Quantum algorithms for lattice problems

Yixin Shen

King's College London

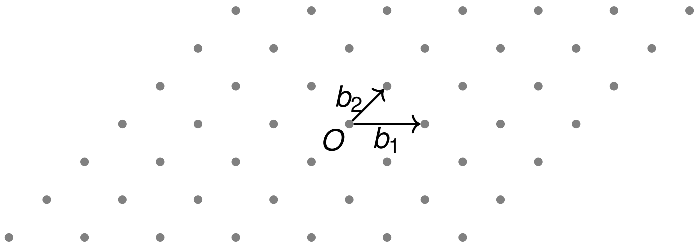
October 16, 2023

Outline

What is a (Euclidean) lattice?

Definition

$\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \left\{\sum_{i=1}^{n} x_i \boldsymbol{b}_i : x_i \in \mathbb{Z}\right\}$ where $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is a basis of $\mathbb{R}^n$.

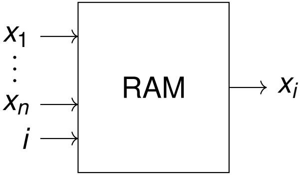


Quantum memory models

classical access

quantum access

classical data

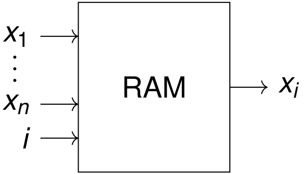
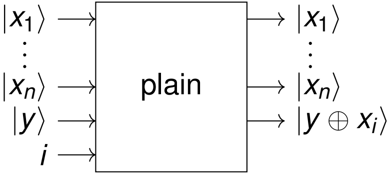


standard

quantum data

Assumption: $O(1)$ time cost

Quantum memory models

	classical access	quantum access
classical data	 standard	
quantum data	 standard	

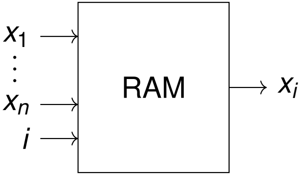
Assumption: $O(1)$ time cost

Quantum memory models

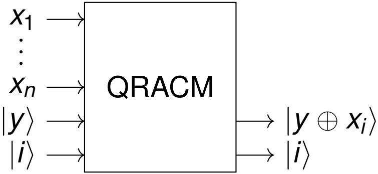
classical access

quantum access

classical data

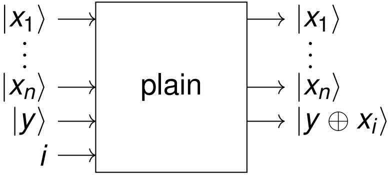


standard



potentially strong assumption

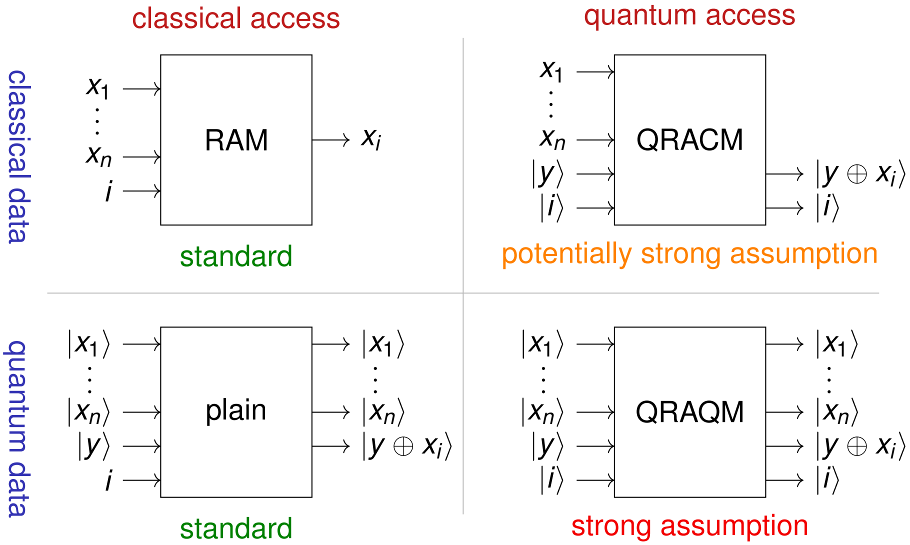
quantum data



standard

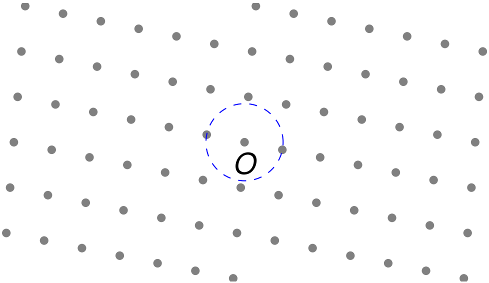
Assumption: $O(1)$ time cost

Quantum memory models



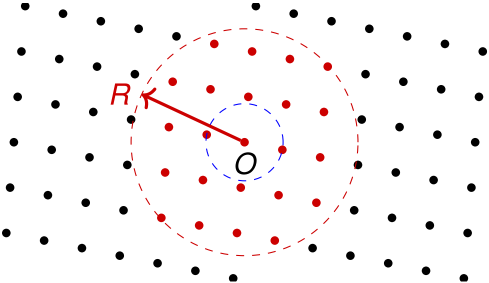
Assumption: $O(1)$ time cost

Shortest Vector Problem (SVP)



Shortest Vector Problem (SVP): given a basis of a lattice, find a shortest nonzero vector.

