RIKKYO UNIVERSITY

研究集会
「量子暗号理論と耐量子暗号」
早稲田大学
（Zoomによるオンライン開催）

最短ベクトル問題を解くための格子基底簡約とその大規模並列化

2022年3月18日（金）
11:00～12:00

安田雅哉（立教大学）

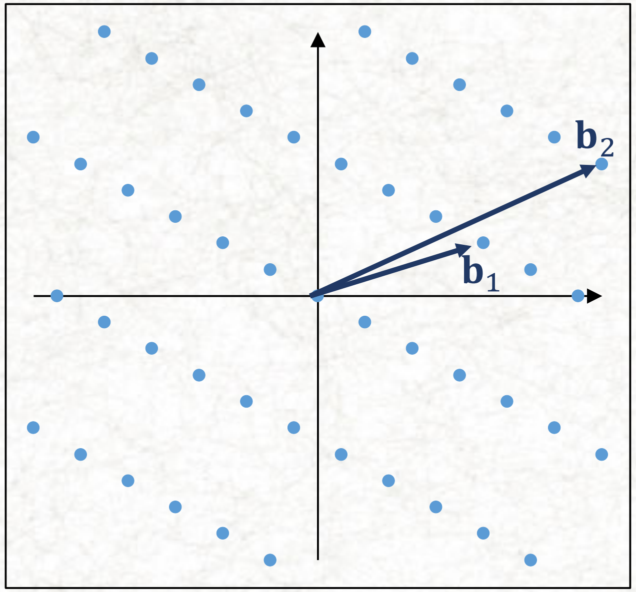
Basics on Lattices

- For linearly independent $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^n$,

Integral combination

$$L = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) := \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$$

  is a (full-rank) **lattice** of dimension n
  - $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$: a **basis** of L
    - Regard it as the $n \times n$ matrix
  - Infinitely many bases if $n \geq 2$
    - If $\mathbf{B}_1$ and $\mathbf{B}_2$ span the same lattice,
    - then $\exists \mathbf{V} \in \mathrm{GL}_n(\mathbb{Z})$ such that $\mathbf{B}_1 = \mathbf{B}_2 \mathbf{V}$
  - $\mathrm{vol}(L) = |\det(\mathbf{B})|$: the **volume** of L
    - Independent of the choice of bases



A lattice of dimension $n = 2$

Lattices in Cryptography

RIKKYO UNIVERSITY

- **Post-Quantum Cryptography (PQC) Standardization**
  - Since 2015, National Institute of Standards and Technology (NIST) has proceeded a standardization project for PQC
    - To standardize quantum-resistant public-key cryptographic algorithms
    - Post-Quantum Cryptography | CSRC (nist.gov)
  - In July 2020, NIST selected **7 Finalists** and **8 Alternates**
    - **7 lattice-based schemes** are now in evaluation at the 3rd round
      - **5 Finalists (Kyber, NTRU, SABER, Dilithium, Falcon)**
      - **2 Alternates (FrodoKEM, NTRUprime)**

	Signatures		KEM/Encryption		Overall	
	Rd 1	Rd 2	Rd 1	Rd 2	Rd 1	Rd 2
Lattice-based	5	3	21	9	26	12
Code-based	2		17	7	19	7
Multi-variate	7	4	2		9	4
Hash/Symmetric	3	2			3	2
Other	2		5	1	7	1
Total	**19**	**10**	**45**	**16**	**64**	**26**

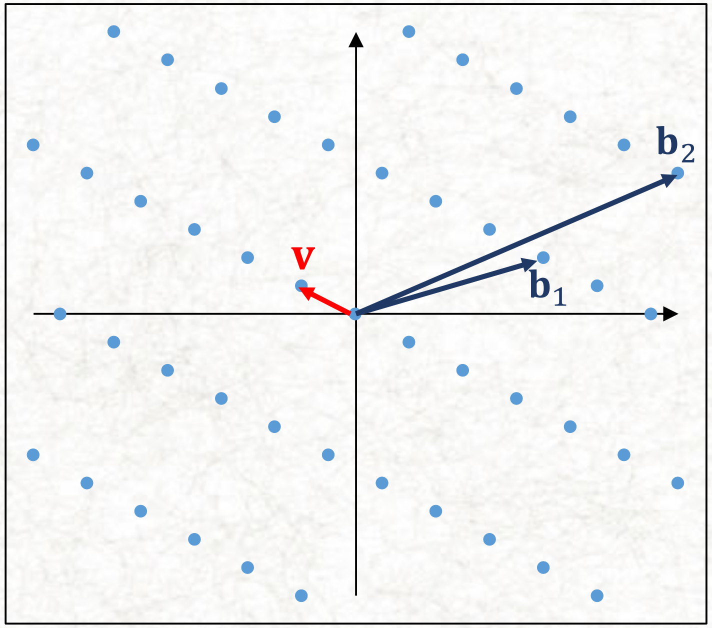
	Finalists	Alternates
KEMs/Encryption	Kyber NTRU SABER Classic McEliece	Bike FrodoKEM HQC NTRUprime SIKE
Signatures	Dilithium Falcon Rainbow	GeMSS Picnic SPHINCS+

NIST Status Update on the 3rd Round (PDF)

Lattice Problems

RIKKYO UNIVERSITY

- **Algorithmic problems for lattices**
  - **SVP** (Shortest Vector Problem) ***Our focus***
    - Given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L
    - Find a non-zero shortest vector in L
  - **CVP** (Closest Vector Problem)
  - **LWE** (Learning with Errors)
  - **NTRU**, etc.
- **Relationship with cryptography**
  - The security of lattice-based cryptography is based on the hardness of lattice problems
  - In particular, the hardness of SVP and CVP supports the security of most schemes



SVP in a two-dimensional lattice

- Given linearly independent $\mathbf{b}_1, \mathbf{b}_2$
- Find a non-zero shortest vector $\mathbf{v} = a_1\mathbf{b}_1 + a_2\mathbf{b}_2$ for some $a_1, a_2 \in \mathbb{Z}$

Known Results for $\lambda_1(L)$

RIKKYO UNIVERSITY

- **The first successive minimum**
  - Define $\lambda_1(L)$ as the length of a non-zero shortest vector in a lattice L
  - SVP is the problem that finds $\mathbf{s} \in L$ such that $\|\mathbf{s}\| = \lambda_1(L)$
- **Theoretical results**
  - Minkowski's convex body theorem implies

$$\lambda_1(L) \leq 2\omega_n^{-\frac{1}{n}}\mathrm{vol}(L)^{\frac{1}{n}}$$

    for any lattice L of dimension n (ω_n: the volume of the unit ball in $\mathbb{R}^n$)
- **Heuristic results**
  - The Gaussian Heuristic implies

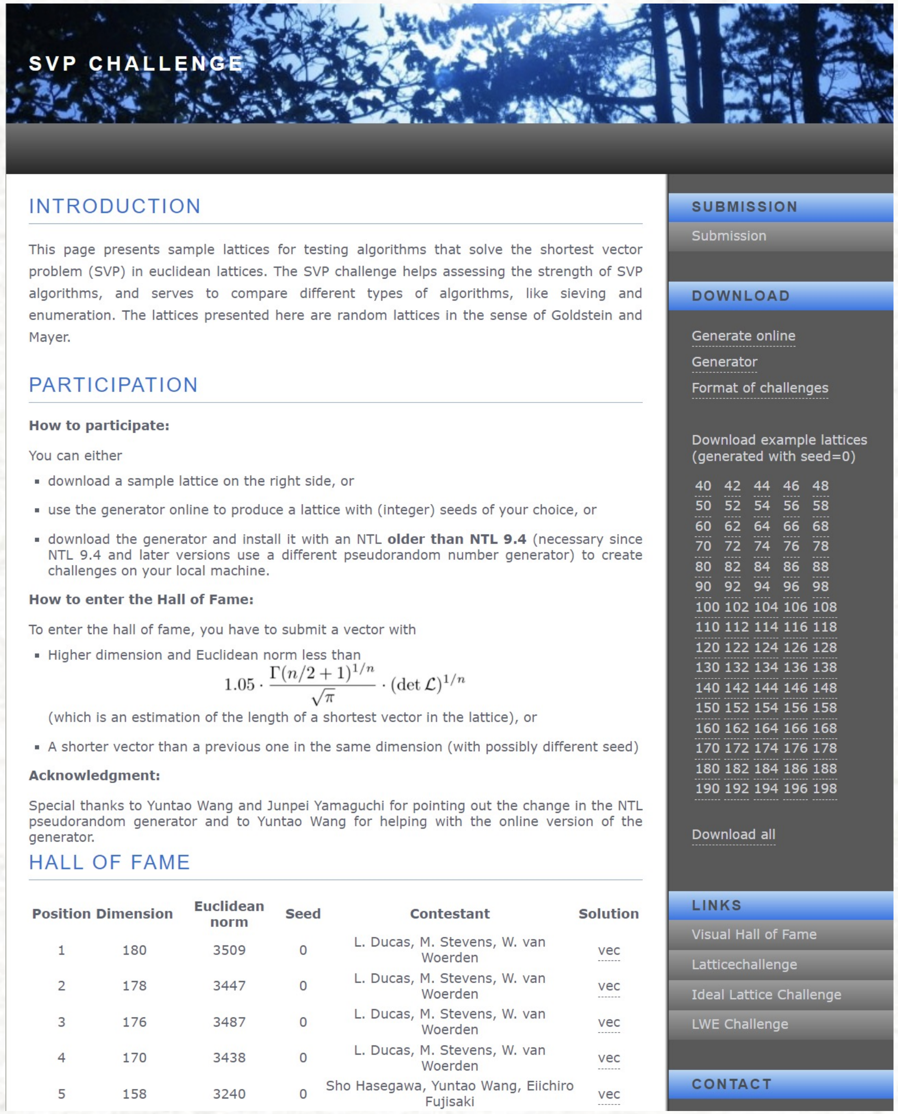
$$\lambda_1(L) \approx \omega_n^{-\frac{1}{n}}\mathrm{vol}(L)^{\frac{1}{n}} \sim \sqrt{{}^{n}/_{2\pi e}}\,\mathrm{vol}(L)^{\frac{1}{n}} =: \mathrm{GH}(L)$$

    - The Gaussian Heuristic: The number of vectors in $L \cap S$ is roughly equal to $\mathrm{vol}(S)/\mathrm{vol}(L)$ for a measurable set S in $\mathbb{R}^n$
    - It holds in practice for "random" lattices in high dimensions $n \geq 50$

SVP Challenge

RIKKYO UNIVERSITY

- **The Darmstadt SVP challenge**
  - Sample bases $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ are presented for dimensions $40 \leq n \leq 198$
  - Any vector $\mathbf{v} \in L = \mathcal{L}(\mathbf{B})$ of length
    $$||\mathbf{v}|| < 1.05\mathrm{GH}(L) \approx 1.05\lambda_1(L)$$
    can be submitted to the hall of fame
    - That is, **approximate SVP with factor 1.05**
  - The current highest dimension to be solved is $n = 180$ (141.pdf (iacr.org))
    - It took about **51.6 days** on a server with 4 NVIDIA Turing GPUs with 1.5 TB RAM
    - But the record must be not the shortest (since its approximation factor is about 1.04)



SVP CHALLENGE

INTRODUCTION

This page presents sample lattices for testing algorithms that solve the shortest vector problem (SVP) in euclidean lattices. The SVP challenge helps assessing the strength of SVP algorithms, and serves to compare different types of algorithms, like sieving and enumeration. The lattices presented here are random lattices in the sense of Goldstein and Mayer.

