Variational Quantum Algorithms for Shortest Vector Problem

Memoria del Proyecto

Autor: Zhengkai Zhu

Asignatura: Arquitectura y Programación de Computadores Cuánticos.

Índice

1. Introducción

1.1. Motivación

En los últimos años la palabra Computación Cuántica está cada vez más en boca de mucha gente, tanto en la prensa como en el mundo académico, aunque no muchos son los que se dedican a ello hoy en día.
Para aquellas personas que desconocen sobre el tema, la carta de presentación a la Computación Cuántica siempre ha sido que, si se llega a escalar lo suficiente, sería capaz de superar en capacidad de cómputo a los superordenadores clásicos de hoy en día, cosa que no es del todo cierto cuando ya se profundiza más en el campo.
Una de sus aplicaciones más llamativas y también una de las cuales por la que me ha empezado a interesar desde el principio la Computación Cuántica es su uso para romper los métodos de encriptación de datos y mensajes que está generalizado en el mundo actual.

La criptografía es el estudio de los límites de la capacidad de cómputo de los ordenadores, es decir, investigar sobre problemas que no pueden ser computados de manera eficiente y utilizarlos para proteger los datos y el envío de mensajes.
Hoy en día la mayoría de sistemas de encriptado de claves se basan principalmente en dos problemas que no se pueden resolver de manera eficiente con los computadores clásicos actuales. Estos son:

- Problema de factorización de enteros grandes (RSA).
- Problema del logaritmo discreto (DSA).

No obstante, en 1994 todo cambió cuando Peter Shor mostró una serie de algoritmos cuánticos que serían capaces de resolver estos dos problemas si existiera un computador cuántico lo suficientemente "grande". Todavía a día de hoy no existen computadores cuánticos capaces de hacerlo, pero en los últimos años se ha avanzado mucho en las implementaciones de computadores cuánticos y poco a poco se están escalando más y más. No obstante, todavía no supone un peligro para la criptografía actual ni se prevé que lo sea pronto en los años venideros.
A pesar de todo, esto ha puesto en alerta a empresas y gobiernos, por lo que se ha empezado a investigar en nuevos métodos de encriptado de claves y firmas digitales seguros ante ese posible computador cuántico lo suficientemente potente.
De ahí que ha surgido dos campos:

- Post-Quantum Cryptography: métodos clásicos seguros (por ahora) ante computadores cuánticos.
- Quantum-Cryptography: métodos de encriptado cuántico.

1.2. Objetivos

Este proyecto se va a centrar en la Post-Quantum Cryptography que investiga sobre problemas que son computacionalmente duros tanto para computadores clásicos como cuánticos.
Entre ellos se está viendo mucho potencial los problemas basados en Retículos (Lattice-Based Problems). Tanto es el potencial que el NIST (National Institute of Standards and Technology) de

Estados Unidos ha seleccionado este año 2025, 5 nuevos estándares de encriptación y 4 de ellos son Lattice-Based. [6]

El objetivo principal de este proyecto va a ser introducir la Teoría de Retículos explicando los términos y propiedades básicos para entender el Problem del Vector Más Corto (Shortest Vector Problem SVP) que es el problema NP-difícil principal y del cual parten todos los demás Lattice-Based Problems.
Con esta base, se mostrará la manera en la cual se podría resolver este problema utilizando Algoritmos Variacionales Cuánticos, específicamente el Variational Quantum Eigensolver (VQE). Se explicará cómo se puede codificar el SVP a un problema de buscar el estado base o de energía mínima de un Hamiltoniano. Es decir, se presentará la manera de obtener la función QUBO (Quadratic Unconstrained Binary Optimization), cotas superiores para las variables enteras de la función a minimizar, su conversión a variables binarias y cómo modelar la restricción del vector nulo.
Con esta información, se realizará una pequeña implementación para el problema en dimensión 2 utilizando el framework de IBM, Qiskit.

No obstante, el núcleo de este proyecto será principalmente la base matemática para entender el problema del vector más corto y no en la implementación que se ha realizado para mostrar en esta memoria, pues será más crucial la idea que hay detrás

2. Retículos (Lattices).

En esta sección se realizará un resumen sobre los principios y teoremas básicos sobre la Teoría de Retículos que se explica en el curso de [2].

2.1. Definiciones.

Dado una base de vectores:

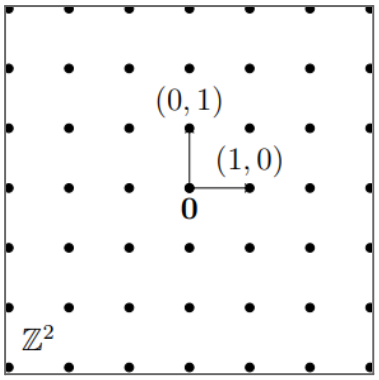
$$B = [b_1, b_2, \dots, b_n] \quad \text{tal que} \quad B \in \mathbb{R}^{dxn}$$

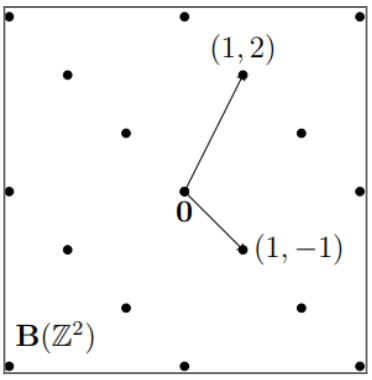
Un retículo o lattice es una estructura algebraica que se define de la siguiente manera:

$$\mathcal{L}(B) = \{Bx : x \in \mathbb{Z}^n\} = \{\sum_{i=1}^{n} x_i b_i : \forall i.\, x_i \in \mathbb{Z}\}$$

