mathematics

MDPI

Article

Graph Burning: Mathematical Formulations and Optimal Solutions

Jesús García-Díaz [1,2,*], Lil María Xibai Rodríguez-Henríquez [1,2], Julio César Pérez-Sansalvador [1,2] and Saúl Eduardo Pomares-Hernández [2,3]

[1] Consejo Nacional de Ciencia y Tecnología, Mexico City 03940, Mexico
[2] Instituto Nacional de Astrofísica, Óptica y Electrónica, Coordinación de Ciencias Computacionales, Puebla 72840, Mexico
[3] Laboratoire d'Analyse et d'Architecture des Aystèmes du Centre National de la Recherche Scientifique (LAAS-CNRS), Université de Toulouse, INSA, F-31400 Toulouse, France
* Correspondence: jesus.garcia@conacyt.mx

Abstract: The graph burning problem is an NP-hard combinatorial optimization problem that helps quantify how vulnerable a graph is to contagion. This paper introduces three mathematical formulations of the problem: an integer linear program (ILP) and two constraint satisfaction problems (CSP1 and CSP2). Thanks to off-the-shelf optimization software, these formulations can be solved optimally over arbitrary graphs; this is relevant because the only algorithms designed to date for this problem are approximation algorithms and heuristics, which do not guarantee to find optimal solutions. We empirically compared the proposed formulations using random graphs and off-the-shelf optimization software. The results show that CSP1 and CSP2 tend to reach optimal solutions in less time than the ILP. Therefore, we executed them over some benchmark graphs of order at most 5908. The previously best-known solutions for some of these graphs were improved. We draw some empirical observations from the experimental results. For instance, we find the tendency: the larger the graph's optimal solution, the more difficult it is to find it. Finally, the resulting set of optimal solutions might be helpful as a benchmark dataset for the performance evaluation of non-exact algorithms.

Keywords: graph burning; integer programming; constraint satisfaction; social contagion

MSC: 90C27

Citation: García-Díaz, J.; Rodríguez-Henríquez, L.M.X.; Pérez-Sansalvador, J.C.; Pomares-Hernández, S.E. Graph Burning: Mathematical Formulations and Optimal Solutions. *Mathematics* **2022**, *10*, 2777. https://doi.org/10.3390/math10152777

Academic Editor: Mikhail Goubko

Received: 4 February 2022
Accepted: 1 August 2022
Published: 5 August 2022

1. Introduction

The graph burning problem (GBP) is an NP-hard combinatorial optimization problem introduced in 2014 in the context of social contagion [1]. This problem, concerned with the sequential spread of information over a graph, considers that information can be spread from different places and times [1,2]. In this paper, by graph we refer to an undirected simple graph [3]. Computer network message propagation is an example of a real-world problem that the GBP may model; in this scenario, an initial spreading entity can send a message one host at a time, while these hosts can propagate the message only to their neighbors [4,5]. The GBP may model other real-world problems: social contagion in social networks and the spread of viral infections under a very idealistic context [1,6].

The GBP receives a graph $G = (V, E)$ as input, and its goal is to find a minimum length sequence of vertices $(s_1, s_2, ..., s_k)$ that burns all graph's vertices by following the burning process. This process consists of repeating the following steps from $i = 1$ to k, where all vertices are unburned at the beginning, and once a vertex is burned, it remains in that state.

a The neighbors of the burned vertices get burned.

b Vertex s_i gets burned.

Any sequence that burns all the vertices of the input graph by following the burning process is known as a burning sequence. Thus, the GBP seeks a burning sequence of minimum length; this length is known as the burning number and is denoted by $b(G)$ [1,2]. Figure 1 shows the execution of the burning process over a nine vertices path with optimal burning sequence (v_3, v_7, v_9). Certainly, this example is very simple. Thus, for further clarification, Figure 2 shows the optimal burning sequence of more interesting graphs. These optimal solutions were computed using the proposed mathematical formulations, which are introduced in Section 3. In all these graphs, the bigger vertices are elements s_i of the burning sequence, and the color assigned to each vertex is the color of the vertex s_i with the smaller index that burns it.

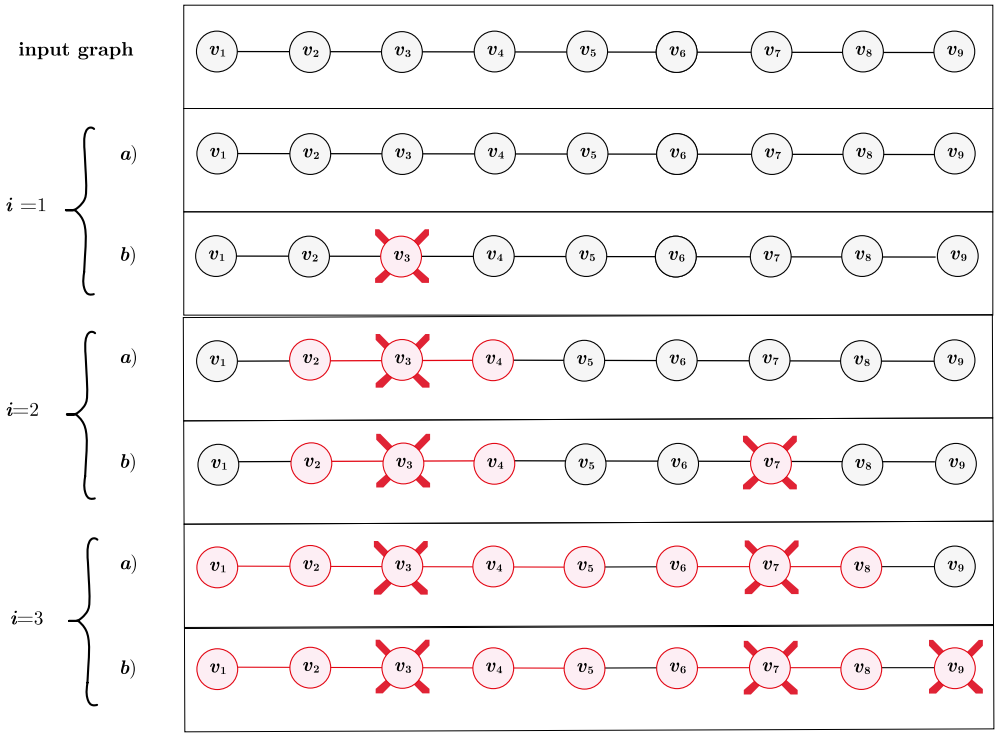


Figure 1. An optimal burning sequence for a nine vertices path is (v_3, v_7, v_9). The burning process of this sequence is depicted by each $i \in \{1, 2, 3\}$ with its corresponding steps a and b. Notice that vertex v_3 burns all vertices in $N_2[v_3]$, vertex v_7 burns all vertices in $N_1[v_7]$, and vertex v_9 burns all vertices in $N_0[v_9]$.

Due to its NP-hard nature, the GBP has been approached chiefly through approximation algorithms and heuristics. Some of these proposals are based on centrality measures and binary search over the possible values of the burning number $b(G)$. According to experimental results, these heuristics and approximation algorithms have an acceptable performance [4–8]. However, they do not guarantee to find optimal solutions. Therefore, there is a need for a mechanism for finding optimal solutions for arbitrary graphs. One of these mechanisms is mathematical formulations, which can be solved using off-the-shelf optimization software. For this reason, this paper introduces three novel mathematical formulations for the GBP: an integer linear program (ILP) and two constraint satisfaction problems (CSP1 and CSP2). Given the NP-hard nature of the problem, solving these formulations can take exponential time. Therefore, there must be a limit to their practicality. Through experimentation, we estimated such a limit.

The remaining part of the paper is organized as follows. Section 2 presents the background of the problem and the main definitions used in this document. Section 3 introduces the proposed mathematical formulations. Section 4 presents an empirical performance comparison among these formulations using random graphs and off-the-shelf

optimization software. Section 5 reports optimal solutions found by the implemented CSP1 and CSP2 over some synthetic and real-world graphs using off-the-shelf optimization software. Finally, Section 6 presents the concluding remarks.

