Deterministic Approximation Algorithm for Graph Burning

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

Abstract

Graph Burning models a contagion spreading in a network as a process such that in each step one node is infected and also the infection spreads to all neighbors of previously infected nodes. Formally, the burning number $b(G)$ of a given graph $G = (V, E)$, possibly with edge lengths, is the minimum number g such that there exists a sequence of nodes $v_1, \dots, v_g$ satisfying the property that for each $w \in V$ there exists $i \in \{1, \dots, g\}$ so that the distance between w and v_i is at most $g - i$.

We present an elegant deterministic 2.314-approximation algorithm for the Graph Burning problem on general graphs with arbitrary edge lengths. This algorithm matches the approximation ratio of the previous randomized 2.314-approximation algorithm and improves on the previous deterministic 3-approximation algorithm.

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

Keywords and phrases Graph Algorithms, Approximation Algorithms, Graph Burning

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.108

Funding *Matej Lieskovský*: Supported by GAUK project 234723, GA ČR project 19-27871X, and the grant SVV–2025–260822.

Acknowledgements We wish to thank our reviewers and colleagues for their insightful feedback.

1 Introduction

Graph Burning was introduced by Bonato et al. [2, 1] as a model of a contagion or information spreading in social networks. It demonstrates how strategically selecting a sequence of nodes to “infect” while letting this infection spread over time may reach the entire network relatively quickly. Formally, we study the following process, where *burned* nodes in a graph represent the infected part of the network.

We are given a graph G and a sequence S of length g where the individual elements $S_1, \dots, S_g$ of the sequence are either nodes of the graph or “null”. At time $t = 0$, no nodes are burned. At time $t = 1$, we burn node S_1. At each time step $t > 1$, all neighbors of already burned nodes also become burned and we also burn node S_t unless S_t is “null”. If all nodes are burned at time $t = g$, we call S a *burning sequence* of G of length g. The burning number $b(G)$ of a graph G is the length of its shortest burning sequence.

We can also view a burning sequence as a set of centers where the center (v, r) corresponds to S_{g-r} being the node v, ensuring that any node w with distance $d(w, v) \le r$ is burned. This generalizes naturally in the presence of arbitrary edge lengths and showcases the relation of Graph Burning to the k-Center problem. We call r the *radius* of center (v, r).

We know that $b(G) \le |V(G)|$ holds for all graphs. If we prove that some algorithm finds a valid burning sequence of length g in polynomial time whenever $g \ge \lfloor \rho\, b(G) \rfloor$ for some ρ, then we can create a ρ-approximation algorithm for the Graph Burning problem by trying all possible values of g. For the rest of this paper, we talk about these algorithms that take the target value of g as input and take as granted that they imply the corresponding approximation algorithms.

1.1 Our results

We present a simple α-approximation algorithm for the Graph Burning problem on graphs with arbitrary edge lengths where $\alpha = 2e^2/(e^2-1) < 2.314$. We use this value of α throughout the paper.

1.2 Previous results

Bonato and Kamali [3] adapted a well-known greedy 2-approximation algorithm for the k-Center problem into a $(3 - 2/b(G))$-approximation algorithm for Graph Burning. A direct greedy algorithm was given by García-Díaz, Pérez-Sansalvador, and Rodríguez-Hénríquez [5]. We are not aware of any better deterministic algorithms for general graphs.

Lieskovský, Sgall, and Feldmann [11] gave a randomized $(\alpha + \varepsilon)$-approximation algorithm for Graph Burning as well as a pair of lower bounds – 4/3 for graphs with unit-length edges and 2 for graphs with edge lengths. Their algorithm repeatedly selects an arbitrary uncovered node and uses randomization to select a radius, creating a new center. Martingales are used to estimate the number of centers needed and a collision resolution method inspired by linear probing prevents the creation of multiple centers with the same radius.

Approximation algorithms and approximation schemes were developed for various special graph classes by Bonato and Kamali [3], Bonato and Lidbetter [4] and others, ultimately resulting in a PTAS for graphs of small treewidth by Lieskovský and Sgall [10].

2 Definitions and algorithm

In this section we first establish some fundamental concepts and then describe our algorithm. The concepts established in this section will be crucial for the analysis of our algorithm in the following two sections.

