Scientific Research Publishing

Open Journal of Discrete Mathematics, 2022, 12, 64-77
https://www.scirp.org/journal/ojdm
ISSN Online: 2161-7643
ISSN Print: 2161-7635

N-Set Distance-Labelings for Cycle Graphs

Alissa Shen[1], Jian Shen[2]

[1]St. Stephen's Episcopal School, Austin, TX, USA
[2]Department of Mathematics, Texas State University, San Marcos, TX, USA
Email: shenalissa@gmail.com

How to cite this paper: Shen, A. and Shen, J. (2022) N-Set Distance-Labelings for Cycle Graphs. *Open Journal of Discrete Mathematics*, **12**, 64-77.
https://doi.org/10.4236/ojdm.2022.123005

Received: May 18, 2022
Accepted: July 26, 2022
Published: July 29, 2022

Abstract

Let $G=(V,E)$ be a graph and C_m be the cycle graph with m vertices. In this paper, we continued Yeh's work [1] on the distance labeling of the cycle graph C_m. An n-set distance labeling of a graph G is the labeling of the vertices (with n labels per vertex) of G under certain constraints depending on the distance between each pair of the vertices in G. Following Yeh's notation [1], the smallest value for the largest label in an n-set distance labeling of G is denoted by $\lambda_1^{(n)}(G)$. Basic results were presented in [1] for $\lambda_1^{(2)}(C_m)$ for all m and $\lambda_1^{(n)}(C_m)$ for some m where $n\geq 3$. However, there were still gaps left unstudied due to case-by-case complexities. For these uncovered cases, we proved a lower bound for $\lambda_1^{(n)}(C_m)$. Then we proposed an algorithm for finding an n-set distance labeling for $n\geq 3$ based on our proof of the lower bound. We verified every single case for $n=3$ up to $n=500$ by this same algorithm, which indicated that the upper bound is the same as the lower bound for $n\leq 500$.

Keywords

Graph, Channel Assignment, Distance Labeling, Cycle Graph

1. Introduction

The mathematical field of Graph Theory first came into prominence after Euler published his paper on the Königsberg 7 bridges problem. Following that, numerous mathematicians expanded the field and created new classifications of graphs as well as new problems, such as the famous 4 color problem (which is a special case of the T-coloring of graphs). There have long been many real world applications from graph theory, such as radio transmission and map coloring. The issue that arises with these types of problems is a conflict of organization.

Vertices must be carefully positioned so that edge crossing can be avoided under certain circumstances, and with increasingly complicated situations, comes a need for greater detail and precision in its graph labeling. For example, in the case of radio transmissions, signals within close vicinity cannot overlap and therefore must be assigned to avoid conflicts. Each radio station is assigned a unique radio frequency. The closer the locations for radio stations are, the greater the risk for signal interference. To solve this, our paper will signify radio stations as vertices and radio frequencies as labels and make it into a graphing problem.

The problem of T-coloring of graphs has long been heavily researched. Starting in the 1980s, a new application of this problem was the issue of channel frequency assignment [2] [3] [4] [5]. Another wave of research on this topic began in 1990s, with an effort to more efficiently assign radio channels by using non negative integers, whereas channels at close distances receive frequencies further apart in value than channels at far apart distances [6] [7] [8] [9]. This should prevent signal interference between radio stations close to each other. In particular, Griggs and Yeh's work in [6] on graph labeling with a condition at distance 2 vertices is of great importance. They proposed a new way to label a graph. Given a graph $G=(V,E)$ and a positive integer d, an $L_d(2,1)$ labeling of G is a real-valued function $f:V\to[0,\infty)$ such that, for any two vertices x and y in G,

$$|f(x)-f(y)|\geq\begin{cases}2d & \text{if } x \text{ and } y \text{ are adjacent;}\\ d & \text{if } x \text{ and } y \text{ have distance two apart.}\end{cases}$$

The $L_d(2,1)$-labeling number, denoted by $\lambda(G,d)$, is the smallest integer m such that G has an $L_d(2,1)$-labeling $f:V\to[0,m]$. When $d=1$, Griggs and Yeh proved that $\lambda(G,1)\leq\Delta^2+2\Delta$, where Δ is the maximum degree of G. Their result was later improved by Král' and Skrekovski [10], who proved $\lambda(G,1)\leq\Delta^2+\Delta-1$ when $\Delta\geq 2$.

