ÉCOLE NORMALE SUPÉRIEURE DE LYON

INTERNSHIP REPORT

Lower bound on the entropy of two dimensional shifts of finite type

Author:
Xuan Hoang LA

Supervisors:
Dr. Nathalie AUBRUN
Dr. Michaël RAO

A report submitted in fulfillment of the requirements for the bachelor degree's internship

in the

LIP MC2
Computer Science Department

Introduction

In coding theory, shifts spaces are also called symbolic dynamical systems and they represent the evolution of a discrete system. Two dimensional shift spaces are colorings of a plane which avoid a set of forbidden patterns. In data storage, they can model a magnetic disk where the spins are arranged under some constraints. The *entropy*, also known as the *capacity* of a shift, gives an approximation of the storage capability of a disk.

Shifts of finite type (SFT) are the simplest to study from a combinatorial point of view. Yet, seemingly simple problems such as determining if a multidimensional SFT is nonempty turns out to be undecidable [Berger, 1966][Robinson, 1971]. In the general case, the entropy function of a 2D-SFT is non computable.This report will present the necessary tools to understand (1) and be able to compute the entropy for 1D-SFTs (2) before moving onto approximation methods for 2D-SFTs (3). This work results from a bibliographical study of *An Introduction to Symbolic Dynamics and Coding* written by Douglas Lind and Brian Marcus [Lind and Marcus, 1995], and *Accurate Lower Bounds on Two-Dimensional Constraint Capacities From Corner Transfer Matrices* written by Yao-ban Chan and Andrew Rechnitzer [Chan and Rechnitzer, 2014].

Contents

Chapter 1

Shifts of finite type

1.1 What is a shift space?

A (one dimensional) shift space is a particular set of bi-infinite words w whose letters are taken from a finite set $\mathcal{A}$ of symbols called the alphabet. An example of a word is $w = ...01010101...$ where $\mathcal{A} = \{0, 1\}$. A word will be denoted by $w = (w_i)_{i\in\mathbb{Z}}$ or

$$w = ...w_{-1}w_0.w_1w_2... \in \mathcal{A}^{\mathbb{Z}}.$$

A block u over $\mathcal{A}$ is a finite sequence of letters from $\mathcal{A}$. The length of a block is denoted by $|u|$. An empty block is a block with length 0 and is denoted by ε. For example, $u = 101010 \in \mathcal{A}^6$. Let w be a word over $\mathcal{A}$, we define

$$w_{[i,j]} = w_iw_{i+1}...w_{j-1}w_j \in \mathcal{A}^{j-i+1}.$$

An infinite word $w \in \mathcal{A}^{\mathbb{Z}}$ contains a block $u \in \mathcal{A}^n$ if there exists $i \in \mathbb{Z}$ such that $w_{[i,i+n-1]} = u$.

Let the set of forbidden blocks $\mathcal{F}$ be a collection of blocks over $\mathcal{A}$, we define the shift space $X_{\mathcal{F}}$ as the collection of words of $\mathcal{A}^{\mathbb{Z}}$ that do not contain any block from $\mathcal{F}$.
Examples of shift spaces over $\mathcal{A} = \{0, 1\}$:

- $\mathcal{F} = \varnothing$, $X_{\mathcal{F}} = \mathcal{A}^{\mathbb{Z}}$ is called the full 2-shift.
- $\mathcal{F} = \{11\}$, $X_{\mathcal{F}}$ is called the golden mean shift.
- $\mathcal{F} = \{10^n1 |\ n \text{ is odd}\}$, $X_{\mathcal{F}}$ is called the even shift.

Let $\mathcal{Y}$ be a subset of the full shift and $\mathcal{L}_n(\mathcal{Y})$ be the set of n-blocks (blocks of length n) that appears in $\mathcal{Y}$'s words, we define the language of $\mathcal{Y}$ as follows :

$$\mathcal{L}(\mathcal{Y}) = \bigcup_{n\in\mathbb{N}} \mathcal{L}_n(\mathcal{Y}).$$

Examples of language of shift spaces over $\mathcal{A} = \{0, 1\}$:

- The full 2-shift has language $\{\varepsilon, 0, 1, 00, 01, 10, 11, ...\} = \mathcal{A}^*$.
- The golden mean shift has language $\{\varepsilon, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, ...\}$.
- The even shift has language $\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 110, 111, ...\}$.

Notice that the complement of the language $\mathcal{A}^* - \mathcal{L}(\mathcal{Y})$ (also noted $\mathcal{L}(\mathcal{Y})^C$) is the biggest (for inclusion) set of forbidden blocks that defines the shift space $\mathcal{Y}$.

