5-2018

Weak Colorings of Computable Hypergraphs

Conner Hatton
University of Northern Colorado

University of Northern Colorado

Greeley, Colorado

Weak Colorings of Computable Hypergraphs

A Thesis Submitted in Partial Fulfillment for Graduation with Honors Distinction and the Degree of Bachelor of Science

Conner Hatton

College of Natural and Health Sciences

May 2018

Weak Colorings of Computable Hypergraphs

PREPARED BY: _____

APPROVED BY: _____

HONORS ADVISOR: _____

HONORS DIRECTOR: _____

RECEIVED BY THE UNIVERSITY THESIS PROJECT COMMITTEE

ON:

05/05/2018

Weak Colorings of Computable Hypergraphs

May 7, 2018

Contents

Abstract 1

Acknowledgments 2

1 Introduction 2

2 Preliminaries 3

2.1 Hypergraphs 3

2.2 Hypergraph Colorings 5

2.3 Computability 6

3 Computable Hypergraphs 8

4 Highly Computable Hypergraphs 15

5 Conclusion 19

Abstract

After introducing the reader to hypergraphs and their colorings, we generalize computable and highly computable graphs to develop the notion of computable and highly computable hypergraphs. If for a graph G we define $\chi(G)$ as the chromatic number of G and $\chi^C(G)$to be the computable chromatic number of G, then Bean showed that for every connected computable and highly computable graph G where $\chi(G) = 2$, then $\chi^C(G) = 2$. We show that there exists a 3-uniform, connected hypergraph $\mathcal{H}$ such that $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) = \infty$. Furthermore, we show that there exists a connected highly computable hypergraph $\mathcal{H}$ such that $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) = 3$. Lastly, we show that for every highly computable hypergraph $\mathcal{H}$ where $\chi(\mathcal{H}) = k$, it follows that $\chi^C(\mathcal{H}) \leq 2k$.

Keywords: Hypergraphs, Colorings, Computability

Acknowledgments

Foremost, I would like to express my gratitude to my adviser, Dr. Oscar Levin, who allowed me to complete this thesis under his guidance. He provided copious amounts of feedback, support, and knowledge. I would also like to thank the Mathematics and Honors departments at the University of Northern Colorado for providing endless opportunities for my education. Lastly, I would like to thank my family and friends for their constant support.

1 Introduction

Computability theory studies the nature of computation and its relationship with various mathematical structures. We may call any structure, or property of a structure, computable if it is given by an algorithm. For instance, computable sets are sets in which an algorithm exists that determines element membership for any particular potential member. Since we only consider graphs that are countable, it is natural to define a computable counterpart. Doing this allows us to compare the attributes of graphs and their computable companions. In particular, we study computable colorings on computable graphs. For graphs, we define a coloring to be a labeling of the vertices with natural numbers such that no edge has two vertices which are labeled the same. For any particular graph G, we can find the minimum number of colors necessary to properly color the vertices of G. We call this the chromatic number of G and denote it by $\chi(G)$. Similarly, we can determine the minimal coloring given by all algorithms. We call this the computable chromatic number and denote it by $\chi^C(G)$. For any finite graph G it is known that $\chi(G) = \chi^C(G)$[3]. However, Bean, who studied computable graph theory, showed in his 1976 paper "Effective Coloration"[1], that there exists computable graphs G such that $\chi(G) \neq \chi^C(G)$. In fact, Bean showed there exists a graph G such that $\chi(G) = 3$ and $\chi^C(G) = \infty$. Given

that algorithms may be arbitrarily bad at coloring some computable graphs, Bean considered a stronger notion of computability for graphs called highly computable graphs. Bean showed that for any highly computable graph G that if $\chi(G) = k$, then $\chi^C(G) \leq 2k$ [1] (It was later shown in [3] that $\chi^C(G) \leq 2k - 1$.)

In this paper, we provide generalizations of computable and highly computable graphs to develop the notion of computable and highly computable hypergraphs. We then show that there exists a connected 3-uniform computable hypergraph $\mathcal{H}$ where $\chi(\mathcal{H}) = 2$, yet $\chi^C(\mathcal{H}) = \infty$. This result provides greater motivation for the study of computable hypergraphs because of the graph theory result that for any connected graph G where $\chi(G) = 2$, then $\chi^C(G) = 2$ [3]. We further this result by constructing a connected, highly computable hypergraph $\mathcal{H}$ such that $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) = 3$. Finally, we provide a similar bound as Bean on the computable chromatic number of highly computable hypergraphs $\mathcal{H}$. That is, if $\chi(\mathcal{H}) = k$, then $\chi^C(\mathcal{H}) \leq 2k$.

2 Preliminaries

2.1 Hypergraphs

Definition 2.1. Let $V = \{v_1, v_2, \ldots, v_n\}$ be a finite set, and let $E = \{e_1, e_2, \ldots e_m\}$ be a family of subsets of V. We call $\mathcal{H} = (V, E)$ a **hypergraph** whose vertex set is V and whose edge set is E.

