Automating Weight Function Generation in Graph Pebbling

Dominic Flocco * Jonad Pulaj * Carl Yerger*

June 30, 2022

Abstract

Graph pebbling is a combinatorial game played on an undirected graph with an initial configuration of pebbles. A pebbling move consists of removing two pebbles from one vertex and placing one pebble on an adjacent vertex. The pebbling number of a graph is the smallest number of pebbles necessary such that, given any initial configuration of pebbles, at least one pebble can be moved to a specified root vertex. We present a computational framework that produces a set of tree strategy weight functions that are capable of proving pebbling number upper bounds on general graphs. Our mixed-integer linear programming approach automates the generation of large sets of such functions and provides verifiable certificates of pebbling number upper bounds. We apply the model to the 4^{th} weak Bruhat to prove $\pi(B_4) \leq 66$ and to the Lemke square graph to produce a set of certificates that verify $\pi(L \square L) \leq 96$.

1 Introduction

Graph pebbling is a combinatorial game played on an undirected graph with an initial configuration of pebbles. The game is composed of a sequence of pebbling moves that consist of removing two pebbles from one vertex, and placing one pebble on an adjacent vertex. The graph pebbling model was first introduced by Chung [4] and can be used to model transportation networks where resources are consumed in transit, such as the dissipation of energy, the flow of information through a particular medium or the consumption of fuel by oil tankers. Thus, graph pebbling may be viewed as a network optimization model that has applications to operations research, logistics and applied science. The central question of the model asks how many pebbles are necessary to ensure that, given any initial configuration of this many pebbles, one pebble can be moved to any particular vertex.

Conventional graph pebbling techniques employ topics in number theory [4, 9, 16, 19], combinatorics [7, 15] and graph theory [2, 3, 17]. Since its conceptualization in 1989 [4], there have been many interesting and useful results presented through these theoretical mediums, particularly on small graphs and families of graphs. In fact, the graph pebbling model was brought forth by Lagarias and Saks in response to a combinatorial

*Department of Mathematics and Computer Science, Davidson College, Davidson, NC 28036, {doflocco, jopulaj, cayerger}@davidson.edu

number theory conjecture on zero-sum problems by Erdos and Lemke. These methods are successful in defining properties that provide sufficient grounds for pebbling classifications, and producing general results with diverse application.

A recent line of inquiry in the graph pebbling community explores computational approaches to calculating pebbling bounds. Calculating pebbling numbers is difficult: Watson [20] shows determining whether a configuration of pebbles is solvable is NP-complete, and Clark and Milans [14] prove that determining whether the pebbling number is at most k is Π_2^P-complete. To address this complexity, we apply methods in linear and discrete optimization to the graph pebbling model, building off the work of [6], [11] and [12]. Namely, we use mixed-integer linear programming (MILP) to calculate pebbling bounds, building upon theoretical results and improving bounds on graphs of interest. Through exploring popular problems in graph pebbling literature, we develop a computational framework that can be applied to general networks. This computational framework is packaged into an open source graph pebbling codebase that includes the MILP implementations, verification and visualization tools and pebbling graph classifications. The suite is complete with a user-interface and documentation to allow for easy experimentation by future researchers.[1]

1.1 Notation & Terminology

Let $G = (V, E)$ be a simple, connected and undirected graph with vertex set $V = V(G)$ and edge set $E = E(G)$, with $|V| = n$ vertices and $|E| = m$ edges. We define $\text{dist}(x, y)$ to be the number of edges (length) of the shortest path from vertex x to y. The *eccentricity* $\text{ecc}(G, u)$ for a vertex $u \in V$ is given by $\text{ecc}(G, u) = \max_{v \in V} \text{dist}(v, u)$, and the *diameter* $\text{diam}(G)$ of G is then given by $\text{diam}(G) = \max_{u \in V} \text{ecc}(G, u)$. We will also consider the Cartesian product of two graphs, denoted $\square$, where the product of $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is defined by the vertex set $V(G_1 \square G_2) = \{(v_1, v_2) | v_1 \in V(G_1), v_2 \in V(G_2)\}$ and edge set $E(G_1 \square G_2) = \{\{(v_1, v_2), (u_1, u_2)\} | (v_1 = u_1 \text{ and } (v_2, u_2) \in E(G_2)) \text{ or } (v_2 = u_2 \text{ and } (v_1, u_1) \in E(G_1))\}$.

For a graph $G = (V, E)$, a *configuration* of pebbles is a function $C : V \to \mathbb{Z}_{\geq 0}$. Let $C(v)$ be the number of pebbles placed on vertex $v \in V$, and define the *size* of the configuration as $|C| = \sum_{v \in V} C(v)$, which is the number of pebbles placed on all vertices in G. A *pebbling move* from u to v along an edge $(u, v) \in E$ consists of removing two pebbles from vertex $u \in V$ and placing one pebble at adjacent vertex $v \in V$. We say that a pebble can be *moved* to the *root* vertex r if we can repeatedly apply pebbling steps to G such that r has one pebble in the resulting configuration.

Consider a graph G rooted at r with a configuration of pebbles C. We define the *rooted pebbling number* $\pi(G, r)$ as the smallest integer k such that for any configuration of $|C| = k$ pebbles onto the vertices of G, one pebble can be moved to the specified root vertex r. The *pebbling number* $\pi(G)$ is the smallest integer k such that for any configuration of k pebbles onto the vertices of G, one pebble can be moved to each $r \in V(G)$. Given a configuration of pebbles C on the vertices of G, if it is possible to move a pebble

[1] `https://github.com/dominicflocco/Graph_Pebbling`

to the root vertex r by a sequence of pebbling moves, then we say that C is *r-solvable*. If not, C is *r-unsolvable*. If C is r-solvable for each $r \in V$ then we say C is *solvable*, and *unsolvable* otherwise.

1.2 Results

The following paper explores the use of linear and mixed-integer programming to prove pebbling number bounds on a number of graphs of interest. We offer the theoretical framework, implementation and formal definition of two MILPs that build on the foundation set by Cranston et al. [6] and Hurlbert [11]. Our optimization problems generate larger, more useful sets of weight functions to improve pebbling bounds previously found with manually generated strategies. Namely, we present the Tree Strategy Optimization Problem $\mathbf{TS}_{G,r}$, which uses mixed-integer programming to generate sets of tree strategies that minimize the pebbling number bound. Then, we leverage the symmetry of tree strategies in Cartesian product graphs to reduce the number of constraints in the MILP in the Symmetric Tree Strategy Optimization Problem $\mathbf{STS}_{G,r}$. Together, these optimization problems produce useful weight functions that have the potential to solve a plethora of graph pebbling models and have proven successful in improving pebbling bounds on a number of graphs of interest in the literature, such as the Lemke square $L \square L$ and 4^{th} weak Bruhat B_4.

The optimization problems presented in our paper show significant promise in improving pebbling bounds on a number of important families of graphs. Most notably, we prove that $\pi(L \square L) \leq 96$, improving the upper bound of 108 found in [11]. Kenter [12, 13] uses an integer programming approach to numerically suggest evidence that $\pi(L \square L) \leq 91$ and then improves this bound to 85. While these results are impressive, our mixed-integer programming framework not only provides certificates that can be checked by hand but also bypasses the chance of error imposed by floating-point arithmetic possible in Kenter's approach, a problem that Cook [5] shows possible. Additionally, the tree strategy visualizations generated by our MILP approach can provide intuition into the underlying structures of the graphs and insight into their pebbling properties. We also prove that $\pi(B_4) \leq 66$ using only tree strategies, further improving the results of Hurlbert [11], and confirming the results of Cranston et al. [6] obtained using non-tree weight functions.

We begin the paper by presenting the necessary graph pebbling theoretical framework upon which our mixed-integer programs rely in Section 2. Then we offer a formal definition of the MILP framework and provide a theoretical discussion on the model in Section 3. Lastly, we present important and novel results obtained through these techniques in Section 4 before providing further questions to explore in the research in Section 5.

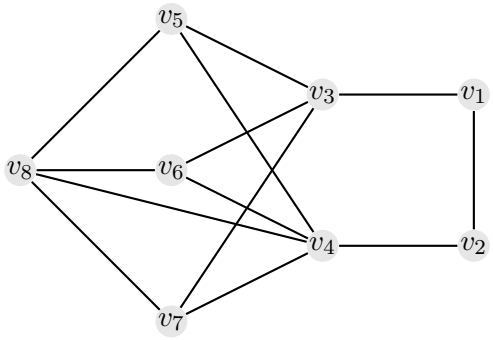


Figure 1: Original Lemke graph L on $n = 8$ vertices.

2 Background

2.1 Graph Products

In the initial formalization of the graph pebbling problem in Chung [4] the reader is left with a conjecture by Ronald Graham. Since its proposition, Graham's conjecture has been of great interest to the pebbling community.

Conjecture 2.1 (Graham)**.** *For all graphs G and H, we have $\pi(G\Box H) \leq \pi(G)\pi(H)$.*

While many useful results in support of this conjecture have been offered, the conjecture has yet to be proven for all graphs. The conjecture is verified when G and H are trees [15], cycles [9], a clique and a graph with the 2-pebbling property [4] and when G has the 2-pebbling property and H is a complete multi-partite graph [19]. Feng [8] also verified the conjecture for complete bipartite graphs and wheels or fans. Later, Hurlbert [10] proves Graham's conjecture for a more general product. While these results are encouraging, there is evidence to suggest that the Lemke square graph L may provide a counterexample to the conjecture, and computational methods could be key to finding a counterexample. Note that $\pi(L, r) = 8$ for all $r \in V(L)$, so $\pi(L)\pi(L) = 64$; thus, if one can show $\pi(L\Box L) > 64$; Graham's conjecture will be disproved. As a result, the following work focuses on the application of the MILP computational approaches to weight function generation to the Cartesian product of graphs.

2.2 Computational Approaches

Due to the difficulty of calculating tight pebbling bounds by hand or with theory, the pebbling literature has seen a recent rise in computational pursuits. The algorithmic approach has been successful in computing pebbling numbers and bounds on graphs with useful characteristics. Bekmetjev and Cusack [1] explore algorithms to determine the solvabilility of pebbling configurations on graphs with diameter two, proving the existence of an algorithm whose running time depends on the vertex connectivity and

the size of the graph. Sieben [18] extends this approach and presents an algorithm that computes the pebbling number on all connected graphs with up to 9 vertices with diameter 2. The implementation is sufficiently fast on a large number of small graphs, yet does not scale well as the number of vertices increases. While the algorithmic approach has proven successful for small graphs with certain properties, other promising computational approaches have recently appeared in the literature.

Most recently, Kenter [12, 13] presents an integer-programming (IP) approach to bounding $\pi(L\Box L)$ and Cartesian products more generally. This approach is specific to $L\Box L$, yet can be expanded to provide bounds on the pebbling numbers of different types of graph products. The model leverages the 2-pebbling property of $L\Box L$ to refine a more general IP approach that offers numerical evidence to suggest $\pi(L\Box L) \leq 85$. In contrast, our MILP approach chooses an ideal set of the exponentially many subtrees in a graph to generate weight functions that offer verifiable certificates of pebbling bounds.