1.2 Higher block shifts

The notion of a higher block shift will be useful later in the chapter.
Let $\mathcal{Y}$ be a shift space over $\mathcal{A}$ and $\mathcal{A}_{\mathcal{Y}}^n = \mathcal{L}_n(\mathcal{Y})$. The n-th higher block code $\beta_n : \mathcal{Y} \to (\mathcal{A}_{\mathcal{Y}}^n)^{\mathbb{Z}}$ is defined by

$$\beta_n(x)_i = x_{[i,i+n-1]}.$$

The image of w under β_2 has the form:

$$\beta_2(w) = ...(w_{-1}w_0).(w_0w_1).(w_1w_2)...$$

You can notice from this example that consecutive 2-blocks always overlap. We say that two n-blocks $u = u_1...u_n$ and $v = v_1...v_n$ overlap progressively if

$$u_2...u_n = v_1...v_{n-1}.$$

The n-th higher block shift $\mathcal{Y}^n$ is $\beta_n(\mathcal{Y})$. We will prove in the next section that $\mathcal{Y}^n$ is always a shift space over $\mathcal{A}_{\mathcal{Y}}^n$.
Now, let us look at a simple example of a higher block shift. Let $\mathcal{Y}$ be the golden mean shift: $\mathcal{A}_{\mathcal{Y}}^2 = \{00, 01, 10\}$ and $\mathcal{Y}^2 = X_{\mathcal{F}}$ where $\mathcal{F} = \{00.10, 01.00, 01.01, 10.10\}$ since those blocks do not overlap progressively and notice that 11 is naturally forbidden in the alphabet $\mathcal{A}_{\mathcal{Y}}^2$.

1.3 Sliding block codes and conjugacy

In dynamical system theory, a classical in difficult problem is to determine which systems are "the same". Usually, this notion of similarity is defined through conjugacy, that transforms a system into an equivalent one (the orinal system can be recovered from the letter by another conjugacy). In symbolics dynamics, conjugacy are defined through sliding block codes.

Let $\mathcal{A}$ and $\mathcal{B}$ be alphabets, m and n be integers and $\mathcal{Y}$ be a shift space over $\mathcal{A}$. We call $\Phi : \mathcal{L}_{m+n+1}(\mathcal{Y}) \to \mathcal{B}$ a block map with memory m and anticipation n and $\varphi : \mathcal{Y} \to \mathcal{B}^{\mathbb{Z}}$ the induced sliding block code defined as follows:

$$\varphi(w)_i = \Phi(w_{[i-m,i+n]}).$$

For example, if $\mathcal{B} = \mathcal{A}, m = 0, n = 1$ and $\Phi(w_0w_1) = w_1$ then φ is called the shift map which will be denoted by σ.
If $\mathcal{B} = \mathcal{A}_{\mathcal{Y}}^N, m = 0, n = N$ and $\Phi(w_{[i,i+N]}) = w_{[i,i+N]}$ then $\varphi = \beta_N$, the N-th higher block code from the previous section. The following proposition will prove that $\beta_N(\mathcal{Y})$ is a shift space.

Proposition: The image of a shift space under a sliding block code is a shift space.

Proof. Let $\mathcal{X}$ and $\mathcal{Y}$ be shift spaces and $\varphi : \mathcal{X} \to \mathcal{Y}$ be a sliding m+n+1-block code induced by Φ. Let $\mathcal{L} = \{\varphi(w) | w \in \mathcal{L}(\mathcal{X})\}$ (φ's definition is extended to blocks to simplify notations). Let us show that $\varphi(\mathcal{X}) = X_{\mathcal{L}^C}$, in other words, that $\varphi(\mathcal{X})$ is a

shift space.
If $x \in \mathcal{X}$ then every block in $\varphi(x)$ is in $\mathcal{L}$ so $\varphi(x) \in X_{\mathcal{L}^C}$ which proves $\varphi(\mathcal{X}) \subseteq X_{\mathcal{L}^C}$.
Now, let $y \in X_{\mathcal{L}^C}$: there exists a x^{k_0} such that $\varphi(x^{k_0}_{[-m,n]}) = y_0$. Since there are only finitely many different m+n+1-blocks, there is an infinite set S_0 of integers k_0 for which x^{k_0} verifies the above equation. Next, there is an infinite set $S_1 \subseteq S_0$ of integers k_1 for which $\varphi(x^{k_1}_{[-1-m,1+n]}) = y_{[-1,1]}$.
And so on, we can build a sequence $(x^{k_l})_{l\in\mathbb{N}}$ such that:

$$\varphi(x^{k_l}_{[-l-m,l+n]}) = y_{[-l,l]}.$$

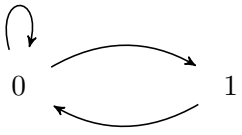
Now, let define $x \in \mathcal{X}$ with $x_{[-l-m,l+n]} = x^{k_l}_{[-l-m,l+n]}$ for every l so that $\varphi(x) = y$ which proves that $X_{\mathcal{L}^C} \subseteq \varphi(\mathcal{X})$. □

