國立臺灣海洋大學

資訊工程學系

碩士學位論文

指導教授：張雅惠博士

基於地標之淹水警示研究

The Research on Flood Forecasting Based on Landmarks

研究生：劉昱德 撰

中華民國 103 年 7 月

基於地標之淹水警示研究

The Research on Flood Forecasting Based on Landmarks

研究生：劉昱德　　Student：Yu-Te Liu

指導教授：張雅惠　　Advisor：Ya-Hui Chang

國立臺灣海洋大學

資訊工程學系

碩士論文

A Thesis
Submitted to Department of Computer Science and Engineering
College of Electrical Engineering and Computer Scienece
National Taiwan Ocean University
In Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science and Engineering
July 2014
Keelung, Taiwan, Republic of China

中華民國 103 年 7 月

摘要

台灣常年因颱風與豪雨事件所帶來的降雨，釀成嚴重的淹水災害，因此在淹水發生前的預警措施成為重要的研究議題。在之前的研究裡，我們已經建置一個洪氾預警系統，利用水理專家開發的 TwoFD 水理模組輸出預測的淹水區塊，但是直接顯示所有訊息在地圖上往往費時很久且不夠清楚。因此，我們提出了「基於地標的警示訊息」，一方面訊息精簡易於傳送，二方面也可以通知地標內所在的人員盡速疏散。針對如何「快速」地輸出「有效」的警示資訊，我們提出了兩個方法。首先，VC 方法利用 Voronoi diagram 及 R-tree 讓淹水區塊與地標 Voronoi cell 的最小邊界矩形作空間連接，進而找出附近淹水的地標。為了剔除離地標過遠的淹水區塊，再提出 C2L 方法，事先記錄與地標距離小於最小距離門檻的演算區塊並建立 B-tree。最後，透過一系列的實驗以及各種資料集，比較兩個演算法的的可行性與效率。實驗結果顯示，雖然 C2L 事先建立的索引需要較大的空間，但在有效性和效率方面皆優於 VC 方法。

Abstract

The disaster brought by heavy rain has become more and more serious in Taiwan, and it has been an important research issue to provide warning messages before flood. In the past research, we have built a flood forecasting system based on the TwoFD module developed by hydraulic experts, but directly displaying all the flooded cells on maps might take a lot of time. Therefore, we propose to summarize flooded cells based on landmarks. Such messages will be short and clear, and we can evaculate the persons within the landmark in advance. To do this, the main research issue is how to find the effective warning message rapidly. We propose two methods. The VC method uses Voronoi diagram and R-tree to perform spatial join between flooded cells and the minimum bounding rectangle of landmarks' Voronoi cells, and then find the landmarks nearby flooded cells. We also propose the C2L method to pre-calculate the distance between cells and landmarks to preclude those cells whose distances are more than a given threshold. At the end, we have performed a series of experiments and used a variety of data sets to examine the effectiveness and the efficiency of our two methods. Experimental results show that although the C2L method requires more space for the pre-built indices, it performs better in terms of effectiveness and efficiency.

誌謝

首先，感謝指導教授張雅惠博士，對於本論文給予許多幫助，且在研究論文期間不時地共同討論，解決論文的諸多疑點，使學生順利的完成論文。

除此之外，感謝許為元博士和台北科技大學劉傳銘博士百忙之中抽空參與論文審查工作，也感謝林川傑博士給予本論文意見與幫助，提供寶貴的意見，使論文更趨近完善。

最後，感謝實驗室的學長姐，帶領著我們盡快適應實驗室環境以及提供各種方法解決問題，感謝實驗室學弟妹與同學，陪伴著我渡過有趣的碩士生活。最後感謝親愛的家人，在學習階段給予支持和精神上的鼓勵。在此一併致上謝意，謝謝你們。

目錄

圖目錄

表目錄

第 1 章序論

在此章，我們先說明本論文的研究動機與目的，以及提出本論文的研究方法與貢獻，並介紹相關的研究，最後說明各章節的內容及本論文的架構。

1.1 研究動機與目的

台灣地區降雨豐沛，年平均降雨量高達 2515 公釐，降雨量時空分布極不均勻，常因大雨宣洩不及釀成災害。而因颱風與豪雨事件所帶來的降雨，釀成嚴重的淹水災害，造成人民生命及經濟上的重大損失。且當暴雨來襲時，無法事先警告民眾進行必要的預防措施，因而造成極大的災難，之前莫拉克颱風造成的八八水災，就是一個實例。

之前的研究已經建置一個洪犯預警系統[吳 12]，該系統利用水理專家開發的 TwoFD 水理模組，集結所有所需的邊界條件和雨量資料後，執行二維淹水演算模式，以模擬演算河川下游低窪地區的水深高度，做為即時淹水預警區域的研判依據。藉由 TwoFD 水理模組輸出的淹水區塊和水位，利用 Google Maps API，將 Google Maps 地圖資料嵌入到 Web 應用中，如圖 1-1。

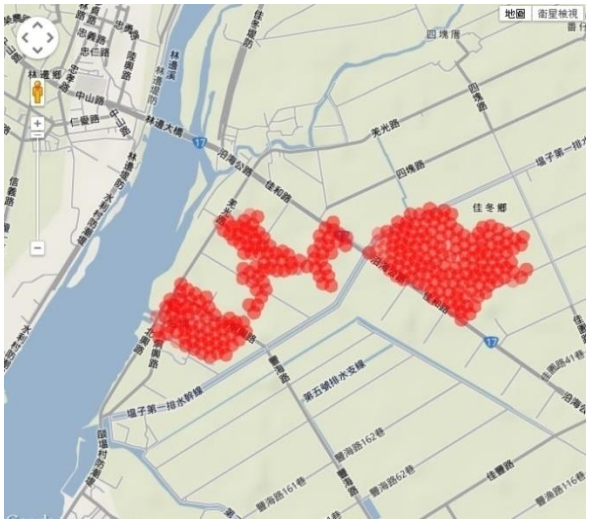


圖 1-1：警戒圖示重疊的現象

不過，該系統將所計算出的每個淹水區塊都顯示在地圖上，有以下兩個缺點：(1)對於低窪地區，淹水區塊相當密集，導致對應的警示圖示互相重疊而雜亂，如此讓救災單位無法做精準的判斷。(2)當淹水嚴重時，需要畫出大量圖示，有時會造成瀏覽器顯示不及而有錯誤訊息。所以，我們提出以地標為主將淹水資訊整合的概念，也就是找出距離淹水區域附近的地標，並輸出了「基於地標的警示訊息」(landmark-based warning message)。所謂的「地標」，是指大家熟知的機關學校或建築物，如圖 1-2，林邊鄉區域一隅，以地圖上的 Marker 標示，而「餉潭國小」即為其中一個。我們希望將 TwoFD 水理模組產生的淹水區塊摘要成數筆簡單的文字訊息，譬如，在餉潭國小南邊的淹水區塊可摘要為：「*餉潭國小南方約 5 公里處即將發生大約 0.55 平方公里的淹水*」。

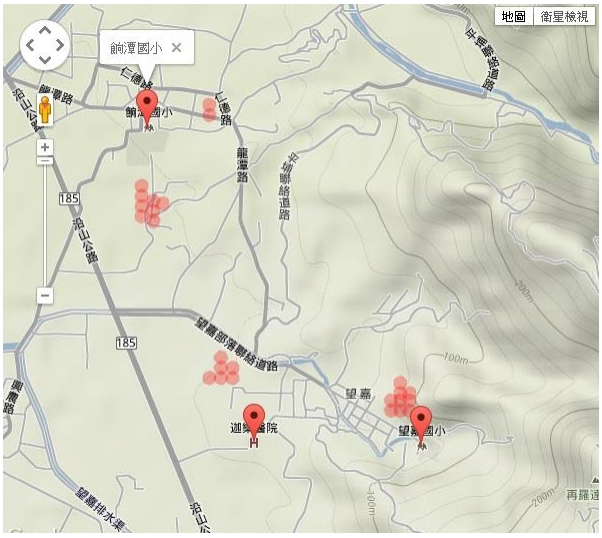


圖 1-2：林邊鄉的地標與淹水範例

此摘要訊息提供了數個好處。首先，在暴雨來臨時，有許多機關單位都不知道附近即將發生淹水。由於機關單位會被列入地標中，所以此摘要訊息能提醒我們盡速通知正在機關單位當中的人員疏散。其次，由於大多數的民眾都會熟知那些地標的所在位置，所以此摘要資訊也能明確地指出危險的區域警告一般人不要前往。最後，面積大小能夠用以判斷淹水的嚴重性，在救難人員不足的情況下，做為優先救災的順序依據。

1.2 研究方法與貢獻

此研究的主要議題，在於如何「快速」地輸出「有效」的警示資訊。本論文

提出了兩種做法輸出警示資訊,第一個做法是利用 Voronoi diagrams[Fo87]的概念決定每個淹水區塊影響的地標，稱作「VC 方法（The VC approach）」。在此方法中，我們先在前處理階段以地標為中心切出一個個 Voronoi Cell，接著再利用這些 Voronoi Cell 找出最小邊界矩形並建立 R-tree 索引；而在線上查詢時，利用淹水區塊和此 R-tree 進行空間連結（spatial join）的運算以決定哪些地標附近嚴重淹水。為了減少搜尋 R-tree 的次數，進一步提出了改良後的演算法，將較靠近的淹水區塊做合併的動作，減少搜尋的次數進而讓效率更佳。

不過在此方法中，就算淹水區塊距離地標很遠，只要落在其 VC 中，也視為會影響該地標，所以我們提出第二個做法，利用最小距離門檻，剔除影響力較低的淹水區塊，同時，我們也探討直接利用淹水區塊的中心點而非區塊本身進行空間連接對結果是否有所影響，此方法稱作「C2L 方法（The cell-to-landmark approach）」。

本論文的貢獻總結如下所示：

1. 我們提出基於地標顯示警示資訊的想法，而不是直接將淹水點標示於地圖上。

2. 我們提出的 VC 方法，在淹水發生前事先切出所有地標的 Voronoi cell 並找出其最小邊界矩形，且建立 R-tree 索引，並在淹水發生時，利用 TwoFD 水理模組所提供的淹水區塊與建好的 R-tree 索引作空間連接,快速的找出附近淹水最嚴重的地標，以給予適當的摘要化警示標語，且在線上查詢時，為了減少搜尋 R-tree 的次數，提出了改良過後的 VC+方法。

3. 我們進一步提出 C2L 方法，為了讓輸出更準確，以最小距離門檻而非 VC 的概念來決定淹水區塊影響的地標，並利用座標建立 B-tree 索引，於淹水區塊產生時，利用淹水區塊的座標查詢並給予更準確的摘要化警示標語。

4. 最後，分別利用林邊鄉真實資料以及人工資料針對 VC、VC+與 C2L 進行一系列的實驗，實驗結果顯示 C2L 方法在在各種情況下的時間花費皆較 VC 方法少，但 VC+在最小合併門檻的設定下，也可能較 C2L 的效率佳。

1.3 相關研究

首先，我們先討論關於淹水預警系統的相關研究。由於淹水是隨時可能發生的，所以水利署提出了一個行動水情的手機應用程式[MWR]，完整提供了全台灣每一條主要溪流水位，雨量監測站和水庫水位的即時資訊，以及每個縣市鄉鎮是否達到淹水的標準，可分為一級警戒和二級警戒。另外，水利署和台灣 Google 在 2013 年 7 月宣布啟用「台灣災害應變資訊平台」[GPA]，是日本之後第 2 個採行此平台的亞洲國家；提供中央氣象局、水土保持局等政府機關防災資訊，如圖 1-3，為 2014 年 6 月 3 日梅雨季所帶來的豪大雨警示圖，右邊地圖標示需警示的地點，左邊則可將顯示的方式依所有地點或目前附近的地點排列顯示，並依照關連性或日期排序。

