Quantum speed-ups for sieving algorithms for the shortest vector problem

Elena Kirshanova

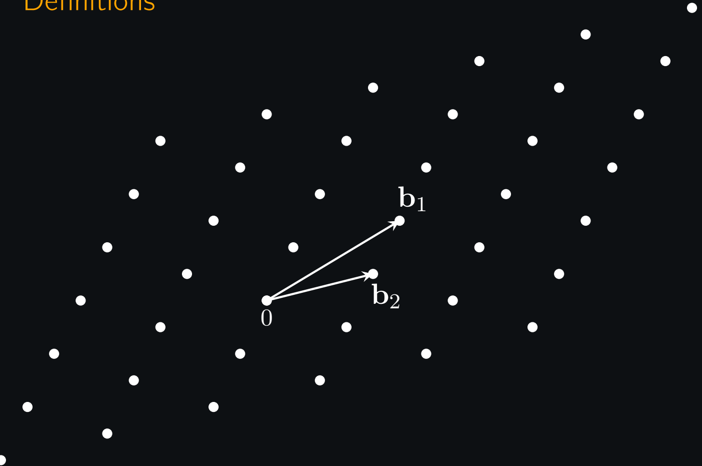
based on joint work with Erik Mårtensson, Eamonn W. Postlethwaite, Subhayan Roy Moulik

presented at AsiaCrypt'19

TQC 2020, Riga, Latvia

June 11, 2020

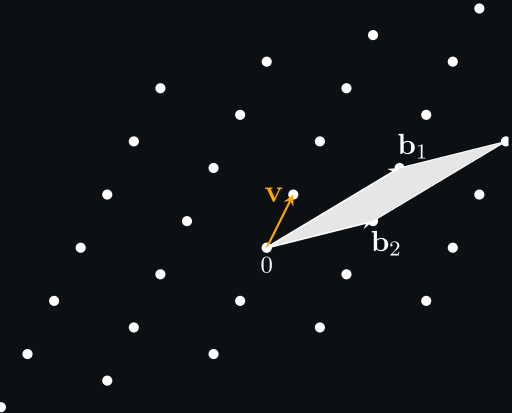
Definitions



A lattice is a set $\mathcal{L} = \{\sum_{i \le n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ for some linearly independent $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$ – a basis of $\mathcal{L}$

Definitions



Minimum

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v}\in\mathcal{L}\setminus\mathbf{0}} \|\mathbf{v}\|$$

Determinant

$$\det(\mathcal{L}) = |\det(\mathbf{b}_i)_i|$$

Minkowski bound

$$\lambda_1(\mathcal{L}) \leq \sqrt{n}\cdot\det(\mathcal{L})^{\frac{1}{n}}$$

A lattice is a set $\mathcal{L} = \{\sum_{i\leq n} x_i\mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ for some linearly independent $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$ – a basis of $\mathcal{L}$

SVP

The Shortest Vector Problem (SVP) asks to find $\mathbf{v}_{\text{shortest}} \in \mathcal{L}$:

$$\|\mathbf{v}_{\text{shortest}}\| = \lambda_1(\mathcal{L})$$

SVP

The Shortest Vector Problem (SVP) asks to find $\mathbf{v}_{\text{shortest}} \in \mathcal{L}$:

$$||\mathbf{v}_{\text{shortest}}|| = \lambda_1(\mathcal{L})$$

Often we are satisfied with an approximation (γ-SVP) to $\mathbf{v}_{\text{shortest}}$:

$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot \lambda_1(\mathcal{L})$$

Why is SVP interesting?

$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot \lambda_1(\mathcal{L})$$

Hardness of (approx)-SVP underlies all lattice-based cryptographic constructions.

- For $\gamma = 2^{\log^{1-\varepsilon} n}$ SVP is NP-hard
- Crypto is based on $\gamma = \mathrm{poly}(n)$
- We assume SVP is infeasible for $n > 350$
- What we can achieve now is $n = 170$ using lots of RAM and GPUs, see TU Darmstadt's SVP challenge[1]
- There is an open-source library G6K[2] for solving SVP

[1] https://www.latticechallenge.org/svp-challenge/
[2] https://github.com/fplll/g6k

Asymptotics ($+o()$ everywhere) of γ-SVP, $\gamma < \mathrm{poly}(n)$, $n := \dim \mathcal{L}$

Classical	Quantum

Asymptotics ($+o()$ everywhere) of γ-SVP, $\gamma < \mathrm{poly}(n)$, $n := \dim \mathcal{L}$

Classical	Quantum

Enumeration

$\log \mathsf{Time} = \frac{1}{2e} n \log n$	$\log \mathsf{Time} = \frac{1}{4e} n \log n$
$\mathsf{Mem} = \mathrm{poly}(n)$ [HS07]	$\mathsf{Mem} = \mathrm{poly}(n)$ [ANS18]

Asymptotics ($+o()$ everywhere) of γ-SVP, $\gamma < \mathrm{poly}(n)$, $n := \dim \mathcal{L}$

Classical	Quantum
Enumeration	
$\log \text{Time} = \frac{1}{2e} n \log n$	$\log \text{Time} = \frac{1}{4e} n \log n$
$\text{Mem} = \mathrm{poly}(n)$ [HS07]	$\text{Mem} = \mathrm{poly}(n)$ [ANS18]
Based on discrete Gaussian samplers	
$\log \text{Time} = 1n$	$\log \text{Time} = 1.2553n$
$\log \text{Mem} = 1n$ [ADRS15]	$\log \text{Mem} = 0.5n$ [LLK18]

Asymptotics ($+o()$ everywhere) of γ-SVP, $\gamma < \mathrm{poly}(n)$, $n := \dim \mathcal{L}$

