使用線性鬆弛技術解 *p*-中心位點問題

Using linear programming relaxation to solve the *p*-centdian problem

Tsai Chueh Wang[1], Yen Hung Chen[2], Yu Xiang Zhu[3]

Department of Computer Science,
University of Taipei, Taiwan
peter771109@gmail.com[1] , yhchen@utaipei.edu.tw[2], das781016@gmail.com[3]

摘要

給定一個無向圖$G(V,E,l)$，其中 l 是一個非負數函數表示該圖中每一個邊的長度，和一個正整數p，$0<p<|V|$，對於一個p個節點的集合V'，$|V'|=p$，$d_G(v,V')$為節點v與集合V'內最近節點的距離。在一個圖中，V'的離心率$\pounds_C(V')$定義為每個節點v離V'的距離中最遠的數值(即，$\pounds_C(V')=max_{v\in V}\ d_G(v,V')$)。$V'$的中位距離則定義為每個節點$v$離$V'$的距離值的總和(即，$\pounds_M(V')=\sum_{v\in V} d_G(v,V')$)。*p*-中心位點問題(*p*-centdian problem)定義為在圖G中找一個p個節點的集合P，使得加總$\pounds_C(P)$和$\pounds_M(P)$要最小。*p*-中心位點問題不難證明為NP-hard。本研究為此問題設計一個整數規劃(Integer programming)來找到此問題的最佳解，並利用線性規劃鬆弛(Linear programming relaxation)技術來加快其執行時間以及透過TSPLibrary(TSP-Lib)中的資料(節點個數在22~157範圍間)，來模擬測試此技術與最佳解之間的倍率及時間。實驗結果顯示我們的方法所得的解與最佳解的倍率大約為1.0至1.35之間，最高到3.15倍。在執行效率上，我們的線性規劃鬆弛演算法所用之時間平均為175至190秒（s）。

1 緒論

在組合最佳化問題中，*瓶頸問題*(*bottleneck problems*，或稱為 *min-max problems*)及*加總問題*(*min-sum* problems) 是兩種常用的*準則(criteria)*在許多通訊網路的設計(communication network design)、作業研究中企業或工廠上的倉儲(warehouse)及配送中心等服務規劃(service planning)、交通運輸網路上的緊急設施規劃(emergency facility planning)、工作排程(scheduling)、資源管理(resource management)、容量規劃(capacity planning)等應用上 [1]。瓶頸問題的顯著特性是會存在一個最差的(worst case)值或條件(稱 bottleneck)在整個(網路或企業)環境中，目的是希望找到最差的情況(bottleneck)要最小。一個非常著名的瓶頸問題即為*中心點問題*(*center problem*)。中心點問題最早可追朔到在 1869 年 Jordan [2]提出圖論上的中心點問題：在圖中找到一個節點(中心點)，使得圖中剩下來的節點可透過最短路徑連到此中心點最遠的距離要最小(即，*1-center problem*)。1965 年 Hakimi [3]擴展 1-center problem 到 *p-center problem*。給定一個無向圖$G(V,E,l)$，其中 l 是一個非負數函數表示該圖中每一個邊的長度和一個正整數 p，$0<p<|V|$，對於一個p個節點的集合V'，$|V'|=p$，$d_G(v,V')$為節點 v 與集合 V'內最近節點的距離。如果節點 v 為V'內的一個節點，則$d_G(v,V')=0$。在一個圖中，V'的離心率$\pounds_C(V')$定義為每個節點 v 離 V'的距離中最遠的數值(即，$\pounds_C(V')=max_{v\in V}\ d_G(v,V')$)。*p-中心點問題*(*p-center* (*pC*) *problem*)定義為在圖 G 中找到一個 p 個節點的集合 P，使得 P 的離心率要最小 [3]。*中位點問題*(*median problem*)類似於中心點問題，但要求的準則為最小加總(min-sum)。中位點問題是希望在圖中找到一個節點(中位點)，使得圖中每個剩下的節點可透過最短路徑連到此中位點的距離加總要最小(即，*1-median problem*)。Hakimi [3]同樣於 1965 年將問題擴展到 *p-median problem*。給定一個無向圖 $G(V,E,l)$，其中 l 是一個非負數函數表示該圖中每一個邊的長度和一個正整數 p，$0<p<|V|$，對於一個 p 個節點的集合 V''，V''的中位距離則定義為每個節點 v 離 V''的距離值的總和(即，$\pounds_M(V'')=\sum_{v\in V} d_G(v,V'')$)。*p-中位點問題*(*p-median* (*pM*) *problem*)定義為在圖 G 中找到一個 p 個節點的集合 P，使得 P 的中位距離要最小 [3]。*pC* 及 *pM* 問題可應用在設備配置(facility location) [3-18]與社交網路(social network)上 [19]。Halpern [20-21]提出了一種方式要同時滿足最大傳輸(延遲)時間及整體傳輸(延遲)時間的一種*雙準則*(*bicriteria*)問題：*中心位點*(*centdian*)問題。Hooker 等人 [22] 擴展 *centdian* 問題到 *p-centdian problem*。給定一個無向圖 $G(V,E,l)$，其中 l 是一個非負數函數表示該圖中每一個邊的長度，和一個正整數 p，$0<p<|V|$，*p-中心位點問題*(*p-centdian* (*pD*) *problem*)定義為在圖 G 中找一個 p 個節點的集合 P(稱為 *p*-中心位點)，使得加總$\pounds_C(P)+\pounds_M(P)$要最小 [22]。因為 *pC* 及 *pM* 問題為 NP-hard，不難證明 *pD* 問題也是 NP-hard 就算輸入的圖形是 metric graphs（完全圖並滿足三角不等式）下。在圖形為 metric graphs 下，Tamir 等人 [23]認為(只用一句話帶過)透過 Bartal [24-25]提出的在解 *pM* 問題的 randomized 演算法下，會有一個

