The burning number of directed graphs
Bounds and computational complexity

Janssen, Remie

DOI
10.20429/TAG.2020.070108

Publication date
2020

Document Version
Final published version

Published in
Theory and Applications of Graphs

Theory and Applications of Graphs

Volume 7 | Issue 1 Article 8

June 2020

The Burning Number of Directed Graphs: Bounds and Computational Complexity

Remie Janssen
Delft University of Technology, remiejanssen92@gmail.com

Recommended Citation

Janssen, Remie (2020) "The Burning Number of Directed Graphs: Bounds and Computational Complexity," *Theory and Applications of Graphs*: Vol. 7 : Iss. 1 , Article 8.
DOI: 10.20429/tag.2020.070108
Available at: https://digitalcommons.georgiasouthern.edu/tag/vol7/iss1/8

The Burning Number of Directed Graphs: Bounds and Computational Complexity

Cover Page Footnote

Research funded by the Netherlands Organization for Scientific Research (NWO), Vidi grant 639.072.602 of dr. Leo van Iersel. The author would like to thank Josse van Dobben de Bruyn for bringing graph burning to his attention, and Mark Jones for lending his expertise in parametrized complexity.

Abstract

The burning number of a graph was recently introduced by Bonato et al. Although they mention that the burning number generalises naturally to directed graphs, no further research on this has been done. Here, we introduce graph burning for directed graphs, and we study bounds for the corresponding burning number and the hardness of finding this number. We derive sharp bounds from simple algorithms and examples. The hardness question yields more surprising results: finding the burning number of a directed tree with one indegree-0 node is NP-hard, but FPT; however, it is W[2]-complete for DAGs. Finally, we give a fixed-parameter algorithm to find the burning number of a digraph, with a parameter inspired by research in phylogenetic networks.

1 Introduction

The burning number of a graph was recently introduced by Bonato et al. [BJR14] as a model of social contagion. Social contagion is the spread of rumours, behaviour, emotions, or other social information through social networks (e.g., [BFJ$^+$12, KGH14]). This information transfer can be active, but may also happen passively, for example by reading posts on social media.

In the discussion of [BJR16], Bonato et al. remark that the burning number generalises naturally to directed graphs. However, no further research on burning directed graphs has been done. Nevertheless, the topic seems relevant, for example in the same setting as undirected graphs, social contagion: some communication occurs in mostly one direction, such as when people follow other people on social media. Hence, in this paper, we introduce the directed version of graph burning and we study some basic problems related to graph burning.

The central concept in graph burning is the burning number. The burning number of a graph is the number of steps it takes to 'burn' this graph. Here, burning is a step-wise process roughly defined as follows. In each step, all neighbours of all burned nodes are burned, and one extra node is chosen that is burned as well. This models social contagion in the sense that neighbours start to burn correspond to information being spread through the network. The basic questions we study relate to bounds on the burning number for different classes of graphs, and to the computational complexity of determining the burning number of a graph. Both of these problems have been studied for undirected graphs [LL16, BBJ$^+$18, BL19, BBJ$^+$17].

Structure of the paper. We start by defining our notation, and basic concepts in graph burning in Section 2. Then, in Section 3, we give bounds on the burning number for weakly connected digraphs, single sourced DAGs, and strongly connected digraphs. Following this, in Section 4, we turn to the complexity of the graph burning problem in directed graphs. We show that the problem is NP-hard for directed trees with one indegree-0 node, but solvable in FPT time; and for DAGs, we show that the problem is W[2]-hard. We finish this section with an algorithm for digraph burning. Finally, in Section 5, we discuss the relation between directed graph burning, undirected graph burning and some other problems.

2 Preliminaries

2.1 DAGs

A directed acyclic graph (DAG) is a digraph D without directed cycles, whose nodes are denoted $V(D)$ and its arcs $A(D)$. For DAGs, we use the following terminology: a *source* of a DAG D is a node $v \in V(D)$ with indegree-0; a *sink* is a node with outdegree-0. A *single source DAG* is a DAG with exactly one source.

We now introduce some specific DAGs studied in this paper. The first one is the *directed path* P_n on n nodes. This is the single source DAG with one sink which has the undirected path on n nodes as its underlying undirected network. Secondly, the *two-legged spider* Λ_n, which is the single source DAG constructed from two copies of P_n by identifying their source nodes. Equivalently, it is the single source DAG with the undirected path on $2n-1$ nodes as underlying undirected network whose source node is the middle node of this undirected path. Lastly, a *directed tree* is a single source DAG whose underlying undirected graph is an undirected tree. Note that all nodes of a directed tree have indegree one, except for the source, which has indegree zero.

2.2 Digraph burning

The burning number of directed graphs is defined analogously to that of undirected graphs, with one adaptation: burning only occurs in the direction of arcs. This leads to the following definition of digraph burning.

A sequence $(v_1, \ldots, v_b)$ of nodes is a *burning sequence* for a digraph D if after b steps of the following process, every node of D is *burned*. The i-th step consists of burning all out-neighbours of all currently burning nodes, and then burning the node v_i. Note that burning of neighbours occurs before burning the chosen node, so after the first step, only the chosen node is burned. The *burning number* of a digraph is the length of a shortest burning sequence for that graph. This leads to the following computational problem.

Digraph Burning
Input: A digraph D.
Parameter: A natural number b.
Output: YES if a burning sequence for D of length at most b exists; NO otherwise.