Recently, Yeh continued his previous work in [6]: instead of assigning a single number to a vertex, he assigned a set of numbers to each vertex. Let $[k]$ be the integer set $\{0,1,2,\ldots,k\}$ and $\binom{[k]}{n}$ be the set of all n-subsets of $[k]$. Given a positive integer k and non-negative integers δ_1 and δ_2 with $\delta_1\geq\delta_2$, an $L^{(n)}(\delta_1,\delta_2)$-labeling is a function $f:V\to\binom{[k]}{n}$ such that, for any two vertices x and y in G,

$$|f(x)-f(y)|\geq\begin{cases}\delta_1 & \text{if } x \text{ and } y \text{ are adjacent;}\\ \delta_2 & \text{if } x \text{ and } y \text{ have distance two apart;}\end{cases}$$

where $|f(x)-f(y)|=\min\{|a-b|:a\in f(x) \text{ and } b\in f(y)\}$ is the defined set difference. **Figure 1** is an illustration of the above definition.

The smallest k satisfying the above definition is called the $L^{(n)}(\delta_1,\delta_2)$-labeling number of G and denoted by $\lambda^{(n)}(G;\delta_1,\delta_2)$. In particular, when $\lambda_1=\lambda_2=1$, $\lambda^{(n)}(G;1,1)$ is denoted by $\lambda_1^{(n)}(G)$ for short. Yeh applied this idea and obtained

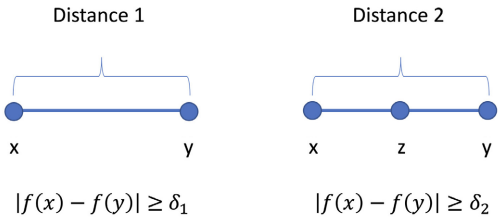


Figure 1. Illustration of $L^{(n)}(\delta_1,\delta_2)$ -labeling.

the exact value of $\lambda_1^{(n)}(G)$ for several types of graphs: trees, wheels, the square lattice, the hexagonal lattice, and the triangular lattice. For the cycle graphs C_m, Yeh has completely solved the problem for $\lambda_1^{(2)}(C_m)$. (Note: There was a typo in Corollary 3.5 [6], where the results there should be read only for $\lambda_1^{(2)}(C_m)$.) For the general cases when $n\geq 3$, Yeh proved that

$$\lambda_1^{(n)}(C_m)\begin{cases}=3n-1 & \text{if } m\equiv 0(\bmod 3);\\ =3n & \text{if } m\equiv 1(\bmod 3) \text{ and } m\geq 3n+1;\\ =3n & \text{if } m\equiv 2(\bmod 3) \text{ and } m\geq 6n+2;\\ \geq 3n & \text{otherwise.}\end{cases}$$

In this paper, we will expand on Yeh's work for $\lambda_1^{(n)}(C_m)$ with $n\geq 3$ on the following uncovered cases:

1) $m\equiv 1(\bmod 3)$ with $m=4,7,\cdots,3n-2$;
2) $m\equiv 2(\bmod 3)$ with $m=5,8,\cdots,6n-1$.

For the rest of this paper, we will refer any of the above unstudied cases as an uncovered case. Our goal is to fill in the gaps and find out the $\lambda_1^{(n)}(C_m)$ for the uncovered cases.

2. $\lambda_1^{(n)}(C_m)$ for $m=4$ and $m=5$

Yeh [6] proved that, if G is a graph with diameter 2 and order m, then $\lambda_1^{(n)}(G)=mn-1$. Since both C_4 and C_5 have diameter 2, as shown in **Figure 2**, the following is a direct consequence of Yeh's observation.

Lemma 1 $\lambda_1^{(n)}(C_4)=4n-1$ *and* $\lambda_1^{(n)}(C_5)=5n-1$.

3. Proving Lower Bound for $\lambda_1^{(n)}(C_m)$ for Uncovered Cases

In order to obtain a general formula for the lower bound of $\lambda_1^{(n)}(C_m)$ for uncovered m values given in Section 1, we first introduce the following lemma:

3.1. Upper Bound for Label Repetition

Lemma 2 *In any* $L^{(n)}(1,1)$ *-labeling of* C_m, *no label can be used more than* $\left\lfloor\frac{m}{3}\right\rfloor$ *times.*

Proof. We will prove the lemma by contradiction. Let $V(C_m)=\{v_1,v_2,\cdots,v_m\}$, where v_i is adjacent to v_{i+1} for each i with $v_{m+1}=v_1$, shown in **Figure 3**.

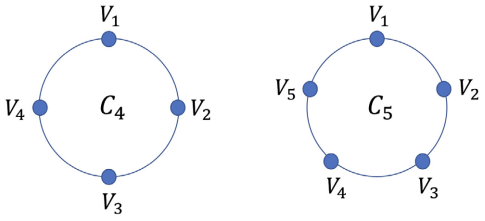


Figure 2. Cyclic graphs for $m=4$ and $m=5$.

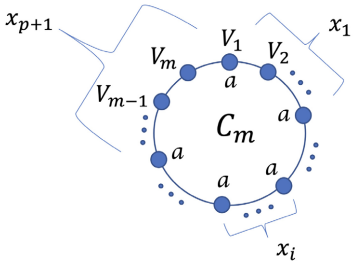


Figure 3. Cyclic graph with m vertices.