期望值為 $O(log\ |V|\ log\ log\ |V|)$的近似演算法可解 *pD* 問題。Kalcsics, Nickel [26]則是提出了一個$O(log\ |V|\ log\ log\ |V|)$的近似演算法解*p-facility ordered median problems*（此問題不等同於我們提出的 *pM* 及 *pD* 問題）。因此目前並無任何 deterministic approximation algorithm 的論文被提出，對於 *pD* 問題。在正確演算法方面，Brito 和 Perez [27]設計了一個 $O(|V|^{p+2}|E|^p)$的正確演算法來找出 generalized *p*D 問題的最佳解。

本論文提出第一個整數規劃(Integer programming)演算法來找到 *pD* 問題的最佳解。接著我們利用此整數規劃做放寬成線性規劃，稱為線性規劃鬆弛(Linear programming relaxation)，再透過線性規劃求得的解(有小數)做兩個 round 的方法變成整數解來求得 *p*D 問題的合法解。之後觀察這兩個 round 的方法所得到的解與最佳解之間的關係(我們設計的方法所產生的合法解的數值除以整數規劃產生的最佳解的數值)。針對這三個方法，我們將使用 TSPLibrary(TSP-Lib) [28]的資料進行模擬。

本論文第二節我們將描述我們的整數規劃及線性鬆弛演算法來解決 *pD* 問題。第三節利用我使用一些 TSP-Lib 資料對第二節的演算法進行模擬(simulation)測試並觀察我們設計的演算法的效率以及正確性。*pD* 問題的未來研究方向則在第四節呈現。

2 *p*-centdian 問題的一些演算法

本節中，我們設計一個整數規劃的演算法來解決 *p*D 問題，找出其最佳解。接著再用鬆弛(relaxation)到整數條件式變成線性規劃，後再利用兩種 rounding 的技術找出問題的合法解。我們令 n 為圖 G 中的節點個數(即，$|V|$)。P 為 *p*D 問題最佳解(p 個節點的集合)。對於 *p*D 問題的一個合法解 P_f，如果節點 v，$v\in V$，連結到 P_f 內最近的節點u，則我們稱節點v被u 管(*assign*)。

