Tuple Lattice Sieving

Shi Bai, Thijs Laarhoven and **Damien Stehlé**

IBM Research Zurich and **ENS de Lyon**

August 29th 2016

Main result

Our algorithmic goal: the Shortest Vector Problem (SVP)

Input: $\mathbf{B} \in \mathbb{Z}^{n \times n}$ full rank.
Output: $\mathbf{s} \in \mathbf{B} \cdot \mathbb{Z}^n \setminus \mathbf{0}$ shortest.

Our main result

An algorithm that (heuristically) solves SVP and (heuristically) runs in time $2^{O(n)}$ and **space** $\leq 2^{c \cdot n}$ **with** $c = 0.189 < 0.207$.

Why do we consider this problem?

Finding short lattice vectors is a basic task in computational number theory, cryptanalysis, communications theory, combinatorial optimization, etc.

Main result

Our algorithmic goal: the Shortest Vector Problem (SVP)

Input: $\mathbf{B} \in \mathbb{Z}^{n \times n}$ full rank.
Output: $\mathbf{s} \in \mathbf{B} \cdot \mathbb{Z}^n \setminus \mathbf{0}$ shortest.

Our main result

An algorithm that (heuristically) solves SVP and (heuristically) runs in time $2^{O(n)}$ and **space** $\leq 2^{c \cdot n}$ **with** $c = 0.189 < 0.207$.

Why do we consider this problem?

Finding short lattice vectors is a basic task in computational number theory, cryptanalysis, communications theory, combinatorial optimization, etc.

Main result

Our algorithmic goal: the Shortest Vector Problem (SVP)

Input: $\mathbf{B} \in \mathbb{Z}^{n \times n}$ full rank.
Output: $\mathbf{s} \in \mathbf{B} \cdot \mathbb{Z}^n \setminus \mathbf{0}$ shortest.

Our main result

An algorithm that (heuristically) solves SVP and (heuristically) runs in time $2^{O(n)}$ and **space** $\leq 2^{c \cdot n}$ **with** $c = 0.189 < 0.207$.

Why do we consider this problem?

Finding short lattice vectors is a basic task in computational number theory, cryptanalysis, communications theory, combinatorial optimization, etc.

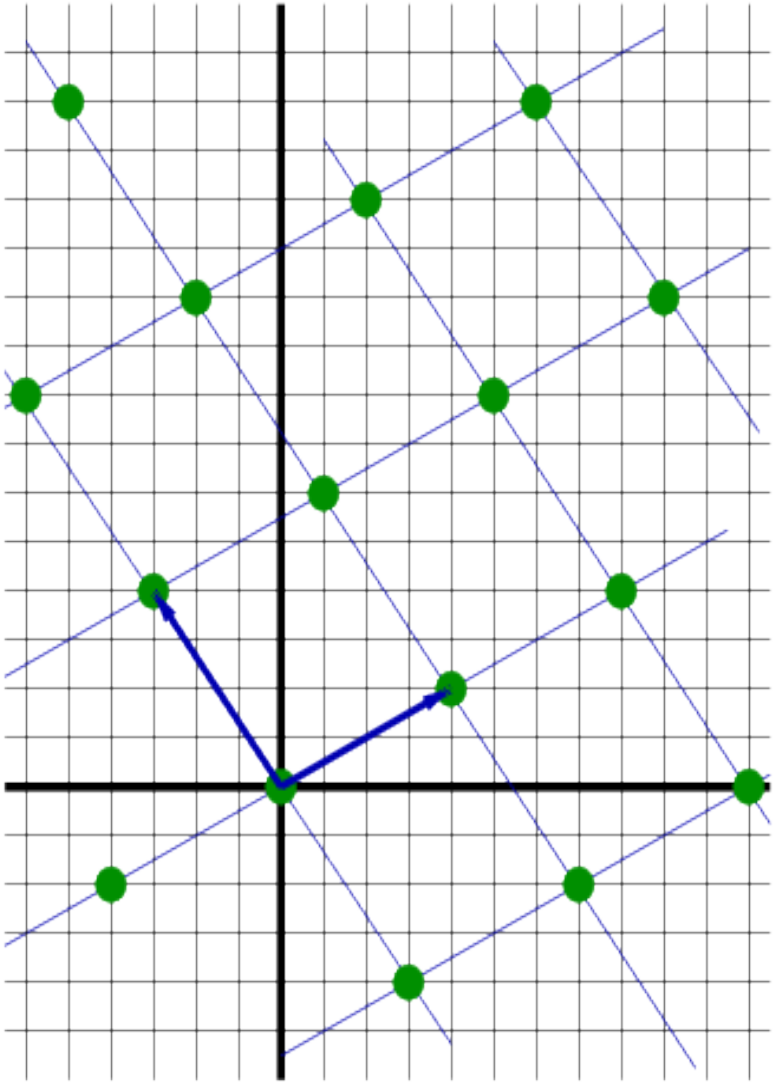
Roadmap

1. **Background**
2. Solving SVP by sieving
3. The new algorithm

Euclidean lattices

Lattice $\Lambda \equiv \left\{\sum_{i\leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\right\}$,
for linearly indep. $\mathbf{b}_i$'s in $\mathbb{R}^n$,
referred to as **lattice basis**

