PRIME LABELINGS OF INFINITE GRAPHS

MATTHEW KENIGSBERG AND OSCAR LEVIN

Abstract. A finite graph on n vertices has a prime labeling provided there is a way to label the vertices with the integers 1 through n such that every pair of adjacent vertices has relatively prime labels. In this paper, we extend the definition of prime labeling to infinite graphs and give a simple necessary and sufficient condition for an infinite graph to have a prime labeling. We then measure the complexity of prime labelings of infinite graphs using techniques from computability theory to verify that our condition is as simple as possible.

1. Introduction

A *graph labeling* is essentially an assignment of integers to the vertices (or sometimes edges or both) of a graph subject to certain conditions. In the last 50 or so years, a multitude of graph labelings have been described and studied. The dynamic survey [6] by Gallian describes over 50 types of graph labelings with results drawn from over 2000 papers. All but a handful of these consider only *finite* graphs. Here we consider one type of graph labeling and see how we can extend the definition to infinite graphs, with the hope that understanding this limit case might shed some light on open problems for finite graphs.

For a finite graph $G(V,E)$, a *prime labeling* is a bijection $f : V \to \{1, 2, \ldots, |V|\}$ such that for all $\{u, v\} \in E$, $f(u)$ and $f(v)$ are relatively prime ($\gcd(f(u), f(v)) = 1$). If a graph admits a prime labeling, we call the graph *prime*. This notion of graph labeling originates with Entringer, and was first described in a paper by Tout, Dabboucy, and Howalla [14]. Most of the results on prime labelings have been to show that large classes of graphs are in fact prime, but little is known in general. For example, Pikhurko proves in [10] that all trees with up to 50 vertices are prime. Recently (2011) Haxell, Pikhurko, and Taraz proved in [9] that all large trees are prime. However, the Entringer-Tout conjecture, that all trees are prime, remains open.

A similar story emerges for another class of graphs: ladders ($P_n \,\square\, P_2$ for some n). T. Varkey conjectured in an unpublished work that all ladders are prime. Work on this question has been done in [3], [12], and [13], and a recent preprint [8] claims to prove the conjecture.

In this present work, we ask which *infinite* graphs admit prime labelings. As far as we know, this is the first attempt at such an investigation, although we note that other types of labelings have successfully been extended to infinite graphs, such as in [5] for magic labelings or [4] for graceful labelings. The latter is particularly interesting in that it classifies precisely which infinite trees have graceful labelings,

---

Date: November 9, 2018.

2010 *Mathematics Subject Classification.* 05C78, 05C63, 05C85, 03D80.

Key words and phrases. graph labelings, infinite graphs, prime labelings, computability theory.

despite the long open conjecture that all (finite) trees are graceful. In Section 4, we will similarly prove that all infinite trees and all infinite ladders are prime.

We will start in Section 2 with some preliminary definitions and notation. Then in Section 3 we give an algorithm which produces a prime labeling of many infinite graphs that have prime labelings. This will lead us to a classification theorem for which infinite graphs are prime, which we state and prove in Section 4. We consider issues of complexity in Section 5. Finally, we conclude with some open questions in Section 6.

2. Preliminaries

Before we can study prime labelings of infinite graphs, we must decide what exactly we mean by this. First, by an infinite graph $G = (V, E)$ we will always mean a countably infinite graph (while there are uncountable graphs, it does not make sense to label these with integers). We could safely take $V = \mathbb{N} = \{0, 1, 2, \ldots\}$, but we will usually use $v_0, v_1, v_2, \ldots$ for the names of the vertices to avoid confusion with their labels. The edge set E will simply be a set of two-element subsets of V. Note this allows for finite or countably infinite numbers of edges, and does not prohibit vertices having infinite degree.

We will freely generalize standard notation for graphs to the infinite case: $K_{2,\infty}$, for example, will be the complete bipartite graph which has two vertices in one part and infinitely many in the other. The only time standard notation becomes ambiguous is with infinite paths: since P_n is a path with n edges, it makes sense to consider P_∞ as a path with infinitely many edges. However, there are two options here. The path could extend infinitely in both directions (a *two-way infinite path*) or just one (a *one-way infinite path*). We will use P_∞ to represent the one-way infinite path and not adopt a notation for the former.