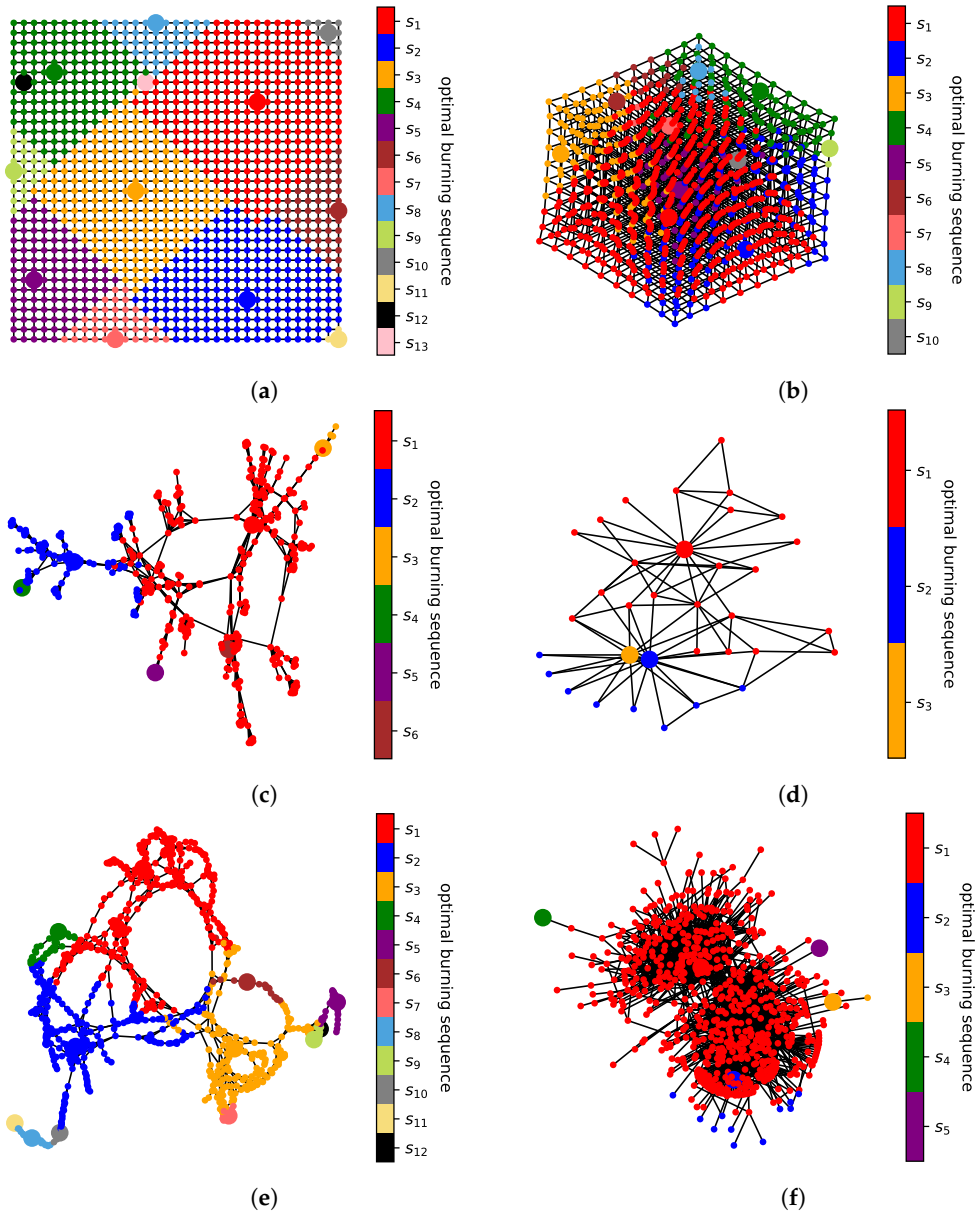


Figure 2. Optimal burning sequence $(s_1, s_2, ..., s_{b(G)})$ of (**a**) a two-dimensional lattice, (**b**) a three-dimensional lattice, (**c**) the ca-netscience graph, (**d**) the karate-club graph, (**e**) the DD199 graph, and (**f**) the web-polblogs graph.

2. Background

Let us begin by listing some basic definitions used throughout the paper. Observe that a graph is often called an undirected simple graph to distinguish it from directed graphs and multigraphs; however, in this paper, we only use the word graph to refer to this mathematical object.

Definition 1. *Subsets with k elements are known as k-element subsets [3].*

Definition 2. *A graph $G = (V, E)$ is an ordered pair consisting of a set of vertices V and a set of edges E, where E contains 2-element subsets of V. The vertices and edges of any given graph J are also represented by $V(J)$ and $E(J)$, respectively [3].*

Definition 3. *Given a graph $G = (V, E)$, the distance $d(u, v)$ between vertices $u, v \in V$ is defined as the number of edges in their shortest path.*

Definition 4. *Given a graph $G = (V, E)$, the open neighborhood $N(v)$ of a vertex $v \in V$ is the set of vertices at distance one from v. Notice that $v \notin N(v)$.*

Definition 5. *Given a graph $G = (V, E)$, the closed neighborhood $N[v]$ of a vertex $v \in V$ is the set of vertices at distance at most one from v. In other words, $N[v] = N(v) \cup \{v\}$.*

Definition 6. *Given a graph $G = (V, E)$, the closed k^{th} neighborhood $N_k[v]$ of a vertex $v \in V$ is the set of vertices at distance at most k from v. Notice that $N_0[v] = \{v\}$ and $N_1[v] = N[v]$.*

Definition 7. *A finite sequence with no repeated elements is a bijective function*

$$f : S \to \{x \in \mathbb{Z}^+ \mid x \leq |S|\}, \tag{1}$$

where S is the set of objects in the sequence and $|S|$ is its length. Notice that $f(v)$ is v's position in the sequence and that each object in S is assigned to exactly one position.

The decision version of the GBP is NP-complete. This problem receives as input a graph $G = (V, E)$ and a positive integer k; it asks if a burning sequence of length at most k exists. The problem remains NP-complete when restricted to trees of maximum degree three, chordal graphs, bipartite graphs, planar graphs, spider graphs, and disconnected graphs [2]. The optimization version of the problem also remains NP-hard for trees and graphs with disjoint paths [7]. Regarding arbitrary graphs, two approximation algorithms are reported in the literature; they have an approximation factor of 3 and $3 - 2/b(G)$, respectively [7,8]. There is a 2-approximation algorithm for trees, a 1.5-approximation algorithm for graphs with disjoint paths, and a 2-approximation algorithm for square grids [6,7]. For minimization problems, a ρ-approximation algorithm returns solutions of size at most $\rho \cdot OPT$, where OPT is the size of the optimal solution and $\rho \geq 1$. In the case of maximization problems, a ρ-approximation algorithm returns solutions of size at least $(1/\rho) \cdot OPT$ [9]. Besides approximation algorithms, some heuristics have been proposed too [4,5]; these are mainly based on centrality measures and binary search over the set of possible values of $b(G)$.

The GBP has been approached mostly from a theoretical point of view. As a result, many of its properties over specific graph families have been identified. Among these families are paths [2,7], trees [2,7], grids [6], intervals [6], fences [10], theta [11], spiders [12], path-forests [2,7,12], caterpillars [13], products [14], and generalized Petersen graphs [15]. Some of the main properties of the GBP are the following. All paths and cycles G of order n have $b(G) = \lceil n^{1/2} \rceil$, all graphs G with a Hamiltonian path have $b(G) \leq \lceil n^{1/2} \rceil$ [1,2], all spiders and caterpillars G have $b(G) \leq \lceil n^{1/2} \rceil$ [12,13], all complete graphs G of order at least two have $b(G) = 2$, and all perfect binary trees G of depth r have $b(G) = r + 1$. Based on these properties, a conjecture on the upper bound of the burning number of connected graphs was formulated by Bonato et al. [1]:

Conjecture 1. *Every connected graph G of order n has burning number $b(G) \leq \lceil n^{1/2} \rceil$.*

This conjecture, known as the burning number conjecture (BNC), is one of the most important open questions in the area. To date, the best-known bound for the burning number of arbitrary connected graphs is $b(G) \leq \lceil (4 \cdot n/3)^{1/2} \rceil + 1$ [16]. From the BNC, Conjecture 2 for disconnected graphs follows.

Conjecture 2. *The burning number $b(G)$ of a disconnected graph $G = (V, E)$ with p connected components $\{H_1, H_2, ..., H_p\}$ is at most $\sum_{i=1}^{p} \lceil |V(H_i)|^{1/2} \rceil$.*

In case Conjecture 1 is true, Conjecture 2 can be proved by observing the following facts. The concatenation of the optimal burning sequences of each H_i component is a burning sequence for the whole graph G. This is because concatenation does not reduce the burning capacity of any vertex. So, $b(G)$ is upper bounded by the length of the described concatenation.

$$b(G) \leq \sum_{i=1}^{p} b(H_i) \tag{2}$$

Assuming the BNC is true, $b(H_i) \leq \lceil |V(H_i)|^{1/2} \rceil$, where $|V(H_i)|$ is the number of vertices in the connected component H_i. Therefore,

$$b(G) \leq \sum_{i=1}^{p} \lceil |V(H_i)|^{1/2} \rceil \tag{3}$$

Anyway, from the best-known bound on the burning number over arbitrary connected graphs, we can prove the following lemma.

Lemma 1. *The burning number $b(G)$ of a disconnected graph $G = (V, E)$ with p connected components $\{H_1, H_2, ..., H_p\}$ is at most $p + \sum_{i=1}^{p} \lceil (4 \cdot |V(H_i)|/3)^{1/2} \rceil$.*

Proof. If we concatenate the optimal burning sequences of each H_i component, the resulting sequence is a burning sequence for the whole graph G. Since the concatenation does not reduce the burning capacity of any vertex, $b(G)$ is upper bounded by the length of this concatenation.

$$b(G) \leq \sum_{i=1}^{p} b(H_i) \tag{4}$$

Then, since $b(H_i) \leq \lceil (4 \cdot |V(H_i)|/3)^{1/2} \rceil + 1$ [16],

$$b(G) \leq p + \sum_{i=1}^{p} \lceil (4 \cdot |V(H_i)|/3)^{1/2} \rceil \tag{5}$$

□

Thanks to the best-known bounds on the burning number, the size of the explored search space may be reduced. Consistent with this observation, the proposed formulations tend to be solved faster when tighter lower and upper bounds on the burning number are available. Of course, feasible solutions returned by heuristics and approximation algorithms might help find better lower and upper bounds. We used such an approach in Section 5 to solve the problem optimally over some benchmark graphs.

The GBP resembles other NP-hard problems, such as the vertex k-center problem (VKCP) and the firefighter problem (FP). The VKCP consists of finding the best location for a set of $k \in \mathbb{Z}^+$ centers, where such locations are the ones that minimize the maximum distance a *customer* has to travel to its nearest center [17–19]. Although the VKCP and the GBP are different, their approximation algorithms are conceptually similar [7,8,19]. This similarity comes from the fact that the VKCP has a polynomial-time reduction to the minimum dominating set problem, which can be viewed as the problem of burning all vertices in parallel in one single step [19–22]. Regarding the FP, it aims at protecting vertices from burning given an initial set of *fire* sources [23–25]. Although GBP and FP have different goals, the latter's integer linear program (ILP) inspired us to define an ILP for the GBP.