2.3 Weight Functions

Linear optimization techniques have proven successful in improving pebbling number bounds on large graphs and represent a state of the art computational approach to pebbling. Hurlbert [11] first introduces a linear optimization problem to compute upper bounds on pebbling numbers and lays the foundation for the following investigation. The foundational tool used in the linear optimization problem is the Weight Function Lemma 2.1.

Let G be a graph and $\mathcal{T}$ be a subtree of G rooted at r. For each vertex $i \in V(\mathcal{T})\backslash\{r\}$, let i^+ denote the *parent* of i, where $\text{dist}(i^+, r) = \text{dist}(i, r) - 1$.

Definition 2.1 (Tree Strategy)**.** *Let $G = (V, E)$ be an undirected graph with root r. We call $\mathcal{T} = (V, E, w)$, with vertex set $V(\mathcal{T})$, edge set $E(\mathcal{T})$ and nonnegative weight function $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$, a valid tree strategy when w satisfies the following three properties: (1) $w(r) = 0$, (2) $w(i^+) \geq 2w(i)$ for all $i \in V(\mathcal{T})\backslash\{r\}$ that are not adjacent to r, and (3) $w(i) = 0$ if $i \notin V(\mathcal{T})$.*

Note that the weight function w may also be viewed as a vector $w \in \mathbb{R}^n_{\geq 0}$ representing the $n = |V(G)|$ weights of the tree strategy, where w_i corresponds to the weight on v_i. We can also define a configuration of pebbles $C \in \mathbb{Z}^n_{\geq 0}$ in a similar manner and let $\mathbf{1}_G$ be the vector on $V(G)$ in which every entry is 1. The Weight Function Lemma 2.1 allows us to consider all subtrees rooted at r simultaneously by considering the conic combination of strategies. We will see that the ability to consider all subtrees simultaneously provides a more powerful and versatile approach to pebbling upper bound calculation.

Lemma 2.1 (Weight Function Lemma)**.** *[11] Let $\mathcal{T}$ be a tree strategy of G rooted at r, with associated weight function w. If C is an r-unsolvable configuration of pebbles on $V(G)$, then $w \cdot C \leq w \cdot \mathbf{1}_G$.*

The proof idea is rather intuitive. If we suppose that C is a configuration such that $w \cdot C > w \cdot \mathbf{1}_G$, there must be some vertex v in T that has at least two pebbles.

Now, make a pebbling move from v towards the root to v^+ and denote the resulting configuration C'. By the definition of a weight function w, we have $w(v^+) = 2w(v)$, so $w \cdot C' = w \cdot C > w \cdot \mathbf{1}_G$. Repeating this process we can eventually move a pebble to root r. It follows that for every graph G and root vertex r, the pebbling number is bounded above by one plus the number of pebbles in the largest configuration C such that $w \cdot C \leq w \cdot \mathbf{1}_G$.

2.4 Linear Programming Approach

The Weight Function Lemma 2.1 allows the definition of a linear optimization problem capable of computing bounds on large graphs and opens the door to a new computational approach with the potential of generating impressive results on many graphs of interest. Suppose we have T tree strategies with indices $1, 2, \ldots, T$ in graph G rooted at r. We now arrive at the following integer linear program first offered by Hurlbert [11]:

$$\text{Maximize: } \sum_{v \in V \setminus \{r\}} C(v) \tag{2.1}$$

$$\text{Subject to: } w_t \cdot C \leq w_t \cdot \mathbf{1}_G \quad 1 \leq t \leq T. \tag{2.2}$$

Let $z_{G,r}$ be the optimal value of this integer linear optimization problem and let $\hat{z}_{G,r}$ be the optimum of the linear relaxation. Thus, $z_{G,r} \leq \lfloor \hat{z}_{G,r} \rfloor$ and we obtain $\pi(G, r) \leq z_{G,r} + 1 \leq \lfloor \hat{z}_{G,r} \rfloor + 1$. Therefore, by producing such a configuration, the linear program above can be used to prove the upper bound on pebbling numbers with intelligently designed weight functions. If we let w' be a conic combination of the associated weight functions for each of the T tree strategies $w_1, \ldots, w_T$, then we can generalize this idea to obtain the following Lemma.

Lemma 2.2 (Covering Lemma)**.** *[6] Given a graph G rooted at $r \in V(G)$, let w' be a conic combination of tree strategies for r, and let C and M be positive constants. If $w'(v) \geq C$ for all $v \in V(G) \setminus \{r\}$ and $\sum_{v \in V(G) \setminus \{r\}} w'(v) < M$, then $\pi(G, r) \leq \left\lfloor \frac{M}{C} \right\rfloor + 1$.*

Since w' is a conic combination of tree strategies, we can think of the constant C as the minimum weight on a vertex $v \in V(G) \setminus \{r\}$ across all tree strategies on the graph G rooted at r. The constant M is then the sum of weights on vertices for all tree strategies plus one. Then, the value $\left\lfloor \frac{M}{C} \right\rfloor$ can be interpreted as the average tree weight across all strategies under the assumption that the vertex with minimum weight has average weight of one. The difficulty in this task lies choosing a small subset of the potentially exponential subtrees of a graph to generate the best possible pebbling bound. The approach to this problem is unclear and will be the focus of our work.

Hurlbert [11] uses this approach to obtain results on many families of graphs, namely trees, cycles, cubes, Petersen, Lemke and 4^{th} weak Bruhat. Cranston et al. [6] build upon this technique and generalize the set of weight functions beyond trees. This generalization proves successful in improving the bounds on the Lemke graph, 3-cube and 4^{th} weak Bruhat graph from [11]. The Weight Function Lemma 2.1, Covering Lemma 2.2,

and the accompanied linear optimization problem 2.1-2.2 introduced and explored in [11] and [6] prove successful in computing pebbling bounds in reasonable time and on large graphs. In our research, we explore and extend this optimization approach and further exhibit its versatility and strengths by automating the generation of weight functions.

3 Mixed-Integer Linear Programming Approach

We introduce a tree strategy optimization problem to automate the task of tree generation on a given graph G rooted at r. As there are an exponential number of subtrees in G, our task is to choose an *ideal* set of strategies that produce low pebbling bounds using the Covering Lemma 2.2. To do so, we fix the number of tree strategies T as a parameter to the linear optimization problem and index tree strategies from the ordered set $\{1, 2, \ldots, T\} = [T]$. Further, we define parameter ℓ to be an upper bound on the tree length, formally defined as $\ell = \max_{v \in \mathcal{T}} \text{dist}(r, v)$. We wish to minimize the average tree weight to obtain a small pebbling number bound. Thus, we seek a set of tree strategies that minimize the average tree weight M while requiring $C \geq T$. Hence, we assume that that the vertex with minimum weight has average weight of one.

3.1 Decision Variables

We begin by specifying an arbitrary root vertex $r \in V(G)$ and transforming G to a bidirected graph G' to capture child-parent vertex relationships. Specifically, for all undirected edges $(i, j) \in E(G)$, add directed edges $(i, j) \in E(G')$ and $(j, i) \in E(G')$ to G'. Let V be the set of vertices in G' and A be the set of directed arcs in G, where a cycle of length two from node i back to node i through j is an arc, denoted $(i, j) \in A$ with $|A(G)| = m$. Next, define a set of integer decision variables $\mathcal{X} \in \{0, 1\}^{m \times T}$ such that $X^t \in \{0, 1\}^m$ represents the arcs of G in strategy t:

$$x_{i,j}^t = \begin{cases} 1 & \text{if } (i, j) \in E(t) \\ 0 & \text{if } (i, j) \notin E(t) \end{cases} \tag{3.1}$$

where $t \in [T]$ is a tree strategy and $(i, j) \in A$ is an arc in G'. Further, define a set of integer decision variables $\mathcal{Y} \in \{0, 1\}^{n \times T}$ such that $Y^t \in \{0, 1\}^n$ represents the vertices of G in strategy t:

$$y_i^t = \begin{cases} 1 & \text{if } i \in V(t) \backslash \{r\} \\ 0 & \text{if } i \notin V(t) \backslash \{r\} \end{cases} \tag{3.2}$$

Lastly, define a set of continuous decision variables $\mathcal{Z} \in \mathbb{R}_{\geq 0}^{T \times n}$ where $Z^t \in \mathbb{R}_{\geq 0}^n$ represents the weights on the vertices of G in strategy t. We denote z_i^t as the weight of vertex $i \in V$ in strategy t, where $0 \leq z_i^t \leq 2^{\ell - 1}$ for all $i \in V(G)$.

3.2 Linear Optimization Problem

We propose the following Tree Strategy Optimization problem $\mathbf{TS}_{G,r}$ that takes in a fixed number of tree strategies T and maximum tree length ℓ as input, and outputs a set of T strategies:

$$\text{Minimize: } \sum_{t=1}^{T} \sum_{i \in V(t)} z_i^t \tag{3.3}$$

$$\text{Subject to: } \sum_{j \in V(t):(j,i) \in A} x_{j,i}^t = y_i^t \qquad 1 \le t \le T \text{ and } \forall i \in V(G) \tag{3.4}$$

$$\sum_{i \in V(G):(r,i) \in A} y_i^t \ge 1 \qquad 1 \le t \le T \tag{3.5}$$

$$\sum_{t=1}^{T} z_i^t \ge T \qquad \forall i \in V(G) : i \ne r \tag{3.6}$$

$$z_i^t - 2z_j^t + 2^\ell (1 - x_{i,j}^t) \ge 0 \qquad 1 \le t \le T \text{ and } \forall (i,j) \in A(G) : i, j \ne r \tag{3.7}$$

$$z_i^t \le 2^{\ell-1} \cdot y_i^t \qquad 1 \le t \le T \text{ and } \forall i \in V(G) \tag{3.8}$$

$$x_{i,j}^t \in \{0,1\} \qquad 1 \le t \le T \text{ and } \forall (i,j) \in A(G) \tag{3.9}$$

$$y_i^t \in \{0,1\} \qquad 1 \le t \le T \text{ and } \forall i \in V(G) \tag{3.10}$$

$$z_i^t \in \mathbb{R}_{\ge 0} \qquad 1 \le t \le T \text{ and } \forall i \in V(G) \tag{3.11}$$

The goal of the optimization is to produce a set of tree strategies that minimizes the average tree weight. The tree strategies will be structured such that they bound pebbling numbers using the Covering Lemma 2.2 and the optimization method outlined in [11] and defined in Equations 2.1-2.2. These constraints and bounds can be modified and enhanced to generate a more complex set of strategies useful for larger, more complex graphs. Automating the generation of these strategies through the approach defined above will improve the application of the pebbling optimization problems introduced in [6] and [11], and have the potential to produce useful pebbling bounds on a wide range of graphs.

The first two constraints define structural constraints on the set of tree strategies. Constraint 3.4 ensures the subtree is acyclic and connected, and all edges in the tree strategy are directed away from the root vertex r. Further, constraint 3.5 enforces that each tree strategy is rooted at r. Along with the input parameter ℓ, which places an upper bound on each strategy's tree length, these two characteristics define the structure of tree strategies we wish to generate.

