖測試檔案，參考其歷史資料，作為本實驗的地圖資料庫來源，每條路段車速將會抓取地圖資料庫所提供的道路速限做為路況車速值，同時也做為模擬實驗速度值，接著再將其車速值和距離推估為油耗值，最後再以最低油耗為每段路的成本值，規劃出最短的行駛路徑。
2. VBA*路徑規劃法以 Java 程式語言和 Eclipse 為模擬工具，模擬環境為根據交通部運輸研究所[2]提供的台北市交通路網數值地圖測試檔案，參考其歷史資料，作為本實驗的地圖資料庫來源，在建立的車輛網路環境下進行共乘演算法測試，來驗證本架構的實用性，圖 13 為本實驗地圖範圍。每條路段車速將會抓取歷史路況資訊，每十五分鐘將路段道路狀態顏色依照道路型態，轉換為相對應速度值，若該道路無即時車況資訊，則以速限作為其速度值，做為模擬實驗速度值，而得到了該路段的車速值和距離，進而推估出行駛的時間。

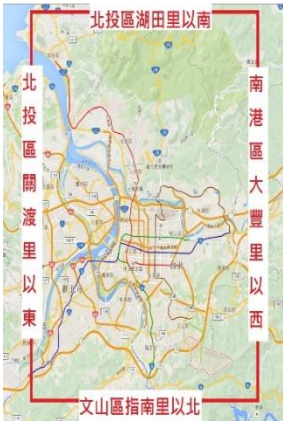


圖 13、實驗地圖

4.1.2 共乘演算法設定

預期要比較之三種共乘路徑演算法如下：

1. OPT+Dijkstra：使用最短路徑演算法(Dijkstra)，以每條路段車速將會抓取地圖資料庫所提供的道路速限做為路況車速值，估計出每個路段的預期油費與兩點之間最低油費的行車路線，接著再運用所得到的資訊結合最佳法(OPT)，全面搜尋整個地圖，找出所有配對結果最佳的共乘油耗以及共乘配對組合。
2. TSP+Dijkstra：使用最短路徑演算法(Dijkstra)以每條路段車速將會抓取地圖資料庫所提供的道路速限做為路況車速值，估計出每個路段的預期油費與兩點之間最低油費的行車路線，接著再運用所得到的資訊結合 TSP。TSP 提出整張地圖的九宮格概念，運用乘客終點尋找附近九宮格司機終點位置的概念來做共乘篩選與配對，進而計算出共乘油耗以及共乘配對組合。
3. CSVI+VBA*：本論文的方法是使用 VBA*演算法，每條路段車速將會抓取歷史路況資訊，每十五分鐘將路段道路狀態顏色依照道路型態，轉換為相對應速度值，估計出每個路段的預期油費與兩點之間最低油費的行車路線，接著再運用所得到的資訊結合本共乘系統中「共乘系統架構」中的「共乘配對演算法」、「油錢分攤演算法」與「配對範圍初始與擴大演算法」等演算法與「起終點順向與否與覆蓋路口油費的檢測」、「軌跡範圍搜尋乘客」，縮小共乘乘客的搜尋範圍，進而計算出共乘油耗以及共乘配對組合。

這三種方法的比較主要是想呈現出使用 VBA*演算法代替最短路徑演算法(Dijkstra)在缺乏即時路況之下計算路徑造成實際行駛時間的差距。由數據圖觀察出儘管最佳法(OPT)用最短路徑演算法(Dijkstra)的方式可以找到較多的共乘配對組合，但在共乘油耗和演算法執行時間上卻更多；相較之下 CSVI 方法用 VBA*演算法的方式雖然少了幾組共乘配對組合，但是共乘油耗和時間卻和最佳法差距不大。而在時間計算的比較，也顯示出本論文提出的 CSVI 方法大幅的減少計算複雜度，縮短了演算法計算的時間；與第二類 TSP[9]的方式比較欲呈現出本共乘系統中「共乘系統架構」中的「共乘配對演算法」、「油錢分攤演算法」與「配對範圍初始與擴大演算法」等演算法優於 TSP[9]的共乘方法，在同樣使用司機／乘客人數、位置、時間需求的狀況下，CSVI 的方法在以下的效能項目呈現出更好的結果。