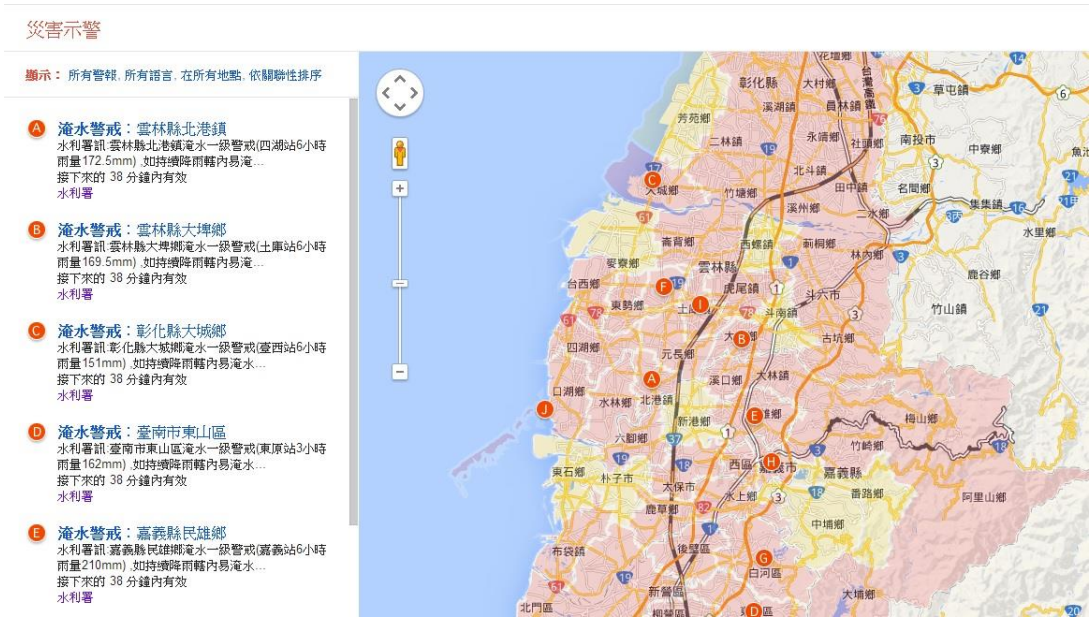


圖 1-3：台灣災害應變資訊平台

其次,「最近鄰居」(the nearest neighbors)搜尋[RKV95]已經出現在許多應用，如地理資訊系統，是利用 R-tree[Gu84]找出最近鄰居。R-tree 是一種類似 B-tree[BM70]的多維度樹狀結構索引裝置，由中間節點和葉節點所構成，常用來快速存取大量的空間物件。換句話說，B-tree 是傳統一維數值或文字型態的索引資料結構，而 R-tree 則是多維的空間物件的索引資料結構。另外也有許多研究是有關最近鄰居搜尋的變形議題，如 *K*NN。舉例來說，針對交通運輸網路上靜止不動的物體去處理 snapshot *K*NN queries，有研究者提出一個圖形嵌入的技術[SKS03]，將交通運輸網路轉換成一個高維度的歐幾里德空間，經由此轉換，傳統發展來有效地處理 Euclidean *K*NN query 的方法便可以直接運用。

至於 Voronoi diagrams 則是一個古老的問題，產生的方式為預先指定一組種子（generators），每個種子會有一個相對應的區域，而在該區域內所有的點到該相對應種子的距離，會比到其他區域種子的距離更接近，那些區域被稱為 Voronoi cell。第一個演算法是在 1977 年由 Green 與 Sibson 所發表[GS77]，使用 incremental method，並且需要 $O(n^2)$ 的時間。第一個 $O(n \log_2 n)$ 的演算法則是由 Shamos 與 Hoey （1975）所提出[PS85]，使用 divide and conquer 的方法，然而要實作是公認的複雜。到了 1987 年，Fortune 發明用 plane-sweep 的方法[Fo87]，兼具 incremental method 的容易實做，與 divide and conquer 的 $O(n \log_2 n)$ 複雜度，所以本論文採用此方法。

最後列出一些在空間處理方面相關的論文，[QZKL+12]提出一個“min-dist location selection problem”，該問題是希望在可能位置的集合中選一個位置建立新的公共設施，使得所有的客戶到其最靠近的公共設施的平均距離能最短。該論文討論空間的不同切割方式，包括「quasi Voronoi cell」和圓形的切割方式，並且利用 R-tree 的技術 [Gu84] 建立對應的索引來加速查詢的速度。另一方面，[QCCX12]則討論在大型網路中，如何估算兩點的距離。目前的做法大多選定一些點作為 global landmark，事先計算好所有節點到這些 global landmark 的距離，

並利用三角不等式估算兩個查詢節點的最短距離，但是此方法常會帶來很大的誤差。[QCCX12]提出根據查詢句選擇 local landmark，以得到更好的結果。

1.4 論文架構

本論文其餘各章節的架構如下：於第二章敘述本論文的相關技術背景，其中包含既有的 TwoFD 水理模組及相關定義，藉以對本論文的研究有基礎的認識。第三章介紹 VC 方法，說明其整體架構及如何利用 Voronoi cell 與 R-Tree 找出優先通知的地標。第四章介紹 C2L 方法，說明其整體架構及如何找出每個淹水區塊小於距離門檻的地標，並找出我們需要的資料。於第五章中以實驗比較兩演算法的效率，並於第六章提出本論文的結論與未來方向。

第 2 章技術背景

在此章，我們說明相關的定義，包括介紹 TwoFD 水理模組的主要功能及本論文欲解決的問題。

2.1 TwoFD 水理模組介紹

TwoFD 水理模組是利用多核心伺服器進行平行運算數值模擬，並配合空間與時間的離散處理方式，以發展高效能的二維淹水演算，縮短傳統二維淹水演算計算耗時的缺失，進而預測演算河川下游低窪地區的淹水區塊，以做為淹水預警區域的研判依據。

TwoFD 水理模組在演算前先集結所需要的雨量與流量資料，並將演算區域根據其地理特性切割成數千至數萬個演算區塊（cell），每個區塊皆由大小不一的等腰三角型組成，包含三角形的三個頂點與中心點的經緯度座標（在本論文中以 x、y 座標表示）及面積（area），面積大約為 1-8 平方公里。演算後此模組會輸出預測未來會淹水的區塊（flooded cell）的中心點經緯度座標及面積，且所有輸出的淹水區塊會先以緯度座標軸排序，再以經度的座標軸排序。

2.2 相關定義

地標（landmark）通常分為自然景觀和人為的地標，自然景觀可能為某處景點，如野柳的女王頭，人為的地標，舉凡各地機關、學校和標誌性的建築物等，都稱為地標。

本論文希望藉由 TwoFD 水理模組輸出的淹水區塊，快速找出附近產生淹水的地標，我們並希望附近淹水面積越大的優先輸出，正式的定義如下所示：

給定淹水區塊序列 FCList，地標集合 LMSet，令 $m \in$ LMSet， $m_f = \{ f \mid f \in$ FCList, *f influences m*$\}$，AREA(*f*)代表 *f* 的面積，且定義：

$$A_{m_f} = \sum_{f \in m_f} \text{AREA}(f) \tag{1}$$

將所有的A_{m_f}由大至小排序，然後依序輸出對應的 *m* 編號。

以圖 2-1 為範例，說明了這個問題：[f_1, f_2, f_3, f_4, f_5]為淹水區塊序列，為了易於討論，假設其形狀為正方形，且面積均為 1 平方公里，其中淹水區塊先由 *y* 軸排序，再由 *x* 軸排序，{m_1, m_2, m_3}為地標的集合。首先我們必須找出受淹水區塊影響的地標，很明顯的看出 f_1 和 f_2 影響的地標為 m_1，f_3 影響的地標為 m_2，f_4 和 f_5 影響的地標為 m_3，也就是 $m_1=\{f_1, f_2\}$、$m_2=\{f_3\}$、$m_3=\{f_4, f_5\}$，根據(1)式分別計算如下：

$$\sum_{f \in m_1} \text{AREA}(f) = 1 + 1 = 2、\sum_{f \in m_2} \text{AREA}(f) = 1、\sum_{f \in m_3} \text{AREA}(f) = 1 + 1 = 2$$

再將其由大到小排序後，輸出對應的地標編號為（m_1, m_3, m_2）。

為了讓輸出更容易解讀，本論文會再將每個地標附近的淹水區塊進一步的算出更詳細的方位和距離，也就是針對（1）式所輸出的地標 *m*，我們會同時輸出 m_f 和 *m* 之間的平均距離和方位。在此例中，我們的輸出為：

m_1 北方距離 2 將發生面積 2 的淹水、m_2 西方距離 2 將發生面積 1 的淹水

m_3 南方距離 3 將發生面積 1 的淹水、m_3 東方距離 4 將發生面積 1 的淹水

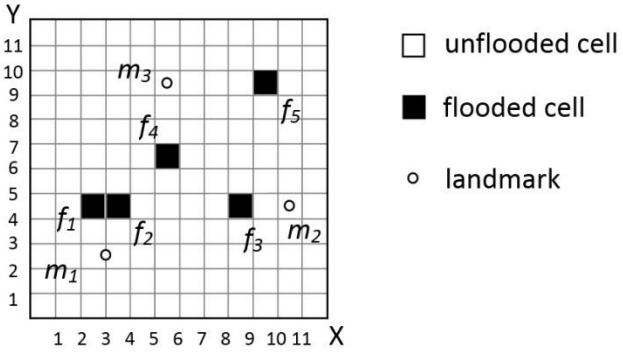


圖 2-1：問題的範例

2.3 問題假設

我們假設所有的資料（地標、淹水區塊）均分佈在歐式空間，其中，地標為點，淹水區塊為區塊。函數 getDist(f_x, f_y, m_x, m_y)代表從淹水點 f 的中心點座標至地標 m 的大圓距離（Great-circle distance）[GB97]，使用的座標系統為 WGS84（World Geodetic System 1984），地球半徑為 6378 公里，其公式為：

Haversine formula:
$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$
$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{1-a})$$
$$\text{distance} = R \cdot c, \quad (2)$$

φ_1、φ_2為地標與淹水區塊的經度，$\Delta\varphi$、$\Delta\lambda$ 為地標與淹水區塊經度和緯度相差的角度，R 為地球半徑(半徑=6378km)。函數 atan2 是正切函數的一個變種，對於任意不同時等於0的實參數 x 和 y，atan2(y, x)所表達的意思是坐標原點為起點，指向(x, y)的射線在坐標平面上與 x 軸正方向之間的角的角度。

函數 getDir(f_x, f_y, m_x,m_y)代表取出 f 的中心點座標位於 m 的初始方位（initial bearing）[IB]，初始方位的定義為如果以 m 為起點，面向 f 時，以面向的那個方向與正北方的夾角，我們將所有角度平均分為八個角度，分別是北、東北、東、東南、南、西南、西與西北，其公式為：

$$\theta = \operatorname{atan2}(\sin\Delta\lambda \cdot \cos\varphi_2, \cos\varphi_1 \cdot \sin\varphi_2 - \sin\varphi_1 \cdot \cos\varphi_2 \cdot \cos\Delta\lambda) \quad (3)$$

此式中的 φ_1、φ_2、$\Delta\lambda$ 同(2)式

第 3 章VC 方法

在本章中，我們介紹 VC（Voronoi cell）方法，包括前處理階段的 MakeVC 演算法及線上查詢階段的 FindNLm 演算法（Find Nearest Landmark）及 FindFCGroup 演算法（Find Flooded Cell Group）。

3.1 前處理階段

VC 方法主要分為兩個部分，前處理階段與線上查詢階段。前處理階段（MakeVC 演算法）是基於 Voronoi cell 的觀念，我們使用 Steven Fortune 的 plane-sweep 方法[Fo87]，將所有的地標視為種子（generator），以地標為基準切割空間。由於考慮到 Voronoi cell 的形狀不規則，所以進一步將切好的 cell 取出最小邊界矩形（minimal bounding rectangle；MBR）[Br+94]，最小邊界矩形是指以若干二維形狀（例如點、直線、多邊形）的最大範圍，即以給定的二維形狀各頂點中的最大橫坐標、最小橫坐標、最大縱坐標、最小縱坐標定下邊界的矩形。最後我們再將每個地標的 Voronoi cell 的最小邊界矩形（簡稱 VC_MBR）建立 R-Tree 索引。另外，我們也會在此階段將每個演算區塊的等腰三角形利用三個頂點座標算出最小邊界矩形。其目的是當 TwoFD 水理模組產生淹水區塊時，於線上查詢將淹水區塊的 MBR（簡稱 FC_MBR）與建好的 R-Tree 索引作空間連接，以便知道每個淹水區塊影響到哪個地標。對應的 MakeVC 演算法如圖 3-1。

