Improved Classical and Quantum Algorithms for the Shortest Vector Problem

Divesh Aggarwal
Centre for Quantum Technologies

Yanlin Chen
CWI Centrum Wiskunde & Informatica

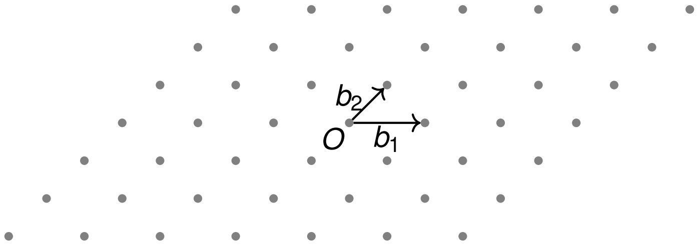
Rajendra Kumar
Centre for Quantum Technologies

Yixin Shen
Royal Holloway University of London

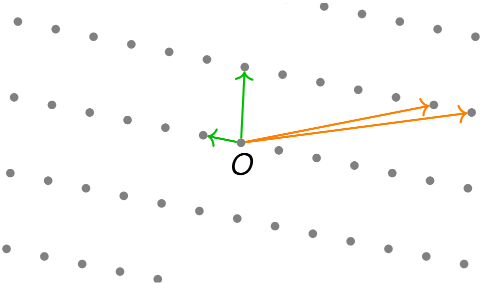
What is a (Euclidean) lattice?

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 $\mathbb{R}^n$.

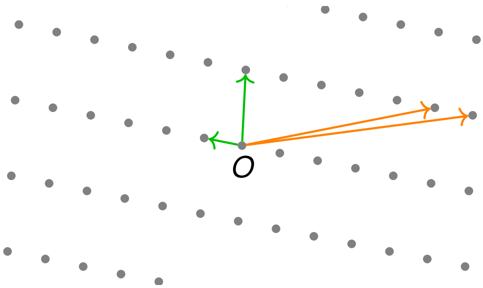


Lattice-based cryptography: fundamental idea



- good basis: private information, makes problem easy
- bad basis: public information, makes problem hard

Lattice-based cryptography: fundamental idea

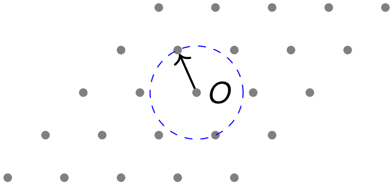


- good basis: private information, makes problem easy
- bad basis: public information, makes problem hard

Basis reduction: transform a bad basis into a good one
Main tool: BKZ algorithm and its variants

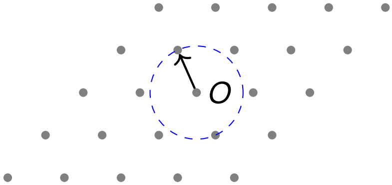
Requires to solve the (approx-)SVP problem in smaller dimensions.

The Shortest Vector Problem



Shortest Vector Problem (SVP):
Given a basis for the lattice $\mathcal{L}$, find a shortest nonzero lattice vector.
$\lambda_1(\mathcal{L})$ = length of such a vector.

The Shortest Vector Problem



Shortest Vector Problem (SVP):
Given a basis for the lattice $\mathcal{L}$, find a shortest nonzero lattice vector.
$\lambda_1(\mathcal{L})$ = length of such a vector.

Main approaches for SVP:

- Enumeration: $2^{O(n \log(n))}$ time and $\mathrm{poly}(n)$ space
- Sieving: $2^{O(n)}$ time and $2^{O(n)}$ space

Sieving

- Heuristic algorithms: fastest in practice
- Provable algorithms: important for theory $\rightarrow$ our work

Results in the Classical Setting

Provable algorithms for SVP:

Time Complexity	Space Complexity	Reference
$n^{\frac{n}{2e}+o(n)}$	$\mathrm{poly}(n)$	[Kan87,HS07]
$2^{n+o(n)}$	$2^{n+o(n)}$	[ADRS15]
$2^{2.05n+o(n)}$	$2^{0.5n+o(n)}$	[CCL18]
$2^{1.669n+o(n)}$	$2^{0.5n+o(n)}$	Our work

Results in the Classical Setting

Provable algorithms for SVP:

Time Complexity	Space Complexity	Reference
$n^{\frac{n}{2e}+o(n)}$	$\mathrm{poly}(n)$	[Kan87,HS07]
$2^{n+o(n)}$	$2^{n+o(n)}$	[ADRS15]
$2^{2.05n+o(n)}$	$2^{0.5n+o(n)}$	[CCL18]
$2^{1.669n+o(n)}$	$2^{0.5n+o(n)}$	Our work

Our work: first provable smooth time/space trade-off for SVP

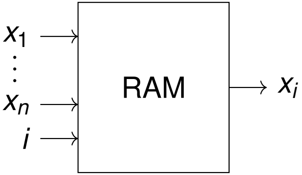
time $q^{13n+o(n)}$ space $\mathrm{poly}(n) \cdot q^{\frac{16n}{q^2}}$ $q \in [4, \sqrt{n}]$

- $q = \sqrt{n}$: time $n^{O(n)}$ and space $\mathrm{poly}(n)$, not as good as [Kan87].
- $q = 4$: time $2^{O(n)}$ and space $2^{O(n)}$, not as good as [ADRS15].

