Approximation Algorithms and Lower Bounds for Graph Burning

Matej Lieskovský
Faculty of Mathematics and Physics, Computer Science Institute of Charles University, Prague, Czech Republic

Jiří Sgall
Faculty of Mathematics and Physics, Computer Science Institute of Charles University, Prague, Czech Republic

Andreas Emil Feldmann
Department of Computer Science, University of Sheffield, UK

Abstract

Graph Burning models information spreading in a given graph as a process such that in each step one node is infected (informed) and also the infection spreads to all neighbors of previously infected nodes. Formally, given a graph $G = (V, E)$, possibly with edge lengths, the burning number $b(G)$ is the minimum number g such that there exist nodes $v_0, \ldots, v_{g-1} \in V$ satisfying the property that for each $u \in V$ there exists $i \in \{0, \ldots, g-1\}$ so that the distance between u and v_i is at most i.

We present a randomized 2.314-approximation algorithm for computing the burning number of a general graph, even with arbitrary edge lengths. We complement this by an approximation lower bound of 2 for the case of equal length edges, and a lower bound of 4/3 for the case when edges are restricted to have length 1.

This improves on the previous 3-approximation algorithm and an APX-hardness result.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases Graph Algorithms, approximation Algorithms, randomized Algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.9

Category APPROX

Funding *Matej Lieskovský*: Partially supported by GAUK project 234723, and GA ČR project 19-27871X.
Jiří Sgall: Partially supported by GA ČR project 19-27871X.

Acknowledgements We are grateful to anonymous referees for detailed reviews and helpful comments.

1 Introduction

Graph Burning was introduced by Bonato et al. [4] as a model of information spreading and, more specifically, as a model of contagion or influence in social networks. The idea is that "infecting" several nodes in the network and spreading information from them may reach the whole network very fast. Formally, it is defined as the following process, where *burned* nodes in a graph represent the infected part of the network.

At time $t = 0$, no node of the graph is burned. At time $t = 1$, we choose a node and burn it. At each time step $t > 1$, all neighbors of already burned nodes are also burned, and we may choose another node to burn. The process stops when all nodes are burned. We call the sequence of chosen nodes a burning schedule of the graph. The burning number $b(G)$ of a graph G is the minimum number of steps needed for all nodes of G to be burned.

For a burning schedule of length ℓ, the node selected at time t ensures that all nodes within distance $\ell - t$ from it will be burned. We can thus reformulate the Graph Burning problem as looking for the minimal g such that all nodes of the graph can be covered by some placement of g balls with unique radii from $g-1$ to 0. Placing a ball with radius r at node v thus ensures the coverage of all nodes within the ball and represents selecting node v to be set on fire at time $g - r$ in the original formulation of the problem. This formulation also works naturally in the presence of arbitrary edge lengths.

The ball formulation emphasizes the relation of Graph Burning to the k-Center problem, where we try to cover the graph by k balls of equal radius, with the objective of minimizing the radius. In the non-uniform variant of the k-Center problem [7], different balls (centers) can have different radii, and in particular, the variant with a constant number of different radii is well-studied [11, 10]. Graph Burning can be viewed as an extreme version of the Non-Uniform k-Center problem, where neither the number of centers nor the number of distinct radii are fixed. This setting is challenging both for the design of efficient algorithms and for proving lower bounds. Thus studying Graph Burning can help to develop new techniques and insights that may be useful even for variants of k-Center and/or other facility location problems.

1.1 State of the art

Adapting a well-known greedy 2-approximation algorithm for the k-Center problem, Bonato and Kamali [5] gave a 3-approximation algorithm for Graph Burning. In fact, the approximation ratio of the algorithm is $3 - 2/b(G)$, which is a slight improvement if $b(G)$ is small; see also Appendix 3 for a review of this algorithm. Another $(3 - 2/b(G))$-approximation algorithm for arbitrary graphs was reported in [9]. In [5] it was stated as an open problem to find an improved algorithm, but no $(3 - \varepsilon)$-approximation algorithm for general graphs was known before our work.

In previous work approximation algorithms and approximation schemes were developed for trees [5] and other special graph classes [6], we refer to survey [3] for further references. This stream of research is now subsumed by [15], where it is shown that for graphs of small treewidth, a PTAS exists. The parameterized complexity of Graph Burning was studied in [12, 13, 14].

On the hardness side, it is known that computing the burning number of a graph is NP-complete even on simple classes of graphs such as trees or even disjoint unions of paths with unit edge lengths [2]. Answering an open question from [5], Mondal et al. [17, 18] have shown that Graph Burning is APX-hard, using a complex reduction from vertex cover in cubic graphs, but did not give an explicit lower bound on the approximation ratio.

1.2 Our results

In this paper, we present a randomized $(\alpha + \varepsilon)$-approximation algorithm for the Graph Burning problem on graphs with arbitrary edge lengths, where $\alpha = 2e^2/(e^2 - 1) < 2.314$ and $\varepsilon > 0$ is arbitrarily small. This is the first improvement of the previous easy 3-approximation algorithm based on k-Center results.

It has been speculated in [5] that an improved approximation algorithm will need to use a better lower bound on the burning number, presumably combinatorial. We do not use this line of attack. Instead, we use the power of randomization and analyze the following procedure: we greedily process the yet uncovered nodes and, on each such node, we put a center with radius chosen uniformly from a range $0, \ldots, R$ for a carefully chosen R. This

process is not easy to analyze, as the choices of nodes to be covered depend on the previous random choices. Nevertheless, we are able to show that it has a martingale property and use this to prove that the process succeeds with high probability.

On the hardness side, we give simple and explicit lower bounds on the approximation ratio for Graph Burning. We first prove a lower bound of 2 on the approximation ratio for graphs with edge lengths, which is close to our upper bound. This is an important insight since the methods we use in our algorithms but also other current approaches work even in the presence of arbitrary edge lengths, so they are all subject to this lower bound. We note that the lower bound works even if all the edge lengths are equal. We also prove a lower bound of 4/3 for graphs where all edges have length 1.

In [5] the authors note that known hardness results for related problems such as the k-Center or Dominating Set problems do not apply to Graph Burning. Indeed, they cannot be applied directly, as in approximations for Graph Burning we implicitly use both more centers and larger radii of the balls. Lower bounds for this kind of bicriteria approximations are naturally harder to show, which explains the slow progress on the inapproximability side.

We use a somewhat indirect reduction from the Dominating Set problem. For edges of length 1 the reduction is more complicated, as we need to subdivide the edges of the Dominating Set instance. As a consequence, when converting the dominating set to a Graph Burning solution, we have to use larger radii to cover the new vertices, which decreases the inapproximability bound. When other edge lengths are allowed, the subdivided edges can be replaced by long edges, which leads to the improved lower bound of 2. This is close to the performance of our algorithm.

Closing the remaining gap is an interesting open problem. We conjecture that the optimal approximation ratio is 2 at least for the version with arbitrary edge lengths.

1.3 Paper overview

In Section 2 we introduce some notations and conventions. In Section 3, we review the 3-approximation algorithm in our framework, which amounts to using the same radius for all centers. In Sections 4 and 5 we gradually introduce the building blocks of our algorithm. We examine the possibility of using a constant number of different radii, which leads to a deterministic algorithm for a logarithmic number of centers (Section 4). Then we examine the possibility of randomly choosing among two radii in Section 5. This leads to an approximation ratio below 3 and allows us to gently introduce the necessary probabilistic machinery, including the martingale approach. In Section 6 we apply all the ingredients and present the final 2.314-approximation algorithm. Finally, in Section 7 we present our lower bounds.

2 Preliminaries

We start with a formal definition of Graph Burning.

