A Three-Level Sieve Algorithm for the Shortest Vector Problem

Feng Zhang, Yanbin Pan and Gengran Hu

Key Laboratory of Mathematics Mechanization
Academy of Mathematics and Systems Science, NCMIS, Chinese Academy of Sciences

SAC 2013
08/14/2013

Outline

Outline

What is Lattice?

For any linearly independent vectors $v_1, \cdots, v_m \in \mathbb{R}^n$, the lattice spanned by them is defined as below:

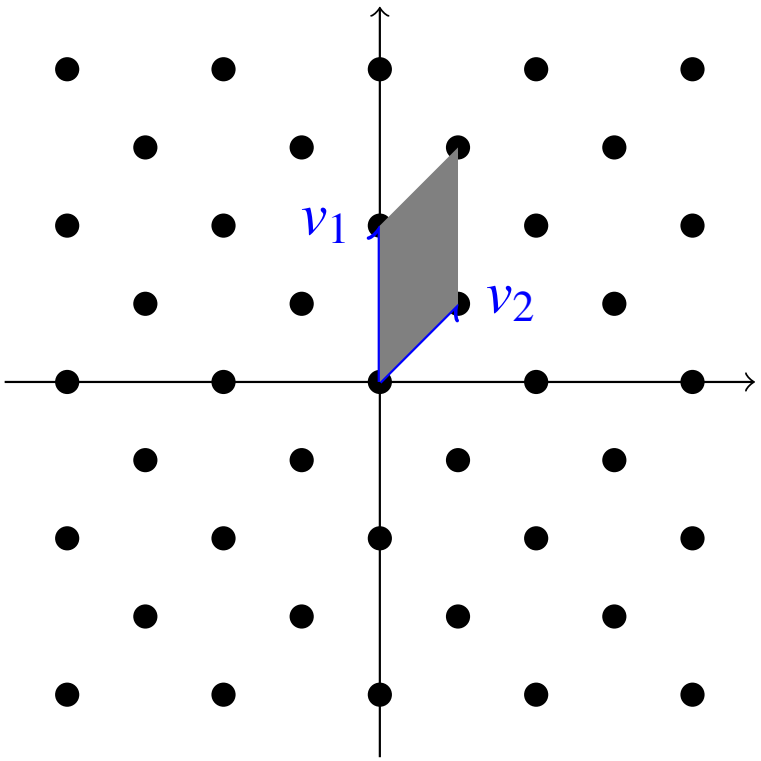
Lattice

$$\mathcal{L}(v_1, \cdots, v_m) = \{\sum_{i=1}^{m} a_i v_i | a_i \in \mathbb{Z}\}.$$

Basis

$v_1, \cdots, v_m \in \mathbb{R}^n$ is a basis.

A lattice has infinite basis.



What is Lattice?

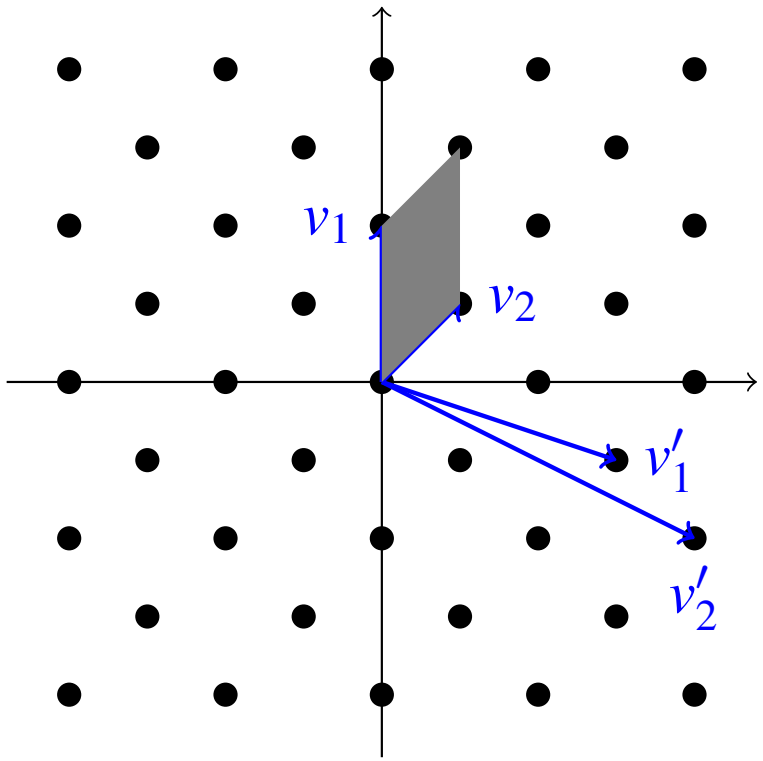
For any linearly independent vectors $v_1, \cdots, v_m \in \mathbb{R}^n$, the lattice spanned by them is defined as below:

Lattice

$$\mathcal{L}(v_1, \cdots, v_m) = \{\sum_{i=1}^{m} a_i v_i | a_i \in \mathbb{Z}\}.$$

Basis

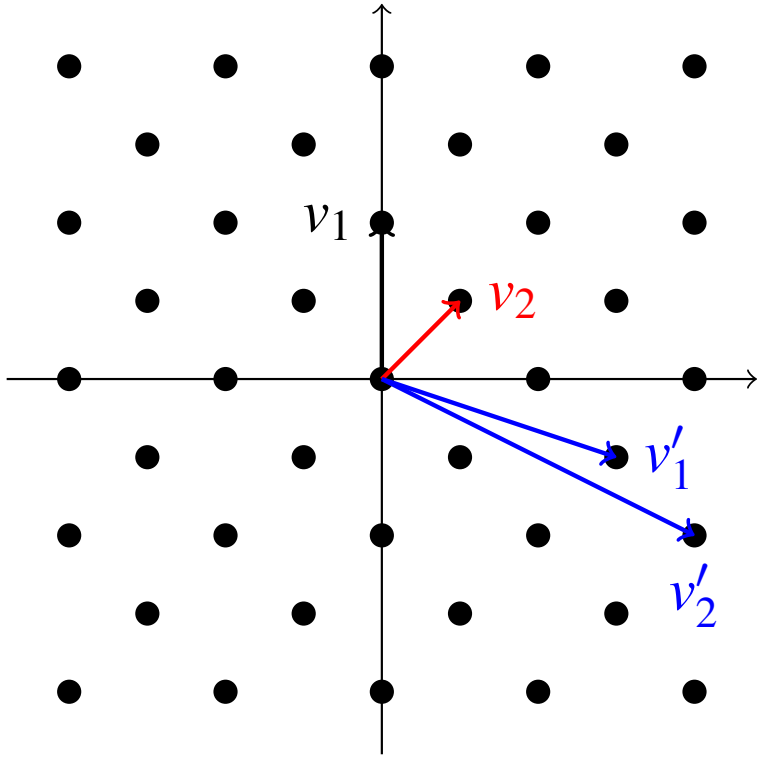
$v_1, \cdots, v_m \in \mathbb{R}^n$ is a basis.
A lattice has infinite basis.