2.1 Fundamental concepts

For the analysis of our algorithm, we fix a graph G and some optimal solution OPT, which we will view as set of centers (v, r) with radii from 0 to $b(G) - 1$. In the interest of clarity we will, for the rest of this paper, use "centers" exclusively to refer to centers in this optimal solution while the algorithm will be modifying a "sequence" of nodes of some given length g, until it becomes a valid burning sequence of the graph.

We assign each node $w \in V$ to a unique center $c_{\text{OPT}}(w)$ 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 center (v, r) we denote the set of nodes assigned to it as $V(v, r)$. We say that a center (v, r) is *covered* by a sequence S if all nodes in $V(v, r)$ are burned by the sequence.

A central part of our algorithm is the process for modifying the sequence. We view the sequence S as a queue of length g and use the procedure push(v, S) to add v to the front of the sequence and remove the last element. Thus if $S = S_1, S_2, S_3, \ldots, S_{g-1}, S_g$, then push$(v, S) = v, S_1, S_2, S_3, \ldots, S_{g-1}$.

Our algorithm works in *cycles*, selecting one unburned node and pushing it into the sequence in each cycle. Note that cycles are distinct from the time steps in the definition of the Graph Burning problem. We number the cycles starting from 0. We say that a center (v, r) is *serviced* whenever any node in $V(v, r)$ is selected by the algorithm.

Suppose that a $w \in V(v, r)$ is selected in cycle i. At the start of cycle $i + 1$, w is at the start of the burning sequence, ensuring that all nodes within distance $g - 1$ from it are burned by the sequence. With each further cycle, the node is moved to later position in the sequence, reducing the area it ensures the burning of.

▶ **Observation 1.** *If center (v, r) is serviced in cycle i, then it will be covered at the start of cycles $i + 1$ to $i + g - 2r$.*

2.2 Our algorithm

The algorithm is given a graph G and a guess g for the burning number. It starts with a sequence of length g where all elements are "null", which obviously leaves all nodes unburned. While any such unburned nodes remain, the algorithm selects one of them arbitrarily and pushes it into the sequence. When no unburned nodes remain, the sequence S is a valid burning sequence of G.

Algorithm 1 Deterministic Approximation Algorithm for Graph Burning.

```
1: function BURN(graph $G$, $g \in \mathbb{N}$)
2:     $S \leftarrow$ queue containing $g$ "null" elements
3:     while an unburned node exists do
4:         select arbitrary unburned node $v$
5:         $S \leftarrow$ push$(v, S)$
6:     return $S$
```

3 Simple results

In this section, we analyse our algorithm at three different levels of detail, resulting in gradually improving approximation ratios. We also show a general observation about the Graph Burning problem in Subsection 3.3. The results from this section will play a minor role in the handling of some special cases by the final analysis of our algorithm in Section 4.

3.1 Proof of 3-approximation

This simple proof matches the previous result of Bonato and Kamali.

▶ **Lemma 2.** *Assuming $g \geq 3b(G) - 2$, the algorithm always concludes in at most $b(G)$ cycles.*

Proof of Lemma 2. Given that centers have radius at most $b(G) - 1$, any center serviced before cycle $b(G)$ will, by Observation 1, still be covered at the start of cycle $b(G)$. Given that there are at most $b(G)$ centers, the algorithm concludes no later than at the start of cycle $b(G)$ as all centers will be covered. ◀

3.2 Easy improvement

Observe what might happen if we were to run the algorithm with $g = 3b(G) - 3$. If the algorithm services the center with radius $b(G) - 1$ in cycle 0, then it is possible that this center is no longer covered at the start of cycle $b(G)$, preventing the algorithm from concluding. Fortunately, this can be fixed by running the algorithm for one more cycle as the second largest center, with radius $b(G) - 2$, can stop being covered no earlier than at the start of cycle $b(G) + 2$

▶ **Lemma 3.** *Assuming $g \geq 3b(G) - 3$, the algorithm always concludes in at most $b(G) + 1$ cycles.*

Proof of Lemma 3. Let us separate the 1 "large" center with $r = b(G) - 1$ and the $b(G) - 1$ "small" centers with $r \leq b(G) - 2$. Any small serviced center will be covered at the start of cycle $b(G) + 1$.