4.2 預期比較之路由協定效能項目

常見用來比較的共乘效能項目(metric)如下：（固定的司機/乘客人數、位置、時間需求）

1. 共乘成功配對的司機百分比：共乘成功配對的司機總數／所有司機總數（共乘＋未共乘）。檢驗司機的共乘成功率是否有隨著乘客數或者司機數成長。
2. 共乘成功配對的乘客百分比：共乘成功配對的乘客總數／所有乘客總數（共乘＋未共乘）。檢驗乘客的共乘成功率是否有隨著乘客數或者司機數成長。
3. 平均每台車輛的乘客數目：共乘成功配對的乘客總數／所有司機總數（共乘＋未共乘）。檢驗每台車輛的共乘乘客數目是否有隨著乘客數或者司機數成長。
4. 所有共乘路線司機分擔的節省油費平均值：共乘成功配對的司機節省總油費／所有司機總數（共乘＋未共乘）。所有司機共乘後分擔油費的平均值越低，節省油費越高，表示共乘演算法確實能找到司機最低油費的路線。
5. 所有共乘路線乘客分擔的節省油費平均值：共乘成功配對的乘客節省總油費／所有乘客總數（共乘＋未共乘）。所有乘客共乘後分擔油費的平均值越低，節省油費越高，表示共乘演算法確實能找到乘客省油費的路線。
6. 演算法執行時間：檢驗所設計的 CSVI 計算時間是否真的有比 OPT 大幅減少。

4.3 實驗步驟與設定

實驗步驟：

步驟 1. 圖 13 為本實驗地圖範圍，以南港區大豐里、北投區關渡里、北投區湖田里及文山區指南里所圍成區域，模擬範圍大小約為 58.8 km x 15.9km。

步驟 2. 根據台北市交通歷史軌跡資訊[2]，將時間分為壅塞時段及普通時段，如表 33，壅塞時段包含 07:00 至 09:00、12:00 至 13:00、

17:00 至 19:00 及 22:00 至 23:00，其餘時間皆為普通時段，執行 VBA*求出最省油費的路線，本實驗假設時段為壅塞時段 07:00 至 09:00。

步驟 3. 在模擬中，選擇較符合 CSVI 共乘系統的參數，包括：「配對範圍初始與擴大演算法」中延伸司機起終點連線的 r 值與最大擴大範圍的次數(E_{MAX})、「共乘配對演算法」中司機的最大乘載人數(P_{MAX})等；而 TSP 的參數為整個地圖的九宮格 R×R 大小(B_r)。

步驟 4. 接著透過車輛網路傳送記錄的路況資料給共乘配對伺服器（由一部桌上型電腦執行），進行 Dijkstra 或 VBA*演算法估計出每個路段的預期油費與兩點之間最低油費的行車路線，隨後執行共乘系統的訊息流程，設定共乘需求條件，發送與接收共乘訊息，顯示最後的共乘順序、路線、時間與分擔油費。當共乘開始後，車輛行駛於道路時，若發生塞車或意外狀況，VBA*會使用車輛網路 V2V 方式，直接在司機與共乘乘客之間傳遞通知訊息，進行必要的共乘資訊的改變。

步驟 5. 分別以不同預設參數等進行 10 次的模擬，畫出多種數據圖，觀察不同參數對效能的影響，驗證共乘演算法的有效性。

表 3、時間狀態區間表

開始時間	結束時間	狀態
23：00	07：00	普通
07：00	09：00	擁塞
09：00	12：00	普通
12：00	13：00	擁塞
13：00	17：00	普通
17：00	19：00	擁塞
19：00	22：00	普通
22：00	23：00	擁塞

表 4、模擬環境實驗預設參數設定

項目	設定值
模擬範圍	58.8 km x 15.9km
實驗時段	07:00～09:00
司機數	10,20,30,40,50（預設值 30）
乘客數	100,200,300,400,500（預設值 300）
最大乘載人數(p_{MAX})	0,1,2,3,4（預設值 2）
最大擴大次數(E_{MAX})	0,1,2,3,4（預設值 2）
搜尋乘客的擴大範圍大小(Er)	2,3,4,5,6（預設值 4）
搜尋乘客的軌跡範圍大小(Rr)	0,2,4,6,8（預設值 4）
油價(1 公升多少台幣油費)	23.5
整張地圖的九宮格大小(B_r)	2×2,6×6,10×10,14×14,18×18（預設值 10×10）

各項參數的預設值設定如表 4 所示。各效能項目相對應的參數為「最大乘載人數(p_{MAX})」、「最大擴大次數(E_{MAX})」、「搜尋乘客的擴大範圍大小(Er)」、「搜尋乘客的軌跡範圍大小(Rr)」等。以上這些參數將與演算法的時間複雜度有密切關係，也會影響配對成功率的結果，而其中配對結果又影響了共乘省了多少油費。