Classical	Quantum
Enumeration	
$\log \text{Time} = \frac{1}{2e} n \log n$	$\log \text{Time} = \frac{1}{4e} n \log n$
$\text{Mem} = \mathrm{poly}(n)$ [HS07]	$\text{Mem} = \mathrm{poly}(n)$ [ANS18]
Based on discrete Gaussian samplers	
$\log \text{Time} = 1n$	$\log \text{Time} = 1.2553n$
$\log \text{Mem} = 1n$ [ADRS15]	$\log \text{Mem} = 0.5n$ [LLK18]
Sieving (provable)	
$\log \text{Time} = 2.465n$	$\log \text{Time} = 1.799n$
$\log \text{Mem} = 1.325n$ [PS09]	$\log \text{Mem} = 1.286n$ [LMP15]

Asymptotics ($+o()$ everywhere) of γ-SVP, $\gamma < \mathrm{poly}(n)$, $n := \dim \mathcal{L}$

Classical	Quantum
Enumeration	
$\log \text{Time} = \frac{1}{2e} n \log n$	$\log \text{Time} = \frac{1}{4e} n \log n$
$\text{Mem} = \mathrm{poly}(n)$ [HS07]	$\text{Mem} = \mathrm{poly}(n)$ [ANS18]
Based on discrete Gaussian samplers	
$\log \text{Time} = 1n$	$\log \text{Time} = 1.2553n$
$\log \text{Mem} = 1n$ [ADRS15]	$\log \text{Mem} = 0.5n$ [LLK18]
Sieving (provable)	
$\log \text{Time} = 2.465n$	$\log \text{Time} = 1.799n$
$\log \text{Mem} = 1.325n$ [PS09]	$\log \text{Mem} = 1.286n$ [LMP15]
Sieving (heuristic) +LSH	
$\log \text{Time} = 0.292n$	$\log \text{Time} = 0.265n$
$\log \text{Mem} = 0.208n$ [BDGL16]	$\log \text{Mem} = 0.265n$ [Laa16]
Sieving (heuristic)	
$\log \text{Time} = 0.396n$	$\log \text{Time} = 0.299n$
$\log \text{Mem} = 0.189n$ [HK17]	$\log \text{Mem} = 0.139n$ [KMPR19]

Time/Memory trade-offs exist

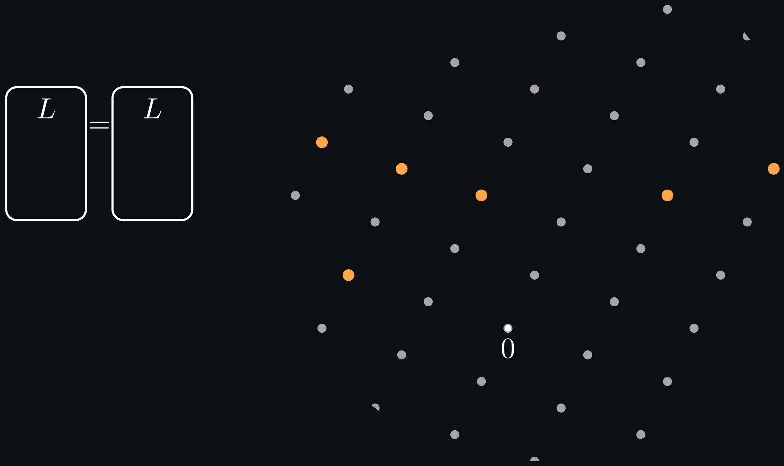
For quantum algorithms Mem means quantumly addressable classical RAM.

Sieving for SVP

https://sites.google.com/a/x.bestledlights.cf/a223/1-32772214555

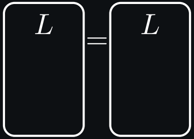
Basic 2-Sieve (Nguyen-Vidick sieve)

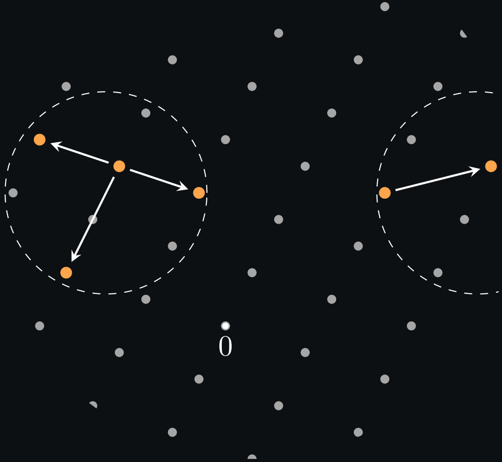
Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors



Basic 2-Sieve (Nguyen-Vidick sieve)

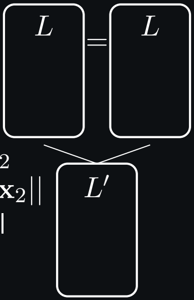
Main idea in all sieving algorithms: saturate space
with enough lattice vectors so that their sums give short(er) vectors



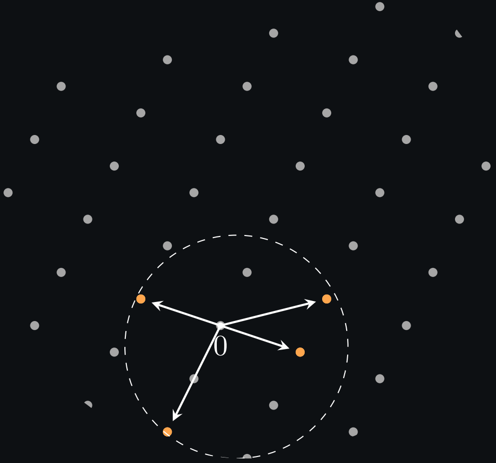


Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space
with enough lattice vectors so that their sums give short(er) vectors

