Sieving Algorithms for the Shortest Vector Problem

Bogdan Ursu

École normale supérieure Paris-Saclay

M1 MPRI Internship

bogdanbear@live.com

1^{st} of February - 30^{th} of June

September 8, 2016

Hosting Institution

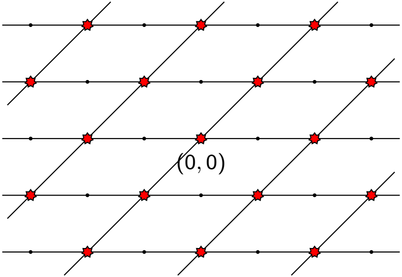
LASEC

- Main activities: research and education.
- Security of communication and information systems.
- Cryptography.

Members:

- Prof. Serge Vaudenay
- 5 PhD students

Lattices



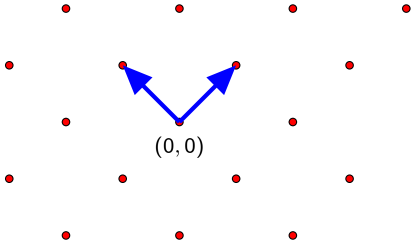
A lattice $\mathcal{L} \subseteq \mathbb{Z}^2$.

Definition

A lattice $\mathcal{L}$ of dimension n is a discrete subgroup of $\mathbb{R}^n$.

Geometrically, $\mathcal{L}$ is the intersection points of a n-dimensional grid.

Lattices



A basis of $\mathcal{L}$.

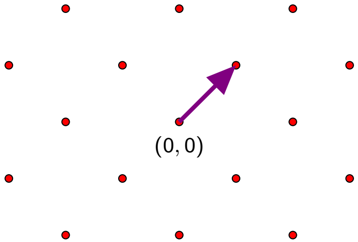
- $\mathcal{L}$ is also the set of all integer combinations of n linearly independent vectors $\{\mathbf{b_1}, ..., \mathbf{b_n}\}$.
- Vectors $\{\mathbf{b_1}, ..., \mathbf{b_n}\}$ will form a basis for $\mathcal{L}$.
- We focus on integral bases.

$$\mathcal{L}(\{\mathbf{b_1}, ..., \mathbf{b_n}\}) \stackrel{def}{=} \{\sum_{i=1}^{n} x_i \mathbf{b_i} : x_i \in \mathbb{Z}\}.$$

Shortest Vector Problem

Definition (Shortest vector problem (SVP))

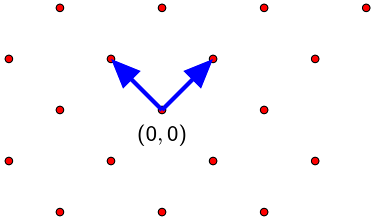
Given a lattice $\mathcal{L} \subseteq \mathbb{R}^n$, find a non-zero lattice vector of minimal length.



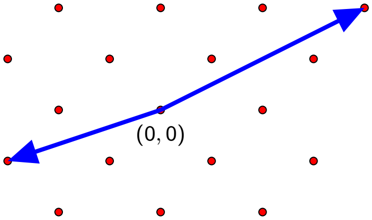
A shortest non-zero vector of $\mathcal{L}$.

The length of the shortest non-zero lattice vector is denoted as $\lambda_1(\mathcal{L})$ - here denoted as λ_1.

Hardness of SVP



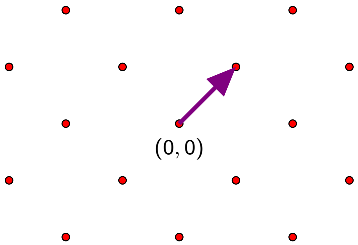
A nice basis of $\mathcal{L}$.



An ugly basis of $\mathcal{L}$.

Hardness of SVP

SVP is *NP*-hard under randomized reductions [1].



A shortest non-zero vector of $\mathcal{L}$.

Motivation

SVP is fundamental to lattice-based crypto, just like integer factorization is for RSA and discrete logarithm for elliptic curves.

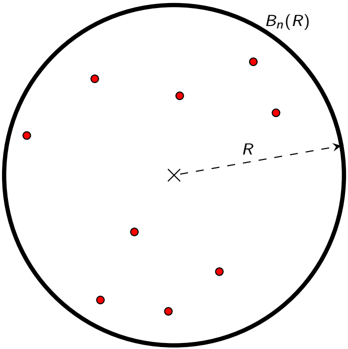
Decision versions of the SVP are at the basis of post-quantum cryptographic primitives [15, 16].