▶ **Definition 2.1.** *An instance of* Graph Burning *is an undirected graph $G = (V, E)$ with non-negative edge lengths. The* distance *$d(u, v)$ is defined as the length of the shortest path from u to v. We let $n = |V|$.*

A center *is a pair $c = (v, r)$ with $v \in V$ and r a non-negative integer; v denotes the node where the center is placed and r denotes the radius. Given a center c we denote its radius by $r(c)$. A solution of a Graph Burning instance is a set of centers $\{(v_0, 0), (v_1, 1), \ldots, (v_{g-1}, g-1)\}$. It is feasible if for any node $w \in V$ there exists $i < g$ such that $d(w, v_i) \leq i$. The objective of the Graph Burning problem is to minimize g over all feasible solutions. The minimum g is called the* burning number *of G and is denoted $b(G)$.*

Conventions for the algorithms. When running our algorithms, we will be building up a set of centers ALG. We allow the set ALG not to use all the radii between 0 and the maximal radius. At the end of the algorithm, if needed, balls with the remaining radii can be placed arbitrarily completing a feasible solution without changing the maximal radius.

The set $\{(v_0, 0), \ldots, (v_{n-1}, n-1)\}$ is a *trivial* valid solution for any ordering of $V = \{v_0, \ldots, v_{n-1}\}$. Thus $b(G) \leq n$ even with arbitrary edge lengths. We assume that $V \neq \emptyset$ and thus $b(G) \geq 1$. For most of our algorithms, we will assume that the algorithm is given an integer g, presumably the burning number, and its goal is to find a feasible solution of size at most σg for some approximation ratio σ, or possibly fail if g is smaller than $b(G)$. We then try all n possible values of g in our final algorithm.

Notations for the optimum. In the analysis of our algorithms, we fix some optimal solution OPT using centers with unique radii from 0 to $b(G) - 1$. Furthermore, we assign each node $w \in V$ to a unique center $c_{\text{OPT}}(w) = (v, r) \in \text{OPT}$ that is covering it (i.e., $d(w, v) \leq r$), thus creating a partition of the set of nodes of the graph. For every $c_{\text{OPT}} \in \text{OPT}$ we denote the set of nodes assigned to it as $V(c_{\text{OPT}})$.

Good centers. For any center c_{OPT} in the optimal solution, if the computed solution ALG contains a center $c_{\text{ALG}} = (v, r)$ where $r \geq 2r(c_{\text{OPT}})$ and $v \in V(c_{\text{OPT}})$, then all nodes in $V(c_{\text{OPT}})$ are covered by c_{ALG}. If such a c_{ALG} exists, we say that c_{OPT} is made *good* by c_{ALG}.

▶ **Definition 2.2.** *A center $c_{OPT} \in OPT$ is* good *if there exists a $c_{ALG} = (v, r) \in ALG$ with $r \geq 2r(c_{OPT})$ and $v \in V(c_{OPT})$.*

Note that c_{OPT} being good is not a necessary condition for all of $V(c_{\text{OPT}})$ to be covered – a well-placed center of radius $r < 2r(c_{\text{OPT}})$, multiple smaller centers, and/or centers located outside of $V(c_{\text{OPT}})$ might cover all of $V(c_{\text{OPT}})$.

We design our algorithms so that the (expected) number of good centers increases in each step and rely on the fact that a solution ALG is feasible whenever it makes all centers of the optimum good. While it may happen that not all centers of OPT become good, eventually all nodes end up covered as the number of good centers is increasing and it cannot increase beyond the number of centers in the optimal solution.

Classes of centers. Most of our algorithms divide the radii at their disposal using a series of thresholds. The thresholds are given as a list $q_1, q_2, q_3, \ldots$ of increasing numbers (not necessarily integers). For a center $c_{\text{OPT}} \in \text{OPT}$, we define its *class* as the smallest i such that using q_i for $v \in V(c_{\text{OPT}})$ ensures that c_{OPT} becomes good:

▶ **Definition 2.3.** *For a center $c_{OPT} \in OPT$ we refer to the minimal i such that the threshold q_i is larger than or equal to $2r(c_{OPT})$ as the* class *of c_{OPT}.*

Procedure for selecting a radius. Whenever our algorithms need to use a radius to add a center to ALG, they call the procedure SELECT(q), which takes a number q (typically a threshold) as its input. It then returns the minimal integer radius that has not yet been used by the algorithm and is at least q.

This procedure guarantees that our algorithms always use centers of distinct radii, at the cost of possibly increasing the largest radius and thus the size of the solution. In the analysis, we then need to prove that the maximal radius used is (with high probability) small enough.

Notations. We use $\exp(x)$ to denote e^x. We also use the convention that $x/yz = x/(yz)$.

3 Deterministic 3-approximation algorithm and overall approach

We revisit the 3-approximation algorithm from [4]. See Algorithm 1 for our formalization; recall that we assume that the algorithm is given g. We use a single threshold $q_1 = 2(g-1)$. This means that all centers c_{OPT} are class 1, as the optimum uses radii up to $g-1$, while our algorithm will use radii $2g-2$, $2g-1$ and so on.

Algorithm 1 The deterministic 3-approximation algorithm.

ALG $\leftarrow \emptyset$; $q_1 = 2g-2$
while an uncovered node exists **do**
  Select an arbitrary uncovered node v
  $r = \textsc{Select}(q_1)$
  Add center (v, r) to ALG and update the set of uncovered nodes
return ALG

Observe that whenever we select an uncovered node v, it is covered by some class 1 center $c_{\text{OPT}}(v)$ that is not good. Placing a center c_{ALG} with radius at least q_1 at v is guaranteed to make $c_{\text{OPT}}(v)$ good. Thus, once we place g centers c_{ALG} with radii at least q_1, all centers of OPT will be good and the algorithm finishes. Note that all nodes might be covered without all centers of OPT being good, making the algorithm finish sooner. Either way, we are guaranteed to never select a radius greater than $q_1 + g - 1 = 3g - 3$, making this a 3-approximation algorithm. More precisely, the algorithm uses $3g-2$ centers, as opposed to optimal $b(G) \le g$. So the approximation ratio becomes $3 - 2/b(G)$, as mentioned in the introduction.

4 Deterministic algorithm for small burning number

Let us now explore an approach that uses z thresholds equally spaced between 0 and $2(g-1)$, for some constant z. This means, disregarding rounding, that the optimum centers are divided into z classes, where the class i has g/z centers of radii between $(i-1)g/z$ and ig/z. The algorithm will aim to cover the centers of class i using thresholds approximately $2ig/z$, each for g/z nodes. As a result the largest radius can be bounded by $2g + g/z$.

Similar to before, this algorithm will choose an uncovered node in each iteration as a center for the approximate solution. This would work easily if we knew in advance the class of the center c_{OPT} covering the chosen node, so that we could use the corresponding threshold in the algorithm. Instead, we try all the possible sequences of thresholds, using each threshold for at most $\lceil g/z \rceil$ nodes, matching the number of optimal centers of each class. The analysis is somewhat subtle, as the node to be processed depends on the previous choices of the algorithm, so the sequence of thresholds used in the optimum needs to be constructed based on the nodes chosen by the algorithm.

Formalizing this idea with some attention to rounding and also the fact that the optimum can use fewer than g centers, we obtain an algorithm that finds a Graph Burning solution with objective at most $(2+1/z)g$ in time $\mathcal{O}(z^g \text{poly}(n))$ for any given integer z and $g \ge b(G)$; see Algorithm 2.

▶ **Theorem 4.1.** *If $g \ge b(G)$ then Algorithm 2 runs in $\mathcal{O}(z^g \text{poly}(n))$-time and achieves an approximation ratio of $(2 + 1/z)$.*

For the full proof of Theorem 4.1 see Appendix A. (For $z \ge g$, the proof actually gives ratio $2 - 1/g$. We omit that in the statement, as in the relevant case z is small.)

