QUANTUM CHANNELS AND THEIR MINIMUM OUTPUT ENTROPIES

An Undergraduate Research Scholars Thesis

by

NATHAN MEHLHOP and WILLIAM OGLETREE

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Michael Brannan

May 2018

Major: Mathematics

TABLE OF CONTENTS

Page

ABSTRACT

Quantum Channels and Their Minimum Output Entropies

Nathan Mehlhop and William Ogletree
Department of Mathematics
Texas A&M University

Research Advisor: Dr. Michael Brannan
Department of Mathematics
Texas A&M University

We investigate non-random quantum channels generated from the representation theory of orthogonal quantum groups to perform numerical computations for estimates of their minimum output entropies. In doing so, we address the violation of the additivity conjecture for minimum output entropies, previously established through probabilistic proofs. This violation is relevant because given two quantum channels whose minimum output entropies sum to a value greater than the minimum output entropy of their tensor product channel it can be shown that more information can be sent with greater fidelity via the two channels in parallel than by using each separately. This is directly related to the relevance of quantum entanglement in quantum information theory.

We numerically compute minimum output entropies for single quantum channels and tensor products of quantum channels with respect to both Von Neumann and Renyi entropies. We then proceed to compute entropy estimates for nonplanar channels directly with Temperley-Lieb Category Theory. We conclude by discussing which of our choices for channels are optimal for entangled inputs.

LIST OF FIGURES

FIGURE Page

1. PRELIMINARIES

1.1 Quantum Channels and Von Neumann Entropy

Shannon's breakthrough paper *A Mathematical Theory of Communication* established that a communication channel has a well-defined capacity and provided a formula for calculating it. Generalizing Shannon's theory to incorporate quantum effects became a prominent focus in the field of information theory. In doing so, we define a stochastic communication channel that generalizes to what we call a *quantum channel*.

Definition 1. *[2] Let H and K be finite-dimensional Hilbert spaces. A linear map $\Phi : End(H) \longrightarrow End(K)$ is said to be*

- *positive if $\Phi(A) \geq 0$ for any positive matrix $A \in End(H)$.*

  A positive matrix is self-adjoint with all eigenvalues real and nonnegative.

- *n-positive if $\Phi \otimes I_n$ is positive, where*

  $$\Phi \otimes I_n : End(H) \otimes \mathbb{M}_n(\mathbb{C}) \longrightarrow End(K) \otimes \mathbb{M}_n(\mathbb{C})$$

  is the linear map, such that

  $$\Phi \otimes I_n(A \otimes B) = \Phi(A) \otimes B \text{ for } A \in End(H) \text{ and } B \in \mathbb{M}_n(\mathbb{C}).$$

- *completely positive if it is n-positive $\forall\ n \geq 1$.*

Definition 2. *[2] Let H and K be finite-dimensional Hilbert spaces. A quantum channel*

$$\Phi : End(H) \longrightarrow End(K)$$

is a linear, completely positive, and trace-preserving map.

For the remainder of this paper, all Hilbert spaces are assumed to be finite dimensional, and H or similar denotes a Hilbert space when unspecified. These properties are preserved under taking tensor products. Namely, given quantum channels $\Phi_1 : H_1 \longrightarrow K_1$ and $\Phi_2 : H_2 \longrightarrow K_2$, the product $\Phi_1 \otimes \Phi_2 : H_1 \otimes H_2 \longrightarrow K_1 \otimes K_2$ is also a quantum channel.

Definition 3. *[2] The set of density operators on a Hilbert space H is*

$$D(H) \coloneqq \{M : H \longrightarrow H \mid M \text{ is linear, positive, and } Tr(M) = 1\}.$$

An element ρ of $D(H)$ is called a state.

A state ρ of $D(H)$ defined above is the input for tensor quantum channels. Typically, we use the maximally-entangled Bell state as the input for the tensorquantum channels for which we will compute the minimum output entropy later in this paper.

We can establish a geometric picture for our quantum channels by writing our channels in terms of its canonical, isometric *Stinespring representation*.

Definition 4. *[2] Let H and K be Hilbert spaces. Let $\Phi : End(H) \longrightarrow End(K)$ be a quantum channel. A Stinespring representation of Φ is a pair (E, α) consisting of a Hilbert space E and an isometry $\alpha : H \longrightarrow K \otimes E$ such that*

$$\Phi(A) = Tr_E(\alpha A \alpha^*) \tag{1.1}$$

*for any $A \in End(H)$. The map Tr_E denotes the **partial trace** over E.*

Definition 5. *Given a composite space $H_A \otimes H_B$, the partial trace Tr_B is a mapping from the density matrices ρ_{AB} of the composite space onto density matrices ρ_A on H_A defined by*

$$Tr_B(S \otimes T) = Tr(T)S. \tag{1.2}$$

for $S \otimes T$ a density operator on $H_A \otimes H_B$.

In fact, the Stinespring representation is a complete classification theorem for quantum channels. An expression of this form is always a quantum channel, and every quantum channel Φ admits such a representation for some Hilbert space E and an isometry α.

We would like to compute the classical capacity for a quantum channel, but it is unknown exactly how to do so. An analogous quantity in the case of a quantum channel is

the *Holevo Capacity*.

Definition 6. *[3] The Holevo capacity is the classical capacity for a quantum channel with the restriction that there are no entangled input states allowed across many uses of the channel. Letting $\chi()$ denote the Holevo capacity of a quantum channel,*

$$\chi(\Phi) = \min_{\substack{\rho_i \in \mathbb{M}_n(\mathbb{C})_+,\ p_i \geq 0, \\ Tr(\rho_i)=1,\ \sum_i p_i = 1}} \Big\{ S(\sum_i p_i \Phi(\rho_i)) - \sum_i p_i S(\Phi(\rho_i)) \Big\}. \quad (1.3)$$

where $S(\rho)$ denotes the entropy of the state ρ.

Thus, we see that the entropy of a state, which John von Neumann defined in his *Mathematical Foundations of Quantum Mechanics* from 1932 in analogy with the quantity in physics, has significance in its relation to channel capacities. For the remainder of the paper, we will only consider entropies themselves and leave their relevance to other quantities as a separate matter.

Definition 7. *We denote the von Neumann entropy of a positive state $\rho \in D(H)$ as*

$$S(\rho) \coloneqq -Tr(\rho \cdot log\rho). \quad (1.4)$$

We can use the von Neumann entropy to define what we call the *minimum output entropy* of a quantum channel Φ. The von Neumann entropy is defined for a state, and since the outputs of quantum channels are states, we can consider the entropies of all possible outputs of the quantum channel, and, in applications, entropy is usually sought to be minimized, so this *minimum output entropy* of the channel is the interesting quantity to consider.

Definition 8. *[4] Given a quantum channel Φ, we define the* ***minimum output entropy***

(MOE) *S^{min} of the quantum channel Φ to be*

$$S^{min}(\Phi) = \min_{\rho \in D(H)} S(\Phi(\rho)). \tag{1.5}$$

First, it should be noted that the minimum output entropy is a true minimum, not merely an infimum, because $D(H)$ is a compact set, and S is a continuous function. Second, it suffices to consider minimizing S only over the pure states, i.e. rank one projections, in $D(H)$ because $D(H)$ is a convex set and S is a convex function; hence, we need only minimize over the extremal points of $D(H)$. Thus, we write

$$S^{min}(\Phi) = \min_{|\psi\rangle} S(\Phi(|\psi\rangle \langle\psi|)), \tag{1.6}$$

where $|\psi\rangle$ is a unit vector in H [4].

1.2 Violation of Minimum Output Entropy Additivity

In quantum information theory, it was a question of interest to understand the relationship between the the MOE of a tensor product of two quantum channels with the MOEs of the individual channels. This is because the tensor product of quantum channels corresponds to running the two channels in parallel and the MOE is related to the information capacity of the channel. Hence, it was vital to understand whether there is any information-theoretic benefit to using such tensor product channels.

It is clear that, for any quantum channels Φ_1 and Φ_2, we have

$$S^{min}(\Phi_1 \otimes \Phi_2) \leq S^{min}(\Phi_1) + S^{min}(\Phi_2).$$

Namely, supposing that ρ_1 and ρ_2 are minimizers of S for Φ_1 and Φ_2 respectively, we have

$$S_{min}(\Phi_1 \otimes \Phi_2) \leq S(\Phi_1 \otimes \Phi_2(\rho_1 \otimes \rho_2)) = S(\Phi_1(\rho_1) \otimes \Phi_2(\rho_2)) =$$
$$S(\Phi_1(\rho_1)) + S(\Phi_2(\rho_2)) = S_{min}(\Phi_1) + S_{min}(\Phi_2). \tag{1.7}$$

It was conjectured before that the above inequality was actually an equality for any pair of quantum channels, but this was eventually proven to be false.

Proposition 1. *[4] (Hastings) There exist quantum channels Φ_1 and Φ_2 such that*

$$S^{min}(\Phi_1 \otimes \Phi_2) < S^{min}(\Phi_1) + S^{min}(\Phi_2). \tag{1.8}$$

However, the proof Hastings gives and others like it rely on random non-constructive methods to show the existence of such quantum channels that have MOE additivity violation [4]. It is an open question to find explicit examples of such quantum channels.

Another thing to note is that since we showed $S(\Phi_1 \otimes \Phi_2(\rho_1 \otimes \rho_2)) = S(\Phi_1(\rho_1)) + S(\Phi_2(\rho_2))$ in equation 1.7 for any state of the form $\rho_1 \otimes \rho_2$ (a so-called separable state), finding a pair of quantum channels that exhibit MOE addivity violation would be impossible if you only checked such states. Thus, non-separable states, called entangled states, are the only inputs to the tensor product channel for which we could possibly illustrate a violation of MOE additivity.

1.3 SU(2) and O_N^+ Quantum Groups

M. Al Nuwairan in her thesis [2] showed that while entanglement is always achieved using quantum channels arrising from the compact, non-commutatuve group $SU(2)$, we can not achieve a high enough degree of entanglement to obtain a violation [2]. Hence, we turn to a different class of compact quantum groups called the *free orthogonal quantum*

groups [3]. The quantum group O_N^+ has a representation theory that closely reflects that of $SU(2)$, and the unitary irreducible representations of O_N^+ have the same fusion rules as $SU(2)$. Namely, from the representation theory of O_N^+, we acquire finite-dimensional Hilbert spaces $(H_k)_{k\in\mathbb{N}}$ with the fusion rules $H_l \otimes H_m \simeq \bigoplus_{r=0}^{min\{m,l\}} H_{m+l-2r}$ and isometries $\alpha_k^{l,m} : H_k \hookrightarrow H_l \otimes H_m$ from which we get quantum channels $\Phi_k^{\bar{l},m}$ and $\Phi_k^{l,\overline{m}}$ using the Stinespring representation theorem. Furthermore, the construction of these quantum channels is well-understood using the planar calculus of Temperley-Lieb category theory [3]. Ultimately, using O_N^+ results in a much larger degree of entanglement in the subrepresentations of tensor product channels compared to those from $SU(2)$ [3].

2. OUTPUT SPECTRUM OF ENTANGLED INPUTS

The collection of quantum channels for which we will perform our numerical estimates of minimum output entropies are denoted by $\Phi_{k_1}^{\overline{l_1},m_1} \otimes \Phi_{k_2}^{l_2,\overline{m_2}}$ where the indices $k_1, k_2, l_1, m_1, l_2, m_2 \geq 1$ parameterize the collection. In particular, we calculate the spectra of specific output states

$$\Phi_{k_1}^{\overline{l_1},m_1} \otimes \Phi_{k_2}^{l_2,\overline{m_2}}\Big(\frac{\alpha_i^{k_1,k_2}\alpha_i^{k_1,k_2*}}{[i+1]_q}\Big)$$

where i is yet another parameter. Note that all of these parameters must satisfy certain relations with respect to each other in order to be *admissible*, which we describe later, and the most important case is the case when $i = 0$ in which the input state is merely a maximaly entangled Bell state. Of course, any of these calculations constitute an upper bound on the MOE of the tensor product quantum channel. However, the hope is that this upper bound is relatively tight so as to have some hope of achieving good estimate approximations.

We will compare these to known numerical estimates of the individual non-tensored quantum channels $\Phi_k^{\bar{l},m}$ where the bar denotes taking a partial trace over that factor. Without loss of generality in notation, for the individual channel case, we only write the partial trace for the first argument.

As usual, these quantum channels have Stinespring representations

$$\Phi_k^{\bar{l},m}(A) = Tr_{H_l}(\alpha_k^{l,m} A \alpha_k^{l,m*}).$$

However, it is their properties arising from the representation theory of the O_N^+ quantum group that gives us concrete information about the actual form of the α isometry. Re-

markably, the structure of the resulting quantum channels is describable entirely through the framework of Temperley-Lieb planar calculus, for which many published numerical computations already exist [5].

The construction of these quantum channels using the O_N^+ category theory is summarized in [3]. In the context of this chapter, we will only work with the numerical formulas found from this framework. In chapter 3, we will address the Temperley-Lieb planar calculus more explicitly in order to address a variation of the current problem.

2.1 Formulae

The fundamental object we will use to compute the entropies for our quantum channels are *quantum integers*.

Formula 1. *[3] Let $N > 0$. Then let q be such that $N = q + q^{-1}$. We then obtain*

$$q = \frac{N-\sqrt{N^2-4}}{2}, \qquad q^{-1} = \frac{N+\sqrt{N^2-4}}{2}. \tag{2.1}$$

Formula 2. (Quantum Integer) *[3] Let q and n be given. Then the n^{th}* **quantum integer** *is*

$$[n]_q := \frac{q^n - q^{-n}}{q - q^{-1}}, \qquad [0]_q := 1, \tag{2.2}$$

and we define

$$[n]_q! := [n]_q[n-1]_q \dots [1]_q[0]_q. \tag{2.3}$$

As a point of interest, these quantum integers mainly arise in this problem in that for $\Phi_k^{\bar{l},m}$, we have [3]

$$dim(H_k) = [k+1]_q.$$

Theorem 1. *Let q and N be given as above. Then the sequence of quantum integers is*

given by the recurrence relation

$$[n]_q = [2]_q[n-1]_q - [n-2]_q, \tag{2.4}$$

for $n \geq 3$, $[1]_q \coloneqq 1$, $[2]_q \coloneqq N$.

Proof. We define the generating function for the sequence of quantum integers

$$\begin{gathered} Z(x) \coloneqq \sum_{n=1}^{\infty}[n]_q x^n = x + Nx^2 + \sum_{n=3}^{\infty}[n]_q x^n = \\ x + Nx^2 + \sum_{n=3}^{\infty}(N[n-1]_q - [n-2]_q)x^n = \\ x + Nx^2 + N\sum_{n=3}^{\infty}[n-1]_q x^n - \sum_{n=3}^{\infty}[n-2]_q x^n = \\ x + Nx^2 + Nx\sum_{n=2}^{\infty}[n]_q x^n - x^2\sum_{n=1}^{\infty}[n]_q x^n = \\ x + Nx^2 + Nx(Z(x) - x) - x^2 Z(x) = x + NxZ(x) - x^2 Z(x). \end{gathered} \tag{2.5}$$

Thus, $Z(x) = x + NxZ(x) - x^2 Z(x)$, so $Z(x) = \dfrac{x}{1 - Nx + x^2}$. The denominator factors as $(x-q)(x-q^{-1})$, so we can decompose $Z(x)$ into partial fractions such that

$$Z(x) = \frac{1}{q - q^{-1}}\left(\frac{q}{x-q} - \frac{q^{-1}}{x - q^{-1}}\right) = \frac{1}{q-q^{-1}}\left(\frac{1}{1-qx} - \frac{1}{1-\frac{x}{q}}\right). \tag{2.6}$$

By using the geometric series formula, we get

$$Z(x) = \frac{1}{q-q^{-1}}\left(\sum_{n=1}^{\infty} q^n x^n - \sum_{n=1}^{\infty} q^{-n}x^n\right) = \frac{1}{q-q^{-1}}\sum_{n=1}^{\infty}(q^n - q^{-n})x^n. \tag{2.7}$$

Thus, by comparison with the original definition of the generating function, we get the closed form $[n]_q \coloneqq \dfrac{q^n - q^{-n}}{q - q^{-1}}$ for $n \geq 1$.

□

Formula 3. *[3] Let q and a be given. We say the* **signed quantum integer** *of a is*

$$\begin{gathered} f(a) \coloneqq (-1)^{a-1}[a]_q, \\ f(a)! \coloneqq f(a)f(a-1)\dots f(2)f(1), \textit{ and } f(0) \coloneqq 1. \end{gathered} \tag{2.8}$$

Formula 4. (Theta Net) *[3] Given $a, b,$ and i, let $r \coloneqq \frac{b+i-a}{2}$. Then*

$$\theta_q(a,b,i) := \frac{[a+r+1]_q![i-r]_q![b-r]_q![r]_q!}{[a]_q![b]_q![i]_q!}. \tag{2.9}$$

This quantity comes up in defining the isometry $\alpha_k^{l,m}$. Namely, $A_k^{l,m}$ is a *three-vertex* in the Temperley-Lieb algebra, and $\alpha_k^{l,m}$ is a rescaling of it given by

$$\alpha_k^{l,m} = \Big(\frac{[k+1]_q}{\theta_q(k,l,m)}\Big)^{\frac{1}{2}} A_k^{l,m}.$$

$\theta_q(k,l,m) = Tr_{H_k}((A_k^{l,m})^* A_k^{l,m}) =$

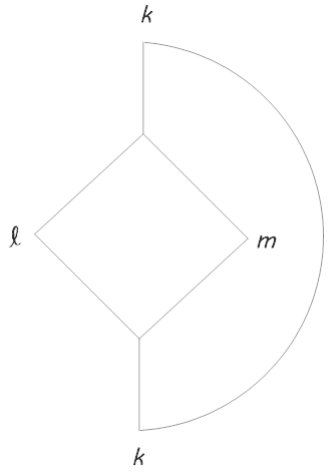


Figure 2.1: Temperley-Lieb Diagrammatic Representation [1] of a Theta Net

Figure 2.1 above is a diagrammatic representation [1] of formula 4 for our chosen dimensions $k, l,$ and m. It gives us a clear picture for our theta net and why we compute $Tr_{H_k}((A_k^{l,m})^* A_k^{l,m})$. We give more discussion of such diagrams in the Temperley-Lieb planar calculus later in chapter 3.

Formula 5. (Tetrahedral Net)*[3] Let l, k, l_1, l_2, m_1, m_2, be given. Then*

$$
\begin{aligned}
J &:= \prod_{1\leq i\leq 4, 1\leq j\leq 3} f(b_j - a_i)! \\
E &:= f(m_1)!f(l_1)!f(m_2)!f(l_2)!f(l)!f(k)! \\
a_1 &:= \frac{m_1 + l_2 + l}{2} \\
a_2 &:= \frac{l_1 + m_2 + l}{2} \\
a_3 &:= \frac{m_1 + l_1 + k}{2} \\
a_4 &:= \frac{m_2 + l_2 + k}{2} \\
b_1 &:= \frac{l_1 + l_2 + l + k}{2} \\
b_2 &:= \frac{m_1 + m_2 + l + k}{2} \\
b_3 &:= \frac{m_1 + l_1 + m_2 + l_2}{2} \\
m &:= \max_{1\leq i\leq 4} a_i \\
M &:= \min_{1\leq j\leq 3} b_j
\end{aligned}
\tag{2.10}
$$

and we have the formula

$$Tet\begin{bmatrix} m_1 & l_1 & l \\ m_2 & l_2 & k \end{bmatrix} \coloneqq \frac{J}{E}\sum_{s=m}^{M} \frac{(-1)^s f(s+1)!}{\prod_{1\leq i\leq 4} f(s-a_i)! \prod_{1\leq j\leq 3} f(b_j-s)!}. \quad (2.11)$$

Formula 6. (Quantum 6-j Symbol) *[3] Let $a, b, c, d, i,$ and j be given. Then*

$$\begin{Bmatrix} a & b & i \\ c & d & j \end{Bmatrix}_q \coloneqq \frac{Tet\left[\begin{smallmatrix} m_1 & l_1 & l \\ m_2 & l_2 & k \end{smallmatrix}\right][i+1]_q}{\theta(a,d,i)\theta(b,c,i)}. \quad (2.12)$$

That now concludes the collection of formulas that are used to describe the spectrum of $\Phi_{k_1}^{\overline{l_1},m_1} \otimes \Phi_{k_2}^{l_2,\overline{m_2}}\Big(\frac{\alpha_i^{k_1,k_2}\alpha_i^{k_1,k_2*}}{[i+1]_q}\Big)$. We can only perform this calculation for *admissible* parameters.

