DOMATIC PARTITIONS OF COMPUTABLE GRAPHS

MATTHEW JURA, OSCAR LEVIN, AND TYLER MARKKANEN

Abstract. Given a graph G, we say that a subset D of the vertex set V is a dominating set if it is near all the vertices, in that every vertex outside of D is adjacent to a vertex in D. A domatic k-partition of G is a partition of V into k dominating sets. In this paper, we will consider issues of computability related to domatic partitions of computable graphs. Our investigation will center on answering two types of questions for the case when $k = 3$. First, if domatic 3-partitions exist in a computable graph, how complicated can they be? Second, a decision problem: given a graph, how difficult is it to decide whether it has a domatic 3-partition? We will completely classify this decision problem for highly computable graphs, locally finite computable graphs, and computable graphs in general. Specifically, we show the decision problems for these kinds of graphs to be Π^0_1-, Π^0_2-, and Σ^1_1-complete, respectively.

1. Introduction

Computability theory has proved to be a valuable tool in analyzing the complexity of various results in graph theory (and indeed much of combinatorics — see [3] for a nice survey). A large portion of this work has been to sort out the effective nature of proper vertex colorings and the chromatic number of a graph. A vertex coloring is a partition of the vertices of the graph into *independent* sets — ones for which no two vertices in the set are adjacent.

The graph theoretic complementary concept to independent set is that of *dominating* set — one for which every vertex of G not in the set is adjacent to a vertex in the set. A partition of the vertices into dominating sets is called a *domatic partition*, and the maximum number of sets in such a partition is called the *domatic number* of the graph. In this paper we will investigate the complexity of domatic partitions and domatic numbers in much the way others have looked at the complexity of proper colorings (independent partitions) and chromatic numbers.

The k-domatic number problem, written k-DNP, is the question of whether a given graph G has a domatic k-partition, i.e., whether the domatic number of G is at least k. The k-domatic number problem for finite graphs is an example of a graph partitioning problem that is known to be NP-complete when $k \geq 3$, as shown in [7]. Compare this to other questions in effective graph theory: in a 1997 paper by Hirst and Lempp [6], the authors look at graph-theoretic problems that are NP-complete in the finite case to see if there is a pattern in the computability-theoretic complexity of infinite versions of the same problems. They conclude that

Date: August 2014.

2000 *Mathematics Subject Classification.* 03D45, 05C63, 05C69.

Key words and phrases. computability theory, computable graph theory, domatic number, dominating set, reverse mathematics.

This is a preprint. The final version is available at http://link.springer.com.

NP-complete problems can have infinite versions that behave vastly differently, depending on how the infinite version of the problem is stated.

They noted that there had been recent work by Harel [5] to show that deciding whether a given computable graph (i.e., a graph $G = (V, E)$ in which the vertex set V and edge relation E are computable) has a Hamilton path (NP-complete in the finite case) is Σ^1_1-complete in the infinite case. This is in contrast to the example of deciding whether a given computable graph has an Euler path. This problem is polynomial-time computable in the finite case, and has been shown to be Π^0_3-complete in the case of computable graphs, and Π^0_2-complete in the case of highly computable graphs (i.e., computable graphs in which every vertex has computable degree). Hirst and Lempp demonstrate that this parity does not always occur by showing that there is an NP-complete problem from finite complexity theory that does not become Σ^1_1-complete in some variations of the infinite case. They note that determining whether a finite graph is 3-chromatic is NP-complete, and then show that the set of indices of computable graphs that are 3-chromatic is Π^0_2-definable, the set of indices of finitely colorable graphs is Δ^0_3 definable, and in fact, that the set of indices of graphs with finitely colorable connected components is Π^0_3-complete. They also show, on the other hand, that the set of indices of computable graphs with colorings that use one color infinitely often is Σ^1_1-complete.

So it is not obvious how an infinite version of the k-domatic number problem will behave (at least for $k \geq 3$). Will it behave more like the Hamilton path problem and the coloring problem when one color is used infinitely often (Σ^1_1-complete), or will it behave more like one of the coloring problems that are arithmetical? We will resolve this question in section 4.

In this paper we will first look at the problem of deciding whether a computable graph has a domatic 2-partition (actually polynomial-time computable in the finite case), and then move on to the problem of deciding whether a computable graph has a domatic k-partition for $k \geq 3$. We focus on the case when $k = 3$. We also see what happens when we restrict ourselves to the class of highly computable graphs and locally finite computable graphs (i.e., computable graphs in which every vertex has finite degree).

In addition to asking how hard it is to detect a domatic k-partition in a given computable graph, we can also ask how complicated that domatic partition might be. This is analogous to the work done in [1] on the difference between chromatic number and computable chromatic number. (However, in that case, we know there are graphs with small chromatic number and unbounded computable chromatic number. Here, the computable domatic number could never be larger than the classical domatic number, since it is easier to have a smaller domatic partition than a larger one.)

2. Preliminaries

We assume that the reader is familiar with the basics of graph theory, computability theory and reverse mathematics. For the basics of graph theory, we refer the reader to Diestel [2], although the necessary definitions for domatic partitions will be presented here. For background on computability theory we refer the reader to Soare [10] and for reverse mathematics we refer the reader to Simpson [9].

We begin with a few necessary definitions.

Definition 2.1. Let G be a graph with vertex set V. A subset $D \subseteq V$ is a *dominating* set if for every vertex $v \in V \setminus D$, v is adjacent to at least one vertex in D.

Definition 2.2. A *domatic partition* of a graph G is a partition of the vertices V of G for which each class in the partition is a dominating set. The *domatic number* of a graph, $d(G)$, is the maximal number of classes of a domatic partition. We say that G has a *domatic n-partition* if there is a domatic partition of G into n dominating sets.

It is sometimes useful to view a domatic n-partition as a function from the vertices of the graph to $\{0, \ldots, n-1\}$. This is more in line with the work done on chromatic number, and is helpful if we wish to define domatic partitions in RCA_0 for the sake of reverse mathematics.

Definition 2.3. (RCA_0) We say that a function $f : V \to k$ is a domatic k-partition of G if

$$(\forall v_1)(\forall v_2)[f(v_1) \neq f(v_2) \to (\exists v_3)[f(v_3) = f(v_2) \wedge E(v_3, v_1)]]$$
$$\wedge \bigwedge_{i=0}^{k-1} ((\exists x_i)[f(x_i) = i]) .$$

Note that by this definition, we know that detecting a domatic 3-partition in a computable graph is not harder than Σ^1_1.

We wish to investigate the complexity of (finding) domatic partitions. To this end, it will be useful to understand what features of a graph allow or disallow domatic partitions. First notice that the domatic number of a graph is bounded by $\delta(G) + 1$, where $\delta(G)$ is the minimal (vertex) degree of the graph. This is because if vertex v has degree $\delta(G)$, then it can be dominated by at most $\delta(G)$ many sets. So the total number of dominating sets is at most $\delta(G) + 1$ — the set that v is in, plus one set for each of the $\delta(G)$ many neighbors of v. On the other hand, there is a graph with domatic number 2 that has arbitrarily large minimum vertex degree, as shown in [11].