Hypergraphs are a generalization of standard graphs or graphs where all edges connect only two vertices, in that, the edges of a hypergraph, called hyperedges, can connect more than two vertices. In fact, we can see that for any set of vertices V the edge set E can contain any element of the power set of V. We use the definitions and notations used by Voloshin [6].

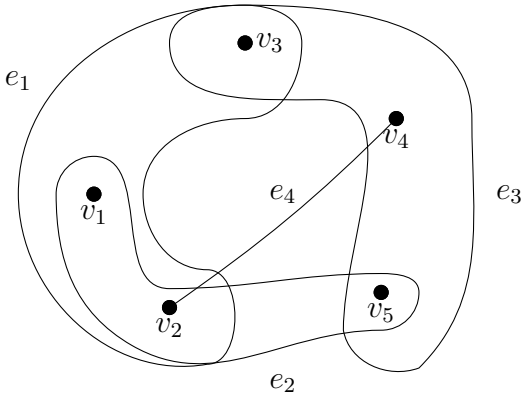

Figure 1: A hypergraph

Definition 2.2. We call $|V| = n$ the **order** of a hypergraph.

Definition 2.3. We say that two *vertices* v_i, v_j are **adjacent** if there is an edge $e_k \in E$ such that $v_i, v_j \in e_k$.

Definition 2.4. Two *edges* e_i, e_j are **adjacent** provided $e_i \cap e_j \neq \emptyset$, i.e. if they share a vertex.

Definition 2.5. The **degree** of a *vertex* is the number of edges which contain that vertex and is denoted by $|v_i|$.

Definition 2.6. The **degree** of an *edge* is the number of vertices contained within that edge and is denoted by $|e_i|$.

Definition 2.7. A **simple** hypergraph is a hypergraph in which no edges are subsets of another edge. That is, there is no e_i such that $e_i \subseteq e_k$ for any $i \neq k$.

For this paper, all hypergraphs will be simple.

Definition 2.8. A **k-uniform** hypergraph is a hypergraph whose edges all have degree equal to k.

Definition 2.9. A **complete k-uniform** hypergraph is the simple hypergraph $K_n^k = (X, \mathcal{D})$ such that $|X| = n$ and $\mathcal{D}(K_n^r)$ coincides with all the k-subsets of X. The complete k-uniform hypergraph and the family of its edges both are denoted by K_n^k.

Figure 2 illustrates all four complete hypergraphs on three vertices.

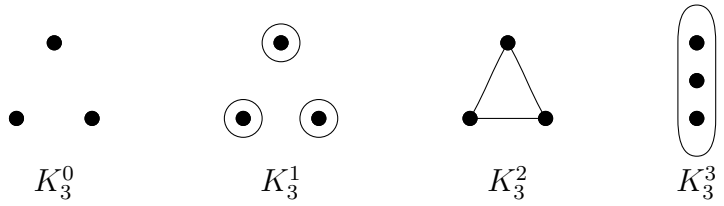

Figure 2: Complete hypergraphs on three vertices

2.2 Hypergraph Colorings

There are a few different notions for a coloring of a hypergraph, but we only consider weak colorings (we could use strong or mixed colorings).

Definition 2.10. A **weak k-coloring** of a hypergraph $\mathcal{H} = (V, E)$ is a labeling of its vertices V with the colors from the set $\{1, 2, \ldots, k\}$ in such a way that every edge $e_i \in E$ such that if $|e_i| \geq 2$ has at least two vertices colored differently.

Therefore a weak coloring of a hypergraph $\mathcal{H}$ is a labeling of the vertices with natural numbers such that no edge is monochromatic. Notice that this is a generalization of the colorings of graphs since, for any edge of degree two, weak colorings always give a proper coloring.

Definition 2.11. We call the minimum k colors used in which a weak coloring is successful the **chromatic** number. The chromatic number of a hypergraph $\mathcal{H}$ is denoted by $\chi(\mathcal{H})$.

In Figure 3, we see two hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$ with five vertices. Notice that $\chi(\mathcal{H}_1) = 2$ since the current coloring satisfies the coloring condition and uses only two colors, which is the least number of colors possible for any hypergraph. However, $\mathcal{H}_2$ uses three colors, which is a proper coloring, but observe that both edges will still be non-monochromatic if we color the vertex that is currently colored with a 3 with a 1. Therefore, despite the current coloring using three colors, $\chi(\mathcal{H}_2) = 2$.