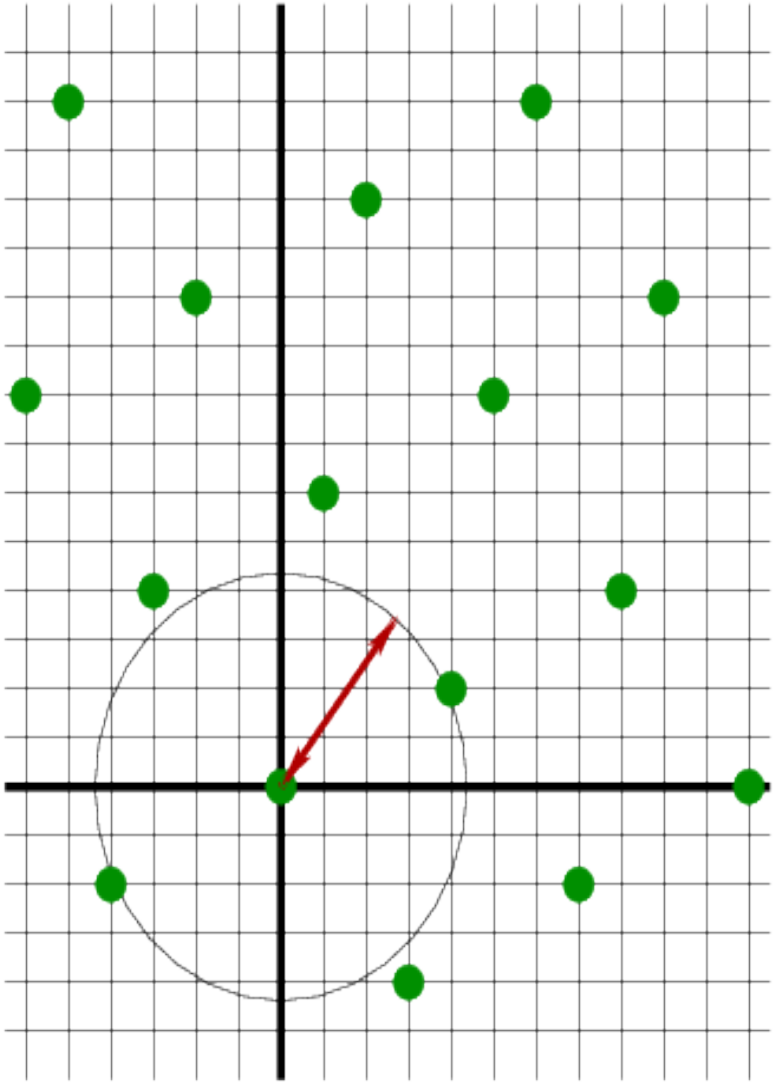
Minimum
$\lambda(\Lambda) = \min\left(\|\mathbf{b}\| : \mathbf{b} \in \Lambda \setminus \mathbf{0}\right)$



Euclidean lattices

Lattice $\Lambda \equiv \left\{\sum_{i\leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\right\}$,
for linearly indep. $\mathbf{b}_i$'s in $\mathbb{R}^n$,
referred to as **lattice basis**

Minimum
$\lambda(\Lambda) = \min\ (\ \|\mathbf{b}\| : \mathbf{b} \in \Lambda \setminus \mathbf{0}\ )$



The Shortest Vector Problem (SVP)

SVP

Input: $\mathbf{B} \in \mathbb{Z}^{n \times n}$ a basis matrix of Λ.
Output: $\mathbf{s} \in \Lambda \setminus \mathbf{0}$ shortest.

- NP hard under randomized reductions [Ajtai98].
- Still so with approximation factor $2^{(\log n)^{1-\varepsilon}}$ [HaRe07].

Best known fully analyzed algorithms

	Time upper bound	Space upper bound	Deterministic or Probabilistic
via enumeration [FiPo'83,Kan'83,HaSt'07]	$n^{n/(2e)+o(n)}$	$\mathcal{P}oly(n)$	Deterministic
via sieving [AjKuSi'81, MiVo'10, PuSt'09]	$2^{2.247n+o(n)}$	$2^{1.325n+o(n)}$	Probabilistic
via Voronoi cell [MiVo'10]	$2^{2n+o(n)}$	$2^{n+o(n)}$	Deterministic
via Gaussians [ADRS'16]	$2^{n+o(n)}$	$2^{n+o(n)}$	Probabilistic

