Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator

Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi

Yuncong Zhang

April 10, 2020

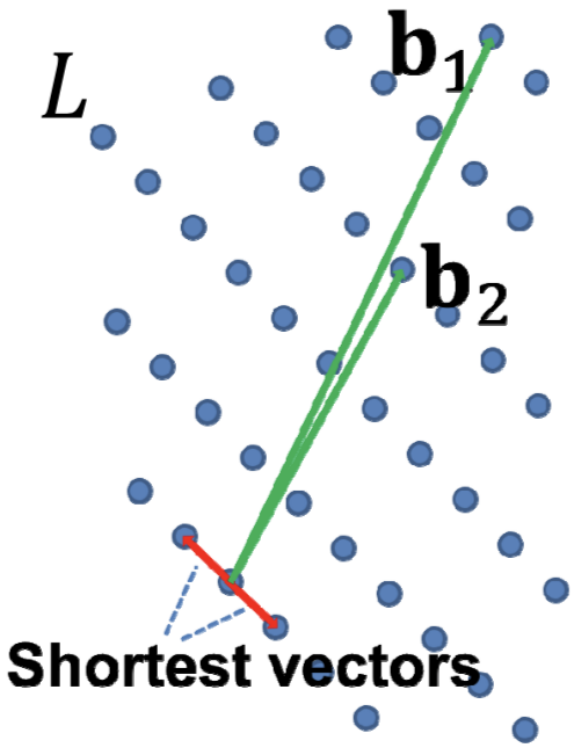
Outline

1. Introduction
2. Preliminaries
   - Mathematical Definitions
   - Enumeration and LLL Algorithm
   - Previous BKZ Algorithms
3. Improved Progressive BKZ
   - Optimizations
   - Implementation Detail
   - Simulation and Comparison

Introduction

The *Shortest Vector Problem* (SVP):

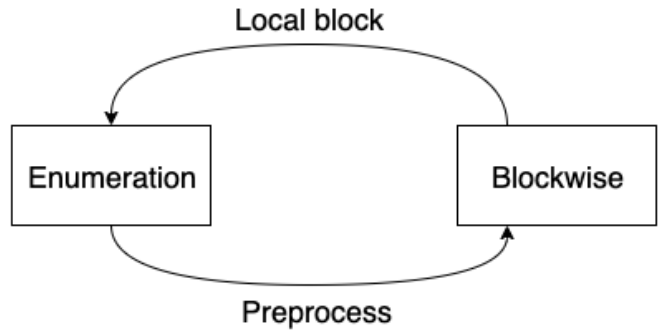
- Given the lattice $L = L(\vec{b}_1, \cdots, \vec{b}_n)$
- Find the shortest non-zero vector $\vec{v}^* \in L$, denote $\lambda_1(L) := \|\vec{v}^*\|$
- γ-SVP: find $\vec{v}$ with $\|\vec{v}\| \leq \gamma \cdot \lambda_1(L)$



Introduction

Current algorithms for solving SVP:

- **Blockwise (LLL, BKZ)**: Optimize the entire basis
- **Enumeration**: Enumerate vectors using given basis
- **Seiving**: Consumes large amount of memory, was impractical, but quite competitive now



Efficiency	Exponential	Fast ☺
Quality	Exact solution ☺	Exponential approx factor

Remark

Blockwise and Enumeration algorithms rely on and compliment each other.

- BKZ calls Enumeration on local blocks
- Enumeration efficiency relies on basis quality, often require basis preprocessed by blockwise algorithm

Introduction

History of BKZ Algorithms:

- Basic BKZ (Schnorr 1994)
- BKZ 2.0 (Chen-Nguyen 2011): a combination of improvements
- Progressive BKZ: used locally in BKZ 2.0, never formally proposed as a standalone algorithm

Preliminaries

- Mathematical Definitions
- Enumeration Algorithm
- LLL Algorithm
- Previous BKZ Algorithms

Mathematical Definitions

Gram-Schmidt Basis for $B = (\vec{b}_1, \cdots, \vec{b}_n)$:

- $B^* := (\vec{b}_1^*, \cdots, \vec{b}_n^*)$
- $\vec{b}_i^* := \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \cdot \vec{b}_j^*$ where $\mu_{ij} := \langle \vec{b}_i, \vec{b}_j^* \rangle / \|\vec{b}_j^*\|^2$ called GS coefficients

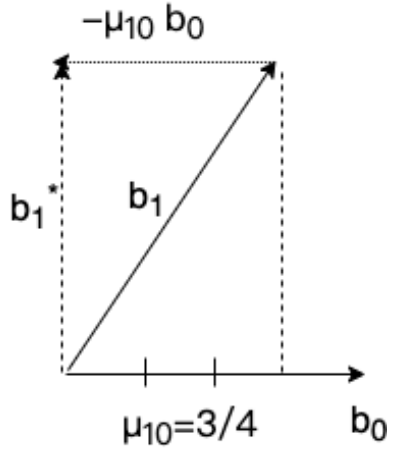


Figure: Example of GS coefficient μ_{10}

Mathematical Definitions

Gram-Schmidt reduction preserves volume, which is volume of a lattice cell:

$$\mathrm{vol}(L) := \det(L) := \det(B) = \det(B^*) = \prod_{i=1}^{n} \|\vec{b}_i^*\|$$

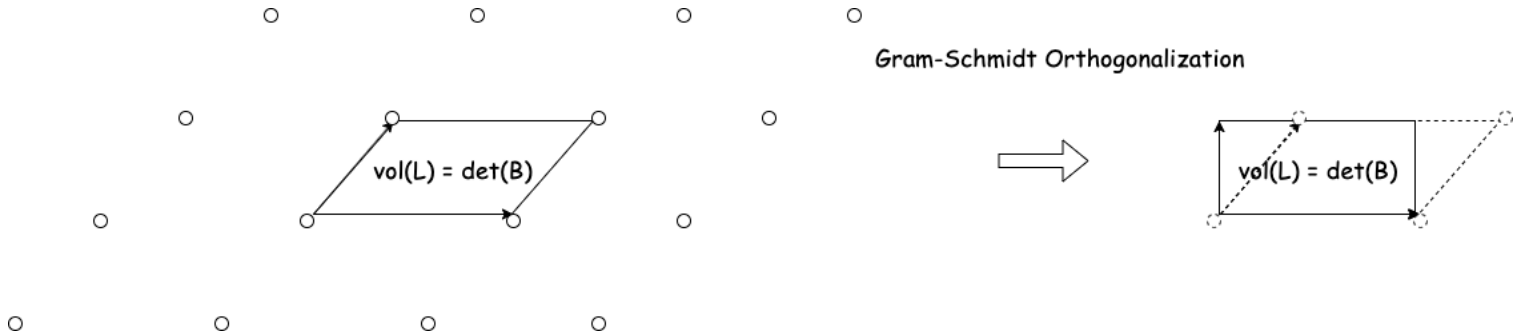


Figure: Gram-Schmidt orthogonalization and volume of lattice

Mathematical Definitions

Projection $\pi_i : \mathbb{R}^n \to \text{span}(\vec{b}_1, \cdots, \vec{b}_{i-1})^\perp$

$$\pi_i(v) := \vec{v} - \sum_{j=1}^{i-1} \langle \vec{v}, \vec{b}_j^* \rangle \vec{b}_j^* / \|\vec{b}_j^*\| \tag{1}$$

which effectively removes the proportion of the vector inside the space $\text{span}(\vec{b}_1, \cdots, \vec{b}_{i-1})$.

Remark

- *Gram-Schmidt reduction:* $\vec{b}_i^* = \pi_i(\vec{b}_i)$
- $\ker \pi_i = \text{span}(\vec{b}_1, \cdots, \vec{b}_{i-1})$. *In particular,* $\pi_i(\vec{b}_j) = 0$ *for* $j < i$
- $\pi_1(\cdot)$ *is the identity map*

