Optimizing over k-extendable states

Tom Guo

January 22, 2012

Abstract

We introduce a quantum inspired approximation algorithm for computing the injective tensor norm, an NP-hard problem. The injective tensor norm has been shown to be equivalent to a convex optimization over Sep, and our algorithm approximates it by maximizing over k-extendable states, a chain of supersets converging to Sep. This approximation algorithm could be superior to known classical ones in precision and/or time complexity, and such a comparison deserves to be further investigated both theoretically and empirically.

Contents

[1]This is a working copy of work in progress.

1 Introduction

While a vector has one index and a matrix has two, a *tensor* has 3 or more. For one index, the injective tensor norm is the length of a vector and for two indices, it is the largest singular value of a matrix, but for 3 or more indices, computing the injective tensor norm is NP-hard.

We will introduce in this paper an approximation algorithm for the injective tensor norm for 3 indices. Our algorithm is based on the equivalence of the injective tensor norm to a maximum over separable quantum states.

1.1 The injective tensor norm

From now on, we use B_n to denote $\{v \in \mathbb{C}^n : ||v|| = 1\}$, the set of unit complex vectors.

Definition 1.1. *The injective tensor norm of a tensor A with three indices $i, j, k \in [n]$ is*

$$||A||_{\text{inj}} = \max_{x,y,z \in B_n} \left| \sum_{i,j,k} A_{ijk} x_i y_j z_k \right|,$$

Note how without the third index, this is simply the largest singular value.

Identity 1.2.

$$\max_{x,y,z \in B_n} \left| \sum_{i,j,k} A_{ijk} x_i y_j z_k \right| = \max_{x,y \in B_n} \left|\left| \sum_{i,j,k} A_{ijk} x_i y_j e_k \right|\right|,$$

where $e_k = |k\rangle$.

Proof.

$$\begin{aligned}
\max_{x,y,z \in B_n} \left| \sum_{i,j,k} A_{ijk} x_i y_j z_k \right| &= \max_{x,y \in B_n} \max_{z \in B_n} \left| \sum_{i,j,k} A_{ijk} x_i y_j z_k \right| \\
&= \max_{x,y \in B_n} \max_{z \in B_n} \left| \sum_{i,j,k} A_{ijk} x_i y_j e_k \cdot \sum_k z_k e_k \right| \\
&= \max_{x,y \in B_n} \left|\left| \sum_{i,j,k} A_{ijk} x_i y_j e_k \right|\right| \left|\left| \sum_k z_k e_k \right|\right|, \text{ by Cauchy} - \text{Schwarz inequality} \\
&= \max_{x,y \in B_n} \left|\left| \sum_{i,j,k} A_{ijk} x_i y_j e_k \right|\right| \cdot 1 \\
&= \max_{x,y \in B_n} \left|\left| \sum_{i,j,k} A_{ijk} x_i y_j e_k \right|\right|.
\end{aligned}$$

□

1.2 Quantum formulation of the injective tensor norm

Lemma 1.3. *We can compute an n^2 by n^2 positive definite, hermitian matrix M satisfying*

$$||A||_{\text{inj}}^2 = \max_{x,y \in B_n} \text{Tr}\left(M(xx^* \otimes yy^*)\right).$$

Proof.

$$\begin{aligned} ||A||_{\text{inj}}^2 &= \max_{x,y\in B_n} \left(\sum_{i_1,j_1,k} A_{i_1j_1k}x_{i_1}y_{j_1}e_k\right)^* \left(\sum_{i_2,j_2,k} A_{i_2j_2k}x_{i_2}y_{j_2}e_k\right) \\ &= \max_{x,y\in B_n} \sum_{i_1,j_1,i_2,j_2} \sum_k \overline{A_{i_1j_1k}} A_{i_2j_2k} \overline{x_{i_1}} x_{i_2} \overline{y_{j_1}} y_{j_2}. \end{aligned}$$