PARTICIPATION

How to participate:

You can either

- download a sample lattice on the right side, or
- use the generator online to produce a lattice with (integer) seeds of your choice, or
- download the generator and install it with an NTL **older than NTL 9.4** (necessary since NTL 9.4 and later versions use a different pseudorandom number generator) to create challenges on your local machine.

How to enter the Hall of Fame:

To enter the hall of fame, you have to submit a vector with

- Higher dimension and Euclidean norm less than
  $$1.05 \cdot \frac{\Gamma(n/2+1)^{1/n}}{\sqrt{\pi}} \cdot (\det \mathcal{L})^{1/n}$$
  (which is an estimation of the length of a shortest vector in the lattice), or
- A shorter vector than a previous one in the same dimension (with possibly different seed)

Acknowledgment:

Special thanks to Yuntao Wang and Junpei Yamaguchi for pointing out the change in the NTL pseudorandom generator and to Yuntao Wang for helping with the online version of the generator.

HALL OF FAME

Position	Dimension	Euclidean norm	Seed	Contestant	Solution
1	180	3509	0	L. Ducas, M. Stevens, W. van Woerden	vec
2	178	3447	0	L. Ducas, M. Stevens, W. van Woerden	vec
3	176	3487	0	L. Ducas, M. Stevens, W. van Woerden	vec
4	170	3438	0	L. Ducas, M. Stevens, W. van Woerden	vec
5	158	3240	0	Sho Hasegawa, Yuntao Wang, Eiichiro Fujisaki	vec

SUBMISSION

Submission

DOWNLOAD

Generate online

Generator

Format of challenges

Download example lattices (generated with seed=0)

40 42 44 46 48
50 52 54 56 58
60 62 64 66 68
70 72 74 76 78
80 82 84 86 88
90 92 94 96 98
100 102 104 106 108
110 112 114 116 118
120 122 124 126 128
130 132 134 136 138
140 142 144 146 148
150 152 154 156 158
160 162 164 166 168
170 172 174 176 178
180 182 184 186 188
190 192 194 196 198

Download all

LINKS

Visual Hall of Fame

Latticechallenge

Ideal Lattice Challenge

LWE Challenge

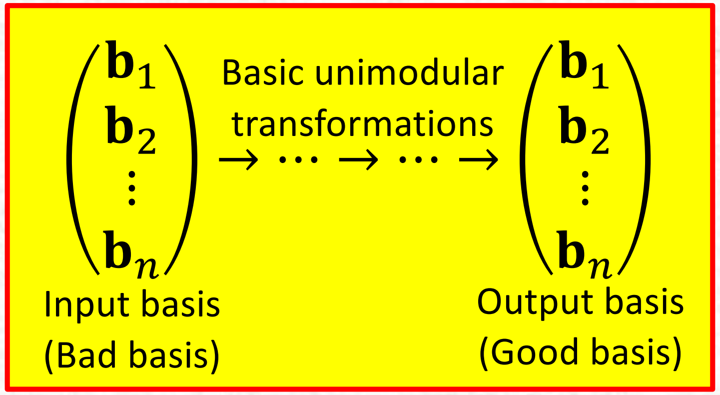
CONTACT

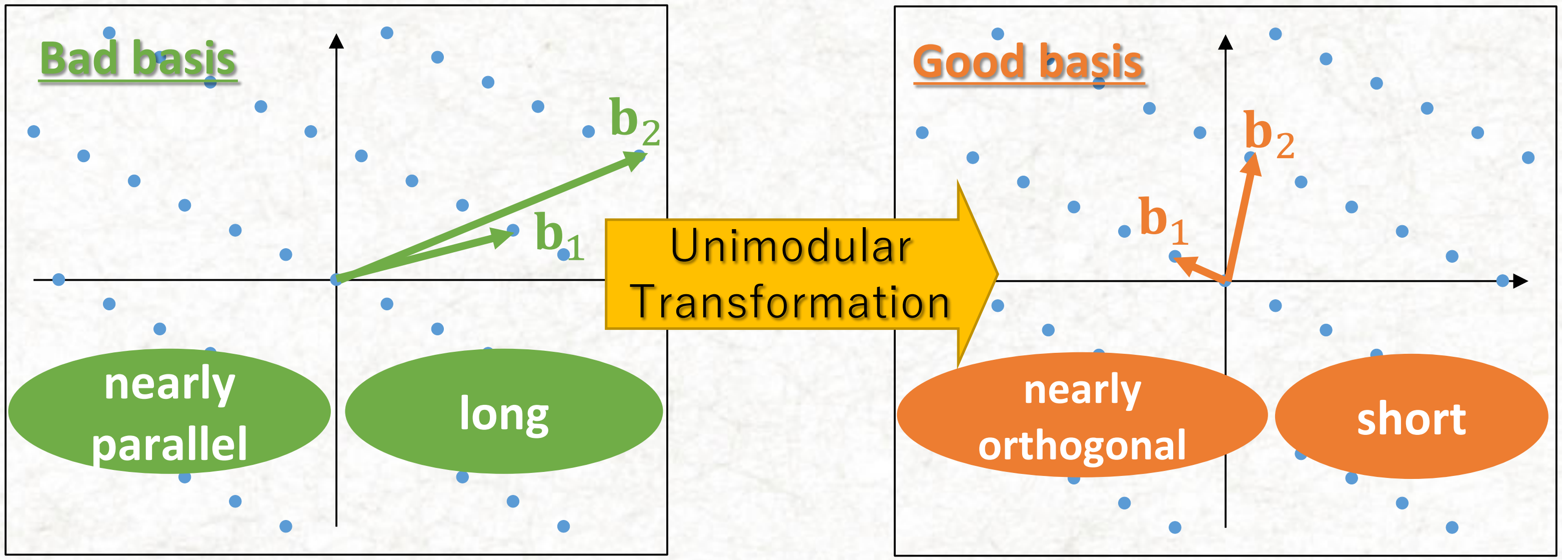
SVP Challenge (latticechallenge.org)

Lattice Basis Reduction

RIKKYO UNIVERSITY

- **Strong tool for solving lattice problems including SVP**
  - Find a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ with short and nearly orthogonal vectors
    - Such a basis is called "**good**" or "**reduced**"
    - Some basis vectors $\mathbf{b}_i$'s are very short
  - Consist of basic unimodular transformations
    - ① Multiply by (-1): $\mathbf{b}_i \leftarrow -\mathbf{b}_i$
    - ② Swap $\mathbf{b}_i$ and $\mathbf{b}_j$
    - ③ Multiply (by integer)-Add: $\mathbf{b}_i \leftarrow \mathbf{b}_i + a\mathbf{b}_j$ $(a \in \mathbb{Z})$





LLL (1/3): Definition and Properties

RIKKYO UNIVERSITY

- **Lenstra-Lenstra-Lovász (LLL)-reduction** [LLL82]
  - $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ is **δ-LLL-reduced** if it satisfies two conditions
    - ① **Size-reduced**: $|\mu_{ij}| \leq \frac{1}{2}$ for all $1 \leq j < i \leq n$
    - ② **Lovász' condition**: $\|\mathbf{b}_k^*\|^2 \geq (\delta - \mu_{k,k-1}^2)\|\mathbf{b}_{k-1}^*\|^2$
      - $\frac{1}{4} < \delta < 1$: reduction parameter (e.g., $\delta = 0.99$ for practice)
      - $\mathbf{B}^* = (\mathbf{b}_1^*, \dots, \mathbf{b}_n^*)$, $\mu = (\mu_{ij})$: Gram-Schmidt information of $\mathbf{B}$:

$$\mathbf{b}_1^* = \mathbf{b}_1,\ \mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b}_j^*,\ \mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$$

  - Every LLL-reduced basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ of a lattice L satisfies
    - $\|\mathbf{b}_1\| \leq \alpha^{\frac{n-1}{2}} \lambda_1(L)$, where $\alpha = \frac{4}{4\delta - 1} > \frac{4}{3}$
    - $\|\mathbf{b}_1\| \leq \alpha^{\frac{n-1}{4}} \mathrm{vol}(L)^{\frac{1}{n}}$

[LLL82] A.K. Lenstra, H.W. Lenstra and L. Lovász, "Factoring polynomials with rational coefficients", Mathematische Annalen 261 (4): 515—534 (1982).