A sliding block code that is one to one is called an embedding, one that is onto is called a factor code and an embedding which is also a factor code is called a conjugacy.
If we look back at $\beta_N : \mathcal{Y} \to \beta_N(\mathcal{Y})$, it is onto by definition and clearly one to one (if you look at the last letter from each N-block in $\beta_N(y)$, you can recognize y) thus making it a conjugacy. The relevance of this property relies on the fact that conjugacies preserve entropy which we will talk about in the second chapter.

1.4 Shifts of finite type

A shift of finite type (SFT) is a shift space $\mathcal{Y}$ such that there exists a finite set $\mathcal{F}$ of forbidden blocks that verifies $\mathcal{Y} = X_{\mathcal{F}}$. For example, the golden mean shift is an SFT since $\mathcal{F} = \{11\}$.
Notice that, in the case where $\mathcal{F}$ is finite, every block in $\mathcal{F}$ could be "made into" M+1-blocks where M+1 is the length of $\mathcal{F}$'s longest block simply by replacing it with all M+1-blocks that contain it ($\mathcal{F}$ stays finite in the process). This remark leads us to defining a M-step SFT $\mathcal{Y}$ as a SFT such that there exists a set $\mathcal{F}$ of forbidden M+1-blocks that verifies $\mathcal{Y} = X_{\mathcal{F}}$.

The interesting thing about SFTs is that they can be represented by finite graphs so let us define what a graph is (as many different definitions of graphs exist):
A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a set of vertices $\mathcal{V}$ (often $\{1, ..., n\}$) and a set of edges $\mathcal{E} \subseteq \mathcal{V}^2$. Such a graph can be represented by a 0/1-adjacency matrix A where $A_{i,j} = 1$ if and only if $(i, j) \in \mathcal{E}$ and 0 otherwise.
For example, the graph represented by the matrix $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ is the following:



We define a vertex shift as the collection of words created by reading the letters on the vertices during bi-infinite walks on the graph, more formally, the vertex shift $\mathcal{X}_{\mathcal{G}}$ (or $\mathcal{X}_A$ where A is the adjacency matrix) over the alphabet $\mathcal{V}$ is defined as follows:

$$\mathcal{X}_{\mathcal{G}} = \{x = (x_i)_{i\in\mathbb{Z}} \in \mathcal{V}^{\mathbb{Z}} | \forall i \in \mathbb{Z}, A_{x_i,x_{i+1}} = 1\}.$$

To prove that this is shift space, you only have to look at the set $\mathcal{F} = \{ij | A_{i,j} = 0\}$ of forbidden 2-blocks and you can see that vertex shifts are 1-step SFTs.
Conversely, 1-step SFTs are also vertex shifts if you define A as follows: $A_{i,j} = 0$ if $ij \in \mathcal{F}$ and 1 otherwise.
If you look back at the graph in this section's example, you will realise that it is the graph representation of the golden mean shift which is an 1-step SFT because the set of forbidden blocks is $\{11\}$.
Now, not every SFT is 1-step but a forbidden M+1-block in $\mathcal{Y}$ is actually a forbidden 2-block in $\mathcal{Y}^M$ which, as a result, is 1-step. More precisely, let $u = u_1...u_{M+1} \in \mathcal{F}$, u being forbidden in $\mathcal{Y}$ is equivalent to $(u_1...u_M).(u_2...u_{M+1})$ being forbidden in $\mathcal{Y}^M$.

To sum up, you can represent any (M-step) SFT with a graph whose vertex shift is its M-th higher block shift (which is conjugate to the initial SFT). The point of representing an SFT by a graph is to compute its entropy, the purpose of the second chapter.

