Spring 2019

Successful Pressing Sequences in Simple Pseudo-Graphs

Hays Wimsatt Whitlatch

Successful Pressing Sequences in Simple Pseudo-Graphs

by

Hays Wimsatt Whitlatch

Bachelor of Science
University of Iowa 2008

Master of Science
Middle Tennessee State University 2014

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Mathematics

College of Arts and Sciences

University of South Carolina

2019

Accepted by:

Joshua Cooper, Major Professor

Éva Czabarka, Committee Member

Stephen Dilworth, Committee Member

László Székely, Committee Member

Csilla Farkas, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

Dedication

To my parents who tricked me into enrolling in college. Thank you for all of your love and support - I wouldn't have made it without you.

Acknowledgments

First and foremost, Joshua Cooper, thank you for being my advisor, mentor, instructor, ally, and collaborator. Above all, you have been a friend that always believed in me. You never gave up on your dream of transforming us into real mathematicians (sorry, maybe Greg will work out...). Thank you for never giving up on me - none of this would have been possible without you. I look forward to expanding our friendship and collaborations.

Thank you, Erin Hanna and Peter Gartland. Lil' sis - you rock! I hope you keep your passion for teaching and learning, South Carolina will be a better state because of you. (I'm 100% quasitive about that!). Peter, we did some awesome math together (lots of it appears here)! I hope we get to continue collaborating - best of luck in your Computer Science PhD.

To Éva Czabarka and László Székely, thank you. Some of the best things come in pairs (not just smelly socks). You have welcomed me into your home and made U of SC a wonderful place. I am particularly grateful to Éva Czabarka. I still think I deserved an A in graph theory, but I won't argue it anymore if you just sign off on my thesis :). You have a unique way of knowing how to treat each person individually so that they are supported and challenged. Thank you for always providing a challenge and supporting me in a solution. Your mentorship has been invaluable. Thank you as well for sharing your biological knowledge.

I am grateful to Stephen Dilworth and Csilla Farkas who have courageously agreed to be on my committee. I am also grateful to all the wonderful professors from whom I have learned while at U of SC: Joshua Cooper, Éva Czabarka, Stephen Dilworth,

Lincoln Lu, George McNulty, and László Székely.

A very special thank you to my "guide on the side" and teaching mentor Sean Yee. Your guidance has helped me mature as a teacher and mentor - LeConte is lucky to have you!

Finally, thank you to all of my family and friends who supported me in this endeavor. Thank you to Alberto & Betsy, Meggy & Toni, and Mom & Dad for supporting me from far away. A special thank you to Gregory Clark, Ralph Howard, Chris Edgar, and Duncan Wright, who helped me prepare for my qualifying and comprehensive exams. Last, but certainly not least, a loving thank you to Hannah Kimbrell and Athena the toe-licking goddess - I am so lucky that you are in my life!

This work was supported in part by a SPARC Graduate Fellowship from the Office of the Vice President for Research at the University of South Carolina.

Abstract

Motivated by the study of genomes evolving by reversals, the primary topic of this thesis is "successful pressing sequences" in simple pseudo-graphs. Pressing sequences where first introduced by Hannenhali and Pevzner in 1999 where they showed that sorting signed permutation problem can be solved in polynomial time, therefore demonstrating that the length of a most parsimonious solution to the genome inversion only rearrangement problem can be determined efficiently.

A signed permutation is an integer permutation where each entry is given a sign: plus or minus. A reversal in a signed permutation is the operation of reversing a subword and flipping the signs of the subword's entries. The primary computational problem of sorting signed permutations by reversals is to find the minimum number of reversals needed to transform a signed permutation into the positive identity permutation. Hannenhalli and Pevzner showed that the signed sorting problem can be solved in polynomial-time in contrast to the problem of sorting unsigned permutations, which is known to be NP-hard in general. At the core of the argument given by Hannenhali and Pevzner is the study of successful pressing sequences on vertex 2-colored graphs.

The connection between permutation sorting and phylogenetics dates back to at least the 1930's, when two biologists, Dobzhansky and Sturtevant, wrote a series of papers in which they argued that the relationships between possible gene arrangements within a given chromosome encode critical information about the evolutionary history of species containing those genomes. In particular, they introduced the idea that the degree of disorder between the genes in two genomes is an indicator of the

evolutionary distance between two organisms. This has inspired extensive work in the fields of computational biology, bio-informatics and phylogenetics. In particular, researchers have pursued the question of how a common ancestral genome may have been transformed by evolutionary events into distinct, yet homologous, genomes. In mathematics and computer science, we often represent genomes as signed permutations (signed since DNA is oriented between two strands) and evolutionary events are encoded as operations on signed permutations. Among the most studied operations are block transpositions, prefix-reversals, and reversals, all of which correspond to common evolutionary mechanisms.

In addition to the study of pressing sequences in simple pseudo-graphs, in this thesis we discuss related topics such as Cholesky factorizations of matrices over finite-fields, a sampling algorithm to generate simple pseudo-graphs uniformly at random, and the complexity of the "pressing space" of a simple pseudo-graph (the space of all successful pressing sequences of a simple pseudo-graph). This work includes collaborative work with Dr. Joshua Cooper (Mathematics, University of South Carolina), M.S. graduate Erin Hanna (Mathematics, University of South Carolina), and M.S. candidate Peter Gartland (Mathematics, University of South Carolina).

Table of Contents

List of Figures

Chapter 1

Origin of Pressing Sequences in Graphs

1.1 Introduction

In the late 1930's, two biologists, Dobzhansky and Sturtevant, wrote a series of papers in which they argued that the relationships between possible gene arrangements within a given chromosome encode critical information about the evolutionary history of species containing those genomes (see, e.g., [12, 27]). In particular, they introduced the idea that the degree of disorder between the genes in two genomes is an indicator of the evolutionary distance between two organisms. This has inspired extensive work in the fields of computational biology, bio-informatics and phylogenetics (see, e.g., [14]). In particular, researchers have pursued the question of how a common ancestral genome may have been transformed by evolutionary events into distinct, yet homologous, genomes. Genomes are sets of chromosomes and chromosomes are built from deoxyribonucleic acid (DNA). DNA is a double-stranded molecule where each strand is oriented (corresponding to the direction that the ribosome reads the DNA) in opposite order. This means that in any of the DNA strand you have a footprint of all genes on the chromosomes: a gene is either in the order of the DNA strand (meaning the ribosome reads the gene from this strand) or the ribosome reads it from the other strand and what you see on the chromosome is the reverse-complement of the gene sequence. For this reason, in mathematics and computer science, we often represent genomes as signed permutations (the signs corresponding to the orientations of the strands) and evolutionary events are encoded as operations on signed

permutations. Among the most studied operations are block transpositions (e.g., [2, 17, 30]), prefix-reversals (e.g., [7, 8, 15, 18]), and reversals (e.g., [1, 23, 16, 17, 20, 26, 28]), all of which correspond to common evolutionary mechanisms.

1.2 Genome Mutations

The genome of a species can be thought of as a set of ordered sequences of genes. Each molecule is called a chromosome, and the set of all chromosomes is what we will call the genome. Chromosomes are made of deoxyribonucleic acid (DNA), a double-stranded molecule in which each strand is a long succession of nucleotides. DNA is made up of two complementary strands. During replication, these strands are separated and two copies of DNA are produced. Cellular proofreading and error-checking mechanisms ensure that the results are near-identical, but allow for some minor *point mutations*. There are three different kinds of point mutations: insertions, deletions, and substitutions. An insertion occurs when a nucleotide is inserted (added) to the sequence; a deletion occurs when a nucleotide is removed (subtracted) from the sequence; a substitution occurs when a nucleotide is removed from the sequence and replaced with another.

A large-scale genetic mutation is referred to as a *genetic rearrangement*. While point mutations happen with some frequency during replication and typically have a minor effect on the outcome of the organism, genetic rearrangements happen infrequently, have a large effect on the outcome of the reproduction, and therefore are a common metric used to estimate the evolutionary distance between organisms ([12, 27]). The most commonly studied rearrangements include deletions, transpositions, inversions, duplications, reciprocal translocations, fusion, fission, and horizontal transfer. ([14]).

- Deletions: A segment of the genome is lost.

- Translocation: A segment of the genome moves to another location.
- Inversion: A segment of the genome is reversed and the strands are exchanged.
- Duplication: A segment of DNA is copied and inserted in the genome.
- Reciprocal translocation: two broken off chromosomal pieces (each containing a telomere) are exchanged.
- Fusion: Two chromosomes are joined into one.
- Fission: One chromosome splits into two.
- Horizontal transfer: A segment of the genome is copied from one genome to another.

Originating from Greek words meaning same (homo) and proportion (logos), the term *homology* is use to describe the relationship between two objects that are similar in position or structure, but not necessarily in function. In mathematics, homology is a general way of associating a sequence of algebraic objects to other mathematical objects such as topological spaces. In genetics, two objects are said to be homologous when their genetic sequences derive from a common origin. Observe that in the inversion model no genetic information is deleted or introduced, it is simply rearranged. Thus, in this model, we consider two genomes to be homologous when they who present the same genetic information but in different orders. We attempt to understand how the two organisms may have evolved from a common ancestor by inversion rearrangements.

In its simplest form, the genome rearrangement problem is formulated as follows: given a pair of genomes and a set of possible evolutionary events, find a most likely set of events that transforms one genome into the other. This thesis is motivated by studying the genome rearrangement problem restricted to inversions. Since rearrangements are relatively rare events, scenarios minimizing their number are more

likely to be close to reality ([14]). Thus, we may rephrase the genome rearrangement problem as such: given a pair of genomes and a set of possible evolutionary events, find a shortest set of events transforming one genome into the other; in particular, in the inversion model we are tasked with finding a shortest sequence of inversions that transform one genome into another.

In the following sections we describe breakpoint graphs, and their corresponding overlap graphs. We will introduce successful pressing sequences on the bicolored graphs. In [20], the authors use *successful pressing sequences* on overlap graphs to demonstrate that the genome rearrangement problem restricted to inversions is solvable in polynomial-time. In particular, if we let $N(A, B)$ be the minimum number of inversion operations need to transform genome A into homologous genome B, then for some integer k, $N(A, B)$ can be computed in $O(n^k)$ where n is the number of nucleotides in either genome. However, knowing $N(A, B)$ does not necessarily tell us which sequence of $N(A, B)$ edits can transform A into B; nor does it tell use how many such sequences exist. This question has been explored in [4] and in [9], and is the entry point for this thesis were we continue to explore this question as well as some related questions about sampling and complexity.

1.3 Breakpoint Graphs

In this section we introduce breakpoint graphs. This construction comes from [20]; a simplified version of the argument can be found in [3]. We begin by observing that, in the restricted view of evolution via reversals, a pair of genome can be considered homologous only if they contain the same set of nucleotides but appearing in different orders and in different orientations. Thus, given a pair of homologous genome with n nucleotides, we can label the nucleotides of one with the positively-signed identity permutation:

$$\mathrm{id} = (+1, +2, \ldots, +n)$$

and the other with a signed permutation

$$\sigma = (\pm\sigma_1, \pm\sigma_2, \ldots, \pm\sigma_n)$$

where σ_i is positive if and only if the nucleotide it represents is oriented in the same direction as the nucleotide that corresponds to it in the identity labeled genome.

Example 1.1. Suppose we have a (very simplified) pair of homologous genomes, A and B, as illustrated in Figure 1.1.

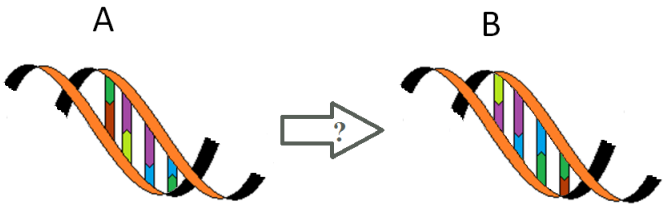

Figure 1.1 Homologous Genomes

We will label the 4 entries of genome B by $(+1, +2, +3, +4)$. Genome A is then represented with the signed permutation $(+4, -1, +2, +3)$. The genome rearrangement problem is to find a minimum number of evolutionary events (in this case reversals) to transform A into B, let that number be $N(A, B)$. The following set of permutation reversals demonstrate that $N(A, B) \leq 5$:

$$\begin{aligned}
(+4, \underline{-1, +2}, +3) &\rightsquigarrow (+4, -2, +1, +3) \\
(\underline{+4, -2, +1}, +3) &\rightsquigarrow (-1, +2, -4, +3) \\
(\underline{-1}, +2, -4, +3) &\rightsquigarrow (+1, +2, -4, +3) \\
(+1, +2, -4, \underline{+3}) &\rightsquigarrow (+1, +2, -4, -3) \\
(+1, +2, \underline{-4, -3}) &\rightsquigarrow (+1, +2, +3, +4)
\end{aligned}$$

The previous example illustrates that it is possible to upper-bound the number of inversions needed to transform one genome into another, however, these upper-

bounds may be far from optimal. This lead to the creation of overlap graphs. We begin first by replacing each entry of σ with a pair of entries according to the following rule: if σ_i is positively signed then replace σ_i with $2 \cdot \sigma_i - 1, 2 \cdot \sigma_i$, otherwise, when σ_i is negatively signed we replace σ_i with $2 \cdot \sigma_i, 2 \cdot \sigma_i - 1$.

Example 1.2. Suppose we have the same genomes as in the previous example. Genome A was represented with $(+4, -1, +2, -3)$ is now represented with $(7, 8, 2, 1, 3, 4, 6, 5)$. Genome B was represented with $(+1, +2, +3, +4)$ is now represented with $(1, 2, 3, 4, 5, 6, 7, 8)$. The reversal process in the previous example now becomes:

$$\begin{aligned}
(7, 8, \underline{2, 1, 3, 4}, 5, 6) &\rightsquigarrow (7, 8, 4, 3, 1, 2, 5, 6) \\
(\underline{7, 8, 4, 3, 1, 2}, 5, 6) &\rightsquigarrow (2, 1, 3, 4, 8, 7, 5, 6) \\
(\underline{2, 1}, 3, 4, 8, 7, 5, 6) &\rightsquigarrow (1, 2, 3, 4, 8, 7, 5, 6) \\
(1, 2, 3, 4, 8, 7, \underline{5, 6}) &\rightsquigarrow (1, 2, 3, 4, 8, 7, 6, 5) \\
(1, 2, 3, 4, \underline{8, 7, 6, 5}) &\rightsquigarrow (1, 2, 3, 4, 5, 6, 7, 8)
\end{aligned}$$

Observe that in every reversal the pairs of the form $\{2k-1, 2k\}$ are not separated at any point. The permitted reversals now are any reversals over a contiguous interval that has an even number of permutation entries to the left and to the right of it.

We will now proceed to describe how to build the breakpoint graphs from a pair of permutations. Let σ and τ be unsigned permutations. By relabeling we may assume that τ is the identity.

$$\tau = (1, 2, \ldots, 2n) \quad \text{and} \quad \sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{2n})$$

We will construct a plane graph (a graph embedded into the plane) with vertex set $V = \{0, 1, 2, \ldots, 2n, 2n+1\}$ drawn, in lex order, along a straight line (say the x-axis). Let $A = \{\{2k, 2k+1\} \mid 0 \leq k \leq n\}$ and $B = \{\{\sigma_{2k}, \sigma_{2k+1}\} \mid 0 \leq k \leq n\}$ where $\sigma_0 = 0$

and $\sigma_{2n+1} = 2n + 1$. Let the edge set of the graph be $E = A \cup B$, draw the edges of A as arcs above the x-axis with constant curvature, and draw the edges of B in a rectilinear fashion.

Example 1.3. Consider the permutations $\sigma = (7, 8, 2, 1, 3, 4, 5, 6)$ and $\tau = (1, 2, 3, 4, 5, 6, 7, 8)$. The breakpoint graph of σ and τ (Figure1.2) has vertex set $\{0, 1, \ldots, 9\}$, rectilinear edges $\{\{0, 7\}, \{8, 2\}, \{1, 3\}, \{4, 5\}, \{6, 9\}\}$ and curved arcs $\{\{0, 1\}, \{2, 3\}, \{4, 5\}, \{6, 7\}, \{8, 9\}\}$. Observe that arcs $a_1 = \{x, x + 1\}$ and $a_2 = \{y, y + 1\}$ will cross in the breakpoint graph if exactly one of x and $x + 1$ appear between y and $y+1$ in σ. Equivalently, if exactly one of y and $y+1$ appear between x and $x+1$ in σ. Any proper drawing of the breakpoint graph will have 4 arc crossings (crossings of the interior portion of the arcs).

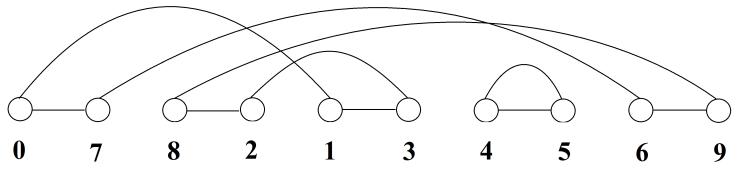

Figure 1.2 The Breakpoint Graph of σ and τ

The problem of sorting signed permutations can be translated into performing a sequence of operations on the breakpoint graph which transform it into the identity breakpoint graph which has only trivial cycles. Before proceeding we introduce some terminology associated with breakpoint graphs.

Definition 1.4. A *breakpoint* in a permutation occurs whenever two adjacent elements are non-consecutive.

Definition 1.5. The *cycle number* of a breakpoint graph, $c(G)$, is the number of cycles in any proper drawing of the breakpoint graph. The breakpoint graph in Figure 1.2 has cycle number $c = 2$.

Definition 1.6. The orientation of an arc is determined by the parity of the sum of their respective position. In particular, the arc $a = \{x, y\}$ is *positively oriented* if the sum of the number of vertices drawn to the left of x and y is even; otherwise we say that the arc is *negatively oriented*.

Observe that the goal of sorting signed permutations is to eliminate all breakpoints (implying that all the edges are positively oriented) by using the least number of reversals. In [2], Bafna and Pevzner showed that the reversal distance, $d(G)$, in a breakpoint graph (the minimum number of reversals needed to transform the graph into the identity graph) satisfies the inequality $d(G) \geq |V(G)|+1-c(G)$. Equivalently, $d(G) \geq b(G) - c'(G)$ where $c'(G)$ is the number of cycles that contain four or more vertices.

Definition 1.7. A cycle in a breakpoint graph is said to be an *hurdle* if all of its arcs are negatively oriented.

In [20], Hannenhali and Pevzner showed that the reversal distance, $d(G)$, in a breakpoint graph is

$$d(G) = b(G) - c(G) + h(G) + f(G)$$

where $b(G)$ is the number of breakpoints in the non-identity permutation, $c(G)$ is the cycle number of of the graph, $h(G)$ is the number of hurdles in the graph, and $f(G) \in \{0, 1\}$ is an indicator of he number of fortresses in a graph (we will not define fortresses here except to say that they are a sort of hurdle that protects another hurdle). The argument that Hannenhali and Pevzner used relies on the construction of overlap graphs from breakpoint graphs.

1.4 Overlap Graphs

From a breakpoint graph we may construct an overlap graph in the following manner. Let $B = (V(B), E(B))$ be a breakpoint graph with $V(B) = \{0, 1, \ldots, 2n+1\}$. Consider that B is already embedded into the plane. Let $G = (V, E)$ be a bicolored graph with $V = \{v_0, v_1, \ldots, v_n\}$ and $\{v_i, v_j\} \in E$ if arcs $a_i = \{2i, 2i+1\}$ and $a_j = \{2j, 2j+1\}$ cross in the drawing of B. Color $v \in V$ blue if $\deg_G(v) \equiv 1 \pmod 2$.

Example 1.8. Consider the permutations $\sigma = (7, 8, 2, 1, 3, 4, 5, 6)$ and $\tau = (1, 2, 3, 4, 5, 6, 7, 8)$. Let B be the breakpoint graph of σ and τ (see Figure 1.2). The overlap graph of σ and τ is illustrated in Figure 1.3.

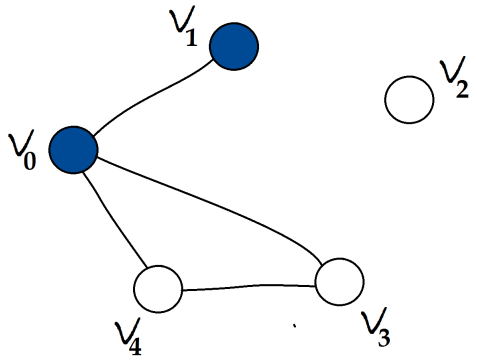

Figure 1.3 The Overlap Graph of σ and τ

In the following chapter we will introduce simple pseudo-graphs which are the generalization of overlap graphs. The operation of pressing in a simple pseudo-graph corresponds to making a reversal in a breakpoint graph, and a successful pressing sequence corresponds to a most-parsimonious sequence of reversals that transform on permutation into the other.

Chapter 2

Uniquely Pressable Graphs

2.1 Introduction

A signed permutation is an integer permutation where each entry is given a sign: plus or minus. A reversal in a signed permutation is the operation of reversing a subword and flipping the signs of the subword's entries. The primary computational problem of sorting signed permutations by reversals is to find the minimum number of reversals needed to transform a signed permutation into the positive identity permutation. Hannenhalli and Pevzner showed that the signed sorting problem can be solved in polynomial time (see [3, 20]) in contrast to the problem of sorting unsigned permutations, which is known to be NP-hard in general (see, e.g., [6]). At the core of the analysis given in [20] is the study of "successful pressing sequences" on vertex 2-colored graphs, which we refer to as "bicolored" graphs. In [9], the authors discuss the existence of a number of nonisomorphic such graphs which have exactly one pressing sequence, the "uniquely pressables". In this paper we use a combination of graph theory and combinatorial matrix algebra over $\mathbb{F}_2$ to characterize and count the set of uniquely pressable bicolored graphs.

This topic originated in computational phylogenetics, where Hannenhalli and Pevzner showed that certain bicolored graphs correspond to pairs of genomes and that the reversal distance between these genomes is the minimum length of a successful pressing sequence of said graph. In this interpretation, uniquely pressable graphs correspond to pairs of genomes linked by a unique minimum-distance putative evo-

lutionary history. The connection between permutation sorting and phylogenetics dates back to at least the 1930's, when two biologists, Dobzhansky and Sturtevant, wrote a series of papers in which they argued that the relationships between possible gene arrangements within a given chromosome encode critical information about the evolutionary history of species containing those genomes (see, e.g., [12, 27]). In particular, they introduced the idea that the degree of disorder between the genes in two genomes is an indicator of the evolutionary distance between two organisms. This has inspired extensive work in the fields of computational biology, bio-informatics and phylogenetics (see, e.g., [14]). In particular, researchers have pursued the question of how a common ancestral genome may have been transformed by evolutionary events into distinct, yet homologous, genomes. In mathematics and computer science, we often represent genomes as signed permutations (since DNA is double-stranded and hence oriented) and evolutionary events are encoded as operations on signed permutations. Among the most studied operations are block transpositions (e.g., [2, 17, 30]), prefix-reversals (e.g., [7, 8, 15, 18]), and reversals (e.g., [1, 23, 16, 17, 20, 26, 28]), all of which correspond to common evolutionary mechanisms.

We continue the study of sorting by reversals by investigating its graph-theoretic generalization, "pressing sequences", in graphs. Previous work in the area (see [4, 9, 14, 20]) has employed the language of black-and-white vertex-colored graphs in discussing successful pressing sequences. For various reasons (such as simplifying definitions and notation), we find it more convenient to replace the black/white vertex-coloring with looped/loopless vertices. Thus, the object of study will be simple pseudo-graphs: graphs that admit loops but not multiple edges (sometimes known as "loopy graphs"). However, for the purposes of illustration we borrow the convention from [4, 9] and elsewhere that the loops of a simple pseudo-graph are drawn as black vertices. Given a simple pseudo-graph G, denote by $V(G)$ the vertex set of G; $E(G) \subseteq V(G) \times V(G)$, symmetric as a relation, its edge set;

and $G[S] = (S, (S \times S) \cap E(G))$ the induced subgraph of a set $S \subset V(G)$. Let $N(v) = N_G(v) = \{w \in V(G) : vw \in E(G)\}$ the neighborhood of v in $V(G)$. Observe that $v \in N(v)$ iff v is a looped vertex.

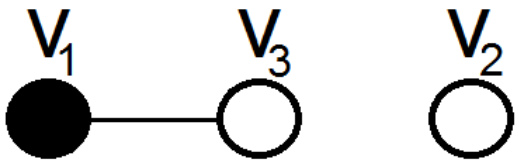

Figure 2.1 A simple pseudo-graph:
$G = (\{v_1, v_2, v_3\}, \{v_1v_1, v_1v_3\})$

After this introduction, the discussion is arranged into four sections. In Section 2, we develop some terminology and notation, and give a useful matrix factorization which we refer to as the "instructional Cholesky factorization" of a matrix (over $\mathbb{F}_2$), and discuss some of its properties. In Section 3, we present our main result, Theorem which characterizes the uniquely pressable graphs as those whose instructional Cholesky factorizations have a certain set of properties. In Section 4 we explore some consequences of the main theorem, such as the existence of a cubic-time algorithm for recognizing a uniquely pressable graph, a method for generating the uniquely pressable graphs by iteratively appending vertices to the beginning or end of a pressing sequence, and a counting argument which shows that there exist, up to isomorphism, exactly $(3 - (-1)^n)/2 \cdot 3^{\lfloor n/2 \rfloor - 1}$ uniquely pressable graphs on n non-isolated vertices. In the final section we discuss some open questions in this area. Before proceeding to Section 2 we list some basic notation for later use. Other terminology/notation employed below can be found in [5] or [11].

- We often write xy to represent the edge $\{x, y\}$ for concision. In particular if $x = y$ then xy is a loop.

- $[n] := \{1, 2, \ldots, n\}$ and $[k, n] := \{k, k+1, \ldots, n\}$ for all $k, n \in \mathbb{N}$.

- When $S = V(G) \setminus \{x\}$ we write the induced subgraph of G on S, $G[S]$, as $G - x$. In general, $G - S$ denotes $G[V(G) \setminus S]$.
- For integers x and y, $x \equiv y \pmod 2$ is abbreviated as $x \equiv y$.
- For a square matrix M with rows and columns identically indexed by a set X, for all $x \in X$, $M_{\hat{x}}$ denotes the submatrix of M with row and column x removed.
- When $\{x_\lambda\}_{\lambda \in \Lambda} \subset \mathbb{F}_2 \cup \mathbb{Z}$, the notation $\sum_{\lambda \in \Lambda} x_\lambda$ denotes addition over $\mathbb{Z}$ of $\overline{x}_\lambda$, where $\overline{x}_\lambda = x_\lambda$ if $x_\lambda \in \mathbb{Z}$ and $\overline{x}_\lambda$ is the least non-negative integer representation of x_λ in $\mathbb{Z}$ if $x_\lambda \in \mathbb{F}_2$. When referring to addition modulo 2 we use symbols $\oplus$ and $\bigoplus$. For example, if $x_1 = x_2 = 1 \in \mathbb{F}_2$ and $x_3 = 3 \in \mathbb{Z}$ then

$$\sum_{i=1}^{3} x_i = 1 + 1 + 3 = 5 \quad \text{and} \quad \bigoplus_{i=1}^{3} x_i = 1 \oplus 1 \oplus 1 = 1.$$

2.2 Pressing and Cholesky Roots

Definition 2.1. Consider a simple pseudo-graph G with a looped vertex $v \in V(G)$. "Pressing v" is the operation of transforming G into G', a new simple pseudo-graph in which $G[N(v)]$ is complemented. That is,

$$V(G') = V(G), \quad E(G') = E(G) \triangle (N(v) \times N(v))$$

We denote by $G_{(v)}$ the simple pseudo-graph resulting from pressing vertex v in $V(G)$ and we abbreviate $G_{(v_1)(v_2)\cdots(v_k)}$ to $G_{(v_1,v_2,\ldots,v_k)}$. For $k \geq 1$ we abbreviate $(1, 2, \ldots, k)$ as $\boldsymbol{k}$ so that when $V(G) = [n]$ for some $n \geq k$ then we may simplify $G_{(1,2,\ldots,k)}$ to $G_{\boldsymbol{k}}$. $G_{\mathbf{0}}$ and $G_{()}$ are interpreted to mean G.

Given a simple pseudo-graph G, $(v_1, v_2, \ldots, v_j)$ is said to be a *successful pressing sequence* for G whenever the following conditions are met:

- $\{v_1, v_2, \ldots, v_k\} \subseteq V(G)$,
- v_i is looped in $G_{(v_1,v_2,\ldots,v_{i-1})}$ for all $1 \leq i \leq k$,

- $G_{(v_1,v_2,\ldots,v_k)} = (V(G), \emptyset)$

In other words, looped vertices are pressed one at a time, with "success" meaning that the end result (when no looped vertices are left) is an empty graph. From the definition of "pressing" v we see that once a vertex is pressed it becomes isolated and cannot reappear in a valid pressing sequence. It was shown in [9] that if $G_{(v_1,v_2,\ldots,v_k)} = (V(G), \emptyset) = G_{(v'_1,v'_2,\ldots,v'_{k'})}$ then $k = k'$, i.e., the length of all successful pressing sequences for G are the same. We refer to this length k as the *pressing length of* G.