Shortest Vector Problem (SVP)



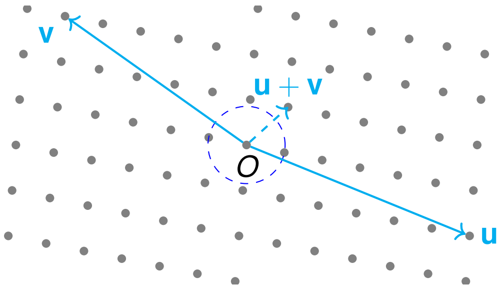
Shortest Vector Problem (SVP): given a basis of a lattice, find a shortest nonzero vector.

Two main approaches:

Approach: enumeration

1. choose a radius R
2. enumerate all vectors of length smaller than R
3. keep the shortest one

Shortest Vector Problem (SVP)



Shortest Vector Problem (SVP): given a basis of a lattice, find a shortest nonzero vector.

Two main approaches:

Approach: enumeration

1. choose a radius R
2. enumerate all vectors of length smaller than R
3. keep the shortest one

Approach: sieving

1. generate a lot of random vectors
2. combine them recursively to reduce their length

Enumeration = tree exploration

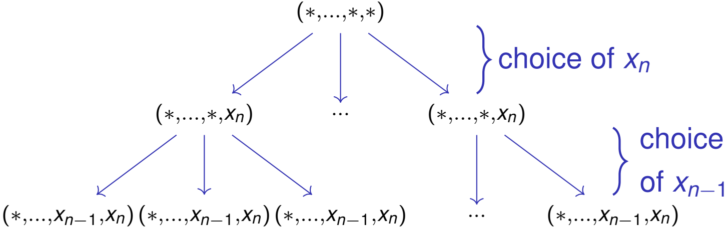
Enumerate all $X = x_1 b_1 + \cdots + x_n b_n$ such that $\|X\| \leqslant R$:

$$
\begin{tikzcd}
 & (*,\dots,*,*) \arrow[dl] \arrow[d] \arrow[dr] & \\
(*,\dots,*,x_n) & \cdots & (*,\dots,*,x_n)
\end{tikzcd}
$$

choice of x_n

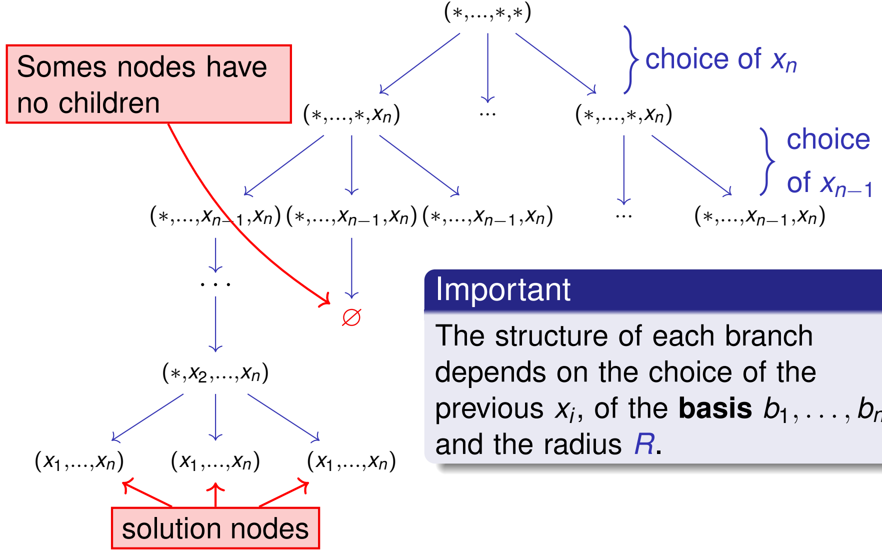
Enumeration = tree exploration

Enumerate all $X = x_1 b_1 + \cdots + x_n b_n$ such that $\|X\| \leqslant R$:



Enumeration = tree exploration

Enumerate all $X = x_1 b_1 + \cdots + x_n b_n$ such that $\|X\| \leqslant R$:



Important

The structure of each branch depends on the choice of the previous x_i, of the **basis** $b_1, \ldots, b_n$ and the radius R.

Enumeration and quantum

Many variants of enumeration to reduce the size of the tree:

- cylindrical pruning [GNR10]
- discrete pruning [AN17]
- extreme pruning [GNR10]

$\rightsquigarrow$ can all be seen as searching for marked nodes in a tree

Enumeration and quantum

Many variants of enumeration to reduce the size of the tree:

- cylindrical pruning [GNR10]
- discrete pruning [AN17]
- extreme pruning [GNR10]

$\rightsquigarrow$ can all be seen as searching for marked nodes in a tree

Quantum backtracking [Montanaro15]

Assume black-box access to tree nodes

- requests give the local tree structure only

$\tilde{O}(\sqrt{T})$ requests to find a solution node (tree with T nodes)

Can also estimate the size of a tree with a quadratic speed-up [AK17]

Enumeration and quantum

Many variants of enumeration to reduce the size of the tree:

- cylindrical pruning [GNR10]
- discrete pruning [AN17]
- extreme pruning [GNR10]

$\rightsquigarrow$ can all be seen as searching for marked nodes in a tree

Quantum backtracking [Montanaro15]

Assume black-box access to tree nodes

- requests give the local tree structure only

$\tilde{O}(\sqrt{T})$ requests to find a solution node (tree with T nodes)

Can also estimate the size of a tree with a quadratic speed-up [AK17]