LLL (2/3): Basic Algorithm

RIKKYO UNIVERSITY

- **It consists of two procedures to find an LLL-reduced basis**
  - ① **Size-reduction**: $\mathbf{b}_k \leftarrow \mathbf{b}_k - q\mathbf{b}_j$ with $q = \lfloor \mu_{k,j} \rceil$
  - ② **Swap adjacent vectors**: $\mathbf{b}_{k-1} \leftrightarrow \mathbf{b}_k$ if they do not satisfy Lovász' condition

Algorithm: The basic LLL Lenstra et al. (1982)

Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, and a reduction parameter $\frac{1}{4} < \delta < 1$
Output: A δ-LLL-reduced basis $\mathbf{B}$ of L

```
Compute Gram–Schmidt information μ_{i,j} and ||b_i*||^2 of the input basis B
k ← 2
while k ≤ n do
    ① Size-reduce B = (b_1, ..., b_n) // At each k, we recursively change b_k ← b_k − ⌊μ_{k,j}⌉b_j for 1 ≤ j ≤ k − 1 (e.g., see Galbraith 2012, Algorithm 24)
    if (b_{k−1}, b_k) satisfies Lovász' condition then
        k ← k + 1
    else
        ② Swap b_k with b_{k−1}, and update Gram–Schmidt information of B
        k ← max(k − 1, 2)
    end if
end while
```

LLL (3/3): Sage Code

RIKKYO UNIVERSITY

```
def GSO(B, n):
    GS = Matrix(QQ, n)
    mu = Matrix(QQ, n)
    for i in range(n):
        GS[i] = B[i]
        mu[i, i] = 1
        for j in range(i):
            mu[i,j] = B[i].inner_product(GS[j])/GS[j].norm()^2
            GS[i] -= mu[i,j]*GS[j]
    return GS, mu

def LLL(B, n, delta):
    GS, mu = GSO(B, n)
    BB = vector(QQ, n)
    for i in range(n):
        BB[i] = GS[i].norm()^2
    k=1
    while k<=n-1:
        for j in range(k)[::-1]:
            if abs(mu[k,j])> 0.50:
                q=round(mu[k, j])
                B[k] -= q*B[j]
                for l in range(j+1):
                    mu[k,l] -= q*mu[j,l]
        if BB[k] >= (delta - mu[k, k-1]^2)*BB[k-1]:
            k+=1
        else:
            v = B[k-1]; B[k-1]=B[k]; B[k]=v;
            GS, mu=GSO(B, n)
            for i in range(n):
                BB[i] = GS[i].norm()^2
            k=max(k-1, 1)
    return true
```

```

n = 10; d = 100000
B = Matrix(ZZ, n)
for i in range(0, n):
    B[i, i] = 1
    B[i, 0] = randint(-d, d)
print("Input basis")
show(B)
LLL(B, n, 0.99)
print("¥nOutput basis")
show(B)
```

Please use [Sage Cell Server (sagemath.org)](https://sagecell.sagemath.org)

Enumeration (1/3): Basic Idea

RIKKYO UNIVERSITY

- **Enumerate all vectors $\mathbf{s} = \sum v_i \mathbf{b}_i \in \mathcal{L}(\mathrm{B})$ such that $\|\mathbf{s}\| \leq R$**
  - $R > 0$: search radius (e.g., $R = 1.05\mathrm{GH}(L)$)
  - With Gram-Schmidt information, write

$$\mathbf{s} = \sum_{j=1}^{n} \left(v_j + \sum_{i=j+1}^{n} \mu_{ij} v_i \right) \mathbf{b}_j^*$$

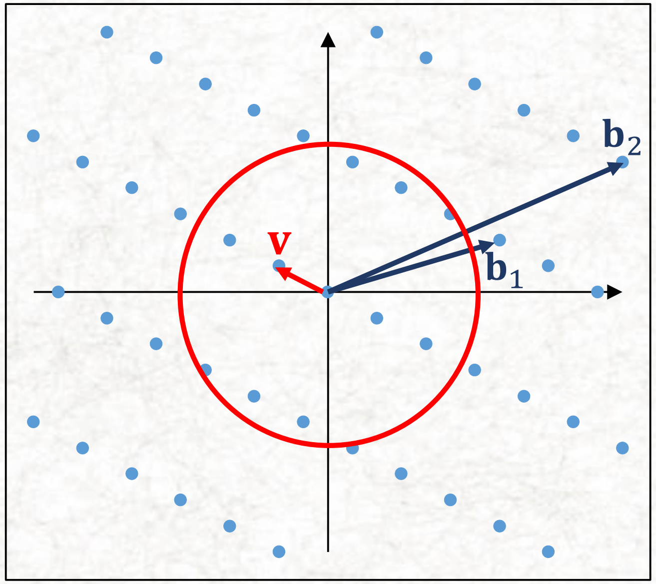
  - By the orthogonality of Gram-Schmidt vectors,

$$\|\pi_k(\mathbf{s})\|^2 = \sum_{j=k}^{n} \left(v_j + \sum_{i=j+1}^{n} \mu_{ij} v_i \right)^2 \|\mathbf{b}_j^*\|^2$$

  for $1 \leq k \leq n$, where π_k denotes the projection map to $\langle \mathbf{b}_k^*, \dots, \mathbf{b}_n^* \rangle_{\mathbb{R}}$

  - Consider n inequalities $\|\pi_k(\mathbf{s})\|^2 \leq R^2$ for $1 \leq k \leq n$:

$$\begin{cases} v_n^2 \leq R^2 / \|\mathbf{b}_n^*\|^2 \\ (v_{n-1} + \mu_{n,n-1} v_n)^2 \leq \left(R^2 - v_n^2 \|\mathbf{b}_n^*\|^2 \right) / \|\mathbf{b}_{n-1}^*\|^2 \\ \vdots \end{cases}$$



Enumeration (2/3): Basic Algorithm

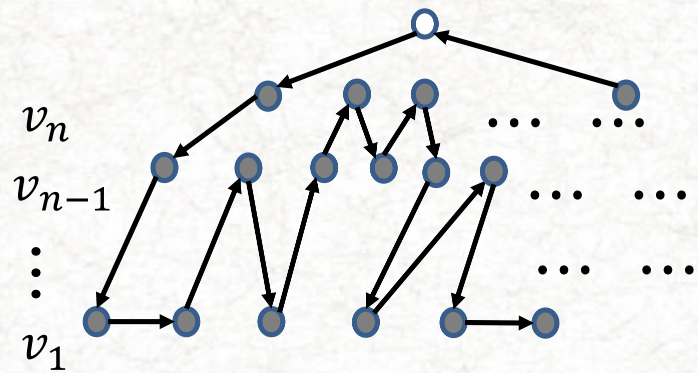
RIKKYO UNIVERSITY

Algorithm: The basic Schnorr–Euchner enumeration Schnorr and Euchner (1994)

Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L and a radius R with $\lambda_1(L) \leq R$
Output: The shortest non-zero vector $\mathbf{s} = \sum_{i=1}^{n} v_i \mathbf{b}_i$ in L

Compute Gram–Schmidt information $\mu_{i,j}$ and $\|\mathbf{b}_i^*\|^2$ of $\mathbf{B}$
$(\rho_1, \ldots, \rho_{n+1}) = \mathbf{0}$, $(v_1, \ldots, v_n) = (1, 0, \ldots, 0)$, $(c_1, \ldots, c_n) = \mathbf{0}$, $(w_1, \ldots, w_n) = \mathbf{0}$
$k = 1$, $\texttt{last\_nonzero} = 1$ // largest i for which $v_i \neq 0$
while $\texttt{true}$ **do**
  $\rho_k \leftarrow \rho_{k+1} + (v_k - c_k)^2 \cdot \|\mathbf{b}_k^*\|^2$ // $\rho_k = \|\pi_k(\mathbf{s})\|^2$
  if $\rho_k \leq R^2$ **then**
    if $k = 1$ **then** $R^2 \leftarrow \rho_k$, $\mathbf{s} \leftarrow \sum_{i=1}^{n} v_i \mathbf{b}_i$; // update the squared radius
    else $k \leftarrow k-1$, $c_k \leftarrow -\sum_{i=k+1}^{n} \mu_{i,k} v_i$, $v_k \leftarrow \lfloor c_k \rceil$, $w_k \leftarrow 1$;
  else
    $k \leftarrow k+1$ // going up the tree
    if $k = n+1$ **then return** $\mathbf{s}$;
    if $k \geq \texttt{last\_nonzero}$ **then** $\texttt{last\_nonzero} \leftarrow k$, $v_k \leftarrow v_k + 1$;
    else
      if $v_k > c_k$ **then** $v_k \leftarrow v_k - w_k$; **else** $v_k \leftarrow v_k + w_k$; // zig-zag search
      $w_k \leftarrow w_k + 1$
    end if
  end if
end while

- Enumerate lattice vectors
  $$\mathbf{s} = \sum v_i \mathbf{b}_i \in L$$
  such that $\|\mathbf{s}\| \leq R$
- Built an enumeration tree to find integral combinations $(v_1, \ldots, v_n)$



[A Survey of Solving SVP Algorithms and Recent Strategies for Solving the SVP Challenge | SpringerLink](#)

Enumeration (3/3): Sage Code

RIKKYO UNIVERSITY

```
def GSO(B, n):
    GS = Matrix(QQ, n)
    mu = Matrix(QQ, n)
    for i in range(n):
        GS[i] = B[i]
        mu[i, i] = 1
        for j in range(i):
            mu[i,j] = B[i].inner_product(GS[j])/GS[j].norm()^2
            GS[i] -= mu[i,j]*GS[j]
    return GS, mu

def ENUM(B, n, R):
    GS, mu = GSO(B, n)
    BB = vector(QQ, n)
    for i in range(n):
        BB[i] = GS[i].norm()^2
    sigma = Matrix(QQ, n+1, n)
    r = vector(ZZ, n+1)
    rho = vector(QQ, n+1)
    v = vector(ZZ, n)
    c = vector(QQ, n)
    w = vector(ZZ, n)
    for i in range(n+1):
        r[i] = i
    v[0] = 1
    last_nonzero = 1
    k = 1
    while (1):
        rho[k-1] = rho[k] + (v[k-1] - c[k-1])^2*BB[k-1]
        if RR(rho[k-1]) <= RR(R):
            if k==1:
                print("Solution found"); return v
            k = k-1
            r[k-1] = max(r[k-1], r[k])
            for i in range(k+1, r[k]+1)[::-1]:
                sigma[i-1, k-1] = sigma[i, k-1] + mu[i-1, k-1]*v[i-1]
            c[k-1] = -sigma[k, k-1]
            v[k-1] = round(c[k-1])
            w[k-1] = 1
        else:
            k = k+1
            if k==n+1:
                print("No solution"); return false
            r[k-1] = k
            if k>=last_nonzero:
                last_nonzero = k
                v[k-1] = v[k-1] + 1
            else:
                if RR(v[k-1]) > RR(c[k-1]):
                    v[k-1] = v[k-1] - w[k-1]
                else:
                    v[k-1] = v[k-1] + w[k-1]
                w[k-1] = w[k-1] + 1

#Main
n = 20
B = random_matrix(ZZ, n, x=0, y = 30)
B.LLL()
print("LLL-reduced basis =¥n", B)
R = 0.99*RR(B[0].norm()^2)
while (1):
    v = vector(ZZ, n)
    v = ENUM(B, n, R)
    if v != false:
        vec = v[0]*B[0]
        for i in range(1, n):
            vec += v[i]*B[i]
        R = 0.99*RR(vec.norm()^2)
        print("Norm=", RR(vec.norm()), ", Vector=", vec)
    else:
        break
print("End")
```

