A Row Generation Algorithm for Finding Optimal Burning Sequences of Large Graphs

Felipe de Carvalho Pereira[1]
Institute of Computing, University of Campinas, Campinas, Brazil

Pedro Jussieu de Rezende
Institute of Computing, University of Campinas, Campinas, Brazil

Tallys Yunes
Miami Herbert Business School, University of Miami, Coral Gables, USA

Luiz Fernando Batista Morato
Institute of Computing, University of Campinas, Campinas, Brazil

Abstract

We propose an exact algorithm for the Graph Burning Problem (GBP), an NP-hard optimization problem that models the spread of influence on social networks. Given a graph G with vertex set V, the objective is to find a sequence of k vertices in V, namely, $v_1, v_2, \ldots, v_k$, such that k is minimum and $\bigcup_{i=1}^{k} \{u \in V : d(u, v_i) \leq k - i\} = V$, where $d(u, v)$ denotes the distance between u and v. We formulate the problem as a set covering integer programming model and design a row generation algorithm for the GBP. Our method exploits the fact that a very small number of covering constraints is often sufficient for solving the integer model, allowing the corresponding rows to be generated on demand. To date, the most efficient exact algorithm for the GBP, denoted here by GDCA, is able to obtain optimal solutions for graphs with up to 14,000 vertices within two hours of execution. In comparison, our algorithm finds provably optimal solutions approximately 236 times faster, on average, than GDCA. For larger graphs, memory space becomes a limiting factor for GDCA. Our algorithm, however, solves real-world instances with more than 3 million vertices in less than 19 minutes, increasing the size of graphs for which optimal solutions are known by a factor of 200. Additionally, we conduct tests on the proposed algorithm using a series of challenging instances composed of grid graphs containing up to 5,000 vertices. As a result, we achieve novel optimal solutions and tight optimality gaps that have not been previously reported in the literature.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Combinatorial optimization; Mathematics of computing → Integer programming

Keywords and phrases Graph Burning, Burning Number, Burning Sequence, Set Covering, Integer Programming, Row Generation

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.67

Related Version *Previous Version*: https://arxiv.org/abs/2404.17080 [10]

Supplementary Material *Collection (Source Code and Benchmarks)*: https://doi.org/10.17632/c95hp3m4mz [9]

Funding This work was supported in part by grants from: *Santander Bank*, Brazil; *Brazilian National Council for Scientific and Technological Development* (CNPq), Brazil, #314293/2023-0, #313329/2020-6; *São Paulo Research Foundation* (FAPESP), Brazil, #2023/04318-7; #2023/14427-8.

Acknowledgements We sincerely thank the anonymous referees for their valuable feedback.

[1] Corresponding author.

32nd Annual European Symposium on Algorithms (ESA 2024).
Editors: Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman; Article No. 67; pp. 67:1–67:17
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The Graph Burning Problem (GBP) is a combinatorial optimization problem that models a form of contagion diffusion on social networks in which one seeks to propagate influence over the entire network as quickly as possible [5]. By representing networks as graphs, a contagion can be thought of as a fire that spreads throughout the vertices of a graph following its adjacency relations.

In this problem, we are given an undirected graph $G = (V, E)$ that represents a social network, where each vertex in V corresponds to an individual and each edge $\{u, v\} \in E$ indicates a reciprocal influence relationship between the individuals represented by u and v.

A *burning process* in G unfolds in a series of *rounds.* In every round $i \geq 1$, each vertex is either assigned the *burned* or *unburned* state. Initially, all vertices are unburned. In each round $i \geq 1$, exactly one vertex (a *fire source*) is chosen to be set on fire and becomes burned. Moreover, starting in round $i = 2$, each unburned vertex that has at least one burned neighbor in round $i - 1$ becomes burned and remains in that state until the last round.

A sequence $(v_1, v_2, \ldots, v_k) \in V^k$, where v_i is the i-th fire source, constitutes a *burning sequence* for G, if the entire graph is burned by the k-th round.

For each $u \in V$, we denote by $N_j[u] = \{v \in V : d(u, v) \leq j\}$ the *j-th closed neighborhood* of u, where $d(u, v)$ denotes the *distance* between u and v, *i.e.*, the number of edges in a shortest path in G that connects these vertices.

Formally, $(v_1, v_2, \ldots, v_k)$ is a burning sequence for G if

$$\bigcup\nolimits_{i=1}^{k} N_{k-i}[v_i] = V \tag{1}$$

It follows from the burning process that v_i ascertains that each vertex in $N_{k-i}[v_i]$ (including v_i) gets burned by round k.

▶ **Problem 1** (Graph Burning Problem). *Given an undirected graph G, find a burning sequence for G of minimum length.*

The length of a shortest burning sequence for G is called its *burning number*, denoted $b(G)$, and was introduced in the literature as a graph parameter that measures the speed at which a propagation can spread throughout a network: the smaller the burning number of G is, the more susceptible G is to fast contagions. Although this parameter is of special interest for social networks, the problem has also been investigated for various other classes of graphs [2, 3, 4, 7, 8, 16, 18, 19, 20, 26, 27].

The GBP is NP-hard [2, 5] and, so far, four exact approaches have been proposed to solve the problem for arbitrary graphs [12, 14]. The currently best known exact algorithm [12], referred to, here, as GDCA, is able to find provably optimal solutions for real-world networks with up to 14,000 vertices in less than two hours. For larger graphs, memory space becomes a limiting factor.

1.1 Our Contributions

In this paper, we propose an exact algorithm for the GBP based on an integer programming (IP) formulation together with a row generation procedure. Through a series of computational experiments, we are able to demonstrate that the proposed method significantly outperforms GDCA. More specifically, our algorithm:

- Finds provably optimal solutions 236 times faster, on average, than GDCA for networks with up to 14,000 vertices;

- Solves real-world networks with over 3 million vertices in less than 19 minutes, hence increasing the size of the vertex set of graphs for which optimal solutions are known by a factor of 200.

This paper is organized as follows. In Section 2, we review the literature on the `GBP` and, in Section 3, we describe our exact algorithm. Section 4 contains a full report on computational experiments and an analysis of the results. Lastly, in Section 5, we present concluding remarks and address future work.

2 Previous Work

The `GBP` was proposed in [5] and has been extensively studied both from theoretical and practical points of view. In this section, we present a brief background review of the problem for arbitrary graphs, focusing on upper bounds, heuristics, approximation algorithms, and mathematical models. For a probe regarding the `GBP` on specific families of graphs, we refer the reader to the survey [3].

Let $G = (V, E)$ be an arbitrary undirected graph with $p \geq 1$ connected components and let $n_1, n_2, \ldots, n_p$ be the numbers of vertices of the these components. Denote by $b(G)$ the burning number of G.

Theorem 2 provides the best known upper bound for $b(G)$ [1, 14].