Every graph trivially has a domatic 1-partition (the entire vertex set of the graph forms the 1-set partition). It is easy to see that (classically) any graph with no isolated vertices (so with $\delta(G) \geq 1$) has a domatic 2-partition. In fact, we will show that this result is effective.

Proposition 2.4. *Let G be a computable graph with no isolated vertices. Then there exist computable dominating sets A and B which form a domatic partition of G.*

Proof. Let $\{v_1, v_2, \ldots\}$ be the vertices of G. We construct A and B in stages. At stage 1, place $v_1 \in A$. At stage n, place v_n in either A or B, determined as follows. Starting with v_1, check whether v_k is adjacent to v_n for each $k < n$. If there is at least one $k < n$ such that v_n is adjacent to v_k and $v_k \in A$, then put $v_n \in B$. If no such k exists, put v_n in A. Clearly this process partitions the vertices into the sets A and B. Further, A and B are computable — to determine whether $v_n \in A$, simply run the construction though stage n.

We claim that A and B are both dominating sets. Let v_n be a vertex of G. Suppose first that $v_n \notin A$. Then v_n was placed in B by the construction. The only way this could happen is if v_n was found to be adjacent to a vertex (with smaller index) already in A. So v_n is adjacent to a vertex in A. Therefore A is a dominating set.

On the other hand, suppose $v_n \notin B$. Then v_n was placed into A at stage n because there was no $k < n$ such that $v_k \in A$ and v_k is adjacent to v_n. If v_n is adjacent to some v_k with $k < n$, then it must be that v_k is in B. If not, then v_n is not adjacent to any v_k with $k < n$. But G contains no isolated vertices, so v_n is adjacent to some v_t with $t > n$. At stage t, we will place v_t into B, since v_t is adjacent to a vertex (with smaller index) already in A, namely v_n. So in either case, v_n is adjacent to a vertex in B, making B a dominating set. □ □

This proposition settles both types of questions we ask in this paper. First, given a computable graph with a domatic 2-partition, we know the graph has a computable domatic 2-partition. Second, given a computable graph, we now know the complexity of deciding whether or not the graph *has* a domatic 2-partition — it will if and only if the graph has no isolated vertices. To say that G has an isolated vertex is to say

$$\exists v \forall w [\neg E(v, w)].$$

Therefore the set of indices of computable graphs with a domatic 2-partition (no isolated vertices) is Π^0_2-definable. It would be a simple matter to prove that in this respect the decision problem is Π^0_2 complete. However, the complexity present here is due solely to the problem of determining the minimal degree for the graph. To exclude this, we might ask for the complexity of deciding whether a graph has a domatic 2-partition, *given the graph meets the minimal degree requirement.* Then the answer is that the decision problem is computable.

The remainder of the paper will be devoted to addressing these questions for larger numbers of partitions, where we will see the complexity can be considerably higher.

3. Computable Domatic Number

We wonder if Proposition 2.4 can be extended to larger numbers of partitions. That is, is there an algorithm for partitioning the graph into k dominating sets (provided such a partition exists)? We show that even if $k = 3$, the answer is no. This establishes that the domatic number of a graph need not agree with the *computable* domatic number.

Proposition 3.1. *There is a computable graph G with domatic number 3 containing no computable domatic 3-partition (i.e., G has computable domatic number 2).*

Proof. We will build a graph G computably, ensuring that G has a domatic 3-partition while simultaneously diagonalizing against all computable functions which appear to partition G into three sets. The final graph will consist of a collection of gadgets: either K_4 or K_4 together with a fifth vertex adjacent to two of the vertices of K_4. In both cases, these gadgets have a domatic 3-partition — for the graphs in Figure 1, each vertex is labeled with A, B, or C, denoting to which set in the partition it belongs.

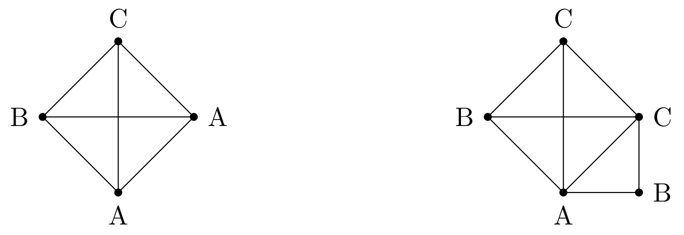

FIGURE 1. Domatic 3-partitions on gadgets with four and five vertices

Notice that each vertex is adjacent to at least one vertex in the other two sets, so each set in the partition is dominating. For the graph on the right, we had to reassign vertices to sets to allow the new vertex to be dominated by the set C, but this did not inhibit the creation of a domatic 3-partition. In fact, it is exactly this phenomenon we will exploit.

The construction of the graph will proceed in stages. Start at stage 1, by building a copy of K_3. This will ensure that the maximal number of classes in a domatic partition (i.e., the domatic number) of the final graph is 3. Also build a copy of K_4, and do so for each future stage. For each $e \leq s$ check whether $\varphi_e(v_i)$ has converged for all four vertices v_i in the copy of K_4 built at stage e. For each such e, if φ_e appears to partition the vertices of its copy of K_4 into three sets, then we act. Such a partition will assign each of the four vertices to one of three sets. Therefore there will be a pair of vertices in the copy of K_4 assigned to the same set. Use the first so far unmentioned vertex and set it adjacent to these two commonly labeled vertices. If no such e exists, simply continue with the construction. Finally, build another copy of K_4 (using four so far unmentioned vertices).

Clearly this construction gives a computable graph — to determine whether two vertices are adjacent, simply run the construction until a stage after each are first mentioned — the vertices will be adjacent if they are adjacent at that stage. The constructed graph will contain a domatic 3-partition as it consists entirely of copies of K_4 or K_4 plus an extra vertex as shown in Figure 1. However, there is no computable domatic 3-partition. Suppose, for contradiction, there were. Then it would be φ_e for some e and for each vertex v_i, $\varphi_e(v_i) \in \{A, B, C\}$. At some point $\varphi_e(v_i)$ would converge for all four v_i in the e-th copy of K_4 in the graph. Two vertices would be assigned the same set (say, without loss of generality, A). These two vertices would be adjacent to a fifth vertex v_k. Now if $\varphi_e(v_k) = B$ then C is not a dominating set, since no vertex in C is adjacent to v_k. Similarly, if $\varphi_e(v_k) = C$, then B is not a dominating set. Therefore φ_e cannot be a domatic 3-partition. □ □

The graph constructed in the proof of Proposition 3.1 is not connected, but if such a graph were desired, we could simply create a chain of the copies of K_4 — adding edges to a graph can only increase the domatic number.

We can do better with the following result.

Proposition 3.2. *For any natural number $n \geq 2$, there is a computable graph with domatic number n which has computable domatic number* 2.

Proof. Notice that K_2 works for $n = 2$, so fix a natural number $n \geq 3$. First, note that if a graph has a computable domatic k-partition, then the graph has a computable domatic j-partition for each $j \leq k$ — put vertices in the partitions numbered above j into the j-th partition. Thus all we need show is that there is a computable graph with domatic number n with no computable domatic 3-partition.

