啟發式演算法在 *p*-中心位點問題

Heuristic algorithms for the *p*-centdian problem

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

Department of Computer Science,
University of Taipei, Taiwan
dsa781016@gmail.com[1], yhchen@utaipei.edu.tw[2], peter771109@gmail.com[3]

摘要

給定一個無向圖$G(V,E,l)$及一個非負數的長度函數l在邊上，和一個正整數p，$0<p<|V|$，對於一個p個節點的集合V'，$|V'|=p$，而V'以外的每個節點可透過最短路徑連到V'內最近的節點，用符號$d_G(v,V')$表示。而一個圖的離心率$£_C(V')$對於一個p個節點的集合V'，定義為V'以外的每個節點到其V'內最近的節點中最遠的那一段距離(路徑)的值(i.e., $£_C(V')=max_{v\in V\setminus V'} d_G(v,V')$)。一個圖的中位距離$£_M(V')$對於一個$p$個節點的集合$V'$，則定義為$V'$以外的每個節點到其$V'$內最近的節點中的距離(路徑)值加總(i.e., $£_M(V')=\sum_{v\in V\setminus V'} d_G(v,V')$)。$p$-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$£_C(P)+£_M(P)$要最小。$p$-中心位點問題可應用在設備配置(facility location)上需同時滿足最大傳輸時間(離心率)及整體傳輸時間(中位距離)的條件。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exhaustive search的正確演算法來找到此問題的最佳解，並採用local search技術來設計兩個啟發式演算法以加快其執行時間，並透過TSPLibrary(TSP-Lib)資料模擬測試這兩個啟發式演算法與最佳解之間的倍率及時間，實驗結果顯示我們的演算法所得的解與最佳解的倍率大約為1.017至1.046之間，最高到1.21倍。在執行效率上，我們的啟發式演算法1所用之時間平均為2.7至17毫秒（ms）及啟發式演算法2所用之時間平均為1.5至7毫秒（ms）。

1 緒論

本計畫所要研究的是 *p-中心點問題*(*p-Center* (*pC*) *problem*) [1,2,3]與 *p-中位點問題*(*p-Median* (*pM*) *problem*) [1,3,4]的 *雙準則網路設計問題* (*bicriteria network design problems*)：*p-中心位點問題*(*p-centDian* (*pD*) *problem*) [5,6]。給定一個無向圖 $G(V,E,l)$及一個非負數的長度函數 l 在邊上和一個正整數 p，$0<p<|V|$，對於一個 p 個節點的集合 V'，$|V'|=p$，而 V'以外的每個節點可透過最短路徑連到 V'內最近的節點，用符號 $d_G(v,V')$表示。而一個圖的*離心率*$£_C(V')$對於一個 p 個節點的集合 V'，定義為 V'以外的每個節點到其 V'內最近的節點中最遠的那一段距離(路徑)的值(i.e., $£_C(V')=max_{v\in V\setminus V'} d_G(v,V')$)。一個圖的中位距離$£_M(V')$對於一個 p 個節點的集合 V'，則定義為 V'以外的每個節點到其 V'內最近的節點中的距離(路徑)值加總(i.e., $£_M(V')=\sum_{v\in V\setminus V'} d_G(v,V')$)。*p-中心點*(*p-Center* (*pC*)) 問題定義為在圖 G 中找一個 p 個節點的集合 P，使得 P 的離心率要最小 [1,2,3]。*p-中位點*(*p-Median* (*pM*))問題定義為在圖 G 中找一個 p 個節點的集合 P，使得 P 的中位距離要最小 [1,2,4]。*pC* 及 *pM* 問題有許多的應用在設備配置(facility location) [1-4,7-15]與社交網路(social network)關鍵人物(influential person)選定及分析 [16,17]上。然而 *pC* 與 *pM* 問題的最佳解在設備配置問題上都有可能造成 *tradeoff*，要整體的傳輸(延遲)時間(中位距離)最短但是最大傳輸(延遲)時間(離心率)可能會很長或是最大傳輸(延遲)時間(離心率)很短但是整體的傳輸(延遲)時間(中位距離)卻很長。Halpern [18-19]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 [5]等人擴展 centdian 問題到 p-centdian problem。給定一個無向圖 $G(V,E,l)$及一個非負數的長度函數 l 在邊上，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，使得$£_C(P)+£_M(P)$要最小 [5,6]。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 [6]宣稱透過 Bartal [20,21]提出的在解 *pM* 問題的 randomized 演算法下，*p*D 問題將可得到一個近似率的期望值為 $O(log\ |V|\ log\ log\ |V|)$ 的 randomized 近似演算法。本論文我們將提出兩個簡單的啟發式演算法(heuristic algorithm)去解 *pD* 問題，並觀察其執行時間且實作窮舉的方法找出最佳解來觀察我們的啟發式演算法所得的解與最佳解之間的倍率（我們的演算法所得出的解的值除以最佳解的值）。我們的啟發式演算法將使用 TSPLibrary(TSP-Lib) [22]內的資料進行模擬。