Let M be an n^2 by n^2 matrix indexed by $(i_1, j_1), (i_2, j_2)$, with $M_{(i_1,j_1),(i_2,j_2)} = \sum_k \overline{A_{i_1j_1k}} A_{i_2j_2k}$. It is not hard to verify that M is positive definite and hermitian. It follows that

$$\begin{aligned} \max_{x,y\in B_n} \sum_{i_1,j_1,i_2,j_2} \sum_k \overline{A_{i_1j_1k}} A_{i_2j_2k} \overline{x_{i_1}} x_{i_2} \overline{y_{j_1}} y_{j_2} &= \max_{x,y\in B_n} \sum_{i_1,j_1,i_2,j_2} M \overline{x_{i_1}} x_{i_2} \overline{y_{j_1}} y_{j_2} \\ &= \max_{x,y\in B_n} \operatorname{Tr}\left(M(xx^* \otimes yy^*)\right). \end{aligned}$$

The last step can be verified by computing the trace as the sum of the main diagonal of $M(xx^* \otimes yy^*)$. □

Theorem 1.4.

$$||A||_{\text{inj}}^2 = \max_{\rho\in \text{Sep}(n,n)} \operatorname{Tr}(M\rho),$$

where Sep(n, n) is the set of dimension n^2 states which can be written as the tensor product of two dimension n states.

Proof.

$$\begin{aligned} ||A||_{\text{inj}}^2 &= \max_{x,y\in B_n} \operatorname{Tr}\left(M(xx^* \otimes yy^*)\right) && \textit{by Lemma 1.3} \\ &= \max_\rho \operatorname{Tr}(M\rho), \rho \in \text{conv}\{xx^* \otimes yy^* : x, y \in B_n\} \end{aligned}$$

where conv(S) represents the convex hull of set S. The two expressions are equal because the maximum of a linear function over a convex set must occur at an extreme point. It is a well known fact in quantum computing that $\text{conv}\{xx^* \otimes yy^* : x, y \in B_n\} = \text{Sep}(n, n)$, and hence we are done. □

We believe approximating the injective tensor norm using its quantum formulation may be more promising than using a traditional or classical approach where little progress has been made. In this paper, we introduce an approximation algorithm based on the quantum interpretation described above. Though we believe this algorithm may be faster than ones previously developed, such a complexity bound remains to be proven. Once this algorithm is efficiently implemented, we can use our implementation to collect numerical data that could provide clues on the time complexity of the approximation algorithm relative to the precision. This may enable us to establish a numerically supported conjecture which we can then attempt to prove with a more reasonable chance of success.

Before describing our approximation algorithm, we will first introduce necessary background math and definitions.

2 Symmetric subspaces and exchangeable operators

2.1 Permutation operatiors

For each permutation $\pi \in S_k$, we define a unitary operator W_π on $\mathbb{C}^{d^{\otimes n}}$, where

$$W_\pi(u_1 \otimes \cdots \otimes u_k) = u_{\pi^{-1}(1)} \otimes \cdots \otimes u_{\pi^{-1}(k)}$$

with $u_1, \cdots, u_k \in \mathbb{C}^d$. In other words, W_π permutes the contents of the registers according to π.

2.2 The symmetric subspace

The symmetric subspace is set of vectors invariant under action of W_π for all $\pi \in S_n$, which clearly form a subspace. More precisely, the symmetric subspace

$$\mathrm{Sym}(k) = \{u \in \mathbb{C}^{d^{\otimes k}} : u = W_\pi \text{ for all } \pi \in S_n\}$$

2.3 Urn-marble representations

To construct an orthonormal basis $\mathrm{Sym}(k)$, we must first introduce urn-marble representations. Consider the set $\mathrm{Urn}(k,d)$ of functions of the form $\phi : [d] \mapsto \mathbb{N}$ such that $\sum_{a\in[d]} \phi(a) = k$. Informally, this set represents all possible *urns* containing k marbles, where each marble is labelled by an element of $[d]$. Simple combinatorics shows that $|\mathrm{Urn}(k,d)| = \binom{k+d-1}{d-1}$.

In the approximation algorithm, we will work with three different representations of $\mathrm{Urn}(k,d)$: index, string, and type. Each has its own advantages with regard to the algorithm, and it will be useful to make conversions between them.

2.3.1 Index representation

This representation, as its name suggests, simply indexes the elements of set with $[\binom{k+d-1}{d-1}]$. We will use this to index the elements in the approximation algorithm.

