私立東海大學資訊工程與科學研究所

碩士論文

指導教授：林祝興 博士

Dr. Chu-Hsing Lin

應用基因演算法於智慧型行車資訊系統尋找最短時間路徑

A Genetic Algorithm for Finding Routes with Shortest Driving Time in Intelligent Transportation Systems

研究生：李嘉仁

(Chia-Ren Lee)

中　華　民　國　九　十　七　年　六　月

東海大學碩士學位論文考試審定書

東海大學資訊工程與科學系 研究所

研究生 李 嘉 仁 所提之論文

應用基因演算法於智慧型行車資訊系統尋找最短時間路徑

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 _____ 簽章

委 員 _____

_____

_____

_____

指 導 教 授 _____ 簽章

中華民國 97 年 6 月 30 日

摘 要

行車導航是一項很熱門的需求應用，靠著手持裝置與GPS的普及性，人人都可以進行導航操作。而導航的準確性與快速性一直是大家要求的，在地圖資料改良與實際經驗下，這二項要求也漸漸達到完善的目標。為了提供更多的資訊，行車導航也開始需要考慮即時車流量與車子的行駛速率等，因系統的變因增加，計算所需的資源也就需要越多，而基因演算法可以在不需要耗用太多資源下解決這類的問題。

基因演算法由演化法則的概念而來，藉由可以表現系統狀態的染色體(chromosome)對問題進行編碼，再透過交配(crossover)、突變(mutation)這些基因演算法內的操作方法，使最佳解逐漸逼進出來。基因演算法解決最短路徑問題的效率不錯，且節點數很多的情況下仍然收斂很快。

一般針對行車導航的問題只是尋找出最短的行駛路徑而已，沒有考量到在不同路況車子也是會有不同的行駛速率，本文在此對這類的問題加以敘述，並說明使用基因演算法的解決辦法，而這類型的問題變化種類還有很多，多半只需要重新調整基因演算法的一些參數，就可以一樣簡便的尋求其解。

關鍵詞：基因演算法、智慧型行車資訊系統、最佳路徑、嵌入式系統、最短時間路徑

Abstract

The route guidance system, which provides driving advice based on traffic information about an origin and a destination, has become very popular along with the advancement of handheld devices and the global position system. Since the accuracy and efficiency of route guidance depend on the accuracy of the traffic conditions, the route guidance system needs to include more variables in calculation, such as real time traffic flows and allowable vehicle speeds. As variables considered by the route guidance system increase, the cost to compute multiplies. As handheld devices have limited resources, it is not feasible to use them to compute the exact optimal solutions by some well-known algorithm, such as the Dijkstra's algorithm, which is usually used to find the shortest path with a map of reasonable numbers of vertices.

To solve this problem, we propose to use the genetic algorithm to alleviate the rising computational cost. We use the genetic algorithm to find the shortest time in driving with diverse scenarios of real traffic conditions and varying vehicle speeds. The effectiveness of the genetic algorithm is clearly demonstrated when applied on a real map of modern city with very large vertex numbers

Keywords: Genetic Algorithm, Intelligent Transportation System, Optimal Route, Embedded System, Shortest Time Path

致 謝

本論文的完成，首先要感謝林祝興老師的指導，在這三年的研究下來，經過多次的報告、會議、計畫，祝興老師指導我研究方向不遺餘力，透過長期的討論與修正，最後再用實驗的結果逐步完成論文，非常感謝老師。劉榮春老師在實驗室幫助修改大家的論文並與大家一起討論，獲益匪淺，在此一並感謝。

在實驗室參與了不少計畫，了解一個計畫的完成要協力合作，與佳男、懋樺合作使我得以專心的研究技術並一起討論各種做法，雖然過程中有時會爭議或激動之處，這都是難能可貴的經驗。

感謝麗靜學姐指導各種文件的製作技巧與投影片的風格，也感謝仁傑、彥菱同學的幫忙，實驗室的衍緯、嘉瀚、掄元、育瑩、建廷學弟、美君學妹們當然也不能忘記，大家一起快樂的活動是我永遠的回憶。

三年的研究生涯讓我成長不少，從一開始不善長報告到後來可以準備一小時的報告，還有研究計畫行車資訊系統與嵌入式防火牆這都是讓我了解架構系統的方式，而這些成果都可以在網路上找到相關的資料，歡迎一起討論。

目 錄

圖表列表 6

第壹章 導論 7

第貳章 背景介紹 8

第一節 最短路徑問題 8

第二節 基因演算法 9

第三節 智慧型行車資訊系統 12

第 1 節 全球定位系統 12

第 2 節 嵌入式系統平台 14

第 3 節 ITS系統平台 17

第參章 最短行車時間問題及其解法 18

第一節 最短行車時間問題簡介 18

第二節 最短行車時間問題之參數設定 19

第三節 最短行車時間問題之基因演算法解法 20

第肆章 實驗與結果 21

第一節 使用固定速率資訊以基因演算法尋找路徑 21

第 1 節 基因演算法單次導航 21

第 2 節 基因演算法多次導航 25

第二節 使用變動速率資訊以基因演算法尋找路徑 28

第伍章 結論與未來工作 32

參考文獻 33

圖表列表

圖 2-1 基因演算法流程圖.....10
圖 2-2 智慧型行車資訊系統.....12
圖 2-3 DMA 2410.....14
圖 2-4 Qtopia 平台架構.....15
圖 2-5 軟體架構圖.....16
圖 2-6 行車系統執行圖.....17
圖 4-1 4x4 square matrix map.....21
圖 4-2 演算代數收斂圖.....27
圖 4-3 16x16 中有留存上一代的染色體的收斂圖比較.....31

表 2-1 GPS 資料標頭.....13
表 2-2 GPGGA 格式.....13
表 4-1 平均尋找最佳路徑所需要的演化代數.....22
表 4-2 平均與使用 DA 演算法所求得的最佳解的誤差比.....23
表 4-3 使用基因演算法於 Real Map 的結果.....24
表 4-4 多次導航的效果比較.....25
表 4-5 應用 ITS 的即時更新資訊與原始最短時間路徑的比較 1.....29
表 4-6 應用 ITS 的即時更新資訊與原始最短時間路徑的比較 2.....30

第壹章 導論

現今社會中使用 PDA 做為行車導航工具已經是很普遍的事情了[1-2]，而不管是專用的導航機器或是自行安裝的導航程式皆可達到解決最短路徑問題，使自己行走的道路為最短，進而節省行車時間，以換取更多的利益。

一般的行車導航軟體多半只考慮起點與終點之間的相對距離，進而推算出固定解答的路徑，從未考慮到行車的路況以及時間的變化，所以導航出來的路徑只是道路距離最短而已，而不一定是最佳的行車路徑。如果要考慮這些變數進來，就會增加行動裝置的運算負擔，一般而言行動裝置的運算能量較小，解決這類的運算問題所耗的時間就較多。而變通的方法之一是將問題傳回伺服器，由能力較強的伺服器使用其運算能力與記憶體空間，進而求出答案，再將答案回傳。不過缺點是如果連不上伺服器將無法進行此項操作，服務就中斷了。另一種解決方案是使用所需運算能量較小的方法去尋求最接近的答案，而基因演算法應用在這類的問題上就相當的適合。

本文使用基因演算法的應用不僅僅只是解決最短行車路徑的問題，還需考量行車時所使用的行車速率，所以為解決最短行車時間問題。我們做了相關的實驗來說明基因演算法於這個問題上面的結果，而且也與智慧型行車資訊系統所給予的動態速率資訊做搭配，實驗出改良的成果。

第貳章 背景介紹

解決最短路徑的問題的演算法與基因演算法為本論文中最基本的二個演算法，最短路徑問題可以求得二點之間的最短路徑，而基因演算法是一種應用工具，適合去搜尋最適合的解。另外還有智慧型行車資訊系統的設置與功能介紹。