Quantum acceleration [ANS18]

Quadratic quantum speed-up on all variants of enumeration

Complexity: super-exponential time but polynomial number of qubits

Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
    - Sieve vectors

Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
  - Sieve vectors

Sieve (parameter $\gamma < 1$):
Input: many vectors of length $\leqslant \ell$
Output: many vectors of length $\leqslant \gamma\ell$

Combine pairs of vectors to produce shorter vectors

Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
  - Sieve vectors

Sieve (parameter $\gamma < 1$):

Input: many vectors of length $\leqslant \ell$

Output: many vectors of length $\leqslant \gamma\ell$

Combine pairs of vectors to produce shorter vectors

Idea: LLL reduced $\rightsquigarrow \ell \leqslant 2^{O(n)}\lambda_1$, sieve $O(n \log \frac{1}{\gamma})$ times, solve SVP

Heuristic: at each stage, vectors are uniformly distributed of length ℓ

Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
  - Sieve vectors

Sieve (parameter $\gamma < 1$):
Input: many vectors of length $\leqslant \ell$
Output: many vectors of length $\leqslant \gamma\ell$

Combine pairs of vectors to produce shorter vectors

Idea: LLL reduced $\rightsquigarrow \ell \leqslant 2^{O(n)}\lambda_1$, sieve $O(n \log \frac{1}{\gamma})$ times, solve SVP

Heuristic: at each stage, vectors are uniformly distributed of length ℓ

Avoid testing all pairs of vectors: locality sensitive filtering [BDGL15]:

- partition vectors into "buckets" (e.g. quarters, cones)
- two vectors in the same bucket are more likely to be "close"
- quantum: use Grover in each bucket [Laarhoven16]

Collision Finding

We can view sieving as finding pairs of vectors with common attributes
$\rightsquigarrow$ collision finding (e.g. find two vectors in the same bucket)

Collision Finding

We can view sieving as finding pairs of vectors with common attributes $\rightsquigarrow$ collision finding (e.g. find two vectors in the same bucket)

Collision Finding

Given random $f : \{0,1\}^n \to \{0,1\}^m$, $n \leq m \leq 2n$, find 2^k collision pairs, where $k \leq 2n - m$.

Extensively studied in the classical case.

Collision Finding

We can view sieving as finding pairs of vectors with common attributes $\rightsquigarrow$ collision finding (e.g. find two vectors in the same bucket)

Collision Finding

Given random $f : \{0,1\}^n \rightarrow \{0,1\}^m, n \leq m \leq 2n$, find 2^k collision pairs, where $k \leq 2n - m$.

Extensively studied in the classical case. Several quantum algorithms:

- BHT algorithm based on Grover search (+QRACM)
- Algorithms based on quantum walks [Ambainis03] [BCS**S**23]

Collision Finding

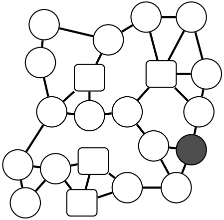
We can view sieving as finding pairs of vectors with common attributes $\rightsquigarrow$ collision finding (e.g. find two vectors in the same bucket)

Collision Finding

Given random $f : \{0,1\}^n \to \{0,1\}^m, n \le m \le 2n$, find 2^k collision pairs, where $k \le 2n - m$.

Extensively studied in the classical case. Several quantum algorithms:

- BHT algorithm based on Grover search (+QRACM)
- Algorithms based on quantum walks [Ambainis03] [BCS**S**23]



Classical walk:

- graph: search space
- marked nodes: solutions

Start anywhere, move to random neighbors until we find a marked vertex

Collision Finding

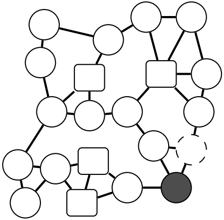
We can view sieving as finding pairs of vectors with common attributes $\leadsto$ collision finding (e.g. find two vectors in the same bucket)

Collision Finding

Given random $f : \{0,1\}^n \to \{0,1\}^m, n \leq m \leq 2n$, find 2^k collision pairs, where $k \leq 2n - m$.

Extensively studied in the classical case. Several quantum algorithms:

- BHT algorithm based on Grover search (+QRACM)
- Algorithms based on quantum walks [Ambainis03] [BCS**S**23]



Classical walk:

- graph: search space
- marked nodes: solutions

Start anywhere, move to random neighbors until we find a marked vertex

Collision Finding

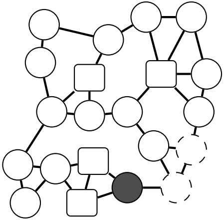
We can view sieving as finding pairs of vectors with common attributes $\rightsquigarrow$ collision finding (e.g. find two vectors in the same bucket)

Collision Finding

Given random $f : \{0,1\}^n \to \{0,1\}^m, n \leq m \leq 2n$, find 2^k collision pairs, where $k \leq 2n - m$.

Extensively studied in the classical case. Several quantum algorithms:

- BHT algorithm based on Grover search (+QRACM)
- Algorithms based on quantum walks [Ambainis03] [BCS**S**23]



Classical walk:

- graph: search space
- marked nodes: solutions

Start anywhere, move to random neighbors until we find a marked vertex

Collision Finding

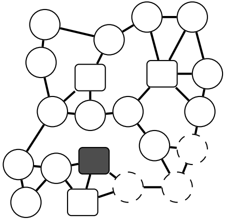
We can view sieving as finding pairs of vectors with common attributes
$\rightsquigarrow$ collision finding (e.g. find two vectors in the same bucket)

Collision Finding

Given random $f : \{0,1\}^n \to \{0,1\}^m, n \leq m \leq 2n$, find 2^k collision pairs, where $k \leq 2n - m$.