The next set of constraints enforce properties of the associated nonnegative weight function w given in Definition 2.1. Constraint 3.7 ensures that for each vertex $i \in V(t)$, $w(i) \ge 2w(i^+)$, which fits the definition of a *non-basic* tree strategy introduced in Hurlbert [11]. Additionally, we place an upper bound of $2^{\ell-1}$ on each weight with constraint 3.8. Lastly, we fix the minimum weight on each vertex, which is represented by constant C in Lemma 2.2, by constraint 3.6. This constant is fixed by the number

of tree strategies T; therefore, by minimizing the total weight M by objective function, $\mathbf{TS}_{G,r}$ generates tree strategies that minimize the pebbling bound $\pi(G,r) \leq \lfloor \frac{M}{C} \rfloor + 1$.

3.3 Theoretical Discussion

We would like to prove that the tree strategies generated by $\mathbf{TS}_{G,r}$ produces useful tree strategies that fit the definition and structure of tree strategies. Along these lines, we present proof that shows the program outputs the desired results while minimizing the necessary variables for pebbling bounds. Definition 2.1 characterises the necessary properties of tree strategies and will guide our theoretical discussion. If a tree strategy $\mathcal{T}$ satisfies constraints 3.4-3.8, we say the strategy is *feasible*. We will argue that all feasible strategies satisfy Definition 2.1.

Additionally, in $\mathbf{TS}_{G,r}$ we seek strategies that are useful in computing low pebbling bounds. Given a graph G rooted at r, we calculate an upper bound on $\pi(G,r)$ by the Covering Lemma 2.2, defining constants C and M and bounding the pebbling number by $\pi(G,r) \leq \lfloor \frac{M}{C} \rfloor + 1$. We fix the minimum vertex weight C to be the number of strategies T by constraint 3.6 and minimize the total weight M by the objective function. More specifically, let $\hat{z}$ be the objective value produced by a feasible set of T strategies. We then calculate bounds by

$$\pi(G,r) \leq \left\lfloor \frac{M}{C} \right\rfloor + 1 = \left\lfloor \frac{\hat{z}}{T} \right\rfloor + 1. \tag{3.12}$$

Now, we offer proof that an arbitrary feasible tree strategy $\mathcal{T}$ produced by $\mathbf{TS}_{G,r}$ satisfies Definition 2.1 and thereby corresponds to a valid tree strategy.

Lemma 3.1. *If $\mathcal{T}$ is a feasible tree strategy generated by $\mathbf{TS}_{G,r}$, then $\mathcal{T}$ is acyclic, connected and rooted at r.*

Proof. Let $\mathcal{T} = (V, E, w)$ be an arbitrary feasible tree strategy generated by $\mathbf{TS}_{G,r}$. First, $\mathcal{T}$ is rooted at r by constraint 3.5. We will prove that $\mathcal{T}$ is a connected graph by induction on $|V(\mathcal{T})\backslash\{r\}| = n$. For the base case, we consider when $n = 1$. Let $v_1 \in V(\mathcal{T})\backslash\{r\}$ be this vertex. By definition, $y^t_{v_1} = 1$ and by constraint 3.5 there must exist a directed edge $(r, v_1) \in E(\mathcal{T}_t)$. Thus, when $n = 1$, the tree strategy t is connected. Now, suppose that the feasible tree strategy t is connected with $k < n$ vertices. Consider the vertex set $V(\mathcal{T}) \cup \{v'\}$ for some $v \in V(G)$. If we add v' to $V(\mathcal{T})$, $y^t_{v'} = 1$ by definition. Then, constraint 3.4 implies that there exists one $j \in V(\mathcal{T})$ such that $x^t_{j,v'} = 1$ so $(j, v') \in E(\mathcal{T}_t)$. By the inductive hypothesis, $j \in V(\mathcal{T})$ is a part of the connected tree strategy t with $k < n$ vertices. Since, $(j, v') \in E(\mathcal{T})$, $V(\mathcal{T}) \cup \{v'\}$ is connected. Therefore, by induction, all feasible tree strategies are connected.

Next, we will show that the generated tree strategies are acyclic. For contradiction, $\mathcal{T}$ contains a cycle. This means that there exists a vertex $v \neq r$ with directed edges $(u, v), (v, w) \in E(\mathcal{T})$ for some $u, w \in V(\mathcal{T})$. By definition, this implies that $y^t_v = 1$, $x^t_{u,v} = 1$ and $x^t_{v,w} = 1$. Since $v \neq r$ and $\mathcal{T}$ is connected, there must exist another vertex incident to v, we will call this vertex $k \in V(\mathcal{T})$ with directed edge $(k, v) \in E(\mathcal{T})$. By the

definition of our binary variables, $y_k^t = 1$ and $x_{k,v}^t = 1$. Since $\mathcal{T}$ is feasible, it satisfies constraint 3.4 which would imply

$$\sum_{j \in V(G):(j,v) \in A} x_{j,v}^t \geq x_{k,v}^t + x_{u,v}^t = 2.$$

However, we also have $y_v^t = 1$. Since the sum is lower bounded by 2, constraint 3.4 does not hold, thus $\mathcal{T}$ is not feasible. Hence, by contradiction, all feasible tree strategies are acyclic.

We have shown that $\mathcal{T}$ is acyclic, connected and rooted at r. Therefore, we conclude that all feasible strategies produced by $\mathbf{TS}_{G,r}$ satisfy the structural constraints of trees. □

We have shown that feasible tree strategies generated by $\mathbf{TS}_{G,r}$ satisfy the structural conditions of a subtree given in Definition 2.1. Next, we will prove that feasible tree strategies have valid associated weight functions.

Lemma 3.2. *If $\mathcal{T}$ is a feasible tree strategy generated by $\mathbf{TS}_{G,r}$, then $\mathcal{T}$ has a valid associated weight functions.*

Proof. Let $\mathcal{T} = (V, E, w)$ be an arbitrary feasible tree strategy generated by $\mathbf{TS}_{G,r}$ with index t. Let $v \in V(\mathcal{T})$ be an arbitrary vertex in the tree strategy $\mathcal{T}$ with parent $v^+ = u \in V(\mathcal{T})$, such that $(u, v) \in E(\mathcal{T})$. By definition, $y_v^t = 1$, $y_u^t = 1$ and $x_{u,v}^t = 1$. Let $z_v^t = w(v)$ and $z_u^t = w(u)$, where $0 \leq z_v^t \leq 2^{\ell-1}$ and $0 \leq z_u^t \leq 2^{\ell-1}$ by the bound constraint on z. Then, constraint 3.7 says

$$\begin{aligned} z_u^t - 2z_v^t + 2^\ell(1 - x_{v,u}^t) = z_u^t - 2z_v^t \geq 0 \\ z_u^t \geq 2z_v^t. \end{aligned}$$

Therefore, $w(v) \geq 2w(v^+)$ for all $v \in V(\mathcal{T})$.

Similarly, we must show that $w(v) = 0$ for all $v \notin V(\mathcal{T})$. Let $v \notin V(\mathcal{T})$ be an arbitrary vertex not in the tree strategy $\mathcal{T}$. By definition, $y_v^t = 0$ since $v \notin V(\mathcal{T})$. Furthermore, $x_{i,v}^t = 0$ and $x_{v,i}^t = 0$ for all $i \in V(G)$. By constraint 3.8, we have $z_v^t \leq 2^{\ell-1} y_v^t \Rightarrow z_v^t \leq 0$. Recall that we have the bound $0 \leq z_i^t \leq 2^{\ell-1}$ for all $i \in V(G)$. Hence, $z_v^t = 0$ as desired. Lastly, we must have that $w(r) = 0$. Let $r \in V(G)$ be the specified root vertex. By definition, $y_r^t = 0$, so the same logic implies that $w(r) = 0$. As such, the tree strategies produced by $\mathbf{TS}_{G,r}$ have valid associated weight functions. □

Combining the results from Lemma 3.2 and Lemma 3.1, we conclude that all feasible tree strategies generated by the linear program satisfy Definition 2.1. More specifically, feasible strategies have valid associated weight functions and are trees rooted at r. Lastly, we consider the usefulness of the tree strategies generated from the optimization problem. Our goal is to calculate tight pebbling bounds by leveraging the objective function to produce feasible strategies that drive down the pebbling bound. If $\mathcal{T}^*$ is an optimal solution to $\mathbf{TS}_{G,r}$, with objective value z^*, then $\mathcal{T}^*$ is feasible and has the property that

$\hat{z}^* \leq \hat{z}$, where $\hat{z}$ is the objective value of any arbitrary feasible solution. Recall that the objective of $\mathbf{TS}_{G,r}$ is to minimize the total weight M across all strategies. Since constraint 3.6 fixes $C = T$, we can apply the Covering Lemma 2.2 and use Equation 3.12 to obtain the pebbling lower bound. Therefore, $\mathbf{TS}_{G,r}$ seeks to generate feasible strategies that minimize pebbling bounds.

3.4 Symmetric Tree Strategy Optimization Problem

The benefit of the linear programming approach to tree strategy generation is that it allows us to explore large sets of tree strategies, far more than could reasonably be generated by hand. To increase the number of tree strategies generated by $\mathbf{TS}_{G,r}$, we increase the input parameter T, which increases the number of constraints, increasing the runtime complexity of the optimization solver. To enable the optimization solver to produce a large set of tree strategies for large graphs efficiently, we introduce a modification to $\mathbf{TS}_{G,r}$ that utilizes symmetric tree strategies. For this exploration, we consider Cartesian products of graphs, namely the Lemke square $L \square L$.

Let $G = (V, E)$ be a graph rooted at r and $G \square G$ be its Cartesian product defined in Section 2. Consider a set of tree strategies $\mathcal{T}_r$ generated by $\mathbf{TS}_{G\square G,r}$ for the graph $G \square G$ rooted at r. By Lemma 2.1 and Lemma 3.1, $\mathcal{T}$ is a valid tree strategies that satisfies Definition 2.1. We introduce symmetric tree strategies to modify the linear integer program to improve optimization performance.

Definition 3.1 (Symmetric Tree Strategy)**.** *Let $\mathcal{T} = (V, E, w)$ be an arbitrary, valid tree strategy generated by $\mathbf{TS}_{G\square G,r}$ for some Cartesian product $G \square G$ with root $r = (r_1, r_2)$. A tree strategy $\mathcal{T}' = (V', E', w')$ with root $r' = (r_2, r_1)$ is said to be symmetric to $\mathcal{T}$ if for all $(v, u) \in V \backslash \{r\}$ and all $((v, u), (x, y)) \in E$, we have $(u, v) \in V' \backslash \{r\}$ and $((u, v), (y, x)) \in E'$. Further, for all $(v, u) \in V$, $w(v, u) = w'(u, v)$.*

Suppose we wish to generate T tree strategies for a graph $G \square G$, where T is even. Utilizing symmetric pairs of trees, we may generate $T/2$ strategies with $\mathbf{TS}_{G\square G,r}$ and find the corresponding symmetric pair of the trees in this set to generate T total strategies. Evidently, this allows us to generate large amounts of tree strategies at roughly half the computational cost. Before introducing the modified optimization problem that leverages symmetry, we show that the corresponding symmetric tree strategy $\mathcal{T}'$ for a valid tree strategy $\mathcal{T}$ is also valid.

