Burn and Win

Pradeesha Ashok[1], Sayani Das[2], Lawqueen Kanesh[3], Saket Saurabh[2,4], Avi Tomar[1], and Shaily Verma[2]

[1] International Institute of Information Technology, Bangalore
{pradeesha,Avi.Tomar}@iiitb.ac.in
[2] The Institute of Mathematical Sciences, Chennai
{sayanidas,saket,shailyverma}@imsc.res.in
[3] Indian Institute of Technology Jodhpur
lawqueen@iitj.ac.in
[4] University of Bergen, Norway

Abstract. Given a graph G and an integer k, the Graph Burning problem asks whether the graph G can be burned in at most k rounds. Graph burning is a model for information spreading in a network, where we study how fast the information spreads in the network through its vertices. In each round, the fire is started at an unburned vertex, and fire spreads from every burned vertex to all its neighbors in the subsequent round burning all of them and so on. The minimum number of rounds required to burn the whole graph G is called the *burning number* of G. Graph Burning is NP-hard even for the union of disjoint paths. Moreover, Graph Burning is known to be W[1]-hard when parameterized by the burning number and para-NP-hard when parameterized by treewidth. In this paper, we prove the following results:

- In this paper, we give an explicit algorithm for the problem parameterized by treewidth, τ and k, that runs in time $k^{2\tau}4^k5^\tau n^{O(1)}$. This also gives an FPT algorithm for Graph Burning parameterized by burning number for apex-minor-free graphs.
- Y. Kobayashi and Y. Otachi [Algorithmica 2022] proved that the problem is FPT parameterized by distance to cographs and gave a double exponential time FPT algorithm parameterized by distance to split graphs. We improve these results partially and give an FPT algorithm for the problem parameterized by distance to cographs $\cap$ split graphs (threshold graphs) that runs in $2^{\mathcal{O}(t \ln t)}$ time.
- We design a kernel of exponential size for Graph Burning in trees.
- Furthermore, we give an exact algorithm to find the burning number of a graph that runs in time $2^n n^{\mathcal{O}(1)}$, where n is the number of vertices in the input graph.

Keywords: Burning number · fixed-parameter tractability · treewidth · threshold graphs· exact algorithm.

1 Introduction

Given a simple undirected graph $G = (V, E)$, the graph burning problem is defined as follows. Initially, at round $t = 0$, all the nodes are unburned. At each round $t \geq 1$, one new unburned vertex is chosen to burn, if such a node exists, and is called a fire source. When a node is burned, it remains burned until the end of the process. Once a node is burned in round t, its unburned neighbors become burned in round $t+1$. The process ends when there are no unburned vertices in the graph. The *burning number* of a graph G is the minimum number of rounds needed to burn the whole graph G, denoted by $b(G)$. The sources chosen in each round form a sequence of vertices called a *burning sequence* of the graph. Let $\{b_1, b_2, \cdots, b_k\}$ be a burning sequence of graph G. For $v \in V, N_k[v]$ denotes the set of all vertices within distance k from v, including v. Then, $\bigcup_{1\leq i\leq k} N_{k-i}[b_i] = V$.

Given a graph G and an integer k, the GRAPH BURNING problem asks if $b(G) \leq k$? This problem was first introduced by Bonato, Janssen, and Roshanbin [3,4,13]. For any graph G with radius r and diameter d, $\lceil (d+1)^{1/2} \rceil \leq b(G) \leq r+1$. Both bounds are tight, and paths achieve the lower bound.

The GRAPH BURNING is not only NP-complete on general graphs but for many restricted graph classes. It has been shown that GRAPH BURNING is NP-complete when restricted to trees of maximum degree 3, spider and path-forests [1]. It was also shown that this problem is NP-complete for caterpillars of maximum degree 3 [8,12]. In [7], authors have shown that GRAPH BURNING is NP-complete when restricted to interval graphs, permutation graphs, or disk graphs. Moreover, the GRAPH BURNING problem is known to be polynomial time solvable on cographs and split graphs [10].

The burning number has also been studied in directed graphs. Computing the burning number of a directed tree is NP-hard. Furthermore, the GRAPH BURNING problem is W[2]-complete for directed acyclic graphs [9]. For further information about GRAPH BURNING , the survey by Bonato [2] can be referred to.

The parameterized complexity of GRAPH BURNING was first studied by Kare and Reddy [10]. They showed that GRAPH BURNING on connected graphs is fixed-parameter tractable parameterized by distance to cluster graphs and by neighborhood diversity. In [11], the authors showed that GRAPH BURNING is fixed-parameter tractable when parameterized by the clique-width and the maximum diameter among all connected components, which also implies that GRAPH BURNING is fixed-parameter tractable parameterized by modular-width, by tree-depth, and by distance to cographs. They also showed that this problem is fixed-parameter tractable parameterized by distance to split graphs. It has also been shown that GRAPH BURNING parameterized by solution size, k, is W[2]-complete. The authors also showed that GRAPH BURNING parameterized by vertex cover number does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.