演算法名稱：MakeVC 輸入：lmList：地標序列 輸出：每個地標的 Voronoi cell 的 MBR 並建立 R-tree 索引（R_{vc}） 變數：VCSet：每個地標的 Voronoi cell 的集合 　　MBRSet：每個地標 Voronoi cell 的 MBR 集合
MakeVC（lmList）

```
L01  VCSet= VORONOI_DIAGRAM(lmList) //利用演算法[Fo87]
L02  MBRSet = MakeMBR(VCSet)
L03  create R-tree index R_vc on MBRSet
L04  return R_vc
```

圖 3-1：MakeVC 演算法

如圖 3-2，對應三個地標（m_1、m_2、m_3）的 Voronoi cell，為以實線隔出的三個多邊形，分別稱作 VC_{m1}、VC_{m2} 與 VC_{m3}。而圖 3-3 中，虛線框則是 VC_{m1}、VC_{m2} 與 VC_{m3} 各自的 MBR，簡稱為 VC_MBR_1、VC_MBR_2、VC_MBR_3。雖然每個地標的 Voronoi cell 彼此間不相交，但是對應的 MBR 可能會有重疊的部分。如 f_4 淹水區塊正好落在 VC_MBR_1 和 VC_MBR_2 的交集區域中（淺灰區塊），所以會被認為同時影響地標 m_1 和地標 m_3。

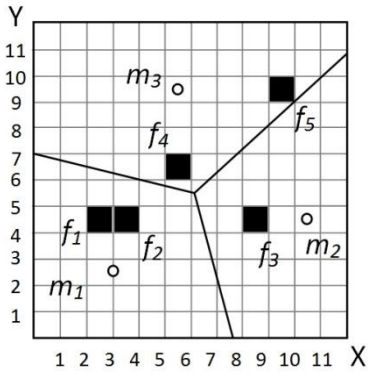


圖 3-2：地標的 Voronoi cell

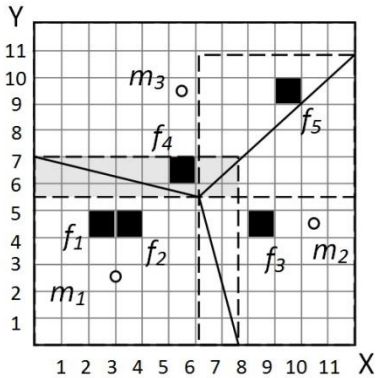


圖 3-3：Voronoi cell 的最小邊界矩形（VC_MBR）

我們利用 VC_MBR 建立的 R-Tree 索引，稱做 R_{vc}，其 leaf node 裡面的每一筆資料(entry)對應到一個地標，如圖 3-4，為一 entry 結構，記錄地標編號(LmID)與 x、y 座標和 Voronoi cell 對應的最小邊界矩形（VC_MBR），VC_MBR 中記錄了 x、y 軸的最小與最大值（x_1、x_2、y_1、y_2）。至於 internal node 則是以其 VC_MBR 值，記錄包含所有小孩的 MBR，以 childnode 指到其所有小孩，如圖 3-5，root node 記錄了包含所有小孩的 VC_MBR。

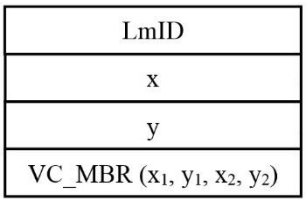


圖 3-4：lmList 的 entry 結構（代表一個地標）

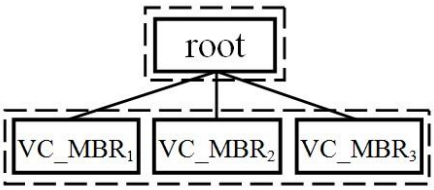


圖 3-5：root node

3.2 FindNLm 演算法

當 TwoFD 水理模組產生淹水區塊時，將淹水區塊的 MBR 與建好的 R-Tree 索引作空間連接(spatial join)，當淹水區塊的 MBR 落入 R_{vc} 中 leafnode 裡的任一 entry 時，便將淹水區塊與其對應地標的距離、方位、地標編號和面積存入 SIF(Summarize Information)，其中 SIF 為一個 i×j 的二維全域陣列，“i”代表第 i 個地標，“j”則為第 j 個方位，陣列的存放方式則是第 i 個地標對應第 j 個方位，陣列裡的結構為對應地標方位的淹水總面積(totalArea)、淹水總距離（totalDist）和相交次數（count）。其中，“j”我們定義為八個方

位，並用數字代表八個方位。每一個淹水區塊 *f* 則記錄其 *x*、*y* 座標、area 和 MBR。LmtotalArea 為一個大小為 *i* 的一維陣列，“*i*”代表地標的總數，陣列裡的結構為每個地標的編號（LmID）和總淹水面積（area）。

演算法名稱：FindNLm

輸入：R_{vc}：Voronoi cell 的最小邊界矩形建立的 R-Tree 索引

FCList：淹水區塊序列

numLm：地標的總個數

輸出：依據淹水嚴重的情況輸出地標編號

變數：dir：方位

FindNLm（R_{vc} , *FCList*）

```
L01  Initialize  SIF, LmtotalArea
L02  for each  f ∈ FCList  do
L03    if  f.MBR  intersects  Rvc.rootnode.VC_MBR  then
L04      PN（Rvc.rootnode, f）
L05  i←0
L06  while  i < numLm  do
L07    LmtotalArea[i].LmID←i
L08    for each  direction  j  do
L09      LmtotalArea[i].area←LmtotalArea[i].area+ SIF[i][j].totalArea
L10    i++
L11  Quicksort（LmtotalArea）
L12  i←0
L13  while  i<numLm && LmtotalArea[i].area > 0 do
L14    Output  LmtotalArea[i]. LmID  LmtotalArea[i].area
L15  k←0
L16  for  k< numLm  do
L17    for each  direction  j  do
L18      if  SIF[k][j].count>0  then
L19        Output  k  j  SIF[k][j].totalDist/ SIF[k][j].count  SIF[k][j].totalArea
L20    k++
```

```
PN（N, f）
L01  if  N  is a leaf node then
L02   for each e∈N  do
L03    if  f.MBR  intersects  e.VC_ MBR  then
L04     dir←getDir（f.x, f.y, e.x, e.y）
L05     SIF[e.LmID][dir].totalDist←SIF[e.LmID][dir].totalDist+getDist（f.x, f.y, e.x, e.y）
L06     SIF[e.LmID][dir].totalArea←SIF[e.LmID][dir].totalArea +f.area
L07     SIF[e.LmID][dir].count++
L08  else
L09   if  f.MBR  intersects  N.VC_ MBR  then
L10     for c ∈ N.childnode
L11       PN（c, f ）
```

圖 3-6：FindNLm 演算法

完整的 FindNLm 演算法如圖 3-6 所示，首先我們初始化 SIF 和 LmtotalArea 兩個陣列，將其內的所有變數都先設為 0，接著，我們判斷每個淹水區塊是否與 R_{vc} 所代表的全部區塊有相交，如果有的話，我們呼叫副程式 PN（Processing Node）演算法做進一步的處理。PN 演算法定義兩個參數，其中 N 代表索引中的某一節點，而 f 則代表某一個淹水區塊。PN 演算法為一個遞迴程序，如果節點 N 為一個 leaf node（第 1 行至第 7 行），則針對該節點中的每筆 entry，我們判斷淹水區塊 f 是否落入該 entry 所代表的空間範圍，如果有的話，便用 dir 變數記錄該淹水區塊與 entry 對應地標的方位，並將淹水總面積、淹水總距離和相交次數存入 SIF 陣列。另一方面，如果節點 *N* 為一個 internal node（第 8 行至第 11 行），我們先行判斷淹水區塊 *f* 和節點 *N* 代表的空間範圍是否相交，若有的話，則遞迴呼叫 PN 演算法處理 *N* 的子節點和淹水區塊 *f*。最後，由 SIF 陣列統計每個地標於每個方位的資料，LmtotalArea 陣列統計每個地標不分方向的總淹水面積，Quicksort 函數利用快速排序法將 LmtotalArea 陣列依地標的淹水總面績排序（第 11 行），並由主程式依序輸出對應的地標編號和總淹水面積（第 14 行），並輸出每個地標

附近淹水區塊的方位與距離（第 19 行）。

延續圖 3-3 的範例，在主程式 SIF 陣列中，利用 LmtotalArea 陣列統計各地標的淹水總面積排序後輸出，且依序輸出每個地標，對應的每個方位，相交次數大於 0 的，即輸出地標編號、方位、平均距離（淹水總距離除以相交次數）和淹水總面積，其中 m_1 北方的平均距離為 m_1 至 f_1 與 f_2 的總距離（$2 * \sqrt{2^2 + 0.5^2}$）除以 2。以 0-7 分別代表北、東北、東、東南、南、西南、西與西北八個方位，我們算得以下的結果：

LmID	LmtotalArea
m_1	3
m_2	2
m_3	2

k	*j*	avgDis	totalArea
m_1	0(北)	2.06	2
m_1	1(東北)	4.71	1
m_2	6(西)	2	1
m_2	0(北)	5.02	1
m_3	2(東)	4	1
m_3	4(南)	3	1

圖 3-7：FindNLm 演算法輸出結果

3.3 FindFCGroup 演算法

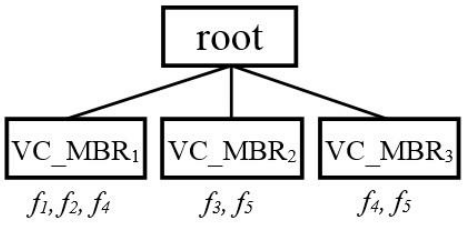


圖 3-8：淹水區塊的走訪

注意到，由於 R-tree 本身的特性，每個淹水區塊可能走訪 R-tree（從 root 到 leaf）多次。第一個狀況是雖然每個地標的 Voronoi cell 彼此間不相交，但是對應的 MBR 可能會重疊，如圖 3-3 的 f_4 淹水區塊正好落在 VC_MBR_1 和 VC_MBR_3 中，所以會走訪到兩個 leaf，如圖 3-8 所示。另外，也可能因為淹水區塊面積大或所在地點的關係，直接與多個 VC 相交，如 f_5 的位置若往東方移一格，則會落入 VC_1 和 VC_3 中。為了減少搜尋 R-tree 的次數，我們利用淹水區塊地理上經常出現的群聚現象，將淹水區塊組成一群，再進行空間連接，以加快查詢的速度。由於 TwoFD 水理模組產生淹水區塊時，由 *y* 軸排序，再由 *x* 軸排序，所以我們將鄰近淹水區塊的中心點、MBR 和面積做合併的動作，合併的方法為循序比較淹水區塊中心點彼此間 *x* 與 *y* 座標，若淹水區塊彼此間座標差距小於最小合併距離門檻（Merged threshold），便將鄰近的淹水區塊合併成一個新的淹水區塊，新的淹水區塊包括合併過後的 MBR、中心點和面積，最後將合併過後的淹水區塊做為 FindNLm 的輸出與 R_{vc} 作空間連接，合併淹水區塊對應的演算法如下：

演算法名稱：FindFCGroup 輸入：FCList（簡稱 *f* ）、Merged threshold 輸出：*mf*：合併過後的淹水區塊陣列 變數：area：淹水區塊面積 　　MergedMBR：合併兩個淹水區塊的方法

```
L01   i←0;   j←0;
L02   while（i <| f |）do
L03       totalX← f [i].x;   totalY← f [i].y;   totalarea← f [i].area;   count←1;
L04       while（| f [i].x- f [i+1].x| <Merged threshold ）&&
L05             （| f [i].y- f [i+1].y | <Merged threshold ）  do
L06           f [i+1].MBR←MergedMBR(f [i], f [i+1])
L07           totalX←totalX + f [i+1].x
L08           totalY← totalY + f [i+1].y
```

```
L09          totalArea←totalArea+ f [i+1].area
L10          count++
L11          i ++
L12      mf [ j].MBR ← f [i-1].MBR
L13      mf [ j].x ←totalX/count
L14      mf [ j].y ←totalY/count
L15      mf [ j].area←totalArea
L16      j++
L18   return mf
```

