Improved Algorithms for Burning Graph Families

by

Mohammadmasoud Shabanijou

A thesis submitted in conformity with
the requirements for the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada

Abstract

In the graph burning problem, the input is an undirected, unweighted, finite, and simple graph. A fire starts at a vertex at each round, and when a particular vertex is burned, all of its adjacent vertices are burned in the next round. We assume that the rounds are synchronous and discrete. At each round, one new fire can start in a new vertex. The goal is to select the vertices at which fires are started so that all vertices are burned as quickly as possible. Finding an optimal burning sequence is known to be NP-hard and the problem remains NP-hard even for simple graph families such as trees or a set of disjoint paths. The best approximation algorithm for general graphs has an approximation factor of 3.

In this thesis, we investigate this problem on different graph families of sparse graphs, and in particular, we look at cactus graphs and melon graphs and study algorithms that aim to burn these graphs as quickly as possible. For both graph families, we show that the problem is NP-complete, and provide approximation algorithms with approximation factors smaller than 3.

Contents

List of Figures

Acknowledgement

I would like to express my sincere gratitude to my parents without whom I would have never reached this point...

I would like to thank Leila for supporting me through this journey. Her kind words and kindness always filled my heart with happiness. I am so lucky to have her in my life.

I would like to thank my friends Shahriar, Amir Winnnipegi, Matt, David, Khashayar, Soheil, Ali Eghabl, Pouya, Dr. Ramun kakhal, Ali yari, Behnam (Amoo Karam), Amoo Asad, Saeed, Behdad, Amir Kheiri, Alireza Sadeghi, Ali tavas, Amir Salimi, Mamad, Erfan Sb, Ali khosarvi (Ar7), Ali ghobad, and all others who supported me through this journey.

1 Introduction

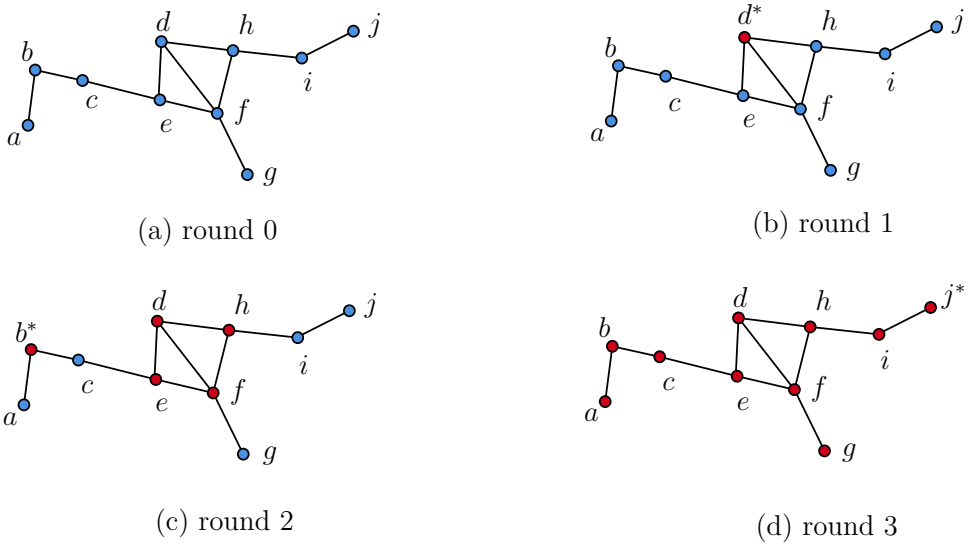
1.1 The burning problem

There has been considerable effort in the past few years to develop models for the analysis of social contagion in networks [5]. The literature in this area analyzes a network and investigates how fast a meme or gossip could spread throughout the network. The graph burning problem is one example of such abstractions that have been studied in recent years [46]. In the graph burning problem, the main goal is to "burn" vertices of a graph as quickly as possible. The burning number [46] of a graph indicates the smallest number of rounds needed to spread a fire throughout the graph. In the burning problem, our input is a simple graph that is undirected, unweighted, and finite. We consider discrete rounds to model the time, and the data (that is, the fire) is passed between the nodes in discrete and synchronous rounds [25].

In the graph burning problem, initially, all vertices are unburned. At the beginning of each round, one vertex is selected by the algorithm and is set on fire. At each consequent round, all nodes that are adjacent to a burning vertex are also burned [5]. Note that, at the beginning of every round, the algorithm can choose a new node to burn, that is, a new fire is started, while the existing ones continue to spread. The algorithm will continue until all vertices of the graph are burned [5].

To give some examples on the burning procedure, burning a complete graph takes only 2 rounds. Since all of the vertices are connected, any vertex that is burned at round 1 causes all its adjacent vertices to burn at round 2 (there is no need to start a new fire at round 2). Similarly, burning a complete bipartite graph $G = (A, B)$

requires only 3 rounds. A fire starting at round 1 at one of the vertices in A spreads to all vertices on the other side B at round 2, and by round 3 all vertices are burned (again, one fire is sufficient to burn the graph as quickly as possible in this case). Figure 1 provides an illustration of the burning process.



(a) round 0 (b) round 1

(c) round 2 (d) round 3

Figure 1: An illustration of the burning process. Blue vertices indicate unburned vertices and red vertices are burned ones. Initially (at time 0), all vertices are unburned. At round 1, an algorithm burns vertex d. At round 2, all vertices adjacent to vertex d are burned while the algorithms starts a new fire at b. At round 3, a new fire starts at j^* while old fires extend to the neighboring vertices. By the end of round 3, the burning process completes and all vertices are burned.

Approximation algorithms

In this thesis, we consider designing approximation algorithms for the burning problem, which is the common approach for finding an algorithmic solution to NP-complete problems. An approximation algorithm is a polynomial-time algorithm

that provides a solution that is "close" to the optimal solution. In the context of the burning problem, an approximation algorithm is an algorithm A that for any graph G outputs burning schemes that complete within $c \cdot bn(G)$ rounds, where $bn(G)$ is the burning number of G. Here c is called the *absolute approximation factor* of algorithm A. In addition, an algorithm might output schemes that complete in $c' \cdot bn(G) + d$, where $d = o(bn(G))$. In this case, c' is called the asymptotic approximation scheme of G. In this thesis, for simplicity, we refer to the asymptotic approximation factor as *approximation factor*.

Polynomial-Time Approximation Schemes (PTASs)

An approximation algorithm is said to be a polynomial-time approximation scheme, if for any $\epsilon > 0$, it provides an approximation factor of $1 + \epsilon$ and runs in polynomial time with respect to n (the size of graph), but possibly exponential time with respect to $1/\epsilon$. If the time complexity of the algorithm is polynomial to both n and $1/\epsilon$, it is said to be a fully-polynomial time approximation scheme (FPTAS). Finally, an Asymptotic Polynomial-Time Approximation Scheme (APTAS) is an algorithm that accepts a parameter ϵ and has an asymptotic approximation ratio of $(1 + \epsilon)$ [28].

1.2 Contributions

The burning problem is known to be a computationally hard problem [1]. In particular, the problem is NP-complete for very basic graph families such as path forests (a set of disjoint paths) and trees. The best existing approximation algorithm for burning general graphs has an approximation factor of 3 [2, 5]. Better approximation algorithms exist for certain graph families, e.g., trees [5] and graphs of bounded treelength, e.g., interval graphs [25].

In this thesis, we study the burning problem for two simple families of sparse

graphs, that is, cactus graphs, and melon graphs. A cactus graph is a graph in which every edge belongs to exactly one cycle. A melon graph is a graph formed by connecting endpoints of a set of paths to two distinguished vertices (poles). See Figure 2 for examples of cactus and melon graphs.

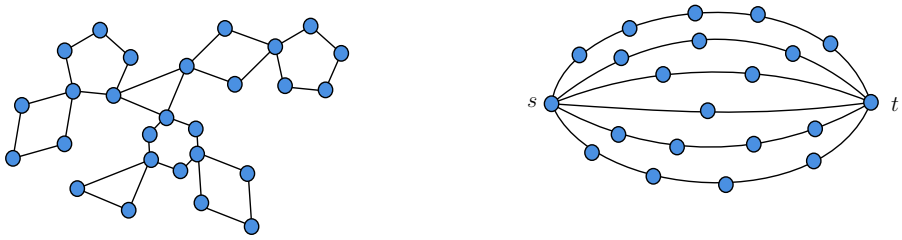


Figure 2: An example of a cactus graph (left) and a melon graph (right).

Our contributions in this thesis are as follows:

- We observe that the existing results for the hardness of the burning problem for graph families do not extend to the two families of graphs studied in this thesis. For example, cactus or melon graphs do not generalize trees and/or path forests and hence we cannot conclude the hardness of the problem from the hardness of burning trees and path forests. Similarly, the existing approximation algorithms for graph families do not extend to cactus and melon graphs. For example, the known polynomial-time approximation scheme for burning path forests [1] or the 2-approximation algorithm for burning trees [5] do not extend to cactus and melon graphs.

- For cactus graphs, we use a reduction from burning path forests to show it is NP-hard to burn a cactus graph. This shows that approximation algorithms are needed for burning cactus graphs.

- For a specific family of cactus graphs, known as necklace graphs (formed by a "path of uniform cycles"), we propose an algorithm that burns a necklace of n vertices in $\sqrt{n} + o(\sqrt{n})$.

- For the general family of cactus graphs, we provide an algorithm with an approximation factor of at most 2.5. This is an improvement over the approximation factor of 3 known for general graphs.

- We show that the burning problem is NP-complete in melon graphs, and provide an approximation algorithm with an approximation factor of 2 for burning melon graphs that runs in $O(n \log n)$. We also provide a (slower) polynomial-time approximation scheme for burning melon graphs.

