Improved Approximation Algorithm for Graph Burning on Trees

Rahul Kumar Gautam, Anjeneya Swami Kare, and Durga Bhavani S.

School of Computer and Information Sciences,
University of Hyderabad,
Hyderabad, India
19mcpc06@uohyd.ac.in, askcs@uohyd.ac.in, sdbcs@uohyd.ernet.in

Abstract. Given a graph \( G = (V,E) \), the problem of Graph Burning is to find a sequence of nodes from \( V \), called burning sequence, in order to burn the whole graph. This is a discrete-step process, in each step an unburned vertex is selected as an agent to spread fire to its neighbors by marking it as a burnt node. A node that is burnt spreads the fire to its neighbors at the next consecutive step. The goal is to find the burning sequence of minimum length. The Graph Burning problem is NP-Hard for general graphs and even for binary trees. A few approximation results are known, including a 3-approximation algorithm for general graphs and a 2-approximation algorithm for trees. In this paper, we propose an approximation algorithm for trees that produces a burning sequence of length at most \( \lfloor 1.75b(T) \rfloor + 1 \), where \( b(T) \) is length of the optimal burning sequence, also called the burning number of the tree \( T \). In other words, we achieve an approximation factor of \( (\lfloor 1.75b(T) \rfloor + 1)/b(T) \).

1 Introduction

In a public information campaign, it is important to choose people who can spread authentic information within, say, a village network in the least possible amount of time. This problem is modeled as Graph Burning problem which is defined as a discrete step sequential process. One node is selected at each time-step \( t \) as a source of fire which burns all of its neighbors at the next time step \( t+1 \). Another node is picked as the source of fire at time \( t+1 \), and the process of burning continues till all the nodes of the graph are burned. The nodes that are burned, can burn their neighbours at the next time step. That is, the information can pass from people who have been informed in the previous time step. The sequence of nodes which are selected as sources of fire form a burning sequence. An informed node remains in informed or burned state throughout the process. The goal is to compute the burning sequence of minimum length. The length of the optimal burning sequence is called the burning number of the graph, denoted by \( b(G) \). The Graph Burning problem is similar to the \( k \)-centre problem \([6]\), in which all the sources of fire are selected at the first step itself.

Let \( G(V,E) \) be a finite, connected, unweighted and undirected graph having set of vertices \( V \) and edges \( E \). Let \( S = \{v_0, v_1, v_2, \cdots v_{b-1}\} \) be a burning sequence.
At time $t_0$, the vertex $v_0$ is burnt. At time $t_1$ the vertex $v_1$ is burnt and the neighbors of vertex $v_0$ are also burnt. Similarly at time $t_i$, the vertex $v_i$ is burnt and all the vertices which caught fire at the time $t_{i-1}$ will burn their unburned neighbors. The process continues until all the nodes of the graph are burnt.

2 Related Work

Bonato et al. [2] proposed the Graph Burning problem. They studied characteristics and proposed bounds for the Graph Burning problem. Bessy et al. [17] proved that the Graph Burning problem is NP-Complete for general graphs, binary trees, spider graphs, and path forests. Bessy et al. [1] gave upper bounds for Graph Burning for connected graphs and trees.

Parameterized algorithms for the Graph Burning problem were studied in [10,11]. The Graph Burning problem was studied for different classes of graphs such as spider graphs [3], generalized peterson graphs [16], dense graphs [9] and theta graphs [13]. A generalization of Graph Burning problem called $k$-burning problem is studied in [14]. Further, several heuristics [18,5,7] have been proposed for the Graph Burning problem.

Bonato et al. [3] proposed a 3-approximation algorithm for general graphs, a 2-approximation for trees and a 1.5-approximation algorithm for disjoint paths. Bonato et al. [3] also showed that the Graph Burning problem can be solved in polynomial time on path forests, where the number of paths is constant. Further, they proposed approximation schemes for the path forests where the number of paths is not constant. Recently, Diaz et al. [8] proposed an approximation algorithm with approximation factor $3 - 2/b(G)$ for general graphs.

In this paper, we propose an approximation algorithm for the Graph Burning on trees. The proposed approximation algorithm produces a burning sequence of length at most $\lfloor 1.75b(T) \rfloor + 1$. In other words, we achieve an approximation factor of $(1.75b(T)) + 1)/b(T)$, where $b(T)$ is the burning number of the tree $T$. We have also implemented the existing approximation algorithms and the proposed approximation algorithm to compared their performance on the real world datasets and on randomly generated trees. The proposed approximation algorithm outperforms the existing approximation algorithms in most of the cases.

Rest of the paper is organized as follows: In Section 3, we discuss the back-ground of the existing 2-approximation algorithm. In Section 4, we discuss the proposed approximation algorithm. In Section 6, we discuss the details of the results. We conclude with Section 7.