底下兩小節我們敘述我們的整數規劃演算法及兩個線性鬆弛演算法的設計過程。

2.1 整數規劃對於 *p*-centdian 問題

此整數規劃是結合之前 Daskin [7]的書中所設計在 *p*M 及 *p*C 問題的整數規劃所產生。底下我們說明如何設計 *p*D 問題的整數規劃。

(1) 對於一個圖 $G=(V,E)$，圖內有 n 個節點，分別為節點 *1*，節點 *2*，…，節點 n。

(2) 對於所有的j，$1\le j\le n$，變數 y_j 對應到圖中的每個節點j，如果節點j 是在合法解 P_f內，則對應的 y_j 設為 *1*，否則為 *0*。

(3) 對於所有的 i, j，$1\le i\le n$，$1\le j\le n$，變數 l_{ij} 為一個 2 維矩陣紀錄節點 i 到節點 j 的最短路徑，l_{ij}計算可透過 Floyd-Warshall Algorithm 解 all-pair shortest path problem 得到此矩陣的數值 [29]。

(4) 對於所有的 i, j，$1\le i\le n$，$1\le j\le n$，變數 x_{ij} 為一個 2 維矩陣，如果節點 i 是被節點 j 管(assign)，則 x_{ij} 設為 *1*，否則為 *0*。

(5) 變數 z，為一個合法解 P_f的離心率 $\pounds_C(P_f)$

*p*D 問題的整數規劃 (I)

Minimize $\Sigma_{i=1\ to\ n}\Sigma_{j=1\ to\ n}\, l_{ij}x_{ij}+z$-------------------(6)

Subject to $\Sigma_{j=1\ to\ n}\, x_{ij}=1$，$\forall i$，$1\le i\le n$--------(7)

$\Sigma_{j=1\ to\ n}\, y_j=p$---------------------------(8)

$x_{ij}\le y_j$，$\forall i,j$，$1\le i\le n$，$1\le j\le n$----(9)

$\Sigma_{j=1\ to\ n}\, l_{ij}x_{ij}\le z$，$\forall$ i，$1\le i\le n$-----(10)

$x_{ij}, y_j\in\{0,1\}$-------------------------(11)

底下我們描述我們的**整數規劃演算法(Integer programming algorithm):**

輸入: 一個無向完全圖 $G=(V,E,l)$，l 是一個非負數函數(需滿足三角不等式)表示該圖中每一個邊的長度，及一個數值 p（$p\ge 1$）。

輸出: p-中心位點 P，$|P|=p$。

Step 1: 使用整數規劃(I)找出 p 個節點集合 P 使得$\pounds_C(P)+\pounds_M(P)$ *為最小*。

Step 2: Return P。

利用上述整數規劃是可以找出 *p*D 問題的最佳解。然而整數規劃是 NP-hard [29]，目前並無多項式時間的正確演算法解決此問題，所以我們將此問題的整數條件(11) relaxation，使得此整數規劃 (I)變成線性規劃(II)。

2.2 線性規劃鬆弛對於 *p*-centdian 問題

*p*D 問題的線性規劃 (II)

Minimize $\Sigma_{i=1\ to\ n}\Sigma_{j=1\ to\ n}\, l_{ij}x_{ij}+z$------------------(12)

Subject to $\Sigma_{j=1\ to\ n}\, x_{ij}=1$，$\forall i$，$1\le i\le n$---------(13)

$\Sigma_{j=1\ to\ n}\, y_j=p$-------------------------(14)

$x_{ij} \leq y_j$，$\forall i,j$，$1\leq i\leq n$，$1\leq j\leq n$--(15)

