

On Computing the Balance Index Sets and the Product-Cordial Index Sets of Cubic Trees

Ping-Tsai Chung

Department of Computer Science
Long Island University
Brooklyn, NY 11201, USA

Sin-Min Lee

Department of Computer Science
San Jose State University
San Jose, CA 95192, USA

In memory of our colleague - Prof. Michael L. Gargano.

Abstract Let G be a graph with vertex set $V(G)$ and edge set $E(G)$, a labeling $f: V(G) \rightarrow Z_2$ induces an edge partial labeling $f^*: E(G) \rightarrow Z_2$ defined by $f^*(xy) = f(x)$, if and only if $f(x)=f(y)$ for each edge $xy \in E(G)$. For $i \in A$, let $v_f(i) = \text{card}\{v \in V(G) : f(v) = i\}$ and $e_f(i) = \text{card}\{e \in E(G) : f^*(e) = i\}$. A labeling f of a graph G is said to be **friendly** if $|v_f(0) - v_f(1)| \leq 1$. If $|e_f(0) - e_f(1)| \leq 1$ then G is said to be **balanced**. The **balance index set** of the graph G , $BI(G)$, is defined as $\{|e_f(0) - e_f(1)| : \text{the vertex labeling } f \text{ is friendly}\}$.

A labeling $f: V(G) \rightarrow Z_2$ induces an edge labeling $f^*: E(G) \rightarrow Z_2$ defined by $f^*(xy) = f(x) \otimes f(y)$, for each edge $xy \in E(G)$. For $i \in Z_2$, let $v_f(i) = \text{card}\{v \in V(G) : f(v) = i\}$ and $e_f(i) = \text{card}\{e \in E(G) : f^*(e) = i\}$. A labeling f of a graph G is said to be **friendly** if $|v_f(0) - v_f(1)| \leq 1$, and G is said to be **product-cordial** if, in addition, $|e_f(0) - e_f(1)| \leq 1$. The **product-cordial index set** of the graph G , $PCI(G)$, is defined as $\{|e_f(0) - e_f(1)| : \text{the vertex labeling } f \text{ is friendly}\}$. A tree is called **cubic** if all its internal vertices have degree 3. Analytic results parallel to the concept of computing the balance index sets, the product-cordial index sets of cubic trees are presented.

Key words: Friendly labeling, balance index set, product-cordial index set, strongly balanced.

AMS 2000 MSC: 05C78, 05C25

1. Introduction

S. M. Lee et al [17] considered a new labeling problem of graph theory in 1992. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$. A vertex labeling of G is a mapping f from $V(G)$ into the set $\{0, 1\}$. For each vertex labeling f of G , we can define a partial edge labeling f^* of G in the following way. For each edge $(u, v) \in E(G)$, where $u, v \in V(G)$, we have $f^*(u, v) = 1$ if $f(u) = f(v) = 1$, and $= 0$ if $f(u) = f(v) = 0$. Note that if $f(u) \neq f(v)$, the edge (u, v) is not labeled by f^* . Thus f^* is a partial function from $E(G)$ into the set $\{0, 1\}$, and we shall refer to f^* as the induced partial edge labeling. For $i = 0, 1$, let $v_f(i) = |\{v \in V(G) \mid f(v) = i\}|$ and $e_f(i) = |\{e \in E(G) \mid f^*(e) = i\}|$.

With these notations, we now introduce the notion of a balanced graph in Definition 1 and Definition 2.

Definition 1.1. A graph G is said to be a **balanced** graph or G is **balanced** if there is a vertex labeling f of G satisfying $|v_f(0) - v_f(1)| \leq 1$ and $|e_f(0) - e_f(1)| \leq 1$.

Remark 1.1: We will use $v(0), v(1), e(0), e(1)$ instead of the more complicated $v_f(0), v_f(1), e_f(0), e_f(1)$, when the context is clear. A graph G is said to be **strongly vertex-balanced** if G is balanced and $v(0) = v(1)$. Similarly, a graph G is **strongly edge-balanced** if it is balanced and $e(0) = e(1)$. If G is a strongly vertex-balanced and strongly edge-balanced graph, then we say that G is a **strongly balanced** graph.

Definition 1.2. The **balance index set** of a graph G , $BI(G)$, is defined as $\{|e_f(0) - e_f(1)| : \text{the vertex labeling } f \text{ is friendly}\}$.

Example 1. $BI(G) = \{0, 1, 2\}$, and G is balanced since $0 \in BI(G)$.

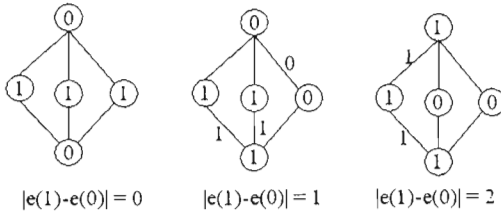


Figure 1.

Balanced labeling problem is similar to the cordial labeling of graphs which was introduced by Cahit [2]. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$, a labeling $f : V(G) \rightarrow \mathbb{Z}_2$ induces an edge partial labeling $f^* : E(G) \rightarrow \mathbb{Z}_2$ defined by $f^*(xy) = f(x)$, if and only if $f(x) = f(y)$ for each edge $xy \in E(G)$. For $i \in A$, let $v_f(i) = \text{card}\{v \in V(G) : f(v) = i\}$ and $e_f(i) = \text{card}\{e \in E(G) : f^*(e) = i\}$.

A labeling f of a graph G is said to be **friendly** if $|v_f(0) - v_f(1)| \leq 1$. If $|e_f(0) - e_f(1)| \leq 1$ then G is said to be **balanced**. The **balance index set** of the graph G , $BI(G)$, is defined as $\{|e_f(0) - e_f(1)| : \text{the vertex labeling } f \text{ is friendly}\}$.

