MAT 4900: Research Project

William Kennedy

April 4, 2024

Abstract

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 $\mathrm{GL}_n(K)$. 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 computation 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 the minimal faith dimension of a

This paper aims to use techniques in quantum computing to minimize the minimal faithful dimension of a p-group $\mathscr{G}_q$:=exp($\mathfrak{g} \otimes_{\mathbb{Z}} \mathbb{F}_q$), which is defined as rank minimization problem of the following form:

Theorem 1.1. *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*

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

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

Above we have a function $F_{\mathfrak{g}}$ whoes domain is the set of nxn skew-symmetric matrices and is defined as follows:

Definition 1.1 (Commutator Matrix)**.** The commutator matrix of $\mathfrak{g}$ is a skew-symmetric matrix of linear forms defined by:

$$F_{\mathfrak{g}}(T_1,\cdots,T_n) := [\Lambda_{ij}(T_1,\cdots,T_m)]_{1\le i,j\le n} \in M_n(\mathbb{Z}[T_1,\cdots,T_m]) \tag{2}$$

for $T_1,\cdots,T_m \in \mathrm{GL}_q^m(\mathbb{F}_q)$

2 Variational Quantum Algorithms

2.1 Preliminaries

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 Intermiate 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 Algebra's.

As formulated in [1], 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 parameters for a paramterized quantum circuit which gives us a high probability of finding the solution to our problem upon measuring the circuit.

2.2 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, this optimal solution guides the function to find the optimal parameter for the 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 2.1 (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

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

Definition 2.2 (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

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

Thus minimizing g will minimize f. The difficulty of this problem is appropiately choosing U, without any information on 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$(CB_n) = F \supseteq F_{quant}$.

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 [1] appendix B.

In [1] their approach to optimizing the guiding function is by using stochastic optimization, in short they sample a distribution of size in f's domaina 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.

2.3 Convex Optimization and Gradient Descent

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, a Python program is used to verify the results of examples.

Definition 2.3 (Convex Function)**.** A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if 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 $i = 1, \cdots, m$.

2.3.1 Descent Methods

Descents 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 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 $0 < \alpha < 0.5$, $0 < \beta < 1$.

1.3 Formulating the Problem in terms of a VQA

The first task for applying this optimization framework to finding the minimal faithful dimension is to define the parameterized quantum circuit such that it satisfies the guiding function's requirements.

The parameterized quantum circuit needs to be defined such that g is a guiding function and p_θ is convex. To start take the Taylor Expansion of a generic unitary matrix $e^i H\theta$ where H is a Hermitian matrix truncated to two terms.

$$e^i H\theta \approx I + iH\theta \tag{13}$$

If H is Hermitian and unitary, then the function $U(\theta) = e^{iH\theta}$ truncated to two terms is unitary.

1.3.1 Single Dimension Case

The single dimensional case for this problem is quite simple, let $f : GL_{\iota 1} \to \mathbb{R}^+$ be the minimal faithful dimensional for $\iota 1 = 1$ which means $GL_{\iota 1} = \mathbb{F}_q$ 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 F_q, to represent each of these scalars as a quantum bit we need

$$lg(p^m) = \lfloor log_2(p^m) \rfloor + 2 \text{ qubits.} \tag{14}$$

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

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

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

$$
g_{mean}(\theta) = \left(\sum_{j=1}^{n} \langle x_j|0\rangle + \frac{(-1)^n\theta^{2n}}{2^n} \left(\prod_{j=1}^{n} \sum_{y\in\{0,1\}} \langle x_j|y\rangle \right)^2 \right) f(x) \tag{15}
$$

It can be seen trivially that g is a convex function when $n = 2k$ for some $k \in \mathbb{Z}^+$

Example 1

Let the parameterized circuit $U(\theta) = e^{iH\theta} \approx I + iH\theta$ where H is the hadamard matrix. For any $x \in F_{117,649}$, where $p = 7$ and $m = 6$, $lg(117,649) = 18$ qubits are needed to represent every element as a quantum state. Thus $p_\theta(x)$ is

$$
p_\theta(x) = \sum_{j=1}^{18} \langle x_j|0\rangle + \frac{\theta^{36}}{2^{18}} \left(\prod_{j=1}^{18} \sum_{y\in\{0,1\}} \langle x_j|y\rangle \right)^2 \tag{16}
$$

Thus the guiding function for this example is:

$$
g_{mean}(\theta) = \left(\sum_{j=1}^{18} \langle x_j|0\rangle + \frac{\theta^{36}}{2^{18}} \left(\prod_{j=1}^{18} \sum_{y\in\{0,1\}} \langle x_j|y\rangle \right)^2 \right) f(x) \tag{17}
$$

I've written a gradient descent algorithm in Python and optimized the above equation, start with an arbitrary $\theta = 15 \in \mathbb{R}$.

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

In this section a different approach is taken 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 3.1 (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 $k \in \mathbb{N}$.

3.1 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.

3.1.1 Constraint Satisfaction Problem

Definition 3.2 (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.

3.1.2 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 3.3 (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.

In [**2**] the algorithm below can be 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.

Algorithm 2 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 $bt(y)$

Theorem 3.1. *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 $O(\sqrt{Tn}log(1/\delta))$ 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 3.2. *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 $O(\sqrt{Tn}log^3(n)log(1/\delta))$ 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.*

3.1.3 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 3.1. 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.

In the quantum setting there are two conceptions 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 3.4 (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:

$$|\psi_2\rangle = U\,|\psi_1\rangle \tag{18}$$

Both continous and discrete quantum walks are applied to discrete graphs.