Esta definición lo que quiere decir es que un retículo es un conjunto de puntos o vectores distribuidos uniformemente en un espacio vectorial o subespacio vectorial si la base no es del espacio total. Es decir, es un conjunto discreto pero infinito de vectores o puntos que son resultado de combinaciones lineales de los vectores de la base mediante valores que son números enteros.
Ejemplos:

$$B = [(1,0),(0,1)] \qquad\qquad B = [(1,-1),(1,2)]$$





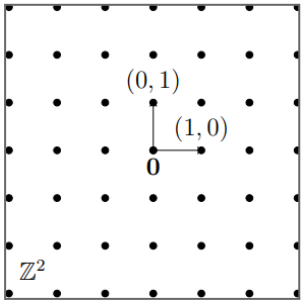
- **Rango del retículo**: número de vectores en la base, en nuestro caso "n".
- **Retículo de rango completo**: retículo de rango máximo, es decir, que tiene rango del espacio vectorial en el que se está trabajando. Si el espacio vectorial tiene dimensión "n" y el retículo es generado por una base de "n" vectores, decimos que el retículo es de rango completo (*full dimensional lattice*).

2.2. Bases.

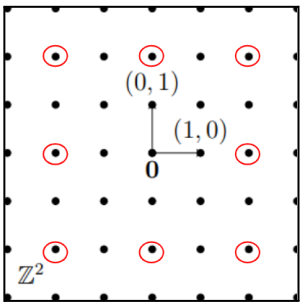
En la sección anterior hemos visto que un retículo se puede generar dado una base B.
Esta definición nos puede recordar mucho a cómo se puede generar un Espacio Vectorial (también subespacio vectorial), pero en los retículos funciona de manera diferente y es lo que hace interesante

Sabemos que dos bases diferentes con "n" vectores generan el mismo espacio vectorial. No obstante, esas dos bases no tienen por qué generar el mismo retículo.
Ejemplo:

$$B = [(1,0),(0,1)]$$
$$\mathcal{L}(B) = \mathbb{Z}^2$$



$$B' = [(2,0),(0,2)]$$
$$\mathcal{L}(B') = 2\mathbb{Z}^2$$



La base *B'* es simplemente la base *B* donde sus vectores se han multiplicado por 2. A pesar de ello, el retículo formado por *B'* es el retículo formado por *B* pero que los puntos se separan cada 2, tal y como se ve en la imagen.

Entonces, una pregunta natural que nos puede surgir es la siguiente:
¿Pueden dos bases diferentes formar el mismo retículo? ¿Si es afirmativa la respuesta, qué propiedades cumplen estas dos bases?

La respuesta a la primera pregunta es sí, dos bases pueden generar el mismo retículo, de hecho no son únicas. Un retículo puede ser representado por una infinidad de bases diferentes.

Definición:

Una matriz cuadrada de enteros $U \in \mathbb{Z}^{n\times n}$ *es invertible si existe una matriz* $V = U^{-1} \in \mathbb{Z}^{n\times n}$ *tal que* $VU = UV = I$.

En esta definición buscamos la inversa de una matriz cuadrada entera como otra matriz entera cuadrada, lo que nos servirá para las bases "equivalentes" del mismo retículo.

Definimos $GL(n, \mathbb{Z})$ como el conjunto de las matrices cuadradas de enteros invertibles (según la definición de inversa que hemos dado).

Teorema:

Sean B y C dos bases de retículos.

$$\mathcal{L}(B) = \mathcal{L}(C) \iff \exists U \in GL(n, \mathbb{Z}) \text{ } _{tal\ que} \text{ } B = CU$$

Lo que quiere decir el teorema es que dos bases generarán el mismo retículo si se puede llegar de una base a otra mediante mediante el producto por una matriz de enteros cuadrada.

La demostración del teorema la dejaré en las referencias.

Esto también quiere decir que podemos llegar de una base B a otra C, si realizamos una serie de operaciones sobre B llamadas ***operaciones de columnas elementales*** (si los vectores lo consideramos como vectores columna en la matriz).

Estos son:

- SWAP(i, j): Cambiamos la columna i con la j.
  $$(b_i, b_j) \leftarrow (b_j, b_i)\ i \neq j$$
- INVERT(i): Cambiamos el signo del vector en la columna i.
  $$(b_i) \leftarrow (-b_i)$$
- ADD(i, c, j): Sumamos c veces el vector de la columna j al vector de la columna i.
  $$(b_i, b_j) \leftarrow (b_i + c * b_j, b_j)\ i \neq j \wedge c \in \mathbb{Z}$$

Al aplicar estas operaciones de columnas elementales sobre una base, obtenemos otra base tal que ambos generan el mismo retículo.

Además cabe destacar que si tenemos una serie de operaciones de columnas elementales σ aplicados sobre una base B se puede entender como:

$\sigma(B) = \sigma(BI) = B\sigma(I)$ donde $\sigma(I) \in \mathbb{Z}^{n\times n}$.

Es fácil demostrar que $\sigma(I) = U \in GL(n, \mathbb{Z})$.

A continuación voy a aportar una definición que nos servirá más adelante para el punto de los determinantes.

Definición:

Una matriz $B \in \mathbb{Z}^{n \times n}$ *es unimodular si* $|det(B)| = 1$.

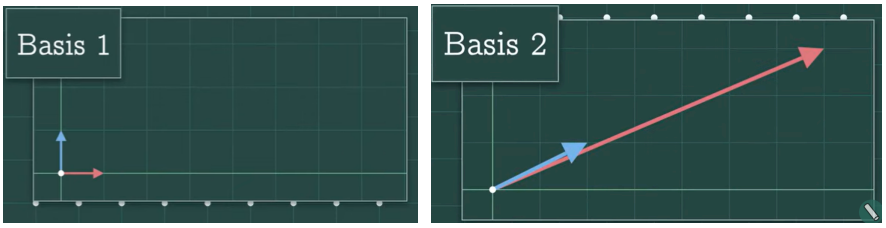
Con todo se deduce que una matriz cuadrada de enteros invertible U es unimodular pues es el fruto de operaciones de columnas elementales sobre la matriz identidad.