Theorem 3.1. *If $\mathcal{T}$ is an arbitrary valid tree strategy generated by $\mathbf{TS}_{G\square G,r}$, then the corresponding symmetric tree $\mathcal{T}'$ is also valid.*

Proof. Let $\mathcal{T} = (V, E, w)$ be a valid tree strategy of a graph $G \square G$ rooted at $r = (r_1, r_2)$. Consider the symmetric tree $\mathcal{T}' = (V', E', w')$ such that for all $(v, u) \in V$ and all $((v, u), (x, y)) \in E$, $(u, v) \in V' \backslash \{r\}$ and $((u, v), (y, x)) \in E'$. Further, for all $(v, u) \in V \backslash \{r\}$, $w(v, u) = w'(u, v)$. To show that $\mathcal{T}'$ is a valid tree strategy, we must show that $\mathcal{T}'$ is acyclic, connected and rooted at $r' = (r_1, r_2)$, and that w' is a valid weight function. First, by Definition 3.1, $\mathcal{T}'$ must be rooted at $r' = (r_1, r_2)$. Next, suppose that there

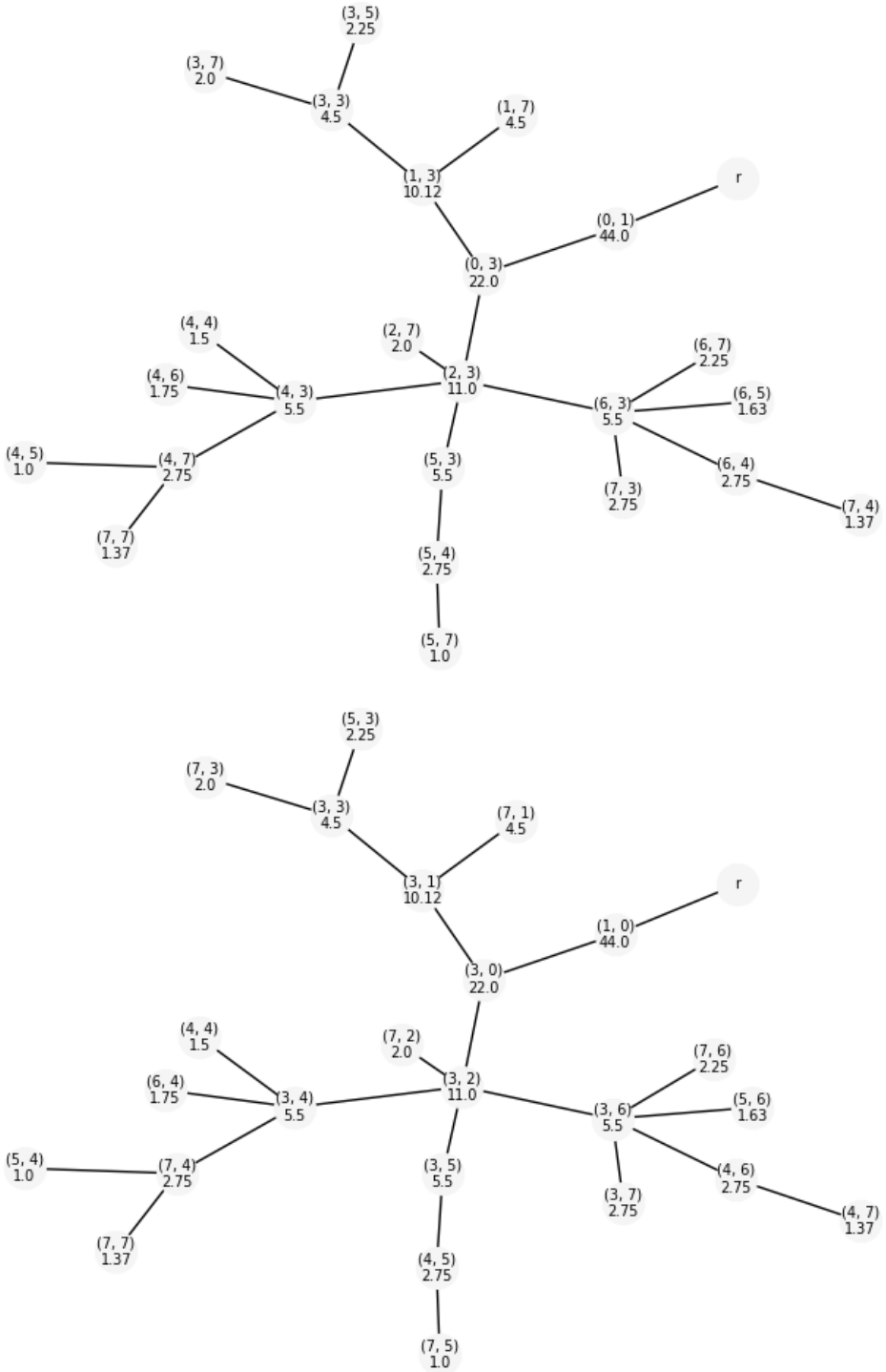


Figure 2: Example of two symmetric tree strategies generated by $\mathbf{STS}_{G,r}$ on $L \square L$ with root $r = (v_1, v_1)$. Labels (i, j) on top of each node correspond to vertex (v_{i+1}, v_{j+1}) and labels on bottom represent vertex weight.

exists a cycle in $\mathcal{T}'$ and denote this cycle $C' = ((v_k, v_n), \ldots, (v_i, v_j), \ldots, (v_k, v_n))$ such that $(v_i, v_j) \neq r$ and for all $i, j \geq 0$. By the definition of a symmetric tree strategy, there must exist a corresponding cycle $C = ((v_n, v_k), \ldots, (v_j, v_i), \ldots, (v_n, v_k))$. Thus, we have reached a contradiction since T must be acyclic, so T' must be acyclic. Lastly, the definition of symmetric trees preserves connectivity, therefore if T is connected, so is T'.

Lastly, the weight function w' of the symmetric tree T' must be valid. Recall that a weight function is valid if $w(v) \geq 2w(v^+)$, where v^+ is the parent node of v, such that $(v^+, v) \in E(T)$. Let $(u, v), (x, y) \in V$ be arbitrary vertices in T such that (x, y) is the parent node of (u, v), so $((u, v), (x, y) \in E$. Since T is valid, we must have $w(u, v) \geq 2w(x, y)$. Now, consider the corresponding symmetric nodes $(v, u), (y, x) \in V'$ and the edge $((v, u), (y, x)) \in E'$. By the definition of symmetric tree strategies, $w'(v, u) = w(u, v)$ and $w'(y, x) = w(x, y)$. This implies $w'(v, u) \geq 2w'(y, x)$, as desired. Therefore, the validity of the weight function is preserved through symmetry. □

Since all tree strategies produced by $\mathbf{TS}_{G\square G,r}$ have a corresponding valid tree strategy, we may generate a set of $T/2$ strategies and use the symmetric pair of each strategy to produce the desired T strategies while preserving computational cost. To generate these strategies, we define a modified version of $\mathbf{TS}_{G\square G,r}$ to account for the symmetric strategies. Suppose we have a graph $G\square G$ rooted at r and we would like to find T tree strategies, where T is even. For each $i = (v, u) \in V(G\square G)$, define $i' = (u, v) \in V(G\square G)$. The general framework for the Symmetric Tree Strategy Optimization Problem $\mathbf{STS}_{G\square G,r}$ is then defined as follows:

$$\text{Minimize: } \sum_{t=1}^{T/2} \sum_{i \in V(t)} z_i^t + z_{i'}^t \tag{3.13}$$

$$\text{Subject to: } \sum_{j \in V(t):(j,i) \in A} x_{j,i}^t = y_i^t \qquad 1 \leq t \leq T/2 \text{ and } \forall i \in V(G) \tag{3.14}$$

$$\sum_{i \in V(G):(r,i) \in A} y_i^t \geq 1 \qquad 1 \leq t \leq T/2 \tag{3.15}$$

$$\sum_{t=1}^{T/2} z_i^t + z_{i'}^t \geq T \qquad \forall i \in V(G) : i \neq r \tag{3.16}$$

$$z_i^t - 2z_j^t + 2^\ell(1 - x_{i,j}^t) \geq 0 \qquad 1 \leq t \leq T/2 \text{ and } \forall (i, j) \in A(G) : i, j \neq r \tag{3.17}$$

$$z_i^t \leq 2^{\ell-1} \cdot y_i^t \qquad 1 \leq t \leq T/2 \text{ and } \forall i \in V(G) \tag{3.18}$$

$$x_{i,j}^t \in \{0, 1\} \qquad 1 \leq t \leq T/2 \text{ and } \forall (i, j) \in A(G) \tag{3.19}$$

$$y_i^t \in \{0, 1\} \qquad 1 \leq t \leq T/2 \text{ and } \forall i \in V(G) \tag{3.20}$$

$$z_i^t \in \mathbb{R}_{\geq 0} \qquad 1 \leq t \leq T/2 \text{ and } \forall i \in V(G) \tag{3.21}$$

We can then generate a complete set of T tree strategies by taking the corresponding symmetric pair of each strategy produced by $\mathbf{STS}_{G\square G,r}$. Let $\hat{z}$ be the objective value

associated with a feasible set of T strategies. As before, the pebbling bound is given by

$$\pi(G\square G, r) \leq \left\lfloor \frac{\hat{z}}{T} \right\rfloor + 1$$

$\mathbf{STS}_{G\square G,r}$ preserves the structural tree constraints presented in $\mathbf{TS}_{G\square G,r}$ (3.14 and 3.15) as well as the weight function constraints (3.17 and 3.18). Aside from reducing the number of tree strategies by a factor of 1/2, and modifying the objective function to reflect the symmetric properties of the weight functions generated, we modify constraint 3.16. Recall that constraint 3.16 serves to encourage diverse tree strategies by ensuring that each vertex weight is bounded below by the total number of strategies produced. In the case of the symmetric strategies, if we produce $T/2$ symmetric strategies, the total number of strategies used to prove the pebbling bound is T. Further, by symmetry $z_i^t = z_{i'}^t$, with vertex i appearing in $\mathcal{T}$ and i' appearing in its symmetric counterpart $\mathcal{T}'$. Thus, to ensure i and i' each has weight at least $T/2$ in $\mathcal{T}$ and symmetric strategy $\mathcal{T}'$, respectively, we impose constraint 3.16.