本論文第二章我們將描述我們的啟發式演算

法來解決 *pD* 問題。第三章利用我們的啟發式演算法對一些 TSP-Lib 內的資料進行模擬(simulation)測試，藉此說明我們的啟發式演算法與正確演算法的關係。最後第四章為結論及說明 *pD* 問題的未來研究方向。

2 啟發式演算法應用於 *p*-centdian 問題

本章中，我們設計兩個啟發式演算法來解決 *pD* 問題，找出問題的合法解。底下我們敘述我們演算法的設計過程

我們的演算法是採用簡單的 local search [23]的技術。此方法的精神是先選取一個合法解(feasible solution)，接下來找尋某個在合法解外的節點加入合法解並移除原先合法解中的某個節點，產生新的合法解。如果此新的合法解的距離小於原先合法解就替換，否則就再找到合法解外的節點重複下去，直到不能再透過替換一個節點便可改善距離就結束。此方法用在設備配置的 *p*-median 問題上會得到一個 5 倍的近似解。然而用在設備配置的 *p*-center 問題的近似率為 $O(\log |V|)$ [23,24] (Note. 設備配置的 *p*-median & *p*-center 問題不同於本論文中定義的 *p*-median & *p*-center 問題)。我們想要實作此演算法在 *pD* 問題下會有什麼結果。底下我們列出我們的 local search 演算法針對 *pD* 問題。我們用 *len(P)*來表示一個 *p*D 問題的合法解 *P*(*p* 個節點)其$£_C(P)+£_M(P)$的值。

Algorithm Local Search.

1. $P \leftarrow$ any feasible solution for the *p*-centdian problem
2. **While** $\exists s \in P$ and $s' \notin P$ such that
   $len(P - s + s') < len(P)$,
   do $P \leftarrow P - s + s'$
3. **Return** *P*

Arya 等人證明此演算法有 5 倍的近似率針對設備配置的 *p*-median 問題 [23]。然而此方法的時間複雜度會是指數時間。因此他們將步驟 2 修正為

2. While $\exists s \in P$ and $s' \notin P$ such that
   $len(P - s + s') < (1-\varepsilon)\, len(P)$,
   do $P \leftarrow P - s + s'$

很簡單的觀察會發現修正後此演算法執行的時間是跟 input size 和 ε^{-1} 成多項式關係。

現在，我們詳細的描述我們的 Local search 演算法如下

啟發式演算法 1 (Heuristic Algorithm 1) :

輸入: 一個無向完全圖 $G=(V,E)$，每個邊上有非負數之長度函數(需滿足三角不等式)，以及數值 p（$p \geq 1$）。

輸出: 一個節點集合 P，$|P|=p$，

Step 1: 任選 V 內的 p 個節點形成一個節點集合 P 並計算$£_C(P)+£_M(P)$。

Step 2: **While** ∃ a vertex $s \in P$ and a vertex $s' \notin P$ such that $£_C(P - s + s')+£_M(P - s + s') < £_C(P)+£_M(P)$
do $P \leftarrow P - s + s'$

Step 3: **Return** *P*

很明顯地此演算法一定會停止，當我們每次改良一個合法解直到找到 *P* 為 *pD* 問題的最佳解時。然而因為此演算法會一直執行，直到沒有 s'能夠與 s 交換，最糟可能會執行到指數時間，所以我們也用 Greedy 方式稍微修正啟發式演算法 1 得到啟發式演算法 2 (Greedy local search)如下:

啟發式演算法 2 (Heuristic Algorithm 2) :