Definition 2.2. An *ordered simple pseudo-graph*, abbreviated OSP-graph, is a simple pseudo-graph with a total order on its vertices. In this paper, we will assume that the vertices of an OSP-graph are subsets of the positive integers under the usual ordering "$<$". An OSP-graph G is said to be *order-pressable* if there exists some initial segment of $V(G)$ that is a successful pressing sequence, that is, if it admits a successful pressing sequence $(v_1, v_2, \ldots, v_k)$ satisfying $v_1 < v_2 < \cdots < v_k$ and $v_k < v'$ for all $v' \in V(G) \setminus \{v_1, v_2, \ldots, v_k\}$. An OSP-graph G is said to be *uniquely pressable* if it is order-pressable and G has no other successful pressing sequence.

Lemma 2.3. *If G is a connected OSP-graph that is uniquely pressable then the pressing length of G is $|V(G)|$.*

Proof. Suppose not. Then at some point in the pressing sequence two vertices become isolated and unlooped by one press. Call these vertices a and b, and suppose a is the vertex pressed. Then at this point in the pressing sequence a and b must have identical neighborhoods and hence b can replace a in the pressing sequence, contradicting that the pressing sequence is unique. □

We say a component of G is trivial if it is a loopless isolated vertex.

Proposition 2.4. *[9] A simple pseudo-graph G admits a successful pressing sequence if and only if every non-trivial component of G contains a looped vertex.*

Corollary 2.5. *If G is a uniquely pressable OSP-graph with at least one edge then G contains exactly one non-trivial component C and the pressing length of G is $|V(C)|$.*

Proof. Let $G = ([n], E)$. Let C_1 and C_2 be (possibly distinct) non-trivial connected components of G. Let $C_3 = G - (V(C_1) \cup V(C_2))$ so that G is the (possibly disjoint) union of C_1, C_2 and C_3. As G is uniquely pressable it has unique pressing sequence $\sigma = \boldsymbol{n}$. Observe that pressing a vertex only makes changes to its closed neighborhood, a set which is contained within a single connected component. Let σ_i be the restriction of σ to the vertices of C_i, $i = 1, 2, 3$. Then σ_i is a successful pressing sequence for $G[C_i]$. If $C_1 \neq C_2$ then G is the disjoint union of $G[C_1]$, $G[C_2]$ and $G[C_3]$, where the last one may be empty. Then pressing the vertices of $G[C_2]$ followed by pressing the vertices of $G[C_1]$ followed by pressing the vertices of $G[C_3]$ gives a successful pressing sequence for G, contradicting the uniqueness of σ. It follows that $C_1 = C_2$ and $\sigma_3 = \emptyset$, and therefore G contains exactly one non-trivial connected component. □

This shows that in order to understand uniquely pressable OSP-graphs, it suffices to understand connected, uniquely pressable OSP-graphs.

Notation 2.6. $\mathbf{CUP}_n$ is the set of connected, uniquely pressable ordered ($<_{\mathbb{N}}$) simple pseudo-graphs on n positive integer vertices.

Definition 2.7. Given an OSP-graph $G = ([n], E)$ define the *adjacency matrix* $A = A(G) = (a_{i,j}) \in \mathbb{F}_2^{n \times n}$ by

$$a_{i,j} = \begin{cases} 1 & \text{if } ij \in E, \\ 0 & \text{otherwise.} \end{cases}.$$

Note that $A(G)$ is always symmetric. Previous work in the area refers to such matrices as *augmented* adjacency matrices as the diagonal entries are nonzero where

the vertices are colored black; since we have used looped vertices instead, the term "augmented" is not necessary. Define the *instructional Cholesky root* of G, denoted $U = U(G) = (u_{i,j}) \in \mathbb{F}_2^{n\times n}$, by

$$u_{i,j} = \begin{cases} 1 & \text{if } i \leq j \text{ and } j \in N_{G_{\boldsymbol{i-1}}}(i), \\ 0 & \text{otherwise.} \end{cases}.$$

Observe that the j^{th} row of U is given by the j^{th} row of the adjacency matrix of $G_{\boldsymbol{j-1}}$, and that $u_{i,j} = 1$ precisely when the act of pressing i during a successful pressing sequence of G flips the state of j. Thus, U provides detailed "instructions" on how to carry out the actual pressing sequence. In Proposition 2.10 we justify use of the name Cholesky.

Definition 2.8. For an order-pressable graph $G = ([n], E)$ with instructional Cholesky root $U = (u_{i,j})$ we define the *dot product of two vertices* $i, j \in V(G)$ as the dot product over $\mathbb{F}_2$ of the i^{th} and j^{th} columns of U:

$$\langle i, j\rangle_G = \bigoplus_{t=1}^{n} u_{t,i}u_{t,j}.$$

We define the *(Hamming) weight of a vertex* $j \in [n]$ by

$$\mathrm{wt}_G(j) = \sum_{t=1}^{n} u_{t,j}$$

and observe that $\mathrm{wt}_G(j) \equiv \langle j, j\rangle_G$.

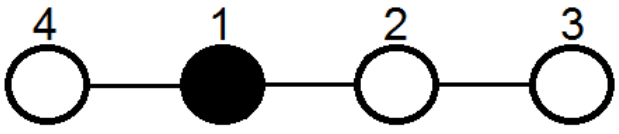

Figure 2.2 $\mathrm{wt}(1) = 1$, $\mathrm{wt}(2) = 2$, $\mathrm{wt}(3) = 2$, $\mathrm{wt}(4) = 4$, $\langle 1, 2\rangle = 1$, $\langle 1, 3\rangle = 0$, $\langle 1, 4\rangle = 1$, $\langle 2, 3\rangle = 1$, $\langle 2, 4\rangle = 0$, $\langle 3, 4\rangle = 0$

Definition 2.9. The *weight of a column* C in a matrix M, written $\mathrm{wt}_M(C)$, is the sum of the entries in column C (again, as elements of $\mathbb{Z}$).

Observe that if U is the instructional Cholesky root of G then the column weights of U correspond to the vertex weights of G.

Proposition 2.10. *If $G = ([n], E)$ is an OSP-graph with successful pressing sequence $1, 2, \ldots, k$, adjacency matrix A, and instructional Cholesky root U then $U^T U = A$.*

Proof. Let $U^T U = B = (b_{i,j})$ and $A = (a_{i,j})$. Observe that $b_{i,j}$ is the result (modulo 2) of dotting the i^{th} and j^{th} columns of U. Hence $b_{i,j} = \langle i, j \rangle_G$.

For $i, j \in [n]$, let $S_{i,j} = \{t \in [k] \colon ij \in E(G_{\boldsymbol{t-1}}) \triangle E(G_{\boldsymbol{t}})\}$ and $T_{i,j} = \{t \in [k] \colon ti \in E(G_{\boldsymbol{t-1}})$ and $tj \in E(G_{\boldsymbol{t-1}})\}$.

Observe that $S_{i,j}$ lists the times during the pressing sequence that the pressed vertex results in the state of edge ij being flipped. This occurs if and only if both i and j are in the neighborhood of the vertex being pressed. Hence $S_{i,j} = T_{i,j}$. The state of the edge/non-edge ij in $G_{\boldsymbol{k}}$ is determined by its original state in G and by the number of times the state of the edge/non-edge ij was flipped during the pressing sequence. However, $G_{\boldsymbol{k}} = ([n], \emptyset)$ so $ij \notin E(G_{\boldsymbol{k}})$ and therefore the number of times that the state of the edge/non-edge ij is flipped during the pressing sequence must agree in parity to with the original state of the edge/non-edge ij. It follows that $|S_{i,j}| \equiv a_{i,j}$. On the other hand $T_{i,j} = \{t \in [n] \colon u_{t,i} = u_{t,j} = 1\}$ list the common 1's in columns i and j of the instructional Cholesky root. Hence $|T_{i,j}|$ has the same parity as dotting the i^{th} and the j^{th} column of U. It follows that $b_{i,j} = \langle i, j \rangle_G \equiv |T_{i,j}| = |S_{i,j}| \equiv a_{i,j}$.

Since the matrix entries are elements of $\mathbb{F}_2$, we have $b_{i,j} = a_{i,j}$ for $i, j \in [n]$ and therefore $U^T U = A$. $\square$

Observation 2.11.

Given an OSP-graph G with adjacency matrix A and instructional Cholesky root U: $ij \in E(G)$ if and only if $\langle i, j\rangle_G = 1$, since $a_{i,j}$ is the result of dotting the i^{th} and j^{th} columns of U. In particular i is looped in G if and only if $\mathrm{wt}_G(i) \equiv 1$.

In the theory of complex matrices, decompositions of the form $A = U^T U$ are known as "Cholesky" factorizations, so we repurpose this terminology here. While a symmetric full-rank matrix over $\mathbb{F}_2$ has a unique Cholesky decomposition (see [9]), a matrix $M \in \mathbb{F}_2^{n\times n}$ of less than full rank may have more than one Cholesky decomposition. On the other hand, the adjacency matrix A of an OSP-graph G with successful pressing sequence $1, 2, \ldots, k$ has a unique instructional Cholesky root U as the first k rows are determined by the sequence of graphs $G, G_{\mathbf{1}}, G_{\mathbf{2}}, \ldots, G_{\boldsymbol{k-1}}$ and the remaining rows (should they exist) are all zero. Throughout the paper we will take advantage of this by referring interchangeably to a pressable OSP-graph G, its (ordered) adjacency matrix A, and its instructional Cholesky root U.

Example 2.12. Consider the $\mathbb{F}_2^{5\times 5}$ matrix M and its (unique) instructional Cholesky root U:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The following matrices also offer Cholesky factorizations for M:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Definition 2.13. Consider a pressable OSP-graph $G = (V, E)$ where $V = \{v_i\}_{i\in[n]}$ has the order implied by its indexing. For each $j \in [n]$ we say that v_j has *full weight in* G provided $\mathrm{wt}_G(v_j) = j$.. In particular if $V(G) = [n]$ under the usual ordering then vertex j has full weight if and only if $\mathrm{wt}_G(j) = j$ if and only if $j \in N_{G_{i-1}}(i)$ for all $i \in [j]$.

The following notation will be used to simplify inductive arguments.

Notation 2.14. For a given OSP-graph $G = ([n], E)$ with looped vertex j denote by $G^{(j)} = G_{(j)} - j$ the result of pressing vertex j and then deleting it from the vertex set. Furthermore we let G^j denote the result of pressing and deleting vertices $1, 2, \ldots, j$ in order from G.

Lemma 2.15. *If $G \in \mathbf{CUP}_n$ has instructional Cholesky root U then $G^{\mathbf{1}} \in \mathbf{CUP}_{n-1}$ and the instructional Cholesky root of $G^{\mathbf{1}}$ is $U_{\hat{1}}$.*

Proof. Let $G = ([n], E) \in \mathbf{CUP}_n$ with instructional Cholesky root U. The unique successful pressing sequence of G is $\boldsymbol{n}$ which is realized by

$$G, G_{\mathbf{1}}, G_{\mathbf{2}}, \ldots, G_{\boldsymbol{n}}$$

and hence $G_{\mathbf{1}}$ admits a successful pressing sequence: $2, 3, \ldots, n$. Furthermore, if $(v_1, v_2, \ldots, v_{n-1})$ is a successful pressing sequence of $G_{\mathbf{1}}$, then $(1, v_1, v_2, \ldots, v_{n-1})$ is a successful pressing sequence of G. By uniqueness it follows that $v_i = i + 1$ for each $i \in [n-1]$. Then $G_{\mathbf{1}}$ admits exactly one successful pressing sequence $2, 3, \ldots, n$, and therefore so does $G^{\mathbf{1}}$. It follows that $G^{\mathbf{1}} \in \mathbf{CUP}_{n-1}$. Let V be the instructional

Cholesky root of $G^{\mathbf{1}}$. The first row of $G^{\mathbf{1}}$ is given by the neighborhood of 2 in $G_{\mathbf{1}}$ and in general the j^{th} row of V is given by $N_{G^{\mathbf{1}}(2,3,\ldots,j)}(j+1)$. However

$$N_{G^{\mathbf{1}}(2,3,\ldots,j)}(j+1) = N_{G_{\boldsymbol{j}}}(j+1)$$

for each $j \in [n-1]$. Therefore the j^{th} row of V is the $(j+1)^{th}$ row of U with the first entry deleted, since 1 is not a vertex in $G^{\mathbf{1}}$. V is the principal submatrix of U restricted to rows and columns $2, 3, \ldots, n$. □

Proposition 2.16. *[9] An OSP-graph $G = ([n], E)$ has pressing sequence $\boldsymbol{n}$ if and only if every leading principal minor of its adjacency matrix is nonzero.*

Lemma 2.17. *Let $G = ([n], E) \in \mathbf{CUP}_n$ with instructional Cholesky root U and let $H = G - n$ be the induced subgraph of G on $[n-1]$. Then $H \in \mathbf{CUP}_{n-1}$ and the instructional Cholesky root of H is $U_{\hat{n}}$.*

Proof. If $n = 1$ then $H = (\emptyset, \emptyset)$ which has only the empty sequence as a successful pressing sequence and its instructional Cholesky root is the empty matrix. Let $n > 1$. Observe that $N_H(1) = N_G(1) - \{n\}$ so 1 is a looped vertex in H and therefore may be pressed to obtain $H_{\mathbf{1}}$. For all $j \in [n-1]$:

$$\begin{aligned}
N_{H_{\mathbf{1}}}(j) &= \begin{cases} N_H(j) \triangle N_H(1), & 1j \in E(H) \\ N_H(j), & 1j \notin E(H) \end{cases} \\
&= \begin{cases} (N_G(j) \triangle N_G(1)) - \{n\}, & 1j \in E(H) \leftrightarrow 1j \in E(G) \\ N_G(j) - \{n\}, & 1j \notin E(H) \leftrightarrow 1j \notin E(G) \end{cases} \\
&= \begin{cases} N_{G_{\mathbf{1}}}(j) - \{n\}, & 1j \in E(G) \\ N_{G_{\mathbf{1}}}(j) - \{n\}, & 1j \notin E(G) \end{cases}
\end{aligned}$$

Assume $H_{\boldsymbol{i}} = G_{\boldsymbol{i}} - n$ for some $1 \leq i < n-1$. Then $i+1$ is looped in $G_{\boldsymbol{i}}$, implying that it is looped in $H_{\boldsymbol{i}}$, so for all $j \in [n-1]$:

$$
\begin{aligned}
N_{H_{\boldsymbol{i+1}}}(j) &= \begin{cases} N_{H_{\boldsymbol{i}}}(j) \triangle N_{H_{\boldsymbol{i}}}(i+1), & \{j, i+1\} \in E(H_{\boldsymbol{i}}) \\ N_{H_{\boldsymbol{i}}}(j), & \{j, i+1\} \notin E(H_{\boldsymbol{i}}) \end{cases} \\
&= \begin{cases} (N_{G_{\boldsymbol{i}}}(j) \triangle N_{G_{\boldsymbol{i}}}(i+1)) - \{n\}, & \{j, i+1\} \in E(G_{\boldsymbol{i}}) \\ N_{G_{\boldsymbol{i}}}(j) - \{n\}, & \{j, i+1\} \notin E(G_{\boldsymbol{i}}) \end{cases} \\
&= \begin{cases} N_{G_{\boldsymbol{i+1}}}(j) - \{n\}, & \{j, i+1\} \in E(G_{\boldsymbol{i}}) \\ N_{G_{\boldsymbol{i+1}}}(j) - \{n\}, & \{j, i+1\} \notin E(G_{\boldsymbol{i}}) \end{cases}
\end{aligned}
$$

By induction it follows that $\boldsymbol{n-1}$ is a valid pressing sequence for H and $H_{\boldsymbol{i}} = G_{\boldsymbol{i}} - \{n\}$ for all $i \in [n-1]$. We proceed to show that $\boldsymbol{n-1}$ is the only successful pressing sequence for H. Let A be the adjacency matrix of G (under the ordering $\boldsymbol{n}$) and let U be its instructional Cholesky root. Let $\sigma = (v_1, v_2, \ldots, v_{n-1})$ be a valid pressing sequence for H and let $\tau = (v_1, v_2, \ldots, v_{n-1}, n)$. Let P be the permutation matrix that encodes τ. Then $A_{\hat{n}}$ is the adjacency matrix of H under the usual ordering $<$ and $P_{\hat{n}} A_{\hat{n}} {P_{\hat{n}}}^T$ is the adjacency matrix of H under the ordering given by σ. Let V be the instructional Cholesky root of H under σ. Observe that by Proposition 2.16, $\det(A) \neq 0$ and so $\det(PAP^T) = \det(P)\det(A)\det(P^T) = \det(A) \neq 0$. Furthermore,

$$
PAP^T = \left[\begin{array}{ccc|c} & P_{\hat{n}} & & \begin{matrix}0\\\vdots\\0\end{matrix} \\ \hline 0 & \cdots & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} & A_{\hat{n}} & & \begin{matrix}*\\\vdots\\*\end{matrix} \\ \hline * & \cdots & * & 1 \end{array}\right] \left[\begin{array}{ccc|c} & P_{\hat{n}} & & \begin{matrix}0\\\vdots\\0\end{matrix} \\ \hline 0 & \cdots & 0 & 1 \end{array}\right]^T
$$

$$
= \left[\begin{array}{ccc|c} & P_{\hat{n}} A_{\hat{n}} {P_{\hat{n}}}^T & & \begin{matrix}*\\\vdots\\*\end{matrix} \\ \hline * & \cdots & * & * \end{array}\right] = \left[\begin{array}{ccc|c} & V^T V & & \begin{matrix}*\\\vdots\\*\end{matrix} \\ \hline * & \cdots & * & * \end{array}\right].
$$

Recall that H has $\boldsymbol{n-1}$ as a successful pressing sequence and so every successful pressing sequence must have length $n-1$. It follows that all the diagonal entries of (upper/lower-triangular matrices) V and V^T must be 1, implying that every leading principal minor of PAP^T is non-zero. By Proposition 2.16, τ is a successful pressing sequence for G. By uniqueness $\tau = \boldsymbol{n}$ and hence $\sigma = \boldsymbol{n-1}$. We may conclude that $H \in \mathbf{CUP}_{n-1}$. □

Corollary 2.18. *Let $G \in \mathbf{CUP}_n$ with instructional Cholesky root U. Then any principal submatrix of U on k consecutive rows and columns is the instructional Cholesky root of a $\mathbf{CUP}_k$ graph.*

Proof. Follows by iteratively applying Lemmas 2.15 and 2.17. □

Corollary 2.19. *If U is the instructional Cholesky root of $G \in \mathbf{CUP}_n$ then U must have all 1's on the main diagonal and super-diagonal.*

Proof. Let $H = ([2], E) \in \mathbf{CUP}_2$. Since it is connected, $\{1,2\} \in E$; since it is order-pressable, 1 must be looped; and since it is uniquely pressable, $N_H(1) \neq N_H(2)$. Therefore $\mathbf{CUP}_2 = \{([2], \{\{1,1\},\{1,2\}\})\}$ which corresponds to instructional Cholesky root $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. The result holds by application of Corollary 2.18. □

2.3 Characterizing Unique Pressability

Definition 2.20. For an upper-triangular matrix $M \in \mathbb{F}_2^{n\times n}$ with columns C_1, C_2, …, C_n with respective column weights $w_1, w_2, \ldots, w_n$, we say:

- $C_j = (c_{1,j}, c_{2,j}, \ldots, c_{n,j})^T$ has *Property 1* if $\begin{cases} c_{i,j} = 1, & j - w_j < i \leq j \\ c_{i,j} = 0, & \text{otherwise} \end{cases}$.
- M has *Property 1* if each of its columns have Property 1.
- M has *Property 2* if $1 = w_1 \leq w_2 \leq \cdots \leq w_n$

- M has *Property 3* if $w_i > 2$ implies $w_{i+2} > w_i$, for $i \in [n-2]$.
- M has *Property 4* if, whenever some non-initial column has odd weight, then it must have full weight and so must each column to its right.

In other words, Property 1 is the condition that the nonzero entries in each column are consecutive and end at the diagonal; Property 2 is the condition that the weights of the columns are nondecreasing; and Property 3 is the condition that any column must have weight greater than that of the column two indices to its left if the latter has weight more than 2. Note also that Property 4 implies that any even-indexed column must have even weight; otherwise, it would have full weight, i.e., weight equal to the column index, which is even, a contradiction. Let $\mathcal{M}_n = \{M \in \mathbb{F}_2^{n\times n} \mid M \text{ satisfies Properties 1, 2, 3 and 4}\}$. Observe that if $M \in \mathcal{M}_n$ then $M_{\hat{n}} \in \mathcal{M}_{n-1}$.

Lemma 2.21. *Let $n > 1$ and $M \in \mathbb{F}_2^{n\times n}$ with columns and rows indexed by $1, 2, \ldots, n$. If $M \in \mathcal{M}_n$ then $M_{\hat{1}} \in \mathcal{M}_{n-1}$.*

Proof. Let $M = (c_{i,j})_{i,j\in[n]}$ so that $M_{\hat{1}} = (c_{i,j})_{2\le i,j\le n}$. Let $w_1, w_2, .., w_n$ be the column weights of M. Let the columns and rows of $M_{\hat{1}}$ be $C_2, \ldots, C_n$ with weights $w'_2, \ldots, w'_n$, respectively. Observe that $w'_j = w_j - c_{1,j}$ for each $2 \le j \le n$. It is immediate that $M_{\hat{1}}$ inherits Property 1 from M. By Property 4 we know that the second column of M (as well as any even-indexed column of M) has even weight, it follows that $w_2 = 2$ and so $w'_2 = 2 - c_{1,2} = 1$. Suppose towards a contradiction $w'_i > w'_{i+1}$ for some $2 \le i \le n-1$. Then

$$w'_i + 1 > w'_{i+1} + 1 \ge w'_{i+1} + c_{1,i+1} = w_{i+1} \ge w_i \ge w'_i$$

and so $w'_i = w_i$. Then

$$w'_{i+1} < w'_i = w_i \le w_{i+1} = w'_{i+1} + c_{1,i+1} \le w'_{i+1} + 1$$

and so $w_{i+1} = w'_{i+1} + 1$. Hence we have

$$w_{i+1} = w'_{i+1} + 1 < w'_i + 1 = w_i + 1 \quad \Rightarrow \quad w_{i+1} \le w_i.$$

It follows that $w_{i+1} = w_i$ and therefore column $i+1$ does not have full weight. By Property 1 of M we have $c_{1,i+1} = 0$ and therefore $w'_{i+1} = w_{i+1}$, a contradiction. It follows that $M_{\hat{1}}$ has Property 2.

Suppose now that $2 < w'_j$ for some $2 \leq j \leq n-2$. Then $w_j \geq w'_j > 2$ so $w_{j+2} > w_j$ by Property 3 of M. If $w_j = w_{j+2} - 1$ then either column j or column $j+2$ has odd weight which implies $w_{j+2} = j+2$ by Property 4 of M. But then $w_j = j+1$ which is not possible. Therefore,

$$w_j < w_{j+2} - 1$$

and

$$w'_j \leq w_j < w_{j+2} - 1 \leq w'_{j+2}$$

which shows that $M_{\hat{1}}$ has Property 3. To show Property 4 suppose $w'_k \equiv 1$ for some $k > 2$ (2 is the initial column of $M_{\hat{1}}$). Observe that

$$w'_k + 1 \geq w_k \geq w'_k.$$

If $w_k = w'_k$ then $w_k \equiv 1$ and not full weight, contradicting Property 4. Hence

$$w_k = w'_k + 1$$

and $c_{1,k} = 1$. It follows from Property 1 that $w_k = k > 2$. Hence $w'_k > 1$ and since $w'_k \equiv 1$ then $w'_k \geq 3$ and $w_k \geq 4$.

By applying Property 3 of M we get $k+2 \geq w_{k+2} \geq w_k + 1 = k+1$. Since $1 \equiv w'_k = k-1$ we have that w_{k+2} is the weight of an even-indexed column, and so $w_{k+2} \equiv 0$ and $w_{k+2} = k+2$.

By arguing inductively, for all $j \in [\lfloor (n-k)/2 \rfloor]$:

$$0 \equiv k + 2j \geq w_{k+2j} \geq w_{k+2(j-1)} + 1 = k + 2j - 1 \equiv 1$$

hence $w_{k+2j} = k + 2j$.

It follows that for all $j \in [\lfloor (n-k)/2 \rfloor]$:

$$w'_{k+2j} = (k+2j) - c_{1,k+2j} = k + 2j - 1.$$

Furthermore, for all $j \in [0, \lceil (n-k-1)/2 \rceil]$, $w_{k+2j+1} \geq w_{k+2j} = k+2j$ and so $w_{k+2j+1} = k+2j+c_{1,k+2j+1}$. Therefore, for all $j \in [0, \lceil (n-k-1)/2 \rceil]$:

$$w'_{k+2j+1} = w_{k+2j+1} - c_{1,k+2j+1} = k+2j.$$

It follows that, in $M_{\hat{1}}$, all the columns of index at least k have full weight and therefore $M_{\hat{1}}$ has Property 4. □

Observe that the previous lemma can be extended to any matrix $M \in \mathcal{M}_n$ by relabeling the rows and columns. We now proceed to our main theorem, which characterizes the set $\mathbf{CUP}_n$. This in turn provides a characterization of all the uniquely pressable simple pseudo-graphs (up to isomorphism), since the unique non-trivial, connected component of a simple pseudo-graph can always be relabeled to be a **CUP** graph.

Notation 2.22. For an OSP-graph G let $\mathcal{L}(G) = \{v \mid v \in V(G) \text{ is a looped vertex}\}$.

Theorem 2.23. *Let $G = ([n], E)$ with instructional Cholesky root U. Then $G \in \mathbf{CUP}_n$ if and only if $U \in \mathcal{M}_n$.*

Proof. For $n = 1$ the conditions of $\mathcal{M}_1$ are only met by $G = ([1], \{(1,1)\})$ which in turn is the only full-length uniquely pressable OSP-graph on vertex set $[1]$. Let $n > 1$ and assume towards an inductive argument that the statement holds for $n-1$. We begin by showing sufficiency, that is if $U \in \mathcal{M}_n$ then $G \in \mathbf{CUP}_n$. Choose and fix $U = (u_{i,j}) \in \mathcal{M}_n$. Let $G = ([n], E)$ be the OSP-graph with instructional Cholesky root U. By Properties 1 and 2, $u_{i,i} = 1$ for each $i \in [n]$. This implies that vertex i is looped in $G_{\boldsymbol{i-1}}$ for each $i \in [n]$. It follows that $\boldsymbol{n}$ is a successful pressing sequence for G. We will show it is the only successful pressing sequence for G. Fix a successful pressing sequence $\sigma = \sigma_1, \ldots, \sigma_n$ for G. If $\sigma_1 = 1$ then $G^{(\sigma_1)}$ has adjacency matrix $A(G^{\mathbf{1}}) = U_{\hat{1}}^T U_{\hat{1}}$

By Lemma 2.21, $U_{\hat{1}} \in \mathcal{M}_{n-1}$ and therefore $G^{(\sigma_1)} \in \mathbf{CUP}_{n-1}$ by the inductive hypothesis. Hence $G^{(\sigma_1)}$ has exactly one pressing sequence, and, since $G^{(\sigma_1)} = G^{\mathbf{1}}$, the sequence is $(2, 3, \ldots, n)$. We may conclude that if $\sigma_1 = 1$ then $\sigma = \boldsymbol{n}$. Assume, by way of contradiction, that $\sigma_1 = t > 1$. We will show that $G^{(t)}$ contains a non-trivial loopless component and therefore is not pressable. Since t is a looped vertex it must have odd weight, and therefore full weight by Property 4. Let $k = \min_{2 \leq i \leq n} \{i \mid \mathrm{wt}_G(i) \equiv 1\}$ and let

$$L = [2, k-1], \quad R = [k, n], \quad \overline{L} = L \cup \{1\}, \quad \text{and} \quad \overline{R} = R \cup \{1\}.$$

By Property 4 of U, all the vertices in $\overline{R}$ have full weight. For all $i \in [n]$ and $r \in \overline{R}$:

$$\langle i, r \rangle_G = \bigoplus_{k=1}^{n} u_{k,i} u_{k,r} = \bigoplus_{k=1}^{r} (u_{k,i} \cdot 1) \oplus \bigoplus_{k=r+1}^{n} (u_{k,i} \cdot 0) = \min \{\mathrm{wt}_G(i), \mathrm{wt}_G(r)\}.$$

By Property 4, if $\mathrm{wt}_G(i) \equiv 1$ then $i \in \overline{R}$. It follows that for all $r \in \overline{R}$,

$$N_G(r) = \begin{cases} \mathcal{L}(G) \cap [r-1], & \text{if } r \equiv 0 \\ \mathcal{L}(G) \cup [r, n], & \text{if } r \equiv 1 \end{cases} \quad \subseteq \quad \mathcal{L}(G) \subseteq \overline{R}.$$

Then

$$\mathcal{L}\left(G^{(t)}\right) = \mathcal{L}(G) \triangle N_G(t) \subseteq \overline{R}.$$

However, $\langle 1, t \rangle = 1$ because t has full weight, so $1 \in \mathcal{L}(G) \cap N_G(t)$ and

$$\mathcal{L}\left(G^{(t)}\right) \subseteq R.$$

Similarly, for $r \in R$

$$N_{G^{(t)}}(r) = N_G(r) \triangle N_G(t) \subseteq R$$

since $1 \in N_G(r) \cap N_G(t)$. It follows that the induced subgraphs $G_{(t)}\left[\overline{L}\right]$ and $G_{(t)}[R]$ are contained not connected by a path in $G_{(t)}$. By applying Corollary 2.19, observe that $2 \notin N_G(t)$ and $2 \in N_G(1)$, so

$$N_{G^{(t)}}(1) = N_G(1) \triangle N_G(t) \ni 2,$$

and therefore $G_{(t)}\left[\overline{L}\right]$ contains an edge but no loops. It follows that $G^{(t)}$ does not admit a successful pressing sequence, a contraction. Therefore $\sigma_1 \not> 1$.

