針對路徑群組的天際線查詢演算法之研究
A Preliminary Study on Skyline Query Algorithms for Path Group

張玉圖、陳奕中、楊東麟*
逢甲大學資訊工程學系
Yu-Tu Chang, Yi-Chung Chen, Don-Lin Yang
chang.yutu@gmail.com, mitsukoshi901@gmail.com, dlyang@fcu.edu.tw

摘　要

本篇研究是關於有效率的查詢最具競爭力的路徑群組方法進行軟體塑模，我們在一個無方向的路網圖上，圖中每個路段上有多個維度的屬性值，給定 k 個到訪地點及群組基數 c，路徑群組規劃可以查詢出多個最佳解，每一個路徑群組需滿足路徑個數為 c 及涵括 k 個到訪地點等 2 個要件。路徑群組查詢可以運用在運輸物流或是旅遊行程的規劃，例如快遞公司安排多個快遞員每日貨物運送路線，或是規劃 5 天旅遊行程中，每日旅遊路線。有效率的計算路徑群組是一件極具挑戰的工作，這項作業的困難點在於，路徑群組是由不同起迄點及長短不一的路徑所組成，如果要一個一個找出所有的路徑群組，然後互相做支配檢測，需要很多的運算資源及運算時間。我們的研究針對路徑群組的資料特性，提出路徑群組快速查詢演算法(ESPGA)及路徑群組進階查詢演算法(ASPGA)，利用有效的裁剪機制，儘可能刪除不必要計算的路徑群組資料，在我們的模擬實驗中顯示，本研究所提出的方法和搜尋軟體設計能夠有效提高處理效率，獲得比較好的效能。

關鍵字：路徑規劃、天際線查詢演算法、路徑群組查詢、搜尋軟體設計

一、緒論

路徑規劃是一種運用相當廣泛的 Location-based service，有效率的查詢最具競爭力的路徑是其中相當重要的課題，最近幾年來有許多相關研究及有效的方法被提出[1][2][3][4]，其中 Path Skyline query 是在一個多維度屬性的路網圖中，依據使用者給定的一組起點 s 及終點 t，找出從起點到終點之間最具競爭力的路徑。

路徑群組規劃(Skyline path group query)是路徑規劃的延伸應用，我們以貨運公司安排快遞人員送貨路徑為例子來做說明，圖 1 是一個無方向的路網圖，圖上每個點代表交叉路口，每一個點與圖上其它的點皆有路段相連，線段代表連結二個點的路段，每個路段上的屬性值分別代表二點之間的距離以及從一個地點到另一個地點所需的時間。快遞公司 G 有 2 位快遞員 X 與 Y ，假設今天貨運站中有五件貨物，分別要送達圖 1 中的五個地點，可能的規劃方式有 X 走<A, B >這條路徑，Y 走<C, D, E>，或是 X 走<D, E>，Y 走<A, B, C>，…不同的路徑群組。令 pg 代表路徑群組，我們將圖 1 中找到的路徑群組標記成 pg1 = {<A,B>, <C,D,E>}, pg2 = {<A,B>, <C,E,D>}…, pg30 = {<D,E>, <B,A,C>}, 我們用 w(pg) 表示路徑群組的屬性值，假設路徑群組的屬性值是組合中所有路徑的各個維度屬性值加總，分別計算 pg1 ,pg2…, pg30 我們可以得到 w(pg1) = (18, 21)，w(pg2) = (15, 17)…，w(pg30) = (19, 20)。若我們找出所有的路徑群組，然後計算每個路徑群組的屬性值，我們將所得到的路徑群組及屬性值整理如下頁表 1，觀察 30 個路徑群組的屬性值後，我們發現 pg25 及 pg26 在距離及時間屬性皆優於 pg1, pg2…, pg24, pg27 …, pg30，而任一個 pg25, pg26 路徑群組，都無法在距離及時間屬性上皆優於另外一個路徑群組，因此 pg25, pg26 是 G 公司 2 位快遞人員，分別將貨物送達 5 個地點，最具競爭力的兩個路徑群組。

計算 Skyline Path Group 一個最簡單的方法是找出所有的路徑群組，同時計算每一個路徑群組的屬性值，將每一個路徑群組與剩下的路徑群組做支配檢測(Dominance test)是需要很高的計算成本，而且複雜度會隨著路與趣點路網圖上的路段增加而提高，然而在大部份的情況下，演算法的處理過程中，可能會浪費許多計算資源來計算不可能成為結果的路徑群組。

Combinatorial Skyline Queries (CSQ) 是由 Chung et al. [5] 提出，在他們的研究中，利用二項式係數的概念所發展出的 Decomposition Algorithm，能夠有效淘汰沒有競爭力的資料，但是他們所用的股票資料並無法適用於本篇研究所要探討的路徑群組資料上。

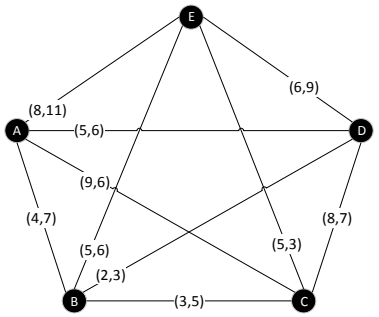

圖 1 五個地點互連無方向路網圖

* 本研究接受科技部編號：MOST 104-2218-E-035-012 和 MOST 104-2119-M-035-002 研究計畫經費補助

Tian et al. [1] 首先將 Skyline 方法運用在 Path query 上，[2][3]研究提出了改進的方法，這些研究主要方法是利用 Greedy algorithm，從起點開始，逐步找出通往終點的可能路徑，利用 Partial path dominance test 及 Full path dominance test 方法，裁剪不可能成為 Skyline 的路徑，然而 Path query 的方法用在本篇研究所探討的 Path group 案例中，裁剪的效果有限，原因在於 Path group 中的路徑並不是固定的起點與終點，而且路徑長短也不同，這些條件不一的路徑尚未配對組合完成前，Path dominance test 的方法無法斷定目前的路徑與其它的路徑群組後，一定是沒有競爭力的路徑群組，因此在路徑 Traversal 的過程中無法有效的淘汰那些不可能成為 Skyline path group 的路徑。

