Computer Modeling in Engineering & Sciences

Tech Science Press

https://doi.org/10.32604/cmes.2026.083407

ARTICLE

Improving ENUM-Sieve Reduction Algorithm for Prime Cyclotomic Lattices

Kazutaka Toda[1], **Yuntao Wang**[1,*], **Hyungrok Jo**[2] **and Yang Li**[1]

[1]Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo, Japan
[2]Institute of Advanced Sciences, Yokohama National University, Yokohama, Kanagawa, Japan

*Corresponding Author: Yuntao Wang. Email: y-wang@uec.ac.jp

Received: 03 April 2026; Accepted: 22 June 2026; Published: 27 July 2026

ABSTRACT: The rapid evolution of quantum computing poses a fundamental challenge to classical public-key cryptosystems, accelerating the adoption of lattice-based post-quantum cryptography in large-scale digital infrastructures, including Future Mobile Internet Technologies (FMIT) and their convergence applications (FMIT-CA). As lattice-based cryptography is expected to play an important role in such environments, accurate hardness estimation and parameter assessment of underlying lattice problems have become increasingly important. Since the security of these cryptographic schemes is closely related to the computational hardness of the Shortest Vector Problem (SVP), improving practical SVP-solving techniques contributes indirectly to the security evaluation of such systems. Among practical SVP solvers, sieve-based approaches such as the General Sieve Kernel (G6K) achieve state-of-the-art performance, yet their exponential complexity and resource demands constrain scalability in high-dimensional settings. In this work, we propose an improved version of the hybrid algorithm ENUM-Sieve Reduction (ESR) proposed by Toda et al. in ProvSec 2025. We refer to our proposal as ENUM-Sieve Reduction 2.0 (ESR 2.0). It integrates Block Korkine-Zolotarev 2.0 (BKZ 2.0) and extreme-pruning enumeration into the reduction pipeline and introduces a unimodular-matrix-based strategy for partial basis generation. Experimental results suggest that these enhancements enable stronger parameter configurations and more efficient execution in higher dimensions under realistic computational constraints. Experimental evaluations on prime cyclotomic ideal lattices demonstrate the practical usefulness of ESR 2.0 produces vectors equal to or shorter than those obtained by G6K in 75% of the tested instances for dimensions ranging from 96 to 130. Compared with ESR, ESR 2.0 achieves the same or shorter vectors in 62.5% of the tested instances. Although ESR 2.0 requires longer CPU time due to additional enumeration steps, GPU time and peak memory usage remain comparable to those of G6K and ESR. Furthermore, ESR 2.0 renewed record norms in the TU Darmstadt Ideal Lattice Challenge for dimensions 112, 126, 136, 148 and 156. These results indicate the usability of ESR 2.0, providing a competitive and practical framework for high-dimensional SVP solving. The proposed improvements contribute to a more accurate assessment of lattice hardness, which is essential for secure parameter selection in post-quantum cryptographic systems supporting future mobile and converged digital environments.

KEYWORDS: Post-quantum cryptography; ideal lattice; shortest vector problem; sieve algorithms; ENUM-sieve reduction

1 Introduction

Rapid advances in digital and communication technologies have transformed modern society, enabling large-scale data exchange, cloud services, and mobile computing infrastructures. In emerging environments such as 5G/6G networks, Internet of Things (IoT) ecosystems, and future mobile internet technologies (FMIT), secure and efficient cryptographic mechanisms are indispensable. As digital services increasingly

converge across heterogeneous platforms, ensuring robust information security under resource-constrained and high-throughput conditions has become a fundamental requirement. In particular, large-scale distributed services envisioned in FMIT environments require cryptographic techniques that can provide strong security guarantees while maintaining practical computational performance.

At the same time, the rapid progress of quantum computing poses a serious threat to classical public-key cryptosystems such as RSA (Rivest-Shamir-Adleman 1977) and elliptic curve cryptography. These cryptosystems rely on the computational hardness of the integer factorization problem and the discrete logarithm problem, both of which can be solved in polynomial time by Shor's quantum algorithm [1] on a sufficiently large quantum computer. As a consequence, the development of quantum-resistant cryptographic schemes, known as Post-Quantum Cryptography (PQC), has become a common research topic. To facilitate the transition to PQC, the National Institute of Standards and Technology (NIST) has been conducting a long-term standardization project. In the NIST post-quantum cryptography standardization process, several lattice-based schemes have been selected for standardization or further evaluation, highlighting the central role of lattice-based cryptography in the post-quantum era.

Lattice-based cryptography is regarded as one of the most promising approaches to PQC due to its strong theoretical foundations and versatility in constructing advanced cryptographic primitives. Its security relies on the computational hardness of several lattice problems, among which the Shortest Vector Problem (SVP) is one of the most fundamental. Given a lattice, the SVP asks for the shortest non-zero lattice vector. The difficulty of solving SVP in high dimensions forms the security basis of many lattice-based cryptographic constructions. Therefore, accurately evaluating the hardness of SVP is essential for selecting secure parameters in practical cryptographic systems. As lattice-based cryptography is expected to be deployed in large-scale digital infrastructures, including FMIT-related environments, accurate hardness estimation and parameter assessment of underlying lattice problems have become increasingly important. Therefore, improving SVP-solving techniques contributes indirectly to the security evaluation and parameter selection of lattice-based cryptographic systems intended for such environments.

Algorithms for solving SVP can be broadly classified into three main categories: basis reduction algorithms, enumeration algorithms, and sieve algorithms. Basis reduction algorithms aim to transform a given lattice basis into a reduced basis consisting of relatively short and nearly orthogonal vectors. One of the most well-known examples is the Lenstra–Lenstra–Lovász (LLL) algorithm [2], which can compute an approximate shortest vector in polynomial time. Although the quality of the resulting vector is limited, LLL is widely used as a preprocessing step in many lattice algorithms.

A more powerful basis reduction method is the Block Korkine-Zolotarev (BKZ) algorithm [3], which improves upon LLL by applying lattice reduction within small blocks of the basis. BKZ repeatedly performs LLL reduction and solves smaller SVP instances inside a block of dimension β, leading to significantly higher-quality reduced bases. Later improvements such as Block Korkine-Zolotarev 2.0 (BKZ 2.0) [4] incorporate advanced techniques including extreme pruning enumeration [5] to further enhance practical efficiency. Further improved reduction algorithms can be found in [6–8].

Enumeration (ENUM) algorithms provide another important approach to solving SVP. The Schnorr–Euchner enumeration algorithm [9] systematically searches lattice vectors in a tree structure and can find the exact shortest vector within a given search radius. While enumeration can produce exact solutions, its time complexity grows exponentially with the lattice dimension. Various optimization techniques, including pruning strategies, have been proposed to reduce the practical search space and improve efficiency.

Among these approaches, sieve algorithms currently achieve the best practical performance for solving SVP in high dimensions. The first sieve algorithm was proposed by Ajtai et al. [10], which demonstrated that

random sampling techniques could be used to solve SVP. Later, Nguyen and Vidick introduced a heuristic sieve that significantly improved practical efficiency [11]. Micciancio and Voulgaris subsequently proposed the Gauss Sieve [12], which further refined the reduction process. Recent work has focused on improving nearest-neighbor search techniques and progressive sieving strategies. In particular, the General Sieve Kernel (G6K) [13] integrates progressive lattice sieving [14] and dimension-for-free techniques [15], achieving state-of-the-art performance in the international SVP Challenge [16].

In addition to general lattices, many cryptographic constructions rely on structured lattices such as ideal lattices. These lattices possess algebraic properties that enable efficient arithmetic operations and compact representations. Consequently, specialized algorithms that exploit the structure of ideal lattices have been actively studied. For instance, Schneider proposed a sieving method for ideal lattices that leverages rotational symmetries to accelerate the search process [17]. Later, Narisada et al. introduced a faster rotation-based Gauss sieve that further improves efficiency when solving SVP on general ideal lattices [18]. These studies demonstrate that exploiting algebraic structures can significantly influence the practical performance of lattice algorithms.

To address these challenges, Toda and Wang proposed ENUM-Sieve Reduction (ESR), a hybrid framework combining enumeration and sieve techniques while exploiting the algebraic structure of ideal lattices [19]. ESR demonstrated that coordinated CPU-GPU execution and structured sub-basis processing can improve practical performance. However, further enhancements were necessary to support stronger parameter settings and larger lattice dimensions.

In this paper, we present ENUM-Sieve Reduction 2.0 (ESR 2.0), which significantly extends the original ESR algorithm in [19]. ESR 2.0 integrates BKZ 2.0 and extreme pruning enumeration into the reduction pipeline and redesigns the partial basis generation strategy using unimodular matrix transformations. Our experiments indicate that these refinements improve the scalability of the algorithm to operate effectively under stronger parameter configurations and to scale to higher-dimensional lattices within realistic computational constraints.