$\Sigma_{j=1 \text{ to } n}\, l_{ij} x_{ij} \leq z$，$\forall$i，$1\leq i\leq n$-----(16)

$0\leq x_{ij}, y_j \leq 1$----------------------------(17)

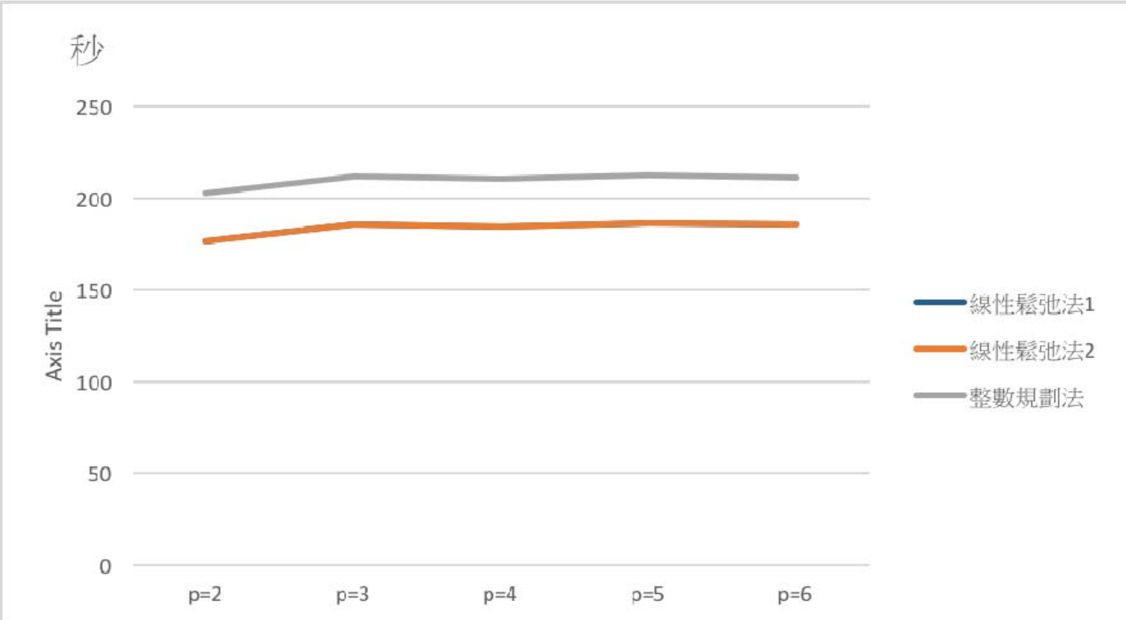

圖 1：輸入為 TSP-Lib 內的完全圖，節點個數在 22~157 個範圍間的情形，線性鬆弛演算法 1、線性鬆弛演算法 2(線性鬆弛演算法 1、2 時間相同，故重疊為一條線)與整數規劃演算法之耗時比較（單位：秒）。圖內的數值是分別執行 25 組資料後的平均時間。

由於線性規劃可執行在多項式時間 [30]，而且所得到的解其值會小於等於整數規劃所得到的解的值(是其下限)。因為用線性規劃(II)我們得到 y_j 的值會是小數，$1\leq j\leq n$，所以我們無法得到 pD 問題的合法解。因此我們的做法是採用 round 的技術，取出經由線性規劃產生的最佳解的 y_j 數值後，將 y_j 數值排序後，取前 p 大或前 p 小的數值，將這些對應的節點設為 1。我們的兩個線性鬆弛的演算法如下:

線性鬆弛演算法 1 (Linear programming relaxation algorithm 1):