第一節 最短路徑問題

二點之間最短的距離為直線，這是每一個人都知道的事實，但是如果二點之間無直接路徑而需經過第三點才能到達，最短路徑問題就會浮現出來。在圖論上最短路徑問題可以分為 Single-source shortest path problem 與 All-pairs shortest path problem，兩者之間的差別為，Single-source shortest path problem 只是解決了起點與終點間最短路徑問題，著名的演算法有 Dijkstra Algorithm(DA)與 Bellman-Ford Algorithm。All-pairs shortest path problem 為求解所有 Vertices 間最短路徑的問題，常用的演算法為 Floyd-Warshall Algorithm。

基本行車導航問題只需要考慮起點與終點且不需要考慮 Weights 可能為負數的情況下，解決此 Single-source shortest path problem 只需要使用 Dijkstra Algorithm 即可。因其方法實作簡單，而演算法的複雜度只需要 $O(|V|^2+|E|)$，$|V|$為 Vertices 的個數，而$|E|$為 Edges 的個數，所以適合解決這個問題。不過在節點數目上的大量增加，此方法所要耗時的時間也就需要越多，且計算所需要的空間也是越多，在行動裝置(手持裝置)上，因必需運作 OS 與顯示地圖資訊，拿來運作的記憶體空間有限，如果節點數量過大，將無法使用這個方法所以如果不必保證可求得最佳解，那還有其他替代方法，而基因演算法就是其中一個方案。

第二節 基因演算法

基因演算法(Genetic algorithm, GA)為求近似解的搜尋技術[3-10]，使其最佳化的方法之一，具體概念是源自於自然界中的演化法則，成熟的基因演算法概念在 1975 年由 John H. Holland 在其 Adaptation in Natural and Artificial System 書中提出。Holland 透過自然界中生物基因的 DNA 編碼與繁殖的現象獲的靈感，認為在自然環境與人造環境中，很多的現象可以依其屬性進行編碼，就如同一組 DNA 序列一樣，而在物種的繁衍過程中，以產生下一代較適合環境的方法，透過函數的設計選出較適合的子代，而此構想就為完整的基因演算法。而基因演算法應用的範圍則為很廣，生物基因學、經濟學、遊戲理論、樣式辨認與工程上的控制、類神經網路、模糊理論、航空、太空軌道，都有應用的實例。

基因演算法的方法為，可分為以下幾個步驟進行：

1. 初使化編碼 (Initialization)：根據問題進行編碼，稱為染色體 (Chromosome Representation)，而染色體中代表一種狀態稱之為基因 (Gene)，編碼方式可以為位元串列(bits string)、整數串列或其他型式的串列，也分成固定長度編碼與可變長度編碼的不同方式。依問題的本身尋找最適合的編碼方式可以加速其求解的速度與逼進最佳解。

2. 選定適應函數(Fitness function)：Fitness Function 為計算染色體所代表的「適應值」，觀察其值的高低可以了解其適應的程度。後續決定是否留存則根據此值做判斷。

3. 選擇(Selection)：根據適應值的高低，採用輪盤法(roulette wheel)或競賽法(tournament)，高適應值的父代(parents)被選中的機率較高，低適應值的被選中的機率就較低，之後再產生新的子代(offspring)，所

以適應值會影響被選中的機率。

4. 交配(Crossover)：將選定的染色體進行兩兩交配，交配方法有單點法(one-point crossover)、雙點法(two-point crossover)與均勻交配(uniform crossover)，依照問題的情況再選擇適當的方法。

5. 突變(Mutation)：因自然界中在染色體的複制過程也不一定完全成功，而有時會產生一個基因的變化，正常的機率很低。在基因演算法中則為低機率的選擇一個染色體的基因，進行編碼的改變，則為突變。突變可以帶來搜尋未知解的可能，但也可能使系統平衡度下降。

6. 替換(Replacement)：以新產生出來的子代，替換掉本來的染色體，可以分成全部取代或部份取代。替換後以提高整個系統的適應值為目標。

7. 終結(Terminal)：當達到所設定的世代數(generations)或已達到最佳解，則系統即可結束而得到解。

以上的流程圖為

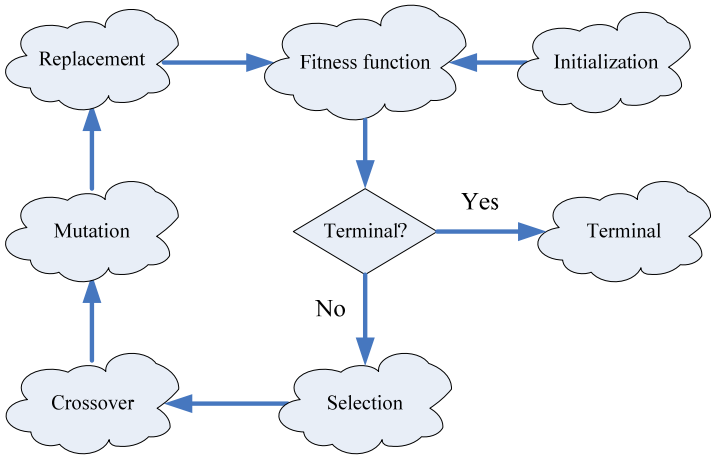


圖 2-1 基因演算法流程圖

如前一節所述，使用 Dijkstra Algorithm 相當適合解決小型的最短路徑問題，不過在大型的網路或需要即時的通訊應用上，此方法因其 Vertices 與 Edges 的增加，而所耗用的計算成本就越高，進而無法滿足需求。而基因演算法因可以調整演化的代數，染色體的表示方法和演算法本身的特性，在 Vertices 與 Edges 增加的同時，並不會快速的增加其計算成本，進而求出近似解以滿足這類的要求。

使用基因演算法尋找最短路徑的演算法有好幾個，例如 Munemoto's Algorithm、Inagaki's Algorithm 與 Chang's Algorithm。這些演算法中 Munemoto 使用了可變動長度的染色體表示法，Inagaki 使用了固定長度的表示法，而 Chang 所提出來的作法又優於以上二者，所以我們將使用此方法當作基礎求解最短時間路徑的問題。

真實的道路地圖上一個城市所具有的道路節點常常是超過上千個 nodes，如果跨越不同城市間道路節點將會上萬個，要求出最佳解則更為困難。此情況下，使用 Dijkstra Algorithm 來尋找最佳路徑的速度，將遠遠不及基因演算法，除非使用以 Dijkstra Algorithm 為基礎的近似演算法，不然所耗的時間差距將會相當的大。

一般以 GA 演算法所需要的運算時間多半以演算法收斂的代數為評估的指標，同時也需要看 chromosomes 的數量，但如果在有效的平行化處理中，chromosomes 的數量帶來的演算時間負擔將會降低很多。使用基因演算法所需要的記憶體空間就為 chromosomes 的空間與產生下一代的空間，其他變數需要運算空間相當的少，所以非常適合在小型記憶體的環境中運算。

第三節 智慧型行車資訊系統

智慧型行車資訊系統(The Intelligent Transportation System, ITS)的重點在於能提供地圖資訊及路徑導航等實用功能，並透過使用者的行車情況，利用 sensors 收集行車資訊應用於智慧型車輛運輸系統[11][12][15]如圖 2-2 所示。使用智慧型行車資訊系統是以 GPS 搭配地圖資訊系統進行行車導航，依照一開始的所設定的導航路徑，會逐步的指示所需行走的路徑，最後達到目的點。一般沒使用 ITS 的路徑導航系統為封閉式系統，使用者必須不斷地定時購買新的地圖資訊，方能保有最新版的資料。而加入 ITS 的使用者因為有與 Server 互動，進而可以獲得新的地圖資訊，以及即時的交通流量，導航的結果更符合實際需要。

