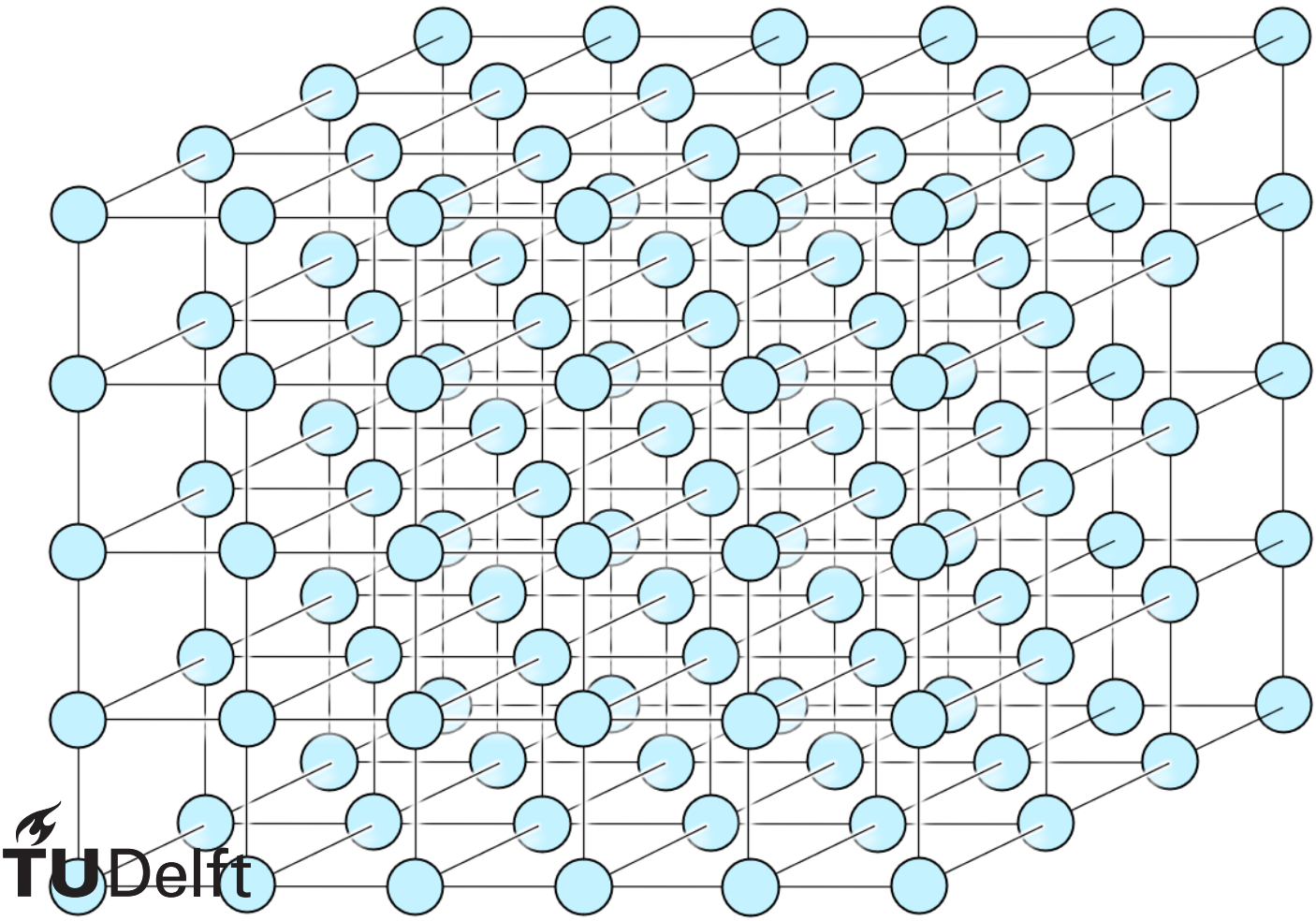
Graph Burning

Bounds on the burning number of higher-dimensional grid and king's graphs.

A.H. Grootendorst



TU Delft

Graph Burning

Bounds on the burning number of higher-dimensional grid and king's graphs.

by

A.H. Grootendorst

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Tuesday June 23, 2026 at 15:30.

Student number:	5538831	
Project duration:	April 20, 2026 – June 23, 2026	
Thesis committee:	Dr. Y. Murakami,	TU Delft, supervisor
	Dr. N. D. Verhulst,	TU Delft, supervisor
	Dr. A. Geyer,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Summary

Graph burning models the transmission of viruses and information. Graph burning is an iterative process in which vertices go from unburned to burned. In every round, first, all neighbors of burned vertices are also burned, second, one unburned vertex is selected as a source and also burned. These rounds are repeated until every vertex is burned. The burning number of a graph is the minimum number of rounds needed to burn every vertex in that graph.
Finding the burning number for a general graph is an NP-complete problem. Therefore, computing the burning number for relatively big graphs takes a long time. One way to reduce this time is by finding lower and upper bounds for the burning number that are close to the actual burning number.
In this thesis, we find lower and upper bounds for the burning number of Cartesian and strong products of multiple path graphs. These graphs are also known as higher-dimensional grid and king's graphs. The technique used to find these bounds is based on the technique Mitsche et al. used to asymptotically determine the burning number of grid graphs and king's graphs. A grid graph is the Cartesian product of two path graphs and a king's graph is the strong product of two path graphs.

For the burning number of higher-dimensional grid graphs, we found an implicit lower bound. The lower bound for the burning number of an i-dimensional grid graph can be calculated using the lower bound for the burning number of the $(i-1)$-dimensional grid graph.

We found that the burning number of the strong product of i paths of length n is larger than $\frac{1}{2}n^{\frac{i}{i+1}}$ and smaller than $\lceil \frac{1}{2}(n+1) \rceil$. For the strong product of i paths of lengths $m_1, m_2, \ldots, m_i$, we proved the burning number is larger than $\frac{1}{2}\prod_{j=1}^{i} m_j^{\frac{1}{i+1}}$.

Layman's summary

Graph burning models the transmission of viruses and information. A network, for example Instagram, is modeled as a graph that consists of vertices, which are the users in this case, and edges. There is an edge between two vertices if the two corresponding users are friends and therefore follow each other. When shown a post, a user shows it to all his friends the next day. Every day one user that has not seen the post, will be shown the post, independent of his friends having seen the post. This process is repeated until all users have seen the post. The minimum number of days it can take for everyone to see the post is called the burning number. The goal is to minimize the burning number. In this thesis we look at grid-like graphs, that are grids in multiple dimensions. The structure of these graphs helps to find the lower and upper bound for the burning number, values the burning number will always be in between. Having an upper and lower bound close to each other, makes calculating the actual burning number faster.

Contents

1 Introduction

In recent years, the dangers of misinformation have become increasingly clear. Through social media misinformation spreads faster than ever, often outpacing accurate information [11]. One way to model the spread of (mis)information is through graph burning [7]. Here, the network to be analyzed is represented in a graph in which users become vertices. If two users are connected (for example, by following each other) there is an edge between their corresponding vertices. In graph burning, vertices are either burned or unburned. When they are burned they cannot become unburned. The burning is as follows: In each round there are two steps. First, all unburned vertices connected (by an edge) to a burned vertex are also burned. Second, a new source is assigned. This source can be any vertex that was previously unburned but is now burned. This process of burning neighbors and then assigning a source is repeated until all vertices are burned. The goal of graph burning is to burn the graph in as few rounds as possible. This minimum number of rounds is called the burning number of the graph.

There are two main ways to determine the burning number: analytically using proofs and through optimization techniques such as integer linear programming (ILP).

This thesis focuses on the burning of Cartesian and strong products of path and cycle graphs. The Cartesian product of paths and cycles creates a grid structure in the graph. The strong product of paths and cycles creates a grid structure with added diagonals. The strong product of two path graphs is called a king's graph. The regularity in these graphs, as the neighborhoods for most pairs of vertices have a similar shape, is used to determine upper and lower bounds on the burning number. Combining these analytically found bounds with an ILP makes finding the exact burning number faster. (And this can potentially increase the size of the graphs which can be analyzed using an ILP.)

While the burning numbers of Cartesian and strong products of two paths (grid and king's graphs) have been studied and determined asymptotically [13, 14], the burning number of Cartesian and strong products of at least three paths has not been determined. Mitsche et al. showed limits on the burning number for the product of any two graphs [14]. This provides an upper bound for the burning number of graph products, but by utilizing the structure of the graphs studied in this thesis, the upper bound can be improved to a lower value.

The main objective of this thesis is to investigate whether we can find upper and lower bounds for the burning number of the Cartesian and strong product of multiple path graphs, using a technique similar to that of Mitsche. We have found an implicit lower bound for the Cartesian product of multiple path graphs. In addition, we determined an explicit lower and upper bound for the burning number of the strong product of multiple path graphs. For the strong product of three short paths of equal length (up to 17), the upper bound we found equals the actual burning number, and the lower bound is at most 1 below the actual burning number.

In Chapter 2, we introduce the basic principles and ideas of graph burning, and we list previous results. In Chapter 3, we explore the burning number of Cartesian products of multiple paths and cycles. First, we explain the idea of a proof for the asymptotic bound on the burning number of a grid. Then, we illustrate the impact of having a close but accurate lower and upper bound in calculating the exact burning number using

an ILP. Lastly, we introduce an implicit lower bound on the burning number of the Cartesian product of i path graphs. In Chapter 4, we explore burning strong products of multiple path graphs. For the strong product of three path graphs, one lower and two upper bounds for the burning number are found and compared for graphs of different sizes.

2

Definitions and previous results

2.1. Graph definitions

A *graph* $G = (V, E)$ is a collection of a set of *vertices* V and a set of *edges* E. For $u, v \in V$, if $uv \in E$, then u and v are *neighbors*. The *degree* of vertex v is the number of edges connected to that vertex. The number of vertices in the graph is $|V|$. This is called the *order* of the graph.

A *walk* $(x_1, x_2, \ldots, x_p)$ from x_1 to x_p is an ordered set of vertices where $x_i x_{i+1} \in E$ for $i = 1, \ldots, p-1$. A *path* is a walk where $x_i \neq x_j$ if $i \neq j$. We call the graph P_n a path graph if $|V(P_n)| = n$ and the vertices of P_n can be ordered to form a path $(x_1, x_2, \ldots, x_n)$. A *cycle* is a walk where $x_1 = x_p$ and $(x_1, x_2, \ldots, x_{p-1})$ is a path. We call graph C_n a cycle graph if $|V(C_n)| = n$ and the vertices of C_n can be ordered to form a cycle $(x_1, x_2, \ldots, x_n, x_1)$. In a *connected* graph, there exists a walk between any two vertices.

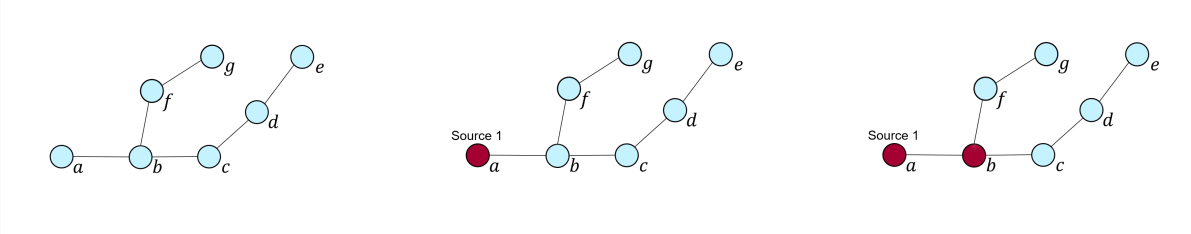
The *distance* between $u \in V$ and $v \in V$ is $d(u, v) = \min\{n : (u = x_1, x_2, \ldots, x_n = v) \text{ is a path}\}$. Note that $d(u, u) = 0$. The *neighborhood* of v is the set containing all neighbors of v, including v: $N[v] = \{u \in V : d(u, v) \leq 1\}$. The *k-neighborhood* $N_k[v]$ of v is the set containing all vertices $u \in V$ where $d(u, v) \leq k$. We write $N_k[v] = \{u \in V : d(u, v) \leq k\}$. In particular $N[v] = N_1[v]$. We call k the *radius* of $N_k[v]$.

A *subgraph* H of G ($H \subseteq G$) is a graph $H = (W, F)$ where $W \subseteq V$ and $F \subseteq E$. If $xy \in F$, then $x, y \in W$. Graphs G and H are *isomorphic* if there exists a projection $f(v)$ from $V(G)$ to $V(H)$ such that for every $v \in V(G)$ there is exactly one $f(v) \in V(H)$ and there is an edge $uv \in E(G)$ if and only if there is an edge $f(u)f(v) \in E(H)$. A subgraph H of graph G is an *induced subgraph* if for every $u, v \in V(H)$, there is an edge $uv \in E(H)$ if and only if there is an edge $uv \in E(G)$.
A vertex is sometimes also referred to as a *node*.

2.2. Graph burning

Graph burning was introduced by Bonato in 2016 as a way to model the transmission of diseases and information [7]. Earlier, in 1989, Noga Alon worked on a similar concept of transmitting in the n-dimensional cube, as a way to describe the processes inside certain machines [1]. Both processes describe a similar procedure, simply under different names. Graph burning is an iterative process that repeats until all vertices change from their starting state, *unburned*, to their final state, *burned*. This is done in rounds. In every round, there are two steps. First, all neighbors of already burned vertices are also burned. Second, a new *source* is chosen. Any unburned vertex can be chosen as a source. If there are no unburned vertices left at step k, the process is finished. The ordered set of vertices that are used as sources for the rounds is called a *burning sequence*. A burning sequence is not always unique. There can be multiple sequences of vertices that burn the graph in the same number of rounds.

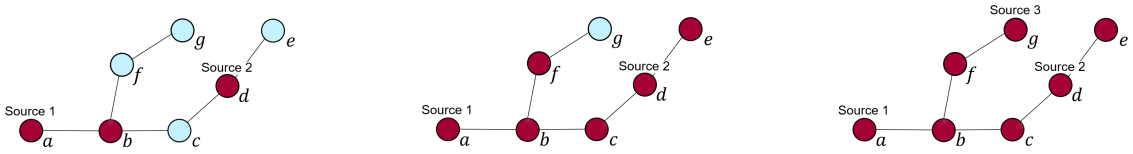
The goal of graph burning is to minimize the number of rounds in which all vertices can be burned. The *burning number* $b(G)$ of graph G is the minimum number of rounds needed to burn all vertices of graph G using the burning process. In other words, the burning number of graph G is the length of the shortest burning sequence for graph G. In figure 2.1, (a, d, g) is an example of a burning sequence for graph G. Because the burning sequence has length 3, we know $b(G) \leq 3$.