1.1 Contributions

The main contributions of this work are summarized as follows.

1. We redesign the ESR framework by incorporating BKZ 2.0 and extreme pruning enumeration, thereby strengthening the reduction quality of intermediate bases.
2. We introduce a unimodular-matrix-based partial basis generation strategy that improves sub-basis diversity while preserving lattice structure.
3. We demonstrate through extensive experiments that ESR 2.0 produces vectors equal to or shorter than those of G6K in 75% of evaluated instances, while maintaining comparable GPU time and memory usage.
4. We demonstrate through extensive experiments that ESR 2.0 produces vectors equal to or shorter than those of ESR in 62.5% of evaluated instances.
5. We provide new benchmark results on the Ideal Lattice Challenge, establishing record norms for multiple instances up to dimension 156. This improvement is enabled by ESR 2.0, which can handle stronger parameter settings and larger sub-basis dimensions than ESR.

These contributions advance the practical understanding of SVP hardness in structured lattices and provide insights relevant to parameter evaluation for lattice-based cryptographic systems in future mobile internet environments.

2 Preliminaries

2.1 Lattice

Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a collection of n vectors in the vector space $\mathbb{R}^m$. The set consisting of all integer linear combinations of these vectors is defined as

$$\mathrm{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) := \left\{ \sum_{i=1}^{n} a_i \mathbf{b}_i \in \mathbb{R}^m : a_i \in \mathbb{Z} \right\}.$$

When $\mathbb{R}^m$ is regarded as an additive group, the set $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ generated by $\mathbf{b}_1, \ldots, \mathbf{b}_n$ forms a subgroup of $\mathbb{R}^m$.

If the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ are linearly independent, the set $\mathcal{L} = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called a lattice in $\mathbb{R}^m$. The elements of $\mathcal{L}$ are referred to as lattice points or lattice vectors. The ordered tuple $\mathbf{b}_1, \ldots, \mathbf{b}_n$ of linearly independent vectors generating $\mathcal{L}$ is called a basis, or more specifically, a lattice basis. Each individual vector $\mathbf{b}_i$ in this tuple is called a basis vector. The integer n is defined as the dimension of the lattice and is denoted by $\dim(\mathcal{L})$. In the special case where $m = n$, the lattice is said to be full-rank.

Each basis vector $\mathbf{b}_i = (b_{i1}, \ldots, b_{im})$ can be interpreted as a row of an $n \times n$ matrix:

$$\mathbf{B} = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{pmatrix} = \begin{pmatrix} b_{11} & \ldots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{n1} & \ldots & b_{nm} \end{pmatrix}$$

This matrix is referred to as the basis matrix, or lattice basis matrix, associated with $\mathcal{L}$. The lattice generated by the row vectors of $\mathbf{B}$ is denoted by $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$.

2.2 Successive Minimum

For an n-dimensional lattice $\mathcal{L}$, the i-th successive minimum $\lambda_i(\mathcal{L})$ is defined by

$$\lambda_i(\mathcal{L}) := \min_{\{\mathbf{b}_1, \ldots, \mathbf{b}_i \in \mathcal{L}\}} \max\{\|\mathbf{b}_1\|, \ldots, \|\mathbf{b}_i\|\}$$

for each $1 \le i \le n$. This quantity represents the smallest radius such that $\mathcal{L}$ contains i linearly independent vectors whose norms do not exceed that radius.

2.3 Shortest Vector Problem (SVP)

Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be a basis of an n-dimensional integer lattice $\mathcal{L} \subseteq \mathbb{Z}^n$. The Shortest Vector Problem (SVP) consists of finding a non-zero lattice vector $\mathbf{v} \in \mathcal{L}$ satisfying

$$\|\mathbf{v}\| = \lambda_1(\mathcal{L}).$$

In other words, SVP seeks a shortest non-zero vector in the lattice.

2.4 Approximate Shortest Vector Problem (Approximate SVP)

Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be a basis of an n-dimensional integer lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, and let $\gamma(n) \ge 1$ be an approximation factor. The Approximate Shortest Vector Problem asks for a non-zero vector $\mathbf{v} \in \mathcal{L}$ such that

$$\|\mathbf{v}\| \le \gamma(n)\lambda_1(\mathcal{L}).$$

Thus, instead of requiring an exact shortest vector, the problem permits solutions within a multiplicative factor $\gamma(n)$ of optimality.

2.5 Gaussian Heuristic

For a full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$, the number of lattice points contained in the intersection $\mathcal{L} \cap C$, where $C \subseteq \mathbb{R}^n$ is any measurable set, can be heuristically estimated by $\mathrm{vol}(C)/\mathrm{vol}(\mathcal{L})$. This approximation is referred to as the Gaussian heuristic. In particular, if C is chosen to be the open n-dimensional ball $\mathrm{B}(\mathbf{0}, \lambda_1(\mathcal{L}))$ with radius $\lambda_1(\mathcal{L})$, one expects

$$\frac{\mathrm{vol}(C)}{\mathrm{vol}(\mathcal{L})} \approx \#(\mathcal{L} \cap C) \approx 1.$$

Using the identity $\mathrm{vol}(C) = v_n \lambda_1(\mathcal{L})^n$, we obtain the approximation

$$\lambda_1(\mathcal{L}) \approx \left(\frac{\mathrm{vol}(\mathcal{L})}{v_n}\right)^{\frac{1}{n}} \sim \sqrt{\frac{n}{2\pi e}}\mathrm{vol}(\mathcal{L})^{\frac{1}{n}}.$$

This expression provides a useful heuristic estimate for the length of the shortest lattice vector in high dimensions. In this paper, $\mathrm{GH}(\mathbf{B})$ denotes the Gaussian Heuristic estimate of the shortest vector length in the lattice generated by the basis $\mathbf{B}$.

2.6 Ideal Lattices

Let $f = x^n + f_n x^{n-1} + \cdots + f_1 \in \mathbb{Z}[x]$ be a monic polynomial of degree n, and consider the quotient ring $\mathbf{R} = \mathbb{Z}[x]\langle f(x)\rangle$. The elements of $\mathbf{R}$ are polynomials whose degree is at most $n-1$. For an ideal $\mathbf{I} \subseteq \mathbf{R}$, every element admits a unique representation through its coefficient vector in $\mathbb{Z}^n$ as follows:

$$v = \sum_{i=1}^{n} v_i x^{i-1} \in I \longmapsto \mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^n.$$

Since an ideal is an additive submodule, the collection of coefficient vectors corresponding to elements of $\mathbf{I}$ forms a lattice. Such a lattice is referred to as an ideal lattice. For any $\mathbf{v} \in \mathbf{R}$, the set $\{x^i \cdot \mathbf{v}(i \in [n])\}$ constitutes a basis of the associated ideal lattice.

Multiplication by x induces a cyclic shift of the coefficients of $\mathbf{v}$, and this transformation is called a rotation. For a suitable nonzero generator $\mathbf{v}$ of an ideal $\mathbf{I}$, the coefficient vectors of $\mathbf{v}, x \cdot \mathbf{v}, \ldots, x^{n-1} \cdot \mathbf{v}$ generate the corresponding ideal lattice. This rotation can be realized by multiplication with the matrix:

$$\mathrm{rot} = \left(\begin{array}{cc} \mathbf{0}_{n-1} & \mathbf{I}_{n-1} \\ \hline -f & \end{array}\right)$$

where $\mathbf{0}_{n-1}$ denotes a zero column vector of length $n-1$. We write $\mathrm{rot}(\mathbf{v})$ to denote the rotated vector, and the rotation satisfies the following properties:

- $\mathrm{rot}(\mathrm{rot}(\mathbf{v})) = \mathrm{rot}^2(\mathbf{v})$
- $\mathrm{rot}(\mathrm{rot}^{-1}(\mathbf{v})) = \mathrm{rot}^{-1}(\mathrm{rot}(\mathbf{v})) = \mathbf{v}$

If f is irreducible, then for any $\mathbf{v} \in \mathbf{R}$, the set $\{\mathbf{v}, x \cdot \mathbf{v}, \ldots, x^{n-1} \cdot \mathbf{v}\}$ is linearly independent. Consequently, the set $x^i \cdot \mathbf{v}(i \in [n])$ indeed forms a basis of the ideal lattice. In particular, when $f = x^n + x^{n-1} + \cdots + 1$ and $n+1$ is a prime number, the resulting ideal lattice is known as a prime cyclotomic lattice, namely an ideal lattice defined over a prime cyclotomic polynomial ring.

Fig. 1 shows a two-dimensional visualization of a prime cyclotomic lattice. In this example, we use $\mathbf{R} = \mathbb{Z}[x]/(x^2 + x + 1)$ and the generator $\mathbf{v} = (1, -1)$. The lattice is generated by the basis $\{\mathbf{v}, \mathrm{rot}(\mathbf{v})\}$ where $\mathrm{rot}(\mathbf{v}) = (1, 2)$. The plotted points are obtained by taking all integer linear combinations of these basis vectors.