We now proceed to show necessity: if $G \in \mathbf{CUP}_n$, then $U \in \mathcal{M}_n$. Let $G \in \mathbf{CUP}_n$ with instructional Cholesky root $U = (u_{i,j})$. By Lemmas 2.15 and 2.17 we have that $G_{\mathbf{1}}, G - n \in \mathbf{CUP}_{n-1}$ and therefore by the inductive hypothesis $U_{\hat{1}}, U_{\hat{n}} \in \mathcal{M}_{n-1}$. For simplicity we let $H = G - n$ and $w_i = \mathrm{wt}_G(i)$ for $i \in V(G)$ throughout the rest of this proof. It suffices to show the following four conditions hold for U:

(I) Property 1 holds for the n^{th} column,

(II) $w_n \geq w_{n-1}$,

(III) If $w_{n-2} > 2$ then $w_n > w_{n-2}$,

(IV) If $w_n \neq n$ then the first column of M is the only one with odd weight.

We have four cases to consider.

First Case: $u_{1,n} = u_{2,n} = 1$**.** Since $u_{2,n} = 1$ then Property 4 of $U_{\hat{1}}$ gives us $u_{i,n} = 1$ for all $2 \leq i \leq n$. It follows that $w_n = n$ and therefore (I), (II), (III), and (IV) hold.

Second Case: $u_{1,n} = u_{2,n} = 0$. (I) holds by Property 1 of $U_{\hat{1}}$. Recall that the diagonal and super-diagonal entries of U must be 1 by Corollary 2.19 so we need not consider the case where $n \leq 3$. For $n = 4$ we have two matrices to consider,

$$V_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad V_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

V_1 satisfies (I) - (IV). $V_2 \notin \mathbf{CUP}_4$ since $(3, 4, 1, 2)$ is also a successful pressing sequence. Thus we may assume $n \geq 5$ to show (II), (III) and (IV) hold.

Claim 2.24. $u_{1,n-1} = 0$

Proof of Claim 2.24. Assume towards a contradiction that $u_{1,n-1} = 1$. Property 1 of $U_{\hat{n}}$ tells us that

$$w_{n-1} = \mathrm{wt}_{U_{\hat{n}}}(n-1) = n-1$$

and so

$$\mathrm{wt}_{U_{\hat{1}}}(n-1) = w_{n-1} - u_{1,n-1} = n-2.$$

By Property 4 of $U_{\hat{1}}$, since $u_{2,n} = 0$, then

$$\mathrm{wt}_{U_{\hat{1}}}(n-1) \equiv 0.$$

It follows that

$$w_{n-1} = n-1 \equiv 1.$$

Recalling that $u_{1,n} = u_{2,n} = 0$, by Property 2 of $U_{\hat{1}}$ we have

$$n-2 \geq \mathrm{wt}_{U_{\hat{1}}}(n) \geq \mathrm{wt}_{U_{\hat{1}}}(n-1) = n-2$$

and so

$$w_n = \mathrm{wt}_{U_{\hat{1}}}(n) = n-2 \equiv 0.$$

Let

$$k = \min_{1<i\leq n} \{i \mid w_i \equiv 1\}.$$

Since $G \in \mathbf{CUP}_n$ and $k \neq 1$ then $G^{(k)}$ is not a pressable graph. We will use this to arrive at a contradiction. Since $w_n \equiv 0$,

$$\mathcal{L}(G) = \mathcal{L}(H).$$

By the minimality of k, by Property 4 of $U_{\hat{n}}$, and since $n-1 \equiv 1$:

$$\mathcal{L}(H) = \{1\} \cup \{k, k+2, \ldots, n-1\}.$$

Observe that

$$\langle k,n\rangle_G = \bigoplus_{i=1}^{n} u_{i,k}u_{i,n} = \bigoplus_{i=1}^{2}(u_{i,k}\cdot 0)\oplus\bigoplus_{i=3}^{k}(1\cdot 1)\oplus\bigoplus_{i=k+1}^{n}(0\cdot u_{i,n}) \equiv k-2 \equiv 1$$

and for $i \in [n-1]$

$$\langle k,i\rangle_G = \langle k,i\rangle_H \equiv \min(\mathrm{wt}_H(k),\mathrm{wt}_H(i))$$

and so

$$N_G(k) = \{1\}\cup[k,n].$$

It follows that

$$\mathcal{L}(G^{(k)}) = \mathcal{L}(G)\triangle N_G(k) = \{k+1,k+3,\ldots,n-2,n\}.$$

Since $N_G(k)\cap[k-1] = \{1\}$, the only potential edge in $G[[k-1]]$ affected by pressing k is the loop on 1. Furthermore $G[[k-1]] \in \mathbf{CUP}_{k-1}$ by Corollary 2.18 so we may conclude that $G^{(k)}[[k-1]]$ is connected. If $k \neq n-1$ then $\langle k+1,n\rangle_G \equiv k-1 \equiv 0$ so

$$N_G(k+1) = \{1,k\}$$

and so

$$N_{G^{(k)}}(k+1) = N_G(k)\triangle N_G(k+1) = [k+1,n].$$

Then it follows that $G^{(k)}[[k+1,n]]$ is a connected graph with looped vertex $(k+1)$. Observe that

$$\langle 1,n\rangle_G = 0, \quad \langle 1,k\rangle_G = \langle k,n\rangle_G = 1$$

and so

$$\langle 1,n\rangle_{G^{(k)}} = 1$$

Therefore, if $k \neq n-1$, $G^{(k)}$ is a connected graph with at least one looped vertex, contradicting the fact that $G \in \mathbf{CUP}_n$. We must conclude that $k = n-1$. Then $N_G(k) = \{1,n-1,n\}$ and so pressing k only affects four pairs of vertices:

$$\{(1,1),(1,n),(n-1,n-1),(n,n)\}.$$

Once again, since $G[[k-1]]$ is connected, $G^{(k)}[[k-1]]$ must be connected as well (although possibly without loops). However

$$\langle n, n\rangle_G = 0, \quad \langle 1, n\rangle_G = 0, \text{ and } \quad \langle 1, k\rangle_G = \langle k, n\rangle_G = 1$$

so

$$\langle 1, n\rangle_{G^{(k)}} = 1 \quad \text{and} \quad \langle n, n\rangle_{G^{(k)}} = 1.$$

It follows that $G^{(k)}$ is a connected graph with at least one looped vertex, namely n. This implies $G^{(k)}$ is a pressable graph, contradicting that $G \in \mathbf{CUP}_n$. Claim 2.24 is established. □

Claim 2.25. $u_{1,n-2} = 0$

Proof of Claim 2.25. Assume towards a contradiction that $u_{1,n-2} = 1$. By Property 1 of $U_{\hat{n}}$

$$w_{n-2} = \mathrm{wt}_{U_{\hat{n}}}(n-2) = n-2.$$

By Claim 1

$$\mathrm{wt}_{U_{\hat{n}}}(n-1) < n-1$$

and so by Property 4

$$\mathrm{wt}_{U_{\hat{n}}}(n-2) \equiv 0$$

and therefore $n \equiv 0$. We then have

$$\mathrm{wt}_{U_{\hat{1}}}(n-2) = w_{n-2} - u_{1,n-2} = n-3 \equiv 1$$

which contradicts Property 4 of $U_{\hat{1}}$ since $u_{2,n} = 0$. This establishes Claim 2.25. □

We may now assume $u_{1,n-2} = u_{1,n-1} = 0$ and proceed to show (II)-(IV). (II) follows from Property 2 of $U_{\hat{1}}$ since

$$w_n = \mathrm{wt}_{U_{\hat{1}}}(n) \geq \mathrm{wt}_{U_{\hat{1}}}(n-1) = w_{n-1}.$$

To verify (III), observe that if $w_{n-2} > 2$ then

$$\mathrm{wt}_{U_{\hat{1}}}(n-2) > 2 \quad \Rightarrow \quad w_n = \mathrm{wt}_{U_{\hat{1}}}(n) > \mathrm{wt}_{U_{\hat{1}}}(n-2) = w_{n-2}$$

by Property 3 on $U_{\hat{1}}$. (IV) is established by observing that

$$u_{1,n-1} = 0 \quad \Rightarrow \quad \mathrm{wt}_{U_{\hat{n}}}(n-1) < n-1 \quad \Rightarrow \quad \mathrm{wt}_{U_{\hat{n}}}(i) \equiv 0 \text{ for all } 2 < i < n$$

by Property 4 of $U_{\hat{n}}$ and so

$$w_i = \mathrm{wt}_{U_{\hat{n}}}(i) \equiv 0 \text{ for all } 2 < i < n.$$

Third Case: $u_{1,n} = 0$, $u_{2,n} = 1$.

By Property 1 of $U_{\hat{1}}$ we have that $u_{i,n} = 1$ for all $2 \leq i \leq n$. It follows that $w_n = n-1$ and so (I) and (II) hold. Furthermore, since $w_{n-2} \leq n-2$ then (III) holds. To verify (IV), we need to show that $\mathcal{L}(G) = \{1\}$.

Claim 2.26. $\mathcal{L}(G) \neq \{1, n\}$.

Proof of Claim 2.26. Assume, by way of contradiction, that $\mathcal{L}(G) = \{1, n\}$. Since $G \in \mathbf{CUP}_n$ and $n \neq 1$ then $G^{(n)}$ must contain a non-trivial loopless component C. Choose and fix $p \in V(C)$. Since C is non-trivial we may assume $p \neq 1$. Since $H \in \mathbf{CUP}_{n-1}$ there must exist a path from p to a looped vertex in H. Since $\mathcal{L}(H) = \{1\}$ this looped vertex must be 1. Choose such a path P,

$$P = v_0 \dots v_\ell$$

where $p = v_0$ and $v_\ell = 1$. Observe that

$$\langle 1, n \rangle_G = 0$$

so 1 is a looped vertex in $G^{(n)}$ as well. Then some interior edge of P must be removed upon pressing n as otherwise p would have a path to a looped vertex in $G^{(n)}$. Let

$$j = \min_{0 \leq i < \ell} \{i \mid \langle v_i, n \rangle_G = 1\}$$

then

$$P' = v_0 P v_j$$

is a path from p to a looped vertex in $G^{(k)}$, a contradiction. This establishes Claim 2.26. □

Claim 2.27. If $n \equiv 0$ then $\mathcal{L}(G) = \{1\}$.

Proof of Claim 2.27. Let $n \equiv 0$. Assume, by way of contradiction, that $\mathcal{L}(G) \neq \{1\}$ and let

$$k = \min_{3 \leq i \leq n-1} \{i \mid w_i \equiv 1\}.$$

Choose a non-trivial loopless component C of $G^{(k)}$ and a vertex $p \in V(C) \setminus \{1\}$. Recall that $u_{1,n} = 0$ and $u_{2,n} = 1$. Then

$$\langle n, n \rangle_G = \bigoplus_{i=1}^{n} u_{i,n} \equiv n - 1 \equiv 1 \quad \text{and} \quad \langle k, n \rangle_G = \bigoplus_{i=1}^{n} u_{i,k} \equiv k - 1 \equiv 0$$

implies

$$\langle n, n \rangle_{G^{(k)}} = 1$$

so $n \neq p$. For any $j \in [k+1, n-1] \setminus \mathcal{L}(G)$, since $H \in \mathbf{CUP}_{n-1}$ and $w_k \equiv 1$, Property 3 implies that $\langle k, j \rangle_H = 1$, whence

$$\langle j, j \rangle_G = 0 \text{ and } \langle k, j \rangle_G = 1 \quad \text{implies} \quad \langle j, j \rangle_{G^{(k)}} = 1$$

so j is looped in $G^{(k)}$ and therefore $p \notin [k+1, n-1] \setminus \mathcal{L}(G)$. If $j \in [k+1, n-1] \cap \mathcal{L}(G)$ then $j \in [k+2, n-1]$ and

$$\langle j, j-1 \rangle_G = 0 \text{ and } \langle k, j-1 \rangle_G = \langle k, j \rangle_G = 1 \quad \text{implies} \quad \langle j, j-1 \rangle_{G^{(k)}} = 1$$

so $p \notin [k+1, n-1] \cap \mathcal{L}(G)$. Therefore

$$p \in [2, k-1].$$

Since $G[[k-1]] \in \mathbf{CUP}_{k-1}$ then there exists a path P in $G[[k-1]]$ connecting p to a looped vertex; since $\mathcal{L}(G[[k-1]]) = \mathcal{L}(G) \cap [k-1] = \{1\}$ then the looped vertex must 1.

$$P = v_0 \dots v_\ell$$

where $v_0 = p$ and $v_\ell = 1$. Note that

$$N_G(k) \cap V(P) = \{1\}$$

because, if $i = \min\{j : v_j \in N_G(k)\}$ is not 1, then v_i is looped in $G^{(k)}$ and in the same component (namely, C) with p, a contradiction. Then all the interior edges of P are unaffected by pressing k (although the loop on 1 is removed). If $2 \in V(P)$ then

$$P' = v_0 P 2n$$

is a path from p to n in $G^{(k)}$. If $2 \notin V(P)$ then

$$P' = v_0 P v_\ell 2n$$

is a path from p to n in $G^{(k)}$. Since n is looped in $G^{(k)}$ this contradicts that C is a non-trivial loopless component. Claim 2.27 is established. □

By Claims 2.26 and 2.27 we have only one case left to consider. Let $n \equiv 1$ and assume, by way of contradiction, that $\mathcal{L}(G) \neq \{1\}$. Let

$$k = \min_{3 \leq i \leq n-1} \{i \mid w_i \equiv 1\}.$$

Choose a non-trivial component C of $G^{(k)}$ and a vertex $p \in V(C) \setminus \{1\}$. Observe that $n-1 \equiv 0$ so $\mathrm{wt}_G(n-1) \equiv 0$ by Property 4 of H. Then, by Property 4 of H

$$\langle n-1, n-1 \rangle_G \equiv n-1 \equiv 0 \text{ and } \langle k, n-1 \rangle_G \equiv k \equiv 1$$

which implies

$$\langle n-1, n-1 \rangle_{G^{(k)}} = 1$$

so $n-1 \neq p$. Similarly,

$$\langle n-1, n\rangle_G = 1 \cdot 0 \oplus \bigoplus_{i=2}^{n-1} 1 \cdot 1 \equiv n-2 \equiv 1 \text{ and } \langle k, n\rangle_G = 1 \cdot 0 \oplus \bigoplus_{i=2}^{k} 1 \cdot 1 \equiv k-1 \equiv 0$$

implies

$$\langle n-1, n\rangle_{G^{(k)}} = 1$$

so $n \neq p$ as it is connected to a looped vertex. If $j \in [k+1, n-2] \setminus \mathcal{L}(G)$ then

$$\langle j, j\rangle_G = 0 \text{ and } \langle k, j\rangle_G = 1 \quad \text{implies} \quad \langle j, j\rangle_{G^{(k)}} = 1$$

so $p \notin [k+1, n-2] \setminus \mathcal{L}(G)$. If $j \in [k+1, n-2] \cap \mathcal{L}(G)$ then $j \in [k+2, n-2]$ and

$$\langle j, j-1\rangle_G = 0 \text{ and } \langle k, j-1\rangle_G = \langle k, j\rangle_G = 1 \quad \text{implies} \quad \langle j, j-1\rangle_{G^{(k)}} = 1$$

so $p \notin [k+1, n-2] \cap \mathcal{L}(G)$. It follows that

$$p \in [2, k-1].$$

Since $G[[k-1]] \in \mathbf{CUP}_{k-1}$ has only 1 as a looped vertex then $G[[k-1]]$ contains a path

$$P = v_0 \dots v_\ell$$

where $v_0 = p$ and $v_\ell = 1$. Since $N_G(k) \cap [k-1] = \{1\}$ none of the interior edges are removed upon pressing k. If $2 \in V(P)$ we have

$$P' = v_0 P 2n(n-1)$$

is a path to a looped vertex in $G^{(k)}$ and if $2 \notin V(P)$ then

$$P' = v_0 P v_\ell 2n(n-1)$$

is a path to a looped vertex in $G^{(k)}$. This contradicts that C is a loopless component in $G^{(k)}$. We conclude that $\mathcal{L}(G) = \{1\}$ and therefore establish (IV).

Fourth Case: $u_{1,n} = 1$, $u_{2,n} = 0$.

We show that this case cannot occur. Assume it does to reach a contradiction. Since $u_{2,n} = 0$ then by Property 4 only the initial column of $U_{\hat{1}}$ has odd weight. It follows that

$$w_n = \mathrm{wt}_{U_{\hat{1}}}(n) + 1 \equiv 1.$$

Let $k = \min_{2 \leq i \leq n} \{i \mid w_i \equiv 1\}$.

Claim 2.28. $k \neq n$

Proof of Claim 2.28. Assume, by way of contradiction, that $k = n$. Then $\mathcal{L}(G) = \{1, n\}$. Since $G \in \mathbf{CUP}_n$ we may conclude that $G^{(n)}$ contains a non-trivial loopless component C. Choose and fix $p \in V(C) \setminus \{1\}$. Since $H = G[[n-1]] \in \mathbf{CUP}_{n-1}$ there exists a path P in H that connects p to a looped vertex, namely 1. Let

$$P = v_0 \dots v_\ell$$

where $v_0 = p$ and $v_\ell = 1$. Observe that

$$\langle v_\ell, n \rangle_G = \langle 1, n \rangle_G = 1$$

and let

$$m = \min_{0 \leq i \leq \ell} \{i \mid v_i \in N_G(n)\}.$$

If $m < \ell$ then

$$P' = v_0 P v_m$$

is a path to a looped vertex in $G^{(n)}$, contrary to assumption. Assume $m = \ell$ and let $q = v_{\ell-1}$.

$$1 = \langle v_\ell, v_{\ell-1} \rangle_G = \langle 1, q \rangle_G$$

so $u_{1,q} = 1$. By Property 1 of $U_{\hat{n}}$, $u_{i,q} = 1$ for all $i \in [q]$ and so $w_q = q$. Let

$$r = \min_{2 \leq i \leq n} \{i \mid u_{i,n} = 1\}.$$

Since $\langle q, n\rangle_G = \langle v_{\ell-1}, n\rangle_G = 0$ then

$$0 = \bigoplus_{i=1}^{n}(u_{i,q}u_{i,n}) = (1\cdot 1)\oplus\bigoplus_{i=2}^{r-1}(u_{i,q}\cdot 0)\oplus\bigoplus_{i=r}^{q}(1\cdot 1)\oplus\bigoplus_{i=q+1}^{n}(0\cdot 1)$$

and therefore

$$\langle q, n\rangle_G = 0 = 1\oplus\bigoplus_{i=r}^{q}1$$

which implies $r \leq q$. Furthermore $q \neq n-1$ since otherwise we would have

$$w_{n-1} = w_q = q = n-1$$

and

$$n-1 \geq w_n = u_{1,n} + \mathrm{wt}_{U_{\hat{1}}}(n) = 1 + \mathrm{wt}_{U_{\hat{1}}}(n) \geq 1 + \mathrm{wt}_{U_{\hat{1}}}(n-1) = 1 + (n-2)$$

which would imply

$$w_n = n-1 = w_q \equiv 0$$

contradicting that n is a looped vertex. Thus $q \neq n-1$ and so $q+1 < n$ which gives us that $w_{q+1} \equiv 0$ by Property 4 of $U_{\hat{1}}$ and the fact that $u_{2,n} = 0$. However,

$$q = w_q = \mathrm{wt}_{U_{\hat{n}}}(q) \leq \mathrm{wt}_{U_{\hat{n}}}(q+1) = w_{q+1} \leq q+1$$

implies $w_{q+1} = q$ since $q+1 \equiv 1$ and therefore $u_{1,q+1} = 0$ by Property 1 of $U_{\hat{n}}$. Then

$$\langle q, q+1\rangle_G = \bigoplus_{i=1}^{n}(u_{i,q}\cdot u_{i,q+1}) = (1\cdot 0)\oplus\bigoplus_{i=2}^{q}(1\cdot 1)\oplus(0\cdot 1)\oplus\bigoplus_{i=q+2}^{n}(0\cdot 0) = 1$$

and

$$\langle q+1, n\rangle_G = (0\cdot u_{1,n})\oplus\bigoplus_{i=2}^{r-1}(u_{i,q+1}\cdot 0)\oplus\bigoplus_{i=r}^{q+1}(1\cdot 1)\oplus\bigoplus_{i=q+2}^{n}(0\cdot u_{i,n})$$

and so

$$\langle q+1, n\rangle_G = \bigoplus_{i=r}^{q+1}1 = 1\oplus\bigoplus_{i=r}^{q}1 = \langle q, n\rangle_G = 0$$

Observe that

$$\langle 2, q+1\rangle_G = \bigoplus_{i=1}^{2}u_{i,q+1} = 1$$

so

$$P' = q(q+1)2$$

is a path in G whose $q(q+1)$ and $(q+1)2$ edges are unaffected by pressing n (since $(q+1) \notin N_G(n)$) and since

$$\langle 2,2\rangle_G = \bigoplus_{i=1}^{2} u_{i,2} = 0 \quad \text{and} \quad \langle 2,n\rangle_G = \bigoplus_{i=1}^{2} u_{i,n} = 1$$

$$P^* = pPqP'2$$

is a path from p to a looped vertex in $G^{(n)}$, contrary to assumption that p is contained in a loopless component of $G^{(n)}$. This contradicts that $G \in \mathbf{CUP}_n$ and establishes Claim 2.28. □

We proceed under the assumption that

$$k = \min_{2\le i\le n}\{i \mid w_i \equiv 1\} < n,$$

once again in search of a contradiction. Observe that we need not consider the case where $n \le 3$ since the super-diagonal entries must be all 1. Furthermore, if $n = 4$ we have only two matrices to consider, both of which have an additional successful pressing sequence given by $(4,3,2,1)$:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Assume $n \ge 5$. By Property 4 of $U_{\hat{n}}$ we have $u_{1,n-1} = 1$ and so by Property 1 and 2 of $U_{\hat{n}}$

$$n-1 \ge w_n = \mathrm{wt}_{U_{\hat{1}}}(n) + 1 \ge \mathrm{wt}_{U_{\hat{1}}}(n-1) + 1 = (n-2) + 1 = n-1$$

which gives us $w_n = n-1$. Observing that $\mathrm{wt}_{U_{\hat{1}}}(n) = n-2$ and $u_{2,n} = 0$ we conclude from Property 4 of $U_{\hat{1}}$ that $n-2 \equiv 0$, so n is looped in G. Since $G \in \mathbf{CUP}_n$ then

$G_{(n)}$ must contain a non-trivial loopless component, say C. Choose and fix a vertex $p \in V(C) \setminus \{1\}$. Observe that $G[[p]] \in \mathbf{CUP}_p$ by Corollary 2.18. If $p \in \mathcal{L}(G)$ then pressing n must remove its loop so

$$\langle 2, p\rangle_G = 0 \quad \text{and } \langle 2, n\rangle_G = \langle p, n\rangle_G = 1$$

which implies that $\langle 2, p\rangle_{G_{(n)}} = 1$ and $2 \in \mathcal{L}(G_{(n)})$, contradicting that C is loopless. It follows that $p \notin \mathcal{L}(G)$ and therefore, in $G[[p]]$, we have a path P from p to a looped vertex b:

$$P = v_0 \dots v_\ell$$

where $v_0 = p$, $v_\ell = b$, and v_i is loopless in $G[[p]]$ for $0 < i \leq \ell$. Observe that $N_G(n) \cap \{v_1, v_2, \dots, v_\ell\} = \emptyset$ since otherwise we would have a looped vertex in C. It follows that the interior edges of P are unaffected by pressing n in G and therefore it must be the case that pressing n in G removes the loop from $b = v_\ell$. By Property 4 on $U_{\hat{n}}$ looped vertices in H have full weight. Observe that

$$\bigoplus_{i=1}^{n}(u_{i,b}u_{i,n}) = (1 \cdot 1) \oplus (u_{2,b} \cdot 0) \oplus \bigoplus_{i=3}^{n}(u_{i,b} \cdot 1) \equiv \begin{cases} 1, & \text{if } b = 1 \\ 1 + (b-2), & \text{if } b \neq 1 \end{cases}$$

Since $1 = \langle b, n\rangle_G = \bigoplus_{i=1}^{n}(u_{i,b}u_{i,n})$ this implies that $b = 1$. (Otherwise, b is even but looped in G, contradicting Property 4.) Observe that $w_3 = 3$ would imply $w_4 = \mathrm{wt}_{U_{\hat{n}}}(4) = 4$ by Property 4 of $U_{\hat{n}}$, and then $\mathrm{wt}_{U_{\hat{1}}}(4) \equiv 1$ but $u_{2,n} = 0$, contradicting Property 4 of $U_{\hat{1}}$. Then

$$w_3 = \mathrm{wt}_{U_{\hat{n}}}(3) = 2$$

by Property 2 of $U_{\hat{n}}$ and so

$$\langle 3, 3\rangle_G = 0 \quad \text{and} \quad \langle 3, n\rangle_G = 1 \quad \text{implies} \quad \langle 3, 3\rangle_{G^{(n)}} = 1$$

and so $3 \notin V(P)$. Furthermore

$$\langle 1, 3\rangle_G = 0 \quad \text{and} \quad \langle 1, n\rangle_G = \langle 3, n\rangle_G = 1 \quad \text{implies} \quad \langle 1, 3\rangle_{G^{(n)}} = 1.$$

Therefore

$$P' = v_0 P v_\ell 3$$

is a path to a looped vertex in $G^{(n)}$. This implies $G \notin \mathbf{CUP}_n$ contrary to assumption. Therefore Case 4 cannot occur. □

2.4 Recognition and Enumeration

A straightforward and very slow way to check if a simple pseudo-graph on n vertices is uniquely pressable is to check the pressability of each one of its $n!$ orderings. Here we offer a substantially faster algorithm.

Algorithm 1: Find a Pressing Order

1: **input:** Adjacency matrix A with entries $a_{i,j}$ for $i, j \in [n]$
2: **output:** Re-indexed matrix $P^T AP$
3: $M \leftarrow A$
4: $P \leftarrow 0_{n\times n}$
5: $t \leftarrow 1$
6: **while** $t \leq n$ **do**
7: $\quad maxDegree \leftarrow 0$
8: $\quad indexMaxDegree \leftarrow 0$
9: $\quad i \leftarrow 1$
10: $\quad$ **while** $i \leq n$ **do**
11: $\quad\quad deg_i \leftarrow 0$
12: $\quad\quad$ **if** $m_{i,i} = 1$ **then**
13: $\quad\quad\quad deg_i \leftarrow \sum_{j=1}^{n} m_{i,j}$
14: $\quad\quad\quad$ **if** $deg_i > maxDegree$ **then**
15: $\quad\quad\quad\quad maxDegree \leftarrow d_i$
16: $\quad\quad\quad\quad indexMaxDegree \leftarrow i$
17: $\quad\quad i \leftarrow i + 1$

if $indexMaxDegree = 0$ **then**
  if $\sum_{\ell=1}^{n} \sum_{j=1}^{n} m_{\ell,j} > 0$ **then**
    return False {Not a Pressable Graph}
  else
    $k \leftarrow indexMaxDegree$
    $p_{t,k} \leftarrow 1$
    $t \leftarrow t + 1$
    for $\ell \in [n]$ **do**
      for $j \in [n] \setminus \{k\}$ **do**
        $m_{\ell,j} \leftarrow m_{\ell,j} \oplus (m_{k,j} \cdot m_{\ell,k})$
    for $j \in [n]$ **do**
      $m_{k,j} \leftarrow 0$
return $P^T AP$

Algorithm 2: Construct instructional Cholesky root

input: Adjacency matrix A with entries $a_{i,j}$ for $i, j \in [n]$
output: Instructional Cholesky matrix U
$U \leftarrow 0_{n \times n}$
$k \leftarrow 1$
while $k \leq n$ **do**
  if $a_{k,k} = 1$ **then**
    for $j \in [n]$ **do**
      $u_{k,j} \leftarrow a_{k,j}$
    for $i \in [k+1, n]$ **do**
      for $j \in [n]$ **do**
        $a_{i,j} \leftarrow a_{i,j} \oplus (a_{k,j} \cdot a_{i,k})$

12: $k \leftarrow k+1$

13: **else**

14: $k \leftarrow n+1$

15: **return** U

Algorithm 3: Does this instructional Cholesky correspond to a uniquely pressable OSP?