▶ **Theorem 2.** $b(G) \leq p + \sum_{i=1}^{p} \left\lceil (4n_i/3)^{1/2} \right\rceil$.

Conjecture 3 suggests a tighter upper bound for $b(G)$, but that result remains open since the problem was introduced [5, 14].

▶ **Conjecture 3.** $b(G) \leq \sum_{i=1}^{p} \left\lceil n_i^{1/2} \right\rceil$.

Moreover, Theorem 4 states that Conjecture 3 holds assymptotically for connected graphs with n vertices [23].

▶ **Theorem 4.** $b(G) \leq (1 + o(1))n^{1/2}$.

Among the plethora of heuristics proposed for the `GBP` [11, 12, 15, 22, 28, 29], the most recent are a centrality-based genetic algorithm [22] and the greedy algorithm from [12]. On the other hand, there are two 3-approximation algorithms for the `GBP` [2, 7], and a $(3 - 2/b(G))$-approximation [13] referred to, here, as `BFF`. Algorithm 1 describes `BFF`.

Algorithm 1 `BFF` (as used in [12])

```
Input   : Graph $G = (V, E)$; distances between all pairs of vertices of $G$
Output : A burning sequence $S$
Select $v_1$ arbitrarily
$S \leftarrow (v_1)$
$i \leftarrow 2$
while $S$ is not feasible do
    $v_i = \arg\max_{u \in V} (\min\{d(u, v_1), d(u, v_2), \ldots, d(u, v_{i-1})\})$
    $S \leftarrow (v_1, v_2, \ldots, v_i)$
    $i \leftarrow i + 1$
return $S$
```

BFF progressively builds a burning sequence S by iteratively selecting the i-th fire source as the vertex that is farthest from any of the previous selected fire sources. In [13], it is shown that $|S| \leq 3 \cdot b(G) - 2$ and that BFF has worst-case time complexity $\mathcal{O}(|V|^2)$, provided that the distances between all pairs of vertices are computed a priori.

2.1 Existing Mathematical Models

Regarding exact formulations for the GBP, three IP models, namely, ILP, CSP1 and CSP2, were proposed in [14]. While ILP consists of an optimization model, the last two are decision models that, for a given integer B, determine whether a burning sequence of length B exists.

In both ILP and CSP1, the main variables are indexed by each pair $(v, i) \in V \times \{1, 2, \ldots, U\}$, where U is a known upper bound for $b(G)$, and each of them indicates whether v is burned in round i. This idea has also been applied for the design of mathematical models for related problems, such as the well studied Target Set Selection Problem and some of its variants [25].

In CSP2, the main variables are indexed by each pair $(u, v) \in V \times V$ and each of them specifies whether u is responsible for v getting burned, if u is a fire source. The assemblage of CSP2 requires that the distances between every pair of vertices be known.

2.2 The Current Best Known Exact Algorithm

We now describe GDCA, an exact algorithm that leads to better performance results when compared to simply solving the models cataloged in the previous section, as was empirically demonstrated by experiments reported in [12].

GDCA relies on the fact that the GBP can be seen as a set covering problem. This was first observed in [6] and later formalized in [12] by means of a reduction of the GBP to the Clustered Maximum Coverage Problem (CMCP) [12] that we now describe.

▶ **Problem 5** (Clustered Maximum Coverage Problem). *Given a set P and $k \geq 1$ sets (clusters) $C_1, C_2, \ldots, C_k$, each one containing subsets of P, find k sets $S_1, S_2, \ldots, S_k$ such that $S_i \in C_i$, for $i = 1, 2, \ldots, k$, and $\left|\bigcup_{i=1}^{k} S_i\right|$ is maximum.*

Given an undirected graph $G = (V, E)$, let $P = V$, $k = B$ and, for each $i = 1, 2, \ldots, B$, take $C_i = \{S_{i,v} : v \in V\}$, where $S_{i,v} = N_{B-i}[v]$. The value of an optimal solution for CMCP corresponds to the maximum number of vertices that can be burned in G using a sequence of B vertices [12]. If such number equals $|V|$, then an optimal solution for CMCP, say, $S_{1,v_1}, S_{2,v_2}, \ldots, S_{k,v_k}$, corresponds to a burning sequence $(v_1, v_2, \ldots, v_k)$ for G. Otherwise, one can conclude that $b(G) > B$.

In [12], the following IP model, originally designed for the CMCP, and referred to, here, as CMCP-IP, is used to decide whether a burning sequence of length B for G exists. Let $X = \{x_{v,i} : v \in V, i \in \{1, 2, \ldots, B\}\}$ and $Y = \{y_v : v \in V\}$ be sets of binary variables s.t. $x_{v,i} = 1$ iff v is the i-th fire source and $y_v = 1$ iff v gets burned during the burning process.

$$\texttt{CMCP-IP}\left\{\begin{array}{lll} & \max \sum_{v \in V} y_v & \quad (2) \\ \text{s.t.} & \sum_{v \in V} x_{v,i} = 1 \quad \forall i \in \{1, 2, \ldots, B\} & \quad (3) \\ & \sum_{i=1}^{B} \sum_{u \in V : v \in N_{B-i}[u]} x_{u,i} \geq y_v \quad \forall v \in V & \quad (4) \end{array}\right.$$

The objective function (2) maximizes the number of burned vertices. Constraints (3) establish that exactly one vertex is assigned to each position in the burning sequence. Lastly, Constraints (4) ensure that if v is in the burned state, then there is at least one fire source u (possibly v itself) such that $d(u, v) \leq B - i$, where i is the position of u in the burning sequence. CMCP-IP has a total of $\mathcal{O}(|V| \cdot B)$ binary variables and $\mathcal{O}(|V|)$ constraints. Loading this model requires that the distances between every pair of vertices be known.

GDCA performs a binary search in a certain interval of candidate values for $b(G)$ and uses CMCP-IP to solve each of the decision problems encountered during the search. Algorithm 2 describes the procedure.

Algorithm 2 GDCA

```
Input  : Graph $G = (V, E)$
Output : Optimal burning sequence $S$
$D \leftarrow$ distance matrix of $G$
$S \leftarrow$ BFF($G, D$)
$U \leftarrow |S|$
$L \leftarrow \lceil (|S| + 2)/3 \rceil$
while $L \leq U$ do
    $B \leftarrow \lfloor (L + U)/2 \rfloor$
    $(obj, S') \leftarrow$ SolveCMCP-IP($G$, $D$, $B$)
    if $obj = |V|$ then
        $S \leftarrow S'$
        $U \leftarrow B - 1$
    else
        $L \leftarrow B + 1$
return $S$
```

First, the algorithm computes the distances between all pairs of vertices (*e.g.*, by $|V|$ breadth-first searches, totaling $\mathcal{O}(|V|^2 + |V| \cdot |E|)$ time). Then, BFF is applied to obtain a burning sequence S for G. Next, GDCA computes upper and lower bounds for $b(G)$, namely, $U = |S|$ and $L = \lceil (|S| + 2)/3 \rceil$. The lower bound follows from the fact that, since BFF is a $(3 - 2/b(G))$-approximation, $|S| \leq (3 - 2/b(G)) \cdot b(G)$ and, therefore $\lceil (|S| + 2)/3 \rceil \leq b(G)$. Then, a binary search is employed to solve $\mathcal{O}(\log(U - L + 1))$ GBP decision problems.