(a) Graph G containing only unburned vertices.

(b) Round 1, step 2: Selecting a source (a).

(c) Round 2, step 1: Burning all unburned neighbors of previously burned vertices (b).



(d) Round 2, step 2: Selecting a source (d).

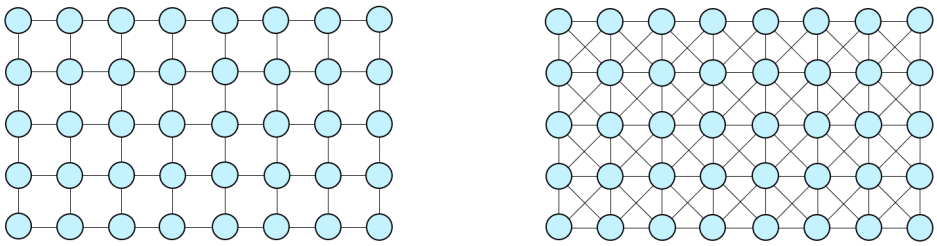
(e) Round 3, step 1: Burning all unburned neighbors of previously burned vertices (c,e, and f).

(f) Round 3, step 2: Selecting a source (g). Every vertex is now burned. The process is finished.

Figure 2.1: Burning a graph in 3 rounds. The burning number of this graph is at most 3.

2.3. Graph products

The *Cartesian product,* $H = G_1 \square G_2$, of graphs G_1 and G_2 is defined in the following way: $V(H) = \{(x,y) : x \in V(G_1), y \in V(G_2)\}$. There is an edge between $a = (x_1, y_1)$ and $b = (x_2, y_2)$ if $x_1 = x_2$ and $y_1 y_2 \in E(G_2)$, or if $y_1 = y_2$ and $x_1 x_2 \in E(G_1)$. We use $\square_{j=1}^{i} G_j$ as another way of writing $G_1 \square G_2 \square \cdots \square G_i$.



(a) Graph $G_1 = P_5 \square P_8$ is a 5×8 grid graph.

(b) Graph $G_2 = P_5 \boxtimes P_8$ is a 5×8 king's graph.

Figure 2.2: Examples of a grid graph and a king's graph.

An $m \times n$ *grid graph* is the Cartesian product of two path graphs, P_m and P_n. Figure 2.2a is an example of a 5×8 grid graph. We call $\square_{j=1}^{i} P_{m_j}$ an *i-dimensional grid graph.*

The *strong product* of graphs G_1 and G_2, $H = G_1 \boxtimes G_2$, is $H = (V, E)$ where if $u \in V(G_1)$ and $v \in V(G_2)$, then $(u, v) \in V(H)$. Also if $u \in V(G_1)$ and $v_1 v_2 \in E(G_2)$, then $(u, v_1)(u, v_2) \in E(H)$. Similarly if $v \in V(G_2)$ and $u_1 u_2 \in E(G_1)$, then $(u_1, v), (u_2, v) \in E(H)$. Lastly, if $u_1 u_2 \in E(G_1)$ and $v_1 v_2 \in G_2$, then $(u_1, v_1), (u_2, v_2) \in E(H)$ and $(u_1, v_2), (u_2, v_2) \in E(H)$ (creates the diagonals). We use $\boxtimes_{j=1}^{i} G_j$ as another way of writing $G_1 \boxtimes G_2 \boxtimes \cdots \boxtimes G_i$.

The *king's graph* is similar to the grid graph. The grid and king's graphs can be thought of as a chessboard. The vertices are the squares in which a piece can stand. In a king's graph, pieces can move one square diagonally and orthogonally, like the king in chess. Similar to a grid, an $m \times n$ king's graph is drawn with its vertices arranged in an $m \times n$ rectangle. In addition to the edges in a grid pattern, it has edges that go across

diagonally. The graph in Figure 2.2b is a 5×8 king's graph.

The $m \times n$ *king's graph* is defined as the strong product $P_m \boxtimes P_n$ of P_m and P_n (path graphs). We call $\boxtimes_{j=1}^{i} P_{m_j}$ an *i-dimensional king's graph.*

2.4. Burning number

There are multiple ways to determine whether a sequence of vertices is a burning sequence or not. Bonato et al. gave Lemma 1, showing the relation between the neighborhoods of a sequence and the burning number of the corresponding graph [7]. This lemma will be used to show a lower bound for the burning number of i-dimensional grid graphs.

Lemma 1 ([7]). *If $(x_1, x_2, \ldots, x_k)$ is a sequence of vertices in a graph G, such that $N_{k-1}[x_1] \cup N_{k-2}[x_2] \cup \cdots \cup N_0[x_k] = V(G)$, then $b(G) \leq k$.*

With the introduction of the burning number, Bonato included the burning number for path graphs [7].

Theorem 1 ([7]). $b(P_n) = \lceil \sqrt{n} \rceil$

He conjectured that any connected graph on n vertices will have a burning number lower than or equal to the burning number of a path on n vertices.

Conjecture 1 ([7]). *Let $G = (V, E)$ be a connected graph with $|V| = n$. Then*

$$b(G) \leq \lceil \sqrt{n} \rceil.$$

This conjecture was proven to hold for sufficiently large connected graphs of minimum degree 3, by Bastide et al. [2]. Murakami proved the conjecture holds for trees without degree-2 vertices [15]. This was recently expanded to all trees of order n with at most $\lfloor \sqrt{n-1} \rfloor$ degree-2 vertices [16]. Norin et al. proved the conjecture holds asymptotically for all connected graphs [17]. Because it has not been proven for all graph types, the conjecture remains an open problem. The closest upper bound for the burning number of a connected graph G known thus far is $b(G) \leq \left\lceil \sqrt{\frac{4n}{3}} \right\rceil + 1$ [2].

2.5. Algorithms for graph burning

Several algorithms have been created to compute the burning number. Computing the burning number for general graphs is an NP-complete problem [5].
A problem is NP-complete if it is NP-hard and in the class NP. When the problem is in NP, it means that there is an algorithm that checks in polynomial time whether a given burning sequence is indeed a burning sequence. For this problem, being NP-hard means that this decision problem is at least as hard as any other decision problem in NP. In practice, NP-completeness means there exists no polynomial-time algorithm to find the burning number of a graph.

When restricted to trees of maximum degree three, the problem remains NP-complete [5]. The problem is also NP-complete when restricted to path-forests, caterpillars, and interval graphs [5, 9, 12].For very specific graph classes, such as paths , cographs, and split graphs, there exist polynomial-time algorithms [6, 10].

Several algorithms have been created to compute the exact burning number of a general graph. Examples include ILP-COV, ILP-CMCP, and PRYM [8, 18]. The time complexities of these algorithms depend not only on the size of the graph, but also on the upper and lower bound provided the user. The closer the input bounds are to the actual burning number, the faster the value is computed. In an ILP, the problem is transformed into variables and linear constraints, making the problem solvable using solver software such as Gurobi.

2.6. Upper and lower bounds

The burning number of a graph G depends not only on its order $|V(G)|$ but also on the arrangement and number of its edges. If a graph G has an edge from one specific vertex to every other vertex, the burning number must be 2 [19].

Theorem 2 ([19]). *A graph G with a vertex v of degree $|V(G)| - 1$ has $b(G) = 2$.*

Theorem 3 ([2]). *For any connected graph G with $|V(G)| = n$, if every vertex in G has degree at least 3, then*

$$b(G) \leq \lceil \sqrt{n} \rceil + 2.$$

The radius of graph G is $rad(G) = \min_r\{r = \max_u\{d(u,v)\} : v \in V(G)\}$.

Theorem 4 ([14])**.** *For connected graphs G and H:*

$$\max\{b(G), b(H)\} \leq b(G \boxtimes H) \leq b(G \square H) \leq \min\{b(G) + rad(H), b(H) + rad(G)\}$$

Mitsche et al. found the asymptotic value for the burning number of grid graphs and kings graphs [13, 14]. This was done by determining a lower bound for the burning number and showing the upper bound is of the same order. Theorem 5 gives the asymptotic burning number of a grid graph and Theorem 10 gives the asymptotic burning number of a king's graph. In this thesis, the techniques used to find lower and upper bounds for burning numbers for grid and king's graphs are at the basis of the techniques used to find lower and upper bounds for Cartesian and strong products of multiple path graphs .

3

Higher-dimensional grid graphs

In this chapter the burning number for the Cartesian product of multiple path graphs is approximated.

3.1. The burning number of grids

Continuing on Theorem 1, Mitsche et al. (2017) found an asymptotic value for the burning number of $m \times n$ grid graphs [13].

Theorem 5 ([13]). $b(P_m \square P_n) = \begin{cases} (1+o(1))\sqrt[3]{\frac{3}{2}mn} & \text{if } \sqrt{n} \le m \le n \\ \Theta(\sqrt{n}) & \text{if } m < \sqrt{n} \end{cases}$

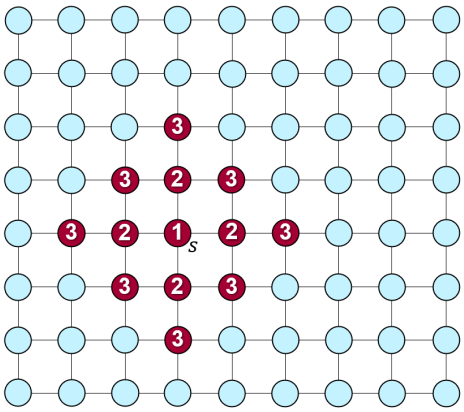


Figure 3.1: An 8×9 grid graph. The dark red vertex s is the source. The surrounding dark red vertices are burned by s after 3 rounds. The light blue vertices are unburned. The numbers inside the burned vertices denote the round in which they were burned.

In a grid graph, the k-neighborhood of a vertex looks like a diamond (a square rotated 45 degrees). A diamond with radius k around source s represents the $(k-1)$-neighborhood of s: $N_{k-1}[s]$. After r rounds of burning, the set of vertices burned by s is $N_{r-1}[s]$, which is shaped like a diamond of radius r. Figure 3.1 shows the pattern as a result of burning a single source in the grid graph. The red vertices are burned. Vertex s is the source. The light blue vertices are unburned. The burned vertices are in the 2-neighborhood of s. This forms a diamond of radius 3.

Lemma 1 says if for graph $G = P_m \square P_n$ there is a sequence of vertices $(s_r, s_{r-1}, \ldots, s_1)$ such that $N_{r-1}[s_r] \cup N_{r-2}[s_{r-1}] \cup \cdots \cup N_0[s_1] = V(G)$, then $b(G) \le r$. A completely burned grid graph would look like a collection of diamonds, one of each radius from 1 to r if r is the burning number. They would be arranged such that every

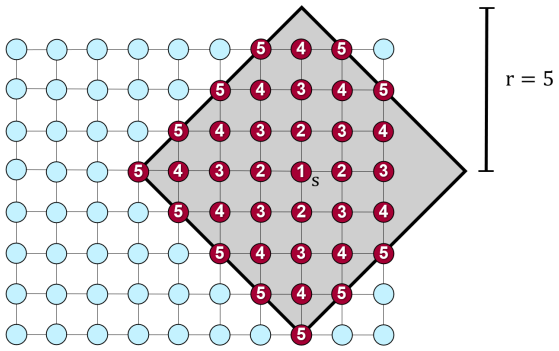


Figure 3.2: An 8×10 grid graph G. All dark red vertices are burned by vertex s in 5 rounds of burning. The numbers inside the burned vertices denote the round in which they were burned. The burned vertices are in a diamond of radius 5.

vertex is covered by at least one diamond. It is allowed for diamonds to overlap. If for some $i \neq j$, there is a vertex v in G for which $v \in N_{i-1}[s_i]$ and $v \in N_{j-1}[s_j]$, then we still have $N_{r-1}[s_r] \cup N_{r-2}[s_{r-1}] \cup \cdots \cup N_0[s_1] = V(G)$.

Theorem 5 was proved by finding a lower and upper bound for the grid burning number. We give an intuition for the proof.

The lower bound was determined by calculating the maximum number of vertices $v(r)$ that can be covered in r rounds. If $v(r)$ is lower than the number of vertices in the $m \times n$ grid graph $(m \times n)$,the burning number must be larger than r.

The upper bound was determined by applying a specific way to choose sources that guarantees burning all of them. This method does not always use the minimum number of rounds possible, but it does give an upper bound.

It can be shown that the graph can be covered using diamonds of radius between $k_1 = \frac{(mn)^{\frac{1}{3}}}{\gamma^{\frac{1}{6}}}$ and $k_2 = \left(\frac{3}{2}\right)^{\frac{1}{3}} (mn)^{\frac{1}{3}} \left(1 + \frac{1}{\gamma^{\frac{1}{6}}}\right)$, where $\gamma = \frac{m}{\sqrt{n}}$.

The diamonds are placed one by one, starting with the smallest (with radius k_1). The radius of a newly placed diamond is always 1 larger than the radius of the previously placed diamond. k_2 was chosen such that the radius of the largest (and last) diamond is smaller than k_2.

The diamonds are arranged in strips. Figure 3.3 shows an example of what strip S_{i+1} could look like. If the last (largest) diamond of strip S_i has radius r_i, then the first diamond in strip S_{i+1} has radius $r_i + 1$. The strip is defined to be as wide as its first diamond. In figure 3.3, the strip S_{i+1} is the shaded area. A new diamond in the strip is placed such that its topmost vertex is the rightmost vertex of the previous diamond. A strip is finished when placing another diamond in the strip would not make the strip cover any extra vertices.

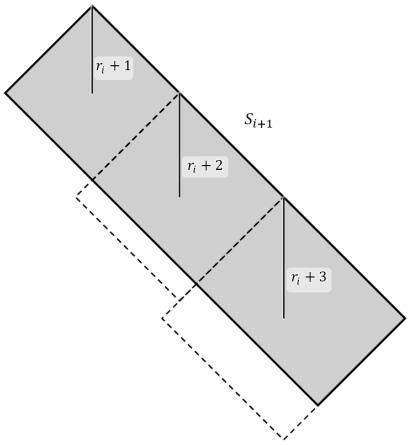


Figure 3.3: Strip S_{i+1} has the width of its smallest diamond.