Extensively studied in the classical case. Several quantum algorithms:

- BHT algorithm based on Grover search (+QRACM)
- Algorithms based on quantum walks [Ambainis03] [BCS**S**23]



Classical walk:

- graph: search space
- marked nodes: solutions

Start anywhere, move to random neighbors until we find a marked vertex

Classical and quantum walks

Classical framework

- Setup a starting arbitrary vertex (S)
- Move from one vertex to one of its neighbors (U)
- Check if a vertex is marked (C)

We will find a marked vertex in time:

$$\mathsf{S} + \underbrace{\frac{1}{\epsilon}}_{\text{Walk steps}} \left(\underbrace{\frac{1}{\delta}}_{\text{Mixing time}} \mathsf{U} + \mathsf{C} \right)$$

where

- ϵ: proportion of marked vertices
- δ: spectral gap of the graph (number of updates before we reach a new uniformly random vertex)

Classical and quantum walks

MNRS framework

- Setup creates a superposition over all vertices (S)
- Move from one vertex to one of its neighbors (U)
- Check if a vertex is marked (C)

We will find a marked vertex in quantum time:

$$\mathsf{S} + \underbrace{\sqrt{\frac{1}{\epsilon}}}_{\text{Walk steps}} \left(\underbrace{\sqrt{\frac{1}{\delta}}}_{\text{Mixing time}} \mathsf{U} + \mathsf{C} \right)$$

where

- ϵ: proportion of marked vertices
- δ: spectral gap of the graph (number of updates before we reach a new uniformly random vertex)

⚠ Requires a QRAQM (strongest quantum RAM model)

Classical and quantum walks

MNRS framework

- Setup creates a superposition over all vertices (S)
- Move from one vertex to one of its neighbors (U)
- Check if a vertex is marked (C)

We will find k marked vertex in quantum time[CL21]:

$$k\left(\mathsf{S} + \underbrace{\sqrt{\frac{1}{\epsilon}}}_{\text{Walk steps}}\left(\underbrace{\sqrt{\frac{1}{\delta}}}_{\text{Mixing time}}\mathsf{U} + \mathsf{C}\right)\right)$$

where

- ϵ: proportion of marked vertices
- δ: spectral gap of the graph (number of updates before we reach a new uniformly random vertex)

⚠ Requires a QRAQM (strongest quantum RAM model)

Classical and quantum walks

MNRS framework

- Setup creates a superposition over all vertices (S)
- Move from one vertex to one of its neighbors (U)
- Check if a vertex is marked (C)

We will find *k* marked vertex in quantum time[BCS**S**23]:

$$\mathsf{S} + k \underbrace{\sqrt{\frac{1}{\epsilon}}}_{\text{Walk steps}} \left(\underbrace{\sqrt{\frac{1}{\delta}}}_{\text{Mixing time}} \mathsf{U} + \mathsf{C} \right)$$

where

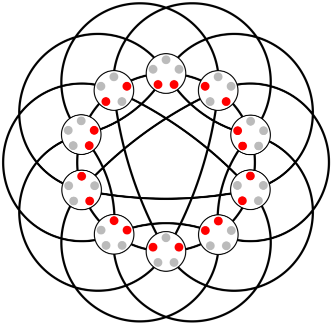
- ϵ: proportion of marked vertices
- δ: spectral gap of the graph (number of updates before we reach a new uniformly random vertex)

⚠ Requires a QRAQM (strongest quantum RAM model)

Example: Walk-based collision finding

Definition (Johnson graph)

- Nodes are sets of k elements among n ($k \ll n$)
- N_1 and N_2 are adjacents if $|N_1 \cap N_2| = k - 1$
- $\frac{1}{\delta} = \frac{k(n-k)}{n} \simeq k$ (We need to replace all elements.)



$k = 2, n = 5$

Example: Walk-based collision finding

Definition (Johnson graph)

- Nodes are sets of k elements among n ($k \ll n$)
- N_1 and N_2 are adjacents if $|N_1 \cap N_2| = k - 1$
- $\frac{1}{\delta} = \frac{k(n-k)}{n} \simeq k$ (We need to replace all elements.)

Collision finding with Johnson graph

- Create a random list of elements of size $k = 2^r$
- Repeat until a collision is found:
  - Walk 2^r times
  - Check whether the node contains a collision

Example: Walk-based collision finding

Definition (Johnson graph)

- Nodes are sets of k elements among n ($k \ll n$)
- N_1 and N_2 are adjacents if $|N_1 \cap N_2| = k - 1$
- $\frac{1}{\delta} = \frac{k(n-k)}{n} \simeq k$ (We need to replace all elements.)

Collision finding with Johnson graph

- Create a random list of elements of size $k = 2^r$
- Repeat until a collision is found:
  - Walk 2^r times
  - Check whether the node contains a collision

Classical complexity

$$2^r + \frac{1}{2^{2r-m}}(2^r \times 1 + 1) \approx \max(2^r, 2^{m-r}) \quad \rightsquigarrow \quad \text{optimal for } r = m/2$$

Example: Walk-based collision finding

Definition (Johnson graph)

- Nodes are sets of k elements among n ($k \ll n$)
- N_1 and N_2 are adjacents if $|N_1 \cap N_2| = k - 1$
- $\frac{1}{\delta} = \frac{k(n-k)}{n} \simeq k$ (We need to replace all elements.)