1: **input:** Instructional Cholesky matrix U with entries $u_{i,j}$ for $i,j \in [n]$

2: **output:** True or False

3: $j \leftarrow 1$

4: **while** $j \leq n$ **do**

5: $i \leftarrow 1$

6: **while** $i \leq j$ **do**

7: **if** $u_{i,j} = 0$ **then**

8: $i \leftarrow i+1$

9: **else if** $\sum_{\ell=i}^{j} u_{\ell,j} < j-i+1$ **then**

10: **return** False

11: $j \leftarrow j+1$

12: $j \leftarrow 1$

13: **while** $j < n$ **do**

14: **if** $\sum_{\ell=1}^{j} u_{\ell,j} > \sum_{\ell=1}^{j+1} u_{\ell,j+1}$ **then**

15: **return** False

16: **else if** $j \leq n-2$ and $\sum_{\ell=1}^{j} u_{\ell,j} > 2$ and $\sum_{\ell=1}^{j+2} u_{\ell,j+2} = \sum_{\ell=1}^{j} u_{\ell,j}$ **then**

17: **return** False

18: **else if** $\sum_{\ell=1}^{j+1} u_{\ell,j+1} \equiv 1$ and $\sum_{\ell=1}^{j+1} u_{\ell,j+1} \neq j+1$ **then**

19: **return** False

20: **else**

21: $j \leftarrow j + 1$

22: **return** True

Corollary 2.29. *The unique pressability of G can be decided in time $O(n^3)$.*

Proof. Let $G = (V, E)$ be a simple pseudo-graph on n vertices. Let $G' = ([n], E')$ be the result of relabeling of $V(G)$ so that G' is order-pressable. If G is uniquely pressable then the instructional Cholesky root of G' is in $\mathcal{M}_n$. Observe that for each $k \in \mathcal{L}(G') \setminus \{1\}$,

$$N_{G'}(k) = \{1\} \cup [k, n]$$

since by Property 4 each $\ell \in [k, n]$ has full weight and therefore $\langle k, \ell \rangle_{G'} \equiv k \equiv 1$. However, for each $k \in \mathcal{L}(G') \setminus \{1\}$ and $\ell \in [k, n]$, $\langle 1, \ell \rangle_{G'} = 1$ by Property 4, and

$$\langle 1, 2 \rangle_{G'} = 1 \quad \text{and} \quad \langle 2, k \rangle_{G'} \equiv 2 \equiv 0.$$

Thus, $N_{G'}(1) \supseteq \{2\} \cup N_{G'}(k) \supsetneq N_{G'}(k)$.

Therefore 1 is the unique looped vertex of largest degree. It follows that to find a (potentially unique) pressing order it suffices to iterate the process of finding the looped vertex of largest degree and pressing it. Index the vertices of graph G arbitrarily and define $A = (a_{i,j}) \in \mathbb{F}_2^{n \times n}$, the adjacency matrix of the graph. We proceed to describe three algorithms which have been included in the appendix.

Algorithm 1 finds a successful pressing sequence for G (given that one exists) by finding the looped vertex of largest degree and pressing it; this has running time $O(n^3)$, as it amounts to performing in-place Gaussian elimination on an $n \times n$ matrix. Algorithm 2 computes the instructional Cholesky root of an ordered adjacency matrix and once again is done by performing Gaussian elimination. Finally, Algorithm 3 checks if an upper-triangular matrix has the properties of $\mathcal{M}_n$ which is done by computing no more than n partial sums for each of n columns and comparing them sequentially. Algorithm 3 also has running time $O(n^3)$.

□

Corollary 2.30. *Let $n > 0$. Let $G = ([n], E)$ and let $H = ([n+1], E(H))$ be the result of adding a vertex "$n+1$" adjacent to each looped vertex in G, with a loop at $n+1$ if and only if n is even. If $G \in \mathbf{CUP}_n$, then $H \in \mathbf{CUP}_{n+1}$.*

Proof. Suppose $G \in \mathbf{CUP}_n$ with adjacency matrix A and instructional Cholesky root $U = (u_{i,j})$. Observe that $H = ([n+1], E(H))$ where

$$E(H) = \begin{cases} E(G) \cup \{(i, n+1) \mid i \in \mathcal{L}(G)\} \cup \{(n+1, n+1)\}, & \text{if } n \equiv 0 \\ E(G) \cup \{(i, n+1) \mid i \in \mathcal{L}(G)\}, & \text{if } n \equiv 1 \end{cases}$$

Let $V = \left[\begin{array}{ccc|c} & & & 1 \\ & U & & \vdots \\ & & & 1 \\ \hline 0 & \cdots & 0 & 1 \end{array}\right]$ and observe that

$$V^T \cdot V = \left[\begin{array}{ccc|c} & & & b_{1,n+1} \\ & A & & \vdots \\ & & & b_{n,n+1} \\ \hline b_{n+1,1} & \cdots & b_{n+1,n} & b_{n+1,n+1} \end{array}\right]$$

where $b_{n+1,i} = 1$ if and only if $\mathrm{wt}_V(i) \equiv 1$ if and only if $i \in \mathcal{L}(G)$ or $i = n+1 \equiv 1$. It follows that $V^T V$ is a Cholesky factorization for the adjacency matrix of H. Since $G \in \mathbf{CUP}_n$ then U has 1's along the diagonal and so U and V are full rank matrices. Since Cholesky factorizations are unique for full-rank matrices [9] then V must be the instructional Cholesky root of H. V inherits the properties of $\mathcal{M}_n$ in its first n columns from U. Furthermore the last column of V has full weight $n+1$, so $V \in \mathcal{M}_{n+1}$, and therefore $H \in \mathbf{CUP}_{n+1}$. □

Corollary 2.31. *Let $n > 0$. Let $G = ([2, n+1], E)$ and $H = ([n+1], E(H))$ be the result of removing all the edges, including loops, from $G[\mathcal{L}(G)]$, adding a looped vertex " 1 " adjacent to all of $\mathcal{L}(G)$. If $G \in \mathbf{CUP}_n$ then $H \in \mathbf{CUP}_{n+1}$.*

Proof. Let $G \in \mathbf{CUP}_n$, recall that by Property 4 (of its instructional Cholesky root) the looped vertices in G form a clique. The only looped vertex in H is 1 so if H admits a successful pressing sequence it must begin with 1. However $H^{(1)} = G$ since pressing and deleting 1 creates an edge between any two vertices in $\mathcal{L}(G)$. It follows that H has exactly one successful pressing sequence: $\mathbf{n+1}$. $H \in \mathbf{CUP}_{n+1}$. □

Notation 2.32. $\mathbf{CUP}_{[n]}$ is the set of connected, uniquely pressable ordered $<_{\mathbb{N}}$ simple pseudo-graphs on vertex set $[n]$.

Corollary 2.33. *Up to isomorphism, the number of connected, uniquely pressable simple pseudo-graphs on $n > 1$ vertices is*

$$\left|\mathbf{CUP}_{[n]}\right| = \begin{cases} 3^{(n-2)/2}, & n \text{ is even} \\ 2 \cdot 3^{(n-3)/2}, & n \text{ is odd.} \end{cases}$$

Proof. For $n = 2$, the result holds since

$$\mathbf{CUP}_{[2]} = \{([2], \{(1,1),(1,2)\})\}.$$

We proceed by induction. Let $n \geq 2$ be even and assume $\left|\mathbf{CUP}_{[n]}\right| = 3^{(n-2)/2}$. Choose and fix $G = ([n], E) \in \mathbf{CUP}_{[n]}$ with adjacency matrix A and instructional Cholesky root $U = (u_{i,j})$. Let $G' = ([2, n+1], E')$ be a re-indexing of G given by $i \mapsto i+1$ for all $i \in [n]$. Let $H_1 = ([n+1], E_1)$, $H_2 = ([n+1], E_2)$ where

$$E_1 = E \cup \{(i, n+1) \mid i \in \mathcal{L}(G)\} \cup \{(n+1, n+1)\}$$

and

$$E_2 = E' \triangle ((\mathcal{L}(G') \cup \{1\}) \times (\mathcal{L}(G') \cup \{1\})).$$

By Corollaries 2.30 and 2.31, $H_1, H_2 \in \mathbf{CUP}_{[n+1]}$. H_1 has at least two looped vertices (1 and $n+1$) and H_2 has only one looped vertex (1), so $H_1 \neq H_2$. Furthermore, the instructional Cholesky roots of H_1 and H_2 include U as a principal submatrix on consecutive rows and columns so it is not possible that we would have gotten H_1 or H_2 by applying Corollaries 2.30 or 2.31 to other graphs in $\mathbf{CUP}_{[n]}$. It follows that

$$\left|\mathbf{CUP}_{[n+1]}\right| \geq 2 \cdot \left|\mathbf{CUP}_{[n]}\right| = 2 \cdot 3^{(n-3)/2}.$$

Consider now any $H \in \mathbf{CUP}_{[n+1]}$ with instructional Cholesky root $V = (v_{i,j})$. If $v_{1,n+1} = 0$ then let $G' = H^{(1)}$ and let G be the re-indexing of $V(G)$ given by $i \mapsto i-1$ for all $i \in [2, n+1]$. By Lemma 2.15, $G \in \mathbf{CUP}_{[n]}$ and therefore H can be constructed from G using Corollary 2.31. If $v_{1,n+1} = 1$ then $v_{i,n+1} = 1$ for all $i \in [n+1]$ because of Property 4 by appeal to Theorem 2.23. Let $G = H - \{n+1\}$. By Lemma 2.17, $G \in \mathbf{CUP}_{[n]}$ and hence H can be obtained by applying Corollary 2.30 to G. It follows that

$$\left|\mathbf{CUP}_{[n+1]}\right| \leq 2 \cdot \left|\mathbf{CUP}_{[n]}\right| = 2 \cdot 3^{(n-3)/2}.$$

We count $\left|\mathbf{CUP}_{[n+2]}\right|$ in a similar, though slightly more complicated, manner. Given $G \in \mathbf{CUP}_{[n]}$, let H_1 be result of two successive applications of Corollary 2.30. Let H_2 be the result of applying Corollary 2.30 followed by Corollary 2.31, let H_3 be the result of applying Corollary 2.31 followed by Corollary 2.30, and let H_4 be the result of applying Corollary 2.31 twice successively. We first show that $H_2 = H_3$ and then argue that H_1, H_2, H_4 are pairwise distinct. Let V_2 and V_3 be the instructional Cholesky roots of H_2 and H_3, respectively. Then

$$V_2 = \left[\begin{array}{c|ccc|c} 1 & b_{1,2} & \cdots & b_{1,n} & 1 \\ \hline 0 & & & & 1 \\ \vdots & & U & & \vdots \\ 0 & & & & 1 \\ \hline 0 & 0 & \cdots & 0 & 1 \end{array}\right] = V_3$$

where $b_{1,i} = 1$ whenever it is positioned above a column of odd weight in U. Therefore $H_2 = H_3$. Observe that H_1 has at least two looped vertices, 1 and $n+1$, whereas H_2 and H_4 have only one looped vertex "1". Since n is even, vertex n in G is loopless by Property 4. Then the instructional Cholesky root of H_4 has the form

$$V_4 = \left[\begin{array}{c|c|ccc} 1 & 1 & b_{1,3} & \cdots & b_{1,n+2} \\ \hline 0 & 1 & b_{2,3} & \cdots & b_{2,n+2} \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & U & \\ 0 & 0 & & & \end{array}\right]$$

where $b_{1,n+2} = b_{2,n+2} = 0$ and so $H_4 \neq H_2$. It follows that

$$\left|\mathbf{CUP}_{[n+2]}\right| \geq 3 \cdot \left|\mathbf{CUP}_{[n]}\right| = 3^{((n+2)-3)/2}.$$

Choose and fix $H \in \mathbf{CUP}_{[n+2]}$ with instructional Cholesky root $V = (v_{i,j})$. Let $G_1 = (H - \{n+2\}) - \{n+1\}$, $G_2' = H^{(1)} - \{n+2\}$, and $G_3' = H^{(1,2)}$. Let G_2 and G_3 be (order-preserving) re-indexings of G_2' and G_3' so that $V(G_2) = V(G_3) = [n]$. By (repeated) applications of Lemmas 2.15 and 2.17; $G_1, G_2, G_3 \in \mathbf{CUP}_{[n]}$. Let $\alpha = v_{1,n+1} + v_{1,n+2}$. If $\alpha = 2$ then $v_{1,n+1} = v_{1,n+2} = 1$ and by Property 1, $v_{i,n+1} = 1$ for all $i \in [n+1]$ and $v_{i,n+2} = 1$ for all $i \in [n+2]$. Then H can be constructed from two applications of Corollary 2.30 to G_1. If $\alpha = 1$ then $v_{1,n+1} = 0$ and $v_{1,n+2} = 1$ by Property 4. Furthermore, Property 4 implies that 1 is the only looped vertex, since $n+2$ is even and $v_{1,n+1} = 0$. Observing that $n+2$ must be a full weight vertex, we conclude that H can be constructed from G_2 by application of Corollaries 2.30 and 2.31 (in either order). Finally if $\alpha = 0$ then $v_{1,n+2} = 0$ and by Property 4, and since n is even, $v_{2,n+2} = 0$. Furthermore by Property 4 it follows that H and $H^{(1)}$ have each only one looped vertex. Then H can be constructed from G_3 by two applications of Corollary 2.31. It follows that

$$\left|\mathbf{CUP}_{[n+2]}\right| \leq 3 \cdot \left|\mathbf{CUP}_{[n]}\right| = 3^{((n+2)-3)/2}.$$

Therefore

$$\left|\mathbf{CUP}_{[n]}\right| = \begin{cases} 3^{(n-2)/2}, & \text{if } n \text{ is even} \\ 2 \cdot 3^{(n-3)/2}, & \text{if } n \text{ is odd} \end{cases}.$$

□

Corollary 2.34. *The number of uniquely pressable simple pseudo-graphs on $n > 1$ vertices up to isomorphism is*

$$T_n = \begin{cases} \left(5 \cdot 3^{(n-2)/2} + 1\right)/2, & \textit{if } n \textit{ is even} \\ \left(3^{(n+1)/2} + 1\right)/2, & \textit{if } n \textit{ is odd} \end{cases}$$

Proof. There are three non-isomorphic uniquely pressable simple pseudo-graphs on 2 vertices: the edgeless (loopless) graph, the disconnected graph containing one looped vertex and one unlooped vertex, and the connected graph containing one looped vertex and one unlooped vertex. We proceed by induction on n. Observe that for every $k \leq n$ and for every $H \in \mathbf{CUP}_k$, we can create a (distinct) uniquely pressable graph G on n vertices by adding $n - k$ isolated vertices to H. Similarly, if G is a uniquely pressable graph, then it is either the edgeless (loopless) graph or it contains exactly one non-trivial component which must be a $\mathbf{CUP}_k$ graph for some $k \leq n$. Hence

$$T_n = \sum_{k=0}^{n} \left|\mathbf{CUP}_{[k]}\right| = T_{n-1} + \left|\mathbf{CUP}_{[n]}\right|.$$

The result follows by observing that

$$\frac{5 \cdot 3^{((n-1)-2)/2} + 1}{2} + 2 \cdot 3^{(n-3)/2} = \frac{5 \cdot 3^{(n-3)/2} + 1 + 4 \cdot 3^{(n-3)/2}}{2} = \frac{3^{(n+1)/2} + 1}{2}$$

and

$$\frac{3^{((n-1)+1)/2} + 1}{2} + 3^{(n-2)/2} = \frac{3 \cdot 3^{(n-2)/2} + 1 + 2 \cdot 3^{(n-2)/2}}{2} = \frac{5 \cdot 3^{(n-2)/2} + 1}{2}$$

□

Chapter 3

Autonomous Posets

3.1 Introduction

A simple pseudo-graph is a graph that admits loops but not multiple edges (sometimes known as a "loopy graph"). Given a simple pseudo-graph G, denote by $V(G)$ the vertex set of G; $E(G) \subseteq V(G) \times V(G)$, symmetric as a relation, its edge set. Let $N(v) = N_G(v) = \{w \in V(G) : vw \in E(G)\}$ the *neighborhood* of v in $V(G)$. Observe that $v \in N(v)$ iff v is a looped vertex. For $S \subset V$, we denote by $G[S]$ the vertex-induced subgraph on S.

Definition 3.1. Consider a simple pseudo-graph G with a looped vertex $v \in V(G)$. "Pressing v" is the operation of transforming G into G', a new simple pseudo-graph in which $G[N(v)]$ is complemented. That is,

$$V(G') = V(G), \quad E(G') = E(G) \triangle (N(v) \times N(v))$$

We denote by $G_{(v)}$ the simple pseudo-graph resulting from pressing vertex v in $V(G)$ and we abbreviate $G_{(v_1)(v_2)\cdots(v_k)}$ to $G_{(v_1,v_2,\ldots,v_k)}$. For $k \geq 1$ we abbreviate $(1, 2, \ldots, k)$ as $\boldsymbol{k}$ so that when $V(G) = [n]$ for some $n \geq k$ then we may simplify $G_{(1,2,\ldots,k)}$ to $G_{\boldsymbol{k}}$. $G_{\mathbf{0}}$ and $G_{()}$ are interpreted to mean G. To aid with inductive arguments, we let $G^{(v)} = G_{(v)} - v$: the result of pressing v in G (which leaves it isolated, loopless, and thenceforth unpressable) and then removing the pressed vertex.

Given a simple pseudo-graph G, $(v_1, v_2, \ldots, v_j)$ is said to be a *successful pressing sequence* for G whenever the following conditions are met:

- $\{v_1, v_2, \ldots, v_k\} \subseteq V(G)$,
- v_i is looped in $G_{(v_1,v_2,\ldots,v_{i-1})}$ for all $1 \leq i \leq k$,
- $G_{(v_1,v_2,\ldots,v_k)} = (V(G), \emptyset)$

In other words, looped vertices are pressed one at a time, with "success" meaning that the end result (when no looped vertices are left) is an empty graph. This topic originated in computational phylogenetics, where Hannenhalli and Pevzner showed that certain simple pseudo-graphs correspond to pairs of genomes and that the reversal edit distance between these genomes is the minimum length of a successful pressing sequence of said graph [20]. In phylogenetics, the simple pseudo-graph corresponds to a pair of homologous genomes and its successful pressing sequences corresponds to a most plausible (i.e., parsimonious) evolutionary history between the genomes (see [12, 27]). In the present work we look at the set of simple pseudo-graphs whose pressing sequences correspond to the linear extensions of a single poset. Since linear extensions can be efficiently sampled asymptotically uniformly, this shows that pressing sequences, and hence the evolutionary histories of the pairs of genomes giving rise to said pseudo-graphs, can be sampled near-uniformly.

Definition 3.2. An *ordered simple pseudo-graph*, abbreviated OSP-graph, is a simple pseudo-graph with a total order on its vertices. In this paper, we will assume that the vertices of an OSP-graph are subsets of the positive integers under the usual ordering "$<$". An OSP-graph G is said to be *order-pressable* if there exists some initial segment of $V(G)$ that is a successful pressing sequence.

Definition 3.3. It was shown in [9] that pressing the vertices of a simple-pseudo-graph is essentially equivalent to performing Gaussian elimination with no row swaps on its adjacency matrix; therefore, the length of any successful pressing sequence of a simple pseudo-graph is the $\mathbb{F}_2$-rank of its adjacency matrix. Thus, we define the *rank*

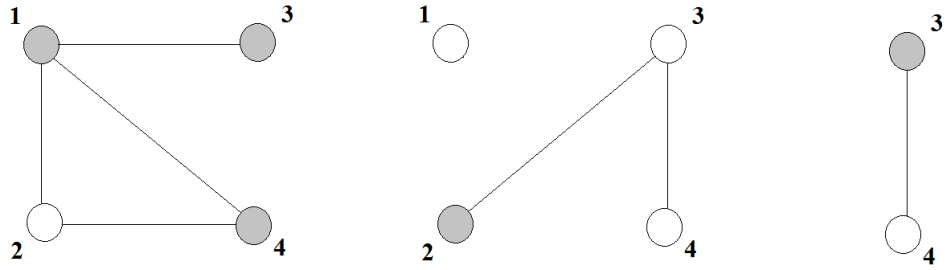

Figure 3.1 Left to right: an OSP-graph G; $G_{(1)}$, the result pressing 1 in G; and $G^{\mathbf{2}}$, the result of pressing and then removing vertices 1 and 2 in G. Loops are drawn a shaded vertices.

of a simple pseudo-graph to be the $\mathbb{F}_2$-rank of its adjacency matrix. The rank of a simple pseudo-graph on n vertices can vary from 0 (in the case that it is an edgeless simple pseudo-graph) to n (such as is the case in Figure 1). We say G is *full-rank* if its adjacency matrix is invertible over $\mathbb{F}_2$.

Call a matrix M "Cholesky" if there exists an upper-triangular matrix U so that $M = U^T U$. In [9] a proof was given that Cholesky decompositions of full-rank, $\mathbb{F}_2$ matrices are unique; in [10] it was shown that for every OSP-graph and adjacency matrix A there exists a particular Cholesky decomposition of A that encodes the pressing instructions for G.

Definition 3.4. Let G be OSP-graph with adjacency matrix A (whose rows and columns are ordered by the identity permutation). The *instructional Cholesky root of* G (over $\mathbb{F}_2$) is the upper triangular matrix U where for all $(i, j) \in [n] \times [n]$, $U[i, j] = 1$ if and only if $ij \in E(G_{\mathbf{i-1}})$. In [10] it was shown that U satisfies that $U^T U = A$, therefore is a Cholesky decomposition of G.

The reason this matrix is called "instructional" is that it contains the instructions for how vertices affect one another during the corresponding pressing sequence: the (i, j) entry is 1 iff pressing i flips the state of j. Since the (instructional) Cholesky

matrices are upper-triangular we may also regard U as the adjacency matrix of a directed acyclic graph with vertex set $\{v \mid v$ is pressed at some point in the successful pressing sequence$\}$. Furthermore, the transitive closure of this digraph can be considered as a poset. Although it is possible to define these instructional posets for less-than-full-rank OSP-graphs, presently we are only concerned with the posets of full-rank OSP-graphs.

We refer to the set of looped vertices in a graph G by $\mathcal{L}(G)$ and the set of successful pressing sequences for G as $\Sigma(G)$.

Lemma 3.5 ([**9**], Theorem 9)**.** *Let G be a full-rank OSP-graph and $\sigma \in \Sigma(G)$. Let A be the adjacency matrix of G with rows and columns ordered by σ. $\sigma \in \Sigma(G)$ if and only if A has a Cholesky decomposition over $\mathbb{F}_2$.*

Definition 3.6. Let G be a full-rank OSP-graph and $\sigma \in \Sigma(G)$. Let U be the instructional Cholesky root of A=adj(G), with rows and columns ordered identically by σ, and D the digraph with vertex set $V(G)$ and adjacency matrix U. The *instructional poset of G under σ* is $\mathrm{Poset}(G, \sigma) = (V(G), \preceq)$ where $y \preceq x$ (equivalently $x \succeq y$) if there is an x to y path in D, i.e., $\mathrm{Poset}(G, \sigma)$ is the transitive closure of D.

We say *$\mathcal{P}$ is generated by G*, or equivalently *G is a generator of $\mathcal{P}$*, if $\mathrm{Poset}(G, \sigma) = \mathcal{P}$ for some $\sigma \in \Sigma(G)$. If σ is the natural order given by G (typically the identity permutation) we simply write $\mathrm{Poset}(G)$. We denote the set of instructional posets of an OSP-graph G by $\mathfrak{S}(G)$.

Example 3.7. Let $\mathcal{P}$ be a poset on the element set $[4] = \{1, 2, 3, 4\}$ with cover relations $1 \succ 3$, $2 \succ 3$, $3 \succ 4$. Then any OSP-graph that generates $\mathcal{P}$ must have an adjacency matrix $A = U^T U$ where U is of the form $\begin{bmatrix} 1 & 0 & 1 & * \\ 0 & 1 & 1 & * \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. It follows that $\mathcal{P}$

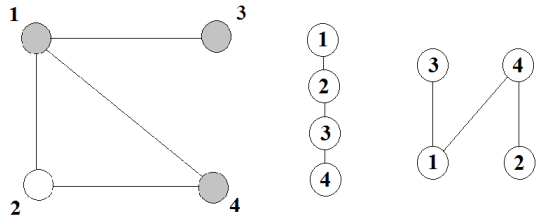

Figure 3.2 An order-pressable graph G and the Hasse diagrams of the two posets it generates.

has four generators, as shown below.

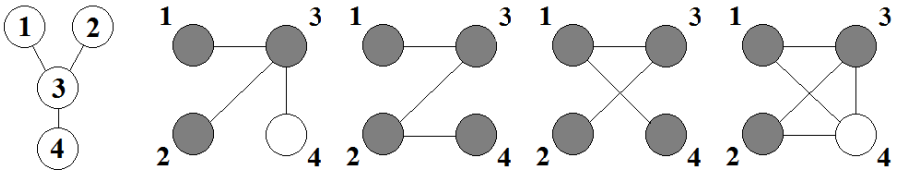

Figure 3.3 The Hasse diagram of $\mathcal{P}$ and its four generators.

We finish this section with two more lemmas from [9] which we will need below.

Lemma 3.8 ([9], Proposition 1)**.** *Let G be an OSP-graph. $\Sigma(G) \neq \emptyset$ if and only if every component of G containing two or more vertices contains a looped vertex.*

Lemma 3.9 ([9], Theorem 9)**.** *Let G be a full-rank OSP-graph and $\sigma \in \Sigma(G)$. Let A be the adjacency matrix of G with rows and columns ordered by σ. $\sigma \in \Sigma(G)$ if and only if every leading principal minor (over $\mathbb{F}_2$) of A is non-zero.*

3.2 Structure of Autonomous Posets

We denote the set of linear extensions of a poset $\mathcal{P}$ by $\mathrm{LE}(\mathcal{P})$.

Lemma 3.10. *If G is a full-rank OSP-graph then* $\mathrm{LE}(\mathcal{P}(G,\sigma)) \subseteq \Sigma(G)$ *for all* $\sigma \in \mathrm{Press}(G)$. *That is,* $\Sigma(G) = \bigcup_{\mathcal{P}\in\mathfrak{S}(G)} \mathrm{LE}(\mathcal{P})$.

Proof. Let $G = ([n], E)$ be an OSP-graph of rank n ordered by successful pressing sequence σ. By relabeling G we may assume σ is the identity permutation. Let A be the adjacency matrix of G (with rows and columns ordered by σ) and U be its instructional Cholesky root (identically ordered). Let $D = ([n], \overrightarrow{E})$ be the directed acyclic graph (aka "DAG") with adjacency matrix U. Let $\mathcal{P} = \mathrm{Poset}(G) = ([n], \preceq_P)$ and observe that if $(a, b) \in \overrightarrow{E}$ then $a \succeq_p b$.

Fix a linear extension $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$ of $\mathcal{P}$. By the previous observation, if $(\tau_i, \tau_j) \in \overrightarrow{E}$ then $\tau_i \succeq_P \tau_j$ and hence τ_i must appear before τ_j in $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$. Thus, $(\tau_i, \tau_j) \in \overrightarrow{E}$ implies $i \leq j \in \mathbb{N}$. By contraposition, we have that

$$i > j \text{ implies } (\tau_i, \tau_j) \notin \overrightarrow{E}.$$

Let P be the permutation matrix encoding τ. The previous assertion can be restated as

$$\left[P^T U P\right]_{i,j} = 0 \text{ for all } i > j.$$

Then $V = P^T U P$ is an upper-triangular matrix and

$$V^T V = (P^T U P)^T (P^T U P) = P^T U^T U P = P^T A P.$$

Observe that $P^T A P$ is a full-rank symmetric matrix with a Cholesky decomposition given by V. It follows from Lemma 3.5 that τ is a successful pressing sequence for G.

□

Definition 3.11. We say an OSP-graph G is an *autonomous graph* if $\Sigma(G) = \mathrm{LE}(\mathrm{Poset}(G))$. We say $\mathcal{P}$ is an *autonomous poset* if there exists an autonomous graph G that generates $\mathcal{P}$. That is, if there exists an OSP-graph G such that $\mathrm{Poset}(G, \sigma) = \mathcal{P}$ for some $\sigma \in \Sigma(G)$ and $\Sigma(G) = \mathrm{LE}(\mathcal{P})$.

In our main theorem, we will show that the set of autonomous posets is precisely the set of induced N-free and induced bowtie-free posets (referred to in [25] as "$\mathcal{V}$-posets").

Definition 3.12. For a graph G and a vertex $x \notin V(G)$ we let $x \oplus G$ be the graph with vertex set $V(G) \cup \{x\}$, edge set $E(G) \triangle \binom{\mathcal{L}(G) \cup \{x\}}{2}$, and $\mathcal{L}(x \oplus G) = \{x\}$. Equivalently, $x \oplus G$ is the graph that results from adding a looped vertex x to $V(G)$ and making it incident to each looped vertex in G to get an intermediate graph H, then switching the state of each edge (including loops and non-loops) in $N_H(x) \setminus \{x\}$. We refer to this process as *left-appending x to G*, we justify this terminology in the following observation.