Sieving solvers for the SVP have been introduced in 2001 [2] and continue to be a very active area of current research. Some of the latest papers:

Aug 2015 $\rightarrow$ Present: [3, 5, 4, 10, 11, 13]

Sieving Algorithms (AKS) [2]

- It is possible to sample a large number of lattice vectors in a hyperball $B_n(R)$.
- Radius R is exponentially large, $R = 2^{O(n)}\lambda_1$.
- By computing vector differences, obtain lattice vectors in $B_n(\gamma R)$, where $0 < \gamma < 1$. This process is called sieving.

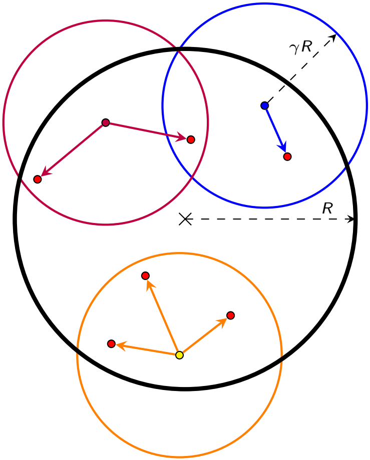


Lattice vectors in hyperball $B_n(R)$ $\longrightarrow$ lattice vectors in $B_n(\gamma R)$.

Algorithm 1 [2]:Sieving

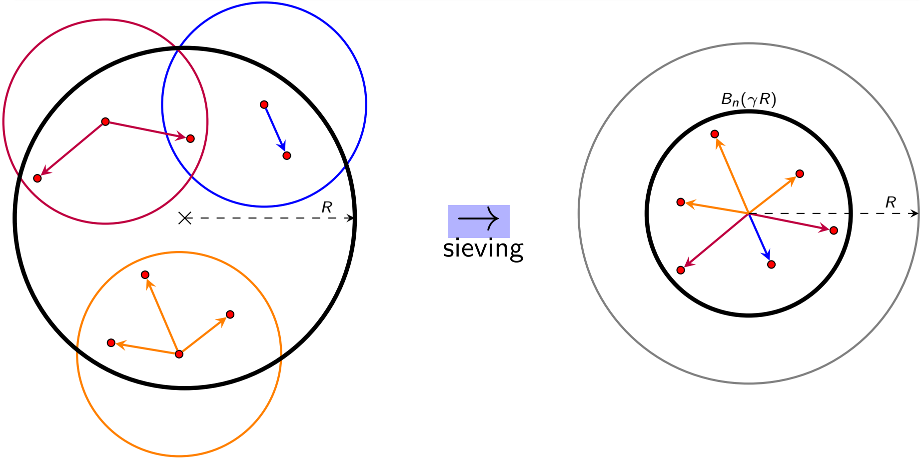
Input: R, $0 < \gamma < 1$ and a list *List* of lattice vectors.

Centers $\leftarrow \emptyset$.
Result $\leftarrow \emptyset$.
for $\mathbf{v} \in$ *List* **do**
  if there exists $\mathbf{c} \in$ *Centers* such that $\|\mathbf{c}-\mathbf{v}\| \leq \gamma R$ **then**:
    Add $\mathbf{v} - \mathbf{c}$ to *Result*
  else add $\mathbf{v}$ to *Centers*.
  end if
end for
return *Result*.



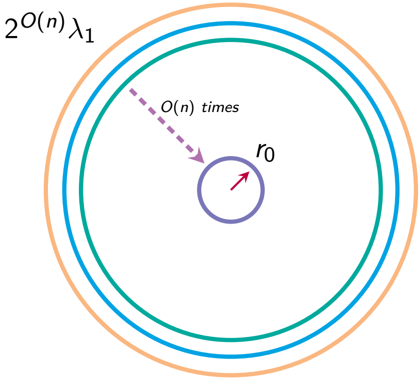
Norm Reduction

Applying the sieving procedure results in a list of lattice vectors in $B_n(\gamma R)$, albeit at the cost of losing the center vectors.



Radius Reduction

As the initial radius is of the form $2^{O(n)}\lambda_1$, applying the sieving procedure $O(n)$ times will yield vectors of norm close to λ_1.



The last radius r_0 will be close to λ_1.

But how to prove correctness?