The graph will be built in much the same way as the previous construction. We build a copy of K_n at the beginning to ensure that the maximal number of classes in a domatic partition of the final graph (i.e., the domatic number) will be n. Now each gadget will consist of a copy of $K_{3(n-2)+1}$. We wait for φ_e to partition the vertices in its gadget into three sets. If φ_e does not do so, then it cannot be a domatic 3-partition. If it does, then there must be at least $n-1$ vertices in the gadget belonging to the same set (out of the three). At this stage, we pick a new vertex and connect it to the $n-1$ vertices. Now this new vertex is adjacent only to one of the three sets in the supposed partition, so there is at least one set which does not dominate it. However, the graph created can still be partitioned into n dominating sets by putting each of the $n-1$ vertices into different sets. □ □

Proposition 3.3. *For any pair of natural numbers n, k such that $2 \leq k \leq n$, there is a computable graph with domatic number n which has computable domatic number k.*

Proof. We will build a computable graph G having the desired property. First note that $G = K_n$ satisfies case $n = k$, so assume $k < n$. It suffices to build a computable graph G that has domatic number n, a computable k-partition, but no computable $(k+1)$-partition. The construction will be similar to that in Propositions 3.1 and 3.2.

Build the graph in stages, diagonalizing against φ_e, the proposed computable domatic $(k+1)$-partitions. Along the way, we build a computable domatic k-partition $p : V \to \{1, \ldots, k\}$. At stage $s = 0$, put in a copy of K_n to ensure that the maximal number of classes in a domatic partition (i.e., the domatic number) of G will be n. Put in a copy of $K_{(k+1)(n-k)+1}$, and we will continue putting one of these in at every subsequent stage, calling it the *gadget* of that stage. Initialize p by defining a domatic k-partition on G as follows. Starting at one vertex of the copy of K_n, label the vertices counterclockwise $1, \ldots, k$, and do the same on the gadget of $s = 0$.

At stage s, do the following. Put the gadget of stage s into G. Extend p in the same way as stage 0. For each $0 \leq e \leq s$, check whether φ_e has partitioned the gadget of stage e into exactly $k+1$ classes $A_1, \ldots, A_{k+1}$. If it has, "spring a trap" as follows. Add an extra vertex v to the gadget, and connect it to $n-k+1$ vertices that are in the same φ_e-class $A \in \{A_1, \ldots, A_{k+1}\}$. (Note that at least $n-k+1$ vertices in the gadget must be in the same φ_e-class.) Connect v to $k-2$ more vertices in the gadget so that now v is connected to vertices from all but at most one p-class — let b denote this one possibly missing p-class — this will be the p-class for v itself. (Note that such $k-2$ many vertices exist. Indeed, in the worst-case scenario, *all* of the original $n-k+1$ vertices connected to v will have the same p-value c. (Note $c \neq b$ because we assumed v is not connected to a vertex in the p-class b.) But there will still be $(k+1)(n-k)+1-(n-k+1) = k(n-k)$, that is at least k (because $k(n-k) \geq k$), more vertices to choose from in the gadget. And *all* of the remaining $k-1$ many p-classes, that is, each class in $\{1, \ldots, k\} \setminus \{c\}$, must be represented among those k or more vertices.) Finally, put v into the p-class b.

If φ_e does not partition the gadget into exactly $k+1$ classes or if we have already sprung a trap on the gadget once, do nothing. This ends the construction.

Verification. First, G has a domatic n-partition, because (1) K_n does, (2) each unsprung gadget (a copy of $K_{(k+1)(n-k)+1}$) does (because it has at least $n+1$ vertices, although only needing n), and (3) the vertex v of a sprung gadget is connected to $(n-k+1)+(k-2) = n-1$ many vertices on the gadget. Also, the domatic number of G is n, because K_n does not have a domatic $(n+1)$-partition. Next, p defines a k-partition on G, because it places all vertices adjacent to v in $k-1$ different classes (which include the possibly single p-class containing the original $n-k+1$ many vertices connected to v, plus the $k-2$ many p-classes containing the $k-2$ many extra vertices connected to v) and places v in the final, k-th, class.

Finally, there is no computable $(k+1)$-partition of G. For if there were a computable function φ_e that partitioned the e-th gadget into exactly $k+1$ classes, then we defeat φ_e from being a domatic partition by preventing it from dominating the class containing v. Indeed, v is adjacent to vertices in at most $k-1$ many φ_e-classes, because the original $n-k+1$ many vertices it connects to are all in a single φ_e-class, and v connects to only $k-2$ more vertices afterward. Therefore, G has computable domatic number k. □ □

Even if we restrict ourselves to the class of highly computable graphs, we cannot extend Proposition 2.4 to a computable domatic 3-partition, as we see from the following result.

Proposition 3.4. *There is a highly computable graph with domatic number* 3 *which has computable domatic number* 2.

Proof. We build the graph in a similar way as before. Now each gadget will consist of a copy of K_3 with two attached vertices (as shown in Figure 2, left). These two vertices are the start of two tails emanating from the copy of K_3. At every subsequent stage, add a vertex to one of the tails, alternating between the two tails ($v_1, v_3, \dots$ on the odd stages and $v_2, v_4, \dots$ on the even stages). Either both tails will grow indefinitely, or at a particular stage we will "spring a trap," i.e., connect the tails changing them into a loop (shown Figure 2, right).

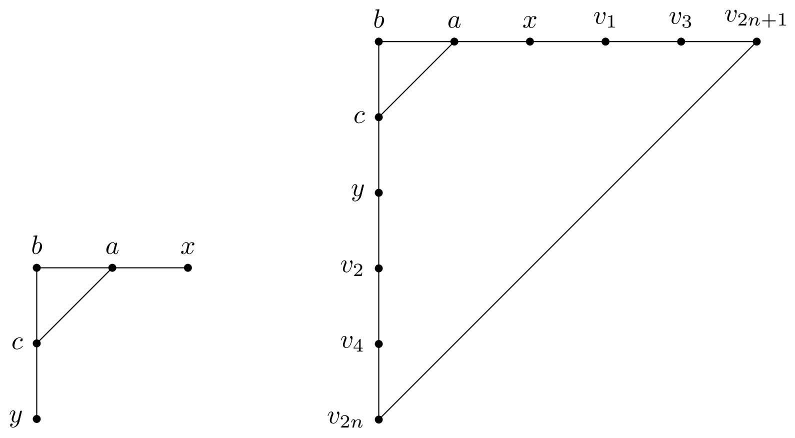

FIGURE 2. Gadget of K_3 with tails (left) and sprung trap (right)

Decide when to spring the trap, if at all, according to the following procedure. Wait until φ_e has given a domatic 3-partition of the four vertices a, b, c, and y. Notice a, b, and c must be in three different sets. Also because y will have degree 2, it must be in the same set as either a or b. If it's a, then spring the trap on the next stage s such that $s + 1$ is divisible by 3. If it's b, do so when $s + 2$ is divisible by 3. This ends the construction.

Notice, if we never spring the trap, then φ_e must not give a domatic 3-partition (otherwise it would have given such a partition of a, b, c, and y at some stage), and the gadget has a domatic 3-partition (illustrated in Figure 3).

