7-6-2012

Bandwidth of trees of diameter at most 4

Mark Bilinski

Kwang Ju Choi

Deborah Chun

Guoli Ding

Stan Dziobiak

See next page for additional authors

Recommended Citation

Bilinski, M., Choi, K., Chun, D., Ding, G., Dziobiak, S., Farnham, R., Iverson, P., Leu, S., & Lowrance, L. (2012). Bandwidth of trees of diameter at most 4. *Discrete Mathematics, 312* (12-13), 1947-1951. https://doi.org/10.1016/j.disc.2012.03.006

Discrete Mathematics 312 (2012) 1947–1951

Contents lists available at SciVerse ScienceDirect

Discrete Mathematics

journal homepage: www.elsevier.com/locate/disc

Note

Bandwidth of trees of diameter at most 4

Mark Bilinski, Kwang Ju Choi, Deborah Chun, Guoli Ding, Stan Dziobiak, Rodrigo Farnham, Perry Iverson, Shirley Leu, Lisa Warshauer Lowrance

Department of Mathematics, 303 Lockett Hall, Baton Rouge, LA 70803-4918, USA

ARTICLE INFO

Article history:
Received 26 January 2009
Received in revised form 5 March 2012
Accepted 8 March 2012
Available online 5 April 2012

Keywords:
Bandwidth
Tree
Polynomial time algorithm

ABSTRACT

For a graph G, let $\gamma : V(G) \to \{1, \dots, |V(G)|\}$ be a one-to-one function. The bandwidth of γ is the maximum of $|\gamma(u) - \gamma(v)|$ over $uv \in E(G)$. The bandwidth of G, denoted $b(G)$, is the minimum bandwidth over all embeddings γ, $b(G) = \min_\gamma\{\max\{|\gamma(u) - \gamma(v)| : uv \in E(G)\}\}$. In this paper, we show that the bandwidth computation problem for trees of diameter at most 4 can be solved in polynomial time. This naturally complements the result computing the bandwidth for 2-caterpillars.

1. Introduction

The graph terminology used here will follow West [8]. Let G be a graph. For $x \in V(G)$, let $d(x)$ denote the degree of x in G and let $\Delta(G)$ be the maximum degree among all vertices in G. Let

$$\gamma : V(G) \to \{1, \dots, |V(G)|\}$$

be a one-to-one function. The *bandwidth* of γ, denoted $b(\gamma)$, is the maximum of $|\gamma(u) - \gamma(v)|$ for $uv \in E(G)$. The bandwidth of G, denoted $b(G)$, is the minimum bandwidth over all embeddings γ. That is,

$$b(G) = \min_\gamma\{\max\{|\gamma(u) - \gamma(v)| : uv \in E(G)\}\}.$$

The *diameter* of a graph G, denoted $\operatorname{diam}(G)$, is the greatest distance between any two vertices in G. The *local density*, $\rho(G)$, is defined as

$$\rho(G) = \left\lceil \max_{G'} \frac{|V(G')| - 1}{\operatorname{diam}(G')} \right\rceil$$

where G' is taken over all connected subgraphs of G. Considering the vertices receiving the largest and smallest labels among those in G' yields the well-known lower bound $b(G) \geq \rho(G)$ [8].

k-Caterpillars are the trees obtained from paths by appending edge-disjoint paths of lengths at most k. For $k = 1$, they are simply referred to as caterpillars.

For caterpillars [5,7], there exist polynomial time algorithms to find the bandwidth. For 2-caterpillars [1], the bandwidth is known to be the local density, which can be computed in polynomial time. On the other hand, the bandwidth computation problem has been shown to be NP-complete for 3-caterpillars [6]. For more results related to bandwidth and bandwidth-like problems, the interested reader can turn to surveys [2,4].

Let $\mathcal{T}$ be the family of trees such that every leaf is within distance 2 of a root vertex r. Caterpillars are the trees obtained from paths by appending leaves, while $\mathcal{T}$ is the family obtained from stars by appending leaves. While the bandwidth equals local density for 2-caterpillars [1], this is not the case for $\mathcal{T}$. Fig. 1 shows an optimal numbering of a tree T in $\mathcal{T}$ having bandwidth 4, but $\rho(T) = 3$. Thus a different method is needed to compute the bandwidth of trees in $\mathcal{T}$.