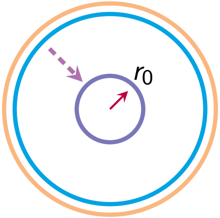
1. Bound the maximal number of centers N_T which can be lost at one iteration of the sieve using the best known upperbounds on the kissing constant [6].

$$N_T \leq 2^{(-log_2(\gamma)+0.401)n+o(n)}.$$

2. Choose initial number of vectors sampled to:

$$\#(sieve\ iterations) \times N_T + 1?$$

Problem: What guarantees that the resulting vectors are non-zero?



Perturbations [2]

Solution: Perturb all lattice vectors $\mathbf{v}$ by real vectors $\mathbf{x}$ in a small hyperball $B_n(\epsilon)$.

When sieving use pairs $(\mathbf{v}, \mathbf{v} + \mathbf{x})$ as in:

Centers $\leftarrow \emptyset$.
for each $(\mathbf{v}, \mathbf{v} + \mathbf{x}) \in \mathit{List}$ **do**
    if there is $(\mathbf{v}', \mathbf{v}' + \mathbf{x}') \in \mathit{Centers}$ such that $\|(\mathbf{v}' + \mathbf{x}') - (\mathbf{v} + \mathbf{x})\| \leq \gamma R$ **then**:
        Add $(\mathbf{v} - \mathbf{v}', (\mathbf{v} + \mathbf{x}) - \mathbf{v}')$ to output.
    else add $(\mathbf{v}, \mathbf{v} + \mathbf{x})$ to *Centers*.
    end if
end for

This sieving procedure preserves perturbations along the execution of the algorithm.

Sampling

Algorithm 2 [2]:Sample

Input: An basis of integral lattice $\mathcal{L}$.
Output: A vector pair $(\mathbf{v}, \mathbf{v}+\mathbf{x}) \in \mathcal{L} \times B_n(R)$.

1: Draw $\mathbf{x}$ uniformly at random from $B_n(\epsilon)$.
2: Compute $\mathbf{v}$ as $v_i = -int(x_i)$ for all $i = \overline{1, n}$.
3: **return** $(\mathbf{v}, \mathbf{v}+\mathbf{x})$.

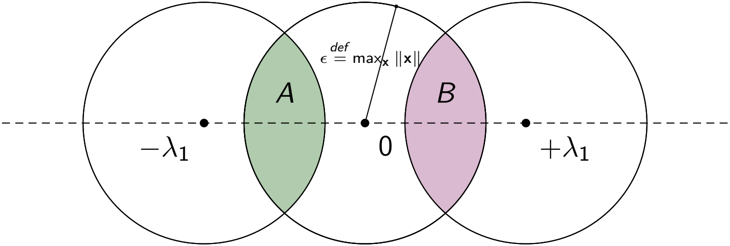
Example

- Sample uniformly a real perturbation $\mathbf{x} = (7.4, -2.3, ..., 192.1)$ from a small hyperball $B_n(\epsilon)$.
- Compute $\mathbf{v} = (-7, 2, ..., -192)$.
- The resulting pair is $((-7, 2, ..., -192), (0.4, -0.3, ..., 0.1))$.

Tossing

Let $\mathbf{s}$ be the shortest vector and apply a tossing argument.

We define a function τ which sends perturbations from A to B and viceversa. For each $\mathbf{x}_i$, decide with probability $\frac{1}{2}$ whether to apply τ or not.

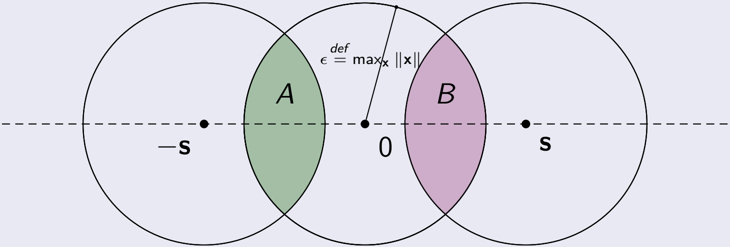


Good perturbations in the case where $\max_{\mathbf{x}} \|\mathbf{x}\| < \lambda_1$

We define a perturbation $\mathbf{x}$ to be good when $\mathbf{x} \in A \cup B$. Similarly, a pair $(\mathbf{v}, \mathbf{v}+\mathbf{x})$ is good iff $\mathbf{x}$ is good.

Tossing

Function τ might be uncomputable.



$\mathbf{s}$ denotes a shortest non-zero vector.