When we restrict the input to smaller classes of digraphs, like directed trees or DAGs, we change the name of the problem accordingly, so it becomes Tree Burning resp. DAG Burning.

Like for undirected graphs, the directed burning number can alternatively be defined using covers of neighbourhoods [BJR16]. The *k-out-neighbourhood* $\mathcal{N}_k^+(v)$ of a node v consists of all nodes that can be reached from v using at most k arcs. Similarly, the *k-in-neighbourhood* $\mathcal{N}_k^-(v)$ of v consists of all nodes from which v can be reached using at most k arcs. Note that $\mathcal{N}_0^-(v) = \mathcal{N}_0^+(v) = \{v\}$.

https://digitalcommons.georgiasouthern.edu/tag/vol7/iss1/8
DOI: 10.20429/tag.2020.070108

Now let D be a digraph and $(v_1, \ldots, v_b)$ a sequence of nodes, then $(v_1, \ldots, v_b)$ is a burning sequence for D if and only if

$$V(D) = \bigcup_{i=1}^{b} \mathcal{N}_{i-1}^{+}(v_{b-i+1}).$$

Note that, in the burning process for a burning sequence, a node v_i may already be burning when we try to burn it in step i. For undirected graphs, this is often not allowed. However, there is always a shortest burning sequence in which this does not happen. For directed graphs, the same holds: instead of choosing to burn an already burning node, one can burn an arbitrary other node.

For algorithmic purposes, we also consider variants of the directed burning problem. For recursion, we need to consider instances where parts of the graph are already burned, and, hence, do not need to be burned anymore; and we need to consider instances where we are no longer allowed to choose a node to burn in each step.

These variations naturally lead to the concepts of partial burning—first defined for undirected graphs in [BEKM19]—and burning ranges. For the first, we no longer ask for the whole digraph to be burned at the end of the process, but only a selected subset of nodes. In other words, the digraph only needs to be partially burned. For the second, it is convenient to consider the covering definition of the burning number where the sequence is replaced by a map. That is, the burning number is at most b if there is a *burning assignment* $\phi : [b] \to V(D)$ such that

$$V(D) = \bigcup_{i=1}^{b} \mathcal{N}_{i-1}^{+}(\phi(i)).$$

In this formulation, we call the set $[b] = \{1, \ldots, b\}$ the set of *burning ranges*. Of course, the set of burning ranges can actually be any multiset of natural numbers. This leads to the following generalization of the burning problem.

Partial Digraph Burning With Ranges
Input: A Digraph $D = (V, A)$, a set $X \subseteq V$, and a (multi-)set of burning ranges R.
Output: YES if there is a burning assignment $\phi : R \to V$ such that $X \subseteq \bigcup_{r \in R} \mathcal{N}_{r-1}^{+}(\phi(r))$; NO otherwise.

3 Bounds for the directed burning number

For undirected graphs, people have attempted to prove the conjecture that any connected graph with n nodes has burning number at most $\lceil \sqrt{n} \rceil$. Therefore, the worst case for undirected graphs seems to be the path, on which this bound is attained. Here, we investigate similar bounds for directed graphs with several connectivity assumptions. We start by showing that not assuming any connectivity makes the problem quite uninteresting. Later, we show that for directed graphs, the directed path is the worst case among all sufficiently connected digraphs.

Lemma 3.1. *Let D be a digraph with n nodes. If D contains no arcs, or $n = 1, 2$, the burning number is equal to n. If $n > 2$ nodes and D has at least one arc, the burning number is at most $n - 1$, and this bound is sharp.*

Proof. The first part of the lemma is obvious, so we assume D has at least one arc, and more than 2 nodes. We first burn the tail end of an arc. The head of this arc will burn in the next time step. In the subsequent time steps, we choose to burn all other nodes. This takes $n-1$ steps in total. The bound is sharp, as we can take the digraph with $n-2$ isolated vertices, and the remaining 2 nodes connected by an arc. □

This proof relies on the digraphs being poorly connected. As this makes no sense in the setting of social contagion, we now restrict to weakly connected digraphs.

Lemma 3.2. *For weakly connected digraphs with $n > 2$ nodes, the burning number is at most $n - 1$, and this bound is sharp.*

Proof. As weakly connected digraphs with $n > 2$ nodes have at least one arc, Lemma 3.1 applies and the bound of $n - 1$ holds. The bound is still sharp: take the digraph consisting of $n - 1$ sources, all with an arc pointing to the remaining node. As each source has to be part of the burning sequence, this digraph has burning number at least $n - 1$. □

In this case, the number of sources makes it easy to force a high burning number. Therefore, we now limit the number of sources as well, before turning to strongly connected digraphs.

3.1 Single source DAGs

We start by computing the burning number of the simplest single source DAG, the directed path. Then, we show that directed paths have the largest burning number, in the sense that among all single source DAGs with n nodes, the burning number of the directed path is the largest.

