UPPER BOUNDS ON THE $L(2,1)$-LABELING NUMBER OF GRAPHS WITH MAXIMUM DEGREE Δ

ANDREW LUM

ADVISOR: DAVID GUICHARD

ABSTRACT. $L(2,1)$-labeling was first defined by Jerrold Griggs [Gr, 1992] as a way to use graphs to model the channel assignment problem proposed by Fred Roberts [Ro, 1988]. An $L(2,1)$-*labeling* of a simple graph G is a nonnegative integer-valued function $f : V(G) \to \{0,1,2,\ldots\}$ such that, whenever x and y are two adjacent vertices in $V(G)$, then $|f(x)-f(y)| \geq 2$, and, whenever the distance between x and y is 2, then $|f(x)-f(y)| \geq 1$. The $L(2,1)$-*labeling number* of G, denoted $\lambda(G)$, is the smallest number m such that G has an $L(2,1)$-labeling with no label greater than m. Much work has been done to bound $\lambda(G)$ with respect to the maximum degree Δ of G ([Cha, 1996], [Go, 2004], [Gr, 1992], [Kr, 2003], [Jo, 1993]). Griggs and Yeh [Gr, 1992] conjectured that $\lambda \leq \Delta^2$ when $\Delta \geq 2$.

In §1, we review the basics of graph theory. This section is intended for those with little or no background in graph theory and may be skipped as needed. In §2, we introduce the notion of $L(2,1)$-labeling. In §3, we give the labeling numbers for special classes of graphs. In §4, we use the greedy labeling algorithm to establish an upper bound for λ in terms of Δ. In §5, we use the Chang-Kuo algorithm to improve our bound. In §6, we prove the best known bound for general graphs.

1. PRELIMINARIES

1.1. **Basic Graph Theory.** We begin with the problem known to have started it all. The city of Königsburg, Prussia was located on the Pregel River and included two islands connected to each other and the mainland by seven bridges, as illustrated in Figure 1. The locals wondered if it was possible to start on one of the land masses, cross all seven bridges exactly once, and return to the starting point. Evidently, it is impossible. To help see why, we must first introduce the notion of a graph.

Definition 1.1. A *(simple) graph*[1] G is a pair (V,E) consisting of a set of vertices $V(G)$ and a set of edges $E(G)$, where edges are of the form $\{v_1,v_2\} \subseteq V(G)$ with $v_1 \neq v_2$. Two vertices $v_1,v_2 \in V(G)$ are said to be *adjacent* or *neighbors* if $\{v_1,v_2\} \in E(G)$.

Date: April 27, 2007.

Key words and phrases. $L(2,1)$-labeling, channel assignment problem, distance dependent graph coloring analogues.

[1]For the purposes of this paper, all graphs henceforth will be considered simple.

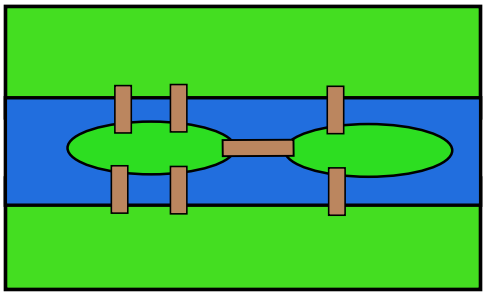

FIGURE 1. The Seven Bridges of Königsburg

Informally, then, a graph is a set of vertices, some of which are connected by edges. Graphs are most often visualized pictorially. Figure 2 gives an example of a graph on 6 vertices.

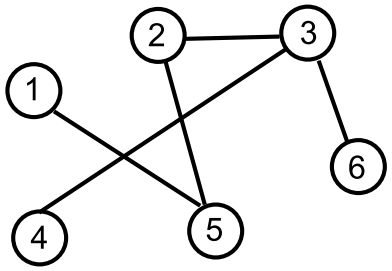

FIGURE 2. A graph with 6 vertices and 5 edges

Definition 1.2. The *degree* of a vertex v, denoted $\Delta(v)$, is the number of edges incident to it. The maximum degree of a graph G, denoted $\Delta(G)$, is the greatest $\Delta(v)$ over all $v \in V(G)$.

For example, in the graph G in Figure 2, vertex 3 has degree 3. Since it is the vertex with the greatest number of edges incident to it, $\Delta(G) = 3$.

Graphs model pairwise relations between objects. For example, Figure 3 shows a graph modeling the seven bridges of Königsburg, where vertices represent land masses and edges represent bridges. This graph makes it easy to argue that the desired traversal is impossible. Notice each vertex has odd degree, either three or five. If we start on a vertex with degree three, we must leave through one edge, come back through another, and leave again through the remaining edge, making it impossible to return while crossing each edge exactly once. Likewise if we start on the vertex with degree five, we leave, come back, leave, come back, and leave again. Thus, no matter where we start, we will never be able to return to our starting position if we are required to cross every bridge exactly once.

Definition 1.3. A *complete graph* is a graph whose vertices are all pairwise adjacent. The complete graph on n vertices is denoted K_n.

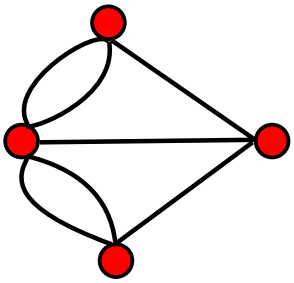

FIGURE 3. The Seven Bridges of Königsburg as a graph

Definition 1.4. A *path* is a graph whose vertices can be ordered so that two vertices are adjacent if and only if they are consecutive in the list. An *n-path*, denoted P_n, is a path on n vertices. See Figure 4.

A path has exactly two vertices of degree 1 (these are the first and last vertices in the ordered list); the rest are of degree 2.

Definition 1.5. A *cycle* is a graph with an equal number of vertices and edges whose vertices can be placed around a circle so that two vertices are adjacent if and only if they appear consecutively along the circle. An *n-cycle*, denoted C_n, is a cycle on n vertices.

Deleting one edge from a cycle produces a path as illustrated in Figure 4. To generalize this concept, we introduce the notion of a subgraph.

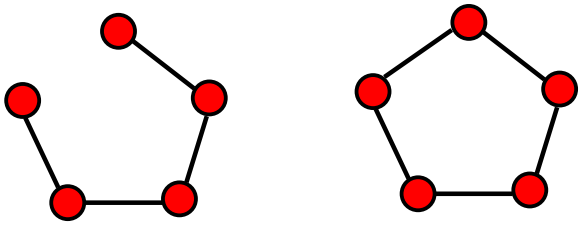

FIGURE 4. A 5-path P_5 (left) and a 5-cycle C_5 (right)

Definition 1.6. A *subgraph* of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We say that "G contains H."

As in the previous example (Figure 4), P_5 is a subgraph of C_5.

Definition 1.7. A *u,v-path* is a path whose vertices of degree 1 (its endpoints) are u and v. If a graph G contains a u,v-path, then the *distance* between u and v, denoted $d(u,v)$, is the number of edges in the shortest u,v-path.

For example, in Figure 2, the distance between vertices 1 and 6 is 4.

Definition 1.8. A graph is *connected* if it contains a u,v-path whenever $u,v \in V(G)$.

In other words, a graph is connected if we can get from any point on the graph to any other point on the graph by following edges. All the graphs illustrated so far are connected. However, if we delete any edge from the graph in Figure 2, the graph will no longer be connected. For our purposes, we will mainly be concerned with connected graphs.

Definition 1.9. A *tree* is a connected graph containing no cycles. A *leaf* is a vertex of degree 1.

An example of a tree is given in Figure 5. Its leaves are v_1, v_2, v_3, v_4, v_6, v_7, v_8, and v_9. Notice the path between any two vertices is unique. This is true for trees in general.