BKZ (1/3): Definition and Properties

RIKKYO UNIVERSITY

- **Block Korkine-Zolotarev (BKZ)-reduction**
  - A blockwise generalization of LLL with blocksize β
  - $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ is **β-BKZ-reduced** if it satisfies two conditions
    - ① It is size-reduced (same as LLL)
    - ② The k-th Gram-Schmidt vector $\mathbf{b}_k^*$ is shortest in $L_{[k,\, \ell]}$ with $\ell = \min(k + \beta - 1,\ n)$ for all $1 \leq k < n$

$$
\begin{array}{lccccccc}
L_{[1,\,\beta]}: & \mathbf{b}_1 & \cdots & \cdots & \mathbf{b}_\beta & & & \\
L_{[2,\,\beta+1]}: & & \pi_2(\mathbf{b}_2) & \cdots & \cdots & \pi_2(\mathbf{b}_{\beta+1}) & & \\
\vdots & & & \ddots & & & \ddots & \\
L_{[n-\beta+1,\,n]}: & & & & \pi_{n-\beta+1}(\mathbf{b}_{n-\beta+1}) & \cdots & \cdots & \pi_{n-\beta+1}(\mathbf{b}_n)
\end{array}
$$

  - Every β-BKZ-reduced basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ of a lattice L satisfies

$$\|\mathbf{b}_1\| \leq \gamma_\beta^{\frac{n-1}{\beta-1}} \lambda_1(L)$$

    - γ_β: Hermite's constant of dimension β, i.e., $\gamma_\beta = \sup_L {}^{\lambda_1(L)^2}/_{\mathrm{vol}(L)^{2/n}}$
    - As β increases, $\gamma_\beta^{1/(\beta-1)}$ decreases and thus $\mathbf{b}_1$ can be shorter

BKZ (2/3): Basic Algorithm

RIKKYO UNIVERSITY

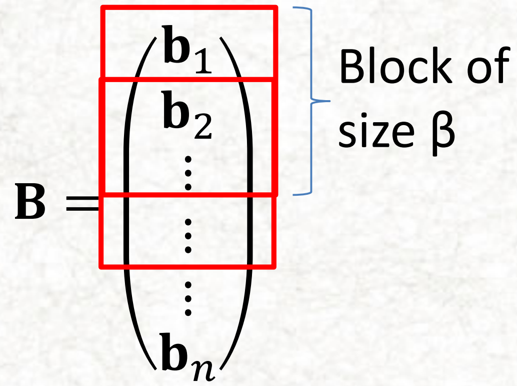
- **It consists of LLL and ENUM:**
  - Call ENUM to find a non-zero shortest vector in $L_{[k,\ell]}$
  - Call LLL to reduce a projected block basis of $L_{[k,\ell]}$

Algorithm: The basic BKZ Schnorr and Euchner (1994)

Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, a blocksize $2 \le \beta \le n$, and a reduction parameter $\frac{1}{4} < \delta < 1$ of LLL

Output: A β-DeepBKZ-reduced basis $\mathbf{B}$ of L

```
B ← LLL(B, δ)  // Compute μ_{i,j} and ‖b*_j‖² of the new basis B together
z ← 0, j ← 0
while z < n − 1 do
    j ← (j mod (n − 1)) + 1, k ← min(j + β − 1, n), h ← min(k + 1, n)
    Find v ∈ L such that ‖π_j(v)‖ = λ_1(L_[j,k]) by enumeration or sieve
    if ‖π_j(v)‖² < ‖b*_j‖² then
        z ← 0 and call LLL((b_1, ..., b_{j−1}, v, b_j, ..., b_h), δ)  // Insert v ∈ L and
        remove the linear dependency to obtain a new basis
    else
        z ← z + 1 and call LLL((b_1, ..., b_h), δ)
    end if
end while
```



※ As reference, please look at BKZ-60 – YouTube by Martin Albrecht

BKZ (3/3): Sage Code

RIKKYO UNIVERSITY

```
def ENUM(B, n, R, g, h):
    BB, U = GSO(B, n)
    Bnn = vector(QQ, n)
    for i in range(n):
        Bnn[i] = BB[i].norm()^2
    BB, U = GSO(B, n)
    sigma = Matrix(QQ, n+1, n)
    r = vector(ZZ, n+1)
    rho = vector(QQ, n+1)
    v = vector(ZZ, n)
    c = vector(QQ, n)
    w = vector(ZZ, n)
    for i in range(n+1):
        r[i] = i
    v[g] = 1
    last_nonzero = 1
    k = g + 1
    flag = 0
    v1 = vector(ZZ, n)
    while (1):
        rho[k-1] = rho[k] + (v[k-1] - c[k-1])^2*Bnn[k-1]
        if rho[k-1] <= R:
            if k==g+1:
                R = 0.99*rho[k-1]
                flag += 1
                for i in range(n):
                    v1[i] = v[i]
            k = k-1
            r[k-1] = max(r[k-1], r[k])
            for i in range(k+1, r[k]+1)[::-1]:
                sigma[i-1, k-1] = sigma[i, k-1] + U[i-1, k-1]*v[i-1]
            c[k-1] = -sigma[k, k-1]
            v[k-1] = round(c[k-1])
            w[k-1] = 1
        else:
            k = k+1
            if k==h+1:
                if flag == 0:
                    return False
                else:
                    vv = v1[g]*B[g]
                    for i in range(g+1, h+1):
                        vv += v1[i]*B[i]
                    return vv
            r[k-1] = k
            if k>=last_nonzero:
                last_nonzero = k
                v[k-1] = v[k-1] + 1
            else:
                if v[k-1] > c[k-1]:
                    v[k-1] = v[k-1] - w[k-1]
                else:
                    v[k-1] = v[k-1] + w[k-1]
                w[k-1] = w[k-1] + 1
```

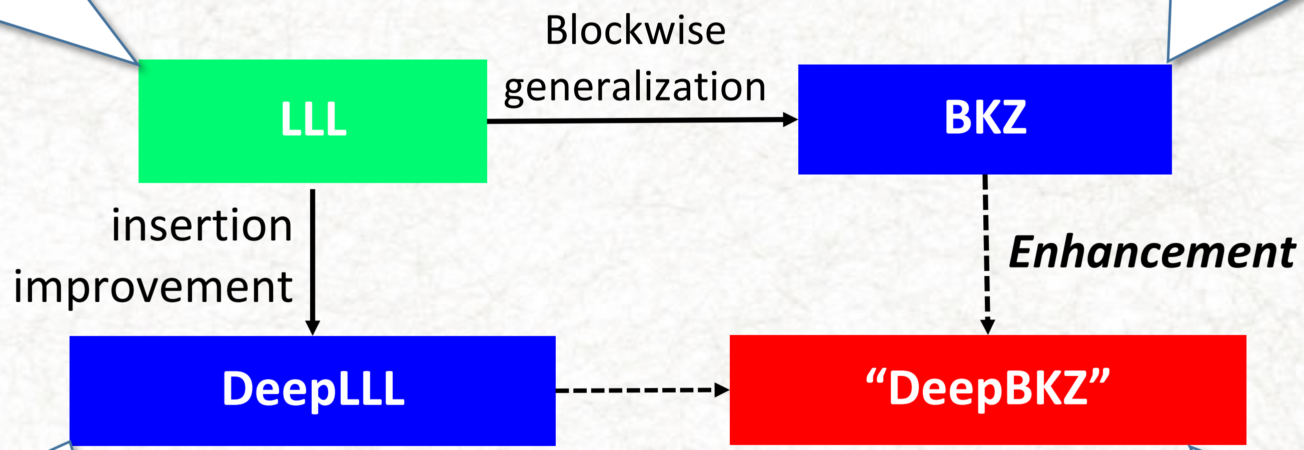
```
def BKZ(B, n, block):
    B.LLL()
    BB, U = GSO(B, n)
    Bnn = vector(QQ, n)
    for i in range(n):
        Bnn[i] = BB[i].norm()^2
    z = 0
    k = -1
    while z < n-1:
        k = lift(mod(k+1, n-2))
        l = min(k+block-1, n-1)
        h = min(l+1, n-1)
        print("(k, l, h) = ", k, l, h)

        R = 0.99*Bnn[k]
        v = 0
        v = ENUM(B, n, R, k, l)
        if v != 0:
            z = 0
            C = Matrix(ZZ, h+1, n)
            for i in range(k):
                C[i] = B[i]
            C[k] = v
            for i in range(k+1, h+1):
                C[i] = B[i-1]
            C = C.LLL()
            for i in range(1, h+1):
                B[i-1] = C[i]
            BB, U = GSO(B, n)
            Bnn = vector(QQ, n)
            for i in range(n):
                Bnn[i] = BB[i].norm()^2
        else:
            z += 1
            B = B.LLL()
            BB, U = GSO(B, n)
            Bnn = vector(QQ, n)
            for i in range(n):
                Bnn[i] = BB[i].norm()^2

n = 20; d = 1000000
B = Matrix(ZZ, n)
for i in range(0, n):
    B[i, i] = 1
    B[i, 0] = randint(-d, d)
show(B)
B = B.LLL()
BKZ(B, n, 10)
show(B)
```

DeepBKZ (1/6): New Reduction

RIKKYO UNIVERSITY

- The most famous and typical
- Only adjacent basis vectors can be swapped
- Polynomial-time algorithm

- Blockwise generalization of LLL
- It calls enumeration over block projected lattices, and LLL before enumeration



- Simple generalization of LLL
- Non-adjacent basis vectors can be swapped
- No longer polynomial-time

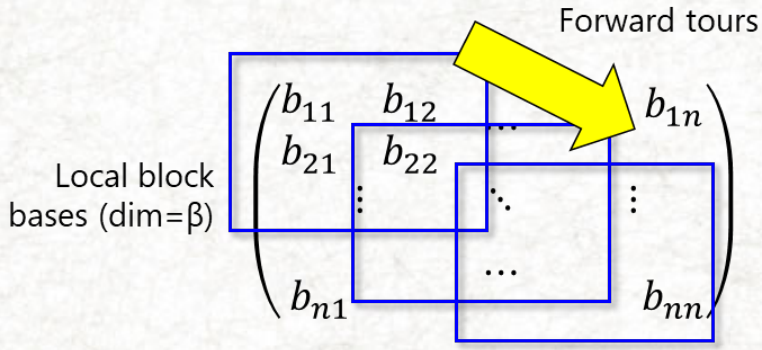
- **New Reduction** [YY17]
- **A combination of DeepLLL and BKZ**

[YY17] J. Yamaguchi and M. Yasuda, Explicit formula for Gram-Schmidt vectors in LLL with deep insertions and its applications, in: NuTMiC 2017, Lecture Notes in Computer Science 10737, Springer, pp. 142–160, 2017.