Lemma 3.3. *The burning number of the directed path on n nodes, P_n, is*

$$\left\lceil \sqrt{2n + \frac{1}{4}} - \frac{1}{2} \right\rceil.$$

Proof. In t time steps, we can burn $t(t+1)/2$ nodes of the directed path. Hence, to burn a directed path of length n, we need t_n steps such that $n \leq t_n(t_n+1)/2$, or $t_n \geq \sqrt{2n + \frac{1}{4}} - \frac{1}{2}$. As the burning number is integral, we take the ceiling to get the burning number. □

To prove that all single source DAGs with n nodes have burning number at most $\left\lceil \sqrt{2n + 1/4} - 1/2 \right\rceil$, we use the concept of eccentricity. The *eccentricity* of a node v is the maximal distance to any node that can be reached from v. Note that we only look at the distances to nodes that can be reached. Hence, the eccentricity of a node is always a finite number. For the proof of the next lemma, we use the partial order induced by a DAG: imagining all arcs to be drawn downward, we say u is *below* v if there is a directed path from v to u.

Lemma 3.4. *Let D be a single source DAG, and let ρ be its source with eccentricity e. For all $i \in [e]$, there is a node in D with eccentricity i.*

Proof. We prove the claim using induction on the eccentricity from high to low. For the basis, note that the source has eccentricity e, so there exists a *lowest* node of eccentricity e: a node v with eccentricity e such that there is no node with eccentricity e below v.

Now assume there is a node v_i with eccentricity $i \in [e]$ with no node of eccentricity at least i below it. We prove there exists a node with eccentricity $i-1$ with no node of eccentricity at least $i-1$ below it. Note that at least one node $c \in \mathcal{N}_1^+(v_i)$ has eccentricity at least $i-1$, otherwise the shortest distance of i from v_i to some other node cannot be achieved. As v_i there is no node below v_i with eccentricity i or greater, c has to have eccentricity $i-1$. Now look for the lowest node with eccentricity $i-1$ below c (including c). This node has no nodes of eccentricity $i-1$ or greater below it.

Therefore, for each $i \in [e]$, there is a node with eccentricity i and no node of eccentricity at least i below it. In particular, for each $i \in [e]$, there is a node with eccentricity i, which is what we needed to prove. □

Proposition 3.1. *The directed burning number of a single source DAG with n nodes is at most $\left\lceil \sqrt{2n + \frac{1}{4}} - \frac{1}{2} \right\rceil$.*

Proof. For convenience in notation, we define b_k as the smallest b such that $k \leq \sum_{i=1}^{b} i$, i.e., $b_k := \left\lceil \sqrt{2k + \frac{1}{4}} - \frac{1}{2} \right\rceil$. We use induction on the number of nodes in the digraph to prove the claim. For the induction basis, we say the empty digraph has burning number 0.

Now assume each single source DAG with $n < N$ nodes has burning number at most b_n. Consider a single source DAG D with N nodes. First suppose the source s has eccentricity at most $b_N - 1$. In that case, $V(D) = \mathcal{N}^+_{b_N-1}(s)$, and the burning number of D is at most b_N. Now suppose the source has eccentricity greater than $b_N - 1$, then there exists a node v of eccentricity $b_N - 1$ (Lemma 3.4). The neighbourhood $\mathcal{N}^+_{b_N-1}(v)$ contains at least b_N nodes, and $D' = D \setminus \mathcal{N}^+_{b_N-1}(v)$ is again a single source DAG. As b_N is the smallest b such that $N \leq \sum_{i=1}^{b} i$, we have $b_{N-b_N} \leq b_N - 1$. Hence, by the induction hypothesis, D' has burning number at most $b_N - 1$. Taking the corresponding cover of D', and adding $\mathcal{N}^+_{b_N-1}(v)$, we see that D has burning number at most b_N. □

3.2 Strongly connected digraphs

Lemma 3.5. *The burning number of the directed cycle on n nodes, C_n, is $\left\lceil \sqrt{2n + \frac{1}{4}} - \frac{1}{2} \right\rceil$.*

Proof. The same as the proof for the directed path. □

Theorem 3.6. *The burning number of a strongly connected digraph is at most $\left\lceil \sqrt{2n + \frac{1}{4}} - \frac{1}{2} \right\rceil$, and this bound is sharp.*

Proof. Take an arbitrary vertex v of a strongly connected digraph G with n nodes, and consider the subgraph consisting of all the arcs from all shortest paths from v to all other nodes. This is a single source DAG with n nodes, so it has burning number at most $\left\lceil \sqrt{2n + \frac{1}{4}} - \frac{1}{2} \right\rceil$

by Proposition 3.1. This implies that the burning number of G is also at most $\left\lceil \sqrt{2n+\frac{1}{4}}-\frac{1}{2} \right\rceil$. By Lemma 3.5, this bound is sharp. □

4 Complexity of computing the directed burning number

In this section, we consider the complexity of the directed graph burning problem. It is easy to see that the problem is NP-hard in general. Indeed, a reduction from the undirected version—which is NP-hard as well [BBJ+17]—can easily be given: take an undirected graph, and replace each edge by two arcs in both directions. Although this shows hardness, it does not highlight the differences between the directed and the undirected versions of the graph burning problem.

Here, we restrict to digraphs that are not strongly connected: directed trees and DAGs. We show that the problem remains NP-hard, even if we restrict to directed trees, but that this problem can be solved in FPT time. Then, we turn to DAGs, for which we show the problem is W[2]-complete with the burning number as parameter, which indicates there is probably no fixed parameter algorithm with this parameter. Finally, we give two algorithms that, considering the previous, solve the problem relatively efficiently.