E-mail address: mark.bilinski@gmail.com (M. Bilinski).

0012-365X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.disc.2012.03.006

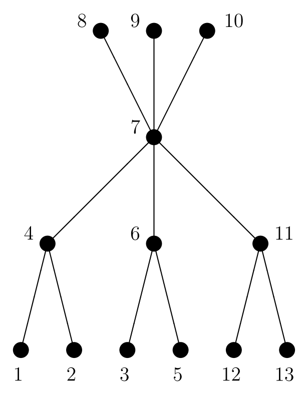

Fig. 1. Optimal numbering of a tree $T \in \mathcal{T}$ where $b(T) \neq \rho(T)$.

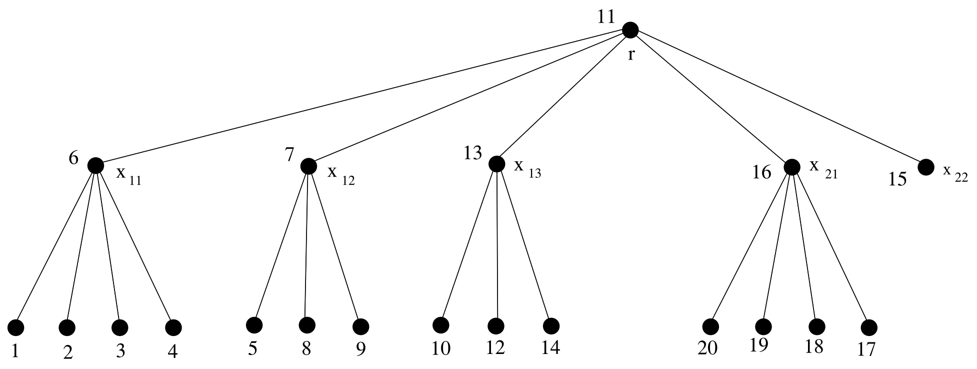

Fig. 2. Tree with root r labeled using the algorithm in Lemma 2.1.

An important step in our algorithm is to show that the bandwidth problem for $\mathcal{T}$ is equivalent to a version of the classic optimization problem PARTITION. PARTITION takes n positive integers and asks if they can be partitioned into two sets that have the same sum. While PARTITION is NP-complete [3], our algorithm runs in polynomial time thanks to a difference in input size. We show that the bandwidth computation problem can be solved in polynomial time if and only if a version of PARTITION can be solved in polynomial time based instead on the size of the tree.

2. Our bandwidth problem is a partition problem

The following technical lemma implies that the bandwidth problem for $\mathcal{T}$ is equivalent to a partition problem.

Lemma 2.1. *If $T \in \mathcal{T}$, with r the root of T and C the set of neighbors of r, then $b(T) \leq k$ if and only if:*

(1) $\Delta(T) \leq 2k$
(2) $|V(T)| \leq 4k + 1$
(3) *C has a partition (C_1, C_2) such that:*
  (3a) *$|C_1| \leq k$, and $|C_2| \leq k$*
  (3b) *$|C_1| + |C_2| + |D_1| \leq 3k$, and $|C_1| + |C_2| + |D_2| \leq 3k$, where D_i is the set of children of C_i for $i = 1, 2$.*

Proof. Suppose $b(T) \leq k$. Let γ be a numbering of T with $b(\gamma) = b(T)$. Let C_1 and C_2 be the sets of neighbors of r with labels less than $\gamma(r)$ and greater than $\gamma(r)$, respectively. Note that $|C_1| \leq k$ and $|C_2| \leq k$. Also at most $2k - |C_1|$ neighbors of C_1 have labels less than $\gamma(r)$, and at most $k - |C_2|$ neighbors of C_1 have labels greater than $\gamma(r)$. Thus $|D_1| \leq 3k - |C_1| - |C_2|$, as desired. Similarly, $|D_2| \leq 3k - |C_1| - |C_2|$. Since $\operatorname{diam}(T) \leq 4$, the local density bound yields $|V(T)| \leq 4k + 1$ and $\Delta(T) \leq 2k$.