Definition 9. *A triple $(k,l,m) \in \mathbb{N}_0^3$ is admissible if and only if $\exists$ an integer $0 \leq r \leq min\{l,m\}$ such that $k = l+m-2r$.*

Each triple (k_1, l_1, m_1) and (k_2, l_2, m_2) must be admissible, and i must satisfy a triangle inequality:

$$|k_1 - k_2| \leq i \leq k_1 + k_2.$$

The admissibility condition is directly related to the fusion rules for the representation theory of O_N^+ (which are the same as for SU(2)). Namely, a triple $(k,l,m) \in \mathbb{N}_0^3$ is admissible $\iff$ $H_k < H_l \otimes H_m$, i.e. H_k is a subrepresentation of $H_l \otimes H_m$.

Formula 7. (General Eigenvalue Formula)*[3] The state $\Phi_{k_1}^{\overline{l_1},m_1} \otimes \Phi_{k_2}^{l_2,\overline{m_2}}\Big(\frac{\alpha_i^{k_1,k_2}\alpha_i^{k_1,k_2*}}{[i+1]_q}\Big)$*

has eigenvalues as follows (with multiplicity $[l+1]_q$)

$$\lambda_{i,l}^{m_1,l_2} = \left(\frac{[k_1+1]_q[k_2+1]_q\theta_q(l,m_1,l_2)}{[l+1]\theta_q(k_1,l_1,m_1)\theta_q(k_2,l_2,m_2)\theta_q(i,k_1,k_2)}\right)$$

$$\times \sum_{\substack{j=2t,\\ 0\leq t\leq min(k_1,k_2)}} \frac{\begin{Bmatrix} k_1 & k_2 & j \\ k_2 & k_1 & i \end{Bmatrix}_q Tet\begin{bmatrix} k_1 & m_1 & j \\ m_1 & k_1 & l_1 \end{bmatrix} Tet\begin{bmatrix} k_2 & l_2 & j \\ l_2 & k_2 & m_2 \end{bmatrix} \begin{Bmatrix} m_1 & l_2 & l \\ l_2 & m_1 & j \end{Bmatrix}_q}{\theta_q(m_1,m_1,j)\theta_q(l_2,j,l_2)} \tag{2.13}$$

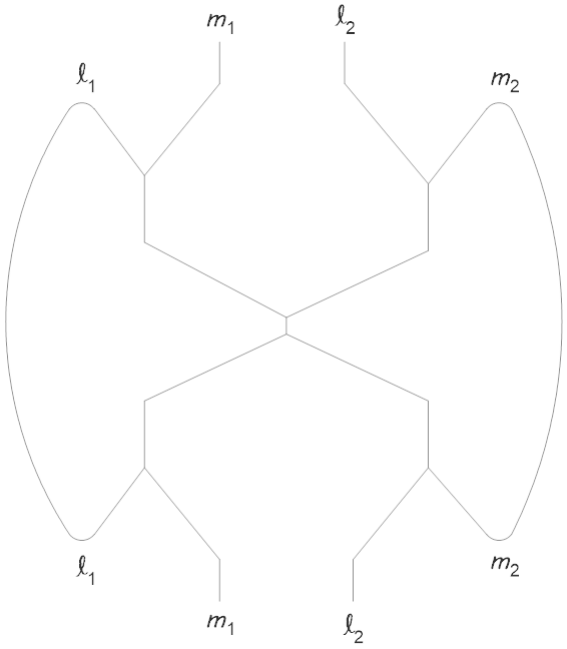


Figure 2.2: Temperley-Lieb Diagrammatic Representation [1] of $\Phi_{k_1}^{\overline{l_1},m_1} \otimes \Phi_{k_2}^{l_2,\overline{m_2}}$ with O_N^+ Equivariant Isometries

In figure 2.2, we give another diagram for illustration of the form of the quantum channels and their inputs. Consider the above figure in the case where $i = 0$. Note that

$i = 0 \implies k_1 = k_2$. Then the center of figure 2.2 can be redrawn in the following way.

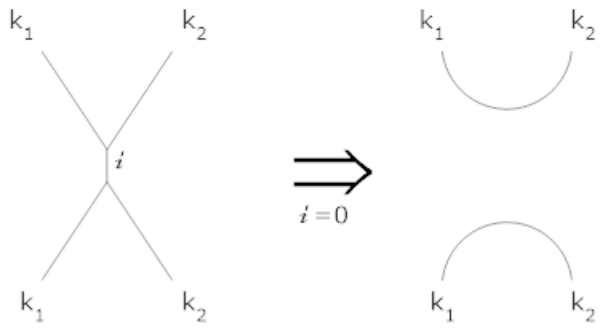


Figure 2.3: Center of $\Phi_{k_1}^{\overline{l_1},m_1} \otimes \Phi_{k_2}^{l_2,\overline{m_2}}$ when $i = 0$

If we suppose $k_1 = k_2$ and $i = 0$ as shown in figure 2.3 (for which we acquire the Bell state input), our general eigenvalue formula case simplifies nicely.

Formula 8. *[3] Assume $k_1 = k_2 \coloneqq k$ and $i = 0$. Then we can write*

$$\lambda_{i,l}^{m_1,l_2} = \lambda_{0,l}^{m_1,l_2} = \frac{[k+1]_q Tet\left[\begin{smallmatrix} m_1 & l_1 & l \\ m_2 & l_2 & k \end{smallmatrix}\right]^2}{\theta_q(l_1, m_1, k)\theta_q(l_2, m_2, k)\theta_q(m_1, l_2, l)\theta_q(l_1, m_2, l)}. \tag{2.14}$$

Formula 9. (Minimum Output Entropy of Individual Quantum Channel) *[3] Let $l, m,$ and k be given. Then the minimum output entropy of the channel $\Phi_k^{l,m}$ is bounded as follows*

$$S_{min}(\Phi_k^{\bar{l},m}) \geq \log\left(\frac{\theta_q(k,l,m)}{[k+1]_q}\right). \tag{2.15}$$

This follows from an estimate in [3] for the largest eigenvalue for output states of a single quantum channel of the type we're studying. Namely, $\lambda_{max} \leq \frac{[k+1]_q}{\theta_q(k,l,m)}$. Then we merely apply the von Neumann entropy formula:

$$S_{min}(\Phi_k^{\bar{l},m}) = \sum_{\lambda} -\lambda log(\lambda) \geq \sum_{\lambda} -\lambda log(\lambda_{max})$$

$$= -log(\lambda_{max}) \sum_{\lambda} \lambda = -log(\lambda_{max}) = log(\frac{1}{\lambda_{max}})$$

2.2 Von Neumann Entropy

At this time, we can use the eigenvalues we found to calculate an upper bound for the tensor product channel minimum output entropies. Note that the Bell state ρ on $H_l \otimes H_m$ is a maximally-entangled quantum state, i.e. $\rho = |\psi\rangle \langle\psi|$, where $|\psi\rangle = \frac{1}{[k+1]_q^{\frac{1}{2}}} \sum_i e_i \otimes f_i$ such that $(e_i)_i$ and $(f_i)_i$ are orthonormal bases of H_l and H_m respectively.

Formula 10. (Von Neumann Entropy of Tensor Quantum Channels) *[3] Let $m_1, m_2, l_1, l_2,$ and k be given. Let ρ be the Bell state. Then the minimum output entropy estimate for the tensor channel $\mathbf{\Phi_k^{\overline{l_1},m_1}} \otimes \mathbf{\Phi_k^{l_2,\overline{m_2}}}$ with respect to the von Neumann entropy is*

$$S(\Phi_k^{\overline{l_1},m_1} \otimes \Phi_k^{l_2,\overline{m_2}}(\rho)) = - \sum_{\substack{0 \leq r \leq min\{m_1,l_2\}, \\ l = m_1 + l_2 - 2r}} [l+1]_q \lambda_{0,l}^{m_1,l_2} \log\left(\lambda_{0,l}^{m_1,l_2}\right). \tag{2.16}$$

Hence, we want to find quantum channels such that

$$S(\Phi_k^{\overline{l_1},m_1} \otimes \Phi_k^{,\overline{m_2}}(\textit{Bell state})) - S(\Phi_k^{l_1,m_1}) - S(\Phi_k^{l_2,m_2}) < 0. \tag{2.17}$$

The above computation is done in practice by Hastings [4] and others. However, we propose a simplified formula to reduce the computational complexity of formula 10. As it

stands, we must calculate

$$-\sum_{\substack{0\le r\le min\{m_1,l_2\},\\ l=m_1+l_2-2r}} [l+1]_q \lambda_{0,l}^{m_1,l_2} \log\left(\lambda_{0,l}^{m_1,l_2}\right) - \log\left(\frac{\theta_q(k,l_1,m_1)}{[k+1]_q}\right) - \log\left(\frac{\theta_q(k,l_2,m_2)}{[k+1]_q}\right).$$

Again, it seems reasonable that we might be able to cancel out like-terms and reduce the complexity of our computation. We may lose some of the mathematical intuition behind some of our formulae, but from a computational standpoint, it would be beneficial.

Corollary 1. *Looking at formula 2.14, we can eliminate like-terms to obtain*

$$\lambda_{0,l}^{m_1,l_2} = \frac{[k+1]_q J_0 \left(\sum_{s=m}^{M} \frac{f(s+1)!}{\Pi_{1\le i\le 4} f(s-a_i)! \Pi_{1\le j\le 3} f(b_j-s)!}\right)^2}{\left[\frac{m_1+k+l_1+2}{2}\right]_q! \left[\frac{m_2+k+l_2+2}{2}\right]_q! \left[\frac{m_2+l+l_1+2}{2}\right]_q! \left[\frac{l_2+l+m_1+2}{2}\right]_q!}, \tag{2.18}$$

where J_0 is the unsigned version of J in formula 5.

Proof. By formula 8,

$$\lambda_{0,l}^{m_1,l_2} = \frac{[k+1]_q Tet\left[\begin{smallmatrix} m_1 & l_1 & l \\ m_2 & l_2 & k \end{smallmatrix}\right]^2}{\theta_q(l_1,m_1,k)\theta_q(l_2,m_2,k)\theta_q(m_1,l_2,l)\theta_q(l_1,m_2,l)} =$$

$$\frac{[k+1]_q \frac{J^2}{E^2} \left(\sum_{s=m}^{M} \frac{f(s+1)!}{\Pi_{1\le i\le 4} f(s-a_i)! \Pi_{1\le j\le 3} f(b_j-s)!}\right)^2}{\left(\frac{\left[\frac{m_1+k-l_1}{2}\right]! \left[\frac{m_1-k+l_1}{2}\right]! \left[\frac{-m_1+k+l_1}{2}\right]! \left[\frac{m_1+k+l_1+2}{2}\right]!}{[l_1]_q [m_1]_q [k]_q}\right) \cdots \left(\frac{\left[\frac{m_2+l-l_1}{2}\right]! \left[\frac{m_2-l+l_1}{2}\right]! \left[\frac{-m_2+l+l_1}{2}\right]! \left[\frac{m_2+l+l_1+2}{2}\right]!}{[l_1]_q [m_1]_q [k]_q}\right)} =$$

$$\frac{\left([l_1]_q [l_2]_q [m_1]_q [m_2]_q [l]_q [k]_q\right)^2 [k+1]_q J^2 \left(\sum_{s=m}^{M} \frac{f(s+1)!}{\Pi_{1\le i\le 4} f(s-a_i)! \Pi_{1\le j\le 3} f(b_j-s)!}\right)^2}{E^2 \left[\frac{m_1+k-l_1}{2}\right]! \left[\frac{m_1-k+l_1}{2}\right]! \left[\frac{-m_1+k+l_1}{2}\right]! \left[\frac{m_1+k+l_1+2}{2}\right]! \ldots \left[\frac{m_2+l-l_1}{2}\right]! \left[\frac{m_2-l+l_1}{2}\right]! \left[\frac{-m_2+l+l_1}{2}\right]! \left[\frac{m_2+l+l_1+2}{2}\right]!}. \tag{2.19}$$

But $E^2 =$

$$(f(m_1)!f(l_1)!f(m_2)!f(l_2)!f(l)!f(k)!)^2 =$$
$$\left((-1)^{m_1+l_1+m_2+l_2+l+k-6}\right)^2 ([l_1]_q[l_2]_q[m_1]_q[m_2]_q[l]_q[k]_q)^2 = ([l_1]_q[l_2]_q[m_1]_q[m_2]_q[l]_q[k]_q)^2, \tag{2.20}$$

and $J^2 =$

$$(f(\frac{m_1+k-l_1}{2})!f(\frac{m_1-k+l_1}{2})!f(\frac{-m_1+k+l_1}{2})!f(\frac{m_2+k-l_2}{2})!f(\frac{m_2-k+l_2}{2})!f(\frac{-m_2+k+l_2}{2})!$$
$$f(\frac{m_2+l-l_1}{2})!f(\frac{m_2-l+l_1}{2})!f(\frac{-m_2+l+l_1}{2})!f(\frac{l_2+l-m_1}{2})!f(\frac{l_2-l+m_1}{2})!f(\frac{-l_2+l+m_1}{2})!)^2 =$$
$$\left((-1)^{m_1+m_2+l_1+l_2+l+k}\right)^2 \left(\left[\frac{m_1+k-l_1}{2}\right]!\dots\left[\frac{-m_2+l+l_1}{2}\right]!\right)^2 = \left(\left[\frac{m_1+k-l_1}{2}\right]!\dots\left[\frac{-m_2+l+l_1}{2}\right]!\right)^2. \tag{2.21}$$

Hence,

$$\lambda_{0,l}^{m_1,l_2} = \frac{[k+1]_q J_0\left(\sum_{s=m}^{M} \frac{f(s+1)!}{\Pi_{1\le i\le 4} f(s-a_i)!\Pi_{1\le j\le 3} f(b_j-s)!}\right)^2}{\left[\frac{m_1+k+l_1+2}{2}\right]!\left[\frac{m_2+k+l_2+2}{2}\right]!\left[\frac{m_2+l+l_1+2}{2}\right]!\left[\frac{l_2+l+m_1+2}{2}\right]!}. \tag{2.22}$$

□

Using the formulae above, we now calculate the Von Neumann entropy for individual quantum channels and tensored quantum channels, varying our values for parameters $k, l,$ and m such that $0 \le k, l, m \le 30$ for individual quantum channels and $0 \le k, l, m \le 20$ for tensor quantum channels. For the purposes of visualizing how measured entropies for channels vary as $k, l,$ and m vary, we let $l_1 = l_2$ and $m_1 = m_2$ naively when we calculate the entropy for tensor quantum channels.

We implement all the above-mentioned formulae in Python using an open-source, arbitrary-precision arithmetic library called *mpmath* which enables us to compute output entropies for channels with large choices of $k, l,$ and m. The code we used can be found in the appendix.

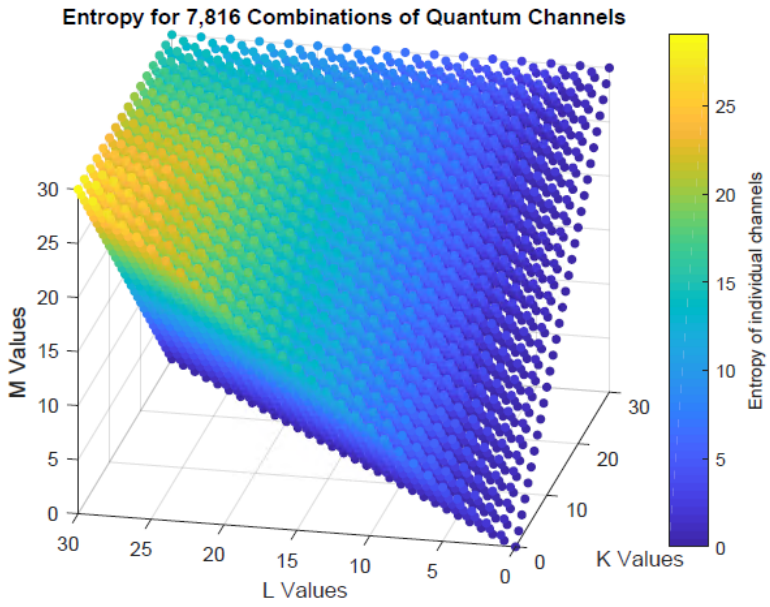


Figure 2.4: Minimum Output Entropies of Individual Quantum Channels

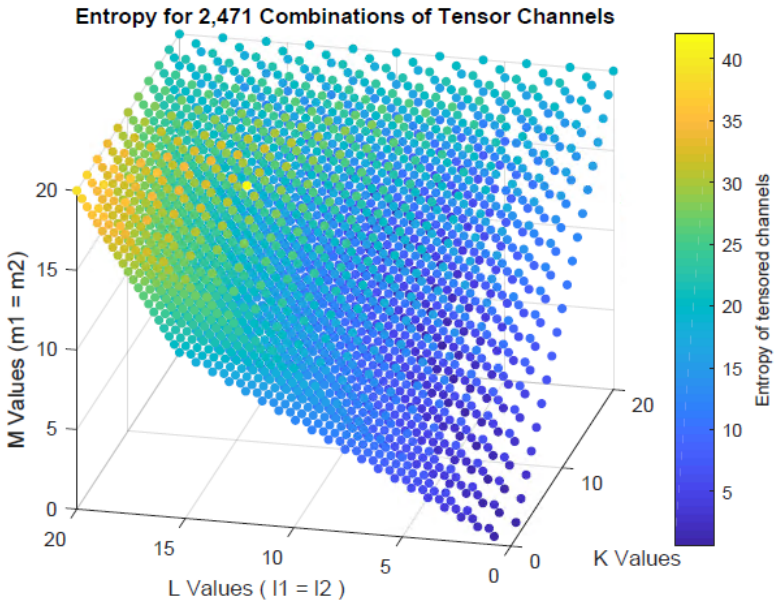


Figure 2.5: Minimum Output Entropies of Tensor Quantum Channels

The above figures allows us to visualize how the minimum output entropies for our individual channels and tensor channels change as we vary our choices for $k, l,$ and m. The way to interpret the figures in terms of obtaining a violation is that we want to find $k, l,$ and m such that the corresponding color in figure 2.4 is significantly *warmer* than the corresponding color in figure 2.5.

Unfortunately, it is clear that the minimum output entropies for both our individual channels and tensor channels increase proportionally. Ideally, we could find a choice for $k, l,$ and m such that this was not the case. Note that in figure 2.5, letting $k = 12$ and $l = m = 15$, the corresponding quantum channel has an unusually large entropy. However, this is the *opposite* of what we want. We would like to find choices for $k, l,$ and m such that the corresponding entropy in figure 2.4 is unusually high.

2.3 Renyi Entropy

In fact, the von Neumann entropy is a special case of a more general entropy formula called the p-Renyi entropy.

Formula 11. (Renyi Entropy of a state) *The Renyi entropy of a state ρ with parameter $p \in (0, \infty)$ is denoted by S_p and has formula*

$$S_p(\rho) = \frac{1}{1-p} log\Big(\sum_{\lambda} \lambda^p\Big) \tag{2.23}$$

Note that the Von Neumann entropy is equivalent to the Renyi entropy in the limit where $p \to 1$. Furthermore, all other formulae found in section 2.1 are used identically for both the Renyi and Von Neumann entropies.

Using the formulae above, we now calculate the Renyi entropy estimates for tensor product quantum channel outputs (similarly as to how we did in section 2.2), varying our values for parameters $k, l,$ and m such that $0 \leq k, l, m \leq 20$ for tensor quantum channels.