4.1 Tree Burning

In this subsection, we show that the directed burning problem is NP-hard, even when considering only directed trees. Like for undirected graph burning [BBJ+17], the proof uses a reduction from Distinct 3-Partition, which is a strongly NP-hard problem [HWW08].

> Distinct 3-Partition
> **Input:** A finite set $A = \{a_1, a_2, \ldots, a_{3n}\}$ of positive distinct integers, and a positive integer B where $\sum_{i=1}^{3n} a_i = nB$, and $B/4 < a_i < B/2$, for $1 \leq i \leq 3n$.
> **Output:** YES if there is a partition of A in n triples such that the elements of each triple sum up to B; NO otherwise.

The reduction in the proof of the following lemma constructs a directed tree from a sequence of small components. These components are directed paths P_n on n nodes, and *2-legged spiders* Λ_n constructed from two copies of P_n by identifying their source nodes.

Lemma 4.1. Tree Burning *is NP-complete.*

Proof. Tree Burning is in NP as a sequence of burning nodes is a certificate that can be checked in polynomial time, so we focus on NP-hardness.

We reduce from Distinct 3-Partition. Let $A = \{a_i\}_{i=1}^{3n}$ be an instance of Distinct 3-Partition with $B = \frac{1}{n}\sum a_i$ and $m = \max A$, and let $Z = [m] \setminus A = \{z_i\}_{i=1}^{m-3n}$ be the set of integers up to m not contained in A. We will create an instance of Tree Burning from the sequence of digraphs

$$\Lambda_{z_1}, \Lambda_{m+1}, \ldots, \Lambda_{z_{|Z|}}, \Lambda_{m+|Z|}, P_B, \Lambda_{m+|Z|+1}, \ldots, P_B, \Lambda_{m+|Z|+n-1}, P_B,$$

which consists of two parts. The first part consists of pairs Λ_{z_i} followed by Λ_{m+i}, and the second part consists of n copies of P_B alternating with $\Lambda_{m+|Z|+1}$ up to $\Lambda_{m+|Z|+n-1}$.

To create the instance of Tree Burning, first take the disjoint union of all these digraphs. Then, for each digraph except the last: add an arc from a sink of this digraph to the source of the next digraph in the sequence (See Figure 1).

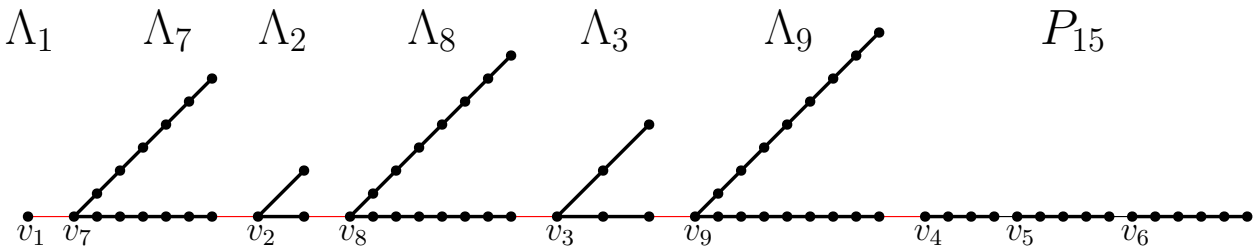

Figure 1: Example of Tree Burning hardness reduction. The constructed YES-instance of Tree Burning for the Distinct 3-Partition YES-instance $A = \{4, 5, 6\}$ and $B = 15$, which gives $Z = \{1, 2, 3\}$ and $m = 6$. All arcs are directed to the right. The graph is constructed from the sequence $\Lambda_1, \Lambda_7, \Lambda_2, \Lambda_8, \Lambda_3, \Lambda_9, P_{15}$. The disjoint union of the digraphs in the sequence consists of the black edges; the red edges are the edges connecting the subsequent digraphs in the sequence. The burning sequence is $(v_9, \ldots, v_1)$.

Note that each part corresponding to an element of Z or of the n copies of P_B is followed by a 2-legged spider. The instance of Tree Burning consists of this digraph, and the integer $m+|Z|+n-1$. Note that the digraph has fewer than $2(1+\cdots+(m+|Z|+n-1)) < 2(1+\cdots+3m) = O(m^2)$ edges. Because 3-Partition is strongly NP-hard, we may assume the instance size is polynomial in B. Hence, as $m < B$, the reduction can be performed in polynomial time.

We now prove the reduction provides a yes-instance of Tree Burning iff the instance of Distinct 3-Partition is a yes-instance. First, note that the main path has length exactly $1 + \cdots + (m + |Z| + n - 1)$. Hence, to burn the digraph in exactly $m + |Z| + n - 1$ steps, all the burning neighbourhoods must be disjoint, and all chosen burning nodes must lie on this directed path. Note that the legs of the Λ_k that do not correspond to elements of Z are of length $m + 1, \ldots, m + |Z| + n - 1$: all longest ranges. Hence, they must be burned at a time step such that this leg burns exactly to the end. This implies that each top node of a Λ_k must be burned in step $m + |Z| + n - k$. The remaining lengths are exactly the a_i, and they can cover the P_Bs exactly if and only if the Distinct 3-Partition instance is solvable (a partition cannot contain parts of size other than three, because $B/4 < a_i < B/2$, for $1 \leq i \leq 3n$). □

Algorithm 1: TreeBurning(T, R)