Some properties of the tossing

Algorithm 3 [2]:Sample

Input: An basis of integral lattice $\mathcal{L}$.

Output: A vector pair $(\mathbf{v}, \mathbf{v}+\mathbf{x}) \in \mathcal{L} \times B_n(R)$.

1: Draw $\mathbf{x}$ uniformly at random from $B_n(\epsilon)$.

2: Apply τ with $\frac{1}{2}$ probability if $\mathbf{x} \in A \cup B$.

3: Compute $\mathbf{v}$ as $v_i = -int(x_i)$ for all $i = \overline{1, n}$.

4: **return** $(\mathbf{v}, \mathbf{v}+\mathbf{x})$.

- Consider all pairs $(\mathbf{v}, \mathbf{v}+\mathbf{x})$, applying the τ function will yield $(\mathbf{v} \pm \mathbf{s}, \mathbf{v}+\mathbf{x})$:
- The perturbed vector remains unchanged and so does the distribution of $\mathbf{x}$ over $B_n(\epsilon)$.

Correctness

Extra step: when all sieving is complete, compute all the differences of the obtained lattice vectors and output minimal non-zero one.

Finding a collision $(\mathbf{v}, \mathbf{v} + \mathbf{x})$ and $(\mathbf{v}, \mathbf{v} + \mathbf{x}')$ in the algorithm without tossing $\rightarrow$ Recover $\mathbf{s}$ when using tossing $\rightarrow$ Recover $\mathbf{s}$ without tossing.

$$P[\text{final list contains } (\mathbf{v}, \mathbf{v} + \mathbf{x_1}), (\mathbf{v}, \mathbf{v} + \mathbf{x_2})] \geq \frac{1}{2} P[\text{final list contains } (\mathbf{v} + \mathbf{s}, \mathbf{v} + \mathbf{s} + \mathbf{x_1}), (\mathbf{v}, \mathbf{v} + \mathbf{x_2})].$$

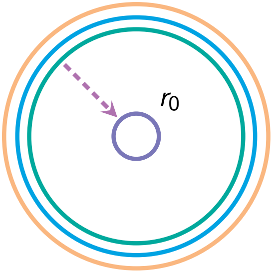
Initial List Size

In order to have a high probability of success, the initial number of vectors sampled will have to be at least:

$$poly(n) \times \frac{1}{P(\mathbf{x} \in A \cup B)} \times \Big((\# sieve\ iterations \times N_T) + N_B + 1 \Big).$$

N_B is the maximal number of vectors in $B_n(r_0)$ we can get without incurring a collision.

$N_B = 2^{(\log_2(r_0)+0.401)n+o(n)}$.



Idea for Improvement

Instead of using a list size like this:

$$poly(n) \times \frac{1}{P(\mathbf{x} \in A \cup B)} \times \Big((\#sieve\ iterations \times N_T) + N_B + 1\Big).$$

Why not use something like this?

$$poly(n) \times \Big((\#sieve\ iterations \times N_T) + \frac{1}{P(\mathbf{x} \in A \cup B)}(N_B + 1)\Big).$$

Center Preselection

Idea: Separate list of pairs into two lists:

- List C will be used to provide centers and reductions.
- List S will be used for providing collisions on good pairs.

Center Preselection, contd [7]

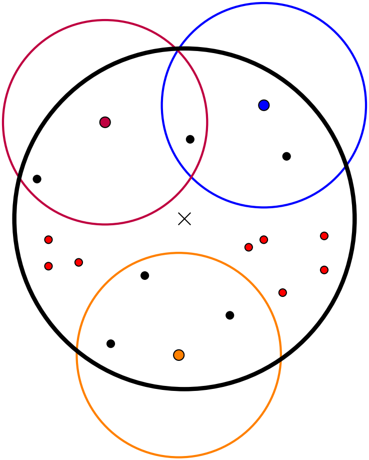
Before sieving, extract some pairs from C and label them as *Centers*.

Algorithm 4 [7]:Sieving

Input: *Centers*, Lists C and S, param. γ and R.