Note that we do not calculate the Renyi entropy for individual tensor channels because the formula for calculating such channels is unchanged:

$$S_{p_{min}}(\Phi) = \frac{1}{1-p} log(\sum_{\lambda} \lambda^p) = \frac{1}{1-p} log(\sum_{\lambda} \lambda \lambda^{p-1}) \geq$$

$$\frac{1}{1-p} log(\sum_{\lambda} \lambda \lambda_{max}^{p-1}) = \frac{1}{1-p} log(\lambda_{max}^{p-1}) = \frac{p-1}{1-p} log(\lambda_{max}) = log(\frac{1}{\lambda_{max}}).$$

The two seemingly different cases for the inequality, namely $p \leq 1$ and $p \geq 1$, actually turn out to have the same overall number of sign changes. This is because the loss of a minus sign from $\frac{1}{1-p}$ when p moves from less than 1 to greater than 1 is picked up by the exponent of λ inside the logarithm after we factor out one factor of λ changing from greater than 1 to less than 1.

For the purposes of visualizing how measured entropies for channels vary as k, l, and m vary, we let $l_1 = l_2$ and $m_1 = m_2$ naively when we calculate the entropy for tensor quantum channels. We will consider a couple choices for p $(0.5, 5)$ to observe how the measured entropy varies based on our choice for the dimension. Like for the von Neumann entropy, the implementation was handled in Python, and the code we used can be found in the appendix.

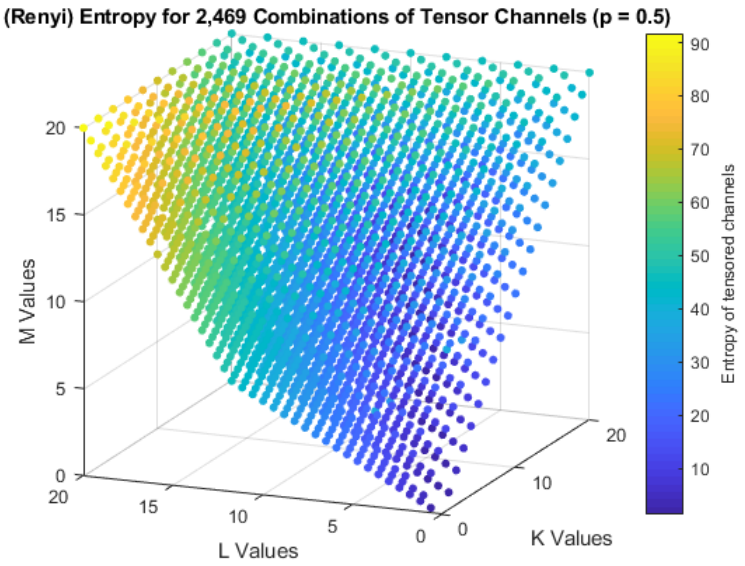


Figure 2.6: Minimum Output Entropy of Tensor Quantum Channels w.r.t the Renyi Entropy for $p = 0.5$

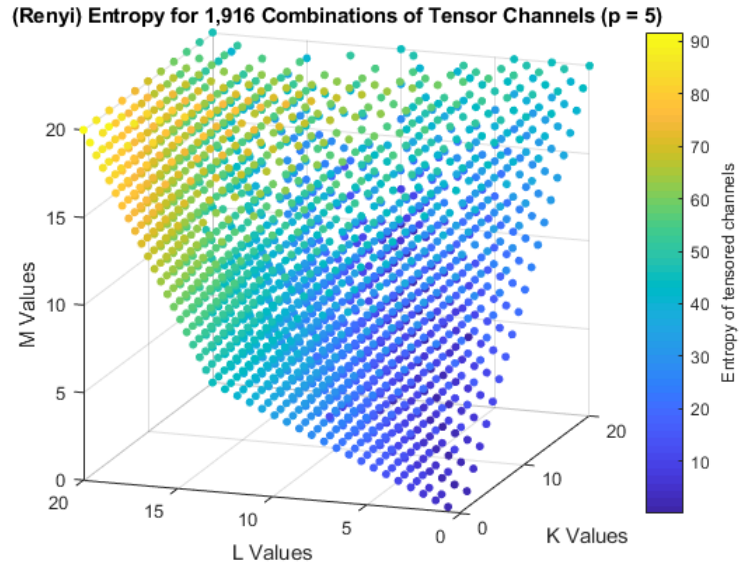


Figure 2.7: Minimum Output Entropy of Tensor Quantum Channels w.r.t the Renyi Entropy for $p = 5$

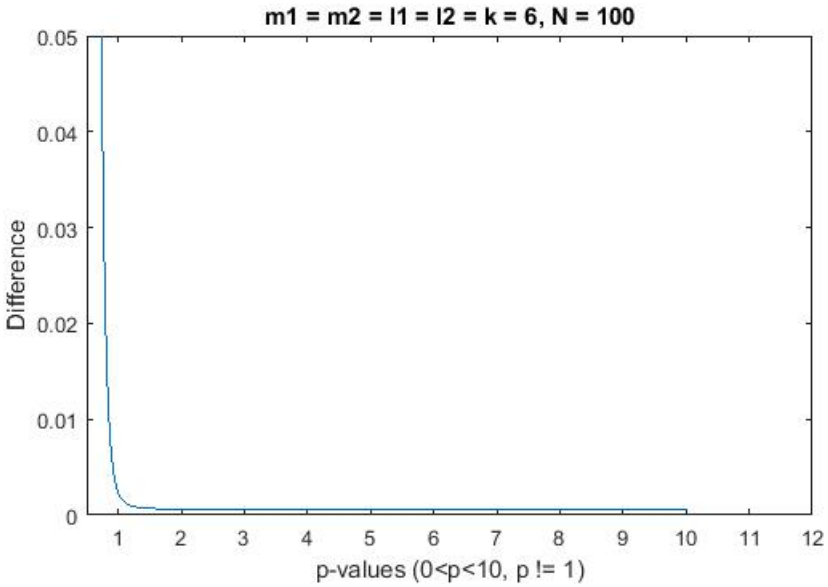


Figure 2.8: Minimum Output Entropy for a Specific Quantum Channel w.r.t the Renyi Entropy, Varying $0 < p < 10, p \neq 1$

We observe that our choice of $p = 0.5$ in figure 2.6 yields greatly increased minimum output entropies uniformly across the domain we've considered; furthermore, looking at figure 2.8, it is clear that for $0 < p < 1$, our measured minimum output entropy is less optimal than in the case of the von Neumann entropy (such is the case for any choice of quantum channel). As we continue to increase our choice of p past 1, however, we observe that the difference in our Renyi entropy estimates for our individual and tensor channels becomes slightly more optimal than our previous estimates with respect to the von Neumann entropy. This change is reflected in figure 2.7 where we let $p = 5$.

3. COMPUTING ENTROPIES WITH TEMPERLEY-LIEB CATEGORY THEORY

3.1 Computing Entropy Estimates for *Planar* Channels

In the planar calculus of Temperley-Lieb category theory, we have a correspondence between certain planar graphs and matrices. In definition 4 of chapter 1, we introduce the *Stinespring representation* of a map $\Phi : End(H) \longrightarrow End(K)$ and its isometry $\alpha : H \longrightarrow K \otimes E$.

The evaluation of a quantum channel at a particular state can be represented diagramatically using Temperley-Lieb category theory. Figure 2.2 is an example of such an isometry, where the upper ends of the diagram are the range and the lower ends are the domain. We will show how such a diagram can be formed step by step by its constituent maps as follows.

First, α is a map from H to $K \otimes E$. Thus, it has the following abstract diagram.

Figure 3.1: Temperley-Lieb Diagrammatic Representation [1] of $\alpha : H \longrightarrow K \otimes E$

In figure 3.1, the input space is drawn at the bottom and the output spaces are drawn at the top. This will be the rule for all diagrams. The Temperley-Lieb planar algebra has as its multiplication operation composition of diagrams simply by placing diagrams above each other and connecting the strands that match up. Now α^*, the adjoint of α, has the same diagram as α except it is upside down, so the total diagram for $\Phi_k^{\bar{l},m}$ is as follows.

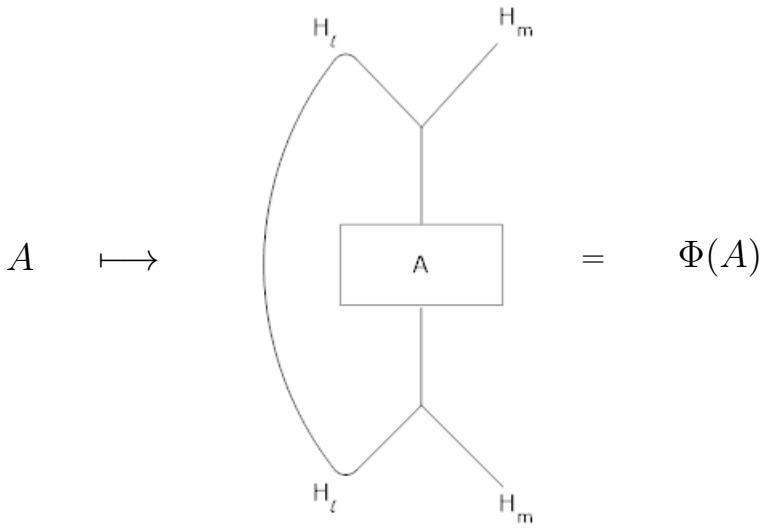


Figure 3.2: Temperley-Lieb Diagrammatic Representation [1] of $\Phi_k^{\bar{l},m}$

The left most strand in figure 3.2 corresponds to the partial trace. The box labeled A represents a density operator input for $\Phi_k^{\overline{l_1},m_1}$. Putting two diagrams next to each other corresponds to taking tensor products of the corresponding maps, so the diagram for the final tensor product channel with input $\rho = A$ is as follows.

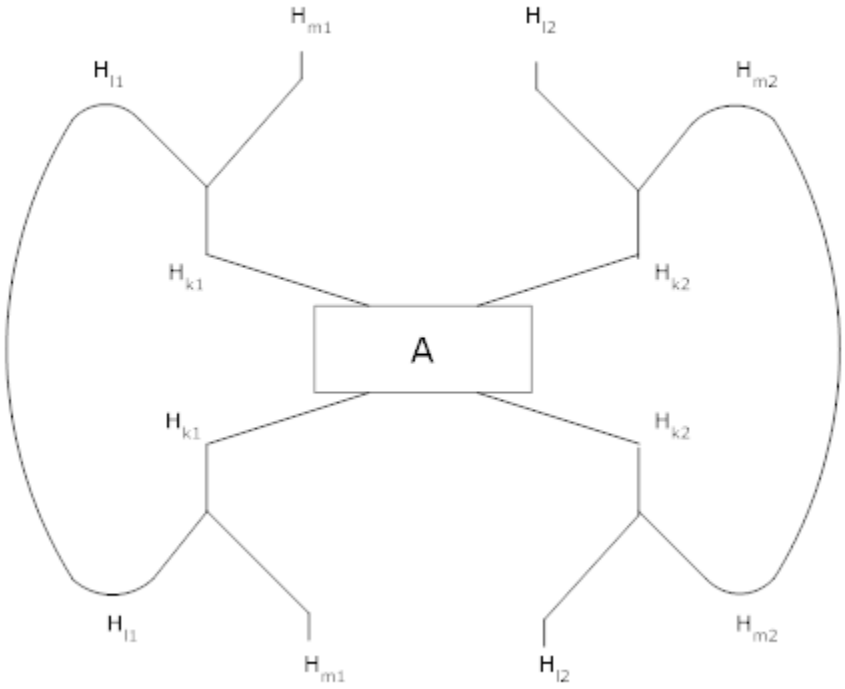


Figure 3.3: Temperley-Lieb Diagrammatic Representation [1] of $\Phi_k^{\overline{l_1},m_1} \otimes \Phi_k^{l_2,\overline{m_2}}$

Here in figure 3.3, the box labeled A instead represents a density operator on the tensor product Hilbert space as input for $\Phi_k^{\overline{l_1},m_1} \otimes \Phi_k^{l_2,\overline{m_2}}$ whereas, before, it was a density operator on an individual Hilbert space.

The diagrams we've given thus far are actually abstractions of the true diagrams which are planar diagrams without intersections as we've drawn them. There are three main simplifications made in drawing these abstract diagrams. First a single strand as we've drawn it stands for some number of k of parallel strands where k stands for the Hilbert space H_k. Second, an intersection of strands as we've drawn them only represents some grouping of parallel strands diverging from each other and leading to different spaces (the

real strands never cross since these are planar diagrams). Third, the isometries α are actually linear combinations of a few different diagrams we give later, so one must use the distributive law of the algebra to get the actual diagrams.

To proceed towards giving the true diagrams for the quantum channels with Bell state inputs explicitly, we first give some more theory behind the structure of the Temperley-Lieb algebra.

The main interest in using the Temperley-Lieb algebra is that there is a correspondence between diagrams of the Temperley-Lieb algebra and their associated matrices.

As a simple example, recall the abstract diagram we gave for α. In the case where the dimension of E and K are 1 and the dimension of H is 2, there are two possible planar smoothings of this diagram. In fact, the first diagram corresponds to the identity map normalized to have a trace of 1 and the second diagram corresponds to the Bell state $\rho = |\psi\rangle\langle\psi|$, where $|\psi\rangle = \dfrac{1}{[k+1]_q^{\frac{1}{2}}} \sum_i e_i \otimes f_i$ such that $(e_i)_i$ and $(f_i)_i$ are orthonormal bases of H_l and H_m respectively.

Also, whenever a loop is created from multiplying two diagrams, that corresponds to multiplying by a scalar $N = [k+1] = dim(H_k)$. Then α is actually a particular linear combination of these maps $K \longrightarrow E \otimes K$.

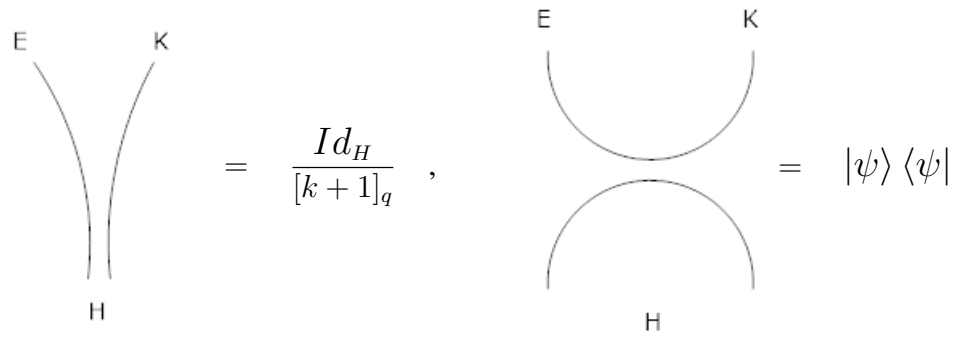


Figure 3.4: Elementary Tangles for $H = 2$ and $E = k = 1$

These two simple cases from figure 3.4 are fundamental because, in moving to a higher numbers of strands, the analogues of these diagrams are generators for the Temperley-Lieb algebra on n strands.

Definition 10. *The elementary tangles $U_1, U_2, ..., U_{n-1} \in T_n$ where T_n denotes the Temperley-Lieb algebra on n-strands are the diagrams such that for U_i, the k^{th} input is connected to the k^{th} output for $k \neq i, i+1$ while the i^{th} input is connected to the $(i+1)^{th}$ input and the i^{th} output is connected to the $(i+1)^{th}$ output [5].*

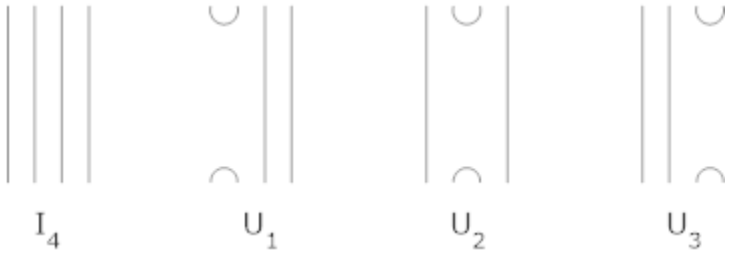


Figure 3.5: The Elementary Tangles of the Temperley-Lieb Algebra T_4

Figure 3.5 above shows the elementary tangles on 4 strands along with the identity diagram. The elementary tangles generate the full Temperley-Lieb algebra and have the following properties [5]:

1. $U_i^2 = NU_i$.
2. $U_iU_{\pm 1}U_1 = U_i$.
3. $U_iU_j = U_jU_i$ for $i \neq j$.

From the definition of the elementary tangles, it can be proven that there is a unique non-zero element f of the Temperley-Lieb algebra T_n for a particular number of strands n such that both [5]

1. $f^2 = f$, i.e. f is a projection, and

2. $fU_i = U_i f = 0$ for $i = 1, 2, ..., n-1$.

These projections, one for each n, are called the Jones-Wenzl projections and have the following diagrammatic recursion relation:

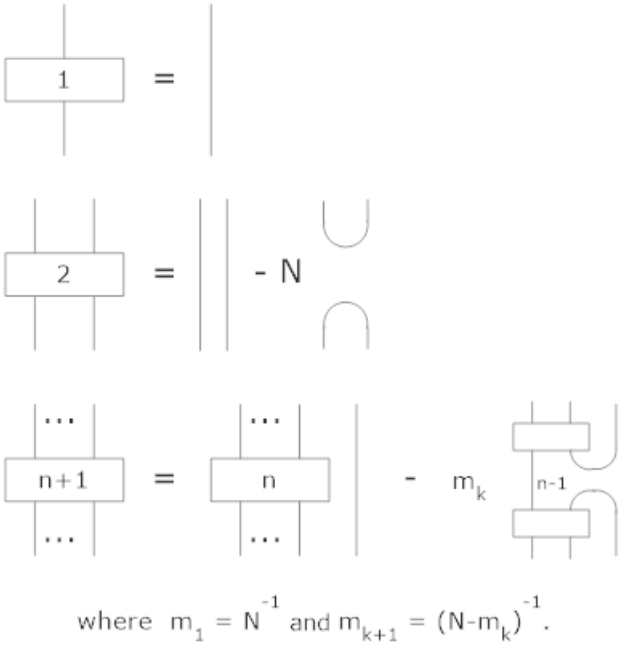


Figure 3.6: Recurrence Relation of Jones-Wenzl Projections

These Jones-Wenzl projections shown in figure 3.6 are used in defining the actual form of the $\alpha_k^{l,m}$ isometries. The transition from the abstract original diagram to the real diagram is given below. We previously stated $\alpha_k^{l,m} = \left(\frac{[k+1]_q}{\theta_q(k,l,m)}\right)^{\frac{1}{2}} A_k^{l,m}$. The following diagrams are for $A_k^{l,m}$ since we omit coefficients for brevity.

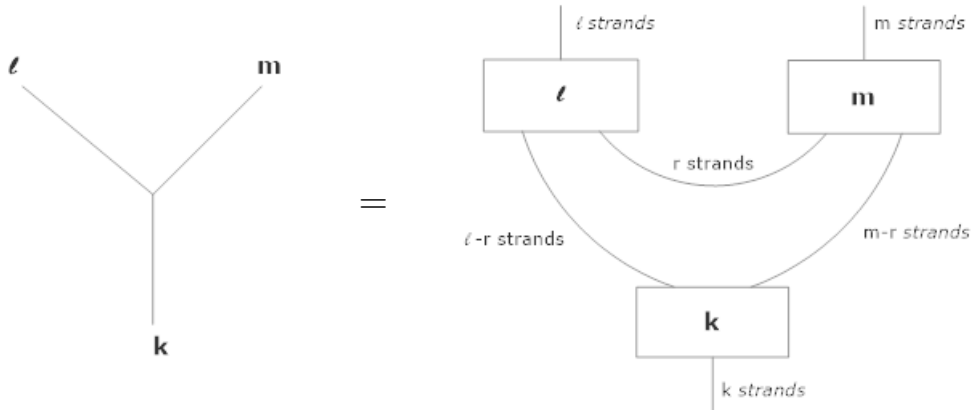


Figure 3.7: Temperley-Lieb Diagrammatic Representation of $A_k^{l,m}$

With those abstractions sorted out (as in figure 3.7), we can now give an explicit calculation of the output entropy of one of our quantum channels with Bell state input.