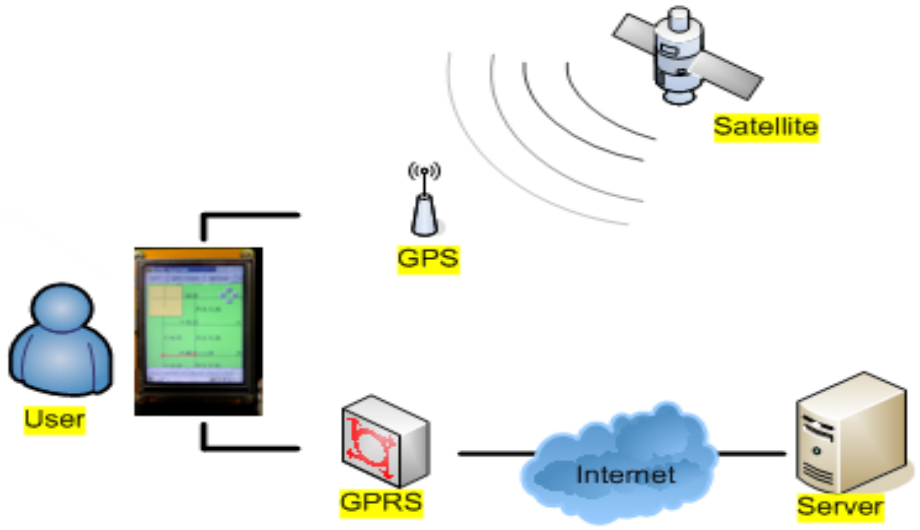


圖 2-2 智慧型行車資訊系統

第 1 節 全球定位系統

全球定位系統 (Global Positioning System, GPS)是美國國防部研製與維護，可滿足地表 98%的準確定位、測速和高精度的時間標準。該系統有 24 顆 GPS 衛星於天空中繞行，使用者最少需接收 4 顆衛星的資訊，才能迅速的確定在地球上的所在位置及海拔高度，能接收到的衛星數越多，定位出來的結果就越準確。

目前市售的 GPS 定位器定位的時間越來準確，以 Holux GR-213 的規格為例，其暖開機約 38 秒就可完成定位，如果再配合最近越來越流行的 AGPS(輔助全球衛星定位系統)的技術，將可縮短在 10 秒以內。

GPS 所傳送的格式有七種

表 2-1 GPS 資料標頭

NMEA 記錄	說明
GGA	定位後衛星定位資訊
GSA	一種偏差資訊, 說明衛星定位訊號的強弱狀態
GSV	可視衛星狀態
RMC	GPS 建議最小傳輸資料
RMB	建議最小導覽資料
RMA	一般民可取得之資訊
VTG	地表運動方向與速度

定位後可以取得 GGA 的資料，裡面有經度、緯度的資料，其資料格式如下

表 2-2 GPGGA 格式

名稱	實例	單位	敘述
訊息代號	$GPGGA		GGA 規範表頭
標準定位時間	161229.487		時時分分秒秒.秒秒秒
緯度	3723.2475		度度分分.分分分分
北/南半球指示器	N		北半球(N)或南半球(S)
經度	12158.3416		度度度分分.分分分分
東/西半球指示器	W		東半球(E)或西半球(W)
定位代號指示器	1		0: 未定位, 1:已定位
使用中的衛星數目	07		00 to 12
水平稀釋精度	1.0		0.5 ~ 99.9 米
海拔高度	9.0	米	-9999.9 至 99999.9 米
單位	M	米	
大地水準面分隔		米	
單位	M	米	
差分修正		秒	
基地台代碼	0000		
總和檢查碼	*18		
<CR> <LF>			訊息終點

第 2 節 嵌入式系統平台

1. 硬體環境

行動裝置或手持裝置都是嵌入式的一種，要在上面開發軟體可以準備一個開發板，我們使用的嵌入式開發平台必需具備觸控式螢幕、GPS 接收器，外部儲存裝置、網路功能。

圖 2-3 是長高科技(DMATEK)的產品 DMA2410，包含兩組 RS232 介面可以同時使用 GPS 及 GPRS 功能，外部儲存裝置可以透過 SD 界面儲存所需要的地圖資料，而觸控螢幕則是方便使用者進行輸入的工作，尚還有音效的輸出入，可以擴充功能成語音導航。

透過嵌入式系統上面的 IO Port，可以再與車輛上之行車電腦或感應器作溝通，以提供車輛的各種狀況給 ITS Server 做統整，由 ITS Server 分析資料、判斷狀況再回傳資訊給嵌入式的使用者。

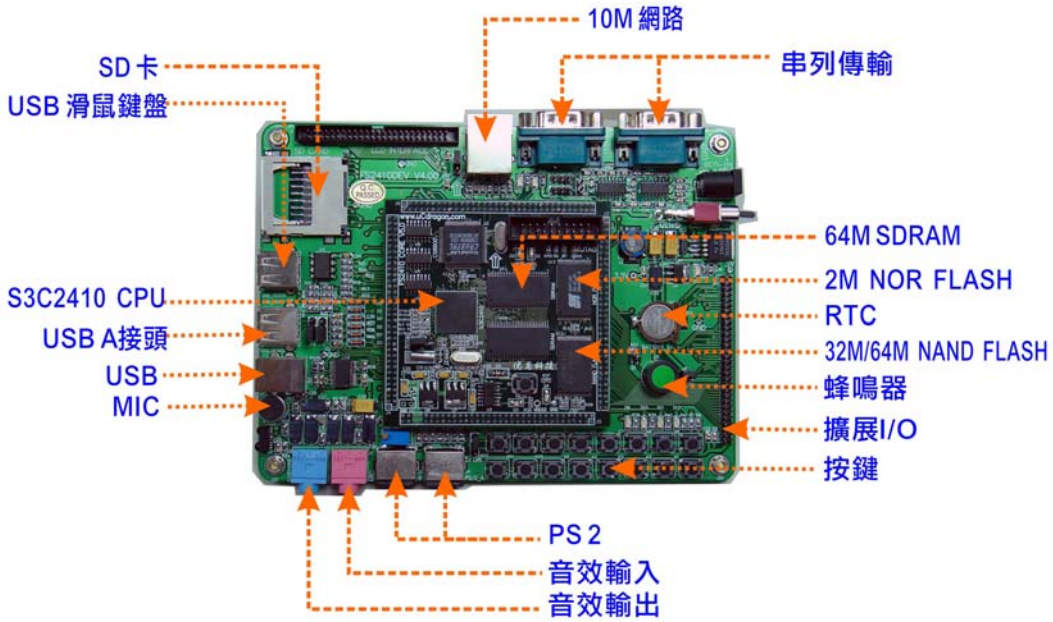


圖 2-3 DMA 2410

2. 軟體環境

因 DMA2410 平台上是一個 ARM9 的核心，有完整的 MMU 的功能，所以可以執行完整的 Linux 功能，不必特別牽就於 uClinux 這類特別的 Linux 核心，因為核心完整，所提供的各種系統支援也就相當完整。

Qt/Qtopia

Qt 是一套跨平台的 GUI framework，底層大部份的程式都是由 C++所完成，Qtopia 是一套專為那嵌入式系統上開發之圖形操作架構。這二樣軟體都是 Trolltech 商業公司所開發的[18]，使用的授權有分商業版與 GPL 二種授權方式，雖然是商業公司，但是 GPL 的授權方式卻一直維持的很久且更新快速，並提供完整的線上手冊，比起由社群獨立維護的 GTK+系列支援度是比較豐富。而 Qtopia 的架構如圖 2-4

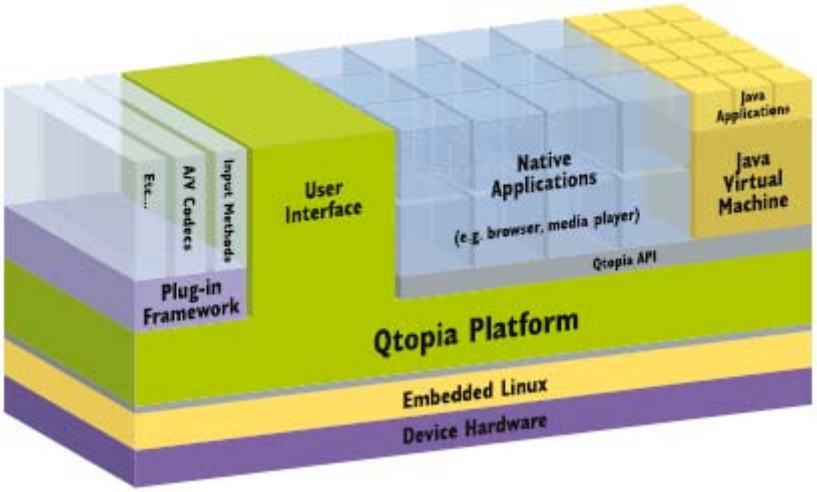


圖 2-4 Qtopia 平台架構

行車導航軟體

我們開發的行車資訊系統在嵌入式平台上共有四大子系統

(1) 地圖管理子系統(Map Manage Subsystem, MAP)