$C' = S' = \emptyset$
for $(\mathbf{v}, \mathbf{v}+\mathbf{x}) \in C$ **do**
    if there is a pair $(\mathbf{v}', \mathbf{v}'+\mathbf{x}') \in$ *Centers* such that $\|\mathbf{v}'+\mathbf{x}' - (\mathbf{v}+\mathbf{x})\| \leq \gamma R$ **then**
        Add $(\mathbf{v}-\mathbf{v}', (\mathbf{v}+\mathbf{x})-\mathbf{v}')$ to C'.
    end if
end for
for $(\mathbf{v}, \mathbf{v}+\mathbf{x}) \in S$ **do**
    if there exists $(\mathbf{v}', \mathbf{v}'+\mathbf{x}') \in$ *Centers* such that $\|\mathbf{v}'+\mathbf{x}' - (\mathbf{v}+\mathbf{x})\| \leq \gamma R$ **then**
        Add $(\mathbf{v}-\mathbf{v}', (\mathbf{v}+\mathbf{x})-\mathbf{v}')$ to S'.
    end if
end for
return C', S'

Supplementary Loss [7]



Red vectors are not centers and will still be lost, as they are not close to any center.

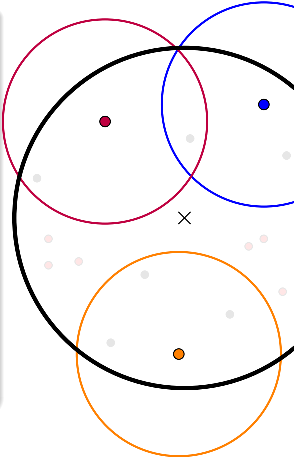
Bounding the Loss [7]

Let $\#Centers = n^2 N_T$.

- Process the $n^2 N_T$ centers sequentially.
- $p_i \stackrel{def}{=}$ probability that the i^{th} center is far away from all previous centers j, with $j < i$.
- $E[\text{far away pairs}] \leq N_T$, which means that:

$$\sum_{i=1}^{n^2 N_T} p_i \leq N_T \implies n^2 N_T \times p_{n^2 N_T} \leq N_T.$$

As i grows, p_i decreases and thus $p_{n^2 N_T} \leq \frac{1}{n^2}$.



Bounding the Loss, contd [7]

We have processed the center pairs sequentially. Non-center pairs are processed sequentially after center preselection, so p_i remains defined the same for them.

$$p_{n^2 N_T} \leq \frac{1}{n^2} \implies P(\text{a pair cannot be reduced}) \leq \frac{1}{n^2}.$$

Bounding the Loss, contd [7]

At the end of all the $O(n)$ sieving steps, we expect to be left with a fraction of $(1 - \frac{1}{n^2})^{O(n)}$ pairs. Then, we lose at most a fraction of

$$1 - \left(1 - \frac{1}{n^2}\right)^{O(n)} \approx O\left(\frac{1}{n}\right) \text{ pairs.}$$

Therefore, the new initial list size must be:

$$poly(n) \times \Big((\#sieve\ iterations \times n\, N_T) + \frac{\text{poly(n)}}{P(\mathbf{x} \in A \cup B)} (N_B + 1) \Big).$$

Does It Work?

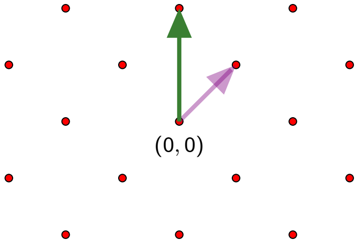
Optimizing the complexity of the algorithm and using the birthday paradox for collisions as used in [7], the optimal time complexity obtained goes from $2^{2.648n+o(n)}$ to $\boxed{2^{2.4847n+o(n)}}$.

Unfortunately, the state-of-the-art is $\boxed{2^{2.465n+o(n)}}$.

Generalization of the SVP

Definition (μ-Approximate SVP or μSVP)

Given a lattice $\mathcal{L} \subseteq \mathbb{R}^n$, find a non-zero lattice vector of length at most $\mu\lambda_1$.



A $\sqrt{2}$ approximation of λ_1.

NP-hardness for $\mu = \sqrt{n}$ would imply *NP* =co-*NP*, and for $\mu = \sqrt{n/O(\log(n))}$ the implication would be that co-$NP \subseteq AM$. [8]

Impact

Our algorithm starts over-performing the state-of-the-art at around $\mu \approx 2.71$.

From $\mu \approx 3.37$, it achieves both lower time and lower space complexities.

Time comparison with previously known bounds

	Modified ListSieve[12]	μListSieve-Birthday[17]	new
μ	c_{time}	c_{time}	c_{time}
2.71	1.99758	2.36552	1.75721
3.61	1.77978	1.99933	1.47898
8	1.36434	1.42456	1.0868
15	1.16723	1.19071	0.951187
100	0.903217	0.904852	0.824121

Comparison of previously known time complexity bounds and our new analysis. A table entry c_{time} indicates a time complexity of $2^{c_{time}n+o(n)}$, when $n \to \infty$.