4.4 根據台北市地圖模擬數據圖

4.4.1 司機數對於各種 metric 之影響

圖 14、圖 15 顯示，由於 OPT 法是整個地圖去做搜尋，而 CSVI 則是縮小範圍搜尋共乘的配對組合，所以 OPT 比 CSVI 能夠搜尋到更多的共乘組合狀況，也能夠找更多的共乘配對最佳組合。而 OPT 和 CSVI 皆優於 TSP 是因為 TSP 是用整個地圖的九宮格概念，依據乘客的終點位置去搜尋附近九宮格內的司機終點位置，一次性的大幅篩掉乘客來做時間和油費的進一步篩選，錯失掉更多能夠共乘的配對組合，所以配對成功的數目就會較少。而以上三種方法隨著司機數越來越多，以固定的乘客數去做配對，越來越多司機可以去接送乘客，但由於司機數是以倍數再增加，而成功共乘配對的司機數並非以倍數再增加，所以 OPT 和 CSVI 的配對成功司機百分比才會緩慢下降；此外，越來越多司機可以去接送乘客，相對的所配對乘客成功率也會逐漸提升。圖 16 我們可以觀察出，OPT 比 CSVI 能夠搜尋到更多的共乘組合狀況，也能夠找更多更省油的共乘配對最佳組合。而圖 17，共乘成功的乘客越來越多，整體的乘客平均省油費也會逐漸提升

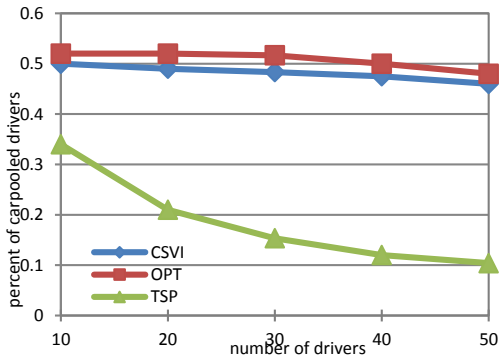


圖 14、台北市地圖-司機數對配對成功司機百分比之影響

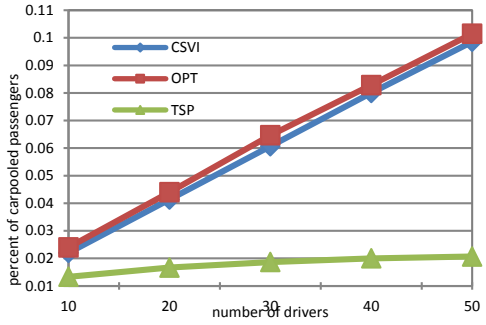


圖 15、台北市地圖-司機數對配對成功乘客百分比之影響

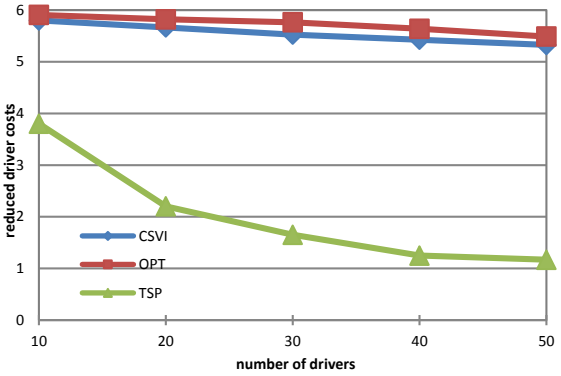


圖 16、台北市地圖-司機數對司機平均節省油費之影響

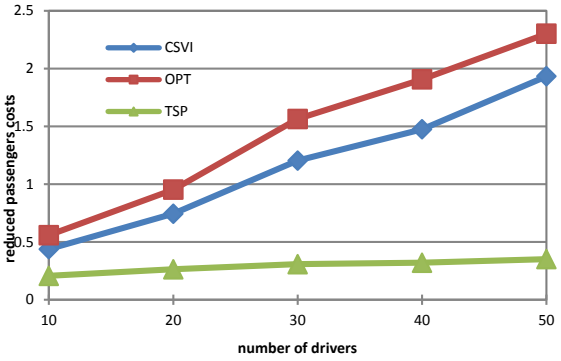


圖 17、台北市地圖-司機數對乘客平均省油費之影響

圖 18 顯示，以上三種方法隨著司機數越來越多，以固定的乘客數去做配對，越來越多司機可以接送乘客，而每台車平均下來的共乘乘客數相對的就會逐漸下降，其中 TSP 因圖 15 數據圖顯示乘客的配對數隨著司機數增加，而明顯上升，所以這樣平均下來，TSP 每台車平均共乘人數的數據曲線圖就會明顯的下降，而 OPT 與 CSVI 法因圖 15 數據圖顯示乘客的配對數隨著司機數增加，而明顯上升，所以每台車平均共乘人數的數據曲線圖則逐漸緩慢下降，表示這兩種方法能達到穩定的配對。