負責讀取地圖資料與管理地圖資料，當GUI 子系統需要顯示地圖資料時，需快速的將資料交給GUI 子系統。

(2) GUI 子系統(Graphics User Interface Subsystem, GUI)

為圖形使用者介面，負責處理一切的使用者的輸入，以及顯示相關的圖形資料，並管理地圖管理子系統、GPS 子系統、GPRS 子系統。

GUI 子系統與GPS 子系統取得定位資訊，再依此資訊與地圖管理子系統取得相關的地圖資料，再進行顯示動作，並依使用者的設定，再與GPRS 子系統連線，把相關的資料傳回伺服器管理子系統。

(3) GPS 子系統(Global Positioning Subsystem, GPS)

透過GPS 裝置所收集的定位資訊，並統計分析後，修正誤差，再把相關的資料儲存好，供GUI 子系統的GPS 模組讀取顯示資料。

(4) GPRS 子系統(General Packet Radio Service Subsystem, GPRS)

透過GPRS 發送器，就可以與伺服器進行網路連線，進行更新作業。並等候GUI 子系統進行網路連線。

整個軟體架構圖如圖 2-5，實際執行畫面如圖 2-6

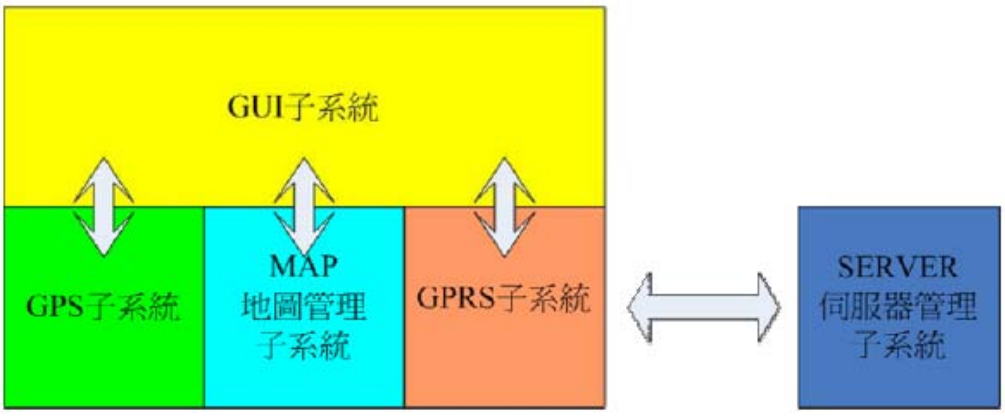


圖 2-5 軟體架構圖

圖 2-6 行車系統執行圖

第 3 節 ITS 系統平台

ITS Server是由一台桌上型的PC所擔任，使用的開發軟體為Microsoft Visual Studio，ITS Server必需處理各種運作中的行車導航系統所傳回的資料，紀錄路況、所在地位置、使用者所提供的資訊…等。然後分析有用且需要的資訊讓使用者選擇接收，以此增加一般行車導航系統目前所沒有提供的功能。

ITS Server所使用的平面道路資訊，可以是由交通部運輸研究所[16]所提供的，目前為全台最詳盡的地圖，包含國道、省道(含快速道路)、縣道、鄉道、都市道路、產業道路及無路名道路等既有道路，這些路網的比例都有達到1/5000。ITS Server的即時路況可經由交通部的公路總局[17]所提供的即時路況資料庫來達到即時的路況變更，使用ITS功能的使用者都可以快速而安全的達到目的地。

第參章 最短行車時間問題及其解法

本章節說明的是，如果把行車時所運行的速率考慮進來，則就會產生所需要多少行車時間問題。現實生活在行經 nodes 間，其行駛的速率並非完全一樣，所以還要考慮其變動速率的情況，而目標不再是最短距離，而改為最短時間內到達目的，以下將敘述問題情況。

第一節 最短行車時間問題簡介

在傳統的最短路徑問題上，往往只考慮跨越二點所需的代價，稱之為 costs，如無特殊定義時，在道路上則為 node 間距離。假設行車時為固定速率 V，則起點至終點的 Total Time = (Total Distance / Velocity)，所以求最短行車時間(T)即為最短距離(D)除以行車的速率(V)。

假設行車時為變動速率 v_{ij}，即 node i 至 node j 的最高行車速率，而 node i 至 node j 的距離為 d_{ij}，則每段最短行車時間 $T_{ij} = d_{ij} / v_{ij}$，此問題一樣可以使用 Dijkstra Algorithm 來解，需要將本來的 costs 代表的距離改成花費的時間即可，同時基因演算法亦可以解決相同的問題，即求解 minimize (T_{ij} · U_{ij})。

以上情況為基本的考量行車速率下的結果，但實際行車情況有可能因為改變交通工具或道路在不同時段有不同時速的限制而產生變化，但皆可使用基因演算法求解出最佳解。

第二節 最短行車時間問題之參數設定

此問題可以公式化成以下的參數，這些數學模型如下所示

Problem parameters:

N	set of all nodes
A	set of all links
S	source node, $\in N$
D	destination node, $\in N$
i,j	index of node, i,j , $\in N$
$<i,j>$	node i to node j, directional
E_{ij}	link node i to node j
d_{ij}	distance of node i to node j
v_{ij}	velocity of node i to node j

Problem decision variables:

T_{ij}	Cost time of node i to node j, $\in R^+$
U_{ij}	binary, 1 if the link from node i to node j exists in the routing path, 0 otherwise
t	Total drive time, $\in R^+$

在這個 Model 下，行車時為變動速率 v_{ij}，即 node i 至 node j 的最高行車速率，是向 ITS Server 取得。而 node i 至 node j 的距離為 d_{ij}，記載在本機的地圖資訊系統中。則每段最短行駛時間 $T_{ij} = d_{ij} / v_{ij}$，而想求得最短行駛時間 t，即為求解

Minimize $t = \sum_{i=S}^{D} \sum_{j=S}^{D} T_{ij} U_{ij}$ subject to

$$\sum_{\substack{j=S \\ j \neq i}}^{D} U_{ij} - \sum_{\substack{j=S \\ j \neq i}}^{D} U_{ji} = \begin{cases} 1, & \text{if } i = S \\ -1, & \text{if } i = D \\ 0, & \text{otherwise} \end{cases}$$

$$\sum_{\substack{j=S \\ j \neq i}}^{D} U_{ij} \begin{cases} \leq 1, & \text{if } i \neq D \\ = 0, & \text{if } i = D \end{cases}$$

$U_{ij} \in \{0,1\}$, for all i.