DeepBKZ (2/6): Basic Construction

RIKKYO UNIVERSITY

- **Enhancement of BKZ**
  - Call **DeepLLL** as a subroutine in BKZ
    - DeepLLL is a straightforward generalization of LLL
    - It is called before every SVP oracle over a β-dimensional lattice



A) DeepLLL in global
B) SVP oracle over local blocks

LLL	DeepLLL
Only adjacent basis vectors are swapped	Non-adjacent basis vectors can be changed ⇒ shorter basis vectors can be found
$\mathbf{B} \leftarrow (\mathbf{b}_1, \ldots, \mathbf{b}_{i+1}, \mathbf{b}_i, \ldots, \mathbf{b}_n)$ New basis **swap**	$\mathbf{B} \leftarrow (\mathbf{b}_1, \ldots, \mathbf{b}_k, \mathbf{b}_i, \ldots, \mathbf{b}_{k-1}, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n)$ **deep insertion**

- **Features**
  - Even small β can find a very short lattice vector
  - DeepLLL is somewhat costly
  - But for β≧30, SVP-calls are dominant and the cost is same as BKZ

DeepBKZ (3/6): Gram-Schmidt Formula

RIKKYO UNIVERSITY

- **Complex basis transformation (general form)**
  - $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \to \mathbf{C} = (\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}, \mathbf{v}, \mathbf{b}_k, \ldots, \mathbf{b}_{n-1})$ *insert*
    - $\mathbf{v} = \sum_{i=1}^{n} x_i \mathbf{b}_i = \sum_{i=1}^{n} \nu_i \mathbf{b}_i^*$ for some $x_i \in \mathbb{Z}$ with $x_n = \pm 1$
- **Gram-Schmidt formula in DeepLLL[YY+15, YY17]**
  - This enables to make DeepLLL practical like LLL

Proposition: Gram-Schmidt orthogonalization $[\mathbf{c}_1^*, \ldots, \mathbf{c}_n^*]$ of $\mathbf{C}$

Set $m = \max\{k \le i \le n \mid \nu_i \neq 0\}$. Then we have

$$\mathbf{c}_j^* = \begin{cases} \sum_{i=k}^{m} \nu_i \mathbf{b}_i^* & \text{for } j = k, \\ \frac{D_j}{D_{j-1}} \mathbf{b}_{j-1}^* - \sum_{i=j}^{m} \frac{\nu_i \nu_{j-1} \|\mathbf{b}_{j-1}^*\|^2}{D_{j-1}} \mathbf{b}_i^* & \text{for } k+1 \le j \le m+1, \\ \mathbf{b}_{j-1}^* & \text{for } m+2 \le j \le n+1, \end{cases}$$

where $D_\ell = \sum_{i=\ell}^{m} \nu_i^2 \|\mathbf{b}_i^*\|^2$ for $1 \le \ell \le m$. In particular, $\mathbf{c}_{m+1}^* = \mathbf{0}$.
For $k+1 \le j \le m$, we have

$$\|\mathbf{c}_j^*\|^2 = \frac{D_j}{D_{j-1}} \|\mathbf{b}_{j-1}^*\|^2.$$

[YY+15] M. Yasuda, K. Yokoyama et al. Analysis of decreasing squared-sum of Gram-Schmidt lengths for short lattice vectors, Journal of Mathematical Cryptology, Vol. 11, No. 1, pp. 1—24 (2015).

DeepBKZ (4/6): Properties of Reduction

RIKKYO UNIVERSITY

- **$\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$: DeepBKZ-reduced**
  - **① δ-DeepLLL-reduced (1/4<δ<1)**
    - Size-reduced
    - $\delta \|\mathbf{b}_i^*\|^2 \leq \|\pi_i(\mathbf{b}_k)\|^2$ for all $i < k$
  - **② β-BKZ-reduced (2≦β≦n)**
    - $\|\mathbf{b}_i^*\| = \lambda_1(\pi_i(L))$ for all $1 \leq i \leq n$

      (π_i is the orthogonal projection)
- Then we have

$$\delta \|\mathbf{b}_1\|^2 \leq \left\|\mathbf{b}_{\beta+1}\right\|^2 \leq \left\|\mathbf{b}_{\beta+1}^*\right\|^2 + \frac{1}{4}\sum_{j=1}^{\beta} \left\|\mathbf{b}_j^*\right\|^2$$

$$\Rightarrow \left(\delta - \frac{1}{4}\right)\frac{\|\mathbf{b}_1\|^2}{\left\|\mathbf{b}_{\beta+1}^*\right\|^2} \leq 1 + \frac{1}{4}\sum_{j=2}^{\beta} \frac{\left\|\mathbf{b}_j^*\right\|^2}{\left\|\mathbf{b}_{\beta+1}^*\right\|^2} \leq 1 + \frac{C_\beta}{4},$$

where $C_\beta = \max \sum_{i=1}^{\beta-1} \frac{\left\|\mathbf{b}_i^*\right\|^2}{\left\|\mathbf{b}_\beta^*\right\|^2}$ over all HKZ-reduced $(\mathbf{b}_1, \ldots, \mathbf{b}_\beta)$

DeepBKZ (5/6): Provable Output Quality

RIKKYO UNIVERSITY

- **Lemma**[YNY20]
  - Every DeepBKZ-reduced basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ satisfies

$$\frac{\|\mathbf{b}_1\|^2}{\left\|\mathbf{b}_{i\beta+1}^*\right\|^2} \leq \alpha\left(1 + \frac{C_\beta}{4}\right)\left\{1 + \frac{\alpha\left(1 + C_\beta\right)}{4}\right\}^{i-1}$$

for $i \geq 1$, where $\alpha = \frac{4}{4\delta - 1} > \frac{4}{3}$

- **Theorem**[YNY20]
  - $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$: (δ, β)-DeepBKZ-reduced basis of L
  - Assume n is divisible by β with $p = \frac{n}{\beta} \geq 2$
  - Then we have

$$\frac{\|\mathbf{b}_1\|}{\mathrm{vol}(L)^{1/n}} \leq \sqrt{\gamma_\beta}\left\{\alpha\left(1 + \frac{C_\beta}{4}\right)\right\}^{\frac{\beta(p-1)}{2n}}\left\{1 + \frac{\alpha\left(1 + C_\beta\right)}{4}\right\}^{\frac{\beta(p-1)(p-2)}{4n}}$$

where γ_β is Hermite's constant of dimension β

[YNY20] M. Yasuda, S. Nakamura and J. Yamaguchi, Analysis of DeepBKZ reduction for finding short lattice vectors, Design, Codes and Cryptography, Vol. 88, No. 19, pp. 2077—2100 (2020).

DeepBKZ（6/6）: Practical Output Quality

RIKKYO UNIVERSITY

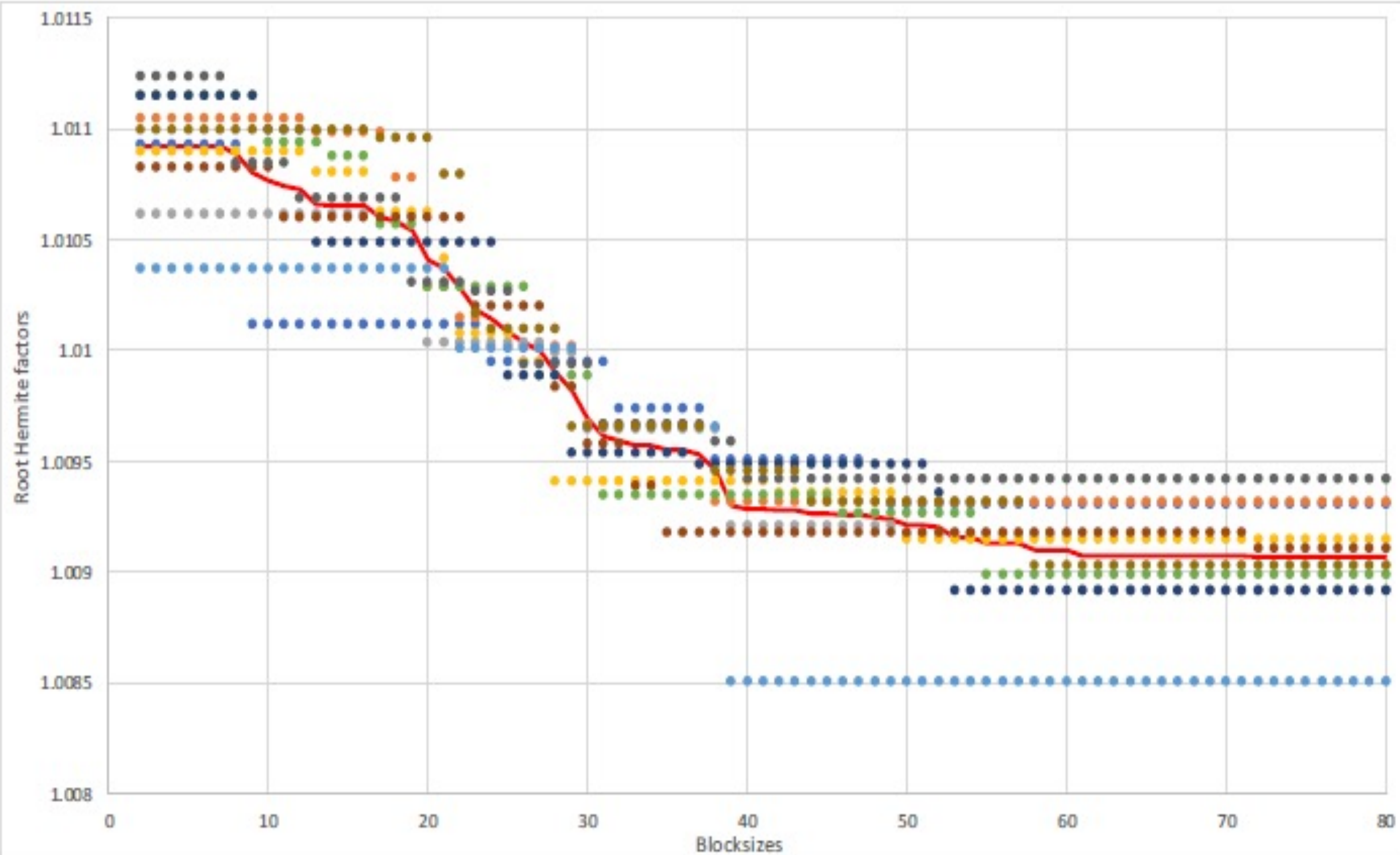


Fig. 1 The root Hermite factor of DeepBKZ with blocksizes $2 \leq \beta \leq 80$ for the SVP challenge in dimension $n = 115$ with seeds 0–9 (Each dot denotes the root Hermite factor for some seed, and the polygonal line denotes the average.)