First, the Bell state input corresponds to replacing the box labelled A with the "cup-cap" diagram.

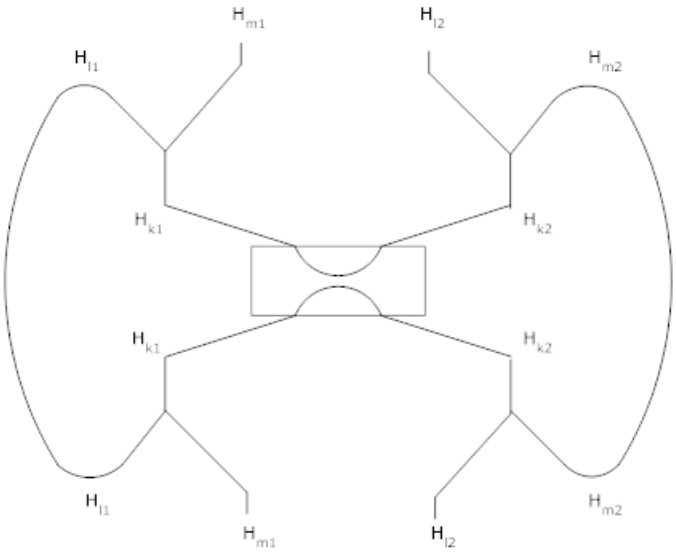


Figure 3.8: $\Phi_k^{\overline{l_1},m_1} \otimes \Phi_k^{l_2,\overline{m_2}}$ with Bell State

Next, we clarify the actual form of the $\alpha_k^{l,m}$ isometry sub-diagrams in figure 3.8 by including the Jones-Wenzl projections in the diagram. However, we omit the inclusion of the trivial Jones-Wenzl projections in the picture since they are just a single strand.

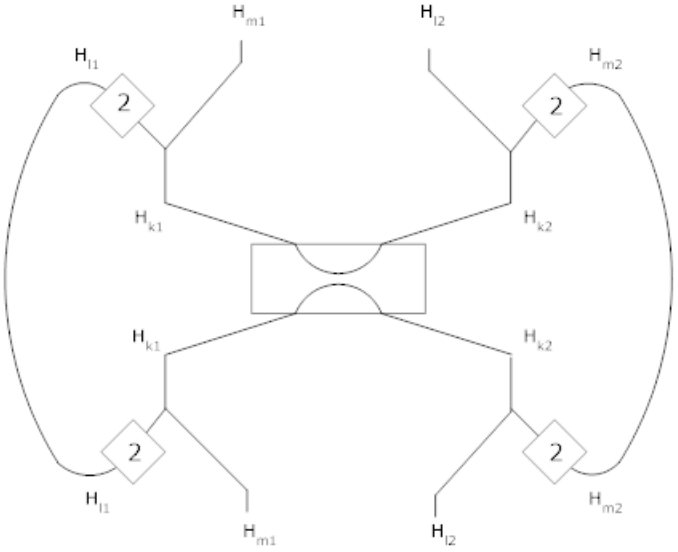


Figure 3.9: $\Phi_k^{\overline{l_1},m_1} \otimes \Phi_k^{l_2,\overline{m_2}}$ with Bell State and Projections

Next, since when diagrams or subdiagrams are concatenated vertically as shown in figure 3.9, that corresponds to composition of the corresponding maps, and the Jones Wenzl projections square to themselves, (that is, they are projections), we can combine superfluous projections in the diagrams.

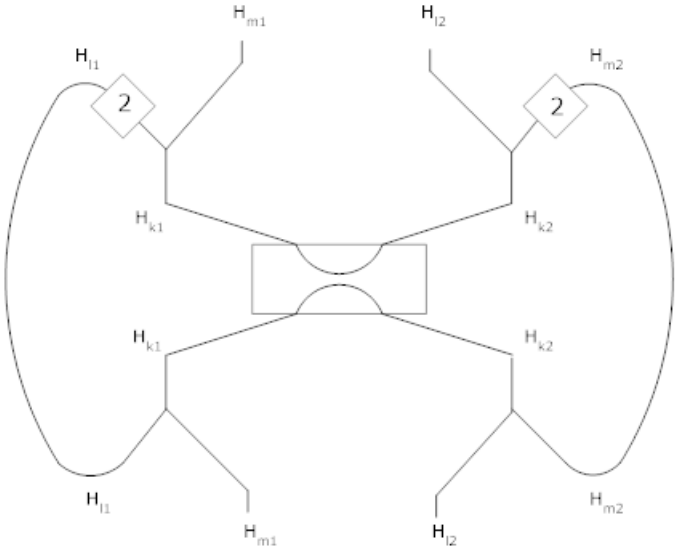


Figure 3.10: $\Phi_k^{\overline{l_1},m_1} \otimes \Phi_k^{l_2,\overline{m_2}}$ with Bell State and Simplified Projections

Lastly, it is a straightforward computation to perform the distribution of multiplication over addition in the algebra to expand figure 3.10 into all of its terms arising from the fact that the Jones-Wenzl projections are, in general, linear combinations of individual planar diagrams. Performing the computation yields that the above diagram simplifies to:

Using the correspondances between planar diagrams and matrices described earlier, this allows us to calculate the matrix corresponding to this quantum channel output and compute its eigenvalues. In general, for $N \in \mathbb{N}$ and ρ a Bell state, we compute the resulting matrix of

$$\frac{[k+1]_q}{\theta(1,2,1)^2}\left(\left(N-\frac{2}{N}\right)Id_{[k+1]_q^2}+\frac{1}{N}\rho_{[k+1]_q^2}\right),$$

where the coefficient $\dfrac{[k+1]_q}{\theta(1,2,1)^2}$ comes from the coefficients of $\alpha_k^{l,m}$ and the input ρ.

Letting $N = 3$, we evaluate

$$\frac{3}{64}\left(\frac{7}{3}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} + \frac{1}{9}\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}\right)$$

and get the matrix

$$\begin{pmatrix} \frac{11}{96} & 0 & 0 & 0 & \frac{1}{192} & 0 & 0 & 0 & \frac{1}{192} \\ 0 & \frac{7}{64} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{7}{64} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{7}{64} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{192} & 0 & 0 & 0 & \frac{11}{96} & 0 & 0 & 0 & \frac{1}{192} \\ 0 & 0 & 0 & 0 & 0 & \frac{7}{64} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{7}{64} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{7}{64} & 0 \\ \frac{1}{192} & 0 & 0 & 0 & \frac{1}{192} & 0 & 0 & 0 & \frac{11}{96} \end{pmatrix}$$

with eigenvalues $\{\frac{1}{8}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}\}$ in agreement with the formula 8 from chapter 2.

3.2 Computing Entropy Estimates for *Nonplanar* Channels

So far, we have only computed minimum output entropy estimates for planar diagrams, and none have violated additivity. We now look to compute minimum output entropies for nonplanar diagrams. For example, the channel $\Phi_1^{\overline{2},1} \otimes \Phi_1^{2,\overline{1}}$ is planar, and the channel $\Phi_1^{\overline{2},1} \otimes \Phi_1^{\overline{2},1}$ is nonplanar.

Nonplanar diagrams do not directly have any matrix interpretation. However, we refer to *Temperley-Leib Recoupling Theory and Invariants of 3-Manifolds* [5] by Louis H. Kauffman [5] which gives relations for expressing the nonplanar crossings in a diagram as a weighted combination of their two possible planar smoothings:

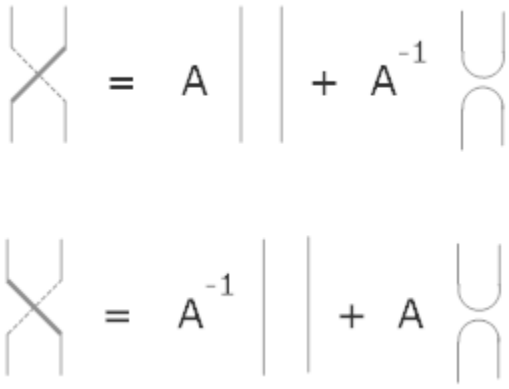


Figure 3.11: Relations between Crossing Strands and Planar Tangles

In figure 3.11, $q = -A^2$. Since homotopically-equivalent diagrams are the same, we systematize drawing nonplanar diagrams in such a way that can be easily generalized for any values of k, l, and m. In the following diagram, we redraw $\Phi_1^{\overline{2},1} \otimes \Phi_1^{\overline{2},1}$ to be more structured.

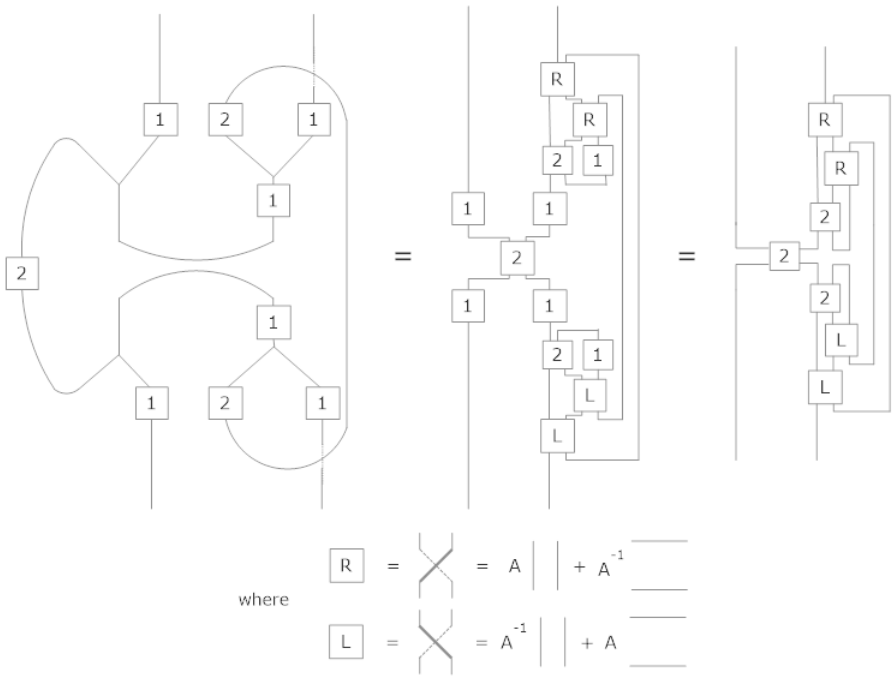


Figure 3.12: Rewriting Diagrammatic Representation of $\Phi_1^{\overline{2},1} \otimes \Phi_1^{\overline{2},1}$

The Jones-Wenzl projections used in figure 3.12 are defined in the same way as before. Furthermore, we decompose *L* and *R* "boxes" in a similar manner to projections when we use the distributive rule to expand such diagrams into the diagrams whose corresponding matrices we know. Since the diagram just given has 7 boxes, each of which represents a linear combination of two different subdiagrams, the full expansion of this diagram has 128 terms. The full computation of the terms of this diagram is given in the appendix: we list all 128 planar smoothings of the diagram along with the coefficient associated to each term as determined by the definition of the R, L, and 2 boxes, and by the appearance of loops in the resultant diagram, each of which contributes a scalar factor of N. Throughout the appendix, the overall factor of $\dfrac{[k+1]_q}{\theta(1,2,1)^2}$ is omitted.

In this example, it turns out that after collecting terms and using the relations given for relating A, N, and q, we get the same result as in the corresponding planar case of

$\Phi_1^{\overline{2},1} \otimes \Phi_1^{1,\overline{2}}$, namely, we get eigenvalues of $\{\frac{1}{8}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}, \frac{7}{64}\}$ again. It is unknown if this happens in general.

As alluded to above, we now draw a nonplanar diagram with generalized dimensions. The following diagram illustrates this.

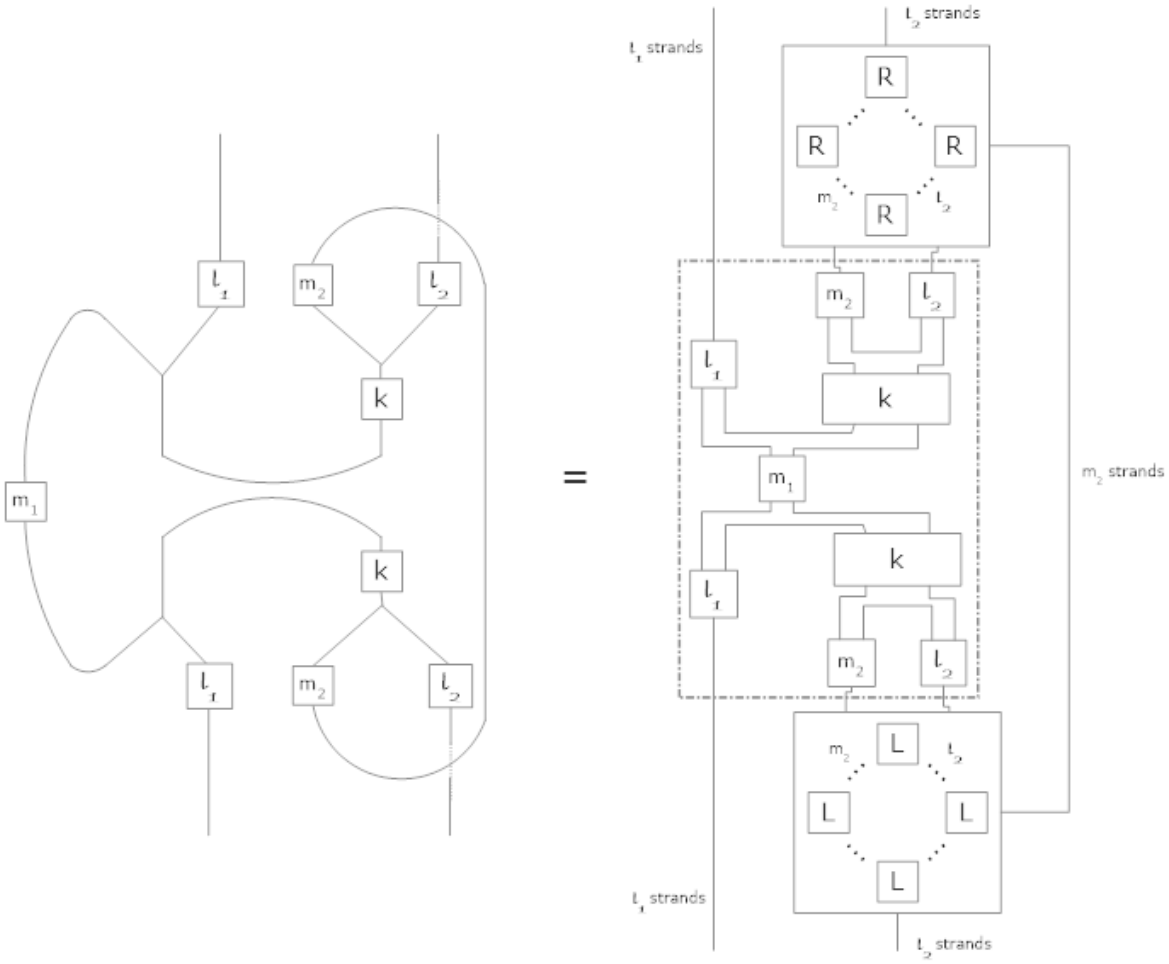


Figure 3.13: Rewriting Diagrammatic Representation of $\Phi_k^{\overline{l_1},m_1} \otimes \Phi_k^{\overline{l_2},m_2}$

Note the dashed box in figure 3.13. Given three projections connected together in a triangular fashion, there is only a single nontrivial way to connect the strands (that is, one

that avoids cups and caps on any individual projection). Within the specified box, we connect all m_1, l_1, k and m_2, l_2, k in this way.

See the appendix for an informal algorithm to compute the minimum output entropy of nonplanar diagrams.

3.3 Conclusion

In this paper, we have shown the feasibility of computing estimates of minimum output entropies for quantum channels arising from the representation theory of the O_N^+ quantum group. We explored the option of using Temperley-Lieb category theory to perform similar computations for nonplanar diagrams. In no case did we find an example of an additivity violation of the minimum output entropy as shown to exist by Hastings.

However, for quantum channels parameterized by N, we seem to be able to make the difference in output entropies for individual channels and their corresponding tensor product channels very small for some choices of channels and very large for others. For example, given admissible parameters $(10, 10, 10)$ and $(10, 10, 10)$ and a Bell state ρ, the expression

$$S\big(\Phi_{10}^{\overline{10},10} \otimes \Phi_{10}^{10,\overline{10}}(\rho)\big) - S\big(\Phi_{10}^{\overline{10},10}\big) - S\big(\Phi_{10}^{10,\overline{10}}\big)$$

evaluates to the following as N varies:

- ≈ 1.50162 for $N = 3$.
- ≈ 0.14678 for $N = 10$.
- ≈ 0.28755 for $N = 25$.
- ≈ 0.00234 for $N = 100$.
- ≈ 0.00012 for $N = 500$.
- ≈ 0.00000156 for $N = 5000$, etc.

However, given the admissible parameters $(20, 20, 2)$ and $(20, 20, 2)$ and a Bell state ρ, the expression

$$S\big(\Phi_{20}^{\overline{20},2} \otimes \Phi_{20}^{20,\overline{2}}(\rho)\big) - S\big(\Phi_{20}^{\overline{20},2}\big) - S\big(\Phi_{20}^{20,\overline{2}}\big)$$

evaluates to the following as N varies:

- ≈ 17.7537 for $N = 3$.
- ≈ 41.2943 for $N = 10$.
- ≈ 57.9157 for $N = 25$.
- ≈ 82.8916 for $N = 100$.
- ≈ 111.863 for $N = 500$.
- ≈ 153.309 for $N = 5000$.
- ≈ 194.756 for $N = 50000$, etc.

It appears that choices for tensor product channels applied to ρ whose largest eigenvalues have lower multiplicities are the most likely candidates for an additivity violation as $N \longrightarrow \infty$. In the above two examples, for $N = 3$, the eigenvalues of our resulting matrices are

Table 3.1: Eigenvalues for Resulting Matrix of $S\big(\Phi_{10}^{\overline{10},10} \otimes \Phi_{10}^{10,\overline{10}}(\rho)\big)$

Multiplicity	Value
$2.67914e+8$	$4.71227e-30$
$3.90882e+7$	$1.47775e-21$
$5.70289e+6$	$7.50079e-15$
832040	$7.72858e-10$
121393	$1.56477e-6$
17711	$3.65394e-5$
2584	$5.32273e-5$
377	$5.59843e-5$
55	$5.63937e-5$
8	$5.64535e-5$
1	$5.64621e-5$

Table 3.2: Eigenvalues for Resulting Matrix of $S\big(\Phi_{20}^{\overline{20},2} \otimes \Phi_{20}^{20,\overline{2}}(\rho)\big)$

Multiplicity	Value
$1.83631e+9$	$6.82817e-26$
$2.67914e+8$	$3.73254e-9$
$3.90882e+7$	$3.20779e-24$

respectively where tables 3.1 and 3.2 are in agreement with the above conjecture. This suggests that for certain choices of channels, the methods proposed in this paper can be fruitful in the future either for a family of quantum channels that is better than those from the O_N^+ quantum group or if estimates for either side of inequality $S^{min}(\Phi_1 \otimes \Phi_2) < S^{min}(\Phi_1) + S^{min}(\Phi_2)$ are improved.

REFERENCES

[1] S. Morrison, "A Formula for the Jones-Wenzl Projections," Mar. 2015.

[2] M. A. Nuwairan, *SU(2)-Irreducibly Covariant Quantum Channels and Some Applications*. PhD thesis, University of Ottawa, 2015.

[3] M. Brannan and B. Collins, "Highly Entangled, Non-Random Subspaces of Tensor Products from Quantum Groups," Dec. 2016.

[4] M. B. Hastings, "Superadditivity of Communication Capacity Using Entangled Inputs," Dec. 2009.

[5] L. H. Kauffman and S. L. Lins, *Temperley-Leib Recoupling Theory and Invariants of 3-Manifolds*, vol. Annals of Mathematics Studies. Princeton University Press, 1994.