$T_{ij} = d_{ij} / v_{ij}$

第三節 最短行車時間問題之基因演算法解法

我們使用 Chang's Genetic Algorithm 為基礎，用以下的步驟來解決這個問題。[6][13][14]

(1) Genetic Representation: 使用變動長度的染色體表示，但長度最多為 N。染色體的起點為 S，終點為 D，例如有一路徑$(S \rightarrow N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow N_{k-1} \rightarrow N_k \rightarrow D)$。

(2) Population Initialization: 產生第一代的染色體，需依照母群體的大小來決定要使用多少個染色體來代表初始狀態。第二個要考慮的部份是產生來源是透探索式(heuristic initialization)或隨機式(random initialization)，而在此決定使用隨機式產生第一代染色體，因擔心用探索式的方法產生的第一代，其涵蓋的區域不夠廣。

(3) 適應函數(Fitness Function)：適應函數的選擇很明確的定義為

$$f = \frac{1}{\sum d_{ij} / v_{ij}}$$

f 表示為染色體的適應值(fitness value)，d_{ij} 為 node i to node j 的距離，v_{ij} 為 node i to node j 的速率。

(4) Selection: 使用 tournament 來進行，將會選擇較好的二個染色體進行交配，然而相同的染色體不應該被使用兩次以上。

(5) Crossover: 因 Genetic Representation 是可變動長度的染色體，交配的方法為在二個染色體中找出可以 crossover 的點後，再隨機選擇一點做 crossover point. 因其交配與突變完後，在染色體上面的表示式可能存在迴圈(loop)，這時可以進行修復，把這些重復的資訊去掉，在計算其適應值的時候才會正確。

(6) Mutation: 使用 5-10%的突變機率做為產生新路徑的方法。在原染色體中其中一點突變出去隨機尋找一條新的路徑到 D 的點來取代原先的路徑，太低的數值不容易變化，很難收斂最佳解，太高的數值又會造成收斂困難，不過設定多少要依問題的情況而定。

第肆章 實驗與結果

我們使用 ARM 9 S3C2410 嵌入式系統做為我們行動裝置的平台，使用一台 PC 主機當作 ITS Server 提供根據目前交通流量所允許行駛的速率資料。我們做了二種實驗，一種為固定速率下的情況，另一種是 ITS 會動態調整允許行駛的速率。

使用的演算法有基因演算法與 Dijkstra Algorithm，不過隨著 nodes 的增加，DA 所需求的運算空間越多，將無法在有限記憶體的嵌入式系統上運作，所以只列出 GA 的執行結果。

第一節 使用固定速率資訊以基因演算法尋找路徑

第 1 節 基因演算法單次導航

在此小節中，我們使用 ITS Server 一開始所提供的速率做為導航所需要的數值，並實驗二種不同的地圖，一種為虛擬的 square matrix map 與 real map。

1. Matrix Map

這部份的地圖是使用 4x4、8x8、16x16 與 32x32 的方陣地圖，設左上角為起點，右下角為終點，這樣所經過的距離為最長，可以充份的面臨各種狀況。每個 node 的距離為固定 240，如圖 4-1 所示。根據交通部公路總局的資料[16]，一般道路的行駛速率為 30-80，依照這個數值，我們隨機分佈行駛速率於地圖上，另外基因演算法所用的 crossover 機率為 90%，mutation 機率為 8%，最多演化 40 代，每一份實驗做 1000 次取平均值。

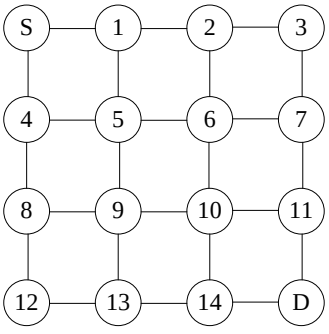


圖 4-1 4x4 square matrix map

表 4-1 欄位代表 map size，每列代表所使用的 chromosomes 數目，表格中的數值為平均達到最佳行駛時間所需要的演化代數。由表可知基因演算法的收斂速度很快，即使有 1024 個 nodes，收斂所需的代數也不到 30。

此外可以由 nodes 數目為 16 的欄看出，在地圖不大的情況下，收斂的代數大約在 8 代多就可以達到極限，無法再成長下去而求得最佳解。在其他 nodes 比較大的場合，其成長也是緩慢上升而已，不過會因為 nodes 的增加，所需要的代數增加比率也會大一些。

而 nodes 數目為 1024 的欄位可以看出來，因為 chromosomes 的不足，所以無法探索比較大的空間，會壓抑代數成長的速度，因此在 40 至 160，所須要的代數都會呈現沒什麼上升狀態。nodes 為 64 與 256 的這二欄也是有這個情況，只是比較不明顯。

所以綜合上面二種情況，演化所需要的代數，同時受地圖的大小與 chromosomes 數目影響非常的大。

表 4-1 平均尋找最佳路徑所需要的演化代數

		地圖的 NODE 數目			
		16	64	256	1024
染色體的數目	20	5.64	7.31	11.60	17.76
	40	7.37	9.19	11.91	20.18
	60	8.14	10.80	12.12	20.94
	80	8.40	11.88	12.69	20.14
	120	8.41	13.32	14.08	19.56
	160	8.70	15.04	15.36	20.11
	320	8.55	17.17	18.04	22.11
	640	8.42	18.71	21.55	24.73
	1280	8.45	19.41	24.33	26.65

表 4-2 代表的是最多演化 40 代內有達到最短行駛時間的機率。由表可知 chromosomes 的增加可以增加取得最佳路徑的機率，不過相對而言會如表 4-1 所示，所需要的收斂代數要增加而增加一些運算時間。

在比較小的地圖上，例如 nodes 為 16 或 64 的項目上，取得的解相當接近最佳解，誤差可以收斂到 3%以下。所以可以看出來增加 chromosomes 的數目是非常有幫助的。不過隨著地圖的大小增加與最佳解的誤差也成長到 25%，這是基因演算法上的一些限制，因為沒有探索所有解的空間而造成結果。

表 4-2 平均與使用 DA 演算法所求得的最佳解的誤差比

		地圖的 NODE 數目			
		16	64	256	1024
染色體的數目	20	8.79%	20.34%	28.15%	43.32%
	40	4.72%	17.06%	25.99%	32.52%
	60	3.07%	15.48%	25.37%	30.65%
	80	2.22%	14.16%	24.84%	29.14%
	120	1.36%	11.77%	23.64%	28.06%
	160	0.85%	10.37%	22.65%	28.20%
	320	0.22%	6.75%	21.33%	27.19%
	640	0.09%	4.65%	18.63%	26.57%
	1280	0.03%	2.70%	16.44%	25.47%

2. Real Map

真實地圖是使用台灣的台中市的城市地圖為實驗地圖，此份地圖一共包含有8039 個 nodes，使用的速率依交通部公路總局所規定的道路等級分為六種不同等級的允許行駛速度，而起始位置與終點位置則為隨機選擇，實驗結果如表 4-3。

在下表中得知 chromosome 的增加從 40 到 1280，而誤差值也從 269%，成功的縮減到 42%左右，在如此大的地圖中，基因演算法所需要的代數仍然控制在 50 代之內，可以顯示基因演算法的收斂效果仍然不錯。

表 4-3 使用基因演算法於 Real Map 的結果

		平均演算代數	平均誤差
染色體的數目	40	24.53	269.17%
	80	29.97	152.65%
	160	35.70	120.99%
	320	40.16	84.31%
	640	44.47	64.10%
	1280	47.24	42.81%

第 2 節 基因演算法多次導航

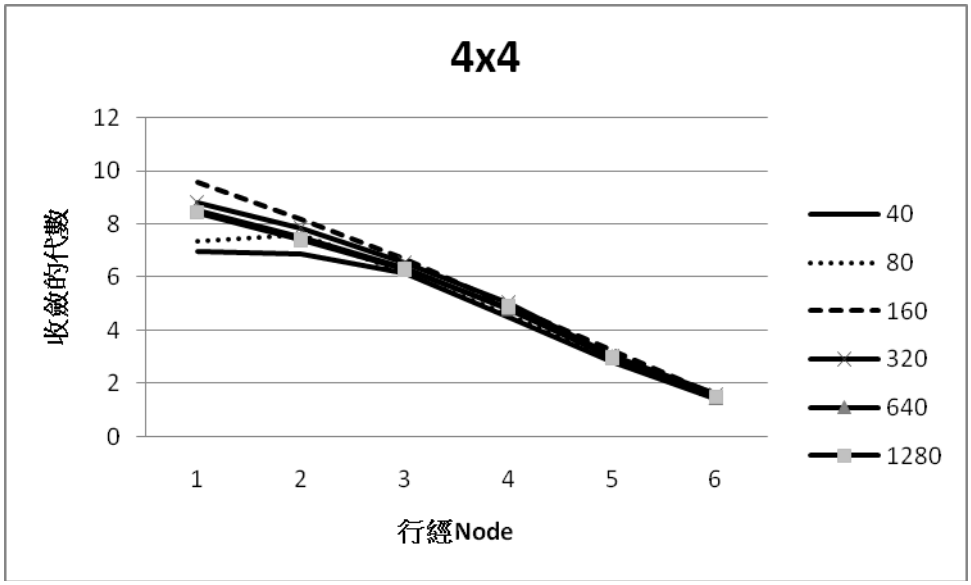
此小節所討論的是，因為真正的行車所需要的時間是相當長的，所以可以用多次導航來增加基因演算法的準確度。我們一樣使用 Square matrix map 為我們的實驗地圖，並在每次的交叉點重新做基因演算法的導航。基因演算法所用的 crossover 機率為 90%，mutation 機率為 8%，最多演化 40 代，每一份實驗做 100 次取平均值。所得的結果如表 4-4。