Computational Problems of Lattice

There are two main computational problems:

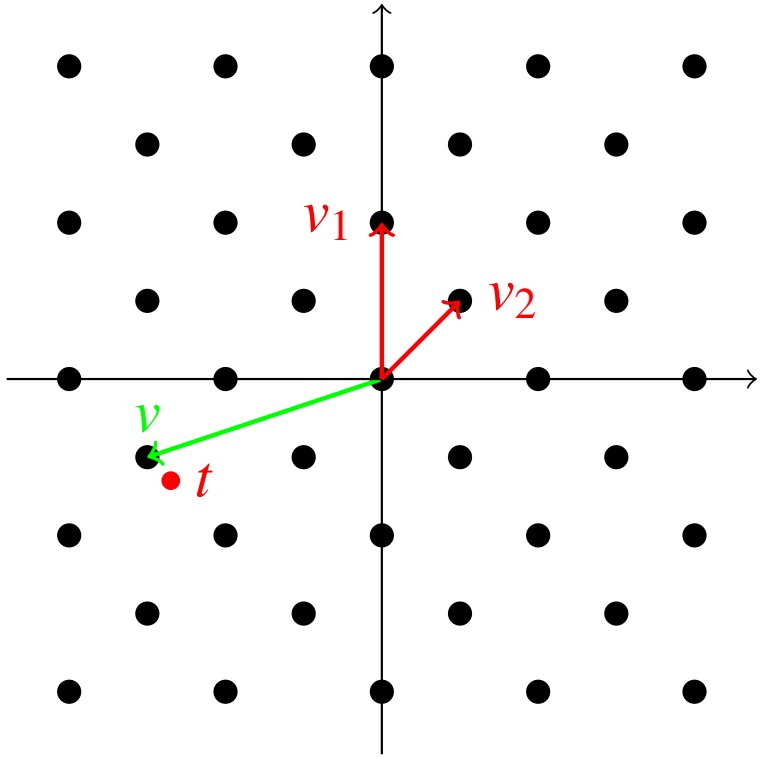
- SVP, which is NP-hard under random reductions.
- CVP, which is in NP-C.



Computational Problem of Lattice

There are two main computational problems:

- SVP, which is NP-hard under random reductions.
- CVP, which is in NP-C.



Outline

Approximation Algorithms to Solve SVP

- LLL Algorithm(exponential factor, polynomial, 1982).
- BKZ Algorithm(Schnorr, blocksize k, 1987).
- Others.

Exact Algorithms to Solve SVP

Algorithm	Time	Space	Remark
Kannan et.al.'s	$2^{O(n \log n)}$	poly(n)	Deterministic
Voronoi Cell-based	$2^{O(cn)}$	$2^{O(cn)}$	Deterministic
Sieve Algorithm	$2^{O(cn)}$	$2^{O(cn)}$	Random

Exact Algorithms to Solve SVP

Algorithm	Time	Space	Remark
Kannan et.al.'s	$2^{O(n \log n)}$	poly(n)	Deterministic
Voronoi Cell-based	$2^{O(cn)}$	$2^{O(cn)}$	Deterministic
Sieve Algorithm	$2^{O(cn)}$	$2^{O(cn)}$	Random

Sieve Algorithms for SVP

- Random sieve: with perturbation

Algorithm	Time	Space
AKS'01	$2^{O(n)}$	$2^{O(n)}$
Regev'04	$2^{16n+o(n)}$	$2^{8n+o(n)}$
NV'08	$2^{5.9n+o(n)}$	$2^{2.95n+o(n)}$
ListSieve'10	$2^{3.2n+o(n)}$	$2^{1.6n+o(n)}$
ListSieve'10(birth.)	$2^{2.465n+o(n)}$	$2^{1.233n+o(n)}$

Heuristic Sieve Algorithms for SVP

- Heuristic sieve: under a natural heuristic assumption

Algorithm	Time	Space
NV'08	$2^{0.415n+o(n)}$	$2^{0.2075n+o(n)}$
WLTB'10(2-level)	$2^{0.3836n+o(n)}$	$2^{0.2557n+o(n)}$
GaussSieve'10	-	$2^{0.41n+o(n)}$

Heuristic Sieve Algorithms for SVP

- Heuristic sieve: under a natural heuristic assumption

Algorithm	Time	Space
NV'08	$2^{0.415n+o(n)}$	$2^{0.2075n+o(n)}$
WLTB'10(2-level)	$2^{0.3836n+o(n)}$	$2^{0.2557n+o(n)}$
GaussSieve'10	-	$2^{0.41n+o(n)}$

Basic Framework

Algorithm 1 Finding short lattice vectors based on sieving

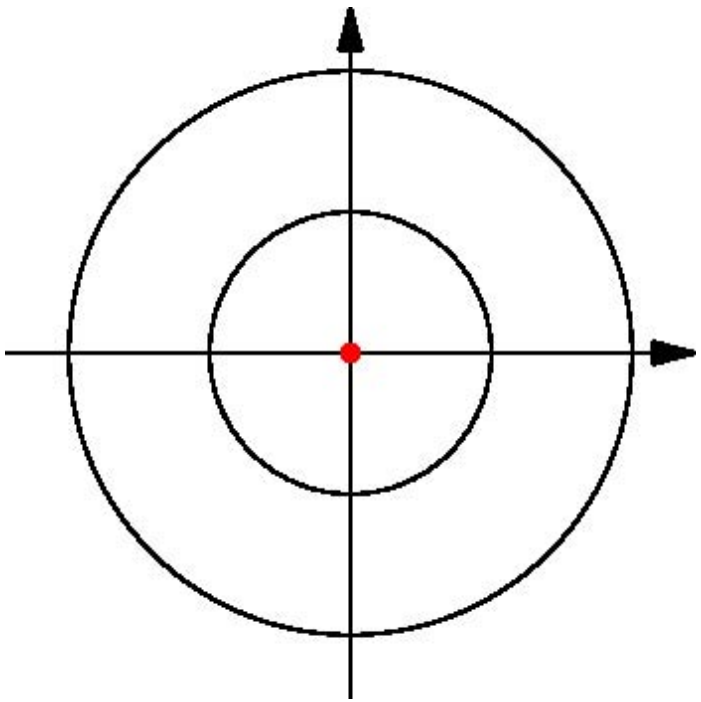
Input: An LLL-reduced basis $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ of a lattice $\mathcal{L}$, sieve factors and a number N.

