Practical Improvements on BKZ Algorithm

Ziyu Zhao, Jintai Ding

Tsinghua University

November 30, 2022

Introduction

For lattice-based cryptosystems, the most effective attack is usually the lattice attack.
We want to know the concrete hardness of the following problem:

Given a target length ℓ and a basis of a (random) lattice L, find a lattice vector with length less than ℓ.

Introduction

SVP algorithms: Enumeration, Sieving...

BKZ algorithm: calls the SVP algorithms on d dimensional local projected lattices for several times, and outputs a rather short vector $\mathbf{v}$, achieves the same root Hermite factor as the SVP subroutines.

$$\left(\frac{||\mathbf{v}||}{\det(L)^{\frac{1}{n}}}\right)^{\frac{1}{n}} \approx \left(\sqrt{\frac{d}{2\pi e}}\right)^{\frac{1}{d}}$$

We give some techniques on BKZ, which will provide about 10 times speedup in real attacks.

Introduction

With these techniques, we solved some lattice challenges in https://www.latticechallenge.org/ideallattice-challenge
The details are listed below:

dim	length	Hermite factor	total cost	based on
656	670275	1.00993^{700}	380 CPUhours	Enum
700	659874	1.00928^{700}	1787 CPUhours	Sieving

Notations

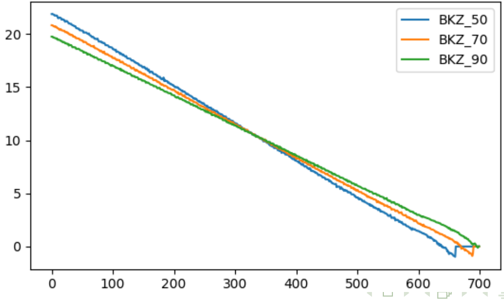
If $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n)$ is a basis of L, $(\mathbf{b}_1^*, \mathbf{b}_2^*, \cdots, \mathbf{b}_n^*)$ is the Gram-Schmidt orthogonalization, $B_i = ||\mathbf{b}_i^*||^2$ and π_i is the orthogonal projection to $\text{span}(\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{i-1})^\perp$, then we define the local projected lattice $L_{[i,j]}$ to be the lattice spanned by $B_{[i,j]} = (\pi_i(\mathbf{b}_i), \cdots, \pi_i(\mathbf{b}_j))$. And we call

$$[||\mathbf{b}_1^*||^2, ||\mathbf{b}_2^*||^2, \cdots, ||\mathbf{b}_n^*||^2]$$

the distance vector of the basis.

The Distance Vector

- Schnorr's Geometric Series Assumption (GSA, see [Sch03]).
- The x-axis shows the indexes i, and the y-axis shows the logarithm of $||\mathbf{b}_i^*||$.



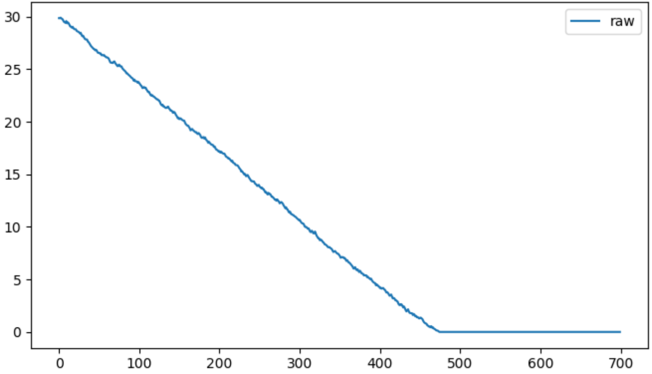
Local Basis Processing

It's always a good choice to use local basis processing instead of inserting a single short vector.

