On Gracefully Labeling Paths

A Thesis Presented to

The Faculty of the Mathematics Program

California State University Channel Islands

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science

by

Cory C. Yi

October 31, 2011

Signature page for the Masters in Mathematics Thesis of Cory Chong Yi

APPROVED FOR THE MATHEMATICS PROGRAM

Cynthia J. Wyels 11/4/11

Dr. Cynthia J. Wyels, Thesis Advisor Date

[illegible] 10/31/11

Dr. Gregory Wood, Thesis Committee Date

APPROVED FOR THE UNIVERSITY

[illegible] 11-7-11

Dr. Gary A. Berg, AVP Extended University Date

Abstract

An unproven claim is that all trees may be gracefully labeled. However there are some special classes of trees that are proven to have graceful labelings. A path is the simplest form of a tree, and it has been proven that all paths can be gracefully labeled. The focus of this study is on the characteristics of gracefully labeled paths and a method for producing graceful labelings of P_n with given properties. We report on progress towards a proof that labelings of paths of any size may assign the label 1 to any node and be completed as graceful labelings. Representations such as the *Edge Tree Diagram* and the *Matrix-Entry Choosing* methods are developed. We also prove that certain assignments of labels to the first two vertices of a path guarantee that a labeling may not be completed to form a graceful labeling. Finally we develop a computer program to generate gracefully labeled paths to assist in examining the results and identifying conjectures worthy of further study.

I. Introduction

A very well-known Graceful Tree Conjecture, also known as Ringel-Kotzig or Rosa's Conjecture, which states that all trees have a graceful labeling. Although there were many independent studies done in attempt to prove the Graceful Tree Conjecture, it still remains as an open problem. However, some progress has been made including proving some specialized classes of trees have graceful labelings, and with the aid of computer programs, all trees with vertices up to 29 vertices have been proven to have graceful labelings. The classes of trees that are known to have graceful labelings are described in section IB. The simplest form of these trees, paths (sometimes called chains), is the focus of this study. We develop an algorithm to generate graceful labelings of P_n and identify common properties of gracefully labeled paths. A one such property leads to a problem known as "labeling completion". Given an assignment of labels to same vertices of a path, one asks whether labels may be assigned to the remaining vertices in a way that produces a graceful labeling. We investigate one such labeling completion problem, which states if a labeling $f(P_n)$ satisfies $f(v_1) = 2$ and $f(v_2) = n - k$ for $k \geq 3$ then f may not be completed to form a graceful labeling. In later sections, we also discuss some tools produced along the way to help investigate the graceful labeling problems. One such tool is a computer program called *Labelit* that can gracefully label P_n with properties specified by the user. This is discussed in the last sections of the paper.

Examination of the collections of gracefully labeled paths produced by the *Labelit* application leads to further conjectures regarding properties of gracefully labeled paths. We discuss such observations and conjectures. So, the goal of this study is adding to what we already know about gracefully labeling trees and building blocks that may be useful towards proving the graceful tree conjecture.

A. Graceful Labeling

A graceful labeling is an assignment of the integers $\{1 \cdots n\}$ to vertices of a graph such that, once each edges is labeled with the absolute difference of its incident vertices, with each integer in $\{1 \cdots n - 1\}$ is used once and only once (see FIG. 1).

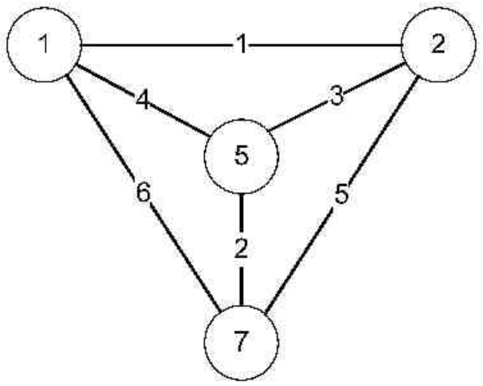


FIG. 1 A graceful labeling of a graceful graph K_4

A tree is said to be gracefully labeled if all *n* vertices of the tree are labeled with integers [*1*...*n*] such that the edge labels induced by taking absolute difference between the two end vertices are exactly the set [*1*...*n-1*] (see FIG. 2).

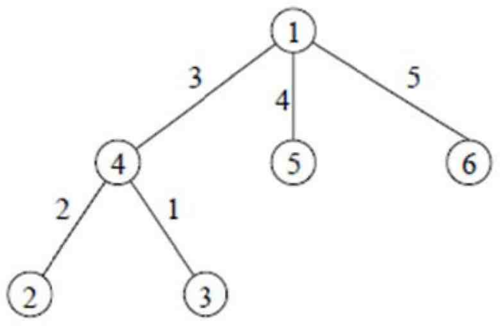


FIG. 2 Example of a gracefully labeled tree

B. Classes of Trees Known to have Graceful Labeling

Some specialized classes of trees proven to have graceful labeling are paths, caterpillars, *m*-stars, olive trees, banana trees, tp-trees, and product trees. These classes of trees are described below.

1. Paths

A path, also called a trail or a chain, is a sequence of distinct vertices $x_1, x_2 \ldots, x_n$, such that (x_1, x_2), $(x_2, x_3), \ldots, (x_{n-1}, x_n)$ are the edges of the graph. A path is the simplest form of a tree (see FIG. 3).

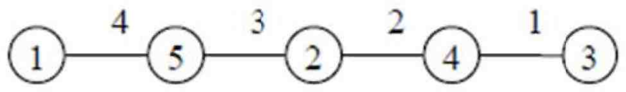


FIG. 3 A graceful labeling of P_5

A path can be gracefully labeled using the standard labeling method described below. It is a simple method in generating gracefully labeled paths of length *n*. We call this the standard labeling of the path on *n* vertices; the proof of why the standard labeling is also a graceful labeling is available in section C.

The standard labeling of P_n *is* $g: V(P_n) \rightarrow \{1,2,\ldots,n\}$ where

$$g(v_i) = \begin{cases} \frac{i+1}{2}, & i \text{ is odd}, \\ n - \frac{i}{2} + 1, & i \text{ is even}. \end{cases}$$

Alternatively, we may write $g(v_{2j+1}) = j$ and $g(v_{2j}) = n - j + 1$.

2. Caterpillars

A caterpillar is a tree in which every graph vertex is on a central stalk or only one graph edge away from the stalk (in other words, removal of its endpoints leaves a path graph; Gallian 2007). A tree is a caterpillar if and only if all nodes of degree at least three are surrounded by at most two nodes of degree two or greater [3] (see FIG. 4).

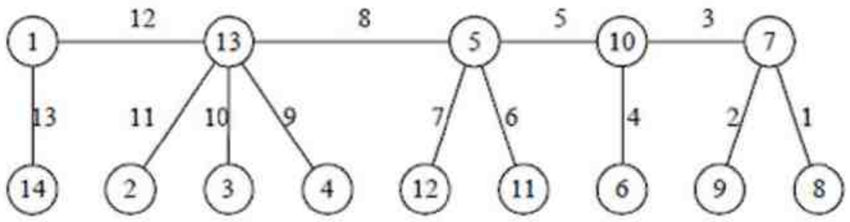


FIG. 4 A caterpillar can be recognized by its long center stalk

A graceful labeling of caterpillars is possible using the *Caterpillar Labeling Algorithm* described below. The algorithm uses the traversing nodes of a tree that is similar to the "depth first" traversing approach. The only difference is that a node of the stalk is considered before its legs are considered while traversing.

a) Caterpillar Labeling Algorithm

1. Traversing from left to right and starting from the left most node of the first leg, do the following
   - Visit and label nodes in increasing order
   - Current node gets a label if it does not have a node incident with a labeled node
   - Stop if there are no more nodes to be labeled to the right
2. Starting from the rightmost unlabeled node, traverse from right to left and do the following
   - Label nodes in increasing order
   - Current node gets a label if does not already have a label
   - Stop if all nodes have been labeled

The following example of a graceful labeling of a caterpillar is completed using the *Caterpillar Labeling Algorithm*. Nodes with labels 1-7 are labeled by the first step and the nodes 8-13 are labeled by the second step of the algorithm (see FIG. 5).

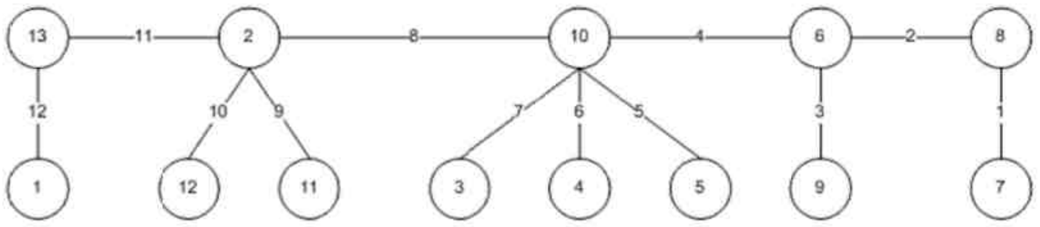


FIG. 5 Caterpillar Labeling Algorithm generated a graceful labeling of caterpillar with $n = 13$

3. *M*-Stars

An m-star has a single root node with any number of paths of length m attached to it (see FIG. 6). Cahit and Cahit proved that all m-stars are graceful (Cahit & Cahit 1975).

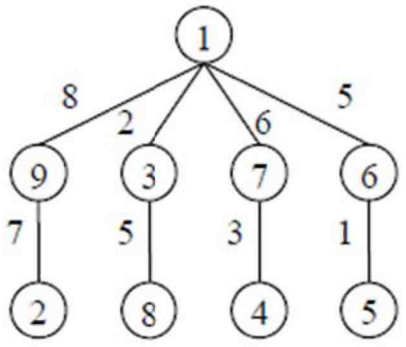


FIG. 6 A 2-star, proved to have graceful labeling by Cahit and Cahit's algorithm

4. Additional Classes of Trees

a) Olive trees

An olive tree has a root node with k branches attached: the i^{th} branch has length i (FIG. 7). Patel and Raynud proved that all olive trees are graceful (Pastel & Raynaud 1978).

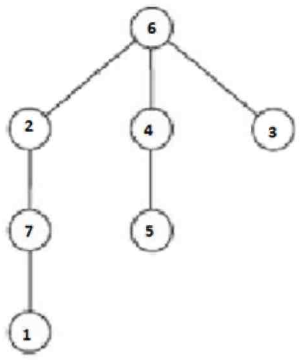


FIG. 7 The $k = 3$ olive tree

b) Banana trees

A banana tree is constructed by bringing multiple m-stars together at a single vertex (Chen et al. 1997) (see FIG. 8). Banana trees have not been proved graceful, although Bhat-Nayak and Deshmukh have proven the gracefulness of certain classes of banana tree (Bhat-Nayak & Deshmukh 1996).

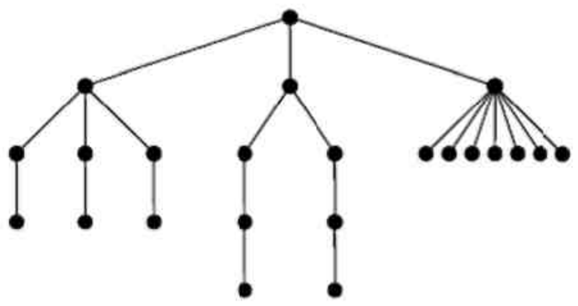


FIG. 8 A banana tree constructed from a 2-star, 3-star and 1-star

c) Tp-Trees

Hegde and Shetty defined a class of tree called "Tp-trees" created by taking a gracefully labeled path and shifting some of the edges (see FIG. 9). They proved that these can always be gracefully labeled using the original path labels (Hedge & Shetty 2002).