Best known heuristic algorithms

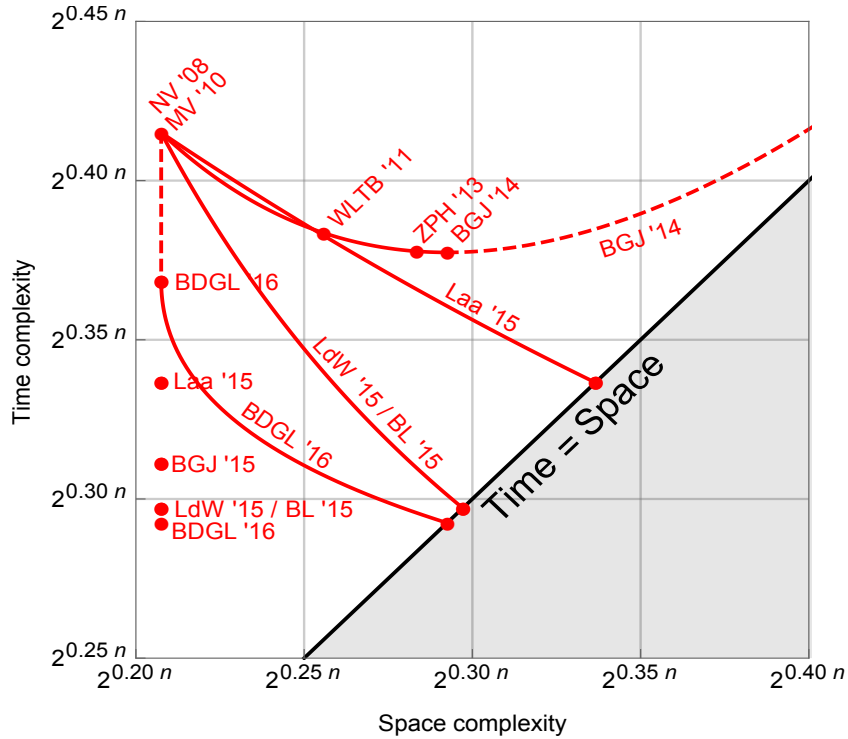
Enumeration with **pre-processing** [Kan'83] and **extreme pruning** [GNR'10].

Best known heuristic algorithms

Enumeration with **pre-processing** [Kan'83] and **extreme pruning** [GNR'10].

Sieving without **perturbations** and with **locality sensitive hashing**.

(Figure courtesy of T. Laarhoven)



Our main contribution

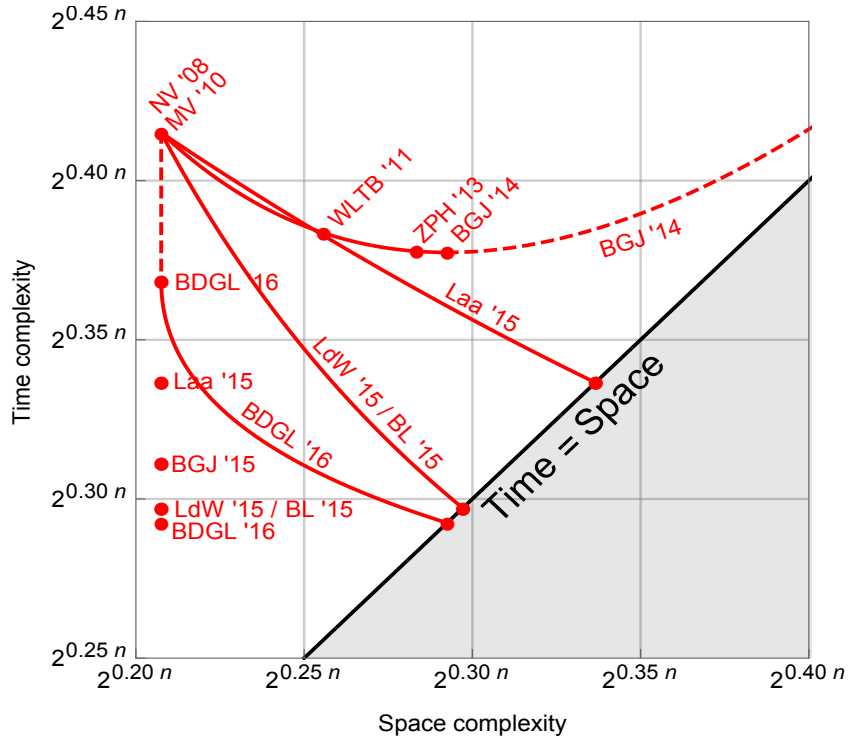
We go to the left of the "Space= $2^{0.207n}$" line.
While keeping a $2^{O(n)}$ space complexity.

Best known heuristic algorithms

Enumeration with **pre-processing** [Kan'83] and **extreme pruning** [GNR'10].

Sieving without **perturbations** and with **locality sensitive hashing**.

(Figure courtesy of T. Laarhoven)



Our main contribution

We go to the left of the "Space= $2^{0.207n}$" line.
While keeping a $2^{O(n)}$ space complexity.

In practice

Enumeration with **extreme pruning** and **pre-processing**.

But... sieving algorithms may beat enumeration for cryptographically relevant costs (e.g., run-time 2^{80} or more).

	$60^\star$	$70^\star$	$80^\star$	Record dimension
"Enumeration"	2	30	360	> 130
"Sieving"	41	910	38000	112

$\star$ Timings in seconds, using fplll-5.0 on a standard laptop

For large costs, the space consumption becomes an issue.
This justifies looking at space-efficient variants of sieving.