- compute the transform matrix of local processing (on the local projected lattice)
- apply it on the vectors of the original basis then size reduce the basis
- mentioned in literature (e.g. [ADH+19]) for sieving based BKZ

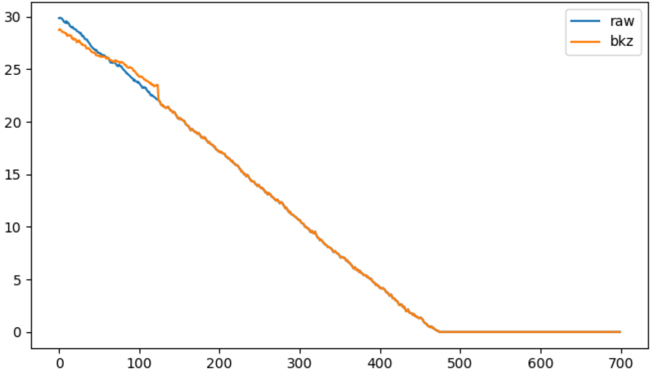
Local Basis Processing

What does BKZ based on local basis processing look like?



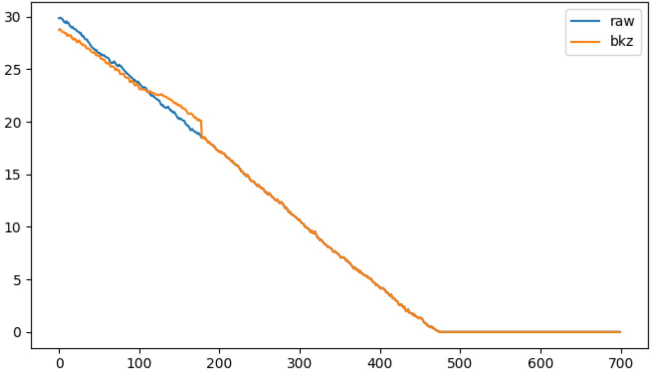
Local Basis Processing

What does BKZ based on local basis processing look like?



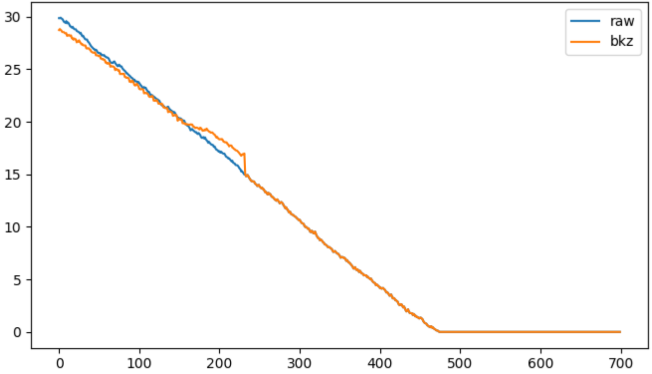
Local Basis Processing

What does BKZ based on local basis processing look like?



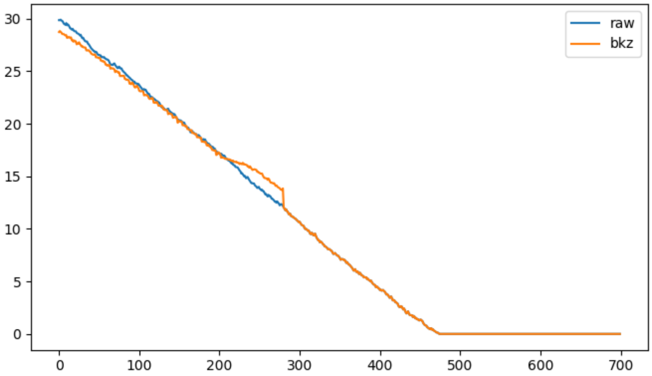
Local Basis Processing

What does BKZ based on local basis processing look like?



