Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
DOI : 10.47974/JIOS-2176

Log-linear algorithm to generate prime trees

Karnam Gurunadhan Tharunraj †
P. Ragukumar *
Department of Mathematics
School of Advanced Sciences
Vellore Institute of Technology
Vellore 632014
Tamil Nadu
India

Abstract

A graph G is considered to have a prime labeling when each of its $|V|$ vertices is assigned a unique label from the set $\{1, 2, 3, 4, \ldots, |V|\}$, ensuring that the labels of any two connected vertices are coprime. In 1980, Roger Entringer proposed the conjecture that ``All trees have a Prime labeling", which is not settled till today. In spite of many researchers working on prime labeling, conjecture on prime trees is still open. Up to our knowledge, no one has given an algorithm to generate prime trees. In this paper, given an arbitrary tree T, we develop an algorithm to construct a prime labeled tree T' such that T is a subtree of T'. We extend this algorithm for k arbitrary trees T_i, where $i = 1, 2, 3, \ldots, k$ to construct a larger prime labeled tree T' such that each T_i is a subtree of T'. We also develop a log-linear algorithm to construct a larger prime labeled tree T' from k prime trees T_i, where $i = 1, 2, 3, \ldots, k$ such that T_i is a subtree of T'. Further, we prove the correctness of the proposed algorithms and obtain the time complexity of the proposed algorithms.

Subject Classification: *05C78, 05C76.*

Keywords: *Graph labeling, Prime labeling, Prime tree conjecture, Prime labeling algorithm.*

1. Introduction

A graph G can be described as an ordered triple $(V(G), E(G), \gamma(G)$ consisting of a non empty set $V(G)$, a set $E(G)$ of edges, and an incidence function $\gamma(G)$, that associates with each edge of G an unordered pair of vertices of G [1].

† *ORCID-ID:* **0009-0001-3479-852X**
* *ORCID-ID:* **0000-0002-2923-3268**

† *E-mail:* **tharunrajkg1436@gmail.com**
* *E-mail:* **ragukumar2003@gmail.com (Corresponding Author)**

For definitions and terminologies, we refer [1]. Throughout this paper, we consider simple, finite and undirected graphs.

Graph labeling refers to the process of assigning integers to the vertices, edges, or both, under specific constraints. The foundation for many labeling techniques can be traced back to a method proposed by Rosa in 1967 [6]. He identified three types of labelings, namely α-labeling, β-labeling and ρ-labeling. Other labeling techniques which comprises Harmonious labeling, Magic labeling, Anti-Magic labeling, Prime labeling, Vertex prime labeling and Prime cordial labeling started to emerge while Graceful labelings were investigated across numerous classes of graphs. Throughtout this paper, we deal with Prime labeling of trees.

The concept of prime labeling was introduced by Roger Entringer in the year 1980 and was considered by Tout, Dabboucy and Howalla in the paper titled Prime labeling of graphs [8]. A graph G is considered to have a prime labeling when each of its $|V|$ vertices is assigned a unique label from the set $\{1,2,3,4,\dots,|V|\}$, ensuring that the labels of any two connected vertices are coprime.

In 1980, Roger Entringer proposed the conjecture that " All trees have a Prime labeling" , which is not settled till today. Prime labeling has been verified for numerous tree structures such as banana tree, binomial tree, and palm trees, as well as olive trees, spiders, complete binary trees, stars, and paths (see [7], [4], [8], [2], [5]). For an extensive survey on graph labeling, one can refer to the dynamic survey by Gallian [3]. In spite of many researchers working on prime labeling, conjecture on prime trees is still open. Up to our knowledge, no one has given an algorithm to generate prime trees. This motivates us to develop an algorithm to construct larger prime tree given an arbitrary tree. We develop an algorithm to construct a prime labeled tree T' such that T is a subtree of T'. We extend this algorithm for k arbitrary trees T_i, where $i = 1,2,3,\dots,k$ to construct a larger prime labeled tree T' such that T_i is a subtree of T'. We also develop a log-linear algorithm to construct a larger prime labeled tree T' from k prime trees T_i, where $i = 1,2,3,\dots,k$ such that T_i is a subtree of T'. Our proposed algorithms ensures that the original input tree T is a subtree of the newly generated prime tree T'.

Contributions in this paper

- Given an arbitrary tree T, we develop an algorithm to construct prime labeled tree T'.
- Given k arbitrary trees T_i where $1 \leq i \leq k$, we develop an algorithm to construct larger prime labeled tree T'.
- Given k prime trees T_i where $1 \leq i \leq k$, we develop a log-linear algorithm to construct a larger prime labeled tree T'.

2. Main results

In this section, we develop an algorithm that takes any random tree T as input, and the following algorithm is executed on tree T, resulting in output T'. This algorithm ensures that T is a subtree of T'.

2.1 *Prime Labeling Algorithm*

Input:

Consider a tree $T = (V, E)$, where the vertex set is given by $V = \{v_1, v_2, \dots, v_n\}$, and E denotes the set of edges.