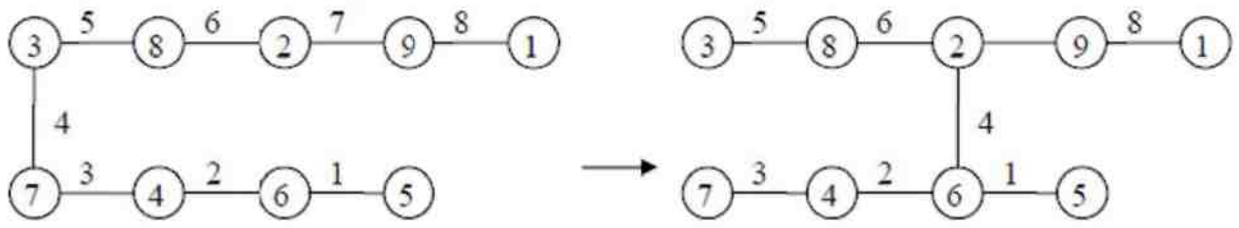


FIG. 9 rearranging a gracefully labeled path to generate a gracefully labeled Tp-tree

d) Product Trees

Some proofs show that certain graceful trees can be combined to give a larger graceful tree. Koh et al. show how 'rooted product' trees are always graceful (Koh et al. 1980). An example from their paper is given in Figure I-8. Each of the trees labeled G shares one vertex with the tree labeled H (see FIG. 10).

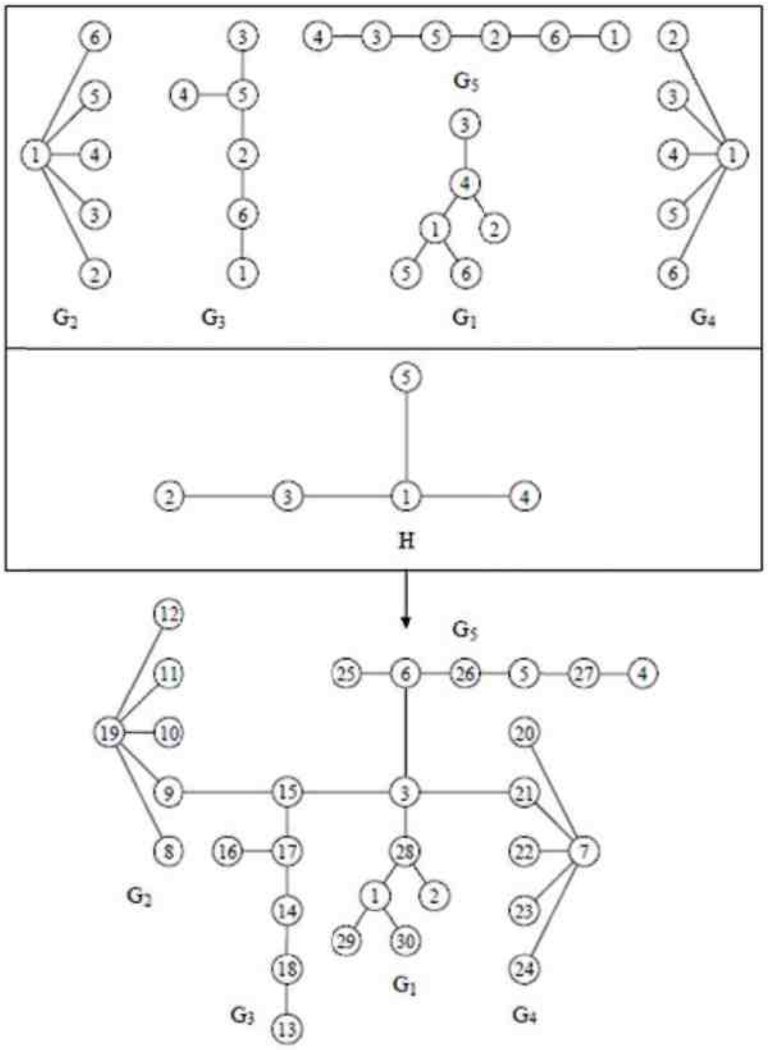


FIG. 10 Example of a graceful product tree

C. Standard Notations and Conventions

We establish some standard notation and conventions that are useful for presenting and proving graceful labeling problems.

We denote the path on *n* vertices as P_n ; the vertex set is $V(P_n) = \{v_1, v_2, \dots, v_n\}$ and the edge set is $E(P_n) = \{(v_i, v_{i+1}) \mid 1,2, \dots, n-1\}$.

1. Graceful Labeling

Graceful Labeling: A graceful labeling on graph *G* is a function f $V(G) \rightarrow \{1,2, \dots |E(G)+1|\}$ satisfying

- $f(v_i) \neq f(v_j)$ for $i \neq j$ (vertex labels are distinct), and

- The set of induced edge labels, where edge $e = \{v_i, v_j\}$ receives label $|f(v_i) - f(v_j)|$, are exactly the set $\{1, 2, \dots, |E(G)|\}$.

2. Standard Labeling Proof

As described in section B, the standard labeling is a simple technique used to gracefully label P_n. We start by duplicating the definition here and followed with a simple proof that the standard labeling is a graceful labeling.

Standard Labeling: The *standard labeling* of P_n *is* $g: V(P_n) \rightarrow \{1,2,\dots,n\}$ where

$$g(v_i) = \begin{cases} \frac{i+1}{2}, & i \text{ is odd}, \\ n - \frac{i}{2} + 1, & i \text{ is even}. \end{cases}$$

Proof that the Standard Labeling is a Graceful Labeling:

Let $X = \{g(v_i) \mid i\ odd\}$ and $Y = \{g(v_i) \mid i\ even\}$, then the elements of the sets X and Y are non-decreasing numbers $1 \leq g(v_i) \leq \left\lceil \frac{n}{2} \right\rceil$, and non-increasing numbers $n \geq g(v_i) \geq \left\lceil \frac{n+1}{2} \right\rceil$ respectively. Since $\frac{i+1}{2} \neq n - \frac{i}{2} + 1$ for $i = 1..n$ we have $X \cap Y = \emptyset$ and $X \cup Y = \{1, \dots, n\}$. In other words, $\{g(v_i) \mid 1 \leq i \leq n\} = \{1, 2, \dots, n\}$.

Next consider two sets $L = \{|g(v_i) - g(v_{i+1})| : 1 \leq i < n\ and\ i\ is\ odd\}$ and $M = \{|g(v_i) - g(v_{i+1})| : 2 \leq i < n\ and\ i\ is\ even\}$ Then the largest element of L and M is n-1 for $g(v_i) = 1$ and $g(v_{i+1}) = n$ or $g(v_i) = n$ and $g(v_{i+1}) = 1$ case. The smallest is 1 for cases where $|g(v_i) - g(v_{i+1})| = 1$. The sets L and M are necessarily distinct sets with unique elements since $g(v_i)$ is unique for all $i = 1..n$. In addition, since $|L| = \frac{n}{2}$ and $|M| = n - \frac{n}{2} - 1$, we see $|L| + |M| = \frac{n}{2} + n - \frac{n}{2} - 1 = n - 1$. And hence $L \cap M = \emptyset$ with the smallest and largest elements being 1 and $n - 1$ respectively, it is necessary that $L \cup M = \{1, \dots, n-1\}$. This proves that the standard labeling is a graceful labeling.

3. Trees with diameter not exceeding 5

The distance between two vertices is the number of edges in a shortest path connecting them. Diameter is then maximum distance taken over all pairs of vertices. Hrnciar and Havier extended the proof for caterpillars to show that all trees with $diameter \leq 5$ are graceful (Hrncia & Havier 2001).

II. Representations and Results

A. Representations

Some graceful labeling problems are better understood through the graphical representations. We discuss representations used for studying graceful labeling the Edge Tree Diagram and the Symmetry Matrix. Employing such representations is particularly useful for manifesting a visual proof of edge-pair selections such that no graceful labeling of P_n exists. Example of proofs using Edge Tree Diagram and Symmetry Matrix representations are described in later sections.

1. The Edge Tree Diagram

Since graceful labeling requires that each edge must be uniquely labeled, an Edge Tree Diagram can be constructed by generating all possible distinct vertex-pairs [i.e. differing respect to at least one vertex-pair] using the numbers 1, $..,n$. We next group all pairs inducing the same edge label, on each row (see FIG. 11).

$$\begin{gathered}\{(1,n)\}\\ \{(1,n-1),(2,n)\}\\ \{(1,n-2),(2,n-1),(3,n)\}\\ \{(1,n-3),(2,n-2),(3,n-1),(4,n)\}\\ \vdots\\ \{(1,2),(2,3),(3,4),\qquad \cdots,\qquad (n-1,n)\}\end{gathered}$$

FIG. 11 Diagram of Edge Tree of *n-1* rows with each row yielding same edge labels

The result is an Edge Tree Diagram of n - 1 rows with each row representing vertex label pairs yielding same edge labels. The diagonals of the Edge Tree Diagram have the same length with a common vertex label (see FIG. 12).

(1, 9)
(1, 8) (2, 9)
(1, 7) (2, 8) (3, 9)
(1, 6) (2, 7) (3, 8) (4, 9)
(1, 5) (2, 6) (3, 7) (4, 8) **(5, 9)**
(1, 4) **(2, 5)** (3, 6) (4, 7) **(5, 8)** (6, 9)
(1, 3) (2, 4) **(3, 5)** (4, 6) **(5, 7)** (6, 8) (7, 9)
(1, 2) (2, 3) (3, 4) **(4, 5) (5, 6)** (6, 7) (7, 8) (8, 9)

FIG. 12 Row and diagonal of an Edge Tree Diagram shown in bold

A graceful labeling of P_n with *n*-1 edges can be constructed by making pair selections from the Edge Tree Diagram according to the following set of rules:

1. Exactly one pair must be selected from each row.
2. No more than two pairs can reside on the same adjacent diagonal.
3. A selection leading to a cycle is not allowed

A Graceful labeling of P_n is a sequence $\{x_1, x_2, \dots, x_n\}$ such that $\{(x_1, x_2), (x_2, x_3), \dots, (x_{n-1}, x_n)\}$ is a collection of $n-1$ pairs from the edge tree diagram and the x_i are distinct. When making selections, the individual vertex label must occur twice in the collection of pairs with the exception of two that occur once each. Rule 1 ensures that each possible edge value occurs exactly once in the collection while rule 2 enforces having no more than two appearances of the same vertex label in the collection of pairs. Selection of $n-1$ pairs that lead to a labeling of the entire path is enforced by rule 3. Note that the adjacent diagonals mentioned in rule 2 designate all pairs having a common vertex label.

For example, we use the Edge Tree Diagram to construct a graceful labeling of P_9 by collecting pairs (1,9), (9,2), (2,8), (8,3), (3,7), (7,4), (4,6), and (6,5) which are selected from each row while observing three rules stated above (see FIG. 13). This gives the labeling [1, 9, 2, 8, 3, 7, 4, 6, 5] shown in FIG. 14.

