Counting Monochromatic Components in 2-Player Graph Burning

Lewis Dyer (2299195)

February 10, 2022

ABSTRACT

We introduce 2-player graph burning, an extension of the graph burning process to contagion with competition. We develop multiple results on the number of monochromatic components on graph colourings resulting from 2-player graph burning.

1. INTRODUCTION

INTRO GOES HERE

2. BACKGROUND

BACKGROUND GOES HERE, PROBABLY DESCRIBING GRAPH BURNING

3. 2-PLAYER GRAPH BURNING

3.1 Defining 2-Player Graph Burning

First, we formalise the process of 2-Player Graph Burning on a graph G. Throughout, we presume that G is a finite, simple, undirected graph.

DEFINITION 3.1. *2-Player Graph Burning (or 2-GB for short) is a discrete-time graph process for two players. Each vertex is assigned one of 4 colours:*

1. White *vertices have not been burned by either player yet.*
2. Red *vertices have been burned by player 1.*
3. Blue *vertices have been burned by player 2.*
4. Green *vertices have been burned by* both *players.*

At time $t = 0$, all vertices are initially white. Each round consists of two main steps. First, both players simultaneously choose a white vertex, burning the vertex into that player's respective colour. Secondly, all non-white vertices spread their colour onto adjacent white vertices, according to the following rules:

- *If a white vertex is burned by multiple vertices, all with the same colour, the white vertex is also burned with the same colour.*
- *If a white vertex is burned by both red and blue vertices, the white vertex will become green.*
- *If a white vertex is burned by either red or blue vertices (but not both), and also by green vertices, the white vertex will become red or blue, respectively.*

In particular, these rules ensure that this process is symmetrical, so the choice of player is unimportant.

Our main interest in this problem will be to count the maximum number of monochromatic components that can appear in any colouring of G resulting from an instance of the 2-GB process. Our definition of "monochromatic components" differs slightly from most other uses of the term, so we define it separately here to account for these differences, and use slightly different terminology to emphasise these differences:

DEFINITION 3.2. *Given a graph G, with an assignment of colours resulting from the 2-GB process, a* red cluster *is a connected component in the subgraph induced by all red and green vertices in G.*

We also introduce the restriction that, in the 2-GB process, both players may only choose to burn the same vertex if there are no other remaining white vertices to burn.

We note that only red clusters are defined here - blue clusters could of course be defined similarly, though since the 2-GB process is symmetrical we need only define red clusters.

In addition, the latter restriction on 2-GB is relatively minor, and mainly aims to remove some degenerate cases when counting red clusters, without needing to explicitly define exceptions for later results.

3.2 Burning Sequences

We now introduce a compact notation for describing a particular instance of 2-player graph burning.

DEFINITION 3.3. *Given a graph G, a* burning sequence *of length n, $B \in (V(G) \times V(G))^n$, is a sequence of tuples, $(r_1, b_1), (r_2, b_2), \ldots, (r_n, b_n)$, such that player 1 burns vertex r_t and player 2 burns vertex b_t at time t.*

This sequence describes possible choices for the first n rounds of the 2-GB process, but some further care is required to ensure this sequence describes valid choices for this process:

DEFINITION 3.4. *Given a burning sequence B, B is said to be* valid *if:*

1. *For every vertex that appears in the i^{th} term of B, say v_i, for every vertex in the j^{th} term of B such that $j < i$, say v_j the distance between v_i and v_j is at most $(i - j)$.*
2. *After the burning sequence described by B is completed on G, every vertex is G is non-white.*

From this definition, an immediate corollary is that the original graph burning problem is a subset of 2-player graph burning:

COROLLARY 3.1. *Given a valid burning sequence for the original graph burning problem, say $(v_1, v_2, \ldots, v_n)$, the burning sequence $(v_1, v_1), (v_2, v_2), \ldots, (v_n, v_n)$ describes a valid burning sequence in 2-player graph burning.*

3.3 Bounding the length of burning sequences

A useful question in the 2-player graph burning problem is to consider the maximum number of rounds required for this process to terminate, providing an upper bound for the number of valid burning sequences for a graph G. The following result gives a useful bound for connected graphs:

THEOREM 3.1. *For a connected graph G with n vertices, the maximum number of rounds required for the 2-GB process to terminate is $\lceil \frac{n}{3} \rceil$.*

PROOF. Given a graph G, consider U, the subgraph induced by all white vertices in G. We aim to show that, if U contains at least 3 vertices, each round of 2-GB removes at least 3 vertices from U.