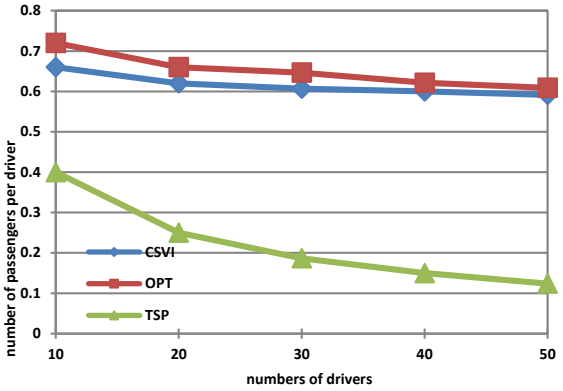


圖 18、台北市地圖-司機數對每台車平均乘客人數之影響

圖 19 顯示，OPT 是整個地圖都去搜尋，並且列出所有的共乘組合，接著再用時間和油費來篩選出最佳的共乘配對組合。而 1 位共乘乘客的所有排列順序$(D^+ \rightarrow Pi^+ \rightarrow Pi^- \rightarrow D^-)$有$\binom{n}{1}$種，而$P_{MAX}$位共乘乘客的所有排列順序$(D^+ \rightarrow (P_1^+ \rightarrow P_2^+ \ldots \rightarrow P_{P_{MAX}}^+) \rightarrow (P_i^- \rightarrow P_j^- \ldots \rightarrow P_k^-) \rightarrow D^-)$有$\binom{n}{P_{MAX}}$種共乘乘客選法與各$(P_{MAX}!)$種上車與下車順序，共$\binom{n}{P_{MAX}} \times (P_{MAX}!)^2$種排列順序。因此全部要比對的共乘順序有$m \times \sum_{i=1}^{P_{MAX}} \binom{n}{i} \times (i!)^2$種，m 為司機數，n 為乘客數，當 m,n 大時，這是相當大的次數，所以 OPT 的演算法計算時間很明顯的比 CSVI 和 TSP 還要長許多。CSVI 提出了進一步的縮小比對範圍的方法，減少需要比對能否共乘的乘客數目 n，其主要概念是要避免比對「偏離目前的共乘路線太遠的乘客」，避免司機繞路太遠去接送乘客，不僅可以降低分擔的油費，也可以降低共乘演算法的計算量。TSP 之所以演算法計算時間會比 CSVI 還要少是因為 TSP 是依據乘客的終點位置去搜尋附近九宮格內的司機終點位置，一次性的大幅篩掉乘客來做時間和油費的進一步篩選，錯失掉更多能夠共乘的配對組合，當然演算法計算時間也比 CSVI 要少一些。

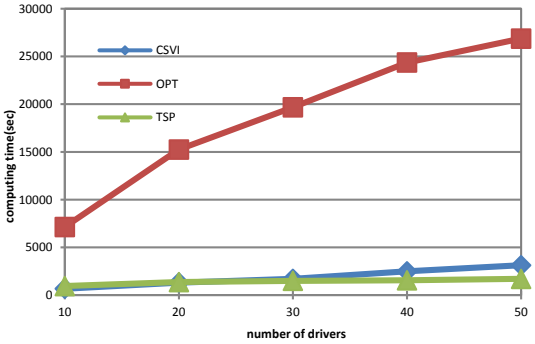


圖 19、台北市地圖-司機數對演算法計算時間之影響

4.4.2 乘客數對於各種 metric 之影響

圖 20、圖 21、圖 22、圖 23 顯示，由於 OPT 是整個地圖去做搜尋，不會錯失任何的共乘配對組合，可以找到最佳共乘解，而 CSVI 則是縮小範圍搜尋共乘的配對組合，所以相較於 OPT，CSVI 能找到的共乘配對組合就會較少。OPT 和 CSVI 皆優於 TSP 是因為 TSP 是一次性的大幅篩掉乘客來做篩選，容易錯失掉更多能夠共乘的配對組合，所以配對成功的數目就會較少。而以上三種方法隨著乘客數越來越多，以固定的司機數做配對，越來越多乘客提供司機做選擇，相對的所配對到的司機成功百分比也會逐漸提升，雖然越來越多乘客配對成功，但由於配對成功的上升幅度較緩慢，相較於倍數成長的乘客數，乘客成功的百分比就會逐漸下降。而隨著司機數與乘客數的成功百分比影響，同時也會影響到省油費的部份。圖 22 顯示，由於司機成功百分比曲線是呈現上升的狀態，而乘客成功的百分比是下降的狀態，因此平均下來每台車的共乘乘客人數是呈現上升的狀態。