Entonces tenemos el siguiente ejemplo de bases:

$$B = [(1,0),(0,1)] \qquad\qquad B' = [(2,1),(7,3)]$$

Es fácil ver que ambas son bases del mismo retículo.

$$\mathcal{L}(B) = \mathcal{L}(B') = \mathbb{Z}^2$$



No obstante, son dos vectores que a simple vista son muy diferentes entre sí. La base B es la base canónica donde tenemos dos vectores con norma 1 y son ortogonales entre sí. Viendo está base podemos fácilmente generar o intuir el retículo que generan.
Por otro lado, B' ya no es tan claro, está formado por dos vectores con módulo mayor y no son ortogonales formando un ángulo muy cerrado entre sí. A esta base podemos denominarlo como una **base "mala".**
Las bases "buenas" y "malas" cobran vital importancia para en los problemas basados en retículos y es lo que los hace problemas NP-difíciles. Esto significa que dependiendo de las bases que tengamos, afecta en la eficacia (tanto complejidad temporal y de memoria como en la calidad de la solución) de los algoritmos para resolver problemas como el SVP, el CVP (Closest Vector Problem), las versiones aproximadas y variantes.

2.3. Gram-Schmidt Orthogonalization.

En esta sección voy a mostrar un procedimiento para "transformar" una base B en una base que es ortogonal mediante el método de Ortogonalización de Gram-Schmidt.
La base ortogonal será útil para calcular el determinante o el área del paralelepípedo formado por los vectores de una base B, establecer una cota inferior para el Problema del Vector Más Corto y también aportar un coeficiente llamado coeficiente de Gram-Schmidt , el cual será necesario tener en cuenta para los algoritmos para "mejorar" bases "malas" (LLL, BKZ).

Definición:

Dado una base $B = [b_1, b_2, \ldots, b_n]$ *definimos la operación proyección ortogonal de Gram-Schmidt* $\pi_i : \mathbb{R}^n \to \mathbb{R}^n$ *como :*

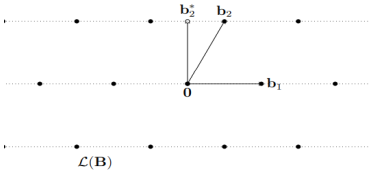
$$\pi_i(x) = x \perp [b_1, \ldots, b_{i-1}]$$

Esta función lo que hace es proyectar un vector del espacio vectorial sobre el subespacio vectorial perpendicular al subespacio generado por los vectores *i = 1, ..., i - 1,* de la base B.
Entonces definimos:

Definición:

Dado una base $B = [b_1, b_2, \ldots, b_n]$ *su base ortogonal Gram-Schmidt es la base* $B^* = [b_1^*, \ldots, b_n^*]$ *donde cada* $b_i^* = \pi_i(b_i)$.

Cabe destacar que B* no es, en general, una base del retículo $\mathcal{L}(B)$ pero ambos generan el mismo espacio vectorial.
Ejemplo:



De esta definición deducimos el siguiente lema:

Lema:

$b_i^* = b_i - \sum_{j<i} \mu_{i,j} b_j^*$ *donde* $\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$ *llamado coeficiente de Gram-Schmidt.*

2.4. Determinante.

El determinante de una base B o también se denomina como el volumen del retículo es un concepto fundamental a la hora de establecer cotas superiores para el Problema del Vector Más Corto.

Definición:

Dado una base $B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$ *, el paralelepípedo fundamental asociado a B es el conjunto de puntos:*

$$\mathcal{P}(B) = \{\sum_{i=1}^{n} x_i b_i : \forall i : 0 \leq x < 1\}$$

Entonces el volumen de este espacio de puntos es el siguiente:

$$vol(\mathcal{P}(B)) = det(B) = \prod_i \|b_i^*\|$$

donde cada b_i^* es un vector de la base ortogonal de Gram-Schmidt $B^* = [b_1^*, \ldots, b_n^*]$ *de B.*

Cabe destacar que $\|b_i^*\| \leq \|b_i\|$.

Teorema:

Dado dos bases B y C de un retículo. Se cumple que:

$$\mathcal{L}(B) = \mathcal{L}(C) \Rightarrow det(B) = det(C)$$

La demostración del teorema es sencilla si consideramos que podemos pasar de una base a otra mediante operaciones de columnas elementales que hemos descrito en el apartado sobre bases (apartado 2.2).

Este teorema es fundamental a la hora de establecer una cota superior para el problema del vector más corto, pues para un mismo retículo no importa la base que escojamos que lo genere, el determinante es el mismo. Con ello, tenemos un valor que es constante para un mismo retículo.

3. Shortest Vector Problem.

En este punto vamos a considerar la norma Euclídea, pero es extrapolable para otras normas.

Definición:

Dado una base $B = [b_1, b_2, \ldots, b_n]$ *y el retículo que genera* $\mathcal{L}(B)$ *se define la distancia mínima del retículo como la menor distancia entre cualquier par de puntos del retículo:*

$$\lambda_1(\mathcal{L}(B)) = inf\{\|x - y\| : x, y \in \mathcal{L}(B), x \neq y\}$$

Es decir, es el módulo del vector más corto dentro del retículo:

$$\lambda_1(\mathcal{L}(B)) = inf\{\|v\| : v \in \mathcal{L}(B), v \neq 0\}$$

Sabemos que dado cualquier retículo esta distancia existe. Si consideramos una hiperesfera con centro el vector nulo y obligamos que haya al menos un punto del retículo tenemos garantizado que existirá un vector de distancia mínima de distancia mínima más allá del vector nulo. La razón es que tenemos un conjunto discreto de puntos ordenado en función del módulo, está claro que existirá mínimo/s.

