Using Quantum Algorithms to Find the Minimal Faithful Dimension for P-Groups

William Kennedy
Supervisor: Hadi Salmasian

May 5, 2024

Abstract

The purpose of this paper is to explore two quantum algorithms for computing the optimal value for a function, the motivation for this project came from wanting to find the minamal faithful dimension of G group over a field K is defined to be the smallest integer n such that G embeds into $\mathrm{GL}_n(K)$. This paper explores the minimization of the minimal faithful dimension [**1**] for a p-group of the form $\mathscr{G}_q :=\exp(\mathfrak{g} \otimes_{\mathbb{Z}} \mathbb{F}_q)$ associated to $\mathfrak{g}_q = g \otimes_{\mathbb{Z}} \mathbb{F}_q$ in the Lazard Correspondence, where $\mathfrak{g}$ is a nilpotent $\mathbb{Z}$-Lie algebra which is finitely generated on an abelian group. This paper attempts to overcome this hard computational problem by using tools and properties of quantum mechanicsk. Two proposed algorithms are given to solve this problem: First, a framework for a Variational Quantum Algorithm is provided to optimize the problem through a hybrid approach that utilizes both classical optimization and quantum optimization. The main benefit of this approach is VQA's can be used on Noisy Intermediate Scale Quantum(NISQ) Computers, as of writing this paper we are in the NISQ era and the hope is we can apply this approach in the near term. Second, the problem is reformulated into a rank optimization problem which is solved by using a Quantum Walk to traverse a backtracking with vertices that act as a partial assignment to a SAT problem.

1 Introduction

This paper explores quantum optimization algorithms that can be used to calculate the minimal faithful dimension for representations p-groups associated via the Lazard correspondance $\mathbb{Z}$-Lie algebra such as the p-group $\mathscr{G}_q :=\exp(\mathfrak{g} \otimes_{\mathbb{Z}} \mathbb{F}_q)$. Two approaches are researched in this paper, one of which can be applied on near term quantum computers and another that uses Grover's search algorithm:

1. Variational Quantum Algorithm uses classical and quantum processes to optimize a given function. A two stage algorithm where in the first stage a function whose output is the parameter of a parameterized quantum circuit is optimal using some classical optimization strategy. The second uses a paratemeterized quantum circuit to find an optimal solution for the target function.
2. A Quantum Walk is used to traverse a rooted backtracking tree whose internal vertices are mapped to partial solutions of a constraint satisfaction problem. The framework takes an arbitrary function $F : \mathcal{V} \to \mathbb{Z}$ and determines if for $v \in \mathcal{V}$, whose coordinates are mapped to literal values for a constraint satisfaction problem, and if $F(v)$ is minimized, that is for some $a, b \in \mathbb{Z}$ $a \leq F(v) \leq b$, such as determining the rank of a commutator matrix for a graded Lie algebra.

The paper is divided into the following sections:

- Section 2 gives the defines the elementary structures and concepts needed to understand Lie algebras and the minimal faithful dimension problem.
- Section 3 discusses variational quantum algorithms, here we define the guiding function and the procedures required to optimize the function in the general context of finding the optimal parameter for a parameterized quantum circuit. Then an example is defined where the guiding function is described in terms of the minimal faithful dimension problem and how convex optimization can be used to find an approximate solution for the parameterized quantum circuit.
- Section 4 explores quantum walks on rooted backtracking trees whose vertices correspond to partial and complete solutions of constraint satisfaction problems. First we explore the concepts and algorithms defined in [**3**] and [**4**], then we define an algorithm that uses these ideas to find the optimal value of a function $F : \mathcal{V} \to \mathbb{Z}$ where $mathcalV$ is a vector space.
- Section 5 explores applying the concepts and algorithms in section 4 to finding the minimal faithful dimension. An example based on elliptic curves defined in [**1**] is used to describe the procedure of using a quantum walk to find the minimal faithful dimension.
- Section 6 is the conclusion, it discusses and summarizes the topics in this paper and suggests ideas for additional research on this topic.

2 Preliminaries on Lie Algebra's

With the introduction in mind, below are the elementary definitions and statements required to understand the group theoretic minimal faithful dimension problem.

Definition 2.1 (Lie Algebra)**.** A Lie Algebra is a vector space $\mathfrak{g}$ over w a field $\mathbb{F}$ with a binary operation $[\cdot, \cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ called a Lie bracket where for all $a, b, c \in \mathbb{F}$ and for all $x, y, z \in \mathfrak{g}$ the Lie bracket satisfies:

- Bilinearity,
  $[ax + by, z] = a[x, z] + b[y, z]$

  $[x, by + cz] = b[x, y] + c[x, z]$
- Antisymmetry Property,
  $[x, x] = 0$
- Jacobi Identity,
  $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$
- Anticommutativity, this property is a result of the Bilinearity and Alternating property,
  $[x, y] = -[y, x]$

The $\mathbb{Z}$-Lie Algebra described is a $\mathbb{Z}$-module that satisfies the contraints of a Lie algebra, meaning its underlying structure is not a vector space over a field but a module over the ring $\mathbb{Z}$.

The following definition of a commutator matrix is integral to Lie algebra's and to computing the minimal faithful dimension, it is the matrix representation of the Lie bracket. If $\{X_1, \cdots, X_n\}$ are the basis elements of a Lie algebra then $[X_i, X_j]$ is a matrix whose entries are determined by the Lie bracket operation of $[X_i, X_j]$ which measures the commutativity of the basis elements X_i and X_j.

Definition 2.2 (Commutator Matrix)**.** The commutator matrix of $\mathfrak{g}$ is a skew-symmetric matrix of linear forms defined by:

$$F_{\mathfrak{g}}(T_1, \cdots, T_n) := [\Lambda_{ij}(T_1, \cdots, T_m)]_{1 \leq i,j \leq n} \in M_n(\mathbb{Z}[T_1, \cdots, T_m]) \tag{1}$$

for $T_1, \cdots, T_m \in \mathbb{F}_q$

The minimal faithful dimension of a group G over a field K is the smallest $n \in \mathbb{Z}$ in which G is embedded into $\mathrm{GL}_n(K)$. There are many applications that have an interest in computing the minimal faithful dimension of a group over a generic field.

Definition 2.3 (Minimal Faithful Dimension)**.** Given a finite group G and a field K, the faithful dimension of G over K is defined to be the smallest integer n such that G embeds into $\mathrm{GL}_n(K)$

Theorem 2.1. *Let $\mathfrak{g}$ be a nilpotent $\mathbb{Z}$-Lie Algebra of nilpotency class c which is finitely generated as an abelian group. If $p > max\{c, C_1, C_2, C_3\}$, then*

$$m_{faithful}(\mathscr{G}_q) = min\left\{ \sum_{l=1}^{l_1} fq^{\frac{rk_{\mathbb{F}_q}(F_{\mathfrak{g}}(x_{l1}, \cdots, x_l m))}{2}} : \begin{pmatrix} x_{11} & \cdots & x_{1l_1} \\ \vdots & \ddots & \vdots \\ x_{l_1 1} & \cdots & x_{l_1 l_1} \end{pmatrix} \in \mathrm{GL}_{l_1}(\mathbb{F}_q) \right\} + fl_2 \tag{2}$$

where $m := rk_{\mathbb{Z}}([\mathfrak{g}, \mathfrak{g}])$, $l_1 := rk_{\mathbb{Z}}[\mathfrak{g}, \mathfrak{g}] \cap Z(g))$, and $l_2 := rk_{\mathbb{Z}}(Z(g)/Z(g) \cap [\mathfrak{g}, \mathfrak{g}])$.