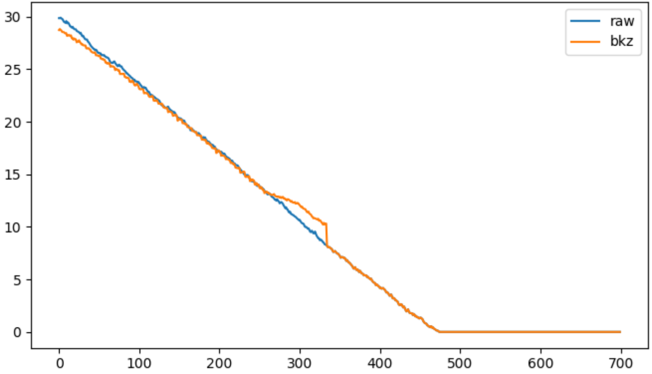
Local Basis Processing

What does BKZ based on local basis processing look like?



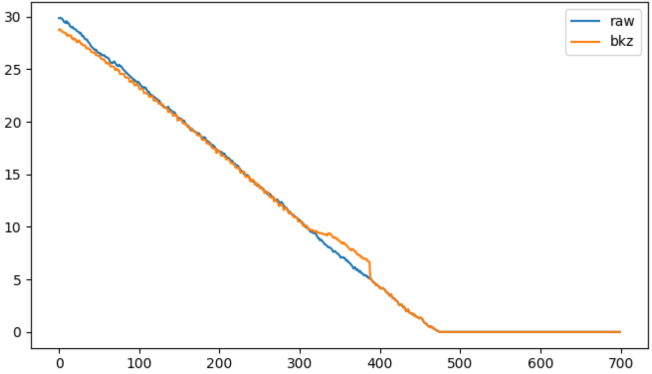
Local Basis Processing

What does BKZ based on local basis processing look like?



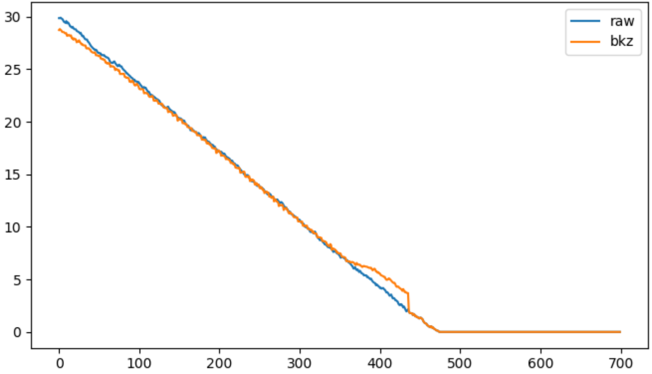
Local Basis Processing

What does BKZ based on local basis processing look like?



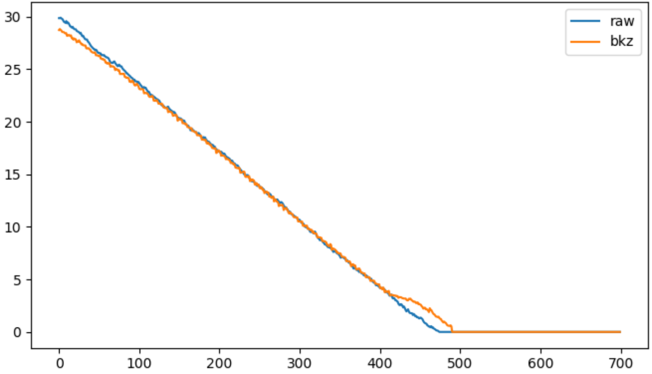
Local Basis Processing

What does BKZ based on local basis processing look like?



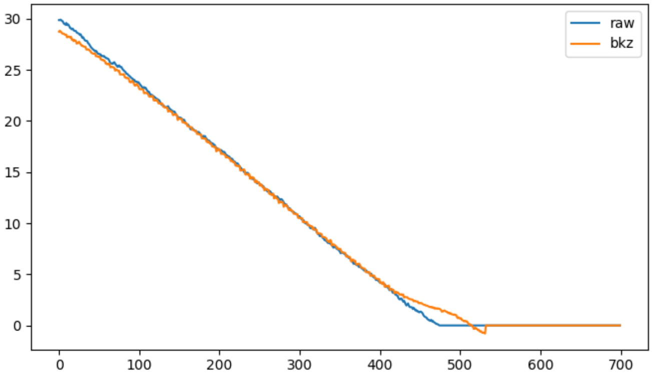
Local Basis Processing

What does BKZ based on local basis processing look like?