2.3.2 String representation

We use strings of the form $x_1x_2\cdots x_k$, the x_is $\in [d]$, in non-decreasing order. x_i is the label of the ith marble. Let $\mathrm{Str}(k,d)$ denote the index-to-array corresponding to $\mathrm{Urn}(k,d)$. We generate it by partitioning $\mathrm{Str}(k,d)$ based on the first element of each string, which ranges from 1 to d. If the first element is $i \in [d]$, we take $\mathrm{Str}(k-1, d-i+1)$ and increment every label by $i-1$ (since our strings need to be non-decreasing) and append every one of those strings to i. This generates the set of all strings in $\mathrm{Str}(k,d)$ where the first element is i. We start with a trivial base case and use dynamic programming to build up numerically sorted arrays and eventually we are able to construct a numerically sorted array for k, d.

To go from the a string to its index, we can precompute a hash table with the strings as keys to the index.

Now suppose the string need not be non-decreasing. We say that this string is *consistent* with a particular function $\phi \in \mathrm{Urn}(k,d)$ if it describes the labels of one possible ordering of the marbles in the urn corresponding to ϕ. To find the urn that the string corresponds to, simply sort the string to get the index using the precomputed hash table. More formally, we can define a function

$$f_{a_1\cdots a_k}(b) = |\{j \in [k] : b = a_j\}|,$$

and say that the string $a_1\cdots a_n$ is consistent with ϕ if and only if $f_{a_1\cdots a_k} = \phi$. We denote the number of distinct strings consistent with ϕ with

$$\binom{n}{\phi} \equiv \frac{n!}{\prod_{a\in[d]}(\phi(a)!)}.$$

2.3.3 Type representation

Here, we use $0 \le t_1, t_2, \cdots, t_d \le k$, $t_1 + \cdots + t_d = k$. t_i represents the number of marbles with label i. A indexed array of this representation can be constructed using dynamic programming, similar to what was done for the string representation. Unlike strings, types represent only urns, types cannot represent different permutations of an urn. On the other hand, the type representation is much more compact, and we do not have to deal with sorting. For small d, the type representation clearly has its advantages, since only d

numbers are needed. Though it would be more efficient to pre-compute a hash table to map a type to to its index, there is interestingly an $O(k)$ technique to do so, namely

$$f(t_1, \cdots, t_d) = \sum_{i=1}^{d} \sum_{j=0}^{t_i - 1} \binom{k - (t_1 + t_2 + \cdots + t_{i-1}) - j + d - i}{d - i}.$$

2.4 Orthonormal basis of the symmetric subspace

We will use the string representation to construct an orthonormal basis for $\mathrm{Sym}(k)$. Formally, it can be expressed as $\{v_\pi : \phi \in \mathrm{Urn}(k, d)\}$, where

$$v_\phi = \binom{n}{\phi}^{-1/2} \sum_{\substack{a_1 \cdots a_k \in [k] \\ f_{a_1 \cdots a_n} = \phi}} e_{a_1} \otimes \cdots \otimes e_{a_k}.$$

Loosely speaking, v_π is the uniform superposition over all of the strings that are consistent with ϕ, every permutation is a bijection with respect to each class of consistent strings. Hence, these basis vectors are invariant under any permutation and thus they belong to $\mathrm{Sym}(k)$. $\binom{n}{\phi}^{-1/2}$ is used as a normalization factor, and clearly, the vectors are mutually orthogonal. We illustrate this with an example, where $k = 3, d = 2$:

$$\begin{aligned} v_1 &= |000\rangle \\ v_2 &= \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle) \\ v_3 &= \frac{1}{\sqrt{3}}(|011\rangle + |101\rangle + |110\rangle) \\ v_4 &= |111\rangle . \end{aligned}$$