輸入: 一個無向完全圖 $G=(V,E)$，每個邊上有非負數之長度函數(需滿足三角不等式)，以及數值 p（$p \geq 1$）。

輸出: 一個節點集合 $P=\{s_1,s_2,...s_P\}$，$|P|=p$，

Step 1: 任選 V 內的 p 個節點形成一個節點集合 $P=\{s_1,s_2,...,s_p\}$並計算$£_C(P)+£_M(P)$。

Step 2: **For** $i=1$ to p **do**
For *each* vertex $s' \notin P$ **do**
Step 2.1
If $£_C(P-s_i+s')+£_M(P-s_i+s') < £_C(P)+£_M(P)$
Then
Step 2.1.1 $P \leftarrow P - s_i + s'$
Step 2.1.2 $s_i \leftarrow s'$
EndIf
EndFor
EndFor

Step 3: **Return** *P*

很明顯地，此演算法的時間複雜度為:$O(p|V|^3)$。

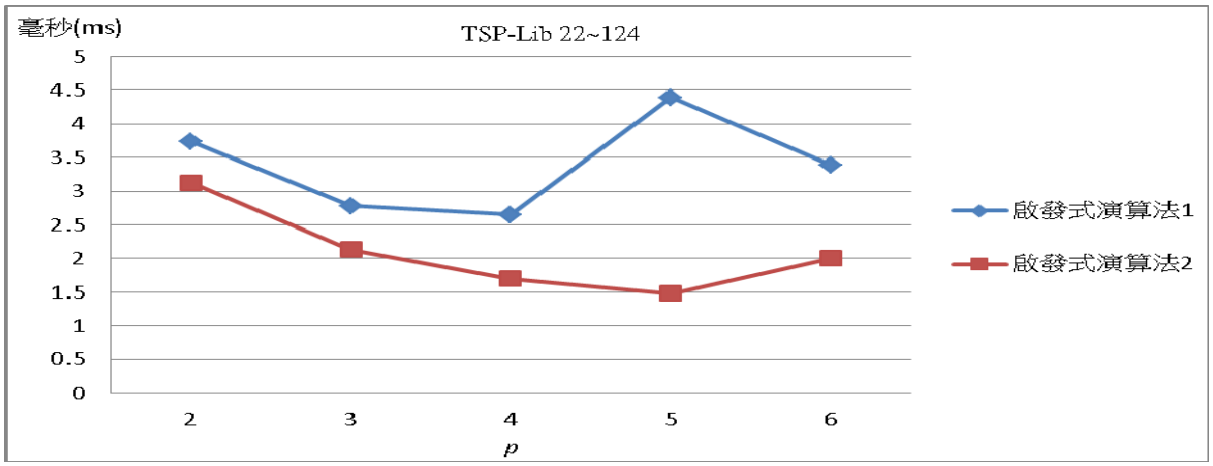

圖 1：輸入為在 TSP-Lib 內的完全圖，節點數在 22~124 個範圍時，啟發式演算法 1、啟發式演算法 2 耗時比較（單位：毫秒）。每個點的值是執行 23 組資料後的平均值。

3 實例驗證

本章中，我們將針對第二章所設計的演算法來進行實際的程式模擬。我們藉由 exhaustive search 的正確演算法(exact algorithm)、啟發式演算法 1 與啟發式演算法 2 進行時間效率的分析，我們也會觀察兩個啟發式演算法所產生的 $£_C(P)+£_M(P)$ 的值與最佳解的值之間的倍數（近似率）關係。近似率(ratio)是拿我們設計的啟發式演算法所得出的解的值除以最佳解的值。正確演算法是利用暴力法窮舉所有 p 個節點組合得到的最佳解。本次模擬實驗使用 Eclipse 撰寫 Java 程式，作業系統平臺為 64 位元 Microsoft Windows 7 SP1。處理器使用 Intel Core i5-6500，時脈 3.2GHz，記憶體為 8GB。實驗是以 TSPLibrary(TSP-Lib) [22]資料來模擬測試。我們分別使用 22~124 個節點與 127~299 個節點之完全圖（長度符合 metric）並用二維陣列儲存無向完全圖之節點及其距離。第一小節中，我們模擬的情形是使用 22~124 個節點的完全圖(TSP-Lib 22~124)。第二小節中，我們模擬的情形是使用 127~299 個節點的完全圖(TSP-Lib 127~299)。

3.1 圖的節點數為 22~124 時的案例

本小節中我們實測當輸入的圖有 22~124 個節點時，啟發式演算法 1、啟發式演算法 2 與正確演算法所產生的解之間的關係。模擬將 p 設定為 2 到 6 個節點的情形，實驗中欲輸入的圖我們取自 TSP-Lib。因為 TSP-Lib 內的圖大致節點數都不一樣，所以我們在當中選取出 23 組完全圖，其節點範圍在 22~124 之間並將執行後的結果取平均。