A labeling $f : V(G) \rightarrow \mathbb{Z}_2$ induces an edge labeling $f^* : E(G) \rightarrow \mathbb{Z}_2$ defined by $f^*(xy) = f(x) \otimes f(y)$, for each edge $xy \in E(G)$. For $i \in \mathbb{Z}_2$, let $v_f(i) = \text{card}\{v \in V(G) : f(v) = i\}$ and $e_f(i) = \text{card}\{e \in E(G) : f^*(e) = i\}$. A labeling f of a graph G is said to be **friendly** if $|v_f(0) - v_f(1)| \leq 1$, and G is said to be **product-cordial** if, in addition, $|e_f(0) - e_f(1)| \leq 1$. The **product-cordial index set** of the graph G , $PCI(G)$, is defined as $\{|e_f(0) - e_f(1)| : \text{the vertex labeling } f \text{ is friendly}\}$.

Example 2. The graph $P_2 \cup C_3$ is not product-cordial, for its $PCI(P_2 \cup C_3) = \{2, 4\}$.

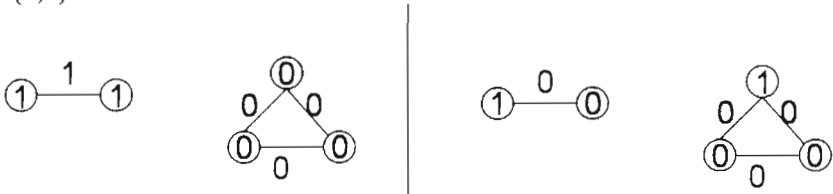
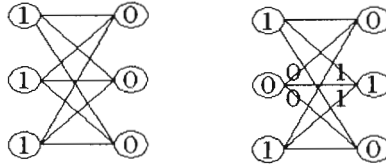


Figure 2.

Example 3. $K_{3,3}$ is balanced since $BI(K_{3,3}) = \{0\}$.



$$|e(1) - e(0)| = 0$$

Figure 3.

However, $K_{3,3}$ is not product-cordial since $PCI(K_{3,3}) = \{5, 9\}$.

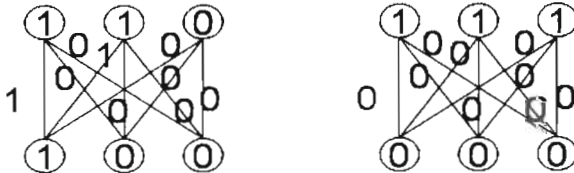


Figure 4.

A vertex in a tree with degree one is called a *pendant vertex* or a *leaf*. A tree is called *cubic* if all its internal vertices are of degree 3. We determine the BI, PCI sets of cubic trees in this paper. Results parallel to the concept of **balance index sets**, product-cordial index sets of cubic trees are presented.

2. Balanced Index Sets of Cubic Trees

Shiu and Kwong [26] introduced an algebraic approach for finding balance index sets. Given a vertex labeling $f: V \rightarrow \{0,1\}$ of a graph $G = (V, E)$, define an associated vertex labeling $g: V \rightarrow \{-0.5, 0.5\}$ by $g = f - 0.5$. It induces an edge labeling $g^*: E \rightarrow \{-1, 0, 1\}$ defined by $g^*(uv) = g(u) + g(v)$ for every $uv \in E$. In this way, every edge is labeled under g .

Let $v_g(i)$ be the number of vertices of G that are labeled i under g and $e_g(i)$ be the number of edges of G that are labeled i under g , where $i \in \{0, 1\}$. Then $g^*(e) = -1$ means $f^*(e) = 0$, $g^*(e) = 1$ means $f^*(e) = 1$, and $g^*(e) = 0$ means the edge e is balanced. Note that f is friendly (i.e., g is a friendly vertex labeling of G) if and only if $|v_g(-0.5) - v_g(0.5)| \leq 1$ (i.e., $|v_f(0) - v_f(1)| \leq 1$).

The balance index set of the graph G under f , $BI(G) = BI_f(G)$, is defined as $\{|e_f(0) - e_f(1)| : \text{the vertex labeling } f \text{ is friendly}\}$. The balance index set of G under g , $BI_g(G)$, is defined as $\{|e_g(1) - e_g(-1)| : \text{the vertex labeling } g \text{ is friendly}\}$. Thus G is balanced if and only if there is a friendly labeling g such that $|BI_g(G)| \leq 1$. That is, G is balanced if and only if $BI_g(G) \cap \{-1, 0, 1\} \neq \emptyset$. A graph is called **strongly balanced** if $|BI_g(G)| = \{0\}$ for any friendly labeling g .

Lemma 2.1. [26] Let g be any friendly labeling of a graph $G = (V, E)$, then $BI_g(G) = \sum_{e \in E} g^*(e) = \sum_{u \in V} \deg(u)g(u)$, where $\deg(u)$ is the degree sequence of $u \in V$.

A graph is **(r, s)-regular** ($r \neq s$) if the degree of each vertex is either r or s . A graph is **biregular** if it is (r, s) -regular for some distinct integers r and s .

The simplest biregular graphs are paths and stars. Each of them is (r, l) -regular for some $r \geq 2$. A **cubic tree** is (r, l) -regular, where $r=3$. Let $G = (V, E)$ be (r, s) -regular graph and g a labeling of G . From Lemma 2.1, we could find

$$BI_g(G) = r \sum_{\substack{u \in V \\ \deg(u)=r}} g(u) + s \sum_{\substack{v \in V \\ \deg(v)=s}} g(v) \quad (2.1)$$

For any friendly labeling g of a graph G , and let $v_{g,k}^+ = v_{g,k}(0.5)$ and $v_{g,k}^- = v_{g,k}(-0.5)$ be the number of vertices of degree k in G that are assigned positive label 0.5 and negative label -0.5, respectively. Therefore, for computing balance index sets of cubic trees, we have

$$BI_g(G) = \left| 3 \sum_{\substack{u \in V \\ \deg(u)=3}} g(u) + \sum_{\substack{v \in V \\ \deg(v)=1}} g(v) \right| \quad (2.2)$$

$$= \left[\left[\frac{3}{2}(v_{g,3}^+ - v_{g,3}^-) + \frac{1}{2}(v_{g,1}^+ - v_{g,1}^-) \right] \right],$$

where $\sum_{\substack{u \in V \\ \deg(u)=3}} g(u) = \frac{1}{2}(v_{g,3}^+ - v_{g,3}^-)$ and $\sum_{\substack{v \in V \\ \deg(v)=1}} g(v) = \frac{1}{2}(v_{g,1}^+ - v_{g,1}^-)$.

