アルゴリズム 32－8
（1993. 3. 18）

2連結グラフに対する 根を中心とする生成木を求める線形時間アルゴリズム

和田 幸一　川口 喜三男

名古屋工業大学

グラフ $G=(V,E)$ の点 v の離心数 $e(v)$ は v から到達できる最大距離で定義される。最小の離心数を持つ点を G の中心と呼ぶ。

本稿では与えられた2連結グラフ $G=(V,E)$ と任意の点 $r\in V$ に対して、r が T の中心となる G の生成木 T を見つける線形時間アルゴリズムを示す。

A Linear-Time Algorithm for Centering a Spanning Tree of a Biconnected Graph

Koichi Wada and **Kimio Kawaguchi**
Nagoya Institute of Technology
Gokiso-cho, Syowa-ku, Nagoya 466, JAPAN

Given a biconnected graph $G=(V,E)$ and any vertex r in V, we show a linear-time algorithm to construct a spanning tree T of G with r in the center of T.

1 Introduction

In a graph $G = (V, E)$, the shortest path length between u and v is denoted by $dis_G(u, v)$. The *eccentricity* $e(v)$ of a vertex v is defined to be
$e(v) = max\{dis_G(v, u)|u \in V\}$.
The *center* of a graph G consists of the set of vertices having minimum eccentricity.

It was shown in [1] that for any vertex r in a biconnected graph G, there exists a spanning tree T of G such that r is in the center of T and an $O(|V| \cdot |E|)$ algorithm was given for constructing such a spanning tree. In this paper, we show a linear-time algorithm for it.

2 Preliminary

We refer readers to [3] for basic graph terminology. We deal with a simple undirected graph $G = (V, E)$. For a vertex v of G, $N_G(v) = \{u|(v, u) \in E\}$. A graph G is *biconnected* if there exist two vertex-disjoint paths between any two vertices in G.

A *tree* T is an undirected graph that is connected and acyclic. A *rooted tree* is a tree T with a distinguished vertex r, called the *root*. The distance $dis_T(r, v)$ from the root to v is called the *depth* of v and is denoted by $depth_T(v)$. The *depth of* T is defined to be the maximum of $depth_T(v)$.

The following proposition is an alternative characterization for a root being in the center of a tree, which is useful for the purpose of the algorithm shown in this paper.

Proposition 1 [1] *Let T be a rooted tree with root r. The root r is in the center of T iff there exist two vertices a and b such that*

1. *the path between r and a and the path between r and b are vertex-disjoint,*

2. $|depth_T(r, a) - depth_T(r, b)| \leq 1$ *and*

3. for every vertex u, $depth_T(r,u) \leq max(depth_T(r,a), depth_T(r,b))$.

3 s-t Numbering

An s-t numbering for a biconnected graph is developed in the linear time algorithm for testing planarity of a graph [2] and it is used to solve several graph problems in linear time such as bipartition of biconnected graphs [5], 2-path tree problem [4] and optimal routings for connected graphs [6]. In this paper, for a biconnected graph G and a vertex r of G we will construct a spanning tree T with root r such that r is in the center of T in linear time by using the s-t numbering.

Given an edge (s,t) of a biconnected graph $G = (V,E)$, a bijective function $g : V \rightarrow \{1, 2, \ldots, |V| = n\}$ is called an *s-t numbering* if the following conditions are satisfied:

- $g(s) = 1$, $g(t) = n$ and
- Every vertex $v \in V - \{s,t\}$ has two adjacent vertices u and w such that $g(u) < g(v) < g(w)$.

