Werk

Label: Article

Jahr: 1987

PURL: https://resolver.sub.uni-goettingen.de/purl?312901348_50-51|log7

Kontakt/Contact

Digizeitschriften e.V.
SUB Göttingen
Platz der Göttinger Sieben 1
37073 Göttingen

info@digizeitschriften.de

UNIVERSITAS COMENIANA
ACTA MATHEMATICA UNIVERSITATIS COMENIANAE
L—LI—1987

THE p-CENTER PROBLEM IN A UNICYCLIC GRAPH

MOHAMED HASSAN, Bratislava

1. Introduction

Let $G = G(V, E)$ be a finite connected graph, n be the number of its vertices, and $|E|$ be the number of its edges. We assume that a distance matrix on G is given. We construct efficiently a minimal spanning tree T of the graph G such that a 1-center of Tcoincide with a 1-center of G. We also dexcribe an algorithm of complexity $O(n)$ finding a dominating set of radius r, and algorithm of complexity $O(n^3 . \lg n)$ ($O(n^2 \lg n)$, respectively) finding an absolute (a vertex, respectively) p-center in a vertex-weighted unicyclic graph for $1 < p < n$.

Let G be a connected undirected graph with a nonnegative number $w(v)$ (called the weight of vertex v) associated with each of its $|V| = n$ vertices, and a postivive number $l(e)$ (called the length of edge e) associated with each of its $|E|$ edges.

Let $X_p = \{x_1, x_2, \ldots, x_p\}$ be a set of p points on G, where by a point on G we mean a point along any edge of G which may or may not be a vertex of G. We define the distance $d(v, X_p)$ between a vertex v of G and a set X_p in G by

$$d(v, X_p) = \min_{1 \leq i \leq p} \{d(v, x_i)\}, \tag{1.1}$$

where $d(v, x_i)$ is the length of a shortest path in G between the vertex v and the point x_i. Let

$$F(X_p) = \max_{v \in V} \{w(v) . d(v, X_p)\}. \tag{1.2}$$

Let X_p^* be such that

$$F(X_p^*) = \min_{X_p \text{ on } G} \{F(X_p)\}. \tag{1.3}$$

Then X_p^* is called an absolute p-center of G and $F(X_p^*)$ is called the absolute p-radius of G and is usually denoted by r_p.

If X_p and X_p^* in (1.3) are restricted to be sets of p vertices of G, then X_p^* is called a vertex p-center and $F(X_p^*)$ is called thevertex p-radious of G.

If all the vertices of the graph $G(V, E)$ have the same wight c, then without loss of generality we shall assume that $c = 1$ and we refer to this case as the vertex-unweighted case. Otherwise, we say that $G(V, E)$ is a vertex-weighted graph. We shall assume that $p < n$, since if $p = n$, then $X_p^* = V$, $r_p(G) = 0$, while $p > n$ has no mathematical meaning. Further assume that the graph G contains neither loops normultiple edges. Finally, we assume that for each edge $e = (v_r, v_s)$ the length of e is equal to the distance between v_r and v_s (i.e. $l(e) = d(v_r, v_s)$), because otherwise, the edge e could be eliminated without affecting the p-radius of G. The inverse of the p-center problem is defined as follows: Given a graph $G(V, E)$ and a positive integer r, find the smallest positive integer p such that the p-radius of G is not greater than r. This number p is called the (absolute) domination number of radius r of G while a corresponding p-center is called an (absolute) dominating set of radius r. The vertex domination number of radius r and the vertex dominating set of radius r are similarly defined.

The problem of finding a p-center of G was originated by Hakimi [5], [6] and is discussed in a number of papers [3], [4], [10], [8], [9], [12]. In [7], Hakimi, Schmeichel and Pierce discussed improvements and generalizations of various existing algorithms for finding p-centers of graphs and gave the corresponding orders of complexity. We assume that the distance-matrix, which gives the distance $d(v_i, v_k)$ between every pair of vertices v_i and v_k, is known.