Note how for small d and large k, the dimension of $\mathrm{Sym}(k)$ is very small, and for this reason, we will use $\mathrm{Sym}(k)$ in the approximation of $\mathrm{Tr}(M\rho)$. For that purpose, we will define the projector onto $\mathrm{Sym}(k)$:

$$\sum_{t=1}^{\binom{k+d-1}{d-1}} |v_t\rangle \langle v_t| .$$

It follows that the projector onto $\mathbb{C}^d \otimes \mathrm{Sym}(k)$ is

$$P = \sum_{t=1}^{\binom{k+d-1}{d-1}} I^A \otimes |v_t\rangle \langle v_t| .$$

2.5 Exchangeable operators

An operator in $\mathrm{Pos}(\mathbb{C}^{d^{\otimes k}})$ is exchangeable if and only if $P = W_\pi P W_\pi^*$ for all $\pi \in S_k$. Informally, an exchangeable operator in invariant under any permutation of the k registers. Every exchangeable operator can be expressed as a linear combination of outer products of vectors in $\mathrm{Sym}(k)$.

3 K-extendable states

A state ρ^{AB_1} is k-extendable if and only if there exists a $\sigma^{AB_1B_2\cdots B_k}$ exchangeable over $B_1, B_2, \cdots, B_k$ such that $\rho = \mathrm{Tr}_{B_2B_3\cdots B_k}(\sigma)$. For now, let Ext_k denote the set of k-extendable states. In particular, one can show that

$$\{\text{states}\} = \mathrm{Ext}_1 \supseteq \mathrm{Ext}_2 \supseteq \mathrm{Ext}_3 \supseteq \cdots \supseteq \mathrm{Ext}_k \supseteq \mathrm{Ext}_{k+1} \supseteq \cdots \supseteq \mathrm{Ext}_\infty = \mathrm{Sep}$$

This suggests that optimizing over Ext_k for increasing k may yield a good approximation for the maximization over Sep.

4 Algorithm for optimizing over k-extendable states

4.1 Writing the maximum as the largest eigenvalue, or 2-norm, of a positive semidefinite matrix

Lemma 4.1. *Any $\sigma^{AB_1B_2\cdots B_k}$ can be expressed as*

$$\sum c_{ij}\alpha_i \otimes \beta_j,$$

for some

$$c_{ij} \in \mathbb{C},\ \alpha_i \in \mathrm{L}(\mathbb{C}^{AB_1}),\ \beta_i \in \mathrm{L}(\mathbb{C}^{B_2\cdots B_k}),\ \text{and } \mathrm{Tr}(\beta_i) = 1.$$

Proof. By definition of the tensor product space, any $\sigma^{AB_1B_2\cdots B_k}$ can be expressed as a linear combination of tensor products of elements of complementary subspaces, in our case $\mathrm{L}(\mathbb{C}^{AB_1})$ and $\mathrm{L}(\mathbb{C}^{B_2\cdots B_k})$. That is, we can write $\sigma^{AB_1B_2\cdots B_k}$ as

$$\sum c'_{ij}\alpha_i \otimes \beta'_j,$$

for some

$$c'_{ij} \in \mathbb{C},\ \alpha_i \in \mathrm{L}(\mathbb{C}^{AB_1}),\ \text{and } \beta'_i \in \mathrm{L}(\mathbb{C}^{B_2\cdots B_k}).$$

We will require $\mathrm{Tr}(\beta'_j) \neq 0$ for all j so that we can later divide by it. Hence, for any j where $\mathrm{Tr}(\beta'_j) = 0$, we will rewrite β'_j as $(\beta'_j + X) - X$, $\mathrm{Tr}(X) \neq 0$, and distribute with respect to $c'_{ij}\alpha_i$. Then, in our new summation all β'_js will have non-zero trace, and it follows that

$$\begin{aligned}\sigma &= \sum c'_{ij}\alpha_i \otimes \beta'_j \\ &= \sum c'_{ij}\mathrm{Tr}(\beta'_j)\alpha_i \otimes \frac{\beta'_j}{\mathrm{Tr}(\beta'_j)}.\end{aligned}$$

Here, we let $c_{ij} = c'_{ij}\mathrm{Tr}(\beta'_j)$ and $\beta_j = \frac{\beta'_j}{\mathrm{Tr}(\beta'_j)}$, which satisfies $\mathrm{Tr}(\beta_j) = 1$, and hence completes the proof. □