We see that this approach drastically reduces the number of constraints in the MILP. Constraints 3.14 and 3.15 are structural constraints that ensure solutions are trees rooted at r, since we are now placing this constraint on $T/2$ trees, we have $T/2$ less constraints than in $\mathbf{TS}_{G\square G,r}$. Constraints 3.17 and 3.18 enforce and bound valid weight functions in a given tree. We have $T/2 \cdot |V|$ and $T \cdot |E|$ instances of constraints 3.17 and 3.18, respectively, so we reduce each of these constraints by a factor of 1/2. Constraints 3.14, 3.15, 3.17 and 3.18 enforce the definition of a tree strategy in 2.1, so they remain unchanged to ensure the tree strategies generated are valid, and their symmetric counterpart is valid in turn. When calculating pebbling numbers through this approach, we're primarily interested in the weight function associated with a structurally valid tree. As such, we modify the objective function and constraint 3.16 to incorporate the symmetric approach.

4 Pebbling Results

The MILP framework above give many encouraging and impressive results using non-basic tree strategies. We begin the section by discussing the computational tools used to implement the mixed-integer linear programs presented above. As previously mentioned, the code used to implement the optimization problems are available as an open source code base[2] for further experimentation. While we present notable results in the Appendix; the `lp` and `log` files, along with certificates and strategy visualizations for all experiments, can also be found in the repository. For the results presented below, we use the popular `Gurobi`[3] mathematical optimization software, yet the code base also implements the `IBM ILOG CPLEX`[4] solver. Neither solver showed significant improvement over the other in experimentation. For each solver, we modify the MILP Focus to find solutions quickly. Furthermore, we use the `NetworkX`[5] Python Library to represent our

[2] `https://github.com/dominicflocco/Graph_Pebbling`

[3] `https://www.gurobi.com`

[4] `https://www.ibm.com/analytics/cplex-optimizer`

[5] `https://networkx.org`

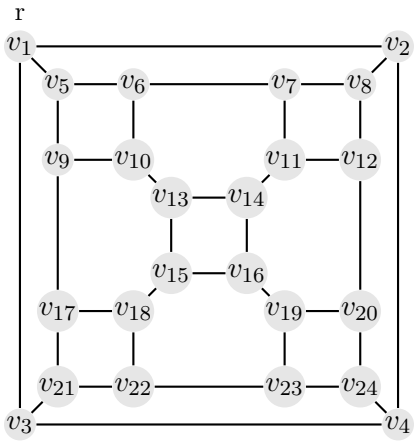


Figure 3: 4^{th} weak Bruhat Graph B_4 on $n = 24$ vertices with $r = v_1$.

pebbling graphs. All experiments were run on a system with an Intel Xeon Processor E5-2620 v4 with 16 cores, each running at 2.1GHz. The system has 128GB of memory and two NUMA nodes. We begin by displaying the usefulness and correctness of the linear programs by confirming the pebbling bounds found in the literature by hand, and then apply the LPs to larger graphs to improve pebbling bounds.

4.1 Bound Improvements

We now apply the Tree Strategy Optimization problem to larger graphs in an effort to improve bounds found using manually generated tree strategies in Hurlbert [11]. Applying linear optimization techniques to automate the generation of tree strategies is particularly useful for large graphs because devising useful strategies for such graphs is often difficult and tedious. Furthermore, the challenge of generating these strategies manually only increases as the number of strategies generated grows. There may be more useful tree strategies that improve pebbling bounds with a large set of strategies infeasible to devise by hand. The two graphs of interest for this section are the 4^{th} weak Bruhat B_4 (Figure 3) and the Lemke square graph $L \square L$. We show that improved pebbling bounds are possible by utilizing larger parameters $|\mathcal{T}_r|$ and ℓ, which would be unreasonable to generate by hand.

The experiment on the 4^{th} weak Bruhat graph B_4 shows that improved pebbling bounds are possible as we increase T. In [11], Hurlbert presents a certificate and visualization that shows $\pi(B_4, r) \leq 72$ using 3 strategies. However, using $\mathbf{TS}_{B_4,r}$ with larger T and ℓ, we obtain the following result.

Theorem 4.1. *Let B_4 be the Bruhat graph of order 4. Then $\pi(B_4) \leq 66$.*

Proof. The $T = 6$ strategies with $r = v_0$ in Figure 4 certify the result. Since the graph is vertex transitive, only one root must be checked. □

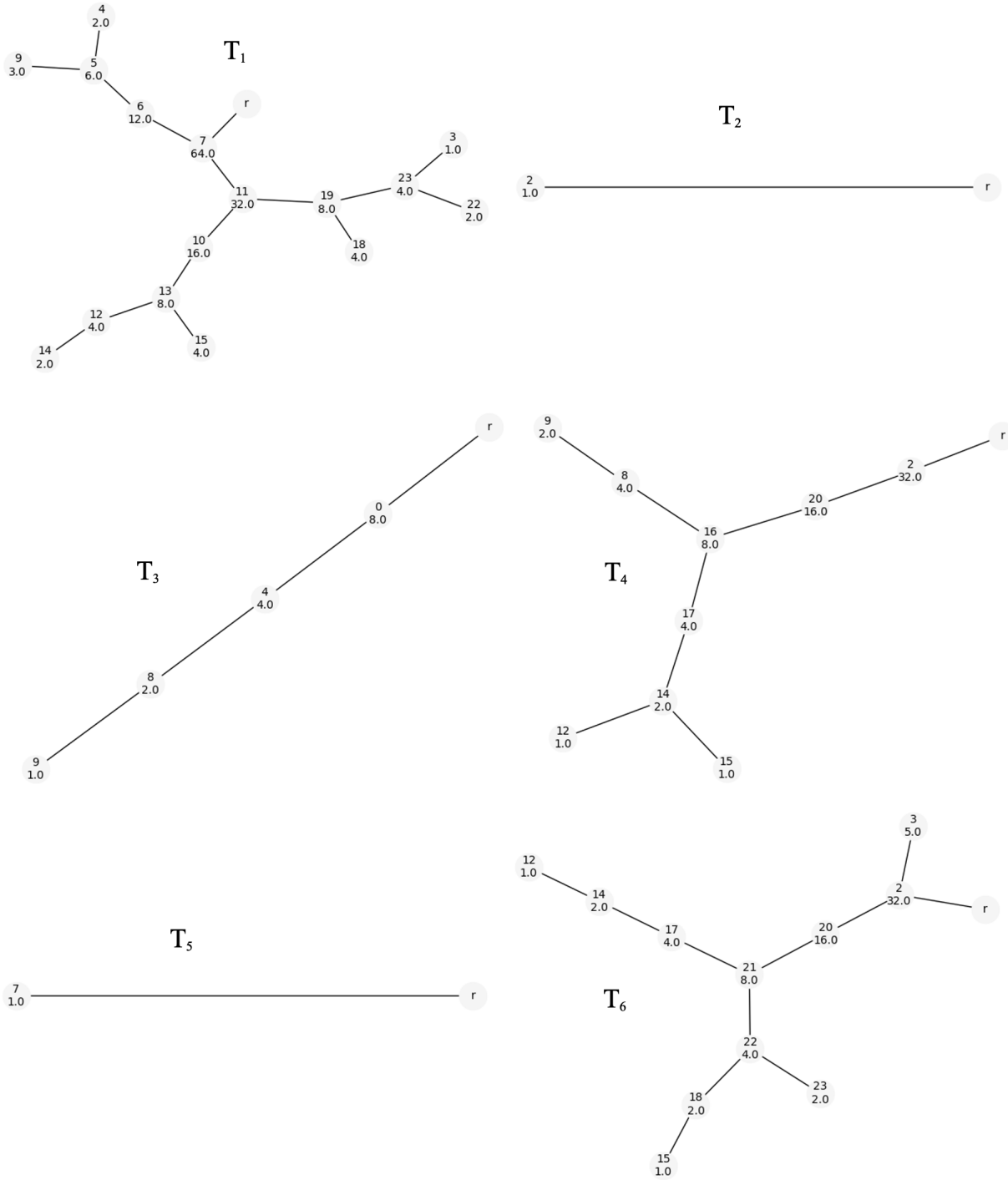


Figure 4: Set of tree strategies $\mathcal{T}_r = \{T_1, T_2, T_3, T_4, T_5, T_6\}$ that prove $\pi(B_4) \leq 66$.

T	ℓ	Runtime (sec)	Solver	Threads
6	16	3656.2365	`Gurobi`	16

Table 1: Experimental parameters for $\mathbf{TS}_{B_4,v_1}$ with root $r = v_1$ that produced bound $\pi(B_4) \leq 66$.

T	ℓ	Runtime (sec)	Solver	Threads
10	16	43438.1156	`Gurobi`	16

Table 2: Experimental parameters for $\mathbf{STS}_{L\square L,r}$ with root $r = (v_1, v_1)$ that produced bound $\pi(L\square L, (v_1, v_1)) \leq 96$.

To obtain this bound, we use the parameters listed in Table 1. This is a significant improvement on the results of Hurlbert and exhibits the usefulness and limitations of using linear optimization to generate tree strategies. Further, it raises the following conjecture.

Conjecture 4.1. *If B_4 is the 4^{th} weak Bruhat graph, then the bound $\pi(B_4) \leq 66$ is best possible using tree strategy weight functions.*

4.2 Improving Results with Symmetry

The Lemke square is of particular interest to pebbling researchers since it is a candidate for a counterexample to Graham's Conjecture 2.1. Due to the size and complexity of the Lemke square, generating tree strategies by hand is difficult has presents many limitations. Hurlbert used manually generated certificates to show that

$$\pi(L\square L, (r, r)) \leq \begin{cases} 108 & \text{if } r = v_1, \\ 96 & \text{if } r = v_8, \\ 68 & \text{if } r = v_4. \end{cases},$$

using 4 strategies for $r = v_4$, 4 strategies for $r = v_8$ and 5 strategies for $r = v_1$. Further Kenter [13] uses an integer program to provide strong evidence that $\pi(L\square L) \leq 85$. Our goal is to use linear optimization to generate larger sets of strategies in hopes of improving these bounds and to provide certificates that serve to validate the proof. We apply both $\mathbf{TS}_{G,r}$ and $\mathbf{STS}_{G,r}$ to the Lemke square graph in this pursuit. In addition to successfully improving the bounds of Hurlbert [11], the experimentation also provides useful insights into the computational properties of the linear programs.

Due to the size and complexity of the Lemke square, the solving of the MILPs above, even with state-of-the-art optimization solvers, is computationally expensive. Given a graph with n vertices and m edges, generating a set of T strategies involves $T \cdot (2n + m)$ total decision variables, $T \cdot n$ of which are continuous and $n + m$ of which are binary, and $T(2m + 2n + 1) + n$ constraints. As such, the magnitude and complexity of the MILP proved to be a difficult task for the solvers, and it should be noted that none of

T	ℓ	Runtime (sec)	Solver	Threads
8	16	10800	`Gurobi`	16

Table 3: Experimental parameters for $\mathbf{STS}_{L\square L,r}$ with root $r \neq (v_1, v_1)$ that produced bound $\pi(L\square L, r) \leq 96$ for all $r \in V(L\square L)$.

the MILPs ran to optimality on the Lemke square. Thus, we focus our attention on the computational performance of the MILP on the `Gurobi` software. As a result of the magnitude of the Lemke square, the best bound on the Lemke square was found using $\mathbf{STS}_{L\square L,r}$. As noted by Hurlbert [11], the root $r = (v_1, v_1)$ proves to be the most difficult root for which to obtain low upper bound due to its low degree and location in the graph. Through experimentation, it is apparent that (v_1, v_1) restricts the upper bound most tightly. In fact, most other roots produce an upper bound close to the desired bound of 64. Using $T = 10$ symmetric tree strategies we prove that $\pi(L\square L, (v_1, v_1)) \leq 96$, and prove that this bound holds for all roots $r \neq (v_1, v_1)$ with $T = 6$ symmetric strategies.