New SVP Solutions by (Parallel) DeepBKZ

RIKKYO UNIVERSITY

- **DeepBKZ found many new solutions for the SVP challenge**
  - In most dimensions up to $n = 128$
  - We used blocksizes $\beta = 30$--45
  - Our solutions are the shortest or very close to it
    - Since their approximation factors are close to 1.0 (0.98470 for $n = 128$)
  - For $n = 128$, it took about 57.5 hours by massive parallel computation using 24,576 cores

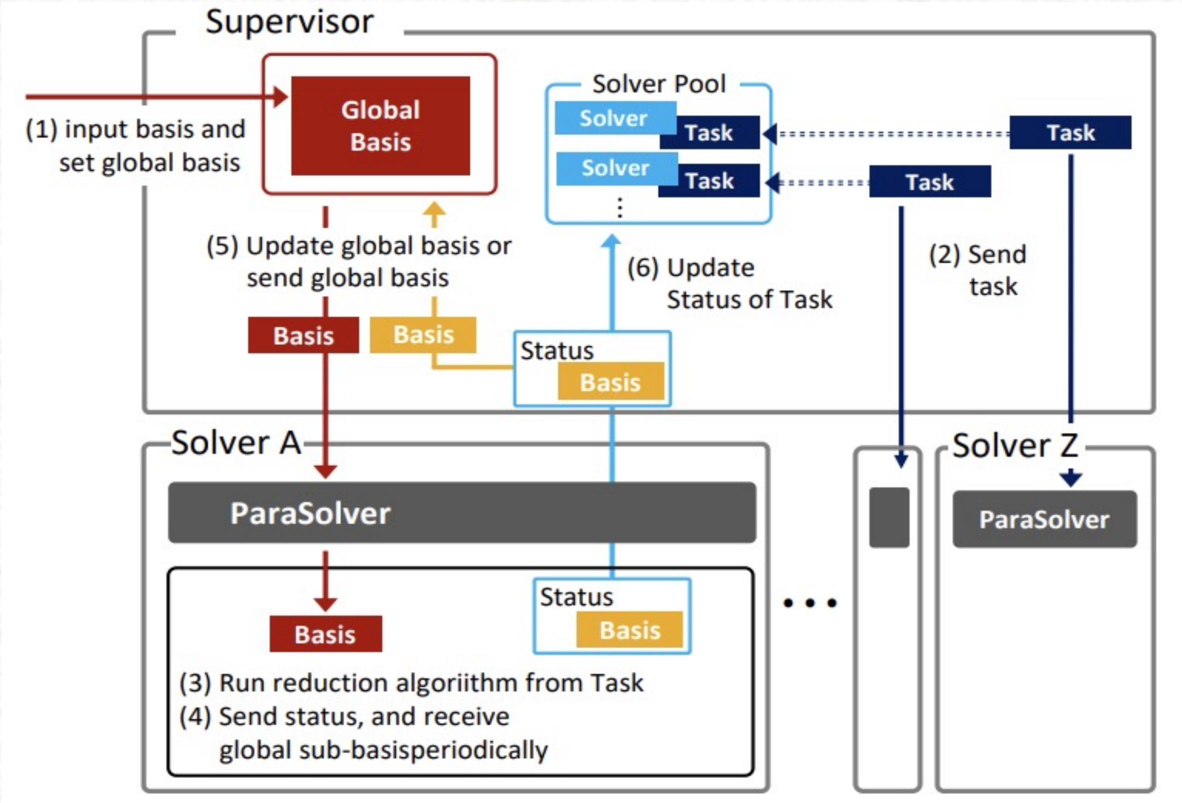
67	130	3025	0	Kenji Kashiwabara and Masaharu Fukase	vec	Other	2013-11-15	1.04787
68	129	2818	1	Yuga Miyagi and Eiichiro Fujisaki	vec	Sieving	2019-04-11	0.98161
69	129	2855	0	Yuga Miyagi, Tomohiro Sekiguchi, Eiichiro Fujisaki	vec	Sieving	2019-03-26	0.99172
70	129	2875	0	Martin Albrecht, Leo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn Postlethwaite, Marc Stevens	vec	Sieving	2018-08-30	0.99878
71	129	2988	0	Martin R. Albrecht, Leo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn Postlethwaite and Marc Ste	vec	Sieving	2018-08-30	1.03813
72	128	2812	1	N. Tateiwa, Y. Shinano, K. Yamamura, A. Yoshida, S. Kaji, M. Yasuda, K. Fujisawa	vec	BKZ	2021-10-16	0.98470
73	128	2882	0	Kenji KASHIWABARA and Tadanori TERUYA	vec	Other	2018-07-9	1.00477
74	128	2948	0	Kenji KASHIWABARA and Tadanori TERUYA	vec	Other	2018-06-20	1.02755
75	128	2974	0	Kenji KASHIWABARA and Tadanori TERUYA	vec	Other	2018-05-9	1.03665
76	128	2984	0	Kenji Kashiwabara and Masaharu fukase	vec	Other	2013-09-23	1.04017
77	128	2992	0	Kenji Kashiwabara and Masaharu Fukase	vec	Other	2013-09-19	1.04313
78	127	2790	3	N. Tateiwa, Y. Shinano, A. Yoshida, S. Nakamura, S. Kaji, M. Yasuda, Y. Aono, K. Fujisawa	vec	ENUM,BKZ,Other	2020-06-7	0.97573
79	127	2890	1	Yuga Miyagi and Eiichiro Fujisaki	vec	Sieving	2019-04-11	1.01429
80	127	2898	0	Yuga Miyagi, Tomohiro Sekiguchi, Eiichiro Fujisaki	vec	Sieving	2019-03-25	1.01626
81	127	2932	2	Junpei Yamaguchi, Masaya Yasuda and Takuya Hayashi	vec	Other	2018-01-12	1.02804
82	126	2812	0	Tadanori TERUYA	vec	Sieving,Other	2019-04-2	0.99052
83	126	2855	0	Yoshinori Aono and Phong Nguyen	vec	ENUM,BKZ	2014-09-9	1.00556
84	126	2897	0	Kenji KASHIWABARA and Tadanori TERUYA	vec	Other	2014-08-27	1.02051
85	126	2906	0	Yoshinori Aono	vec	ENUM,BKZ	2014-07-14	1.02357
86	126	2944	0	Kenji Kashiwabara and Masaharu Fukase	vec	Other	2013-09-4	1.03679
87	126	2969	42	Yuanmi Chen and Phong Nguyen	vec	ENUM,BKZ	2013-04-12	1.04356
88	125	2806	3	Jim Johnson	vec	Sieving	2021-10-22	0.99077
89	125	2834	0	Tadanori TERUYA	vec	Sieving,Other	2019-04-2	1.00341
90	125	2907	3	Junpei Yamaguchi, Masaya Yasuda and Takuya Hayashi	vec	Other	2017-11-20	1.02649
91	125	2922	8	Junpei Yamaguchi, Masaya Yasuda and Takuya Hayashi	vec	ENUM,Other	2017-10-15	1.03203

SVP Challenge (latticechallenge.org)

Massive Parallelization of DeepBKZ (1/8)

RIKKYO UNIVERSITY

- **Parallel sharing DeepBKZ**
  - Distributed and asynchronous system using randomization and DeepBKZ
  - Using CMAP-LAP[TS+21], a general framework for lattice algorithms



<u>Supervisor-Solvers Style</u>

- Every solver runs DeepBKZ on a randomized basis independently
- Supervisor collects short basis vectors from solvers, and distributes them to solvers
- **Every solver uses distributed vectors to accelerate its reduction process**

(See [TS+20] for sharing a shortest basis vector)

[TS+20] N. Tateiwa, Y. Shinano, S. Nakamura, A. Yoshida, S. Kaji, M. Yasuda and K. Fujisawa, Massive Parallelization for Finding Shortest Lattice Vectors Based on Ubiquity General Framework, High Performance Computing, Networking, Storage, and Analysis (SC 20).

[TS+21] N. Tateiwa, Y. Shinano, K. Yamamura, A. Yoshida, S. Kaji, M. Yasuda and K. Fujisawa, CMAP-LAP: Configurable Massively Parallel Solver for Lattice Problems, High Performance Computing, Data, and Analytics (HiPC 2021).