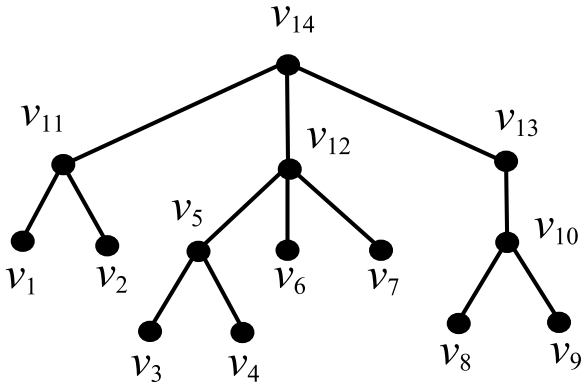

FIGURE 5. A tree

Definition 1.10. A *rooted tree* is a tree with one vertex r chosen as the root. For each vertex v, let $P(v)$ be the unique v,r-path. The *parent* of v is its neighbor on $P(v)$; its children are its other neighbors.

In Figure 5, for example, if we choose v_{14} as the root, v_5 has v_{12} as a parent and v_3 and v_4 as children.

Definition 1.11. A *spanning subgraph* of a graph G is a subgraph with vertex set $V(G)$. A *spanning tree* is a spanning subgraph that is a tree.

We can get a spanning tree from any connected graph by repeatedly removing an edge from a cycle.

Next, we consider another problem that can be solved using graph theory.

1.2. **Coloring and Brooks' Theorem.** Suppose we want to assign time slots for final exams so that two courses with a common student have different slots. To solve this and similar problems, we use the notion of graph coloring.

Definition 1.12. A *k-coloring* of a graph G is a labeling $f : V(G) \to S$, where $|S| = k$. The labels are called *colors* and the vertices of one color form a *color class*. A k-coloring is *proper* if no two adjacent vertices are in the same color class. A graph is *k-colorable* if it has a proper k-coloring. (See Figure 6.)

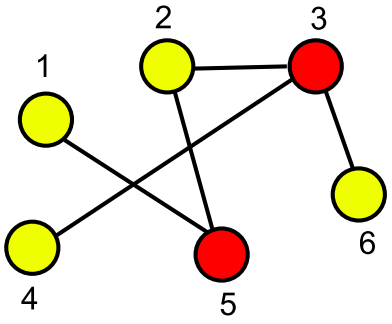

FIGURE 6. The graph in Figure 2 is 2-colorable.

Typically when we speak of graph coloring, we mean proper coloring.

Definition 1.13. The *chromatic number* of a graph G, written $\chi(G)$, is the least k such that G is k-colorable.

For example, the chromatic number of the graph in Figure 6 is 2 since it has a proper 2-coloring and it cannot be properly colored with only 1 color.

So, to solve our scheduling problem, we can create a graph where vertices represent courses and edges represent a common student in both courses. The least number of time slots needed, then, is the chromatic number of the graph. In general, finding the chromatic number of a graph can be used to solve a wide variety of optimization problems. As a result, it becomes useful to find upper bounds on the chromatic number based on the structure of the graph. This leads to Brooks' Theorem.

Brooks' Theorem. [We, 1993] *If G is a connected graph other than a complete graph or an odd cycle, then $\chi(G) \leq \Delta(G)$.*

Brooks' Theorem gives an upper bound for the chromatic number of graphs in terms of its maximum degree. As we will see in the next section, a natural extension of graph coloring uses this same idea.

2. BASIC IDEAS BEHIND GRAPH LABELING

Suppose we have a distribution of radio station transmitters scattered across town. Now suppose we wanted to assign channel frequencies to these transmitters. However, we have to be careful. Transmitters that are close together cannot be assigned the same channel frequency or else they would interfere with each other. For the same reason, transmitters that are very close together must be assigned channel frequencies at least two units apart. One way to approach this problem is reduce our situation to a graph. We can have vertices represent radio station transmitters and edges connect vertices that are "very close." The assignment of channel frequencies will be represented through what we call $L(2,1)$-labeling.

Definition 2.1. An $L(2,1)$-*labeling* (sometimes simply referred to as a *labeling*) of a graph G is a nonnegative integer-valued function $f : V(G) \to \{0,1,2,\ldots\}$ such that, whenever x and y are two adjacent vertices in $V(G)$, then $|f(x) - f(y)| \geq 2$, and, whenever the distance between x and y is 2, then $|f(x) - f(y)| \geq 1$.

So, if two vertices are a distance two apart, they cannot be assigned the same label. If two vertices are adjacent, their labels must differ by at least two. An example of this is given in Figure 7.

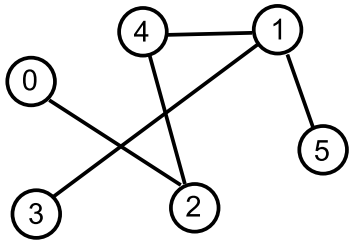

FIGURE 7. An $L(2,1)$-labeling

Definition 2.2. The $L(2,1)$-*labeling number* of G, denoted $\lambda(G)$, is the smallest number m such that G has an $L(2,1)$-labeling with no label greater than m.

The $L(2,1)$-labeling number of a graph labeling is analogous to the chromatic number of graph coloring. In the next section, we give examples of the labeling numbers of special classes of graphs.

3. Labeling Numbers of Special Classes of Graphs

3.1. **Complete Graphs.** Consider the complete graph on n vertices, K_n.

Proposition 3.1. $\lambda(K_n) = 2n-2$.

Proof. Given K_n with vertices $v_1, v_2, \ldots, v_n$, the function $f : V(G) \to \{0,1,2,\ldots,2n-2\}$ defined by $f(v_i) = 2i-2$ is a labeling of K_n. So, $\lambda(K_n) \leq 2n-2$. We claim we can't label K_n with just the numbers 0, 1, 2, $\ldots$, $2n-3$. Note that we have $2n-2$ labels that need to be assigned to n vertices. We can think of this as $n-1$ disjoint pairs of consecutive labels in which n vertices must be placed. By the pigeon hole principle, one of these pairs of consecutive labels must contain two vertices. However, since these two vertices are adjacent in K_n, this violates our labeling condition. Thus, $\lambda(K_n) = 2n-2$. □

Figure 8 illustrates a minimal labeling of K_5.

3.2. **Paths.** First, consider P_2. We start by labeling one vertex 0. This forces the other vertex to be at least 2. So $\lambda(P_2) = 2$, as in Figure 9.

Proposition 3.2. $\lambda(P_3) = 3$.

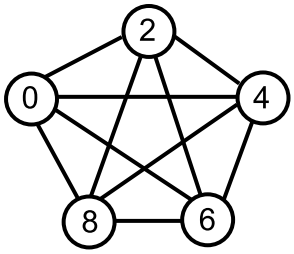

FIGURE 8. An $L(2,1)$-labeling of K_5

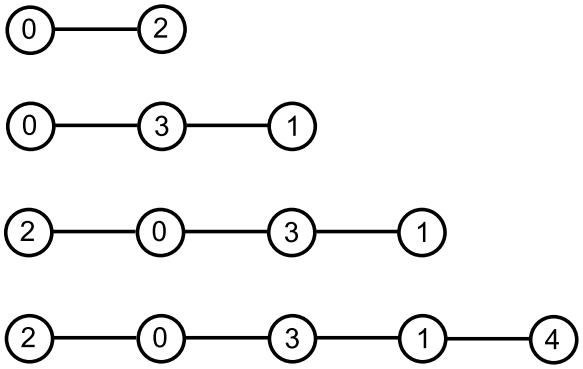

FIGURE 9. $L(2,1)$-labelings of P_2, P_3, P_4, and P_5 (from top to bottom)

Proof. For P_3, we can label the leftmost vertex 0, the middle vertex 3, and the rightmost vertex 1, as shown in Figure 9. So, $\lambda(P_3) \leq 3$. We claim we can't label P_3 with just the numbers 0, 1, and 2. The label 1 could not be used anywhere or else it would have to be adjacent to 0, 1, or 2, all of which violates the adjacency rule. This leaves us with two labels (0 and 2) that must be assigned to three vertices. By the pigeon hole principle, two of these vertices must receive the same label, which necessarily violates the condition. □

Before we consider the P_4, we need the following lemma.