Data: A directed tree T and a multiset of allowed burning ranges R.
Result: A map $\phi : R \to V(T)$ if T can be burned by using R, NO otherwise.

```
Let s be a sink of T furthest from the source ρ;
if d(ρ, s) ≤ max(R) − 1 then
    return {r ↦ ρ : r ∈ R} ;
end
for r in R do
    Let v be the node with d(v, s) = r − 1;
    Let T′ be T with all nodes of distance r − 1 away from v removed;
    Calculate TreeBurning(T′, R \ {r});
    if the result is a map φ′ then
        let φ be the map obtained from φ′ by adding r ↦ v;
        return φ;
    end
end
return NO;
```

Lemma 4.2. *Let T be a directed tree, R a multiset of burning ranges which can burn T, and s a sink furthest from the source ρ. Then, either $d(\rho, s) \leq \max(R) - 1$, or there exists a burning assignment such that, for some $r \in R$, the node v with $d(v, s) = r - 1$ is burned with range r.*

Proof. First note that if $d(\rho, s) \leq \max(R) - 1$, then assigning $\max(R)$ to ρ and assigning all other ranges arbitrarily gives a burning assignment that burns T. So, for the rest of the proof, we assume $\max(R) - 1 < d(\rho, s)$.

Let $\phi : R \to V(T)$ be a burning assignment, and let v be a node with $\phi(r) = v$ and $s \in \mathcal{N}^+_{r-1}(v)$. If $d(v, s) = r - 1$ or $v = \rho$, then the statement of the lemma holds. Hence, suppose that $d(v, s) < r - 1$. As $d(v, s) < r - 1 \leq \max(R) - 1 < d(\rho, s)$, there is a node u on the directed path from ρ to v with $d(u, s) = r - 1$. Because the distance from ρ to s is maximal among all sinks, $\mathcal{N}^+_{r-1}(x) = \mathcal{N}^+_{d(v,s)}(x)$ for all x with $d(x, s) \leq r - 1$, so in particular, the same holds for u. As $\mathcal{N}^+_{r-1}(v) \subseteq \mathcal{N}^+_{r-1}(u)$, reassigning r to u gives a new burning assignment of R that burns T for which the condition of the lemma holds. □

Theorem 4.3. *The directed tree burning problem,* Tree Burning*, on T with n nodes and parameter b can be solved in $O(b! \operatorname{poly}(n))$ time.*

Proof. We claim that Algorithm 1 can decide whether the burning number of a directed tree T is at most b by calling TreeBurning$(T, [b])$.

By Lemma 4.2, we have the following: if a directed tree can be burned with the given ranges R, then it can either be burned with one range at the source, or there is a solution that does not overshoot the furthest sink. The algorithm recursively searches for solutions of that type, and will therefore find a solution if it exists.

For the running time, note that each time the algorithm branches, the set of ranges reduces by one. This means there are at most $b!$ recursive calls of the algorithm in total. Each call takes polynomial time, so the running time is $O(b! \operatorname{poly}(n))$. □

https://digitalcommons.georgiasouthern.edu/tag/vol7/iss1/8
DOI: 10.20429/tag.2020.070108

Corollary 4.4. Tree Burning *is FPT with the burning number as parameter.*

4.2 DAG Burning

In the previous section, we saw that Tree Burning is NP-hard, but solvable in FPT time. One could hope that an fixed-parameter algorithm also exists for DAG Burning with the burning number as the parameter. However, as we will show in this section, this is quite unlikely, as DAG Burning is W[2]-complete. This implies that the increased hardness is a consequence of nodes with higher indegree. Hence, we give an fixed-parameter algorithm for DAG Burning with a parameter combining the burning number and the 'extra indegree'.

Set Cover
Input: A universe $\mathcal{U}$ and a set $\mathcal{S}$ of subsets $S \subseteq \mathcal{U}$ with $\cup_{S\in\mathcal{S}} S = \mathcal{U}$.
Parameter: A natural number k, the number of subsets in the cover.
Output: YES if there is a cover $C \subset \mathcal{S}$ of $\mathcal{U}$ with $|C| \leq k$; NO otherwise.

To prove W[2]-hardness, we give a parameterized reduction from Set Cover to DAG Burning, and to prove the problem is in W[2], we give a parameterized reduction in the other direction. As Set Cover is W[2]-complete ([CFK+15] Theorem 13.21), this suffices to show that DAG Burning is W[2]-complete as well.

Lemma 4.5. *The* DAG Burning *problem is W[2]-hard with the burning number as parameter.*

Proof. We give a parameterized reduction from Set Cover. Let $(\mathcal{U}, \mathcal{S}, k)$ be an instance of Set Cover, we construct an instance (D, b) of DAG Burning as follows. We set $b = k+2$, and D to the DAG consisting of the nodes

$$\left\{v_A^1, \ldots, v_A^{k+1}\right\}_{A\in X} \cup \left\{v_u\right\}_{u\in\mathcal{U}} \cup \left\{\rho, v_{(\rho,1)}^{k+2}\right\},$$

and the edges

$$\left\{(v_S^{k+1}, v_u) : u \in S\right\}_{S\in\mathcal{S}} \cup \left\{(v_A^i, v_A^{i+1}) : i \in [k]\right\}_{A\in X} \cup \left\{(\rho, v_A^1)\right\}_{A\in X} \cup \{(v_{(\rho,1)}^{k+1}, v_{(\rho,1)}^{k+2})\},$$

where $X = \{(\rho, j)\}_{j=1}^{b} \cup \mathcal{S}$ (e.g., Figure 2). This digraph can be constructed in polynomial time, and $b(k) = k + 2$ is a computable function. Hence, the only thing left to check is that (D, b) is a yes-instance of DAG Burning if and only if $(\mathcal{U}, \mathcal{S}, k)$ is a yes-instance of Set Cover.

