國立臺灣大學理學院地理環境資源學系

碩士論文

Department of Geography

College of Science

National Taiwan University

Master Thesis

空氣污染暴露路線選擇空間決策支援系統之研究

A Spatial Decision Support System for Route Choice of Air Pollution Exposure

曹宇鈞

Yu-Chun Tsao

指導教授：孫志鴻　博士

Advisor: Chih-Hong Sun, Ph.D

中華民國 108 年 7 月

July, 2019

doi:10.6342/NTU201903250

國立臺灣大學碩（博）士學位論文
口試委員會審定書

空氣污染暴露路線選擇決策支援系統之研究

A Study on Decision Support System for Route Choice of Air Pollution Exposure

本論文係曹宇鈞君（R06228020）在國立臺灣大學地理環境資源學系、所完成之碩（博）士學位論文，於民國 108 年 6 月 14 日承下列考試委員審查通過及口試及格，特此證明。

口試委員：

張志沛 （簽名）		
周雪珍 （指導教授）	蔡博文	

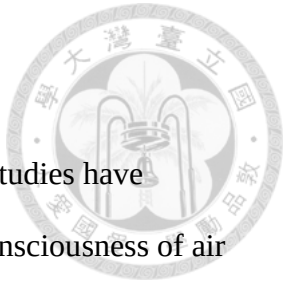
doi:10.6342/NTU201903250

摘要

世界上許多城市面臨空氣污染的問題，多數研究已證實空氣污染對於人體健康的危害，民眾對於空氣品質的意識快速高漲，雖然近幾年政府提出相關減少空氣污染排放的方案，但無法立即且大量的減少污染排放源，在改善空氣品質的過渡期中，透過預警與建議的方式，給予民眾對於空氣品質的資訊，輔助活動上的決策將能降低空氣污染暴露。空氣污染暴露量的計算，與測站及人類活動在時間及空間上都有密切的關係，隨著近幾年環境微感測器的數量持續增加，空氣污染監測資料於空間及時間上的密度有所提升，更能即時反應民眾所接觸到的空氣品質狀態，而在目前的預警應用上，對於空氣污染暴露量的路線選擇仍有待發展，過去研究在於空氣污染暴露量的評估上，多數強調最低暴露量路線所帶來的益處，卻忽略了實際生活中，民眾對於路線選擇上的成本考量以及路線變化後所帶來的影響，且在實際應用上，將需要蒐集及維護網絡圖層以進行路線規劃，資料的更新及維護上將是一個問題。故本研究利用微感測器資料，進行都市區域空氣污染暴露量的推估，考量距離與空氣污染暴露量進行路線規劃，提供日常生活中路線選擇上的幫助，並基於線上地圖服務建置空間決策支援系統，嘗試改善資料蒐集及維護上的問題。

Keywords：空氣污染暴露量、路線規劃、WebGIS、空間決策支援系統

doi:10.6342/NTU201903250



Abstract

In this world, many cities are facing air pollution problems. As studies have confirmed that air pollution is harmful to human health, the public consciousness of air quality has been raised rapidly. Although the government has proposed a plan to reduce air pollution emissions in recent years, it cannot reduce the source of pollution immediately and massively. During the transition period to improve air quality, early warning and recommendations will be given to people about air quality information, and decision-making on supporting activities will reduce air pollution exposure. The location of the air quality sensor and human activities at different times and spaces will affect the calculation of air pollution exposure. In recent years, the number of environmental micro-sensors has continued to increase, the spatial and temporal resolution of data has increased, and air quality values can be received more quickly. In the current air pollution warning application, the route selection for air pollution exposure still needs to be developed. In past studies, the assessment of air pollution exposure mostly emphasized the benefits of the minimum exposure route, but ignored the cost of route selection and the impact of route changes in real life. In practical applications, they will need to collect and maintain network layers for route planning. Updating and maintaining the data on the system will be a problem. Therefore, this study use the micro-sensor data to estimate the exposure of urban air pollution, consider the distance and air pollution exposure to carry out route planning, and provide assistance in route selection in daily life. And based on online map services to build a spatial decision support systems, try to improve data collection and maintenance issues.

Keywords： Air pollution exposure、Route planning、WebGIS、Spatial Decision Support Systems

doi:10.6342/NTU201903250

目錄

doi:10.6342/NTU201903250

圖目錄

doi:10.6342/NTU201903250

doi:10.6342/NTU201903250

表目錄

doi:10.6342/NTU201903250

第一章　緒論

1.1 研究動機

世界上許多區域都面臨空氣污染的問題，近幾年許多研究陸續證實了空氣污染對於健康的影響，其中粒徑小於 2.5 微米的懸浮微粒 (particulate matter, PM) 可能導致人體罹患缺血性心臟病、腦血管病、慢性阻塞性肺病、肺癌和下呼吸道感染等疾病 (Cohen et al., 2017)，世界衛生組織(World Health Organization, WHO)的報告中也提及，於 2016 年約有 420 萬人因為戶外的空氣污染暴露而導致過早死亡(World Health Organiztion, 2018)。

世界衛生組織於 2005 年提出的空氣品質準則(Air Quality Guidelines, AQGs)，對於懸浮微粒(particulate matter, PM)的年平均及 24 小時平均，給予了一個建議的數值，希望能作為各國於空氣污染治理上的目標，減少空氣污染對於人體健康的危害(World Health Organiztion, 2018)。在於都市區域空氣污染物的主要來源為汽機車排放，首當其衝的受害者，即是生活於都市內的居民，且又以機車通勤族為最直接的受到空氣汙染的影響。政府雖然能透過政策及法規進行空氣污染的治理，但於此過程中，民眾仍然生活在高空氣污染暴露的環境下，因此透過預警的方式，提供民眾在於活動上的決策資訊，將會是目前減少空氣污染暴露的方法之一。

近年民眾對於空氣污染的意識提高，常因大氣擴散不佳或是境外污染物的影響，導致空氣品質達到空氣品質指標(Air Quality Index, AQI)中對人體非常不健康的等級，造成氣喘、呼吸道發炎、肺部……等等疾病的發生。行政院環保署雖有設立空氣品質測站，但此監測數值代表著一個大區域的空氣品質平均值，民眾更想了解的是居住地附近的空氣品質狀況。隨著物聯網(Internet of things, IoT)及微

doi:10.6342/NTU201903250

型感測器技術的成熟，使得環境感測裝置成本降低，更容易的設置於生活環境中，且能獲得更大量的感測數據進行預警或分析使用，2015 年 LASS(Location Aware Sensor System)計畫以開源和公益的方式發起，將 Airbox 及其他自製的微型感測器架設於自家周圍以偵測空氣品質狀況，很快地引起社會共鳴，微感測器持續的部屬於生活中，提供小區域且較即時的空氣品質狀況，這使得民眾可以透過監測數據了解到一些地區性、即時性的事件，來進行防護等措施。

空氣品質微型感測器的出現，使得環境感測數據於空間及時間密度上有所提升，但在預警應用上，目前許多應用程式的通報機制僅考慮使用者所訂閱的測站，卻忽略了使用者活動時於空間上的變化，導致發出的警訊無法完全代表使用者所處位置的空氣品質，Park and Kwan (2017)比較了考量空氣污染的時空變化以及個人的移動模式的不同組合，進行空氣污染暴露的計算，說明過去許多空氣污染暴露的研究，忽略前述任一參數時，也許會導致暴露量計算的誤差。

由於民眾活動時會於空間上產生變化，預警或建議方式就不能單單侷限於點位上的通報，而需要路線選擇和規劃上的建議。Park and Kwan (2017)透過 google map direction service 模擬了 80 筆的日常移動，計算了 24 小時的空氣污染暴露量；Li et al. (2017)透過 dijkstra 演算法進行最低空氣污染暴露量路線的計算，並與日常路線進行比較；Molter and Lindley (2015) 模擬小學兒童上學路徑受到空氣污染的影響。

過去的研究均希望提供空氣污染暴露風險的建議，但多數皆提供最低暴露量的結果，以路線選擇為例，當民眾選擇最低空氣污染暴露路線時，相反的可能需要付出更多的時間成本在於該次旅程中，由於 dijkstra 演算法的特性將持續尋找最低暴露量的線段前進，可能大幅改變行經路線，導致使用者不熟悉此路況，雖

doi:10.6342/NTU201903250

然降低了空氣污染的暴露量，卻可能大幅改變原有路線，導致使用者的不熟悉，進而提升其他潛在的安全性問題。如何在移動成本與空氣污染暴露量之間進行抉擇，推薦一條符合成本效益的路線給民眾，如 Homayoon et al. (2015)將距離與暴露量賦予不同的權重值後，再進行最佳路徑的分析，因此獲得了一條考量距離及空氣污染暴露量的路線，希望給予使用者自行調整權重值，以符合個人的偏好。

然而過去研究的成果，均是基於路網圖資以 dijkstra 或是其他變形算法進行路徑分析，在系統建置、維護及擴大服務範圍時，將面臨資料蒐集與更新的問題，將可能影響服務的提供，若是透過線上地圖服務如 Google Maps 進行系統的開發，將能夠免除資料蒐集及維護的問題，並且擁有更完整且豐富的路況資訊及興趣點(Point Of Interest, POI)能提供給使用者。

基於以上問題，本研究想在考量距離與空氣污染暴露量的情況下，提供既符合移動成本且能降低空氣污染暴露的路線，整合相關空間資訊技術進行系統的建置，並引入線上地圖服務，嘗試改善過去研究中資料維護及更新上的問題，提供一套決策支援系統讓使用者能依照自身的偏好，且基於熟悉的路線下進行修正，給予民眾在於路線選擇上的決策建議。

1.2 研究目的

基於前述動機可知，過去研究對於空氣污染暴露量的評估建議上，多數均是提供最低污染暴露量給予民眾，將可能大幅度改變原有使用者所熟悉的路線，且實際於路線的選擇上，民眾將對於距離與空氣污染暴露量進行抉擇，以評估移動成本。在應用層面上，路網圖資與道路資訊的蒐集及更新維護上將會是一個問題。因此本研究目的是要建立一套以空氣品質資訊為基礎的決策支援系統，協助民眾在可接受的暴露風險及移動時間成本下，選擇一條較佳的移動路線，為完成

doi:10.6342/NTU201903250

此一研究目的，本研究之研究問題如下：

- 探討考量空氣污染暴露量及移動成本的情況下進行路線修正的方法。
- 整合相關地理資訊技術建構一套決策支援系統，以協助民眾考量距離與空氣污染暴露量下選擇較佳的移動路線。

1.3 研究限制

在路線暴露量計算的部分，本研究僅使用微感測器數據作為路線暴露量的計算依據，不同的移動模式之間僅有呼吸量有所差異，無法確保不同移動行為時所對應到的空氣汙染暴露量。

於空間推估的部分僅使用反距離權重法進行，實際上空氣污染可能受到許多環境因子的影響，而不單單只是考量距離的影響來推估空氣污染數值。

doi:10.6342/NTU201903250

第二章　文獻回顧

本研究首先針對空氣污染對於人體健康上的危害進行回顧；第二節則回顧本研究將使用到的空氣品質資料，對於目前不同來源的數據進行比較，並了解資料的處理方式以及在應用上如何呈現與視覺化；第三節將對於空氣污染資訊於路徑選擇上的應用進行探討，了解空氣污染暴露量的計算方式，並比較過去研究中如何在考量空氣污染暴露量下進行路徑的選擇。

2.1　空氣污染現況及對健康的影響

空氣污染是多種有毒化學物質和氣體的混合物，空氣污染物會根據來源與氣象條件的不同而有不同的種類，較常被注意的空氣污染物有可吸入顆粒物、氮氧化物、二氧化碳、臭氧和揮發性有機化合物，各國政府也會訂定自己的空氣污染指標進行監控，以台灣為例，由行政院環境保護署 (2018a)所制定的空氣品質指標，包含了細懸浮微粒 (particulate matter of a diameter less than 2.5 micrometres, PM2.5)、懸浮微粒 (particulate matter of a diameter less than 2.5 micrometres, PM10)、二氧化硫 (sulfur dioxide, SO_2)、氮氧化物(Nitrogen Oxides, NO_x)、一氧化碳 (carbon monoxide, CO)及臭氧 (ozone, O_3)。

其中懸浮微粒為空氣污染的一種指標性物質，對於人體健康的影響大於其他污染物，由懸浮在空氣中的固體或液體物質組成，主要成分包含硫酸鹽、硝酸鹽、氨、氯化鈉、炭黑、礦物粉塵和水。當懸浮微粒的粒徑小於 10 微米(μm)時，將能通過人體纖毛及黏液進入到肺部；粒徑小於 2.5 微米的顆粒，更能穿過肺部的屏障進入血液系統。懸浮微粒的暴露量對於死亡率及發病率有密切的關係，並不會因為濃度低而不造成人體健康的影響，而是透過累積的方式持續地進行(World Health Organiztion, 2018)。

doi:10.6342/NTU201903250

Lelieveld et al. (2015)透過大氣化學模型評估 PM2.5 對於各區域死亡率的影響，研究指出在 2010 年約有 330 萬人因為戶外的空氣污染而導致過早死亡；世界衛生組織也指出 2016 年約有 420 萬人因為戶外空氣污染而導致過早死亡。世界衛生組織於 2005 年提出的空氣品質準則(Air Quality Guidelines, AQGs)，對於 PM2.5 的年平均及 24 小時平均濃度，分別訂下 10 $\mu g/m^3$及 25 $\mu g/m^3$的目標，世界上許多開發中國家的 PM2.5 年平均濃度，降低約 35$\mu g/m^3$，將能符合 WHO 所提出的標準，將可以降低 15%的死亡率，但目前世界上約有 91%的人暴露在年平均超過 10 $\mu g/m^3$ 的 PM2.5 濃度下，而台灣希望於 2020 年達成 PM2.5 全國年平均目標值 15$\mu g/m^3$。

由台灣行政院環保署的清淨空氣行動計畫中指出，國內的污染源對於細懸浮微粒的年平均濃度影響比率為 56.7%，其中包含了移動式污染源的 37%、工業污染源 31%、其他固定源 43%(如圖 2.1)(行政院環境保護署, 2015)，而截至 2019 年 3 月，根據交通部統計查詢網[1]的資料顯示，全台的機動車數量約有 2190 萬輛，汽車數量約有 800 萬輛，政府近年也透過政策補助民眾淘汰二行程機車及老舊的大型柴油車，以及推廣大眾運輸工具的使用，欲減少移動式污染源的比例。而在工業污染源則針對電力設施、鍋爐、餐飲油煙、河川揚塵進行管制；其他固定源則是以農業廢棄物燃燒管制、改善風俗習慣以及營建及堆置揚塵管制為主。

[1] 交通部統計查詢網 https://stat.motc.gov.tw/mocdb/stmain.jsp?sys=100&funid=a3301

doi:10.6342/NTU201903250

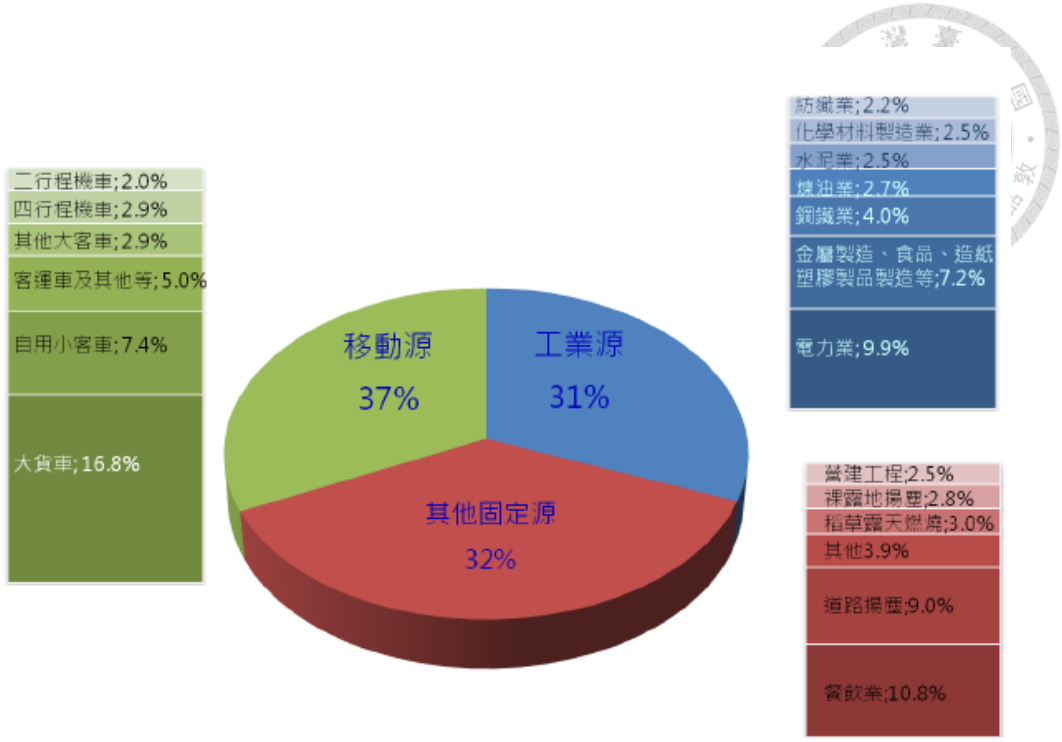


圖 2.1 國內各類污染源對於細懸浮微粒(PM2.5)濃度影響比率
(行政院環境保護署, 2015)

2.2 空氣品質資料蒐集與應用

本節會探討目前空氣品質資料的來源以及資料處理的方式，並了解目前應用上是如何將資料進行呈現與視覺化，使監測數據轉換成容易理解的空氣品質資訊給民眾，使其進行防護等措施以降低空氣污染暴露。