Chapter 2

Entropy

2.1 What is the entropy of a shift space?

The entropy measures the complexity, the information capacity of a shift space. If you recall, the language of a shift space $\mathcal{L}_n(\mathcal{Y})$ gives the different n-blocks that appear in $\mathcal{Y}$ and its number of elements grows similarly to 2^{cn} where c is a constant. We are interested in the growth rate c for large n, which leads to the following definition of entropy $h(\mathcal{Y})$ (where $\mathcal{Y}$ is a shift space):

$$h(\mathcal{Y}) = \lim_{n\to\infty} \frac{1}{n} log_2 |\mathcal{L}_n(\mathcal{Y})|.$$

For example, the entropy of the full 2-shift ($\{0,1\}^{\mathbb{Z}}$) is 1 since $|\mathcal{L}_n(\{0,1\}^{\mathbb{Z}})| = 2^n$.

Proposition: The entropy is preserved through conjugacy.

Proof. Let $\mathcal{X}$ and $\mathcal{Y}$ be shift spaces and $\varphi : \mathcal{X} \to \mathcal{Y}$ a conjugacy induced by Φ, a m+n+1-block map:
φ is a factor code (onto) then every block in $\mathcal{L}_k(\mathcal{Y})$ is the image under Φ of a block in $\mathcal{L}_{m+n+k}(\mathcal{X})$, which means $|\mathcal{L}_k(\mathcal{Y})| \leq |\mathcal{L}_{m+n+k}(\mathcal{X})|$ and

$$\begin{aligned} h(\mathcal{Y}) = \lim_{k\to\infty} \frac{1}{k} |\mathcal{L}_k(\mathcal{Y})| &\leq \lim_{k\to\infty} \frac{1}{k} |\mathcal{L}_{m+n+k}(\mathcal{X})| \\ &\leq \lim_{k\to\infty} (\frac{m+n+k}{n}) \frac{1}{m+n+k} |\mathcal{L}_{m+n+k}(\mathcal{X})| \\ &= h(\mathcal{X}). \end{aligned}$$

φ is also an embedding (one to one) and the same reasoning is valid to prove that $h(\mathcal{X}) \leq h(\mathcal{Y})$ thus proving $h(\mathcal{X}) = h(\mathcal{Y})$. □

This proposition proved that the n-th higher block shift in the first chapter preserve the entropy of the initial shift which allows us to compute the entropy of a SFT through its graph representation.

2.2 Computing the entropy

First, to be able to compute the entropy of a SFT, we have to understand some graph and linear algebra properties. Some vocabularies and properties will not be specified or be proven since they are common to graphs and linear algebra and the intricate details of such specifications or proofs do not contribute to the purpose of this report.

As we have mentioned in the first chapter, a word in an SFT is a bi-infinite walk on the graph so a block would be a finite walk, more precisely, a n-block corresponds to a path of length n. To be able to compute the entropy, we are interested in the number of n-blocks that could appear in a word, which translates to the number of paths of length n in the corresponding graph. Conveniently, the number of paths of length n from i to j is given by $A^n_{i,j}$ where A is the corresponding adjacency matrix:

$$|\mathcal{L}_n(\mathcal{X}_A)| = \sum_{i,j} A^n_{i,j}.$$

Proposition: Let A be a nonnegative and nonzero matrix that has a positive eigenvector, the corresponding eigenvalue λ is positive and there exist constants c and d such that:

$$c\lambda^n \leq \sum_{i,j} A^n_{i,j} \leq d\lambda^n.$$

With this proposition and a simple calculation, you can obtain the following equation:

$$h(\mathcal{X}_A) = log_2(\lambda).$$

The Perron-Frobenius theorem: Let A be a nonzero and irreducible matrix. Then A has a positive eigenvector v with corresponding positive eigenvalue $\lambda_P(A)$, called the Perron eigenvalue, that is geometrically simple (the corresponding eigenspace is one dimensional) and algebraically simple (simple root of the characteristic polynomial). If μ is another eigenvalue for A, then $|\mu| < \lambda_P(A)$. Any positive eigenvector for A is a positive multiple of v.

An irreducible matrix is a matrix that verifies: $\forall(i,j), \exists n, A^n_{i,j} > 0$ which translates to: there is a path from any vertex to any other on the corresponding graph.
Thanks to the Perron-Frobenius theorem, if a shift has an irreducible graph, we know that its entropy is $log_2(\lambda_P(A))$. Of course, not every shift can be represented by an irreducible graph, which leads us to this next proposition.

Proposition: Let $\mathcal{G}$ be a graph with $(\mathcal{G}_i)_{i\in I}$ irreducible components:

$$h(\mathcal{X}_\mathcal{G}) = max_i h(\mathcal{X}_{\mathcal{G}_i}) = max_i h(\lambda_P(\mathcal{G}_i)).$$

With this last proposition, we have all the tools to compute the entropy of an arbitrary SFT, of course, supposing that we know how to find the Perron eigenvalue of the corresponding adjacency matrix, but that is not the topic.

