天際線巡視路徑查詢

王瑞評, 莊帛誠, 蔡秉辰, 林倚婕, 謝宇竣, 李柏寬, 陳麒永, 陳奕中
Department of Information Engineering and Computer Science,
Feng Chia University, Taichung 40724, Taiwan, R.O.C.
z2589906@gmail.com, d0240430@fcu.edu.tw, d0280208@fcu.edu.tw,
d0280212@fcu.edu.tw, d0240009@fcu.edu.tw, alex.860916@gmail.com,
macay12392@gmail.com, mitsukoshi901@gmail.com,

Abstract

工廠的巡視一直是工廠管理團隊的一大問題.因為工廠內需要巡視的地點可能很多,且常有突發情況的發生而需要特別巡視,但管理團隊的人力及時間都是有限的,所以不可能每天都巡視到工廠內所有的地點.因此如何替管理團隊規劃每天的巡視路線便成為一件非常重要的事情.有鑑於此,本論文定義了一個新的多條件 query-skyline visiting scheduling query,這個新的 query 可以把工廠內需要巡視的地點及其之間的聯絡路徑轉化成一個 Graph,且這個 graph 的每個點及邊上都會有一些權重值,分別代表每個地點及路徑的各種巡視條件,如時間成本,金錢成本,以及巡視的重要性等.而藉由這個新的 query 及目標的 graph,我們可以替工廠的巡視團隊找出幾條可能的最佳巡視路徑,幫助巡視團隊快速且有效的完成巡視工作.實驗結果則證實了我們所提出 query 的效率及有效性.

Key words: 天際線查詢, inspection route planning, graph

1 Introduction

工廠的巡視一直是工廠管理團隊的一大問題.因為工廠內需要巡視的地點可能很多,且常有突發情況的發生而需要特別巡視.舉例來說,圖一是一個大型工廠園區的各部門地圖,而部門與部門間的數值則代表管理團隊的移動成本,如金錢耗費及時間耗費.而管理團隊每天都要到各部門巡視,例如某一天管理團隊必須任選園區中的四個部門巡視,包含了資訊處 2hr,倉儲部門 3hr,公關部 4hr,以及生產部門 2hr,然而該管理團隊只有 8hr,可供巡視,這時候管理團隊就必須找出幾個合理的移動路徑來選擇,像是 A 路徑:資訊處→倉儲部門→生產部門,B 路徑:資訊處→生產部門→倉儲部門,及 C 路徑: 資訊處→公關部都是合理的巡視路徑,因為三條路徑都能在 8hr 內完成巡視(包含移動時間及巡視時間),但是 D 路徑:資訊處→倉儲部門→公關部卻不是,因為 D 路徑的

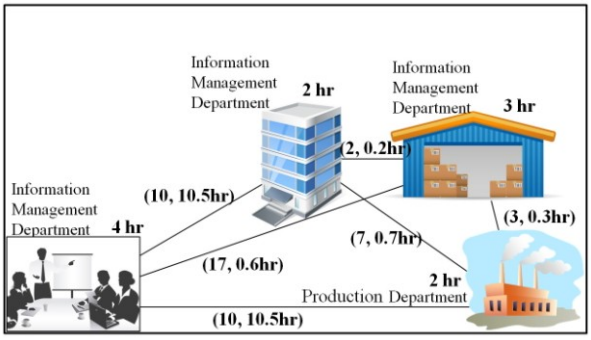

Fig. 1. An example of Factory inspections

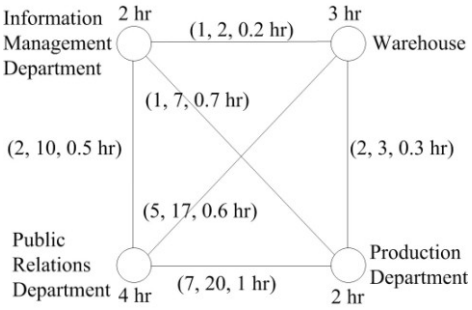

Fig. 2. The graph presentation of Fig. 1.