表 4-4 展現出，多次使用基因演算法可以改良的效果，由地圖大小為 16，因為多次導航的效果，原本在 chromosome 數為 640 時還有接近 0.1%的距離誤差，成功的達到 100%的最佳導航效果。而在地圖大小為 64 中也是使原本還有 3%左右的誤差，成功的縮減到 1%以下。就算在比較大的地圖當中，多次導航的效果仍然顯示出可以縮減所需要的行駛時間。

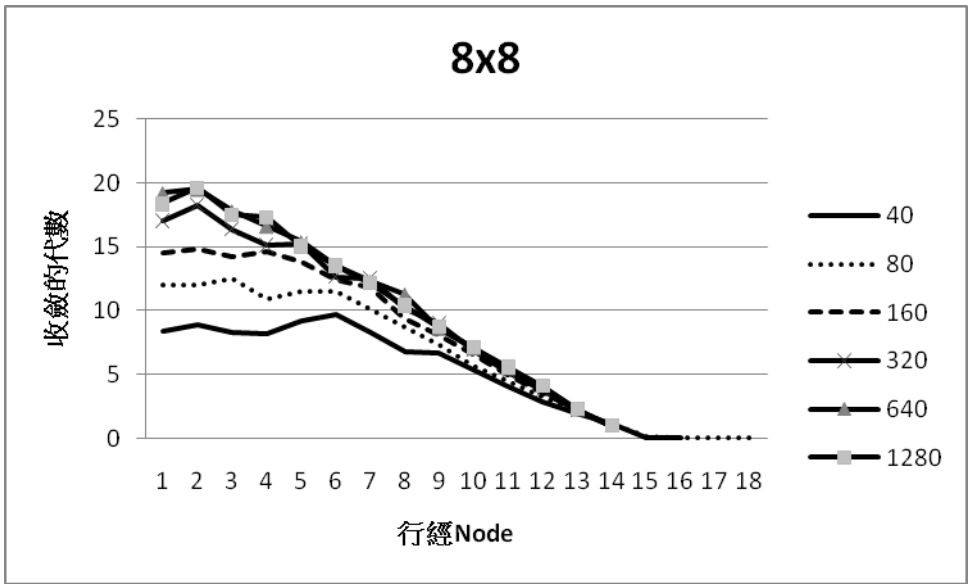
表 4-4 多次導航的效果比較

		地圖的 NODE 數目							
		16		64		256		1024	
		Before	After	Before	After	Before	After	Before	After
染色體的數目	40	4.72%	5.23%	17.06%	13.03%	25.99%	23.09%	32.52%	30.54%
	80	2.22%	1.16%	14.16%	9.54%	24.84%	20.13%	29.14%	28.10%
	160	0.85%	0.27%	10.37%	7.13%	22.65%	17.71%	28.20%	26.02%
	320	0.22%	0.12%	6.75%	4.21%	21.33%	13.53%	27.19%	25.02%
	640	0.09%	0.00%	4.65%	2.56%	18.63%	10.70%	26.57%	23.14%
	1280	0.03%	0.00%	2.70%	0.87%	16.44%	9.98%	25.47%	20.55%

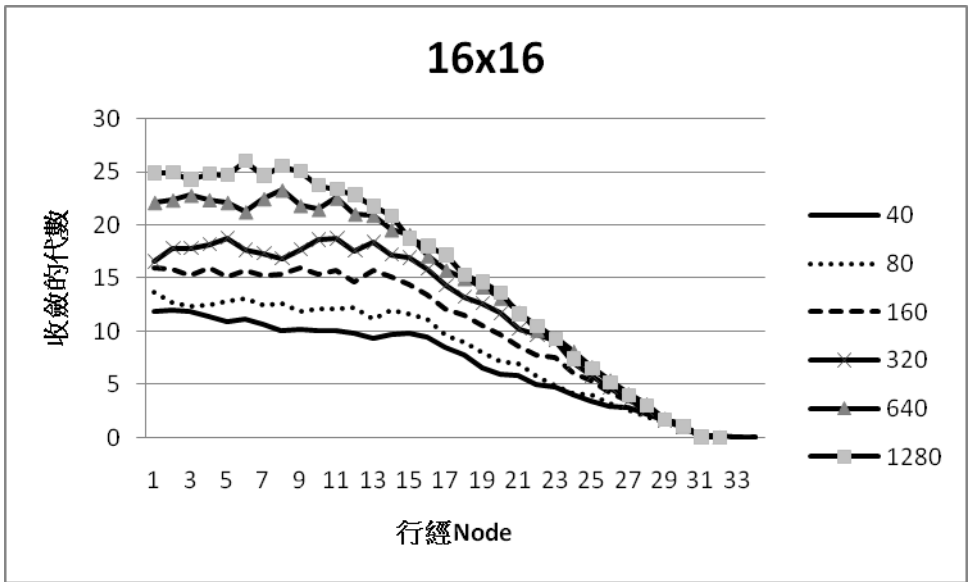
此外由圖 4-2-(a)至 4-2-(d)中顯示在運算過程代數的收斂情況，可以了解越接近終點，所需要的演算代數就越少，所以在使用多次導航所需要的運算 Power 並不會如一開始的這麼高，而是會慢慢遞減下來。



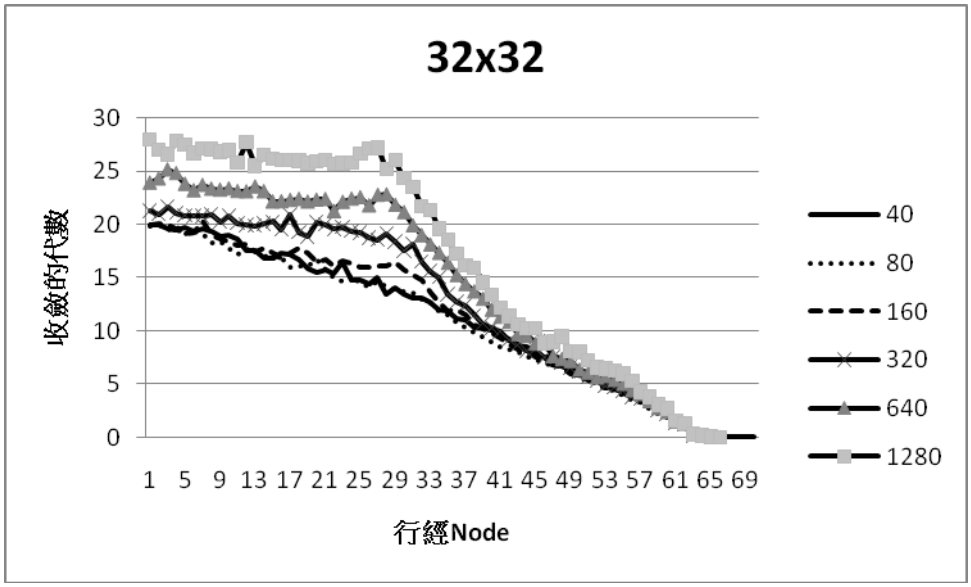
(a)