Algorithm 2 Deterministic algorithm for small g.

if $z > g$ **then** $z \leftarrow g$
Order V arbitrarily
$q_i \leftarrow 2gi/z - 2$ for $i \in \{1, \ldots, z\}$
Let $\mathbb{S}$ be a set of all possible sequences $S = s_0, s_1, \ldots, s_{g-1}$ with $s_t \in \{1, \ldots, z\}$
  such that each value is used at most $\lceil g/z \rceil$ times
for $S \in \mathbb{S}$ **do**
  ALG $\leftarrow \emptyset$
  for $t \leftarrow 0, 1, \ldots, g-1$ **do**
    Let v be the first uncovered node; $r \leftarrow$ SELECT(q_{s_t})
    Add center (v, r) to ALG and update the set of uncovered nodes
    if all nodes are covered **then return** ALG
return Fail

The running time is polynomial if $g \in \mathcal{O}(\log n)$ and $z \in \mathcal{O}(1)$, thus we get a deterministic $(2 + 1/z)$-approximation algorithm in this case. We will later use this result in our final algorithm with $z = 4$, ensuring an approximation ratio of $2 + 1/4 < 2.314$ for small $b(G)$.

5 Randomized approach with two classes of centers

As a warmup, in this section we introduce the main ideas of our techniques by considering a simpler setup. In particular, let us only use two thresholds $g - 1$ and $2(g - 1)$, and choose a radius randomly among the two values of the threshold; see Algorithm 3.

Algorithm 3 The randomized algorithm with two classes.

ALG $\leftarrow \emptyset$; $q_i \leftarrow i(g-1)$ for $i \in \{1, 2\}$
while an uncovered node exists **do**
  Select an arbitrary uncovered node v
  Let $k_v \in \{1, 2\}$ be chosen uniformly at random, independently from other choices
  $r \leftarrow$ SELECT(q_{k_v})
  Add center (v, r) to ALG and update the set of uncovered nodes
return ALG

Note that, as the radii start with 0, the optimum has $\lceil g/2 \rceil$ class 1 centers and $\lfloor g/2 \rfloor$ class 2 centers. Observe that a class 1 center c_{OPT} becomes good whenever we select an uncovered node from $V(c_{\text{OPT}})$. A class 2 center c_{OPT} becomes good if we select an uncovered node from $V(c_{\text{OPT}})$ and then randomly decide to use threshold q_2. If we were to make all centers of OPT good, we would use in expectation one radius per class 1 center and two radii per class 2 center, evenly distributed between threshold q_1 and threshold q_2. Since each class of centers of OPT contains roughly $0.5g$ centers, we expect at most $1.5g$ total radii being needed. Of these, we expect half, i.e., $0.75g$, to be at least the threshold $q_2 \leq 2g$. This indicates that roughly a 2.75-approximation algorithm should be the result.

Using a detailed analysis it is indeed possible to prove that Algorithm 3 is a $(2.75 + \varepsilon)$-approximation algorithm, improving on the previously known 3-approximation. For simplicity and to introduce the key ideas for our main algorithm, we will however only prove that this is a 2.9-approximation algorithm for g divisible by 10 (to avoid rounding issues), and then move on to give our 2.314-approximation in the following section. We show that the following holds with high probability:

- less than $1.6g$ centers c_{ALG} are placed by Algorithm 3, and
- less than $0.9g$ centers c_{ALG} with radius at least q_2 are placed, conditioned on the previous event (i.e., that the algorithm uses less than $1.6g$ centers).

5.1 Total number of centers

We will use martingales to analyze the algorithm; see the textbooks [1, 16] for reference. For convenience, we use a slight modification of the one-sided Azuma inequality for supermartingales. This follows easily, as any supermartingale can be converted to a martingale by shifting (increasing) the random variables appropriately.

▶ **Definition 5.1.** *A sequence of random variables $X_0, X_1, X_2, \dots$ is a supermartingale if and only if both $|\mathbb{E}[X_t]| < \infty$ and $\mathbb{E}[X_{t+1} \mid X_0, \dots, X_t] \leq X_t$ for all t.*

▶ **Theorem 5.2** (Azuma Inequality)**.** *If $X_0, X_1, X_2, \dots$ is a supermartingale with $|X_t - X_{t-1}| \leq \delta$ for all t, the following inequality holds:*

$$\Pr[X_t - X_0 \geq \Delta] \leq \exp\left(\frac{-\Delta^2}{2\delta^2 t}\right).$$

We will track the progress of our algorithm by defining a potential describing the difference between the algorithm's true and expected behaviour. Formally, let C_t be a random variable denoting the set of centers of OPT made good by the first t centers c_{ALG} placed by Algorithm 3. Let us define $f(c_{\text{OPT}})$ as the class of c_{OPT}. Finally, let the potential be

$$\Phi_t = t - \sum_{c_{\text{OPT}} \in C_t} f(c_{\text{OPT}}).$$

The value $f(c_{\text{OPT}})$ is chosen so that a center c_{OPT} is expected to become good after the algorithm selects $f(c_{\text{OPT}})$ centers from $V(c_{\text{OPT}})$. Thus, at a given moment, the sum in the potential denotes the number of centers the algorithm was expected to select to make all $c_{\text{OPT}} \in C_t$ good. If t, the actual number of centers of the algorithm, is smaller than the sum, the potential is negative and the algorithm is progressing better than expected. If t is smaller than the sum, the potential is positive and the algorithm is progressing worse than expected. Our goal is to show that the potential is unlikely to be positive and large.

The random choice of a threshold influences whether the current $c_{\text{OPT}}(v)$ enters C_t. Besides that, the random choices also possibly change the set of covered nodes, influencing the later choices of the algorithm. Thus the changes of the potential in different steps are not independent (even if the choices of thresholds are) and we need to analyze the process as a martingale.

Observe that $\Phi_0, \Phi_1, \dots$ is a sequence of random variables with $\Phi_0 = 0$. When we select node v as the $(t+1)$-st node to cover, only $c_{\text{OPT}} = c_{\text{OPT}}(v)$ can become good. If center c_{OPT} is class 1, then $f(c_{\text{OPT}}) = 1$ and we make it good with probability 1, which guarantees $\Phi_{t+1} = \Phi_t$. If center c_{OPT} is class 2, $f(c_{\text{OPT}}) = 2$. We make it good with probability at least $1/2$ and in this case $\Phi_{t+1} = \Phi_t - 1$. Otherwise $\Phi_{t+1} = \Phi_t + 1$. Thus $\mathbb{E}[\Phi_{t+1} \mid \Phi_0, \dots, \Phi_t] \leq \Phi_t$ and $\Phi_0, \Phi_1, \dots$ is a supermartingale with $\Phi_0 = 0$ and $\delta = 1$.

We have $\sum_{c_{\text{OPT}} \in \text{OPT}} f(c_{\text{OPT}}) \leq 1.5g$, corresponding to the fact that the expected number of radii needed for the algorithm is $1.5g$. Thus, if the algorithm does not stop before placing $1.6g$ centers, we have $\Phi_{1.6g} = 1.6g - \sum_{c_{\text{OPT}} \in C_{1.6g}} f(c_{\text{OPT}}) \geq 0.1g$. We use Azuma's inequality to show that the probability of this happening is small, keeping in mind that $\Phi_0 = 0$ and $\delta = 1$:

$$\Pr[\Phi_{1.6g} \geq 0.1g] \leq \exp\left(\frac{-0.01g^2}{3.2g}\right) = \exp\left(\frac{-g}{320}\right).$$

5.2 Number of large centers

To show that less than $0.9g$ centers c_{ALG} with radius at least q_2 are placed among the first $1.6g$ centers chosen by the algorithm, we use the standard Chernoff bound, see the textbooks [1, 16].