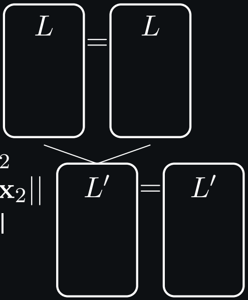


$\mathbf{x}_1 \pm \mathbf{x}_2$
$||\mathbf{x}_1 \pm \mathbf{x}_2||$
is small



Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space
with enough lattice vectors so that their sums give short(er) vectors

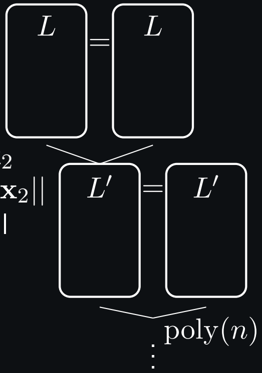


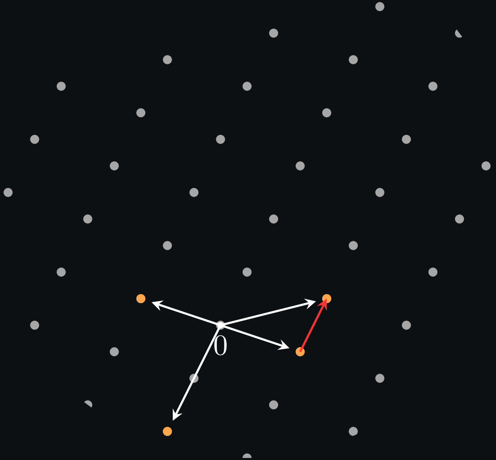
$\mathbf{x}_1 \pm \mathbf{x}_2$
$||\mathbf{x}_1 \pm \mathbf{x}_2||$
is small

Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space
with enough lattice vectors so that their sums give short(er) vectors

$\mathbf{x}_1 \pm \mathbf{x}_2$
$||\mathbf{x}_1 \pm \mathbf{x}_2||$
is small

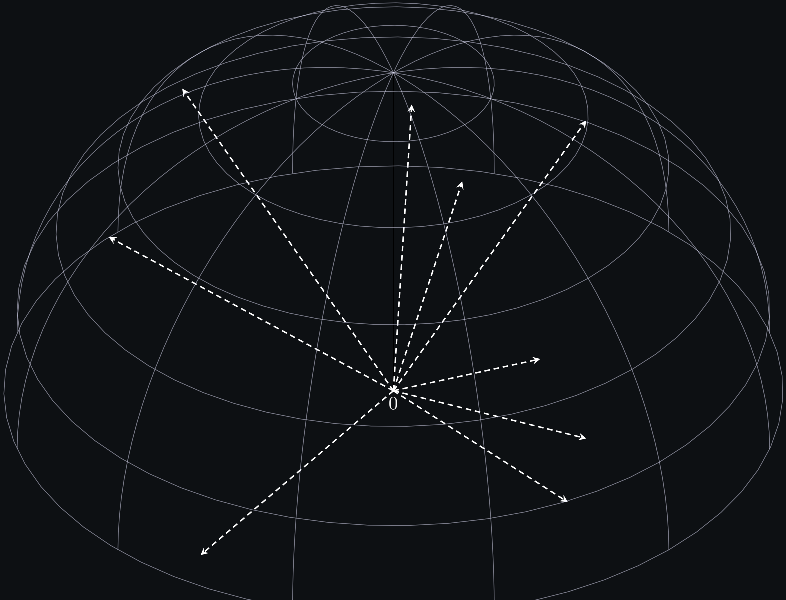




How large $|L|$ should be?

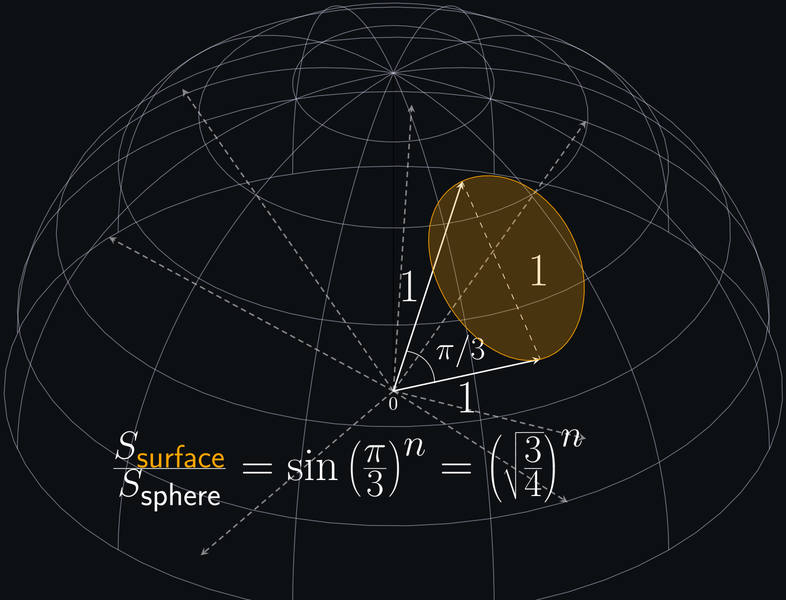
How large $|L|$ should be?

Assumption: vectors (normalized) in $|L|$ are uniform iid on S^{n-1}.



How large $|L|$ should be?