The first strip is placed such that the center (source) vertex of the first diamond is the right top corner vertex of the grid.

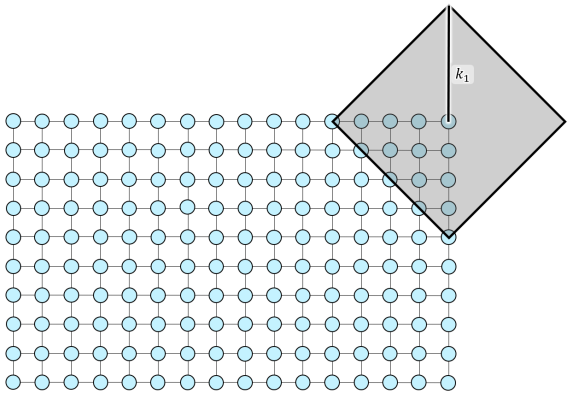


Figure 3.4: The first strip is placed on a 10×16 grid graph. The strip contains one diamond. $k_1 = 5$.

As long as not all vertices on the top row of the grid are covered, new strip S_{i+1} is placed left of the previous strip S_i. The topmost diamond of strip S_{i+1} is placed such that its rightmost vertex is the vertex left of the leftmost vertex of the first diamond of strip S_i.

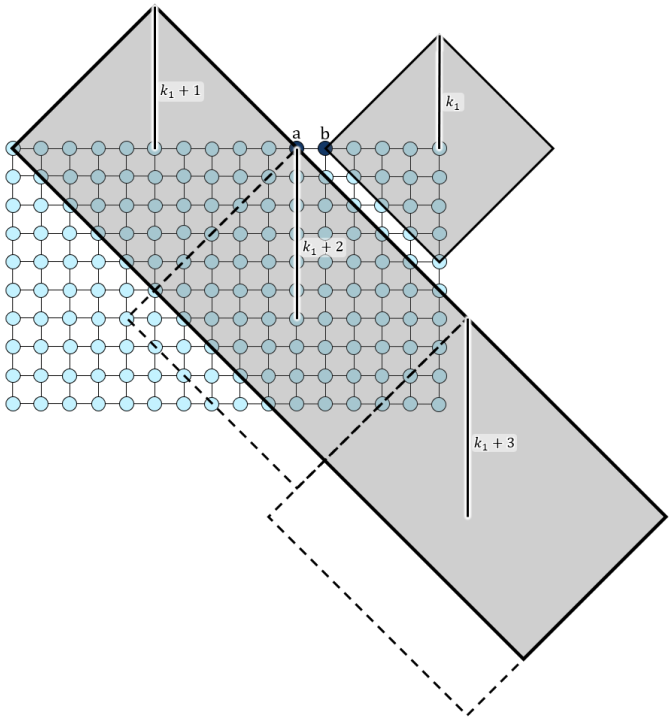


Figure 3.5: Strip S_2 is placed such that the rightmost vertex a of its first placed diamond is the left neighbor of the leftmost vertex b of the first diamond of strip S_1.

When all vertices of the top row of the grid are covered, new strips are placed below the previous strip. The topmost vertex of strip S_{i+1} is one vertex below the lowest covered vertex of the leftmost column of the grid.

When all vertices of the leftmost column of the grid are covered by the strips, it follows that all vertices of the grid are covered by at least one strip.

As can be seen in Figure 3.6, there are sections of the diamonds that are outside the grid and therefore do not cover any vertices. There are also vertices in the grid that are covered by more than one diamond. These areas together are called "wasted". Figure 3.7 shows the "wasted" areas in dark blue.

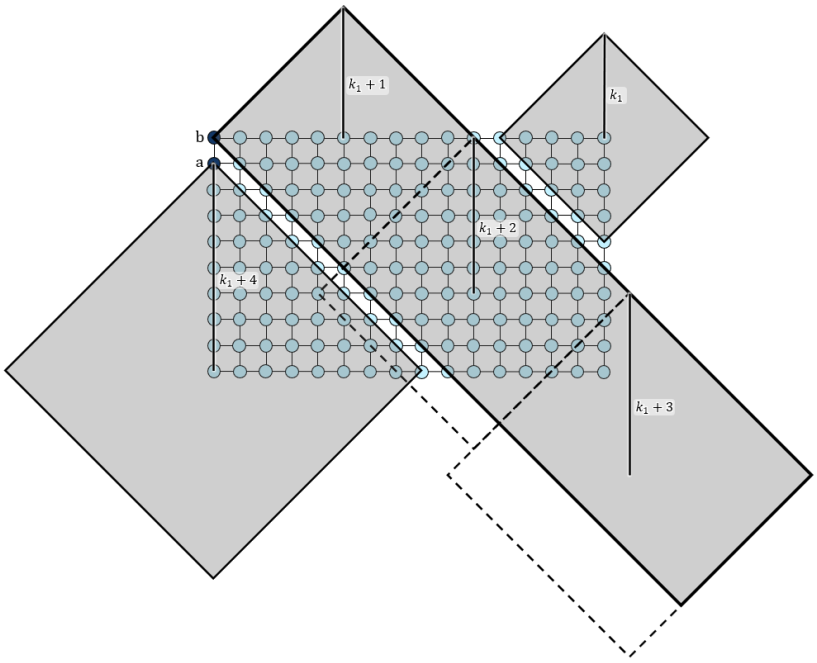


Figure 3.6: Strip S_3 is placed such that its topmost vertex a is the one below the lowest covered vertex b of the leftmost column of the grid.

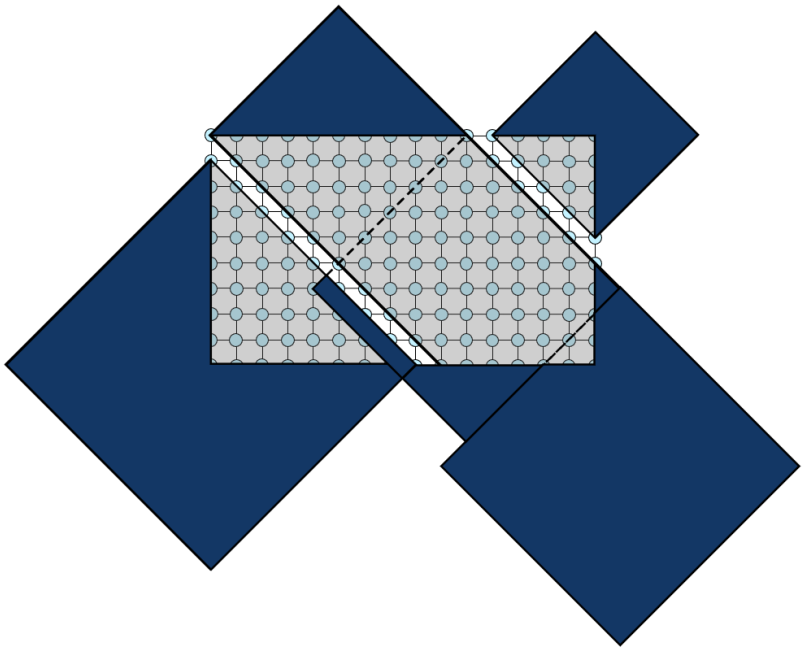


Figure 3.7: Every section of a diamond outside the grid and every section where diamonds overlap is wasted. Wasted areas are colored dark blue.

Say X is the number of vertices burned by the diamonds if there was no waste. Now let Y be the number of actually wasted vertices. The number of actually burned vertices is then $Z = X - Y$. By overestimating Y, Mitsche was able to show that for any $m \times n$ grid G, $Z \geq V(G)$. Therefore, all vertices can be burned in k_2 rounds and $b(G) \leq k_2$.

3.2. Torus Grids

An $m \times n$ *torus grid* $T_{m,n}$ is defined as $C_m \square C_n$, where C_m and C_n are cycles. $T_{m,n}$ is very similar to the $m \times n$ grid graph. They have the same number of vertices:$\left|V(T_{m,n})\right| = mn$, and the $m \times n$ grid graph is a subgraph of $T_{m,n}$. The only difference is on the boundary of the grid. Where there are vertices of degree 2 (the corners) and vertices of degree 3 (the sides) in the grid graph, all vertices in the torus grid have degree 4 (like any vertex in the middle of the grid). The extra edges on the torus are such that the grid continues no matter where you are on the grid. Using a coordinate system to describe the vertices, say the left bottom vertex is $(1,1)$, the left top is $(1,m)$, and the right top is (n,m), the extra edges are: $(1,i)-(n,i)$ and $(j,1)-(j,m)$ for $i = 1,\ldots,m$ and for $j = 1,\ldots,n$. If $m = 2$ or $n = 2$, some of these extra edges already exist. These edges are not added again. Between every pair of vertices, there is at most one edge.

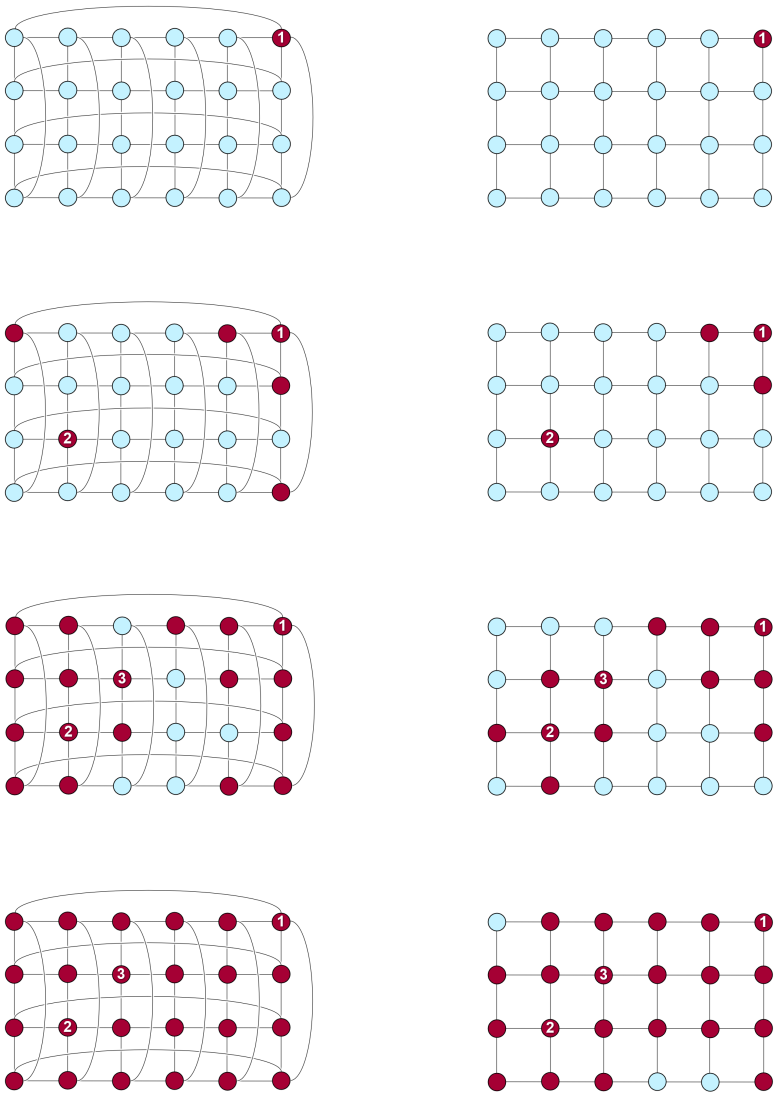


Figure 3.8: Comparison between burning a 4×6 torus grid graph in the left column, and a 4×6 grid graph in the right column, using the same sequence of vertices. The numbered vertices are sources. The number depicts the round in which the source was burned.

Because the general structure of a torus grid is the same as that of a 'normal' grid, the maximum number of vertices burned by one source after r rounds, and the total maximum number of vertices burned after r rounds is the same for the torus as it is the normal grid. Similarly, any burning sequence for the grid graph is also a burning sequence for the torus grid graph. On the contrary, not all burning sequences for the torus grid are burning sequences for the normal grid. Figure 3.8 gives an example of a sequence that is a burning

sequence for the torus grid but not for the normal grid. The torus contains extra edges that can propagate the burning in directions the normal grid cannot. Therefore, $b(C_m \square C_n) \leq b(P_m \square P_n)$.

Theorem 6 ([13]). $b(C_m \square C_n) = \begin{cases} (1+o(1))\sqrt[3]{\frac{3}{2}mn} & \text{if } \sqrt{n} \leq m \leq n \\ \Theta(\sqrt{n}) & \text{if } m < \sqrt{n} \end{cases}$

For smaller grid sizes (up to 20×20), the lower bound resulting from Theorem 5 is a better approximation for the torus grid burning number than for the grid burning number. When excluding $m = 1$, the lower bound is at most 1 lower than the actual burning number.

