Various Topics in Combinatorics

By

KIRILL PARAMONOV

B.S. (Moscow State University) 2013

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Professor Anne Schilling
University of California, Davis

Professor Eugene Gorskiy
University of California, Davis

Professor James Sharpnack
University of California, Davis

Committee in Charge

2018

Contents

Tyrrell B. McAllister
June 2006
Meteorology

Rain is Wet

Abstract

This dissertation is . . . like . . . *abstract*.

Acknowledgments

I'd like to thank the little people.

CHAPTER 1

Introduction

This thesis consists of three parts, each corresponding to a single research project. Although the connection between the projects is loose, each of them represents a different part of Combinatorics: Chapter 2 for Algebraic Combinatorics, Chapter 3 for Enumerative Combinatorics, and Chapter 4 for applications of Combinatorics in Data Mining algorithms. Therefore, the reader can follow the chapters in any order, depending on their academic interests.

In this chapter we introduce the necessary background and describe the main results of each chapter to follow.

1.1. Crystals and type C Stanley Symmetric Functions.

The goal of this project is to find a Schur decomposition of the type C Stanley symmetric functions. We'll start with general theory of symmetric functions.

1.1.1. Schur functions. A *Young diagram* is a finite collection of boxes, or cells, arranged in left-justified rows, with row lengths in non-decreasing order. The non-decreasing row lengths form a *partition* $\lambda = (\lambda_1, \ldots, \lambda_s)$, called a *shape* of the Young diagram.

A *semistandard Young tableau* T is obtained by filling the boxes of the Young diagram with positive integers so that entries weakly increase along each row, and strictly increasing along each column. The *weight* of a semistandard Young tableau $w(T)$ is vector $(w_1, w_2, \ldots)$, where w_i counts the number of occurrences of the number i in T.

A *Schur function* $s_\lambda(\mathbf{x})$ is defined to be the characteristic function of the set of all semistandard Young tableaux of shape λ. That is,

$$s_\lambda(x_1, x_2, \ldots) = \sum_T x_1^{w_1} x_2^{w_2} \ldots = \sum_T \mathbf{x}^{w(T)},$$

where the sum is taken over all semistandard Young tableau of shape λ. Here, we used notation $\mathbf{x} = (x_1, x_2, \ldots)$ to represent the infinite vector of variables x_i.

Schur functions play an important role in algebraic combinatorics and representation theory because they represent characters of irreducible representations of the symmetric group, so they often serve as building blocks for characters of symmetric group representations. Therefore, decomposing a symmetric function F into a linear combination of Schur functions is important for understanding representation corresponding to function F.

The representation $F(\mathbf{x}) = \sum_\lambda K_\lambda s_\lambda(\mathbf{x})$ is called a *Schur decomposition*. Our goal is to find a Schur decomposition of type C Stanley symmetric functions.

1.1.2. Stanley symmetric functions. The *Coxeter group of type A_n*, denoted by S_n, is a finite group generated by $\{s_0, \ldots, s_{n-1}\}$ subject to the quadratic relations $s_i^2 = 1$ for all $i \in I = \{0, \ldots, n-1\}$, the commutation relations $s_i s_j = s_j s_i$ provided $|i - j| > 1$, and the braid relations $s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1}$ for all i.

The *Coxeter group of type C_n*, denoted by T_n, has the same generators and relations as S_n, except the braid relation $s_0 s_1 s_0 = s_1 s_0 s_1$ from type A changes to $s_0 s_1 s_0 s_1 = s_1 s_0 s_1 s_0$ in type C.

It is often convenient to write down an element of a Coxeter group as a sequence of indices of s_i in the product representation of the element. For example, the element $w \in T_3$ with $w = s_2 s_1 s_2 s_1 s_0 s_1 s_0 s_1$ is represented by the word $\mathbf{w} = 2120101$. A word of shortest length ℓ is referred to as a *reduced word* and $\ell(w) := \ell$ is referred as the length of w. The set of all reduced words of the element w is denoted by $R(w)$.

EXAMPLE 1.1. *The set of reduced words for type C word $w = s_2 s_1 s_2 s_0 s_1 s_0$ is given by*

$$R(w) = \{210210, 212010, 121010, 120101, 102101\}.$$

An element $v \in S_n$ is called *decreasing* if there is a reduced word $i_1 \cdots i_m$ for v such that $i_1 > \cdots > i_m$. The identity is considered to be decreasing. Given $w \in S_n$, a *decreasing factorization* of w is a factorization $w = w^k \cdots w^1$ with $w^k \cdots w^1$ being reduced and each w^i is decreasing.

CHAPTER 2

Crystal Analysis of type C Stanley Symmetric Functions

2.1. Introduction

Schubert polynomials of types B and C were independently introduced by Billey and Haiman [**BH95**] and Fomin and Kirillov [**FK96**]. Stanley symmetric functions [**Sta84**] are stable limits of Schubert polynomials, designed to study properties of reduced words of Coxeter group elements. In his Ph.D. thesis, T.K. Lam [**Lam95**] studied properties of Stanley symmetric functions of types B (and similarly C) and D. In particular he showed, using Kraśkiewicz insertion [**Kra89**, **Kra95**], that the type B Stanley symmetric functions have a positive integer expansion in terms of P-Schur functions. On the other hand, Stembridge [**Ste89**] proved that the P-Schur functions expand positively in terms of Schur functions. Combining these two results, it follows that Stanley symmetric functions of type B (and similarly type C) have a positive integer expansion in terms of Schur functions.

Schur functions $s_\lambda(\mathbf{x})$, indexed by partitions λ, are ubiquitous in combinatorics and representation theory. They are the characters of the symmetric group and can also be interpreted as characters of type A crystals. In [**MS16**], this was exploited to provide a combinatorial interpretation in terms of highest weight crystal elements of the coefficients in the Schur expansion of Stanley symmetric functions in type A. In this paper, we carry out a crystal analysis of the Stanley symmetric functions $F_w^C(\mathbf{x})$ of type C, indexed by a Coxeter group element w. In particular, we use Kraśkiewicz insertion [**Kra89**, **Kra95**] and Haiman's mixed insertion [**Hai89**] to find a crystal structure on primed tableaux, which in turn implies a crystal structure $\mathcal{B}_w$ on signed unimodal factorizations of w for which $F_w^C(\mathbf{x})$ is a character. Moreover, we present a type A crystal isomorphism $\Phi\colon \mathcal{B}_w \to \bigoplus_\lambda \mathcal{B}_\lambda^{\oplus g_{w\lambda}}$ for some combinatorially defined nonnegative integer coefficients $g_{w\lambda}$; here $\mathcal{B}_\lambda$ is the type A highest weight crystal of highest weight λ . This implies the desired decomposition $F_w^C(\mathbf{x}) = \sum_\lambda g_{w\lambda} s_\lambda(\mathbf{x})$ (see Corollary 2.30) and similarly for type B.

The paper is structured as follows. In Section 3.2, we review type C Stanley symmetric functions and type A crystals. In Section 2.3 we describe our crystal isomorphism by combining a slight generalization

of the Kraśkiewicz insertion [**Kra89**, **Kra95**] and Haiman's mixed insertion [**Hai89**]. The main result regarding the crystal structure under Haiman's mixed insertion is stated in Theorem 2.24. The combinatorial interpretation of the coefficients $g_{w\lambda}$ is given in Corollary 2.30. In Section **??**, we provide an alternative interpretation of the coefficients $g_{w\lambda}$ in terms of semistandard unimodal tableaux. Appendices 2.6 and 2.7 are reserved for the proofs of Theorems 2.24 and 2.26.

Acknowledgments. This research was partially supported by NSF grant DMS–1500050.

2.2. Background

2.2.1. Type C Stanley symmetric functions. The *Coxeter group* W_C of type C_n (or type B_n), also known as the hyperoctahedral group or the group of signed permutations, is a finite group generated by $\{s_0, s_1, \ldots, s_{n-1}\}$ subject to the quadratic relations $s_i^2 = 1$ for all $i \in I = \{0, 1, \ldots, n-1\}$, the commutation relations $s_i s_j = s_j s_i$ provided $|i - j| > 1$, and the braid relations $s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1}$ for all $i > 0$ and $s_0 s_1 s_0 s_1 = s_1 s_0 s_1 s_0$.

It is often convenient to write down an element of a Coxeter group as a sequence of indices of s_i in the product representation of the element. For example, the element $w = s_2 s_1 s_2 s_1 s_0 s_1 s_0 s_1$ is represented by the word $\mathbf{w} = 2120101$. A word of shortest length ℓ is referred to as a *reduced word* and $\ell(w) := \ell$ is referred as the length of w. The set of all reduced words of the element w is denoted by $R(w)$.

Example 2.1. *The set of reduced words for $w = s_2 s_1 s_2 s_0 s_1 s_0$ is given by*

$$R(w) = \{210210, 212010, 121010, 120101, 102101\}.$$

We say that a reduced word $a_1 a_2 \ldots a_\ell$ is *unimodal* if there exists an index v, such that

$$a_1 > a_2 > \cdots > a_v < a_{v+1} < \cdots < a_\ell.$$

Consider a reduced word $\mathbf{a} = a_1 a_2 \ldots a_{\ell(w)}$ of a Coxeter group element w. A *unimodal factorization* of $\mathbf{a}$ is a factorization $\mathbf{A} = (a_1 \ldots a_{\ell_1})(a_{\ell_1+1} \ldots a_{\ell_2}) \cdots (a_{\ell_r+1} \ldots a_L)$ such that each factor $(a_{\ell_i+1} \ldots a_{\ell_{i+1}})$ is unimodal. Factors can be empty.

For a fixed Coxeter group element w, consider all reduced words $R(w)$, and denote the set of all unimodal factorizations for reduced words in $R(w)$ as $U(w)$. Given a factorization $\mathbf{A} \in U(w)$, define the *weight* of a

factorization $\mathrm{wt}(\mathbf{A})$ to be the vector consisting of the number of elements in each factor. Denote by $\mathrm{nz}(\mathbf{A})$ the number of non-empty factors of $\mathbf{A}$.

Example 2.2. *For the factorization* $\mathbf{A} = (2102)()(10) \in U(s_2s_1s_2s_0s_1s_0)$*, we have* $\mathrm{wt}(\mathbf{A}) = (4, 0, 2)$ *and* $\mathrm{nz}(\mathbf{A}) = 2$.

Following [**BH95**,**FK96**,**Lam95**], the *type C Stanley symmetric function* associated to $w \in W_C$ is defined as

$$F_w^C(\mathbf{x}) = \sum_{\mathbf{A} \in U(w)} 2^{\mathrm{nz}(\mathbf{A})} \mathbf{x}^{\mathrm{wt}(\mathbf{A})}. \tag{2.1}$$

Here $\mathbf{x} = (x_1, x_2, x_3, \ldots)$ and $\mathbf{x}^{\mathbf{v}} = x_1^{v_1} x_2^{v_2} x_3^{v_3} \cdots$. It is not obvious from the definition why the above functions are symmetric. We refer reader to [**BHRY14**], where this fact follows easily from an alternative definition.

Type B Stanley symmetric functions are also labeled by $w \in W_C$ (as the type B and C Coxeter groups coincide) and differ from $F_w^C(w)$ by an overall factor $2^{-o(w)}$

$$F_w^B(\mathbf{x}) = 2^{-o(w)} F_w^C(\mathbf{x}),$$

where $o(w)$ is the number of zeroes in a reduced word for w. Loosely speaking, our combinatorial interpretation in the type C case respects this power of 2 – that is, we will get a valid combinatorial interpretation in the type B case by dividing by $2^{o(w)}$.

2.2.2. Type *A* crystal of words. Crystal bases [**Kas94**] play an important role in many areas of mathematics. For example, they make it possible to analyze representation theoretic questions using combinatorial tools. Here we only review the crystal of words in type A_n and refer the reader for more background on crystals to [**BS17**].

Consider the set of words $\mathcal{B}_n^h$ of length h in the alphabet $\{1, 2, \ldots, n+1\}$. We impose a crystal structure on $\mathcal{B}_n^h$ by defining lowering operators f_i and raising operators e_i for $1 \leqslant i \leqslant n$ and a weight function. The weight of $\mathbf{b} \in \mathcal{B}_n^h$ is the tuple $\mathrm{wt}(\mathbf{b}) = (a_1, \ldots, a_{n+1})$, where a_i is the number of letters i in $\mathbf{b}$. The crystal operators f_i and e_i only depend on the letters i and $i+1$ in $\mathbf{b}$. Consider the subword $\mathbf{b}^{\{i,i+1\}}$ of $\mathbf{b}$ consisting only of the letters i and $i+1$. Successively bracket any adjacent pairs $(i+1)i$ and remove these pairs from the word. The resulting word is of the form $i^a(i+1)^b$ with $a, b \geqslant 0$. Then f_i changes this subword within $\mathbf{b}$ to $i^{a-1}(i+1)^{b+1}$ if $a > 0$ leaving all other letters unchanged and otherwise annihilates $\mathbf{b}$. The operator e_i

changes this subword within $\mathbf{b}$ to $i^{a+1}(i+1)^{b-1}$ if $b > 0$ leaving all other letters unchanged and otherwise annihilates $\mathbf{b}$.

We call an element $\mathbf{b} \in \mathcal{B}_n^h$ *highest weight* if $e_i(\mathbf{b}) = \mathbf{0}$ for all $1 \leqslant i \leqslant n$ (meaning that all e_i annihilate $\mathbf{b}$).

THEOREM 2.3. *[KN94] A word $\mathbf{b} = b_1 \dots b_h \in \mathcal{B}_n^h$ is highest weight if and only if it is a Yamanouchi word. That is, for any index k with $1 \leqslant k \leqslant h$ the weight of a subword $b_k b_{k+1} \dots b_h$ is a partition.*

EXAMPLE 2.4. *The word* 85744234654333222211111 *is highest weight.*

Two crystals $\mathcal{B}$ and C are said to be *isomorphic* if there exists a bijective map $\Phi\colon \mathcal{B} \to C$ that preserves the weight function and commutes with the crystal operators e_i and f_i. A *connected component* X of a crystal is a set of elements where for any two $\mathbf{b}, \mathbf{c} \in X$ one can reach $\mathbf{c}$ from $\mathbf{b}$ by applying a sequence of f_i and e_i.

THEOREM 2.5. *[KN94] Each connected component of $\mathcal{B}_n^h$ has a unique highest weight element. Furthermore, if $\mathbf{b}, \mathbf{c} \in \mathcal{B}_n^h$ are highest weight elements such that* $\mathrm{wt}(\mathbf{b}) = \mathrm{wt}(\mathbf{c})$*, then the connected components generated by $\mathbf{b}$ and $\mathbf{c}$ are isomorphic.*

We denote a connected component with a highest weight element of highest weight λ by $\mathcal{B}_\lambda$. The *character* of the crystal $\mathcal{B}$ is defined to be a polynomial in the variables $\mathbf{x} = (x_1, x_2, \dots, x_{n+1})$

$$\chi_{\mathcal{B}}(\mathbf{x}) = \sum_{\mathbf{b} \in \mathcal{B}} \mathbf{x}^{\mathrm{wt}(\mathbf{b})}.$$

THEOREM 2.6 ([KN94]). *The character of $\mathcal{B}_\lambda$ is equal to the Schur polynomial $s_\lambda(\mathbf{x})$ (or Schur function in the limit $n \to \infty$).*

2.3. Crystal isomorphism

In this section, we combine a slight generalization of the Kraśkiewicz insertion, reviewed in Section 2.3.1, and Haiman's mixed insertion, reviewed in Section 2.3.2, to provide an isomorphism of crystals between the crystal of words $\mathcal{B}^h$ and certain sets of primed tableaux. Our main result of this section is stated in Theorem 2.18, which asserts that the recording tableaux under the mixed insertion is constant on connected components of $\mathcal{B}^h$.

2.3.1. Kraśkiewicz insertion. In this section, we describe Kraśkiewicz insertion. To do so, we first need to define the *Edelman–Greene insertion* [EG87]. It is defined for a word $\mathbf{w} = w_1 \dots w_\ell$ and a letter k

such that the concatenation $w_1 \dots w_\ell k$ is an A-type reduced word. The Edelman–Green insertion of a letter k into an *increasing* word $\mathbf{w} = w_1 \dots w_\ell$, denoted by $\mathbf{w} \leftsquigarrow k$, is constructed as follows:

(1) If $w_\ell < k$, then $\mathbf{w} \leftsquigarrow k = \mathbf{w}'$, where $\mathbf{w}' = w_1 w_2 \dots w_\ell\, k$.

(2) If $k > 0$ and $k\,k+1 = w_i\, w_{i+1}$ for some $1 \leqslant i < \ell$, then $\mathbf{w} \leftsquigarrow k = k+1 \leftsquigarrow \mathbf{w}$.

(3) Else let w_i be the leftmost letter in $\mathbf{w}$ such that $w_i > k$. Then $\mathbf{w} \leftsquigarrow k = w_i \leftsquigarrow \mathbf{w}'$, where $\mathbf{w}' = w_1 \dots w_{i-1}\, k\, w_{i+1} \dots w_\ell$.

In the cases above, when $\mathbf{w} \leftsquigarrow k = k' \leftsquigarrow \mathbf{w}'$, the symbol $k' \leftsquigarrow \mathbf{w}'$ indicates a word $\mathbf{w}'$ together with a "bumped" letter k'.

Next we consider a reduced unimodal word $\mathbf{a} = a_1 a_2 \dots a_\ell$ with $a_1 > a_2 > \cdots > a_v < a_{v+1} < \cdots < a_\ell$. The *Kraśkiewicz row insertion* [**Kra89**, **Kra95**] is defined for a unimodal word $\mathbf{a}$ and a letter k such that the concatenation $a_1 a_2 \dots a_\ell k$ is a C-type reduced word. The Kraśkiewicz row insertion of k into $\mathbf{a}$ (denoted similarly as $\mathbf{a} \leftsquigarrow k$), is performed as follows:

(1) If $k = 0$ and there is a subword 101 in $\mathbf{a}$, then $\mathbf{a} \leftsquigarrow 0 = 0 \leftsquigarrow \mathbf{a}$.

(2) If $k \neq 0$ or there is no subword 101 in $\mathbf{a}$, denote the decreasing part $a_1 \dots a_v$ as $\mathbf{d}$ and the increasing part $a_{v+1} \dots a_\ell$ as $\mathbf{g}$. Perform the Edelman-Greene insertion of k into $\mathbf{g}$.

   (a) If $a_\ell < k$, then $\mathbf{g} \leftsquigarrow k = a_{v+1} \dots a_\ell k =: \mathbf{g}'$ and $\mathbf{a} \leftsquigarrow k = \mathbf{d}\mathbf{g} \leftsquigarrow k = \mathbf{d}\,\mathbf{g}' =: \mathbf{a}'$.

   (b) If there is a bumped letter and $\mathbf{g} \leftsquigarrow k = k' \leftsquigarrow \mathbf{g}'$, negate all the letters in $\mathbf{d}$ (call the resulting word $-\mathbf{d}$) and perform the Edelman-Greene insertion $-\mathbf{d} \leftsquigarrow -k'$. Note that there will always be a bumped letter, and so $-\mathbf{d} \leftsquigarrow -k' = -k'' \leftsquigarrow -\mathbf{d}'$ for some decreasing word $\mathbf{d}'$. The result of the Kraśkiewicz insertion is: $\mathbf{a} \leftsquigarrow k = \mathbf{d}[\mathbf{g} \leftsquigarrow k] = \mathbf{d}[k' \leftsquigarrow \mathbf{g}'] = -[-\mathbf{d} \leftsquigarrow -k']\,\mathbf{g}' = [k'' \leftsquigarrow \mathbf{d}']\mathbf{g}' = k'' \leftsquigarrow \mathbf{a}'$, where $\mathbf{a}' := \mathbf{d}'\mathbf{g}'$.

Example 2.7.

$$31012 \leftsquigarrow 0 = 0 \leftsquigarrow 31012, \quad 3012 \leftsquigarrow 0 = 0 \leftsquigarrow 3102,$$

$$31012 \leftsquigarrow 1 = 1 \leftsquigarrow 32012, \quad 31012 \leftsquigarrow 3 = 310123.$$

The insertion is constructed to "commute" a unimodal word with a letter: If $\mathbf{a} \leftsquigarrow k = k' \leftsquigarrow \mathbf{a}'$, the two elements of the type C Coxeter group corresponding to concatenated words $\mathbf{a}\,k$ and $k'\mathbf{a}'$ are the same.

The type C Stanley symmetric functions (2.1) are defined in terms of unimodal factorizations. To put the formula on a completely combinatorial footing, we need to treat the powers of 2 by introducing signed

unimodal factorizations. A *signed unimodal factorization* of $w \in W_C$ is a unimodal factorization $\mathbf{A}$ of w, in which every non-empty factor is assigned either a $+$ or $-$ sign. Denote the set of all signed unimodal factorizations of w by $U^{\pm}(w)$.

For a signed unimodal factorization $\mathbf{A} \in U^{\pm}(w)$, define $\mathrm{wt}(\mathbf{A})$ to be the vector with i-th coordinate equal to the number of letters in the i-th factor of $\mathbf{A}$. Notice from (2.1) that

$$F_w^C(\mathbf{x}) = \sum_{\mathbf{A} \in U^{\pm}(w)} \mathbf{x}^{\mathrm{wt}(\mathbf{A})}. \tag{2.2}$$

We will use the Kraśkiewicz insertion to construct a map between signed unimodal factorizations of a Coxeter group element w and pairs of certain types of tableaux $(\mathbf{P}, \mathbf{T})$. We define these types of tableaux next.

A *shifted diagram* $\mathcal{S}(\lambda)$ associated to a partition λ with distinct parts is the set of boxes in positions $\{(i, j) \mid 1 \leqslant i \leqslant \ell(\lambda),\ i \leqslant j \leqslant \lambda_i + i - 1\}$. Here, we use English notation, where the box $(1, 1)$ is always top-left.

Let X_n° be an ordered alphabet of n letters $X_n^\circ = \{0 < 1 < 2 < \cdots < n-1\}$, and let X_n' be an ordered alphabet of n letters together with their primed counterparts as $X_n' = \{1' < 1 < 2' < 2 < \cdots < n' < n\}$.

Let λ be a partition with distinct parts. A *unimodal tableau* $\mathbf{P}$ of shape λ on n letters is a filling of $\mathcal{S}(\lambda)$ with letters from the alphabet X_n° such that the word P_i obtained by reading the ith row from the top of $\mathbf{P}$ from left to right, is a unimodal word, and P_i is the longest unimodal subword in the concatenated word $P_{i+1}P_i$ [**BHRY14**] (cf. also with decomposition tableaux [**Ser10**, **Cho13**]). The *reading word* of a unimodal tableau $\mathbf{P}$ is given by $\pi_{\mathbf{P}} = P_\ell P_{\ell-1} \dots P_1$. A unimodal tableau is called *reduced* if $\pi_{\mathbf{P}}$ is a type C reduced word corresponding to the Coxeter group element $w_{\mathbf{P}}$. Given a fixed Coxeter group element w, denote the set of reduced unimodal tableaux $\mathbf{P}$ of shape λ with $w_{\mathbf{P}} = w$ as $\mathcal{UT}_w(\lambda)$.

A *signed primed tableau* $\mathbf{T}$ of shape λ on n letters (cf. semistandard Q-tableau [**Lam95**]) is a filling of $\mathcal{S}(\lambda)$ with letters from the alphabet X_n' such that:

(1) The entries are weakly increasing along each column and each row of $\mathbf{T}$.

(2) Each row contains at most one i' for every $i = 1, \dots, n$.

(3) Each column contains at most one i for every $i = 1, \dots, n$.

The reason for using the word "signed" in the name is to distinguish the set of primed tableaux above from the "unsigned" version described later in the chapter.

Denote the set of signed primed tableaux of shape λ by $\mathcal{PT}^{\pm}(\lambda)$. Given an element $\mathbf{T} \in \mathcal{PT}^{\pm}(\lambda)$, define the weight of the tableau wt($\mathbf{T}$) as the vector with i-th coordinate equal to the total number of letters in $\mathbf{T}$ that are either i or i'.

EXAMPLE 2.8. $\left(\begin{array}{|c|c|c|c|c|} \hline 4 & 3 & 2 & 0 & 1 \\ \hline \end{array} \atop \begin{array}{|c|c|c|} \hline 2 & 1 & 2 \\ \hline \end{array} \atop \begin{array}{|c|} \hline 0 \\ \hline \end{array} , \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 2' & 3' & 3 \\ \hline \end{array} \atop \begin{array}{|c|c|c|} \hline 2' & 2 & 3' \\ \hline \end{array} \atop \begin{array}{|c|} \hline 4 \\ \hline \end{array} \right)$ *is a pair consisting of a unimodal tableau and a signed primed tableau both of shape* $(5, 3, 1)$.

For a reduced unimodal tableau $\mathbf{P}$ with rows $P_\ell, P_{\ell-1}, \ldots, P_1$, the Kraśkiewicz insertion of a letter k into tableau $\mathbf{P}$ (denoted again by $\mathbf{P} \leftsquigarrow k$) is performed as follows:

(1) Perform Kraśkiewicz insertion of the letter k into the unimodal word P_1. If there is no bumped letter and $P_1 \leftsquigarrow k = P'_1$, the algorithm terminates and the new tableau $\mathbf{P}'$ consists of rows $P_\ell, P_{\ell-1}, \ldots, P_2, P'_1$. If there is a bumped letter and $P_1 \leftsquigarrow k = k' \leftsquigarrow P'_1$, continue the algorithm by inserting k' into the unimodal word P_2.

(2) Repeat the previous step for the rows of $\mathbf{P}$ until either the algorithm terminates, in which case the new tableau $\mathbf{P}'$ consists of rows $P_\ell, \ldots, P_{s+1}, P'_s, \ldots, P'_1$, or, the insertion continues until we bump a letter k_e from P_ℓ, in which case we then put k_e on a new row of the shifted shape of $\mathbf{P}'$, so that the resulting tableau $\mathbf{P}'$ consists of rows $k_e, P'_\ell, \ldots, P'_1$.

EXAMPLE 2.9. $\begin{array}{|c|c|c|c|c|} \hline 4 & 3 & 2 & 0 & 1 \\ \hline \end{array} \atop \begin{array}{|c|c|c|} \hline 2 & 1 & 2 \\ \hline \end{array} \atop \begin{array}{|c|} \hline 0 \\ \hline \end{array} \leftsquigarrow 0 = \begin{array}{|c|c|c|c|c|} \hline 4 & 3 & 2 & 1 & 0 \\ \hline \end{array} \atop \begin{array}{|c|c|c|} \hline 2 & 1 & 0 \\ \hline \end{array} \atop \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array}$.

LEMMA 2.10. *[**Kra89**] Let* $\mathbf{P}$ *be a reduced unimodal tableau with reading word* $\pi_{\mathbf{P}}$ *for an element* $w \in W_C$. *Let* k *be a letter such that* $\pi_{\mathbf{P}}k$ *is a reduced word. Then the tableau* $\mathbf{P}' = \mathbf{P} \leftsquigarrow k$ *is a reduced unimodal tableau, for which the reading word* $\pi_{\mathbf{P}'}$ *is a reduced word for* ws_k.

LEMMA 2.11. *[**Lam95**, Lemma 3.17] Let* $\mathbf{P}$ *be a unimodal tableau, and* $\mathbf{a}$ *a unimodal word such that* $\pi_{\mathbf{P}}\mathbf{a}$ *is reduced. Let* $(x_1, y_1), \ldots, (x_r, y_r)$ *be the (ordered) list of boxes added when* $\mathbf{P} \leftsquigarrow \mathbf{a}$ *is computed. Then there exists an index* v, *such that* $x_1 < \cdots < x_v \geqslant \cdots \geqslant x_r$ *and* $y_1 \geqslant \cdots \geqslant y_v < \cdots < y_r$.

Let $\mathbf{A} \in U^{\pm}(w)$ be a signed unimodal factorization with unimodal factors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$. We recursively construct a sequence $(\emptyset, \emptyset) = (\mathbf{P}_0, \mathbf{T}_0), (\mathbf{P}_1, \mathbf{T}_1), \ldots, (\mathbf{P}_n, \mathbf{T}_n) = (\mathbf{P}, \mathbf{T})$ of tableaux, where $\mathbf{P}_s \in \mathcal{UT}_{(\mathbf{a}_1\mathbf{a}_2\ldots\mathbf{a}_s)}(\lambda^{(s)})$ and $\mathbf{T}_s \in \mathcal{PT}^{\pm}(\lambda^{(s)})$ are tableaux of the same shifted shape $\lambda^{(s)}$.

To obtain the *insertion tableau* $\mathbf{P}_s$, insert the letters of $\mathbf{a}_s$ one by one from left to right, into $\mathbf{P}_{s-1}$. Denote the shifted shape of $\mathbf{P}_s$ by $\lambda^{(s)}$. Enumerate the boxes in the skew shape $\lambda^{(s)}/\lambda^{(s-1)}$ in the order they appear in $\mathbf{P}_s$. Let these boxes be $(x_1, y_1), \ldots, (x_{\ell_s}, y_{\ell_s})$.

Let v be the index that is guaranteed to exist by Lemma 2.11 when we compute $\mathbf{P_{s-1}} \leftsquigarrow \mathbf{a_s}$. The *recording tableau* $\mathbf{T}_s$ is a primed tableau obtained from $\mathbf{T}_{s-1}$ by adding the boxes $(x_1, y_1), \ldots, (x_{v-1}, y_{v-1})$, each filled with the letter s', and the boxes $(x_{v+1}, y_{v+1}), \ldots, (x_{\ell_s}, y_{\ell_s})$, each filled with the letter s. The special case is the box (x_v, y_v), which could contain either s' or s. The letter is determined by the sign of the factor $\mathbf{a}_s$: If the sign is $-$, the box is filled with the letter s', and if the sign is $+$, the box is filled with the letter s. We call the resulting map the *primed Kraśkiewicz map* KR′.

Example 2.12. *Given a signed unimodal factorization* $\mathbf{A} = (-0)(+212)(-43201)$*, the sequence of tableaux is*

$$(\emptyset, \emptyset), \quad (\boxed{0}, \boxed{1'}), \quad \left(\begin{array}{|c|c|c|}\hline 2&1&2\\\hline\end{array}\;\begin{array}{|c|}\hline 0\\\hline\end{array}\;,\;\begin{array}{|c|c|c|}\hline 1'&2'&2\\\hline\end{array}\;\begin{array}{|c|}\hline 2\\\hline\end{array}\right), \quad \left(\begin{array}{|c|c|c|c|c|}\hline 4&3&2&0&1\\\hline\end{array}\;\begin{array}{|c|c|c|}\hline 2&1&2\\\hline\end{array}\;\begin{array}{|c|}\hline 0\\\hline\end{array}\;,\;\begin{array}{|c|c|c|c|c|}\hline 1'&2'&2&3'&3\\\hline\end{array}\;\begin{array}{|c|c|}\hline 2&3'\\\hline\end{array}\;\begin{array}{|c|}\hline 3'\\\hline\end{array}\right).$$

If the recording tableau is constructed, instead, by simply labeling its boxes with $1, 2, 3, \ldots$ in the order these boxes appear in the insertion tableau, we recover the original Kraśkiewicz map [**Kra89**,**Kra95**], which is a bijection

$$\mathrm{KR}\colon R(w) \to \bigcup_{\lambda} [\mathcal{UT}_w(\lambda) \times \mathcal{ST}(\lambda)],$$

where $\mathcal{ST}(\lambda)$ is the set of *standard shifted tableau* of shape λ, i.e., the set of fillings of $\mathcal{S}(\lambda)$ with letters $1, 2, \ldots, |\lambda|$ such that each letter appears exactly once, each row filling is increasing, and each column filling is increasing.