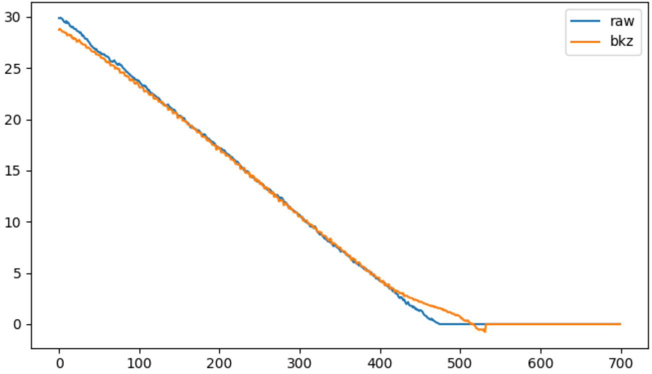
Local Basis Processing

What does BKZ based on local basis processing look like?



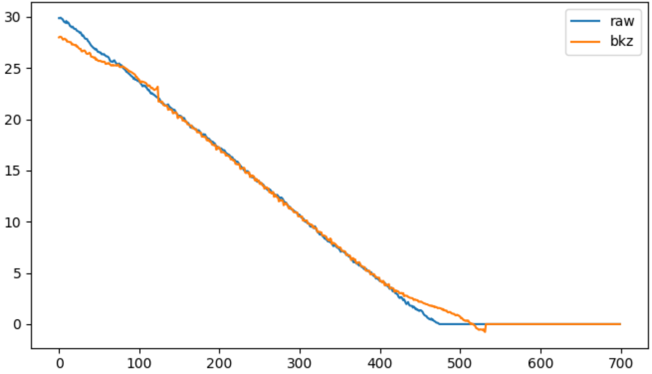
Local Basis Processing

What does BKZ based on local basis processing look like?



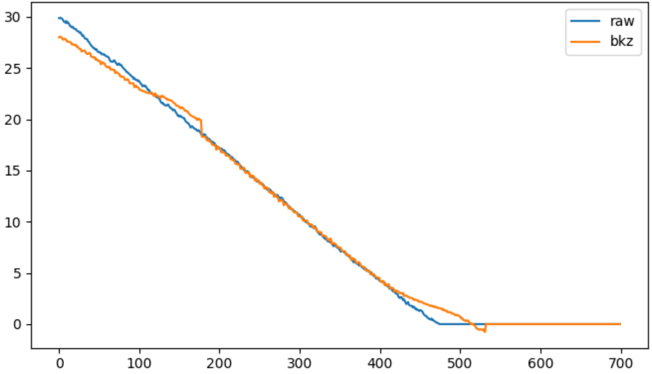
Local Basis Processing

What does BKZ based on local basis processing look like?



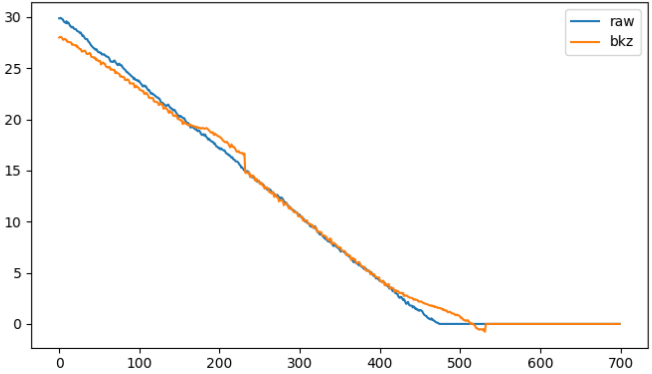
Local Basis Processing

What does BKZ based on local basis processing look like?