Roadmap

1. Background
2. **Solving SVP by sieving**
3. The new algorithm

General principle

Start with a basis of lattice Λ.

1. Sample a large list L of random vectors in Λ.
2. Compute L', the list of $(\mathbf{u}+\mathbf{v})$'s for $\mathbf{u}, \mathbf{v} \in L$ such that $\mathbf{u}+\mathbf{v}$ is short.
3. If some vector in $L' \setminus \mathbf{0}$ is deemed short enough, then output it and stop.
4. Else set $L := L'$ and go back to Step 2.

End up with a short(est) vector in $\Lambda \setminus \mathbf{0}$.

Making it more precise

1. Sample a large list L of random vectors in Λ.

By using Gaussian sampling, the vectors in L have almost the same norm R.

3. Compute L', the list of $(\mathbf{u}+\mathbf{v})$'s for $\mathbf{u},\mathbf{v} \in L$ such that $\mathbf{u}+\mathbf{v}$ is short.

We are happy if $\|\mathbf{u}+\mathbf{v}\| \leq R \cdot (1-\varepsilon)$ for some constant $\varepsilon > 0$.

This ensures that the number of sieves is $O(\log(R/\lambda(\Lambda)))$.

Using LLL preprocessing, this can be made $\mathcal{P}oly(n)$.

4. If some vector in $L' \setminus \mathbf{0}$ is deemed short enough,

For example, we stop if the next sieve produces only $\mathbf{0}$.

There are ways to make this rigorous, but they are costly.

Making it more precise

1. Sample a large list L of random vectors in Λ.

By using Gaussian sampling, the vectors in L have almost the same norm R.

3. Compute L', the list of $(\mathbf{u}+\mathbf{v})$'s for $\mathbf{u},\mathbf{v} \in L$ such that $\mathbf{u}+\mathbf{v}$ is short.

We are happy if $\|\mathbf{u}+\mathbf{v}\| \leq R \cdot (1-\varepsilon)$ for some constant $\varepsilon > 0$.

This ensures that the number of sieves is $O(\log(R/\lambda(\Lambda)))$.

Using LLL preprocessing, this can be made $\mathcal{P}oly(n)$.

4. If some vector in $L' \setminus \mathbf{0}$ is deemed short enough,

For example, we stop if the next sieve produces only $\mathbf{0}$.

There are ways to make this rigorous, but they are costly.

Making it more precise

1. Sample a large list L of random vectors in Λ.

By using Gaussian sampling, the vectors in L have almost the same norm R.

3. Compute L', the list of $(\mathbf{u}+\mathbf{v})$'s for $\mathbf{u},\mathbf{v} \in L$ such that $\mathbf{u}+\mathbf{v}$ is short.

We are happy if $\|\mathbf{u}+\mathbf{v}\| \leq R \cdot (1-\varepsilon)$ for some constant $\varepsilon > 0$.

This ensures that the number of sieves is $O(\log(R/\lambda(\Lambda)))$.

Using LLL preprocessing, this can be made $\mathcal{P}oly(n)$.

4. If some vector in $L' \setminus \mathbf{0}$ is deemed short enough,

For example, we stop if the next sieve produces only $\mathbf{0}$.

There are ways to make this rigorous, but they are costly.

Making it more precise

1. Sample a large list L of random vectors in Λ.

By using Gaussian sampling, the vectors in L have almost the same norm R.

3. Compute L', the list of $(\mathbf{u}+\mathbf{v})$'s for $\mathbf{u},\mathbf{v} \in L$ such that $\mathbf{u}+\mathbf{v}$ is short.

We are happy if $\|\mathbf{u}+\mathbf{v}\| \leq R \cdot (1-\varepsilon)$ for some constant $\varepsilon > 0$.

This ensures that the number of sieves is $O(\log(R/\lambda(\Lambda)))$.

Using LLL preprocessing, this can be made $\mathcal{P}oly(n)$.

4. If some vector in $L' \setminus \mathbf{0}$ is deemed short enough,

For example, we stop if the next sieve produces only $\mathbf{0}$.

There are ways to make this rigorous, but they are costly.

Space complexity of solving SVP by sieving

Space complexity driven $|L|$

We need $L' = (L + L) \cap \mathcal{B}(R \cdot (1 - \varepsilon))$ to be large enough.

- Assume all points in L are i.i.d. uniform on $\mathcal{S}(R)$, the sphere of radius R.
- $\mathbf{u} + \mathbf{v} \in L'$ iff $-\mathbf{u}$ and $\mathbf{v}$ have an angle $\leq \pi/3 - \varepsilon'$.
- Each $\mathbf{u} \in \mathcal{S}$ covers a fraction of the sphere which is

$$\approx \sin^n(\pi/3) = (\sqrt{4/3})^{-n} \approx 2^{-0.207n}.$$

- Assuming the intersection of these cones is negligible, we need $\approx 2^{0.207n}$ of them to cover the whole sphere.