APPENDIX

A.1 Algorithm to Compute Minimum Output Entropies of Nonplanar Channels

Using Temperley-Lieb category theory in conjunction with basic graph theory, we propose an algorithm to systematically compute the entropy of nonplanar channels by decomposing our nonplanar channels into equivalent planar representations.

We want to replace all *R* and *L* boxes and projections in our Temperley-Lieb representations of nonplanar channels with sets of vertices and corresponding adjacency lists in a bipartite graph G. Fortunately, every individual resulting planar decomposition of our nonplanar channel has the following properties that alleviate some of the complexity involved in implementing Temperley-Lieb planar calculus with graph theory:

1. G is always disconnected.

2. Every vertex corresponding to a projection, *R* box, or *L* box will have exactly two adjacent vertices. Input and output vertices will have exactly one adjacent vertex.

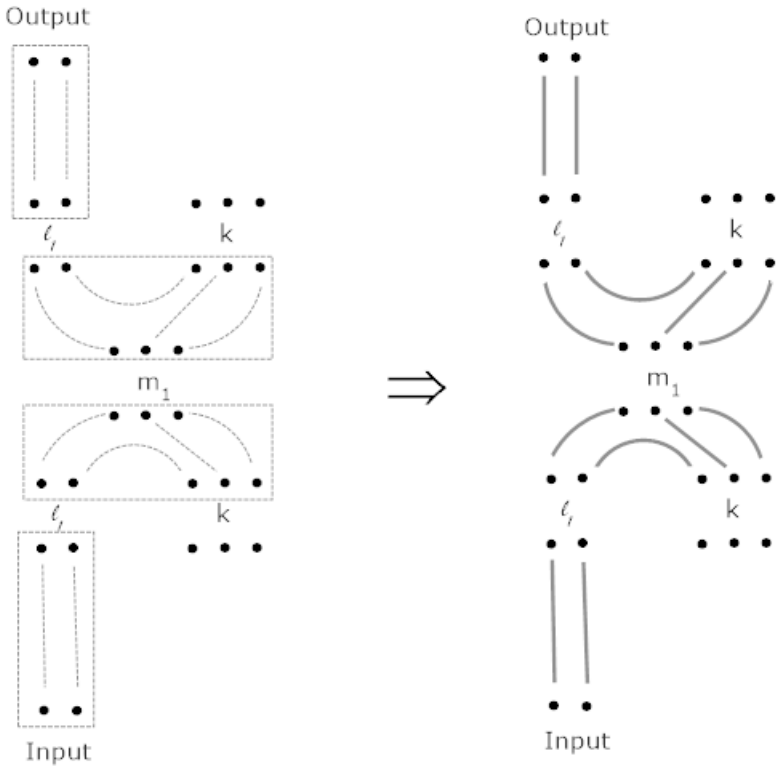
3. If we perform a depth-first search starting at an input or output vertex, we must end at an input vertex or output vertex. If we start at an input vertex and end at an output vertex (or vice versa), we have a straight line. Else, we start and end at an input or output vertex and obtain either a cap or cup, respectively. Note that any type of searching algorithm will suffice since our starting vertices have a single adjacent vertex, and all subsequent visited vertices will have only a single unvisited vertex.

4. If after all input and output vertices have been depth-first searched there exist unvisited nodes in our graph G, then we necessarily have loops. If we then perform depth-first searches on arbitrary unvisited nodes until all nodes in G are visited, then

we obtain a number of constants equal to the number of depth-first searches we were required to perform.

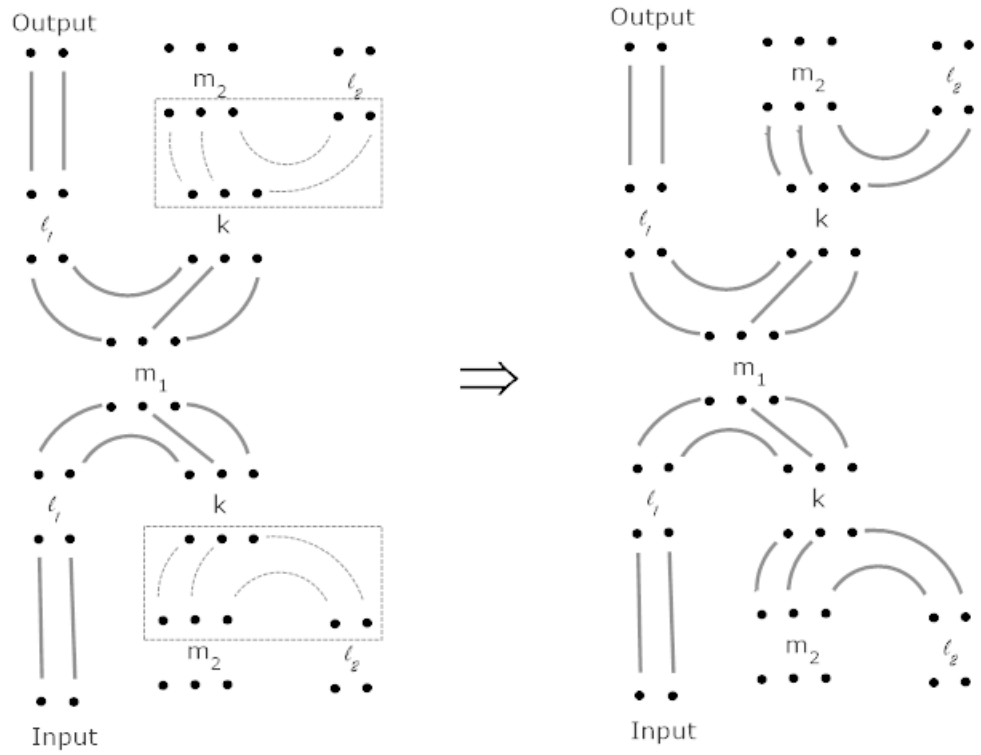
For the sake of clear understanding, we simultaneously work through an example with a slightly more complicated channel than previously mentioned, $\Phi_3^{\overline{3},2} \otimes \Phi_3^{\overline{3},2}$. Note that when we say *connect* two vertices, we mean appending each vertex's *identifier* to the other vertex's adjacency list.

1. Create 4 vertices for all *R* and *L* boxes (of which there are $m_2 * l_2$ of each), $2 * P$ vertices for each projection P, and an additional $2 * (l_1 + l_2)$ vertices for inputs and outputs. Instantiate all vertices such that each vertex has a unique identifier.



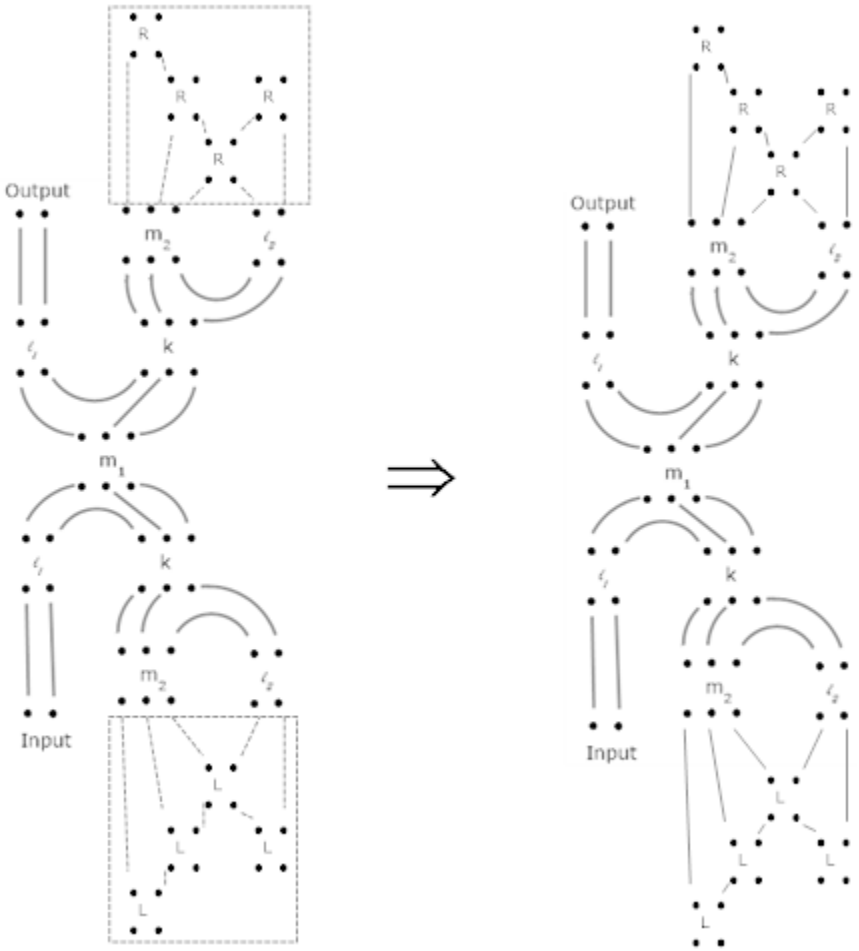
We omit unique identifiers associated with each vertex.

2. Begin with the m_1 projection. *Place* l_1 and k projections above and below as shown. Treat each dashed box as a separate problem. Because we can only connect three projections in a triangular fashion a single way, connect vertices accordingly. Then connect remaining vertices on either l_1 projection to input and output vertices.



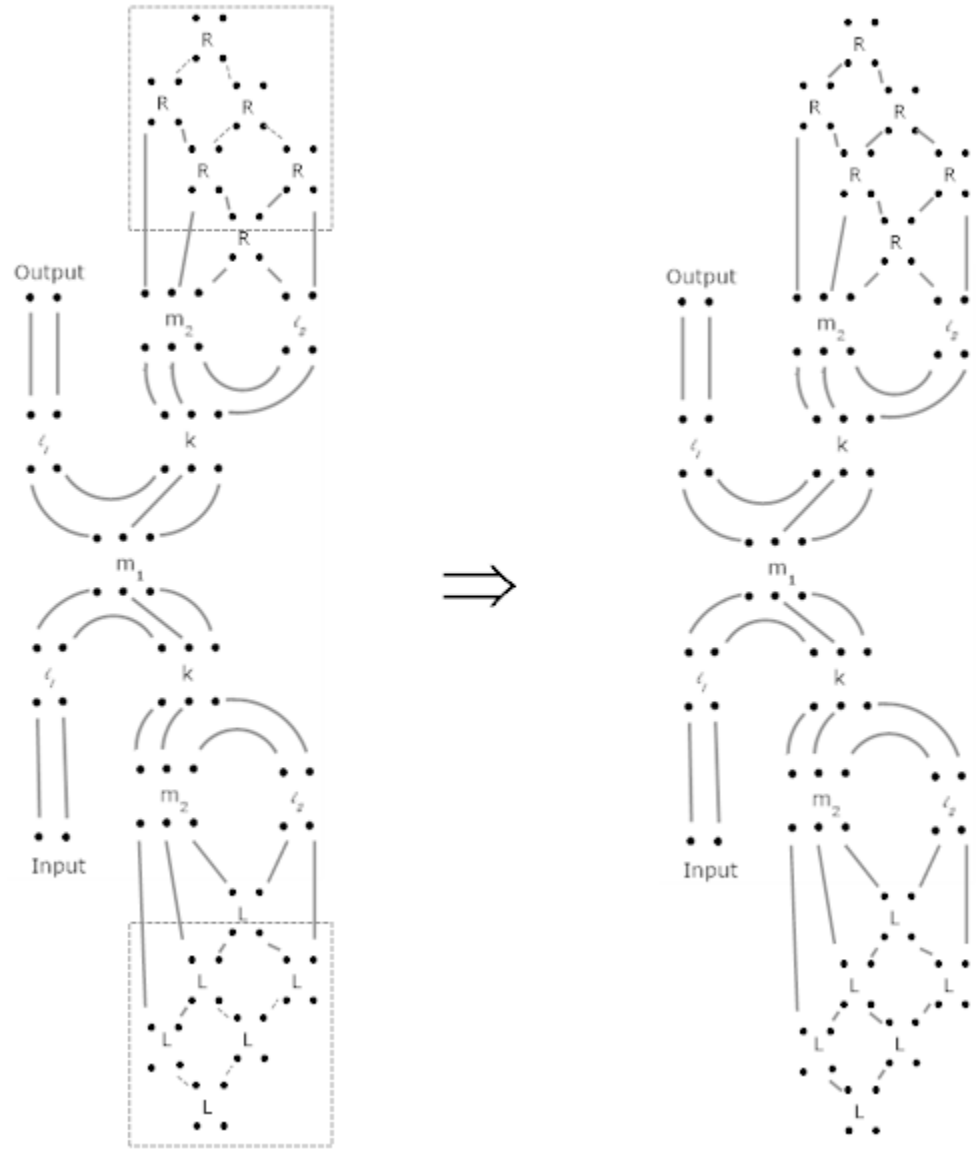
At this point, all projections have been handled.

3. Again, place m_2 and l_2 projections above and below the k projections as shown. Repeat same logic from step 2.

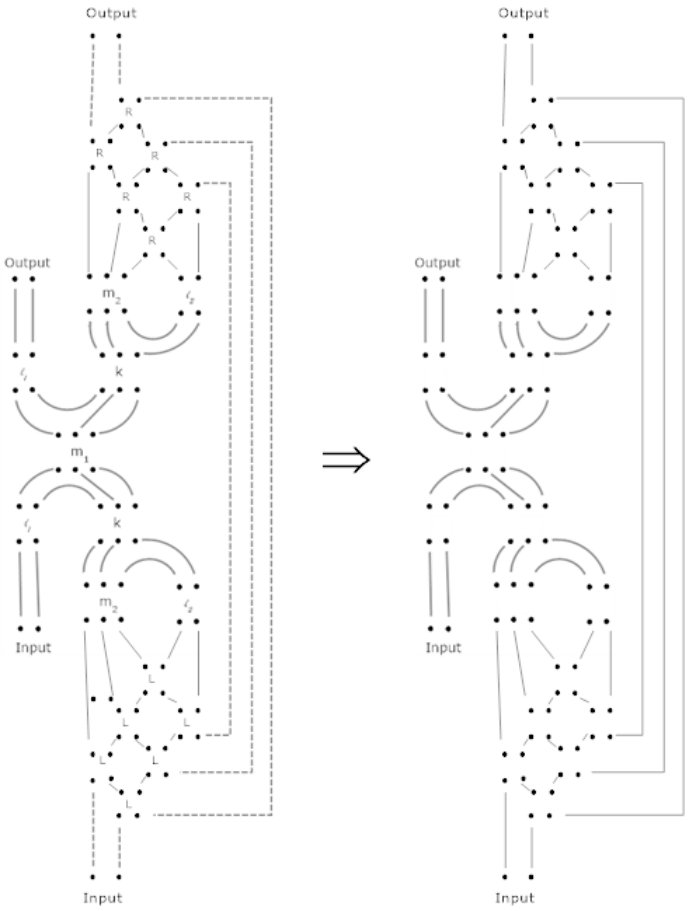


Each collection of *R* and *L* boxes are the start of the $m_2 \times l_2$ matrix from figure 3.10.

4. Connect remaining vertices on either m_2 and l_2 projections to *R* and *L* boxes, respectively, in the manner shown. Then connect corners of adjacent *R* and *L* boxes naturally.



5. Complete the $m_2 \times l_2$ matrix of *R* and *L* boxes (see figure 3.10), and connect adjacent corners of *R* and *L* boxes naturally.

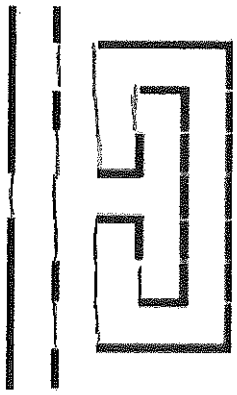


We remove labels to make more clear where we are to iterate over possible configurations.

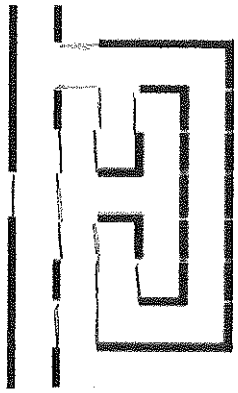
6. Connect all remaining vertices of each *R* and *L* box with eachother and any remaining input and output vertices (do **not** connect *R* boxes to *R* boxes and vice versa). Note that there is only one way to do this in a planar fashion that avoids cups and caps. The resulting graph is our intermediate graph G' from which we then construct all possible configurations of our Temperley-Lieb diagram.

A.2 Expanding Terms of $\Phi_1^{\overline{2},1} \otimes \Phi_1^{\overline{2},1}(\rho)$

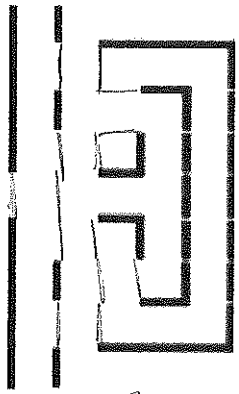
The following is the expansion of the nonplanar diagram $\Phi_2^{\overline{2},1} \otimes \Phi_1^{\overline{2},1}(\rho)$ into planar subdiagrams with coefficients included.