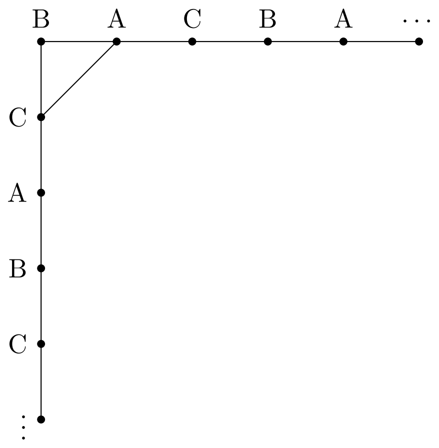

FIGURE 3. Domatic 3-partition on infinite tails

The two diagrams in Figure 4 demonstrate the sprung trap when y is in the same set as a or b, respectively. The left diagram gives an example when y and a

are in the same class, where we spring the trap at stage say $s = 5$ (noting 3 divides $s - 2 = 3$). The right diagram is when y and b are in the same class and we spring the trap at stage say $s = 4$ (noting 3 divides $s - 1 = 3$).

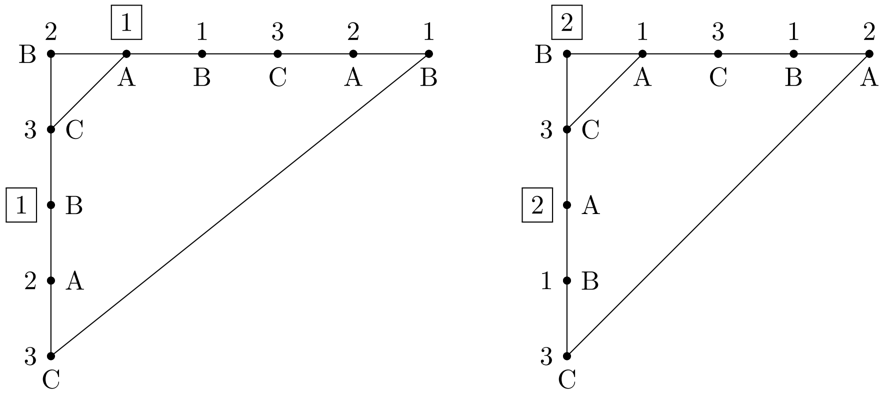

Figure 4. Traps sprung on a computable domatic 3-partition

The diagrams in Figure 4 suggest the following facts:

i. the 3-partition that φ_e gives cannot be domatic (shown as numbers $1, 2, 3$ — we started at y and labeled the vertices in the counterclockwise direction)
ii. there is a domatic 3-partition of the gadget (shown as letters A, B, C)

Finally the graph is highly computable, because in each gadget the vertices a, b, and c always have degree 3, and all other vertices are known to eventually have degree 2 from inception. □ □

The above results say that there is as much separation between domatic number and computable domatic number as possible — so even in highly computable graphs with domatic partitions, those partitions need not be computable. It is natural to ask how complicated the domatic partitions might be. It turns out that for highly computable graphs, the answer is not that complicated.

Proposition 3.5. *Every highly computable graph with domatic number n has a domatic n-partition of low degree.*

Proof. For the purposes of the proof, we will think of n-partitions as functions from the vertices to $\{0, \ldots, n-1\}$. Let G be a highly computable graph with domatic number n. List the set of vertices of G as $\{v_0, v_1, v_2, \ldots\}$. We will build a computable n-branching tree T such that the infinite paths through T correspond to the domatic n-partitions of G. For each $i \geq 0$, let G_i be the subgraph of G containing the vertices $V_i = \{v_0, \ldots, v_i\}$ and all vertices adjacent to elements of V_i. Suppose $\sigma \in n^{<\omega}$ and $|\sigma| = k + 1$. Treat σ itself as an n-partition of the vertices $v_0, \ldots, v_k$. Add σ to T if and only if there is an n-partition of G_k that agrees with σ on V_k and under which every element of V_k is dominated.

First, T is a tree, because the n-partition associated to a given $\sigma \in T$ also serves as an appropriate n-partition for every initial segment of σ. Next, we have:

$$f \in [T] \text{ iff } f \text{ is a domatic } n\text{-partition of } G.$$

The backward direction is clear. For the forward direction, given $f \in [T]$ and $v_k \in V$, take the restriction of f to the vertices of G_k. This is an n-partition of G_k under which v_k is dominated. Since v_k was arbitrary, f in its entirety is a domatic n-partition of G.

Finally, $[T]$ is nonempty, because G has a domatic n-partition by assumption. So by the Low Basis Theorem, $[T]$ has a low element, which implies that G has a domatic n-partition of low degree. □ □

For computable locally finite (but not highly computable) graphs, we can make a similar argument. However now the tree would be Σ^0_1-computable, since when defining G_k we need to ask whether there are any more vertices adjacent to the given vertex of V_k. Luckily the low basis theorem relativizes, so we quickly get the following.

Proposition 3.6. *Every computable locally finite graph with domatic number n has a domatic n-partition f with deg$(f)' \leq \mathbf{0}''$.*

Proof. Use $\emptyset'$ as an oracle in the proof of Prop. 3.5 along with the Relativized Low Basis Theorem. □ □

From the other side, we can argue that the degree of the domatic partition really can be this large.

Proposition 3.7. *There is a computable locally finite graph G which has a domatic 3-partition, but no domatic 3-partition computable from $\emptyset'$.*

Proof. The proof will be similar to that of Proposition 3.1. We will build a trap copy of K_4 for each $\emptyset'$-computable function with the goal of diagonalizing against these — when $\Phi_e^{\emptyset'}$ looks like a domatic 3-partition on the e-th trap, we act to ensure it is not. The difference now is that we must be able to "un-spring" a sprung trap.

Fix an enumeration of $\emptyset'$. At stage s of our construction, we only have the computable approximation $\emptyset'_s$ of $\emptyset'$, so we can only compute $\Phi_{e,s}^{\emptyset'_s}$. Now if this halts on the four vertices of the e-th trap, we act, springing the trap as we did in Proposition 3.1. However, at a later stage t, we might have $\emptyset'_t \neq \emptyset'_s$, and this might cause $\Phi_{e,t}^{\emptyset'_t}(x) \neq \Phi_{e,s}^{\emptyset'_s}(x)$ for some of the x in the e-th trap. In this case we must act again to ensure that $\Phi_{e,t}^{\emptyset'_t}$ cannot possibly be a domatic 3-partition on the e-th trap. This may happen many times, but only finitely many provided $\Phi_e^{\emptyset'}$ halts on the e-th trap.[1]

Now we explain how the e-th trap can get sprung and unsprung as many times as needed. We start with K_4 and wait for $\Phi_{e,s}^{\emptyset'_s}$ to halt on the four vertices. If it ever does, then we take the two vertices it puts into the same set (say v_i and v_j) and connect them via a path of 7 as yet unmentioned vertices. Doing so guarantees v_i and v_j must be in different sets in any 3-partition, and as such ensures $\Phi_{e,s}^{\emptyset'_s}$ is not a domatic 3-partition. To see this, consider the graph:

[1]This is because if $\Phi_e^{\emptyset'}(x)\downarrow$ then it does so with some finite use of the oracle, and there will be a stage s at which $\emptyset'_s = \emptyset'$ up to that use.