Theorem 4.2. *Let L be the Lemke graph. Then $\pi(L\square L, (r_1, r_2)) \leq 96$, for all $(r_1, r_2) \in L\square L$.*

Proof. See Appendix A.2 for certificates and Appendix A.3 for the visualizations of each tree strategy with accompanying weight function for $r = (v_1, v_1)$. Certificates and visualizations for all $r \in V(L\square L)\backslash\{(v_1, v_1)\}$ are presented in the accompaning repository [6], and the bounds obtained for these vertices are presented in Table 4 The results were obtained using the parameters summarized in Table 2 for $r = (v_1, v_1)$ and in Table 3 for $r \neq (v_1, v_1)$. Without loss of generality, we compute bounds for the set of 32 roots presented in Table 4. By the symmetry of $L\square L$, $\pi(L\square L, (r_1, r_2)) = \pi(L\square L, (r_2, r_1))$, which provides bounds for the remaining 32 vertices. □

We again see the usefulness of the MILP framework in automating the generation of weight functions to prove pebbling bounds on large graphs. Using linear optimization to produce large sets of tree strategies proved useful in improving the pebbling bounds on the Lemke square found [11]. The method has the benefit of providing verifiable proofs, producing a large set of tree strategies and the adaptability to be applied more general classes of graphs. Through automating the generation of tree strategies, we are able to consider large sets of these strategies, which improve bounds. The bound obtained is a string upper bound for all possible roots $r \in V(L\square L)$, as shown in Figure 4. Note that the pebbling bound dropped below 96 for all roots $r \neq (v_1, v_1)$, and often very quickly (typically in less than 5 minutes using 32 threads). Further, these bounds were obtained using only 6 symmetric tree strategies. While the $\mathbf{TS}_{L\square L,r}$ performed impressively on smaller graphs that were not products, it was unable to produce a bound better than $\pi(L\square L, (v_1, v_1)) \leq 98$ with 10 symmetric strategies. It should be noted that $\mathbf{TS}_{L\square L,r}$ achieved the bound of 96 with 30 non-symmetric strategies. Nonetheless, the $\mathbf{TS}_{L\square L,r}$ produced similar results to $\mathbf{STS}_{L\square L,r}$ for roots $r \neq (v_1, v_1)$, displaying the potential to

[6] `https://github.com/dominicflocco/Graph_Pebbling`

r	$\pi(L\square L, r) \leq$	r	$\pi(L\square L, r) \leq$	r	$\pi(L\square L, r) \leq$
(v_8, v_5)	72	(v_2, v_7)	65	(v_3, v_7)	65
(v_5, v_2)	65	(v_1, v_8)	64	(v_7, v_5)	67
(v_3, v_1)	71	(v_3, v_6)	65	(v_1, v_7)	65
(v_6, v_2)	65	(v_6, v_4)	66	(v_1, v_6)	65
(v_5, v_4)	66	(v_4, v_8)	71	(v_1, v_4)	65
(v_1, v_2)	75	(v_6, v_8)	71	(v_2, v_4)	64
(v_2, v_8)	64	(v_7, v_8)	71	(v_3, v_2)	65
(v_6, v_5)	67	(v_6, v_7)	66	(v_7, v_7)	72
(v_5, v_3)	65	(v_1, v_5)	65	(v_8, v_8)	88
(v_5, v_5)	75	(v_3, v_4)	65	(v_4, v_4)	66
(v_3, v_8)	67	(v_3, v_3)	68	(v_2, v_2)	75
(v_4, v_7)	66	(v_6, v_6)	70	(v_1, v_1)[a]	96

[a] Using parameters outlined in Table 2.

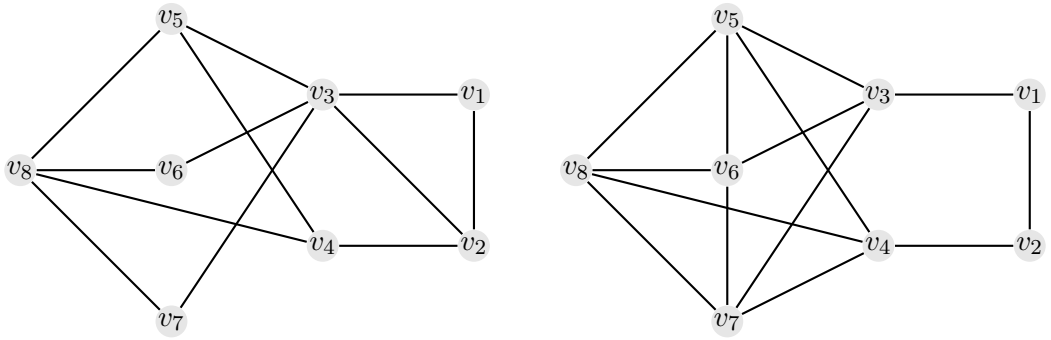
Table 4: Upper bounds obtained with $\mathbf{STS}_{L\square L,r}$ for $\pi(L\square L)$ for all roots $r \in V(L\square L)\backslash\{(v_1, v_1)\}$ using parameters outlined in Table 3 unless noted otherwise.

produce useful upper bounds. However, it is evident that leveraging the symmetry of graph products significantly improved performance on the Lemke square.

Upon closer examination of the tree strategies produced by $\mathbf{STS}_{L\square L,r}$ for $r = (v_1, v_1)$, we notice that some generated strategies are larger and more complex than others. For example, strategies T_4 (Figure 9) and T_{10} (Figure 15) are comprised of only one other vertex and the root. This suggests that the bound of 96 can be found with fewer strategies, however the optimization solver could not find a feasible solution to the MILP that produced this bound due to the problems computational complexity. While the objective value plateaued at 95.1999 after running on 16 threads for 18 hours, the MILP appears to have wasted two strategies in its computation. After much experimentation on $L\square L$ using different softwares, problem parameters and computational power, we are convinced that the best attainable bound using tree strategies on the Lemke square is in fact 96. However, the strategies produces with $T = 20$ indicate that this bound can be obtained with fewer strategies and is a potential avenue for future research.

To further exhibit the versatility of our MILP approach, we apply $\mathbf{STS}_{G,r}$ to all non-isomorphic Lemke square graphs on 8 vertices. There are only two non-isomorphic Lemke graphs in addition to the original Lemke graph L, which we denote L_2 and L_3, shown in Figures 5a and 5b, respectively. Using the parameters outlined in Table 3, we are able to compute an upper bound for the rooted pebbling number on both $L_2\square L_2$ and $L_3\square L_3$ for all roots, and obtain the pebbling bounds $\pi(L_2\square L_2) \leq 97$ and $\pi(L_3\square L_3) \leq 97$. The results for each root are presented in Tables 5 and 6 in Appendix A.1, and the certificates are offered in the accompanying repository.[7] Not only do these results offer further insight into properties of Lemke square graphs, but they display the utility of

[7] https://github.com/dominicflocco/Graph_Pebbling



(a) Second non-isomorphic Lemke graph L_2. (b) Third non-isomorphic Lemke graph L_3.

Figure 5: Non-isomorphic Lemke graphs L_2 and L_3 on $n = 8$ vertices.

our MILP approach.

5 Conclusion

Through our research, we present a number of automation approaches to weight function generation in graph pebbling which yeild improved pebbling bounds. Namely, applying mixed-integer linear programming to generate an ideal set of tree strategies that attempt to minimize the pebbling bound of a graph is capable of producing weight function certificates that offer lower pebbling bounds than feasible by hand. It is evident that lower pebbling bounds can be found by generating more tree strategies, lending itself to computational approaches, which are capable of producing a large number of weight functions with continuous weights. Our MILP approach to tree strategy optimization is successful in improving the bound on the Bruhat graph using only tree strategies to 66, confirming the results of [6] and improving results of [11]. This technique is also capable of producing tree strategies on smaller graphs in a short amount of time, indicating that the MILP will be successful when applied to more general graphs, possibly arising from applications.

The research was initially motivated by Graham's Conjecture 2.1, leading to the investigation of graph products and the Lemke square as a potential counterexample. Through leveraging the symmetry of such graphs, we were able to produce a set of tree strategy certificates that prove $\pi(L\square L, r) \leq 96$ for all $r \in V(L\square L)$. This improves the results of [11]. The benefit of our approach is the generation of certificates that can be verified by hand and that the computation bypasses the potential of precision error in arithmetic. We also show that the upper bound of $\pi(L\square L, r)$ drops close to 64, and even to 64, for some roots. These results exhibit the interesting properties of the Lemke square and its potential to be a counterexample for Graham's Conjecture.

References

[1] A. Bekmetjev and C. Cusack. Pebbling algorithms in diameter two graphs. *SIAM Journal on Discrete Mathematics*, 23:634–646, 2009.

[2] A. Blasiak and J. Schmitt. Degree sum conditions in graph pebbling. *Australas. J. Combin.*, 42:83–90, 2008.

[3] B. Bukh. Maximum pebbling number of graphs of diameter three. *Journal of Graph Theory*, 52(4):353–357, 2006.

[4] F. R. K. Chung. Pebbling in hypercubes. *SIAM Journal on Discrete Mathematics*, 2:467–472, 1989.

[5] W. J. Cook, T. Koch, D. E. Steffy, and K. Wolter. A hybrid branch-and-bound approach for exact rational mixed-integer programming. *Mathematical Programming Computation*, 5:305–344, 2013.

[6] D. W. Cranston, L. Postle, C. Xue, and C. Yerger. Modified linear programming and class 0 bounds for graph pebbling. *Journal of Combinatorial Optimization*, 43:114–132, 2017.

[7] A. Czygrinow, G. Hurlbert, H. Kierstead, and W. Trotter. A note on graph pebbling. *Graphs and Combinatorics*, 18:219–225, 2002.

[8] R. Feng and J. Y. Kim. Graham's pebbling conjecture on product of complete bipartite graphs. *Science in China Series A: Mathematics*, 44(817):470–478, 2001.

[9] D. S. Herscovici. Graham's pebbling conjecture on products of many cycles. *Discrete Mathematics*, 308(24):6501–6512, 2008.

[10] G. Hurlbert. Two pebbling theorems. *Congress. Numer*, 135:6–3, 2000.

[11] G. Hurlbert. A linear optimization technique for graph pebbling. *arXiv: Combinatorics*, 2010.