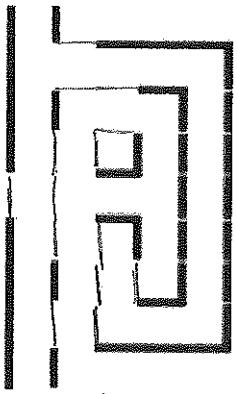
N Id



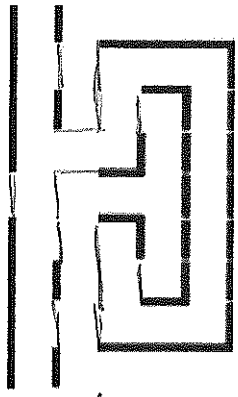
A^{-2} Id



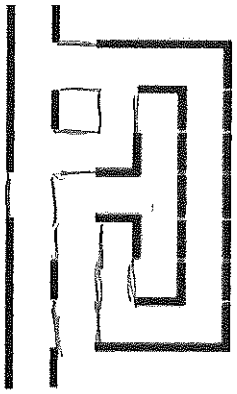
$N^2 A^{-2}$ Id



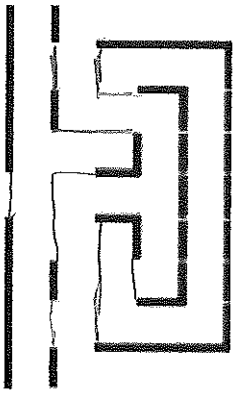
$N A^{-4}$ Id



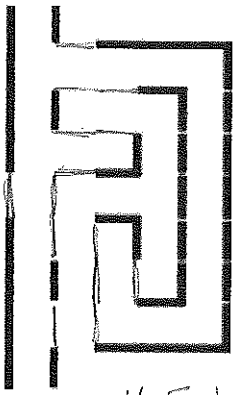
$-N^{-1}$ Id



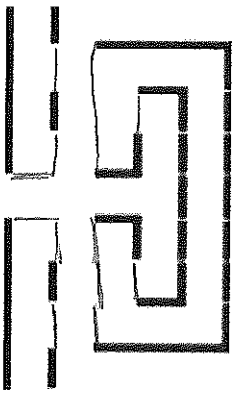
$-A^{-2}$ Id



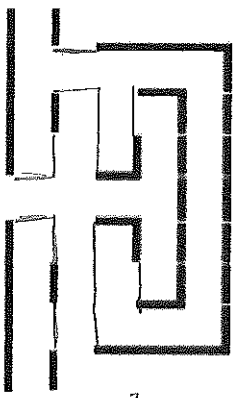
$-A^{-2}$ Id



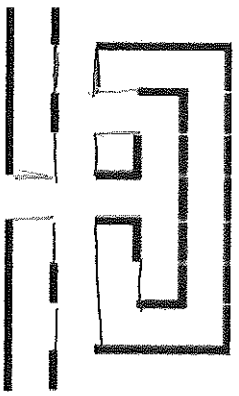
$-N^{-1} A^{-4}$ Id



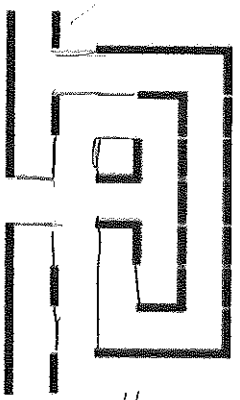
$-P$



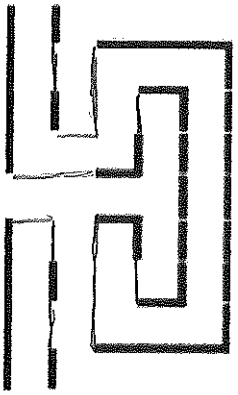
$-N^{-1} A^{-2}$ P



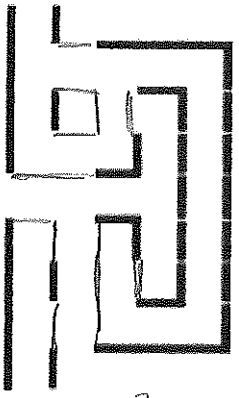
$-N A^{-2}$ P



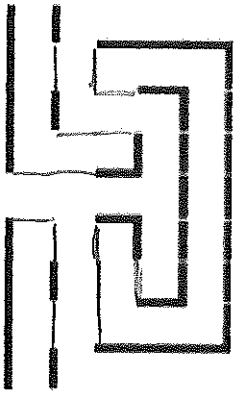
$-A^{-4}$ P



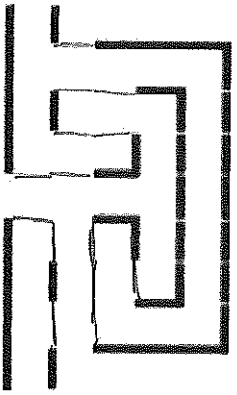
N^{-2} P



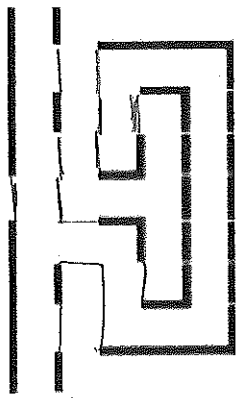
$N^{-1} A^{-2}$ P



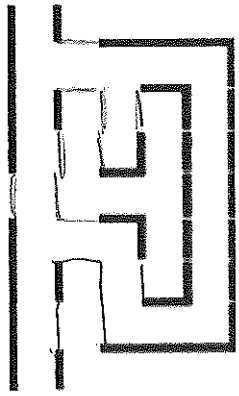
$N^{-1} A^{-2}$ P



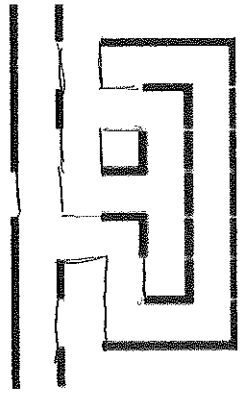
$N^{-2} A^{-4}$ P



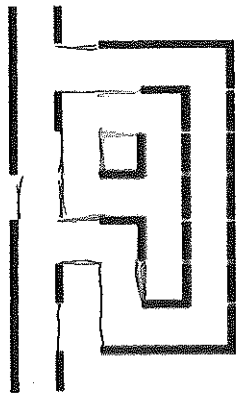
$-N^{-1}\,Id$



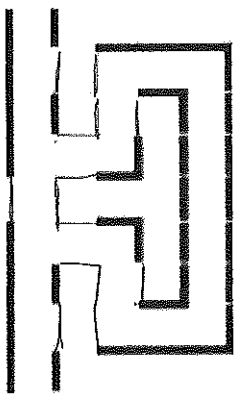
$-A^{-2}\,Id$



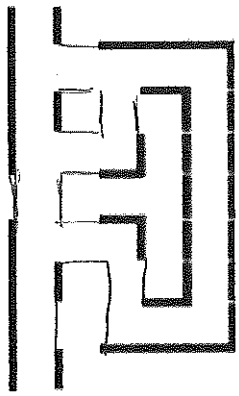
$-A^{-2}\,Id$



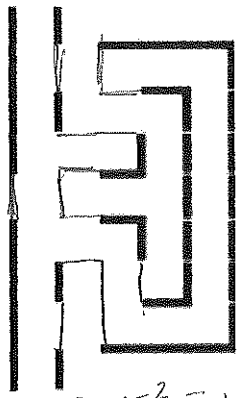
$-NA^{-4}\,Id$



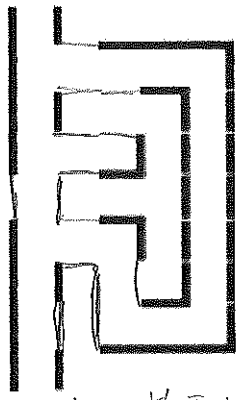
$N^{-1}\,Id$



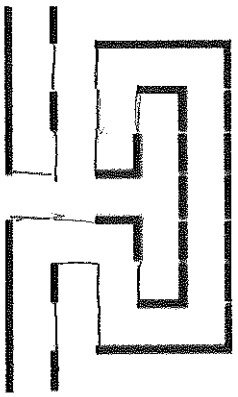
$A^{-2}\,Id$



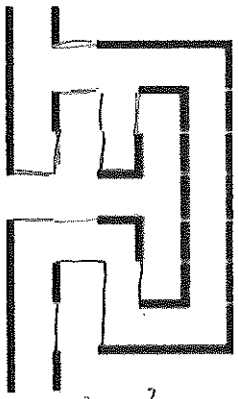
$N^{-2}A^{-2}\,Id$



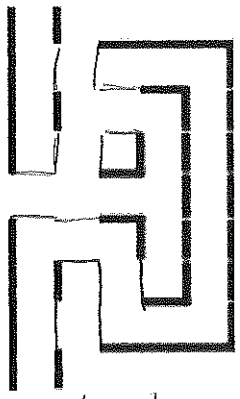
$N^{-1}A^{-4}\,Id$



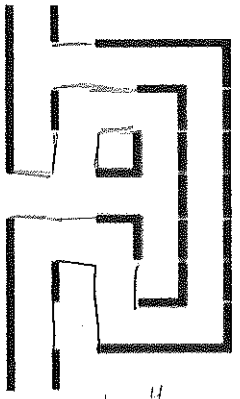
$N^{-2}\,P$



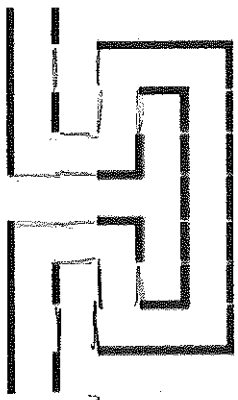
$N^{-2}A^{-2}\,Id$



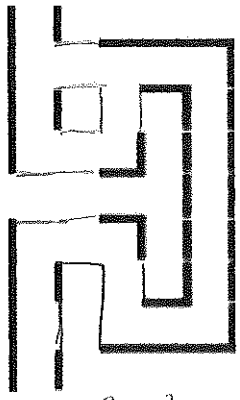
$N^{-1}A^{-2}P$



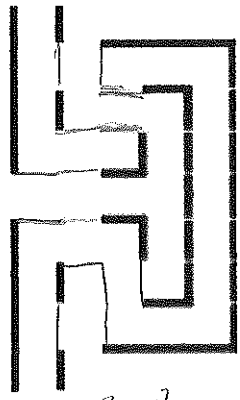
$N^{-1}A^{-4}\,Id$



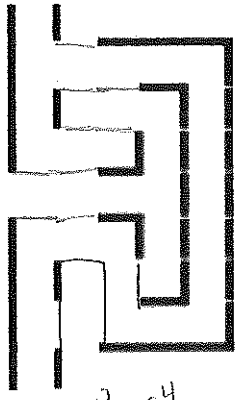
$-N^{-3}\,Id$



$-N^{-2}A^{-2}\,Id$

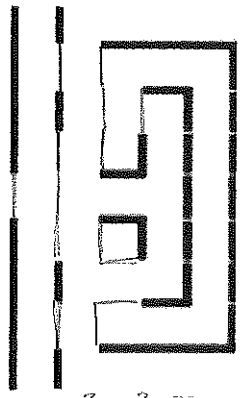


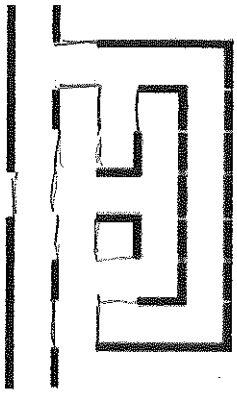
$-N^{-3}A^{-2}P$

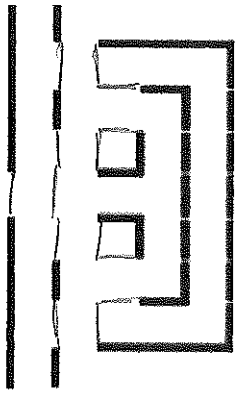


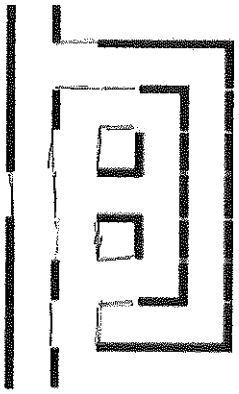
$-N^{-3}A^{-4}\,Id$

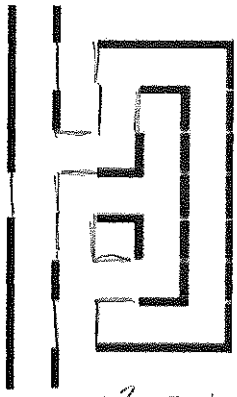
3

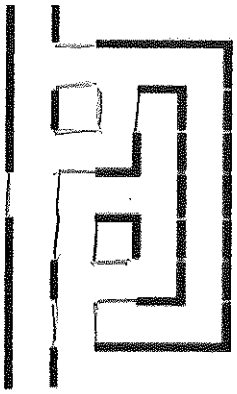

$N^2A^2\,Id$

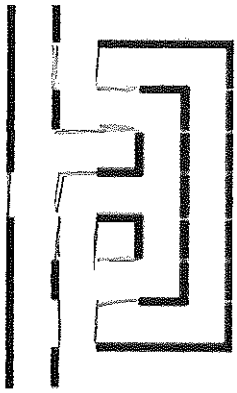

$N\,Id$

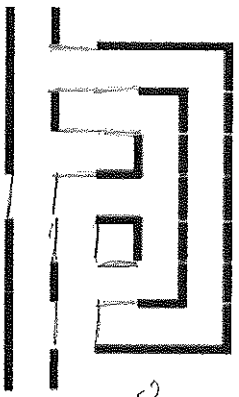

$N^3\,Id$

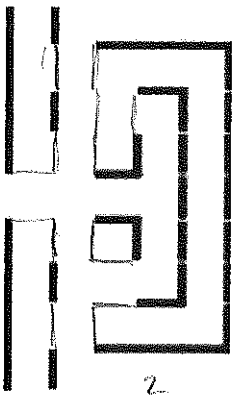

$N^2A^{-2}\,Id$

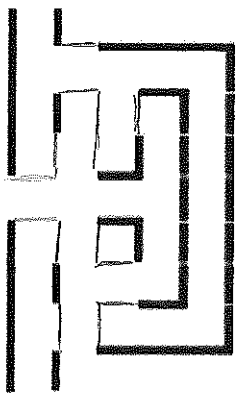

$A^2\,Id$

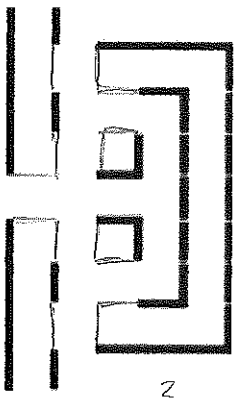

$-N\,Id$

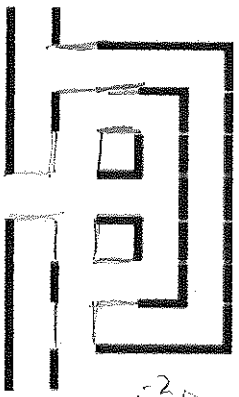

$-N\,Id$

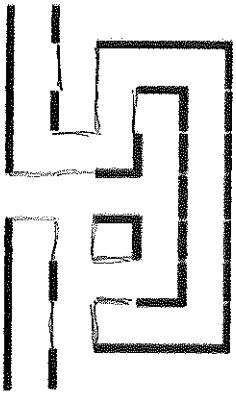

$-A^{-2}\,Id$

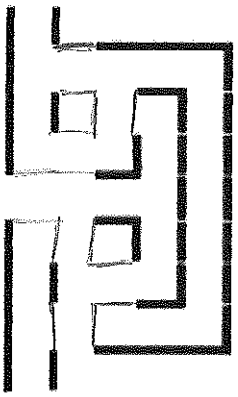

$-NA^2P$

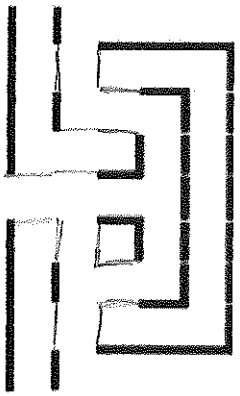

$-P$

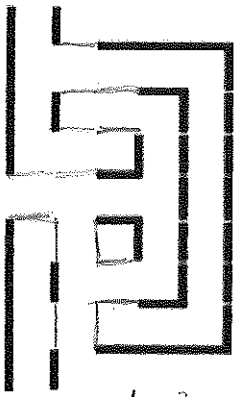

$-N^2P$

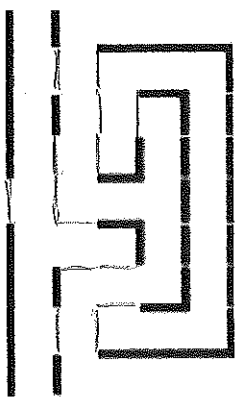

$-NA^{-2}P$

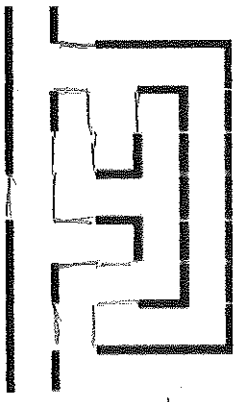

$N^{-1}A^2P$

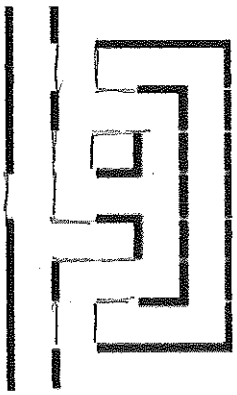

P

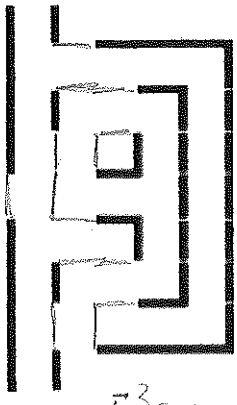

P

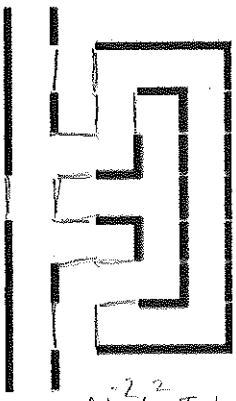