Theorem 2.13. *The primed Kraśkiewicz map is a bijection*

$$\mathrm{KR}'\colon U^{\pm}(w) \to \bigcup_{\lambda} [\mathcal{UT}_w(\lambda) \times \mathcal{PT}^{\pm}(\lambda)].$$

Proof. First we show that the map is well-defined: Let $\mathbf{A} \in U^{\pm}(w)$ such that $\mathrm{KR}'(A) = (\mathbf{P}, \mathbf{Q})$. The fact that $\mathbf{P}$ is a unimodal tableau follows from the fact that KR is well-defined. On the other hand, $\mathbf{Q}$ satisfies Condition (1) in the definition of signed primed tableaux since its entries are weakly increasing with respect to the order the associated boxes are added to $\mathbf{P}$. Now fix an s and consider the insertion $\mathbf{P_{s-1}} \leftsquigarrow \mathbf{a_s}$. Refer to the set-up in Lemma 2.11. Then, $y_1 < \cdots < y_v$ implies there is at most one s' in each row and $y_v \geqslant \cdots \geqslant y_{\ell_s}$

implies there is at most one s in each column, so Conditions (2) and (3) of the definition have been verified, implying that indeed $\mathbf{Q}$ is a signed primed tableau.

Now suppose $(\mathbf{P}, \mathbf{Q}) \in \bigcup_\lambda [\mathcal{UT}_w(\lambda) \times \mathcal{PT}^{\pm}(\lambda)]$. The ordering of the alphabet X' induces a partial order on the set of boxes of $\mathbf{Q}$. Refine this ordering as follows: Among boxes containing an s', box b is greater than box c if box b lies below box c. Among boxes containing an s, box b is greater than box c if box b lies to the right of box c. Let the standard shifted tableau induced by the resulting total order be denoted $\mathbf{Q}^*$.

Let $w = \mathrm{KR}^{-1}(\mathbf{P}, \mathbf{Q}^*)$. Divide w into factors, where the size of the s-th factor is equal to the s-th entry in $\mathrm{wt}(\mathbf{Q})$. Let $\mathbf{A} = \mathbf{a}_1 \dots \mathbf{a}_n$ be the resulting factorization, where the sign of $\mathbf{a}_s$ is determined as follows: Consider the lowest leftmost box in $\mathbf{Q}$ that contains an s or s' (such a box must exist if $\mathbf{a}_s \neq \emptyset$). If this box contains an s give $\mathbf{a}_s$ a positive sign, and otherwise a negative sign. Let $b_1, \dots, b_{|\mathbf{a}_s|}$ denote the boxes of $\mathbf{Q}^*$ corresponding to $\mathbf{a}_s$ under KR^{-1}. The construction of $\mathbf{Q}^*$ and the fact that $\mathbf{Q}$ is a primed shifted tableau imply that the coordinates of these boxes satisfy the hypothesis of Lemma 2.11. Since these are exactly the boxes that appear when we compute $\mathbf{P_{s-1}} \leftsquigarrow \mathbf{a}_s$, Lemma 2.11 implies that $\mathbf{a}_s$ is unimodal. It follows that $\mathbf{A}$ is a signed unimodal factorization mapping to $(\mathbf{P}, \mathbf{Q})$ under KR'. It is not hard to see $\mathbf{A}$ is unique. □

Theorem 2.13 and Equation (2.2) imply the following relation:

$$F_w^C(\mathbf{x}) = \sum_\lambda \left|\mathcal{UT}_w(\lambda)\right| \sum_{\mathbf{T} \in \mathcal{PT}^{\pm}(\lambda)} \mathbf{x}^{\mathrm{wt}(\mathbf{T})}. \tag{2.3}$$

Remark 2.14. *The sum $\sum_{\mathbf{T} \in \mathcal{PT}^{\pm}(\lambda)} \mathbf{x}^{\mathrm{wt}(\mathbf{T})}$ is also known as the Q-Schur function. The expansion (2.3), with a slightly different interpretation of Q-Schur function, was shown in [**BH95**].*

At this point, we are halfway there to expand $F_w^C(\mathbf{x})$ in terms of Schur functions. In the next section we introduce a crystal structure on the set $\mathcal{PT}(\lambda)$ of unsigned primed tableaux.

2.3.2. Mixed insertion. Set $\mathcal{B}^h = \mathcal{B}^h_\infty$. Similar to the well-known RSK-algorithm, mixed insertion [**Hai89**] gives a bijection between $\mathcal{B}^h$ and the set of pairs of tableaux $(\mathbf{T}, \mathbf{Q})$, but in this case $\mathbf{T}$ is an (unsigned) primed tableau of shape λ and $\mathbf{Q}$ is a standard shifted tableau of the same shape.

An *(unsigned) primed tableau* of shape λ (cf. semistandard P-tableau [**Lam95**] or semistandard marked shifted tableau [**Cho13**]) is a signed primed tableau $\mathbf{T}$ of shape λ with only unprimed elements on the main diagonal. Denote the set of primed tableaux of shape λ by $\mathcal{PT}(\lambda)$. The weight function $\mathrm{wt}(\mathbf{T})$ of $\mathbf{T} \in \mathcal{PT}(\lambda)$ is inherited from the weight function of signed primed tableaux, that is, it is the vector with i-th coordinate

equal to the number of letters i' and i in $\mathbf{T}$. We can simplify (2.3) as

$$F_w^C(\mathbf{x}) = \sum_{\lambda} 2^{\ell(\lambda)} \left|\mathcal{UT}_w(\lambda)\right| \sum_{\mathbf{T}\in\mathcal{PT}(\lambda)} \mathbf{x}^{\mathrm{wt}(\mathbf{T})}. \tag{2.4}$$

REMARK 2.15. *The sum $\sum_{\mathbf{T}\in\mathcal{PT}(\lambda)} \mathbf{x}^{\mathrm{wt}(\mathbf{T})}$ is also known as a P-Schur function.*

Given a word $b_1 b_2 \dots b_h$ in the alphabet $X = \{1 < 2 < 3 < \cdots\}$, we recursively construct a sequence of tableaux $(\emptyset,\emptyset) = (\mathbf{T}_0,\mathbf{Q}_0), (\mathbf{T}_1,\mathbf{Q}_1),\dots,(\mathbf{T}_h,\mathbf{Q}_h) = (\mathbf{T},\mathbf{Q})$, where $\mathbf{T}_s \in \mathcal{PT}(\lambda^{(s)})$ and $\mathbf{Q}_s \in \mathcal{ST}(\lambda^{(s)})$. To obtain the tableau $\mathbf{T}_s$, insert the letter b_s into $\mathbf{T}_{s-1}$ as follows. First, insert b_s into the first row of $\mathbf{T}_{s-1}$, bumping out the leftmost element y that is strictly greater than b_i in the alphabet $X' = \{1' < 1 < 2' < 2 < \cdots\}$.

(1) If y is not on the main diagonal and y is not primed, then insert it into the next row, bumping out the leftmost element that is strictly greater than y from that row.

(2) If y is not on the main diagonal and y is primed, then insert it into the next column to the right, bumping out the topmost element that is strictly greater than y from that column.

(3) If y is on the main diagonal, then it must be unprimed. Prime y and insert it into the column on the right, bumping out the topmost element that is strictly greater than y from that column.

If a bumped element exists, treat it as a new y and repeat the steps above – if the new y is unprimed, row-insert it into the row below its original cell, and if the new y is primed, column-insert it into the column to the right of its original cell.

The insertion process terminates either by placing a letter at the end of a row, bumping no new element, or forming a new row with the last bumped element. The shapes of $\mathbf{T}_{s-1}$ and $\mathbf{T}_s$ differ by one box. Add that box to $\mathbf{Q}_{s-1}$ with a letter s in it, to obtain the standard shifted tableau $\mathbf{Q}_s$.

EXAMPLE 2.16. *For a word* 332332123*, some of the tableaux in the sequence* $(\mathbf{T}_i,\mathbf{Q}_i)$ *are*

$$\left(\begin{array}{|c|c|}\hline 2 & 3'\\\hline 3\\\cline{1-1}\end{array}, \begin{array}{|c|c|}\hline 1 & 2\\\hline 3\\\cline{1-1}\end{array}\right), \quad \left(\begin{array}{|c|c|c|c|}\hline 2 & 2 & 3' & 3\\\hline 3 & 3\\\cline{1-2}\end{array}, \begin{array}{|c|c|c|c|}\hline 1 & 2 & 4 & 5\\\hline 3 & 6\\\cline{1-2}\end{array}\right), \quad \left(\begin{array}{|c|c|c|c|c|}\hline 1 & 2' & 2 & 3' & 3\\\hline 2 & 3' & 3\\\cline{1-3} 3\\\cline{1-1}\end{array}, \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 4 & 5 & 9\\\hline 3 & 6 & 8\\\cline{1-3} 7\\\cline{1-1}\end{array}\right).$$

THEOREM 2.17. ***[Hai89]*** *The construction above gives a bijection*

$$\mathrm{HM}\colon \mathcal{B}^h \to \bigcup_{\lambda\vdash h} [\mathcal{PT}(\lambda)\times\mathcal{ST}(\lambda)].$$

The bijection HM is called a *mixed insertion*. If $\mathrm{HM}(\mathbf{b}) = (\mathbf{T}, \mathbf{Q})$, denote $P_{\mathrm{HM}}(\mathbf{b}) = \mathbf{T}$ and $R_{\mathrm{HM}}(\mathbf{b}) = \mathbf{Q}$.

Just as for the RSK-algorithm, the mixed insertion has the property of preserving the recording tableau within each connected component of the crystal $\mathcal{B}^h$.

THEOREM 2.18. *The recording tableau $R_{\mathrm{HM}}(\cdot)$ is constant on each connected component of the crystal $\mathcal{B}^h$.*

Before we provide the proof of Theorem 2.18, we need to define one more insertion from [**Hai89**], which serves as a dual to the previously discussed mixed insertion.

We use the notion of *generalized permutations*. Similar to a regular permutation in two-line notation, a generalized permutation w consists of two lines $\binom{a_1 a_2 \cdots a_h}{b_1 b_2 \cdots b_h}$, which gives a correspondence between a_s and b_s, but there can be repeated letters now. We order the pairs (a_s, b_s) by making the top line weakly increasing $a_1 \leqslant \cdots \leqslant a_h$, and forcing $b_s \leqslant b_{s+1}$ whenever $a_s = a_{s+1}$. The inverse of a generalized permutation w^{-1} consists of pairs (b_s, a_s), ordered appropriately. Given a word $\mathbf{b} = b_1 \dots b_h$, it can be represented as a generalized permutation w by setting the first line of the permutation to be $1\ 2\ \dots h$ and the second line to be $b_1\ b_2\ \dots b_h$. Since the inverse of the generalized permutation w exists, it also defined $\mathbf{b}^{-1}$.

Now, let $w = \binom{a_1 a_2 \cdots a_h}{b_1 b_2 \cdots b_h}$ be a generalized permutation on the alphabet X, where the second line consists of distinct letters. We recursively construct a sequence of tableaux $(\emptyset, \emptyset) = (\mathbf{Q}_0, \mathbf{T}_0), (\mathbf{Q}_1, \mathbf{T}_1), \dots, (\mathbf{Q}_h, \mathbf{T}_h) = (\mathbf{Q}, \mathbf{T})$, where $\mathbf{Q}_s \in \mathcal{ST}(\lambda_s)$ and $\mathbf{T}_s \in \mathcal{PT}(\lambda_s)$. To obtain the tableau $\mathbf{Q}_s$, insert the letter b_s into $\mathbf{Q}_{s-1}$ as follows:

- Insert b_s into the first row of $\mathbf{Q}_{s-1}$, and insert each bumped element into the next row until either an element is inserted into an empty cell and the algorithm terminates, or an element b has been bumped from the diagonal. In the latter case, insert b into the column to its right and continue bumping by columns, until an empty cell is filled.
- The shapes of $\mathbf{Q}_{s-1}$ and $\mathbf{Q}_s$ differ by one box. Add that box to $\mathbf{T}_{s-1}$ with a letter a_s in it. Prime that letter if a diagonal element has been bumped in the process of inserting b_s into $\mathbf{Q}_{s-1}$.

The above insertion process is called a *Worley–Sagan insertion algorithm*. The insertion tableau $\mathbf{Q}$ will be denoted by $P_{\mathrm{WS}}(w)$ and the recording tableau $\mathbf{T}$ is denoted by $R_{\mathrm{WS}}(w)$.

THEOREM 2.19. *[**Hai89**, Theorem 6.10 and Corollary 6.3] Given $\mathbf{b} \in \mathcal{B}^h$, we have $R_{\mathrm{HM}}(\mathbf{b}) = P_{\mathrm{WS}}(\mathbf{b}^{-1})$.*

Next, we want to find out when the Worley–Sagan insertion tableau is preserved. Fortunately, other results from [**Hai89**] provide this description.

THEOREM 2.20. *[**Hai89**, Corollaries 5.8 and 6.3] If two words with distinct letters* $\mathbf{b}$ *and* $\mathbf{b}'$ *are related by a shifted Knuth transformation, then* $P_{\mathrm{WS}}(\mathbf{b}) = P_{\mathrm{WS}}(\mathbf{b}')$.

Here, a *shifted Knuth transformation* is an exchange of consecutive letters in one of the following forms:

(1) Knuth transformations: $cab \leftrightarrow acb$ or $bca \leftrightarrow bac$, where $a < b < c$,

(2) Worley–Sagan transformation: $xy \leftrightarrow yx$, where x and y are the first two letters of the word.

We are now ready to prove the theorem.

PROOF OF THEOREM 2.18. If $\mathbf{b}$ and $\mathbf{b}'$ are two words in the same connected component of $\mathcal{B}^h$, their RSK-recording tableaux $R_{\mathrm{RSK}}(\mathbf{b})$ and $R_{\mathrm{RSK}}(\mathbf{b}')$ are the same. Thus, $P_{\mathrm{RSK}}(\mathbf{b}^{-1})$ and $P_{\mathrm{RSK}}(\mathbf{b}'^{-1})$ are the same, and the second lines of $\mathbf{b}^{-1}$ and $\mathbf{b}'^{-1}$ are related by a sequence of Knuth transformations. This in turn means that $P_{\mathrm{WS}}(\mathbf{b}^{-1})$ and $P_{\mathrm{WS}}(\mathbf{b}'^{-1})$ are the same, and $R_{\mathrm{HM}}(\mathbf{b}) = R_{\mathrm{HM}}(\mathbf{b}')$ by Theorem 2.20. □

Let us fix a recording tableau $\mathbf{Q}_\lambda \in \mathcal{ST}(\lambda)$. Define a map $\Psi_\lambda \colon \mathcal{PT}(\lambda) \to \mathcal{B}^h$ as $\Psi_\lambda(\mathbf{T}) = \mathrm{HM}^{-1}(\mathbf{T}, \mathbf{Q}_\lambda)$. By Theorem 2.18, the set $\mathrm{Im}(\Psi_\lambda)$ consists of several connected components of $\mathcal{B}^h$. The map Ψ_λ can thus be taken as a crystal isomorphism, and we can define the crystal operators and weight function on $\mathcal{PT}(\lambda)$ as

$$e_i(\mathbf{T}) := (\Psi_\lambda^{-1} \circ e_i \circ \Psi_\lambda)(\mathbf{T}), \quad f_i(\mathbf{T}) := (\Psi_\lambda^{-1} \circ f_i \circ \Psi_\lambda)(\mathbf{T}), \quad \mathrm{wt}(\mathbf{T}) := (\mathrm{wt} \circ \Psi_\lambda)(\mathbf{T}). \tag{2.5}$$

Although it is not clear that the crystal operators constructed above are independent of the choice of $\mathbf{Q}_\lambda$, in the next section we will construct explicit crystal operators on the set $\mathcal{PT}(\lambda)$ that satisfy the relations above and do not depend on the choice of $\mathbf{Q}_\lambda$.

EXAMPLE 2.21. *For* $\mathbf{T} = \begin{array}{|c|c|c|c|c|}\hline 1 & 2' & 2 & 3' & 3 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3'} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{} \\ \cline{2-4} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \\ \cline{3-3} \end{array}$, *choose* $\mathbf{Q}_\lambda = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} & \multicolumn{1}{c}{} \\ \cline{2-4} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{9} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \\ \cline{3-3} \end{array}$. *Then* $\Psi_\lambda(\mathbf{T}) = 333332221$ *and* $e_1 \circ \Psi_\lambda(\mathbf{T}) = 333331221$. *Thus,*

$$e_1(\mathbf{T}) = (\Psi_\lambda^{-1} \circ e_1 \circ \Psi_\lambda)(\mathbf{T}) = \begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 2 & 3' & 3 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3'} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{} \\ \cline{2-4} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \\ \cline{3-3} \end{array}, \quad f_1(\mathbf{T}) = f_2(\mathbf{T}) = \mathbf{0}.$$

To summarize, we obtain a crystal isomorphism between the crystal $(\mathcal{PT}(\lambda), e_i, f_i, \mathrm{wt})$, denoted again by $\mathcal{PT}(\lambda)$, and a direct sum $\bigoplus_\mu \mathcal{B}_\mu^{\oplus h_{\lambda\mu}}$. We will provide a combinatorial description of the coefficients $h_{\lambda\mu}$

in the next section. This implies the relation on characters of the corresponding crystals $\chi_{\mathcal{PT}(\lambda)} = \sum_\mu h_{\lambda\mu} s_\mu$. Thus we can rewrite (2.4) one last time

$$F_w^C(\mathbf{x}) = \sum_\lambda 2^{\ell(\lambda)} \big|\mathcal{UT}_w(\lambda)\big| \sum_\mu h_{\lambda\mu} s_\mu = \sum_\mu \Big(\sum_\lambda 2^{\ell(\lambda)} \big|\mathcal{UT}_w(\lambda)\big|\, h_{\lambda\mu}\Big) s_\mu.$$

2.4. Explicit crystal operators on shifted primed tableaux

We consider the alphabet $X' = \{1' < 1 < 2' < 2 < 3' < \cdots\}$ of primed and unprimed letters. It is useful to think about the letter $(i+1)'$ as a number $i + 0.5$. Thus, we say that letters i and $(i+1)'$ differ by half a unit and letters i and $(i+1)$ differ by a whole unit.

Given an (unsigned) primed tableau $\mathbf{T}$, we construct the *reading word* $\mathrm{rw}(\mathbf{T})$ as follows:

(1) List all primed letters in the tableau, column by column, in decreasing order within each column, moving from the rightmost column to the left, and with all the primes removed (i.e. all letters are increased by half a unit). (Call this part of the word the *primed reading word*.)

(2) Then list all unprimed elements, row by row, in increasing order within each row, moving from the bottommost row to the top. (Call this part of the word the *unprimed reading word*.)

To find the letter on which the crystal operator f_i acts, apply the bracketing rule for letters i and $i+1$ within the reading word $\mathrm{rw}(\mathbf{T})$. If all letters i are bracketed in $\mathrm{rw}(\mathbf{T})$, then $f_i(\mathbf{T}) = \mathbf{0}$. Otherwise, the rightmost unbracketed letter i in $\mathrm{rw}(\mathbf{T})$ corresponds to an i or an i' in $\mathbf{T}$, which we call *bold unprimed* i or *bold primed* i respectively.

If the bold letter i is unprimed, denote the cell it is located in as x.

If the bold letter i is primed, we *conjugate* the tableau $\mathbf{T}$ first.

The *conjugate* of a primed tableau $\mathbf{T}$ is obtained by reflecting the tableau over the main diagonal, changing all primed entries k' to k and changing all unprimed elements k to $(k+1)'$ (i.e. increase the content of all boxes by half a unit). The main diagonal is now the North-East boundary of the tableau. Denote the resulting tableau as $\mathbf{T}^*$.

Under the transformation $\mathbf{T} \to \mathbf{T}^*$, the bold primed i is transformed into bold unprimed i. Denote the cell it is located in as x.

Given any cell z in a shifted primed tableau $\mathbf{T}$ (or conjugated tableau $\mathbf{T}^*$), denote by $c(z)$ the content of z. Denote by z_E the cell to the right of z, z_W the cell to its left, z_S the cell below, and z_N the cell above.

Denote by z^* the corresponding conjugated cell in $\mathbf{T}^*$ (or in $\mathbf{T}$). Now, consider the box x_E (in $\mathbf{T}$ or in $\mathbf{T}^*$) and notice that $c(x_E) \geqslant (i+1)'$.

Crystal operator f_i on primed tableaux:

(1) If $c(x_E) = (i+1)'$, the box x must lie outside of the main diagonal and the box right below x_E cannot have content equal to $(i+1)'$. Change $c(x)$ to $(i+1)'$ and change $c(x_E)$ to $(i+1)$ (i.e. increase the content of x and x_E by half a unit).

(2) If $c(x_E) \neq (i+1)'$ or x_E is empty, then there is a maximal connected ribbon (expanding in South and West directions) with the following properties:

   (a) The North-Eastern most box of the ribbon (the tail of the ribbon) is x.

   (b) The contents of all boxes within a ribbon besides the tail are either $(i+1)'$ or $(i+1)$.

   Denote the South-Western most box of the ribbon (the head) as x_H.

   (a) If $x_H = x$, change $c(x)$ to $(i+1)$ (i.e. increase the content of x by a whole unit).

   (b) If $x_H \neq x$ and x_H is on the main diagonal (in case of a tableau $\mathbf{T}$), change $c(x)$ to $(i+1)'$ (i.e. increase the content of x by half a unit).

   (c) Otherwise, the content $c(x_H)$ must be $(i+1)'$ due to the bracketing rule. We change $c(x)$ to $(i+1)'$ and change $c(x_H)$ to $(i+1)$ (i.e. increase the content of x and x_H by half a unit).

In the case when the bold i in $\mathbf{T}$ is unprimed, we apply the above crystal operator rules to $\mathbf{T}$ to find $f_i(\mathbf{T})$

Example 2.22. *In the following examples, we mark the bold i (if it exists):*

$$f_2\Big(\begin{array}{|c|c|c|c|}\hline 1 & 2' & 2 & 3' \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3'} & \multicolumn{1}{c}{3} \\ \cline{2-4}\end{array}\Big) = \mathbf{0}, \quad f_2\Big(\begin{array}{|c|c|c|c|}\hline 1 & 2' & \mathbf{2} & 3' \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3'} & \multicolumn{1}{c}{4} \\ \cline{2-4}\end{array}\Big) = \begin{array}{|c|c|c|c|}\hline 1 & 2' & 3' & 3 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3'} & \multicolumn{1}{c}{4} \\ \cline{2-4}\end{array}, \quad f_2\Big(\begin{array}{|c|c|c|c|}\hline 1 & 1 & 2 & \mathbf{2} \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4'} & \multicolumn{1}{c}{4} \\ \cline{2-4}\end{array}\Big) = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 2 & 3 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4'} & \multicolumn{1}{c}{4} \\ \cline{2-4}\end{array},$$

$$f_2\Big(\begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 2' & \mathbf{2} & 3 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3'} & \multicolumn{1}{c}{} \\ \cline{2-4} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{} \\ \cline{3-4}\end{array}\Big) = \begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 2' & 3' & 3 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3'} & \multicolumn{1}{c}{} \\ \cline{2-4} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{} \\ \cline{3-4}\end{array}, \quad f_2\Big(\begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 1 & \mathbf{2} & 3 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3'} & \multicolumn{1}{c}{} \\ \cline{2-4} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4'} & \multicolumn{1}{c}{} \\ \cline{3-4}\end{array}\Big) = \begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 1 & 3' & 3 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{} \\ \cline{2-4} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4'} & \multicolumn{1}{c}{} \\ \cline{3-4}\end{array}.$$

In the case when the bold i is primed in $\mathbf{T}$, we first conjugate $\mathbf{T}$ and then apply the above crystal operator rules on $\mathbf{T}^*$, before reversing the conjugation. Note that Case 2b is impossible for $\mathbf{T}^*$, since the main diagonal is now on the North-East.

EXAMPLE 2.23.

$$\textit{Let } \mathbf{T} = \begin{array}{|c|c|c|c|}\hline 1 & \mathbf{2'} & 2 & 3 \\ \hline \multicolumn{1}{c}{} & 3 & 4' \\ \cline{2-3} \multicolumn{1}{c}{} & \multicolumn{1}{c|}{4} \\ \cline{2-2}\end{array}, \quad \textit{then } \mathbf{T}^* = \begin{array}{|c|c|c|}\hline 2' & & \\ \hline \mathbf{2} & 4' & \\ \hline 3' & 4 & 5' \\ \hline 4' & & \\ \cline{1-1}\end{array} \quad \textit{and } f_2(\mathbf{T}) = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3' & 3 \\ \hline \multicolumn{1}{c}{} & 3 & 4' \\ \cline{2-3} \multicolumn{1}{c}{} & \multicolumn{1}{c|}{4} \\ \cline{2-2}\end{array}.$$

THEOREM 2.24. *For any* $\mathbf{b} \in \mathcal{B}^h$ *with* $P_{\mathrm{HM}}(\mathbf{b}) = \mathbf{T}$ *and* $f_i(\mathbf{b}) \neq \mathbf{0}$*, the operator* f_i *defined on above satisfies*

$$P_{\mathrm{HM}}(f_i(\mathbf{b})) = f_i(\mathbf{T}).$$

Also, $f_i(\mathbf{b}) = \mathbf{0}$ *if and only if* $f_i(\mathbf{T}) = \mathbf{0}$.

The proof of Theorem 2.24 is quite technical and is relegated to Appendix 2.6. It implies that the explicit operators f_i in this section are indeed equal to those defined in (2.5) and that they are independent of the choice of $\mathbf{Q}_\lambda$. We also immediately obtain:

SECOND PROOF OF THEOREM 2.18. Given a word $\mathbf{b} = b_1 \dots b_h$, let $\mathbf{b}' = f_i(\mathbf{b}) = b'_1 \dots b'_h$, so that $b_m \neq b'_m$ for some m and $b_i = b'_i$ for any $i \neq m$. We show that $Q_{\mathrm{HM}}(\mathbf{b}) = Q_{\mathrm{HM}}(\mathbf{b}')$.

Denote $\mathbf{b}^{(s)} = b_1 \dots b_s$ and similarly $\mathbf{b}'^{(s)} = b'_1 \dots b'_s$. Due to the construction of the recording tableau Q_{HM}, it suffices to show that $P_{\mathrm{HM}}(\mathbf{b}^{(s)})$ and $P_{\mathrm{HM}}(\mathbf{b}'^{(s)})$ have the same shape for any $1 \leqslant s \leqslant h$.

If $s < m$, this is immediate. If $s \geqslant m$, note that $\mathbf{b}'^{(s)} = f_i(\mathbf{b}^{(s)})$. Using Theorem 2.24, one can see that $P_{\mathrm{HM}}(\mathbf{b}'^{(s)}) = P_{\mathrm{HM}}(f_i(\mathbf{b}^{(s)})) = f_i(P_{\mathrm{HM}}(\mathbf{b}^{(s)}))$ has the same shape as $P_{\mathrm{HM}}(\mathbf{b}^{(s)})$. □

The next step is to describe the raising operators $e_i(\mathbf{T})$. Consider the reading word $\mathrm{rw}(\mathbf{T})$ and apply the bracketing rule on the letters i and $i+1$. If all letters $i+1$ are bracketed in $\mathrm{rw}(\mathbf{T})$, then $e_i(\mathbf{T}) = \mathbf{0}$. Otherwise, the leftmost unbracketed letter $i+1$ in $\mathrm{rw}(\mathbf{T})$ corresponds to an $i+1$ or an $(i+1)'$ in $\mathbf{T}$, which we will call bold unprimed $i+1$ or bold primed $i+1$, respectively. If the bold $i+1$ is unprimed, denote the cell it is located in by y. If the bold $i+1$ is primed, conjugate $\mathbf{T}$ and denote the cell with the bold $i+1$ in $\mathbf{T}^*$ by y.

Crystal operator e_i on primed tableaux:

(1) If $c(y_W) = (i+1)'$, then change $c(y)$ to $(i+1)'$ and change $c(y_W)$ to i (i.e. decrease the content of y and y_W by half a unit).

(2) If $c(y_W) < (i+1)'$ or y_W is empty, then there is a maximal connected ribbon (expanding in North and East directions) with the following properties:

(a) The South-Western most box of the ribbon (the head of the ribbon) is y.

(b) The content of all boxes within a ribbon besides the tail is either i or $(i+1)'$.

Denote the North-Eastern most box of the ribbon (the tail) as y_T.

(a) If $y_T = y$, change $c(y)$ to i (i.e. decrease the content of y by a whole unit).

(b) If $y_T \neq y$ and y_T is on the main diagonal (in case of a conjugate tableau $\mathbf{T}^*$), then change $c(y)$ to $(i+1)'$ (i.e. decrease the content of y by half a unit).

(c) If $y_T \neq y$ and y_T is not on the diagonal, the content of y_T must be $(i+1)'$ and we change $c(y)$ to $(i+1)'$ and change $c(y_T)$ to i (i.e. decrease the content of y and y_T by half a unit).

When the bold $i+1$ is unprimed, $e_i(\mathbf{T})$ is obtained by applying the rules above to $\mathbf{T}$. When the bold $i+1$ is primed, we first conjugate $\mathbf{T}$, then apply the raising crystal operator rules on $\mathbf{T}^*$, and then reverse the conjugation.

Proposition 2.25.

$$e_i(\mathbf{b}) = \mathbf{0} \quad \textit{if and only if} \quad e_i(\mathbf{T}) = \mathbf{0}.$$

Proof. According to Lemma 2.32, the number of unbracketed letters i in $\mathbf{b}$ is equal to the number of unbracketed letters i in $\mathrm{rw}(\mathbf{T})$. Since the total number of both letters i and $j = i+1$ is the same in $\mathbf{b}$ and in $\mathrm{rw}(\mathbf{T})$, that also means that the number of unbracketed letters j in $\mathbf{b}$ is equal to the number of unbracketed letters j in $\mathrm{rw}(\mathbf{T})$. Thus, there are no unbracketed letters j in $\mathbf{b}$ if and only if there are no unbracketed letters j in $\mathbf{T}$. □

Theorem 2.26. *Given a primed tableau $\mathbf{T}$ with $f_i(\mathbf{T}) \neq \mathbf{0}$, for the operators e_i defined above we have the following relation:*

$$e_i(f_i(\mathbf{T})) = \mathbf{T}.$$

The proof of Theorem 2.26 is relegated to Appendix 2.7.

Corollary 2.27. *For any $\mathbf{b} \in \mathcal{B}^h$ with $\mathrm{HM}(\mathbf{b}) = (\mathbf{T}, \mathbf{Q})$, the operator e_i defined above satisfies*

$$\mathrm{HM}(e_i(\mathbf{b})) = (e_i(\mathbf{T}), \mathbf{Q}),$$

given the left-hand side is well-defined.

The consequence of Theorem 2.24, as discussed in Section 2.3.2, is a crystal isomorphism $\Psi_\lambda : \mathcal{PT}(\lambda) \to \bigoplus \mathcal{B}_\mu^{\oplus h_{\lambda\mu}}$. Now, to determine the nonnegative integer coefficients $h_{\lambda\mu}$, it is enough to count the highest weight elements in $\mathcal{PT}(\lambda)$ of given weight μ.

Proposition 2.28. *A primed tableau* $\mathbf{T} \in \mathcal{PT}(\lambda)$ *is a highest weight element if and only if its reading word* $\mathrm{rw}(\mathbf{T})$ *is a Yamanouchi word. That is, for any suffix of* $\mathrm{rw}(\mathbf{T})$*, its weight is a partition.*

Thus we define $h_{\lambda\mu}$ to be the number of primed tableaux $\mathbf{T}$ of shifted shape $\mathcal{S}(\lambda)$ and weight μ such that $\mathrm{rw}(\mathbf{T})$ is Yamanouchi.

