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)[7] of T is defined as the sum of the distances between all pairs of leaves of T. That is

$$TW(T) = \sum_{1 \le i < j \le k} d_T(v_i, v_j)$$

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

$$TW(T) = \sum_e p(e)p'(e) \qquad (1)$$

where p(e) and p′(e) are the number of leaves in two components of T − e[6].

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 lemma1 below.

Lemma1:

Let T be a tree composed of two disjoint trees T_1 and T_2 respectively. Let x ∈ V(T_1) , y ∈ V(T_2) and xy be a cutedge in T. Let l_1 and l_2 be the number of leaves in T_1 and T_2 respectively.

For any vertex u ε V(T) let $d^+(u)$ denote the sum of the distances from u to every leaf in T. Then $TW(T)=TW(T_1)+TW(T_2)+l_1d^+(y)+l_2d^+(x)+l_1l_2$. (2)

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

T_{f-1} and T_{f0} are both fibonacci trees consisting of a single node.

For k>=1, T_{fk} consist of two fibonacci trees T_{fk-1} and T_{fk-2} of orders k-1 and k-2 respectively, where T_{fk-2} is the rightmost child of the root of the other.

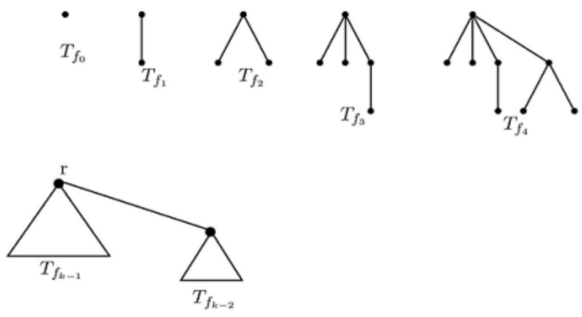

Figure 1. A Fibonacci tree T_{f_k}

Let T_{fk}^b denote the binary Fibonacci tree of order k [2,10], is defined recursively in the following way:

T_{f0}^b and T_{f1}^b are both rooted trees consisting of no nodes and a single node respectively.

For k>=2, T_{fk}^b consist of a root with two fibonacci trees, T_{fk-1}^b and T_{fk-2}^b as left and right child respectively.

Figure 2 shows binary fibonacci trees T_{f1}^b through T_{f4}^b.

Terminal Wiener index of a Fibonacci tree

Theorem 2.1

Let T_{fk} be a fibonacci tree. Then its terminal Wiener index is given by

$TW(T_{fk})=TW(T_{fk-1})+TW(T_{fk-2})+F_k d_{Tf}^+(k-2)+F_{k-1} d_{Tf}^+(k-1)+F_k F_{k-1}$, k ≥3 (3)

where k is its order.

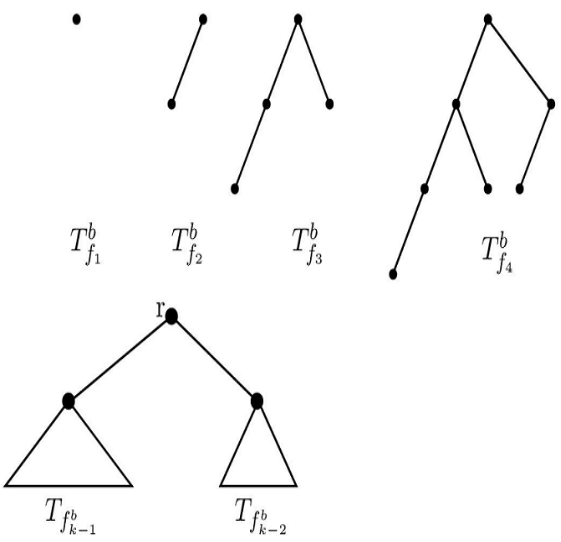

Figure 2. Binary Fibonacci tree $T_{f_k}^b$

Proof.

In order to compute $TW(T_{fk})$, we first obtain a closed form expression for $d^+_{Tf}(k)$, sum of the distance between root of $T_f(k)$ and its leaves. From fig.1, we have

$d^+_{Tf}(k)=d^+_{Tf}(k-1)+d^+_{Tf}(k-2)+F_{k-1}$, (4)

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

We introduce the generating function G(z) as

Revised Manuscript Received on February 05, 2020.

Sulphikar A, Department of Computer Science and Engineering, National Institute of Technology, Trichy, Tamilnadu INDIA

$G(z) = d^{+}_{Tf}(1)z + d^{+}_{Tf}(2)z^2 + d^{+}_{Tf}(3)z^3 + d^{+}_{Tf}(4)z^4 + \ldots$ (5)

2.2

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

$(1-z-z^2)G(z)=1+F_1z + F_2z^2 + F_3z^3+F_4z^4 + \ldots=1+ \frac{z}{1-z-z2}$ (6)

G(z) can be obtained from (eq:6) as

$G(z)=1/(1-z-z^2)+(1-z-z^2)^2\}=1/5z(1/(1-\varphi z)^2+1/(1-\varphi' z)^2\}+4z/(1-z-z^2)$

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

It follows(see [8]) that

$$[z^k]G(z)=d^{+}_{Tf}(k) = F_k+ \sum_{j=2}^{k+1} F_{j-1} F_{k-j+1}= \frac{1}{5}[(k+4)F_k+2kF_{k-1}].$$

substituting $l_1=F_k$ and $l_2=F_{k-1}$ in *lemma1* we get

$TW(T_{fk})=TW(T_{fk-1})+TW(T_{fk-2})+F_kd_{Tf}^{+}(k-2)+F_{k-1}d_{Tf}^{+}(k-1)+F_kF_{k-1}$, $k>=3$

with $TW(T_{f1})=0$ and $TW(T_{f2})=2$.