First note that, to burn all directed paths consisting of the $v_{(\rho,j)}^i$ nodes, the source ρ needs to be burned in the first time step. Otherwise, the burning sequence needs to contain a node in each of these paths. Then, the remaining node $v_{(\rho,1)}^{k+2}$ can be burned in the last step by adding it in the burning sequence; doing it earlier does not gain any extra burned nodes, so a burning sequence can be assumed to do this. Lastly, we have the second to the $(k + 1)$-th element of the burning sequence. To burn the nodes $\{v_u\}_{u\in\mathcal{U}}$, these remaining elements have to be chosen from the $v_{S_j}^i$; in fact, we may choose to pick them from the $v_{S_j}^{k+1}$, as choosing higher nodes will not result in any more nodes being burned in the end. It is easy to see that a choice of k of these nodes corresponds to a choice of k subsets from $\mathcal{S}$, and that all nodes v_u are burned iff these corresponding subsets form a cover of $\mathcal{U}$. Therefore, (D, b) is a yes-instance of DAG Burning if and only if $(\mathcal{U}, \mathcal{S}, k)$ is a yes-instance of Set Cover. □

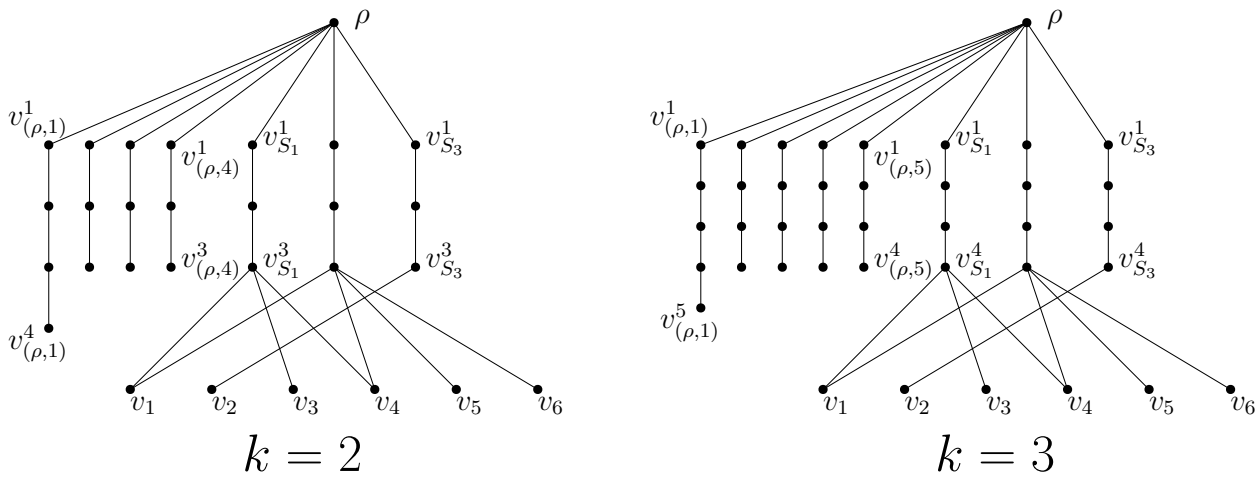

Figure 2: DAG burning W[2]-hardness reduction. The constructed instances of DAG Burning for the Set Cover instances $([6], \mathcal{S}, 2)$ and $([6], \mathcal{S}, 3)$ with $\mathcal{S} = \{\{1,3,4\},\{1,4,5,6\},\{2\}\}$. All arcs are directed down. The first is a no-instance, the second a yes-instance.

Lemma 4.6. *The* DAG Burning *problem is in W[2] with the burning number as parameter.*

Proof. We give a parameterized reduction to Set Cover. Let (D, b) be an instance of DAG Burning, we construct an instance $(\mathcal{U}, \mathcal{S}, k)$ of Set Cover as follows. Let $\mathcal{U}$ be the set consisting of all nodes of D together with a nodes x_i for each $i \in [b]$. The subsets $\mathcal{S}$ are the burning neighbourhoods of all nodes, i.e.,

$$\left\{\mathcal{N}^+_{i-1}(v) \cup \{x_i\} : v \in V(D), i \in [b]\right\},$$

lastly, we set $k := b$. As neighbourhoods can be calculated in polynomial time, and there are $nb < n^2$ neighbourhoods to compute, the instance can be produced in polynomial time. Hence, the remaining part is to prove that (D, b) is a yes-instance of DAG Burning if and only if $(\mathcal{U}, \mathcal{S}, k)$ is a yes-instance of Set Cover.