Remark 2.1. For any friendly labeling f of a graph G , and let $v_{f,k}^1 = v_{f,k}(1)$ and $v_{f,k}^0 = v_{f,k}(0)$ be the number of vertices of degree k in G that are assigned positive label 1 and negative label 0, respectively. Then we define an associated vertex labeling $g : V \rightarrow \{-0.5, 0.5\}$, by $g = f - 0.5$, which is also a friendly labeling of G , and let $v_{g,k}^+ = v_{g,k}(0.5)$ and $v_{g,k}^- = v_{g,k}(-0.5)$ be the number of vertices of degree k in G that are assigned positive label 0.5 and negative label -0.5, respectively. Therefore,

$$v_{f,k}^0 = v_{g,k}^- \quad \text{and} \quad v_{f,k}^1 = v_{g,k}^+ \quad (2.3)$$

and by (2.2) we have

$$BI(G) = BI_g(G) = \left[\left[\frac{3}{2}(v_{g,3}^+ - v_{g,3}^-) + \frac{1}{2}(v_{g,1}^+ - v_{g,1}^-) \right] \right]$$

$$= BI_f(G) = \left[\left[\frac{3}{2}(v_{f,3}^1 - v_{f,3}^0) + \frac{1}{2}(v_{f,1}^1 - v_{f,1}^0) \right] \right] \quad (2.4)$$

Lemma 2.2. If T is a cubic tree with n internal vertices, then the number of leaves of $T, L(T)$, is $n+2$.

Proof. We use the prove by Induction method to show the Lemma.

Basis: If $n = 1$, then T is $St(3)$, it has $n+2 = 3$ leaves. Suppose that T is a cubic tree with $n-1$ internal vertices, then $L(T) = (n-1)+2 = n+1$.

Induction Step: We consider T' , which is transformed from T by converting one leaf to an internal vertex with degree three, has $(n-1)+1 = n$ internal vertices, then $L(T') = [(n+1)-1]+2 = n+2$. !!

For the following, we develop four *data structures* for computing BI sets of cubic trees. There are two main purposes for our discussion.

First, we would like to compare the relationships between the BI computations (in section 2) and the PCI computations (in section 3); For each algorithm, it generates all possible bi's (or pci's) and their corresponding edge labeling information.

Second, we would like to set up a framework so that we could extend the computation work for any arbitrary graph topology easily.

Data Structures –

(1) An ***adjacency matrix, Adj***, for the graph $G=(V,E)$ is an $n \times (2n+2)$ matrix such that $Adj(i,j) = 1$ if $(i,j) \in E$ and $Adj(i,j) = 0$, otherwise.

(2) A ***vertex labeling list, Vertex-Labeling***, for the graph $G=(V,E)$ is an array of size $2n+2$ such that $Vertex-Labeling(i) = 1$ if the vertex specified by the i th position of $Vertex-Labeling$ has a label 1 and $Vertex-Labeling(i) = 0$, if the vertex specified by the i th position of $Vertex-Labeling$ has a label 0.

(3) An ***edge labeling matrix, Edge-Labeling***, for the graph $G=(V,E)$ is an $n \times (2n+2)$ matrix such that $Edge-Labeling(i,j) = 1$ if $Adj(i,j) = 1$ and both the $Vertex-Labeling(i) = Vertex-Labeling(j) = 1$; $Edge-Labeling(i,j) = 0$ if $Adj(i,j) = 1$ and both the $Vertex-Labeling(i) = Vertex-Labeling(j) = 0$; $Edge-Labeling(i,j)$ is not defined, if $Adj(i,j) = 1$ and the $Vertex-Labeling(i) \neq Vertex-Labeling(j)$.

(4) A ***Friendly Vertex Labeling Matrix*** with size $(2n+2) \times C_1$, where

$$C_1 = C \binom{2n+2}{n+1} = \frac{(2n+2)!}{(n+1)!(n+1)!} \text{ which is the maximum number of Friendly}$$

Vertex Labelings of T since T has $2n+2$ vertices, a friendly labeling f of T when $|v_i(0) - v_i(1)| \leq 1$.

Example 4.

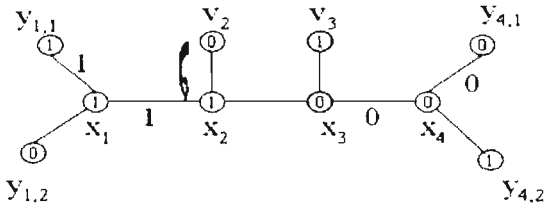


Figure 5.

For the following, we list some data structures for computing BI sets of cubic trees in Figure 5:

	x_1	x_2	x_3	x_4	y_{11}	y_{12}	v_2	v_3	y_{41}	y_{42}
x_1	0	1	0	0	1	1	0	0	0	0
x_2	1	0	1	0	0	0	1	0	0	0
x_3	0	1	0	1	0	0	0	1	0	0
x_4	0	0	1	0	0	0	0	0	1	1

Table 1. Adjacency matrix $Adj_{4,10}$.

x_1	x_2	x_3	x_4	y_{11}	y_{12}	v_2	v_3	y_{41}	y_{42}
1	1	0	0	1	0	0	1	0	1

Table 2. Vertex labeling list $Vertex\text{-}Labeling_{10}$.

	x_1	x_2	x_3	x_4	y_{11}	y_{12}	v_2	v_3	y_{41}	y_{42}
x_1		1			1					
x_2	1									
x_3				0						
x_4			0						0	

Table 3. Edge labeling matrix $Edge\text{-}Labeling_{4,10}$ based on BI computation.

For the following, we provide a **BI Algorithm**, for computing $BI(T)$ of Cubic Trees.

BI Algorithm – for computing $BI(T)$ of Cubic Trees and generates all possible bi's and their corresponding edge labeling information.

Input: An Adjacency Matrix Adj for a Cubic Tree $T=T(V,E)$ and a set $BI(T) \leftarrow \emptyset$.

Output: A sequence of balanced index bi's and for each bi, it print an Edge-Labeling matrix; Finally, it prints out $BI(T)$.