▶ **Theorem 5.3** (Chernoff bound). *Suppose $X_1, \ldots, X_J \in \{0,1\}$ are independent random variables. Let $X = \sum X_j$ and $\mu = \mathbb{E}[X] = \sum \mathbb{E}[X_j]$. Then for any $\varepsilon \in (0,1]$,*

$$\Pr[X \geq (1+\varepsilon)\mu] \leq \exp\left(\frac{-\varepsilon^2\mu}{3}\right).$$

Suppose that the algorithm chooses J centers for some $J \leq 1.6g$. Let X_j, $j = 1, \ldots, J$, be a 0-1 indicator variable equal to 1 if we use a threshold q_2 when choosing the j-th center in the algorithm. Since every X_j is equal to 1 with probability $1/2$ we have $\mu = \mathbb{E}[\sum X_j] = J/2 \leq 0.8g$. By Chernoff's bound, the probability of using threshold 2 more than $0.9g$ times is at most

$$\Pr[X \geq (1+\tfrac{1}{8})\mu] \leq \exp\left(\frac{-(\frac{1}{8})^2\mu}{3}\right) \leq \exp\left(\frac{-g}{320}\right).$$

5.3 Final bound

▶ **Theorem 5.4.** *The probability that Algorithm 3 uses a radius larger than or equal to $2.9g$ is at most $2\exp(-g/320)$.*

Proof. Using the previous two estimates, the probability that the algorithm uses more than $1.6g$ centers in total or more than $0.9g$ times threshold q_2 is bounded by $2\exp(-g/320)$.

If none of these events happen, we distinguish two cases. Either the algorithm uses all the radii between q_1 and q_2; then we use the fact that the total number of centers is at most $1.6g$ and thus the largest radius is at most $q_1 + 1.6g < 2.6g$. Otherwise, radii q_2 and larger are used only when threshold q_2 is selected, which happens at most $0.9g$ times and thus the largest radius is less than $q_2 + 0.9g < 2.9g$. ◀

6 Main randomized algorithm

The performance of Algorithm 3 can be further improved by using more thresholds. Ultimately, this leads to using a range of integers as thresholds. In particular, for a given g we use all integers up to approximately αg where α is the desired approximation ratio given by the following definition.

▶ **Definition 6.1.** *We set $\alpha = 2e^2/(e^2-1) < 2.314$.*

Let us present and analyze Algorithm 4 which is the key part of the final algorithm. For any $\varepsilon > 0$, we shall show that its approximation ratio is at most $(1+\varepsilon)^4\alpha$. We assume that $\varepsilon < 1$ (as otherwise we could use the known 3-approximation), $\alpha g < |V|$ (as otherwise a trivial solution is an α-approximation), $g \geq b(G)$ (as we will later iterate over all g) and $g \geq 100/\varepsilon^3$ (as we will use Algorithm 2 otherwise).

6.1 Expected number of centers

Next we analyze the expected number of centers that the algorithm uses. This is the key part that determines the approximation ratio and our choice of α. In particular, we set α so that the expected number of centers used is αg. On one hand, the largest radius is

Algorithm 4 The randomized algorithm for large g.

ALG $\leftarrow \emptyset$; $R \leftarrow (1+\varepsilon)^2 \alpha g$
while an uncovered node exists **do**
  Select an arbitrary uncovered node v
  Let $k_v \in \{0, \ldots, \lfloor R \rfloor\}$, be chosen uniformly and independently at random
  $r \leftarrow \textsc{Select}(k_v)$
  Add center (v, r) to ALG and update the set of uncovered nodes
return ALG

necessarily at least the number of centers used, on the other hand, once the range of radii exceeds the expected number of centers by a small factor, we will be able to show that with high probability the largest radius used is not much larger than the expected number of centers. This eventually gives an approximation factor arbitrarily close to the chosen α.

Similar to Section 5, the value $f(c_{\mathrm{OPT}})$ is chosen so that a center c_{OPT} is expected to become good after the algorithm selects $f(c_{\mathrm{OPT}})$ centers from $V(c_{\mathrm{OPT}})$.

▶ **Definition 6.2.** *For any center c_{OPT} of the optimum, define*

$$f(c_{OPT}) = \frac{\alpha g}{\alpha g - 2r(c_{OPT})} = \frac{1}{1 - \frac{2r(c_{OPT})}{\alpha g}}\,.$$

▶ **Lemma 6.3.** *For any given center c_{OPT}, whenever Algorithm 4 selects a node in $V(c_{OPT})$, the probability that c_{OPT} is made good is at least $1/f(c_{OPT})$. The expected number of nodes in $V(c_{OPT})$ selected by Algorithm 4 is bounded by $\mathbb{E}\left[|ALG \cap V(c_{OPT})|\right] \le f(c_{OPT})$.*

Proof. Every time the algorithm selects a node in $V(c_{\mathrm{OPT}})$, the number of possible values for uniformly random k_v is $1 + \lfloor R \rfloor \ge R \ge \alpha g$. Thus the algorithm selects a radius $r' \ge 2r(c_{\mathrm{OPT}})$ with probability at least $p = 1 - 2r(c_{\mathrm{OPT}})/\alpha g = 1/f(c_{\mathrm{OPT}})$ and this makes c_{OPT} good. Once c_{OPT} is made good, no more nodes in $V(c_{\mathrm{OPT}})$ are selected. Since the choices of k_v are independent, we get $\mathbb{E}\left[|\mathrm{ALG} \cap V(c_{\mathrm{OPT}})|\right] \le 1/p = f(c_{\mathrm{OPT}})$. ◀

▶ **Lemma 6.4.** *The expected total number of radii used by Algorithm 4 is at most αg.*

Proof. Each node of G is in $V(c_{\mathrm{OPT}})$ for exactly one c_{OPT}. Thus, by Lemma 6.3 and linearity of expectation,

$$\mathbb{E}\left[|\mathrm{ALG}|\right] = \mathbb{E}\left[\sum_{c_{\mathrm{OPT}} \in \mathrm{OPT}} |\mathrm{ALG} \cap V(c_{\mathrm{OPT}})|\right] = \sum_{c_{\mathrm{OPT}} \in \mathrm{OPT}} \mathbb{E}\left[|\mathrm{ALG} \cap V(c_{\mathrm{OPT}})|\right] \le \sum_{c_{\mathrm{OPT}} \in \mathrm{OPT}} f(c_{\mathrm{OPT}})\,.$$

We have, using $\alpha = 2e^2/(e^2-1)$ in the penultimate step,

$$\begin{aligned}\sum_{c_{\mathrm{OPT}} \in \mathrm{OPT}} f(c_{\mathrm{OPT}}) &= \sum_{r=0}^{g-1} \frac{\alpha g}{\alpha g - 2r} < \int_0^g \frac{\alpha g}{\alpha g - 2r} dr = \left[-\frac{\alpha g}{2} \ln(\alpha g - 2r)\right]_0^g \\ &= \frac{\alpha g}{2} \ln\left(\frac{\alpha g}{\alpha g - 2g}\right) = \frac{\alpha g}{2} \ln\left(\frac{\alpha}{\alpha - 2}\right) = \frac{\alpha g}{2} \ln\left(e^2\right) = \alpha g\,.\end{aligned}$$

◀

6.2 High-probability bound

Next, we need to prove that there is a low probability that Algorithm 4 will use significantly more radii than expected. Similar to Section 5 we define a potential Φ_t comparing the number of the algorithm's centers to the expected progress and use martingale bounds to show that the potential is unlikely to be large.

▶ **Definition 6.5.** *Let C_t be a random variable denoting the set of centers of* OPT *made good by the first t centers c_{ALG} placed by the algorithm and*

$$\Phi_t = t - \sum_{c_{OPT} \in C_t} f(c_{OPT}) .$$

Observe that $\Phi_0 = 0$ and that $\Phi_0, \Phi_1, \ldots$ is a sequence of random variables, since C_t depends on the random choices of the values k_v in Algorithm 4. We now prove that the potential is a supermartingale and thus we can use Azuma's inequality to show that Algorithm 4 is unlikely to use more than $(1+\varepsilon)\alpha g$ radii.