3 Approximation Algorithm

Bonato et al. [3] presented a 2-approximation algorithm for the Graph Burning on trees. In their paper [3], they referred the algorithm as $\text{BURN-\text{GUESS-TREE}}(T, b)$. It returns a burning sequence of length at most $2b$ or it returns $\text{BAD-GUESS}$. If the algorithm returns $\text{BAD-GUESS}$, they proved that the $b(T) > b$. Let $b$
be the minimum number for which \textsc{burn-guess-tree}(T, b) returns a burning sequence of length $2b$. As \textsc{burn-guess-tree}(T, b - 1) returns \textsc{bad-guess}, the burning number $b(T) \geq b$. So the algorithm produces a 2-approximation solution for the \textit{Graph Burning} on trees.

**Definition 1 (From [3])** A ‘b-site partition’ of $V$ is a set of at most $b$ vertices, called $b$ ‘sites’, so that every vertex of the tree is within distance $b$ of its closest site.

**Lemma 1 (From [3])** If \textsc{burn-guess-tree}(T, $b - 1$) returns \textsc{bad-guess} then the tree $T$ does not admit a $b$-site partition and hence $b(T) \geq b + 1$.

**Theorem 2 (From [3])** There is a polynomial time 2-approximation algorithm for the \textit{Graph Burning} problem on trees.

4 Improved Approximation Algorithm

The outline of the approximation algorithm is: 1) Firstly we cut the tree into equal height subtrees. 2) We merge pair of subtrees into a single subtree. 3) The center vertex of each subtree is placed in the burning sequence at suitable index so that the entire tree is burned. Based on the number of merge operations and the number of subtrees, either we achieve the approximate solution or we get a lower bound.

The main procedure \textsc{burn-guess-tree}(T, $b$) of our approximation algorithm is shown in Algorithm 1. The algorithm takes a tree $T$ and a positive integer $b$ as input and returns a burning sequence or \textsc{bad-guess}. In the lines of Bonato’s 2-approximation algorithm: If the algorithm returns a \textsc{bad-guess}, then we prove that the burning number of the tree is at least $b + 1$, otherwise the algorithm returns a burning sequence of length at most $\lfloor 1.75b \rfloor + 1$. If $b$ is the minimum number for which \textsc{burn-guess-tree}(T, $b$) returns a burning sequence, then $bn(T) \geq b$ and hence we achieve the approximation factor of $(\lfloor 1.75b \rfloor + 1)/b$.

The algorithm calls two procedures \textsc{get-subtrees} and \textsc{merge-and-burn}. The \textsc{get-subtrees} procedure cuts the tree $T$ into $b$ height subtrees. The \textsc{merge-and-burn} procedure tries to merge a pair of subtrees into a single subtree and generates the burning sequence.

4.1 Cutting of the Tree

If $T$ is a tree, we remove all the pendant vertices from the tree. Let the resulting tree be $T^1$. We call $T^1$ as a 1-cutting tree. Having cut all pendant vertices of $T^1$, we get a tree $T^2$. We call $T^2$ as a 2-cutting tree. Having cut all pendant vertices of $T^2$, we get a tree $T^3$. We call $T^3$ as a 3-cutting tree. So after $b$ cutting the resulting tree is $T^b$. We call $T^b$ as a $b$-cutting tree of $T$. The procedure is given in Algorithm 2. Let $S = \{v_1, v_2, v_3, \ldots, v_x\}$ be the set of pendant vertices in the tree $T^b$. For each node $v_i \in S$ there is a subtree $T_{v_i}$ of height $b$ in $T$. And for each
Case 2. We check if there is any unprocessed subtree $RS[i]$ such that $d(RS[i].\text{root}, RS[j].\text{root}) \leq b + 1$. If such a subtree exists, we merge the subtrees $RS[i]$ and $RS[j]$. The merge operation produces a new subtree $T_{ij}$ which is the union of subtrees $T_{v_i}, T_{v_j}$ and the shortest path between the subtree roots $v_i$ and $v_j$. The center vertex of the shortest path between the subtree roots $v_i$ and $v_j$ will be the root of new subtree with radius $r' = (d(v_i, v_j) + 2b)/2$. The minimum burn radius $r \in BR$ such that $r \geq r'$ is allocated to the merged subtree $T_{ij}$ and the number $r$ is removed from $BR$. The center vertex of the subtree $T_{ij}$ is placed in the burning sequence at $BS[[1.75b] - r]$. If no such radius $r$ is available in $BR$, we do not merge subtrees $RS[i]$ and $RS[j]$. The subtree $RS[i]$ will be burned independently as described in Case 1.

Case 2. There is no subtree $RS[j]$ such that $d(RS[i].\text{root}, RS[j].\text{root}) \leq b + 1$ or the merge was unsuccessful is Case 1.

4.2 Merging

The merge-and-burn procedure as given in Algorithm 5 takes as input, a tree $T$, a positive integer $b$ and the list of roots of subtrees $RS$ returned by the get-subtrees procedure. The merge-and-burn procedure returns either a burning sequence of length at most $\lfloor 1.75b \rfloor + 1$ or returns $\phi$. It maintains a set of burning radii $BR$ and burning sequence $BS$. $BR = \{\lfloor 1.75b \rfloor, \lfloor 1.75b \rfloor - 1, \ldots, 1, 0\}$ and $BS$ contains the nodes of the burning sequence.