Output:

A prime tree T' which admits prime labeling.

Step 1: Initialization

Bipartite the vertex set V of T into two disjoint sets say (V_1, V_2) that is, $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \phi$. Ensure that $|V_1| \geq |V_2|$.

Step 2: Assignment of labels to the vertices in set V

Consider the vertex set V with n vertices, say $\{v_1, v_2, v_3, \dots, v_n\}$

Step 2.1: Assignment of labels to the vertices in set V_1

For i, $1 \leq i \leq |V_1|$, define $l(v_i) = i$.

Step 2.2: Assignment of labels to the vertices in set V_2

Start with $P = |V_1| + 1$,
For each vertex $y \in V_2$,
    If P is not prime:
        Add a pendant vertex z to a vertex $v_1 \in V_1$ with $l(v_1) = 1$,
    else
        Assign $l(y) = P$,
    Increment P,
End For

Next, we present some observations on the Prime Labeling Algorithm, that assigns unique labels to the vertices of a given arbitrary input tree T.

Some Observations on Prime Labeling Algorithm

Observation 1: It can be observed from the algorithm (2.1), that the labels assigned to neighboring vertices are relatively prime.

Since the algorithm assigns consecutive integers to one set of the bipartition (say (V_1)) and generates coprime integers for the other set (say V_2), each pair of adjacent vertices (one from (V_1) and one from (V_2)) will have labels that are coprime by construction. This guarantees that no two adjacent vertices share a common divisor other than 1.

Observation 2: The algorithm (2.1) is efficient and tractable.

Partitioning the vertex set V into two disjoint subsets (V_1, V_2) to form a bipartition can be performed in $O(n+m)$ time, where n and m denote the number of vertices and edges, respectively. Assigning labels to V_1 and generating coprimes for V_2 are efficient and manageable even for large trees, making the algorithm tractable for practical use.

Observation 3: For a tree T consisting of n vertices, the algorithm (2.1) operates with a time complexity of $O(n^2)$

Time complexity of the algorithm (2.1) can be derived from the complexities of its three main steps, bipartition the tree, assign labels to set V_1, assign labels to set V_2. From the observation 2 the time complexity for bipartition the tree is $O(n)$, Assign consecutive integers starting from 1 to the vertices in V_1. This involves iterating through the vertices in V_1, resulting in a time complexity of $O(|V_1|)$. Since $|V_1| \leq n$, this step has a time complexity of $O(n)$. Assign the prime numbers to the set V_2, since $|Y| \leq n$ in the worst case, this step requires $O(n^2)$ time. Therefore, the time complexity of the proposed algorithm is $O(n^2)$.

Correctness of Prime Labeling Algorithm

The correctness of the prime labeling algorithm is established in the following section, where Theorem 1 helps to prove this result.

Theorem 1: *Given a tree $T = (V, E)$ with bipartite sets V_1 and V_2, the prime labeling algorithm produces an output tree $T' = (V', E')$ such that T is a subtree of T'.*

Proof: For a given tree T, the prime labeling algorithm produces a corresponding labeled tree T'. Let the tree T be defined by the sets V, E, V_1, V_2, where V_1 and V_2 are disjoint subsets forming a bipartition of V and the sets V', E', V'_1, V'_2 that are generated during the execution of the prime labeling algorithm. Consider the vertex set V with n vertices, say $\{v_1, v_2, v_3, \dots, v_n\}$. Let $|V_1| = m$ and $|V_2| = n - m$, where $n = |V|$. Vertices in V_1 are labeled with distinct integers from 1 to m. This forms the set of labels $L_{V_1} = \{1,2,3, \dots, m\}$. Starting from $m+1$, each subsequent prime number is used to label vertices in V_2. This forms the set of labels $L_{V_2} = \{P_1, P_2, P_3, \dots, P_{n-m}\}$. When encountering a composite number during the iteration in step 2.2, a new pendant vertex is added to a vertex in V_1 whose vertex label is 1. We denote the set of all new vertices by Z, and C be the set of composite numbers used to label these vertices. The vertex set of the resulting tree T' is given by $V' = V \cup Z$. The edge set of the resulting tree T' is given by $E' = E \cup \{(1, z) | z \in Z\}$. The vertices of T retain their original labels and positions in T', ensuring T is a subset of T'. For every $(u, v) \in E$, the gcd of labels is 1 because, vertices in V_1 are labeled with integers, and vertices in V_2 are labeled with primes. T remains a subtree within T' because no vertices or edges from T are removed. Thus, T is indeed a subtree of T'.

For the arbitrary tree in Fig.1, we obtained tree T' in Fig.2 as an output of prime labeling algorithm.

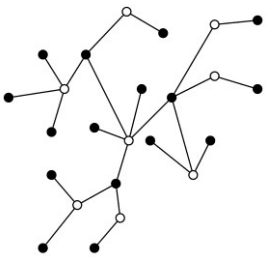

Figure 1
An arbitrary tree T with 27 vertices.