(b)



(c)



(d)

圖 4-2 演算代數收斂圖

第二節 使用變動速率資訊以基因演算法尋找路徑

在思考 ITS Server 的實際運作情況是會隨著現實的交通流量不斷的更新即時道路速度，而導航演算法也需要把變動的結果考慮進去，Dijkstra Algorithm 這類的演算法，遇到這種問題就是重新設定起點與終點再重新做一次，每次計算所消耗的電力都一樣。而我們的基因演算法可以使用上次演算的結果，保留一部份的 chromosome，再產生一部份新的 chromosome，新的運算就會比較快速的收斂完成，達到節省電力與時間的需求。

我們的實驗地圖仍然以 Square matrix map 當目標，以左上角為起點，右下角為終點的來進行導航實驗，在行經交叉點的時由 ITS Server 更新部份資訊，更動的部份分為 5%、10%、20%，代表意義為有 5%、10%、20%的速率資料被更新了。在這個情況下我們以基因演算法進行運算，並且與未參考 ITS Server 所提供的資訊，以 DA 演算法算出的原始最短時間路徑做一個比較。結果如表 4-5。

另外我們考慮如果在比較大的交通流量下或更大的地圖下，行車的速率會因車流量的關係變化的範圍更大，而跨越了更多不同的道路等級也使可行駛的速率的變化更大，在此我們把行車速率由 30-80，改為 20-110。結果如表 4-6

保留上次導航結果的一部分的 chromosome 可以減少下次導航所需要的時間，這部份我們實驗以保留 50%的 chromosome 在下一個交叉點直接拿來應用，展示收斂的加速效果。

表 4-5 可以顯示出來因為 ITS Server 不斷的更新最新道路資訊，所以使用基因演算法進行多次導航，不斷的重覆修正最佳路徑，所獲得的結果反而比原始未參考更新資訊而走原始的路徑的情況還好。

我們以 16x16 的地圖上，變動率為 10%，使用的 chromosome 為 1280，這時可以比未應用 ITS 的變動速率資訊，單純使用 DA 找出來的最短時間路徑還要快上 6.64%。所以表中負數的意思即為比原始路徑所需要的行駛時間還要來得少。

在 8x8 的地圖上，資料顯示在 10%變動率上會從 4.28%變化到-5.46%，因為一開始的 chromosome 還不足，所以探索的情況還不夠多，等到數量到一定程度後，效果就明顯的出現了。根據實驗，在地圖的變化量越大的情況，由 5%到 20%，所求得的最短行車時間亦來的更好，所以應用 ITS 的動態更新速率帶來的效果是非常的明顯。

表 4-5 應用 ITS 的即時更新資訊與原始最短時間路徑的比較 1

		地圖的 NODE 數目					
		16			64		
變動%數		5	10	20	5	10	20
染色體的數目	40	-0.39%	1.28%	-3.15%	6.51%	4.28%	-0.85%
	80	0.00%	-1.65%	-2.47%	3.14%	0.49%	-3.19%
	160	0.00%	-1.06%	-2.97%	-0.15%	-0.75%	-5.96%
	320	-0.60%	-1.76%	-4.64%	-0.93%	-4.76%	-7.52%
	640	-0.73%	-2.08%	-3.43%	-3.36%	-4.75%	-8.54%
	1280	-1.44%	-2.83%	-4.49%	-2.53%	-5.46%	-9.70%
		地圖的 NODE 數目					
		256			1024		
變動%數		5	10	20	5	10	20
染色體的數目	40	10.14%	4.32%	1.69%	8.53%	3.22%	1.56%
	80	6.89%	3.38%	-0.95%	5.35%	2.71%	0.86%
	160	3.78%	0.29%	-1.50%	5.32%	1.19%	-0.87%
	320	2.50%	-1.93%	-4.74%	3.03%	0.07%	-1.22%
	640	0.27%	-4.76%	-6.97%	2.06%	-1.97%	-3.17%
	1280	-1.14%	-6.64%	-8.44%	-0.58%	-2.77%	-3.98%

表 4-6 的資料是如果道路上可行駛的速率變化更大的情況下的比較圖表，這部份的數值是考量到更大的地圖範圍及更廣泛的速率限制情況，用的速率可以從 20 到 110，所以最大值與最小值的差距倍率更大。

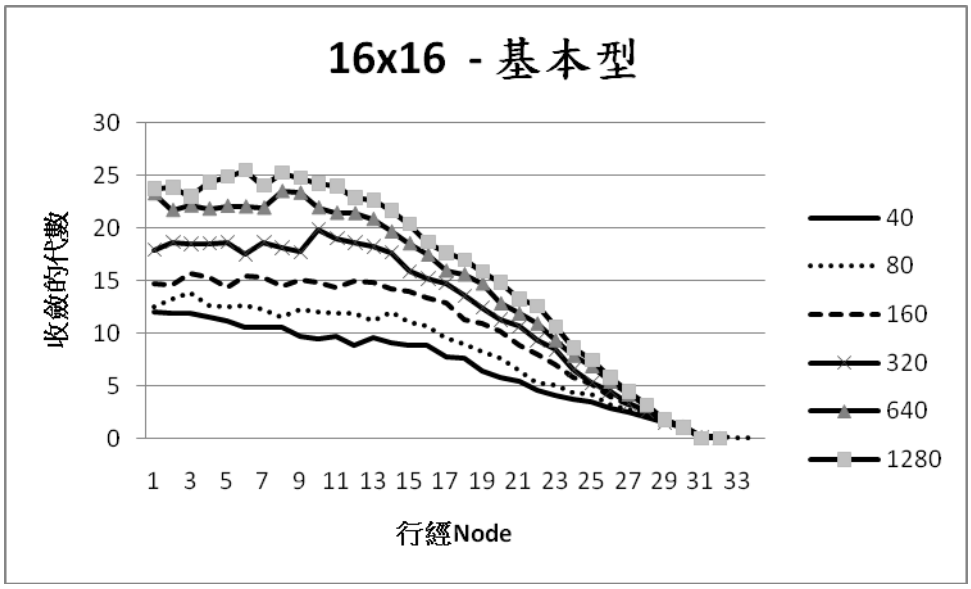
在 8x8 的地圖上，資料顯示在 10%變動率上會從 5.72%變化到-10.3%，這個變動幅度也比在表 4-5 中，4x4 的地圖，變更率 10%，道路可變動速率 30-80 間來的更大。同樣地，變更率由 5%到 20%，縮短的幅度也就更大，所以只要變化越大的速率系統，我們就更需要 ITS 來幫我們做動態的資訊更新。

我們導航結果會縮短的主要原因是有參考最新的 ITS Server 所提供的資訊兼基因演算法使用多次導航都有套用最新的資訊進去。相對而言如果不使用這些資訊，反而更容易因為其他人都只參考預設值，同時走入快速的路段，進而流量暴增，使這個路段變得很難走，這樣導航的意義就失去了。