2 Literature Review

In this chapter, we review the previous work on the burning problem and some of its related problems.

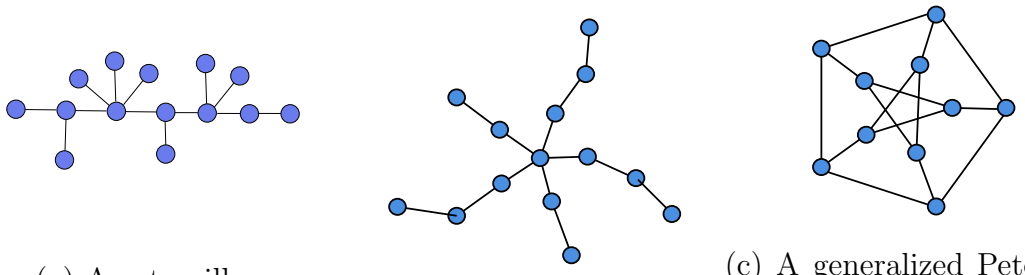
2.1 Burning number

An introduction to the graph burning problem in terms of social contagion was presented by Bonato et al. [4, 3]. Bonato et al. [3] proved that $2\lceil\sqrt{n}\rceil - 1$ is an upper bound for the burning number in connected graphs [3]. They also made a conjecture that, in any connected graph, the burning number is at most $\lceil\sqrt{n}\rceil$. It is rather easy to observe that this conjecture holds for paths. Mitsche et al. [33, 34] proved the conjecture for other graph families. They also studied a different model in which the selection of the burning sequence follows a probabilistic rule. Land and Lu [31] improved the upper bound of [3] for the burning number of connected graphs to $\lceil\frac{\sqrt{6}}{2}\sqrt{n}\rceil \approx 1.224\sqrt{n} + O(1)$. Bonato and Kamali [6] were able to further improve this upper bound and provide the best existing upper bound for the burning number which is $\sqrt{(4/3)n} + O(1) \approx 1.154\sqrt{n} + O(1)$.

2.2 Burning graph families

Liu et al. [32] studied the burning number of caterpillars and showed the burning conjecture (see Section 2.1) holds for caterpillars. A caterpillar graph is a tree in which there is a "backbone path" of vertices such that every vertex either belongs to the backbone path or is adjacent to exactly one vertex in the backbone (see Fig-

ure 3). Bonato et al. [22] investigated the problem in path forests and spiders (see Figure 3). A spider graph is a tree in which at most one vertex has a degree of more than two. Sim et al.[44] improved the bound for the burning number of generalized Petersen graphs. A Petersen graph is a graph formed by connecting the vertices of a regular polygon to the corresponding vertices of a star polygon (see Figure 3 for a Patersen graph with polygons of size $k = 5$). The burning number of some other families and some products of graphs is studied in [24]. Other works in this area include research on the burning number in random geometric graphs [33] and binomial random graphs [33]. Kamali et al. [25] presented an algorithm that burns any connected n-vertex graphs with the minimum degree δ in at most $\lceil\sqrt{\frac{24n}{\delta+1}}\rceil$ rounds. In particular, this algorithm proves the burning conjecture for graphs of minimum degree $\delta \geq 23$.



(a) A caterpillar.

(b) A spider graph.

(c) A generalized Petersen graph.

Figure 3: Examples caterpillar, spider, and Petersen graphs

models as well as graphs $G(d)$ [17].

2.3 Computational complexity

Bonato et al. [1] have proved that it is NP-hard to find a burning sequence that completes the burning in the minimum number of rounds. This hardness result holds even for basic graph families such as spider graphs, and path forests. Mondal et al. [35] have claimed that the burning problem is APX-hard.

Bonato and Kamali [5] investigated some approximation algorithms for this problem and introduced a general approximation algorithm for burning any graph G with the approximation ratio of 3. In addition, they also provided an algorithm with the approximation ratio of 2 for burning trees and polynomial-time approximation schemes (PTAS) for path-forests.

A graph G has *treelength* at most k if there is a tree decomposition of such that the maximum distance between vertices in each bag of the tree decomposition is at most k [15]. Note that there is no direct connection between the treewidth [43] and treelength of a graph. For example, a cycle formed by n vertices has treewidth 2 and treelength $\Theta(n)$, while a complete graph has treewidth $n-1$ and treelength 1. The concept of pathlength is defined similarly when the tree decomposition is indeed a path decomposition. In particular, a graph has pathlength 1 if and only if it is an interval graph. Kamali et al. [25] provided an algorithm with the approximation ratio of $1+o(1)$ for graphs that have bounded pathlength and another algorithm with the approximation ratio of $2+o(1)$ for burning connected graphs of bounded treelength.

2.4 Similar problems

Broadcasting a gossip

There are several broadcasting and gossiping protocols that have similarities to the burning problem. For instance, in the *telephone broadcasting problem* [5], the

input is a directed or undirected graph. A particular vertex is the *originator* of a gossip that needs to be transmitted throughout the network via "telephone calls". The communication takes place in synchronous rounds. Initially, only the originator knows the gossip. At each round, any vertex that knows the gossip makes a telephone call to at most one of its (outgoing) neighbors to pass the gossip. The goal is to schedule these telephone calls so that all vertices of the graph receive the gossip in a minimum number of rounds [5]. This telephone broadcasting is NP-hard [19, 45], and an approximation algorithm with a sublogarithmic factor exists[16]. An important open problem asks whether an algorithm with constant approximation factor exists. You could find more details about telephone broadcasting on [21, 37, 41]. The telephone model is not a good model for some cases where a node can expose all its neighbors in a single round (e.g., by exposing all neighbors simultaneously via a post in social media). In these scenarios, the *radio model* is more relevant, which is often studied under a distributed setting where it is assumed that vertices have a restricted amount of information regarding the structure of the graph [11, 20, 30, 40].

Viral marketing

Social contagion is very important from a viral marketing perspective[5]. This is because social contagion can prove the fact that choosing a few users can result in a word-of-mouth effect in the environment of a social network [5]. Domingos and Richardson [14, 42] introduced the *influence maximization problems*. In those problems, the goal is to define a number of users who are activated from the beginning and are capable of influencing the maximum number of users in a network. These problems are generally NP-hard, and there are approximation algorithms for uncomplicated diffusion models in these problems [26, 27]. We refer to [29] for further details

on these problems. We note that the burning problem is different from the problem of influence maximization. The reason is that in burning problem users (vertices) start fires at different rounds, while in influence maximization problem starting notified users begin rolling out data simultaneously [5].

The firefighter problem

There is another problem similar to the burning problem which is called *firefighter problem* [5]. This problem is very close to the burning problem because it also considers burning a graph in synchronous and discrete rounds [5]. Assume that the graph G is given to us. At round 1, a fire will initiate at a given node x of the graph G. At each following round, one firefighter is able to defend a non-burning vertex and in the meantime, the fire will spread over all neighbors of the burning vertex which are undefended. In a situation where the state of a vertex is "defended" or "burning", then that vertex will keep that state for all subsequent phases. The termination of the process takes place when the fire is no longer able to spread. The objective of the algorithm is to keep the maximum number of vertices that can be saved defended, which are not burning at the end of the process [5]. See Figure 4 for an example of the firefighter problem. Even though there are some resemblances between the firefighter model and the burning problem but their main goals are different from one another [5]. Namely, in the burning problem, an algorithm wants to burn the graph in a minimum number of rounds, while in the firefighter problem the algorithm wants to save a maximum number of vertices from the fire.

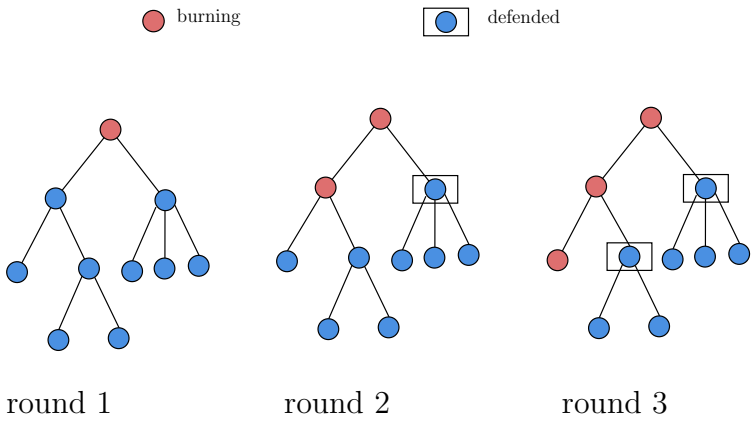


Figure 4: An example of the firefighter problem. At round 1, a fire starts at the root of the tree (a). At round 2, the firefighter defends the right child of the root, while the left child is burned (b). At round 3, the firefighter defended another vertex while the fire spreads to the leftmost leaf of the root. Note that this is the best strategy for the firefighter, which saves all but three vertices of the graph. The example is taken from [18].

Epidemics modelling

Modeling epidemics is a related topic to the burning problem which has been studied in a few previous works [12, 36, 38, 47, 48]. A very general model was introduced by Wang et al.[8], where a simulation model of a virus spreading through a stochastic graph was proposed. Similar work was introduced by Comellas et al.[9, 10]. Their approach was similar in terms of the discrete probability model, where it was assumed that when a vertex is infected, it will keep that statue for a specific time window. In addition, an infected vertex is also able to infect an α number of neighbors during each time unit. The authors studied the expected time it takes for a virus to spread all over the graph and also a setting where several defenders are placed on the different sections of the graph prior to the initiation of the process. These so-called "defenders" are capable of not only allowing a vertex that is infected, spread the virus

to its neighbors. In this sense, the model is more similar to the firefighter problem. We note that the stochastic assumption behind these modelings makes them different from the graph burning problem.