2.2.1 空氣品質資料來源

目前台灣對於空氣品質的監測，除了行政院環保署的官方測站之外，近年來民眾對於空氣污染意識提高，由民間發起的環境感測計畫，透過微型感測器進行空氣品質監測也持續的發展中，此章節將對於官方及民間的空氣品質監測數據部分進行比較，探討數據之間的特性與差異。

(1) 政府測站

行政院環境保護署空氣品質監測網，類型包含一般、工業、交通、國家公園、背景及其他特殊目的測站，站址選定的原則及採樣口設置原則都

doi:10.6342/NTU201903250

有明確的規範[2] (如下表 2.1)，透過監測空氣污染物的變化，評估對於民眾及自然環境的危害，以及作為法規及管制上的依據，並發展空氣品質擴散模式。

表 2.1 站址選定及採樣口設置原則

站址選定原則	採樣口設置原則
• 測站種類 • 污染源分布、類型及污染物濃度分布 • 地形、地勢及氣象條件 • 人口分布及交通狀況 • 有益於防制對策效果之判定 • 都市計畫、區域計畫或其他土地利用計畫	• 不直接受煙道及排氣口等污染影響之處所 • 避免採樣口附近障礙物對氣流及污染物濃度之干擾 • 避免採樣口附近建築物或障礙物表面對污染物濃度之影響 • 依監測站附近污染物垂直方向濃度分布情形，決定採樣口離地面高度

環保署也提供空氣污染指標(Air Quality Index, AQI)，給予空氣品質對於健康影響與活動上的建議，但該指標為每日及每小時的數值進行發布，且該數值經過公式的計算，為長時間的平均值，較無法反映出當下時刻的空氣品質狀況，另外因為測站數量較少，對於小區域的空氣品質變化也較難掌握，以臺北市為例，環保署測站僅有 7 個，分別為士林、大同、中山、古亭、松山、陽明及萬華測站(如圖 2.2)，民眾於日常生活中將無法真實的獲得自身周遭的空氣品質數據，以進行相對應的活動及防護措施。

[2]站址選定及採樣口設置原則：https://taqm.epa.gov.tw/taqm/tw/b0101.aspx

doi:10.6342/NTU201903250

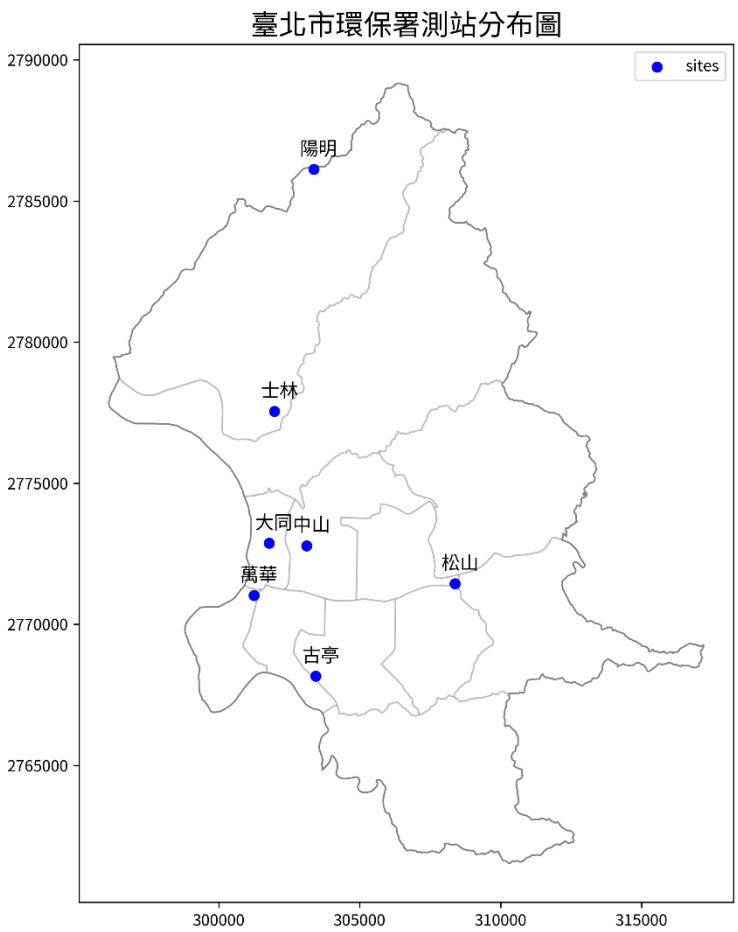


圖 2.2 臺北市環保署測站分布圖

(2) 微型感測器

近年民眾對於空氣品質的意識抬頭，逐漸了解到空氣污染對於人體健康的影響，並且因為物聯網技術的普及，在於環境變化的即時感測也更加容易，包含空氣品質、水質、土石流……等等許多的監測應用。如在政府智慧城鄉的計畫中，預計在 2017 至 2020 年間，布建 1 萬 200 點空氣品質感測器，蒐集大量且時間解析度密集的空氣品質數據，配合相關的工廠資料，來進行污染區的鑑定，以及追蹤污染排的來源。而由民間發起的 LASS(Location Aware Sensor System)是一套「環境感測器網路系統」，以開源及公益的定位發展於 2015 年，透過微感測器進行空氣品質的監測，至 2018 年 8 月止，包含空氣盒子(Airbox)等其他專案，台灣已架設超過 3600

doi:10.6342/NTU201903250

台空氣品質感測器，並於每五分鐘回傳一次數據回伺服器，資料以開源的方式進行發布(LASS-Community, 2018)。

微感測器基於成本低且容易架設的優勢，數量持續的增加，相對於環保署測站於空間上更能反映小區域的空氣品質，且感測頻率較高，可對於臨時性的空氣污染事件進行反映，但由於為了減少感測器體積以及成本，微感測器對於粒徑、溫溼度等干擾並未納入考量，因此測值容易出現誤差。雖然微感測器的數據品質尚無法達到標準測站的規範，不過兩者對於區域空氣品質的污染濃度具有相同向量的趨勢反應(行政院環境保護署, 2018b)，Chen et al. (2017)也針對微感測器的狀態、感測之數據進行可性度評比，有效的推斷異常事件，這將有助於檢視資料的正確性。

2.2.2 空氣品質資料處理方式

由於空氣品質監測站數量於空間上的限制，我們無法得知空間上所有區域的空氣品質狀況。Wong et al. (2004)比較了在計算空氣品質時所使用的空間統計方法，包含空間平均(spatial averaging)、最鄰近法(nearest neighbor)、反距離權重(inverse distance weighting)及克利金(kriging)。空間平均是指將某一區域內的監測資料進行平均計算，作為該區域的空氣品質數值；最鄰近法則是指定最靠近該區域中心的測站數值，作為區域的空氣品質數值；反距離權重法則以測站與其他測站之間距離的倒數作為權重，透過搜尋半徑(search radius)限制測站的影響程度，計算出研究區域的空氣品質；克利金法考量了測站間於空間上的相關程度，以給予不同的權重進行推估。

2.2.3 空氣品質資訊呈現與視覺化

於生活中面臨空氣污染問題的民眾，若是直接提供空氣品質的監測數值給

doi:10.6342/NTU201903250

予民眾，在沒有相關的專業知識背景支持下，將較難理解目前的空氣品質狀況，因此如何將資料進行呈現以及視覺化，傳遞空氣品質的資訊給與民眾，達到減少空氣污染暴露的目的將是值得探討的問題。透過調查目前現有服務的狀況，也可作為本研究決策支援系統建置上的參考。

以行政院環境保護署空氣品質監測網[3]對於空氣品質的呈現，透過地圖的方式進行展示(如圖 2.3)，以不同的圖例描述目前空氣品質的狀況，讓民眾可以透過地圖進行互動，選擇特定測站以了解該測站的空氣品質狀況，此外也提供曲線圖，來展示空氣污染數值的變化趨勢(如圖 2.4)。

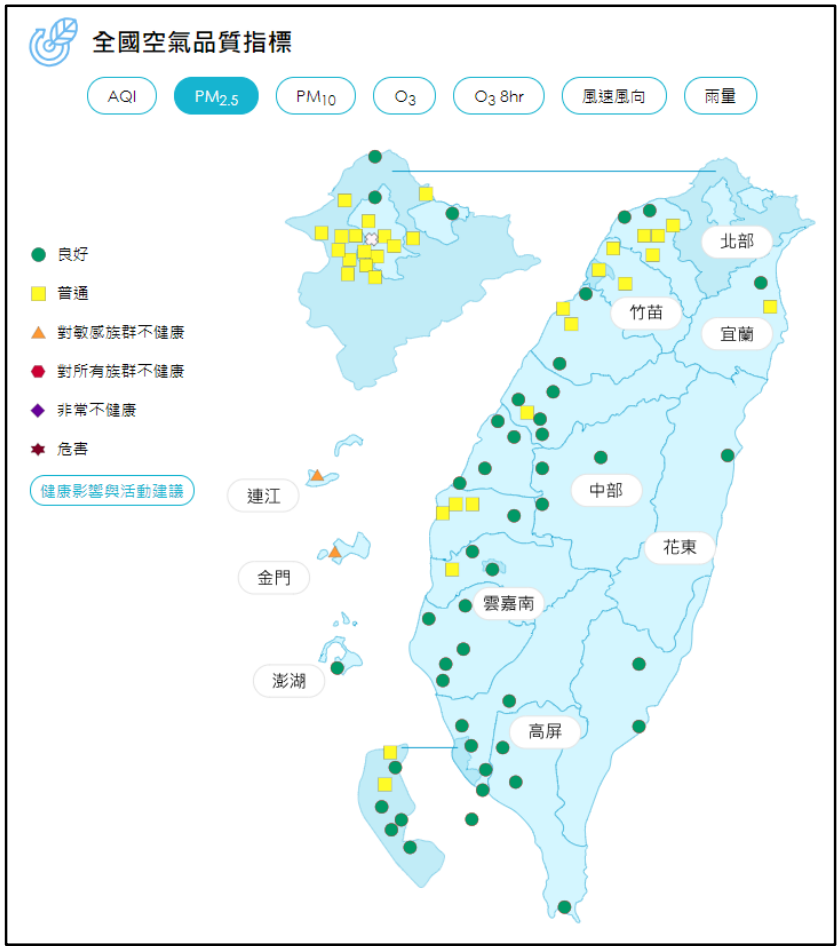


圖 2.3 全國空氣品質指標地圖展示介面

[3] 行政院環境保護署 - 空氣品質監測網：https://airtw.epa.gov.tw/default.aspx

doi:10.6342/NTU201903250

圖 2.4 各空氣污染物變化曲線圖

環境即時通(3.0)，是由行政院環保署提供的 APP 服務，提供使用者適地性服務(Location-based service, LBS)以及常用位置的設定，讓民眾獲得即時的環境資訊與氣象，針對各種環境指標提供給一般民眾和敏感族群在於外出上的建議，並可以透過推播的方式提出警示提醒民眾(行政院環境保護署, 2018c)。在於資訊展示的部分，除了透過儀表板呈現最新訊息及活動建議外(如圖 2.5)，並以地圖(如圖 2.6)與測站列表(如圖 2.7)的方式，提供民眾進行查詢各空氣品質測站的資訊，其中地圖測站的部分，透過空氣品質指標(Air Quality Index, AQI)或是各空氣污染物的副指標將測站進行分級展示(如圖 2.8)，以視覺化的方式，讓民眾更輕易的了解空氣品質狀況。

doi:10.6342/NTU201903250

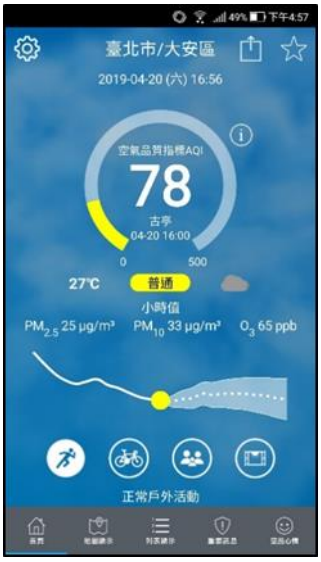


圖 2.5 環境即時通儀表板

圖 2.6 環境即時通地圖

圖 2.7 環境即時通列表

空氣品質指標(AQI)							
AQI指標	O_3 (ppm) 8小時平均值	O_3 (ppm) 小時平均值(1)	$PM_{2.5}$ ($\mu g/m^3$) 24小時平均值	PM_{10} ($\mu g/m^3$) 24小時平均值	CO (ppm) 8小時平均值	SO_2 (ppb) 小時平均值	NO_2 (ppb) 小時平均值
良好 0～50	0.000 - 0.054	-	0.0 - 15.4	0 - 54	0 - 4.4	0 - 35	0 - 53
普通 51～100	0.055 - 0.070	-	15.5 - 35.4	55 - 125	4.5 - 9.4	36 - 75	54 - 100
對敏感族群不健康 101～150	0.071 - 0.085	0.125 - 0.164	35.5 - 54.4	126 - 254	9.5 - 12.4	76 - 185	101 - 360
對所有族群不健康 151～200	0.086 - 0.105	0.165 - 0.204	54.5 - 150.4	255 - 354	12.5 - 15.4	186 - 304(3)	361 - 649
非常不健康 201～300	0.106 - 0.200	0.205 - 0.404	150.5 - 250.4	355 - 424	15.5 - 30.4	305 - 604(3)	650 - 1249
危害 301～400	(2)	0.405 - 0.504	250.5 - 350.4	425 - 504	30.5 - 40.4	605 - 804(3)	1250 - 1649
危害 401～500	(2)	0.505 - 0.604	350.5 - 500.4	505 - 604	40.5 - 50.4	805 - 1004(3)	1650 - 2049

圖 2.8 空氣品質指標

而空氣品質指標各個分級將分別對應到健康影響(如圖 2.9)以及活動上的建議(如圖 2.10)，這將能給予民眾在於戶外活動上的一些參考，未來將能依照使用者各個人資訊，如年齡、健康狀況……等等訊息，提供個人化的服務給予民眾。

doi:10.6342/NTU201903250

空氣品質指標 (AQI)	0～50	51～100	101～150	151～200	201～300	301～500
對健康影響與活動建議	良好 Good	普通 Moderate	對敏感族群不健康 Unhealthy for Sensitive Groups	對所有族群不健康 Unhealthy	非常不健康 Very Unhealthy	危害 Hazardous
狀態色塊	綠	黃	橘	紅	紫	褐紅
人體健康影響	空氣品質為良好，污染程度低或無污染。	空氣品質普通；但對非常少數之極敏感族群產生輕微影響。	空氣污染物可能會對敏感族群的健康造成影響，但是對一般大眾的影響不明顯。	對所有人的健康開始產生影響，對於敏感族群可能產生較嚴重的健康影響。	健康警報：所有人都可能產生較嚴重的健康影響。	健康威脅達到緊急，所有人都可能受到影響。

圖 2.9 空氣品質指標(AQI)與健康影響

空氣品質指標 (AQI)	0～50	51～100	101～150	151～200	201～300	301～500
對健康影響與活動建議	良好 Good	普通 Moderate	對敏感族群不健康 Unhealthy for Sensitive Groups	對所有族群不健康 Unhealthy	非常不健康 Very Unhealthy	危害 Hazardous
狀態色塊	綠	黃	橘	紅	紫	褐紅
一般民眾活動建議	正常戶外活動。	正常戶外活動。	1.一般民眾如果有不適，如眼痛，咳嗽或喉嚨痛等，**應該考慮減少**戶外活動。 2.學生**仍可進行**戶外活動，但**建議減少**長時間劇烈運動。	1.一般民眾如果有不適，如眼痛，咳嗽或喉嚨痛等，**應減少**體力消耗，特別是減少戶外活動。 2.學生**應避免**長時間劇烈運動，進行其他戶外活動時**應增加休息**時間。	1.一般民眾**應減少**戶外活動。 2.學生**應立即停止戶外活動**，並將課程調整於室內進行。	1.一般民眾**應避免**戶外活動，室內應緊閉門窗，必要外出應配戴口罩等防護用具。 2.學生應立即**停止戶外活動**，並將課程調整於室內進行。
敏感性族群活動建議	正常戶外活動。	**極特殊敏感**族群建議注意可能產生的咳嗽或呼吸急促症狀，但仍可正常戶外活動。	1.有心臟、呼吸道及心血管疾病患者、孩童及老年人，**建議減少**體力消耗活動及戶外活動，必要外出應配戴口罩。 2.具有氣喘的人可能需增加使用吸入劑的頻率。	1.有心臟、呼吸道及心血管疾病患者、孩童及老年人，**建議留在室內並減少**體力消耗活動，必要外出應配戴口罩。 2.具有氣喘的人可能需增加使用吸入劑的頻率。	1.有心臟、呼吸道及心血管疾病患者、孩童及老年人**應留在室內並減少**體力消耗活動，必要外出應配戴口罩。 2.具有氣喘的人應增加使用吸入劑的頻率。	1.有心臟、呼吸道及心血管疾病患者、孩童及老年人**應留在室內並避免**體力消耗活動，必要外出應配戴口罩。 2.具有氣喘的人應增加使用吸入劑的頻率。

圖 2.10 空氣品質指標(AQI)與活動建議

其他網站如 g0v 零時空汙觀測網[4]、EDiGreen 空氣盒子[5]……等等，在於空氣品質數據的展示，均是透過 WebGIS 的技術，將資料進行地圖化的展示，並依據空氣品質的分級進行資料的視覺化，協助資訊的傳遞。