Output: A short non-zero vector of $\mathcal{L}$.

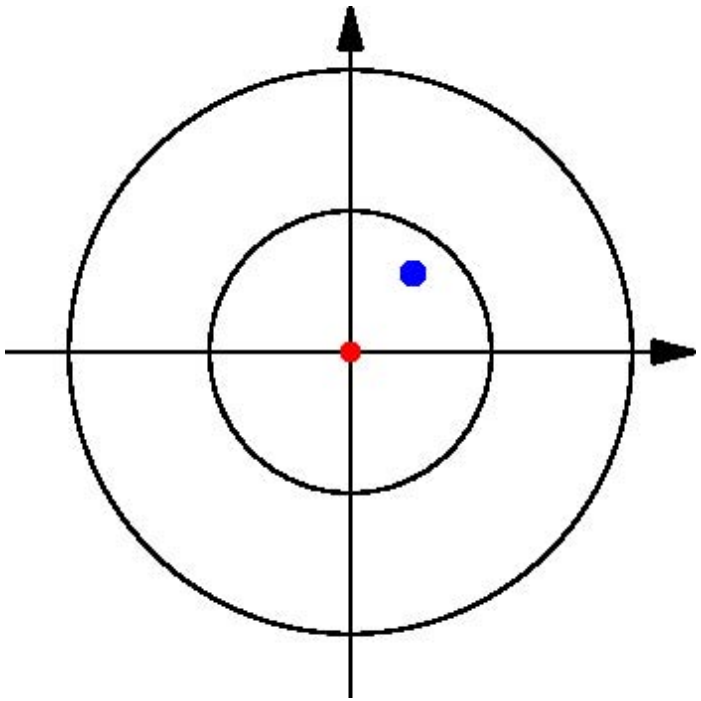
1: $S' \leftarrow \emptyset$
2: **for** $j = 1$ to N **do**
3: $\quad S' \leftarrow S' \cup$ sampling($\boldsymbol{B}$) using Klein's algorithm
4: **end for**
5: Remove all zero vectors from S'
6: **Repeat**
7: $\quad S \leftarrow S'$
8: $\quad S' \leftarrow$ sieve(S, sieve factors) using **Sieve Algorithm**
9: $\quad$ Remove all zero vectors from S'
10: **until** $S' = \emptyset$
11: Compute $\boldsymbol{v}_0 \in S$ such that $\|\boldsymbol{v}_0\| = \min\{\|\boldsymbol{v}\|, \boldsymbol{v} \in S\}$
12: **Return** $\boldsymbol{v}_0$

Main to improve Sieve Algorithm in line 8!