Our Results: In Section 2 we add all the necessary definitions. In Section 3, we use nice tree decomposition of G to give an FPT algorithm for GRAPH BURNING parameterized by treewidth and solution size. This result also implies that GRAPH BURNING parameterized by burning number is FPT on apex-minor-free graphs. In Section 4, we show that GRAPH BURNING is fixed-parameter tractable when parameterized by distance to cographs $\cap$ split graphs, also known as *threshold graphs*, which partially improve the results given in [11]. In Section 5, we design an exponential kernel for Graph Burning in trees. In Section 6, we give a non trivial exact algorithm for finding the burning number in general graphs.

2 Preliminaries

We consider the graph, $G = (V, E)$, to be simple, finite, and undirected throughout this paper. $G[V\backslash X]$ represents the subgraph of G induced by $V\backslash X$. $N_G(v)$ represents the set of neighbors of the vertex v in graph G. We simply use $N(v)$ if there is no ambiguity about the corresponding graph. $N_G[S] = \{u : u \in N_G(v), \forall v \in S\}$. For $v \in V$, $N_k[v]$ denotes the set of all vertices within distance k from v, including v itself. $N_1[v] = N[v]$, the closed neighborhood of v. For any pair of vertices $u, v \in V$, $dist_G(v, u)$ represents the length of the shortest path between vertices u and v in G. The set $\{1, 2, \cdots, n\}$ is denoted by $[n]$. For definitions related to parameterized complexity, refer the book by Cygan et al. [5].

A graph G is an *apex graph* if G can be made planar by removing a vertex. For a fixed apex graph H, a class of graphs $\mathcal{S}$ is *apex-minor-free* if every graph in $\mathcal{S}$ does not contain H as a minor. A *threshold graph* can be built from a single vertex by repeatedly performing the following operations.
(i) Add an isolated vertex.
(ii) Add a dominating vertex, i.e., add a vertex that is adjacent to every other vertex.
Thus for a threshold graph G, there exists an ordering of $V(G)$ such that any vertex is either adjacent to every vertex that appears before that in the ordering or is adjacent to none of them.

3 Parameterized by treewidth and burning number

We prove the following result in this section.

Theorem 1. Graph Burning *is FPT for apex-minor-free graphs when parameterized by burning number.*

To prove this result, we first give an FPT algorithm for Graph Burning parameterized by treewidth+burning number. For definitions and notations related to treewidth, refer the book by Cygan et al. [5].

Theorem 2. Graph Burning *admits an FPT algorithm that runs in* $k^{2\tau}4^k5^\tau n^{\mathcal{O}(1)}$ *time, when parameterized by the combined parameter, treewidth* τ *and burning number* k.

Proof. We use dynamic programming over a nice tree decomposition T of G. We shall assume a nice tree decomposition of G of width τ is given.

We use the following notation as in [5]. For every node t in the nice tree decomposition, we define G_t to be a subgraph of G where $G_t = (V_t, E_t = \{e : e \text{ is introduced in the subtree of } t\})$. V_t is the union of all vertices introduced in the bags of the subtree rooted at t. For a function $f : X \to Y$ and $\alpha \in Y$, we define a new function $f_{v\to\alpha} : X \cup \{v\} \to Y$ as follows:

$$f_{v\to\alpha}(x) = \begin{cases} f(x), & \text{when } x \neq v \\ \alpha, & \text{when } x = v \end{cases}$$

We define subproblems on every node $t \in V(T)$. We consider the partitioning of the bag X_t by a mapping $\Psi : X_t \to \{B, R, W\}$, where B, R, and W respectively represent assigning black, grey, and white colors to vertices. Intuitively, a black vertex represents a fire source, a grey vertex is not yet burned, and a white vertex is burned by another fire source through a path that is contained in G_t. Each vertex is further assigned two integer values by two functions $FS : X_t \to [k]$ and $D : X_t \to [k-1] \cup \{0\}$. For a vertex $v \in X_t$, $FS(v)$ intuitively stores the index of the fire source that will burn v, and $D(v)$ stores the distance between the fire source and v.

Let S be the set $\{*, \uparrow, \downarrow\}$. For every bag, we consider an array $\gamma \in S^k$. Here the entries in γ represent the location of each fire source with respect to the bag X_t. More precisely, for $1 \leq i \leq k$, $\gamma[i]$ is $*$ when the i-th fire source is in X_t, is $\downarrow$ when the i-th fire source is in $V_t \setminus X_t$ and $\uparrow$ otherwise.