MergedMBR(f_1, f_2)

```
L01  if（f1.x1≦ f2.x1） then  MBR.x1←f1.x1  else  MBR.x1←f2.x1
L02  if（f1.y1≦ f2.y1） then  MBR.y1←f1.y1  else  MBR.y1←f2.y1
L03  if（f1.x2≧ f2.x2） then  MBR.x2←f1.x2  else  MBR.x2←f2.x2
L04  if（f1.y2≦ f2.y2） then  MBR.y2←f1.y2  else  MBR.y2←f2.y2
L05  return MBR
```

圖 3-9：FindFCGroup 演算法

完整的 FindFCGroup 演算法如圖 3-9 所示，我們判斷鄰近淹水區塊 *x* 與 *y* 座標是否小於 Merged threshold，如果有的話，我們呼叫副程式 MergedMBR 合併兩個淹水區塊的 MBR（第 6 行），並利用 totalX、totalY、totalarea 及 count 變數記錄合併過後的資訊（第 7 行至第 10 行）。最後，我們用 *mf* 陣列記錄合併過後的淹水區塊（第 12 行至第 15 行）。

考慮圖 3-3，假設最小合併距離門檻為 1，由於 f_1 和 f_2 符合條件，所以我們將此二淹水區塊合併，接著，由於 f_2 與 f_3 的距離過遠，且 f_3 與 f_4、f_4 與 f_5 的距離也過遠，所以我們把 f_1 與 f_2 視為一群，f_3、f_4、f_5 則不變，如圖 3-10 所示。將 *mf* 取代 FCList 做為 FindNLm 的輸入，可讓淹水區塊序列尋訪 R_{vc} 的次數大幅變少。至於在這個例子裡，結果如圖 3-11，相較於原做法的結果（圖 3-7），合併過後的 f_1 與 f_2，由於合併後為新的中心點，所以只有在地標

為 m_1 時，平均距離相差了一點。

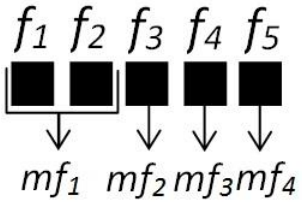


圖 3-10：合併過後的 mf

LmID	LmtotalArea
m_1	3
m_2	2
m_3	2

k	*j*	avgDis	totalArea
m_1	0(北)	2	2
m_1	1(東北)	4.71	1
m_2	6(西)	2	1
m_2	0(北)	5.02	1
m_3	2(東)	4	1
m_3	4(南)	3	1

圖 3-11：*mf* 取代 FCList 後的輸出結果

第 4 章C2L 方法

如前所示，在 FindNLm 演算法中，淹水區塊的 MBR 至地標的距離可能有過遠的情況，如圖 3-3 的 f_5，為了剔除這種情況，我們讓 threshold 代表最小距離門檻，將影響力較低的淹水區塊剔除。在本章中，我們介紹 C2L（Cell to Landmark）方法，此方法除了事先針對每個演算區塊找出距離小於門檻值的地標，也會改成以「點」為空間連接的基準。

4.1 前處理階段

C2L 主要分為兩個部分，前處理階段的 MakeC2L 演算法與線上查詢階段的 FindLm 演算法，首先，在前處理階段先找出所有演算區塊到地標的距離和方位，並篩選出距離小於門檻值的地標，若淹水區塊與地標之間存在著障礙物，為了增加距離的準確性，我們會呼叫 RDWO 演算法（Reckon Distance with Obstacle）針對每個距離淹水區塊小於門檻值的地標，算出具障礙物的距離，此部分將於附錄 A 討論。接著，在線上查詢前，利用演算區塊的 *x* 座標以及 *y* 座標建立 B-tree 索引，之後利用淹水區塊座標以供線上查詢。

此階段所找出距離小於門檻值的資料會存放至 cell2Lm，其中 cell2Lm 為一個 $i \times j$ 的二維陣列，"*i*"代表演算區塊的 *x* 座標，"*j*"代表演算區塊的 *y* 座標，陣列內存放的是演算區塊 *x*、*y* 座標對應的相關資訊，如圖 4-1(a)，陣列裡存放由 entry 串成的 list，而每個 entry 的結構為演算區塊對應的地標編號、面積、距離和方位。此演算法的輸入如下，cellList 為演算區塊陣列，每個 cell 記錄其中心點 *x*、*y* 座標、面積（area）和 MBR（以 x_1, y_1, x_2, y_2 記錄了 *x*、y 軸的最小與最大值），lmList 為地標陣列，每個 lm 記錄其 *x*、*y* 座標及地標編號（lmID），threshold 變數為最小距離門檻，ob 為障礙物集合。其中，getMDist 函數為了讓淹水區塊至地標以點計算的距離更準確，為淹水區塊至地標的距離扣掉淹水區塊 MBR 的外接圓半

徑，如圖 4-2 顯示地標 lm 和一淹水區塊 MBR 的外接圓，可看到外接圓半徑 r 為斜對角兩點距離的一半，所以實際距離 MDist 為：

MDist= getDist（*o.x, o.y, lm.x, lm.y*）-sqrt（（x_1-x_2）^{2}+（y_1 - y_2）^{2}）/2 (4)

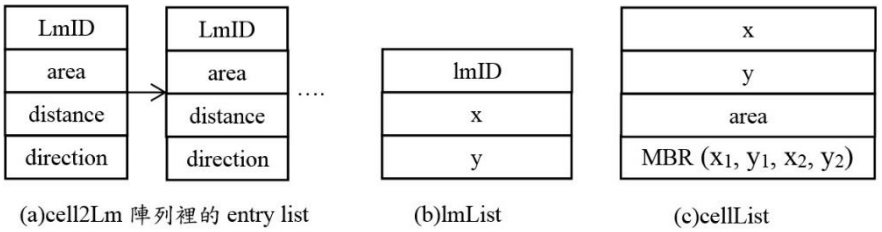


圖 4-1：cell2Lm 陣列、演算區塊陣列與地標陣列裡每一個 entry 的結構

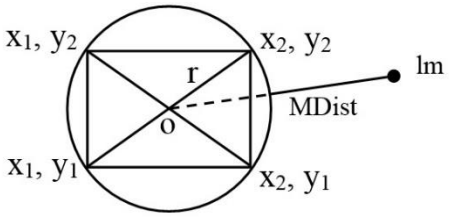


圖 4-2：淹水區塊 MBR 的外接圓半徑

演算法名稱：MakeC2L 輸入：cellList, lmList threshold：最小距離門檻 ob：障礙物集合 輸出：cell2Lm：所有演算區塊對於地標距離小於門檻的地標編號、距離、演算區塊面積和方位 變數：*count* 陣列：記錄每個演算區塊小於最小距離門檻值地標的個數
L01 **for each** *c*∈ cellList **do** L02 count←0 L03 **for each** *l* ∈ lmList **do** L04 dir←getDir（*c.x, c.y, l.x, l.y*） L05 dist←getMDist（*c, l*）

```
L06          if  dist ≦ threshold  then
L07             entry.LmID←l.lmID;
L08             entry.area←c.area;
L09             entry.distance←dist;
L10             entry.direction←dir;
L11             count++;
L12             cell2Lm[c.x][c.y].append(entry) ;
L13          RDWO（l.x, l,y, cell2Lm, count , ob）
L14   return cell2Lm
```

```
getMDist（c, l）
```

```
L01   MDist←0
L02   MDist ←getDist（c.x, c.y, l.x, l.y）- sqrt（(（c.x1 - c.x2）2+（c.y1 - c.y2）2）/2
L03   return   MDist
```

圖 4-3：MakeC2L 演算法

完整的 MakeC2L 演算法如圖 4-3 所示，當演算區塊與地標的距離小於最小距離門檻，則將對應的資料存入 cell2Lm（第 1 行至第 12 行），接著再算出每個演算區塊至地標具障礙物的距離（第 13 行），其中，RDWO 模組的輸入為 lmList 的 *x*、*y* 座標、cell2Lm、ob 和 count 變數。最後，輸出 cell2Lm（第 14 行）。

延續我們的範例（圖 2-1），若 *threshold* 為無限大，且只考慮有淹水的 5 個淹水區塊，所有淹水區塊的面積為 1、外接圓半徑均為 0.7（sqrt（1^2+1^2）除以 2），而 f_1 至 m_1 距離為 sqrt ($(2.5 - 3)^2$ + ($4.5 - 2.5)^2$) - 0.7 =1.36，MakeC2L 演算法所得到的結果如圖 4-4(a)，threshold 設為 3 時，剔除過後的結果如圖 4-4(b)。

x	*y*	*LmID*	*area*	*distance*	*direction*
2.5	4.5	m_1（3, 2.5）	1	1.36	北
		m_2（10.5, 4.5）	1	7.3	西
		m_3（5.5, 9.5）	1	5.13	西南
3.5	4.5	m_1	1	1.36	北
		m_2	1	6.3	西
		m_3	1	4.68	西南
8.5	4.5	m_1	1	5.15	東北
		m_2	1	1.3	西
		m_3	1	5.13	東南
5.5	6.5	m_1	1	4.01	東北
		m_2	1	4.68	西北
		m_3	1	2.3	南
9.5	9.5	m_1	1	8.85	東北
		m_2	1	4.39	北
		m_3	1	3.3	東

(a)

x	*y*	*LmID*	*area*	*distance*	*direction*
2.5	4.5	m_1	1	1.36	北
3.5	4.5	m_1	1	1.36	北
8.5	4.5	m_2	1	1.3	西
5.5	6.5	m_3	1	2.3	南

(b)

圖 4-4：MakeC2L 演算法範例

4.2 FindLm 演算法

最後 FindLm 演算法再利用 MakeC2L 所輸出的每個演算區塊距離小於 threshold 的淹水區塊座標、地標編號、距離和方位，當淹水區塊產生時，利用淹水區塊座標找出受影響的地標。

對應的 FindLm 演算法如圖 4-5 所示，TwoFD 水理模組將淹水區塊產生時，利用 *x* 與 *y* 座標在 MakeC2L 演算法輸出的 cell2Lm 陣列搜尋對應的地標編號、

距離、方位與淹水面積並存入 SIF 陣列，其中，SIF 陣列的結構同 3-2 節所述，取得 SIF 資料後所處理的步驟則如圖 3-6 的第 5 行至第 20 行。

演算法名稱：FindLm 輸入：cell2Lm, FCList 輸出：基於地標的警示訊息
L01 Initialize SIF, LmtotalArea L02 **for each** *f* ∈ FCList **do** L03 **for each** entry *e* in cell2Lm[*f.x*][*f.y*] **do** L04 dir←*e*.direction L05 SIF[*e*.LmID][dir].totaDist←SIF[*e*.LmID][dir].totaDist +*e*.distance L06 SIF[*e*.LmID][dir].totalArea←SIF[*e*.LmID][dir].totalArea +*e*.area L07 SIF[*e*.LmID][dir].count++ L08 以下同圖 3-6 演算法 FindNLm 的 L05-L20

圖 4-5：FindLm 演算法

FindLm 演算法的結果如圖 4-6，其為將圖 4-4(b)根據 LmID 和 direction 統計過後的結果，4-6(a)對應到公式(1)，4-6(b)則存詳細資料，將其結果與圖 3-7 比較，圖 4-6 (a)針對地標編號的排序大致相同，但在圖 4-6 (b)的詳細資料部分，地標 m_1 的東北方、地標 m_2 的北方和地標 m_3 的東方，由於距離大於 threshold(設為 3)，所以被剃除了。

LmID	LmtotalArea
m_1	2
m_2	1
m_3	1

(a)

k	*l*	avgDis	totalArea
m_1	北	1.36	2
m_2	西	1.3	1
m_3	南	2.3	1

(b)