3 Problem Definition

The graph burning problem is defined as follows.

Definition 1. *Assume that an undirected and unweighted graph G is given. A fire spreads throughout G in synchronous rounds as follows. At round $t = 0$, none of the vertices of the graph are burned. At round $t = 1$, we select a vertex x_1 to burn. At round $t = 2$, we select a vertex x_2 to burn, while the fire at x_1 extends to all its neighbours. Similarly, at any subsequent round i, at most one fire starts at a vertex x_i, while all unburned vertices that are adjacent to a vertex that is burned at round x_{i-1} also burn. This process continues until all vertices are burned. When a vertex is burned, it will preserve its state as burned in all consequent rounds. In other words, the vertices do not change their state from "burned" to "unburned". The goal is to select the starting location of the fires in a way to minimize the number of rounds that it takes to burn the graph.*

The *burning number* of a graph G, denoted by $bn(G)$, is the smallest number of rounds that is needed to burn all the vertices of a graph.

Observation 1. *[25] Suppose we are given a path graph P with n vertices. Then $\lceil \sqrt{n} \rceil$ rounds are necessary and sufficient in order to burn this graph completely. In other words $bn(P) = \lceil \sqrt{n} \rceil$.*

To see why the above observation holds, let $k = bn(P)$. The fire started at round $i \leq k$ burns at most $2(k - i) + 1$ vertices. The total number of burned vertices is

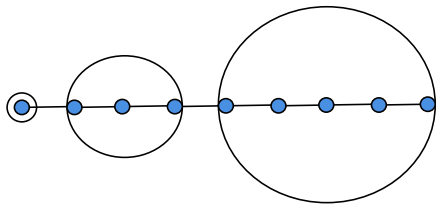


Figure 5: A path of length n can be covered with disks with radii of $0, \ldots, \lceil\sqrt{n}\rceil - 1$ i.e., it can be burned in at most $\sqrt{n}$ rounds. Here, $n = 9$, and disks of radii $0, 1, 2$ cover the path.

then at most $\sum_{i=1}^{k} 2(k-i)+1 = \sum_{i=0}^{k-1}(2i+1) = k(k-1)+(k-1) = k^2-1$. This means that $k^2 \geq n+1$or $k \geq \lceil\sqrt{n}\rceil$. The upper bound argument is similar [25]. A burning scheme can be devised by "covering" nodes in a path with disks of radii $0, 1, \ldots, \lceil\sqrt{n}\rceil - 1$. A disk of radii i is associated with a fire started at round $\lceil\sqrt{n}\rceil - 1$. See Figure 5.

4 Burning Cactus Graphs

4.1 Overview

This chapter is dedicated to studying the burning problem in cactus graphs (see Definition 2). We will show that the burning problem is not trivial for this graph family, in the sense a simple algorithm cannot yield to an optimal or approximate solution. We provide three findings. First, we consider a subfamily of cactus graphs, named necklace graphs, and show that the burning process can complete within $\sqrt{n}+$ $o(\sqrt{n})$ rounds in necklace graphs. Second, we show that the burning problem is NP-complete in cactus graphs. Finally, we introduce an improved approximation algorithm for burning cactus graphs which has an approximation ratio of 2.5.

Definition 2. *[39] A connected graph is called a cactus graph if any edge in the graph is a part of exactly one cycle in the graph. See Figure 6.*

We note that sometimes in the literature, a graph is defined to be a cactus iff any edge is a part of "at most" one cycle. In this thesis, we considered the definition above. In particular, we assume trees are not cactus graphs.

4.2 Necklace graphs

We start this chapter by studying the burning problem in a subfamily of cactus graphs that called necklace graphs.

Definition 3. *A necklace graph is a cactus graph formed by a set of cycles of uniform length k that are connected sequentially in a "path-way" manner. More formally, a connected graph is a necklace graph iff 1) every edge is a part of exactly one cycle, 2) every cycle has at most two vertices that participate in other cycles. 3) all cycles have the same length. See Figure 6.*

Note that a single cycle of length n is a necklace graph. Moreover, a cycle graph cannot be burned in fewer than $\lceil\sqrt{n}\rceil$ rounds. This is because a disk of radius i cannot burn more than $2i+1$ vertices in a cycle. We conclude that there are necklace graphs that require $\lceil\sqrt{n}\rceil$ rounds to burn. In what follows, we provide an almost matching bound, showing that burning a necklace graph with n vertices can complete within $\sqrt{n}+O(1)$ rounds.

Recall that a graph G has treelength k iff there is a tree decomposition for G such that vertices in each bag of the tree decomposition are at pairwise distance at most k [15]. We note that the existing approximation algorithms for burning graphs of small treelength [25] do not always yield to algorithms for burning cactus graphs because we can make the following observation.

Proposition 1. *There are cactus graphs that have unbounded treelength.*

The above proposition holds because a cycle of arbitrary large length is a cactus graph (in fact, a necklace graph) with unbounded treelength.

Theorem 1. *Any necklace graph G of size n can be burned within $\sqrt{n}+o(\sqrt{n})$ rounds.*

Proof. Let k denote the number of cycles in G. First assume $k \in o(n)$. Since there are k cycles, there will be $k-1$ vertices that appear in more than one cycle. We form a Travelling Salesperson (TSP) tour of G, which has length $n+k-1 = n+o(n)$ (this

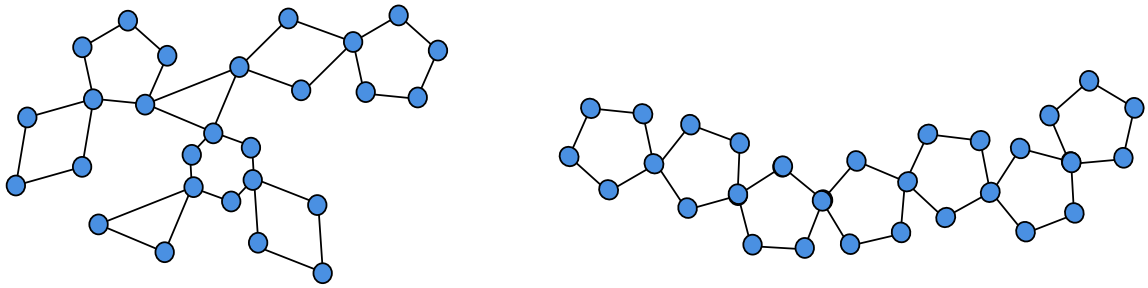


Figure 6: Examples of cactus graphs (left) and necklace graphs (right).

is because it visits the vertices that are shared between two cycles twice, and other vertices once). We can "stretch" the TSP tour to form a path of length $n + o(n)$, in which $o(n)$ vertices are repeated. The fact that some nodes are repeated in this path does not increase the burning time. Therefore we can use the same algorithm for burning paths, as in Observation 1, to burn the TSP tour in $\sqrt{n + o(n)} = \sqrt{n}+o(\sqrt{n})$ rounds. Given that the TSP path covers all vertices in the graph, all vertices will beurned within $\sqrt{n} + o(\sqrt{n})$ rounds.

Next, suppose $k \in \Theta(n)$. Then each cycle has length $n/k \in O(1)$. Consider a path starting form any vertex in the leftmost cycle and ending at any vertex at the rightmost cycle. The length of the path is less than n and hence all vertices on the path can be burned within $\lceil\sqrt{n}\rceil$ rounds. All other vertices are within distance $O(1)$ of the path, and hence the burning process completes in $\sqrt{n} + O(1)$ rounds.

□

4.3 NP-completeness

The burning problem in general graph is in NP and therefore it is also NP in any graph family, and in particular in cactus graphs. In what follows, we establish the NP-hardness of the problem in cactus graphs. For that, we use a reduction from the burning problem in disjoint path forests which is known to be NP-hard [2]. Given a

graph P formed by a set of disjoint paths, the decision problem asks whether it is possible to burn P within k rounds, for some $k \geq 1$. We first prove that burning a set of disjoint cycles is NP-hard, and use this problem as an intermediate step towards establishing the NP-hardness of cactus graphs.

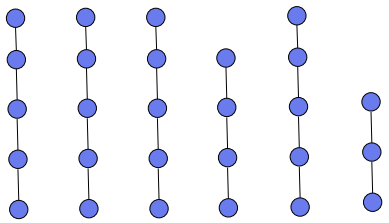


Figure 7: A set P of disjoint paths. Burning P is NP-hard, and we use a reduction from this problem to settle NP-hardness of the burning problem in cactus graphs.

Theorem 2. *It is NP-hard to decided whether a set of disjoint cycles can be burned within k rounds.*

Proof. Consider the decision problem that asks whether a given set P_0 of paths can be burned within k rounds, i.e., it can be covered with disks of radii $\{0, 1, \ldots, k-1\}$. We assume the number of disjoint paths in P_0 is at most k, for if it is larger than k, the answer to the decision problem is trivially "no". This is because any of the paths require at least one fire.