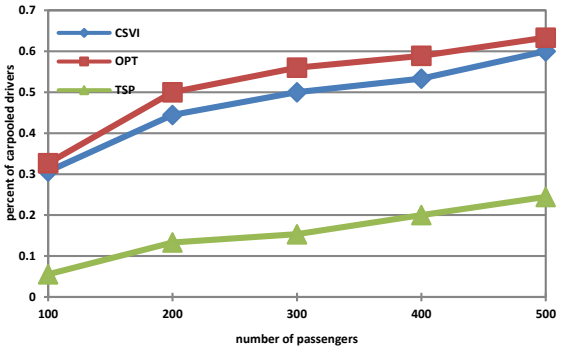


圖 20、台北市地圖-乘客數對配對成功司機百分比之影響

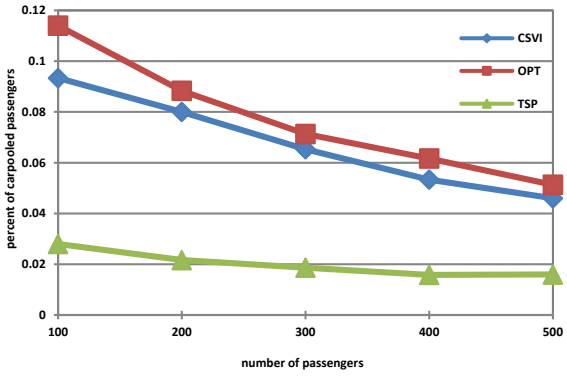


圖 21、台北市地圖-乘客數對配對成功乘客百分比之影響

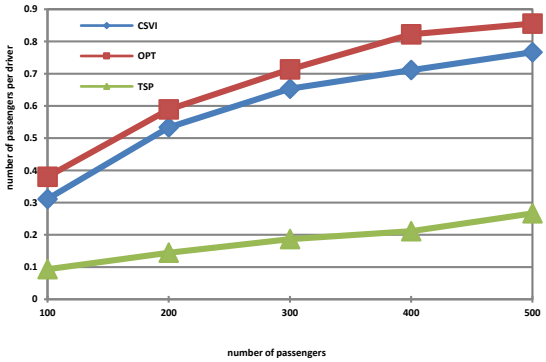


圖 22、台北市地圖-乘客數對每台車平均共乘人數之影響

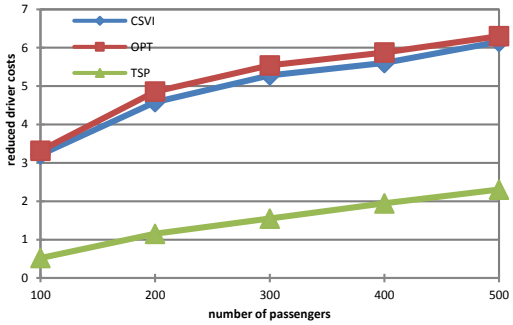


圖 23、台北市地圖-乘客數對司機平均節省油費之影響

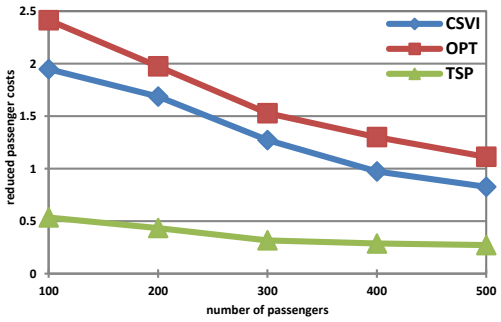


圖 24、台北市地圖-乘客數對乘客平均省油費之影響

圖 25 顯示，OPT 是整個地圖都去搜尋，並且列出所有的共乘組合接著再用時間和油費來篩選出最佳的共乘配對組合，所以 OPT 的演算法計算時間很明顯的比 CSVI 和 TSP 還要長許多。CSVI 提出了進一步的縮小比對範圍的方法，減少需要比對能否共乘的乘客數目 n，可以降低共乘演算法的計算量。TSP 之所以演算法計算較少是因為 TSP 是用一次性的大幅篩掉乘客來做時間和油費的篩選，錯失掉更多能夠共乘的配對組合，當然演算法計算時間也比 CSVI 要少一些。