圖 4-6：FindLm 演算法輸出結果

第 5 章實驗

在本章，我們進行實驗來比較所提出方法在線上查詢效率上的差異。首先，先說明我們進行實驗的環境，我們以個人電腦作為實驗的環境，其 CPU 為四核心的 Intel Core 2 Q8300，核心時脈為 2.5GHz，而記憶體為 4GB，所採用的作業系統為 Windows 7 企業版 SP1。

在本章中，3.1 節的 MakeVC 搭配 3.2 節的 FindNLm 稱為方法「VC」，若是以 3.3 節的 FindFCGroup 搭配 3.2 節的 FindNLm，則稱為「VC+」，第四章的方法則稱為「C2L」。三個方法在比較效能時，均輸出如第二章的(1)式，將地標依淹水面積排序。

5.1 實驗採用之實作方法

前處理階段皆以 Microsoft Visual C++ 2010 進行實作，線上查詢的 FindNLm 演算法和 FindLm 演算法，則使用 Microsoft Visual C# 2010 與 Mysql Workbench 6.0。資料部份是儲存於 MySQL 中，表格定義如圖 5-1，其中 lm 表格對應到如圖 3-4 的 lmList，allcell 表格對應到如圖 4-1(c)的 cellList。表格 lm 的主鍵為地標編號 landmarkID，*x*、*y* 為地標座標，而 vc_mbr 為地標的 MBR。allcell 表格的 *x*、*y* 座標為演算區塊座標，area 和 mbr 分別為演算區塊面積與最小邊界矩形。至於 twofd 表格和 cell2lm 則是修正過後的 FCList 和 cell2lm 陣列。表格 twofd 的主鍵為淹水區塊 *x* 與 *y* 座標，其面積可透過 allcell 表格取得。表格 cell2lm 的主鍵為演算區塊 *x* 與 *y* 座標及地標編號 landmarkID，而 direction 及 distance 則為演算區塊對於地標的方位與距離。

Table: lm

Columns:	
landmarkID	int(11) PK
name	varchar(65)
x	double
y	double
vc_mbr	geometry

Table: allcell

Columns:	
x	double PK
y	double PK
area	double
mbr	geometry

Table: twofd

Columns:	
x	double PK
y	double PK

Table: cell2lm

Columns:	
x	double PK
y	double PK
landmarkID	int(11) PK
direction	varchar(45)
distance	double

圖 5-1：表格的 schema

在 VC 方法中，我們在 MakeVC 演算法中必須建立 R-tree 索引，而 MySQL 在空間索引的 geometric 型態支援 R-Tree 索引，而且必須在 MyISAM 資料庫引擎，其使用的 R-Tree 分裂方法為 Guttman 的 Quadratic-Split 方法[Gu84]。我們的空間索引是建立在 lm 表格的 vc_mbr 欄位，建立 R-Tree 索引的語法如下：

```
CREATE SPATIAL INDEX Rvc ON lm (vc_mbr);
```

前處理階段根據各地標產生的地標編號、*x*、*y* 座標和地標的 MBR 時，便匯入 MySQL 資料庫中的 lm 表格。當 TwoFD 水理模組產生淹水區塊則匯入圖 5-1 所示之 twofd 表格，其 *x*、*y* 為淹水區塊座標。至於圖 3-5 的 FindNLm 演算法是利用 SQL 語法查詢淹水區塊的最小邊界矩形影響哪個地標，利用 *x* 與 *y* 座標將 twofd 表格合併 allcell 表格，找出淹水區塊對應的 MBR 與面積後，再將淹水區塊的 MBR 與地標的 MBR 作空間連接，並輸出(1)式，對應的 SQL 如下：

```
SELECT    lm.landmarkID, SUM(allcell.area)  AS  totalarea
FROM      allcell  JOIN  twofd  ON  ((allcell.x = twofd.x) AND (allcell.y =twofd.y))
          JOIN  lm  ON  MBRIntersects(allcell.mbr, lm.vc_mbr)
GROUP BY  landmarkID
ORDER BY  totalArea  DESC
```

至於在 MakeC2L 演算法中，我們將產生的 cell2lm 陣列匯入 cell2lm 表格，為了在後續的 FindLm 能快速地以淹水區塊的 *x*、*y* 座標找到對應的資料，我們以該座標建立 B-tree。在此做法中，我們採用 MySQL 的 innodb 資料庫引擎，建立 B-tree 索引的語法如下：

```
CREATE INDEX Bc ON cell2Lm(x,y)  USING BTREE;
```

接下來，我們描述如何根據 twofd 表格記錄的淹水區塊，輸出針對(1)式對應的 SQL 語法：

```
SELECT  cell2lm.landmarkID, SUM(allcell.area)  AS  totalarea
FROM    cell2lm  JOIN  twofd  ON ((cell2lm.x = twofd.x) AND (cell2lm.y
        =twofd.y))
        JOIN  allcell  ON  ((allcell.x = twofd.x) AND (allcell.y = twofd.y))
GROUP BY  landmarkID
ORDER BY  totalArea  DESC
```

此查詢句利用 twofd 表格的淹水區塊經緯度座標，連接至 cell2lm 表格，輸出這些淹水區塊對應的地標編號以及淹水總面積，並依據淹水最嚴重的優先輸出。

以下補充輸出詳細資料的 SQL，但因為做法二的距離等資料是事先計算的，而做法一是 online 計算，所以為了公平起見，在實驗時並不計算執行此 SQL 的時間。

FindNLm 演算法：

```
SELECT   landmarkID  AS  LmID, getDir(allcell.x,allcell.y,lm.x,lm.y)  AS
         direction, getDist(allcell.x,allcell.y,lm.x,lm.y)  AS  distance,area
FROM     allcell  JOIN  twofd  ON  (allcell.x=twofd.x)
JOIN     lm  ON  MBRIntersects(allcell.mbr,lm.vc_mbr)
```

函式 getDir()的 SQL 如下(參考 2.3 節的公式(3))，其中，lm 為地標、allcelltwofd 為淹水區塊，而*PI()/180 則為將經緯度轉換成弧度(radians)：

```
(atan2((sin(((lm.y-allcelltwofd.y)*PI()/180)) *
cos((lm.x*PI()/180))),(cos((allcelltwofd.x*PI()/180))*sin((lm.x*PI()/180)) -
sin((allcelltwofd.x*PI()/180))*cos((lm.x*PI()/180))*cos(((lm.y-allcelltwofd.y)*PI
()/180)))))*180/PI()
```

函式 getDist()的 SQL 如下(參考 2.3 節的公式(2))，其中，SQRT 為開根號、POW 為回傳指數運算結果：

```
2 * 6378* ATAN2 (SQRT(POW(SIN(PI() * (allcelltwofd.x - lm.x)/360), 2) +
COS(PI() * allcelltwofd.x / 180)* COS(lm.x * PI() / 180) * POW(SIN(PI() *
(allcelltwofd.y - lm.y) / 360), 2)),1-(SQRT(POW(SIN(PI() * (allcelltwofd.x -
lm.x)/360), 2) + COS(PI() * allcelltwofd.x / 180)* COS(lm.x * PI() / 180) *
POW(SIN(PI() * (allcelltwofd.y - lm.y) / 360), 2))))
```

FindLm 演算法：

```
SELECT   cell2lm.landmarkID, cell2lm.distance, cell2lm.direction,
         sum(allcell.area)  AS  totalarea
FROM     cell2lm  JOIN  twofd  ON  ((cell2lm.x = twofd.x)  AND  (cell2lm.y
         =twofd.y))
```

JOIN allcell **ON** ((allcell.x = twofd.x) **AND** (allcell.y = twofd.y))
GROUP BY landmarkID,direction
ORDER BY totalarea **DESC**

5.2 資料集

我們測試所使用的資料集包括根據林邊鄉淹水情況的真實資料與人工產生出來的資料。首先，林邊鄉的演算區域的經度範圍為 120°29'34"- 120°40'48"，緯度範圍為 22°24'24"- 22°32'30"，真實資料為 TwoFD 水理模組將林邊鄉所切割成的 31753 個演算區塊，以及根據莫拉克颱風即將登陸時，所預測的 496 個淹水區塊，而該區域的地標有 59 個。

人工所產生出來的資料為把經度範圍為 120°-121°，緯度範圍為 23°-24°的二維平面切割成 1000*1000 個演算區塊，一個演算區塊 x 軸距離為 103 公尺，而 y 軸距離為 111 公尺，且根據隨機分佈、高斯分佈（μ=0, σ^2=1.5）與群集分佈不重複的產生地標與淹水區塊，分佈的情況如圖 5-2。其中，群集分佈為均勻隨機的選擇 10 個地點，將剩餘的地點利用高斯分佈（μ=0, σ^2=0.5）平均的產生地標與淹水區塊。這三種分佈代表的地理意義如下：隨機分佈中的地標與淹水區塊代表著地標與淹水區塊均勻的分佈於地圖中；而高斯分佈的地標可能代表著地標密集的位於市中心，淹水區塊可能代表淹水時，市中心為低窪的盆地地形，造成市中心淹水嚴重；而群聚分佈的地標可能代表著一個城市中，地標密集的分佈於數個市中心區域，淹水區塊可能代表著數個低窪地形的淹水情況。

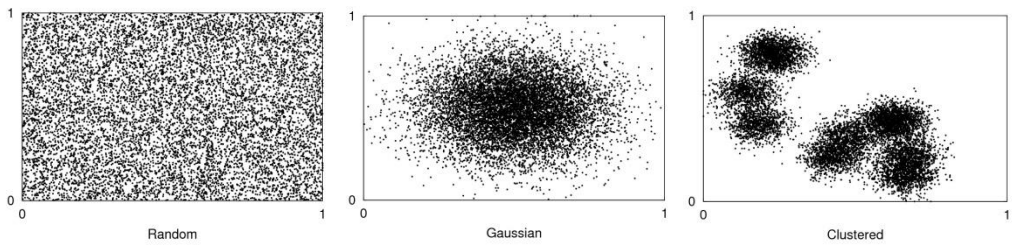


圖 5-2：資料集分佈的情況

所有資料集之各項參數如表 5-1。我們測量執行時間的方式為：從匯入淹水區塊至資料庫開始計算至依淹水總面積排序輸出，連續執行十次，量取系統第二次至第十次執行所花費時間的平均值作為比較。

表 5-1：資料集之各項參數

Parameter	**Setting**	
Data distribution	Uniform, Gaussian, Clustered	Real
Cell point	1000*1000	31753
Landmark point	10, 100, 1000	59
Flooded cell point	100, 1000, 10000	496
Flooded cell size(km^2)	0.01, 0.1, 1	1-10
Threshold(km)	1	1
Merged threshold(km)	0.2	0.16

5.3 真實資料之實驗

首先，為了分辨三個做法在結果輸出上的不同，我們將其結果和人為判別的結果比較，人為判別的結果為三人依照地標附近的淹水嚴重性排序，若排名有出入，則以投票的方式表決決定。在總共 59 個地標中，VC、VC+和 C2L 的輸出結果為不太相同的前十四名，我們在三個方法輸出結果的最後部分，補上彼此未出現的兩個地標，所以比較的地標總共十六個。比較的方式，為利用 Kendall 等級相關係數（Kendall rank correlation coefficient，肯德爾等級相關係數）[Ke55]，將人為判別的地標排名與三個方法各自的地標排名算出 Kendall's tau 係數。此係數是一種計算次序變數之間相關係數的方法，其值介於-1 與 1 之間，如果變數間的相關係數的值為 0 時，稱為『零相關』，相關係數的值大於 0 時，稱為『正相關』，相關係數的值小於 0 時，稱為『負相關』。設樣本量為 n，考察兩個變數 X 和 Y 之間的相關關係，X 和 Y 的取值記為 $X_i, Y_i, i=1,\ldots,n$。所有像$(X_i,Y_i), i, j=1,\ldots, n$ 對的個數為 $n(n-1)/2$。如果對於 X_j-X_i 和 Y_j-Y_i 有相同的順序則稱為同序數對，而對於 X_j-X_i 和 Y_j-Y_i 有相反的順序則稱為逆序數對。總體同序數對的個數記為 n_c，逆序數對的個數記為 n_d，則 Kendall's tau 係數的定義為：

$$\tau = \frac{n_c - n_d}{n(n-1)/2}$$

如表 5-2，計算的方法為利用人為判別的排名的前十六名分別與 VC、VC+和 C2L 方法的前十六名比較。算得 τ 值為分別為 0.283、0.3 和 0.4，其中 C2L 的算式如下：首先將人為判別視為正確答案，所以尋找 C2L 結果對於人為判別結果的同序數對，首先針對 C2L 的第一順位地標為例，X_1(地標 12)對於 X_i(所有地標)和人為判別結果 Y_1(地標 12)對於 Y_i(所有地標)都是同序數對，所以算得相同的同序數對個數為 15，接著看到 C2L 的 X_3(地標 6)與 X_i(地標 5、地標 56、地標 55、地標 57、地標 23、地標 53)和人為判別的 Y_3(地標 6)與 Y_i(地標 5、地標 56、地標 55、地標 57、地標 23、地標 53)為同序數對，所以同序數對個數為 6，逆序數對為 13-6=7，其餘的依此類推，算得 n_c=15+14+6+12+3+8+3+2+6+5+2+3+3+2+0+0=84，且 n_d=0+0+7+0+8+2+6+6+1+1+3+1+0+0+1+0=36。最後算得 τ 值為=84-36/(16*15/2)=0.4。