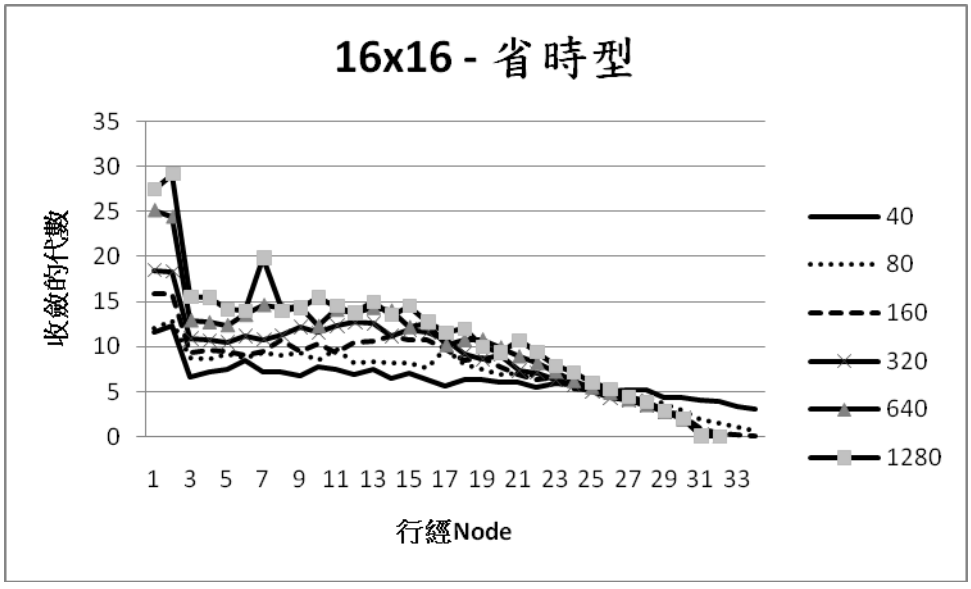
表 4-6 應用 ITS 的即時更新資訊與原始最短時間路徑的比較 2

		地圖的 NODE 數目								
		16			64			256		
變動%數		5	10	20	5	10	20	5	10	20
染色體的數目	40	7.19%	2.30%	0.29%	11.49%	5.72%	0.37%	14.18%	8.36%	0.86%
	80	1.77%	-1.89%	-2.73%	6.42%	-2.03%	-6.92%	8.85%	1.12%	-2.57%
	160	-1.61%	-2.76%	-4.69%	1.93%	-6.64%	-8.53%	5.01%	-0.59%	-8.64%
	320	-0.28%	-1.42%	-4.16%	-2.03%	-8.23%	-9.39%	-1.10%	-8.10%	-12.90%
	640	-1.71%	-3.77%	-7.01%	-4.66%	-9.07%	-12.02%	-0.70%	-13.50%	-14.20%
	1280	-2.29%	-2.65%	-5.86%	-6.14%	-10.30%	-11.76%	-9.20%	-16.20%	-16.30%

在 ITS Server 會不斷的提供更新資訊下，我們可以留存上次的運算結果來加速運算速度。如果以保留一半上次的運算結果而且 ITS Server 的地圖變動率為 10%下，16x16 的 matrix map 中，以圖 4-3-(a)，原先在 chromosome 為 1280，原先收斂的代數一開始約都為 25 左右，大約要經過 11 個 node 後才會下開始下降。如果在圖 4-3-(b)中，如果留存一半的 chromosome，一開始所需要代數約為 26，在第三次的導航，所需要的收斂代數就只要 15 代，而且後續也是慢慢下降。



(a)



(b)

圖 4-3 16x16 中有留存上一代的染色體的收斂圖比較

第伍章 結論與未來工作

行車導航系統在應用上日益增加，一個好的行車導航系統不應該只是單純的計算出起點與終點的最短路徑而已，可以透過一些 sensors 收集道路導航資訊與透過網路下載目前的車流量資訊，進而更準確的判斷出最適合的行車路徑，這些都是行車導航系統發展的目標。

我們的系統加入了一些額外資訊，不是只有地圖資訊上二點的經緯度，還有 ITS Server 所傳來的最新導航所用的車速、每一路段目前所能行駛的最高車速，所以簡單方法所需要計算量都會相當的大，就需要使用一些近似解的方法，而基因演算法在道路節點數量相當多的情況下，不需要經過太複雜的計算即可快速的求得近似解，正是一個良好的解決辦法。

基因演算法解決了第參章所提的最短行車時間的導航需求，一樣可以解決相同的類似問題，甚至是更多變因進來。越多的變因使系統越複雜，但是在基因演算法的特性下，並不會增加多少計算資源，一樣是透過演化法則下，尋找出最適合的路徑，這是一個相當好的特性。

目前我們所尋找出來的最短時間路徑在大型的地圖下效果還未很好，要解決這個問題目前我們必需要使用更好的探尋方法來避免只收斂到區域最佳解的情況，將來我們可以按照地圖的方向性，盡量排除不可能的路徑，使探索的速度加快，同時也增進找到更佳的解的可能。

未來我們將考慮更多的系統變因，例如轉彎的次數、道路的安全性、特殊使用者習慣，依不同的使用者的習慣，決定不同的演化方式，帶來更便利與舒適的行駛路徑，真正解決以一般最短路徑為主的導航演算法所尋找出來但不一定適合駕駛習慣的路徑的問題。

參考文獻

[1] Shaojun Feng and Choi Look Law, “Assisted GPS and its impact on navigation in intelligent transportation systems”, in *Proc. Intelligent Transportation Systems*, 2002, pp. 926-931

[2] Victor Chang, “Web Service Testing and Usability for Mobile Learning”, in *Networking, International Conference on Systems and International Conference on Mobile Communications and learning Technologies*, 2006, pp. 221-227

[3] Shubin Xu and James C. Bean, “A Genetic Algorithm for Scheduling Parallel Non-identical Bath Processing Machines”, in *Computational Intelligence in Scheduling*, 2007, pp. 143-150.

[4] M. Munemoto, Y. Takai, and Y. Sato, “A migration scheme for the genetic adaptive routing algorithm”, in *Proc. IEEE Int. Symp. Circuits and Systems*, 1999, pp. 137-140.

[5] J. Inagaki, M. Haseyama, and H. Kitajima, “A genetic algorithm for determining multiple routes and its applications”, in *Proc. IEEE International Symposium on Circuits and Systems*, 1999, vol. 6, pp. 137-140

[6] Chang Wook Ahn and R.S. Ramakrishnal, “A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations”, in *Evolutionary Computation, IEEE Transactions,* 2002, pp. 566-579.

[7] Mitsuo Gen and Lin Lin, “A New Approach for Shortest Path Routing Problem by Random Key-based GA”, in *Proc. Genetic and evolutionary computation*, July 2006, pp. 1411-1412

[8] Zhenjiang Li, J. J. Garcia-Luna-Aceves, “Quality of service in wireline networks: A distributed approach for multi-constrained path selection and routing optimization”, 2006

[9] Qinqfu Zhang and Yiu-Wing Leung, “An orthogonal genetic algorithm for multimedia multicast routing”, in *Evolutionary Computation, IEEE Transactions*, 1999, pp. 53-62

[10] Yue Chengjun and Jing yuanwei, “Solving the Problem of the Link Optimizing and Delay-constrained Multicast Routing Based on GA”, in *Control Conference Chinese*, 2006, pp. 1783-1786.

[11] Housroum H., Hsu T., Dupas R. and Goncalves G., “A hybrid GA approach for solving the Dynamic Vehicle Routing Problem with Time Windows”, in *Information and Communication Technologies*, April 2006, pp. 787-792

[12] Hitoshi Kanoh and Nobuaki Nakamura, “Route Guidance with Unspecified Staging Posts Using Genetic Algorithm for Car Navigation Systems”, in *Proc. Intelligent Transportation Systems*, 2000, pp. 119-124

[13] Yi-Liang Xu, Meng-Hiot Lim and Meng-Joo Er, “Investigation on Genetic Representations for Vehicle Routing Problem”, in *Systems, Man and Cybernetics, IEEE International Conference*, 2005, pp. 3083-3088

[14] Mattiussi. C and Floreano. D., “Analog Genetic Encoding for the Evolution of Circuits and Networks”, in *Evolutionary Computation, IEEE Transactions*, 2007, pp. 596-607

[15] Young-uk Chung and Dong-Ho Cho, “Enhanced soft-handoff scheme for real-time streaming services in intelligent transportation systems based on CDMA”, in *Intelligent Transportation Systems, IEEE Transactions*, 2006, pp. 147-155

[16] Directorate General of Highways, MOTC http://www.thb.gov.tw/index.aspx

[17] Institute of Transportation. MOTC http://www.iot.gov.tw/mp.asp

[18] Trolltech, a Nokia company http://www.trolltech.com/

[19] Chu-Hsing Lin, Jui-Ling Yu, Jung-Chun Liu, Chia-Jen Lee, “Genetic Algorithm for Shortest Driving Time in Intelligent Transportation Systems”, in *Multimedia and Ubiquitous Engineering*, 2008, pp.402-406