Massive Parallelization of DeepBKZ（2/8）

RIKKYO UNIVERSITY

- **Efficacy of parallel sharing DeepBKZ**
  - Sharing k = 16 short basis vectors among solvers for dimension d = 120

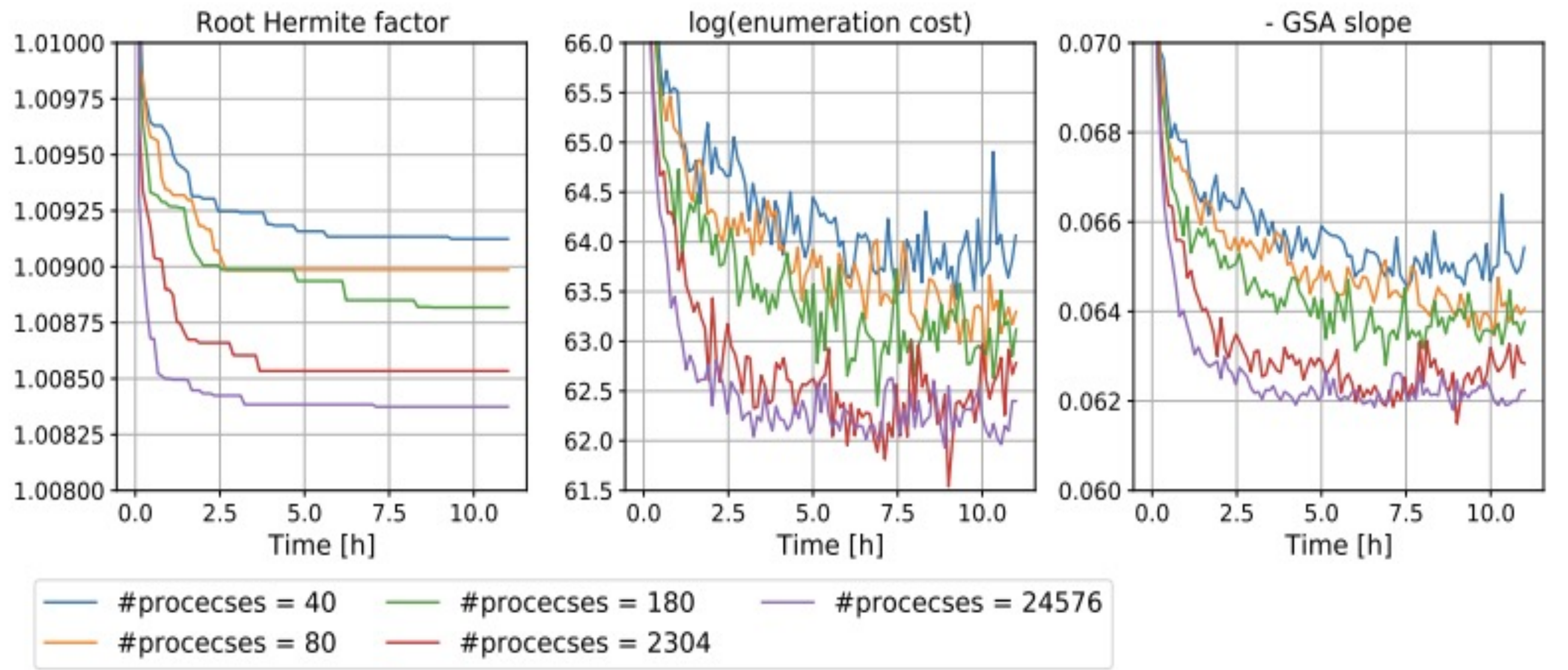


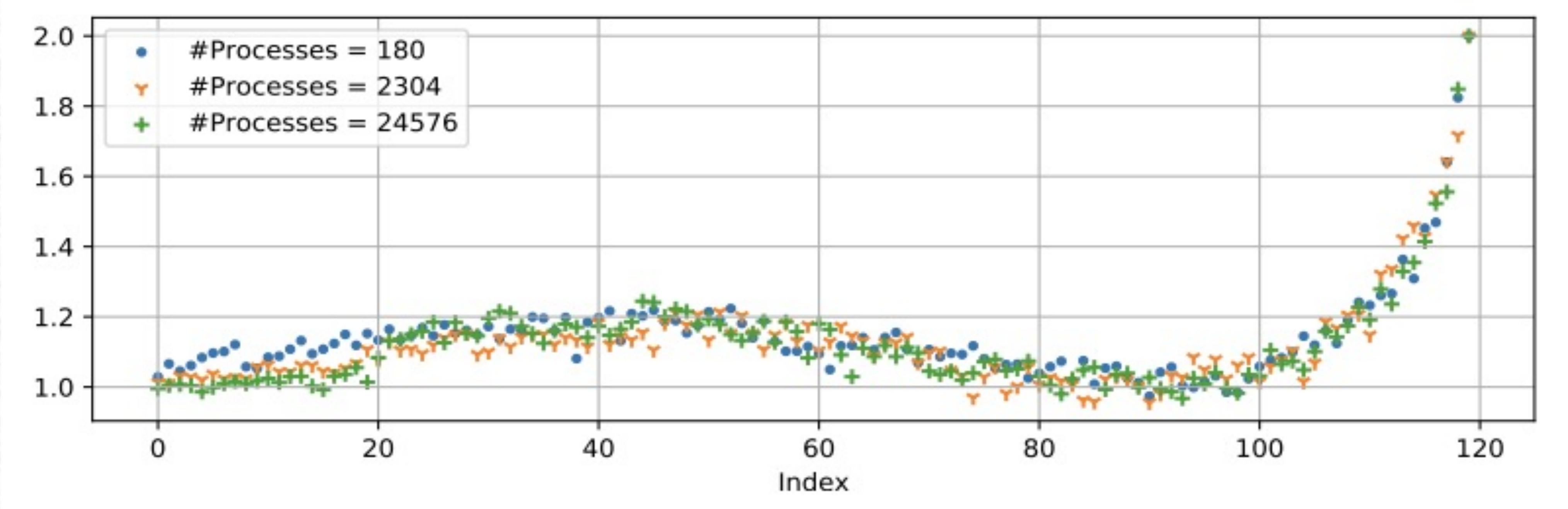
Fig. 11 Same as Figure 7, but the dimension is $d = 120$ and lines in each metric represent difference by different numbers of processes (We used $k = 16$ as the number of shares)

Massive Parallelization of DeepBKZ (3/8)

RIKKYO UNIVERSITY

- **Output quality of parallel sharing DeepBKZ**
  - For an output basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of parallel sharing DeepBKZ,
  - Gaps $\frac{\|\mathbf{b}_i^*\|}{\mathrm{GH}(\pi_i(L))}$ are shown in the below Figure

  ⇒ First k = 16 basis vectors are close to the shortest in projected lattices



Massive Parallelization of DeepBKZ (4/8)

RIKKYO UNIVERSITY

- **Similarity of lattice bases**
  - Our parallel system is based on randomness of lattice bases
  - But we have a trade-off between randomness and shared information
- **Measurement of similarity based on Grassmann metrics**
  - Given a multiset $\mathcal{B} = \{\mathbf{B}_1, \dots, \mathbf{B}_m\}$ of lattice bases in dimension d
  - The **total projected diversity** is defined as $\mathrm{Div}\left(\mathcal{B}, d_g\right) = \frac{1}{d}\sum_{i=1}^{d} \mathrm{Div}^i(\mathcal{B}, d_g)$
    - $\mathrm{Div}^i\left(\mathcal{B}, d_g\right) = \frac{1}{\#P(\mathcal{B})}\sum_{(\mathbf{B}_j,\mathbf{B}_k)\in P(\mathcal{B})} d_g\left(Y^i\left(\mathbf{B}_j\right), Y^i(\mathbf{B}_k)\right)$: the i-th projected diversity
      - $Y^i([\mathbf{b}_1, \dots, \mathbf{b}_d]) = \left(\frac{\mathbf{b}_{i+1}^*}{\|\mathbf{b}_{i+1}^*\|}, \dots, \frac{\mathbf{b}_d^*}{\|\mathbf{b}_d^*\|}\right)$: normalized Gram-Schmidt vectors
    - d_g: Grassmann metric (equipped for the Grassmannian manifold $Gr(k,n)$)
      - $Gr(k,n)$ consists of k-dimensional linear subspaces in $\mathbb{R}^n$
        - » Every subspace is considered as a point in $Gr(k,n)$
      - d_g measures a distance between two points (=subspaces) in $Gr(k,n)$
        - » E.g., geodesic, chordal, projection 2-norm distances

Massive Parallelization of DeepBKZ（5/8）

RIKKYO UNIVERSITY

- **Experiments for the diversity of bases in our parallel system**
  - The number of shares k≦16 does not lose the diversity significantly for d≧90
  - In contrast, k≧32 loses the diversity gradually in reduction process

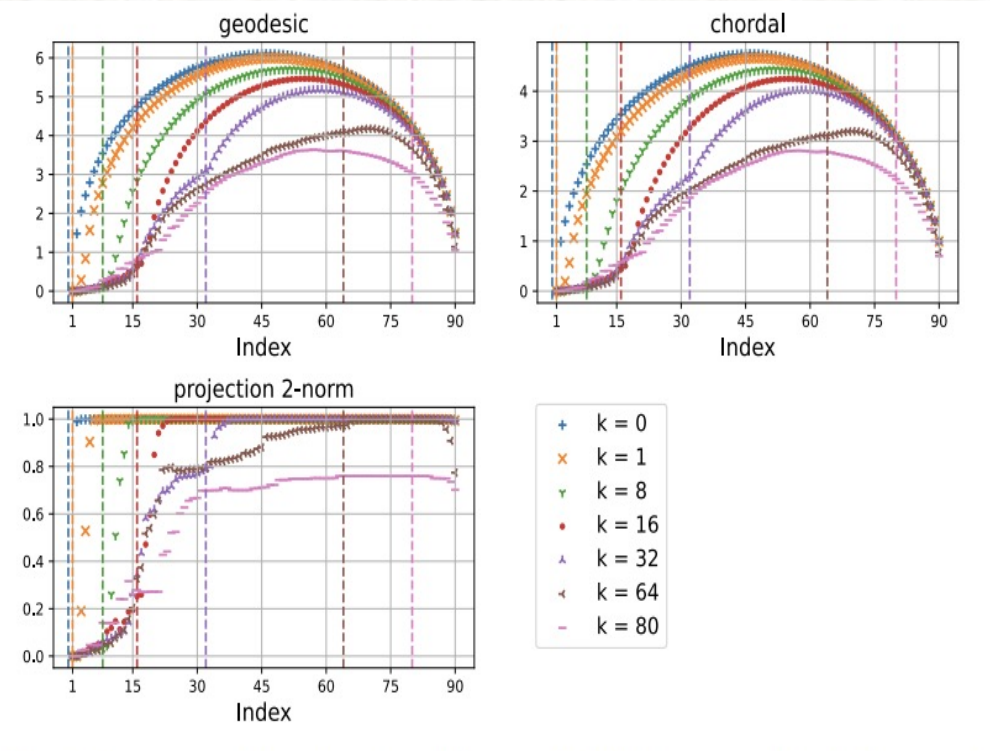


Fig. 2 The average of the i-th projected diversity $\mathsf{Div}^i(\mathcal{B}, d_g)$ computed for 90-dimensional lattice bases with different numbers of shared vectors k right after 100 DeepBKZ tours.)

