PS-BKZ: Decoupling Outer Propagation and Local Processing in BKZ by Parameter Separation

Hutao Song[1], Hua Guo[1,2*], Yang Sun[1], Fengju Gao[1], Zongyang Zhang[1], Qianhong Wu[1]

[1]School of Cyber Science and Technology, Beihang University, Beijing, 100191, China.
[2]State Key Laboratory of Complex & Critical Software Environment (CCSE), Beijing, 100191, China.

*Corresponding author(s). E-mail(s): hguo@buaa.edu.cn;

Abstract

This work provides a systematic study of parameter separation in Block Korkine–Zolotarev (BKZ) reduction. We propose a general framework with separate parameters for outer propagation and the dominant local computation. The framework covers the local parameter settings of standard BKZ and variants using extended preprocessing, dimensions for free, or progressive sieving, with corresponding adaptations to the outer operations. This allows less expensive local processing without narrowing the outer propagation range.

Within this framework, we give idealized and executable constructions that use inner BKZ with a smaller blocksize to supply local vectors for larger outer blocks. We derive a worst-case quality bound for the idealized construction and analyze its quality–cost behavior using a practical fixed-point model. Experiments on the executable construction cover random q-ary bases up to dimension 300 and SVP Challenge bases up to dimension 240. The detailed comparisons at $n \geq 200$ show lower mean root-Hermite factors (RHFs) and 14.1%–59.3% shorter mean running times than BKZ 2.0. Joint improvements generally become more frequent with dimension. On q-ary inputs against BKZ 2.0 with blocksize 50, their frequency increases from 15.56% at $n = 150$ to 73.33% at $n = 300$.

Keywords: Lattice-based cryptanalysis, Lattice reduction, BKZ, Parameter separation

1 Introduction

Lattice-based cryptography is a major area of post-quantum cryptography, with security based on the presumed hardness of lattice problems [1, 2]. BKZ [3] and its variants [4–6] are among the most widely used lattice reduction algorithms, and they play a central role in both security estimation [7–10] and parameter selection for lattice-based schemes [11–13].

BKZ [3, 4] reduces a lattice basis through tours over overlapping projected blocks. At each position, a local short-vector subroutine searches the projected block. When it finds a vector shorter than the current first projected vector, that vector is inserted and the basis is updated. These updates affect the projected blocks encountered later in the tour, so local reductions influence the evolution of the whole basis over successive tours. This interaction is captured by dynamical analyses of BKZ [14, 15]. The blocksize β sets the maximum rank of a projected block and the overlap between consecutive blocks. For the same input lattice, a shorter first vector indicates better output quality, and the time and memory required to obtain it measure computational cost. Increasing β usually improves reduction quality but also sharply increases cost.

Local processing in BKZ has developed mainly along two lines: enumeration and sieving. On the enumeration side, BKZ 2.0 [4] combines pruning, preprocessing, and simulation in practical BKZ reduction, establishing the basic structure of modern enumeration-based BKZ. Recent works [16, 17] improve the quality–cost behavior of reduction based on enumeration by separating the preprocessing range from the enumeration range and using approximate enumeration oracles. These improvements target the time needed to reach a given reduction quality in the polynomial-space setting. On the sieving side, dimensions for free [18] and progressive sieving [19] reduce the practical cost of finding short vectors. Building on these techniques, G6K [20] and Pump-and-Jump BKZ [5, 20] improve the efficiency of sieving by using different parameters for the BKZ blocksize and the actual sieving dimension. For sieving-based BKZ, both running time and memory are central costs, since commonly used sieving algorithms require exponential memory in the sieving dimension.

These works demonstrate the benefits of parameter separation in BKZ, but their analyses are tied to specific enumeration or sieving procedures. We study the effects of parameter separation across different local processing procedures:

How does parameter separation itself affect the overall quality–cost behavior of BKZ?