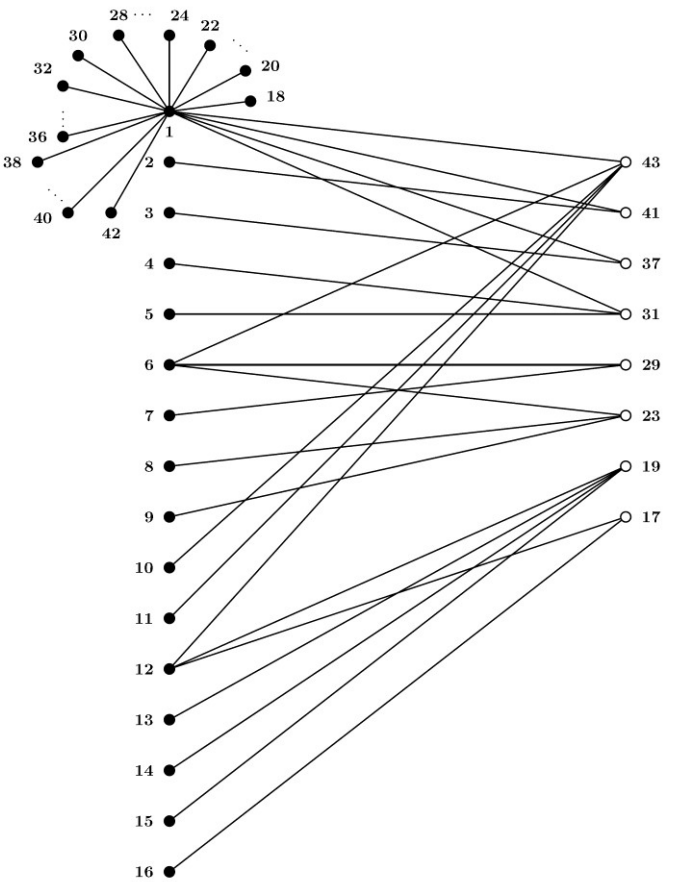

Figure 2
Construction of prime tree T′ obtained after applying the prime labeling algorithm.

Next we present algorithm for generation of larger prime tree from various arbitrary trees, that assigns unique labels to the vertices of a given arbitrary input trees T_i.

2.2 *Algorithm for Generation of Larger Prime Tree from Various Arbitrary Trees*

Input:

Let $T_i = (V_i, E_i)$, $\forall i = 1,2,3, \dots, k$ trees with vertex set V_i and edge set E_i.

Output:

A Prime tree T' which admits prime labeling.

Step 1: Initialization-Bipartite Partitioning

For each tree T_i, partition V_i into two disjoint sets X_i and Y_i such that every edge $(u, v) \in E_i$ has $u \in X_i$ and $v \in Y_i$.

Step 1.1: Union of Bipartite sets

Define $V'_1 = \bigcup_{i=1}^{k} X_i$ and $V'_2 = \bigcup_{i=1}^{k} Y_i$, Ensure that $|V'_1| \geq |V'_2|$.

Step 1.2: Adding an edge between V'_1 and V'_2

For each $i = 1,2,3, \dots, k$

select any one vertex $x_i \in X_i$

For each $j = 2,3, \dots, k$

select any one vertex $y_j \in Y_j$

Add an edge between x_i and y_j

Step 2: Assignment of labels to the vertices in set V_i

Consider the vertex set V'_1 with n vertices, say $\{v_1, v_2, v_3, \dots, v_n\}$

Step 2.1: Assignment of labels to the vertices in set V'_1

For $i = 1,2,3, \dots, |V'_1|$

Define $l(v_i) = i$.

Step 2.2: Assignment of labels to the vertices in set V'_2

Start with $P = |V'_1| + 1$,

For each vertex $y \in V'_2$,

If P is not prime:

Add a pendant vertex z to a vertex $v_1 \in V'_1$ with $l(v_1) = 1$,

else

Assign $l(y) = P$,

Increment P,

End For

Next we present some observations on generation of larger prime tree from various arbitrary trees algorithm, that assigns unique labels to the vertices of a given arbitrary input trees T_i.

Some Observations on Algorithm for Generation of Larger Prime Tree from Various Arbitrary Trees

Observation 4: The algorithm (2.2) guarantees that the generated tree T' is a prime tree.

The prime labeling of vertices in T' ensures that for any edge (u, v) in T', the labels $l(u)$ and $l(v)$ are coprime. This is achieved through careful assignment of consecutive integers and prime numbers to the vertices in V_1 and V_2 respectively. By maintaining the coprimeness property in step 2.2 throughout the tree, the algorithm guarantees that T' is a prime tree.

Observation 5: The algorithm (2.2) ensures that each tree T_i is a subtree of the generated prime tree T'.

The algorithm partitions each tree T_i into bipartite sets X_i and Y_i. It then constructs the union sets V_1 and V_2 from these bipartite sets, ensuring $|V_1| \geq |V_2|$. By adding edges between selected vertices from V_1 and V_2, it incorporates the structure of each T_i into the larger tree T', making each T_i a subtree of T'.