Interlude: quantum memory models

classical access | quantum access

classical data



standard

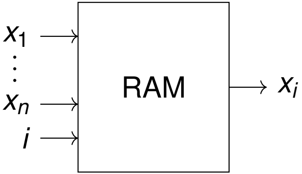
quantum data

Assumption: $O(1)$ time cost

Interlude: quantum memory models

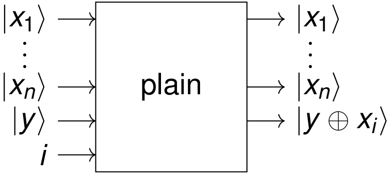
classical access | quantum access

classical data



standard

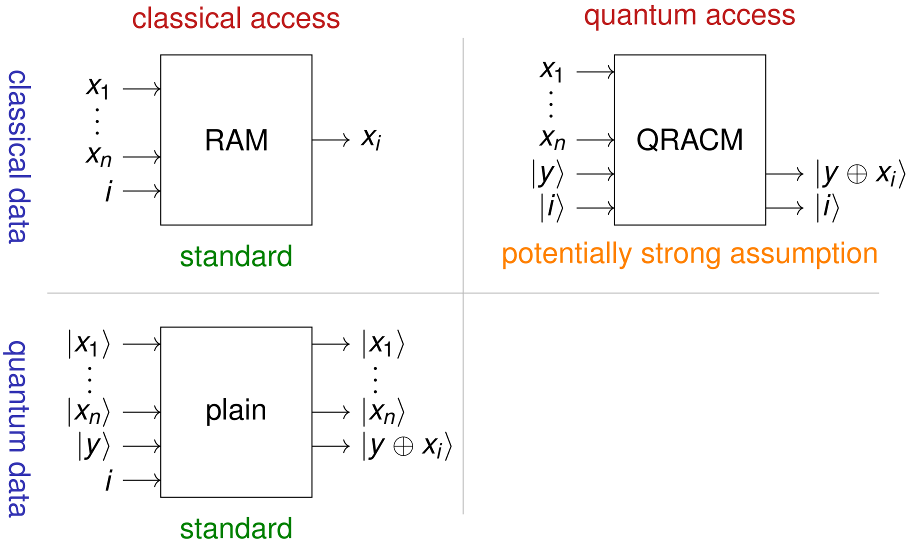
quantum data



standard

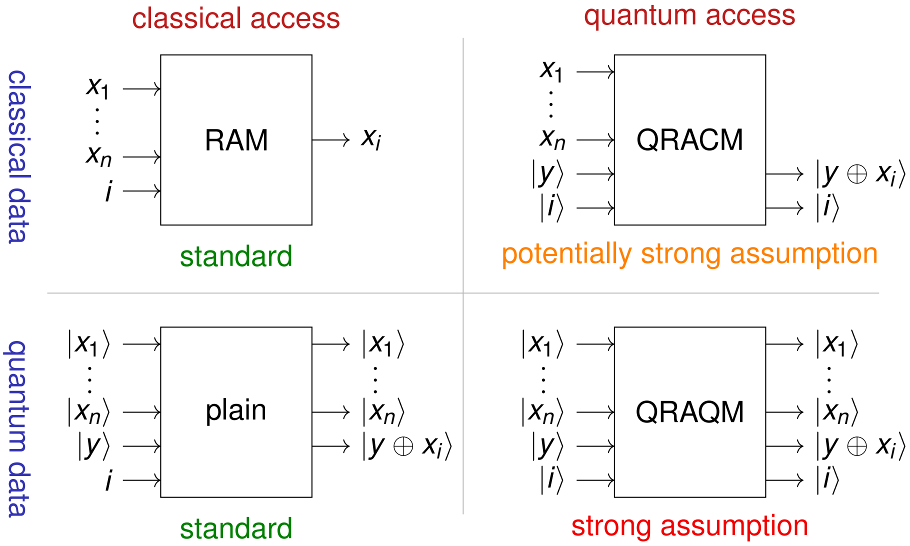
Assumption: $O(1)$ time cost

Interlude: quantum memory models



Assumption: $O(1)$ time cost

Interlude: quantum memory models



Assumption: $O(1)$ time cost

Results in the Quantum Setting

Provable quantum algorithms for SVP:

Time Complexity	Space Complexity			Reference
	Classical	Quantum	Model	
$2^{1.799n+o(n)}$	$2^{1.286n+o(n)}$	$2^{1.286n+o(n)}$	QRACM	[LMP15]
$2^{1.2553n+o(n)}$	$2^{0.5n+o(n)}$	poly(n)	plain	[CCL18]
$2^{n+o(n)}$	$2^{n+o(n)}$	classical algorithm!		[ADRS15]
$2^{0.950n+o(n)}$	$2^{0.5n+o(n)}$	poly(n)	plain	Our work
$2^{0.835n+o(n)}$	$2^{0.5n+o(n)}$	$2^{0.293n+o(n)}$	QRACM	Our work

Remark on quantum heuristic algorithms:

- better complexity: $2^{0.265n+o(n)}$ [Laarhoven15], requires QRACM
- even better complexity: $2^{0.257n+o(n)}$ [CL21], requires QRAQM

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:

Input: many vectors of length $\leqslant \ell$
Output: many vectors of length $\leqslant \frac{\ell}{2}$

Combine pairs of vectors to produce shorter vectors

Sieving Algorithms

Original idea [AKS01]:

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

Sieve:

Input: many vectors of length $\leqslant \ell$
Output: many vectors of length $\leqslant \frac{\ell}{2}$

Combine pairs of vectors to produce shorter vectors

Idea: LLL reduced $\rightsquigarrow$ length $\ell \leqslant 2^{O(n)}\lambda_1$, sieve $O(n)$, solve SVP

Sieving Algorithms

Original idea [AKS01]:

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

Sieve:
Input: many vectors of length $\leqslant \ell$
Output: many vectors of length $\leqslant \frac{\ell}{2}$

Combine pairs of vectors to produce shorter vectors

Idea: LLL reduced $\rightsquigarrow$ length $\ell \leqslant 2^{O(n)}\lambda_1$, sieve $O(n)$, solve SVP

Many heuristic variants: local sensitive hash, tuple sieve, ...
All control the length of the vectors.

Sieving Algorithms

Original idea [AKS01]:

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

Sieve:
Input: many vectors of length $\leqslant \ell$
Output: many vectors of length $\leqslant \frac{\ell}{2}$

Combine pairs of vectors to produce shorter vectors

Idea: LLL reduced $\rightsquigarrow$ length $\ell \leqslant 2^{O(n)}\lambda_1$, sieve $O(n)$, solve SVP

Many heuristic variants: local sensitive hash, tuple sieve, ...
All control the length of the vectors.

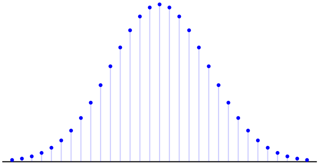
[ADRS15]'s new idea: control distribution instead of length of vectors

Discrete Gaussian Sampling

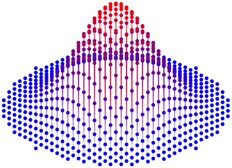
$$\rho_s(\boldsymbol{x}) = \exp\left(-\pi \frac{\|\boldsymbol{x}\|^2}{s^2}\right), \qquad D_{L,s}(\boldsymbol{x}) = \frac{\rho_s(\boldsymbol{x})}{\rho_s(L)}, \qquad \boldsymbol{x} \in \mathbb{R}^n, s > 0.$$

Definition (Discrete Gaussian Distribution)

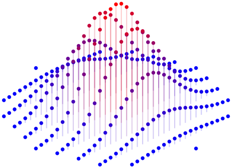
On lattice L with parameter s: probability of $\boldsymbol{x} \in L$ is $D_{L,s}(\boldsymbol{x})$.