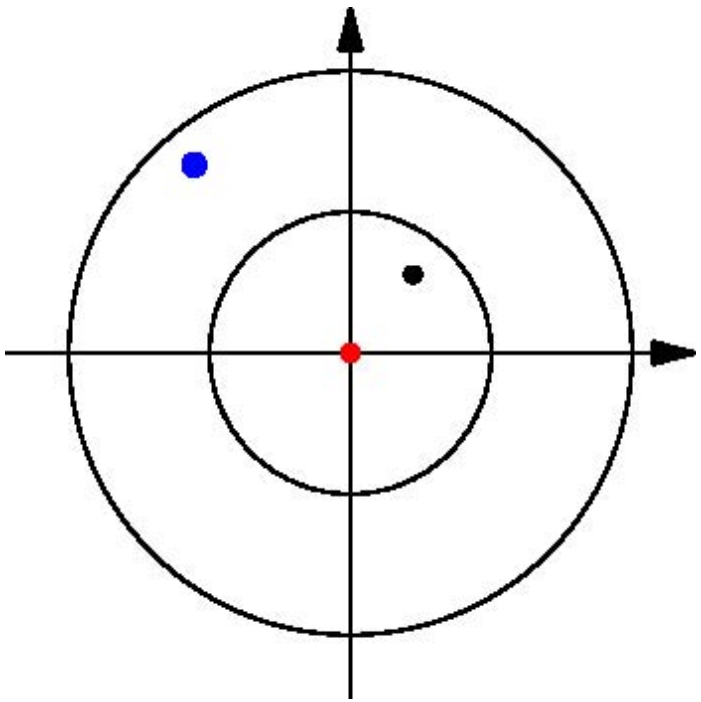
Nguyen-Vidick Sieve Algorithm



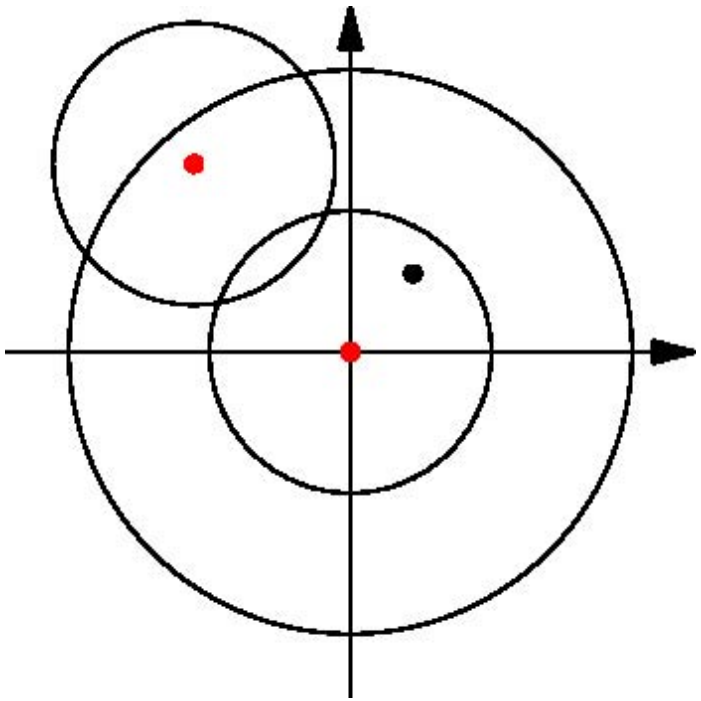
Nguyen-Vidick Sieve Algorithm



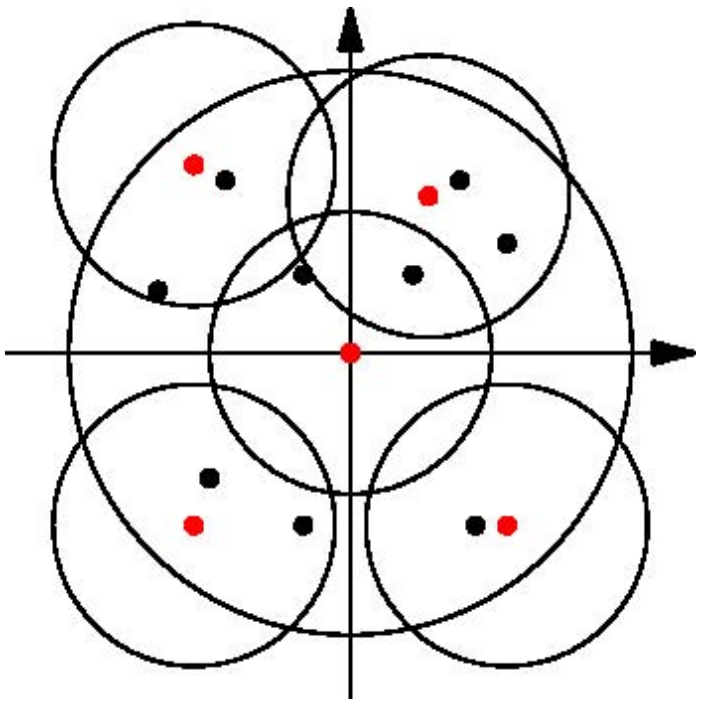
Nguyen-Vidick Sieve Algorithm



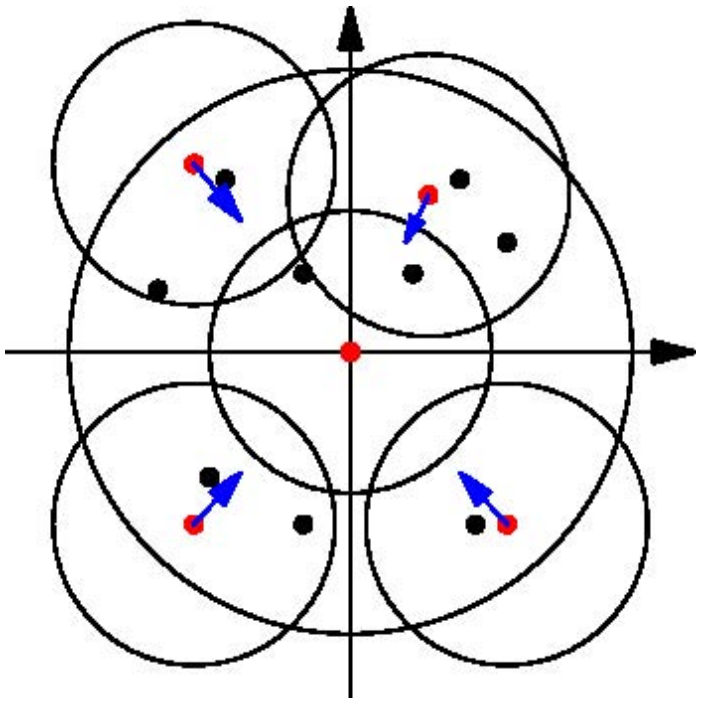
Nguyen-Vidick Sieve Algorithm



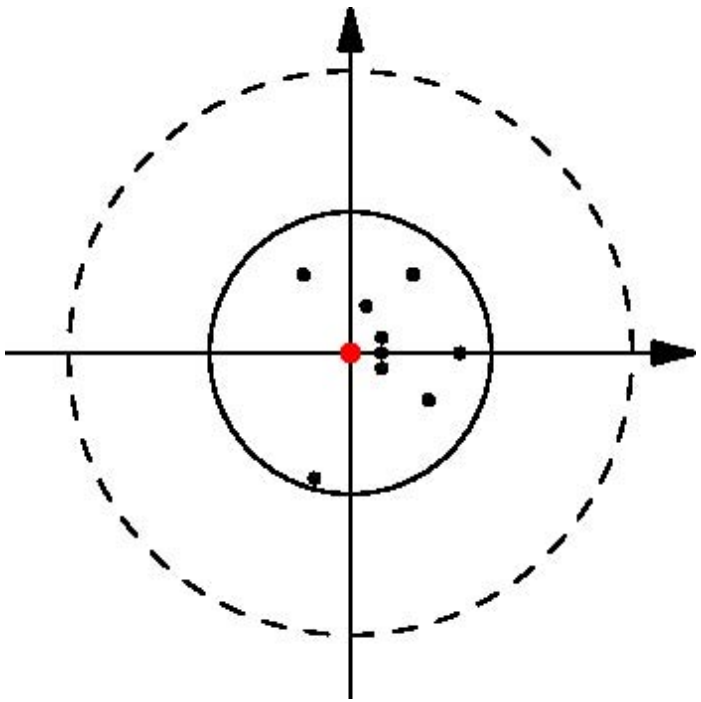
Nguyen-Vidick Sieve Algorithm



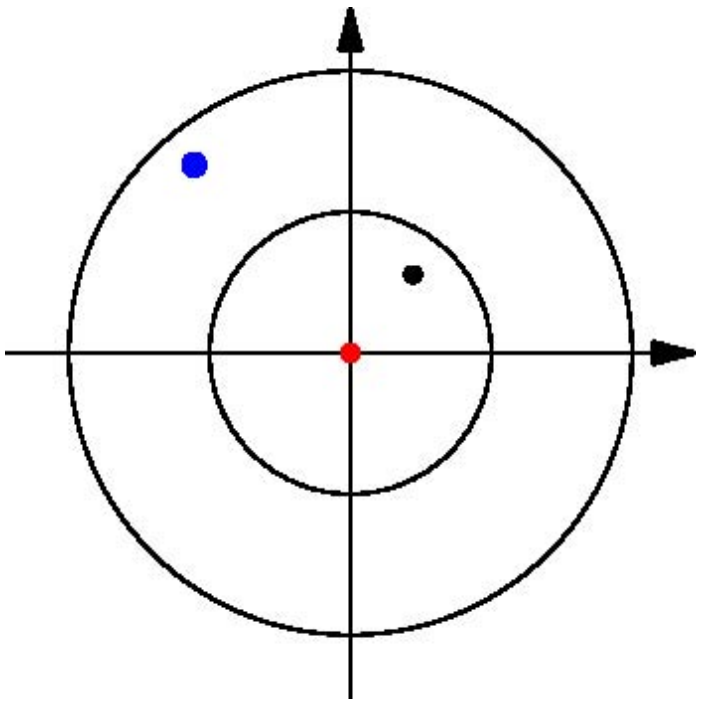
Nguyen-Vidick Sieve Algorithm



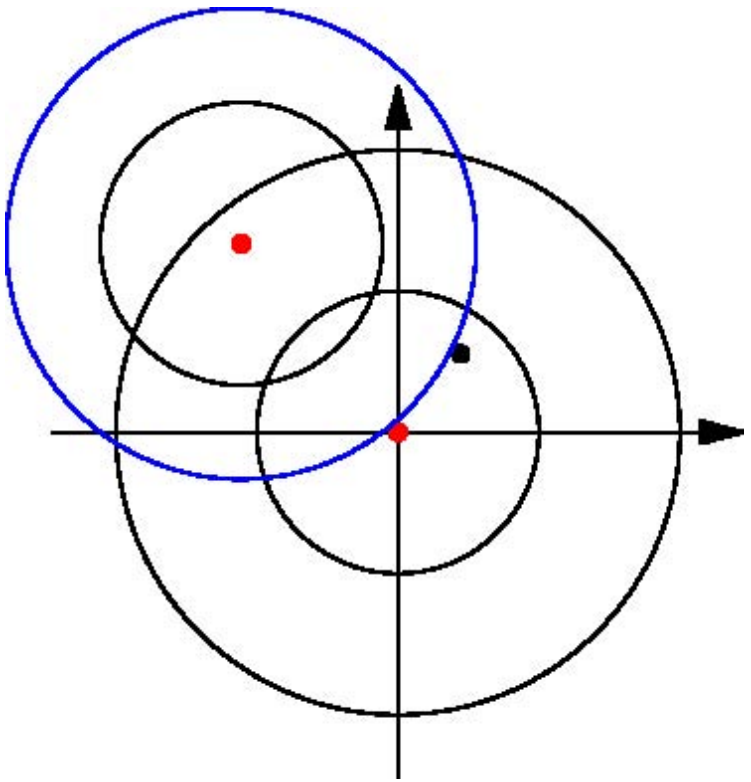
Nguyen-Vidick Sieve Algorithm



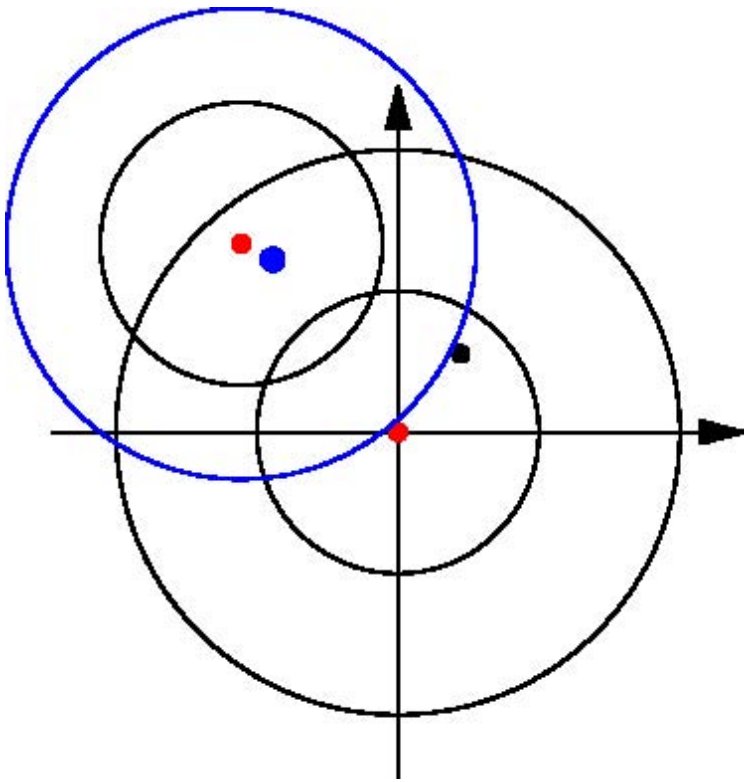
WLTB Sieve Algorithm



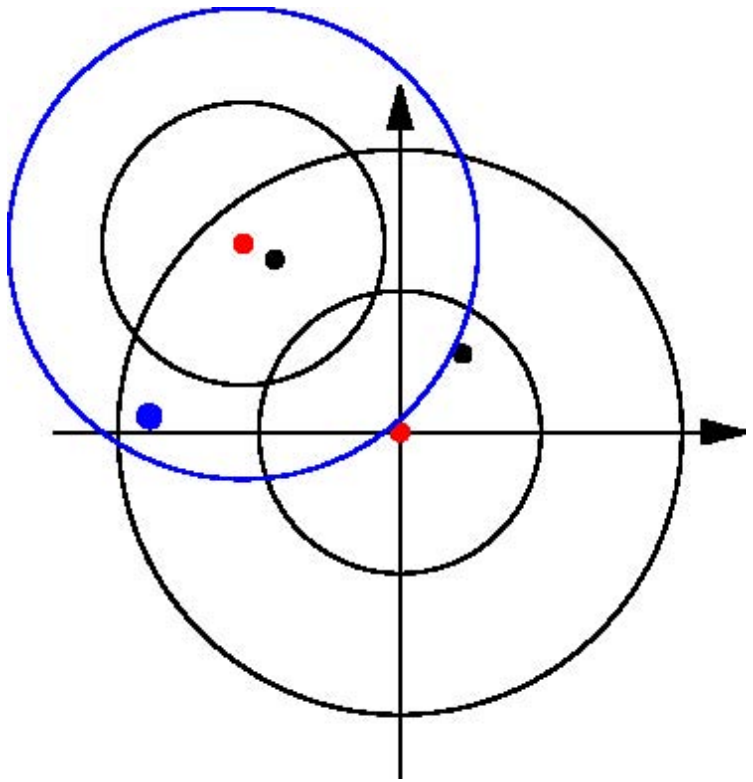
WLTB Sieve Algorithm



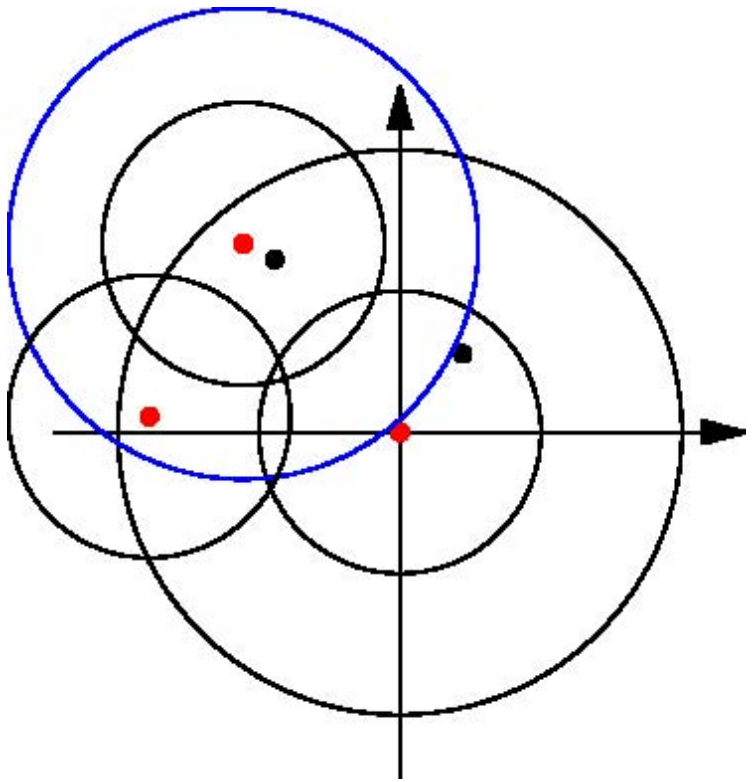
WLTB Sieve Algorithm



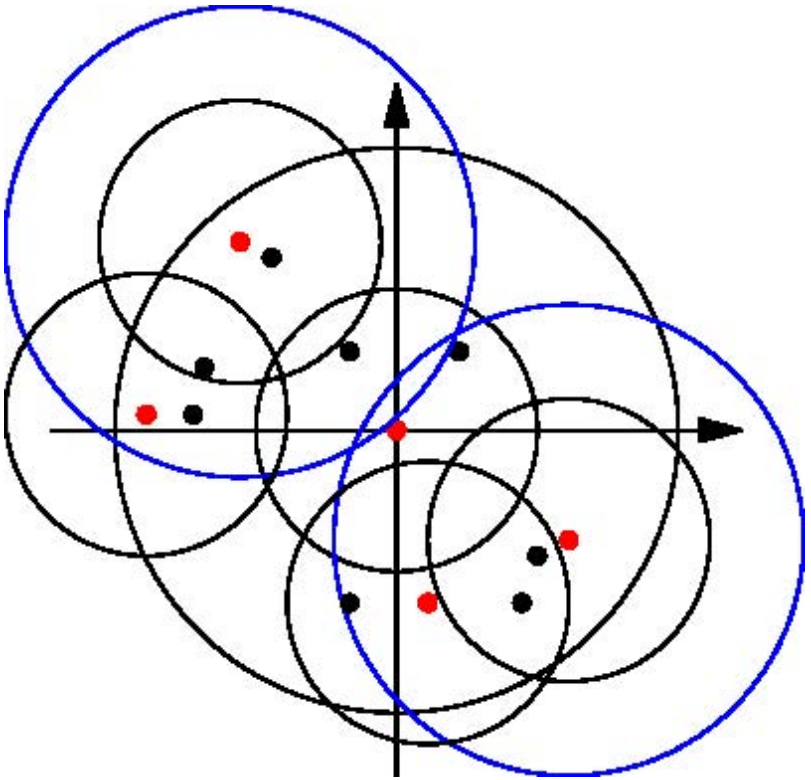
WLTB Sieve Algorithm



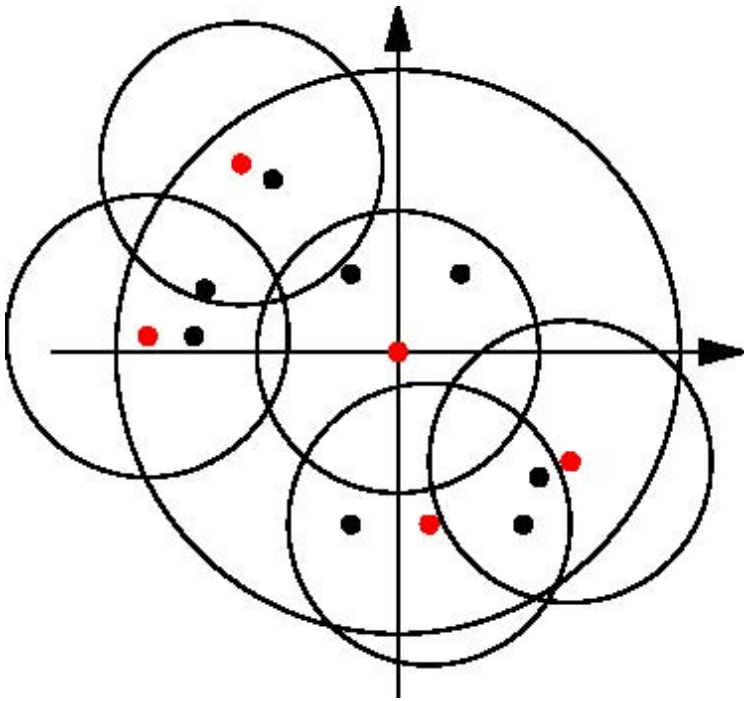
WLTB Sieve Algorithm



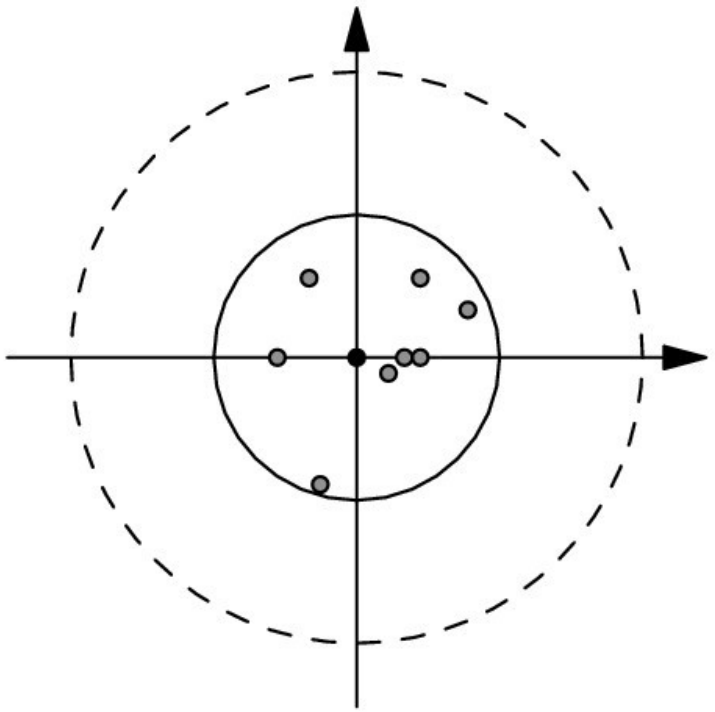
WLTB Sieve Algorithm