At the start of cycle $b(G) - 1$, at least $b(G) - 1$ centers have been serviced and are covered, leaving at most one center uncovered. Over the next 2 cycles, only the single large center can become uncovered. Thus the algorithm concludes no later than at the start of cycle $b(G) + 1$ as all centers will be covered. ◀

3.3 General observation

Note that slight improvements such as Lemma 3 are generally easy for Graph Burning, as demonstrated by the following observation:

▶ **Observation 4.** *If, for some $\rho \geq 2$, there exists an algorithm $\mathcal{A}$ that, assuming $g \geq \lfloor \rho\, b(G) \rfloor$, always finds a burning sequence of length g in $O(f(n))$ time, then, for any $k \leq b(G)$, there exists an algorithm $\mathcal{B}$ that always finds a burning sequence of length $g - k$ in $O(n^k f(n))$ time.*

Proof of Observation 4. Algorithm $\mathcal{B}$ is given a graph G and a guessed burning number g. For each of the $O(n^k)$ possible length-k prefixes of a burning sequence of length $g - k$, it uses algorithm $\mathcal{A}$ to look for a burning sequence of length $g - 2k$ for the uncovered part of the graph.

One of those prefixes is $P = v_1, v_2, v_3, \ldots, v_k$ such that all $(v_i, b(G) - i)$ are centers of OPT. Since $g - k \geq b(G)$, these k centers are covered by any burning sequence of length $g - k$ that starts with prefix P. The uncovered part of the graph thus has a burning number at most $b(G) - k$ and algorithm $\mathcal{A}$ will be able to find a burning sequence of length $g - 2k$. ◀

3.4 More significant improvement

In order to motivate further exploration of our algorithm, let us now show that our algorithm is indeed capable of finding burning sequences of length $\rho\, b(G)$ where $\rho < 3$. In particular, we demonstrate that $g \geq \frac{8}{3} b(G)$ is sufficient and the algorithm concludes in $\left\lceil \frac{4}{3} b(G) \right\rceil$ cycles.

We use Observation 1 to show how long each center stays covered after being serviced. At the start of each cycle of our algorithm, each center of OPT is in one of the following three states.

1. Untouched - this center has not been serviced yet and might thus be uncovered.
2. Covered - this center has been serviced in the last $g - 2r$ cycles and thus is covered.
3. Expired - this center has last been serviced more than $g - 2r$ cycles ago and thus might not be covered any more.

For the purposes of our analysis, we will divide the cycles of the algorithm into two phases, with each phase shorter than $b(G)$ cycles. During the first phase, the algorithm is only servicing untouched centers. When sufficiently many cycles have passed for a center to possibly become expired, the algorithm enters the second phase. During the second phase, the algorithm services the remaining untouched centers and any expired centers. The second phase concludes before any center can expire for the second time and we show that all centers must be covered.

We divide the centers into a smaller number of large centers and a larger number of small centers. Small centers are the centers with $r < \frac{2}{3} b(G)$. When serviced, these remain covered for $> \frac{4}{3} b(G)$ cycles, which is at least the number of cycles used by the algorithm. Thus they do not become expired. Large centers are the centers with $r \geq \frac{2}{3} b(G)$. When serviced, these remain covered until the end of the phase, but might expire during the second phase if covered during the first phase.

After carefully setting the length of the phases, we show that sufficiently many small centers must be serviced during the first phase so that any untouched small centers and all large centers can be serviced during the second phase. This ensured that all centers are covered even if all the large centers have expired.

▶ **Lemma 5.** *Assuming $g \geq \frac{8}{3}b(G)$, the algorithm always concludes in at most $\lceil \frac{4}{3}b(G) \rceil$ cycles.*

Proof of Lemma 5. A center (v, r) is called large if $r \geq \frac{2}{3}b(G)$. Note that there are at most $\lfloor \frac{1}{3}b(G) \rfloor$ large centers. The remaining centers are small; they have radius $r < \frac{2}{3}b(G)$ and there are at most $\lceil \frac{2}{3}b(G) \rceil$ small centers. The first phase consists of $\lceil \frac{2}{3}b(G) \rceil$ cycles and the second phase consists of $\lfloor \frac{2}{3}b(G) \rfloor$ cycles.

All centers have $r \leq b(G) - 1$. By Observation 1, any center serviced during the first phase is still covered at the start of cycle $\lceil \frac{2}{3}b(G) \rceil$. Thus, at the end of the first phase, $\lceil \frac{2}{3}b(G) \rceil$ centers have been serviced and are covered, leaving at most $\lfloor \frac{1}{3}b(G) \rfloor$ uncovered centers.