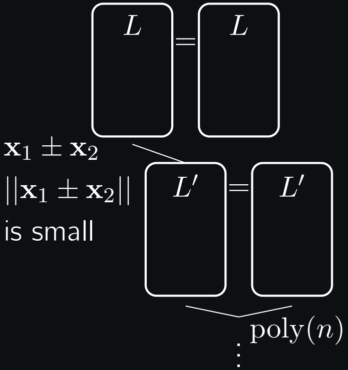
Assumption: vectors (normalized) in $|L|$ are uniform iid on S^{n-1}.



$$\frac{S_{\text{surface}}}{S_{\text{sphere}}} = \sin\left(\frac{\pi}{3}\right)^n = \left(\sqrt{\frac{3}{4}}\right)^n$$

Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors



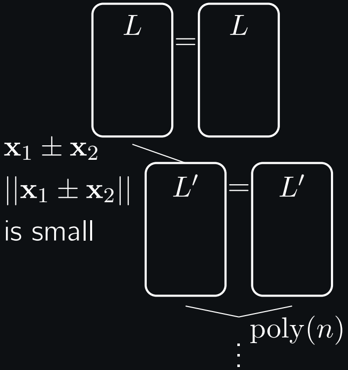
$$\text{Mem} = \left(\sqrt{\frac{3}{4}}\right)^{-n} = 2^{0.2075n}$$

$$\text{Time}^{\text{Class}} = |L|^2 = 2^{0.415n}$$

All $o(n)$ terms are omitted

Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors



$$\text{Mem} = \left(\sqrt{\frac{3}{4}}\right)^{-n} = 2^{0.2075n}$$

$$\text{Time}^{\text{Class}} = |L|^2 = 2^{0.415n}$$

Grover over L :

$$\text{Time}^{\text{Quant}} = 2^{0.311n}$$

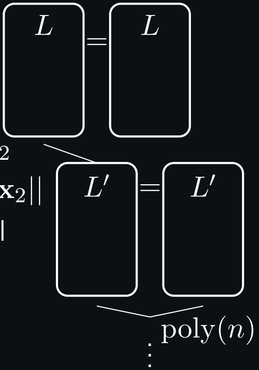
All $o(n)$ terms are omitted

Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors

$\mathbf{x}_1 \pm \mathbf{x}_2$

$||\mathbf{x}_1 \pm \mathbf{x}_2||$ is small



$$\text{Mem} = \left(\sqrt{\frac{3}{4}}\right)^{-n} = 2^{0.2075n}$$

$$\text{Time}^{\text{Class}} = |L|^2 = 2^{0.415n}$$

Grover over L :

$$\text{Time}^{\text{Quant}} = 2^{0.311n}$$

- we need to find almost all $((1-o(1))$–fraction of) close pairs
- 'close' pairs are much closer than the two random ones

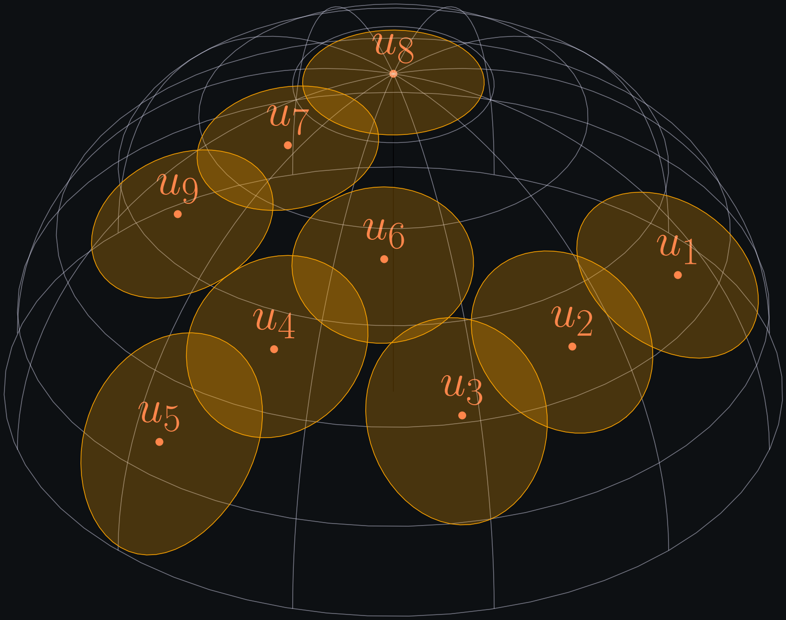
All $o(n)$ terms are omitted

How to improve classical runtime to $T = 2^{0.292n+o(n)}$?

Use Near Neighbour search!

Locality-sensitive filtering [BGJ15, BDGL16]

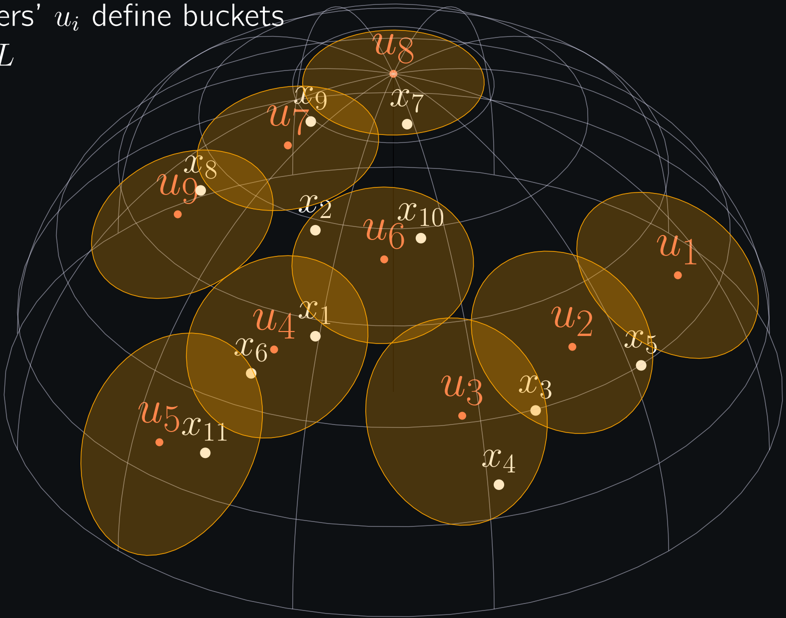
Locality-sensitive filtering [BGJ15, BDGL16]