Lemma 3.3. *If H is a subgraph of G, then $\lambda(H) \leq \lambda(G)$.*

Proof. Let $\lambda(G) = m$ with corresponding labeling $f : V(G) \to \{0,1,\ldots,m\}$. Then $g : V(H) \to \{0,1,\ldots,m\}$, defined by $g(v) = f(v)$ for all $v \in V(H)$, is a labeling of H that uses no label greater than m. Thus, $\lambda(H) \leq m = \lambda(G)$. The idea is we can use the same labels we use on G to label the corresponding vertices of H. □

Proposition 3.4. $\lambda(P_4) = 3$.

Proof. Since P_3 is a subgraph of P_4, from our previous result we know $\lambda(P_4) \geq \lambda(P_3) = 3$. Figure 9 shows we can label P_4 with no label greater than 3. Thus $\lambda(P_4) \leq 3$ and the result follows. □

Proposition 3.5. $\lambda(P_n) = 4$ *for* $n \geq 5$.

Proof. First, we show $\lambda(P_5) = 4$. Figure 9 shows we can label P_5 with no label greater than 4. So, $\lambda(P_5) \leq 4$. We claim we can't label P_5 with just the numbers 0, 1, 2, and 3. The labels 1 and 2 cannot be assigned to any non-endpoint vertex without violating either the adjacency rule or the distance two rule. To see this, suppose one of the non-endpoint vertices of P_5 were labeled 1. Then only the label 3 can be assigned to its neighbors without violating the adjacency rule. However, if both its neighbors receive the label 3, the distance two rule is violated. So, this leaves us with two labels (0 and 3) that must be assigned to the three non-endpoint vertices. Again, by the pigeon hole principle, two of these vertices must receive the same label, which necessarily violates the condition. So, $\lambda(P_5) = 4$.

Next, we show $\lambda(P_n) = 4$ for $n > 5$. Let P_n be a path with more than 5 vertices. Since P_5 is a subgraph of P_n, we know $\lambda(P_n) \geq \lambda(P_5) = 4$. Notice we can cyclically repeat the labels in P_5 (2, 0, 3, 1, 4, 2, 0, 3, ...) and still get a proper labeling for any P_n. Thus $\lambda(P_n) \leq 4$ and the result follows. □

3.3. **Cycles.** Now let's consider the labeling numbers of cycles. Recall that we can get a cycle by adjoining the endpoints of a path.

Proposition 3.6. $\lambda(C_n) = 4$ *for* $n \geq 3$.

Proof. Since $C_3 = K_3$, from Proposition 3.1, we have $\lambda(C_3) = 2(3) - 2 = 4$. Now consider C_4. Figure 10 shows we can label C_4 with no label greater than 4. So, $\lambda(C_4) \leq 4$. We claim we can't label C_4 with just the numbers 0, 1, 2, and 3. Since every vertex is C_4 is adjacent to two other vertices, we cannot use labels 1 and 2, as explained earlier, without violating the rules. This leaves us with two labels (0 and 3) that must be assigned to the four vertices. Again, by the pigeon hole principle, two of these vertices must receive the same label, which necessarily violates the condition since any pair of vertices in C_4 are at most distance two apart. So, $\lambda(C_4) = 4$.

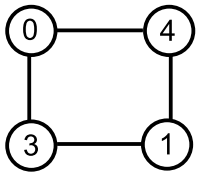

FIGURE 10. An $L(2,1)$-labeling of C_4

Now consider C_n, where $n \geq 5$. Since C_n contains P_5 as a subgraph, $\lambda(C_n) \geq \lambda(P_5) = 4$. Now we want to show $\lambda(C_n) \leq 4$ by defining a labeling on C_n using no label greater than 4. We have three cases. First, suppose $n \equiv 0 \pmod 3$. Then we can label our vertices (starting at one vertex and proceeding clockwise) 0, 2, 4, 0, 2, 4,.... Next, suppose $n \equiv 1 \pmod 3$. Then we can label our vertices 0, 2, 4, 0, 2, 4,..., 0, 2, 4, 0, 3, 1, 4. If $n \equiv 2 \pmod 3$, then we can label our vertices 0, 2, 4, 0, 2, 4,..., 0, 2, 4, 1, 3. This is illustrated in Figure 11. In each case, we repeat the labeling 0, 2, 4 as many times as necessary. This completes the proof.

□

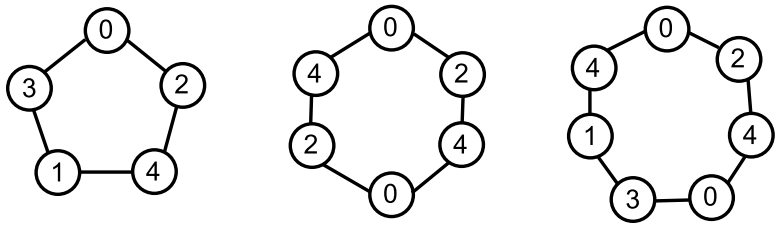

FIGURE 11. $L(2,1)$-labelings of C_5, C_6, and C_7 (from left to right)

4. GREEDY LABELING

We now seek ways to $L(2,1)$-label graphs in general. We start with a greedy labeling algorithm.

Algorithm 4.1 (Greedy labeling)**.** For a given graph G with vertices $v_1,...,v_n$, label vertices in the order $v_1,...,v_n$ by assigning the smallest nonnegative integer s such that $|s-r| \geq 2$ for any r assigned to a lower indexed neighbor, and $|s-t| \geq 1$ for any t assigned to a lower indexed vertex at a distance 2.

So, given a graph with ordered vertices, we go through the vertices in order assigning the smallest number that doesn't violate any of our labeling conditions based on the previously assigned labels. Figure 12 gives an example of a graph with ordered vertices and its greedy labeling.

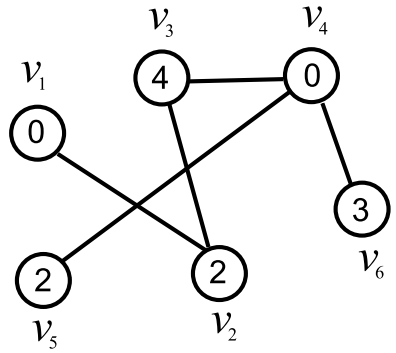

FIGURE 12. A graph with ordered vertices and its greedy labeling

Now we are able to prove an easy bound.

Theorem 4.2. *Let G be a graph with maximum degree Δ. Then $\lambda(G) \leq \Delta^2 + 2\Delta$.*

Proof. Arbitrarily order the vertices of G and perform the greedy labeling algorithm. A vertex $v \in V(G)$ has at most Δ neighbors. Each of these neighbors can rule out at most 3 labels for v (e.g. if v is neighbors with a vertex labeled 2, it cannot be labeled 1, 2, or 3). For each neighbor of v, there are at most $\Delta - 1$ vertices a distance two from v (since we don't consider v at distance two from itself). So there are a total of at most $\Delta(\Delta-1) = \Delta^2 - \Delta$ vertices a distance two away from v.

