An Analysis of Gaussian Boson Sampling for Graph Isomorphism & Graph Classification using Supervised Machine Learning

Amanuel S. Anteneh
Distinguished Major Thesis in Computer Science
University of Virginia

April 2022

Advisor: Professor Olivier Pfister, Department of Physics
Second Reader: Professor Israel Klich, Department of Physics
This thesis is completed in fulfillment of the University of Virginia Department of Computer Science Distinguished Major Program.
This work lead to the discovery of the results presented in ArXiv:2301.01232 [quant-ph], forthcoming in Physical Review A.

Abstract

We analyze the complexity of a quantum algorithm based on Gaussian boson sampling for the GraphIsomorphism problem as well as develop a new quantum algorithm for the task of graph classification. We first analyze the complexity of the graph isomorphism algorithm under three different scaling assumptions of the two parameters that determine its time complexity: the number of modes M of the boson sampler, or equivalently the number of vertices of the graphs, and the number of photons n sent into the interferometer of the boson sampler. We then present a way of using Gaussian boson sampling for a less difficult but equally practical task: graph classification. This is done by defining a quantum feature map that maps graphs to a vector in a quantum Hilbert space. Our simulations and complexity analysis indicate that the quantum feature map is competitive in terms of classification accuracy and speed with current state-of-the-art graph kernels used for graph classification.

List of Symbols & Abbreviations

The next list describes several symbols and abbreviations that will be later used within the body of the document

$\lvert\Omega\rvert$	The number of elements in the sample space of the probability distribution we sample from
$\mathbf{n}$	A detection pattern generated from sampling a boson sampler
GBS	Gaussian boson sampler
PNRD	Photon number resolving detector
M	The number of modes of the boson sampler and the number of vertices in a graph
n	The total number of photons sent into the interferometer of the boson sampler
n_i	Number of photons detected in the ith mode of a boson sampler
$o(f(x))$	The set of all functions strictly upper bounded by the function $f(x)$
$w(f(x))$	The set of all functions strictly lower bounded by the function $f(x)$

Contents

1 Introduction **5**

1.1 Graph Theory . . . 6

1.1.1 Basic Theory & Notation . . . 6

1.1.2 Graph Isomorphism . . . 8

1.1.3 Graph Classification . . . 8

1.1.4 Supervised Machine Learning . . . 9

1.1.5 Algorithmic Complexity . . . 10

1.2 Quantum Optics . . . 11

1.2.1 Basic Theory & Notation . . . 11

1.2.2 Boson Sampling . . . 13

1.2.3 Gaussian Boson Sampling . . . 15

2 GBS for Graph Isomorphism **16**

2.1 Encoding a Graph into a GBS . . . 16

2.2 Testing for Isomorphism Between Graphs . . . 18

2.3 Analysis of Complexity . . . 18

3 GBS for Graph Classification **24**

3.1 Classical Graph Kernels . . . 24

3.2 The Current GBS Feature Maps . . . 24

3.2.1 The Xanadu Feature Map . . . 24

3.2.2 GBS Feature Map with Linear Space Complexity . . . 25

3.2.3 Numerical Simulation . . . 25

3.2.4 Preprocessing of Data Sets . . . 26

3.2.5 Model Selection . . . 26

3.2.6 Sample Complexity . . . 28

3.3 GBS Feature Map with Linear Space & Sample Complexity . . . 28

4 Summary and Open Problems **29**

Appendices **31**

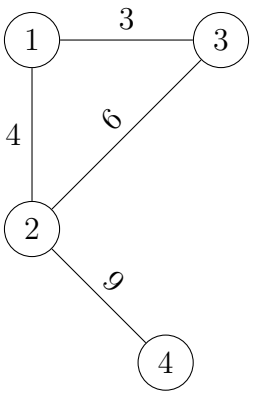
A Factorial Identity **31**

B Tight Bounds for $\lfloor x \rfloor$ and $\lceil x \rceil$ **32**

C Computational Complexity Theory **32**
C.1 Basic Theory & Notation 32
C.2 Complexity Classes 33

1 Introduction

Quantum computing promises a quantum computational advantage over classical computing, through the use of quantum algorithms, in terms of exponential speed-ups for some problems that may be intractable on classical computers, such as integer factorization, and some that are known to be intractable on classical computers such as quantum simulation. The power of these quantum algorithms comes from the fact that they exploit useful properties of quantum particles such as quantum entanglement and superposition. However for these quantum algorithms to be able to solve problems of non-trivial size they require quantum computers with many thousands of fault tolerant qubits. The development of these computers however is still very much in its early stages with quantum computers having even 1000 fault tolerant qubits being a massive challenge to construct. Due to this fact researchers have tried to find ways of demonstrating quantum computational advantage with the smallest number of resources (qubits) possible. The task many researchers have used to demonstrate quantum computational advantage is sampling. More specifically it is the task of sampling from a probability distribution that describes a quantum system. The now famous paper from Google [1] and the more recent paper from USTC, in China, [2] both demonstrate this by sampling from a probability distribution that describe a random quantum circuit and a random quantum optics network respectively. However these sampling tasks, on their own, are not useful for anything practical beyond the demonstration of quantum advantage which is why researchers have recently moved in the direction of trying to find useful applications for them. Reconstructing the whole probability distribution through sampling would provide useful information for applications but as we will show later reconstructing the whole distribution can not be done in a reasonable amount of time as the number samples required to do so is prohibitively high. The Toronto-based quantum start-up Xanadu has published research [3, 4, 5, 6] on how sampling from a probability distribution that depends on a quantum optics network can be used to solve graph based problems. The goals of this thesis were, first, to come to a thorough understanding of the advantages and limitations of the Xanadu approach and, second, to explore new research ideas, specifically for machine learning with graph based data, that build off these advantages while also trying to strengthen the weaknesses. The topics covered in this thesis and the research mentioned previously span a range of topics in quantum physics, theoretical computer science, and machine learning. The following section is meant to clearly define many of the terms and concepts used very often in the literature of the relevant sub-fields.



(a) Undirected weighted 4-vertex graph with 4 edges

$$\begin{bmatrix} 0 & 4 & 3 & 0 \\ 4 & 0 & 6 & 9 \\ 3 & 6 & 0 & 0 \\ 0 & 9 & 0 & 0 \end{bmatrix}$$

(b) Adjacency matrix of graph

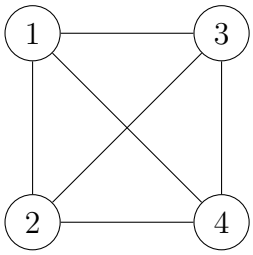
Figure 1: 4-vertex graph and its corresponding adjacency matrix

1.1 Graph Theory

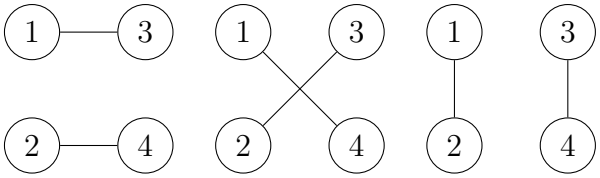
1.1.1 Basic Theory & Notation

In this paper we define a graph $G = (V, E)$ as a set of vertices $V = \{v_1, v_2, ...\}$ and a set of edges $E = \{v_1v_2, v_2v_1, ...\}$. Vertices of the graph are connect by the edges. The edges of the graph can have an edge weight $w_i \in \mathbb{R}$ associated with them. We will define a graph as unweighted if all edges have the same edge weight $w_i = 1\ \forall i$ and weighted otherwise. Graphs can be directed meaning the edges can point from one vertex to another. For example for a directed graph we can have the edge v_1v_2 be an arrow pointing from v_1 to v_2 but not from v_2 to v_1 in which case $v_1v_2 \neq v_2v_1$. For undirected graphs, which is what we will exclusively work with in this paper, $v_1v_2 = v_2v_1$. The degree of a vertex v is the number of edges that are connected to it. The maximum degree of a graph is the largest degree of a vertex in its vertex set. Graphs can be represented in a number of ways such as a diagram as in figure 1a or a more computationally useful way as an adjacency matrix. The adjacency matrix of an *undirected* graph G with n vertices is an $n \times n$ symmetric matrix A with entries a_{ij} where a_{ij} is the weight of the edge connecting vertices i and j. The *spectrum* of a graph is the set of eigenvalues of its adjacency matrix. An example of a graph with 4 vertices and 4 edges and its adjacency matrix is shown in figure 1. The Hafnian and the permanent of a matrix are two concepts that play a central role in the application of quantum optics to graph problems discussed later in this paper. The permanent of a $n \times n$ matrix A is defined as

$$\text{Perm}(A) = \sum_{\pi \in S_n} \prod_{i=1}^{n} A_{i,\pi(i)}, \tag{1}$$



(a) Complete graph of 4 vertices



(b) The three perfect matchings of the complete 4-vertex graph

Figure 2: The complete graph of 4 vertices and its corresponding perfect matching

where S_n is the set of all permutations of the list $(1, 2, 3..., n)$ and $\pi(i)$ is the ith element of the list π. The formula for the permanent is very similar to that of the determinant of an $n \times n$ matrix A

$$\mathrm{Det}(A) = \sum_{\pi \in S_n} \mathrm{sgn}(\pi) \prod_{i=1}^{n} A_{i,\pi(i)}, \tag{2}$$

where $\mathrm{sgn}(\pi)$ is the signature of the permutation π and is $+1$ when the permutation is even and -1 when it is odd. The presence of $\mathrm{sgn}(\pi)$ allows for the determinant to be calculated in a time that is polynomial in the matrix size, placing it in the computational complexity class P. However the problem of computing the permanent of a $n \times n$ matrix is in the complexity class $\#\mathsf{P}$ and in fact is $\#\mathsf{P}$-complete [7]. These complexity classes and others are explained in appendix C. The Hafnian of a $2n \times 2n$ matrix is defined in a similar form as

$$\mathrm{Haf}(A) = \sum_{\pi \in M_n} \prod_{(u,v) \in \pi} A_{u,v}, \tag{3}$$

where M_n is the partition of the set $\{1, 2, ..., 2n\}$ into unordered disjoint pairs. For example if $n = 2$ then $M_n = (\{(1, 2), (3, 4)\}, \{(1, 4), (2, 3)\}, \{(1, 3), (2, 4)\})$. If A is the adjacency matrix of a unweighted graph then the Hafnian is equal to the number of perfect matchings of the vertices of the graph. A perfect matching is a partition of the vertex set of a graph into pairs such that each vertex is connected to exactly one edge from the edge set. All perfect matchings of the complete 4-vertex graph are shown in figure 2. The permanent is related to the Hafnian via the relationship

$$\mathrm{Perm}(A) = \mathrm{Haf} \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}. \tag{4}$$

Therefore computing the Hafnian must be as difficult as computing the permanent.