Proposition 2 [2] *Let $G = (V,E)$ be a biconnected graph. For any edge $(s,t) \in E$, an s-t numbering can be computed in $O(|E|)$ time.*

4 A Linear-Time Algorithm

We assume that $G = (V,E)$ is a biconnected graph whose vertices are r-r' numbered for an edge $(r,r') \in E$. For a vertex v in $V - \{r\}$ and an r-r' numbering g, we define two vertices $p(v)$ and $s(v)$ as follows:

$g(p(v)) = min\{g(u) | u \in N_G(v)\}$ and

$g(s(v)) = max\{g(u)|u \in N_G(v)\}(if\ v \neq r'), s(r') = r.$

For a vertex v in V, Left subgraph $L(v)$ and right subgraph $R(v)$ are defined to be

$L(v) = (V_L = \{u|g(u) \leq g(v)\}, E_L = \{(u, p(u))|u \in V_L\})$ and

$R(v) = (V_R = \{u|g(v) < g(u)\} \cup \{r\}, E_R = \{(u, s(u))|u \in V_R - \{r\}\}).$

Clearly both $L(v)$ and $R(v)$ are trees with root r. Let $C(v) = (V_L \cup V_R, E_L \cup E_R)$. Let $d(L(v))$ and $d(R(v))$ be the depth of trees $L(v)$ and $R(v)$. Figure 1 shows examples of G and left and right trees.

The depth of $L(v)$ is monotone non-decreasing function of $g(v)$ from $d(L(r)) = 0$ to $d(L(r'))$ and the depth of $R(v)$ is monotone non-increasing function of $g(v)$ from $d(R(r))$ to $d(R(r')) = 0$. And it holds that $d(R(r)) - d(L(r)) = d(R(r)) > 0$ and $d(R(r')) - d(L(r') = -d(L(r')) < 0$. Moreover, for u and v such that $u, v \in V$ and $g(u) = g(v) + 1$, it holds that $d(L(v)) \leq d(L(u)) \leq d(L(v)) + 1$ and $d(R(v)) \geq d(R(u)) \geq d(R(v)) - 1$. Therefore, it holds that $0 \geq (d(R(u)) - d(L(u))) - (d(R(v)) - d(L(v))) \geq -2$ for v and u such that $g(u) = g(v) + 1$. This property implies that there exists a vertex x such that $d(R(x)) - d(L(x)) \leq 1$ and we can show the following main lemma.

Lemma 1 *For a biconnected graph $G = (V, E)$ and an r-r' numbering of G ($(r, r') \in E$), there exists a vertex $x \in V$ such that r is in the center of the tree $C(x)$.*

Proof By using the properties about $d(L(v))$ and $d(R(v))$ explained above, we can find a vertex x such that $d(R(x)) + 1 \geq d(L(x)) \geq d(R(x))$. If let a be a vertex whose depth is $d(L(x))$ in $L(x)$ and let b be a vertex whose depth is $d(R(x))$ in $R(x)$, then a and b satisfy the three conditions of Proposition 1. □