Each of these vertices can rule out at most 1 label for v. Thus when it comes time to label v, there are at most $3\Delta + \Delta^2 - \Delta = \Delta^2 + 2\Delta$ numbers we must avoid. So, v can be labeled with some number in $\{0, 1, 2, \ldots, \Delta^2 + 2\Delta\}$. Therefore, we have that $\lambda(G) \leq \Delta^2 + 2\Delta$. □

5. The Chang-Kuo Algorithm

Though the previous result is nice, we can do better. The result and proof presented in this section is due to Chang and Kuo [Cha, 1996]. Their method will be the basis for proving the best known bound. We start with a definition and an algorithm.

Definition 5.1. For any fixed positive integer k, a *k-stable set* of a graph G is a subset S of $V(G)$ such that every two distinct vertices in S are of distance greater than k.

For example, in Figure 12, $\{v_1, v_4\}$ form a 2-stable set since v_1 and v_4 are more than a distance 2 apart. Similarly, $\{v_2, v_5\}$ also form a 2-stable set. Notice that every vertex in a 2-stable set can be assigned the same label without violating any of our conditions. The next algorithm uses this idea to give a labeling scheme.

Algorithm 5.2. For any graph G, start with all vertices unlabeled. Let $S_{-1} = \emptyset$. When S_{i-1} is determined and not all vertices in G are labeled, let

$$F_i = \{x \in V(G) \mid x \text{ is unlabeled and } d(x, y) \geq 2 \text{ for all } y \in S_{i-1}\}.$$

Choose a maximal 2-stable subset S_i of F_i. Label all vertices in S_i by i. Increase i by one and continue the above process until all vertices are labeled.

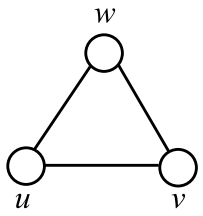

FIGURE 13. K_3

Example 5.3. To see how this works, let's apply Algorithm 5.2 to K_3 (shown in Figure 13). First, we have all vertices unlabeled and

$$S_{-1} = \emptyset.$$

Now we determine F_0. Since all vertices, at this point, are unlabeled and it is vacuously true that all vertices are at least a distance 2 from all vertices in the empty set, we have that

$$F_0 = \{u, v, w\}.$$

Next, we determine S_0 by choosing a maximal 2-stable subset of $F_0 = \{u, v, w\}$. Let's have

$$S_0 = \{u\}$$

and label u with 0. Now we determine F_1. Since no vertex is at least a distance 2 from u, we have

$$F_1 = \emptyset.$$

So

$$S_1 = \emptyset.$$

Now we determine F_2. Since both of our remaining unlabeled vertices are at least a distance 2 from all vertices in the empty set, we have that

$$F_2 = \{v, w\}.$$

Next, we determine S_2 by choosing a maximal 2-stable subset of $F_2 = \{v, w\}$. Let's have

$$S_2 = \{v\}$$

and label v with 2. Now we determine F_3. Since no vertex is at least a distance 2 from v, we have

$$F_3 = \emptyset.$$

So

$$S_3 = \emptyset.$$

Now we determine F_4. Since our remaining unlabeled vertex is at least a distance 2 from all vertices in the empty set, we have that

$$F_4 = \{w\},$$

which means

$$S_4 = \{w\}.$$

And we label w with 4. The finished labeling is shown in Figure 14.

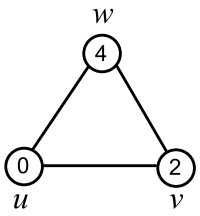

FIGURE 14. K_3 after applying Algorithm 5.2

Notice the difference between the greedy labeling algorithm and the Chang-Kuo algorithm. The greedy labeling algorithm goes through each vertex and assigns it the smallest possible label, whereas the Chang-Kuo algorithm goes through each label and assigns it to a maximal set of possible vertices. As we will see, this difference will allow us to establish a better bound.

Theorem 5.4. *Let G be a graph with maximum degree Δ. Then $\lambda(G) \leq \Delta^2 + \Delta$.*

Proof. Let G be a graph with maximum degree Δ. Perform Algorithm 5.2 on G. Let k be the maximum label used and let x be a vertex whose label is k. Let

$I_1 = \{i \mid 0 \leq i \leq k-1 \text{ and } d(x,y) = 1 \text{ for some } y \in S_i\}$,
$I_2 = \{i \mid 0 \leq i \leq k-1 \text{ and } d(x,y) \leq 2 \text{ for some } y \in S_i\}$,
$I_3 = \{i \mid 0 \leq i \leq k-1 \text{ and } d(x,y) \geq 3 \text{ for all } y \in S_i\}$.

Then we have

$$|I_2| + |I_3| = k. \tag{1}$$

Since the total number of vertices y with $1 \leq d(x,y) \leq 2$ is at most $\Delta + \Delta(\Delta-1) = \Delta^2$, we have

$$|I_2| \leq \Delta^2. \tag{2}$$

Also, there are at most Δ vertices adjacent to x so

$$|I_1| \leq \Delta. \tag{3}$$

Now for any $i \in I_3$, $x \notin F_i$. Otherwise, $S_i \cup \{x\}$ is a 2-stable subset of F_i, which contradicts the choice of a maximal S_i. This means $d(x,y) = 1$ for some vertex $y \in S_{i-1}$. So, $i-1 \in I_1$. Thus,

$$|I_3| \leq |I_1|. \tag{4}$$

Therefore, combining (1), (2), (3), and (4) gives

$$\lambda(G) \leq k = |I_2| + |I_3| \leq |I_2| + |I_1| \leq \Delta^2 + \Delta.$$

□

6. GONÇALVES BOUND

As we will see, the Chang-Kuo bound can be improved. The next result is due to Gonçalves [Go, 2004] and is currently the best known bound for graphs in general.

Theorem 6.1. [Go, 2004] *For any graph G with maximum degree $\Delta \geq 3$, we have $\lambda(G) \leq \Delta^2 + \Delta - 2$.*

The proof for this theorem uses the Chang-Kuo [Cha, 1996] algorithm. Before we proceed, however, it will be useful to look at a slight modification of the algorithm.

Modified Chang-Kuo Algorithm. Let G be any graph on n vertices.

(1) Randomly order the vertices of G as $v_1, v_2, \ldots, v_n$.
(2) Starting with label $i = 0$, consider the vertices in the chosen order and label the vertex i if possible.
(3) After considering the last vertex v_n, increase i by 1 and repeat until all vertices are labeled.

Notice the difference between the original Chang-Kuo algorithm and the modified version. In the original, we were given the freedom to choose a maximal 2-stable set of vertices for each label. In the modified version, because we are assigning an order to the vertices, we are forced to pick a specific maximal 2-stable set for a given label. An example of a graph labeled using the modified Chang-Kuo algorithm is given in Figure 15.

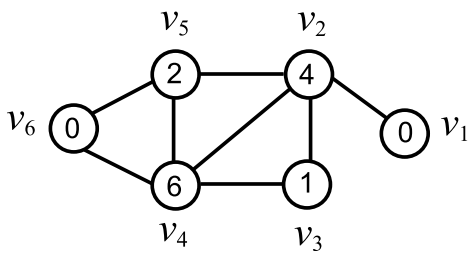

FIGURE 15. This graph was labeled using the modified Chang-Kuo algorithm.