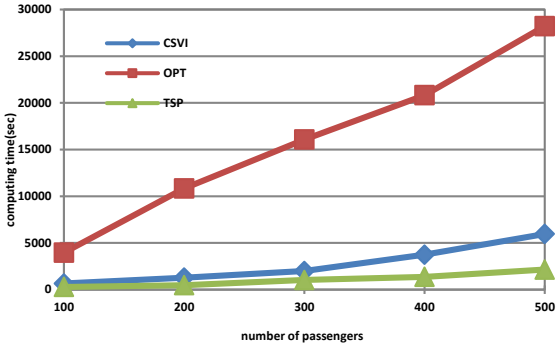


圖 25、台北市地圖-乘客數對演算法計算時間之影響

4.4.3 最大乘載人數對於各種 metric 之影響

從圖 26 顯示，由於 OPT 是整個地圖去做搜尋，而 CSVI 則是提出「共乘乘載範圍演算法」以及「軌跡範圍搜尋乘客」來縮小範圍搜尋共乘的配對組合，所以 OPT 比 CSVI 能夠搜尋到更多的共乘組合狀況，也能夠找更多的共乘配對最佳組合。而 OPT 和 CSVI 皆優於 TSP 是因為 TSP 是用整個地圖的九宮格概念，依據乘客的終點位置去搜尋附近九宮格內的司機終點位置，一次性的大幅篩掉乘客來做時間和油費的進一步篩選，錯失掉更多能夠共乘的配對組合，所以配對成功的數目就會較少。而在最大乘載人數為 0 的時候，其代表意義也是相當於未提供共乘的狀況，無論是司機或者乘客的成功百分比當然就皆為 0，而以上三種方法在最大乘載人數為 1～4 人的時候會呈現一個持平的狀態，是因為最大乘載人數為 1 的時候，若該位司機因為時間和油費檢驗都無法有配對的乘客共乘組合，就算放寬條件最大乘載人數為 2～4 人時，也會再因為時間和油費的問題而被篩選掉，導致相同的配對失敗結果；若該位司機能夠接送一位乘客的時候，在放寬條件最大乘載人數為 2～4 人的時候，就是相同配對成功的司機去做更多的共乘組合配對，因此成功共乘的司機百分比固定不變。

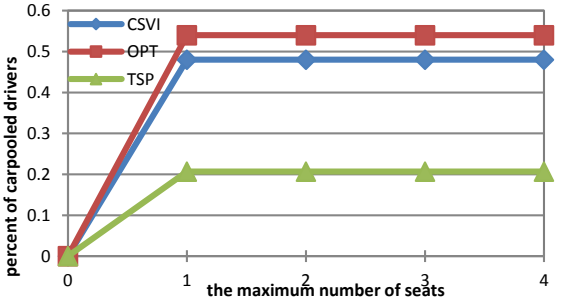


圖 26、台北市地圖-最大乘載人數對配對成功司機百分比之影響

圖 27 顯示，在最大乘載人數為 0 的時候，其代表意義也是相當於未提供共乘的狀況，乘客的成功數當然就皆為 0，而以上三種方法隨著每位司機的最大乘載人數越來越多人，以固定的司機與乘客數去做配對，越來越多座位提供乘客做共乘配對，相對的所配對到的乘客成功百分比也會逐漸提升。

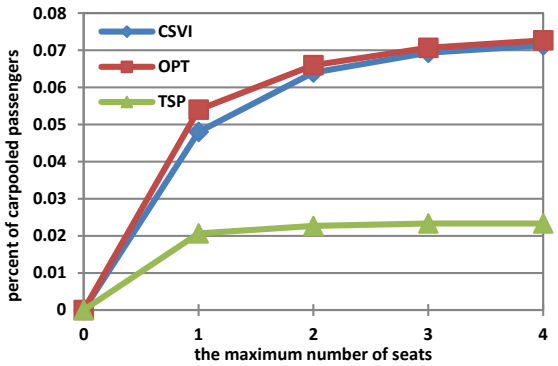


圖 27、台北市地圖-最大乘載人數對配對成功乘客百分比之影響

圖 28 顯示，在最大乘載人數為 0 的時候，其代表意義也是相當於未提供共乘的狀況，而每台車平均下來的乘客數當然就是為 0，而以上三種方法隨著每位司機的最大乘載人數越來越多人，以固定的司機與乘客數去做配對，越來越多座位提供乘客做共乘配對，所以平均每台車的共乘人數也會逐漸上升。