Observation 3.13.

Consider OSP-graphs G and $H = x \oplus G$. Let $\tau = (\tau_1, \tau_2, \ldots, \tau_{n+1}) \in \Sigma(H)$. Since $\mathcal{L}(H) = \{x\}$ we have that $\tau_1 = x$. Furthermore, pressing x switches the state of every edge in $N_H(x)$ so $H^{(x)} = G$. Thus, the successful pressing sequences of H are exactly those resulting from left-appending x to the successful pressing sequences of G. If G is order-pressable with instructional Cholesky root U, then $x \oplus G$ is order-pressable and has instructional Cholesky root V that satisfies

$$V[i,j] = \begin{cases} U[i-1, j-1] & \text{if } i,j \geq 2 \\ 1 & \text{if } i = 1 \text{ and } j \in \mathcal{L}(G) \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3.14. For a graph G and a vertex $x \notin V(G)$ we let $G \oplus x$ be the graph with vertex set $V(G) \cup \{x\}$, edge set $E(G) \cup \{lx \mid l \in \mathcal{L}(G)\}$, and

$$\mathcal{L}(G \oplus x) = \begin{cases} \mathcal{L}(G) & \text{if } |V(G)| \text{ is odd} \\ \mathcal{L}(G) \cup \{x\} & \text{if } |V(G)| \text{ is even} \end{cases}$$

Equivalently, $G \oplus x$ is the graph that results from adding a vertex x to $V(G)$, making it incident to each looped vertex in G, and, if the resulting graph has an odd number of vertices, then we add a loop to x. We refer to this process as *right-appending x to G*.

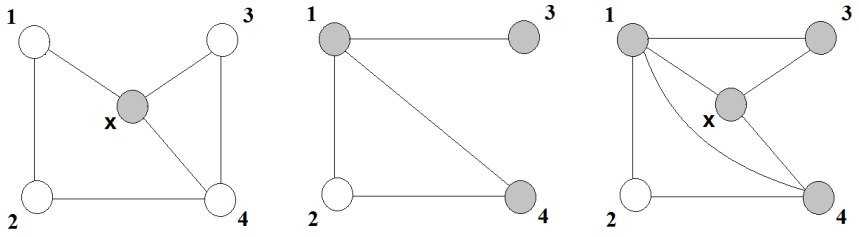

Figure 3.4 OSP-graphs $x \oplus G$, G, and $G \oplus x$, respectively.

Recall that the instructional Cholesky root of an OSP-graph is unique. In particular, if H is a full-rank graph and $V^T V$ is a Cholesky factorization of $A = \text{adj}(H)$ then V must be the instructional Cholesky root of H; from this we get the following observation.

Observation 3.15.

If G is order-pressable graph on n vertices and has instructional Cholesky root U then $G \oplus x$ is order-pressable and has instructional Cholesky root V where

$$V[i,j] = \begin{cases} U[i,j] & \text{if } i,j \leq n \\ 1 & \text{if } j = n+1 \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 3.16. *If G is autonomous then so is $x \oplus G$.*

Proof. Let $H = x \oplus G$. Since $\mathcal{L}(H) = \{x\}$ we have only one candidate vertex for an initial press. Furthermore, by Observation 3.13, $H^{(x)} = G$. It follows that any

pressing sequence must start with x and then continue as a pressing sequence for G. Therefore, the only instructional poset of H is that of G with a maximum element x appended. This demonstrates that H is also autonomous. □

Lemma 3.17. *If G is autonomous then so is $G \oplus x$.*

Proof. If $|V(G)| = 1$ and G is order-pressable then G is the graph on a single looped vertex and $G \oplus x$ is the graph with one looped vertex, one unlooped vertex and an edge between them; both of these graphs are uniquely pressable and therefore autonomous. Assume now towards an inductive argument that $|V(G)| > 1$ and that the inductive hypothesis holds for $|V(G)| - 1$. Let $G = ([n], E)$ and $H = G \oplus x$. By Observation 3.15, every pressing sequence of G can be extended to a pressing sequence for H by appending x to the end of the sequence. Therefore, we need only show that $|\Sigma(H)| = |\Sigma(G)|$ to conclude that H generates only one poset, namely, Poset(G) with the addition of a minimal element x. Since $N_H(x) = \mathcal{L}(G)$, the result of pressing x (should it be looped) in H would be a loopless graph – by Lemma 3.8 such a graph cannot be successfully pressed. Thus, every successful pressing sequence for H must begin with some element of $\mathcal{L}(H) \setminus \{x\} = \mathcal{L}(G)$. Choose and fix $j \in \mathcal{L}(G)$ that is the initial vertex in a successful pressing sequence for H. Assume, by way of contradiction, that j is not maximal in Poset(G). It follows that no successful pressing sequence for G begins with j, hence (by Lemma 3.8) $G^{(j)}$ contains a loopless component on two or more vertices; call this component C.

Consider now the result of pressing j in H. Since

$$N_H(j) \cap V(C) = \mathcal{L}(G) \cap V(C) = N_H(x) \cap V(C),$$

we have that every edge from x to $V(C)$ is deleted upon pressing j and $V(C)$ is a set of unlooped vertices in $H^{(j)}$. Finally, observe that any vertex that is incident to x in $H^{(j)}$ must be in a different component than C, as it was in G. It follows that $H^{(j)}$ contains a non-trivial loopless component, contradicting that j was the beginning of a

successful pressing sequence. Thus, the initial presses of H are those of G. Observing that $H^{(j)} = G^{(j)} \oplus x$ the result follows from the inductive hypothesis.

□

Lemma 3.18. *If $\mathcal{P}$ is an autonomous poset and k is a minimal element, then $\mathcal{P} - k$ is also an autonomous poset. Furthermore, if $\mathfrak{S}(G) = \{\mathcal{P}\}$ then $\mathfrak{S}(G-k) = \{\mathcal{P}-k\}$.*

Proof. Let $\mathcal{P}$ is an autonomous poset on n elements. By relabeling, we may assume that the elements of $\mathcal{P}$ are the integer set $[n] = \{1, 2, \ldots, n\}$, so that $(1, 2, \ldots, n)$ is a linear extension of $\mathcal{P}$. By relabeling the minimal elements, we may assume the element we remove is n.

Let $G = ([n], E)$ such that G generates only $\mathcal{P}$. Let A be the adjacency matrix of G. By Lemma 3.9 and the fact that $\mathfrak{S}(G) = \{\mathcal{P}\}$, for any permutation matrix P we have that $P^T AP$ has all non-singular leading principal minors (i.e., is LPN) if and only if P encodes a linear extension of $\mathcal{P}$. Let A' denote the $(n-1) \times (n-1)$ leading principal submatrix of A. Choose and fix an $(n-1) \times (n-1)$ permutation matrix P'.

Suppose $P'^T A' P'$ is LPN. Then

$$\left[\begin{array}{c|c} P' & 0 \\ \hline 0 & 1 \end{array}\right]^T \left[\begin{array}{c|c} A' & * \\ \hline * & a \end{array}\right] \left[\begin{array}{c|c} P' & 0 \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} P'^T A' P' & * \\ \hline * & a \end{array}\right]$$

is LPN if and only if $\left[\begin{array}{c|c} P'^T A' P' & * \\ \hline * & a \end{array}\right]$ is invertible, which occurs if and only if $\left[\begin{array}{c|c} A' & * \\ \hline * & a \end{array}\right]$ is invertible. Since A is invertible, we may conclude that if $P'^T A' P'$ is LPN then

$$\left[\begin{array}{c|c} P' & 0 \\ \hline 0 & 1 \end{array}\right]^T \cdot A \cdot \left[\begin{array}{c|c} P' & 0 \\ \hline 0 & 1 \end{array}\right]$$

is LPN. It follows that every successful pressing sequence for a graph G' with adjacency matrix A' can be extended to a successful pressing sequence for G by appending

n to the end of the sequence. Furthermore, the instructional Cholesky root of A' is the $(n-1)\times(n-1)$ leading principal submatrix of A; hence G', the graph whose adjacency matrix is A', generates $\mathcal{P}-n$. □

Lemma 3.19. *If $\mathcal{P}$ is an autonomous poset and k is a maximal element of $\mathcal{P}$, then $\mathcal{P}-k$ is also an autonomous poset.*

Proof. Suppose $\mathcal{P}$ is autonomous and G is an OSP-graph such that $\mathfrak{S}(G)=\{\mathcal{P}\}$. Let U be the $n\times n$ intructional Cholesky root of G. Then the intructional Cholesky root of $G^{(1)}$ is the $(n-1)\times(n-1)$ trailing principal submatrix of U. Thus, $G^{(1)}$ is a generator of $\mathcal{P}-k$. However, every successful pressing sequence of $G^{(1)}$ can be left-appended by k to obtain a successful pressing sequences for G. Hence, $\left|\Sigma\left(G^{(1)}\right)\right|=|\Sigma(G)|$, so that $\mathcal{P}-k$ is the only poset generated by $G^{(1)}$. □

Lemma 3.20. *Let $\mathcal{P}$ be an autonomous poset on $n\geq 3$ elements. If $\mathcal{P}$ has a maximum element x and a minimal element z such that x covers z, then any graph G that generates only $\mathcal{P}$ must satisfy $|\mathcal{L}(G)|=1$.*

Proof. By assumption that x is maximum we have that $\mathcal{P}$ is connected; therefore, if $y\in\mathcal{P}\setminus\{x,z\}$, then $x\succ y$ and y is incomparable to z. Suppose first that $n=3$, whence $\mathcal{P}=(\{x,y,z\},\preceq)$ with x covering both y and z. If G is an OSP-graph that generates $\mathcal{P}$ then the adjacency matrix A of G must have an instructional Cholesky root U encoding the cover relations of $\mathcal{P}$. Hence

$$U=\begin{bmatrix}1&1&1\\0&1&0\\0&0&1\end{bmatrix}\quad\text{and so}\quad A=U^TU=\begin{bmatrix}1&1&1\\1&0&1\\1&1&0\end{bmatrix}.$$

As the result holds for $n=3$, we proceed by induction on $n\geq 4$. Choose a minimal element $y\in\mathcal{P}\setminus\{x,z\}$, let $\mathcal{P}'=\mathcal{P}-y$, and let $G'=G-y$.

By Lemma 3.18, $\mathcal{P}'$ is autonomous and $\mathfrak{S}(G')=\{\mathcal{P}'\}$. Furthermore, $\mathcal{P}'$ has a maximum element x and a minimal element z such that x covers z, so we may apply

the inductive hypothesis; $|\mathcal{L}(G')| = 1$, in particular, $\mathcal{L}(G') = \{x\}$ (since it must be a pressable vertex). It follows that $\mathcal{L}(G) \subseteq \{x, y\}$. Assume, by way of contradiction, that $y \in \mathcal{L}(G)$. If $xy \notin E(G)$ then pressing y would create a looped vertex in every component of $G^{(y)}$, therefore there is a pressing sequence that begins with y, contradicting that $\mathcal{P}$ is autonomous. Thus, we must conclude that $xy \in E(G)$. Since z is a minimal element covered by x, then z is an isolated looped vertex in $G^{(x)}$ and hence $N_G(z) = N_G(x)$. In particular, $yz \in E(G)$.

Let $S = N_G(x) \setminus N_G(y)$ and $T = N_G(y)$. Assume, towards a contradiction, that $S \neq \emptyset$. Observe that $sx, sz \in E\left(G^{(y)}\right)$ for all $s \in S$ and hence there is a connected component in $G^{(y)}$ containing x and z (as well as the elements of S), and z is looped in $G^{(y)}$. Every other connected component in $G^{(y)}$ was created by deleting an edge between the vertices of T and hence contains an element of T which is now looped. It follows that $G^{(y)}$ can be successfully pressed, which is a contradiction. Thus, we may proceed under the assumption that $S = \emptyset$.

If $v \in N_G(y) \setminus N_G(x)$ then v is looped in $G^{(y)}$, $xy \in E\left(G^{(y)}\right)$, and every other connected component in $G^{(y)}$ was created by deleting the edge between two unlooped vertices and therefore would contain a looped vertex. It follows that $N_G(y) = N_G(x)$, therefore x and y can be interchanged in any successful pressing sequence. This contradicts that G is autonomous, so we must conclude that $y \notin \mathcal{L}(G)$, as desired. □

Definition 3.21. Let $\mathcal{P} = (X, \preceq)$ be a poset. We say (a, b, c, d) is an *occurrence of the pattern* N in $\mathcal{P}$ if $\{a, b, c, d\} \subseteq X$ and $a \succ c$, $a \succ d$, and $b \succ d$. We say (a, b, c, d) is an *induced occurrence* of the pattern N in $\mathcal{P}$ if $a \succ c$, $a \succ d$, $b \succ d$ and otherwise a, b, c and d are pairwise incomparable.

We say (a, b, c, d) is an *occurrence of the pattern bowtie* in $\mathcal{P}$ if $\{a, b, c, d\} \subseteq X$ and $a \succ c$, $a \succ d$, $b \succ c$, and $b \succ d$. We say (a, b, c, d) is an *induced occurrence* of the pattern bowtie in $\mathcal{P}$ if $a \succ c$, $a \succ d$, $b \succ d$ and otherwise a, b, c and d are pairwise incomparable.

We say $\mathcal{P}$ is *induced N-free* if it contains no induced occurrences of the pattern N. Similarly, $\mathcal{P}$ is *induced bowtie-free* if it contains no induced occurrences of the pattern bowtie.

It is worth noting that the literature varies on the definitions of "N-free poset". In our terminology a poset may include an occurrence of the pattern N yet be induced N- and bowtie-free. Such an example is the poset $\mathcal{P} = ([4], \{1 \succ 2 \succ 3 \succ 4\})$.

Lemma 3.22. *Autonomous posets are induced N-free.*

Proof. Let $\mathcal{P}'$ be an autonomous poset and assume towards a contradiction (a, b, y, z) is an induced occurrence of the pattern N in $\mathcal{P}'$. Let $\mathcal{P} = (X, \preceq)$ be the result of iteratively removing maximal and minimal elements from $\mathcal{P}'$ until a, b are the only maximal elements and y, z are the only minimal elements. By Lemmas 3.18 and 3.19, $\mathcal{P}$ is an autonomous poset with an induced occurrence of the pattern N, namely (a, b, y, z). Observe that if there exists $(a', b', y', z') \neq (a, b, y, z)$ that induces the pattern N in $\mathcal{P}$ then we may repeat the process of iteratively removing elements until only a', b', y', z' are extremal elements; thus, we proceed under the assumption that $\mathcal{P}$ has exactly one induced occurrence of the pattern N.

Choose $x \in \mathcal{P}$ such that $x \succ y$ (hence $x \neq y$). By assumption that only a and b are maximal in $\mathcal{P}$ we have that $a \succeq x$ or $b \succeq x$. Since (a, b, y, z) is an induced occurrence of the pattern N we have $b \not\succ y$ and hence $b \not\succeq x$, therefore $a \succeq x$. Observe that if $x \not\succeq z$ then (a, b, x, z) is an induced occurrence of the pattern N, contrary to assumption. Thus, $x \succ z$ (since $x \neq z$) and it follows that (x, b, y, z) is an induced occurrence of the pattern N implying that $x = a$, therefore a covers y.

Now choose $w \in \mathcal{P}$ such that $b \succ w$, observe that $w \neq a$. Since $b \not\succ y$ we have $w \not\succeq y$, hence $w \succeq z$. If $a \succ w \succ z$ then (a, b, y, w) is an induced occurrence of the pattern N, contrary to assumption. Hence, $a \succeq w$ if and only if $w = z$. However, if

$w \neq z$ then (a, w, y, z) is an induced occurrence of the pattern N, again contrary to assumption. Therefore, $w = z$ and it follows that b covers z.

By assumption that $\mathcal{P}$ is autonomous there exists a graph G that generates only $\mathcal{P}$. Fix such a G. Since $b \in \mathcal{P}$ is maximal, there is a successful pressing sequence beginning with $b \in V(G)$; thus $b \in \mathcal{L}(G)$. A sequence $\sigma' = (\sigma_1, \ldots, \sigma_k)$ is successful in $G^{(b)}$ exactly when $\sigma = (b, \sigma_1, \ldots, \sigma_k)$ is successful in G. Since G generates an autonomous poset then so does $G^{(b)}$ and hence $\mathcal{P} - b$ is autonomous. Further $\mathcal{P} - b$ meets the description of Lemma 3.20 so $\mathcal{L}\left(G^{(b)}\right) = \{a\}$, therefore $\mathcal{L}(G) = \{a, b\} \cup N_G(b)$. Now observe that if $v \in N_G(b)$, then pressing b affects v and hence $b \succeq v$. It follows that $N_G(b) = \{b, z\}$, therefore $\mathcal{L}(G) = \{a, b, z\}$. We proceed to show that z can be pressed in G, contradicting that $\mathfrak{S}(G) = \{\mathcal{P}\}$

Suppose first that $a \notin N_G(z)$. Then $N_G(z) \setminus \{b\} \subseteq \mathcal{L}\left(G^{(z)}\right)$ and $bv \in E\left(G^{(z)}\right)$ for all $v \in N_G(z) \setminus \{b\}$. It follows that any component created by pressing z in G has a looped vertex, and hence there is a successful pressing sequence starting with z in $\Sigma(G)$, a contradiction. Thus we must conclude that $\{a, b, z\} \subseteq N_G(z)$. Observe that the only elements comparable to y in $\mathcal{P}$ are a and y itself. Thus in any successful pressing sequence of G, a must be pressed before y and no other vertex affects (or is affected by) y. Hence $y \notin \mathcal{L}(G)$ and $N_G(y) = N_G(a) \setminus \{y\}$. Then $\{a, b, y, z\} \subseteq N_G(z)$. Since $ab, by \notin E(G)$ we have that $ab, by \in E\left(G^{(z)}\right)$ and hence a, b and y are path connected and $y \in \mathcal{L}\left(G^{(z)}\right)$. Similarly, if $v \in N_G(z) \setminus \{a, b, y, z\}$ then $bv \in E\left(G^{(z)}\right)$. It follows that every non-trivial component created by pressing z in G contains a looped vertex, therefore z is the initial press of for some $\sigma \in \Sigma(G)$, a contradiction.

□

Before proceeding, we state the main theorem of [10], which will be used below.

Theorem 3.23 ([**10**], Theorem 1)**.** *Let $G = ([n], E)$ be full rank with instructional Cholesky root U. Then G is uniquely pressable (i.e., has exactly one pressing se-*

quence) if and only if U has columns C_1, ..., C_n whose weights (number of nonzero entries) are w_1, ..., w_n respectively, satisfying:

- *For each j, if $C_j = (c_{1,j}, c_{2,j}, \ldots, c_{n,j})^T$ then* $\begin{cases} c_{i,j} = 1, & j - w_j < i \leq j \\ c_{i,j} = 0, & \text{otherwise} \end{cases}$.
- $1 = w_1 \leq w_2 \leq \cdots \leq w_n$.
- *$w_i > 2$ implies $w_{i+2} > w_i$, for $i \in [n-2]$.*
- *If w_i is odd for $i > 1$, then $w_j = j$ for all $j \geq i$.*

For an integer n, let $\Lambda(n)$ denote the poset with element set $[n]$ such that $n-2$ covers n and i covers $i+1$ for all $i \in [n-2]$. The Hasse diagram of $\Lambda(n)$ consist of two minimal elements ($n-1$ and n) below a chain of length $n-2$. Let $G_{\Lambda(n)}$ be the OSP-graph with vertex set $V(G) = [n]$, edge set $E(G) = \{(i, i+1) \mid i \in [n-1]\} \cup \{(1,1), (n-2, n)\}$.

Lemma 3.24. *$\Lambda(n)$ is an autonomous poset and $G_{\Lambda(n)}$ is the unique graph which generates only $\Lambda(n)$.*

Proof. Observe that for $n = 3$ we have only one instructional Cholesky that generates $\Lambda(n)$;

$$U = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

It follows that the only graph that generates $\Lambda(3)$ has adjacency matrix

$$A = U^T U = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

which is the adjacency matrix of $G_{\Lambda(3)}$.

For $n = 4$ we need only consider instructional Cholesky roots of the form:

$$\begin{bmatrix} 1 & 1 & *_1 & *_2 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $*_1, *_2 \in \{0, 1\}$. A quick check reveals that setting $*_1 = *_2 = 0$ yields a graph with two successful pressing sequences $(1, 2, 3, 4)$ and $(1, 2, 4, 3)$, and otherwise the resulting graph has 3 or more successful pressing sequences; hence the claim holds for $n = 4$.

We proceed by induction on $n \geq 5$. Let G be an OSP-graph that generates only $\Lambda(n)$. Since $\Lambda(n)$ has maximum element 1, we have that $1 \in \mathcal{L}(G)$ and $G^{(1)}$ has instructional poset $\Lambda(n) - 1$. But $\Lambda(n) - 1$ is isomorphic to $\Lambda(n - 1)$. By the inductive hypothesis we have that $G^{(1)}$ is isomorphic to $G_{\Lambda(n-1)}$.

Let U be the instructional Cholesky of G under the identity permutation. Let $A = U^T U$ and let U' be the $(n - 1) \times (n - 1)$ leading principal submatrix of U, $A' = U'^T U'$ and $G' = ([n - 1], E')$ the graph with adjacency matrix A'. Choose and fix $\sigma \in S_n$ such that $\sigma(n) = n$ and let P_σ be the permutation matrix encoding σ. Let $P_{\sigma'}$ be the $(n - 1) \times (n - 1)$ leading principal submatrix of $P_{\sigma'}$ and σ' its corresponding permutation. Observe that since G is full-rank then A is invertible. Hence, $\sigma' \in \Sigma(G')$ if and only if $P_{\sigma'}^T A' P_{\sigma'}$ is in LPN form, which occurs if and only if $P_\sigma^T A P_\sigma$ is in LPN form, which in turn occurs if and only if $\sigma \in \Sigma(G)$.

Since $\Sigma(G) = \{(1, 2, \ldots, n - 2, n - 1, n), (1, 2, \ldots, n - 2, n, n - 1)\}$ we have that the only successful pressing sequence of G' is $\sigma' = (1, 2, \ldots, n - 2, n - 1)$ and hence G' is a uniquely pressable graph (has only one pressing sequence). By Theorem 3.23, if $U'[1, i] = 1$ then $U'[2, i] = U'[2, i + 1] = 1$ and hence for $2 \leq i \leq n - 2$ if $U[1, i] = 1$ then $U[2, i] = U[2, i + 1] = 1$. However the intructional Cholesky root of $G^{(1)}$ is the $(n - 1) \times (n - 1)$ trailing principal minor of U and $G^{(1)}$ is isomorphic to $G_{\Lambda(n-1)}$. It

follows that $U[2, i+1] = 0$ for all $3 \leq i \leq n-1$ thus $U[1, i] = 0$ for all $3 \leq i \leq n-1$, hence $[3, n-1] \cap N_G(1) = \emptyset$. Observe that by relabeling n to $n-1$ and vice-versa we can make the same argument and conclude that $n \notin N_G(1)$, therefore $U[1, n] = 0$. We conclude that $G = G_{\Lambda(n)}$.

□

For an integer n we let $X(n)$ denote the poset with element set $[n]$ so that 1 covers 3, $n-2$ covers n, and i covers $i+1$ for all $i \in [2, n-2]$. The Hasse diagram of $X(n)$ consist of a chain of length $n-4$ joining two minimal elements ($n-1$ and n) to two maximal elements (1 and 2).

Lemma 3.25. *$X(n)$ is not an autonomous poset.*

Proof. Assume, by way of contradiction, that $X(n)$ is an autonomous poset and let G be any graph that generates only $X(n)$. Every successful pressing sequence of G must begin with $1, 2, 3$ or $2, 1, 3$. Thus, $\{1, 2\} \subseteq \mathcal{L}(G)$. Since 3 must be looped after pressing 1 and 2, and since the instructional Cholesky root instructs that both 1 and 2 switch the state of 3 upon being pressed, then $3 \in \mathcal{L}(G)$. Observe that $X(n)-1$ and $X(n)-2$ are isomorphic to $\Lambda(n-1)$ and hence $G^{(1)}$ and $G^{(2)}$ are isomorphic to $G_{\Lambda(n-1)}$ and hence each have exactly one looped vertex. In particular, $\mathcal{L}\left(G^{(i)}\right) = \{j\}$ for $\{i, j\} = \{1, 2\}$. Since 1 and 2 are both maximal in $X(n)$ then $(1, 2) \notin E(G)$. It follows that $N_G(j) = N_{G^{(i)}}(j)$ for $\{i, j\} = \{1, 2\}$. Therefore, by considering the structure of $G_{\Lambda(n-1)}$, we see $N_G(1) \setminus \{1\} = N_G(2) \setminus \{2\} = \{3\}$; furthermore, $\mathcal{L}(G) = \{1, 2, 3\}$. Consider the result of pressing 3 in G: $(1, 2), (1, 4), (2, 4)$ become edges, 4 becomes looped, and every other vertex incident to 3 in G becomes incident to both 1 and 2 in $G^{(3)}$. Thus, there is exactly one component in $G^{(3)}$ and it contains a looped vertex at 4. By Lemma 3.8 there is a successful pressing sequence in G that begins with 3, a contradiction. We conclude that $X(n)$ is not an autonomous poset.

□

Lemma 3.26. *Autonomous posets are induced bowtie-free.*

Proof. Let $\mathcal{P}$ be an autonomous poset. By Lemma 3.22, $\mathcal{P}$ is induced N-free. Assume, towards a contradiction, that (a, b, y, z) is an induced occurrence of the pattern bowtie. By iteratively removing maximal and minimal elements, and by application of Lemmas 3.18 and 3.19, we may assume a, b, y, and z are the only extremal elements of $\mathcal{P}$, and that $\mathcal{P}$ does not properly contain another occurrence of the pattern bowtie.

If the only elements of $\mathcal{P}$ are a, b, y, z then

$$U = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and hence

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

which has a successful pressing sequence of $(4, 3, 2, 1)$, contrary to assumption.

Choose and fix $x \in \mathcal{P}$ such that $x \notin \{a, b, y, z\}$. Since x is not extremal in $\mathcal{P}$ we may assume, without loss of generality, that $a \succ x \succ y$. If $b \succ x \not\succ z$ then (a, b, x, z) induces a bowtie, contrary to assumption. Similarly, if $b \not\succ x \succ z$ then (x, b, y, z) induces a bowtie. Observe that if $b \not\succ x \not\succ z$ then (a, b, x, z) induces an N, contradicting Lemma 3.22. Thus we must proceed under the assumption that $b \succ x \succ z$.

Observe that the choice of x was arbitrary so any $w \in \mathcal{P} \setminus \{a, b, y, z\}$ must also satisfy $a \succ w \succ y$ and $b \succ w \succ z$. If x and w are incomparable then (a, b, x, w) and (x, w, y, z) induce a smaller bowtie, contrary to assumption. Hence, any two elements

in $\mathcal{P} \setminus \{a, b, y, z\}$ must be comparable, therefore $\mathcal{P} = X(m)$ for some $m \geq 5$. This contradicts Lemma 3.25.

□

3.3 Main Result

In [21] (and later in [25]) the authors gave a simple description of posets that are both induced N-free and induced bowtie-free which we include here as Definition 3.27 and Theorem 3.28.

Definition 3.27. A poset is called a *$\mathcal{V}$-poset* if it can be generated by beginning with the singleton poset and then iteratively applying any of the following three operations:

(1) a disjoint union,

(2) adding a new greatest element,

(3) adding a new least element.

Theorem 3.28 ([21], Theorem 4.3)**.** *A poset is induced N-free and induced bowtie-free if and only if it is a $\mathcal{V}$-poset.*

Theorem 3.29. *$\mathcal{P}$ is autonomous if and only if $\mathcal{P}$ is induced N-free and induced bowtie-free.*

Proof. By Lemmas 3.22 and 3.26, if $\mathcal{P}$ is autonomous then $\mathcal{P}$ is induced N-free and induced bowtie-free. By Theorem 3.28 it suffices to show that $\mathcal{V}$-posets are autonomous.

A poset on one element is autonomous as it corresponds to the uniquely pressable graph on a single looped vertex. We proceed by induction. Let $n \geq 2$ and assume that all $\mathcal{V}$-posets on $n-1$ vertices are autonomous. Let $\mathcal{P}$ be a $\mathcal{V}$-poset on n vertices. If $\mathcal{P}$ is the disjoint union of multiple posets then each of its connected subposets is a

smaller $\mathcal{V}$-poset. By inductive hypothesis for each connected subposet there is a graph that generates it and has only the pressing sequences dictated by said subposet. It follows that in this case $\mathcal{P}$ is autonomous as well. Suppose now that $\mathcal{P}$ is connected. It then follows that $\mathcal{P}$ has a unique maximal or a unique minimal element. Let $\mathcal{P} - x$ be the result of removing a unique maximal or minimal element from $\mathcal{P}$. Observe that $\mathcal{P} - x$ is a $\mathcal{V}$-poset and thus by induction is autonomous; let H be a graph such that $\mathfrak{S}(H) = \{\mathcal{P} - x\}$. By Lemmas 3.13 and 3.15, $x \oplus H$ or $H \oplus x$ generates only $\mathcal{P}$ and therefore is autonomous. □

3.4 $\mathcal{V}$-poset Recognition

For a poset $\mathcal{P}$ we let $n_{\mathcal{P}}$ and $e_{\mathcal{P}}$ denote the number of vertices and edges in the Hasse diagram of the poset, respectively. We let $h_{\mathcal{P}}$ denote the sum of the heights of components of $\mathcal{P}$ (the height of a poset is the length of its longest chain), $c_{\mathcal{P}}$ denote the number of components of $\mathcal{P}$, and $M_{\mathcal{P}}$ and $m_{\mathcal{P}}$ denote the number of maximal and minimal elements in $\mathcal{P}$, respectively.