Example 2.29. *Let* $\lambda = (5,3,2)$ *and* $\mu = (4,3,2,1)$*. There are three primed tableaux of shifted shape* $\mathcal{S}((5,3,2))$ *and weight* $(4,3,2,1)$ *with a Yamanouchi reading word, namely*

$$\begin{array}{|c|c|c|c|c|}\hline 1&1&1&1&2'\\\hline\end{array}\!\!\begin{array}{|c|c|c|}\hline 2&2&3'\\\hline\end{array}\!\!\begin{array}{|c|c|}\hline 3&4'\\\hline\end{array}\quad,\quad \begin{array}{|c|c|c|c|c|}\hline 1&1&1&1&3'\\\hline\end{array}\!\!\begin{array}{|c|c|c|}\hline 2&2&2\\\hline\end{array}\!\!\begin{array}{|c|c|}\hline 3&4'\\\hline\end{array}\quad \textit{and} \quad \begin{array}{|c|c|c|c|c|}\hline 1&1&1&1&4'\\\hline\end{array}\!\!\begin{array}{|c|c|c|}\hline 2&2&2\\\hline\end{array}\!\!\begin{array}{|c|c|}\hline 3&3\\\hline\end{array}\quad.$$

Therefore $h_{(5,3,2)(4,3,2,1)} = 3$.

We summarize our results for the type C Stanley symmetric functions as follows.

Corollary 2.30. *The expansion of* $F_w^C(\mathbf{x})$ *in terms of Schur symmetric functions is*

$$F_w^C(\mathbf{x}) = \sum_\lambda g_{w\lambda} s_\lambda(\mathbf{x}), \quad \textit{where} \quad g_{w\lambda} = \sum_\mu 2^{\ell(\mu)} \left|\mathcal{UT}_w(\mu)\right| h_{\mu\lambda}\,. \tag{2.6}$$

Replacing $\ell(\mu)$ by $\ell(\mu) - o(w)$ gives the Schur expansion of $F_w^B(\mathbf{x})$. Note that since any row of a unimodal tableau contains at most one zero, $\ell(\mu) - o(w)$ is nonnegative. Thus the given expansion makes sense combinatorially.

Example 2.31. *Consider the word* $w = 0101 = 1010$*. There is only one unimodal tableau corresponding to* w*, namely* $\mathbf{P} = \begin{array}{|c|c|c|}\hline 1&0&1\\\hline\end{array}\!\!\begin{array}{|c|}\hline 0\\\hline\end{array}$*, which belongs to* $\mathcal{UT}_{0101}(3,1)$*. Thus,* $g_{w\lambda} = 4h_{(3,1)\lambda}$*. There are only three possible highest weight primed tableaux of shape* $(3,1)$*, namely* $\begin{array}{|c|c|c|}\hline 1&1&1\\\hline\end{array}\!\!\begin{array}{|c|}\hline 2\\\hline\end{array}$, $\begin{array}{|c|c|c|}\hline 1&1&2'\\\hline\end{array}\!\!\begin{array}{|c|}\hline 2\\\hline\end{array}$ *and* $\begin{array}{|c|c|c|}\hline 1&1&3'\\\hline\end{array}\!\!\begin{array}{|c|}\hline 2\\\hline\end{array}$*, which implies that* $h_{(3,1)(3,1)} = h_{(3,1)(2,2)} = h_{(3,1)(2,1,1)} = 1$ *and* $h_{(3,1)\lambda} = 0$ *for other weights* λ*. The expansion of* $F_{0101}^C(\mathbf{x})$ *is thus*

$$F_{0101}^C = 4s_{(3,1)} + 4s_{(2,2)} + 4s_{(2,1,1)}.$$

2.5. Implementation in Sage

The class of shifted primed tableau $\mathcal{PT}(\lambda)$ has been implemented in Sage[1]. We give several examples of the basic functionality. The object of the shifted primed tableau can be initialized as a nested list, with string inputs for primed entries:

```
sage: t = ShiftedPrimedTableau([[1,1,"2p","3p"],[2,2,"3p"]])
sage: s = ShiftedPrimedTableau([[1,1,"2'","3'"],[2,2,"3'"]])
sage: s==t
True
```

The tableau can be represented in ascii, unicode or as a pretty print.

```
sage: t
[(1, 1, 2', 3'), (2, 2, 3')]
sage: t.pp()
 1  1  2' 3'
    2  2  3'
sage: ascii_art(t)
+---+---+---+---+
| 1 | 1 | 2'| 3'|
+---+---+---+---+
    | 2 | 2 | 3'|
    +---+---+---+
```

The code also includes a support for the skew version.

```
sage: s = ShiftedPrimedTableau([["2p","3p"],[2,2,"3p"]],skew=[2])
sage: s
[(None, None, 2', 3'), (2, 2, 3')]
sage: s.shape()
[4, 3] / [2]
sage: s.pp()
 .  .  2' 3'
    2  2  3'
```

To initialize the class of shifted primed tableaux of fixed shape $\mathcal{PT}(\lambda)$, we use `ShiftedPrimedTableaux`:

```
sage: Tabs = ShiftedPrimedTableaux(shape=[3,1])
sage: Tabs
```

[1]See the Trac ticket: https://trac.sagemath.org/ticket/22921.

```
    Shifted Primed Tableaux of shape [3, 1]
```

For the elements of the class `Tabs`, we can apply crystal operators f_i and e_i using the algorithm from this chapter.

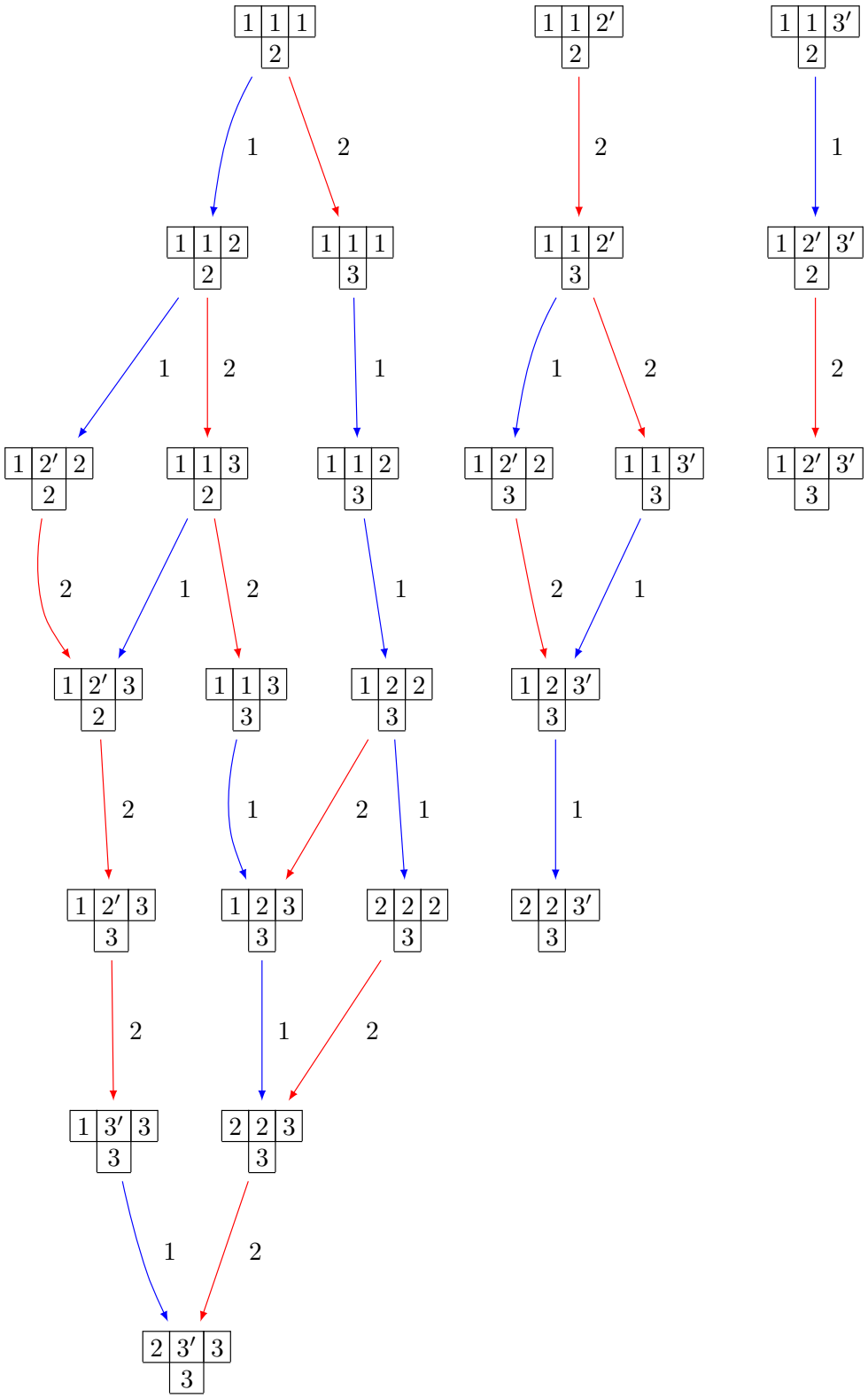
```
    sage: t = Tabs([[1,"2p",2],[2]])
    sage: t.parent()
    Shifted Primed Tableaux of shape [3, 1]
    sage: t.pp()
     1  2' 2
        2
    sage: t.e(1).pp()
     1  1  2
        2
    sage: t.f(2).pp()
     1  2' 3
        2
```

Specifying additional argument `max_entry` constructs a finite set of elements from $\mathcal{PT}(\lambda)$ with entries less then or equal to `max_entry`.

```
    sage: Tabs = ShiftedPrimedTableaux(shape=[3,1], max_entry=3)
    sage: Tabs[:4]
    [[(1, 1, 1), (2,)],
     [(1, 1, 2), (2,)],
     [(1, 2', 2), (2,)],
     [(1, 1, 3), (2,)]]
    sage: Tabs.cardinality()
    24
```

The crystal class `crystals.ShiftedPrimedTableaux` is another way to define the crystal on $\mathcal{PT}(\lambda)$. We use `view` to construct a LaTeX diagram of the crystal.

```
    sage: crystal = crystals.ShiftedPrimedTableaux([3,1],3)
    sage: [t for t in crystal.module_generators if t.is_highest_weight()]
    [[(1, 1, 1), (2,)], [(1, 1, 2'), (2,)], [(1, 1, 3'), (2,)]]
    sage: view(crystal)
```


1 1 1
2
1 1 2′
2
1 1 3′
2
1 1 2
2
1 1 1
3
1 1 2′
3
1 2′ 3′
2
1 2′ 2
2
1 1 3
2
1 1 2
3
1 2′ 2
3
1 1 3′
3
1 2′ 3′
3
1 2′ 3
2
1 1 3
3
1 2 2
3
1 2 3′
3
1 2′ 3
3
1 2 3
3
2 2 2
3
2 2 3′
3
1 3′ 3
3
2 2 3
3
2 3′ 3
3

2.6. Proof of Theorem 2.24

In this section, we provide the proof of Theorem 2.24.

2.6.1. Preliminaries. We use the fact from [**Hai89**] that taking only elements smaller or equal to $i+1$ from the word $\mathbf{b}$ and applying the mixed insertion corresponds to taking only the part of the tableau $\mathbf{T}$ with elements $\leqslant i+1$. Thus, it is enough to prove the theorem for a "truncated" word $\mathbf{b}$ without any letters greater than $i+1$. To shorten the notation, we set $j = i+1$ in this appendix. We sometimes also restrict to just the letters i and j in a word w. We call this the $\{i, j\}$-*subword* of w.

First, in Lemma 2.32 we justify the notion of the reading word $\mathrm{rw}(\mathbf{T})$ and provide the reason to use a bracketing rule on it. After that, in Section 2.6.2 we prove that the action of the crystal operator f_i on $\mathbf{b}$ corresponds to the action of f_i on $\mathbf{T}$ after the insertion.

Given a word $\mathbf{b}$, we apply the crystal bracketing rule for its $\{i, j\}$-subword and globally declare the rightmost unbracketed i in $\mathbf{b}$ (i.e. the letter the crystal operator f_i acts on) to be a bold i. Insert the letters of $\mathbf{b}$ via Haiman insertion to obtain the insertion tableau $\mathbf{T}$. During this process, we keep track of the position of the bold i in the tableau via the following rules. When the bold i from $\mathbf{b}$ is inserted into $\mathbf{T}$, it is inserted as the rightmost i in the first row of $\mathbf{T}$ since by definition it is unbracketed in $\mathbf{b}$ and hence cannot bump a letter j. From this point on, the tableau $\mathbf{T}$ has a *special* letter i and we track its position:

(1) If the special i is unprimed, it is always the rightmost i in its row. When a letter i is bumped from this row, only one of the non-special letters i can be bumped, unless the special i is the only i in the row. When the non-diagonal special i is bumped from its row to the next row, it will be inserted as the rightmost i in the next row.

(2) When the diagonal special i is bumped from its row to the column to its right, it is inserted as the bottommost i' in the next column.

(3) If the special i is primed, it is always the bottommost i' in its column. When a letter i' is bumped from this column, only one of the non-special letters i' can be bumped, unless the special i' is the only i' in the column. When the primed special i is bumped from its column to the next column, it is inserted as the bottommost i' in the next column.

(4) When i is inserted into a row with the special unprimed i, the rightmost i becomes special.

(5) When i' is inserted into a column with the special primed i, the bottommost primed i becomes special.

LEMMA 2.32. *Using the rules above, after the insertion process of* $\mathbf{b}$*, the special* i *in* $\mathbf{T}$ *is the same as the rightmost unbracketed* i *in the reading word* $\mathrm{rw}(\mathbf{T})$ *(i.e. the definition of the bold* i *in* $\mathbf{T}$*). Moreover, the number of unbracketed letters* i *in* $\mathbf{b}$ *is equal to the number of unbracketed letters* i *in* $\mathrm{rw}(\mathbf{T})$.

PROOF. First, note that since both the number of letters i and the number of letters j are equal in $\mathbf{b}$ and $\mathrm{rw}(\mathbf{T})$, the fact that the number of unbracketed letters i is the same implies that the number of unbracketed letters j must also be the same. We use induction on $1 \leqslant s \leqslant h$, where the letters $b_1 \dots b_s$ of $\mathbf{b} = b_1 b_2 \dots b_h$ have been inserted using Haiman mixed insertion with the above rules. That is, we check that at each step of the insertion algorithm the statement of our lemma stays true.

The induction step is as follows: Consider the word $b_1 \dots b_{s-1}$ with a corresponding insertion tableau $\mathbf{T}^{(s-1)}$. If the bold i in $\mathbf{b}$ is not in $b_1 \dots b_{s-1}$, then $\mathbf{T}^{(s-1)}$ does not contain a special letter i. Otherwise, by induction hypothesis assume that the bold i in $b_1 \dots b_{s-1}$ by the above rules corresponds to the special i in $\mathbf{T}^{(s-1)}$, that is, it is in the position corresponding to the rightmost unbracketed i in the reading word $\mathrm{rw}(\mathbf{T}^{(s-1)})$. Then we need to prove that for $b_1 \dots b_s$, the special i in $\mathbf{T}^{(s-1)}$ ends up in the position corresponding to the rightmost unbracketed i in the reading word of $\mathbf{T}^{(s)} = \mathbf{T}^{(s-1)} \leftsquigarrow b_s$. We also need to verify that the second part of the lemma remains true for $\mathbf{T}^{(s)}$.

Remember that we are only considering "truncated" words $\mathbf{b}$ with all letters $\leqslant j$.

Case 1. Suppose $b_s = j$. In this case j is inserted at the end of the first row of $\mathbf{T}^{(s-1)}$, and $\mathrm{rw}(\mathbf{T}^{(s)})$ has j attached at the end. Thus, both statements of the lemma are unaffected.

Case 2. Suppose $b_s = i$ and b_s is unbracketed in $b_1 \dots b_{s-1} b_s$. Then there is no special i in tableau $\mathbf{T}^{(s-1)}$, and b_s might be the bold i of the word $\mathbf{b}$. Also, there are no unbracketed letters j in $b_1 \dots b_{s-1}$, and thus all j in $\mathrm{rw}(\mathbf{T}^{(s-1)})$ are bracketed. Thus, there are no letters j in the first row of $\mathbf{T}^{(s-1)}$, and i is inserted in the first row of $\mathbf{T}^{(s-1)}$, possibly bumping the letter j' from column c into an empty column $c+1$ in the process. Note that if j' is bumped, moving it to column $c+1$ of $\mathbf{T}^{(s)}$ does not change the reading word, since column c of $\mathbf{T}^{(s-1)}$ does not contain any primed letters other than j'. The reading word of $\mathbf{T}^{(s)}$ is thus the same as $\mathrm{rw}(\mathbf{T}^{(s-1)})$ except for an additional unbracketed i at the end. The number of unbracketed letters i in both $\mathrm{rw}(\mathbf{T}^{(s)})$ and $b_1 \dots b_{s-1} b_s$ is thus increased by one compared to $\mathrm{rw}(\mathbf{T}^{(s-1)})$ and $b_1 \dots b_{s-1}$. If b_s is the bold i

of the word $\mathbf{b}$, the special i of tableau $\mathbf{T}^{(s)}$ is the rightmost i on the first row and corresponds to the rightmost unbracketed i in $\mathrm{rw}(\mathbf{T}^{(s)})$.

Case 3. Suppose $b_s = i$ and b_s is bracketed with a j in the word $b_1 \dots b_{s-1}$. In this case, according to the induction hypothesis, $\mathrm{rw}(\mathbf{T}^{(s-1)})$ has an unbracketed j. There are two options.

Case 3.1. If the first row of $\mathbf{T}^{(s-1)}$ does not contain j, b_s is inserted at the end of the first row of $\mathbf{T}^{(s-1)}$, possibly bumping j' in the process. Regardless, $\mathrm{rw}(\mathbf{T}^{(s)})$ does not change except for attaching an i at the end (see Case 2). This i is bracketed with one unbracketed j in $\mathrm{rw}(\mathbf{T}^{(s)})$. The special i (if there was one in $\mathbf{T}^{(s-1)}$) does not change its position and the statement of the lemma remains true.

Case 3.2. If the first row of $\mathbf{T}^{(s-1)}$ does contain a j, inserting b_s into $\mathbf{T}^{(s-1)}$ bumps j (possibly bumping j' beforehand) into the second row, where j is inserted at the end of the row. So, if the first row contains $n \geqslant 0$ elements i and $m \geqslant 1$ elements j, the reading word $\mathrm{rw}(\mathbf{T}^{(s-1)})$ ends with $\dots i^n j^m$, and $\mathrm{rw}(\mathbf{T}^{(s)})$ ends with $\dots j i^{n+1} j^{m-1}$. Thus, the number of unbracketed letters i does not change and if there was a special i in the first row, it remains there and it still corresponds to the rightmost unbracketed i in $\mathrm{rw}(\mathbf{T}^{(s)})$.

Case 4. Suppose $b_s < i$. Inserting b_s could change both the primed reading word and unprimed reading word of $\mathbf{T}^{(s-1)}$. As long as neither i nor j is bumped from the diagonal, we can treat primed and unprimed changes separately.

Case 4.1. Suppose neither i nor j is not bumped from the diagonal during the insertion. This means that there are no transitions of letters i or j between the primed and the unprimed parts of the reading word. Thus, it is enough to track the bracketing relations in the unprimed reading word; the bracketing relations in the primed reading word can be verified the same way via the transposition. After we make sure that the number of unbracketed letters i and j changes neither in the primed nor unprimed reading word, it is enough to consider the case when the special i is unprimed, since the case when it is primed can again be checked using the transposition. To avoid going back and forth, we combine these two processes together in each subcase to follow.

Case 4.1.1. If there are no letters i and j in the bumping sequence, the unprimed $\{i, j\}$-subword of $\mathrm{rw}(\mathbf{T}^{(s)})$ is the same as in $\mathrm{rw}(\mathbf{T}^{(s-1)})$. The special i (if there is one) remains in its position, and thus the statement of the lemma remains true.

Case 4.1.2. Now consider the case when there is a j in the bumping sequence, but no i. Let that j be bumped from the row r. Since there is no i bumped, row r does not contain any letters i. Thus, bumping j from row r to the end of row $r+1$ does not change the $\{i, j\}$-subword of $\mathrm{rw}(\mathbf{T}^{(s-1)})$, so the statement of the lemma remains true.

Case 4.1.3. Consider the case when there is an i in the bumping sequence. Let that i be bumped from the row r.

Case 4.1.3.1. If there is a (non-diagonal) j in row $r+1$, it is bumped into row $r+2$ (j' may have been bumped in the process). Note that in this case the i bumped from row r could not have been a special one. If there are $n \geqslant 0$ elements i and $m \geqslant 1$ elements j in row r, the part of the reading word $\mathrm{rw}(\mathbf{T}^{(s-1)})$ with $\ldots i^n j^m i \ldots$ changes to $\ldots j i^{n+1} j^{m-1} \ldots$ in $\mathrm{rw}(\mathbf{T}^{(s)})$. The bracketing relations remain the same, and if row $r+1$ contained a special i, it would remain there and would correspond to the rightmost i in $\mathrm{rw}(\mathbf{T}^{(s)})$.

Case 4.1.3.2. If there are no letters j in row $r+1$, and j' in row $r+1$ does not bump a j, the $\{i, j\}$-subword does not change and the statement of the lemma remains true.

Case 4.1.3.3. Now suppose there are no letters j in row $r+1$ and j' from row $r+1$ bumps a j from another row. This can only happen if, before the i was bumped, there was only one i in row r of $\mathbf{T}^{(s-1)}$, there is a j' right below it, and there is a j in the column to the right of i and in row $r' \leqslant r$.

If $r' = r$, then after the insertion process, i and j are bumped from row r to row $r+1$. Since there was only one i in row r and there are no letters j in row $r+1$, the $\{i, j\}$-subword of $\mathrm{rw}(\mathbf{T}^{(s-1)})$ does not change and the statement of the lemma remains true.

Otherwise $r' < r$. Then there are no letters i in row r' and by assumption there is no letter j in row $r+1$. Thus, moving i to row $r+1$ and moving j to the row $r'+1$ does not change the $\{i, j\}$-subword of $\mathrm{rw}(\mathbf{T}^{(s-1)})$ and the statement of the lemma remains true.

Case 4.2. Suppose i or j (or possibly both) are bumped from the diagonal in the insertion process.

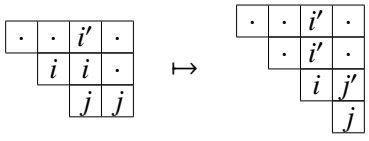
Case 4.2.1. Consider the case when the insertion sequence ends with $\cdots \to z \to j[j']$ with $z < i$ and possibly $\to j$ right after it. Let the bumped diagonal j be in column c. Then columns $1, 2, \ldots, c$ of $\mathbf{T}^{(s-1)}$ could only contain elements $\leqslant z$, except for the j on the diagonal. Thus, the bumping process just moves j from the unprimed reading word to the primed reading word without changing the overall order of the $\{i, j\}$-subword.

Case 4.2.2. Consider the case when the insertion sequence ends with $\cdots \to i' \to i \to j[j']$ and possibly $\to j$. Let the bumped diagonal j be in row (and column) r. Note that r must be the last row of $\mathbf{T}^{(s-1)}$. Then i has to be bumped from row $r-1$ (and, say, column c) and i' also has to be in row $r-1$ (moreover, it has to be the only i' in column $c-1$). Also, since there are no letters j' in column c (otherwise it would be in row r, which is impossible), bumping i' to column c does not change the $\{i, j\}$-subword of $\mathrm{rw}(\mathbf{T}^{(s-1)})$. Note that after i' moves to column c, there are no i' or j' in columns $1, \ldots, r$, and thus priming j and moving it to column $r+1$ does not change the $\{i, j\}$-subword. If the last row r contains n elements j, the $\{i, j\}$-subword of $\mathbf{T}^{(s-1)}$ contains $\ldots j^n i \ldots$ and after the insertion it becomes $\ldots j i j^{n-1} \ldots$, where the left j is from the primed subword. Thus, the number of bracketed letters i does not change. Also, if we moved the special i in the process, it could only have been the bumped i'. Its position in the reading word is unaffected.

Case 4.2.3. The case when the insertion sequence does not contain i', does not bump i from the diagonal, but contains i and bumps j from the diagonal is analogous to the previous case.

Case 4.2.4. Suppose both i and j are bumped from the diagonal. That could only be the case with diagonal i bumped from row (and column) r, bumping another letter i from the row r and column $r+1$, and bumping j from row (and column) $r+1$ (and possibly bumping j to row $r+2$ at the end). Let the number of letters i' in column $r+1$ be n and let the number of letters j in row $r+1$ be m.

Case 4.2.4.1 Let $m \geqslant 2$. Then the $\{i, j\}$-subword of $\mathrm{rw}(\mathbf{T}^{(s-1)})$ contains $\ldots i^n j^m i i \ldots$ and after the insertion it becomes $\ldots j i^{n+1} j i j^{m-2} \ldots$. The number of unbracketed letters i stays the same. Since $m \geqslant 2$, the special i of $\mathbf{T}^{(s-1)}$ could not have been involved in the bumping procedure. However, the special i might have been the bottommost i' in column $r+1$ of $\mathbf{T}^{(s-1)}$, and after the insertion the special i would still be the bottommost i' in column $r+1$ and would correspond to the rightmost unbracketed i in $\mathrm{rw}(\mathbf{T}^{(s)})$:



Case 4.2.4.2. Let $m = 1$. Then the $\{i, j\}$-subword of $\mathbf{T}^{(s-1)}$ contains $\ldots i^n j i i \ldots$ and after the insertion it becomes $\ldots j i^{n+1} i$. The number of unbracketed letters i stays the same. If the special i was in row r and column $r+1$, then after the insertion it becomes a diagonal one, and it would still correspond to the rightmost unbracketed i in $\mathrm{rw}(\mathbf{T}^{(s)})$.

Case 4.2.5. Suppose only i is bumped from the diagonal (let that i be on row and column r). Note that there cannot be an i' in column r.

Case 4.2.5.1. Suppose i from the diagonal bumps another i from column $r+1$ and row r. In that case there are no letters j in row $r+1$. No letters j or j' are affected and thus the $\{i, j\}$-subword of $\mathbf{T}^{(s)}$ does not change, and the special i in $\mathbf{T}^{(s)}$ (if there is one) still corresponds to the rightmost unbracketed i in $\mathrm{rw}(\mathbf{T}^{(s)})$.

Case 4.2.5.2. Suppose i from the diagonal bumps j' from column $r+1$ and row r. Note that j' must be the only j' in column $r+1$. Suppose also that there is one j in row $r+1$. Denote the number of letters i' in column $r+1$ of $\mathbf{T}^{(s-1)}$ by n. If there is a j in row $r+1$ of $\mathbf{T}^{(s-1)}$, then the $\{i, j\}$-subword of $\mathbf{T}^{(s-1)}$ contains $\ldots i^n j j i \ldots$ and after the insertion it becomes $\ldots j i^{n+1} j \ldots$. If there is no j in row $r+1$ of $\mathbf{T}^{(s-1)}$, then the $\{i, j\}$-subword of $\mathbf{T}^{(s-1)}$ contains $\ldots i^n j i \ldots$ and after the insertion it becomes $\ldots j i^{n+1} \ldots$. The number of unbracketed letters i is unaffected. If the special i of $\mathbf{T}^{(s-1)}$ was the bottommost i' in column $r+1$ of $\mathbf{T}^{(s-1)}$, after the insertion the special i is still the bottommost i' in column $r+1$ and corresponds to the rightmost unbracketed i in $\mathrm{rw}(\mathbf{T}^{(s)})$. □

Corollary 2.33.

$$f_i(\mathbf{b}) = \mathbf{0} \quad \textit{if and only if} \quad f_i(\mathbf{T}) = \mathbf{0}.$$

2.6.2. Proof of Theorem 2.24. By Lemma 2.32, the cell x in the definition of the operator f_i corresponds to the bold i in the tableau $\mathbf{T}$. Furthermore, we know how the bold i moves during the insertion procedure. We assume that the bold i exists in both $\mathbf{b}$ and $\mathbf{T}$, meaning that $f_i(\mathbf{b}) \neq \mathbf{0}$ and $f_i(\mathbf{T}) \neq \mathbf{0}$ by Corollary 2.33. We prove Theorem 2.24 by induction on the length of the word $\mathbf{b}$.

Base. Our base is for words $\mathbf{b}$ with the last letter being a bold i (i.e. rightmost unbracketed i). Let $\mathbf{b} = b_1 \ldots b_{h-1} b_h$ and $f_i(\mathbf{b}) = b_1 \ldots b_{h-1} b'_h$, where $b_h = i$ and $b'_h = j$. Denote the mixed insertion tableau of $b_1 \ldots b_{h-1}$ as $\mathbf{T}_0$, the insertion tableau of $b_1 \ldots b_{h-1} b_h$ as $\mathbf{T}$, and the insertion tableau of $b_1 \ldots b_{h-1} b'_h$ as $\mathbf{T}'$. Note that $\mathbf{T}_0$ does not have letters j in the first row. If the first row of $\mathbf{T}_0$ ends with $\ldots j'$, then the first row of $\mathbf{T}$ ends with $\ldots \mathbf{i}\, j'$ and the first row of $\mathbf{T}'$ ends with $\ldots j' j$. If the first row of $\mathbf{T}_0$ does not contain j', the first row of $\mathbf{T}$ ends with $\ldots \mathbf{i}$ and the first row of $\mathbf{T}'$ ends with $\ldots j$, and the cell x_S is empty. In both cases $f_i(\mathbf{T}) = \mathbf{T}'$.

Induction step. Now, let $\mathbf{b} = b_1 \ldots b_h$ with operator f_i acting on the letter b_s in $\mathbf{b}$ with $s < h$. Denote the mixed insertion tableau of $b_1 \ldots b_{h-1}$ as $\mathbf{T}$ and the insertion tableau of $f_i(b_1 \ldots b_{h-1})$ as $\mathbf{T}'$. By induction

hypothesis, we know that $f_i(\mathbf{T}) = \mathbf{T}'$. We want to show that $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$. In Cases 1-3 below, we assume that the bold letter i is unprimed. Since almost all results from the case with unprimed i are transferrable to the case with primed bold i via the transposition of the tableau $\mathbf{T}$, we just need to cover the differences in Case 4.

Case 1. Suppose $\mathbf{T}$ falls under Case (1) of the rules for f_i: the bold i is in the non-diagonal cell x in row r and column c and the cell x_E in the same row and column $c + 1$ has content j'. Consider the insertion path of b_h.

Case 1.1. If the insertion path of b_h in $\mathbf{T}$ contains neither cell x nor cell x_E, the insertion path of b_h in $\mathbf{T}'$ also does not contain cells x and x_E. Thus, $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 1.2. Suppose that during the insertion of b_h into $\mathbf{T}$, the bold i is row-bumped by an unprimed element $d < i$ or is column-bumped by a primed element $d' \leqslant i'$. This could only happen if the bold i is the unique i in row r of $\mathbf{T}$. During the insertion process, the bold i is inserted into row $r + 1$. Since there are no letters i in row r of $\mathbf{T}'$, inserting b_h into $\mathbf{T}'$ inserts d in cell x, bumps j' to cell x_E, and bumps j into row $r + 1$. Thus we are in a situation similar to the induction base. It is easy to check that row $r + 1$ does not contain any letters j in $\mathbf{T}$. If it contains j', this j' is bumped back into row $r + 1$. Similar to the induction base, $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 1.3. Suppose that during the insertion of b_h into $\mathbf{T}$, an unprimed i is inserted into row r. Note that in this case, row r in $\mathbf{T}$ must contain a j (or else the i from row r would not be the rightmost unbracketed i in rw($\mathbf{T}$)). Thus inserting i into row r in $\mathbf{T}$ shifts the bold i to column $c + 1$, shifts j' to column $c + 2$ and bumps j to row $r + 1$. Inserting i into row r in $\mathbf{T}'$ shifts j' to column $c + 1$ with a j to the right of it, and bumps j into row $r + 1$. Thus $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 1.4. Suppose that during the insertion of b_h into $\mathbf{T}$, the j' in cell x_E is column-bumped by a primed element d' and the cell x is unaffected. Note that in order for $\mathbf{T} \leftsquigarrow b_h$ to be a valid primed tableau, i must be smaller than d', and thus d' could only be j'. On the other hand, j' cannot be inserted into column $c + 1$ of $\mathbf{T}'$ in order for $\mathbf{T}' \leftsquigarrow b_h$ to be a valid primed tableau. Thus this case is impossible.