The merge-and-burn procedure processes the subtrees in the list $RS$ in decreasing order of index. If $l$ is the number of subtrees in the list $RS$, the subtrees will be processed in order from $l$ to 1. Let us assume that we are processing the subtree $RS[i], 1 \leq i \leq l$. The following cases will arise:

Case 1. $RS[i].\text{height} = b$

We check if there is any unprocessed subtree $RS[j]$ such that $d(RS[i].\text{root}, RS[j].\text{root}) \leq b + 1$. If such a subtree exists, we merge the subtrees $RS[i]$ and $RS[j]$. The merge operation produces a new subtree $T_{ij}$ which is the union of subtrees $T_{v_i}, T_{v_j}$ and the shortest path between the subtree roots $v_i$ and $v_j$. The center vertex of the shortest path between the subtree roots $v_i$ and $v_j$ will be the root of new subtree with radius $r' = (d(v_i, v_j) + 2b)/2$. The minimum burn radius $r \in BR$ such that $r \geq r'$ is allocated to the merged subtree $T_{ij}$ and the number $r$ is removed from $BR$. The center vertex of the subtree $T_{ij}$ is placed in the burning sequence at $BS[[1.75b] - r]$. If no such radius $r$ is available in $BR$, we do not merge subtrees $RS[i]$ and $RS[j]$. The subtree $RS[i]$ will be burned independently as described in Case 1.

Case 2. There is no subtree $RS[j]$ such that $d(RS[i].\text{root}, RS[j].\text{root}) \leq b + 1$ or the merge was unsuccessful is Case 1.
The minimum \( r \in BR \) such that \( r \geq RS[i].radius \) is allocated to the subtree \( RS[i] \) and \( r \) is removed from \( BR \). The vertex \( RS[i].center \) is placed in the burning sequence at \( BS[[1.75b] - r] \).

**Case 3.** \( RS[i].height < b \) (\( i = l \))

In this case, no merging will take place. We straightaway assign the burn radius required to burn the subtree. The minimum number \( r \in BR \) such that \( r \geq RS[l].radius \) is allocated to the subtree \( RS[l] \). The center vertex of the subtree \( RS[l] \) is placed in the burning sequence at \( BS[[1.75b] - r] \). Note that the valid indices for \( BS \) are 0 to \([1.75b]\). Every time a number \( r \) is allocated, we remove \( r \) from \( BR \) to make sure that no burn radius is allocated twice. This case will be handled at the end after assigning burn radii to all the subtrees \( RS[l-1..1] \).

In all the above cases, if a minimum suitable radius \( r \) is not available in \( BR \) then the algorithm returns \( \emptyset \) indicating that the burning is unsuccessful. Note that a merged subtree cannot be covered with a single \( b \)-site. We need 2 \( b \)-sites to cover the subtree. And for a subtree of height \( b \) we need one \( b \)-site to cover the subtree.

**Lemma 3 (From [3])** For a positive integer \( r \), if there are \( r \) vertices which are at pairwise distance of at least \( 2r - 1 \) then \( b(G) \geq r \).

From Lemma 3 we have the following corollary. The proof of the corollary is very similar to the proof of Lemma 3 of Bonato et. al [3].

**Corollary 1.** For a positive integer \( r \), if there are \( r + 1 \) vertices which are at pairwise distance of at least \( 2r - 1 \) in \( G \), then \( b(G) \geq r + 1 \).

**Proof.** Let \( S = \{x_1, x_2, \cdots, x_{r+1}\} \) be the vertices which are at a pairwise distance of at least \( 2r - 1 \). If there is a burning sequence of length \( r \), each node in the burning sequence spreads fire to the nodes which are at a distance of at most \( r - 1 \). For each \( x_i \in S \), consider a circle of radius \( r - 1 \) around it. No two circles will intersect as the distance between the centers is at least \( 2r - 1 \). To burn a center vertex, either we should put fire at the center vertex or put fire at a vertex which is in its \( r - 1 \) circle. Any burned vertex in one circle can not spread fire to another circle’s center, because the distance between the vertex and the other center is greater than \( r - 1 \). So we should include at least one vertex from each circle in the burning sequence. Therefore, we can not burn the graph in \( r \) rounds and hence \( b(G) \geq r + 1 \). \( \square \)
Algorithm 1: Approximation Algorithm for Graph Burning on Tree.