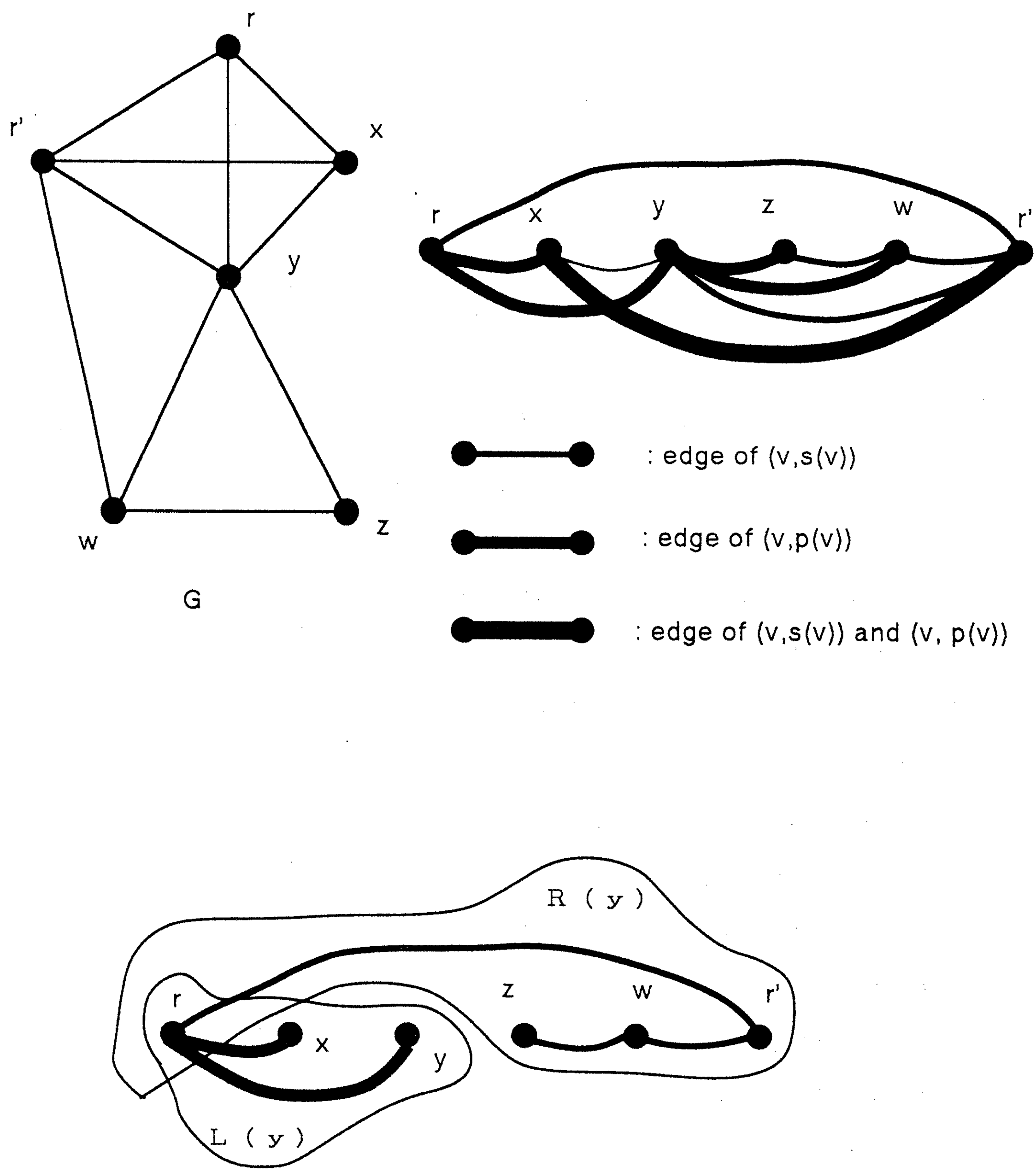

Figure 1: An example of G, $L(v)$ and $R(v)$.

We can obtain the following algorithm **CST** from Lemma 1.

Algorithm CST:Centering a Spanning Tree

{ **Input:** $G = (V, E)$ and $r \in V$}

{ **Output:** a spanning tree T of G with r in the center of T}

begin

1. Compute an r-r' numbering g for an edge $(r, r') \in E$;

{$g^{-1}(i)$ represents the vertex v such that $g(v) = i$.}

2. Select edges $p(v)$ and $s(v)$ for $v \in V - \{r\}$;

3.1 Compute the depth of each $v \in V$ in the tree $L(r')$ and

store it in $dL[g(v)]$;

3.2 Compute the depth of $v \in V$ in the tree $R(r)$ and

store it in $dR[g(v)]$;

4.1 Compute the depth of $L(v)$ for $v \in V$ and

store it in $maxdL[g(v)]$;

4.2 Compute the depth of $R(v)$ for $v \in V$ and

store it in $maxdR[g(v)]$;

5.1 $dLx \leftarrow maxdL[1]$; $dRx \leftarrow maxdR[1]$;

5.2 $i \leftarrow 1$;

{$dLx \leq dRx$}

5.3 while $dLx \leq dRx + 2$ **do**

5.3.1 $i \leftarrow i + 1$;

5.3.2 if $maxdL[i+1] > maxdL[i]$ **then** $dLx \leftarrow dLx + 1$;