Suppose P_0 contains q paths of length 1 ($q \geq 0$), and let P be a copy of P_0 in which these q paths are replaced with q paths of length 2. We claim that P_0 can be burned in k rounds if and only if P can be burned in k rounds. Suppose P_0 can be burned in k rounds, i.e., we can cover P_0 with disks of distinct radii from $\{0, 1, \ldots, k-1\}$. Then each of the q paths in P_0 has been covered with a disk of radii at least 1; such disk is enough to cover a path of length 2 in P, and hence we can burn P in k rounds. The other direction is obvious: if we can cover vertices of P with disks of distinct radii

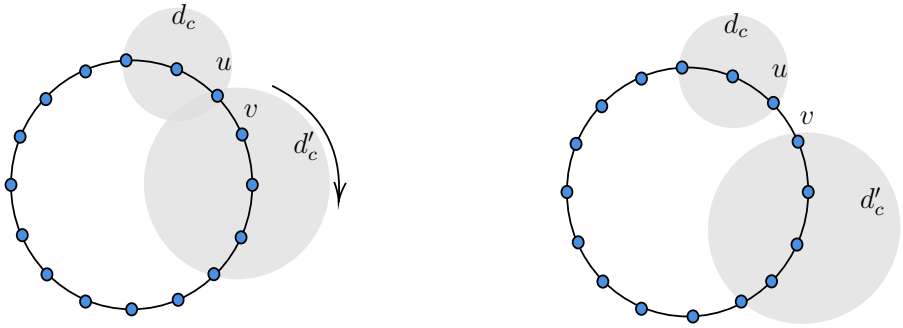


Figure 8: It is possible to shift disks in the burning scheme for cycles such that edges added to create C are not used.

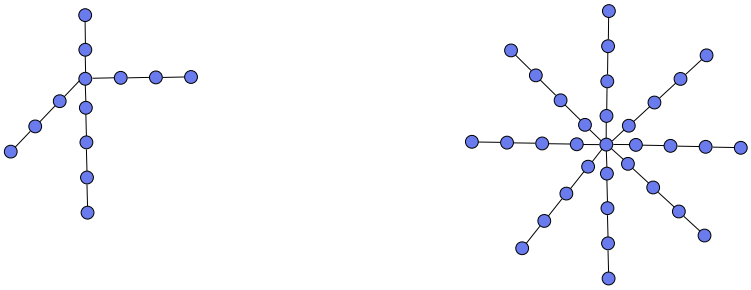
from $\{0, \ldots, k\}$, we can clearly use the same covering scheme to cover vertices of P_0 (because P_0 is an induced subgraph of P).

Next, we reduce the burning problem in P into an instance of burning problem in disjoint cycles. Create an instance C of the burning problem in cycles by adding an edge between the endpoints of any of the paths in P. Given that all paths in P have length at least 2, C is a set of disjoint cycles. We show that it is possible to burn P in k rounds if and only if it is possible to burn C in k rounds. First, suppose we can burn P in k rounds. Since P is a spanning subgraph of C (it contains all vertices of C), the same burning scheme can be used to burn C in k rounds.

Next, suppose C can be burned in k rounds. That is, we can cover vertices of C with disks of radii $\{0, 1, \ldots, k-1\}$. For any cycle $c \in C$, let d_c be any disk used to burn vertices in C. In case there is one or more disk d'_c such that d_c and d'_c intersect (they both cover a same vertex), we shift d'_c so that it does not intersect d_c (see Figure 8), while all vertices in C are still covered by at least one disk. This shifting process ensures that, in the updated burning scheme, any cycle $c \in C$ contains an edge from which the fire is not transferred, namely the edge that separates d_c and d'_c. Therefore, the resulting scheme burns P in k rounds. □

Next, we reduce the burning problem in a set C of disjoint cycles into the burning problem in cactus graphs. For that, we add some modifications and also gadget subgraphs to C. To construct our gadgets, we consider the family of *spider* graphs.

Definition 4. *A spider graph is a tree in which at most one vertex has a degree of more than two. In other words, it is a collection of paths, known as the legs of the spider, that share one endpoints. An $r-$uniform spider graph is a spider graph in which the length of all legs is equal to r. See Figure 9.*



(a) A (non-uniform) spider (b) A uniform spider of length 4.

Figure 9: Examples of spider graph.

Next, we convert spider graphs into a special gadget graph that could help us with the construction of cactus graphs.

Definition 5. *A flower graph is an $r-$uniform spider graph that has an even number of legs such that the ending points of pairs of legs are attached by an edge. The value of r is called the radius of the flower. See Figure 10.*

Note that a flower graph is a cactus graph.

We are now ready to settle the NP-hardness of burning cactus graphs.

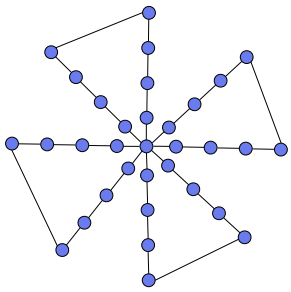


Figure 10: An example of a flower graph with 8 legs.

Theorem 3. *It is NP-hard to burn cactus graphs.*

Proof. Consider the decision problem that asks whether a give set C of $m \geq 2$ disjoint cycles with n vertices in total can be burned within k rounds. Note that we have $m \leq k$; otherwise the answer to the decision problem is trivially "no". As proved in Theorem 2, this problem is NP-hard. From C, we create an instance of the burning problem in cactus graph as follows. We arbitrarily order cycles in P, and connect consecutive pairs of cycles via flower graphs F_r that have distinct radii from $r \in \{k, k+1, \ldots, k+m-2\}$. The number of legs in each flower graph is set to be $2k^2$. Intuitively, this number should large enough so that any optimal burning scheme needs to start burning spiders from their center. Figure 11 illustrates the reductions we used to construct our cactus graphs.

We first observe that the graph G is a cactus graph. This is because any edge in G belongs to either one of the cycles in G or a flower graph. In both cases, it is a part of exactly one cycle. We also note that our construction takes polynomial time.

We claim that it is possible to burn C in k rounds if and only if one can burn G in $k+m-1$ rounds.

First, suppose it is possible to burn C in k rounds. We use the same burning scheme to burn the vertices of C in G. That is, we cover these vertices with disks of

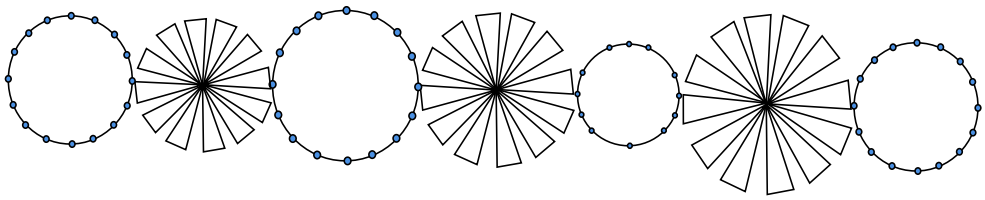


Figure 11: Construction of a cactus graph G by connecting cycles via flower graphs of distinct radii from $\{k, \ldots, k+m-2\}$.

distinct radii from $\{0, \ldots, k-1\}$. It is also possible to cover the flowers in G with disks of distinct radii from $\{k, \ldots, k+m-2\}$. For that, it suffices to put the center of these disks at the center of the flower graphs. We conclude that we can cover all vertices of G using disks of distinct radii from $\{0, 1, \ldots, k+m-2\}$, which means there is a burning schedule for G that completes in $k+m-1$ rounds.

Next, we show that if a burning scheme for G completes within $k+m-1$ rounds, then there is a burning scheme S for C that completes in k rounds. Note that the largest flower graph F_{k+m-2} has radius $k+m-2$. We claim that the scheme S places a disk of radius $k+m-2$ at the center of such graph. Consider otherwise, i.e., assume a smaller disk is located at the center of F_{k+m-2} or a disk is located at a non-center of F_{k+m-2}. In both cases, at least $2k^2-2$ vertices of F_{k+m-2} are not covered. That is, at most two leaves of the spider graph of F_{k+m-2} are covered. Given that F_{k+m-2} has $2k^2$ leaves, it is not possible to cover all its vertices with k disks. We conclude that F_{k+m-2} is covered with a disk whose center is at the center of F_{k+m-2}. With a similar argument, F_{k+m-3} must be covered by a disk of radius $k+m-3$, and more generally the largest $m-1$ disks in G must be covered by disks of distinct radii from $\{k+m-2, k+m-3, \ldots, k\}$ in S. We conclude that the remaining disks with radii from $\{0, \ldots, k-1\}$ must cover vertices in C. That is, there is a burning scheme for C that completes within k rounds. □

Given that the burning problem is in NP for the general graphs, we can conclude the following theorem.

Theorem 4. *The Burning Problem is NP-complete for cactus graphs.*

4.4 Approximation algorithms

Overview and necessary lemmas

In this section, we present an algorithm that has an approximation factor of at most 2.5 and runs in $O(n \log n)$ for burning cactus graphs of size n. We start with proving the following lemmas that are all necessary for proving the desired approximation factor.

Lemma 1. *[5] For any positive integer r, if a graph G contains r vertices with pairwise distances of at least $2r$, then any burning scheme for G needs at least r rounds to complete.*

Proof. Let $v_1, v_2, ..., v_r$ indicate r vertices that are at pairwise distance of at least $2r - 1$. No two of these vertices can be covered with a disks with radius $\leq r - 1$. Therefore, to burn these vertices within r rounds (to cover them with disks of distinct radii from $\{0, \ldots, r - 1\}$), no two v_i and v_j can be burned by the same fire (can be covered by the same disk). Therefore, there must be at least r disks, one for each v_i. □

Lemma 2. *Let s and t be two vertices of a cactus graph. Let g be any positive integer, and suppose $d(s, t) \geq g$. Then there are at most 2 vertices that are at distance g of s and lie on a shortest path between t and s.*

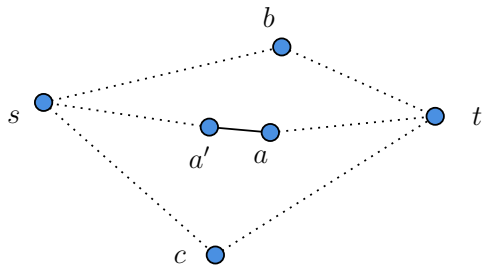


Figure 12: An illustration of Lemma 2

Proof. For the sake of contradiction, suppose there are three vertices a, b, c located on the shortest paths between s and t (see Figure 12) and all three vertices are at distance g of s. Let a' be the vertex at distance $g-1$ of s that appears before a on the shortest path from s to a. Note that $a' \notin \{a, b, c\}$. Therefore, (a', a) is an edge that is part of the following two cycles: one that goes through $(s, \ldots, a', \ldots, a, \ldots, t, \ldots, b, \ldots, s)$ and one that goes through $(s, \ldots, a', \ldots, a, \ldots, t, \ldots, c, \ldots, s)$. Since (a', a) is a part of two cycles, the input graph cannot be a cactus, which is a contradiction. □