輸入: 一個無向完全圖 $G=(V,E,l)$，l 是一個非負數函數(需滿足三角不等式)表示該圖中每一個邊的長度，及一個數值 p（$p\geq 1$）。

輸出: p-中心位點 P'，$|P'|=p$。

Step 1: 使用線性規劃(II)找出滿足此線性規劃的最佳解。

Step 2: 將變數 y_j 的數值排序，$1\leq j\leq n$，取出前 p 大的數值，後將其對應的節點放入 P'。

Step 3: Return P'。

第二個演算法只是將選取前 p 小的數值的節點取代前 p 大的數值。

線性鬆弛演算法 2 (Linear programming relaxation algorithm 2):

輸入: 一個無向完全圖 $G=(V,E,l)$，l 是一個非負數函數(需滿足三角不等式)表示該圖中每一個邊的長度，及一個數值 p（$p\geq 1$）。

輸出: p-中心位點 P''，$|P''|=p$，

Step 1: 使用線性規劃(II)找出滿足此線性規劃的最佳解。

Step 2: 將變數 y_j 的數值排序，$1\leq j\leq n$，取出前 p 小的數值，後將其對應的節點放入 P''。

Step 3: Return P''

很明顯的線性鬆弛演算法 1 及 2 都可以執行在多項式時間。兩個演算法都是先用線性規劃找出問題在線性規劃下的最佳解，然後再對變數 y_j 的數值排序，$1\leq j\leq n$，差別在於演算法 1 是取出前 p 大的數值，但演算法 2 是取出前 p 小的數值。

3 實測驗證

本節中，我們將針對第二節所設計的演算法來進行實際的程式模擬。我們藉由整數規劃的正確演算法(exact algorithm)找到 pD 問題的最佳解 P 並計算其$\pounds_C(P)+\pounds_M(P)$的值。接下來我們實作線性鬆弛演算法 1 和線性鬆弛演算法 2 來產生 pD 問題的合法解 P'和 P''並計算$\pounds_C(P')+\pounds_M(P')$和$\pounds_C(P'')+\pounds_M(P'')$的值。藉由合法解的數值除以最佳解的數值來找出這兩個演算法的*倍率*。實驗模擬採用 Eclipse 撰寫 Python 程式，作業系統平臺為 64 位元 OS X Yosemite version 10.10.5。處理器使用 Intel Core i5，時脈 2.9GHz，記憶體為 16GB。實驗數據採用 TSPLibrary(TSP-Lib) [28] 資料來進行模擬測試，因為 TSP-Lib 是許多計算的問題再進行模擬程式實驗室時常用的 benchmark。用二維陣列儲存無向完全圖之節點及其距離。我們從 TSP-Lib 選取 25 組資料，因為 TSP-Lib 內的每個圖其節點數大多不一樣，所以這 25 組測試資料我們選擇圖的節點個數在 22~157 的範圍間。演算法為第二節所提到的線性鬆弛演算法 1、線性鬆弛演算法 2 與正確演算法所產生的解。正確演算法是利用整數規劃演算法得到的最佳解。p 的設定範圍為 2 到 6 個節點之間，結果為 25 組測資執行後所得到的解的平均值。

為了說明我們的演算法所產生的解與最佳解之間並不會相距太遠，表 1.分別列出了線性鬆弛演算法 1 與線性鬆弛演算法 2 所產生出的解與最佳解之間的倍率(我們演算法所得到的合法解的數值除以最佳解的數值)。()內為其標準差。線性鬆弛演算法 1 的倍率幾乎接近 1。線性鬆弛演算法 2 在 p 設為 2~6 時得到的倍率平均值最高到 1.35，顯示出線性鬆弛演算法 1 有較好的效率(選出線性規劃解中的前 p 大的節點)。而且線性鬆弛演算法 1 跟最佳解非常接近。在最佳解的結果方面，線性鬆弛演算法 2 會因為 p 增加而導致最佳解倍率增加，但 p 對線性鬆弛演算法 1 的影響較小，大致是維持在一定倍率內。而線性鬆弛演算法 1 在整體近似倍率上也能提供比線性鬆弛演算法 2 更接近最佳解。雖然我們未能以數學證明我們的解距離最佳解的倍率，然而實驗數據顯示我們的解不失為一個好的結果。

p	2	3	4	5	6
倍率(線性鬆弛演算法 1)	1.0027 (0.006)	1.0063 (0.031)	1.0052 (0.015)	1.0075 (0.006)	1.0038 (0.004)
倍率(線性鬆弛演算法 2)	1.0343 (0.165)	1.0999 (0.191)	1.2117 (0.348)	1.2524 (0.343)	1.3522 (0.5149)

