

On Product-Cordial Index Sets of Cylinders

Harris Kwong

Dept. of Math. Sci.

SUNY Fredonia

Fredonia, NY 14063, USA

Sin-Min Lee

Dept. of Comp. Sci.

San Jose State University

San Jose, CA 95192, USA

Ho Kuen Ng

Dept. of Math.

San Jose State University

San Jose, CA 95192, USA

Abstract

Let $G(V, E)$ be a simple graph. A vertex labeling $f : V \rightarrow \mathbb{Z}_2$ induces an edge labeling $f^* : E \rightarrow \mathbb{Z}_2$ defined by $f^*(xy) = f(x)f(y)$ for each edge $xy \in E$. For each $i \in \mathbb{Z}_2$, let $v_f(i) = |\{v \in V : f(v) = i\}|$, and $e_f(i) = |\{e \in E : f^*(e) = i\}|$. A vertex labeling f of G that satisfies the condition $|v_f(0) - v_f(1)| \leq 1$ is said to be friendly. The product-cordial index set of the graph G is defined as $\{|e_f(0) - e_f(1)| : f \text{ is friendly}\}$. In this paper, we study the product-cordial index set of cylinder graph $C_m \times P_n$.

1 Introduction

Given a simple graph $G = (V, E)$, a vertex labeling $f : V \rightarrow \mathbb{Z}_2$ assigns either 0 or 1 to each vertex of G . Denote by $v_f(i)$ the number of vertices that are labeled i . A vertex labeling is said to be *friendly* if $|v_f(0) - v_f(1)| \leq 1$. One could define an induced edge labeling $g : E \rightarrow \mathbb{Z}_2$ based on certain rule(s). By further imposing condition(s) on the resulting edge labels, one could define various kinds of graph labelings with interesting results. For instance, given a vertex labeling f , we can label an edge

- 0 if both of its endpoints are labeled 0,
- 1 if both of its endpoints are labeled 1, and
- leave the edge unlabeled if its endpoints are labeled differently.

The result is a partial edge labeling $\tilde{f} : E \rightarrow \mathbb{Z}_2$.

Define $e_f(i)$ as the number of edges that are labeled i . A graph is **balanced** if it admits a friendly labeling f such that $|e_f(0) - e_f(1)| \leq 1$. In other words, a graph is balanced if we can label its vertices with 0 and 1 such that its vertices are evenly divided into 0- and 1-vertices, and its edges are also evenly divided into 0- and 1-edges under \bar{f} . The concept of balanced graphs was first introduced by the second author, Liu, and Tan [8] in 1992. Obviously, many graphs are not balanced. To measure how close a graph to being balanced, Lee, the second and third authors [7] studied the **balanced index set** of a graph:

$$BI(G) = \{|e_f(0) - e_f(1)| : f \text{ is friendly}\}.$$

Simply put, $BI(G)$ is the set of possible values that $|e_f(0) - e_f(1)|$ can assume as we go over all the friendly labelings of G . In this regard, a graph is balanced if $BI(G) \cap \{0, 1\} \neq \emptyset$.

In a balanced labeling, no computation is involved in the assignment of the edge labels. We can of course assign the edge labels based on certain mathematical operations on the labels of their two endpoints. For example, define $f^+ : V \rightarrow \mathbb{Z}_2$ according to

$$f^+(uv) = f(u) + f(v), \quad uv \in E.$$

In other words,

$$f^+(uv) = \begin{cases} 0 & \text{if } u \text{ and } v \text{ are labeled the same way,} \\ 1 & \text{if } u \text{ and } v \text{ are labeled differently.} \end{cases}$$

If $|e_{f^+}(0) - e_{f^+}(1)| \leq 1$, we say f is **cordial**, a notion introduced by Cahit [1] around 1987 as an alternative to studying graceful labeling. As expected, many graphs are far from being cordial. The second and third authors [6] initiated in 2004 the study of **friendly index set** of a graph:

$$FI(G) = \{|e_{f^+}(0) - e_{f^+}(1)| : f \text{ is friendly}\}$$

Hence, a graph is cordial if $FI(G) \cap \{0, 1\} \neq \emptyset$. Although their work on friendly index sets of bipartite graphs did not appear until 2008, it has inspired many research projects on similar subjects. To cite a few examples: [3, 4, 5, 10].