In paper [11] Kariv and Hakimi described an $O(|E| . n \lg n)$ algorithm for finding an absolute 1-center in a vertex-weighted graph, and an $O(|E| . n + n^2 . \lg n)$ algorithm for finding an absolute 1-center ina vertex-unweighted graph. Kariv and Hakimi also, in [11] described an $O[|E|^p . n^{2p-1}/(p-1)! \lg n]$ algorithm (respectively an $O[|E|^p . n^{2p-1}/(p-1)!]$ algorithm) for finding an absolute p-center $(1 < p < n)$ in a vertex-weighted (respectively, vertex-unweighted) graph. In the case, when the graph G is a tree, Kariv and Hakimi in [11] describe the following algorithms:

(i) An $O(n . \lg n)$ algorithm for finding the (vertex or absolute) 1-center of a vertex-weighted tree.

(ii) An $O(n)$ algorithm for finding a (vertex or absolute) domination set of radius r. And

(iii) an $O(n^2 \lg n)$ algorithm for finding a (vertex or absoute) p-ceter for any $1 < p < n$ of a vertex-weighted tre.

Kariv and Hakimi showd in [11] that the problem of finding a (vertex or absolute) p-center (for $1 < p < n$) of a vertex-weighted graph, and the problem of finding a dominating set of radius r are NP-hard even in the case where the graph has a simple structure (e.g., a planar graph of maximum degree 3). Moreover, the problem to find a near optimal vertex (or absolute) p-center is also NP-hard [13].

2. The minimal vertex-weighted spanning tree of graph G

Algorithm 2.1 for the absolute center of graph G (The method of Hakimi [2])

1. For each edge a_k of the graph find that point (or points) Y_k^* on a_k which has the minimum radius.

2. Choose the smallest of all the $y_k^*(k = 1, 2, \ldots, m)$ as the absolute center of G.

Note 2.1

The first step in Algorithm 2.1 is done as follows:
Consider the edge a_k of the graph G as shown in *Fig. 1.*

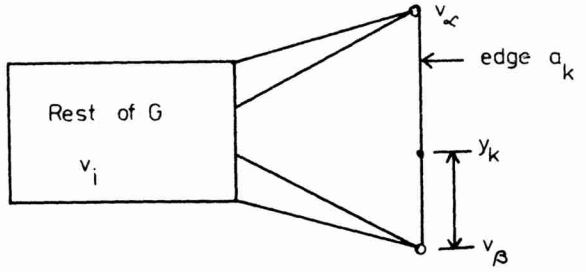

Fig. 1

From eqn. (1.2), we have:

$$F(y_k) = \max_{v_i \in V} [w(v_i) \,.\, d(y_k, v_i)] = \tag{2.1}$$

$$= \max_{v_i \in V} [w(v_i) \,.\, \min \{L(y_k, v_\beta) + d(v_\beta, v_i),$$

$$L(y_k, v_\alpha) + d(v_\alpha, v_i)\}] \,.$$

Now let $L(y_k, v_\beta) = \xi$. Since $L(y_k, v_\alpha) = c_{\alpha\beta} - L(y_k, v_\beta) = c_{\alpha\beta} - \xi$; from eqn. (2.1) we get

$$F(y_k) = \max_{v_i \in V} \min [w(v_i)\{\xi + d(v_\beta, v_i)\},$$

$$w(v_i)\{c_{\alpha\beta} + d(v_\alpha, v_i) - \xi\}] \,. \tag{2.2}$$

For a given v_i, we can find the smallest of the two terms in the square parentheses of eqn. (2.2) for every value of ξ, $(0 \leqslant \xi \leqslant c_{\alpha\beta})$, by plotting the two

terms

$$T_i = w(v_i)\{\xi + d(v_\beta, v_i)\}, \tag{2.3}$$

$$T_i' = w(v_i)\{c_{\alpha\beta} + d(v_\alpha, v_i) - \xi\}.$$

separately against ξ and taking the lower envelope of the two resulting strainght lines.

We repeat this process for all the other $v_i \in V$ and obtain all other lower envelopes on the same plot. We now draw the overall upper envelope to all the previously obtained lower envelopes and this final envelope according to eqn. (2.2) gives the radius $F(y_k)$ for all values of ξ. The position of the point y_k which produces the lowest minimum is, according to eqn. (1.3), the absolute center y_k^* subject to the initial restriction that y_k must lie on edge a_k.