It is then reasonable to extend the definition of prime labeling to infinite graphs as follows:

Definition 2.1. Given an infinite graph $G = (V, E)$, a *prime labeling* is a bijection $f : V \to \{1, 2, \ldots\}$ such that $\gcd(f(u), f(v)) = 1$ for all $\{u, v\} \in E$.

In what follows, it will sometimes be useful to exclude 1 from the codomain. Following Vaidya and Prajapati who introduced and studied k-prime labelings for finite graphs in [15], we define k-prime labelings of infinite graphs as follows:

Definition 2.2. Given an infinite graph $G = (V, E)$, a *k-prime labeling* is a bijection $f : V \to \{k, k+1, k+2, \ldots\}$ such that $\gcd(f(u), f(v)) = 1$ for all $\{u, v\} \in E$.

Note that a 1-prime labeling is the same as a prime labeling. Thus trivially, every prime graph is k-prime for some k, and every graph that is k-prime for all k will be prime. We will see shortly that there are infinite graphs that are prime but not 2-prime. However, it turns out that every infinite 2-prime graph is k-prime for *all* k. This can be seen by considering an algorithm for producing a k-prime labeling, as we now proceed to do.

3. An algorithm for prime labelings

We begin by describing a procedure which we think is a reasonable way to produce a k-prime labeling of an infinite graph. As usual, we take the vertex set to be $V = \{v_0, v_1, \ldots\}$.

We will proceed in stages, so that the every vertex is assigned some label at a finite stage, and in the limit, the labeling of the graph is k-prime. At the start of stage s, we will assume that we have labeled a finite subsets $V_s \subseteq V$ without mistakes (i.e., the greatest commond divisor of labels on any two adjacent vertices in V_s is 1), and proceed to find and label two vertices appropraitely.

Algorithm 3.1. *Proceed in stages.*

Stage $s = 0$: label v_0 with k and set $V_1 = \{v_0\}$.
Stage $s > 0$: Given labeled $V_s \subset V$:

(1) *Find the least natural number i such that v_i is not adjacent to any vertex in V_s, and label it with the least integer greater than k not yet used as a label.*
(2) *Find the least integer j such that v_j is unlabeled, and label it with a prime not yet used as a label, larger than any label of vertices adjacent to v_j.*
(3) *Let $V_{s+1} = V_s \cup \{v_i, v_j\}$ and proceed to the next stage.*

By design, this algorithm will always label adjacent vertices with numbers that are relatively prime. Since there are infinitely many prime numbers, it is always possible to complete step (2) of each stage. Thus, in order to show that this algorithm produces a k-prime labeling for a graph, it is only necessary to show that it is always possible to find a vertex v_i such that v_i is not adjacent to any vertex in V_s.

To illustrate the algorithm, we give some examples of infinite graphs that have prime labelings as well as some that do not.

Example 3.2. *The graph $P_\infty \square P_2$ with vertices arranged as in Figure 1 receives a prime labeling from Algorithm 3.1.*

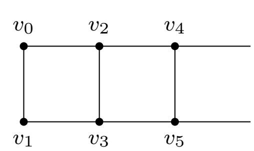

FIGURE 1. A (one-way) infinite ladder.

The result of the first eight stages of the algorithm is shown in Figure 2. Since the graph extends infinitely, it will always be possible to find a vertex not adjacent to any of the already labeled vertices. This means the algorithm will produce a prime labeling.

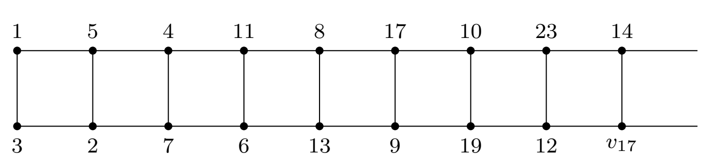

FIGURE 2. The result of the first eight stages of the algorithm.

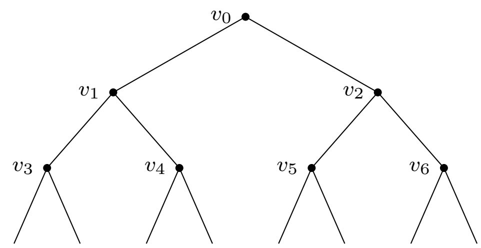

FIGURE 3. The top of a complete infinite binary tree.