Chapter 3

Two dimensional shifts of finite type

3.1 What is a two dimensional shift of finite type?

A two dimensional shift space is a particular set of infinite grids (words) filled with letters taken from a finite alphabet. One way to represent it is through an infinite matrix:

$$w = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \\ \dots & 1 & 0 & 1 & 0 & \dots \\ \dots & 0 & 1 & 0 & 1 & \dots \\ \dots & 1 & 0 & 1 & 0 & \dots \\ \dots & 0 & 1 & 0 & 1 & \dots \\ & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \in \mathcal{A}^{\mathbb{Z}^2}.$$

Here, a block would be a (finite or infinite) matrix, thus, the set of forbidden blocks $\mathcal{F}$ is a set of matrices and the shift space $X_{\mathcal{F}}$ is the collection of words that do not contain any block from $\mathcal{F}$.
A shift of finite type (2D-SFT) is a shift space (also called "constraint") whose set of forbidden blocks is finite.
Examples of 2D-SFTs over $\mathcal{A} = \{0, 1\}$ (here "$*$" may be replaced by 0 or 1):

- $\mathcal{F} = \{\begin{bmatrix}1 & 1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\}$, $X_{\mathcal{F}}$ is called the hard square (HS) constraint.
- $\mathcal{F} = \{\begin{bmatrix}1 & 1\end{bmatrix}, \begin{bmatrix}1 & *\\ * & 1\end{bmatrix}, \begin{bmatrix}* & 1\\ 1 & *\end{bmatrix}\}$, $X_{\mathcal{F}}$ is called the read-write isolated memory (RWIM) constraint.
- $\mathcal{F} = \{\begin{bmatrix}1 & 1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1 & *\\ * & 1\end{bmatrix}, \begin{bmatrix}* & 1\\ 1 & *\end{bmatrix}\}$, $X_{\mathcal{F}}$ is called the non attacking kings (NAK) constraint.

Some other examples of two dimensional shift spaces:

- If the even restriction ($\mathcal{F} = \{10^n1 |\ n \text{ is odd}\}$, the set of forbidden blocks for the 1D-even shift) is not only a restriction on rows but also columns, you obtain the 2D-even shift (or even constraint).
- If the charge(3) restriction ($\mathcal{F} = \{u = ...+1-1+1+1-1... |$ sum of u's charges is between -3 and +3 included$\}$, the set of forbidden blocks for the 1D-charge(3) shift) is not only a restriction on rows but also columns, you obtain the 2D-charge(3) shift (or charge(3) constraint).

- $\mathcal{F} = \{ \begin{bmatrix} a \\ a \end{bmatrix}, \begin{bmatrix} a & a \end{bmatrix} | a \in \{0, ..., q-1\}\}$, $X_{\mathcal{F}}$ is called the q-colouring constraint.

The entropy h of a constraint $\mathcal{Y}$ is defined as follows:

$$h(\mathcal{Y}) = \lim_{n\to\infty} \lim_{m\to\infty} \frac{1}{nm} |\mathcal{L}_{nm}(\mathcal{Y})|$$

where $\mathcal{L}_{nm}(\mathcal{Y})$ is the collection of different allowed matrices of size $n \times m$.

Unlike the one dimensional case, only a handful of 2D constraint have known entropy, in the general case, the entropy of a 2D-SFT is non computable [Hochman and Meyerovitch, 2010]. In the next section, we will discuss one technique to approximate the entropy of any given 2D-SFT.

3.2 Strip systems

This method of numerical approximation is called the strip system method.
The strip system S_n of a 2D-SFT $\mathcal{Y}$ is a 1D-SFT $S_n(\mathcal{Y})$ defined as the set of allowed matrices of $\mathcal{Y}$ of height n over the alphabet of all allowed columns of height n. A word $w \in S_n(\mathcal{Y})$, called a n-high strip, has the following form:

$$w = \begin{bmatrix} \dots & 1 & 0 & 1 & 0 & \dots & \uparrow \\ \dots & 0 & 1 & 0 & 1 & \dots & n \\ \dots & 1 & 0 & 1 & 0 & \dots & | \\ \dots & 0 & 1 & 0 & 1 & \dots & \downarrow \end{bmatrix}.$$

Moreover, we have the following relation as a direct consequence of the definition of the entropy:

$$\lim_{n\to\infty} \frac{h(S_n(\mathcal{Y}))}{n} = h(\mathcal{Y}).$$

As a result, we can approximate 2D-SFTs entropy by calculating it for 1D-SFTs. However, this method requires a lot of calculations as it increases exponentially as n grows larger and the convergence rate is quite slow if you want more precise approximations. Many other methods exist and in the next section, we will present a method to obtain a good lower bound on the entropy efficiently for some particular SFTs.