Instead of sum, Sundaram, Ponraj and Somasundaram [11]–[13] studied in 2004–2006 the induced labeling

$$f^*(uv) = f(u) \cdot f(v), \quad uv \in E.$$

They called f **product-cordial** if it is friendly, and $|e_{f^*}(0) - e_{f^*}(1)| \leq 1$. Salehi [9] introduced in 2009 the concept of a **product-cordial set**:

$$PC(G) = \{|e_{f^*}(0) - e_{f^*}(1)| : f \text{ is friendly}\}.$$

Since this is the multiplicative version of $FI(G)$, we will call it a **product-cordial index set**, and write $PCI(G)$.

When the context is clear, for simplicity, we will suppress the subscript f^* and write $|e(0) - e(1)|$ instead of the more complicated $|e_{f^*}(0) - e_{f^*}(1)|$. Let q be the number of edges in G . Then

$$e(0) - e(1) = [q - e(1)] - e(1) = q - 2e(1).$$

The fact that $e(0) - e(1) \equiv q \pmod{2}$ immediately proves the next result.

Lemma 1.1 For any simple graph G with q edges,

$$PCI(G) \subseteq \begin{cases} \{0, 2, 4, \dots, q\} & \text{if } q \text{ is even,} \\ \{1, 3, 5, \dots, q\} & \text{if } q \text{ is odd.} \end{cases}$$

In this paper, we discuss the PCI sets of the **cylinder graph** $C_m \times P_n$, where $m \geq 3$ and $n \geq 1$. In [2], Chou, the second author and Wang determined $PCI(C_m \times P_n)$ for $m = 3, 4$. We extend their results to all m and n .

2 PCI Sets of $C_m \times P_n$

There are $p = mn$ vertices and $q = m(2n - 1)$ edges on $C_m \times P_n$. Let the vertices on the i th cycle be a_{ij} , where $1 \leq i \leq n$, and $1 \leq j \leq m$. There are two kinds of edges: either along a cycle or a path. Note that an edge uv is an 1-edge if and only if both u and v are 1-vertices. Hence $e(1)$ is the number of pairs of adjacent 1-vertices. It is clear that $e(1) < q/2$, hence $e(0) - e(1) = q - 2e(1) > 0$. The case of $n = 1$ is different from the case of $n > 1$.

Theorem 2.1 For $m \geq 3$,

$$PCI(C_m) = \begin{cases} \{m, m-2, m-4, \dots, 2\} & \text{if } m \text{ is even,} \\ \{m, m-2, m-4, \dots, 1\} & \text{if } m \text{ is odd.} \end{cases}$$

Proof. For each t satisfying $1 \leq t \leq \lfloor \frac{m}{2} \rfloor$, label the first t vertices of C_m with 1, followed by $\lfloor \frac{m}{2} \rfloor - t$ pairs of 01's, and the remaining $\lfloor \frac{m}{2} \rfloor - \lfloor \frac{m}{2} \rfloor + t$ vertices 0. The result is a friendly labeling with $v(1) = \lfloor \frac{m}{2} \rfloor \leq \lfloor \frac{m}{2} \rfloor = v(0)$, and $e(1) = t - 1$. The result for even m follows immediately from $e(0) - e(1) = m - 2e(1)$. But for odd n , the smallest value $e(0) - e(1)$ generated thus far is 3.

However, for odd m , we could also have $v(1) = \lceil \frac{m}{2} \rceil$, and $v(0) = \lfloor \frac{m}{2} \rfloor$. We could label the vertices of C_m with $\lfloor \frac{m}{2} \rfloor$ consecutive 1-vertices, followed by $\lfloor \frac{m}{2} \rfloor$ consecutive 0-vertices. Then $e(1) = \lfloor \frac{m}{2} \rfloor - 1$, and the proof is complete. \square

In light of Theorem 2.1, we will assume $n \geq 2$ in the rest of the paper. The maximum index is obtained when $e(1)$ is minimized, which occurs when the vertices are labeled alternately with 0 and 1:

$$\begin{array}{cccccccc}
 0 & 1 & 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 \\
 1 & 0 & 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 \\
 0 & 1 & 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 & & & & & & & & & & & & & & & & & m \text{ is odd}
 \end{array}$$