[12] F. Kenter and D. Skipper. Integer-programming bounds on pebbling numbers of cartesian-product graphs. In *Proceedings of the 12th International Conference on Combinatorial Optimization and Applications*, pages 681–695, 2018.

[13] F. Kenter, D. Skipper, and D. Wilson. Computing bounds on product-graph pebbling numbers, 2019.

[14] K. Milans and B. Clark. The complexity of graph pebbling. *SIAM Journal on Discrete Mathematics*, 20:769–798, 2006.

[15] D. Moews. Pebbling graphs. *Journal of Combinatorial Theory, Series B*, 55(2):244–252, 1992.

[16] L. Pachter, H. Snevily, and B. Voxman. On pebbling graphs. *Congressus Numerantium*, 107:65–80, 1995.

[17] L. Postle, N. Streib, and C. R. Yerger. Pebbling graphs of diameter three and four. *J. Graph Theory*, 72:398–417, 2013.

[18] N. Sieben. A graph pebbling algorithm on weighted graphs. *J. Graph Algorithms Appl.*, 14:221–244, 2010.

[19] S. S. Wang. Pebbling graham's conjecture. *Discrete Mathematics*, 226:431–438, 2000.

[20] N. Watson. The complexity of pebbling and cover pebbling. *arXiv: Combinatorics*, 2005.

A Appendix

A.1 Rooted pebbling number bounds on $L_2\square L_2$ and $L_3\square L_3$

r	$\pi(L\square L, r) \leq$	r	$\pi(L\square L, r) \leq$	r	$\pi(L\square L, r) \leq$
(v_8, v_5)	72	(v_2, v_7)	65	(v_3, v_7)	65
(v_5, v_2)	65	(v_1, v_8)	64	(v_7, v_5)	67
(v_3, v_1)	71	(v_3, v_6)	65	(v_1, v_7)	67
(v_6, v_2)	65	(v_6, v_4)	67	(v_1, v_6)	68
(v_5, v_4)	68	(v_4, v_8)	71	(v_1, v_4)	66
(v_1, v_2)	75	(v_6, v_8)	71	(v_2, v_4)	67
(v_2, v_8)	64	(v_7, v_8)	71	(v_3, v_2)	65
(v_6, v_5)	68	(v_6, v_7)	70	(v_7, v_7)	73
(v_5, v_3)	64	(v_1, v_5)	65	$(v_8, v_8)^{\text{a}}$	87
$(v_5, v_5)^{\text{a}}$	69	(v_3, v_4)	64	(v_4, v_4)	74
(v_3, v_8)	65	(v_3, v_3)	65	(v_2, v_2)	71
(v_4, v_7)	67	(v_6, v_6)	78	$(v_1, v_1)^{\text{a}}$	97

[a] Runtime 21,600 seconds.

Table 5: Upper bounds obtained with $\mathbf{STS}_{L_2\square L_2, r}$ for $\pi(L_2\square L_2)$ for all roots $r \in V(L_2\square L_2)$ using parameters outlined in Table 3 unless noted otherwise.

r	$\pi(L\square L, r) \leq$	r	$\pi(L\square L, r) \leq$	r	$\pi(L\square L, r) \leq$
(v_8, v_5)	71	(v_2, v_7)	65	(v_3, v_7)	66
(v_5, v_2)	64	(v_1, v_8)	65	(v_7, v_5)	67
(v_3, v_1)	71	(v_3, v_6)	71	(v_1, v_7)	65
(v_6, v_2)	65	(v_6, v_4)	66	(v_1, v_6)	65
(v_5, v_4)	66	(v_4, v_8)	71	(v_1, v_4)	64
(v_1, v_2)	78	(v_6, v_8)	72	(v_2, v_4)	71
(v_2, v_8)	65	(v_7, v_8)	72	(v_3, v_2)	65
(v_6, v_5)	72	(v_6, v_7)	72	(v_7, v_7)	70
(v_5, v_3)	66	(v_1, v_5)	65	$(v_8, v_8)^{\text{a}}$	90
(v_5, v_5)	69	(v_3, v_4)	65	$(v_4, v_4)^{\text{a}}$	69
(v_3, v_8)	65	(v_3, v_3)	69	(v_2, v_2)	97
(v_4, v_7)	66	(v_6, v_6)	91	$(v_1, v_1)^{\text{a}}$	97

[a] Runtime 21,600 seconds.

Table 6: Upper bounds obtained with $\mathbf{STS}_{L_3\square L_3, r}$ for $\pi(L_3\square L_3)$ for all roots $r \in V(L_3\square L_3)$ using parameters outlined in Table 3 unless noted otherwise.

A.2 Lemke Square $L \square L$ Certificates

T_1	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1			40.00					
v_2			20.00	1.88	10.00	10.00	10.00	1.50
v_3								
v_4			9.00	1.50	5.00	5.00	5.00	2.50
v_5					2.50	2.25		1.25
v_6			3.75		2.50		2.50	1.25
v_7				1.25	2.50	1.37	2.50	1.25
v_8				1.00	1.25	2.50	2.50	1.25

T_2	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1		32.00						
v_2		16.00						
v_3								
v_4		8.00		4.00	1.50			
v_5		1.00						
v_6				2.00		1.00	1.00	1.00
v_7								
v_8				2.00			1.00	1.00

Table 7: Certificates T_1 and T_2 for $\pi(L \square L) \leq 96$ proof.

T_3	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1		1.00						
v_2								
v_3								
v_4								
v_5								
v_6								
v_7								
v_8								

T_4	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1								
v_2								
v_3	32.00							
v_4								
v_5	16.00	8.00	3.00	4.00	2.00			
v_6				1.00				
v_7								
v_8	2.00		1.00	2.00	1.00			1.00

Table 8: Certificates T_3 and T_4 for $\pi(L \square L) \leq 96$ proof.

T_5	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1			36.00					
v_2								
v_3			18.00		9.00		9.00	1.00
v_4					1.00		2.25	
v_5							4.50	
v_6			2.25	2.00	4.50	1.00	4.50	2.25
v_7			7.50		4.50	1.00	3.00	2.25
v_8				1.00	2.25			1.13

T_6	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1								
v_2								
v_3	56.00						3.50	
v_4								1.75
v_5		1.00						
v_6	28.00	10.00	14.00	1.25	7.00	7.00	7.00	3.50
v_7							1.00	
v_8		5.00	5.00		3.13	3.50	3.50	1.75

Table 9: Certificates T_5 and T_6 for $\pi(L \square L) \leq 96$ proof.

T_7	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1			48.00				24.00	12.00
v_2							10.00	6.00
v_3								6.00
v_4							5.00	3.00
v_5								3.00
v_6								3.00
v_7				1.00			2.50	3.00
v_8			1.00				2.25	1.50

T_8	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1			32.00		16.00			6.00
v_2								3.00
v_3					8.00			4.00
v_4					1.00			1.00
v_5				2.00	4.00			2.00
v_6				1.00		1.00	1.00	2.00
v_7					4.00		1.00	2.00
v_8				1.00	2.00			1.00

Table 10: Certificates T_7 and T_8 for $\pi(L \square L) \leq 96$ proof.

T_9	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1		44.00		22.00				
v_2				10.12				4.50
v_3				11.00				2.00
v_4				4.50		2.25		2.00
v_5				5.50	1.50	1.00	1.75	2.75
v_6				5.50	2.75			1.00
v_7				5.50	2.75	1.63		2.25
v_8				2.75	1.37			1.37

T_{10}	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1								
v_2	1.00							
v_3								
v_4								
v_5								
v_6								
v_7								
v_8								

Table 11: Certificates T_9 and T_{10} for $\pi(L \square L) \leq 96$ proof.

A.3 Lemke Square $L \square L$ Strategy Visualizations

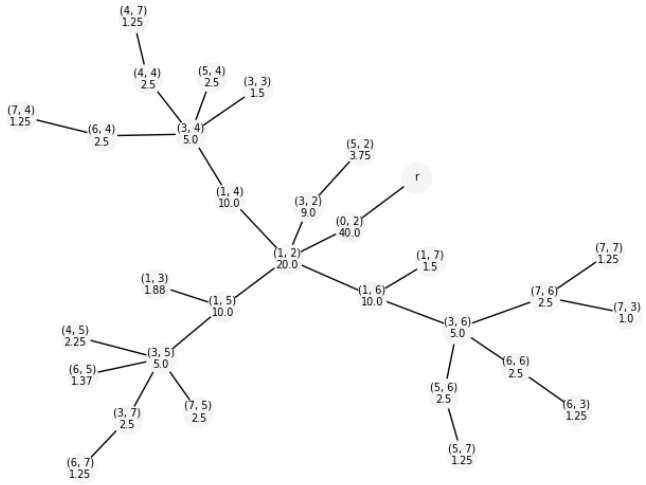


Figure 6: Visualization of T_1 for $\pi(L \square L) \leq 96$ proof.

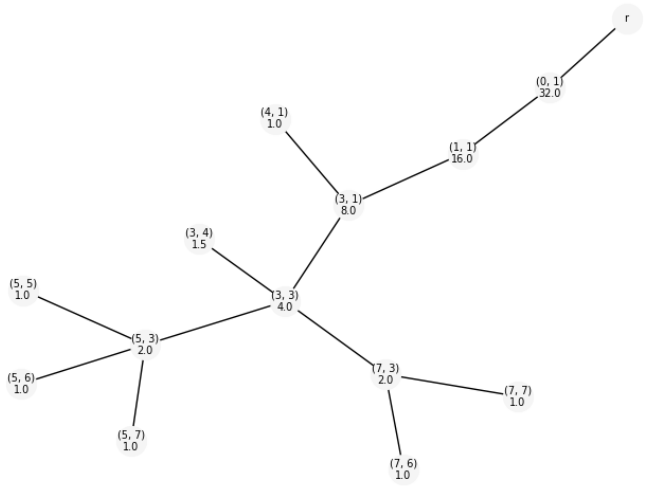


Figure 7: Visualization of T_2 for $\pi(L \square L) \leq 96$ proof.

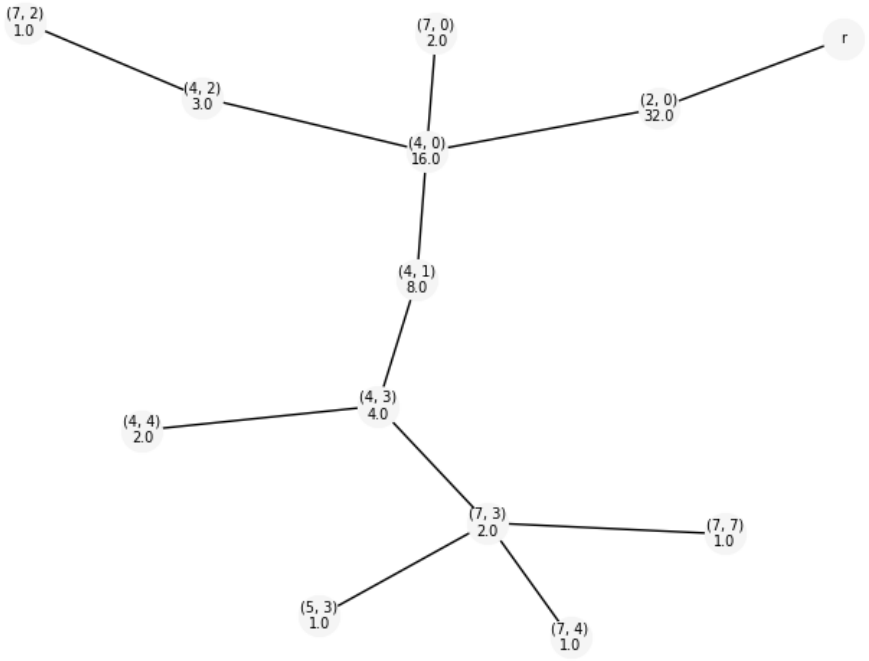


Figure 8: Visualization of T_3 for $\pi(L \square L) \leq 96$ proof.