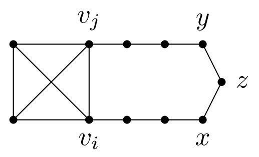

FIGURE 5. Possible trap sprung for the first time

Since v_i and x are connected by a 2-path, any domatic 3-partition must put v_i and x in the same set. Similarly, v_j and y must be put in the same set. However, to make z dominated, x and y must be in different sets, in turn forcing v_i and v_j to be in different sets.

To "un-spring" the trap, we can simply attach a triangle to z — put in two new vertices, adjacent to each other and each adjacent to z. Now z can be dominated by these two new vertices, releasing vertices x and y to once again possibly be in the same set (if we want v_i and v_j to be in the same set). We would do this if at a later stage t we had $\Phi_{e,t}^{\emptyset'_t}$ put v_i and v_j into different sets. Adding this triangle to z does not force v_i and v_j to be in the same set, but rather removes any restrictions on them. We would then spring a new trap on whichever pair of vertices from the original K_4 were placed in the same set by $\Phi_{e,t}^{\emptyset'_t}$.

All that is left to say is how to spring the trap on a pair of vertices that we have previously addressed. We could simply attach a new 7-vertex path, but this increases the degree of the vertices there, which might lead to a vertex with infinite degree (in the case that $\Phi_e^{\emptyset'}$ turns out to diverge on the vertices of its K_4). To avoid this, we instead attach the 7-vertex path to the previous 7-vertex path for the same two vertices. Using the graph in Figure 5, we would build this 7-vertex path off of vertices x and y. This would force any domatic 3-partition to put x and y in different sets, which in turn puts v_i and v_j into different sets. Figure 6 shows what the graph might look like after we spring the trap a third time:

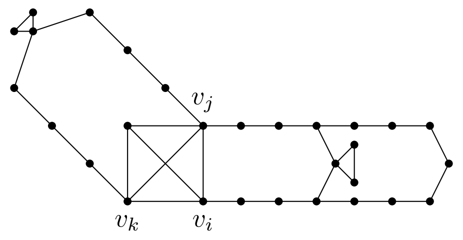

FIGURE 6. Possible trap sprung for the third time

Here we first forced v_i and v_j to be in different sets, then released the requirement. Then we forced v_k and v_j to be in different sets, and then released that requirement as well. Finally, we returned to v_i and v_j, which are currently forced to be in different sets. This ends the construction.

First notice that if there is no s for which $\Phi_{e,s}^{\emptyset'_s}$ converges on all four of the original vertices of the e-th trap, then $\Phi_e^{\emptyset'}$ is not a domatic 3-partition of G. Also, if $\Phi_{e,s}^{\emptyset'_s}$ converges on the original four vertices of the e-th trap and does not produce a domatic 3-partition, then $\Phi_{e_s}^{\emptyset'_s}$ cannot be a domatic 3-partition of G, but there might be some larger s for which the computation changes and it is. So assume that it does converge and produce a domatic 3-partition. Then we spring the trap as described above. Then if, at some later stage t, $\emptyset'_t$ changes, resulting in $\Phi_{e,t}^{\emptyset'_t} \neq \Phi_{e,s}^{\emptyset'_s}$, then $\Phi_{e,t}^{\emptyset'_t}$ could possibly be a domatic 3-partition of the sprung trap. In this case we un-spring the sprung trap by adding a triangle and spring a different trap for $\Phi_{e,t}^{\emptyset'_t}$. If the computation of $\Phi_e^{\emptyset'}$ is going to converge, then the computation of $\Phi_{e,s}^{\emptyset'_s}$ can change at most finitely many times. Therefore, in the case that $\Phi_e^{\emptyset'}$ converges, eventually we will have sprung a final trap for which $\Phi_e^{\emptyset'}$ is not a domatic 3-partition. In any case, by the way we have constructed G itself, it is easy to see that G always has a domatic 3-partition, even if the trap is sprung and unsprung infinitely often (in this case $\Phi_e^{\emptyset'}$ does not converge and therefore cannot be a domatic 3-partition of G). □ □

4. Decision Problems

We turn now to the question of how hard it is to detect a domatic partition in a graph. We begin with the case when the vertices of $G = (V, E)$ all have finite degree. For a finite set of vertices V_0 in a graph, define $V_0^{+\,\mathrm{adj}}$ to be V_0 together will all vertices adjacent to at least one vertex from V_0, i.e.,

$$V_0^{+\,\mathrm{adj}} := \{v \in V : (\exists u \in V_0)[vEu]\}.$$

Proposition 4.1. *Let $n \geq 3$. If G is a computable locally finite graph, then G has a domatic n-partition iff the following condition holds:*

(1) *for every finite set V_0 of vertices, the subgraph induced by $V_0^{+\,\mathrm{adj}}$ has an n-partition in which V_0 is dominated.*

So the set of indices of computable locally finite graphs which have a domatic n-partition is in Π^0_2.

Proof. The forward direction is clear, as the domatic n-partition of G itself will work for each finite set V_0. To prove the backward direction, suppose (1) holds. Let $V = \{v_0, v_1, v_2, \ldots\}$ be the vertex set of G. For each k, let $V_k = \{v_0, \ldots, v_k\}$. We consider a finite-branching tree of n-partitions of V_k for all k (where τ extends σ in T provided τ is an extension of σ as an n-partition). For n-partition σ on V_k, we put σ into T if and only if σ extends to an n-partition of $V_k^{+\,\mathrm{adj}}$ in which each vertex in V_k is dominated.

We claim that this really is a tree: if $\tau \in T$ and σ is a prefix of τ, then $\sigma \in T$. Now σ is a prefix of τ provided τ is an extension of σ (as an n-partition). Since $\tau \in T$, there is an n-partition of $V_k^{+\,\mathrm{adj}}$ in which every vertex in V_k is dominated, where V_k is the domain of τ. The extension that works for τ will also be an extension of σ, and since the domain of σ is a subset of the domain of τ, every vertex in the domain of σ will be dominated. Note that the tree is finite-branching because there are finitely many n-partitions for each k.

By (1), T is infinite. Thus by König's Lemma, T has an infinite path f. We claim that f is a domatic n-partition of G, that is, every vertex v_k of G is dominated by f. Indeed, fix a vertex v_k. Consider $V_k^{+\,\mathrm{adj}}$ and the restriction $\hat{f}$ of f to this finite set. We have $\hat{f} \in T$, since f is a path of T. But this means that $\hat{f}$ has an extension in which all the vertices of $V_k^{+\,\mathrm{adj}}$ are dominated, including v_k. Now v_k is only adjacent to vertices in $V_k^{+\,\mathrm{adj}}$, so in fact v_k is dominated by $\hat{f}$ itself and thus by f. □ □

Proposition 4.1 shows that detecting a domatic partition in a computable locally finite graph is Π_2^0-definable. The following proposition shows hardness for Π_2^0, and therefore we have classified the problem as being Π_2^0-complete. Also, since the following proposition is proved for graphs with minimal vertex degree larger than 1, we see that the Π_2^0-hardness does not arise solely from vertices of degree 1.

Recall that the index set $\mathrm{Inf} = \{e \in \omega : |W_e| = \infty\}$ where W_e denotes the e-th c.e. set (i.e., the domain of the e-th partial computable function φ_e) is Π_2^0-complete.