Teorema (cota inferior):

Para cualquier retículo generado por B y su base ortogonal Gram-Schmidt B tenemos que*

$$\lambda_1(\mathcal{L}(B)) \geq min_i\|b_i^*\|$$

Definición:

El Problema del Vector Más Corto (Shortest Vector Problem, SVP) consiste en que dado una base B de un retículo, queremos encontrar el vector más corto diferente de cero del retículo:

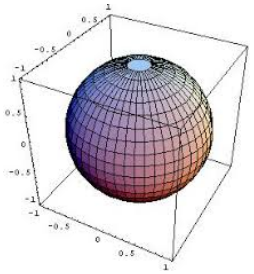
$$v \in \mathcal{L}(B) : \|v\| = \lambda_1(\mathcal{L}(B))$$

3.1. Complejidad y algoritmos de SVP.

A día de hoy el Problema del Vector Más Corto es un problema computacionalmente difícil, tanto la versión exacta del problema como la aproximada por un factor γ.
Debido a esta complejidad, que aplica también para los computadores cuánticos, están siendo clave los problemas basados en retículos para la investigación de la Criptografía Post Cuántica.
En clásica existen numerosos algoritmos cuya eficacia en el caso peor lo hace un problema NP-difícil en la mayoría de casos. Todos ellos dependen de la "calidad" de la base de entrada que afecta en la aproximación obtenida del vector, el tiempo de ejecución y en la memoria.

Algoritmos de enumeración

Son un tipo de algoritmos que consisten principalmente en acotar el espacio de búsqueda, es decir, dado una cota superior del vector más corto, centrar la búsqueda en la hiperesfera que tiene como centro el vector nulo y radio la cota superior.



Reducido el espacio de búsqueda tenemos como variables el vector $x = (x_1, x_2, ..., x_n) \in \mathbb{Z}^{n\,x\,n}$. Entonces el procedimiento de estos es fijando el valor de una componente del vector para la base $B = [b_1, b_2, ..., b_n]$. A continuación buscamos el vector para una base con un vector menos y un vector con una variable menos.

- *Babai's Nearest Plane Algorithm*: algoritmo voraz que fija heurísticamente el valor de las variables. Es un algoritmo que nos proporciona un vector aproximado, que no es el más corto. Además, le afecta tanto la "calidad de la base" que el vector que devuelve puede ser en función de un factor de aproximación arbitrariamente grande. Es un algoritmo polinómico pero la calidad de la solución no lo es.
- *Branch and Bound:* algoritmo que nos devuelve el vector exacto, y se basa en la ramificación lo que nos indica que tendrá complejidad como mínimo exponencial. La idea del algoritmo es probar diferentes valores para cada x_i y hacer llamadas recursivas en función de cada valor. Luego busca entre las soluciones de las llamadas recursivas el vector más corto.

Algoritmos de reducción de bases

Tal y como se ha ido comentando hasta este momento, para poder mejorar la eficacia o complejidad a la hora de resolver estos problemas Lattice-based, es de crucial importancia encontrar mejores

bases. Entonces muchos algoritmos se centran principalmente en reducir las bases para que sean mejores antes de aplicar sobre ellos otros algoritmos que lo resuelvan.

- *LLL algorithm:* algoritmo de Lenstra, Lenstra y Lovász (1982). Este algoritmo permite mejorar algo la base de entrada en función del coeficiente de Gram-Schmidt. De paso también consigue aportar un vector más corto pero con un factor de aproximación exponencial, lo cual tampoco es demasiado bueno, pese a ser polinomial en tiempo.
- *Schnorr LLL algorithm:* variante del algoritmo de LLL, pero aumentando los bloques que considera, lo cual sacrifica tiempo de ejecución por una mejor aproximación de base. Otra variante sería el *BKZ algorithm*.

3.2. Cotas superiores y Teorema Minkowski.

Estamos trabajando con retículos que es un conjunto infinito y no podemos hacer una búsqueda en todo el espacio, por lo que es de crucial importancia establecer una cota superior de la distancia mínima para así poder reducir el espacio de búsqueda.
En esta sección se va a presentar el Teorema de Minkowski para cuerpos convexos el cual nos proporciona unas bases para establecer cotas superiores que sean, únicamente dependientes de la dimensión y del determinante de la base con la que estemos trabajando (que es el mismo para todas las bases del retículo). No obstante, no se va a entrar en los detalles de demostración (se dejará en la bibliografía).

Teorema de Minkowski

Sea L un retículo de rango completo o máximo. Si $S \subseteq \mathbb{R}^n$ *un cuerpo convexo y simétrico con volumen* $vol(S) > 2^n det(L)$ *entonces S contiene al menos un vector del retículo distinto de 0.*

De este teorema se puede deducir una serie de cotas superiores:

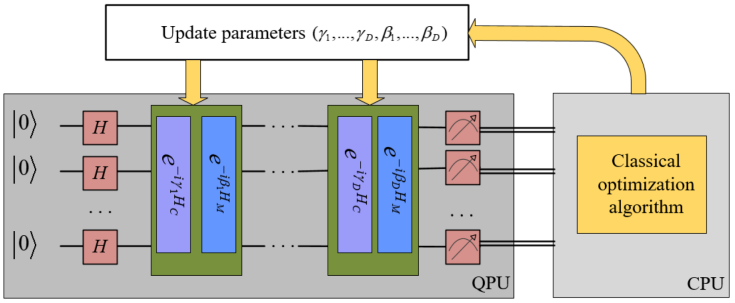
- La más general es:
  $$\lambda_1(\mathcal{L}(b)) \leq \sqrt{n} \cdot det(B)^{1/n}$$
- Otra depende el volumen de la esfera de radio una unidad en dimensión n:
  $$\lambda_1 \leq \frac{2}{vol(\mathbb{S}^n)^{1/n}} det(\mathcal{L}(B))^{1/n}$$
  $vol(\mathbb{S}^n)$ Es el volumen de la esfera de radio unitario en dimensión n.
- Heurística Gaussiana:
  Como el volumen de la esfera es diferente para cada número de dimensión, podemos estimar el valor del volumen:
  $$vol(\mathbb{S}^n) \approx (\frac{2\pi e}{n})^{n/2} \qquad n >> 0 \implies \lambda_1 \leq\approx \sqrt{\frac{2n}{\pi e}} det(\mathcal{L}(B))^{1/n}$$

Este es el denominado Heurística Gaussiana. Es una estimación general de la cota superior en base de una aproximación del volumen de la esfera en dimensión n. Es adecuado para n muy grandes, pero para n pequeños no sirve pues es demasiado optimista y puede dejar fuera el vector mínimo.

