The Connected p-Center Problem on Cactus Graphs

Chunsong Bai[1], Liying Kang[2], and Erfang Shan[3(✉)]

[1] Fuyang Normal College, Fuyang 236041, P.R. China
csbai@fync.edu.cn
[2] Department of Mathematics, Shanghai University, Shanghai 200444, P.R. China
lykang@shu.edu.cn
[3] School of Management, Shanghai University, Shanghai 200444, P.R. China
efshan@i.shu.edu.cn

Abstract. In this paper, we study a variant of the p-center problem on cactus graphs in which the p-center is asked to be connected, and this problem is called the *connected p-center problem*. For the connected p-center problem on cactus graphs, we propose an dynamic programming algorithm and show that the time complexity is $O(n^2p^2)$, where n is number of vertices.

Keywords: Location problem · Connected p-center problem · Cactus graph · Dynamic programming

1 Introduction

This paper concerns the connected p-center location problem on cactus graphs. Given a simple graph $G = (V, E)$ with n vertices and m edges, a classical p-center problem on a graph $G = (V, E)$ is to determine a p-vertex set V_p in G such that the maximum distance between V_p and V is minimized.

The p-center problem on an arbitrary graph has been known to be NP-hard [3,4]. Olariu [5] presented an $O(n)$ time algorithm for the 1-center problem on interval graphs. Tamir [6] showed that the weighted and un-weighted p-center problems on networks can be solved in $O(n^p m^p \log^2 n)$ time and $O(n^{p-1} m^p \log^3 n)$ time, respectively. Frederickson [2] showed how to solve this problem for trees in optimal linear time using parametric search.

The connected p-center problem is proposed by Yen and Chen [7]. They showed that the CpC problem is NP-hard even when the underlying graph is a bipartite graph or a split graph, and gave an $O(n)$ time algorithm to solve the problem on tree graphs. In [8], Yen proved that the CpC problem on block graph is NP-hard even when (1) $w(v) = 1$, for all $v \in V$, and $l(e) \in \{1, 2\}$, for all $e \in E$, and (2) $w(v) \in \{1, 2\}$, for all $v \in V$, and $l(e) = 1$, for all $e \in E$, respectively.

Research was partially supported by NSFC (grant numbers 11571222, 11471210).

T-H.H. Chan et al. (Eds.): COCOA 2016, LNCS 10043, pp. 718–725, 2016.
DOI: 10.1007/978-3-319-48749-6_53

2 Notations and Basic Properties

Let $G = (V, E)$ be a simple cactus graph, where each vertex $v \in V$ is associated with a unit weight $w(v) = 1$ and each edge $e \in E$ is associated with a length $l(e) > 0$. Denote by $P[u, v]$ the shortest path in G from u to v, $u, v \in V$.

In order to facilitate the overview of the proposed algorithms for the center problems in cactus networks, we start with the well-known tree structure of a cactus network [1]. The vertex set V is partitioned into three different subsets: C-vertices, G-vertices and hinges.

It is easy to see that a cactus consists of blocks, which are either a cycle or a graft. Thus, we can use a tree T_G to represent the skeleton over G, where each element in T_G represents a block or a hinge of G.

To make the tree T_G ready for use as intended, we convert it into a *rooted tree* as follows: We pick an arbitrary block, e.g., B_0, as the "root" of T_G. For each block B in T_G, we define the *level* $Lev(B)$ of B to be the number of edges on $P[B, B_0]$. Denote by $L = \max_{B \in T_G}\{Lev(B)\}$. If it exists, the *father* of a block B is always a hinge h, called its *companion hinge*. For simplicity, we pick an arbitrary vertex $h_0 \in B_0$ as the *virtual* companion hinge of B_0. Denote by B_h the block B whose companion hinge is h.

For each block B_h in T_G, denote by G_h the sub-cactus of G induced by the vertices of B_h and all sub-cacti hanging from B_h. Specially, $G = G_{h_0}$. For each hinge h of G_{h_0}, denote by g_h the vertex of $G_{h_0} \setminus G_h$ which is the farthest to h. Denote by $g(h) = d(h, g_h)$.