For $\mu = 8$ and n=150, 2^{205} is improved to 2^{165}.

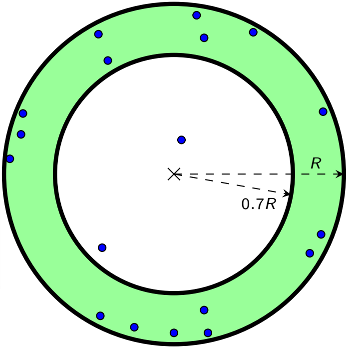
Space comparison with previously known bounds

	Modified ListSieve[12]	μListSieve-Birthday[17]	new
μ	c_{space}	c_{space}	c_{space}
2.71	0.833343	0.922173	0.878607
3.61	0.749856	0.800109	0.739491
8	0.59624	0.608519	0.543402
15	0.526077	0.53057	0.475594
100	0.435	0.435284	0.41206

Comparison of previously known time complexity bounds and our new analysis. A table entry c_{space} indicates a space complexity of $2^{c_{space} n + o(n)}$, when $n \to \infty$.

The NV-Sieve Heuristic[14]

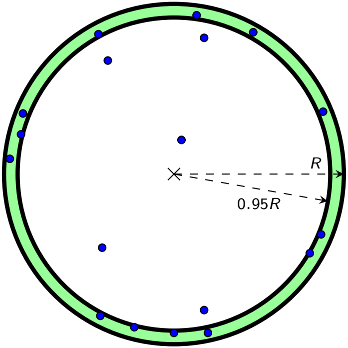
- Sample only lattice vectors, no perturbations.
- Assume that at every step, all lattice vectors are uniformly distributed in the corona.
- Covering the corona optimally requires at most $2^{0.2075n+o(n)}$ vectors and $\gamma \to 1$.



Sufficient to sieve from the corona.

The NV-Sieve Heuristic, contd

- Initial list size: $\#\textit{sieve iterations} \times 2^{0.2075n+o(n)} + 1$.
- Overall complexity $2^{0.4150+o(n)}$.



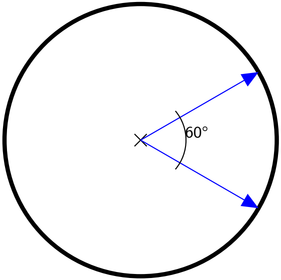
Sufficient to sieve from the corona.

Angular distances

Search for a way to improve quadratic sieve complexity...

When $\gamma \to 1$, we have that:

$$\|\mathbf{v} - \mathbf{w}\| \leq \gamma R \iff \theta(\mathbf{v}, \mathbf{w}) \leq 60°.$$

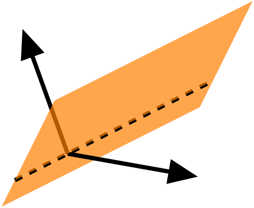


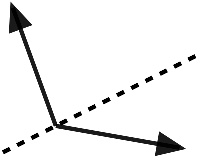
The limiting case for reducing vectors.

Random Hyperplanes

- Use hyperplanes to classify vectors according to common angles.
- Drawing a hyperplane at random will separate two vectors $\mathbf{v}$, $\mathbf{w}$ with probability:

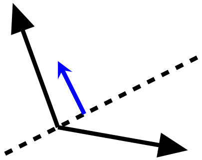
$$\frac{\theta(\mathbf{v}, \mathbf{w})}{\pi}.$$



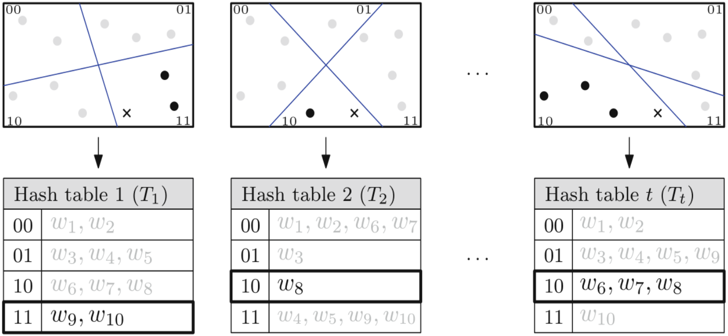


Angular Hashes

- Store a normal vector $\mathbf{n_i}$ for each hyperplane.
- For each vector $\mathbf{v}$, compute $sign\langle \mathbf{n_i}, \mathbf{v} \rangle$ and determine the relative position of $\mathbf{v}$.