$L = \mathbb{Z}, s = 7$



$L = \mathbb{Z}^2, s = 7$



$L = \mathbb{Z} \times 4\mathbb{Z}, s = 7$

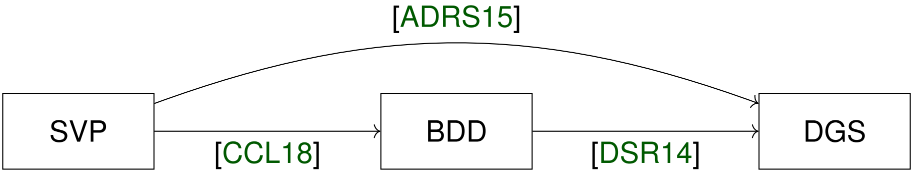
Discrete Gaussian Sampling (DGS)

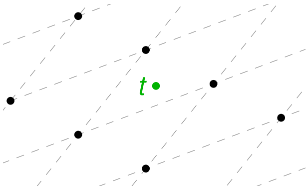
- **input:** L and s
- **output:** random $\boldsymbol{x} \in L$ according to $D_{L,s}$.

DGS, BDD and SVP

$$\begin{tikzcd} \boxed{\text{SVP}} \arrow[rr, bend left, "\text{[ADRS15]}"] & & \boxed{\text{DGS}} \end{tikzcd}$$

DGS, BDD and SVP

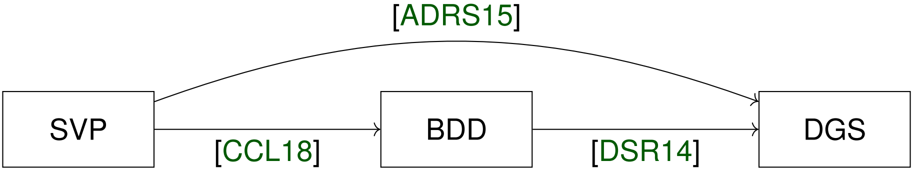


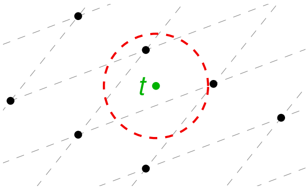


Bounded Distance Decoding ($\alpha-$BDD):

Given a lattice $\mathcal{L}$ and a target vector $t \in \mathbb{R}^n$

DGS, BDD and SVP

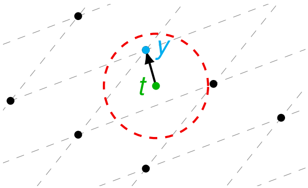




Bounded Distance Decoding (α−BDD):
Given a lattice $\mathcal{L}$ and a target vector $t \in \mathbb{R}^n$ with distance to lattice $\leq \alpha \cdot \lambda_1(\mathcal{L})$

DGS, BDD and SVP

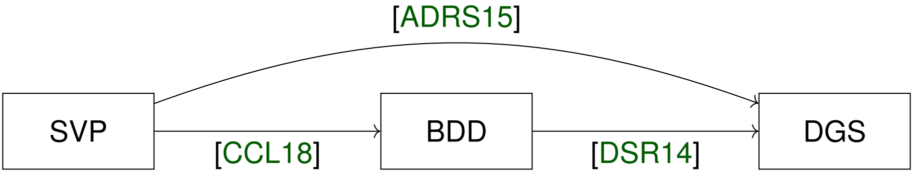
$$\begin{tikzcd} \boxed{\text{SVP}} \arrow[r, "\text{[CCL18]}"'] \arrow[rr, bend left, "\text{[ADRS15]}"] & \boxed{\text{BDD}} \arrow[r, "\text{[DSR14]}"'] & \boxed{\text{DGS}} \end{tikzcd}$$

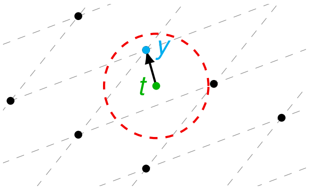


Bounded Distance Decoding ($\alpha-$BDD):
Given a lattice $\mathcal{L}$ and a target vector $t \in \mathbb{R}^n$ with distance to lattice $\leq \alpha \cdot \lambda_1(\mathcal{L})$, find the closest vector $y \in \mathcal{L}$.

- α is the decoding radius
- $\alpha < \frac{1}{2}$ for unique solution

DGS, BDD and SVP





Bounded Distance Decoding ($\alpha-$BDD):
Given a lattice $\mathcal{L}$ and a target vector $t \in \mathbb{R}^n$ with distance to lattice $\leq \alpha \cdot \lambda_1(\mathcal{L})$, find the closest vector $y \in \mathcal{L}$.

- α is the decoding radius
- $\alpha < \frac{1}{2}$ for unique solution

The two reductions use completely different DGS parameter regimes!

Hardness of Discrete Gaussian Sampling

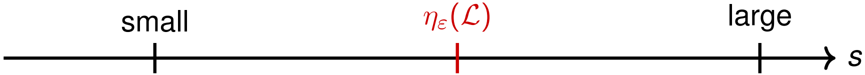
Parameter s (width/standard deviation) of $D_{\mathcal{L},s}$:

- hard to sample
- SVP

- easy to sample

Hardness of Discrete Gaussian Sampling

Parameter s (width/standard deviation) of $D_{\mathcal{L},s}$:



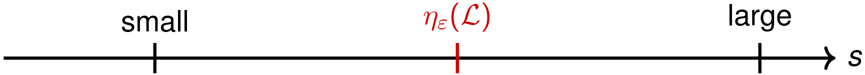
- hard to sample
- SVP

smoothing parameter

- easy to sample

Hardness of Discrete Gaussian Sampling

Parameter s (width/standard deviation) of $D_{\mathcal{L},s}$:



- hard to sample
- SVP

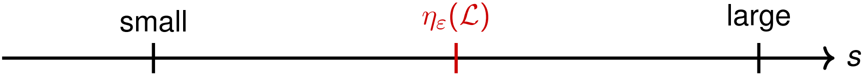
smoothing parameter

- easy to sample

- Open problem: $2^{O(n)}$ time, $2^{o(n)}$ space algorithm for $s = \eta_\varepsilon(\mathcal{L})$

Hardness of Discrete Gaussian Sampling

Parameter s (width/standard deviation) of $D_{\mathcal{L},s}$:



- hard to sample
- SVP

smoothing parameter

- easy to sample

- Open problem: $2^{O(n)}$ time, $2^{o(n)}$ space algorithm for $s = \eta_\varepsilon(\mathcal{L})$

Theorem (ADRS15, best known result)

There is an algorithm that solves DGS for $s = \sqrt{2}\eta_{1/2}(\mathcal{L})$ in time and space $2^{n/2+o(n)}$.