(1, 9)
(1, 8) **(2, 9)**
(1, 7) **(2, 8)** (3, 9)
(1, 6) (2, 7) **(3, 8)** (4, 9)
(1, 5) (2, 6) **(3, 7)** (4, 8) (5, 9)
(1, 4) (2, 5) (3, 6) **(4, 7)** (5, 8) (6, 9)
(1, 3) (2, 4) (3, 5) **(4, 6)** (5, 7) (6, 8) (7, 9)
(1, 2) (2, 3) (3, 4) (4, 5) **(5, 6)** (6, 7) (7, 8) (8, 9)

FIG. 13 Selection of pairs from each row yielding a graceful labeling of P_9

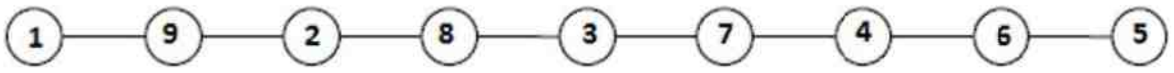


FIG. 14 A graceful labeling of P_9

2. Symmetry Matrix

A Symmetry Matrix is similar to the Edge Tree Diagram together with its reflection about its base. It is an $n \times n$ matrix with each row and column selection representing the vertex-pair inducing an edge label. The rows and columns of the symmetry matrix represent different combinations in creating vertex-pair and the diagonals of the matrix are the vertex-pair combinations inducing the same edge label. This is analogous to the diagonals and rows of the Edge Tree Diagram. On the main diagonal of the symmetry matrix, we put zeroes to indicate 0 inducing blocks (see FIG. 15). The additional information provided in the symmetry matrix is the reflection about its main diagonal, which indicates that vertex-pairs $(1, n)$ and $(n, 1)$ induces the same edge label. We represent the main diagonal as 1, and the number of the diagonal increases as it moves away from the main diagonal. So diagonal n represents blocks $(1, n)$ and $(n, 1)$. We mark the block with 'X' to show selected vertex-pair (row, col).

ROW\COL	1	2	3	4	5	6
1	0			▨		X
2		0			▨ X	X
3			0	X	X	▨
4	▨		X	0		

5		X	X		0	
6	X	X	▨			0

FIG. 15 Symmetry Matrix representing standard labeling with $n = 6$

B. Results

We now show key ideas in employing the graphical representations to address a labeling completion problem. Three independent proofs are provided for the same problem: The first is a straightforward method, the second and third employ the Edge Tree Diagram and the Symmetric Matrix respectively. A labeling completion problem is stated below as *Theorem 1*, which states that with an initial vertex label of 2 is followed by a vertex label that is insufficiently large results in a labeling that cannot be completed to form a graceful labeling.

1. Theorem 1

Theorem 1: Let f be a labeling of P_n, $n \geq 4$. If $f(v_1) = 2$ and $f(v_2) = n - k$ for $k \geq 3$, then f is not graceful.

We provide the first proof using a straightforward method, which gathers all different ways to generate edge labels and make selections for graceful labeling.

Proof 1.1 (Straightforward method): We construct all possible distinct vertex-pairs:
$\{(i_n, j_n), (i_{n-1}, j_{n-1}), (i_{n-2}, j_{n-2}), (i_{n-3}, j_{n-3}), \ldots, (i_2, j_2)\}$

so that

$(i_n, j_n) \in \{(1, n)\}$

$(i_{n-1}, j_{n-1}) \in \{(1, n-1), (2, n)\}$

$(i_{n-2}, j_{n-2}) \in \{(1, n-2), (2, n-1), (3, n)\}$

$(i_{n-3}, j_{n-3}) \in \{(1, n-3), (2, n-2), (3, n-1), (4, n)\}$

$$\vdots$$

$$(i_2, j_2) \in \{(1,2), (2,3), (3,4), \cdots, (n-1, n)\}$$

The first pair (i_n, j_n) has only the choice $(1, n)$, the second pair has two choices $(1, n-1)$ *and* $(2, n)$, ..., the last pair has $(n-1)$ choices $(1,2), (2,3), (3,4), \cdots$, *and* $(n-1, n)$. All these choices can be made independently.

Suppose $f(v_1) = 2$ *and* $f(v_2) \leq n - 3$ and the set of selection pairs $\{(i_n, j_n), (i_{n-1}, j_{n-1}), (i_{n-2}, j_{n-2}), (i_{n-3}, j_{n-3}), \ldots, (i_2, j_2)\}$ represents a graceful labeling of a path of length n. Then the vertex-pairs $(2, n), (2, n-1),$ *and* $(2, n-2)$ are not in the selection since $f(v_2) \leq n - 3$. In addition, since a vertex label cannot be used more than twice in the selection of pairs, vertex labels 1 and n cannot be used more than twice in the selection of pairs. Hence the first three pairs selected from the rows 1 through 3 must be the pairs with $(i_n, j_n) = (1, n)$, $(i_{n-1}, j_{n-1}) = (1, n - 1)$, and $(i_{n-2}, j_{n-2}) = (3, n)$. This leaves $(i_{n-3}, j_{n-3}) = (3, n - 1)$ as the only choice available for the fourth row. Since there are no other selections available for the first four rows, and the selection creates a cycle, then a set of pairs $\{(i_n, j_n), (i_{n-1}, j_{n-1}), (i_{n-2}, j_{n-2}), (i_{n-3}, j_{n-3}), \ldots, (i_2, j_2)\}$ is not a set of pairs satisfying graceful labeling.

The next proof of *Theorem 1* employs an Edge Tree Diagram and systematically removes rows and diagonals as edge pairs are selected from the edge tree diagram.

Proof 1.2 (using Edge Label Diagram): Since $f(v_1) = 2$, we systematically remove the set of pairs $(2, n), (2, n - 1), (2, n - 2), \ldots, and\ (2,3)$ from the set of remaining pairs available for selection. Otherwise having additional pairs with vertex label 2 cannot be completed to a gracefully labeling. Also note that since $f(v_2) \leq n - 3$, $f(v_2)$ must be selected from the Edge Label Diagram with a $row \geq 5$. Hence, the only selections available from the first four rows after removing the diagonal pairs having vertex label 2 are highlighted in bold (see FIG. 16). However since the available selections induces a cycle, there cannot exist a graceful labeling.

$$\{(\mathbf{1}, \boldsymbol{n})\}$$

$$\{(\mathbf{1}, \boldsymbol{n} - \mathbf{1}), (2, n)\}$$

$$\{(1, n-2), (2, n-1), (\mathbf{3}, \boldsymbol{n})\}$$

$$\{(1, n-3), (2, n-2), (\mathbf{3}, \boldsymbol{n} - \mathbf{1}), (4, n)\}$$

$$\{(1, n-4), (2, n-3), (3, n-2), (4, n-1), (5, n)\}$$

$$\vdots$$

$$\{(1,2), \quad (2,3) \quad (3,4), \quad \cdots, \quad (n-1, n)\}$$

FIG. 16 An Edge Tree showing selections creating a cycle.

Proof by Symmetric Matrix representation is addressed next. It also involves systematically removing rows and columns as each selection is made, thus signifying no further choices are available in the row and column.

Proof 1.3 (by Symmetric Matrix): We start by removing row 2 and column 2; since $f(v_1) = 2$ and $f(v_2) \leq n - 3$. Next we mark (with an X) the blocks $(n, 1)$ and $(1, n)$, since there is only one selection (see FIG. 17). Moving onto the diagonal $n - 1$, there is only one selection available so we mark blocks $(1, n - 1)$ and $(n - 1, 1)$ and remove the row and column 1 respectively from further consideration (see FIG. 18). Continuing in a similar fashion, we find that the selections create a cycle. This completes the proof that a graceful labeling does not exist.

ROW\COL	1	2	3	4	...				n-1	n
1	0				...					X
2		0			...					
3			0		...					
4				0	...					
...	...	...	...	...	...	...	...	...	...	...
n-4					...					0
n-3					...					
n-2					...					
n-1					...					
n	X				...					

FIG. 17 Column 2 is removed since the first vertex is labeled 2 ($f(v_1) = 2$) and the second vertex is labeled $f(v_2) = k \; for \; some \; k \leq n - 3$

ROW\COL	1	2	3	4	...				n-1	n
1	0				...				X	X
2		0			...					
3			0		...					
4				0	...					
...	...	...	...	...	...	...	...	...	...	...
n-4					...					0
n-3					...					
n-2					...					
n-1	X				...					
n	X				...					

FIG. 18 Only one choice available for the diagonal *n-1*

ROW\COL	1	2	3	4	...				n-1	n
1	0				...				X	X
2		0			...					
3			0		...				X	X
4				0	...					
...	...	...	...	...	...	...	...	...	...	...
n-4					...					0
n-3					...					
n-2					...					
n-1	X		X		...					
n	X		X		...					

FIG. 18 No selection available for diagonal $n - 4$ leading to no graceful labeling

2. Corollary 1

Corollary 1: Let f be a labeling of $P_n, n \geq 4$. If $f(v_1) = n - 1$ and $f(v_2) = k$ for $k > 3$, then f is not graceful.

The proof of the above Corollary 1 is not provided here since the logic is analogous to that in the proof of Theorem 1.

We have shown that both Theorem 1 and Corollary 1 manifests an example of restrictions on the second vertex label given the first. However are there any restrictions on beginning with any vertex label on any vertex?

Question: Let f be a labeling of P_n, and suppose $f(v_i) = j$ for $i, j \in \{1, \cdots, n\}$. Are there values of i, j, and n for which f may not be graceful?

Answer: Yes (example with $n = 5, i = j = 3$). Note that it appears the answer seems to be "no" for sufficiently large n. The proof of this may be included in the further studies of the interested.

III. Conjectures and Supporting Evidence

The proof of the main conjecture described in section B was attempted but partially proved and still remains as an open problem. Nonetheless some evidence shows that the conjecture may be true and some supporting data is provided. We start with some standard notions and phrases that are helpful for supporting the proofs.

A. Main Conjecture

In this section, we show a progress towards a proof that labelings of paths of any size may assign the label 1 to any node and be completed as graceful labelings. The *Labelit* program is used to generate all graceful labeling of P_n with $n \leq 18$ and the results appears to support in favor of the Main Conjecture. Some computer generated labelings of P_{10} is shown on FIG. 19. Additional computer generated results are also available in the Appendix.

1-10-2-9-3-8-4-7-5-6

5-**1**-10-2-9-3-8-6-7-4

4-10-**1**-9-2-5-6-8-3-7

7-2-10-**1**-8-6-3-9-5-4

4-7-2-10-**1**-8-6-5-9-3

8-4-9-2-10-**1**-7-6-3-5

3-8-5-9-2-10-**1**-7-6-4

6-8-5-4-9-2-10-**1**-7-3

4-7-6-8-3-9-2-10-**1**-5

6-5-7-4-8-3-9-2-10-**1**

FIG. 19 Examples of graceful labeling of P_{10} having label "1" on every vertex

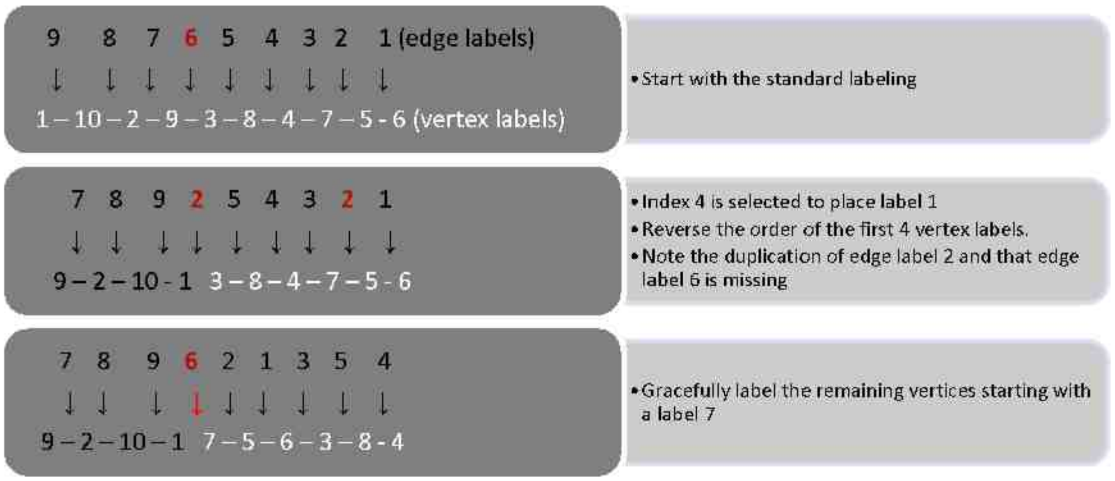
1. Main Conjecture

Main Conjecture: *For each i between 1 and n, there is a graceful labeling f $V(P_n) \to \{1, \dots, n\}$ with $f(v_i) = 1$.*

The proof of the *main conjecture* is useful for constructing and building up gracefully labeled trees by "*gluing*" smaller gracefully labeled trees together. The proofs of the main conjecture require two parts which we describe next.