Mathematical Definitions

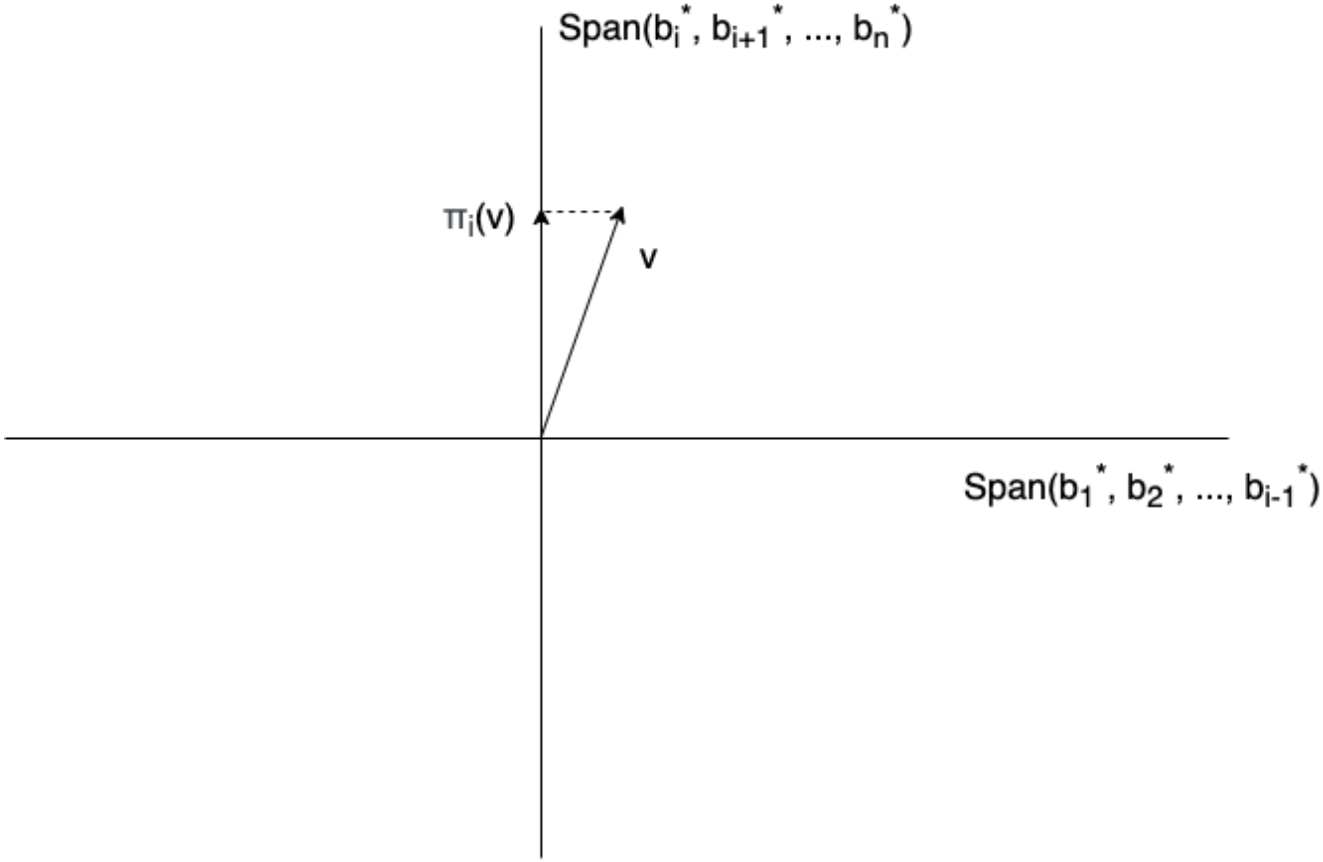


Figure: Projection π_i eliminates the portion of $\vec{v}$ in span$(\vec{b_1}, \cdots, \vec{b_{i-1}})$

Mathematical Definitions

Projective *local block* (sublattice)

$$L_{[i:j]} \quad := \quad \pi_i(L(\vec{b}_i, \vec{b}_{i+1}, \cdots, \vec{b}_j)) \tag{2}$$

Use $B_i := L_{[i:i+\beta-1]}$ when the blocksize β is clear

Mathematical Definitions

Gaussian Huristic

- For convex set S: $|S \cap L| \approx \mathrm{vol}(S)/\mathrm{vol}(L)$
- Let S be n-dimensional ball, solving $V_n(R) = \mathrm{vol}(L)$ gives $R = (\mathrm{vol}(L)/V_n(1))^{1/n}$
- Denote by $\mathrm{GH}(L) := (\mathrm{vol}(L)/V_n(1))^{1/n}$, approximates $\lambda_1(L)$

Remark

$V_n(R)$ is the volume of the n-dimensional ball.

$$V_n(R) = R^n \cdot \frac{\pi^{n/2}}{\Gamma(n/2+1)}$$

Enumeration Algorithm

Given a lattice basis $B = (\vec{b}_1, \cdots, \vec{b}_n)$, finds the shortest vector $\vec{v}^*$

$$\vec{v}^* = a_1\vec{b}_1 + a_2\vec{b}_2 + \cdots + a_n\vec{b}_n \qquad \forall i \in [n]$$

Observation:

- $\|\pi_n(\vec{v}^*)\| \leq \|\pi_{n-1}(\vec{v}^*)\| \leq \cdots \leq \|\pi_1(\vec{v}^*)\| = \|\vec{v}^*\| \leq R$
- $\pi_i(\vec{v}^*) = \pi_i(a_i\vec{b}_i + \cdots + a_n\vec{b}_n)$

Enumeration Algorithm

Select radius R, get searching tree:

- Layer at depth $k+1$ consists of all possible $(a_n, \cdots, a_{n-k})$ s.t. $\|\pi_{n-k}(a_n\vec{b}_n + \cdots + a_{n-k}\vec{b}_{n-k})\| \leq R$
- Father-child relations: $(a_n, \cdots, a_{k+1}) \rightarrow \{(a_n, \cdots, a_{k+1}, a_k)\}$ for all possible a_k

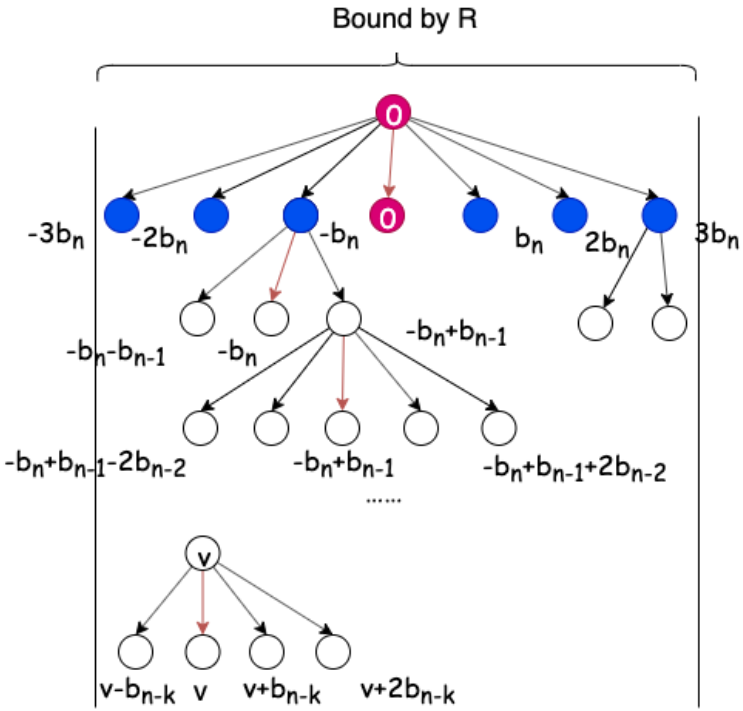


Figure: Searching Tree

Enumeration Algorithm

Cost of enumeration algorithm is measured by total number of nodes in tree, calculated by adding the size of all layers. Full Enumeration Cost (FEC) is the number of nodes when $R = \mathrm{GH}(L)$:

$$N = \mathrm{FEC}(B) := \frac{1}{2} \sum_{k=1}^{n} \frac{V_k(\mathrm{GH}(L))}{\prod_{i=n-k+1}^{n} \|\vec{b}_i^*\|} \tag{3}$$

- $\mathrm{vol}(\pi_{n-k+1}(L)) = \prod_{i=n-k+1}^{n} \|\vec{b}_i\|$
- If $\vec{v}$ is in the tree, $-\vec{v}$ too, therefore it suffices to search only half of the tree

Enumeration Algorithm

Pruning: select seperate bound $(R_1, \cdots, R_n)$ for each layer

- $R_n \leq R_{n-1} \leq \cdots \leq R_1 = R$
- The pruned searching tree may fail to find any $\|\vec{v}\| \leq R$, and has a success probability p which can be estimated given $(R_1, \cdots, R_n)$
- For target probability p, radius $\alpha \cdot \mathrm{GH}(L)$, denote by

$$\mathrm{ENUMCost}(B; \alpha, p) := \min_{R_1, \cdots, R_n} N$$

the minimized enumeration cost over all possible bounding functions. The minimization method is proposed by Chen-Nguyen.