透過本結的文獻回顧，可以了解到目前空氣品質數據存在的差異，雖然微感測器的監測數值準確度仍待商榷，但整體的數值是與國家監測站的空氣品質監測數據具有一定的變化趨勢，而在數量及監測頻率上，微型感測器的數據則具有相對的優勢，能夠快速的反應出小區域的空氣品質變化，能更即時的提供民眾活動上的建議。

微型感測器的設置，提升了空間上的監測密度，但仍並非所有區域都具有

[4] g0v 零時空汙觀測網：https://v5.airmap.g0v.tw/#/map
[5] EdiGreen 空氣盒子：https://airbox.edimaxcloud.com/

doi:10.6342/NTU201903250

空氣品質感測器，需要透過空間推估的方式，計算未設站區域的空氣品質狀況，空氣污染會受到許多環境因子的影響，基於資料蒐集上的限制，我們將難以相當準確的推算出每個未設站區域的空氣品質狀況。

在目前的應用上，多數是透過 WebGIS 技術進行地圖視覺化，提供民眾空氣品質的資訊，也基於使用者訂閱的測站或是 LBS 的方式透過推播傳遞資訊給使用者，以避開空氣污染或是進行相關防護。目前使用者雖能透過地圖了解到該區域的空氣品質狀況，但在日常生活中空間的移動上，協助民眾挑選一條能避開高空氣污染區域的路線，是目前應用上仍缺乏的。

2.3 空氣污染資訊於路徑選擇上的應用

本節將探討在考量空氣污染資訊下進行路線的選擇，以降低空氣污染暴露的影響，將分別對於空氣污染暴露量的計算，以及如何在考量空氣污染暴露下進行路線規劃進行回顧。

2.3.1 空氣污染暴露量

由於空氣污染暴露將造成死亡率以及發病率的提升，且影響為累積的方式，並不會因為空氣品質數值低就不造成健康上的危害，因此除了減少空氣污染的排放之外，如何有效的避免空氣污染暴露，將能有效降低受到空氣污染對於健康的危害。

在空氣污染暴露的評估上或預警上，過去許多研究或應用忽略了人類於空間上的活動，導致評估的結果或預警應用上無法與真實狀況符合。如 Park and Kwan (2017)針對空氣污染暴露進行評估，並考量是否移動及不同時間尺度下的空氣污染數值所產生的四種組合，模擬 80 筆的一日所承受的空氣污染暴

doi:10.6342/NTU201903250

露，指出若忽略人類活動的時空變化，所計算的評估數值將可能有錯誤存在，因為人們不會受到長時間平均的空氣污染數值所影響，較短的監測數值，才能更準確地反映出民眾在不同時間段受到空氣汙染的影響，且由於民眾於日常生活中將具有移動的行為，應將其移動因素納入考量，以進行空氣污染暴露量的評估(如圖 2.11)。

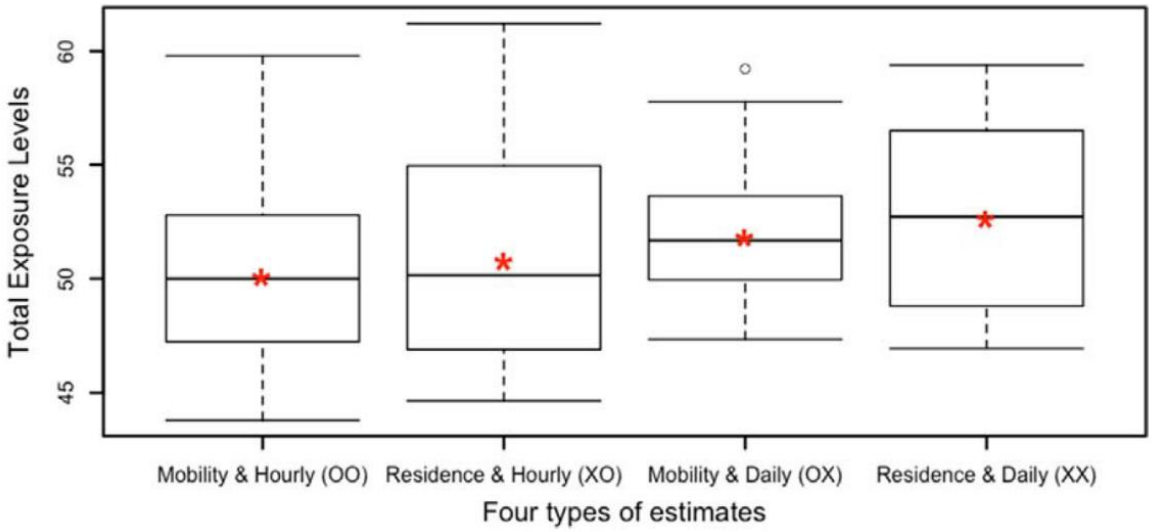


圖 2.11 移動與否及每小時/每日平均空氣污染數值的四種組合
(Park & Kwan, 2017)

Liu et al. (2015)針對臺北市 120 個 19-24 歲的受試者進行通勤方式的研究，分別比較捷運、公車、汽車及步行等不同通勤方式對於暴露量的影響，pm2.5 的平均暴露量分別約為 22.3、32.2、29.2 及 42.1μg/m^3，其中又以步行的通勤方式在此四種通勤方式中空氣污染暴露量最高；行政院環保署於 2017 年於台北都會區進行民眾通勤的空氣污染暴露量程度調查，以平均通勤距離約 10.8 公里進行 PM2.5 的暴露量監測，分別對於汽車、捷運、公車及摩托車進行調查，調查結果 PM2.5 濃度分別為 7.6、21.9、23.5 及 32.1μg/m^3，其中以騎乘摩托車的空氣污染暴露量最高，由於與機動車輛的距離較近，直接受到排氣的影響(行政院環境保護署空保處, 2018)，根據交通部統計查詢網的統計數據可

doi:10.6342/NTU201903250

知，目前台灣每一百人約有 92 人均有摩托車，對於這些通勤的民眾，將直接面臨空氣污染的危害。Ham et al. (2017)針對了六種不同的交通模式進行暴露量的評估，研究成果發現自行車的路線選擇，在遠離交通密集的自行車專用道上空氣污染暴露量最低，將能降低 15%-75%的空氣污染暴露量。因此日常生活中除了民眾鼓勵搭乘大眾運輸工具及自身攜帶口罩防護之外，遠離通勤尖峰路段將會是另一個降低空氣污染暴露的選擇。

2.3.2 空氣污染暴露量下的路線選擇

在於路線選擇方面，Li et al. (2017)透過 dijkstra 演算法進行最低空氣污染暴露量路線的計算，比較了日常行經路線與最低空氣污染暴露路線的差異(如圖 2.12)，考量進行不同移動方式時的呼吸量，依空間推估所獲得的空氣污染數值，計算通勤路線的空氣汙染總暴露量，公式如下：

$$\text{Exposure} = \sum_{i=1}^{n} C_i \times \frac{T}{n} \times \text{R} = C_T \times \frac{T}{n} \times R$$

參數說明：

C_i：某條線段空氣污染濃度

C_T：該路程總空氣污染濃度

n：線段數量

T：通勤時間

R：呼吸比率

doi:10.6342/NTU201903250

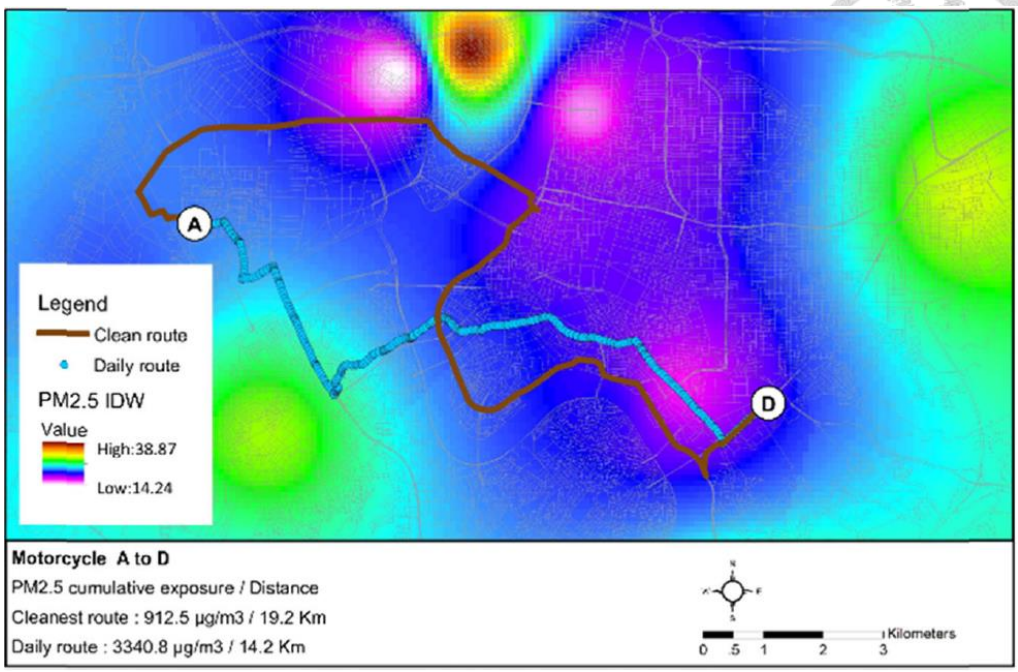


圖 2.12 最低暴露量路線與日常路線比較(Li et al., 2017)

Jarjour et al. (2013)則比較不同交通流量的道路所產生的空氣污染，給予自行車路線選擇上的建議；。但上述對於空氣污染暴露路線選擇的研究，都未考量選擇低暴露量路線時，所額外產生的移動成本，這在實際應用中將無法給予使用者太大的幫助，同時考量空氣污染暴露量以及路線距離，才符合民眾於實際生活中的需求。

Anowar et al. (2017)透過網路問卷的方式，針對 695 名騎車自行車的民眾，在於路線的選擇上，如何對於空氣污染、道路類型、腳踏車基礎設施和旅程時間等因子進行抉擇，研究的統計發現，約有 12%的人對於空氣污染暴露並不在乎，而在旅程時間與空氣污染的抉擇下，受試者願意在旅程時間不超過 4 分鐘的狀況下，選擇一條二氧化氮濃度降低 5ppb 的路線。此研究顯示出，民眾對於路線的選擇上，將不會全然的選擇最低暴露量路線，甚至有時仍會忽略空氣污染的影響。

doi:10.6342/NTU201903250

基於上述所提到的問題，Homayoon et al. (2015)則將距離與空氣污染暴露量給予權重後，生成一個既考量距離及空氣污染暴露的數值作為路徑分析的成本(如圖 2.13)，圖中的三條路線由左到右分別為：最低暴露量路線、基於兩者權重後的路線以及最短路線，因此該研究不局限於最短路徑和最低暴露量路徑的比較，而多產生了一條考量兩個目標下所找出的路線。此方式將提供使用者依照個人偏好調整距離與空氣污染暴露量的權重，以影響路徑分析的結果。

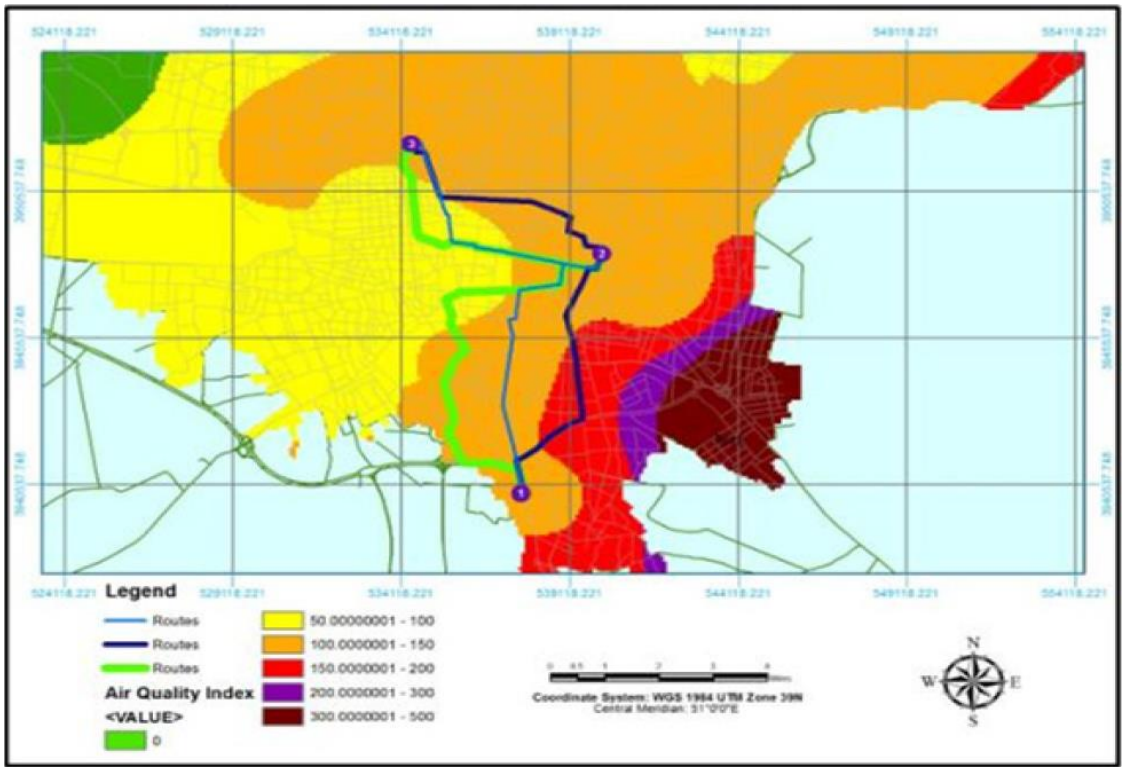


圖 2.13 考量距離與空氣污染暴露量的路線規劃結果(Homayoon et al., 2015)

Vamshi and Prasad (2018)則提出了一個最小空氣污染暴露路線選擇的系統框架，基於線上開放資料集的路網圖層作為框架設計的資料來源，並依照距離、路線類型及路網節點密度計算後作為路徑分析成本，以 Yen's K-shortest path 演算法進行路徑分析，同時考量距離以及其他影響因子，使路線於實際應用上更具選擇價值。

透過權重計算的方式，基於空氣污染暴露量與距離的成本下找出一條適合

doi:10.6342/NTU201903250

的路線，在上述研究中雖已獲得一定的成果，但此研究與多數研究在進行路線規劃時，均基於路網圖資所進行，在計算空氣污染暴露量上，所需要的行經時間仍是需要準備的資料，這在實際應用上若想要擴充應用範圍時，資料的維護以及更新上將是一大問題。

透過文獻得知，空氣污染暴露的計算上，若是忽略人類活動及測站在空間及時間上的變化，將可能導致計算結果產生偏差或錯誤，而在於路線選擇上，如何考量距離及空氣污染暴露量，提供一條滿足使用者偏好的路線，而非追求低空氣污染暴露，導致增加移動上的成本，才將是符合民眾的需求。而過去研究上所進行的路線規劃，多是藉由自行準備路網圖資，進行最短路徑演算法進行計算，除了資料維護與更新的問題外，這些路線規畫產生的結果將可能大幅度的變動使用者原有熟悉的路線，導致使用者的不便或是提升其他潛在的危險。

2.4 文獻評析

透過文獻回顧，了解到空氣污染對於人體健康的影響，將是一種持續性的傷害，因此我們需要盡量減少空氣污染暴露量，而近年來物聯網的普及，環境微感測器的架設使得我們能獲得小區域且更即時的空氣品質變化趨勢，更能貼近民眾生活中所接觸到的空氣品質狀況。而在空氣污染暴露量評估的研究中，對於考量空氣污染暴露量的路線選擇方案，多數是提供最低暴露量路線，而忽略了移動上距離或時間的成本，且可能大幅度的變動原有使用者熟悉的路線，導致其他潛在的危險性提高。而過去的研究成果於實際的應用上，由於路線規劃需基於路網圖層進行，且於計算上可能需要道路的即時資訊，皆將面臨資料維護及更新上的問題。

doi:10.6342/NTU201903250

故本研究將透過微感測器所監測的空氣品質數據作為輸入，利用此數據在時間及空間上的優勢，來作為路線選擇上對於空氣污染暴露量的參考，嘗試使用線上地圖服務來進行路線規劃，改善路網資料及相關資訊蒐集及維護的困難，並以 WebGIS 的方式提供互動，協助使用者在考量空氣污染暴露量與移動成本下，基於原有熟悉路線進行路線修正，給予較佳的路線選擇。

doi:10.6342/NTU201903250

第三章　研究方法

3.1 研究流程

本研究想在考量空污染暴露量與移動成本下進行路線選擇，提供民眾路線選擇上的建議。研究內容主要有三個次項問題，首先了解目前空氣汙染的現況以及現有的預警方式是否存在不足的地方；接續對於過去在考慮空氣污染下進行路線規劃的方法進行比較，以及這些研究成果在於實際應用上是否存在問題；最終本研究將整合相關地理資訊技術建立一套資訊化系統，改善過去研究於應用上的問題，並以 WebGIS 的方式與使用者進行互動，協助使用者在可接受的暴露風險及移動成本下，進行路線選擇提供決策上的建議，本研究的研究流程如圖 3.1。