▶ **Lemma 6.6.** *$\Phi_0, \Phi_1, \ldots$ is a supermartingale with $|\Phi_t - \Phi_{t-1}| \leq 7$ for all t.*

Proof. Let v be the t-th selected node. Since v is not covered when it is selected, we have $c_{\text{OPT}}(v) \notin C_{t-1}$. Only $c_{\text{OPT}}(v)$ can be made good by a center c_{ALG} placed at v and thus Φ_t is equal to either $\Phi_{t-1} + 1 - f(c_{\text{OPT}}(v))$ or $\Phi_{t-1} + 1$. By Lemma 6.3, the probability p of center $c_{\text{OPT}}(v)$ being made good is at least $1/f(c_{\text{OPT}}(v))$, making $\mathbb{E}[\Phi_t | \Phi_0, \ldots, \Phi_{t-1}] = p(\Phi_{t-1} + 1 - f(c_{\text{OPT}}(v)) + (1-p)(\Phi_{t-1} + 1) = \Phi_{t-1} + 1 - pf(c_{\text{OPT}}(v)) \leq \Phi_{t-1}$ and thus $\Phi_0, \Phi_1, \ldots$ is a supermartingale.

As $\alpha = 2e^2/(e^2-1) > 16/7$ and $r(c_{\text{OPT}}) < g$, we get $f(c_{\text{OPT}}) < \alpha g/(\alpha g - 2g) < 8$. Thus $\Phi_t - \Phi_{t-1}$ is at least $1 - f(c_{\text{OPT}}) \geq -7$ and at most 1, meaning that $|\Phi_t - \Phi_{t-1}| \leq 7$ for all t. ◀

▶ **Lemma 6.7.** *The probability that Algorithm 4 uses at least $(1+\varepsilon)\alpha g$ radii is at most $1/e$.*

Proof. We know that $\Phi_0 = 0$. By Lemma 6.4 we know that $\sum f(c_{\text{OPT}}) \leq \alpha g$. Thus, if the process has not finished before using $T = \lceil (1+\varepsilon)\alpha g \rceil$ radii, the value of Φ_T is at least $\varepsilon \alpha g$. By Lemma 6.6 we know that $\Phi_0, \Phi_1, \ldots$ is a supermartingale with differences at most 7. From Azuma's inequality (Theorem 5.2) for $X_t = \Phi_t$, $\Delta = \varepsilon \alpha g$, $T = \lceil (1+\varepsilon)\alpha g \rceil$ and $\delta = 7$, using also the bounds $g \geq 100/\varepsilon^3$ and $\varepsilon < 1$ we get

$$\Pr[\Phi_T \geq \varepsilon \alpha g] \leq \exp\left(\frac{-(\varepsilon \alpha g)^2}{2 \cdot 7^2 \cdot \lceil (1+\varepsilon)\alpha g \rceil}\right) < \exp\left(\frac{-\varepsilon^2 g}{100}\right) < \frac{1}{e} . \qquad ◀$$

6.3 Bounding the maximum radius

The only possibility for Algorithm 4 to use a radius larger than R is due to the procedure SELECT in which case the larger radii are used one by one in increasing order. In particular, to use a radius larger than $R + b$, the algorithm must, for some x, generate at least $b + x$ values k_v that are at least $R - x$ and thus cause SELECT to pick a radius larger than R at least b times. Using Chernoff bounds, we now show that this event is unlikely for $b = \varepsilon R$.

▶ **Lemma 6.8.** *Conditioned on the event that Algorithm 4 uses no more than $(1+\varepsilon)\alpha g$ radii, Algorithm 4 uses a radius of value at least $(1+\varepsilon)^3 \alpha g$ with probability at most $1/100$.*

Proof. Recall that $R = (1+\varepsilon)^2 \alpha g$ and let $b = \varepsilon R = (1+\varepsilon)^2 \varepsilon \alpha g$. Therefore, $(1+\varepsilon)^3 \alpha g = (1+\varepsilon)R = R + b$. We proceed to show that the probability of Algorithm 4 using radius $\lceil R + b \rceil$ is small.

Let r_{min} be a random variable denoting the minimal radius such that all radii from $\{r_{min}, \ldots, \lceil R + b \rceil\}$ are used by Algorithm 4; if $\lceil R + b \rceil$ is not used, r_{min} is undefined. Algorithm 4 uses a radius at least $R + b$ if and only if r_{min} is defined.

Informally, the proof idea is as follows. We partition this event according to the value of r_{min}, grouping its possible values into blocks of size b (disregarding rounding) so that block i contains values in $(R - ib, R - (i-1)b]$ for $i \in \{1, \ldots, \lceil 1/\varepsilon \rceil\}$. If r_{min} is in block 1, i.e., larger than $R - b$, it must be the case that k_v was chosen larger than $R - b$ at least b times. In general, for block i, k_v must have been chosen to be larger than $R - ib$ at least ib times. This probability decreases geometrically with i; this follows from Chernoff bounds, using also the conditioning in the lemma, which implies that the number of chosen radii is smaller than R by a constant factor $(1 + \varepsilon)$.

For a full version of the proof, please see Appendix B. ◀

6.4 Final algorithm

We combine the preceding algorithms to get the final Algorithm 5.

Algorithm 5 The final algorithm.

Input: Graph $G(V, E)$ and real number $\bar{\varepsilon} > 0$
Output: A burning schedule for G of length at most $(\alpha + \bar{\varepsilon})b(G)$
$\varepsilon \leftarrow (1 + \bar{\varepsilon}/\alpha)^{1/4} - 1$; $\ell \leftarrow 100/\varepsilon^3$ ▷ Note that $\varepsilon \in \Theta(\bar{\varepsilon})$
for $g \leftarrow 1, \ldots, \lfloor |V|/\alpha \rfloor$ **do**
  if $g \leq \ell$ **then**
    Use Algorithm 2 with $z = 4$ to get ALG or Fail
    if ALG was found **then return** ALG
  else
    Use Algorithm 4 to get ALG
    if ALG uses no radius greater than $(1 + \varepsilon)^3 \alpha g - 1$ **then return** ALG
return a trivial solution $\{(v_0, 0), \ldots, (v_{n-1}, n-1)\}$ for $V = \{v_0, \ldots, v_{n-1}\}$

▶ **Theorem 6.9.** *Algorithm 5 is an $(\alpha + \bar{\varepsilon})$-approximation algorithm for any given $\bar{\varepsilon} > 0$ and runs in time $2^{\mathcal{O}(1/\bar{\varepsilon}^3)}\mathrm{poly}(n)$.*

Proof. By Theorem 4.1, Algorithm 2 is guaranteed to succeed if run with $g \geq b(G)$ and it never uses a radius greater than $2.25g - 1$. Thus, if $b(G) \leq \ell$, Algorithm 5 always finds a 2.25-approximation.

If $b(G) > \ell$, either Algorithm 2 succeeds with $g < b(G)$, or we run Algorithm 4 while increasing g. By Lemma 6.7, once $g \geq b(G)$, Algorithm 4 uses more than $(1 + \varepsilon)\alpha g$ radii with probability at most $1/e$. By Lemma 6.8, if Algorithm 4 uses at most $(1 + \varepsilon)\alpha g$ radii, it uses a radius greater than $(1 + \varepsilon)^3 \alpha g - 1$ with probability at most $1/100$. Thus it succeeds with probability at least $1/2$. This probability is independent and increasing for each successive iteration, ensuring that the expected number of failures is at most 2. Thus the expected maximum radius is at most $(1 + \varepsilon)^3 \alpha(b(G) + 2) - 1$. We get $(1 + \varepsilon)^3 \alpha(b(G) + 2) = (1+\varepsilon)^3 \alpha b(G) + 2(1+\varepsilon)^3 \alpha \leq (1+\varepsilon)^3 \alpha g + 50 \leq (1+\varepsilon)^4 \alpha b(G) - 1$, using the value of α for the first inequality and for the last one also the case condition $b(G) > \ell$ and $\ell \geq 100/\varepsilon \geq 51/\varepsilon(1+\varepsilon)^3\alpha$. By the choice of ε in the algorithm, the approximation ratio is at most $(1 + \varepsilon)^4 \alpha = \alpha + \bar{\varepsilon}$.