**Input:** A tree $T$ and a positive integer $b$

**Output:** Return a Burning sequence or "BAD-GUESS"

1. **BURN-GUESS-TREE** $(T, b)$:
   2. $T' \leftarrow T$
   3. $RS \leftarrow \text{GET-SUBTREES}(T', b)$
   4. $BS \leftarrow \text{MERGE}(g, RS, b)$
   5. if $BS \neq \emptyset$ then
      6. return $BS$
   7. end
   8. else
      9. return BAD-GUESS
   10. end

Algorithm 2: $b$-cutting

**Input:** A tree $T$ and a positive integer $b$.

**Output:** A set of roots of $b$-height subtrees and resulting tree $(T')$ after $b$-cutting $(T^b)$.

1. **$b$-CUTTING** $(T, b)$:
   2. $T' \leftarrow T$
   3. $T'' \leftarrow T$
   4. for $i = 1$ to $b$ do
      5. foreach $v \in T'$ do
         6. if $\deg(v) \leq 1$ then
            7. $T'' \leftarrow T'' \setminus \{v\}$
            8. if $V(T'') = \emptyset$ then
               9. $S \leftarrow S \cup \{v\}$
            10. end
      11. end
   12. for $v \in T'$ do
      13. if $\deg(v) \leq 1$ then
         14. $S \leftarrow S \cup \{v\}$
      15. end
   16. return $S, T'$
Algorithm 3: Cut the tree into $b$-height subtrees.

**Input:** A tree $T$ and positive integer $b$.

**Output:** A list of subtrees of height $b$.

```
1 GET-SUBTREES $(T, b)$:
2     $RS \leftarrow \phi$ ;
3     $T' \leftarrow T$ ;
4     $i \leftarrow 1$ ;
5     while $V(T') \neq \phi$ do
6         $(S, T^b) \leftarrow \text{b-CUTTING}(T', b)$ ;
7             foreach $v \in S$ do
8                 $u \leftarrow \text{neighbours}(T', v)$ ;
9                     $T \leftarrow T \setminus \{(u, v)\}$ ;
10                    $T_v \leftarrow \text{connected component in } T \text{ containing } v$ ;
11                        $T \leftarrow T \setminus \{T_v\}$ ;
12                        $T_v.radius \leftarrow \text{radius}(T_v)$ ;
13                        $T_v.root \leftarrow v$ ;
14                        $T_v.center \leftarrow \text{center}(T_v)$ ;
15                        $T_v.height \leftarrow \text{height}(T_v)$ ;
16                        $RS[i] \leftarrow T_v$ ;
17                        $i \leftarrow i + 1$ ;
18             $T' \leftarrow T$ ;
19     return $RS$
```

Algorithm 4: To allocate radius to a subtree and update $BR, BS, RS$.

**Input:** Subtree root set $RS$, burning sequence $BS$ and burning radii set $BR$.

**Output:** It returns updated $RS, BS, BR$ or $\phi$.

```
1 burn_subtree $(RS, BS, BR, r, c)$:
2     $S \leftarrow \{r' \in BR : r' \geq r\}$ ;
3     if $S \neq \phi$ then
4         $BS[\text{min}(S)] \leftarrow c$ ;
5         $RS \leftarrow RS \setminus \{RS[i]\}$ ;
6         $BR \leftarrow BR \setminus \{\text{min}(S)\}$ ;
7         return $(RS, BS, BR)$
8     end
9 else
10     return $\phi$
11 end
```
Fig. 1. The subtrees rooted at $v_i$ and $v_j$, $d(v_i, v_j) \geq b + 1$. The height of the subtrees is $b$, radius $\geq \lceil 0.75b \rceil$. $a_i, b_i \in T_{v_i}$, $(a_j, b_j \in T_{v_j})$ are end nodes of the diameter path of $T_{v_i}$ ($T_{v_j}$).

Lemma 4 For a positive integer $0 < p \leq \lfloor 0.25b \rfloor$, $b \in \mathbb{Z}^+$ and a root-subset $S \subseteq \{v_1, v_2, v_3 \cdots v_x\}$ of subtrees $\{T_{v_1}, T_{v_2}, T_{v_3} \cdots T_{v_x}\}$ each of height $b$. If

1. $\text{radius}(T_{v_i}) \geq b - p$, $\forall v_i \in S$.
2. $|S| \geq \lceil 0.5b \rceil + p$.
3. $\text{dist}(v_i, v_j) > b$, $\forall v_i, v_j \in S$.

Then $b(T) \geq b + 1$.

Proof. Clearly $\text{radius}(T_{v_i}) \geq \lceil 0.75b \rceil$ (since $b - p \geq \lceil 0.75b \rceil$). Let $a_i$ and $b_i$ be the end points of the diameter path of subtree $T_{v_i}$. And assume that $b_i$ is the node with highest depth in the subtree $T_{v_i}$ as shown in Figure 1. Clearly

$$\text{dist}(a_i, b_i) \geq 2 \times (b - p) - 1, \quad (1)$$
$$\text{dist}(a_i, a_j) > 2b \quad \text{and} \quad \text{dist}(b_i, b_j) > 2b \quad (2)$$

For the sake of contradiction, let us assume that $b(T) \leq b$. During the burning process, if we burn a vertex in $T_{v_i}$, the fire will not spread to the other subtrees as subtree roots are at a distance more than $b$. Let us consider the placement of the first $p$ source vertices of the burning process. The fire will spread to at most $p$ subtrees. The number of remaining subtrees is $|S| - p \geq \lceil 0.5b \rceil$.

