Terminal Wiener Index of Fibonacci trees

Sulphikar A

Abstract: The terminal Wiener index of a tree is defined as the sum of distances between all leaf pairs of T. We derive closed form expression for the terminal Wiener index of fibonacci trees. We also describe a linear time algorithm to compute terminal Wiener index of a tree.

Keywords: Terminal Wiener index, fibonacci tree, Pendent vertex, distance in graphs.

I. INTRODUCTION

For a tree T, the terminal Wiener index TW(T) of T is defined as the sum of the distances between all pairs of leaves of T. That is

\[ TW(T) = \sum_{1 \leq i < j \leq k} d_T(v_i, v_j) \]

where \( v_i, v_j \) are leaves in T.

Let T be an n-vertex tree with k leaves. Then TW(T) can be expressed as

\[ TW(T) = \sum_e p(e)p'(e) \]

where \( p(e) \) and \( p'(e) \) are the number of leaves in two components of \( T - e \).

Section 2 outline an algorithmic approach to compute terminal Wiener index of fibonacci trees and binary fibonacci trees. Section 3 explain an algorithm to compute TWI of a tree.

The paper[9] describe a method to compute terminal Wiener index of balanced trees.

II. PROPOSED METHOD

We propose a method to compute terminal Wiener index of fibonacci trees.

2.1 Fibonacci trees and Binary fibonacci trees

We begin with lemma 1 below.

Lemma 1:

Let T be a tree composed of two disjoint trees \( T_1 \) and \( T_2 \) respectively. Let \( x \in V(T_1) \), \( y \in V(T_2) \) and \( xy \) be a cutedge in T. Let \( I_1 \) and \( I_2 \) be the number of leaves in \( T_1 \) and \( T_2 \) respectively.

For any vertex \( u \in V(T) \) let \( d'(u) \) denote the sum of the distances from \( u \) to every leaf in T. Then

\[ TW(T) = TW(T_1) + TW(T_2) + I_1d'(y) + I_2d'(x) + I_1I_2. \]  

(2)

Let \( F_k \) denote the \( k \)th Fibonacci number. The Fibonacci tree \( T_{fb} \) of order \( k \) [2,10], is defined recursively in the following way:

\( T_{fb} \) and \( T_{rb} \) are both rooted trees consisting of no nodes and a single node respectively.

For \( k \geq 2 \), \( T_{fb} \) consist of a root with two fibonacci trees, \( T_{fb-1} \) and \( T_{fb-2} \) as left and right child respectively. Figure 2 shows binary fibonacci trees \( T_{fb} \) through \( T_{fb} \) and Terminal Wiener index of a Fibonacci tree

Theorem 2.1

Let \( T_{fb} \) be a fibonacci tree. Then its terminal Wiener index is given by

\[ TW(T_{fb}) = TW(T_{fb-1}) + TW(T_{fb-2}) + F_kd_{fb} \]

(3)

where \( k \) is its order.

Proof.

In order to compute \( TW(T_{fb}) \), we first obtain a closed form expression for \( d_{fb}'(k) \), sum of the distance between root of \( T_{fb} \) and its leaves. From fig.1, we have

\[ d_{fb}'(k) = d_{fb}'(k-1) + d_{fb}'(k-2) + F_k, \quad k \geq 2. \]

(4)

with \( d_{fb}'(0) = 0 \), \( d_{fb}'(1) = 1 \), and \( d_{fb}'(2) = 2 \).

We introduce the generating function \( G(z) \) as
Terminal Wiener Index of Fibonacci trees

\[ G(z) = d^+_{T_f}(1)z + d^+_{T_f}(2)z^2 + d^+_{T_f}(3)z^3 + d^+_{T_f}(4)z^4 + \ldots \]
(5)

2.2

From (eq:4) and (eq:5) we get

\[ (1-z^{-3})G(z)=1+Fz^2 + Fz^3 + Fz^4 + \ldots = 1 + \frac{z}{1-z^{-2}} \]
(6)

\[ G(z) \text{ can be obtained from (eq:6) as} \]
\[ G(z)=1/(1-1/z^{-3}) \]

\[ (1-\varphi z^3) = 1/5z(1-\varphi z^3) + 1/(1-\varphi z^3) \]

where \( \varphi = (1+\sqrt{5})/2 \) and \( \varphi^* = (1-\sqrt{5})/2 \).