Basic Idea:

Given a Cubic Tree T , we encode the n internal vertices of T into n -bit binary number $(x_1, x_2, \dots, x_n)_2 = X_{10}$, where each x_i is either 0 or 1; X_{10} is the equivalent decimal value of $(x_1, x_2, \dots, x_n)_2$. Note that the maximum decimal number of X_{10} is $2^n - 1$. Similarly, we encode the given $n+2$ leaves of Cubic Tree T into an $n+2$ -bit binary number. That is, $(y_1, y_2, \dots, y_n, y_{n+1}, y_{n+2})_2 = Y_{10}$, where each y_i is either 0 or 1; Y_{10} is the equivalent decimal value of $(y_1, y_2, \dots, y_n, y_{n+1}, y_{n+2})_2$. Let w_x be the number of 1's in $(x_1, x_2, \dots, x_n)_2$ and z_y be the number of 1's in $(y_1, y_2, \dots, y_n, y_{n+1}, y_{n+2})_2$.

Based on the information of w_x and z_y , we could first determine each Friendly Vertex Labeling of T , and Keep it into the Friendly Vertex Labeling Matrix. For each Friendly Vertex Labeling of T , we compute (1) $A := w_x - (n - w_x)$ for the internal vertices of T ; (2) $B := z_y - [(n+2) - z_y]$ for the leaves of T ; and (3) $\left\lceil \left[\frac{3A}{2} + \frac{B}{2} \right] \right\rceil \rightarrow BI(T)$; finally, we compute Edge-Labeling of T and Count $e_f(1)$, $e_f(0)$; then print results.

Main Procedure:

- (1) **for** $K = 0$ to $2^n - 1$ **do**
begin
- (2) Convert K_{10} into $(x_1, x_2, \dots, x_n)_2$; Compute w_x ;
- (3) **for** $J = 0$ to $2^{n+2} - 1$ **do**
begin
- (4) Convert J_{10} into $(y_1, y_2, \dots, y_n, y_{n+1}, y_{n+2})_2$; Compute z_y ;
- (5) **if** $n < w_x + z_y \leq \left(\frac{2n+2}{2} \right) = n+1$ **then** /* A Friendly Vertex Labeling of T found */
Move $(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, y_{n+1}, y_{n+2})_2$ into the **Friendly Vertex Labeling Matrix**;
end;
- end**;
- (6) **while** the Friendly Vertex Labeling Matrix $\neq \emptyset$ **do**
remove a **Friendly Vertex Labeling of T** in the Friendly Vertex Labeling Matrix into the **Vertex Labeling List of T** /*
 $(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, y_{n+1}, y_{n+2})_2 \rightarrow$ **Vertex Labeling List of T** */

begin

- (7) $A := w_K - (n - w_K);$ /* Compute $A = (v_{f,3}^1 - v_{f,3}^0) */$
 (8) $B := z_j - [(n+2) - z_j];$ /* Compute $B = (v_{f,1}^1 - v_{f,1}^0) */$
 (9) Compute $bi = \left\lfloor \left\lfloor \frac{3A}{2} + \frac{B}{2} \right\rfloor \right\rfloor \rightarrow BI(T);$
 (10) Compute **Edge-Labeling of T**;
 (11) Count $e_f(\mathbf{1}), e_f(\mathbf{0});$
 (12) Print **Edge-Labeling of T**;
end;
 (13) Print BI(T).

Lemma 2.3. If T is a cubic tree with n internal vertices, the time complexity for the BI Algorithm in $O(C_1 n^2)$, where $C_1 = C_1(n) = C \binom{2n+2}{n+1} = \frac{(2n+2)!}{(n+1)!(n+1)!}$.

The BI Algorithm computes BI(T) of Cubic Trees and generates all possible bi's and their corresponding edge labeling information.

Proof. For step (1) to step (5) in the BI algorithm, the outer loop iterates 2^n times, step (2) needs $O(n)$; the inner loop iterates 2^{n+2} times, step (4) needs $O(n+2)$, step 5 needs $O(2n+2)$; however, the BI computation in the

inner loop took place only if $n < w_K + z_j \leq \binom{2n+2}{2} = n+1$, (i.e., when a

Friendly Vertex Labeling of T found), we could calculate that the BI

computation loops mainly at most $C_1 = C \binom{2n+2}{n+1} = \frac{(2n+2)!}{(n+1)!(n+1)!}$ iterations

since T has $2n+2$ vertices, a friendly labeling f of T when $|v_f(0) - v_f(1)| \leq 1$.

Thus, the time complexity for the step (1) to step (5) is in $O(C_1 n)$. From Step (6)

to step (12), the while loop iterates at most C_1 times. Inside the while loop, step

(7), (8), (9), each needs only a constant time. To compute **Edge-Labeling** of T

in step (10), we need $O(n * (2n+2)) = O(2n^2 + 2n) \approx O(2n^2)$ and to count $e_f(\mathbf{1}),$

$e_f(\mathbf{0}),$ we need $O(|E|) = O(2n+1)$; finally, it takes $O(2n^2)$ to print out the **Edge-**

Labeling matrix of T; Therefore, it takes about $O(4C_1 n^2)$ in total. Thus, the BI

algorithm is in $O(C_1 n + 4C_1 n^2) \approx O(C_1 n^2)$, where

$$C_1 = C_1(n) = C \binom{2n+2}{n+1} = \frac{(2n+2)!}{(n+1)!(n+1)!} = \frac{(2n+2)(2n+1)!}{(n+1)! * 2 * 1} \quad "$$

Remark 2.2. A faster BI Algorithm could be developed as follows:

(1) Using an *Adjacency List Representation* (instead of using an Adjacency Matrix) for the Tree, especially, for the sparse graphs. Therefore, we could build two Adjacency Lists, one for the tree configuration (or topology), the other one for the edge labeling purpose. Furthermore, we could apply a *Depth First Search procedure* to traverse (i.e., search) the tree structure. Therefore, to compute BI(T), Edge-Labeling of T, and count $e_f(1)$, $e_f(0)$, and print out the Edge-Labeling of T, we only need $O(|V|) = O(2n+2) \approx O(n)$. !