Note 2.2 [2]

The method of Hakimi requires a search for a local center along every existing edge in the graph G. The modification to the method determines upper and lower bounds on the absolute local radii associated with the local centers on each edge, and these bounds may therefore enable one to reduce the number of edges that may have to be searched. Any local center located on edge (v_i, v_j) will (as can be seen from eqn. (2.1) with the L terms set to zero) have associated with it an absolute local radius (r_{ij}), whic must be at least as large as p_{ij}; where

$$p_{ij} = \max_{v_s \in V - \{v_i, v_j\}} [w(v_s) \min \{d(v_s, v_i), d(v_s, v_j)\}]. \tag{2.4}$$

This p_{ij} is a lower bound on the absolute radius of the graph, provided that the absolute center lies on edge (v_i, v_k). Hence the quantity

$$P = \min_{(v_i, v_j) \in E} [p_{ij}] \tag{2.5}$$

is a valid lower bound on the absolute radius.

Suppose the absolute center is in the center of edge (v_i, v_j); then, according to eqn. (2.1), the absolute radius is $p_{ij} + w(v_s^*) \cdot c_{ij}/2$ where $w(v_s^*)$ is the value of that v_s which produces the maximum p_{ij} according to eqn. (2.4). Hence the quantity

$$H = \lim_{(v_i, v_u) \in E} [p_{ij} + w(v_s^*) \cdot c_{ij}/2] \tag{2.6}$$

is a valid upper bound on the absolute radius. Thus any edge (v_{i_0}, v_{j_0}) for which $p_{i_0 j_0} \geqslant H$ may by exclude from the search for the absolute center. We denote by E' the set of all edges (v_i, v_k) of G for which $p_{ij} < H$.

Definition 2.1

Let G be a connected vertex-weighted graph. Then a vetex-weighted spanning tree T of the graph G is called minimal central spanning tree, when a 1-center of G and a 1-center of T coincide and their radii are equal.

The following algorithm finds a minimal central spanning tree T of a given connected vertex-weighted graph G. Thus such a tree always exists.

Algorithm 2.2

Step 1: If G has only one vertex, then it is sufficiento to put $T = G$. Assume that G has at least two vertices and at least one edge (G is connected). Then for each edge (v_i, v_k) of G we compute the lower bound p_{ij} on the absolute radius according to (2.4), and the upper bound H on the absolute radius according to (2.5).

Step 2; We denote the edges of G, which have $p_{ij} < H$, $e_1, e_2, \ldots, e_{|E'|}$.

Step 3: For every edge $e_k (1 \leqslant k \leqslant |E'|)$ we form a spanning tree of G as follows:

(a) Let $e_k = (v_\alpha, v_\beta)$, and let s_k be the middle point of the edge e_k. We put $T_{s_k} := \{s_k\}$, $A_{s_k} := \emptyset$; and assign to vertex v_α a pair $[a_\alpha, q_\alpha]$ and to v_β a pair $[a_\beta, q_\beta]$,
where $a_\alpha := s_k$, $q_\alpha := w(v_\alpha) \cdot c(v_\alpha, v_\beta)/2$,
$a_\beta := s_k$, $q_\beta := w(v_\beta) \cdot c(v_\alpha, v_\beta)/2$.
To the remaining vertices $[0, \infty]$ is asigned.

(b) Let v_j^* be a vertex of G, such that $q_j^* = \min\limits_{v_j \notin T_{s_k}} [q_j]$, and let $a_i^* \in T_{s_k}$ be such that $c(a_i^*, v_j^*) = \min\limits_{a_i \in T_{s_k}} \{c(a_i, v_j^*)\}$. We put $T_{s_k} := T_{s_k} \cup \{v_j^*\}$, $A_{s_k} := A_{s_k} \cup \cup \{(a_i^*, v_j^*)\}$.

