Embedding an Arbitrary Tree in a Graceful Tree

[1]G. Sethuraman,* [1]P. Ragukumar and [2]Peter J. Slater

[1]Department of Mathematics
Anna University
Chennai 600 025, India
sethu@annauniv.edu
[2]Department of Mathematical Sciences and Department of Computer Science
University of Alabama in Huntsville
Huntsville, AL 35899 USA.
slaterp@uah.edu and pslater@cs.uah.edu

Abstract

A function f is called a graceful labeling of a graph G with m edges if f is an injective function from $V(G)$ to $\{0, 1, 2, \cdots, m\}$ such that when every edge uv is assigned the edge label $|f(u) - f(v)|$, then the resulting edge labels are distinct. A graph which admits a graceful labeling is called a graceful graph. The popular Graceful Tree Conjecture states that every tree is graceful. The Graceful Tree Conjecture remains open for over four decades. Though there are a few general results and techniques on the construction of graceful trees, settling the conjecture seems to be very hard. In this paper, we have introduced a new and different method of constructing graceful trees from a given arbitrary tree. More precisely, we show that every tree can be embedded in a graceful tree with at most km edges, $k < \lceil \frac{m}{4} \rceil$, where m is the number of edges of the given arbitrary tree. Further, we pose a related open problem towards settling the Graceful Tree Conjecture.

Mathematics Subject Classification: 05C78;05C05
Keywords: Graceful Tree;Graceful Tree Conjecture;Graceful Tree Embedding;Graceful Labeling;Graph Labeling

*Corresponding Author

1 Introduction

All the graphs considered in this paper are finite and simple graphs. The terms which are not defined here can be referred from [30]. In 1963, Ringel posed his celebrated conjecture, popularly called Ringel Conjecture [22], which states that, K_{2n+1}, the complete graph on $2n+1$ vertices can be decomposed into $2n+1$ isomorphic copies of a given tree with n edges. In [15], Kotzig independently conjectured the specialized version of the Ringel Conjecture that the complete graph K_{2n+1} can be cyclically decomposed into $2n+1$ copies of a given tree with n edges. In an attempt to solve both the Ringel and Kotzig Conjectures, in 1967, Rosa, in his classical paper [23] introduced hierarchical series of labelings called σ, ρ, β and α labelings as a tool to attack both the Ringel and Kotzig Conjectures. Later, β-labeling was called as graceful labeling by Golomb [11], and now this term is being widely used. A function f is called a graceful labeling of a graph G with m edges, if f is an injective function from $V(G)$ to $\{0, 1, 2, \cdots, m\}$ such that, when every edge uv is assigned the edge label $|f(u) - f(v)|$, then the resulting edge labels are distinct. A graph which admits graceful labeling is called a graceful graph. In [23], Rosa also proved that if T is a graceful tree with n edges, then K_{2n+1} can be cyclically decomposed into $2n+1$ copies of T. This significant theorem led to the Rosa-Kotzig-Ringel Conjecture, popularly called the "Graceful Tree Conjecture": All Trees are Graceful. Over four decades, many interesting and significant results [1–8, 11–16, 18–21, 24–29, 31] were proved to support the Graceful Tree Conjecture, but still it remains open. [9, 10, 17] can be referred for an exhaustive survey on Graceful Tree Conjecture and related results.

In the literature of the graceful tree conjecture, one can observe that researchers have followed three different approaches to prove results supporting the Graceful Tree Conjecture.

- Any tree with at most k vertices is graceful, for some positive integer k. [It was shown that $k = 27$ in [2] and $k = 29$ in [18].

- Special classes of trees (like caterpillars, banana trees etc.,) are graceful. Refer [12–14, 19, 23–25, 29].
- Known graceful trees are combined or modified to produce larger or different graceful trees. [In [16], two graceful trees are combined to get a larger graceful tree. In [21], the diameter 4 graceful trees are modified to obtain all the diameter 5 graceful trees.]

However, in spite of many significant and interesting results established on graceful trees, settling the Graceful Tree Conjecture seems to be very hard. In this paper, we introduce a new and different method of constructing graceful trees from a given arbitrary tree. More precisely, an algorithm is given to construct graceful trees containing a given arbitrary tree as its subtree with atmost km edges, $k < \lceil \frac{m}{4} \rceil$, where m is the number of edges of a given arbitrary tree. This will imply an interesting and significant result that any tree with m edges can be embedded in a graceful tree with atmost km edges, where $k < \lceil \frac{m}{4} \rceil$. In [1], B.D. Acharya et al have observed that every tree can be embedded in a Δ-ary tree, a well known graceful tree (refer [3]), where Δ is the maximum degree of the given tree. In fact, this result motivated us to design an exclusive algorithm to construct a graceful tree from a given arbitrary tree with considerably less number of edges containing the given arbitrary tree as its subtree. Finally, at the end of the paper, we pose a related open problem towards settling the Graceful Tree Conjecture.

2 Main Result

In this section, an algorithm to construct graceful trees from a given arbitrary tree T with m edges is presented.

Labeling Algorithm

Input: Arbitrary tree T with m edges

Step 1: Initialization

Consider a bipartition of the vertex set of T. Let (V_1, V_2) be a bipartition of $V(T)$, and let $|V_1| = p$ and $|V_2| = q$. Let the vertices of V_1 be arranged as $v_0, v_1, v_2, \cdots, v_{p-1}$ such that v_0 is the top most vertex and v_{p-1} is the bottom most vertex. (We refer to this arrangement of vertices of V_1 as "Top to bottom order of V_1"). Now, we arrange all the vertices of V_2 to the right side of the vertices of V_1 in the following order.

Arrange all the adjacent vertices of v_0 on the top. Consider the adjacent vertices of v_1 which are not adjacent to v_0. Then arrange them just below all the adjacent vertices of v_0. Now, consider only those adjacent vertices of v_2 which are neither adjacent to v_0 nor adjacent to v_1, and arrange such adjacent vertices of v_2 just below the adjacent vertices of v_1. Continue this arrangement with the adjacent vertices of $v_3, v_4, v_5, \cdots, v_{p-1}$. Thus, in this arrangement, for any v_j, $0 \leq j \leq p-1$, all adjacent vertices of v_j which are commonly adjacent with any vertex v_i, $i < j$ always appear above the remaining adjacent vertices of v_j. Let the vertices of V_2 arranged as above be described as $u_{q-1}, u_{q-2}, u_{q-3}, \cdots, u_2, u_1, u_0$ such that u_{q-1} is the top most vertex and u_0 is the bottom most vertex. (This ordering of vertices of V_2 is referred to as "Bottom to top ordering of V_2").

Step 2: Vertex Labeling

Step 2.1: Labeling of 0th iteration

Define $l_0(v_i) = i, \text{ for } 0 \leq i \leq p-1$

$l_0(u_i) = (i+1)p, \text{ for } 0 \leq i \leq q-1$

Step 2.2: Labeling of jth iteration for j, $1 \leq j \leq q-1$.

For a fixed j, $1 \leq j \leq q-1$, find $r = min[l_0(N(u_j))]$, where $N(u_j)$ denotes the set of adjacent vertices of u_j in T, and $l_0(N(u_j)) = \{l_0(v) : v \in N(u_j)\}$.

If $r > 0$, then

define $l_j(v_i) = l_0(v_i) = i$, for $0 \leq i \leq p-1$

$$l_j(u_s) = \begin{cases} l_{j-1}(u_s), & \text{if } 0 \leq s < j \\ l_{j-1}(u_s) - r, & \text{if } j \leq s \leq q-1 \end{cases}$$

If $r = 0$, then no more iteration is defined beyond $(j-1)$th iteration. Terminate the process.

Step 3: Edge Labeling

For each j, $0 \leq j \leq q-1$, if the jth iteration is defined in Step 2.2, then for every edge $uv \in E(T)$ define edge label $l'_j(uv) = |l_j(v) - l_j(u)|$.