Notice that this algorithm assigns each class of labels greedily. That is, for each label i, it considers the vertices in a given order and labels the vertex i if it can. After reaching the last vertex, it increases i by one and starts over. Let $l(v)$ be the label the algorithm assigns to the vertex v. Note that if $i > j$, a vertex v_j can only forbid an adjacent vertex, v_i, to be labeled $l(v_j)$ and $l(v_j)+1$. For example, in Figure 15, the vertex v_3 only forbids the labels 1 and 2 to vertex v_4 (v_3 would not forbid the label 0 to v_4 because when the algorithm considered whether or not to label v_4 with 0, the vertex v_3 was unlabeled). Similarly, if $i > j$, a vertex v_j can only forbid a vertex at distance 2 (or two-neighbor), v_i, to be labeled $l(v_j)$. For example, in Figure 15, the vertex v_3 only forbids the label 1 to v_5. Thus, if we let $F(v_j, v_i)$ be the set of all labels forbidden by v_j to v_i, where $i > j$, then we have that

$$F(v_j,v_i) = \begin{cases} \{l(v_j), l(v_j)+1\}, & \text{if } d(v_j,v_i) = 1 \text{ and } i > j \\ \{l(v_j)\}, & \text{if } d(v_j,v_i) = 2 \text{ and } i > j \\ \emptyset, & \text{if } d(v_j,v_i) > 2 \text{ and } i > j \end{cases}.$$

Suppose the algorithm considers the vertices in the order of $v_1, v_2, \ldots, v_n$. Note that if $i < j$, then v_j will not forbid the label $l(v_j)$ to v_i because when the algorithm considered whether or not to assign the label $l(v_j)$ to v_i, the vertex v_j was not yet labeled. In this case, we say that v_j is *posterior* to v_i. For example, in Figure 15, the vertex v_5 would not forbid the label 2 to v_2 because when the algorithm considered whether or not to label v_2 with 2, the vertex v_5 was unlabeled. Thus, if $i < j$, we have that

$$F(v_j,v_i) = \begin{cases} \{l(v_j)+1\}, & \text{if } d(v_j,v_u) = 1 \text{ and } i < j \\ \emptyset, & \text{if } d(v_j,v_u) \geq 2 \text{ and } i < j \end{cases}.$$

When $i = j$, we have $F(v_j, v_i) = \emptyset$.

If we let $F(v)$ be the set of all values forbidden to v, then we have that

$$F(v) = \bigcup_{u \in V(G)} F(u,v).$$

Since the algorithm labels v with the smallest value not in $F(v)$, we have that

$$\begin{aligned} l(v) &\leq |F(v)| \\ &\leq \sum_{u \in V(G)} |F(u,v)| \\ &\leq 2\Delta + \Delta(\Delta - 1) \\ &= \Delta^2 + \Delta. \end{aligned}$$

To improve the $\Delta^2 + \Delta$ bound we have to be careful on the order in which the algorithm considers the vertices. So, to improve our bound to $\Delta^2 + \Delta - 2$, we'd like to assign an order to our vertices so that

- all v_i, with $i \leq n-2$, have 2 posterior neighbors or two-neighbors,
- $l(v_{n-1}) \leq \Delta^2 + \Delta - 2$, and
- $l(v_n) \leq \Delta^2 + \Delta - 2$.

For the first requirement, consider a graph G with spanning tree T rooted in r. Order the vertices from the leaves to the root, so that any vertex v is indexed lower than its parent. This ordering of a graph based on a rooted tree T is called a T-ascending order. An example is given in Figure 16.

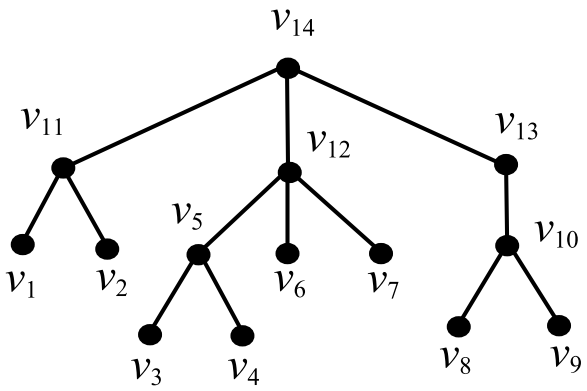

FIGURE 16. The vertices in this tree, rooted in v_{14}, are indexed in a T-ascending order.

We must have that r is ordered last and one of its children is ordered second to the last. This ordering will ensure that all v_i, with $i \leq n-2$, will have 2 posterior neighbors or two-neighbors. Indeed, the vertices with parents and grandparents will have 2 posterior neighbors or two-neighbors. Additionally, all the children of v_n (except for v_{n-1}) will be neighbors with v_n and two-neighbors with v_{n-1}.

Now we want to be able to construct our spanning tree T and a T-ascending order so that our last two requirements are met. To do this, we need the following structural lemma.

Lemma 6.2. [Go, 2004] *Every graph G with maximal degree $\Delta \geq 3$ has either:*

(1) *a vertex v with degree less than Δ,*
(2) *a cycle of length three,*

(3) *two cycles of length four passing through the same vertex,*
(4) *a vertex v with three neighbors u, x, y, such that there is a cycle of length four passing through the edge uv and such that the graph $G\backslash\{x,y\}$ is connected, or*
(5) *a vertex u with two adjacent vertices v and w such that the graph $G\backslash X$ is connected, where X is the set $(N(v)\cup N(u))\backslash\{w\}$.*[2]

This lemma will allow us to prove the Gonçalves bound by cases.

Case (1) Suppose a graph G on n vertices has a vertex v_n with degree less than Δ. Consider any spanning tree T of G rooted in v_n. Now consider any T-ascending order. Let's look at v_n. Since v_n has degree less than Δ, it has at most $\Delta-1$ neighbors and $(\Delta-1)^2$ two-neighbors. Thus, we have that

$$l(v_n) \leq |F(v_n)| \leq 2(\Delta-1)+(\Delta-1)^2 < \Delta^2+\Delta-2,$$

which takes care of the third requirement. Now let's look at v_{n-1}. It has at most Δ neighbors, including v_n which is posterior in the ordering, and at most $\Delta(\Delta-1)-1$ two-neighbors. Thus, we have that

$$l(v_{n-1}) \leq |F(v_{n-1})| \leq 2(\Delta-1)+1+\Delta(\Delta-1)-1 = \Delta^2+\Delta-2,$$

which takes care of the second requirement.

Case (2) Suppose a graph G on n vertices has a cycle of length three passing through the edge uv. Consider a spanning tree T rooted in v that uses the edge uv. Now consider a T-ascending order such that $u = v_{n-1}$ and $v = v_n$. Notice that a vertex in a cycle of length three has at most $\Delta(\Delta-1)-2$ two-neighbors (the two other vertices in the three cycle are not two-neighbors; see Figure 17). Thus we have that both $|F(v_n)|$ and $|F(v_{n-1})|$ are less than or equal to $\Delta^2+\Delta-2$, which fulfills the second and third requirement.

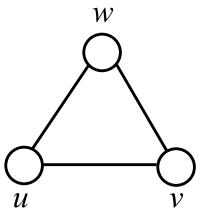

FIGURE 17. A three cycle. The vertex u has at most $\Delta(\Delta-1)$ two-neighbors. However, this formula counts both w and v, which are both 1-neighbors.