(c) If $|T_{s_k}| = |V + 1|$, assign: $T_{s_k} := T_{s_k} - \{s_k\}$ $A_{s_k} := A_{s_k} \cup \{(v_\alpha, v_\beta)\} - \{(v_\alpha, s_k), (s_k, v_\beta)\}$, where $c(v_\alpha, v_\beta) = c(v_\alpha, s_k) + c(s_k, v_\beta)$ and halt [the resultant tree is minimal central spanning tree with minimal radius of G, starting from the middle point of the edge e_k].
If $|T_{s_k}\} < |V + 1|$, then for every vertex $v_i \notin T_{s_k}$ which is adjacent to v_j^*, if

$$f_i = w(v_i)[d(v_{s_k}, v_j^*) + c(v_j^*, v_i)] < q_i$$

then assign $q_i := f_i$, $a_i := v_j^*$ and go to (b).

Step 4: On every vertex-weighted spanning tree $T_{s_k} (1 \leqslant k \leqslant |E'|)$, we use Note 2.1 and find the local absolute 1-center c_k and radius r_k of the spanning tree, which must lay according to Note 2.1 on the edge e_k.

Step 5: We denote the spanning tree, with radius $r^* = \min\limits_{1 \leqslant k \leqslant |E'|} \{r_k\}$, by T^*. And let x^* be its 1-absolute center, then T^* will be desired spanning tree.

For each k ($1 \leqslant k \leqslant |E'|$), we will proceed as follows:

In the step 1 we compute the lower bound and the upper bound of the upper bound of the absolute radius for each edge e_k of the graph G

In step 3 we find the minimal central spanning tree T_{s_k} with minimal radius of G, starting from the middle point of the edge e_k.

In step 4 we find the local absolute 1-center c_k and radius r_k of the spanning tree T_{s_k}.

In step 5 we find the spanning tree with minimal radius and its 1-absolute center.

This absolute 1-center according to notes 2.1, 2.2 and the algorithm 2.1 is an absolute 1-center of the graph G. This proves that the algorithm works correctly as desired.

Provided that the distance matrix is given, it is easy to see that Step 1 of the algorithm needs $O(|E| + n)$ operations, Step 3 needs $O(|E'| \,.\, n^2)$ operations, Step 4 needs $O(|E'| \,.\, n \,.\, \lg n)$ operations and Step 5 needs $O(n)$ operations. Thus the total complexity of the algorithm is $O(|E'| \,.\, n^2)$, which can be better than $O(|E| \,.\, n \,.\, \lg n)$ [11] whenever $|E'|$ is relatively small.

3. A dominating set of radius r and a p-center in a vertex-weighted unicyclic graph

3.1. A dominating set of radius r in a vertex-weighted unicyclic graph

In this section we describe an $O(n)$ algorithm to find an (absolute or vertex) dominating set of radius r in a vertex-weighted unicyclic graph. Namely: Given a positive number r and a vertex-weighted unicyclic graph $G = G(V, E)$, find the smallest positive integer p such that the absolute or vertex p-radius of G is not greater than r.

The following algorithm finds both an absolute and vertex dominating set of radius r. The algorithm is carried out a search on the edges of G, starting from the leaves and moving toward the "middle". During this search, we locate the points of the desired dominating set of radius r in an "optimal fashion" until the whole graph is covered by some points within (weighted) radius r.

To do this, we use a copy G' of the original graph as an auxiliary graph on which the algorithm is carried out, and we attach a variable $R(v) = -r/w(v)$ to each vertex v of G'.

1. If v_r is a leaf of G' and $e = (v_r, v_s)$ is the edge incident to it, then we traverse (search) the edge e from v_r to v_s, or update the variable $R(v_s)$, and remove the

vertex v_r and then edge e from the graph G'. As a result, a new vertex may become a leaf of G', and the process is repated until all the leaves of G' are deleted.

2. G' does not contain any leaf

(a) G' is a cycle with H vertices ($H \leqslant n$). The algorithm consecutively searches all the vertices $v_{c_1}, v_{c_2}, \ldots, v_{c_H}$ of the cycle, deletes one and then the other edge incident to the vertex, and transforms the cycle into a path in this way. Then to this resulting path the same process as described in 1 is applied.