If U contains at least 3 vertices, then we choose 2 vertices to initially burn, say v_1 and v_2, and we need to show that at least one other distinct vertex is adjacent to v_1 or v_2. Considering the possible degrees of v_1 and v_2 in U, there are two potentially problematic cases, while the other cases are straightforward.

If v_1 and v_2 are both adjacent and have degree 1, then no other vertices are adjacent to them. However, since U has at least 3 vertices, some other vertex v_3 must be in U, but not connected to v_1 or v_2. However, since G is connected, v_3 must be adjacent to some non-white vertex in G, so v_3 must also be removed this round. A similar argument holds when v_1 and v_2 are both degree 0.

Hence, while at least 3 vertices remain unburned, at least 3 vertices are burned each round, and if there are ever less than 3 vertices remaining they will all be burned in 1 round, so the maximum number of rounds to burn all vertices is $\lceil \frac{n}{3} \rceil$.

□

4. COLOURING CATERPILLAR GRAPHS

4.1 Monochromatic components on paths

Our first key observation when counting red clusters is that at most one cluster can be added per round. This is because red vertices can be added in two different ways. If red vertices are added via spreading from adjacent vertices, no new clusters are created in this way (though existing clusters may be merged together, *reducing* the number of red clusters on the graph). So red clusters can only be created by player 1 choosing vertices to burn, but only one vertex can be burned by player 1 each turn. Coupled with Theorem 3.1, this gives a useful bound for the number of clusters on connected graphs:

COROLLARY 4.1. *Let G be a connected graph on n vertices. Then the maximum number of red clusters on G is $\lceil \frac{n}{3} \rceil$, and this maximum is attainable when G is a path on n vertices.*

PROOF. Firstly, since each round can introduce at most 1 red cluster, and since the maximum number of rounds on G is $\lceil \frac{n}{3} \rceil$ by Theorem 3.1, the maximum number of red clusters on G is $\lceil \frac{n}{3} \rceil$.

Now let G be the path on n vertices. Listing the vertices of G in order from $v_1, v_2, \ldots, v_n$, the burning sequence $(v_1, v_2), (v_4, v_5), \ldots, (v_{n-2}, v_{n-1})$ contains $\lceil \frac{n}{3} \rceil$ red clusters.

□

4.2 Caterpillar graphs

We now consider a natural extension of paths, and provide a method for counting clusters on these graphs.

DEFINITION 4.1. *A* caterpillar graph *is a graph obtained by taking a path, known as the* spine *of a caterpillar graph, and adding any number of vertices of degree 1 (known as leaves), that are adjacent to a vertex on the original spine.*

In order to count clusters on caterpillar graphs, it will be useful to introduce more compact notation for describing caterpillar graphs:

DEFINITION 4.2. *Given a caterpillar graph built on a spine with n vertices, its* caterpillar string *is a sequence $C \in \mathbb{N}^n$ of the form $(c_1, \ldots, c_n)$, where c_i denotes the number of leaves adjacent to the i^{th} spine vertex. For instance, the caterpillar string $(1, 0, 2, 3)$ represents the caterpillar graph shown in Figure [INSERT A PICTURE HERE].*

Clearly, there are a countably infinite number of caterpillar strings of length n. However, for the purposes of counting clusters, the following lemma shows that we can reduce the set of caterpillar strings to a finite subset of strings of a given form:

LEMMA 4.1. *Given a caterpillar string C, the maximum number of red clusters in C is equal to the maximum number of red clusters in the caterpillar string given by*

$$C'_i = \begin{cases} 0 & C_i = 0 \\ 1 & C_i \geq 1 \end{cases}$$

Moreover, given such a caterpillar string C', if the first or last element in C' is 1, this can be removed and two zeros can be appended to the start or end of the sequence respectively, giving an equivalent caterpillar string, say C''. After performing these two operations, we say that C'' is a reduced caterpillar string.