Here, each row represents the vertex labels on a cycle. Note that the respective vertices on the first and last columns are adjacent. We find

$$e(1) \geq \begin{cases} 0 & \text{if } m \text{ is even,} \\ \lfloor \frac{m}{2} \rfloor & \text{if } m \text{ is odd.} \end{cases}$$

$$e(0) - e(1) \leq \begin{cases} q & \text{if } m \text{ is even,} \\ q - 2\lfloor \frac{q}{2} \rfloor & \text{if } m \text{ is odd.} \end{cases}$$

Hence
 If $e(1) = 1$, there is only one pair of adjacent 1-vertices. Since $n \geq 2$, these two 1-vertices are surrounded by at least four 0-vertices. This is clearly impossible when $mn = 6$. Even if $mn \geq 8$, among the remaining $mn - 6$ vertices, $v(1) - v(0) \geq 2$, it is impossible to maintain friendliness with no other adjacent 1-vertices if m is even. Hence $e(0) - e(1) \neq q - 2$ if m is even.

To minimize $e(0) - e(1)$, we want $e(1)$ to be as large as possible. This requires all the 1-vertices to be packed together either along the paths next to each other, or along the cycles atop each other. In the first case, we may assume the 1-vertices occupy the last $\lfloor m/2 \rfloor$ paths, and also the top $\lfloor n/2 \rfloor$ vertices in the middle path if m is odd:

$$\begin{array}{cccccccc}
 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\
 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\
 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\
 & & & & & & & & & & & & & & & & & m \text{ is even}
 \end{array}$$

We find

$$e(1) \leq \begin{cases} n(m-1) - \frac{m}{2} & \text{if } m \text{ is even,} \\ n(m-1) - \frac{m+1}{2} & \text{if } m \text{ is odd, and } n \text{ is even,} \\ n(m-1) - \frac{m-1}{2} & \text{if } m \text{ and } n \text{ are odd.} \end{cases}$$

Hence

$$e(0) - e(1) \geq \begin{cases} 2n & \text{if } m \text{ is even,} \\ 2n+1 & \text{if } m \text{ is odd, and } n \text{ is even,} \\ 2n-1 & \text{if } m \text{ and } n \text{ are odd.} \end{cases}$$

In the second case, we may assume the 1-vertices appear on the top $\lfloor n/2 \rfloor$ cycles, and also the last $\lfloor m/2 \rfloor$ vertices in the middle cycle if n is odd:

$$\begin{array}{cccccccc}
 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\
 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\
 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\
 & & & & & & & & & & & & & & & & & n \text{ is odd}
 \end{array}$$

We find

$$e(1) \leq \begin{cases} m(n-1) & \text{if } n \text{ is even,} \\ m(n-1) - 1 & \text{if } n \text{ is odd and } m \text{ is even,} \\ m(n-1) & \text{if } n \text{ and } m \text{ are odd.} \end{cases}$$

Therefore

$$e(0) - e(1) \geq \begin{cases} m & \text{if } n \text{ is even,} \\ m+2 & \text{if } n \text{ is odd, and } m \text{ is even,} \\ m & \text{if } n \text{ and } m \text{ are odd.} \end{cases}$$

We use two methods (Algorithm P and C, respectively) to generate the values of $e(1)$ within the bounds indicated above. Merging the resulting PC indices will yield the desired PCI set.

Both algorithms compose of n stages, and start with the vertex labeling with alternating 0- and 1-vertices. Each stage (except the last) consists of two parts: Stage T_iA and T_iB , where T is either P or C. The details vary, depends on the parity of m and n . However, the general idea is the same.

In Stage P1, we move the 1-vertices from the first two cycles to the top cycle. In Stage P1A, the 1-vertices on the top cycle are moved to the left so that they are adjacent to the 1-vertices in the second cycle. In Stage P1B, the 1-vertices on the second cycle are moved to the first cycle, and replace the 0-vertices between the 1-vertices. This yields a cycle of 1-vertices atop a cycle of 0-vertices.