Let $T \in \mathcal{T}$ satisfying (1) and (2), and suppose a partition of the children of the root, (C_1, C_2) satisfying (3) exists. Let $C_1 = \{x_{11}, \ldots, x_{1n_1}\}$ with $d(x_{11}) \geq \cdots \geq d(x_{1n_1})$, and let $C_2 = \{x_{21}, \ldots, x_{2n_2}\}$ with $d(x_{21}) \geq \cdots \geq d(x_{2n_2})$, and further assume $|C_1| + |D_1| \geq |C_2| + |D_2|$. The following algorithm produces a labeling demonstrating $b(T) \leq k$. Fig. 2 shows a tree with root r and children partitioned as (C_1, C_2) that satisfies conditions (1)–(3) for $k = 5$. Its vertices have been labeled according to the algorithm. Note that steps (1)–(4) assign labels 1–10, steps (5)–(6) assign labels 11–14, and steps (7)–(10) assign labels 20 to 15.

Algorithm. Input: k, a positive integer and $T \in \mathcal{T}$ satisfying conditions (1)–(3) for k and partition (C_1, C_2).
Output: $\gamma(v)$ a labeling of the vertices of T into $\{1, \dots, |V(T)|\}$.
Label with the smallest unassigned value beginning with 1:

(1) At most k grandchildren, from children of x_{11} through children of x_{1n_1}.
(2) x_{11} through x_{1i}, where x_{1i} is the last vertex whose child has a label.
(3) The remaining children of x_{1i}.
(4) Up to label $2k$: half of the children of x_{1j} (rounded up), x_{1j}, then the remaining children of x_{1j} for j from $i + 1$ to n_1.
(5) r.
(6) Resume (4), without the $2k$ restriction, for the rest of x_{1j} and their children.

Label with the largest unassigned value beginning with $|V(T)|$:

(7) At most k grandchildren, from children of x_{21} through children of x_{2n_2}.
(8) x_{21} through x_{2l}, where x_{2l} is the last vertex whose child has a label.
(9) The remaining children of x_{2l}.
(10) Half of the children of x_{2j} (rounded up), x_{2j}, then the remaining children of x_{2j}, for j from $l + 1$ to n_2.

It remains to show this algorithm demonstrates a bandwidth of at most k.

Consider x_{1j} and y, where $j < i$ and y is a child of x_{1j}. By Step 1 and as the x_{1j} are sorted by degree, $|\gamma(x_{1j}) - \gamma(y)| \le k$.

Next consider x_{1i} and y, where y is a child of x_{1i}. As above, if $\gamma(y) < \gamma(x_{1i})$, then $|\gamma(x_{1i}) - \gamma(y)| \le k$. Assume for contradiction that there are more than $k - 1$ children of x_{1i} with label greater than $\gamma(x_{1i})$. Then $d(x_{1j}) \ge k + 1$ for $j < i$ and hence x_{1j} has at least k children. By steps (1)–(2), $i = 1$. But then x_{11} has at least $2k$ children, contradicting $\Delta(T) \le 2k$. Thus there are at most $k - 1$ children of x_{1i} with label greater than $\gamma(x_{1i})$ and hence by step (3) $|\gamma(x_{1i}) - \gamma(y)| \le k$ for any child y of x_{1i}.

Next consider x_{1j} and y, where $j > i$ and y is a child of x_{1j}. As the label of any such x_{1j} is centered among the labels of its children by steps (4) and (6), $|\gamma(x_{1j}) - \gamma(y)| \le k$ for any x_{1j} all of whose children have labels on one side of $\gamma(r)$. Further if $i \ge 2$, then by construction x_{1j} has less than k children and hence $|\gamma(x_{1j}) - \gamma(y)| \le k$. Thus we may assume $i = 1$ and x_{12} has children with labels both less and greater than $\gamma(r)$. Note that $d(x_{11}) + d(x_{12}) \le |C_1| + |D_1| \le 3k$ by condition (3b). Thus in any case, $|\gamma(x_{1j}) - \gamma(y)| \le k$, where $j = 2$.

By construction the x_{2j} and their children all receive a label. By the symmetry between steps (1)–(4) and (7)–(10) in the algorithm, $|\gamma(x_{2j}) - \gamma(y)| \le k$ for any child y of x_{2j}.