Hardness of Discrete Gaussian Sampling

Parameter s (width/standard deviation) of $D_{\mathcal{L},s}$:

small — $\eta_\varepsilon(\mathcal{L})$ — large — s

- hard to sample
- SVP

smoothing parameter

- easy to sample

- Open problem: $2^{O(n)}$ time, $2^{o(n)}$ space algorithm for $s = \eta_\varepsilon(\mathcal{L})$
- No known time/space trade-off for $s \ll \eta_\varepsilon(\mathcal{L})$

$$\boxed{\text{SVP}} \xrightarrow{\text{[CCL18]}} \boxed{\text{BDD}} \xrightarrow{\text{[DSR14]}} \boxed{\text{DGS}}$$

new provable time/space trade-off

$\rightsquigarrow$ first provable time/space trade-off for SVP

DGS time/space trade-off (simplified)

Idea: **if** $X_1, \ldots, X_k \sim D_{\mathcal{L},s}$ and $\sum_i X_i \in q\mathcal{L}$ **then** $(\sum_i X_i)/q \approx D_{\mathcal{L},s\sqrt{k}/q}$

$\rightsquigarrow$ progress when $k < q^2$, repeat many times to reach $\eta_\varepsilon(\mathcal{L})$

DGS time/space trade-off (simplified)

Idea: **if** $X_1, \dots, X_k \sim D_{\mathcal{L},s}$ and $\sum_i X_i \in q\mathcal{L}$ **then** $(\sum_i X_i)/q \approx D_{\mathcal{L},s\sqrt{k}/q}$
$\rightsquigarrow$ progress when $k < q^2$, repeat many times to reach $\eta_\varepsilon(\mathcal{L})$

Algorithm: given a list of N vectors in $\mathcal{L}$, find $k = q^2 - 1$ of them such that their sum $\in q\mathcal{L}$, then repeat (q is a parameter)

DGS time/space trade-off (simplified)

Idea: **if** $X_1, \ldots, X_k \sim D_{\mathcal{L},s}$ and $\sum_i X_i \in q\,\mathcal{L}$ **then** $(\sum_i X_i)/q \approx D_{\mathcal{L},s\sqrt{k}/q}$

$\rightsquigarrow$ progress when $k < q^2$, repeat many times to reach $\eta_\varepsilon(\mathcal{L})$

Algorithm: given a list of N vectors in $\mathcal{L}$, find $k = q^2 - 1$ of them such that their sum $\in q\,\mathcal{L}$, then repeat (q is a parameter)

- Space: need $N \gtrsim q^{n/q^2}$ to be successful — decrease with q
- Time: q^n to produce one vector — increase with q

DGS time/space trade-off (simplified)

Idea: **if** $X_1, \ldots, X_k \sim D_{\mathcal{L},s}$ and $\sum_i X_i \in q\,\mathcal{L}$ **then** $(\sum_i X_i)/q \approx D_{\mathcal{L},s\sqrt{k}/q}$
$\rightsquigarrow$ progress when $k < q^2$, repeat many times to reach $\eta_\varepsilon(\mathcal{L})$

Algorithm: given a list of N vectors in $\mathcal{L}$, find $k = q^2 - 1$ of them such that their sum $\in q\,\mathcal{L}$, then repeat (q is a parameter)

- Space: need $N \gtrsim q^{n/q^2}$ to be successful — decrease with q
- Time: q^n to produce one vector — increase with q

Difficulties:

- independence of samples
- errors in distributions

Theorem (Simplified)

For $q \in [4, \sqrt{n}]$, there is an algorithm that produces q^{16n/q^2} vectors from $D_{\mathcal{L},s}$ with $s \geqslant \eta_\varepsilon(\mathcal{L})$ in time q^{13n} and space q^{16n/q^2}.

Time-Space Tradeoff for SVP

$$\boxed{\text{SVP}} \xrightarrow[\text{[CCL18]}]{} \boxed{\tfrac{0.1}{q}\text{-BDD}} \xrightarrow[\text{[DSR14]}]{} \boxed{\text{DGS}}$$

new provable time/space trade-off

Smooth time-space tradeoff for BDD: create a $\frac{0.1}{q}$-BDD oracle in time q^{13n}, space q^{16n/q^2}, each call takes time q^{16n/q^2}.

Gives a smooth time-space tradeoff for SVP:

Theorem

Let $n \in \mathbb{N}, q \in [4, \sqrt{n}]$ be a positive integer. Let $\mathcal{L}$ be a lattice of rank n. There is a randomized algorithm that solves SVP in time $q^{13n+o(n)}$ and in space $poly(n) \cdot q^{\frac{16n}{q^2}}$.

SVP to BDD reduction [CCL18]

Lemma (CCL18, simplified)

Given a α-BDD oracle and p an integer, one can enumerate all lattice points in a ball of radius $p\alpha\lambda_1$ using p^n queries to the oracle.

SVP to BDD reduction [CCL18]

Lemma (CCL18, simplified)

Given a α-BDD oracle and p an integer, one can enumerate all lattice points in a ball of radius $p\alpha\lambda_1$ using p^n queries to the oracle.

Solve SVP by using a α-BDD oracle:

- Set $p = \lceil \frac{1}{\alpha} \rceil$.
- Enumerate all points in a ball of radius $> \lambda_1$.

SVP to BDD reduction [CCL18]

Lemma (CCL18, simplified)

Given a α-BDD oracle and p an integer, one can enumerate all lattice points in a ball of radius $p\alpha\lambda_1$ using p^n queries to the oracle.

Solve SVP by using a α-BDD oracle:

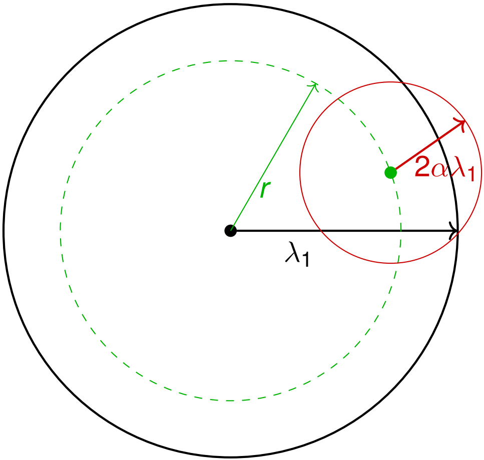
- Set $p = \lceil \frac{1}{\alpha} \rceil$.
- Enumerate all points in a ball of radius $> \lambda_1$.