Each subtree has two vertices which satisfies equations $1$ and $2$. Therefore, there exists at least $2 \times \lceil 0.5b \rceil \geq b - p + 1$ (since $p > 0$) vertices which are at pairwise distance of at least $2 \times (b - p) - 1$. According to Corollary 1 burning number of the remaining subtrees $\geq b - p + 1$. Then the minimum number of rounds needed to burn the graph is $\geq (b - p + p - 1)$. Therefore, the burning number of the tree $b(T) \geq b + 1$. $\Box$
Algorithm 5: Merge procedure

**Input:** A tree $T$, a positive integer $b$ and list of $b$-height subtree $RS$.
**Output:** Return a burning sequence or "φ"

1. \text{MERGE-AND-BURN} $(T, RS, b)$:
   2. $BR \leftarrow \lfloor 1.75 \times b \rfloor, \lfloor 1.75 \times b \rfloor - 1, \lfloor 1.75 \times b \rfloor - 2 \ldots 3, 2, 1, 0$;
   3. $BS \leftarrow \phi$;
   4. $l \leftarrow RS.length$;
   5. for $i \leftarrow l$ to $1$ do
      6. if $i = l$ and $RS[i].height < b$ then
         7. continue;
      8. else
         9. $S \leftarrow \{ v : \text{dist}(v, RS[i].root) \leq b + 1 \text{ and } v \in RS \setminus\{RS[i].root\} \}$;
         10. $f \leftarrow 0$;
         11. for $v \in S$ do
            12. path $\leftarrow \text{singleSourceShortestPath}(T, v, RS[i].root)$;
            13. $r \leftarrow (\text{path.length} + 2 \times b)/2$;
            14. $t \leftarrow \text{burn}_\text{sub}(RS, BS, BR, r, path[\text{path.length}/2])$;
            15. if $t \neq \phi$ then \textit{Case 1}
               16. $(RS, BS, BR) \leftarrow t$;
               17. $f \leftarrow 1$;
               18. break;
            end
         end
      19. if $f = 0$ then \textit{Case 2}
         20. $t \leftarrow \text{burn}_\text{sub}(RS, BS, BR, RS[i].radius, RS[i].center)$;
         21. if $t = \phi$ then
            22. return $\phi$
         23. else
            24. $(RS, BS, BR) \leftarrow t$;
         end
      end
   25. if $RS[i].height < b$ then \textit{Case 3}
      26. $t \leftarrow \text{burn}_\text{sub}(RS, BS, BR, RS[i].radius, RS[i].center)$;
      27. if $t \neq \phi$ then
         28. $(RS, BS, BR) \leftarrow t$
      29. else
         30. return $\phi$
      end
   end
   31. return $BS$
end
5 Calculating Approximation Factor

Let $BR$ be the set of burn radii in the range $[0, \alpha b]$.  
$BR = \{ [\alpha b, \cdots, [1.5b + 0.5], \cdots, b, b - 1, \cdots, 1, 0] \}$  

Let $F' = \{ e \in F | e \geq [1.5b + 0.5] \}$. That is, all the elements in $F'$ are $\geq [1.5b + 0.5]$. When we merge two sub-trees $T_v$ and $T_u$, the resulting sub-tree is $T_{ij}$. The diameter of $T_{ij} \leq 3b + 1$ and radius of $T_{ij} \leq [1.5b + 0.5]$. We have $|F| = \alpha b - b$ and $|F'| = \alpha b - 1 - [1.5b + 0.5] + 1 = \alpha b - [1.5b + 0.5]$.

If all the numbers in $F'$ are assigned to merged subtrees, the number of $b$-sites equals $2|F'| = 2(\alpha b - [1.5b + 0.5])$. And each of the remaining numbers in $F$ will give one $b$-site. Which is $|F| - |F'| = \alpha b - b - (\alpha b - [1.5b + 0.5]) = [1.5b + 0.5] - b$. For what value of $\alpha$, do we get a total of at least $b$ number of $b$-sites?

\[
2|F'| + |F| - |F'| \geq b \Rightarrow |F'| + |F| \geq b  
\]
\[
\alpha b - [1.5b + 0.5] + \alpha b - b \geq b  
\]
\[
\Rightarrow \alpha b - [1.5b + 0.5] + \alpha b \geq 2b  
\]
\[
2\alpha b - [1.5b + 0.5] \geq 2b \Rightarrow 2\alpha b \geq 2b + [1.5b + 0.5]  
\]
\[
\alpha \geq \frac{[1.5b + 0.5] + 2b}{2b}  
\]
\[
\Rightarrow \alpha \geq \frac{1.75b + 0.25}{b} = \frac{1.75b + 1}{b}  
\]

Therefore the minimum value of $\alpha$ is $\frac{1.75b + 1}{b}$.

Lemma 5 If the Algorithm returns $\text{BAD-GUESS}$ then $b(T) \geq b + 1$.

