Discrete Applied Mathematics 103 (2000) 281–287

DISCRETE
APPLIED
MATHEMATICS

Note

Linear k-arboricities on trees

Gerard J. Chang[a, 1], Bor-Liang Chen[b], Hung-Lin Fu[a], Kuo-Ching Huang[c, *,2]

[a]*Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan*
[b]*Department of Business Administration, National Taichung Institue of Commerce, Taichung 404, Taiwan*
[c]*Department of Applied Mathematics, Providence University, Shalu 433, Taichung, Taiwan*

Received 31 October 1997; revised 15 November 1999; accepted 22 November 1999

Abstract

For a fixed positive integer k, the linear k-arboricity $\mathrm{la}_k(G)$ of a graph G is the minimum number ℓ such that the edge set $E(G)$ can be partitioned into ℓ disjoint sets and that each induces a subgraph whose components are paths of lengths at most k. This paper studies linear k-arboricity from an algorithmic point of view. In particular, we present a linear-time algorithm to determine whether a tree T has $\mathrm{la}_k(T) \leqslant m$.

Keywords: Linear forest; Linear k-forest; Linear arboricity; Linear k-arboricity; Tree; Leaf; Penultimate vertex; Algorithm; NP-complete

1. Introduction

All graphs in this paper are simple, i.e., finite, undirected, loopless, and without multiple edges. A *linear k-forest* is a graph whose components are paths of length at most k. A *linear k-forest partition* of G is a partition of the edge set $E(G)$ into linear k-forests. The *linear k-arboricity* of G, denoted by $\mathrm{la}_k(G)$, is the minimum size of a linear k-forest partition of G.

* Corresponding author.
E-mail addresses: gjchang@math.nctu.edu.tw (G.J. Chang), kchuang@simon.pu.edu.tw (K.-C. Huang).
[1] Supported in part by the National Science Council under grant NSC86-2115-M009-002.
[2] Supported in part by the National Science Council under grant NSC87-2115-M126-003.

PII: S0166-218X(99)00247-4

The notion of linear k-arboricity was introduced by Habib and Peroche [19]. It is a natural refinement of the linear arboricity introduced by Harary [21], which is the same as linear k-arboricity except that the paths have no length constraints. Suppose $\chi'(G)$ is the chromatic index of G and $\mathrm{la}(G)$ the linear arboricity. Let $\Delta(G)$ denote the maximum degree of a vertex in G. The following proposition is easy to verify.

Proposition 1. *If H is a subgraph of graph G with n vertices and m edges, then*
(1) $\mathrm{la}_k(G) \geqslant \mathrm{la}_k(H)$ *for* $k \geqslant 1$,
(2) $\mathrm{la}(G) = \mathrm{la}_{n-1}(G) \leqslant \mathrm{la}_{n-2}(G) \leqslant \cdots \leqslant \mathrm{la}_2(G) \leqslant \mathrm{la}_1(G) = \chi'(G)$,
(3) $\mathrm{la}_k(G) \geqslant \max\{\lceil \Delta(G)/2 \rceil, \lceil m/\lfloor kn/(k+1) \rfloor \rceil\}$.

On the other hand, Habib and Peroche [19] made the following conjecture:

Conjecture 2 (Habib and Peroche [19]). *If G is a graph with n vertices and $k \geqslant 2$, then*

$\mathrm{la}_k(G) \leqslant \lceil \Delta(G)n + \alpha/2\lfloor kn/(k+1) \rfloor \rceil$ *where* $\alpha = 1$ *when* $\Delta(G) < n-1$ *and* $\alpha = 0$ *when* $\Delta(G) = n-1$.

This conjecture subsumes Akiyama's conjecture [2] as follows.

Conjecture 3 (Akiyama [2]). $\mathrm{la}(G) \leqslant \lceil (\Delta(G)+1)/2 \rceil$.

Considerable work has been done for determining exact values and bounds for linear k-arboricity, aimed at these conjectures (see the references at the end of this paper).