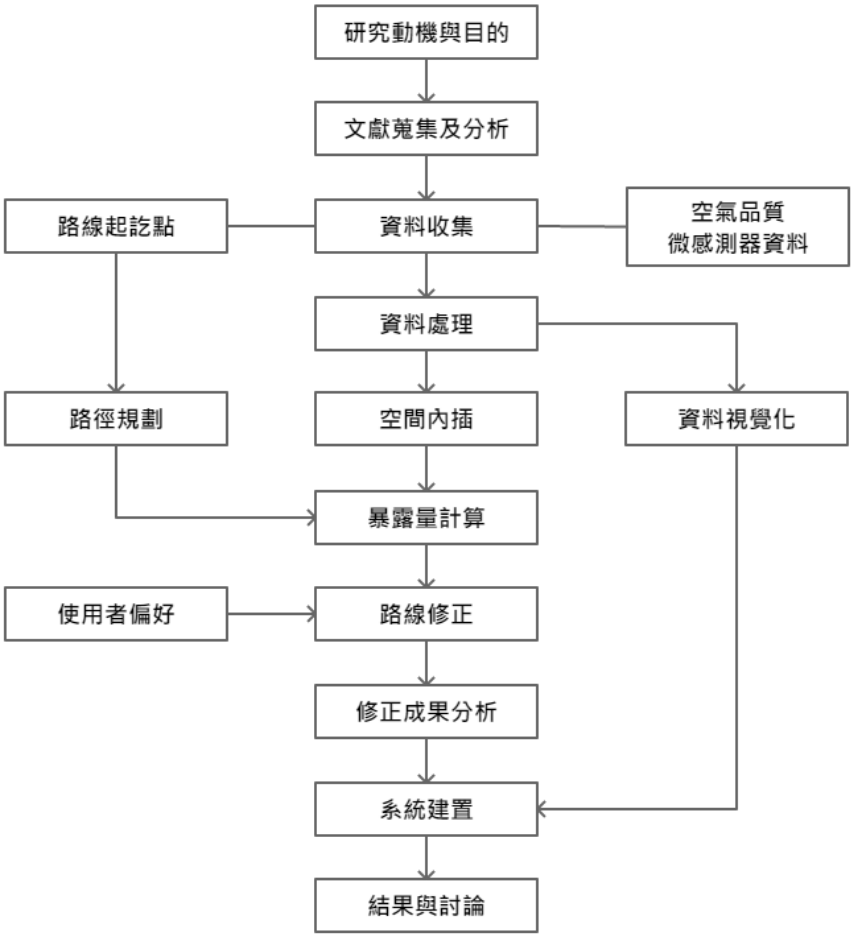


圖 3.1 研究流程圖

doi:10.6342/NTU201903250

3.2 研究範圍

本研究以臺北市作為研究範圍，面積約為 271.8 平方公里，2018 年 7 月的總人口數為 2,674,063 人。依據內政部戶政司[6]以及行政院環保署[7]的統計數據顯示，臺北市在 2018 年 7 月的人口密度及 2018 年 5 月的機動車密度統計上，均為台灣各縣市最高(如圖 3.2 及 3.3)，由於都市區域主要的污染排放是來自於移動是污染源，民眾於都市內的活動將直接的受到空氣污染的影響，因此本研究選定臺北市作為研究區域樣區。

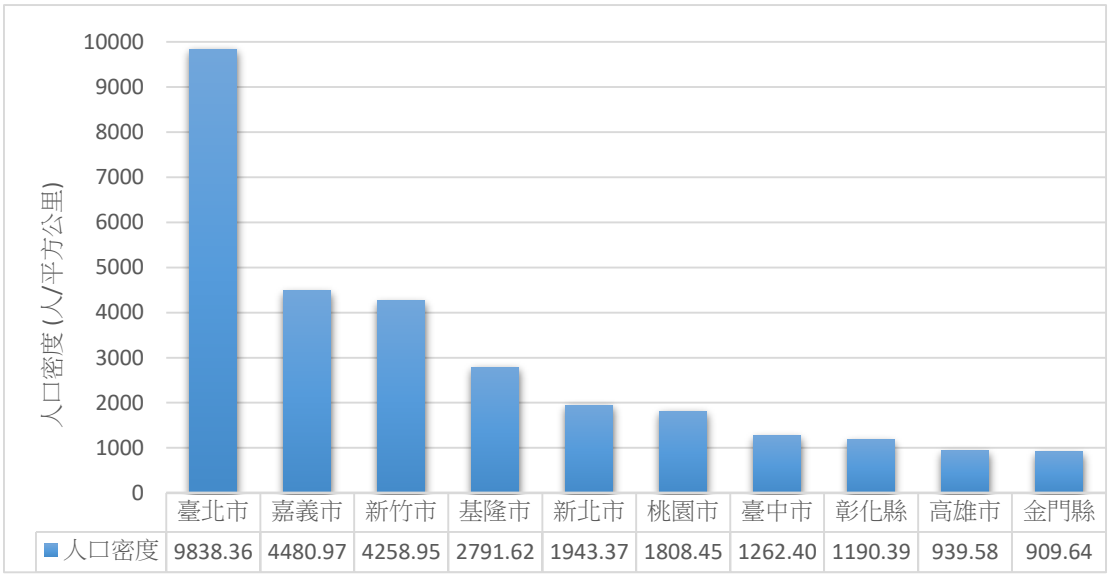


圖 3.2 台灣人口密度十大縣市

[6] 人口統計資料 https://www.ris.gov.tw/346

[7] 機動車登記統計 https://erdb.epa.gov.tw/DataRepository/ReportAndStatistics/StatSceMotors.aspx

doi:10.6342/NTU201903250

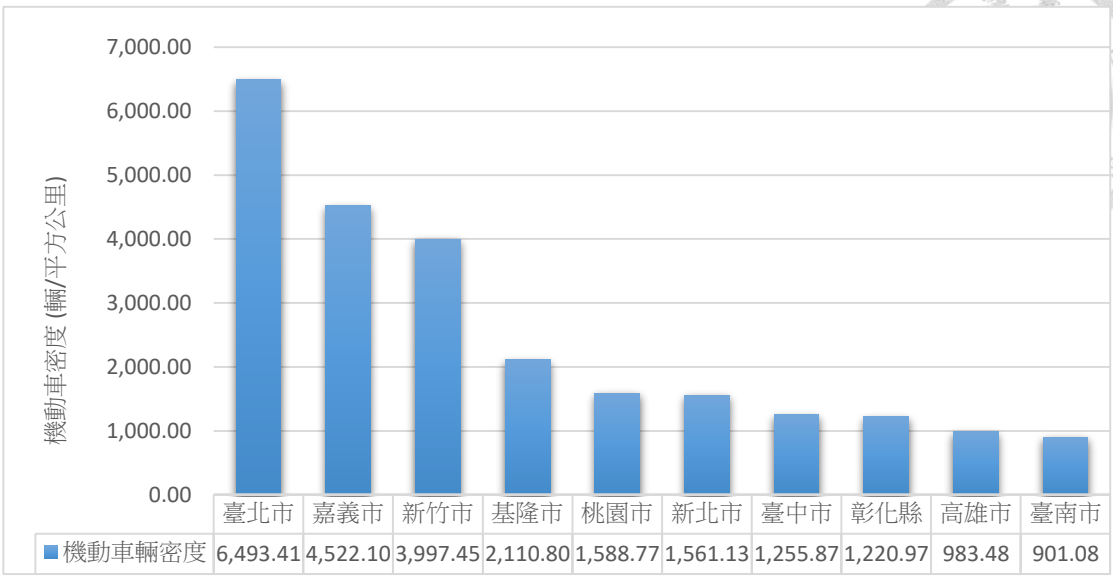


圖 3.3 台灣機動車密度十大縣市

3.3 資料說明

本研究希望計算行經路線的空氣污染暴露量，來提供給民眾於路線選擇上的建議，為了較能即時反應出民眾於不同區域受到的空氣品質狀況，本研究將選用 LASS 微感測器，而並非政府測站的空氣品質監測數值，利用微型測站於空間上的密度及監測頻率高的優勢，希望能更確實的反應出於不同路段上的空氣品質狀況。

資料來源將使用 PM2.5 Open Data 平台所提供的數據，此平台是由中央研究院及 LASS 社群共同維護，介接了透過 IDW 視覺化即時 PM2.5 地圖網站[8]中整理好的數據來源，該數據是由中研院資訊科學研究所網路系統與服務實驗室進行整理，其資料格式以 JSON (JavaScript Object Notation)格式進行提供，資料的更新頻率為每 5 分鐘，資料時區為 UTC+0，資料的欄位主要包含經度、緯度、pm2.5 監測數值、溫度、濕度……等等其他相關資訊。

[8] Realtime PM2.5 map using Inverse Distance Weighting (IDW)：https://pm25.lass-net.org/GIS/IDW/

doi:10.6342/NTU201903250

將透過 Python requests 套件，以 HTTP GET 的方式調用 API 獲取該空氣品質監測資料，並解析 JSON 格式的資料進行使用，資料範例如表 3.1。

表 3.1 微感測器相關數據

OID	gps_lat	gps_lon	pm2.5	temperature	humidity
0	24.765434	120.969095	22.0	29.62	79.0
1	24.635265	120.973405	34.0	29.25	97.0
2	24.629104	121.008379	39.0	27.50	91.0
3	13.168244	100.927360	28.0	33.00	70.0
4	25.143471	121.734014	46.0	29.62	100.0

3.4 分析流程

本研究透過微感測器數據作為空氣品質的參考，希望能即時的反應小區域的空氣品質狀況，在考量空氣污染暴露量及移動成本之下進行路線規劃的修正，來提供民眾在於路線選擇上的建議，故研擬以下分析流程(圖 3.4)，透過開放資料 API 獲取空氣品質微感測器數據，篩選出研究區域範圍內的測站後，進行空間推估以獲得未設站區域的空氣品質狀況數值，後續將針對路線進行暴露量的計算，依照空氣品質指標由空間推估的結果篩選出障礙區，來調整路線規劃的結果，產生一條避開空氣污染程度較高的區域，降低民眾在通勤中空氣污染暴露量。

3.4.1 資料前處理

由開放資料 API 獲取 JSON 格式的微感測器資料，其中包含經緯度、pm2.5 監測數值、溫度及濕度，此步驟依照資料中的經緯度欄位，以 Python Geopandas 套件進行空間資料的產生。

doi:10.6342/NTU201903250

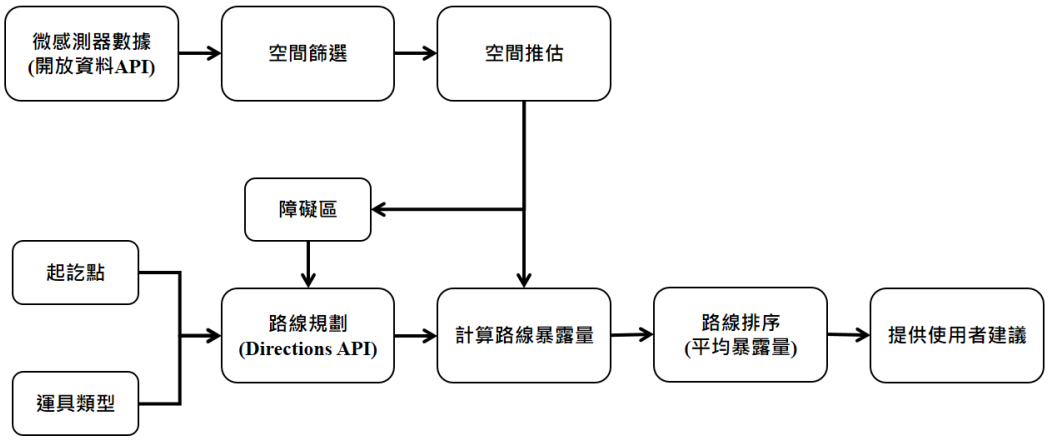


圖 3.4 分析流程圖

3.4.2 空間篩選

由於微感測器數據不僅僅侷限於台灣，還包含了其他 29 個國家在內，因此在本研究的使用上，需篩選出研究範圍台北市內的空氣品質測站。由前述資料說明的部分有提到微感測器資料座標是以經緯度的方式進行提供，而本研究希望後續在於空間分析及處理上計算能以公尺為單位以方便計算路線的距離，故在此進行座標轉換的動作，將座標系統由 WGS84(epsg:4326)轉換為 TWD97(epsg:3826)。

3.4.3 空間推估

由於本研究主要是針對路線選擇方法上進行改進，為提升運算效率，因此在於空間推估方法上不考量相關的環境影響因子進行推估，使用反距離權重法假設空氣污染擴散依距離而遞減。在反距離權重法中，參數使用微感測器中的 pm2.5 監測數值，將台北市範圍內的微感測器數據作為輸入，推估結果將以網格資料的格式輸出，此推估結果會作為後續空氣污染暴露量計算以及路線修正的依據。

doi:10.6342/NTU201903250

3.4.4 路線規劃

在於路線規劃的部分，透過文獻回顧可以得知，過去研究在於路線規劃均是基於自身所收集的路網圖資，在實際應用上，路網資料及道路訊息的維護及更新將是個問題，為了解決資料面所衍伸出資料維護及更新上的問題，路線規劃可由相關的地圖網路服務來進行，如 Google Maps Directions API、Open Route Service、TomTom Routing API、Mapbox Directions API……等等許多的網路服務，藉此資料的維護及更新將由這些大公司及單位進行，應用服務範圍的延伸也將能輕易的變更。

這些路線規劃的 API 提供開發者輸入路線起迄點來獲得一條路線，缺點是無法加入空氣污染濃度作為最短路徑演算法中的線段成本，因此本研究欲改以避開障礙區的方式進行路線的修正，避開空氣污染數值較高的區域，以提供更多的路線選擇給使用者。Google maps 雖然在於空間資訊的內容較為完整，但由官方提供的 API 文件[9]可知，其 Directions API 並未提供障礙區的參數，參考 S. Nedkov (2011)研究中所使用的方式，以加入路線的中途點(waypoints)來影響路線規劃的結果，但其成效不佳，無法正確的決定中途點的所在位置，若是落在非道路上，Google Maps 將自動吸附最鄰近的道路，這將導致路線的結果常出現折返或繞路的狀況，因此無法提供正確的路線規畫結果給使用者。

上述 Google Maps Directions API 參數上的限制，同樣發生於 Mapbox Directions API 及 TomTom Routing API 上，將無法獲得有效的路線提供給民眾使用。而 Open Route Service Directions API 不同的地方在於，它提供了 avoid_polygons 的參數選擇，讓開發者能夠提供障礙區以影響路線規劃的結

[9] Google Web Services Direction API 文件：
https://developers.google.com/maps/documentation/directions/intro

doi:10.6342/NTU201903250

果，在本研究的案例中，障礙區即可以空氣污染數值偏高的區域進行設置，來影響路線規劃的結果，因此本研究選用 Open Route Service Directions API 來進行路線規劃。

而本研究以 Open Route Service 來進行路線規劃的動作，將透過 Python 進行 API 請求，藉此本研究所提出的架構，在於應用上將不需擔心資料維護及更新，且能獲取到相關的道路訊息，以作為後續路線空氣污染暴露量計算的依據，Open Route Service 提供的障礙區參數設置，將符合本研究的需求，以下將對於 Open Route Service Directions API 參數以及資料回傳格式在本研究上使用的部分進行介紹。

(1) API 參數

在本研究中希望給予民眾於生活中的路線選擇提供決策建議，以避開障礙區的方式來降低空氣污染暴露量，並提供民眾進行不同運具的選擇進行路線規劃，透過 Open Route Service Directions API 相關參數來達到這些目的，在本研究中使用到 Directions API 的參數如下表 3.2 所整理。

表 3.2 本研究使用的 Open Route Service Directions API 相關參數說明

參數名稱	參數數值	參數說明
coordinates	((121.xxx, 25.xxx),(121.xxx, 25.xxx))	起點終點
format_out	geojson	回傳格式
profile	driving-car, foot-walking, cycling-road…	運具類型
preference	shortest	偏好最短路線

doi:10.6342/NTU201903250

instructions	True	路線指示
options	avoid_polygons	可加入障礙區
key	5b3c…	API 金鑰

(2) 路線規劃結果的資料格式

本研究將選用 GeoJSON 作為 Open Route Service Directions API 路線規劃的回傳格式，GeoJSON 是一種基於 JSON 的開源標準格式，用來表示各種地理資料，也是一種普遍且通用的地理資料交換格式，將利於模組間資料的整合及傳遞。