Lemma 3. *Let g be an even integer. Consider a set of g disjoint paths, all including at most $2g$ vertices. It is possible to burn all vertices in these disjoint paths within $1.5g$ rounds.*

Proof. We would like to show it is possible to cover all vertices in the paths with disks of distinct radii from $\{0, 1, \ldots, 1.5g\}$. In order to cover half the paths ($0.5g$ paths), we use each of the largest $0.5g$ disks, i.e., those with radii in $\{g, \ldots, 1.5g\}$. Note that the radius of these disks is large enough to cover the $2g$ vertices in a path. For burning the remaining $0.5g$ paths with disks of radii $\{0, 1, \ldots, g-1\}$, we pair disks of radii i and $g-i$ to burn the i'th path. The total number of vertices that can be covered by the two disks is $2(i + g - i) = 2g$. See Figure 13. □

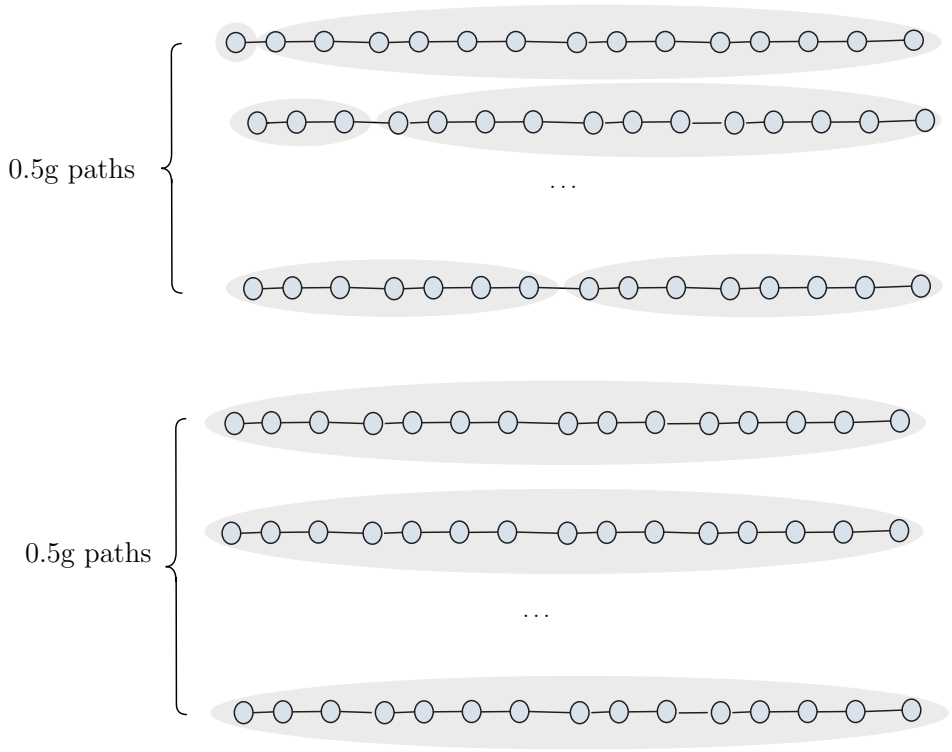


Figure 13: An illustration of burning a forest of g paths, each of length of at most $2g$ length, within $1.5g$ rounds.

Burn-guess-cactus procedure

Our approximation algorithm is based on a procedure called Burn-guess-cactus(G, g) that receives as input a cactus graph G and an integer parameter g, and returns one of the following outputs:

1. A burning scheme that completes the process in at most $2.5g + 2$ rounds.
2. "Bad-guess" that guarantees burning cannot be complete in $g - 1$ rounds (i.e., $b(G) \geq g$).

For the algorithm, it suffices to find the smallest guess value g^* for which Burn-

guess-cactus(G, g) returns a schedule. That is, the algorithm returns a schedule for the guess value $g = g^*$ and Bad-guess for $g = g^* - 1$. This way, the returned schedule completes in at most $2.5g^* + 2$ rounds while an optimal burning scheme requires at least g^* rounds to complete. This results in an algorithm with an asymptotic approximation ratio of at most 2.5.

In what follows, we consider the cactus graph G as a rooted cactus, i.e., we select an arbitrary vertex r as the root of the cactus and define the "depth" of a node as its distance to the root.

We maintain an initially empty set of center-terminal pairs. Each pair has a "center", which is a path of length at most $2g$, and a "terminal", which is singular vertex. The algorithm is based on a marking scheme. At the beginning, all vertices are unmarked. There are two phases in the algorithm. At the end of the first phase, either all vertices are marked, in which case the algorithm proceeds with the second phase, or Bad-Guess is returned.

Phase 1 of the algorithm works in iterations. At each iteration i, we select (any of) the deepest unmarked vertex (the one with the maximum distance from the root) and mark it as a terminal T_i. We will select a path C_i that together with T_i form a "center-path pair". For that, we consider the shortest paths between T_i and the root r of the cactus. By Lemma 2, there are at most two vertices like x and x' that are at distance g of T_i and lie on the shortest path between r and T_i. Three scenarios might happen at this point:

(i) Suppose T_i is within distance g or closer to the root. Since T_i is the deepest unmarked node, all unmarked nodes must be within distance g of the root r. In this case, we mark r as a center path, formed by a single vertex, and mark vertices within distance g of r. Note that all vertices will be marked. In this case, Phase 1 of the algorithm completes and we proceed with Phase 2.

(ii) Suppose there are two vertices x and x' at distance g of T_i that lie on the shortest path between r and T_i. For the distance between x and x', we can write $d(x, x') \leq d(x, T_i) + d(r, T_i') = 2g$. Therefore, the shortest path between x and x' has length at most $2g$. We treat this path as a center C_i to form a center-terminal pair (C_i, T_i). We proceed with marking all vertices that are within distance g of any vertex in C_i. Note that, since T_i is the deepest unmarked node in the graph, the shortest path between any unmarked node and T_i passes through either x and x', and we can conclude that any unmarked node is at distance at least $2g$ from T_i. This ensures that all future terminals will be at distance $2g$ or larger from T_i. See Figure 14a.

(iii) Suppose there is only one vertex x at distance g of T_i. In this case, we consider a path C formed by a singular vertex x and let (C_i, T_i) form a center-terminal pair. We proceed with marking all vertices within distance g of x. Similarly to case (iii), all unmarked vertices (and hence future terminals) will be at distance at least $2g$ of the terminal T_i. See Figure 14.

After g iterations of the above process, either all vertices are marked, in which case we proceed to Phase 2, or we have managed to find g pairs (C_i, T_i). In the second case, Burn-Guess-Cactus returns Bad-Guess.

Lemma 4. *If Burn-Guess-Cactus(G, g) returns Bad-Guess, then it is not possible to burn G with less than g rounds.*

Proof. The algorithm returns Bad-Guess if we have found g center-terminal pairs. As mentioned earlier, all terminals are at pairwise distance at least $2g$. Then we can apply Lemma 1 (with $r = g$) to conclude that it is not possible to burn G within less than g rounds. □

Next we discuss Phase 2, where the algorithm marks all vertices with less than g iterations. That means we have found at most $g-1$ pairs (C_i, T_i) such that all vertices in the graph are within distance g of some C_i. Recall that each C_i is either a single vertex or a path of length at most $2g$. So, we can use Lemma 3 to burn all vertices that are a part of some C_i in at most $\lceil 1.5g + 1.5 \rceil \leq 1.5g + 2$ rounds. The extra 1.5 factor is for the case when g is an odd number, in which case, we have to use $g+1$ in Lemma 3. Since all vertices are within distance g of some paths, an extra g number of rounds ensures that all vertices are burned within $2.5g + 2$ rounds. We can conclude the following lemma:

Lemma 5. *The Burn-Guess-Cactus(G, g) either returns Bad-Guess or a schedule that completes burning g within $2.5g + 2$ rounds.*

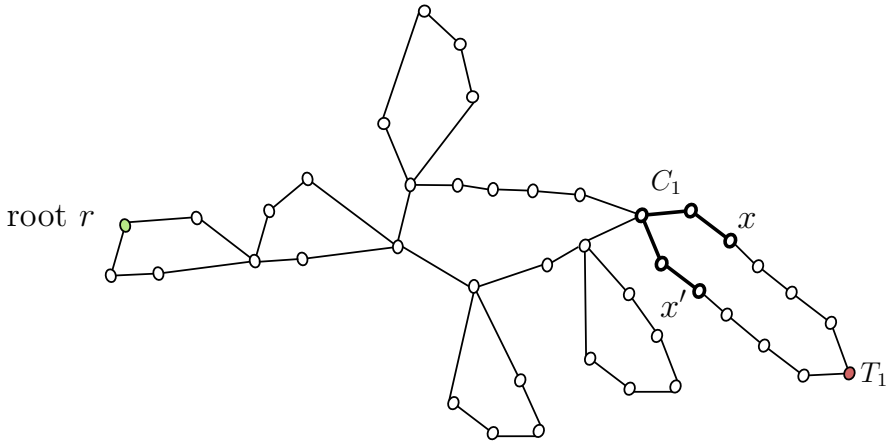
After g iterations of the above process, either all vertices are marked, in which case we proceed to Phase 2, or we have managed to find g pairs (C_i, T_i). In the second case, Burn-Guess-Cactus returns Bad-Guess.