3 Variational Quantum Algorithms

A Variational Quantum Algorithms (VQA) is a hybrid quantum optimization model that utilizes classical computation and quantum computation, the classical component of the algorithm reduces the circuit depth of the quantum circuit so the solutions produced are viable for current Noisy Intermediate Scale Quantum(NISQ) computers. Hence there is a lot of interest in exploring the applicability of algorithms that run on NISQ computers. With this in mind we apply the VQA framework to optimize the minimal faithful dimension problem for nilpotent Lie algebras.

As formulated in [**2**] the following optimization problem will be fit to the VQA framework and used on our minimal faithful dimension minimization task:

$$\min_{x \in \{0,1\}^n} \quad f(x) \tag{3}$$

The hybrid VQA works as alternating between two parts, a classical optimization problem and a quantum circuit. The classical optimization, in our instance, uses convex optimization to optimize a guiding function which leads to an optimal parameter for a paramterized quantum circuit which gives us a high probability of finding the solution to our problem upon measuring the circuit. We will denote $\mathcal{F}$ as the set of optimal solutions.

3.1 Preliminaries

3.1.1 Guiding Function of the Classical Part

The classical part of our VQA is described in two parts: the function that guides the optimization and the optimization method. The input for the guiding function is $\theta \in \mathbb{R}^d$ and the parameterized circuit takes $|0\rangle^{\otimes n}$. As it name describes, the purpose of the guiding function is to find the optimal $\theta^* \in \mathbb{R}^d$ for the paired

parameterized quantum circuit U. The optimizing the guiding function gives an optimal parameter for the paramterized quantum circuit which in turn allows us to maximize the probability that upon measurement of the circuit we receive an input that minimizes $f(x)$.

Definition 3.1 (Parameterized Quantum Circuit)**.** A parameterized quantum circuit is a continous function $U : \mathbb{R}^d \to M_{2^n}(\mathbb{C})$ mapping any $\theta \in \mathbb{R}^d$ to a unitary matrix $U(\theta)$.

The set of quantum states that are in a superposition of the optimal solutions of problem (1) will be denoted as

$$F_{quant} = \{\sum_{s\in\mathcal{F}} \psi_s |s\rangle : \sum_{s\in F} |\psi|^2 = 1\} \tag{4}$$

Definition 3.2 (Guiding Function)**.** Let $g : \mathbb{R}^d \to \mathbb{R}$ be a function for some $d \in \mathbb{N}$ and G be its set of minimizers. The function g is a guiding function for f with respect to parameterized quantum circuit U if g is continuous and

$$\{U(\theta) |0\rangle^{\otimes n} : \theta \in G\} \subseteq \mathcal{F}_{quant} \tag{5}$$

Thus minimizing g will minimize f. The difficulty of this problem is appropiately choosing U, without any information on $\mathcal{F}$, we need to choose a quantum circuit U such that for any optimal solution $s \in F$ such that:

$$\{U(\theta) |0\rangle^{\otimes n} : \theta \in G\} \supseteq CB_n \tag{6}$$

Where CB_n is the basis set of F, meaning the span$(CB_n) = F \supseteq \mathcal{F}_{quant}$.

A popular choice for a guiding function, and the guiding function chosen for this problem is the mean function

$$g_{mean}(\theta) = \sum_{x\in\{0,1\}^n} p_\theta(x) f(x) \tag{7}$$

where $p_\theta(x) = |\langle x| U(\theta) |0\rangle^{\otimes n}|^2$ is the probability of finding x when $U |0\rangle^{\otimes n}$ is measured. The proof of $g_{mean}(\theta)$ being a guiding function can be found in [**2**] Appendix B.

In [**2**] their approach to optimizing the guiding function is by using stochastic optimization, in short they sample a distribution of size in f's domain and aim to minimize g by equating it to the expectation of this distribution. More succintly,

$$\min_{\theta} \quad g(\theta) = \mathbb{E}[f(\xi_\theta)] \tag{8}$$

where $\xi_\theta^j \in \{0,1\}^n$ is the sampling of the discrete random variable ξ_θ. However our approach constrains the guiding function such that it is convex and convex optimization can be used to simply matters.

3.1.2 Convex Optimization

Convex optimization adheres to the idea of finding a global minimum, which is ensured to exist and is reachable in the definition of a convex function. In this paper convex optimization is implemented with Gradient Descent.

Definition 3.3 (Convex Function)**.** A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^d$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1$, $\alpha \geq$, $\beta \geq 0$ the following is satisfied

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y) \tag{9}$$

A convex optimization problem is of the following form:

$$\begin{aligned} &\text{minimize} \quad f(x) && \text{(10a)}\\ &\text{subject to} \\ &f_i(x) \leq b_i, \text{ for } i = 1, \cdots, m, && \text{(10b)}\\ &h_i(x) = 0), \text{ for } i = 1, \cdots, p && \text{(10c)} \end{aligned}$$

where f_0 is a convex function, f_i are inequality constraint functions and h_i are equality constraint functions for each $i = 1, \cdots, m$.

3.1.3 Gradient Descent

Descent methods aim to minimize a sequence $(x_n)_{n=1}$ where each iterate is of the form

$$x_{n+1} = x_n + t_n \Delta x_n \text{ for } t_n > 0 \text{ except when } x_n \text{ is optimal} \tag{11}$$

The vector $\Delta x \in \mathbb{R}^n$ is called the search direction and t_n is the step length at iteration n. In descent methods the minimizing sequence will satisfy the following except for when x_n is optimal:

$$f(x_{n+1}) < f(x_n) \tag{12}$$

Algorithm 1 General Descent Method

Require: Let $f : \mathbb{R}^d \to \mathbb{R}$ and $x \in \mathbb{R}^d$

Repeat the following:

1. Determine a descent direction for Δx.
2. Line Search. Choose a step size $t > 0$.
3. Update. $x_{n+1} := x_n + t\Delta x_n$.

Backtracking Line Search Backtracking line search is an inexact line search method, meaning that is does not guarantee the minimum possible step size in that direction is found, used in gradient descent. The step length is chosen to approximately mininimize f along the ray $\{x + t\delta x | t \geq 0\}$ and depends on the two constants α, β where $0 < \alpha < 0.5$, $0 < \beta < 1$. The general algorithm for backtracking line search is:

Algorithm 2 Backtracking Line Search

Require: Given a descent direction Δx for f where $x \in \text{dom}(f)$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.

t:=1

while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$ **do** $t := \beta t$

end while

The reason it is called a backtracking line search is because it starts with a unit step size, i.e $t = 1$ and then with each iteration reduces it by a factor of β. Eventually the program will terminate once the stopping condition, in this case $f(x + t\Delta x) \leq f(x) + \alpha t \nabla f(x)^T \Delta x$, is satisfied. The stopping condition is predicated on finding a new point $x + t\delta x$ such that this new point meaningful minimizes the objective function.

With these tools in mind a natural choice for our decsent method is **Gradient Descent** which takes a search direction $\Delta x = -\nabla f(x)$, it can be used to optimize a convex function. It can be implemented with the following algorithm:

Algorithm 3 Gradient Descent

Require: A starting point $x \in dom(f)$, and $\eta > 0$
while $\|\nabla f(x)\|_2 \leq \eta$ **do**

1. $\Delta x = -\nabla f(x)$.
2. Line Search. Choose an appropiate step size t with backtracking line search.
3. Update. $x := x + t\Delta x$.

end while

When the guiding function $g(\theta)$ is convex the gradient descent method can be used to find a global minimum and thus lead to an optimal parameter for the parameterized quantum circuit.