Case (3) Suppose a graph G on n vertices has two cycles of length four passing through the same vertex v. Let u be a neighbor of v in one of these cycles. Consider a spanning tree T rooted in v that uses the edge uv. Now order the vertices in T-ascending order with $u = v_{n-1}$ and $v = v_n$. Let's look at v_{n-1}. It has a posterior

[2]Here, $N(v)$ denotes the set of all neighbors of v.

neighbor in v_n and at most $\Delta(\Delta-1)-1$ two-neighbors (the vertex opposite v_{n-1} in the four cycle is counted twice; see Figure 18). So we have

$$l(v_{n-1}) \leq |F(v_{n-1})| \leq 2\Delta - 1 + \Delta(\Delta-1) - 1 = \Delta^2 + \Delta - 2.$$

Now let's look at v_n. Since it has two different vertices opposite it in two different four cycles, it has at most $\Delta(\Delta-1)-2$ two-neighbors (see Figure 18). So we have

$$l(v_n) \leq |F(v_n)| \leq 2\Delta + \Delta(\Delta-1) - 2 = \Delta^2 + \Delta - 2.$$

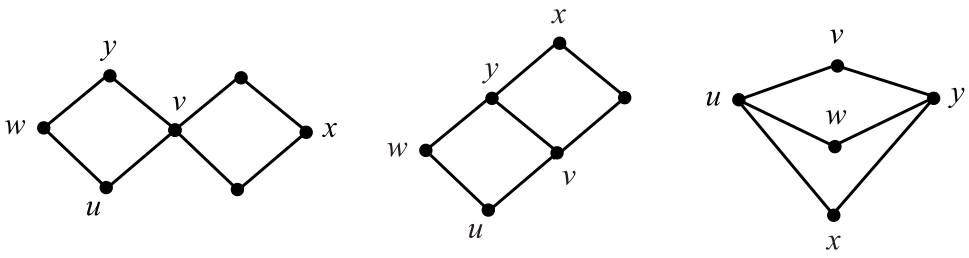

FIGURE 18. Three possible configurations of two cycles of length four passing through the same vertex v. The vertex u has at most $\Delta(\Delta-1)$ two-neighbors, however, this formula counts the vertex y twice. So, u really only has at most $\Delta(\Delta-1)-1$ two-neighbors. Likewise, the vertex v has at most $\Delta(\Delta-1)$ two-neighbors. However, this formula counts the vertices w and x twice. So, v really only has at most $\Delta(\Delta-1)-2$ two-neighbors.

Case (4) Suppose a graph G on n vertices has a vertex v with three neighbors u, x, y, such that there is a cycle of length four passing through the edge uv and such that the graph $G\backslash\{x,y\}$ is connected (see Figure 19). We construct a spanning tree T of G rooted in v from a spanning tree of $G\backslash\{x,y\}$ by adding the edges vx and vy. Since x and y are leaves in T, there is a T-ascending order such that $x = v_1$, $y = v_2$, $u = v_{n-1}$, and $v = v_n$.

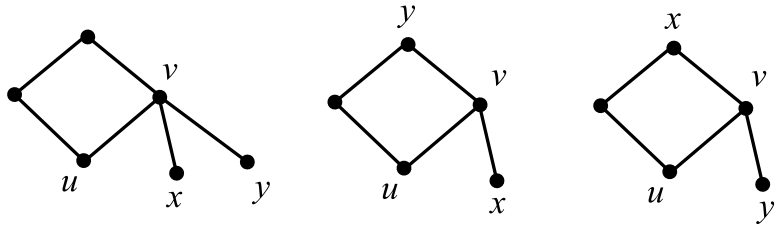

FIGURE 19. Three possible configurations for a vertex v with three neighbors u, x, y, such that there is a cycle of length four passing through the edge uv.

As in the previous case, the vertex v_{n-1} has a posterior neighbor v_n and at most $\Delta(\Delta-1)-1$ two-neighbors (since it is in a four cycle). Thus we have $l(v_{n-1}) \leq \Delta^2 + \Delta - 2$.

For v_n, it also has at most $\Delta(\Delta-1)-1$ two-neighbors (since it is in a four cycle), which reduces the bound by 1. To reduce the bound by 1 again, we show that there are two distinct vertices v_a and v_b such that $F(v_a,v_n)\cap F(v_b,v_n)\neq\emptyset$. In other words, the vertices v_a and v_b share a label forbidden to v_n, which means we would have double counted.

Since v_1 is the first vertex in the order, the algorithm automatically labels it 0. Since v_2 is at distance 2 from v_1, it cannot be labeled 0. Now we consider two cases for the label of v_2. See Figure 20. If $l(v_2)=1$, we have $1\in F(v_1,v_n)\cap F(v_2,v_n)$. If $l(v_2)\neq 1$, then it must have a neighbor v_x labeled 0. Since v_x is at distance two from v_n, we have $0\in F(v_1,v_n)\cap F(v_x,v_n)$.

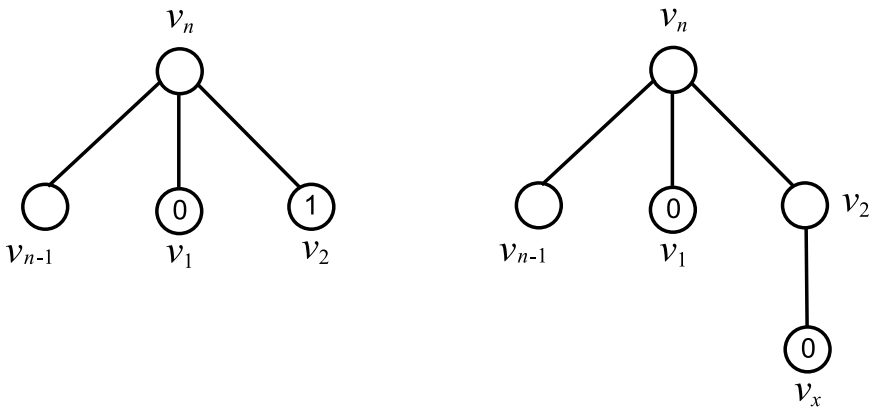

FIGURE 20. When $l(v_2)=1$ (left), both v_1 and v_2 forbid the label 1 to v_n. When $l(v_2)\neq 1$ (right), both v_1 and v_x forbid the label 0 to v_n.

Case (5) Suppose we are in none of the previous cases and that G has a vertex u with two adjacent vertices v and w such that the graph $G\backslash X$ is connected, where X is the set $(N(v)\cup N(u))\backslash\{w\}$. First, we construct a spanning tree of $G\backslash X$. Then, we add all the edges adjacent to u and all the edges adjacent to v to get a spanning tree T of G rooted in v. Since the neighbors of u and v, except u, v, and w are leaves in T, we can put them at the beginning of a T-ascending order so that

- w, u, and v are ordered v_{n-2}, v_{n-1}, and v_n, respectively
- $N(v_n)=\{v_{n-1},v_1,v_2,\ldots,v_{\Delta-1}\}$
- $N(v_{n-1})=\{v_n,v_{n-2},v_\Delta,v_{\Delta+1},\ldots,v_{2\Delta-3}\}$.

This is illustrated in Figure 21 when $\Delta=3$. Note that since we are assuming we are not in case 1, every vertex has degree Δ.

Again, since v_n is a posterior neighbor to v_{n-1}, we already reduce the bound on $l(v_{n-1})$ by 1. To reduce it by 1 more, like the previous case, we need to show that there are two distinct vertices v_a and v_b such that $F(v_a,v_{n-1})\cap F(v_b,v_{n-1})\neq\emptyset$. For v_n, to reduce the bound on $l(v_n)$ by 2, we need to show that there are two distinct pairs of vertices $v_a\neq v_b$ and $v_c\neq v_d$ such that $F(v_a,v_n)\cap F(v_b,v_n)\neq\emptyset$ and

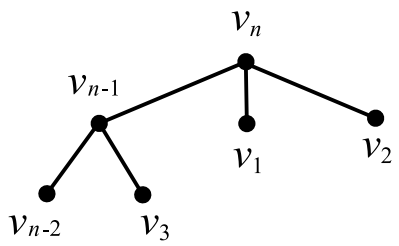

FIGURE 21. For $\Delta = 3$, for example, we have $N(v_n) = \{v_{n-1}, v_1, v_2\}$ and $N(v_{n-1}) = \{v_n, v_{n-2}, v_3\}$.