Lemma 3.30. *If $\mathcal{P}$ is a $\mathcal{V}$-poset then*

$$e_{\mathcal{P}} = 2n_{\mathcal{P}} + c_{\mathcal{P}} - M_{\mathcal{P}} - m_{\mathcal{P}} - h_{\mathcal{P}} \leq 2n_{\mathcal{P}} - 2$$

Proof. We show that $e_{\mathcal{P}} = 2n_{\mathcal{P}} + 1 - M_{\mathcal{P}} - m_{\mathcal{P}} - h_{\mathcal{P}}$ for a connected poset; the equality above follows by summing over components, and the inequality is immediate. Observe that if $n_{\mathcal{P}} = 1$ then $\mathcal{P}$ is a poset one element and hence $(2n_{\mathcal{P}} + 1) - (M_{\mathcal{P}} + m_{\mathcal{P}} + h_{\mathcal{P}}) = 0 = e_{\mathcal{P}}$. Assume towards an inductive argument that $n_{\mathcal{P}} \geq 2$. Since $\mathcal{P}$ is connected it must have a unique minimal or maximal element, say x, which we assume will be maximal (as the argument is identical for a minimal element). Let $\mathcal{Q} = \mathcal{P} - \{x\}$. Then, by applying the inductive hypothesis to $\mathcal{Q}$,

$$e_{\mathcal{P}} - M_{\mathcal{Q}} = e_{\mathcal{Q}} = 2n_{\mathcal{Q}} + 1 - M_{\mathcal{Q}} - m_{\mathcal{Q}} - h_{\mathcal{Q}}$$

$$e_{\mathcal{P}} = 2n_{\mathcal{Q}} + 1 - m_{\mathcal{Q}} - h_{\mathcal{Q}} = 2(n_{\mathcal{P}} - 1) + 1 - m_{\mathcal{P}} - (h_{\mathcal{P}} - 1)$$

$$e_{\mathcal{P}} = 2n_{\mathcal{P}} - m_{\mathcal{P}} - h_{\mathcal{P}}$$

By noting that $M_{\mathcal{P}} = 1$, we have our result. □

We now give a different edge count that uses width (referred to as $w_{\mathcal{P}}$ in the statement) instead of heights. While both of these edge counts are necessary for the property of being a $\mathcal{V}$-poset, even when taken together, they are not sufficient.

Lemma 3.31. *If $\mathcal{P}$ is a $\mathcal{V}$-poset then*

$$e_{\mathcal{P}} = n_{\mathcal{P}} + w_{\mathcal{P}} - M_{\mathcal{P}} - m_{\mathcal{P}}$$

Proof. As in the previous proof, we show that $e_{\mathcal{P}} = n_{\mathcal{P}} + w_{\mathcal{P}} - M_{\mathcal{P}} - m_{\mathcal{P}}$ for a connected poset; the equality above follows by summing over components since the width of a disconnected poset is the sum of the width of its connected components (i.e. the length of a maximal antichain). Observe that if $n_{\mathcal{P}} = 1$ then $\mathcal{P}$ is a poset one element and $n_{\mathcal{P}} + w_{\mathcal{P}} - M_{\mathcal{P}} - m_{\mathcal{P}} = 0 = e_{\mathcal{P}}$. Assume towards an inductive argument that $n_{\mathcal{P}} \geq 2$. Since $\mathcal{P}$ is connected it must have a unique minimal or maximal element, say x, which we assume will be maximal (as the argument is identical for a minimal element). Let $\mathcal{Q} = \mathcal{P} - \{x\}$. Then, by applying the inductive hypothesis to $\mathcal{Q}$,

$$\begin{aligned} e_{\mathcal{P}} &= e_{\mathcal{Q}} + M_{\mathcal{Q}} = (n_{\mathcal{Q}} + w_{\mathcal{Q}} - M_{\mathcal{Q}} - m_{\mathcal{Q}}) + M_{\mathcal{Q}} \\ &= n_{\mathcal{Q}} + w_{\mathcal{Q}} - m_{\mathcal{Q}} = n_{\mathcal{P}} - 1 + w_{\mathcal{P}} - m_{\mathcal{P}} = n_{\mathcal{P}} - M_{\mathcal{P}} + w_{\mathcal{P}} - m_{\mathcal{P}}. \end{aligned}$$

□

We propose an algorithm for the recognition of autonomous posets that operates on an arbitrary directed acyclic graph whose transitive closure is the poset in question. As a subroutine, we employ an algorithm found in [29] that detects if a directed acyclic graph contains an induced copy of the pattern N and, if the input is found to be induced N-free, it also returns the transitive reduction of the input. The

aforementioned subroutine is guaranteed to run in $O(|V|+|E|)$. Observe that by the proof of Lemma 3.26, in order to determine if an induced N-free poset is a $\mathcal{V}$-poset we need only to verify that its transitive-reduction does not contain a sub-DAG that is isomorphic to $([4], \{(1,3),(2,3),(1,4),(2,4)\})$ (as done in Subroutine 2) and does not contain sub-DAG whose transitive closure (interpreted as a poset) is isomorphic to $X(n)$, $(n \geq 5)$.

Lemma 3.30 shows that if we present the poset by the transitively-reduced directed acyclic graph with cover relations as edges then the run-time is $O(|V|)$. Observe that in Subroutines 2 and 3 each edge is traversed at most twice, hence these algorithms have run-time $O(|V|+|E|)$. Thus the presented algorithm has the same run-time as Subroutine 1.

Algorithm 1

1: **input:** a directed acyclic graph D.
2: **output:** **true** or **false**. True if the transitive closure of D is a $\mathcal{V}$-poset, False otherwise.
3: Bool $\leftarrow$ **true**
4: **if** IsSeriesParallel(D)[Bool]=False **then**
5: Bool $\leftarrow$ **false**
6: **else**
7: D $\leftarrow$ IsSeriesParallel(D)[DAG]
8: **if** IsBowtieFree(D)= **false then**
9: Bool $\leftarrow$ **false**
10: **else**
11: **if** ClosureIsVPoset(D) = **false then**
12: Bool $\leftarrow$ **false**
13: **return** Bool

Subroutine 1: IsSeriesParallel()

1: **input:** a directed acyclic graph D.

2: **output:** (Bool, DAG). Bool= **true** when D has a series-parallel decomposition and Bool= **false** otherwise, and DAG is the transitive reduction of D.

3: Algorithm found in [29]

4: **return** (Bool, DAG)

Subroutine 2: IsBowtieFree()

1: **input:** an induced N-free, transitively reduced directed acyclic graph D.

2: **output: true** or **false**. False if some induced subgraph of D is isomorphic to the bowtie digraph $(\{a, b, c, d\}, \{(a, c), (a, d), (b, c), (b, d)\})$, True otherwise.

3: Bool $\leftarrow$ **true**, Current $\leftarrow \emptyset$, Parents $\leftarrow \emptyset$, Visited $\leftarrow \emptyset$

4: **for** $v \in V(D)$ **do**

5: **if** OutDegree$(v) = 0$ **then**

6: Current.Add(v)

7: **while** Current$\neq \emptyset$ **do**

8: **for** $v \in$ Current **do**

9: **for** $u \in$ InNeighborhood(v) **do**

10: Parents.Add(u)

11: **for** $v \in$ Parents **do**

12: **if** OutDegree$(v) > 1$ **then**

13: **for** $u \in$ OutNeighborhood(v) **do**

14: **if** InDegree$(u) > 1$ **then**

15: Bool $\leftarrow$ **false** (Break **while** loop)

16: Visited.Add(u)

17: **for** $v \in$ Current **do**

18: Visited.Add(v)

```
    Current ← ∅
    for v ∈Parents do
        if v ∉ Visited then
            Current.Add(v)
            Visited.Add(v)
    Parents ← ∅
return Bool
```

Subroutine 3: ClosureIsVPoset()

```
input: an induced N-free, induced bowtie-free, transitively reduced directed acyclic graph D.
output: true or false. True if the transitive closure of D is a 𝒱-poset, False otherwise.
Bool ← true, Current ← ∅, Parents ← ∅, Visited ← ∅, Multiple ← ∅
for v ∈ V(D) do
    if OutDegree(v) = 0 then
        Current.Add(v)
while Current ≠ ∅ do
    for v ∈ Current do
        if v ∈ Multiple and InDegree(v) > 1 then
            Bool ← false (Break while loop)
        for u ∈ InNeighborhood(v) do
            Parents.Add(u)
            if v in Multiple then
                Multiple.Add(u)
    for v ∈ Parents do
        if OutDegree(v) > 1 then
            Multiple.Add(v)
```

```
18:    for v ∈ Current do
19:        Visited.Add(v)
20:    Current ← ∅
21:    for v ∈Parents do
22:        if v ∉ Visited then
23:            Current.Add(v)
24:    Parents ← ∅
25: return Bool
```

Chapter 4

Cholesky Roots

4.1 Introduction

Over the complex field $\mathbb{C}$, a square matrix M is said to have a Cholesky factorization if there exists an upper-triangular matrix U so that $U^*U = M$, where * denotes the conjugate transpose of a matrix (or simply transpose when restricting to $\mathbb{R}$). For a prime power q, we say M has a Cholesky factorization if there exists an upper-triangular matrix U so that $U^TU = M$. Observe that not all matrices have a Cholesky factorization. For example, only symmetric matrices have Cholesky factorizations since

$$(U^*U)^* = U^*(U^*)^* = U^*U.$$

For a matrix with complex entries there are a multitude of equivalent characterizations that determine if a matrix allows a Cholesky factorization (see, e.g. [22]). One particular example of this is the notion of positive-definiteness which, with some care, can be extended to some finite fields. For a wonderful survey, and some surprising results, on positive-definiteness over finite fields see [19]. In [9], the authors describe a surprising connection between successful pressing sequences of an ordered simple pseudo-graph (OSP-graph) and Cholesky factorizations of its ordered adjacency matrices. In particular, they argue that an OSP-graph G with vertex set $\{v_1, v_2, \ldots, v_n\}$ can be pressed in order $v_1, v_2, \ldots, v_n$ exactly when the adjacency matrix A of G, with rows and columns ordered by $v_1, v_2, \ldots, v_n$, has a Cholesky factorization. This equivalency only holds for OSP graphs satisfying that each vertex appears in (each)

successful pressing sequence. In Chapter 2, we define a special *instructional* Cholesky factorizations that extends the above equivalency to OSP graphs whose pressing sequences do not include every vertex.

In the following sections we explore how many distinct Cholesky factorizations exist for a matrix with entries from $\mathbb{F}_2$.

4.2 A Bijective Argument

For all positive integer n, we let 1_n and 0_n denote the $n \times n$, $\mathbb{F}_2$ multiplicative and additive identity matrices (respectively). For $r \leq n$, we let $\mathcal{U}_n(r)$ be the set of $n \times n$, rank r, upper-triangular matrices with entries from $\mathbb{F}_2$. For $n \geq 1$ and $r \leq n$ we define

$$\begin{aligned} \mathcal{X}_n(r) &= \{U \in \mathcal{U}_n(r) \mid U^2 = 1_n\} \\ \mathcal{Y}_n(r) &= \{U \in \mathcal{U}_n(r) \mid U^2 = 0_n\} \\ \mathcal{Z}_n(r) &= \{U \in \mathcal{U}_n(r) \mid U^T U = 0_n\} \end{aligned}$$

and

$$X_n = \sum_{0 \leq r \leq n} |\mathcal{X}_n(r)|, \quad Y_n = \sum_{0 \leq r \leq n} |\mathcal{Y}_n(r)|, \quad Z_n = \sum_{0 \leq r \leq n} |\mathcal{Z}_n(r)|$$

Observation 4.1. For all $n \geq 1$, $X_n = Y_n$.

Proof. Let $\varphi : \mathbb{F}_2^{n \times n} \to \mathbb{F}_2^{n \times n}$ by $\varphi(A) = A + 1_n$. Then

$$U^2 = 0_n \Leftrightarrow (\varphi(U))^2 = 1_n$$

□

Theorem 4.2. *For all $n \geq 1$ and $0 \leq r \leq n$:*

$$|\mathcal{Y}_n(r)| = |\mathcal{Z}_n(r)|$$

Proof. Observe that

$$\mathcal{Y}_1(0) = \left\{ \begin{bmatrix} 0 \end{bmatrix} \right\} = \mathcal{Z}_1(0),$$

and $\mathcal{Y}_1(r) = \mathcal{Z}_1(r) = \emptyset$ for all $r \neq 0$. We proceed by induction. Let $n > 1$ and assume that $|\mathcal{Y}_{n-1}(r)| = |\mathcal{Z}_{n-1}(r)|$ for all $r \leq n-1$. Choose and fix a rank r, $n \times n$ upper-triangular matrix B. Observe that by Sylvester's rank inequality $\mathcal{Y}_n(n) = \mathcal{Z}_n(n) = \emptyset$, so we may proceed with the assumption that $r < n$. Let B' be the $n-1 \times n-1$ principal submatrix of B.

$$B^2 = \left[\begin{array}{c|c} B' & \mathbf{v} \\ \hline \mathbf{0}^T & b \end{array}\right] \left[\begin{array}{c|c} B' & \mathbf{v} \\ \hline \mathbf{0}^T & b \end{array}\right] = \left[\begin{array}{c|c} B'^2 & B'\mathbf{v} + b\mathbf{v} \\ \hline \mathbf{0}^T & b^2 \end{array}\right].$$

Then $B \in \mathcal{Y}_n$ if and only if $b = 0$ and $B'\mathbf{v} = \mathbf{0}$ and $B' \in \mathcal{Y}_{n-1}$. However $B'\mathbf{v} = \mathbf{0}$ if and only if $v \in Null(B')$, the null space of B'. If $B' \in \mathcal{Y}_{n-1}$ then the column space of B', $Col(B')$, must be a subset of $Null(B')$. It follows that if $B \in \mathcal{Y}_n$ then $\mathbf{v} \in Col(B')$ or $\mathbf{v} \in Null(B') \setminus Col(B')$.

It follows that for each r:

$$|\mathcal{Y}_N(r)| = |\mathcal{Y}_{N-1}(r)| \cdot 2^r + |\mathcal{Y}_{N-1}(r-1)| \cdot \left(2^{\dim(Null(B'))} - 2^{r-1}\right)$$

$$|\mathcal{Y}_N(r)| = |\mathcal{Y}_{N-1}(r)| \cdot 2^r + |\mathcal{Y}_{N-1}(r-1)| \cdot \left(2^{N-r} - 2^{r-1}\right)$$

Choose and fix a rank r, $N \times N$ upper-triangular matrix C. Let C' be the $N-1 \times N-1$ principal submatrix of C.

$$C^T C = \left[\begin{array}{c|c} C'^T & \mathbf{0} \\ \hline \mathbf{w}^T & c \end{array}\right] \left[\begin{array}{c|c} C' & \mathbf{w} \\ \hline \mathbf{0}^T & c \end{array}\right] = \left[\begin{array}{c|c} C'^T C' & C'^T \mathbf{w} \\ \hline \mathbf{w}^T C' & \mathbf{w}^T \mathbf{w} + c^2 \end{array}\right].$$

$C \in \mathcal{Z}_N$ if and only if $\mathbf{w}^T\mathbf{w} + c^2 = 0$ and $\mathbf{w}^T C' = \mathbf{0}$ and $C' \in \mathcal{Z}_{N-1}$. Equivalently $C \in \mathcal{Z}_N$ if and only if $C' \in \mathcal{Z}_{N-1}$ and

$$\overline{C}\overline{\mathbf{w}} = \left[\begin{array}{c|c} C'^T & \mathbf{0} \\ \hline \mathbf{1} & 1 \end{array}\right] \left[\begin{array}{c} \mathbf{w} \\ \hline c \end{array}\right] = \mathbf{0}.$$

This occurs if and only if $\overline{w} \in Null(\overline{C})$. Observe that if $c = 0$ then $\overline{w} \in Null(\overline{C})$ exactly when $w \in Row(C') \cap Null(\overline{C})$ or $w \in Null(\overline{C}) \setminus Row(C')$. On the other hand if $c = 1$ then $\overline{w} \in Null(\overline{C})$ exactly when $w \in Null(\overline{C}) \setminus Row(C')$.

It follows that for each r:

$$|\mathcal{Z}_N(r)| = |\mathcal{Z}_{N-1}(r)| \cdot 2^r + |\mathcal{Z}_{N-1}(r-1)| \cdot \left(2^{\dim(Null(\overline{C}))} - 2^{r-1}\right)$$

$$|\mathcal{Z}_N(r)| = |\mathcal{Z}_{N-1}(r)| \cdot 2^r + |\mathcal{Z}_{N-1}(r-1)| \cdot \left(2^{N-r} - 2^{r-1}\right)$$

□

Theorem 4.3.

$$Y_n = \begin{cases} \sum\limits_j \left[\binom{n}{n/2-3j} - \binom{n}{n/2-3j-1}\right] 2^{n^2/4-3j^2-j}, & \text{if } n \text{ is even} \\ \sum\limits_j \left[\binom{n+1}{(n-1)/2-3j} - \binom{n}{(n-1)/2-3j-1}\right] 2^{(n-1)^2/4+(n-1)/2-3j^2-2j}, & \text{if } n \text{ is odd} \end{cases}$$

Proof. Theorem 1 in [13]. □

Corollary 4.4.

$$Z_n = \begin{cases} \sum\limits_j \left[\binom{n}{n/2-3j} - \binom{n}{n/2-3j-1}\right] 2^{n^2/4-3j^2-j}, & \text{if } n \text{ is even} \\ \sum\limits_j \left[\binom{n+1}{(n-1)/2-3j} - \binom{n}{(n-1)/2-3j-1}\right] 2^{(n-1)^2/4+(n-1)/2-3j^2-2j}, & \text{if } n \text{ is odd} \end{cases}$$

Corollary 4.5. *For all $n \geq 1$, $\mathcal{Y}_n(r) = \mathcal{Z}_n(r) = \emptyset$ whenever $r \geq n/2$.*

Proof. If $A \in \mathcal{Y}_n(r)$ then $\mathrm{Col}(U) \subset \mathrm{Null}(U)$ where the inclusion is strict since $[0, \ldots, 0, 1]^T \in \mathrm{Null}(U) \backslash \mathrm{Col}(U)$. That is $r < n - r$. Since there is a rank-preserving bijection between $\mathcal{Y}_n(r)$ and $\mathcal{Z}_n(r)$ the result holds. □

Corollary 4.6. *Let A be a symmetric, rank r matrix with entries from $\mathbb{F}_2$. For each $k \in [n]$, let A_k be the leading principal submatrix of A. If A satistfies that $\det(A_k) = 1$ if and only if $k \leq r$ (a.k.a. A is in leading principal minors non-negative form) then*

the number of distinct Cholesky factorizations for A is

$$\begin{cases} \sum_j \left[\binom{n}{n/2-3j} - \binom{n}{n/2-3j-1} \right] 2^{n^2/4-3j^2-j}, & \text{if } n \text{ is even} \\ \sum_j \left[\binom{n+1}{(n-1)/2-3j} - \binom{n}{(n-1)/2-3j-1} \right] 2^{(n-1)^2/4+(n-1)/2-3j^2-2j}, & \text{if } n \text{ is odd} \end{cases}$$

where n is the corank of A.

Proof. Let r be the rank of A and let $A_{1,1}$ be the principal $r \times r$ submatrix of A:

$$A = \left[\begin{array}{c|c} A_{1,1} & A_{1,2} \\ \hline A_{2,1} & A_{2,2} \end{array}\right] = \left[\begin{array}{c|c} A_{1,1} & A_{1,2} \\ \hline A_{1,2}^T & A_{2,2} \end{array}\right]$$

If $B^T B = A$ is a Cholesky factorization of A then

$$B^T B = \left[\begin{array}{c|c} B_{1,1}^T & 0 \\ \hline B_{1,2}^T & B_{2,2}^T \end{array}\right] \left[\begin{array}{c|c} B_{1,1} & B_{1,2} \\ \hline 0 & B_{2,2} \end{array}\right] = \left[\begin{array}{c|c} B_{1,1}^T B_{1,1} & B_{1,1}^T B_{1,2} \\ \hline B_{1,2}^T B_{1,1} & B_{1,2}^T B_{1,2} + B_{2,2}^T B_{2,2} \end{array}\right]$$

But A has an instructional Cholesky of the form

$$V^T V = \left[\begin{array}{c|c} V_{1,1}^T & 0 \\ \hline V_{1,2}^T & 0 \end{array}\right] \left[\begin{array}{c|c} V_{1,1} & V_{1,2} \\ \hline 0 & 0 \end{array}\right] = \left[\begin{array}{c|c} V_{1,1}^T V_{1,1} & V_{1,1}^T V_{1,2} \\ \hline V_{1,2}^T V_{1,1} & V_{1,2}^T V_{1,2} \end{array}\right]$$

By uniqueness of Cholesky decompositions of full-rank matrices over $GF(2)$ we have that $B_{1,1} = V_{1,1}$. Then by invertibility we have

$$B_{1,2} = \left(V_{1,1}^T\right)^{-1} B_{1,1}^T B_{1,2} = \left(V_{1,1}^T\right)^{-1} V_{1,1}^T V_{1,2} = V_{1,2}$$

and hence

$$V_{1,2}^T V_{1,2} = B_{1,2}^T B_{1,2} + B_{2,2}^T B_{2,2} \Rightarrow B_{2,2}^T B_{2,2} = 0$$

Then

$$C^T C = \left[\begin{array}{c|c} V_{1,1}^T & 0 \\ \hline V_{1,2}^T & C_{2,2}^T \end{array}\right] \left[\begin{array}{c|c} V_{1,1} & V_{1,2} \\ \hline 0 & C_{2,2} \end{array}\right] = A$$

if and only if $C_{2,2}^T C_{2,2} = 0$ □

4.3 Asymptotic Behavior

Here we will look at the asymptotic behavior of Z. Let $C_n = \frac{\binom{2n}{n}}{n+1}$ denote the n^{th} Catalan number, let $(x)_k = x(x-1)\cdots(x-k+1)$ denote the descending factorial, and let

$$E_{2n} = \sum_j \binom{2n}{n-3j}\left(\frac{6j+1}{n+3j+1}\right)2^{-j(3j+1)} - \sum_j \binom{2n}{n}\left(\frac{6j+1}{n+1}\right)2^{-3j^2-j}$$

Then

$$|E_{2n}| = \left| C_n \sum_j \left(\frac{(n+1)_{3j+1}}{(n+3j+1)_{3j+1}} - 1 \right)(6j+1)2^{-3j^2-j} \right|$$

and

$$1 > \frac{(n+1)_{3j+1}}{(n+3j+1)_{3j+1}} = \frac{n+1}{n+3j+1}\cdots\frac{n-3j+1}{n+1} \geq \left(\frac{n-3j+1}{n+1}\right)^{3j+1}$$

However

$$\left(\frac{n-3j+1}{n+1}\right)^{3j+1} = \left(1+\frac{-3j}{n+1}\right)^{3j+1} = \sum_{t=0}^{3j+1}\binom{3j+1}{t}(-1)^t\left(\frac{3j}{n+1}\right)^t$$

$$\geq \sum_{t=0}^{1}\binom{3j+1}{t}(-1)^t\left(\frac{3j}{n+1}\right)^t = 1 - \frac{3j(3j+1)}{n+1} \geq 1 - \frac{9j^2}{n+1}$$

Hence

$$|E_{2n}| \leq \frac{C_n}{n+1}\left|\sum_{|j|\leq n/3} 9j^2(6j+1)2^{-3j^2-j}\right| \leq C_n\frac{7.671}{n+1}$$

Thus,

$$\begin{aligned}
Z_{2n} &= \sum_j \left[\binom{2n}{n-3j} - \binom{2n}{n-3j-1}\right]2^{n^2-3j^2-j} \\
\frac{Z_{2n}}{2^{n^2}} &= \sum_j \binom{2n}{n-3j}\left(\frac{6j+1}{n+3j+1}\right)2^{-3j^2-j} \\
\frac{Z_{2n}}{2^{n^2}} &= \left[\sum_j \binom{2n}{n}\left(\frac{6j+1}{n+1}\right)2^{-3j^2-j}\right] + E_{2n} \\
\frac{Z_{2n}}{2^{n^2}} &= C_n\left(\left[\sum_j (6j+1)\,2^{-3j^2-j}\right] + \frac{E_{2n}}{C_n}\right)
\end{aligned}$$

Using Wolfram Alpha yields

$$\lim_{n\to\infty} \sum_{|j|\leq n/3} (6j+1)2^{-3j^2-j} \quad \approx \quad 0.17755$$

and

$$\lim_{n\to\infty} \frac{E_{2n}}{C_n} \quad \leq \quad \lim_{n\to\infty} \frac{7.671}{n+1} = 0$$

It follows that

$$Z_{2n} \approx 0.17755 \cdot 2^{n^2} C_n$$

Moreover,

$$X_{2n} = Y_{2n} = Z_{2n} \approx 0.17755 \cdot 2^{n^2} C_n$$

Chapter 5

Uniform Sampling Algorithm

5.1 Introduction

In 1969, Jesse MacWilliams published a paper counting the order of the subgroup of matrices U in $GL(n,q)$ that satisfy $UU^T = I$. In order to do so, he first gave a recursive algorithm that counts the number of symmetric matrices with entries in $GF(q)$ of size $t \times t$ and rank r. By reversing this algorithm we are able to construct a method for uniformly and efficiently sampling symmetric matrices of a given size and rank over a finite field. In the context of simple pseudo graphs his means that we can randomly construct a simple pseudo-graph on t vertices whose successful pressing sequences will be of length r.

5.2 Algorithm for sampling

We begin with some notation that will be useful in understanding the construction of sampling algorithm.

Notation:

q denotes a prime power,

$GF(q)$ is the finite field of q elements,

$GL(n,q)$ the group of $n \times n$ invertible matrices with entries in GF_q,

$\mathcal{O}(n,q)$ the subgroup of matrices U in $GL(n,q)$ that satisfy $UU^T = I$,

$S(t, r, q)$ the group of symmetric matrices with entries in $GF(q)$ of size $t \times t$ and rank r,

$N(t, r, q)$ the number of symmetric matrices with entries in $GF(q)$ of size $t \times t$ and rank r.

The following key lemma was proven in [24]. Nevertheless, we present a brief proof here since it is instrumental in constructing the sampling algorithm.

Lemma 5.1. *Let A be a symmetric $t \times t$ matrix of rank r with entries in $GF(q)$ and let $N_i(A)$ denote the number of symmetric $(t+1) \times (t+1)$ matrices of rank $r+i$ with entries in $GF(q)$ that contain A as a leading principal submatrix.*

$$N_i(A) = \begin{cases} q^{t+1} - q^{r+1}, & \text{if } i = 2 \\ (q-1)q^r, & \text{if } i = 1 \\ q^r, & \text{if } i = 0 \\ 0, & \text{otherwise.} \end{cases}$$

Proof. Fix A and consider a matrix of the form $\left[\begin{array}{c|c} A & y^T \\ \hline y & y_0 \end{array}\right]$ where y is an $1 \times t$ row vector with entries in $GL(q)$ and $y_0 \in GL(q)$. If y is not linearly dependent on the rows of A then the resulting matrix has rank $r+2$. There are $q^t - q^r$ such choices for y. Since we are free to choose any entry for y_0 this yields $N_2(A) = q(q^t - q^r)$. Suppose now that y is a linear dependent on A, that is $y = \mathbf{n}A$ for some $1 \times t$ vector $\mathbf{n}$. If $y_0 = \mathbf{n}y^T$ then the resulting matrix has rank r, otherwise if $y_0 \neq \mathbf{n}y^T$ then the resulting matrix has rank $r+1$. Since there are q^r choices for y in $\text{span}(A)$ we have $N_1(A) = (q-1)q^r$ and $N_0(A) = q^r$. □

Observe that from this we obtain

$$N(t+1, r, q) = (q^{t+1} - q^{r-1})N(t, r-2, q) + (q-1)q^{r-1}N(t, r-1, q) + q^r N(t, r)$$

Fix t, q, r where $t > 0$, q a prime power, and $0 \leq r \leq t$. Assume that for all $r' \leq r$ we can sample $S(t, r', q)$ uniformly at random. Consider the following sampling algorithm for $S(t+1, r, q)$. Let

$$p_i = \begin{cases} q^r \frac{N(t,r,q)}{N(t+1,r,q)}, & \text{i=0} \\ (q-1)q^{r-1}\frac{N(t,r-1,q)}{N(t+1,r,q)}, & \text{i=1} \\ (q^{t+1}-q^{r-1})\frac{N(t,r-2,q)}{N(t+1,r,q)}, & \text{i=2} \end{cases}$$

and observe that

$$\sum_{i=0}^{2} p_i = \frac{q^r N(t,r,q) + (q-1)q^{r-1}N(t,r-1,q) + (q^{t+1}-q^{r-1})N(t,r-2,q)}{N(t+1,r,q)} = 1.$$

Select i with probability p_i then take a sample from $S(t, r-i, q)$ uniformly at random, call this matrix A. If $i = 2$, choose y uniformly at random from the $q^t - q^r$ vectors that are linearly independent of the rows of A, choose y_0 uniformly at random from $\{0, 1, \ldots, q-1\}$. Otherwise, if $i \neq 2$ choose, uniformly at random, a vector $\mathbf{n}$ with entries in $GF(q)$ and set $y = \mathbf{n}A$. Furthermore, if $i = 0$ set $y_0 = \mathbf{n}y^T$ and if $i = 1$ choose y_0 uniformly at random from $\{0, 1, \ldots, q-1\} \setminus \{\mathbf{n}y^T\}$.