4. Variational Quantum Algorithms (VQA).

4.1. Introducción a los VQA's.

Los algoritmos cuánticos variacionales (VQA's [4]) son algoritmos híbridos que mezclan una parte que es cuántica y otra clásica.



Estos algoritmos son los que más potencial tienen en los ordenadores cuánticos que tenemos hoy en día, denominados NISQ (Noisy Intermediate-Scale Quantum). Los computadores cuánticos que tenemos a día de hoy son todavía muy susceptibles al ruido y por ello al error si tenemos un circuito algo más complejo.

Por lo tanto los algoritmos variacionales cuánticos se aprovechan de construir circuitos cuánticos parametrizados no muy grandes para realizar la parte de cálculo más compleja del problema y el resultado pasarlo a un optimizador clásico el cual busca los mejores parámetros.

Son algoritmos que se utilizan para problemas de optimización en los cuales estamos buscando el estado de energía mínima de un Hamiltoniano (VQE y QAOA son los más importantes).

Entonces los pasos a realizar para implementar un algoritmo VQA para nuestro problema son los siguientes:

1. Codificar nuestro problema es buscar el estado de energía mínima de un Hamiltoniano. Para ello tenemos que codificar nuestra función a optimizar en una función QUBO (Quadratic Unconstrained Binary Optimization).
2. Con la función QUBO podemos construir nuestro operador Hamiltoniano $\boldsymbol{\mathcal{H}}$ el cual estima la energía de un estado cuántico dado.

   $$C(\psi) = \langle\psi|H|\psi\rangle$$

   En nuestro caso vamos a implementar el algoritmo VQE codificando un Hamiltoniano de tipo ising que solo involucra puertas de Pauli Z y la I (no se va a entrar en los detalles del algoritmo pues no es el propósito del proyecto, pero se deja en la bibliografía [1] , [4]).
3. Luego tenemos que escoger nuestro circuito Ansätze, el cual es un circuito parametrizado que nos devuelve un estado cuántico que pasamos al Hamiltoniano para estimar el coste o energía de dicho estado.

4. Finalmente la media de los costes obtenidos del Hamiltoniano lo pasamos a un optimizador clásico el cual se encarga de buscar los mejores parámetros para el circuito Ansätze.

Función QUBO general:

$$C(s_1 s_2 \ldots s_n) = c + \sum_i c_{ii} s_i + \sum_{i \neq j} c_{ij} s_i s_j$$

donde s1,s2,...sn son variables binarias y
$c, \{c_{ij}\}_{1 \leq i,j \leq n}$ son los coeficientes.

4.2. Codificación del SVP.

El SVP es un problema de optimización donde estamos buscando los valores de n variables enteras para minimizar la función de módulo o norma Euclídea. Por lo tanto, podemos codificarlo como un problema de buscar el estado de energía mínima de un Hamiltoniano.

Función cuadrática a minimizar:

$$f(x) = x^T B^T B x \qquad x \in \mathbb{Z}^n$$

$$Bx = y \in \mathcal{L}(B)$$

$$x = (x_1, x_2, \ldots, x_n) \quad x_i \in \mathbb{Z}$$

Entonces cada x_i es una variable de nuestra función.

Formulación completa del problema:

$$\lambda_1^2 = \min_{y \in \mathcal{L}(B) \setminus \{0\}} \|y\|^2$$

$$= \min_{x \in \mathbb{Z}^n \setminus \{0\}} \sum_{i=1}^{n} x_i \cdot G_{ii} + 2 \sum_{1 \leq i < j \leq n} x_i \cdot x_j \cdot G_{ij} \qquad G = B^T B$$

Hay que tener en cuenta que estamos considerando matrices columna y de ahí esta formulación del módulo.

Aún queda dos problemas a considerar para completar la formulación:

- Convertir las variables enteras en variables binarias y por ello acotarlos.
- Restricción del vector nulo.

4.3. Acotar las variables enteras.

Para que sea una función QUBO, necesitamos que las variables del problema sean variables binarias, es decir que tomen valores en $\{0, 1\}$.

Entonces, al tener una cantidad finita de qubits, tenemos que acotar nuestras variables enteras para así tener una cantidad finita de variables binarias. Además queremos que esa cota sea lo más ajustada posible para limitar al máximo el número de qubits.