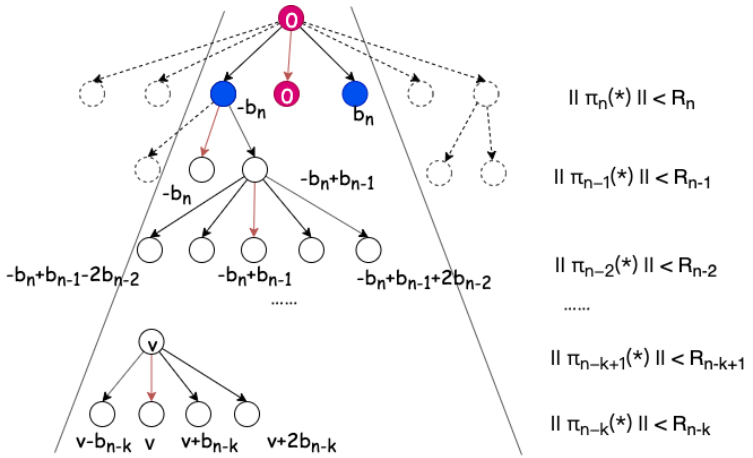
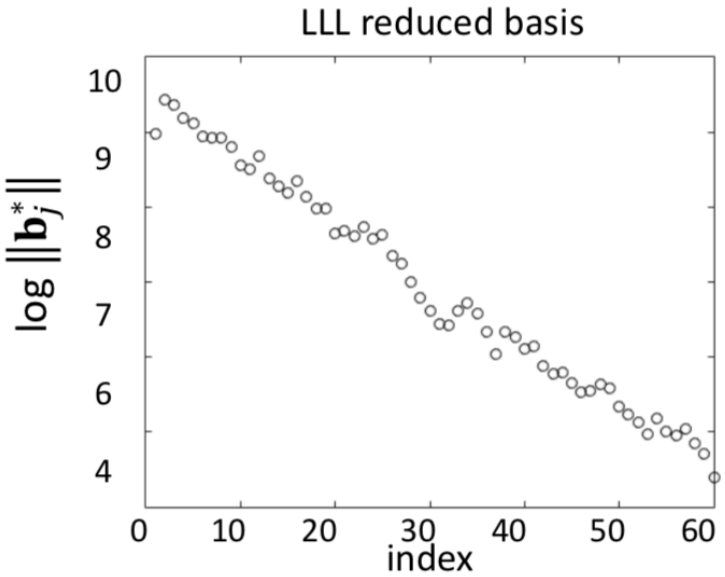


Figure: Pruning Searching Tree

LLL Algorithm

On input basis $B = \{\vec{b}_1, \cdots, \vec{b}_n\}$, output B':

- $\vec{b}'_1$ is short
- $\vec{b}'_1, \cdots, \vec{b}'_n$ are nearly orthogonal



Remark

Lengths of Gram-Schmidt reduced vectors are used to measure the quality of basis:

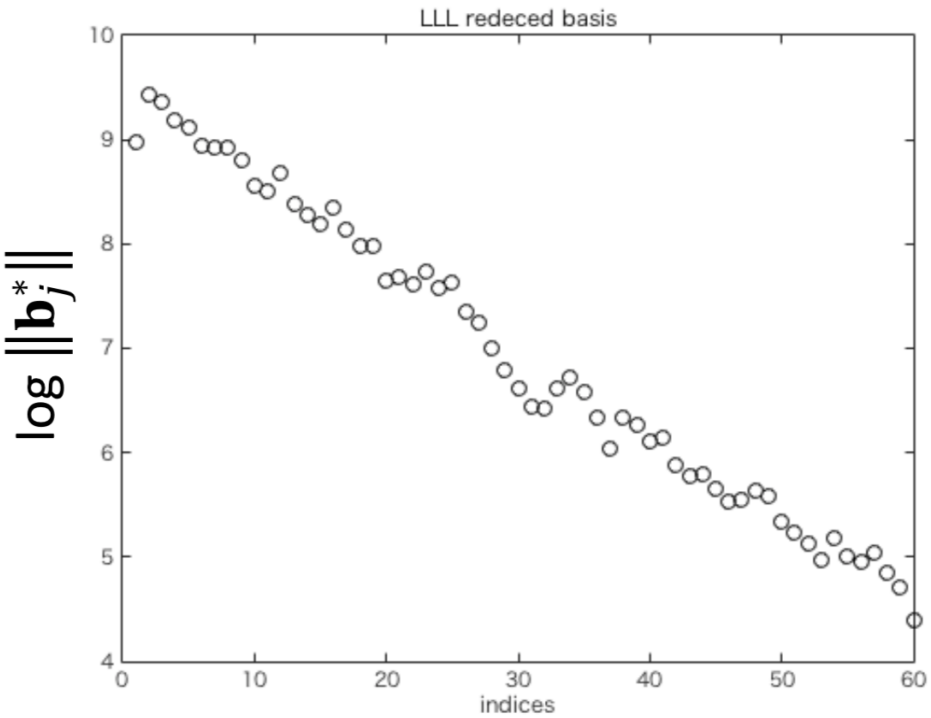
- *The product* $\prod_{i=1}^{n} \|\vec{b}_i^*\|$ *is fixed to* $\mathrm{vol}(L)$*, therefore, the size of under the curve is fixed*
- *The flatter the graph, the better the quality*

Previous BKZ Algorithms

- Basic BKZ Algorithm
- BKZ 2.0
- Progressive BKZ

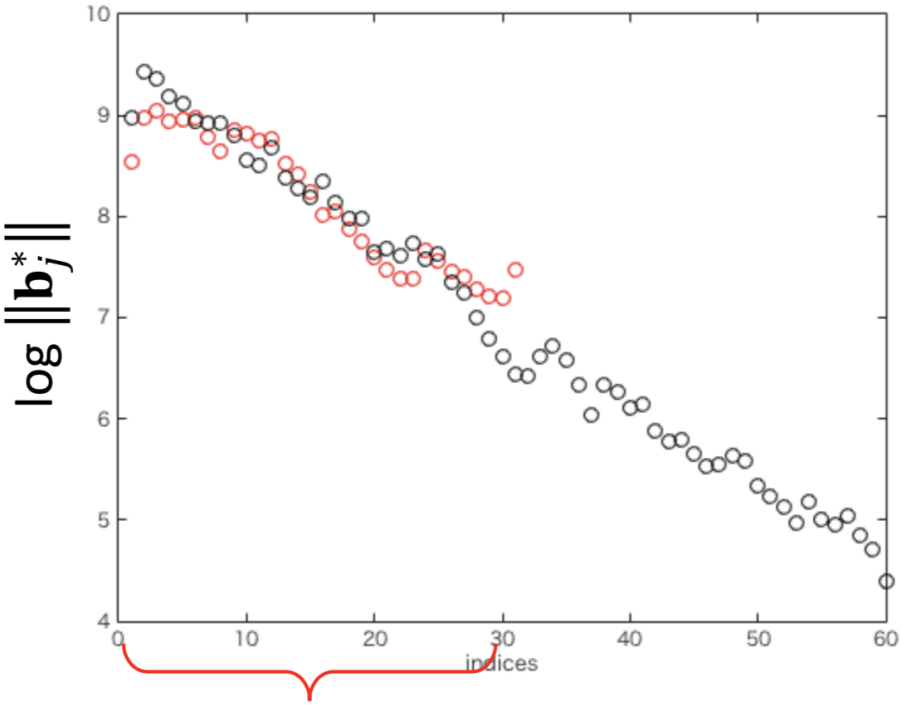
Basic BKZ Algorithm

First apply LLL on the basis.



Basic BKZ Algorithm

- $B_1 = \pi_1(L(\vec{b}_1, \cdots, \vec{b}_\beta))$
- $\vec{v} \leftarrow \mathrm{ENUM}(B_1)$
- If $\|\vec{v}\| \leq \|\vec{b}_1\|$, $\mathrm{LLL}(\vec{v}, \vec{b}_1, \cdots, \vec{b}_\beta)$

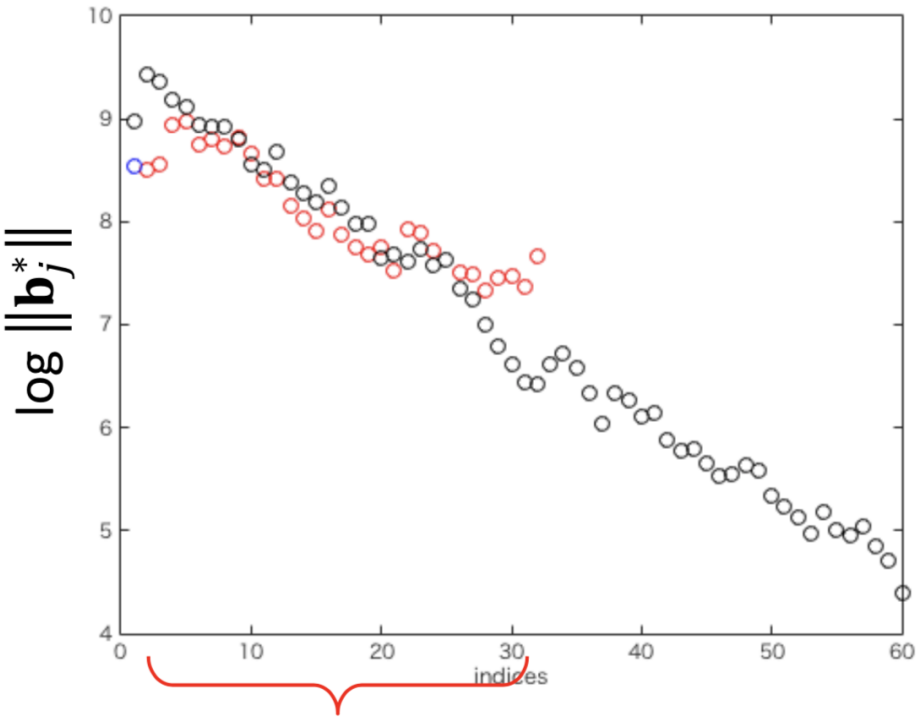


Basic BKZ Algorithm

- $B_2 = \pi_2(L(\vec{b}_2, \cdots, \vec{b}_{\beta+1}))$
- $\pi_2(\vec{v}) \leftarrow \text{ENUM}(B_2)$
- If $\|\vec{v}\| \leq \|\vec{b}_2\|$, $\text{LLL}(\vec{v}, \vec{b}_2, \cdots, \vec{b}_{\beta+1})$