Locality-sensitive filtering [BGJ15, BDGL16]

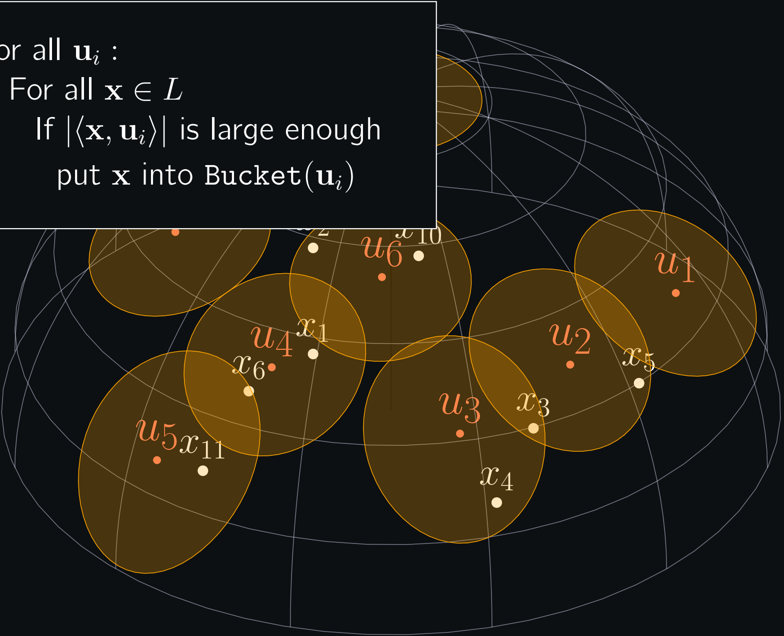
'centers' u_i define buckets

$x_i \in L$



Locality-sensitive filtering [BGJ15, BDGL16]

For all $\mathbf{u}_i$:
    For all $\mathbf{x} \in L$
        If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ is large enough
            put $\mathbf{x}$ into $\texttt{Bucket}(\mathbf{u}_i)$



Locality-sensitive filtering [BGJ15, BDGL16]

For all $\mathbf{u}_i$:

  For all $\mathbf{x} \in L$

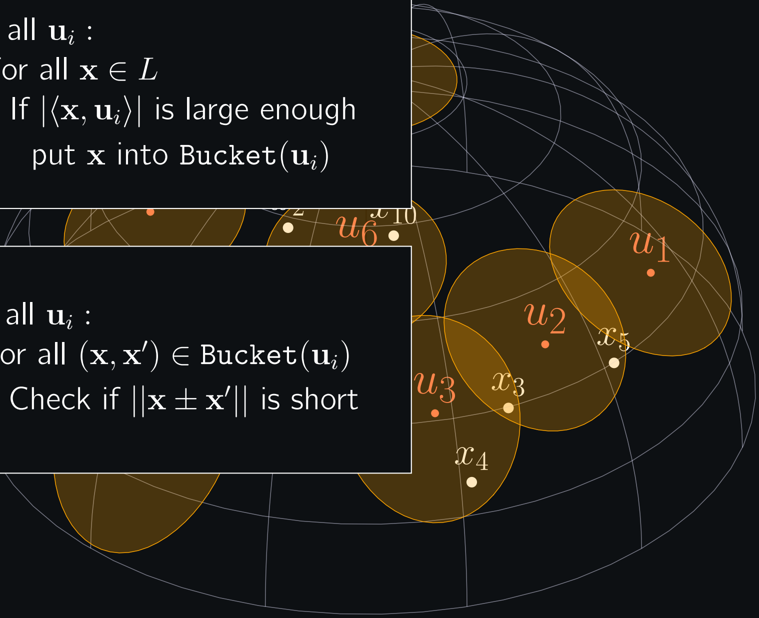
    If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ is large enough

      put $\mathbf{x}$ into $\texttt{Bucket}(\mathbf{u}_i)$

For all $\mathbf{u}_i$:

  For all $(\mathbf{x}, \mathbf{x}') \in \texttt{Bucket}(\mathbf{u}_i)$

    Check if $||\mathbf{x} \pm \mathbf{x}'||$ is short



Locality-sensitive filtering [BGJ15, BDGL16]

For all $\mathbf{u}_i$:

  For all $\mathbf{x} \in L$

    If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ is large enough

      put $\mathbf{x}$ into $\texttt{Bucket}(\mathbf{u}_i)$

For all $\mathbf{u}_i$:

  For all $(\mathbf{x}, \mathbf{x}') \in \texttt{Bucket}(\mathbf{u}_i)$

    Check if $||\mathbf{x} \pm \mathbf{x}'||$ is short

For $2^{(0.142+o(1))n}$ many $\mathbf{u}_i$'s:

$$T = 2^{(0.349+o(1))n}$$

When $\mathbf{u}_i$'s are of special form

$$T = 2^{(0.292+o(1))n}$$

Locality-sensitive filtering [BGJ15, BDGL16]