The reduction is space efficient

But $\alpha < \frac{1}{2} \implies p \geq 3 \implies$ at least 3^n queries

Faster SVP to BDD reduction

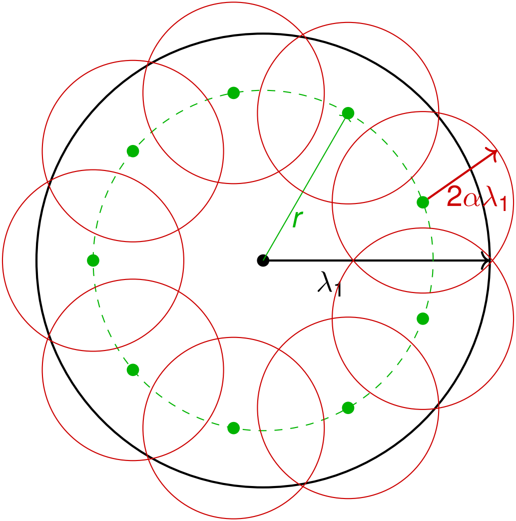
Cover the sphere of radius $\lambda_1(\mathcal{L})$ by balls of radius $2\alpha\lambda_1(\mathcal{L})$:



Use 2^n α−BDD queries to enumerate points in balls of radius $2\alpha\lambda_1$

Faster SVP to BDD reduction

Cover the sphere of radius $\lambda_1(\mathcal{L})$ by balls of radius $2\alpha\lambda_1(\mathcal{L})$:

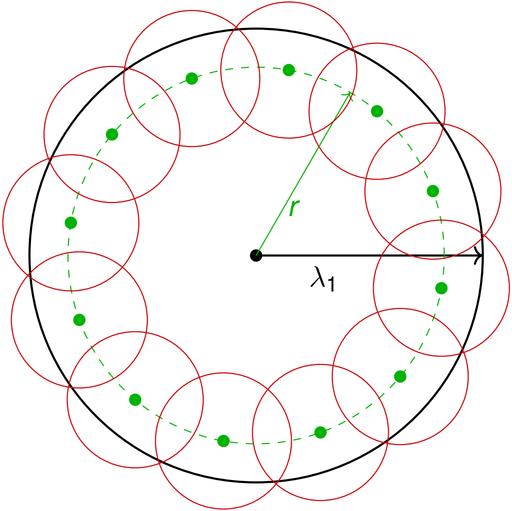


Use 2^n $\alpha-$BDD queries to enumerate points in balls of radius $2\alpha\lambda_1$

Each ball covers a spherical cap.

Faster SVP to BDD reduction

Cover the sphere of radius $\lambda_1(\mathcal{L})$ by balls of radius $2\alpha\lambda_1(\mathcal{L})$:



Use 2^n $\alpha-$BDD queries to enumerate points in balls of radius $2\alpha\lambda_1$

Each ball covers a spherical cap.

Smaller α:

- More balls
- Less expensive BDD

$\rightsquigarrow$ Trade-off

SVP to DGS via BDD

Classical SVP to BDD: do 3^n queries to $1/3$-BDD and keep minimum

$$\begin{tikzcd} \boxed{\text{SVP}} \arrow[r, "{[\text{CCL18}]}"'] & \boxed{1/3\text{-BDD}} \arrow[r, "{[\text{DSR14}]}"'] & \boxed{\text{DGS}} \end{tikzcd}$$

[ADRS15]
+ new
lemma

SVP to DGS via BDD

Classical SVP to BDD: do 3^n queries to $1/3$-BDD and keep minimum

$$\begin{tikzcd}[column sep=huge] \boxed{\text{SVP}} \arrow[r, "\text{[CCL18]}"'] & \boxed{1/3\text{-BDD}} \arrow[r, "\text{[DSR14]}"'] & \boxed{\text{DGS}} \end{tikzcd}$$

[ADRS15]
+ new
lemma

"Improved" SVP to BDD: do $M(n, \alpha)$ queries to α-BDD
Details omitted in this presentation, M is a complicated function

$$\begin{tikzcd}[column sep=huge] \boxed{\text{SVP}} \arrow[r, "\text{new reduction}"'] & \boxed{\alpha\text{-BDD}} \arrow[r, "\text{[DSR14]}"'] & \boxed{\text{DGS}} \end{tikzcd}$$

[ADRS15]
+ new
lemma

SVP to DGS via BDD

Classical SVP to BDD: do 3^n queries to $1/3$-BDD and keep minimum

$$\begin{tikzcd} \boxed{\text{SVP}} \arrow[r, "\text{[CCL18]}"'] & \boxed{1/3\text{-BDD}} \arrow[r, "\text{[DSR14]}"'] & \boxed{\text{DGS}} \end{tikzcd}$$

[ADRS15]
+ new
lemma

"Improved" SVP to BDD: do $M(n, \alpha)$ queries to α-BDD
Details omitted in this presentation, M is a complicated function

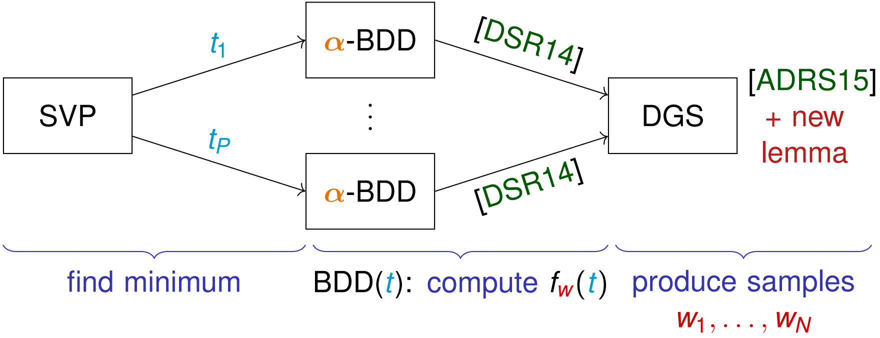
$$\begin{tikzcd} \boxed{\text{SVP}} \arrow[r, "\text{new reduction}"'] & \boxed{\alpha\text{-BDD}} \arrow[r, "\text{[DSR14]}"'] & \boxed{\text{DGS}} \end{tikzcd}$$

[ADRS15]
+ new
lemma

- Not obvious which one is better: less queries to more expensive BDD oracle
- Same structure for both reductions, but different parameters: will be useful for the quantum reduction!

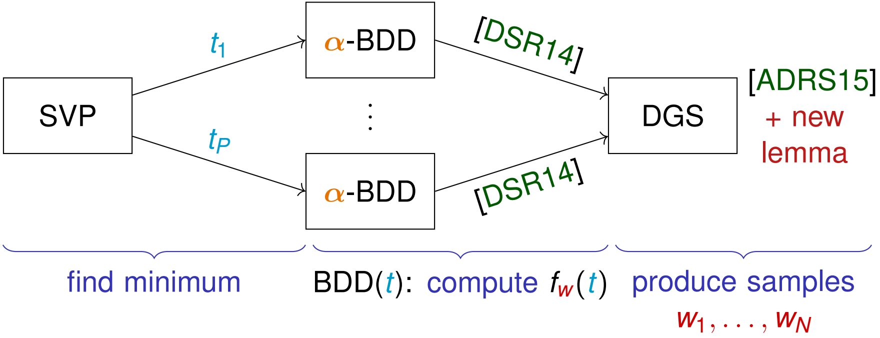
Reduction

- SVP makes $P := M(n, \alpha)$ calls to α-BDD with argument $t_1, \ldots, t_P$
- each BDD call requires N samples $w_1, \ldots, w_N$ from DGS
- $w_1, \ldots, w_N$ can be **shared** across all BDD calls: independent of t_i