Collision finding with Johnson graph

- Create a random list of elements of size $k = 2^r$
- Repeat until a collision is found:
  - Walk 2^r times
  - Check whether the node contains a collision

Quantum complexity

$$2^r + \sqrt{\frac{1}{2^{2r-m}}}\left(\sqrt{2^r \times 1} + 1\right) \approx \max(2^r, 2^{(m-r)/2}) \rightsquigarrow \text{ optimal for } r = m/3$$

Back to sieving

- Locality sensitive filtering + quantum collision finding
  Exponential time and size QRACM (Grover)/QRAQM (walks)

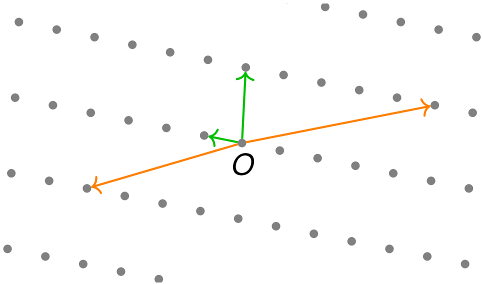
Back to sieving

- Locality sensitive filtering + quantum collision finding

  Exponential time and size QRACM (Grover)/QRAQM (walks)
- Tuple sieve [BLS16,HK17,HKL18,KMPR19,CL23]

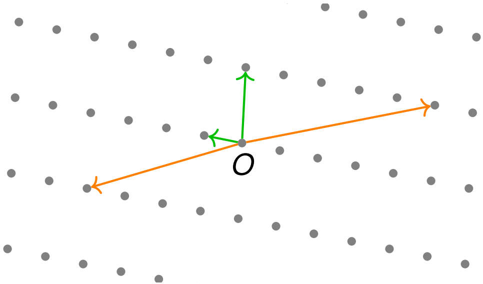
  Sieve k vectors instead of pairs, look for “configurations” satisfying certain properties
  Use quantum amplitude amplification to find tuples that satisfy the configuration.

Lattice reduction algorithms



- good basis: short and orthogonal*ish* vectors, makes problem easy
- bad basis: long and parallel*ish* vectors, makes problem hard

Lattice reduction algorithms



- good basis: short and orthogonal*ish* vectors, makes problem easy
- bad basis: long and parallel*ish* vectors, makes problem hard

Basis reduction: transform a bad basis into a good one
Algorithms: LLL, BKZ and its variants

Strong lattice reduction: BKZ algorithm

$$\begin{pmatrix} \overbrace{\boxed{\mathbf{b}_1 \quad \mathbf{b}_2 \quad \mathbf{b}_3 \quad \mathbf{b}_4 \quad \mathbf{b}_5}}^{\text{block size } \beta = 5} & \mathbf{b}_6 & \mathbf{b}_7 & \cdots \end{pmatrix}$$

Strong lattice reduction: BKZ algorithm

$$\left(\begin{array}{cccccccc} \multicolumn{5}{c}{\overbrace{\hspace{20em}}^{\text{block size } \beta = 5}} & & & \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \mathbf{b}_4 & \mathbf{b}_5 & \mathbf{b}_6 & \mathbf{b}_7 & \cdots \\ \multicolumn{5}{c}{\mathbf{x} \leftarrow \mathtt{SVP}(\mathbf{b}_1, \ldots, \mathbf{b}_5)} & & & \end{array}\right)$$

- solve SVP for the block

Strong lattice reduction: BKZ algorithm

$$\left(\overbrace{\boxed{\mathbf{b}'_1 \quad \mathbf{b}'_2 \quad \mathbf{b}'_3 \quad \mathbf{b}'_4 \quad \mathbf{b}'_5}}^{\text{block size } \beta = 5} \quad \mathbf{b}_6 \quad \mathbf{b}_7 \quad \cdots \right)$$

$$\mathbf{x} \leftarrow \texttt{SVP}(\mathbf{b_1}, \ldots, \mathbf{b}_5)$$

$$(\mathbf{b_1}', \ldots, \mathbf{b}'_5) \leftarrow \texttt{LLL}(\mathbf{b_1}, \ldots, \mathbf{b}_5, \mathbf{x})$$

- solve SVP for the block
- apply LLL to block + SVP
- replace by reduced basis

Strong lattice reduction: BKZ algorithm

$$\left(\begin{array}{ccc} \mathbf{b}'_1 & \overbrace{\boxed{\pi_1(\mathbf{b}'_2)\ \pi_1(\mathbf{b}'_3)\ \pi_1(\mathbf{b}'_4)\ \pi_1(\mathbf{b}'_5)\ \pi_1(\mathbf{b}'_6)}}^{\text{block size } \beta = 5} & \mathbf{b}_7 \quad \cdots \end{array} \right)$$

$\pi_i(\mathbf{v})$: project $\mathbf{v}$ orthogonally to $\mathbf{b}_1, \ldots, \mathbf{b}_i$

- For each block:
  - project block
  - solve SVP for the block
  - apply LLL to block + SVP
  - replace by reduced basis

Strong lattice reduction: BKZ algorithm

$$\left(\begin{array}{ccc} \mathbf{b}'_1 & \overbrace{\boxed{\pi_1(\mathbf{b}'_2)\ \pi_1(\mathbf{b}'_3)\ \pi_1(\mathbf{b}'_4)\ \pi_1(\mathbf{b}'_5)\ \pi_1(\mathbf{b}'_6)}}^{\text{block size } \beta = 5} & \mathbf{b}_7 \quad \cdots \end{array} \right)$$