1.1.2 Graph Isomorphism

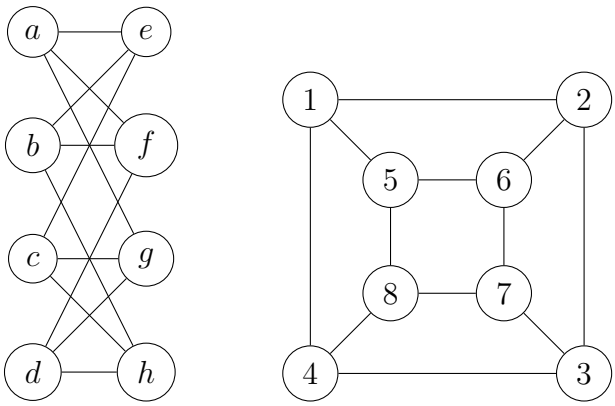
The GraphIsomorphism problem can be summarized as: given 2 graphs $G = (V, E)$ and $G' = (V', E')$, is there a bijection between the 2 vertex sets V and V' that preserves the adjacency of vertices that is, two vertices u and v in G are adjacent if and only if they are adjacent in G' under the bijection.

Definition 1 (GraphIsomorphism)**.** *Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. Then these graphs are isomorphic if and only if there exists a bijection $\phi : V \to V'$ that preservers the edges in the graph so if $v_i v_j \in E$ then $\phi(v_i)\phi(v_j) \in E'$.* GraphIsomorphism *is the problem of determining if such a bijection exists.*

An example of two isomorphic graphs and the bijection of their vertex sets is show in figure 3. The GraphIsomorphism problem is in the class NP but it is not known if it is NP-hard. As explained in [8] the belief that there exists an efficient quantum algorithm that solves GraphIsomorphism is supported by the fact that there exists one for the problem of integer factorization (Shor's algorithm) which has similar complexity to GraphIsomorphism and the fact that there already exists an adiabatic quantum-annealing method which solves the problem. Since the method is adiabatic its complexity is not known but since adiabatic and algorithmic quantum computing are equivalent [9] the implication is that there also exists an algorithmic solution to the problem. A graph invariant $I(G)$ of a graph G is a property that all isomorphic graphs share. Examples of invariants include number of vertices, number of edges, graph spectrum among others. One can show two graphs are *not* isomorphic if one has an invariant property the other doesn't. For example if two graphs have a different number of vertices they can not be isomorphic however having the same number of vertices is not enough to deduce the graphs are isomorphic. Such an invariant is precisely what was discovered in [6] for graphs encoded into a Gaussian boson sampler. A graph invariant is *complete* if $I(G) = I(G')$ implies G and G' are isomorphic. The second graph invariant presented in [6] is complete only for *isospectral* graphs that is, graphs whose adjacency matrices have equal sets of eigenvalues.

1.1.3 Graph Classification

In many real world applications of graph theory we aren't so much interested in if two graphs are isomorphic but rather if they belong to the same class. Graph classification is the problem of determining the class of a graph given a representation of it such as its adjacency matrix. For example chemical compounds can be represented as graphs with the vertices being atoms and and the edges being different types of bonds. We would then want to determine what class each molecules belongs to given a set of possible classes. For example



(a) Two isomorphic graphs

$$\phi(a) = 1$$
$$\phi(b) = 6$$
$$\phi(c) = 8$$
$$\phi(d) = 3$$
$$\phi(e) = 5$$
$$\phi(f) = 2$$
$$\phi(g) = 4$$
$$\phi(h) = 7$$

(b) The bijection between the two graphs

Figure 3: Two isomorphic graphs and their corresponding bijection

the graph data set PTC_FM is a collection of various molecules where the classes are the molecules' carcinogenicity for rodents. These types of classification problems can be handled via the use of supervised machine learning.

1.1.4 Supervised Machine Learning

Most supervised machine learning problems can be formulated as follows: given an initial input set $\mathcal{X} = (\vec{x}_1, ..., \vec{x}_m)$ of d-dimensional vectors and a set of class labels $\mathcal{Y} = (y_1, ..., y_m)$, where y_i is the corresponding class label for the element x_i, how can we use this input set and class label set to train a classifier that will be able to accurately classify new input sets. In practice the input set and class label set are represented as data tables where the rows are the elements (vectors) and columns are the dimensions (features) of the vectors. For example if we have a data set, as in figure 4, that consists of people who do and do not have dementia, along with information about each person such as hippocampus volume, blood pressure and age, the rows of the data table, and thus the elements in our input set, would be each patient and the columns would corresponds to age, blood pressure, hippocampus volume and dementia diagnosis. The dementia diagnosis column would be our class label set with each entry either being 1 or 0 corresponding to whether the patient is diagnosed with dementia or not. The task would now be to train machine learning classifiers with this data so that they can accurately classify new patients. In machine learning a feature map is a way of mapping data to higher dimensional spaces to make tasks like classification easier. This is because data that is not linearly separable in its original space can become linearly separable when mapped to a higher dimensional space. An example of such an instance is given in figure 5.

	d_1	d_2	d_3	$\mathcal{Y}$
$\mathcal{X}$	3.3	54	0.3	1
	5.6	62	0.2	0
	4.1	34	0.8	1

Figure 4: Example data table for ML applications. We have three data points (rows) which are the elements of $\mathcal{X}$ and three features (d_1,d_2,d_3) which are the dimensions of the vectors in $\mathcal{X}$, and the label column $\mathcal{Y}$ which contains the class label of each data point.

Definition 2 (**Feature Map**). *Let $\mathcal{F}$ be a Hilbert space which we call the feature space, $\mathcal{X}$ an input set and x an element from the input set. A feature map is a map φ: $\mathcal{X} \to \mathcal{F}$ from inputs to vectors in the Hilbert space. The vectors $\varphi(x) \in \mathcal{F}$ are called feature vectors.*

A kernel is a function that measures the similarity of two elements from the input set in the feature space. Kernel methods refer to machine learning algorithms that learn by comparing pairs of data points using this similarity measure [10]. In our context we have a set of graphs $\mathcal{G}$ and we call a kernel, κ, a *graph kernel* if it is of the form $\kappa : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$.

Definition 3 (**Kernel Function**). *Let $\mathcal{X}$ be a nonempty set, called the input set. A function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \times \mathcal{X}$ is the set of all ordered pairs (x_i, x_j) with $x_i, x_j \in \mathcal{X}$, is called a kernel if the Gram matrix K with entries $K_{i,j} = \kappa(x_i, x_j)$, where $i, j \in (1, 2, .., m)$, is positive semidefinite, in other words, if for any finite subset $\{x_1, ..., x_m\} \subseteq \mathcal{X}$ with $m \geq 2$,*

$$\lambda \geq 0 \quad \forall \lambda \in \{\lambda_1, ..., \lambda_m\}, \tag{5}$$

where $\{\lambda_1, ..., \lambda_m\}$ is the set of eigenvalues of the Gram matrix.

The inner product $\kappa(x, x') = \langle \phi(x), \phi(x') \rangle$ is the classic example of a kernel function and since all feature spaces are Hilbert spaces, and therefore have an inner product defined on them, all feature maps give rise to a kernel. Since graph based data is most often represented as an adjacency matrix it is useful to define a feature map, and thus a kernel function, that maps each graphs adjacency matrix to a feature vector thereby allowing us to use them as inputs to machine learning classifiers such as a support vector machine (SVM).

1.1.5 Algorithmic Complexity

In computer science and machine learning when designing algorithms we want to quantify the amount of resources an algorithm will need to solve a problem as the size of the problem increases. We define time complexity as the number of elementary operations, such as

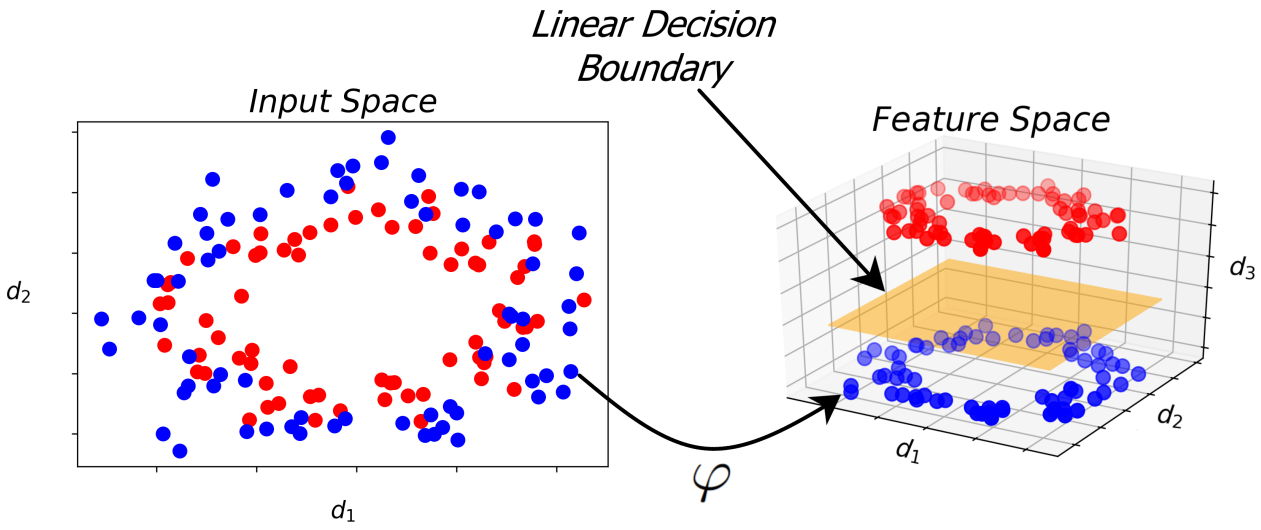


Figure 5: In the original 2-D input space the data points, which belong either to the class 'red' or 'blue', are not separable by a linear function (the decision boundary) but after mapping the points to feature vectors in a higher dimensional space a linear function is able to separate the two classes. This linear decision boundary is calculated by the supervised machine learning models like the SVM.

multiplication of scalars, an algorithm must perform as the size of the problem it is solving increases. We define space complexity as the amount of memory an algorithm requires to solve a problem as the size of the problem increases. Lastly we will define the sample complexity of an algorithm as how many times it must sample from a particular distribution that describes the problem we are trying to solve to accurately approximate it's probabilities.