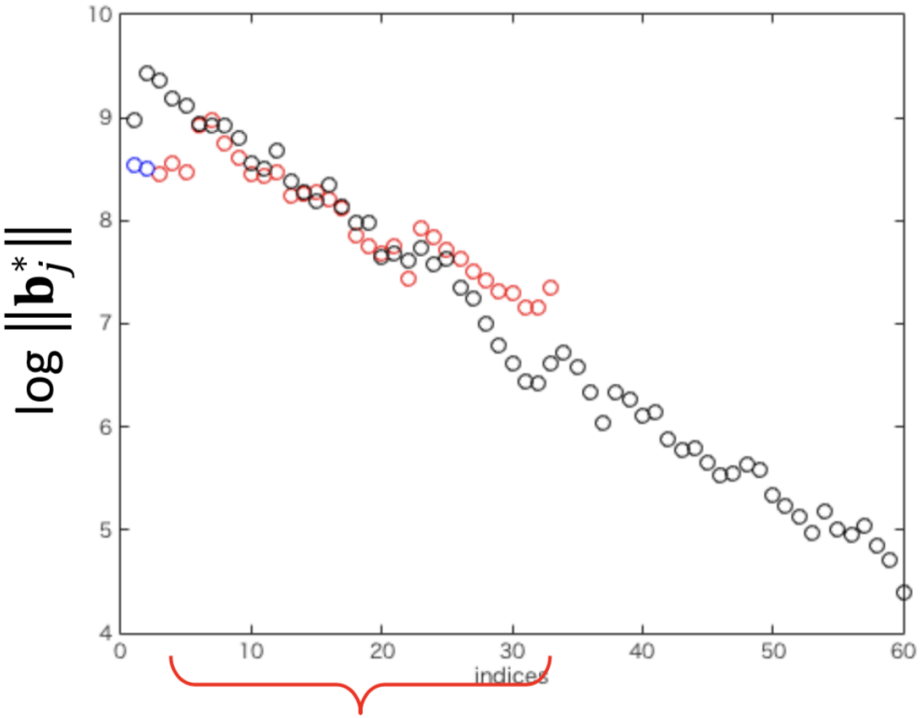
Remark

Assume ENUM found $\pi_2(\vec{v}) = x_2\pi_2(\vec{b}_2) + \cdots + x_{\beta+1}\pi_2(\vec{b}_{\beta+1})$, *then* $\vec{v} = x_2\vec{b}_2 + \cdots + x_{\beta+1}\vec{b}_{\beta+1}$



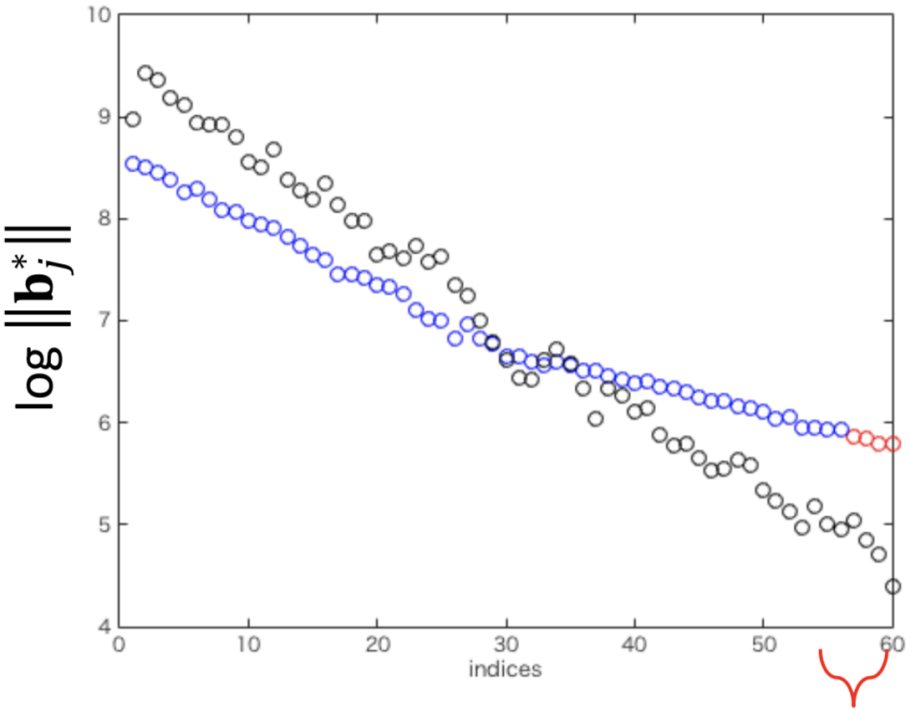
Basic BKZ Algorithm

- $B_3 = \pi_3(L(\vec{b}_3, \cdots, \vec{b}_{\beta+2}))$
- $\pi_3(\vec{v}) \leftarrow \mathrm{ENUM}(B_3)$
- If $\|\vec{v}\| \leq \|\vec{b}_3\|$,
  $\mathrm{LLL}(\vec{v}, \vec{b}_3, \cdots, \vec{b}_{\beta+2})$



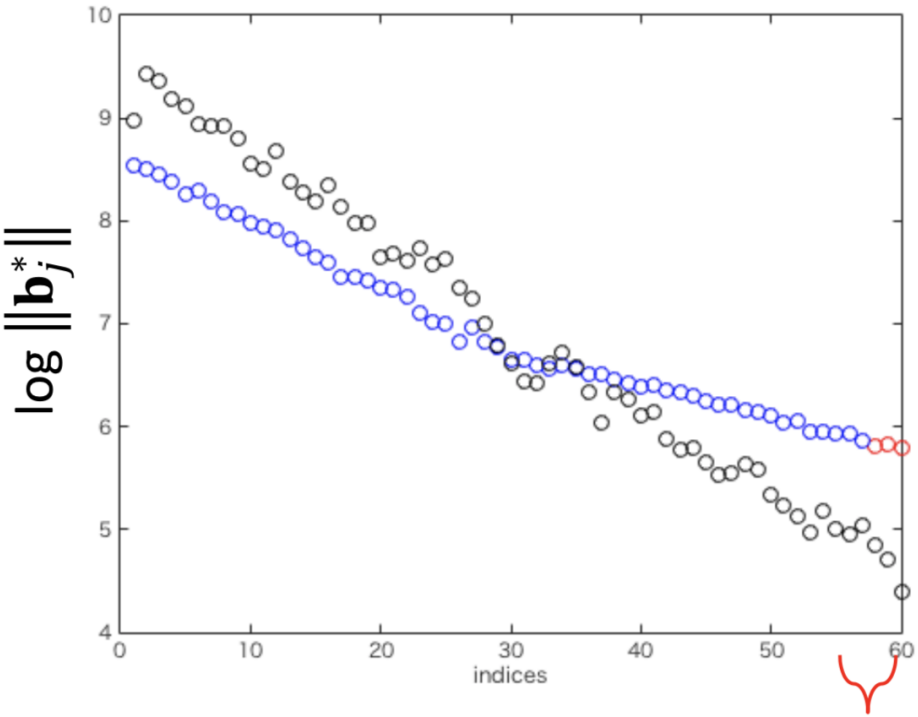
Basic BKZ Algorithm

- $B_{n-3} = \pi_{n-3}(L(\vec{b}_{n-3}, \cdots, \vec{b}_n))$
- $\pi_{n-3}(\vec{v}) \leftarrow \mathrm{ENUM}(B_{n-3})$
- If $\|\vec{v}\| \leq \|\vec{b}_{n-3}\|$, $\mathrm{LLL}(\vec{v}, \vec{b}_{n-3}, \cdots, \vec{b}_n)$