To end this section, notice that the GBP can be stated in different ways. For instance, it can be formulated in terms of the burning process. However, it can be formulated as a covering problem too [1,2,6,13]:

Definition 8. *Given a simple graph $G = (V, E)$, the GBP consists of finding a minimum cardinality set $S \subseteq V$, and a bijective function $f : S \to \{1, 2, ..., |S|\}$ such that Equation (6) holds, where $b(G) = |S|$, and $N_{b(G)-f(v)}[v]$ is the closed $(b(G) - f(v))^{th}$ neighborhood of v.*

$$\bigcup_{v \in S} N_{b(G)-f(v)}[v] = V \tag{6}$$

By Definition 8, the GBP is a covering problem that seeks an optimal burning sequence with no repeated elements. Namely, it consists of finding a minimum length sequence $(s_1, s_2, ..., s_{b(G)})$ that cover all graph's vertices:

$$N_0[s_{b(G)}] \cup N_1[s_{b(G)-1}] \cup \cdots \cup N_{b(G)-2}[s_2] \cup N_{b(G)-1}[s_1] = V \tag{7}$$

3. Proposed Mathematical Formulations

This section introduces three novel mathematical formulations for the GBP: an ILP and two CSPs (CSP1 and CSP2). The ILP is based on the burning process and is inspired by the ILP for the FP [24]. Both CSP1 and CSP2 receive a guess B on $b(G)$ as input. In case $B \geq b(G)$, CSP1 returns a solution of length at most B, while CSP2 returns a solution of exactly length B. The difference between CSP1 and CSP2 is not arbitrary; it comes from their nature. Specifically, while CSP1 comes naturally from the ILP, CSP2 is based on Definition 8.

3.1. An Integer Linear Program

Expressions (8)–(20) define an ILP for the GBP. This formulation is based on the burning process and is mostly inspired by the ILP for the FP. This ILP has $O(n \cdot U)$ binary variables and $O(|E| \cdot U)$ linear constraints, where $G = (V, E)$ is an input graph of order n and $U \in \mathbb{Z}^+$ is an upper bound on $b(G)$. In case an upper bound is not available, U can be set to n. However, assuming the BNC is true (Conjecture 1), it is convenient to set $U = \lceil n^{1/2} \rceil$ if the graph is connected; otherwise, U can be set to $\sum_{i=1}^{p} \lceil |V(H_i)|^{1/2} \rceil$ (Conjecture 2), where $\{H_1, H_2, ..., H_p\}$ is the set of connected components of the graph. Of course, it might be safer to use the best-known bounds. This way, U can be set to $\lceil (4 \cdot n/3)^{1/2} \rceil + 1$, if the graph is connected [16]; otherwise, U can be set to $p + \sum_{i=1}^{p} \lceil (4 \cdot |V(H_i)|/3)^{1/2} \rceil$ (Lemma 1). For clarity, let us assume that the input graph's vertices are labeled as $\{v_1, v_2, ..., v_n\}$.

$$min \quad U - \sum_{j=1}^{U} b'_j \tag{8}$$

$$s.t. \quad s_{i,j-1} \leq s_{i,j} \qquad \forall v_i \in V, \forall j \in [1, U] \tag{9}$$

$$s_{i,j} \leq b_{i,j} \qquad \forall v_i \in V, \forall j \in [1, U] \tag{10}$$

$$b_{k,j-1} \leq b_{i,j} \qquad \forall v_i \in V, \forall v_k \in N(v_i), \forall j \in [1, U] \tag{11}$$

$$b_{i,j} \leq s_{i,j} + \sum_{v_k \in N(v_i)} b_{k,j-1} \qquad \forall v_i \in V, \forall j \in [1, U] \tag{12}$$

$$\sum_{v_i \in V} (s_{i,j} - s_{i,j-1}) = 1 \qquad \forall j \in [1, U] \tag{13}$$

$$b'_j \leq b_{i,j} \qquad \forall v_i \in V, \forall j \in [1, U] \tag{14}$$

$$b'_j \geq \sum_{v_i \in V} b_{i,j} - (n-1) \qquad \forall j \in [1, U] \tag{15}$$

$$\begin{aligned}
&\textit{where} && n = |V| && && (16)\\
& && U \in [b(G), n] && && (17)\\
& && s_{i,j}\ ,\ b_{i,j}\ ,\ b'_j \in \{0,1\} && \forall v_i \in V, \forall j \in [1,U] && (18)\\
& && b_{i,0} = 0 && \forall v_i \in V && (19)\\
& && s_{i,0} = 0 && \forall v_i \in V && (20)
\end{aligned}$$

Next, each formulation's element is explained.

- The main variables are $b_{i,j}, s_{i,j} \in \{0,1\}$; they model the burning process as follows.
  - $b_{i,j} = 1$ if and only if vertex v_i is burned at round j of the burning process.
  - $s_{i,j} = 1$ if and only if vertex v_i is in the sequence at round j of the burning process.
- Constraint (9) indicates that once a vertex is added to the sequence, it stays there.
- Constraint (10) indicates that a vertex added to the sequence at round j is considered a burned vertex.
- Constraint (11) indicates that all vertices in the neighborhood of some previously burned vertex must get burned.
- Constraint (12) prevents a vertex from getting burned if it does not have a neighbor previously burned or currently added to the sequence.
- Constraint (13) indicates that one new vertex must be added to the sequence at every round.
- By Constraints (14) and (15), the binary variable b'_j has a value of 1 if and only if all vertices are burned at round j. Namely, if 1 is interpreted as *True* and 0 as *False*, then these constraints are equivalent to the following logical constraint.

$$b'_j = \bigwedge_{v_i \in V} b_{i,j} \qquad \forall j \in [1,U] \tag{21}$$

- Variables b'_j are helpful for computing the value of the Objective function (8), which minimizes the number of rounds needed to burn all vertices. Notice that minimizing the Objective function (8) is equivalent to maximizing $\sum_{j=1}^{U} b'_j$. Furthermore, the length of the optimal burning sequence is $U - \sum_{j=1}^{U} b'_j + 1$. Namely, one unit is added because the last sequence's vertex is not reflected in the objective function.
- If a lower bound L on $b(G)$ is known, one can add the following restriction.

$$L \leq U - \sum_{j=1}^{U} b'_j + 1 \tag{22}$$

- Finally, Equations (19) and (20) indicate that all vertices are unburned at the beginning and that the burning sequence begins as an empty sequence.

Together, Expressions (8)–(20) define an ILP for the GBP. Thus, solving this ILP means solving the GBP for an arbitrary graph.

3.2. Two Constraint Satisfaction Problems

This section introduces two mathematical formulations for the problem: CSP1 and CSP2. Both have a parameter B as input, a guess on $b(G)$. Both formulations do not need an objective function to optimize due to this parameter. Thus, if $B \geq b(G)$, the solution to both formulations is a feasible solution of length at most B for the GBP. In case $B < b(G)$, the problems are unfeasible. Naturally, the obtained solution is feasible and optimal if $B = b(G)$. CSP1 comes from a slight modification of the ILP, and CSP2 is based on Definition 8. CSP1 has $O(n \cdot B)$ binary variables and $O(|E| \cdot B)$ linear constraints. CSP2 has $O(n^2)$ binary variables, $O(n)$ integer variables, and $O(n^2)$ linear constraints.

3.2.1. Constraint Satisfaction Problem 1

Expressions (23)–(34) define the CSP1. This formulation comes naturally from the proposed ILP. Variables $b_{i,j}$ and $s_{i,j}$ have the same meaning as before, and b'_j variables were removed. The set of constraints is almost the same, too; only Constraints (14) and (15) were replaced by Constraint (29). This constraint guarantees that all vertices are burned at the end of the burning process. If $B \geq b(G)$, a burning sequence of length at most B is codified into variables $s_{i,j}$. In case $B = b(G)$, the solution must be optimal. If $B < b(G)$, the model is unfeasible.

$$
\begin{aligned}
&find && s_{i,j} && \forall v_i \in V, \forall j \in [1,B] && (23)\\
&s.t. && s_{i,j-1} \leq s_{i,j} && \forall v_i \in V, \forall j \in [1,B] && (24)\\
& && s_{i,j} \leq b_{i,j} && \forall v_i \in V, \forall j \in [1,B] && (25)\\
& && b_{k,j-1} \leq b_{i,j} && \forall v_i \in V, \forall v_k \in N(v_i), \forall j \in [1,B] && (26)\\
& && b_{i,j} \leq s_{i,j} + \sum_{v_k \in N(v_i)} b_{k,j-1} && \forall v_i \in V, \forall j \in [1,B] && (27)\\
& && \sum_{v_i \in V} \left(s_{i,j} - s_{i,j-1}\right) = 1 && \forall j \in [1,B] && (28)\\
& && \sum_{v_i \in V} b_{i,B} = n && && (29)\\
&where && n = |V| && && (30)\\
& && B \in [b(G), n] && && (31)\\
& && s_{i,j}\ ,\ b_{i,j} \in \{0,1\} && \forall v_i \in V, \forall j \in [1,B] && (32)\\
& && b_{i,0} = 0 && \forall v_i \in V && (33)\\
& && s_{i,0} = 0 && \forall v_i \in V && (34)
\end{aligned}
$$

In general, $b(G)$ cannot be known in advance. Therefore, the possible values it can take have to be explored. In order to make this search less time-consuming, a binary search can be performed. Section 3.2.3 shows how to incorporate CSP1 into a binary search.