Starting with $\approx 2^{0.207n}$ vectors in the initial list suffices to go through the successive sieves.

Space complexity of solving SVP by sieving

Space complexity driven $|L|$

We need $L' = (L + L) \cap \mathcal{B}(R \cdot (1 - \varepsilon))$ to be large enough.

- Assume all points in L are i.i.d. uniform on $\mathcal{S}(R)$, the sphere of radius R.
- $\mathbf{u} + \mathbf{v} \in L'$ iff $-\mathbf{u}$ and $\mathbf{v}$ have an angle $\leq \pi/3 - \varepsilon'$.
- Each $\mathbf{u} \in \mathcal{S}$ covers a fraction of the sphere which is

$$\approx \sin^n(\pi/3) = (\sqrt{4/3})^{-n} \approx 2^{-0.207n}.$$

- Assuming the intersection of these cones is negligible, we need $\approx 2^{0.207n}$ of them to cover the whole sphere.

Starting with $\approx 2^{0.207n}$ vectors in the initial list suffices to go through the successive sieves.

Space complexity of solving SVP by sieving

Space complexity driven $|L|$

We need $L' = (L + L) \cap \mathcal{B}(R \cdot (1 - \varepsilon))$ to be large enough.

- Assume all points in L are i.i.d. uniform on $\mathcal{S}(R)$, the sphere of radius R.
- $\mathbf{u} + \mathbf{v} \in L'$ iff $-\mathbf{u}$ and $\mathbf{v}$ have an angle $\leq \pi/3 - \varepsilon'$.
- Each $\mathbf{u} \in \mathcal{S}$ covers a fraction of the sphere which is

$$\approx \sin^n(\pi/3) = (\sqrt{4/3})^{-n} \approx 2^{-0.207n}.$$

- Assuming the intersection of these cones is negligible, we need $\approx 2^{0.207n}$ of them to cover the whole sphere.

Starting with $\approx 2^{0.207n}$ vectors in the initial list suffices to go through the successive sieves.

Space complexity of solving SVP by sieving

Space complexity driven $|L|$

We need $L' = (L + L) \cap \mathcal{B}(R \cdot (1 - \varepsilon))$ to be large enough.

- Assume all points in L are i.i.d. uniform on $\mathcal{S}(R)$, the sphere of radius R.
- $\mathbf{u} + \mathbf{v} \in L'$ iff $-\mathbf{u}$ and $\mathbf{v}$ have an angle $\leq \pi/3 - \varepsilon'$.
- Each $\mathbf{u} \in \mathcal{S}$ covers a fraction of the sphere which is

$$\approx \sin^n(\pi/3) = (\sqrt{4/3})^{-n} \approx 2^{-0.207n}.$$

- Assuming the intersection of these cones is negligible, we need $\approx 2^{0.207n}$ of them to cover the whole sphere.

Starting with $\approx 2^{0.207n}$ vectors in the initial list suffices to go through the successive sieves.

Time complexity of solving SVP by sieving

Space complexity

Starting with $\approx 2^{0.207n}$ vectors in the initial list suffices to go through the successive sieves.

Time complexity of naive sieve: $\approx 2^{0.414n}$.

There have been refinements over the naive sieve (e.g., based on locality sensitive hashing).

But, before our work, no improvement on the list size.

Roadmap

1. Background
2. Solving SVP by sieving
3. **The new algorithm**

Main idea

Instead of using pairs to get shorter vectors, use triples!

$$\begin{array}{rcl} L' & = & (L+L) \cap \mathcal{B}(R \cdot (1-\varepsilon)) \\ & \Downarrow & \\ L' & = & (L+L+L) \cap \mathcal{B}(R \cdot (1-\varepsilon)) \end{array}$$

The rest of the algorithm remains identical.

Naturally, this extends to any $k \geq 2$.

Simplifying assumptions: $\varepsilon = 0$, $R = 1$, $\mathcal{S} = \mathcal{S}(1)$.

Main idea

Instead of using pairs to get shorter vectors, use triples!

$$\begin{array}{rcl} L' & = & (L+L) \cap \mathcal{B}(R \cdot (1-\varepsilon)) \\ & \Downarrow & \\ L' & = & (L+L+L) \cap \mathcal{B}(R \cdot (1-\varepsilon)) \end{array}$$

The rest of the algorithm remains identical.

Naturally, this extends to any $k \geq 2$.

Simplifying assumptions: $\varepsilon = 0$, $R = 1$, $\mathcal{S} = \mathcal{S}(1)$.

Main idea

Instead of using pairs to get shorter vectors, use triples!

$$\begin{array}{rcl} L' & = & (L+L) \cap \mathcal{B}(R \cdot (1-\varepsilon)) \\ & \Downarrow & \\ L' & = & (L+L+L) \cap \mathcal{B}(R \cdot (1-\varepsilon)) \end{array}$$

The rest of the algorithm remains identical.

Naturally, this extends to any $k \geq 2$.

Simplifying assumptions: $\varepsilon = 0,\ R = 1,\ \mathcal{S} = \mathcal{S}(1)$.