Observation 6: Time complexity of the algorithm (2.2) is $O(n^2)$.

Time complexity of the algorithm can be derived from the complexities of its main steps. The time complexity for bipartitioning each tree T_i is $O(|V_i| + |E_i|)$. As the process is repeated across all k trees, the cumulative time complexity is $O(\sum_{i=1}^{k} (|V_i| + |E_i|))$. Defining $V_1 = \bigcup_{i=1}^{k} X_i$ and $V_2 = \bigcup_{i=1}^{k} Y_i$ involves concatenating the sets. The time complexity for this step is $O(k)$. For each $(i = 1,2,3, \dots, k)$, selecting a vertex from each bipartite set and adding an edge takes $O(k)$ time. Assigning consecutive integers starting from 1 to the vertices in V_1 results in a time complexity of $O(|V_1|)$. Assigning prime numbers to the vertices in V_2 involves checking if a number is prime. Given that $|V_2| \leq n$, the worst-case time complexity of this step is $O(n^2)$. Therefore, the time complexity of the proposed algorithm is $O(n^2)$.

Correctness of Algorithm for Generation of Larger Prime Tree from Various Arbitrary Trees

This section establishes the correctness of the algorithm for generation of larger prime tree from various arbitrary trees. Theorem 2 helps to verify the algorithm's correctness for generation of larger prime tree from various arbitrary trees.

Theorem 2: *Given k trees $T_i = (V_i, E_i)$ with bipartite sets X_i and Y_i , $\forall$ $i = 1,2,3, \dots, k$. The algorithm for generation of larger prime tree from various arbitrary trees produces an output tree $T' = (V', E')$ such that $\bigcup_{i=1}^{k} T_i$ is a subtree of T'.*

Proof: For an arbitrary input of k trees $T_i = (V_i, E_i)$, generate the output tree T' using the algorithm for generation of larger prime tree from various trees. Consider the sets X_i and Y_i for each tree T_i and define $V'_1 = \bigcup_{i=1}^{k} X_i$ and

$V'_2 = \bigcup_{i=1}^{k} Y_i$. The sets V'_1 and V'_2 are the union of bipartite sets of all k trees. Now, for each $i = 1,2,3,\dots,k$ select any one vertex $x_i \in X_i$ and for each $j = 2,3,4,\dots,k$ select any one vertex $y_j \in Y_j$. Add an edge between x_i and y_j . Consider the vertex set V'_1 with n vertices, say $\{v_1, v_2, v_3, \dots, v_n\}$. Label the vertices in V'_1 with consecutive integers starting from 1 to $|V'_1|$. Let $P = \{p_1, p_2, \dots, p_{|V'_2|}\}$ be the set of the first $|V'_2|$ prime numbers greater than or equal to $|V'_1| + 1$. Assign these primes to the vertices in V'_2. While generating the primes for V'_2, any composite numbers c encountered are used to label new pendant vertices added to a vertex in V'_1 labeled with 1, for each composite c, add a new vertex z and connect it to a vertex $v_1 \in V'_1$, where $l(v_1) = 1$. Label the new vertex z with c , where $l(z) = c$. The vertex set of the resulting tree T' is $V' = V'_1 \cup V'_2 \cup Z$, where Z denotes the set of additional vertices added as a result of composite numbers. The edge set of the resulting tree T' is $E' = \left(\bigcup_{i=1}^{k} E_i\right) \cup [(x_i, y_j) | x_i \in X_i, y_j \in Y_j] \cup [(v_1, z) | v_1 \in V'_1, z \in Z]$. For any edge (u, v) in the original trees, $u \in V'_1$ and $v \in V'_2$, $l(u)$ is an integer from 1 to $|V'_1|$ and the label assigned to vertex v is a prime number that is not less than $|V'_1| + 1$ The gcd of any integer and a prime greater than it is always 1: $\gcd(l(u), l(v)) = 1$, $\forall (u, v) \in \bigcup_{i=1}^{k} E_i$. Any composite number c used to label a new vertex z connected to a vertex v_1 with $l(v_1) = 1$. The gcd of 1 and any composite number c is also 1: $\gcd(1, c) = 1$. For any edge (v_1, y_j) where y_j is in V'_2 and v_1 is the vertex in V'_1 labeled with 1 , the GCD is 1 , $\gcd(1, l(y_j)) = 1$. Therefore, T' contains $\bigcup_{i=1}^{k} T_i$ as a subtree.

For the arbitrary trees in T_1 in Fig.3 and T_2 in Fig.4, we obtained the larger prime tree T' in Fig.5 as an output of generation of larger prime tree from various arbitrary trees.

Next we present a log-linear algorithm for generation of larger prime tree from various prime trees, that assigns unique labels to the vertices of a given prime trees T_i, where $1 \leq i \leq k$.