Notation

For a given input tree T, let θ denote the last iteration that is defined by the Labeling Algorithm for the input tree T. Observe that $\theta \leq q-1$, where $q = |V_2|, (V_1, V_2)$ is the bipartition of the vertex set $V(T)$ of the input tree T that is described in Step 1 of the Labeling Algorithm. For $0 \leq j \leq \theta$, let T_j represents the vertex labeled input tree T, with vertex labels that are defined in the jth iteration of the Labeling Algorithm.

Output of the Labeling Algorithm

For an input tree T, the Labeling Algorithm defines a sequence of iterations, $0th, 1st, 2nd, \cdots, \theta$th iterations and produces the corresponding output as a sequence of vertex labelled trees $T_0, T_1, T_2, \cdots, T_\theta$ (where θ is the last iteration defined by the Labeling Algorithm for the input tree T).

Theorem 1. *The vertex labels of the vertices of each of the output tree T_j, for j, $0 \leq j \leq \theta$, where θ is the last iteration that is defined in the Labeling Algorithm for the input tree T, are all distinct.*

Proof. We prove the theorem by induction on j. When $j = 0$, then by Step 2.1 of the Labeling Algorithm, we have, $l_0(v_i) = i$, for $0 \leq i \leq p-1$ and

$l_0(u_i) = (i+1)p$, $0 \leq i \leq q-1$. It is clear that the labels $l_0(v)$, for all $v \in V(T_0)$ are distinct. We assume that the vertex labels of each output tree T_j that is defined in the jth iteration, for j, $1 \leq j \leq k$, of the Labeling Algorithm are all distinct. That is, the vertex labels $l_j(v)$ for all $v \in V(T)$ are distinct, for $0 \leq j \leq k$. Thus, for each j, $1 \leq j \leq k$, the vertex labels,

$$l_j(v_i) = l_0(v_i) = i, \text{ for } 0 \leq i \leq p-1 \tag{1}$$

$$l_j(u_i) = \begin{cases} l_{j-1}(u_i), & \text{if } 0 \leq i < j \\ l_{j-1}(u_i) - r, & \text{if } j \leq i \leq q-1 \end{cases} \tag{2}$$

where $r = min[l_j(N(u_j))](= min[l_0(N(u_j))])$ are all distinct. Now we prove that the vertex labels of all the vertices of the output tree T_{k+1} that is defined in the $(k+1)$th iteration are all distinct. By Step 2.2 of the Labeling Algorithm, the vertex labels of the tree T_{k+1} are defined as:

$$l_{k+1}(v_i) = l_0(v_i) = i, \text{ for } 0 \leq i \leq p-1 \tag{3}$$

$$l_{k+1}(u_i) = \begin{cases} l_k(u_i), & \text{if } 0 \leq i < k+1 \\ l_k(u_i) - r_1, & \text{if } k+1 \leq i \leq q-1 \end{cases} \tag{4}$$

where $r_1 = min[l_{k+1}(N(u_{k+1}))] = min[l_0(N(u_{k+1}))]$. By inductive assumption, the vertex labels of all the vertices of T_j, $0 \leq j \leq k$ are distinct. From (3), and by the inductive assumption, the vertex labels of the vertices in $V_1(T_{k+1})$ are distinct. From (4), and by the inductive assumption, the vertex labels of the vertices in the set $\{u_i \in V_2(T_{k+1}) : i \leq k\}$ are all distinct. First we claim that $l_{k+1}(u_{k+1}) > l_{k+1}(u_k)$. Suppose not. Then,

$$l_{k+1}(u_{k+1}) \leq l_{k+1}(u_k). \tag{5}$$

From (4), the inequality (5) can be rewritten as $l_k(u_{k+1}) - r_1 \leq l_k(u_k)$, where $r_1 = min[l_{k+1}(N(u_{k+1}))] = min[l_0(N(u_{k+1}))]$. Again, using (2) when $j = k$, the above inequality can be rewritten as $l_{k-1}(u_{k+1}) - r - r_1 \leq l_{k-1}(u_k) - r$, where $r = min[l_k(N(u_k))] = min[l_0(N(u_k))]$. Thus, we have $l_{k-1}(u_{k+1}) - r_1 \leq l_{k-1}(u_k)$. Further, using (2) when $j = k-1$, the inequality can be rewritten as $l_{k-2}(u_{k+1}) - r' - r_1 \leq l_{k-2}(u_k) - r'$, where $r' = min[l_{k-1}(N(u_{k-1}))] =$

$min[l_0(N(u_{k-1}))]$. Similarly if we continue, finally we get $l_0(u_{k+1}) - r_1 \leq l_0(u_k)$. Thus, $l_0(u_{k+1}) - l_0(u_k) \leq r_1$. Then, by the definition of labeling l_0, we have $p \leq r_1$. But $r_1 = min[l_{k+1}(N(u_{k+1}))] = min[l_0(N(u_{k+1}))] < p$. A contradiction. Therefore, $l_{k+1}(u_{k+1}) > l_{k+1}(u_k)$.
Next, we claim that $l_{k+1}(u_{k+2}) > l_{k+1}(u_{k+1})$. Suppose not. Then,

$$l_{k+1}(u_{k+2}) \leq l_{k+1}(u_{k+1}). \tag{6}$$

Using (4), we can write the inequality (6) as

$$l_k(u_{k+2}) - r_1 \leq l_k(u_{k+1}) - r_1 \tag{7}$$

where $r_1 = min[l_{k+1}(N(u_{k+1}))] = min[l_0(N(u_{k+1}))]$. Therefore,

$$l_k(u_{k+2}) \leq l_k(u_{k+1}). \tag{8}$$

Again, by (2) when $j = k$, the inequality (8) can be rewritten as $l_{k-1}(u_{k+2}) - r \leq l_{k-1}(u_{k+1}) - r$, where $r = min[l_j(N(u_j))] = min[l_0(N(u_j))]$. This implies,

$$l_{k-1}(u_{k+2}) \leq l_{k-1}(u_{k+1}). \tag{9}$$

Again, by (2) when $j = k-1$, the inequality (9) can be written as $l_{k-2}(u_{k+2}) - r' \leq l_{k-2}(u_{k+1}) - r'$, where $r' = min[l_{k-1}(N(u_{k-1}))] = min[l_0(N(u_{k-1}))]$. Thus, $l_{k-2}(u_{k+2}) \leq l_{k-2}(u_{k+1})$. Similarly, if we continue, finally we get $l_0(u_{k+2}) \leq l_0(u_{k+1})$. Thus, $l_0(u_{k+2}) - l_0(u_{k+1}) \leq 0$. But, $l_0(u_{k+2}) - l_0(u_{k+1}) = p > 0$. A contradiction. Therefore, $l_{k+1}(u_{k+2}) > l_{k+1}(u_{k+1})$. Similarly, we can prove that $l_{k+1}(u_{h+1}) > l_{k+1}(u_h)$ for any h, $k+1 \leq h \leq q-2$. Therefore, $l_{k+1}(u_{k+1}), l_{k+1}(u_{k+2}), \cdots, l_{k+1}(u_{q-1})$ form a monotonically increasing sequence. Hence, the vertex labels of all the vertices of T_{k+1} are distinct. This completes the induction. □

Theorem 2. *The edge labels of the edges of each of the output tree T_j, for j, $0 \leq j \leq \theta$, where θ is the last iteration that is defined in the Labeling Algorithm are all distinct for the input tree T.*