表 1 路徑群組

ID	Path Group	Weight	ID	Path Group	Weight
pg_1	<A,B>,<C,D,E>	(18,23)	pg_{16}	<B,D>,<A,C,E>	(16,12)
pg_2	<A,B>,<C,E,D>	(15,19)	pg_{17}	<B,D>,<A,E,C>	(15,17)
pg_3	<A,B>,<D,C,E>	(17,17)	pg_{18}	<B,D>,<C,A,E>	(19,20)
pg_4	<A,C>,<B,D,E>	(17,18)	pg_{19}	<B,E>,<A,C,D>	(22,19)
pg_5	<A,C>,<B,E,D>	(20,21)	pg_{20}	<B,E>,<A,D,C>	(18,19)
pg_6	<A,C>,<D,B,E>	(16,15)	pg_{21}	<B,E>,<C,A,D>	(19,18)
pg_7	<A,D>,<B,C,E>	(13,14)	pg_{22}	<C,D>,<A,B,E>	(17,20)
pg_8	<A,D>,<B,E,C>	(15,15)	pg_{23}	<C,D>,<A,E,B>	(21,24)
pg_9	<A,D>,<C,B,E>	(13,17)	pg_{24}	<C,D>,<B,A,E>	(20,25)
pg_{10}	<A,E>,<B,C,D>	(19,23)	pg_{25}	<C,E>,<A,B,D>	(11,13)
pg_{11}	<A,E>,<B,D,C>	(18,21)	pg_{26}	<C,E>,<A,D,B>	(12,12)
pg_{12}	<A,E>,<C,B,D>	(13,19)	pg_{27}	<C,E>,<B,A,D>	(14,16)
pg_{13}	<B,C>,<A,D,E>	(14,20)	pg_{28}	<D,E>,<A,B,C>	(13,21)
pg_{14}	<B,C>,<A,E,D>	(17,25)	pg_{29}	<D,E>,<A,C,B>	(18,19)
pg_{15}	<B,C>,<D,A,E>	(15,22)	pg_{30}	<D,E>,<B,A,C>	(19,22)

在多維度屬性值的路網圖中，找出 Skyline path group 是一件非常複雜度的計算工作，我們必須比對所有的路徑群組資料後，才能得到最後的結果，本研究提出路徑群組快速查詢演算法 ESPGA (Efficient Skyline Path Group Algorithm)及路徑群組進階查詢演算法 ASPGA (Advanced Skyline Path Group Algorithm) 演算法，在路徑群組快速查詢演算法中，我們針對有效率計算 Skyline path group queries 問題，以路徑群組擴展方式，逐步找出完整的路徑群組，在路徑群組擴展過程中，利用 Partial path dominance test 裁剪策略及減少排序比對的處理方式，淘汰不可能成為最有競爭力的路徑群組。另外針對完整的路徑群組，提出「路徑支配」的裁剪方法，「路徑支配」資料集中最初儲存各路段的支配關係，而在路徑群組的產生過程中，儲存已計算過路段的支配關係，對於已擴展成完整的路徑群組資料集，可以藉由查詢「路徑支配」資料集中的資料，有效的 Pruning 沒有競爭力的資料，避免重覆計算，提高計算效能。在路徑群組進階查詢演算法中，我們進一步的利用 Full path dominance test 裁剪策略，對於擴展中的路徑群組，加上預估擴展後的最小成本，來預測是否可能成為有競爭力的路徑群組。最後我們將透過模擬環境驗證我們提出的方法效能。

本研究接下來在第二節說明相關論文研究，第三節是問題定義及窮舉法，第四節介紹我們提出的研究方法，分別為路徑群組快速查詢演算法(ESPGA)及路徑群組進階查詢演算法(ASPGA)，第五節說明實驗與結果分析，最後一節為結論。

二、文獻研究與回顧

天際線查詢 (Skyline query) 的方法是由 Borzsonyi et al. [6]首先提出，作者在論文中以一個尋找旅館案例來說明天際線查詢適用情境，在這個案例中，天際線查詢能夠為遊客找出距離海邊不遠而且費用不昂貴的旅館清單，這些旅館不論在價格或是離海灘的距離都是最有競爭力的，能夠讓遊客縮小搜尋以及比較的範圍，從中挑選適合的旅館。

BNL (Block-Nested-Loop) [10][11][12]是一個天際線查詢最直覺的處理方法，BNL 演算法的概念是利用一個天際線候選佇列 (Candidate skyline queue) 來儲存最近一次支配比對後所得到天際線資料集，資料處理的做法是將所有的資料依序一次取出一筆，然後與天際線候選佇列中的資料做支配比對，做完全部的資料後得到 Skyline 的結果。

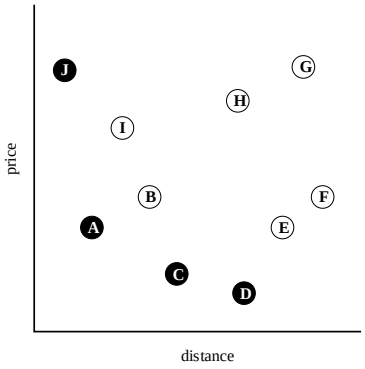

圖 2 天際線查詢資料集

我們以圖 2 為例來說明 BNL 的處理方法，假設 Q 是天際線候選佇列，我們依序從旅館 A 開始處理，最初 Q 的佇列中沒有任何資料，我們將 A 放入天際線候選佇列中，接著將旅館 B 與天際線候選佇列中的 A 做支配比對，A 在距離與價格皆優於 B，於是將旅館 B 淘汰，接下取出旅館 C 與天際線候選佇列中的 A 做支配比對，A 在距離優於 C 但是 C 在價格上優於 A，A 與 C 都無法支配對方，於是將 C 放入天際線候選佇列中，接下取出旅館 D，依序與天際線候選佇列中 A 與 C 做支配比對，A 與 C 在距離上優於 D，D 在價格上優於 A 與 D，因此 A, C, D 皆無法支配對方，因此將 D 放入天際線候選佇列，接下來依序取出 E, F, G, H, I 的資料，這些旅館與天際線候選佇列中 A 做支配比對時，旅館 A 在距離與價格皆優於旅館 E, F, G, H, I，因此將 E, F, G, H, I 等 5 個旅館淘汰，旅館 J 在距離優於 A, C, D，而 A, C, D 在價格優於旅館 J，最後將 J 放入天際線候選佇列中，比對完所有資料

後，在天際線候選佇列中所保留不被支配的 A, C, D, J 等 4 個旅館，即為天際線查詢的結果。BNL 的方法能夠避免將待處理的資料與其它剩餘的資料一一做支配比對，減少不必要的資料比對程序，缺點是當天際線候選佇列的資料筆數過多時，每一次取出的資料也必須與天際線候選佇列中的資料一一做支配比對，因此效能會隨天際線查詢結果資料的多寡成反比。