WLTB Sieve Algorithm



WLTB Sieve Algorithm



Basic Idea of the Improvement

- The most time consuming part of sieve is to determine which ball a new sample belongs to.
- A natural question : why not take three-level?

Outline

A Three-Level Sieve Algorithm

- Denote by γ_1, γ_2 and γ_3 the sieve factors, where $\gamma_1 > \gamma_2 > 1 > \gamma_3$.
- $C_1 = \{$ center of a big ball covering the spherical shell$\}$.
- $C_2^{\mathbf{c_1}} = \{$ center of a medium ball covering the big ball with center $\mathbf{c_1} \in C_1\}$.
- $C_3^{\mathbf{c_1},\mathbf{c_2}} = \{$ center of a small ball covering the medium ball with center $\mathbf{c_2} \in C_2^{\mathbf{c_1}}\}$.

A Three-Level Sieve Algorithm

Algorithm 2 A three-level sieve algorithm

Input: A subset $S \subseteq B_n(R)$ of vectors in a lattice $\mathcal{L}$ where $R \leftarrow \max_{\boldsymbol{v}\in S} \|\boldsymbol{v}\|$ and sieve factors $0.88 < \gamma_3 < 1 < \gamma_2 < \gamma_1 < \sqrt{2}\gamma_3$.