Theorem 4.2. *For any $\rho^{AB_1} = \mathrm{Tr}_{B_2\cdots B_k}(\sigma^{AB_1B_2\cdots B_k}), M \in \mathrm{Pos}(\mathbb{C}^d \otimes \mathbb{C}^d)$,*

$$\mathrm{Tr}(M\rho) = \mathrm{Tr}\left((M \otimes I^{B_2\cdots B_k})\sigma\right).$$

Proof.

$$\begin{aligned}\mathrm{Tr}(M\rho) &= \mathrm{Tr}\left(M\mathrm{Tr}_{B_2\cdots B_k}(\sigma)\right) \\ &= \mathrm{Tr}\left(M(I^{AB_1} \otimes \mathrm{Tr})(\sum c_{ij}\alpha_i \otimes \beta_j)\right) \quad \textit{(by Lemma 4.1)} \\ &= \mathrm{Tr}\left(M\sum c_{ij}\alpha_i\right) \\ &= \mathrm{Tr}\left(\sum c_{ij}M\alpha_i \otimes \beta_j\right) \\ &= \mathrm{Tr}\left((M \otimes I^{B_2\cdots B_k})\sigma\right).\end{aligned}$$

□

Hence, by definition, if $\rho \in \mathrm{Ext}_k$ and $\mathrm{Tr}\left((M \otimes I^{B_2\cdots B_k})\,\sigma\right)$ for some exchangeable σ, it suffices to maximize over all exchangeable σ. Since our domain, the exchangeable density operators, is a convex set, the

maximum of our linear function must occur at one of the extreme points. The extreme points of our convex set are the pure states, $|\psi\rangle\langle\psi|, |\psi\rangle \in \mathrm{Sym}(k)$. Hence,

$$\begin{aligned}
\max \mathrm{Tr}\left((M \otimes I^{B_2 \cdots B_k})\sigma\right) &= \max \mathrm{Tr}\left((M \otimes I^{B_2 \cdots B_k})|\psi\rangle\langle\psi|\right) \\
&= \max \langle\psi|(M \otimes I^{B_2 \cdots B_k})|\psi\rangle \\
&= \text{largest eigenvalue of } M \otimes I^{B_2 \cdots B_k} \text{ projected onto } \mathbb{C}^d \otimes \mathrm{Sym}(k) \\
&= ||P(M \otimes I^{B_2 \cdots B_k})P||_2.
\end{aligned}$$

4.2 Computing $||P(M \otimes I^{B_2 \cdots B_k})P||_2$

Observe how $P(M \otimes I^{B_2 \cdots B_k})P$ can be written in the form

$$\sum_{\substack{i',i \in [d] \\ t',t \in [\binom{k+d-1}{d-1}]}} c_{i'it't} |i'\rangle\langle i| \otimes |v_{t'}\rangle\langle v_t|.$$

This is very useful, because what we have is equivalent to a $d\binom{k+d-1}{d-1}$ by $d\binom{k+d-1}{d-1}$ matrix with respect to the orthonomal basis $\{|i'\rangle\langle i| \otimes |v_{t'}\rangle\langle v_t| : i', i \in [d],\ t', t \in [\binom{k+d-1}{d-1}]\}$. Since eigenvalues are invariant under change of basis, we only need to compute the largest eigenvalue of a $d\binom{k+d-1}{d-1}$ by $d\binom{k+d-1}{d-1}$ matrix, instead of a d^{k+1} by d^{k+1} one. Time complexity wise, this is feasible when one of k, d is small, and for those cases, we can implement our algorithm to obtain the numerical data that may provide hints on the complexity of the approximation algorithm with respect to the precision.