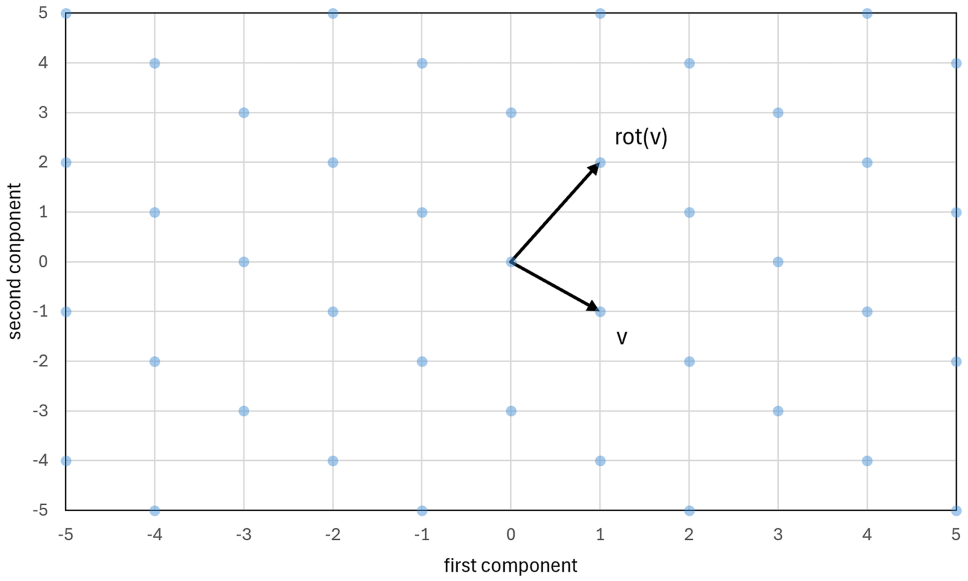

Figure 1: An example of the prime cyclotomic lattice.

3 Related Work

3.1 Enumeration (ENUM) Algorithms

Unlike basis reduction algorithms, enumeration-based algorithms attempt to identify the shortest lattice vector without modifying the input basis. Enumeration (ENUM) [20] is a representative algorithm of this type (Algorithm 1). Given a lattice basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of $\mathcal{L}$, any lattice vector can be expressed as an integer linear combination of the basis vectors:

$$\mathbf{v} = v_1\mathbf{b}_1 + \cdots + v_n\mathbf{b}_n \quad (\exists v_1, \ldots, \exists v_n \in \mathbb{Z}).$$

Algorithm 1: ENUM algorithm

Require: GSO coefficients $\mu_{i,j}$ $(1 \le j < i \le n)$, GSO vectors' norm $\|\mathbf{b}_i^*\|^2$ $(1 \le i \le n)$, and a search bound R.
Ensure: the integer coefficients of the shortest vector $\mathbf{v} = \sum_{i=1}^{n} \tilde{x}_i\mathbf{b}_i \in \mathcal{L}$.

1: $(x_1, \cdots, x_n) \leftarrow (1, 0, \cdots, 0)$
2: $(\tilde{x}_1, \cdots, \tilde{x}_n) \leftarrow (1, 0, \cdots, 0)$
3: $(\rho_1, \cdots, \rho_n, \rho_{n+1}) \leftarrow (0, 0, \cdots, 0)$
4: $(c_1, \cdots, c_n) \leftarrow (0, 0, \cdots, 0)$
5: $(w_1, \cdots, w_n) \leftarrow (0, 0, \cdots, 0)$
6: $k \leftarrow 1,\ last\_nonzero \leftarrow 1$
7: **while** true **do**
8: $\quad \rho_k \leftarrow \rho_{k+1} + (c_k + x_k)^2 \|\mathbf{b}_k^*\|^2$
9: $\quad$ **if** $\rho_k < R$ **then**
10: $\quad\quad$ **if** $k = 1$ **then**

(Continued)

Algorithm 1 (continued)

$\tilde{\mathbf{x}} \leftarrow \mathbf{x}$
$R \leftarrow \rho_1$
else
$k \leftarrow k - 1$
$c_k \leftarrow \sum_{i=t+1}^{n} x_i \mu_{i,k}$
$x_k \leftarrow \lfloor -c_k \rceil$
$w_k \leftarrow 1$
end if
else
$k \leftarrow k + 1$
if $k = n + 1$ **then**
return $\tilde{\mathbf{x}}$
end if
if $last\_nonzero \le k$ **then**
$last\_nonzero \leftarrow k$
$x_k \leftarrow x_k + 1$
else
if $c_k < x_k$ **then**
$x_k \leftarrow x_k - w_k$
else
$x_k \leftarrow x_k + w_k$
end if
end if
$w_k \leftarrow w_k + 1$
end if
end while

Let $\mathbf{b}_i^*$ denote the Gram–Schmidt orthogonalized (GSO) vectors and $\mu_{i,j}$ the corresponding GSO coefficients. Each basis vector $\mathbf{b}_i$ can then be written as

$$\mathbf{b}_i = \mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*.$$

Substituting this representation into $\mathbf{v}$, we obtain

$$\mathbf{v} = \sum_{i=1}^{n} v_i \left(\mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^* \right) = \sum_{j=1}^{n} \left(v_j + \sum_{i=j+1}^{n} \mu_{i,j} v_i \right) \mathbf{b}_j^*.$$

The squared norm of the projected vector at level k is given by

$$\|\pi_k(\mathbf{v})\|^2 = \sum_{j=k}^{n} \left(v_j + \sum_{i=j+1}^{n} \mu_{i,j} v_i \right)^2 B_j,$$

where $B_j = \|\mathbf{b}_j^*\|^2 (1 \le j \le n)$. Let $R > 0$ be a predefined upper bound. Finding a vector $\mathbf{v} \in \mathcal{L}$ satisfying $\|\mathbf{v}\| \le R$ is therefore equivalent to enforcing the inequality

$$\sum_{j=k}^{n}\left(v_j+\sum_{i=j+1}^{n}\mu_{i,j}v_i\right)^2 B_j \le R^2.$$

This inequality can be rewritten recursively in order to derive bounds for each coefficient v_k. For notational convenience, we define

$$T_k = R^2 - \sum_{j=k+1}^{n}\left(v_j+\sum_{i=j+1}^{n}\mu_{i,j}v_i\right)^2 B_j.$$

Then the admissible range of v_k is determined by

$$\left(v_k+\sum_{i=k+1}^{n}\mu_{i,k}v_i\right)^2 \le \frac{T_k}{B_k}.$$

The algorithm proceeds in a depth-first manner. At level $k = n$, integer values of v_n satisfying the bound are selected, where symmetry allows the restriction $v_n \le 0$. At level $k = n-1$, admissible values of v_{n-1} are determined, and backtracking is performed if no feasible choice exists. This recursive search continues for $1 \le k \le n$, enumerating all coefficient tuples $(v_1, \ldots, v_n)$ such that $\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{b}_i$ satisfies $\|\mathbf{v}\| \le R$. When R is appropriately chosen, the shortest vector is guaranteed to be included among the enumerated candidates.

The time complexity of ENUM is known to be $2^{O(n \log n)}$, while the space complexity remains polynomial in n.

In addition to the basic enumeration algorithm described above, several improved variants have been proposed to reduce the practical search cost. One of the most influential improvements is extreme pruning [5], introduced by Gama, Nguyen, and Regev.

Extreme pruning modifies the standard enumeration procedure by introducing a sequence of intermediate bounds (pruning coefficients) that restrict the search region at each level of the enumeration tree. Instead of exploring all candidate nodes within a fixed global radius, the algorithm discards branches early when partial norms exceed carefully optimized level-dependent thresholds. These pruning coefficients are determined by solving an optimization problem that minimizes the expected number of visited nodes subject to a prescribed success probability.

As a result, extreme pruning significantly reduces the number of explored nodes in practice compared to classical enumeration strategies. While the worst-case time complexity remains exponential, as in the original enumeration algorithm, the practical running time is substantially improved due to the reduction of the search space. The space complexity remains polynomial in the lattice dimension, since the algorithm still performs a depth-first traversal of the enumeration tree and does not require storing an exponential number of vectors.

In addition, recently, enumeration with more refined pruning strategies has also been studied. Luan et al. [21] improved discrete-pruned enumeration by refining the randomness assumption, introducing implementation-level optimizations, and constructing a cost simulator for estimating the running time under given parameters. Their results suggest that optimized discrete pruning can outperform extreme pruning in certain settings, while retaining the polynomial-space nature of enumeration. These studies indicate that enumeration remains an important practical SVP-solving technique, especially when memory consumption is a critical constraint.