Proof. In the algorithm we use $RS$ (root set), $BR$ (burn radii).
$BR = \{ [1.75b], \cdots, [1.5b + 0.5], \cdots, b, b - 1, \cdots, 1, 0 \}$ and the list of $b$-height sub-trees returned by the $\text{GET-SUBTREES}$. $RS = \{ T_{v_1}, T_{v_2}, \cdots T_{v_l} \}$. The height of each sub-tree in $RS$ is $b$, except the last sub-tree, whose height $\leq b$.

If the algorithm returns $\text{BAD-GUESS}$, then it means that all radii in $F$ range are used, 0 or more radii from $E$ range are used and there are one or more subtrees which are not burned (not allocated any radius from $BR$). We get the following cases, in each of the cases we show that $b(T) \geq b + 1$.

Case 1. Every radius in the $F'$ range is allocated to merged subtrees: Each sub-tree in $F'$ range brings $2 b$-sites and each sub-tree in $F \setminus F'$ brings one $b$-site. The minimum number of $b$-sites: $|F'| \ast 2 + |F| - |F'| = |F'| + |F|$. 

As we know $|F'| = [1.75b] - [1.5b + 0.5] + 1$ and $|F| = [1.75b] - b + 1$.

\[
|F'| + |F| \\
= [1.75b] - [1.5b + 0.5] + 1 + [1.75b] - b + 1 \\
= 2 * [1.75b] - [1.5b + 0.5] + 2 - b \\
= [3.50b] - [1.5b + 1] + 2 - b \\
= 3b + [0.50b] - b - [0.50b] - 1 + 2 - b = b + 1.
\]

(3)

We get $b$ number of $b$-sites and there are vertices which are not covered by these $b$-sites. It means that the tree does not have $b$-site partitioning. By Lemma 1 of Bonato et al. [3], $b(T) \geq b + 1$.

**Case 2.** For a positive integer $0 < p \leq [0.25b]$, $p$ radii in $F'$ range are not allocated to merged subtrees and no radii in $E$ range is used:

It means that, in $F'$ range, $|F'| - p$ radii are allocated to merged subtrees and $p$ radii are allocated to $b$-height subtrees. Among the subtrees assigned to $F$ range, $|F|-(|F'| - p)$ radii are allocated to $b$-height subtrees. The pairwise distance among the roots of these $|F|-(|F'| - p)$ subtrees is $> b + 1$, otherwise they would have been merged. As no radii in $E$ range is used, radius of each subtree is $b$.

The number of subtrees with pairwise distance $> b + 1$ is $|F| - |F'| + p$.

\[
|F| - |F'| + p \\
= [1.75b] - b + 1 - [1.75b] + [1.5b + 0.5] + p - 1 \\
= [0.5b + 0.5] + p
\]

Let the $S = \{T_{v_1}, T_{v_2}, \cdots \}$ be the subtrees with pairwise distance $> b + 1$

1. $radius(T_{v_i}) = b, \forall v_i \in S$
2. $|S| \geq [0.5b + 0.5] + p.$
3. $dist(v_i, v_j) > b, \forall \{v_i, v_j\} \in S.$

From Lemma 2, the burning number of the tree, $b(T) \geq b + 1$.

**Case 3.** For a positive integer, $0 < p \leq [0.25b]$, $p$ radii in $F'$ range are not allocated to merged subtrees and one or more radii in $E$ range are used:

Let us assume that $E' \subseteq E$ radii are used from the $E$ range. The total number of $b$-sites $\beta = |F| + |F'| - p + |E'| = b + 1 - p + |E'|$ (since $|F| + |F'| = b + 1$). If $\beta \geq b$, by Case 1, the burning number of the tree $b(T) \geq b + 1$. Otherwise, $\beta < b$ \Rightarrow $|E'| < \beta - 1 \Rightarrow |E'| < [0.25b] - 1 \Rightarrow |E'| < [0.25b]$.

Let $S = \{T_{v_1}, T_{v_2}, \cdots \}$ be the subtrees in $F$ range with pairwise distance $> b + 1$. The number of subtrees with pairwise distance $> b + 1$ is $|F| - |F'| + p$.

Let $E'' = \{e \in E|[0.75b] \leq e \leq b - 1\}$, $|E''| \geq [0.25b]$ and $|E'| > |E'|$. It means that at least one radius in $E''$ is not allocated to any subtree as $|E'| < [0.25b]$.

For each $T_{v_i} \in S$, $radius(T_{v_i}) \geq [0.75b]$, otherwise it would have been allocated a vacant radius from $|E''|$ and one radius in $F$ would be free which contradicts the assumption that all radii in $F$ range are utilized.
1. \( \text{radius}(T_{v_i}) \geq \lceil 0.75b \rceil, \forall v_i \in S \)
2. \(|S| \geq \lceil 0.5b + 0.5 \rceil + p + 1 \)
3. \( \text{dist}(v_i, v_j) > b, \forall v_i, v_j \in S \).