In Stage P_i , where $2 \leq i < n$, the 1-vertices on the $(i+1)$ st cycle are moved to the i th cycle, and placed at the right-half of the cycle. In Stage P_iA , the 1-vertices are moved up, one at a time, to be placed underneath the 1-vertices at the "bottom" of the current configuration (which

could be either an 1-vertex on the second-half of the i th cycle, or an 1-vertex on the first-half of the top cycle), in such a way that every time $e(1)$ is increased by 1. There are some complications in some special cases. The full details are discussed in the next section. In Stage P*i*B, these 1-vertices are moved to the rightmost positions of the second half of the i th cycle.

After $n - 1$ stages, the top cycle is occupied by 1-vertices, and in each of the next $n - 2$ cycles, all the 1-vertices appear on the right-half of the cycle, and the last cycle consists of 0-vertices only. The last stage of Algorithm P moves the 1-vertices occupying the first half of the top cycle to the right half of the bottom cycle. The result is a vertex labeling with all 1-vertices packed together along the paths as described in the first case.

In Algorithm C, Stage C1 is identical to Stage P1. For $2 \leq i < n$, Stage C*i* is similar to Stage P*i*, except that the 1-vertices are pushed to the second half of a cycle if i is even, and placed on the first half if i is odd. The last stage will place the 1-vertices from the last cycle at either the second half or the first half of whatever cycle is available.

3 The Case of Even m

We will first describe Algorithms P and C, and the values of $e(0) - e(1)$ that they generate. Then we can merge them to form the PCI set.

3.1 Algorithm P

The initial vertex labeling produces $e(1) = 0$.

Stage P1A: Interchange the labels of a_{11} and a_{12} .

```

1 0 0 1 0 1 ... 0 1 0 1
1 0 1 0 1 0 ... 1 0 1 0
0 1 0 1 0 1 ... 0 1 0 1
: : : : : : : : : : : :
: : : : : : : : : : : :

```

Now both $a_{1m}a_{11}$ and $a_{11}a_{21}$ are 1-edges. Hence $e(1) = 2$. Next, swap the labels of a_{13} and a_{14} , then a_{15} and a_{16} , a_{17} and a_{18} , ..., until $a_{1,m-3}$ and $a_{1,m-2}$:

```

1 0 1 0 0 1 ... 1 0 0 1
1 0 1 0 1 0 ... 1 0 1 0
0 1 0 1 0 1 ... 0 1 0 1
: : : : : : : : : : : :
: : : : : : : : : : : :

```

Each time, the value of $e(1)$ increased by 1. If we continue to switch the labels of $a_{1,m-1}$ and a_{1m} , a new 1-edge will be created as usual, but the

edge $a_{1m}a_{11}$ is no longer an 1-edge, so the value of $e(1)$ remains unchanged:

```

1 0 1 0 1 0 ... 1 0 1 0
1 0 1 0 1 0 ... 1 0 1 0
0 1 0 1 0 1 ... 0 1 0 1
: : : : : : : : : : : :
: : : : : : : : : : : :

```

This terminates Stage P1A. It generates $\frac{m}{2} - 1$ values of $e(1)$: from 2 through $\frac{m}{2}$.

Stage P1B: Switch the labels of a_{21} and a_{12} , then a_{23} and a_{14} , a_{25} and a_{16} , ..., until $a_{2,m-1}$ and a_{1m} :

```

1 1 1 1 1 1 ... 1 1 1 1
0 0 0 0 0 0 ... 0 0 0 0
0 1 0 1 0 1 ... 0 1 0 1
: : : : : : : : : : : :
: : : : : : : : : : : :

```

The value of $e(1)$ is increased by 1 in every switch. We have generated the next $\frac{m}{2}$ values of $e(1)$: from $\frac{m}{2} + 1$ through m .