Proof. We prove the theorem by induction on j. When $j = 0$, the labels of the vertices of T_0 are defined in Step 2.1 as $l_0(v_i) = i$, for $0 \leq i \leq p-1$, and $l_0(u_i) = (i+1)p$, for i, $0 \leq i \leq q-1$, where $p = |V_1|, q = |V_2|$ and (V_1, V_2) is a bipartition of the input tree T that is described in Step 1 of the Labeling Algorithm. Consider any two consecutive vertices u_t, u_{t+1} of V_2, where $0 \leq t \leq q-2$. Let $x = min[l_0(N(u_t))]$ and let $y = max[l_0(N(u_{t+1}))]$. We claim that $l_0(u_t) - x < l_0(u_{t+1}) - y$. Suppose not. Then, $l_0(u_t) - x \geq l_0(u_{t+1}) - y$. By the definition of the labeling l_0, we have, $l_0(u_t) - x \geq l_0(u_t) + p - y$. Thus, $y - x \geq p$. But, since $0 \leq x, y \leq p-1$, we have $y - x < p$. A contradiction. Hence, all the edge labels of the edges in T_0 are distinct. Assume that the edge labels of the edges of each output tree T_j, for j, $0 \leq j \leq k$ are all distinct. We prove that the edge labels of all the edges of the output tree T_{k+1} are distinct. By the definition of l_{k+1} given in Step 2.2 of the Labeling Algorithm and by the inductive assumption, the edge labels of the edges that are incident with vertices u_t, for t, $0 \leq t \leq k$ are all distinct. Therefore, it is enough to prove that the edge labels of the edges that are incident with vertices u_t, for t, $k+1 \leq t \leq q-1$ are distinct. Consider the two consecutive vertices u_t, u_{t+1} of $V_2(T_{k+1})$ in the bottom to top ordering of V_2, where $k \leq t \leq q-2$. Let $a = min[l_{k+1}(N(u_t))]$ and let $b = max[l_{k+1}(N(u_{t+1}))]$. Observe that $l(u_t) - l(v) \leq l(u_t) - a$ for any $v \in N(u_t)$ and $l(u_{t+1}) - l(w) \geq l(u_{t+1}) - b$ for any $w \in N(u_{t+1})$. We claim that $l_{k+1}(u_t) - a < l_{k+1}(u_{t+1}) - b$. Suppose not. Then,

$$l_{k+1}(u_t) - a \geq l_{k+1}(u_{t+1}) - b. \tag{10}$$

Case 1: $t = k$.

Then, (10) becomes $l_{k+1}(u_k) - a \geq l_{k+1}(u_{k+1}) - b$. By the definition of l_{k+1}, we have $l_k(u_k) - a \geq l_k(u_{k+1}) - r_1 - b$, where $r_1 = min[l_{k+1}(N(u_{k+1}))]$. Thus, $l_k(u_k) - l_k(u_{k+1}) \geq a - r_1 - b$. Again, by the definition of l_k, we have $[l_{k-1}(u_k) - r] - [l_{k-1}(u_{k+1}) - r] \geq a - r_1 - b$, where $r = min[l_k(N(u_k))]$. This implies, $l_{k-1}(u_k) - l_{k-1}(u_{k+1}) \geq a - r_1 - b$. Then by the definition of l_{k-1}, we have $[l_{k-2}(u_k) - r'] - [l_{k-2}(u_{k+1}) - r'] \geq a - r_1 - b$, where $r' = min[l_{k-1}(N(u_{k-1}))]$. Therefore, $l_{k-2}(u_k) - l_{k-2}(u_{k+1}) \geq a - r_1 - b$. Similarly, if we continue, finally we get $l_0(u_k) - l_0(u_{k+1}) \geq a - r_1 - b$. That is, $l_0(u_k) - [l_0(u_k) + p] \geq a - r_1 - b$. This implies,

$$r_1 - a \geq p - b. \tag{11}$$

As the vertex u_{k+1} appears above the vertex u_k in the bottom to top ordering of V_2 that is defined in Step 1 of the Labeling Algorithm and since $a = min[l_{k+1}(N(u_k))]$, by Step 1 of the Labeling Algorithm, the vertex u_{k+1} should be either adjacent to a or to some other vertex of V_1 which appears above the vertex labeled a in the top to bottom ordering of V_1. This implies u_{k+1} is either adjacent to a vertex with label a or some other vertex whose label is less than a. Therefore, $r_1 \leq a$.

Case 1.1: $r_1 < a$**.**

Then $p - b \leq r_1 - a < 0$. But, $p > b \geq 0$, $p - b > 0$. A contradiction.

Case 1.2: $r_1 = a$**.**

Then, $a - r_1 = 0$. Therefore, $p - b \leq 0$. That is, $b \geq p$. But $0 \leq b < p$. A contradiction. Hence, $l_{k+1}(u_t) - a < l_{k+1}(u_{t+1}) - b$ is true when $t = k$.

Case 2: $t > k$**.**

Let $t = k + \alpha, 0 < \alpha \leq (q-2) - k$. Then the inequality (10) becomes $l_{k+1}(u_{k+\alpha}) - a \geq l_{k+1}(u_{k+\alpha+1}) - b$. By the definition of l_{k+1}, we have $l_k(u_{k+\alpha}) - r_1 - a \geq l_k(u_{k+\alpha+1}) - r_1 - b$, where $r_1 = min[l_{k+1}(N(u_{k+1}))]$. This implies, $l_k(u_{k+\alpha}) - a \geq l_k(u_{k+\alpha+1}) - b$. Then, by the definition of l_k, we have $l_{k-1}(u_{k+\alpha}) - r - a \geq l_{k-1}(u_{k+\alpha+1}) - r - b$, where $r = min[l_k(N(u_k))]$. Then, $l_{k-1}(u_{k+\alpha}) - a \geq l_{k-1}(u_{k+\alpha+1}) - b$. By the definition of l_{k-1}, we have $l_{k-2}(u_{k+\alpha}) - r' - a \geq l_{k-2}(u_{k+\alpha+1}) - r' - b$, where $r' = min[l_{k-1}(N(u_{k-1}))]$. Thus, $l_{k-2}(u_{k+\alpha}) - a \geq l_{k-2}(u_{k+\alpha+1}) - b$. Similarly, if we continue, finally we get $l_0(u_{k+\alpha}) - a \geq l_0(u_{k+\alpha+1}) - b$. This implies $l_0(u_{k+\alpha}) - a \geq l_0(u_{k+\alpha}) + p - b$. Therefore, $b - a \geq p$. That is, $b - a \geq p$. But since $0 \leq a, b \leq p-1$, $b - a < p$. A contradiction. Thus, $l_{k+1}(u_t) - a < l_{k+1}(u_{t+1}) - b$ for all t, $k+1 \leq t \leq q-1$. Thus $max\{l'_{k+1}(u_t v) : v \in N(u_t)\} < min\{l'_{k+1}(u_{t+1} w) : w \in N(u_{t+1})\}$. As the vertex labels defined in the labeling l_{k+1} are distinct, the edge values in the set $\{l'_{k+1}(u_t v) : v \in N(u_t)\}$ are distinct and the edge values in the set $\{l'_{k+1}(u_t w) : w \in N(u_{t+1})\}$ are distinct. Thus, the edge labels of the edges that are incident to vertices u_t, for $k \leq t \leq q-1$ are distinct. This completes the proof. □

Embedding Algorithm

Input: Any arbitrary tree T

Step 1:

Step 1.1:

Run Labeling Algorithm on input tree T and get the output. Let $T_0, T_1, T_2, \cdots, T_\theta$ be the sequence of vertex labeled trees obtained as output from the Labeling Algorithm for the input tree T, where θ is the last iteration of the Labeling Algorithm for the input tree T.

Step 1.2:

For each tree T_j, $0 \leq j \leq \theta$, define
Vertex Label Set
$V_j = V(T_j) = \{0, 1, 2, \cdots, p-1, p, \alpha_1, \alpha_2, \cdots, \alpha_{q-1} = M_j\}$,

where the elements of V_j are the vertex labels of the vertices of the input tree T that is defined in the jth iteration of the Labeling Algorithm,

Edge Label Set $E_j = \{l'_j(e_1), l'_j(e_2), \cdots, l'_j(e_m)\}$,

where $l'_j(e_i)$, is the edge labels of the edge e_i, for $1 \leq i \leq m$ of T defined in the jth iteration of the Labeling Algorithm,

All label set $X_j = \{0, 1, 2, \cdots, M_j\}$,
Common label set $I_j = V_j \cap E_j$,
Exclusive vertex label set $\hat{V}_j = (V_j - \{0\}) - I_j$,
Exclusive edge label set $\hat{E}_j = E_j - I_j$ and
Missing vertex label set $\hat{X}_j = X_j - V_j$.