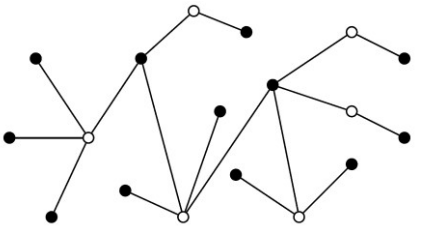

Figure 3
Arbitrary Tree T1 with 18 vertices

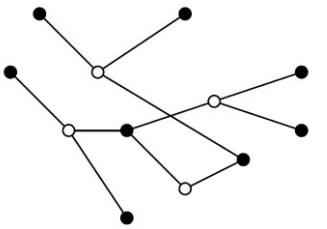

Figure 4
Arbitrary Tree T2 with 12 vertices

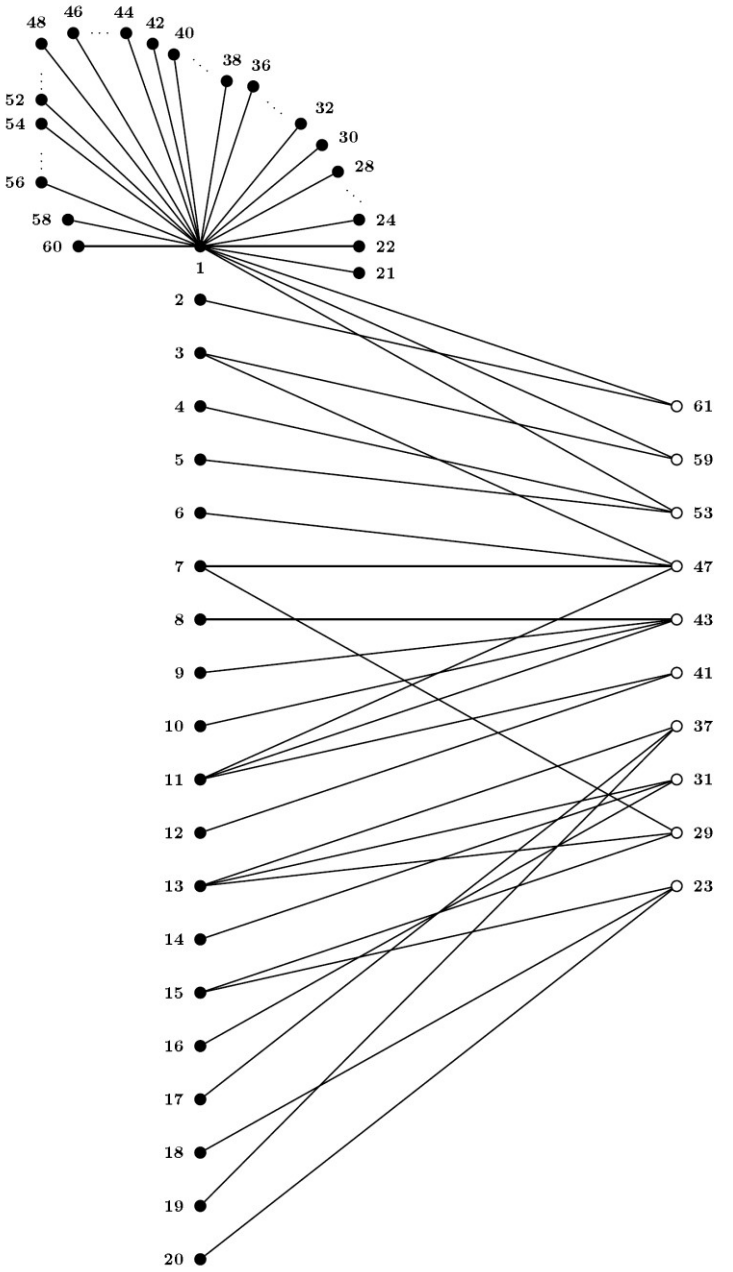

Figure 5
Construction of larger prime tree T′ from arbitrary trees T1 and T2.

2.3 *Algorithm for Generation of Larger Prime Tree from various Prime Trees*

Input:

$T_1, T_2, \dots, T_k$ are k prime trees (except star and double star).

consider $V_1 = \{u_{1,1}, u_{1,2}, u_{1,3}, \dots, u_{1,n_1}\}$, $V_2 = \{u_{2,1}, u_{2,2}, u_{2,3}, \dots, u_{2,n_2}\}$, ..., $V_k = \{u_{k,1}, u_{k,2}, u_{k,3}, \dots, u_{k,n_k}\}$ are k vertex sets associated with the trees $T_1, T_2, \dots, T_k$ respectively, where each V_i consists of n_i vertices, with $1 \leq i \leq k$.

$f_1(V_1), f_2(V_2), \dots, f_k(V_k)$ are k prime labeling functions representing the labels of the vertices for the trees $T_1, T_2, \dots, T_k$ respectively.

Output:

A larger prime tree T' constructed from the given prime trees which admits prime labeling.

Algorithm:

Step 1: Initialization

Let V be the union of vertices from all prime trees, that is, $V = V_1 \cup V_2 \cup \dots \cup V_k$.

Define a function $f: V \to \{1,2, \dots, (n_1 + n_2 + \cdots + n_k), \dots n_r\}$ for labeling vertices in T'.