Proposition 4.2. *There is a computable sequence of computable locally finite graphs $\langle G_i = (V_i, E_i) : i \in \omega \rangle$, all of which have minimal vertex degree 2, such that $e \in \mathrm{Inf}$ if and only if G_e has a domatic 3-partition.*

Proof. To build G_e for each $e \in \omega$, enumerate W_e in stages. Dovetail the construction so that we eventually build every G_e. Fix $e \in \omega$. At stage $s = 0$, initialize G_e by putting in the following pair of triangles, where each triangle has a string of four vertices as its base. Note vertices v_0 and v_1 labeled in Figure 7 for the purposes of stage $s + 1$.

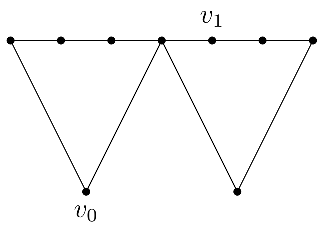

FIGURE 7. Initialization of G_e at stage $s = 0$

At stage $s+1$, if $W_{e,s+1} \setminus W_{e,s} \neq \emptyset$, that is if an element enters W_e, then extend G_e as follows. Add a string of three new vertices between v_0 and v_1, and append a new triangle to the right of G_e, as in Figure 8. Otherwise, go to the next stage.

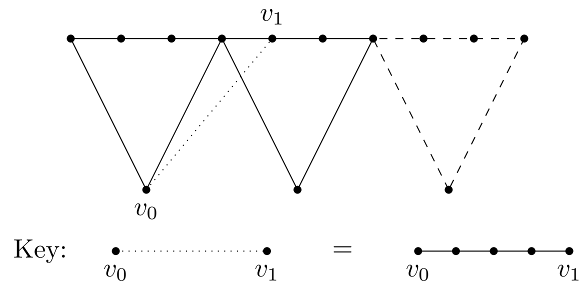

FIGURE 8. Connections made in G_e when an element enters W_e

If $e \in \mathrm{Inf}$, then W_e is infinite, and an element enters it infinitely often. So as we demonstrate in Figure 9, G_e has a domatic 3-partition.

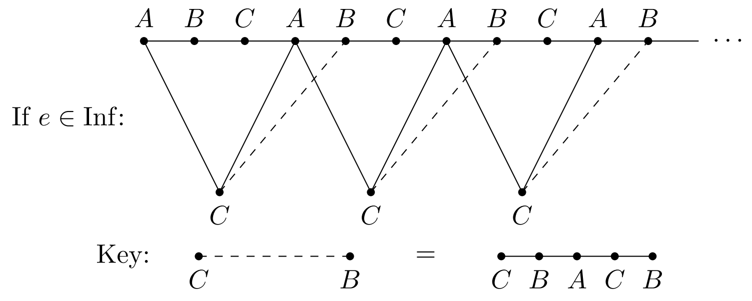

FIGURE 9. What G_e looks like if $e \in \mathrm{Inf}$

On the other hand, if $e \notin \mathrm{Inf}$, then W_e is finite, and there is a final triangle that we appended to the right of G_e. So as we demonstrate in Figure 10, G_e does not have a domatic 3-partition. To see this, note that the two vertices colored A were forced to be colored the same if we are to have a domatic 3-partition. So without loss of generality we indeed colored each of them A. Now the bottom vertex v_2 cannot be dominated.

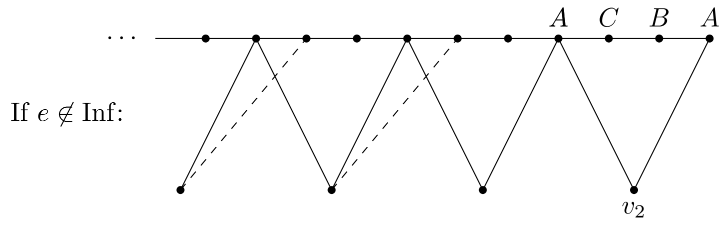

FIGURE 10. What G_e eventually looks like if $e \notin \mathrm{Inf}$

□ □

Corollary 4.3. *There is a computable sequence of computable locally finite graphs $\langle G_i = (V_i, E_i) : i \in \omega\rangle$, with minimal vertex degree larger than one, such that $\emptyset''$ is computable in*

$$\{k \in \omega : G_k \text{ has a domatic } 3\text{-partition}\}.$$

Notice that in Proposition 4.1, the defining sentence for G having a domatic n-partition says something to the effect of "for every finite set of vertices V_0, there is a finite extension $B \supseteq V_0$ such that in the subgraph of G induced by B, there is a partition of the vertices of B in which each vertex of V_0 is dominated." In that proposition this is truly a $\forall\exists$ sentence, as we know that B is finite because G is a locally finite graph, but we do not know the complete extent of B because even though it is finite, we cannot computably say when we will be done enumerating its vertices (also, it is computable to find a domatic n-partition of a finite graph). When we look at the case in which G is a highly computable graph, we computably know all of the neighbors of a given vertex, so given a finite set of vertices V_0, we can computably determine the subgraph of G induced by the set consisting of all the vertices of V_0 together with all of the vertices that are neighbors of vertices of V_0 (in this case, B is the set consisting of V_0 together with all of its neighbors). So the defining sentence for G having a domatic n-partition is reduced to being a $\forall$ sentence. Therefore the decision problem there is in Π^0_1. That the decision problem is Π^0_1-complete is a corollary to the following.

Proposition 4.4. *There is a computable sequence of highly computable graphs $\langle G_i = (V_i, E_i) : i \in \omega\rangle$, all of which have minimal degree 2, such that $k \in \overline{K}$ iff G_k has a domatic 3-partition.*

Proof. Build the graphs G_i in stages, and dovetail their construction so that we eventually build all the graphs. Fix k. Build G_k to be a string of connected vertices that grows in length outwardly in both directions as the stage number of the construction increases. At the same time, enumerate K. If k enters K, connect the two ends of the current finite string of vertices. This turns G_k into a loop similar to that used in the proof of Proposition 3.4. When connecting the two endpoints, add sufficiently many vertices to ensure that no domatic 3-partition of the loop exists (i.e., make the loop have length 1 or 2 mod 3). □ □

What is perhaps surprising is that allowing vertices to have infinite degree elevates the complexity of the decision problem as high as it can possibly go.

Theorem 4.5. *The set of indices of computable graphs which have a domatic 3-partition is Σ^1_1-complete.*

Proof. We have already seen that the set of indices of computable graphs which have a domatic 3-partition is in Σ^1_1, now we demonstrate Σ^1_1-hardness. By Theorem XX in Chapter 16 of Rogers [8], it suffices to show that given any computable tree T, there is a computable graph G such that

(2) G has a domatic 3-partition iff T has an infinite path.

Here is how to construct G. Assume that T has elements of length at least 1. The root node of T will be represented by v_0. Adjacent to v_0 are vertices v_1 and v_σ for each $\sigma \in T$ with $|\sigma| = 1$. For each $n > 1$ for which there is some $\tau \in T$ of length n, we put in vertices v_n and u_n with edges (v_{n-1}, u_n) and (u_n, v_n).