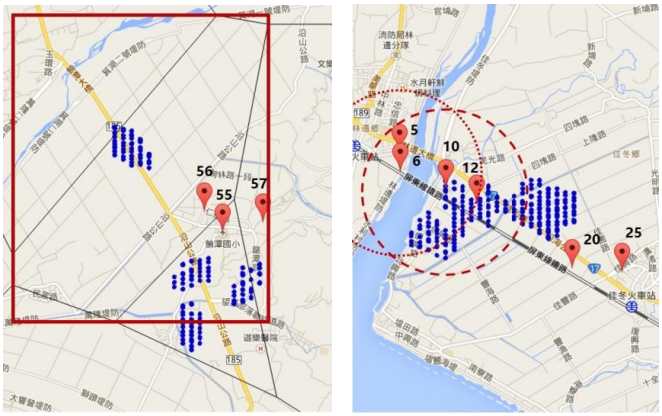
由於 C2L 的值最大，代表 C2L 方法的結果，也就是利用門檻值決定淹水區塊影響地標的方法與人為判別的結果相關性，較 VC 和 VC+方法高。

表 5-2：林邊鄉的淹水嚴重性排名

Rank	**VC**	**VC+**	**C2L**	**Human**
1	12	12	12	12
2	56	56	10	10
3	34	42	6	20
4	42	34	20	25
5	10	10	55	42
6	55	19	19	19
7	20	55	56	18
8	19	20	57	34
9	18	18	42	24
10	53	53	18	6
11	57	57	5	5
12	6	6	34	56
13	24	24	25	55
14	23	23	24	57
15	25	25	53	23
16	5	5	23	53

以下解釋為何不同方法的排名會有所出入。林邊鄉淹水的部分情況如圖 5-3，其中，排名順序為第二名時，如圖 5-3 (a)所示，圖中的粗矩形對應到

VC 與 VC+方法的地標 56 的 Voronoi cell 的 MBR，由於涵蓋的面積很大，包含很多淹水區塊，所以排名較前面。另一方面，參考圖 5-3 (b)，圖中圓圈的半徑即為最小距離門檻，C2L 方法的地標 10，對應到圖中的虛線圓，由於在 threshold 的設定上讓它符合門檻的總淹水面積較大，所以有較好的排名。而在 C2L 方法和人為判別的第三名上，由於地標 6 在圖中明顯已經越過河川，對應到圖中的點圓，所以造成的嚴重性在人為判別上並不嚴重。如果剔除過河的地標 5 與地標 6，算得人為判別與 C2L 方法的 Kendall's tau 係數為 0.49。所以得知，剔除過河的地標後，C2L 方法與人為判別的相關性提昇了不少，此種狀況未來可以再以條件限制將其排除。



(a)地標 VC 的 MBR　　(b)C2L 門檻的外接圓

圖 5-3：不同方法的淹水情況

另一方面，人為判別傾向於分區排名，而系統沒有此限制，若將淹水區塊分成三個區域，如圖 5-4，分別為[5, 6, 10, 12, 20, 23, 24, 25]、[18, 19, 23, 24, 34, 42]、[55, 56, 57]，其中，地標 23 與地標 24 同時為兩個區域的地標。將此三個區域分別算出人為判別與 C2L 方法的 Kendall's tau 係數，分別是 0.64、0.86 和 0.33，明顯看出經分區後，C2L 方法與人為判別的相關性明顯提昇，但在圖 5-4 的右邊區域由於 X 與 Y 的變數只有三筆，所以算得的相關性較低。

表 5-3：分區後的淹水嚴重性排名

Rank	C2L	Human
1	12	12
2	10	10
3	6	20
4	20	25
5	5	24
6	25	6
7	24	5
8	23	23

C2L	Human
19	42
42	19
18	18
34	34
24	24
23	23

C2L	Human
55	56
56	55
57	57

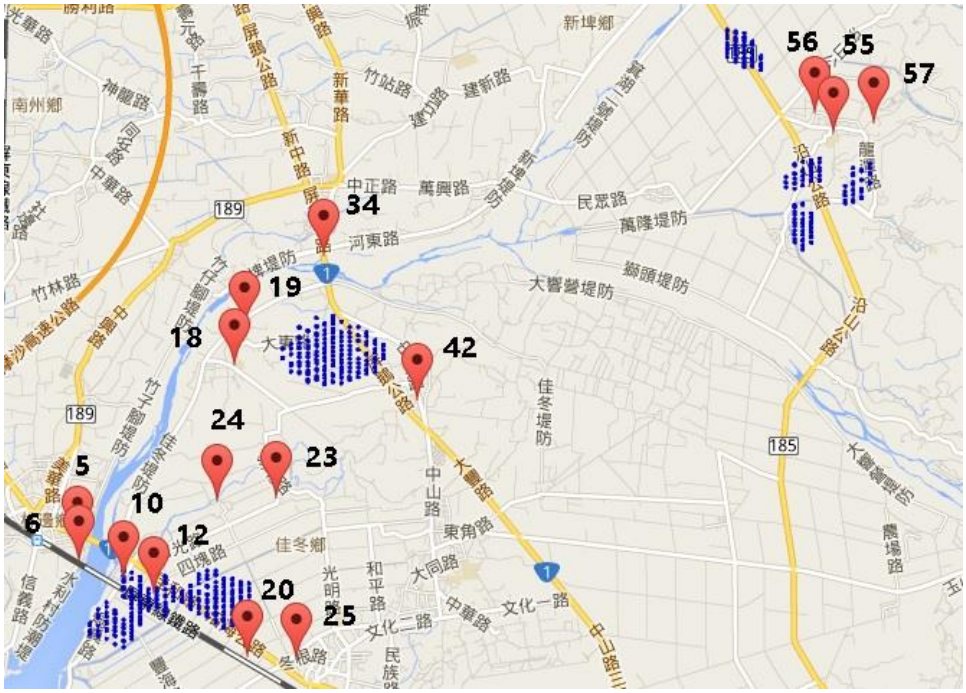


圖 5-4：林邊鄉的淹水情況

其次，有關效率的實驗結果如圖 5-5，首先看到右圖，C2L 的 B-tree 索引約為 10,240kb，而 VC 和 VC+的 R-tree 索引約為 9kb，大約為 1000 倍，原因除了內部結構略有不同，主要原因在於用來建 B-tree 的表格 cell2lm 有 29373 筆，而用來建 R-tree 的表格 lm 只有 59 筆資料列，相距約 500 倍。雖然 B-tree 大很多，但是由於其走訪的效率比 R-tree 好，所以在左圖中看出其效率比 VC 好。另一方面，VC+因為降低搜尋 R-tree 的次數，執行時間小於 VC，甚至 C2L。

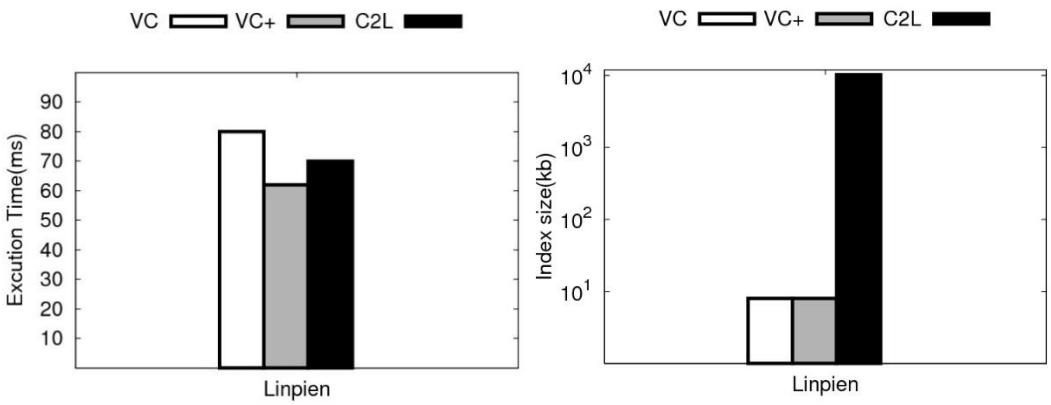


圖 5-5：真實資料集

5.4 隨機分佈之實驗

以下的實驗我們著重於改變資料集的參數，我們分別改變淹水區塊的數量和地標數量。

(1)這次實驗，如圖 5-6，我們探討改變淹水區塊的數量的影響，landmark point 固定為 100、flooded cell size 為 0.01，隨著淹水區塊的增加，三個方法的執行時間皆以線性的方式增加，不過以 C2L 的效率最佳，VC+大約比 VC 快 1.3 倍，而 C2L 在效率上比 VC 快 12 倍。VC 和 VC+的 index 大小均為 8Kb，而 C2L 為 10,240Kb。

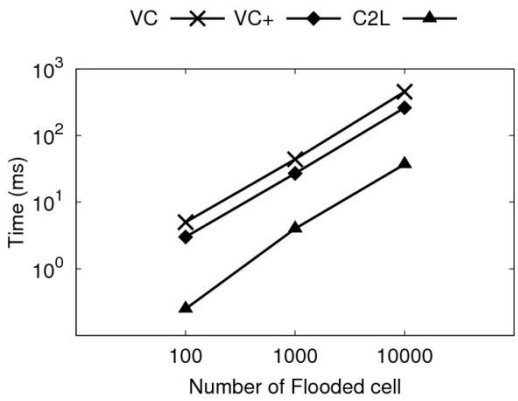
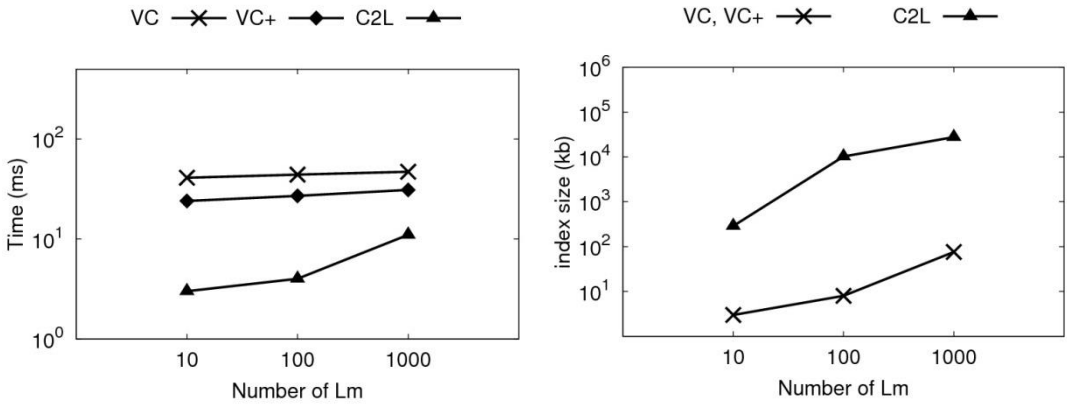