表 1. pD 問題的倍率模擬結果(找出最佳解分別與線性鬆弛演算法 1 及線性鬆弛演算法 2 的解間的差異)，針對輸入為在 TSP-Lib 內的完全圖，節點數在 22~157 的範圍時情形，()內為其標準差。倍率為我們演算法所得到的合法解的數值除以最佳解的數值。每個數據的值是執行 25 組資料後的平均值

在時間分析上，圖 1.可以觀察到線性鬆弛演算法 1、線性鬆弛演算法 2 與整數規劃演算法在執行時的效率。橫座標是 p，縱座標是執行的時間(秒)。Note：線性鬆弛演算法 1、2 時間會是一樣，是因為我們的這兩個演算法是都透過解線性規劃後，對 y_j 的數值排序，$1\leq j\leq n$，差別只在線性鬆弛演算法 1 是取出前 p 大的數值而線性鬆弛演算法 2 是取出前 p 小的數值。因此這兩個線性鬆弛演算法的時間會是一樣。在 p=2 時，整數規劃演算法在執行上平均需花 202.4502 秒、線性鬆弛演算法 1 及 2 的平均時間為 176.6884 秒。在 p=6 時，整數規劃演算法平均需花 211.3155 秒、線性鬆弛演算法 1 及 2 平均需花 185.4813 秒。實驗結果顯示，p 設定為 2 到 6 個節點時，線性鬆弛演算法 1 和 2 的執行時間是相同的(差在取點方式不同)，但跟整數規劃演算法相比，執行時間較快。線性鬆弛演算法 1 與線性鬆弛演算法 2 的時間平均落在 175 至 187 秒之間。

4 未來研究方向

本研究主要是分析及設計整數規劃及線性鬆弛的演算法針對p-中心位點問題(p-centdian problem)。我們設計了第一個整數規劃的演算法，並透過簡單的線性鬆弛技術找到問題的合法解。實驗顯示我們的線性鬆弛演算法1倍率約為1.0至1.007之間，最高到1.15倍。線性鬆弛演算法2倍率約為1.0至1.35之間，最高到3.15倍。離最佳解相差不大然而執行的時間比整數規劃找最佳解的演算法快。未來我們希望能找到一個此整數規劃的duality，並利用primal and dual的技術對此問題設計一個近似演算法。

Acknowledgements

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 105-2221-E-845-002 。Corresponding author: Yen Hung Chen。

Reference

[1] D.S. Hochbaum, A. Pathria, Generalized p-center problems: complexity results and approximation algorithms. *European Journal of Operational Research 100*, *594–607*, *1997*.

[2] C. Jordan, Sur les assemblages des lignes. *Journal für die reine und angewandte Mathematik 70*, *185-190*, *1869*.

[3] S.L. Hakimi, Optimal distribution of switching centers in a communication network and some related theoretic graph theoretic problems. *Operations Research 13*, *462–475*, *1965*.

[4] B. Ben-Moshe, A. Dvir, M. Segal, A. Tamir, Centdian computation for sensor networks. In Proceedings of *the 7th Annual Conference Theory and Applications of Models of Computation*, *LNCS 6108*, *187–198*, *2010*.

[5] B. Ben-Moshe, A. Dvir, M. Segal, A. Tamir, Centdian computation in cactus graphs. *Journal of Graph Algorithms and Applications 16*, *199–224*, *2012*.