3.2.2. Constraint Satisfaction Problem 2

Expressions (35)–(52) define the CSP2, where B is a guess on $b(G)$. If $B \geq b(G)$, the solution to this problem is a feasible solution of length B for the GBP. In case $B < b(G)$, the problem is unfeasible. If $B = b(G)$, the obtained solution is feasible and optimal. The CSP2 has $O(n^2)$ binary variables, $O(n)$ integer variables, and $O(n^2)$ linear constraints. This formulation is based on Definition 8, which states the GBP as a covering problem. Furthermore, CSP2 requires knowing the distance $d_{i,j}$ between every pair of vertices $v_i, v_j \in V$. These distances can be computed in polynomial-time by applying an appropriate algorithm over the input graph $G = (V, E)$, such as a Breadth-First Search-based algorithm [26]. If the input graph is disconnected, the undefined distance between vertices in different connected components must be replaced by $n + 1$, which is greater than the maximum distance between connected vertices.

$$
\begin{aligned}
&find && p_i && \forall v_i \in V && (35)\\
&s.t. && \sum_{v_i \in V} x_i = B && && (36)\\
& && x_i \leq p_i && \forall v_i \in V && (37)\\
& && p_i \leq x_i \cdot B && \forall v_i \in V && (38)\\
& && \sum_{v_i \in V} z_{i,j} = 1 && \forall j \in [1,B] && (39)
\end{aligned}
$$

$$\sum_{j=1}^{B+1} z_{i,j} = 1 \qquad \forall v_i \in V \tag{40}$$

$$p_i = \sum_{j=1}^{B} j \cdot z_{i,j} \qquad \forall v_i \in V \tag{41}$$

$$y_{i,j} \leq x_i \qquad \forall v_i, v_j \in V \tag{42}$$

$$x_i \cdot d_{i,j} \leq x_i \cdot B - p_i + (1 - y_{i,j}) \cdot M \qquad \forall v_i, v_j \in V \tag{43}$$

$$x_i \cdot d_{i,j} \geq x_i \cdot (B + \epsilon) - p_i - y_{i,j} \cdot M \qquad \forall v_i, v_j \in V \tag{44}$$

$$\sum_{v_i \in V} y_{i,j} \geq 1 \qquad \forall v_j \in V \tag{45}$$

where

$$n = |V| \tag{46}$$

$$B \in [b(G), n] \tag{47}$$

$$x_i \, , \, z_{i,j} \in \{0, 1\} \qquad \forall v_i \in V, \forall j \in [1, B+1] \tag{48}$$

$$p_i \in \{0, 1, ..., B\} \qquad \forall v_i \in V \tag{49}$$

$$d_{i,j} \in [0, n+1] \qquad \forall v_i, v_j \in V \tag{50}$$

$$y_{i,j} \in \{0, 1\} \qquad \forall v_i, v_j \in V \tag{51}$$

$$M \geq n + 1 + \epsilon \tag{52}$$

Like CSP1, CSP2 needs to know the burning number $b(G)$ in advance; this seems to be a disadvantage. However, this issue can be lessened by performing a binary search over the set of possible burning number values (See Section 3.2.3). Nevertheless, let us first describe the CSP2, assuming that the burning number $b(G)$ is known in advance. Namely, $B = b(G)$. In general terms, CSP2 seeks to assign values to variables p_i that codifies a bijective function $f : S \to \{1, 2, ..., B\}$ such that Equation (53) holds, where $|S| = B$.

$$\bigcup_{v \in S} N_{B - f(v)}[v] = V \tag{53}$$

The variables $x_i \in \{0, 1\}$ take a value $x_i = 1$ if and only if vertex $v_i \in S$. The variables $p_i \in \{0, 1, ..., B\}$ indicate the position of each vertex in the burning sequence. More specifically, $p_i = f(v_i)$ for all vertices $v_i \in V$ such that $x_i = 1$. If a vertex $v_i \in V$ is not in the burning sequence, then $p_i = 0$. In order to avoid inconsistencies in the output, variables $z_{i,j}$ and $y_{i,j}$ are added to the formulation. Next, we proceed to explain each formulation's element in detail.

- First, let us define variables x_i and p_i, where $v_i \in V$:
  Variables x_i indicate which vertices are part of the burning sequence, and variables p_i indicate the position of a vertex in the burning sequence.
  - If vertex $v_i \in V$ is part of the burning sequence, then $x_i = 1$; otherwise, $x_i = 0$.
  - If vertex $v_i \in V$ is part of the burning sequence, then p_i is equal to the position of vertex v_i into the burning sequence, i.e., $p_i = f(v_i)$; otherwise, $p_i = 0$.
  - Constraint (36) indicates that the number of vertices in the burning sequence must be equal to B.
  - To avoid inconsistencies, Constraints (37) and (38) indicate that $p_i = 0$ if and only if $x_i = 0$. Namely, vertices not in the sequence do not have an assigned position.
- Secondly, we define variables $z_{i,j}$. These variables are used to guarantee that each element of the burning sequence has a unique position. That is to say that $f : S \to \{1, 2, ..., B\}$ is bijective.
  - There are $n \cdot (B + 1)$ variables $z_{i,j}$. If a vertex v_i is at position p_i in the burning sequence, then $z_{i,p_i} = 1$ and $z_{i,j} = 0$ for all $j \neq p_i$. If vertex v_i is not in the burning sequence, then $z_{i,B+1} = 1$ and $z_{i,j} = 0$ for all $j \in \{1, 2, ..., B\}$.
  - Constraint (39) indicates that each burning sequence position must be associated with exactly one vertex.

- Constraint (40) indicates that each vertex must be associated to exactly one position or to no position at all, where index $j = B + 1$ is used to indicate which vertices are not in the burning sequence.
- Constraints (40) and (41) guarantee that variables p_i and $z_{i,j}$ are consistent. Namely, if v_i is assigned to some position $j \in \{1, 2, ..., B\}$, then $\sum_{j=1}^{B} z_{i,j} = 1$, and p_i is equal to the index j such that $z_{i,j} = 1$.
- Together, Constraints (39)–(41) indicate that each vertex v_i in the burning sequence must have a unique position. In other words, they guarantee that the assignment function $f : S \to \{1, 2, ..., B\}$ is bijective.

- Finally, we define variables $y_{i,j}$.
  These binary variables are used to guarantee that every vertex is burned. In other words, they guarantee that Equation (53) holds. If a vertex v_j is burned by a vertex v_i in the burning sequence, then $y_{i,j} = 1$; otherwise, $y_{i,j} = 0$.
  - $d_{i,j}$ is the distance between v_i and v_j.
  - Constraint (42) indicates that a vertex v_j can only be burned by vertices in the burning sequence. Namely, v_j can be burned only by vertices v_i such that $x_i = 1$.
  - Constraints (43) and (44) require two constants M and ϵ. Regarding ϵ, Constraint (44) is equivalent to:

$$x_i \cdot d_{i,j} > x_i \cdot B - p_i - y_{i,j} \cdot M \tag{54}$$

    However, since optimization solvers do not accept strict inequalities, ϵ had to be incorporated. The value assigned to ϵ must be any positive number strictly smaller than one and will depend on the arithmetic precision of the computer being used. In our experimental setup, we set $\epsilon = 0.001$. Regarding M, its minimum value that respects Constraints (43) and (44) under all the possible values of variables x_i, $y_{i,j}$, and p_i is $M = n + 1 + \epsilon$, which is greater than the distance between any two connected vertices in the graph.
  - Constraint (43) indicates that $d_{i,j}$ has to be at most $B - p_i$ when $y_{i,j} = 1$ and $x_i = 1$. Namely, v_i is responsible for burning v_j only if they are at an appropriate distance, which depends on the position of v_i in the burning sequence.
  - Constraint (44) indicates that $d_{i,j}$ has to be greater than $B - p_i$ when $y_{i,j} = 0$ and $x_i = 1$. Namely, v_i is not responsible for burning v_j when the distance between them is not appropriate.
  - Notice that Constraints (43) and (44) are automatically satisfied by all vertices not in the burning sequence. That is, vertices v_i such that $x_i = 0$.
  - Together, Constraints (42)–(44) indicate that if $d(v_i, v_j) \leq B - p_i$, then $y_{i,j} = 1$, where v_i is a vertex in the burning sequence; otherwise, $y_{i,j} = 0$. In other words, $y_{i,j} = 1$ if vertex v_i is at an appropriate distance for burning v_j.
  - As mentioned before, the undefined distance between vertices v_i and v_j in different connected components must be replaced by $n + 1$. This way, constraint (43) cannot be satisfied by $x_i = 1$ and $y_{i,j} = 1$. Namely, if vertex v_i is in the optimal burning sequence, then it cannot be responsible for burning a vertex v_j at distance $n + 1$ (because they are in different connected components). Moreover, notice that $x_i = 1$ and $y_{i,j} = 0$ do satisfy Constraint (43) when the distance between v_i and v_j is $n + 1$ because $M \geq n + 1 + \epsilon$. Take into account that $b(G) \leq B \leq n$.
  - Constraint (45) indicates that all vertices have to be burned.

Expressions (35)–(52) define the CSP2. Thus, solving this problem with $B = b(G)$ means solving the GBP for an arbitrary graph G. This formulation can be implemented and solved by off-the-shelf optimization software if the burning number $b(G)$ is known. However, knowing $b(G)$ in advance for arbitrary graphs is not possible. One way to solve this issue is by trying all n possible values of $b(G)$ but at the expense of time. In order to reduce the number of guesses from n to $\log n$, a binary search can be performed.

3.2.3. Adding Binary Search

Algorithms 1 and 2 incorporate CSP1 and CSP2 into a binary search. The binary search aims to explore at most $\log n$ possible values of $b(G)$ until its exact value is reached. This way, CSP1 and CSP2 are solved with such value, and the optimal solution to the problem is returned. Lemmas 2 and 3 show the correctness of both procedures.