Step 2:

Initiate $T_j^* \leftarrow T_j$,
$V(T_j^*) \leftarrow V(T_j)$,
$E(T_j^*) \leftarrow E(T_j)$.
While $\hat{X}_j \neq \phi$, find $min\hat{X}_j = c$.

Step 3:

If $c \notin \hat{E}_j$, then consider a new vertex with label c and add a new edge between the vertex labeled 0 and the new vertex with label c to T_j^*.
Update $T_j^* \leftarrow T_j^* + (0, c)$,
$V(T_j^*) \leftarrow V(T_j^*) \cup \{c\}$,
$E(T_j^*) \leftarrow E(T_j^*) \cup \{(0, c)\}$.
Delete c from $\hat{X}_j$ and go to Step 2.

Step 4:

If $c \in \hat{E}_j$, then find $min\hat{V}_j = d$ and find $\beta = c - d$. Consider a new vertex with label c and add a new edge between the vertex labeled β and the new vertex labeled c to T_j^*.
Update $T_j^* \leftarrow T_j^* + (\beta, c)$,
$V(T_j^*) \leftarrow V(T_j^*) \cup \{c\}$,
$E(T_j^*) \leftarrow E(T_j^*) \cup \{(\beta, c)\}$.
Delete c from $\hat{X}_j$ and delete d from $\hat{V}_j$ and go to Step 2.

Observation: If $\hat{X}_j \neq \phi$, then the Embedding Algorithm executes Step 2 and $min\hat{X}_j = c$ is found. This means that there are c vertices in the current tree T_j^* with labels $0, 1, 2, \cdots, c-1$. Thus, after the execution of Step 3 or Step 4, the latest updated tree T_j^* will have $c+1$ vertices with vertex labels $0, 1, 2, \cdots, c-1, c$.

Lemma 1. *The β defined in Step 4 of the Embedding Algorithm is always a positive integer, and it exists as the vertex label of a vertex of the current tree T_j^* that is being used in that execution of Step 4.*

Proof. Step 4 of the Embedding Algorithm is executed when $\hat{X}_j \neq \phi$. Further $c = min\hat{X}_j$, $d = min\hat{V}_j$ and $\beta = c - d$ are found in Step 4. We claim that β is a positive integer. That is, we claim that $c > d$. Suppose not. Then $c \leq d$. Since $c \in \hat{X}_j$, $d \in \hat{V}_j$, and $\hat{X}_j \cap \hat{V}_j = \phi$, $c \neq d$. Thus $c < d$. Since $min\hat{X}_j > p$, where $p = |V_1|$, and (V_1, V_2) is the bipartition of the vertex set of the input tree T, we have $c > p$. This implies that $d > p$. Since d is the minimum over $\hat{V}_j$, any label less than d cannot exist in $\hat{V}_j$. By Step 2 of the Labeling Algorithm, any vertex of T_j whose label is greater than or equal to p must only appear in the right side partition V_2 of T. This means that T_j has only one vertex on the left side partition V_1 of T and that should have been labeled with 0, and all the other remaining vertices must appear on the right side partition V_2 of T. Since T is a tree, the vertex labeled 0 (since $0 \in V_j^*$) should be adjacent to all the vertices of V_2. Thus, T_j must be a star of size m. Then, by the Labeling Algorithm the star T_j should have been labeled as shown in Figure 1. Thus, $V(T_j) = \{0, 1, 2, \cdots, m\}$ and $E(T_j) = \{1, 2, 3, \cdots, m\}$.

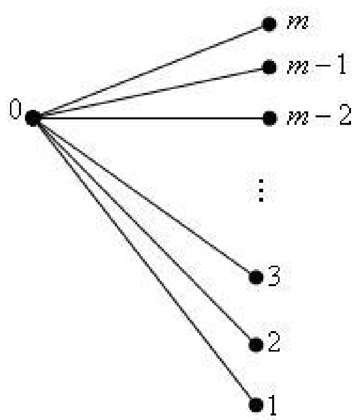

Figure 1: Labeled Star with the labels defined by the Labeling Algorithm.

Hence $X_j = \{0, 1, 2, \cdots, m\}$, and $I_j = \{1, 2, 3, \cdots, m\}$. Therefore, $\hat{X}_j = X_j - V_j = \phi$. But, we have $\hat{X}_j \neq \phi$. A contradiction. This implies $c > d$. Hence β is a positive integer. Since $c = min\hat{X}_j$, the current tree T_j^* should contain all the vertex labels $0, 1, 2, \cdots, c - 1$. Thus, as $\beta = (c - d) < c$, β must be a label of a vertex in that current tree T_j^*. □

Theorem 3. *Let $T_0^*, T_1^*, T_2^*, \cdots, T_\theta^*$ be the output trees of the Embedding Algorithm, where θ is the last iteration executed for the input tree T by the Labeling Algorithm. Then each tree T_j^*, for j, $0 \leq j \leq \theta$ is graceful and contains the input arbitrary tree T as its subtree.*

Proof. By Step 1.2 of the Embedding Algorithm, we have $\hat{X}_j = X_j - V_j$. Then, $X_j = \hat{X}_j \cup V_j$. Since $\hat{E}_j \subset \hat{X}_j$, we can write $X_j = \hat{X}_j \cup V_j = ((\hat{X}_j - \hat{E}_j) \cup \hat{E}_j) \cup V_j$. Observe that $\hat{E}_j \cap V_j = \phi$, $\hat{E}_j \cap (\hat{X}_j - \hat{E}_j) = \phi$ and $V_j \cap (\hat{X}_j - \hat{E}_j) = \phi$. Thus, the sets $(\hat{X}_j - \hat{E}_j), \hat{E}_j$ and V_j are mutually disjoint. Note that V_j consists of all the vertex labels of T_j. $\hat{E}_j$ consists of the edge labels of T_j that are not vertex labels of T_j. $\hat{X}_j - \hat{E}_j$ consists of the members of X_j which are neither the vertex labels of T_j nor the edge labels of T_j. Consider $c = min\hat{X}_j$, obtained by an(any) execution of Step 2 of the Embedding Algorithm. If $c \notin \hat{E}_j$, then by Step 3 of the Embedding Algorithm, the vertex label c is obtained in the updated tree T_j^* by adding the new edge $(0, c)$ to the current tree T_j^*. Also c is removed from $\hat{X}_j$. Since c was removed from $\hat{X}_j$, the vertex label c will never be obtained again.
If $c \in \hat{E}_j$, then by Step 4 of the Embedding Algorithm, the vertex label c is obtained in the updated tree T_j^* by adding the new edge (β, c) in the current tree T_j^* where $\beta = c - d$, and $d = min\hat{V}_j$. Further, c is removed from $\hat{X}_j$ and d is also removed from $\hat{V}_j$. Since c is removed from $\hat{X}_j$ and d is also removed from $\hat{V}_j$, the vertex label c will never be obtained again.
Thus, after executing Step 3 of the Embedding Algorithm $|\hat{X}_j - \hat{E}_j|$ times and Step 4 of the Embedding Algorithm $|\hat{E}_j|$ times, T_j^* contains all the vertex labels $0, 1, 2, \cdots, M_j$. Observe that all the vertex labels obtained from the Embedding Algorithm are distinct and belong to $X_j - V_j$. By Theorem 1, all the vertex labels of T_j are also distinct. Thus for each j, $0 \leq j \leq \theta$, vertex labels of all the vertices of T_j^* are distinct, and the final updated tree T_j^* has $M_j + 1$ vertices with vertex set $V(T_j^*) = \{0, 1, 2, \cdots, M_j\}$. (where a vertex of T_j^* is referred by its corresponding label)