3.2 BKZ Algorithms

The Block Korkine-Zolotarev (BKZ) algorithm [3] (Algorithm 2) can be viewed as a blockwise relaxation of Hermite-Korkine-Zolotarev (HKZ) reduction. It combines the LLL reduction procedure with enumeration in projected sublattices to progressively improve the basis quality.

Algorithm 2: BKZ reduction algorithm

Require: a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ in lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, a blocksize $2 \leq \beta \leq n$, a LLL Reduction parameter $\frac{1}{4} < \delta < 1$.
Ensure: a BKZ-β reduced basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathcal{L} \subseteq \mathbb{Z}^n$.

LLL($\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}, \delta$)
$z \leftarrow 0, k \leftarrow 0$
while $z < n - 1$ **do**
  $k \leftarrow (k \mod (n-1)) + 1$
  $l \leftarrow \min(k + \beta - 1, n)$
  $h \leftarrow \min(l + 1, n)$
  $\mathbf{v} \leftarrow$ENUM$(\mu_{[k,l]}, B_k, \ldots, B_l)$
  if $\|\mathbf{b}^*{}_k\| > \|\pi_k(\mathbf{v})\|$ **then**
    $z \leftarrow 0$
    $\{\mathbf{b}_1, \ldots, \mathbf{b}_h\} \leftarrow$ insert $\mathbf{v}$ as k'th basis vector
  else
    $z \leftarrow z + 1$
    LLL($\{\mathbf{b}_1, \ldots, \mathbf{b}_h\}, \delta$)
  end if
end while
return $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$

Definition 1 (HKZ-reduced basis): *Let* $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ *be a basis with GSO vectors* $\{\mathbf{b}_i^*\}$. *The basis is said to be HKZ-reduced if:*

1. *The basis is size-reduced.*
2. $\|\mathbf{b}^*{}_i\| = \lambda_1(\pi_i(L))$ *for all* $1 \leq i \leq n$.

HKZ reduction enforces optimality within each projected lattice, but its computational cost grows rapidly with dimension.

Definition 2 (BKZ β-reduced basis): *Given block size* $2 \leq \beta \leq n$, *a basis is said to be BKZ- β-reduced if:*

1. *It is size-reduced.*
2. *For each* $1 \leq k \leq n - \beta + 1$, *the projected sublattice* $\mathcal{L}_{[k,k+\beta-1]}$ *has a basis that is HKZ-reduced.*

In contrast to full HKZ reduction, BKZ enforces HKZ conditions only within sliding blocks of size β, thereby providing a trade-off between reduction quality and computational effort.

The BKZ procedure iteratively applies three main operations. First, LLL reduction is applied to the current basis, and the GSO coefficients $\mu_{i,j}$ and squared norms B_i are computed and stored. Let $l = \min(k + \beta - 1, n)$ and $h = \min(l + 1, n)$. Second, ENUM is executed on the projected block $\{\pi_k(\mathbf{b}_k), \ldots, \pi_k(\mathbf{b}_l)\}$ in order to obtain the shortest vector $\mathbf{v}$ in the corresponding projected sublattice. Third, the algorithm determines whether the discovered vector improves the basis.

If $\|\mathbf{b}^*_k\| > \|\pi_k(\mathbf{v})\|$, the vector $\mathbf{v}$ is inserted between $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$, after which the basis is reduced again, for example using LLL. If equality holds, the insertion step is skipped, but LLL is still applied to the block to facilitate subsequent enumeration steps.

Through repeated application of these steps over all valid block positions, the basis gradually approaches a BKZ-β-reduced form.

BKZ 2.0 [4] is an improved variant of the classical Schnorr–Euchner BKZ algorithm that refines the block reduction procedure while preserving the overall structure of BKZ. As in the original algorithm, BKZ 2.0 performs lattice reduction by iteratively applying a shortest vector subroutine to projected sublattices of dimension β, followed by size reduction and basis updates. The algorithm proceeds in tours, repeatedly sweeping through the basis and reducing each block in turn.

The primary improvement in BKZ 2.0 lies in the enhancement of the block SVP step. In particular, BKZ 2.0 integrates advanced pruning techniques into the enumeration algorithm used to solve the β-dimensional SVP instances. These pruning strategies significantly reduce the search space of enumeration while maintaining a high probability of finding short vectors, leading to a substantial improvement in practical efficiency compared to the original Schnorr–Euchner implementation.

In addition, BKZ 2.0 incorporates a more refined strategy for controlling and analyzing the evolution of the Gram–Schmidt norms during reduction. By modeling the behavior of these norms, the algorithm achieves a more stable and predictable reduction process, especially for large block sizes.

Overall, BKZ 2.0 maintains the fundamental block-based reduction framework of classical BKZ but improves the efficiency and robustness of the local SVP solving phase, resulting in better practical performance.

Furthermore, more recently, Li and Nguyen [22] provided a rigorous dynamic analysis of the practical BKZ algorithm. Unlike earlier analyses that were heuristic or applied only to theoretical BKZ variants, their analysis gives guarantees on the quality of the current basis during the execution of BKZ. They also considered variants in which the exact SVP oracle is replaced by an approximate oracle, which is closely related to practical BKZ implementations using pruned enumeration or sieving as subroutines. This line of work further clarifies the theoretical role of BKZ as a central preprocessing and basis-improvement component in practical SVP solvers.

3.3 Sieve Algorithms

Sieve algorithm series is another major approach to solving SVP. Unlike enumeration, which performs a depth-first search over coefficient vectors, sieve algorithms maintain a large list of lattice vectors and repeatedly combine them to obtain shorter vectors. The Gauss Sieve [12] is a practical heuristic sieve algorithm that recursively reduces sampled lattice vectors and was one of the first sieve algorithms to become competitive with enumeration in practice. Several improvements have been proposed to make sieving more practical. Progressive sieving [14] starts from lower-dimensional projected sublattices and gradually increases the sieving dimension, allowing the algorithm to reuse previously obtained short vectors and improve convergence. The General Sieve Kernel (G6K) [13] further systematized these ideas by providing a stateful framework for lattice sieving. G6K incorporates techniques such as vector reuse, lifting, dimensions for free, and deferred insertion, and has achieved new records in the Darmstadt SVP Challenge.

Recent studies have further improved practical sieving implementations. Ducas et al. [23] accelerated G6K using GPUs and Tensor Cores, showing that advanced sieving algorithms can benefit significantly from low-precision parallel computation. Their implementation reached dimension 180 in the Darmstadt SVP Challenge and demonstrated large improvements in wall-clock time and energy efficiency. Zhao et al. [24]

studied the memory-access cost of sieving and proposed an optimized BGJ (Becker-Gama-Joux 2015) sieve with streaming memory access. Their work highlights that in large-scale SVP computations, memory access patterns can be as important as the asymptotic number of arithmetic operations.

For ideal lattices, sieve algorithms can also exploit algebraic structure. Narisada et al. [18] proposed a rotation-based Gauss Sieve for general ideal lattices. By using the rotation properties of ideal lattices, their method reduces the internal cost of the Gauss Sieve and extends the applicability of ideal lattice sieving techniques beyond previously covered dimensions. These recent developments show that practical SVP solving is increasingly shaped by both algorithmic refinements and architecture-aware implementations.

3.4 ESR Algorithm

The ENUM-Sieve Reduction (ESR) algorithm [19] is a lattice reduction method in which short vectors obtained through sieving are incorporated back into the basis, followed by the application of the BKZ algorithm to reorganize and further improve the basis quality. The ESR algorithm is presented in Algorithm 3 and the rotation process in Algorithm 4.

Algorithm 3: ENUM-sieve reduction: $\mathrm{ESR}(\mathbf{B}, \gamma)$

Require: a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ in ideal lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, norm bound parameter γ of output vector.
Ensure: a vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \gamma \mathrm{GH}(\mathbf{B})$.