In [12], it is shown that GDCA is able to find optimal solutions for networks with up to 12,000 vertices. In the next section, we show how to improve GDCA to design a more effective and efficient exact algorithm for the GBP.

3 A Row Generation Algorithm

In this section, we introduce an exact algorithm for the GBP, denoted by PRYM, preceded by the presentation of some useful results and an IP formulation for which a row generation method is employed in PRYM.

Let $G = (V, E)$ be an undirected graph. Since any sequence containing all vertices of V constitutes a trivial feasible solution for G, we have that $b(G) \leq |V|$. Moreover, there exists a burning sequence of length k for G for each integer k, with $b(G) \leq k \leq |V|$ since we may append (dummy) fire sources to any given burning sequence.

Furthermore, although the definition of GBP does not forbid burning sequences that contain reoccurring vertices, it is easy to prove that for each $k \in \mathbb{Z}$, where $b(G) \leq k \leq |V|$, there exists a burning sequence of length k for G that does not contain repeated vertices.

We now propose an integer program for the decision version of the `GBP`, denoted by `GBP-IP`. This model determines whether there exists a burning sequence of length $B \leq |V|$ for G. Let $X = \{x_{v,i} : v \in V, i \in \{1, 2, \ldots, B\}\}$ be a set of binary variables such that $x_{v,i} = 1$ iff v is the i-th fire source in a burning sequence for G.

$$\texttt{GBP-IP}\begin{cases} & \text{Find } X & & (5)\\ \text{s.t.} & \sum_{i=1}^{B} x_{v,i} \leq 1 & \forall v \in V & (6)\\ & \sum_{v \in V} x_{v,i} = 1 & \forall i \in \{1,2,\ldots,B\} & (7)\\ \sum_{i=1}^{B} & \sum_{u \in V : v \in N_{B-i}[u]} x_{u,i} \geq 1 & \forall v \in V & (8) \end{cases}$$

Constraints (6) ensure that each $v \in V$ appears at most once in the burning sequence. Constraints (7) establish that exactly one vertex is assigned to each position in the burning sequence. Lastly, Constraints (8) ensure that each vertex is burned by the end of round B, *i.e.*, that Equation (1) is satisfied. From now on, we also refer to (8) as *covering constraints*. Whenever (8) is satisfied for a vertex v, we say that v is *covered*, otherwise, v is *uncovered*.

We remark that although Constraints (6) are not necessary for the correctness of the model, they cut off integer solutions with repeated vertices, which reduces the search space. Indeed, Constraints (6) accelerates the resolution of `GBP-IP` for some graphs by up to 18% in practice (see Section 4).

`GBP-IP` has a total of $\mathcal{O}(B \cdot |V|)$ binary variables and $\mathcal{O}(|V|)$ constraints. Observe that `GBP-IP` can be obtained from `CMCP-IP` by: removing the objective function (2), adding Constraints (6), and setting $y_v = 1$ for each $v \in V$.

▶ **Proposition 6.** *If $X = \{x_{v,i} : v \in V, i \in \{1, 2, \ldots, B\}\}$ is a feasible solution for `GBP-IP`, then there exists a burning sequence S of length B for G such that for each $v \in V$, if $x_{v,i} = 1$, then v is the i-th fire source of S.*

Proof. Let X be a feasible solution for `GBP-IP`. By (7), for each $i \in \{1, 2, \ldots, B\}$, there exists exactly one vertex v such that $x_{v,i} = 1$. Take a sequence $S = (v_1, v_2, \ldots, v_B)$ such that $x_{v_i,i} = 1$. Since X satisfies (8), S satisfies Equation (1) and, therefore, S is a burning sequence for G. ◀

▶ **Proposition 7.** *If there is a burning sequence of length B for G, then `GBP-IP` is feasible.*

Proof. Let $S = (v_1, v_2, \ldots, v_B)$ be a burning sequence for G with no repeated vertices. Take $X = \{x_{v,i} : v \in V, i \in \{1, 2, \ldots, B\}\}$ such that $x_{v,i} = 1$ iff v is the i-th fire source of S. By construction, X satisfies Constraints (6) and (7). Also, since S satisfies Equation (1), X satisfies Constraints (8). Therefore, X is a feasible solution for `GBP-IP`. ◀

Now, let L and U be lower and upper bounds for $b(G)$. Since there is no burning sequence for G of length less than $b(G)$, it follows from Proposition 6 that for every $B \in [L, b(G) - 1]$, `GBP-IP` is infeasible. Similarly, since there is a burning sequence for G of length $b(G) + q$ for every $q \in \mathbb{Z}_{\geq 0}$, it follows from Proposition 7 that for every $B \in [b(G), U]$, `GBP-IP` admits a feasible solution from which a burning sequence of length B can be built. Therefore, one can perform a binary search to determine the smallest value B in the interval $[L, U]$ for which a feasible solution of GBP-IP exists, leading to an optimal solution of length B for G. This idea is similar to the one employed in GDCA (see Algorithm 2) and is the core of PRYM, which is described in Algorithm 3.

Algorithm 3 PRYM

```
Input   : Graph $G = (V, E)$
Output  : Optimal burning sequence $S$
$S \leftarrow$ BFF-d($G$)
$U \leftarrow |S|$
$L \leftarrow \lceil (|S| + 2)/3 \rceil$
while $L < U$ do
    $B \leftarrow \lfloor (L + U)/2 \rfloor$
    $(answer, S') \leftarrow$ SolveGBP-IP($G$, $B$)
    if $answer = feasible$ then
        $S \leftarrow S'$
        $U \leftarrow B$
    else
        $L \leftarrow B + 1$
return $S$
```

First, PRYM obtains a feasible solution S by running a modified version of BFF (BFF-d) which, instead of being provided with all pairwise distances between vertices, as in GDCA, computes only the required distances on demand. In other words, during the i-th iteration of BFF-d, right before vertex v_i is selected as the i-th fire source of S (see step 5 of Algorithm 1), BFF-d computes the distance between v_{i-1} and each vertex in V by means of a single breadth-first search. These distances are stored in memory until PRYM halts. The motivation for this change is that in that iteration only the distances from each vertex in V to the $i - 1$ previous fire sources are needed. Since the lengths of burning sequences are often much smaller than $|V|$, this modification speeds up BFF-d to a total complexity of $\mathcal{O}(|S|(|V| + |E|))$, leading to a significant improvement in practice.