Stage P*i*A ($2 \leq i < n$): The objective of this part of Stage P*i* is to move the 1-vertices on the $(i + 1)$ st cycle, one at a time from the right end, to make contact with the 1-vertices at the bottom of the current configuration, more specifically, $a_{i-1,m}$, $a_{i-1,m-2}$, ..., so that $e(1)$ is increased by 1 each time. When we exhaust the 1-vertices on the $(i - 1)$ st cycle, we will continue with the a_{1j} 's. If $n \equiv 0 \pmod{4}$, we need to put the last 1-vertex (either $a_{i+1,1}$ or $a_{i+1,2}$) under one of the a_{2j} that has just been changed into an 1-vertex. These scenarios are illustrated in the following examples (both Stage P3A):

```

1 1 1 1 1 1      1 1 1 1 1 1
0 1 0 1 1 1      0 1 0 1 1 1
0 0 0 1 0 1      0 0 1 0 0 1
0 0 0 0 0 0      0 0 0 0 0 0
0 1 0 1 0 1      0 1 0 1 0 1

```

For $m = 4$, we may need to move the last 1-vertex to make contact with an 1-vertex on the next row, as in

```

1 1 1 1      1 1 1 1      1 1 1 1
0 0 1 1      0 0 1 1      0 0 1 1
0 0 0 0      0 0 0 1      0 0 0 1
1 0 1 0      1 0 0 0      0 1 0 0
0 1 0 1      0 1 0 1      0 1 0 1

```

and in Stage $n-1$, we may even need to put the last 1-vertex under $a_{n-1,4}$:

```

1 1 1 1      1 1 1 1      1 1 1 1
0 0 1 1      0 0 1 1      0 0 1 1
0 0 1 1 ~    0 0 1 1 ~    0 0 1 1
0 0 0 0      0 0 0 1      0 0 0 1
0 1 0 1      0 1 0 0      0 0 0 1

```

In all cases, this substage generates the next $\frac{m}{2}$ values of $e(1)$.

Stage PiB ($2 \leq i < n$): Notice that in the i th cycle, the last vertex is already an 1-vertex. We only need to move the other 1-vertices, starting from the right end, to column $\frac{m}{2} + 1$ through $m-1$: switch the labels of $a_{i,m-2}$ with $a_{i,m-1}$, then $a_{i,m-4}$ with $a_{i,m-3}$, ..., until we obtain

```

1 1 ... 1 1 1 ... 1 1      1 1 ... 1 1 1 ... 1 1
0 0 ... 0 1 1 ... 1 1      0 0 ... 0 1 1 ... 1 1
: : ... : : : ... : :      : : ... : : : ... : :
0 0 ... 0 1 1 ... 1 1      0 0 ... 0 1 1 ... 1 1
0 0 ... 0 0 0 ... 0 0      or 0 0 ... 0 0 0 ... 0 0
0 1 ... ... ... 0 1        1 0 ... ... ... 1 0
: : ... ... ... : :        : : ... ... ... : :
: : ... ... ... : :        : : ... ... ... : :
: : ... ... ... : :        : : ... ... ... : :

```

Each time the value of $e(1)$ is increased by 1, hence $\frac{m}{2} - 1$ more values of $e(1)$ are generated. Together, each of Stage i generates $m-1$ values of $e(1)$. Hence the last value of $e(1)$ generated after $n-1$ stages is $(n-1)(m-1)+1$.

Stage n : Switch the labels of a_{11} with a_{nm} . The value of $e(1)$ is decreased by 1. However, when we switch the labels of a_{12} with $a_{n,m-1}$, the value of $e(1)$ will be increased by 1. Continue switching a_{13} with $a_{n,m-2}$, a_{14} with $a_{n,m-3}$, ..., until $a_{1,m/2}$ with $a_{n,(m/2)+1}$:

```

0 0 ... 0 0 1 1 ... 1 1
0 0 ... 0 0 1 1 ... 1 1
0 0 ... 0 0 1 1 ... 1 1
: : ... : : : ... : :
0 0 ... 0 0 1 1 ... 1 1
0 0 ... 0 0 1 1 ... 1 1

```

Since only $\frac{m}{2} - 2$ new values of $e(1)$ are generated in Stage n , the last value of $e(1)$ is $(n-1)(m-1) + \frac{m}{2} - 1 = n(m-1) - \frac{m}{2}$.

Summary: $\{q\} \cup \{q-4, q-6, \dots, 2n\} \subseteq \text{PCI}(C_m \times P_n)$.

3.2 Algorithm C

Note that Algorithm P generates consecutive values of $e(1)$. This ensures that all the values within the bounds are attainable. In this regard, we need not concern too much about what values $e(1)$ will generate in each stage, provided their values are consecutive.