$N^{-1}A^{-2}P$

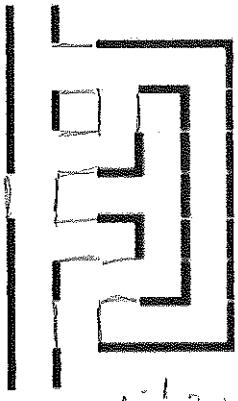

$-A^2 I_d$

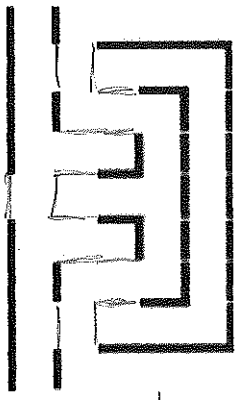

$-N^{-1} I_d$

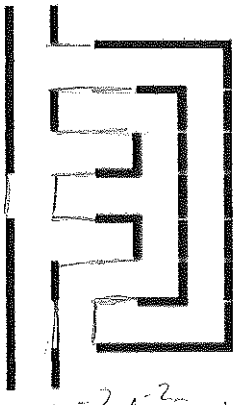

$-N I_d$

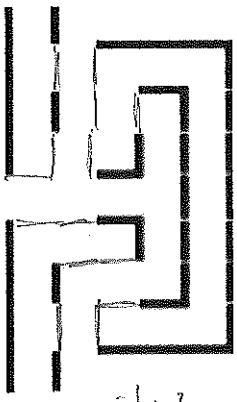

$-A^2 I_d$

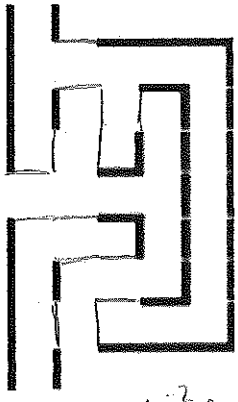

$N^{-2} A^2 I_d$

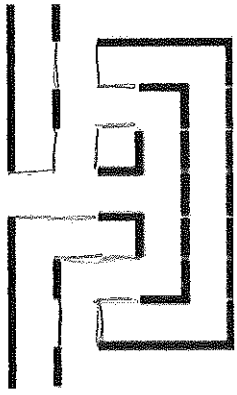

$N^{-1} I_d$

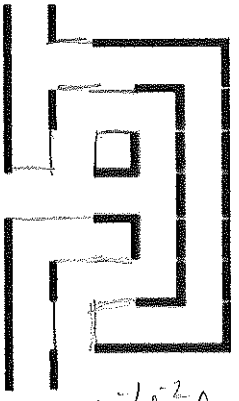

$N^{-1} I_d$

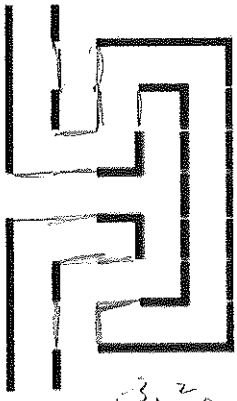

$N^{-2} A^{-2} I_d$

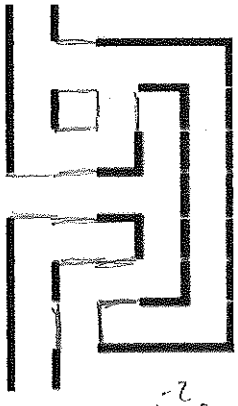

$N^{-1} A^2 P$

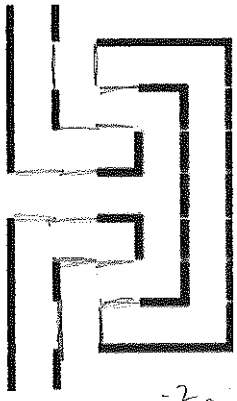

$N^{-2} P$

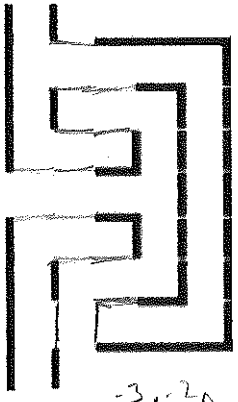

P


$N^{-1} A^{-2} P$

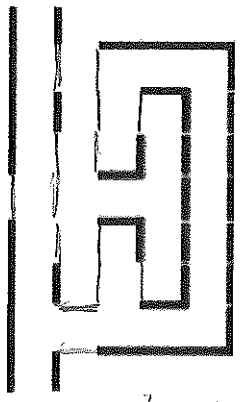

$-N^{-3} A^2 P$

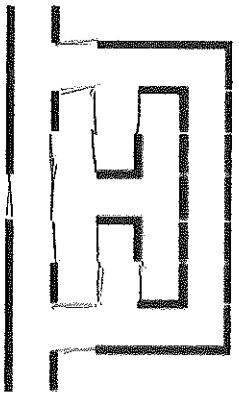

$-N^{-2} P$

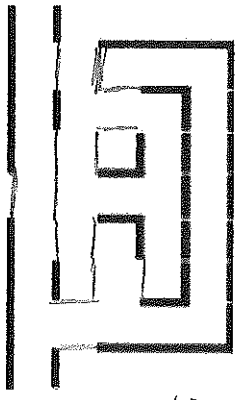

$-N^{-2} P$

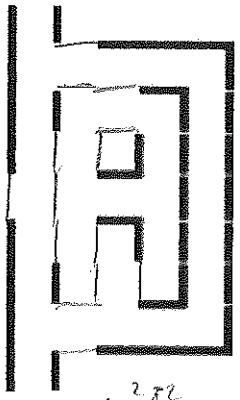

$-N^{-3} A^{-2} P$

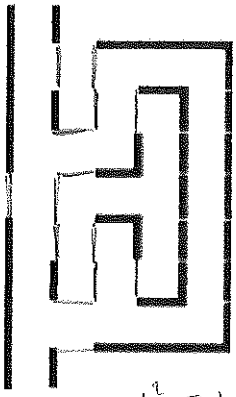
5

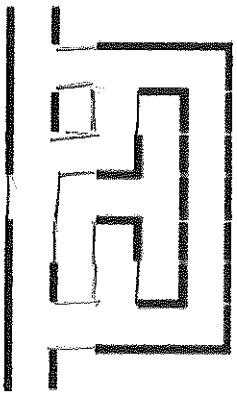

$A^2 Id$

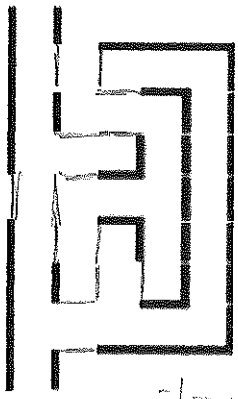

$N Id$

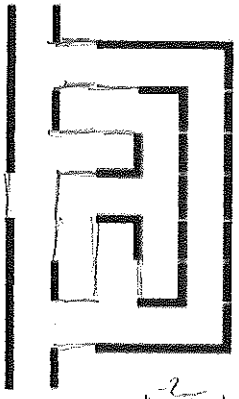

$N Id$

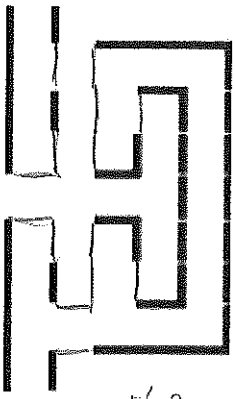

$N^2 A^2 Id$

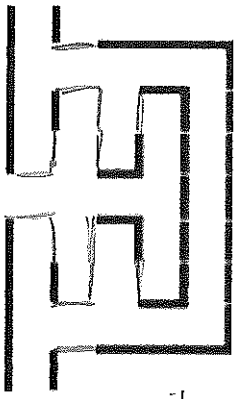

$-A^2 Id$

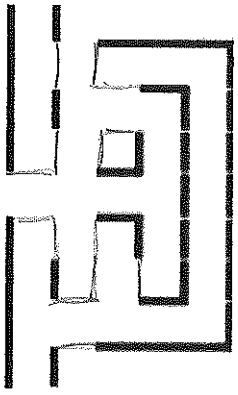

$-N Id$

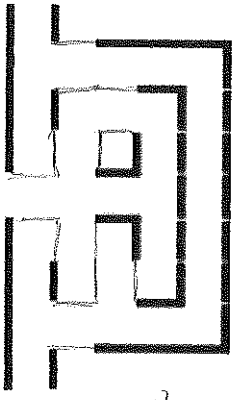

$-N^{-1} Id$

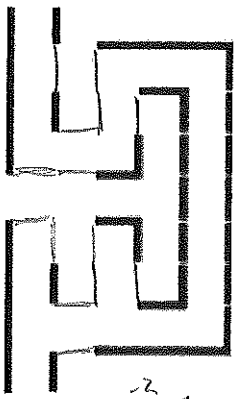

$-A^{-2} Id$

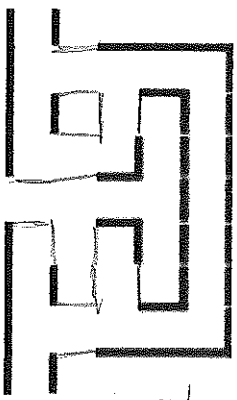

$-N^{-1} A^2 P$

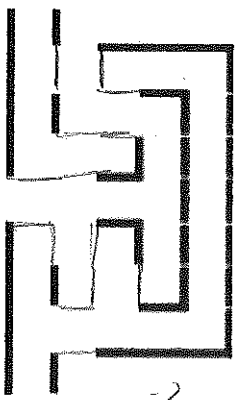

$-N^{-1} Id$

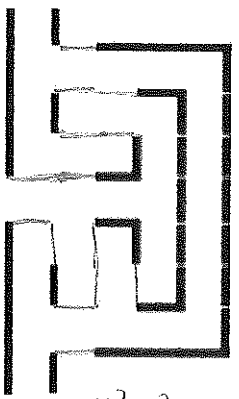

$-P$


$-A^{-2} Id$


$N^{-2} A^2 Id$

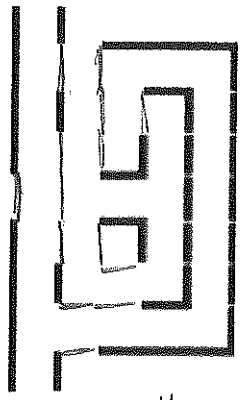

$N^{-1} Id$

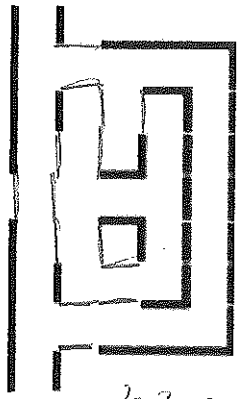

$N^{-2} P$

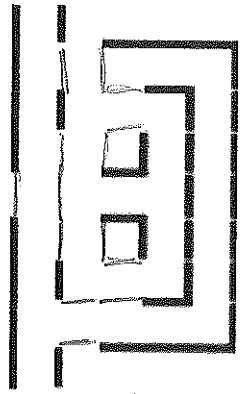

$N^{-2} A^{-2} Id$

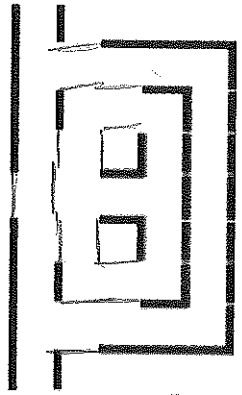
$-A^2 I_d$
$-N I_d$
$-N I_d$
$-N^2 A^{-2} I_d$
$A^2 I_d$
$N I_d$
$N^{-1} I_d$
$A^{-2} I_d$
$N^{-1} A^2 P$
$N^{-1} I_d$
P
$\bar{A}^2 I_d$
$-N^{-2} A^2 I_d$
$-N^{-1} I_d$
$-N^{-2} P$
$-N^{-2} \bar{A}^2 I_d$

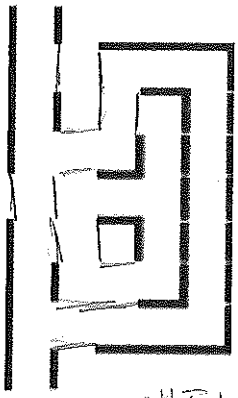
7.

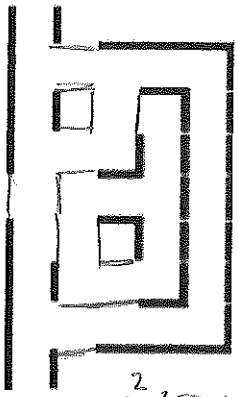

NA^4Id

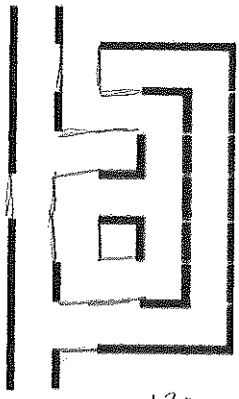

N^2A^2Id

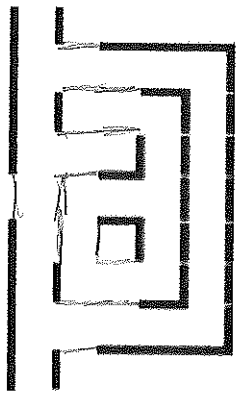

N^2A^2Id

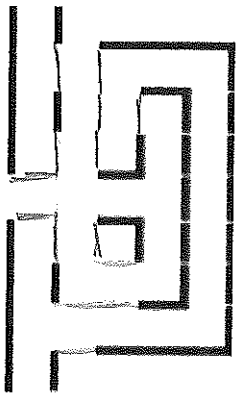

N^3Id

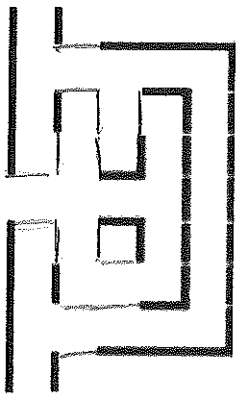

$-NA^4Id$

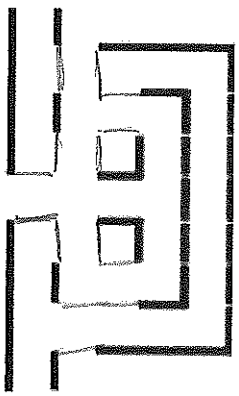

$-N^2A^2Id$

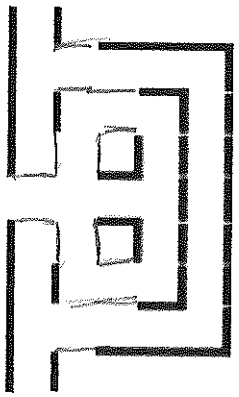

$-A^2Id$

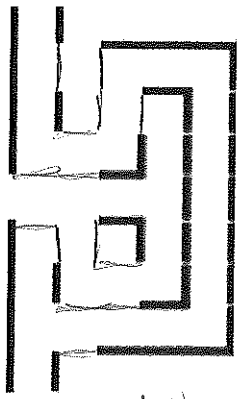

$-NId$

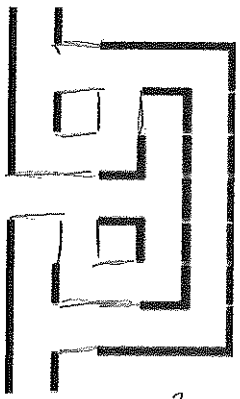

$-A^4P$

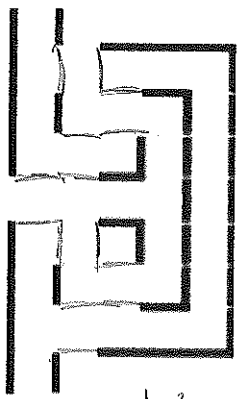

$-A^2Id$

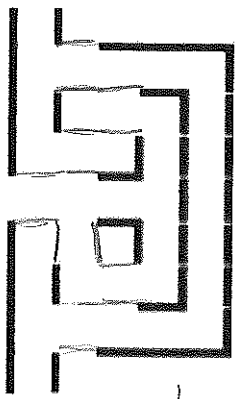

$-NA^2P$


$-NId$

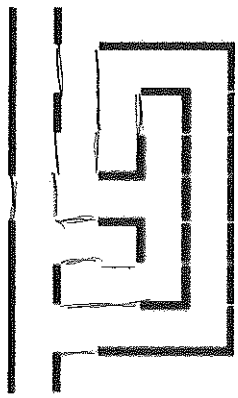

$N^{-1}A^4Id$


A^2Id

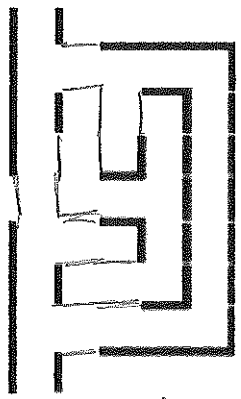

$N^{-1}A^2P$


$N^{-1}Id$

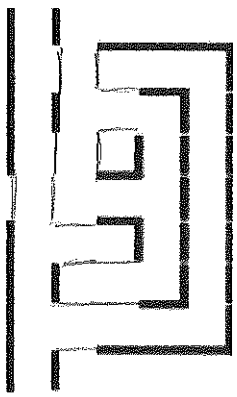
8.