(2) We could develop a faster procedure for finding a friendly labeling f of T when $|v_f(0) - v_f(1)| \leq 1$. So that we could reduce the cost of

$$C_1 = C_1(n) = C \binom{2n+2}{n+1}.$$

Lemma 2.4. If T is a cubic tree with n internal vertices, the time complexity for the faster BI Algorithm is in $O(C_1 n)$, where $C_1 = C_1(n) = C \binom{2n+2}{n+1}$

$$= \frac{(2n+2)!}{(n+1)!(n+1)!}. \text{ Note that the faster BI Algorithm computes BI(T) of Cubic}$$

Trees and generates all possible bi's and their corresponding edge labeling information.

Theorem 2.5. If T is cubic tree with n internal vertices, then BI(T) is

(1) $\{0, 2, \dots, n-2, n\}$, if n is even, and

(2) $\{1, 3, \dots, n-2, n\}$, if n is odd.

Proof. By Lemma 2.3, we see that T has $2n+2$ vertices and $2n+1$ edges. If T is a cubic tree with n internal vertices, then $L(T) = n+2$. Then by (2.4), we could

$$\text{compute BI(T) by } \left[\left[\frac{3}{2}(v_{f,3}^1 - v_{f,3}^0) + \frac{1}{2}(v_{f,1}^1 - v_{f,1}^0) \right] \right].$$

For the following, let $A = (v_{f,3}^1 - v_{f,3}^0)$ and $B = (v_{f,1}^1 - v_{f,1}^0)$, we list all possible indexes in BI(T) for friendly labelings of T in Table 1 if n is even. Note that the first row of Table 1 shows that if we label all the internal vertices by 1 (i.e., $x = n$ since $v_{f,3}^1 = n$ and $v_{f,3}^0 = 0$) and label one leaf of T by 1 and all the other leaf vertices by 0, (i.e., $y = -n$ since $v_{f,1}^1 = 1$ and $v_{f,1}^0 = n+1$), we see that this is friendly labeling and $e(0) = 0$ and $e(1) = n$. Thus $\max(\text{BI(T)}) = n$. The

remaining rows of Table 1 show that **when, n is even. $BI(T) = n-2k$ is achieved for $k=1,2,3,\dots$**

A	B	$e_f(1) - e_f(0)$	$BI(T) = e_f(0) - e_f(1) $
n	-n	n	n
n-2	-(n-2)	n-2	n-2
n-4	-(n-4)	n-4	n-4
.....
2	-2	2	2
0	0	0	0
-2	2	-2	2
.....
-(n-4)	n-4	-(n-4)	n-4
-(n-2)	(n-2)	-(n-2)	n-2
-n	n	-n	n

Table 4. All possible indexes in $BI(T)$ for friendly labelings of T , if n is even.

We now list all possible indexes in $BI(T)$ for friendly labelings of T in Table 2 **when n is odd. It shows that $BI(T) = n-2k$ is achieved for $k=0,1,2,3,\dots$!**

A	B	$e_f(1) - e_f(0)$	$BI(T) = e_f(0) - e_f(1) $
n	-n	n	n
n-2	-(n-2)	n-2	n-2
n-4	-(n-4)	n-4	n-4
.....
3	-3	3	3
1	-1	1	1
-3	3	-3	3
.....
-(n-4)	n-4	-(n-4)	n-4
-(n-2)	(n-2)	-(n-2)	n-2
-n	n	-n	n

Table 5. All possible indexes in $BI(T)$ for friendly labelings of T , if n is odd.

Example 5. Figure 6 depicts the cubic tree T , with 4 internal vertices and $BI(T) = \{0,2,4\}$.

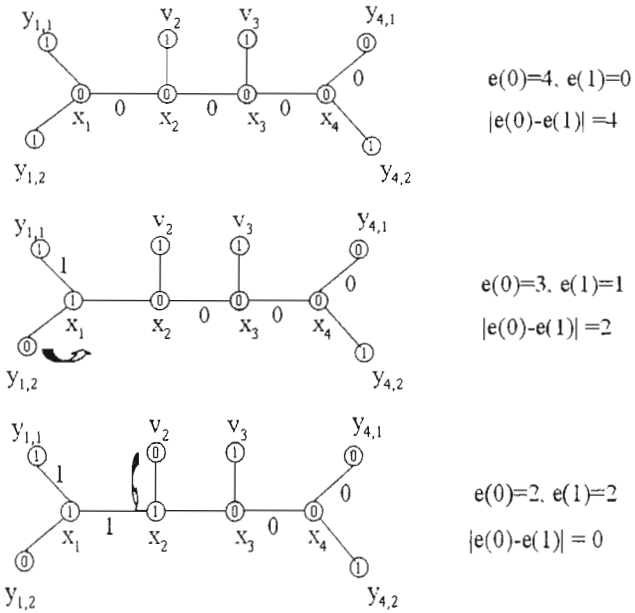
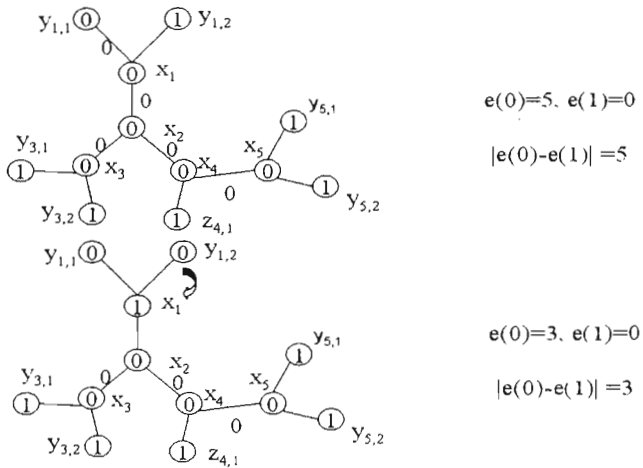
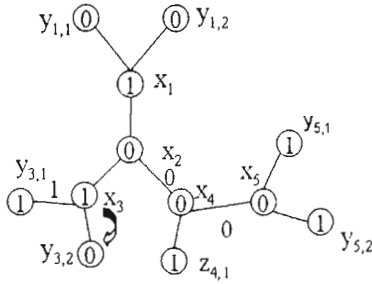


Figure 6.