From Lemma 4 the burning number of the tree, \( b(T) \geq b + 1 \).

Using the algorithm we compute the minimum \( b \) for which the algorithm returns a burning sequence of length \( \lfloor 1.75b \rfloor + 1 \). As the algorithm returns BAD-GUESS for \( b - 1 \) from Lemma 5 the burning number of tree \( \geq b + 1 \). With this we state the following theorem.

**Theorem 6** There exists a polynomial time \( \lfloor 1.75b(T) \rfloor + 1 \) approximation algorithm for the Graph Burning on trees, where \( b(T) \) is the burning number of the tree.

The time complexity of the algorithm is \( O(|V||E|) \) as both procedures GET-SUBTREES and MERGE-AND-BURN take \( O(|V||E|) \) time.

### 6 Results and Discussion

As part of the experimentation, we have implemented the 3-approximation algorithm [3] for Graph Burning on general graphs, 2-approximation algorithm [3] for Graph Burning on trees and the proposed \( \lfloor 1.75b(T) \rfloor + 1 \) approximation algorithm for trees. For an arbitrary graph \( G \) and a spanning tree of \( G \) \( (SPT(G)) \), \( b(G) \leq b(SPT(G)) \). The burning number of any spanning tree of the graph will give an upper bound on the burning number of the graph.

The experimentation is carried out on 10 real world data sets from Networks Data Repository [15] and from SNAP data set [12] and 20 randomly generated trees. For datasets other than trees, the Burning number of the graphs are estimated by considering a spanning tree of the graph. The algorithms are implemented in Python.

The experimental results are shown in Tables 1 and 2 for real world data sets and trees respectively. Even though the approximation factor is close to 1.75, if we consider best known lower bound the approximation factor obtained is much smaller. For example, for the dataset Cite-DBLP, the approximation algorithm achieves the best known lower bound of 41, which is the actual burning number of the graph. In the case of randomly generated trees, the actual approximation factor is less than 1.5 in most of the cases. For example, for the tree T17, the actual approximation factor is less than 1.38.

Now we give examples to show that our proposed algorithm produces (i) better lower bound, and (ii) better upper bound when compared to the 2-approximation algorithm of Bonato et al. [3]. The examples are shown in the Figures 2 and 3 respectively. Consider the tree in Figure 2. The algorithm returns BAD-GUESS for \( b = 5 \) and hence \( b(T) \geq 6 \), whereas the 2-approximation algorithm of Bonato et al. [3] gives a lower bound of 5 when \( C_5 \) is chosen as a root. So using Algorithm 1 we get an improved lower bound.
Fig. 2. A tree with 5-cutting, $C_1, C_2, C_3, C_4,$ and $C_5$ are roots of subtrees obtained.

Fig. 3. A tree with 5-cutting. $C_1, C_2, C_3, C_4$ and $C_5$ are the centers of the subtrees obtained after 5-cutting. The center $C_{45}$ is the new center at merging the subtrees with center $C_4$ and $C_5$.

Table 1. Comparison of lower bounds $L$ and estimated burning numbers ($B$) of different approximation algorithms for some real world data sets.

| Real world dataset | $|V|$ | $|E|$ | 3-APPRX | 2-APPRX | Our Algo |
|--------------------|------|------|---------|---------|----------|
|                    |      |      |  $L$    | $B \sim L$ | $B \sim L$ | $B$     |
| Reed98             | 962  | 961  | 3 6 4 8 | 3 6     |          |         |
| Mahindas           | 1258 | 1257 | 4 9 4 8 | 3 6     |          |         |
| Cite-DBLP          | 12.6K| 49.7K| 41 120 41 | 82 23 41 |          |         |
| Chameleon          | 2.2K | 2.2k | 4 9 5 10 | 5 9     |          |         |
| TVshow             | 3.8K | 3.8K | 6 15 9 18 | 9 16    |          |         |
| Ego-Facebook       | 4K   | 4K   | 4 9 4 8 | 3 6     |          |         |
| Squirrel           | 5K   | 5K   | 4 9 5 10 | 5 9     |          |         |
| Politician         | 5.9K | 5.9K | 5 12 7 14 | 6 11    |          |         |
| Government         | 7K   | 7K   | 4 9 5 10 | 5 9     |          |         |
| Crocodile          | 11K  | 11K  | 5 12 6 12 | 5 9     |          |         |
**Table 2.** Comparison of lower bounds \( L \) and estimated burning numbers \( B \) of different approximation algorithms for randomly generated trees.