D&C (Divide and Conquer) 演算法[6][9]的作法是將資料切割成小部份分別處理，資料處理的方法主要分為三個步驟，首先將資料集分割成數個部份子資料集 (Partition) ，例如依記憶體的大小分割子資料集，第二個步驟分別計算每個子資料集中的 Skyline，最後再將每個子資料集所找到的 Skyline 合併起來，最後得到整個資料集的 Global Skyline。D&C 的演算法可以有效改進 BNL 的缺點，在 D&C 第二個步驟中可以套用 BNL 的資料處理方式，由於 D&C 的處理模式是將資料切割成較小的子資料集，因此能夠減少 BNL 演算法的比對過程中，因為天際線候選佇列中資料過多所造成效能低落的問題。

Tian et al. [1]最先提出在路網中查詢天際線路徑的方法，在他們的研究中，路網上的每一個 edge 支援多維度的屬性值，如果給一個查詢的起點 s 與終點 t 的條件，作者的演算法可以找出 s 到 t 所有的 Skyline 路徑。

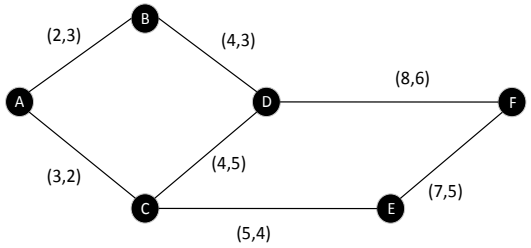

圖 3 路徑查詢路網圖

以圖 3 舉例說明 Skyline path query，圖 3 是一個無方向的路網圖，圖上每個點代表交叉路口，線段代表連結二個點的路段，每個路段上的屬性值分別代表二點之間的距離以及從一個地點到另一個地點所需的時間。如果我們要從 A 點到 F 點，我們可以找到四條路徑，分別是 p1=<A, B, D, F>, p2=<A, B, D, C, E, F>, p3=<A, C, D, F>, p4=<A, C, E, F>，假如路徑的屬性值等於加總路徑中所有路段的屬性值，那麼 p1 距離屬性值是 2+4+8 = 14，時間屬性值為 3+3+6 = 12，我們以 w(P)表示路徑 P 的屬性值，在上述的例子中，我們分別計算 p1, p2, p3, p4 後得到，w(p1)=(14, 12), w(p2)=(22, 20) , w(p3)=(15, 13) , w(p4)=(15, 11)，觀察四個路徑的屬性值後，我們發現 p1 在距離及時間屬性皆優於 p2 與 p3，p1 與 p4 互相比較，p1 在距離屬性雖然優於 p4，但是時間屬性上 p4 卻優於 p1，最後我們得到 p1 與 p4 這兩條路徑，我們說 p1 與 p4 是 A 點到 F 點的 Skyline path。

作者的演算法[2][3][4][7]是使用 Greedy algorithm，在查詢 s 與 t 中間的所有路徑的過程中，利用中繼點 n 來淘汰沒有競爭力的路徑。第一種裁剪方法是在 s 點與 n 點中間的子路徑(Sub-path)資料集中，先做一次支配檢測，因為被淘汰的 Sub-path 加上 n 點與 t 點中的所形成的 Path 也一定不會是 Skyline path，以圖 3 中的路徑 p1=<A, B, D, F> 以及路徑 p3=<A, C, D, F> 來做說明，假設地點 D 為 A 點到 F 點的中繼點，路徑 p1 中 A 點到 D 點的子路徑為<A, B, D>，權重為(6, 6)，路徑 p3 中 A 點到 D 點的子路徑為<A, C, D>，權重為(7, 7)，兩條子路徑<A, B, D>與<A, C, D>做支配比對後得到 <A, B, D> 支配 <A, C, D>，p3 路徑包含子路徑為<A, C, D>，我們不用計算出 p3 路徑的全部權重，在中繼點 D 的時候就可以得知所有包含子路徑 <A, C, D> 的路徑都會被 <A, B, D>支配。第二種裁剪方法是在 n 點所得到的子路徑再加上 n 點到 t 點的各個屬性最小值相加，如果會被已知的 Skyline path dominate，那這個子路徑也無法成為 Skyline path。

三、問題定義與窮舉法

在這章節中將說明組合式路徑天際線查詢的環境、假設及問題定義，並且討論直覺的解決方法-窮舉法（Brute-Force Method）所面對到的難題。

- 本篇研究假設在一個集中式的處理環境，伺服器上有路網中所有路段及它們的屬性值，同時也知道使用者給定的組合中的路徑個數。
- 在一個無方向的路網圖上選取的興趣點 N 以及連結興趣點的路段 E，$GOI(N, E)$表示興趣點及其連結路段所形成的興趣點路網圖(**G**raph **O**f **I**nteresting)。
- 路段的屬性是一個 |d| 維度的資料空間 $S=\{d_1, d_2,\ldots, d_{|d|}\}$，$dom(S)$表示 Domain 中所有屬性資料。

定義 1. 地點 (Nodes)。$N=\{ n_1, n_2,..., n_{|n|} \}$，$N$ 為無方向路網上的興趣點資料集，|n|是地點個數。

定義 2. 路段 (Edges)。$E=\{ e_1, e_2,..., e_{|e|} \}$，$E$ 表示無方向路網上連結 2 個地點的路段資料集，|e| 是路段個數，假設 e_k 是路網上的其中一個路段，$e_k \in E$，$e_k=\{n_i, n_j\}$，其中 $n_i, n_j \in N$，而且 $i \neq j$。

定義 3. 屬性值 (Attribute)。假設 e_k 是路網上的其中一個路段， $e_k \subseteq E$，令 $w(e_k)=\{ w_{k1}, w_{k2},..., w_{k|d|} \}$ 代表路段 e_k 的屬性，是一個 |d| 維度的屬性值，$w(e_k) \subseteq dom(S)$。

定義 4. 路徑 (Path)。路徑是由一序列的地點所組成的 *tuple*，$p=< n_{k1}, n_{k2},..., n_{k|p|} >$，$n_{ki} \subseteq N$，|p| 代表路徑中的地點個數，$n_{k1}$ 是路徑的起始點，$n_{k|p|}$ 是路徑的迄止點，任意 2 個不同路徑 p_j, p_k 中的地點序列不會相同，而且路徑 p 中的地點不會重覆。