1.2 Quantum Optics

1.2.1 Basic Theory & Notation

In quantum mechanics the standard way of denoting the state of a particle is through Dirac notation. For example a spin-$\frac{1}{2}$ particle can either be in the state $+z$ or $-z$ or any superposition of the two states. So in general we can denote the quantum state of the particle, $|\psi\rangle$, as

$$|\psi\rangle = c_+ |+z\rangle + c_- |-z\rangle \,. \tag{6}$$

Where $|c_+|^2$ and $|c_-|^2$ are the probabilities that the particle is in the $+z$ and $-z$ state respectively upon measurement. A spin-$\frac{1}{2}$ particle is one model of what's called a qubit: the quantum computing analogy of a classical bit. For a system of N particles we can denote the quantum state of the system as the tensor product of the states of each of the individual

particles

$$|\Psi\rangle = \bigotimes_{i=1}^{N} |\psi_i\rangle = |\psi_1\rangle \otimes ... \otimes |\psi_N\rangle = |\psi_1, ..., \psi_N\rangle \,. \tag{7}$$

By contrast in quantum optics instead of qubits the system used to represent bits are the modes of the quantized electromagnetic field, called qumodes. While qubits are described by a two-dimensional Hilbert space, qumodes are described by an infinite dimensional Hilbert space. A general qumode state can be written as

$$|\psi\rangle = \sum_{n=0}^{\infty} c_n \, |n\rangle \,, \tag{8}$$

where we have the basis states $|0\rangle \,, |1\rangle \,, |2\rangle \,, ...$ which are known as Fock states. The Fock state $|n\rangle$ can be interpreted as a qumode which contains n photons. The ith qumode of a quantum system can be described as the familiar quantum harmonic oscillator. The Hamiltonian of a mechanical oscillator is defined as

$$H_i = \frac{\hat{p_i}^2}{2m} + \frac{k\hat{x_i}^2}{2}. \tag{9}$$

The dimensionless quadrature operators Q and P are defined as

$$Q = \beta \hat{x} \tag{10}$$

$$P = \frac{\hat{p}}{\beta \hbar}, \tag{11}$$

where β has dimensions of inverse length and is defined as

$$\beta = \sqrt{\frac{m\omega}{\hbar}} \tag{12}$$

with $\omega = \sqrt{\frac{k}{m}}$ being the resonance frequency of the quantum harmonic oscillator. We also define the non-Hermitian annihilation and creation ladder operators a and $a^\dagger$ as

$$a = \frac{1}{\sqrt{2}}(Q + iP) \tag{13}$$

$$a^\dagger = \frac{1}{\sqrt{2}}(Q - iP) \tag{14}$$

The quadrature operators can also be written in terms of the ladder operators as

$$Q = \frac{1}{\sqrt{2}}(a + a^\dagger) \tag{15}$$

$$P = \frac{1}{i\sqrt{2}}(a - a^\dagger) \tag{16}$$

The state of a quantum system with M modes can be uniquely characterized by its Wigner function $W(\boldsymbol{q}, \boldsymbol{p})$ where $\boldsymbol{q} \in \mathbb{R}^M$ and $\boldsymbol{p} \in \mathbb{R}^M$ are the position and momentum quadrature vectors respectively. A Gaussian state is a state whose Wigner function is a Gaussian and can be described by a $2M \times 2M$ covariance matrix $\boldsymbol{V}$ and two M-dimensional vectors of means $\bar{\boldsymbol{q}}$, $\bar{\boldsymbol{p}}$. It is often useful to write the covariance matrix in terms of the complex amplitude $\boldsymbol{\alpha} = \frac{1}{\sqrt{2\hbar}}(\boldsymbol{q} + i\boldsymbol{p}) \in \mathbb{C}^M$, which is complex-normal distributed with mean $\bar{\boldsymbol{\alpha}} = \frac{1}{\sqrt{2\hbar}}(\bar{\boldsymbol{q}} + i\bar{\boldsymbol{p}}) \in \mathbb{C}^M$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{C}^{2M \times 2M}$. Lastly the modes of the system can have displacements applied to them. This displacement is described by a displacement vector $\mathbf{d} = (d_1, ..., d_M, d_1^*, ..., d_M^*)^T \in \mathbb{C}^{2M}$. A displacement changes the mean of the M-mode Gaussian quantum state but leaves the covariance matrix unaffected.

1.2.2 Boson Sampling

Definition 4 (BosonSampling)**.** *Let M be the number of modes in a linear interferometer whose input is n single-photon Fock states, $|1\rangle$, in the first n modes with the other $M-n$ modes being vacuum states $|0\rangle$. Let $\hat{U}$ be the unitary matrix of the interferometer acting on the M modes. Let $\mathcal{D}_{\hat{U}}$ be the probability distribution of elements of the sample space, which consists of all lists $(s_1, .., s_M)$ of non-negative integers satisfying $s_1 + \cdots + s_M = n$.* BosonSampling *is the problem of sampling either exactly or approximately from the distribution $\mathcal{D}_{\hat{U}}$. Where sampling approximately from the distribution means the algorithm samples from a distribution $\mathcal{D'}_{\hat{U}}$ such that $||\mathcal{D'}_{\hat{U}} - \mathcal{D}_{\hat{U}}||_1 \leq \epsilon$ for some error ϵ where $||\cdot||_1$ is the L_1 distance.*

At its most basic level boson sampling consists of passing photons thorough a linear interferometer, that consists of beamsplitters and phase-shifters, and observing their output configurations which we call detection events and denote as $\mathbf{n} = (n_1, ..., n_M)$ where n_i is the number of photons detected in the ith mode. In the quanutm mechanics formulation this output pattern can be written as the quantum state

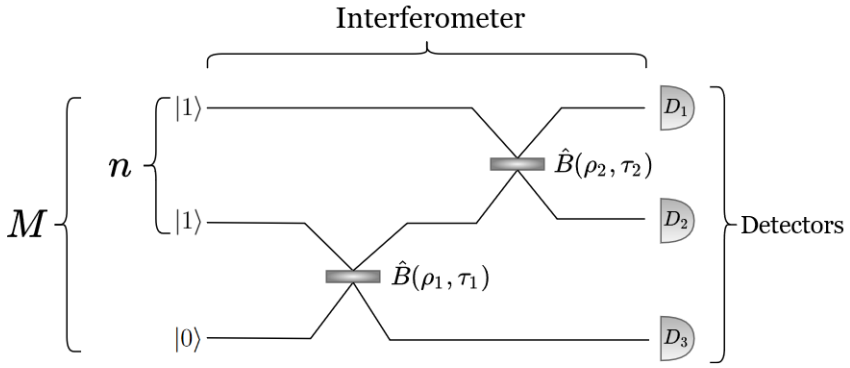
$$|\Psi\rangle = |n_1\rangle \otimes ... \otimes |n_M\rangle = |n_1, ..., n_M\rangle\,. \tag{17}$$

The detectors in the device can be either photon number resolving detectors (PNRDs) or threshold detectors. If the detectors are PNRDs then they count how many photons are

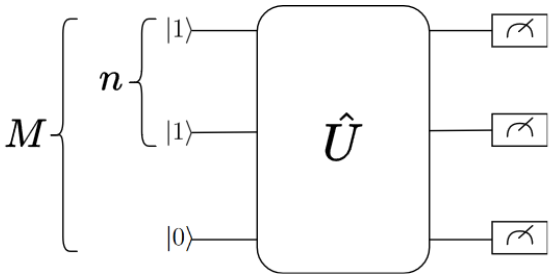
detected in that mode so $n_i \in \mathbb{N}$. If the detectors are threshold detectors then they simply detect if any photons have been detected in that mode or not. So a 'click' would correspond to detecting one (or several) photon(s), $n_i = 1$, and no click corresponds to no photons detected, $n_i = 0$, so $n_i \in \{0, 1\}$. We will focus on GBS with PNRDs for the first part of the thesis. A boson sampler is a non-universal model of photonic quantum computing, meaning it's not as computationally powerful as a quantum computer with error correction. However simulation of sampling from a boson sampler has been argued to be intractable on a classical computer. This was first argued by Aaronson & Arkhipov in [11] where they prove that the existence of a polynomial-time classical algorithm that samples *exactly* from the probability distribution of a boson sampler would mean that $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$ and thus the polynomial hierarchy would collapse to the third level which would be a shocking result in complexity theory as explained in appendix C. They also show that if there exists a fast classical algorithm that samples *approximately* from the probability distribution of a boson sampler this would mean that there is also a $\mathsf{BPP}^{\mathsf{NP}}$ algorithm for estimating $|\mathrm{Perm}(X)|^2$ with high probability for a Gaussian random matrix $X \in \mathbb{R}^{n\times n}$. This fact, along with two conjectures that Aaronson & Arkhipov give evidence for but do not prove, would also imply the collapse of the polynomial hierarchy to the third level. They also show that for a M-mode boson sampler with photon detection events $\mathbf{n} = (n_1, n_2, ..., n_M)$, where $n_i \in \mathbb{N}$, the probability of detecting a specific output photon pattern $\mathbf{n}$ is given by

$$p(\mathbf{n}) = \frac{|\mathrm{Perm}(U_{\mathbf{n},\mathbf{m}})|^2}{\mathbf{n}!\mathbf{m}!}. \tag{18}$$

Where $\mathbf{n}! = n_1!n_2!\cdots n_M!$, $\mathbf{m}$ is the input photon pattern as opposed to the output photon pattern $\mathbf{n}$ and $\mathbf{m}! = m_1!m_2!\cdots m_M!$ and $\mathrm{Perm}(U_{\mathbf{n},\mathbf{m}})$ is the permanent of the matrix $U_{\mathbf{n},\mathbf{m}}$. The matrix $U_{\mathbf{n},\mathbf{m}}$ is a submatrix of the linear transformation that characterizes the interferometer and it depends on both the input and output patterns of the photons [12, 13]. Since calculating the permanent of a matrix is known to be #P-complete the calculation becomes quickly intractable as the dimensions of the matrix increase. Because calculating the probabilities of measurement outcomes for a boson sampler requires calculating the permanent of $2M \times 2M$ matrices it follows that the BosonSampling problem is also classically hard, at least for the exact case. Thus the hardness of classically simulating a boson sampler along with the relatively small number of resources needed to implement it makes it a good candidate for demonstrating quantum advantage with minimal resources, with the caveat that BosonSampling is a *sampling* problem which is fundamentally different from a *decision* problem, such as GraphIsomorphism, potentially making it more difficult to find computational applications.



(a) The first n modes start in the single photon Fock state $|1\rangle$ with the other $M - n$ modes in the vacuum state $|0\rangle$ before being sent through the network of beamsplitters (the interferometer).