Next, PRYM computes lower and upper bounds for searching for $b(G)$, namely, $L = \lceil (|S| + 2)/3 \rceil$ and $U = |S|$. We notice that the other approximation algorithms for GBP from [2, 7] could be applied in these first steps of PRYM, despite their weaker approximation ratios in comparison to BFF-d. Although, at times, the actual performance of approximation algorithms can be superior (on average) than their worst-case theoretical guarantee, we chose to employ BFF-d for PRYM since its standard version, BFF, is used in GDCA, whose performance is compared to that of PRYM in Section 4.

Lastly, PRYM performs a binary search on the interval of values between L and U, solving $\mathcal{O}(\log(U - L))$ GBP decision problems by means of solving the GBP-IP model on each query.

We remark that to solve the GBP-IP model, PRYM does not need to load the whole set of covering constraints from the start. Instead, these constraints are loaded on demand following the traditional *lazy constraint* strategy: whenever the IP solver finds an integer solution, we separate a violated covering constraint, if it exists, and add it to the model as a lazy constraint. This approach comes from the observation that, very often, a small subset of the covering constraints may be sufficient to prove that GBP-IP is either infeasible or feasible.

As an illustration, consider these real-world networks obtained from [24]: ia-enron-only

($|V| = 143$, $|E| = 623$, $b(G) = 4$); `DD244` ($|V| = 291$, $|E| = 822$, $b(G) = 7$); and `ca-netscience` ($|V| = 379$, $|E| = 914$, $b(G) = 6$) and depicted in Figure 1. For each of them, the infeasibility of the `GBP-IP` model for $B = b(G) - 1$ can be established by loading only the covering constraints associated with the colored vertices. Moreover, in Section 4, we show that, for instances with up to millions of vertices, between just dozens and a few hundred covering constraints are sufficient to prove that there is no burning sequence of length $B = b(G) - 1$.

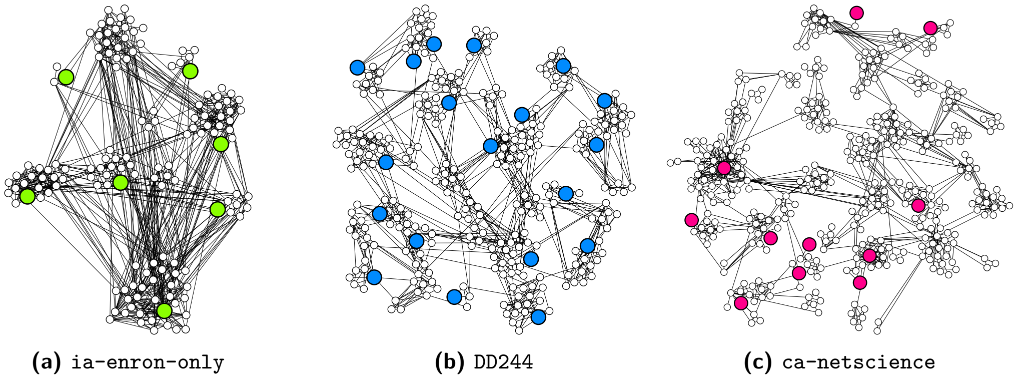
(a) `ia-enron-only` **(b)** `DD244` **(c)** `ca-netscience`

Figure 1 Illustration of the vertices (colored and enlarged) whose covering constraints suffice to prove the infeasibility of `GBP-IP` for $B = b(G) - 1$.

In light of that, whenever `PRYM` invokes an `IP` solver to solve `GBP-IP`, only the covering constraints associated to the vertices in the burning sequence obtained by `BFF-d` are initially loaded into the model. The rest of the covering constraints used by the solver are added according to the following separation procedure.

Given an integer solution found by the solver, we first extract the sequence $S = (v_1, v_2, \ldots, v_B)$ such that $x_{v_i,i} = 1$ in $\mathcal{O}(|V| \cdot B)$ time. Then, for each $i \in \{1, 2, \ldots, B\}$, we determine the distances between v_i and all vertices of G by breadth-first search, in $\mathcal{O}(|V| + |E|)$ time, whenever they had not been previously computed by `PRYM`. Next, for each $u \in V$, we calculate the distance between u and its closest vertex among those burned by round B in the burning process. This value, denoted by $d_S(u)$, can be calculated in $\mathcal{O}(B)$ time as $d_S(u) = \max(0, \min\{d(u, v_1) - (B - 1), d(u, v_2) - (B - 2), \ldots, d(u, v_B)\})$.

If $d_S(u) = 0$, then u is covered, otherwise, $d_S(u) \geq 1$ and u is uncovered. If $d_S(v) = 0$ for every $v \in V$, then no constraint is violated and S is a burning sequence for G. Otherwise, we select $w = \arg\max_{u \in V} d_S(u)$, *i.e.*, the uncovered vertex that is farthest from any burned vertex. Then, we calculate the distances between w and all vertices of G by a breadth-first search, in $\mathcal{O}(|V| + |E|)$ time, if they were not calculated previously. Lastly, we compute the covering constraint for w and load it onto the solver as a lazy constraint.

As a branching rule for solving the `GBP-IP`, `PRYM` determines that for each vertex v, the variable $x_{v,i}$ has a higher priority for branching than $x_{v,j}$ for every $j > i$. The purpose of this approach is to decide the first positions of the burning sequence earlier in the search.

We now highlight the advantages of `PRYM` over `GDCA`. First, observe that `PRYM` addresses the `GBP` decision problem directly via the `GBP-IP` model, while `GDCA` attempts to solve, via the `CMCP-IP` model, an instance of an optimization problem all the way through, even when $B < b(G)$ and an upper bound less than $|V|$ for the objective function (2) is determined. This shows that `PRYM` can be particularly more expedient than `GDCA` in these cases by saving valuable computing time.

Another important time-saving strategy is that PRYM computes, on demand, only distances between pairs of vertices of G known to be necessary instead of GDCA's $|V|^2$ such computations. As we show in Section 4, memory space becomes a limiting factor for GDCA for graphs with upwards of 14,000 vertices, while PRYM is able to handle instances with millions of vertices before memory space starts to become a limitation. Also, in practice, for most of the instances used in Section 4 with at least 10,000 vertices, the asymptotic polynomial complexity of computing all distances far exceeded the actual time PRYM spent in attaining optimal solutions.

Lastly, the major advantage stems from the row generation approach whose separation algorithm is able to discover a small number of decisive covering constraints that are often sufficient for the solver to find a feasible solution or to prove the infeasibility of GBP-IP for a given value of the parameter B. Moreover, since covering constraints tend to involve a substantial number of variables for large graphs, loading a small subset of these constraints ultimately speeds up the resolution of the linear relaxation.