定義 5. 路徑屬性(Path attribute)。任一條路徑 $p=<n_{k1},n_{k2},..., n_{k|p|}>$ 的屬性為 $w(p)=\{pw_1, pw_2,\ldots,$

$pw_{|d|}\}$，|d|表示路徑屬性的維度值，令 pw_i 為路徑 p 第 i 個屬性，屬性的計算方式為路徑中所有路段的第 i 個屬性累加，公式(1)表示 pw_i 的計算方式。

$$pw_i = \sum_{k=1}^{|p|-1} w_{ki} \qquad (1)$$

w_{ki} 表示路徑中第 k 個路段 e_k 的第 i 個屬性，e_k={ n_k, n_{k+1} }

定義 6. Path p_i dominate path p_j。假設 p_i 與 p_j 是兩個不同的路徑，若路徑 p_i 在每一個維度的屬性值都不差於路徑 p_j，而且路徑 p_i 至少有一個維度的屬性值勝過路徑 p_j，我們可以說路徑 p_i dominate 路徑 p_j。

定義 7. 路徑群組(Path Group)。路徑群組是由路徑資料集所組成，路徑群組 $pg = \{ p_1, p_2, \ldots, p_{|p|} \}$，|p| 表示路徑個數，本研究中不考慮路徑群組中的路徑資料集順序，因此{ p_1, p_2} = { p_2, p_1 }。

假設使用者指定組合中的路徑個數(Cardinality)為 q，給定一個路徑群組 $pg_i = \{ p_1, p_2, \ldots, p_{|p|} \}$，令 V_i 為路徑群組 pg_i 中所有的地點資料集，$V_i = \{ v_1, v_2, \ldots, v_{|v|}\}$，$N$ 是選取的興趣點資料集，若 $V_i \cap N = N$ 而且 $|p| = q$，那我們可以說 pg_i 是一個完整路徑群組(Full path group)。

定義 8. 路徑群組屬性(Attribute of a path group)。給定一個路徑群組 $pg = \{ p_1, p_2, \ldots, p_{|q|} \}$，路徑 pg 的屬性為 $w(pg) = \{ pgw_1, pgw_2, \ldots, pgw_{|d|}\}$，|d| 表示路徑屬性的維度值，令 pgw_i 為路徑群組 pg 第 i 個屬性值，屬性的計算方式為路徑群組中所有路徑的第 i 個屬性值累加，公式(2)表示 pgw_i 的計算方式。

$$cpw_i = \sum_{j=1}^{|q|} pw_{ji} \qquad (2)$$

pw_{ji} 表示路徑群組中第 j 個路徑的第 i 個屬性。

由公式(1)(2)可以得到，pgw_i 可以由路徑群組中第 j 個路徑中所有的路段的第 i 個屬性值累加，(3)表示計算方式。

$$cpw_i = \sum_{j=1}^{|q|} \sum_{k=1}^{|p|-1} p_j w_{ki} \qquad (3)$$

$p_j w_{ki}$ 表示路徑群組中第 j 個路徑中，第 k 個路段的第 i 個屬性值。

定義 9. Path group pg_i dominate path group pg_j。假設 pg_i 與 pg_j 是兩個不同的路徑群組，若路徑群組 pg_i 在每一個維度的屬性值都不差於路徑群組 pg_j，而且路徑群組 pg_i 至少有一個維度的屬性值勝過路徑群組 pg_j，我們可以說路徑群組 pg_i Dominate 路徑群組 pg_j。本研究中用 $pg_i \vdash pg_j$ 表示 pg_i Dominate pg_j，Dominate 數學式表示如下：

$$\forall k, w_k \in S, p_i w_k \geq p_j w_k \wedge \exists\, w_l \in S, p_i w_l > p_j w_l$$

定義 10. Skyline 路徑群組(Skyline path group)。假設沒有其它的路徑群組 pg_j，pg_j dominate pg_i，那我們可以說 pg_i 是 Skyline 路徑群組。

處理 Skyline path group query 時，一個直覺的方法就是列舉所有可能的路徑群組，然後再從這群路徑群組中，找出符合使用者指定的路徑個數的所有 Skyline path group。圖 4 是興趣點路網圖 GOI (8,28)，路網圖中包含 8 個地點，每一個地點與圖中的其它地點皆有路段相連，路段數為$C\left(\frac{8}{2}\right) = 28$ 個路段，每一個路段上有一個 2 維度的屬性值。令使用者給定的組合中路徑個數是 3，我們要從興趣點路網圖 4 中找出所有的路徑組合，每一個路徑組合需滿足以下條件，(1)涵括 8 個地點，(2)路徑群組中的路徑數為 3。我們以字典詞彙順序 (Lexically) 逐一找出所有的路徑群組方式可以分為兩個步驟，第一個步驟是找出每一個路徑群組的地點數群組，第二個步驟依每地點數群組找出每種群組的所有路徑。

找出每一個路徑群組的地點數群組的方法，依路徑群組中的指定路徑數建立數字集合，然後將最短路徑數 2 填入集合中的每個數字，修改集合中最後一個數值為地點個數減去集合中其它數值加總，如此可以得到第一組地點數群組，接著從集合中第一個數值開始往後尋找目前數值最大且大於前一個元素的數值，然後將該數值減 1 之後取代既有的數值，假設這個新數值為 n，然後往前尋找集合中第一個小於 n 的數值，將該數值加 1 後得到一個新的地點數群組，依此方式重覆尋找直到找不到新的地點數群組為。圖 4 找出第一個地點數群組為{2, 2, 4}，第二個地點數群組為{2, 3, 3}。

第二個步驟是將地點以字典詞彙順序 (Lexical) 填入地點群組，以地點數{2, 2, 4}為例，依序取得路徑群組如 pg1 ={<A, B>, <C, D>, <E, F, G, H>}, pg2={<A, B>, <C, D>, <E, F, H, G>}, pg3={<A, B>, <C, D>, <E, G, F, H>}…，做完第一個地點數群組後，接著找出地點數群組{2, 3, 3}的路徑群組，例如 pg100 ={<A, B>, <C, D, E>, < F, G, H>}, pg101 ={<A, B>, <C, D, E>, < F, H, G>}, pg102 ={<A, B>, <C, D, E>, < G, F, H>}…, 找出所有的路徑群組後，再進行支配比對。