For all $\mathbf{u}_i$:
  For all $\mathbf{x} \in L$
    If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ is large enough
      put $\mathbf{x}$ into $\texttt{Bucket}(\mathbf{u}_i)$

For all $\mathbf{u}_i$:
  For all $(\mathbf{x}, \mathbf{x}') \in \texttt{Bucket}(\mathbf{u}_i)$
    Check if $||\mathbf{x} \pm \mathbf{x}'||$ is short

For $2^{(0.142+o(1))n}$ many $\mathbf{u}_i$'s:

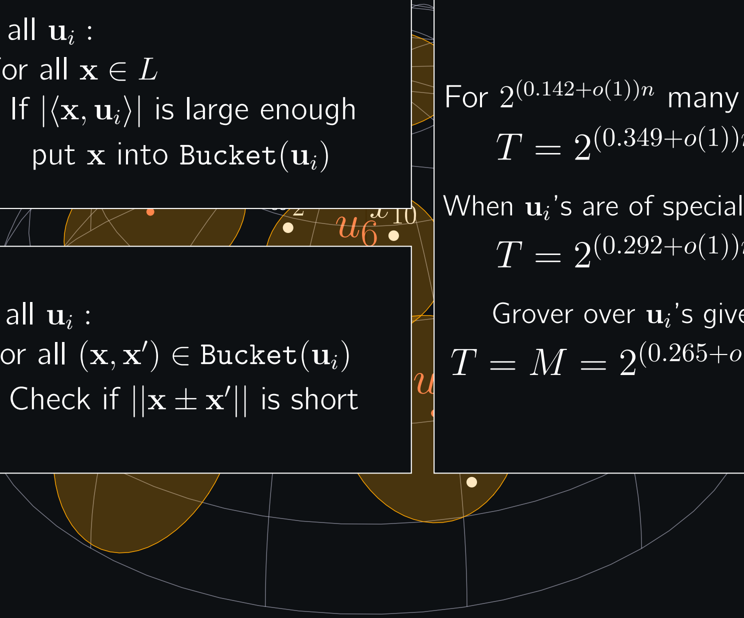
$$T = 2^{(0.349+o(1))n}$$

When $\mathbf{u}_i$'s are of special form

$$T = 2^{(0.292+o(1))n}$$

Grover over $\mathbf{u}_i$'s gives

$$T = M = 2^{(0.265+o(1))n}$$



Locality-sensitive filtering [BGJ15, BDGL16]

For all $\mathbf{u}_i$:
    For all $\mathbf{x} \in L$
        If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ is large enough
            put $\mathbf{x}$ into $\texttt{Bucket}(\mathbf{u}_i)$

For all $\mathbf{u}_i$:
    For all $(\mathbf{x}, \mathbf{x}') \in \texttt{Bucket}(\mathbf{u}_i)$
        Check if $||\mathbf{x} \pm \mathbf{x}'||$ is short

For $2^{(0.142+o(1))n}$ many $\mathbf{u}_i$'s:

$$T = 2^{(0.349+o(1))n}$$

When $\mathbf{u}_i$'s are of special form

$$T = 2^{(0.292+o(1))n}$$

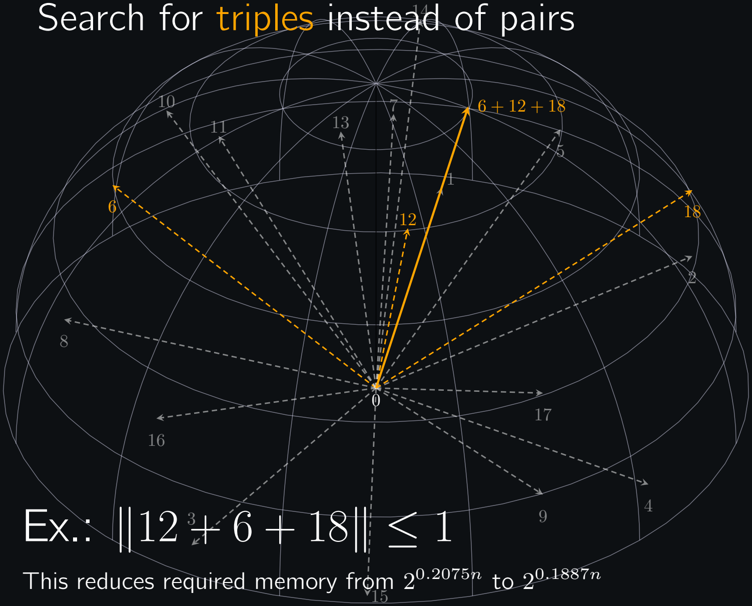
Grover over $\mathbf{u}_i$'s gives

$$T = M = 2^{(0.265+o(1))n}$$

Lower memory?