In the next section, we report a series of experiments comparing the efficiency and efficacy of PRYM and GDCA on a large benchmark of instances.

4 Computational Experiments

We now describe the experiments we carried out to empirically evaluate PRYM. For this purpose, we used a machine equipped with an Intel® Xeon® E5-2630 v4 processor, 64 GB of RAM, and the Ubuntu 22.04.1 LTS operating system. For IP solver, we used Gurobi v10.0.3 running on a single thread of execution. The benchmark of instances employed here extends the one used in the experiments reported in [12] to a total of 78 real-world networks obtained from the Network Repository [24] and the Stanford Large Network Dataset Collection [17]. We refer the reader to a publicly available repository [9] that accompanies this paper and includes the source code, problem instances, and the solutions obtained.

The instances were divided into three sets according to the number of vertices: $\Delta_{10\text{K}}$ is the set of instances with at most 10K vertices, $\Delta_{100\text{K}}$ comprises the instances with more than 10K vertices and at most 100K vertices, and $\Delta_{4\text{M}}$ consists of the instances with more than 100K vertices and less than 4M vertices.

The results for $\Delta_{10\text{K}}$, $\Delta_{100\text{K}}$ and $\Delta_{4\text{M}}$ are presented in Tables 1, 2 and 3, respectively. The first three columns describe the instances while the two subsequent ones display the lower and upper bounds for $b(G)$, L and U, obtained via BFF-d. The "Opt" column shows the optimal value, $b(G)$, while the following two columns display the running times (rounded up to the nearest second) spent by GDCA and PRYM to find provably optimal solutions. The last two columns show the number of covering constraints (of type (8)) loaded by PRYM while solving GBP-IP for $B = b(G) - 1$ and for $B = b(G)$.

Recall that when $B = b(G) - 1$, GBP-IP is infeasible and B is a lower bound for $b(G)$. On the other hand, when $B = b(G)$, GBP-IP admits a feasible solution and B is an upper bound for $b(G)$. We highlight that according to Tables 1, 2 and 3, the number of covering constraints used by PRYM for these cases falls significantly short of the number of vertices in each graph. In fact, when $B = b(G) - 1$ and $B = b(G)$, PRYM had to load, on average, only 4.69% and 6.43% of the whole set of possible covering constraints, respectively.

Also, the burning numbers that appear in bold in Tables 2 and 3 are newly obtained results. In the running time columns, the entries containing '–' indicate that the execution was halted due to memory overflow. Also, in the last column, whenever the initial upper bound U was equal to the burning number, '–' is used to reflect that there was no need for PRYM to solve GBP-IP for $B = U = b(G)$.

Table 1 Quantifying the empirical results obtained for $\Delta_{10\text{K}}$.

Instance			Bounds		Opt	Time (sec)		#Cov. Constr. for:	
Name	$\lvert V\rvert$	$\lvert E\rvert$	L	U	$b(G)$	`GDCA`	`PRYM`	$b(G)-1$	$b(G)$
karate	34	78	2	4	3	1	1	4	7
chesapeake	39	170	2	3	3	1	1	7	–
dolphins	62	159	3	6	4	1	1	7	12
rt-retweet	96	117	3	5	5	1	1	9	–
polbooks	105	441	3	5	4	1	1	10	9
adjnoun	112	425	2	4	4	1	1	6	–
ia-infect-hyper	113	2196	2	3	3	1	1	9	–
C125-9	125	6963	2	3	3	1	1	42	–
ia-enron-only	143	623	3	5	4	1	1	7	12
c-fat200-1	200	1534	3	7	7	1	1	37	–
c-fat200-2	200	3235	3	5	5	1	1	16	–
c-fat200-5	200	8473	2	3	3	1	1	5	–
sphere	258	1026	4	9	7	2	1	44	33
DD244	291	822	4	10	7	1	1	19	41
ca-netscience	379	914	4	8	6	1	1	11	25
infect-dublin	410	2765	3	6	5	1	1	12	8
c-fat500-1	500	4459	5	11	9	1	1	33	72
c-fat500-2	500	9139	4	8	7	1	1	36	13
c-fat500-5	500	23191	3	5	5	1	1	32	–
bio-diseasome	516	1188	4	10	7	1	1	13	15
web-polblogs	643	2280	3	7	5	2	1	7	15
DD687	725	2600	4	9	7	21	2	17	85
rt-twitter-copen	761	1029	4	8	7	2	1	9	13
DD68	775	2093	5	12	9	7	2	28	85
ia-crime-moreno	829	1475	3	7	7	21	1	58	–
DD199	841	1902	7	18	12	9	6	96	87
soc-wiki-Vote	889	2914	3	6	6	4	1	15	–
DD349	897	2087	6	16	12	7	13	113	110
DD497	903	2453	6	15	10	14	14	30	146
socfb-Reed98	962	18812	2	4	4	4	1	7	–
lattice3D	1000	2700	5	12	10	1767	1091	295	103
bal-bin-tree-9	1023	1022	5	11	10	1	7	513	16
delaunay-n10	1024	3056	5	11	9	42	9	58	140
stufe	1036	1868	6	15	12	424	147	204	163
lattice2D	1089	2112	7	19	13	1424	95	107	264
bal-ter-tree-6	1093	1092	4	8	7	1	1	44	21
email-univ	1133	5451	3	6	5	7	1	9	18
econ-mahindas	1258	7513	3	6	5	32	1	10	15
ia-fb-messages	1266	6451	3	5	5	9	1	9	–
bio-yeast	1458	1948	5	11	9	23	1	14	27
tech-routers-rf	2113	6632	3	7	6	30	1	9	21
chameleon	2277	36101	3	6	6	36	1	12	–
tvshow	3892	17262	4	10	9	302	2	14	18
facebook	4039	88234	3	5	4	26	1	5	5
DD6	4152	10320	9	24	16	716	189	135	221
squirrel	5201	198493	3	6	6	405	1	9	–
politician	5908	41729	4	8	7	452	2	11	14
government	7057	89455	3	6	6	749	1	10	–

Table 2 Quantifying the empirical results obtained for $\Delta_{100\mathrm{K}}$.