Example 3.3. *An infinite complete binary tree with vertices arranged as in Figure 3 receives a prime labeling from Algorithm 3.1.*

Once again, it will always be possible to find a vertex not connected to the labeled part of the graph, so the algorithm produces a prime labeling. The result of the first four stages of the algorithm is shown in Figure 4.

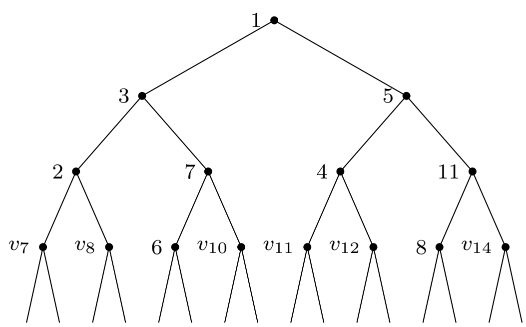

FIGURE 4. The labeling after four stages.

Example 3.4. *Algorithm 3.1 does not produce a prime labeling for an infinite star (the graph $K_{1,\infty}$).*

In order to produce a prime labeling, the algorithm must label the center of the star. After labeling the center of the star, step (1) of the next stage will attempt to find the least natural number i such that v_i is not adjacent to any vertex in the set of already labeled vertices, which includes the center of the star. Since the center of the star is adjacent to all other vertices, this is impossible, and the algorithm will not produce a prime labeling.

Note that if the infinite vertex was removed from the graph, the algorithm could easily produce a 2-prime labeling for the resulting graph. If the center of the star was then labeled with 1, the union of the two labelings would be a prime labeling for $K_{1,\infty}$.

Example 3.5. *Algorithm 3.1 does not produce a prime labeling for the infinite bipartite graph $K_{\infty,\infty}$.*

To see this, consider any graph $K_{\infty,\infty}$. Let a be the least natural number such that the vertex v_a is adjacent to v_0.

After a finite number of stages, v_a will be labeled. At the next stage, step (1) will look for the least natural number i such that v_i is not adjacent to any element of the set of labeled vertices $V_s \supset \{v_0, v_a\}$. Since every vertex is adjacent to either v_0 or v_a, this is not possible, and as such the algorithm will not be able to label the rest of the graph.

Unlike with the infinite star, there is no way to adjust the algorithm to produce a prime labeling of $K_{\infty,\infty}$.

Proposition 3.6. *$K_{\infty,\infty}$ has no prime labeling.*

Proof. Let $a \neq 1$ and $b \neq 1$ be any two labels of a pair of vertices in separate partite sets, and consider $n = ab$. Whatever vertex gets labeled with n (or indeed, any multiple of n) cannot be adjacent to either of the vertices labeled a or b. However, every vertex is adjacent to one of these vertices, a contradiction. Thus the graph has no prime labeling. □

4. Classification of Infinite Graphs

We have seen that not all graphs have prime labelings. The issue illustrated in Proposition 3.6 demonstrates a particular obstruction, which we summarize in the following lemma. Let $N(S)$ denote the set of vertices adjacent to one or more vertices in S (the *open neighborhood* of S) and $N[S] = N(S) \cup S$ (the *closed neighborhood* of S).

Lemma 4.1. *If an infinite graph $G = (V, E)$ has a finite set $S \subset V$, for which $N[S]$ contains all but finitely many vertices of G, then G does not have a k-prime labeling.*

Proof. Suppose G has a k-prime labeling, and consider such a finite set $S \subset V$. Let n be the product of the labels on the vertices of S. As such the infinitely many multiples of n must be assigned to vertices not in $N[S]$. Thus $N[S]$ cannot be co-finite, contrary to hypothesis. □

Note that if S is finite and $N[S]$ is co-finite, then there is a finite set S' for which $N[S'] = V$ (add to S all finitely many elements not in $N[S]$). Such a set S' is called a *dominating set*. Thus another way to describe the obstruction to a graph having a k-prime labeling is to say the graph has a finite dominating set. We will see that graphs that avoid this obstruction will always have a k-prime labeling at least for each $k \geq 2$. Thus we make the following definition.