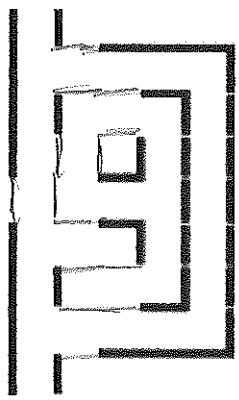
$-N^{-1}A^4Id$



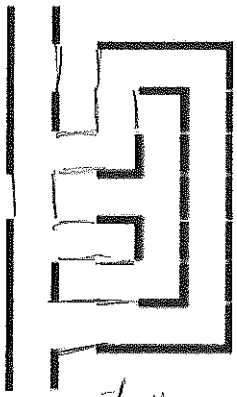
$-A^2Id$



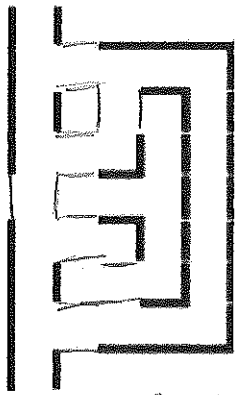
$-A^2Id$



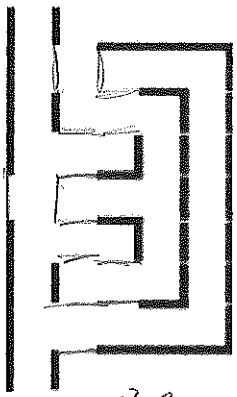
$=NId$



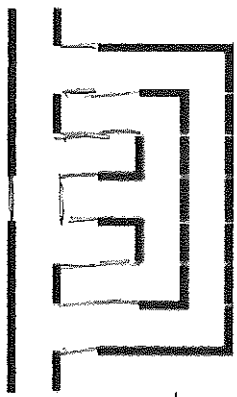
$N^{-1}A^4Id$



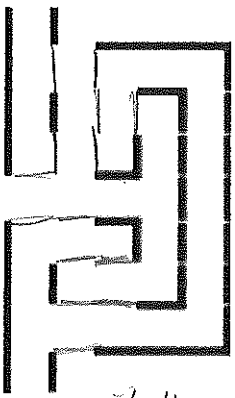
A^2Id



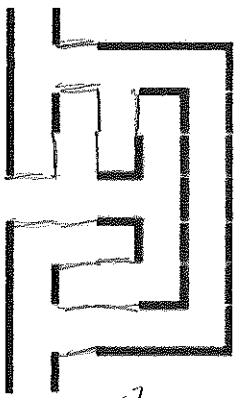
$N^{-2}A^2Id$



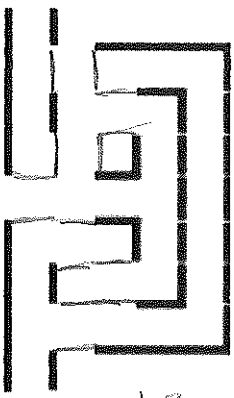
$N^{-1}Id$



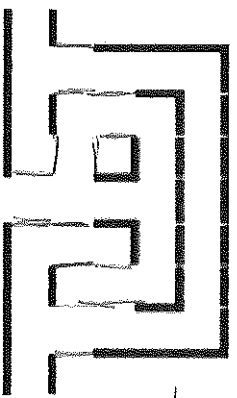
$N^{-2}A^4P$



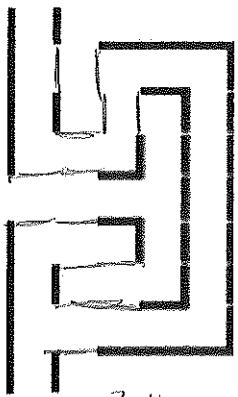
$N^{-2}A^2Id$



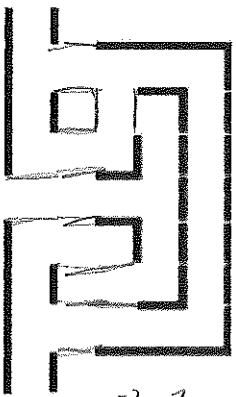
$N^{-1}A^2P$



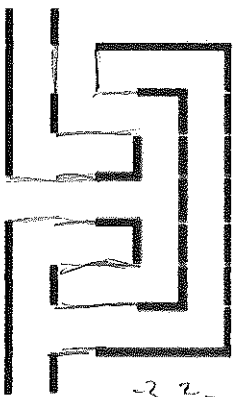
$N^{-1}Id$



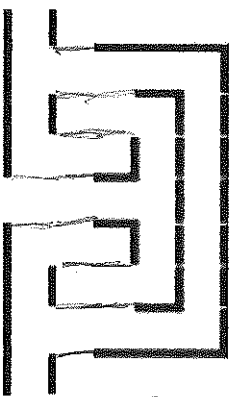
$-N^{-3}A^4Id$



$-N^{-2}A^2Id$



$-N^{-3}A^2P$



$-N^{-3}Id$

A.3 Python Code to Compute Minimum Output Entropy

The following code is used to efficiently compute minimum output entropies with respect to the von Neumann entropy for a range of possible parameters and output the results to a $.txt$ file.

```
'''
*Must have Python and the free external Python library 'mpmath' installed for this implementation to work.*

This improves on code previously implemented in MATLAB and also implements simplification of eigenvalue equation. The extent
   of this improvement is the following. Calculating for N = 3, M1 = M2 = L1 = L2 = K = 70:

   MATLAB                        : ~676 seconds
   Python w/out simplification   : ~31 seconds
   Python with simplification    : ~24 seconds
'''

# Libraries to import
from mpmath import *
from timeit import default_timer as timer

# Decimal digit precision
mp.dps = 1000
mp.pretty = True

# Precision for quantum integer calculations
PREC = 500

# List to store values of 'q' for simple table lookup
q_val = 0
q_ints = []

''' Formula 1. '''
def q( N ):

    global q_val
    q_val = fdiv( fsub( N , sqrt( fsub( power(N,2) , 4 ) ) ) , 2 )

''' Populating  '''
def populate_q( length ):

    for i in range(0,length):
        q_ints.append( quantum_integer(i) )

''' Formula 2. '''
def quantum_integer( a ):

    with workdps(PREC):
        if a == 0:
            return 1
```

```
        else:
            return fdiv( fsub( power(q_val,a) , power(q_val,-a) ) , fsub( q_val , power(q_val,-1) ) )

''' Formula 3. '''
def f_quantum_integer( a ):

    with workdps(PREC):
        if a == 0:
            return 1
        else:
            return fmul( power(-1,a-1) , q_ints[a] )

''' Formula 2. '''
def qInt_factorial( num ):

    if num == 0:
        return 1
    else:
        temp = 1
        for i in range(1, num+1):
            temp = fmul( temp , q_ints[i] )
        return temp

''' Formula 3. '''
def f_qInt_factorial( num ):

    if num == 0:
        return 1
    else:
        temp = 1
        for i in range(1, num+1):
            temp = fmul( temp , fmul( power(-1,i-1) , q_ints[i]) )
        return temp

''' Computing the sum in formula 2.18. '''
def tet_sum(m1,m2,l1,l2,l,k):

    a_s = [ (m1+l2+l)/2 , (l1+m2+l)/2 , (m1+l1+k)/2 , (m2+l2+k)/2 ]
    b_s = [ (l1+l2+l+k)/2 , (m1+m2+l+k)/2 , (m1+l1+m2+l2)/2 ]
    m = max(a_s)
    M = min(b_s)

    tot = 0
    for s in range(m, M+1):
        leftProduct = 1
        rightProduct = 1

        for a in a_s:
            leftProduct = fmul( leftProduct , f_qInt_factorial(s-a) )
        for b in b_s:
            rightProduct = fmul( rightProduct , f_qInt_factorial(b-s) )
        if s%2 == 0:
```

```
            numerator = fneg( f_qInt_factorial(s+1) )
        else:
            numerator = f_qInt_factorial(s+1)

        tot = fadd( tot , fdiv( numerator , fmul( leftProduct , rightProduct ) ) )

    return tot

''' Simplified eigenvalue formula in case where i = 0, k1 = k2. '''
def lmbda(m1,m2,l1,l2,l,k):

    a_s = [ (m1+l2+l)/2 , (l1+m2+l)/2 , (m1+l1+k)/2 , (m2+l2+k)/2 ]
    b_s = [ (l1+l2+l+k)/2 , (m1+m2+l+k)/2 , (m1+l1+m2+l2)/2 ]

    J = 1
    for i in range(0,4):
        for n in range(0,3):
            J = fmul( J , qInt_factorial(b_s[n] - a_s[i]) )

    return fdiv( fmul( fmul( q_ints[k+1] , J ) , power( tet_sum(m1, m2, l1, l2, l, k) , 2 )) , \
                 fmul( fmul( qInt_factorial((m1+k+l1+2)/2), qInt_factorial((m2+k+l2+2)/2) ) , \
                       fmul( qInt_factorial((m2+l+l1+2)/2) , qInt_factorial((l2+l+m1+2)/2) ) ) )


''' Computing value of expression at top of page 19. '''
if __name__ == '__main__':

    # Retrieving parameter values from user
    N = input("Please enter an appropriate value for N.\n")
    m1_low, m1_high = map(int, raw_input("Please enter a range of values for m1, simply with a space between (i.e. '4 6').\n").split() )
    m2_low, m2_high = map(int, raw_input("Please enter a range of values for m2, simply with a space between.\n").split() )
    l1_low, l1_high = map(int, raw_input("Please enter a range of values for l1, simply with a space between.\n").split() )
    l2_low, l2_high = map(int, raw_input("Please enter a range of values for l2, simply with a space between.\n").split() )
    k_low, k_high = map(int, raw_input("Please enter a range of values for k, simply with a space between.\n").split() )

    # Setting up table-lookups for improved efficiency in the case where we are testing a large number of parameters
    q(N)
    populate_q( max( [ (k_high+l2_high+m2_high+m1_high+2)/2 + 2 , (k_high+l1_high+m1_high+m2_high+2)/2 ] ) + 2 )

    # File to store data
    outputFile = open("output_vonNeumann.txt","w+")

    # Calculate differences for only appropriate admissible values in the specified range
    for m1 in range(m1_low , m1_high+1):
        for m2 in range(m2_low , m2_high+1):
            for l1 in range(l1_low , l1_high+1):
                for l2 in range(l2_low , l2_high+1):
                    for k in range(k_low , k_high+1):
                        if (l1+m1-k)%2 == 0 and (l1-m1+k)%2 == 0 and (-l1+m1+k)%2 == 0 and (k+l1+m1)%2 == 0 and -l2+m2+k >= 0 and\
                           (l2+m2-k)%2 == 0 and (l2-m2+k)%2 == 0 and (-l2+m2+k)%2 == 0 and (k+l2+m2)%2 == 0 and m1+l1 >= k and\
                           (l1-m2+m1+l2)%2 == 0 and l1-m2+abs(m1-l2) >= 0 and (m2-l1+m1+l2)%2 == 0 and m2-l1+abs(m1-l2) >= 0 and\
                           l1+m2 >= m1+l2 and l1-m1+k >= 0 and l1-m1+k >= 0 and -l1+m1+k >= 0 and l2+m2-k >= 0 and l2-m2+k >= 0:
```

```
                    # Calculating entropies for individual quantum channels (combined)
                    temp_numerator = [ (l1+m1-k)/2 , (l1-m1+k)/2 , (-l1+m1+k)/2 , (k+l1+m1+2)/2 , \
                                       (l2+m2-k)/2 ,(l2-m2+k)/2 , (-l2+m2+k)/2 , (k+l2+m2+2)/2 ]
                    temp_denominator = [ k+1 , k+1 , l1 , l2 , m1 , m2 ]

                    individual_channels = fmul( qInt_factorial(temp_numerator[0]) , qInt_factorial(temp_numerator[1]) )
                    for i in range(2,8):
                        individual_channels = fmul( individual_channels , qInt_factorial( temp_numerator[i] ) )
                        individual_channels = fdiv( individual_channels , qInt_factorial( temp_denominator[i-2] ) )

                    individual_channels = ln( individual_channels )

                    # Calculating entropy of tensor channel
                    total = 0

                    for r in range(0, min( [m1,l2] ) + 1):
                        l = m1 + l2 - 2*r
                        calc = lmbda(m1,m2,l1,l2,l,k)
                        total = fadd( total , fmul( q_ints[l+1] , fmul( calc , ln( calc ) ) ) )

                    total = fneg(total)

                    # Writing to output file at 'output_vonNeumann.txt'
                    outputFile.write(str(m1)+", "+str(m2)+", "+str(l1)+", "+str(l2)+", "+str(k)+", \
                                Entropy = "+str(nstr(total-individual_channels))+"\n")

    print("Done.\n")
```

The following code computes minimum output entropies for a range of parameters as done above, except the output entropy is computed with respect to the Renyi entropy.

```
'''
This improves on code previously implemented in MATLAB and also implements simplification of eigenvalue equation. The extent
   of this improvement is the following, calculating for N = 3, M1 = M2 = L1 = L2 = K = 70:

   MATLAB                        : ~676 seconds
   Python w/out simplification   : ~31 seconds
   Python with simplification    : ~24 seconds

'''

# Libraries to import
from mpmath import *
import time

# Decimal digit precision
mp.dps = 1000
mp.pretty = True

# Precision for quantum integer calculations
PREC = 500
```

```
# List to store values of 'q' for simple table lookup
q_val = 0
q_ints = []


''' Formula 1. '''
def q( N ):

    global q_val
    q_val = fdiv( fsub( N , sqrt( fsub( power(N,2) , 4 ) ) ) , 2 )

''' Populating  '''
def populate_q( length ):

    for i in range(0,length):
        q_ints.append( quantum_integer(i) )

''' Formula 2. '''
def quantum_integer( a ):

    with workdps(PREC):
        if a == 0:
            return 1
        else:
            return fdiv( fsub( power(q_val,a) , power(q_val,-a) ) , fsub( q_val , power(q_val,-1) ) )

''' Formula 3. '''
def f_quantum_integer( a ):

    with workdps(PREC):
        if a == 0:
            return 1
        else:
            return fmul( power(-1,a-1) , q_ints[a] )

''' Formula 2. '''
def qInt_factorial( num ):

    if num == 0:
        return 1
    else:
        temp = 1
        for i in range(1, num+1):
            temp = fmul( temp , q_ints[i] )
        return temp

''' Formula 3. '''
def f_qInt_factorial( num ):

    if num == 0:
        return 1
```

```
    else:
        temp = 1
        for i in range(1, num+1):
            temp = fmul( temp , fmul( power(-1,i-1) , q_ints[i]) )
        return temp

''' Computing the sum in formula 2.18. '''
def tet_sum(m1,m2,l1,l2,l,k):

    a_s = [ (m1+l2+l)/2 , (l1+m2+l)/2 , (m1+l1+k)/2 , (m2+l2+k)/2 ]
    b_s = [ (l1+l2+l+k)/2 , (m1+m2+l+k)/2 , (m1+l1+m2+l2)/2 ]
    m = max(a_s)
    M = min(b_s)

    tot = 0
    for s in range(m, M+1):
        leftProduct = 1
        rightProduct = 1

        for a in a_s:
            leftProduct = fmul( leftProduct , f_qInt_factorial(s-a) )
        for b in b_s:
            rightProduct = fmul( rightProduct , f_qInt_factorial(b-s) )
        if s%2 == 0:
            numerator = fneg( f_qInt_factorial(s+1) )
        else:
            numerator = f_qInt_factorial(s+1)

        tot = fadd( tot , fdiv( numerator , fmul( leftProduct , rightProduct ) ) )

    return tot

''' Simplified eigenvalue formula in case where i = 0, k1 = k2. '''
def lmbda(m1,m2,l1,l2,l,k):

    a_s = [ (m1+l2+l)/2 , (l1+m2+l)/2 , (m1+l1+k)/2 , (m2+l2+k)/2 ]
    b_s = [ (l1+l2+l+k)/2 , (m1+m2+l+k)/2 , (m1+l1+m2+l2)/2 ]

    J = 1
    for i in range(0,4):
        for n in range(0,3):
            J = fmul( J , qInt_factorial(b_s[n] - a_s[i]) )

    return fdiv( fmul( fmul( q_ints[k+1] , J ) , power( tet_sum(m1, m2, l1, l2, l, k) , 2 )) , \
                 fmul( fmul( qInt_factorial((m1+k+l1+2)/2), qInt_factorial((m2+k+l2+2)/2) ) , \
                       fmul( qInt_factorial((m2+l+l1+2)/2) , qInt_factorial((l2+l+m1+2)/2) ) ) )


'''
Computing expression at the top of page 19 with formula 11 substituted
       in for the output entropy of the tensor quantum channel.
'''
```

```
if __name__ == '__main__':

    # Retrieving parameter values from user
    N = input("Please enter an appropriate value for N.\n")
    p = input("Please enter an appropriate value for p.\n")
    m1_low, m1_high = map(int, raw_input("Please enter a range of values for m1, simply with a space between (i.e. '4 6').\n").split() )
    m2_low, m2_high = map(int, raw_input("Please enter a range of values for m2, simply with a space between.\n").split() )
    l1_low, l1_high = map(int, raw_input("Please enter a range of values for l1, simply with a space between.\n").split() )
    l2_low, l2_high = map(int, raw_input("Please enter a range of values for l2, simply with a space between.\n").split() )
    k_low, k_high = map(int, raw_input("Please enter a range of values for k, simply with a space between.\n").split() )

    # Setting up table-lookups for improved efficiency in the case where we are testing a large number of parameters
    q(N)
    populate_q( max( [ (k_high+l2_high+m2_high+m1_high+2)/2 + 2 , (k_high+l1_high+m1_high+m2_high+2)/2 ] ) + 2 )

    # File to store data
    outputFile = open("output_Renyi.txt","w+")

    # Calculate differences for only appropriate admissible values in the specified range
    for m1 in range(m1_low , m1_high+1):
        for m2 in range(m2_low , m2_high+1):
            for l1 in range(l1_low , l1_high+1):
                for l2 in range(l2_low , l2_high+1):
                    for k in range(k_low , k_high+1):
                        if (l1+m1-k)%2 == 0 and (l1-m1+k)%2 == 0 and (-l1+m1+k)%2 == 0 and (k+l1+m1)%2 == 0 and\
                           (l2+m2-k)%2 == 0 and (l2-m2+k)%2 == 0 and (-l2+m2+k)%2 == 0 and (k+l2+m2)%2 == 0 and m1+l1 >= k and\
                           (l1-m2+m1+l2)%2 == 0 and l1-m2+abs(m1-l2) >= 0 and (m2-l1+m1+l2)%2 == 0 and m2-l1+abs(m1-l2) >= 0 and\
                           l1+m2 >= m1+l2 and l1-m1+k >= 0 and l1-m1+k >= 0 and -l1+m1+k >= 0 and l2+m2-k >= 0 and l2-m2+k >= 0 \
                         and -l2+m2+k >= 0:

                            # Calculating entropies for individual quantum channels (combined)
                            temp_numerator = [ (l1+m1-k)/2 , (l1-m1+k)/2 , (-l1+m1+k)/2 , (k+l1+m1+2)/2 , \
                                               (l2+m2-k)/2 ,(l2-m2+k)/2 , (-l2+m2+k)/2 , (k+l2+m2+2)/2 ]

                            temp_denominator = [ k+1 , k+1 , l1 , l2 , m1 , m2 ]

                            individual_channels = fmul( qInt_factorial(temp_numerator[0]) , qInt_factorial(temp_numerator[1]) )
                            for i in range(2,8):
                                individual_channels = fmul( individual_channels , qInt_factorial( temp_numerator[i] ) )
                                individual_channels = fdiv( individual_channels , qInt_factorial( temp_denominator[i-2] ) )

                            individual_channels = ln( individual_channels )

                            # Calculating entropy of tensor channel
                            total = 0

                            for r in range(0, min( [m1,l2] ) + 1):
                                l = m1 + l2 - 2*r

                                calc = lmbda(m1,m2,l1,l2,l,k)

                                total = fadd( total , fmul( q_ints[l+1] , power(calc,p) ) )
```

```
                    total = fdiv( ln(total) , 1-p )

                    outputFile.write(str(m1)+", "+str(m2)+", "+str(l1)+", "+str(l2)+", "+str(k)+",
                                Entropy = "+str(nstr(total-individual_channels))+"\n")

    print("Done.\n")
```

The following code computes minimum output entropies for a range of parameters as done above, except the output entropy is computed with respect to the more general formula for eigenvalues found in formula 7. It is important to note that unlike the two programs above, the following code does not check for validity of input parameters. The user must be more vigilant in ensuring that entered parameters are admissible.

```
# Libraries to import
from mpmath import *
import time

# Decimal digit precision
mp.dps = 1000
mp.pretty = True

# Precision for quantum integer calculations
PREC = 500

# List to store values of 'q' for simple table lookup
q_val = 0
q_ints = []

''' Formula 1. '''
def q( N ):

    global q_val
    q_val = fdiv( fsub( N , sqrt( fsub( power(N,2) , 4 ) ) ) , 2 )

''' Populating  '''
def populate_q( length ):

    for i in range(0,length):
        q_ints.append( quantum_integer(i) )

''' Formula 2. '''
def quantum_integer( a ):

    with workdps(PREC):
        if a == 0:
            return 1
```

```
        else:
            return fdiv( fsub( power(q_val,a) , power(q_val,-a) ) , fsub( q_val , power(q_val,-1) ) )

''' Formula 3. '''
def f_quantum_integer( a ):

    with workdps(PREC):
        if a == 0:
            return 1
        else:
            return fmul( power(-1,a-1) , q_ints[a] )

''' Formula 2. '''
def qInt_factorial( num ):

    if num == 0:
        return 1
    else:
        temp = 1
        for i in range(1, num+1):
            temp = fmul( temp , q_ints[i] )
        return temp

''' Formula 3. '''
def f_qInt_factorial( num ):

    if num == 0:
        return 1
    else:
        temp = 1
        for i in range(1, num+1):
            temp = fmul( temp , fmul( power(-1,i-1) , q_ints[i]) )
        return temp

''' Formula 4. '''
def theta( a , b , c ):

    r = (b+c-a)/2
    return fdiv( fmul( fmul( qInt_factorial(r) , qInt_factorial(b-r) ) , fmul( qInt_factorial(c-r) , qInt_factorial(a+r+1) ) ) ,\
                fmul(fmul( qInt_factorial(a) , qInt_factorial(b) ) , qInt_factorial(c) ) )

''' Formula 5. '''
def tet(m1,m2,l1,l2,l,k):

    a_s = [ (m1+l2+1)/2 , (l1+m2+1)/2 , (m1+l1+k)/2 , (m2+l2+k)/2 ]
    b_s = [ (l1+l2+1+k)/2 , (m1+m2+1+k)/2 , (m1+l1+m2+l2)/2 ]
    m = max(a_s)
    M = min(b_s)

    J = 1
    for i in range(0,4):
        for n in range(0,3):
```

```
            J = fmul( J , f_qInt_factorial(b_s[n] - a_s[i]) )

    E = fmul( fmul( f_qInt_factorial(m1) , fmul( f_qInt_factorial(m2) , f_qInt_factorial(l1) )), \
              fmul( f_qInt_factorial(l2) , fmul( f_qInt_factorial(l) , f_qInt_factorial(k) )))

    tot = 0
    for s in range(m, M+1):
        leftProduct = 1
        rightProduct = 1

        for a in a_s:
            leftProduct = fmul( leftProduct , f_qInt_factorial(s-a) )
        for b in b_s:
            rightProduct = fmul( rightProduct , f_qInt_factorial(b-s) )
        if s%2 == 0:
            numerator = fneg( f_qInt_factorial(s+1) )
        else:
            numerator = f_qInt_factorial(s+1)

        tot = fadd( tot , fmul(fdiv( numerator , fmul( leftProduct , rightProduct ) ) , fdiv( J , E ) ) )

    return tot

''' Formula 6. '''
def q6j(m1,m2,l1,l2,l,k):

    return fdiv(fmul( tet(m1, m2, l1, l2, l, k), quantum_integer(l+1)) , fmul(theta(m1, l2, l),theta(l1, m2, l)))

''' Formula 7 for eigenvalue formula. '''
def lmbda(m1,m2,l1,l2,l,k1,k2,i):

    temp = fdiv(fmul(fmul(quantum_integer(k1+1), quantum_integer(k2+1)), theta(l, m1, l2)) , \
                fmul(fmul(quantum_integer(l+1), theta(k1, l1, m1)), fmul(theta(k2, l2, m2), theta(i, k1, k2))))
    tot = 0
    for t in range(0,min([k1,k2])):
        j = 2*t
        tot = fadd(tot , fdiv(fmul(fmul(q6j(k1, k2, k2, k1, j, i), tet(k1, m1, m1, k1, j, l1)), \
                                   fmul(tet(k2, l2, l2, k2, j, m2), q6j(m1, l2, l2, m1, l, j))) , fmul(theta(m1, m1, j),theta(l2, j, l2))))

    return fmul(temp, tot)

''' Computing value of expression at top of page 19. '''
if __name__ == '__main__':

    # Retrieving parameter values from user
    N = input("Please enter an appropriate value for N.\n")
    i = input("Please enter an appropriate value for i.\n")
    m1_low, m1_high = map(int, raw_input("Please enter a range of values for m1, simply with a space between (i.e. '4 6').\n").split() )
    m2_low, m2_high = map(int, raw_input("Please enter a range of values for m2, simply with a space between.\n").split() )
    l1_low, l1_high = map(int, raw_input("Please enter a range of values for l1, simply with a space between.\n").split() )
    l2_low, l2_high = map(int, raw_input("Please enter a range of values for l2, simply with a space between.\n").split() )
    k1_low, k1_high = map(int, raw_input("Please enter a range of values for k1, simply with a space between.\n").split() )
```

```
k2_low, k2_high = map(int, raw_input("Please enter a range of values for k2, simply with a space between.\n").split() )

# File to store data
outputFile = open("output_generalEigenvalue.txt","w+")

q(N)
populate_q( max( [ (k1_high+l2_high+m2_high+m1_high+2)/2 + 2 , (k1_high+l1_high+m1_high+m2_high+2)/2 ] ) + 2 )

# Calculate differences for only appropriate admissible values in the specified range
for m1 in range(m1_low , m1_high+1):
    for m2 in range(m2_low , m2_high+1):
        for l1 in range(l1_low , l1_high+1):
            for l2 in range(l2_low , l2_high+1):
                for k1 in range(k1_low , k1_high+1):
                    for k2 in range(k2_low , k2_high+1):

                        # Calculating entropies for individual quantum channels (combined k1 and k2)
                        temp_numerator = [ (l1+m1-k1)/2 , (l1-m1+k1)/2 , (-l1+m1+k1)/2 , (k1+l1+m1+2)/2 , \
                                           (l2+m2-k2)/2 ,(l2-m2+k2)/2 , (-l2+m2+k2)/2 , (k2+l2+m2+2)/2 ]
                        temp_denominator = [ k1+1 , k2+1 , l1 , l2 , m1 , m2 ]

                        individual_channels = fmul( qInt_factorial(temp_numerator[0]) , qInt_factorial(temp_numerator[1]) )
                        for te in range(2,8):
                            individual_channels = fmul( individual_channels , qInt_factorial( temp_numerator[te] ) )
                            individual_channels = fdiv( individual_channels , qInt_factorial( temp_denominator[te-2] ) )

                        individual_channels = ln( individual_channels )

                        # Calculating entropy of tensor channel
                        total = 0

                        for r in range(0, min( [m1,l2] ) + 1):
                            l = m1 + l2 - 2*r

                            calc = lmbda(m1,m2,l1,l2,l,k1,k2,i)

                            total = fadd( total , fmul( q_ints[l+1] , fmul( calc , ln( calc ) ) ) )

                        total = fneg(total)

                        outputFile.write(str(m1)+", "+str(m2)+", "+str(l1)+", "+str(l2)+", "+str(k)+",
                                         Entropy = "+str(nstr(total-individual_channels))+"\n")

print('Done.\n')
```