(b) As a quantum circuit the interferometer is represented by a unitary quantum gate $\hat{U}$

Figure 6: Diagram of 3-mode boson sampler both in its experimental form and circuit form

1.2.3 Gaussian Boson Sampling

Due to certain hardware constraints, in particular the lack of deterministic single-photon sources, other variations of boson sampling have been proposed such as Gaussian boson sampling which replaces the single-photon sources with squeezers. A M-mode Gaussian boson sampler (GBS), depicted in figure 7, consists of an array of M squeezers outputting squeezed light to an interferometer that is made up of beam splitters between any two of the M modes of squeezed light followed by M photon number resolving detectors to detect how many photons are in each mode after the photons pass through the interferometer. The squeezers take in a vacuum state and produce photons whose quantum state is a superposition of even photon Fock states. For example the state of photons in each mode before entering the interferometer is written as

$$\hat{S}(r_i)\,|0\rangle = \frac{1}{\sqrt{\cosh r_i}} \sum_{n=0}^{\infty} (\tanh r_i)^n \frac{\sqrt{(2n)!}}{2^n n!}\,|2n\rangle\,. \tag{19}$$

Where r_i is the squeezing parameter of the ith squeezer and n is the photon number. In terms of the annihilation and creation operators, a_i and $a_i^\dagger$, on the ith mode the squeezing

operator can be written as

$$S(r_i) = \exp\left[\frac{r_i}{2}(a_i^{\dagger 2} - a_i^2)\right]. \tag{20}$$

The only difference introduced with Gaussian boson sampling as opposed to normal boson sampling is the replacement of the single photon sources with single-mode squeezers. However now that the photons are in a superposition of even photon number Fock states the number of photons being sent into the interferometer is no longer constant. In fact the number of photons in the ith mode can be any $2n \in \mathbb{N}$ with probability

$$p(2n) = \left|\frac{(\tanh r_i)^n}{\sqrt{\cosh r_i}}\frac{\sqrt{(2n)!}}{2^n n!}\right|^2 \tag{21}$$

This function decreases very quickly with n so very high photon counts are less likely. Given this it would be more appropriate to talk of an average photon number, $\bar{n}$, being sent into the interferometer. However in practice we can simply post-select samples that have the the total photon number, n, that we want to send into the device. The probability of measuring a specific photon pattern with a GBS is now characterized by the Hafnian as opposed to the permanent of a matrix

$$p(\mathbf{n}) = \frac{1}{\sqrt{\det(Q)}}\frac{\mathrm{Haf}(A_{\mathbf{n}})}{\mathbf{n}!} \tag{22}$$

where

$$Q = \mathbf{\Sigma} + \mathbb{I}_{2M}/2, \quad A = X(\mathbb{I}_{2M} - Q^{-1}), \quad X = \begin{bmatrix} 0 & \mathbb{I}_M \\ \mathbb{I}_M & 0 \end{bmatrix}. \tag{23}$$

Here $\mathbf{\Sigma}$ is the $2M \times 2M$ covariance matrix that fully describes the Gaussian state. $A_{\mathbf{n}}$ is related to A, the $2M \times 2M$ symmetric matrix, via the following relationship: $A_{\mathbf{n}}$ is obtained by repeating rows and columns i and $i + M$ according to the measurement pattern $\mathbf{n}$. If $n_i = 0$ rows and columns i and $i+M$ are deleted from A but if $n_i > 0$ then rows and columns i and $i + M$ are repeated n_i times. For example the probability of detecting the event where each mode has exactly one photon $\mathbf{n} = (1, 1, ..., 1)$ would be proportional to the Hafnian of the original matrix A since $A_{\mathbf{n}} = A$.

2 GBS for Graph Isomorphism

2.1 Encoding a Graph into a GBS

To conduct the isomorphism test it is first necessary to encode the two graphs into GBS devices. This can be done by mapping the adjacency matrix of the graph A to the symmetric, positive definite $2M \times 2M$ covariance matrix of a Gaussian state of M modes. First a doubled

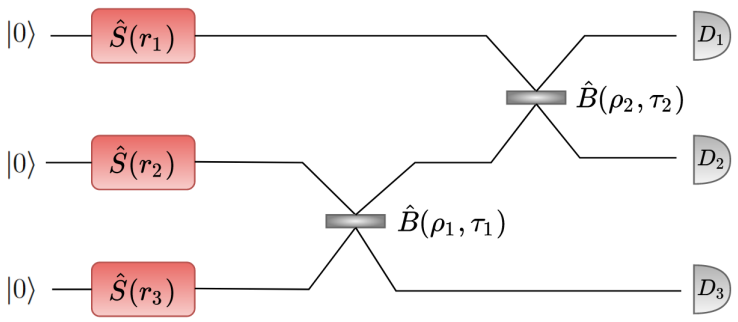


Figure 7: Diagram of an example of a 3-mode Gaussian boson sampler. Each mode starts in the vacuum state $|0\rangle$ before having squeezer $\hat{S}(r_i)$ applied to it and then passed through the network of beamsplitters (the interferometer).

adjacency matrix is constructed,

$$\tilde{A} = c \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} = c(A \oplus A), \tag{24}$$

where c is a rescaling constant chosen such that $0 < c < 1/\lambda_{\max}$ where $\lambda_{\max}$ is the maximum singular value of A [5]. $\tilde{A}$ can now be used to determine the parameters of the quantum gates that compose the interferometer by first taking the Takagi-Autonne decomposition to obtain

$$\tilde{A} = U \text{diag}(\lambda_1, ..., \lambda_M) U^T, \tag{25}$$

where U is the unitary matrix that characterizes the interferometer and the λ's are the matrix's eigenvalues and uniquely determine the squeezing parameters r_i of the squeezers via the relationship $\tanh(r_i) = \lambda_i$ [14]. The eigenvalues uniquely determine the mean photon number $\bar{n}$ of the distribution as well according to

$$\bar{n} = \sum_{i=1}^{M} \frac{\lambda_i^2}{1 - \lambda_i^2}. \tag{26}$$

U can be further decomposed to give the parameters of the beamsplitter and rotation gates of the interferometer [15]. It is also worth noting that by using the identity $\text{Haf}(\tilde{A}) = \text{Haf}(A \oplus A) = \text{Haf}^2(A)$ we can write the original formula for $p(\mathbf{n})$ as

$$p(\mathbf{n}) = \frac{1}{\sqrt{\det(Q)}} \frac{\text{Haf}(\tilde{A}_\mathbf{n})}{\mathbf{n}!} = \frac{1}{\sqrt{\det(Q)}} \frac{|\text{Haf}(A_\mathbf{n})|^2}{\mathbf{n}!}. \tag{27}$$

2.2 Testing for Isomorphism Between Graphs

Once a graph is encoded into the M-mode GBS apparatus we can send a certain number of photons, n, through it and and obtain a measurement outcome of the form $(n_1, n_2, n_3, ..., n_M)$ where n_i is the number of photons detected by the ith detector. For boson sampling with no photon loss n is also the sum of all photons detected by the PNRDs at the end of the interferometer. This means that for a given measurement outcome for example for a 4-mode interferometer (n_1, n_2, n_3, n_4), where we have $M = 4$, we have $\sum_{i=1}^{4} n_i = n$. And in general we have $\sum_{i=1}^{M} n_i = n$ where M is the the number of modes of the interferometer. The important property of the GBS for the quantum graph isomorphism algorithm is its probability distribution. The probability distribution of a GBS consists of the possible measurement outcomes $\mathbf{n}$ and their probabilities $p(\mathbf{n})$. The measurement outcome $\mathbf{n}$ of a boson sampler can be thought of as a M-dimensional random vector $\mathbf{X} = (X_1, X_2, ..., X_M)$ with the X_i's being multinomial random variables since a PNRD measurement n_i can result in an outcome from the set $\{0, 1, 2, ..., n\}$. The boson sampler thus returns samples from the multivariate normal distribution of the random vector $\mathbf{X} \sim \mathcal{N}_M(\mu, \sigma)$ where the covariance matrix of the distribution σ corresponds to the the doubled adjacency matrix $\tilde{A}$ and the mean vector is related to the displacement vector $\mathbf{d}$ by $\mu = Q^{-1}\mathbf{d}^{\dagger}$, where $Q = \mathbf{\Sigma} + \mathbb{I}_{2M}/2$. In [6] it was shown that the output probability distribution from GBS is a graph invariant up to a permutation. This means that if two isomorphic graphs are encoded into a GBS their output probability distributions of photon events will be equivalent up to a permutation. Thus it is possible to show 2 graphs are not isomorphic by sampling from their GBS devices and comparing the resulting probability distributions to see if they are permutations of one another which can be done in $\mathcal{O}(D^2)$ time where D is the number of elements in the distribution.

2.3 Analysis of Complexity

A crucial question to answer for the complexity analysis of this algorithm is how many times one has to sample from a given probability distribution until the sample distribution begins to approximate the sampled distribution within a given error range. In [16] the following theorem was presented:

Theorem 1. *Let $\mathcal{D}$ be a probability distribution over the sample space $\Omega = \{1, 2, ..., D\}$ and $\mathcal{D}'$ the empirical probability distribution obtained from sampling. For a given $\epsilon > 0$ and $\delta > 0$,*

$$S = \left\lceil \frac{2(\ln(2)D + \ln(\frac{1}{\delta}))}{\epsilon^2} \right\rceil. \tag{28}$$

samples are needed to ensure that $p(||\mathcal{D} - \mathcal{D}'||_1 \geq \epsilon) \leq \delta$. Where $||\mathcal{D} - \mathcal{D}'||_1$ is the L_1 norm

or distance.

In other words the number of samples S needed to approximate a probability distribution (the sampled distribution) with D outcomes or more precisely D elements in its sample space, with probability of at most δ that the sum of the absolute values of the errors on the empirical probability distribution (the sample distribution) is ϵ or greater is

$$S = \left\lceil \frac{2(\ln(2)D + \ln(\frac{1}{\delta}))}{\epsilon^2} \right\rceil. \tag{29}$$

This shows the number of samples needed to accurately approximate the distribution scales linearly with the number of elements in the sample space (for constant δ and ϵ). Therefore it is important to analyze how the number of outcomes of the boson sampler, which is also the number of elements in the sample space which we denote as $|\Omega|$, grows as we increase the size of the graphs and thus the size of the interferometer (the number of modes M).

Theorem 2. *The size $|\Omega|$ of the sample space Ω of a M-mode boson sampler with n photons as inputs grows at least exponentially with the number of modes under the three approximations:*

1. *The number photons scales quadratically with the number of photons: $n \in \Theta(M^2)$*
2. *The number modes scales quadratically with the number of photons: $M \in \Theta(n^2)$*
3. *The number photons scales linearly with the number of modes: $n \in \Theta(M)$*

The number of outcomes for a 4-mode interferometer is equal to the number of ways one can write $n_1 + n_2 + n_3 + n_4 = n$ where the order of the summands matters. In mathematics this is called the number of weak compositions of the integer n into M parts and it is equal to

$$|\Omega| = \binom{n + M - 1}{n} = \frac{(n + M - 1)!}{n!(M - 1)!}. \tag{30}$$

First we will show that under the approximation that the number of photons is quadratic in the number of modes, $n \in \Theta(M^2)$, this function is in $\omega(2^M)$ and thus grows more than exponentially with M. While n is an important variable for the isomorphism algorithm's complexity we chose to analyze growth with M as it determines the size of the graph.