PROOF. For the first reduction, suppose a given spine vertex v_i has at least 2 adjacent leaves. Pick two of these leaves without loss of generality, calling them l_1 and l_2, and consider the subgraph induced by v_i, l_1 and l_2.

Suppose that this subgraph contains 2 red clusters. Then both leaves must be coloured red. However, this is only possible when v_i is also red. Since both leaf vertices are not adjacent, 2 rounds are required to colour them both red, and hence the first one coloured red will spread to v_i, colouring it red. And if v_i was already coloured blue, it would spread its colour to at least one of l_1 and l_2, so by contradiction this subgraph can contain at most 1 red cluster. And since the leaves were chosen without any loss of generality, either no leaves are red, all leaves are red along with their common spine vertex, or exactly one leaf is red, so all but one leaf can be removed without changing the overall number of clusters.

For the second reduction, if one of the endpoints has a leaf, the leaf can be treated as a spine vertex rather than a leaf vertex, removing a leaf from the existing endpoint and becoming the new endpoint instead. □

Therefore, the set of reduced caterpillar strings of length n is given by the set of bitstrings of length n beginning and ending with a 0, hence there are 2^{n-2} reduced caterpillar strings of length n. In practice this may be reduced even further, since reversing the reduced caterpillar string gives another string that may be different, but which always represents an isomorphic caterpillar graph.

4.3 Monochromatic components on caterpillar graphs

As discussed in Lemma 4.1, we can transform any caterpillar graph into a reduced caterpillar graph while keeping the same number of maximum red clusters on the graph. We now present the following algorithm to count the maximum number of red clusters on any reduced caterpillar graph:

Given a reduced caterpillar string, partition this string into pieces of length 3, leaving any remainder characters in the last piece which may be shorter than 3 characters, and treat each of these pieces as a (not necessarily reduced) caterpillar string.

For each of these pieces, we can easily compute the maximum number of red clusters which originate in this piece, given some starting and ending constraints. These constraints are comprised of 4 distinct cases:

- In case RW, when colouring of this piece begins, the first spine vertex is white, but the previous adjacent spine vertex, which if it exists is not part of this piece, is coloured red.
- Case BW is equivalent to case RW, except the previous spine vertex is blue instead of red. This is also the case applied for the very first piece, in order to ensure that colouring the first spine vertex red correctly counts the start of a new red cluster.
- Case R is where the first spine vertex is already coloured red, since the colouring of the previous piece is not fully contained within that piece.
- Case B is analogous to case R.

These cases are defined for the start of a piece, but the ending constraints are analogous, and crucially correspond to the same starting constraint of the next piece.

Since these pieces are small, each of these computations is small and relatively simple to perform. For each piece, this then produces a weighted directed bipartite graph G_P, with vertex set $\{RW_{start}, BW_{start}, R_{start}, B_{start}, RW_{end}, BW_{end}, R_{end}, B_{end}\}$, such that an edge exists from X_{start} to Y_{end} if there exists a colouring of piece P with starting constraint X and ending constraint P, with weight equal to the number of clusters originating in P.

Once all of these bipartite graphs are computed, the ending vertices of each piece can be merged with the starting vertices of the next piece, along with adding in a common source and sink vertex, removing every case bar case BW from the first piece, generating a directed acyclic multipartite graph G. Then, the maximum number of red clusters on the reduced caterpillar graph is equal to the length of the longest spine in G.

For future reference, we shall summarise this algorithm below:

Definition 4.3. *Given a reduced caterpillar graph G, and assuming that each piece of length at most 3 has been pre-computed, a colouring for G can be attained through the following algorithm:*

Partition the reduced caterpillar graph into as many connected subgraphs containing at most 3 path vertices as possible, leaving any remaining vertices in a piece at the end which may be shorter than length 3.

For each piece, define a bipartite graph with vertices

$$\{RW_s, BW_s, R_s, B_s, RW_e, BW_e, R_e, B_e\}$$

such that an edge exists from X_s to Y_e with weight w if the maximum number of red colourings originating this piece on a valid colouring with starting contraint X and ending constraint Y is w.