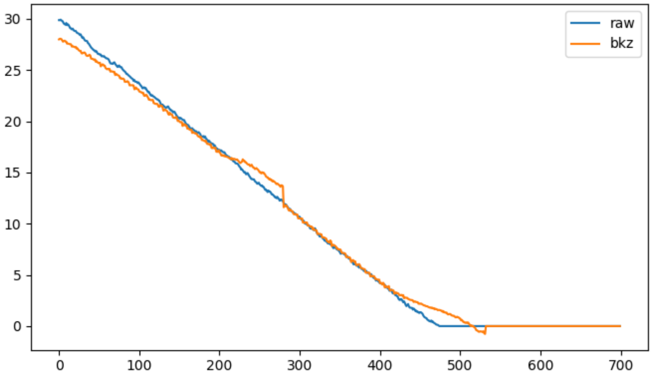
Local Basis Processing

What does BKZ based on local basis processing look like?



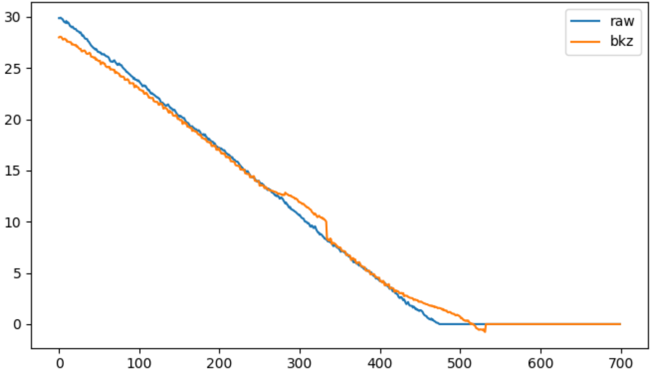
Local Basis Processing

What does BKZ based on local basis processing look like?



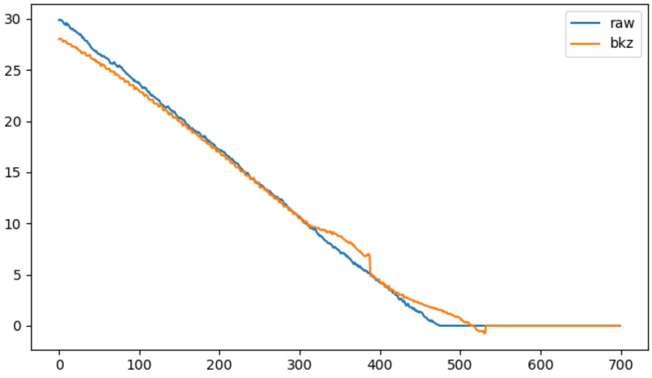
Local Basis Processing

What does BKZ based on local basis processing look like?



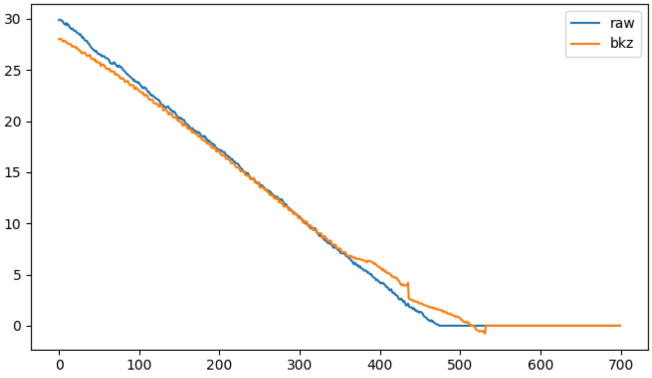
Local Basis Processing

What does BKZ based on local basis processing look like?