Output: A subset $S' \subseteq B_n(\gamma_3 R) \cap \mathcal{L}$.

$S' \leftarrow \emptyset, C_1 \leftarrow \emptyset$.
for $\boldsymbol{v} \in S$ **do**
  if $\|\boldsymbol{v}\| \leq \gamma_3 R$ **then**
    $S' \leftarrow S' \cup \{\boldsymbol{v}\}$
  else
    if $\exists\, \boldsymbol{c}_1 \in C_1, \|\boldsymbol{v} - \boldsymbol{c}_1\| \leq \gamma_1 R$ **then**
      if $\exists\, \boldsymbol{c}_2 \in C_2^{\boldsymbol{c}_1}, \|\boldsymbol{v} - \boldsymbol{c}_2\| \leq \gamma_2 R$ **then** $\backslash C_2^{\boldsymbol{c}_1}$ is initialized as $\emptyset\backslash$
        if $\exists\, \boldsymbol{c}_3 \in C_3^{\boldsymbol{c}_1,\boldsymbol{c}_2}, \|\boldsymbol{v} - \boldsymbol{c}_3\| \leq \gamma_3 R$ **then** $\backslash C_3^{\boldsymbol{c}_1,\boldsymbol{c}_2}$ is initialized as $\emptyset\backslash$