Reduction

- SVP makes $P := M(n, \alpha)$ calls to α-BDD with argument $t_1, \ldots, t_P$
- each BDD call requires N samples $w_1, \ldots, w_N$ from DGS
- $w_1, \ldots, w_N$ can be **shared** across all BDD calls: independent of t_i



Classical cost: DGS cost + $P \times$ BDD cost $\approx$ $\mathrm{poly}(n) \times (N + PN)$

Reduction from BDD to DGS

Periodic Gaussian function $f(\boldsymbol{t}) := \frac{\rho(\boldsymbol{t}+\mathcal{L})}{\rho(\mathcal{L})} = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{D}_{\mathcal{L}^*}}[\cos(2\pi\langle\boldsymbol{w},\boldsymbol{t}\rangle)]$

- f achieves maximum on lattice points
- a constant number of gradient ascent steps solves BDD

Reduction from BDD to DGS

Periodic Gaussian function $f(\boldsymbol{t}) := \frac{\rho(\boldsymbol{t}+\mathcal{L})}{\rho(\mathcal{L})} = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{D}_{\mathcal{L}^*}}[\cos(2\pi\langle \boldsymbol{w}, \boldsymbol{t}\rangle)]$

- f achieves maximum on lattice points
- a constant number of gradient ascent steps solves BDD

Approximate f by

$$f_W(t) = \frac{1}{N}\sum_{i=1}^{N}\cos(2\pi\langle w_i, t\rangle)$$

where $w_1, \ldots, w_N$ are i.i.d. DGS samples: small error if N is very large.

Reduction from BDD to DGS

Periodic Gaussian function $f(\boldsymbol{t}) := \frac{\rho(\boldsymbol{t}+\mathcal{L})}{\rho(\mathcal{L})} = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{D}_{\mathcal{L}^*}}[\cos(2\pi\langle \boldsymbol{w}, \boldsymbol{t}\rangle)]$

- f achieves maximum on lattice points
- a constant number of gradient ascent steps solves BDD

Approximate f by

$$f_W(t) = \frac{1}{N}\sum_{i=1}^{N} \cos(2\pi\langle w_i, t\rangle)$$

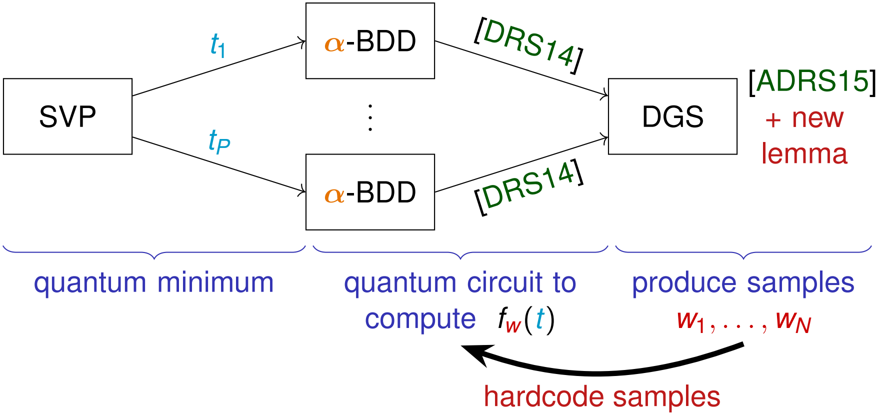
where $w_1, \ldots, w_N$ are i.i.d. DGS samples: small error if N is very large.

Theorem ([DRS14] (Informal))

There is an algorithm that solves α-BDD using N samples from $D_{\mathcal{L}^*,\eta_\varepsilon(\mathcal{L}^*)}$ in time $N \cdot \mathrm{poly}(n)$, where $N = O\left(n^{\frac{\log(1/\varepsilon)}{\sqrt{\varepsilon}}}\right)$ and $\alpha = \alpha(\varepsilon)$.

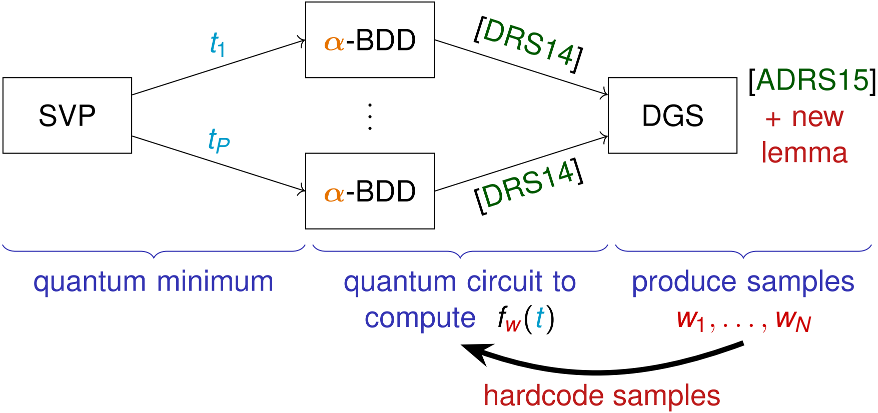
Quantum Reduction

- hardcode DGS samples into a quantum circuit to create a BDD oracle
- use this oracle in a quantum minimum finding algorithm



Quantum Reduction

- hardcode DGS samples into a quantum circuit to create a BDD oracle
- use this oracle in a quantum minimum finding algorithm



Quantum cost: DGS cost + $\sqrt{P} \times$ BDD cost $\approx \mathrm{poly}(n) \times \left(N + \sqrt{P}N\right)$

Reduction from BDD to DGS with QRACM

$$f_W(t) = \frac{1}{N} \sum_{i=1}^{N} \cos(2\pi \langle w_i, t \rangle)$$

where $w_1, \ldots, w_N$ are i.i.d. DGS samples: small error if N is very large.

Reduction from BDD to DGS with QRACM

$$f_W(t) = \frac{1}{N}\sum_{i=1}^{N}\cos(2\pi\langle w_i, t\rangle)$$

where $w_1, \ldots, w_N$ are i.i.d. DGS samples: small error if N is very large.

Our algorithm: approximate f_W quantumly in time $\sqrt{N} \cdot \mathrm{poly}(n)$
Use amplitude estimation and show that the error stays small

Reduction from BDD to DGS with QRACM

$$f_W(t) = \frac{1}{N}\sum_{i=1}^{N}\cos(2\pi\langle w_i, t\rangle)$$

where $w_1, \ldots, w_N$ are i.i.d. DGS samples: small error if N is very large.