$\pi_i(\mathbf{v})$: project $\mathbf{v}$ orthogonally to $\mathbf{b}_1, \ldots, \mathbf{b}_i$

- For each block:
  - project block
  - solve SVP for the block
  - apply LLL to block + SVP
  - replace by reduced basis
- Repeat until basis is reduced

Strong lattice reduction: BKZ algorithm

$$\left(\quad \mathbf{b}'_1 \quad \overbrace{\boxed{\pi_1(\mathbf{b}'_2)\ \pi_1(\mathbf{b}'_3)\ \pi_1(\mathbf{b}'_4)\ \pi_1(\mathbf{b}'_5)\ \pi_1(\mathbf{b}'_6)}}^{\text{block size } \beta = 5} \quad \mathbf{b}_7 \quad \cdots \quad \right)$$

$\pi_i(\mathbf{v})$: project $\mathbf{v}$ orthogonally to $\mathbf{b}_1, \ldots, \mathbf{b}_i$

- For each block:
  - project block
  - solve SVP for the block
  - apply LLL to block + SVP
  - replace by reduced basis
- Repeat until basis is reduced

Key elements:

- bigger $\beta \rightsquigarrow$ more expensive, better reduction, smaller $\mathbf{b}_1$
- very complex behaviour
- quantum BKZ: use a quantum SVP oracle

Learning with errors (LWE)

Let $n = 4$, $m = 6$ and $q = 17$.

secret

$$A \in \mathbb{Z}_q^{m \times n} \qquad s \in \mathbb{Z}_q^{n} \qquad b \in \mathbb{Z}_q^{m}$$

$$\begin{bmatrix} 14 & 12 & 2 & 5 \\ 5 & 3 & 1 & 7 \\ 14 & 7 & 2 & 5 \\ 0 & 9 & 8 & 4 \\ 8 & 11 & 5 & 12 \\ 5 & 1 & 3 & 14 \end{bmatrix} \times \begin{bmatrix} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \\ 14 \\ 6 \\ 12 \\ 13 \end{bmatrix}$$

Given A and b, find s

Learning with errors (LWE)

Let $n = 4$, $m = 6$ and $q = 17$.

secret

$$\underset{A \in \mathbb{Z}_q^{m \times n}}{\begin{bmatrix} 14 & 12 & 2 & 5 \\ 5 & 3 & 1 & 7 \\ 14 & 7 & 2 & 5 \\ 0 & 9 & 8 & 4 \\ 8 & 11 & 5 & 12 \\ 5 & 1 & 3 & 14 \end{bmatrix}} \times \underset{s \in \mathbb{Z}_q^{n}}{\begin{bmatrix} 1 \\ 2 \\ 1 \\ 5 \end{bmatrix}} = \underset{b \in \mathbb{Z}_q^{m}}{\begin{bmatrix} 11 \\ 5 \\ 14 \\ 6 \\ 12 \\ 13 \end{bmatrix}}$$

Given A and b, find s

$\leadsto$ Very easy (e.g. Gaussian elimination) and in polynomial time

Learning with errors (LWE)

Let $n = 4$, $m = 6$ and $q = 17$.

$$\underset{\textbf{random}}{\underbrace{A \in \mathbb{Z}_q^{m \times n}}} \qquad \underset{\textbf{secret}}{s \in \mathbb{Z}_q^n} \qquad \underset{\textbf{noise}}{e \in \mathbb{Z}_q^m} \qquad b \in \mathbb{Z}_q^m$$

$$\begin{bmatrix} 14 & 12 & 2 & 5 \\ 5 & 3 & 1 & 7 \\ 14 & 7 & 2 & 5 \\ 0 & 9 & 8 & 4 \\ 8 & 11 & 5 & 12 \\ 5 & 1 & 3 & 14 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ 1 \\ 5 \end{bmatrix} + \begin{bmatrix} -3 \\ -1 \\ 2 \\ -3 \\ 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \\ 14 \\ 6 \\ 12 \\ 13 \end{bmatrix}$$

Learning with errors (LWE)

Let $n = 4$, $m = 6$ and $q = 17$.

$$\begin{array}{ccccccc}
\textbf{random} & & \textbf{secret} & & \textbf{noise} & & \\
A \in \mathbb{Z}_q^{m \times n} & & s \in \mathbb{Z}_q^n & & e \in \mathbb{Z}_q^m & & b \in \mathbb{Z}_q^m \\
\begin{bmatrix} 14 & 12 & 2 & 5 \\ 5 & 3 & 1 & 7 \\ 14 & 7 & 2 & 5 \\ 0 & 9 & 8 & 4 \\ 8 & 11 & 5 & 12 \\ 5 & 1 & 3 & 14 \end{bmatrix} & \times & \begin{bmatrix} \; \\ \; \\ \; \\ \; \end{bmatrix} & + & \begin{bmatrix} \; \\ \; \\ \; \\ \; \\ \; \\ \; \end{bmatrix} & = & \begin{bmatrix} 11 \\ 5 \\ 14 \\ 6 \\ 12 \\ 13 \end{bmatrix}
\end{array}$$

Given A and b, find s assuming e is small

$\rightsquigarrow$ Suspected hard problem, even for quantum algorithms

Can always assume that s is small (same hardness)

LWE: security and attacks

LWE is fundamental to lattice-based cryptography:

- several lattice-based NIST selected PQC algorithms rely on LWE
- extensive literature
- all evidence points to resistance against quantum attacks

LWE: security and attacks

LWE is fundamental to lattice-based cryptography:

- several lattice-based NIST selected PQC algorithms rely on LWE
- extensive literature
- all evidence points to resistance against quantum attacks

Two types of attacks:

- Primal attack:
  - more efficient in most cases
  - no quantum speed-up known (besides BKZ)
- Dual attack:
  - originally less efficient, now catching up
  - some controversies about recent advanced dual attacks[DP23]
  - has quantum speed-up (besides BKZ) [A**S**22,P**S**23]

Primal attack

We can formulate $b - A \cdot s \equiv e \pmod q$ as

$$\begin{pmatrix} q\mathbf{I} & -A \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} * \\ s \end{pmatrix} + \begin{pmatrix} b \\ 0 \end{pmatrix} = \begin{pmatrix} e \\ s \end{pmatrix}.$$

Primal attack

We can formulate $b - A \cdot s \equiv e \pmod q$ as

$$\begin{pmatrix} q\mathbf{I} & -A \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} * \\ s \end{pmatrix} + \begin{pmatrix} b \\ 0 \end{pmatrix} = \begin{pmatrix} e \\ s \end{pmatrix}.$$

And make it homogenous with

$$\mathbf{M} := \begin{pmatrix} q\mathbf{I} & -A & b \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{M} \begin{pmatrix} * \\ s \\ 1 \end{pmatrix} = \begin{pmatrix} e \\ s \\ 1 \end{pmatrix}$$

Primal attack

We can formulate $b - A \cdot s \equiv e \pmod{q}$ as

$$\begin{pmatrix} q\mathbf{I} & -A \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} * \\ s \end{pmatrix} + \begin{pmatrix} b \\ 0 \end{pmatrix} = \begin{pmatrix} e \\ s \end{pmatrix}.$$

And make it homogenous with

$$\mathbf{M} := \begin{pmatrix} q\mathbf{I} & -A & b \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{M} \begin{pmatrix} * \\ s \\ 1 \end{pmatrix} = \begin{pmatrix} e \\ s \\ 1 \end{pmatrix}$$

The lattice spanned by **M** has an “unusually” small vector
$\leadsto$ unique shortest vector.

Reduction to uSVP, use BKZ (or more advanced algorithms) to reduce the basis and find the unusually short vector

quantum speed-up: quantum BKZ/SVP

Dual attack

Given $b = A \cdot s + e$, split into two parts:

$$A = \begin{pmatrix} A_{\text{guess}} & A_{\text{dual}} \end{pmatrix}, \qquad s = \begin{pmatrix} s_{\text{guess}} \\ s_{\text{dual}} \end{pmatrix}$$

Consider dual lattice

$$L = \{x \in \mathbb{Z}^{n_{\text{dual}}} : x^T A_{\text{dual}} = 0 \bmod q\}$$

Dual attack

Given $b = A \cdot s + e$, split into two parts:

$$A = \begin{pmatrix} A_{\text{guess}} & A_{\text{dual}} \end{pmatrix}, \qquad s = \begin{pmatrix} s_{\text{guess}} \\ s_{\text{dual}} \end{pmatrix}$$

Consider dual lattice

$$L = \{x \in \mathbb{Z}^{n_{\text{dual}}} : x^T A_{\text{dual}} = 0 \bmod q\}$$

1. Find (exponentially) many short vectors $x_1, \ldots, x_N \in L$, define

$$g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Dual attack

Given $b = A \cdot s + e$, split into two parts:

$$A = \begin{pmatrix} A_{\text{guess}} & A_{\text{dual}} \end{pmatrix}, \qquad s = \begin{pmatrix} s_{\text{guess}} \\ s_{\text{dual}} \end{pmatrix}$$

Consider dual lattice

$$L = \{x \in \mathbb{Z}^{n_{\text{dual}}} : x^T A_{\text{dual}} = 0 \bmod q\}$$

1. Find (exponentially) many short vectors $x_1, \ldots, x_N \in L$, define
$$g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$
2. Compute
$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t)$$

Claim: $\tilde{s}_{\text{guess}} = s_{\text{guess}}$ with high probability (for N sufficiently large)

Quantum dual attack

1. Find many short vectors $x_1, \ldots, x_N$ in L
   - can use BKZ many times $\rightsquigarrow$ quantum BKZ
   - can use discrete Gaussian sampling $\rightsquigarrow$ quantum BKZ + PTIME Klein sampler

Quantum dual attack

1. Find many short vectors $x_1, \dots, x_N$ in L
   - can use BKZ many times $\rightsquigarrow$ quantum BKZ
   - can use discrete Gaussian sampling $\rightsquigarrow$ quantum BKZ + PTIME Klein sampler
2. Compute

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

   - can be done efficiently by discrete Fourier transform (DFT)
     ⚠ classical only, can do quantum Fourier transform (QFT) but does not give a speed-up

Quantum dual attack

1. Find many short vectors $x_1, \ldots, x_N$ in L
   - can use BKZ many times $\rightsquigarrow$ quantum BKZ
   - can use discrete Gaussian sampling $\rightsquigarrow$ quantum BKZ + PTIME Klein sampler
2. Compute

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

   - can be done efficiently by discrete Fourier transform (DFT)

     ⚠ classical only, can do quantum Fourier transform (QFT) but does not give a speed-up
   - quantum: Grover search on t + quantum amplitude estimation to approximate $g(t)$ + QRACM

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Failed approach:

1. Create superposition of short vectors[a]

$$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle$$

[a] Requires a QRACM if samples are sampled classically.
[b] If both x_i and $-x_i$ are in the list, the QFT has real amplitudes.