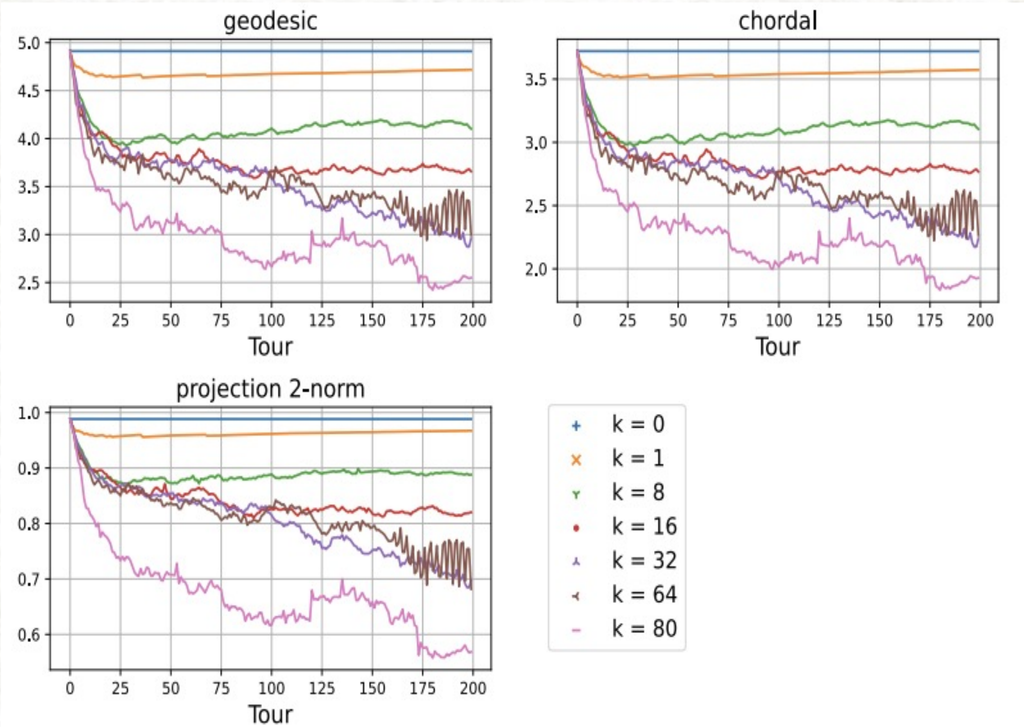


Fig. 3 Transition of the total diversity $\mathsf{Div}(\mathcal{B}, d_g)$ computed for 90-dimensional lattice bases with different numbers of shared vectors k after each tour of DeepBKZ.

[TS+21] N. Tateiwa, Y. Shinano, M. Yasuda, S. Kaji, K. Yamamura and K. Fujisawa, "Massively parallel sharing lattice basis reduction", ZIB-Report 21-38, available at https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/8520

Massive Parallelization of DeepBKZ（6/8）

RIKKYO UNIVERSITY

- **Large-scale experiments**
  - A very short lattice vector in a lattice of dimension around d = 130 can be found within 100 hours on supercomputers
    - Without using the sub-sieve strategy
  - **Small blocksizes β = 30—40** are enough for parallel sharing DeepBKZ

Table 1 Computing platforms, operating systems, compilers and libraries

Machine	Memory / node	CPU	CPU frequency	# nodes	# cores
Lisa	384 GB	Xeon Platinum 9242	2.30 GHz	1,080	103,680
Emmy	384 GB	Xeon Platinum 9242	2.30 GHz	128	12,288
ITO	192 GB	Xeon Gold 6154	3.00 GHz	128	4,608
CAL A	256 GB	Xeon CPU E5-2640 v3	2.60 GHz	4	64
	256 GB	Xeon CPU E5-2650 v3	2.30 GHz	4	80
CAL C	32 GB	Xeon CPU E3-1284L v3	1.80 GHz	45	180

Operating systems and versions: Lisa and Emmy [CentOS Linux release 7.7.1908], ITO [Red Hat Enterprise Linux Server release 7.3.1611], CAL A and CAL C [CentOS Linux release 7.9.2009]. *Compilers and versions*: Lisa and Emmy [intel19.0.5, impi2019.5], ITO [icc 19.1.1.217, impi2019.4], CAL A [icc 19.1.3.304, openmpi4.0.5], CAL C [icc19.1.3.304, impi2020.4.304]. *Libraries and versions*: NTL v11.3.3, Eigen v3.3.7, gsl v2.6, OpenBLAS v0.3.7, fplll v5.2.1.

Table 5 Large-scale experimental results of CMAP-DeepBKZ for SVP instances in dimensions d = 128, 130 and 132 ($\mathbf{b}_1$ denotes a shortest basis vector of all solver's bases, and "Updated time" is wall time to update final shortest vectors found)

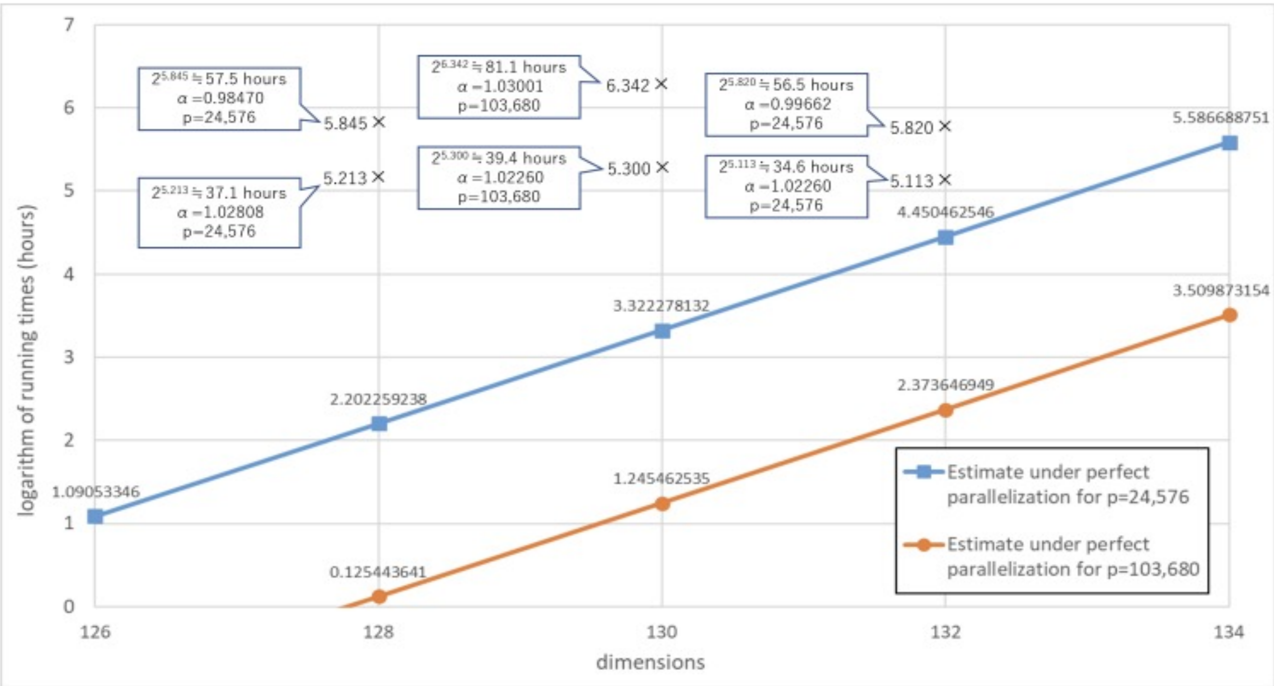
SVP Instance Dim.	Seed	# of cores*	Updated time [h]	Norm of $\mathbf{b}_1$	Approx. factor $\frac{\|\mathbf{b}_1\|}{\mathrm{GH}(L)}$	Root Hermite factor $\gamma^{1/d}$	Machine* (Table 1)
128	1†	24,576	57.5	2812.00	0.98470	1.00796	Emmy
	2	24,576	37.1	2947,45	1.02808	1.00830	Emmy
130	3	103,680	81.1	2968.73	1.03001	1.00825	Lisa
	7	103,680	39.4	2914.22	1.01236	1.00811	Lisa
132	1	24,576	34.6	2968.05	1.02260	1.00812	Emmy
	2	24,576	56.5	2899.90	0.99662	1.00818	Emmy

† a new solution for the Darmstadt SVP challenge [26] in dimension 128 (see also Table 6 for other dimensions). * We list the maximum number of cores and machines used for executions, including restarts, and the wall time for the updated time.

Massive Parallelization of DeepBKZ（7/8）

RIKKYO UNIVERSITY

- **Comparison with perfect parallelization (ideal model)**
  - [ABF+20] proposed a variant of enumeration-based BKZ with several improvements
  - It estimated the cost of enumeration-based SVP solving (without parallelization)
  - We compare running times of our parallel system with perfect parallelization of [ABF+20]
    - In case of using p = 24,576, our system is about 10 times slower than an ideal model for d = 128, 132
    - In case of using p = 103,680, our system is much slower for d = 130 (ours could be considerably improved)



[ABF+20] Albrecht, M.R., Bai, S., Fouque, P.A., Kirchner, P., Stehle, D., Wen, W.: Faster enumeration-based lattice reduction: Root Hermite factor k^{1/(2k)} time k^{k/8+o(k)}. In: Advances in Cryptology–CRYPTO 2020, Lecture Notes in Computer Science, vol. 12171, pp. 186–212.

Massive Parallelization of DeepBKZ (8/8)

RIKKYO UNIVERSITY

- **Future Work: Use of our CMAP-LAP framework**[TS+21]
  - Supervisor-Worker parallelization type
  - **Execution of heterogeneous algorithms** (Reduction, ENUM, Sieve)
  - Acceleration by asynchronously sharing lattice vectors via vector pool

  ⇒ We will embed optimal algorithms (e.g., pruned ENUM, sieve) in our framework for solving lattice problems of higher dimensions