Case 2. Suppose tableau $\mathbf{T}$ falls under Case (2a) of the crystal operator rules for f_i. This means that for a bold i in cell x (in row r and column c) of tableau $\mathbf{T}$, the cell x_E has content j or is empty and cell x_S is

empty. Tableau $\mathbf{T}'$ has all the same elements as $\mathbf{T}$, except for a j in the cell x. We are interested in the case when inserting b_h into either $\mathbf{T}$ or $\mathbf{T}'$ bumps the element from cell x.

Case 2.1. Suppose that the non-diagonal bold i in $\mathbf{T}$ (in row r) is row-bumped by an unprimed element $d < i$ or column-bumped by a primed element $d' < j'$. Element d (or d') bumps the bold i into row $r + 1$ of $\mathbf{T}$, while in $\mathbf{T}'$ (since there are no letters i in row r of $\mathbf{T}'$) it bumps j from cell x into row $r + 1$. Thus we are in the situation of the induction base and $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 2.2. Suppose x is a non-diagonal cell in row r, and during the insertion of b_h into $\mathbf{T}$, an unprimed i is inserted into the row r. In this case, row r in $\mathbf{T}$ must contain a letter j. The insertion process shifts the bold i one cell to the right in $\mathbf{T}$ and bumps a j into row $r + 1$, while in $\mathbf{T}'$ it just bumps j into the row $r + 1$. We end up in Case (2a) of the crystal operator rules for f_i with bold i in the cell x_E.

Case 2.3. Suppose that during the insertion of b_h into $\mathbf{T}'$, the j in the non-diagonal cell x is column-bumped by a j'. This means that j' was previously bumped from column $c - 1$ and row $\geqslant r$. Thus the cell x_{SW} (cell to the left of an empty x_S) is non-empty. Moreover, right before inserting j' into the column c, the cell x_{SW} has content $< j'$. Inserting j' into column c of $\mathbf{T}$ just places j' into the empty cell x_S. Inserting j' into column c of $\mathbf{T}'$ places j' into x, and bumps j into the empty cell x_S. Thus, we end up in Case (2c) of the crystal operator rules after the insertion of b_h with $y = x_S$.

Case 2.4. Suppose that x in $\mathbf{T}$ is a diagonal cell (in row r and column r) and that it is row-bumped by an element $d < i$. Note that in this case there cannot be any letter j in row $r + 1$. Also, since d is inserted into cell x, there cannot be any letters i' in columns $1, \ldots, r$, and thus there cannot be any letters j' in column $r + 1$ (otherwise the i in cell x would not be bold). The bumped bold i in tableau $\mathbf{T}$ is inserted as a primed bold i' into the cell z of column $r + 1$.

Case 2.4.1. Suppose that there are no letters i in column $r + 1$ of $\mathbf{T}$. In this case, the cell z in $\mathbf{T}$ either contains j (and then that j would be bumped to the next row) or is empty. Inserting b_h into tableau $\mathbf{T}'$ bumps the diagonal j in cell x, which is inserted as a j' into cell z, possibly bumping j after that. Thus, $\mathbf{T} \leftsquigarrow b_h$ falls under Case (2a) of the "primed" crystal rules with the bold i' in cell z (note that there cannot be any j' in cell $(z*)_E$ of the tableau $(\mathbf{T} \leftsquigarrow b_h)*$). Since $\mathbf{T} \leftsquigarrow b_h$ and $\mathbf{T}' \leftsquigarrow b_h$ differ only by the cell z, $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 2.4.2. Suppose that there is a letter i in cell z of column $r+1$ of $\mathbf{T}$. Note that cell z can only be in rows $1, \ldots, r-1$ and thus z_{SW} contains an element $< i$. Thus, during the insertion process of b_h into $\mathbf{T}$, diagonal bold i from cell x is inserted as bold i' into cell z, bumping the i from cell z into cell z_S (possibly bumping j afterwards). On the other hand, inserting b_h into $\mathbf{T}'$ bumps the diagonal j from cell x into cell z_S as a j' (possibly bumping j afterwards). Thus, $\mathbf{T} \leftsquigarrow b_h$ falls under Case (1) of the "primed" crystal rules with the bold i' in cell z, and so $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 2.5. Suppose that x is a diagonal cell (in row r and column r) and that during the insertion of b_h into $\mathbf{T}$, an unprimed i is inserted into row r. In this case, the content of cell x_E has to be j and the diagonal cell x_{ES} must be empty. Inserting i into row r of $\mathbf{T}$ bumps a j from cell x_E into cell x_{ES}. On the other hand, inserting i into row r of $\mathbf{T}'$ bumps a j from the diagonal cell x, which in turn is inserted as a j' into cell x_E, which bumps j from cell x_E into cell x_{ES}. Thus, $\mathbf{T} \leftsquigarrow b_h$ falls under Case (2b) of the crystal rules with bold i in cell x_E and $y = x_{ES}$, and so $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 3. Suppose that $\mathbf{T}$ falls under Case (2b) or (2c) of the crystal operator rules. That means x_E has content j or is empty and x_S has content j' or j. There is a chain of letters j' and j in $\mathbf{T}$ starting from x_S and ending on a box y. According to the induction hypothesis, y is either on the diagonal and has content j or y is not on the diagonal and has content j'. The tableau $\mathbf{T}' = f_i(\mathbf{T})$ has j' in cell x and j in cell y. We are interested in the case when inserting b_h into $\mathbf{T}$ affects cell x or affects some element of the chain. Let r_x and c_x be the row and the column index of cell x, and r_y, c_y are defined accordingly. Note that during the insertion process, j' cannot be inserted into columns $c_y, \ldots, c_x$ and j cannot be inserted into rows $r_x + 1, \ldots, r_y$, since otherwise $\mathbf{T} \leftsquigarrow b_h$ would not be a primed tableau.

Case 3.1. Suppose the bold i in cell x (of row r_x and column c_x) of $\mathbf{T}$ is row-bumped by an unprimed element $d < i$ or column-bumped by a primed element $d' < i$. Note that in this case, bold i in row r_x is the only i in this row, so row $r_x + 1$ cannot contain any letter j. Therefore the content of x_S must be j'. In tableau $\mathbf{T}$, the bumped bold i is inserted into cell x_S and j' is bumped from cell x_S into column $c_x + 1$, reducing the chain of letters j' and j by one. Notice that since x_E either contains a j or is empty, j' cannot be bumped into a position to the right of x_S, so Case (1) of the crystal rules for $\mathbf{T} \leftsquigarrow b_h$ cannot occur. As for $\mathbf{T}'$, inserting d into row r_x (or inserting d' into column c_x) just bumps j' into column $c_x + 1$, thus reducing the length of the chain by one in that tableau as well. Note that in the case when the length of the chain is one (i.e. $y = x_S$),

we would end up in Case (2a) of the crystal rules after the insertion. Otherwise, we are still in Case (2b) or (2c). In both cases, $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 3.2. Suppose a letter i is inserted into the same row as x (in row r_x). In this case, x_E must contain a j (otherwise the bold i would not be in cell x). After inserting b_h into $\mathbf{T}$, the bold i moves to cell x_E (note that there cannot be a j' to the right of x_E) and j from x_E is bumped to cell x_{ES}, thus the chain now starts at x_{ES}. As for $\mathbf{T}'$, inserting i into the row r_x moves j' from cell x to the cell x_E and moves j from cell x_E to cell x_{ES}. Thus, $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 3.3. Consider the chain of letters j and j' in $\mathbf{T}$. Suppose an element of the chain $z \neq x, y$ is row-bumped by an element $d < j$ or is column-bumped by an element $d' < j'$. The bumped element z (of row r_z and column c_z) must be a "corner" element of the chain, i.e. in $\mathbf{T}$ the contents of boxes must be $c(z) = j'$, $c(z_E) = j$ and $c(z_S)$ must be either j or j'. Therefore, inserting b_h into $\mathbf{T}$ bumps j' from box z to box z_E and bumps j from box z_E to box z_{ES}, and inserting b_h into $\mathbf{T}'$ has exactly the same effect. Thus, there is still a chain of letters j and j' from x_S to y in $\mathbf{T}$ and $\mathbf{T}'$, and $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 3.4. Suppose $\mathbf{T}$ falls under Case (2c) of the crystal rules (i.e. y is not a diagonal cell) and during the insertion of b_h into $\mathbf{T}$, j' in cell y is row-bumped (resp. column-bumped) by an element $d < j'$ (resp. $d' < j'$). Since y is the end of the chain of letters j and j', y_S must be empty. Also, since it is bumped, the content of y_E must be j. Thus, inserting b_h into $\mathbf{T}$ bumps j' from cell y to cell y_E and bumps j from cell y_E into row $r_y + 1$ and column $\leqslant c_y$. On the other hand, inserting b_h into $\mathbf{T}'$ bumps j from cell y into row $r_y + 1$ and column $\leqslant c_y$. The chain of letters j and j' now ends at y_E and $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 3.5. Suppose $\mathbf{T}$ falls under Case (2b) of the crystal rules (i.e. y with content j is a diagonal cell) and during the insertion of b_h into $\mathbf{T}$, j in cell y is row-bumped by an element $d < j$. In this case, the cell y_E must have content j. Thus, inserting b_h into $\mathbf{T}$ bumps j from cell y (making it j') to cell y_E and bumps j from cell y_E to the diagonal cell y_{ES}. On the other hand, inserting b_h into $\mathbf{T}'$ has exactly the same effect. The chain of letters j and j' now ends at the diagonal cell y_{ES}, so $\mathbf{T} \leftsquigarrow b_h$ falls under Case (2b) of the crystal rules and $f_i(\mathbf{T} \leftsquigarrow b_h) = \mathbf{T}' \leftsquigarrow b_h$.

Case 4. Suppose the bold i in tableau $\mathbf{T}$ is a primed i. We use the transposition operation on $\mathbf{T}$, and the resulting tableau $\mathbf{T}^*$ falls under one of the cases of the crystal operator rules. When b_h is inserted into $\mathbf{T}$, we can easily translate the insertion process to the transposed tableau $\mathbf{T}^*$ so that $[\mathbf{T}^* \leftsquigarrow (b_h + 1)'] = [\mathbf{T} \leftsquigarrow b_h]^*$:

the letter $(b_h + 1)'$ is inserted into the first column of $\mathbf{T}^*$, and all other insertion rules stay exactly same, with one exception – when the diagonal element d' is column-bumped from the diagonal cell of $\mathbf{T}^*$, the element d' becomes $(d - 1)$ and is inserted into the row below. Notice that the primed reading word of $\mathbf{T}$ becomes an unprimed reading word of $\mathbf{T}^*$. Thus, the bold i in tableau $\mathbf{T}^*$ corresponds to the rightmost unbracketed i in the *unprimed* reading word of $\mathbf{T}^*$. Therefore, everything we have deduced in Cases 1-3 from the fact that bold i is in the cell x will remain valid here. Given $f_i(\mathbf{T}^*) = \mathbf{T}'^*$, we want to make sure that $f_i(\mathbf{T}^* \leftsquigarrow (b_h + 1)') = \mathbf{T}'^* \leftsquigarrow (b_h + 1)'$.

The insertion process of $(b_h + 1)'$ into $\mathbf{T}^*$ falls under one of the cases above and the proof of $f_i(\mathbf{T}^* \leftsquigarrow (b_h + 1)') = \mathbf{T}'^* \leftsquigarrow (b_h + 1)'$ is exactly the same as the proof in those cases. We only need to check the cases in which the diagonal element might be affected differently in the insertion process of $(b_h + 1)'$ into $\mathbf{T}^*$ compared to the insertion process of $(b_h + 1)'$ into $\mathbf{T}'^*$. Fortunately, this never happens: in Case 1 neither x nor x_E could be diagonal elements; in Cases 2 and 3 x cannot be on the diagonal, and if x_E is on diagonal, it must be empty. Following the proof of those cases, $f_i(\mathbf{T}^* \leftsquigarrow (b_h + 1)') = \mathbf{T}'^* \leftsquigarrow (b_h + 1)'$.

2.7. Proof of Theorem 2.26

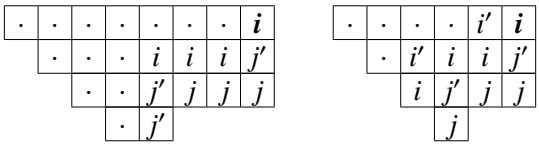
This section provides the proof of Theorem 2.26. Here, we set $j = i + 1$. We begin with two preliminary lemmas.

2.7.1. Preliminaries.

LEMMA 2.34. *Consider a shifted tableau* $\mathbf{T}$.

(1) *Suppose tableau* $\mathbf{T}$ *falls under Case (2c) of the* f_i *crystal operator rules, that is, there is a chain of letters* j *and* j' *starting from the bold* i *in cell* x *and ending at* j' *in cell* x_H*. Then for any cell* z *of the chain containing* j*, the cell* z_{NW} *contains* i*.*

(2) *Suppose tableau* $\mathbf{T}$ *falls under Case (2b) of the* f_i *crystal operator rules, that is, there is a chain of letters* j *and* j' *starting from the bold* i *in cell* x *and ending at* j *in the diagonal cell* x_H*. Then for any cell* z *of the chain containing* j *or* j'*, the cell* z_{NW} *contains* i *or* i' *respectively.*



PROOF. The proof of the first part is based on the observation that every j in the chain must be bracketed with some i in the reading word rw($\mathbf{T}$). Moreover, if the bold i is located in row r_x and rows $r_x, r_x + 1, \ldots, r_z$ contain n letters j, then rows $r_x, r_x + 1, \ldots, r_z - 1$ must contain exactly n non-bold letters i. To prove that these elements i must be located in the cells to the North-West of the cells containing j, we proceed by induction on n. When we consider the next cell z containing j in the chain that must be bracketed, notice that the columns $c_z, c_z + 1, \ldots, c_x$ already contain an i, and thus we must put the next i in column $c_z - 1$; there is no other row to put it than $r_z - 1$. Thus, z_{NW} must contain an i.

This line of logic also works for the second part of the lemma. We can show that for any cell z of the chain containing j, the cell z_{NW} must contain an i. As for cells z containing j', we can again use the fact that the corresponding letters j in the primed reading word of $\mathbf{T}$ must be bracketed. Notice that these letters j' cannot be bracketed with unprimed letters i, since all unprimed letters i are already bracketed with unprimed letters j. Thus, j' must be bracketed with some i' from a column to its left. Let columns $1, 2, \ldots, c_z$ contain m elements j'. Using the same induction argument as in the previous case, we can show that z_{NW} must contain i'. □

Next we need to figure out how y in the raising crystal operator e_i is related to the lowering operator rules for f_i.

LEMMA 2.35. *Consider a pair of tableaux* $\mathbf{T}$ *and* $\mathbf{T}' = f_i(\mathbf{T})$.

(1) *If tableau* $\mathbf{T}$ *(in case when bold* i *in* $\mathbf{T}$ *is unprimed) or* $\mathbf{T}^*$ *(if bold* i *is primed) falls under Case (1) of the* f_i *crystal operator rules, then cell* y *of the* e_i *crystal operator rules is cell* x_E *of* $\mathbf{T}'$ *or* $(\mathbf{T}')^*$*, respectively.*

(2) *If tableau* $\mathbf{T}$ *(in case when bold* i *in* $\mathbf{T}$ *is unprimed) or* $\mathbf{T}^*$ *(if bold* i *is primed) falls under Case (2a) of the* f_i *crystal operator rules, then cell* y *of the* e_i *crystal operator rules is located in cell* x *of* $\mathbf{T}'$ *or* $(\mathbf{T}')^*$*, respectively.*

(3) *If tableau* $\mathbf{T}$ *falls under Case (2b) of the* f_i *crystal operator rules, then cell* y *of the* e_i *crystal operator rules is cell* x^* *of* $(\mathbf{T}')^*$.

(4) *If tableau* $\mathbf{T}$ *(in case when bold* i *in* $\mathbf{T}$ *is unprimed) or* $\mathbf{T}^*$ *(if bold* i *is primed) falls under Case (2c) of the* f_i *crystal operator rules, then cell* y *of the* e_i *crystal operator rules is cell* x_H *of* $\mathbf{T}'$ *or* $(\mathbf{T}')^*$*, respectively.*

PROOF. In all the cases above, we need to compare reading words rw($\mathbf{T}$) and rw($\mathbf{T}'$). Since f_i affects at most two boxes of $\mathbf{T}$, it is easy to track how the reading word rw($\mathbf{T}$) changes after applying f_i. We want to check where the bold j under e_i ends up in rw($\mathbf{T}'$) and in $\mathbf{T}'$, which allows us to determine the cell y of the e_i crystal operator rules.

Case 1.1. Suppose $\mathbf{T}$ falls under Case (1) of the f_i crystal operator rules, that is, the bold i in cell x is to the left of j' in cell x_E. Furthermore, f_i acts on $\mathbf{T}$ by changing the content of x to j' and by changing the content of x_E to j. In the reading word rw($\mathbf{T}$), this corresponds to moving the j corresponding to x_E to the left and changing the bold i (the rightmost unbracketed i) corresponding to cell x to j (that then corresponds to x_E). Moving a bracketed j in rw($\mathbf{T}$) to the left does not change the $\{i, j\}$ bracketing, and thus the j corresponding to x_E in rw($\mathbf{T}'$) is still the leftmost unbracketed j. Therefore, this j is the bold j of $\mathbf{T}'$ and is located in cell x_E.

Case 1.2. Suppose the bold i in $\mathbf{T}$ is primed and $\mathbf{T}^*$ falls under Case (1) of the f_i crystal operator rules. After applying lowering crystal operator rules to $\mathbf{T}^*$ and conjugating back, the bold primed i in cell x^* of $\mathbf{T}$ changes to an unprimed i, and the unprimed i in cell $(x^*)_S$ of $\mathbf{T}$ changes to j'. In terms of the reading word of $\mathbf{T}$, it means moving the bracketed i (in the unprimed reading word) corresponding to $(x^*)_S$ to the left so that it corresponds to x^*, and then changing the bold i (in the primed reading word) corresponding to x^* into the letter j corresponding to $(x^*)_S$. The first operation does not change the bracketing relations between i and j, and thus the leftmost unbracketed j in rw($\mathbf{T}'$) corresponds to $(x^*)_S$. Hence the bold unprimed j is in cell x_E of $(\mathbf{T}')^*$.

Case 2.1. If $\mathbf{T}$ falls under Case (2a) of the f_i crystal operator rules, f_i just changes the content of x from i to j. The rightmost unbracketed i in the reading word of $\mathbf{T}$ changes to the leftmost unbracketed j in rw($\mathbf{T}'$). Thus, the bold j in rw($\mathbf{T}'$) corresponds to cell x.

Case 2.2. The case when $\mathbf{T}^*$ falls under Case (2a) of the f_i crystal operator rules is the same as the previous case.

Case 3. Suppose $\mathbf{T}$ falls under Case (2b) of f_i crystal operator rules. Then there is a chain starting from cell x (of row r_x and column c_x) and ending at the diagonal cell z (of row and column r_z) consisting of elements j and j'. Applying f_i to $\mathbf{T}$ changes the content of x from i to j'. In rw($\mathbf{T}$) this implies moving the bold i from the unprimed reading word to the left through elements i and j corresponding to rows $r_x, r_x + 1, \ldots, r_z$,

then through elements i and j in the primed reading word corresponding to columns $c_z - 1, \ldots, c_x$, and then changing that i to j which corresponds to cell x. But according to Lemma 2.34, the letters i and j in these rows and columns are all bracketed with each other, since for every j or j' in the chain there is a corresponding i or i' in the North-Western cell. (Notice that there cannot be any other letter j or j' outside of the chain in rows $r_x + 1, \ldots, r_z$ and in columns $c_z - 1, \ldots, c_x$.) Thus, moving the bold i to the left in rw($\mathbf{T}$) does not change the bracketing relations. Changing it to j makes it the leftmost unbracketed j in rw($\mathbf{T}'$). Therefore, the bold j in rw($\mathbf{T}'$) corresponds to the primed j in cell x of $\mathbf{T}'$, and the cell y of the e_i crystal operator rules is thus cell x^* in $(\mathbf{T}')^*$.

Case 4.1. Suppose $\mathbf{T}$ falls under Case (2c) of the f_i crystal operator rules. There is a chain starting from cell x (in row r_x and column c_x) and ending at cell x_H (in row r_H and column c_H) consisting of elements j and j'. Applying f_i to $\mathbf{T}$ changes the content of x from i to j' and changes the content of x_H from j' to j. Moving j' from cell x_H to cell x moves the corresponding bracketed j in the reading word rw($\mathbf{T}$) to the left, and thus does not change the $\{i, j\}$ bracketing relations in rw($\mathbf{T}'$). On the other hand, moving the bold i from cell x to cell x_H and then changing it to j moves the bold i in rw($\mathbf{T}$) to the right through elements i and j corresponding to rows $r_x, r_x + 1, \ldots, r_H$, and then changes it to j. Note that according to Lemma 2.34, each j in rows $r_x + 1, r_x + 2, \ldots, r_H$ has a corresponding i from rows $r_x, r_x + 1, \ldots, r_H - 1$ that it is bracketed with, and vise versa. Thus, moving the bold i to the position corresponding to x_H does not change the fact that it is the rightmost unbracketed i in rw($\mathbf{T}$). Thus, the bold j in rw($\mathbf{T}'$) corresponds to the unprimed j in cell x_H of $\mathbf{T}'$.

Case 4.2. Suppose $\mathbf{T}$ has a primed bold i and $\mathbf{T}^*$ falls under Case (2c) of the f_i crystal operator rules. This means that there is a chain (expanding in North and East directions) in $\mathbf{T}$ starting from i' in cell x^* and ending in cell x^*_H with content i consisting of elements i and j'. The crystal operator f_i changes the content of cell x^* from i' to i and changes the content of x^*_H from i to j'. For the reading word rw($\mathbf{T}$) this means moving the bracketed i in the unprimed reading word to the right (which does not change the bracketing relations) and moving the bold i in the primed reading word through letters i and j corresponding to columns $c_x, c_x + 1, \ldots, c_H$, which are bracketed with each other according to Lemma 2.34. Thus, after changing the bold i to j makes it the leftmost unbracketed j in rw($\mathbf{T}'$). Hence the bold primed j in $\mathbf{T}'$ corresponds to cell x^*_H. Therefore y from the e_i crystal operator rules is cell x_H of $(\mathbf{T}')^*$. □

2.7.2. Proof of Theorem 2.26. Let $\mathbf{T}' = f_i(\mathbf{T})$.

Case 1. If $\mathbf{T}$ (or $\mathbf{T}^*$) falls under Case (1) of the f_i crystal operator rules, then according to Lemma 2.35, e_i acts on $\mathbf{T}'$ (or on $(\mathbf{T}')^*$) by changing the content of cell $y_W = x$ back to i and changing the content of $y = x_E$ back to j'. Thus, the statement of the theorem is true.

Case 2. If $\mathbf{T}$ (or $\mathbf{T}^*$) falls under Case (2a) of the f_i crystal operator rules, then according to Lemma 2.35, e_i acts on $\mathbf{T}'$ (or on $(\mathbf{T}')^*$) by changing the content of the cell $y = x$ back to i. Thus, the statement of the theorem is true.

Case 3. If $\mathbf{T}$ falls under Case (2b) of the f_i crystal operator rules, then according to Lemma 2.35, e_i acts on cell $y = x^*$ of $(\mathbf{T}')^*$. Note that according to Lemma 2.34, there is a maximal chain of letters i and j' in $(\mathbf{T}')^*$ starting at y and ending at a diagonal cell y_T. Thus, e_i changes the content of cell $y = x^*$ in $(\mathbf{T}')^*$ from j to j', so the content of cell x in $\mathbf{T}'$ goes back from j' to i. Thus, the statement of the theorem is true.

Case 4. If $\mathbf{T}$ (or $\mathbf{T}^*$) falls under Case (2c) of the f_i crystal operator rules, then according to Lemma 2.35, e_i acts on cell $y = x_H$ of $\mathbf{T}'$ (or of $(\mathbf{T}')^*$). Note that according to Lemma 2.34, there is a maximal (since $c(x_E) \neq j'$ and $c(x_E) \neq i$) chain of letters i and j' in $\mathbf{T}'$ (or $(\mathbf{T}')^*$) starting at y and ending at cell $y_T = x$. Thus, e_i changes the content of cell $y = x_H$ in $(\mathbf{T}')^*$ from j back to j' and changes the content of $y_T = x$ from j' back to i. Thus, the statement of the theorem is true.

2.8. Outlook

There are several other generalizations of the results in this paper that one could pursue. First of all, it would be interesting to consider affine Stanley symmetric functions of type B or C. As in affine type A, this would involve a generalization of crystal bases as the expansion is no longer in terms of Schur functions. Another possible extension is to consider K-theoretic analogues of Stanley symmetric functions, such as the (dual) stable Grothendieck polynomials. In type A, a crystal theoretic analysis of dual stable Grothendieck polynomials was carried out in [**Gal15**]. Finally, type D should also be considered from this point of view.

CHAPTER 3

Cores with distinct parts and bigraded Fibonacci numbers

3.1. Introduction

For two coprime integers a and b, the rational Catalan number $C_{a,b}$ and its bigraded generalization $C_{a,b}(q,t)$ have caught the attention of different researchers due to their connection to algebraic combinatorics and geometry [**AHJ14**, **ALW16**, **GM16**, **GMV16**]. Catalan numbers can be analyzed from the perspective of different combinatorial objects: rational (a,b)-Dyck paths, simultaneous (a,b)-core partitions and abacus diagrams.

In 2015, Amdeberhan [**Amd**] conjectured that the number of $(a,a+1)$-cores with distinct parts is equal to the Fibonacci number F_{a+1}, and also conjectured the formulas for the largest size and the average size of such partitions. This conjecture has been proven by Xiong:

THEOREM 3.1. *(Xiong, [**Xioa**]) For $(a,a+1)$-core partitions with distinct parts, we have*

(1) *the number of such partitions is F_{a+1};*

(2) *the largest size of such partition is $\lfloor \frac{1}{3}\binom{a+1}{2} \rfloor$;*

(3) *there are $\frac{3-(-1)^{a \mod 3}}{2}$ such partitions of maximal size;*

(4) *the total number of these partitions and the average sizes are, respectively, given by*

$$\sum_{i+j+k=a+1} F_i F_j F_k \quad \text{and} \quad \sum_{i+j+k=a+1} \frac{F_i F_j F_k}{F_{a+1}}.$$

Part (1) of the above theorem was independently proved by Straub [**Str16**].

Another interesting conjecture of Amdeberhan is the number of $(2k-1,2k+1)$-cores with distinct parts. This conjecture has been proven by Yan, Qin, Jin and Zhou:

THEOREM 3.2. *(YQJZ, [**YQJZ17**]) The number of $(2k-1,2k+1)$-cores with distinct parts is equal to 2^{2k-2}.*

The proof uses somewhat complicated arguments about the poset structure of cores. Results by Zaleski and Zeilberger [**ZZ17**] improve the argument using Experimental Mathematics tools in Maple. More recently Baek, Nam and Yu provided a simpler bijective proof in [**BNY**].

Another set of combinatorial objects that has caught the attention of a number of researchers [**NS17**, **Str16**, **Xiob**, **Zal**] is the set of $(a, as \pm 1)$-cores with distinct parts. In particular, there is a Fibonacci-like recursive relation for the number of such cores:

THEOREM 3.3. *(Straub, [**Str16**]) The number $E_s^-(a)$ of $(a, as-1)$-core partitions with distinct parts is characterized by $E_s^-(1) = 1,\ E_s^-(2) = s$ and, for $a \geq 3$,*

$$E_s^-(a) = E_s^-(a-1) + sE_s^-(a-2).$$

THEOREM 3.4. *(Nath and Sellers, [**NS17**]) The number $E_s^+(a)$ of $(a, as+1)$-core partitions with distinct parts is characterized by $E_s^+(1) = 1,\ E_s^+(2) = s+1$ and, for $a \geq 3$,*

$$E_s^+(a) = E_s^+(a-1) + sE_s^+(a-2).$$

In this paper, we analyze simultaneous core partitions in the context of Anderson's bijection and in Section 3.3 we provide a simple description of the set of $(a, as+1)$-cores with distinct parts in terms of abacus diagrams, which also allows us to provide another proof of Theorem 3.1 parts (1), (2) and (3) in Section 3.4.

In Section 3.5 we use the connection between cores and Dyck paths to provide another simple proof of Theorem 3.2.

In Section 3.6 we introduce graded Fibonacci numbers

$$F_{a,b}(q) = \sum_{\kappa} q^{area(\kappa)},$$

where the sum is taken over all (a, b)-cores κ with distinct parts and *area* is some statistic on (a, b)-cores. We show that $F_{a,a+1}(1) = F_{a+1}$- the regular Fibonacci sequence, and prove recursive relations for $F_a^{(s)}(q) := F_{a,as+1}(q)$. Using properties of $F_{a,a+1}(q)$ we provide another proof of Theorem 3.4 and another proof of Theorem 3.1 part (4).

In Section 3.7 we introduce bigraded Fibonacci number as a summand of bigraded Catalan numbers:

$$F_a^{(s)}(q,t) = \sum_{\pi} q^{area(\pi)} t^{bounce(\pi)},$$

where the sum is taken over all $(a, as+1)$- Dyck paths corresponding to $(a, as+1)$-cores with distinct parts, and statistics $(area, bounce)$ are two standard statistics on Dyck paths (see [**Loe05**]).

Using abacus diagrams, we can get a simple formula for $F_a^{(s)}(q,t)$ and prove a theorem that gives recursive relations similar to the recursive relations for regular Fibonacci numbers. We use the standard notation $(s)_r = 1 + r + \cdots + r^{s-1}$.

THEOREM 3.5. *Normalized bigraded Fibonacci numbers $\tilde{F}_a^{(s)}(q,t)$ satisfy the recursive relations*

$$\tilde{F}_{a+1}^{(s)}(q,t) = \tilde{F}_a^{(s)}(q,t) + qt^a\,(s)_{qt^a}\,\tilde{F}_{a-1}^{(s)}(q,t) = \tilde{F}_a^{(s)}(qt,t) + qt\,(s)_{qt}\,\tilde{F}_{a-1}^{(s)}(qt^2,t),$$

with initial conditions $\tilde{F}_0^{(s)}(q,t) = \tilde{F}_1^{(s)}(q,t) = 1$.

Acknowledgments. The author would like to thank Evgeny Gorskiy, Anne Schilling and Tewodros Amdeberhan for suggesting the problem and providing helpful discussions and comments. This research was partially supported by NSF grant DMS-1500050.

3.2. Background and notation

For two coprime numbers a and b consider a rectangle $R_{a,b}$ on the square lattice with bottom-left corner at the origin and top-right corner at (a,b). We call the diagonal from $(0,0)$ to (a,b) the *main diagonal* of the rectangle $R_{a,b}$. An *(a,b)-Dyck path* is a lattice path from $(0,0)$ to (a,b) that consists of North and East steps and that lies weakly above the main diagonal. Denote the set of (a,b)-Dyck paths by $\mathrm{D}_{a,b}$.

For a box in $R_{a,b}$ with bottom-right corner coordinates (x,y), define the *rank of the box* to be equal to $ay-bx$ (see Fig. 3.1, left). Note that a box has positive rank if and only if it lies above the main diagonal. For a rational Dyck path π, we define the area statistic *area*(π) to be the number of boxes in $R_{a,b}$ with positive ranks that are below π.