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Failed approach:

1. Create superposition of short vectors[a]

$$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle$$

2. Apply QFT to get[b]

$$\frac{1}{\sqrt{N q^{n_{\text{guess}}}}} \sum_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(t)\, |t\rangle$$

[a] Requires a QRACM if samples are sampled classically.
[b] If both x_i and $-x_i$ are in the list, the QFT has real amplitudes.

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Failed approach:

1. Create superposition of short vectors[a]

$$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle$$

2. Apply QFT to get[b]

$$\frac{1}{\sqrt{N q^{n_{\text{guess}}}}} \sum_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(t) \, |t\rangle$$

3. Extract vector with highest amplitude:

⚠ No known efficient algorithm, but interesting problem!

[a] Requires a QRACM if samples are sampled classically.

[b] If both x_i and $-x_i$ are in the list, the QFT has real amplitudes.

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Alternative approach:

1. For each t construct
$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle |t\rangle |0\rangle |0\rangle$

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \underset{t \in \mathbb{Z}_q^{n_{\text{guess}}}}{\arg\max}\, g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Alternative approach:

1. For each t construct

$$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle\, |t\rangle\, |0\rangle\, |0\rangle$$

$$\xrightarrow[\text{Product Oracle}]{\text{Cosine Inner}} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle\, |t\rangle\, |\cos(2\pi\langle x_i, t\rangle)\rangle\, |0\rangle$$

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Alternative approach:

1. For each t construct

$$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle\,|t\rangle\,|0\rangle\,|0\rangle$$

$$\xrightarrow[\text{Product Oracle}]{\text{Cosine Inner}} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle\,|t\rangle\,|\cos(2\pi\langle x_i, t\rangle)\rangle\,|0\rangle$$

$$\xrightarrow[\text{Rotation}]{\text{Controlled}} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle\,|t\rangle\,|\cos(2\pi\langle x_i, t\rangle)\rangle \begin{pmatrix} \sqrt{1 - \cos(2\pi\langle x_i, t\rangle)}\,|0\rangle \\ +\sqrt{\cos(2\pi\langle x_i, t\rangle)}\,|1\rangle \end{pmatrix}$$

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Alternative approach:

1. For each t construct

$$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle\, |t\rangle\, |0\rangle\, |0\rangle$$

$$\xrightarrow[\text{Product Oracle}]{\text{Cosine Inner}} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle\, |t\rangle\, |\cos(2\pi \langle x_i, t \rangle)\rangle\, |0\rangle$$

$$\xrightarrow[\text{Rotation}]{\text{Controlled}} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle\, |t\rangle\, |\cos(2\pi \langle x_i, t \rangle)\rangle \begin{pmatrix} \sqrt{1 - \cos(2\pi \langle x_i, t \rangle)}\, |0\rangle \\ + \sqrt{\cos(2\pi \langle x_i, t \rangle)}\, |1\rangle \end{pmatrix}$$

$$= \sqrt{\tfrac{1}{N} g(t)}\, |\phi_0\rangle\, |1\rangle + \sqrt{1 - \tfrac{1}{N} g(t)}\, |\phi_1\rangle\, |0\rangle$$

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Alternative approach:

1. For each t construct

$$|\psi_t\rangle = \sqrt{\frac{1}{N} g(t)}\,|\phi_0\rangle\,|0\rangle + \sqrt{1 - \frac{1}{N} g(t)}\,|\phi_1\rangle\,|1\rangle$$

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Alternative approach:

1. For each t construct
$$|\psi_t\rangle = \sqrt{\tfrac{1}{N} g(t)}\, |\phi_0\rangle\, |0\rangle + \sqrt{1 - \tfrac{1}{N} g(t)}\, |\phi_1\rangle\, |1\rangle$$
2. Use amplitude estimation to approximate $g(t)$

Quantum amplitudes and why the QFT does not work

$$\tilde{s}_{\text{guess}} = \arg\max_{t \in \mathbb{Z}_q^{n_{\text{guess}}}} g(b - A_{\text{guess}} t), \qquad g(t) = \sum_{i=1}^{N} \cos(2\pi \langle x_i, t \rangle)$$

Alternative approach:

1. For each t construct
$$|\psi_t\rangle = \sqrt{\tfrac{1}{N} g(t)}\, |\phi_0\rangle\, |0\rangle + \sqrt{1 - \tfrac{1}{N} g(t)}\, |\phi_1\rangle\, |1\rangle$$
2. Use amplitude estimation to approximate $g(t)$
3. Use quantum maximum finding to find best t

Some other nice papers

- [GK17] LWE is easy with quantum samples of the form

$$\frac{1}{q^n} \sum_{a \in \mathbb{Z}_q^n} |a\rangle \, |a \cdot s + e_a \bmod q\rangle$$

- [CLZ21]:

$$\mathrm{C}\,|\mathrm{LWE}\rangle : \sum_{s \in \mathbb{Z}_q^n} \bigotimes_{i=1}^{m} \Big(\sum_{e_i \in \mathbb{Z}_q} f(e_i)\, |a_i \cdot s + e_i \bmod q\rangle \Big)$$

and

$$\mathrm{S}\,|\mathrm{LWE}\rangle : \Big(a_i, \sum_{e_i \in \mathbb{Z}_q} f(e_i)\, |a_i \cdot s + e_i \bmod q\rangle \Big)$$

can be constructed in polynomial time in certain regimes, and used to solve the Short Integer Solution problem SIS^{∞}.