Lemma 6. *The Burn-Guess-Cactus(G, g) either returns Bad-Guess or a schedule that completes burning g within $2.5g + 2$ rounds.*

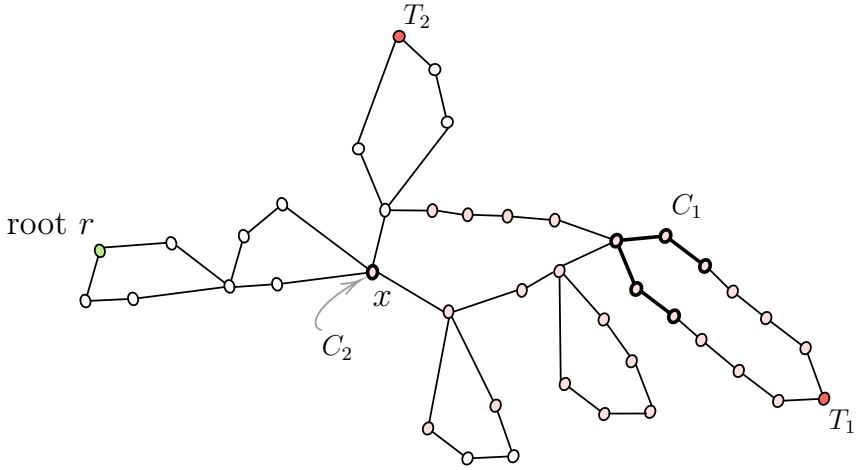
Proof. If the algorithm does not return Bad-Guess, then it continues to Phase 2, and as discuss above, we can burn all vertices in $2.5g + 2$. □

Theorem 5. *There is an approximation algorithm with approximation factor at most 2.5 for burning cactus graphs.*

Proof. We apply the Burn-Guess-Cactus(G, g) successively to find the smallest value of $g = g^*$ for which a burning schedule is returned. By Lemma 6, the schedule completes burning within $2.5g^* + 2$ rounds. Meanwhile, since Burn-Guess-Cactus(G, g)



(a) Iteration 1: an example of case (ii); the pair (C_1, T_1) is formed, where C_1 is the highlighted shortest path between x and x'.



(b) Iteration 2: an example of case (iii); the center-terminal pair (C_2, T_2) is formed, where C_2 is a singular vertex x.

Figure 14: An illustration of the Burn-Guess-Cactus procedure with $g = 4$. White vertices are unmarked, pink vertices are marked, and red vertices are terminals. The center paths are highlighted. At the end of the second iterations, all vertices will be marked, and Phase 1 ends.

returns Bad-Guess for $g = g^* - 1$, by Lemma **??**, at least g^* rounds are needed to burn G. We conclude that there is an approximation algorithm with approximation factor 2.5. □

5 Burning Melon Graphs

5.1 Overview

In this chapter, we study the burning problem in *melon* graphs. A melon graph is formed by a set of paths that share both endpoints (see Definition 6). Melon graphs are special instances of Series-Parallel (SP) graphs. Therefore, the study of melon graphs can pave the road for studying more general graph families such as graphs with bounded treewidth (because SP-graphs have treewidth 2) and planar graphs. As we will show, burning melon graphs is not trivial and hence they are worth studying. In particular, we show that it is NP-complete to find an optimal burning scheme in melon graphs. We complement this result with two polynomial-time approximation algorithms. First, we provide a simple algorithm with an approximation factor of 2. This algorithm runs in $O(n \log n)$ time to burn a melon graph of size n. Second, we provide an asymptotic polynomial-time approximation scheme (APTAS) for burning melon graphs.

5.2 Definition and related families

A melon graph is defined as follows:

Definition 6. *A melon graph consists of a set of paths and two vertices named poles. Each of those paths is connected to one of the poles (t) from one end and they are connected to another pole (s) by their other end [13]. See Figure 15.*

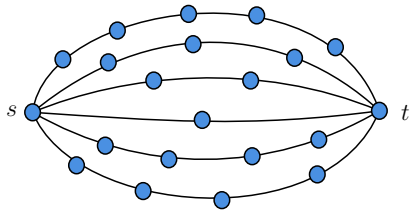


Figure 15: An example melon graph with vertices s and t being its poles.

Before studying algorithms for burning melon graphs, we need to discuss their relationship with other graph families. First, we show that melon graphs are not related to cactus graphs, and hence our approximation algorithm from the previous chapter cannot be used to burn melon graphs.

Proposition 2. *There are melon graphs that are not cactus graphs, and there are cactus graphs that are not melon graphs.*

Some melon graphs are not cactus graphs. For example, in the melon graph of Figure 15, any edge is a part of multiple cycles.

Some Cactus graphs are not melon graphs. The definition of melon graphs requires every vertex to be on on a simple path between the poles of the graph. Some Cactus graphs do not hold this about every vertex. See Figure 16.

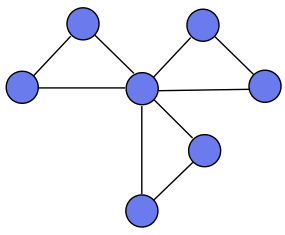


Figure 16: An example of a Cactus graph that is not a melon graph.

Next, we show that the existing approximation algorithms for burning graphs of small treelength cannot be used to burn melon graphs.

Proposition 3. *There are melon graphs that have unbounded treelength.*

The above proposition holds because a cycle of arbitrary large length is a melon graph with unbounded treelength.

Melon graphs have treewidth at most 2. This is because they are special instances of SP-graphs, which are known to have treewidth 2 [7]. Unfortunately, however, there is no previous work for burning graphs of small treewidth. Therefore, to provide algorithms for burning melon graphs, new approaches might be needed.

5.3 NP-completeness

In this section, we show that burning melon graphs is NP-complete.

Lemma 7. *The burning problem, when restricted to melon graphs, is NP-complete.*

Proof. Burning melon graphs is a subcategory of the burning graph problem. Bonato et al. [1] prove that the burning problem is in NP. Considering the fact burning melon graphs is a subcategory of the more general problem (burning general graphs) we can conclude that burning melon graphs is NP.

Next, we provide a hardness result. We use a reduction from burning a set of disjoint paths, which is known to be NP-complete [1]. Let P be a set of m disjoint paths (of different lengths), and suppose a decision problem asks whether P can be burned in at most k rounds (for some integer $k > 1$). We create an instance of the burning problem in melon graphs as follows.

Create a melon graph by creating a copy of P. We add a vertex s that is linked to one endpoint of all paths via paths of length k. That is, there are $k-1$ vertices between s and the endpoint of any path. Similarly, we add a vertex t that is linked to the other endpoints of all paths via paths of length $k-1$. That is, there are $k-2$ vertices between t and the endpoint of any path. We also add to G another $k+1$ paths between s and t. These new paths each include $2k-3$ vertices other than s and t. Note that graph G is clearly a melon graph with s and t being its poles. See Figure 17.

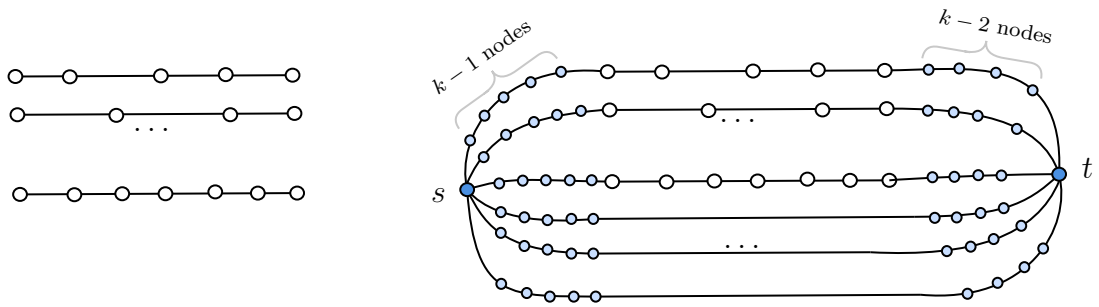


Figure 17: An illustration of the reduction used for the NP-hardness. On the left, we have a set P of disjoint paths, and on the right, we have the constrcuted melon graph G.

The decision problem for burning melon graph asks whether it is possible to burn G in $k+2$ rounds. In what follows, we show that it is possible to burn P in k rounds if and only if it is possible to burn G in $k+2$ rounds.

First, we show that if one can burn P in k rounds, it is possible to burn G in $k+2$ rounds. Assume that it is possible to burn P in k rounds. That is, we can cover all vertices in P with disks of radii $0, 1, \ldots, k-1$. A burning scheme for G can be devised by first burning s at round 0, then t at round 1, and then applying the same burning scheme that was used for burning P in k rounds. The first two fires, initiated at s and t, will burn all vertices that are in G and not in P. This is because the extra vertices added to G are either within distance $k-1$ of s or distance $k-2$ of t. The remaining fires (which were used for burning P) will be used to burn all vertices of G that are copied from P. Note that these fires will be associated with disks of the same radii used for burning P. See Figure 18.