Algorithm 1 CSP1 + BS

Input: A graph $G = (V, E)$, a lower bound L, and an upper bound U
Output: A burning sequence s
1: $low = L$
2: $high = U$
3: **while** $low \leq high$ **do**
4: $B = \lfloor (high + low)/2 \rfloor$
5: $s =$ Solution to the CSP1 over (G, B)
6: **if** s exists **then**
7: $high = length(s) - 1$
8: **else**
9: $low = B + 1$
10: **end if**
11: **end while**
12: **return** the last computed feasible solution s

Algorithm 2 CSP2 + BS

Input: A graph $G = (V, E)$, a lower bound L, and an upper bound U
Output: A burning sequence s
1: $D =$ All-pairs shortest paths of G
2: $low = L$
3: $high = U$
4: **while** $low \leq high$ **do**
5: $B = \lfloor (high + low)/2 \rfloor$
6: $s =$ Solution to the CSP2 over (G, D, B)
7: **if** s exists **then**
8: $high = B - 1$
9: **else**
10: $low = B + 1$
11: **end if**
12: **end while**
13: **return** the last computed feasible solution s

Lemma 2. *If $L \leq b(G) \leq U$, Algorithm 1 (CSP1 + BS) returns the optimal burning sequence of G.*

Proof. CSP1 + BS performs a binary search over the ordered set of possible values of $b(G)$, which goes from L to U. At each iteration of the binary search, the CSP1 is solved (line 5); if a burning sequence of length at most B is found (line 6), then a shorter burning sequence may exist. Therefore, $high$ is set to $length(s) - 1$ (line 7) because the sequence returned by the CSP1 might have a length smaller than B. If a burning sequence of size B does not exist (line 8), then a shorter burning sequence neither exists. Thus, low is set to $B + 1$ (line 9). This way, at the last iteration of the binary search, the value of B can be one of the following: the largest that makes the problem unfeasible or the smallest that makes the problem feasible. Since the returned solution is the last computed feasible one, it has to be optimal. □

Lemma 3. *If $L \leq b(G) \leq U$, Algorithm 2 (CSP2 + BS) returns the optimal burning sequence of G.*

Proof. The proof is very similar to the previous one. CSP2 + BS performs a binary search over the ordered set of possible values of $b(G)$, which goes from L to U. CSP2 + BS requires the distance between every pair of vertices (line 1). These values can be obtained by executing an appropriate polynomial-time algorithm, such as a Breadth-First Search-based algorithm. At each iteration of the binary search, the CSP2 is solved (line 6); if a burning sequence of length B is found (line 7), then a shorter burning sequence may exist. Therefore, *high* is set to $B - 1$ (line 8). If a burning sequence of size B does not exist (line 9), then a shorter burning sequence neither exists. Thus, *low* is set to $B + 1$ (line 10). This way, at the last iteration of the binary search, the value of B can be one of the following: the largest that makes the problem unfeasible or the smallest that makes the problem feasible. Since the returned solution is the last computed feasible one, it has to be optimal. □

4. Experimental Results

This section presents two sets of experiments. The first one aims to test the correct implementation of the proposed formulations (Section 4.1); the second one empirically compares them using random graphs (Section 4.2). We implemented the proposed mathematical formulations using Gurobi version 9.5.1 [27]. This off-the-shelf optimization software executes optimization techniques such as simplex, branch-and-bound, branch-and-cut, cutting planes, parallelism, and heuristics to find optimal solutions to mathematical formulations with linear and quadratic constraints. In all the experiments, we used default Gurobi's parameters, including $Presolve = 2$, $Heuristics = 0.05$, and $MIPGap = 0$. $Presolve = 2$ generates an equivalent and smaller formulation mapped into the original one; this usually improves the running time. $Heuristics = 0.05$ indicates that 5% of the running time is spent by heuristic search. $MIPGap = 0$ guarantees that the formulation is solved to optimality. All experiments were executed on an Asus F15 laptop with a 2.50 GHz Intel i5-10300H processor, 32 GB of RAM, and a Windows 11 operating system.

Section 4.1 shows the results obtained by applying the implemented formulations over a set of graphs with a known burning number. This way, we provide inductive evidence of their correct implementation. Section 4.2 presents an empirical performance comparison among the formulations using random graphs.

4.1. Empirical Validation

In order to check the correct implementation of the proposed mathematical formulations, we executed them over a set of graphs with a known burning number. These graphs are paths, and a family of tight examples for the BFF algorithm, a recently published approximation algorithm for the GBP [8]. Since setting tight lower and upper bounds might help improve the running time, we set $L = b(G) - 2$ and $U = b(G) + 2$ for all tests. Tables 1 and 2 show two running times for each formulation: t_o and t_p. The time it takes Gurobi to reach the optimal solution is denoted by t_o. The time it takes Gurobi to find the optimal solution and guarantee it is optimal is denoted by t_p. In other words, t_p is the time to proven optimality. We set a maximum running time of 2000 s; a reported value of $>$2000 means that an optimal solution could not be found within that time.

Tables 1 and 2 show that the ILP and CSP1 + BS could not be solved in less than 2000 s for most graphs. Regarding CSP2 + BS, it could solve all the problems within 2000 s. However, this is not evidence of any practical superiority of CSP2 + BS against the other formulations. Recall that the primary goal of this subsection is to provide inductive evidence of the correct implementation of the mathematical formulations. All experiments that finished in less than 2000 s arrived to the known optimal solution. For a detailed comparison of these formulations, we introduce the following section.

Table 1. Running time of the implemented mathematical formulations over a set of paths.

Paths				Bounds		ILP		CSP1 + BS		CSP2 + BS	
$\|V\|$	$\|E\|$	ρ	$b(G)$	L	U	t_o	t_p	t_o	t_p	t_o	t_p
16	15	0.125	4	2	6	0.059	0.068	0.015	0.016	0.016	0.060
25	24	0.080	5	3	7	0.231	0.239	0.016	0.078	0.023	0.173
36	35	0.056	6	4	8	1.086	3.9	0.184	0.637	0.078	0.410
49	48	0.041	7	5	9	86.4	242.2	1.5	5.2	0.148	0.769
64	63	0.031	8	6	10	23.9	>2000	2.3	105.9	0.250	1.3
81	80	0.025	9	7	11	>2000	>2000	21.0	202.4	0.429	2.2
100	99	0.020	10	8	12	>2000	>2000	>2000	>2000	1.0	3.8
121	120	0.017	11	9	13	>2000	>2000	>2000	>2000	2.2	6.4
144	143	0.014	12	10	14	>2000	>2000	>2000	>2000	1.8	8.1
169	168	0.012	13	11	15	>2000	>2000	>2000	>2000	3.0	12.3
196	195	0.010	14	12	16	>2000	>2000	>2000	>2000	5.6	19.1
225	224	0.009	15	13	17	>2000	>2000	>2000	>2000	16.8	36.9
256	255	0.008	16	14	18	>2000	>2000	>2000	>2000	142.9	170.7
289	288	0.007	17	15	19	>2000	>2000	>2000	>2000	63.8	100.7
324	323	0.006	18	16	20	>2000	>2000	>2000	>2000	126.1	177.7
361	360	0.006	19	17	21	>2000	>2000	>2000	>2000	347.4	417.7

Table 2. Running time of the implemented mathematical formulations over a set of connected tight examples for the BFF algorithm.

BFF's Connected Tight Examples				Bounds		ILP		CSP1 + BS		CSP2 + BS	
$\|V\|$	$\|E\|$	ρ	$b(G)$	L	U	t_o	t_p	t_o	t_p	t_o	t_p
6	5	0.333	2	1	4	0.009	0.012	0.001	0.004	0.002	0.014
19	18	0.105	3	1	5	0.016	0.016	0.001	0.011	0.012	0.080
40	39	0.050	4	2	6	0.110	0.110	0.001	0.042	0.056	0.354
69	68	0.029	5	3	7	0.662	0.662	0.110	0.188	0.221	1.2
106	105	0.189	6	4	8	6.0	6.0	0.701	0.874	0.629	3.1
151	150	0.013	7	5	9	31.0	31.0	4.9	5.4	1.3	6.8
204	203	0.010	8	6	10	449.8	449.8	4.5	6.7	2.9	13.5
265	264	0.008	9	7	11	>2000	>2000	1400	1400	5.8	24.9
334	333	0.006	10	8	12	>2000	>2000	1900	1900	11.5	44.5
411	410	0.005	11	9	13	>2000	>2000	>2000	>2000	21.1	75.3
496	495	0.004	12	10	14	>2000	>2000	>2000	>2000	40.1	139.1
589	588	0.003	13	11	15	>2000	>2000	>2000	>2000	68.2	232.4
690	689	0.003	14	12	16	>2000	>2000	>2000	>2000	102.1	336.2
799	798	0.003	15	13	17	>2000	>2000	>2000	>2000	172.8	528.8
916	915	0.002	16	14	18	>2000	>2000	>2000	>2000	256.3	799.7
1041	1040	0.002	17	15	19	>2000	>2000	>2000	>2000	349.1	1100