Now we describe a method to write $P(M \otimes I^{B_2 \cdots B_k})P$ in our desired form. From now, let

$$\begin{aligned}
|v_t\rangle &= \frac{1}{\sqrt{|v_t|}} \sum_i |m_{ti}\rangle \otimes |n_{ti}\rangle, |m_{ti}\rangle \in B_1, |n_{ti}\rangle \in B_2 \otimes \cdots \otimes B_k, \\
&\quad |v_t| = \text{size of the class of strings indexed by } t, \\
&\quad i \text{ indexes distinct strings of the class}
\end{aligned}$$

and

$$M = \sum_{u,v,x,y} M_{uvxy} |u\rangle\langle v| \otimes |x\rangle\langle y| \otimes I^{B_2 \ldots B_k}.$$

Then,

$$\begin{aligned}
P(M \otimes I^{B_2 \cdots B_k})P &= \left(\sum_{t'} I^A \otimes |v_{t'}\rangle\langle v_{t'}|\right)\left(\sum_{u,v,x,y} M_{uvxy}|u\rangle\langle v| \otimes |x\rangle\langle y| \otimes I^{B_2 \ldots B_k}\right)\left(\sum_t I^A \otimes |v_t\rangle\langle v_t|\right) \\
&= \left(\sum_{t',i'_1,i'_2} I^A \otimes \frac{1}{|v_{t'}|}|m_{t'i'_1}\rangle\langle m_{t'i'_2}| \otimes |n_{t'i'_1}\rangle\langle n_{t'i'_2}|\right) \\
&\quad \left(\sum_{u,v,x,y} M_{uvxy}|u\rangle\langle v| \otimes |x\rangle\langle y| \otimes I^{B_2 \ldots B_k}\right) \\
&\quad \left(\sum_{t,i_1,i_2} I^A \otimes \frac{1}{|v_t|}|m_{ti_1}\rangle\langle m_{ti_2}| \otimes |n_{ti_1}\rangle\langle n_{ti_2}|\right) \\
&= \sum_{\substack{t',i'_1,i'_2 \\ u,v,x,y \\ t,i_1,i_2}} \frac{M_{uvxy}}{|v_{t'}||v_t|}|u\rangle\langle v| \otimes |m_{t'i'_1}\rangle\langle m_{t'i'_2}|x\rangle\langle y|m_{ti_1}\rangle\langle m_{ti_2}| \otimes |n_{t'i'_1}\rangle\langle n_{t'i'_2}|n_{ti_1}\rangle\langle n_{ti_2}|.
\end{aligned}$$

Thankfully, that last expression boils down to something simple. After close scrutiny, we notice how pdue to $\langle n_{t'i'_2}|n_{ti_1}\rangle$, the coefficient for t', t can be non-zero only if there exists a pair $\pi', \pi \in S_k$ such that the strings $\pi' v_{t'}, \pi v_t$ are identical everywhere except the first index, where they may or may not differ. Less formally, one can think of the strings as multisets where each element is in $[d]$, and in this case, the corresponding coefficient can be non-zero only if the multisets differ by on element or are identical. We will call such pairs of strings *close* pairs. If two strings are not *close*, then $\langle n_{t'i'_2}|n_{ti_1}\rangle$ clearly must vanish for all i'_2, i_1 and hence the coefficient must be zero.

This means we can compute the desired coefficients by iterating through all *close* pairs. To construct *close* pairs, we take every element in $\mathrm{Str}(k-1, d)$. Then, we append each distinct $(x, y) \in [d]^2$ to form a unique *close* pair. Furthermore, we can generate more *close* pairs by permuting the identical substrings indexed from 2 to k.

Now we examine the $\langle m_{t'i'_2}|x\rangle \langle y|m_{ti_1}\rangle$ part of the equation. For now, fix x and y. Clearly, there is only one $(m_{t'i'_2}, m_{ti_1})$ such that $\langle m_{t'i'_2}|x\rangle \langle y|m_{ti_1}\rangle$ does not vanish, namely (x, y). Hence, given any element of $\mathrm{Str}(k-1, d)$ as the value of $n_{t'i'_2} = n_{ti_1}$ sorted, if we are to generate a non-zero coefficient, it is necessary to use x, y for the two ms. To map to correct indices t', t, we sort the strings $xn_{t'i'_2}, yn_{ti_1}$ and then use the hash table described in section 2.3.2.

We can calculate the coefficient $c_{t'tuv}$ by iterating through $\mathrm{Str}(k-1, d)x[d]^2$ for each possible (u, v). Each iteration involves some element $n \in \mathrm{Str}(k-1, d)$, some $x, y \in [d]$, and some t', t which we determine using the method we just described. It contributes $\frac{M_{uvxy}}{\sqrt{|v_{t'}||v_t|}}$ times the number of permutations of the length $k-1$ string n.