To cover the set $V(D) \cup \{x_i\}_{i=1}^b$ with b sets, we must choose exactly one set containing x_i for each i. This corresponds to burning the i-neighbourhood for a chosen node for each $i = 1 \ldots, b$. That is, if there is a set cover, then we can burn D in b steps. The other direction is analogous: if we can burn D in b steps, then the corresponding sets will be a set cover. Therefore, (D, b) is a yes-instance of DAG Burning if and only if $(\mathcal{U}, \mathcal{S}, k)$ is a yes-instance of Set Cover, and we have a parameterized reduction from DAG Burning to Set Cover, which proves that DAG Burning is in W[2]. □

Theorem 4.7. *The* DAG Burning *problem is W[2]-complete with the burning number as parameter.*

This theorem implies there is probably no fixed-parameter algorithm for DAG Burning with the burning number as parameter. However, this does not exclude the possibility that there are fixed-parameter algorithms with a slightly larger parameter, or non fixed-parameter algorithms that have relatively efficient running time when the burning number is small. An

algorithm in the last category is Algorithm 2, an adapted version of the guessing algorithm from [KMZ19]. In Theorem 2 in that paper, they essentially give an algorithm with running time $O(n^b \operatorname{poly}(n))$, which is polynomial time when the burning number is bounded by a constant. In their case, this happens when the diameter is bounded by a constant (as the burning number is at most the diameter).

Algorithm 2: BRUTEFORCEBURNING(D, b)

Data: A digraph $D = (V, A)$ and an integer b.
Result: YES if $b(D) \leq b$, NO otherwise.

```
for S ∈ V^b do
    if S is a burning sequence for D then
        return YES;
    end
end
return NO;
```

Proposition 4.1. *Algorithm 2 solves* DIGRAPH BURNING *in* $O(n^b \operatorname{poly}(n))$ *time.*

For a more sophisticated approach, we try to extend the parameter to include the 'extra indegree', which makes DAG BURNING hard compared to TREE BURNING. The extra indegree is inspired by the reticulation number of a phylogenetic network (see, for example, [JJE+18]), and is defined as follows. The *reticulation number* $k(D)$ of a digraph $D = (V, A)$ is $\sum_{v \in V} \max(0, \operatorname{indeg}(v) - 1)$. The following algorithm is a fixed-parameter algorithm for DIGRAPH BURNING with $(k(D) + 1)b$ as parameter.

Algorithm 3: PARTIALDIGRAPHBURNINGWITHRANGES(D, X, R)

Data: A digraph D, a subset X of nodes to be burned, and a multiset of allowed burning ranges R.
Result: A map $\phi : R \to V(D)$ if X can be burned by using R, NO otherwise.

```
if X = ∅ then
    return the map φ : ∅ → ∅;
end
Let x ∈ X be arbitrary;
for r in set(R) do
    for v ∈ N^-_{r-1}(x) do
        Set X' := X \ N^+_{r-1}(v);
        Calculate DIGRAPHBURNING(D, X', R \ {r});
        if the result is a map φ' then
            let φ be the map obtained from φ' by adding r ↦ v;
            return φ;
        end
    end
end
return NO;
```

Lemma 4.8. *Algorithm 3 solves* PARTIAL DIGRAPH BURNING WITH RANGES *in* $O(((k(D)+1)\max(R)|\operatorname{set}(R)|)^{|R|} \operatorname{poly}(n))$ *time.*

Proof. We first show the algorithm is correct. Note that each node in X has to be burned by some range. Hence, we can pick an arbitrary node $x \in X$, and try each possible assignment of a burning range $r \in \operatorname{set}(R)$ to a node $v \in V$ such that $x \in \mathcal{N}^+_{r-1}(v)$. If there exists a burning assignment, then one of these choices has to be part of such an assignment. For each of these choices, the algorithm checks whether the remaining problem $(D, X \setminus \mathcal{N}^+_{r-1}(v), R \setminus \{r\})$ has a solution. If it does, then this solution together with $r \mapsto v$ is a solution to the original problem. If no such solution exists for any choice, then the original problem is not solvable, and the algorithm correctly returns NO.

Now we consider the running time. For any fixed range r, the size of the neighbourhood $\mathcal{N}^-_{r-1}(x)$ is at most $(k(D)+1)\max(R)$. For each of the nodes in $\mathcal{N}^-_{r-1}(x)$, we have to try at most $|\operatorname{set}(R)|$ burning ranges. Branching like this, we have to go to depth $|R|$ in the search tree to try all options. Each of the recursive calls takes $\operatorname{poly}(n)$ time for computing the neighbourhoods and the new instances. Hence, the total running time is $O(((k(D)+1)\max(R)|\operatorname{set}(R)|)^{|R|}\operatorname{poly}(n))$. □

Theorem 4.9. *Algorithm 3 solves* Digraph Burning *in* $O(((k(D)+1)b^2)^b \operatorname{poly}(n))$ *time.*

Proof. A direct consequence of Lemma 4.8 when setting $X = V(D)$ and $R = [b]$, in which case $r = |R| = b$. □

Although the running time bound seems quite high, there may be heuristic improvements. For example in Line 1, where the algorithm chooses a node $x \in X$ arbitrarily. Of course, this is not always optimal: one can choose a node x to minimize the size of the neighbourhood $\mathcal{N}^-_{r-1}(x)$. This makes sure the number of branches is much lower than the theoretical bound of $(k(D)+1)\max(R)|R|$. For example, when x is a source, the number of branches is $|R|$.