Our framework separates the roles of the outer tour structure and local output quality in the reduction process. This allows us to study weaker local processing while retaining a larger outer propagation range. A common strategy is to preprocess the local block and then obtain a short vector by enumeration or sieving [4, 16, 17, 20]. The search may use a smaller dimension or a relaxed target. We consider a more aggressive weakening of local processing. We run inner BKZ with a blocksize below the projected local rank and directly use the first reduced basis vector in the outer tour, without an additional enumeration or sieving call after the inner reduction. We study whether larger outer blocks can compensate for the weaker local quality guarantees of inner BKZ and improve the quality–cost tradeoff.

1.1 Contributions

A parameter-separated BKZ framework. We propose a general BKZ framework with separate parameters for outer propagation and the dominant local computation. We prove that, for a fixed input basis and outer tour structure, local processing affects the global quality bound only through the quality of the vectors it returns. This allows less expensive local processing to be used without narrowing the outer propagation range. Table 1 shows how our framework covers the local parameter settings of standard BKZ and variants using extended preprocessing, dimensions for free, or progressive sieving. These correspondences require suitable choices of the local processing procedure and parameters, together with adaptations to the outer tour structure, vector insertion, and basis updates.

Table 1 Local parameter correspondences within our framework. In the last two columns, each equality gives the parameter correspondence, followed by a brief description of the corresponding parameter in the cited work.

BKZ	Outer propagation range	Dominant local computation
Our framework	β_1	β_2
Standard BKZ [3]	$\beta_1 = \beta$ block rank	$\beta_2 = \beta$ enumeration rank
BKZ 2.0 [4]	$\beta_1 = \beta$ block rank	$\beta_2 = \beta$ enumeration rank
Progressive BKZ [21]	$\beta_1 = \beta$ current blocksize	$\beta_2 = \beta$ enumeration rank
Self-dual BKZ [22]	$\beta_1 = \beta$ block rank	$\beta_2 = \beta$ enumeration rank
BKZ with D4F [18]	$\beta_1 = \beta$ block rank	$\beta_2 = \beta - f$ sieving dimension
BKZ with PS [19]	$\beta_1 = \beta$ block rank	$\beta_2 = d_{\mathrm{sv}}$ sieving dimension
pnj-BKZ [20]	$\beta_1 = \beta$ block rank	$\beta_2 = \beta - f$ sieving dimension
DeepBKZ 2.0 [23]	$\beta_1 = \beta$ block rank	$\beta_2 = \beta$ enumeration rank
BKZ with EP [16]	$\beta_1 = m$ preprocessing rank	$\beta_2 = k$ enumeration rank
BKZ with EP [17]	$\beta_1 = m$ preprocessing rank	$\beta_2 = k_\alpha$ enumeration rank
Improved pnj-BKZ [5]	$\beta_1 = \beta$ block rank	$\beta_2 = \beta - f$ sieving dimension
PotBKZ [24]	$\beta_1 = \beta$ block rank	$\beta_2 = \beta$ enumeration rank

D4F, PS, and EP denote dimensions for free, progressive sieving, and extended preprocessing, respectively. Here, f is the number of free dimensions, d_{sv} is the current sieving dimension, and k_α is the enumeration rank used with relaxation parameter α. Block and enumeration ranks refer to full blocks; tail blocks may be smaller.

Concrete constructions and quality–cost analysis. We give idealized and executable constructions that use inner BKZ with blocksize β_2 to supply local vectors for outer tours with a larger blocksize β_1. For the idealized construction, we derive a closed-form worst-case quality bound depending on both blocksizes. This bound does not capture how the changing projected inputs affect inner BKZ quality. We therefore develop a practical fixed-point model that jointly estimates local quality and the global basis profile. Numerical evaluation finds settings with lower modeled root-Hermite

factors (RHFs) at no higher modeled cost than standard BKZ. The fraction of standard BKZ blocksize settings admitting such improvements increases with dimension in the evaluated range, reaching about 56% at $n = 350$.