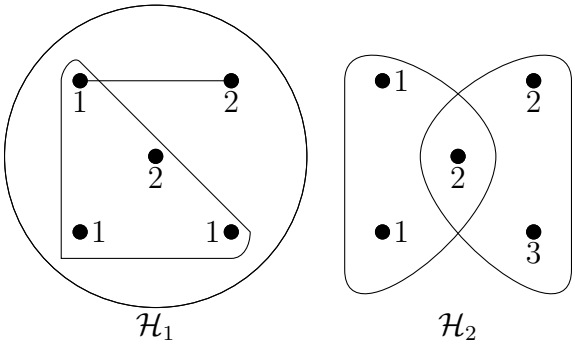

Figure 3: Possible colorings of two hypergraphs on 5 vertices

2.3 Computability

We take a rather informal approach to our notion of computability. While we could be a bit more formal by introducing the concept of a Turing Machine, our purposes are well served by assuming that by "computable", we mean that it can be given by an algorithm. In any case, the definition suffices, since the Church-Turing thesis states that there are multiple computational models that are computationally equivalent to Turing machines and that the notion captured by an algorithm is such a model[4]. Therefore, if we say that any object is computable, we mean that there exists an algorithm that can compute it. Two canonical examples are computable sets and computable functions. A computable set is, as stated in the introduction, a set that can be computed by an algorithm, meaning, that for any input there exists an algorithm that determines if that input is an element of that set. A computable function, on the other hand, is a function in which there exists an algorithm that takes an input from the domain and gives the corresponding output for the range. An algorithm is a computational process that is deterministic and effective. A process is deterministic if given the same input the algorithm will terminate on the same output every time. A process is considered effective if it consists of exact instructions that, if it halts on a given input, will consist of a finite number of steps. Notice that it is not necessary that every input will yield a result, or halts, but that any input that

does give a result must do so at some time.

There exists an effective list of these algorithms which we call $\varphi_0, \varphi_1, \ldots$.
To see why the list is infinite, consider each distinct algorithm that, given any input, returns a natural number. There is such an algorithm for each $n \in \mathbb{N}$. To see why the list is countable, think of the possible number of C++ programs. C++ is a Turing complete programming language [5]; therefore, it can, though perhaps not in practice, implement any algorithm in our effective list. C++ implements these algorithms by creating programs that are composed of a finite number of finite strings of characters that are elements of a finite alphabet. Thus, all valid programs are a subset of all possible finite strings. It is a well-known result [2] that the set of all possible finite strings of a finite alphabet is a countable set. Therefore, the number of C++ programs is infinite and a subset of a countable set, so it must also be a countable set. We may then enumerate them into a list,$\varphi_0, \varphi_1, \ldots$, as we do above. When an algorithm halts on some input x we denote this by $\varphi(x) \downarrow$. In the proofs that follow, we achieve our results by building infinite hypergraphs that can "defeat" every algorithm in the effective list that we just gave. This process is called *diagonalization*.

While not every algorithm gives a coloring, those that do are called computable colorings.

Definition 2.12. A **computable coloring** of a hypergraph is a computable function $f : \mathbb{N} \to \{1, 2, 3, \ldots, k\}$, for some $k \in \mathbb{N}$ which assigns numbers to vertices in which each edge has at least two distinct colors.

Furthermore, we call the coloring by the algorithms which uses the least colors the computable chromatic number.

Definition 2.13. The **computable chromatic number**, denoted by $\chi^C(\mathcal{H})$ is the minimum k colors necessary to give a weak coloring of $\mathcal{H}$ given by some algorithm.

With the preliminaries discussed, we now develop the notion of computable hy-

pergraphs.

3 Computable Hypergraphs

We desire the definition of computable hypergraphs to be a generalization of computable graphs. Bean [1] used the following standard definition for a computable graph.

Definition 3.1. A graph is **computable** if the edge relation R is computable, i.e. there is an algorithm that decides whether or not two vertices are adjacent.

We propose the following definition for computable hypergraphs.

Definition 3.2. A hypergraph is **computable** provided the edge set E is a computable set.

Notice that the proposed definition is a generalization of computable graphs since for graphs knowing whether or not two vertices are adjacent is computationally equivalent to computing the edge set. Also, notice that we cannot use the same definition, because a single hyperedge allows for more than two vertices to be adjacent. Thus, it is possible that we can have multiple distinct hypergraphs with the same adjacency relationship. We see this in Figure 4.

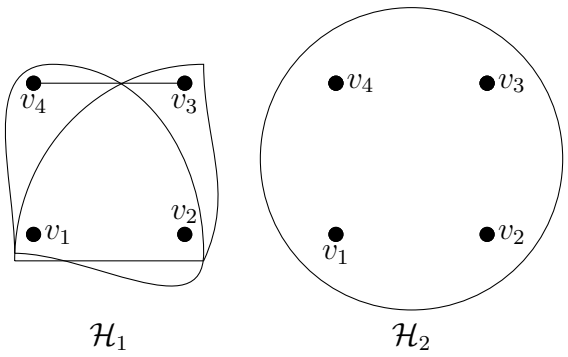

Figure 4: Two distinct hypergraphs with the same adjacency relationship

Recall that Bean showed that there exists a computable graph G such that $\chi(G) = 3$ and $\chi^C(G) = \infty$. However, we also know that for any connected graph G if $\chi(G) = 2$, then $\chi^C(G) = 2$ [3]. Since hypergraphs allow for adjacency relationships between more than two vertices with a single edge, it may not be the case that for any connected hypergraph $\mathcal{H}$ if $\chi(\mathcal{H}) = 2$ then $\chi^C(\mathcal{H}) = 2$. The following theorem and corollary show that, even with the extra condition that every edge must contain exactly three vertices (be 3-uniform), algorithms can be arbitrarily bad at coloring computable hypergraphs.