Let $p=\left\lfloor \frac{m}{3} \right\rfloor$. Without loss of generality, we may assume that a label "a" is assigned to v_1 and is used $p+1$ times in an $L^{(n)}(1,1)$-labeling of C_m. Let $x_i\,(i=1,2,\cdots,p+1)$ be the distance between a pair of adjacent "a" labels in the cyclic order (Please refer to **Figure 3**). The total distance is m because there are m vertices:

$$x_1+x_2+\cdots+x_{p+1}=m.$$

This implies that

$$\min_{1\le i\le p+1} x_i \le \frac{x_1+x_2+\cdots+x_{p+1}}{p+1}=\frac{m}{p+1}<\frac{m}{m/3}=3.$$

Thus the smallest x_i must be either 1 or 2; in other words, there are two vertices of C_m with distance at most two receiving the same label "a". This contradicts the distance labeling condition for $L^{(n)}(1,1)$, and thus Lemma 2 is proved. □

3.2. Lower Bound for $\lambda_1^{(n)}(C_m)$ for All Uncovered Cases

Theorem 1 *For all integers m and n with* $m,n\ge 3$,

$$\lambda_1^{(n)}(C_m)\ge\left\lceil \frac{mn}{p} \right\rceil-1,$$

where $p=\left\lfloor \frac{m}{3} \right\rfloor$.

Proof. In any $L^{(n)}(1,1)$-labeling of C_m, each vertex receives m labels. So there are together mn labels (counting the repetition of each label). By Lemma 2, each distinct label can only be repeatedly used at most p times. This implies that there are at least mn/p distinct labels. Also labels can start from 0 and $\lambda_1^{(n)}(C_m)$ is the largest label, we have $\lambda_1^{(n)}(C_m)\ge\lceil mn/p\rceil-1$. □

4. $\lambda_1^{(n)}(C_m)$ for $n=3$

We find all values for $\lambda_1^{(3)}(C_m)$.

Theorem 2

$$\lambda_1^{(3)}(C_m)=\begin{cases}8 & \text{if } m\equiv 0(\bmod 3);\\ 9 & \text{if } m\equiv 1(\bmod 3) \text{ with } m\geq 10 \text{ or } m\equiv 2(\bmod 3) \text{ with } m\geq 20;\\ 10 & \text{if } m=7,11,14,17;\\ 11 & \text{if } m=4,8;\\ 14 & \text{if } m=5.\end{cases}$$

Proof. By Yeh's work [6] and Lemma 1, we only need to prove the theorem for the uncovered cases when $m=7,8,11,14,17$. By Theorem 1,

$$\lambda_1^{(3)}(C_m)\geq\left\lceil\frac{3m}{\left\lfloor\frac{m}{3}\right\rfloor}\right\rceil-1=\begin{cases}10 & \text{if } m=7,11,14,17;\\ 11 & \text{if } m=8.\end{cases}$$

On the other hand, we can prove that the above lower bounds are also upper bounds by constructing actual label assignments.

For $\boldsymbol{m=7}$, the following label assignment meets the $L(1,1)$-labeling requirement. (Each block of three labels is assigned to a vertex of C_7 in the cyclic order. Similar label assignments are applied for other uncovered cases $m=8,11,14,17$ below.)

0 1 2 || 3 4 5 || 6 7 8 || 9 10 0 || 1 2 3 || 4 5 6 || 7 8 9 ||

This indicates: $\lambda_1^{(3)}(C_7)\leq 10$

For $\boldsymbol{m=8}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 || 3 4 5 || 6 7 8 || 9 10 11 || 0 1 2 || 3 4 5 || 6 7 8 || 9 10 11 ||

This indicates: $\lambda_1^{(3)}(C_8)\leq 11$

For $\boldsymbol{m=11}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 || 3 4 5 || 6 7 8 || 9 10 0 || 1 2 3 || 4 5 6 || 7 8 9 || 10 0 1 || 2 3 4 || 5 6 7 || 8 9 10 ||

This indicates: $\lambda_1^{(3)}(C_{11})\leq 10$

For $\boldsymbol{m=14}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 || 3 4 5 || 6 7 8 || 9 10 0 || 1 2 3 || 4 5 6 || 7 8 9 || 10 0 1 || 2 3 4 || 5 6 7 || 8 9 10 || 0 1 2 || 3 4 5 || 6 7 8 ||

This indicates: $\lambda_1^{(3)}(C_{14})\leq 10$

For $\boldsymbol{m=17}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 || 3 4 5 || 6 7 8 || 9 10 0 || 1 2 3 || 4 5 6 || 7 8 9 || 10 0 1 || 2 3 4 || 5 6 7 || 8 9 10 || 0 1 2 || 3 4 5 || 6 7 8 || 9 10 0 || 3 4 5 || 6 7 8 ||

This indicates: $\lambda_1^{(3)}(C_{17})\leq 10$

By combining the upper and lower bounds together, we conclude that $\lambda_1^{(3)}(C_m)=10$ when $m=7,11,14,17$ and that $\lambda_1^{(3)}(C_8)=11$. □

5. $\lambda_1^{(n)}(C_m)$ for $n=4$

In this section, we find all values for $\lambda_1^{(4)}(C_m)$.