For a tuple $f[t, \gamma, FS, D, \Psi]$, we define that a burning sequence of G *realizes* the tuple if the fire sources in the burning sequence match γ i.e., for $1 \leq i \leq k$, the i-th fire source is in X_t, $V_t \setminus X_t$ and $V \setminus V_t$ if $\gamma[i]$ is $*, \downarrow$ and $\uparrow$ respectively and the following conditions are met.

1. A black vertex $v \in X_t$ is part of the burning sequence at index $FS(v)$.
2. A white vertex $v \in X_t$ is burned by a fire source with index $FS(v)$ by a path of length $D(v)$ that lies entirely in G_t.
3. A grey vertex $v \in X_t$ is not burned in G_t.
4. For a vertex v in $V_t \setminus X_t$, v is either burned or there exists a path from v to a grey vertex $u \in X_t$ such that $FS(u) + D(u) + dist_G(u, v) \leq k$.

We now define a sub-problem $f[t, \gamma, FS, D, \Psi]$ that returns $True$ if and only if there exists a burning sequence that realizes $[t, \gamma, FS, D, \Psi]$.
From the above definition, it is easy to see that G admits a burning sequence of length k if and only if $f[t, \gamma, FS, D, \Psi]$, where t is the root node (and therefore empty), γ contains all entries as $\downarrow$ and FS, D, Ψ are null, returns $True$.

A tuple (t, FS, D, γ, Ψ) is *valid* if the following conditions hold for every vertex $v \in X_t$.

- $\Psi(v) = B$ if and only if $D(v) = 0$ and $\gamma[i] = *$, where $i = FS(v)$.
- $\Psi(v) = W$, if and only if $D(v) > 0$ and $\gamma[i] = *$ or $\gamma[i] = \downarrow$, where $i = FS(v)$.
- For all vertices v, $D(v) \leq k - FS(v)$.
- For all $1 \leq i \leq k$ such that $\gamma[i] = *$, there exists exactly one vertex v in X_t such that $FS(v) = i$ and $D(v) = 0$.

For an invalid tuple, $f[.]$, by default, returns $False$. We now define the values of $f[.]$ for different types of nodes in T.

Leaf Node: In this case, $X_t = \emptyset$. So $f[t, \gamma, FS, D, \Psi]$ returns $True$ if $\gamma[i] = \uparrow$, for all $1 \le i \le k$ and FS, D, Ψ are null functions. Otherwise, this returns $False$.

Introduce Vertex Node : Let v be the vertex being introduced and t' be the only child node of t such that $v \notin X_{t'}$ and $X_t = X_{t'} \cup \{v\}$.

$$f[t, \gamma, FS, D, \Psi] = \begin{cases} f[t', \gamma', FS_{|X_{t'}}, D_{|X_{t'}}, \Psi_{|X_{t'}}], & \text{if } \Psi(v) = B \\ f[t', \gamma, FS_{|X_{t'}}, D_{|X_{t'}}, \Psi_{|X_{t'}}], & \text{if } \Psi(v) = R \\ False, & \text{if } \Psi(v) = W \end{cases}$$

In the recurrence, γ' is the same as γ except that $\gamma'[FS(v)] = \uparrow$. The correctness of the recurrence follows from the fact that v is an isolated vertex in G_t and v is not present in $G_{t'}$.

Forget Vertex Node : Let t' be the only child node of t such that $\exists v \notin X_t$ and $X_{t'} = X_t \cup \{v\}$.

We now give the recurrence as follows.

$$f[t, \gamma, FS, D, \Psi] = \begin{cases} \bigvee\limits_{i:\ \gamma[i] = \downarrow} f[t', \gamma_{\gamma[i] \to *}, FS_{v \to i}, D_{v \to 0}, \Psi_{v \to B}] \\ \bigvee\limits_{\substack{i:\ \gamma[i] \neq \uparrow, \\ 1 \le j \le k-i}} f[t', \gamma, FS_{v \to i}, D_{v \to j}, \Psi_{v \to W}] \\ \bigvee\limits_{\substack{1 \le j \le k \\ \exists w \in X_t \text{ such that} \\ j \le k - FS(w), D(w) = j - dist_G(v,w) \\ \text{and } \Psi(w) = R}} f[t', \gamma, FS_{v \to FS(w)}, D_{v \to j}, \Psi_{v \to R}] \end{cases}$$

In the last case, we consider the case where v is burned by a path P that lies outside G_t, at least partially. The feasibility of P is tracked by a vertex $w \in X_t$ that is closer to the fire source in P.

Lemma 1. *[*] The recurrence for Forget Vertex Node is correct.*

Introduce Edge Node: Let t be an introduce edge node with child node t' and let (u, v) be the edge introduced at t. We compute the value of f based on the following cases.