After defining these graphs, define a multipartite graph M, merging the ending constraints of each piece and the corresponding starting constraint of the subsequent piece, along with adding source and sink vertices. Then the maximum number of red clusters on the reduced caterpillar graph is equal to the length of the longest path in M.

Assuming that the bipartite graphs for each possible piece are pre-computed beforehand, producing the directed acyclic graph G takes $O(p)$ time where p is the number of pieces to combine, since each piece requires combining two constant sets of vertices together, along with constant time operations to add source and sink vertices. Since each piece adds at most 4 additional vertices to G, and at most 16 additional edges, the number of vertices and edges in G are both $O(p)$. And finding the longest path in a directed acyclic graph with V vertices and E edges takes $O(V+E)$ time, this algorithm takes $O(p)$ time overall. But since each piece is of length at most 3, the number of pieces is linear in the length of the caterpillar, so this algorithm takes $O(n)$ time, being linear in the length of the reduced caterpillar string.

In order to justify the correctness of this method, namely that it provides the exact maximum number of red clusters attainable on G, we must first justifying colouring each piece in sequence.

Lemma 4.2. *For a (not necessarily reduced) caterpillar graph G split into a series of pieces of length at most 3 as in the previous algorithm, say $P_1, \dots, P_n$, the optimal colouring in terms of maximising red clusters is attained by colouring P_1, then P_2, and so on up to P_n.*

Proof. This proof proceeds by induction on the number of pieces in G.

If G has just one piece, then clearly colouring this piece will attain the optimal colouring, so the base case trivially holds.

Now suppose the lemma holds for any (not necessarily reduced) caterpillar graph consisting of at most $n-1$ pieces. Then G, a caterpillar graph with n pieces, is made up of the concatenation of two caterpillar graphs - one graph with $n-1$ pieces followed by one piece.

If either end piece is fully coloured first, the lemma holds from the inductive hypothesis. If a different piece is fully covered first, then G is split into two smaller graphs, each with length less than $n-1$ pieces. If an optimal colouring for one of these graphs is attained using the inductive hypothesis, then the colouring will spread to the other piece, so it is not possible to colour each of these pieces sequentially - and hence, this colouring cannot be better than the optimal colouring, so the lemma holds in this case.

Now suppose the first piece coloured is not fully covered, so another piece starts being coloured before the first piece is finished. Then the possible colourings on this first piece are a subset of all possible colourings, so the number of red clusters on this first piece cannot be any better than optimal. Then applying the inductive hypothesis to the remaining pieces means that the result holds.

□

Note that this lemma holds for caterpillar graphs even if they are not fully reduced - in particular, their strings need not start or end in a 0. This allows induction to be applied, since concatenating non-reduced caterpillar graphs can still lead to a reduced caterpillar graph.

The algorithm being exact follows from this lemma, since the multipartite graph constructed during the algorithm describes all possible ways to colour the caterpillar graph sequentially, and since the longest path is chosen the most optimal way to colour the whole graph is chosen, even if this means deviating from the greedy approach.

4.4 Graph diameter in 2-player graph burning

A key heuristic when counting the maximal number of monochromatic components for a particular graph G is the *diameter* of G:

Definition 4.4. *Given a graph G with vertex set $V(G)$, the* diameter *of G is given by*

$$D(G) = \max_{u \in V(G), v \in V(G)} d(u, v)$$

where $d(u, v)$ is the length of the shortest path between u and v, and if no such path exists we say that $d(u, v) = \infty$.

In the context of 2-player graph burning, this means that any instance of 2-player graph burning on G will terminate within $D(G)$ rounds, when G is connected. The following example suggests that graphs with low diameter tend to admit fewer clusters than high diameter graphs:

Lemma 4.3. *Let G be a cycle on n vertices. Then the maximum number of red clusters on G is $\lceil \frac{n}{4} \rceil$, and this bound is tight.*

Proof. Firstly, labelling the vertices of G as $v_1, \ldots, v_n$ in order, keeping in mind that v_1 is adjacent to v_n, the burning sequence $(v_1, v_n), (v_{n-2}, v_3), (v_5, v_{n-4}), \ldots$ attains $\lceil \frac{n}{4} \rceil$ red clusters, since at each round except possibly the last, 4 vertices are burned and 1 new red cluster is started.