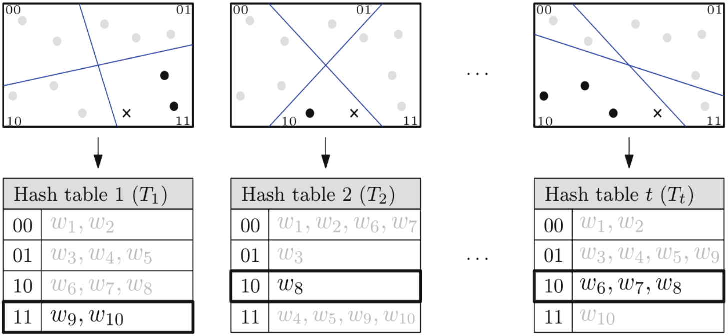
Creating Hashes[9]



Hyperplane hashing[9]

- Two vectors will be close when they collide in at least one hashtable.
- Group hyperplanes in groups of k and put each group in a hashtable.
- Use t hashtables. Time complexity decreases from $2^{0.4151n+o(n)}$ to $2^{0.3366n+o(n)}$.

Improvement Ideas



- Use groups of v hashtables and a collision will mean that the vectors collide in all v hashtables.
- Or start with hashtables and then group them into groups of v.
- Attempt to use hashing for AKS?

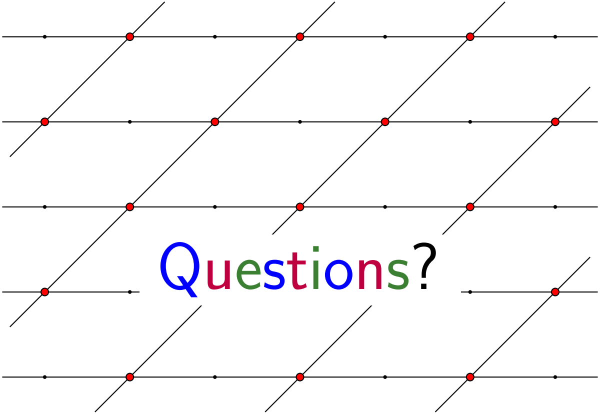
Conclusion and Future Work

Conclusion

- Survey the most important sieving algorithms and heuristics. ✓
- Clean-up the case when perturbation size is larger than λ_1. ✓
- Our improved analysis/algorithm for μSVP is exponentially faster in both time and space than the state-of-the-art. ✓

Future Work

- Control the distribution of lattice vectors.
- Multi-level hash functions might improve nearest neighbour search (NNS).
- Improve tuple sieving with locality-sensitive hashing LSH. [3]


Questions?

References

[1] Miklós Ajtai. The Shortest Vector Problem in L2 is NP-hard for randomized reductions (extended abstract). In *STOC (Symposium on Theory of Computing)*, pages 10–19, New York, NY, USA, 1998. ACM. `http://doi.acm.org/10.1145/276698.276705`.

[2] Miklos Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC (Symposium on Theory of Computing)*, pages 266–275. ACM, July 2001. `https://dl.acm.org/citation.cfm?id=380857`.

[3] Shi Bai, Thijs Laarhoven, and Damien Stehlé. Tuple lattice sieving. Cryptology ePrint Archive, Report 2016/713, 2016. `http://eprint.iacr.org/2016/713`.

[4] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24. `http://epubs.siam.org/doi/abs/10.1137/1.9781611974331.ch2`.

[5] Anja Becker and Thijs Laarhoven. Efficient (ideal) lattice sieving using cross-polytope LSH. In *AFRICACRYPT*, pages 3–23, 2016. `http://dx.doi.org/10.1007/978-3-319-31517-1_1`.

[6] V. I. Levenshtein G. A. Kabatiansky. On bounds for packings on a sphere and in space. *Probl. Peredachi Inf.*, 14(1):3–25, 1978. `http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ppi&paperid=1518&option_lang=eng`.

[7] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. *Coding and Cryptology: Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, chapter Algorithms for the Shortest and Closest Lattice Vector Problems. Springer Berlin Heidelberg, 2011. `http://dx.doi.org/10.1007/978-3-642-20901-7_10`.