We can write the set $X_j - \{0\} = \hat{X}_j \cup (V_j - \{0\}) = (\hat{X}_j - \hat{E}_j) \cup E_j \cup \hat{V}_j$. Observe that the sets $(\hat{X}_j - \hat{E}_j), \hat{E}_j, \hat{V}_j$ and I_j are mutually disjoint. The

elements in $\hat{E}_j$ and I_j are already existing as edge labels in T_j. Consider, $min\hat{X}_j = c$, obtained at an (any) execution of Step 2 of the Embedding Algorithm. If $c \notin \hat{E}_j$, then by Step 3 of the Embedding Algorithm, the edge label c is obtained in the updated tree T_j^* by adding the new edge $(0, c)$ to the current tree T_j^* and c is removed from $\hat{X}_j$. Since c was removed from $\hat{X}_j$, the edge label c will never be obtained again. If $c \in \hat{E}_j$ is found in an execution of Step 4 of the Embedding Algorithm, then $d = min\hat{V}_j$ is found in that execution, and the edge label d is obtained in the updated tree T_j^* from the new edge (β, c) which was added to the current tree T_j^*, where $\beta = c - d$, and d is removed from $\hat{X}_j$. Also d is removed from $\hat{V}_j$. Since c is removed from $\hat{X}_j$ and d is removed from $\hat{V}_j$ the edge label d will never be obtained again.
Thus, after executing Step 3 of the Embedding Algorithm $|\hat{X}_j - \hat{E}_j|$ times and Step 4 of the Embedding Algorithm $|\hat{V}_j| (= |\hat{E}_j|)$ times in the final updated tree T_j^*, the edge labels belonging to $(X_j - \{0\}) - E_j$ are all obtained as distinct edge labels. As T_j^* was initiated with m edges having distinct edge labels belonging to the set E_j, the final updated tree T_j^* has distinct edge labels $1, 2, \cdots, M_j$ for its M_j edges. Thus, the final updated tree T_j^* is graceful. □

Remark: From Theorem 3, we observe that the Embedding Algorithm takes arbitrary tree T with m edges as its input and produces a sequence of output trees $T_0^*, T_1^*, \cdots, T_\theta^*$, such that each of T_j^*, $0 \leq j \leq \theta$ is graceful, where θ is the last iteration of the Labeling Algorithm. It is clear that, the output tree T_0^* of the Embedding Algorithm has $M_0 = |V_1||V_2|$ edges, where (V_1, V_2) is the bipartition of the vertex set of the input tree T considered in the Labeling Algorithm. It follows that $M_0 \leq \lceil \frac{m^2}{4} \rceil$. Then, the output tree T_1^* of the Embedding Algorithm has $M_1 = M_0 - r_1$ edges, where $r_1 = min[l_0(N(u_1))]$. In general, for $1 \leq j \leq \theta$, the output tree of the Embedding Algorithm T_j^* has M_j edges, where $M_j = M_0 - r_1 - r_2 - \cdots - r_j$, where $r_i = min[l_0(N(u_k))]$ for $1 \leq i \leq j$, where l_0 is the labeling defined in Step 1 of the Labeling Algorithm. Thus, T_θ^* has the least possible number of edges, $M_\theta = M_0 - \sum_{i=1}^{\theta} r_i$. It is easy to see that $M_\theta \leq km$, where $k < \lceil \frac{m}{4} \rceil$. It could be observed that the complexity of the Embedding Algorithm is $O(m^2)$, where m is the size of the input tree T.

3 Discussion

For a given input arbitrary tree T with m edges, the Embedding Algorithm will construct a sequence of graceful trees $T_0^*, T_1^*, T_2^*, \cdots, T_\theta^*$as output, where θ is the last iteration of the Labeling Algorithm for the input tree T, $\theta \leq |V_2|$ and (V_1, V_2) is a bipartition of the vertex set of T considered in the Labeling Algorithm. Observe that T_θ^* is the graceful tree having the least possible number of edges in the output sequence of graceful trees $T_0^*, T_1^*, T_2^*, \cdots, T_\theta^*$ and having the number of edges $M_\theta < km$, where $k < \lceil \frac{m}{4} \rceil$. Observe that those $M_\theta - m$ additional edges are added to the input tree T sequentially one by one by the Embedding Algorithm to obtain the output graceful tree T_θ^*. Thus it is tempting to ask the question,

> *If G is a graceful tree and v is any one degree vertex of G, is it true that $G - v$ is graceful?*

If this question is answered affirmatively, then those additional edges of T introduced for constructing the graceful tree T_θ^* by the Embedding Algorithm could be plugged out in some order so that the given arbitrary tree T becomes graceful. This would imply that Graceful Tree Conjecture is true.

4 Illustrative Examples

Input tree with 26 edges is given in Figure 2.

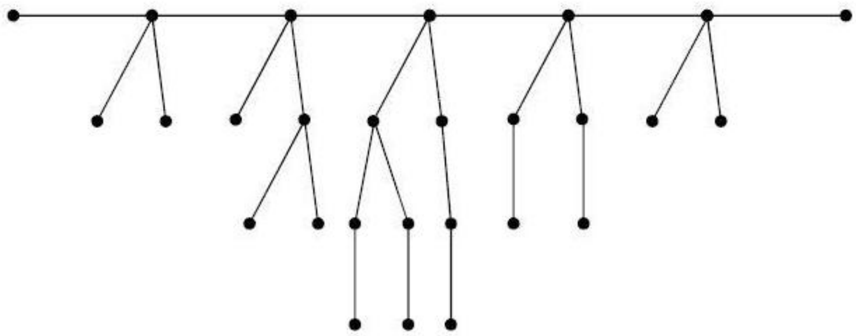

Figure 2: Input tree T

The bipartition of the input tree T defined by Step 1 of the Labeling Algorithm is given in Figure 3 and the labeled tree T_0 defined from the input tree T by Step 2.1 of the Labeling Algorithm is given in Figure 4.

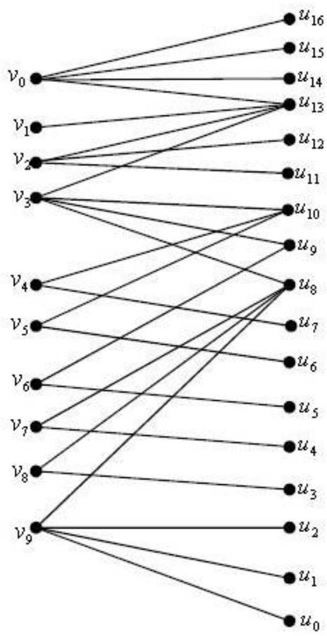

Figure 3: Bipartition of vertices of Input tree T.

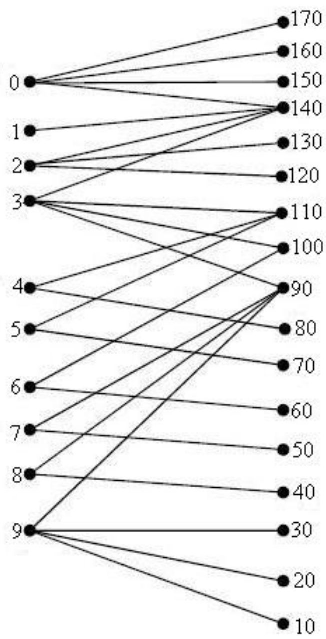

Figure 4: Labeled tree T_0

For the labeled tree T_0, the sets $V_0, E_0, X_0, I_0, \hat{V_0}, \hat{E_0}, \hat{X_0}$ defined by Step 1.2 of the Embedding Algorithm are given below.

$$V_0 = \{0, 1, 2, \cdots, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170\}$$

$$E_0 = \{1, 11, 21, 32, 43, 54, 65, 76, 81, 82, 83, 87, 94, 97, 105, 106, 107, 118, 128, 137, 138, 139, 140, 150, 160, 170\}$$

$$X_0 = \{0, 1, 2, \cdots, 170\}$$

$$I_0 = \{1, 140, 150, 160, 170\}$$

$$\hat{V_0} = \{2, 3, \cdots, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130\}$$

$$\hat{E_0} = \{11, 21, 32, 43, 54, 65, 76, 81, 82, 83, 87, 94, 97, 105, 106, 107, 118, 128, 137, 138, 139\}$$

$$\hat{X_0} = \{11, \cdots, 19, 21, \cdots, 29, 31, \cdots, 39, 41, \cdots, 49, 51, \cdots, 59, 61, \cdots, 69, 71, \cdots, 79, 81, \cdots, 89, 91, \cdots, 99, 101, \cdots, 109, 111, \cdots, 119, 121, \cdots, 129, 131, \cdots, 139, 141, \cdots, 149, 151, \cdots, 159, 161, \cdots, 169\}$$

As $\hat{X_0} \neq \phi$, Step 2 finds $min\hat{X_0} = 11$. Since $11 \in \hat{E_0}$, Step 4 finds $min\hat{V_0} = 2$ and $\beta = 11 - 2 = 9$. Thus, Step 4 adds a new vertex with label 11 and it joins with the vertex labeled 9. Thus the vertex label 11 is obtained and the edge label 2 is also obtained. This new addition of the edge is shown in Figure 5.