[6] G.G. Cornuéjols, G.L. Nemhauser, L.A. Wolsey, The uncapacitated facility location problem. In P.Mirchandani and R. Francis, editors, *Discrete Location Theory, John Wiley and Sons, Inc.*, *New York*, *119–171*, *1990*.

[7] M.S. Daskin, Network and Discrete Location: Models Algorithms and Applications. Wiley, New York, *1995*.

[8] M.S. Daskin, What you should know about location modeling. *Naval Research Logistics 55*, *283–294*, *2008*.

[9] Z. Drezner, H.W. Hamacher, Facility Location Applications and Theory. 2nd edition, Springer, *2004*.

[10] D.S. Hochbaum, D.B. Shmoys, A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM 33*, *533–550*, *1986*.

[11] D.S. Hochbaum, A. Pathria, Generalized p-center problems: complexity results and approximation algorithms. *European Journal of Operational Research 100*, *594–607* , *1997*.

[12] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems I: the p-centers. *SIAM Journal of Applied Mathematics 37*, *514–538*, *1979*.

[13] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems II: the p-medians. *SIAM Journal of Applied Mathematics 37*, *539–560*, *1979*.

[14] J. Mihelič, B. Robič, Facility location and covering problems. In Proceedings of *the 7th International Multiconference Information Society*, *vol. D–Theoretical Computer Science*, *Ljubljana*, *Slovenia*, *2004*.

[15] P.B. Mirchandani, R.L. Francis, Discrete Location Theory. Wiley, New York, *1990*.

[16] C.S. Revelle, H.A. Eiselt, Location analysis: A synthesis and survey. *European Journal of Operational Research 165*, *1–19*, *2005*.

[17] C.S. Revelle, H.A. Eiselt, M.S. Daskin, A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research 184*, *817–848*, *2008*.

[18] B.C. Tansel, R.L. Francis, T.J. Lowe, Location on networks: a survey. Part I: the p-center and p-median problems. *Management Science 29*, *482–497*, *1983*.

[19] A. Chaudhury, P. Basuchowdhuri, S. Majumder, Spread of information in a social network using influential nodes. *Advances in Knowledge Discovery and Data Mining*, *LNCS 7302*, *121–132*, *2012*.

[20] J. Halpern, The location of a center-median convex combination on an undirected tree. *Journal of Regional Science 16*, *237–245*, 1976.

[21] J. Halpern, Finding minimal center-median convex combination (cent-dian) of a graph. *Management Science 24*, *535–544*, *1978*.

[22] J.N. Hooker, R.S. Garfinkel, C.K. Chen, Finite dominating sets for network location problems. *Operation Research 39*, *100–118*, *1991*.

[23] A. Tamir, D. Perez-Brito, J.A. Moreno-Perez, A polynomial algorithm for the p-centdian problem on a tree. *Networks 32*, *255–262*, *1998*.

[24] Y. Bartal, Probabilistic approximation of metric spaces and its algorithmic applications. In Proceedings of *the 37th Annual IEEE Symposium on Foundations of Computer Science*, *184–193*, *1996*.

[25] Y. Bartal, On approximating arbitrary metrics by tree metrics. In Proceedings of *the 30th Annual ACM Symposium on Theory of Computing*, *161–168* , *1998*.

[26] J. Kalcsics, S. Nickel, J. Puerto, Multifacility ordered median problems on networks: a further analysis. *Networks 41*, *1-12*, *2003*.

[27] D.P. Brito, J.A.M. Perez, The generalized p-centdian on network. *Top 8*, *265-285*, *2000*.

[28] G. Reinelt, TSPLIB—A traveling salesman problem library. *INFORMS Journal on Computing 3*, *376–384*, *1991*.

[29] T. H. Cormen, C.E. Leiserson. R.L. Rivest, C. Stein, Introduction to Algorithms. Third Edition. The MIT Press. Cambridge, Massachusetts, *2009*.

[30] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica 4*, *373–395*, *1984*.