Let $A' = \left[\begin{array}{c|c} A & y^T \\ \hline y & y_0 \end{array}\right]$, by the proof of Lemma 5.1 A' is a uniform selection among the matrices in $S(t+1, r, q)$ who contain A as a leading principal submatrix.

Fix positive integer n and integer $0 \leq r \leq n$. If $r = 0$ there is only one matrix to consider. Otherwise, choose $i \in \{0, 1, 2\}$ with probability $p_i := p_i(t, r, q)$ (above). Set $k = 1$. While $n > 1$, repeat the following process:

- set $i_k = i$
- replace t with $t - 1$
- replace r with $r - i$
- select and replace $i \in \{0, 1, 2\}$ with probability p_i

- replace k with $k+1$

Clearly this process ends since at each step t is reduced. If at the end of the process $r = 0$, set $A_0 = [0]$. Otherwise, choose a uniformly at random from $\{1, 2, \ldots, q-1\}$ and set $A_0 = [a]$. for each $i \in k$, construct A_{i+1} from A_i by uniformly increasing the rank of A_i by i_{k-i}. This results in a $t \times t$ matrix A_{k-1} with rank r.

Choose and fix $r \geq 0$ and $t \geq 0$. Select A uniformly at random from $S(t+1, r, q)$. Observe that if $t = 0$ the algorithm selects with A with probability $\frac{1}{q}$. Assume $t \geq 1$ and that the algorithm selects uniformly for all values up to $t-1$. The probability that the algorithm selects A is

$$\mathbb{P}(A) = \mathbb{P}(A|i=0)\mathbb{P}(i=0) + \mathbb{P}(A|i=1)\mathbb{P}(i=1) + \mathbb{P}(A|i=2)\mathbb{P}(i=2)$$

Let B be the $t \times t$ leading principal submatrix of A.

$$\mathbb{P}(A|i) = \mathbb{P}(B \in S(t, r-i, q))\frac{1}{N(t, r-i, q)}\frac{1}{N_i(B)}$$

where $N_i(B)$ is the number of symmetric $t+1 \times t+1$ matrices of rank $(r-i)+i$ that contain B as a leading principal submatrix.

$$\mathbb{P}(B \in S(t, r-i, q)) = \frac{N(t, r-i, q)}{N(t, r-2, q) + N(t, r-1, q) + N(t, r, q)}$$

so

$$\begin{aligned}
\mathbb{P}(A) &= \sum_{i=0}^{2} \frac{N(t, r-i, q)}{N(t, r-2, q) + N(t, r-1, q) + N(t, r, q)} \frac{1}{N(t, r-i, q)} \frac{1}{N_i(B)} p_i \\
&= \frac{1}{X}\left(\frac{N(t, r, q)p_0}{N(t, r, q)N_0(B)} + \frac{N(t, r-1, q)p_1}{N(t, r-1, q)N_1(B)} + \frac{N(t, r-2, q)p_2}{N(t, r-2, q)N_0(B)}\right) \\
&= \frac{1}{X}\left(\frac{N(t, r, q)}{N(t+1, r, q)} + \frac{N(t, r-1, q)}{N(t+1, r, q)} + \frac{N(t, r-2, q)}{N(t+1, r, q)}\right) = \frac{1}{N(t+1, r, q)}
\end{aligned}$$

where $X = N(t, r-2, q) + N(t, r-1, q) + N(t, r, q)$

Below, we present the algorithm in more detail.

Algorithm 1: NewRandomMatrix

input: An integer tuple: (n, r, q)

output: A $n \times n$, rank r, symmetric matrix with entries in $\mathbb{F}_q$

if $n = 1$ **then**

  if $r = 0$ **then**

    $M = \begin{bmatrix} 0 \end{bmatrix}$

  else

    $x \leftarrow$RandomNonZero($\mathbb{F}_q$)

    $M = \begin{bmatrix} x \end{bmatrix}$

else

  for i in $\{0, 1, 2\}$ **do**

    $p_i \leftarrow$ ProbFrom(n, r, i)

  Choose an element of $\{0, 1, 2\}$ with probability (p_1, p_2, p_3)

  $M \leftarrow$ NewRandomMatrix$(n - 1, r - j, q)$

  $M \leftarrow$ RankIncrease(M,j)

return M

Algorithm 2: RankIncrease

input: An $n \times n$, rank r, symmetric matrix M with entries from $\mathbb{F}_q$ and an integer $j \in \{0, 1, 2\}$

output: An $(n+1) \times (n+1)$, rank $r + j$, symmetric matrix with entries from $\mathbb{F}_q$

if j=2 **then**

  $V \leftarrow$ NotInSpanVector(M)

  $v \leftarrow$ RandomElement($\mathbb{F}_q$)

  return $\left[\begin{array}{c|c} v & V^T \\ \hline V & M \end{array}\right]$

7: **else**

8: $V \leftarrow$ InSpanVector(M)

9: $x = \sum_{i=1}^{n} r_i V_{i,1}$

10: **if** j=0 **then**

11: for $i \in [n]$: $r_i \leftarrow$ RandomElement$(\mathbb{F}_q)$

12: $v \leftarrow x$

13: **else**

14: Choose v uniformly at random from $\mathbb{F}_q \setminus \{x\}$

15: **return** $\left[\begin{array}{c|c} v & V^T \\ \hline V & M \end{array}\right]$

Algorithm 3: NotInSpanVector

1: **input:** An $n \times n$ matrix M of rank k with entries in $\mathbb{F}_q$

2: **output:** An $n \times 1$ vector in the complement of $span(M)$

3: $L' \leftarrow$ FindOutBasis(M)

4: Set $\ell = n - k$ and choose a random integer r from $[1, q^\ell - 1]$

5: Cast r to a q-ary vector R

6: $R \leftarrow FIELD(R)$

7: $V \leftarrow$InSpanVector(M)

8: **return** $L' \cdot R + V$

Algorithm 4: InSpanVector

1: **input:** An $n \times n$ matrix M of rank k with entries in $\mathbb{F}_q$

2: **output:** An $n \times 1$ vector in the span of M

3: $L \leftarrow$ FindInBasis(M)

4: for $i \in [1,k] : r_i \leftarrow RandomElement(\mathbb{F}_q)$

5: $V = \begin{bmatrix} r_1 & r_2 & \cdots & r_k \end{bmatrix}^T$

6: **return** $L \cdot V$

Algorithm 5: FindOutBasis

1: **input: a matrix** M

2: **output: a basis for the complement of** M

3: $B \leftarrow$FindInBasis(M)

4: find C to complete B to an $n \times n$ basis

5: **return** C

Algorithm 6: FindInBasis

1: **input: a matrix** M

2: **output: a basis for** M

3: use in place Gaussian elimination to find column basis to find basis B

4: **return** B

Algorithm 7: ProbFrom(n,r,i)

1: **input:** Integer tuple (n, r, i)

2: **output:** A probability p

3: Denom $\leftarrow$ Numb$(n-1, r, 0)$+Numb$(n-1, r-1, 1)$+ Numb$(n-1, r-2, 2)$

4: **return** Numb$(n-1, r-i, i)$/Denom

Algorithm 8: Numb(n, r, i)

1: **input:** Integer tuple (n, r, i)

2: **output:** The number of symmetric $(n+1) \times (n+1)$ matrices of rank $r+i$ with entries in $\mathbb{F}_q$ that can be generated by appending the same vector as a row and column, along with a corner entry, to any symmetric $n \times n$ rank r matrix with entries in $\mathbb{F}_q$

3: **if** i $= 0$ **then**

4: **return** NumbSymm(n,r,q) q^r

5: **else if** i $= 1$ **then**

6: **return** NumbSymm(n,r,q) $(q-1)q^r$

7: **else if** i $= 2$ **then**

8: **return** NumbSymm(n,r,q) $(q^{n+1} - q^{r+1})$

Algorithm 9: NumbSymm

1: **input:** An integer tuple (n, r, q) where $n \geq r \geq 0$ and q is a prime power

2: **output:** the number of symmetric rank r $n \times n$ matrices over $\mathbb{F}_2$

3: $k = \lfloor r/2 \rfloor$

4: $f(x) = q^{2x}/(q^{2x} - 1)$

5: $g(x) = q^{n-x} - 1$

6: **return** $\left(\prod_{i=1}^{k} f(i)\right)\left(\prod_{i=1}^{r-1} g(i)\right)$

Algorithm 10: RandomNonZero

1: **input:** q

2: **output:** a random element of $\mathbb{F}_q \setminus \{0\}$

3: **return** Random element from $\mathbb{F}_q \setminus \{0\}$

Chapter 6

The Pressing Space of a Graph

6.1 Introduction

In 1999, Hannenhali and Pevzner demonstrated that a shortest path of reversal edits from one permutation to another can be determined in polynomial-time. The polynomial-time argument is justified by observing that a set of permutations can be converted into a simple pseudo-graph in polynomial-time (see the introduction of this thesis) and then the length of such a conversion path is simply the rank (over $\mathbb{F}_2$) of the adjacency matrix of the graph (which can be done in sub-cubic time). On the other hand, this does not answer how many shortest path of reversal edits exist. Thus, we propose the following question:

Question 6.1. What is the computational complexity of determining the number of successful pressing sequences of a simple pseudo-graph?

For a full-rank, ordered simple psuedo-graph G on n vertices, there are $n!$ permutations to consider. Attempting to press G in order σ is equivalent to performing Gaussian elimination without pivoting on the adjacency matrix of G, this can be done in polynomial-time. Since the average graph has $\frac{n!}{2^n}$ successful pressing sequences, the exhaustive method is not a practical computation technique for determining $|\mathfrak{S}(G)|$.

6.2 Block Pressing and Exponential Complexity

In what follows we will give a method that demonstrate that $|\mathfrak{S}(G)|$ can be determined in exponential-time. For simplicity of the argument we (re)introduce some notation

here.

A *pressable graph* is a simple pseudo-graph with a looped vertex in every non-trivial component. A vertex in a pressable graph is said to be *stuck* if it is looped but pressing the vertex creates a loopless graph with at least one edge. A vertex in a pressable graph is said to be *eventually stuck* if it is looped but pressing the vertex creates a loopless component with at least one edge. A looped vertex that is not eventually stuck is referred to as a *pressable vertex*. In [9], the authors showed that if G is pressable graph then we can always find a pressable vertex by creating a set X of looped vertices with the fewest possible number of neighboring looped vertices and then selecting $x \in X$ such that $deg(x)$ is maximal in X.

Let $V' \subseteq V(G)$ and let $G' = G[V']$ the induced subgraph of G. If G' is a pressable graph then we can find a pressing sequence by iteratively finding a pressable vertex and pressing it. We say V' is *block pressable* if there exists a successful pressing sequence σ' of G' that can be extended into a successful pressing sequence in G. That is, if pressing V' in the order given by σ' does not create any (eventually) stuck vertices in G. For the following lemma we let $G_{\sigma'}$ denote the result of pressing the vertices in appearing in σ' in G.

Lemma 6.2. *Let $G = (V, E)$ be a simple pseudo-graph and let $V' \subset V$ and suppose G' is block pressable. Then $G_{\sigma'} = G_{\tau'}$ for all $\sigma', \tau' \in \mathfrak{S}(G')$.*

Proof. Let $G = (V, E)$ be an ordered simple pseudo-graph and let $G' = (V', E')$ be a block pressable subgraph of G. By re-indexing, we may assume that both G and G' are identity pressable. Let A be the adjacency matrix of G and A' the adjacency matrix of G'. Recall that a gaph is order pressable if and only if its ordered adjacency matrix is in leading principal minors non-negative form ([9]).

Then

$$A = \left[\begin{array}{c|c} A' & B \\ \hline B^T & C \end{array}\right]$$

and both A and A' are in leading principal minor form. Let τ' be any successful pressing sequence of G' and let $P_{\tau'}$ be the permutation matrix encoding τ'. Then

$$P_{\tau'}^T A P_{\tau'}$$

is in leading principal minor form. Consider

$$P_\tau^T A P_\tau = \left[\begin{array}{c|c} P_{\tau'} & 0 \\ \hline 0 & I \end{array}\right]^T \left[\begin{array}{c|c} A' & B \\ \hline B^T & C \end{array}\right] \left[\begin{array}{c|c} P_{\tau'} & 0 \\ \hline 0 & I \end{array}\right] = \left[\begin{array}{c|c} P_{\tau'}^T A' P_{\tau'} & P_{\tau'}^T B \\ \hline B^T P_{\tau'} & C \end{array}\right]$$

Then the columns of $P_{\tau'}^T B$ are linearly dependent on $P_{\tau'}^T A' P_{\tau'}$ exactly when the columns of B are linearly dependent on A'. It follows that τ' can be extended to a successful pressing sequence of G. □

Example 6.3. The vertices in G' can be be pressed in two orders: (v_1, v_3, v_2) and (v_3, v_1, v_2). The resulting graph is the same.

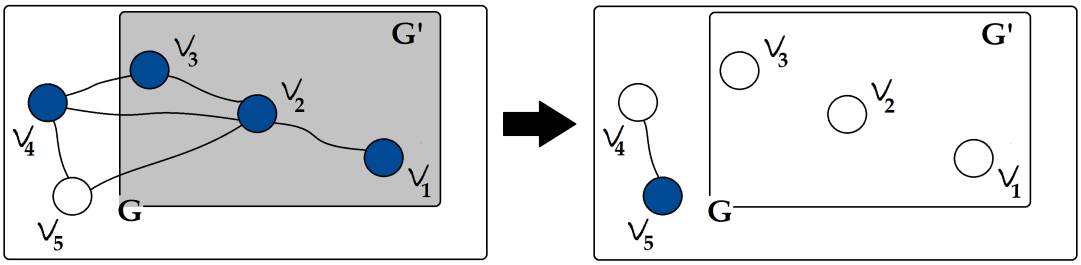

Figure 6.1 Block Pressing

Theorem 6.4. *Let G be a full-rank, ordered simple pseudo-graph. Then $|\mathfrak{S}(G)|$ can be determined in exponential-time.*

Proof. Let $G = ([n], E)$ and let $\mathcal{B}_n$ be the boolean lattice on the set $\{1, 2, \ldots, n\}$. Let $\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H}))$ be the Hasse diagram of $\mathcal{B}_n$.

We define two functions iteratively, one on the elements of $\mathcal{H}$ and one on its cover relations. $f : E(\mathcal{H}) \to \mathbb{Z}$ and $g : V(\mathcal{H}) \setminus \emptyset \to \mathbb{Z}$ defined iteratively by

$$f(\{\emptyset, \{x\}\}) = \begin{cases} 1, & \text{if } x \text{ is pressable in } G \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad g(\{x\}) = f(\{\emptyset, \{x\}\})$$

for each $x \in [n]$.

For each $S \subset [n]$ and $y \notin S$ let

$$f(\{S, S \cup \{y\}\}) = \begin{cases} 1, & \text{if } g(S) \neq 0 \text{ and } y \text{ is pressable in } G \text{ after pressing } S \\ 0, & \text{otherwise} \end{cases}$$

and for each $T \subseteq [n]$

$$g(T) = \sum_{t \in T} f(\{T \setminus \{t\}, T\}) g(T \setminus \{t\})$$

Observe that when $g(T) \neq 0$ it records the number of successful way to press the vertices of the induced graph $G[T]$. Hence $g([n]) = |\mathfrak{S}(G)|$. The number of points considered by f is $n2^{n-1}$ and the number of points considered by g is 2^n. Each function evaluation consisted of addition (linear time) and block pressing which is polynomial-time. Therefore, the time complexity of computing $g([n])$ is $O(2^n)$. $\square$

6.3 The Pressing Game Conjecture

A sequence $\pi = \pi_1, \ldots, \pi_k$ is a *subsequence* of permutation $\sigma = (\sigma_1, \ldots, \sigma_n)$ if $(\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k},) = (\pi_1, \ldots, \pi_k)$ for some $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. The sequence π is a *common subsequence* to σ and τ if it is a subsequence of both σ and τ. We let $\mathrm{lcs}(\sigma, \tau)$ denote the maximum length of such a subsequence, that is

$$\mathrm{lcs}(\sigma, \tau) = \max_{\pi \in S_n} (|\pi| : \pi \text{ is a common subsequence to } \sigma \text{ and } \tau)$$

Given two permutations, σ and τ, the edit distance between σ and τ, denoted $d(\sigma, \tau)$, is the minimum number of entries that must be removed and replaced (in any position

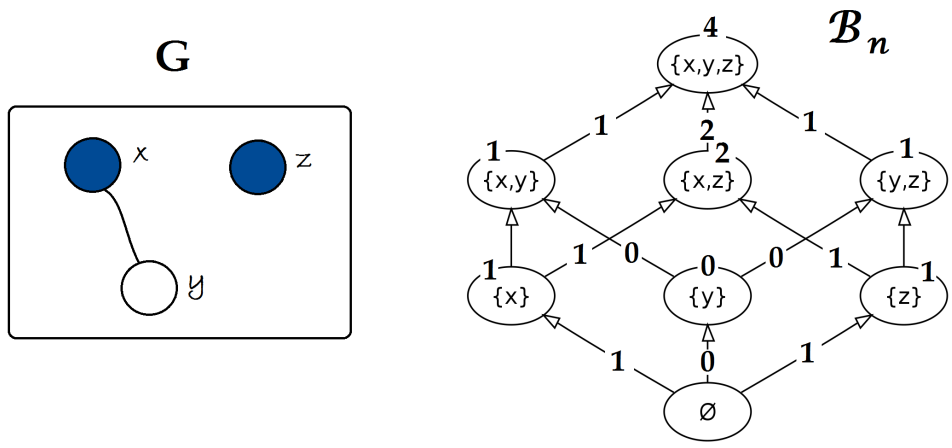

Figure 6.2 Computing the pressing number in exponential time

and order) from σ to achieve τ. A simpler formulation is

$$d(\sigma, \tau) = n - \text{lcs}(\sigma, \tau)$$

where $|\sigma| = n = |\tau|$.

Let $G = ([n], E)$ be an ordered simple pseudo-graph. The pressing space of G, denoted $\mathfrak{S}(G)$, is the set of all successful pressing sequences of G. From this we create a family of metagraphs $\{M_k(G)\}$ where $V(M_k(G)) = \mathfrak{S}(G)$ and $E(M_k(G)) = \{\{\sigma, \tau\} \mid d(\sigma, \tau) \leq k\}$.

In [4] we find the following conjecture.

Conjecture 6.5. *Every successful pressing sequence can be reached from every other one by a sequence of edits that involve at most four deletions or insertions.*

Using the terminology establish above the conjecture can be rephrased as $M_4(G)$ is a connected graph for all ordered simple pseudo-graphs G. The conjecture has been shown to be true on very specific types of graphs (see [4]), but remains unresolved in general. The following example demonstrates that $M_3(G)$ is not connected.

Example 6.6. Let $G = (V, E)$ be a simple pseudo graph with $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{(1, 1), (1, 2), (1, 5)(2, 3), (3, 4), (3, 6), (5, 5), (5, 6)\}$ where we use parenthesis to avoid ambiguity on the looped vertices. With some effort we determine that $\mathfrak{S}(G) = \{(1, 2, 3, 4, 5, 6), (5, 6, 3, 4, 1, 2)\}$. However,

$$d((1, 2, 3, 4, 5, 6), (5, 6, 3, 4, 1, 2)) = 4$$

since any longest increasing subsequence in $(5, 6, 3, 4, 1, 2)$ is of length 2.

This previous example demonstrates that if the pressing game conjecture is true (at distance 4) then it is best possible. While the relatively small size of the previous example may cast some doubt on the conjecture, it has held up (so far) to all tests. In the next section, we (joint work with Joshua Cooper and Peter Gartland) present a weaker conjecture and a proposed method to resolve it.

6.4 The Weak Pressing Game Conjecture

Conjecture 6.7 (WPGC)**.** *Every successful pressing sequence can be reached from another successful pressing sequence by a sequence of edits that involve at most four deletions or insertions.*

The previous conjecture is implied by the pressing game conjecture and can be restate as such:

WPGC: $M_4(G)$ has no isolated points

In what follows, we give an argument as to why Conjecture 6.7 should be true.

In Chapter 2, we describe the set of uniquely pressable graphs by characterizing their Cholesky roots. Below is the characterization again, but using some more colloquial language.

An $n \times n$ upper-triangular matrix U is the Cholesky matrix of a uniquely pressable graph provided

i. U has ones along the diagonal and super-diagonal;

ii. in the upper-triangular portion of a matrix, a one is never above a zero;

iii. if column j contains $J > 2$ ones, then column $j+1$ contains at least J ones (for all $j < n-1$), and column $j+2$ contains at least $J+2$ ones, (if $j \leq n-2$);

iv. if column $j > 1$ has an odd number of ones, then column k has only ones above the diagonal for each $j \leq k \leq n$.

Definition 6.8. We say that an $n \times n$ upper-triangular matrix U is *minimally non-uniquely pressable* (hereafter referred to as MNUP(n)) if it is not the Cholesky matrix of a uniquely pressable graph, but every one of its principal submatrices is. We say G is a minimally non-uniquely pressable graph, or simple *MNUP graph*, if its identity ordered instructional Cholesky is minimally non-uniquely pressable.

One can easily verify that there are no MNUP graphs on 3 vertices. Let M be a MNUP(n) matrix for some $n \geq 3$. Let U and V be the $(n-1) \times (n-1)$ leading and trailing principal submatrices of M, respectively. Since U is uniquely pressable we have that the first $n-1$ columns of M satisfy uniquely pressable laws i, ii, iii, iv. For simplicity of argument, for any matrix A let $\alpha(A)$ denote the entry in the first row and last column of A. For $i = 1, 2$, let X_i be the $n \times n$ matrix with a 1 in the i^{th} row and n^{th} column. We consider four cases.

Case 1: $\alpha(V) = 0$ and $\alpha(M) = 0$. M inherits laws i and ii from U and V. If $\alpha(U) = 0$ then M also abides to laws iii and iv, and therefore is a uniquely pressable. This is contrary to assumption. We proceed under the assumption that $\alpha(U) = 1$ and M is in violation of either law iii or iv. Since $\alpha(U) = 1$ we have that the penultimate column of V is all ones above the diagonal. By law iii the last column of V has at least $n-2$ ones as well. Thus M has exactly $n-2$ ones. Observe that if n is odd then V would violate law iv, therefore n is even. Finally, observe that $M + X_2$ satisfies all the uniquely pressable laws.

Case 2: $\alpha(V) = 1$ and $\alpha(M) = 1$. By law ii, since $\alpha(V) = 1$, V contains all ones in its final column. Observe that M then inherits every law (i, ii, iii, iv) from U. M is not a MNUP(n).

Case 3: $\alpha(V) = 1$ and $\alpha(M) = 0$. Observe that the final column of M must contain $n - 1$ ones (by law ii on V). Therefore M inherits laws i and ii from U and V. If n is even then law iv is violated, but $M + X_1$ does not violate any laws. Otherwise, if n is odd then the only way that M can break laws iii and iv is if the ante-penultimate column of M contains $n-2$ ones. Observe that in this case $M+X_1$ does not violate any laws.

Case 4: $\alpha(V) = 0$ and $\alpha(M) = 1$. M violates law ii. Assume, by way of contradiction, that both $M + X_1$ and $M + X_2$ violate some law. Since $M + X_2$ violates then it must be the case that the third row, n^{th} column entry of $M + X_2$ is a zero (otherwise it is the same argument as in case 2). Then V contains at most $n - 3$ ones in its final column. This implies, by law iii, that the penultimate column of V contains at most $n - 3$ ones and the ante-penultimate column contains at most $\max(2, n - 5)$ ones. By law ii, the final column of U contains at most $n - 3$ ones and the penultimate column of U contains at most $\max(2, n-5)$ ones. These measurement extend to M and therefore M satisfies all four laws, a contradiction.

Thus, we can make the following observation:

Observation 6.9. If M is a minimally non-uniquely pressable matrix then $M + X_1$ or $M + X_2$ is uniquely pressable.

Corollary 6.10. *There are at most* $3-(-1)^n \cdot 3^{\lfloor n/2 \rfloor - 1}$ *minimally non-uniquely pressable matrices on* n *vertices.*

Conjecture 6.11. *If* G *is a minimally non-uniquely pressable graph, then every successful pressing sequence can be reached from another successful pressing sequence by a sequence of edits that involve at most four deletions or insertions.*

Proposition 6.12. *The Weak Pressing Game Conjecture on MNUPs is equivalent to the Weak Pressing Game Conjecture.*

Proof. Observe that if the Weak Pressing Game Conjecture is true, then it must also be true on MNUPs. We proceed to show that it suffices to verify the conjecture on MNUPs. Assume, by way of contradiction, that the Weak Pressing Game on MNUPs does not imply the Weak Pressing Game Conjecture. Let U be the instructional Cholesky matrix of a non-uniquely pressable graph G that is identity pressable but no other pressing sequence within four edits of the identity is successful. Let U be the instructional Cholesky of G. Observing that any 1×1 principal submatrix if U is uniquely pressable and G, by assumption, is not uniquely pressable, we may conclude that U contains a MNUP as a principal submatrix, choose one and call it M. Let $A = U^T U$ and consider the block construction of A:

$$\left[\begin{array}{c|c|c} A_1 & A_2 & A_3 \\ \hline A_2^T & M^T M & A_4 \\ \hline A_3^T & A_4^T & A_5 \end{array}\right]$$

Since the WPGC holds on MNUPs, there is a permutation matrix P such that $P^T M^T M P$ is in pressing order and P can be converted into the identity permutation by removing and subsequently inserting at most four rows and columns (simultaneous operations on the same indexed row and column). However,

$$\left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & P^T & 0 \\ \hline 0 & 0 & I \end{array}\right] \left[\begin{array}{c|c|c} U_1 & U_2 & U_3 \\ \hline 0 & M & U_4 \\ \hline 0 & 0 & U_5 \end{array}\right]^T \left[\begin{array}{c|c|c} U_1 & U_2 & U_3 \\ \hline 0 & M & U_4 \\ \hline 0 & 0 & U_5 \end{array}\right] \left[\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & P & 0 \\ \hline 0 & 0 & I \end{array}\right] =$$

$$\left[\begin{array}{c|c|c} A_1 & A_2 P & A_3 \\ \hline P^T A_2 & P^T M^T M P & P^T A_4 \\ \hline A_3 & A_4 P & A_5 \end{array}\right]$$

which can be successfully pressed by block-pressing A_1, followed by block-pressing $M^T M$ (in the order given by P) and then block-pressing A_5. However we now have a new pressing order for G that is within four edits of the identity, a contradiction. $\square$

Conjecture 6.13. *The MNUPs satisfy the Weak Pressing Game Conjecture.*

In particular, we conjecture that if M is a MNUP(n) it can be pressed by (exactly) one of the following sequences:

- $(n-1, n, 2, 3, 4, \ldots, n-2, 1, 2)$, if $M[1, n] = M[2, n] = 0$;
- $(\ell, \ell+1, n, 2 \ldots, \ell-1, \ell+2, \ldots, n-1, 1)$, if $\ell \neq n$ is the first column other than 1 with an odd number of ones;
- $(n, 2, \ldots, k-1, k+2, \ldots, n, 1, k, k+1)$ if M satisfies that $M[i, n] = 0$ if and only if $2 \leq i \leq k+1$ for some even integer k;
- $(n, 2, \ldots, n-1, 1)$ otherwise.

Chapter 7

Future Directions and Open Questions

In this thesis we have explored several problems that are related to pressing sequences. Some of them we have been able to solve, many of them we have explored, and a few we have set on the back-burner. What follows are some future directions for this line of research as well as some open questions. We begin by addressing the elephant in the room.

Question 7.1. Is the pressing game conjecture true? (Conjecture: Every successful pressing sequence can be reached from every other one by a sequence of edits that involve at most four deletions or insertions.)

A very small first step in the direction towards proving the pressing game conjecture would be to show that the Weak Pressing Game Conjecture holds up. Should the pressing game conjecture be false, it may still be true that successful pressing sequences can be reached by a relatively small edits.