3.2 Finding the Minimal Faithful Dimension with a Variational Quantum Algorithm

Unfortunately $\mathcal{U}$ cannot be both unitarity and have the guiding function be convex, to ensure convexity we will truncate the Taylor series of the Matrix exponential to two terms and accept the residual error.

Every unitary matrix $\mathcal{U}$ can be expressed as e^{iH} where H is hermitian, i.e. $H = H^*$, and can be parameterized by expressing $\mathcal{U}$ with a parameter $\theta \in \mathbb{R}$ and hermitian matrix such that $\mathcal{U} = e^{iH/theta}$

$$\mathcal{U} = e^{iH\theta} \tag{13}$$

where the Taylor expansion yields

$$\begin{aligned} e^{iH\theta} &= I + iH\theta - \frac{\theta^2}{2!} - iH\frac{\theta^3}{3!} + \cdots \\ &= Icos(\theta) + iHsin(\theta) \end{aligned}$$

The guiding function can be convex when the Taylor expansion of the unitary matrix will be truncated to two terms:

$$e^{iH\theta} \approx I + iH\theta \tag{14}$$

When truncated to the nth term, the upper bound on the truncation error can be defined as:

$$R_n(\theta) = \frac{(iH\theta)^{n+1}}{(n+1)!} \tag{15}$$

Given that $\mathcal{U}$ is unitary, it will not change the norm length of any quantum state. The necessity of considering the truncation error is to take into consideration that the approximation will not allow $\mathcal{U}$ to retain its

unitarity, and measuring the parameterized quantum circuit may yield an approximate probability whose error is bounded above by the truncation error.

With truncation error

$$\begin{aligned} Err &= \|e^{iH\theta} - (I + iH\theta)\|_F \\ &= \|Icos(\theta) + iHsin(\theta) - (I + iH\theta)\|_F \\ &= \|I(cos(\theta) - 1) + iH(sin(\theta) - \theta)\|_F \\ &\leq \|R_1(\theta)\|_F \\ &= \left\| \frac{(iH\theta)^2}{2!} \right\|_F \\ &= \left\| \frac{-\theta^2 I}{2} \right\|_F \\ &= \frac{\theta^2\sqrt{2}}{2} \end{aligned}$$

3.2.1 Guiding Function- Single Dimension Case

In this section we derive the general form of guiding when it is a univariate function taking $\theta \in \mathbb{R}$, The probability of finding x when measuring $\mathcal{U}(\theta)\,|0\rangle$ for $\theta \in \mathbb{R}$ and $|x\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ will be defined as:

$$\begin{aligned} p_\theta(x) &= \left|\langle x|e^{iH\theta}|0\rangle\right|^2 \\ &= \left|cos(\theta)\,\langle x|0\rangle + isin(\theta)\,\langle x|H|0\rangle\right|^2 \\ &= |\alpha|^2 cos^2(\theta) + |\langle x|H|0\rangle|^2\, sin^2(\theta) \end{aligned}$$

which is unitary but not convex. However when it is truncated to two terms we have a measurement probability of:

$$\begin{aligned} p_\theta(x) &= |\langle x|\,(I + iH\theta)\,|0\rangle|^2 \\ &= |\langle x|0\rangle + i\theta\,\langle x|H|0\rangle|^2 \\ &= |\alpha|^2 + \theta^2\,|\langle x|H|0\rangle|^2 \end{aligned}$$

with truncated error $R_n(\theta) = \frac{\theta^2\sqrt{2}}{2}$

The single dimensional case for this problem is quite simple, let $f : GL_{l_1} \to \mathbb{C}^+$ be the minimal faithful dimensional for $l_1 = 1$ which means $GL_{l_1} = \mathbb{F}_q/\{0\}$ and $q = p^m$ for some prime p and $m \in \mathbb{N}$. For the single dimensional case each vector is a scalar value in $\mathbb{F}_q$, to represent each of these scalars as a quantum bit we need

$$lg(p^m) = \lfloor log_2(p^m) \rfloor + 2 \text{ qubits.} \tag{16}$$

Then $p_\theta(x)$ can be expressed as:

$$
\begin{aligned}
p_\theta(x) &= \left| \langle x| \left(e^{iH\theta}\right)^{\otimes n} |0\rangle^{\otimes n} \right|^2 \\
&= \left| \langle x|0_n\rangle + i^n\theta^n \langle x|H^{\otimes n}|0_n\rangle \right|^2 \\
&= \left| \sum_{j=1}^{n} \langle x_j|0\rangle + i^n\theta^n \langle x|H^{\otimes n}|0_n\rangle \right|^2 \\
&= \left| \sum_{j=1}^{n} \langle x_j|0\rangle \right|^2 + (-1)^n\theta^{2n} \left| \langle x|H^{\otimes n}|0_n\rangle \right|^2
\end{aligned}
$$

This in turn leads to $g_{mean}(\theta)$ being expressed as

$$
g_{\text{mean}}(\theta) = \left(\left| \sum_{j=1}^{n} \langle x_j|0\rangle \right|^2 + (-1)^n\theta^{2n} \left| \langle x|H^{\otimes n}|0_n\rangle \right|^2 \right) f(x) \tag{17}
$$

It can be seen trivially that g is a convex function when $n = 2k$ for some $k \in \mathbb{Z}^+$. Additonally, it is worth discussing that $f(x)$ is bounded. For some $M \in \mathbb{Z}$,

$$
|f(x)| \leq M \tag{18}
$$

which, together with the truncation error, would imply the following inequality

$$
\begin{aligned}
g(x) &= \left(\left| \sum_{j=1}^{n} \langle x_j|0\rangle \right|^2 + (-1)^n\theta^{2n} \left| \langle x|H^{\otimes n}|0_n\rangle \right|^2 \right) f(x) \\
&\leq \left(\sum (1) f(x) \right) \\
&\leq 2^n M
\end{aligned}
$$

If the convex optimization works reasonably well, given the truncation error,

Example 3.1. Let the parameterized circuit $U(\theta) = e^{iH\theta} \approx I + iH\theta$. For any $x \in \mathbb{F}_{117,649}$, where $p = 7$, $m = 6$, and $7^6 = 117,649$ where $lg(117,649) = 18$ qubits are needed to represent every element as a quantum state. Thus the probability of observing $x \in \mathbb{F}_{117,649}$ is:

$$
p_\theta(x) = \left| \sum_{j=1}^{18} \langle x_j|0\rangle \right|^2 + \theta^{36} \left| \langle x|H^{\otimes 18}|0_{18}\rangle \right|^2 \tag{19}
$$

Thus the guiding function for this example is:

$$
g_{\text{mean}}(\theta) = \left(\left| \sum_{j=1}^{18} \langle x_j|0\rangle \right|^2 + \theta^{36} \left| \langle x|H^{\otimes 18}|0_{18}\rangle \right|^2 \right) f(x) \tag{20}
$$