We study linear k-arboricity from an algorithmic point of view in this paper. Habib and Peroche [20] showed the first result along this line. They gave an algorithm to prove that if T is a tree with exactly one vertex of maximum degree $2m$, then $\mathrm{la}_2(T) \leqslant m$. Using this as the induction basis, they then gave a characterization for a tree T with maximum degree $2m$ to have $\mathrm{la}_2(T) = m$. Chang [10] recently pointed out that this characterization has a flaw. He then presented a linear-time algorithm for determining whether a tree T satisfies $\mathrm{la}_2(T) \leqslant m$; and gave a new characterization for a tree T with maximum degree $2m$ to have $\mathrm{la}_2(T) = m$. Holyer [22] proved that determining $\mathrm{la}_1(G)$ is NP-complete, Peroche [26] that determining $\mathrm{la}(G)$ is NP-complete, and Bermond et al. [9] that determining whether $\mathrm{la}_3(G) = 2$ is NP-complete for cubic graphs of $4m$ vertices. Bermond et al. [9] conjectured that it is NP-complete to determine $\mathrm{la}_k(G)$ for any fixed k.

The purpose of this paper is to give a linear-time algorithm for answering whether a tree T satisfies $\mathrm{la}_k(T) \leqslant m$ for a fixed k. This answers a question raised in [10].

2. Linear k-arboricities on trees

We recall the following result in [10].

Theorem 4 (Chang [10]). *If T is a tree with $\Delta(T) = 2m - 1$, then $\mathrm{la}_k(T) = m$ for $k \geqslant 2$. If T is a tree with $\Delta(T) = 2m$, then $m \leqslant \mathrm{la}_k(T) \leqslant m + 1$ for $k \geqslant 2$.*

So, it remains to determine whether $\mathrm{la}_k(T)$ is m or $m + 1$ when $\Delta(T) = 2m$. The aim of this paper is to give a linear-time algorithm for determining if $\mathrm{la}_k(T) \leqslant m$ for a tree T.

A *leaf* is a vertex of degree one. A *penultimate vertex* is a vertex that is not a leaf and all of whose neighbors are leaves, with the possible exception of one. Note that a penultimate vertex of a connected graph is always adjacent to a non-leaf, unless the graph is a star. It is well known that a non-trivial tree has at least two leaves, and a tree with at least three vertices has at least one penultimate vertex.

To study linear k-arboricity on trees, we actually make the problem in a more general setting as follows. Suppose G is a graph in which every edge e is associated with a positive integer $L(e) \leqslant k$. The *L-length* of a path P is $L(P) = \sum_{e \in E(P)} L(e)$. A *linear* (k,L)*-forest* is a graph whose components are paths and $L(P) \leqslant k$ for each path P. The *linear* (k,L)*-arboricity* of G, denoted by $\mathrm{la}_{k,L}(G)$, is the minimum number of linear (k,L)-forests needed to partition the edge set $E(G)$ of G. It is clear that $\mathrm{la}_{k,L}(G) = \mathrm{la}_k(G)$ when $L(e) = 1$ for all edges e in G.

Suppose $s = (a_1, a_2, \ldots, a_r)$ is a sequence of positive integers. An (m,k)*-partition* of s is a "partition" of $\{1, 2, \ldots, r\}$ into m disjoint (but possibly empty) sets $I_1, I_2, \ldots, I_m$, each of size at most two, with the property that $\sum_{j \in I_i} a_j \leqslant k$ for $1 \leqslant i \leqslant m$. The *value* of an (m,k)-partition $\{I_1, I_2, \ldots, I_m\}$ of s is $\min\{\sum_{j \in I_i} a_j : |I_i| \leqslant 1\}$. $f_{m,k}(s)$ is defined to be the minimum value of an (m,k)-partition of s; $f_{m,k}(s) = \infty$ if s has no (m,k)-partition. Note that for convenience, $\min \emptyset = \infty$, $\sum_{j \in I_i} a_j = 0$ when I_i is an empty set, and $f_{m,k}(s) = 0$ when $r = 0 < m$.