Experimental evaluation. Experiments on random q-ary bases up to dimension 300 and SVP Challenge bases [25] up to dimension 240 show simultaneous improvements in RHF and running time over BKZ 2.0. The parameter choices $(\beta_1, \beta_2) = (75, 40)$ and $(60, 45)$ improve both mean RHF and mean running time over BKZ 2.0 with blocksizes 50 and 55, respectively, in every reported case with $n \geq 200$. The time reductions range from 14.1% to 59.3%. Across the full tested grid, joint improvements are generally more frequent at higher dimensions. On the q-ary inputs against BKZ 2.0 with blocksize 50, their frequency increases from 15.56% at $n = 150$ to 73.33% at $n = 300$. A controlled experiment further shows that increasing the outer propagation range can improve RHF even when the inner blocksize and local input rank are fixed.

1.2 Technical Overview

We first analyze the general framework and then study the quality and cost of the construction using inner BKZ. Reducing the cost of an individual local call does not determine the quality–cost tradeoff of the complete reduction. A local insertion changes the basis and can affect the projected inputs of subsequent calls. For inner BKZ, output quality also depends on the inputs created by the outer reduction. We build on the existing tour analysis to connect local outputs with global quality, and then combine this analysis with the cost of inner BKZ over the outer tours.

To analyze different local processing procedures within the same tour-level relation, we express each local output through its effective local quality q_{eff}, obtained by normalizing the vector length. This supplies the local quality parameter q_j required by the Generic BKZ (GBKZ) analysis [15]. If the returned vector is liftable to the current outer lattice and the subsequent update is GBKZ-compatible, the corresponding step remains a valid GBKZ step. In the resulting tour-level relation, the propagation matrix is determined by the outer tour structure, and the local processing procedure enters through the sequence of effective local qualities.

For the construction using inner BKZ, this relation still requires local quality bounds or estimates. We consider the idealized choice $\mathcal{P}(\Lambda; \beta_2) = \mathrm{BKZ}(\Lambda; \beta_2)$, with outer blocksize β_1 and inner blocksize $\beta_2 < \beta_1$. Here, $\mathrm{BKZ}(\Lambda; \beta_2)$ denotes the first vector of the basis returned by inner BKZ. Applying the standard BKZ bound to each inner call gives local quality bounds that hold uniformly across outer tours. Substituting them into the GBKZ relation yields a closed-form bound on the first-vector quality after sufficient outer tours. This guarantee is independent of the changing input profile. In practice, the outer reduction changes the inputs to inner BKZ, whose outputs in turn affect the subsequent outer reduction. This feedback makes it difficult to build a reliable simulator for the executable construction.

We study this dependence through a practical fixed-point model that combines information about the current local state with a generic estimate of inner BKZ quality. Its state is a normalized profile of logarithmic GSO lengths, where GSO denotes Gram–Schmidt orthogonalization. At each outer position, the current profile estimate and the generic estimate from the standard practical BKZ baseline are combined heuristically,

subject to a local quality floor that prevents overly optimistic estimates. Embedding the estimate into the GBKZ fixed-point relation makes the local quality and the outer profile interdependent. The resulting map is nonlinear. We construct a compact convex set that it maps into itself and apply Brouwer's fixed-point theorem to prove the existence of a fixed point. At such a point, the local quality estimates used in the outer relation agree with those obtained from the resulting profile, giving a self-consistent estimate of first-vector quality.

We then combine the quality at this modeled profile with the estimated cost of the complete reduction. A smaller inner blocksize limits the cost of each enumeration, while the total cost also includes repeated inner reductions and outer tours. We use enumeration cost estimates together with the inner and outer tour structure to account for these repetitions. For each standard BKZ blocksize, we consider two-parameter choices whose modeled cost is no larger and record whether any of them gives a lower modeled RHF.

To evaluate the construction experimentally, we implement local processing as follows. We form an explicit floating-point matrix to represent each projected local lattice. Since the BKZ implementation takes an integer matrix as input, we round this matrix to obtain an integer working matrix for inner BKZ. We retain the transformation matrix produced by inner BKZ. Its first row gives integer coefficients for recovering the local vector on the original projected representation. The recovered vector is evaluated on that representation, and the same coefficients are used to lift it to the current outer lattice for insertion and update.