Theorem 3.3. *For every $n \geq 2$ there exists a connected 3-uniform computable hypergraph $\mathcal{H}$ such that $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) > n$.*

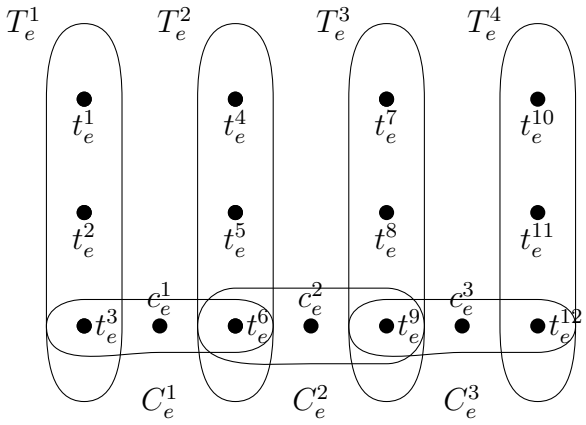

Figure 5: Gadget G_e for $n = 4$

Proof. Let $n \geq 2$. We build the hypergraph $\mathcal{H}$ in stages and we diagonalize against all φ_e by giving each φ_e a gadget G_e. To keep track of the vertices consider the following construction of each G_e.

G_e consist of n copies of the hypergraph K_3^3 which we call $T_e^1, T_e^2, \ldots, T_e^n$ where for $1 \leq i \leq n$ we have $T_e^i = \{t_{i,e}^1, t_{i,e}^2, t_{i,e}^3\}$. These n copies of K_3^3 are connected by $n-1$ hyperedges $C_e^1, C_e^2, \ldots C_e^{n-1}$ such that for any $1 \leq k \leq n-1$ $C_e^k = \{t_{k,e}^3, t_{k+1,e}^3, c_k\}$ where c_k is an additional vertex only belonging to the edge C_k.

We build $\mathcal{H}$ by the following construction. Whenever we add a vertex we take this to be the least natural number not yet given in $\mathcal{H}_s$.

Construction: At stage zero we add G_0 and initially declare all φ_e as not defeated.

At stage $s \geq 1$ we add G_s, a vertex c_s and a hyperedge C_s such that $C_s = \{t^3_{n,s-1}t^3_{1,s}, c_s\}$, so that $\mathcal{H}$ is connected.

For each $e \leq s$ where φ_e is still not defeated, run φ_e on the vertices of G_e for s steps. If $\varphi_{e,s} \downarrow$ for all vertices in G_e, and the n-coloring given by φ_e satisfies our coloring condition, then

1. By the Pigeon Hole Principle, there exists a pair of vertices that are colored the same. Of the possible pairs, choose the pair such that the vertices and colorings are given by the least natural numbers available. Call this pair of vertices p^1_e. For the same reason, there must also exist another distinct pair of vertices p^2_e that are colored the same, but is different than the color of the vertices of p^1_e. We now have two pairs of vertices which are colored differently from each other (there may be more such pairs but this is not guaranteed, so we always identify the pairs by the first vertices that satisfy the condition in the enumeration of the vertices of $\mathcal{H}$).

   We now begin a process in which we will add some vertices and edges to G_e and then wait for φ_e to color these new vertices.

2. Let $k_{e,s}$ equal the number of uniquely colored pairs of vertices that are identified at stage s for gadget $G_{e,s}$ (the state of the gadget e at stage s).

   While $(n - k_{e,s}) \geq 2$ and $G_{e,s}$ has a good n-coloring do the following:

i.) Add $n - k_{e,s}$ copies of K_3^3 to $G_{e,s}$ such that there is no current edge relation with the other vertices. Connect every pair $p_e^1, p_e^2, \ldots p_e^{k_{e,s}}$ with each new vertex in each new K_3^3 copy.

ii.) Continue building $\mathcal{H}$. For each $e \leq s$ where φ_e is still not defeated, run φ_e on the vertices of $G_{e,s}$ for s steps. If $\varphi_{e,s} \downarrow$ for all vertices in $G_{e,s}$, and the coloring given by φ_e satisfies our coloring condition then it follows by the pigeon hole principle that there will be two new unique pairs of vertices. Thus when we repeat i we will add two less vertices and φ_e will have two less colors available to use on the $n - k_{e,s}$ copies of K_3^3 whose vertices are each have an edge of degree three with the listed unique pairs.

3. Once $(n - k_{e,s}) < 2$ add one copy of K_3^3 to $G_{e,s}$ such that there is no edge relation with the other vertices. Notice that $k_{e,s} = (n-1)$. Connect the three new vertices with the $n-1$ pairs of uniquely identified pairs given by the process above.