Simple case analysis of $|D_1|$ and $|D_1| + |C_1|$ shows that $|\gamma(r) - \gamma(x_{1j})| \le k$ for any j. A similar result holds for the x_{2j}. $\square$

Note that the algorithm runs in time $O(|V(T)|)$, given a partition satisfying conditions (1)–(3) and given that the children of the root are sorted by degree. If the children are not sorted by degree, we must of course sort them, and the algorithm runs in time $O(|V(T)| \log |V(T)|)$.

3. Pseudo-polynomial partition algorithm

In this section, we develop an algorithm to solve the partition problem in pseudo-polynomial time. This algorithm will then be used to solve the bandwidth computation problem for $\mathcal{T}$ in polynomial time because the input size for the bandwidth problem is the size of tree, which is slightly larger than the input size for the standard partition problem.

First we formally define a variant of the standard partition problem.

FIXED SIZE SUBSET SUM.

Instance: $a_1, \dots, a_n, m, N$ non-negative integers.

Question: Is there a subset $I \subseteq \{1, \dots, n\}$ such that $|I| = m$ and $\sum_{i \in I} a_i = N$?

Note that the input size is $O(n + \log_2 m + \log_2 N + \sum_{i=1}^{n} \lfloor \log_2 a_i \rfloor)$. It is easy to see that FIXED SIZE SUBSET SUM is NP-complete and hence no known algorithm solves the problem in polynomial time based on the size of the input.

Below we give a dynamic programming algorithm that solves the problem in pseudo-polynomial time, but before we begin, we first give a definition to make clear what is meant in this algorithm. Let $\mathcal{I}$ be a family of sets $I \subseteq \{1, .., n\}$. I is said to be the *lexicographically minimum set* if for any $J \in \mathcal{I}$, $\min\{I \triangle J\} \in I$, where $\triangle$ is the symmetric difference. For example, if $\mathcal{I} = \{\{1, 3, 4\}, \{2, 4\}, \{1, 2, 4\}\}$, $\{1, 2, 4\}$ would be the lexicographically minimum set. As the algorithm will deal with finite families of finite sets, the lexicographically minimum set is well defined.

Lemma 3.1. *The problem* FIXED SIZE SUBSET SUM *can be solved in polynomial time based on the input size: $n + m + N + \sum_{i=1}^{n} \lfloor \log_2 a_i \rfloor$.*

Proof. For any $I \subseteq \{1, \dots, n\}$ let $a(I) := \sum_{i \in I} a_i$. For any two non-negative integers p, q, let

$$f(p, q) = \begin{cases} * & \text{if no } I \text{ satisfies } |I| = p \text{ and } a(I) = q \\ \{a_i : i \in I\} & \text{otherwise, where } I \text{ is the lexicographically minimum set that satisfies } |I| = p \text{ and } a(I) = q. \end{cases}$$

Note that $f(m, N)$ either demonstrates the partition for a positive outcome of FIXED SIZE SUBSET SUM or indicates that no such partition exists. Below we present an algorithm which iteratively populates the two-dimensional table of values for $f(p, q)$ and ultimately outputs the desired $f(m, N)$.

Algorithm. Input: $a_1, \dots, a_n, m, N$ of non-negative integers.

```
Output: $f(m, N)$
    if $pq = 0$, then
        $f(p,q) := \begin{cases} * & \text{if } p = 0 \\ & \text{if } q = 0 < p \text{ and there are fewer than } p \text{ indices } i \text{ with } a_i = 0 \\ \{a_{i_1}, \dots, a_{i_p}\} & \text{if } q = 0 < p \text{ and } a_{i_1}, \dots, a_{i_p} \text{ are the first } p \text{ terms with } a_i = 0 \end{cases}$
for $p = 1, \dots, m$
    for $q = 1, \dots, N$
        for $i = 1, \dots, n$
            if $f(p-1, q-a_i) = *$ or $f(p-1, q-a_i) = \{a_{i_1}, \dots, a_{i_{p-1}}\}$ with $i_{p-1} \geq i$, then $i := i + 1$;
            if $f(p-1, q-a_i) = \{a_{i_1}, \dots, a_{i_{p-1}}\}$ with $i_{p-1} < i$, then $f(p,q) := \{a_{i_1}, \dots, a_{i_{p-1}}, a_i\}$ and break;
            $f(p,q) := *$;
        $q := q + 1$;
    $p := p + 1$;
return $f(m, N)$.
```