1. If $\Psi(u) = \Psi(v) = R$, set $f[t, \gamma, FS, D, \Psi] = f[t', \gamma, FS, D, \Psi]$
2 If $FS(u) + D(u) = FS(v) + D(v)$, set $f[t, \gamma, FS, D, \Psi] = f[t', \gamma, FS, D, \Psi]$
3 If $FS(u) + D(u) + 1 = FS(v) + D(v)$ and $FS(u) \neq FS(v)$, set $f[t, \gamma, FS, D, \Psi] = f[t', \gamma, FS, D, \Psi]$
4 If $\Psi(v) \in \{B, W\}, \Psi(u) = W, FS(u) = FS(v)$ and $D(u) = D(v) + 1$, then set $f[t, \gamma, FS, D, \Psi] = f[t', \gamma, FS, D, \Psi_{u \to R}] \bigvee f[t', \gamma, FS, D, \Psi]$
5 For all other cases, set $f[t, \gamma, FS, D, \Psi] = False$

Lemma 2. *[*] The recurrence for Introduce Edge Node is correct.*

[4] Proofs of results that are marked with [*] are omitted due to the space constraint.

Join Node: Let t be a join node and t_1, t_2 be the child nodes of t such that $X_t = X_{t_1} = X_{t_2}$. We call tuples $[t_1, \gamma_1, FS, D, \Psi_1]$ and $[t_2, \gamma_2, FS, D, \Psi_2]$ as $[t, \gamma, FS, D, \Psi]$-consistent if the following conditions hold.
For all values of i, $1 \leq i \leq k$,
if $\gamma[i] = *$ then $\gamma_1[i] = \gamma_2[i] = *$
if $\gamma[i] =\uparrow$ then $\gamma_1[i] = \gamma_2[i] =\uparrow$
if $\gamma[i] =\downarrow$ then either $\gamma_1[i] =\downarrow$, $\gamma_2[i] =\uparrow$ or $\gamma_1[i] =\uparrow$, $\gamma_2[i] =\downarrow$
For all $v \in X_t$,
if $\Psi(v) = B$ then $\Psi_1(v) = \Psi_2(v) = B$
if $\Psi(v) = R$ then $\Psi_1(v) = \Psi_2(v) = R$
if $\Psi(v) = W$ then $(\Psi_1(v), \Psi_2(v)) \in \{(W, W), (W, R), (R, W)\}$

We give a short intuition for the above. The cases where $\gamma[i] = *$ and $\gamma[i] =\uparrow$ are easy to see. When $\gamma[i] =\downarrow$, the i-th fire source is below the bag. By the property of the tree decomposition, $V_{t_1} \setminus X_t$ and $V_{t_2} \setminus X_t$ are disjoint. Therefore, exactly one of $\gamma_1[i]$ and $\gamma_2[i]$ is set to $\downarrow$. Similarly, $\Psi(v) = B$ and $\Psi(v) = R$ are easy to see. When $\Psi(v) = W$, the vertex is already burned below. Here again, there are two possibilities: v is burned in exactly one of G_{t_1} and G_{t_2} and v is burned in both of them (possibly by different paths). Therefore, $(\Psi_1(v), \Psi_2(v)) \in \{(W, W), (W, R), (R, W)\}$.

Then, the recurrence is as follows, where the OR operations are done over all pairs of tuples, which are $[t, \gamma, FS, D, \Psi]$-consistent.

$$f[t, \gamma, FS, D, \Psi] = \bigvee (f[t_1, \gamma_1, FS, D, \Psi_1] \wedge f[t_2, \gamma_2, FS, D, \Psi_2])$$

Lemma 3. *[*] The recurrence for Join node is correct.*

Running Time : Note that we can compute each entry for $f[\cdot]$ in time $k^{2\tau} 3^k 3^\tau n^{\mathcal{O}(1)}$, except for join nodes. For join nodes, we require extra time as we are computing over all possible consistent tuples. Let $(\gamma, \gamma_1, \gamma_2)$ and (Ψ, Ψ_1, Ψ_2) be such that $(t_1, \gamma_1, FS, D, \Psi_1)$ and $(t_2, \gamma_2, FS, D, \Psi_2)$ are (t, γ, FS, D, Ψ) consistent then, $\forall i \in [k]$, $(\gamma[i], \gamma_1[i], \gamma_2[i]) \in \{(*, *, *), (\uparrow, \uparrow, \uparrow), (\downarrow, \downarrow, \uparrow), (\downarrow, \uparrow, \downarrow)\}$ and $\forall v \in X_t$, $(\Psi[v], \Psi_1[v], \Psi_2[v]) \in \{(B, B, B), (R, R, R), (W, W, W), (W, W, R), (W, R, W)\}$. Therefore, the total number of consistent tuples over all join nodes is upper bounded by $k^{2\tau} 4^k 5^\tau n^{\mathcal{O}(1)}$. Hence the running time of the algorithm can be bounded by $k^{2\tau} 4^k 5^\tau n^{\mathcal{O}(1)}$.