$F(v_c, v_n) \cap F(v_d, v_n) \neq \emptyset$. Now we consider two cases for the size of Δ.

Case (5) with $\Delta \geq 4$. Suppose $\Delta \geq 4$. First, we want to find two distinct vertices v_a and v_b such that $F(v_a, v_{n-1}) \cap F(v_b, v_{n-1}) \neq \emptyset$. Since $\Delta \geq 4$, v_{n-1} is adjacent to v_Δ and $v_{\Delta+1}$. So, we have that $d(v_\Delta, v_{\Delta+1}) \leq 2$, which means $l(v_\Delta) \neq l(v_{\Delta+1})$. Without loss of generality, we can assume $l(v_\Delta) < l(v_{\Delta+1})$. Now, if $l(v_{\Delta+1}) < l(v_\Delta) + 2$, then $l(v_{\Delta+1}) \in F(v_\Delta, v_{n-1}) \cap F(v_{\Delta+1}, v_{n-1})$. So, assume $l(v_{\Delta+1}) \geq l(v_\Delta) + 2$. Then there are two possible reasons why $v_{\Delta+1}$ was not labeled $l(v_\Delta) + 1$:

(1) $v_{\Delta+1}$ had a neighbor v_x labeled $l(v_\Delta)$ or $l(v_\Delta) + 1$, or
(2) $v_{\Delta+1}$ had a lower indexed 2-neighbor v_y labeled $l(v_\Delta) + 1$.

In the first case, v_x would be at distance two from v_{n-1} and we would have $l(v_x) \in F(v_x, v_{n-1}) \cap F(v_\Delta, v_{n-1})$. In the second case, if v_y is labeled $l(v_\Delta) + 1$ before this label is proposed to $v_{\Delta+1}$, then $y < \Delta$. So, v_y is a neighbor of v_n and a 2-neighbor of v_{n-1}. Thus, we would have $l(v_\Delta) + 1 \in F(v_y, v_{n-1}) \cap F(v_\Delta, v_{n-1})$.

Now we turn our attention to v_n. Recall that we want to find two distinct pairs of vertices $v_a \neq v_b$ and $v_c \neq v_d$ such that $F(v_a, v_n) \cap F(v_b, v_n) \neq \emptyset$ and $F(v_c, v_n) \cap F(v_d, v_n) \neq \emptyset$. Let's consider the labels the algorithm assigns to v_1, v_2, and v_3. Certainly, $l(v_1) = 0$ since v_1 is the first vertex considered. Now since v_1, v_2, and v_3 are all at distance two from each other, they have different labels. Without loss of generality, we can assume $l(v_1) < l(v_2) < l(v_3)$. We now consider different cases for $l(v_1)$, $l(v_2)$, and $l(v_3)$.

- If these vertices are labeled 0, 1, and 2 (see Figure 22), the label 1 and 2 are each forbidden twice to v_n. So, $1 \in F(v_1, v_n) \cap F(v_2, v_n)$ and $2 \in F(v_2, v_n) \cap F(v_3, v_n)$.
- If these vertices are labeled 0, 1, and $l(v_3) > 2$ (see Figure 23), then v_3 must have some neighbor v_x labeled 0 or 1. Since v_x is at distance two from v_n, we have $1 \in F(v_1, v_n) \cap F(v_2, v_n)$ and $l(x) \in F(v_1, v_n) \cap F(v_x, v_n)$.
- If these vertices are labeled 0, $l(v_2) > 1$, and $l(v_3) > 1$ (see Figure 24), then v_2 and v_3 must have respective neighbors v_y and v_z labeled 0. Since we are assuming we are not in case 4, v_y and v_z must be distinct. Since v_y and v_z are at distance two from v_n, we have $0 \in F(v_1, v_n) \cap F(v_y, v_n)$ and $0 \in F(v_1, v_n) \cap F(v_z, v_n)$.

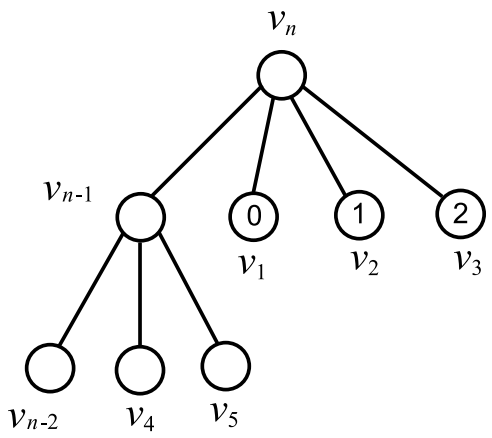

FIGURE 22. When $\Delta = 4$ and v_1, v_2, and v_3 are labeled 0, 1, and 2.

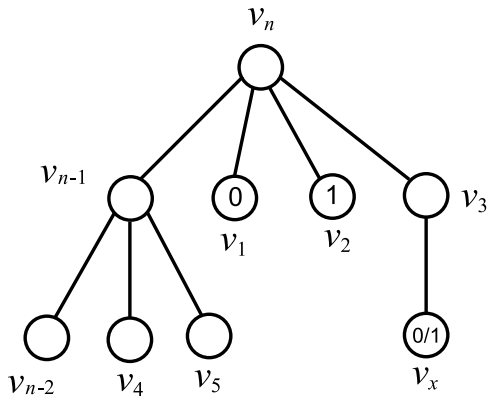

FIGURE 23. When $\Delta = 4$ and v_1, v_2, and v_3 are labeled 0, 1, and $l(v_3) > 2$.

Case (5) with $\Delta = 3$. Suppose $\Delta = 3$, as in Figure 21. Recall that we are assuming we are in none of the previous cases. Then $d(v_1, v_3)$ and $d(v_2, v_3)$ are greater than 1, otherwise we'd be in case 4. Notice that $d(v_1, v_3)$ and $d(v_2, v_3)$ are also less than or equal to 3 by how we constructed our spanning tree with T-ascending order (again, see Figure 21). So, $d(v_1, v_3)$ and $d(v_2, v_3)$ are either 2 or 3.

First, suppose $d(v_1, v_3) = 3$. Then v_1 and v_3 will both be labeled 0 according to the modified Chang-Kuo algorithm. Thus, we have

$$0 \in F(v_1, v_n) \cap F(v_3, v_n) \text{ and } 0 \in F(v_1, v_{n-1}) \cap F(v_3, v_{n-1}).$$

It remains to find another pair of vertices $v_a \neq v_b$ such that $F(v_a, v_n) \cap F(v_b, v_n) \neq \emptyset$. Now if $l(v_2) = 1$, then $1 \in F(v_1, v_n) \cap F(v_2, v_n)$. So, assume $l(v_2) \neq 1$. Then v_2 must have a neighbor v_x labeled 0. Since we are not in case 4, v_x is distinct from v_3. Thus, $0 \in F(v_1, v_n) \cap F(v_x, v_n)$.

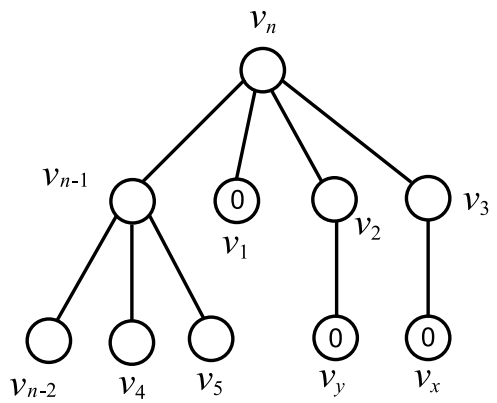

FIGURE 24. When $\Delta = 4$ and v_1, v_2, and v_3 are labeled 0, $l(v_2) > 1$, and $l(v_3) > 1$.