Conversión a variables binarias:

Supongamos que tenemos una cota a_i establecida para cada variable entera x_i.

$$|x_i| \leq a_i$$

Entonces podemos definir unas nuevas variables binarias para cada x_i :

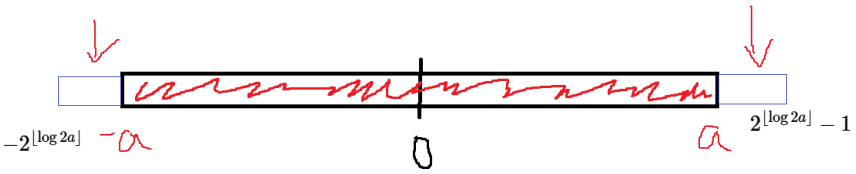
$$\{\tilde{x}_{ij}\}_{0 \leq j \leq \lfloor \log 2a \rfloor}$$

Nos quedaría la siguiente expresión para cada variable binaria:

$$|x_i| \leq a \implies \tag{2}$$

$$x_i = -a + \sum_{j=0}^{\lfloor \log 2a \rfloor - 1} 2^j \, \tilde{x}_{ij} + (2a + 1 - 2^{\lfloor \log 2a \rfloor}) \cdot \tilde{x}_{i, \lfloor \log 2a \rfloor}$$

Esta conversión lo que hace es codificar las variables binarias para que tomen valores (en Complemento a 2) del siguiente intervalo:



Si los valores "caen" fuera del intervalo $[-\ a,\ a]$ la expresión penaliza sumándole valores positivos a la función QUBO.

Discusión de cotas superiores:

En la sección que he enunciado el Teorema de Minkowski para Cuerpos Convexos, he derivado 3 posibles cotas superiores.

La primera de ella no sería la más adecuada pues es demasiado pesimista.

La Heurística Gaussiana es la menor de las 3, pero no funciona en general para dimensiones pequeñas. En dimensiones grandes funcionaría para la mayoría de los casos, pero al ser una cota aproximada/heurística, daría problemas cuando la distancia mínima sea mayor que la heurística.

Entonces, la segunda es la más ajustada y que garantiza que siempre encontraremos el vector mínimo si buscamos en la hiperesfera de dimensión n de ese radio.

$$\lambda_1 \leq \frac{2}{vol(\mathbb{S}^n)^{1/n}} det(\mathcal{L}(B))^{1/n}$$

Cota superior para cada variable x_i:

Una vez teniendo una cota superior A de la distancia mínima, podemos establecer una cota superior para cada una de las variables y eso depende de la base dual del retículo.

Definición:

Dado una base $B = [b_1, b_2, \dots, b_n]$ *y el retículo que genera* $\mathcal{L}(B)$ *definimos el dual como:*

$$\widehat{\mathcal{L}(B)} = \{w \in \mathbb{R}^d : \langle w, x \rangle = z, \forall x \in \mathcal{L}(B), z \in \mathbb{Z}\}$$

Es decir, es el conjunto de puntos del espacio vectorial generado por B que su producto escalar con vectores de $\mathcal{L}(B)$ es un número entero.
Además, tiene como base:

$$\hat{B} = [\hat{b_1}, \ldots, \hat{b_n}] \text{ donde } \hat{B} = B(B^T B)^{-1}$$

Y para bases de rango completo o máximo la base dual es: $\hat{B} = B^{-T}$
Con ello deducimos que el dual de un retículo también es un retículo.
El retículo dual cumple una serie de propiedades que lo relacionan íntimamente con el retículo original y son fundamentales para diferentes algoritmos de reducción de bases (no se va a entrar en detalles, [2], [3]).

Para nuestro problema lo que nos interesa es el siguiente lema.

Lema ([1]):

Sea $B = [b_1, b_2, \ldots, b_n]$ *una base de un retículo y sean* $x_1, x_2, \ldots, x_n$ *variables tales que*

$$\|x_1 \cdot \boldsymbol{b}_1 + \cdots + x_n \cdot \boldsymbol{b}_n\| \leqslant A$$

Entonces

$$|\boldsymbol{x_i}| \leq \boldsymbol{A}\|\hat{\boldsymbol{b_i}}\| \quad \forall \boldsymbol{i} = \boldsymbol{1}, \ldots \boldsymbol{n}$$

Así ya tenemos una cota superior para cada una de las variables enteras.

4.4. Restricción del vector nulo.

Otro de los problemas que tenemos que resolver de alguna manera es el problema del vector nulo para que nuestra función cuadrática ya sea QUBO definitivamente.
Hay diferentes procedimientos para afrontar la restricción y las más destacadas son:

- Añadirlo como restricción a la función QUBO mediante una penalización cuando todas las variables tomen el valor 0. No obstante, este procedimiento implicaría añadir más qubits a nuestro problema. La explicación de cómo se realizaría se explica en el Apéndice A del siguiente *paper* [1].
- Alterar la función de coste o el coste final que se va a pasar al optimizador clásico para que de energía infinita cuando el estado sea el estado 0.
  Cambiar nuestra función de coste:
  $$C(\psi) = \langle\psi|H|\psi\rangle$$
  por la siguiente función de coste.
  $$C(\psi) = \langle\psi|H|\psi\rangle \frac{1}{1-|\langle\psi|\psi_0\rangle|^2}$$
  Se multiplica el coste o la energía estimada del estado por la inversa de la probabilidad de no medir el estado base que es el estado de todo 0's. Entonces, esto aporta energía infinita al estado del vector nulo y la minimización de esta función de coste ya nos devuelve el vector mínimo.
  Como necesitamos saber la amplitud del estado |0>, para implementarlo necesitamos realizar dos tipos de mediciones diferentes sobre el circuito Ansätze en la cantidad de *shots*

que realicemos de dicho circuito . Hacemos una medición que pase por el Hamiltoniano (para estimar la energía) y otro que no (para estimar la amplitud del estado |0>).

$$C'(\theta) = \left(\frac{N}{\tilde{N}}\right) \frac{1}{N} \sum_{i=1}^{N} m_i$$

Ñ es el número de mediciones en los que no se ha medido el estado 0 y N el número de shots totales. El sumatorio es la energía estimada.

