Burning Graphs

Xianghe Xu, Xinrui Chen, Nancy Jia and Haozhen Zheng

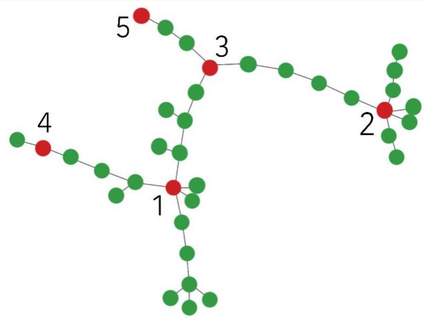
1. Abstract

Graph Burning is a way to simulate the spread of information in a network. Our goal was to find an upper bound on the number of rounds it takes to burn a graph in relation to the number of its vertices, and in particular, attempt to prove the **Burning Number Conjecture**. We explored properties that a minimal counterexample to the conjecture would have to have. We also created the concept of (k, ℓ)-burnability, which is a relaxation of regular graph burning that can be computationally easier if $k < \ell$. We created an algorithm to check if a graph is (k, ℓ)-burnable, which runs quickly on small graphs, and our hope is to use this algorithm to find either a counterexample to the burning number conjecture or to build intuition on which kinds of graphs are hard to burn.

2. Introduction

Given a finite, simple, undirected graph G, the burning of G is a discrete-time process; at each step, vertices of G may either be **burned** or **unburned**. At each round, we pick one vertex (a **source**) and set it on fire. In addition, all the unburned neighbors of vertices that were burned in a previous round will also be burned. The process repeats until all the vertices in G are burned.

The **burning sequence** is the sources chosen, in order. The **burning number** of a graph G, $b(G)$, is the minimum number of rounds that is needed for G to be burned entirely. The smaller the burning number of a graph, the faster a piece of information can spread in this represented network.



Above is a tree that can be burned in 5 rounds. The labeled red vertices are the sources that were picked, while the labels represent which round they were chosen.

The Burning Number Conjecture, [1] (Bonato, Janssen, Roshanbin, 2016): For a connected graph G of order n,

$$b(G) \leq \lceil \sqrt{n} \rceil.$$

Note that paths realize the upper bound in the Burning Number Conjecture. That is, $b(P_n) = \lceil \sqrt{n} \rceil$ for all n.

Bonato and Kamali currently hold the best known upper bound for the burning number of a connected graph.

Theorem, [2] (Bonato, Kamali, 2021): If G is a connected graph of order n, then

$$b(G) \leq \left\lceil \frac{\sqrt{12n+64}+8}{3} \right\rceil = \left\lceil \sqrt{\frac{4n}{3}} \right\rceil + O(1)$$

3. Reductions

Tree Reduction Theorem, [1] (Bonato, Janssen, Roshanbin, 2016): For a connected graph G we have that

$$b(G) = min\{b(T) : T \text{ is a spanning tree of } G\}$$

The **Tree Reduction Theorem** implies that if the burning conjecture is false, it is false for a tree, so we focus only on trees. In this project, we approach this problem by considering a tree that is a minimal counterexample, meaning it has the fewest number of vertices among all counterexamples.

Lemma: If the tree T is a minimal counterexample, we can assume T

- Has $\leq 2\sqrt{n} - 1$ leaves
- Has no pendant paths of length $\geq 2\sqrt{n} - 1$

note: A pendant path of T is a path whose removal won't disconnected the tree.

We prove the first assumption by contradiction. Assume T has $\geq 2\sqrt{n}$ leaves, and let T^* denote the tree formed by deleting every leaf of T. Notice that $b(T) \leq b(T^*) + 1$ since after T^* burns, all the leaves of T will burn after at most one more round. If T^* is not a counterexample, we can burn it in $\leq \left\lceil \sqrt{n - 2\sqrt{n}} \right\rceil$ rounds. Then

$$b(T) \leq \left\lceil \sqrt{n - 2\sqrt{n}} \right\rceil + 1 \leq \sqrt{n},$$

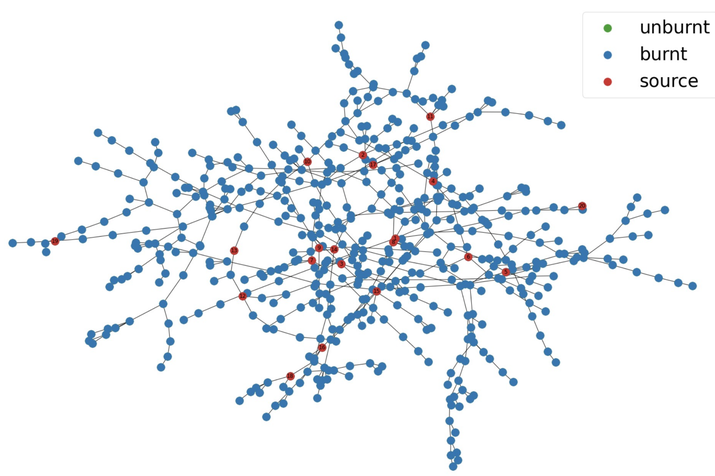
which contradicts the assumption that T was a minimal counterexample. The proof for the second assumption is similar.