2. Strategy for the Proof

The general idea behind the proof of the *main conjecture* takes on two parts. The first part proves that any edge label from the *standard labeling* can be induced from the vertex labels to its right. The proof of the second part necessitates proof for the existence of a graceful labeling which begins with any vertex label $i = 1, \dots, n$ (see FIG. 20). The second part of the proof is still an open problem and we only provide some information gained from the study. Hence the main conjecture is still an open problem.



9 8 7 6 5 4 3 2 1 (edge labels)
1 – 10 – 2 – 9 – 3 – 8 – 4 – 7 – 5 - 6 (vertex labels)
• Start with the standard labeling
7 8 9 2 5 4 3 2 1
9 – 2 – 10 - 1 3 – 8 – 4 – 7 – 5 - 6
• Index 4 is selected to place label 1
• Reverse the order of the first 4 vertex labels.
• Note the duplication of edge label 2 and that edge label 6 is missing
7 8 9 6 2 1 3 5 4
9 – 2 – 10 – 1 7 – 5 – 6 – 3 – 8 - 4
• Gracefully label the remaining vertices starting with a label 7

FIG. 20 General idea behind proving the *initial conjecture*

3. Theorem 2 (the first part):

Theorem 2: *For every standard labeling of P_n, there exists some $k > j$ such that* $1 < j \leq \left\lceil \frac{n}{2} \right\rceil$ *and* $\left|g(v_{j+1}) - g(v_j)\right| = g(v_k) - 1$.

The proof of the theorem 2 takes on two cases; we use whether the index j has even or odd values to determine how to bipartition g.

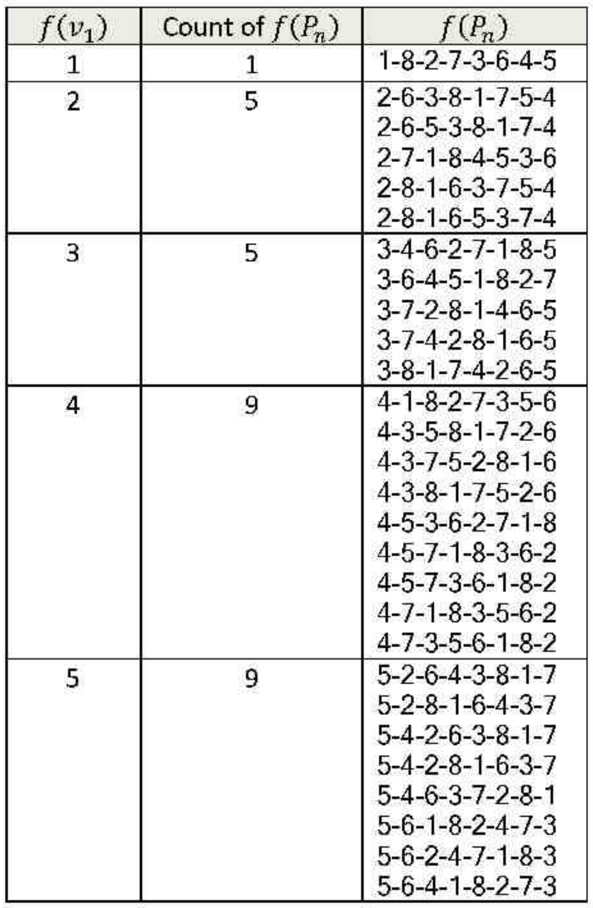
Proof 2.1 (case for j odd): We can bipartition g into two disjoint sets S and T at j, so that $S = \{g(v_p) \mid 1 \leq p \leq j\}$ *and* $T = \{g(v_q) \mid j + 1 \leq q \leq n\}$. The induced edge labels of set S are exactly the set $E(S) = \{n - 1, n - 2, \dots, n - j + 1\}$. An edge label $e = n - j$ adjoins S and T. T is the set of vertex labels alternating between $n - \frac{q}{2} + 1$ *and* $\frac{q+1}{2}$. The largest label of T is the first element of T: $n - \frac{j+1}{2} + 1$. We need a label $g(v_k) = n - j + 1$. Since $j \leq \left\lceil \frac{n}{2} \right\rceil$, the value $n - j + 1$ occurs when counting down from $n - \frac{j+1}{2} + 1$.

Proof 2.2 (case for j even): We can bipartition g into two disjoint sets S and T at j, so that $S = \{g(v_p) \mid 1 \le p \le j\}$ and $T = \{g(v_q) \mid j+1 \le q \le n\}$. The induced edge labels of set S are exactly the set $E(S) = \{n-1, n-2, \dots, n-j\}$. An edge label $e = n - j$ adjoins S and T. T is the set of vertex labels alternating between $\frac{q+1}{2}$ and $n - \frac{q}{2} + 1$. The largest label of T is the second element of T: $n - \frac{j}{2}$. We need a label $g(v_k) = n - j + 1$. Since $j \le \left\lceil \frac{n}{2} \right\rceil$, $n - j + 1$ occurs when counting down from $n - \frac{j}{2}$.

4. Conjecture 2 (the second part)

Conjecture 2: *For each i in $\{1, 2, \dots, n\}$, there exists a graceful labeling f of P_n for which $f(v_1) = i$.*

Conjecture 2 indicates that graceful labeling can assign any number $i = 1, \dots, n$ as its first vertex label and still be completed to for a graceful labeling of P_n. Although the proof is not available, using *Labelit* to generate graceful labelings of P_n provides some evidence in favor of conjecture 2. That is, graceful labeling of P_n exists with any $i = 1, \dots, n$ as its first vertex label, for $n \le 18$. Some computer generated results are shown, note that labeling the first vertex with 1 or *n* produces the fewest graceful labelings with a single solution; and the number of ways to gracefully label P_n increases as its first vertex label approaches $\left\lfloor \frac{n}{2} \right\rfloor$ (see FIG. 21).



$f(v_1)$	Count of $f(P_n)$	$f(P_n)$
1	1	1-8-2-7-3-6-4-5
2	5	2-6-3-8-1-7-5-4 2-6-5-3-8-1-7-4 2-7-1-8-4-5-3-6 2-8-1-6-3-7-5-4 2-8-1-6-5-3-7-4
3	5	3-4-6-2-7-1-8-5 3-6-4-5-1-8-2-7 3-7-2-8-1-4-6-5 3-7-4-2-8-1-6-5 3-8-1-7-4-2-6-5
4	9	4-1-8-2-7-3-5-6 4-3-5-8-1-7-2-6 4-3-7-5-2-8-1-6 4-3-8-1-7-5-2-6 4-5-3-6-2-7-1-8 4-5-7-1-8-3-6-2 4-5-7-3-6-1-8-2 4-7-1-8-3-5-6-2 4-7-3-5-6-1-8-2
5	9	5-2-6-4-3-8-1-7 5-2-8-1-6-4-3-7 5-4-2-6-3-8-1-7 5-4-2-8-1-6-3-7 5-4-6-3-7-2-8-1 5-6-1-8-2-4-7-3 5-6-2-4-7-1-8-3 5-6-4-1-8-2-7-3

		5-8-1-7-2-6-4-3
6	5	6-1-8-2-5-7-3-4 6-2-5-7-1-8-3-4 6-2-7-1-8-5-3-4 6-3-5-4-8-1-7-2 6-5-3-7-2-8-1-4
7	5	7-1-8-3-4-6-2-5 7-1-8-3-6-2-4-5 7-2-8-1-5-4-6-3 7-3-4-6-1-8-2-5 7-3-6-1-8-2-4-5
8	1	8-1-7-2-6-3-5-4

FIG. 21 Example of graceful labeling P_n having $f(v_1) = i$ where $i = 1, \dots, n$

Consider again the rules for gracefully labeling P_n with $n - 1$ edges from the Edge Tree Diagram described in the previous sections:

1. Exactly one pair must be selected from each row.
2. No more than two pairs can reside on the same adjacent diagonal.
3. A selection leading to a cycle is not allowed

The first rule ensures the unique edge labels are used while the second keeps individual vertex labels from occurring more than twice in the selections of pairs. If we ignore the last two rules, there are $(n - 1)!$ ways to make selections from each row and we have over counted to include graphs with cycles and graphs that are not paths (e.g. having more than two individual vertex labels, such as a star graph, S_{n-1} can be constructed by selecting from the first vertex-pairs from each row). If we assume that there is no graceful labeling for $f(v_1) = x$ for some $x = 1..n$, then it implies that all such selections must be one of pairs having three or more appearance of individual vertex labels, selections leading to cycles, or combination of both. Although the proof for this is not provided here, we can perhaps use this as a contradiction. That is, $(n - 1)!$ number of ways in selecting pairs from the edge tree diagram is larger than all possible selections that can be made with stated properties. The difference of the two is then counts all the graceful labelings of P_n.

In FIG. 22, we have listed some examples of graceful labeling of P_{10} with $f(v_1) = x$ for some x in $\{1, \dots, 10\}$. There are additional examples, generated using the *Labelit* program, of graceful labeling of P_n with $f(v_1) = x$ for some x in $\{1, \dots, 10\}$ in the Appendix.

1-10-2-9-3-8-4-7-5-6	6-4-3-8-5-9-2-10-1-7
2-9-1-10-4-6-5-8-3-7	7-2-10-1-8-6-3-9-5-4
3-5-10-1-9-2-8-4-7-6	8-3-5-6-2-9-1-10-4-7
4-8-5-7-6-1-10-2-9-3	9-2-10-1-7-5-6-3-8-4
5-6-2-10-1-8-3-9-7-4	10-1-9-2-8-3-7-4-6-5

FIG. 22 Examples of gracefully labeled path having first vertex labeled $f(v_1) = x$ for some $x = 1..10$.

B. Gvozdiak's Conjecture

The following conjecture comes from the references made by Rastislav Královič in his PhD thesis [Graceful Tee Labeling, Rastislav Královič, PhD, p18]. Královič quotes Gvozdiak's conjecture found on his PhD thesis regarding a graceful labeling of P_n. Gvozdiak is uses notation to indicate that if a graceful labeling $f: V(P_n) \to \{0, \dots, n\}$ assigns $f(v_0) =$ a and $f(v_n) =$ b, then such labeling is denoted as $(a, b; n)$.

Gvozdiak's Conjecture: An $(a, b; n)$ – graceful labeling exists if and only if the integers a, b, n satisfy these conditions:

- $b - a$ has the same parity as $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$
- $0 < b - a \le \frac{n}{2} \le a + b \le \frac{3n}{2}$.

Gvozdiak's paper indicates that both conditions are necessary, but it is still not proven whether they are also sufficient. Computed experiments show that this conjecture holds for all paths with $n \le 22$.

C. Gvozdiak's Conjecture Modified

We make a slight modification to Gvozdiak's conjecture fitting our model of indexing from 1 instead of 0 (i.e., $f(v_0) \to f(v_1)$) and using only the integer results, yield the following:

- $b - a$ has the same parity as $1 + 2 + \cdots + n - 1 = \frac{n(n-1)}{2}$
- $0 < |b - a| \le \frac{n}{2} \le a + b \le \lceil 3n/2 \rceil$

For example, a graceful labeling of P_7 with vertices labeled sequentially as 7, 1, 6, 2, 5, 3, 4 reveals $b - a = 4 - 7 = -3$ and $\frac{n(n-1)}{2} = \frac{42}{2} = 21$ having the same parity. And applying the modified bullet point reveals $0 < 3 \le 3 \le 11 \le 11$ which holds for $(7, 4; 7)$.