Theorem 3

$$\lambda_1^{(4)}(C_m)=\begin{cases}11 & \text{if } m\equiv 0(\bmod 3);\\ 12 & \text{if } m\equiv 1(\bmod 3) \text{ with } m\geq 13 \text{ or } m\equiv 2(\bmod 3) \text{ with } m\geq 26;\\ 13 & \text{if } m=7,10,14,17,20,23;\\ 14 & \text{if } m=11;\\ 15 & \text{if } m=4,8;\\ 19 & \text{if } m=5;\end{cases}$$

Proof. By Yeh's work [6] and Lemma 1, we only need to prove the theorem for the uncovered cases when $m=7,8,10,11,14,17,20,23$. By Theorem 1,

$$\lambda_1^{(4)}(C_m)\geq\left\lceil\frac{4m}{\left\lfloor\frac{m}{3}\right\rfloor}\right\rceil-1=\begin{cases}13 & \text{if } m=7,10,14,17,20,23;\\ 14 & \text{if } m=11;\\ 15 & \text{if } m=8;\end{cases}$$

On the other hand, we can prove that the above lower bounds are also upper bounds by constructing actual label assignments.

For $\boldsymbol{m=7}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 ||

This indicates: $\lambda_1^{(4)}(C_7)\leq 13$

For $\boldsymbol{m=8}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 14 15 || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 14 15 ||

This indicates: $\lambda_1^{(4)}(C_8)\leq 15$

For $\boldsymbol{m=10}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 ||

This indicates: $\lambda_1^{(4)}(C_{10})\leq 13$

For $\boldsymbol{m=11}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 14 0 || 1 2 3 4 || 5 6 7 8 || 9 10 11 12 || 13 14 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 ||

This indicates: $\lambda_1^{(4)}(C_{11})\leq 14$

For $\boldsymbol{m=14}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 ||

This indicates: $\lambda_1^{(4)}(C_{14}) \le 13$

For $\boldsymbol{m=17}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 || || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 ||

This indicates: $\lambda_1^{(4)}(C_{17}) \le 13$

For $\boldsymbol{m=20}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 || || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 4 5 6 7 || 8 9 10 11 ||

This indicates: $\lambda_1^{(4)}(C_{20}) \le 13$

For $\boldsymbol{m=23}$, the following label assignment meets the $L(1,1)$-labeling requirement:

0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 || 12 13 0 1 || 2 3 4 5 || 6 7 8 9 || 10 11 12 13 || 0 1 2 3 || 4 5 6 7 || 8 9 10 11 || **12 13 2 3** || 4 5 6 7 || 8 9 10 11 || **12 13 2 3** || 4 5 6 7 || 8 9 10 11 ||

This indicates: $\lambda_1^{(4)}(C_{23}) \le 13$

Here is the general process of label assignment for $n=4$ and $m=23$, whose complete data set is: 0 1 2 3 4 5 6 7 8 9 10 11 12 13.

1) We iterate this data set repeatedly by combining 4 labels in a group and assigning this group to one node.

2) This process continues until we finish the 23rd node.

Please note that there is something special with the label assignment for $m=23$. For the last two repetitions of the data set, the "0" and "1" were taken out, indicated by the **bold** numbers. This is to avoid violation of $L(1,1)$-labeling requirements. This algorithm will be explained in detail in the next section.

By combining the upper and lower bounds together, we conclude that $\lambda_1^{(4)}(C_m)=13$ when $m=7,10,14,17,20,23$, $\lambda_1^{(4)}(C_{11})=14$ and $\lambda_1^{(4)}(C_8)=15$. □

There is not a general formula for the uncovered cases. For any given *n* value, we need to work on each corresponding uncovered *m* value individually.

6. Algorithm for Finding Upper Bound for All Uncovered Cases

We have proven the lower bound for all the uncovered cases. In this section, we will work on the upper bound. We will use the prove-by-construction method and provide a general algorithm to find label assignment for all the uncovered cases based on the proved lower bound. The finding of such label assignment gives an upper bound to $\lambda_1^{(n)}(C_m)$.

6.1. Sample Case of $\lambda_1^{(6)}(C_{35})$

We will use the following case to illustrate how we will construct the label assignment for all the vertices based on the lower bound.