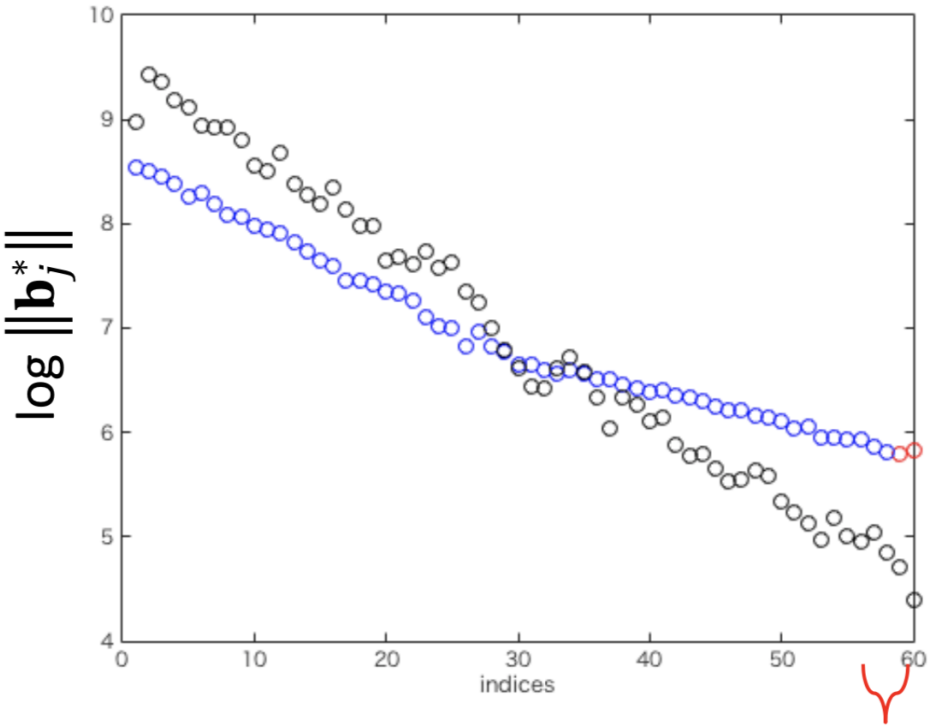
Basic BKZ Algorithm

- $B_{n-2} = \pi_{n-2}(L(\vec{b}_{n-2}, \vec{b}_{n-1}, \vec{b}_n))$
- $\pi_{n-2}(\vec{v}) \leftarrow \mathrm{ENUM}(B_{n-2})$
- If $\|\vec{v}\| \leq \|\vec{b}_{n-2}\|$, $\mathrm{LLL}(\vec{v}, \vec{b}_{n-2}, \vec{b}_{n-1}, \vec{b}_n)$



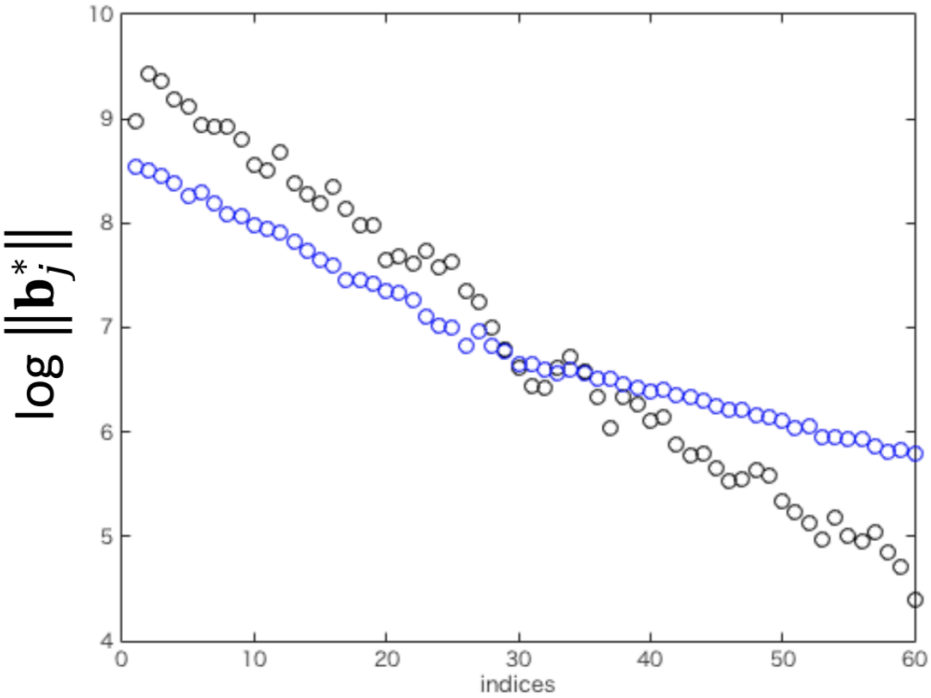
Basic BKZ Algorithm

- $B_{n-1} = \pi_{n-1}(L(\vec{b}_{n-1}, \vec{b}_n))$
- $\pi_{n-1}(\vec{v}) \leftarrow \mathrm{ENUM}(B_{n-1})$
- If $\|\vec{v}\| \leq \|\vec{b}_{n-1}\|$, $\mathrm{LLL}(\vec{v}, \vec{b}_{n-1}, \cdots, \vec{b}_n)$



Basic BKZ Algorithm

Now we complete one ROUND of BKZ.

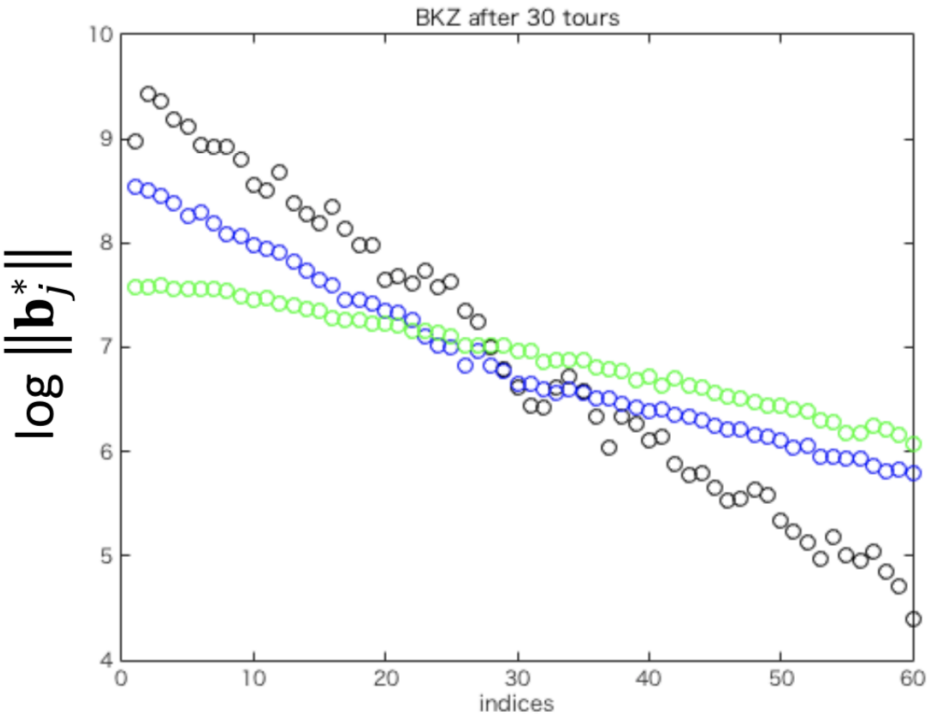


Basic BKZ Algorithm

Repeat until the basis is not updated in some round. Now the basis is called *BKZ-reduced*.