| Random trees | \( |V| \) | 3-APPRX \( L \) | 2-APPRX \( L \) | Our Algo \( L \) | Our Algo \( B \) |
|--------------|----------|---------------|---------------|----------------|---------------|
| T1           | 1000     | 12            | 33            | 15             | 30            |
| T2           | 2000     | 15            | 42            | 19             | 38            |
| T3           | 3000     | 19            | 54            | 22             | 44            |
| T4           | 4000     | 20            | 57            | 25             | 50            |
| T5           | 5000     | 20            | 57            | 26             | 52            |
| T6           | 6000     | 23            | 66            | 31             | 62            |
| T7           | 7000     | 22            | 63            | 30             | 60            |
| T8           | 8000     | 25            | 72            | 33             | 66            |
| T9           | 10000    | 26            | 75            | 36             | 72            |
| T10          | 11000    | 28            | 81            | 37             | 74            |
| T11          | 12000    | 29            | 84            | 38             | 76            |
| T12          | 13000    | 31            | 90            | 38             | 76            |
| T13          | 14000    | 29            | 84            | 39             | 78            |
| T14          | 15000    | 29            | 84            | 41             | 82            |
| T15          | 16000    | 31            | 90            | 42             | 84            |
| T16          | 17000    | 32            | 93            | 43             | 86            |
| T17          | 18000    | 33            | 96            | 45             | 90            |
| T18          | 19000    | 35            | 102           | 45             | 90            |
| T19          | 25500    | 37            | 114           | 50             | 100           |
| T20          | 50000    | 43            | 132           | 60             | 120           |
For the tree given in Figure 3, the 2-approximation algorithm of Bonato et al. [3] produces a burning sequence of length 10 when $C_2$ is chosen as the root, whereas the Algorithm 1 produces a burning sequence of length 9.

7 Conclusions and Future Work

In this paper, we have proposed an approximation algorithm for the Graph Burning on trees. The proposed algorithm produces burning sequence of length at most $\frac{1.75b(T) + 1}{b(T)}$, a significant improvement on the known 2-approximation algorithm. We have done extensive experimentation on real world networks of sizes up to $11K$ nodes, as well as on a few randomly generated trees of sizes up to $19K$. Results show a clear improvement on the estimation of the burning number compared to the other known approximation algorithms. The main idea of the algorithm is based on the merge procedure which optimizes the number of burning radii of the burning sequence. The ideas of extracting equal height subtrees of the tree and the merge technique may be explored for the other related problems in information diffusion. As future work we would like to explore the approximation algorithms for the generalized Graph Burning problem namely $k$-burning problem.

References

1. Stéphane Bessy, Anthony Bonato, Jeannette Janssen, Dieter Rautenbach, and Elham Roshanbin. Bounds on the burning number. Discrete Applied Mathematics, 235:16–22, 2018.

2. Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. Burning a graph as a model of social contagion. In Algorithms and Models for the Web Graph, WAW 2014, volume 8882, pages 13–22. Lecture Notes in Computer Science, Springer, 2014.

3. Anthony Bonato and Shahin Kamali. Approximation algorithms for graph burning. In Theory and Applications of Models of Computation. TAMC 2019, pages 74–92. Lecture Notes in Computer Science, Springer, 2019.

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

5. Zahra Rezai Farokh, Maryam Tahmasbi, Zahra Haj Rajab Ali Tehrani, and Yousof Bual. New heuristics for burning graphs. ArXiv, abs/2003.09314, 2020.

6. J. Garcia, R. Menchaca, J. Sanchez, and R. Menchaca. Local search algorithms for the vertex k-center problem. IEEE Latin America Transactions, 16(6):1765–1771, 2018.

7. Rahul Kumar Gautam, Anjeneya Swami Kare, and Durga Bhavani. Faster heuristics for graph burning. Applied Intelligence, 2021.

8. Lil María X. Rodríguez Henríquez Jesús García Díaz, Julio César Pérez Sansalvador. Burning graphs through farthest-first traversal. 2021.

9. Shahin Kamali, Avery Miller, and Kenny Zhang. Burning two worlds: Algorithms for burning dense and tree-like graphs. In Theory and Practice of Computer Science, SOFSEM 2020, pages 113–124. Lecture Notes in Computer Science, Springer, 2020.
10. Anjeneya Swami Kare and I Vinod Reddy. Parameterized algorithms for graph burning problem. In *International Workshop on Combinatorial Algorithms*, pages 304–314. Springer, 2019.

11. Yasuaki Kobayashi and ota Otachi. Parameterized complexity of graph burning, *ArXiv*, 2021.

12. Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection. [http://snap.stanford.edu/data](http://snap.stanford.edu/data), 2014.

13. Huiqing Liu, Ruiting Zhang, and Xiaolan Hu. Burning number of theta graphs. *Applied Mathematics and Computation*, 361:246–257, 2019.

14. Debajyoti Mondal, N. Parthiban, V. Kavitha, and Indra Rajasingh. Apx-hardness and approximation for the k-burning number problem. *ArXiv*, 2021.

15. Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, page 4292–4293. AAAI Press, 2015.

16. Kai Sim, Ta Sheng Tan, and Kok Wong. On the burning number of generalized petersen graphs. *Bulletin of the Malaysian Mathematical Sciences Society*, 41:1657–1670, 2017.

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

18. Marek Šimon, Ladislav Huraj, Iveta Luptáková, and Jiří Pospíchal. Heuristics for spreading alarm throughout a network. *Applied Sciences*, 9:3269, 2019.