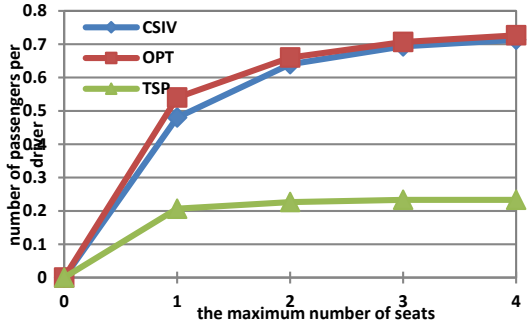


圖 28、台北市地圖-最大乘載人數對每台車共乘人數之影響

圖 29 以上三個方法隨著每台車的座位數越來越多，越來越多乘客可以分攤司機的油費，司機所省下的油費相對的就會越來越多。

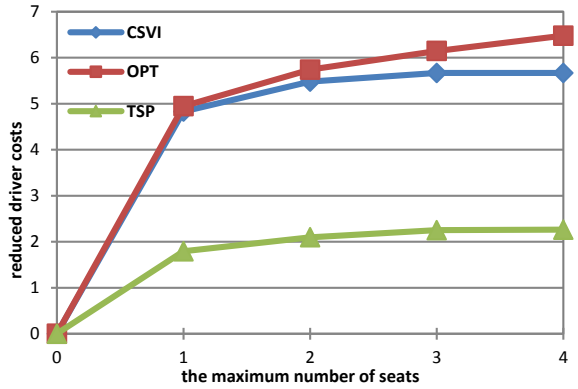


圖 29、台北市地圖-最大乘載人數對司機平均省油費之影響

圖 30 顯示，以上三個方法隨著司機的座位數越來越多，共乘配對成功的乘客越來越多，整體的乘客平均省油費也會逐漸提升。

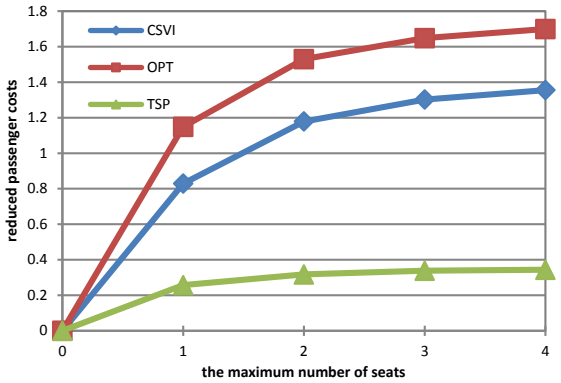


圖 30、台北市地圖- 最大乘載人數乘客平均省油費之影響

圖 31 顯示，由於 OPT 是整個地圖的乘客都做搜尋並且找出最佳的共乘配對組合，所以 OPT 的演算法計算時間很明顯的比 CSVI 和 TSP 還要長許多。CSVI 提出了進一步的縮小比對範圍的方法，不僅可以降低分擔的油費，也可以降低共乘演算法的計算量。TSP 之所以演算法計算時間會較少是因為 TSP 是用整個地圖的九宮格概念，一次性的大幅篩掉乘客來做時間和油費的篩選，錯失掉更多能夠共乘的配對組合，當然演算法計算時間也比 CSVI 還要少。而以上三種方法隨著每台車可以共乘的人數(P_{MAX})越來越多，所需要列出的共乘組合也相對的越多，所以時間才會越來越長。

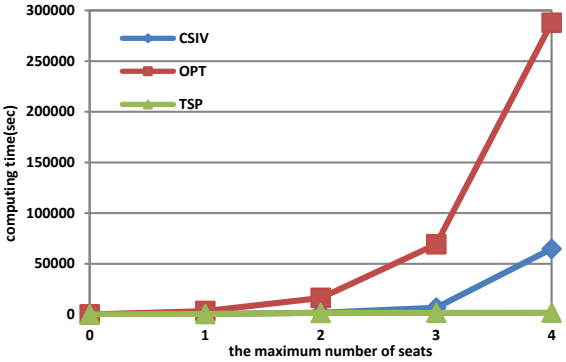


圖 31、台北市地圖- 最大乘載人數演算法計算時間之影響

4.4 實驗結論

總結以上各項參數的實驗模擬，可以觀察出以下幾點：

1. 最短路徑演算法(Dijkstra)在缺乏即時路況之下計算出的路徑資訊造成實際行駛時間的差距，尤其是當有塞車狀況發生的時候，會進而影響行駛的時間和油費，由數據圖觀察出儘管 OPT 是整個地圖搜尋共乘配對的乘客，可以找到較多的共乘配對組合，但在計算時間上卻花費較多，相較之下 CSVI 減少檢驗許多共乘配對組合，節省大量計算時間。
2. OPT 是整個地圖搜尋共乘配對的乘客，雖然可以找到較多甚至更佳的共乘配對組合，但在演算法計算時間上卻花費較多，相較之下 CSVI 提出「配對範圍初始與擴大演算法」和「軌跡範圍搜尋乘客」方式，大量減少比對的配對組合，雖然少了幾組共乘配對的結果，而其中這個減少的共乘配對組合數量只有個位數的差距，但演算法計算時間卻是大幅縮短，平均相差 4～5 倍的時間差距，尤其當每台車最大的乘載人數越來越多的時候，差距更為顯著。
3. TSP 提出的九宮格概念，用乘客終點尋找九宮格附近的司機終點來篩選配對，其目的就是想要縮小範圍搜尋可以配對的乘客，但只有依據乘客和司機的終點來做篩選，並沒有考慮到乘客和司機的起點相距有多遠，而這樣的情況下配對成功的數量卻少之又少，同時也錯失了很多共乘配對的最佳組合，不僅失敗率高，平均省的油費也少。而本篇論文提出的 CSVI 設計了「配對範圍初始與擴大演算法」以及「軌跡範圍搜尋乘客」有效的縮小了搜尋共乘乘客的範圍，且乘客和司點的起點和終點都有考慮到，相較之下比較不會錯失過多能夠配對的組合，在各種的數據圖上都顯示出本篇論文提出的 CSVI 除了計算時間稍微多一些外，其他的效能項目都優於論文 (TSP)的共乘方法。