5. Implementación sencilla con IBM Qiskit.

5.1. Librerías principales utilizadas.

Para implementar las ideas explicadas en este proyecto he utilizado el framework de IBM llamado Qiskit [7].
Cabe destacar que se ha utilizado la versión de Qiskit 2.0, por lo que las librerías de "qiskit_algorithms", para el momento en el que se ha realizado el proyecto, no son compatibles para dicha versión de Qiskit.
Se ha utilizado entonces las principalmente las librerías "qiskit_optimization" y "qiskit_ibm_runtime" para el uso de los estimadores, samplers, conversores de problemas cuadráticos a QUBO, traductores a Hamiltonianos de tipo ising, entre otros.
Para representar las matrices se ha utilizado las librerías de "numpy" y "sympy" de Python.
Además, se ha utilizado la función "minimize" de la librería "scipy" de Python.
Por otro lado, el circuito Ansätze que se ha utilizado para esta implementación es el circuito parametrizado EfficientSU2 de la librería "qiskit.circuit.library".
Para más detalles ver el Jupyter Notebook asociado al proyecto.

Para otra posible implementación de una VQE consultar el libro [5].

5.2. Implementación del VQE.

La implementación que se ha realizado en este proyecto para aplicar las ideas que se han explicado es la del VQE.
Los puntos más importantes de esta implementación son las siguientes:

- Con la clase QuadraticToQubo de *qiskit_optimization.converters* no nos permite convertir un QuadraticProgram con restricciones cuadráticas a una función QUBO, por lo para restringir el vector nulo lo que hemos hecho ha sido realizar el procedimiento de modificar la función de coste del estado.
- Mediante el EstimatorV2 de *qiskit_ibm_runtime* nos permite estimar el coste de un estado dado un Ansätze, los parámetros y el Hamiltoniano. No obstante, para contar y estimar la amplitud del estado |0> necesitamos un Sampler (se utiliza SamplerV2 de *qiskit_ibm_runtime*) también con el que podemos obtener la cuenta de cuántas veces medimos 0's en los shots.
- El optimizador que se ha utilizado para realizar las pruebas es "Cobyla" de la librería *scipy.optimize.minimize* de Python.

- Además todas las pruebas realizadas son con simulación, pues en backend real se tomaría demasiado tiempo y no son tan accesibles las backend's de IBM.

5.3. Pruebas.

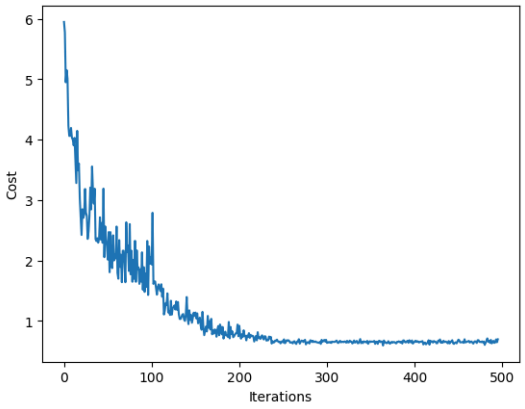
En este proyecto se ha probado con dos bases que generan el mismo retículo en dimensión 2 para obtener circuitos medianamente reducidos.
Una de las bases es una base "buena" y la otra es una base "mala".

$$B = \begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix}$$

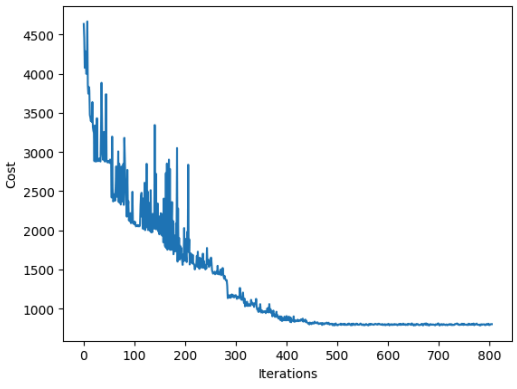
$$C = [(11, 4), (-9, 3)]$$

$$C = \begin{pmatrix} 11 & -9 \\ -4 & 3 \end{pmatrix}$$

Se ha ejecutado el algoritmo para ambas bases y nos han arrojado los siguientes resultados de evolución del valor de energía:
Base B



Base C



Para la base B, el optimizador ha conseguido devolvernos (1, 0) que representa el vector (1, 1) del retículo, el cual es el vector mínimo.
En cambio, para la base C no ha ocurrido lo mismo, nos devuelve la función las variables (-2, -2) que representa el vector (-4, -2) el cual no es un vector mínimo, pero está algo cerca de serlo.
La razón para explicar esto es que al ser C una base con vectores de valores algo "grandes", se ha necesitado una cantidad mayor de qubits para representarlo (8 qubits) que para un circuito Ansätze repetido 3 veces, tenemos 54 parámetros para que optimice el optimizador clásico, lo cual es muy

complicado. Para la base B solo se ha necesitado 4 qubits y un Ansätze repetido 4 veces tiene tan solo 40 parámetros.
Se deja al lector probar para un Ansätze con menos iteraciones para ver si sacrificando más expresividad del circuito podamos optimizar algo mejor.

5.4. Detalles importantes de implementación.

Hay que considerar que la clase QuadraticToQubo [7] NO transforma las variables enteras a variables binarias en Complemento a 2.
Esto quiere decir que si tenemos 2 variables enteras y si a cada una se cambia por 3 variables binarias, el valor en binario 000 de una variable no representa el 0.
Ejemplo:
Tengo las variables enteras x1 y x2 tales que:
-3<= x1 <= 3
-3 <= x2 <= 3
La variable x1 pasa a las variables x10, x11, x12
x12 x11 x10 = 000 el conversor QuadraticToQubo lo interpreta como -3 y no 0 en número enteros.

Con la función *.interpret()* de QuadraticToQubo nos permite que dado un array de numeros binarios, nos devuelve un array con los valores originales que representan.
Entonces para contar cuántas mediciones del Sampler obtenemos la cadena de 0's, necesitamos primero encontrar para qué valores binarias el QuadraticToQubo lo interpreta como 0.