[8] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, September 2005.

[9] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *CRYPTO 2015*, pages 3–22. Springer Berlin Heidelberg, 2015. `http://dx.doi.org/10.1007/978-3-662-47989-6_1`.

[10] Thijs Laarhoven and Benne de Weger. Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In *LATINCRYPT*, pages 101–118, 2015. `http://dx.doi.org/10.1007/978-3-319-22174-8_6`.

[11] Thijs Laarhoven, Michele Mosca, and Joop Pol. Finding shortest lattice vectors faster using quantum search. *Des. Codes Cryptography*, 77(2-3):375–400, December 2015. `http://dx.doi.org/10.1007/s10623-015-0067-5`.

References

[12] Mingjie Liu, Xiaoyun Wang, Guangwu Xu, and Xuexin Zheng. Shortest lattice vectors in the presence of gaps. Cryptology ePrint Archive, Report 2011/139, 2011. http://eprint.iacr.org/.

[13] Artur Mariano, Thijs Laarhoven, and Christian Bischof. Parallel (probable) lock-free hashsieve: a practical sieving algorithm for the svp. 2015. http://eprint.iacr.org/2015/041.

[14] P. Q. Nguyen and T. Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, pages 181–207, July 2008. ftp://ftp.di.ens.fr/pub/users/pnguyen/JoMC08.pdf.

[15] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *In STOC*, pages 333–342, 2009. http://doi.acm.org/10.1145/1536414.1536461.

[16] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *In STOC*, pages 84–93, 2005.

[17] Wei Wei, Mingjie Liu, and Xiaoyun Wang. Finding shortest lattice vectors in the presence of gaps. In *Topics in Cryptology — CT-RSA 2015*, pages 239–257. Springer International Publishing, April 2015. http://dx.doi.org/10.1007/978-3-319-16715-2_13.

Good and Bad Bases

Lattice reduction algorithms want to obtain bases with short, almost orthogonal vectors.

The orthogonality defect δ is defined as:

$$\delta(\mathcal{L}) \stackrel{def}{=} \frac{\prod_{i=1}^{n} \|b_i\|}{\prod_{i=1}^{n} \|b_i^*\|}.$$

It follows that $\delta(\mathcal{L}) \geq 1$ with equality when orthogonal.

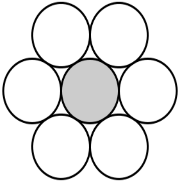
Fundamental Parallelepiped

Input: ϵ, $\mathbf{B} = [\mathbf{b_1}|...|\mathbf{b_n}]$
Output: A vector pair $(\mathbf{v}, \mathbf{v}+\mathbf{x}) \in \mathcal{L} \times B_n(R) \subseteq \mathcal{L} \times \mathbb{R}^n$.

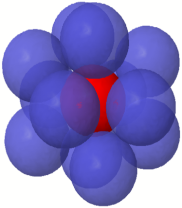
1: Draw $\mathbf{x}$ uniformly at random from $B_n(\epsilon)$
2: Compute $\mathbf{v} = -\mathbf{x} + (\mathbf{x} \mod P(B))$
3: **return** $(\mathbf{v}, \mathbf{v}+\mathbf{x})$

$$frac(x) = \begin{cases} x - floor(x), & \text{for } x \geq 0 \\ x - ceil(x), & \text{for } x < 0. \end{cases}$$

In geometry, the kissing number is defined as the maximal number of non-overlapping unit spheres that can be arranged such that they each touch another given unit sphere.



$n = 2$, the kissing number is 6



$n = 3$, the kissing number is 12

Supplementary slide: GapSVP

GapSVP$_\beta$

The *GapSVP*$_\beta$ problem consists in differentiating the instances when:

- $\lambda \leq 1$.
- $\lambda > \beta$.
- It is a promise problem, so we are allowed to err for instances between 1 and β.

Tuple Sieving[3]

Input: list L of lattice vectors, R and sub-unitary positive γ.

```
Result ← ∅.
for each v, w, x in L do
    if ‖v ± w ± x‖ ≤ γR then
        Add v ± w ± x to Result.
    end if
end for
return Result.
```

Importance of LLL and Lattice Theory outside Crypto

The original applications of LLL were:

- Give polynomial-time algorithms for factorizing polynomials with rational coefficients.
- Finding simultaneous rational approximations to real numbers.
- Solving the integer linear programming problem in fixed dimensions.