The following algorithm computes $TW(T_{fk})$.

1. Procedure TWI-FIB(k)
2. For i=1 to k do
3. Compute F_i
4. $TW_1= 0$
5. $TW_2=2$
6. For j=3 to k do
7. $D = 0.2((j+3)F_{j-1} + 2(j-1)F_{j-2})$
8. $D' =0.2((j+2)F_{j-2}+ 2(j-2)Fj-3)$
9. $TW = TW_1+TW_2+ F_{j-1}D+F_jD'$
10. $TW_1=TW_2$
11. $TW_2=TW$
12. return TW
13. EndProcedure

Theorem 2.2

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

Proof.

It is easy to see that in TWI-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_{fk}^{b} be a binary fibonacci tree of order k[10]. Then its terminal Wiener index is given by

$TW(T_{fk}^{b})=TW(T_{fk-1}^{b})+TW(T_{fk-2}^{b})+F_{k-1}d_{Tf}^{b+}(k-2)+F_{k-2}d_{Tf}^{b+}(k-1)+F_{k-1}F_{k-2}, k\geq4.$ (7)

Proof.

Consider the Fibonacci tree T_{fk} of fig.2 . In computing $TW(T_{fk})$, we first obtain a closed form expression for $d^{+}_{Tf}{}^{b}(k)$. From fig.\ref{fig:7}, we have

$d^{+}_{Tf}b(k)=d^{+}_{Tf}{}^{b}(k-1)+d^{+}_{Tf}{}^{b}(k-2)+F_k$, (8)

with $d^{+}_{Tf}{}^{b}(0)=0$, $d^{+}_{Tf}{}^{b}(1)=0$, and $d^{+}_{Tf}{}^{b}(2)=1$.

Using method similar to section 2.2, we get

$[z^k]G(z)=d^{+}_{Tf}{}^{b}(k)=\sum_{j=1}^{k+1} FjF\ k-j+1-Fk$

$=\frac{1}{5}[kF_{k+2}+(k-3)F_k]$.

By taking $l_1=F_{k-1}$ and $l_2=F_{k-2}$, by *lemma1* we get

$TW(T_{fk}^{b})=TW(T_{fk-1}^{b})+TW(T_{fk-2}^{b}\})+F_{k-1}d_{Tf}^{b+}(k-2)+F_{k-2}d_{Tf}^{b+}(k-1)+F_{k-1}F_{k-2}$, $k \geq 4$. (9)

with $TW(T_{f2}^{b})=0$ and $TW(T_{f3}^{b})=3$.

Using eq:9 we can compute $TW(T_{fk}^{b})$ similar to the algorithm TWI-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 i is visited or not
6. **if** $degree[v_i] = 1$ **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** $degree(v_i) = 1$ and $f[i] = 1$ **then** ▷ Check whether v_i is pendent vertex or not
13. Choose a neighbor y of v_i.
14. $w[y] \leftarrow w[y] + p - 1$
15. Remove the edge (v_i, y)
16. Remove the vertex v_i
17. **end if**
18. **end for**
19. w'=w
20. **while** E is not empty **do**
21. Select a pendent vertex v
22. Choose a neighbor u of v .
23. $x \leftarrow w'[v]/(p-1)$
24. $w'[u] \leftarrow w'[u] + w'[v]$
25. $w[u] \leftarrow w[u] + w[v] + x(p-x)$
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 while 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 pendent 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 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.

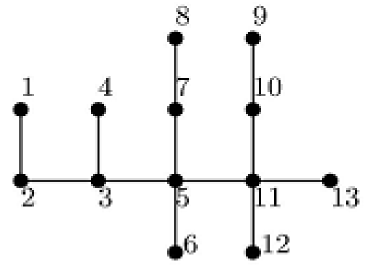

Figure 3. A tree T with $TW(T) = 82$

vertex(v)	edge(v-u)	w(u)	w'(u)
1	1-2	6	-
4	4-3	6	-
8	8-7	6	-
6	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 fibonacci trees, we developed an algorithm to compute TWI in O(log(Fk)) time. We also intoduced 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. Klavazar and P. Zigert(2002) Wiener index of hexagonal sytems , 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.

AUTHORS PROFILE

Mr.Sulphikar.A B.Tech, M.Tech is doing Ph.D at National institute of technology, Tiruchirappalli, Tamilnadu,India. His research areas include algorithms and graph theory.