$n = 6$ (number of labels per vertex)

$m = 35$ (number of vertices)

The lower bound is calculated according to Section 3.2:

$p = 11$ (max number of repetitions per label)

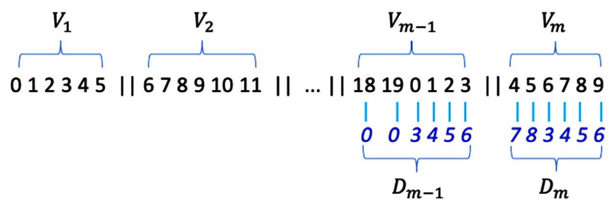
$L = 19$ (lower bound of $L(1,1)$ for $\lambda_1^{(6)}(C_{35})$)

Data set available for the labels to be used for each vertex: 0, 1, 2, 3, … 16, 17, 18, 19.

Initial label assignment:

0 1 2 3 4 5 || 6 7 8 9 10 11 || 12 13 14 15 16 17 || 18 19 0 1 2 3 || 4 5 6 7 8 9 || 10 11 12 13 14 15 || 16 17 18 19 0 1 || 2 3 4 5 6 7 || 8 9 10 11 12 13 || 14 15 16 17 18 19 || 0 1 2 3 4 5 || 6 7 8 9 10 11 || 12 13 14 15 16 17 || 18 19 0 1 2 3 || 4 5 6 7 8 9 || 10 11 12 13 14 15 || 16 17 18 19 0 1 || 2 3 4 5 6 7 || 8 9 10 11 12 13 || 14 15 16 17 18 19 || 0 1 2 3 4 5 || 6 7 8 9 10 11 || 12 13 14 15 16 17 || 18 19 0 1 2 3 || 4 5 6 7 8 9 || 10 11 12 13 14 15 || 16 17 18 19 0 1 || 2 3 4 5 6 7 || 8 9 10 11 12 13 || 14 15 16 17 18 19 || 0 1 2 3 4 5 || 6 7 8 9 10 11 || 12 13 14 15 16 17 || 18 19 0 1 2 3 || 4 5 6 7 8 9 ||

Initial label assignment for the first and last two vertices:



Violations found:

- 4 labels (0, 1, 2, 3) in V_{m-1} are overlapping with that of V_1: distance 2 type of violation;
- 2 labels (4, 5) in V_m are overlapping with that of V_1: distance 1 type of violation, which requires more corrective shifts towards the left;
- 4 labels (6, 7, 8, 9) in V_m are overlapping with that of V_2: distance 2 type of violation.

Corrections:

(1) Use two arrays D_{m-1} and D_m to record the minimum displacement needed for each label position in V_{m-1} and V_m, in case a corrective shift (to the left) is needed for a violation. $D_m[i] = 0$ indicates no violation for the *ith* position in V_m. $D_{m-1}[i] = 0$ indicates no violation for the *ith* position in V_{m-1}.

(2) Assignment of displacement values for V_{m-1}:

- $D_{m-1}[0] = 0$ because $V_{m-1}[0] = 18$, and 18 does not overlap with any labels in V_1;
- $D_{m-1}[1] = 0$ because $V_{m-1}[1] = 19$, and 19 does not overlap with any

labels in V_1;

- $D_{m-1}[2]=3$ because $V_{m-1}[2]=0$, and 0 overlaps with a label in V_1; 0 needs to shift at least 3 positions towards left to resolve this violation;
- $D_{m-1}[3]=4$ because $V_{m-1}[3]=1$, and 1 overlaps with a label in V_1; 1 needs to shift at least 4 positions towards left to resolve this violation;
- $D_{m-1}[4]=5$ because $V_{m-1}[4]=2$, and 2 overlaps with a label in V_1; 2 needs to shift at least 5 positions towards left to resolve this violation;
- $D_{m-1}[5]=6$ because $V_{m-1}[5]=3$, and 3 overlaps with a label in V_1; 3 needs to shift at least 6 positions towards left to resolve this violation;

(3) Assignment of displacement values for V_m:

- $D_m[0]=7$ because $V_m[0]=4$, and 4 overlaps with a label in V_1; 4 needs to shift at least 7 positions towards left to resolve this distance 1 type of violation;
- $D_m[1]=8$ because $V_m[1]=5$, and 5 overlaps with a label in V_1; 5 needs to shift at least 8 positions towards left to resolve this distance 1 type of violation;
- $D_m[2]=3$ because $V_m[2]=6$, and 6 overlaps with a label in V_2; 6 needs to shift at least 3 positions towards left to resolve this violation;
- $D_m[3]=4$ because $V_m[3]=7$, and 7 overlaps with a label in V_2; 7 needs to shift at least 4 positions towards left to resolve this violation;
- $D_m[4]=5$ because $V_m[4]=8$, and 8 overlaps with a label in V_2; 8 needs to shift at least 5 positions towards left to resolve this violation;
- $D_m[5]=6$ because $V_m[5]=9$, and 9 overlaps with a label in V_2; 9 needs to shift at least 6 positions towards left to resolve this violation;

(4) $D_{\max}=\max\left(D_{m-1}[i],D_m[i]\right)$ for all *i*.