   This completes the construction.

Now any coloring given by φ_e will fail as all vertices in the last K_3^3 copy must all be colored with the nth color that has yet to be given by the pairs $p_1, p_2, \ldots, p_{n-1}$ and therefore there will be a monochromatic edge.

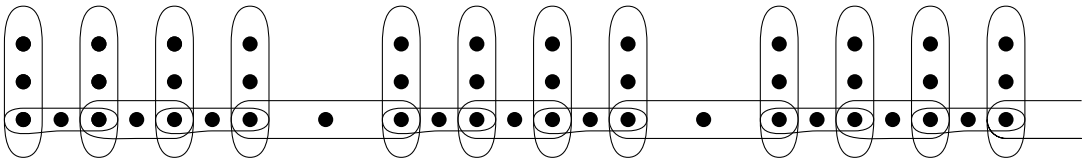

Figure 6: Possible start of $\mathcal{H}$ for $n = 4$

□

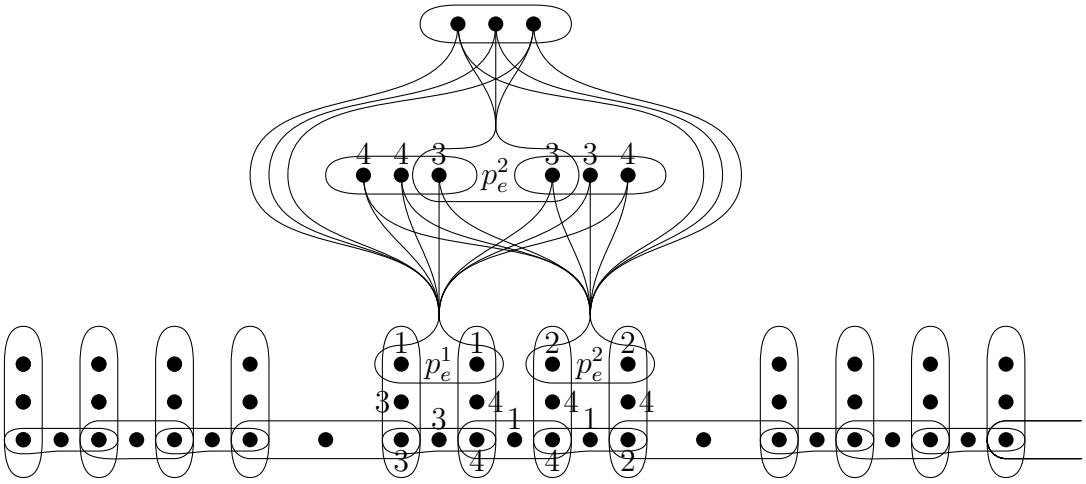

Figure 7: If φ_1 halts (Note: vertices connected to the pairs p_e^1, p_e^2, p_e^3 by an edge represent a single hyperedge containing that vertex and the pair).

Verification: We show that $\mathcal{H}$ is 2-colorable by showing that there exists a 2-coloring for each gadget. We do the following:

i.) For each gadget G_e identify all like-colored pairs $p_e^1, p_e^2, \ldots$. For each pair color the vertex with the least natural number associated with it 1 and the other 2. All edges that perform the function of connecting the additional K_3^3 copies will now satisfy our coloring condition.

ii.) We now color the rest of the vertices in the K_3^3 copies. Let $E_{i,e} \in G_e$ such that $E_{i,e} \neq C_e^k$ for any i, k. If $E_{i,e}$ contains no vertices that were in the listed pairs $p_e^1, p_e^2, \ldots$, then none of the vertices are currently colored so color the least two vertices 1 and the greatest vertex 2. All of these edges will now satisfy the coloring condition.

If $E_{i,e}$ contains only one vertex that was in the listed pairs $p_e^1, p_e^2, \ldots$, then color the vertex that is in that edge whose natural number is closest the opposite color of the given vertex from the pair. Color the remaining vertex 1. Now all of these edges will satisfy our coloring condition.

If $E_{i,e}$ contains two vertices from the listed pairs $p_e^1, p_e^2, \ldots$, then we color the remaining vertex the opposite color of the vertex of the pair with the least natural number associated with it. All of these edges will now satisfy the coloring condition. (Notice that if an edge contains three vertices of from $p_e^1, p_e^2, \ldots$, then it will already have a good coloring as we colored the vertices properly in i).

iii.) All that remains is to color the vertices that are only members of C_e^k and the vertices in the edges that connect gadgets. If the two vertices are adjacent to the given vertex are the same, then we color it the opposite color. If, on the other hand, they are different, then we color the vertex 1. In either case, the coloring condition is satisfied for the remaining edges.

Therefore every edge has now been appropriately colored using 2 colors and so $\chi(\mathcal{H}) = 2$.