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

The quantum walk presented in [**2**] 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 such that:

$$rk(F_{\mathfrak{g}}(v_1)) \leq rk(F_{\mathfrak{g}}(v_2)) \leq \cdots \leq rk(F_{\mathfrak{g}}(v_n)) \tag{19}$$

3.2.1 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 $1 \leq \ell \leq n$, $v_i \in \mathbb{F}_q$ and, with the exception of the root which is an empty sequence, is connected to its parent which is a sequence of the form $(\ell - 1, (i_1, v_1), \cdots, (i_{\ell-1}, v_{\ell-1}))$. 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 $w \in \mathbb{F}_q$ and $P((i_1, v_1), \cdots, (i_\ell, v_\ell), (j, w))$ is not false.

Vectors in ℓ dimensions over the field $\mathbb{F}_q$ are mapped to vertices in the ℓth level of the tree G by the function $H : \mathbb{F}_q^\ell \to \mathcal{H}^{(n)}$ 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 def 3.2, is that $\mathcal{H}$ does not use an ancilliary coin space. The walk uses a set of diffusion operators D_x.

Definition 3.5 (Diffusion Operator)**.** A Diffusion Operator in quantum mechanics refers to the spread of quantum particles within a quantum system.

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 $D_x = I - 2\left|\psi_x\right\rangle\left\langle\psi_x\right|$ where:

$$\left|\psi_x\right\rangle = \frac{1}{\sqrt{d_x}}\left(\left|x\right\rangle + \sum_{y,x\to y}\left|y\right\rangle\right) \tag{20}$$

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

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

$$\left|\psi_r\right\rangle = \frac{1}{\sqrt{1+d_x n}}\left(\left|r\right\rangle + \sqrt{(}n)\sum_{y,r\to y}\left|y\right\rangle\right) \tag{21}$$

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

Proposition 3.1. *The diffusion operator* $D_x = I - 2\left|\psi_x\right\rangle\left\langle\psi_x\right|$ *applied to a general quantum state* $\sum_k \alpha_k\left|k\right\rangle$ *gives:*

$$D_x\sum_k \alpha_k\left|k\right\rangle = \sum_k\left[-\alpha_k + 2\langle\alpha\rangle\right]\left|k\right\rangle \tag{22}$$

Proof

$$D_x\sum_k \alpha_k\left|k\right\rangle = (I - 2\left|\psi_x\right\rangle\left\langle\psi_x\right|)\sum_k \alpha_k\left|k\right\rangle$$

Algorithm 3 Quantum Walk: Finds a marked vertex of 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 3 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:

$$R_A = \bigoplus_{x\in A} D_x \tag{23}$$

$$R_B = |r\rangle\langle r| + \bigoplus_{y\in B} D_y \tag{24}$$

To effectively utilize algorithm 3 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 $\{|*\rangle, |0\rangle, \cdots, |d-1\rangle\}$ by mapping $|*\rangle \to |\psi_{\alpha,S}\rangle$, where

$$|\phi_{\alpha,S}\rangle := \frac{1}{\sqrt{\alpha|S|+1}}\left(|*\rangle + \sqrt{\alpha}\sum_{i\in S}|i\rangle\right) \tag{25}$$

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

Algorithm 4 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 $a = *$)

R_A uses $I-2|\phi\rangle_{n,S}\langle\phi|_{n,S}$ 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 $\ell = 0$, 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 $I - 2\left|\psi\right\rangle_y \left\langle\psi\right|_y$ using the knowledge of the children of y
- Step 7-8: Uncompute junk and reverse the first map.

3.2.2 The Algorithm

Let the triple $\langle X, D, C\rangle$ be the constraint satisfaction problem for the rank optimization problem where for each $D_i \in D$, $D_i = \mathbb{F}_q$, meaning each literal in X can take on a value in $\mathbb{F}_q$. The CSP is of the form $(v_1 \wedge \cdots \wedge v_n)$ for $v = (v_1, \cdots, v_n) \in \mathbb{F}_q^n \cup \{*\}$ 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 $v_i \in \mathbb{F}_q$ 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 $\{v_1, \cdots, v_n\} \subset \mathbb{F}_q^n$ such that $rk(F_{\mathfrak{g}}(v_1)) \leq rk(F_{\mathfrak{g}}(v_2)) \leq \cdots \leq rk(F_{\mathfrak{g}}(v_n))$ for each $j = 1, \cdots, n$ the rank of the commutator matrix can be $rk(F_{\mathfrak{g}}(v_j)) = i = 0, 2, 4, \cdots, \ell$ where $rk(F_{\mathfrak{g}}(v_j)) = m$ is a full rank commutator matrix. If the input for the commutator matrix 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 5 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 $v \in \mathbb{F}_q^n$ such that $rk(F_{\mathfrak{g}}(v)) = i$.

Algorithm 5 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 for the commutator matrix $F_{\mathfrak{g}}$ and a finite field $\mathbb{F}_q$, where $rk(F_{\mathfrak{g}}(v)) = i$ 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 3 on $|r\rangle$
  if Algorithm 3 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 6 is derived to traverse a tree that is fit for the Rank minimization problem, where for iteration $rk(F_{\mathfrak{g}}(v)) = i$ we determine all the vertices that satisfies that rank value for the commutator matrix and

from there we traverse to each leaf to create a set of vectors, from which a maximally linearly independent set is found and it is determined whether or not the algorithm needs to move on to the next iteration or if the algorithm can terminate.

Algorithm 6 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 3 on $|v\rangle$:
    if Algorithm 3 returns "marked vertex exists" and $|v\rangle$ has no children vertices **then**
      Return $|v\rangle$
    else if Algorithm 3 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 3 returns "no marked vertex does not exist" **then**
      End loop
    end if
  end while

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.