It is easy to see that the algorithm iteratively creates a two-dimensional table of values f, where the entry $f(p,q)$ holds the value $*$ (for 'False') if there is no I satisfying $|I| = p$ and $a(I) = q$, and $\{a_i : i \in I\}$ if I is the lexicographically minimum set that satisfies $|I| = p$ and $a(I) = q$. Note that initializing the first row and the first column of the table f (corresponding to $p = 0$ or $q = 0$) takes $O(n)$ time, and once inside the three 'for' loops, the 'if' statements and setting $f(p,q)$ take a combined $O(1)$ time. Hence the running time of the above algorithm is $O(n^2N)$. Alternatively, the running time is $O(x^3)$, where $x := n + m + N + \sum_{i=1}^{n} \lfloor \log_2 a_i \rfloor$. $\square$

4. Our bandwidth problem in polynomial time

The characterization from Lemma 2.1 combined with the FIXED SIZE SUBSET SUM algorithm gives the following result.

Theorem 4.1. *If $T \in \mathcal{T}$, then $b(T)$ can be computed in polynomial time.*

Proof. Let r be the root of T, and let $C = \{x_1, \dots, x_n\}$ be the set of children of r. For $i = 1, \dots, n$, let a_i be the number of children of x_i, and let $a := \sum_{i=1}^{n} a_i$. Let $|T| = |V(T)|$.

We will give an algorithm that computes the bandwidth $b(T)$. It is motivated by the following facts. Note that $\max\{\lceil \frac{|T|-1}{4} \rceil, \lceil \frac{\Delta(T)}{2} \rceil\} \leq \rho(T) \leq b(T) \leq |T| - 1$. Hence, by Lemma 2.1, there is a k such that $\max\{\lceil \frac{|T|-1}{4} \rceil, \lceil \frac{\Delta(T)}{2} \rceil\} \leq k \leq |T| - 1$ satisfying conditions (1)–(3) of the lemma. The algorithm will find the smallest such k (thus finding $b(T)$) by invoking the FIXED SIZE SUBSET SUM algorithm with a specific input.

First, let $k := b(T)$. Then, k is the smallest integer with $\max\{\lceil \frac{|T|-1}{4} \rceil, \lceil \frac{\Delta(T)}{2} \rceil\} \leq k \leq |T| - 1$ for which the conditions (1)–(3) of Lemma 2.1 are satisfied.

Let $m := |C_1|$, and $N := |D_1|$, so that $|C_2| = n - m$, and $|D_2| = a - N$. Hence, condition (3a) implies $n - k \leq m \leq k$ and condition (3b) implies $a - 3k + n \leq N \leq 3k - n$. Up to a reindexing of the x_i's (and the corresponding a_i's) we have that $C_1 = \{x_1, \dots, x_m\}$. Then, since D_1 is the set of children of C_1, we have that $\sum_{i=1}^{m} a_i = N$, so that for these prescribed values of k, m, and N, FIXED SIZE SUBSET SUM $(a_1, \dots, a_n, m, N)$ returns 'True'. In short, because of these structural properties of the tree, FIXED SIZE SUBSET SUM returns 'True' for this particular combination of k, m, N.

Thus we create an algorithm to test different combinations of k, m, N, as we know FIXED SIZE SUBSET SUM will return 'True' at least for that precise combination. Further, we need not test all combinations of these three positive integers – instead we need only test them between the already mentioned finite bounds. Note that it will remain to show that only such a precise combination resulting from the structure of T will result in a 'True' output from FIXED SIZE SUBSET SUM within this algorithm.