1. What is known

The first bullet point describes the necessary condition that is required to uniquely induce n-1 edge labels. In other words, the two end-vertex labels of P_n must support the n-1 number of edge labels needed to be a graceful labeling. We provide a simple example using case $n = 3$ and concerning the parity. The three graceful labeling of P_3 are $\{3, 1, 2\}$, $\{2, 3, 1\}$, and $\{1, 3, 2\}$. The parity of "$b - a$" from the three graceful labeling happens to be all odds; $2 - 3 = -1$, $1 - 2 = -1$, and $2 - 1 = 1$ respectively. For it to be an even, two end vertex labels must be of either "odd and odd" or "even and even" combination. However, the only available labeling with two end points with the same parity is the labeling $\{1, 2, 3\}$. It is an "odd and odd" combination (i.e., $3 - 1 = 2$) and is not a graceful labeling. An important fact here is that a labeling $\{1, 2, 3\}$ can only induce two odd parity edge labels (e.g., $1 - 2 = 1$ and $2 - 3 = 1$), but we need both even and odd parity edge labels (e.g., 1 and 2). Why is it equal to the permutation of the vertex-pairs available from the numbers $\{1..n\}$ is not fully understood.

The second bullet point, the condition $0 < b - a \leq \frac{n}{2}$ indicates that the two end-vertex labels of a graceful labeling must be unique and their distance cannot exceed half of *n*. An attempt was made to prove by contradiction method, supposing that there exists a graceful labeling with a two end-vertex labels meeting the requirement b-a > n/2. That is, *there is a graceful labeling $f: V(P_n) \to \{1, \ldots, n\}$ with $f(v_1) = b$ and $f(v_n) = a$ such that $b - a > \frac{n}{2}$*. A successful proof would result in contradiction. However, the proof is not available since it isn't fully understood.

Use of the *Labelit* program provides some supporting results for n up to 14 (see FIG. 23). Refer to the appendix for additional supporting results from the *Labelit* program.

Graceful Labeling [Edge Labels]	$0 < \|b - a\| \leq \frac{n}{2} \leq a + b \leq \lceil 3n/2 \rceil$	PAR(b-a) = PAR($\frac{n(n-1)}{2}$)
6-12-3-13-2-14-1-9-11-4-8-7-10-5 [6-9-10-11-12-13-8-2-7-4-1-3-5]	(0<1<=7<=11<=21)	PARITY(1) = PARITY(91)
7-12-3-13-2-14-1-9-11-4-10-6-5-8 [5-9-10-11-12-13-8-2-7-6-4-1-3]	(0<1<=7<=15<=21)	PARITY(1) = PARITY(91)
5-12-3-13-2-14-1-9-11-6-7-10-4-8 [7-9-10-11-12-13-8-2-5-1-3-6-4]	(0<3<=7<=13<=21)	PARITY(3) = PARITY(91)
8-11-3-13-2-14-1-10-4-6-7-12-5-9 [3-8-10-11-12-13-9-6-2-1-5-7-4]	(0<1<=7<=17<=21)	PARITY(1) = PARITY(91)
6-11-3-13-2-14-1-10-4-7-5-12-8-9 [5-8-10-11-12-13-9-6-3-2-7-4-1]	(0<3<=7<=15<=21)	PARITY(3) = PARITY(91)
9-11-3-13-2-14-1-10-4-8-5-12-7-6 [2-8-10-11-12-13-9-6-4-3-7-5-1]	(0<3<=7<=15<=21)	PARITY(3) = PARITY(91)
6-11-3-13-2-14-1-10-4-8-7-5-12-9 [5-8-10-11-12-13-9-6-4-1-2-7-3]	(0<3<=7<=15<=21)	PARITY(3) = PARITY(91)

FIG. 23 Labelit program generated graceful labeling of P_{14} and Gvozdiak's conjecture

IV. Graceful Labeling Application: Labelit

A recursive backtracking algorithm is used for generating gracefully labelings of paths of size *n*. It is a better method than brute force since identical partial paths are treated as a single partial solution while independently evaluating all possible solutions.

The *Labelit* program starts by initializing the starting label and the index location (user provided). There are two recursive subroutines; *GracefullyLabelLeft ()* and *GracefullyLabelRight ()* performing the backtracking algorithm. Once the selected vertex is labeled with a starting label, *GracefullyLabelRight ()* is recursively called until the last vertex of a path is reached. It then makes a subsequent call to *GracefullyLabelLeft ()* to complete the process for finding all gracefully labeled paths of size *n*.

The high level description of the algorithm is provided below. There are some opportunities for improvements in execution speed by pruning paths that do not provide solutions.

GracefullyLabelRight Recursive Function

This function finds partial solutions moving from the starting vertex location to the right most vertex.

```
$Solution = ELabels = VLabels \leftarrow \emptyset$          // Solution, edge labels, and vertex label lists
```

BOOL GracefullyLabelRight (index k)

```
1. If $index\ k = n$ then                  // Done--Reached the last right most vertex
    a. return TRUE
2. else                 // Independently, find solutions using recursive backtracking method
    a. For each $label$ from 1 to $n$ do
        i. Set $VLabels[k+1] \leftarrow label$
        ii. If IS_GRACEFUL($VLabels[k+1]$) = TRUE then
            1. $ELabels[ABS(VLabels[k] - VLabels[k+1])] \leftarrow USED$
            2. If $GracefullyLabelRight(k+1)$ then
                a. $GracefullyLabelLeft(startIndex)$
            3. $ELabels[ABS(VLabels[k] - VLabels[k+1])] \leftarrow NOTUSED$
    b. end for loop
3. end if
4. return FALSE
```

BOOL GracefullyLabelLeft (index k)

This function finds solutions moving from the starting vertex location to the left first vertex.

```
1. If $index\ k = 1$ then                  // First vertex reached--done
    a. $WriteSolution(VLabels)$
    b. $Increment\ SolutionCount$
    c. return TRUE
2. else                 // Independently, find solutions using recursive backtracking method
    a. For each $label$ from 1 to $n$ do
        i. Set $VLabels[k-1] \leftarrow label$
        ii. If IS_GRACEFUL($VLabels[k-1]$) = TRUE then
            1. $ELabels[ABS(VLabels[k] - VLabels[k-1])] \leftarrow USED$
            2. $GracefullyLabelLeft(k-1)$
            3. $ELabels[ABS(VLabels[k] - VLabels[k-1])] \leftarrow NOTUSED$
    b. end for loop
3. end if
4. return FALSE
```

Improvements

Improvements were made to the original algorithm by pruning to reduce the size of decision tree by removing sections of the tree that result in no solution. In addition slight modification to the original algorithm is made to allow the initial vertex label to be on any node of the path. See the Source Code for the *Labelit* Program in the Appendix for details.

V. References

Richard A. Brualdi, Introductory Combinatorics (4th edition): *Introduction to Graph Theory, NJ: Prentice Hall, 2004*

Koh, K. M., Rogers, D. G. & Tan, T. 1980, “Products of Graceful Trees”, Discrete Mathematics, no, 31, pp. 279-292.

Gallian, j. “Dynamic Survey of Graph Labeling.” Elec. J. Combin. 14, No. DS6, Jan. 3, 2007.

Harary, Frank; Schwenk, Allen J. (1973), “The number of caterpillars”, Discrete Mathematics 6 (4): 359-365.

Bondy, J. A; Murty, U. S. R. (1976). Graph Theory with Applications. North Holland. pp 12-21.

Golomb, S. W. 1972, ‘How to number a graph’, in Graph Theory and Computing, ed. R. C. Read, Academic Press, NY, pp. 23-37.

Chen, W. C; Lu, H. I.; and Yeh, Y. N. “Operations of Interlaced Trees and Graceful Trees.” Southeast Asian Bul. Math. 21, 337-348, 1997.

Cahit, I. & Cahit, R. 1975, “On the graceful numbering of spanning trees”, Information Processing Letters, vol. 3, no. 4, pp. 115-118

Ringel, G. 1964, ‘Problem 25’ in Theory of Graphs and its Applications, Proceedings Sympositum Smolenice, Prague.

Rosa, A. 1967, ‘On certain valuations of the vertices of graph’, in Theory of Graphs (International Symposium, Rome, July 1966), Gordon and Breach, New York, pp. 349-355.

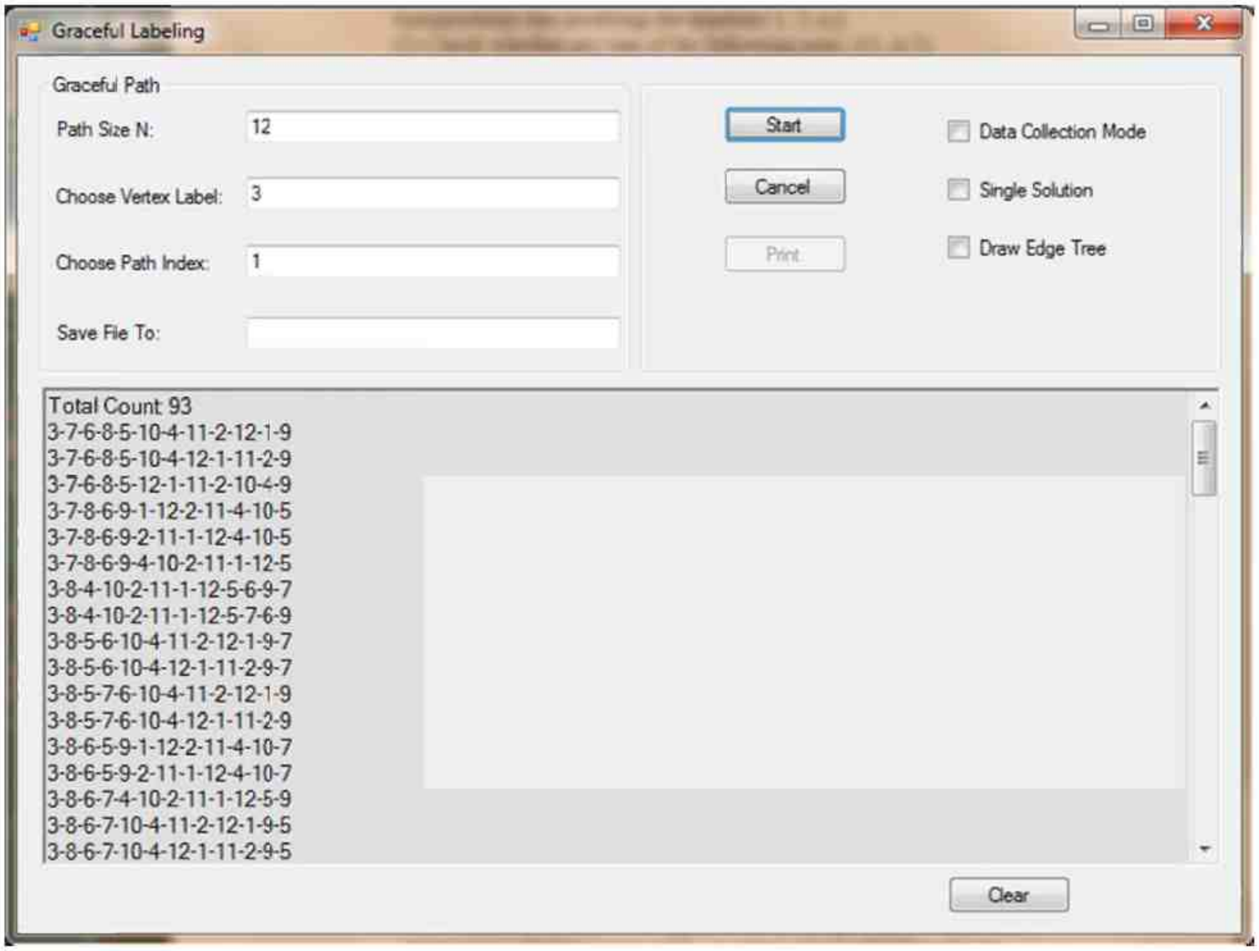
Gvozdjak, P. On the oberwolfach problem for cycles with multiple lengths, Ph.D. thesis, Simon Fraser University, 2004