Algorithm C is very similar to Algorithm P, except that it has only $n-1$ stages. In particular, when $n=2$, there is only one stage, and $\text{PCI}(C_m \times P_2) \supseteq \{q\} \cup \{q-4, q-6, \dots, m\}$. Another main difference is, when $i > 1$ is odd, the 1-vertices are placed before the 1-vertices from the previous stage, so that the m vertices from the two stages will fill up the whole cycle. This, however, creates a slight problem. When the last 1-vertex is filled into the last position in the cycle, the value of $e(1)$ increases by 2 instead of 1. See below for an example:

```

1 1 1 1 1 1      1 1 1 1 1 1      1 1 1 1 1 1
0 1 0 1 1 1      0 0 1 1 1 1      0 1 1 1 1 1
0 0 1 0 1 ~    0 0 1 0 1 ~    0 0 0 0 1 ~
0 0 0 0 0      0 0 0 0 0      0 0 0 0 0
0 1 0 1 0 1      0 1 0 1 0 1      0 1 0 1 0 1

```

If there is still some rows of 0- and 1-vertices that have not been processed yet, we can temporarily move some of the 1-vertices around to create an additional 1-edge so as to generate the next value of $e(1)$. For instance, in the example above, insert two extra switchings between the last two:

```

1 1 1 1 1 1      1 1 1 1 1 1      1 1 1 1 1 1
0 1 1 1 1 1      0 1 1 1 1 1      0 1 1 1 1 1
0 0 0 0 1 ~    0 0 0 0 1 ~    0 0 0 0 1 ~
0 0 0 0 0      0 1 0 0 0      0 0 0 0 0
0 1 0 1 0 1      0 1 0 1 0 0      0 1 0 1 0 1

```

However, this step is not possible in the last stage if n is even, which means the second to the largest possible value of $e(1)$ cannot be obtained. Consequently, for $n \geq 3$,

$$\text{PCI}(C_m \times P_n) \supseteq \begin{cases} \{q\} \cup \{q-4, q-6, \dots, m+4\} \cup \{m\} & \text{if } n \text{ is even,} \\ \{q\} \cup \{q-4, q-6, \dots, m+2\} & \text{if } n \text{ is odd.} \end{cases}$$

Notice that these are the values of $e(0) - e(1)$ that are generated by Algorithm C. Some of them could also be generated by Algorithm P. In particular, when n is even, the missing value $m+2$ may be attainable from Algorithm P. Nevertheless, these two algorithms generate the largest ranges of values of $e(0) - e(1)$, and the final configurations in them are the only ways to obtain the smallest possible values in $e(0) - e(1)$. Starting with

the final configuration in Algorithm C, no matter from where we move the 1-vertex, the value of $e(0) - e(1)$ will be increased by at least 4. Therefore, when n is even, if Algorithm P does not generate $e(0) - e(1) = m + 2$, we deduce that $m + 2 \notin \text{PCI}(C_m \times P_n)$.

3.3 The PCI Set

Theorem 3.1 *Let $q = m(2n - 1)$. When m and n are both even, the PCI set of $C_m \times P_n$ is*

$$\begin{cases} \{q\} \cup \{q - 4, q - 6, \dots, 2n\} & \text{if } m \geq 2n, \\ \{q\} \cup \{q - 4, q - 6, \dots, m\} & \text{if } m = 2n - 2, \\ \{q\} \cup \{q - 4, q - 6, \dots, m + 4\} \cup \{m\} & \text{if } m < 2n - 2. \end{cases}$$

Proof. The result for $n = 2$ is immediate, so we may assume $n \geq 3$. Let S_P and S_C be the sets of values of $e(1)$ that Algorithm P and Algorithm C generate. It is clear that $S_C \subseteq S_P$ if $m \geq 2n$, and $S_P \subseteq S_C$ if $m < 2n - 2$. If $m = 2n - 2$, then $2n = m + 2$ is the value that S_C misses, hence the PCI set is $S_C \cup \{m + 2\}$. \square

Similar argument leads to the following result.