□

For apex-minor-free graphs, the treewidth is bounded by the diameter of the graph, as shown in [6]. It has been established that the diameter of a graph is bounded by a function of the burning number of the graph [4]. As a result, the treewidth of apex-minor-free graphs is bounded by a function of the burning number. This observation, along with Theorem 2, proves Theorem 1.

4 Parameterized by distance to threshold graphs

In this section, we give an FPT algorithm for Graph Burning parameterized by the distance to threshold graphs. Recall that the problem is known to be in FPT; the paper [11] shows that GRAPH BURNING is FPT parameterized by distance to cographs and gives a double-exponential time FPT algorithm when parameterized by distance to split graphs. Since both these parameters are smaller than the distance to threshold graphs that are precisely the graphs in graph-class cographs $\cap$ split, these results imply fixed-parameter tractability when the distance to threshold graphs is a parameter. Here, we give an FPT algorithm that runs in single-exponential time, which improves the previously known algorithms. We will consider a connected graph $G = (V, E)$ and a subset $X \subseteq V$ with $|X| = t$, such that the induced subgraph $G[V \setminus X]$ is a threshold graph. It is assumed that the set X is given as part of the input.

Theorem 3. GRAPH BURNING *on G can be solved in time $t^{2t}n^{\mathcal{O}(1)}$.*

Proof. Since $G[V \setminus X]$ is a threshold graph, there exists an ordering Π of vertices such that every vertex is either a dominating vertex or an isolated vertex for the vertices preceding it in Π. Let $v_d \in V \setminus X$ be the last dominating vertex in Π and (D, I) be the partition of $V \setminus X$ such that D is a set that contains the vertices in Π till the vertex v_d and I is the set containing all remaining vertices. Thus, in $G[V \setminus X]$, I is a maximal set of isolated vertices.

We observe that G can be burned in at most $t + 3$ steps. In at most t steps, we set each vertex in X as a fire source. Since every vertex in I has at least one neighbor in X, all vertices in I are burned in at most $t + 1$ steps. Similarly, since at least one vertex from D has a neighbor in X and D induces a graph of diameter 2, every vertex in D is burned in at most $t + 3$ steps. Therefore, we assume $k < t + 3$ for the rest of the proof.

For a valid burning sequence of length k of the graph G, for every vertex $v \in V$, let $(fs(v), d(v))$ be a pair where $1 \leq fs(v) \leq k$ and $0 \leq d(v) \leq k - fs(v)$, such that $fs(v)$ is the index of the fire source that burns the vertex v and $d(v)$ is the distance between that fire source and v. It also implies that v is going to burn at the $(fs(v) + d(v))$-th round. When two fire sources can simultaneously burn v in the same round, $fs(v)$ is assigned the index of the earlier fire source. The basic idea of the algorithm is to guess the pair $(fs(v), d(v))$ for every $v \in X$ and then extend this guess into a valid burning sequence for G in polynomial time.

Consider two families of functions $\mathcal{F} = \{fs : X \to [k]\}$ and $\mathcal{D} = \{d : X \to \{0\} \cup [k-1]\}$ on X. A pair $(fs, d) \in \mathcal{F} \times \mathcal{D}$ corresponds to a *guess* that decides how a vertex in X is burnt. We further extend this to an *augmented guess* by guessing the fire sources in D. We make the following observation based on the fact that every pair of vertices in D is at a distance at most two since v_d is adjacent to every vertex in D.

Observation 1 *There can be at most two fire sources in D. Moreover, if there are two fire sources in D, they are consecutive in the burning sequence.*

Thus, an augmented guess can be considered as extending the domain of the functions fs and d to $X \cup D'$ where $D' \subseteq D$ and $0 \leq |D| \leq 2$. Note that, for all $v \in D'$, $d(v) = 0$ since we are only guessing the fire sources in D.

An augmented guess is considered *valid* if the following conditions are true.
1. For all v in the domain of the functions, $0 \leq d(v) \leq k - fs(v)$.
2. For all $1 \leq i \leq k$, there exists at most one vertex v such that $fs(v) = i$ and $d(v) = 0$.
3. For all u, v in the domain of the functions, $|(fs(u)+d(u)) - (fs(v)+d(v))| \leq dist_G(u,v)$.