(5) Use of $D_{\max}$ for corrections.

In order to accomplish the corrective displacement for the above labels, we will shift every label $D_{\max}$ positions toward the left for the last two vertices. This indicates that we need to create $D_{\max}$ extra positions from the previous vertices. A divide-and-conquer strategy is used.

p = 11 (max number of repetitions per label)
L = 19 (lower bound of $L(1,1)$ for C_m)
r = 2 (number of labels which can be safely removed without violation of $L(1,1)$)
D_{max} = 996
Number of shifts = D_{max} / r = 8 / 2 = 4

In this case, a complete data set has 20 numbers (0, 1, 2, 3, …, 16, 17, 18, 19) available to use for the label assignment. We can safely remove 2 numbers from each data set to help with the shift-left task, because three consecutive vertices need a total of 18 different labels to avoid violation of $L(1,1)$ rules. These two numbers can be any number from 0 to 19. For example,

Complete data set : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Shrinked data set : 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 (6.1)
Alternative shrinked data set : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Without loss of generality, we will remove the first two numbers (0, 1) from

each of the last 4 data sets as indicated in (6.1), creating a total of 8 extra positions for the later data to fill in. This will satisfy the $D_{\max}$ required for resolving violations.

New label assignment:

0 1 2 3 4 5 || 6 7 8 9 10 11 || 12 13 14 15 16 17 || 18 19 0 1 2 3 || 4 5 6 7 8 9 || 10 11 12 13 14 15 || 16 17 18 19 0 1 || 2 3 4 5 6 7 || 8 9 10 11 12 13 || 14 15 16 17 18 19 || 0 1 2 3 4 5 || 6 7 8 9 10 11 || 12 13 14 15 16 17 || 18 19 0 1 2 3 || 4 5 6 7 8 9 || 10 11 12 13 14 15 || 16 17 18 19 0 1 || 2 3 4 5 6 7 || 8 9 10 11 12 13 || 14 15 16 17 18 19 || 0 1 2 3 4 5 || 6 7 8 9 10 11 || 12 13 14 15 16 17 || **18 19** * * **2 3 4 5** || 6 7 8 9 10 11 || 12 13 14 15 16 17 || **18 19** * * **2 3 4 5** || 6 7 8 9 10 11 || 12 13 14 15 16 17 || **18 19** * * **2 3 4 5** || 6 7 8 9 10 11 || 12 13 14 15 16 17 || **18 19** * * **2 3 4 5** || 6 7 8 9 10 11 || 12 13 14 15 16 17 ||

New label assignment for the first and last two vertices:

0 1 2 3 4 5 || 6 7 8 9 10 11 || … || 6 7 8 9 10 11 || 12 13 14 15 16 17

We now have no more violations! Please note the spots marked with * within the groups of **bold** numbers. These are the positions which were originally occupied by 0 s and 1 s.

Therefore, we successfully constructed the label assignment with the lower bound. This means that the lower bound is also the upper bound for $n = 6$ with $m = 35$. Thus, we have $\lambda_1^{(6)}(C_{35}) = 19$.

6.2. General Algorithm for Finding Label Assignment Based on Lower Bound

In this section, we provide a general algorithm to find the label assignment for the purpose of finding the upper bound of $\lambda_1^{(n)}(C_m)$ with $n \geq 3$ for the following uncovered cases:

$$\begin{aligned} &1)\ m \equiv 1 \pmod 3 \text{ with } m = 4, 7, \cdots, 3n-2; \text{ and} \\ &2)\ m \equiv 2 \pmod 3 \text{ with } m = 5, 8, \cdots, 6n-1. \end{aligned} \tag{6.2}$$

Number of uncovered vertices:

$$M = (3n-2-7)/3+1+(6n-1-8)/3+1 = (n-2)+(2n-2) = 3n-4$$

For each m value listed in (6.2):

(1) Calculate the following parameters:

$p = \left\lfloor \frac{m}{3} \right\rfloor$; p is the maximum number of repetitions per label.

$L = \left\lceil \frac{mn}{p} \right\rceil - 1$; L is the lower bound for $\lambda_1^{(n)}(C_m)$.

$r = L - (3n-1)$; r is the number of labels which can be safely removed from each data set without violation of $L(1,1)$.

(2) Initialize index, whose value ranges from 0 to L; One complete data set runs from 0 to L.

(3) Initial label assignment:

For each of the m vertices

For each of the n positions available for labels

Assign a proper index to each label position;

Increase index by 1;

If (index > L) reset index =0;

Complexity: mn.