Since $\hat{X_0} \neq \phi$, Step 2 of the Embedding Algorithm is executed again. Therefore, $min\hat{X_0} = 12$ is found. Since $12 \notin \hat{E_0}$, Step 3 of the Embedding Algorithm is executed. Thus, the new vertex with label 12 is added to T_0 by adding a new edge between the vertex labeled 0 and the vertex labeled 12. This addition of new edge is shown in Figure 6.

As $\hat{X_0} \neq \phi$ again, Step 2 of the Embedding Algorithm is executed again. Therefore, $min\hat{X_0} = 13$ is found. Since $13 \notin \hat{E_0}$, Step 3 of the Embedding Algorithm is executed. Thus, the new vertex with label 13 is added to T_0 by adding a new edge between the vertex labeled 0 and the vertex labeled 13. This addition of new edge is shown in Figure 7.

Similarly, the vertices with vertex labels $14, 15, 16, 17, 18, 19$ are all added to tree T_0 by making them adjacent to the vertex label 0 by using Step 3

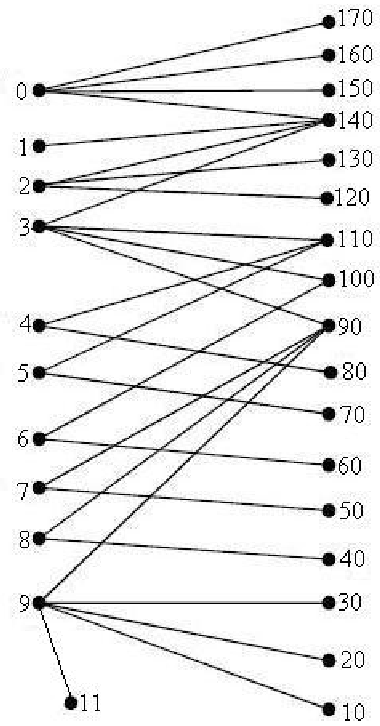

Figure 5: Addition of Vertex 11 to Tree T_0

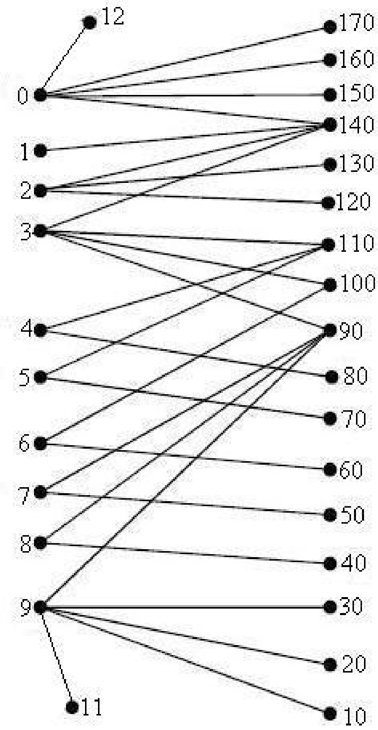

Figure 6: Addition of Vertex 12 to Tree T_0

repeatedly. Tree T_0 with these new edges with labels 14,15,16,17,18 and 19 is shown in Figure 8.

As $\hat{X}_0 \neq \phi$, Step 2 of the Embedding Algorithm is executed again. Thus, $min\hat{X}_0 = 21$ is found and since $21 \in \hat{E}_0$, consequently Step 4 of the Embedding Algorithm is executed, thus, $min\hat{V}_0 = 3$ and $\beta = 21 - 3 = 18$ are found. Hence, the new vertex with label 21 is added to T_0 by adding a new edge between the vertex labeled 21 and the vertex labeled 18. Figure 9 illustrates this new addition of edge. Figure 10 shows the output graceful tree T_0^* of the Embedding Algorithm having the input tree T as its subtree.