Let $\delta_{G_h}(V_k)$ be the *maximum weighted distance from a k-vertex set V_k to a sub-cactus G_h*, that is,

$$\delta_{G_h}(V_k) = \max_{u \in V(G_h)}\{w(u)d(u, V_k)\},$$

where $d(u, V_k) = \min_{v \in V_k} d(u, v)$.

The Connected p-Center (CpC) Problem: Given a connected graph $G = (V, E)$ and a positive integer $p \geq 2$, identify a p-vertex set $V_p \subseteq V$ such that $\delta_G(V_p)$ is minimized under the restriction that the subgraph induced by V_p is connected. V_p is called a *connected p-center* of G.

For each graft B_h, we define a problem $P(G_h, v, k)$: Given a vertex v of B_h and a positive integer $k \leq p$, identify a connected k-vertex set $V(G_h, v, k)$ of G_h such that $\delta_{G_h}(V(G_h, v, k))$ is minimized, under the restriction that v is the closest vertex to h in $V(G_h, v, k) \cap V(B_h)$. $V(G_h, v, k)$ is called a v-restricted connected k-center of G_h.

For each cycle B_h with s indexed vertices $v_1 = h, v_2, \ldots, v_s$, we define a problem $P(G_h, \{v_i, v_j\}, k)$ ($P^{co}(G_h, \{v_i, v_j\}, k)$): Given two vertices $v_i, v_j \in V(B_h)$ with $i \leq j$, and a positive integer $k \leq p$, identify a connected k-vertex set $V(G_h, \{v_i, v_j\}, k)$ ($V^{co}(G_h, \{v_i, v_j\}, k)$) of G_h such that $\delta_{G_h}(V(G_h, \{v_i, v_j\}, k))$ ($\delta_{G_h}(V^{co}(G_h, \{v_i, v_j\}, k))$) is minimized, under the restriction that $V(G_h, \{v_i, v_j\}, k) \cap V(B_h)$ contains only the vertices of the path from v_i to v_j on B_h in clockwise (counter-clockwise) direction. $V(G_h, \{v_i, v_j\}, k)$

($V^{co}(G_h, \{v_i, v_j\}, k)$) is called a $\{v_i, v_j\}$-restricted clockwise (counter-clockwise) connected k-center of G_h.

For all sub-cacti G_h, denote by $\mathcal{V}_1$ (resp. $\mathcal{V}_2$) the set of $V(G_h, v, p)$ (resp. $V(G_h, \{v_i, v_j\}, p)$ and $V^{co}(G_h, \{v_i, v_j\}, p)$).

Lemma 1. *There exists a connected p-center of G_{h_0} in $\mathcal{V}_1 \cup \mathcal{V}_2$.*

Proof. Let V_p be a connected p-center of G_{h_0}. We assume that $v \in V_p$ is the closest vertex to h_0, and B_h is the block that contains v. We distinguish the following two cases.

Case 1. B_h is a graft of G_{h_0}, assume that $V(G_h, v, p)$ is a v-restricted connected p-center of G_h. It is easy to see that:

$$\begin{aligned}\delta_{G_{h_0}}(V_p) &= \max\{\delta_{G_h}(V_p), d(v,h) + g(h)\}\\ &\geq \max\{\delta_{G_h}(V(G_h, v, p)), d(v,h) + g(h)\}\\ &= \delta_{G_{h_0}}(V(G_h, v, p)),\end{aligned}$$

which implies $V(G_h, v, p)$ is also an optimal solution to CpC problem.

Case 2. B_h is a cycle of G_{h_0}. W.l.o.g., we only consider the case $V_p \cap V(B_h)$ contains only the vertices of the path from v_i to v_j on B_h in clockwise direction, where $i \leq j$ (the other case can be handled similarly). Assume that $V(G_h, \{v_i, v_j\}, p)$ is a $\{v_i, v_j\}$-restricted connected p-center of G_h. By the similar discussion in Case 1, we have:

$$\begin{aligned}\delta_{G_{h_0}}(V_p) &= \max\{\delta_{G_h}(V_p), \max\{d(v_i,h), d(v_j,h)\} + g(h)\}\\ &\geq \max\{\delta_{G_h}(V(G_h, \{v_i, v_j\}, p)), \max\{d(v_i,h), d(v_j,h)\} + g(h)\}\\ &= \delta_{G_{h_0}}(V(G_h, \{v_i, v_j\}, p)),\end{aligned}$$

which implies $V(G_h, \{v_i, v_j\}, p)$ is also an optimal solution to CpC problem. □

Based on Lemma 1, we are going to devise an algorithm to identify all restricted connected p-centers in $\mathcal{V}_1 \cup \mathcal{V}_2$.