Secondly, since every vertex has degree 2, at least 4 vertices must be burned every round, with the proof of this following a very similar structure to Corollary 4.1. And since at most 1 cluster can be created per round, the maximum number of red clusters on G is at most $\lceil \frac{n}{4} \rceil$. □

In conjunction with Corollary 4.1, the previous lemma presents a surprising fact: Since the maximum number of clusters on a path is $\lceil \frac{n}{3} \rceil$, and $\lceil \frac{n}{4} \rceil$ on a cycle, adding a single edge can reduce the maximum possible number of clusters on a graph by an arbitrary amount. This suggests that the diameter of a graph is important.

5. COLOURING TREES

Compared to our previous work on paths and caterpillar trees, trees present some unique challenges when counting monochromatic components. In particular, our strategies for colouring paths and caterpillar trees generally rely on being able to limit the spread of vertices, and proceed through the graph in a clear sequence. However, in general trees do not possess a convenient start or end point, so such a sequence is not so clear to find. As a result, rather than the exact results in previous sections, we focus on approximate results for trees, improving on our existing upper bounds. Our first aim is to provide some sort of sequence to a tree, and we start by using that all trees contain a caterpillar graph:

Definition 5.1. *Given a tree T, the* maximal caterpillar *of T, denoted C_T, is the subgraph induced by all vertices in the longest path in T, along with all vertices which are distance 1 from said path. If there are multiple longest paths in T, the maximal caterpillar of T is induced by the path such that the number of vertices in the caterpillar is maximised.*

Once this caterpillar has been obtained, this can be coloured to attain an initial bound on red clusters for T:

Corollary 5.1. *Given a tree T, the maximum number of red clusters on T is at least as many as the maximum number of red clusters on C_T.*

Proof. Consider the colouring on C_T generated by our previous procedure - we claim this colouring can also be applied when C_T is a subgraph of T. Every vertex in $T \setminus C_T$ is adjacent to at most one vertex in C_T, since being adjacent to more than one vertex would result in a cycle. Therefore, these vertices are only coloured when their adjacent vertex in C_T is already coloured, so the colouring of C_T is unaffected by these vertices. □

A natural suggestion is that the tightness of this bound depends on how many vertices of T are included in the maximal caterpillar, which is discussed further in the following lemma:

Lemma 5.1. *For any tree T, denote the* caterpillar ratio *of T as the number of vertices in M_T divided by the number of vertices in T. This ratio can be arbitrarily close to zero.*

Proof. Let B_h be the perfect binary tree with height h. Then the longest path on B_h is obtained by travelling between a leaf node in the left subtree of the root and a leaf node in the right subtree of the root, and this path has length $2h+1$. This path has $2(h-1)$ internal nodes, and each internal node has 2 children, with one child being included in the longest path and the other child being included in the maximum caterpillar, so the number of vertices in M_{B_h} is equal to $4h - 1$.

However, the number of vertices in B_h is $2^{h+1} - 1$, so the caterpillar ratio of B_h is $\frac{4h-1}{2^{h+1}-1}$, which tends to 0 as h tends to ∞. □

This shows that, while colouring the maximal caterpillar gives a lower bound on the maximum number of red clusters in a tree G, an arbitrarily large proportion of the graph is not contained within the maximal caterpillar. Hence, colouring the remainder of $G \setminus C_T$ will be key to obtaining a tighter lower bound on the maximum number of red clusters in G.

We first remark that $G \setminus C_T$ must be disconnected, since otherwise G would contain a cycle. In particular, $G \setminus C_T$ is comprised of a set of connected components that are all trees, so we define $G \setminus C_T := F$ to denote that this subgraph is a forest. Our next aim will be to understand the structure of F, and how this structure can be exploited to improve our bounds on the maximum number of red clusters in T.