Michael Horton, 2003, “Graceful Trees: Statistics and Algorithms”

VI. Appendix

Labelit Program Graphical User Interface

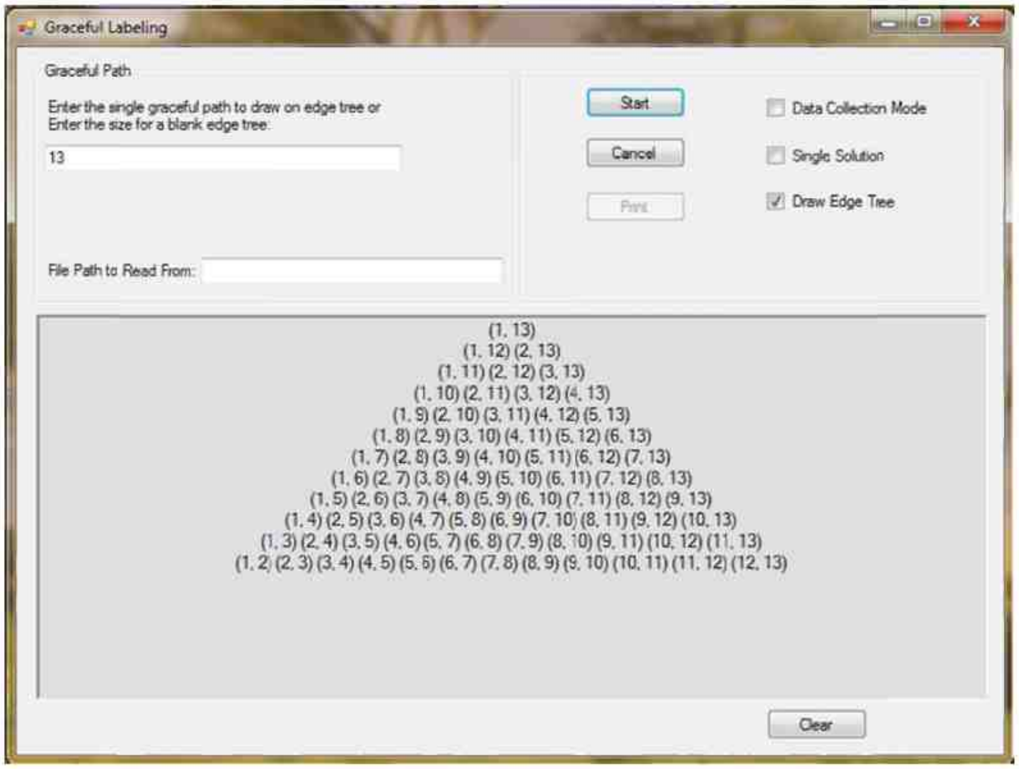
Following are the two graphical user interfaces provided for the Labelit program. The first is used for collecting input parameters from the user that is required for generating gracefully labeling of P_n (see FIG. 24). If the check box, "Draw Edge Tree" is selected, the second graphical user interface is shown (see FIG. 25). Providing a full path of the output file on the "File Path to Read From" field will draw each gracefully labeled path with its vertex-pairs selected highlighted (see FIG. 26).



Path Size N: 12
Choose Vertex Label: 3
Choose Path Index: 1
Save File To:

```
Total Count 93
3-7-6-8-5-10-4-11-2-12-1-9
3-7-6-8-5-10-4-12-1-11-2-9
3-7-6-8-5-12-1-11-2-10-4-9
3-7-8-6-9-1-12-2-11-4-10-5
3-7-8-6-9-2-11-1-12-4-10-5
3-7-8-6-9-4-10-2-11-1-12-5
3-8-4-10-2-11-1-12-5-6-9-7
3-8-4-10-2-11-1-12-5-7-6-9
3-8-5-6-10-4-11-2-12-1-9-7
3-8-5-6-10-4-12-1-11-2-9-7
3-8-5-7-6-10-4-11-2-12-1-9
3-8-5-7-6-10-4-12-1-11-2-9
3-8-6-5-9-1-12-2-11-4-10-7
3-8-6-5-9-2-11-1-12-4-10-7
3-8-6-7-4-10-2-11-1-12-5-9
3-8-6-7-10-4-11-2-12-1-9-5
3-8-6-7-10-4-12-1-11-2-9-5
```

FIG. 24 Graceful Labeling Application *Labelit* showing all possible gracefully labeled path of size 12 with its first vertex labeled 3.


Graceful Labeling

Graceful Path

Enter the single graceful path to draw on edge tree or
Enter the size for a blank edge tree:

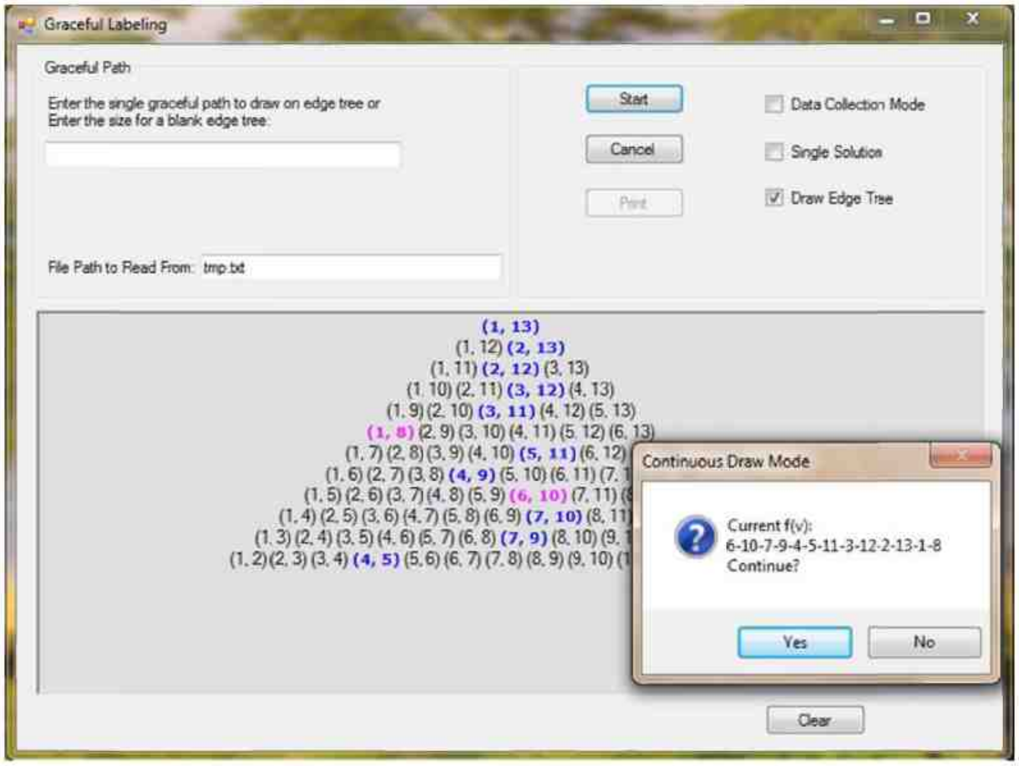
13

File Path to Read From:

☐ Data Collection Mode
☐ Single Solution
☑ Draw Edge Tree

```
                                     (1, 13)
                                 (1, 12) (2, 13)
                             (1, 11) (2, 12) (3, 13)
                         (1, 10) (2, 11) (3, 12) (4, 13)
                      (1, 9) (2, 10) (3, 11) (4, 12) (5, 13)
                   (1, 8) (2, 9) (3, 10) (4, 11) (5, 12) (6, 13)
                (1, 7) (2, 8) (3, 9) (4, 10) (5, 11) (6, 12) (7, 13)
             (1, 6) (2, 7) (3, 8) (4, 9) (5, 10) (6, 11) (7, 12) (8, 13)
          (1, 5) (2, 6) (3, 7) (4, 8) (5, 9) (6, 10) (7, 11) (8, 12) (9, 13)
       (1, 4) (2, 5) (3, 6) (4, 7) (5, 8) (6, 9) (7, 10) (8, 11) (9, 12) (10, 13)
    (1, 3) (2, 4) (3, 5) (4, 6) (5, 7) (6, 8) (7, 9) (8, 10) (9, 11) (10, 12) (11, 13)
(1, 2) (2, 3) (3, 4) (4, 5) (5, 6) (6, 7) (7, 8) (8, 9) (9, 10) (10, 11) (11, 12) (12, 13)
```

FIG. 25 Graceful Labeling Application *Labelit* drawing edge-tree diagram using size 13.


Graceful Labeling

Graceful Path

Enter the single graceful path to draw on edge tree or
Enter the size for a blank edge tree:

File Path to Read From: tmp.txt

☐ Data Collection Mode
☐ Single Solution
☑ Draw Edge Tree

```
                                     (1, 13)
                                 (1, 12) (2, 13)
                             (1, 11) (2, 12) (3, 13)
                         (1, 10) (2, 11) (3, 12) (4, 13)
                      (1, 9) (2, 10) (3, 11) (4, 12) (5, 13)
                   (1, 8) (2, 9) (3, 10) (4, 11) (5, 12) (6, 13)
                (1, 7) (2, 8) (3, 9) (4, 10) (5, 11) (6, 12)
             (1, 6) (2, 7) (3, 8) (4, 9) (5, 10) (6, 11) (7, 1
          (1, 5) (2, 6) (3, 7) (4, 8) (5, 9) (6, 10) (7, 11) (
       (1, 4) (2, 5) (3, 6) (4, 7) (5, 8) (6, 9) (7, 10) (8, 11
    (1, 3) (2, 4) (3, 5) (4, 6) (5, 7) (6, 8) (7, 9) (8, 10) (9, 1
(1, 2) (2, 3) (3, 4) (4, 5) (5, 6) (6, 7) (7, 8) (8, 9) (9, 10) (1
```

Continuous Draw Mode

Current f(v):
6-10-7-9-4-5-11-3-12-2-13-1-8
Continue?

FIG. 26 Graceful Labeling Application *Labelit* drawing edge-tree diagram with path highlighted.