Also, since whenever we add an edge, it contains a vertex not yet mentioned, so the hypergraph is computable. By construction $\mathcal{H}$ is connected and 3-uniform. Furthermore, we have given each possible algorithm φ_e a gadget to color and shown that for any specific n it cannot give a n-coloring.

Therefore there exists a connected, 3-uniform computable hypergraph $\mathcal{H}$ such that for every $n \geq 2$ we have $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) > n$

Corollary 3.4. *There exists a connected 3-uniform computable hypergraph $\mathcal{H}$ such that $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) = \infty$*

Proof. We use the previous result and its construction to build the hypergraph $\mathcal{H}$. Each algorithm φ_e will get a sequence of connected gadgets $G_{e,n}$ for every $n \in \mathbb{N}$. The gadget $G_{e,n}$ is the gadget from Theorem 3.3 used for $n + 2$. (We use $n + 2$ because we start at $n = 2$ when building the hypergraph). Therefore $G_{e,3}$ is the gadget used in Theorem 3.3 when $n = 5$. We use the same labeling of vertices as in Theorem 3.3 with the exception that we replace each suffix e with e, n. To make sure that the

sequence of gadgets are connected whenever we add a gadget $G_{e,n}$ we add a vertex $v^1_{e,n}$ such that it is not an element of any other current edges, and draw the hyperedge $E_{e,n,v^1} = \{t^3_{e,n}, t^3_{e,0}, v_{e,n}\}$. Similarly, to make sure that $\mathcal{H}$ is connected whenever $G_{e,0}$ is added for a given $e \geq 1$ we create the vertex $v^2_{e,0}$, with no edge adjacency, and the hyperedge $E_{e,n,v^2} = \{t^3_{e-1,0}, t^3_{e,0}, v^2_{e,0}\}$ (Again, notice that these hyperedges and vertices won't effect the coloring given by φ_e for any e since we may choose how these vertices are colored since either the two adjacent vertices are colored the same or differently. Therefore we can always color the vertex either 1 or 2).

As in the last construction, whenever we add a vertex, we take this to be the least natural number not yet given in $\mathcal{H}_s$. Consider the following construction of $\mathcal{H}$.

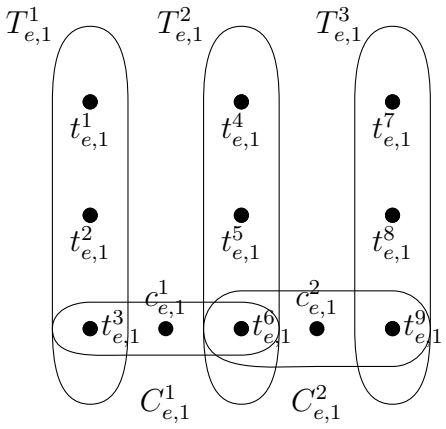

Figure 8: Gadget $G_{e,1}$ added at stage $e+1$

Construction: At stage zero we add $G_{0,0}$ and initially declare all φ_e as not defeated.

At stage $s \geq 1$ we build the gadgets $G_{e,s-e}$ for each e and their respective vertices $v_{e,n}$ and connecting edges E_{e,n,v^1}, and E_{e,n,v^2} if $s - e = 0$.

For each pair $\langle e, n\rangle$ where $e \leq s$ and $n \geq s - e$ and where φ_e is still not defeated, run φ_e on the vertices on every $G_{e,n}$ for s steps. If $\varphi_{e,s} \downarrow$ for all vertices in $G_{e,n}$, and the coloring given by φ_e satisfies our coloring condition, then

1. Proceed exactly as in Theorem 3.3. That is, identify vertices into pairs that are colored the same and add the given edges and repeat until every φ_e fails on each $G_{e,n}$.

This completes the construction.

Verification: Clearly the hypergraph is 3-uniform and connected by construction. Also, it must be 2 colorable since for all $n \geq 2$ in Theorem 3.3 $\mathcal{H}$ is 2 colorable. Furthermore, $\chi^C(\mathcal{H}) = \infty$ because for every $n \in \mathbb{N}$ φ_e will fail to color each the components associated with $G_{e,n}$.

Therefore there exists a connected 3-uniform computable hypergraph $\mathcal{H}$ such that $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) = \infty$

□

This result tells us hypergraph colorings are, in some sense, more complex than graph colorings. However, computable graphs can be arbitrarily difficult for algorithms to color, too. In the same spirit as Bean, we now seek to develop a stronger notion of computability for hypergraphs: that is, highly computable hypergraphs.

4 Highly Computable Hypergraphs

As with computable graphs, we seek to generalize the definition of highly computable graphs Bean [1] used. The definition is as follows.

Definition 4.1. A graph is *highly computable* if there is a computable function $f : \mathbb{N} \to \{\text{sequence of numbers}\}$ such that $f(i) = \langle i_1, i_2, \ldots, i_n\rangle$ means vertex i is adjacent to exactly the vertices $i_1, i_2, \ldots, i_n$.

We propose the following definition for highly computable hypergraphs.