The following is the foundation of our algorithm for the linear (k,L)-arboricity on trees.

Theorem 5. *Suppose T is a tree in which x is a penultimate vertex adjacent to a vertex y and $r \geqslant 1$ leaves $x_1, x_2, \ldots, x_r$. Suppose $T' = T - \{x_1, x_2, \ldots, x_r\}$, and L' is defined by $L'(e) = L(e)$ for all edges $e \in E(T')$ except $L'(yx) = L(yx) + f_{m,k}(L(xx_1), L(xx_2), \ldots, L(xx_r))$. Then, $\mathrm{la}_{k,L}(T) \leqslant m$ if and only if $\mathrm{la}_{k,L'}(T') \leqslant m$.*

Proof. $(\Rightarrow)$ Suppose $\mathrm{la}_{k,L}(T) \leqslant m$. Choose a linear (k,L)-forest partition $\mathscr{P} = \{F_1, F_2, \ldots, F_m\}$ for T. Without loss of generality, we may assume that yx is in a path P_1 that is a component of F_1. Let $I_i = \{j : xx_j \text{ is in } F_i \text{ and } 1 \leqslant j \leqslant r\}$ for $1 \leqslant i \leqslant m$. Then, $|I_i| \leqslant 2$ and $\sum_{j \in I_i} L(xx_j) \leqslant k$ for $1 \leqslant i \leqslant m$. Also, $|I_1| \leqslant 1$ as yx is in F_1. Therefore, $f_{m,k}(L(xx_1), L(xx_2), \ldots, L(xx_r)) \leqslant \sum_{j \in I_1} L(xx_j) = \sum_{xx_j \in P_1} L(xx_j)$.

Delete all edges $xx_1, xx_2, \ldots, xx_r$ from the linear (k,L)-forest partition $\mathscr{P}$ to yield a linear forest partition $\mathscr{P}'$ for T'. For any path P' that is a component of a forest F' in $\mathscr{P}'$, P' is a subpath of some path P that is a component of a forest F in $\mathscr{P}$. Then, $L'(P') = L(P') \leqslant L(P) \leqslant k$, except when P' contains the edge yx. For the exceptional case, $P' \subseteq P_1$ and

$$\begin{aligned} L'(P') &= L'(P' - yx) + L'(yx) \\ &= L(P' - yx) + L(yx) + f_{m,k}(L(xx_1), L(xx_2), \ldots, L(xx_r)) \\ &\leqslant L(P' - yx) + L(yx) + \sum_{xx_j \in P_1} L(xx_j) \leqslant L(P_1) \leqslant k. \end{aligned}$$

Therefore, $\mathscr{P}'$ is a linear (k, L')-forest partition for T' and then $\mathrm{la}_{k,L'}(T') \leqslant m$.

($\Leftarrow$) On the other hand, suppose $\mathrm{la}_{k,L'}(T') \leqslant m$. Choose a linear (k, L')-forest partition $\mathscr{P}' = \{F_1', F_2', \ldots, F_m'\}$ for T' such that yx is in a component P_1' of F_1'. Let $\{1, 2, \ldots, r\}$ be the disjoint union of sets $I_1, I_2, \ldots, I_m$, each of size at most two and $|I_1| \leqslant 1$, such that $\sum_{j \in I_i} L(xx_j) \leqslant k$ for $1 \leqslant i \leqslant m$ and $\sum_{j \in I_1} L(xx_j) = f_{m,k}(L(xx_1), L(xx_2), \ldots, L(xx_r))$. For $1 \leqslant i \leqslant m$, let $F_i = F_i' + P_i$, where P_i is the (possibly empty) path forming by the edge(s) xx_j with $j \in I_i$. Then, each component of an F_i is a path P. In fact, each path P is a component of some $F_{i'}'$ with $L(P) = L'(P) \leqslant k$, except when P is $P_1' + P_1$ or P_i with $2 \leqslant i \leqslant m$. Note that

$$\begin{aligned} L(P_1' + P_1) &= L(P_1' - yx) + L(yx) + \sum_{j \in I_1} L(xx_j) \\ &= L(P_1' - yx) + L(yx) + f_{m,k}(L(xx_1), L(xx_2), \ldots, L(xx_r)) \\ &= L'(P_1' - yx) + L'(yx) = L'(P_1') \leqslant k. \end{aligned}$$