5 Discussion

In this paper, we have studied the directed burning problem. We have given sharp upper bounds on the burning number for digraphs in general $(n-1)$, for weakly connected digraphs $(n-1)$, for single source DAGs $(\lceil\sqrt{2n+1/4}-1/2\rceil)$, and for strongly connected digraphs $(\lceil\sqrt{2n+1/4}-1/2\rceil)$. The simple proofs for these bounds contrasts sharply with the undirected case, where the question whether all graphs have burning number $\lceil\sqrt{n}\rceil$ remains open.

Then, we turned to the computational complexity of computing the burning number. We have shown that computing the burning number of a directed tree is NP-hard. This result may be surprising, as the direction on the trees seems to severely restrict the number of ways one can burn leaves. However, considering that graph burning is hard on undirected trees as well, the result might be less surprising. Another new result for trees is that computing the burning number of a directed tree is FPT with the burning number as parameter. A similar result has not been proven for undirected trees yet, so it would be interesting to see if our results can be leveraged in the undirected version of the problem.

As may be expected, directed graph burning gets harder when more general digraphs are considered: When the burning problem is considered on DAGs with the burning number as parameter, the complexity jumps to W[2]-complete, which we showed with reductions to and

from Set Cover. This implies there probably is no fixed-parameter algorithm for directed graph burning with the burning number as parameter. Whether there is such an algorithm for undirected graph burning is still an open problem [KR19]. Again, it might be interesting to try to use our techniques or results in the undirected case; either by finding parameterized reductions to and from Set Cover, or directly to and from DAG Burning.

Finally, we have shown that there is an fixed-parameter algorithm for directed graph burning with a parameter consisting of the burning number and the reticulation number, a parameter inspired by phylogenetics research. The algorithm uses recursion, which meant we had to solve the more general problem Partial Digraph Burning With Ranges. It is worth mentioning that this problem not only 'contains' Digraph Burning, but is general enough to 'contain' other graph covering problems as well.

Consider, for example, a directed version of Dominating Set where the question is the following: given a digraph D and a number k, can we choose a set $C \subseteq V(D)$ of size at most k such that each node v is either in C, or has an in-edge (c, v) for some node $c \in C$. Solving this question is equivalent to solving the instance $(D, V(D), \{2, 2, \ldots, 2, 2\})$ with k twos of Partial Digraph Burning With Ranges, because each chosen node accounts for itself and its out-neighbours in both problems. Therefore, reductions between these problems are easily given.

To really understand the computational complexity of the burning problem, it may be worth studying more restricted problems of Partial Digraph Burning With Ranges. For example, what happens if we look for the partial burning number of a DAG where we need to burn only the sinks? Or one could try bounding other graph parameters, such as the tree-length (or scan-width for DAGs, [BSW20]) used for undirected burning in [KMZ19], and look for approximation algorithms.

References

[BBJ+17] Stéphane Bessy, Anthony Bonato, Jeannette Janssen, Dieter Rautenbach, and Elham Roshanbin. Burning a graph is hard. *Discrete Applied Mathematics*, 232:73–87, 2017.

[BBJ+18] Stéphane Bessy, Anthony Bonato, Jeannette Janssen, Dieter Rautenbach, and Elham Roshanbin. Bounds on the burning number. *Discrete Applied Mathematics*, 235:16–22, 2018.

[BEKM19] Anthony Bonato, Sean English, Bill Kay, and Daniel Moghbel. Improved bounds for burning fence graphs. *arXiv preprint arXiv:1911.01342*, 2019.

[BFJ+12] Robert M Bond, Christopher J Fariss, Jason J Jones, Adam DI Kramer, Cameron Marlow, Jaime E Settle, and James H Fowler. A 61-million-person experiment in social influence and political mobilization. *Nature*, 489(7415):295, 2012.

[BJR14] Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. Burning a graph as a model of social contagion. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 13–22. Springer, 2014.

[BJR16] Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. How to burn a graph. *Internet Mathematics*, 12(1-2):85–100, 2016.

[BL19] Anthony Bonato and Thomas Lidbetter. Bounds on the burning numbers of spiders and path-forests. *Theoretical Computer Science*, 794:12–19, 2019.

[BSW20] Vincent Berry, Celine Scornavacca, and Mathias Weller. Scanning phylogenetic networks is np-hard. 2020.

[CFK+15] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

[HWW08] Heather Hulett, Todd G Will, and Gerhard J Woeginger. Multigraph realizations of degree sequences: Maximization is easy, minimization is hard. *Operations Research Letters*, 36(5):594–596, 2008.

[JJE+18] Remie Janssen, Mark Jones, Péter L Erdős, Leo Van Iersel, and Celine Scornavacca. Exploring the tiers of rooted phylogenetic network space using tail moves. *Bulletin of mathematical biology*, 80(8):2177–2208, 2018.

[KGH14] Adam DI Kramer, Jamie E Guillory, and Jeffrey T Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24):8788–8790, 2014.

[KMZ19] Shahin Kamali, Avery Miller, and Kenny Zhang. Burning two worlds: algorithms for burning dense and tree-like graphs. *arXiv preprint arXiv:1909.00530*, 2019.

[KR19] Anjeneya Swami Kare and I Vinod Reddy. Parameterized algorithms for graph burning problem. In *International Workshop on Combinatorial Algorithms*, pages 304–314. Springer, 2019.

[LL16] Max R Land and Linyuan Lu. An upper bound on the burning number of graphs. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 1–8. Springer, 2016.