路徑群組支配比對 (Path group dominance test process) 是依序取出路徑群組的資料集中群組資料，分別計算每一個路徑群組的成本，路徑群組的成本計算方式為群組中每一條路徑的成本加總，舉例來說，計算路徑群組 pg1 的屬性，首先找出路徑 <A, B>, <C, D>, <E, F, G, H>的成本，分別為(4, 5), (8, 7), (17, 13) ，路徑群組 pg1 的屬性值為 w(pg1) = w(<A, B>) + w(<C, D>) + w(<E, F, G, H>) = (4,5) + (8, 7) + (17,13) = (29, 25)，接著以相同的計算步驟計算 pg2 的成本，pg2={<A, B>, <C, D>, <E, F, H, G>}，w(pg2) = w(<A, B>) + w(<C, D>) + w(<E, F, H, G>) = (4,5) + (8, 7) + (17,12) = (29, 24)，接著計算

pg3 的成本，pg3={<A, B>, <C, D>, <E, G, F, H>}，w(pg3) = w(<A, B>) + w(<C, D>) + w(<E, G, F, H>) = (4,5) + (8, 7) + (8,11) = (20, 23)…，計算完所有的路徑群組後，將所有路徑群組依序進行支配比對，pg2 支配 pg1, pg3 支配 pg2 …, 最後得到不被支配的路徑群組資料集就是 Skyline path group。

本研究的 Brute Force 方法，於實驗過程中，在找出所有路徑群組資料後，使用 BNL 演算法進行支配比對，在比對開始時，會建立一個候選佇列資料集，然後依序取出路徑群組資料集中的路徑群組，與候選佇列資料集中的路徑群組做支配比對，比對的過程中被支配的路徑群組將移除，支配路徑群組排序值調整為候選佇列的第一筆資料，當路徑群組資料集中的每一筆資料都處理完後，留在候選佇列資料集中不被裁剪的路徑群組即為查詢結果。

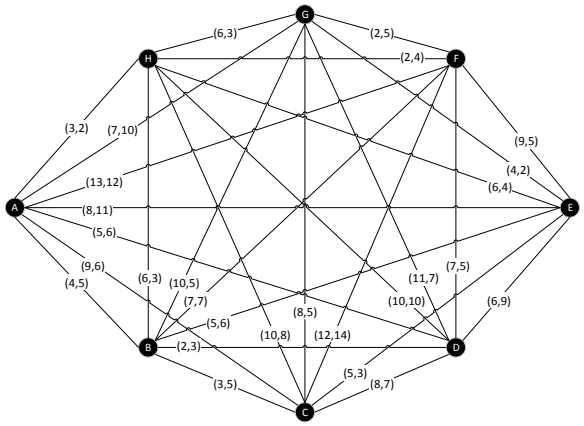

圖 4 興趣點路網圖 GOI(8,28)

四、研究方法

在上一個章節中 Brute Force 的做法，是先找出所有路徑群組資料集，然後使用 BNL 的演算法，從資料集中依序取出路徑群組，然後做支配比對，路徑群組支配比對的過程需要很高的計算成本，而且複雜度會隨著路興趣點路網圖上的路段增加而提高。此外當路段的屬性值維度越高時，除了每一筆路徑群組做支配比對時，比對維度資料的次數隨著增加，同時因為候選佇列中的路徑群組資料增加，每一筆路徑群組所需比對次數也增加。然而在大部份情況下，演算法的處理過程中，可能會浪費許多計算資源來比對不可能成為結果的路徑群組。

在我們的研究中提出一個提昇效率的路徑群組查詢方法，這個方法是利用 Greedy 演算法以及 D&C 演算法的概念，希望能夠很快的找出 Skyline 路徑群組資料，這些 Skyline 路徑群組資料將擔任 Killer 的角色，在後續的路徑群組資料產生過程中，能夠過快速的裁剪那些不可能成為 Skyline 的路徑群組。此外為了避免逐筆路徑群組資料與候選佇列資料相互比對，我們採用了 D&C 演算法的分割概念，一次找出數個資料內容關聯性較高的路徑群組，先將這批路徑群組做一次支配比對，取得 Localize skyline 路徑群組，再將這 Localize skyline 路徑群組與候選佇列中的路徑群組做支配比對，減少不必要的比對次數。本篇論文所提出的方法，結合了漸進式組合路徑方法及有效的裁剪策略，進而能夠快速計算出路徑群組天際線查詢的結果，本研究方法的特點簡述如下：

- 漸進式路徑組合 (Incremental path combination)：利用基本路徑群組擴展的方式，產生完整的路徑群組，基本路徑群組指的是路徑群組中的每一個路徑皆為最短路徑，利用基本路徑群組擴展方式優點為，可以不必一次找出所有的路徑群組資料集，佔用較大的記憶體儲存空間，而每一次擴展後可以得到關連性較高的路徑群組，有助於裁剪作業的進行。
- 路段成本排序(Sorted with cost of edge)：令路段的成本為路段中各維度的屬性值加總，假設成本較低的路段支配成本較高的路段，則成本的較低的路段比較有可能成為 Skyline 路段。因此我們先將路段依成本值由低排到高，從最低的路段成本開始加入基本路徑群組中，成本較低的基本路徑群組擴展而成的路徑群組的成本也比較低，也比較有可能成為 Skyline。利用此方法可以越早找到 Skyline 路徑群組，而這些 Skyline 路徑群組可以當作是成為 Skyline 路徑群組門檻值，對於隨後產生的路徑群組，發揮更多有效的裁剪作用。
- 裁剪策略(Prune strategies)：Partial path dominance test 和 Full path dominance test 等 2 個方法，可以在路徑產生的過程中，裁剪掉不可能成為 Skyline 的路徑資料。Partial path dominance test: 對於不同路徑中的相同起迄點的 Sub-path，如果 Sub-path sp_1 Dominate Sub-path sp_2，則所有包含 sp_2 的路徑都會被包含 sp_1 的路徑所支配。Full path dominance test:對於目前處理的 Sub-path 如果已經被已知的 Skyline dominate，那包含這個 Sub-path 的路徑也無法成為 Skyline。本篇論文利用這 2 個裁剪策略，提出利用路徑支配關係及最小值支配檢驗裁剪方法，針對漸進式路徑組合的過程中，所產生一批完整的路徑組合資料集，能夠快速裁剪沒有競爭力的路徑群組，減少比對次數，提昇效率。