表 1.分別列出了啟發式演算法 1 與啟發式演算法 2 所產生出的解與最佳解之間的近似率(ratio)的平均值，()內為標準差。實驗結果顯示這兩個演算法的近似率十分接近 1。而且啟發式演算法 1 比啟發式演算法 2 好，這是相當直覺的，不過啟發式演算法 2 最高會到 1.21 倍，這樣的結果看似也可以接受。雖然我們未能以數學證明我們的解離最佳解的倍率，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
近似率(啟發式演算法 1)	1.0048 (0.0131)	1.0047 (0.0159)	1.0090 (0.0136)	1.0106 (0.0126)	1.0202 (0.0359)
近似率(啟發式演算法 2)	1.0196 (0.0366)	1.0209 (0.0266)	1.0457 (0.0478)	1.0274 (0.0175)	1.0382 (0.0359)

表 1. pD 問題的近似率模擬結果(找出最佳解分別與啟發式演算法 1 及啟發式演算法 2 的解間的差異)，針對輸入為在 TSP-Lib 內的完全圖，節點數在 22~124 個範圍時的情形。近似率為我們這兩個啟發式演算法的解的值除以最佳解的值。每個數據的值是執行 23 組資料後的平均值，()內為標準差。

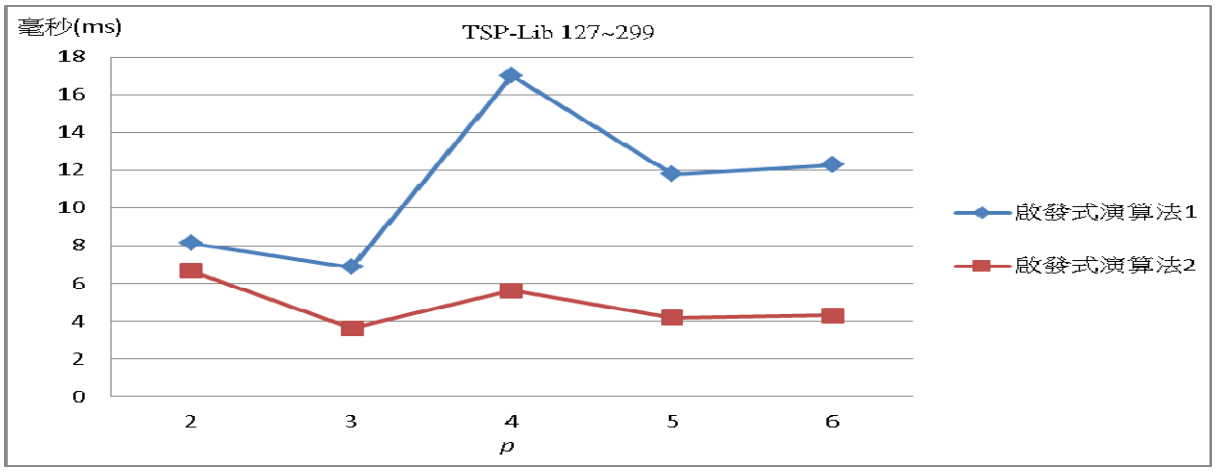

圖 2：輸入為在 TSP-Lib 內的完全圖，節點數在 127~299 個範圍時，啟發式演算法 1、啟發式演算法 2 之耗時比較（單位：毫秒）。每個點的值是執行 23 組資料後的平均值。

圖 1.列出了啟發式演算法 1 與啟發式演算法 2 在時間上的比較結果，因正確演算法執行的時間差距非常大，所以未將其時間列於圖中。橫座標是 p，縱座標是執行的時間(毫秒)。在 p 為 2 時，正確演算法在執行上平均需花 15.609 毫秒、啟發式演算法 1 平均需花 3.739 毫秒，啟發式演算法 2 平均需花 3.13 毫秒。在 p 為 5 時，正確演算法平均需花 8059697.391 毫秒、啟發式演算法 1 平均需花 4.391 毫秒，而啟發式演算法 2 平均僅花 1.478 毫秒。啟發式演算法 1 平均落在 2.7 至 4.4 毫秒之間，而啟發式演算法 2 則平均落在 1.5 至 3.1 毫秒之間，啟發式演算法 2 在執行速度上是優於啟發式演算法 1 的。而且最差時會差距 18 倍以上。然而實驗結果顯示啟發式演算法 2 其合法解的值也是相當接近最佳解的值，且在時間上明顯有效率。