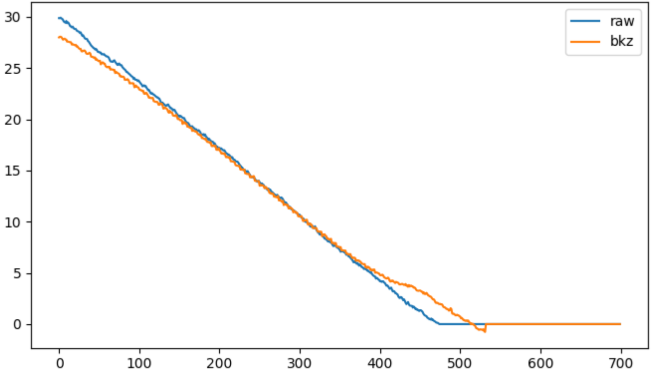
Local Basis Processing

What does BKZ based on local basis processing look like?



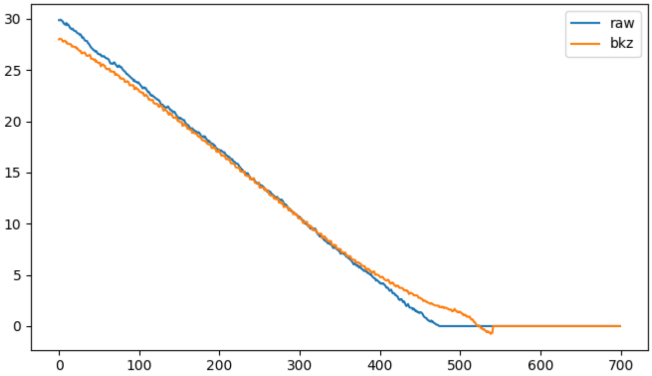
Local Basis Processing

What does BKZ based on local basis processing look like?



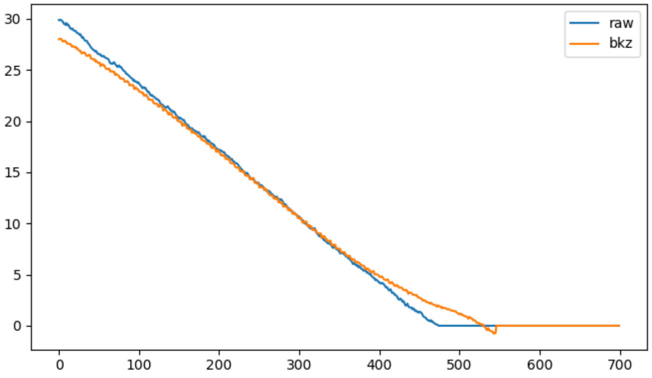
Local Basis Processing

What does BKZ based on local basis processing look like?



Local Basis Processing

What does BKZ based on local basis processing look like?



Local Basis Processing

We can simulate the distance vector as following if we know the average distance vector (denote by $[D_1, D_2, \cdots, D_d]$) of a lattice with det $= 1$ after running the SVP subroutine:

Algorithm 4: Simulation of BKZ algorithm

Input: a distance vector $[B_1, B_2, \cdots, B_n]$, blocksize d and the average distance vector $[D_1, \cdots, D_d]$

Output: the new distance vector after run one tour of BKZ-d

```
for s = 1, ..., n − d + 1 do
    det = (∏_{i=s}^{s+d−1} B_i)^{1/d};
    for i = 0, ..., d − 1 do
        B_{i+s} = det · D_{i+1};
```

Jumping strategy

What will happen if we work on $L_{[i+s,j+s]}$ after $L_{[i,j]}$?

- The number of the SVP subroutines in each BKZ tour is only $\frac{1}{s}$ as before.
- How to evaluate the quality?

$$\mathrm{Pot}(L) = \prod_{i=1}^{n} B_i^{n+1-i}$$

Jumping strategy

- If GSA is true, Pot is an increase function of $||\mathbf{b}_1||$.
- We want to make Pot decrease as fast as possible.
- Run binary search on d and s (by simulation) to find the optimal choice.
- We may get a speed up of $2^{1.65}$ if we use an HKZ-reduction with time complexity $2^{0.386d}$ as the SVP subroutine.