3.3. Determining the exact burning number

For small graphs, the burning number can be calculated by hand. Simply try every sequence of vertices that can form a burning sequence until the shortest one that works is found. For larger graphs, this takes too long. Here, integer linear programs provide help. The ILP consists of the input, variables, objective function, and constraints. The input is predetermined and cannot be changed by the program. The variables are unknown. Constraints are linear (in)equalities to ensure the combination of selected values for the variables is an actual solution. The objective function is used to determine which of the possible solutions is the best (minimal or maximal). To calculate the burning number of specific grid graphs, an implementation by Jesús García-Díaz, José Alejandro Cornejo-Acosta, and Joel Antonio Trejo-Sánchez was used [8]. It is implemented in Python, using Gurobi to find optimal solutions. Their ILP-COV and ILP-CMCP implementations change the original ILP to minimize the size of the input to help the solver find solutions more quickly. To get burning numbers for as many graphs as possible in a reasonable time, it helps to find lower and upper bounds as close to the actual burning number as possible. This avoids the program spending too much time searching through impossible options. At the same time, it is critical to make sure that the lower bound is actually smaller than or equal to the burning number. Given a lower bound lb that is larger than the actual burning number b, the program will find a burning sequence of length lb, and as this is the smallest it finds, it assumes the burning number is lb, whereas it is actually b.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	2	2	2	3	3	3	3	3	4	4	4	4	4	4	4	5	5	5	5
2		2	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	5	5	5
3			3	3	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6
4				4	4	4	4	5	5	5	5	5	5	5	6	6	6	6	6	6
5					4	4	5	5	5	5	5	5	6	6	6	6	6	6	6	7
6						5	5	5	5	5	6	6	6	6	6	6	6	7	7	7
7							5	5	6	6	6	6	6	6	6	7	7	7	7	7
8								6	6	6	6	6	6	7	7	7	7	7	7	7
9									6	6	6	7	7	7	7	7	7	8	8	
10										6	7	7	7	7	7	7	8	8	8	
11											7	7	7	7	8	8	8	8	8	
12												7	7	7	8	8	8	8	8	
13													7	8	8	8	8	8	9	
14														8	8	8	8	9	9	
15															8	8	8	9	9	
16																8	9	9	9	
17																	9	9	9	
18																		9	9	
19																			9	10
20																				10

(a) Burning numbers for $m \times n$ grid graphs up to 20×20.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	2	2	2	3	3	3	3	3	4	4	4	4	4	4	4	5	5	5	5
2		2	3	3	3	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5
3			3	3	3	4	4	4	4	4	5	5	5	5	5	5	5	6	6	6
4				3	4	4	4	4	4	5	5	5	5	5	5	5	6	6	6	6
5					4	4	4	5	5	5	5	5	5	6	6	6	6	6	6	6
6						4	5	5	5	5	5	5	6	6	6	6	6	6	6	6
7							5	5	5	5	6	6	6	6	6	6	7	7	7	7
8								5	5	6	6	6	6	6	6	7	7	7	7	7
9									6	6	6	6	6	6	7	7	7	7	7	7
10										6	6	6	6	7	7	7	7	7	7	8
11											6	7	7	7	7	7	7	7	8	
12												7	7	7	7	7	7	8	8	
13													7	7	7	7	8	8	8	8
14														7	7	8	8	8	8	
15															8	8	8	8	8	
16																8	8	8	8	
17																	8	8	9	9
18																		9	9	9
19																			9	9
20																				9

(b) Burning numbers for $m \times n$ torus grid graphs up to 20×20. If the number for $a \times b$ is green, $b(P_m \square P_n) > b(C_m \square C_n)$. If they are white, the burning numbers are the same.

Figure 3.9: Comparison between grid and torus grid burning numbers. $b(P_m \square P_n) \geq b(C_m \square C_n)$ for all m, n.

3.3.1. Lower bounds for grid graphs

The outcome of calculations using ILP-COV,ILP-CMCP, and other ILPs can vary depending on the lower and upper bounds given to the program.

The burning number of the 19×20 torus grid graph is 9. It can be seen in the first row of Table 3.1 that if an ILP is given a lower bound that is higher than the actual burning number, the ILP will not find the actual burning number. Instead, it will assume the lowest value for which it finds a burning sequence is the burning number. In this case, the given lower bound is 10, while the burning number is 9. The ILP wrongly gives 10 as

input		output	
Lower bound	Upper bound	Burning number found	Runtime (seconds)
10	11	10	1.07
9	10	9	0.88
9	9	9	0.80
8	9	9	10.24
7	9	9	10.18
6	9	9	11.97
5	9	9	12.79

Table 3.1: Comparing the calculation of the burning number of $G = C_{19} \square C_{20}$ using different lower and upper bounds as input.

the burning number. Even when the bounds are correct, their value can influence the time it takes to find a correct answer. Generally, the closer the lower and upper bounds are to the actual burning number, the faster the calculation will be.

To obtain a lower bound closer to the actual burning number, we can first use the fact that the burning number of an $m \times n$ grid graph G is smaller than or equal to the burning number of $a \times b$ grid graph H if m is smaller than a, and n is smaller than b.

Theorem 7. *Take two graphs $G = P_m \square P_n$ and $H = P_a \square P_b$, where $m \leq a$ and $n \leq b$. Then $b(G) \leq b(H)$.*

Proof. For $m \leq a$ and $n \leq b$ let $G = P_m \square P_n$ and $H = P_a \square P_b$. Then H has a subgraph which is isomorphic to G. Say $b(G) = g$ and $b(H) = h$. Then there exists a burning sequence $S = (v_1, ..., v_h)$ for graph H with $v_i \in V(H)$ for $i = 1, ..., h$. If also $v_i \in V(G)$ for $i = 1, ..., h$, then S burns all vertices of G in h rounds, so $b(G) \leq h = b(H)$. Now, if there are any vertices in sequence S outside of G, a new burning sequence T can be made, where G is burned in the order of S, but if we come across a source s in S that is not inside (sub)graph G, we replace it with the vertex in G that is closest to that source. (Minimizing the distance between the original and the new source.) Any vertex inside G that was burned by the original source will also be burned by the new source.

Let s be a source not in $V(G)$. Say it burns $N_k^G[s] = N_k[s] \cap V(G)$. Now let $c \in V(G)$ be the closest vertex to s. Then if $c \notin N_k^G[s]$,we have $d > k$. So $N_k^G[s] = \varnothing$. Then definitely $N_k^G[s] \subseteq N_k^G[c] \neq \varnothing$. Otherwise, if $c \in N_k[s]$,we have $d(c, s) \leq k$. So we need to show $N_k^G[s] \subseteq N_k^G[c]$.

Let $v \in N_k^G[s]$. For ease we give the vertices coordinates. $s = (x_s, y_s), c = (x_c, y_c), v = (x_v, y_v)$. Let the bottom left vertex have coordinates $(1, 1)$ and the top right (n, m). Without loss of generality, we can assume s is to the right and possibly above the grid (rotation and mirroring make all other positions equivalent). So $x_s > x_c, y_s \geq y_c$ and $x_s > x_v, y_s \geq y_v$. c is chosen such that $x_s - x_c$ is minimal and $y_s - y_c$is minimal. So for an $m \times n$ grid,$x_c = n$ and $y_c = \begin{cases} y_s, \text{ if } y_s \leq m \\ m, \text{ otherwise} \end{cases}$.

From the construction of c and v we know $x_v \leq n = x_c < x_s$. By definition $y_c \leq y_s$ So there are three options: either $y_c = y_s$ and $y_v \leq y_c$, or $y_c = y_s$ and $y_v > y_c$, or $y_s > m = y_c$ and $y_v \leq y_c$.

If $y_c = y_s$ and $y_v \leq y_c$ or $y_s > m = y_c$ and $y_v \leq y_c$, then $y_v \leq y_c \leq y_s$. So

$$\begin{aligned} d(v, c) + d(c, s) &= (x_c - x_v) + (y_c - y_v) + (x_s - x_c) + (y_s - y_c) \\ &= x_s - x_v + y_s - y_v \\ &= d(v, s) \leq k. \end{aligned}$$

So $d(v, c) \leq k - d(c, s) < k$. This means $v \in N_k^G[c]$

If $y_c = y_s$ and $y_v > y_c$ then

$$\begin{aligned} d(v, c) + d(c, s) &= (x_c - x_v) + (y_v - y_c) + (x_s - x_c) + (y_c - y_s) \\ &= x_s - x_v + y_v - y_s \\ &= d(v, s) \leq k. \end{aligned}$$

So $d(v, c) \leq k - d(c, s) < k$. This means $v \in N_k^G[c]$.

We can conclude $N_k^G[s] \subset N_k^G[c]$.

Thus all vertices inside G are burned by burning sequence T. Once again we have a burning sequence T for G. The length of T is equal to the length of S, so $b(G) \leq h = b(H)$. □

Using this, we can say that if the burning number for an $m \times n$ grid graph is known, the lower bound for an $a \times b$ grid graph where $a \geq m$ and $b \geq n$ is at least $b(P_m \square P_n)$.

A more direct lower bound is found in Theorem 5. For an $m \times n$ grid, we take the lower bound to be $\left\lceil \sqrt[3]{\frac{3}{2}mn} \right\rceil$. This does not require knowing the burning number for a smaller grid and is therefore useful for calculating the burning number of many grids of different sizes at the same time. Figure 3.10 shows the lower bounds that follow from Theorems 5 and 6 for $m \times n$ grid graphs and $m \times n$ torus grid graphs.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1,14	1,44	1,65	1,82	1,96	2,08	2,19	2,29	2,38	2,47	2,55	2,62	2,69	2,76	2,82	2,88	2,94	3	3,05	3,11
2		1,82	2,08	2,29	2,47	2,62	2,76	2,88	3	3,11	3,21	3,30	3,39	3,48	3,56	3,63	3,71	3,78	3,85	3,91
3			2,38	2,62	2,82	3	3,16	3,30	3,43	3,56	3,67	3,78	3,88	3,98	4,07	4,16	4,25	4,33	4,41	4,48
4				2,88	3,11	3,30	3,48	3,63	3,78	3,91	4,04	4,16	4,27	4,38	4,48	4,58	4,67	4,76	4,85	4,93
5					3,35	3,56	3,74	3,91	4,07	4,22	4,35	4,48	4,60	4,72	4,83	4,93	5,03	5,13	5,22	5,31
6						3,78	3,98	4,16	4,33	4,48	4,63	4,76	4,89	5,01	5,13	5,24	5,35	5,45	5,55	5,65
7							4,19	4,38	4,55	4,72	4,87	5,01	5,15	5,28	5,40	5,52	5,63	5,74	5,84	5,94
8								4,58	4,76	4,93	5,09	5,24	5,38	5,52	5,65	5,77	5,89	6	6,11	6,21
9									4,95	5,13	5,30	5,45	5,60	5,74	5,87	6	6,12	6,24	6,35	6,46
10										5,31	5,48	5,65	5,80	5,94	6,08	6,21	6,34	6,46	6,58	6,69
11											5,66	5,83	5,99	6,14	6,28	6,42	6,55	6,67	6,79	6,91
12												6	6,16	6,32	6,46	6,60	6,74	6,87	6,99	7,11
13													6,33	6,49	6,64	6,78	6,92	7,05	7,18	7,31
14														6,65	6,80	6,95	7,09	7,23	7,36	7,49
15															6,96	7,11	7,26	7,40	7,53	7,66
16																7,27	7,42	7,56	7,70	7,83
17																	7,57	7,71	7,85	7,99
18																		7,86	8,01	8,14
19																			8,15	8,29
20																				8,43

Figure 3.10: The lower bounds for the burning number of $m \times n$ (torus) grids based on Theorem 6: $b(P_m \square P_n) \geq b(C_m \square C_n) \geq \left(\frac{3}{2}\right)^{\frac{1}{3}} (mn)^{\frac{1}{3}}$. Rounded to 2 decimals.

3.4. Expanding the lower bound to higher dimensions

Figure 3.11, Figure 3.12, and Figure 3.13 are examples of graphs after some rounds of burning. The number inside each vertex depicts the round in which it was first burned. Vertices without numbers are unburned. Figure 3.11 is an example of a path after 3 rounds of burning.

We can see that in a path, at most $5+3+1=9$ vertices can be burned in 3 rounds. After one round, only 1

Figure 3.11: Path graph P_{12} after 3 rounds of burning.

vertex has been burned, and after 2 rounds, $3+1=4$ vertices have been burned. Figure 3.12 is an example of burning a grid graph.

The 6×12 grid can visually be cut horizontally into 6 slices that form paths of length 12. When counting from

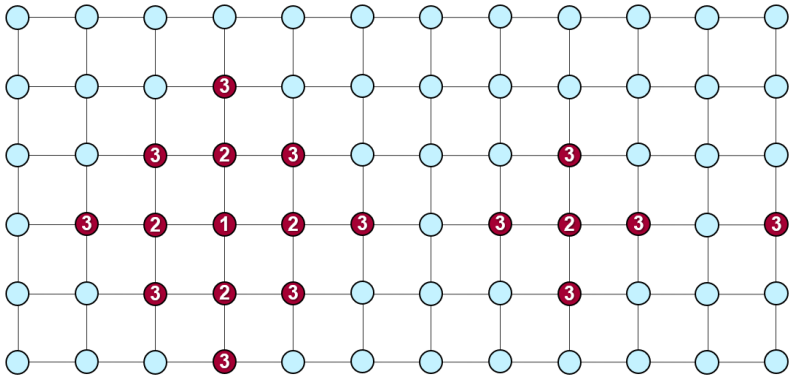


Figure 3.12: Grid graph $P_6\square P_{12}$ after 3 rounds of burning.

above, the first slice (top row) does not contain any burned vertices. The second and sixth slices both contain 1 burned vertex. This is the same number of vertices that can be burned in a path after 1 round. The third and fifth slices both contain $3+1=4$ burned vertices. This is the same number of vertices burned in a path after 2 rounds of burning. The third slice contains $5+3+1=9$ burned vertices, the same number of vertices that can be burned in a path after 3 rounds of burning. In total, $0+1+4+9+4+1=19$ vertices can be burned in the grid after 3 rounds of burning.

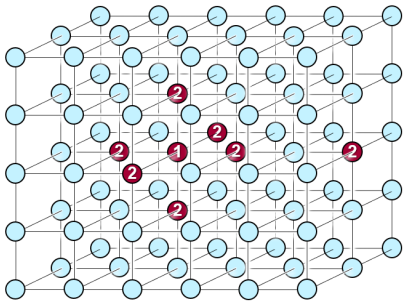


Figure 3.13: $P_3\square P_5\square P_6$ after 2 rounds of burning.

For a graph G, let $s \in V(G)$ be the first source that is burned in the first round of the burning process. Then the set of vertices burned after r rounds by source s is $N_{r-1}[s]$. We use $N^G_{r-1}[s]$ to specify the set of vertices burned by s in graph G. If H is an induced subgraph of G, then $N^H_{r-1}[s] \subseteq N^G_{r-1}[s]$.