3.2.2 Guiding Function- Multidimensional Case

The more interesting and relevant problem is the multidimensional case. Finding $\vec{\theta} = (\theta_1, \cdots, \theta_d) \in \mathbb{C}$ such that $F_{\mathfrak{g}}(\vec{\theta})$ is minimized, this makes the underlying problem more complicated since the situation now evaluates a multivariate matrix exponential. The parameterized quantum circuit will take on the following form:

$$\mathcal{U}(\vec{\theta}) = \bigotimes_{j=1}^{d} e^{iH\theta_j} \tag{21}$$

Then for $\vec{\theta} = (\theta_1, \cdots, \theta_d) \in \mathbb{C}^d$:

$$\begin{aligned} p_\theta(x) &= \left| \langle x | \bigotimes_{j=1}^{n} e^{iH\theta_j} |0_n\rangle \right|^2 \\ &= \left| \sum_{j=1}^{n} \langle x_j | I + iH\theta_j |0\rangle \right|^2 \\ &= \left| \sum_{j=1}^{n} \langle x_j|0\rangle + i \sum_{j=1}^{n} \theta_j \langle x_j|H|0\rangle \right|^2 \\ &= \left(\sum_{j=1}^{n} \langle x_j|0\rangle \right)^2 + \left(\sum_{j=1}^{n} \theta_j \langle x_j|H|0\rangle \right)^2 \end{aligned}$$

Hence the guiding function it the multidimensional case is:

$$g_{\text{mean}}(\theta) = \sum_{x \in \{0,1\}^n} \left(\left(\sum_{j=1}^{n} \langle x_j|0\rangle \right)^2 + \left(\sum_{j=1}^{n} \theta_j \right) \langle x_j|H|0\rangle \right)^2 f(x) \tag{22}$$

Example 3.2. Continuing from example 3.1, for any $x \in \mathbb{F}_{117,649}$ 18 qubits are needed and if $d = n = 18$, where d is the dimension of g's domain and n is the number of qubits in the initial superposition, and arbitrary starting point $\vec{\theta} \in \mathbb{R}^{18}$ then:

$$p_\theta(x) = \left(\sum_{j=1}^{18} \langle x_j|0\rangle \right)^2 + \left(\sum_{j=1}^{18} \theta_j \langle x_j|H|0\rangle \right)^2 \tag{23}$$

Which gives us the following guiding function:

$$g_{\text{mean}}(\vec{\theta}) = \sum_{x \in \{0,1\}^{18}} \left(\left(\sum_{j=1}^{18} \langle x_j|0\rangle \right)^2 + \left(\sum_{j=1}^{18} \theta_j \langle x_j|H|0\rangle \right)^2 \right) f(x) \tag{24}$$

4 Solving Constraint Satisfaction Problems with Quantum Walks on a Backtracking Tree

This section describes a generalized approach to use a Quantum Walk to find solutions to a Contraint Satisfaction problem that takes vectors as inputs and produces integer solutions that exist within a bound. The quantum walk will traverse a rooted tree with a backtracking structure with an exponential speedup over traditional backtracking algorithms. This rooted tree will contain vertices mapped to partial solutions and complete solutions of the constraint satisfaction problem.

Definition 4.1 (Minimization Function)**.** Let V be a vector space and $F : V \to \mathbb{Z}$ where $F(v) \in \mathbb{Z}$ is bounded by $a \leq c \leq b$, where the coordinates of v have the possibility to exist as partial solutions and complete solutions to Constraint Satisfaction Problem whose outcome are integers within a defined bound.

4.1 Preliminaries

This section describes the ideas and tools used to solve the rank minimization problem, the goal is to solve a constraint satisfaction problem by traversing a rooted tree, defined with a backtracking structure and logic function, with quantum walk algorithm that provides an exponential speedup over the classical backtracking algorithm described in algorithm 1.

4.1.1 Constraint Satisfaction Problem

Definition 4.2 (Constraint Satisfaction Problem, CSP)**.** A k-ary Constraint Satisfaction Problem is a triple $\langle X, D, C\rangle$ where each set is defined as:

1. $X = \{X_1, \cdots, X_k\}$, a set of variables, that are defined in conjunction to satisfy the problem,
2. $D = \{D_1, \cdots, D_k\}$, a set of domains with each X_i having a domain D_i of possible values, and
3. $C = \{C_1, \cdots, C_m\}$, a set of constraints with each C_i involving some subset of the variables and specifies the allowable combinations of values for that subset.

A valid partial assignment of a given problem is given as a set $\{X_i = v_i, X_j = v_j, \cdots\}$ which does not violate any of the constraints. If the size of the set is equal to n and does not validate any constraints then that is a complete assignment; a complete valid assignment gives a solution to the CSP.

4.1.2 Backtracking Algorithms

A popular algorithm for solving Constraint Satisfaction Problems are backtracking algorithms which have the main objective of using brute force to find all the optimal solutions to a problem. A benefit of a backtracking algorithm is it can elminate partial solutions that do not satisfy the CSP before the algorithm finishes, thus negating any necessity to traverse the entire tree.

Definition 4.3 (Classical Backtracking Algorithm)**.** A classical backtracking algorithm enumerates a set of partial solutions, these partial solutions have the potential to become complete solutions that satsify a given problem, such as a CSP. It sequential searches the tree with the vertices represented as partial solutions and complete solutions, eliminating invalid partial solutions in the process.

Algorithm 4 General Classical Backtracking Algorithm

Require: A predicate function $P : D \to \{true, false, indeterminate\}$ and a heuristic function $h : D \to \{1, \cdots, k-1\}$. Return $bt(*^n)$, where bt is a recursively defined function: $bt(x)$

1. If $P(x)$ is true, output x and return.
2. If $P(x)$ is false, or x is a complete assignment, return.
3. Set $j = h(x)$
4. For $a \in [k]$
   (a) Set y to x with the $j'th$ entry replaced with w
   (b) Call $bt(y)$

In [**2**] the algorithm below is used to solve a k-SAT, Boolean satisfiability problem which is a subset of CSP's, where it either outputs all the problems

The main results of [**2**] prove two theorems, both of which states that for any rooted tree with T vertices there exists a quantum algorithm that provides an exponential speedup over a classical backtracking algorithm that satisfy a predicate function.

Theorem 4.1. *Let T be an upper bound on the number of vertices in the tree explored by Algorithm 2. Then for any $0 < \delta < 1$ there is a quantum algorithm which, given T, evaluates P and h $O(\sqrt{T}nlog(1/\delta))$ times each, outputs true if there exists x such that $P(x)$ is true, and outputs false otherwise. The algorithm uses $poly(n)$ space, $O(1)$ auxiliary operations per use of P and h, and fails with probability at most δ.*

Theorem 4.2. *Let T be the number of vertices in the tree explored by Algorithm 3. Then for any $0 < \delta < 1$ there is a quantum algorithm which makes $O(\sqrt{T}n^{3/2}log(n)log(1/\delta))$ evaluations of each of P and h, and outputs x such that $P(x)$ is true, or "not found" if no such x exists. If we are promised that there exists a unique x_0 such that $P(x_0)$ is true, there is a quantum algorithm which outputs x_0 making $O(\sqrt{T}nlog^3(n)log(1/\delta))$ evaluations of each of P and h. In both cases the algorithm uses $poly(n)$ space, $O(1)$ auxiliary operations per use of P and h, and fails with probability at most δ.*

These theorems suggest there may be a possible quantum advantage to solving constraint satisfaction problems.

4.1.3 Quantum Walks

A quantum walk is the quantum analogue to the classical random walk, whereas as a random walk is a path that consists of a sequence of random steps on some outcome space where there is an associated probability between transitioning into one of several states.