It follows (see [8]) that k+1

\[ [z^k]G(z)=d^+_{T_f}(k) = F_k + \sum F_{j,1}F_{k,j+1} = \frac{1}{5} [(k+4)F_k+2kF_{k-1}], \]

j=2

substituting \( l_1=F_k \) and \( l_2=F_{k-1} \) in lemma 1 we get

\[ TW(T_{f(k)})=TW(T_{f(k-1)})+TW(T_{b(k-2)})+F_kd^+_{T_f}(k-2)+F_{k-1}d^+_{T_f}(k-1)+F_kF_{k-1} \]
\[ TW(T_{f(k)}) = 0 \text{ and } TW(T_{b(k)})=2. \]

The following algorithm computes \( TW(T_{f(k)}) \).

1. Procedure TW-FIB(k)
2. For i=1 to k do
3. Compute \( F_i \)
4. \( TW_i=0 \)
5. \( TW_{i+1}=2 \)
6. For j=3 to k do
7. \( D = 0.2((j+3)F_j + (j-1)F_{j-2}) \)
8. \( D' = 0.2((j+2)F_j + (j-2)F_{j-3}) \)
9. \( TW = TW_i + TW_{i+1} + F_j + D + F_j' \)
10. \( TW_i=TW_{i+1} \)
11. \( TW_{i+1}=TW \)
12. return TW
13. EndProcedure

Theorem 2.2

For a fibonacci tree of order k denoted by \( T_k \) we can compute \( TW(T_{f(k)}) \) in \( O(\log F_k) \). time.

Proof.

It is easy to see that in TW-FIB(k) step 2 requires atmost \( \log F_k \) additions and steps 7-12 require atmost \( \log F_k \) multiplications and additions.

2.3 Terminal Wiener index of a Binary Fibonacci tree

Theorem 2.3

Let \( T_k \) be a binary fibonacci tree of order k[10]. Then its terminal Wiener index is given by

\[ TW(T_k) = TW(T_{f(k)}) + TW(T_{b(k)}) + F_kd^+_{T_f}(k-2) + F_{k-1}d^+_{T_f}(k-1) + F_kF_{k-2}, k \geq 4. \]
(7)

Proof.

Consider the Fibonacci tree \( T_k \) of fig.2. In computing \( TW(T_{f(k)}) \), we first obtain a closed form expression for \( d^+_{T_f}(k) \). From fig. 7, we have

\[ d^+_{T_f}(k) = d^+_{T_f}(k-1) + d^+_{T_f}(k-2) + F_k, \]
(8)

with \( d^+_{T_f}(0)=0 \), \( d^+_{T_f}(1)=0 \), and \( d^+_{T_f}(2)=1 \).

Using method similar to section 2.2, we get

\[ [z^k]G(z)=d^+_{T_f}(k) = \sum_{j=1}^{k-2} F_jF_k \text{ } \text{ } \text{ } k + j = 1 - F_k \]

\[ = \frac{1}{2} (kF_{k-1} + F_kF_{k-3}) \]

By taking \( l_1=F_{k-1} \) and \( l_2=F_{k-2} \), by Lemma 1 we get

\[ TW(T_{b(k)}) = TW(T_{f(k)}) + TW(T_{b(k-2)}) + F_kd^+_{T_f}(k-2) + F_{k-1}d^+_{T_f}(k-1) + F_kF_{k-2}, k \geq 4. \]
(9)

with \( TW(T_{f(2)}) = 0 \) and \( TW(T_{b(3)}) = 3 \).

Using eq:9 we can compute \( TW(T_{b(k)}) \) similar to the algorithm TW-FIB(k).

III. AN ALGORITHM TO COMPUTE TERMINAL WIENER INDEX

We outline an algorithm to compute TWI of a tree T, which uses tree reduction and vertex weighting. Each vertex \( u \) is assigned two weights, \( w[u] \) and \( w'[u] \).