Our algorithm: approximate f_W quantumly in time $\sqrt{N}\cdot\mathrm{poly}(n)$
Use amplitude estimation and show that the error stays small

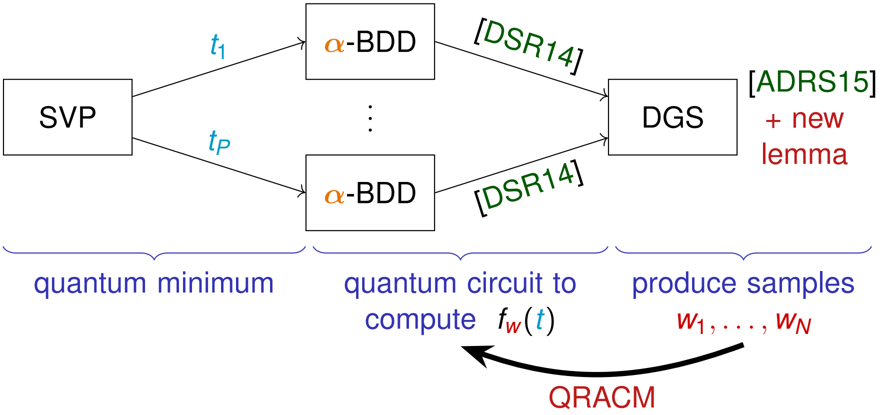
Theorem (Informal)

There is an quantum algorithm that solves α-BDD using N samples from $D_{\mathcal{L}^*,\eta_\varepsilon(\mathcal{L}^*)}$ in time $\sqrt{N}\cdot\mathrm{poly}(n)$, where $N = O\left(n^{8\frac{\log(1/\varepsilon)}{\sqrt{\varepsilon}}}\right)$ and $\alpha = \alpha(\varepsilon)$. It requires a QRACM of size N and $O(N)$ preprocessing time.

$\rightsquigarrow$ gain when doing lots of BDD calls

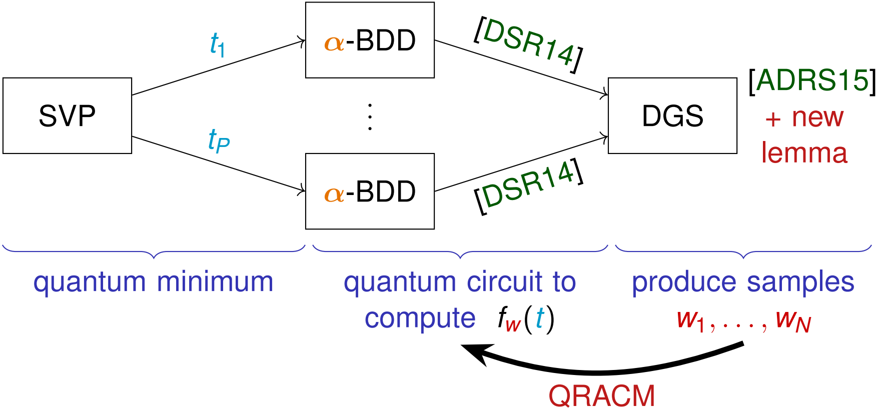
Quantum Reduction with QRACM

- put DGS samples in a QRACM
- BDD oracle uses QRACM + amplitude estimation



Quantum Reduction with QRACM

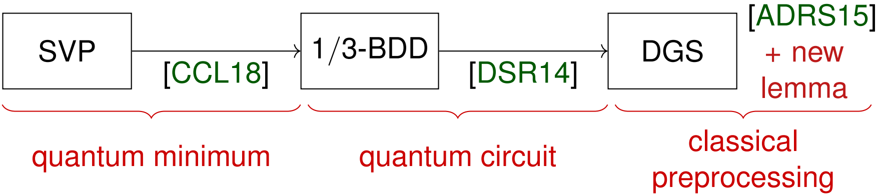
- put DGS samples in a QRACM
- BDD oracle uses QRACM + amplitude estimation



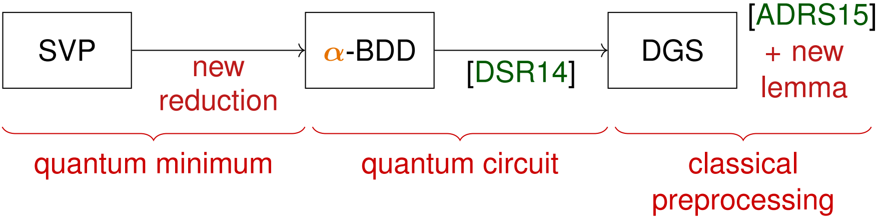
Cost (QRACM): DGS cost + $\sqrt{P} \times$ BDD cost $\approx \mathrm{poly}(n) \times \left(N + \sqrt{PN}\right)$

Quantum SVP

Classical SVP to BDD: do 3^n queries to $1/3$-BDD and keep minimum



"Improved" SVP to BDD: do $M(n, \alpha)$ queries to α-BDD



- Not obvious which one is better: less queries to more expensive BDD oracle

SVP and Generalized Kissing Number

Number of lattice points in a ball of radius r is $\leqslant c^{n+o(n)} r^n$

$$\beta(\mathcal{L}) = \text{smallest } c \text{ that works for all } r$$

- Upper bound: $\beta(\mathcal{L}) \leqslant 2^{0.401}$ [KL78]
- Conjectured to be $\beta(\mathcal{L}) \approx 1$ for most lattices

SVP and Generalized Kissing Number

Number of lattice points in a ball of radius r is $\leqslant c^{n+o(n)} r^n$

$$\beta(\mathcal{L}) = \text{smallest } c \text{ that works for all } r$$

- Upper bound: $\beta(\mathcal{L}) \leqslant 2^{0.401}$ [KL78]
- Conjectured to be $\beta(\mathcal{L}) \approx 1$ for most lattices

$$\boxed{\alpha\text{−BDD}} \xrightarrow[\text{reduce}]{} \boxed{\text{DGS}_{\eta_\varepsilon}}$$

Best known relations between α and ε depends on $\beta(\mathcal{L})$:

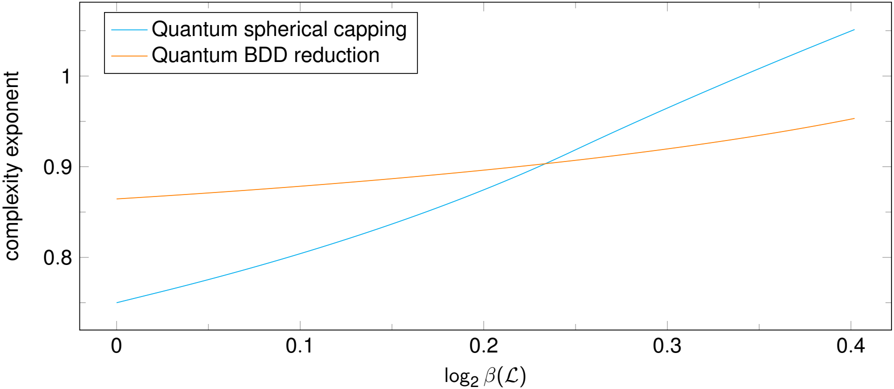
small $\beta(\mathcal{L})$ $\rightsquigarrow$ bigger α for fixed ε $\rightsquigarrow$ less expensive BDD