Some Results Supporting Main Conjecture

$N = 7$

1-7-2-6-3-5-4	7-1-6-2-5-3-4	2-7-1-5-4-6-3
6-2-7-1-4-3-5	5-3-4-7-1-6-2	3-5-6-2-7-1-4
4-5-3-6-2-7-1		

$N = 8$

1-8-2-7-3-6-4-5	8-1-7-2-6-3-5-4	2-8-1-6-3-7-5-4
7-2-8-1-5-4-6-3	3-7-2-8-1-4-6-5	3-7-4-2-8-1-6-5
6-5-3-7-2-8-1-4	5-4-6-3-7-2-8-1	

$N = 12$

1-12-2-11-3-10-4-9-5-8-6-7	12-1-11-2-10-3-9-4-8-5-7-6
2-12-1-10-3-11-5-7-6-9-4-8	11-2-12-1-9-3-10-5-6-8-4-7
3-11-2-12-1-8-4-10-5-6-9-7	10-3-11-2-12-1-7-4-9-5-6-8
4-10-3-11-2-12-1-6-7-9-5-8	9-4-10-3-11-2-12-1-5-8-6-7
6-9-4-10-3-11-2-12-1-5-7-8	6-7-9-4-10-3-11-2-12-1-5-8
7-8-6-9-4-10-3-11-2-12-1-5	7-6-8-5-9-4-10-3-11-2-12-1

Graceful Labeling of P_{12} with $f(v_1) = x$ for some x

1-12-2-11-3-10-4-9-5-8-6-7	2-10-3-12-1-11-5-7-6-9-4-8
3-7-6-8-5-10-4-11-2-12-1-9	4-6-7-10-1-12-2-9-5-11-3-8
5-1-12-2-11-3-10-4-9-6-8-7	6-1-12-2-11-3-10-4-7-5-9-8
7-1-12-2-11-3-10-5-4-8-6-9	8-1-12-2-11-3-5-9-4-10-7-6
9-1-12-2-11-4-10-5-8-6-7-3	10-1-12-2-7-4-11-3-9-5-6-8
11-1-12-3-8-4-10-2-9-6-5-7	12-1-11-2-10-3-9-4-8-5-7-6

