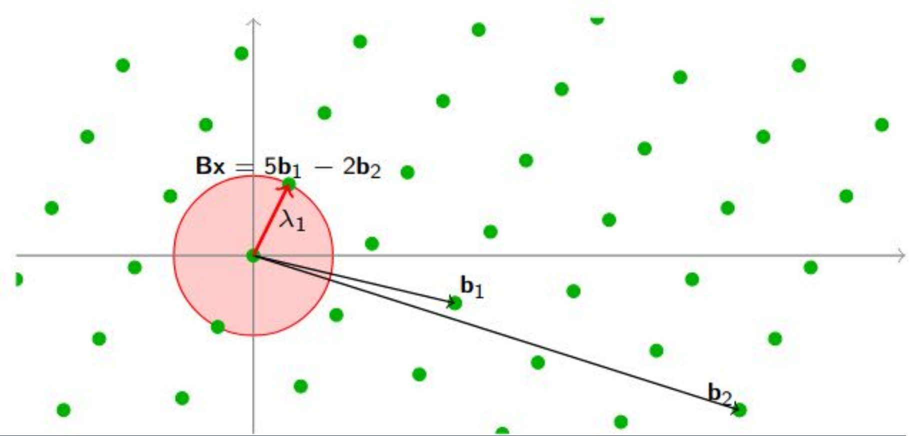
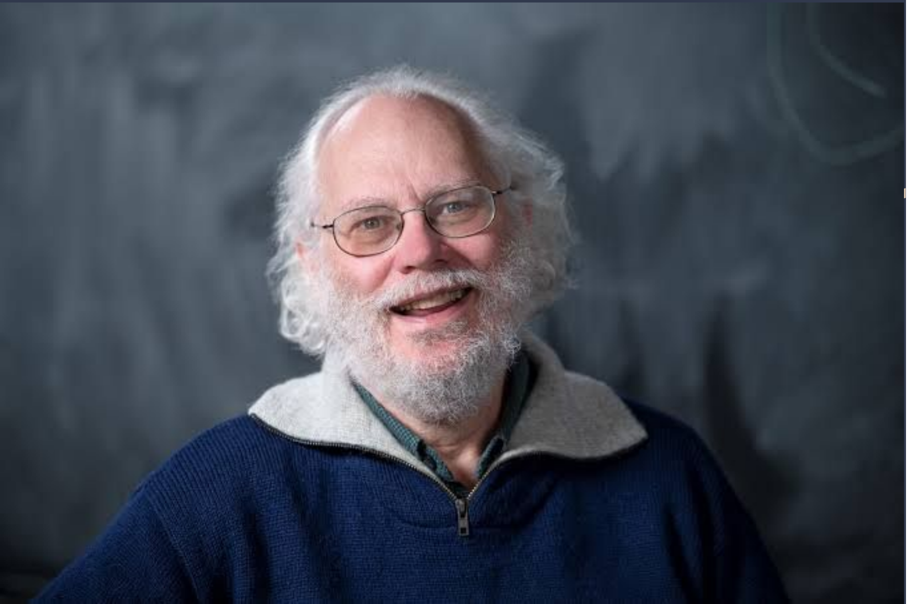
Algoritmos Variacionales Cuánticos para el Problema del Vector Más Corto (SVP)

Autor: Zhengkai Zhu

Índice

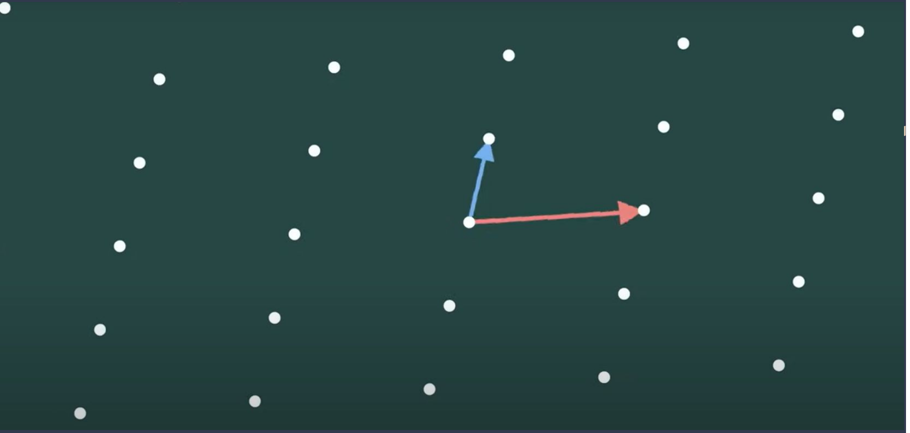


1. Introducción



- El algoritmo de Shor (1994) y aumento progresivo en escala de los computadores cuánticos hacen peligrar los sistemas de encriptado de hoy en día.
  - **Problema de factorización en primos.**
  - **Problema de logaritmo discreto.**
- Post-Quantum Cryptography.
  - **Problemas basado en Retículos**