巡視時間超過 8hr.而對那些合理路徑來說,管理團隊一般也會從中做選擇,例如: 在 A 路徑上巡視的部門數,耗費金錢,時間, (3, $5, 7.5hr)比 B 路徑(3, $10, 8hr)來說省,所以管理團隊會選擇 A 路徑巡視而非 B 路徑.但對 A 路徑及 C 路徑來說,兩者是無法比較的,因為 A 路徑上移動的耗費金錢與時間(3, $5, 7.5hr)雖然比 C 路徑(2, $10, 6.5hr)來說省且巡視較多部門,但巡視時間卻比較長,所以管理團隊不管選擇 A 路徑還是 C 路徑都是合理的.

上述的例子我們將其稱為一個 skyline inspection route planning query.這個查詢將包含一個 graph,且 graph 上的每個 edge 及 point 都會有一些條件.這些條件可能是 multi-criteria 也可能是 single-criterion 的.例如,我們可將圖一中的工廠園區轉換成一個 graph,如圖二所示,且這個 graph 上的每個 point 代表一個部門,每個 edge 則代表移動的路徑,其中 point 上的數字代表巡視

Table 1. All of the possible combinations of inspection routes for Fig. 2.

Route	No. of dep.	Financial costs	Time costs
P1: Information → Warehouse → Production	3	5	7.5
P2: Information → Production → Warehouse	3	10	8
P3: Warehouse → Information → Production	3	9	7.9
P4: Warehouse → Production → Information	3	10	8
P5: Production → Information → Warehouse	3	9	7.9
P6: Production → Warehouse → Information	3	5	7.5
P7: Information → Public relations	2	10	6.5
P8: Public relations → Information	2	10	6.5
P9: Warehouse → Public relations	2	17	7.6
P10: Public relations →Warehouse	2	17	7.6
P11: Production → Public relations	2	20	7
P12: Public relations → Production	2	20	7

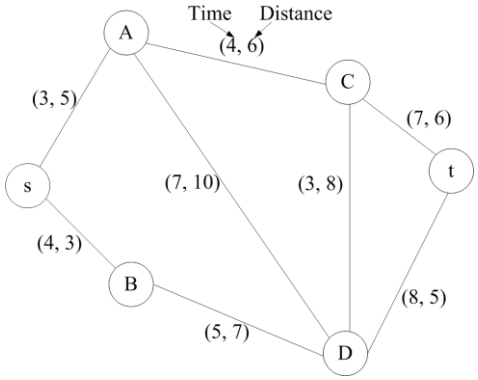

Fig. 3. An example of skyline query on road networks.

Table 2. All possible paths in Fig. 3 with their time costs and monetary costs

Path	Time cost	Money cost
P_1<s, A, C, t>	14	17
P_2<s, A, D, C, t>	20	29
P_3<s, A, D, t>	18	20
P_4<s, B, D, t>	15	15
P_5<s, B, D, C, t>	17	24

所需的時間(single-criterion),edge 上的數字則代表在部門間移動所需的金錢與時間(multi-criteria).在這個查詢中,user 會給予一個 multi-criteria 限制,並希望找出移動路徑的 multi-criteria 最佳解.例如在上述例子中,若我們將所有可能組合巡視路徑列於表一,並依據(巡視部門數,耗費金錢,及時間)三個條件下去選擇,我們會發現 P1, P6, P7,及 P8 為最佳解,因為沒有其他巡視路徑會比這些路徑好.

在過往,已經有一些論文針對路網上的天際線查詢進行研究.像是 Tian et al. [8], Kriegel et al. [5], and Chen and Lee [3] 所提出的查詢,都能夠在 user 給予一組起終點時,找出在多個條件下都符合 user 需求的移動路徑.然而,我們必須了解到這類型的查詢都是替特定的起終點組合找出中間可能的移動路徑,which 與本論文所提出的問題有相當大的差距.因為本論文所討論的問題並不考慮起終點,而是考慮路徑上各點的組合結果.這也就是說,過往路網上的各種已發展的天際線查詢並不能直接使用在本 work 中.此外,在過往天際線查詢的相關研究中,也有一些學者提出了 combinational 天際線查詢的概念[4][7],which 可針對同時選取多個物件的組合進行天際線查詢.例如: 股票市場上有 1000 支股票,user 希望從這 1000 支股票中購買兩支股票,which 可以獲得最大利益及最低風險,則他可以使用 combinational 天際線查詢來進行查詢.然而,我們必須了解到 combinational 天際線查詢所查詢的對象都是獨立的物件,which 與本論文必須考量地理上的不同地點及地點間連接方式有所不同,因此過往 combinational 天際線查詢的方法也無法直接應用於本論文的議題中.因此,一些新的演算法必須被提出來改善此一缺點.