1. procedure TWI(T) \> This computes TWI of a tree using tree reduction
2. \( p \leftarrow 0 \)
3. for \( i = 1 \) to \( |V(T)| \) do
4. \( w[i] \leftarrow 0 \)
5. \( f[i] \leftarrow 0 \) \> Indicates whether vertex \( v \) is visited or not
6. if degree[\( v \)] = 1 then
7. \( p \leftarrow p + 1 \)
8. \( f[i] \leftarrow 1 \)
9. end if
10. end for
11. for \( i = 1 \) to \( |V(T)| \) do
12. if degree[\( v \)] = 1 and \( f[i] = 1 \) then \> Check whether \( v \) is pendant vertex
13. Choose a neighbor \( y \) of \( u \).
14. \( w[y] \leftarrow w[y] + p - 1 \)
15. Remove the edge \((v,y)\)
16. Remove the vertex \( v \)
17. end if
18. end for
19. \( w' = w \)
20. while \( E \) is not empty do
21. Select a pendant vertex \( v \)
22. Choose a neighbor \( u \) of \( v \).
23. \( x \leftarrow w'[u]/(p - 1) \)
24. \( w'[u] \leftarrow w'[u] + w'[x] \)
25. \( w'[u] \leftarrow w'[u] + w'[x] + x \leftarrow x \leftarrow x \leftarrow x \)
26. Remove the edge \((u,v)\)
27. Remove the vertex \( v \)
28. end while
29. end procedure

The best case occurs if T is a star and the worst case occurs if T is a path. If the T is a star, the whole loop in lines 20-28 is not executed at all. The maximum number of iterations for both while loops together can not exceed the number of vertices in T. Therefore, the complexity of the above algorithm is \( O(n) \).

IV. RESULT ANALYSIS

The above algorithm is implemented using Python 2.7 and NetworkX. The input to the algorithm is a tree and the tree is reduced by removing pendant vertices one at a time.
Each time a pendent vertex is removed, the weights are updated. Finally the tree reduces to single vertex, in which case its weight gives \( \text{TWI} \) of the tree. Fig.3 shows a tree with terminal Wiener index \( = 82 \) and Table 1 list the values of \( w \) and \( w' \) during the execution of the algorithm.

![Figure 3. A tree T with TWI(T) = 82](image)

| vertex(v) | edge(v-u) | w(u) | w'(u) |
|-----------|-----------|------|-------|
| 1         | 1-2       | 6    | -     |
| 4         | 3-4       | 6    | -     |
| 8         | 6-5       | 6    | -     |
| 9         | 9-10      | 6    | -     |
| 12        | 12-11     | 6    | -     |
| 13        | 13-11     | 12   | -     |
| 2         | 2-3       | 18   | 12    |
| 3         | 3-5       | 34   | 18    |
| 7         | 7-5       | 46   | 24    |
| 10        | 10-11     | 24   | 18    |
| 5         | 5-11      | 82   | 42    |

Table 1. Steps for computing TWI of tree in fig.3

V. CONCLUSION

This paper introduces an efficient way to compute terminal Wiener index of Fibonacci trees and binary Fibonacci trees. For binary trees, we developed an algorithm to compute TWI in \( O(\log(F_k)) \) time. We also introduced an algorithm for computing TWI of any tree in linear time.

REFERENCES

1. Y. H. Chen, X. D. Zhang (2013) On Wiener and terminal Wiener indices of trees, MATCH Commun. Math. Comput. Chem. 70, 591-602.
2. T. H. Cormen, C. E. Leiserson and R. L. Rivest (1994) Introduction to Algorithms, The MIT Press.
3. P. Dankelmann (1993) Computing the average distance of an interval graph, Inform. Process. Lett. 48, 311-314.
4. A. A. Dobrynin, R. Entringer, I. Gutman (2001) Wiener index of trees: theory and applications, Acta Appl. Math. 66, 211-249.
5. A. A. Dobrynin, I. Gutman, S. Klavzar and P. Zergert (2002) Wiener index of hexagonal systems, Acta Appl. Math. 72, 247-294.
6. I. Gutman and B. Furtula (2010) A survey on terminal Wiener index, in I. Gutman and B. Furtula (Eds.) Novel Molecular Structure Descriptors - Theory and Applications, Kragujevac, 173-190.
7. I. Gutman, B. Furtula and M. Petrovic (2009) Terminal Wiener index, J. Math. Chem. 46, 522-531.
8. D. E. Knuth The Art of Computer Programming, vol. 1, 3/e, Addison-Wesley, 1997.
9. A Sulphikar (2019) Terminal Wiener index of balanced trees, vol. 8, Int'l Journal of Recent Technology and Engg, 2270-74.
10. K. Viswanathan Iyer, K. R. Uday Kumar Reddy (2009) Wiener index of binomial trees and Fibonacci trees, Int'l J. Math. Engg. with Comp.
11. The distances between internal vertices and leaves of a tree, European Journal of Combinatorics, 41, 79-99.