Example 6. Figure 7 depicts the cubic tree T , with 5 internal vertices and $BI(T) = \{1,3,5\}$.





$$e(0)=2, e(1)=1$$

$$|e(0)-e(1)|=1$$

Figure 7.

3. Product-Cordial Index Sets of Cubic Trees

In this section, we generalize the computing approach for finding balance index sets of cubic trees in Section 2 to the case of computing the product-cordial index sets of cubic trees.

Note that any cubic tree T with n internal vertices has $2n+2$ vertices. Thus all cubic trees have even order.

Lemma 3.1. For any (p,q) -graph G , all elements of $PCI(G)$ have the same parity as q .

Proof. By definition, all edges will be labeled under the edge-labeling f . The number, $e_f(0) \neq e_f(1)$, in fact, is equal to q . Therefore, the element of $PCI(G)$, " $e_f(0) - e_f(1)$ ", has the same parity as q by algebraic reason. !

Lemma 3.2. If T is cubic tree of order $2n+2$, with n internal vertices, then the maximum product-cordial index of T , $\max(PCI(T))$, is $2n+1$.

Proof. We see that T has $2n+2$ vertices and $2n+1$ edges. If we label all the internal vertices by 0 and one leaf of T by 0 and all the other leaf vertices by 1, we see that this is friendly labeling and $e(0)=2n+1$ and $e(1)=0$. Thus $\max(PCI(T))= 2n+1$. !

Corollary 3.3. If T is cubic tree of order $2n+2$, with n internal vertices, for any friendly labeling f of T , (i.e., $|v_f(0) - v_f(1)| \leq 1$), then $\max(BI(T)) < \max(PCI(T))$.

Proof. The result immediately follows from Theorem 2.4 and Lemma 3.2. !

Lemma 3.4. If T is cubic tree of order $2n+2$, with n internal vertices, then the minimum product-cordial index of T , $\min(PCI(T))$, is 1.

Proof. If we label all the internal vertices by 1 and one leaf of T by 1 and all the other leaf vertices by 0, we see that this is friendly labeling. Note that all the internal vertices are connected in a subtree with $n-1$ edges; all these $n-1$ edges must be labeled by 1. Since only one leaf is labeled by 1, the edge connected with this leaf with the internal vertex must be labeled by 1. Therefore, we have $e(1) = n-1+1 = n$ and all the remaining $n+1$ edges of T must be labeled by 0 (i.e., $e(0) = n+1$). Thus, $\min(\text{PCI}(T)) = |e(1) - e(0)| = 1$. !

Also, we show the following result.

Theorem 3.5. If T is cubic tree of order $2n+2$, with n internal vertices, then $\text{PCI}(T) = \{1, 3, \dots, 2n+1\}$.

Proof. By Lemma 3.2, we have $\max(\text{PCI}(T)) = 2n+1$, where we labeled all n internal vertices by 0 and one leaf of T by 0 and all the other $n+1$ leaf vertices by 1, then it is a friendly labeling with $e(0) = 2n+1$ and $e(1) = 0$. That all $2n+1$ edges are labeled 0.

Also, by Lemma 3.4, we know that $\min(\text{PCI}(T)) = 1$. We now show that **$2n+1-2k$ is achieved for $k=1, 2, 3, \dots, n-1$** . First, we pick up a "**root node**" (i.e., the "**root of subtree of internal vertices**") in the subtree of all the internal vertices.

For case $k=1$, we start by labeling root node by 1 and all other $n-1$ internal vertices by 0. We then label one leaf, connected to the root node, by 1 and $n-1$ other leaves by 1 and the remaining 2 leaves by 0. We see that this is friendly labeling and $e(0) = 2n$ and $e(1) = 1$. Thus, $\text{pci} = |e(1) - e(0)| = |1 - 2n| = 2n-1$, for the case $k=1$.

For case $k=2$, we start by labeling 2 internal vertices (i.e., the root node and its child) by 1 and all other $n-2$ internal vertices by 0. We then label one leaf, connected to the root node, by 1 and $n-2$ other leaves, connected with 0 label internal vertices, by 1 and the remaining 3 leaves by 0. We see that this is friendly labeling and $e(1) = 2$, (since the edge, connected root node and its child, is labeled 1 and the leaf, connected to the root node, is labeled by 1). Clearly, $e(0) = 2n-1$. Therefore, $\text{pci} = |e(1) - e(0)| = |2 - (2n-1)| = 2n-3$, for the case $k=2$.

For case $k=i$, where $i > 2$, we start by labeling the root node and its $i-1$ immediate descendants by 1 and all other $n-i$ internal vertices by 0. Note that all the $k=i$ internal vertices are connected in a subtree with $k-1=i-1$ edges; all these $k-1=i-1$ edges are labeled by 1. We then label one leaf, connected to the root node, by 1 and label another $(n-i)$ leaves, each connected with a label 0 internal vertex, by 1, and label all other $i+1$ leaves by 0. We see that this is friendly labeling and $e(1) = k=i$ and $e(0) = [(2n+1)-i]$. Therefore, $\text{pci} = |e(1) - e(0)| = |i - [(2n+1)-i]| = 2n + (1-2i)$, for the case $k=n-1$.

For case $k=n-1$, we start by labeling the root node and its $n-2$ immediate descendants by 1 and the other one internal vertex by 0. Note that all the $k=n-1$ internal vertices are connected in a subtree with $k-1=n-2$ edges; all these $k-1=n-2$ edges are labeled by 1. We then label one leaf, connected to the root node, by 1 and label another one leaf, connected with the label 0 internal vertex, by 1, and label all other leaves by 0. We see that this is friendly labeling and $e(1) = n-1$ and $e(0) = n+2$. Therefore, $pci = |e(1)-e(0)| = |(n-1) - (n+2)| = 3$, for the case $k=n-1$. !

Example 7. Figure 8 shows cubic tree T_2 has 4 internal vertices and 6 leaves and $PCI(T_2) = \{1, 3, 5, 7, 9\}$.