圖 5-6：隨機資料集中改變淹水區塊的數量的影響

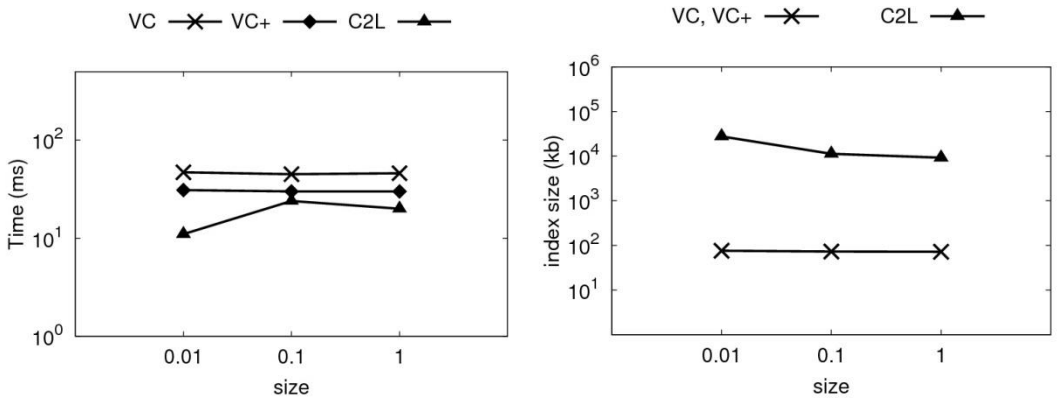
(2)這次實驗，如圖 5-7，我們探討改變地標的數量的影響，flooded cell point 固定為 1000、flooded cell size 為 0.01。VC 和 VC+的 Index size 大小依序為 3kb、8kb 和 76kb，C2L 的 Index size 大小依序為 288 kb、10240 kb 和 27648 kb。C2L 的執行時間隨著 index 增加而增加。接著看到 VC 和 VC+，在地標數量變多時，時間隨著地標數量增加而有些微增加，VC+大約比 VC 快 1.6 倍，而 C2L 在效率上比 VC 快 9 倍。



(a)Time (b)Index size

圖 5-7：隨機資料集中改變地標的數量的影響

(3)這次實驗，如圖 5-8，我們探討改變淹水區塊大小的影響，flooded cell point 固定為 1000、landmark point 固定為 1000。所有實驗的確實數據如表 5-4 所列。size=0.01 時，cell 總個數為 1000*1000；size=0.1 時，cell 總個數為 330*330；size=1 時，cell 總個數為 100*100；



(a)Time (b)Index size

圖 5-8：隨機資料集中改變淹水區塊大小的影響

表 5-4：隨機資料集所有實驗的查詢時間

淹水點數量	VC(ms)	VC+(ms)	C2L(ms)
100	5	3	0.25
1000	44	27	4
10000	454	261	37

地標數量	VC(ms)	VC+(ms)	C2L(ms)
10	41	24	3
100	44	27	4
1000	47	31	11

淹水點大小	VC(ms)	VC+(ms)	C2L(ms)
0.01	47	31	11
0.1	45	30	24
1	46	30	20

5.5 高斯分佈之實驗

接下來的實驗我們改變資料集的分佈，將地標和淹水區塊的分佈為高斯分佈取樣取樣。

(1)這次實驗，如圖 5-9，我們探討改變淹水區塊的數量的影響，landmark point 固定為 100、flooded cell size 為 0.01，執行時間如同隨機資料集隨著淹水區塊增加，執行時間也以線性成長，VC+大約比 VC 快 1.6 倍，而 C2L 在效率上比 VC 快 5 倍。

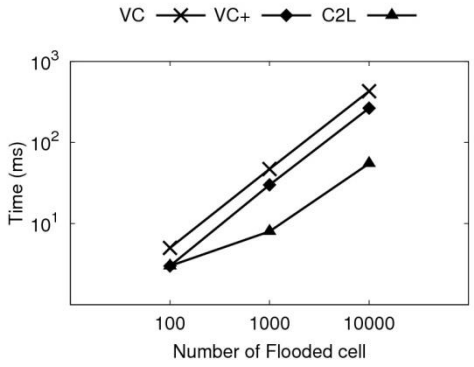
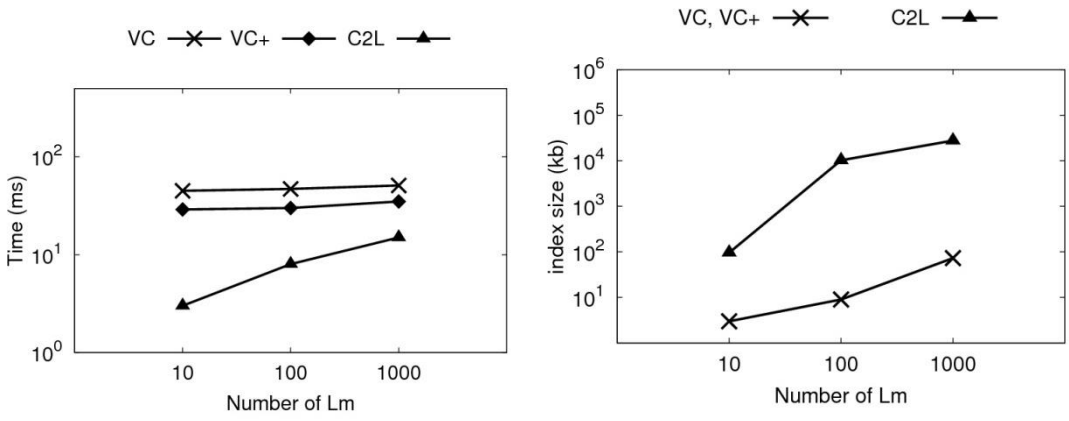


圖 5-9：高斯資料集中改變淹水區塊的數量的影響

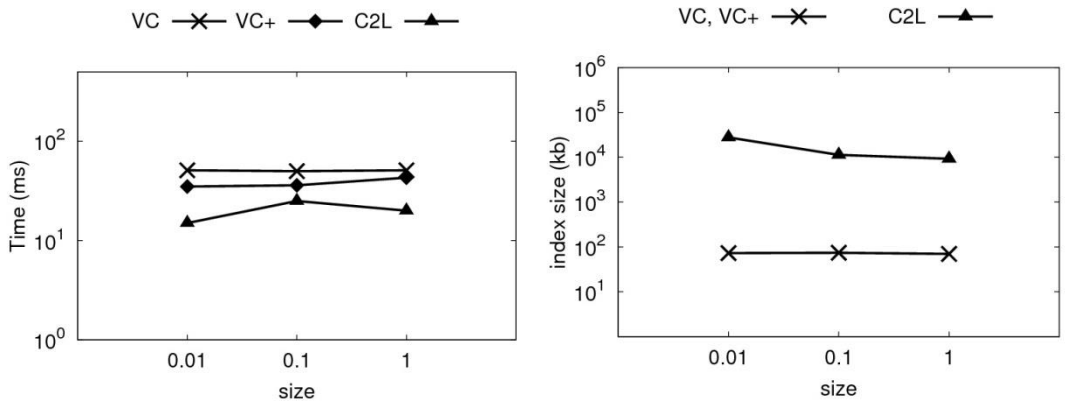
(2)這次實驗，如圖 5-10，我們探討改變地標的數量的影響，flooded cell point 固定為 1000、flooded cell size 為 0.01，VC 和 VC+的 Index size 大小依序為 3kb、9kb 和 73kb，C2L 的 Index size 大小依序為 96 kb、10240 kb 和 27648 kb。其中 VC 及 VC+在地標個數為 1000 時，由於高斯分佈的地標在中央區域較密集，所以 VC 和 VC+的執行時間比隨機資料集有些微增加。而 C2L 的執行時間則隨著 index 增加而增加，VC+大約比 VC 快 1.5 倍，而 C2L 在效率上比 VC 快 8 倍。所有實驗的確實數據如表 5-5 所列。



(a)Time (b)Index size

圖 5-10：高斯資料集中改變地標的數量的影響

(3)這次實驗，如圖 5-11，我們探討改變淹水區塊大小的影響，flooded cell point 固定為 1000、landmark point 固定為 1000。所有實驗的確實數據如表 5-5 所列。



(a)Time (b)Index size

圖 5-11：高斯資料集中改變淹水區塊大小的影響

表 5-5：高斯資料集所有實驗的查詢時間

淹水點數量	VC(ms)	VC+(ms)	C2L(ms)
100	5	3	3
1000	47	30	8
10000	429	265	55

地標數量	VC(ms)	VC+(ms)	C2L(ms)
10	45	29	3
100	47	30	8
1000	51	35	15

淹水點大小	VC(ms)	VC+(ms)	C2L(ms)
0.01	51	35	15
0.1	50	36	25
1	51	43	20

5.6 群聚分佈之實驗

接下來的實驗我們改變資料集的分佈，將地標和淹水區塊的分佈為群集分佈取樣。

(1)這次實驗，如圖 5-12，我們探討改變淹水區塊的數量的影響，landmark point 固定為 100、flooded cell size 為 0.01，執行時間如同隨機資料集隨著淹水區塊增加，執行時間也以線性成長，VC+大約比 VC 快 1.6 倍，而 C2L 在效率上比 VC 快 8 倍。

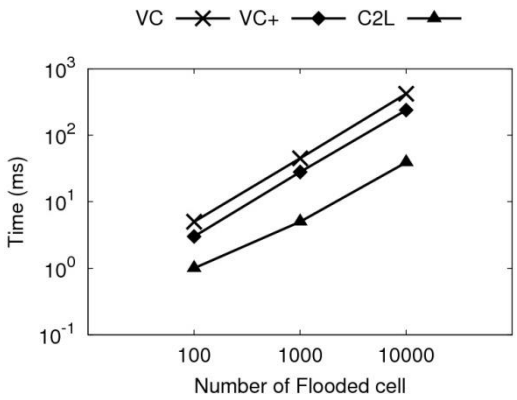
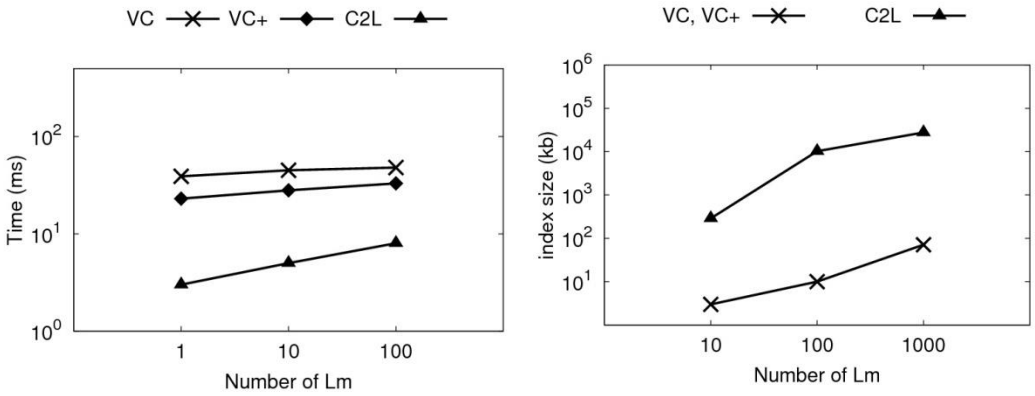