These coefficients represent a sparse $d\binom{k+d-1}{d-1}$ by $d\binom{k+d-1}{d-1}$ matrix, and we use the power method compute its largest eigenvalue, the end result of our approximation algorithm.

Note that here, the coefficients were computed using the string representation. Needless to say, we could have also used type representation to do so. The details of that I will not explain sice the process is largely the same.

5 Sparsity of the matrix

Clearly, our $d\binom{k+d-1}{d-1}$ by $d\binom{k+d-1}{d-1}$ matrix is sparse, since based on how it was computed, there can be no more than $d^4\binom{k+d-2}{d-1} \leq d^2\binom{k+d-1}{d-1}^2$ non-zero elements. The power method of computing the largest eigenvalue works especially well for sparse matrices. Hence, to establish precisely the complexity of computing the largest eigenvalue of our matrix, we determine the exact number of non-zero entries.

Note that the number of non-zero elements is d^2 times the number of $(t', t) \in [\binom{k+d-1}{d-1}]^2$ such that (t', t) are *close*, because as mentioned in the previous section, if (t', t) is not a *close* pair, its corresponding matrix entry $C_{t't}$ must vanish. We now introduce some quick notation and identities which we use to calculate the number of non-zero entries.

Definition 5.1. *Let S denote the set of all $(t', t) \in [\binom{k+d-1}{d-1}]^2$ where (t', t) is close.*

Definition 5.2. *For fixed t', let $S_{t'} = \{(t', t) : t \in [\binom{k+d-1}{d-1}], (t', t) \in S\}$.*

Definition 5.3. *For $i \in [d]$ we say that t' contains i if and only if the string of t' contains i.*

Definition 5.4. *Let $f(t')$ denote the number of $i \in [d]$ such that t' contains i.*

Identity 5.5. $|S_{t'}| = 1 + (d-1)f(t')$.

Proof. The 1 is the special case where $t = t'$. In all other cases, one of the $f(t')$ numbers t' contains is pulled out of t' and replaced with one of the $d-1$ numbers in $[d]$ other than itself. By the rule of product, this would account for the $(d-1)f(t')$. □

Identity 5.6. $|S| = \binom{k+d-1}{d-1} + (d-1)\sum_{t'} f(t')$.

Proof.

$$\begin{aligned} |S| &= \sum_{t'} |S_{t'}| \\ &= \sum_{t'} 1 + (d-1)f(t') \qquad \textit{(by Identity 5.5)} \\ &= \tbinom{k+d-1}{d-1} + (d-1)\sum_{t'} f(t'). \end{aligned}$$

□

Identity 5.7. $\displaystyle\sum_{t'} f(t') = \sum_{i=1}^{d} i\binom{d}{i}\binom{k-1}{i-1}.$

Proof. We partition $[\binom{k+d-1}{d-1}]$ into d classes, $t'_i, i \in [d]$, where S_i is the subset defined by $f(t') = i$ for all its elements t'. Clearly, the union of these disjoint classes is $[\binom{k+d-1}{d-1}]$. Hence,

$$\sum_{t'} f(t') = \sum_{i=1}^{d} i|S_i|.$$

Any string in S_i contains exactly i numbers, and there are $\binom{d}{i}$ such combinations of i numbers. For each combination, there are $|\mathrm{Urn}(k-i, i)| = \binom{k-1}{i-1}$ distinct strings, since our size k string must contain at least one of each of our i numbers, and then we are left with $k-i$ elements to be split among our i numbers, or equivalently, we are left with $k-i$ indistinguishable marbles to be put into i distinguiable urns. Hence by rule of product, $|S_i| = \binom{d}{i}\binom{k-1}{i-1}$, and consequently,

$$\begin{aligned} \sum_{t'} f(t') &= \sum_{i=1}^{d} i|S_i| \\ &= \sum_{i=1}^{d} i\binom{d}{i}\binom{k-1}{i-1}. \end{aligned}$$

□

Identity 5.8. $\displaystyle\sum_{i=1}^{d} i\binom{d}{i}\binom{k-1}{i-1} = d\binom{k+d-2}{d-1}.$