2. Retículos (Lattices)

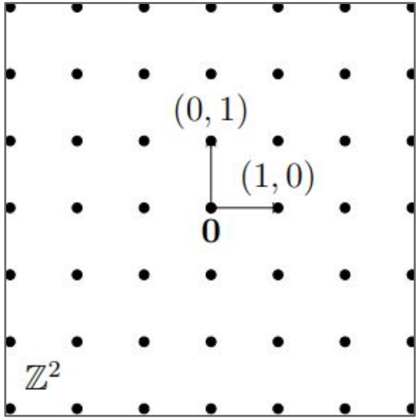


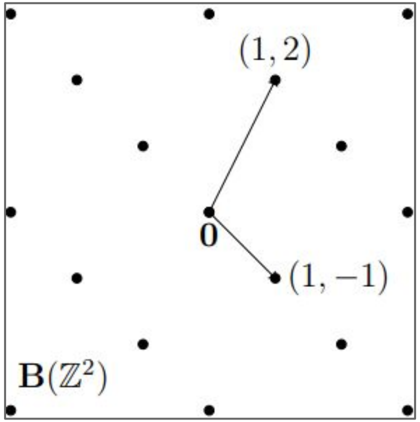
Dado una base de vectores:

$$B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$$

Un Retículo o Lattice es:

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



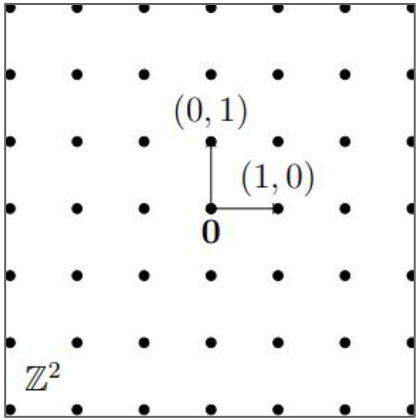


2. Retículos (Lattices)

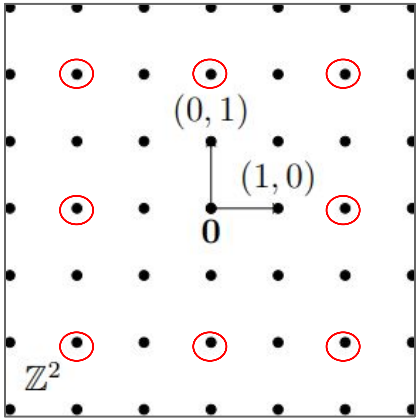
¿Qué es lo que hace interesante de los retículos?

- Dos bases B y B’ pese a ser bases de un subespacio $\mathbb{R}^n$ en general crean dos retículos diferentes.

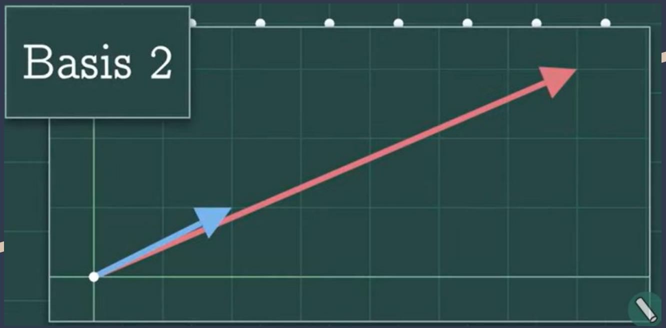
$$B = [(1,0),(0,1)]$$
$$\mathcal{L}(B) = \mathbb{Z}^2$$



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



2. Retículos (Lattices)



No obstante dos bases sí pueden generar el mismo retículo.

Ejemplo:

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

Ambos generan el mismo retículo

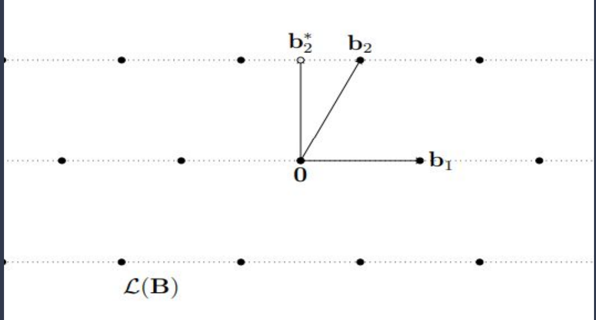
$$\mathcal{L}(B) = \mathcal{L}(B') = \mathbb{Z}^2$$

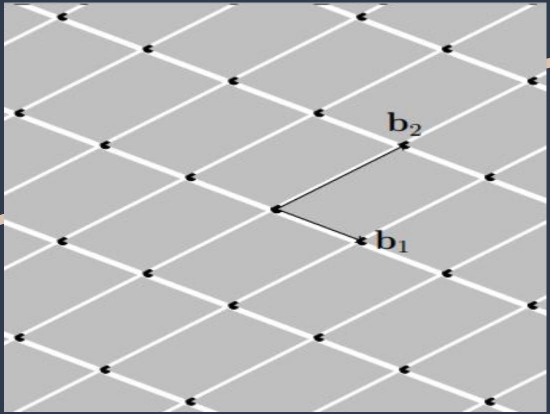
Con una base como B' es menos evidente cuál es el retículo que forma. Sus vectores tienen mayor módulo y forman entre ellos un ángulo muy cerrado o poco ortogonal. Decimos que B' es una **base mala**.