Additionally, for each $\sigma \in T$ with $|\sigma| = n$, we will have vertices v_σ, v_σ^{lw}, v_σ^l, v_σ^r, and v_σ^{rw} (the superscripts indicating the positions of the vertices relative to v_σ, each respectively standing for *left wing*, *left*, *right*, and *right wing*) with edges (v_n, v_σ^{lw}), $(v_\sigma^{lw}, v_\sigma^l)$, (v_σ^l, v_σ), (v_σ, v_σ^r), $(v_\sigma^r, v_\sigma^{rw})$ and (v_σ^{rw}, v_n).

Finally, whenever $\tau \in T$ is an immediate successor of a non-root $\sigma \in T$, we put in edges (v_σ^l, v_τ), (v_σ, v_τ), and (v_σ^r, v_τ).

The resulting G is not easy to draw even for simple trees, but Figure 11 shows an example.

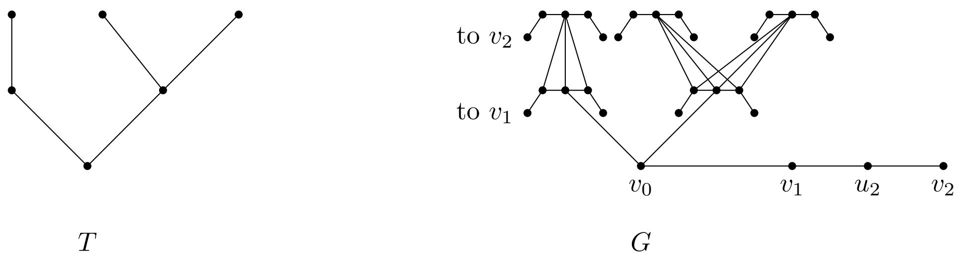

Figure 11. A tree T and its corresponding graph G

In G, note the instances of the 5-vertex "wing" gadget, consisting of vertices v_σ^{lw}, v_σ^l, v_σ, v_σ^r, and v_σ^{lw} for each $\sigma \in T$, $|\sigma| \geq 1$. The wing gadget is shown in Figure 12.

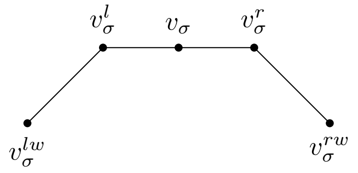

Figure 12. The wing gadget

For readability purposes, we placed the phrase "to v_n" (for $n = 1, 2$) in the picture of G, to indicate that the vertices v_σ^{lw} and v_σ^{rw} in the wing gadgets for $|\sigma| = n$ are adjacent to v_n.

Note that G will be computable (given that T is). We must argue that G has a domatic 3-partition if and only if T has an infinite path. Consider the vertices near a single v_σ with $|\sigma| = n$, $|\kappa| = n - 1$ and $|\tau| = n + 1$ to see what can happen. In Figure 13, σ is a child of κ. If $|\sigma| = 1$, then the dashed edges are just a single edge leading down to v_0 (that is, $v_\kappa = v_0$). The dotted edges leading up to v_τ only exists if there is a v_τ — if σ has a child (τ).

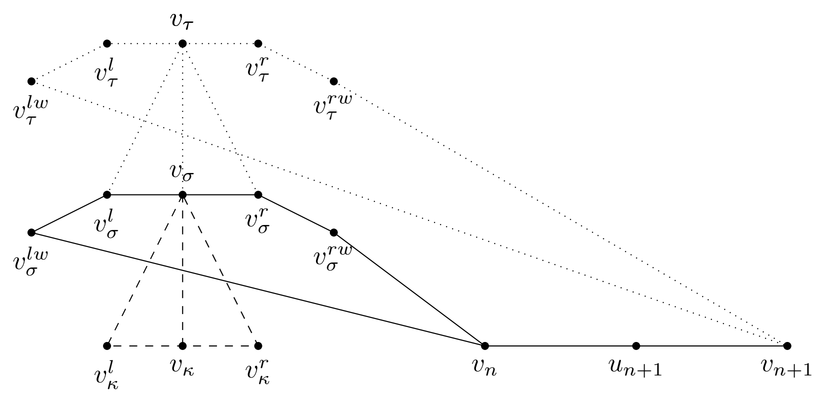

FIGURE 13. Part of G with corresponding nodes κ, σ, and possibly τ from T

First, for the backward direction of (2), we will show that the graph does have a domatic 3-partition if T has an infinite path. Suppose T has an infinite path. Construct a domatic 3-partition of G as follows. Put $v_0 \in A$ and $v_1 \in B$. For each $\sigma \in T$ with $|\sigma| = 1$ put $v_\sigma \in B$ for σ off the path and $v_\sigma \in C$ for σ on the path. Notice now that v_0 is dominated, and if we partition the v_σ^q, $q \in \{lw, l, r, rw\}$, in the obvious way (alternating A and C), we also have v_1 and each v_σ dominated, for σ off the path. Additionally, all v_σ^q are dominated except (so far) for v_σ^l and v_σ^r for the σ on the path — these vertices are not adjacent to any vertex in the set B. We will fix this by putting $v_\tau \in B$, where τ is the element of length 2 on the path. Notice this makes v_σ now dominated, for σ on the path. For all other $\tau \in T$ of length 2, we put $v_\tau \in C$ as well as $v_2 \in C$. The v_τ^q are alternately put into A and B, with $u_2 \in A$. Again, v_τ is dominated for each τ of length 2 off the path, as is v_2 and u_2. For each τ off the path, all the v_τ^q are dominated — we need only worry about v_τ^l and v_τ^r for τ on the path — these are not adjacent to any vertex in C (yet). But because τ is on the path, there is another vertex adjacent to both these which we can put into C, which also makes v_τ now dominated. And so on.

The resulting partition of this process has $v_0 \in A$, $v_n \in B$ for odd n and $v_n \in C$ for even $n > 1$. For each $\sigma \in T$ not in the path, $v_\sigma \in B$ if $|\sigma|$ is odd, and $v_\sigma \in C$ if $|\sigma| \neq 0$ is even. For each $\sigma \in T$ in the path, $v_\sigma \in C$ if $|\sigma|$ is odd, and $v_\sigma \in B$ if $|\sigma| \neq 0$ is even. All the other vertices of G are partitioned in the obvious way.

We now prove the forward direction of (2). Again referring to Figure 13, for two strings σ and τ, we say the vertex v_σ *has a pyramid* to the vertex v_τ if the edges (v_σ^l, v_τ), (v_σ, v_τ), and (v_σ^r, v_τ) exist — of course by the construction, if one of these edges exist, all three do.