The graph G' will serve as an axiliary graph, therefore we have to store the original copy of this cycle which will be denoted C_{mem}, to be able to reconstruct the graph G' whenever needed. Further we will store the number of points of the dominating set already placed in the course of the algorithm in the variable denoted p_{mem}. The cycle C_{mem} consists of H vertices, each of them being incident to two edges. Hence, the process of deleting one edge from the cycle will be repeated $2H$-times. The variable I, $1 \leqslant I \leqslant 2H$, serves as indicator of the order of repetition of the deleting process. Without loss of generality we can assume that I is odd. Then both the processes (number I and number $I + 1$) start at the same vertex v_r.

If $R(v_r) \geqslant 0$, then v_r is already covered.

If $R(v_r) < 0$, we arrive into the situation denoted by the variable SPECIAL in the algorithm ; its value equals $'+'$ in such a case. We will store the vertex v_r in the variable denoted by $v_{r_{mem}}$; its adjacent vertice will be denoted v_{s_1}, v_{s_2} (in any order). Our aim is to find a point of the dominating set whic corves $v_{r_{mem}}$. By this procedure the orientation of searching the vertices of G' has to be saved as long as SPECIAL $= '+'$. For odd I this orientation is determined by the vertex $v_s = v_{s_1}$ — after the starting deletion of the edge $(v_r, v_s) = (v_{r_{mem}}, v_{s_1})$ we put $v_r = v_{s_1}$ and as the next edge (v_r, v_s) the edge incident with this new vertex v_r is taken, etc. For $I + 1$, i.e. for even I, for the starting vertex $v_s = v_{s_2}$ has to be taken. There are two possibilities how to find the point of the dominating set covering the vertex $v_{r_{mem}}$: either such a point already exists (i.e. has been constructed — see the algorithm 7c), or it still has to be placed. Any way, in both cases we have to consider the distance of this point of the doinating set from the starting vertex $v_{r_{mem}}$ which will be stored in the variable denoted by d. This d wil be then substituted for the value of $R(v_{r_{mem}})$ into parts 8, 9 of the algorithm. At this moment the situation denoted as 'APECIAL $= '+''$ is finished; the value of the variable has to be changed to SPECIAL $= '-'$.

We continue in the usual way as in 1 of this section. We have achieved a partial solution, in the first p points we have the number of the dominating set, which we compare with the value of the variable $p_{\min}$, storing this so far best solution. If $p < p_{mi}$, then assign $p_{\min} := p$ and in the graph $C_{\min}$, which is a copy of C_{mem}, we will store the points of the dominating set, which have been placed

in the I-th cycle just finished. Otherwise the value of $p_{\min}$ remains unchanged.

We increase $I: = I + 1$ and if $i \leqslant 2H$, we reconstruct the copy $G': = C_{mem}$, as well as the values of the variables $p: = p_{mem}$, $d: = 0$ and repeat the process. If $I > 2H$, then $p_{\min}$ is the optimal number of points of the dominating set of radius r covering the original graph G.

(b) The graph G' contains a single vertex, already. If $I = -1$ (initial value), then G does not contain any cycle, the algorithm stops; otherwise this fact indicates the end of the I-th searching of the cycle.

Algorithm 3.1

A dominating set of radius r in a vertex-weighted unicyclic graph

Step 1: Assign $G': = G, p: = = 0, I: = -1$, SPECIAL: $= '-', p_{\min}: = \infty$. For each vertex v of G' assign $R(v): = -r/w(v)$.

Step 2: If there exists no leaf v_r of the auxiliary graph G' such that $R(v_r) \geqslant 0$, then go to 5.

If the graph G' consists of a single vertex v_r with $R(v_r) \geqslant 0$ then:

If $I < 0$ then STOP (G is an acyclic graph).

If $p < p_{\min}$ then assign: $p_{\min}: = p$, $C_{\min}: = C_{mem}$,

In $C_{\min}$ store points of the dominating set having been arised in the I-th process.

If $I = 2H -$ STOP $- C_{\min}$ and $p_{\min}$ will be solutions. Else assign: $I: = I + 1$, $: = p_{mem}$, $G': = C_{mem}$, go to 5.

Else let v_r be a leaf of G' such that $R(v_r) \geqslant 0$ and let $e = (v_r, v_s)$ be the edge incident to v_r in G'. Remove e and v_r from the graph G'.