Existen teoremas y propiedades que demuestran cuándo dos bases generan el mismo retículo.

2. Retículos (Lattices)

Otras características





Gram-Schmidt orthogonalization

Dado una base B, su base ortogonal Gram-Schmidt es B* tal que:

$$B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d\times n}$$

$$B^* = [b_1^*, b_2^*, \ldots b_n^*] \quad b_i^* = b_i \perp [b_1, .. b_{i-1}]$$

Determinante o Volumen del retículo

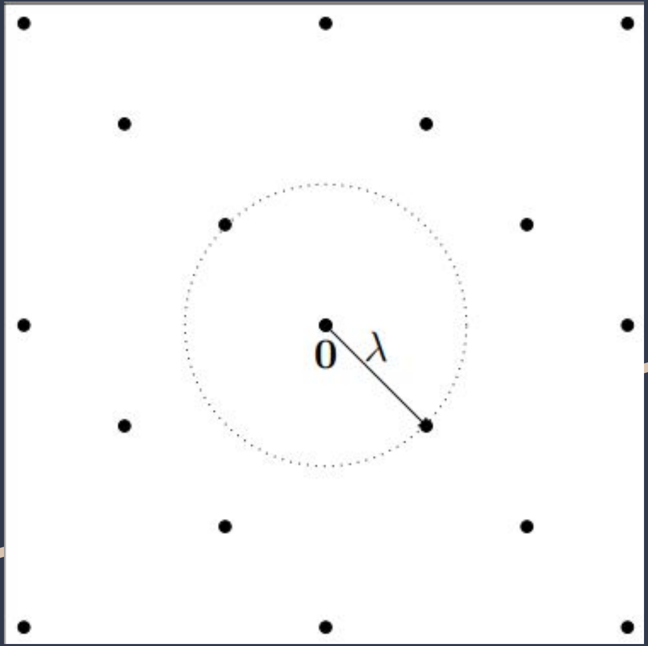
Volumen del paralelepípedo formado por los vectores de la base. Ese paralelepípedo también se llama Región Fundamental del retículo.

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

Una propiedad fundamental es que dos bases si forman el mismo retículo, tienen el mismo determinante.

$$det(B) = det(B') \Leftrightarrow \mathcal{L}(B) = \mathcal{L}(B')$$

2.1. Shortest Vector Problem (SVP)



Formulación:

"Dado una base $\boldsymbol{B}$ *de un retículo encontrar el vector distinto de 0 más corto del mismo según una norma (Euclidiana en general)"*

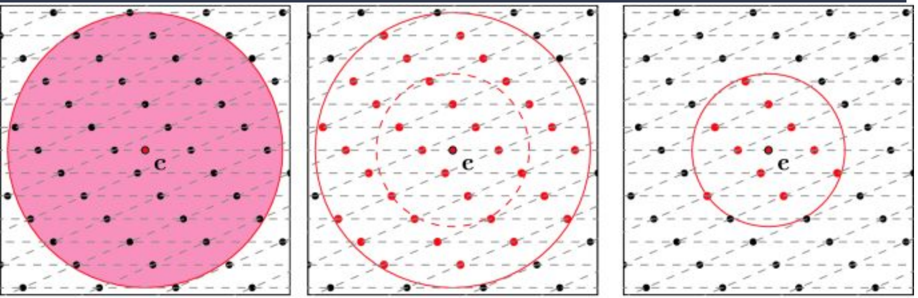
Es decir, dado $B \in \mathbb{R}^{d \times n}$

Encontrar el vector ***v*** tal que:

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

2.1. Shortest Vector Problem (SVP)

Algoritmos Clásicos



(a) $\mathcal{D}$ (b) $\mathcal{D}_{\{\mathcal{L}\cap\Omega'_c\}}$ (c) $\mathcal{D}_{\{\mathcal{L}\cap\Omega_c\}}$

Los algoritmos principales son los llamados Enumeration Algorithms:

- **Babai's Nearest Plane Algorithm** (Algoritmo Voraz): algoritmo aproximado
- **Branch and Bound**: algoritmo que devuelve el vector exacto.