Definition 4.2. A hypergraph is *highly computable* if there is a computable function $f : \mathbb{N} \to \{\text{sequence of sets}\}$ such that $f(i) = \langle e_{(i,1)}, e_{(i,2)}, \ldots, e_{(i,n)} \rangle$ where $e_{(i,1)}, e_{(i,2)}, \ldots, e_{(i,n)}$ are exactly the edges in which $i \in e_{(i,j)}$, for each $j \in \{1, 2, \ldots . n\}$.

Observe that 4.2 is a generalization of 4.1 since for graphs an adjacency relationship is computationally equivalent to describing an edge set. Also, notice that this generalization is necessary for the same reason described in the previous section. Since hypergraphs allow for adjacency relationships between more than two vertices with a single edge, knowing the vertices that are adjacent to each other only describes possible edges, not unique edges.(Which is necessary for the definition to align with our informal definition of it being given by an algorithm, i.e., to be deterministic.) All connected graphs G, in particular highly computable graphs, have the property that $\chi(G) = 2$ also have the property that $\chi^C(G) = 2$ [3]. The following results shows that this property is not true for connected hypergraphs $\mathcal{H}$ which are 2-colorable.

Theorem 4.3. *There exists a connected highly computable hypergraph $\mathcal{H}$ such that $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) = 3$*

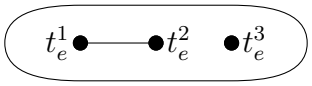

Figure 9: Gadget for $\mathcal{H}$

Proof. Consider the hypergraph K_3^3. We are going to build $\mathcal{H}$ in stages using infinitely many copies of K_3^3 to diagonalize against all φ_e. Thus $\mathcal{H} = \bigcup_{s \in \mathbb{N}} \mathcal{H}_s$. Each vertex of K_3^3 will be connected to additional vertices such that these vertices will form paths emanating from the vertices in K_3^3. These paths will receive an additional edge and vertex at each stage s.

To keep track of vertices and edges we use T_i for the ith copy of K_3^3 and call its vertices $t_i^1, t_i^2,$ and t_i^3.

Construction: At stage zero we add T_0 and declare all φ_e as undefeated.
At stage $s \geq 1$, we add T_s and a hyperedge connecting T_{s-1} and T_s using the vertices $t^3_{s-1}, t^1_s,$ and v_s where v_s is a vertex which belongs only to the connecting edge. We also add the vertices $t^s_i, t^{2s}_i, t^{3s}_i$ for all $i < s$ which connects to the previous three vertices added at each stage via a standard graph edge and such that they connect to the vertex that shares the same multiple $(1, 2, 3)$.
For each $e \leq s$ where φ_e is still not defeated, run φ_e on the vertices of T_e for s steps. If $\varphi_{e,s} \downarrow$ for all $v \in T_e$, and φ_e satisfied our coloring condition using 2 colors, then

1. Since φ_e gave a 2-coloring to vertices t^1_e, t^2_e, t^3_e, by the pigeon hole principle two vertices must be colored the same. Let these two vertices be denoted by v^1_e, v^2_e and let the vertex of a different color be denoted by v^*_e.

2. Since the coloring was allowed only two colors; it follows that once a vertex on a path is colored the coloring of the entire path is determined. Therefore once φ_e halts we can add 6 more vertices to T_e such that each path gets two more vertices (and their respective edges).

3. Consider the path emanating from the v^1_e. Using the vertex that is at the end of the path draw an edge such that it connects with the last vertex on the path emanating from v^2_e.

Figure 10: Possible start for $\mathcal{H}$

Verification: Notice that any 2-coloring of the remaining vertices will give a bad coloring, since the two vertices that we connected will be the same color and cannot be the other color, so it must use a third color and is therefore not 2-colorable. Also, notice that if we change the coloring by changing the parity of the path emanating