Definition 4.1. An infinite graph $G = (V, E)$ is called *finitely dominated* provided there is some finite dominating set S, that is, a finite S such that $N[S] = V$.

Theorem 4.2. *An infinite graph G has a k-prime labeling for $k \geq 2$ if and only if G is not finitely dominated.*

Proof. The forward direction is Lemma 4.1.

Conversely, if G is not finitely dominated, then for any finite set S of vertices there is a vertex not adjacent to any element in S. This means that Algorithm 3.1 will produce a k-prime labeling: at each stage, V_s is finite, so it is always possible to find the least natural number i such that v_i is not adjacent to any vertex in the set V_s of already labeled vertices. □

We saw in Example 3.4 that the infinite star does not get a k-prime labeling from Algorithm 3.1, and by this theorem, we see that in fact it cannot have a k-prime labeling for any $k \geq 2$ (the center vertex is dominating). However, the infinite star *is* prime, since we can eliminate the "problem" by labeling the center vertex 1. This works in general and provides our main classification theorem.

We write $G - v$ for the graph resulting from removing the vertex v (and all incident edges).

Theorem 4.3. *An infinite graph G has a prime labeling if and only if there is a vertex v such that $G - v$ is not finitely dominated.*

Proof. Suppose first that G has a prime labeling f for which $f(v) = 1$. Then $G^- = G - v$ is 2-prime, witnessed by $f|_{G^-}$. By Theorem 4.2, G^- is not finitely dominated, as required.

Conversely, if $G - v$ is not finitely dominated, then $G - v$ has a 2-prime labeling by Theorem 4.2. The vertex that was removed can be labeled with 1, giving a prime labeling of G. □

Note, another way to state this result is that a graph will have a prime labeling if and only if there is possible to remove one vertex such that the remaining graph has a 2-prime labeling.

We can now state the relationship between k-prime graphs for different values of k.

Corollary 4.4. *If a graph has a k-prime labeling for any $k \geq 2$, it has a k-prime labeling for all k.*

Proof. According to Theorem 4.2, the condition for a graph to have a k-prime labeling is exactly the same for any $k \geq 2$. So if a graph satisfies that condition for any $k \geq 2$, it satisfies it for all $k \geq 2$. Further, if a graph is 2-prime, then it is not finitely dominated. But then $G - v_0$ will also not be finitely dominated, so by Theorem 4.3, G will have a prime labeling. □

As a result of our classification theorem, some natural classes of graphs will clearly have prime labelings.

Corollary 4.5. *All infinite trees are prime.*

We say a graph is *locally finite* if every vertex has finite degree.

Corollary 4.6. *All infinite locally finite graphs are prime. In particular, the infinite ladder is prime.*

The reason locally finite graphs allow our algorithm to work is that the neighborhood of any finite set must be finite. But even if this doesn't happen, we could always have enough vertices not adjacent to the finite set for other reasons. For example, the graph could have infinitely many connected components or one of the connected components could have infinite diameter.

Corollary 4.7. *All infinite graphs with infinitely many connected components or containing a connected component with infinite diameter have prime labelings.*

5. Computable Graphs

We turn now to the question of complexity of prime labelings for infinite graphs. In the finite case, we would consider computational complexity: you might ask whether deciding if a finite graph has a prime labeling is NP-complete. For infinite graphs, we use ideas from *computability theory.*

To do this, we must restrict our attention to *computable* graphs. Essentially, we identify graphs with their edge set, taking the vertex set to be $\mathbb{N}$, and require the edge set to be a computable set. This means that there is an algorithm that, given any two vertices (natural numbers) as input, returns whether the two vertices are adjacent. A more precise definition is beyond the scope of this paper, but the interested reader can see [11] for background on computability theory in general or [7] for a survey of the use of computability theory in combinatorics.

The first natural question to consider in this context is whether all computable graphs that have prime labelings have *computable* prime labelings (note that since we insist $V = \mathbb{N}$, a computable graph must necessarily be infinite). In other words, if the graph is nicely presented, will it always be possible to nicely describe a prime labeling? Somewhat surprisingly, the answer here is yes. (This is surprising given that many graph theoretic properties do not behave so nicely: there are computable graphs with 3-colorings with no computable 3-coloring [1] and computable graphs with Euler paths with no computable Euler path [2], for example.)