Denote the set of ranks of all the area boxes of π as $\alpha(\pi)$. Note that $\alpha(\pi)$ does not contain any multiples of a or b and it has an *(a,b)-nested property*, that is,

$$(i \in \alpha(\pi),\ i > a) \Rightarrow i - a \in \alpha(\pi), \quad (j \in \alpha(\pi),\ j > b) \Rightarrow j - b \in \alpha(\pi). \tag{3.1}$$

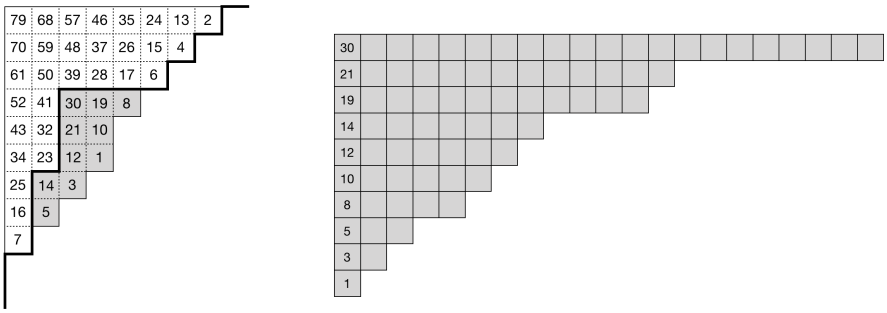


FIGURE 3.1. $(9, 11)$-Dyck path π and $(9, 11)$-core κ with $\mathbf{core}(\pi) = \kappa$.

Also note that $\alpha(\pi)$ completely determines the Dyck path π.

REMARK 3.6. *The (a, b)-nested property of $\alpha(\pi)$ is equivalent to the (a, b)-invariant property of the complement of $\alpha(\pi)$ (see [**GMV16**]).*

Consider $i \in \{0, \ldots, a-1\}$. If we can find a column in $R_{a,b}$ with a box of rank i, define $e_i(\pi)$ to be the number of boxes in that column below π and above the main diagonal. If there is no box of rank i (i.e. if $b|i$), define $e_i(\pi)$ to be zero. Note that

$$e_i(\pi) = \Big|\{x \in \alpha(\pi) \mid x \equiv i \pmod a\}\Big|.$$

The vector $e(\pi) = (e_0(\pi), \ldots, e_{a-1}(\pi))$ is defined to be the *area vector* of π. Note that $e_0(\pi)$ is always zero.

A *partition* λ of n is a finite non-increasing sequence $(\lambda_1, \lambda_2, \ldots, \lambda_l)$ of positive integers which sum up to n. Integers λ_i are called *parts* of the partition λ and n is called the *size* of the partition. A partition λ is sometimes represented by its Young diagram (we will use English notation) (see Fig. 3.1, right). The *hook length* of a box in the Young diagram of λ is defined to be the number of boxes directly below and directly to the right of the given box, including that box itself.

We say that a partition κ is an (a, b)-*core* if there are no boxes in the Young diagram of κ with hook length equal to a or b. Denote the set of all (a, b)-cores as $\mathrm{K}_{a,b}$. For example, a partition $\kappa = (21, 13, 12, 8, 7, 6, 5, 3, 2, 1)$ belongs to $\mathrm{K}_{9,11}$ (see Fig. 3.1).

Define the size statistic on cores $\mathrm{size}(\kappa)$ to be the sum of all parts of κ. Given an (a, b)-core κ, denote the set of hook lengths of the boxes in the first column by $\beta(\kappa)$. It is not hard to show that $\beta(\kappa)$ also satisfies (a, b)-nested property (3.1) (in fact, the nested condition of $\beta(\kappa)$ and the fact that $\beta(\kappa)$ does not contain

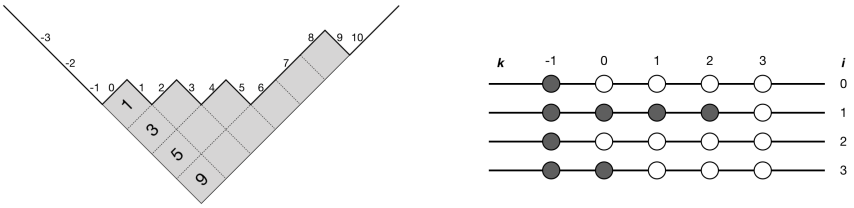


FIGURE 3.2. $(4, 13)$-core $(6, 3, 2, 1)$ and the corresponding abacus diagram with $d = (0, 3, 0, 1)$.

multiples of a or b are sufficient for κ to be an (a, b)-core). Note that the set $\beta(\lambda)$ completely determines the partition λ.

For two coprime numbers a and b, there is a bijection **path**: $\mathrm{K}_{a,b} \to \mathrm{D}_{a,b}$ due to J. Anderson [**And02**]. We can describe the map **path** by specifying how it acts on sets α and β, namely $\alpha(\mathbf{path}(\kappa)) = \beta(\kappa)$.

We also denote the map **core**: $\mathrm{D}_{a,b} \to \mathrm{K}_{a,b}$ by $\mathbf{core} = \mathbf{path}^{-1}$. It follows that $\beta(\mathbf{core}(\pi)) = \alpha(\pi)$.

Define the corresponding area statistic on cores as $area(\kappa) := area(\mathbf{path}(\kappa))$. Note that $area(\kappa) = |\alpha(\mathbf{path}(\kappa))| = |\beta(\kappa)|$ is equal to the number of rows in partition κ. Similarly, an *area vector* of κ is defined as $c(\kappa) = (c_0(\kappa), \ldots, c_{a-1}(\kappa))$ with $c_i(\kappa)$ equal to the number of elements in $\beta(\kappa)$ with residue i modulo a.

In addition to Dyck paths and cores, we will also use the notion of abacus diagrams. An *a-abacus diagram* consists of a rows called runners indexed by $\{0, 1, \ldots, a-1\}$. Each row represents a real line with integer positions filled with black or white beads (see Fig. 3.2, right).

There is a correspondence between partitions and abacus diagrams. Let λ be the Young diagram of a partition in Russian notation with each row going in North-East direction (see Fig. 3.2, left). Reading from left to right, the boundary of λ (denoted by $\partial(\lambda)$) consists of NE-steps and SE-steps. We enumerate the steps so that the first NE-step has number 0, and the enumeration is consecutively increasing for the steps going from left to right.

A corresponding a-abacus diagram is constructed by filling position k on runner i with a black bead if the step $i + ak$ of $\partial(\lambda)$ is an SE-step and filling that position with a white bead otherwise. According to our construction all negative positions on runners are filled with black beads and the bead in position 0 on runner 0 is always white. The importance of that construction becomes clear when we let partition λ be an a-core.

Proposition 3.7 and Corollary 3.8 are well-known results, and we only include proofs for useful observations and notation.

Proposition 3.7. *Partition λ is an a-core if and only if for any SE-step j in $\partial(\lambda)$ step $j - a$ is also an SE-step.*

Proof. Given a box x of a Young diagram in Russian notation of partition λ, there is an SE-step of $\partial(\lambda)$ directly in the NE direction of x with number $se(x)$ and there is an NE-step of $\partial(\lambda)$ directly in the SE direction of x with number $ne(x)$. The hook length of box x is then given by $se(x) - ne(x)$.

($\Leftarrow$) If for every SE-step j the step $j - a$ is also an SE-step, there is no box x with $se(x) - ne(x) = a$ and λ is an a-core.

($\Rightarrow$) Conversely, for any pair of SE-step j and NE-step k with $j > k$, there exists a box x with $se(x) = j$ and $ne(x) = k$. Thus, if there are no boxes x of hook length a, there are no pairs (j, k) with $j - k = a$, and for any SE-step j the step $j - a$ must be also SE. □

Corollary 3.8. *A partition λ is an a-core if and only if the set of black beads in the corresponding a-abacus diagram is left-justified, i.e. for any runner i there exist an integer $d_i \geq 0$ such all positions $k < d_i$ are filled with black beads and all positions $k \geq d_i$ are filled with white beads.*

Proof. Remember that a black bead of an abacus diagram of λ on a runner i in position k corresponds to an SE-step $j = i + ak$ of $\partial(\lambda)$.

Note that an abacus diagram is left-justified if and only if for any black bead on runner i in position k, the bead in position $k - 1$ is also a black one. In turn, that is equivalent to the fact that for any SE-step $j = i + ak$ of $\partial(\lambda)$ the step $j - a = i + a(k - 1)$ is also an SE-step. □

Given integers d_i from Corollary 3.8, denote $d = (d_0, \ldots, d_{a-1})$. It is useful to think about d_i as a number of black beads in nonnegative positions of a runner i. We will call d the *abacus vector* of an a-core λ.

Denote the map from a-cores λ to integer a-dimensional abacus vectors d by **abac**(λ).

Proposition 3.9. *Given an (a, b)-core κ, the area vector $c(\kappa)$ is equal to the a-abacus vector* **abac**(κ).

Proof. Let **abac**(κ) $= (d_0, \ldots, d_{a-1})$. We will make use of notation from Proposition 3.7. Let $\{x_1, \ldots, x_l\}$ be the set of boxes in the first column of the Young diagram of κ. Then $ne(x_i) = 0$ for any i and the set $\{se(x_1), \ldots, se(x_l)\}$ covers all positive SE-steps of $\partial(\kappa)$.

The hook length of x_i is thus equal to $se(x_i) - ne(x_i) = se(x_i)$, and the set of hook lengths of $\{x_1, \ldots, x_l\}$ is equal to $\{se(x_1), \ldots, se(x_l)\}$.

By definition, the set of hook lengths of $\{x_1, \ldots, x_l\}$ is $\beta(\kappa)$. The number of elements in $\beta(\kappa)$ with residue i modulo a is equal to the number of positive SE-steps in $\partial(\kappa)$ with residue i modulo a, so $c_i(\kappa) = d_i$ and $c(\kappa) = d$. □

Corollary 3.10. *Given $\kappa \in \mathrm{K}_{a,b}$, the abacus vector $(d_1, \ldots, d_{a-1}) = \mathbf{abac}(\kappa)$ is equal to the area vector $e(\pi)$ of the Dyck path $\pi = \mathbf{path}(\kappa)$.*

From our definition d_0 is always equal to 0, and thus we often omit that coordinate. For an arbitrary a-core λ the only condition on the other coordinates $(d_1, \ldots, d_{a-1})$ is that they are all non-negative. It is often useful to think about $(d_1, \ldots, d_{a-1})$ as coordinates of an a-core in $\mathbb{Z}^{a-1}$.

Remark 3.11. *Embedding of the set of (a, b)-cores in $\mathbb{Z}^{a-1}$ has been studied from the point of view of Ehrhart theory in [**Joh**].*

To describe the area statistic in terms of vectors d, notice that the number of rows in λ is equal to the number of positive SE-steps in $\partial(\lambda)$, which correspond to non-negative black beads in the a-abacus diagram. Thus, we define the area statistic of d to be $area(d) = \sum d_i$.

3.3. Simultaneous cores with distinct parts.

Let us consider an a-core λ. We can express the condition that λ has distinct parts in terms of the boundary $\partial(\lambda)$.

Proposition 3.12. *An a-core λ has distinct parts if and only if for each positive SE-step j in $\partial(\lambda)$, the steps $j-1$ and $j+1$ are NE-steps.*

Proof. Suppose there are two consecutive SE-steps $j+1$ and j in $\partial(\lambda)$. Then there are two rows row_i and row_{i+1} in the Young diagram of λ that are bordered by steps $j+1$ and j from NE-side. Then $\lambda_i = \mathrm{length}(row_i)$ and $\lambda_{i+1} = \mathrm{length}(row_{i+1})$ are equal, and thus we arrive to a contradiction. □

That property can be formulated in terms of vectors $d = \mathbf{abac}(\lambda)$. Given a subset S of $\{0, 1, \ldots, a-1\}$, we call S an *a-sparse set* if for any two elements $n, m \in S$, $|n - m| \neq 1$ and $0 \notin S$. Whenever size a of the superset is not relevant, we will refer to S as just a *sparse set*.

The *support* of the vector d (denoted by $\mathrm{supp}(d)$) is defined to be the set of all indexes i with $d_i > 0$. We call vectors $d = (d_0, \ldots, d_{a-1})$ with a-sparse support to be *a-sparse vectors*.

PROPOSITION 3.13. *An a-core λ has distinct parts if and only if the vector $d = \mathbf{abac}(\lambda)$ is an a-sparse vector.*

PROOF. ($\Rightarrow$) Consider an index $i \in \{1, \ldots, a-1\}$ such that $d_i > 0$. For the corresponding a-abacus diagam that means the bead on runner i in position 0 is a black one, and the i-th step of $\partial(\lambda)$ is an SE-step. If λ has distinct parts, that means steps $i-1$ and $i+1$ of $\partial(\lambda)$ are NE-steps and thus runners $i-1$ and $i+1$ have no black beads on nonnegative positions, and $d_{i-1} = d_{i+1} = 0$.

($\Leftarrow$) Conversely, suppose $d = \mathbf{abac}(\lambda)$ is an a-sparse vector. Any step SE-step j of $\partial(\lambda)$ corresponds to some black bead on runner i in position k. Because of the sparsity, beads on runners $i-1$ and $i+1$ in position k must be white ones, and thus steps $j-1$ and $j+1$ of $\partial(\lambda)$ must be NE-steps. Note that for any SE-step j corresponding to the runner $i = 1$, step to the left of j is on the runner 0 and thus is always an NE-step. Similarly, when $i = a-1$ the step to the right of j is on the runner 0 in positive position and thus is always an NE-step. □

We now consider an additional structure of simultaneous (a, b)-cores κ in terms of their a-abacus diagrams. We use the Proposition 3.7 one more time, but now looking at κ as a b-core.

PROPOSITION 3.14. *Let κ be an a-core with $d = \mathbf{abac}(\kappa)$ and consider positive integer number $b = sa + r$ with $0 \le r < a$. Then κ is a simultaneous (a, b)- core if and only if for any index i between 1 and $a-1$, one of the following is true:*

(1) *$i \ge r$ and $d_i \le d_{i-r} + s$,*

(2) *$i < r$ and $d_i \le d_{i+a-r} + s + 1$.*

PROOF. ($\Rightarrow$) Fix an index $i \in \{1, \ldots, a-1\}$ and consider all black beads on runner i in nonnegative positions $k = 0, \ldots, d_i - 1$. The corresponding SE-steps of $\partial(\kappa)$ enumerated by $j = i,\ i+a, \ldots,\ i+a(d_i-1)$.

From Proposition 3.7, if κ is a b- core, then for any positive SE-step j in $\partial(\kappa)$ the step $j-b$ is also an SE-step.

If $i \ge r$, steps $j - b = (i-r) + a(k-s)$ with $k = 0, \ldots, d_i - 1$ correspond to the black beads on the runner $(i-r)$ in positions $k = -s,\ -s+1, \ldots,\ d_i - s - 1$, so the number of nonnegative black beads on the runner $(i-r)$ is greater or equal to $(d_i - s)$, and thus $d_i \le d_{i-r} + s$.

Similarly, if $i < r$, steps $j - b = (i + a - r) + a(k - s - 1)$ with $k = 0, \ldots, d_i - 1$ correspond to the black beads on the runner $(i + a - r)$ in positions $k = -s - 1,\ -s, \ldots,\ d_i - s - 2$, so the number of nonnegative black beads on the runner $(i + a - r)$ is greater or equal to $(d_i - s - 1)$, and thus $d_i \leq d_{i+a-r} + s + 1$.

($\Leftarrow$) Conversely, consider any SE-step j in $\partial(\lambda)$ with the corresponding black bead on runner i and position $k < d_i$.

If $i \geq r$, the step $j-b$ corresponds to a bead on runner $(i-r)$ and position $(k-s)$. Since $k-s < d_i-s \leq d_{i-r}$, that bead must be a black one and the step $(j - b)$ is an SE-step.

If $i < r$, the step $j - b$ corresponds to a bead on runner $(i + a - r)$ and position $(k - s - 1)$. Since $k - s - 1 < d_i - s - 1 \leq d_{i+a-r}$, that bead must be a black one and the step $j - b$ is an SE-step.

□

Together with Proposition 3.13, Proposition 3.14 gives a complete description of the simultaneous (a, b)-cores with distinct parts.

3.4. Maximum size of $(a, a+1)$-cores with distinct parts.

Combining Proposition 3.13 and Proposition 3.14 for the case $b = a + 1$, we get the following result.

THEOREM 3.15. *An a-core κ is an $(a, a+1)$-core with distinct parts if and only if $d = \mathbf{abac}(\kappa)$ has entries $d_i \in \{0, 1\}$ and the support set $\mathrm{supp}(d) = \{i : d_i = 1\}$ is an a-sparse set.*

PROOF. From Proposition 3.13, the fact that κ has distinct parts is equivalent to the sparsity of the set $\mathrm{supp}(d)$. Now, taking $s = 1$ and $r = 1$ in Proposition 3.14, κ is an $a + 1$-core if and only if $d_i \leq d_{i-1} + 1$ for all $i = 1, \ldots a - 1$ (condition $d_0 \leq d_{a-1} + 2$ is always satisfied).

If i is not in $\mathrm{supp}(d)$, then $d_i = 0$ and the equation $d_i \leq d_{i-1} + 1$ is true. If i is in $\mathrm{supp}(d)$, then $d_{i-1} = 0$ because of the sparsity of $\mathrm{supp}(d)$ and so $d_i \leq d_{i-1} + 1$ is equivalent to $d_i = 1$. □

THEOREM 3.16. *The number of $(a, a+1)$-cores with distinct parts is equal to the Fibonacci number F_{a+1}.*

PROOF. By Theorem 3.15, all $(a, a+1)$-cores are in a bijection with a-sparse sets $S = \mathrm{supp}(\mathbf{abac}(\kappa))$. Let the number of a-sparse sets be G_a. Then, depending on whether an element $a - 1$ is in a set, we can divide all a-sparse sets into two classes, so that $G_a = G_{a-1} + G_{a-2}$ and $G_1 = 1,\ G_2 = 2$. Thus $G_a = F_{a+1}$. □

H.Xiong [**Xioa**] proved Theorem 3.16 together with conjectures about the largest size of $(a, a+1)$-cores with distinct parts and the number of such cores of maximal size (see Theorem 3.1). Here we provide another proof, which it is formulated in a different framework and which also will be useful for our future discussion.

THEOREM 3.17. *The largest size of an* $(a, a+1)$*-core with distinct parts is* $\lfloor \frac{1}{3}\binom{a+1}{2} \rfloor$. *Moreover, the core of maximal size is unique whenever* $(a \mod 3)$ *is* 0 *or* 2 *and there are two cores of maximal size when* $(a \mod 3)$ *is* 1.

PROOF. Given an $(a, a+1)$-core κ with distinct parts, take $d = \mathbf{abac}(\kappa)$, $\mathrm{supp}(d) = S = \{i_1, \ldots, i_n\}$, where $n = |\mathrm{supp}(d)|$ and indexes $0 < i_1 < \ldots < i_n < a$. Denote the gaps between i_j and i_{j+1} as $g'_0 = i_1$, $g'_j = i_{j+1} - i_j - 1$ for $j = 1, \ldots, n-1$ and $g'_n = a - 1 - i_n$.

Since S is an a-sparse set, $g'_j \geq 1$ for $j = 0, \ldots, n-1$, and thus we can instead consider nonnegative integer sequence $g_j = g'_j - 1$ for $j = 0, \ldots, n-1$ and $g_n = g'_n$. Notice that $\sum_{j=0}^{n} g'_j = a - n$ and $\sum_{j=0}^{n} g_j = a - 2n$.

LEMMA 3.18. *Given* $(a, a+1)$*-core* κ *with distinct parts,*

$$\mathrm{size}(\kappa) = \frac{1}{6} 3n(2a + 1 - 3n) - \sum_{j=0}^{n} j g_j. \tag{3.2}$$

PROOF. Following the construction of an abacus diagram (see Fig. 3.2), each row row_{n-j+1} of the partition κ is bordered by an SE-step $se_j \in \partial(\kappa)$, which in turn corresponds to a black bead on the runner i_j of the abacus diagram.

The length of that row κ_{n-j+1} is determined by the number of NE-steps in $\partial(\kappa)$ before the step se_j. In the abacus diagram, those NE-steps would correspond to the white beads on runners $i = 0, \ldots, i_j - 1$ in position 0, and the number of those white beads is equal to $\sum_{k=1}^{j-1} g'_k$.

Summing over all j,

$$\mathrm{size}(\kappa) = \sum_{j=1}^{n} \kappa_{n-j+1} = \sum_{j=1}^{n} \sum_{k=0}^{j-1} g'_k = \sum_{j=0}^{n} (n-j) g'_j = n \sum_{j=0}^{n} g'_j - \sum_{j=0}^{n} j g'_j =$$

$$= n(a-n) - \frac{n(n-1)}{2} - \sum_{j=0}^{n} j g_j = \frac{1}{6} 3n(2a + 1 - 3n) - \sum_{j=0}^{n} j g_j.$$

□

Thus, to find a core of largest size, we maximize over n and all nonnegative integer sequences $\{g_j\}_{j=0}^n$ with $\sum g_j = a - 2n$.

$$\max_{\kappa} \text{size}(\kappa) = \max_{n} \max_{g_j \geq 0} \Big(\frac{1}{6} 3n(2a+1-3n) - \sum_{j=0}^{n} j g_j \Big) = \max_{n} \Big(\frac{1}{6} 3n(2a+1-3n) - \min_{g_j \geq 0} \sum_{j=0}^{n} j g_j \Big).$$

The minimum of $\sum j g_j$ over nonnegative sequences $\{g_j\}_{j=0}^n$ with $\sum g_j = a - 2n$ is equal to 0 and is uniquely achieved when $g_0 = a - 2n$ and $g_j = 0$ for $j \neq 0$. Thus,

$$\max_{\kappa} \text{size}(\kappa) = \max_{n} \Big(\frac{1}{6} 3n(2a+1-3n) \Big).$$

The parabola on the right-hand side has a rational supremum point at $\frac{2a+1}{6}$. Therefore, a value of n that maximizes the function over integers is an integer point that has the minimal distance to $\frac{2a+1}{6}$.

(1) When $(a \mod 3) = 0$, the maximum is achieved at the unique point $n = \frac{a}{3}$ and the value of the maximum is $\frac{1}{3}\binom{a+1}{2}$.

(2) When $(a \mod 3) = 2$, the maximum is achieved at the unique point $n = \frac{a+1}{3}$ and the value of the maximum is $\frac{1}{3}\binom{a+1}{2}$.

(3) When $(a \mod 3) = 1$, there are two integer points equally close to a number $\frac{2a+1}{6}$, which are $n_1 = \frac{a-1}{3}$ and $n_2 = \frac{a+2}{3}$. Both integers give the maximum value equal to $\frac{1}{3}\frac{(a-1)(a+2)}{2} = \lfloor \frac{1}{3}\binom{a+1}{2} \rfloor$.

□

In Section 3.6 we give a generalization of the Proposition 3.15 and provide another proof of part (4) of Theorem 3.1. Before we do that, however, we need to develop a notion of graded Fibonacci numbers in Section 3.6.

3.5. Number of $(2k-1, 2k+1)$-cores with distinct parts.

The number of $(2k-1, 2k+1)$-cores with distinct parts was observed by T. Amdeberhan [**Amd**] and A. Straub [**Str16**] to be 2^{2k-2}. That conjecture has been proved by Yan, Qin, Jin and Zhou in [**YQJZ17**] using surprisingly deep arguments. Here, we present another perspective on $(2k-1, 2k+1)$-cores using their connection with Dyck paths.

THEOREM 3.19. *The number of $(2k-1, 2k+1)$-cores with distinct parts is equal to 2^{2k-2}.*

79	68	57	46	35	24	13	2	1	10	19	28	37	46
70	59	48	37	26	15	4	3	12	21	30	39	48	57
61	50	39	28	17	6	5	14	23	32	41	50	59	68
52	41	30	19	8	7	16	25	34	43	52	61	70	79

FIGURE 3.3. The rectangle P_5 with a C-symmetric path ζ. Here, $B(\zeta) = \{8\}$, $A^T(\zeta) = \{1, 3, 5, 10, 12, 14, 21\}$ and $C(\zeta) = C^T(\zeta) = \{19, 30\}$. Also, $i(\zeta) = 1$ and $j(\zeta) = 0$.

PROOF. We will use the Dyck path interpretation of cores. For a Dyck path π, we denoted $\alpha(\pi)$ to be the set of ranks of the area boxes of π (see Fig. 3.1). We will also make use of a standard notation $[n] = \{1, 2, \ldots, n\}$.

LEMMA 3.20. *Under the bijection* **path**, *the set of* $(2k-1, 2k+1)$*-cores with distinct parts maps to the set of* $(2k-1, 2k+1)$*-Dyck paths* π *such that* $\alpha(\pi) \cap [2k-1]$ *is a* $(2k-1)$*-sparse set.*

PROOF OF THE LEMMA. Given an $(2k-1, 2k+1)$- core κ with distinct parts and $\pi = \mathbf{path}(\kappa)$, the sparse set $\beta(\kappa)$ is equal to $\alpha(\pi)$. Therefore, we need to prove that the sparsity of $\alpha(\pi)$ is implied by the $(2k-1)$-sparsity of $\alpha(\pi) \cap [2k-1]$.

Suppose for the sake of contradiction that $\alpha(\pi) \cap [2k-1]$ is $(2k-1)$-sparse and there are two elements j and $j+1$ in $\alpha(\pi)$. Since $\alpha(\pi)$ is a $2k-1$-nested set, elements $(j \bmod 2k-1)$, $(j+1 \bmod 2k-1)$ are in $\alpha(\pi)$, they are both in $[2k-1]$, and they differ by one (since there are no multiples of $2k-1$ in $\alpha(\pi)$). Therefore, we get a contradiction. □

Denote T_k to be the upper triangle of $R_{2k-1,2k+1}$, i.e. T_k consists of all boxes with positive rank in $R_{2k-1,2k+1}$ (see Fig. 3.1). Separate T_k into three parts by a vertical line $x = k-1$ and a horizontal line $y = k+2$. Below the line $y = k+2$ the boxes of T_k form a staircase-like shape A that contains, among other boxes, the boxes of odd contents from $[2k-1]$. To the right of the line $x = k-1$ the boxes of T_k form a staircase shape B that contains the boxes of even contents from $[2k-1]$. Above $y = k+2$ and to the left of $x = k-1$ the boxes of T_k form a square shape C of size $k-1$.

Now we reflect the shape $A \cup C$ over the main diagonal $y = x$ and denote the resulting shape as $A^T \cup C^T$. Put that shape to the right of $C \cup B$ to form a rectangular region $P_k = C \cup B \cup A^T \cup C^T$ (see Fig. 3.3).

We call a boundary between B and A^T to be the main diagonal of P_k. Consider the paths ζ from the SW corner of P_k to the NE corner of P_k consisting of N and E steps. Denote $C(\zeta)$ to be the set of contents of

boxes below ζ in C, denote $B(\zeta)$ to be the set of contents of boxes below ζ in B, denote $A^T(\zeta)$ to be the set of contents of boxes *above* ζ in A^T and denote $C^T(\zeta)$ to be the set of contents of boxes *above* ζ in C^T.

We call ζ to be C-symmetric when $C(\zeta) = C^T(\zeta)$ (see Fig. 3.3 and compare with Fig. 3.1).

LEMMA 3.21. *The set of C-symmetric paths ζ in P_k is in bijection ϕ with the set of $(2k-1, 2k+1)$-Dyck paths with $(2k-1)$-sparse $\alpha(\pi) \cap [2k-1]$. Moreover, $\alpha(\phi(\zeta)) = C(\zeta) \cup B(\zeta) \cup A^T(\zeta)$.*

PROOF OF THE LEMMA. We can define ϕ by the property above: $\phi(\zeta) = \pi$ if and only if $\alpha(\pi) = C(\zeta) \cup B(\zeta) \cup A^T(\zeta)$. First, we need to make sure the map ϕ is well-defined, i.e. the set $\gamma(\zeta) := C(\zeta) \cup B(\zeta) \cup A^T(\zeta)$ is a $(2k-1, 2k+1)$-nested set (i.e. check conditions (3.1)).

Let $i \in \gamma(\zeta)$. If $i \in B(\zeta)$ or $i \in A^T(\zeta)$, conditions (3.1) are satisfied since $B(\zeta)$ and $A^T(\zeta)$ are nested sets by construction. If $i \in C(\zeta) = C^T(\zeta)$, then $i - (2k+1) \in C(\zeta) \cup B(\zeta)$, since $C(\zeta) \cup B(\zeta)$ is $(2k+1)$-nested and $i - (2k-1) \in A^T(\zeta) \cup C^T(\zeta)$ since $A^T(\zeta) \cup C^T(\zeta)$ is $(2k-1)$-nested.

Second, we need to check that $\alpha(\phi(\zeta)) \cap [2k-1] = \gamma(\zeta) \cap [2k-1]$ is $(2k-1)$-sparse. Consider $i \in (B(\zeta) \cup A(\zeta)) \cap [2k-1]$, and assume without loss of generality that $i \in B(\zeta)$. Then i is even and it is bordering odd boxes $i-1$ and $i+1$ from E and S directions. Since ζ goes above the box i, it can not go below boxes $i-1$ and $i+1$ in A^T, and thus $i-1, i+1 \notin A^T(\zeta)$. □

Now we want to count the number of C-symmetric paths ζ in P_k. If $C(\zeta)$ is non-empty, call C-shape of ζ to be the shape of the diagram under ζ in C, and C^T-shape of ζ is defined correspondingly. Denote $i(\zeta)$ to be the the width of the C-shape of ζ minus 1 (or the height of C^T-shape minus 1). Denote $j(\zeta)$ to be the height of the C-shape of ζ minus 1 (or the width of C^T-shape minus 1).

The number of C-symmetric paths with fixed $i(\zeta) = i$ and fixed $j(\zeta) = j$ is the number of possible paths in C times the number of possible paths in $B \cup A^T$, which is equal to $\binom{i+j}{i}\binom{k+1+(k-3-i-j)}{k+1}$. If $C(\zeta)$ is empty, the number of paths is equal to $\binom{k+1+(k-1)}{k+1}$. Thus, the total number of paths is

$$\binom{2k}{k+1} + \sum_{i,j\geq 0} \binom{i+j}{i}\binom{2k-2-(i+j)}{k+1} = \binom{2k}{k+1} + \sum_{i\geq 0}\sum_{j'\geq 0} \binom{j'}{i}\binom{2k-2-j'}{k+1} =$$

$$= \binom{2k}{k+1} + \sum_{i\geq 0} \binom{2k-1}{k+2+i} = \binom{2k-1}{k} + \binom{2k-1}{k+1} + \sum_{i\geq 0} \binom{2k-1}{k+2+i} =$$

$$= \sum_{k\leq i'\leq 2k-1} \binom{2k-1}{i'} = \frac{1}{2} \sum_{0\leq i'\leq 2k-1} \binom{2k-1}{i'} = 2^{2k-2}.$$

□

3.6. Graded Fibonacci numbers and $(a, as + 1)$-cores with distinct parts.

Unfortunately, for general b there is no easy way to combine Proposition 3.13 and Proposition 3.14. However it can be achieved for specific values of b.

THEOREM 3.22. *Let κ be an a-core and $b = as+1$ for some integer s. Then κ is an (a, b)-core with distinct parts if and only if the abacus vector $d = \mathbf{abac}(\kappa)$ is a-sparse and $d_i \leq s$ for $i = 1, \ldots, a - 1$.*

PROOF. Similar to the proof of Theorem 3.15, we use Proposition 3.13 to get the sparsity of $\mathrm{supp}(d)$. Moreover, using Proposition 3.14 with $r = 1$ we see that κ is an (a, b)-core if and only if $d_i \leq d_{i-1} + s$ for all $i = 1, \ldots a - 1$ (condition $d_0 \leq d_{a-1} + s + 1$ is automatically satisfied since $d_0 = 0$ for any abacus vector d).

If i is not in $\mathrm{supp}(d)$, then $d_i = 0$ and the inequality $d_i \leq d_{i-1} + s$ is true. If i is in $\mathrm{supp}(d)$, then $d_{i-1} = 0$ because of the sparsity of $\mathrm{supp}(d)$ and so $d_i \leq d_{i-1} + s$ is equivalent to $1 \leq d_i \leq s$. □

We use similar argument for the case $b = as - 1$.