1.3 Related Work

In addition to the works discussed above, earlier work established the basic enumeration techniques used for local processing. Classical enumeration methods go back to Kannan [26] and Fincke–Pohst [27]. Extreme pruning [28] reduces the cost of lattice enumeration by searching only the most promising parts of the enumeration tree.

Other works modify the outer framework or implementation of BKZ, including progressive blocksize schedules and sharp simulators [21], self-dual BKZ variants [22], deep-insertion variants [23], and segmented or parallel implementations [6].

BKZ has also been studied through dynamic analysis frameworks. Hanrot, Pujol, and Stehlé [14] introduced an analysis of blockwise lattice algorithms based on dynamical systems. Li and Nguyen [29] developed a viewpoint based on Rankin profiles for studying volume profiles of lattice bases. Building on these two works, Li and Nguyen [15] later gave a complete dynamic analysis of BKZ through the GBKZ framework. These analyses describe BKZ through its basis profile and present local processing and outer propagation as distinct but interacting components. This viewpoint provides a foundation for analyzing the different roles of the two parameters in our framework.

Using inner BKZ for local processing gives the construction a recursive form. Recursive approaches to lattice reduction have also been studied in earlier work. Aggarwal et al. [30] develop a general recursive lattice reduction framework based on recursively finding lower-rank dense sublattices or their duals. Our focus is different: we preserve

the BKZ outer tour structure over projected local lattices and study how outer propagation and local processing can be decoupled within this structure and how this decoupling affects quality–cost behavior.

1.4 Discussion

The local processing procedure $\mathcal{P}$ need not use BKZ. Other choices, including procedures based on enumeration [4, 16, 17] or sieving [20], may be considered if their dominant local computation can be controlled by their own parameter and they return a local vector that can be used by the outer update. Each choice changes the local quality and cost analysis and therefore requires separate study.

Our framework could also be combined with modified tour structures [20], blocksize schedules [21], and dual or self-dual processing [22]. The current basis or block position could guide adaptive choices of β_1 and β_2. Such combinations require studying how the modified outer operations interact with the two parameter choices and affect quality and cost.

Building a reliable simulator for the executable construction remains a direction for future work. Existing BKZ simulators [4, 21] estimate the evolution of the basis profile. For our construction, a simulator would also have to estimate how inner BKZ responds to the projected inputs produced by the outer reduction. The practical fixed-point model studies this dependence through self-consistent local quality estimates. Extending this approach to describe the evolution over successive outer tours requires further study.

2 Preliminaries

Throughout the paper, $B = (b_1, \ldots, b_n)$ denotes an ordered basis of a rank-n lattice L. For $j \geq 1$, π_j denotes the orthogonal projection onto $\mathrm{span}(b_1, \ldots, b_{j-1})^{\perp}$. We use GSO as shorthand for Gram–Schmidt orthogonalization, and write $b_1^*, \ldots, b_n^*$ for the GSO vectors. The GSO data of the current basis are used to evaluate projections, projected lengths, and quantities related to determinants. For indices $a \leq b$, we write $B[a, b]$ for the projected block $(\pi_a(b_a), \ldots, \pi_a(b_b))$, and $L(B[a, b])$ for the lattice generated by this block. We use $\mathrm{vol}(\cdot)$ for lattice volume. All logarithms are base 2 unless the base is explicitly indicated. We also set $R_0(B) = 0$ when using Rankin profiles. Profile vectors are row vectors, and inequalities between them are componentwise.

2.1 BKZ and Its Tour Structure

Given a blocksize β, define $\kappa_j = \min\{\beta, n - j + 1\}$ and $n_j = j + \kappa_j - 1$. At position j, BKZ uses the projected block $B[j, n_j] = (\pi_j(b_j), \ldots, \pi_j(b_{n_j}))$ and the corresponding projected local lattice $L(B[j, n_j])$.