Figure 9: Visualization of T_4 for $\pi(L \square L) \leq 96$ proof.

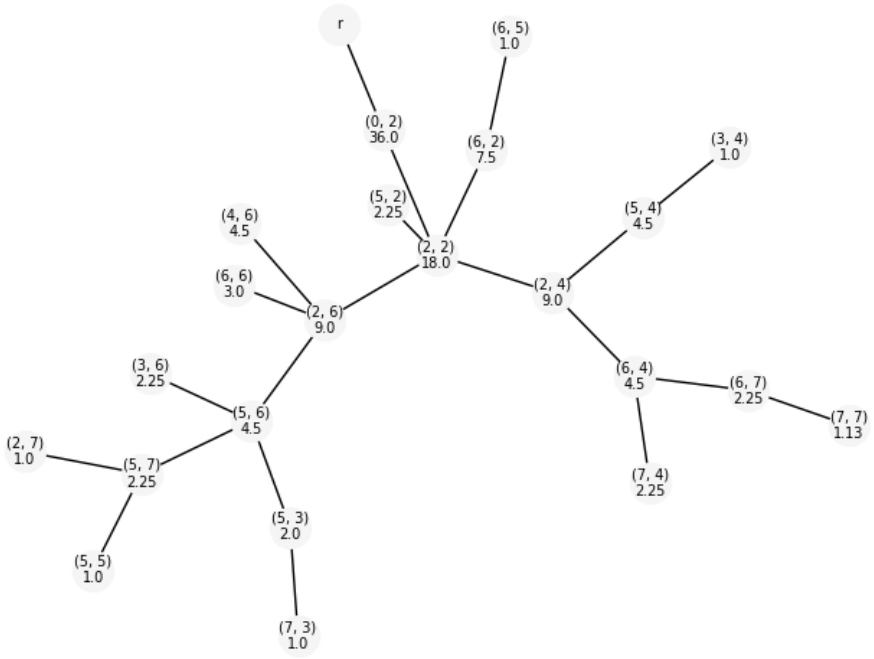


Figure 10: Visualization of T_5 for $\pi(L\square L) \leq 96$ proof.

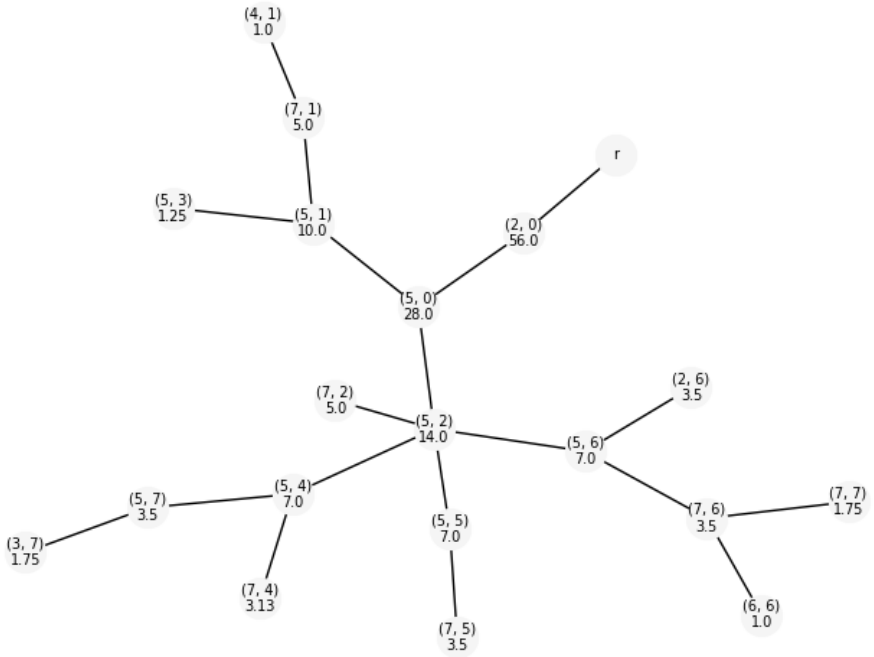


Figure 11: Visualization of T_6 for $\pi(L\square L) \leq 96$ proof.

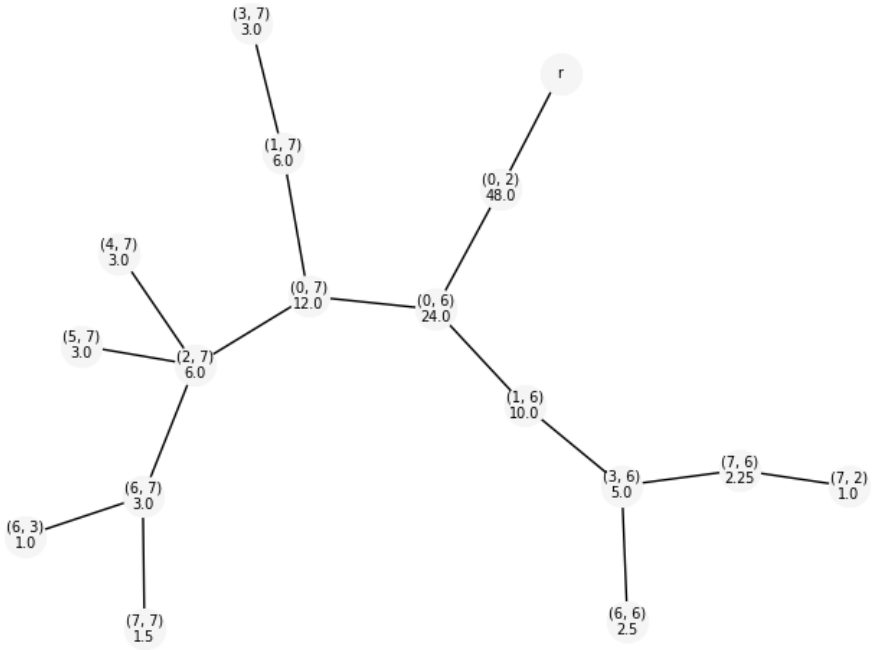


Figure 12: Visualization of T_7 for $\pi(L\square L) \leq 96$ proof.

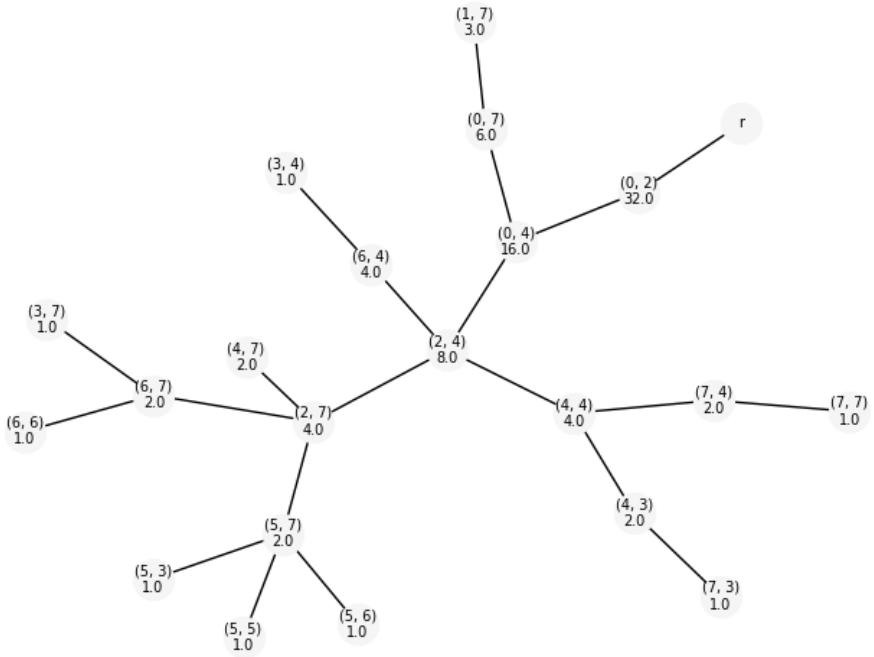


Figure 13: Visualization of T_8 for $\pi(L\square L) \leq 96$ proof.

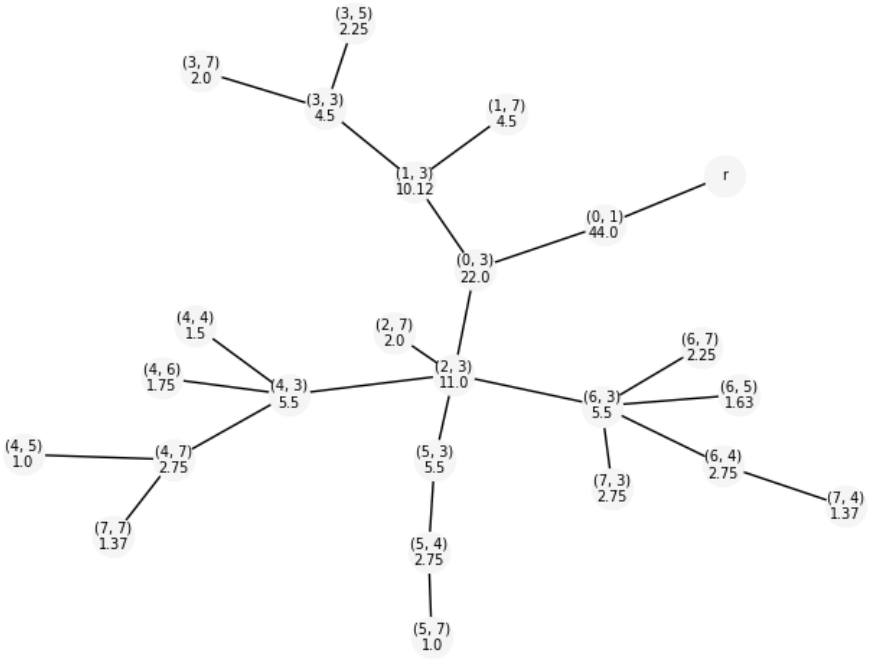


Figure 14: Visualization of T_9 for $\pi(L \square L) \leq 96$ proof.

Figure 15: Visualization of T_{10} for $\pi(L \square L) \leq 96$ proof.