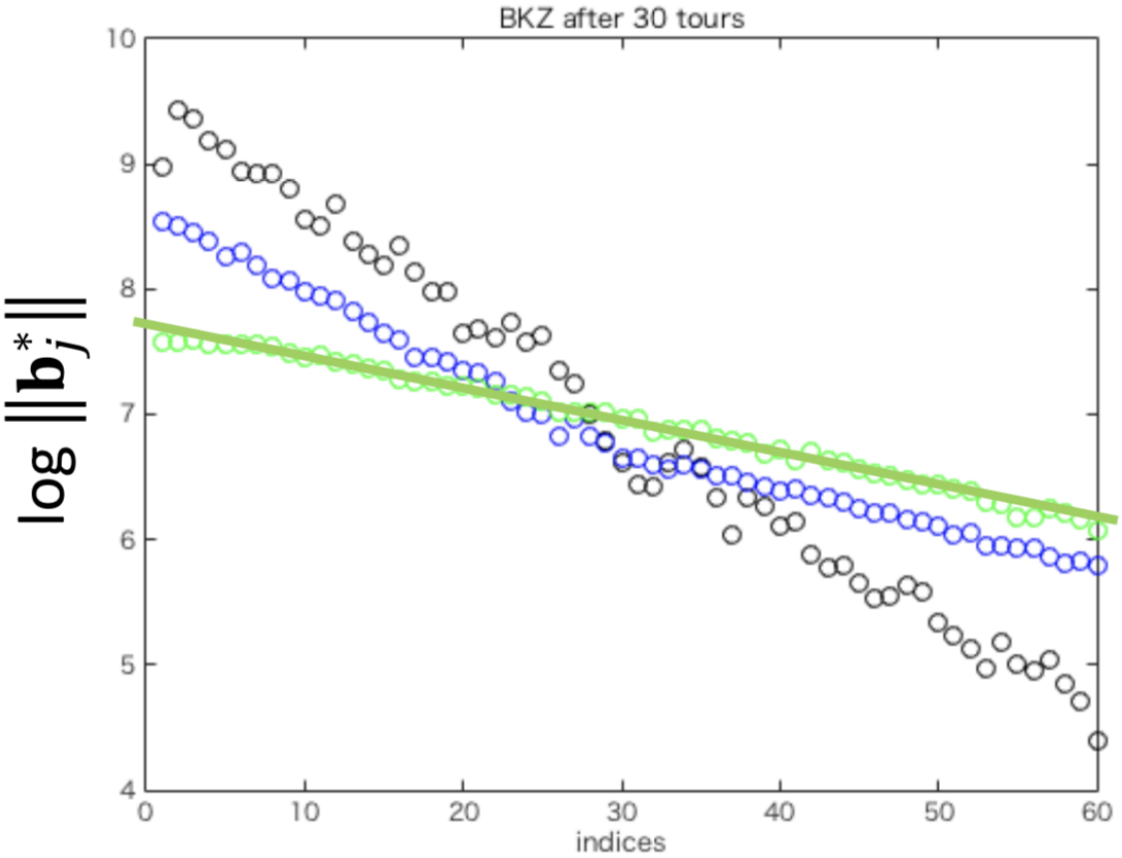
Definition: BKZ-reduced

$B = (\vec{b}_1, \cdots, \vec{b}_n)$ is called BKZ-β-reduced, if for each $i = 1, \cdots, n$, $\vec{b}_i^*$ is the smallest vector in B_i.



Basic BKZ Algorithm

Gram-Schmidt Assumption (GSA): for BKZ-reduced basis B, $\|\vec{b}_i^*\|^2/\|\vec{b}_1\|^2 = r^{i-1}$, where $r \in [3/4, 1)$ is GSA constant.



BKZ 2.0

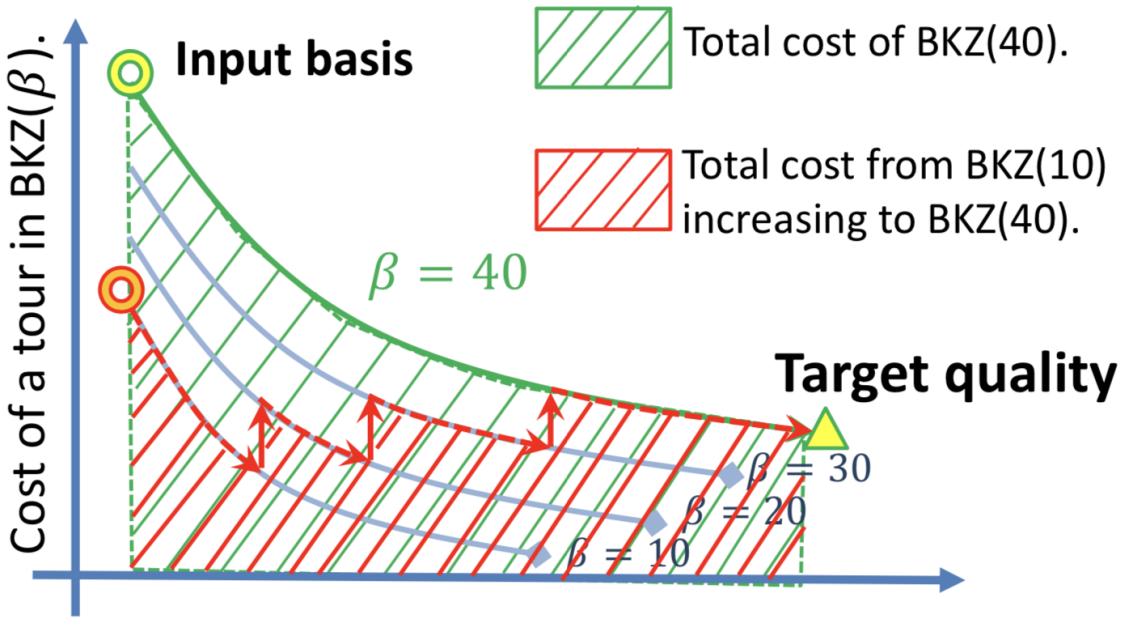
Proposed by Chen-Nguyen in AC 2011.

- Improvements
  - **Extreme pruning** in enumeration: randomize B_i to M different blocks, and apply enumeration to each with extremely low probability $p = 1/M$
  - **Set searching radius** $R = \alpha \cdot \mathrm{GH}(B_i) \quad \alpha = \sqrt{1.1}$ based on observation that usually $\lambda_1(B_i) \leq 1.05\mathrm{GH}(B_i)$
  - **Early termination**, based on observation that the basis quality is already close to the final result long before the algorithm ends
  - **Preprocessing local blocks** using BKZ recursively with smaller blocksizes
- BKZ 2.0 Simulator
  - The computational cost of BKZ is hard to express in close formula, better to be estimated by simulator
  - Execute the BKZ algorithm, modifying simulated GS lengths $\ell_1, \cdots, \ell_n$ instead of the basis B

Progressive BKZ

Observation behind progressive technique:

- Larger blocksize β produces $\vec{v}$ with higher quality
- For fixed blocksize β, the enumeration cost decreases with rounds, because the efficiency increases with quality of basis
- Increasing blocksize to trade efficiency for quality of $\vec{v}$



Improved Progressive BKZ

- Optimizations
- Implementation Detail
- Simulation And Comparison

Optimizations

- Optimizing Parameters
- Sharp Simulator
- Blocksize Strategy

Optimizing Parameters

Parameters inside a BKZ round:

- α: search radius
- β: block size
- p: success probability
- r: GSA constant

For fixed (β, r):

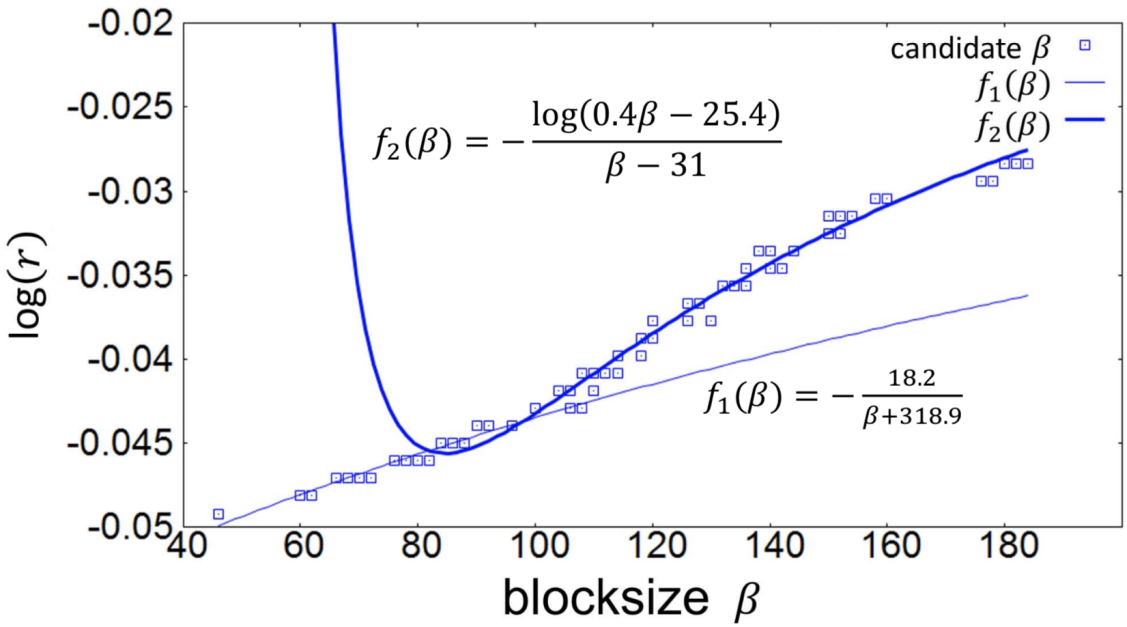
1. Solve for α by (5)
2. Calculate p by (4)
3. Compute $\mathrm{ENUMCost}(B; \alpha, p)$

$$p = \frac{2}{\alpha^{\beta}} \tag{4}$$

$$r = \left(\frac{\beta+1}{\alpha\beta}\right)^{\frac{4}{\beta-1}} \cdot V_{\beta}(1)^{\frac{4}{\beta(\beta-1)}} \tag{5}$$

Optimizing Parameters

Given blocksize β, find r that minimizes $\mathrm{ENUMCost}(B; \alpha, p)$.



$$\log(r) = \begin{cases} f_1(\beta) = \dfrac{-18.2139}{\beta + 318.978} & \beta \leq 100 \\ f_2(\beta) = \dfrac{1.06889}{\beta - 31.0345} \cdot \log(0.417419\beta - 25.4889) & \beta > 100 \end{cases} \quad (6$$