Also, $L(P_i) = \sum_{j \in I_i} L(xx_j) \leqslant k$ for $2 \leqslant i \leqslant m$. Thus, $\{F_1, F_2, \ldots, F_m\}$ is a linear (k, L)-forest partition of T, which implies that $\mathrm{la}_{k,L}(T) \leqslant m$. $\square$

Based on Theorem 5, we have the following algorithm.

Algorithm L. Test whether $\mathrm{la}_{k,L}(T) \leqslant m$ for a tree T.

Input. Positive integers k and m and a tree T in which every edge e is associated with a positive integer $L(e) \leqslant k$.

Output. "Yes" if $\mathrm{la}_{k,L}(T) \leqslant m$ and "no" otherwise.

Method.

while (T is not an edge) **do**
choose a penultimate vertex x adjacent to a vertex y
(which may be a leaf) and $r \geqslant 1$ leaves $x_1, x_2, \ldots, x_r$;
$L(yx) \leftarrow L(yx) + f_{m,k}(L(xx_1), L(xx_2), \ldots, L(xx_r))$;
if $L(yx) > k$ **then output** "no" and **stop**;
$T \leftarrow T - \{x_1, x_2, \ldots, x_r\}$;
end while;
output "yes".

To implement the algorithm, we need to find a penultimate vertex and to compute $f_{m,k}(L(xx_1), L(xx_2), \ldots, L(xx_r))$ efficiently.

For finding a penultimate vertex, we choose a vertex v^* and order the vertices of T into $v_1, v_2, \ldots, v_n$ such that

$$d_T(v_1, v^*) \geqslant d_T(v_2, v^*) \geqslant \cdots \geqslant d_T(v_n, v^*),$$

where $d_T(v_i, v^*)$ is the distance from v_i to v^* in T. It is then clear that the first vertex v_i that is not a leaf is a penultimate vertex. This gives an easy way to choose a penultimate vertex. The other operations in the algorithm are easily implemented.

To compute $f_{m,k}(L(xx_1), L(xx_2), \ldots, L(xx_r))$ efficiently, we use the following lemma.

Lemma 6. *Suppose $s = (a_1, a_2, \ldots, a_r)$ is a non-decreasing sequence of positive integers less than or equal to k. Let r' be the maximum index less than r such that $a_{r'} + a_r \leqslant k$; and s' be obtained from s by deleting r* and *r' (if it exists).*

(1) *If $r \geqslant 2m+1$, then s has no (m,k)-partition. If $r \geqslant 2m$, then $f_{m,k}(s) = \infty$.*

(2) *s has an (m,k)-partition if and only if s' has an $(m-1,k)$-partition. In this case,*

$$f_{m,k}(s) = \begin{cases} f_{m-1,k}(s') & \text{if } r' \text{ exists}, \\ \min\{a_r, f_{m-1,k}(s')\} & \text{if } r' \text{does not exist}. \end{cases}$$

Proof. (1) follows from definition easily.

(2) First consider the case in which r' exists. Suppose $\mathscr{I} = \{I_1, I_2, \ldots, I_m\}$ is an (m,k)-partition of s. Let $r \in I_i$ and $r' \in I_j$. We may assume $j = i$, for otherwise repartitioning $I_i \cup I_j$ into $I_i' = \{r, r'\}$ and $I_j' = (I_i \cup I_j) - I_i'$ results in a new (m,k)-partition of s whose value is no more than the value of $\mathscr{I}$. In this case, $\{I_1, \ldots, I_{i-1}, I_{i+1}, \ldots, I_m\}$ is an $(m-1,k)$-partition of s' with the same value as $\mathscr{I}$. This also gives $f_{m,k}(s) \geqslant f_{m-1,k}(s')$. Conversely, suppose $\mathscr{I}'$ is an $(m-1,k)$-partition of s'. Then $\mathscr{I}' \cup \{\{r, r'\}\}$ is an (m,k)-partition of s with the same value as $\mathscr{I}$. This also gives $f_{m,k}(s) \leqslant f_{m-1,k}(s')$.