Example 4.1 (Random Walk with Integers)**.** A basic example is a random walk on the set of integers, starting at 0 there is a equal probability of taking a step in the $+1$ direction or the -1 direction, it is clear that the distribution has an expected value of 0. If t is the time step and n is the location on the integer line then probability of this random walk being on position n at time t is:

$$p(t, n) = \frac{1}{2^t} \begin{pmatrix} t \\ \frac{t+n}{2} \end{pmatrix} \tag{25}$$

where $k \in \{-t, -t+2, \cdots, t-2, t\}$ which makes the distribution a well-defined binomial distribution. Which has the following expected value:

$$\begin{aligned}\mathbb{E}(n) &= \sum_{n=-\infty}^{\infty} np(t,n) \\ &= 0\end{aligned}$$

Since $p(t,n) = p(t,-n)$ by symmetry

In the quantum setting there are two descriptions of a quantum walk, a **discete quantum walk** and a **continuous quantum walk**. For our purposes we only be exploring the discrete quantum walk.

Definition 4.4 (Discrete Quantum Walk)**.** Consists of two quantum systems, one of which is called a walker and the other is called a coin, and there is an evolution operator which is applied to both systems in discrete time steps. The evolution of this system for a evolution operator(unitary matrix) U is:

$$|\psi_2\rangle = U|\psi_1\rangle \tag{26}$$

Both continous and discrete quantum walks will be applied to a graph G that is defined by the pair (V,E) where V is the vertex set and E is the edge set.

Example 4.2 (Discrete Quantum Walk with Integers)**.** The most basic example is a *coined model*, continuing from the classical random walk example, a one-dimensional line will be used. In the new quantum context the problem must be *quantized* and the the position of the walker will be described by the quantum state $|n\rangle$ in a Hibert Space $\mathcal{H}_\mathcal{P}$ that has an infinite dimension with computational basis $\{|n\rangle : n \in \mathbb{Z}\}$. The evolution of the walk, described in the above definition, depends on a quantum coin.

In this context we have a walking space denoted $\mathcal{H}_\mathcal{P}$ and a coin space denoted as $\mathcal{H}_\mathcal{C}$, with the Hilbert space of the system being $\mathcal{H} = \mathcal{H}_\mathcal{P} \otimes \mathcal{H}_\mathcal{C}$ where $\mathcal{H}_\mathcal{C}$ has the computation basis of $\{|0\rangle, |1\rangle\}$. So our coin space takes on a superposition of being in either state 0 or 1, and its unitary operator can be defined as C. Hence the shift from $|n\rangle$ to either $|n+1\rangle$ or $|n-1\rangle$ is described by the following unitary shift operator S:

$$\begin{aligned}S|0\rangle|n\rangle &= |0\rangle|n+1\rangle \\ S|1\rangle|n\rangle &= |1\rangle|n-1\rangle\end{aligned}$$

If the action of S on on the computational basis of $\mathcal{H}$ is known, then we have a complete description of this linear operation:

$$S = |0\rangle\langle 0| \otimes \sum_{n=-\infty}^{\infty} |n+1\rangle\langle n| + |1\rangle\langle 1| \otimes \sum_{n=-\infty}^{\infty} |n-1\rangle\langle n| \tag{27}$$

The quantum walk starts by applying the the operator $\mathcal{C} \otimes \mathcal{I}_\mathcal{P}$ to the initital state, which is analogous to the classical case. $\mathcal{C}$ changes the coin state but does not affect the walker.

4.2 Solving the Rank Optimization Problem with a Quantum Walk on a Backtracking Tree whose vertices satisfy a Constraint Satisfaction Problem

4.2.1 The Setup

Consider a rooted tree G with T vertices labelled $r, 1, \cdots, T-1$ where r is the root vertex and the distance from the root to any leaf is at most n. Let $\ell(x)$ be the distance of x from the root. Assume that we do not need

to know the structure of the tree in advance but we can compute $\ell(x)$. With this tree partial assignments can be to the vertices of the tree, each vertex will be a sequence of the form $(\ell, (i_1, v_1), \cdots, (i_\ell, v_\ell))$ for $1 \leq \ell \leq n$, $v_i \in \mathbb{F}$ and, with the exception of the root which is an empty sequence, is connected to its parent which is a sequence of the form $(\ell - 1, (i_1, v_1), \cdots, (i_{\ell-1}, v_{\ell-1}))$, each parent vertex will have q children. The sequence is a partial assignment for $x \in D$ where $x_{i_k} = v_k$ for $k = 1, \cdots, \ell$ and $x_j = *$ otherwise for the unlabelled assignments. The children of the vertex are of the form $((i_1, v_1), \cdots, (i_\ell, v_\ell), (j, w))$ where $j = h((i_1, v_1), \cdots, (i_\ell, v_\ell))$ for $w \in \mathbb{F}$ and $P((i_1, v_1), \cdots, (i_\ell, v_\ell), (j, w))$ is not false.

Vectors in ℓ dimensions over the field $\mathbb{F}$ are mapped to vertices in the ℓth level of the tree G by the function $H : \mathbb{F}^\ell \to \mathcal{H}^{(n)}$ where $H((v_1, \cdots, v_\ell)) = (\ell, (i_1, v_1), \cdots, (i_\ell, v_\ell))$ for $1 \leq \ell \leq n$.

Let A be the set of vertices that are an even distance from the root, and B be the set of vertices that are an odd distance from the root, they are defined as:

- $A = \{(\ell, ((i_1, v_1), \cdots, (i_\ell, v_\ell)(i_{\ell+1}, v_{\ell+1}), \cdots, (i_n, v_n))) : \ell = 2k \in \mathbb{N}, v_i \in \mathbb{F}_q \text{ for } 1 \leq i \leq \ell \text{ and } v_j = * \text{ for } \ell + 1 \leq j \leq n\}$
- $B = \{(\ell, ((i_1, v_1), \cdots, (i_\ell, v_\ell)(i_{\ell+1}, v_{\ell+1}), \cdots, (i_n, v_n))) : \ell = 2k + 1 \in \mathbb{N}, v_i \in \mathbb{F}_q \text{ for } 1 \leq i \leq \ell \text{ and } v_j = * \text{ for } \ell + 1 \leq j \leq n\}$

The only difference between the definitions of the two sets is that ℓ is even in A and ℓ is odd in B. These sets of vertices are used to determine how the tree is traversed in the quantum setting by the linear operators R_A and R_B.

A child y of a vertex x can be expressed as $x \to y$, let d_x be the degree of the vertex x in an undirected graph. For all vertices $x \neq r$, $d_x = |\{y : x \to y\}| + 1$ and $d_r = |\{y : r \to y\}|$. The Hilbert Space $\mathcal{H}$ that the quantum walk acts on is defined by the span of $\{|r\rangle\} \cup \{|x\rangle : x \in \{1, \cdots, T - 1\}\}$ and begins in state $|r\rangle$. A significant difference from the space defined here, and how a discrete quantum is defined in definition 3.2 is that $\mathcal{H}$ does not use an ancilliary coin space. The walk uses a set of diffusion operators D_x.

Definition 4.5 (Diffusion Operator)**.** Denoted as D_x, is a differential operator on R^n that can be written as:

$$D_x = \sum_{i,j=1} \sigma_{ij}(x) \frac{\partial^2}{\partial x_i \partial x_j} + \sum_{i=1} b_i(x) \frac{\partial}{\partial x_i} \tag{28}$$

where b_i and σ_{ij} are continuous functions on $\mathbb{R}^n$.