THEOREM 3.23. *Let κ be an a-core and $b = as-1$ for some integer s. Then κ is an (a, b)-core with distinct parts if and only if the abacus vector $d = \mathbf{abac}(\kappa)$ is a-sparse, $d_i \leq s$ for $i \neq a - 1$ and $d_{a-1} \leq s - 1$.*

PROOF. Again, we use Proposition 3.13 to get the sparsity of $\mathrm{supp}(d)$, and use Proposition 3.14 with $r = a - 1$ to get inequalities $d_i \leq d_{i+1} + s$ for $i = 0, \ldots, a - 2$ and $d_{a-1} \leq d_0 + (s - 1)$.

If i is not in $\mathrm{supp}(d)$, then $d_i = 0$ and the inequality $d_i \leq d_{i+1} + s$ is true. If i is in $\mathrm{supp}(d)$ and $i \neq a - 1$, then $d_{i+1} = 0$ because of the sparsity of $\mathrm{supp}(d)$ and so $d_i \leq d_{i-1} + s$ is equivalent to $1 \leq d_i \leq s$. If $i = a - 1$ and i is in $\mathrm{supp}(d)$, note that d_0 is always 0, and thus $d_{a-1} \leq d_0 + (s - 1)$ equivalent to $1 \leq d_{a-1} \leq s - 1$. □

For the further analysis we will need a generating function of the area statistic of cores κ. We will call that function to be a graded Fibonacci number.

DEFINITION 3.24. *For two integers a and b, the graded Fibonacci number is*

$$F_{a,b}(q) = \sum_{\kappa} q^{area(\kappa)}, \tag{3.3}$$

where the sum is taken over all (a, b)-cores κ with distinct parts.

Remark 3.25. *If the sum above was taken over all (a,b)-cores, we would have obtained a graded Catalan number (see [**Loe05**]).*

Remark 3.26. *We do not require a and b to be coprime. Despite the fact that the sum would be infinite, the power series would converge for $|q| < 1$. For the further analysis of Catalan numbers with a, b not coprime, see [**GMV**].*

Remark 3.27. *When we set $s \to \infty$, the set of $(a, as+1)$-cores coveres the set of all a-cores with distinct parts. Thus we will also be interested in the limit of $F_a^{(s)}$ when $s \to \infty$.*

In the light of Theorem 3.22 from here and until the end of the paper we will only consider the case $b = as + 1$ (although all results that follow are applicable in the case $b = as - 1$ with minor modifications). To shorten the notation, we define $F_a^{(s)} = F_{a,as+1}$. It is helpful to rewrite the sum (3.3) in terms of vectors $d = \mathbf{abac}(\kappa)$. Denote the set of all a-sparse vectors $d = (d_0, \dots, d_{a-1})$ with $d_i \le s$ as $\mathcal{A}_a^{(s)}$.

Theorem 3.28.

$$F_a^{(s)}(q) = \sum_{d \in \mathcal{A}_a^{(s)}} q^{\sum d_i}. \tag{3.4}$$

Proof. From Proposition 3.22, the map $\mathbf{abac}\colon \kappa \to d$ is a bijection from the set of all (a,b)-cores κ with distinct parts to the set of a-sparse vectors $d = (d_0, \dots, d_{a-1})$ with $d_i \le s$, i.e. the set $\mathcal{A}_a^{(s)}$.

Also note that bijection **abac** sends the *area* statistic of κ to the sum $\sum_{i=0}^{a-1} d_i$, since the number of rows in κ is equal to the number of positive SE-steps of $\partial(\kappa)$, which in turn is equal to the number of nonnegative black beads in the abacus diagram of κ.

Thus,

$$F_a^{(s)}(q) = \sum_{\kappa} q^{area(\kappa)} = \sum_{d = \mathbf{abac}(\kappa)} q^{\sum d_i} = \sum_{d \in \mathcal{A}_a^{(s)}} q^{\sum d_i}.$$

□

Justification of the term "graded Fibonacci numbers" comes from the proposition below. We will use a standard notation $(s)_q = 1 + q + \dots + q^{s-1} = \frac{1-q^s}{1-q}$.

Theorem 3.29. *Graded Fibonacci numbers $F_a^{(s)}(q)$ satisfy recurrence relation*

$$F_a^{(s)}(q) = F_{a-1}^{(s)}(q) + q\,(s)_q\, F_{a-2}^{(s)}(q) \tag{3.5}$$

with initial conditions $F_0^{(s)}(q) = F_1^{(s)}(q) = 1$.

Proof. We divide the sum in (3.4) into two parts: one over vectors d with $a-1 \notin \operatorname{supp}(d)$, and the other over vectors d with $a-1 \in \operatorname{supp}(d)$.

$$F_a^{(s)}(q) = \sum_{d\in\mathcal{A}_a^{(s)}} q^{\sum d_i} = \sum_{\substack{d\in\mathcal{A}_a^{(s)}\\ d_{a-1}=0}} q^{\sum d_i} + \sum_{\substack{d\in\mathcal{A}_a^{(s)}\\ d_{a-1}\neq 0}} q^{\sum d_i} =$$

$$= \sum_{d\in\mathcal{A}_{a-1}^{(s)}} q^{\sum d_i} + \sum_{d_{a-1}=1}^{s} q^{d_{a-1}} \sum_{d\in\mathcal{A}_{a-2}^{(s)}} q^{\sum d_i} = F_{a-1}^{(s)}(q) + q\,(s)_q\, F_{a-2}^{(s)}(q).$$

For initial conditions, notice that $F_1^{(s)}(q) = 1$, since there is only one $(1, s+1)$-core, which is empty. The number of $(2, 2s+1)$-cores with distinct parts is equal to $s+1$, with corresponding $d_1 \in \{0, 1, \ldots, s\}$, and thus $F_2^{(s)}(q) = 1 + q\,(s)_q$.

Following the recurrence we proved above, we can set $F_0^{(s)}(q) = 1$ for all s. □

We can now present another proof of Theorem 3.4 as a corollary of Theorem 3.29 by setting $q = 1$.

Corollary 3.30. *The number of* $(a, as+1)$*-cores with distinct parts is equal to* $F_a^{(s)}(1)$ *and satisfies the recursive relation*

$$F_a^{(s)}(1) = F_{a-1}^{(s)}(1) + sF_{a-2}^{(s)}(1), \qquad F_0^{(s)}(1) = F_1^{(s)}(1) = 1.$$

In particular, $F_a^{(1)}(1) = F_{a+1}$ *is a classical Fibonacci number.*

Remark 3.31. *In the limit* $s \to \infty$*, relation* (3.5) *has the form*

$$F_a^{(\infty)}(q) = F_{a-1}^{(\infty)}(q) + \frac{q}{1-q} F_{a-2}^{(\infty)}(q).$$

In light of Remark 3.27, relation above is the recurrence for the generating function of area(κ) *over all a-cores with distinct parts.*

Theorem 3.32.

$$F_a^{(s)}(q) = \sum_{n=0}^{\lfloor a/2\rfloor} \left(q\,(s)_q\right)^n \binom{a-n}{n}. \tag{3.6}$$

Proof. For a fixed a-sparse support set $\mathbf{S} = \mathrm{supp}(d)$, the sum in (3.4) is equal to

$$\sum_{\mathrm{supp}(d)=\mathbf{S}} q^{\sum d_i} = \prod_{i\in\mathbf{S}} \sum_{d_i=1}^{s} q^{d_i} = \left(q\,(s)_q\right)^{|\mathbf{S}|}. \tag{3.7}$$

For fixed $n = |\mathbf{S}|$, the number of possible a-sparse support sets $\mathbf{S}$ is the number n-element subsets of $\{1, 2, \ldots, a-1\}$ such that no two elements are neighboring each other. The number of such subsets is equal to $\binom{a-n}{n}$.

Summing (3.7) over all $\mathbf{S}$,

$$F_a^{(s)}(q) = \sum_{\mathbf{S}} \left(q\,(s)_q\right)^{|\mathbf{S}|} = \sum_{n=0}^{\lfloor a/2\rfloor} \sum_{|\mathbf{S}|=n} \left(q\,(s)_q\right)^n = \sum_{n=0}^{\lfloor a/2\rfloor} \left(q\,(s)_q\right)^n \binom{a-n}{n}.$$

□

Theorem 3.33. *The generating function for $F_a^{(s)}(q)$ with respect to a is*

$$G^{(s)}(x;q) := \sum_{a=0}^{\infty} x^a F_a^{(s)}(q) = \frac{1}{1 - x - q\,(s)_q\, x^2}. \tag{3.8}$$

Proof. We use the recurrence (3.5).

$$\begin{aligned} G^{(s)}(x;q) = \sum_{a=0}^{\infty} x^a F_a^{(s)}(q) &= 1 + x + \sum_{a=2}^{\infty} x^a F_a^{(s)}(q) = \\ &= 1 + x + \sum_{a=2}^{\infty} x^a F_{a-1}^{(s)}(q) + \sum_{a=2}^{\infty} x^a q\,(s)_q\, F_{a-2}^{(s)}(q) = \\ &= 1 + x + x\sum_{a=1}^{\infty} x^a F_a^{(s)}(q) + q\,(s)_q\, x^2 \sum_{a=0}^{\infty} x^a F_a^{(s)}(q) = \\ &= 1 + x + x\left(G^{(s)}(x;q) - 1\right) + q\,(s)_q\, x^2 G^{(s)}(x;q) = \\ &= 1 + \left(x + q\,(s)_q\, x^2\right) G^{(s)}(x;q). \end{aligned}$$

Thus

$$G^{(s)}(x;q) = \frac{1}{1 - x - q\,(s)_q\, x^2}.$$

□

REMARK 3.34. *In the limit $s \to \infty$,*

$$F_a^{(\infty)}(q) = \sum_{n=0}^{\lfloor a/2 \rfloor} \left(\frac{q}{1-q}\right)^n \binom{a-n}{n}, \qquad G^{(\infty)}(x,q) = \frac{1}{1 - x - \frac{q}{1-q}x^2}.$$

Now we consider the case $s = 1$ to give a proof of part (4) of Theorem 3.1.

REMARK 3.35. *When $s = 1$,*

$$F_a^{(1)}(q) = \sum_{n=0}^{\lfloor a/2 \rfloor} q^n \binom{a-n}{n}, \qquad G^{(1)}(x,q) = \frac{1}{1 - x - qx^2}, \tag{3.9}$$

and

$$F_a^{(1)}(q) = F_{a-1}^{(1)}(q) + qF_{a-2}^{(1)}(q), \qquad F_0^{(1)}(q) = F_1^{(1)}(q) = 1. \tag{3.10}$$

THEOREM 3.36. *The total sum of the sizes and the average size of $(a, a+1)$-cores with distinct parts are, respectively, given by*

$$\sum_{i+j+k=a+1} F_iF_jF_k \quad \text{and} \quad \sum_{i+j+k=a+1} \frac{F_iF_jF_k}{F_{a+1}}. \tag{3.11}$$

PROOF. Denote Φ_a to be the total sum of sizes of $(a, a+1)$-cores with distinct parts. Since the generating function of Fibonacci numbers $\sum_{i=1}^{\infty} x^iF_i$ is equal to $\frac{x}{1-x-x^2}$, then in order to prove the theorem it is enough to show that the generating function $\Gamma(x) := \sum_{a=2}^{\infty} x^{a+1}\Phi_a$ is equal to

$$\Gamma(x) \stackrel{?}{=} \sum_{a=2}^{\infty} x^{a+1} \sum_{i+j+k=a+1} F_iF_jF_k = \left(\sum_{i=1}^{\infty} x^iF_i\right)^3 = \left(\frac{x}{1-x-x^2}\right)^3. \tag{3.12}$$

We use the equation (3.2) to find a formula for Φ_a.

$$\Phi_a = \sum_{\kappa} \text{size}(\kappa) = \sum_n \sum_{\substack{g \\ \sum g_i = a-2n}} \left[\frac{1}{6}3n(2a+1-3n) - \sum_{i=0}^{n} ig_i\right] = \tag{3.13}$$

$$= \sum_n \left[\binom{a-n}{n}\frac{n(2a+1-3n)}{2} - \sum_{\substack{g \\ \sum g_i = a-2n}} \sum_{i=0}^{n} ig_i\right]$$

To evaluate the double sum, we notice that taking λ to be a partition with $\lambda = (1^{g_1}2^{g_2}\dots n^{g_n})$,

$$\sum_{\sum g_i = a-2n} \sum_{i=0}^{n} i g_i = \sum_{\substack{\lambda_1 \le n \\ l(\lambda) \le a-2n}} |\lambda| =$$

$$= (\text{number of } \lambda \text{ in a rectangle } n \times (a-2n)) \cdot (\text{average size of } \lambda \text{ in } n \times (a-2n)).$$

Note that the number of partitions that fit rectangle $n \times (a-2n)$ is equal to the number of paths from the bottom-right corner of the rectangle to the top-left corner, and thus is equal to $\binom{a-n}{n}$.

The average size of the partition is equal to half of the area of the rectangle $n \times (a-2n)$ because of the symmetry of partitions. Thus, the average size of λ is equal to $\frac{n(a-2n)}{2}$.

Therefore,

$$\sum_{\sum g_i = a-2n} \sum_{i=0}^{n} i g_i = \binom{a-n}{n} \frac{n(a-2n)}{2}.$$

Thus, the sum in (3.13) evaluates to

$$\Phi_a = \sum_n \binom{a-n}{n} \frac{n\,(a-(n-1))}{2} = \frac{a}{2} \sum_n \binom{a-n}{n} n - \frac{1}{2} \sum_n \binom{a-n}{n} n(n-1).$$

Comparing it with (3.9), the sum simplifies to

$$\Phi_a = \frac{a}{2} F_a'(1) - \frac{1}{2} F_a''(1), \tag{3.14}$$

where the function $F_a(q)$ is a short-hand notation for $F_a^{(1)}(q)$, and the derivative F' is taken with respect to q (note that $F_0'(1) = F_1'(1) = 0$). Using the expression for a generating function $G^{(1)}(x;q)$ in (3.9),

$$\Gamma(x) = \sum_{a=2}^{\infty} x^{a+1} \Phi_a \overset{(3.14)}{=} \frac{a}{2} \sum_{a=1}^{\infty} x^{a+1} F_a'(1) - \frac{1}{2} \sum_{a=0}^{\infty} x^{a+1} F_a''(1) =$$

$$= \frac{x^2}{2} \sum_{a=1}^{\infty} a x^{a-1} F_a'(1) - \frac{x}{2} \sum_{a=0}^{\infty} x^a F_a''(1) = \frac{x^2}{2} \frac{\partial^2 G^{(1)}(x;1)}{\partial x \partial q} - \frac{x}{2} \frac{\partial^2 G^{(1)}(x;1)}{\partial q^2} \overset{(3.9)}{=}$$

$$\overset{(3.9)}{=} \frac{x^2}{2} \frac{2x(1-x-x^2) - 2x^2(-1-2x)}{(1-x-x^2)^3} - \frac{x}{2} \frac{2x^4}{(1-x-x^2)^3} = \frac{x^3}{(1-x-x^2)^3}.$$

□

3.7. Bounce statistic and bigraded Fibonacci numbers.

In light of Remark 3.25, we can look at the summand of the bigraded Catalan numbers corresponding to the set of (a, b)-cores with distinct parts. There is a definition of bigraded Catalan numbers in terms of (a, b)-cores directly (see [**ALW16**]), but the skew length statistic has rather complicated expression in terms of abacus vectors d. Nevertheless, we can use another pair of statistics on the set of (a, b)-Dyck paths in the case $b = as + 1$, namely *area* and *bounce*.

To define the *bounce* statistic of a $(a, as + 1)$-Dyck path π, first we present the construction of a bounce path for a Dyck path π due to N.Loehr [**Loe05**].

We start at the point $(a, as+1)$ of the rectangle $R_{a,as+1}$ and travel in West direction until we hit an N-step (North step) of π. Denote v_1 to be the number of W-steps we did in the process, and travel $w_1 := v_1$ steps in the South direction.

After that we travel in West direction until we hit an N-step of π again. Denote v_2 to be the number of W-steps made this time, and travel in South direction $w_2 := v_2 + v_1$ steps if $s > 1$ or $w_2 := v_2$ steps if $s = 1$.

In general, on k-th iteration, after we travel $v_k \geq 0$ steps in West direction before hitting an N-step of π, we then travel in S direction $w_k := v_k + \ldots + v_{k-s+1}$ steps if $k \geq s$ or $w_k := v_k + \ldots + v_1$ steps if $k < s$. The bounce path always stays above the main diagonal and eventually hits the point $(0, 1)$, where the algorithm terminates (see [**Loe05**] for details).

To calculate the *bounce* statistic of π, each time our bounce path reaches an N-step x of π after traveling West, add up the number of squares to the left of π and in the same row as x. We will call those rows *bounce rows* (see Fig. 3.4).

Definition 3.37. *[**Loe05**] Bigraded rational Catalan number is defined by the equation*

$$C_{a,as+1}(q,t) = \sum_{\pi} q^{area(\pi)} t^{bounce(\pi)}, \tag{3.15}$$

where the sum is taken over all $(a, as + 1)$-Dyck paths π.

Following that definition, we restrict the sum to define bigraded Fibonacci numbers.

Definition 3.38. *Bigraded rational Fibonacci number is defined by the equation*

$$F_a^{(s)}(q,t) = \sum_{\pi} q^{area(\pi)} t^{bounce(\pi)}, \tag{3.16}$$

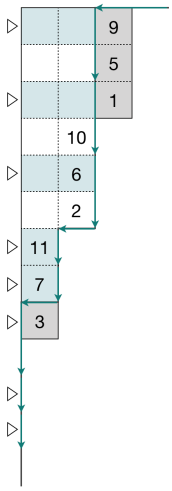


FIGURE 3.4. $(4, 13)$-Dyck path with $\alpha(\pi) = \{1, 3, 5, 9\}$ and $e(\pi) = (3, 0, 1)$ together with its bounce path. The bounce rows are marked by arrows.

where the sum is taken over all $(a, as + 1)$*- Dyck paths such that* $\kappa = \mathbf{core}(\pi)$ *has distinct parts.*

REMARK 3.39. *After the specialization* $t = 1$ *bigraded Fibonacci numbers* $F_a^{(s)}(q, 1)$ *are equal to graded* $F_a^{(s)}(q)$ *from the previous section.*

Under the maps **core** and **abac**, there is a correspondence between the Dyck paths π in (3.16) and abacus vectors $d \in \mathcal{A}_a^{(s)}$. Denote the bounce statistic on $\mathcal{A}_a^{(s)}$ to be a bounce statistic of the corresponding Dyck path π, i.e. $bounce(\mathbf{abac}(\mathbf{core}(\pi))) = bounce(\pi)$.

THEOREM 3.40. *Given an abacus vector* $d \in \mathcal{A}_a^{(s)}$,

$$bounce(d) = s\binom{a}{2} - \sum_{i=1}^{a-1}(a-i)d_i. \tag{3.17}$$

PROOF. Let π be the corresponding Dyck path, i.e. $\mathbf{abac}(\mathbf{core}(\pi)) = d$. It is easier to consider the statistic $bounce'(\pi) = s\binom{a}{2} - bounce(\pi)$. Here $s\binom{a}{2}$ counts the total number of boxes in the upper triangle $T_{a,as+1}$, and thus $bounce'(\pi)$ counts the number of boxes in non-bounce rows to the left of π plus the number of area boxes of π. Thus, we need to show

$$bounce'(d) \stackrel{?}{=} \sum_{i=1}^{a-1}(a-i)d_i. \tag{3.18}$$

We will prove (3.18) by induction on a. Base case $a = 1$ is straightforward. Assume now that (3.18) is true for any $a \le k$ and consider the case $a = k + 1$.

According to Corollary 3.10, the area vector $e(\pi) = (d_1, \dots, d_k)$.

If $d_1 = e_1(\pi) = 0$, the first s iterations of the bounce path algorithm yields values for W-steps $\nu_1 = 1,\ \nu_2 = \dots = \nu_s = 0$ and the corresponding steps in South direction are $w_1 = w_2 = \dots = w_s = 1$, ending at the point $(k, ks + 1)$ (and after that point the values of $\nu_1, \dots,\ \nu_s$ do not contribute to the bounce path).

Thus the top s rows of the upper triangle $T_{k+1,(k+1)s+1}$ are bounce rows and moreover there are no area boxes of π in those rows. Therefore the top s rows do not contribute anything to $bounce'(\pi)$, and we can safely erase them, reducing a by one and reducing all indexes of d by one. Denoting $d' = (d'_0, \dots, d'_{k-1}) = (d_1, \dots, d_k) = e(\pi)$,

$$bounce'(d) = bounce'(d') = \sum_{i=1}^{k-1}(k-i)d'_i = \sum_{i=2}^{k}(k-(i-1))d_i = \sum_{i=1}^{k}((k+1)-i)d_i.$$

If $d_1 = e_1(\pi) > 0$ (see Fig.3.4), then $d_2 = e_2(\pi) = 0$ because of the sparsity of d. The first s iterations of the bounce path algorithm yields values for W-steps $\nu_1 = 1,\ \nu_2 = \dots = \nu_{s-d_1} = 0,\ \nu_{s-d_1+1} = 1,\ \nu_{s-d_1+2} = \dots = \nu_{2s-d_1} = 0$ with the corresponding steps in South direction equal to $w_1 = \dots = w_{s-d_1} = 1,\ w_{s-d_1+1} = \dots = w_s = 2,\ w_{s+1} = \dots = w_{2s-d_1} = 1$, ending at the point $(k-1, (k-1)s+1)$ (and after that point the values of $\nu_1, \dots,\ \nu_{2s-d_1}$ do not contribute to the bounce path).

For the top $2s$ rows of $T_{k+1,(k+1)s+1}$ the non-bounce rows appear exactly when the bounce path travels 2 steps in South direction, i.e. when $w_i = 2$. Thus, there are d_1 non-bounce rows, each contributing $k-1$ boxes to statistic $bounce'$. Besides that, there are d_1 area boxes that also count towards $bounce'$. Thus, the contribution of the top $2s$ rows into $bounce'$ is equal to $d_1 k$. Denoting $d' = (d'_0, \dots, d'_{k-2}) = (d_2, \dots, d_k)$,

$$bounce'(d) = bounce'(d') + d_1 k = \sum_{i=1}^{k-2}(k-1-i)d_{i+2} + d_1 k = \sum_{i=1}^{k}((k+1)-i)d_i.$$

□

Corollary 3.41.

$$t^{-s\binom{a}{2}} F_a^{(s)}(q,t) = \sum_{d \in \mathcal{A}_a^{(s)}} q^{\sum d_i} t^{-\sum (a-i)d_i} = \sum_{d \in \mathcal{A}_a^{(s)}} (qt^{-a})^{\sum d_i} t^{\sum i d_i} = \sum_{d \in \mathcal{A}_a^{(s)}} q^{\sum d_i} t^{-\sum i d_i}. \tag{3.19}$$

Proof. First two equalities follow directly from (3.16) and (3.17). For the last equality we use the symmetry of $\mathcal{A}_a^{(s)}$ under the reflection of indexes $0 \mapsto 0,\ i \mapsto a-i$. □

From (3.19) it is easier to work with normalized polynomials

$$\tilde{F}_a^{(s)}(q,t) := t^{-s\binom{a}{2}} F_a^{(s)}(q,t^{-1}) = \sum_{d \in \mathcal{A}_a^{(s)}} q^{\sum d_i} t^{\sum i d_i} = \sum_{d \in \mathcal{A}_a^{(s)}} (qt^a)^{\sum d_i} t^{-\sum i d_i}. \tag{3.20}$$

Remark 3.42. *In terms of Dyck paths, $\tilde{F}_a^{(s)}(q,t)$ is equal to the sum of $q^{area(\pi)} t^{bounce'(\pi)}$ over $(a, as+1)$-Dyck paths π with* **core**(π) *having distinct parts.*

Using the simple expression of $\tilde{F}_a^{(s)}(q,t)$ in (3.20), we prove recursive relations similar to Proposition 3.29.

Theorem 3.43. *Normalized bigraded Fibonacci numbers $\tilde{F}_a^{(s)}(q,t)$ satisfy the following relations:*

$$\tilde{F}_{a+1}^{(s)}(q,t) = \tilde{F}_a^{(s)}(q,t) + qt^a\,(s)_{qt^a}\,\tilde{F}_{a-1}^{(s)}(q,t) \tag{3.21}$$

$$\tilde{F}_{a+1}^{(s)}(q,t) = \tilde{F}_a^{(s)}(qt,t) + qt\,(s)_{qt}\,\tilde{F}_{a-1}^{(s)}(qt^2,t), \tag{3.22}$$

with initial conditions $\tilde{F}_0^{(s)}(q,t) = \tilde{F}_1^{(s)}(q,t) = 1$.

Proof. For equation (3.21) use the first sum in (3.20) and divide the set $\mathcal{A}_{a+1}^{(s)}$ into two parts corresponding to vectors d with $d_a = 0$ and with $d_a > 0$.

$$\tilde{F}_{a+1}^{(s)}(q,t) = \sum_{d \in \mathcal{A}_{a+1}^{(s)}} q^{\sum d_i} t^{\sum i d_i} = \sum_{d' \in \mathcal{A}_a^{(s)}} q^{\sum d'_i} t^{\sum i d'_i} + \sum_{d_a=1}^{s} (qt^a)^{d_a} \sum_{d' \in \mathcal{A}_{a-1}^{(s)}} q^{\sum d'_i} t^{\sum i d'_i} =$$

$$= \tilde{F}_a^{(s)}(q,t) + qt^a\,(s)_{qt^a}\,\tilde{F}_{a-1}^{(s)}(q,t).$$

For equation (3.22) use the second sum in (3.20) and again divide $\mathcal{A}^{(s)}_{a+1}$ into two parts corresponding to vectors d with $d_a = 0$ and with $d_a > 0$.

$$\tilde{F}^{(s)}_{a+1}(q,t) = \sum_{d\in\mathcal{A}^{(s)}_{a+1}} (qt^{a+1})^{\sum d_i} t^{-\sum i d_i} =$$

$$= \sum_{d'\in\mathcal{A}^{(s)}_{a}} \left((qt)t^{a-1}\right)^{\sum d'_i} t^{-\sum i d'_i} + \sum_{d_a=1}^{s} (qt^{a+1}t^{-a})^{d_a} \sum_{d'\in\mathcal{A}^{(s)}_{a-1}} \left((qt^2)t^{a-2}\right)^{\sum d'_i} t^{-\sum i d'_i} =$$

$$= \tilde{F}^{(s)}_{a}(qt,t) + qt\,(s)_{qt}\,\tilde{F}^{(s)}_{a-1}(qt^2,t)$$

□

Remark 3.44. *Setting $s = 1$ and $a \to \infty$, the recurrence (3.22) gives*

$$\tilde{F}^{(1)}_{\infty}(q,t) = \tilde{F}^{(1)}_{\infty}(qt,t) + qt\tilde{F}^{(1)}_{\infty}(qt^2,t), \tag{3.23}$$

*which is an Andrews q-difference equation related to Rogers-Ramanujan identities (see [**And86**]).*

CHAPTER 4

Estimating Graphlet Statistics via Lifting

4.1. Introduction

In 1970, [Dav70] discovered that transitivity—the tendency of friends of friends to be friends themselves—is a prevalent feature in social networks. Since that early discovery, real-world networks have been observed to have many other common macroscopic features, and these discoveries have led to probabilistic models for networks that display these phenomena. The observation that transitivity and other common subgraphs are prevalent in networks motivated the exponential random graph model (ERGM) [FS86]. [BA99] demonstrated that many large networks display a scale-free power law degree distribution, and provided a model for constructing such graphs. Similarly, the small world phenomenon—that networks display surprisingly few degrees of separation—motivated the network model in [WS98]. While network science is often driven by the observation and modelling of common properties in networks, it is incumbent on the practicing data scientist to explore network data using statistical methods.

One approach to understanding network data is to fit free parameters in these network models to the data through likelihood-based or Bayesian methods. In [WP96], a pseudolikelihood method was used with graphlet counts to fit an ERGM designed to display transitivity, and Monte Carlo Markov Chain (MCMC) methods were developed in [Sni02] for fitting general ERGMs. Fitting such models from data can be cumbersome, and to do so implicitly assumes that the network follows such a model exactly. Network statistics, such as the clustering coefficient, algebraic connectivity, and degree sequence, are more flexible tools. A good statistic can be used to fit and test models, for example, [WS98] used the local clustering coefficient, a measure of the number of triangles relative to wedges, to test if a network is a small-world graph. The clustering coefficient is also used to understand social network graphs, [CF06]. More generally, it was discovered that re-occurring subgraph patterns can be used to differentiate real-world networks, and that genetic networks, neural networks, and internet networks all presented different common interconnectivity

patterns, [MSOI+02]. In this work, we will propose a new method for counting the occurrences of any subgraph pattern, otherwise known as *graphlets*—a term coined in [PCJ04]—or motifs.

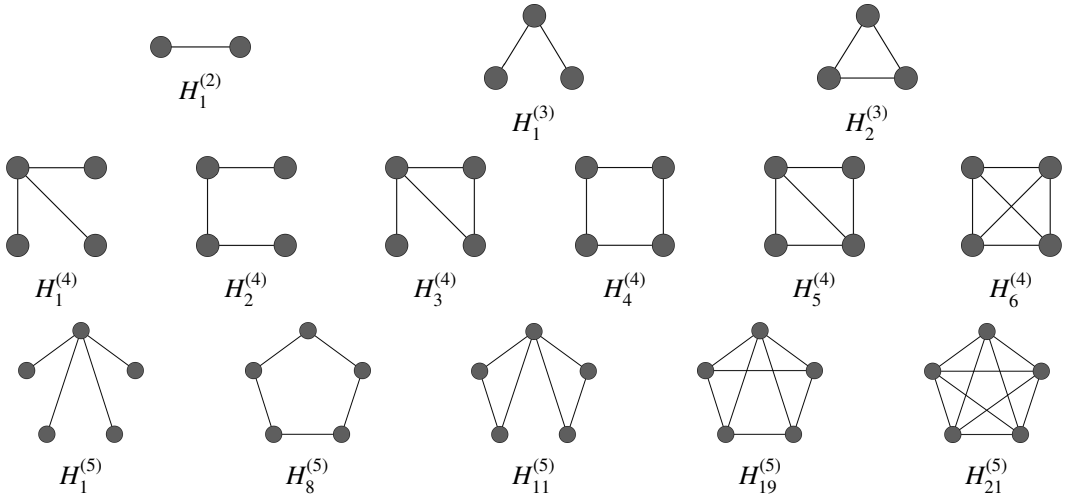


FIGURE 4.1. Examples of graphlets

A graphlet is a small connected graph topology, such as a triangle, wedge, or k-clique, which we will use to describe the local behavior of a larger network (example graphlets of size 3, 4, and 5, can be seen in Figure 4.1). Let the graph in question be $G = (V, E)$ where V is a set of vertices and E is a set of unordered pairs of vertices (G is assumed to be undirected and unweighted). Imagine specifying a k-graphlet and testing for every induced subgraph of the graph (denoted $G|\{v_1, \ldots, v_k\}$ where $v_1, \ldots, v_k \in V$), if it is isomorphic to the subgraph (it has the same topology). We would like to compute the number of Connected Induced Subgraphs of size k (denoted by *k-CIS* throughout) for which this match holds. We call this number the *graphlet counts* and the proportion of the number of such matches to the total number of k-CISs is called the *graphlet coefficient*.

Graphlets are the graph analogue of wavelets (small oscillatory functions that are convolved with a signal to produce wavelet coefficients) because they are small topologies that are matched to induced subgraphs of the original graph to produce the graphlet coefficients. Graphlet coefficients, also referred to as graph moments, are used in the method of moments to fit certain graph models by selecting parameters that match the empirical graph moments to their expectations [BCL+11]. Graphlet coefficients are used to understand biological networks, such as the protein-protein interaction network, and reoccuring patterns are thought to indicate evolutionary conserved modules [PCJ06]. If the graphlet size, k, is small then testing for isomorphism, a problem called graph matching [CFSV04], is feasible, but testing every induced subgraph

can require on the order of n^k iterations in its most naive implementation. We propose a class of methods called lifting that allow us to quickly estimate graphlet coefficients.