Next, consider the case in which r' does not exist. Suppose $\mathscr{I} = \{I_1, I_2, \ldots, I_m\}$ is an (m,k)-partition of s. Then $I_i = \{r\}$ for some i. In this case, $\mathscr{I}' = \{I_1, \ldots, I_{i-1}, I_{i+1}, \ldots, I_m\}$ is an $(m-1,k)$-partition of s'; and the value of $\mathscr{I}$ is the minimum of a_r and the value of $\mathscr{I}'$. So, $f_{m,k}(s) \geqslant \min\{a_r, f_{m-1,k}(s')\}$. Conversely, suppose $\mathscr{I}'$ is an $(m-1,k)$-partition of s'. Then $\mathscr{I}' \cup \{\{r\}\}$ is an (m,k)-partition of s with the value equals to the minimum of a_r and the value of $\mathscr{I}'$. So, $f_{m,k}(s) \leqslant \min\{a_r, f_{m-1,k}(s')\}$. □

According to the above lemma, we have the following linear-time algorithm for computing $f_{m,k}(a_1, a_2, \ldots, a_r)$.

assume $a_1 \leqslant a_2 \leqslant \cdots \leqslant a_r \leqslant k$ by a bucket sort if necessary;
let $a_0 \leftarrow 0$ and store the sequence $s \leftarrow (a_0, a_1, a_2, \ldots, a_r)$ in
  a doubly linked list in which the next element of a_i
  is next[a_i] and the previous element of a_i is prev[a_i];
answer $\leftarrow \infty$; $a_{r'} \leftarrow a_0$;
while ($r \leqslant 2m-1$ **or** ($r = 2m$ **and** answer $\neq \infty$)) **do**
  if ($r = 0$) **then** {**if** $m \neq 0$ **then** answer $\leftarrow 0$; **stop**} ;
  while (next[$a_{r'}$] $\neq a_r$ **and** next[$a_{r'}$] $+ a_r \leqslant k$) **do** $a_{r'} \leftarrow$ next[$a_{r'}$];

if $(a_{r'} = a_0)$ **then**$\{$answer $\leftarrow$ min$\{$answer$, a_r\}$;
$a_r^{\text{old}} \leftarrow a_r$; $a_r \leftarrow$ prev$[a_r]$; delete a_r^{old} from s;
$r \leftarrow r - 1$; $m \leftarrow m - 1$; $\}$
else $\{a_r^{\text{old}} \leftarrow a_r$; $a_{r'}^{\text{old}} \leftarrow a_{r'}$;
if (next$[a_{r'}] = a_r$)
then $\{a_r \leftarrow$ prev$[a_{r'}]$; $a_{r'} \leftarrow$ prev$[a_r]\}$
else $\{a_r \leftarrow$ prev$[a_r]$; $a_{r'} \leftarrow$ prev$[a_{r'}]\}$
delete a_r^{old} and $a_{r'}^{\text{old}}$ from s;
$r \leftarrow r - 2$; $m \leftarrow m - 1$; $\}$
end while;

Note that the bucket sort costs $O(r)$ time. During the above procedure, $a_{r'}$ traverses from the beginning to the end of the linked list, with the modification that after each iteration, $a_{r'}$ may be back one or two steps. So, the total cost for computing $f_{m,k}(a_1, a_2, \ldots, a_r)$ is $O(r)$.

Theorem 7. *Algorithm **L** determines if* $\text{la}_k(T) \leqslant m$ *for a tree* T *in linear time.*

3. For further reading

The following references are also of interest to the reader: [1,3–8,11–18,23–25,27–29].

Acknowledgements

The authors thank the referee for many constructive suggestions. In particular, the suggestions on Algorithm L make it clear that the algorithm is linear.