Small centers have $r < \frac{2}{3}b(G)$ and thus any serviced small center will still be covered at the start of cycle $\lceil \frac{4}{3}b(G) \rceil$. Thus, during the second phase, only the large centers can become uncovered and there are at most $\lfloor \frac{1}{3}b(G) \rfloor$ large centers. In total, no more than $\lfloor \frac{2}{3}b(G) \rfloor$ centers need to be serviced during the second phase, and the algorithm concludes no later than at the end of the second phase. ◀

4 2.314-approximation ratio

The entirety of this section is devoted to the proof of the following theorem, which is the main result of this paper.

▶ **Theorem 6.** *Assuming $g \geq \lfloor \alpha b(G) \rfloor$, the algorithm always concludes in at most $O((b(G))^2)$ cycles.*

First, we focus on a single center (v, r). In an approach inspired by problems such as Bamboo Garden [7][9] and Pinwheel Scheduling [8][6], we imagine a timer that starts at 1, is reset to zero whenever the algorithm services the center, and otherwise increases with every cycle so that the timer is ≥ 1 whenever the center might be uncovered.

Second, we look at all the centers of OPT and consider the sum of their timers. This sum starts at $b(G)$ and, if g is large enough, decreases with every cycle. Given that all centers must be covered if the sum is less than 1, we proceed to show that $g \geq \lfloor \alpha b(G) \rfloor$ is indeed large enough, completing the proof.

Proof of Theorem 6. Let us define a potential $\Phi_i(v, r)$ for center (v, r) at the start of the i-th cycle as follows:

$$\Phi_0(v, r) = 1 \tag{1}$$

$$\Phi_{i+1}(v, r) = \begin{cases} 0 & \text{if center } (v, r) \text{ was serviced} \\ & \text{during the } i\text{-th cycle} \\ \Phi_i(v, r) + \dfrac{1}{g - 2r} & \text{otherwise} \end{cases} \tag{2}$$

Observe that, if $\Phi_i(v, r) < 1$, then center (v, r) is covered at the start of the i-th cycle.

We now define the overall potential at the start of the i-th cycle as the sum of potentials for all centers of OPT and observe that, whenever $\Phi_i < 1$, all centers must be covered.

$$\Phi_i = \sum_{(v,r) \in \text{OPT}} \Phi_i(v, r)$$

We claim that if center (v, r) is serviced in cycle i, then

$$\Phi_{i+1} = \Phi_i + \Delta - \left(1 + \frac{1}{g-2r}\right)$$

where

$$\Delta = \sum_{r=0}^{b(G)-1} \frac{1}{g-2r}.$$

The positive term Δ accounts for the increase of the potential of all centers. The negative term $(1 + 1/(g-2r))$ accounts for the change of the potential of the serviced center (v, r). This is because $\Phi_i(v, r) \geq 1$ must be true for (v, r) to be serviced in cycle i and $\Phi_{i+1}(v, r) = 0$ if (v, r) was serviced in cycle i, which means that the potential is not only decreased by 1, but also does not increase by $1/(g-2r)$ already included in Δ.

We will show that $\Delta \leq 1$ for $g \geq \lfloor \alpha b(G) \rfloor$. To see that this implies the theorem, first observe that $1 + 1/(g-2r) \geq 1 + 1/g$ and thus we will get $\Phi_{i+1} \leq \Phi_i - 1/g$. Since $\Phi_0 = b(G)$, algorithm concludes in at most $g \cdot b(G)$ cycles. For $g \leq \frac{8}{3}b(G)$ the theorem follows directly, while for $g > \frac{8}{3}b(G)$ we use Lemma 5 to get the conclusion of the theorem.

It remains to prove that $\Delta \leq 1$. The value of α was set to satisfy the following integral equation:

$$\int_0^{b(G)} \frac{1}{\alpha b(G) - 2x}\, dx = \left[-\frac{\ln(\alpha b(G) - 2x)}{2}\right]_{x=0}^{b(G)} = \frac{\ln(\alpha) - \ln(\alpha - 2)}{2} = 1.$$

We express Δ as a similar integral:

$$\Delta = \sum_{r=0}^{b(G)-1} \frac{1}{g-2r} = \int_0^{b(G)} \frac{1}{g - 2\lfloor x \rfloor}\, dx.$$

To complete the proof, it is now sufficient to show that the following holds:

$$\int_0^{b(G)} \frac{1}{g - 2\lfloor x \rfloor}\, dx \leq \int_0^{b(G)} \frac{1}{\alpha b(G) - 2x}\, dx.$$

We first observe that this holds whenever $g \geq \alpha b(G)$, using a straightforward substitution for g and a simple upper bound on $\lfloor x \rfloor$. This makes it possible for us to get an α-approximation algorithm by using Observation 4.