5.1 Colouring forests induced by a maximal caterpillar

To begin, each tree in F is adjacent to exactly one burned vertex - if it were adjacent to two or more burned vertices, this would define a cycle. We now use this fact to define some additional characteristics of each tree in F:

Definition 5.2. *Given a tree T in F, a subgraph of a graph G induced by a maximal caterpillar C_G, the* root *of a tree, denoted r_T, is the vertex which is adjacent to a vertex on C_G. The root is* red-adjacent *if this vertex is red, with an analogous definition for a* blue-adjacent *root. In addition, the* size *of T is given by:*

$$s(T) = 1 + \max_{v \in V(T)} d(r_T, v)$$

,

From these definitions, it is apparent that, without choosing any vertices in T to burn, any tree T will be fully burned in $s(T)$ rounds. As a corollary to this, the maximum number of clusters originating in T is at most $s(T)$. This provides an initial upper bound on the maximum number of red clusters in the graph G:

Corollary 5.2. *Consider a partial colouring of G as produced by the above procedure, and the forest of unburned vertices F comprised of a set of trees $\mathcal{T}$. Then the maximum number of red clusters on F is at most*

$$\max_{T \in \mathcal{T}} s(T)$$

Since this bound only considers the maximum size of any tree, this can allow for significant simplification when considering many trees in a forest, particularly for small size. For instance, if every tree has size 1, then every tree will be fully burned after 1 round. Therefore at most 1 cluster can be created, so as long as at least one tree has a blue-adjacent root.

Since the caterpillar in G was chosen to be maximal, this also imposes additional constraints on the size of trees in F. This constraint relies on the underlying caterpillar providing a sequential structure to the tree, as described in the following definition:

Definition 5.3. *Given a tree G with a maximal caterpillar C_G, denoting the endpoints of the caterpillar as vertices s and e, the* caterpillar distance *of any leaf vertex v on C_G, denoted $cd(v)$, is equal to the length of the shortest path from v to an endpoint of the spine of C_G, minus 2.*

We mark that subtracting 2 from the caterpillar distance is necessary in order to removepOnce this notion of distance to a caterpillar endpoint is defined, the maximal nature of this caterpillar implies constraints on the size of trees in the resulting forest $\mathcal{F}$:

Lemma 5.2. *For each tree T in a forest $\mathcal{F}$ induced by the maximal caterpillar C_G, the rooted distance of T is at most $cd(r_T)$.*

If this constraint does not hold for some tree T, then the longest path in the tree G would travel through T, so our caterpillar C_G would not be maximal.

An important subtlety in this lemma is that the caterpillar distance of a vertex is only defined for leaf vertices on C_G. If there existed a tree in $G \setminus C_G$, rooted to some spine vertex v in C_G, then any vertices in the tree adjacent to v could be added to C_G as leaf vertices, contradicting that C_G is maximal, so any such tree must be rooted to a leaf vertex.

Combining this lemma with Corollary 5.2, along with the maximal distance from a spine vertex to a caterpillar endpoint, gives another key bound for the number of clusters in $\mathcal{F}$:

Theorem 5.1. *Consider a partial colouring of G as produced by the above procedure, with a maximal caterpillar of length L (hence G has diameter L), and the forest of unburned vertices F comprised of a set of trees $\mathcal{T}$. Then the maximum number of red clusters on F is at most $\lfloor \frac{L}{2} - 1 \rfloor$.*

The main significance of this result is that the maximum number of clusters in a tree G is proportional to the diameter of G, regardless of how many vertices are outside of the maximal caterpillar C_G. For example, the problem suggested in Lemma 5.1, where arbitrarily many vertices may be outside of the maximal caterpillar, is mostly resolved - while the number of vertices outside of the caterpillar is unlimited, their distribution is constrained, in particular the size of the resulting trees which are formed, meaning that the number of clusters in these remaining vertices can be bounded.

Furthermore, since the maximum number of red clusters on the maximal caterpillar C_M is known, the following result is an easy corollary from Theorem 5.1:

Corollary 5.3. *Given a tree G with diameter L, which is coloured by first colouring its maximal caterpillar C_G, the maximum number of red clusters on G is at most $\frac{7(L-1)}{6}$.*

Proof. As previously discussed, a caterpillar of length n has at most $\frac{2n-2}{3}$ red clusters, since any such caterpillar can be reduced to a caterpillar with at most $2n - 2$ vertices by Lemma 4.1. And Theorem 5.1 shows that the number of red clusters on the rest of the graph outside of the maximal caterpillar of length L has at most $\lfloor \frac{L}{2} - 1 \rfloor$ red clusters, and hence at most $\frac{L}{2} - 1$ red clusters, so summing these results together proves the corollary. $\square$

This corollary leads to a potentially useful heuristic for the maximum number of red clusters in a tree, without performing any explicit colouring. Indeed, Corollary 5.3 only requires computing the diameter of a given tree, which can be computed in $O(|V|)$ time. However, this method assumes that colouring the maximal caterpillar first maximises the total number of red clusters in the tree, which may not be the case.