In quantum computing a diffusion operator has many uses, most prominently it is the central figure in Grover's search which is why it is used here to traverse the rooted backtracking tree.

The diffusion operator acts on the Hilbert Subspace $\mathcal{H}_x = \{|x\rangle\} \cup \{|y\rangle : x \to y\}$ which can be defined as follows:

- If x is marked, then D_x acts as the identity matrix.
- If x is not marked and it is not the root then $D_x = I - 2 |\psi_x\rangle \langle\psi_x|$ where:

$$|\psi_x\rangle = \frac{1}{\sqrt{d_x}} \left(|x\rangle + \sum_{y, x \to y} |y\rangle \right) \tag{29}$$

  If x is a leaf then $|x\rangle = |\psi_x\rangle$

- $D_r = I - 2\left|\psi_r\right\rangle\left\langle\psi_r\right|$ where:

$$\left|\psi_r\right\rangle = \frac{1}{\sqrt{1+d_x n}}\left(\left|r\right\rangle + \sqrt{(}n)\sum_{y,r\to y}\left|y\right\rangle\right) \tag{30}$$

The diffusion operator utilizes part of the Grover search algorithm, in Grover's Unstructured Search algorithm the Grover iterate is defined as $G = (2\left|\psi\right\rangle\left\langle\psi\right| - I)O$ which is a rotation about the vector $\left|\psi\right\rangle$, $2\left|\psi\right\rangle\left\langle\psi\right| - I$ is a reflection about $\left|\psi\right\rangle$ and algorithm 3 uses the reflection to search the subtree of a vertex. Our use of Grover's search algorithm is to search the subspace spanned by a vertex and its children, the amazing part of this algorithm is it simultaneously searches many levels and vertices giving us an indication whether or not a marked vertex exists.

The steps above can be thought of like this; step 1: When x is marked, we do not need to search the all the levels and children vertices of x to determine if that subtree contains a valid assignment because we have already determined that; step 2: if x is not marked and it is not the root then search all of it's children vertices for a marked vertex; step 3: if x is the root vertex then search the entire tree for a marked vertex

Algorithm 5 Quantum Walk: Finds a marked vertex in a graph

Require: Operators R_A, R_B, a failure probability δ, upper bounds on the depth n and the number of vertices T. Let $\beta, \gamma > 0$ be universal constants to be determined.

1. Repeat the following subroutine $K = \lceil \gamma log(1/\delta) \rceil$ times:
   (a) Apply phase estimation to the operator $R_B R_A$on $\left|r\right\rangle$ with precision $\beta/\sqrt{Tn}$
   (b) If the eigenvalues is 1, accept; otherwise, reject.
2. If the number of acceptances is at least $3K/8$, return "marked vertex exists"; otherwise, return "no marked vertex"

Algorithm 5 efficiently determines with local knowledge, that x is or isn't marked and the structure of it's children vertices, if a marked vertex exists within a rooted tree defined by a bracktracking structure that contain solutions to a CSP, meaning it determines that a solution exists for the CSP. The evolution operator of our quantum walk is $R_B R_A$, where:

$$R_A = \bigoplus_{x\in A} D_x \tag{31}$$

$$R_B = \left|r\right\rangle\left\langle r\right| + \bigoplus_{y\in B} D_y \tag{32}$$

To effectively utilize algorithm 5 we need a way to implement R_A and R_B and use them in the quantum walk of a backtracking tree, but before defining the algorithms for those linear operators another linear operator that is apart of a reflection must be defined. Let $U_{\alpha,S}$, for $S \subseteq [d]$ and $\alpha \in \mathbb{R}$, act on $\mathbb{C}^{d+1}$ with basis $\{\left|*\right\rangle, \left|0\right\rangle, \cdots, \left|d-1\right\rangle\}$ by mapping $\left|*\right\rangle \to \left|\psi_{\alpha,S}\right\rangle$, where

$$\left|\phi_{\alpha,S}\right\rangle := \frac{1}{\sqrt{\alpha|S|+1}}\left(\left|*\right\rangle + \sqrt{\alpha}\sum_{i\in S}\left|i\right\rangle\right) \tag{33}$$

It is assumed that $U_{\alpha,S}$ and it's inverse can be performed in time $O(1)$, this allows the reflection $I - 2\left|\phi_{\alpha,S}\right\rangle\left\langle\phi_{\alpha,S}\right|$ to be used in the algorithms for R_A and R_B.

R_A uses $I-2\left|\phi\right\rangle_{n,S}\left\langle\phi\right|_{n,S}$ as a diffusion operator, P, and h as described in Algorithm 4. The implementationR_B is similar to how R_A is implemented, except that: step 1 is replaced with the check "If $P(x)$ is true or $\ell = 0$,

Algorithm 6 Implementation of R_A

Require: A basis state $|\ell\rangle |(i_1, v_1)\rangle \cdots |(i_n, v_n)\rangle \in \mathcal{H}^{(n)}$ corresponding to a partial assignment $x_{i_1} = v_1, \cdots, x_{i_\ell} = v_\ell$. Ancilla regsiters $\mathcal{H}_{anc}$, $\mathcal{H}_{next}$, $\mathcal{H}_{children}$, storing a tuple (a, j, S) where $a \in \{*\} \cup [q]$, $j \in \{0, \cdots, n\}$, $S \subseteq [q]$, initialised to $a = *$, $j = 0$, $S = \emptyset$

1. If $P(x)$ is true, return.
2. If ℓ is odd substract $h((i_1, v_1), \cdots, (i_{\ell-1}, v_{\ell-1}))$ from i_ℓ and swap a with v_ℓ.
3. If $a \neq *$, substract 1 from ℓ. (Now ℓ is even and $(i_{\ell+1}, v_{\ell+1}) = (0, *)$).
4. For each $w \in [d]$:
   (a) If $P((i_1, v_1), \cdots, (i_\ell), (j, w))$ is no false, set $S = S \cup \{w\}$
5. If $\ell = 0$, perform the operation $I - 2 |\phi\rangle_{n,S} \langle\phi|_{n,S}$ on $\mathcal{H}_{anc}$. Otherwise, perform the operation $I - 2 |\phi\rangle_{1,S} \langle\phi|_{1,S}$.
6. Uncompute S and j be reversing steps 5 and 4, in that order.
7. If $a \neq *$, add 1 to ℓ. If ℓ is now odd, add $h((i_1, v_1), \cdots, (i_{\ell-1}, v_{\ell-1}))$ to i_ℓ and swap v_ℓ with a.(Now $a = *$)

return"; "odd" is replaced with "even" in steps 2 and 8; and the check "If $\ell = 0$" is removed from step 6. The first of these changes is because R_B should leave the root of the tree invariant; and the last is because ℓ is always odd at that point in the modified algorithm, so the check is unnecessary.

The goal of the algorithm is to implement $\bigoplus_{x \in A} D_x$, for $x \in A$ D_x only acts on the subspace corresponding to x and its children vertices. This requires that $((i_1, v_1), \cdots, (i_\ell, v_\ell), (j, w))$ for $w \in [q]$, where $j = h((i_1, v_1), \cdots, (i_\ell, v_\ell))$ and $\ell = 2k$ for $k \in \mathbb{Z}$ to a $(d+1)$-dimensional subspace on which $U_{\alpha,S}$ can be implemented and then return the to the original subspace.