Proposition 5.1. *If G is a computable graph which admits a prime labeling, then G has a computable prime labeling.*

Proof. Let G be a computable graph with a prime labeling. By Theorem 4.3, we know that there is a vertex v such that $G - v$ is not finitely dominated. Label v with 1, then proceed with Algorithm 3.1. At step (1) of stage s, we are looking for a vertex not in $N[V_s]$. This can be found in finite time by asking whether v_i is adjacent to v_j for each $v_j \in V_s$, and if ever the answer is yes, we move on to the next potential v_i, which we know we must eventually find since V_s is not dominating. $\square$

The procedure outlined above relies on a certain amount of *non-uniformity*: we must know where to place the label 1. This does not prevent the prime labeling from being computable, since we are only asking for the existence of an algorithm for the prime labeling, not for a procedure to *find* that algorithm. But could we? Is it possible, given the algorithm for a particular graph, to produce the algorithm that gives the prime labeling? Here, we find the answer is negative.

Theorem 5.2. *There is no computable function which, given any computable graph admitting a prime labeling, produces the prime labeling for that graph.*

Before we give the proof, we need a little more background from computability theory. They key fact we will use is that there is an effective list $\varphi_0, \varphi_1, \varphi_2, \dots$ of all *partial* computable functions (again, see [11] for details). The intuition here is that we can consider every possible algorithm, perhaps written in JAVA, arranged alphabetically and by length (all algorithms have finite length). Of course, for any given algorithm, we have no reason to think that this algorithm will halt on all inputs, and this is why we are only considering *partial* computable functions (if it does halt on all inputs, we call it *total*). However, since the list contains every algorithm, partial or total, we know that if there were a computable function which gave the computable prime labeling of every computable graph (admitting a prime labeling), it must be somewhere on the list. Our goal then is to ensure every partial computable function on the list is wrong at least once.

Proof. We will build a sequence $G_0, G_1, \ldots$ of computable graphs, each admitting a prime labeling. While doing so, we will ensure that, for each $e \in \mathbb{N}$, the partial computable function φ_e is not a prime labeling of the graph G_e.

The construction will "dove-tail" the construction of the infinitely many graphs, so that by the end of stage s, we will have described the first s vertices of the first s graphs. The construction of each graph in the sequence will be independent of the others, so we need only describe how we build an arbitrary graph G_e.

In the limit, the graph G_e will be the union of two stars with centers v_0 and v_1, at least one of which is infinite. Notice that such a graph will have a prime labeling, as removing the center of an infinite star produces an infinite set of isolated vertices (we are appealing to Theorem 4.3 here). At each stage, we check whether φ_e has returned the label 1 for either v_0 or v_1. If this has not yet occurred, we add a new vertex adjacent to either v_0 or v_1, whichever we did not add to in the previous stage. If φ_e returns 1 for the label of v_i with $i \in \{0, 1\}$, then we only ever add new vertices adjacent to v_{1-i}.

Note that it is possible that φ_e will never return 1 for v_0 or v_1 (perhaps φ_e is not total, or it labels a different vertex with 1). In this case, G_e will consist of two infinite stars, but there is no way for φ_e to be a prime labeling (the product of the labels of the two centers has nowhere to go, as in Proposition 3.6). On the other hand, if φ_e does label one of the vertices v_0 or v_1 with a 1, then we never add any more neighbors to that vertex, and only the other vertex will be an infinite star. In this case, φ_e also cannot be a prime labeling. Whatever the label of the center of the infinite star is, there are only finitely many vertices (on the other star) that the infinitely many multiples of this label can be assigned to. This completes the proof.

□

The proof above relies on the inability of computable functions to predict whether a vertex of a graph will have infinite degree, and as such, the computable function does not know which vertex to label with 1. However, this is the only barrier to uniformity. If we consider instead 2-prime labelings, then we get uniformity.

The other computability question we should consider is the *decision problem*: given a computable graph, how hard is it to decide whether the graph has a prime labeling? The usual way to analyze this in computability theory is to determine where the decision problem lies inside (or above) the arithmetical hierarchy. One way to think of this task is that we are assessing the complexity of the condition which is equivalent to a graph having a prime labeling. We have a condition given in Theorem 4.3. Is this the simplest necessary and sufficient condition to a graph having a prime labeling?