圖 5-12：群聚資料集中改變淹水區塊的數量的影響

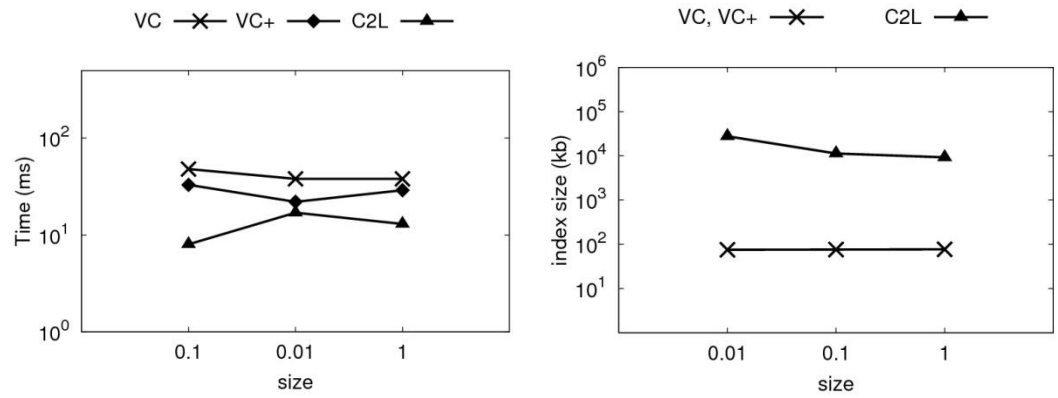
(2)這次實驗，如圖 5-13，我們探討改變地標的數量的影響，flooded cell point 固定為 1000、flooded cell size 為 0.01，VC 和 VC+的 Index size 大小依序為 3kb、8kb 和 75kb，C2L 的 Index size 大小依序為 288 kb、10240 kb 和 27648 kb。其中 VC 與 VC+在地標個數為 100 與 1000 時，執行時間比隨機資料集略為增加，原因為在某些群聚的淹水區塊中，可能與地標有較多空間連接次數，但影響程度沒有高斯分佈來得高，VC+大約比 VC 快 1.5 倍，而 C2L 在效率上比 VC 快 9 倍。所有實驗的確實數據如表 5-6 所列。



(a)Time (b)Index size

圖 5-13：群聚資料集中改變地標的數量的影響

(3)這次實驗，如圖 5-14，我們探討改變淹水區塊大小的影響，flooded cell point 固定為 1000、landmark point 固定為 1000。所有實驗的確實數據如表 5-6 所列。



(a)Time (b)Index size

圖 5-14：群聚資料集中改變淹水區塊大小的影響

表 5-6：群聚資料集所有實驗的查詢時間

淹水點數量	VC(ms)	VC+(ms)	C2L(ms)
100	5	3	1
1000	45	28	5
10000	420	239	39

地標數量	VC(ms)	VC+(ms)	C2L(ms)
10	39	23	3
100	45	28	5
1000	48	33	8

淹水點大小	VC(ms)	VC+(ms)	C2L(ms)
0.01	48	33	8
0.1	38	22	17
1	38	29	13

第 6 章結論與未來方向

在本論文中，我們提出以地標為主將淹水資訊整合的概念，並提出了 3 個方法。首先，我們提出了利用地標建立 Voronoi cell，並利用 Voronoi cell 的 MBR 建立 R-tree 索引的 VC 方法，為了減少搜尋 R-tree 的次數，並提出 VC+的方法以提昇效率。其次，由於淹水區塊的 MBR 至地標的距離可能有過遠的情況，造成 VC 方法會有不準確的情況發生，為了剔除這種情況，提出了 C2L 方法。在此方法中，我們事先設置最小距離門檻將影響力較低的淹水區塊剔除，並針對每個演算區塊找出距離小於門檻值的地標，同時改成以「點」為空間連接的基準。我們進行大量的實驗來評估此三種方法，包括使用真實資料和 3 種不同分佈的人工資料，並分別評估淹水區塊數量和地標數量的影響。首先，針對有效性部分，C2L 明顯地與人為判別的結果較為一致，在效率部分，VC+因為明顯降低搜尋 R-tree 的次數，而在效率上大約比 VC 快 1.6 倍。C2L 雖然事先必須建立較大的 B-tree 索引，但在效率上通常比 VC 快 9 倍。在隨機分布改變淹水區塊數量中，C2L 在效率上可以比 VC 快 14 倍。雖然 VC+在真實資料的實驗上比 C2L 還快，但在其他實驗時都比 C2L 慢。總結來說，C2L 方法需要事先建立大量的索引，但在有效性和效率上都比 VC 方法來的高。群集分佈在部分的區域較密集，所以執行時間比隨機分佈慢。而高斯分佈因為在中間區域較密集，整體的執行時間較其他兩種分佈慢

最後，關於本論文未來的研究方向，希望能夠降低 C2L 索引的大小，同時研究如何進一步提昇其有效性。

參考文獻

[Br+94] Thomas Brinkhoff, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger, “Multi-Step Processing of Spatial Joins,” Proceeding of the SIGMOD conference, Minneapolis, Monnesota, pp. 197-281, 1994.

[BM70] R. Bayer and E. McCreight, “Organization and Maintenance of Large Ordered Indices,” Proceeding of the ACM-SIGFTDET Workshop on Data Description and Access, Houston, Texas, pp. 107-141, 1970.

[Fo87] Steven Fortune, “A Sweepline Algorithm for Voronoi Diagrams”, Algorithmica, Vol. 2, pp.153-174, 1987.

[GB97] Great Britain:Ministry of Defence, “ Admiralty Manual of Navigation”, The Stationery Office, Vol. 1, pp.10, 1997.

[GPA] Google Public Alerts, http://www.google.org/intl/zh-TW/ crisisresponse/publicalerts/

[GS77] P. J. Green and R. Sibson, “Computing Dirichlet tessellations in the plane”, Computational Journal, Vol. 21, No. 2, pp. 168-173, 1977.

[Gu84] Antomn Guttman, “R-trees: A dynamic index structure for spatial searching”, Proceedings of the SIGMOD conference, pp. 47–57, 1984.

[IB] Intial Bearing, http://www.movable-type.co.uk/

[Ke55] Maurice George Kendall, “Rank Correlation Methods”, New York: Hafner Publishing Co, 1955.

[MWR] Mobile Water Regime, http://fhy2.wra.gov.tw/

Pub_Web_2011/page_html/appIntroduction.aspx

[PS85] Franco P. Preparata and Michael Shamos, M.I., “Computational Geometry: an Introduction”, Springer-Verlag, 1985.

[QZKL+12] Jianzhong Qi, Rui Zhang, Lars Kulik, Dan Lin, Yuan Xue, “The Min-dist Location Selection Query”, Proceeding of the ICDE conference, 2012.

[QCCX12] Miao Qiao, Hong Cheng, Lijun Chang, Jeffrey Xu Yu, “Approximate Shortest Distance Computing: A Query-Dependent Local Landmark Scheme”, Proceedings of the ICDE Conference, 2012.

[RKV95] Nick Roussopoulos, Stephen Kelley and Frederic Vincent, “Nearest neighbor queries,” Proceedings of the SIGMOD conference, pp. 71–79, 1995.

[SKS03] Cyrus Shahabi, Mohammad R. Kolahdouzan, and Mehdi Sharifzadeh, “A road network embedding technique for k-nearest neighbor search in moving object databases”, Geoinformation, Vol. 7, No. 3, pp. 255-273, 2003.

[吳12] 吳佩珊, “基於服務導向架構之洪氾預警系統”, 碩士論文, 國立台灣海洋大學資訊工程研究所, 2012.

附錄 A. RDWO 演算法

此演算法的目的是為了處理淹水區塊和地標間有障礙物時，修正淹水區塊和地標間的距離，為繞過障礙物的最短距離。它首先判斷淹水區塊與地標之間是否與障礙物相交，如果相交，則以障礙物四邊形為基準，判斷淹水區塊與地標各在障礙物四周的哪個區域，接著再算出具障礙物的最短距離，反之，則算出淹水區塊和地標之間的直線距離。依據分區的結果可分成兩種可能，第一種情況是淹水區塊與地標在同一線上，則淹水區塊與地標的直線距離則為最短距離，第二種情況利用三角不等式就可找出最短的，其餘的情況則由兩線段中取出最短的，最後即輸出每個淹水區塊到地標具障礙物的距離。

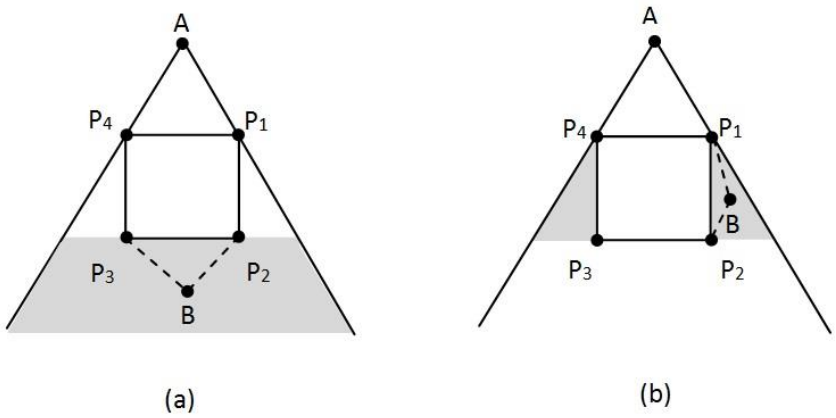


圖 A-1：淹水區塊與地標位於障礙物的可能性

假設輸入點為 A（淹水區塊）、B（地標）和障礙物的 MBR 的四個點（P_1、P_2、P_3和 P_4），且線段 A、B 穿越障礙物的 MBR，以下將障礙物附近所有可能的案例列出來，分別判斷 A 與 B 各在哪些地方，以找出 A、B 最短距離。

I. 如果 B 與$\overline{AP_1}$同一線上

   i. 則$\overline{AB}$即為最短距離。

II. 如果 B 不在$\overline{AP_1}$同一線上

i. 如圖 A-1（b），如果 B 在淺灰色地帶，利用三角不等式即可判斷出最短距離為$\overline{AP_1} + \overline{P_1B}$。

證明：

因為 $\overline{AP_4} + \overline{P_4P_1} > \overline{AP_1}$、$\overline{P_1P_2} + \overline{P_2B} > \overline{P_1B}$，左右相加得 $\overline{AP_4}+\overline{P_4P_1}+\overline{P_1P_2}+\overline{P_2B} > \overline{AP_1}+\overline{P_1B}$，又$\overline{P_4P_1} = \overline{P_3P_2}$，且$\overline{P_1P_2} = \overline{P_4P_3}$，所以 $\overline{AP_4}+\overline{P_3P_2}+\overline{P_4P_3}+\overline{P_2B} > \overline{AP_1}+\overline{P_1B}$，也就是$\overline{AP_4}+\overline{P_4P_3}+\overline{P_3P_2}+\overline{P_2B} > \overline{AP_1}+\overline{P_1B}$

ii. 其餘狀況皆為兩線段中取最短的。如圖 A-1（a），如果 B 在中間淺色地帶，最短距離則為 $\min\{\overline{AP_1}+\overline{P_1P_2}+\overline{P_2B},\ \overline{AP_4}+\overline{P_4P_3}+\overline{P_3B}\}$，若 B 分別在障礙物矩形右下方及左下方，最短距離分別為 $\min\{\overline{AP_1} + \overline{P_1B},\ \overline{AP_4}+\overline{P_4P_3}+\overline{P_3B}\}$和 $\min\{\overline{AP_4} + \overline{P_4B},\ \overline{AP_1}+\overline{P_1P_2}+\overline{P_2B}\}$。

對應的 RDWO 演算法如圖 A-2，cell 變數為一個演算區塊，lm 變數為一個地標，Ob 變數為障礙物集合，flag 用於記錄有無相交障礙物，0 為未相交、1 為第一次相交、大於 1 為相交大於一個障礙物。combineOb 變數記錄合併後障礙物對應的四個端點座標，combines 為合併障礙物方法，兩障礙物取出 *x* 軸與 *y* 軸最大值與最小值。line_rectangleIntersects 方法為淹水區塊至地標與障礙物是否相交，getDist 則為淹水區塊至地標的距離求法。

演算法名稱：RDWO 輸入：cell：淹水區塊 lm：地標 Ob：障礙物集合 輸出：淹水區塊至地標修正過後的距離
RDWO（cell, lm, Ob）
L01 flag←0; L02 **for** *b*∈Ob **do**

```
L03     if   line_rectangleIntersects（cell,lm,b）= true then
L04        flag←flag +1;
L05     if   flag =0   then
L06        continue;
L07     else if   flag =1 then
L08        combineOb←b;
L09     else
L10        combineOb←combineOb   combines   b
L11     if   flag =0 then
L12        return   getDist（cell, lm）
L13     else
L14        if   cell&&lm are in the same line of visible point then
L15           return   getDistance（cell,lm）;
L16        else if   cell&&lm are in the Trigonometric then
L17           return   shortest path;
L18        else
L19           return   min{two paths};
```

圖 A-2：RDWO 演算法