BKZ$(\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}, \beta \geq 20, \delta = 0.99)$
set initial blocksize m_e (ENUM) and m_s (Sieve)
while true **do**
  Partition $\mathbf{B}$ into $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$ using blocksize m_e
  Partition $\mathbf{B}$ into $\mathbf{B}_4, \mathbf{B}_5$ using blocksize m_s
  start parallel computation
  for $i = 1$ to 3 **do** ▷ each i runs in parallel on CPU
    $\mathbf{v}_i \leftarrow \mathrm{ENUM}(\mathbf{B}_i)$
  end for
  for $j = 4$ to 5 **do** ▷ each j runs sequentially on GPU
    $\mathbf{v}_j \leftarrow \mathrm{G6K}(\mathbf{B}_j)$
  end for
  end parallel computation
  for $i = 1$ to 5 **do**
    $\mathbf{v}_i \leftarrow \mathrm{rot}(\mathbf{v}_i)$
  end for
  $\mathbf{v}_{\mathrm{set}} = \{\mathbf{v}_1, \ldots, \mathbf{v}_5\}$
  if $\min_{\mathbf{v} \in \mathbf{v}_{\mathrm{set}}} \|\mathbf{v}\| \leq \gamma \mathrm{GH}(\mathbf{B})$ **then**
    return $\mathbf{v}$
  end if
  insert $\mathbf{v}_{\mathrm{set}}$ to $\mathbf{B}$
  $\mathrm{BKZ}(\mathbf{B}, \beta \geq 20, \delta = 0.99)$
  $\mathbf{v} \leftarrow$ shortest vector in $\mathbf{B}$
  if $\|\mathbf{v}\| \leq \gamma \mathrm{GH}(\mathbf{B})$ **then**
    return $\mathbf{v}$
  else
    progressively enlarge the blocksize m_e, m_s

(Continued)

Algorithm 3 (continued)

28: **end if**
29: **end while**

Algorithm 4: Rotation algorithm: rot($\mathbf{v}$)

Require: a vector $\mathbf{v} = (v_0, \ldots, v_{n-1})$ in ideal lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ with a monic polynomial $f(x) = x^n + x^{n-1} + \cdots + 1$.
Ensure: the rotation of the input vector $\mathbf{v}' \in \mathcal{L}$ with the smallest norm among all its rotation.

1: $\mathbf{v}' = \mathbf{v}$
2: **for** $i = 0$ **to** $n - 1$ **do**
3: $\quad \mathbf{v} = (-v_{n-1}, v_0 - v_{n-1}, \ldots, v_{n-2} - v_{n-1})$
4: $\quad$ **if** $\|\mathbf{v}'\| > \|\mathbf{v}\|$ **then**
5: $\quad\quad \mathbf{v}' = \mathbf{v}$
6: $\quad$ **end if**
7: **end for**
8: **return** $\mathbf{v}'$

First, BKZ algorithm is applied to the input basis $\mathbf{B}$ of the ideal lattice in order to reduce it. Next, the basis is divided into multiple sub-bases. In Progressive Lattice Sieving, sub-bases are created from the top of the basis, and sieving is performed sequentially. In contrast, the ESR algorithm begins by splitting the input basis $\mathbf{B}$ into two halves: a top part and a bottom part. Then, G6K-based sieving is performed on both sub-bases in parallel to obtain two short vectors. These vectors are then refined by applying rotations in the ideal lattice to find their rotated versions with the smallest norm. The resulting vectors are inserted at the beginning of the original basis, and the basis is reduced using the BKZ algorithm. If the shortest vector $\mathbf{v}$ in the new basis satisfies the condition $\|\mathbf{v}\| \leq \gamma \mathrm{GH}(\mathbf{B})$, then the algorithm terminates. Otherwise, the dimension of the sub-basis is incremented, and the process is repeated.

Furthermore, G6K sieving is executed on the GPU, and in parallel, ENUM is performed on the CPU to fully utilize computational resources. The sub-bases used for ENUM are different from those used for sieving: the original basis is split into three parts, from which three vectors are obtained. Each ENUM operation is performed in parallel on its respective sub-basis. Just like the vectors obtained from sieving, these ENUM-derived vectors are also refined using rotations to obtain the version with the smallest norm. Then, all five vectors are inserted into the original basis, and the BKZ algorithm is applied to reorganize. If the condition $\|\mathbf{v}\| \leq \gamma \mathrm{GH}(\mathbf{B})$ is not satisfied, the dimensions of both the sieve and ENUM sub-bases are increased, and the loop continues.

4 Our Proposal

4.1 Motivation

The goal of this study is to fully exploit both CPU and GPU computational resources to obtain shorter lattice vectors than those produced by G6K, while maintaining comparable space complexity. The ESR algorithm in [19] combines enumeration and sieving techniques and applies them not to the entire lattice basis at once, but to a sequence of partial bases obtained by partitioning the original basis. By gradually increasing the dimension of the working basis, ESR eventually applies the solver to the full lattice.

However, this approach has several limitations. First, as the lattice dimension increases, the overall running time grows significantly. In particular, the standard BKZ reduction and classical enumeration

procedures used in ESR become increasingly expensive when stronger parameter settings are required. Second, due to the partitioning of the basis, the quality of the resulting sub-bases can vary, which causes some instances of the enumeration procedure to behave suboptimally and prevents them from achieving their expected performance. As a result, some partial bases contain relatively long and poorly reduced vectors, causing the enumeration procedure to behave suboptimally and reducing the overall efficiency of the framework. Furthermore, although ESR combines CPU-based enumeration and GPU-based sieving, the efficiency of this hybrid framework strongly depends on the quality of the intermediate reduced bases. This motivates the introduction of stronger reduction techniques and more stable partial-basis generation strategies.

Motivated by these observations, we propose an improved version of the ESR algorithm, called ESR 2.0. ESR 2.0 integrates BKZ 2.0 and extreme pruning enumeration in order to reduce the practical computational cost while enabling stronger parameter settings. In addition, ESR 2.0 introduces a unimodular-matrix-based partial basis generation strategy to improve the stability and diversity of the generated sub-bases. These modifications are intended to improve the scalability and robustness of the ESR framework in higher-dimensional settings under realistic computational resource constraints.

4.2 ESR 2.0

In ESR 2.0, we introduce three main modifications to the original ESR framework. These modifications affect the preprocessing, enumeration, and partial basis generation steps. ESR 2.0 is presented in Algorithm 5, and the overall flow is illustrated in Fig. 2.

Algorithm 5: ESR 2.0$(\mathbf{B}, \gamma)$

Require: a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ in ideal lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, norm bound parameter γ of output vector.
Ensure: a vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \gamma \mathrm{GH}(\mathbf{B})$.

BKZ 2.0$(\mathbf{B}, \beta \geq 20, \delta = 0.99)$
set initial blocksize m_e (ENUM) and m_s (Sieve)
while true **do**
  for $k = 1$ to 5 **do**
    generate unimodular matrix $\mathbf{U}_k$
    $\mathbf{B}_k = \mathbf{U}_k \cdot \mathbf{B}$
    BKZ 2.0$(\mathbf{B}_k, \beta \geq 20, \delta = 0.99)$
  end for
  start parallel computation
  for $i = 1$ to 3 **do** ▷ each i runs in parallel on CPU
    $\mathbf{v}_i \leftarrow \mathrm{ExtremePruningENUM}((\mathbf{B}_i)_{1:m_e})$
  end for
  for $j = 4$ to 5 **do** ▷ each j runs in sequential on GPU
    $\mathbf{v}_j \leftarrow \mathrm{G6K}((\mathbf{B}_j)_{1:m_s})$
  end for
  end parallel computation
  for $i = 1$ to 5 **do**
    $\mathbf{v}_i \leftarrow \mathrm{rot}(\mathbf{v}_i)$
  end for
  $\mathbf{v}_{\mathrm{set}} = \{\mathbf{v}_1, \ldots, \mathbf{v}_5\}$
  if $\min_{\mathbf{v} \in \mathbf{v}_{\mathrm{set}}} \|\mathbf{v}\| \leq \gamma \mathrm{GH}(\mathbf{B})$ **then**

(Continued)

Algorithm 5 (continued)

return v
end if
insert $\mathbf{v}_{\text{set}}$ to $\mathbf{B}$
BKZ 2.0$(\mathbf{B}, \beta \geq 20, \delta = 0.99)$
$\mathbf{v} \leftarrow$shortest vector in $\mathbf{B}$
if $\|\mathbf{v}\| \leq \gamma \text{GH}(\mathbf{B})$ **then**
return v
else
progressively enlarge the blocksize m_e, m_s
end if
end while