3.3 Lower bound on the entropy

In the following subsections, we will consider only constraints over $\mathcal{A} = \{0, 1\}$ and the forbidden blocks must be of size at most 2×2 and their symmetry across a vertical line must still be forbidden. More particularly, we are considering the HS, NAK and q-colouring constraints (defined in 3.1). The computation method for other constraints will be a slightly modified version of the method that we are about to discuss.

3.3.1 The corner transfer matrix technique

Since the constraints that we are considering have forbidden blocks of size at most 2×2, we can define:

$$\omega \begin{bmatrix} a & b \\ c & d \end{bmatrix} = 1 \text{ if the block is allowed and } 0 \text{ otherwise.}$$

Then, if we consider a strip of height m with cylindrical boudaries (which means that the top and bottom of the matrix are the same), we can define the column transfer matrix $V^{[m]}$ of size $2^m \times 2^m$:

$$V^{[m]}_{\sigma,\tau} = \prod_{i=1}^{m} \omega \begin{bmatrix} \sigma_{i+1} & \tau_{i+1} \\ \sigma_i & \tau_i \end{bmatrix} = 1 \text{ if } \sigma \text{ and } \tau \text{ can be consecutive columns and } 0 \text{ otherwise.}$$

Since the symmetry across a vertical line of the forbidden blocks are also forbidden, $V^{[m]}$ is symmetric, which means that the Perron eigenvalue is given by:

$$\lambda_P(V^{[m]}) = max_\psi \frac{\psi^T V^{[m]} \psi}{\psi^T \psi}.$$

For now, this is a slightly particular case of the strip system so you may recall that the entropy is simply the logarithm of the Perron eigenvalue, which means that obtaining a lower on this value gives us directly a lower bound on the entropy. And to obtain a lower bound, it suffices to substitute a particular ψ in the latter equation. Our purpose is to find that particular vector to obtain a good lower bound.

Let $\{F(a,b)|a,b \in \{0,1\}\}$ be a set of $n \times n$ matrices so that $F(a,b) = F^T(b,a)$:
We consider the vector of size 2^m defined as follows:

$$\psi_\sigma = Tr[F(\sigma_1,\sigma_2)...F(\sigma_{m-1},\sigma_m)F(\sigma_m,\sigma_1)].$$

So for a given choice of $\{F(a,b)\}$, we obtain a vector ψ. To simplify calculations of $\psi^T\psi$ and $\psi^T V^{[m]}\psi$, we will rewrite them as traces of a new matrices.
Let R be a $(2n^2) \times (2n^2)$ matrix defined by:

$$R_{(\alpha|a|\alpha'),(\beta|b|\beta')} = F(a,b)_{\alpha,\beta} F(a,b)_{\alpha',\beta'}.$$

We have the following equation: $\psi^T\psi = Tr(R^m)$.
Similarly, let S be a $(4n^2) \times (4n^2)$ matrix defined by:

$$S_{(\alpha|a,a'|\alpha'),(\beta|b,b'|\beta')} = F(a,b)_{\alpha,\beta} \omega \begin{bmatrix} b & b' \\ a & a' \end{bmatrix} F(a',b')_{\alpha',\beta'}.$$

Which gives us: $\psi^T V^{[m]}\psi = Tr(S^m)$.
These equations result from the definition of ψ that we have given.
You may notice that R and S are symmetric due to: $F(a,b) = F^T(b,a)$.

Let ξ and η be the Perron eigenvalue of R and S respectively, the following results from a simple calculation:

$$\frac{Tr(S^m)}{Tr(R^m)} \leq \lambda_P(V^{[m]}) \Rightarrow \frac{\eta}{\xi} \leq \lim_{m\to\infty} \lambda_P(V^{[m]})^{1/m}.$$

So our purpose is to find ξ and η.