Question 7.2. Is it true that for a (full-rank) simple pseudo-graph on n vertices every successful pressing sequence can be reached from every other one by a sequence of edits that involve at most $p^*(n)$ deletions or insertions, where $p^*(n)$ is polynomial or less.

Another very open and quite interesting (at least to me) question is:

Question 7.3. What is the time complexity of determining the number of successful pressing sequences of a simple pseudo-graph?

We have shown that the complexity is no more than exponential. While this is an improvement over the previous upper-bound of super-exponential (derived from linear extensions), it is far from being computationally practical. Thus, if the complexity cannot be lowered to polynomial, we propose the following question:

Question 7.4. Can the number of successful pressing sequences of a simple pseudograph be approximated near-uniformly in polynomial time?

An affirmative to the previous question would likely lead to a near-uniform, efficient sampling algorithm for successful pressing sequences.

In Chapter 2 we discussed enumeration of Cholesky roots for the zero matrix. As a corollary we were able to discuss the number of distinct Cholesky factorizations of a matrix (over $\mathbb{F}_2$) that is in leading principal minor non-negative form. This count only offers a lower bound for the number of distinct Cholesky factorizations of a general symmetric matrix over $\mathbb{F}_2$.

Question 7.5. How many Cholesky factorizations are there for a symmetric matrix over $\mathbb{F}_2$? Can we generalize the results for $\mathbb{F}_q$?

In Chapter 3 we demonstrate that the successful pressing sequences of an OSP-graph are the linear extensions of a set of posets that arise from the instructional Cholesky roots of the graph. An autonomous graph has the property that its successful pressing sequences are all linear extensions of a single poset. In particular, in the autonomous case, this poset can be viewed as the intersection of all of the successful pressing sequences of the graph (interpreted as linear extensions). In the case that the OSP-graph is not autonomous then the posets are the intersections of pairwise disjoint families of successful pressing sequences. Thus, we have that if G is an OSP-graph then the instructional posets of G partition $\Sigma(G)$ into disjoint sets $S_1, S_2, \ldots, S_k$ satisfying that $\mathrm{LE}\left(\bigcap S_i\right) = S_i$ for each $i \in [k]$. Observe that this

partition is not sufficient to determine the instructional posets of a graph since, for example $\mathrm{LE}\left(\bigcap\{\sigma\}\right) = \{\sigma\}$.

Question 7.6. In general, how many distinct partitions of $\Sigma(G)$ into disjoint sets $\{S_i\}_i$ exist such that $\mathrm{LE}\left(\bigcap S_i\right) = S_i$ for each i?

Bibliography

[1] David A Bader, Bernard ME Moret, and Mi Yan. "A linear-time algorithm for computing inversion distance between signed permutations with an experimental study". In: *Journal of Computational Biology* 8.5 (2001), pp. 483–491.

[2] Vineet Bafna and Pavel A Pevzner. "Sorting by transpositions". In: *SIAM Journal on Discrete Mathematics* 11.2 (1998), pp. 224–240.

[3] Anne Bergeron. "A very elementary presentation of the Hannenhalli-Pevzner theory". In: *Annual Symposium on Combinatorial Pattern Matching*. Springer. 2001, pp. 106–117.

[4] Eliot Bixby, Toby Flint, and István Miklós. "Proving the pressing game conjecture on linear graphs". In: *Involve, a Journal of Mathematics* 9.1 (2015), pp. 41–56.

[5] Richard A Brualdi and Herbert J Ryser. *Combinatorial matrix theory*. Vol. 39. Cambridge University Press, 1991.

[6] Alberto Caprara. "Sorting by reversals is difficult". In: *Proceedings of the first annual international conference on Computational molecular biology*. ACM. 1997, pp. 75–83.

[7] Bhadrachalam Chitturi et al. "An (18/11) n upper bound for sorting by prefix reversals". In: *Theoretical Computer Science* 410.36 (2009), pp. 3372–3390.

[8] Josef Cibulka. "On average and highest number of flips in pancake sorting". In: *Theoretical Computer Science* 412.8-10 (2011), pp. 822–834.

[9] Joshua Cooper and Jeffrey Davis. “Successful pressing sequences for a bicolored graph and binary matrices”. In: *Linear Algebra and its Applications* 490 (2016), pp. 162–173.

[10] Joshua Cooper and Hays Whitlatch. “Uniquely Pressable Graphs: Characterization, Enumeration, and Recognition”. In: *Advances in Applied Mathematics, to appear. Preprint at arXiv:1706.07468* (2018).

[11] Reinhard Diestel. *Graph theory*. Springer, 2000.

[12] Th Dobzhansky and Alfred H Sturtevant. “Inversions in the chromosomes of Drosophila pseudoobscura”. In: *Genetics* 23.1 (1938), p. 28.

[13] Shalosh B Ekhad and Doron Zeilberger. “The number of solutions of X2= 0 in triangular matrices over GF (q)”. In: *Electron. J. Comb* 3 (1996), R2.

[14] Guillaume Fertin. *Combinatorics of genome rearrangements*. MIT press, 2009.

[15] William H Gates and Christos H Papadimitriou. “Bounds for sorting by prefix reversal”. In: *Discrete Mathematics* 27.1 (1979), pp. 47–57.

[16] Simona Grusea and Anthony Labarre. “The distribution of cycles in breakpoint graphs of signed permutations”. In: *Discrete Applied Mathematics* 161.10 (2013), pp. 1448–1466.

[17] Qian-Ping Gu, Shietung Peng, and Hal Sudborough. “A 2-approximation algorithm for genome rearrangements by reversals and transpositions”. In: *Theoretical Computer Science* 210.2 (1999), pp. 327–339.

[18] E Gyori and E Turan. “Stack of pancakes”. In: *Studia Scientiarum Mathematicarum Hungarica* 13 (1978), pp. 133–137.

[19] Erin Patricia Hanna. “A Quest for Positive Definite Matrices over Finite Fields”. In: (2018).

[20] Sridhar Hannenhalli and Pavel A Pevzner. “Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals”. In: *Journal of the ACM (JACM)* 46.1 (1999), pp. 1–27.

[21] Takahiro Hasebe and Shuhei Tsujie. “Order quasisymmetric functions distinguish rooted trees”. In: *Journal of Algebraic Combinatorics* 46 (2017), pp. 499–515.

[22] Roger A Horn, Roger A Horn, and Charles R Johnson. *Matrix analysis*. Cambridge university press, 1990.

[23] Thaynara Arielly de Lima and Mauricio Ayala-Rincon. “On the average number of reversals needed to sort signed permutations”. In: *Discrete Applied Mathematics* (2017).

[24] Jessie MacWilliams. “Orthogonal matrices over finite fields”. In: *The American Mathematical Monthly* 76.2 (1969), pp. 152–164.

[25] Valisoa Razanajatovo Misanantenaina and Stephan Wagner. “A Tutte-like polynomial for rooted trees and specific posets”. In: *arXiv preprint arXiv:1803.09623* (2018).

[26] Adam C Siepel. “An algorithm to enumerate sorting reversals for signed permutations”. In: *Journal of Computational Biology* 10.3-4 (2003), pp. 575–597.

[27] Alfred H Sturtevant and Th Dobzhansky. “Inversions in the third chromosome of wild races of Drosophila pseudoobscura, and their use in the study of the history of the species”. In: *Proceedings of the National Academy of Sciences* 22.7 (1936), pp. 448–450.

[28] László A Székely and Yiting Yang. “On the expectation and variance of the reversal distance”. In: *Acta Univ. Sapientiae, Mathematica* 1.1 (2009), pp. 5–20.

[29] Jacobo Valdes, Robert E Tarjan, and Eugene L Lawler. "The recognition of series parallel digraphs". In: *Proceedings of the eleventh annual ACM symposium on Theory of computing*. ACM. 1979, pp. 1–12.

[30] Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. "Efficient sorting of genomic permutations by translocation, inversion and block interchange". In: *Bioinformatics* 21.16 (2005), pp. 3340–3346.

Appendix A

Source Code and Examples

A.1 SageMath Code

I have appended this section to preserve some of the SageMath code that we use to explore pressing sequences. Much of this code can be directly attributed to Josh Cooper, and the less elegant code is a result of collaborative work between Peter Gartland, Erin Hanna and I. Blakeley Payne (nae Hoffman) has also created some code to that is similar to what has appeared here - while it does not appear here, it is likely that her work inspired some of our work (thanks Blakeley!)

The symbol "#" is used to denote a comment, and "<<<" is used following a line-break and means that the text which it precedes should be appended to the previous line. Throughout the code a finite field $GF(q)$ is assumed, prime power "q" should be defined prior to executing such codes. In the sections that follows I have included some sample outputs.

```
#########################################################################
# Input0: a vector ``V'' of length n(n+1)/2
# Input1: the corresponding integer ``n''
# Output: an n x n upper-triangular matrix.
#########################################################################
def triangularize(v,n):
    loc = 0
```

```
    outlist = [[0 for i in range(n)] for j in range(n)]
    for i in range(n):
        for j in range(i,n):
            outlist[i][j] = v[loc]
            loc += 1
    return Matrix(GF(q),outlist)


#######################################################################
# Input: a positive integer n
# Output: list of upper-triangular matrices whose square is
# the zero matrix
#######################################################################
def squarerootsofzero(n):
    size = int((n+1)*n/2)
    Z = matrix(GF(q),n,n,0)
    outlist = []
    for i in range(q^size):
        S = triangularize(Integer(i).digits(base=q,
        <<< padto=size),n)
        if S^2 == Z:
            outlist.append(S)
    return outlist


#######################################################################
# Input: a positive integer n
```

```
# Output: list of upper-triangular matrices that when
# multiplied by its transpose yields the zero matrix
#############################################################################
def Choleskyrootsofzero(n):
    size = int((n+1)*n/2)
    Z = matrix(GF(q),n,n,0)
    outlist = []
    for i in range(q^size):
        S = triangularize(Integer(i).digits(base=q,
        <<< padto=size),n)
        if S.transpose()*S == Z:
            outlist.append(S)
    return outlist


#############################################################################
# Input: a square matrix A
# Output: True if the matrix is presented in leading
# principal non-negative minors form, False otherwise.
#############################################################################
def is_LPN_k(A):
    n = len(list(A))
    k = rank(A)
    for j in range(1,k+1):
        M = A.matrix_from_rows_and_columns(range(j),range(j))
        if not M.is_invertible():
                return False
```

```
    for j in range(k+1,n+1):
        M = A.matrix_from_rows_and_columns(range(j),range(j))
        if M.is_invertible():
                return False
    return True


#########################################################################
# Input: an invertible square matrix A
# Output: True if the matrix is presented in leading
# principal non-negative minors form, False otherwise.
#########################################################################
def is_LPN_full(A):
    n = len(list(A))
    for j in range(1,n+1):
        M = A.matrix_from_rows_and_columns(range(j),range(j))
        if not M.is_invertible():
            return False
    return True


#########################################################################
# Input: a matrix A satisfying is_LPN_k(A)=True
# Output: the ''instructional'' Cholesky root of A
#########################################################################
def iChol(A):
    C=copy(A)
```

```
    n=len(list(C))
    B=matrix(GF(2), n, n, 0 )
    for i in range(n):
        B[i]=C[i]
        for k in range(i+1,n):
            C[k]= C[i,i]*C[i]*C[k,i]+C[k]
    return B


##########################################################################
# Input: A pair of posets, P and Q, over the same element set
# Output: minimum edit distance between linear extensions
# of P and Q
##########################################################################
def poset_edit_distance(P,Q):
    if set(P.list()) != set(Q.list()):
        print 'The labels are not the same.'
        return Infinity
    n = len(P.list())
    LP = map(lambda L: [P.list().index(x)+1 for x in L],
    <<< P.linear_extensions())
    LQ = map(lambda L: [P.list().index(x)+1 for x in L],
    <<< Q.linear_extensions())
    smallest_distance = Infinity
    for extP in LP:
        for extQ in LQ:
            perm = Permutation(extP).inverse()*
```

```
            <<<Permutation(extQ)
            d = n - perm.longest_increasing_
            <<<subsequence_length()
            if d < smallest_distance:
                smallest_distance = d
    return smallest_distance


#######################################################################
# Input: A symmetric n x n matrix A
# Output: True if A has exactly one successful pressing
# sequence, False otherwise
#######################################################################
def is_unique_full(A):
    n = len(list(A))
    presscount = 0
    for p in Permutations(n):
        P = p.to_matrix()
        if is_LPN_full(P.transpose()*A*P):
            presscount += 1
            if presscount > 1:
                return False
    if presscount == 1:
        return True
    else:
       return False
```

```
#####################################################################
# Input:A symmetric n x n matrix A
# Output: True if A has exactly two successful pressing
# sequences, False otherwise
#####################################################################
def is_twice_pressable_full(A):
    n = len(list(A))
    presscount = 0
    for p in Permutations(n):
        P = p.to_matrix()
        if is_LPN_full(P.transpose()*A*P):
            presscount += 1
            if presscount > 2:
                return False
    if presscount == 2:
        return True
    else:
       return False


#####################################################################
# Input: an invertible, symmetric n x n matrix A
# Output: the number of successful pressing sequences of A
#####################################################################
def pressing_number(A):
    n = len(list(A))
```

```
    presscount = 0
    for p in Permutations(n):
        P = p.to_matrix()
        if is_LPN_full(P.transpose()*A*P):
            presscount += 1
    return presscount


#######################################################################
# Input: a positive integer n and a positive integer k<n
# Output: Permutations of {1,...,n} of length k extended to
# length n by appending the missing elements in lex order
#######################################################################
def partialperm(n,k):
    permlist = Permutations(n,k)
    outlist = []
    for p in permlist:
        plist = list(p)
        for j in range(1,n+1):
            if j not in plist:
                plist.append(j)
        outlist.append(Permutation(plist))
    return outlist


#######################################################################
# Input: a symmetric n x n matrix A
```

```
# Output: the number of successful pressing sequences of A
#########################################################################
def pressingcount(A):
    n = len(list(A))
    k = A.rank()
    presscount = 0
    for p in partialperm(n,k):
        P = p.to_matrix()
        if is_LPN_k(P.transpose()*A*P):
            presscount += 1
    return presscount


#########################################################################
# Input: A symmetric n x n matrix A
# Output: the list of successful pressing sequences for A
#########################################################################
def pressinglistpartial(A):
    n = len(list(A))
    k = A.rank()
    presslist = []
    for p in partialperm(n,k):
        #print p
        P = p.to_matrix()
        if is_LPN_k(P.transpose()*A*P):
            presslist.append(p)
    return presslist
```

```
#########################################################################
# Input: An invertible symmetric, n x n matrix A
# Output: the list of successful pressing sequences for A
#########################################################################
def pressinglist(A):
    n = len(list(A))
    LIST=[]
    for p in Permutations(n):
        P = p.to_matrix()
        if is_LPN_full(P.transpose()*A*P):
            LIST.append(p)
    return LIST


#########################################################################
# Input: A graph G
# Output: The adjacency matrix of G with entries in GF(2)
#########################################################################
def binaryadjmx(G):
    return Matrix(GF(2),G.adjacency_matrix())


#########################################################################
# Input: A matrix M and a non-negative integer m
# Output: True if M has exactly m successful pressing
```

```
# sequences, False otherwise
#######################################################################
def has_number_of_PS(A,m):
    n = len(list(A))
    k = A.rank()
    presscount = 0
    for p in partialperm(n,k):
        P = p.to_matrix()
        if is_LPN_k(P.transpose()*A*P):
            presscount += 1
            if presscount > m:
                return False
    if presscount == m:
        return True
    else:
        return False


#######################################################################
# Input: a prime power q
# Output: a list indexing the elements of GF(q)
#######################################################################
def FIELD(q):
    return GF(q).list()


#######################################################################
```

```
# Input: positive integers n and r
# Output: The number of symmetric matrices of rank r and
# size n over GF(q)
#####################################################################
def NORMS(n,r):
    if min(n-1, r,n-r)<0:
        return 0
    else:
        k=floor(r/2)
        f(x)=q^(2*x)/(q^(2*x)-1)
        g(x)=q^(n-x)-1
        return prod(f(i) for i in (1..k))*prod(g(i)
        <<<for i in (0..r-1))


#####################################################################
# Input: positive integers n and r and integer i in {0,1,2}
# Output: the number of symmetric matrices of size n+1 which
# can be generated by increasing rank by i=0,1,2
#####################################################################
def NOM_gen(n,r,i):
    if i==0:
        return NORMS(n,r)*q^r
    elif i==1:
        return NORMS(n,r)*(q-1)*q^r
    elif i==2:
        return NORMS(n,r)*(q^(n+1)-q^(r+1))
```

```
    else:
        return 0


#########################################################################
# Input: positive integers n and r
# Output: DENOM(n,r) is used in the subroutines below
#########################################################################
def DENOM(n,r):
    return NOM_gen(n-1,r,0)+NOM_gen(n-1,r-1,1)
    <<<+NOM_gen(n-1,r-2,2)


#########################################################################
# Input: positive integers n and r and integer i in {0,1,2}
# Output: probabilities used in subroutines below
#########################################################################
def prob_from(n,r,i):
    if i in range(0,3):
        return max(NOM_gen(n-1,r-i,i)/DENOM(n,r),0)
    else:
        return 0


#########################################################################
# Input: a symmetric matrix M
```

```
# Output: information about pivot columns of M used in
# subroutines below
##########################################################################
def BASES(M):
    COMP=[]
    n=len(list(M))
    PIV=M.pivots()
    RPIV=M.pivot_rows()
    for i in range(n):
        E=matrix(GF(q),n,1,0)
        if i not in M.pivot_rows():
            e=copy(E)
            e[i,0]=1
            COMP.append(e)
    CB=[]
    for p in PIV:
        CB.append( M.matrix_from_columns([p])  )
    P=[]
    for p in PIV:
        P.append(p)
    return CB,COMP, P


##########################################################################
# Input: a symmetric matrix M
# Output: a vector in the span of M
##########################################################################
```

```
def inspanvector(M):
    L = BASES(M)[0]
    V = matrix(GF(q), len(list(M)),1,0)
    for i in range(len(list(L))):
        r = FIELD(q)[randint(0,q-1)]
        V = V + r*L[i]
    return V


#############################################################################
# Input: a symmetric matrix M
# Output: a vector not in the span of M
#############################################################################
def notinspan(M):
    l = len(BASES(M)[1])
    Y = matrix(GF(q), len(list(M)),1,0)
    s = q^l - 1
    if s ==1:
        w = 1
    else:
        w = random_between(1,q^l-1)
    W = w.digits(q, padto=l)
    for i in range(l):
        Y = Y + FIELD(q)[W[i]]*BASES(M)[1][i]
    return Y
```

```
#########################################################################
# Input: a symmetric matrix M
# Output: a vector that is the sum of a randomly chosen
# vector in the span of M and a randomly chosen vector not
# in the span of M
#########################################################################
def coolvector(M):
    return notinspan(M)+inspanvector(M)


#########################################################################
# Input: a symmetric matrix M and an integer k in {0,1,2}
# Output: a symmetric matrix of size one larger and rank
# increased by k
# Witty exceptions courtesy of Erin Hanna
#########################################################################
def rankinc(M,k):
    if k==2:
        r = M.rank()
        if r == M.nrows():
            print "I can't increase the rank by 2!"
        else:
            v = coolvector(M)
            p = M.rows()
            p[0:0] = (v.transpose())
            P = matrix(p)
            b = v.rows()
```

```
            b[0:0] = [(vector(GF(q), [0]))]
            c = matrix(b)
            G = P.transpose()
            g = G.rows()
            g[0:0] = (c.transpose())
            OMG = matrix(g).transpose()
            BLAH = GF(q).random_element()
            OMG[0,0] = BLAH
            return OMG
    else:
        n = M.nrows()
        eta = []
        for i in range(n):
            r = GF(q).random_element()
            eta.append(r)
        v = matrix(eta) * M
        p = M.rows()
        p[0:0] = (v)
        P = matrix(p)
        b = v.transpose().rows()
        b[0:0] = [(vector(GF(q), [0]))]
        c = matrix(b)
        G = P.transpose()
        g = G.rows()
        g[0:0] = (c.transpose())
        OMG = matrix(g)
        NY = matrix(eta)*v.transpose()
```

```
        if k==0:
            OMG[0,0] = NY[0][0]
        elif k==1:
            x = NY[0][0]
            while x == NY[0][0]:
                x = FIELD(q)[randint(0,q-1)]
            OMG[0,0] = x
        else:
            "you no do this"
        return OMG


#########################################################################
# Input: integers n,r,q
# Output: a uniformly randomly selected symmetric matrix of
#         size n, rank r with entries in GF(q), or a witty
#         exception message
#########################################################################
def nrmx(n,r,q):
    if min(n-1,r,n-r)<0:
        return "dummie"
    elif n==1:
        if r==0:
            M=matrix(GF(q),[[0]])
            return M
        elif r==1:
            M=matrix(GF(q),[ FIELD(q)[randint(1,q-1)]])
```

```
            return M
        else:
            return "exception in n==1 loop"
    else:
        P = [prob_from(n,r,0), prob_from(n,r,1),
        <<<prob_from(n,r,2)]
        X = GeneralDiscreteDistribution(P)
        j = X.get_random_element()
        M=nrmx(n-1,r-j,q)
        M = rankinc(M,j)
        return M
```

A.2 Cholesky Roots of Zero

2×2 square roots of zero:

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

2×2 Cholesky roots of zero:

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

3×3 square roots of zero:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

3×3 Cholesky roots of zero:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

4×4 square roots of zero:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

4×4 Cholesky roots of zero:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A.3 Large Uniquely Pressable Matrices

In Chapter 2 we discuss the set of uniquely pressable graphs by characterizing their Cholesky matrices. One can construct any uniquely pressable matrix by appending a column of full-weight to the end (and completing the matrix to be square), or by appending a new first row and writing a 1 in the entries of the first row exactly when the column previously had odd weight. We refer to to these processes as right-appending and left-appending (respectively). If n is even and U is a uniquely pressable $n \times n$ matrix, then the process of left-appending and then right-appending commutes. That is, all the $(n+2) \times (n+2)$ uniquely pressable matrices can be reached, without repetition, by performing the three following operations to the set of $n \times n$ uniquely pressable matrices: Left append twice, Right append twice, Left/Right append. Using this and the SageMath code below we generate random uniquely pressable matrices.

```
def LeftAppend(l):
    w=[1]
    w.extend(l)
    for i in range(1,len(w)):
        if w[i]%2==1:
            w[i]=w[i]+1
    return w

def RightAppend(l):
    x=copy(l)
    x.append(len(l)+1)
    return x
```

```
def RightRight(l):
    return RightAppend(RightAppend(l))

def LeftLeft(l):
    return LeftAppend(LeftAppend(l))

def RightLeft(l):
    return RightAppend(LeftAppend(l))

def LeftRight(l):
    return LeftAppend(RightAppend(l))

def RandomUniqueEven(n):
    if n==2:
        output=[1,2]
    else:
        i=randint(1,3)
        if i==1:
            output=LeftLeft(RandomUniqueEven(n-2))
        elif i==2:
            output=RightLeft(RandomUniqueEven(n-2))
        elif i==3:
            output=RightRight(RandomUniqueEven(n-2))
    return output

def SendUniqueToMatrix(L):
    n=len(L)
```

```
    M=matrix(GF(2),n,n,0)
    MM=copy(M)
    for i in range(n):
        for j in range(L[i]):
            MM[i-j,i]=1
    return MM


def RandomUniqueCholesky(n):
    if n%2==0:
        return SendUniqueToMatrix(RandomUniqueEven(n))
    elif n%2==1:
        l=RandomUniqueEven(n-1)
        i=randint(1,2)
        if i==1:
            l=LeftAppend(l)
        elif i==2:
            l=RightAppend(l)
        return SendUniqueToMatrix(l)
```

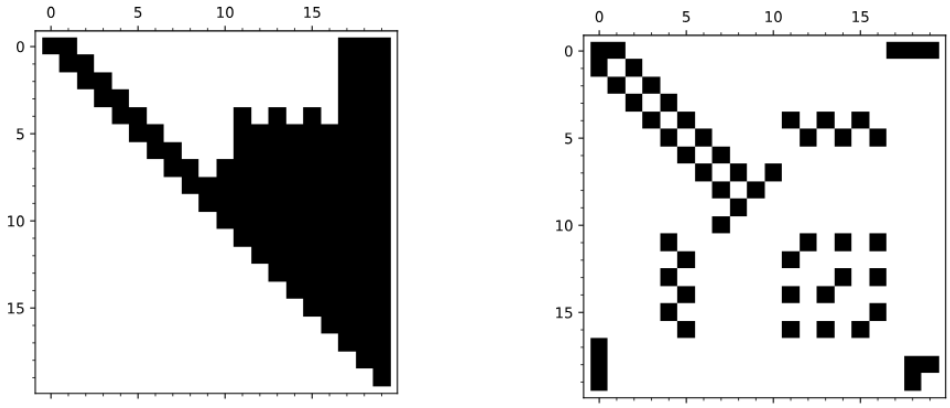

Figure A.1 Random Unique Cholesky and Adjacency Matrix. $N = 20$

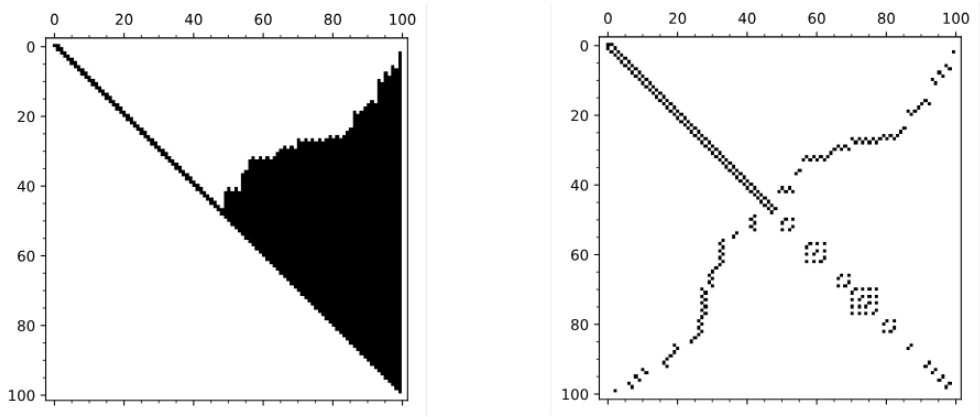

Figure A.2 Random Unique Cholesky and Adjacency Matrix. $N = 100$

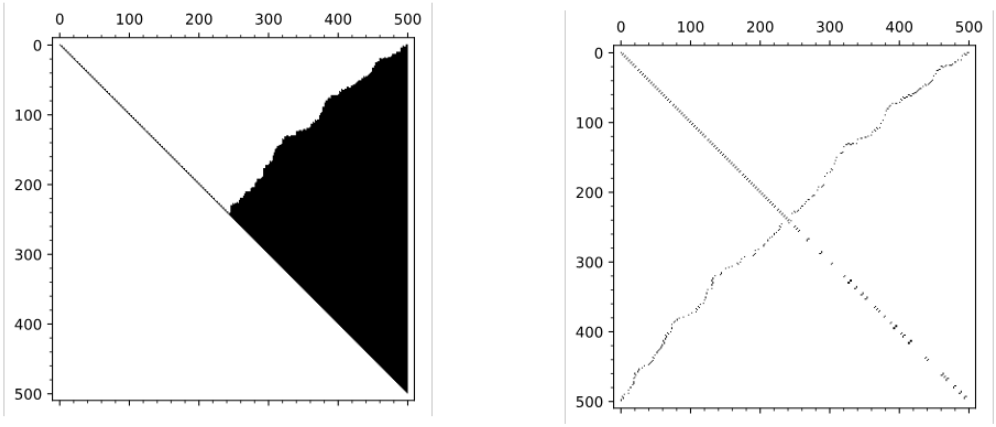

Figure A.3 Random Unique Cholesky and Adjacency Matrix. $N = 500$

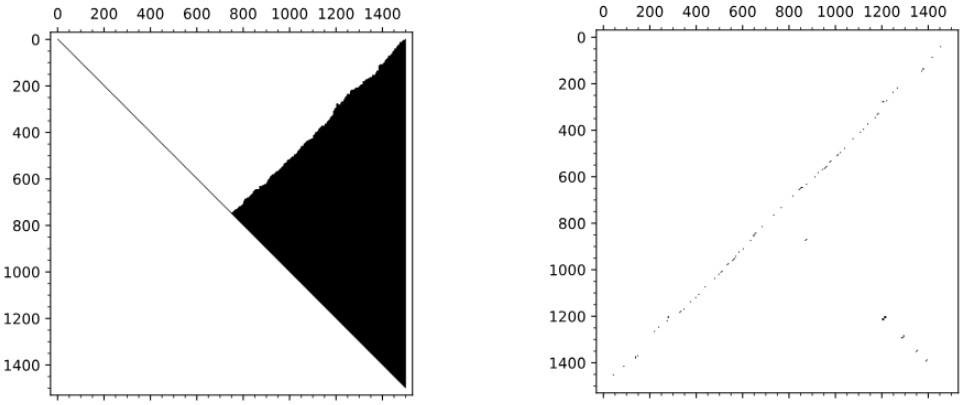

Figure A.4 Random Unique Cholesky and Adjacency Matrix. $N = 1500$