          $S' \leftarrow S' \cup \{\boldsymbol{v} - \boldsymbol{c}_3\}$
        else
          $C_3^{\boldsymbol{c}_1,\boldsymbol{c}_2} \leftarrow C_3^{\boldsymbol{c}_1,\boldsymbol{c}_2} \cup \{\boldsymbol{v}\}$ \ centers of small balls \
        end if
      else
        $C_2^{\boldsymbol{c}_1} \leftarrow C_2^{\boldsymbol{c}_1} \cup \{\boldsymbol{v}\}$ \ centers of medius balls \
      end if
    else
      $C_1 \leftarrow C_1 \cup \{\boldsymbol{v}\}$ \ centers of big balls \
    end if
  end if
end for
return S'

Outline

Complexity of the Algorithm

Denote by N_1, N_2 and N_3 the upper bound of the expected number of lattice points in C_1, $C_2^{\mathbf{c_1}}$(for any $\mathbf{c_1} \in C_1$) and $C_3^{\mathbf{c_1},\mathbf{c_2}}$(for any $\mathbf{c_1} \in C_1$), $\mathbf{c_2} \in C_2^{\mathbf{c_1}}$.

- Time complexity $O(N_1N_2N_3(N_1 + N_2 + N_3))$.
- Space complexity $O(N_1N_2N_3)$.