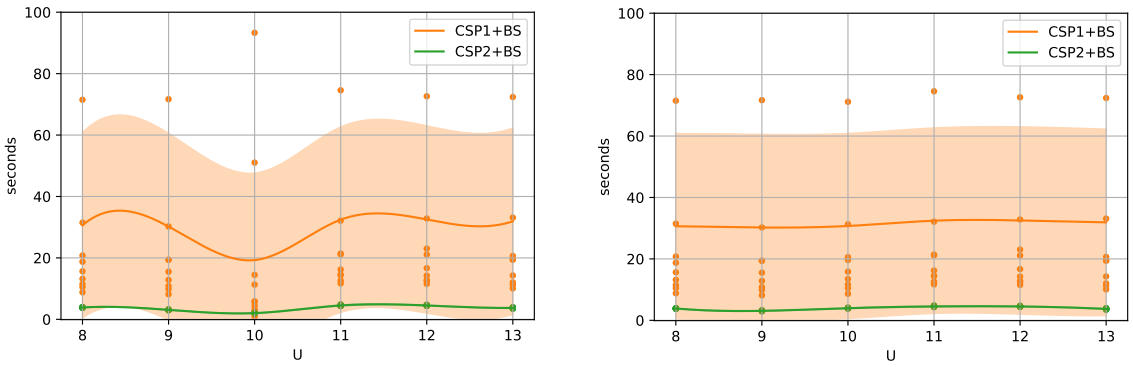
4.2. Empirical Performance Comparison

The results reported in this subsection are related to the following intuitive observations. The sparser (denser) a graph is, the larger (smaller) its burning number tends to be. On the one hand, the densest possible graph of order n is the complete graph K_n, which has $b(K_n) = 2$ (notice that any pair of vertices is enough to burn the whole graph.) On the other hand, the sparsest possible graph of order n is the complement $\overline{K_n}$ of the complete graph K_n, which has $b(\overline{K_n}) = n$ (each vertex can be burned only by itself.) From these observations, we notice the tendency: the sparser (denser) a graph is, its set of optimal solutions tends to be smaller (larger). Therefore, denser connected graphs tend to be easier to solve than sparser connected graphs. The experiments reported in this subsection confirm this tendency. Specifically, we created four sets of connected graphs following the Erdős-Rényi model (also known as random graphs) with parameter $p \in \{0.02, 0.023, 0.03, 0.04\}$ [28]. We selected these values through trial and error. In more detail, we tried several values p, and we observed that values $\{0.02, 0.023, 0.03, 0.04\}$ tend to generate graphs with burning num-

ber $\{8, 7, 6, 5\}$, respectively, (this applies to graphs of order 100.) We excluded disconnected graphs because the Erdős-Rényi model tends to generate graphs with too many connected components when $p \leq 0.02$. The optimal burning sequence of such graphs is usually easy to find by just picking one vertex from each component.

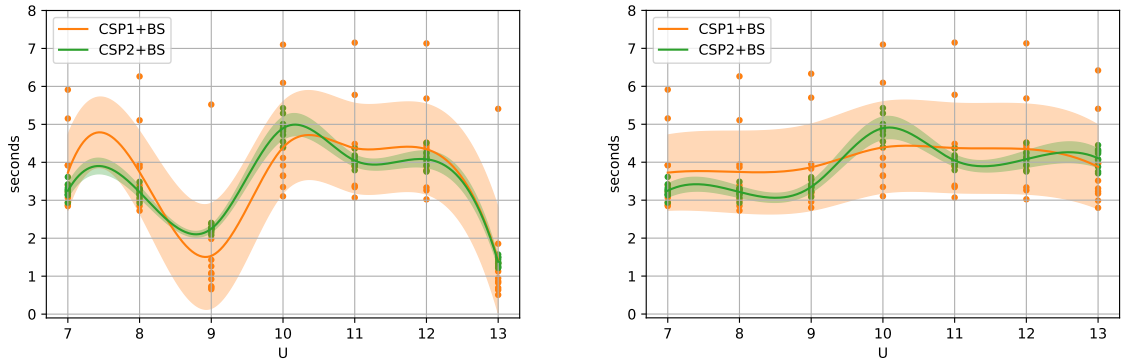
For each value $\{0.02, 0.023, 0.03, 0.04\}$, we created ten connected random graphs of order one hundred. Then, we executed the implemented formulations over them. Figures 3–6 show the obtained results. In these figures, each blue, orange, and green dot represents the running time of the implemented ILP, CSP1 + BS, and CSP2 + BS over one random graph. In the case of the ILP and CSP1 + BS, some dots do not appear because they correspond to values larger than the maximum reported on the vertical axis. The continuous line interpolates the mean of each set of ten dots. The shaded region is the standard deviation around the mean. Regarding the ILP, we do not report its running time over graphs G with $b(G)$ equal to eight and seven because it showed an excessive running time of thousands and hundreds of seconds, respectively. In all these figures, Subfigures (a) and (b) report the time to optimality (t_o) and the time to proven optimality (t_p), respectively.

From Figures 3 and 4, we can observe that the CSP2 + BS has a better performance over graphs with a relatively large burning number (seven or more). Figures 5 and 6 show that the CSP1 + BS is better suited for graphs with a relatively small burning number (six or less). These results are consistent with those reported in Tables 1 and 2. Of course, there can be exceptions to these empirical observations.



(**a**) (**b**)

Figure 3. (**a**) Time to the optimal solution and (**b**) to proven optimality reported by Gurobi for the CSP1 + BS and CSP2 + BS over 10 random graphs of order 100, $p = 0.02$, and $b(G) = 8$. The bounds are $L = 1$ and $U \in [b(G), \lceil (4 \cdot 100/3)^{1/2} \rceil + 1]$.



(**a**) (**b**)

Figure 4. (**a**) Time to the optimal solution and (**b**) to proven optimality reported by Gurobi for the CSP1 + BS and CSP2 + BS over 10 random graphs of order 100, $p = 0.023$, and $b(G) = 7$. The bounds are $L = 1$ and $U \in [b(G), \lceil (4 \cdot 100/3)^{1/2} \rceil + 1]$.

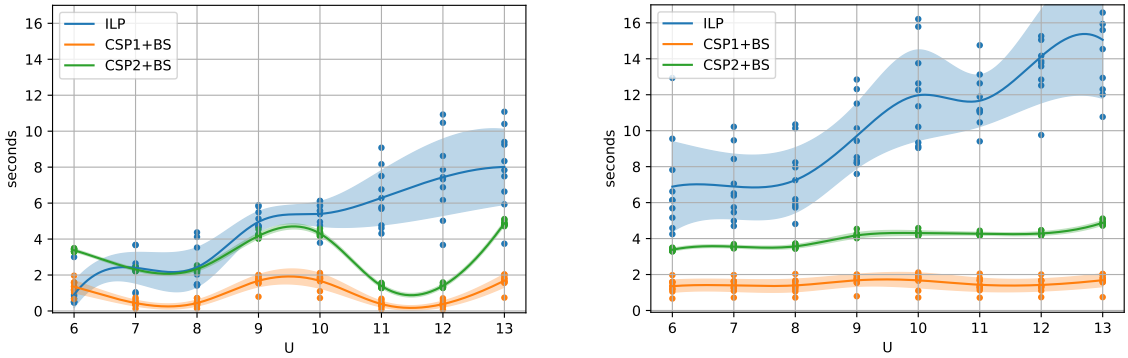


Figure 5. (**a**) Time to the optimal solution and (**b**) to proven optimality reported by Gurobi for the ILP, CSP1 + BS, and CSP2 + BS over 10 random graphs of order 100, $p = 0.03$, and $b(G) = 6$. The bounds are $L = 1$ and $U \in [b(G), \lceil (4 \cdot 100/3)^{1/2} \rceil + 1]$.

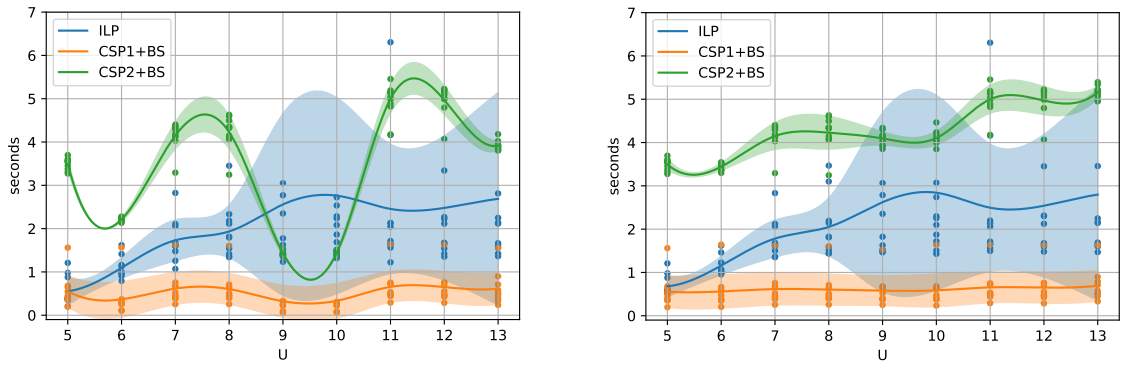


Figure 6. (**a**) Time to the optimal solution and (**b**) to proven optimality reported by Gurobi for the ILP, CSP1 + BS, and CSP2 + BS over 10 random graphs of order 100, $p = 0.04$, and $b(G) = 5$. The bounds are $L = 1$ and $U \in [b(G), \lceil (4 \cdot 100/3)^{1/2} \rceil + 1]$.

5. Computing Optimal Solutions

The previous section shows that the CSP2 + BS seems better suited for solving graphs with a relatively large burning number. Nevertheless, CSP1 + BS seems better for solving graphs with a relatively small burning number. In this section, we executed CSP1 + BS and CSP2 + BS over synthetic and real-world benchmark graphs (See Table 3); most of these were taken from the network repository [29] and the Stanford large network dataset collection (SNAP) [30]. Since this experimentation aims to find optimal burning sequences, we set the lower and upper bounds to the tightest known. To find the upper bound, we executed three state-of-the-art heuristics: improved cutting corners heuristic (ICCH), backbone-based greedy heuristic (BBGH), and component-based recursive heuristic (CBRH) [5]; their authors kindly provided the implementation of these. We also executed the BFF and BFF+ approximation algorithms, where BFF+ returns the best possible solution BFF can find. To set the lower bound, we exploited that all solutions generated by the BFF algorithm have a length of at most $3 \cdot b(G) - 2$ [8]. Therefore, we computed the lower bound with $L = \lceil (worst + 2)/3 \rceil$, where $worst$ is the worst solution BFF can return. Since $b(G)$ is unknown, the time reported is the time to proven optimality (t_p).