References

[1] H. Ait-Djafter, Linear arboricity for graphs with maximum degree six or seven and edge multiplicity two, Ars. Combin. 20 (1985) 5–16.
[2] J. Akiyama, Three developing topics in graph theory, Doctoral Dissertation, University of Tokyo, 1980.
[3] J. Akiyama, A status on the linear arboricity, Lecture Notes in Computer Science, Vol. 108, Springer, Berlin, 1981, 38–44.
[4] J. Akiyama, V. Chvátal, A short proof of the linear arboricity for cubic graphs, Bull. Liber. Arts and Sci., NMS 2 (1981) 1–3.
[5] J. Akiyama, G. Exoo, F. Harary, Covering and packing in graphs III, cyclic and acyclic invariants, Math. Slovaca 30 (1980) 405–417.
[6] J. Akiyama, G. Exoo, F. Harary, Covering and packing in graphs IV, linear arboricity, Networks 11 (1981) 69–72.
[7] J. Akiyama, M. Kano, Path factors of a graph, Graph Theory and its Applications, Wiley, New York, 1984.
[8] N. Alon, The linear arboricity of graphs, Israel J. Math. 62 (1988) 311–325.

[9] J.C. Bermond, J.L. Fouquet, M. Habib, B. Peroche, On linear k-arboricity, Discrete Math. 52 (1984) 123–132.
[10] G.J. Chang, Algorithmic aspects of linear k-arboricity, Taiwanese J. Math. 3 (1999) 73–81.
[11] C.Y. Chen, Y.P. Chen, G.J. Chang, The tree arboricity of a graph, submitted for publication.
[12] B.L. Chen, H.L. Fu, K.C. Huang, Decomposing graphs into forests of paths with size less than three, Austr. J. Combin. 3 (1991) 55–73.
[13] B.L. Chen, K.C. Huang, On the linear k-arboricity of K_n and $K_{n,n}$, Manuscript, 1996.
[14] H. Enomoto, The linear arboricity of 5-regular graphs, Technical Report, Dept. of Information Sci., Univ. of Tokyo, 1981.
[15] H. Enomoto, B. Peroche, The linear arboricity of some regular graphs, J. Graph Theory 8 (1984) 309–324.
[16] H.L. Fu, K.C. Huang, The linear 2-arboricity of complete bipartite graphs, Ars. Combinatoria 38 (1994) 309–318.
[17] F. Guldan, The linear arboricity of 10-regular graphs, Math. Slovaca 36 (1986) 225–228.
[18] F. Guldan, Some results on linear arboricity, J. Graph Theory 10 (1986) 505–509.
[19] M. Habib, P. Peroche, Some problems about linear arboricity, Discrete Math. 41 (1982) 219–220.
[20] M. Habib, P. Peroche, La k-arboricité linéaire des arbres, Ann. Discrete Math. 17 (1983) 307–317.
[21] F. Harary, Covering and packing in graphs I, Ann. New York Acad. Sci. 175 (1970) 198–205.
[22] I. Holyer, The NP-completeness of edge colourings, SIAM J. Comput. 10 (1981) 718–720.
[23] K.C. Huang, Some results on linear k-arboricity, Manuscript, 1996.
[24] F. Jaeger, Etude de quelques invariants et problèmes d'existence en théorie des graphes, Thèse d'Etat, IMAG Grenbole, 1976.
[25] A. Nakayama, B. Peroche, Linear arboricity of digraphs, Networks 17 (1987) 39–53.
[26] B. Peroche, Complexité de l'arboricité lináire d'un graphe, RAIRO 16 (1982).
[27] P. Tomasta, Note on linear arboricity, Math. Slovaca 32 (1982) 239–242.
[28] H.G. Yeh, G.J. Chang, The path-partition problem in bipartite distance-hereditary graphs, Taiwanese J. Math. 2 (1998) 353–360.
[29] T.W. Yeh, Linear arboricities of complete r-partite graphs, Master Thesis, Dept. Applied Math., National Chiao Tung Univ, Hsinchu, Taiwan, June 1997.