Jumping strategy

MSD	68	69	70	71	72	73	74	75
cpu hours	2.30	2.74	3.27	3.88	4.69	5.69	6.93	8.52
$\Delta \log_2 \text{Pot}$	463	758	1166	1400	1910	2254	2674	2949
$\frac{\Delta \log_2 \text{Pot}}{\text{cost}}$	201	277	357	361	407	396	385	346

(MSD, jumping step)	(72, 1)	(73, 2)	(74, 3)	(75, 4)	(76, 5)	(77, 6)	(78, 7)
cpu hours	4.69	2.84	2.31	2.13	2.20	2.30	2.51
$\Delta \log_2 \text{Pot}$	1910	1797	1787	1962	2059	2084	2241
$\frac{\Delta \log_2 \text{Pot}}{\text{cost}}$	407	633	773	920	930	906	858

Reduce Only When We Need

- In practice, we don't need the whole reduced basis.
- For the last $[\frac{n}{d}]$ tours of the algorithm, we don't need to visit all the indexes.

Algorithm 2: The last several tours of our BKZ

Input: an n-dimensional lattice L, blocksize d and an SVP algorithm
Output: a reduced basis

```
m = [n/d];
for k = 1, 2, ..., m do
    // a BKZ tour on L_[1, n-kd+1]
    for i = 1, 2, ..., n - kd + 1 do
        reduce L_[i, i+d-1] by the SVP algorithm;
return L
```

Reduce Only When We Need

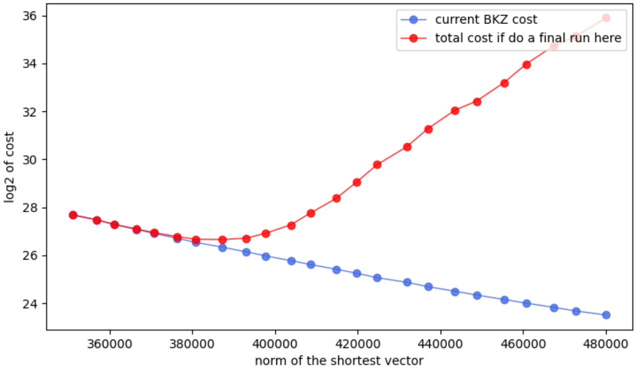
Details for the end of our 700 dimensional challenge are listed below:

MSD	working on	CPU time	a full tour time	min length
91	$L_{[1,578]}$	196.3h	206.4h	812896
92	$L_{[1,466]}$	201.5h	260.2h	805991
92	$L_{[1,354]}$	152.3h	258.9h	787811
93	$L_{[1,242]}$	146.0h	365.2h	755466
94	$L_{[1,130]}$	147.0h	651.9h	729162

A Large Final Run

- We can choose a much larger dimension d' in the last SVP subroutine (working on $[1, d']$) to get a much shorter vector.
- save the time for several tours of BKZ with a normal blocksize, about 1 bit.
- We can use the simulator to choose the optimal strategy.

A Large Final Run



Remark

In real lattice attacks, we actually need to find a hidden key in the lattice. The key will be much shorter than the Gaussian Heuristic of L.
To reduce it to our case, simply take the dual lattice.

$$\text{basis of } L \xrightarrow{\text{dual}} \text{basis of } L^{\text{dual}} \xrightarrow{\text{reduction}} \text{reduced basis of } L^{\text{dual}}$$
$$\xrightarrow{\text{dual}} \text{more orthogonal basis of } L \xrightarrow{\text{size reduce}} \text{key}$$

References

[ADH+19] Martin R Albrecht, Leo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W Postlethwaite, and Marc Stevens, The general sieve kernel and new records in lattice reduction, Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2019, pp. 717-746.

[Sch03] C.P. Schnorr. Lattice reduction by random sampling and birthday methods. In H. Alt and M. Habib, editors, STACS, pages 145-156, 2003.