By the previous set of experiments, it seems likely that CSP1 + BS would be very inefficient over graphs with a burning number of seven or more. Therefore, we executed CSP1 + BS over graphs with an upper bound on the burning number of six or less. Furthermore, since CSP1 + BS has fewer memory requirements than CSP2 + BS, we could execute the former over graphs of order up to 5908. Regarding CSP2 + BS, we executed it over graphs with an upper bound on the burning number of at least seven and order at

most 1458 because Gurobi's branch-and-bound algorithm exhausted all available memory when executed over bigger graphs. This way, the results reported in Table 3 confirm the optimality of most previously known solutions. Furthermore, the optimal solution of some graphs is reported for the first time (ca-netscience, web-polblogs, DD68, DD199, DD497, lattice2D, and tech-routers-rf). The TVshow graph could not be solved within ∼48 h using CSP1 + BS; this goes along with the observation that CSP1 + BS does not seem adequate for solving graphs with a relatively *large* burning number. From Table 3, we also observe that synthetic graphs with a well-defined structure seem more challenging to solve. For instance, lattice3D required 150,000 s (∼42 h) to be solved using CSP2 + BS. In order to find optimal solutions for challenging graphs, we executed CSP1 + BS and CSP2 + BS over some grid graphs (See Table 4). As before, we executed CSP1 + BS and CSP2 + BS over graphs with a known upper bound of at most six and at least seven, respectively. From Table 4, we can observe that the implemented formulations found twelve optimal solutions that the state-of-the-art heuristics could not find.

Table 3. Optimal solutions found by CSP1 + BS and CSP2 + BS over some benchmark graphs. The feasible solutions found by some state-of-the-art algorithms are reported too. Optimal solutions reported for the first time are underlined.

Graphs				Bounds			t_p		BFF		Heuristics		
Name	$\|V\|$	$\|E\|$	ρ	$b(G)$	L	U	**CSP1 + BS**	**CSP2 + BS**	**Worst**	**Best**	**ICCH**	**BBGH**	**CBRH**
karate-club	34	78	0.139	3	3	3	0.015	-	5	3	3	3	4
chesapeake	39	170	0.229	3	2	3	0.023	-	4	3	3	3	3
dolphins	62	159	0.084	4	3	4	0.103	-	7	4	4	5	5
rt-retweet	96	117	0.026	5	4	5	0.125	-	9	5	5	5	5
polbooks	105	441	0.081	4	3	4	0.157	-	6	4	4	4	5
adjnoun	112	425	0.068	4	3	4	0.160	-	6	4	4	4	4
ia-infect-hyper	113	2196	0.347	3	2	3	0.350	-	4	3	3	3	3
C125-9	125	6963	0.898	3	2	3	0.924	-	3	3	3	3	3
ia-enron-only	143	623	0.061	4	3	4	0.350	-	7	5	5	4	4
c-fat200-1	200	1534	0.077	7	3	7	-	13.9	7	7	7	7	7
c-fat200-2	200	3235	0.163	5	3	5	21.7	-	5	5	5	5	5
c-fat200-5	200	8473	0.426	3	2	3	1.1	-	3	3	3	3	3
sphere3	258	1026	0.031	7	4	7	-	27.4	9	8	7	7	7
DD244	291	822	0.019	7	5	7	-	23.5	12	9	7	7	7
ca-netscience	379	914	0.013	<u>6</u>	5	7	-	46.1	11	8	7	7	7
infect-dublin	410	2765	0.033	5	4	5	28.2	-	8	5	5	5	5
c-fat500-1	500	4459	0.036	9	5	9	-	112.5	11	10	10	9	9
c-fat500-2	500	9139	0.073	7	4	7	-	120.3	8	7	7	7	7
c-fat500-5	500	23,191	0.186	5	3	5	297.3	-	5	5	5	5	5
bio-diseasome	516	1188	0.009	7	5	7	-	122.5	13	7	7	7	8
web-polblogs	643	2280	0.011	<u>5</u>	4	6	6.7	-	10	6	6	6	6
rt-twitter-copen	761	1029	0.004	7	5	7	-	805.7	13	7	7	7	7
DD68	775	2093	0.007	<u>9</u>	6	10	-	491.8	16	11	10	10	10
ia-crime-moreno	829	1475	0.004	7	4	7	-	1800	10	7	7	7	7
DD199	841	1902	0.005	<u>12</u>	8	14	-	672.5	21	16	14	14	14
soc-wiki-Vote	889	2914	0.007	6	5	6	21.6	-	11	6	6	6	6
DD497	903	2453	0.006	<u>10</u>	7	12	-	836.7	18	14	11	11	12
socfb-Reed98	962	18,812	0.041	4	3	4	6.7	-	7	4	4	4	4
lattice3D	1000	2700	0.005	10	6	10	-	150,000	14	11	10	10	11
bal_bin_tree_9	1023	1022	0.002	10	7	10	-	381.9	18	10	11	10	10
delaunay_n10	1024	3056	0.006	9	5	9	-	1100	13	10	10	9	10

Table 3. *Cont.*

Graphs				Bounds			t_p		BFF		Heuristics		
Name	$\|V\|$	$\|E\|$	ρ	$b(G)$	L	U	**CSP1 + BS**	**CSP2 + BS**	**Worst**	**Best**	**ICCH**	**BBGH**	**CBRH**
stufe	1036	1868	0.003	12	7	12	-	1300	18	14	13	12	13
lattice2D	1089	2112	0.004	13	8	14	-	2800	20	16	15	14	14
bal_ter_tree_6	1093	1092	0.002	7	5	7	-	260.1	13	7	9	7	7
email-univ	1133	5451	0.009	5	4	5	208.7	-	9	5	6	6	6
econ-mahindas	1258	7513	0.010	5	4	5	128.1	-	8	5	5	5	5
ia-fb-messages	1266	6451	0.008	5	4	5	55.2	-	9	5	5	5	5
bio-yeast	1458	1948	0.002	9	7	9	-	12,000	17	9	9	9	9
tech-routers-rf	2113	6632	0.003	6	5	7	252.1	-	12	7	7	7	7
chameleon	2277	31,421	0.012	6	5	6	8.2	-	11	6	6	6	6
TVshow	3892	17,262	0.002	?	7	10	-	-	18	10	10	10	10
ego-facebook	4039	88,234	0.011	4	3	4	1800	-	7	5	4	4	4
squirrel	5201	198,493	0.015	6	4	6	170,000	-	10	6	6	6	6
politician	5908	41,729	0.002	7	5	7	21,000	-	13	7	7	7	7

Table 4. Optimal solutions found by CSP1 + BS and CSP2 + BS over some square grid graphs. The feasible solutions found by some state-of-the-art algorithms are reported too. Optimal solutions reported for the first time are underlined.

$n \times n$ Grids				Bounds			t_p		BFF		Heuristics		
n	$\|V\|$	$\|E\|$	ρ	$b(G)$	L	U	**CSP1 + BS**	**CSP2 + BS**	**Worst**	**Best**	**ICCH**	**BBGH**	**CBRH**
3	9	12	0.333	3	2	3	0.001	-	4	3	3	3	3
4	16	24	0.200	4	2	4	0.031	-	4	4	4	4	4
5	25	40	0.133	4	3	4	0.047	-	5	4	4	4	5
6	36	60	0.095	5	3	5	0.235	-	6	5	5	5	5
7	49	84	0.071	5	3	5	0.539	-	7	5	5	5	6
8	64	112	0.056	6	3	6	9.1	-	7	6	6	6	6
9	81	144	0.044	6	4	6	11.3	-	8	6	6	6	6
10	100	180	0.036	6	4	7	-	2.5	9	7	7	7	7
11	121	220	0.030	7	4	7	-	5.2	10	7	7	7	7
12	144	264	0.026	7	4	7	-	7.2	10	8	8	7	8
13	169	312	0.022	7	5	8	-	7.5	11	8	8	8	8
14	196	364	0.019	8	5	8	-	14.5	11	9	9	8	8
15	225	420	0.017	8	5	9	-	15.4	12	9	9	9	9
16	256	480	0.015	8	5	9	-	17.2	12	10	9	9	9
17	289	544	0.013	9	5	9	-	35.7	12	10	10	9	10
18	324	612	0.012	9	5	9	-	46.2	13	11	10	9	10
19	361	684	0.011	9	5	10	-	65.5	13	11	10	10	10
20	400	760	0.010	10	6	10	-	108.0	14	11	11	10	11
21	441	840	0.009	10	6	10	-	119.1	14	12	11	10	11
22	484	924	0.008	10	6	11	-	200.1	15	12	11	11	11
23	529	1012	0.007	11	6	11	-	234.3	15	12	12	11	11
24	576	1104	0.007	11	6	11	-	332.9	16	13	12	11	12
25	625	1200	0.006	11	6	12	-	432.8	16	13	12	12	12
26	676	1300	0.006	11	7	12	-	807.4	17	13	13	12	12
27	729	1404	0.005	12	7	12	-	819.4	17	14	13	12	12
28	784	1512	0.005	12	7	12	-	746.3	18	14	13	12	12
29	841	1624	0.005	12	7	13	-	1800	18	14	13	13	13
30	900	1740	0.004	12	7	13	-	7500	19	15	14	13	13
31	961	1860	0.004	13	7	13	-	1600	19	15	14	13	13
32	1024	1984	0.004	13	8	13	-	1500	20	15	14	13	13
33	1089	2112	0.004	13	8	14	-	3000	20	16	15	14	14
34	1156	2244	0.003	13	8	14	-	6600	20	16	15	14	14
35	1225	2380	0.003	14	8	14	-	6800	21	16	15	14	15

6. Discussion

This paper introduces three novel mathematical formulations for the GBP: an ILP and two CSPs (CSP1 and CSP2). Since CSP1 and CSP2 require the burning number in advance, they are integrated into a binary search procedure (CSP1 + BS and CSP2 + BS); this way, the issue of not knowing the burning number in advance is lessened. All these formulations can be solved over arbitrary graphs thanks to off-the-shelf optimization software.