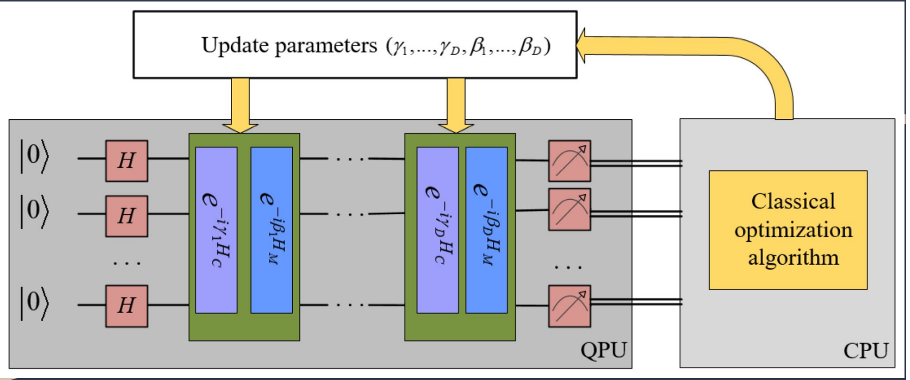
Estos algoritmos se basan en que teniendo una cota superior r tal que:

$$\lambda_1 \leq r$$

Buscamos en la Hiperesfera con radio r, que contenga al menos un vector del retículo. Así buscamos en un espacio acotado y finito.

La actuación de ambos y otros algoritmos depende mucho de la base B que tenemos. Una base mala hace que empeore el tiempo y también la solución.

3. Variational Quantum Algorithms (VQA)



Algoritmos que mezclan cuántica y clásica.

Permite resolver problemas de optimización mediante la búsqueda del estado de mínima energía de un Hamiltoniano.
La parte cuántica calcula el nivel de “energía” de un estado o media de todas las mediciones y luego tenemos un optimizador clásico que va encontrando el parámetro más óptimo para inicializar el estado inicial.

1. Seleccionar un estado Ansatz $|\psi(\theta)\rangle$
2. Codificamos nuestra función como un Hamiltoniano a través de una formulación QUBO.
3. La media de las mediciones obtenidas lo pasamos a un optimizado clásico

3. Variational Quantum Algorithms (VQA)

Construcción del Hamiltoniano

Queremos convertir la función que queremos minimizar del SVP en una función QUBO (Quadratic Unconstrained Binary Optimization)

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.

En SVP la función que queremos minimizar es:

Dado una base B

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

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

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

Cada xi sería una variable de la función.

3. Variational Quantum Algorithms (VQA)

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

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

$$G = B^T B$$

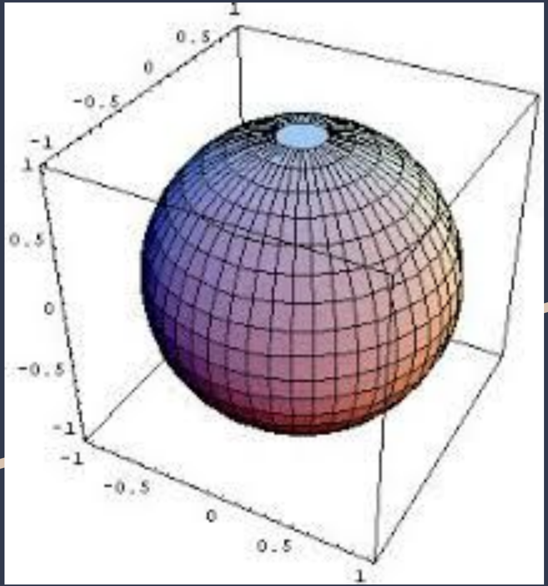
Problemas.

- Necesitamos que las variables estén acotadas para convertirlas en binario (Gaussian Heuristic).

- Necesitamos restringir el vector x = 0

3. Variational Quantum Algorithms (VQA)

Cota superior de la longitud mínima



Necesidad de acotar cada variable entera para reducir el espacio de búsqueda y fijar un número de qubits:

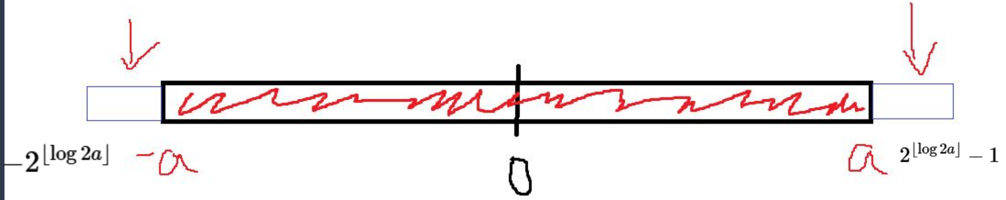
$$|x_i| \leq a_i \qquad x = (x_1, x_2, \ldots, x_i, \ldots, x_n)$$

Definimos nuevas variables binarias para cada variable entera $\{\tilde{x}_{ij}\}_{0 \leq j \leq \lfloor \log 2a \rfloor}$