Cost analysis

Naive **triple sieve** has a cost that is cubic in $|L|$.

Double sieve: saturate the sphere using vectors

How many i.i.d. uniform vectors on $\mathcal{S}$ do we need to cover $\mathcal{S}$? Any vector of $\mathcal{S}$ must be at angle $\leq \pi/3$ of one of these vectors.

$$\Rightarrow (\sqrt{4/3})^n$$

Triple sieve: saturate the sphere using vector pairs

How many i.i.d. uniform vectors on $\mathcal{S}$ do we need to cover $\mathcal{S}$, **using pairs**? Any vector of $\mathcal{S}$ must be at angle $\leq \pi/3$ of the sum of two of these vectors.

Saturating the sphere using **pairs**

Triple sieve: saturate the sphere using vector pairs

How many i.i.d. uniform vectors on $\mathcal{S}$ do we need to cover $\mathcal{S}$, **using pairs**? Any vector of $\mathcal{S}$ must be at angle $\leq \pi/3$ of the sum of two of these vectors.

Let $\mathbf{u}, \mathbf{v}$ be two vectors of $\mathcal{S}$.

- $\mathcal{S} \cap \mathcal{B}(\mathbf{u}+\mathbf{v}, 1)$ is larger when $\mathbf{u}+\mathbf{v}$ is short.

$\Rightarrow$ we want $\mathbf{u}$ and $\mathbf{v}$ to be far from orthogonal.

- But... random vectors are almost orthogonal: angle density function $\approx (\sin\theta)^n$.

Saturating the sphere using **pairs**

Triple sieve: saturate the sphere using vector pairs

How many i.i.d. uniform vectors on $\mathcal{S}$ do we need to cover $\mathcal{S}$, **using pairs**? Any vector of $\mathcal{S}$ must be at angle $\leq \pi/3$ of the sum of two of these vectors.

Let $\mathbf{u}, \mathbf{v}$ be two vectors of $\mathcal{S}$.

- $\mathcal{S} \cap \mathcal{B}(\mathbf{u}+\mathbf{v}, 1)$ is larger when $\mathbf{u}+\mathbf{v}$ is short.

$\Rightarrow$ we want $\mathbf{u}$ and $\mathbf{v}$ to be far from orthogonal.

- But... random vectors are almost orthogonal: angle density function $\approx (\sin\theta)^n$.

Saturating the sphere using **pairs**

Triple sieve: saturate the sphere using vector pairs

How many i.i.d. uniform vectors on $\mathcal{S}$ do we need to cover $\mathcal{S}$, **using pairs**? Any vector of $\mathcal{S}$ must be at angle $\leq \pi/3$ of the sum of two of these vectors.

Let $\mathbf{u}, \mathbf{v}$ be two vectors of $\mathcal{S}$.

- $\mathcal{S} \cap \mathcal{B}(\mathbf{u}+\mathbf{v}, 1)$ is larger when $\mathbf{u}+\mathbf{v}$ is short.

$\Rightarrow$ we want $\mathbf{u}$ and $\mathbf{v}$ to be far from orthogonal.

- But... random vectors are almost orthogonal: angle density function $\approx (\sin\theta)^n$.

Triple sieve is less cost-intensive than double sieve

- A single vector covers more of $\mathcal{S}$ than a sum of two vectors (in most cases).
- A single vector in double sieve is more "powerful" than a sum of two vectors in triple sieve.
- But there are fare more vector pairs than single vectors!

$$\begin{array}{ccc} (4/3)^{n/2} & \rightarrow & (27/16)^{n/4} \\ \wr & & \wr \\ 2^{0.208n} & \rightarrow & 2^{0.189n} \end{array}$$

Triple sieve is less cost-intensive than double sieve

- A single vector covers more of $\mathcal{S}$ than a sum of two vectors (in most cases).
- A single vector in double sieve is more "powerful" than a sum of two vectors in triple sieve.
- But there are fare more vector pairs than single vectors!

$$\begin{array}{ccc} (4/3)^{n/2} & \rightarrow & (27/16)^{n/4} \\ \wr\wr & & \wr\wr \\ 2^{0.208n} & \rightarrow & 2^{0.189n} \end{array}$$

What about time?

Space complexity:

$$\begin{array}{ccc} (4/3)^{n/2} & \rightarrow & (27/16)^{n/4} \\ \wr\wr & & \wr\wr \\ 2^{0.208n} & \rightarrow & 2^{0.189n} \end{array}$$

Time of **naive sieving** grows from $2^{0.416n}$ for double sieve to $2^{0.576n}$ for triple sieve.

Filtered sieving

- The $(\mathbf{u}, \mathbf{v}) \in L^2$ that are most likely to be used in triple sieve are those whose angle is $\approx \arccos(1/3)$.

$\Rightarrow$ Shorten an element of L only against pairs $(\mathbf{u}, \mathbf{v}) \in L^2$ whose angles are $\approx \arccos(1/3)$.

- This leads to a $2^{0.482n}$ time bound.

What about time?