Notice that by Theorem 4.2, a graph has a k-prime labeling for $k \geq 2$ if and only if the for all finite sets of vertices, there is at least one vertex not in the neighborhood of the set. Analyzing the quantifiers, we can state this condition as

$$\forall n \exists k (k > n \wedge k \notin N(\{0, 1, \ldots, n\})).$$

Since saying that a vertex is not in the neighborhood of a finite set of vertices is computable, we see that a graph having a 2-prime labeling is Π^0_2. Similarly, to say a graph has a prime labeling, we need it to be the case that there is a vertex, the removal of which, leaves a 2-prime graph. Thus a graph having a prime labeling is Σ^0_3.

Can we do better? For 2-prime labelings, the answer is no.

Theorem 5.3. *The decision problem for a graph having a k-prime labeling for $k \geq 2$ is Π^0_2-complete.*

Proof. Fix $k \geq 2$. We argued above that having a k-prime labeling is Π^0_2, so we need only show completeness. We will do this by giving a 1-reduction to the known Π^0_2-complete index set $\text{INF} = \{e : |W_e| = \infty\}$, where W_e is the domain of φ_e. That is, we build a sequence of computable graphs $\{G_i\}$ such that G_e has a k-prime labeling if and only if $e \in \text{INF}$.

We build the graphs simultaneously, as in the proof of Theorem 5.2, but this time each graph will either be the disjoint union of an infinite star with a finite path, or the disjoint union of an infinite star with a (one way) infinite path. In the former case, the graph will not be k-prime, in the latter it will k-prime, by Theorem 4.2.

The procedure for building the graph G_e is as follows. Initialize G_e with a center vertex for its star and an initial vertex for its path. At stage s of the construction we assume that we have built a finite star and a finite path. Run $\varphi_e(x)$ on all $x < s$ for which $\varphi_e(x)$ has not already halted at some earlier stage. We continue to run these computations until either $\varphi_e(x)$ halts for some input x, or until each computation has run for s steps, whichever comes first. If we see some $\varphi_e(x)$ halt, this will be the first time we realize that $x \in W_e$, so we have further evidence that $|W_e|$ might be infinite. Thus we add a vertex to the end of the finite path. On the other hand, if no (new) x appears in W_e (i.e., $\varphi_e(x)$ does not halt for any new x by stage s) we work off the assumption that $|W_e|$ is finite and add a vertex to the finite star in G_e.

To verify that this procedure gives us what we want, suppose first that $|W_e| = \infty$. Then there will be infinitely many stages at which we add a vertex to the end of the path, since at each stage we "discover" at most one new x in W_e. Thus in the limit, the path will be infinite (the star will likely be infinite as well, but regardless, G_e will have a k-prime labeling). Conversely, suppose $|W_e|$ is finite. Then there is a last stage at which any x appears in W_e, and so after that stage, we never add vertices to the path, making the path finite. $\square$

What about prime labelings? By the quantifier analysis above, we know that the decision problem cannot be harder than Σ^0_3. Further, a simple modification of the proof for 5.3 shows that the decision problem is at least Π^0_2-hard. We would expect the decision problem to in fact be Σ^0_3-complete, but a proof that it is Σ^0_3-hard goes beyond the scope of this paper. We leave this as an open question.

Question 1. Is the decision problem for a graph having a prime labeling Σ^0_3-complete?

6. Conclusion and Open Questions

We have considered a natural extension of the definition of prime labelings to infinite graphs. For 2-prime labelings, we have a simple necessary and sufficient condition and a condition only slightly less simple for prime labelings. By using tools from computability theory, we see that producing a 2-prime labeling of a

2-prime graph is as straight forward as possible, and only slightly less so for producing prime labelings of prime graphs. We also have that our criterion for 2-prime labelings is as simple as possible, and conjecture that the same is true for prime labelings.

These results mirror those for graceful labelings of infinite graphs, in that working with labelings of infinite graphs seems quite a bit easier than their finite counterparts. This suggests that the difficulty with working with finite graphs is very much tied to finiteness itself. The feeling of "running out of room" is exactly why labeling results are difficult.