Lemma 1. $\frac{(n+M-1)!}{n!(M-1)!} \in \omega(2^M)$ *for* $n = M^2$

Proof.

$$\lim_{M\to\infty} \frac{\frac{(n+M-1)!}{n!(M-1)!}}{2^M} \xrightarrow{n=M^2} \lim_{M\to\infty} \frac{(M^2 + M - 1)!}{(M^2)!(M-1)!2^M}$$

Using the identity from appendix A:

$$= \lim_{M\to\infty} \frac{1}{2^M} \frac{[\prod_{i=1}^{M-1}(M^2+i)](M^2)!}{(M^2)!(M-1)!} = \lim_{M\to\infty} \frac{1}{2^M} \frac{\prod_{i=1}^{M-1}(M^2+i)}{(M-1)!}$$

Since $\prod_{i=1}^{M-1}(M^2+i) > (M^2)^{M-1}$:

$$> \lim_{M\to\infty} \frac{1}{2^M} \frac{(M^2)^{M-1}}{(M-1)!}$$
$$= \lim_{M\to\infty} \frac{M^{2M}}{2^M M^2 (M-1)!} = \lim_{M\to\infty} \frac{M^M M^M}{2^M M^2 (M-1)!}$$
$$= \lim_{M\to\infty} \frac{M^M}{2^M M} \cdot \lim_{M\to\infty} \frac{M^M}{M!} = \infty \cdot \infty = \infty$$

Therefore $\lim_{M\to\infty} \frac{\frac{(n+M-1)!}{n!(M-1)!}}{2^M} = \infty$ and $\frac{(n+M-1)!}{n!(M-1)!} \in \omega(2^M)$ for $n = M^2$. □

If we assume the reverse scaling $M \in \Theta(n^2)$ then a similar conclusion can be reached. We use $\lfloor\sqrt{M}\rfloor$ as $x!$ is only defined for integers.

Lemma 2. $\frac{(n+M-1)!}{n!(M-1)!} \in \omega(\sqrt{M}^{\sqrt{M}})$ *for* $n = \lfloor\sqrt{M}\rfloor$

Proof.

$$\frac{(n+M-1)!}{n!(M-1)!} \xrightarrow{n=\lfloor\sqrt{M}\rfloor} \frac{(\lfloor\sqrt{M}\rfloor+M-1)!}{(\lfloor\sqrt{M}\rfloor)!(M-1)!}$$
$$= \frac{[\prod_{i=1}^{\lfloor\sqrt{M}\rfloor}(M-1+i)](M-1)!}{[\prod_{i=1}^{\lfloor\sqrt{M}\rfloor} i](M-1)!} = \frac{[\prod_{i=1}^{\lfloor\sqrt{M}\rfloor}(M-1+i)]}{[\prod_{i=1}^{\lfloor\sqrt{M}\rfloor} i]}$$
$$= \prod_{i=1}^{\lfloor\sqrt{M}\rfloor} \frac{M-1+i}{i} = \prod_{i=1}^{\lfloor\sqrt{M}\rfloor} [\frac{M-1}{i}+1]$$
$$> \prod_{i=1}^{\lfloor\sqrt{M}\rfloor} [\frac{M-1}{\lfloor\sqrt{M}\rfloor}+1]$$
$$= (\frac{M-1}{\lfloor\sqrt{M}\rfloor}+1)^{\lfloor\sqrt{M}\rfloor} = (\lfloor\sqrt{M}\rfloor+1-\frac{1}{\lfloor\sqrt{M}\rfloor})^{\lfloor\sqrt{M}\rfloor}$$
$$\geq \lfloor\sqrt{M}\rfloor^{\lfloor\sqrt{M}\rfloor}$$

Therefore $\frac{(n+M-1)!}{n!(M-1)!} \in \omega(\sqrt{M}^{\sqrt{M}})$ for $n = \lfloor\sqrt{M}\rfloor$. We drop the floor function for simplicity as $\lfloor x\rfloor \in \Theta(x)$ as shown in appendix B. □

For linear scaling, $n \in \Theta(M)$, the following lower bound can also be established.

Lemma 3. $\binom{n+M-1}{n} \in \omega((2-\frac{1}{M})^M)$ *for* $n = M$

Proof.

$$\frac{(n+M-1)!}{n!(M-1)!} \xrightarrow{n=M} \frac{(M+M-1)!}{M!(M-1)!}$$
$$= \frac{[\prod_{i=1}^{M}(M-1+i)](M-1)!}{[\prod_{i=1}^{M} i](M-1)!}$$
$$= \frac{[\prod_{i=1}^{M}(M-1+i)]}{[\prod_{i=1}^{M} i]}$$
$$= \prod_{i=1}^{M} \frac{M-1+i}{i} = \prod_{i=1}^{M}[\frac{M-1}{i}+1]$$

Since $M! < M^M$:

$$> \prod_{i=1}^{M}[\frac{M-1}{M}+1]$$
$$= (\frac{M-1}{M}+1)^M = (2-\frac{1}{M})^M$$

Therefore $\frac{(n+M-1)!}{n!(M-1)!} \in \omega((2-\frac{1}{M})^M)$ for $n = M$. □

Proof of Theorem 2. The proof follows directly from Lemmas 1, 2 and 3. □

Since the number of possible outcomes for a boson sampler increases exponentially with the number of modes M as does the cardinality of its sample space and it follows that the number of samples needed to sufficiently approximate its output probability distribution also grows exponentially with the number of modes and thus also exponentially with the number of vertices of the graph. Therefore testing for isomorphism between two graphs would require sampling exponentially many times from the GBS devices making the test intractable.

A method was suggested in [6] to coarse-grain the probability distributions by combining outcomes into groups called orbits. Coarse-graining in this sense means to reduce the size of the sample space via this grouping thereby requiring us to take less samples to approximate the new probability distribution. However it is important to keep in mind that this comes with two downsides: i) the new distribution could potentially be simple enough to be sampled from classically, thereby eliminating the quantum advantage, and ii) the new distribution having less information and distinguishing power in terms of differentiating graphs based on isomorphism. Even with these downsides coarse-graining strategies are still useful to investigate in the case that the new distribution could still decide isomorphism in which case it doesn't matter if it can be efficiently sampled from classically as we would still have an efficient algorithm that solves the GraphIsomorphism problem. It is shown in [6] that orbit probabilities are enough to distinguish non-isomorphic *isospectral* graphs. An orbit $O_{\mathbf{n}}$

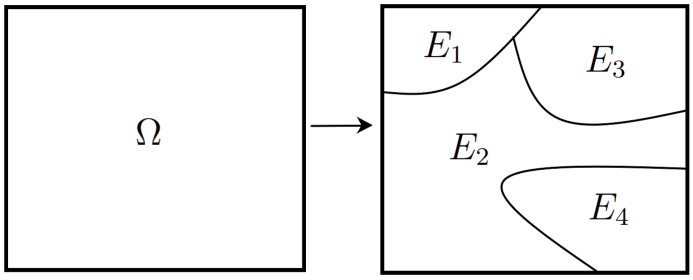


Figure 8: Idea behind coarse-graining the sample space. The sample space Ω could be of very large cardinality which would make approximating it's probability distribution via sampling prohibitively expensive. However if we group the elements of Ω in such a way that partitions the sample space into disjoint subsets we get a new probability distribution over the subsets ($E_1, E_2, E_3,$ & E_4 in this case) which is less expensive to approximate and contains relevant information about the original distribution.

consists of a detection event $\mathbf{n}$ and all of its permutations. For example the orbit that contains the detection event $\mathbf{n} = (1, 2, 2)$ also contains the detection events $(2, 1, 2)$ and $(2, 2, 1)$. The number of orbits for a 4-mode interferometer is equal to the number of ways one can write $n_1 + n_2 + n_3 + n_4 = n$, where the order of the summands does not matter. In mathematics this is called the number of integer partitions of the integer n into M parts and from the number theory literature [17] it has the upper bound

$$|\Omega| \leq \frac{5.44}{n-M} e^{\pi\sqrt{\frac{2(n-M)}{3}}}, 1 \leq M \leq n-1. \tag{31}$$

The cardinality of each orbit is the familiar combinatorial expression for the number of permutations of a set of M elements which can repeat

$$|O_{\mathbf{n}}| = \frac{M!}{g_1 g_2 \cdots g_M} \tag{32}$$

where g_i is the number of times the ith element repeats. This coarse-graining method partitions the sample space of a boson sampler for a given M and n into disjoint subsets. What is important to analyze now is how many subsets there are for a given M and n as that will determine how many times we must sample from the boson sampler. If we use the approximation of $n \in \Theta(M)$ with $n \approx 2M$ we have the following upper bound on the number of orbits

$$\frac{5.44}{M} e^{\pi\sqrt{\frac{2M}{3}}} \in \mathcal{O}(\frac{e^{\pi\sqrt{\frac{2M}{3}}}}{M}). \tag{33}$$

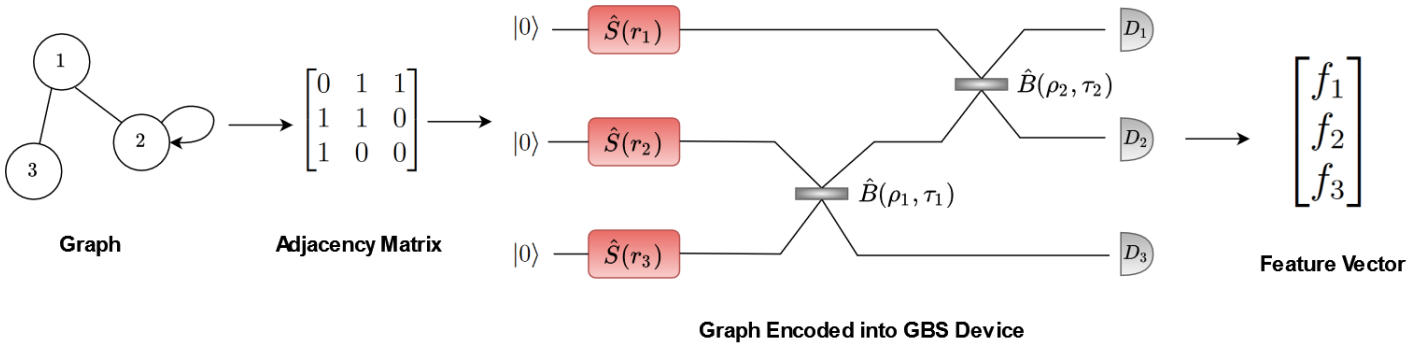


Figure 9: Diagram of the β_s^n feature map. The graph is converted to an adjacency matrix then encoded into the GBS device. Afterwards the device is used to generate s samples which will be used to create the feature vector. Entries of the feature vector are the probabilities that the corresponding detector detects no photons.