Instance			Bounds		Opt	Time (sec)		#Cov. Constr. for:	
Name	$\lvert V\rvert$	$\lvert E\rvert$	L	U	$b(G)$	GDCA	PRYM	$b(G)-1$	$b(G)$
crocodile	11631	170918	3	6	6	1982	1	10	–
athletes	13866	86858	3	7	**7**	5263	2	19	–
company	14113	52310	4	9	**9**	–	6	21	–
musae-facebook	22470	171002	4	9	**8**	–	7	13	22
new-sites	27917	206259	4	8	**8**	–	4	15	–
deezer-europe	28281	92752	5	12	**10**	–	24	16	18
RO-gemsec-deezer	41773	125826	4	10	**10**	–	10	16	–
HU-gemsec-deezer	47538	222887	4	9	**8**	–	30	14	39
artist	50515	819306	3	7	**6**	–	10	8	11
HR-gemsec-deezer	54573	498202	3	7	**7**	–	8	12	–
soc-brightkite	56739	212945	4	9	**9**	–	17	15	–
socfb-OR	63392	816886	4	8	**8**	–	11	13	–
soc-slashdot	70068	358647	3	7	**7**	–	8	9	–
soc-BlogCatalog	88784	2093195	3	5	**5**	–	4	9	–

Table 3 Quantifying the empirical results obtained for $\Delta_{4\mathrm{M}}$.

Instance			Bounds		Opt	Time (sec)		#Cov. Constr. for:	
Name	$\lvert V\rvert$	$\lvert E\rvert$	L	U	$b(G)$	GDCA	PRYM	$b(G)-1$	$b(G)$
soc-buzznet	101163	2763066	2	4	**4**	–	21	40	–
soc-LiveMocha	104103	2193083	3	5	**5**	–	23	29	–
soc-douban	154908	327162	3	6	**6**	–	34	30	–
soc-gowalla	196591	950327	4	8	**8**	–	31	10	–
soc-twitter-follows	404719	713319	3	6	**6**	–	93	30	–
soc-youtube	495957	1936748	4	10	**10**	–	104	12	–
soc-flickr	513969	3190452	4	10	**10**	–	269	18	–
soc-delicious	536108	1365961	4	9	**8**	–	90	14	17
soc-FourSquare	639014	3214986	2	3	**3**	–	47	6	–
soc-digg	770799	5907132	4	9	**9**	–	177	11	–
soc-youtube-snap	1134890	2987624	5	12	**11**	–	652	15	17
soc-lastfm	1191805	4519330	3	6	**6**	–	165	12	–
soc-pokec	1632803	22301964	4	8	**8**	–	533	16	–
soc-flixster	2523386	7918801	4	8	**5**	–	218	13	8
socfb-B-anon	2937612	20959854	4	8	**8**	–	1111	25	–
socfb-A-anon	3097165	23667394	4	8	**7**	–	774	10	15

Considering the 50 instances, 48 from $\Delta_{10\text{K}}$ and 2 from $\Delta_{100\text{K}}$, for which both PRYM and GDCA obtained provably optimal solutions, for 27 instances PRYM was on average 236.7 times faster than GDCA; they attained the same running times for 21 instances, considering the granularity of the time measurements presented (sec); and PRYM only performed slower than GDCA by a few seconds on two instances: DD349 and bal-bin-tree-9. Among these 50 instances, the most remarkable result occurred for the athletes instance, which was solved by PRYM in 2 seconds, in contrast to the 5263 seconds spent by GDCA.

The remaining 28 instances, 12 from $\Delta_{100\text{K}}$ and 16 from $\Delta_{4\text{M}}$, were only solved by PRYM, which spent a maximum of 18.6 minutes per instance. Remarkably, despite the large sizes of these graphs, which have up to 3.09 million vertices, PRYM required just a few dozen covering constraints to solve GBP-IP. Nevertheless, for graphs with more than 4 million of vertices, space becomes the limiting factor for PRYM due to the size of GBP-IP even without Constraints (8).

We now report some additional results for PRYM. In Section 3, we stated that although Constraints (6) are not necessary for GBP-IP's correctness, they reduce the search space by cutting integer solutions that correspond to burning sequences with repeated vertices. In our tests, lattice3D and stufe were the instances that benefited the most from the inclusion of those constraints, resulting in a reduction of 18% in the running times for these graphs.

Moreover, to confirm the efficacy of the greedy criteria employed in PRYM for selecting the covering constraints that are initially loaded into the model as well as the ones that are added by the separation procedure (see Section 3), we tested a variation of PRYM in which the covering constraints were selected at random. In our tests, the original PRYM performed three times faster than its randomized version, on average, for the majority of instances in our benchmark.

We also tested loading extra covering constraints a priori, in addition to the constraints that cover the vertices from the burning sequence returned by BFF-d. Since the average number of covering constraints used by PRYM (see Tables 1, 2 and 3) was $\sim$50, the alternative algorithm was set to initially load exactly $\min(50, |V|)$ covering constraints for which the respective covered vertices were selected under the same greedy criteria applied by BFF-d. For 15 instances, the running times were reduced by 27.24%, on average, but for 25 other instances, the running times increased by $53,38\%$, on average. Thus, we were unable to reach a consistent outcome and decided to discard this alternative.

4.1 Results for Grid Graphs

Lastly, we evaluated the performance of PRYM on graphs that are square grids since these instances are known to be quite challenging to solve in practice [12].

For these experiments, we set a time limit of 1800 seconds for the IP solver to decide each query in which GBP-IP is solved during the binary search. Besides, we set the *Threads* parameter of Gurobi to its default configuration. Whenever the time limit was exceed for a given $B = k$, we interrupted the binary search and invoked the IP solver for the $B = k - 1$ and $B = k + 1$ cases, provided that they had not been solved earlier. Due to this approach, we were able to determine, for several instances, an optimality gap of just 1 unit on the length of an optimal burning sequence.

Results for $n \times n$ grids, with $3 \leq n \leq 70$, are presented in Tables 4 and 5. The first three columns describe the instances and the next two columns report the initial lower and upper bounds for $b(G)$, L and U, obtained via BFF-d. The two subsequent columns display the best bounds found by PRYM. The next two columns report the optimal value, when it could be determined (*i.e.*, "Lower"="Upper"), and the total time spent by PRYM. The bounds and burning numbers that appear in bold are new results not previously reported in the literature. The last two columns show the number of covering constraints (of type (8)) loaded by PRYM while solving GBP-IP for $B = b(G) - 1$ and for $B = b(G)$. For these columns, '$\star$' is used whenever $b(G)$ could not be determined. Also, in the last column, if the initial upper bound U was equal to the burning number, '–' is used to reflect that there was no need for PRYM to solve GBP-IP for $B = U = b(G)$.

Table 4 Results obtained by PRYM when solving n-grids for $3 \leq n \leq 31$.