Since Algorithm 5 iterating over the possible values of g multiplies the time complexity by at most n, the running time is dominated by Algorithm 2, and thus it is bounded by $2^{\mathcal{O}(\ell)}\mathrm{poly}(n)$. Our choice of ε satisfies $\varepsilon = \Theta(\bar{\varepsilon})$ for a small $\bar{\varepsilon} > 0$ (using the fact that $(1 + \varepsilon)^4 \approx 1 + 4\varepsilon$), thus we get $\ell = 100/\varepsilon^3 = \mathcal{O}\left(1/\bar{\varepsilon}^3\right)$ and the bound in the theorem follows. ◀

We note that, instead of using Algorithm 2 for $g \leq \ell$, we can find an optimal solution by exhaustive search in time $n^{\mathcal{O}(g)}$, which is still polynomial. We prefer to use Algorithm 2 to achieve FPT-type running time dependency on ε.

7 Lower bounds for Graph Burning

In this section, we first prove a lower bound of 2 for the approximation ratio of the Graph Burning problem on a graph with edge lengths. We also prove a lower bound of 4/3 for graphs with unit edge lengths.

Both results rely on a reduction from the Dominating Set problem defined as follows.

▶ **Definition 7.1.** *A* dominating set *of an undirected graph $G = (V, E)$ is a set $D \subseteq V$ such that each vertex of G either is in D or has a neighbor in D. In an instance of the Dominating Set problem, we are given an undirected graph and our task is to find a dominating set of the lowest possible size.*

The hardness of approximation of Dominating Set is equivalent to that of Set Cover. After a long line of research in PCP theory, the ultimate result is the following one.

▶ **Theorem 7.2** (Dinur, Steurer [8])**.** *For any $\varepsilon > 0$, there exists no $(1-\varepsilon)\ln n$-approximation algorithm for Dominating Set or Set Cover problems unless $P = NP$.*

In the first reduction, we modify a Dominating Set instance by setting the length of all edges to k for some integer k. We use the fact that in the burning schedule, the centers with radii in $[k, 2k)$ cover exactly their neighbors (due to the edge lengths). Thus a dominating set of size t implies a burning schedule of length $t + k$ in the graph with edge lengths, and a graph burning schedule with length at most $2k$ implies a dominating set of size at most $2k$. By considering a carefully chosen k, this together with a $(2-\varepsilon)$-approximation algorithm for Graph Burning gives an $\mathcal{O}(1)$-approximation algorithm for Dominating Set, contradicting Theorem 7.2.

Technically, we try all values of k and stop at the smallest k that yields a Graph Burning solution of size at most $2k$ and thus a Dominating Set approximation. The number of possible values of k is bounded by $|V|$. Testing only the chosen important values of k used in the proof (i.e., approximately $k = 2t/\varepsilon$ for testing size t Dominating Set) would be sufficient and slightly more efficient, but we prefer to keep the algorithm simple.

▶ **Theorem 7.3.** *For any constant $\varepsilon > 0$, there exists no $(2-\varepsilon)$-approximation algorithm for the Graph Burning problem with arbitrary edge lengths unless $P = NP$. This holds even for the case when restricted to Graph Burning instances with all edge lengths equal.*

Proof. For a contradiction, assume that we have a $(2-\varepsilon)$-approximation algorithm for Graph Burning with arbitrary but equal edge lengths. Let $\sigma = \lceil 1/\varepsilon \rceil$. We give a reduction that results in an approximation algorithm for the Dominating Set problem with an approximation ratio of at most $4\sigma \in \mathcal{O}(1)$.

We use the reduction described above; see Algorithm 6 for a formal description.

First, we claim that the output D is always a dominating set. Indeed, if we stop at a feasible solution with $g \leq 2k$, any $w \in G$ is covered by some center (v_i, i) for $i < 2k$, i.e., the distance between w and v_i is less than $2k$. Since the lengths of the edges are set to k, this means that v_i is a neighbor of or equal to w. In the fallback case, $D = V$ is a dominating set.

Now assume that G has a dominating set $\{w_0, \ldots, w_{t-1}\}$ of size t. We claim that for any given k, the burning number of G with edge lengths is at most $k + t$. Indeed, the set of centers $\{(w_i, k + i) \mid i = 0, \ldots, t-1\}$ augmented by arbitrary centers with radii smaller than k is a required feasible solution.

Algorithm 6 Reduction using edge lengths.

Input: Dominating Set instance $G = (V, E)$
for $k \leftarrow 1, \ldots, \lfloor |V|/2 \rfloor$ **do**
  Set the length of each edge to k
  Run the $(2-\varepsilon)$-approximation algorithm for Graph Burning on G,
    obtaining a feasible solution $\{(v_0, 0), (v_1, 1), \ldots, (v_{g-1}, g-1)\}$ for some g.
  if $g \leq 2k$ **then return** the set $D = \{v_0, v_1, \ldots, v_{g-1}\}$.
return the set $D = V$.

If $t < |V|/4\sigma$, consider $k = 2\sigma t \leq |V|/2$. The choice of k implies that $\varepsilon k \geq 2t$ as $\sigma = \lceil 1/\varepsilon \rceil$. The assumed $(2-\varepsilon)$-approximation algorithm for Graph Burning finds a feasible solution of size at most $(2-\varepsilon)(k+t)$, which is bounded by $(2-\varepsilon)(k+t) < 2k - \varepsilon k + 2t \leq 2k$. Thus the algorithm stops for some $k' \leq k$ with a dominating set D of size at most $2k = 4\sigma t$.

If $t \geq |V|/4\sigma$, the reduction trivially gives a 4σ-approximation. Overall we have obtained a 4σ-approximation for Dominating Set.

By Theorem 7.2, no $\mathcal{O}(1)$-approximation algorithm for the Dominating Set problem exists, unless $P = NP$. This yields the contradiction and the theorem. ◀

We can modify the previous reduction so that, instead of edges of length k, we subdivide them into paths of unit-length edges. As a consequence, when converting the dominating set to a Graph Burning solution, we have to use larger radii to cover the new vertices, which degrades the inapproximability bound. Also, when converting a burning schedule to a dominating set, we cannot use the subdivision points as elements of the dominating set. Instead, we substitute any subdivision point with the two closest original vertices, i.e., the endpoints of the subdivided edge.

▶ **Theorem 7.4.** *For any constant $\varepsilon > 0$, there exists no $(4/3-\varepsilon)$-approximation algorithm for the Graph Burning problem with unit edge lengths, unless $P = NP$.*

Proof. For a contradiction, assume that we have a $(4/3-\varepsilon)$-approximation algorithm for Graph Burning with unit-length edges. Let $\sigma = \lceil 1/\varepsilon \rceil$. We give a reduction that results in an algorithm for Dominating Set with an approximation ratio of at most $8\sigma \in \mathcal{O}(1)$. The reduction follows the outline given above; see Algorithm 7 for a formal description.

Graph G' is of polynomial size, and thus the whole reduction is polynomial. Furthermore, if $u, w \in V$ are vertices of the input graph G, then the distance between u and w in G' is a multiple of $2k$. In particular, if u and w are not equal, their distance in G' is equal to $2k$ if u and w are neighbors in G, and their distance is at least $4k$ otherwise.

We claim that the output D is always a dominating set. Indeed, if we stop at a feasible solution with $g \leq 4k$, any $w \in V$ is covered by some center (v_i, i) for $i < 4k$, i.e., the distance between w and v_i is less than $4k$. The construction of D_i implies that the shortest path between w and v_i contains one of $u \in D_i$. Thus the distance between u and w in G' is less than $4k$ and u and w are neighbors in G. It follows that D is a dominating set. In the fallback case, $D = V$ is a dominating set as well.