While there exist several algorithms that can estimate the proportion of triangles, wedges, and graphlets with four or five vertices (for example, [**ANR+17**, **RBAH14**]), there are few algorithms that can efficiently estimate the proportion of larger graphlets. Many methods that can handle arbitrary graphlets are Monte Carlo sampling procedures that traverse through the space of all graphlets of a certain size within the large network. Two such methods are GUISE algorithm of [**BRAH12**] and the pairwise subgraph random walk of [**WLR+14**], which differ in the way that they perform a random walk between the CIS samples. Another option is to generate a sequence of vertices that induces a CIS sample, which has been done in [**HS16**] using an algorithm called Waddling random walk. Alternatives to random Monte Carlo schemes include the color coding scheme of [**BCK+17**], but it processes the whole graph, while the Monte Carlo schemes can traverse the network locally. In this work, we propose a new Monte Carlo algorithm, called lifting, for estimating the graphlet counts within a large network. The lifting step takes a smaller CIS of size $k-1$ and produces a subgraph of size k by adding an adjacent vertex to it (according to a specific scheme). In this paper we consider procedures that start from vertices or edges and lift to sample CIS of size k. Lifting is a simple, flexible procedure that can be easily distributed to accommodate massive network datasets.

4.1.1. Our contributions. Graphlet coefficients are multipurpose statistical tools that can be used for model fitting and testing, network regression, and exploratory analysis for network data. Any CIS Monte Carlo sampling scheme has three goals: that it provides unbiased estimates of the graphlet coefficients, that the variance of the estimated coefficients is small, and that it does so in as few iterations as possible. Because massive graphs are often stored in distributed databases, we would like the sampling scheme to require only neighborhood queries (requests to the database returns the neighbors of a node) and we will avoid storing or processing the full graph. Because communication is often the bottleneck in distributed computing, neighborhood queries are the basic unit for measuring computational time complexity.

After discussing the precise requirements for any CIS sampling procedure, we will introduce the lifting scheme for subgraphs. The key difficulty in any Monte Carlo method for graphlet counting is calculating the sampling distribution. We provide two methods, the ordered lift estimator and the unordered lift estimator, which differ in the way that subgraphs are represented and counted in the graphlet count. The ordered estimator allows for a modification, called *shotgun sampling* that samples multiple subgraphs in one shot.

For our theoretical component, we prove that the estimated graphlet coefficients are unbiased when the underlying MCMC has reached the stationary distribution (called perfect mixing). We also prove that under perfect mixing, the variance of the estimator scales like Δ^{k-2} where Δ is the maximum degree, and show that the lifting scheme can have significantly lower sample correlations than the subgraph random walk. All proofs can be found in the supplement. We conclude with real-world network experiments that reinforce the contention that subgraph lifting has a lower variance than Waddling and lower sample correlation than subgraph random walks.

4.2. Sampling graphlets

4.2.1. Definitions and notation. Throughout we will assume that our graph $G = (V, E)$ is simple, connected and undirected. For a subset $W \subseteq V$, a subgraph of G induced by W, $G|W$ is a graph with vertices in W and edges in $\binom{W}{2} \cap E$. We call a connected motif on k vertices a k-graphlet. Given a k-graphlet H, we'll be interested in the number of k-subgraphs of G isomorphic to H. The set of all connected induced k-subgraphs (or k-CISs) of G is denoted by $\mathcal{V}_k(G)$ (or simply $\mathcal{V}_k$).

An unordered set of vertices is denoted $\{v_1, \ldots, v_k\}$ while an ordered list is denoted $[v_1, \ldots, v_k]$. Let $H_1, H_2, \ldots, H_l$ be all non-isomorphic motifs for which we would like the graphlet counts. For $T \in \mathcal{V}_k(G)$, we say that "T is subgraph of type m" if T is isomorphic to H_m, and denote this with $T \sim H_m$. The number of k-subgraphs in G of type m is equal to $N_m(G) = \sum_{T \in \mathcal{V}_k(G)} \mathbb{1}(T \sim H_m)$, where $\mathbb{1}(A)$ is the indicator function. For a subgraph $S \subseteq G$, denote V_S to be the set of its vertices, E_S to be the set of its edges. Denote $\mathcal{N}_v(S)$ (vertex neighborhood of S) to be the set of all vertices adjacent to some vertex in S not including S itself. Denote $\mathcal{N}_e(S)$ (edge neighborhood of S) to be the set of all edges that connect a vertex from S and a vertex outside of S. Also, denote $\deg(S)$ (degree of S) to be the number of edges in $\mathcal{N}_e(S)$, and denote $\deg_S(u)$ (S-degree of u) to be the number of vertices from S that are connected to u. Note that $\deg(S) + 2|E_S| = \sum_{v \in V_S} \deg(v)$.

4.2.2. Prior graphlet sampling methods. The ideal Monte Carlo procedure would sequentially sample CISs uniformly at random from the set $\mathcal{V}_k(G)$, classify their type, and update the corresponding counts. Unfortunately, uniformly sampling CISs is not a simple task because they are required to be connected—a random set of k vertices is unlikely to be connected. CIS sampling methods require Monte Carlo Markov

Chains (MCMCs) for which one can calculate the stationary distribution, π, over the elements of $\mathcal{V}_k$. First, let us consider how we update the graphlet counts, $N_m(G)$, given a sample of CISs, $T_1, T_2, \ldots, T_n$.

The desire to sample subgraphs uniformly is natural, because the empirical counts will be unbiased estimates of their population quantities. Instead, suppose that our CIS sample, with $T_i \in \mathcal{V}_k, i = 1, \ldots, n$, is drawn with known sampling distribution π. Then we use Horvitz-Thompson inverse probability weighting to estimate the graphlet counts,

$$\hat{N}_m(G) := \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbb{1}(T_i \sim H_m)}{\pi(T_i)}. \tag{4.1}$$

It is simple to see that this is an unbiased estimate of the graphlet counts as long as π is supported over all elements of $\mathcal{V}_k$. Alternative updating methods include rejection sampling, which can be combined with inverse probability weighting, but we will use (4.1) for ease of presentation.

We find ourselves in a game, where we can choose any CIS Monte Carlo algorithm that induces a stationary distribution π, but we must be able to quickly and accurately compute π in order to use (4.1). We will analyze the theoretical implications of the sampling algorithm based on mixing times of the Markov chain and the variance of the graphlet count estimates. Before we explore the lifting procedure, this paper's algorithmic contribution, we would like to discuss some existing MCMC CIS sampling methods.

The subgraph random walk is described in [**WLR+14**], where they make a modification called the pairwise subgraph random walk (PSRW). In order to perform a random walk where the states are $\mathcal{V}_k$, we form the CIS-relationship graph. Two k-CISs, $T, S \in \mathcal{V}_k$ are connected with an edge if and only if vertex sets of T and S differ by one element, i.e. when $|V(T) \cap V(S)| = k - 1$. Given the graph structure, we sample k-CISs by a random walk on the set $\mathcal{V}_k$, which is called Subgraph Random Walk (SRW). Because the transition from state $S \in \mathcal{V}_k$ is made uniformly at random to each adjacent CIS, then we know that the stationary distribution will sample each edge in the CIS-relationship graph with equal probability. This fact enables [**WLR+14**] to provide a local estimator of the stationary probability $\pi(S)$. PSRW is a modification of the SRW algorithm, where each transition is performed from S to T in $\mathcal{V}_{k-1}$ and then the k-CIS $S \cup T$ is returned.

The mixing time is a critical issue for any MCMC, and subgraph sampling is no exception. Dependence between consecutive samples results in a higher variability of $\hat{N}_m(G)$, and sufficient mixing is required for the stationary distribution to approximate the sampling distribution. It was pointed out in [**BCK+17**] that the

mixing time of the SRW can be of order $O(n^{k-2})$, even if the mixing time of the random walk on the original graph G is of constant order $O(1)$. PSRW also requires global constants based on the CIS-graph, which can be computationally intractable (super-linear time).

Another approach is to sample on ordered sequences of vertices $[v_1, \ldots, v_k]$, denoted by A, which would induce a k-CIS, $T = G|A$. Given a sampling scheme of such sequences with probability $\tilde{\pi}(A)$, the estimator for graphlet counts is given by

$$\hat{N}_m(G) := \frac{\omega_m}{n} \sum_{i=1}^{n} \frac{\mathbb{1}(G|A_i \sim H_m)}{\tilde{\pi}(A_i)} \tag{4.2}$$

for some fixed weights ω_m. The main difference between these types of sampling is that we maintain the ordering of the vertices, while a CIS is an unordered set of vertices.

A naive method for sampling such sequences would be to perform a random walk on the graph G and then sample the k vertices most recently visited. This scheme is appealing because it has an easy to compute stationary distribution, and can 'inherit' the mixing rate from the random walk on G (which is relatively small). Despite these advantages, certain graphlet topologies, such as stars, will never be sampled, and modifications are needed to remedy this defect. [**CLWL16**] combined this basic idea with the SRW by maintaining a l length history of the SRW on CISs of size $k - l + 1$, and unioning the history, but this suffers from the same issues as SRW, such as slow mixing and the need to calculate global constants based on the CIS-graph.

[**HS16**] introduced a *Waddling* protocol which retains a memory of the last s vertices in the random walk on G and then extends this subgraph by $k - s$ vertices from either the first or last vertex visited in the s-subgraph (this extension is known as the 'waddle'). The authors provide a method for calculating the stationary distribution for this MCMC, and prove a bound on the error for the graphlet coefficients. The downside to this method is that the precise Waddling protocol used should depend on the desired graphlet, and the algorithm involves a rejection step which may lead to a loss of efficiency. In contrast, the lifting scheme has the advantage of inheriting the mixing time of the random walk on G, and it can simultaneously sample many CISs of differing sizes without any rejections.

4.3. Subgraph lifting

The lifting algorithm is based on a randomized protocol of attaching a vertex to a given CIS. For any $(k-1)$-CIS, S we lift it to a k-subgraph by adding a vertex from its neighborhood, $\mathcal{N}_v(S)$ at random according to some probability distribution. Note that this basic lifting operation can explore any possible subgraph in $\mathcal{V}_k$. In this work, we show that one can lift from a random walk on the vertices, or another vertex or edge sampling distribution, and achieve favorable properties.

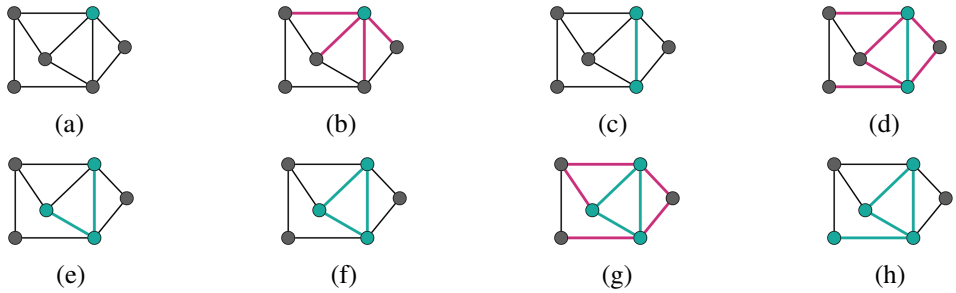


FIGURE 4.2. Lifting procedure

You can see an example of the lifting sampling scheme in Figure 4.2, where the algorithm iteratively builds a 4-CIS from a chosen node. First assume we have a node v_1 sampled from the distribution π_1, a base distribution that can be computed from local information (fig. 4.2a). We assume that $\pi_1(v) = \frac{f(\deg(v))}{K}$, where $f(x)$ is some function (usually a polynomial) and K is some global normalizing constant which is assumed to be precomputed. Denote $S_1 = \{v_1\}$. To start our procedure, sample an edge (v_1, v_2) uniformly from $\mathcal{N}_e(S_1)$ (fig. 4.2b). The vertex v_2 is then attached to S_1, forming a subgraph $S_2 = G|(V_{S_1} + v_2)$ (fig. 4.2c). After that, we sample another edge (v_i, v_3) (with $1 \leq i \leq 2$) uniformly from $\mathcal{N}_e(S_2)$, and the vertex v_3 is then attached to S_2 (fig. 4.2d - 4.2e). At each step we sample an edge (v_i, v_{r+1}) (with $1 \leq i \leq r$) from $\mathcal{N}_e(S_r)$ uniformly at random, and attach the vertex v_{r+1} to the subgraph S_r (fig. 4.2f - 4.2h). After $k-1$ operations, we obtain a k-CIS $T = S_k$. We'll refer to the procedure above as the *lifting procedure starting at vertex* v_1.

By induction, we can see that every k-CIS has a non-zero probability of being visited, assuming that π_1 is supported on every vertex. The Waddling method was motivated by the fact that a simple random walk would not visit all subgraphs in $\mathcal{V}_k$, yet Waddling only partially solved this issue, because not every wadding protocol can sample every k-graphlet. Typically, π_1 is assumed to be the stationary distribution of a simple random walk, but it could be another distribution such as uniform sampling over the vertices or edges. One motivation for lifting is that if we sample the vertices from a simple random walk, then lifting

'inherits' its mixing time, much like Waddling. Hence, lifting is a simple algorithm that can sample any CIS with good mixing rates and no rejections. It remains to be demonstrated that we can calculate the probability of sampling the k-CIS $\pi(S)$ using only local information.

4.3.1. Ordered lift estimator. We can think of a lifting procedure as a way of sampling a sequence $A = [v_1, \dots, v_k]$, ordered from the first vertex sampled to the last, that is then used to generate a CIS. Denote the set of such sequences as V_G^k. Let $S_r = G|\{v_1, \dots, v_r\}$ be the r-CIS obtained by the lifting procedure on step r. The probability of sampling vertex v_{r+1} on the step $r+1$ is equal to

$$\mathbb{P}(v_{r+1}|S_r) := \frac{\deg_{S_r}(v_{r+1})}{|\mathcal{N}_e(S_r)|} = \frac{|E_{S_{r+1}}| - |E_{S_r}|}{\sum_{i=1}^{r} \deg(v_i) - 2|E_{S_r}|}.$$

Thus, the probability of sampling a sequence $A \in V_G^k$ is equal to

$$\tilde{\pi}(A) := \pi_1(v_1) \prod_{r=1}^{k-1} \mathbb{P}(v_{r+1}|S_r) = \frac{f(\deg(v_1))}{K} \prod_{r=1}^{k-1} \frac{|E_{S_{r+1}}| - |E_{S_r}|}{\sum_{i=1}^{r} \deg(v_i) - 2|E_{S_r}|}. \tag{4.3}$$

Critically, this equation can be computed with only neighborhood information about the involved vertices, so it takes $O(k)$ neighborhood queries. Because there are many orderings that could have led to the same CIS T, then we need to apply proper weights in the graphlet count estimate (4.2) by enumerating the number of possible orderings.

Consider the sampled k-CIS $T := S_k$. Denote the set of possible sequences $A = [v_1, \dots, v_k]$ that would form T in the lifting process as $\mathrm{co}(T)$. Notice that $S_r = G|\{v_1, \dots, v_r\}$ must be a connected subgraph for all r. Thus,

$$\mathrm{co}(T) = \{\, [v_1, \dots, v_k] \in V_G^k \mid \{v_1, \dots, v_k\} = V_T, T|\{v_1, \dots, v_r\} \text{ is connected} \,\}. \tag{4.4}$$

Since the elements of $\mathrm{co}(T)$ are just certain orderings of vertices in T, we call an element from $\mathrm{co}(T)$ a *compatible ordering* of T. Note that $|\mathrm{co}(T)|$ only depends on the type of the graphlet isomorphic to T, and it can be precomputed using dynamic programming. Thus, when $T \sim H_m$, the number of compatible orderings are equal: $|\mathrm{co}(H_m)| = |\mathrm{co}(T)|$. Note that $|\mathrm{co}(H_m)|$ can vary from 2^{k-1} (for k-path) to $k!$ (for k-clique). We set up the estimator from (4.2) as

$$\hat{N}_{O,m} := \frac{1}{n} \frac{1}{|\mathrm{co}(H_m)|} \sum_{i=1}^{n} \frac{\mathbb{1}(G|A_i \sim H_m)}{\tilde{\pi}(A_i)}. \tag{4.5}$$

We call it the *ordered lift estimator* for the graphlet count.

Algorithm 1 Ordered Lift Estimator (with optional shotgun sampling)

input Graph G, k-graphlet H_m
output $\hat{N}_m(G)$
  Count $|\text{co}(H_m)|$- the number of compatible orderings in H_m.
  Initialize v at an arbitrary node, $n \leftarrow 0$, $\hat{N}_m(G) \leftarrow 0$
  while stopping criteria is not met **do**
    Sample v_1 from $\pi_1(v)$
    Initialize $V_S \leftarrow \{v_1\}$ and $E_S \leftarrow \{\}$
    Initialize $\mathcal{N}_e(S) \leftarrow \mathcal{N}_e(v_1)$
    Initialize $\pi(S) \leftarrow \pi_1(v_1)$
    while $|V_S| < k - 1$ **do**
      Sample an edge $e = (v, u)$ uniformly from $\mathcal{N}_e(S)$, with $v \in V_S$ and $u \notin V_S$
      Set $E_S(u) \leftarrow \{(v, u) \in \mathcal{N}_e(S)\}$
      Update $\pi(S) \leftarrow \pi(S)\frac{|E_S(u)|}{|\mathcal{N}_e(S)|}$
      Update $V_S \leftarrow V_S \cup \{u\}$ and $E_S \leftarrow E_S \cup E_S(u)$
      Query $\mathcal{N}_e(u)$
      Update $\mathcal{N}_e(S) \leftarrow [\mathcal{N}_e(S) \cup \mathcal{N}_e(u)] \setminus E_S(u)$
    end while
    if not shotgun sampling **then**
      Sample an edge $e = (v, u)$ uniformly from $\mathcal{N}_e(S)$, with $v \in V_S$ and $u \notin V_S$
      Set $E_S(u) \leftarrow \{(v, u) \in \mathcal{N}_e(S)\}$
      Set $\pi(T) \leftarrow \pi(S)\frac{|E_S(u)|}{|\mathcal{N}_e(S)|}$
      Set $V_T \leftarrow V_S \cup \{u\}$ and $E_T \leftarrow E_S \cup E_S(u)$
      if $(V_T, E_T) \sim H_m$ **then**
        Update $\hat{N}_m(G) \leftarrow \hat{N}_m(G) + \pi^{-1}(T)$
      end if
    end if
    if shotgun sampling **then**
      for all $u \in \mathcal{N}_v(S)$ **do**
        Set $E_S(u) \leftarrow \{(v, u) \in \mathcal{N}_e(S)\}$
        Set $V_T \leftarrow V_S \cup \{u\}$ and $E_T \leftarrow E_S \cup E_S(u)$
        if $(V_T, E_T) \sim H_m$ **then**
          Update $\hat{N}_m(G) \leftarrow \hat{N}_m(G) + \pi^{-1}(S)$
        end if
      end for
    end if
    Update $n \leftarrow n + 1$
  end while
  Normalize $\hat{N}_m(G) \leftarrow \frac{1}{n}\frac{1}{|\text{co}(H_m)|}\hat{N}_m(G)$

A drawback of the algorithm is that it takes $k-1$ queries to lift the CIS plus the number of steps required to sample the first vertex (when sampled from Markov chain). To increase the number of samples per query,

notice that if we sample $B = [v_1, \dots, v_{k-1}]$ via lifting, we can get subgraphs induced by $A = [v_1, \dots, v_{k-1}, u]$ for all $u \in \mathcal{N}_v(B)$ without any additional queries.

Thus, for each sampled sequence $B_i \in V_G^{k-1}$, we can compute the sum $\sum_{u \in \mathcal{N}_v(B_i)} \mathbb{1}(G|B_i \cup \{u\} \sim H_m)$ to incorporate the information about all k-CISs in the neighborhood of B_i. We call this procedure *shotgun sampling*. The corresponding estimator based on (4.2) is

$$\hat{N}_{S,m} = \frac{1}{n} \frac{1}{|\mathrm{co}(H_m)|} \sum_{i=1}^{n} \frac{\sum_{u \in \mathcal{N}_v(B_i)} \mathbb{1}(G|B_i \cup \{u\} \sim H_m)}{\tilde{\pi}(B_i)}. \tag{4.6}$$

Shotgun sampling produces more CIS samples with no additional query cost, but the CIS samples generated in a single iteration will be highly dependent. The following proposition states that the resulting estimators are unbiased.

Proposition 4.1. *The ordered lifted estimator, $\hat{N}_{O,m}$, and the shotgun estimator, $\hat{N}_{S,m}$, are unbiased for the graphlet counts N_m.*

4.3.2. Unordered lift estimator. The ordered lift estimators computed the probability of sampling the CIS and the order of included vertices. Alternatively, we can compute the marginal probability of sampling the unordered graphlet, $\pi_U(T)$ for the lifted CIS $T \in \mathcal{V}_k(G)$. One advantage of this approach is that this probability is a function of only the degrees of vertices V_T. This can be done either recursively or directly. Throughout, let the set of vertices of T be $v_1, \dots, v_k$.

We begin the algorithm by querying the probability of obtaining any vertex in T, $\pi_1(v_i), i = 1, \dots, k$. We will build the probability of obtaining any connected subgraph of T inductively. This is possible because the probability of getting T via lifting is given by the sum $\pi_U(T) = \sum_S \mathbb{P}(T|S)\pi_U(S)$, where the sum is taken over all connected $(k-1)$-subgraphs $S \subset T$, and $\mathbb{P}(T|S)$ denotes the probability of getting from S to T in the lifting procedure. Then

$$\pi_U(T) = \sum_{S \subset T} \pi_U(S) \frac{\deg_S(V_T \setminus V_S)}{|\mathcal{N}_e(S)|} = \sum_{S \subset T} \pi_U(S) \frac{|E_T| - |E_S|}{\sum_{u \in S} \deg(u) - 2|E_S|}, \tag{4.7}$$

where the sum is taken over all connected $(k-1)$-subgraphs $S \subset T$.

For a direct formula, we notice that $\pi_U(T) = \sum_{A \in \mathrm{co}(T)} \tilde{\pi}(A)$, where $\mathrm{co}(T)$ is the set of compatible orderings of T from previous section, and $\tilde{\pi}(A)$ is the probability of getting sequence $A \in \mathrm{co}(T)$ in the lifting

process (see (4.4),(4.3)). Then

$$\pi_U(T) = \sum_{A \in \mathrm{co}(T)} \frac{f(\deg(A[1]))}{K} \prod_{r=1}^{k-1} \frac{|E_{S_{r+1}(A)}| - |E_{S_r(A)}|}{\sum_{i=1}^{r} \deg(A[i]) - 2|E_{S_r(A)}|}, \tag{4.8}$$

where, given $A = [v_1, \ldots, v_k]$, $A[i]$ is the ith vertex in A and $S_r(A) = G|\{v_1, \ldots, v_r\}$.

Although calculation of this probability on-the-fly is cost-prohibitive, we can greatly reduce the number of operations by noticing that the probability $\pi_k(T)$ is a function of degrees of the vertices: for a CIS T of type m, let $[v_1, \ldots, v_k]$ be an arbitrary labelling of the vertices of T with $d_i = \deg(v_i)$, then the probability of T is

$$\pi_U(T) = \frac{1}{K} F_m(d_1, \ldots, d_k)$$

for a cached function F_m given by (4.8).

Example. Consider a triangle, which is a 3-graphlet with edges (v_1, v_2), (v_2, v_3) and (v_1, v_3). Given the degrees d_1, d_2, d_3 of the corresponding vertices, the probability function is

$$\begin{aligned} \pi_U(\text{triangle}) = &\left(\frac{\pi_1(d_1)}{d_1} + \frac{\pi_1(d_2)}{d_2}\right)\frac{2}{d_1 + d_2 - 2} + \\ &+ \left(\frac{\pi_1(d_2)}{d_2} + \frac{\pi_1(d_3)}{d_3}\right)\frac{2}{d_2 + d_3 - 2} + \left(\frac{\pi_1(d_3)}{d_3} + \frac{\pi_1(d_1)}{d_1}\right)\frac{2}{d_3 + d_1 - 2}. \end{aligned} \tag{4.9}$$

Example. Consider a wedge, which is a 3-graphlet with edges (v_1, v_2) and (v_1, v_3). Given the degrees d_1, d_2, d_3 of the corresponding vertices, the probability function is

$$\pi_U(\text{wedge}) = \left(\frac{\pi_1(d_1)}{d_1} + \frac{\pi_1(d_2)}{d_2}\right)\frac{1}{d_1 + d_2 - 2} + \left(\frac{\pi_1(d_1)}{d_1} + \frac{\pi_1(d_3)}{d_3}\right)\frac{1}{d_1 + d_3 - 2}. \tag{4.10}$$

We need to only compute functions F_m once before starting the algorithm. When a k-CIS T is sampled via lifting procedure, we find the natural labelling of vertices in T via the isomorphism $H_m \to T$, and use the function F_m together with the degrees $d_1, \ldots, d_k$ of vertices of T to compute the value of $\pi_U(T) = \frac{1}{K} F_m(d_1, \ldots, d_k)$.

4.3.3. Sampling a starting vertex. One advantage of the lifting protocol is that it can be decoupled from the selection of a starting vertex, and our calculations remained agnostic to the distribution π_1 (although, we did require that it was a function of the degrees). There are two method that we would like to

Algorithm 2 Unordered Lift Estimator

input Graph G, k-graphlet H_m
output $\hat{N}_m(G)$

```
Set an ordering [1, ..., k] on the vertices of H_m, precompute the function F_m(d_1, ..., d_k) and the global constant K
Initialize v at an arbitrary node, n ← 0, N̂_m(G) ← 0
while stopping criteria is not met do
    Sample initial vertex v from π_1(v)
    Initialize V_T ← {v} and E_T ← {}
    Initialize N_e(T) ← N_e(v)
    while |V_T| < k do
        Sample an edge e = (v, u) uniformly from N_e(T), with v ∈ V_T and u ∉ V_T
        Set E_T(u) ← {(v, u) ∈ N_e(T)}
        Update V_T ← V_T ∪ {u} and E_T ← E_T ∪ E_T(u)
        Query N_e(u)
        Update N_e(T) ← [N_e(T) ∪ N_e(u)] \ E_T(u)
    end while
    if (V_T, E_T) ~ H_m then
        Determine the ordering [v_1, ..., v_k] of vertices in V_T induced by the isomorphism (V_T, E_T) ~ H_m
        Set d_i = |N_e(v_i)| for all i = 1, ..., k
        Set π(T) = (1/K) F_m(d_1, ..., d_k)
        Update N̂_m(G) ← N̂_m(G) + π^{-1}(T)
    end if
    Update n ← n + 1
end while
Normalize N̂_m(G) ← (1/n) N̂_m(G)
```

consider, one is the uniform selection over the set of vertices, and the other is from a random walk on the vertices, that presumably has reached its stationary distribution.

Consider sampling the starting vertex v independently and from an arbitrary distribution π_1 when we have access to all the vertices. The advantage of sampling vertices independently, is that the lifting process will result in independent CIS samples. A byproduct of this is that the variance of the graphlet count estimator (4.1) can be decomposed into the variance of the individual CIS samples. Given iid draws, the variance of the estimator $\hat{N}_m(G)$ is then

$$V_m^{\perp}(\hat{N}_{U,m}) := \frac{1}{n}\mathrm{Var}\left(\frac{\mathbb{1}(T_n \sim H_m)}{\pi_U(T_n)}\right) = \frac{1}{n}\left(\sum_{T\in\mathcal{V}_k}\frac{\mathbb{1}(T \sim H_m)}{\pi_U(T)} - N_m(G)^2\right), \tag{4.11}$$

which is small when the distribution of $\pi_U(T)$ is close to uniform distribution on $\mathcal{V}_m(G)$. Equation (4.11) demonstrates fundamental property that when $\pi_U(T)$ is small then it contributes more to the variance of the estimator. The variation in (4.11) can be reduced by an appropriate choice of π_1, i.e. the starting distribution.

For example, if $k = 3$, let $\pi_1(v) = \frac{1}{K}\deg(v)(\deg(v) - 1)$, where $K = \sum_{u \in V_G} \deg(u)(\deg(u) - 1)$. Then by (4.9) and (4.10)

$$\pi_U(\text{triangle}) = \frac{6}{K}, \quad \pi_U(\text{wedge}) = \frac{2}{K}.$$

Calculating K takes $O(|V_G|)$ operations (preparation), sampling starting vertex v takes $O(\log(|V_G|))$ operations, and lifting takes $O(\Delta)$, where Δ is the maximum vertex degree in G.

When we don't have access to the whole graph structure, a natural choice is to run a simple random walk (with transitional probabilities $p(i \to j) = \frac{1}{\deg(i)}$ whenever j in connected to i with an edge). Then the stationary distribution is $\pi_1(v) = \deg(v)/(2|E_G|)$, and we can calculate all probabilities π_k accordingly. One feature of the simple random walk is that the resulting edge distribution is uniform: $\pi_U(e) = \frac{1}{|E_G|}$ for all $e \in E_G$ (edges are 2-graphlets). Therefore, the probabilities π_U are the same as if sampling an edge uniformly at random and start lifting procedure from that edge.

4.4. Theoretical Analysis of Lifting

As long as the base vertex distribution, π_1, is accurate then we have that the graphlet counts are unbiased for each of the aforementioned methods. The variance of the graphlet counts will differ between these methods and other competing algorithms such as Waddling and PSRW. The variance of sampling algorithms can be decomposed into two parts, an independent sample variance component and a between sample covariance component. As we have seen the independent variance component is based on the properties of π resulting from the procedure (see (4.11)). The covariance component will be low if the samples are not highly dependent on the past, namely that the Markov chain is well mixed. We have three different estimators: Ordered Lift estimator $\hat{N}_{O,m}$, Shotgun Lift estimator $\hat{N}_{S,m}$ and Unordered Lift estimator $\hat{N}_{U,m}$. For each estimator, we sample different objects: sequences $A_i \in V_G^k$ for Ordered, sequences $B_i \in V_G^{k-1}$ for Shotgun, and CISs $T_i \in \mathcal{V}_k(G)$ for Unordered estimator. Throughout this section, we will denote

(1) for the Ordered Lift estimator,

$$\phi_{O,i} = \frac{\mathbb{1}(G|A_i \sim H_m)}{|\text{co}(H_m)|\tilde{\pi}(A_i)}, \tag{4.12}$$

(2) for the Shotgun Lift estimator,

$$\phi_{S,i} = \frac{\sum_{u \in \mathcal{N}_v(B_i)} \mathbb{1}(G|B_i \cup \{u\} \sim H_m)}{|\mathrm{co}(H_m)|\tilde{\pi}(B_i)}, \tag{4.13}$$

(3) for the Unordered Lift estimator,

$$\phi_{U,i} = \frac{\mathbb{1}(T_i \sim H_m)}{\pi_U(T_i)}. \tag{4.14}$$

Note that $N_m(G) = \mathbb{E}\phi_1$, and $\hat{N}_m(G) = \frac{1}{n}\sum_i \phi_i$ for the corresponding estimators.

The variance can be decomposed into the independent sample variance and a covariance term,

$$\mathrm{Var}(\hat{N}_m(G)) = \frac{1}{n} V_m^{\perp\!\!\!\perp}(\phi_1) + \frac{2}{n^2} \sum_{i<j} \mathrm{Cov}\left(\phi_i, \phi_j\right). \tag{4.15}$$

For Markov chains, the summand in the second term will typically decrease exponentially as the lag $j - i$ increases, due to mixing. Let us focus on the first term, with the goal of controlling this for either choice of base vertex distribution, π_1, and the lifting scheme.

THEOREM 4.2. *Let ϕ_1 be as defined in (4.12), (4.13) or (4.14). Denote the first k highest degrees of vertices in G as $\Delta_1, \ldots, \Delta_k$ and denote $D = \prod_{r=2}^{k-1}(\Delta_1 + \ldots + \Delta_r)$.*

(1) If π_1 is the stationary distribution of the vertex random walk then

$$V_m^{\perp\!\!\!\perp}(\phi_1) \leq N_m(G) \frac{2|E_G|}{|\mathrm{co}(H_m)|} D. \tag{4.16}$$

(2) If π_1 is the uniform distribution over the vertices then

$$V_m^{\perp\!\!\!\perp}(\phi_1) \leq N_m(G) \frac{2\Delta_1|E_G|}{|\mathrm{co}(H_m)|} D. \tag{4.17}$$

This result is comparable to analogous theorems for Waddling, [**HS16**], and PSRW, [**WLR+14**].