Next, we show that if it is possible to burn G in $k+2$ rounds (with disks of radii $0, \ldots, k+1$), then it is possible to burn P in k rounds. Consider a burning scheme that burns G in $k+2$ rounds. We claim that the first two fires must burn the two poles of G. We know that there are $k+1$ paths, each including $2k-3$ vertices (excluding the poles) that link s and t. The middle points (vertices) of these paths are at distance $k-1$ from both poles. So, the pairwise distance between these middle points is $2k-2$. Except for the first fire, all other fires are associated with the disk of radii less than $k-1$. Therefore, if the first fire does not start in any of the poles, no two middle points can share a fire. Given that there are $k+1$ middle points, $k+1$ fires will be needed, which is not possible in a scheme that completes in k rounds. We conclude that the first fire must burn one of the poles. Given the symmetric nature of poles, suppose the first fire burns s. Next, we show that the second fire must start

at t. The fire started at s burns vertices within distance $k-1$ of s. On any of the extra paths added between s and t, the furthest vertex from t that is not burned by the fire at s is at distance $k-2$ of t. The pairwise distance between these "furthest points" is, therefore, $2k-4$. Given that $k+1$ paths added between s and t, there are $k+1$ vertices that are at pairwise distance $2k-4$ which are not burned by the fire at s. If the fire at t does not start at round 2, no two of the furthest vertices can be burned by the same fire (because any such fire must "pass through t"), and hence $k+1$ fires are needed, which is not possible, given that the burning schedule completes in k rounds.

We conclude that if there is a burning scheme that completes in $k+2$ rounds for burning G, it must be that the first two fires burn the poles of G. Therefore, no vertex copied from P can be burned by these first two fires. We conclude that all vertices copied from P are burned by the remaining k fires. That is, they are covered by disks of radii $0, 1, \ldots, k-1$ and hence P can be burned in k rounds. □

5.4 A $O(n \log n)$-time approximation algorithm

In this section, we present an algorithm that has an approximation factor of at most 2 and runs in $O(n \log n)$ for burning melon graphs of size n.

Outline

The algorithm is based on a procedure named Burn-Guess-Melon. This procedure receives a melon graph G of size n and a *guess value* $g \leq 2\sqrt{n}$ (note that $2\sqrt{n}$ is an upper bound for burning connected graphs) returns:

- either a "Bad-Guess", which indicates that it is not possible to burn G in g rounds

- or a burning scheme that completes in at most $2g$ rounds.

Our approximation algorithm works by finding the smallest value of g for which Burn-Guess-Melon returns a scheme. Let g^* denote such value. Since the outcome has been Bad-Guess $g^* - 1$, we can conclude that the optimal burning schedule requires at least g^* rounds. Given that our scheme completes at most $2g^*$, an approximation factor of at most 2 follows.

Burn-Guess-Melon procedure

The Burn-Guess-Melon procedure receives a graph G, and a guess value g and must output either Bad-Guess or a scheme that completes within $2g$ rounds. Let $P_1, P_2, \ldots, P_m$ be the paths resulting from deletion of the poles of G. Let n_i denote the number of vertices in P_i, and let $x_i = n_i \mod (2g-1)$.

We cover vertices of each path (if any), by grouping collections of $2g-1$ consecutive vertices. The middle vertex from each group is marked as a "center". This ensures that all centers are at a pairwise distance of at least $2g-1$. The remaining x_i vertices will be within distance g of one of the poles. See Figure 18.

At the end of this process, if the number of centers becomes larger than g, the procedure returns Bad-Guess. Otherwise, when all vertices are added to a group, we devise the following burning scheme: In the first two rounds, we burn the two poles. In the subsequent $q \leq g$ rounds, we burn the center vertices in an arbitrary order.

Lemma 8. *If Burn-Guess-Melon returns Bad-Guess for the guess value g and a graph G, then it is not possible to burn G within g rounds.*

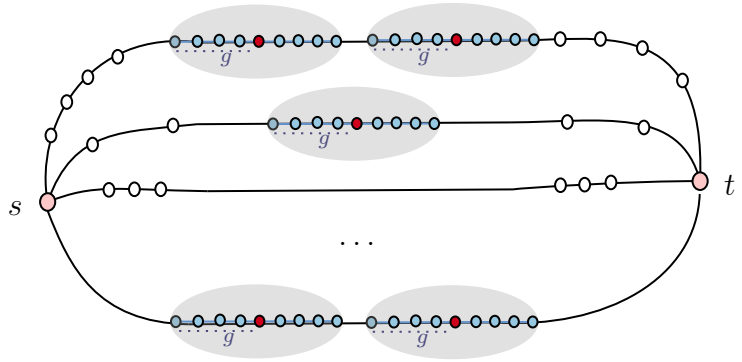


Figure 18: An illustration of the Burn-Guess-Melon for $g = 5$. Vertices in each path in the melon graph are covered in groups of $2g - 1(= 9)$. The centers of the disks are (red vertices) are at pairwise distance $2g - 1$. The disks are located such that the remaining x_i nodes from each paths are within distance g from one of the poles.

Proof. Burn-Guess-Melon returns Bad-Guess only if there are $g + 1$ vertices (centers) of pairwise distance at least $2g - 1$. Suppose for the sake of contradiction that there is a scheme that completes in g rounds, i.e., it is possible to cover vertices of G with disks of radii $0, 1, \ldots, g - 1$. Given that centers are at pairwise distance $2g - 1$, the same disk cannot cover two of them. That is, we need a separate disk for each center, resulting in at least $g + 1$ disks. this contradicts the fact that there are g disks (fires) in scheme S. □

Lemma 9. *If the Burn-Guess-Melon returns a scheme for input graph G and guess value g, then it is possible to burn G in at most $2g + 2$ rounds.*

Proof. Burn-Guess-Melon returns a scheme when a set of at most g centers are formed, and all vertices are within distance $2g$ of a pole or within distance $g-1$ of some center. Since we burn poles in the first two rounds, all vertices within distance $2g$ of a pole will be burned within $2g + 2$ rounds. Given that we burn centers in arbitrary order

and there are $q \leq g$ centers, the last center is burned at round $q+2 \leq g+2$. Since all vertices are within distance $g-1$ of one center, all vertices will be burned within $q+2+g-1 \leq 2g+1$ rounds. To summarize, all vertices will burn by round $2g+2$. □

Theorem 6. *Let $bn(G)$ denote the burning number of G, i.e., the optimal number of rounds to burn a melon graph G. There is a polynomial-time approximation algorithm for burning G that completes within $2bn(G)+2$ rounds.*

Proof. Given a melon graph G, we find the smallest value g^* for which Burn-Guess-Melon(G, g) returns a scheme. Since Burn-Guess-Melon(G, g^*-1) returns Bad-Guess, by Lemma 8, we conclude that the optimal burning schedule requires at least g^* round. Moreover, since a schedule is returned for guess value g^*, by Lemma 9, we get a burning scheme that completes in $2g^*+2$ rounds. The approximation factor of the algorithm is thus at most $\frac{2g^*+2}{g^*}$, which converges to 2 for large values of g^*. □

Running time

To study the running time of the algorithm, we first investigate the running time of Burn-Guess-Melon(G, g). Let n denote the number of vertices in G. Assuming the graph is stored using an adjacency list, one can mark vertices within desired distances of the poles ($g-1+\lfloor x_i/2 \rfloor$ and $g-1+\lfloor x_i/2 \rfloor$ of P_i) by traversing each vertex at most once (using, e.g., a Breadth-First-Approach). Similarly, we can locate centers and mark vertices within distance $2g-1$ of them by traversing each vertex at most once. Therefore, the running time of one run of Burn-Guess-Melon is $O(n)$. In order to find the best guess value g^*, we can apply a binary guess in the range $[1, \lceil 2\sqrt{n} \rceil]$. This takes $O(\log n)$ calls of Burn-Guess-Melon. We can conclude that the running time of our algorithm is $O(n \log n)$

5.5 An APTAS for burning melon graphs

Outline

We extend the algorithm of the previous section to get an asymptotic polynomial-time approximation scheme (APTAS) for burning melon graphs. As before, we devise a procedure, named Burn-Guess-Melon$^*(G, g)$, which receives a melon graph and a guess value g. The Burn-Guess-Melon$^*(G, g)$ either returns Bad-Guess, which implies that it is not possible to burn G within g rounds, or returns a scheme that completes in $(1 + \epsilon)g + c$ rounds, where ϵ is a parameter of the algorithm, and c is a constant independent of the input size. As before, we run Burn-Guess-Melon* on different values of g to find the smallest value g^* for which a schedule is returned. Since the procedure returns Bad-Guess for $g^* - 1$, an optimal scheme requires at least g^* rounds. Meanwhile, our scheme completes in $(1 + \epsilon)g^*2 + 2$ rounds. This ensures an approximation factor of $(1 + \epsilon)$ when the value of g^* is asymptotically large.

Burn-Guess-Melon* procedure

The Burn-Guess-Melon* procedure receives a graph G, and a guess value g and must output either Bad-Guess or a scheme that completes within $(1+\epsilon)g^*+2$ rounds, where $\epsilon < 1$ is a positive parameter.

As before, let $P_1, P_2, \ldots, P_m$ be the paths resulting from deletion of the poles of G. Let n_i denote the number of vertices in P_i. We mark vertices within distance g of the left endpoint of P_i as well as the g vertices of the right endpoint of P_i. So any vertex marked at the beginning is within distance g of one pole of G.

We remove marked vertices and consider the path forest P induced by the remaining vertices. We apply the PTAS of [5] to get a burning scheme S_P for burning P. Let q denote the number of rounds that the PTAS for burning P takes. There are two possibilities to consider:

- If $q \leq (1+\epsilon)g$ rounds, we return a scheme S_G for burning G that starts with first burning the two poles of G and then applies S_P to burn the remainder of the graph. Note that S_G requires two more rounds than S_P.
- If $q > (1+\epsilon)g$, the Burn-Guess-Melon* returns Bad-Guess.