Step 3: If $R(v_r) + l(e) \leqslant r/w(v_s)$, then go to 4.

If $R(v_r) + l(e) > r/w(v_s)$, then return to 2.

Step 4: If $0 \leqslant R(v_s) \leqslant R(v_r) + l(e)$, the return to 2.

If $0 < R(v_r) + l(e) < R(v_s)$, then assign $R(v_s): = R(v_r) + l(e)$ and return to 2.

If $0 < R(v_r) + l(e) \leqslant -R(v_s)$, then asign $R(v_s): = R(v_r) + l(e)$ and return to 2.

If $0 < -R(v_s) < R(v_r) + l(e)$, then return to 2.

Step 5:

(a) If the auxiliary graph G' consists of a single vertex v_r, then assign $p: = p + 1$, locate a new point of the dominating set at the vertex v_r and

If $I < 0$ then STOP (G is an acyclic graph).

If $p < p_{\min}$, then assign $p_{\min}: = p$, $C_{\min}: = C_{mem}$, in $C_{\min}$ store the points of the dominating set having been arised in the I-th process.

If $I = 2H -$ STOP $- C_{\min}$ and $p_{\min}$ will be solutions. Else assign $I: = I + 1$, $p: = p_{mem}$, $G': = C_{mem}$; go to 5.

(b) If the vertex v_r is a leaf of G' and $e = (v_r, v_s)$ is the edge incident to v_r in G'. Remove e and v_r from the graph G' and go to 6.

(c) If $I = -1$, then assign $I: = 1$, $p_{mem}: = p$, and we wil create two copies C_{mem} and $C_{\min}$ of the graph G'. The cycle C_{mem} is formed by vertices v_{c_1}, v_{c_2}, ..., v_{c_H}, where $H \leqslant n$ is the number of vertices of the cycle C_{mem}^{-} and $\{c_1, \ldots c_H\} \leqslant \{1, \ldots, n\}$.

If I is odd, then denote $v_r = v_{c_{(I+1)/2}}$($v_{c_{(I+1)/2}}$ is a vertex of the cycle G'), further v_{s_1}, v_{s_2} are distinct vertices adjacent with v_r. We take v_{s_1} for v_s, $e = (v_r, v_s)$.

If I is even, then denote $v_r = v_{c_1}/2$, we take v_{s_2} for v_s, $e = (v_r, v_s)$.

If $R(v_r) \geqslant 0$, remove $e = (v_r, v_s)$; go to 3.

If $R(v_r) < 0$, assign $d: = 0$, SPECIAL: '+'; denote $v_{r_{mem}} = v_r$, remove the edge $e = (v_r, v_s)$, go to 6.

Step 6: If $-R(v_r) > l(e)$, then if SPECIAL = '+', then assign: $d: = d + l(e)$. Go to 7.

If $-R(v_r) = l(e)$, then SPECIAL = '+', then assign: $d: = d + l(e)$. Go to 8.

If $-R(v_r) < l(e)$, then if SSPECIAL = '+', then assign: $d: = d - -R(v_r)$. Go to 9.

Step 7:

(a) If $-R(v_r) - l(e) < R(v_s)$, then assign $R(v_s): = R(v_r) + l(e)$.
In the case SPECIAL = '+' the algorithm has to continue under assignement $v_r: = v_s$, and v_s asign to the adjacent vertex of the just picked vertex v_r. Go to 2.

(b) If $0 < -R(v_s) \leqslant -R(v_r) - l(e)$, then in the case SPECIAL = '+' the algorithm has to continue under assignement $v_r: = v_s$, and v_s assign to the adjacent vertex of the just picked vertex v_r. Go to 2.

(c) If $0 \leqslant R(v_s) \leqslant R(v_r) - l(e)$, then in the case SPECIAL = '+' assign: $R((v_{r_{mem}}): = d + R(v_s)$, SPECIAL: = '−', ($v_{r_{mem}}$ is covered by already constructed points of the dominating set). Go to 2.