Lemma 2. *If for any sequence of vertices $(x_1, x_2, \ldots, x_k)$ in a graph G, $|N_{k-1}[x_1] \cup N_{k-2}[x_2] \cup \cdots \cup N_0[x_k]| < |V(G)|$, then $b(G) > k$.*

Proof. If for any sequence of vertices $(x_1, x_2, \ldots, x_k)$ in G, $|N_{k-1}[x_1] \cup N_{k-2}[x_2] \cup \cdots \cup N_0[x_k]| < |V(G)|$, then for any sequence of vertices $(x_1, x_2, \ldots, x_k)$ in G, there is a vertex $v \in V(G) \setminus N_{k-1}[x_1] \cup N_{k-2}[x_2] \cup \cdots \cup N_0[x_k]$. Therefore, there is no sequence of vertices $(x_1, x2, \ldots, x_k)$ in graph G, such that $N_{k-1}[x_1] \cup N_{k-2}[x_2] \cup \cdots \cup N_0[x_k] = V(G)$. It is clear that no shorter sequences burn the graph either. From Lemma 1 it follows that $b(G) > k$. □

Lemma 3. *Let $G = \square^i_{j=1} P_{m_j}$ and $H = \square^{i-1}_{j=1} P_{m_j}$. Let $G_1, G_2, \ldots, G_{m_i}$ be induced subgraphs of G that are isomorphic to H, such that $V(G_1) \cup V(G_2) \cup \cdots \cup V(G_{m_i}) = V(G)$. Then for $x \in (1, \ldots, m_i)$, $1 - x \leq y \leq m_i - x$, and $s \in V(G_x)$:*

$$\left|N^{G_{x+y}}_{r-1}[s]\right| = \left|N^{G_x}_{r-1-|y|}[s]\right|$$

Proof. Let $G = \square^i_{j=1} P_{m_j}$ and $H = \square^{i-1}_{j=1} P_{m_j}$. Let $G_1, G_2, \ldots, G_{m_i}$ be induced subgraphs of G that are isomorphic to H, such that $V(G_1) \cup V(G_2) \cup \cdots \cup V(G_{m_i}) = V(G)$.

For $a_x, b_x \in V(G_x)$, let $D = d(a_x, b_x)$. for each y, there is a unique $a_{x+y} \in V(G_{x+y})$ such that $d(a_{x+y}, a_x) = |y|$. Then $d(a_{x+y}, b_x) = d(a_{x+y}, a_x) + d(a_x, b_x) = |y| + D$. This means $a_{x+y} \in N^G_d[b_x]$ if $d \geq |y| + D$.

Let $s \in V(G_x)$.

$$\begin{aligned} N^{G_{x+y}}_{r-1}[s] &= \left\{v_{x+y} \in G_{x+y} : d(v_{x+y}, s) \leq r-1\right\} \\ &= \left\{v_{x+y} \in G_{x+y} : d(v_{x+y}, v_x) + d(v_x, s) \leq r-1, v_x \in V(G_x), d(v_{x+y}, v_x) = |y|\right\} \\ &= \left\{v_{x+y} \in G_{x+y} : d(v_x, s) \leq r-1-|y|, v_x \in V(G_x), d(v_{x+y}, v_x) = |y|\right\} \end{aligned}$$

and

$$N^{G_x}_{r-1-|y|}[s] = \left\{v_x \in G_x : d(v_x, s) \leq r-1-|y|\right\}.$$

The maximum number of vertices of G_{x+y} burned by source s after r rounds is equal to the maximum number of vertices of G_x burned by source s after $r - |y|$ rounds.

$$\begin{aligned} \left|N^{G_x}_{r-1-|y|}[s]\right| &= \left|\{v_x \in G_x : d(v_x, s) \leq r-1-|y|\}\right| \\ &= \left|\{v_{x+y} \in G_{x+y} : d(v_x, s) \leq r-1-|y|, v_x \in V(G_x), d(v_{x+y}, v_x) = |y|\}\right| \\ &= \left|N^{G_{x+y}}_{r-1}[s]\right| \end{aligned}$$

□

Theorem 8. *For an infinite i-dimensional grid graph G, let $g(r, i)$ be the number of vertices burned by one source after r rounds of burning, i.e. for any $s \in V(G)$:*

$$g(r, i) = \left|N^G_{r-1}[s]\right|.$$

The maximum total number of vertices burned after r rounds of burning is

$$f(r, i) = \sum_{k=1}^{r} g(k, i).$$

Then:

$$g(r, i) = f(r, i-1) + f(r-1, i-1).$$

Proof. For infinite i-dimensional grid graph G, let $g(r, i)$ be the number of vertices burned by one source after r rounds of burning. Because the i-dimensional grid is infinite, the size of the $(r-1)$-neighborhood of s is the same for any vertex $s \in V(G)$.

$$g(r, i) = \left|N^G_{r-1}[s]\right|$$

If a sequence $(s_1, s_2, \ldots, s_r)$ is chosen such that for any $i \neq j$: $N^G_{r-i}[s_i] \cap N^G_{r-j}[s_j] = \varnothing$, then then number of vertices burned after r rounds is maximal:

$$f(r,i) = \sum_{k=1}^{r} g(k,i).$$

Our goal is to show $g(r,i+1) = f(r,i) + f(r-1,i)$.

To do this, we "break up" G into subgraphs and count the burned vertices inside these subgraphs.

Let H be an infinite $(i-1)$-dimensional grid graph. Let $H_1, H_2, H_3, \ldots$ be induced subgraphs of G, isomorphic to H, such that $\bigcup_{k=1}^{\infty} V(H_k) = V(G)$, and for any $x \neq y$: $V(H_x) \cap V(H_y) = \varnothing$. Then for any $v \in V(G)$:

$$|N^G_{r-1}[v]| = \left| \bigcup_{k=1}^{\infty} N^{H_k}_{r-1}[v] \right| = \sum_{k=1}^{\infty} \left| N^{H_k}_{r-1}[v] \right|$$

For $x > r-1$, take $s \in V(H_x)$. Then:

$$\begin{aligned}
g(r,i) &= \left|N^G_{r-1}[s]\right| = \sum_{k=1}^{\infty} \left|N^{H_k}_{r-1}[s]\right| \\
&= \sum_{k=1}^{x} \left|N^{H_k}_{r-1}[s]\right| + \sum_{k=x+1}^{\infty} \left|N^{H_k}_{r-1}[s]\right| \\
&= \sum_{k=x+1-r}^{x} \left|N^{H_k}_{r-1}[s]\right| + \sum_{k=x+1}^{x+r-1} \left|N^{H_k}_{r-1}[s]\right| \\
&= \sum_{y=1-r}^{0} \left|N^{H_{x+y}}_{r-1}[s]\right| + \sum_{y=1}^{r-1} \left|N^{H_{x+y}}_{r-1}[s]\right| \\
&= \sum_{y=1-r}^{0} \left|N^{H_x}_{r-1-|y|}[s]\right| + \sum_{y=1}^{r-1} \left|N^{H_x}_{r-1-|y|}[s]\right| \text{ (by Lemma 3)} \\
&= \sum_{k=1}^{r} \left|N^{H_x}_{r-k}[s]\right| + \sum_{k=1}^{r-1} \left|N^{H_x}_{r-1-k}[s]\right| \\
&= \sum_{k=1}^{r} g(k,i-1) + \sum_{k=1}^{r-1} g(k,i-1) \\
&= f(r,i-1) + f(r-1,i-1)
\end{aligned}$$

□

Theorem 9. *Let $g(r,i)$ and $f(r,i)$ as defined in Theorem 8. Let $G = \square_{j=1}^{i}$. Then:*

$$b(G) \geq \min\left\{ r : f(r,i) \geq \prod_{j=1}^{i} m_j \right\}$$

Proof. Let $g(r,i)$ be the number of vertices burned in r rounds by one source in an infinite i-dimensional grid, and let $f(r,i)$ be the maximum total number of vertices burned in r rounds in an infinite i-dimensional grid.

Take $G = \square_{j=1}^{i} P_{m_j}$. Let $a(r)$ be the maximum number of vertices burned by one source after r rounds of burning in G.

$$a(r) = \max\left\{ \left|N^G_{r-1}[v]\right| : v \in V(G) \right\}$$

Then $a(r) \leq g(r,i)$. Let $t(r)$ be the maximum total number of vertices of G burned in r rounds of burning. For every sequence of vertices $(x_1, x_2, \ldots, x_r)$ in G: $|N_{r-1}[x_1] \cup N_{r-2}[x_2] \cup \cdots \cup N_0[x_r]| \leq t(r)$. Then:

$$t(r) \leq \sum_{k=1}^{r} a(k)$$

Combined this gives:

$$t(r) \leq \sum_{k=1}^{r} a(k) \leq \sum_{k=1}^{r} g(k,i) = f(r,i)$$

If $f(r,i) < \prod_{j=1}^{i} m_j$, then $t(r) < \prod_{j=1}^{i} m_j$. From $t(r) < \prod_{j=1}^{i} m_j$, it follows that for every sequence of vertices $(x_1, x_2, \ldots, x_r)$ in G: $|N_{r-1}[x_1] \cup N_{r-2}[x_2] \cup \cdots \cup N_0[x_r]| \leq t(r) < \prod_{j=1}^{i} m_j = |V(G)|$. Therefore, by Lemma 2 we know $b(G) > r$. So indeed:

$$b(G) \geq \max\left\{ r : f(r,i) < \prod_{j=1}^{i} m_j \right\} + 1 = \min\left\{ r : f(r,i) \geq \prod_{j=1}^{i} m_j \right\}.$$

□

i	$g(r,i)$	$f(r,i)$
1	$1+2r$	r^2
2	$1+2r+2r^2$	$\frac{1}{3}r+\frac{2}{3}r^3$
3	$1+2\frac{2}{3}r+2r^2+1\frac{1}{3}r^3$	$\frac{2}{3}r^2+\frac{1}{3}r^4$
4	$1+2\frac{2}{3}r+3\frac{1}{3}r^2+1\frac{1}{3}r^3+\frac{2}{3}r^4$	$-\frac{7}{9}r+10\frac{4}{9}r^3-14\frac{2}{3}r^4+6r^5$
⋮	⋮	⋮
i	$f(r,i-1)+f(r-1,i-1)$	$\sum_{k=1}^{r} g(r,i)$

Table 3.2: $g(r,i)$ is the maximum number of vertices burned by one source after r rounds in $G = \square_{j=1}^{i} P_{m_j}$. $f(r,i)$ is the maximum number of vertices burned after r rounds in G.

Table 3.2 shows how many vertices are at most burned in graphs $G = \square_{j=1}^{i} P_{m_j}$ for $i \leq 4$ after r rounds of burning.

4

Higher-dimensional king's graphs

In this chapter lower and upper bounds for the burning number for the strong product of multiple path graphs are determined.

Burning in the king's graph creates a collection of burned squares. For one source, after the first round, only the source is burned. In the second round, all 8 neighbors of the source are burned. These are the orthogonal neighbors and the diagonal neighbors. Together, they form a 3×3 square around the source. In the next round, the square is 5×5, etc. Every new burned area is the next odd-sided square. So after r rounds, the source has burned $(2r-1)^2$ vertices. Figure 4.1 shows 3 rounds of burning in the king's graph around a single source.

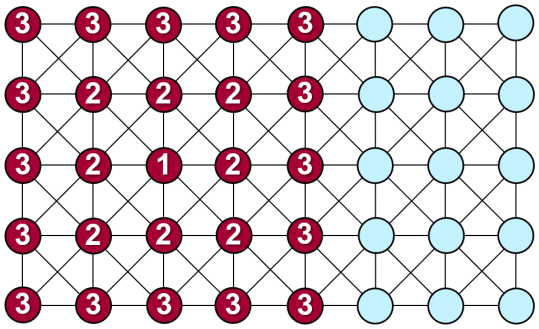


Figure 4.1: A 5×8 king's graph burned in 3 rounds. Dark red vertices are burned. Light blue vertices are unburned. The number inside the vertex denotes the round in which the vertex was burned. The vertex burned in the first round is the source.

Continuing on the bounds for the burning number of a grid graph, Mitsche et al. derived a similar bound for the king's graph [14].

Theorem 10 ([14])**.** *Let $G = P_m \boxtimes P_n$, with $1 \le m \le n$, where $m = m(n)$ is a function of n. Then:*

$$b(G) = \begin{cases} (1+o(1))\sqrt[3]{\frac{3}{4}mn} & \text{if } m = \omega(\sqrt{n}) \\ \Theta(\sqrt{n}) & \text{if } m = O(\sqrt{n}) \end{cases}$$

4.1. King's cube graph

Now the question is: Can we find similar bounds for the king's cube graph $G = P_l \boxtimes P_m \boxtimes P_n$?

If the $m \times n$ king's graph can be represented as a rectangle. A king's cube graph $G = P_l \boxtimes P_m \boxtimes P_n$ can be represented as a box (like a cube but with sides of length l, m, and n).

In graph G, burning goes in cubes. In the first round, one source burns only itself. In the next round, it burns all adjacent vertices. This forms a cube around the source with odd-length sides. So in r rounds, one source can burn $(2r-1)^3$ vertices.

4.1.1. Lower bound for burning cube king's graphs

Lemma 4. *Let $G = P_l \boxtimes P_m \boxtimes P_n$, then*

$$b(G) \geq \sqrt{\frac{1}{4} + \sqrt{\frac{1}{16} + \frac{1}{2} l \cdot m \cdot n}} \geq \left(\frac{1}{2} l \cdot m \cdot n\right)^{\frac{1}{4}}.$$

Proof. Knowing one source burns $(2r-1)^3$ vertices after r rounds, all sources burn at most

$$\begin{aligned}
\sum_{i=1}^{r} (2i-1)^3 &= \sum_{i=1}^{r} (8i^3 - 12i^2 + 6i - 1) \\
&= 8\sum_{i=1}^{r} (i^3) - 12\sum_{i=1}^{r} (i^2) + 6\sum_{i=1}^{r} (i) - \sum_{i=1}^{r} (1) \\
&= 8 \cdot \frac{1}{4} r^2 (r+1)^2 - 12 \cdot \frac{1}{6} r(r+1)(2r+1) + 6 \cdot \frac{1}{2} r(r+1) - r \\
&= 2r^4 - r^2.
\end{aligned}$$

If $\sum_{i=1}^{r} (2i-1)^3 < lmn$, not all vertices in G can be burned in r rounds. So, to find the lower bound for the burning number, we need $r = \min\{r : \sum_{i=1}^{r} (2i-1)^3 \geq lmn\}$. This is when $\sum_{i=1}^{r} (2i-1)^3 = lmn$.

$$\begin{aligned}
2r^4 - r^2 &= l \cdot m \cdot n \\
2(r^2)^2 - (r^2) - l \cdot m \cdot n &= 0 \\
(r^2)^2 - \frac{1}{2}(r^2) - \frac{1}{2} l \cdot m \cdot n &= 0 \\
(r^2 - \frac{1}{4})^2 - \frac{1}{16} - \frac{1}{2} l \cdot m \cdot n &= 0 \\
r^2 - \frac{1}{4} &= \sqrt{\frac{1}{16} + \frac{1}{2} l \cdot m \cdot n} \\
r &= \sqrt{\frac{1}{4} + \sqrt{\frac{1}{16} + \frac{1}{2} l \cdot m \cdot n}}
\end{aligned}$$

□

4.1.2. Upper bound for burning cube king's graphs using equally sized cubes

Lemma 5. *Let $G = P_l \boxtimes P_m \boxtimes P_n$, then*

$$b(G) \leq \min_{odd\, R} \left(\frac{1}{2} R - \frac{1}{2} + \left\lceil \frac{n}{R} \right\rceil \cdot \left\lceil \frac{m}{R} \right\rceil \cdot \left\lceil \frac{l}{R} \right\rceil \right).$$

To find an upper bound for the burning number of $G = P_l \boxtimes P_m \boxtimes P_n$, we need a sequence S of length r_u that burns the whole graph. Then $b(G) \leq r_u$.