Instance			BFF-d		Best Bounds		$b(G)$	Time (sec)	#Cov. Constr. for:	
Name	$\lvert V\rvert$	$\lvert E\rvert$	L	U	Lower	Upper			$b(G) - 1$	$b(G)$
grid_003	9	12	2	3	3	3	3	1	4	–
grid_004	16	24	2	4	4	4	4	1	8	–
grid_005	25	40	2	4	4	4	4	1	7	–
grid_006	36	60	3	5	5	5	5	1	14	–
grid_007	49	84	3	6	5	5	5	1	10	21
grid_008	64	112	3	7	6	6	6	1	32	11
grid_009	81	144	4	8	6	6	6	1	20	21
grid_010	100	180	4	9	6	6	6	1	12	57
grid_011	121	220	4	10	7	7	7	1	36	28
grid_012	144	264	4	10	7	7	7	1	23	43
grid_013	169	312	5	11	7	7	7	1	19	78
grid_014	196	364	5	11	8	8	8	1	51	83
grid_015	225	420	5	12	8	8	8	1	35	96
grid_016	256	480	5	12	8	8	8	1	26	118
grid_017	289	544	5	12	9	9	9	2	65	106
grid_018	324	612	5	13	9	9	9	1	53	88
grid_019	361	684	5	13	9	9	9	2	35	137
grid_020	400	760	6	14	10	10	10	3	107	95
grid_021	441	840	5	13	10	10	10	4	70	156
grid_022	484	924	6	15	10	10	10	5	53	231
grid_023	529	1012	6	14	11	11	11	10	240	152
grid_024	576	1104	6	16	11	11	11	7	100	161
grid_025	625	1200	6	15	11	11	11	8	72	255
grid_026	676	1300	7	17	11	11	11	11	59	259
grid_027	729	1404	6	16	12	12	12	19	179	177
grid_028	784	1512	7	18	12	12	12	16	115	277
grid_029	841	1624	7	17	12	12	12	8	78	173
grid_030	900	1740	7	19	12	12	12	22	70	300
grid_031	961	1860	7	18	13	13	13	48	223	260

Table 5 Results obtained by `PRYM` when solving n-grids for $32 \leq n \leq 70$.

Instance			`BFF-d`		Best Bounds		$b(G)$	Time (sec)	#Cov. Constr. for:	
Name	$\|V\|$	$\|E\|$	L	U	Lower	Upper			$b(G)-1$	$b(G)$
grid_032	1024	1984	7	19	13	13	13	33	149	349
grid_033	1089	2112	7	19	13	13	13	36	114	335
grid_034	1156	2244	8	20	13	13	13	54	89	384
grid_035	1225	2380	8	20	14	14	14	190	258	317
grid_036	1296	2520	8	20	**14**	**14**	**14**	94	186	375
grid_037	1369	2664	8	21	**14**	**14**	**14**	119	132	542
grid_038	1444	2812	8	21	**14**	**14**	**14**	90	109	397
grid_039	1521	2964	8	21	**15**	**15**	**15**	934	394	370
grid_040	1600	3120	8	22	15	15	15	330	225	528
grid_041	1681	3280	8	22	**15**	**15**	**15**	322	177	549
grid_042	1764	3444	8	22	**15**	**15**	**15**	351	149	575
grid_043	1849	3612	8	22	**15**	**16**	–	1868	∗	∗
grid_044	1936	3784	9	23	**16**	**16**	**16**	1214	341	571
grid_045	2025	3960	9	23	**16**	**16**	**16**	789	236	659
grid_046	2116	4140	9	23	**16**	**16**	**16**	556	181	776
grid_047	2209	4324	9	23	**16**	**16**	**16**	1918	151	880
grid_048	2304	4512	9	24	**16**	**17**	–	1950	∗	∗
grid_049	2401	4704	9	24	**16**	**17**	–	2001	∗	∗
grid_050	2500	4900	9	24	**17**	**17**	**17**	1565	251	762
grid_051	2601	5100	9	24	**17**	**17**	**17**	1405	199	797
grid_052	2704	5304	9	25	**17**	**18**	–	2526	∗	∗
grid_053	2809	5512	9	25	**17**	**18**	–	2420	∗	∗
grid_054	2916	5724	9	25	**17**	**18**	–	2181	∗	∗
grid_055	3025	5940	9	25	**17**	**18**	–	2362	∗	∗
grid_056	3136	6160	10	26	**17**	**18**	–	2673	∗	∗
grid_057	3249	6384	10	26	**18**	**19**	–	3021	∗	∗
grid_058	3364	6612	10	26	**18**	**19**	–	2805	∗	∗
grid_059	3481	6844	10	26	**18**	**19**	–	2877	∗	∗
grid_060	3600	7080	10	27	**18**	**19**	–	2599	∗	∗
grid_061	3721	7320	10	27	**18**	**19**	–	3770	∗	∗
grid_062	3844	7564	10	27	**18**	**27**	–	3647	∗	∗
grid_063	3969	7812	10	27	**19**	**20**	–	5013	∗	∗
grid_064	4096	8064	10	28	**19**	**20**	–	3827	∗	∗
grid_065	4225	8320	10	28	**19**	**20**	–	3971	∗	∗
grid_066	4356	8580	10	28	**19**	**28**	–	3966	∗	∗
grid_067	4489	8844	10	28	**19**	**28**	–	3820	∗	∗
grid_068	4624	9112	11	29	**11**	**21**	–	4460	∗	∗
grid_069	4761	9384	11	29	**11**	**21**	–	4963	∗	∗
grid_070	4900	9660	11	29	**11**	**21**	–	5086	∗	∗

For the 58 $n \times n$ grids with $3 \leq n \leq 60$, we solved 46 instances within one hour of execution per grid, and for the remaining 12 grids, we determined a very tight optimality gap of exactly 1 unit on the length of an optimal burning sequence.

Notice that for $n \leq 60$, we have $0 \leq (\text{Upper} - \text{Lower}) \leq 1$, as well as for $n \in \{61, 63, 64, 65\}$. Moreover, since each $n \times n$ square grid is fully contained in the $n + 1 \times n + 1$ grid, we can conclude that the upper bound value 20 for grid_063 implies that grid_062 also has an upper bound 20 (instead of just the value Upper = 27 directly found by PRYM), which establishes a gap of 2 for the 62-grid. In a similar fashion, we can infer that Upper = 21 (established for grid_068) is an upper bound for grid_066 and grid_067.

Although it may appear that solving the GBP on grid graphs is more difficult than on social networks solely because the burning number is typically larger, the high symmetry of grids, compared to social networks, also significantly contributes to this complexity.

5 Concluding Remarks and Future Work

In this paper, we propose an exact algorithm for the GBP, namely PRYM. For a given arbitrary graph, PRYM finds an optimal burning sequence by means of solving multiple decision problems formulated as set covering IP models, while it generates essential covering constraints on demand. The algorithm takes advantage of the fact that a very small number of covering constraints is often sufficient for solving those decision problems, as was confirmed in practice.

Via computational experiments, we demonstrate that PRYM far outperforms the previously best known exact algorithm for GBP, and it is able to solve real-world instances with up to 3.09 million vertices in less than 19 minutes. For a collection of challenging instances composed of grid graphs with up to 5,000 vertices, PRYM was able to obtain newly provably optimal solutions and tight optimality gaps that were not previously reported in the literature.