Source Code for Labelit Program

The *Labelit* application is written in using Microsoft Visual Studio C# 2010.

```
/******************************************************************
 * Module:          GracefulGui.cs
 * Description:     Graceful labeling application
 ******************************************************************
 * Author:         Cory Yi
 * Institution:    CSUCI
 * Date:           25-Sep-2010
 * History         Created for math thesis in graceful labeling
 ******************************************************************/
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.Drawing.Printing;
using GracefulLabelingApp;
using GracefulLableGui;
using EdgeTree;

namespace GracefulLableGui
{
    public delegate void UpdateTextDelegate(string text);

    public partial class GracefulGui : Form
    {
        private Font printFont;
        private StreamReader streamToPrint;
        private PrintDocument docToPrint = new PrintDocument();
        private string graceful_path="";
        private string filePath="";

        public GracefulGui()
        {
            InitializeComponent();
            print_btn.Enabled = false;
        }

        private void updateText(string text)
        {
            resultTextBox.AppendText(text);
        }

        private void size_box_MouseClick(object sender, EventArgs e)
        {
            sizeBox.SelectAll();
        }

        private void size_box_TextChanged(object sender, EventArgs e)
```

```
{
    if (sizeBox.Text.Length > 0)
    {
        size_n = int.Parse(sizeBox.Text);
    }
}

private void index_box_MouseClick(object sender, EventArgs e)
{
    indexBox.SelectAll();
}

private void index_box_TextChanged(object sender, EventArgs e)
{
    if (indexBox.Text.Length > 0)
    {
        index = int.Parse(indexBox.Text);
    }
}

private void fname_box_TextChanged(object sender, EventArgs e)
{
    if (pathBox.Text.Length > 0)
    {
        filePath = pathBox.Text;
    }
}

private void labelBox_MouseClick(object sender, EventArgs e)
{
    labelBox.SelectAll();
}

private void labelBox_TextChanged(object sender, EventArgs e)
{
    if (drawTreeBox.Checked.Equals(true))
    {
        graceful_path = labelBox.Text;
    }
    else
    {
        if (labelBox.Text.Length > 0 && labelBox.TextLength < 3)
        {
            vertex = int.Parse(labelBox.Text);
        }
        else
        {
            MessageBox.Show("Enter size length < 50");
            labelBox.Clear();
        }
    }
}

private void start_btn_Click(object sender, EventArgs e)
{
    int length, idx;
    string EdgeeTreeString="";
    string path = @"C:\Users\Cory\Documents\Thesis\temp\";
```

```
/* Clear previous results if any */
resultTextBox.Clear();

/*********************************************
 * Edge Tree Drawing option
 *********************************************/
if (drawTreeBox.Checked.Equals(true))
{
    /* disable other options */
    if ((labelBox.TextLength > 0) || pathBox.TextLength > 0)
    {
        graceful_path = labelBox.Text;

        /* Backdoor continuous file read option */
        if (String.Compare(graceful_path, "file") == 0)
        {
            try
            {
                if (pathBox.TextLength > 0)
                {
                    path = pathBox.Text;
                }

                var files = from file in
                            Directory.EnumerateFiles(@path)
                            select file;

                /* Process each files in this directory */
                foreach (var file in files)
                {
                    using (FileStream fs = File.OpenRead(file))
                    {
                        foreach (string line in File.ReadLines(file))
                        {
                            if (line.Contains(':'))
                            {
                                graceful_path = line.Substring(0,
                                                line.IndexOf(':'));
                            }
                            else
                            {
                                graceful_path = line;
                            }

                            EdgeeTreeString =
                               DrawEdgeTree.DrawTree(graceful_path);
                            resultTextBox.SelectionAlignment =
                               HorizontalAlignment.Center;
                            resultTextBox.AppendText(EdgeeTreeString);
                            ColorEdgeTree(graceful_path, EdgeeTreeString);

                            var result = MessageBox.Show(graceful_path,
                                            "Continuous Draw Mode",
                                            MessageBoxButtons.YesNo,
                                            MessageBoxIcon.Question);
                            // If the no button was pressed ...
                            if (result == DialogResult.No)
                            {
                                return;
                            }
                            resultTextBox.Clear();
                            break;
                        }
                        fs.Close();
                    }

                }
            }
            catch (UnauthorizedAccessException UAEx)
            {
                Console.WriteLine(UAEx.Message);
            }
            catch (PathTooLongException PathEx)
            {
                Console.WriteLine(PathEx.Message);
            }
        }
        else if ((pathBox.TextLength > 0) && (labelBox.TextLength == 0))
        {
            foreach (string line in File.ReadLines(pathBox.Text))
            {
                graceful_path = line;

                EdgeeTreeString = DrawEdgeTree.DrawTree(graceful_path);
                resultTextBox.SelectionAlignment = HorizontalAlignment.Center;
                resultTextBox.AppendText(EdgeeTreeString);
                ColorEdgeTree(graceful_path, EdgeeTreeString);

                var result = MessageBox.Show(String.Concat("Current f(v):\n",
                                             graceful_path, "\nContinue?"),
                                             "Continuous Draw Mode",
                                             MessageBoxButtons.YesNo,
                                             MessageBoxIcon.Question);
                // If the no button was pressed ...
                if (result == DialogResult.No)
                {
                    return;
                }
                resultTextBox.Clear();
            }

        }
        else
        {
            /* Normal operation */
            EdgeeTreeString = DrawEdgeTree.DrawTree(graceful_path);
            resultTextBox.SelectionAlignment = HorizontalAlignment.Center;
            resultTextBox.AppendText(EdgeeTreeString);

            ColorEdgeTree(graceful_path, EdgeeTreeString);
        }
    }
}
/*********************************************
 * Data Collection mode of operation
 ****************************************************/
else if (dataModeBox.Checked.Equals(true))
{
    for (length = 4; length < 20; length++)
    {
        for (idx = 1; idx <= length; idx++)
        {
            string file_name;

            /* Set current n and index */
            size_n = length;
            index = idx;

            /* Create current file name */
            file_name = pathBox.Text;
            file_name = String.Concat(file_name, size_n.ToString(), "_",
                                          idx.ToString(), ".txt");

            Thread gracefulThread = new Thread(GracefullyLabel);
            gracefulThread.Start();
        }

    } // end for
}
/****************************************************
 * Normal Operation option
 ****************************************************/
else
{
    resultTextBox.SelectionAlignment = HorizontalAlignment.Left;

    if (size_n < index)
    {
        resultTextBox.AppendText("Enter index <= n");
        return;
    }
    else if (size_n > 14)
    {
        const string message =
            "Chain Size > 14 may take awhile. Do you want to continue?";
        const string caption = "Form Graceful Labeling";
        var result = MessageBox.Show(message, caption,
                                     MessageBoxButtons.YesNo,
                                     MessageBoxIcon.Question);

        // If the no button was pressed ...
        if (result == DialogResult.No)
        {
            return;
        }
    }
    else if (size_n < 1)
    {
        MessageBox.Show("Enter chain length n > 1");
        return;
    }
    else if (vertex > size_n)
    {
            MessageBox.Show("Enter vertex label <= n");
            return;
        }

        // Start the graceful labeling
        Thread gracefulThread = new Thread(GracefullyLabel);
        gracefulThread.Start();

    } // end if

}

private void ColorEdgeTree(string path, string tree)
{
    string node = "", digitStr = "";
    bool dashSeen = false, firstNode = false, lastNode = false;
    int firstNum=0, secondNum=0, digitNum;

    path = String.Concat(path, "\n");

    /* Get the first vertex */
    foreach (char curChar in path)
    {
        if (('-' == curChar)     (curChar == '\n'))
        {
            if (curChar == '\n')
            {
                lastNode = true;
            }

            digitNum = int.Parse(digitStr);
            digitStr = "";

            if (dashSeen)
            {
                /* 2-5-4-6-1-7-3 */
                secondNum = digitNum;

                if (secondNum < firstNum)
                {
                    int temp;

                    temp = secondNum;
                    secondNum = firstNum;
                    firstNum = temp;
                }

                /* Build node string and search */
                node = String.Concat("(", firstNum.ToString(), ", ",
                                     secondNum.ToString(), ")");

                /* Find and color the node from the tree */
                resultTextBox.Find(node, 0);
                resultTextBox.SelectionFont = new Font("Verdana", 8,
                                                       FontStyle.Bold);

                if (firstNode    lastNode)
                {
                        resultTextBox.SelectionColor = Color.Magenta;
                        firstNode = false;
                    }
                    else
                    {
                        resultTextBox.SelectionColor = Color.Blue;
                    }

                    /* Reinit node string */
                    node = "";
                    firstNum = digitNum;
                }
                else
                {
                    firstNum = digitNum;
                    firstNode = true;
                }
                dashSeen = true;
            }
            else
            {
                /* Build digit until dash is seen */
                digitStr = String.Concat(digitStr, curChar.ToString());
            }

        } /* end foreach */

    }


    private void cancel_btn_Click(object sender, EventArgs e)
    {
        DialogResult resp = MessageBox.Show("Are you sure you want to cancel?",
                        "Graceful Label", MessageBoxButtons.YesNo);
        if (resp == System.Windows.Forms.DialogResult.Yes)
        {
            GracefulLabeling.CancelGracefulPath();
        }
    }

    /************************************************************
     * Invoked by start button press
     ************************************************************/
    private void GracefullyLabel()
    {
        int total_count;
        string tempFile=@".\tempfile.txt";

        /* Perform the graceful label algorithm and write the result to temp file */
        StreamWriter sw = File.CreateText(tempFile);

        GracefulLabeling.SetGracefulParam(sw, solBox.Checked.Equals(true));
        total_count = GracefulLabeling.FindLabeledPath(size_n, vertex, index);

        if (dataModeBox.Checked.Equals(true))
        {
            sw.Write(total_count.ToString());
            sw.WriteLine();
        }

        sw.Close(); /* Done writing the result file - close it */

        /* Read from the temp file created above and Write results to the window */
        if (dataModeBox.Checked.Equals(false))
        {
            string line = "";
            using (StreamReader sr = new StreamReader(tempFile))
            {
                SetText("Total Count: ");
                SetText(total_count.ToString());
                SetText("\n");

                while ((line = sr.ReadLine()) != null)
                {
                    SetText(line);
                    SetText("\n");
                }
                /* Close and remove the temp file */
                sr.Close();
                if (filePath.Length > 0)
                {
                    try
                    {
                        File.Copy(tempFile, filePath,true);
                    }
                    catch (IOException UAEx)
                    {
                        MessageBox.Show("Problem encountered with the specified file
                                        path");
                    }

                }
                File.Delete(tempFile);
            }
        }
    }

    private void SetText(string text)
    {
        // InvokeRequired required compares the thread ID of the
        // calling thread to the thread ID of the creating thread.
        // If these threads are different, it returns true.
        if (resultTextBox.InvokeRequired)
        {
            UpdateTextDelegate d = new UpdateTextDelegate(SetText);
            Invoke(d, new object[] { text });
        }
        else
        {
            resultTextBox.AppendText(text);
        }
    }

    private void clear_btn_Click(object sender, EventArgs e)
    {
        resultTextBox.Clear();
```

```
}

private void print_btn_Click(object sender, EventArgs e)
{
    /* Allow page ranges to print */
    printDialog1.AllowSomePages = true;

    /* Show the help button */
    printDialog1.ShowHelp = true;

    printDialog1.Document = docToPrint;

    DialogResult result = printDialog1.ShowDialog();

    // If the result is OK then print the document.
    if (result == DialogResult.OK)
    {
        streamToPrint = new StreamReader(filePath.ToString());
        try
        {
            docToPrint.Print();
        }
        finally
        {
            streamToPrint.Close();
        }
    }

}

// The PrintPage event is raised for each page to be printed.
private void document_PrintPage(object sender, PrintPageEventArgs e)
{
    float linesPerPage = 0;
    float yPos = 0;
    int count = 0;
    float leftMargin = e.MarginBounds.Left;
    float topMargin = e.MarginBounds.Top;
    string line = null;

    // Calculate the number of lines per page.
    linesPerPage = e.MarginBounds.Height /
       printFont.GetHeight(e.Graphics);

    // Print each line of the file.
    while (count < linesPerPage &&
       ((line = streamToPrint.ReadLine()) != null))
    {
        yPos = topMargin + (count *
           printFont.GetHeight(e.Graphics));
        e.Graphics.DrawString(line, printFont, Brushes.Black,
           leftMargin, yPos, new StringFormat());
        count++;
    }

    // If more lines exist, print another page.
    if (line != null)
        e.HasMorePages = true;
    else
        e.HasMorePages = false;
}

private void GracefulGui_Load(object sender, EventArgs e)
{

}

private void solBox_CheckedChanged(object sender, EventArgs e)
{

}

private void drawTreeBox_CheckedChanged(object sender, EventArgs e)
{
    if (drawTreeBox.Checked.Equals(true))
    {
        label1.Text = "Enter the single graceful path to draw on edge tree or
                       \nEnter the size for a blank edge tree:";
        label3.Text = "File Path to Read From:";
        sizeBox.Hide();
        indexBox.Hide();
        label.Hide();
        label2.Hide();
        labelBox.Width = 270;
        labelBox.Location = new Point(21,74);
        labelBox.Focus();
    }
```

```
            }
            else
            {
                label1.Text = "Chain Size:";
                label3.Text = "File Path to Save To:";
                sizeBox.Show();
                indexBox.Show();
                label.Show();
                label2.Show();
                labelBox.Width = 229;
                labelBox.Location = new Point(139, 74);
                sizeBox.Focus();
            }
            labelBox.Clear();
        }


    }
}

/*****************************************************************
 * Module:          GracefullyLabel.cs
 * Description:     Graceful labeling application
 *****************************************************************
 * Author:          Cory Yi
 * Institution:     CSUCI
 * Date:            25-Sep-2010
 * History          Created for Graceful Labeling thesis
 *****************************************************************/
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace GracefulLabelingApp
{

    public class GracefulLabeling
    {
        public static StreamWriter fhandle;
        public static List<int> vLabels = new List<int>();  // Used to hold the current
                                                            // graceful labeling solution
        public static List<int> eLabels = new List<int>();  // Used to mark the edge
                                                            // length currently used
        public static List<int> mLabels = new List<int>();  // Used to mark the edge
                                                            // lengths used
        public static int start_index;
        public static int start_vertex;
        public static int size_n;
        public static int start_depth;
        public static int total_count;
        public static bool do_single;
        public static bool done, leafNode;
        public static EdgeLabelList eList;

        public static void SetGracefulParam(StreamWriter sw, bool single_sol)
        {
            fhandle = sw;
            do_single = single_sol;
            done = false;
        }

        /****************************************************************
         * Traverses the left and right tree and finds the traceful
         * path with given path size and the position index for label 1
         ****************************************************************/
        public static int FindLabeledPath(int n, int v, int i)
        {

            /* Validate input parameters */
            if (i > n)
            {
                return 0;
            }

            /* Initialize the local lists */
            for (int j=0; j < n + 1; j++)
            {
                vLabels.Add(0);
                eLabels.Add(0);
            }

            /* Initialize the vertex and its position */
            size_n = n;
            start_vertex = v;
            start_index = i;
    start_depth = n - 1;
    total_count = 0;

    /* Construct edge label list */
    eList = new EdgeLabelList(size_n);

    /***************************************************
     * Get the direction which to populate the solution.
     ***************************************************/
    if (i.Equals(n))
    {
        /* index = n, so check e.g. "...x - 9 - 1" */
        vLabels[n] = start_vertex;
        GracefullyLabelLeft(n);
    }
    else
    {
        /* index = 1, so check e.g. "1 - 9 - x,... " */
        vLabels[start_index] = start_vertex;
        GracefullyLabelRight(start_index);
    }

    vLabels.Clear();
    eLabels.Clear();

    return total_count;
}

/*****************************************************************
 *                     GracefullyLabelLeft
 *****************************************************************
 * Recursively builds graceful path moving to right & left
 *****************************************************************/
public static bool GracefullyLabelLeft(int k)
{
    /* Moving from right to left (negative direction) */
    if (k.Equals(1))
    {
        /* We have found a solution, save to file */
        WriteToFile(fhandle, vLabels);
        total_count++;
        if (do_single)
        {
            done = true;
        }
        return true;
    }
    else
    {
        /* Recursively search for all gracefully labeled path. */
        for (vLabels[k - 1] = 1; vLabels[k - 1] <= size_n; vLabels[k - 1]++)
        {
            if (ValidateGracefulLeft(k - 1))
            {
                // Mark it as graceful and move on
                eLabels[Math.Abs(vLabels[k] - vLabels[k - 1])] = -1;
                GracefullyLabelLeft(k-1);
                eLabels[Math.Abs(vLabels[k] - vLabels[k - 1])] = 0;
            }
        } /* end for */

        /* At this point, we have tried all the labels, now back track */

    } /* end if */

    return false;
}

/*****************************************************************
 *                      GracefullyLabelRight
 *****************************************************************
 * Recursively builds graceful path moving to right & left
 *****************************************************************/
public static bool GracefullyLabelRight(int k)
{
    // Moving from left to right (positive direction)
    if (k.Equals(size_n))
    {
        // Indicate we are done moving positive direction
        return true;
    }
    else
    {
        // Recursively search for all gracefully labeled path.
        for (vLabels[k + 1] = 1; (vLabels[k + 1] <= size_n) &&  done;
             vLabels[k + 1]++)
        {
            if (ValidateGracefulRight(k + 1))
            {
                // Mark it as graceful and move on
                eLabels[Math.Abs(vLabels[k] - vLabels[k + 1])] = -1;

                // If we are done with the right side, do the left side
                if (GracefullyLabelRight(k + 1))
                {
                    GracefullyLabelLeft(start_index);
                }

                eLabels[Math.Abs(vLabels[k] - vLabels[k + 1])] = 0;
            }
        } // end for

    } // end if

    return false;
}


/*****************************************************************
 *                      ValidateGracefulRight
 *****************************************************************
 * Validates if newly added to the right is graceful
 *****************************************************************/
public static bool ValidateGracefulRight(int k)
{
    /* Indicate if working with leaf vertex */
```

```
            if (k.Equals(size_n))
            {
                leafNode = true;
            }
            else
            {
                leafNode = false;
            }

            int edge = Math.Abs(vLabels[k - 1] - vLabels[k]);

            /* To be graceful, each edge label must be used only once */
            if (eLabels[edge] != 0)
            {
                return false;
            }

            /* To be graceful, each Vertex label must be used only once */
            for (int i = 1; i < k; i++)
            {
                if (vLabels[i] == vLabels[k])
                {
                    return false;
                }
            }
#if false
            /* Scan through the edge list for pruning */
            for (int vertex = k; vertex < size_n+1; vertex++)
            {
                /* Get edges for current vertex */
                int edgeCount = eList.getEdgeCount(vertex);

                /* Check if edge is in the list */
                if (eList.edgeInList(vertex, edge))
                {
                    switch (edgeCount)
                    {
                        case 1:
                            /* This case should only occur for leaf nodes */
                            if (!leafNode)
                            {
                                return false;
                            }
                            break;
                        case 2:
                            if (vLabels[k] != vertex)
                            {
                                return false;
                            }
                            break;
                        default:
                            break;
                    } /* end switch */
                } /* end if */
            } /* end for */

            /* Remove the used edge label from the list */
            eList.removeEdge(edge);
```

```
#endif
            return true;
        }

        /*****************************************************************
         *                          ValidateGracefulLeft
         *****************************************************************
         * Validates if newly added to the left is graceful
         *****************************************************************/
        public static bool ValidateGracefulLeft(int k)
        {
            // Note that we will need to scan from the top since here
            // we are assuming that the right side is done
            if (eLabels[Math.Abs(vLabels[k + 1] - vLabels[k])] != 0)
            {
                return false;
            }

            for (int i = k + 1; i <= size_n; i++)
            {
                if (vLabels[i] == vLabels[k])
                {
                    return false;
                }
            } // end for

            return true;
        }

        /*****************************************************************
         *                              WriteToFile
         *****************************************************************
         * Writes the result out to the output file
         *****************************************************************/
        public static void WriteToFile(StreamWriter fhnd, List<int> result)
        {
            for (int i = 1; i < result.Count(); i++)
            {
                fhnd.Write(result[i].ToString());

                if (i < result.Count()-1)
                {
                    fhnd.Write('-');
                }
            }

            fhnd.WriteLine();
        }

        /*****************************************************************
         *                          CancelGracefulPath
         *****************************************************************
         * Cancels previous gracefully labeling request
         *****************************************************************/
        public static void CancelGracefulPath()
        {
            done = true;
        }
```

```
} /* end class GracefulLabeling */


/*********************************************************************
 * Class EdgeLabelList
 *--------------------------------------------------------------------
 * This Generates the edges labels for each vertices and keeps track
 * of the list and removes it from the list as it is being used in
 * the construction of the graceful labeling path
 *********************************************************************/
public class EdgeLabelList
{
    static List<int>[] edgeLabels;
    static int mid_point;

    /* EdgeLabelList Constructor */
    public EdgeLabelList(int n)
    {
        edgeLabels = new List<int>[n + 1];
        mid_point = (int)Math.Ceiling(n / 2.0);

        /* Build edge labels for each vertex labels */
        for (int j=1; (j < n + 1); j++)
        {
            edgeLabels[j] = new List<int>();

            if (j <= mid_point)
            {
                for (int i = 1; i < Math.Abs(n - j)+1; i++)
                {
                    edgeLabels[j].Add(i);
                }
            }
            else
            {
                for (int i = 1; i < Math.Abs(j-1) + 1; i++)
                {
                    edgeLabels[j].Add(Math.Abs(i));
                }
            }

        }

    }

    public bool edgeInList(int vertex, int edge)
    {
        if (edgeLabels[vertex].Contains(edge))
        {
            return true;
        }

        return false;
    }

    public int getEdgeCount(int vertex)
    {
            return edgeLabels[vertex].Count;
        }

        public void removeEdge(int edge)
        {
            for (int v = 1; v < edgeLabels.Count(); v++)
            {
                edgeLabels[v].Remove(edge);
            } /* end for */
        }

    }

}
```