Theorem 3.2 *Let $q = m(2n - 1)$. For even m and odd $n \geq 3$,*

$$\text{PCI}(C_m \times P_n) = \begin{cases} \{q\} \cup \{q - 4, q - 6, \dots, 2n\} & \text{if } m \geq 2n - 2, \\ \{q\} \cup \{q - 4, q - 6, \dots, m + 2\} & \text{if } m < 2n - 2. \end{cases}$$

4 The Case of Odd m

When m is odd, the initial vertex labeling has $e(1) = \lfloor \frac{m}{2} \rfloor$, hence the largest PC index is $q' = m(2n - 1) - 2\lfloor \frac{m}{2} \rfloor$. The first switch in Stage P1A does not increase the value of $e(1)$ by 2 since $a_{1,m}a_{1,1}$ is not an 1-edge. In addition, after Stage P1A, we have

$$\begin{array}{cccccccc} 1 & 0 & 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}$$

At the beginning of Stage P1B, we first swap $a_{2,m}$ with $a_{1,m}$ to keep the value of $e(1)$ unchanged. The rest of Stage P1B will continue as usual.

For $2 \leq i < n$, Stage P_i executes in an almost identical manner as if m is even, except that when i is odd, $\lfloor \frac{m}{2} \rfloor$ vertices on the $(i + 1)$ st cycle are 1-vertices. At the beginning of Stage P_iA , we need to move $a_{i+1,1}$ under

the bottom 1-vertex in the middle column to ensure that the value of $e(1)$ is increased by 1. Here is an example:

$$\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}$$

In addition, when $m = 3$, Stage P2B is not needed for each $i > 1$.

If n is even, we find that Algorithm P generates all the potential values of $e(1)$ within the bounds, hence $\{q', q' - 2, \dots, 2n + 1\} \subseteq \text{PCI}(C_m \times P_n)$.

If n is odd, when Algorithm P terminates, the top $\lfloor \frac{m}{2} \rfloor$ vertices in the middle column are 1-vertices, hence $e(1) = m(n - 1) - \frac{m-3}{2}$. Notice that $v(0) - v(1) = 1$. However, when m is odd, we could have $v(1) - v(0) = 1$. Therefore we need an additional step to account for the extra 1-vertex. In Stage E, the extra 1-vertex can be placed either under the bottom 1-vertex in the middle column, or adjacent to the first 1-vertex on the bottom cycle. The two extra values of $e(1)$ give us the full range of values $e(1)$. We find $\{q', q' - 2, \dots, 2n - 1\} \subseteq \text{PCI}(C_m \times P_n)$.

If n is even, Algorithm C terminates with the second to the largest value of $e(1)$ missing. When n is odd, the full range of values of $e(1)$ within the bounds is attainable. Hence $\text{PCI}(C_m \times P_n)$ contains the following subset:

$$\begin{cases} \{q', q' - 2, \dots, m + 4\} \cup \{m\} & \text{if } n \text{ is even,} \\ \{q', q' - 2, \dots, m\} & \text{if } n \text{ is odd.} \end{cases}$$

Merging the results from the two algorithms yields our last two results. We omit the proofs because they are identical to the ones we used above.

Theorem 4.1 *Let $q' = m(2n - 1) - 2\lfloor \frac{m}{2} \rfloor$. For odd m and even n ,*

$$\text{PCI}(C_m \times P_n) = \begin{cases} \{q', q' - 2, \dots, 2n + 1\} & \text{if } m \geq 2n + 1, \\ \{q', q' - 2, \dots, m\} & \text{if } m = 2n - 1, \\ \{q', q' - 2, \dots, m + 4\} \cup \{m\} & \text{if } m < 2n - 1. \end{cases}$$

Theorem 4.2 *Let $q' = m(2n - 1) - 2\lfloor \frac{m}{2} \rfloor$. When m and n are both odd and at least 3,*

$$\text{PCI}(C_m \times P_n) = \begin{cases} \{q', q' - 2, \dots, 2n - 1\} & \text{if } m \geq 2n - 1, \\ \{q', q' - 2, \dots, m\} & \text{if } m < 2n - 1. \end{cases}$$

5 Closing Remarks

The results stated above in Sections 2 and 3 agree with the examples given in [2] for $m = 3, 4$. The technique should also work on the torus $C_m \times C_n$. We invite the readers to determine the exact value of $\text{PCI}(C_m \times C_n)$.