3.2 圖的節點數為 127~299 時的案例

本小節中我們實測當輸入的圖有 127~299 個節點時，啟發式演算法 1、啟發式演算法 2 與正確演算法產生的解之間的關係。p 仍是設定為 2 到 6 個節點，從 TSP-Lib 中選取出 23 組完全圖，其節點範圍在 127~299 之間並將執行後的結果取平均。

表 2.分別列出了啟發式演算法 1 及啟發式演算法 2 所產生出的解與最佳解之間的近似率(ratio)，()內為標準差。實驗結果顯示啟發式演算法 1 的近似率較低但啟發式演算法 2 平均的近似率也不會高於 1.04 倍。兩個演算法結果都算是不錯的。

p	2	3	4	5	6
近似率(啟發式演算法 1)	1.0051 (0.0124)	1.0070 (0.0136)	1.0078 (0.0123)	1.0045 (0.0059)	1.0145 (0.0189)
近似率(啟發式演算法 2)	1.0173 (0.0356)	1.0250 (0.0281)	1.0284 (0.0298)	1.0284 (0.0220)	1.0392 (0.0267)

表 2. pD 問題的近似率模擬結果(找出最佳解分別與啟發式演算法 1 及啟發式演算法 2 的解間的差異)，針對輸入為在 TSP-Lib 內的完全圖，節點數在 127~299 個範圍時的情形。近似率為我們這兩個啟發式演算法的解的值除以最佳解的值。每個數據的值是執行 23 組資料後的平均值，()內為標準差。

圖 2.列出了啟發式演算法 1 與啟發式演算法 2 兩個演算法在時間上的比較結果。圖的節點數為在 127~299 的範圍，橫座標是 p，縱座標是執行的時間(毫秒)。在 p 為 2 時，正確演算法平均需花 75.652 毫秒，啟發式演算法 1 平均需花 8.13 毫秒，而啟發式演算法 2 需要 6.652 毫秒。在 p 為 4 時，正確演算法平均需花 5518357.783 毫秒，啟發式演算法 1 平均需花 17 毫秒，而啟發式演算法 2 平均僅需 5.609 毫秒。啟發式演算法 1 與啟發式演算法 2 的時間明顯低於正確演算法，這應該是很直覺的，因為正確演算法的時間複雜度成指數成長。分別觀察 3.1 小節與 3.2 小節可以顯示，這兩個啟發式演算法的近似率(倍率)似乎不受圖內節點個數多少而影響。

4 結論及未來研究方向

本研究探討 p-中心位點問題(p-centdian problem)，我們對此問題設計 2 個啟發式演算法來找出問題的合法解。透過實驗顯示啟發式演算法 1(Local search)的倍率大約為 1.005 至 1.02 之間。啟發式演算法 2(Greedy local search)的倍率大約為 1.017 至 1.046 之間。這兩個演算法跟最佳解間的倍率不受圖內節點個數多少而影響。未來我們希望能找出理論證明我們的啟發式演算法 1 有一個常數倍的近似解，並且設計一個正確演算法，其時間可以比暴力演算法來得快。

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] 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*.

[2] 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*.

[3] 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*.

[4] 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*.

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

[6] 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*.

[7] 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*.

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

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

[10] S.L. Hakimi, Optimal locations of switching centers and the absolute centers and medians of a graph. *Operations Research 12*, *450–459*, 1964.

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

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

[13] P.B. Mirchandani, R.L. Francis, Discrete Location Theory. Wiley, New York (1990).

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

[15] A. Tamir, Improved complexity bounds for center location problems on networks by using dynamic data structures. *SIAM Journal of Discrete Mathematics 1*, *377–396*, *1988*.

[16] 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*.

[17] T. Takaoka, Effcient algorithms for the 2-center problems. In Proceedings of *the 13th International Conference on Computational Science and Applications*, *LNCS 6017*, *519–532*, *2010*.

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

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

[20] 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*.

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

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

[23] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit, Local search heuristics for *k*-median and facility location problems. *SIAM Journal on Computing 33*, *544–562*, *2004*.

[24] A. Gupta, K. Tangwongsan, Simpler analyses of local search algorithms for facility location. *arXiv:0809.255*, *2008*.