Let X be the dominant eigenvector (of size $2n^2$) of R, we define $X(a)$ the vector of size n^2 that verifies: $X(a)_{\alpha,\beta} = X_{(\alpha|a|\beta)}$. We have the following result:

$$\xi X = RX \Rightarrow \xi X(a) = \sum_b F(a,b)X(b)F(b,a).$$

Similarly, let Y be the dominant eigenvector (of size $4n^2$) of S, we define $Y(a,b)$ the vector of size n^2 that verifies: $Y(a,b)_{\alpha,\beta} = Y_{(\alpha|a,b|\beta)}$. We have the following result:

$$\eta Y = SY \Rightarrow \eta Y(a,b) = \sum_{c,d} \omega \begin{bmatrix} a & b \\ c & d \end{bmatrix} F(a,c)Y(c,d)F(d,b).$$

Once again, these equalities result from the definition of R and S. With these equations, we can use the power method with random initial vectors to compute ξ and η. This is where the "corner transfer matrix renormalisation group" (CTMRG) method intervenes. It could give us prospective starting vectors that will make the power method converge faster.

The details of this technique was not the purpose of this bibliographical study and we only have to note that there exist a set of $n \times n$ matrices $A(a)$, called the corner transfer matrices, and a set of matrices $F(a,b)$, called the half column/row transfer matrices, that satisfy:

$$X(a) = A^2(a) \; and \; Y(a,b) = A(a)F(a,b)A(b).$$

For commodities, we will say A and F instead of "each matrix in the set A and F". Notice also that defining A and F this way means that the constraint is invariant under rotation by $\pi/2$, we call this the symetric case. Note that all of the constraints that we are considering (HS, NAK, q-colouring) verify this condition.

The important thing about them being the corner transfer matrix and the half column/row transfer matrix is that the following equations will allow us to compute them: suppose we can construct A and F for a matrix of size $2^p \times 2^p$, we can define A_l and F_l of size $2^{p+1} \times 2^{p+1}$

$$A_l(c)_{d,a} = \sum_b \omega \begin{bmatrix} a & b \\ c & d \end{bmatrix} F(d,b)A(b)F(b,a).$$

$$F_l(d,c)_{b,a} = \omega \begin{bmatrix} a & b \\ c & d \end{bmatrix} F(b,a).$$

These equations can be intuitively understood with the following diagram:

$$\begin{bmatrix} & & \nwarrow \\ & A_l & \\ & & \\ c & & \end{bmatrix} = \begin{bmatrix} & \leftarrow & \\ & F & \nwarrow \\ & & A \\ a & b & \\ c & d & F \quad \uparrow \end{bmatrix}$$

To recapitulate, we can compute A and F starting with 1×1 matrices ($= [1]$ by definition) and use the recursive relations to compute bigger As and Fs until we have a sufficient large matrix size. From there, we can compute X and Y and use the power method as we have discussed above.
However, we can increase the efficiency of this algorithm by manipulating directly the eigenvalues of A as they have shown to be the deciding factor on the precision of the final approximation (this observation is empirical). More precisely, we are only interested in the largest (in absolute value) eigenvalues of A. This means that at each step of the way, we diagonalize A (note that A is a symmetric and real matrix) and truncate the matrix to first n (of our choice) largest eigenvalues so that we will have less calculations to go through. Note that the recursive relations are invariant under the similarity transforms:

$$A(a) \to P^T A(a) P(a) \text{ and } F(a,b) \to P^T(a) F(a,b) P(b)$$

where $P(a)$ is $A(a)$'s orthogonal diagonalization basis.

3.3.2 The algorithm

The final algorithm is as follows [Chan and Rechnitzer, 2014]:

1. Start with $A(a) = F(a,b) = [1]$ and $n = 1$.
2. Expand A and F into A_l and F_l.
3. Increase n by 1 under a certain condition (that will be discussed in the next subsection).
4. Diagonalize A_l and let P be the matrix of the eigenvectors corresponding to the n largest eigenvalues.
5. Apply the similarity tranforms.
6. Normalize A and F so that the top-left elements of $A(0)$ and $F(0,0)$ are both 1.
7. Go back to step 2.
8. Once n is sufficient, compute the initial X and Y.
9. Apply the power method until a desired precision is reached.

Even though this algorithm is for the symmetric case, the asymetric case is practically the same with just different equations for different corner tranfer matrices as the equation for the row and column transfer matrices are separated. Both cases have been implemented as well as the case of the even and the charge(3) constraints which are not SFTs but transformations suggested in the article resulted in other constraints which are SFTs with the same entropies (once again, the details are rather technical and the method stays the same with some slight twist).

3.3.3 Implementation and results

The source code can be found at the following link.