The procedure of implementing R_A can be described as:

- Step 1: First it performs the description protocol of the diffusion operator based on whether or not x is marked or not.
- Step $2-3$: Peform a map of the form $|x\rangle \to |y\rangle |*\rangle$ for $x \in A$, and $|x\rangle \to |y\rangle |w\rangle$ for $x \in B$, where w is the value of x at the $h(x')$th position, ie the most recent variable assignment that was made by the backtracking algorithm
- Steps $4-5$: Determine the children of y
- Step 6: Perform the operation $I - 2 |\psi\rangle_y \langle\psi|_y$ using the knowledge of the children of y
- Step $7-8$: Uncompute junk and reverse the first map.

4.2.2 Optimizing the Phase Estimation Step

In [**4**] they observe that the full phase estimation process is not required, just determining if the eigenvalue of a is equal to 1 or very far away from 1. In [**4**] they define this with claim 3 in section 4.1:

conjecture 4.1. If there is a marked vertex, there exists eigenvector $|\psi\rangle$ of $R_B R_A$ with eigenvalue 1 such that $|\langle\psi|r\rangle|^2 \geq 1/2$. Otherwise $\|P_\chi |r\rangle\| \leq \chi\sqrt{Tn}$ for any $\chi \geq 0$ where P_χ is the projector onto the span of eigenvectors of $R_B R_A$ with eigenvalues $e^{2i\theta}$ such that$|\theta| \leq \chi$.

Expanding $|r\rangle$ with its eigenvectors gives $|r\rangle = \sum_j \alpha_j |\psi_j\rangle$ where $|\psi_0\rangle$ has an eigenvalue of 1 and the the

process given in [**4**] yields this following state before the measurement:

$$\frac{1}{M}\sum_j \alpha_j \left(\sum_{x=0}^{M-1} e^{ei\theta_j x}\,|x\rangle\right)|\psi_j\rangle \tag{34}$$

so the probability of the algorithm accepting is:

$$\sum_j |\alpha_j|^2 \mu_j \tag{35}$$

where $\mu_j = \left|\sum_{x=0}^{M-1} e^{ei\theta_j x}\,|x\rangle\right|^2$

4.2.3 Finding a Marked Vertex

In the general problem, to find a marked vertex algorithm 5 is applied to the root and if it returns a "marked vertex exists" then it is applied to each child of the root. This is repeated until a marked vertex is found within the tree. Given we know the upperbound on the number of vertices is $T = n$ where n is the dimension of the of the general linear group, there is at most $O(n)$ repetitions to reach a leaf and $O(1)$ subtrees check at each repetition. To find all the marked vertices, say there are k, will have a running time of $O(k\sqrt{T}n^{3/2}log(n)log(k/\delta))$.

4.2.4 The Algorithm

Let the triple $\langle X, D, C\rangle$ be the constraint satisfaction problem for the rank optimization problem where for each $D_i \in D \subset \mathbb{F}$, where $\mathbb{F}$ is an arbitrary field. The CSP is of the form $(v_1 \wedge \cdots \wedge v_n)$ for $v = (v_1, \cdots, v_n) \in \mathbb{F}^n \cup \{*\}$ and G be the rooted backtracking tree with T vertices and n levels, which has partial solutions of the CSP mapped to its vertices. The predicate function $P : \mathcal{D} \to \{true, false, indeterminate\}$ where $\mathcal{D} = ([q] \cup \{*\})^n$ then P returns:

- "true" if x is a solution to the CSP, at ℓth level of the tree where $v = (v_1, v_2, \cdots, v_\ell, v_{\ell+1}, \cdots, v_n)$ such that $v_i \in \mathbb{F}$ for $1 \leq i \leq \ell$ and $v_j = *$ otherwise(meaning for the vector v that x represents either $F_{\mathfrak{g}}(v) = i$, for the ith iteration, or there exists some child vertex where this is true)
- "false" if x cannot be exteneded to a solution of our CSP, that is, there are no children vertices where the vectors they represent satifies the rank of the commutator matrix.
- "indeterminate" otherwise

The backtracking algorithm takes place in the Hilbert Space $\mathcal{H}^{(n)} = \mathbb{C}^{n+1} \otimes (\mathbb{C}^{n+1} \otimes \mathbb{C}^{d+1})^{\otimes n}$, where each basis vector in $H^{(n)}$ is a partial assignment of the CSP. The first register stores an integer $0 \leq \ell \leq n$ which is the level of the tree that vertex exists on and represents the length of the sequence with decided values, ie values in $\mathbb{F}_q$, the remaining $n - \ell$ values remain undecided, ie they are equal to values in $\{*\}$.

The goal of the proposed algorithm is to find a set of linearly independent vectors $\{v_1, \cdots, v_n\} \subset \mathbb{F}^n$ such that for $a, b \in \mathbb{Z}$ $a \leq F(v_1)) \leq F(v_2)) \leq \cdots \leq F(v_n)) \leq b$ for each $j = 1, \cdots, n$ and arbitrary function $F : \mathbb{F}^n \to \mathbb{Z}$. If the input for the function F is a vector of length ℓ, then all the partial solutions will be found on the ℓth level of the tree and there will be a remaining $n - \ell$ undecided coordinates. Start with a rooted backtracking tree as defined in section 3.2, and iterating through all possible values that the rank of the commutator matrix can take on, which is $\frac{n}{2}$ and gives the outer loop a running time of $O(n)$.

Algorithm 7 Minimization of the Commutator Matrix's Rank

Require: A rooted tree $G(V, E)$ with $|V| = T$ with root vertex r, that is defined with a backtracking tree structure. A function $F : \mathbb{F}^n \to \mathbb{Z}$ and a field $\mathbb{F}$, where $F(v) = i$ for $i \in \{0, 2, 4, \cdots, m\}$.
Result: A set of linearly independent vectors that solves the matrix rank minimization problem
for $i \in \{0, 2, 4, \cdots, m\}$ **do**
  Run algorithm 5 on $|r\rangle$
  if Algorithm 5 returns "there exists a marked vertex" **then**
    Run the Helper Algorithm on each child vertex of $|r\rangle$ and it's subtree
  else
    Move on to the next iteration
  end if
end for

Algorithm 7 is the outer loop of the algorithm, it loops through $i \in \{0, 2, 4, \cdots, m\}$. For each iteration the tree is the same, on a given iteration the tree is searching for all the vectors in $v \in \mathbb{F}^n$ such that $F(v) = i$.

Algorithm 8 traverses all of the rooted subtrees of the original rooted tree that are said to have a marked vertex, the algorithm determines all the vectors $v \in \mathbb{F}$ such that $F(v) = i$. Once the algorithm has concluded for that iteration of the outer a loop it will have compiled a list of vectors $\{v_1, \cdots, v_k\}$ for $k \in \mathbb{Z}$, this algorithm has a running time of $O(kq^{n/2}n^{3/2}log(n)log(k/\delta))$.

Algorithm 8 Recursive Helper Algorithm

Require: A rooted subtree whose root is v, a Predicate function $P : D \to \{true, false, indeterminate\}$
while True **do**
  if $P(v)$ is true and $\ell = m$ **then** Add $v = (v_1, \cdots, v_m)$ to the list of vectors that satisfy the problem.
  end if
  Run Algorithm 5 on $|v\rangle$:
  if Algorithm 5 returns "marked vertex exists" and $|v\rangle$ has no children vertices **then**
    Return $|v\rangle$
  else if Algorithm 5 returns "marked vertex exists" and $|v\rangle$ has children vertices **then**
    Call helper algorithm on each child of $|v\rangle$ from left to right
  else if Algorithm 5 returns "no marked vertex does not exist" **then**
    End loop
  end if
end while

In the following section an explanation will be given for finding the minimal faithful dimension of a $\mathbb{Z}$-Lie algebra using this procedure of algorithm's, and how this procedure can be fit to satisfy that problem.