Space complexity:

$$\begin{array}{ccc} (4/3)^{n/2} & \rightarrow & (27/16)^{n/4} \\ \wr\wr & & \wr\wr \\ 2^{0.208n} & \rightarrow & 2^{0.189n} \end{array}$$

Time of **naive sieving** grows from $2^{0.416n}$ for double sieve to $2^{0.576n}$ for triple sieve.

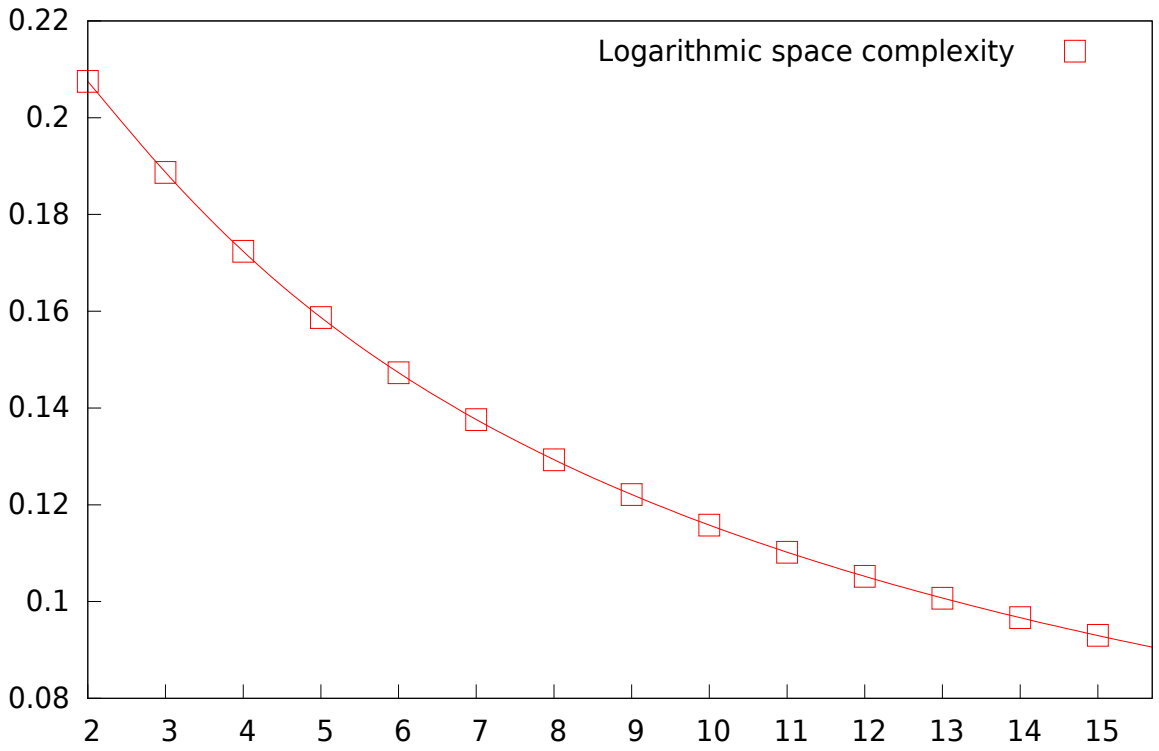
Filtered sieving

- The $(\mathbf{u}, \mathbf{v}) \in L^2$ that are most likely to be used in triple sieve are those whose angle is $\approx \arccos(1/3)$.

$\Rightarrow$ Shorten an element of L only against pairs $(\mathbf{u}, \mathbf{v}) \in L^2$ whose angles are $\approx \arccos(1/3)$.