5. 結論

本論文提出 CSVI 共乘系統，我們提出了「共乘配對演算法」並且結合 VBA*演算法，利用歷史路況資料和 Google Map 即時路況資訊估計出每個路段的預期油費與兩點之間最低油費的行車路線，讓它除了能預測當時路況以外，更可進一步運用在共乘系統上。過去使用最短路徑演算法(Dijkstra)的方法中，不考慮路段是否塞車，而我們採用 VBA*演算法來預測道路的路況資訊並且規劃出最佳的行駛路徑，使用「配對範圍初始與擴大演算法」、「軌跡範圍搜尋乘客」和「起終點順向與否與覆蓋路口的檢測」大幅且有依據的縮小搜尋配對的乘客範圍，以降低計算的時間複雜度，並利用「共乘配對演算法」中的時間篩選和油費篩選，選出符合時間需求內且與未共乘的情況相比，必為更加省油費的共乘組合。並提出「油錢分攤演算法」公平的付費機制，無論是對於司機或者乘客都必為更加省油費。

在實驗模擬中，考慮許多參數包括最大乘載人數(p_{MAX})、最大擴大次數(E_{MAX})、搜尋乘客的擴大範圍大小(Er)…等，由數據可以發現其效能皆趨近於最佳解 OPT，且優於論文(TSP)同類型的共乘系統，對於演算法計算時間也大幅縮短，其中與 OPT 方法比較更為顯著。

參考文獻

1. 共乘王：http://www.carpoolking.com/tw/zh-hk/.
2. 交通部運輸研究所：http://e-iot.iot.gov.tw/
3. BlueNet ： https://play.google.com/store/apps/details?id=com.BlueNetPlay.CarpoolActivity
4. George Dimitra kopoulos, Panagiotis Demestichas,Vera Koutra, "Intelligent Management Functionality for Improving Transportation Efficiency by Means of the Car Pooling Concept," *IEEE Transactions on Intelligent Transportation Systems",* Vol. 13, No. 2, pp. 424-436, June 2012.
5. Ing-Chau Chang, Hung-Ta Tai, Feng-Han Yeh, Dung-Lin Hsieh, and Siao-Hui Chang, "*A VANET-Based A* Route Planning Algorithm for Travelling Time-and Energy-Efficient GPS Navigation App*," International Journal of Distributed Sensor Networks, Hindawi, June, 2013.
6. Jinqi Zhu, Yong Feng, Bang Liu, "PASS: Parking-Lot-Assisted Carpool over Vehicular Ad Hoc Networks," *International Journal of Distributed Sensor Networks",* Hindawi, December 2012.
7. Manel Sghaier, Hayfa Zgaya, Slim Hammadi, Christian Tahon, "*A Novel Approach Based on A Distributed Dynamic Graph Modeling Set Up Over A Subdivision Process to Deal With Distributed Optimized Real Time Carpooling Requests*," *The 14th International IEEE Conference on Intelligent Transportation Systems*, Washington, DC, USA, pp.1311-1316, October 5-7, 2011.
8. Nicola Bicocchi, Marco Mamei, "*Investigating Ride Sharing Opportunities through Mobility Data Analysis*," *Pervasive and Mobile Computing , Vol. 14, pp.83-94,* October 2014.
9. Po-Yu Chen, Je-Wei Liu, and Wen-Tsuen Chen, "*A Fuel-Saving and Pollution-Reducing Dynamic Taxi-Sharing Protocol in VANETs*," *IEEE Vehicular Technology Conference Fall,* Sept. 2010.
10. Yong Ge, HuiXiong, Alexander Tuzhilin, KeliXiao, Marco Gruteser, Michael J. Pazzani, "*An Energy-Efficient Mobile Recommender System*," *International conference of ACM SIGKDD,* pp.899-908, 2010.