Since this implies the number of orbits is in $\mathcal{O}(e^{o(M)})$ it is only sub-exponential in M so the orbits method only provides a moderate coarse-graining of the sample space. Lastly in [5, 18] a second coarse-graining strategy was presented that builds off the previous one and groups orbits into meta-orbits. A meta-orbit is characterized by a total photon number n, and Δ_s which is defined as

$$\Delta_s = \{\mathbf{n} : \sum_i n_i = n \ \wedge \ \forall i : \ n_i \leq s \ \}. \tag{34}$$

Therefore a meta-orbit consists of all outcomes where total photon number is equal to n, where no detector counts more than s photons. This strategy in combination with the previous one coarse-grains the orbits. One can analyze how much this coarse-grains the orbits again by analyzing its mathematical analog which in this case is the number of integer partition of n with at most M parts each of which is less than or equal to s and its generating function which is a Gaussian binomial coefficient is presented in [18]

$$\mathcal{M}(M, s) = \binom{s+M}{M}_x = \prod_{j=1}^{M} \frac{1 - x^{s+M+1-j}}{1 - x^j}. \tag{35}$$

The number of such partitions for a given M and s is given by the coefficient of x^n of the above Gaussian binomial coefficient expanded around $x = 0$. In [18] it is claimed, although not proven explicitly, that this coarse-graining strategy partitions all orbits into a polynomial number of subsets in n. The important caveat with this method is that the subsets are only polynomial if we have n as constant.

3 GBS for Graph Classification

In the following section as in the previous ones we will define the sample complexity of an algorithm by how many times it requires us to sample from the GBS. We will define space complexity in terms of the length or number of dimensions of the feature vectors constructed by a feature map.

3.1 Classical Graph Kernels

Three classical graph kernels were used as a benchmark for the GBS feature maps. The subgraph matching kernel (SM) with time complexity $\mathcal{O}(kN^{k+1})$ where N is the number of vertices and k the size of the subgraphs being considered [19], the graphlet sampling kernel (GS) with worst case time complexity $\mathcal{O}(N^k)$ which can be optimized to $\mathcal{O}(Nd^{k-1})$ for graphs of bounded degree with the restriction that $k \in \{3, 4, 5\}$, where k is the graphlet size and d is the maximum degree of the graph [16], and the random walk kernel (RW) with time complexity $\mathcal{O}(N^3)$ [20]. The accuracies of all three classical kernels from [5] are shown in table 3.

3.2 The Current GBS Feature Maps

3.2.1 The Xanadu Feature Map

The proposed GBS feature map in [5] maps a graph G, which is encoded into a GBS, to a feature vector $\varphi : G \to \mathbf{f} = (f_1, f_2, ..., f_D) \in \mathbb{R}^D$. The entries f_i of $\mathbf{f}$ consists of the detection probabilities of either orbits or meta-orbits

$$f_i = p(O^i_{\mathbf{n}}), \text{ or } f_i = p(\mathcal{M}^i_{n,\Delta_s}) \tag{36}$$

where the probability of an orbit $O_{\mathbf{n}}$ or meta-orbit $\mathcal{M}_{n,\Delta_s}$ is simply the sum of the probability of the detection events in that orbit or event respectively. We will call these the α and α^+ maps where α is the feature map with respect to orbits and α^+ is the feature map with respect to meta-orbits. This feature map was shown to outperform some classical graph kernels such as the graphlet sampling, random walk, and subgraph matching kernels when classifying graphs from graph data sets commonly used to benchmark new graph kernels such as MUTAG. This was demonstrated using classical simulations of GBS with PNRDs. However as shown earlier the number of orbits, and thus the size of the sample space, grows sub-exponentially with the number of vertices of a graph and therefore even with a scalable fault tolerant quantum computer, which is many years away from development, would require

at least sub-exponential amounts of sampling to accurately approximate the probabilities of the orbits that make up the entries of the feature vector. Current day PNRDs can measure about 10^5 samples per second [5], using equation 29 with $\delta = 0.05$, $\epsilon = 0.05$, $D \approx e^{\sqrt{M}}$ and assuming GBS with no photon loss we must sample about 1.22×10^7 times for $M = 100$ modes. The current day detectors can accomplish this in about 122 seconds. One might argue that this means the method could be tractable for graphs on the order of $\sim$100 vertices. However the other drawback for the scalability of these feature vectors is their length. Since the length of the feature vector is directly proportional to the number of orbits this means the length of the feature vectors also grows sub-exponentially with the number of graph vertices and thus would result in a space complexity of $\mathcal{O}(e^{\sqrt{M}})$.

3.2.2 GBS Feature Map with Linear Space Complexity

We define the feature map to be: $\varphi : G \to \mathbf{f} = (f_1, f_2, ..., f_M) \in \mathbb{R}^M$. Where the entries of the feature vector f_i correspond to the probability that the ith detector detects no photons. We call this the β_s^n map. Where n denotes how many photons the simulation was run with and s denotes how many times we sampled from the simulated GBS. The length of these feature vectors scales linearly with the number of modes of the boson sampler and thus linearly with the number of vertices of the graphs.

Table 1: Graph data set statistics before and after prepossessing. Numbers on the left side of the colon indicate value before preprocessing. More detailed descriptions of these data sets can be found in appendix B of [5].

Data set	# of graphs	# of classes	avg. # of vertices	avg. # of edges
MUTAG	188 : 123	2 : 2	17.93 : 17.93	19.79 : 19.79
PTC_FM	349 : 284	2 : 2	14.11 : 14.11	14.48 : 14.48
ENZYMES	600 : 217	6 : 6	32.63 : 31.85	62.14 : 61.09
PROTEINS	1113 : 534	2 : 2	39.06 : 37.39	72.82 : 69.29
ER_MD	446 : 357	2 : 2	21.33 : 21.32	234.85 : 234.84
COX2_MD	303 : 118	2 : 2	26.28 : 26.27	335.12 : 335.12
IMDB_BINARY	1000 : 806	2 : 2	19.77 : 19.77	96.53 : 96.53
BZRD_MD	306 : 257	2 : 2	21.30 : 21.30	225.06 : 225.05

3.2.3 Numerical Simulation

Due to the lack of sufficiently fault tolerant quantum hardware we resort to simulating GBS classically. We use threshold detectors as opposed to PNRDs in our simulation of the GBS when sampling to save significant time on computation. This still retains the quantum

advantage as the probability of a detection outcome $\mathbf{n} = (n_1, n_2, ..., n_M)$ for Gaussian boson sampling with threshold detectors is given by

$$p(\mathbf{n}) = \frac{\mathrm{Tor}(\mathbf{O_n})}{\sqrt{\det(\boldsymbol{\Sigma})}} \tag{37}$$

where

$$\mathbf{O_n} = \mathbb{I} - (\boldsymbol{\Sigma}^{-\mathbf{1}})_{\mathbf{n}}, \tag{38}$$

$$\mathrm{Tor}(A) = \sum_{Z \in P([n])} (-1)^{|Z|} \frac{1}{\sqrt{\det(\mathbb{I} - A_Z)}} \tag{39}$$

and Tor() is the *Torontonian* of a matrix $A \in \mathbb{C}^{2n \times 2n}$. $\boldsymbol{\Sigma}$ is the covariance matrix of the complex amplitude $\boldsymbol{\alpha}$. $P([n])$ is the power set, the set of all possible subsets, of the set $[n] = \{1, 2, ..., n\}$. Since the calculation requires summing over the elements in the power set, which has cardinality of 2^n, it also requires the calculation of 2^n determinants. Given that the fastest determinant algorithms have complexity $\mathcal{O}(n^3)$ for $n \times n$ matrices it is easy to see the complexity of calculating the Torontonian is $\mathcal{O}(n^3 2^n)$ which is the same as the complexity of the fastest Hafnian algorithms [21] thus making it as hard as GBS with PNRDs to classically simulate.

3.2.4 Preprocessing of Data Sets

To make our results more comparable to [5] we decided to only use graphs from each data set that have fewer than 26 and more than 5 vertices. We then follow the convention in [22] and remove isolated vertices from each graph and extracted the largest connected component of the graph if the graph wasn't fully connected. Next, as was done in [23], for each data set we pad the adjacency matrix of all graphs with zeros on the bottom and right sides to make all adjacency matrices the same dimensions as the largest matrix in the filtered data set. For each data set this meant padding each matrix to make its dimensions 25×25. We fetched the data-sets using Python's GraKel library and prepossessed them using the NetworkX library [24, 25].

3.2.5 Model Selection

We test our classification method on 8 of the 11 data sets used in [5] which are shown in table 1. We use a SVM with an rbf kernel and random forest classifier as our two supervised machine learning classifiers with the GBS generated feature vectors as our input set along with the class label set from the data set as their input. We obtain the accuracies by running a nested 10-fold cross-validation for the SVM and nested 5-fold cross-validation for the random

Table 2: Test accuracies of the β_s^n is the feature map whose feature vector entries are the probabilities of the detectors detecting no photons. SVM and RF indicate the accuracies of classification using the SVM and random forest classifiers respectively.

Data set	β_{500}^5 SVM	β_{500}^5 RF	β_{500}^4 SVM	β_{500}^4 RF	β_{1000}^3 SVM	β_{1000}^3 RF
MUTAG	79.93	82.14	79.93	82.14	79.93	**82.66**
PTC_FM	63.74	63.05	**64.43**	54.21	63.71	59.86
ENZYMES	30.45	32.68	33.18	**35.94**	31.36	35.95
PROTEINS	65.34	**68.14**	67.78	67.59	66.11	67.96
ER_MD	65.57	66.12	65.55	65.28	66.42	**68.65**
COX2_MD	**53.41**	47.35	53.40	45.79	52.57	45.79
IMDB_BINARY	60.68	**63.52**	59.93	62.03	59.18	59.55
BZRD_MD	62.96	60.70	63.00	58.75	**63.36**	59.51

Table 3: Test accuracies of different features maps on benchmark data sets. α and α^+ are the Xanadu feature map with 0 displacement whose feature vector dimensions are orbit and meta-orbit probabilities respectively. The accuracies for the three classical graph kernels used for comparison, the graphlet sampling (GS), random walk (RW), and subgraph matching (SM) kernels are also shown. **Runtime* > 20 *days*

Data set	α	α^+	GS	RW	SM
MUTAG	**86.41**	85.64	81.08	83.02	83.14
PTC_FM	53.84	59.14	**59.48**	51.97	54.92
ENZYMES	22.29	25.72	35.87	21.13	**36.70**
PROTEINS	**66.88**	65.73	65.91	56.27	63.03
ER_MD	70.36	**71.01**	65.65	68.75	68.21
COX2_MD	44.98	57.84	55.04	57.72	**66.94**
IMDB_BINARY	64.09	68.14	**68.37**	66.38	out of time*
BZRD_MD	**62.73**	62.01	60.60	49.88	61.90

forest classifier. For the SVM the inner 10-fold cross-validation finds the best value for the C hyper-parameter of the SVM, which controls the penalty on misclassifications, through a grid search between the values $[0.1, 40]$ with a step size of 0.1. The model with best performance is then used in the outer 10-fold cross-validation loop to get the accuracies on the test set. The same method is used for the random forest classifier except with 5-fold as opposed to 10-fold cross-validation. The relevant hyper-parameters used in the grid search for the random forest classifier were maximum depth of the trees and the number of trees with $[1, 10, 50, 100, 250, 750, 1000]$ being the range of values used in the grid search for both parameters. The results of the α and β_s^n feature maps are shown in table 3 and 2 respectively.