This heuristic, however, can be improved further. In particular, this heuristic makes the assumption that the trees outside the maximal caterpillar retain the same size until the caterpillar is completed, whereas in reality vertices within these trees will start burning the round after their root is burned. Therefore, after burning a root vertex, we need to consider the minimum number of rounds to burn the remainder of the caterpillar.

COROLLARY 5.4. *Given a maximal caterpillar C_G with length L, and index its spine vertices from 1 to L, where vertex 1 is coloured first when colouring a caterpillar. After colouring the leaf vertex v adjacent to spine vertex n, the minimum number of rounds required to burn the rest of the caterpillar is at least $\lceil \frac{L-n-3}{6} \rceil + 1$.*

PROOF. First, since we colour a caterpillar by splitting it into pieces containing at most 3 spine vertices, and colouring this piece before colouring the remainder, then at most 3 spine vertices ahead of v can be coloured - at most 2 other vertices in the same piece as v, and potentially one more in the next piece, if the ending constraints B or R are chosen. So the number of remaining spine vertices is at least $L-n-3$.

To colour the remainder of the graph, we colour the spine vertices adjacent to the left and right endpoint of the unburned section respectively. This means that at most 6 spine vertices are burned per round, since the fire from these spine vertices can spread in both directions. This then repeats with the remaining unburned section of the spine until no spine vertices are remaining. At this point, some leaf vertices may still be unburned, but these will all be burned within one round. Therefore the minimum number of rounds required is at most $\lceil \frac{L-n-3}{6} \rceil + 1$, proving the corollary. □

An important remark of this corollary is that moving over one leaf vertex, going from position n to position $n+1$, reduces the minimum number of rounds required by at most 1. And since the maximum rooted diameter of a tree rooted at a leaf at position $n+1$ is at most 1 more than that rooted at position n, the tree with maximum possible rooted diameter is still always found at the saame position as before.

From this result, Theorem 5.1 can be updated further:

THEOREM 5.2. *Consider a partial colouring of G as produced by the above procedure, with a maximal caterpillar of length L (hence G has diameter L), and the forest of unburned vertices F comprised of a set of trees $\mathcal{T}$. Then the maximum number of red clusters on F is at most $\lfloor \frac{7L-26}{6} \rfloor$.*

Then in a similar manner, this theorem leads to updated bounds for Corollary 5.3:

COROLLARY 5.5. *Given a tree G with diameter L, which is coloured by first colouring its maximal caterpillar C_G, the maximum number of red clusters on G is at most $\frac{13(L-2)}{12}$.*

We now move onto colouring graphs containing cycles, starting with cactus graphs.

6. COLOURING CACTUS GRAPHS

First, we provide a definition for cactus graphs:

DEFINITION 6.1. *A* cactus graph*, sometimes known as a cactus tree, is a connected graph where each edge is contained in at most one simple cycle. Equivalently, it is also a connected graph where each simple cycle has at most one vertex in common.*

At first glance, the term "cactus tree" is somewhat counterintuitive, since the defining feature of a tree is the absence of cycles. However, since [not quite sure how to justify this], an intuitive perspective on cactus graphs is to consider them as a tree structure of cycles, providing the key motivation for our work with cactus graphs.

In particular, our initial aim will be to find a natural method of transforming cactus graphs into trees, such that our work in Section 5.1 may be applied. One initial suggestion is to remove each cycle by contracting all edges in the cycle, effectively shrinking down each cycle to a single point. This method will certainly produce a tree, but particularly when the cactus graph contains large cycles, this method removes the notion of distance between two vertices on the same cycle, when this distance may be significant in creating clusters on the resulting tree.