Lemma 10. *If Burn-Guess-Melon$^*(G, g)$ returns Bad-Guess, then it is not possible to burn G within g rounds.*

Proof. Suppose Burn-Guess-Melon* returns Bad-Guess. Since the vertices within distance g of the poles were marked at the beginning, vertices in different components (paths) in P have a pairwise distance larger than $2g$ and hence the same fire cannot be used to burn two of them. This means that if one can burn G within g rounds, the same schedule can be used to burn P within g rounds. We show, however, that it is not possible when the procedure returns Bad-Guess. Since Burn-Guess-Melon* returns Bad-Guess, the PTAS for P outputs a burning scheme that completes in $q \geq (1+\epsilon)g$ rounds. We know that $q \leq (1+\epsilon)bn(P)$, i.e., $bn(P) \geq q/(1+\epsilon)$, any burning scheme for P requires at least $q/(1+\epsilon) = g$ round.

□

Lemma 11. *If the Burn-Guess-Melon$^*(G, g)$ returns a scheme, then it is possible to burn G in at most $(1+\epsilon)g + 2$ rounds.*

Proof. Burn-Guess-Melon returns a scheme when the scheme S_P for burning P completes in $q \leq (1+\epsilon)g$ rounds. The burning scheme for G starts with burning the poles of G and then follows the scheme of S_P. Given that all unmarked vertices at the beginning of the procedure are within distance g of a pole, all of them will be

burned by round $g+2$. All other vertices, i.e., vertices in P are also burned by $g+2$ rounds, given that the burning scheme for them follows S_P by a delay of 2 rounds. □

Theorem 7. *Let $bn(G)$ denote the optimal number of rounds to burn a melon graph G. For any positive parameter $\epsilon < 1$, there is a polynomial-time approximation scheme for burning G that completes in $(1+\epsilon)bn(G)+2$ rounds.*

Proof. Given a melon graph G, we find the smallest value g^* for which Burn-Guess-Melon$^*(G, g)$ returns a scheme. Since Burn-Guess-Melon$^*(G, g^*-1)$ returns Bad-Guess, by Lemma 10, we conclude that the optimal burning schedule requires at least g^* round. Moreover, since a schedule is returned for guess value g^*, by Lemma 11, we get a burning scheme that completes in $(1+\epsilon)g^*+2$ rounds. The approximation factor of the algorithm is thus at most $\frac{(1+\epsilon)g^*+2}{g^*}$, which converges to 2 for asymptotically large values of g^*. □

The PTAS of [23] is based on another PTAS for the bin covering problem. These algorithms are *fully-polynomial*, that is, their running time is polynomial on both $1/\epsilon$ and the number n of vertices in the input. Unfortunately, however, these polynomials grow fast with n and $1/\epsilon$. In particular, the running time of one run of Burn-Guess-Melon* takes $O(\alpha^5 n^2 \log(\alpha n))$, for $\alpha = 1/\epsilon$ [23]. Therefor this PTAS is also considered an FPTAS.

6 Concluding Remarks

In this thesis, we studied the graph burning problem for the cactus and melon family of graphs. We note that the burning problem is generally harder when the underlying graph is sparse (e.g., a tree). This makes the burning problem different from many other optimization problems, which are easy in sparse graphs. As expected, the problem is not trivial for cactus and melon graphs, as we proved via our hardness results.

Here is a short summary of our contributions:

- Proving that burning problem is NP-complete in cactus graphs.
- An approximation algorithm for burning cactus graphs with an approximation ratio of 2.5 (which is an improvement over the approximation factor 3 for burning general graphs).
- An algorithm that burns necklace graphs, a subfamily of cactus graphs, in $\sqrt{n} + o(\sqrt{n})$, which is optimal within lower order terms.
- We prove that the burning problem is NP-complete in melon graphs.
- We provide an approximation algorithm with an approximation factor of 2 for burning melon graphs and a slower polynomial-time approximation scheme (with an approximation factor $1 + \epsilon$) for burning melon graphs.

For future work, improving the current approximation algorithms that we have proposed for cactus graphs is an interesting subject. We believe a PTAS might exist

for cactus, and more generally, for SP-graphs (and possibly for the even more-general class of graphs with bounded treewidth). Another interesting open problem is to investigate the hardness of the burning problem in necklace graphs.

Bibliography

[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] S. Bessy, A. Bonato, J. C. M. Janssen, D. Rautenbach, and E. Roshanbin. Bounds on the burning number. *Discret. Appl. Math.*, 235:16–22, 2018.

[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] A. Bonato and S. Kamali. Approximation algorithms for graph burning. In T. Gopal and J. Watada, editors, *Theory and Applications of Models of Computation*, pages 74–92, Cham, 2019. Springer International Publishing.

[6] A. Bonato and S. Kamali. An improved bound on the burning number of graphs. *arXiv preprint arXiv:2110.01087*, 2021.

[7] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey*. SIAM, 1999.

[8] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (TISSEC)*, 10(4):1–26, 2008.

[9] F. Comellas, M. Mitjana, and J. Peters. Broadcasting in small-world communication networks. *TIC*, 1997:0963, 2002.

[10] F. Comellas, M. Mitjana, and J. G. Peters. Epidemics in small world communication networks. Technical report, Tech. Rep SFU-CMPT-TR-2002, 2002.

[11] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 492–501. IEEE, 2003.

[12] Z. Dezső and A.-L. Barabási. Halting viruses in scale-free networks. *Physical Review E*, 65(5):055103, 2002.

[13] T. Dissaux, G. Ducoffe, N. Nisse, and S. Nivelle. Treelength of series-parallel graphs. In C. E. Ferreira, O. Lee, and F. K. Miyazawa, editors, *Proceedings of the XI Latin and American Algorithms, Graphs and Optimization Symposium, LAGOS 2021, Online Event / São Paulo, Brazil, May 2021*, volume 195 of *Procedia Computer Science*, pages 30–38. Elsevier, 2021.

[14] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66, 2001.

[15] Y. Dourisboure and C. Gavoille. Tree-decompositions with bags of small diameter. *Discrete Mathematics*, 307(16):2008–2029, 2007.

[16] M. Elkin and G. Kortsarz. Sublogarithmic approximation for telephone multicast. *Journal of Computer and System Sciences*, 72(4):648–659, 2006.

[17] R. Elsässer. On the communication complexity of randomized broadcasting in random-like graphs. In *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 148–157, 2006.

[18] S. Finbow and G. MacGillivray. The firefighter problem: a survey of results, directions and questions. *Australas. J Comb.*, 43:57–78, 2009.

[19] M. R. Gary and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[20] M. Ghaffari, B. Haeupler, and M. Khabbazian. Randomized broadcast in radio networks with collision detection. *Distributed Computing*, 28(6):407–422, 2015.

[21] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.

[22] M. Hiller, E. Triesch, and A. M. Koster. On the burning number of p-caterpillars. *arXiv preprint arXiv:1912.10897*, 2019.

[23] K. Jansen and R. Solis-Oba. An asymptotic fully polynomial time approximation scheme for bin covering. *Theoretical Computer Science*, 306(1-3):543–551, 2003.

[24] B. Jyothsna and B. Radhakrishnan Nair. Burning number of some families and some products of graphs. *International Journal of Pure and Applied Mathematics*, 118(18):1489–1501, 2018.

[25] S. Kamali, A. Miller, and K. Zhang. Burning two worlds. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 113–124. Springer, 2020.

[26] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[27] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[28] A. Khan and E. Sharma. Tight approximation algorithms for geometric bin packing with skewed items. *arXiv preprint arXiv:2105.02827*, 2021.

[29] J. Kleinberg. Cascading behavior in social and economic networks. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 1–4, 2013.

[30] D. R. Kowalski and A. Pelc. Optimal deterministic broadcasting in known topology radio networks. *Distributed Computing*, 19(3):185–195, 2007.

[31] 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.

[32] H. Liu, X. Hu, and X. Hu. Burning number of caterpillars. *Discrete Applied Mathematics*, 284:332–340, 2020.

[33] D. Mitsche, P. Prałat, and E. Roshanbin. Burning graphs: a probabilistic perspective. *Graphs and Combinatorics*, 33(2):449–471, 2017.

[34] D. Mitsche, P. Prałat, and E. Roshanbin. Burning number of graph products. *Theoretical Computer Science*, 746:124–135, 2018.

[35] D. Mondal, N. Parthiabn, V. Kavitha, and I. Rajasingh. Apx-hardness and

approximation for the k-burning number problem. *arXiv e-prints*, pages arXiv–2006, 2020.

[36] M. E. Newman, I. Jensen, and R. Ziff. Percolation and epidemics in a two-dimensional small world. *Physical Review E*, 65(2):021904, 2002.

[37] A. Nikzad and R. Ravi. Sending secrets swiftly: Approximation algorithms for generalized multicast problems. In *International Colloquium on Automata, Languages, and Programming*, pages 568–607. Springer, 2014.

[38] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200, 2001.

[39] B. Paten, M. Diekhans, D. Earl, J. S. John, J. Ma, B. Suh, and D. Haussler. Cactus graphs for genome comparisons. *Journal of Computational Biology*, 18(3):469–481, 2011.

[40] D. Peleg. Time-efficient broadcasting in radio networks: A review. In *International Conference on Distributed Computing and Internet Technology*, pages 1–18. Springer, 2007.

[41] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 202–213. IEEE, 1994.

[42] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, 2002.

[43] N. Robertson and P. D. Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[44] K. A. Sim, T. S. Tan, and K. B. Wong. On the burning number of generalized petersen graphs. *Bulletin of the Malaysian Mathematical Sciences Society*, 41(3):1657–1670, 2018.

[45] P. J. Slater, E. J. Cockayne, and S. T. Hedetniemi. Information dissemination in trees. *SIAM Journal on Computing*, 10(4):692–701, 1981.

[46] W. Wang, S. Finbow, and P. Wang. The surviving rate of an infected network. *Theoretical Computer Science*, 411(40-42):3651–3660, 2010.

[47] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[48] D. H. Zanette and M. Kuperman. Effects of immunization in small-world epidemics. *Physica A: Statistical Mechanics and its Applications*, 309(3-4):445–452, 2002.