Proof. Let $R = 2r+1$ be the length of the side of a cube that is formed by burning a source$s$ r rounds. $|N_{r-1}[s]| = (2r+1)^3 = R^3$.

We cover $G = P_l \boxtimes P_m \boxtimes P_n$ using equally sized (same side length) cubes. All cubes have sides of length R, so they have the size they would have after r rounds of burning. First the cubes are placed in a row until they cover the n direction. For this, $\lceil \frac{n}{R} \rceil$ cubes are needed. Next, these rows are placed next to each other in the m direction to form a sheet that covers the $m \times n$ rectangle. For this, $\lceil \frac{m}{R} \rceil$ rows are needed. Last, the sheets are stacked in the l direction to cover all of G. For this, $\left\lceil \frac{l}{R} \right\rceil$ sheets are needed. In total $T = \lceil \frac{n}{R} \rceil \cdot \lceil \frac{m}{R} \rceil \cdot \left\lceil \frac{l}{R} \right\rceil$ cubes are used to cover G. Burning the centers of all T cubes as sources (in T rounds), the first source has formed a $2T-1$ sided cube but the last has only burned itself. To get all cubes to be at least R sided, we have to wait for $r = \frac{1}{2}R - \frac{1}{2}$ extra rounds. In total after $\frac{1}{2}R - \frac{1}{2} + \lceil \frac{n}{R} \rceil \cdot \lceil \frac{m}{R} \rceil \cdot \left\lceil \frac{l}{R} \right\rceil$ rounds, the whole graph is burned. So for any odd R: $b(P_l \boxtimes P_m \boxtimes P_n) \leq \frac{1}{2}R + \frac{1}{2} + \lceil \frac{n}{R} \rceil \cdot \lceil \frac{m}{R} \rceil \cdot \left\lceil \frac{l}{R} \right\rceil$. □

Theorem 11. *Let $G_n = P_n \boxtimes P_n \boxtimes P_n$, and let*

$$R = \frac{1}{\sqrt{2}}\left(\sqrt{3n-3} + \sqrt{\left(3+2\sqrt{6n-6}\right)(n-1)}\right).$$

Let R_{odd} be the closest odd integer to R.

$$R_{odd} = argmin_x \{|x-R| : x = 2k+1\}$$

then

$$b(G_n) \leq \frac{1}{2}R_{odd} - \frac{1}{2} + \left(\frac{n+R_{odd}-1}{R_{odd}}\right)^3.$$

Proof. Let $G_n = P_n \boxtimes P_n \boxtimes P_n$. From Lemma 4.1.2 we know

$$\begin{aligned} b(G_n) &\leq \min_{\text{odd } R}\left(\frac{1}{2}R - \frac{1}{2} + \left\lceil\frac{n}{R}\right\rceil \cdot \left\lceil\frac{n}{R}\right\rceil \cdot \left\lceil\frac{n}{R}\right\rceil\right) \\ &= \min_{\text{odd } R}\left(\frac{1}{2}R - \frac{1}{2} + \left\lceil\frac{n}{R}\right\rceil^3\right). \end{aligned}$$

To find the minimum, we would like to use the derivative of the expression. This is not possible because the expression is not continuous. A simplified version of the expression $r(R)$ is differentiable.

$$\frac{1}{2}R + \frac{1}{2} + \left\lceil\frac{n}{R}\right\rceil^3 \leq \frac{1}{2}R + \frac{1}{2} + \left(\frac{n+R-1}{R}\right)^3 = r(R)$$

The idea is that the real valued R that minimizes r is close to the odd R that minimizes r. If $\frac{dr}{dR}(R_0) = 0$ and $\frac{d^2r}{dR^2}(R_0) \geq 0$, then $r(R_0) = \min_R(r(R))$.

$$\begin{aligned} 0 &= \frac{dr}{dR} \\ &= \frac{d}{dR}\left[\frac{1}{2}R + \frac{1}{2} + \left(\frac{n+R-1}{R}\right)^3\right] \\ &= \frac{d}{dR}\left[\frac{1}{2}R + \frac{1}{2} + 1 + (-3+3n)R^{-1} + (3n^2-6n+3)R^{-2} + (n^3-3n^2+3n-1)R^{-3}\right] \\ &= \frac{1}{2} - (-3+3n)R^{-2} - 2(3n^2-6n+3)R^{-3} - 3(n^3-3n^2+3n-1)R^{-4} \end{aligned}$$

As $R > 0$, we can multiply by $2R^4$:

$$0 = R^4 + (6-6n)R^2 + (-12n^2+24n-12)R + (6n^3+18n^2-18n+6)$$

This gives one root that satisfies $R \in \mathbb{R}_{>0}$:

$$R = \frac{1}{\sqrt{2}}\left(\sqrt{3}\sqrt{n-1} + \sqrt{\left(2\sqrt{6}\sqrt{n-1}+3\right)(n-1)}\right)$$

To check this is indeed a (local) minimum: For $n > 1$:

$$\begin{aligned} -6+6n &> -6+6 = 0 \\ 18n^2-36n+18 &= 18(n-1)^2 \geq 18(0)^2 = 0 \\ 12n^3-36n^2+36n-12 &= 12(n-1)^3 \geq 12(0)^3 = 0 \end{aligned}$$

Therefore

$$\frac{d^2r}{dR^2} = (-6+6n)R^{-3} + (18n^2-36n+18)R^{-4} + (12n^3-36n^2+36n-12)R^{-5} > 0.$$

$r(R)$ is indeed a minimum. Now let R_{odd} be the closest odd integer to R, then we have upper bound:

$$b(P_n \boxtimes P_n \boxtimes P_n) \leq \frac{1}{2}R_{odd} - \frac{1}{2} + \left(\frac{n+R_{odd}-1}{R_{odd}}\right)^3$$

□

For $n = 22$, $R \approx 21{,}96$ so $R_{\text{odd}} = 21$. Then $r(R_{\text{odd}}) = \frac{1}{2}\cdot 21 - \frac{1}{2} + \left(\frac{21+22-1}{21}\right)^3 = 10 + 2^3 = 18$. So $b(G_{22}) \leq 18$. Table 4.1 shows different R_{odd} values and corresponding upper bounds.

n	R_{odd}	$r(R_{\text{odd}})$
5	7	6,88
10	13	10,85
15	17	14,06
20	21	16,91
25	23	19,53
30	27	21,92
40	33	26,39
50	39	30,49
60	45	34,34
70	49	37,97
80	53	41,45
90	59	44,78
100	63	48,00
1000	319	229,53
10000	1693	1175,38

Table 4.1: The left column contains values of n for graph $G_n = P_n \boxtimes P_n \boxtimes P_n$. The middle column gives the corresponding R_{odd} value. The right column gives the upper bound for $b(G_n)$.

4.1.3. Upper bound for burning cube king's graphs using a single cube

When looking strictly at king's cubes ($P_n \boxtimes P_n \boxtimes P_n$), there is another way to find an upper bound for the burning number. One cube can be used to cover all vertices.

Theorem 12. *Let $G_n = P_n \boxtimes P_n \boxtimes P_n$, then*

$$b(G_n) \leq \left\lceil \frac{1}{2}(n+1) \right\rceil.$$

Proof. Let $G_n = P_n \boxtimes P_n \boxtimes P_n$. The vertices burned by one source after r rounds form a $2r-1$ sided cube. One cube contains $(2r-1)^3$ vertices, $|V(G_n)| = n^3$. To cover the whole graph we need $(2r-1)^3 \geq |V(G_n)| = n^3$. So if $2r-1 \geq n$, that one source, if chosen correctly, burns the whole graph in r rounds. That is if $r \geq \frac{1}{2}(n+1)$. For odd n, one source can burn the graph in $\frac{1}{2}(n+1)$ rounds. For even n, one source can burn the graph in $\frac{1}{2}(n+2)$ rounds.

$$b(G_n) \leq \left\lceil \frac{1}{2}(n+1) \right\rceil$$

□

For $n = 22$, $\left\lceil \frac{1}{2}(n+1) \right\rceil = \left\lceil 11\frac{1}{2} \right\rceil = 12$, so $b(G_{22}) \leq 12$. For small n, the upper bound from Theorem 12 is lower than the one from Theorem 11. This can be seen in Table 4.2. Theorem 11 and Theorem 12 can be combined to form one upper bound.

Theorem 13. *Let $G_n = P_n \boxtimes P_n \boxtimes P_n$, and let*

$$R = \frac{1}{\sqrt{2}}\left(\sqrt{3n-3} + \sqrt{\left(3 + 2\sqrt{6n-6}\right)(n-1)}\right).$$

Let R_{odd} be the closest odd integer to R.

$$R_{odd} = argmin_x\{|x-R| : x = 2k+1\}$$

then

$$b(G_n) \leq \min\left\{\frac{1}{2}R_{odd} - \frac{1}{2} + \left(\frac{n+R_{odd}-1}{R_{odd}}\right)^3, \left\lceil \frac{1}{2}(n+1) \right\rceil\right\}.$$

The burning numbers of graphs $G_n = P_n \boxtimes P_n \boxtimes P_n$ for $n = 1, \ldots, 17$, have been calculated using ILP-CMCP [8]. G_{17} is the last graph evaluated because it was decided to only calculate the burning number if it can be calculated within 24 hours. In Table 4.3, the lower bound, upper bound and actual burning number for G_n are compared. For all evaluated G_n, the upper bound is equal to the actual burning number.

n	Thm 11	Thm 12
5	6	3
10	10	6
15	14	8
20	16	11
25	19	13
30	21	16
40	26	21
50	30	26
60	34	31
70	37	36
80	41	41
90	44	46
100	48	51
1000	229	501
10000	1175	5001

Table 4.2: Comparing the upper bound for the burning number of G_n from Theorem 11 and Theorem 12.

n	Lower bound	$b(G_n)$	Upper bound
1	1	1	1
2	2	2	2
3	2	2	2
4	3	3	3
5	3	3	3
6	4	4	4
7	4	4	4
8	5	5	5
9	5	5	5
10	5	6	6
11	6	6	6
12	6	7	7
13	6	7	7
14	7	8	8
15	7	8	8
16	7	9	9
17	8	9	9

Table 4.3: Comparing the lower bound from Lemma 4 and upper bound from Theorem 13, to the actual burning number of graph $G_n = P_n \boxtimes P_n \boxtimes P_n$, calculated using ILP-CMCP [8].

4.2. Beyond cubes

Some of the techniques used to find upper and lower bounds for the burning number of king's cube graphs can also be used for graphs of the form $G_i = \boxtimes_{j=1}^{i} P_{m_j}$.

The technique for finding an upper bound for the burning number of $G_3 = \boxtimes_{j=1}^{3} P_n$ from Theorem 12 can be expanded for any i.

Theorem 14. *Let $G_{i,n} = \boxtimes_{j=1}^{i} P_n$. Then*

$$b(G_{i,n}) \leq \left\lceil \frac{1}{2}(n+1) \right\rceil.$$

Proof. Let $G_{i,n} = \boxtimes_{j=1}^{i} P_n$. The vertices burned by one source after r rounds form a $2r-1$ sided hypercube. One hypercube contains $(2r-1)^i$ vertices, $\left|V(G_{i,n})\right| = n^i$. To cover the whole graph we need $(2r-1)^i \geq \left|V(G_{i,n})\right| = n^i$. So if $2r-1 \geq n$, that one source, if chosen correctly, burns the whole graph in r rounds. That is if $r \geq \frac{1}{2}(n+1)$. For odd n, one source can burn the graph in $\frac{1}{2}(n+1)$ rounds. For even n, one source can burn

the graph in $\frac{1}{2}(n+2)$ rounds.

$$b(G_{i,n}) \leq \left\lceil \frac{1}{2}(n+1) \right\rceil$$

□

Similarly for any $G_i = \boxtimes_{j=1}^{i} P_{m_j}$ an upper bound for its burning number is $b(G_i) \leq \left\lceil \frac{1}{2}(\max\{m_j\}+1) \right\rceil$. Because if the $2r-1$ sided hypercube burns $\boxtimes_{j=1}^{i} P_{\max\{m_j\}}$ in $r = \left\lceil \frac{1}{2}(\max\{m_j\}+1) \right\rceil$ rounds, then it also burns G_i.

Remark. $G_{i,n} = \boxtimes_{j=1}^{i} P_n$ in Theorem 14 contains $\left|V(G_{i,n})\right| = n^i$ vertices. Therefore $b(G_{i,n}) \leq \left\lceil \frac{1}{2}\left|V(G_{i,n})\right|^{\frac{1}{i}} + \frac{1}{2} \right\rceil < \left\lceil \sqrt{\left|V(G_{i,n})\right|} \right\rceil$. The burning number conjecture holds for $G_{i,n}$.

A lower bound for G_i is determined by finding the maximum number of vertices that can be burned in r rounds.

Theorem 15. *Let $G_i = \boxtimes_{j=1}^{i} P_{m_j}$. Then*

$$b(G_i) > \frac{1}{2}\left|V(G_i)\right|^{\frac{1}{i+1}}.$$

To prove this theorem, we need Lemma 6.

Lemma 6. *For $i \in \mathbb{N}_{\geq 1}$ and $r \in \mathbb{N}_{\geq 1}$:*

$$\sum_{k=1}^{r} (2k-1)^i \leq \frac{2^i}{i+1} r^{i+1}$$

The proof of Lemma 6 uses Bernoulli polynomials. An explicit formula for the *Bernoulli polynomial* $B_i(x)$ is:

$$B_i(r) = \sum_{j=0}^{i} \binom{i}{j} B_{i-j} \cdot x^j,$$

where B_{i-j} are Bernoulli numbers. An explicit formula for the *Bernoulli number* B_n is:

$$B_n = \sum_{k=0}^{n} \frac{1}{k+1} \sum_{m=0}^{k} \binom{k}{m} (-1)^m m^n.$$

For odd n, except for $n=1$, $B_n = 0$. For even n, B_n alternates between positive and negative values.