Nos quedaría lo siguiente

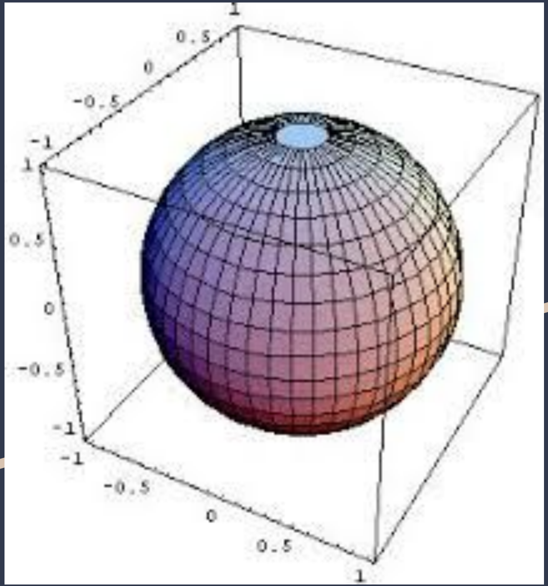
$$|x_i| \leq a \implies \tag{2}$$

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



3. Variational Quantum Algorithms (VQA)

Cota superior de la longitud mínima



Retículo Dual:

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

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

$$\Longrightarrow$$

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

Cota de Minkowski:

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

Heurística Gaussiana

$$vol(\mathbb{S}^n) \approx (\frac{2\pi e}{n})^{n/2} \qquad n >> 0$$

$$\Longrightarrow$$

$$\lambda_1 \leq\approx \sqrt{\frac{2n}{\pi e}} det(\mathcal{L}(B))^{1/n} \qquad n >> 0$$

3. Variational Quantum Algorithms (VQA)

El vector nulo

Para tratar el vector nulo tenemos dos formas:

- Añadirlo como restricción por lo que hacer que en el Hamiltoniano se penalice mucho cuando todos las variables binarias están a 0.

- También podemos añadir a la función de coste del Hamiltoniano un factor multiplicativo para que de energía infinita a estados que sea el vector 0 o en los cuales tenga mucha amplitud.

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

$$\downarrow$$

$$C(\psi) = \langle\psi|H|\psi\rangle \frac{1}{1-|\langle\psi|\psi_0\rangle|^2}$$

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

4. Implementaciones Sencillas

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

$$B = [(1,1),(2,-1)]$$

$$\|(2,-1)\| = \sqrt{(5)}$$

$$\|(1,1)\| = \sqrt{(2)}$$

Para mostrar un poco la forma de implementar el algoritmo, utilizo un ejemplo simple con una base en 2 dimensiones sencilla.

```
B = np.array([[1, 2],
              [1, -1]])
Bt = B.transpose()
print(Bt)

B_inv = np.linalg.inv(B)
print(B_inv)

B_dual = B_inv.transpose() #Es una matriz columna
print(B_dual)
```

```
[[ 1  1]
 [ 2 -1]]
[[ 0.33333333  0.66666667]
 [ 0.33333333 -0.33333333]]
[[ 0.33333333  0.33333333]
 [ 0.66666667 -0.33333333]]
```

```
G = Bt @ B

print(G)
```

```
[[2 1]
 [1 5]]
```

$$G = \begin{pmatrix} 2 & 1 \\ 1 & 5 \end{pmatrix}$$

```
vol = int(abs( np.linalg.det(B)))
print(vol)

circulo = np.pi

A = (2/np.sqrt(circulo)) * np.sqrt(vol) #Cota de Minkowski

print(A)

cotas = np.array([vol * (np.linalg.norm(B_dual.transpose()[i])) for i in range (n)]) #Calculo las cotas superiores de Minkowski

cotas_enteras = np.array([np.ceil(cotas[i]) for i in range (n)]) #Redondeo las cotas superiores de Minkowski

print(cotas)
print(cotas_enteras)
```

```
3
1.9544100476116795
[2.23606798 1.41421356]
[3. 2.]
```

4. Implementaciones Sencillas

```python
mdl = Model("Shortest vector problem")
x = mdl.integer_var_list(range(n), lb = -cotas_enteras, ub = cotas_enteras)

objective = mdl.sum([G[i][i] * x[i]**2 for i in range(n)])
objective += mdl.sum([G[i][j] * x[i] * x[j] for i in range(n) for j in range(i+1, n)])
mdl.minimize(objective)

#mdl.add_constraint( objective >= epsilon)
qp = from_docplex_mp(mdl)
```

```python
print(mdl.lp_string)
#mdl.solve()
qp = from_docplex_mp(mdl)
```

✓ 0.0s

```
\ This file has been generated by DOcplex
\ ENCODING=ISO-8859-1
\Problem name: Shortest vector problem

Minimize
 obj: [ 4 x_0^2 + 2 x_0*x_1 + 10 x_1^2 ]/2
Subject To

Bounds
 -3 <= x_0 <= 3
 -2 <= x_1 <= 2

Generals
 x_0 x_1
End
```

```python
from qiskit_ibm_runtime import QiskitRuntimeService, Session
from qiskit_ibm_runtime import EstimatorV2 as Estimator
from qiskit_ibm_runtime import SamplerV2 as Sampler

from qiskit_optimization.translators import ising
from qiskit_optimization.converters import QuadraticProgramToQubo
```

2] ✓ 2.1s

Generate + Code + Markdown

```python
service = QiskitRuntimeService(channel="local")

backend = service.least_busy()
backend.name
```