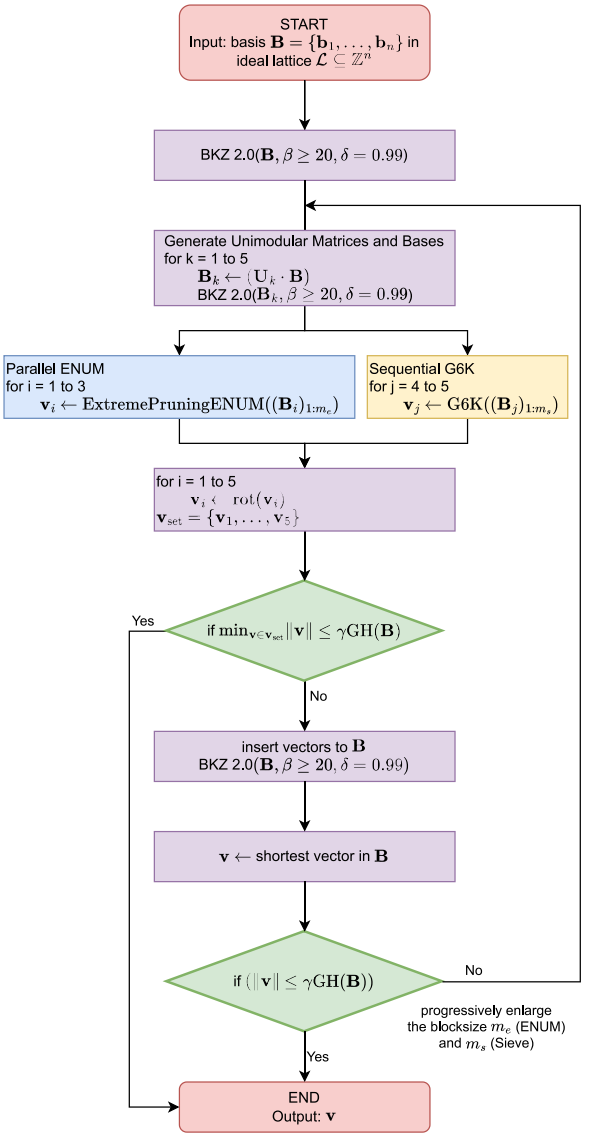

Figure 2: ESR 2.0 algorithm. Blue: CPU, yellow: GPU.

First, we replace the original BKZ algorithm with BKZ 2.0. It is adopted in order to enable the use of larger block sizes compared to the standard BKZ algorithm. In ESR 2.0, BKZ 2.0 is applied to the input basis at the initial stage, after generating partial bases using unimodular matrices, and also after merging short vectors back into the original basis.

Second, we replace all instances of the standard enumeration procedure in the original ESR with extreme pruning ENUM. In the original ESR, enumeration was applied separately to three partial bases; in ESR 2.0, all these enumeration steps are replaced with extreme pruning ENUM in order to reduce execution time. The extreme pruning ENUM procedure takes a success probability as an input parameter. We start with an initial success probability of 0.3, and if the enumeration fails, the probability is gradually increased to 0.5, 0.7, and 0.9. If extreme pruning ENUM still fails with a success probability of 0.9, the procedure is terminated and regarded as a failure.

Finally, we modify the method for generating partial bases by using unimodular matrices. Instead of directly partitioning the input basis as in the original ESR, we first multiply the entire basis by unimodular matrices to obtain multiple different bases representing the same lattice. In our implementation, we generate the unimodular matrix $\mathbf{U}_k$ using the method employed in the `random_unimodular_matrix()` function in SageMath. Namely, $\mathbf{U}_k$ is generated as $\mathbf{U}_k = \mathbf{PDLU}$, where $\mathbf{P}$ is a random permutation matrix, $\mathbf{D}$ is a diagonal matrix whose diagonal entries are independently chosen from $\{-1, 1\}$, and $\mathbf{L}$ and $\mathbf{U}$ are random lower and upper triangular integer matrices with all diagonal entries equal to 1. The off-diagonal entries of $\mathbf{L}$ and $\mathbf{U}$ are randomly sampled from a bounded integer interval $[-K, K]$. Since permutation matrices and sign matrices have determinant ± 1, and unit triangular matrices have determinant 1, the resulting matrix $\mathbf{U}_k$ is unimodular. Therefore, multiplying the basis by $\mathbf{U}_k$ preserves the lattice while generating different basis representations of the same lattice. After applying BKZ 2.0 to each of these bases, we extract the required number of basis vectors from the top rows to form partial bases of the desired dimension. At each iteration of Algorithm 5, five unimodular matrices $\mathbf{U}_1, \ldots, \mathbf{U}_5$ are independently sampled using the same parameter K. The transformed bases $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$ are used for ENUM, while $\mathbf{B}_4$ and $\mathbf{B}_5$ are used for G6K. Here the notation $\mathbf{B}_{1:\alpha}$ in Algorithm 5 denotes the submatrix consisting of the first α rows of the basis matrix $\mathbf{B}$, which are taken as a partial basis.

In the original ESR, directly partitioning the basis often resulted in partial bases with significantly different qualities, where the lower parts tended to contain vectors with larger norms. This imbalance caused the enumeration algorithm to behave suboptimally in some cases. By generating partial bases through unimodular transformations, ESR 2.0 mitigates this issue and enables more stable enumeration behavior across partial bases.

5 Experimental Results

5.1 Setup

All experiments were performed on a workstation equipped with an Intel(R) Xeon(R) Gold 6240 CPU running at 2.60 GHz and an NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPU, running Ubuntu 24.04.4 LTS with CUDA 12.8, Python 3.12.3, and fpylll 0.6.4. CPU time was measured using Python's `time.thread_time()` for each task, and the reported CPU time is the sum of these per-thread CPU times. GPU time was measured using PyTorch CUDA events around the G6K workout procedure. The implementation was written in Python and built upon the pro-pnj-bkz framework [25]. For lattice reduction and enumeration procedures, we employed the BKZ 2.0 algorithm and extreme pruning enumeration as implemented in the fpylll library [26].

In ESR 2.0, the block size and threshold parameters were fixed at $\beta = 25$ and $\gamma = 1.05$, respectively. During each iteration, the sub-basis dimension for both sieving and ENUM was increased in increments of 10. At the initial stage, the sieve was launched from a dimension equal to the full basis dimension minus 20. For ENUM, the starting dimension was set to the smaller value between (full dimension − 20) and 90.

In ESR, the same threshold parameter $\gamma = 1.05$ was used, while the block size was set to $\beta = 20$. The sub-basis dimension for sieving was configured in the same way as in ESR 2.0, starting from a dimension 20 less than the full basis dimension and increasing by increments of 10 during the iterations. For ENUM, the sub-basis dimension was fixed at 55.

Both the enumeration and sieving procedures were configured to output vectors whose norms do not exceed $1.05\mathrm{GH}(\mathbf{B})$, where $\mathrm{GH}(\mathbf{B})$ represents the Gaussian heuristic value of the basis $\mathbf{B}$. For the generation of unimodular matrices, the parameter K controlling the range of the off-diagonal entries of the lower and upper triangular matrices was set to 3. Furthermore, the vectors obtained from G6K were cyclically rotated, and the rotation yielding the smallest norm was selected.

5.2 Results and Discussion

We evaluated the proposed algorithm (ESR 2.0, Algorithm 5) in comparison with ESR and G6K as implemented in the pro-pnj-bkz framework. The experiments were conducted for lattice dimensions $n = \{96, 100, 102, 106, 108, 112, 126, 130\}$, where $n + 1$ is prime. For each dimension, ten randomly generated ideal lattice instances were prepared, and one run was performed for each instance. In all cases, we solved the approximate SVP under the condition $\|\mathbf{v}\| \leq 1.05\,\mathrm{GH}(\mathbf{B})$.

Our evaluation focuses on several performance indicators, including the ratio $\|\mathbf{v}\|/\mathrm{GH}(\mathbf{B})$, CPU Time, GPU Time, and the Peak Memory Usage observed during execution. The detailed results are presented in Figs. 3–6. For lattice dimensions higher than 130, we conducted a single experiment per dimension with the objective of surpassing the existing records in the Ideal Lattice Challenge [27].

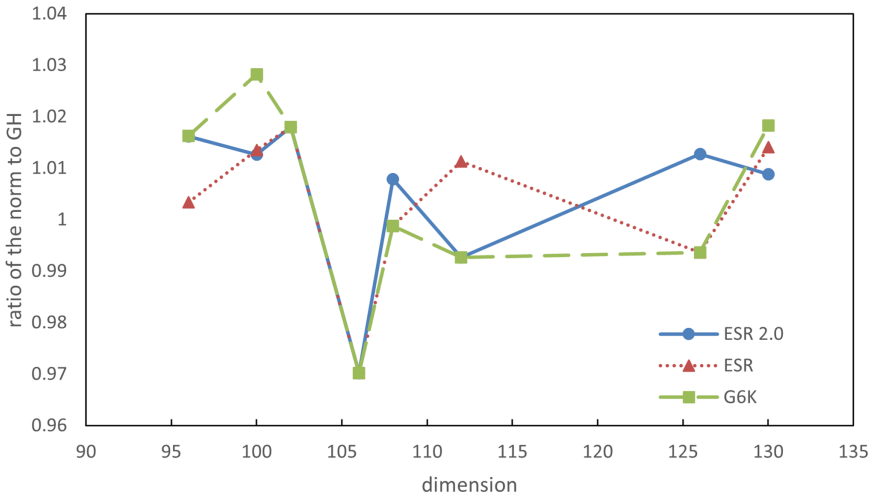

Figure 3: Comparison of minimum ratio $\|\mathbf{v}\|/\mathrm{GH}(\mathbf{B})$ among ESR 2.0, ESR and G6K.