Sharp Simulator

Full Enumeration Cost:

$$\mathrm{FEC}(B) := \frac{1}{2}\sum_{k=1}^{n} \frac{V_k(\mathrm{GH}(L))}{\prod_{i=n-k+1}^{n} \|\vec{b}_i^*\|} \tag{7}$$

Simulated Enumeration Cost:

$$\mathrm{Sim\text{-}FEC}(\ell_1, \cdots, \ell_n) := \sum_{k=1}^{n} \frac{V_k(\mathrm{Sim\text{-}GH}(\ell_1, \cdots, \ell_n))}{\prod_{i=n-k+1}^{n} \ell_i} \tag{8}$$

$$\mathrm{Sim\text{-}GH}(\ell_1, \cdots, \ell_n) := \sum_{k=1}^{n} V_n(1)^{1/n} \prod_{j=1}^{n} \ell_j^{1/n} \tag{9}$$

$$\ell_1, \cdots, \ell_n := \mathrm{Sim\text{-}GS\text{-}Lengths}(n, \beta) \tag{10}$$

Sharp Simulator

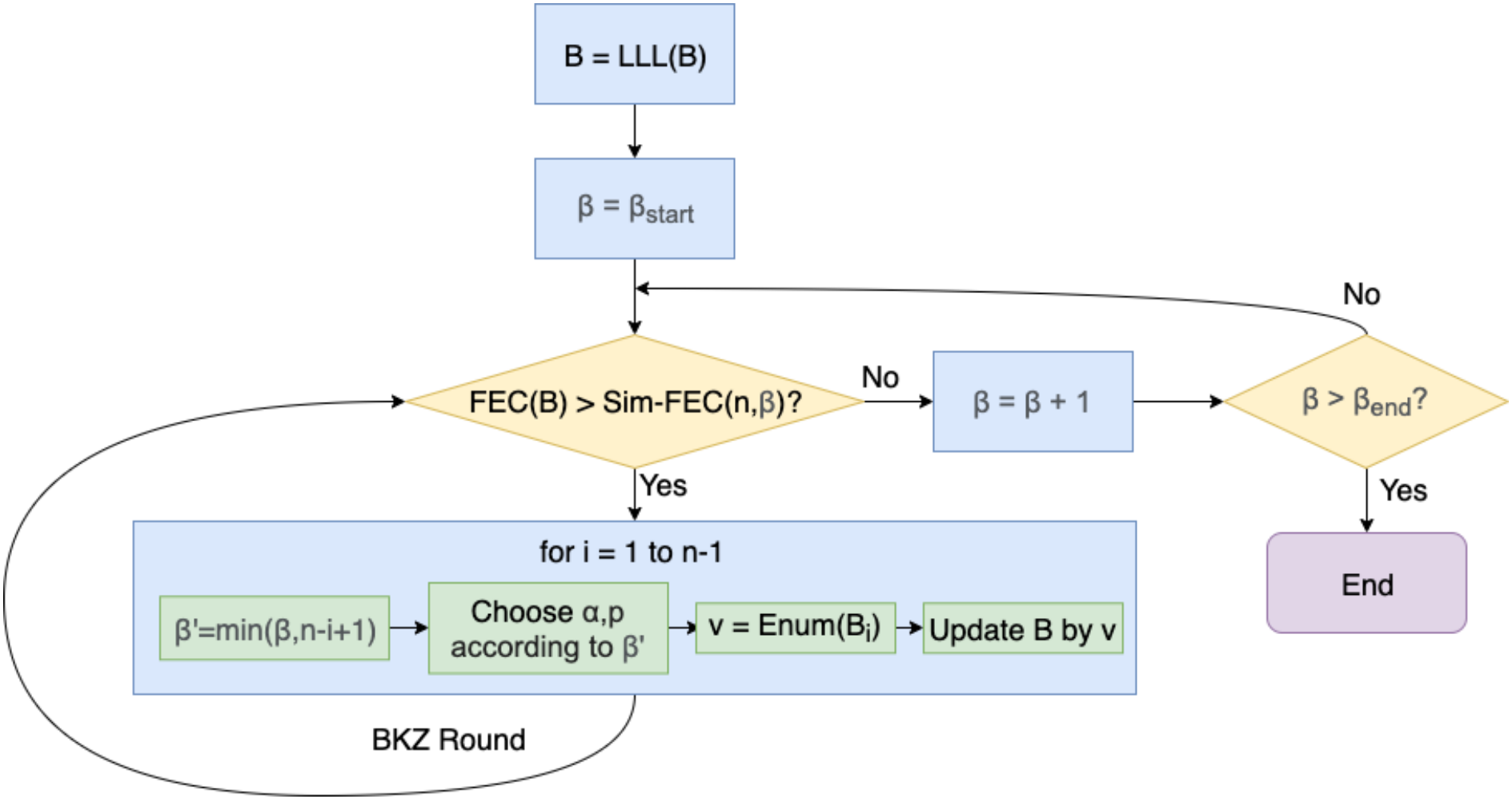


Figure: Basic Variant

Blocksize Strategy

A blocksize strategy is a list $\{(\beta_i^{alg}, \beta_i^{goal})\}_{i=1}^{D}$

- Call the basis β-reduced if $\mathrm{FEC}(B) \leq \text{Sim-FEC}(n, \beta)$
- At the start of strategy i, the basis is β_{i-1}^{goal}-reduced (except for $i = 1$ where B is LLL-reduced)
- Apply BKZ rounds to B with blocksize β^{alg}, until the basis is β_i^{goal}-reduced
- Move to the next strategy, i.e. $i = i + 1$

The basic variant uses the simple strategy:

$$\text{LLL-reduced} \xrightarrow{\beta^{start}} \beta^{start} + 1 \xrightarrow{\beta^{start}+1} \cdots \beta^{end} - 1 \xrightarrow{\beta^{end}-1} \beta^{end}$$

Blocksize Strategy

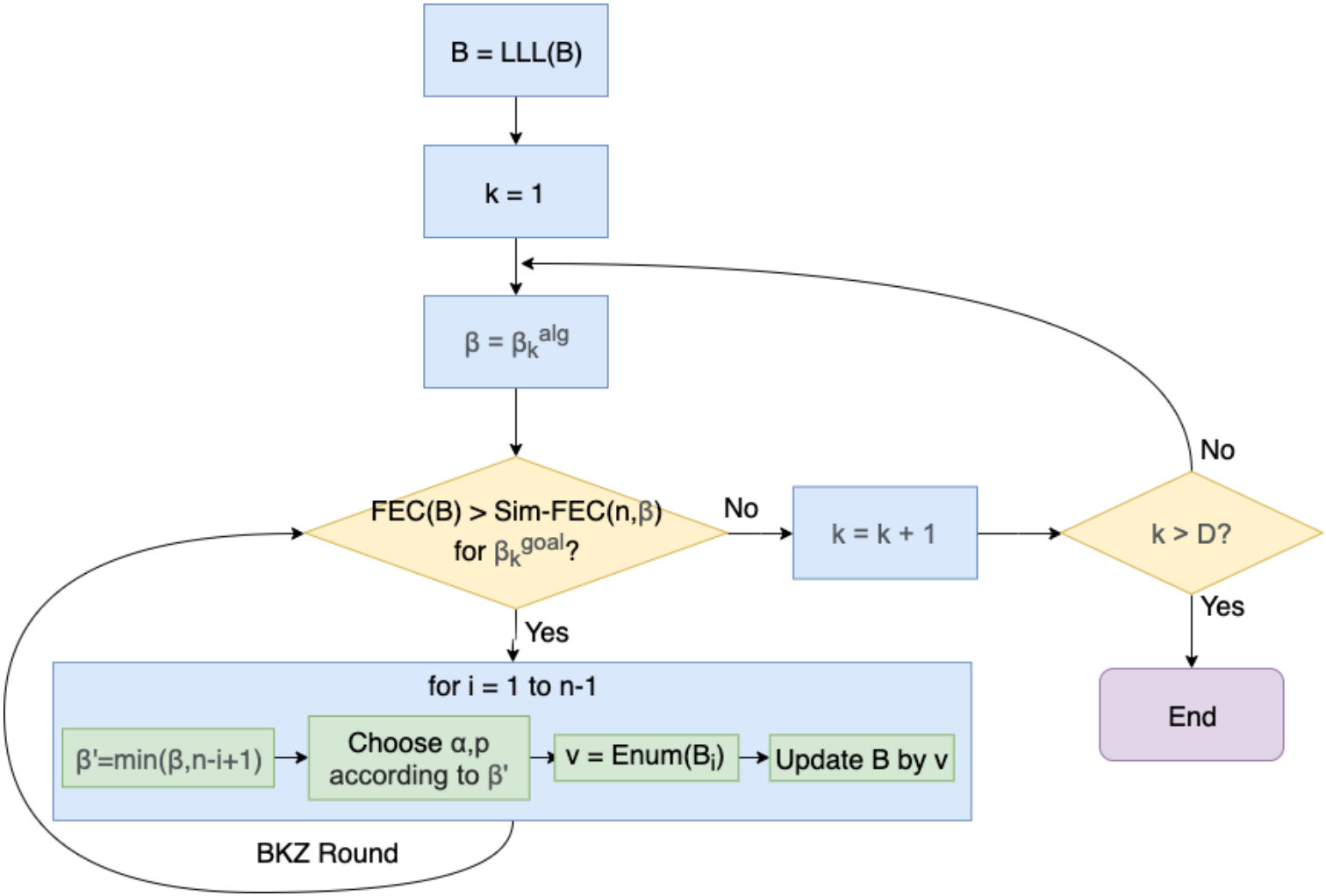


Figure: Progressive BKZ with Blocksize Strategy

Implementation Detail

Preprocessing

1. Preprocess the local block B_i using this improved progressive BKZ algorithm, with much smaller blocksizes.
2. The blocksize strategy takes the simple one-by-one strategy starting from 15.