Finally, we give a full proof of $\Delta \leq 1$ for all $g \geq \lfloor \alpha b(G) \rfloor$. We separate the integration into intervals where $2\lfloor x \rfloor$ is constant, seeking to prove for all natural $a < b(G)$:

$$\int_a^{a+1} \frac{1}{g - 2\lfloor x \rfloor}\, dx \leq \int_a^{a+1} \frac{1}{\alpha b(G) - 2x}\, dx.$$

The following trivially holds:

$$\int_a^{a+1} \frac{1}{g - 2\lfloor x \rfloor}\, dx = \int_a^{a+1} \frac{1}{g - 2a}\, dx = \frac{1}{g-2a}.$$

Consider the function $f(z) = 1/(g + 1 - 2z)$. It is convex on $(-\infty, (g+1)/2)$ and thus for every $z < b(G) \leq g/2$ we have $f(z) \leq \int_{z-1/2}^{z+1/2} f(x)\, dx$. Using this for $z = a + 1/2$ we get the following:

$$\frac{1}{g-2a} = f(a + 1/2) \leq \int_a^{a+1} f(x)\, dx = \int_a^{a+1} \frac{1}{g+1-2x}\, dx \leq \int_a^{a+1} \frac{1}{\alpha b(G) - 2x}\, dx$$

where the last inequality follows from $g + 1 \geq \lfloor \alpha b(G) \rfloor + 1 \geq \alpha b(G)$. This concludes the proof of $\Delta \leq 1$ and thus the proof of Theorem 6. ◀

5 Conclusion

We have shown a simple deterministic algorithm for Graph Burning that achieves good approximation ratios when thoroughly analysed. Despite this, a significant gap remains between our result and the lower bounds of [11].

References

1 Anthony Bonato, Jeannette Janssen, and Elham Roshanbin and. How to burn a graph. *Internet Mathematics*, 12(1-2):85–100, 2016. `doi:10.1080/15427951.2015.1103339`.

2 Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. Burning a graph as a model of social contagion. In Anthony Bonato, Fan Chung Graham, and Paweł Prałat, editors, *Algorithms and Models for the Web Graph*, volume 8882, pages 13–22, Cham, December 2014. Springer International Publishing. `doi:10.1007/978-3-319-13123-8_2`.

3 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*, pages 74–92, Cham, 2019. Springer International Publishing. `doi:10.1007/978-3-030-14812-6_6`.

4 Anthony Bonato and Thomas Lidbetter. Bounds on the burning numbers of spiders and path-forests. *Theoretical Computer Science*, 794:12–19, 2019. Special Issue on Theory and Applications of Graph Searching. `doi:10.1016/j.tcs.2018.05.035`.

5 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`.

6 Leszek Gąsieniec, Benjamin Smith, and Sebastian Wild. *Towards the 5/6-Density Conjecture of Pinwheel Scheduling*, pages 91–103. Society for Industrial and Applied Mathematics, 2022. `doi:10.1137/1.9781611977042.8`.

7 Felix Höhne and Rob van Stee. A 10/7-Approximation for Discrete Bamboo Garden Trimming and Continuous Trimming on Star Graphs. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.APPROX/RANDOM.2023.16`.

8 Akitoshi Kawamura. Proof of the density threshold conjecture for pinwheel scheduling. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, pages 1816–1819, New York, NY, USA, 2024. Association for Computing Machinery. `doi:10.1145/3618260.3649757`.

9 John Kuszmaul. Bamboo trimming revisited: Simple algorithms can do well too. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '22, pages 411–417, New York, NY, USA, 2022. Association for Computing Machinery. `doi:10.1145/3490148.3538580`.

10 Matej Lieskovský and Jiří Sgall. Graph burning and non-uniform k-centers for small treewidth. In Parinya Chalermsook and Bundit Laekhanukit, editors, *Approximation and Online Algorithms*, pages 20–35, Cham, 2022. Springer International Publishing.

11 Matej Lieskovský, Jiří Sgall, and Andreas Emil Feldmann. Approximation Algorithms and Lower Bounds for Graph Burning. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.APPROX/RANDOM.2023.9`.