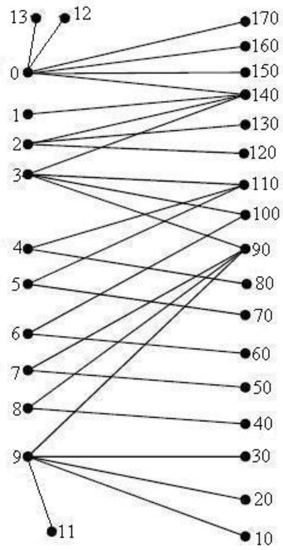

Figure 7: Addition of Vertex 13 to Tree T_0

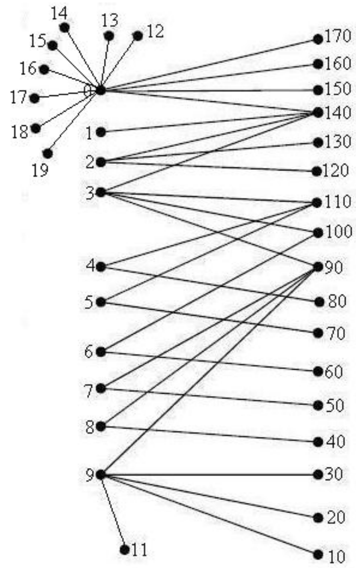

Figure 8: Addition of Vertices 14,15,16,17,18 and 19 to Tree T_0

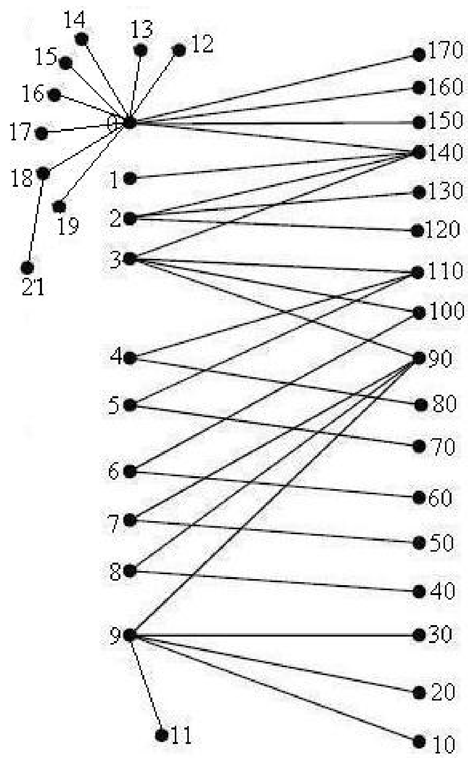

Figure 9: Addition of Vertex 21 to Tree T_0

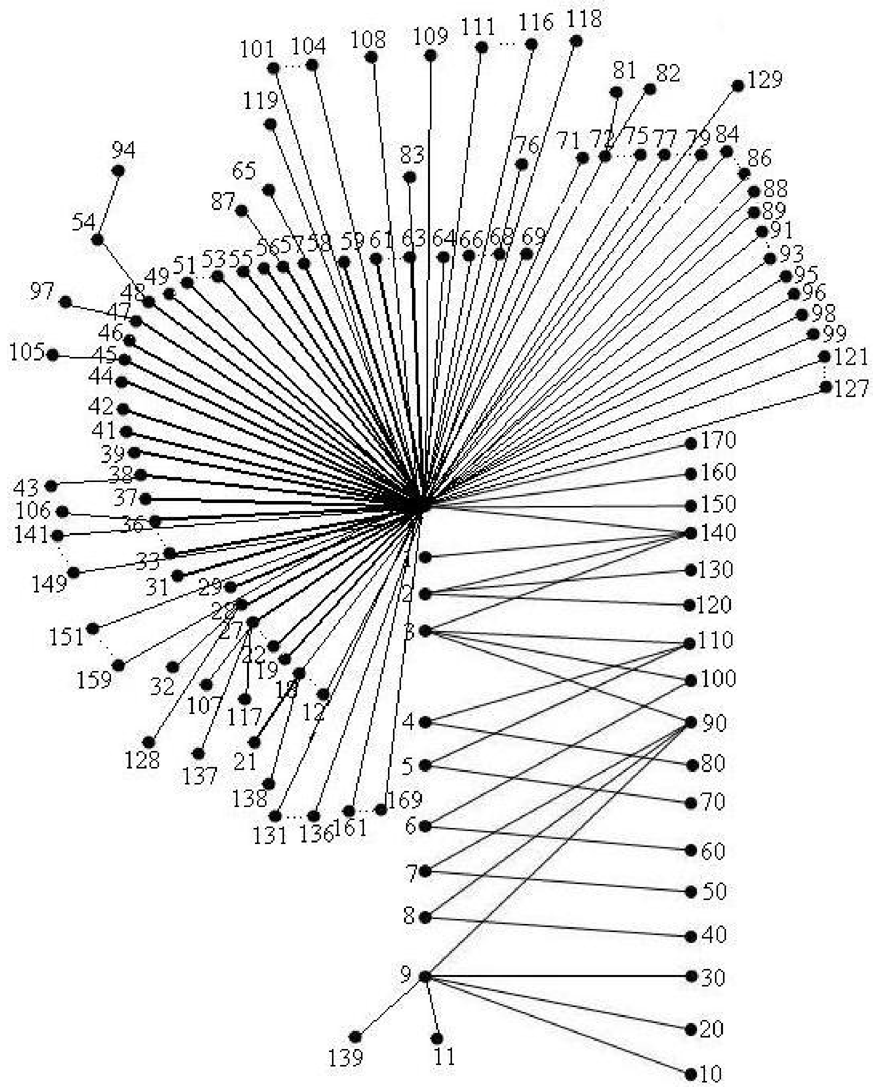

Figure 10: Graceful Tree T_0^* with 170 edges

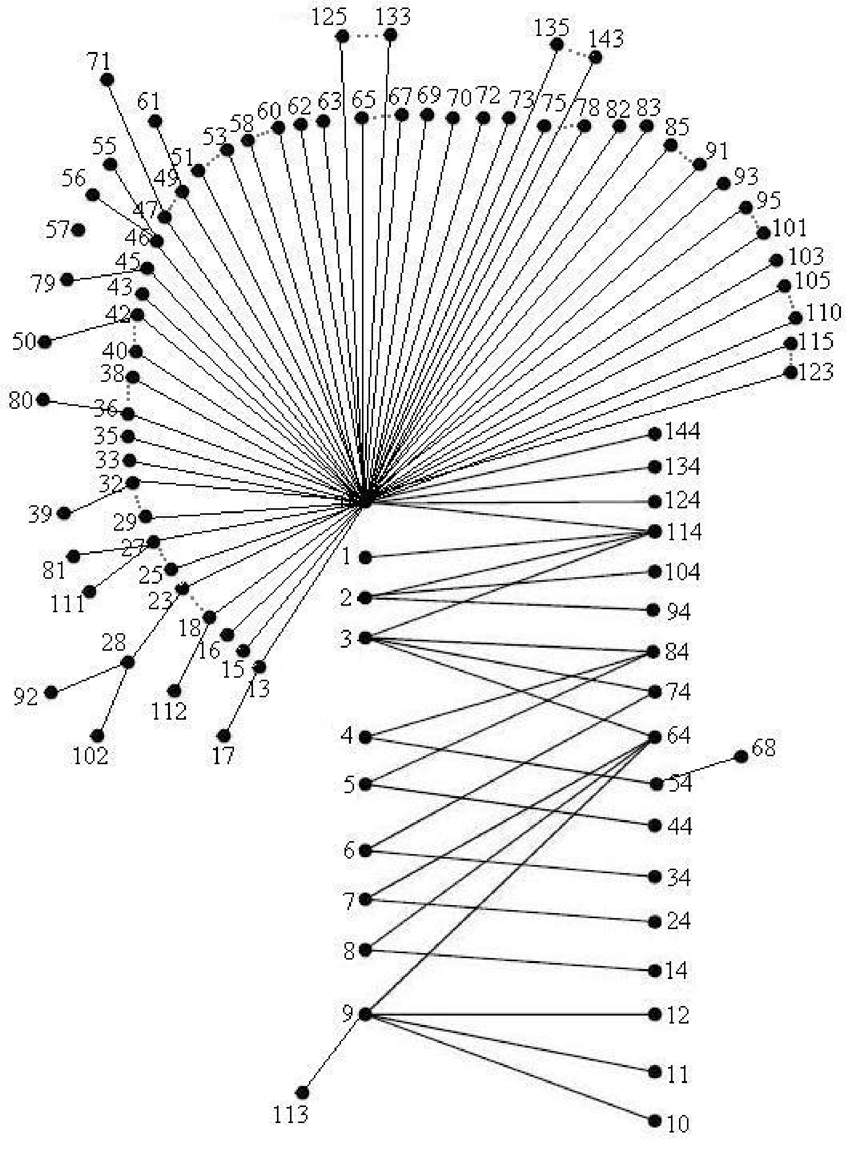

Figure 11: Graceful Tree T_3^* with 144 edges

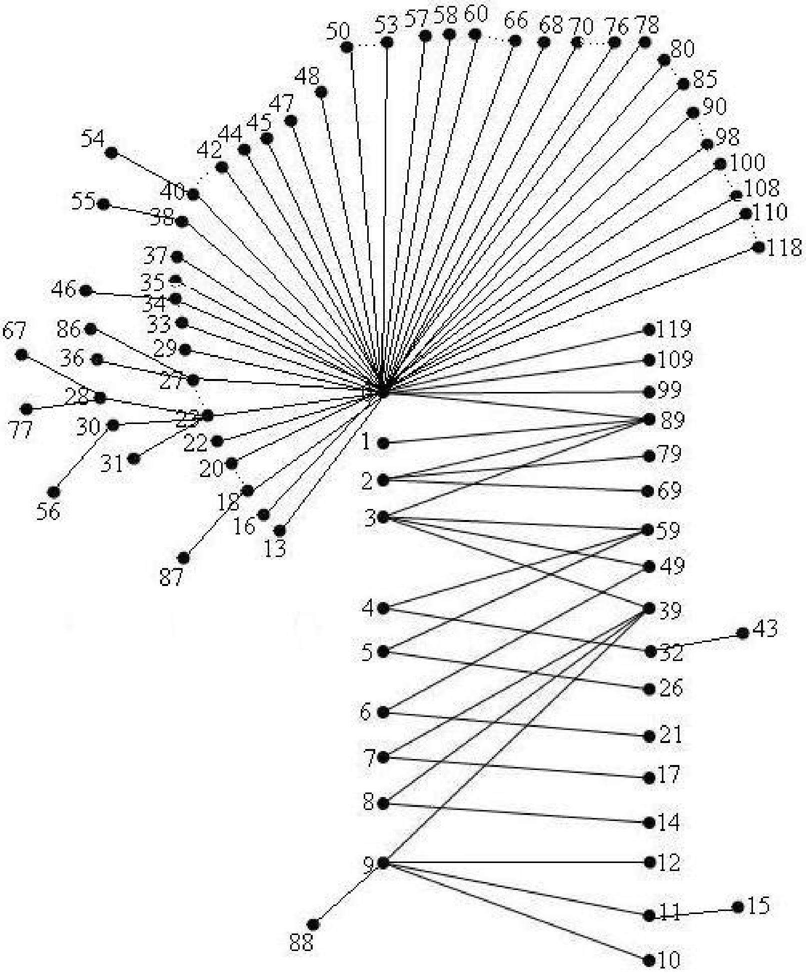

Figure 12: Graceful Tree T_8^* with 119 edges

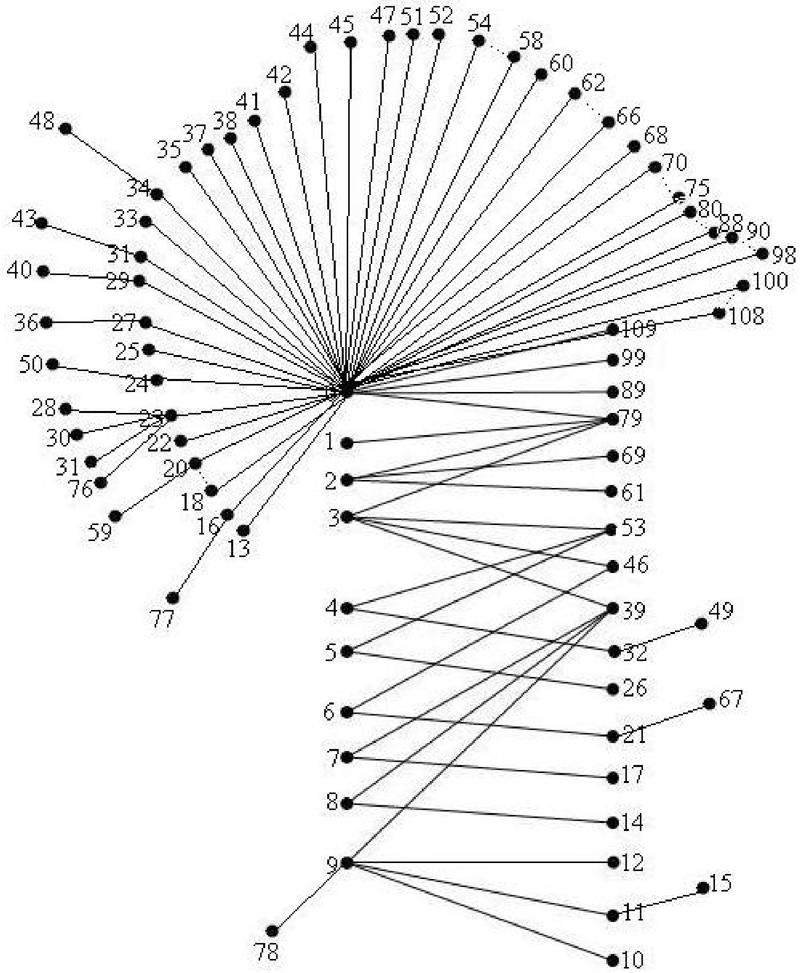

Figure 13: Graceful Tree T_θ^* with 109 edges, $\theta = 12$

Acknowledgments

The Authors would like to thank the referees for the valuable comments and suggestions. The second author thankfully acknowledges Centre for Research, Anna University, Chennai for the financial support through the Anna Centenary Research Fellowship under the grant Ref:CR/ACRF/Jan.2011/31.

References

[1] B.D. Acharya, S.B. Rao, S. Arumugam, *Embedding and NP-Complete problems for Graceful Graphs, Labelings of Discrete Structures and Applications*, B.D. Acharya, S. Arumugam, Alexander Rosa, eds., (2008), 57-62, Narosa Publishing House, New Delhi.

[2] Aldred R.E. and Mckay B.D, *Graceful and harmonious labelings of trees*, Bull. Inst. Comb. Appl., 23, (1998), 69-72.

[3] Bermond J.C. and Sotteau D, *Graph decompositions and G-design*, Proc. 5th British Combin. Conf., 53-72, (second series), 12, (1989), 25-28.

[4] Bermond J.C, *Graceful graphs, radio antennae and French windmills*, Graph Theory and Combinatorics, Pitman, London, (1979), 18-37.

[5] Bloom G.S, *A chronology of the Ringel-Kotzig conjecture and the continuing quest to call all trees graceful*, Ann. N.Y. Acad. Sci., 326, (1979), 35-51.

[6] Burzio M. and Ferrarese G, *The subdivision graph of a graceful tree is a graceful tree*, Disc. Math, 181, (1998), 275-281.

[7] Chen W.C, Lu H.I and Yeh Y.N, *Operations of interlaced trees and graceful trees*, Southeast Asian Bulletin of Mathematics, 21, 337-348.

[8] David Morgan, *Gracefully labeled trees from Skolem sequences*, Proc. of The Thirty-First Southeastern Internat. Conf. on Combin., Graph The-

ory and Computing (Boca Raton, FL, 2000), Congresses Numberantium, 142, 41-48.

[9] Edwards M, Howard L, *A survey of graceful trees*, Atlantic Electronic J. of Mathematics, 1,1, Summer 2006.

[10] J.A.Gallian, *A Dynamic Survey of Graph Labeling*, The Electronic Journal of Combinatorics, 18, (2011), #DS6.

[11] Golomb S.W, *How to number a graph*, Graph Theory and Computing R.C. Read, ed., Academic Press, New York, 1972, 23-37.