3 Algorithm for the *CpC* Problem on Cactus Graphs

3.1 Procedure GRAFT(B, h)

Given a graft $T = B_h$. Root T at the vertex h. Let $leaf(T)$ be all leaves of T. For each vertex v of T, we define the *level* $lev(v)$ of v to be the number of edges on $P[h, v]$ and $L'_m = \max_{v \in V(T)} lev(v)$. If $v \neq h$, then by removing the last edge of $P[h, v]$, we obtain two subtrees of T. Let T_v be the subtree that contains v, and let $T_v^c = T \setminus T_v$. Similarly, we let G_v be the subgraph of G_h induced by the vertices of T_v and the sub-cacti hanging from T_v, and $G_v^c = G_h \setminus G_v$.

For each vertex v in T, let $E(v)$ be the edges of T_v which are adjacent to v. Denote by $s(v) = |E(v)|$. We define an arbitrary order among the edges of $E(v)$, and we denote the lth edge in $E(v)$ by $e(v,l)$. If v_l is the other endpoint of the $e(v,l)$, then we say that v_l is the lth *son* of v, and v is the father $fa(v_l)$ of v_l. Denote by $son(v)$ be the sons of v. Denote by $T_{e(v,l)}$ the maximal connected subgraph of T_v which contains v but does not contain any edge $e(v,j)$ for $j > l$. In particular, $T_{e(v,0)} = v$ and $T_{e(v,s(v))} = T_v$. Similarly, we define $G_{e(v,l)}$ to be the subgraph of G_v induced by the vertices of $T_{e(v,l)}$ and all sub-cacti hanging from $T_{e(v,l)}$.

Let $e(v,l)$ be an arbitrary edge of T. Let $S(e(v,l),k)$ be a connected k-vertex set of G_v which contains v but does not contain any vertex $v_j \in son(v)$ for $j > l$. Then we define a partial distance-value of $S(e(v,l),k)$ over $G_{e(v,l)}$.

Definition 1. *Let $e(v,l)$ be any arbitrary edge of T. For each positive integer k, $1 \leq k \leq \min\{p, |G_{e(v,l)}|\}$, we define:*

$$R^*(e(v,l),k) = \min_{S(e(v,l),k) \subseteq G(e(v,l))} \delta_{G_{e(v,l)}}(S(e(v,l),k)).$$

The corresponding set to $R^*(e(v,l),k)$ is denoted by $S^*(e(v,l),k)$.

Next, for the vertices in G_v^c, we define the value $R^*(G_v^c, v)$ as follow:

$$R^*(G_v^c, v) = \delta_{G_v^c}(v),$$

which is in fact the distance-value of the 1-center v over G_v^c.

Once we obtain the values $R^*(e(v,s(v)),k)$ and $R^*(G_v^c, v)$, the distance-value of $V(G_h, v, k)$ can be computed as:

$$\delta_{G_h}(V(G_h, v, k)) = \max\{R^*(e(v,s(v)),k),\ R^*(G_v^c, v)\}. \tag{1}$$

According to our assumption, when the block B_h to be processed, we can assign for each v in $leaf(T)$ the following values. For each vertex v of degree 0 in G, we assign:

$$R^*(e(v,0),1) = 0$$

and

$$S^*(e(v,0),1) \leftarrow \{v\}.$$

For each vertex v which is the companion hinge of some block B_v, if B_v is a graft, we assign:

$$R^*(e(v,0),k) = \delta_{G_v}(V(G_v, v, k))$$

and

$$S^*(e(v,0),k) \leftarrow V(G_v, v, k).$$

Otherwise, we assign:

$$R^*(e(v,0),k) = \delta_{G_v}(V(G_v, \{v,v\}, k))$$

and

$$S^*(e(v,0),k) \leftarrow V(G_v, \{v,v\}, k).$$

The Computation of $R^*(e(v,l),k)$ and $S^*(e(v,l),k)$. We assume that, when the jth stage begins, the value $R^*(e(v,s(v)),k)$ has been computed for each vertex $v \in T$ of level $lev(v) \geq L'_m - j + 1$. During the jth stage, we search through all vertices of level $L'_m - j$. For each such a vertex v, we compute all values $R^*(e(v,s(v)),k)$ and go on the next vertex of level $L'_m - j$.