References

- [1] I. Cahit, Cordial graphs: a weaker version of graceful and harmonious graphs, *Ars Combin.* **23** (1987), 201–207.
- [2] C.-C. Chou, S.-M. Lee and Y.-C. Wang, On the product-cordial index sets of $C_m \times P_n$, manuscript.
- [3] H. Kwong and S.-M. Lee, On friendly index sets of generalized books, *J. Combin. Math. Comput.* **66** (2008), 43–58.
- [4] H. Kwong, S.-M. Lee and H.K. Ng, On friendly index sets of 2-regular graphs, *Discrete Math.* **308** (2008), 5522–5532.
- [5] H. Kwong, S.-M. Lee and Y.-C. Wang, On friendly index sets of $(p, p+1)$ -graphs, manuscript.
- [6] S.-M. Lee and H.K. Ng, on friendly index sets of bipartite graphs, *Ars Combin.* **86** (2008), 257–271.
- [7] A.N.-T. Lee, S.M. Lee and H.K. Ng, On balance index sets of graphs, *J. Combin. Math. Comput.* **66** (2008), 135–150.
- [8] S.M. Lee, A. Liu and S.K. Tan, On balanced graphs, *Congr. Numer.* **87** (1992), 59–64.
- [9] E. Salehi, PC-labeling of a graphs and its PC-set, *Bull. Inst. Combin. Appl.* **58** (2010), 112–121.
- [10] E. Salehi and S.-M. Lee, On friendly index sets of trees, *Congr. Numer.* **178** (2006), 173–183.
- [11] M. Sundaram, R. Ponraj, and S. Somasundaram, On graph labeling parameters, *J. Discrete Math. Sci. Cryptogr.* **11** (2008), 219–229.
- [12] M. Sundaram, R. Ponraj and S. Somasundaram, Product cordial labeling of graphs, *Bull. Pure Appl. Sci. Sect. E Math. Stat.* **23** (2004), 155–163.
- [13] M. Sundaram, R. Ponraj, and S. Somasundaram, Some results on total product cordial labeling of graphs, *J. Indian Acad. Math.* **28** (2006), 309–320.

FACTORIZATION OF COMPLETE GRAPHS INTO LONG DOUBLE BROOMS

DALIBOR FRONCEK¹, MARIUSZ MESZKA²

¹University of Minnesota Duluth, U.S.A., ²AGH-UST Krakow, Poland

ABSTRACT. A double broom $B_{2n}(k_1, k_2)$ is a pair of stars K_{1, k_1}, K_{1, k_2} whose central vertices are joined by a path with $2n - k_1 - k_2 - 1$ edges. In this paper we investigate factorizations of complete graphs into spanning double brooms. We solve completely the case when $1 \leq k_1 \leq k_2 \leq \lfloor \frac{n-1}{2} \rfloor$.

1. INTRODUCTION

Let G be a graph with at most n vertices. We say that the complete graph K_n has a G -decomposition if there are subgraphs $G_0, G_1, G_2, \dots, G_s$ of K_n , all isomorphic to G , such that each edge of K_n belongs to exactly one G_i . If G has exactly n vertices and none of them is isolated, then G is a factor of K_n and the decomposition is called a G -factorization of K_n .

While decompositions of complete graphs K_{2n+1} for $k \geq 1$ into trees with $n+1$ vertices have been studied extensively for more than forty years by many authors, very little was published on factorizations of complete graphs into isomorphic spanning trees until recently. A simple arithmetic condition shows that only complete graphs with an even number of vertices can be factorized into spanning trees. In [1], Eldergill developed a method of T -factorization of K_{2n} into symmetric trees using two types of graph labelings based on labelings introduced earlier by Rosa [17]. More general methods for n odd were developed by the first author [2, 4] and by the first author with Kubesa [6] and for n even by Kovarova [8, 9]. Many families of trees were investigated by Kubesa [10–16]. A complete characterization for caterpillars of diameter 4 was found by Kubesa [10], Kovarova [8, 9], and

2000 *Mathematics Subject Classification.* 05C70, 05C78.
Key words and phrases. Graph factorization, graph labeling.