Proof of Lemma 6. Let $\sum_{k=1}^{r}(2k-1)^i = \sum_{k=0}^{r-1}(1+2k)^i = S_{1,2}^i(r)$. This sum can be described in terms of Bernoulli polynomials [3].

$$S_{1,2}^i(r) = \frac{2^i}{i+1}\left(B_{i+1}\left(r+\frac{1}{2}\right) - B_{i+1}\left(\frac{1}{2}\right)\right)$$

The leading term of $S_{1,2}^i(r)$ is $\frac{2^i}{i+1} r^{i+1}$ [4]. Let $d_i(r) = S_{1,2}^i(r) - \frac{2^i}{i+1} r^{i+1}$
Our goal is to show

$$d_i(r) \leq 0$$

We do this by induction on r.
Base case: Take $r=1$, then

$$\begin{aligned} d_i(1) &= \left(\sum_{k=1}^{1} (2k-1)^i\right) - \frac{2^i}{i+1}(1)^{i+1} \\ &= 1 - \frac{2^i}{i+1} \\ &\leq 0 \text{ (for } i \geq 1) \end{aligned}$$

Induction step: Now assume for some r, $d_i(r) \leq 0$. We will show $d_i(r+1) \leq d_i(r) \leq 0$ by showing $d_i(r+1) - d_i(r) \leq 0$. We use the difference of two successive Bernoulli polynomials [20].

$$B_{i+1}(x+1) - B_{i+1}(x) = (i+1)x^i$$

This yields:

$$
\begin{aligned}
&d_i(r+1) - d_i(r) \\
&= \left[\frac{2^i}{i+1}\left(B_{i+1}\left(r+\frac{1}{2}+1\right) - B_{i+1}\left(\frac{1}{2}\right)\right) - \frac{2^i}{i+1}(r+1)^{i+1}\right] - \left[\frac{2^i}{i+1}\left(B_{i+1}\left(r+\frac{1}{2}\right) - B_{i+1}\left(\frac{1}{2}\right)\right) - \frac{2^i}{i+1}(r)^{i+1}\right] \\
&= \frac{2^i}{i+1}\left[B_{i+1}\left(r+\frac{1}{2}+1\right) - B_{i+1}\left(r+\frac{1}{2}\right) + r^{i+1} - (r+1)^{i+1}\right] \\
&= \frac{2^i}{i+1}\left[(i+1)\left(r+\frac{1}{2}\right)^i + r^{i+1} - (r+1)^{i+1}\right]
\end{aligned}
$$

The expression of the equation inside the brackets can be written as:

$$
\begin{aligned}
(i+1)\left(r+\frac{1}{2}\right)^i + r^{i+1} - (r+1)^{i+1} &= r^{i+1} - \left[\sum_{j=0}^{i+1}\binom{i+1}{j}(1)^{i+1-j}r^j\right] + (i+1)\left[\sum_{j=0}^{i}\binom{i}{j}\left(\frac{1}{2}\right)^{i-j}r^j\right] \\
&= r^{i+1} - r^{i+1} - \left[\sum_{j=0}^{i}\binom{i+1}{j}r^j\right] + (i+1)\left[\sum_{j=0}^{i}\binom{i}{j}\left(\frac{1}{2}\right)^{i-j}r^j\right] \\
&= \sum_{j=0}^{i}\left[-\binom{i+1}{j}r^j + (i+1)\binom{i}{j}\left(\frac{1}{2}\right)^{i-j}r^j\right] \\
&= \sum_{j=0}^{i} r^j\left((i+1)\binom{i}{j}\left(\frac{1}{2}\right)^{i-j} - \binom{i+1}{j}\right). \qquad (4.1)
\end{aligned}
$$

(4.2)

If every term of this sum is negative, the total sum is also negative.

$$
\begin{aligned}
(i+1)\binom{i}{j}\left(\frac{1}{2}\right)^{i-j} - \binom{i+1}{j} &= \frac{i!(i+1)}{(i-j)!j!}\left(\frac{1}{2}\right)^{i-j} - \frac{(i+1)!}{(i+1-j)!j!} \\
&= \frac{(i+1)!}{(i-j)!j!}\left(\frac{1}{2}\right)^{i-j} - \frac{(i+1)!}{(i-j)!j!(i+1-j)} \\
&= \frac{(i+1)!}{(i-j)!j!}\left(\left(\frac{1}{2}\right)^{i-j} - \frac{1}{i+1-j}\right)
\end{aligned}
$$

It is clear that for $j = 0,\ldots,i$: $2^{i-j} \geq 1 \geq i+1-j$. This implies $\frac{1}{2^{i-j}} \leq \frac{1}{i+1-j}$. Therefore, every term in the sum in Equation 4.1 is non-positive, i.e.,

$$\left(\frac{1}{2}\right)^{i-j} - \frac{1}{i+1-j} \leq 0.$$

This means

$$d_i(r+1) - d_i(r) = \frac{2^i}{i+1}\left[(i+1)\left(r+\frac{1}{2}\right)^i + r^{i+1} - (r+1)^{i+1}\right] \leq 0.$$

Therefore,

$$d_i(r+1) \leq d_i(r) \leq 0,$$

where the second inequality follows from the induction hypothesis. Thus,

$$\frac{2^i}{i+1}r^{i+1} \geq S^i_{1,2}(r).$$

□

This result can be used to prove Theorem 15.

Proof of Theorem 15. Let $G_i = \boxtimes_{j=1}^{i} P_{m_j}$. The graph contains $\prod_{j=1}^{i} m_j$ vertices. Suppose that we have an optimal burning sequence $(x_1, \ldots, x_r)$, where $r = b(G_i)$. Then every vertex of G_i must be burned by the sequence. From Lemma 1, we get:

$$\begin{aligned}\prod_{j=1}^{i} m_j = |V(G_i)| &\leq |N_{r-1}[x_1]| + |N_{r-2}[x_2]| + \cdots + |N_0[x_r]| \\ &\leq \sum_{k=1}^{r} (2k-1)^i \leq \frac{2^i}{i+1} r^{i+1} \text{ (by Lemma 6).}\end{aligned}$$

Then we find

$$\begin{aligned}\frac{2^i}{i+1} r^{i+1} &\geq \prod_{j=1}^{i} m_j \\ r^{i+1} &\geq \frac{i+1}{2^i} \prod_{j=1}^{i} m_j \\ r &\geq \left(\frac{i+1}{2^i} \prod_{j=1}^{i} m_j \right)^{\frac{1}{i+1}}.\end{aligned}$$

We have found $b(G_i) \geq \left(\frac{i+1}{2^i} \prod_{j=1}^{i} m_j \right)^{\frac{1}{i+1}}$. This expression can be simplified.

$$b(G_i) = r \geq \left(\frac{i+1}{2^i} \prod_{j=1}^{i} m_j \right)^{\frac{1}{i+1}} = (i+1)^{\frac{1}{i+1}} 2^{-\frac{i}{i+1}} \prod_{j=1}^{i} m_j^{\frac{1}{i+1}} > \frac{1}{2^{\frac{i}{i+1}}} |V(G_i)|^{\frac{1}{i+1}} > \frac{1}{2} |V(G_i)|^{\frac{1}{i+1}}$$

In conclusion,

$$b(G_i) > \frac{1}{2} |V(G_i)|^{\frac{1}{i+1}}.$$

□

5

Conclusion and discussion

In this thesis, we determined upper and lower bounds for Cartesian and strong products of multiple path graphs. The techniques used to find these bounds were adapted from the techniques used to determine the burning number of grid graphs and king's graphs by Mitsche et al. [13, 14].

In Chapter 3, an implicit lower bound for the burning number of graphs of the form $G_i = \square_{j=1}^{i} P_{m_j}$ was found in Theorem 9. A logical next step is to make this bound explicit. Due to the complicated shapes created when burning in these graphs, we were unable to find an upper bound. Further research is needed to find a way to arrange these shapes such that overlap is minimal. Theorem 7 shows that if G is a subgrid of H, then the burning number of G is smaller than or equal to the burning number of H. In the future, this theorem could be extended to include Cartesian products of multiple path graphs and strong products of multiple path graphs.

In Chapter 4, one lower bound (Lemma 4) and two upper bounds (Theorems 11 and 12) were found for graphs of the form $G_n = P_n \boxtimes P_n \boxtimes P_n$. The upper bound from Theorem 11 was determined using a simplified version of the technique used by Mitsche et al. Instead of covering the graph with growing cubes, it uses cubes of one size. It would be interesting to see whether using growing cubes would strongly improve the upper bound. The upper bound from Theorem 12 uses a single source to burn the graph. For small n, this bound is better than the one from Theorem 11. Theorem 14 expands this upper bound from graphs of type $G_n = P_n \boxtimes P_n \boxtimes P_n$ to graphs of type $G_{i,n} = \boxtimes_{j=1}^{i} P_n$. In Theorem 15, an explicit lower bound was determined for these graphs $G_{i,n}$. It would be interesting to find out whether this lower bound can be improved by letting the maximum number of vertices that can be burned by one source be limited by the shape of the graph.

The lower and upper bounds found in this thesis can increase the speed at which burning numbers for higher-dimensional grid and king's graphs can be calculated using the ILP.

Another interesting line of research would be to find upper and lower bounds for the burning number of other graphs with a predictable pattern of burning. Possible examples are the hexagonal grid graph and the triangular grid graph.

Bibliography

[1] Noga Alon. Transmitting in the n-dimensional cube. *Discrete Applied Mathematics*, 37-38:9–11, 1992. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(92)90121-P. URL https://www.sciencedirect.com/science/article/pii/0166218X9290121P.

[2] 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):P4.2, Oct. 2023. doi: 10.37236/11113. URL https://www.combinatorics.org/ojs/index.php/eljc/article/view/v30i4p2.

[3] A. Bazsó, Á. Pintér, and H.M. Srivasrava. A refinement of faulhaber's theorem concerning sums of powers of natural numbers. *Applied Mathematics Letters*, 25:486–489, 2012. ISSN 08939659. doi: https://doi.org/10.1016/j.aml.2011.09.042.

[4] András Bazsó and István Mező. On the coefficients of power sums of arithmetic progressions. *Journal of Number Theory*, 153:117–123, 2015. ISSN 0022-314X. doi: https://doi.org/10.1016/j.jnt.2015.01.019. URL https://www.sciencedirect.com/science/article/pii/S0022314X15000839.

[5] Stéphane Bessy, Anthony Bonato, Jeannette Janssen, Dieter Rautenbach, and Elham Roshanbin. Burning a graph is hard. *Discrete Applied Mathematics*, 232:73–87, 2017. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2017.07.016. URL https://www.sciencedirect.com/science/article/pii/S0166218X17303384.

[6] Anthony Bonato and Shahin Kamali. Approximation algorithms for graph burning. In T.V. Gopal and Junzo Watada, editors, *Theory and Applications of Models of Computation*, pages 74–92, Cham, 2019. Springer International Publishing. ISBN 978-3-030-14812-6.

[7] Antony Bonato, Jeannette Janssen, and Elham Roshanbin. How to burn a graph. *Internet Mathematics*, 12:85–100, 2016.

[8] J. García-Díaz, J.A. Cornejo-Acosta, and Trejo-Sánchez. A greedy heuristic for graph burning. *Computing*, 107(91), 2025. doi: https://doi.org/10.1007/s00607-025-01436-9.

[9] Arya Tanmay Gupta, Swapnil A. Lokhande, and Kaushik Mondal. *Burning Grids and Intervals*, page 66–79. Springer International Publishing, 2021. ISBN 9783030678999. doi: 10.1007/978-3-030-67899-9_6. URL http://dx.doi.org/10.1007/978-3-030-67899-9_6.

[10] Anjeneya Swami Kare and I. Vinod Reddy. Parameterized algorithms for graph burning problem. In Charles J. Colbourn, Roberto Grossi, and Nadia Pisanti, editors, *Combinatorial Algorithms*, pages 304–314, Cham, 2019. Springer International Publishing. ISBN 978-3-030-25005-8.

[11] Hongbin Lin, Xi Zhang, and Xianghua Fu. A graph convolutional encoder and decoder model for rumor detection. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 300–306, 2020. doi: 10.1109/DSAA49011.2020.00043.

[12] Huiqing Liu, Xuejiao Hu, and Xiaolan Hu. Burning number of caterpillars. *Discrete Applied Mathematics*, 284:332–340, 2020. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2020.03.062. URL https://www.sciencedirect.com/science/article/pii/S0166218X2030161X.

[13] Dieter Mitsche, Pawel Pralat, and Elham Roshanbin. Burning graphs: A probabilistic perspective. *Graphs and Combinatorics*, 33:449–471, 2017.

[14] Dieter Mitsche, Pawel Pralat, and Elham Roshanbin. Burning number of graph products. *Theoretical Computer Science*, 746:124–135, 2018.

[15] Yukihiro Murakami. The burning number conjecture is true for trees without degree-2 vertices. *Graphs and Combinatorics*, 40(82), 2024. doi: 10.1007/s00373-024-02812-6.

[16] Jiajun Ning, Xian'an Jin, and Meiqiao Zhang. The burning number conjecture holds for trees of order n with at most n-1 degree-2 vertices. *Graphs and Combinatorics*, 42(29), 2026. doi: 10.1007/s00373-026-03024-w.

[17] Sergey Norin and Jérémie Turcotte. The burning number conjecture holds asymptotically. *Journal of Combinatorial Theory, Series B*, 168:208–235, 2024. ISSN 0095-8956. doi: https://doi.org/10.1016/j.jctb.2024.05.003. URL https://www.sciencedirect.com/science/article/pii/S009589562400042X.

[18] 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. volume 308, pages 94:1–94:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi: 10.4230/LIPICS.ESA.2024.94. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ESA.2024.94.

[19] Angelin Jemima Rajasingh and N. Parthiban. Large graphs with small burning numbers. In Indra Rajasingh, Sharmila Mary Arul, and Diana Grace Thomas, editors, *Applied Computational Mathematics*, pages 151–153, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-77764-6.