We wonder however, whether a more restrictive definition of labelings for infinite graphs might serve as a better infinite analogue to the finite case. Note that for vertex coloring, it turns out that an infinite graph is k-colorable if and only if every finite subgraph is 4-colorable. Such a result for prime (and other) labelings would be very nice, but with our definition, is clearly false.

We do not know what the "right" definition would be, but we conclude by considering one possible variant of prime labeling that might be a step in the right direction and encourage others to pursue this further.

Definition 6.1. Let G be a graph, v_c be a vertex of that graph (c for *center*), and G_r be the subgraph of G that includes all vertices within distance r of v_c. Then G has a *limit-wise prime labeling* if it is possible to choose v_c and label the graph such that for infinitely many r, G_r has been given a prime labeling.

We call a graph *limit-wise prime* if it has a limit-wise prime labeling.

To get a feel for this, consider the complete infinite binary tree.

Example 6.1. *A complete infinite binary tree has a limit-wise prime labeling.*

Proof. For all $r \geq 3$, each row of the graph can have children labeled with the integers from 2^{r+1} to $2^{r+2} - 1$ as follows:

The lowest even number e has children $2e+1$ and $4e-1$. All other evens e have children $2e-1$ and $2e+1$. The lowest odd number o has children $2o-2$ and $2o+2$. The 2nd greatest odd number o has children $2o-4$ and $2o+4$. The greatest odd number o has children $2o-4$ and $2o-2$. All others odd numbers o have children $2o-4$ and $2o+2$.

The process is shown here for $r = 3$ in Figure 5.

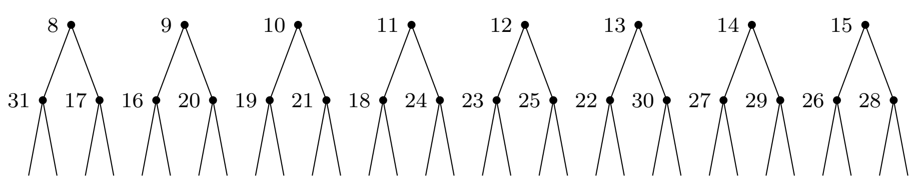

FIGURE 5. A limit-wise prime labeling of rows 3 and 4 of the complete binary tree.

It is straightforward but tedious to show that this will produce a limit-wise prime labeling for the tree after the first four rows are labeled with the numbers 1 to 15 in any manner that is prime. One possibility is shown in Figure 6

□

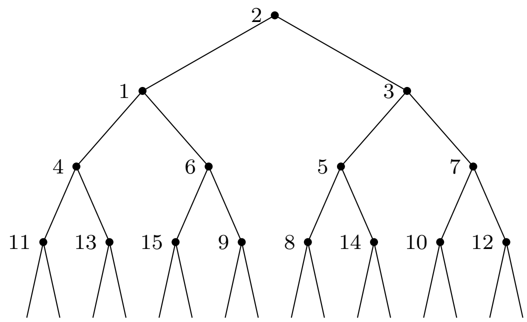

FIGURE 6. The start of a limit-wise prime labeled tree

It certainly appears that giving a limit-wise prime labeling is more difficult that giving a prime labeling. Indeed, there are prime graphs that are not limit-wise prime.

Example 6.2. *Let G be the square of the two-way infinite path, as in Figure 7. Then G has a prime labeling, but not a limit-wise prime labeling*

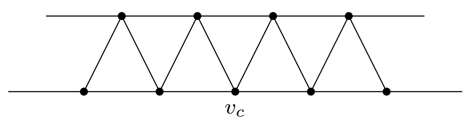

FIGURE 7. A prime graph that is not limit-wise prime

Proof. Since G is locally finite, it has a prime labeling.

To show that G has no limit-wise prime labeling, choose any vertex for v_c and let G_r be the subgraph that includes all vertices within distance r of v_c. G_r contains $4r + 1$ vertices. This means that if G_r has a prime labeling, then $2r$ even labels must be used.

Without loss of generality, let v_c be on the bottom of the graph as shown in Figure 7, and let b and t be the number of vertices with even labels on the bottom and top of the graph respectively. Since there are $2r + 1$ vertices on the bottom and adjacent vertices cannot have even labels, $b \leq r + 1$. Similarly, $t \leq r$. Since $2r$ total even labels must be used, $b + t = 2r$, so we have only two cases to consider: either $b = t = r$ or $b = r + 1$ and $t = r - 1$. We will argue that as soon as $r \geq 2$, both of these cases are impossible.