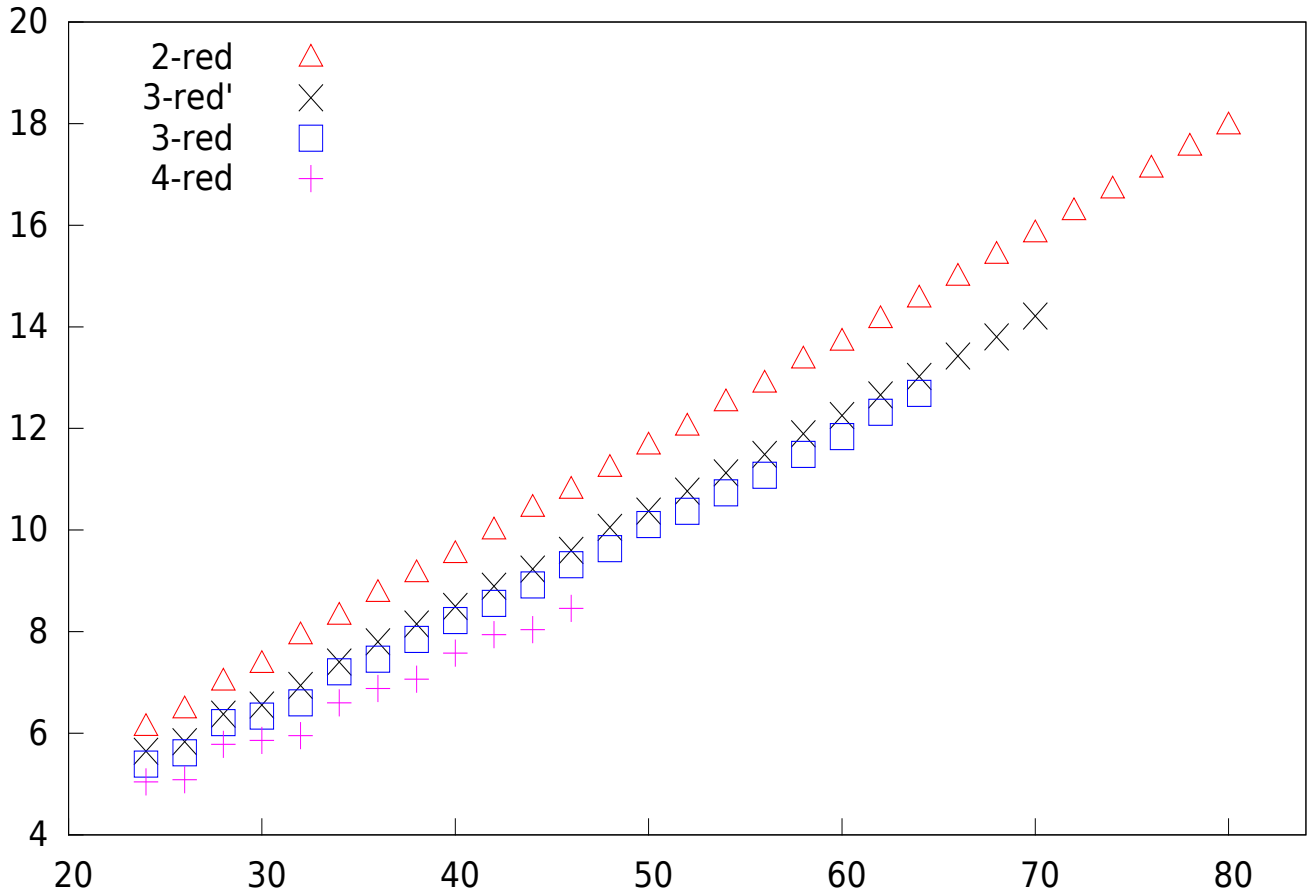
- This leads to a $2^{0.482n}$ time bound.

And larger k?



- Space complexities obtained numerically for $k \geq 4$.
- List size conjectured to decrease as $\approx k^{n/k}$ for large enough k (which is still small compared to n).

Experimentally... it works!



Logarithmic list size (in base 2) versus dimension.

Roadmap

1. Background
2. Solving SVP by sieving
3. The new algorithm

Subsequent work – Herold and Kirshanova

Main observation

For tuple size k, a critical k-tuple of vectors of L forms a regular simplex in dimension $k-1$.

List size

For any k, the list size grows as $\approx \left(\frac{k^{\frac{k}{k-1}}}{k+1} \right)^{n/2}$.

Improved filtering

Filtered triple sieve with space $2^{0.189n}$ and time $2^{0.397n}$.
(This beats naive double sieve.)

Open problems related to efficiency

- Lower the cost, e.g., using locality sensitive hashing. [HeKi16] achieve $2^{0.372n}$: can we do better?
- Parallelize
- How does the analysis extend to non-constant k? Can we get a smooth transition between sieving and enumeration?

When does sieving start beating enumeration?

Open problems related to efficiency

- Lower the cost, e.g., using locality sensitive hashing. [HeKi16] achieve $2^{0.372n}$: can we do better?
- Parallelize
- How does the analysis extend to non-constant k? Can we get a smooth transition between sieving and enumeration?

When does sieving start beating enumeration?

Open problems related to rigour

Remove all the heuristics ...

- Prove correctness
  (i.e., we do not miss all shortest non-zero vectors)
- For the analysis, we assume that list points are i.i.d. uniform on the sphere...
- And that all pairs behave as if they were independent...
- And that the intersection of cones are negligible...

For double-sieve, one relies on the **Kabatiansky-Levenshtein kissing number bound**, to bound the list size.

Is there an analogous result for triples?

Open problems related to rigour

Remove all the heuristics ...

- Prove correctness
  (i.e., we do not miss all shortest non-zero vectors)
- For the analysis, we assume that list points are i.i.d. uniform on the sphere...
- And that all pairs behave as if they were independent...
- And that the intersection of cones are negligible...

For double-sieve, one relies on the **Kabatiansky-Levenshtein kissing number bound**, to bound the list size.

Is there an analogous result for triples?

Open problems related to rigour

Remove all the heuristics ...

- Prove correctness
  (i.e., we do not miss all shortest non-zero vectors)
- For the analysis, we assume that list points are i.i.d. uniform on the sphere...
- And that all pairs behave as if they were independent...
- And that the intersection of cones are negligible...

For double-sieve, one relies on the **Kabatiansky-Levenshtein kissing number bound**, to bound the list size.

Is there an analogous result for triples?