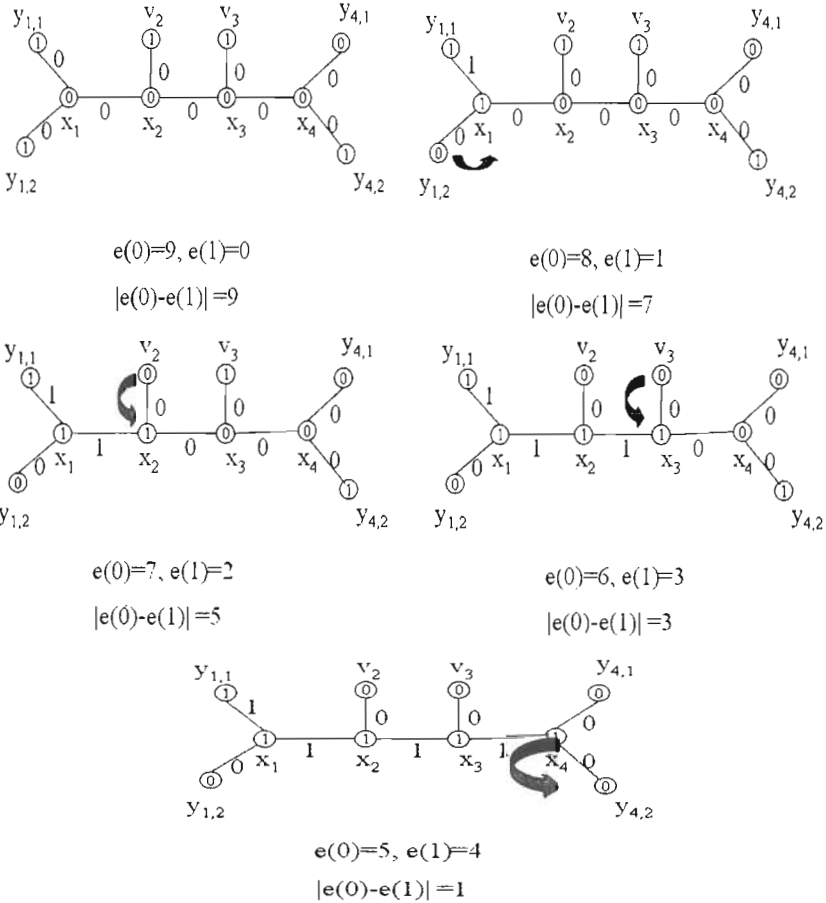


Figure 8.

Example 8. Figure 9 shows a cubic tree T has 4 internal vertices and 6 leaves and $PCI(T) = \{1, 3, 5, 7, 9\}$.

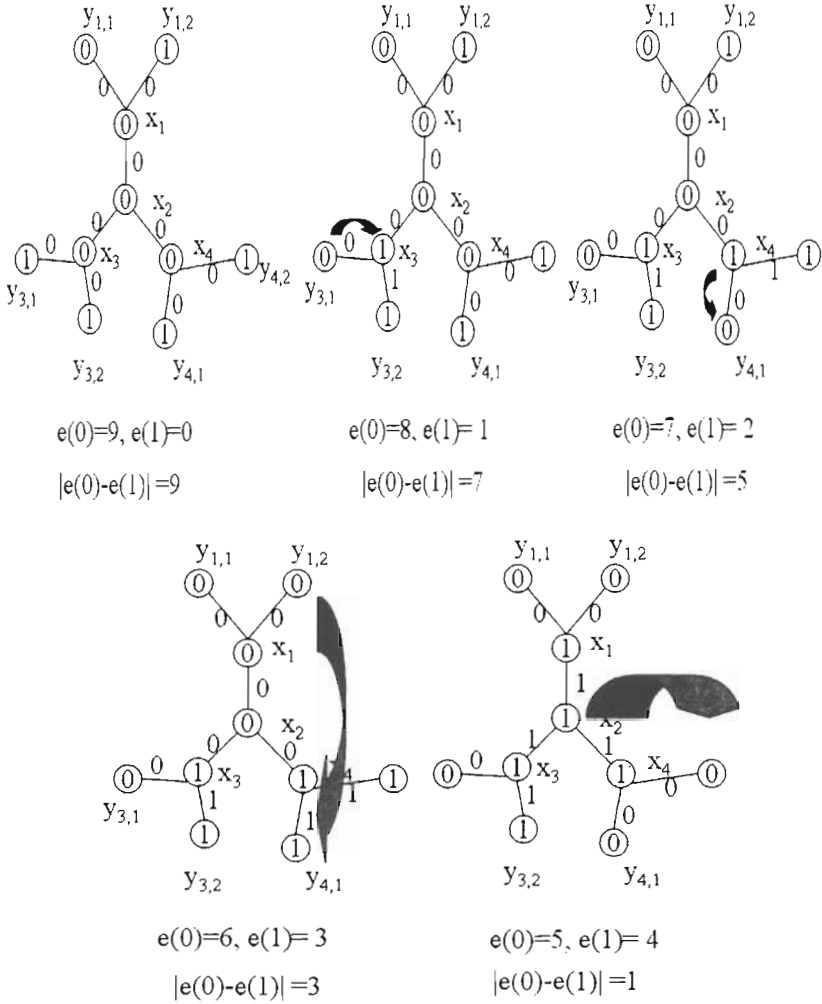


Figure 9.

Example 9:

Consider the Cubic Tree T in Figure 5, if its adjacency matrix $\mathbf{Adj}_{4,10}$ and the vertex labeling list $\mathbf{Vertex-Labeling}_{10}$ are shown in Table 1, Table 2, respectively. The edge labeling matrix $\mathbf{Edge-Labeling}_{4,10}$ based on PCI computation is shown in Table 6.

	x_1	x_2	x_3	x_4	y_{11}	y_{12}	v_2	v_3	y_{41}	y_{42}
x_1		1			1	0				
x_2	1		0				0			
x_3		0		0				0		
x_4			0						0	0

Table 6. Edge labeling matrix $\mathbf{Edge-Labeling}_{4,10}$ based on PCI computation.

For the following, we provide a **PCI Algorithm**, for computing $\mathbf{PCI}(T)$ of Cubic Trees.