Algorithm. Input: $T \in \mathcal{T}$

```
Output: $k = b(T)$
for $k$ from $\max\{\lceil \frac{|T|-1}{4} \rceil, \lceil \frac{\Delta(T)}{2} \rceil\}$ to $|T| - 1$
    for $m$ from $n - k$ to $k$
        for $N$ from $a - 3k + n$ to $3k - n$
            if FIXED SIZE SUBSET SUM $(a_1, \dots, a_n, m, N)$, then return $k$;
            $N := N + 1$;
        $m := m + 1$;
    $k := k + 1$;
return $k$;
```

Conversely, assume that for some k with $\max\{\lceil\frac{|T|-1}{4}\rceil, \lceil\frac{\Delta(T)}{2}\rceil\} \leq k \leq |T|-1$, the FIXED SIZE SUBSET SUM algorithm returns 'True' with input $a_1, \ldots, a_n, m, N$ such that $n-k \leq m \leq k$ and $a-3k+n \leq N \leq 3k-n$. This means that, up to a reindexing of the a_i's (and the corresponding x_i's), $\sum_{i=1}^{m} a_i = N$. Let $C_1 := \{x_1, \ldots, x_m\}$, $C_2 := C - C_1$, and for $i = 1, 2$ let D_i be the set of children of C_i. Hence, $|C_1| = m$, $|C_2| = n-m$, $|D_1| = N$, $|D_2| = a-N$. Then, $n-k \leq m \leq k$ implies that $|C_1| \leq k$ and $|C_2| \leq k$. Similarly, $a-3k+n \leq N \leq 3k-n$ implies that $|D_1| + |C| \leq 3k$, and $|D_2| + |C| \leq 3k$. Also, by assumption, $|T| \leq 4k+1$ and $\Delta(T) \leq 2k$, hence the conditions (1)–(3) of Lemma 2.1 are satisfied.

Thus, we have shown that starting from the value of $k = \max\{\lceil\frac{|T|-1}{4}\rceil, \lceil\frac{\Delta(T)}{2}\rceil\}$ and up to the value of $k = |T|-1$ (if necessary), our algorithm successively checks whether $b(T) \leq k$. At the same time, FIXED SIZE SUBSET SUM is guaranteed to return a 'True' statement for some combination of k, m, N. Hence this algorithm will return the smallest k such that $b(T) \leq k$, which is the bandwidth $b(T)$.

By Lemma 3.1, the FIXED SIZE SUBSET SUM algorithm runs in time cubic in $n+m+N+\sum_{i=1}^{n}\lfloor\log_2 a_i\rfloor$. As $n+m+N+\sum_{i=1}^{n}\lfloor\log_2 a_i\rfloor < 6|T|$, each invocation FIXED SIZE SUBSET SUM runs in time $O(|T|^3)$. Since each of the three 'for' loops of our algorithm iterates at most $O(|T|)$ times, our algorithm runs in time $O(|T|^3|T|^3) = O(|T|^6)$. □

It may be possible to extend this result to trees of any bounded diameter. Having that result would be very interesting, and may have further implications for determining which classes of trees have polynomial-time algorithms for computing the bandwidth.

Acknowledgments

We want to thank the VIGRE program which provided funding to make the LSU VIGRE Combinatorics Crew possible. We would especially like to thank the referees for their very thorough and helpful comments in revising this paper.

References

[1] S.F. Assmann, G.W. Peck, M.M. Syslo, J. Zak, The bandwidth of caterpillars with hairs of length 1 and 2, SIAM J. Algebr. Discrete Methods 2 (1981) 387–393.
[2] P.Z. Chinn, J. Chavatalova, A.K. Dewdney, N.E. Gibbs, The bandwidth problem for graphs and matrices—a survey, J. Graph Theory 6 (1982) 223–254.
[3] M.R. Garey, R.L. Graham, D.S. Johnson, D.E. Knuth, Complexity results for bandwidth minimization, SIAM J. Appl. Math. 34 (3) (1978) 477–495.
[4] Y.L. Lai, K. Williams, A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs, J. Graph Theory 31:2 (1999) 75–94.
[5] Z. Miller, The bandwidth of caterpillar graphs, in: Proc. Twelfth Southeastern Conf. on Combinatorics, in: Graph Theory and Computing, vol. II, 1981, pp. 235–252.
[6] B. Monien, The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete, SIAM J. Algebr. Discrete Methods 7 (4) (1986) 505–512.
[7] M.M. Syslo, J. Zak, The bandwidth problem: critical subgraphs and the solution for caterpillars, Ann. Discrete Math. 16 (1982) 281–286.
[8] D. West, Introduction to Graph Theory, second ed., Prentice Hall, 2001.