Step 2: Determine the smallest primes

Find the smallest prime p_1 such that $p_1 > n_1$.

For $2 \leq i \leq k$, find the smallest prime $p_i > p_{i-1} + n_i$

Step 3: Joining the prime trees

For each pair of T_i and T_{i+1}, where $1 \leq i \leq k-1$

Join an edge between $f_i(n_i)$ with $f_{i+1}(u_{i+1,1})$, where $1 \leq i \leq k-1$.

Step 4: Label the vertices

For the tree T_1, label the vertices directly using $f(u_{1,i}) = f_1(u_{1,i})$, where $1 \leq i \leq n_1$

For the tree T_2, label the vertices using $f(u_{2,i}) = f_2(u_{2,i}) + p_1 - 1$, where $1 \leq i \leq n_2$

Continue this process for the trees $T_3, T_4, \dots, T_k$,

$f(u_{j,i}) = f_j(u_{j,i}) + p_{j-1} - 1$, where $3 \leq j \leq k$ and $1 \leq i \leq n_k$.

Step 5: Find gaps and handling composite numbers

For $i = 1$

Identify all composite numbers c between $n_1 < c < p_1$

For each such c, add an edge between c and $f_1(u_{1,1})$

For $i > 1$, where $2 \leq i \leq k-1$

Find the gap as $[n_i + p_{i-1}, p_i - 1]$,

Identify all composite numbers c in this gap

For each such c, add an edge between c and $f_1(u_{1,1})$

Next we present some observations on a log-linear algorithm for generation of larger prime tree from various prime trees, that assigns unique labels to the vertices of a given prime trees.

Some Observations on Algorithm for Generation of Larger Prime Tree from Various Prime Trees

Observation 7: The vertex labels of the resulting tree T', obtained by the algorithm (2.3) are all unique.

The algorithm labels the vertices of each prime tree T_i, $2 \leq i \leq k$ using unique prime numbers p_i such that $p_1 > n_1$, $p_2 > (p_1 + n_2)$, and so on. This ensures that the labels assigned to vertices in different trees do not overlap. Each prime tree is labeled distinctly, and the sequential choice of prime numbers guarantees that each vertex in the combined tree T' has a unique label. This systematic approach prevents any duplication of vertex labels across the trees.

Observation 8: The algorithm (2.3) ensures that the labels of adjacent vertices are coprime.

The algorithm labels the vertices of each prime tree T_i, $2 \leq i \leq k$. Prime tree T_1 retains the original labels starting from 1 using the function f_1. T_2 is labeled starting from $f_2 + p_1 - 1$, T_3 is labeled starting from $f_3 + p_2 - 1$, and so on. The unique prime numbers p_i ensure that the vertex labels corresponding to different prime trees are shifted in such a way that any two adjacent vertices from different trees have coprime labels. This systematic approach ensures that labels of adjacent vertices in the combined tree T' do not share any common divisor other than 1.

Observation 9: The algorithm (2.3) guarantees that the resulting graph T' remains acyclic.

The algorithm connects the last vertex of T_1 to the first vertex of T_2, the last vertex of T_2 to the first vertex of T_3, and so on up to T_k. This step-by-step connection ensures no cycles are introduced since each tree is initially acyclic and the connections are made in a sequential manner, ensuring that the overall structure remains a tree, which is inherently acyclic. Hence, the resulting structure remains a tree, preserving acyclic property by construction.

Observation 10: The algorithm (2.3) correctly handles composite numbers by connecting them to the vertex labeled in T_1.

For each composite number c in the range $n_1 < c < p_1$, an edge is added between c and the vertex labeled 1 in T_1. For $i \geq 2$, composites in the gap $[n_i + p_{i-1}, p_i - 1]$ are similarly connected. This preserves prime labeling and tree structure.

Observation 11: Time complexity of the algorithm (2.3) is $O(n \log n)$.

The total time complexity can be broken down into, Prime Numbers Selection, finding the first $k - 1$ prime numbers larger than the given conditions

can be done in $O(k \log k)$ time, but considering the largest prime needed can be up to $O(n)$, it can involve generating primes up to approximately $n\log n$, thus this step is $O(n \log n)$. Labeling vertices of T_1 with function f_1 takes $O(n_1)$ time. Labeling subsequent trees $T_2, T_3, \ldots, T_k$ involves adjusting the labels, which cumulatively takes $O(n_2 + n_3 + \cdots + n_k) = O(n)$ time. Joining k trees sequentially involves adding $k-1$ edges, which takes $O(k)$ time. Identifying and connecting composite numbers can add additional vertices. For each composite, we need to check and add vertices, which can be up to $O(n)$ in total, giving an upper bound of $O(n \log n)$ for handling composite numbers due to generating primes. Given these steps, the overall time complexity of the algorithm is $O(n \log n)$.

Correctness of Algorithm for Generation of Larger Prime Tree from Various Prime Trees

This section establishes the correctness of the algorithm for generation of larger prime tree from various prime trees. Theorem 3 helps to verify the algorithm's correctness for generation of larger prime tree from various prime trees.