Algorithm 1 gives the procedure to extend a valid augmented guess to a valid burning sequence by identifying the firesources in I. Specifically, Algorithm 1 takes a valid augmented guess as input and returns YES if it can be extended to a burning sequence of length k.

Algorithm 1

for $1 \leq i < k$ such that $\nexists v$ such that $fs(v) = i$ and $d(v) = 0$ **do**
    $X_i = \{v \in X : fs(v) = i, d(v) = 1\}$
    if X_i is not empty **then**
        $I_i = \{u \in I : X_i \subseteq N(u)\}$; $F_i = I_i$
        for $u \in I_i$ **do**
            if there exists $w \in N(u) \setminus X_i$ such that $(fs(w)+d(w) = i+2) \vee (fs(w)+d(w) = i-1) \vee (fs(w) = i+1 \wedge d(w) = 0)$ **then**
                Delete u from F_i.
        if $(i \neq k-1)$ **then**
            if F_i is not empty **then**
                Let v be an arbitrary vertex in F_i. Set v as the i-th fire source.
            else return NO.
        else
            $F' = \{u \in F_i : \forall w \in N(u) \setminus X_i, fs(w) + d(w) = k\}$
            if $|F'| > 2$ **then** return NO.
            else if $F' \neq \emptyset$ **then**
                set an arbitrary vertex in F' as the i-th fire source.
            else
                set an arbitrary vertex in F_i as the i-th fire source.
if CheckValidity() = True **then**
    Return YES.
else return NO.

Lemma 4. *Algorithm 1 is correct.*

Proof. It is enough to prove that the fire sources in I are correctly identified.

For every $1 \leq i < k$ such that the i-th fire source is not "discovered" yet, we consider the vertices in X_i. $I_i \subseteq I$ is the set of vertices adjacent to every vertex in X_i. The i-th fire source, if exists, should be one of the vertices from I_i. The algorithm further considers a set F_i that is obtained from I_i by filtering out vertices that cannot be the i-th fire source. Specifically, the set F_i contains vertices v such that, $X_i \subseteq N(v)$ and for all $w \in N(v) \setminus X_i$, $i \leq fs(w) + d(w) \leq i+1$. We shall show that a vertex outside this set cannot be the i-th fire source.

Let $v \in I_i$ and $w \in N(v) \setminus X_i$. For all $u \in X_i$, $fs(u) + d(u) = i + 1$ and $dist_G(u, w) \leq 2$, since they have a common neighbor v. By the constraint given in the definition of a valid augmented guess, $i - 1 \leq fs(w) + d(w) \leq i + 3$. If $fs(w) + d(w) \geq i + 2$, then $fs(v) + d(v) \geq i + 1$ and v is not the i-th fire source. Also, if $fs(w) + d(w) = i - 1$, then $fs(w) < i$ and v is not the i-th fire source since an earlier fire source can burn v in the i-th round. Further, if v is the i-th fire source, then the case where $fs(w) = i + 1$ and $d(w) = 0$ is not possible since w will be burned by v in the $(i+1)$-th round. Note that, by definition, if a vertex v can be burned simultaneously by two different fire sources, then $fs(v)$ is assigned the index of the earlier fire source. Thus, the i-th fire source, if exists, should belong to the set F_i.

Assume $i < k - 1$. Let v_1 and v_2 be arbitrary vertices in the set F_i and let v_1 be the i-th fire source in a burning sequence γ of G. Now, we will prove that a sequence γ' obtained by replacing v_1 with v_2 as the i-th fire source in γ is also a valid burning sequence. Note that, in γ, X_i is exactly the set of vertices that are burned by v_1 in the $(i+1)$-th round since any other neighbor of v_1 is burned in the i-th or $(i+1)$-th round by a different fire source. Now, since $X_i \subseteq N(v_2)$, X_i is burned in the $(i+1)$-th round by γ' also. Also, any other neighbor of v_1 or v_2 is burned in the i-th or $(i + 1)$-th round by a fire source that is not the i-th fire source, which also ensures v_1 gets burned before the k-th round. Hence, if there exists a fire source in I_i, then any arbitrary vertex in I_i can be the fire source.

Assume $i = k - 1$. Now we consider the subset $F' = \{u \in F_i : \forall w \in N(u) \setminus X_i, fs(w) + d(w) = k\}$ of F_i. A vertex in F' can be burned only if it is the k-th or $(k - 1)$-th fire source. Hence, we return No if $|F'| > 2$. Otherwise an arbitrary vertex is set as the $(k - 1)$-th fire source.