由於本研究後續將計算路線的空氣污染暴露量，因此需要了解 Open Route Service Directions API 的回傳結果如何進行道路資訊的儲存。每一個路線規劃結果內容將包含一個 routes 物件，每個 routes 裡可能有一個到多個的 segments 物件，一個 segments 中存在著一到多個的 step 物件，記錄著個別路段的距離、所需時間、道路類型、道路名稱、導航指引及節點資訊。 (圖 3.5)。

doi:10.6342/NTU201903250

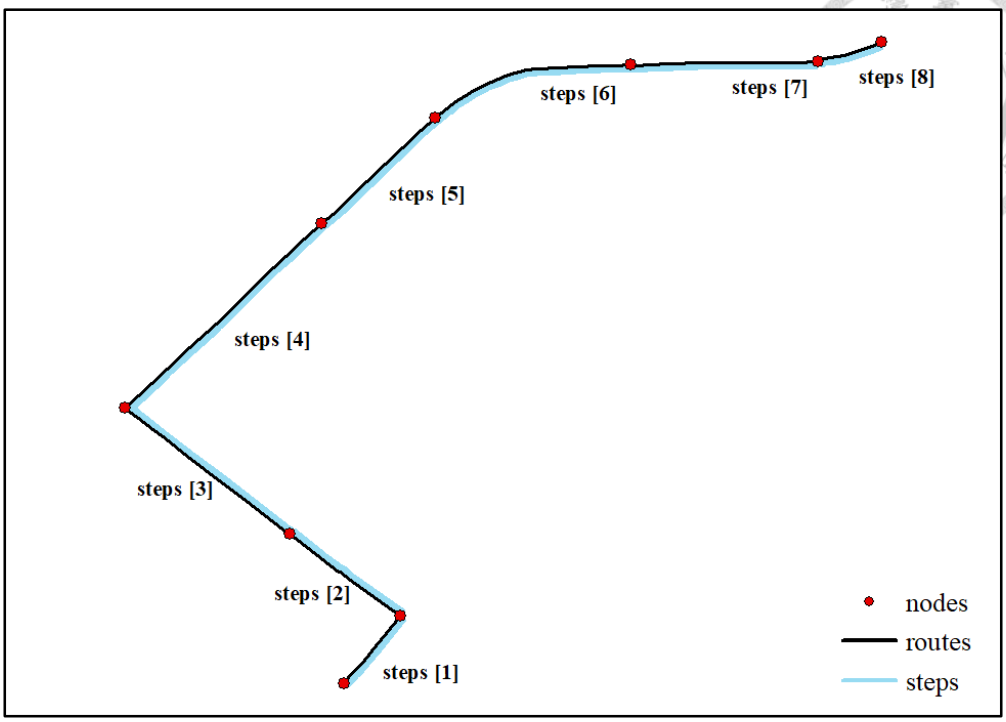
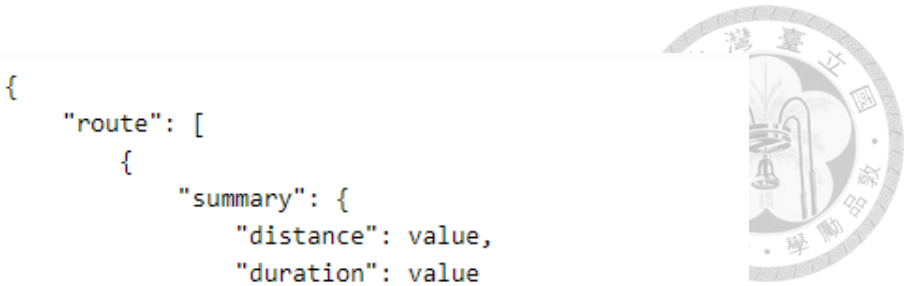


圖 3.5 Open Route Service Directions API 回傳路線範例圖

圖 3.6 為 API 路線規劃的回傳結果，該 GeoJSON 描述著該結果包含了哪些線段(segments)以及個別的座標資訊(geometry)，並有距離(distance)及時間(duration)可提供本研究後續於空氣污染暴露量計算時使用，其中可透過 way_points 抓取出個線段的範圍，以獲取對應的座標資訊。

doi:10.6342/NTU201903250



```
{
    "route": [
        {
            "summary": {
                "distance": value,
                "duration": value
            },
            "segments": [
                {
                    "distance": value,
                    "duration": value,
                    "steps": [
                        "distance": value,
                        "duration": value,
                        "type": value,
                        "instruction": value,
                        "name": value,
                        "way_points": [start, end]
                    ]
                },{
                    ...
                }
            ]
            "bbox": [
                minx, miny, maxx, maxy
            ],
            "geometry": Encoded Polyline,
            "way_points": [...]
        }
    ],
    "bbox":[...]
    "metadata": {...}
}
```

圖 3.6 Open Route Service Directions API 回傳路線 json 結構

3.4.5 路線暴露量計算

此步驟欲將空間推估所獲得的空氣品質數據作為參考，去計算出路線的空氣污染暴露量，以下將對於處理流程進行說明(圖 3.7)。

doi:10.6342/NTU201903250

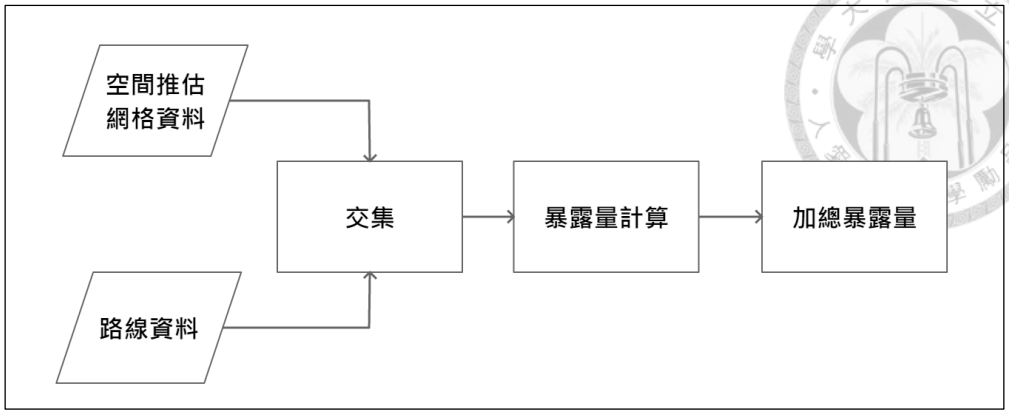


圖 3.7 路線暴露量計算流程圖

(1) 網格資料與路線資料進行交集

由於路段行經的區域可能通過不同的空氣污染數值，因此透過此步驟將行經不同空氣污染數值的路段進行區分，以進行後續的空氣污染暴露量計算。方法則是將微感測器數據透過空間推估所產生的網格資料與 Open Route Service Directions API 路線規劃的結果進行交集(intersect)，以圖 3.5 的路線為例，原有路線中包含了 8 個 steps，經由與網格資料進行交集後，將產生 6 個交集點，將此路線區分為 14 個線段(圖 3.8)，後續將能個別的計算這 14 個線段的空氣污染暴露量。

doi:10.6342/NTU201903250

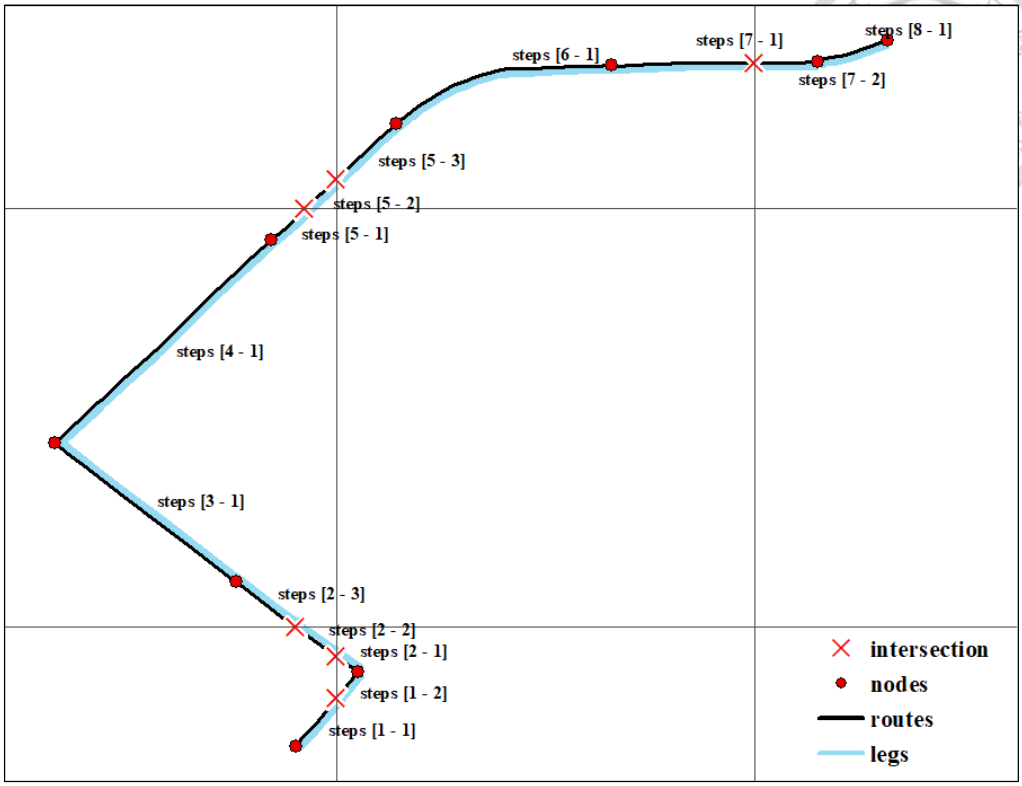


圖 3.8 路線與網格交集圖

(2) 線段暴露量計算

由上一步驟產生了交集後的線段，進一步可以對於每個 steps 所區分區來的線段，依照距離的比例計算出個別線段所需的時間(表 3.3)，計算公式如下：

$$各別線段行經時間 = (線段長度 \div steps\ 總長度) \times steps行經時間$$

表 3.3 路線線段計算

OID	steps	segments	distance	duration	distance (segments)	duration (segments)
1	1	steps [1 - 2]	117	44	41.51	15.61
2	1	steps [1 - 1]	117	44	75.12	28.25

doi:10.6342/NTU201903250

3	2	steps [2 - 1]	181	47	32.22	8.37
4	2	steps [2 - 2]	181	47	59.44	15.44
5	2	steps [2 - 3]	181	47	89.21	23.16
6	3	steps [3 - 1]	273	52	273.24	52.05
7	4	steps [4 - 1]	355	60	354.44	59.91
8	5	steps [5 - 3]	203	28	97.60	13.46
9	5	steps [5 - 1]	203	28	55.16	7.61
10	5	steps [5 - 2]	203	28	50.95	7.03
11	6	steps [6 - 1]	275	44	275.06	44.01
12	7	steps [7 - 1]	246	32	170.71	22.21
13	7	steps [7 - 1]	246	32	75.72	9.85
14	8	steps [8 - 1]	86	18	86.53	18.11

獲得各線段的行經時間後，即可進行各線段的空氣污染暴露量計算，暴露量的計算方式，依照不同的移動模式呼吸量，計算出該移動模式下路線的總空氣污染暴露量，在本研究的案例中，呼吸量將以成人每分鐘平均呼吸量 7 公升帶入計算，計算公式如下：

$$\text{Exposure of steps} = \sum_{i=1}^{N} D_i \times V_i \times R$$

D_i：該線段行經時間(s)

V_i：該線段區域的 pm2.5 數值(μg/m^3)

R：呼吸比率(ℓ/min)

N：steps 與網格交集後線段數量

doi:10.6342/NTU201903250

(3) 加總暴露量

前一步驟已獲得 steps 的空氣污染暴露量，因此最後僅需將所有的 steps 暴露量進行加總，即可獲得 segments 的總暴露量，若 routes 中僅存在一個 segments，即獲得該路線的總暴露量，若該 routes 中存在多個 segments，則加總 segments 的暴露量以獲得該路線的總暴露量。

3.4.6 路線排序及使用者建議

經由 API 回傳的結果，可獲得該路線相關的節點及路線，也包含了行經時間跟距離，前一步驟所獲得路線總暴露量，也將能求出該條路線的平均暴露量，依照暴露量由低至高，給予使用者路線選擇上的建議，也可以依照空氣品質指標提供活動上的參考。

doi:10.6342/NTU201903250

第四章　系統設計

本研究希望透過 WebGIS 的方式與使用者進行互動，搭配響應式設計(Responsive Web Design, RWD)以 WebApp 的方式提供使用，將不會受限於移動裝置的作業系統差異如 Andriod 或 IOS，而導致系統服務出現問題，使用者僅須透過瀏覽器即可獲取服務，提供日常生活中路線上的建議，以地圖視覺化的方式提供使用者更清楚地得知空氣品質狀況及進行路線的選擇。本章節將對於系統設計進行說明。

4.1 系統概述

由於過去研究在於空氣污染暴露路線的建議上，均是基於路網圖資進行路徑分析，實際於應用上將面臨資料維護及更新上的問題，應用範圍的擴充上也受到限制，因此本系統實作將透過 Open Route Service Directions API 進行路線規劃，並獲得道路資訊進行空氣污染暴露計算時的依據，減少應用時資料維護的成本，並且能輕易地擴充應用範圍。

本系統主要目的是提供地圖的互動介面，協助使用者考量空氣污染暴露量及移動成本進行路線上的選擇。前端地圖介面的顯示將使用 Leaflet 來進行開發，Leaflet 是一個開源的 Javascript 庫，適合移動式裝置進行互動地圖的開發；後端資料處理將以 Python 相關套件進行，如 GeoPandas、numpy、pandas……等等，將計算結果以 API 的形式提供給前端使用；空氣品質測站及空間推估的網格資料，將透過 GeoServer 發布 WMTS(Web Map Tile Service)以進行展示；資料儲存方面將選用 MySQL 進行儲存使用者資訊及空間資料，其中空間資料將以 Geometry 的形式儲存。

doi:10.6342/NTU201903250

4.2 系統總體設計

4.2.1 技術選型

(1) 前端地圖展示

本研究將使用 Leaflet 進行開發，由於路線規劃的功能將由 Python 所撰寫的 API 所提供，並以 GeoJSON 的資料型態將路線資料回傳至用戶端，因此前端的地圖框架可以依照需求替換為其他的 JavaScript 庫，如 Google Map、Mapbox、OpenLayers……等等，並不會造成功能上的影響。

(2) 路線規劃

本研究使用 Open Route Service Directions API 進行路線規劃，並獲取相關的道路資訊，提供暴露量計算時所需的時間資訊。

(3) 路線修正

使用 Open Route Service Directions API 中所提供的 Option 參數，將可加入 avoid_polygon 參數設置障礙區域，讓路線規畫能避開指定的地方，於本系統中將使用空間推估的網格結果作為障礙區，可依照空氣品質指標的分級來篩選障礙區域，亦可通過百分位數的方式，來篩選出空氣污染數值偏高的網格區域來作為障礙區的設置。

(4) 資料儲存

透過 MySQL 中的 Geometry 類型進行空間資料的儲存，並使用 GeoJSON 作為進行前後端資料傳遞的格式，以普遍且通用的資料格式進行設計，欲提升系統擴充及相容性。

(5) 圖層發布

doi:10.6342/NTU201903250

以 GeoServer 進行 WMTS 的發布，並透過 Python 調用 GeoServer API 以利於空氣品質圖層的動態更新。

(6) 資料處理與空間分析

由 Python 程式語言進行，搭配相關套件如 GeoPandas、numpy、pandas……等等，降低應用時對於相關軟體的依賴，開發上較有彈性。

4.2.2 系統總體架構

本系統主要提供考量空氣污染暴露量及移動成本進行路線規劃的服務，系統架構主要由資料儲存、路線規劃功能、圖層展示及使用者管理四個面向進行設計。

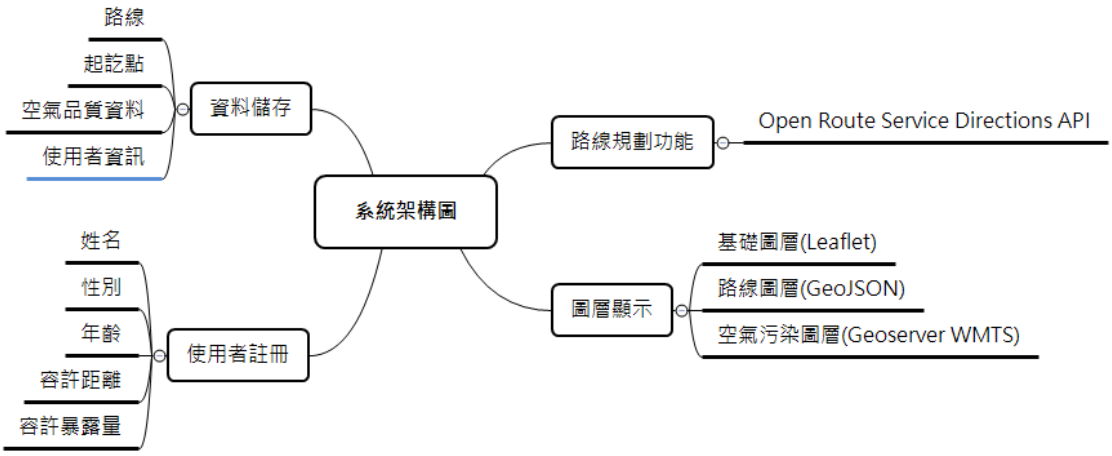


圖 4.1 系統架構圖

4.2.3 系統流程

(1) 使用者註冊與登入

系統將紀錄該使用者的個人資訊以及偏好，以利於路線規劃時自動的帶入相關資訊，提供個人化的服務給與使用者。

doi:10.6342/NTU201903250

(2) 路線規劃請求

接收到客戶端所送出的請求，依照使用者僅需提供的起迄點以及移動模式，系統將會協助進行路線規劃，並計算路線的空氣污染暴露量，不同的移動模式選擇時，將對應到不同的呼吸比率，以計算出相對應的空氣污染暴露量。障礙區選擇的部分，以空氣品質指標中的 PM2.5 副指標為標準，對於空間推估的網格進行篩選，將對所有族群不健康(54.5 – 150.4μg/m^3)、對於敏感族群不健康(35.5 – 54.4μg/m^3)及普通(15.5 – 35.4μg/m^3)三個級別依序作為篩選標準以找出空氣品質較差的區域，後續將這些區域作為障礙區進行路線規劃，系統將依照平均暴露量進行路線的排序，提供前五條路線給予使用者作為參考(如圖 4.2)，系統將依照空氣品質指標對於人體健康影響的程度給予建議，若是空氣品質已達到對所有族群不健康的狀況下，將會建議使用者避免戶外活動或是搭乘大眾運輸工具，以降低空氣污染暴露對於人體健康的危害。