The cost of this step: a constant $A_{Preprocess}$ times the main enumeration cost.

Implementation Detail

Enumeration

- Outputs $h = 16$ vectors $(\vec{v}_1, \cdots, \vec{v}_h)$ with small projective lengths, instead of only one $\vec{v}$ in original BKZ
- The cost is

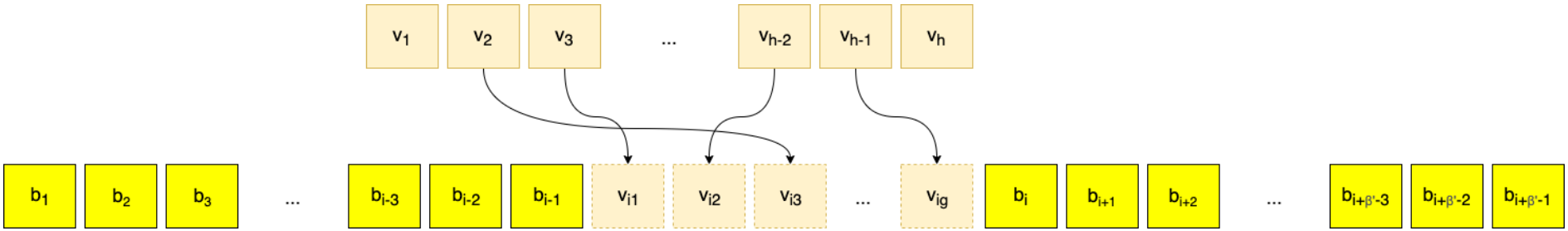
$$A_{Enum} \cdot \beta \cdot \mathrm{ENUMCost}(B_i; \alpha, p)$$

where A_{Enum} is a constant

Implementation Detail

Construct degenerated basis by inserting some of $(\vec{v}_1, \cdots, \vec{v}_h)$ into the basis

- Select g elements $(\vec{v}_{i_1}, \cdots, \vec{v}_{i_g})$ from $(\vec{v}_1, \cdots, \vec{v}_h)$ and insert into the i'th position of $(\vec{b}_1, \cdots, \vec{b}_{i+\beta'-1})$ as follows
- Each $\vec{v}_{i_j}$ is selected to minimize $\pi'(\vec{v}_{i_j})$, and require that $\|\pi'(\vec{v}_{i_j})\| < \|\vec{v}_{i_{j-1}}\|$, where π' maps $\vec{v}$ to its projection on $\mathrm{span}(\vec{b}_1, \cdots, \vec{b}_{i-1}, \vec{v}_{i_1}, \cdots, \vec{v}_{i_{j-1}})$
- Stop if none of the left $\vec{v}_i$'s satisfies the requirement



Implementation Detail

Apply LLL:

- The first $i-1$ vectors in the degenerated basis is already LLL-reduced
- The cost is $A_1 \cdot \beta^2 n^2$, where A_1 is constant

Implementation Detail

Total cost estimation:

$$\begin{aligned}&\mathrm{Time}(n, \beta, A_1, W_1)\\ &= \sum_{\beta^{start}}^{\beta^{goal}} \sum_{t=1}^{\#rounds} \left[A_1 \cdot \beta^2 n^3 + W_1 \cdot \beta \sum_{i=1}^{n-1} \mathrm{ENUMCost}(B_i; \alpha, p) \right]\end{aligned}$$

where A_1 and W_1 are constants that are determined by experiments.

- The coefficients are fitted by standard curve fitting method in semi-log scale.
- Fitting results: $A_1 = 1.5 \cdot 10^{-10}$ and $W_1 = 1.5 \cdot 10^{-8}$.

Pre/Post-Processing

A complete attack on the SVP consists of two steps:

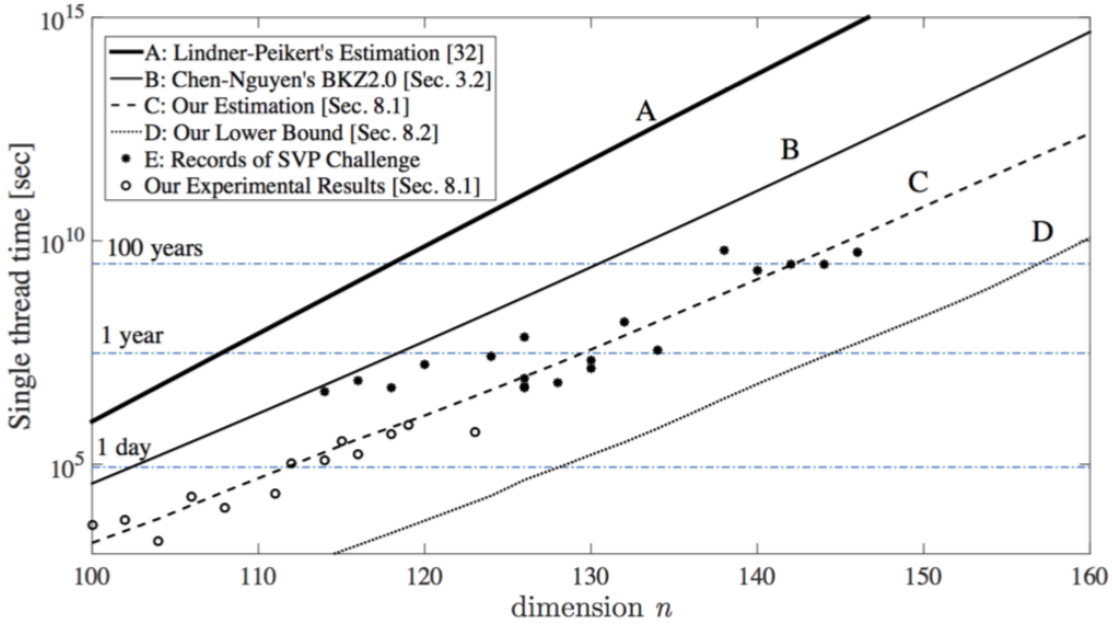
1. Apply BKZ reduction to the basis
2. Use enumeration to find a short vector

The Preprocessing and Postprocessing works similar to extreme pruning:

- **Preprocessing**: Randomize B_i to M different basis
- **BKZ**: Apply the improved progressive BKZ to all the B_i's
- **Postprocessing**: Apply enumeration with probability $p = 2 \cdot \alpha^{-n}/M$

Simulation Results for SVP Challenges

This is the comparison result of **simulated** results, where Lindner-Peikert's estimation is the cost estimation of the basic BKZ.



Real Results

Position	Dimension	Index	Seed	Euclidean norm	Contestant	Solution	Using Ideal Structure	Subm. Date
1	652	653	0	626850	Yuntao Wang; Yoshinori Aono; Takuya Hayashi; Jintai Ding; Tsuyoshi Takagi	vec	yes	2016-03-17
2	652	653	0	626936	Yuntao Wang; Yoshinori Aono; Takuya Hayashi; Tsuyoshi Takagi	vec	no	2016-03-17
3	652	653	0	661210	Jean-Christophe Deneuville	vec	yes	2015-05-27
4	652	653	0	661349	Yuntao Wang, Yoshinori Aono, Takuya Hayashi, Tsuyoshi Takagi	vec	no	2015-05-26
5	600	601	0	542883	Jean-Christophe Deneuville	vec	yes	2015-05-13

Figure: Ideal Lattice Challenge

2016-10-24	70	0.005	Yuntao Wang;Yoshinori Aono; Takuya Hayashi; Tsuyoshi Takagi	1731.68	Details
2016-08-09	40	0.020	Rui Xu; Kazuhide Fukushima; Shinsaku Kiyomoto; Tsuyoshi Takagi	1302.90	Details
2016-08-02	65	0.005	Yuntao Wang; Yoshinori Aono; Takuya Hayashi; Tsuyoshi Takagi	1373.69	Details
2016-07-28	50	0.010	Yuntao Wang; Yoshinori Aono; Takuya Hayashi; Tsuyoshi Takagi	1296.82	Details
2016-07-01	60	0.005	Rui Xu Kazuhide Fukushima Shinsaku Kiyomoto Tsuyoshi Takagi	1089.84	Details

Figure: LWE Challenge

Q & A