Finally, once the fire sources are set, we can check the validity of the burning sequence in polynomial time. □

Extending a valid augmented guess can be done in polynomial time. Thus the running time of the algorithm is determined by the number of valid augmented guesses which is bounded by $t^{2t}n^{O(1)}$. □

5 A Kernel for Trees

In this section, we design a kernel for the Graph Burning problem on trees. Let (T, k) be the input instance of Graph Burning where T is a tree. First, we arbitrarily choose a vertex $r \in V(T)$ to be the *root* and make a rooted tree. Let $L_0, L_1, L_2, \ldots, L_p$ be the levels of tree T rooted at r, where $L_0 = \{r\}$. To give a kernel, we give a marking procedure to mark some vertices in each level and show that we can remove the unmarked vertices. In bottom-up fashion, starting from the last level, in each iteration, we mark at most $k + 1$ children for each vertex in the level, and we remove the subtree rooted at any unmarked children. Observe that while doing that, we maintain the connectedness of the tree. We show that the removal of subtrees rooted on unmarked vertices does not affect the burning number of the tree. Let T_z be the subtree rooted at a vertex z and M_i be the set of marked vertices at level L_i.

Marking Procedure: For all $i \in [p]$, we initialise $M_i = \emptyset$. For a fixed $i,\ i \in [p]$, do as follows: For each vertex $x \in L_{p-i}$, mark at most $k+1$ children of x such that the depth of the subtrees rooted on marked children is highest and add them into M_{p-i+1}.

Reduction Rule 1 *If $z \in L_{p-i}$ such that $z \notin M_{p-i}$, then remove the subtree T_z.*

Lemma 5. *The Reduction Rule 1 is safe.*

Proof. To show the safeness of Reduction Rule 1, we show that (T, k) is a Yes-instance of Graph Burning if and only if $(T - T_z, k)$ is a Yes-instance.

For the forward direction, assume that (T, k) is a Yes-instance. Note that $T - T_z$ is a tree. Let $(b_1, b_2, \ldots, b_k)$ be a burning sequence for T. If any of the b_i belongs to T_z, then we replace b_i by placing a fire source on the first unburned ancestor of z in $T - T_z$. Therefore, we can have a burning sequence of size k. Hence, $(T - T_z, k)$ is a Yes-instance.

For the backward direction, assume that $(T - T_z, k)$ is a Yes-instance, and we need to show that (T, k) is a Yes-instance. Let $P(z)$ be the parent of vertex z in T. Suppose that $x_1, x_2, \ldots, x_{k+1}$ be the set of marked children (neighbors) of $P(z)$. We have to show that any vertex u in the subtree T_z can also be burned by the same burning sequence.

Since the burning number of $T - T_z$ is k, there is at least one marked child x_j of $P(z)$ such that there is no fire source placed in the subtree T_{x_j}. Observe that there exists a vertex u' in the subtree T_{x_j} such that the distances $d(u, P(z))$ and $d(u', P(z))$ are the same since the height of T_{x_j} is at least the height of subtree T_z by the marking procedure. Let the vertex u' get burned by a fire source s. Note that the fire source s is either placed on some ancestor of x_j or some subtree rooted at a sibling of x_j. In both cases, the s-u' path contains the vertex $P(z)$. Since $d(u, P(z)) = d(u', P(z))$, the vertex u also gets burned by the same fire source in the same round. Thus, every vertex in T_z can be burned by some fire source from the same burning sequence as $T \setminus T_z$. Hence, (T, k) is also a Yes-instance. □

Iteratively, for each fixed value of $i,\ i \in [p]$ (starting from $i = 1$), We apply the marking procedure once, and the Reduction Rule 1 exhaustively for each unmarked vertex. After the last iteration ($i = p$), we get a tree T'. Observe that we can complete the marking procedure in polynomial time, and the Reduction Rule 1 will be applied at most n times. Therefore, we can obtain the kernel T' in polynomial time.

Kernel Size: Note that the obtained tree T' is a $(k+1)$-ary tree. Let b_1 be the first fire source in T'; then we know that b_1 will burn vertices up to $(k-1)$ distance. Therefore, we count the maximum number of vertices b_1 can burn.

First, note that b_1 will burn the vertices in the subtree rooted at b_1 up to height $k-1$. Let n_0 be the number of vertices in the subtree rooted at b_1. It follows that $n_0 \leq \frac{(k+1)^{k-1}-1}{k}$, that is, $n_0 \leq (k+1)^{k-1}$. Note that b_1 also burns the vertices on the path between b_1 to root r up to distance $k-1$ and the vertices rooted on these vertices. Let $P = b_1 v_1 v_2 \ldots v_{k-1} \ldots r$ be b_1-r path in T'. Then b_1 also burns the vertices in the subtree rooted at v_i, say T_{v_i}, upto height $k-1-i$, where $i \in [k-1]$. Let $n_i = |V(T_{v_i})|$. Therefore, for any $i \in [k-1]$, $n_i \leq (k+1)^{k-1}$ as $n_i < n_0$. Thus, the total number of vertices b_1 can burn is at most $(k+1)^k$. Since each fire source b_i can burn only fewer vertices than the maximum number of vertices that can be burned by source b_1, the total number of vertices any burning sequence of size k can burn is at most $(k+1)^{k+1}$.