3.2.6 Sample Complexity

The sample space of the distribution we are sampling from to create the β_s^n feature vectors is the set of all possible detection events of the M-mode thresholded GBS. Therefore it is important to analyze the size of the sample space of the GBS with threshold detectors as that will determine how many times we must sample from the device. First let us examine the limiting cases that $n = 1$ and $n = M$, clearly the number of outcomes is in $\mathcal{O}(M)$ and $\mathcal{O}(2^M)$ respectively. Since the n_i's for threshold detectors can be either 0 or 1 we can think of the detection outcomes as binary strings of length M with at most n ones. The number of binary strings of length M with exactly n ones is $\binom{M}{n}$. So the cardinality of the sample space, the number of binary strings of length M with at most n ones, is given by

$$|\Omega| = \sum_{i=0}^{n} \binom{M}{i} \tag{40}$$

If we assume n grows linearly with M, $n = M$, we can show this function grows like 2^M using the binomial expansion

$$2^M = (1+1)^M = \sum_{i=0}^{M} \binom{M}{i} 1^{M-i} 1^i = \sum_{i=0}^{M} \binom{M}{i} \tag{41}$$

In practice we can usually have $n \leq M$ so using the the approximation $n \approx \lfloor \sqrt{M} \rfloor$ we have

$$|\Omega| = \sum_{i=0}^{\lfloor \sqrt{M} \rfloor} \binom{M}{i} < \sum_{i=0}^{M} \binom{M}{i} = 2^M \tag{42}$$

which means the function is in $o(2^M)$ meaning it grows strictly less than 2^M. However we can also show that under the same scaling approximation the function is in $\omega(2^{\lfloor \sqrt{M} \rfloor})$,

$$2^{\lfloor \sqrt{M} \rfloor} = (1+1)^{\lfloor \sqrt{M} \rfloor} = \sum_{i=0}^{\lfloor \sqrt{M} \rfloor} \binom{\lfloor \sqrt{M} \rfloor}{i} < \sum_{i=0}^{\lfloor \sqrt{M} \rfloor} \binom{M}{i}. \tag{43}$$

Therefore the sample complexity of this feature map is somewhere between exponential and sub-exponential in M.

3.3 GBS Feature Map with Linear Space & Sample Complexity

While the results from the β_s^n feature map are promising we would still like to reduce the sample complexity to be polynomial in the graph size as opposed to sub-exponential. To

this end we decide to coarse-grain the sample space of the thresholded GBS by grouping together detection events with exactly $\tilde{n}$ ones, where $\tilde{n} \in (1, ..., n)$. We call these groups *binary orbits*. For example for a 4-mode boson sampler the detection events $(1, 1, 0, 1)$ and $(0, 1, 1, 1)$ belong to the same binary orbit since they both have exactly 3 detector 'clicks' or ones. This would partition the set of all events in the sample space into a linear number of disjoint subsets in n. The new feature map, which we call the γ_s^n map, would then be $\varphi : G \to \mathbf{f} = (f_1, f_2, ..., f_M) \in \mathbb{R}^M$ where the entries f_i would correspond to the probability of detecting a event with exactly i ones or 'clicks'. Since the number of elements in the new coarse-grained sample space is polynomial in the number of photons, which we can tune to grow linearly or sub-linearly with the number of modes of the GBS, we would only have to sample a polynomial number of times from the boson sampler to construct a feature vector for a graph of M vertices. The results of the γ_s^n feature map are shown in table 4. We can see that for most data sets the γ_s^n map is competitive with the classical kernels.

Table 4: Table for the accuracies of the γ_s^n map. Where s is the number of times each graph was sampled from after being encoded into the GBS and n is the number of photons sent into the GBS. Columns with RF and SVM denote accuracies with the random forest classifier and SVM respectively.

Data set	γ^3_{100} SVM	γ^3_{100} RF	γ^4_{100} SVM	γ^4_{100} RF	γ^5_{100} SVM	γ^5_{100} RF	γ^8_{100} SVM	γ^8_{100} RF
MUTAG	67.61	65.34	70.88	61.42	68.17	75.45	**80.49**	76.53
PTC_FM	61.28	52.46	60.94	57.37	60.24	50.70	**61.60**	53.52
ENZYMES	15.12	21.19	**21.19**	20.30	14.28	18.91	16.51	17.10
PROTEINS	59.72	61.41	62.71	61.96	64.25	64.23	64.95	**64.96**
ER_MD	60.79	52.95	**66.40**	58.55	64.41	62.73	64.67	64.45
COX2_MD	43.18	51.55	**58.40**	56.08	45.00	51.48	38.10	48.29
IMDB_BINARY	60.91	61.91	62.28	59.55	62.64	62.40	65.61	**66.25**
BZRD_MD	50.92	57.18	47.06	46.72	54.00	54.17	**62.70**	59.53

4 Summary and Open Problems

In this work we analyze the time and sample complexity of a GBS based quantum algorithm for the GraphIsomorphism problem. We found that the algorithm would require $\omega(\sqrt{M}^{\sqrt{M}})$ sample complexity in the worst case and $\mathcal{O}(e^{\sqrt{M}})$ sample complexity in the best case, where M are the number of vertices in the graph. The time complexity of the GraphIsomorphism algorithm would be $\mathcal{O}(M^{2.8})$ as encoding the graph into a GBS device requires the multiplication and inversion of a $2M \times 2M$ matrix and the fastest known algorithm for matrix multiplication is Strassen's algorithm with complexity $\mathcal{O}(n^{2.8})$. We also analyze the

time, space and sample complexity of the GBS based feature maps introduced in [5] and [23] for the machine learning task of graph classification. We find that they both have time complexity $\mathcal{O}(M^{2.8})$ as they both require matrix multiplication to encode the graphs into a GBS. In the case of the feature map in [23] the space complexity is $\mathcal{O}(M)$ but the sample complexity is $\omega(2^{\sqrt{M}})$. The two feature maps introduce in [5] are the α and α^+ feature maps. In the case of the α feature map both the sample and space complexity are $\mathcal{O}(e^{\sqrt{M}})$. In the case of the α^+ feature map the sample and space complexity are $\mathcal{O}(M)$ if the photon number, n, is constant [18]. We then propose a feature map for graphs of M vertices with sample complexity that is $\mathcal{O}(n)$, i.e., linear in the number of photons, n, sent into the GBS, time complexity that is $\mathcal{O}(M^{2.8})$, i.e, sub-cubic in the number of modes and space complexity that is $\mathcal{O}(M)$, i.e., linear in the number of modes M of the GBS. We can tune the number of photons to scale linearly or sub-linearly with the number of modes of the device so the sample complexity is also linear in the number of modes of the GBS which is also the number of vertices of the graph. This is an improvement over the graphlet sampling (for non-constant graphlet size k), random walk, and subgraph matching kernels which were the classical graph kernels used to benchmark the performance of the GBS feature map in [5]. What is most interesting is that the γ_s^n feature map gives us comparable results to the α^+ feature map but the γ_s^n map used threshold detectors instead of PNRDs. Thus we have a scalable and more simple quantum graph kernel that is competitive with the current state-of-the-art classical graph kernels. A number of questions remain open for investigation such as:

1. To create the α, α^+, and γ_s^n feature vectors we must sample from a coarse-grained version of the probability distribution of GBS detection patterns. In the case of the α map we coarse-grain the sample space into orbits, meta-orbits for α^+ and binary orbits for γ_s^n. But are any of these probability distributions classically easy to sample from? If they are then the quantum advantage is lost. However in the case of γ_s^n we would still have a new graph kernel that is competitive with the current classical graph kernels. If they aren't easy to sample from classically then we retain the quantum advantage.

2. Would adding a uniform displacement to all the modes of the GBS have a drastic increase in the accuracy of the feature map, particularly in the case of the ENZYMES data set, as it did in [5]?

3. How low can we keep the photon number while still getting good accuracies? Could we get good accuracies if we have the number of photons scale sub-linearly with the number of vertices say scaling as $\log(M)$ or $\sqrt{M}$?

4. Can the γ_s^n feature map consistently outperform the classical graph kernels when run with larger values of n?

5. In this paper we consider only a lossless GBS. How does photon loss in the interferometer impact the accuracies?

6. Could some insight be gained from doing a principal component analysis on the feature vectors from the γ_s^n map? Such as which binary orbit probabilities explain the most variance in the data?

Acknowledgements

I would like to extend my sincere thanks to Professor Olivier Pfister and Dr. Miller Eaton for advising me and helping me become a better researcher. I would also like to thank Professor Israel Klich for evaluating the thesis for the distinguished major program and Dr.'s Leonid Petrov, Mohammad Mahmoody, Maria Schuld, Kamil Bradler, Scott Aaronson, Juraj Foldes and Andrew Blance for useful discussions. I am also grateful to Professor's James Rolf and Diana Morris for giving advice on some of the math in the document and for being exceptional math instructors. I also thank my friend Zaki Panjsheeri for providing useful comments.