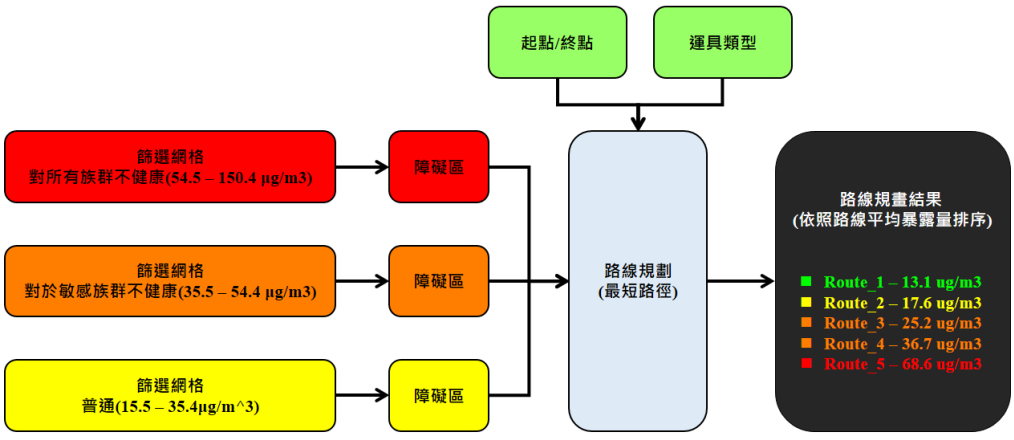


圖 4.2 路線規畫請求執行流程

doi:10.6342/NTU201903250

(3) 圖層顯示

系統所使用到的空間資料包含空氣品質推估結果及路線規劃結果，由於空氣品質資訊對於每位使用者來說並無差異，均是存取同一時間的資料，因此透過 GeoServer 進行 WMTS 的發布，讓每位使用者共同使用；然而路線規劃的結果則依使用者的起迄點與偏好將有所不同，因此由資料庫抓取出該使用者的路線規劃結果，並以 GeoJSON 的方式回傳，由前端進行路線資料的視覺化展示(如圖 4.3)。

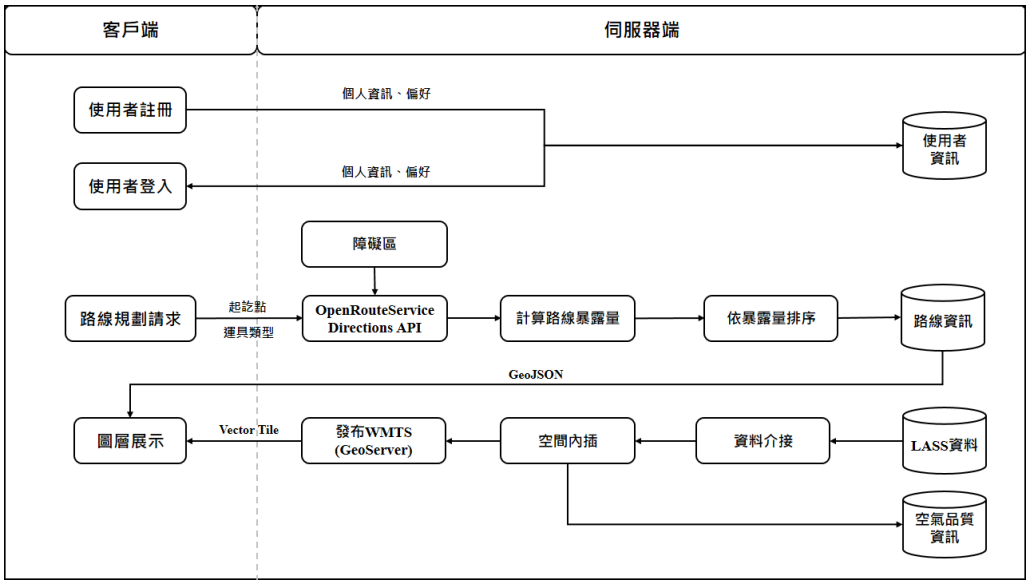


圖 4.3 系統流程圖

4.2.4 系統介面結構

如圖 4.4，介面可分為登入註冊頁面、主功能頁面以及相關功能頁面。登入註冊頁面提供表單輸入的方式進行，註冊建立相關資訊，以提供後續查詢上使用；主功能頁面主要提供路線規劃請求及路線分析比較的功能，為本系統主要目的，讓使用者能得知路線的空氣污染暴露量多寡，提供使用者在於路線選擇上建議；相關功能頁面則提供查詢功能，個人資訊以及過去的路線規劃查詢。

doi:10.6342/NTU201903250

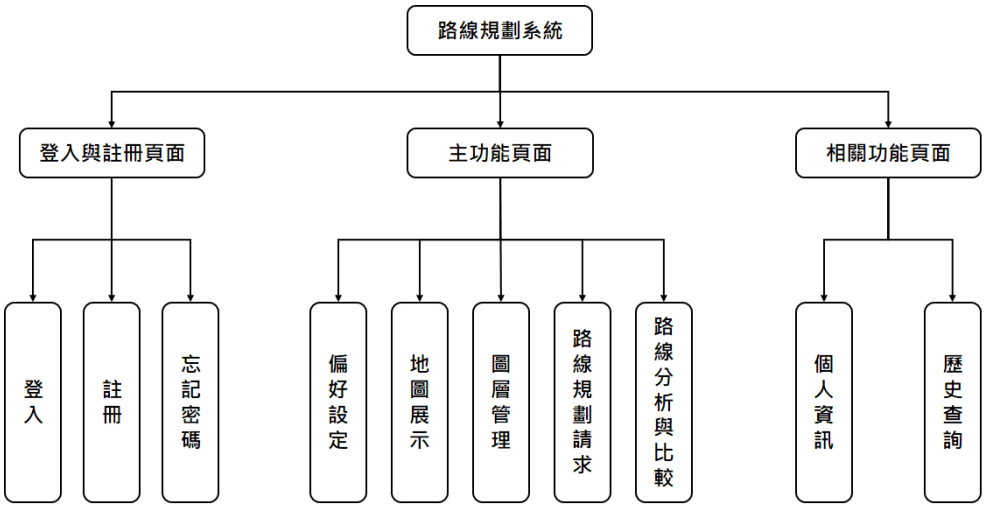


圖 4.4 系統介面結構圖

doi:10.6342/NTU201903250

第五章　成果及討論

本研究欲整合相關地理資訊技術建構一套系統，以協助使用者考量距離與空氣污染暴露量下選擇較佳的移動路線，本章節將對於研究成果，包含系統成果展示、空氣品質資料的獲取、空間推估、路線暴露量計算、路線修正成果及系統建置上進行討論。

5.1 系統成果展示

本研究以 WebGIS 的方式建置決策支援系統，並以響應式設計利於移動式裝置進行使用，民眾可以透過移動式裝置提供路線的起點、終點以及移動模式，此系統將會協助進行路線規劃以及路線的空氣污染暴露量計算，系統將依照路線的平均暴露量來進行結果輸出的排序，連帶提供路線行經時間、距離、總暴露量，並以細懸浮微粒(PM2.5)指標對照表(如圖 5.1)來進行路線的視覺化展示，以利於民眾了解該路線暴露量對於人體健康的危害程度，以及提供相關的建議給予民眾，以作為路線選擇的參考依據(如圖 5.2、5.3)。

doi:10.6342/NTU201903250

指標等級	1	2	3	4	5	6	7	8	9	10
分類	低	低	低	中	中	中	高	高	高	非常高
$PM_{2.5}$ 濃度 (μg/m³)	0-11	12-23	24-35	36-41	42-47	48-53	54-58	59-64	65-70	≧71
一般民眾活動建議	正常戶外活動。			正常戶外活動。			任何人如果有不適，如眼痛，咳嗽或喉嚨痛等，應該考慮減少戶外活動。			任何人如果有不適，如眼痛，咳嗽或喉嚨痛等，應減少體力消耗，特別是減少戶外活動。
敏感性族群活動建議	正常戶外活動。			有心臟、呼吸道及心血管疾病的成人與孩童感受到癥狀時，應考慮減少體力消耗，特別是減少戶外活動。			1．有心臟、呼吸道及心血管疾病的成人與孩童，應減少體力消耗，特別是減少戶外活動。 2．老年人應減少體力消耗。 3．具有氣喘的人可能需增加使用吸入劑的頻率。			1．有心臟、呼吸道及心血管疾病的成人與孩童，以及老年人應避免體力消耗，特別是避免戶外活動。 2．具有氣喘的人可能需增加使用吸入劑的頻率。

圖 5.1 細懸浮微粒指標對照表與活動建議

doi:10.6342/NTU201903250

圖 5.2 系統路線請求使用者介面

圖 5.3 系統路線請求結果展示

5.2 空氣品質資料的獲取

本研究介接 LASS Open API 中的多種微感測器數據，共獲取 177 個微型測站，並透過 Geopandas 進行空間資料的處理，依照經緯度欄位將資料轉為幾何的點座標(Point)，並篩選出台北市範圍內的測站(圖 5.4)，來做為後續的應用。

doi:10.6342/NTU201903250

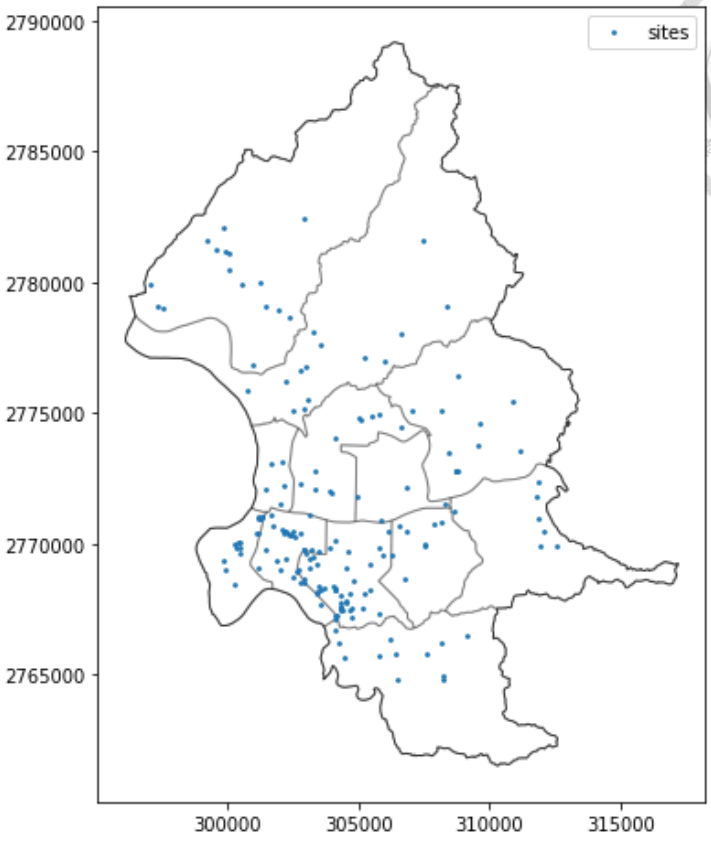


圖 5.4 台北市 LASS 測站分布圖

5.3 空間推估

以台北市範圍內的 LASS 測站，透過 IDW 進行空間推估，並透過行政院環保署所定義的空氣品質指標(圖 5.5)，將 PM2.5 的數值進行分級視覺化(圖 5.6)。網格數值將作為後續空氣污染暴露計算時的依據，其網格也將作為路徑規劃的障礙區使用。

doi:10.6342/NTU201903250

空氣品質指標(AQI)							
AQI指標	O_3 (ppm) 8小時平均值	O_3 (ppm) 小時平均值(1)	$PM_{2.5}$ ($\mu g/m^3$) 24小時平均值	PM_{10} ($\mu g/m^3$) 24小時平均值	CO (ppm) 8小時平均值	SO_2 (ppb) 小時平均值	NO_2 (ppb) 小時平均值
良好 0～50	0.000 - 0.054	-	0.0 - 15.4	0 - 54	0 - 4.4	0 - 35	0 - 53
普通 51～100	0.055 - 0.070	-	15.5 - 35.4	55 - 125	4.5 - 9.4	36 - 75	54 - 100
對敏感族群不健康 101～150	0.071 - 0.085	0.125 - 0.164	35.5 - 54.4	126 - 254	9.5 - 12.4	76 - 185	101 - 360
對所有族群不健康 151～200	0.086 - 0.105	0.165 - 0.204	54.5 - 150.4	255 - 354	12.5 - 15.4	186 - 304(3)	361 - 649
非常不健康 201～300	0.106 - 0.200	0.205 - 0.404	150.5 - 250.4	355 - 424	15.5 - 30.4	305 - 604(3)	650 - 1249
危害 301～400	(2)	0.405 - 0.504	250.5 - 350.4	425 - 504	30.5 - 40.4	605 - 804(3)	1250 - 1649
危害 401～500	(2)	0.505 - 0.604	350.5 - 500.4	505 - 604	40.5 - 50.4	805 - 1004(3)	1650 - 2049

圖 5.5 空氣品質指標(AQI)

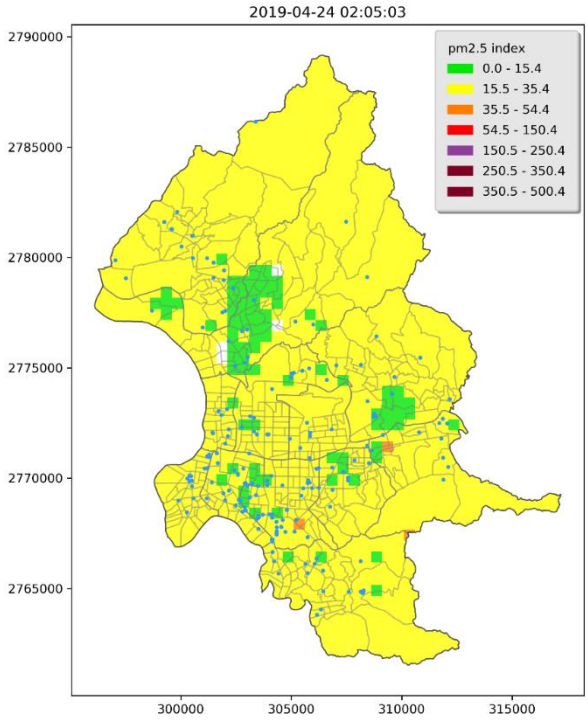


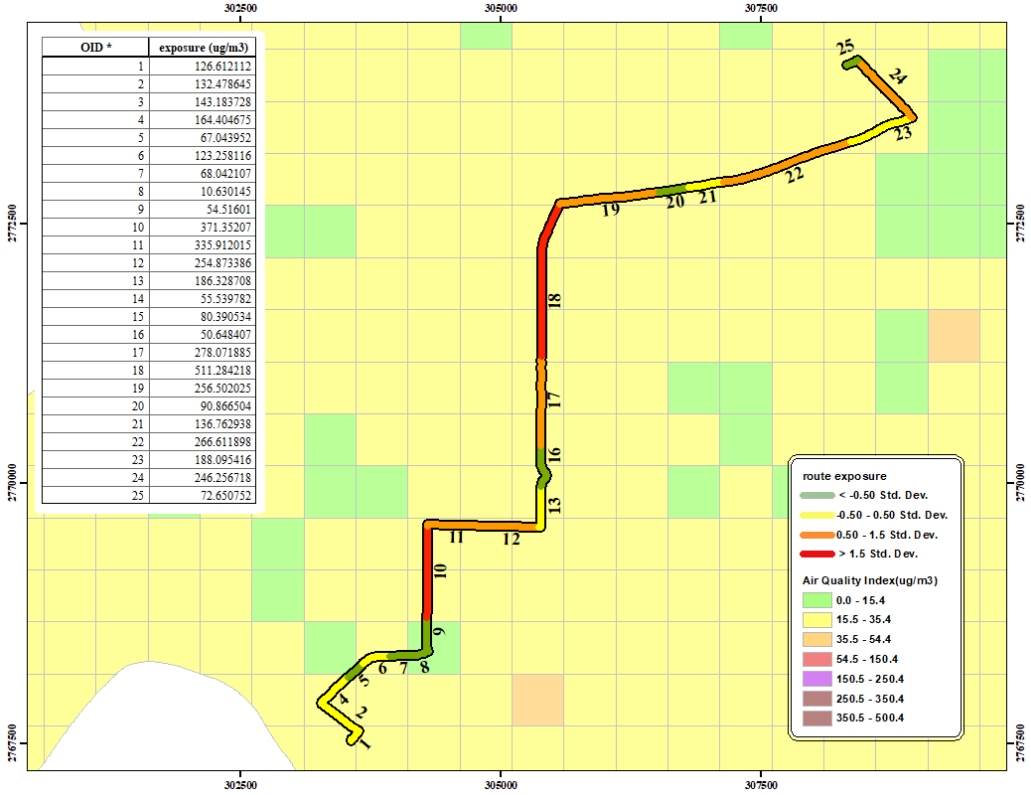
圖 5.6 IDW 空間推估結果

5.4 路線暴露量計算

透過 Open Route Service Directions API 進行路線規劃，所獲取的路線與空間推估結果進行交集，並利用 Directions API 所回傳的路段所需行經時間，用以計

doi:10.6342/NTU201903250

算空氣污染暴露量，以圖 5.7 為例，路線長度為 11.89 公里，計算各路段的暴露量之後，加總可獲得總路線暴露量為 4272μg/m^3。



OID *	exposure (ug/m3)
1	126.612112
2	132.478645
3	143.183728
4	164.404675
5	67.043952
6	123.258116
7	68.042107
8	10.630145
9	54.51601
10	371.35207
11	335.912015
12	254.873386
13	186.328708
14	55.539782
15	80.390534
16	50.648407
17	278.071885
18	511.284218
19	256.502025
20	90.866504
21	136.762938
22	266.611898
23	188.095416
24	246.256718
25	72.650752