Therefore, if there are more than $(k+1)^{k+1}$ vertices in T', then we can conclude that (T, k) is a No-instance of Graph Burning problem. This gives us the following result.

Theorem 4. *In trees,* Graph Burning *admits a kernel of size* $(k+1)^{k+1}$.

6 Exact Algorithm

In this section, we design an exact algorithm for the Graph Burning problem. Here, we reduce the Graph Burning problem to the shortest path problem in a *configuration graph*.

Construction of a Configuration graph: Given a graph $G = (V, E)$, we construct a directed graph $G' = (V', E')$ as follows:
(i) For each set $S \subseteq V(G)$, add a vertex $x_S \in V'$.
(ii) For each pair of vertices $x_S, x_{S'} \in V'$ such that there exists a vertex $w \notin S$ and $N_G[S] \cup \{w\} = S'$, add an arc from x_S to $x_{S'}$.
We call the graph G' as the configuration graph of G.

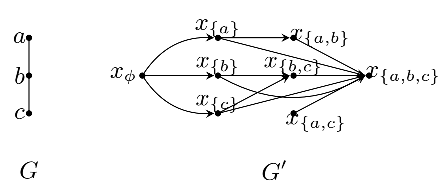

Fig. 1. An illustration of G', the configuration graph of G.

Figure 1 shows an example of a configuration graph.

We have constructed G' in such a way that a shortest path between the vertices x_S and x_T, where $S = \emptyset$ and $T = V(G)$, gives a burning sequence for the original graph G. The following result proves this fact.

Lemma 6. *[*] Let $G' = (V', E')$ be the configuration graph of a given graph G and $S = \emptyset$ and $T = V(G)$. There exists a path of length k between the vertices x_S and x_T in G' if and only if there is a burning sequence for G of size k.*

Lemma 6 shows that a shortest path between x_S and x_T in G' gives the burning sequence for graph G with minimum length. Thus, we can find a minimum size burning sequence in two steps:
(i) We construct a configuration graph G' from G.
(ii) Find a shortest path between the vertices x_S and x_T in G', where $S = \emptyset$ and $T = V(G)$.

Observe that we can construct the graph G' in $(|V(G')| + |E(G')|)$-time and find a shortest path in G' in $\mathcal{O}(|V(G')| + |E(G')|)$-time. We know that $|V(G')| = 2^n$ and note that the total degree (in-degree+out-degree) of each vertex in G' is at most n. Therefore, $|E(G')| \leq n \cdot 2^n$. Therefore, the total running time of the algorithm is $2^n n^{\mathcal{O}(1)}$. Thus, we have proved the next theorem.

Theorem 5. *Given a graph G, the burning number of G can be computed in $2^n n^{\mathcal{O}(1)}$ time, where n is the number of vertices in G.*

References

1. S. Bessy, A. Bonato, J. Janssen, D. Rautenbach, and E. Roshanbin. Burning a graph is hard. *Discrete Applied Mathematics*, 232:73–87, 2017.
2. A. Bonato. A survey of graph burning. *arXiv preprint arXiv:2009.10642*, 2020.
3. A. Bonato, J. Janssen, and E. 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.
4. A. Bonato, J. Janssen, and E. Roshanbin. How to burn a graph. *Internet Mathematics*, 12(1-2):85–100, 2016.
5. M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.
6. D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27:275–291, 2000.
7. A. T. Gupta, S. A. Lokhande, and K. Mondal. Np-completeness results for graph burning on geometric graphs. *arXiv preprint arXiv:2003.07746*, 2020.
8. M. Hiller, A. M. Koster, and E. Triesch. On the burning number of p-caterpillars. In *Graphs and Combinatorial Optimization: from Theory to Applications*, pages 145–156. Springer, 2021.
9. R. Janssen. The burning number of directed graphs: Bounds and computational complexity. *arXiv preprint arXiv:2001.03381*, 2020.
10. A. S. Kare and I. Vinod Reddy. Parameterized algorithms for graph burning problem. In *International Workshop on Combinatorial Algorithms*, pages 304–314. Springer, 2019.
11. Y. Kobayashi and Y. Otachi. Parameterized complexity of graph burning. *Algorithmica*, pages 1–15, 2022.
12. M. R. Land and L. 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.
13. E. Roshanbin. *Burning a graph as a model for the spread of social contagion*. PhD thesis, Dalhousie Univertsity, 2016.