Let v be a vertex of level $L'_m - j$. We start by assigning:

$$R^*(e(v,0),1) = \max_{u \in son(v)} \{d(v,u) + R^*(e(u,0),1)\}.$$

Assume that we already known the values $R^*(e(v,l'),k)$ for all $l' < l$, and we now compute the value $R^*(e(v,l),k)$ as follows:

$$R^*(e(v,l),k) = \min\Big\{ \min_{0 \leq k' \leq k-1} \max\{R^*(e(v,l-1),k'), R^*(e(v_l,s(v_l),k-k')\}, \\ \max\{R^*(e(v,l-1),k), \ell(v,v_l) + R^*(e(v_l,s(v_l)),1)\}\Big\}. \quad (2)$$

On the right-hand side of (2), the first term corresponds to $v_l \in S^*(e(v,l),k)$, and the second term corresponds to $v_l \notin S^*(e(v,l),k)$.

If $v_l \in S^*(e(v,l),k)$, assign:

$$S^*(e(v,l),k) \leftarrow S^*(e(v,l-1),k'') \cup S^*(e(v_l,s(v_l)),k-k''), \quad (3)$$

where k'' is the number such that the first term of the right-hand side of (3) is minimized. Otherwise, assign:

$$S^*(e(v,l),k) \leftarrow S^*(e(v,l-1),k).$$

We can compute all values $R^*(e(v,l),k)$ by passing through all edges in T. Note that there are at most $|T|p$ values $R^*(e(v,l),k)$ must be computed, each of those computations involved the finding of a minimum over at most $2k$ terms. Thus, the total time complexity is $O(|T|p^2)$.

The Computation of $R^*(G^c_v, v)$. The value $R^*(G^c_v, v)$ can be computed by using the distances matrix of T and the values $R^*(e(u,0),1), u \in leaf(T)$, that is:

$$R^*(G^c_v, v) = \max_{u \in leaf(T)} \{d(v,u) + R^*(e(u,0),1)\}.$$

It is easy to see that the total time is $O(|T|^2)$ to compute the values $R^*(G^c_v, v)$.

3.2 The Procedure CYCLE(C, h)

Let C be the cycle B_h with s clockwise indexed vertices $v_1 = h, v_2, \ldots, v_s$. For any pair $v_i, v_j \in V(C)$ $(i \leq j)$, denote by C_{v_i,v_j} ($C^{co}_{v_i,v_j}$) the subgraph induced by the vertices of the path from v_i to v_j in clockwise direction (in counter-clockwise direction), and G_{v_i,v_j} ($G^{co}_{v_i,v_j}$) the subgraph induced by C_{v_i,v_j} ($C^{co}_{v_i,v_j}$) and the sub-cacti hanging from it. Let $G^c_{v_i,v_j} = G_h \setminus G_{v_i,v_j}$.

The Computation of $V(G_h, \{v_i, v_j\}, k)$. Let $V(\{v_i, v_j\}, k)$ be a connected k-vertex set of G_{v_i,v_j} that contains the vertices v_i and v_j. Then we define the partial distance-value of $V(\{v_i, v_j\}, k)$ over G_{v_i,v_j} as follow:

$$R_1^*(\{v_i, v_j\}, k) = \min_{V(\{v_i,v_j\},k)\subseteq G_{v_i,v_j}} \delta_{G_{v_i,v_j}}(V(\{v_i, v_j\}, k)).$$

Let $e_{m(j,i)}$ be the edge that contains the midpoint of the path from v_j to v_i in clockwise direction. Particularly, if the midpoint happens to be a vertex, then it coincides with $v_{m(j,i)}$. By deleting the edge $e_{m(j,i)}$ from $G^c_{v_i,v_j}$, we obtain two subgraphs $G^{c,1}_{v_i,v_j}$ and $G^{c,2}_{v_i,v_j}$, which contain $v_{m(j,i)}$ and $v_{m(j,i)+1}$, respectively. Now we define the following values:

$$R_2^*(\{v_i, v_j\}, v_j) = \delta_{G^{c,1}_{v_i,v_j}}(\{v_j\})$$

and

$$R_3^*(\{v_i, v_j\}, v_i) = \delta_{G^{c,2}_{v_i,v_j}}(\{v_i\})$$

to represent the partial distance-values of v_j and v_i, respectively.