路徑群組快速查詢演算法(ESPGA)及路徑群組進階查詢演算法(ASPGA)的處理程序，皆是利用路徑擴展及最小堆積樹排序的方式(Min-Heap Sort Algorithm)提昇排序及擴展的處理效能。基本路徑群組是群組中的路徑皆為最小路徑，例如在圖 4 興趣點路網圖上，要找出路徑數為 3 的路徑群組時，我們可以從基本路徑群組{<*A, B*>, <*C, D* >, <*E, F* >}來擴展路徑，為了越快找到成本較小的路徑群組，我們先將路段依成本大小排序，成本越小的路段排序值在前面，排序後的路段依序為<A, H>, <B, D>, <E, G>…，接著將排序後的路段順序取出產生基本路徑群組，{<A, H>, <B, D>, <E, G>}, {<A, H>,<B, D>, <H, G>}, {(B, D) , (E, G), (F, H) }…，從 28 個路段任意選擇 3 個路段做組合，總共有 $C\left(\frac{28}{3}\right)$= 3276 種組合，將重覆地點的組合排除掉後，有效的基本路徑群組數 = 420。基本路徑群組擴展方式，自到訪地點清單中，扣除基本路徑群組中涵括的地

點後，將未包含的地點，一次一個加到基本路徑群組中，增加的方式為將新增的地點，依序加到路徑中每個路段中，同時也包含每條路徑的起始地點前及結束地點後。以基本路徑群組{<A, H>, <B, D>, <E, G>}來說明擴展的過程，假設要新增的地點為C，首先將地點 C 分別新增至路徑<A, H> 的前,中,後，可以得到 3 組新的路徑群組分別為{<C, A, H>, <B, D>, <E, G>}, {<A, C, H>, <B, D>, <E, G>}, {<A, H, C>, <B, D>, <E, G>}，以此類推將地點 C 新增至路徑<B, D>, <E, G>，接下來將其他剩餘未包含的地點新增至基本路徑群組{<A, H>, <B, D>, <E, G>}中，全部完成後先判斷這批擴展的路徑群組是否為完整的路徑群組，如果是完整的路徑群組則進行支配比對，否則將這批擴展的路徑群組加至路徑群組資料集，再從路徑群組資料集中取出成本最小的路徑群組擴展。在擴展及排序的程序中，如果每一次擴展之後，將所有的待擴展路徑群組重新排序，必須將待擴展清單中所有的路徑群組一一做比較，每一次排序的時間複雜度是$O(n\log n)$，隨著擴展的路徑群組清單中的數量愈多時，效能會明顯降低，我們提出的改善方法是將待擴展的路徑群組資料放到堆積樹中(Heap Tree)，利用 Min-Heap Sort Algorithm 從擴展的路徑群組清單中，快速得到成本最低的路徑群組資料。

最小堆積樹(Min-Heap Tree)是一個完整的二元樹，樹中所有的節點的值皆小於子節點，樹根是堆積樹中最小的，每一次排序取得最小值的時間複雜度是$O(\log n)$。我們要讓樹根的成本是堆積樹中最小值，每一次增加一筆新的資料時，把該筆資料放到最小堆積樹的最後一個。然後將最後一筆資料做排序。排序方式為將新增加的路徑群組加到最小堆積樹的最後一個節點，然後從最後一個節點開始與父節點做比對，如果子節點的數值小於父節點，則將子節點的位置與父節點交換，如果不是則排序停止。當子節點的位置與父節點的位置交換後，繼續由父節點(假設為 A)與 A 節點的父節點做比對，直到樹根或子節點的數值大於父節點數值時停止。

我們所提出的路徑群組快速查詢演算法，運用 Partial path dominance test 的概念，在每一次路徑群組擴展前，先將待擴展的路徑群組與天際線候選佇列的路徑群組做支配比對，如果待擴展的路徑群組已經被支配，則停止擴展，因為任何由此路徑群組所擴展的路徑，都不會是 Skyline 路徑群組。

對於路徑擴展所產生的完整路徑群組資料集，首先依據擴展前的路徑數區分群組，然後再從每一組的擴展路徑群組中找到差異路徑，最後查詢支配關係表對差異路徑進行裁剪。路徑支配資料集(Path Dominance Dataset) 在查詢處理之前，先將路網圖中每一個路段，相互做支配比對，然後將比對的結果，利用 Linked List 儲存路徑支配關係。以圖 4 的路網圖舉例來說，路段<B, C>, <B, D>, <C, D> 的路段支配關係如表 2 所示，而在產生完整路徑群組如{<E, G>, <A, H, F>, <C, B, D>}, {<E, G>, <A, H, F>, <B, C, D>}, {<E, G>, <A, H, F>, <B, D, C>}，我們可以利用預先建立的路徑支配關係表，快速找到彼此的支配關係，有效的進行裁剪作業。

表 2 路徑支配關係

路徑	支配路徑
<B,C>	<A,B>,<A,C>,<A,D>,<A,E>,<A,F>, <A,G>,<B,E>,<B,F>,<B,G>,<C,D>, <C,F>,<C,G>,<C,H>,<D,E>,<D,F>, <D,G>,<D,H>,<E,F>
<B,D>	<A,B>,<A,C>,<A,D>,<A,E>,<A,F>, <A,G>,<B,C>,<B,E>,<B,F>,<B,G>, <B,H>,<C,D>,<C,E>,<C,F>,<C,G>, <C,H>,<D,E>,<D,F>,<D,G>,<D,H>, <E,F>,<E,H>,<F,G>,<F,H>, <G,H>
<C,D>	<A,E>,<A,F>,<C,F>,<C,H>,<D,G>, <D,H>

路徑群組進階查詢演算法(ASPGA)，在本研究中進一步提出更好的裁剪策略，利用預估的方式，判斷在路徑群組的擴展過程時，所得到路徑群組是否可能成為 Skyline 路徑群組，如果不可能成為 Skyline 路徑群組就直接裁剪掉，無需浪費運算資源做支配檢測。

表 3 路徑支配關係

擴展後路徑群組	擴展路段	路段成本
{<E,G>,<A H,F>,<C,B,D>} {<E,G>,<A,H,F>,<B,C,D>} {<E,G>,<A,H,F>,<B,D,C>}	<B,C> <C,D>	(3,5) (8,7)
{<B,D>,<A,H,F>,<C,E,G>} {<B,D>,<A,H,F>,<E,C,G>} {<B,D>,<A,H,F>,<C,G,E>}	<C,E> <C,G>	(5,3) (8,5)
{<B,D>,<E,G>,<C,A,H,F>} {<B,D>,<E,G>,<A,C,H,F>} {<B,D>,<E,G>,<A,H,C,F>} {<B,D>,<E,G>,<A,H,F,C>}	<A,C> <C,H> <C,F>	(9,6) (10,8) (12,14)