The implementation of this algorithm had been done in C++ with the MPFR library

to manipulate highly precise reals and also with the Eigen library to diagonalize matrices and obtain the corresponding eigenvectors. The main difficulty in the implementation was the unclear condition in step 3. As a result, many different conditions have been tested:

- At first, the natural condition to use is to look at the convergence rate of A and F and increase n when the desired precision is reached. However, for a fixed size n, they do not always converge as the article [Chan and Rechnitzer, 2014] has mentioned explicitly that the convergence of this method has not been proven and it has only been empirically observed.
- The point of diagonalizing and truncating the matrices is to obtain the largest eigenvalues, the elements that have the most impact on the final approximation. Nonetheless, they are not the only deciding factors as the result obtained, by only looking at the convergence rate of the eigenvalues, has been close to known results and the results in the article but still differs for high precision.
- The next natural condition is to look at the eigenvectors (along with the eigenvalues) since they are used to reduce the matrices. One thing to note is that the eigenvectors converge at a slower rate compare to the eigenvalues and sometimes they do not converge at all. Another thing that was noticed during this process is that the orientation of the eigenvectors affects the convergence. By forcing the components of the eigenvectors to stay in the same orientation (if Eigen just happen to choose an eigenvector with the same direction but the opposite orientation) during the process seemed to eliminate this problem at first. However, after testing with different constraints than the ones mentioned in the article, this method does not seem to solve the problem as the relative direction and orientation of the vectors seems to be the real deciding factor of the convergence and this cannot be forced since it changes as the vectors converge (or do not converge).

Even though, a clear condition (that works for every 2D-SFT that verifies the initial conditions) for increasing n has not been found during the internship. Results for the constraints mentioned in the article are close to identical and some others constraints has been tested as well. The results for the ones with known entropy do confirm what have been found.

Results: The following results have been obtained with 256 bits of precision and matrix sizes of 16×16 or 32×32. For most, the precision of the approximation has been chosen to match the known approximations to see how much faster is the computation with this method. The computation time took around a minute to a few minutes for most, to a dozen minutes for charge(3), in constrast to a few days for previous approximations with different methods, which is a net increase in efficiency. The exceptions are for q-colouring constraints where the computation time takes much longer, nonetheless, the results found in the article still exceed the previously known approximations. In these results however, the precision for the q-colouring constraints has been reduced so that the computation time also takes around a few minutes.
(Note that these are the growth rate, you simply have to take the logarithm if you want the entropy)

- HS (around a minute): 1.503 048 082 475 332 264 322 1
- NAK (around a minute): 1.342 643 951

- RWIM (a few minutes): 1.448 957 4
- Even (a few minutes): 1.36
- Charge(3) (around a dozen minutes): 1.4
- 4-color (a few minutes): 2.336
- 5-color (a few minutes): 3.25
- Hard Hexagon ($\mathcal{F} = \{\begin{bmatrix}1 & 1\end{bmatrix}, \begin{bmatrix}1 & * \\ * & 1\end{bmatrix}\}$) [known] (around a minute): 1.395 485 972 479 302 735 2
- 3-color [known] (a few minutes): 1.539 6
- Custom constraint with $\mathcal{F} = \{\begin{bmatrix}1 & 1 \\ 1 & 1\end{bmatrix}\}$ (a few minutes): 1.915 316 674

Bibliography

Berger, Robert (1966). *The Undecidability of the Domino Problem*. American Mathematical Society.

Chan, Yao-ban and Andrew Rechnitzer (2014). "Accurate lower bounds on two-dimensional constraint capacities from corner transfer matrices". In: *IEEE Transactions on Information Theory* 60, pp. 3845–3858.

Hochman, Mike and Tom Meyerovitch (2010). "A characterization of the entropies of multidimensional shifts of finite type". In: *Annals of Mathematics* 171.3, pp. 2011–2038.

Lind, Douglas and Brian Marcus (1995). *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press.

Robinson, Raphael M. (1971). "Undecidability and nonperiodicity for tilings of the plane". In: *Inventiones Mathematicae* 12, pp. 177–209.

Appendix A

Institutional and social context of the internship

The internship took place in team MC2 of the LIP Computer Science laboratory, under the advisement of Dr. Nathalie Aubrun and Dr. Michaël Rao.

The internship was an occasion to witness the researchers at work and to discuss with the PhD students which gave me a clearer view and an appreciation for the domain of research.

I would like to give special thanks to Dr. Sebastián Barbieri and Dr. Nathalie Aubrun for their advices and support as I began to learn about shift spaces, a brand new notion to me. I would also like to thank Dr. Michaël Rao for helping me understand the article [Chan and Rechnitzer, 2014] and implement the algorithm.