3-Sieve as 3-List problem

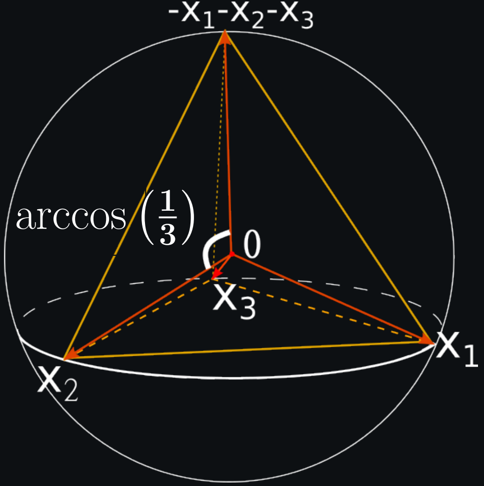
Search for triples instead of pairs



Ex.: $\|12 + 6 + 18\| \leq 1$

This reduces required memory from $2^{0.2075n}$ to $2^{0.1887n}$

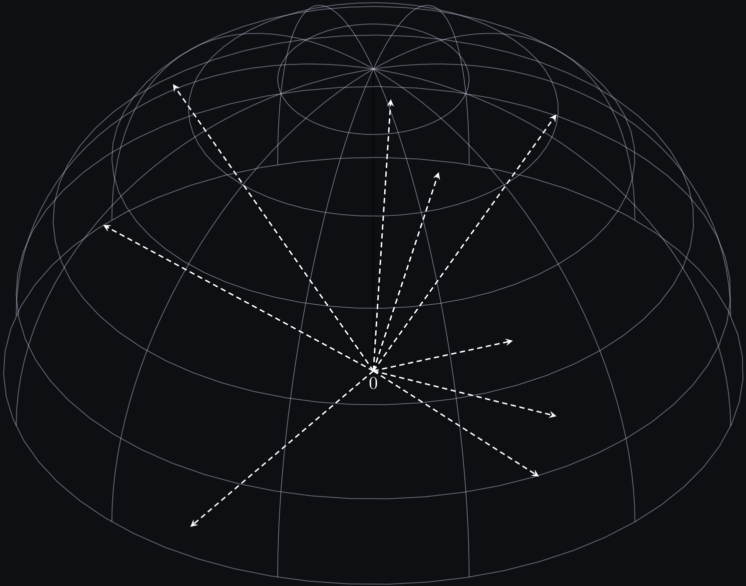
Configuration of good triples, [HK17]



All good triples are concentrated in the shape of 3-simplex

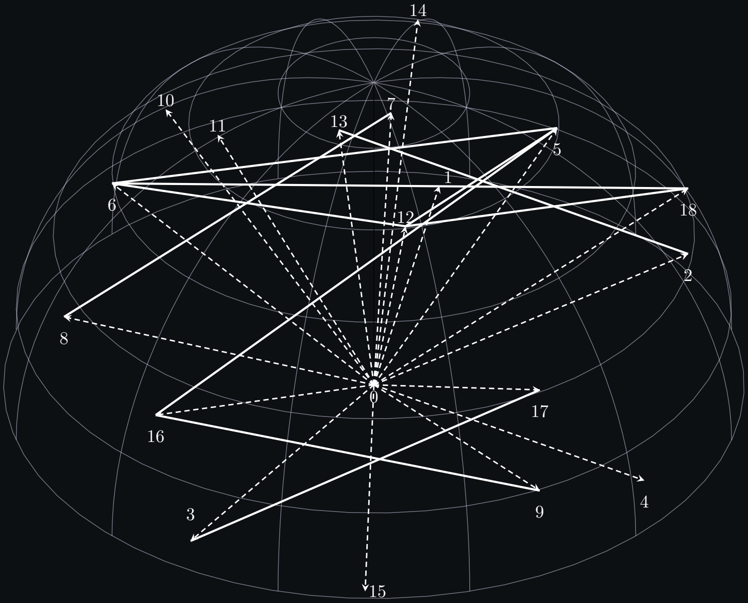
3-Sieve via triangle finding

Connect two points $(i, j) \Leftrightarrow |\langle i \, , j \rangle| \approx 1/3$



3-Sieve via triangle finding

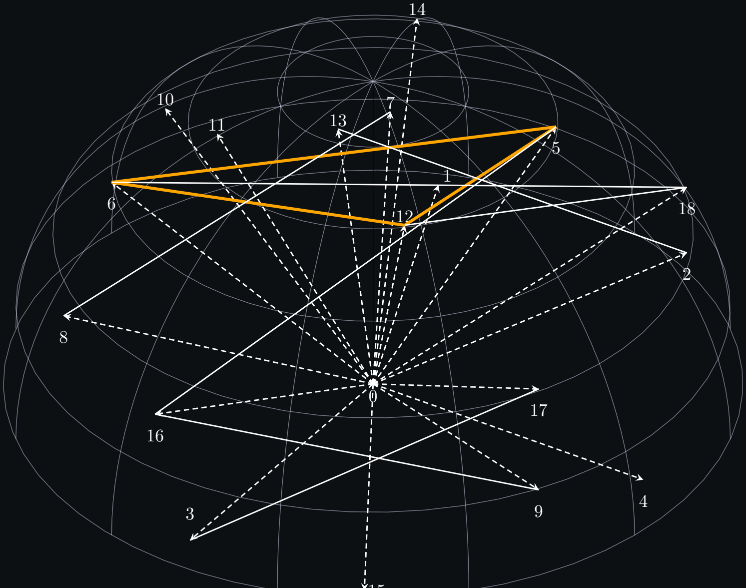
Connect two points $(i, j) \Leftrightarrow |\langle i\ ,\, j \rangle| \approx 1/3$



3-Sieve via triangle finding

Connect two points $(i, j) \Leftrightarrow |\langle i\ ,\ j\rangle| \approx 1/3$

Good triples $(i, j, k) \Leftrightarrow$ triangles



Apply quantum triangle (k-clique) finding

$G = \{V, E\}$, V – lattice vectors, $e(v_i, v_j) \in E \Leftrightarrow |\langle v_i\ , v_j\rangle| \approx 1/3$

Run triangle listing on G (it's a sparse graph!)

Apply quantum triangle (k-clique) finding

$G = \{V, E\}$, V – lattice vectors, $e(v_i, v_j) \in E \Leftrightarrow |\langle v_i\, , v_j \rangle| \approx 1/3$

Run triangle listing on G (it's a sparse graph!)