PCI Algorithm - for computing $\mathbf{PCI}(T)$ of Cubic Trees, which generates all possible pci 's and their corresponding edge labeling information.

Input: An Adjacency List \mathbf{Adj} for a Cubic Tree T and a set $\mathbf{PCI}(T) \leftarrow \emptyset$.

Output: A sequence of balanced index bi 's and for each bi , it print an Edge-Labeling matrix; Finally, it prints out $\mathbf{PCI}(T)$.

Main Procedure:

Note that step (1) – step (6) are exactly the same as step (1) – step (6) specified in **the faster BI Algorithm** (See the, Lemma 2.3, Lemma 2.4 discussions in Section 2.)

/ Based on the information of w_x and z_y , we could first determine each Friendly Vertex Labeling of T , and Keep it into the Friendly Vertex Labeling Matrix. */*

*/*The following while loop (i.e., step (6) – step (10)) computes pci . */*

(6) while the Friendly Vertex Labeling Matrix $\neq \emptyset$ **do**

remove a **Friendly Vertex Labeling of T** in the Friendly Vertex Labeling Matrix into the **Vertex Labeling List of T**

/ $(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, y_{n+1}, y_{n+2})_2 \rightarrow$ Vertex Labeling List of T */*

begin

(7) Compute **Edge-Labeling of T** ;

(8) Count $e_f(1)$, $e_f(0)$ for computing pci ;

(9) Print **Edge-Labeling of T** ;

end;

(10) Print $\mathbf{PCI}(T)$.

Lemma 3.5. If T is a cubic tree with n internal vertices, then the time complexity for *the faster PCI Algorithm* is in $O(C_1n)$, where

$$C_1 = C_1(n) = C \binom{2n+2}{n+1} = \frac{(2n+2)!}{(n+1)!(n+1)!}.$$

Remark 3.1. The *PCI Algorithm*, using the adjacency Matrix and Edge-Labeling matrix, needs $O(C_1n^2)$.

Both *the faster PCI Algorithm* and the *PCI Algorithm* compute $PCI(T)$ of Cubic Trees and generate all possible *pci*'s and their corresponding edge labeling information.

Proof. The analysis is similar to the results from Lemma 2.3, Lemma 2.4. !

Acknowledgment

The authors thank Prof. Andrew Lee for his valuable review comments on the BI computations.

References

1. M. Benson and Sin-Min Lee, On cordialness of regular windmill graphs, *Congr. Numer.*, **68** (1989) 49-58.
2. I. Cahit, Cordial graphs: A weaker version of graceful and harmonious graphs, *Ars Combin.*, **23** (1987) 201-207.
3. I. Cahit, On cordial and 3-equitable graphs, *Utilitas Mathematica*, **37** (1990) 189-198.
4. I. Cahit, Recent results and open problems on cordial graphs, *Contemporary Methods in Graph Theory*, 209-230, Bibliographisches Inst., Mannheim, 1990.
5. N. Cairnie and K. Edwards, The computational complexity of cordial and equitable labelling, *Discrete Math.*, **216** (2000) 29-34.
6. G. Chartrand, Sin-Min Lee and Ping Zhang, On uniformly cordial graphs, *Discrete Math.* **306** (2006), 726-737.
7. A. Elumalai, On graceful, cordial and elegant labelings of cycle related and other graphs, Ph.D. dissertation of Anna University, 2004, Chennai, India.
8. Y.S. Ho, Sin-Min Lee and S.S. Shee, Cordial labellings of the Cartesian product and composition of graphs, *Ars Combin.*, **29** (1990) 169-180.

9. Y.S. Ho, S-M. Lee and S.S. Shee, Cordial labellings of unicyclic graphs and generalized Petersen graphs. *Congr. Numer.*, **68** (1989) 109-122.
10. M. Hovay, A-cordial graphs, *Discrete Math.*, **93** (1991) 183-194.
11. W. W. Kirchherr, On the cordiality of certain specific graphs, *Ars Combinatoria*, **31**, (1991) 127-138.
12. W. W. Kirchherr, Algebraic approaches to cordial labeling, *Graph Theory, Combinatorics, Algorithms, and Applications*, Y. Alavi, et. al., editors, SIAM, (1991), 294-299.
13. W. W. Kirchherr, NEPS operations on cordial graphs, *Discrete Math.*, **115**, (1993), 201-209.
14. S. Kuo, G.J. Chang and Y.H.H. Kwong, Cordial labeling of mK_n , *Discrete Math.*, **169** (1997), 121-131.
15. Harris Kwong, Sin-Min Lee, Ho Kuen Ng, On friendly index sets of 2-regular graphs, *Discrete Math.* **308**(2008), 5522-5532.
16. Sin-Min Lee and A. Liu, A construction of cordial graphs from smaller cordial graphs, *Ars Combin.*, **32** (1991) 209-214.
17. Sin-Min Lee, A. Liu and S.K. Tan, On balanced graphs, *Congressus Numerantium* **87** (1992), 59-64.
18. Sin-Min Lee and H.K. Ng, On friendly index sets of bipartite graphs, *Ars Combin.* (2008),
19. Sin-Min Lee and H.K. Ng, On friendly index sets of total graphs of trees, to appear in *Utilita Mathematica*.
20. Sin-Min Lee and H.K. Ng, On product-cordial index sets of graphs, manuscript.
21. Y.H. Lee, H.M. Lee and G.J. Chang, Cordial labelings of graphs, *Chinese J. Math.*, **20** (1992) 3, 263-273.
22. E. Seah, On the construction of cordial graphs, *Ars Combin.*, **31** (1991) 249-254.
23. M.A. Seoud and A.E.I. Abdel Maqsood, On cordial and balanced labelings of graphs, *J. Egyptian Math. Soc.*, **7** (1999) 127-135.
24. S.C. Shee, The cordiality of the path-union of n copies of a graph, *Discrete Math.*, **151** (1996) 1-3, 221-229.
25. S.C. Shee and Y.S. Ho, The cordiality of one-point union of n copies of a graph, *Discrete Math.*, **117** (1993) 225-243.
26. W. C. Shiu and H. Kwong, An algebraic approach for finding balance index sets, *Australasian Journal of Combinatorics*, to appear.