Theorem 3: Given k trees $T_i = (V_i, E_i)$ that are prime trees (except star and double star), the algorithm described above produces an output tree $T' = (V', E')$ such that $\bigcup_{i=1}^{k} T_i$ is a subtree of T', and T' admits prime labeling.

Proof: To verify the algorithmâ€™s correctness, we aim to establish that each prime tree T_i is labeled correctly with distinct integers. $T_1, T_2, \ldots, T_k$ are k prime trees (except star and double star). Consider $V_1 = \{u_{1,1}, u_{1,2}, u_{1,3}, \ldots, u_{1,n_1}\}$, $V_2 = \{u_{2,1}, u_{2,2}, u_{2,3}, \ldots, u_{2,n_2}\}$, ... $V_k = \{u_{k,1}, u_{k,2}, u_{k,3}, \ldots, u_{k,n_k}\}$ are k vertex sets associated with the trees $T_1, T_2, \ldots, T_k$ respectively, where V_i contains n_i vertices for $1 \le i \le k$. $f_1(V_1)$, $f_2(V_2), \ldots, f_k(V_k)$ are k functions representing the labels of the vertices for the trees $T_1, T_2, \ldots, T_k$ respectively. For each prime tree T_i, the prime numbers p_i are chosen such that: $p_1 > n_1, p_2 > (p_1 + n_2), p_3 > (p_2 + n_3), \ldots, p_k > (p_{k-1} + n_k)$. The sequence of prime trees is combined by attaching the end vertex of T_i to the starting vertex of T_{i+1}, where $1 \le i \le k-1$. Specifically join an edge between $f_i(n_i)$ with $f_{i+1}(u_{i+1}, 1)$, where $1 \le i \le k-1$. This ensures that the prime property is maintained as the prime labels are only extended in sequence. Labeling the vertices of each prime tree T_i is labeled as follows, for prime tree T_1, label the vertices directly using $f(u_{1,i}) = f_1(u_{1,i})$, where $1 \le i \le n_1$, for prime tree T_2, label the vertices using $f(u_{2,i}) = f_2(u_{2,i}) + p_1 - 1$, where $1 \le i \le n_2$, continue this process for all prime tree T_k such that $f(u_{j,i}) = f_j(u_{j,i}) + p_{j-1} - 1$, where $3 \le j \le k$ and $1 \le i \le n_k$. Each prime tree labels are distinct and prime relative to the vertices from other trees because of the chosen prime numbers. For each composite number c in the identified ranges, the algorithm adds an edge between c and the vertex labeled 1 in T_1. For $i = 1$, identify all composite numbers c

such that $n_1 < c < p_1$. For each composite number c in this range, add an edge between c and the vertex labeled 1 in T_1. For $i > 1$, where $2 \leq i \leq k-1$ find the gap as $[n_i + p_{i-1}, p_i - 1]$. Identify all composite numbers in this gap. For each composite number c in this range, add an edge between c and the vertex labeled 1 in T_1. This does not violate the prime property as the composite numbers are only connected to the vertex labeled 1, ensuring that the prime labels within the trees T_i are not compromised. The vertex set of the resulting tree T' is $V' = [\bigcup_{i=1}^{k} V_i \cup c | c \text{is composite}]$. The edge set of the resulting tree T' is $E' = [\bigcup_{i=1}^{k} E_i \cup (T_i(n_i), T_{i+1}(u_{i+1,1})) \cup (1, c) | c \text{is composite}]$. For any edge $(u, v) \in E'$, if $u \in T_i$ and $v \in T_{i+1}$, then $\gcd(l(u), l(v)) = 1$. Since $l(v)$ is a prime number and $l(u)$ is either prime or composite such that $\gcd(l(u), l(v)) = 1$. If $u = 1$ and $v = c$ (composite), then $\gcd(l(1), l(c)) = 1$. Therefore, T' contains $\bigcup_{i=1}^{k} T_i$ as a subtree and T' admits prime labeling.

For the prime trees T_1, T_2, T_3 and T_4 shown in Figures 6,7,8 and 9 respectively, we obtained the larger prime tree T' shown in Fig.10 as an output of generation of larger prime tree from various prime trees.

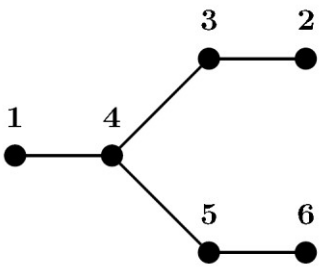

Figure 6
Prime Tree T_1 with 6 vertices.

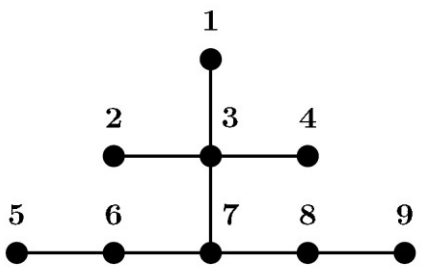

Figure 7
Prime Tree T_2 with 9 vertices.