We underscore that, while the overall success of PRYM relies on the combination of various techniques, the row generation method plays a crucial role, significantly impacting the results. The other techniques, including the computation of distances on demand and the branching rule, serve a secondary duty.

As for future research, we intend to analyze whether a column generation approach can also be successfully applied to the proposed IP formulation, extending PRYM to a branch-and-price algorithm and potentially increasing the suitability of the algorithm for solving the GBP for even larger graphs.

On top of that, we acknowledge that solving GBP-IP can be a bottleneck for PRYM, particularly as instance sizes grow. In future work, we plan to investigate advanced techniques for solving IP models, such as cutting planes, decomposition, reformulations with composite variables, and hybridization with constraint programming or decision diagrams.

Lastly, we aim to extend PRYM to solve variants of the GBP such as the k-Graph Burning Problem [21], in which k fire-souces are chosen to be set on fire in each round, and the GBP for edge-weighted graphs, where the weight of an edge $\{u, v\}$ determines the number of rounds the fire takes to spread from u to v and vice-versa.

References

1 Paul Bastide, Marthe Bonamy, Anthony Bonato, Pierre Charbit, Shahin Kamali, Théo Pierron, and Mikaël Rabie. Improved pyrotechnics: Closer to the burning number conjecture. *The Electronic Journal of Combinatorics*, 30(4), 2023. doi:10.37236/11113.

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 to Discrete Mathematics*, 16(1):185 – 197, 2021. doi:10.11575/cdm.v16i1.71194.

4 Anthony Bonato, Sean English, Bill Kay, and Daniel Moghbel. Improved bounds for burning fence graphs. *Graphs and Combinatorics*, 37(6):2761–2773, 2021. doi:10.1007/s00373-021-02390-x.

5 Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. Burning a graph as a model of social contagion. In *Algorithms and Models for the Web Graph*, pages 13–22, 2014. doi:10.1007/978-3-319-13123-8_2.

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

7 Anthony Bonato and Shahin Kamali. Approximation algorithms for graph burning. In *Theory and Applications of Models of Computation*, pages 74–92, 2019. doi:10.1007/978-3-030-14812-6_6.

8 Anthony Bonato and Thomas Lidbetter. Bounds on the burning numbers of spiders and path-forests. *Theoretical Computer Science*, 794:12–19, 2019. doi:10.1016/j.tcs.2018.05.035.

9 Felipe de Carvalho Pereira, Pedro Jussieu de Rezende, Tallys Yunes, and Luiz Fernando Batista Morato. A Row Generation Algorithm for Finding Optimal Burning Sequences of Large Graphs - Complementary Data. Mendeley Data, V2, 2024. doi:10.17632/c95hp3m4mz.

10 Felipe de Carvalho Pereira, Pedro Jussieu de Rezende, Tallys Yunes, and Luiz Fernando Batista Morato. Solving the graph burning problem for large graphs, 2024. arXiv:2404.17080.

11 Zahra Rezai Farokh, Maryam Tahmasbi, Zahra Haj Rajab Ali Tehrani, and Yousof Buali. New heuristics for burning graphs, 2020. doi:10.48550/arXiv.2003.09314.

12 Jesús García-Díaz and José Alejandro Cornejo-Acosta. A greedy heuristic for graph burning, 2024. doi:10.48550/arXiv.2401.07577.

13 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.

14 Jesús García-Díaz, Lil María Xibai Rodríguez-Henríquez, Julio César Pérez-Sansalvador, and Saúl Eduardo Pomares-Hernández. Graph burning: Mathematical formulations and optimal solutions. *Mathematics*, 10(15), 2022. doi:10.3390/math10152777.

15 Rahul Kumar Gautam, Anjeneya Swami Kare, and Durga Bhavani S. Faster heuristics for graph burning. *Applied Intelligence*, 52(2):1351–1361, 2022. doi:10.1007/s10489-021-02411-5.

16 Arya Tanmay Gupta, Swapnil A. Lokhande, and Kaushik Mondal. Burning grids and intervals. In *Algorithms and Discrete Applied Mathematics*, pages 66–79, 2021. doi:10.1007/978-3-030-67899-9_6.

17 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

18 Huiqing Liu, Xuejiao Hu, and Xiaolan Hu. Burning number of caterpillars. *Discrete Applied Mathematics*, 284:332–340, 2020. doi:10.1016/j.dam.2020.03.062.

19 Huiqing Liu, Ruiting Zhang, and Xiaolan Hu. Burning number of theta graphs. *Applied Mathematics and Computation*, 361:246–257, 2019. doi:10.1016/j.amc.2019.05.031.

20 Dieter Mitsche, Paweł Prałat, and Elham Roshanbin. Burning number of graph products. *Theoretical Computer Science*, 746:124–135, 2018. doi:10.1016/j.tcs.2018.06.036.

21 Debajyoti Mondal, N. Parthiban, V. Kavitha, and Indra Rajasingh. Apx-hardness and approximation for the k-burning number problem. In *WALCOM: Algorithms and Computation*, pages 272–283, 2021. doi:10.1007/978-3-030-68211-8_22.

22 Mahdi Nazeri, Ali Mollahosseini, and Iman Izadi. A centrality based genetic algorithm for the graph burning problem. *Applied Soft Computing*, 144:110493, 2023. doi:10.1016/j.asoc.2023.110493.

23 Sergey Norin and Jérémie Turcotte. The burning number conjecture holds asymptotically. *Journal of Combinatorial Theory, Series B*, 168:208–235, 2024. doi:10.1016/j.jctb.2024.05.003.

24 Ryan Anthony Rossi and Nesreen Kamel Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL: https://networkrepository.com.

25 Paulo Shakarian, Sean Eyre, and Damon Paulo. A scalable heuristic for viral marketing under the tipping model. *Social Network Analysis and Mining*, 3(4):1225–1248, 2013. doi:10.1007/s13278-013-0135-7.

26 Kai An Sim, Ta Sheng Tan, and Kok Bin Wong. On the burning number of generalized petersen graphs. *Bulletin of the Malaysian Mathematical Sciences Society*, 41(3):1657–1670, 2018. doi:10.1007/s40840-017-0585-6.

27 Ta Sheng Tan and Wen Chean Teh. Burnability of double spiders and path forests. *Applied Mathematics and Computation*, 438:127574, 2023. doi:10.1016/j.amc.2022.127574.

28 Marek Šimon, Ladislav Huraj, Iveta Dirgova Luptakova, and Jiri Pospichal. How to burn a network or spread alarm. *MENDEL*, 25(2):11–18, 2019. doi:10.13164/mendel.2019.2.011.

29 Marek Šimon, Ladislav Huraj, Iveta Dirgová Luptáková, and Jiří Pospíchal. Heuristics for spreading alarm throughout a network. *Applied Sciences*, 9(16), 2019. doi:10.3390/app9163269.