5 Finding the Minimal Faithful Dimension

In this section the quantum walk on a rooted backtracking tree will be used to find the minimal faithful dimension of a Nilpotent Lie Algebra $\mathfrak{g}$, the problem is defined as a Rank Minimization Problem that will be optimized a backtracking tree that solves a a constraint minimzation problem. The goal is to minimize the rank of the commutator matrix $F_g(v) \in \mathbb{M}_{mxm}(\mathbb{F}_p)$. A quantum walk algorithm is used to traverse a graph that is defined with a backtracking structure and each leaf is equal to some $v \in \mathbb{F}_q^m$, this algorithm aims to find each $v \in \mathbb{F}_q^m$ such that $F_g(v) = i$ for $i = 1, \cdots, m$ and then determine that maximally linearly independent set that finds the faithful minimal dimension of $\mathscr{G}_q$.

Definition 5.1 (Rank Minimization of a Commutator Matrix)**.** Let $\mathfrak{g}$ be a Nilpotent $\mathbb{Z}$-Lie Algebra of nilpotency class c which is finitely generated as an abelian group. A skew symmetric matrix of linear forms

$F_{\mathfrak{g}}(v) \in M_n(\mathbb{F}_q)$ is a commutator matrix of $\mathfrak{g}$. The minimization of $rk(F_{\mathfrak{g}}(v))$ is to find some $v \in \mathbb{F}_q^n$, the $rk(F_{\mathfrak{g}}(v)) = 2k$ for some $k \in \mathbb{N}$.

The quantum walk presented in [**3**] as algorithm 2 determines if a marked vertex exists, in turn this algorithm can give a partial solution to a CSP. This paper and hypothesizes and proves that n linearly independent vectors in $\mathbb{F}_q^n$ can be found with a quantum walk on a rooted tree with a backtracking structure such that each one will solve the rank minimization problem(def 3.1), that is a set of linearly independent vectors $\{v_1, \cdots, v_n\} \subset \mathbb{F}_q^n$ is found and for $a, b \in \mathbb{Z}$:

$$a \leq rk(F_{\mathfrak{g}}(v_1)) \leq rk(F_{\mathfrak{g}}(v_2)) \leq \cdots \leq rk(F_{\mathfrak{g}}(v_n)) \leq b \tag{36}$$

Example 5.1 (Elliptic Curve)**.** In example 2.1 from [**1**] has a $a \neq 0 \in \mathbb{Z}$ for a $\mathbb{Z}$-Lie algebra $\mathfrak{g}_a$ which is spanned by a free $\mathbb{Z}$-module $\{v_1, \cdots, v_9\}$ subject to the following relations:

where for all other brackets $[v_i, v_j]$ with $i < j$ vanish. It was determined by the authors that:

$$\begin{aligned} m_{\text{faithful}}(exp(\mathfrak{g}_a \otimes_{\mathbb{Z}} \mathbb{F}_p)) &= p^{rk_{\mathbb{F}_p}(M(x_{11},x_{12},x_{13}))} + p^{rk_{\mathbb{F}_p}(M(x_{21},x_{22},x_{23}))} + p^{rk_{\mathbb{F}_p}(M(x_{31},x_{32},x_{33}))} \\ &= 3p^2 \end{aligned}$$

where the solutions are $(0,1,0), (0,0,1), (1,y,z) \in \mathbb{F}_p^3$. Applying the quantum walk to this problem would give a tree rooted at $(*,*,*)$ with a depth of 3 and each node would have p children. Using the optimized version of phase estimation described in section 4.2.2, the probability of finding each vector that satisfies the minimal faithful dimension would be $(\frac{1}{2})^3$, and the aggregate probability would be at worst $(\frac{1}{2})^9$ just in the phase estimation process.

6 Conclusion

This paper researched quantum optimization methods that take vectors as inputs and produce a sequence of outputs within a bounds, specifically with the aim of finding the minimal faithful dimension of a p-group for embedding it into a $\mathrm{GL}_n(\mathbb{F}_q)$. The two approaches had various pro's and con's associated with them:

1. Variational Quantum Algorithms is an approach based in parameterizing a quantum circuit with the optimal solutions of a guiding objective function. For finding the minimal faithful dimension the guiding function $g_{\text{mean}}(\theta)$ was used with $d = n$ and the our parameterized circuit circuit $\mathcal{U} = e^{iH\theta}$ described as a Taylor Expansion truncated to two terms made the guiding function convex and for it to be minimized using convex optimization, in which gradient descent can be used to find $\theta^* \in \mathbb{R}^n$ which will giving it a polynomial running time on the classical portion of the optimization and when the depth of the quantum circuit is small this could theoretically lead to a quantum advantage. However truncating $\mathcal{U}$ to two terms does not retain its unitarity, so there may be issues in the transformation of between quantum states which may create issues in the quantum circuit outputting the correct $x \in \mathbb{F}$.
2. Quantum Walk on a rooted backtracking tree that uses the diffusion operator of Grover's search to amplify the amplitude of marked vertices, such as vertices that are mapped to partial solutions of a constraint satisfaction problem. In this context, the partial solution was as a vector input to the commutator matrix function and it was satisfiable if that vector gave the commutator matrix less than or equal to a given integer. If the solution reached a level in the tree that completely satisfied the constraint satisfactions needs for the rank of the commutator matrix then it would a solution would be found, if there are m required solutions then this process would be iterated m times.

In conclusion, there is potential in using these quantum optimization processes to minimize functions that take vectors as inputs but more work needs to be done and results will need to be proven for using these algorithms to find the minimal faithful dimension of a p-group, such as but not limited to:

- A correctness proof for applying the algorithm described in chapter 4 to finding the minimal faithful dimension.
- Proving if truncating the unitary matrix $\mathcal{U}(\theta) = e^{iH\theta}$ to two terms will still yield an optimal solution in the general setting and the applied setting of finding the minimal faithful dimension.
- Writing a script to simulate the quantum walk process and determining the mathematical validity of it's application to finding minimal faithful dimension of a p-group.

7 Acknowledgement

First, I would like to thank my supervisor, Hadi Salmasian. His insight and experience were invaluable, and he gave me a lot of freedom in exploring different quantum optimization methods, which made the research that much more enjoyable.

Secondly, I would like to thank the Faculty of Mathematics at the University of Ottawa for approving my project and providing me with great instructors and a high-quality education.

Finally, I would like to express my gratitude to my family and girlfriend for all their support. I cannot thank them enough for their encouragement throughout my undergraduate education and during my research.

References

[1] Mohammad Bardestani, Keivan Mallahi-Karai and Hadi Salmasian. *Kirillov's orbit method and polynomiality of the faithful dimension of p-groups*, Compositio Mathematica 155 (2019), no. 8, 1618–1654.

[2] Camille Grange, Michael Poss, and Eric Bourreau *An introduction to variational quantum algorithms on gate-based quantum computing for combinatorial optimization problems*,

[3] Ashley Montanaro. *Quantum-walk speedup of backtracking algorithms*, Theory of Computing, 14(15):1–24, 2018.

[4] Earl Campbell, Ankur Khurana, and Ashley Montanaro *Applying Quantum Algorithms to Constraint Satisfaction Problems*,