Next, suppose $d(v_1, v_3) = 2$ and $d(v_2, v_3) = 3$. For this, we can permute the order of the vertices v_1 and v_2. Doing so allows us to repeat the previous paragraph's argument verbatim.

Finally, suppose $d(v_1, v_3) = d(v_2, v_3) = 2$. Then there is a vertex v_x adjacent to v_1 and v_3 and a vertex v_y adjacent to v_2 and v_3. The vertices v_x and v_y are distinct, otherwise the vertex $v_x = v_y$ would be neighbors with v_1, v_2, and v_3, which would make the graph $G \backslash \{v_1, v_2, v_3\}$ disconnected. By construction of T, the edges v_1v_n, v_2v_n, and v_3v_{n-1} are the only edges in T incident to v_1, v_2, or v_3. So, v_x and v_y are not connected to T by its edges to v_1, v_2, or v_3. So, these vertices have just one adjacent edge in T and are leaves of T, which means we can order them as $v_x = v_4$ and $v_y = v_5$. We know that $d(v_1, v_5) > 1$ and $d(v_2, v_4) > 1$, otherwise $G \backslash \{v_1, v_2, v_3\}$ would be disconnected. Now we consider different cases according to $d(v_1, v_5)$ and $d(v_2, v_4)$.

- Suppose $d(v_1, v_5) > 2$ (see Figure 25). Then we have $l(v_1) = l(v_5) = 0$ and $l(v_2)$, $l(v_3)$, $l(v_4) \geq 2$. Thus, $0 \in F(v_1, v_{n-1}) \cap F(v_5, v_{n-1})$ and $0 \in F(v_1, v_n) \cap F(v_5, v_n)$. If $l(v_2) = 2$, we have to consider if $l(v_{n-1}) = 1$ or not. If it were, then we'd have $1 \in F(v_1, v_n) \cap F(v_{n-1}, v_n)$. If $l(v_2) = 2$ and $l(v_{n-1}) = 2$, then v_4 is labeled 3. Thus, $3 \in F(v_2, v_n) \cap F(v_4, v_n)$. If $l(v_2) = 2$ and $l(v_{n-1}) > 2$, then v_4 is labeled 2. Thus, $2 \in F(v_2, v_n) \cap F(v_4, v_n)$. If $l(v_2) > 2$, it is because its unique neighbor v_z, distinct from v_n and v_5 is labeled 1. So, we have $1 \in F(v_1, v_n) \cap F(v_z, v_n)$.
- Suppose $d(v_1, v_5) = 2$ and $d(v_2, v_4) > 2$. Then we can permute v_1 with v_2 and v_4 with v_5. Doing so allows us to repeat the previous argument.
- Suppose $d(v_1, v_5) = d(v_2, v_4) = 2$. Then we can permute v_2 and v_3 (see Figure 26). With this ordering, the algorithm labels the vertices v_1, v_2, v_3, and v_4 with labels 0, 1, 2, 3, respectively. Thus, $2 \in F(v_2, v_{n-1}) \cap F(v_3, v_{n-1})$, $1 \in F(v_1, v_n) \cap F(v_2, v_n)$, and $3 \in F(v_3, v_n) \cap F(v_4, v_n)$.

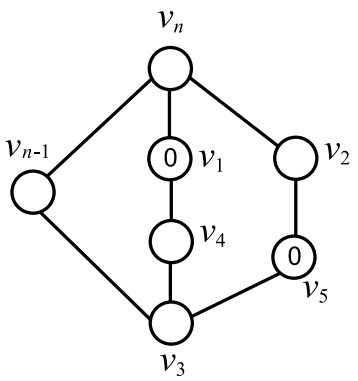

FIGURE 25. When $\Delta = 3$ and $d(v_1, v_5) > 2$.

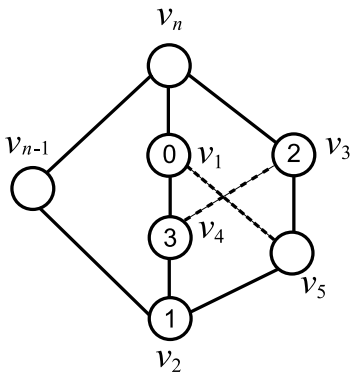

FIGURE 26. When $\Delta = 3$, $d(v_1, v_5) = d(v_2, v_4) = 2$. The figure above shows the graph after we permute v_2 and v_3. Dotted lines represent distance 2 relationships.

7. CONCLUSION

As mentioned earlier, the Gonçalves bound is the best general bound for graphs to date. Though there is still no proof for the Griggs-Yeh conjecture in general, the conjecture has been proven for several classes of graphs:

- graphs with $\Delta = 2$
- diameter 2 graphs [Gr, 1992]
- incidence graphs of a projective plane and polarity graphs of the Galois plane [Gr, 1992]
- outerplanar graphs [Jo, 1993]
- chordal graphs [Sa, 1994]
- Hamiltonian cubic graphs [Ka, 2004]
- planar graphs with $\Delta \neq 3$ [Be, 2005]
- direct and strong products of graphs [Kl, 2006].

REFERENCES

[Be, 2005] Bella, Peter, Daniel Král, Bojan Mohar, and Katarína Quittnerová. *Labeling planar graphs with a condition at distance two.* DMTCS proc. AE, (2005) 41-44

[Cha, 1996] Chang, Gerard J. and David Kuo. *The $L(2,1)$-Labeling Problem on Graphs.* SIAM J. Disc. Math., Vol. 9, (1996) 309-316.

[Go, 2004] Gonçalves, Daniel. *On the $L(d,1)$-labelling of graphs.* LaBRI, U.M.R. 5800, Université Bordeaux I (March 2004), 1-9.

[Gr, 1992] Griggs, Jerrold R. and Roger K. Yeh. *Labelling Graphs with a Condition at Distance 2.* SIAM J. Disc. Math., Vol. 5, No. 4, (November 1992) 586-595.

[Ka, 2004] Kang, J.H. *$L(2,1)$-labeling of 3-regular Hamiltonian graphs.* Ph.D. Thesis, University of Illinois, Urbana-Champaign, IL, 2004.

[Kl, 2006] Klavžar, S. and S. Špacapan *The Δ^2-conjecture for $L(2,1)$-labelings is true for direct and strong products of graphs.* IEEE Transactions on Circuits and Systems II, Vol. 53, No. 4, (April 2006) 274-277.

[Kr, 2003] Král, Daniel and Riste Škrekovski. *A Theorem about the Channel Assignment Problem.* SIAM J. Disc. Math., Vol. 16, No. 3, (2003) 426-437.

[Jo, 1993] Jonas, K. *Graph Coloring Analogues With a Condition at Distance Two: $L(2,1)$-Labelings and List λ-Labelings.* Ph.D. Thesis, Dept. of Math., Univ. of South Carolina, Columbia, SC, 1993.

[Ro, 1988] Roberts, F.S. (1988) private communication to J.R. Griggs.

[Sa, 1994] Sakai, D. *Labeling chordal graphs: distance two condition.* SIAM J. Disc. Math., Vol. 7, (1994) 133-140.

[We, 1993] West, Douglas B. *Introduction to Graph Theory.* 2nd Ed. Prentice Hall, (1993).

WHITMAN COLLEGE
E-mail address: lumaa@whitman.edu