圖 5.7 路線暴露計算成果

5.5 路線修正

本研究透過 Open Route Service Directions API 進行路線規劃，由於本研究是將空間推估所產生的網格結果，經由不同的空氣品質分級進行篩選，以找出空氣品質較差的區域，以作為路線規劃的障礙區，故若是網格設置過大，將可能導致障礙區範圍覆蓋至周圍其他的道路，造成路線規劃結果受到影響(如圖 5.8)，此範例為網格 500 公尺的狀況下，障礙區將涵蓋周圍的道路，進而影響到路線修正的結果。

doi:10.6342/NTU201903250

圖 5.8 網格大小影響路線修正

圖 5.9 的案例為多個障礙區出現，系統會將個別的障礙區進行路線規劃，以獲取多條路線，雖然這些路線可能會穿越某些障礙區，但最後的決定權將由使用者決定，系統仍然會協助計算路線平均暴露量。

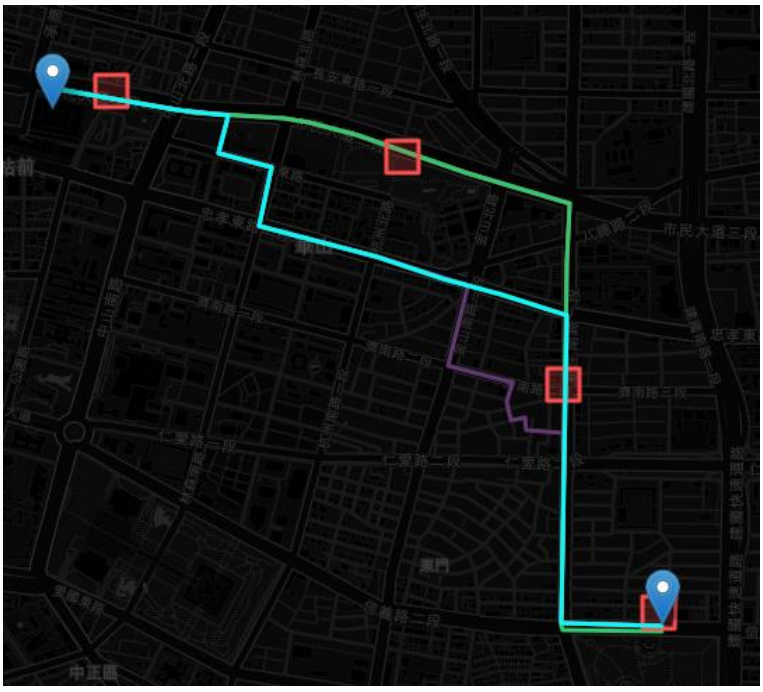


圖 5.9 障礙區個別影響路線

doi:10.6342/NTU201903250

5.6 系統建置

本研究希望在考量空氣污染暴露量及移動成本之下進行路線的選擇，透過 Open Route Service Directions API 進行路線規劃，並依照空氣品質指標篩選出空氣品質較差的區域以作為障礙區，進行後續的路線修正，並計算路線的平均空氣污染暴露量，給予使用者路線選擇上的決策參考。

系統實作上以 WebGIS 的方式呈現，透過 WebApp 與使用者進行互動，前端地圖介面使用 Leaflet 進行開發，以 Python 進行資料的介接及空間資料的處理，透過 GeoServer API 進行 WMTS 的發布，空間資料以 Geometry 的型態進行儲存，並透過 GeoJSON 的格式於傳遞於前後端，藉此本研究提出的架構將獲得以下好處：

(1) Open Route Service Directions API 的使用將減少路線資料及道路資訊維護及更新的成本，並能快速拓展應用範圍不受限於現有圖資，改善過去研究資料層面的問題。

(2) 以 WebGIS 的方式進行呈現，將空氣品質監測數據依照空氣品質指標進行資料視覺化，以地圖的方式傳遞訊息給民眾，讓民眾對於周遭環境的空氣品質狀態及通勤路線選擇上有所幫助。

(3) 透過響應式設計以 WebAPP 的方式提供服務給使用者，將能跨平台的提供服務，減少開發及維護成本。

(4) Python 進行空間資料的處理，將不需依賴相關軟體，且 Python 可於不同作業系統上運行，因此不僅開發上具有彈性，在應用時也能確保系統能穩定部屬。

(5) 透過 GeoServer API 將能動態的更新空氣品質圖層，以提供前端進行展示。

(6) 以 MySQL 中的 Geometry 類別儲存空間資料，透過 WKT(well-known

doi:10.6342/NTU201903250

text)的形式描述空間資料，是由 OGC(Open Geospatial Consortium)所制定的標準類型，因此多數資料庫均可支援，如 SQL Server、PostgreSQL……等等，在資料庫替換上將不會受到侷限。

(7) GeoJSON 是一種地理空間資料交換的格式，透過 JSON 來進行空間圖層及屬性的描述，亦是一種開放且常見的資料型態，在於資料的傳遞上具有輕量的優勢。

doi:10.6342/NTU201903250

第六章　結論與建議

6.1 結論

本研究透過考量空氣污染暴露量以及移動成本下進行路線的修正，改善過去僅提供最低污染暴露量路線時，該路線可能不符合民眾實際生活中的需求。透過研究中提出的架構，在路線的修正上，透過 Open Route Service Directions API 進行路線規劃，改善過去研究於應用上需要維護大量路網資料的問題及應用擴充上存在的困難；並依照空氣品質指標篩選出障礙區域，將能基於原有熟悉的路線進行變更，並透過 WebGIS 的互動方式，提供給使用者路線選擇上的建議。

本研究透過相關地理資訊技術建構的資訊化系統，於資料傳遞及處理上，均是透過開放及通用的資料格式或工具進行，因此各個模組未來可容易地進行更換，以提升整個服務的品質；用戶端則以 WebAPP 的形式提供，避免了移動式裝置作業系統差異的問題，給予使用者個人化適性服務，在路線選擇上進行建議，以減少民眾在通勤時所承受的空氣污染暴露。

6.2 建議

6.2.1 空氣污染暴露量

本研究的空氣品質數值使用微感測器數值進行空間推估，未來可整合不同的測站的數據源、不同的環境影響因子或是以不同的模式進行推估，以更精準地反映出民眾在於路線移動上所受到的空氣污染暴露量。

6.2.2 路線修正

本研究在於路線修正是透過 Open Route Service Directions API 進行，使用空間推估的網格資料作為障礙區，未來將可探討障礙區的大小、形狀是否對於路線修正上具有影響。

doi:10.6342/NTU201903250

參考文獻

Anowar, S., Eluru, N., & Hatzopoulou, M. (2017). Quantifying the value of a clean ride: How far would you bicycle to avoid exposure to traffic-related air pollution? ***Transportation Research Part A: Policy and Practice***, *105*, 66-78. doi:https://doi.org/10.1016/j.tra.2017.08.017

Chen, L.-J., Ho, Y.-H., Hsieh, H.-H., Huang, S.-T., Lee, H.-C., & Mahajan, S. (2017). ADF: an Anomaly Detection Framework for Large-scale PM2.5 Sensing Systems. ***IEEE Internet of Things Journal***, *PP*(99), 1-1. doi:10.1109/jiot.2017.2766085

Cohen, A. J., Brauer, M., Burnett, R., Anderson, H. R., Frostad, J., Estep, K., . . . Forouzanfar, M. H. (2017). Estimates and 25-year trends of the global burden of disease attributable to ambient air pollution: an analysis of data from the Global Burden of Diseases Study 2015. ***Lancet***, *389*(10082), 1907-1918. doi:10.1016/s0140-6736(17)30505-6

Ham, W., Vijayan, A., Schulte, N., & Herner, J. D. (2017). Commuter exposure to PM2.5, BC, and UFP in six common transport microenvironments in Sacramento, California. ***Atmospheric Environment***, *167*, 335-345. doi:https://doi.org/10.1016/j.atmosenv.2017.08.024

Homayoon, Z., Mohsen, S., & Majid, M. (2015). A New Method for Urban Travel Rout Planning Based on Air Pollution Sensor Data. ***Current World Environment***, *30*(48), 699-704. doi:10.12944/CWE.10.Special-Issue1.83

Jarjour, S., Jerrett, M., Westerdahl, D., de Nazelle, A., Hanning, C., Daly, L., . . . Balmes, J. (2013). Cyclist route choice, traffic-related air pollution, and lung function: a scripted exposure study. ***Environmental Health***, *12*. doi:10.1186/1476-069x-12-14

LASS-Community. (2018). PM2.5 OPEN DATA PORTAL. Retrieved from https://pm25.lass-net.org/

Lelieveld, J., Evans, J. S., Fnais, M., Giannadaki, D., & Pozzer, A. (2015). The contribution of outdoor air pollution sources to premature mortality on a global scale. ***Nature***, *525*, 367. doi:10.1038/nature15371

Li, H. C., Chiueh, P. T., Liu, S. P., & Huang, Y. Y. (2017). Assessment of different route choice on commuters' exposure to air pollution in Taipei, Taiwan. ***Environmental Science and Pollution Research***, *24*(3), 3163-3171. doi:10.1007/s11356-016-8000-7

Liu, W.-T., Ma, C.-M., Liu, I. J., Han, B.-C., Chuang, H.-C., & Chuang, K.-J. (2015). Effects of commuting mode on air pollution exposure and cardiovascular health among young adults in Taipei, Taiwan. ***International Journal of Hygiene and***

doi:10.6342/NTU201903250

Environmental Health, *218*(3), 319-323. doi:https://doi.org/10.1016/j.ijheh.2015.01.003

Molter, A., & Lindley, S. (2015). Influence of walking route choice on primary school children's exposure to air pollution--A proof of concept study using simulation. ***Sci Total Environ***, *530-531*, 257-262. doi:10.1016/j.scitotenv.2015.05.118

Park, Y. M., & Kwan, M.-P. (2017). Individual exposure estimates may be erroneous when spatiotemporal variability of air pollution and human mobility are ignored. ***Health & Place***, *43*, 85-94. doi:10.1016/j.healthplace.2016.10.002

S. Nedkov, S. Z. (2011). Enabling obstacle avoidance for Google maps' navigation service.

Vamshi, B., & Prasad, R. V. (2018, 5-8 Feb. 2018). *Dynamic route planning framework for minimal air pollution exposure in urban road transportation systems.* Paper presented at the 2018 IEEE 4th World Forum on Internet of Things (WF-IoT).

Wong, D. W., Yuan, L., & Perlin, S. A. (2004). Comparison of spatial interpolation methods for the estimation of air quality data. ***Journal Of Exposure Analysis And Environmental Epidemiology***, *14*, 404. doi:10.1038/sj.jea.7500338

World Health Organiztion. (2018). Ambient (outdoor) air quality and health. Retrieved from http://www.who.int/mediacentre/factsheets/fs313/en/

行政院環境保護署. (2015). *清淨空氣行動計畫*.

行政院環境保護署. (2018a). 空氣品質指標 - 各項污染物. Retrieved from https://taqm.epa.gov.tw/taqm/tw/b0202.aspx

行政院環境保護署. (2018b). 空氣品質監測網. Retrieved from https://taqm.epa.gov.tw/taqm/tw/b0905.aspx

行政院環境保護署. (2018c). 環境即時通. Retrieved from https://www.epa.gov.tw/Page/6796FBDB3AC1C58F/50cf1d33-e9cb-4b0b-98cf-12b1f8e90a51

行政院環境保護署空保處. (2018). 臺北都會區通勤期間之空氣污染物暴露量調查. Retrieved from https://enews.epa.gov.tw/enews/fact_Newsdetail.asp?InputTime=1070531152746

doi:10.6342/NTU201903250

附錄一 系統程式碼

此部分程式碼主要為後端路由、資料接收及路線計算為主，系統前端展示等完整程式碼將放置於 GitHub 上進行維護及更新，可透過以下連結進行存取：https://github.com/YuChunTsao/AirPollutionRouting。

1. 系統環境

主要以 Python3.6.5 進行開發，透過 Anaconda 進行環境及相關套件的安裝。

2. Open Route Service Direction API 的 Token 申請及設置

本研究使用到 Open Route Service Direction API 進行路線規劃，需事先至 Open Route Service 網站進行註冊及 Token 的申請，並將 API Token 存放於 APIKEY 的檔案中以提供程式碼的存取使用。

3. 使用 Flask 建立 API 服務

Flask 是一個由 Python 所撰寫的輕量型網路框架，本系統用此進行網頁路由的管理及 API 的建立，以下為 app.py 的程式碼。

```
from flask import Flask, render_template, jsonify, request
from module.Direction import Direction

app = Flask(__name__)

# index page
@app.route('/')
```


doi:10.6342/NTU201903250

```
def index():
    return render_template('index.html')


# My direction api
@app.route('/my_direction_api', methods=['POST'])
def my_direction():
    # get data from request
    data = request.get_json()
    origin = data['origin']
    destination = data['destination']
    travelMode = data['travelMode']

    # open route service direction api
    direction = Direction(origin, destination, travelMode)
    # Regular Route
    routes_list = direction.start()

    return jsonify(routes_list)

if __name__ == '__main__':
    app.debug = True
app.run()
```

4. PM2.5 資料篩選及前處理

透過 request 套件接收 https://pm25.lass-net.org/GIS/IDW/data/data.json 的空氣

doi:10.6342/NTU201903250

品質數據，並 Geopandas 套件進行篩選及產生研究中所需要的資料。

```
import requests, json
import numpy as np
import pandas as pd
import geopandas as gpd
from dateutil.parser import parse
from shapely.geometry import MultiPolygon, Polygon, Point


class PM25:
    """
        return pandas dataframe
    """
    def __init__(self):
        self.url = 'https://pm25.lass-net.org/GIS/IDW/data/data.json'
        self.time = None
        # as a filename
        self.parseTime = None
        self.points = None
        self.columns = None
        self.pm25 = None

        self.__get()


    def __get(self):
```


doi:10.6342/NTU201903250

```
        r = requests.get(self.url)
        result = r.json()

        self.time = result['latest-updated-time']
        self.points = result['points']
        self.columns = result['point-meta']

        b = parse(self.time)
        self.parseTime = b.strftime("%Y%m%dT%H%M%SZ")

        self.pm25 = pd.DataFrame(self.points, columns=self.columns)

        print(self.time)
        print(self.parseTime)
        # print(self.points)
        print(self.columns)

def selectPoints(self):
        """ select points by location """
        ### TWD97's WGS84 bounds
        # 114.32 17.36
        # 123.61 26.96
        ###

        self.pm25 = self.pm25.loc[self.pm25['gps_lat'] > 17.36]
```


doi:10.6342/NTU201903250

```
                self.pm25 = self.pm25.loc[self.pm25['gps_lat'] < 26.96]
                self.pm25 = self.pm25.loc[self.pm25['gps_lon'] > 114.32]
                self.pm25 = self.pm25.loc[self.pm25['gps_lon'] < 123.61]

                ### create geometry
                self.pm25['Coordinates'] = list(zip(self.pm25.gps_lon,
self.pm25.gps_lat))
                self.pm25['Coordinates'] = self.pm25['Coordinates'].apply(Point)

                ### create geodataframe
                pm25_point_wgs84 = gpd.GeoDataFrame(self.pm25,
geometry='Coordinates')
                pm25_point_wgs84.crs = {'init' :'epsg:4326'}

                #### reproject (epsg:4326 -> epsg:3826)
                crs = "+proj=tmerc +lat_0=0 +lon_0=121 +k=0.9999 +x_0=250000
+y_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
                pm25_point = pm25_point_wgs84.to_crs(crs)

                ### select layer by location (taipei county - twd97)
                taipei_county_twd97 = gpd.read_file('../data/taipei_county_twd97.shp')

                ### check point(pm25_point_twd97) wthin
polygon(taipei_county_twd97)
                polygon = taipei_county_twd97.geometry[0]
```


doi:10.6342/NTU201903250

```
                result = pm25_point.within(polygon)        # within() will return bool
                result = result * 1

                ### clip
                self.pm25['status'] = result.tolist()
                pm25_point = gpd.GeoDataFrame(self.pm25, geometry='Coordinates')
                pm25_point.crs = {'init' :'epsg:4326'}

                ### reproject
                crs = "+proj=tmerc +lat_0=0 +lon_0=121 +k=0.9999 +x_0=250000
+y_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
                pm25_point = pm25_point.to_crs(crs)
                taipei_pm25_point = pm25_point.loc[pm25_point['status'] == 1]

                # remove bias
                df = taipei_pm25_point[taipei_pm25_point['pm2.5'] >
taipei_pm25_point['pm2.5'].mean() + 3*taipei_pm25_point['pm2.5'].std()]
                taipei_pm25_point = taipei_pm25_point.drop(df.index.tolist())

                ### output file
                taipei_pm25_point.to_file('../data/points/taipei_pm25_point_' +
self.parseTime + '.shp', driver='ESRI Shapefile', encoding='utf-8')

                return taipei_pm25_point
```


doi:10.6342/NTU201903250

```
if __name__ == '__main__':
    pm25 = PM25()
    points = pm25.selectPoints()
    print(points)
```

5. 空氣品質資料進行空間推估

依照各測站的監測數值進行 IDW，並將計算結果放置到自行定義的面資料中儲存，Grid.py 用來產生網格，可定義網格的大小以及邊界，IDW.py 則可將測站的監測數值進行 IDW 的計算，並依照空氣品質指標的級距進行顏色的設置，以下為 Grid.py 及 IDW.py 的程式碼。