本論文提出了一個新的演算法來解決目標的 skyline inspection route planning query.而實驗模擬則證實了本論文演算法的可行性.

本論文接下來的章節如下所示,第二章為 related work,第三章為演算法,第四章為實驗模擬,最後第五章為結論.

2 Related Work

這個部分列舉了兩個與本研究相關的研究,包含了(1)天際線查詢的相關研究,以及(2)skyline 路徑規劃之相關研究.

- A. Related work of 天際線查詢

常見的天際線查詢 演算法包括 Block Nested Loops 演算法[2], Divide and Conquer 演算法[2], Sort and Limit Skyline 演算法[1], Branch and Bound Skyline 演算法[6].然而,由於頁數的限制,以下我們僅簡單介紹 Branch and Bound Skyline 演算法的概念.

Branch and Bound Skyline 演算法 [6]是現在最常被提及的天際線查詢 演算法,因為在資料有建索引架構的情況下,他是被證實最有效率的方法,且其效率與其他不用索引架構的天際線查詢 演算法,如 Block Nested Loops 演算法, Divide and Conquer 演算法, and Sort and Limit Skyline 演算法好上許多,因為這三個演算法在 query 時,必須對資料集的所有點進行多次處理,並造成大量的 I/O cost,但對 Branch and Bound Skyline 演算法來說,他可以利用索引結構快速且大幅度的濾除那些不可能成為 skyline 結果的資料點,並因此加快處理速度.

- B. 天際線查詢應用在路網上的相關研究

此處我們使用圖三與表二來說明傳統 skyline 路徑規劃的定義,假設 G 為一無相圖且是一實際路網,每次的 skyline 路徑規劃查詢皆會提供起點 s 與終點 t,我們可發現會有多條路徑 p 介於 s 至 t,而每條路徑 p 上的各路段皆有時間與距離兩個條件, 將此 p 中所有路段上的各自條件做相加,用以表示成整段路徑的 multi-criteria 如表二.若現有兩條路徑 p1 與 p2,且 p1 在時間與距離所有

條件都優於 p2 時,即會表示成 p1 支配 p2.又如表中 p3 的時間雖少於 p1 但距離卻比 p1 長,此時會表示 p1 與 p3 互無法支配比較彼此,因此兩條都會是候選 skyline 路徑.最後所有條件皆不被其他路徑所支配的路徑(此例中為 p1 與 p3) 即為 skyline 路徑.

Tian et al. [8]首先介紹了 skyline 路徑的概念,他們提出的演算法會用在路網上的邊的參數,找出所有 user 特定出的起點和終點間的 skyline 路徑.演算法會先根據在路徑裡最低的每個參數值的總和,決定在起點和終點間的第一條 skyline 路徑,然後演算法會藉由下列步驟找出其他的 skyline 路徑,包含(1)使用貪婪演算法尋找在起點 s 和終點 t 之間可能的中繼點 a,(2)計算路徑<s,a>的實際參數值,還有預估路徑<a,t>的最小參數值.如果<s,a> + <a,t>的數值被已找到的 skyline 路徑支配,那麼 a 就不會是 skyline 路徑的一部分.而在 a 點的檢查結束,演算法會再次利用貪婪演算法去找出路網中其他可能的中繼點或找出可以連接在 a 之後的中繼點.

Kriegel et al. [5]是另外一篇經常被提及的 skyline 路徑相關論文,他們也利用貪婪演算法去找出在 s 和 t 之間的可能的中繼點 a.然而,他們認為[8]每次計算中繼點 a 與終點 t 之間的估計數值是非常緩慢的,因為 a 與 t 之間的距離可能非常遙遠,需要耗費大量的時間才能完成目標數值的估算.所以他們會在 offline 時就先設計許多參考點 x 去輔助估計.藉由這個方法，他們的演算法在估算<a,x>的時間將比[8]估算<a,t>的時間少上許多,並加快演算法的處理效率.