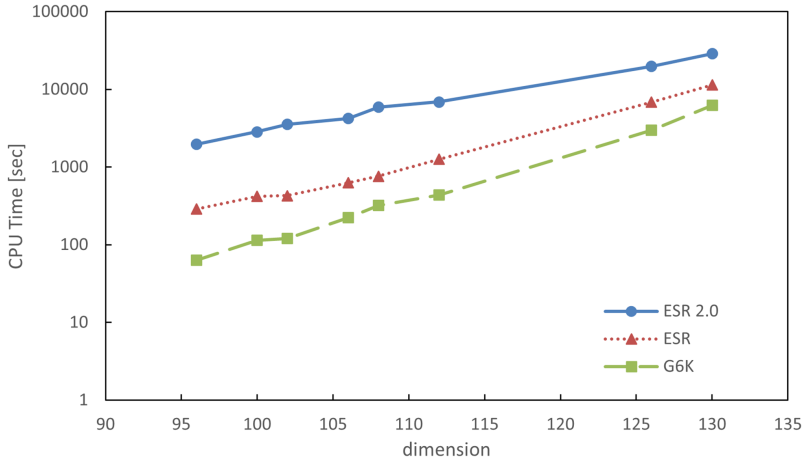

Figure 4: Comparison of average CPU time among ESR 2.0, ESR and G6K.

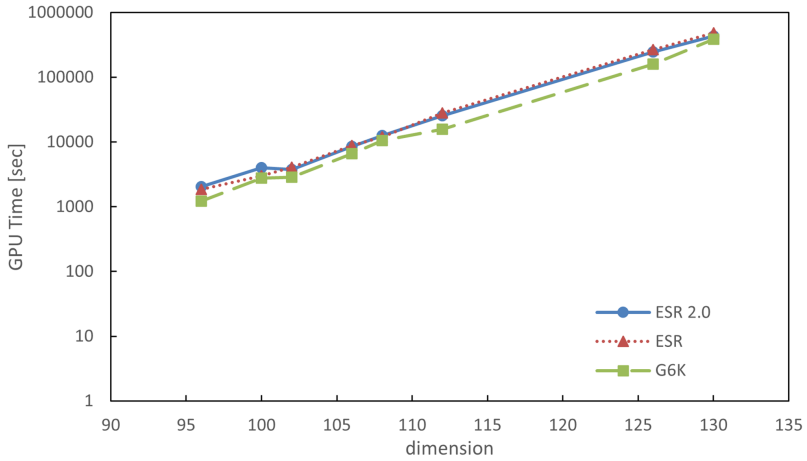

Figure 5: Comparison of average GPU time among ESR 2.0, ESR and G6K.

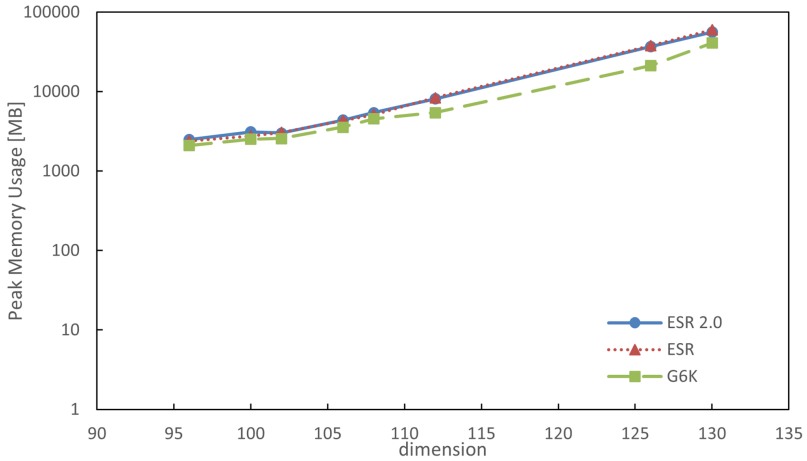

Figure 6: Comparison of average peak memory usage among ESR 2.0, ESR and G6K.

5.2.1 Ratio of the Norm to GH

As shown in Fig. 3, ESR 2.0 achieves output vector norms less than or equal to those obtained by G6K in 75% of the trials. In comparison with ESR, ESR 2.0 produces vectors with norms less than or equal to those obtained by ESR in 62.5% of the trials. Although the overall differences are relatively small, ESR 2.0 demonstrates a slight tendency to produce shorter vectors in several dimensions. Not that such improvements can still be meaningful in practice, since even slight differences in vector quality may contribute to benchmark record updates and may also influence parameter selection and security estimation in lattice-based cryptography. This behavior may be attributed to the modifications introduced in the internal algorithmic components, which, in our experiments, enabled the use of more powerful parameter settings under realistic computational resource constraints.

5.2.2 CPU Time

Fig. 4 shows that G6K and ESR require less CPU time than ESR 2.0. The main source of the additional CPU overhead in ESR 2.0 is the preprocessing stage, especially the repeated application of extreme-pruning enumeration and BKZ 2.0 reduction. Although extreme pruning enumeration used in ESR 2.0 is generally faster than the standard enumeration used in ESR, ESR 2.0 adopts an iterative strategy in which the success probability of extreme pruning enumeration is initially set to 0.3 and increased by 0.2 whenever the enumeration fails, up to a maximum of 0.9. As a result, extreme pruning enumeration may be executed multiple times, which can increase the total CPU time. Furthermore, ESR 2.0 generates multiple partial bases using unimodular matrix transformations, and BKZ 2.0 reduction is applied to each generated basis before enumeration. This repeated preprocessing procedure further increases the computational cost. In addition, ESR 2.0 uses larger sub-basis dimensions for enumeration than ESR and adopts a larger block size β in the BKZ 2.0 reduction. These stronger parameter settings improve basis quality but also increase the computational cost of preprocessing, especially local basis reduction and enumeration. Nevertheless, in our experiments, the relative CPU-time gap tends to decrease for larger dimensions. This suggests enhanced scalability in higher-dimensional regimes.

5.2.3 GPU Time

As illustrated in Fig. 5, no significant difference in GPU time is observed among ESR 2.0, ESR and G6K. Although ESR 2.0 and ESR perform multiple sieving steps, including additional sieving on sub-bases, the GPU time consumed by these sub-basis sieving operations appears to be negligible compared to the cost of the final sieving conducted on the full basis. As a result, the overall GPU time remains comparable to that of G6K, which performs sieving only once on the full basis.

5.2.4 Peak Memory Usage

As illustrated in Fig. 6, there is no notable difference in peak memory consumption among ESR 2.0, ESR and G6K. This can likely be explained by the fact that all three algorithms reach their peak memory usage during the sieving process on the full basis, which occurs in the third round of ESR 2.0 and ESR as well as in the single sieving phase of G6K.

Overall, the experimental results suggest that ESR 2.0 can be interpreted as a trade-off approach that increases CPU computation time in exchange for improved vector quality, while maintaining similar GPU time and memory consumption compared with ESR and G6K. This design may be beneficial in settings where GPU resources and memory capacity are more restrictive than additional CPU computation time.

5.3 Record Results on the Ideal Lattice Challenge

We additionally evaluated ESR 2.0 on selected instances from the Ideal Lattice Challenge. For dimensions below 130, namely 112 and 126, ten independent runs were conducted for each dimension, and the best norm obtained among those runs is reported. In contrast, for dimensions above 130, namely 136, 148, and 156, the computational cost becomes substantially higher, and therefore a single run was performed for each dimension with the primary objective of record improvement.

As a result, shorter vectors were found for several Ideal Lattice Challenge instances and the results are summarized in Table 1. Although the record for dimension 112 has subsequently been surpassed by other work, the results for dimensions 126, 136, 148 and 156 remain unbeaten at the time of writing. In spite of the improvements shown in Fig. 3 are relatively modest, ESR 2.0 appears to become more effective in larger lattice dimensions, which contributes to the record-setting results. We believe that this behavior is related to the progressive basis-quality improvements performed throughout the ESR 2.0 framework. By gradually improving the basis quality, ESR 2.0 can reduce the practical computational burden of solving SVP in larger dimensions compared with approaches such as G6K. These findings further demonstrate the capability of ESR 2.0 in challenging high-dimensional ideal lattice benchmarks.

Table 1: Record norms obtained on the ideal lattice challenge instances.

Dimension	Recorded Norm	Previous Record
112	2750	2763
126	2830	2922
136	2975	3070
148	3161	3207
156	3257	3302

6 Conclusion

In this paper, we presented ENUM-Sieve Reduction 2.0 (ESR 2.0), an enhanced hybrid SVP solver building upon our previous ESR framework. ESR 2.0 integrates BKZ 2.0 and extreme pruning enumeration into the reduction process and redesigns the partial basis generation strategy by employing unimodular matrix transformations. Our experimental results suggest that these modifications enable the use of stronger parameter settings and make it feasible to execute the algorithm in larger lattice dimensions under practical computational resource constraints. As a result, ESR 2.0 appears to improve the applicability and effectiveness of the original ESR framework.