Proof.

$$\begin{aligned} \sum_{i=1}^{d} \left(i\binom{d}{i}\right)\binom{k-1}{i-1} &= \sum_{i=1}^{d} d\binom{d-1}{i-1}\binom{k-1}{i-1} \\ &= d\sum_{i=0}^{d-1} \binom{d-1}{i}\binom{k-1}{i} \\ &= d\sum_{i=0}^{d-1} \binom{d-1}{d-1-i}\binom{k-1}{i} \\ &= d\tbinom{k+d-2}{d-1}. \qquad \textit{(by Vandermonde's identity)} \end{aligned}$$

□

5.1 Formula for number of non-zero entries

Now we are ready to calculate a formula for $d^2|S|$, the number of non-zero entries.

$$\begin{aligned}
d^2|S| &= d^2\left(\tbinom{k+d-1}{d-1} + (d-1)\sum_{t'} f(t')\right) && \textit{(by Identity 5.6)}\\
&= d^2\left(\tbinom{k+d-1}{d-1} + (d-1)\sum_{i=1}^{d} i\binom{d}{i}\binom{k-1}{i-1}\right) && \textit{(by Identity 5.7)}\\
&= d^2\left(\tbinom{k+d-1}{d-1} + d(d-1)\tbinom{k+d-2}{d-1}\right) && \textit{(by Identity 5.8)}\\
&= d^2\left(\tfrac{k+d-1}{k} + d(d-1)\right)\tbinom{k+d-2}{d-1}.
\end{aligned}$$

6 Complexity

Computing the entries of the matrix takes $Md^4\binom{k+d-2}{d-1}$ operations, where $M > 0$ is approximate to the number of operations in the innermost for loop. Asymptotically, for constant d, this is $\mathrm{O}(k^{d-1})$.

Computing the largest eigenvalue of our matrix via the power method takes a number of operations approximate to the number of iterations times the number of non-zeros entries, or $M'd^2\left(\frac{k+d-1}{k} + d(d-1)\right)\binom{k+d-2}{d-1}$, where M' = the number of iterations. Asymptotically, for constant d this is also $\mathrm{O}(k^{d-1})$.

For our purposes, we care more about the practical complexity than the theoretical complexity, because given our rather large constant coefficients on the number of operations, the computation would be rather slow for large k, especially as d becomes larger. For $d = 2$, the algorithm is linear, but as d increases, the algorithm becomes quadratic, cubic, quartic, and so on. For large d, this algorithm is not quite feasible, unless k is very small.

7 Conclusion

This algorithm has been implemented in GNU Octave and C++. We intend to use our fastest implementation to collect numerical data on how fast the maximum converges with respect to k. As mentioned in the introduction, this may help us establish a conjecture on the precision of the approximation, which we can attempt to prove. This paper is meant to focus on the mathematical and algorithmic aspects, and thus, we will not go into the programming and hardware sides of optimizing the runtime of this algorithm.

8 Acknowledgements

I would like to thank Professor Aram Harrow for being my mentor for this project. Aram introduced me to the injective tensor norm problem and its "quantum interpretation," and more generally, he guided me as I learned the fundamentals of quantum computing and information since March 2011. On both the learning side and the research side, Aram has provided me with invaluable guidance based on his knowledge and experience. Furthermore, my work has been supported by Aram's NSF Random Walks Research Experience for Undergraduates (REU) Grant.

I would also like to thank the graduate students of the Quantum Computing Theory Group at the University of Washington for helpful and inspiring discussions.

Finally, I would like to thank the Computer Science & Engineering department of the University of Washington, where I am currently an undergrad and where this research was done.

References

[1] A. C. Doherty, P. A. Parrilo, and F. M. Spedalieri. Complete family of separability criteria. *Physical Review A*, 69(022308), February 2004.

[2] A. Harrow. Injective tensor norms: Hardness and reductions. http://research.microsoft.com/apps/video/dl.aspx?id=148269, April 2011.

[3] J. Watrous. Cs 798 – advanced research topics - theory of quantum computation. http://www.cs.uwaterloo.ca/~watrous/quant-info/. Lectures 1-3, 21.