Appendices

A Factorial Identity

Lemma 4. $(n+c)! = [\prod_{i=1}^{c}(n+i)]n!$ *for* $c \geq 0$

Proof. Clearly

$$(n+2)! = (n+2)(n+1)(n)! = \big[\prod_{i=1}^{2}(n+i)\big]n!$$

And in general from the definition of $n!$:

$$(n+c)! = \big[\prod_{i=1}^{c}(n+i)\big]n!$$

□

B Tight Bounds for $\lfloor x \rfloor$ and $\lceil x \rceil$

Lemma 5. $\lfloor x \rfloor \in \Theta(x)$

Proof.

$$\begin{aligned} \lfloor x \rfloor \leq cx \text{ for } c = 1 \ \& \ x > 2 \\ \therefore \quad \lfloor x \rfloor \in \mathcal{O}(x) \\ \lfloor x \rfloor \geq cx \text{ for } c = \frac{1}{2} \ \& \ x > 2 \\ \therefore \quad \lfloor x \rfloor \in \Omega(x) \\ \therefore \quad \lfloor x \rfloor \in \Omega(x) \cap \mathcal{O}(x) = \Theta(x) \end{aligned}$$

□

Lemma 6. $\lceil x \rceil \in \Theta(x)$

Proof.

$$\begin{aligned} \lceil x \rceil \leq cx \text{ for } c = 2 \ \& \ x > 2 \\ \therefore \quad \lceil x \rceil \in \mathcal{O}(x) \\ \lceil x \rceil \geq cx \text{ for } c = 1 \ \& \ x > 2 \\ \therefore \quad \lceil x \rceil \in \Omega(x) \\ \therefore \quad \lceil x \rceil \in \Omega(x) \cap \mathcal{O}(x) = \Theta(x) \end{aligned}$$

□

C Computational Complexity Theory

C.1 Basic Theory & Notation

In computer science the Turing machine is the universal model of computation that is as powerful as all other classical models of computation. It consists of a head that reads in an infinite tape that is divided into cells as input. The head reads the symbol on each cell in the tape and after reading a symbol transitions into a different state based on some transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\text{L,R}\}$. Where Q is the set of possible states of the machine, Γ is the tape alphabet and L and R indicate whether the head moves to the left or the right cell of the current one. A non-deterministic Turing machine has a transition function of the form

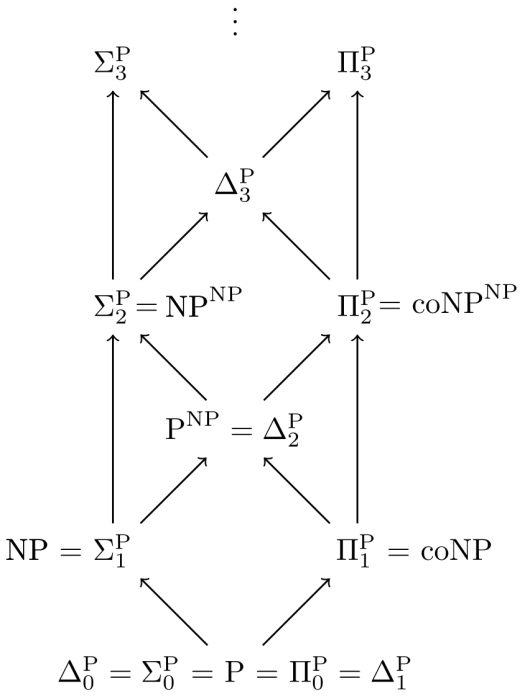


Figure 10: Diagram of the polynomial hierarchy PH. The arrows indicate that the class is contained in the class pointed to.

$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\mathrm{L,R}\})$. This means that for each cell the head reads the transition function can allow the machine to transition into more than one state depending on which of the possible states will lead to an accepting state. A probabilistic Turing machine has two transition functions δ_0 and δ_1 and at each step of its execution applies either transition function with probability $\frac{1}{2}$.

C.2 Complexity Classes

Computational problems such as the ones we will discuss later are classified by being placed in certain complexity classes. The most well known of these complexity classes is P, which is the class of decision problems solvable by a deterministic Turing machine in polynomial time, and NP which is the class of decision problems solvable by a non-deterministic Turing machine in polynomial time or equivalently the class of decision problems whose solutions can be verified in polynomial time by a deterministic Turing machine. A decision problem is NP-hard if there is a polynomial time way to map an instance of any problem in the complexity class NP to an instance of that problem. A decision problem is NP-complete if it's both NP-hard and in the class NP. For a computational problem an oracle $\mathcal{O}$ is a black box that returns a solution for any instance of that problem. Complexity classes can be defined with respect to an oracle as well. For a complexity class an oracle can return a solution for any

instance of a problem in that class. For example a problem is in the complexity class P^{NP} if it can be solved by a deterministic Turing machine which has access to an oracle for a NP-complete problem. The polynomial hierarchy PH, depicted in figure 10, is a multi-leveled tower of complexity classes. It is inductively defined through the use of oracles where level 1 is P, level 2 contains NP, level 3 contains $\mathsf{NP}^{\mathsf{NP}}$, and so on. Mathematically it can be written as

$$\text{Initialize } \mathsf{P} = \Pi_0^{\mathsf{P}} = \Sigma_0^{\mathsf{P}} = \Delta_0^{\mathsf{P}}. \text{ Define:} \tag{44}$$

$$\Pi_{i+1}^{\mathsf{P}} = \mathsf{coNP}^{\Sigma_i^{\mathsf{P}}} \tag{45}$$

$$\Sigma_{i+1}^{\mathsf{P}} = \mathsf{NP}^{\Sigma_i^{\mathsf{P}}} \tag{46}$$

$$\Delta_{i+1}^{\mathsf{P}} = \mathsf{P}^{\Sigma_i^{\mathsf{P}}} \tag{47}$$

$$\mathsf{PH} = \bigcup_{i=0}^{\infty} \Pi_i^{\mathsf{P}} \cup \Sigma_i^{\mathsf{P}} \cup \Delta_i^{\mathsf{P}}. \tag{48}$$

Each level is contained in the level above it and if two adjacent levels i and $i+1$ are equal then they are equal to all the above levels. This is what is called the collapse of the polynomial hierarchy to the ith level. It is widely believed that the polynomial hierarchy does not collapse to any level. In fact showing that a conjecture would imply the collapse of PH is a common way of giving evidence for the conjecture's unlikeliness. This is also why the question of if $\mathsf{P} = \mathsf{NP}$ is important, because it would imply a collapse of PH to the first level which would mean that all problems in PH are solvable in polynomial time by a deterministic Turing machine. The complexity class of interest for the sampling problem relevant later is $\#\mathsf{P}$, which is the set of counting problems that count how many solutions there are to a corresponding problem in NP. Its equivalent complexity class of decision problems is $\mathsf{P}^{\#\mathsf{P}}$ which is the class of decision problems solvable in polynomial time by a deterministic Turing machine with access to an oracle for a $\#\mathsf{P}$-complete problem. From Toda's theorem we have $\mathsf{PH} \subset \mathsf{P}^{\#\mathsf{P}}$ or equivalently $\#\mathsf{P}$ problems are as hard as any problems in the polynomial hierarchy. Lastly BPP is the class of decision problems solvable in polynomial time by a probabilistic Turing machine with an error probability of $\frac{1}{3}$. [26, 27]. BPP is contained in the second level of PH and $\mathsf{BPP}^{\mathsf{NP}}$ in the third level [28].

References

[1] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," Nature **574**, 505 (2019).

[2] H.-S. Zhong *et al.*, "Quantum computational advantage using photons," Science **370**, 1460 (2020).

[3] K. Brádler, P.-L. Dallaire-Demers, P. Rebentrost, D. Su, and C. Weedbrook, "Gaussian boson sampling for perfect matchings of arbitrary graphs," Physical Review A **98**, 032310 (2018).

[4] J. M. Arrazola and T. R. Bromley, "Using Gaussian boson sampling to find dense subgraphs," Physical review letters **121**, 030503 (2018).

[5] M. Schuld, K. Brádler, R. Israel, D. Su, and B. Gupt, "Measuring the similarity of graphs with a Gaussian boson sampler," Physical Review A **101**, 032314 (2020).

[6] K. Brádler, S. Friedland, J. Izaac, N. Killoran, and D. Su, "Graph isomorphism and Gaussian boson sampling," Special Matrices **9**, 166 (2021).

[7] L. G. Valiant, "The complexity of computing the permanent," Theoretical computer science **8**, 189 (1979).

[8] P. Mills, R. Rundle, J. Samson, S. J. Devitt, T. Tilma, V. Dwyer, and M. J. Everitt, "Quantum invariants and the graph isomorphism problem," Physical Review A **100**, 052317 (2019).

[9] D. Aharonov, W. Van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, "Adiabatic quantum computation is equivalent to standard quantum computation," SIAM review **50**, 755 (2008).

[10] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," Applied Network Science **5**, 1 (2020).

[11] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," in Proceedings of the forty-third annual ACM symposium on Theory of computing, pp. 333–342 (2011).

[12] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, "Gaussian boson sampling," Physical review letters **119**, 170501 (2017).

[13] R. Kruse, C. S. Hamilton, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, "Detailed study of Gaussian boson sampling," Physical Review A **100**, 032326 (2019).

[14] T. R. Bromley, J. M. Arrazola, S. Jahangiri, J. Izaac, N. Quesada, A. D. Gran, M. Schuld, J. Swinarton, Z. Zabaneh, and N. Killoran, "Applications of near-term photonic

quantum computers: software and algorithms," Quantum Science and Technology **5**, 034010 (2020).

[15] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, "Optimal design for universal multiport interferometers," Optica **3**, 1460 (2016).

[16] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in Artificial intelligence and statistics, pp. 488–495 (2009).

[17] A. Y. Oruç, "On number of partitions of an integer into a fixed number of positive integers," Journal of Number Theory **159**, 355 (2016).

[18] K. Bradler, R. Israel, M. Schuld, and D. Su, "A duality at the heart of Gaussian boson sampling," arXiv preprint arXiv:1910.04022 (2019).

[19] N. Kriege and P. Mutzel, "Subgraph matching kernels for attributed graphs," arXiv preprint arXiv:1206.6483 (2012).

[20] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," Journal of Machine Learning Research **11**, 1201 (2010).

[21] N. Quesada, J. M. Arrazola, and N. Killoran, "Gaussian boson sampling using threshold detectors," Physical Review A **98**, 062322 (2018).

[22] N. de Lara and E. Pineau, "A simple baseline algorithm for graph classification," arXiv preprint arXiv:1810.09155 (2018).

[23] A. Blance and M. Spannowsky, "Unsupervised event classification with graphs on classical and photonic quantum computers," Journal of High Energy Physics **2021**, 1 (2021).

[24] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and M. Vazirgiannis, "GraKeL: A Graph Kernel Library in Python.," J. Mach. Learn. Res. **21**, 1 (2020).

[25] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using NetworkX," Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008) .

[26] S. Arora and B. Barak, Computational complexity: a modern approach (Cambridge University Press, 2009).

[27] M. Sipser, "Introduction to the Theory of Computation," ACM Sigact News **27**, 27 (1996).

[28] C. Lautemann, "BPP and the polynomial hierarchy," Information Processing Letters **17**, 215 (1983).