[12] Hegde S.M and Shetty S, *On graceful trees*, Applied Mathematics E-Notes, 2, (2002), 192-197.

[13] Jeba Jesintha.J and Sethuraman G, *All arbitrary fixed generalized banana trees are graceful*, Math. Comput. Sci, 5, (2011),1,51-62.

[14] Jeba Jesintha, *New Classes of Graceful Trees*, Ph.D Thesis, (2005), Anna University, Chennai, India.

[15] Kotzig A, *Decompositions of a complete graph into 4k-gons (in Russian)*, Matematicky Casopis, 15, (1965), 229-233.

[16] Koh K.H., Rogers D.G. and Tan T, *Two theorems on graceful trees*, Discrete Mathematics, 25, (1979), 141-148.

[17] Ljiljana Brankovic and Ian M. Wanless, *Graceful Labelling: State of the Art, Applications and Future Directions*, Math. Comput. Sci, 5, (2011),1,11-20.

[18] Michael Horton, *Graceful trees statistics and algorithms*, Master's Thesis, http://eprints.comp.utas.edu.au:81/archieve/00000019/01/.

[19] Ng H.K, *Gracefulness of a class of lobsters*, Notices AMS, 7, (1986), 825-05-294.

[20] Pastel A.M and Raynaud H., *Numerotation gracieuse des olivers*, Colloq. Grenoble, Publications Universite de Grenoble, (1978), 218-223.

[21] Pavel Havier and Alfonz Havier, *All trees of diameter five are graceful*, Discrete Mathematics, 233, (2001), 133-150.

[22] Ringel G, *Problem 25, in Theory of Graphs and its Applications*, Proc. Symposium Smolenice, Prague, (1963) page-162.

[23] Rosa A,*On certain valuations of the vertices of a graph*, Theory of graphs, (International Symposium, Rome, July 1966), Gordon and Breach, N.Y. and Dunod Paris, (1967),349-355.

[24] Sethuraman G. and J. Jeba Jesintha, *All banana trees are graceful*, Advanced Applied Discrete Mathematics, 4, (2009), 1, 53-64.

[25] Sethuraman G. and Venkatesh S, *Decomposition of complete graphs and complete bipartite graphs into α-labeled trees*, Ars Combinatoria, 93, (2009), 371-385.

[26] Sekar C,*Studies in graph theory*, Ph.D. thesis, (2002), Madurai Kamaraj University.

[27] Stanton R. and Zarnke C, *Labeling of balanced trees*, Proc. 4th Southeast Conf. Combin. Graph Theory, Computing, (1973), 479-495.

[28] Van Bussel F.,*Relaxed graceful labelings of trees*, The Electronic Journal of Combinatorics, 9, (2002), #R4.

[29] Wang J.G., Jin D.J., Lu X.G. and Zhang D., *The gracefulness of a class of lobster trees*, Math. Comput. Modelling, 20, (1994), 105-110.

[30] West D.B., *Introduction to Graph Theory*, Prentice Hall of India, 2nd Edition, 2001.

[31] Yao B, Yao M, Cheng H,*On gracefulness of directed trees with short diameters*,Bull. Malays. Math. Sci. Soc.,(2), 35, (2012), 1, 133-146.