3] ✓ 0.6s

```
'fake_algiers'
```

```python
qubo_converter = QuadraticProgramToQubo()

qubo_problem = qubo_converter.convert(qp)
print(qubo_problem)

hamiltonian, offset = ising.to_ising(qubo_problem)
```

4] ✓ 0.0s

4. Implementaciones Sencillas

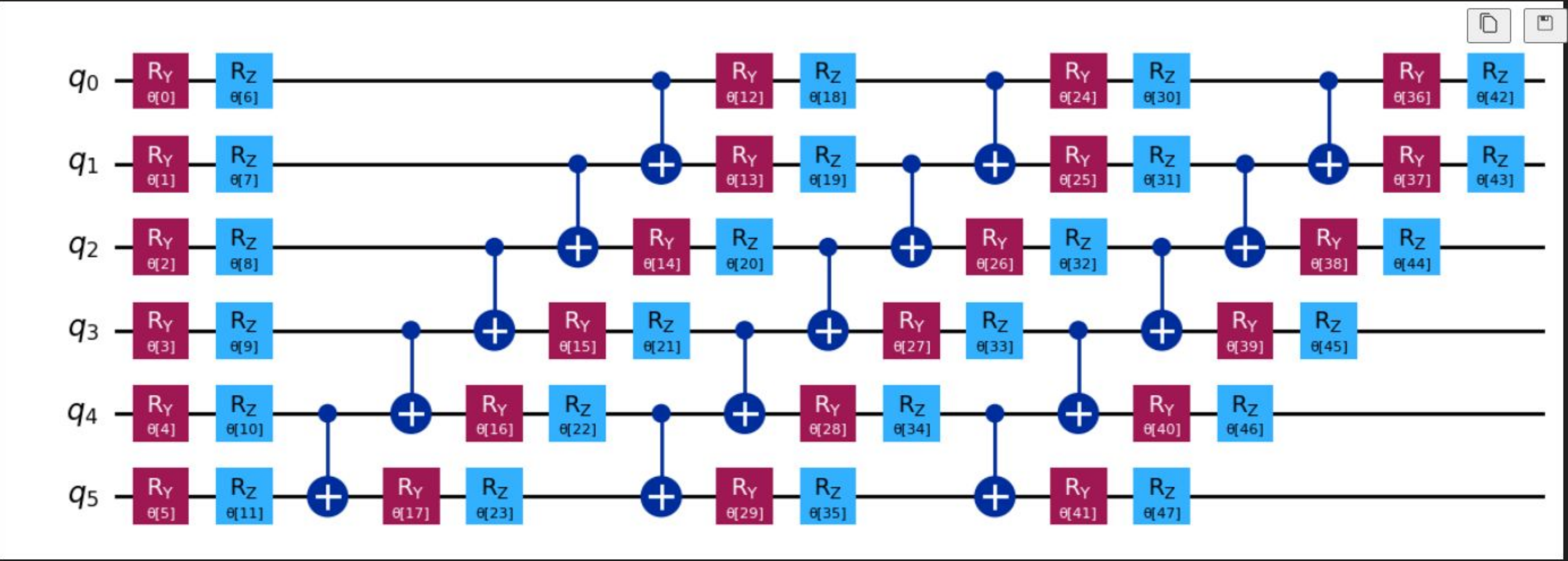
```
Problem name: Shortest vector problem

Minimize
  2*x_0@0^2 + 8*x_0@0*x_0@1 + 12*x_0@0*x_0@2 + x_0@0*x_1@0 + 2*x_0@0*x_1@1
  + x_0@0*x_1@2 + 8*x_0@1^2 + 24*x_0@1*x_0@2 + 2*x_0@1*x_1@0 + 4*x_0@1*x_1@1
  + 2*x_0@1*x_1@2 + 18*x_0@2^2 + 3*x_0@2*x_1@0 + 6*x_0@2*x_1@1 + 3*x_0@2*x_1@2
  + 5*x_1@0^2 + 20*x_1@0*x_1@1 + 10*x_1@0*x_1@2 + 20*x_1@1^2 + 20*x_1@1*x_1@2
  + 5*x_1@2^2 - 14*x_0@0 - 28*x_0@1 - 42*x_0@2 - 23*x_1@0 - 46*x_1@1 - 23*x_1@2
  + 44

Subject to
  No constraints

  Binary variables (6)
    x_0@0 x_0@1 x_0@2 x_1@0 x_1@1 x_1@2
```

4. Implementaciones Sencillas

Aquí definimos un circuito cuántico parametrizado donde codificaré más adelante el estado inicial ansätz.

```
ansatz = EfficientSU2(hamiltonian.num_qubits)
ansatz.decompose().draw("mpl", style="iqp", fold = -1)
```