Grid.py

```
import numpy as np
import pandas as pd
import geopandas as gpd


class Grid:
    def __init__(self, size, bounds):
        self.size = size
        self.bounds = bounds
        self.minx = None
        self.miny = None
        self.maxx = None
        self.maxy = None
        self.rows = None
        self.columns = None
        self.grid = self.create()
```


doi:10.6342/NTU201903250

```
        def create(self):
            self.bounds['minx'] -= self.bounds['minx'] % self.size
            self.bounds['miny'] -= self.bounds['miny'] % self.size
            self.bounds['maxx'] = self.bounds['maxx'] - (self.bounds['maxx'] %
self.size) + self.size
            self.bounds['maxy'] = self.bounds['maxy'] - (self.bounds['maxy'] %
self.size) + self.size

            self.minx = np.min(self.bounds['minx'])
            self.miny = np.min(self.bounds['miny'])
            self.maxx = np.max(self.bounds['maxx'])
            self.maxy = np.max(self.bounds['maxy'])

            # grid
            gridx = np.arange(self.minx, self.maxx, self.size)
            gridy = np.arange(self.miny, self.maxy, self.size)

            self.rows = len(gridy)
            self.columns = len(gridx)

            grid = np.zeros((self.rows, self.columns),dtype='float32')

            return grid
```


doi:10.6342/NTU201903250

IDW.py

```
from PM25 import PM25
from Grid import Grid

import numpy as np
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import MultiPolygon, Polygon, Point

class IDW:
    def __init__(self, x, y, v, grid, power):
        self.x = x
        self.y = y
        self.v = v
        self.grid = grid
        self.power = power
        self.result = self.__calculate()
        self.polygon = self.__toPolygon()

    def __calculate(self):
        grid = self.grid.create()
        for i in range(grid.shape[0]):
            for j in range(grid.shape[1]):
                distance = np.sqrt((self.x-i)**2+(self.y-j)**2)
```

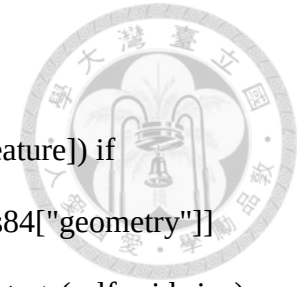

doi:10.6342/NTU201903250

```
                    if (distance**self.power).min()==0:
                        grid[i,j] = self.v[(distance**self.power).argmin()]
                    else:
                        total = np.sum(1/(distance**self.power))
                        grid[i,j] = np.sum(self.v/(distance**self.power)/total)
            return grid

        def __toPolygon(self):
            # x, y origin
            XleftOrigin = self.grid.minx
            XrightOrigin = self.grid.minx + self.grid.size
            YtopOrigin = self.grid.maxy
            YbottomOrigin = self.grid.maxy - self.grid.size

            polygons = []

            for i in range(self.grid.rows):
                Xleft = XleftOrigin
                Xright = XrightOrigin
                for j in range(self.grid.columns):
                    polygons.append(Polygon([(Xleft, YtopOrigin), (Xright,
YtopOrigin), (Xright, YbottomOrigin), (Xleft, YbottomOrigin)]))
                    Xleft = Xleft + self.grid.size
                    Xright = Xright + self.grid.size
                YtopOrigin = YtopOrigin - self.grid.size
```


doi:10.6342/NTU201903250

```
                YbottomOrigin = YbottomOrigin - self.grid.size

            self.result = self.result[::-1]
            pm25_value = self.result.ravel()

            df = pd.DataFrame({
                'value': pm25_value
            })
            crs = "+proj=tmerc +lat_0=0 +lon_0=121 +k=0.9999 +x_0=250000
+y_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
            grid = gpd.GeoDataFrame(df, geometry=polygons, crs=crs)

            return grid
            # grid.plot(column='value', cmap='RdYlGn_r')
            # plt.show()

        def toGeoJSON(self, gdf, output_time):
            crs = "+proj=longlat +datum=WGS84 +no_defs"
            intersection_wgs84 = gdf.to_crs(crs)
            color_list = []
            for i in range(len(intersection_wgs84)):
                value = intersection_wgs84.iloc[i]['value']
                color = self.__setColor(value)
                color_list.append(color)
```


doi:10.6342/NTU201903250



```
                intersection_wgs84["color"] = color_list
                intersection_wgs84["geometry"] = [MultiPolygon([feature]) if
type(feature) == Polygon else feature for feature in intersection_wgs84["geometry"]]
                intersection_wgs84.to_file('../data/grids/taipei_grid_' + str(self.grid.size)
+ 'm_' + output_time + '.geojson', driver='GeoJSON', encoding='utf8')
                intersection_wgs84.to_file('../data/grids/taipei_grid_' + str(self.grid.size)
+ 'm_lastest.geojson', driver='GeoJSON', encoding='utf8')

        def __setColor(self, value):
            color = None
            if value <= 15.4:
                color = '#00E800'
            if value > 15.4 and value <= 35.4:
                color = '#FFFF00'
            if value > 35.4 and value <= 54.4:
                color = '#FF7E00'
            if value > 54.4 and value <= 150.4:
                color = '#FF0000'
            if value > 150.4 and value <= 250.4:
                color = '#8F3F97'
            if value > 250.4 and value <= 350.4:
                color = '#7E0023'
            if value > 350.4 and value <= 500.4:
                color = '#7E0023'
```


doi:10.6342/NTU201903250

```
            return color


    if __name__ == '__main__':
        pm25 = PM25()
        points = pm25.selectPoints()

        # type is Point
        geometry = points.geometry

        taipei_county_twd97 = gpd.read_file('../data/taipei_county_twd97.shp')
        bounds = taipei_county_twd97.bounds
        grid = Grid(500, bounds)

        # get lat, lng and value
        x = [int(np.ceil((lat - grid.minx)/grid.size)) for lat in geometry.x]
        x = np.asarray(x)
        y = [int(np.ceil((lng - grid.miny)/grid.size)) for lng in geometry.y]
        y = np.asarray(y)
        v = [value for value in points['pm2.5']]
        v = np.asarray(v)
        idw = IDW(y,x,v,grid,2)

        intersection = gpd.overlay(taipei_county_twd97, idw.polygon,
how='intersection')
        intersection.to_file('../data/grids/taipei_grid_' + str(grid.size) + 'm_' +
```


doi:10.6342/NTU201903250

pm25.parseTime + '.shp', driver='ESRI Shapefile', encoding='utf-8')

idw.toGeoJSON(intersection, pm25.parseTime)

6. 路線規畫

透過 Open Route Service Direction API 進行路線規劃，並將先前計算好的空間內插網格資料引入，獲取其中的空氣品質數據，來進行空氣污染暴露量的計算，以及作為障礙區的篩選來源，計算完的結果將回傳前五筆路線。

Direction.py

```
import json
import requests
import openrouteservice
from openrouteservice import convert
from module.Exposure import Exposure

from geojson import Feature, Point , LineString, FeatureCollection
import polyline
import geopandas as gpd
from pyproj import Proj
from shapely import geometry
from shapely.geometry import shape, Polygon, mapping, MultiPolygon,
LineString, Point
```


doi:10.6342/NTU201903250

```
class Direction:
    def __init__(self, origin, destination, mode):
        # 設定起點終點進行路線規劃
        print("open route service directions api")
        f = open("./APIKEY", "r")
        self.key = f.read()
        self.origin = origin
        self.destination = destination
        self.mode = mode
        self.client = openrouteservice.Client(key=self.key) # Specify your
personal API key
        self.avoided_point_list = []
        self.idw_grid =
gpd.read_file('./data/taipei_grid_500m_20190610052502.geojson')

        # self.barriers_level_1 = self.__selectBarriers(1)
        # self.barriers_level_2 = self.__selectBarriers(2)
        # self.barriers_level_3 = self.__selectBarriers(3)
        self.barriers = []
        # self.barriers.append(self.barriers_level_1)
        # self.barriers.append(self.barriers_level_2)
        # self.barriers.append(self.barriers_level_3)

    def start(self):
        routes_list = []
```


doi:10.6342/NTU201903250

```
# 第一次路線規劃 - 最短路徑
ORS_route, ORS_route_steps = self.CreateRoute()
exposure = Exposure(ORS_route, ORS_route_steps)

# geodataframe
exposure_route = exposure.calculate()
routes_list.append(exposure_route)

# 選擇障礙區等級
self.barriers_level_1 = self.__selectBarriers(1, ORS_route)
# self.barriers_level_2 = self.__selectBarriers(2, ORS_route)
# self.barriers_level_3 = self.__selectBarriers(3, ORS_route)

self.barriers.append(self.barriers_level_1)
# self.barriers.append(self.barriers_level_2)
# self.barriers.append(self.barriers_level_3)

# barriers of this request
barriers_list = []
for index, barrier in enumerate(self.barriers):
    print("level = " + str(index))
    for geom, value in zip(barrier.geometry, barrier.value):
        poly = list(geom)[0]
        # self.avoided_point_list = []
        self.avoided_point_list.append(poly)
```


doi:10.6342/NTU201903250

```
                    ## 障礙區顯示 - GeoJSON
                    barrier = Feature(geometry=poly, properties={"value": value})
                    barriers_list.append(barrier)

                    try:
                        ORS_route, ORS_route_steps = self.CreateRoute()

                        # calculate air pollution exposure
                        exposure = Exposure(ORS_route, ORS_route_steps)
                        exposure_route = exposure.calculate()

                        # 判斷若路線相同就不回傳
                        total_distance =
exposure_route['route_info']['total_distance']
                        if total_distance in [route['route_info']['total_distance'] for
route in routes_list]:
                            print("same")
                        else:
                            routes_list.append(exposure_route)
                        print('Generated alternative route, which avoids affected
areas.')

                    except Exception:
                        print('Sorry, there is no route available between the
```


doi:10.6342/NTU201903250

requested destination because of too many blocked streets.')

依照空氣污染總暴露量進行排序，由小至大。

routes_list.sort(key=lambda x: x['route_info']['total_exposure'])

routes_list.sort(key=lambda x: x['route_info']['average_exposure'])

回傳前五筆

if len(routes_list) < 5:

result = {"result": routes_list[0:len(routes_list)]}

else:

result = {"result": routes_list[0:5]}

return result

def CreateRoute(self):

coords = [[self.origin['lng'], self.origin['lat']], [self.destination['lng'], self.destination['lat']]]

route_request = {'coordinates': coords,

'format_out': 'geojson',

'profile': self.mode,

'preference': 'shortest',

'instructions': True,

'options': {'avoid_polygons': geometry.mapping(MultiPolygon(self.avoided_point_list))}}

doi:10.6342/NTU201903250

```
        # result = client.directions(coords,    profile=self.mode)
        ORS_route = self.client.directions(**route_request)
        ORS_route_steps = self.__toGeoJSON(ORS_route)

        # with open('./data/ORS_route.geojson', 'w') as outfile:
        #         json.dump(ORS_route, outfile)

        return ORS_route, ORS_route_steps


def __selectBarriers(self, level, route):
        # 普通
        if level == 1:
                min = 15.4
                # max = 35.4

        # 對敏感族群不健康
        if level == 2:
                min = 35.4
                # max = 54.4


        # 對所有族群不健康
        if level == 3:
```


doi:10.6342/NTU201903250

```
        min = 54.4
        # max = 150.4

    barriers_level = self.idw_grid.loc[self.idw_grid['value'] > min]

    # 與路線交集的障礙區
    route_gdf = gpd.GeoDataFrame.from_features(route)
    # route = gpd.read_file('./data/ORS_route.geojson')
    barriers = gpd.sjoin(barriers_level, route_gdf, op='intersects')
    # print(len(barriers))
    # print(barriers)

    return barriers


def __toGeoJSON(self, data):
    geometry = data['features'][0]['geometry']
    steps = data['features'][0]['properties']['segments'][0]['steps']
    coordinates = geometry['coordinates']
    metadata = data['metadata']
    profile = metadata['query']['profile']
    timestamp = metadata['timestamp']

    legs = []
    for step in steps:
```


doi:10.6342/NTU201903250

```
        points_index = step['way_points']
        if points_index[0] == points_index[1]:
            print("line end")
            continue
        else:
            points = coordinates[points_index[0]:points_index[1] + 1]
            linestring = LineString(points)

            leg = Feature(geometry = linestring, properties= {
                'distance': step['distance'],
                'duration': step['duration'],
                'type': step['type'],
                'instruction': step['instruction'],
                'name': step['name'],
                'way_points': step['way_points'],
                'profile': profile,
                'timestamp': timestamp
            })

            legs.append(leg)

legs = FeatureCollection(legs)

# print(json.dumps(legs, indent=4, sort_keys=True))
```


doi:10.6342/NTU201903250

```
        with open('./data/ORS_route_steps.geojson', 'w') as outfile:
            json.dump(legs, outfile)

        return legs
```

7. 路線的空氣污染暴露量計算

依照 IDW 的推估結果及運具類型，計算每條路線的空氣污染暴露量，並將資料處理為 GeoJSON 格式進行回傳，以利於前端展示使用。

Exposure.py

```
import os
import geopandas as gpd
import numpy as np
import pandas as pd
from shapely.ops import transform
from functools import partial
import pyproj
import matplotlib.pyplot as plt
from geojson import Feature, FeatureCollection

class Exposure:
    def __init__(self, ORS_route, ORS_route_steps):
        self.idw_grid =
gpd.read_file('./data/taipei_grid_500m_20190610052502.geojson')
```


doi:10.6342/NTU201903250

```
            # self.route = gpd.read_file('./data/ORS_route.geojson')
            # self.route_steps = gpd.read_file('./data/ORS_route_steps.geojson')
            self.route = gpd.GeoDataFrame.from_features(ORS_route)
            self.route_steps = gpd.GeoDataFrame.from_features(ORS_route_steps)
            self.grid = self.__cutWithRouteBound()

        # cut grid with route's bound
        def __cutWithRouteBound(self):
            bounding_box = self.route.envelope
            df = gpd.GeoDataFrame(gpd.GeoSeries(bounding_box),
columns=['geometry'])
            self.idw_grid['grid_ID'] = self.idw_grid.index.tolist()
            intersections = gpd.overlay(df, self.idw_grid, how='intersection')
            # polygon
            return intersections

        def calculate(self):
            data = []

            # reproject
            project = partial(
                pyproj.transform,
                pyproj.Proj(init='EPSG:4326'),
                pyproj.Proj(init='EPSG:3826')
            )
```


doi:10.6342/NTU201903250

```
        # intersects
        for index1, r in self.route_steps.iterrows():
            for index2, g in self.grid.iterrows():
                if r.geometry.intersects(g.geometry) is True:
                    r_geom = r.geometry
                    r_geom = transform(project, r_geom)
                    g_geom = g.geometry
                    g_geom = transform(project, g_geom)

                    data.append({
                        'SID': index1,
                        'GID': g.grid_ID,
                        'distance': r_geom.intersection(g_geom).length,
                        'total_distance': r.distance,
                        'duration': r.duration *
(r_geom.intersection(g_geom).length / r.distance),
                        'total_duration': r.duration,
                        'grid_pm25': g.value,
                        'instruction': r.instruction,
                        'profile': r.profile,
                        'geometry': r_geom.intersection(g_geom)
                    })

        result = gpd.GeoDataFrame(data)
```


doi:10.6342/NTU201903250

```
            result['ID'] = result.index.tolist()
            result['distance_weights'] = result['distance'] / result['total_distance']
            result['segment_pm25'] = result['grid_pm25'] * result['distance_weights']


            diss_result = result.dissolve(by='SID', aggfunc='sum')
            self.route_steps['segment_pm25'] = diss_result['segment_pm25']
            self.route_steps['distance_value'] = diss_result['distance']
            self.route_steps['duration_value'] = diss_result['duration']

            color_list = []
            for i in range(len(self.route_steps)):
                value = self.route_steps.iloc[i]['segment_pm25']
                color = self.setColor(value)
                color_list.append(color)

            self.route_steps['color'] = color_list

            self.route_steps['segment_exposure'] = self.route_steps['segment_pm25']
* (self.route_steps['duration_value']/60) * 7

            # calculate total exposure
            total_exposure = self.route_steps.segment_exposure.sum()
            # calculate total distance
            total_distance = self.route_steps.distance_value.sum()
```


doi:10.6342/NTU201903250

```
            # calculate total duration
            total_duration = self.route_steps.duration_value.sum()
            # calculate average exposure
            average_exposure = ((self.route_steps['distance_value'] / total_distance)
* self.route_steps['segment_pm25']).sum()

            print("total exposure = " + str(total_exposure) + ' ug/m3')
            print("total distance = " + str(total_distance) + ' meter')
            print("average exposure = " + str(average_exposure) + ' ug/m3')

            result = {
                "route_info": {
                    "total_exposure": total_exposure,
                    "total_distance": total_distance,
                    "total_duration": total_duration,
                    "average_exposure": average_exposure
                },
                "data": self.route_steps.to_json()
            }

            return result

        def setColor(self, value):
            color = None
            if value <= 15.4:
```


doi:10.6342/NTU201903250

```
        color = '#00E800'
if value > 15.4 and value <= 35.4:
        color = '#FFFF00'
if value > 35.4 and value <= 54.4:
        color = '#FF7E00'
if value > 54.4 and value <= 150.4:
        color = '#FF0000'
if value > 150.4 and value <= 250.4:
        color = '#8F3F97'
if value > 250.4 and value <= 350.4:
        color = '#7E0023'
if value > 350.4 and value <= 500.4:
        color = '#7E0023'

return color
```


doi:10.6342/NTU201903250