(d) If $-R(v_r) - l(e) < R(v_s)$, then assign $R(v_s): = R(v_r) + l(e)$.
In the case SPECIAL = ' + ' the algorithm has to continue under assignement $v_r: = v_s$, and v_s assign to the adjacent vertex of the jsut picked vertex v_r. Go to 2.

Step 8: If $R(v_s) \neq 0$, then assiqn $p: = p + 1$, locate a new point of the dominating set at v_s and assign $R(v_s): = 0$.
If SPECIAL = '+', then assign $R(v_{r_{mem}}): = d$, SPECIAL: = '−'. Go to 2.

Step 9: [For absolute dominating set radius r].
Assign $p: = p + 1$, locate a new point of the dominating set on e at a distance $-R(v_r)$ from v_r;
If SPECIAL = '+', then assign $R(v_{r_{mem}}): = d$, SPECIAL: = '−'. Go to 3.

[For vertex dominating set of radus r].
Assign $p: = p + 1$, locate a new point of the dominating set at v_r.
If SPECIAL = '+', then asign $R(v_{r_{mem}}): = d + R(v_r)$.
Assign $R(v_r): = 0$ and go to 3.

The number p which is obtained when the algorithm terminates is the (absolute or vertex) dominating number of radius r, and the set of p points which was constructed during the algorithm is a dominating set of radius r. The validity of Algorithm 3.1 can be established by observing that, at each stage, the present number p is the minimum number of points which are needed to cover those vertices of the graph which are presently covered (that is, a point is added to the dominating set whenever it is necessary).

The deletion all vertices of the graph G' is carried out in the time $O(n)$. Therefore, the complexity algorithm 3.1 is $O(n)$.

3.2 A p-center ($p > 1$) of a vertex-weighted unicyclic graph

In this section, we use Algorithm 3.1, which was designed for finding the (absolute or vertex) dominating number of radius r and a corresponding dominating set, in order to find the (absolute or vertex) p-radius of a vertex-weighted unicyclic graph G and a corresponding p-center.

Kariv and Hakimi showed in [24] that here exist $o(n^3)$ ($O(n^2)$, respectively) possible values of the absolute (vertex, respectively) p-radius for the absolute (the vertex, respectively) p-center. Denote the set of all the possible valius of p-radius by Q. We can find Q in time $O(n^3)$ ($O(n^2)$, respectively) [11].

Let the desired (absolute or vertex) p-radius is dentoted by r_p. Given $r' \in Q$, the (absolute or vertex) domination number p' of radius r' can be found by Algorithm 3.1. If $p' \leqslant p$, then $r_p \leqslant r'$. Therefore, we obtain:

$$r_p = \min_{r' \in Q} \{r'/p' \leqslant p\}.$$

Thus, by using Algorithm 3.1, one can search the $O(n^3)$ or $O(n^2)$ possible values r' and find the p-radius of the given graph.

Algorithm 3.1, which gives the domination number p', also constructs a dominating set of radius p'. Thus, once the p-radius r_p is known, one can construct a dominating set of radius r_p. Let p_1 be the number of points in this set. If we add to the dominating set $p - p_1$ points (arbitrary $p - p_1$ points in the case of absolute p-center, or $p - p_1$ arbitrary vertices in the case of vertex p-center), then we obtain a desired (absolute or vertex) p-center. In this way the following algorithm for finding the (absolute or vertex) p-center is verified.

Algorithm 3.2

A p-center ($p > 1$) of a vertex-weighted unicyclic graph

Step 1: Calculate the $O(n^3)$ ($O(n^2)$, respectively) values r' for the absolute (vertex, respectively) p-center (by Kariv and Hakimi [11]].

Step 2: Arrange on $O(n^3)$ ($O(n^2)$, respectively) values r' in a list L according to a nondecreasing order.

Step 3: By performing a binary search on the list L of the values r', and by using Algorithm 3.1, find the smallest r' for which $p' \leqslant p$ (where p' is the domination number of radius r') Denote this value of r' by r_p, and dentote the domination number of radius r_p by p_1.

Step 4: Let $X^*_{p_1}$ be the dominating set of radius r_p as constructed by Algorithm 2.1. Add any arbitrary $p - p_1$ points to $X^*_{p_1}$ (in the case of the vertex p-center, these points must be vertices).