4. (k, ℓ) - Burnability

Given a tree T, if it can be burned after we pick k sources and let them all burn for ℓ rounds, then T is (k, ℓ)**-burnable**. In this (k, ℓ)-burning process, we assume that $\ell \geq k$. We make this assumption because each time we choose a source, the graph will burn for one more round, then after we are done placing the sources, the fire can still spread $\ell - k$ rounds. The first source will burn for ℓ rounds; the second source will burn for $\ell - 1$ rounds; the ith source will burn for $\ell + 1 - i$ rounds, where $1 \leq i \leq k$.

There are k sources that need to be chosen. For each source, we have n choices, hence the run-time is $O(n^k)$. Since every source will burn for at most ℓ rounds, the total run-time for the algorithm is $O(\ell \cdot n^k)$. This means that it mainly depends on k. So, instead of directly asking if a graph is $(\sqrt{n}, \sqrt{n})$-burnable (which corresponds to the burning number conjecture), we came up with the concept of (k, ℓ)-burning, which allows us to reduce time complexity. By choosing smaller k (polynomial coefficient) and larger ℓ (constant coefficient), when $k \ll \ell$, we can determine if a graph is (k, ℓ)-burnable in less time.

Below is the result of running the algorithm on a tree with 529 vertices. The 23 red vertices are one possible way of choosing the sources to fully burn the tree.



5. (k, ℓ)-Burning Algorithm

In this section, we describe the algorithm we created to check if a graph is (k, ℓ)-burnable.

Algorithm
Input: G, k, ℓ
Output: source, burned, isburnable
Get number of nodes of G: N and return when $N = 1$
Compute L and check if the graph is burnable ▷ edge case
while current level < $k - 1$ **do** ▷ round $1, 2, ..., k - 1$
  if len(R[current level]) = 0 **then**
    modify R, source and go back to current level - 1
  end if
  pick source from R[current level]
  burn the graph for one round
  check: if burnable, return source, burned nodes and isburnable
  generate R[current level + 1], current level += 1
end while
while len($R[k]$) $\neq 0$ **do** ▷ round $k, k + 1, ..., \ell$
  pick source from R[current level]
  burn the graph for ℓ - k rounds
  check: if burnable, return source, burned nodes and isburnable
end while
move up to $k - 1$ level and run first while
return source, bestsequence, False ▷ fail to burn, return best way

Design idea:
In order to make our algorithm run better, we wanted to choose our sources intelligently. As such, instead of choosing any unchosen vertex in every round, we pick ith source by excluding the vertices which have been burned until ith round and whose distance to the leaves is bigger than $\ell - i$ ($i = 1, 2, \ldots, k$).

Challenge:
1. Implement Dynamic structure:
By the design idea, different source picked at previous round will lead to different sets of available sources at the current round. Therefore, we implemented a dynamic *depth-first-search* on the collection of all possible "reasonable" burning sequences. We will record available sources at each round by a list of sublists R, vertices distance, and new burned vertices at each round. With these information, we can *move down* and generate $R[i + 1]$ by applying the action sequence "pick the first node in $R[i]$ as source and record, remove it from $R[i]$ (for $i = 1, \ldots, k$)". Our algorithm can *move up* to $R[i - 1]$ by removing all the nodes burned at ith round and removing the $(i - 1)$st source , then go back to $(i - 1)$st level and generate the new $R[i]$.
2. Edge cases
There are many edge cases in real situations, such as burning too fast, and generating an empty list $R[i]$. To overcome these, we set checkpoints at each round to save time when burn the graph before picking the kth source or burning the ℓth round.

6. Acknowledgements

We would like to thank Sean English, Mina Nahvi, Jingwei Xu and the IGL Program. IGL research is supported by the Department of Mathematics at the University of Illinois at Urbana-Champaign, and by the National Science Foundation under Grant Number DMS-1449269.

7. References

[1] A. Bonato, J. Janssen, and E. Roshanbin. How to burn a graph. *Internet Mathematics* 12.1-2 (2016): 85-100.

[2] A. Bonato, and S. Kamali. An improved bound on the burning number of graphs. `arXiv:2110.01087`, (2021).