Lower Bound Estimation，在每一次路徑群組擴展之前，先估算擴展路徑最小值，計算方法是，找出擴展地點與待擴展路徑群組的每一個地點所形成的路段，計算每個維度最小的屬性值。以圖 4 地點 C 對路徑群組{<B, D>,<E, G>,<A, H, F>}所擴展而成的路徑群資料集來做說明，表 3 是每一個擴展後的路徑群組的擴展路段以及路段成本。路段成本的最小值是(3, 3)，地點 C 對路徑群組{<B, D>,<E, G>,<A, H, F>}擴展預估最小值為路徑群組{<B, D>,<E, G>,<A, H, F>}加上路段成本最小值。

最小值支配檢驗(Min. Dominate Test)：假設路徑群組{<B, D>,<E, G>,<A, H, F>}的成本在圖 5 中 B 點的位置，目前 Skyline 候選佇列的路徑群組資料在圖 5 斜線 SKY 所包含的範圍)。假設從路徑群組的擴展後所得到資料集，加上尚未擴展完成的路徑的預估最小值後，是在圖 5 中 A 的範圍，則代表路徑群組資料集的任一筆路徑群組資料，都不可能成為 Skyline 路徑群組，可以直接裁剪掉。

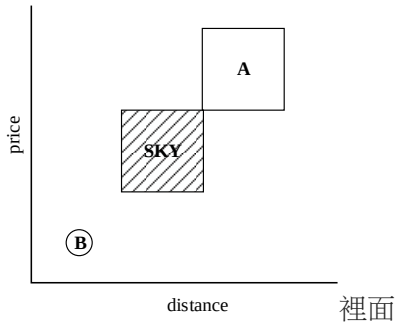

圖 5 最小支配檢驗

路徑群組的排序與擴展的處理過程：首先找出基本路徑群組，然後將基本路徑群組資料集一筆一筆加入最小堆積樹，每一次將路徑群組加至最小堆積樹最後一個節點，然後依據路徑群組的成本做一次排序，排序完成後在最小堆積樹中樹根的路徑群組，是目前在最小堆積樹中成本最小的路徑群組。接著再由這個成本最小的路徑群組開始擴展，然後取得新的一批路徑群組資料集，對於擴展的路徑群組以本研究所提出的裁剪策略進行刪剪，裁剪後不被淘汰的路徑群組如果是完整的路徑群組則與候選佇列中的天際線資料集進行支配比對，其它尚可擴展的路徑群組則遞迴進行加入、排序、擴展等處理程序，因為篇幅限制未能呈現演算法內容。

五、初步實驗與結果分析

本研究模擬一個路網圖，圖上的每一個地點與其它的地點皆有路段相連，路段上第一個維度資料是從德國 Oldenburg 路網上隨機取得的距離資料，接著再利用距離資訊，以亂數方式模擬路段的第 2 至第 6 個維度屬性值。本研究將比較窮舉法、路徑群組快速查詢演算法(ESPGA)及路徑群組進階查詢演算法(ASPGA)三種方法，分別在下列不同的執行條件下，評估執行效能，藉此檢視各個方法的執行效率及優缺點。本研究的路徑群組中的路徑基數設定為 3 個。

- 執行條件：(1)資料屬性維度 (2, 3, 4, 5, 6)。(2)到訪地點 (8, 9, 10)。
- 執行效能評估：(1)執行時間。(2)比對路徑群組數。(3)存取路徑數。(4)裁剪路徑群組數。

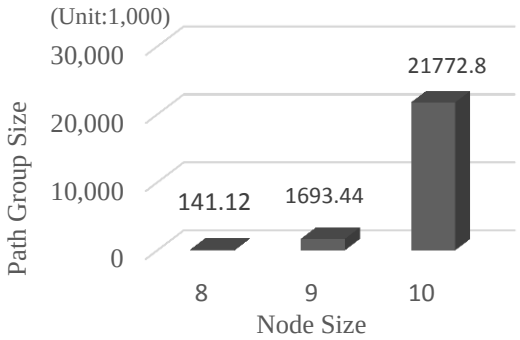

圖 6 存取路徑群組數比較(窮舉法)

圖 6 是窮舉法在不同地點數下，存取的路徑群組數，在路徑群組中的路徑基數設定為 3 的條件下，分別需處理 141120, 1693440 及 21772800 筆路徑群組資料，每增加一個地點數，路徑群組增加 12 倍。圖 7 顯示出本研究的方法與窮舉法於 9 個地點的路網圖上，分別在不同的屬性維度上存取路徑群組數，以二個維度的資料舉例來說，相對於窮舉法需對 1693440 個路徑群組做支配比對，本研究所提出的方法，只需要比對 67544 筆路徑群組，其它 96% 的路徑群組皆在路徑擴展的過程中淘汰。

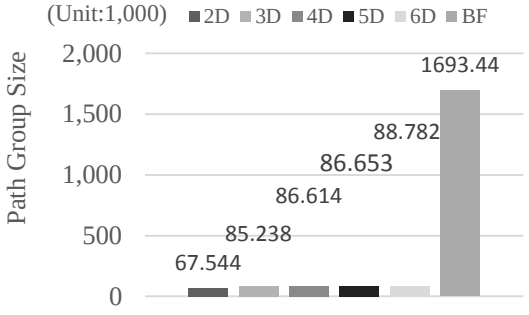

圖 7 存取路徑群組數比較(9 個地點)

圖 8 顯示在 8 個地點的路網上，窮舉法與本研究方法處理效能的比較。窮舉法在資料維度增加的情形下，因為所得到 Skyline 路徑群組會隨著增加，在處理時間上略為較長，處理時間整體來說大約維持在 40 秒左右。本研究所提出的 ESPGA 與 ASPGA 處理時間分別約為 0.35 秒及 0.15 秒，在效能上明顯優於窮舉法。由於 ESPGA 與 ASPGA 這兩個方法在 8 個地點的路網圖上裁剪的路徑群組數差別不大，在處理的時間上並沒有很明顯的差異。

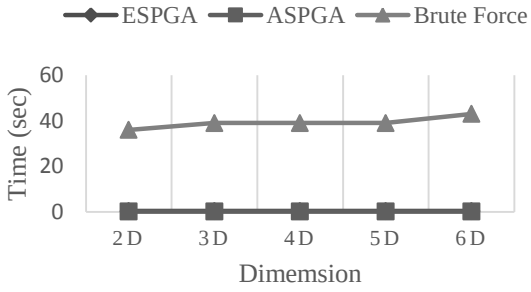

圖 8 處理時間比較(8 個地點)