SVP and Generalized Kissing Number

Number of lattice points in a ball of radius r is $\leqslant c^{n+o(n)} r^n$

$$\beta(\mathcal{L}) = \text{smallest } c \text{ that works for all } r$$

- Upper bound: $\beta(\mathcal{L}) \leqslant 2^{0.401}$ [KL78]
- Conjectured to be $\beta(\mathcal{L}) \approx 1$ for most lattices

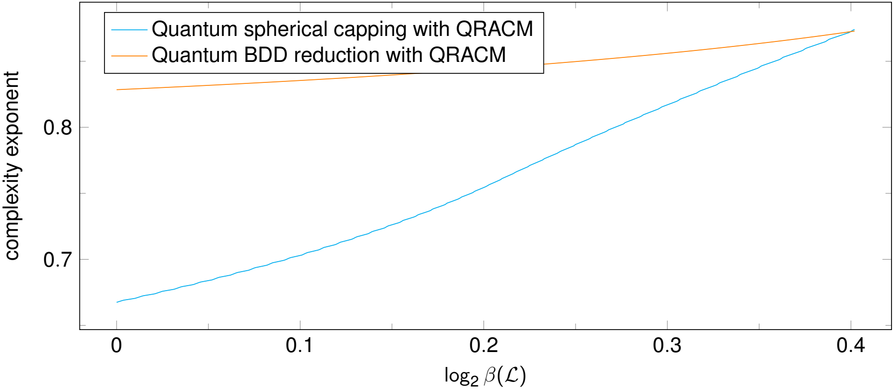


SVP and Generalized Kissing Number

Number of lattice points in a ball of radius r is $\leqslant c^{n+o(n)} r^n$

$$\beta(\mathcal{L}) = \text{smallest } c \text{ that works for all } r$$

- Upper bound: $\beta(\mathcal{L}) \leqslant 2^{0.401}$ [KL78]
- Conjectured to be $\beta(\mathcal{L}) \approx 1$ for most lattices



Conclusions and Future work

Provable SVP:

- classical: time $2^{1.669n+o(n)}$, space $2^{0.5n+o(n)}$
- quantum: $2^{0.950n+o(n)}$, space $2^{0.5n+o(n)}$ and $\mathrm{poly}(n)$ qubits
- quantum: $2^{0.835n+o(n)}$, classical space $2^{0.5n+o(n)}$ and QRACM $2^{0.293n+o(n)}$
- first time/space tradeoff: time q^{13n}, space q^{16n/q^2} for $q \in [4, \sqrt{n}]$
- studied dependency on $\beta(\mathcal{L})$, generalized kissing number

Conclusions and Future work

Provable SVP:

- classical: time $2^{1.669n+o(n)}$, space $2^{0.5n+o(n)}$
- quantum: $2^{0.950n+o(n)}$, space $2^{0.5n+o(n)}$ and $\mathrm{poly}(n)$ qubits
- quantum: $2^{0.835n+o(n)}$, classical space $2^{0.5n+o(n)}$ and QRACM $2^{0.293n+o(n)}$
- first time/space tradeoff: time q^{13n}, space q^{16n/q^2} for $q \in [4, \sqrt{n}]$
- studied dependency on $\beta(\mathcal{L})$, generalized kissing number

Open problems:

- Show that random lattices satisfy $\beta(\mathcal{L}) \approx 1$?
- Fill the gap between provable and heuristic algorithms for sieving?
- Exploit the subexponential space regime in our trade-off for SVP?
- $2^{O(n)}$ time, $2^{o(n)}$ space algorithm for DGS at smoothing parameter?

Backup slides

DGS sampling: new lemma

- [ADRS15]: DGS of parameter $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L})$ in time $2^{n/2}$
- BDD to DGS reduction requires $s = \eta_\varepsilon(\mathcal{L})$ for some $\varepsilon > 0$

Previous work [CCL18]: find ε such that $\eta_\varepsilon(\mathcal{L}) \geqslant \sqrt{2}\eta_{1/2}$
$\rightsquigarrow$ very small ε, larger than necessary BDD radius, too expensive BDD

DGS sampling: new lemma

- [ADRS15]: DGS of parameter $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L})$ in time $2^{n/2}$
- BDD to DGS reduction requires $s = \eta_\varepsilon(\mathcal{L})$ for some $\varepsilon > 0$

Previous work [CCL18]: find ε such that $\eta_\varepsilon(\mathcal{L}) \geqslant \sqrt{2}\eta_{1/2}$
$\rightsquigarrow$ very small ε, larger than necessary BDD radius, too expensive BDD

New idea:

- find a well-chosen lattice $\mathcal{L} \subset \mathcal{L}' \subset \mathcal{L}/2$ such that $\eta_{\varepsilon'}(\mathcal{L}') \leqslant \eta_\varepsilon(\mathcal{L})/\sqrt{2}$ for $\varepsilon' \approx \varepsilon$ [ADRS15]
- run DGS on $\mathcal{L}'$ at $s = \eta_{1/3}(\mathcal{L}) \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L}')$ [ADRS15]
- only keep samples in $\mathcal{L}$ (rejection)

DGS sampling: new lemma

- [ADRS15]: DGS of parameter $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L})$ in time $2^{n/2}$
- BDD to DGS reduction requires $s = \eta_\varepsilon(\mathcal{L})$ for some $\varepsilon > 0$

Previous work [CCL18]: find ε such that $\eta_\varepsilon(\mathcal{L}) \geqslant \sqrt{2}\eta_{1/2}$
$\rightsquigarrow$ very small ε, larger than necessary BDD radius, too expensive BDD

New idea:

- find a well-chosen lattice $\mathcal{L} \subset \mathcal{L}' \subset \mathcal{L}/2$ such that $\eta_{\varepsilon'}(\mathcal{L}') \leqslant \eta_\varepsilon(\mathcal{L})/\sqrt{2}$ for $\varepsilon' \approx \varepsilon$ [ADRS15]
- run DGS on $\mathcal{L}'$ at $s = \eta_{1/3}(\mathcal{L}) \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L}')$ [ADRS15]
- only keep samples in $\mathcal{L}$ (rejection)

Some details:

- $\mathcal{L}'$ is chosen randomly, works with high probability
- need that $|\mathcal{L}'/\mathcal{L}| \approx 2^{n/2}$ for $\varepsilon \approx \varepsilon'$
- rejection: $|\mathcal{L}'/\mathcal{L}| \approx 2^{n/2}$ slowdown, still better than previous work!
- allows to choose $\alpha = 1/3$ for BDD, improved from 0.391 [CCL18]