Once we obtain all values defined above, the distance-value of $V(G_h, \{v_i, v_j\}, k)$ can be computed as:

$$\delta_{G_h}(V(G_h, \{v_i, v_j\}, k)) = \max\{R_1^*(\{v_i, v_j\}, k),\ R_2^*(\{v_i, v_j\}, v_j),\ R_3^*(\{v_i, v_j\}, v_i)\}.$$

Note that the values $R_1^*(\{v_i, v_j\}, k)$ can be computed by applying the procedure GRAFT(B, h), and the total time is $O(|C|^2p^2)$.

Given an edge $e_m = (v_m, v_{m+1})$ in C. Let $\mathcal{P}(e_m) = \{\{v_{l_1}, v_{r_1}\}, \{v_{l_2}, v_{r_2}\}, \ldots,$ $\{v_{l_t}, v_{r_t}\}\}$ be all vertex pairs of C with their middle edges are e_m, where $l_1 \geq l_2 \geq \ldots \geq l_t$. Let $\mathcal{V} = \{v_{l_1}, v_{l_2}, \ldots, v_{l_t}\}$.

Because of the recursion

$$\begin{aligned} R_2^*(\{v_{r_k}, v_{l_k}\}) = \max\{&R_2^*(\{v_{r_{k-1}}, v_{l_{k-1}}\}, v_{l_{k-1}}) + d(v_{l_k} + v_{l_{k-1}}),\\ &\max_{l_{k-1}>j'\geq l_k}\{d(v_{l_k}, v_{j'}) + R_1^*(\{v_{j'}, v_{j'}\}, 1)\}\}, \end{aligned} \tag{4}$$

we can calculate all values $R_2^*(\{v_{r_k}, v_{l_k}\}, v_{l_k})$ for $l_1 \leq l_k \leq l_t$ by passing through all vertices in $\mathcal{V}$ and cost $O(|C|)$ time for comparing and adding operations. Thus, all values can be computed in $O(|C|^2)$ time since there $O(|C|^2)$ values must be computed and $O(|C|)$ edges in C.

The Computation of $V^{co}(G_h, \{v_i, v_j\}, k)$. Let $V^{co}(\{v_i, v_j\}, k)$ be a connected k-vertex set of $G^{co}_{v_i,v_j}$ that contains the vertices v_i and v_j, let $e_{m(i,j)}$ be the edge that contains the midpoint of the path from v_i to v_j in clockwise direction. Particularly, if the midpoint happens to be a vertex, then it coincides with $v_{m(i,j)}$.

Next we define the partial value:

$$R_4^*(\{v_i, v_j\}, k) = \min_{V^{co}(\{v_i,v_j\},k)\subseteq G^{co}_{v_i,v_j}} \delta_{G^{co}_{v_i,v_j}}(V^{co}(\{v_i, v_j\}, k)),$$

as well as the values $R_5^*(\{v_i, v_j\}, v_i)$ and $R_6^*(\{v_i, v_j\}, v_j)$, similar to $R_2^*(\{v_i, v_j\}, v_j)$ and $R_3^*(\{v_i, v_j\}, v_i)$, respectively. Therefore, we can compute the distance-value of $V^{co}(G_h, \{v_i, v_j\}, k)$ as:

$$\delta_{G_h}(V^{co}(G_h, \{v_i, v_j\}, k)) = \max\{R_4^*(\{v_i, v_j\}, k),\ R_5^*(\{v_i, v_j\}, v_i),\ R_6^*(\{v_i, v_j\}, v_j)\}.$$

It is easy to see that all values $R_4^*(\{v_i, v_j\}, k)$, $R_5^*(\{v_i, v_j\}, v_i)$, $R_6^*(\{v_i, v_j\}, v_j)$ can be computed similarly as above, and the time complexity is $O(|C|^2p^2)$.