As previously discussed, one of the most important characteristics to consider when colouring a tree is the tree's diameter, so any transformation of a cactus graph should aim to preserve the graph's diameter, such that the path defining the cactus graph's diameter will serve as the spine for the resulting tree's maximal caterpillar after transformation. This operation, known as *cycle folding*, is described in the following definition:

DEFINITION 6.2. *Given a cactus graph $\mathcal{C}$, and a cycle C in $\mathcal{C}$ with start vertices s and an end vertex e, the* cycle folding *of $\mathcal{C}$ with respect to s and e is the graph produced by the following operation:*

- *Define $P_{s,e}$ as the shortest path between s and e in C, including both endpoints.*
- *For each vertex v in $C \setminus P_{s,e}$, merge v with the corresponding vertex p in $P_{s,e}$ with $d(v,s) = d(p,s)$.*
- *If any vertices in $C \setminus P_{s,e}$ have not been merged, remove these vertices along with the subgraph rooted at this vertex which is not contained in C.*

Note that, since each simple cycle has at most one vertex in common, our choice of s and e uniquely determines the cycle to be folded, hence the notation $\mathcal{C}_{s,e}$ need not explicitly mention the cycle to be folded.

The following procedure describes a reduction from a cactus graph to a tree, by repeatedly applying cycle folding:

DEFINITION 6.3. *The* cactus reduction *of a cactus graph $\mathcal{C}$ is the tree T resulting from the following procedure:*

1. *Determine the diameter of $\mathcal{C}$, and let P be the path between the endpoints u and v that define the diameter.*
2. *For every vertex in P contained in some cycle C, determine the first and last vertex in P contained in C, and label them s and e respectively. Perform cycle folding on $\mathcal{C}$ with respect to s and e.*
3. *For any remaining subgraphs of $\mathcal{C}$ containing cycles, perform the same procedure, fixing one of the endpoints of P to the vertex that has already been mapped at a previous step.*

Our main motivation for introducing cycle folding is to transform a cactus graph into a tree while maintaining its diameter, so our next step is to check this property holds.

THEOREM 6.1. *Given a cactus graph $\mathcal{C}$ with diameter d, the resulting tree T after cactus reduction also has diameter d.*

PROOF. First, consider the path P between endpoints u and v defining the diameter of $\mathcal{C}$. In particular, if an edge in P between two vertices, say x_i and x_{i+1}, is not contained in a cycle, then cycle folding does not change this edge, so the distance between x_i and x_{i+1} is unchanged.

Now if an edge in P is contained in a cycle, consider $P_{s,e}$ as in Definition 6.2, where s and e are vertices in P representing the start and end of the cycle respectively. Since we merge every vertex in $P_{s,e}$ with some other vertex in the cycle, the distance between s and e is unchanged. Therefore, after cactus reduction, the distance between u and v is unchanged, so the diameter of T is at least d.

Moreover, since the distance between specified endpoints in each cycle is maintained by cycle folding, it cannot be the case that any pair of vertices x and y move further apart after cycle folding. Since the maximum distance between any pair of points in $\mathcal{C}$ is d, the diameter of T is at most d, proving the theorem. $\square$

As discussed in Section 5, when colouring trees, the optimal number of monochromatic components in a tree is closely connected to a tree's diameter. In cactus reduction, we have provided a method for transforming a cactus graph into a tree with the same diameter, allowing us to additionally exploit this connection for cactus graphs. We now aim to show the analogue to Corollary 5.1, to show that colouring the cactus reduction of $\mathcal{C}$ can be suitably extended to the whole of $\mathcal{C}$.

COROLLARY 6.1. *Given a valid colouring of T, the cactus reduction of $\mathcal{C}$, this is also a (not necessarily complete) valid colouring on $\mathcal{C}$.*

PROOF. Given any two vertices in T, s and t such that $d(s,t) = x$ for some x, the preimages of s and t must be at least distance x apart in $\mathcal{C}$, with similar reasoning to the final part of the proof of Theorem 6.1. $\square$

Acknowledgments. Firstly, I would like to thank my supervisors - Jessica Enright, William Pettersson and John Sylvester - for their continual guidance and support throughout the project. I would also like to thank my parents, for always caring for me, and supporting my goals when I thought they could never be attained. And, last but not least, I would like to thank my girlfriend Jodie, for her never-ending love and kindness.

7. REFERENCES