Claim 4.5.1. Fix a string σ of length n. Let p be a 3-partition of G in which the vertices v_σ^{lw}, v_σ^l, and u_{n+1} are dominated. If $p(v_\sigma) \neq p(v_n)$, then v_σ has a pyramid to a vertex v_τ for some string τ (so $\tau \supset \sigma$ and $|\tau| = n + 1$), and $p(v_\tau) = p(v_n)$. So $p(v_\tau) \neq p(v_{n+1})$.

of Claim. Since v_σ^{lw} is dominated, v_σ^l and v_n are in different partition sets. So letting A, B, C denote the partition sets, without loss of generality, v_σ^{lw} is in A, v_σ^l is in B, and v_n is in C. Since v_σ and v_n are in different sets, v_σ is in A or

B. Then the degree of v^l_σ must be larger than 2, because neither it nor either of the two vertices to which it is adjacent that we have thus far mentioned are in C. Therefore, by the construction, the edge (v^l_σ, v_τ) exists for some string τ. So the edges (v_σ, v_τ) and (v^r_σ, v_τ) also exists, and v_σ has a pyramid to a vertex v_τ. Also by the construction, this means that $\tau \supset \sigma$ and $|\tau| = n+1$. Notice, therefore, that $p(v_\tau) = C = p(v_n)$ in order to dominate v^l_σ. Because u_{n+1} is dominated, the only two vertices to which it is adjacent cannot be in the same set, namely the vertices v_n and v_{n+1}, so $p(v_\tau) = p(v_n) \neq p(v_{n+1})$. □ □

We will conclude the proof of the forward direction of (2) by showing that if G has a domatic 3-partition, then T has an infinite path. To see this, fix a domatic 3-partition p of G. Let 0 denote the empty string. Recursively define an infinite sequence $\{\sigma_n\}$ of strings in T as follows:

- Let $\sigma_0 = 0$ (the empty string).
- For $n \geq 1$, let σ_n be the leftmost string σ such that $\sigma \supset \sigma_{n-1}$, $|\sigma| = n$, and $p(v_\sigma) \neq p(v_n)$.

We use induction to show that for every $n \geq 0$, the string σ_n exists. The base case $n = 0$ is clear. Before the induction case, we first argue the $n = 1$ case. Every vertex in G is dominated, so in particular, $v_{\sigma_0} = v_0$ is dominated. So, because v_0 is adjacent to v_1, v_0 must be adjacent to some other vertex besides v_1 in a different partition set from that of v_1 (and v_0 for that matter). Then by the construction of G, there is a string $\sigma \supset \sigma_0$ such that $|\sigma| = 1$ and $p(v_\sigma) \neq p(v_1)$. Hence σ_1 is the leftmost such σ, so σ_1 exists. For the induction case, fix $n \geq 1$, and assume σ_n exists. Since $v^{lw}_{\sigma_n}$ and $v^l_{\sigma_n}$ are dominated, the claim above shows there is a string $\tau \supset \sigma_n$ such that $|\tau| = n+1$ and $p(v_\tau) \neq p(v_{n+1})$. Hence σ_{n+1} is the leftmost such τ, so σ_{n+1} exists.

Notice the sequence $\{\sigma_n\}$ satisfies the following conditions:

i. $\{\sigma_n\}$ is an infinite sequence of strings in T;
ii. for every $n \geq 0$, there is exactly one string in $\{\sigma_n\}$ of length n, namely σ_n;
iii. for every $n \geq 0$, there is an extension of string σ_n with length $n+1$ in the sequence $\{\sigma_n\}$, namely σ_{n+1}.

Therefore, $f : \omega \to \omega$ by $f(n) = \sigma_{n+1}(n)$ defines an infinite path in T, completing the proof of the theorem. □ □

5. Conclusion and Open Questions

We have confirmed that determining whether a computable graph has a domatic 3-partition is Σ^1_1-complete, and is therefore at the same computational level as determining whether a computable graph has a Hamilton path. On the other hand, if we restrict the class of graphs to highly computable or even locally finite graphs, then the problem is arithmetical. The complexity of those domatic partitions once found need not be computable, although for locally finite computable or highly computable graphs, the complexity cannot be too severe.

There are plenty of questions left for investigation. We have not addressed questions of uniformity: if we know G has a *computable* domatic 3-partition, how hard is it to locate it (given, say, an index for G)? Additionally, our decision problems show hardness for domatic 3-partitions. Can we argue the same holds for domatic k-partitions for any $k > 2$? We think the answer is yes, but the gadgets required become considerably more intricate.

We might also consider this topic from a reverse mathematics vantage point. For example, by our construction in Theorem 4.5, we can use the following theorem of Simpson [9] to prove a corollary. Note Corollary 5.2 is analogous to Theorem 20 of Gasarch and Hirst [4], on the decision problem for Hamilton paths.

Theorem 5.1 (Simpson, Lemma 1.1, Ch VI in [9])**.** (RCA_0) *The following are equivalent.*

(1) Π^1_1 *comprehension.*
(2) *For any sequence of trees* $\langle T_k : k \in \mathbb{N}\rangle$, $T_k \subseteq \mathbb{N}^{<\mathbb{N}}$, *there exists a set* X *such that* $\forall k(k \in X \leftrightarrow T_k$ *has a path*$)$*.*

Corollary 5.2. (RCA_0) *The following are equivalent.*

(1) Π^1_1 *comprehension.*
(2) *If* $\langle G_i : i \in \mathbb{N}\rangle$ *is a sequence of graphs, then there is a set* $Z \subseteq \mathbb{N}$ *such that for all* $i \in \mathbb{N}$, $i \in Z$ *if and only if* G_i *has a domatic* 3*-partition.*

References

[1] D. Bean, *Effective coloration*, Journal of Symbolic Logic, Vol. 41, No. 2 (1976), 469–480.
[2] Diestel, Reinhard, *Graph Theory*, Second Edition, Springer, New York, 2000.
[3] W. Gasarch, *A Survey of Recursive Combinatorics*, Handbook of Recursive Mathematics Volume 2, Edited by Ershov, Goncharov, Marek, Nerode, and Remmel, Elsevier (1998), 1041–1176.
[4] W. Gasarch, J. Hirst, *Reverse mathematics and recursive graph theory*, MLQ Math. Log. Q., Vol. 44, No. 4 (1998), 465–473.
[5] D. Harel, *Hamiltonian Paths in Infinite Graphs*, Israel Journal of Mathematics, Vol. 76 (1991), 317–336.
[6] J. Hirst, S. Lempp, *Infinite Versions of Some Problems from Finite Complexity Theory*, Notre Dame J. Formal Logic Vol. 37, No. 4 (1996), 545–553.
[7] T. Riege, *The Domatic Number Problem: Boolean Hierarchy Completeness and Exact Exponential-Time Algorithms*, Ph.D. Dissertation, Heinrich-Heine-Universität Düsseldorf, Düsseldorf, (2006).
[8] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*, The MIT Press, (1987).
[9] S. G. Simpson, *Subsystems of Second Order Arithmetic*, Springer-Verlag, New York, (1998).
[10] R.I. Soare, *Recursively Enumerable Sets and Degrees*, Perspectives in Mathematical Logic, Springer-Verlag, New York, (1987).
[11] B. Zelinka, *Domatic number and degrees of vertices of a graph* Mathematica Slovaca, Vol. 33, No. 2 (1983), 145–147.

Department of Mathematics, Manhattan College, 4513 Manhattan College Parkway, Riverdale, NY 10471, USA
E-mail address: `matthew.jura@manhattan.edu`

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

Department of Mathematics, Manhattan College, 4513 Manhattan College Parkway, Riverdale, NY 10471, USA
E-mail address: `tyler.markkanen@manhattan.edu`