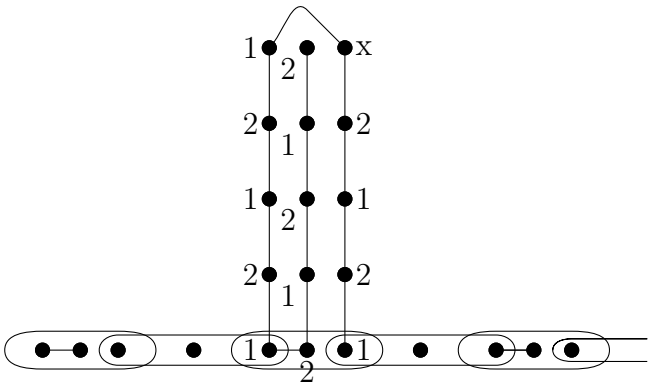

Figure 11: If φ_1 halts

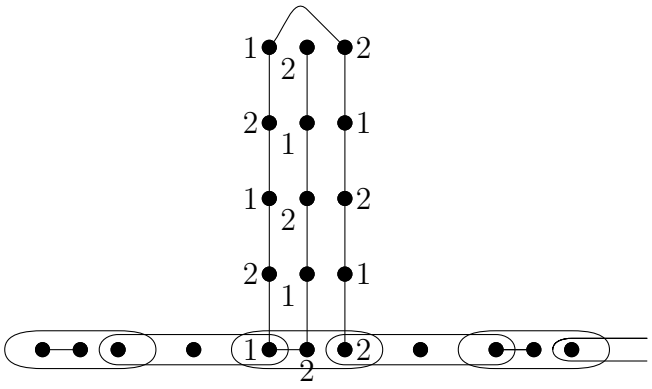

Figure 12: Observe that $\mathcal{H}$ is still 2-colorable

from v_e^1, then it will give a 2-coloring. Observe that the hypergraph is highly computable as well since whenever a vertex is enumerated we say exactly which edges and vertices are adjacent to it. To see that $\chi^C(\mathcal{H}) = 3$, the algorithm could proceed as if it were giving a 2-coloring, but use the third color available to it when it comes to the vertex is adjacent to the two paths and thus giving a 3-coloring.

Therefore there exists a highly computable hypergraph $\mathcal{H}$ such that $\chi(\mathcal{H}) = 2$ and $\chi^C(\mathcal{H}) = 3$. □

We have shown that there exists a highly computable hypergraph $\mathcal{H}$ that is more difficult to color than its highly computable graph counterpart. However, the following result shows that these computable colorings cannot be arbitrarily bad. We establish the same bound given by Bean in[1], which says that for any highly com-

putable graph G where $\chi(G) = k$, it follows that $\chi^C(G) \leq 2k$.

Theorem 4.4. *Every k-colorable highly computable hypergraph $\mathcal{H}$ is computably $2k$-colorable.*

The following proof is a modification of Bean's proof which provides a bound on the computable chromatic number for highly computable graphs.[1]

Proof. Let $\mathcal{H}_0$ be edge 0 and $\mathcal{H}_{n+1}$ be the subgraph of $\mathcal{H}$ which includes all edges that are adjacent to the vertices of $\mathcal{H}_n$ and those vertices that are elements of those edges. Let $\overline{\mathcal{H}}_n$=$\mathcal{H}_n - \mathcal{H}_{n-1}$. Since $\mathcal{H}$ is highly computable we can find $\overline{\mathcal{H}}_n$ effectively. Furthermore, since $\mathcal{H}$ is $k-$colorable and $\overline{\mathcal{H}}_n$ is finite we can effectively color $\overline{\mathcal{H}}_n$. Let us now separate our colors in two evenly divided sets K_1, K_2 such that $K_1 \cap K_2 = \emptyset$ and $|K_1 \cup K_2| = 2k$ (thus each set has k unique colors). We now do the following: Color H_0 with colors from K_1, $\overline{\mathcal{H}}_1$ with colors from K_2, $\overline{\mathcal{H}}_2$ with colors from K_1, and so on, so that $\overline{\mathcal{H}}_{2n}$ is colored with colors from K_1 and $\overline{\mathcal{H}}_{2n+1}$ with colors from K_2. We see that this gives a computable $2k$-coloring since, by definition, no vertex in $\mathcal{H}_n$ is adjacent to a vertex in $\mathcal{H}_{n+2}$. □

5 Conclusion

We have generalized the notions of computable and highly computable graphs to hypergraphs to show that there exists a connected 3-uniform computable hypergraph that is 2-colorable, but computably uncolorable. We also proved that, while highly computable graphs have bounded computable colorings, there exists a connected highly computable hypergraph that is 2-colorable, yet any computable coloring requires three colors. In either case, there is no computable or highly computable graph counterpart. These results offer a modest start to developing computable hypergraph theory, and there remain many fundamental open questions. Considering alternative

colorings like strong colorings, which requires every adjacent vertex to be colored differently, or mixed colorings, which imposes coloring conditions on different families of subsets of vertices, would be a natural direction to take. Also, exploring alternative generalizations for computable and highly computable graphs could lead to interesting, and entirely different, results. In general, any question that determines whether a computable graph theory theorem holds for computable hypergraph theory seems to be worthwhile.

References

[1] Dwight R Bean, *Effective coloration*, The Journal of Symbolic Logic **41** (1976), no. 2, 469–480.

[2] H-D Ebbinghaus, Jörg Flum, and Wolfgang Thomas, *Mathematical logic*, Springer Science & Business Media, 2013.

[3] William Gasarch, *A survey of recursive combinatorics*, Studies in Logic and the Foundations of Mathematics, vol. 139, Elsevier, 1998, pp. 1041–1176.

[4] Oscar Levin and Taylor McMillan, *Computing planarity in computable planar graphs*, Graphs and Combinatorics **32** (2016), no. 6, 2525–2539.

[5] Michael Sipser, *Introduction to the theory of computation*, vol. 2, Thomson Course Technology Boston, 2006.

[6] Vitaly I Voloshin, *Introduction to graph and hypergraph theory*, Nova Science Publ., 2009.