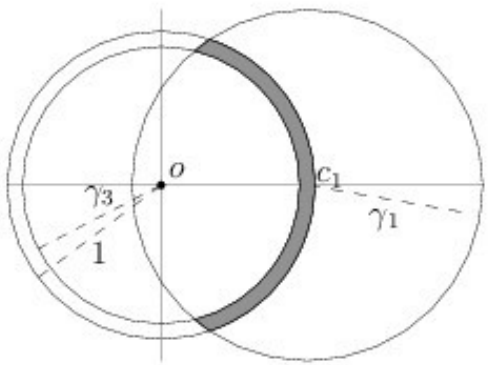
Estimation of N_1

Theorem 1 (Wang *et al.*)

Let n be a non-negative integer and $0.88 < \gamma_3 < 1 < \gamma_1 < \sqrt{2}\gamma_3$. *Then*

$$N_1 = c_{\mathcal{H}_1}^n \lceil 3\sqrt{2\pi} n^{\frac{3}{2}} \rceil,$$

where $c_{\mathcal{H}_1} = 1/(\gamma_1 \sqrt{1 - \frac{\gamma_1^2}{4}})$.



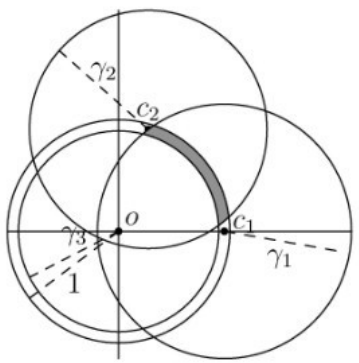
Estimation of N_2

Theorem 2

Let n be a non-negative integer, $0.88 < \gamma_3 < 1 < \gamma_2 < \gamma_1 < \sqrt{2}\gamma_3$, where γ_3 is very close to 1. Then

$$N_2 = c_2\left(\frac{c_{\mathcal{H}_2}}{d_{\min}}\right)^n \lceil n^{\frac{3}{2}} \rceil,$$

where $c_{\mathcal{H}_2} = \frac{\gamma_1}{\gamma_3}\sqrt{1-\gamma_1^2/(4\gamma_3^2)}$, $d_{\min} = \gamma_2\sqrt{1-\gamma_2^2 c_{\mathcal{H}_1}^2/4}$, $c_{\mathcal{H}_1} = 1/(\gamma_1\sqrt{1-\gamma_1^2/4})$ and c_2 is a positive constant unrelated to n.



Estimation of N_3

Theorem 3

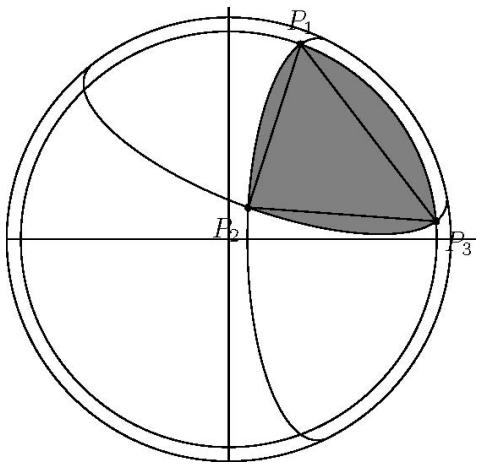
Let n be a non-negative integer, $0.88 < \gamma_3 < 1 < \gamma_2 < \gamma_1 < \sqrt{2}\gamma_3$, where γ_3 is very close to 1. Then

$$N_3 = c_3 n^3 \left(\frac{d_{\max}}{r_{\min}}\right)^n,$$

where $d_{\max} = \sqrt{1 - \left(\frac{\gamma_3^2-\gamma_1^2+1}{2\gamma_3}\right)^2 - \left(\frac{1}{c_{\mathcal{H}_2}}\left(\frac{\gamma_3^2+1-\gamma_2^2}{2} - \frac{2\gamma_3^2-\gamma_1^2}{2\gamma_3}\frac{\gamma_3^2-\gamma_1^2+1}{2\gamma_3}\right)\right)^2}$, $r_{\min} = \sqrt{c_{\mathcal{H}_3} - \left(1 - \frac{\gamma_3^2}{2c_{\mathcal{H}_3}}\right)^2}$, $c_{\mathcal{H}_1} = \frac{1}{\gamma_1\sqrt{1-\frac{\gamma_1^2}{4}}}$, $c_{\mathcal{H}_2} = \frac{\gamma_1}{\gamma_3}\sqrt{1-\frac{\gamma_1^2}{4\gamma_3^2}}$, $c_{\mathcal{H}_3} = \gamma_2^2\left(1 - \frac{\gamma_2^2 c_{\mathcal{H}_1}^2}{4}\right)$, *and c_3 is a positive constant unrelated to n.*

A Technique of Proving Theorem 3

- Compute the lower bound for the volume of an irregular spherical shell.
- Projecting the target region to a hyperplane and compute the integral .
- Estimate the lower bound of the projected region.
- For a two dimensional cutting plane of the projected region, use the area of a inscribed triangular instead of the real area, which makes the integral computation feasible.



Outline

Main Results

Theorem 4

The optimal time complexity of the algorithm is $2^{0.3778n+o(n)}$ *polynomial-time operations with* $\gamma_3 \to 1, \gamma_1 = 1.1399, \gamma_2 = 1.0677$, *and the corresponding space complexity is* $2^{0.2833n+o(n)}$ *polynomially many bits.*

algorithm	time complexity	space complexity
Nguyen-Vidick	$2^{0.415n+o(n)}$	$2^{0.2075n+o(n)}$
WLTB	$2^{0.3836n+o(n)}$	$2^{0.2557n+o(n)}$
Three-Level	$2^{0.3778n+o(n)}$	$2^{0.2833n+o(n)}$

Experimental Results

dimension		10	20	25	30	40	50	60
number of sample		150000	100000	8000	5000	5000	3000	2000
time of sample(sec.)		301	810	87833	73375	147445	120607	167916
Time (sec.)	NV alg.	25005	64351	120	220	625	254	187
	WLTB alg.	23760	18034	35	42	93	46	47
	Our alg.	20942	13947	27	27	57	29	30
	GaussSieve alg.	0.003	0.013	0.068	0.098	0.421	3.181	42.696
$\frac{\|\mathbf{v}\|}{\lambda_1}$	NV alg.	1	1	23.8	38.3	170.1	323	347.7
	WLTB alg.	1	1	25.9	35.1	170.1	323	347.7
	Our three-level alg.	1	1	21.2	38.3	170.1	323	347.7
	GaussSieve alg.	1	1	1	1	1	1	1

Remark

It is natural to extend the three-level sieve to multiple level, such as four-level. However,

- Theoretical analysis will be more complicated.
- The improvement may be not large.

We conjecture that it can not be improved always by increasing the number of levels.

Thanks!