If $t = r$, then it must be that exactly every other vertex on top is even. Since each of these are adjacent to two different vertices on bottom, there is only one vertex on the bottom that can be even, so $b = 1 \neq r$. On the other hand, if $b = r + 1$, then every other vertex on bottom is even, leaving no vertices on top for even vertices, so $t = 0 \neq r$.

So for $r > 1$, G_r does not have a prime labeling, which means G does not have a limit-wise prime labeling, even though it does have a prime labeling.

$\square$

There are plenty of questions to consider about limit-wise prime labelings including whether this is even a useful variant of prime labeling of infinite graphs. Here are a few to get the ambitious reader started.

Question 2. Are all infinite trees limit-wise prime?

Question 3. What are reasonable necessary and/or sufficient conditions for a graph to be limit-wise prime?

Note that if every finite subgraph of an infinite graph is prime, then the graph is limit-wise prime. However, the converse is likely false. This could be investigated further.

There are also questions of complexity:

Question 4. Does every computable graph with a limit-wise prime labeling have a computable limit-wise prime labeling?

Question 5. How hard is it to decide whether a computable graph is limit-wise prime?

References

[1] Dwight R. Bean, *Effective coloration*, J. Symbolic Logic **41** (1976), no. 2, 469–480. MR0416889 (54 #4952)

[2] ———, *Recursive Euler and Hamilton paths*, Proc. Amer. Math. Soc. **55** (1976), no. 2, 385–394. MR0416888

[3] Adam H. Berliner, Nathaniel Dean, Jonelle Hook, Alison Marr, Aba Mbirika, and Cayla D. McBee, *Coprime and prime labelings of graphs*, J. Integer Seq. **19** (2016), no. 5, Article 16.5.8, 14. MR3514551

[4] Tsz Lung Chan, Wai Shun Cheung, and Tuen Wai Ng, *Graceful tree conjecture for infinite trees*, Electron. J. Combin. **16** (2009), no. 1, Research Paper 65, 15. MR2515742

[5] D. Combe and A. M. Nelson, *Magic labellings of infinite graphs over infinite groups*, Australas. J. Combin. **35** (2006), 193–210. MR2239315

[6] Joseph A. Gallian, *A dynamic survey of graph labeling*, Electron. J. Combin. **5** (1998), Dynamic Survey 6, 43 pp. MR1668059

[7] W. Gasarch, *A survey of recursive combinatorics*, Handbook of recursive mathematics, Vol. 2, 1998, pp. 1041–1176. MR1673598

[8] E. Ghorbani and S. Kamali, *Prime Labeling of Ladders*, ArXiv e-prints (October 2016), available at `1610.08849`.

[9] Penny Haxell, Oleg Pikhurko, and Anusch Taraz, *Primality of trees*, J. Comb. **2** (2011), no. 4, 481–500. MR2911187

[10] Oleg Pikhurko, *Trees are almost prime*, Discrete Math. **307** (2007), no. 11-12, 1455–1462. MR2311118

[11] Robert I. Soare, *Recursively enumerable sets and degrees*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1987. A study of computable functions and computably generated sets. MR882921 (88m:03003)

[12] M. Sundaram, R. Ponraj, and S. Somasundaram, *On a prime labeling conjecture*, Ars Combin. **79** (2006), 205–209. MR2218270

[13] ———, *A note on prime labeling of ladders*, Acta Cienc. Indica Math. **33** (2007), no. 2, 471–477. MR2392252

[14] R Tout, AN Dabboucy, and K Howalla, *Prime labeling of graphs*, NATIONAL ACADEMY SCIENCE LETTERS-INDIA **5** (1982), no. 11, 365–368.

[15] S Vaidya and U Prajapati, *Some results on prime and k-prime labeling*, Journal of Mathematics Research **3** (2011), no. 1, 66.

Vanderbilt University, Nashville, TN 37240, USA
Email address: matthew.kenigsberg@vanderbilt.edu

School of Mathematical Sciences, University of Northern Colorado, Greeley, CO 80639, USA
Email address: oscar.levin@unco.edu