Vast literature on quantum triangle finding but in the **query** model

Apply quantum triangle (k-clique) finding

$G = \{V, E\}$, V – lattice vectors, $e(v_i, v_j) \in E \Leftrightarrow |\langle v_i\,, v_j \rangle| \approx 1/3$

Run triangle listing on G (it's a sparse graph!)

Vast literature on quantum triangle finding but in the **query** model

Adapt the triangle **finding** algorithm of [Buhrman–de Wolf-Dürr–Heiligman–Høyer–Magniez–Santha]:

Time (find $\triangle$) $= \sqrt{|E|} \implies$ Time (list all $\triangle's$) $= |V|\sqrt{|E|}$

This gives

$$\text{Time}^{\texttt{Quant}} = 2^{0.335n} \quad cf. \quad \text{Time}^{\texttt{Class}} = 2^{0.396n}$$

Apply quantum triangle (k-clique) finding

$G = \{V, E\}$, V – lattice vectors, $e(v_i, v_j) \in E \Leftrightarrow |\langle v_i\, , v_j \rangle| \approx 1/3$

Run triangle listing on G (it's a sparse graph!)

Vast literature on quantum triangle finding but in the **query** model

Adapt the triangle **finding** algorithm of [Buhrman–de Wolf-Dürr–Heiligman–Høyer–Magniez–Santha]:

Time (find $\triangle$) $= \sqrt{|E|} \implies$ Time (list all $\triangle' s$) $= |V| \sqrt{|E|}$

This gives

$$\text{Time}^{\texttt{Quant}} = 2^{0.335n} \quad cf. \quad \text{Time}^{\texttt{Class}} = 2^{0.396n}$$

The algorithm generalises to larger $k = \Theta(1)$ and time-optimal inner product leading to

$$\text{Time}^{\texttt{Quant}} = 2^{0.299n + o(n)} \qquad \text{Memory}^{\star} = 2^{0.139n + o(n)}$$

$\star$ quantumly addressable classical memory

More results and conclusions

- Overall now we have
  - ✓ best $\texttt{Time} \times \texttt{Area}$

$$\text{Time}^{\texttt{Quant}} = 2^{0.299n+o(n)} \qquad \text{Memory} = 2^{0.139n+o(n)}$$

cf.

$$\text{Time}^{\texttt{Class}} = 2^{0.373n+o(n)} \qquad \text{Memory} = 2^{0.186n+o(n)}$$

  - ✓ best $\texttt{Time}$ achieved with $k = 2$

$$\text{Time}^{\texttt{Quant}} = 2^{0.265n+o(n)} \qquad \text{Memory} = 2^{0.265n+o(n)}$$

More results and conclusions

- Overall now we have
  - ✓ best $\texttt{Time} \times \texttt{Area}$

$$\text{Time}^{\texttt{Quant}} = 2^{0.299n+o(n)} \qquad \text{Memory} = 2^{0.139n+o(n)}$$

cf.

$$\text{Time}^{\texttt{Class}} = 2^{0.373n+o(n)} \qquad \text{Memory} = 2^{0.186n+o(n)}$$

  - ✓ best $\texttt{Time}$ achieved with $k = 2$

$$\text{Time}^{\texttt{Quant}} = 2^{0.265n+o(n)} \qquad \text{Memory} = 2^{0.265n+o(n)}$$

- Quantum memory?

  There exists a quantum circuit that implements 2-Sieve of width $2^{0.2075n+o(n)}$ and depth $2^{0.1037n+o(n)}$.

More results and conclusions

- Overall now we have
  - ✓ best $\texttt{Time} \times \texttt{Area}$

$$\mathrm{Time}^{\texttt{Quant}} = 2^{0.299n+o(n)} \qquad \mathrm{Memory} = 2^{0.139n+o(n)}$$

cf.

$$\mathrm{Time}^{\texttt{Class}} = 2^{0.373n+o(n)} \qquad \mathrm{Memory} = 2^{0.186n+o(n)}$$

  - ✓ best $\texttt{Time}$ achieved with $k = 2$

$$\mathrm{Time}^{\texttt{Quant}} = 2^{0.265n+o(n)} \qquad \mathrm{Memory} = 2^{0.265n+o(n)}$$

- Quantum memory?

  There exists a quantum circuit that implements 2-Sieve of width $2^{0.2075n+o(n)}$ and depth $2^{0.1037n+o(n)}$.

Thank you!

?

References I

- [ADRS15] D. Aggarwal, D. Dadush, O. Regev, N. Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete gaussian sampling.
- [ANS18] Y. Aono, P. Q. Nguyen, Y. Shen. Quantum Lattice Enumeration and Tweaking Discrete Pruning
- [BdWDHHMS01] H. Buhrman, R. de Wolf, C.Dürr, M.Heiligman, P. Høyer, F. Magniez, M. Santha. Quantum algorithms for element distinctness.
- [BDGL16] A. Becker, L. Ducas, N. Gama, T. Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving.
- [CCL17] Y. Chen, K. Chung, C. Lai. Space-efficient classical and quantum algorithms for the shortest vector problem
- [HK17] G. Herold, E. Kirshanova. Improved algorithms for the approximate k-list problem in Euclidean norm.
- [HS07] G.Hanrot, D.Stehlé. Improved Analysis of Kannan's Shortest Lattice Vector Algorithm
- [KMPR19] E.Kirshanova, E. Mårtensson, E. W. Postlethwaite, S. Roy Moulik Quantum Algorithms for the Approximate k-List Problem and their Application to Lattice Sieving
- [Laa15] T. Laarhoven. Search problems in cryptography
- [NV08] P. Q. Nguyen, T.Vidick. Sieve algorithms for the shortest vector problem are practical.