Now assume that G has a dominating set $\{w_0, \ldots, w_{t-1}\}$ of size t. We claim that for any given k, the burning number of G' is at most $3k + t$. Indeed, the set of centers $\{(w_i, 3k+i) \mid i = 0, \ldots, t-1\}$ augmented by $3k$ arbitrary centers with radii smaller than $3k$ is a required feasible solution. Each vertex v in G' has distance at most k to the closest vertex in G, which in turn has distance at most $2k$ to a vertex in the dominating set. Thus the balls of radius at least $3k$ at the vertices of the dominating set cover the whole G'.

Algorithm 7 Reduction for unit edge lengths.

Input: Dominating Set instance $G = (V, E)$
for $k = 1, \ldots, |V|$ **do**
  Create G' from G by replacing each edge of G by a path of $2k$ new edges with $2k - 1$ new internal vertices.
  Run the $(4/3 - \varepsilon)$-approximation algorithm for Graph Burning on G', obtaining a feasible solution $\{(v_0, 0), (v_1, 1), \ldots, (v_{g-1}, g-1)\}$ for some g.
  if $g \leq 4k$ **then**
    for $i = 0, 1, \ldots, g-1$ **do**
      if $v_i \in V$ **then** ▷ i.e., v_i is a vertex of the input graph G
        $D_i \leftarrow \{v_i\}$
      else ▷ i.e., v_i is a new internal vertex on an added path
        $D_i \leftarrow \{u, u'\}$ where $u, u' \in V$ are such that v_i is a new vertex on the path in G' that replaced the edge uu' of G.
    return the set $D = D_0 \cup D_1 \cup \cdots \cup D_{g-1}$.
return the set $D = V$.

If $t < |V|/\sigma$, consider $k = \sigma t \leq |V|$. The choice of k implies that $t \leq \varepsilon k$ as $\sigma = \lceil 1/\varepsilon \rceil$. The assumed $(4/3 - \varepsilon)$-approximation algorithm for Graph Burning finds a feasible solution of size at most $(4/3 - \varepsilon)(3k + t)$. We have

$$\left(\frac{4}{3} - \varepsilon\right)(3k + t) \leq 4k - 3\varepsilon k + \frac{4}{3}t \leq 4k - 3t + \frac{4}{3}t < 4k\,.$$

Thus the algorithm stops for some $k' \leq k$ with a dominating set D of size at most $2 \cdot 4k = 8\sigma t$, since each set D_i contains at most two vertices for each center v_i of the burning schedule.

If $t \geq |V|/\sigma$, the reduction trivially gives an σ-approximation. Altogether we have obtained an 8σ-approximation algorithm for Dominating Set.

By Theorem 7.2, no $\mathcal{O}(1)$-approximation algorithm for the dominating set problem exists, unless $P = NP$. This yields the contradiction and the theorem. ◀

References

1 Noga Alon and Joel H. Spencer. *The Probabilistic Method, Third Edition*. Wiley, 2008.

2 Stéphane Bessy, Anthony Bonato, Jeannette Janssen, Dieter Rautenbach, and Elham Roshanbin. Burning a graph is hard. *Discrete Applied Mathematics*, 232:73–87, 2017. doi:10.1016/j.dam.2017.07.016.

3 Anthony Bonato. A survey of graph burning. *Contributions Discret. Math.*, 16(1):185–197, 2021. URL: https://cdm.ucalgary.ca/article/view/71194.

4 Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. How to burn a graph. *Internet Mathematics*, 12, July 2015. doi:10.1080/15427951.2015.1103339.

5 Anthony Bonato and Shahin Kamali. Approximation algorithms for graph burning. In T. V. Gopal and Junzo Watada, editors, *Theory and Applications of Models of Computation - 15th Annual Conference, TAMC 2019, Kitakyushu, Japan, April 13-16, 2019, Proceedings*, volume 11436 of *Lecture Notes in Computer Science*, pages 74–92. Springer, 2019. doi:10.1007/978-3-030-14812-6_6.

6 Anthony Bonato and Thomas Lidbetter. Bounds on the burning numbers of spiders and path-forests. *Theor. Comput. Sci.*, 794:12–19, 2019. doi:10.1016/j.tcs.2018.05.035.

7 Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform k-center problem. *ACM Trans. Algorithms*, 16(4), June 2020. doi:10.1145/3392720.

8 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633. ACM, 2014. doi:10.1145/2591796.2591884.

9 Jesús García-Díaz, Julio César Pérez-Sansalvador, Lil María Xibai Rodríguez-Henríquez, and José Alejandro Cornejo-Acosta. Burning graphs through farthest-first traversal. *IEEE Access*, 10:30395–30404, 2022. doi:10.1109/ACCESS.2022.3159695.

10 Tanmay Inamdar and Kasturi R. Varadarajan. Non-uniform k-center and greedy clustering. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT*, volume 227 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SWAT.2022.28.

11 Xinrui Jia, Lars Rohwedder, Kshiteej Sheth, and Ola Svensson. Towards non-uniform k-center with constant types of radii. In Karl Bringmann and Timothy Chan, editors, *5th Symposium on Simplicity in Algorithms, SOSA@SODA 2022*, pages 228–237. SIAM, 2022. doi:10.1137/1.9781611977066.16.

12 Shahin Kamali, Avery Miller, and Kenny Zhang. Burning two worlds. In Alexander Chatzigeorgiou, Riccardo Dondi, Herodotos Herodotou, Christos A. Kapoutsis, Yannis Manolopoulos, George A. Papadopoulos, and Florian Sikora, editors, *SOFSEM 2020: Theory and Practice of Computer Science - 46th International Conference on Current Trends in Theory and Practice of Informatics, SOFSEM 2020, Limassol, Cyprus, January 20-24, 2020, Proceedings*, volume 12011 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2020. doi:10.1007/978-3-030-38919-2_10.

13 Anjeneya Swami Kare and I. Vinod Reddy. Parameterized algorithms for graph burning problem. In Charles J. Colbourn, Roberto Grossi, and Nadia Pisanti, editors, *Combinatorial Algorithms - 30th International Workshop, IWOCA 2019, Pisa, Italy, July 23-25, 2019, Proceedings*, volume 11638 of *Lecture Notes in Computer Science*, pages 304–314. Springer, 2019. doi:10.1007/978-3-030-25005-8_25.

14 Yasuaki Kobayashi and Yota Otachi. Parameterized complexity of graph burning. *Algorithmica*, 84(8):2379–2393, 2022. doi:10.1007/s00453-022-00962-8.

15 Matej Lieskovský and Jirí Sgall. Graph burning and non-uniform k-centers for small treewidth. In *Approximation and Online Algorithms - 20th International Workshop, WAOA 2022, Potsdam, Germany, September 8-9, 2022, Proceedings*, volume 13538 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2022. doi:10.1007/978-3-031-18367-6_2.

16 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. doi:10.1017/CBO9780511813603.

17 Debajyoti Mondal, N. Parthiban, V. Kavitha, and Indra Rajasingh. APX-hardness and approximation for the k-burning number problem. In Ryuhei Uehara, Seok-Hee Hong, and Subhas C. Nandy, editors, *WALCOM: Algorithms and Computation - 15th International Conference and Workshops, WALCOM 2021, Yangon, Myanmar, February 28 - March 2, 2021, Proceedings*, volume 12635 of *Lecture Notes in Computer Science*, pages 272–283. Springer, 2021. doi:10.1007/978-3-030-68211-8_22.

18 Debajyoti Mondal, Angelin Jemima Rajasingh, N. Parthiban, and Indra Rajasingh. APX-hardness and approximation for the k-burning number problem. *Theor. Comput. Sci.*, 932:21–30, 2022. doi:10.1016/j.tcs.2022.08.001.