When the vertices are sampled independently, the covariance term disappears, so we will focus on the sampling vertices via random walk in this subsection. One advantage of the lifting procedure over the SRW is that it inherits the mixing properties from the vertex random walk. This can be thought of as a consequence of the data processing inequality in that the lifted CISs are no more dependent then the starting vertices from which they were lifted. To that end, let us review some basics about mixing of Markov chains,

Definition 4.3. *Define the mixing coefficient of a stationary Markov chain with discrete state space $X_t \in \mathcal{X}$ as*

$$\gamma_X(h) = \frac{1}{2} \max_{x_1 \in \mathcal{X}} \sum_{x_2 \in \mathcal{X}} |\mathbb{P}(X_{t+h} = x_2, X_t = x_1) - \pi(x_1)\pi(x_2)|, \tag{4.18}$$

where $\pi(x)$ is the stationary distribution of the Markov chain. Also, define the mixing time of a stationary Markov chain $\{X_t\}$ as

$$\tau_X(\varepsilon) = \min\{\, h \mid \gamma_X(h) < \varepsilon \}. \tag{4.19}$$

Theorem 4.4. ***[Sin92]*** *Given stationary Markov chain $\{X_t\}$ with $\mu < 1$ being the second largest eigenvalue of the transitional matrix,*

$$\gamma_X(h) \leq e^{-(1-\mu)h}. \tag{4.20}$$

There are two consequences of mixing for CIS sampling. First, an initial burn-in period is needed for the distribution π to converge to the stationary distribution (and for the graphlet counts to be unbiased). Second, by spacing out the samples with intermediate burn-in periods and only obtaining CISs every h steps we can reduce the covariance component of the variance of $\hat{N}_m$. Critically, if we wish to wait for h steps, we do not need to perform the lifting scheme in the intervening iterations, since those graphlets will not be counted. So, unlike in other MCMC method, spacing in lifted CIS sampling is computationally very inexpensive. Because burn-in is a one-time cost and requires only a random walk on the graph, we will suppose that we begin sampling from the stationary distribution, and the remaining source of variation is due to insufficient spacing between samples. The following theorem illustrates the point that the lifted MCMC inherits mixing properties from the vertex random walk.

Theorem 4.5. *Consider sampling a starting vertex from a random walk, such that a sufficient burn in period has elapsed and stationarity has been reached. Let h be the spacing between the CIS samples, D be defined as in Theorem 4.2, and μ be the second largest eigenvalue of the transition matrix for the vertex random walk. Let ϕ_i be as defined in (4.12), (4.13) or (4.14), then*

$$|\mathrm{Cov}\,(\phi_i, \phi_{i+1}))| \leq 8N_m(G)|E_G|^2 e^{-(1-\mu)h} D.$$

COROLLARY 4.6. *In the notation of the Theorem 4.5,*

$$\frac{2}{n}\left|\sum_{i<j}\mathrm{Cov}\left(\phi_i,\phi_j\right)\right| \leq 8N_m(G)|E_G|^2\frac{e^{-(1-\mu)h}}{1-e^{-(1-\mu)h}}D.$$

Hence, if we allow h to grow large enough then we can reduce the effect of the covariance term, and our CISs will seem as if they are independent samples.

4.5. Experiments

For our experiments, we picked five networks of different size, density and domain. All networks are available online in the network repository [**RA15**]. The corresponding graphs are undirected, and were preprocessed to be simple and unweighted.

Network information		
Network name	$\lvert V_G\rvert$	$\lvert E_G\rvert$
bio-celegansneural	297	2148
ia-email-univ	1133	5451
misc-polblogs	1224	16718
misc-as-caida	26475	52281
misc-fullb	199187	5754445

We demonstrate performance of the shotgun and unordered lift method compared to two competitive methods: PSRW and Waddling. All algorithms were implemented in Python, and the code is available on GitHub[1].

We will compare the effectiveness of estimators in two ways:

(1) Relative error of a graphlet count estimator with $k = 3$ (wedges $H_1^{(3)}$ and triangles $H_2^{(3)}$) given limited number of queries (or, equivalently, limited number of samples).

(2) Variance and correlation of samples for a graphlet count estimator for 3-stars $H_1^{(4)}$ and 4-paths $H_2^{(4)}$.

In all methods, the first vertex of the lifting procedure is taken from a standard random walk on vertices. Whenever we specify the burn-in, that would correspond to the random walk steps taken to sample the starting vertex in case of Lifting and Waddling, and the number of steps taken between sampling CISs in

[1] github.com/KirillP23/LiftSRW

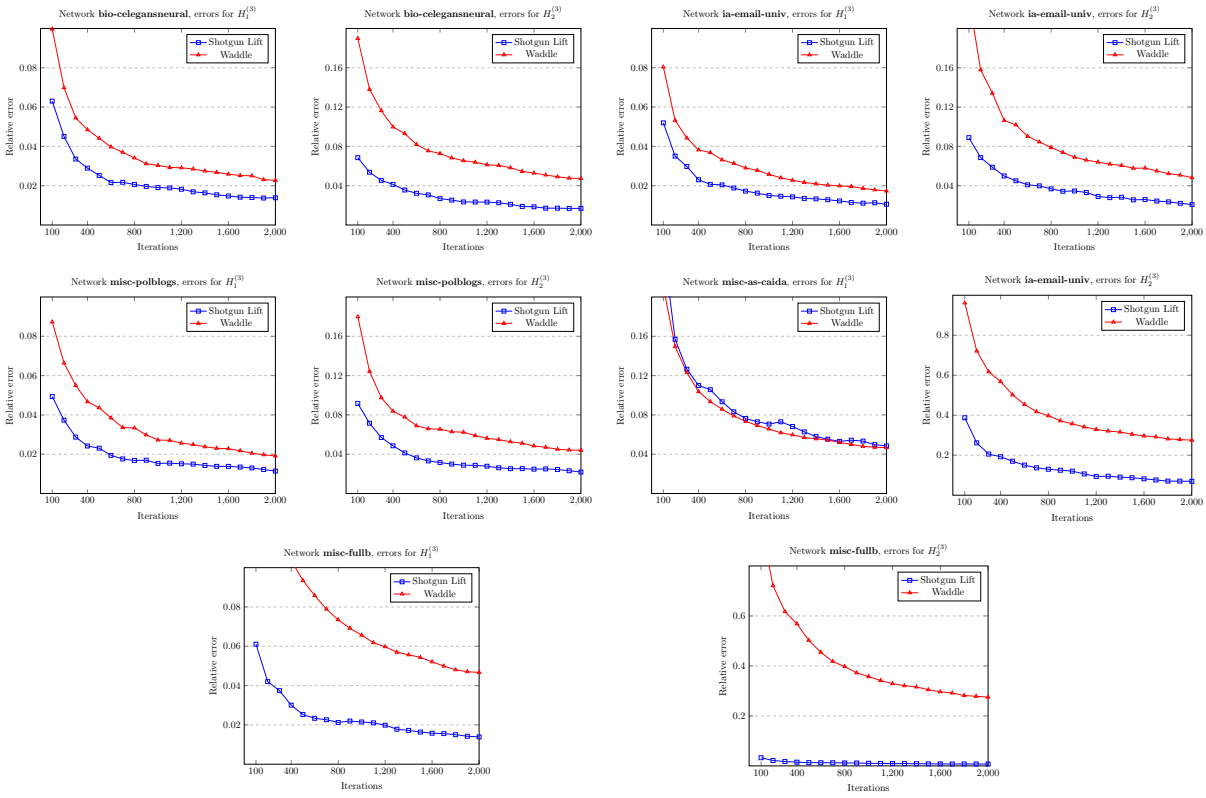


FIGURE 4.3. Relative errors of count estimators for $H_1^{(3)}$ and $H_2^{(3)}$.

case of PSRW. We compare the Shotgun Lift estimator against Waddling estimator for the first problem, and compare the Unordered Lift estimator against PSRW and Waddling for the second problem.

4.5.1. Relative Error given limited queries. The brute force method gives us the exact graphlet count for $k = 3$:

Graphlet counts for $k = 3$		
Network name	$H_1^{(3)}$	$H_2^{(3)}$
bio-celegansneural	4.408E+04	3.241E+03
ia-email-univ	8.038E+04	5.343E+03
misc-polblogs	1.038E+06	1.010E+05
misc-as-caida	1.479E+07	3.636E+04
misc-fullb	1.620E+08	6.021E+07

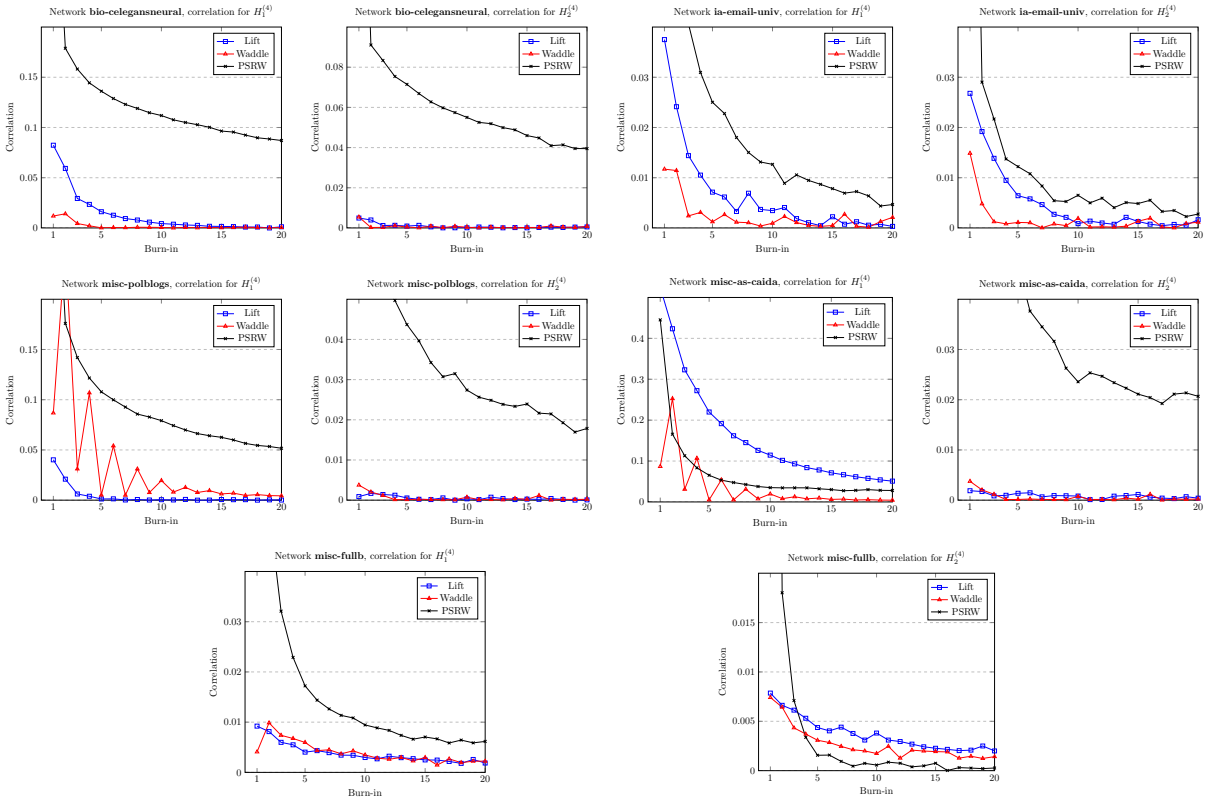


FIGURE 4.4. Correlation of ϕ_i and ϕ_{i+1} depending on the intermediate burn-in time, h, between samples for graphlets $H_1^{(4)}$ and $H_2^{(4)}$.

We will compare the relative error

$$\text{relative error} = \frac{|\hat{N}_m(G) - N_m(G)|}{N_m(G)}$$

between Shotgun Lift estimator and Waddling estimator with burn-in 3. The number of queries required for each sample $B \in V_G^{(2)}$ is 4 for the Shotgun Lift algorithm and the number of queries for each 3-CIS sample is 5 for the Waddling algorithm. We take the average error across 100 runs.

As we see from the plots in Figure 4.3, Shotgun Lift method outperforms Waddling by a factor of 2 in most cases. This agrees with our theory, since the number of actual CISs sampled by the Shotgun method is much bigger than Waddle given the same number of queries.

4.5.2. Variation and correlation of samples. We will use the Unordered Lift estimator, to count all motifs of size 4, but the performance would be measured in terms of the variance of the estimator, both the

"variance under independence" and "covariance" parts. To get an idea about the distribution of 4-graphlets, we count the 4-graphlets with the Unordered Lift estimator.

Graphlet estimates for $k = 4$			
Network name	$H_1^{(4)}$	$H_2^{(4)}$	$H_3^{(4)}$
bio-celegansneural	6.48E+05	5.16E+05	1.86E+05
ia-email-univ	5.40E+05	1.11E+06	2.20E+05
misc-polblogs	3.94E+07	3.10E+07	1.58E+07
misc-as-caida	7.82E+09	2.85E+08	4.62E+07
misc-fullb	1.07E+09	4.83E+09	2.71E+09

Denote $\phi_i = \frac{\mathbb{1}(T_i \sim H_m)}{\pi(T_i)}$ for all estimators. Note that $\mathbb{E}\phi_1 = N_m(G)$. We compare the variance of the estimator $\hat{N}_m$ using equation (4.15). The table below shows estimates for $V_m^{\perp\!\!\!\perp}(\phi_1)$:

Variation under independence				
Network Name	ID	Lift	Waddle	PSRW
bio-celegansneural	$H_1^{(4)}$	6.876	11.064	0.652
	$H_2^{(4)}$	5.470	3.956	5.185
ia-email-univ	$H_1^{(4)}$	5.266	4.164	1.179
	$H_2^{(4)}$	3.272	2.958	2.216
misc-polblogs	$H_1^{(4)}$	5.033	7.210	0.820
	$H_2^{(4)}$	6.214	5.369	6.093
misc-as-caida	$H_1^{(4)}$	4.076	9.028	0.019
	$H_2^{(4)}$	106.06	153.11	78.87
misc-fullb	$H_1^{(4)}$	21.617	10.185	6.179
	$H_2^{(4)}$	6.014	4.041	3.785

We can see that PSRW generally has smaller variation under independence, mostly because of the probability $\pi(T)$ being independent of the degrees of vertices of T. On the other hand, variation under independence of Lift and Waddle methods is comparable, meaning that $\pi(T)$ varies similarly for those two methods. Next, we compare the dependence of ϕ_i and ϕ_{i+1} using correlation for different values of the burn-in h (see Fig.4.4). For Lift and Waddling, the burn-in between ϕ_i and ϕ_{i+1} is the number of steps taken after sampling T_i to get a new starting vertex for T_{i+1}. For PSRW, burn-in is the number of steps between CIS samples in the random walk on subgraphs. From the graphs in Figure 4.4, we see that PSRW produces highly correlated samples compared to Lift and Waddling methods. This agrees with our analysis of PSRW, since it takes many more steps for the subgraph random walk to achieve desired mixing compared to the random walk on vertices.

4.6. Supplement to "Estimating Graphlets via Lifting"

4.6.1. Proof of Prop. 4.1.

Proof. Let ϕ_i be as defined in (4.12), (4.13). For both estimators, because of the form of (4.5) and (4.6), if a single term ϕ_i is unbiased then $\hat{N}_m$ is as well. Let us begin with $\hat{N}_{O,m}$, by considering a draw from the lifting process, $A = [v_1, \ldots, v_k]$ which induces the k-subgraph, $G|A$. By the definition of $\tilde{\pi}$,

$$\mathbb{E}(\phi_{O,1}) = \sum_{A \in V_G^k} \tilde{\pi}(A) \left(\frac{\mathbb{1}(T(A) \sim H_m)}{\mathrm{co}(T(A))\tilde{\pi}(A)} \right) = \sum_{T \in \mathcal{V}_k} \sum_{A \in V_G^k : T(A) = T} \frac{\mathbb{1}(T \sim H_m)}{\mathrm{co}(T)} = \sum_{T \in \mathcal{V}_k} \mathbb{1}(T \sim H_m) = N_m.$$

Hence, the $\hat{N}_{O,m}$ is unbiased. Consider the shotgun estimator, $\hat{N}_{S,m}$,

$$\begin{aligned}\mathbb{E}(\phi_{S,1}) &= \sum_{B \in V_G^{k-1}} \tilde{\pi}(B) \sum_{u \in \mathcal{N}_v(B)} \left(\frac{\mathbb{1}(G|B \cup \{u\} \sim H_m)}{\mathrm{co}(H_m)\tilde{\pi}(B)} \right) \\ &= \sum_{T \in \mathcal{V}_k} \sum_{B \in V_G^{k-1}} \mathbb{1}(G|B \cup \{u\} = T, u \in \mathcal{N}_v(B)) \frac{\mathbb{1}(T \sim H_m)}{\mathrm{co}(T)} = \sum_{T \in \mathcal{V}_k} \mathbb{1}(T \sim H_m) = N_m.\end{aligned}$$

Hence, the shotgun estimator is unbiased as well. □

4.6.2. Proof of Theorem 4.2. We can bound the variance in (4.11) by the second moment, which is bounded by,

$$\mathbb{E}\phi_1^2 \leq \mathbb{E}\phi_1 \max \phi_1 = N_m(G) \max \phi_1.$$

Seeking to control the the maximum of ϕ_1, we see that,

$$\max_T \frac{1}{\pi_U(T)} \leq \max_A \frac{1}{|\mathrm{co}(T)|\tilde{\pi}(A)} \leq \max \frac{\prod_{r=1}^{k-1}(d_1 + \ldots + d_r)}{|\mathrm{co}(H_m)|\pi_1(d_1)},$$

$$\max_B \frac{|\mathcal{N}_v(B)|}{|\mathrm{co}(H_m)|\tilde{\pi}(B)} \leq \max \frac{\prod_{r=1}^{k-1}(d_1 + \ldots + d_r)}{|\mathrm{co}(H_m)|\pi_1(d_1)}.$$

Thus, we can construct a bound on $V_m^{\perp}(\phi_1)$.

4.6.3. Proof of Theorem 4.5. Let ϕ_i be as defined in (4.12), (4.13) or (4.14). Given two starting vertices v_i and v_j of the lifting process, notice that random variables $\phi_i|v_i$ and $\phi_j|v_j$ are independent. Therefore

$$\mathbb{E}(\phi_i\phi_{i+1})) = \mathbb{E}_{\pi_1(v_i)\times\pi_1(v_{i+1})}\mathbb{E}(\phi_i\phi_{i+1}|v_i, v_{i+1}) = \mathbb{E}_{\pi_1(v_i)\times\pi_1(v_{i+1})}(\mathbb{E}(\phi_i|v_i)\,\mathbb{E}(\phi_{i+1}|v_{i+1})).$$

Using the equation above, we can bound the covariance of ϕ_i and ϕ_{i+1} with basic inequalities:

$$|\operatorname{Cov}(\phi_i, \phi_{i+1})| \leq \sum_{x_1, x_2 \in V_G} \mathbb{E}(\phi_i | v_i = x_1)\mathbb{E}(\phi_{i+1} | v_{i+1} = x_2) \left| \mathbb{P}(v_i = x_1, v_{i+1} = x_2) - \pi_1(x_1)\pi_1(x_2) \right| \leq$$

$$\max_{x_2 \in V_G} \mathbb{E}(\phi_{i+1} | v_{i+1} = x_2) \sum_{x_1} \mathbb{E}(\phi_i | v_i = x_1) \max_{x_1} \sum_{x_2} \left| \mathbb{P}(v_i = x_1, v_{i+1} = x_2) - \pi(x_1)\pi(x_2) \right| =$$

$$2\gamma_{G_V}(h) \max_{x_2} \mathbb{E}(\phi_{i+1} | v_{i+1} = x_2) \sum_{x_1} \mathbb{E}(\phi_i | v_i = x_1),$$

where $\gamma_{G_V}(h)$ is the mixing coefficient from (4.18) for the random walk on vertices. Next, estimate factors from the RHS as follows:

$$\sum_x \mathbb{E}(\phi | v = x) \leq \max_x \frac{1}{\pi(x)} \sum_x \mathbb{E}(\phi | v = x)\pi(x) \leq 2|E_G|N_m(G). \tag{4.21}$$

For $\max_x \mathbb{E}(\phi | v = x)$, consider the expressions for ϕ from (4.12), (4.13) or (4.14).

Using notation $D = \prod_{r=2}^{k-1}(\Delta_1 + \ldots + \Delta_r)$, for the Ordered Lift estimator,

$$\max_x \mathbb{E}(\phi_O | v = x) \leq \max_x \sum_A \frac{\mathbb{P}(A | v = x)}{\tilde{\pi}(A)} \leq \max_x \frac{\#\{A \mid A[1] = x\}}{\pi(x)} \leq 2|E_G|D.$$

For the Shotgun Lift estimator,

$$\max_x \mathbb{E}(\phi_S | v = x) \leq \max_x \sum_B |\mathcal{N}_v(B)| \frac{\mathbb{P}(B | v = x)}{\tilde{\pi}(B)} \leq \max_x \frac{|\mathcal{N}_v(B)| \#\{B \mid B[1] = x\}}{\pi(x)} \leq 2|E_G|D,$$

For the Unordered Lift estimator,

$$\max_x \mathbb{E}(\phi_U | v = x) \leq \max_x \sum_T \frac{\mathbb{P}(T | v = x)}{\pi_U(T)} \leq \max_x \frac{\#\{T \mid x \in V_T\}}{\pi(x)} \leq 2|E_G|D.$$

Combining the results, we get the desired bound.

4.7. Conclusion

We introduced a flexible methodology for sampling graphlets, called Lifting. Our experimental results demonstrate our hypothesis that lifting can have smaller variance than Waddling, and better mixing rates than Subgraph Random Walking. This was reinforced by our theoretical results that control the variance of and covariance between graphlet updates. We anticipate that this lifting scheme will have more far reaching

implications, as we may apply graphlet sampling to supervised learning over graphs and other machine learning applications.

Acknowledgements: JS is supported by NSF DMS-1712996.

APPENDIX A

Long Title of Appendix A

Observations of non-wet rain have recently appeared in the literature. In this Appendix, we briefly consider the implications of these observations for the analysis offered in this dissertation.

Bibliography

[AHJ14] D. Armstrong, C. R. H. Hanusa, and B. C. Jones, *Results and conjectures on simultaneous core partitions*, European J. Combin. **41** (2014), 205–220.

[ALW16] D. Armstrong, N. A. Loehr, and G. S. Warrington, *Rational parking functions and Catalan numbers*, Ann. Comb. **20** (2016), no. 1, 21–58.

[Amd] T. Amdeberhan, *Theorems, problems and conjectures*, 1207.4045.

[And86] G. E. Andrews, *q-series: their development and application in analysis, number theory, combinatorics, physics, and computer algebra*, CBMS Regional Conference Series in Mathematics, vol. 66, American Mathematical Society, Providence, RI, 1986.

[And02] J. Anderson, *Partitions which are simultaneously t_1- and t_2-core*, Discrete Math. **248** (2002), no. 1-3, 237–243.

[ANR+17] N. K. Ahmed, J. Neville, R. A. Rossi, N. G. Duffield, and T. L. Willke, *Graphlet decomposition: framework, algorithms, and applications*, Knowledge and Information Systems **50** (2017), no. 3, 689–722.

[BA99] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, science **286** (1999), no. 5439, 509–512.

[BCK+17] M. Bressan, F. Chierichetti, R. Kumar, S. Leucci, and A. Panconesi, *Counting graphlets: Space vs time*, Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (New York, NY, USA), WSDM '17, ACM, 2017, pp. 557–566.

[BCL+11] P. J. Bickel, A. Chen, E. Levina, et al., *The method of moments and degree distributions for network models*, The Annals of Statistics **39** (2011), no. 5, 2280–2301.

[BH95] S. Billey and M. Haiman, *Schubert polynomials for the classical groups*, J. Amer. Math. Soc. **8** (1995), no. 2, 443–482.

[BHRY14] S. Billey, Z. Hamaker, A. Roberts, and B. Young, *Coxeter-Knuth graphs and a signed Little map for type B reduced words*, Electron. J. Combin. **21** (2014), no. 4, Paper 4.6, 39.

[BNY] J. Baek, H. Nam, and M. Yu, *A bijective proof of amdeberhan's conjecture on the number of $(s, s+2)$-core partitions with distinct parts*, 1705.02691.

[BRAH12] M. A. Bhuiyan, M. Rahman, and M. Al Hasan, *Guise: Uniform sampling of graphlets for large graph analysis*, Data Mining (ICDM), 2012 IEEE 12th International Conference on, IEEE, 2012, pp. 91–100.

[BS17] D. Bump and A. Schilling, *Crystal Bases: Representations and Combinatorics*, World Scientific, 2017.

[CF06] D. Chakrabarti and C. Faloutsos, *Graph mining: Laws, generators, and algorithms*, ACM computing surveys (CSUR) **38** (2006), no. 1, 2.

[CFSV04] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, *A (sub) graph isomorphism algorithm for matching large graphs*, IEEE transactions on pattern analysis and machine intelligence **26** (2004), no. 10, 1367–1372.

[Cho13] S. Cho, *A new Littlewood-Richardson rule for Schur P-functions*, Trans. Amer. Math. Soc. **365** (2013), no. 2, 939–972.

[CLWL16] X. Chen, Y. Li, P. Wang, and J. Lui, *A general framework for estimating graphlet statistics via random walk*, Proceedings of the VLDB Endowment **10** (2016), no. 3, 253–264.

[Dav70] J. A. Davis, *Clustering and hierarchy in interpersonal relations: Testing two graph theoretical models on 742 sociomatrices*, American Sociological Review (1970), 843–851.

[EG87] P. Edelman and C. Greene, *Balanced tableaux*, Adv. in Math. **63** (1987), no. 1, 42–99.

[FK96] S. Fomin and A. N. Kirillov, *Combinatorial B_n-analogues of Schubert polynomials*, Trans. Amer. Math. Soc. **348** (1996), no. 9, 3591–3620.

[FS86] O. Frank and D. Strauss, *Markov graphs*, Journal of the american Statistical association **81** (1986), no. 395, 832–842.

[Gal15] P. Galashin, *A Littlewood–Richardson rule for dual stable Grothendieck polynomials*, (2015), arXiv:1501.00051 [math.CO].

[GM16] E. Gorsky and M. Mazin, *Rational parking functions and LLT polynomials*, J. Combin. Theory Ser. A **140** (2016), 123–140.

[GMV] E. Gorsky, M. Mazin, and M. Vazirani, *Rational dyck paths in the non relatively prime case*, 1703.02668.

[GMV16] ———, *Affine permutations and rational slope parking functions*, Trans. Amer. Math. Soc. **368** (2016), no. 12, 8403–8445.

[Hai89] M. Haiman, *On mixed insertion, symmetry, and shifted Young tableaux*, J. Combin. Theory Ser. A **50** (1989), no. 2, 196–225.

[HS16] G. Han and H. Sethu, *Waddling random walk: Fast and accurate mining of motif statistics in large graphs*, 2016 IEEE 16th International Conference on Data Mining (ICDM) (2016), 181–190.

[Joh] P. Johnson, *Lattice points and simultaneous core partitions*, 1502.07934.

[Kas94] M. Kashiwara, *Crystal bases of modified quantized enveloping algebra*, Duke Math. J. **73** (1994), no. 2, 383–413.

[KN94] M. Kashiwara and T. Nakashima, *Crystal graphs for representations of the q-analogue of classical Lie algebras*, J. Algebra **165** (1994), no. 2, 295–345.

[Kra89] W. Kraśkiewicz, *Reduced decompositions in hyperoctahedral groups*, C. R. Acad. Sci. Paris Sér. I Math. **309** (1989), no. 16, 903–907.

[Kra95] ———, *Reduced decompositions in Weyl groups*, European J. Combin. **16** (1995), no. 3, 293–313.

[Lam95] T. K. Lam, *B and D analogues of stable Schubert polynomials and related insertion algorithms*, ProQuest LLC, Ann Arbor, MI, 1995, Thesis (Ph.D.)–Massachusetts Institute of Technology.

[Loe05] N. A. Loehr, *Conjectured statistics for the higher q,t-Catalan sequences*, Electron. J. Combin. **12** (2005), Research Paper 9, 54.

[MS16] J. Morse and A. Schilling, *Crystal approach to affine Schubert calculus*, Int. Math. Res. Not. IMRN (2016), no. 8, 2239–2294.

[MSOI+02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, *Network motifs: simple building blocks of complex networks*, Science **298** (2002), no. 5594, 824–827.

[NS17] R. Nath and J. A. Sellers, *Abaci structures of* $(s, ms \pm 1)$*-core partitions*, Electr. J. Comb. **24** (2017), no. 1, P1.5.

[PCJ04] N. Pržulj, D. G. Corneil, and I. Jurisica, *Modeling interactome: scale-free or geometric?*, Bioinformatics **20** (2004), no. 18, 3508–3515.

[PCJ06] N. Pržulj, D. G. Corneil, and I. Jurisica, *Efficient estimation of graphlet frequency distributions in protein–protein interaction networks*, Bioinformatics **22** (2006), no. 8, 974–980.

[RA15] R. A. Rossi and N. K. Ahmed, *The network data repository with interactive graph analytics and visualization*, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[RBAH14] M. Rahman, M. A. Bhuiyan, and M. Al Hasan, *Graft: An efficient graphlet counting method for large graph analysis*, IEEE Transactions on Knowledge and Data Engineering **26** (2014), no. 10, 2466–2478.

[Ser10] L. Serrano, *The shifted plactic monoid*, Math. Z. **266** (2010), no. 2, 363–392.

[Sin92] A. Sinclair, *Improved bounds for mixing rates of markov chains and multicommodity flow*, pp. 474–487, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[Sni02] T. A. Snijders, *Markov chain monte carlo estimation of exponential random graph models*, Journal of Social Structure, Citeseer, 2002.

[Sta84] R. P. Stanley, *On the number of reduced decompositions of elements of Coxeter groups*, European J. Combin. **5** (1984), no. 4, 359–372.

[Ste89] J. R. Stembridge, *Shifted tableaux and the projective representations of symmetric groups*, Adv. Math. **74** (1989), no. 1, 87–134.

[Str16] A. Straub, *Core partitions into distinct parts and an analog of Euler's theorem*, European J. Combin. **57** (2016), 40–49.

[WLR+14] P. Wang, J. C. S. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan, *Efficiently estimating motif statistics of large networks*, ACM Trans. Knowl. Discov. Data **9** (2014), no. 2, 8:1–8:27.

[WP96] S. Wasserman and P. Pattison, *Logit models and logistic regressions for social networks: I. an introduction to markov graphs andp*, Psychometrika **61** (1996), no. 3, 401–425.

[WS98] D. J. Watts and S. H. Strogatz, *Collective dynamics of 'small-world'networks*, nature **393** (1998), no. 6684, 440–442.

[Xioa] H. Xiong, *Core partitions with distinct parts*, 1508.07918.

[Xiob] ———, *On the largest sizes of certain simultaneous core partitions with distinct parts*, 1709.00617.

[YQJZ17] S. H. F. Yan, G. Qin, Z. Jin, and R. D. P. Zhou, *On* $(2k+1, 2k+3)$*-core partitions with distinct parts*, Discrete Math. **340** (2017), no. 6, 1191–1202.

[Zal] A. Zaleski, *Explicit expressions for the moments of the size of an* $(n, dn-1)$*-core partition with distinct parts*, 1702.05634.

[ZZ17] A. Zaleski and D. Zeilberger, *Explicit expressions for the expectation, variance and higher moments of the size of a (2n + 1, 2n + 3)-core partition with distinct parts*, Journal of Difference Equations and Applications **23** (2017), no. 7, 1241–1254.