(4) Print out the initial label assignment for the first and last two vertices: V_1, V_2, V_{m-1}, V_m.

Complexity: $4n$.

(5) Check possible violation of requirement for $L(1,1)$: we only need to check V_{m-1} against V_1, and V_m against both V_1 and V_2.

Use two arrays D_{m-1} and D_m to record the minimum displacement needed for each label position in V_{m-1} and V_m, in case a corrective shift (to the left) is needed for a violation. $D_m[i]=0$ indicates no violation for the ith position in V_m. $D_{m-1}[i]=0$ indicates no violation for the ith position in V_{m-1}.

Step 1: Compare each label of V_{m-1} with that of V_1. Any overlapped labeling indicates a violation of distance 2 type, and the corresponding $D_{m-1}[i]$ will be recorded. Complexity: n^2.

Step 2: Compare each label of V_m with that of V_1. Any overlapped labeling indicates a violation of distance 1 type which requires more corrective shifts toward left, and the corresponding $D_m[i]$ will be recorded. Complexity: n^2.

Step 3: Compare each label of V_m with that of V_2. Any overlapped labeling indicates a violation of distance 2 type, and the corresponding $D_m[i]$ will be recorded. Complexity: n^2.

Step 4: $D_{\max} = \max\left(D_{m-1}[i], D_m[i]\right)$. This is the minimum corrective displacement needed to shift every label of the last two vertices toward left to avoid $L(1,1)$ violation; Complexity: $2n$.

(6) If $D_{\max} = 0$, no violations; label assignment is done for this m value.

(7) If $D_{\max} != 0$, adjust label assignment:

$$num = \left\lceil \frac{D_{\max}}{r} \right\rceil;$$

For the first $p - num$ data set, use index values from 0 to L to assign labels;

For the rest of data set, only use index values from r to L to assign labels;

Complexity: mn.

(8) Print out the updated label assignment for the first and last two vertices: V_1, V_2, V_{m-1}, V_m.

Complexity: $4n$.

(9) Check possible violation of requirement for $L(1,1)$:

We only need to check V_{m-1} against V_1, and V_m against both V_1 and V_2. Use similar methods described in step 1 to step 4 listed above.

Complexity: $3n^2+2n$.

(10) If $D_{\max}=0$, no violations; the updated label assignment works for this m value. Otherwise, print out error message: Failed label assignment for $n=*, m=*$.

6.3. Runtime and Complexity

From Section 6.2, we can figure out that the runtime complexity to calculate $\lambda_1^{(n)}(C_m)$ and find a no-violation cyclic label assignment for a given pair of (n,m) is:

$$2mn+6n^2+12n \tag{6.3}$$

Applying (6.3), the runtime complexity to calculate $\lambda_1^{(n)}(C_m)$ and find a no-violation cyclic label assignment for all the uncovered m values with a given n value is:

$$\sum_{m=7,10,\cdots}^{3n-2}\left(2mn+6n^2+12n\right)+\sum_{m=8,11,\cdots}^{6n-1}\left(2mn+6n^2+12n\right)=33n^3+13n^2-72n \tag{6.4}$$

The complexity of (6.4) is in the order of n^3.

When we run the program for all the n values from 3 to N, the runtime complexity is:

$$\sum_{n=3}^{N}\left(33n^3+13n^2-72n\right)=\frac{33N^4}{4}+\frac{125N^3}{6}-\frac{85N^2}{4}-\frac{203N}{6}-146 \tag{6.5}$$

The complexity of (6.5) is in the order of N^4.

Laptop configuration: 4 core, 8 thread, i7, 8 GB ram, 100 GB hard drive.

Runtime and size of output file when only the label assignment for the first and last two vertices were printed:

Measures	n = 3 to 10	n = 3 to 50	n = 3 to 100	n = 3 to 200	n = 3 to 500
Runtime: real	0m0.017s	0m2.815s	0m43.136s	67m48.236s	2536m59.302s (42+ hours)
Runtime: user	0m0.016s	0m2.804s	0m43.032s	67m46.960s	2536m0.697s
Runtime: sys	0m0.001s	0m0.005s	0m0.005s	0m0.725s	0m13.060s
Size of output file	0.07 MB	4.3 MB	32 MB	252 MB	4005 MB

Runtime and size of output file when label assignment for all the vertices were printed:

Measures	n = 3 to 10	n = 3 to 50	n = 3 to 100	n = 3 to 200	n = 3 to 500
Runtime: real	0m0.020s	0m4.135s	1m2.933s	73m26.612s	N/A
Runtime: user	0m0.016s	0m3.995s	1m2.124s	71m50.681s	N/A
Runtime: sys	0m0.004s	0m0.048s	0m0.496s	0m36.304s	N/A
Size of output file	0.18 MB	79 MB	1364 MB	22612 MB	N/A