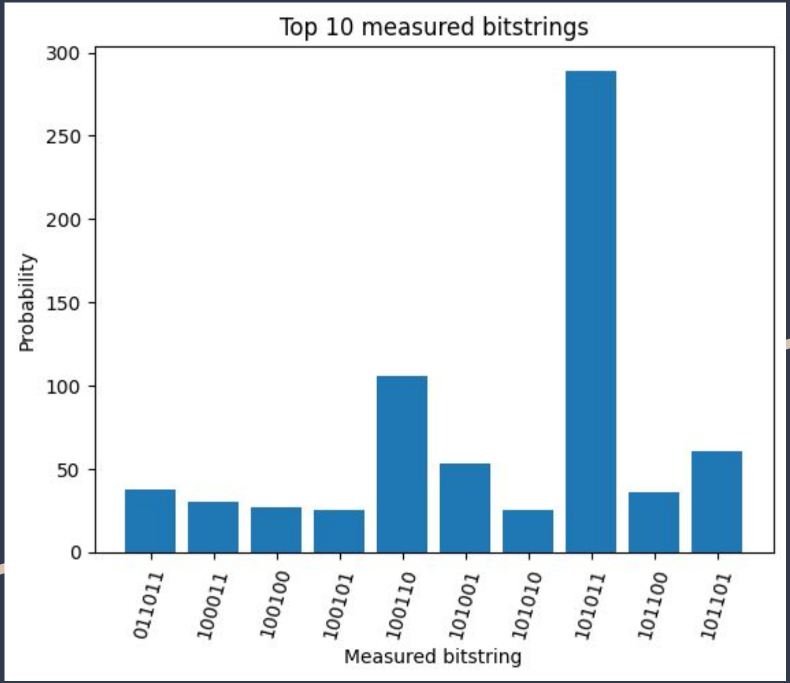
Pytho

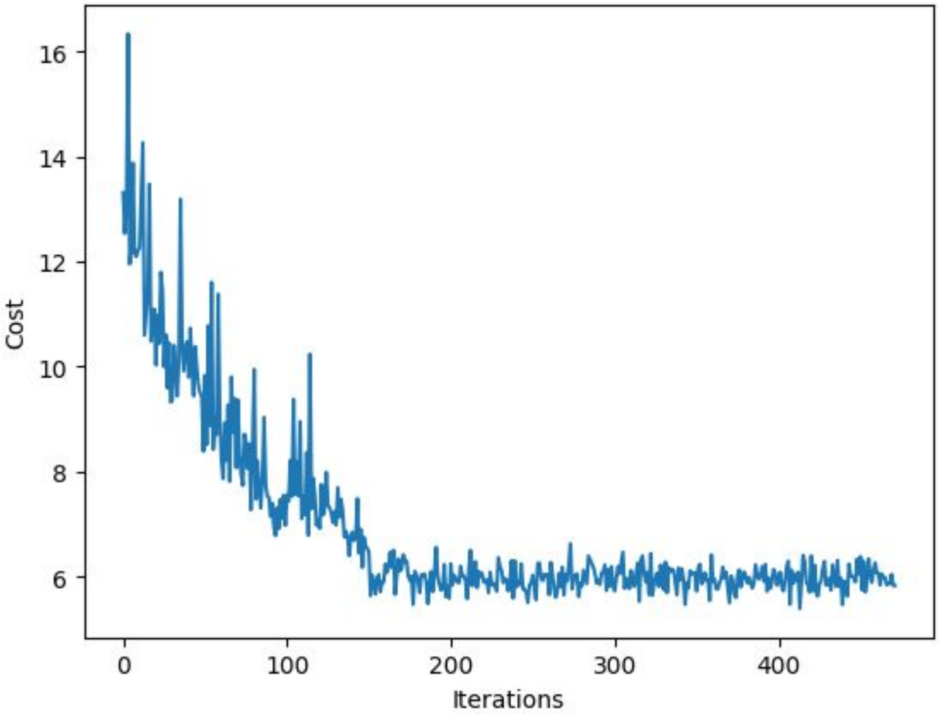


4. Implementaciones Sencillas

```python
def cost_func(params, ansatz, ansatz_meas, hamiltonian, estimator, sampler):

    pub = (ansatz, [hamiltonian], [params])

    result = estimator.run(pubs=[pub]).result()

    energy = result[0].data.evs[0] + offset #Aquí obtenemos un valor inicial de la energía, sin la restricción del vector 0


    pub1 = (ansatz_meas, [params])
    job = sampler.run(pubs = [pub1])
    nresult = job.result()
    for item in nresult._pub_results[0].data.items():
        counts = item[1].get_counts()
    n = sum(count for bitstring, count in counts.items() if not set(bitstring) <= {'0'})
    N = sum(counts.values()) #Es el numero de shots


    if n == 0:
        return np.inf
    else:
        final_energy = (N/n)*energy
        return final_energy
```

4. Implementaciones Sencillas





4. Implementaciones Sencillas

En el gráfico podemos ver que al final el módulo en el cual se queda atascado y que considera como mínimo es uno cercano a 5

Finalmente el vector $x \in \mathbb{Z}^2$ es $(-1, 1)$. Entonces al calcular el punto del retículo al que pertenece es la siguiente:

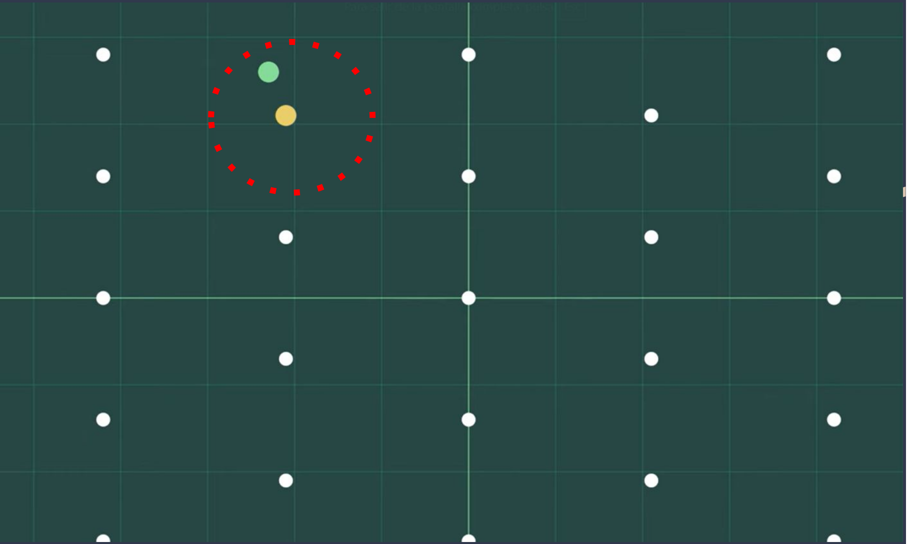
$$\begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & -2 \end{pmatrix}$$

El vector que hemos obtenido es un vector corto, pero no es el más corto.

Razones:

- Elección de parámetros iniciales.
- Circuito ansatze seleccionado es “*hardware efficient*”.

5. Conclusiones y Criptografía Post-Cuántica



Conclusiones:

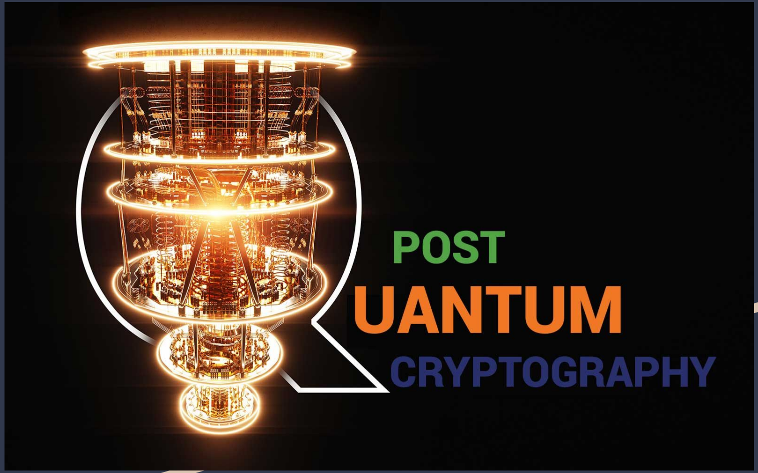
- Para dimensiones altas la densidad de puntos de retículos es mayor. El algoritmo de VQE puede no converger en un algoritmo vector más corto, sino en uno cerca de serlo.
- El algoritmo se puede combinar con otros algoritmos para mejorar bases
- Se puede utilizar como subrutina para resolver el CVP

Closest Vector Problem (CVP)

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

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

5. Conclusiones y Criptografía Post-Cuántica



<u>Post-Quantum Cryptography</u>

- Peligro de romper RSA y DSA en años venideros.
- Necesidad de buscar nuevas maneras de encriptar claves y firmas digitales (en clásica) mediante problemas duros también para computadores cuánticos.
- **<u>Lattice Based + Learning with errors.</u>**

<u>National Institute of Standards and Technology</u>

- En 2017 lanza petición de propuestas de nuevos métodos de encriptado de clave y de firma digital
- Hoy en 2025 se han seleccionado 5:
  - Lattice Based:
    - CRYSTALS-KYBER: Key Encryption
    - HQC: Key Encryption
    - CRYSTALS-DILITHIUM: Digital Signature.
    - FALCON: Digital Signature
  - Hash Based:
    - SPHINCS: Digital Signature.

6. Referencias

- “Variational Quantum Solutions to the Shortest Vector Problem”: https://arxiv.org/abs/2202.06757
- “UCI Mathematics of Cryptography seminar notes”: https://public.csusm.edu/ssharif/crypto/
- “A Practical Guide to Quantum Machine Learning and Quantum Optimization” - Elias F. Combarro.
- “Lattice Algorithms and Applications” seminario de Daniele Miccielano: https://cseweb.ucsd.edu/classes/sp14/cse206A-a/
- “Lattice Based Cryptography”: https://www.youtube.com/watch?v=QDdOoYdb748
- “Learning with errors”: https://www.youtube.com/watch?v=K026C5YaB3A https://www.youtube.com/watch?v=K_fNK04yG4o