3 演算法

本論文演算法的流程圖,這個包含了一個 heap 與一個 list,其中這個 heap 會用來處理目前找到一半的尋訪路徑組合,而 list 則是用來儲存目前所找到的 skyline 尋訪路徑.以下,我們同時使用圖六為例解釋流程圖的內容,其中圖中共有 A 到 F 六個部門,且我們在演算法中會使用(•)來代表一個尋訪路徑,而這個路徑內字母的順序表示尋訪的順序.舉例來說,(A)這個路徑代表管理團隊僅需尋訪 A 部門.(A, B)這個路徑代表管理團隊必須需先尋訪 A 部門,再尋訪 B 部門.此外為了說明方便,以下的例子我們會使用拜訪部門數,移動成本及總尋訪時間來做為選擇 skyline 尋訪路徑的條件.例如,在(A, B)這條路徑中,拜訪部門數為 2,移動成本為拜訪 A, B 部門的成本加上在 AB 部門間移動的成本,總尋訪時間則為在 A, B 部門尋訪的時間加上在 AB 部門間移動的時間.

在演算法的一開始,我們會先找出所有的初始尋訪路徑,如圖六中,所有的初始尋訪路徑僅包含了(A),(B),(C),(D),(E),及(F),代表了在僅考慮初始拜訪部門下,管理團隊有先去 A,先去 B,..., 到先去 F 的 6 種選擇.而在找出所有初始尋訪路徑後,我們會將這些路徑依據拜訪部門數,移動成本及總尋訪時間,經特定公式計算後,由大至小的放入 heap 中.特別的,為了運算方便,此處的特定公式使用了天際線查詢中最常見的方式,將三個條件的結果分別正規化後相加,而相加後結果越小的路徑,其成為天際線查詢查詢結果的可能性越大,所以要優先檢查.

接著,我們會取出 heap 中的第一個路徑進行擴充.舉例來說,若 heap 中第一個路徑為(A, C),則我們會讓路徑(A, C)與該路徑尚未拜訪到的部門 B,D,E,F 這些部門組合,成為路徑(A, C, B), (A, C, D), (A, C, E), (A, C, F),若觀察這些擴充後的路徑,則我們可能獲得以下情況.

- Case 1. 所有擴充後的路徑都超過 user 給予的限制,則我們會將擴充前的路徑與 list 目前已知的 skyline 尋訪路徑相比.若擴充前在拜訪部門數,移動成本及總尋訪時間三個條件都被 list 內的路徑 dominate,則我們會將擴充前的路徑直接從 heap 中刪除,因為這個路徑不可能成為 skyline 尋訪路徑.反之,若擴充前在拜訪部門數,移動成本及總尋訪時間三個條件並沒有被 list 內的任一 skyline 尋訪路徑 dominate,則這條擴充前的路徑會被加入 list 內,因為他即是 skyline 尋訪路徑.

- Case 2. 若擴充後的路徑有一個以上未超過 user 給予的限制,則我們會將未超過限制的擴充後路徑加入 heap 中.其中這些路徑在加入 heap 時,仍會依據該路徑經特定公式計算後的結果由小到大排序.而對那些超過限制的擴充後路徑,我們則將他們直接從演算法中刪除,因為他們不可能成為最終結果.

而如此的步驟將持續的進行,直到 heap 中不再擁有任何組合為止.

4 模擬實驗

本章會使用一些模擬證實我們演算法的有效性,包含了資料點數目與查詢中條件數目對演算法的影響.

- A. 資料集設定

由於真實資料的取得困難,所以本章所使用的資料集為人工產生的虛擬資料集.在這個資料集中,我們會產生 10 點到 30 點的資料點來代表工廠園區的部門.同時,我們也會隨機產生個別部門及兩兩部門間各種條件的數值.特別的,我們此處產生了 2 到 5 種條件的資料,which 是最常被天際線查詢相關研究所討論的條件數量區間.