6. Conclusiones y Post-Quantum Cryptography.

6.1. Conclusiones del VQA para el SVP.

En este proyecto no se pretende encontrar un algoritmo cuántico que permita reducir la complejidad en los problemas basados en retículos, sino que se ha aportado otra idea de algoritmos utilizando Computación Cuántica.
Como ya se ha mencionado al inicio de esta memoria, los problemas basados en retículos como el SVP, CVP y relacionados, son difíciles computacionalmente tanto en clásica como en cuántica [3].
Como hemos visto en las pruebas que hemos realizado, la eficacia del algoritmo variacional para resolver SVP depende mucho también de la base y también de la dimensión. Con una base de valores altos y en dimensiones mayores hace que necesitemos más qubits y como consecuencia más parámetros del Ansätze que optimizar clásicamente si queremos mantener la expresividad.
En estas circunstancias, el VQA no siempre nos devolverá el vector más corto, sino uno aproximado. Entonces, podemos utilizar este vector para sustituirlo de alguna manera por otro de la base original y volver a ejecutar el algoritmo
Por ello en [1] se muestra cómo se puede integrar el VQA en algoritmos de reducción de bases y que nos servirá también para el CVP.

Closest Vector Problem (CVP)

Dado una base $B \in \mathbb{R}^{d \times n}$ *y un vector* $t \in \mathbb{R}^{d}$ *queremos encontrar el vector* $v \in \mathcal{L}(B)$ *tal que:*

$$\|v - t\| = min\{\|x - t\| : x \in \mathcal{L}(B)\}$$

Podemos ver que el SVP es simplemente un caso particular del CVP y ambos son igual de difíciles computacionalmente.

6.2. Post-Quantum Cryptography.

Ya sabemos que en los recientes años, se ha avanzado significativamente en el desarrollo de la Computación Cuántica. Por tanto, la preocupación por que se vean comprometidos los sistemas de encriptados que utilizamos es cada vez mayor. El algoritmo de Shor hace peligrar los sistemas RSA y DSA cuando se desarrolle un computador cuántico lo suficientemente potente y la predicción ya no es tan lejana como se preveía en 1994.
Por tanto, las empresas y la comunidad científica desde hace un tiempo están investigando arduamente en otros problemas que sean computacionalmente difíciles para los ordenadores clásicos y también para los cuánticos. Se está investigando sobre nuevos estándares de encriptado de claves y de firmas digitales para poder migrar lo antes posible los sistemas actuales.
Este es el llamado Criptografía Post Cuántica.

Como ya se ha comentado a lo largo de esta memoria, los problemas más prometedores son los problemas basados en Retículos (o *Lattice-Based Problems*). Son problemas que a día de hoy se consideran computacionalmente duros tanto en clásica como en cuántica.

En 2017, el NIST (National Institute of Standards and Technology) de Estados Unidos lanzó una petición o "competición" por nuevos métodos de encriptado a la comunidad científica y empresarial con el fin de establecer un nuevo estándar. [6]
A día de hoy, en 2025, NIST ha seleccionado 5 sistemas que incluyen encriptado de claves como firmas digitales:

- Lattice-Based:
    - CRYSTALS-KYBER y HQC son para encriptado de claves.
    - CRYSTALS-DILITHIUM y FALCON son para firmas digitales.
- Hash-Based:
    - SPHINCS es para firmas digitales.

4 de ellos están basados en retículos y de ahí la gran importancia conocer sobre este tipo de problemas.

GGH Protocol [8]

Los sistemas de encriptación basados en retículos se están combinando la complejidad de estos problemas junto al paradigma del *Learning with errors*.

Entonces, una primera aproximación sencilla y general de estos sistemas de encriptado es el llamado Protocolo GGH (Goldreich-Goldwasser-Halevi).

Tenemos dos personas Alice y Bob que quieren comunicarse entre sí. Específicamente, Bob quiere enviarle un mensaje a Alice. Entonces, Alice tiene como clave privada una base de un retículo que es buena y la clave pública que tiene Bob y cualquier espía del canal Eve es una base "muy mala" del retículo.

Bob utiliza la base mala para codificar su mensaje en un punto del retículo y le añade un error pequeño (*learning with errors*) obteniendo un punto que no es del retículo pero que está cerca. Alice puede utilizar la base buena para decodificar fácilmente el mensaje encontrando el vector más próximo al que le envía Bob. La seguridad de este protocolo se basa en la alta complejidad computacional de calcular el vector más corto con una base mala en comparación con una base buena.

7. Referencias.

[1]. Martin R. Albrecht, Miloš Prokop, Yixin Shen, Petros Wallden. *Variational quantum solutions to the Shortest Vector Problem 2023*.

[2]. Daniele Micciancio. *Lattice Algorithms and Applications* . *Curso de 6 temas sobre la Teoría de Retículos en el cual se presenta toda gran parte de la matemática detrás de los Lattices y la demostración de los teoremas utilizados en esta memoria memoria.*

[3]. Shahed Sharif y Daniele Micciancio. *UCI Mathematics of Cryptography seminar notes*, *y Lattice-Based Cryptography*

[4]. *M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, y Patrick J. Coles. Variational quantum algorithms*

[5]. *"A Practical Guide to Quantum Machine Learning and Quantum Optimization"* - Elias F. Combarro.

[6]. NIST. "Post-quantum cryptography standardization".

[7]. *Documentación de las librerías de qiskit.*

Optimization converters

qiskit-ibm-runtime

Qiskit API

[8]. Ideas generales de sistemas de encriptados basados en retículos

GGH Protocol.

Learning with errors: Encrypting with unsolvable equations (vídeo de YouTube)

Lattice-based cryptography: The tricky math of dots (vídeo de YouTube)

Chris Peikert (profesor de la Universidad de Michigan) The Learning With Errors Problem and Cryptographic Applications (vídeo de YouTube)