Section 4 shows a series of experiments that validate the correct implementation of the proposed formulations. This same section presents an empirical performance comparison among them using random graphs. From these experiments, we observe that CSP2 + BS tends to be better suited for graphs with a relatively large burning number (we empirically estimated this value as seven or more.) From these same experiments, we observe that CSP1 + BS seems to be a better choice for solving graphs with a relatively small burning number (we empirically estimated this value as six or less.) Of course, these observations cannot be generalized. Thus, rigorous statistical analysis should be performed in the future.

In Section 5, we used CSP1 + BS and CSP2 + BS to compute the optimal solution for some benchmark connected graphs of order at most 5908 (it is worth mentioning that, in all experiments, the BNC holds.) From these, we found seven previously unknown optimal solutions. We could not apply CSP2 + BS over graphs of order greater than 1458 because the memory requirements grew beyond our hardware's capacity. Regarding CSP1 + BS, it solved graphs with a relatively small burning number and order at most 5908. The obtained set of optimal solutions can be helpful as a benchmark dataset for comparing non-exact algorithms for the GBP, i.e., approximation algorithms, heuristics, and metaheuristics.

Finally, as part of the reviewing process, a reviewer pointed to the possibility of reducing the number of variables and constraints from CSP2 by removing variables x_i and $y_{i,j}$. However, we believe such variables cannot be removed because they let us guarantee that the solution has length B and that all vertices are burned. Nevertheless, we agree that mathematical formulations with fewer variables and constraints might exist. Thus, we will seek alternative mathematical formulations of the problem for future work.

Author Contributions: Conceptualization, J.G.-D., L.M.X.R.-H., J.C.P.-S. and S.E.P.-H.; methodology, J.G.-D., L.M.X.R.-H., J.C.P.-S. and S.E.P.-H.; software, J.G.-D.; validation, J.G.-D., L.M.X.R.-H., J.C.P.-S. and S.E.P.-H.; formal analysis, J.G.-D., L.M.X.R.-H., J.C.P.-S. and S.E.P.-H.; investigation, J.G.-D., L.M.X.R.-H., J.C.P.-S. and S.E.P.-H.; data curation, J.G.-D.; writing—original draft preparation, J.G.-D.; writing—review and editing, J.G.-D., L.M.X.R.-H., J.C.P.-S., and S.E.P.-H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this paper are openly available in https://github.com/jesgadiaz/GBP (acccessed on 31 July 2022).

Acknowledgments: We thankfully acknowledge Gurobi for providing a free-of-charge academic license for Gurobi version 9.5.1. We are grateful to anonymous reviewers whose questions, comments, and suggestions helped improve an earlier version of this paper. We also acknowledge Consejo Nacional de Ciencia y Tecnología (CONACYT) and Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) for providing the necessary resources for the development of this research. The authors thankfully acknowledge the computer resources, technical expertise and support provided by the Laboratorio Nacional de Supercómputo del Sureste de México, CONACYT member of the network of national laboratories.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BBGH	Backbone based greedy heuristic
BFF	Burning farthest-first
BNC	Burning number conjecture
BS	Binary search
CBRH	Component based recursive heuristic
CSP	Constraint satisfaction problem
FP	Firefighter problem
GBP	Graph burning problem
ICCH	Improved cutting corners heuristic
ILP	Integer linear program
VKCP	Vertex *k*-center problem

References

1. Bonato, A.; Janssen, J.; Roshanbin, E. Burning a graph as a model of social contagion. In *International Workshop on Algorithms and Models for the Web-Graph*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 13–22.
2. Bessy, S.; Bonato, A.; Janssen, J.; Rautenbach, D.; Roshanbin, E. Burning a graph is hard. *Discret. Appl. Math.* **2017**, *232*, 73–87. [CrossRef]
3. Diestel, R. *Graph Theory*; Springer Graduate Texts in Mathematics; Springer: Berlin/Heidelberg, Germany, 2017.
4. Šimon, M.; Huraj, L.; Luptáková, I.D.; Pospíchal, J. Heuristics for spreading alarm throughout a network. *Appl. Sci.* **2019**, *9*, 3269. [CrossRef]
5. Gautam, R.H.; Kare, A.S.; Surampudi, D.B. Faster heuristics for graph burning. *Appl. Intell.* **2021**, *52*, 1351–1361. [CrossRef] [PubMed]
6. Gupta, A.T.; Lokhande, S.A.; Mondal, K. Burning grids and intervals. In Proceedings of the Algorithms and Discrete Applied Mathematics, Rupnagar, India, 11–13 February 2021; pp. 66–79.
7. Bonato, A.; Kamali, S. Approximation algorithms for graph burning. In Proceedings of the International Conference on Theory and Applications of Models of Computation, Kitakyushu, Japan, 13–16 April 2019; pp. 74–92.
8. García-Díaz, J.; Pérez-Sansalvador, J.C.; Rodríguez-Henríquez, L.M.X.; Cornejo-Acosta, J.A. Burning Graphs Through Farthest-First Traversal. *IEEE Access* **2022**, *10*, 30395–30404. [CrossRef]
9. Vazirani, V.V. *Approximation Algorithms*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
10. Bonato, A.; English, S.; Kay, B.; Moghbel, D. Improved bounds for burning fence graphs. *Graphs Comb.* **2021**, *37*, 2761–2773. [CrossRef]
11. Liu, H.; Zhang, R.; Hu, X. Burning number of theta graphs. *Appl. Math. Comput.* **2019** *361*, 246–257. [CrossRef]
12. Bonato, A.; Lidbetter, T. Bounds on the burning numbers of spiders and path-forests. *Theor. Comput. Sci.* **2019**, *794*, 12–19. [CrossRef]
13. Liu, H.; Hu, X.; Hu, X. Burning number of caterpillars. *Discret. Appl. Math.* **2020**, *284*, 332–340.
14. Mitsche, D.; Prałat, P.; Roshanbin, E. Burning number of graph products. *Theor. Comput. Sci.* **2018**, *746*, 124–135. [CrossRef]
15. Sim, K.A.; Tan, T.S.; Wong, K.B. On the burning number of generalized petersen graphs. *Bull. Malaysian Math. Sci.* **2018**, *41*, 1657–1670. [CrossRef]
16. Bastide, P.; Bonamy, M.; Bonato, A.; Charbit, P.; Kamali, S.; Pierron, T.; Rabie, M. Improved Pyrotechnics: Closer to the Burning Number Conjecture. Preprint. 2022. Available online: https://arxiv.org/abs/2110.10530 (accessed on 31 July 2022).
17. Gonzalez, T.F. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.* **1985**, *38*, 293–306. [CrossRef]
18. Hochbaum, D.S.; Shmoys, D.B. A best possible heuristic for the k-center problem. *Math. Oper. Res.* **1985**, *10*, 180–184. [CrossRef]
19. Garcia-Diaz, J.; Menchaca-Mendez, R.; Menchaca-Mendez, R.; Hernández, S.P.; Pérez-Sansalvador, J.C.; Lakouari, N. Approximation Algorithms for the Vertex K-Center Problem: Survey and Experimental Evaluation. *IEEE Access* **2019**, *7*, 109228–109245. [CrossRef]
20. Cornejo Acosta, J.A.; García Díaz, J.; Menchaca-Méndez, R.; Menchaca-Méndez, R. Solving the Capacitated Vertex K-Center Problem through the Minimum Capacitated Dominating Set Problem. *Mathematics* **2020**, 8, 1551. [CrossRef]
21. Grandoni, F. A note on the complexity of minimum dominating set. *J. Discret. Algorithms* **2006**, *4*, 209–214. [CrossRef]
22. Hernández-Gómez, J.C.; Reyna-Hérnandez, G.; Romero-Valencia, J.; Rosario Cayetano, O. Transitivity on Minimum Dominating Sets of Paths and Cycles. *Symmetry* **2020**, *12*, 2053. [CrossRef]
23. Hartnell, B. Firefighter! An application of domination. Presentation. In Proceedings of the 25th Manitoba Conference on Combinatorial Mathematics and Computing, Winnipeg, MB, Canada, 29 September–1 October 1995.
24. Develin, M.; Hartke, S.G. Fire containment in grids of dimension three and higher. *Discrete Appl. Math.* **2007**, *155*, 2257–2268. [CrossRef]

25. García-Martínez, C.; Blum, C.; Rodriguez, F.J.; Lozano, M. The firefighter problem: Empirical results on random graphs. *Comput. Oper. Res.* **2015**, *60*, 55–66. [CrossRef]
26. Kozen, D.C. Depth-First and Breadth-First Search. In *The Design and Analysis of Algorithms*; Texts and Monographs in Computer Science; Springer: New York, NY, USA, 1992. [CrossRef]
27. Gurobi Optimization. Gurobi Optimizer Reference Manual. 2022. Available online: http://www.gurobi.com/documentation/9.5/refman (accessed on 31 July 2022).
28. Erdős, P.; Rényi, A. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.* **1960**, *5*, 17–60.
29. Rossi, R.A.; Ahmed, N.K. The Network Data Repository with Interactive Graph Analytics and Visualization. AAAI. 2015. Available online: http://networkrepository.com (accessed on 31 July 2022).
30. Leskovec, J.; Krevl, A. Snap Datasets: Stanford Large Network Dataset Collection. 2014. Available online: http://snap.stanford.edu/data (accessed on 31 July 2022).