[20] E. T. Whittaker and G. N. Watson. *The expansion of functions in Infinite Series*, page 125–149. Cambridge Mathematical Library. Cambridge University Press, 1996.

The python code in Appendix A and Appendix D have partially been generated using Vibe (Mistral AI).

A

Generating grid graph files

This code generates a file that describes an $m \times n$ grid graph in the format required to use the ILP [8].

```python
def genereer_grid_bestand(m, n, burning_sequence=None):
    # Bereken het aantal knopen en lijnen
    aantal_knopen = m * n
    aantal_lijnen = 2 * m * n - m - n

    # Open het bestand om te schrijven
    with open(f"grid{m}x{n}.txt", "w") as bestand:
        # Schrijf het aantal knopen (m x n)
        bestand.write(f"{aantal_knopen}\n")

        # Schrijf het aantal lijnen
        bestand.write(f"{aantal_lijnen}\n")

        # Schrijf de burning sequence (optioneel)
        if burning_sequence is not None:
            bestand.write(f"{burning_sequence}\n")
        else:
            bestand.write("\n")  # Lege regel

        # Genereer alle lijnen in de grid
        # Horizontale lijnen: (i, j) -- (i, j+1)
        for i in range(m):
            for j in range(n - 1):
                a = i * n + j + 1  # Knoopnummer (1-based)
                b = i * n + j + 2
                bestand.write(f"{a} {b}\n")

        # Verticale lijnen: (i, j) -- (i+1, j)
        for i in range(m - 1):
            for j in range(n):
                a = i * n + j + 1
                b = (i + 1) * n + j + 1
                bestand.write(f"{a} {b}\n")

    print(f"Bestand 'grid{m}x{n}.txt' is gegenereerd.")
```

B

Generating torus grid graph files

This code generates a file that describes an $m \times n$ torus grid graph in the format required to use the ILP [8].

```
def genereer_torus_bestand(m, n, burning_sequence=None):
    # Bereken het aantal knopen en lijnen
    aantal_knopen = m * n
    aantal_lijnen = 2 * m * n
    if m<3:
        aantal_lijnen -= n
    if n<3:
        aantal_lijnen -= m
    # Open het bestand om te schrijven
    with open(f"torus_grid{m}x{n}.txt", "w") as bestand:
        # Schrijf het aantal knopen (m x n)
        bestand.write(f"{aantal_knopen}\n")

        # Schrijf het aantal lijnen
        bestand.write(f"{aantal_lijnen}\n")

        # Schrijf de burning sequence (optioneel)
        if burning_sequence is not None:
            bestand.write(f"{burning_sequence}\n")
        else:
            bestand.write("\n")  # Lege regel

        # Genereer alle lijnen in de grid
        # Horizontale lijnen: (i, j) -- (i, j+1)
        for i in range(m):
            for j in range(n - 1):
                a = i * n + j + 1  # Knoopnummer (1-based)
                b = i * n + j + 2
                bestand.write(f"{a} {b}\n")


        # Verticale lijnen: (i, j) -- (i+1, j)
        for i in range(m - 1):
            for j in range(n):
                a = i * n + j + 1
                b = (i + 1) * n + j + 1
                bestand.write(f"{a} {b}\n")
        if m>2:
            for j in range(n):
                a = j + 1
                b = (m - 1)* n + j + 1
                bestand.write(f"{a} {b}\n")
        if n>2:
            for i in range(m):
                a = i * n + 1
                b = (i + 1) * n
                bestand.write(f"{a} {b}\n")

    print(f"Bestand 'grid{m}x{n}.txt' is gegenereerd.")
```

C

Generating king's graph files

This code generates a file that describes an $m \times n$ king's graph in the format required to use the ILP [8].

```python
def genereer_king_bestand(m, n, burning_sequence=None):
    # Bereken het aantal knopen en lijnen
    aantal_knopen = m * n
    aantal_lijnen = 2 * m * n - m - n +2*(m-1)*(n-1)

    # Open het bestand om te schrijven
    with open(f"king{m}x{n}.txt", "w") as bestand:
        # Schrijf het aantal knopen (m x n)
        bestand.write(f"{aantal_knopen}\n")

        # Schrijf het aantal lijnen
        bestand.write(f"{aantal_lijnen}\n")

        # Schrijf de burning sequence (optioneel)
        if burning_sequence is not None:
            bestand.write(f"{burning_sequence}\n")
        else:
            bestand.write("\n")  # Lege regel

        # Genereer alle lijnen in de grid
        # Horizontale lijnen: (i, j) -- (i, j+1)
        for i in range(m):
            for j in range(n - 1):
                a = i * n + j + 1  # Knoopnummer (1-based)
                b = i * n + j + 2
                bestand.write(f"{a} {b}\n")

        # Verticale lijnen: (i, j) -- (i+1, j)
        for i in range(m - 1):
            for j in range(n):
                a = i * n + j + 1
                b = (i + 1) * n + j + 1
                bestand.write(f"{a} {b}\n")

        # Diagonale lijnen (schuin naar beneden): (i, j) -- (i+1, j+1)
        for i in range(m - 1):
            for j in range(n-1):
                a = i * n + j + 1
                b = (i + 1) * n + j + 2
                bestand.write(f"{a} {b}\n")

        # Diagonale lijnen (schuin naar boven): (i, j) -- (i+1, j-1)
        for i in range(1, m):
            for j in range(n - 1):
                b = i * n + j + 1
                a = (i-1) * n + j+2
                bestand.write(f"{a} {b}\n")

    print(f"Bestand 'king{m}x{n}.txt' is gegenereerd.")
```

D

Generating cube grid graph files

This code generates a file that describes an $l \times m \times n$ cube grid graph ($P_l \square P_m \square P_n$) in the format required to use the ILP [8].

```python
def genereer_kubus_bestand(k, m, n, burning_sequence=None):
    # Bereken het aantal knopen en lijnen
    aantal_knopen = k * m * n
    aantal_lijnen = 3 * k * m * n - m * n - k * n - k * m

    # Open het bestand om te schrijven
    with open(f"kubus_{k}x{m}x{n}.txt", "w") as bestand:
        # Schrijf het aantal knopen
        bestand.write(f"{aantal_knopen}\n")

        # Schrijf het aantal lijnen
        bestand.write(f"{aantal_lijnen}\n")

        # Schrijf de burning sequence (optioneel)
        if burning_sequence is not None:
            bestand.write(f"{burning_sequence}\n")
        else:
            bestand.write("\n")  # Lege regel

        # Genereer lijnen in de X-richting (binnen elke laag)
        for l in range(k):
            for i in range(m):
                for j in range(n - 1):
                    a = l * m * n + i * n + j + 1
                    b = l * m * n + i * n + j + 2
                    bestand.write(f"{a} {b}\n")

        # Genereer lijnen in de Y-richting (binnen elke laag)
        for l in range(k):
            for i in range(m - 1):
                for j in range(n):
                    a = l * m * n + i * n + j + 1
                    b = l * m * n + (i + 1) * n + j + 1
                    bestand.write(f"{a} {b}\n")

        # Genereer lijnen in de Z-richting (tussen lagen)
        for l in range(k - 1):
            for i in range(m):
                for j in range(n):
                    a = l * m * n + i * n + j + 1
                    b = (l + 1) * m * n + i * n + j + 1
                    bestand.write(f"{a} {b}\n")

    print(f"Bestand 'kubus_{k}x{m}x{n}.txt' is gegenereerd.")
```

E

Generating king's cube graph files

This code generates a file that describes an $l \times m \times n$ cube king's graph ($P_l \boxtimes P_m \boxtimes P_n$) in the format required to use the ILP [8].

```
def genereer_king_cube_bestand(k, m, n, burning_sequence=None):
    # Bereken het aantal knopen en lijnen
    aantal_knopen = k * m * n
    aantal_lijnen = 3 * k * m * n - m * n - k * n - k * m + 2*k*(m-1)*(n-1) + 2*m*(k-1)
    *(n-1) + 2*n*(k-1)*(m-1) + 4*(k-1)*(m-1)*(n-1)

    # Open het bestand om te schrijven
    with open(f"king_cube_{k}x{m}x{n}.txt", "w") as bestand:
        # Schrijf het aantal knopen
        bestand.write(f"{aantal_knopen}\n")

        # Schrijf het aantal lijnen
        bestand.write(f"{aantal_lijnen}\n")

        # Schrijf de burning sequence (optioneel)
        if burning_sequence is not None:
            bestand.write(f"{burning_sequence}\n")
        else:
            bestand.write("\n")  # Lege regel

        # Genereer lijnen in de X-richting (binnen elke laag)
        for l in range(k):
            for i in range(m):
                for j in range(n - 1):
                    a = l * m * n + i * n + j + 1
                    b = l * m * n + i * n + j + 2
                    bestand.write(f"{a} {b}\n")

        # Genereer lijnen in de Y-richting (binnen elke laag)
        for l in range(k):
            for i in range(m - 1):
                for j in range(n):
                    a = l * m * n + i * n + j + 1
                    b = l * m * n + (i + 1) * n + j + 1
                    bestand.write(f"{a} {b}\n")

        # Genereer lijnen in de Z-richting (tussen lagen)
        for l in range(k - 1):
            for i in range(m):
                for j in range(n):
                    a = l * m * n + i * n + j + 1
                    b = (l + 1) * m * n + i * n + j + 1
                    bestand.write(f"{a} {b}\n")

        #genereer diagonalen binnen de lagen (l constant)
        for l in range(k):
            for i in range(m-1):
```

```
                for j in range(n-1):
                    a = l * m * n + i * n + j + 1
                    b = l * m * n + (i + 1) * n + j + 2
                    bestand.write(f"{a} {b}\n")
            for i in range(1, m):
                for j in range(n - 1):
                    b = l * m * n + i * n + j + 1
                    a = l * m * n + (i - 1) * n + j + 2
                    bestand.write(f"{a} {b}\n")

        # genereer diagonalen binnen de lagen (m constant)
        for i in range(m):
            for l in range(k - 1):
                for j in range(n - 1):
                    a = l * m * n + i * n + j + 1
                    b = (l + 1) * m * n + i * n + j + 2
                    bestand.write(f"{a} {b}\n")
            for l in range(1, k):
                for j in range(n - 1):
                    b = l * m * n + i * n + j + 1
                    a = (l - 1) * m * n + i * n + j + 2
                    bestand.write(f"{a} {b}\n")

        # genereer diagonalen binnen de lagen (n constant)
        for j in range(n):
            for l in range(k - 1):
                for i in range(m - 1):
                    a = l * m * n + i * n + j + 1
                    b = (l + 1) * m * n + (i + 1) * n + j + 1
                    bestand.write(f"{a} {b}\n")
            for l in range(1, k):
                for i in range(m - 1):
                    b = l * m * n + i * n + j + 1
                    a = (l - 1) * m * n + (i + 1) * n + j + 1
                    bestand.write(f"{a} {b}\n")

        # genereer diagonalen over alle richtingen
        for l in range(k-1):
            for i in range(m-1): #+1,+1
                for j in range(n-1):
                    a = l * m * n + i * n + j + 1
                    b = (l + 1) * m * n + (i + 1) * n + j + 2
                    bestand.write(f"{a} {b}\n")
                for j in range(1,n):
                    a = l * m * n + i * n + j + 1
                    b = (l + 1) * m * n + (i + 1) * n + j
                    bestand.write(f"{a} {b}\n")
            for i in range(1,m):
                for j in range(n-1):
                    a = l * m * n + i * n + j + 1
                    b = (l + 1) * m * n + (i - 1) * n + j + 2
                    bestand.write(f"{a} {b}\n")
                for j in range(1,n):
                    a = l * m * n + i * n + j + 1
                    b = (l + 1) * m * n + (i - 1) * n + j
                    bestand.write(f"{a} {b}\n")
    print(f"Bestand 'king_cube_{k}x{m}x{n}.txt' is gegenereerd.")
```

F

Graph checklist

This code generates a list of lists containing file names, number of vertices, number of edges, burning number lower bound and burning number upper bound for a specified graph type based on dimensions given by the user. This list can be used to calculate burning numbers for the graphs using the ILP described in Section 3.3.

```
import math

def checklist_grid(dimensions):
    with open(f"checklist_grid.txt", "w") as bestand:
        for [m,n] in dimensions:
            bestand.write(f"['grid{m}x{n}.txt',{m*n},{2 * m * n - m - n},{math.ceil(
    math.sqrt(n))},{math.ceil(math.sqrt(m*n))+1}],\n")
    print(f"File 'checklist_grid.txt' is generated.")

def checklist_king(dimensions):
    with open(f"checklist_king.txt", "w") as bestand:
        for [m,n] in dimensions:
            bestand.write(f"['king{m}x{n}.txt',{m*n},{2 * m * n - m - n +2*(m-1)*(n-1)
    },{math.ceil(math.sqrt(n))},{math.ceil(math.sqrt(m*n))+1}],\n")
    print(f"File 'checklist_grid.txt' is generated.")

def checklist_king_cube(dimensions):
    ## Uses upper and lower bounds from theorems
    with open(f"checklist_king_cube.txt", "w") as bestand:
        for [k,m,n] in dimensions:
            edges = 3 * k * m * n - m * n - k * n - k * m + 2*k*(m-1)*(n-1) + 2*m*(k-1)
    *(n-1) + 2*n*(k-1)*(m-1) + 4*(k-1)*(m-1)*(n-1)
            lower_bound = math.ceil((0.5*k*m*n)**(1/4))
            upper_bound = min(math.ceil(math.sqrt(m*n)),math.ceil(0.5*(n+1)))
            bestand.write(f"['king_cube_{k}x{m}x{n}.txt',{k*m*n},{edges},{lower_bound
    },{upper_bound}],\n")
    print(f"File 'checklist_king_cube.txt' is generated.")

def checklist_torus_grid(dimensions):
    with open(f"checklist_torus.txt", "w") as bestand:
        for [m,n] in dimensions:
            aantal_knopen = m * n
            aantal_lijnen = 2 * m * n
            if m < 3:
                aantal_lijnen -= n
            if n < 3:
                aantal_lijnen -= m
            bestand.write(f"['torus_grid{m}x{n}.txt',{aantal_knopen},{aantal_lijnen},{
    math.ceil(math.sqrt(n))},{math.ceil(math.sqrt(m*n))+1}],\n")
    print(f"File 'checklist_grid.txt' is generated.")
```