Experimental evaluation demonstrated that ESR 2.0 produced vectors with norms equal to or shorter than those obtained by G6K in 75% of the trials. Moreover, compared with ESR, ESR 2.0 produced vectors with norms equal to or shorter than those obtained by ESR in 62.5% of the trials. Although ESR 2.0 required longer CPU time, this can be attributed to its iterative enumeration strategy and stronger parameter settings. In particular, ESR 2.0 repeatedly executes extreme pruning enumeration with gradually increasing success probabilities when enumeration fails, which may increase the total CPU time. Moreover, it employs larger sub-basis dimensions for enumeration and a larger block size β in BKZ 2.0, both of which increase the computational cost. Nevertheless, no substantial differences were observed in GPU time or peak memory usage among the three methods. These results indicate that the enhanced design improves solution quality while maintaining comparable resource efficiency in GPU and memory consumption. Furthermore, ESR 2.0 achieved record norms in the Ideal Lattice Challenge for dimensions 112, 126, 136, 148 and 156. While the

record for dimension 112 has subsequently been surpassed, the results for dimensions 126, 136, 148 and 156 remain unbeaten at the time of writing.

Despite the promising empirical performance, a formal theoretical analysis of ESR 2.0 remains an open problem. The current study focuses exclusively on prime cyclotomic ideal lattices, and extending the framework to other classes of ideal lattices or to general unstructured lattices constitutes an important direction for future research. When applied to other ideal lattices or general lattices, rotational symmetry cannot be exploited, and therefore a decrease in performance is expected. We expect that the hybrid integration of enumeration and sieving techniques may remain beneficial in less structured settings, although its effectiveness must be verified experimentally.

Overall, this work demonstrates that ESR 2.0 is a practical and competitive tool for high-dimensional SVP solving. It contributes to a more accurate understanding of SVP hardness in ideal lattices, which plays a critical role in parameter selection and security evaluation for post-quantum cryptographic schemes.

Acknowledgement: We gratefully acknowledge Prof. Yohei Watanabe for providing access to his GPU server at UEC.

Funding Statement: This work was partially supported by JSPS KAKENHI Grant Numbers JP21K11751, JP25K21805, JP26K02909, JP26K14821, JST K Program Grant Number JPMJKP24U2, Japan.

Author Contributions: Kazutaka Toda and Yuntao Wang conceived and designed this study. Kazutaka Toda and Yuntao Wang developed the methodology and conducted the investigation and formal analysis. Hyungrok Jo and Yang Li performed the validation. Yuntao Wang provided resources, supervised the project, and managed project administration. Kazutaka Toda curated the data and prepared the visualizations. Kazutaka Toda prepared the original draft. Kazutaka Toda, Yuntao Wang, Hyungrok Jo and Yang Li reviewed and edited the manuscript. Funding was acquired by Yuntao Wang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Data and materials are contained within the article.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. Piscataway, NJ, USA: IEEE; 1994. p. 124–34.
2. Lenstra AK, Lenstra HW, Lovász L. Factoring polynomials with rational coefficients. Math Ann. 1982;261(4): 515–34. doi:10.1007/bf01457454.
3. Gama N, Nguyen PQ. Predicting lattice reduction. In: Advances in Cryptology-EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin/Heidelberg, Germany: Springer; 2008. p. 31–51.
4. Chen Y, Nguyen PQ. BKZ 2.0: better lattice security estimates. In: Lee DH, Wang X, editors. Advances in Cryptology-ASIACRYPT 2011. Berlin/Heidelberg, Germany: Springer; 2011. p. 1–20.
5. Gama N, Nguyen PQ, Regev O. Lattice enumeration using extreme pruning. In: Advances in Cryptology-EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin/Heidelberg, Germany: Springer; 2010. p. 257–78.
6. Aono Y, Wang Y, Hayashi T, Takagi T. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In: Advances in Cryptology-EUROCRYPT 2016—35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part I. Berlin/Heidelberg, Germany: Springer; 2016. p. 789–819.

7. Yasuda M, Nakamura S, Yamaguchi J. Analysis of DeepBKZ reduction for finding short lattice vectors. Des Codes Cryptogr. 2020;88(10):2077–100. doi:10.1007/s10623-020-00765-4.
8. Wang L, Wang Y, Wang B. A trade-off SVP-solving strategy based on a sharper pnj-BKZ simulator. In: Liu JK, Xiang Y, Nepal S, Tsudik G, editors. Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security, ASIA CCS 2023; 2023 Jul 10–14; Melbourne, VIC, Australia. New York, NY, USA: ACM; 2023. p. 664–77.
9. Schnorr C, Euchner M. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math Program. 1994;66:181–99.
10. Ajtai M, Kumar R, Sivakumar D. A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing. New York, NY, USA: ACM; 2001. p. 601–10.
11. Nguyen PQ, Vidick T. Sieve algorithms for the shortest vector problem are practical. J Math Cryptol. 2008;2(2):181–207. doi:10.1515/JMC.2008.009.
12. Micciancio D, Voulgaris P. Faster exponential time algorithms for the shortest vector problem. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010. New York, NY, USA: ACM; 2010. p. 1468–80.
13. Albrecht MR, Ducas L, Herold G, Kirshanova E, Postlethwaite EW, Stevens M. The general sieve kernel and new records in lattice reduction. In: Advances in Cryptology-EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Part II. Berlin/Heidelberg, Germany: Springer; 2019. p. 717–46.
14. Laarhoven T, Mariano A. Progressive lattice sieving. In: Lange T, Steinwandt R, editors. Post-Quantum Cryptography: 9th International Conference, PQCrypto 2018. Berlin/Heidelberg, Germany: Springer; 2018. p. 292–311.
15. Ducas L. Shortest vector from lattice sieving: a few dimensions for free. In: Nielsen JB, Rijmen V, editors. Advances in Cryptology-EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part I. Vol. 10820. Berlin/Heidelberg, Germany: Springer; 2018. p. 125–45.
16. Darmstadt T. SVP challenge. 2019 [cited 2026 Jun 5]. Available from: https://www.latticechallenge.org/svp-challenge.
17. Schneider M. Sieving for shortest vectors in ideal lattices. In: Youssef AM, Nitaj A, Hassanien AE, editors. Progress in Cryptology-AFRICACRYPT 2013: 6th International Conference on Cryptology in Africa. Proceedings. Vol. 7918. Berlin/Heidelberg, Germany: Springer; 2013. p. 375–91.
18. Narisada S, Okada H, Fukushima K, Kiyomoto S. Faster rotation-based gauss sieve for solving the SVP on general ideal lattices. IEICE Trans Fundam Electron Commun Comput Sci. 2021;E104(A(1)):79–88. doi:10.1587/transfun.2020cip0014.
19. Toda K, Wang Y. Proposal of an SVP solver on prime cyclotomic lattices. In: Provable and Practical Security: 19th International Conference, ProvSec 2025. Vol. 16172. Berlin/Heidelberg, Germany: Springer; 2025. 166–82.
20. Kannan R. Improved algorithms for integer programming and related lattice problems. In: Proceedings of the 15th Annual ACM Symposium on Theory of Computing. New York, NY, USA: ACM; 1983. p. 193–206.
21. Luan L, Gu C, Zheng Y, Shi Y. Lattice enumeration with discrete pruning: improvements, cost estimation and optimal parameters. Mathematics. 2023;11(3):766.
22. Li J, Nguyen PQ. A complete analysis of the BKZ lattice reduction algorithm. J Cryptol. 2025;38(1):12. doi:10.1007/s00145-024-09527-0.
23. Ducas L, Stevens M, van Woerden WPJ. Advanced lattice sieving on GPUs, with tensor cores. In: Canteaut A, Standaert F, editors. Advances in Cryptology-EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Vol. 12697. Berlin/Heidelberg, Germany: Springer; 2021. p. 249–79.
24. Zhao Z, Ding J, Yang BY. Sieving with streaming memory access. IACR Trans Cryptogr Hardw Embed Syst. 2025;2025(2):362–84. doi:10.46586/tches.v2025.i2.362-384.
25. Summwer. pro-pnj-bkz. 2023 [cited 2026 Jun 5]. Available from: https://github.com/Summwer/pro-pnj-bkz.
26. Development team TF. fpylll, a Python (2 and 3) wrapper for fplll. 2017 [cited 2026 Jun 5]. Available from: https://github.com/fplll/fpylll.
27. Darmstadt T. Ideal lattice challenge. 2019 [cited 2026 Jun 5]. Available from: https://www.latticechallenge.org/ideallattice-challenge.