- B. 部門數量對演算法運算時間的影響

圖七為部門數量對運算時間的影響.特別的,由

於不同數量點下的演算法運算時間差異很大,所以圖七的y軸是採用對數座標.在該圖中,我們可以看到演算法花費時間在部門數目小於25時大致是成指數成長,但在部門數目超過 25 後,成長速度逐漸趨緩.究其原因有二,第一,在部門數目小於 25 時,用一條路徑拜訪所有的部門並不會超過我們給予的限制,所以所有可能尋訪路徑的數量大致會等於所有部門進行排列的結果,且這些尋訪路徑都必須被我們的演算法檢查,所以演算法的檢查時間大致會與所有可能的路徑數目成正比.又所有可能的路徑數量會隨部門數量成指數成長,所以檢查時間當然會隨指數成長.第二,在部門數目超過 25 時,已經不可能用一條路徑拜訪所有的部門,所以所有可能尋訪路徑的數量會少於所有部門進行排列的結果,which 代表所有可能尋訪路徑的數量會比指數成長減緩許多,所以我們演算法的檢查時間也會比指數成長減緩許多.

- C. 條件數目對演算法運算時間的影響

圖八為10個部門時,條件數量對運算時間的影響.明顯的,我們可以看到演算法運算時間大致是隨維度成指數成長.會有這個現象,主要是因為天際線查詢的結果個數通常會隨維度數量變多而成指數成長.又在我們的演算法中,每一條可能路徑都必須與所有的 skyline 結果進行支配比較.這也就是說,當天際線查詢的結果數量隨維度成指數成長時,我們演算法所花費的時間必然也會成指數成長.

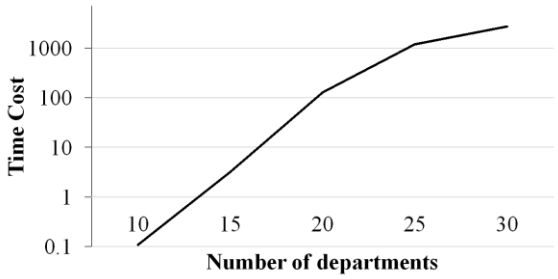

Fig.4 Number of departments and its influence on computation time

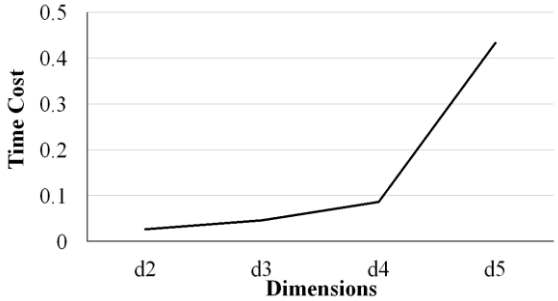

Fig. 5 The number of conditions and its effect on computation time

5 Conclusion

工廠的巡視一直是工廠管理團隊的大問題,因為管理團隊必須依照每天工廠情況的不同進行巡視.然而,工廠中有很多可能的巡視路徑,如何從中挑選最適合的變成為管理團隊最重要的課題.有鑑於此,本論文提出了 skyline inspection route planning query, which 可以協助工廠管理團隊在多個條件下找出最適合的路徑.此外,我們也提出了一個有效率的方法來進行查詢.本論文的實驗結果則證實了本方法的效率.

References

[1] I. Bartolini, P. Ciaccia, and M. Patella, "SaLSa: Computing the skyline without scanning the whole sky," *proceeding on ACM Int. Conf. on Information and Knowledge Management*, pp. 405-414, 2006.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," *proceeding on IEEE Int. Conf. on Data Engineering (ICDE)*, pp. 235-254, 2001.

[3] Y. C. Chen and C. Lee, "Skyline Path Queries with Aggregate Attributes, " *IEEE Access*, vol. 4, pp. 4690-4706, 2016.

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

[5] H. P. Kriegel, M. Renz, and M. Schubert."Route skyline queries: a multi-preference path planning approach" *proceeding on IEEE Int. Conf. on Data Engineering (ICDE)*, pp.261-272, 2010.

[6] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive 演算法 for skyline queries," *proceeding on ACM Special Interest Group on Management of Data*, 2003.

[7] I. F. Su, Y. C. Chung, C. Lee, "Top-k Combinatorial Skyline Queries," *proceeding on Int. Conf. on Database Systems for Advanced Applications (DASFAA)*, pp. 79-93, 2010.

[8] Y. Tian, K. C. K. Lee, and W. C. Lee, "Finding Skyline Paths in Road Networks," *proceeding on the ACM Int. Conf. on Advances in Geographic Information Systems (ACM SIGSPATIAL)*, pp. 444-447, 2009.