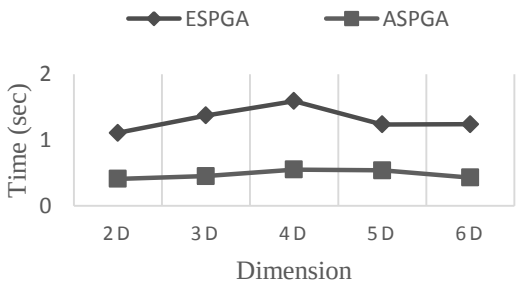

圖 9 處理時間比較(9 個地點)

圖 9 顯示在 9 個地點的路網圖上，窮舉法因為處理時間過長，我們並未顯示在圖 9 及圖 10 中。比較 ESPGA 與 ASPGA 這兩個方法，ASPGA 能夠進一步裁剪沒有競爭力的資料集，觀察 9 個地點 2 個維度的資料，經過 ASPGA 裁剪後所需處理的路徑資料，大約為 ESPGA 處理的資料 40%，在處理時間上 ASPGA 在每一個維度皆優於 ESPGA。

圖 10 顯示 10 個地點 ESPGA 及 ASPGA 處理時間的比較，圖 6 顯示 10 個地點數使用窮舉法所需處理的資料是 9 個地點的 12 倍，觀察 2 個維度的資料條件下，ESPGA 在 10 個地點數所存取的路徑資料是 9 個地點所存取路徑資料 7.5 倍，而 ASPGA 在 10 個地點數所存取的路徑資料是 9 個地點所存取路徑資料 3.5 倍，在處理的時間上，ESPGA 需時 15 秒以上的處理時間，而 ASPGA 處理時間在 2.2 秒以上，由此可知 ESPGA 的方法隨著地點數及路徑群組資料的增加，處理時間增加較多，而 ASPGA 處理時間增加的幅度較少，相較於 ESPGA，效能提高許多。另外在處理不同維度資料時，會隨著找出 Skyline 群組資料的時間點不同，在處理時間上呈現起伏情況，觀察線性 ESPGA，處理時間會隨著維度增加呈現上升的趨勢。

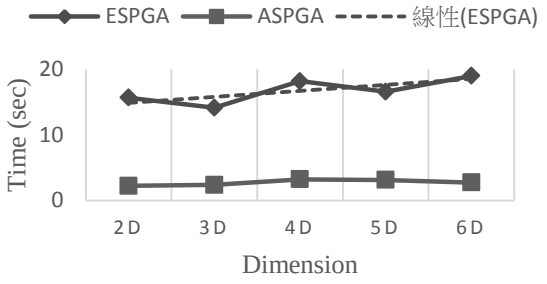

圖 10 處理時間比較(10 個地點)

六、結論

路徑群組的規劃需求，能夠適用在現實生活中的多種案例中，例如機構中的社工員要做個案訪視時，路徑群組規劃可以依社工人員的人數，將欲拜訪的個案分成幾組，然後規劃好每組個案的拜訪路徑。旅遊規劃上的應用，可以將要遊覽的景點，依旅遊天數區分成數個群組，再利用路徑群組規劃找出最有競爭力的旅遊行程。

本研究提出新的路徑群組天際線查詢方法和有效的搜尋軟體設計，快速的獲得處理結果：

- 在路徑的擴展過程中，結合 Heap Sort 演算法，降低排序所需運算的次數。
- 在路徑擴展所產生的子路徑群組(Sub-path group)中，查詢預先建立的路段支配樹的方式，快速排除沒有競爭力的路徑群組。
- 對於擴展中的子路徑群組，利用估計的最小值來預測是否有機會成為 Skyline 路徑群組，如果不可能成為 Skyline 路徑群組則直接裁剪，能夠有效減少需處理的路徑群組資料集，提昇處理效能。

參考文獻

[1] Y. Tian, K. C. K. Lee, and W. C. Lee, "Finding Skyline Paths in Road Networks," *in Proceedings of ACM GIS*, Seattle, USA, pp. 444-447, Nov. 2009.

[2] H. P. Kriegel, M. Renz, and M. Schubert, "Route Skyline Queries: A Multi-Preference Path Planning Approach," *in Proceedings of 2010 IEEE 26th International Conference on Data Engineering (ICDE)*, Long Beach, USA, pp. 261-272, Mar. 2010.

[3] M. Shekelyan, G. Jossé and M. Schubert, "ParetoPrep: Fast computation of Path Skylines Queries," *In proceedings of the 14th International Symposium on Spatial and Temporal Databases*, SSTD 2015, Hong Kong, China, pp. 40-58, 2015.

[4] A. Katiyar, A. Bhattacharya and S. Mitra, "Efficient and Effective Route Planning in Road Networks with Probabilistic Data using Skyline Paths," *In proceedings of the 1st IKDD Conference on Data Sciences*, New Delhi, India, 2014.

[5] Y. C. Chung, I. F. Su and C. Lee, "Efficient computation of combinatorial skyline queries," *Information Systems*, Vol. 38, No. 3, pp. 369-387, 2013.

[6] S. Borzsonyi, D. Kossmann, and K. Stocker, "The Skyline Operator," *in Proceedings of the 17th International Conference on Data Engineering(ICDE)*, Heidelberg, Germany, pp. 421-430, 2001.

[7] K. Deng, Y. Zhou, and H. Shen, "Multi-source query processing in road networks," *In Proceedings of the 23th International Conference on Data Engineering (ICDE)*, Istanbul, Turkey, 2007.

[8] M. R. Garey, D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness, " *W. H. Freeman & Co.* New York, NY, USA, 1990.

[9] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive Skyline Computation in Database Systems, " *ACM Transactions on Database Systems*, Vol. 30, No. 1, pp. 41–82, 2005.

[10] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting, " *In Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Bangalore, India, pp 717–719, Mar. 2003.

[11] I. Bartolini, P. Ciaccia, and M. Patella, "SaLSa: Computing the skyline without scanning the whole sky, " *In Proceedings of EDBT*, pp. 405-414, 2006.

[12] A. Guttman, "R-trees: A dynamic index structure for spatial searching, " *In Proceedings of the ACM Conference on the Management of Data (SIGMOD)*, pp. 47-57, 1984.