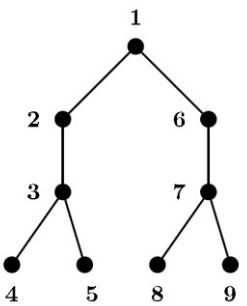

Figure 8
Prime Tree T_3 with 9 vertices.

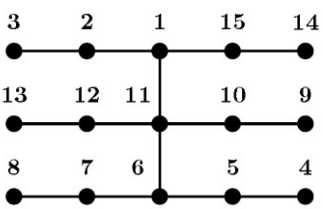

Figure 9
Prime Tree T_4 with 15 vertices.

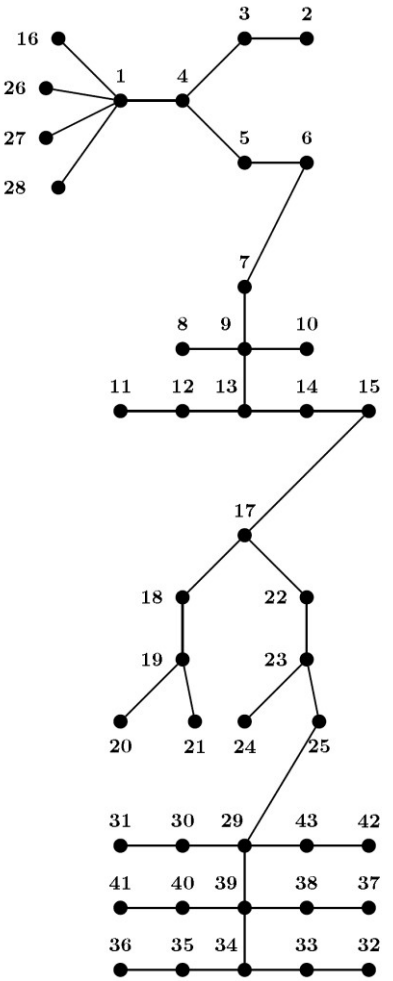

Figure 10
Construction of larger prime tree T′ from prime trees T_1, T_2, T_3 and T_4

3. Conclusion

In this paper, we developed an algorithm to construct a prime labeled tree T' such that T is a subtree of T'. We proved the correctness of the prime labeling algorithm and the time complexity of the prime labeling algorithm is determined to be $O(n^2)$, where n denotes the number of vertices in the input tree T. we extended this algorithm to construct a larger prime labeled tree T' from k arbitrary trees T_i, $i = 1,2, \dots, k$ such that T_i is a subtree of T'. We proved the correctness of the algorithm for generation of larger prime tree from various arbitrary trees and obtain time complexity of the proposed algorithm is $O(n^2)$, where n denotes the number of vertices of k arbitrary trees. We also develop an

algorithm to construct a larger prime labeled tree T' from k prime trees T_i, where $i = 1,2,3, \ldots, k$ such that T_i is a subtree of T'. We proved the correctness of the algorithm for generation of larger prime tree from various prime trees and obtain the overall time complexity of the proposed algorithm is $O(n\log n)$, where n denotes the number of vertices of k prime trees. We found that the time complexities for prime labeling algorithm and algorithm for generation of larger prime tree from various arbitrary trees are $O(n^2)$. our algorithm for generation of larger prime tree from various prime trees is more efficient, with a time complexity of $O(n\log n)$. Further, proposed algorithms in this paper proved that there always exist and can construct a larger prime tree from an arbitrary tree. One can affirmly attack the prime labeling conjecture on how to delete the newly added vertices and edges from the constructed prime tree T' so that it lead to the prime labeling for input tree T.

Acknowledgement

The first author expresses gratitude to Vellore Institute of Technology, Vellore for providing financial support that enabled the author to carry out the Research.

References

[1] J. A. Bondy and U. S. R. Murty, Graph Theory with Applications. London, U.K.: Macmillan (1976).

[2] H. L. Fu and K. C. Huang, "On prime labellings," *Discrete Mathematics*, vol. 127, no. 1–3, pp. 181–186 (1994).

[3] J. A. Gallian, "A dynamic survey of graph labeling," *Electronic Journal of Combinatorics*, vol. 6, no. DS6, pp. 1–623 (2022).

[4] O. Pikhurko, "Trees are almost prime," *Discrete Mathematics*, vol. 307, no. 11–12, pp. 1455–1462 (2007).

[5] L. Robertson and B. Small, "On Newman's conjecture and prime trees," (2009).

[6] A. Rosa, "On certain valuations of the vertices of a graph," in Theory of Graphs (International Symposium, Rome, July 1966). New York, NY, USA: Gordon and Breach, pp. 349–355 (1967).

[7] H. Salmasian, "A result on prime labelings of trees," *Bulletin of the Institute of Combinatorics and its Applications*, vol. 28, pp. 36–38, 2000).

[8] A. Tout, A. N. Dabboucy, and K. Howalla, "Prime labeling of graphs," *National Academy Science Letters*, vol. 11, pp. 365–368 (1982).

Received June, 2025