A Proof of Theorem 4.1

We first prove the following lemma.

▶ **Lemma A.1.** *If $g \geq b(G)$ then Algorithm 2 always returns a solution.*

Proof. Let $v(S,t)$ be the t-th selected node while the algorithm is using sequence S. Observe that $v(S,t)$ only depends on the initial segment $s_1, \ldots, s_{t-1}$ of S. With this in mind, we show in the next paragraph that there exists $S^* \in \mathbb{S}$ such that s^*_t of S^* is the class of $c_{\text{OPT}}(v(S^*,t))$.

Formally, for $t = 1, 2, \ldots$ we define s_t^* as the class of $c_{\text{OPT}}(v(\bar{S}, t))$ for an arbitrary extension $\bar{S} \in \mathbb{S}$ of $s_1^*, \ldots, s_{t-1}^*$. Such an $\bar{S} \in \mathbb{S}$ exists, as the optimum solution OPT contains at most g centers with no more than $\lceil g/z \rceil$ centers of any given class. If $v(\bar{S}, t)$ is undefined (i.e., all nodes are covered at this point), we choose $s_t^* = \bar{s}_t$. As $v(\bar{S}, t)$ and thus s_t^* depends only on $s_1^*, \ldots, s_{t-1}^*$, which is the initial segment of $\bar{S}$, eventually S^* is the sequence of the classes of centers $c_{\text{OPT}}(v(S^*, t))$.

It follows that using this S^* in the loop makes all centers good, thus all nodes are covered and the algorithm stops with a valid solution. ◀

▶ **Theorem A.2** (Theorem 4.1). *If $g \geq b(G)$ then Algorithm 2 runs in $\mathcal{O}(z^g \text{poly}(n))$-time and achieves an approximation ratio of $(2 + 1/z)$.*

Proof. Lemma A.1 implies that the algorithm returns a solution covering all nodes. We first bound the maximal radius used and thus the approximation ratio. We distinguish two cases.

If $z \geq g$, the algorithm uses $z = g$. Each threshold is used at most once, making the maximal used radius at most $q_z = 2g - 2$, giving a solution with $2g - 1$ many centers, and thus approximation ratio of $2 - 1/g$.

If $z < g$ there are at least $\lfloor 2g/z \rfloor$ radii between any two consecutive thresholds as there are at least $\lfloor 2g/z \rfloor$ integers in $[q_i, q_{i+1})$. We use the same threshold at most $\lceil g/z \rceil$ times by the definition of $\mathbb{S}$. We have $\lceil g/z \rceil \leq \lfloor 2g/z \rfloor$, as $\lceil x \rceil \leq \lfloor 2x \rfloor$ for any $x \geq 1$, while we use the same threshold for at most $\lceil g/z \rceil$ nodes by the definition of $\mathbb{S}$. Thus the radii produced by using a given threshold q_i do not exceed the following threshold q_{i+1}. Furthermore no more than $\lceil g/z \rceil$ nodes use the last threshold $q_z = 2g - 2$, making the maximal used radius at most $2g - 2 + \lceil g/z \rceil \leq 2g + g/z - 1$. Thus the objective value is at most $2g + g/z$ and the ratio $2 + 1/z$.

The runtime bound follows since the number of sequences of length g with numbers from $\{1, \ldots, z\}$ is z^g, and this gives an upper bound on the size of $\mathbb{S}$. ◀

B Proof of Lemma 6.8

▶ **Lemma B.1** (Lemma 6.8). *Conditioned on the event that Algorithm 4 uses no more than $(1+\varepsilon)\alpha g$ radii, Algorithm 4 uses a radius of value at least $(1+\varepsilon)^3 \alpha g$ with probability at most $1/100$.*

Proof. Recall that $R = (1 + \varepsilon)^2 \alpha g$ and let $b = \varepsilon R = (1 + \varepsilon)^2 \varepsilon \alpha g$. Therefore, $(1 + \varepsilon)^3 \alpha g = (1 + \varepsilon)R = R + b$. We proceed to show that the probability of Algorithm 4 using radius $\lceil R + b \rceil$ is small.

Let r_{min} be a random variable denoting the minimal radius such that all radii from $\{r_{min}, \ldots, \lceil R + b \rceil\}$ are used by Algorithm 4; if $\lceil R + b \rceil$ is not used, r_{min} is undefined. Algorithm 4 uses a radius at least $R + b$ if and only if r_{min} is defined.

Informally, the proof idea is as follows. We partition this event according to the value of r_{min}, grouping its possible values into blocks of size b (disregarding rounding) so that block i contains values in $(R - ib, R - (i - 1)b]$ for $i \in \{1, \ldots, \lceil 1/\varepsilon \rceil\}$. If r_{min} is in block 1, i.e., larger than $R - b$, it must be the case that k_v was chosen larger than $R - b$ at least b times. In general, for block i, k_v must have been chosen to be larger than $R - ib$ at least ib times. This probability decreases geometrically with i; this follows from Chernoff bounds, using also the conditioning in the lemma, which implies that the number of chosen radii is smaller than R by a constant factor $(1 + \varepsilon)$.

For $i \in \{1, \ldots, \lceil 1/\varepsilon \rceil\}$, we define A_i as the event that r_{min} is defined and i is the smallest integer such that $R - ib < r_{min}$. Exactly one of the events A_i occurs whenever r_{min} is defined. The event A_i implies that the algorithm randomly generated k_v greater than $R - ib$

at least ib times, as by definition of r_{min}, choosing $k_v < r_{min}$ does not allow SELECT to generate a radius equal to or larger than r_{min} and at the same time all at least ib radii between $r_{min} \le R - (i-1)b$ and $\lceil R + b \rceil$ are used.

By the conditioning in the lemma, we assume that the algorithm uses no more than $(1+\varepsilon)\alpha g$ radii. Let p be the probability that upon processing v, the algorithm generates k_v larger than $R - ib$. We have $p \le ib/R$, as out of $\lfloor R \rfloor + 1 > R$ options for k_v, there are at most ib values larger than $R - ib$. Consider X which is a sum of $J = \lfloor (1+\varepsilon)\alpha g \rfloor$ independent indicator random variables, each equal to 1 with probability ib/R. We have $Pr[A_i] \le Pr[X \ge ib]$, as X overestimates the number of k_v larger than $R - ib$ in two ways: First, we possibly increase the number of trials and second, we increase the probability of the indicator variable to be 1 from p to ib/R.

Let $\mu = \mathbb{E}[X]$. We have $\mu = \lfloor (1+\varepsilon)\alpha g \rfloor \cdot ib/R \le (1+\varepsilon)\alpha g ib/R = ib/(1+\varepsilon)$. As $g \ge 100/\varepsilon^3 \ge 100$, the rounding error is small, namely we have $\mu \ge \frac{99}{100}(1+\varepsilon)\alpha g \cdot ib/R = \frac{99}{100}\varepsilon(1+\varepsilon)i\alpha g$, using also $b = \varepsilon R$. Now we use Chernoff bound and get

$$\begin{aligned}\Pr[A_i] \le \Pr[X \ge ib] \le \Pr[X \ge (1+\varepsilon)\mu] &\le \exp\left(\frac{-\varepsilon^2\mu}{3}\right) \le \exp\left(\frac{-\frac{99}{100}\varepsilon^3(1+\varepsilon)i\alpha g}{3}\right) \\ &\le \exp\left(-\frac{3}{4}\varepsilon^3 ig\right) \le \exp(-75i)\,,\end{aligned}$$

where we used $\frac{99}{100}(1+\varepsilon)\alpha/3 > 3/4$. We obtain that the probability of using radius at least $R + b$ is at most

$$\Pr\left[\bigcup_i A_i\right] \le \sum_{i=1}^{\infty} \exp(-75i) \le 1/100\,. \qquad \blacktriangleleft$$