5.3.3 if $maxdR[i+1] < maxdR[i]$ **then** $dRx \leftarrow dRx - 1$;

6. $T \leftarrow C(g^{-1}(i))$

end

i	1	2	3	4	5	6
g-1(i)	r	x	y	z	w	r'
dL	0	1	1	2	2	2
dR	0	2	2	3	2	1
maxdL	0	1	1	2	2	2
maxdR	3	3	3	2	1	0

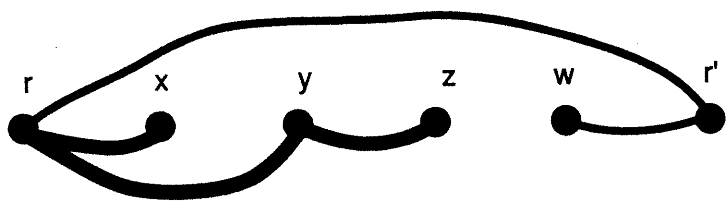

Figure 2: An example of execution of **CST** and a solution

Figure 2 shows an example of execution of **CST** for the graph G and the r-r' numbering g shown in Figure 1 and a solution.

Since every tree $L(v)(1 \leq g(v) \leq |V|)$ is a subtree of $L(r')$ and they have the root r in common, $dL[g(u)]$ gives the depth of u in $L(v)$ for every vertex u in $L(v)$. Similarly, $dR[g(u)]$ gives the depth of u in $R(v)$ for every vertex u in $R(v)$. The while loop of the line 5.3 always exits after at most n iterations from the behavior of $d(R(v)) - d(L(v))$ mentioned above.

Thus, the algorithm **CST** computes a spanning tree T with r in the center of T correctly from Lemma 1.

Theorem 1 *For a biconnected graph $G = (V, E)$ and a vertex $r \in V$, we can construct a spanning tree T with root r such that r is in the center of T in $O(|E|)$ time.*

Proof It is sufficient to show that the time complexity of the algorithm **CST** is $O(|E|)$. From Proposition 2, it takes $O(|E|)$ time in the line 1. For each v, $s(v)$ and $p(v)$ can be selected in $O($ degree of $v)$ time. Thus, it takes $O(|E|)$ time in the line 2. The line 3.1 is computed in $O(|V|)$ time, because the depth of all vertices in the tree $L(r')$ is computed by traversing each vertex with preorder. Similarly, it takes $O(|V|)$ in the line 3.2. Since the depth of $L(v)$(or $R(v)$) can be computed as $max_{1\leq i\leq g(v)} dL[i]$(or $max_{g(v)\leq i\leq |V|} dR[i]$), it takes $O(|V|)$ time in the lines 4.1-4.2. Clearly, it takes $O(1)$ time in the lines 5.1-5.2. Since the while loop of the line 5.3 iterates at most $O(|V|)$ times and it takes $O(1)$ time in the lines 5.3.1-5.3.3, it takes $O(|V|)$ time. Therefore we have proved the theorem. □

Acknowledgement: This research is partly supported by the Grant-in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan under Grant:(A)04750320.

References

[1] G.Cheston, A.Farley, S.Hedetniemi and A. Proskurowski, Centering a spanning tree of a biconnected graph, *Information Processing Letters*,32,5(1989)247-250.

[2] S. Evens, Graph algorithms, *Computer Science Press*, Potomac, Maryland(1979).

[3] F.Harary, Graph theory, *Addison-Wesley*, Reading, MA(1969).

[4] A. Itai and M. Rodeh, The multi-tree approach to reliability in distributed networks, *Information and Computation*,79,(1988),43–59.

[5] H.Suzuki, N.Takahashi and T.Nishizeki, A linear algorithm for bipartition of biconnected graphs, *Information Processing Letters*,33,5(1990)227-232.

[6] K.Wada, Y.Luo and K.Kawaguchi, Optimal fault-tolerant routings for connected graphs, *Information Processing Letters*,41,3(1992),169-174.