3.3 Algorithm for the *CpC* Problem

By Lemma 1, we can now identify a connected p-center V_p^* from $\mathcal{V}_1 \cup \mathcal{V}_2$. The distance-value of V_p^* can be computed by the following relation:

$$\begin{aligned}\delta(V_p^*) = \min\Big\{ &\min_{V(G_h,v,p)\in\mathcal{V}_1} \{\max\{\delta_{G_h}(V(G_h, v, p)), d(v, h) + g(h)\}\},\\ &\min_{V(G_h,\{v_i,v_j\},p)\in\mathcal{V}_2} \max\{\delta_{G_h}(V(G_h, \{v_i, v_j\}, p)), \max\{d(v_j, h), d(v_i, h)\} + g(h)\},\\ &\min_{V^{co}(G_h,\{v_i,v_j\},p)\in\mathcal{V}_2} \max\{\delta_{G_h}(V^{co}(G_h, \{v_i, v_j\}, p)), g(h)\}\Big\}. \qquad (5)\end{aligned}$$

We can now formulate the algorithm for the *CpC* problem.

Algorithm 1. Connected_p-Center_on_Cactus_Graphs.

Input: A cactus graph $G(h_0)$, the corresponding skeleton $T_S(B_0)$ and its maximal level L_m.
Output: A connected p-center V_p^* and its distance-value.

for $i = 1; i <= L_m; i++$ **do**
  for *each block* B *of level* $2L_m - 2i + 1$ **do**
    if B *is a graft* **then**
      Let h be the companion hinge of B, let L'_m be the maximal level of B;
      for $j = 1; j <= L'_m; j++$ **do**
        Call **GRAFT**(B, h) to compute the values $V(G_h, v, k)$ for each vertex v of level j and $1 \le k \le \min\{p, |G_v|\}$;
      end
    end
    if B *is a cycle* **then**
      Let h be the companion hinge of C;
      Call **CYCLE**(B, h) to compute the values $V(G_h, \{v_i, v_j\}, k)$ and $V^{co}(G_h, \{v_i, v_j\}, k)$ for all pair $v_i, v_j \in V(C)$ $(i \le j)$ and all possible numbers k;
    end
  end
end
return Identify a connected p-center V_p^* by using the Eq. (5).

As a preprocessing for Algorithm 1, we first compute the distance-matrix of the given cactus. Then we find a skeleton of the given cactus and compute $g(h)$ for each companion hinge h in the skeleton. This preprocessing requires $O(n^2)$ steps. Then we can find a p-center from $\mathcal{V}_1 \cup \mathcal{V}_2$ by using the binary search method.

Theorem 1. *The CpC problem on a cactus graph of* n *vertices can be solved in* $O(n^2p^2)$ *time.*

4 Conclusions

In this paper we consider the connected p-center on graphs. We devise a dynamic programming algorithm of the complexity $O(n^2p^2)$ for the problem on cactus graphs. In the future, it is very meaningful to extend our algorithm to other classes of graphs, such as interval graphs, circular-arc graphs and planar graphs, etc.

References

1. Burkard, R.E., Krarup, J.: A linear algorithm for the pos/neg-weighted 1-median problem on cactus. Computing **60**, 498–509 (1998)
2. Frederickson, G.: Parametric search and locating supply centers in trees. In: Dehne, F., Sack, J.-R., Santoro, N. (eds.) WADS 1991. LNCS, vol. 519, pp. 299–319. Springer, Heidelberg (1991)
3. Garey, M.R., Johnso, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Bell Laboratories, Murray Hill, Freeman & Co., New York (1978)
4. Kariv, O., Hakimi, S.L.: An algorithmic approach to network location problems, part II: p-medians. SIAM J. Appl. Math. **37**, 539–560 (1979)
5. Olariu, S.: A simple linear-time algorithm for computing the center of an interval graph. Int. J. Comput. Math. **24**, 121–128 (1990)
6. Tamir, A.: Improved complexity bounds for center location problems on networks by using dynamic data structures. SIAM J. Discrete Math. **1**, 377–396 (1988)
7. Yen, W.C.-K., Chen, C.-T.: The p-center problem with connectivity constraint. Appl. Math. Sci. **1**, 1311–1324 (2007)
8. Yen, W.C.-K.: The connceted p-center problem on block graphs with forbidden vertices. Theor. Comput. Sci. **426–427**, 13–24 (2012)