The resulting X^*_p is a p-center of the unicyclic graph.

Step 1 of Algorithm 3.2 is carried out in $O(n^3)$ ($O(n^2)$, respectively) operations Step 2 requires $O(n^3 . \lg n)$ ($O(n^2 . \lg n)$. respectively) operations. The complexity of the binary search which is performed in step 3 is $O(\lg n)$ oerations, while Algorithm 3.1, which is carried out in each search case costs $O(n)$ operations, — and thus the complexity of step 3 is $O(n . \lg n)$.
Therefore, the complexity of Algorithm 3.2 is $O(n^3 . \lg n)$ ($O(n^2 . \lg n)$, respectively).

REFERENCES

1. Behzad, M.—Chartrand, G.—Lesniak—Foster, L.: Graphs and digraphs, Prindle, Boston 1976.
2. Christofides, N.: Graph theory, An algorithmic approach, Academic press, London 1975.
3. Dearing, P. M.—Francis, R. L.: A minimax location problem on a network, Transp. Sci. 8 (1974), 333—343.
4. Goldman, A. J.: Minmax location of a facility on a network, Transp. Sci. 6 (1972), 407—418.
5. Hakimi, S. L.: Optimum locations of swiching centers and the absolute centers and medians of a graph, Operation Res. 12 (1964), 450—459.
6. Hakime, S. L.: Optimal distribution of switching centers in a communications network and some related graph theoretic problems, Operations Res. 13 (1965), 462—475.
7. Hakimi, S. L.—Schmeichel, E. F.—Pierce, J. G.: On p-centers in networks, Transp. Sci. 12 (1978), 1—15.
8. Halfin, S.: On finding the absolute and vertex centers of a tree with distances, Transp. Sci. 8 (1974), 75—77.
9. Handler, G. Y.: Minimax location of a facility in an indirected tree graph, Transp. Sci. 7 (1973), 287—293.
10. Handler, G. Y.: Minimax network location: theory and algorithms, Technical Rep. no. 107, Oper. Res. Center, Mass. Inst. of Tech., Combridge, Mass., Nov. 1974.

11. Kariv, O.—Hakimi, S. L.: An algorithmic approach to network location problems, I: the p-cneters, SIAM J. Appl. Math. 37 (1979)m 513—538.
12. Minieka, E.: the m-center problem, SIAM Review 12 (1970), 138—139.
13. Plesník, J.: On the computional complexity of centers locating in a graph, Aplikace Mat. 25 (1980), 445—452.

Author's adress:
Mohamad Hassan
Lattakia — Ain al Tine 62
Syria

Received: 12. 7. 1985

SÚHRN

PROBLÉM p-CENTRA GRAFU OBSAHUJÚCEHO JEDINÝ CYKLUS

Mohamad Hassan, Bratislava

Nech $G = G(V, E)$ je konečný súvislý graf s počtom vrcholov n.

V práci sa konštruuje minimálna vážená kostra T grafu G, tak, že 1-centrum T je zhodné s 1-centrom grafu G. Tiež sa popisuje algoritmus zložitosti $O(n)$ pre nájdenie dominujúcej množiny polomeru r a algoritmus zložitosti $O(n^3 . \lg n)$ (resp., $O(n^2 . \lg n)$ pre nájdenie absolútneho (resp., vrcholového) p-centra vo vrcholovo-ohodnotenom grafe, obsahujúcom jediný cyklus ($1 < p < n$).

РЕЗЮМЕ

ПРОБЛЕМА p-ЦЕНТРА ОДНОЦИКЛИЧЕСКОГО ГРАФА

Мохамед Гассан, Братислава

Пусть $G = G(V, E)$связный граф, с числом вершин n.

Работа посвящена конструкции минимального взвешенного каркаса T графа G, 1-центр которого совпадает с 1-центром графа G. Дается тоже алгоритм сложности $O(n)$ находящий домирующее множество радиуса r и также алгоритм сложности $O(n^3 . \lg n)$ $O(n^2 . \log n)$ находящий (абсолютный или вершинный) p-центр в вершинно-взвешенном одноцыкличес-ком графе ($1 < p < n$).