6.4. Results

We tested our algorithm for every n value from 3 to 500 and every case was successful. A sample case for $n = 500$ is listed below:

```
n = 500 (number of labels per vertex)
m = 2999 (number of vertices)
p = 999 (max number of repetitions per label)
L = 1501 (lower bound of L(1,1) for C_m)
r = 2 (number of labels which can be safely removed without violation of L(1,1) )

### Initial first two and last two vertices:
0 1 2 3 4 ... 498 499 || 500 501 ... 998 999 || ... || 1006 1007 ... 1501 0 1 2 3 || 4 5 6 7 ... 502 503

### Found violation of requirement for L(1,1)

D_max = 996
Number of shifts = 498

### Updated first two and last two vertices:
0 1 2 3 4 ... 498 499 || 500 501 ... 998 999 || ... || 500 501 ... 998 999 || 1000 1001 ... 1498 1499

### No violation of L(1,1) requirement for the updated label assignment
```

Our results demonstrate that we can use the lower bound to construct the label assignment without violation of $L(1,1)$ for n values up to 500. This indicates the upper bound is the same as the lower bound. Therefore, we have:

Conjecture: For all integers m and n with $m, n \geq 3$, the smallest possible value for the largest label needed is given by

$$\lambda_1^{(n)}(C_m) = \left\lceil \frac{mn}{p} \right\rceil - 1 \tag{6.6}$$

where n is the number of labels per vertex, m is the number of vertices, and $p = \left\lfloor \frac{m}{3} \right\rfloor$ is the maximum number of repetitions per label.

We did not continue with n values bigger than 500 in our code, because the output file size went over 4 GB with $n = 500$ when we only print out the first and last two vertices, and the output file size went over 22 GB with $n = 200$ when we print out label assignment for every vertex. Practically, $n = 500$ is large enough to demonstrate the feasibility of our algorithm, and the correctness of (6.6).

7. Conclusions

In this paper, we studied the label assignment for cycle graphs, and we focused on the $\lambda_1^{(n)}(C_m)$ for the m values which were left uncovered by [1] due to their case-by-case complexities. We achieved a lower bound, expressed by Theorem 1, for all the uncovered m values. Moreover, based on the lower bound, we developed an algorithm in Section 6.2 to construct a no-violation label assignment for all vertices in order to find the corresponding upper bound. We ran every single case for $n \leq 500$ and obtained successful label assignments.

Theoretically, our algorithm can be used for any $n \geq 3$ to find a feasible label

assignment. Anybody interested in uncovered cases for $n > 500$ is welcome to verify the label assignment using our algorithm.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Yeh, R.K. (2021) A Note on n-Set Distance-Labelings of Graphs. *Open Journal of Discrete Mathematics*, **11**, 55-60. https://doi.org/10.4236/ojdm.2021.113005

[2] Cozzens, M.B. and Roberts, F.S. (1982) T-Colorings of Graphs and the Channel Assignment Problem. *Congressus Numerantium*, **35**, 191-208.

[3] Cozzens, M.B. and Wang, D.-I. (1984) The General Channel Assignment Problem. *Congressus Numerantium*, **41**, 115-129.

[4] Furedi, Z., Griggs, J.R. and Kleitman, D.J. (1989) Pair Labellings with Given Distance. *SIAM Journal on Discrete Mathematics*, **2**, 491-499. https://doi.org/10.1137/0402044

[5] Golumbic, M.C. (1980) Algorithmic Graph Theory and Perfect Graphs, Courant Institute of Mathematical Science, New York University. Academic Press, Vol. 1, 980. https://doi.org/10.1016/B978-0-12-289260-8.50008-X

[6] Griggs, J.R. and Yeh, R.K. (1992) Labelling Graphs with a Condition at Distance 2. *SIAM Journal on Discrete Mathematics*, **5**, 586-595. https://doi.org/10.1137/0405048

[7] Roberts, F.S. (1991) T-Colorings of Graphs: Recent Results and Open Problems. *Discrete Mathematics*, **93**, 229-245. https://doi.org/10.1016/0012-365X(91)90258-4

[8] Yeh, R.K. (2006) A Survey on Labeling Graphs with a Condition at Distance Two. *Discrete Mathematics*, **306**, 1217-1231. https://doi.org/10.1016/j.disc.2005.11.029

[9] Yeh, R.K. (2019) Pair L(2, 1)-Labelings of Infinite Graphs. *Discussiones Mathematicae*: *Graph Theory*, **39**, 257-269. https://doi.org/10.7151/dmgt.2077

[10] Král, D. and Skrekovski, R. (2003) A Theorem about the Channel Assignment Problem. *SIAM Journal on Discrete Mathematics*, **16**, 426-437. https://doi.org/10.1137/S0895480101399449