A BKZ tour scans the basis through its block positions. At each position, local processing is performed on the corresponding projected local lattice, and the current basis is updated according to the local output. Algorithm 1 summarizes this tour structure, abstracting away implementation details such as LLL reduction [31], local short-vector search, pruning strategies, and the subsequent insertion and update routines.

Algorithm 1 Basic BKZ Tour Structure [3, 15]

Require: A basis $B = (b_1, \ldots, b_n)$ of a rank-n lattice L, and a blocksize β
Ensure: An updated basis B

```
for each position j = 1, ..., n − 1 do
    Set κ_j ← min{β, n − j + 1} and n_j ← j + κ_j − 1
    Access the projected block B[j, n_j] through the current GSO data
    Run local processing on L(B[j, n_j]) to obtain a local short vector
    Insert/update the current basis according to the local short vector
end for
return B
```

Standard BKZ [3] and its GBKZ abstraction [15] both use this tour structure on overlapping local blocks. From the perspective of this paper, the blocksize β in standard BKZ plays two roles simultaneously. First, it determines how local improvements are propagated by the outer tour. Second, it determines the local dimension on which local processing acts, and it controls the dominant local computation. Thus, standard BKZ ties outer propagation and local processing to the same parameter.

2.2 The GBKZ Dynamic Analysis Framework

The GBKZ framework focuses on how local improvements accumulate into global quality changes over a complete tour and abstracts the behavior inside each local block through a local quality condition and a GBKZ-compatible update condition.

Definition 1 (Rankin profile [15]) Let $B = (b_1, \ldots, b_n)$ be a basis of a rank-n lattice L. The Rankin profile of B is

$$R(B) = (R_1(B), \ldots, R_n(B)), \qquad R_i(B) = \log \frac{\mathrm{vol}(B[1,i])}{\mathrm{vol}(L)^{i/n}}.$$

Here $R_1(B)$ is the logarithm of the normalized first-vector length, and the remaining coordinates record the evolution of prefix volumes.

Definition 2 (Normalized log-GSO profile) Let $b_1^*, \ldots, b_n^*$ be the GSO vectors of B. We define the normalized log-GSO profile of B as $G(B) = (G_1(B), \ldots, G_n(B))$, where

$$G_i(B) = \log \frac{\|b_i^*\|}{\mathrm{vol}(L)^{1/n}} = \log \|b_i^*\| - \frac{1}{n} \log \mathrm{vol}(L).$$

Since $\mathrm{vol}(L) = \prod_{i=1}^{n} \|b_i^*\|$, we have $\sum_{i=1}^{n} G_i(B) = 0$. The Rankin profile and the normalized log-GSO profile satisfy $R_i(B) = \sum_{k=1}^{i} G_k(B)$ and $G_i(B) = R_i(B) - R_{i-1}(B)$.

Below, we refer to $G(B)$ simply as the log-GSO profile.

Lemma 1 (GBKZ one-step propagation lemma [15]) *Suppose that a GBKZ-compatible local step transforms a basis B into a basis C. If, at position j, the local processing returns a vector*

b satisfying $\|\pi_j(b)\| \le q_j \operatorname{vol}(B[j,n_j])^{1/\kappa_j}$*, then* q_j *is the local quality parameter of this step and*

$$R_j(C) \le \frac{\kappa_j - 1}{\kappa_j} R_{j-1}(B) + \frac{1}{\kappa_j} R_{n_j}(B) + \log q_j.$$

Moreover, $R_i(C) \le R_i(B)$ *for* $1 \le i < j$ *or* $n_j \le i \le n$*.*

Lemma 2 (GBKZ tour-level dynamic system [15]) *Let* A_β*,* E*, and* Q *denote, respectively, the tour aggregation matrix for blocksize* β*, the triangular change-of-coordinates matrix between the Rankin profile and the log-GSO profile, and the orthogonal projection matrix onto the constant direction. Given the sequence of local quality parameters* $q = (q_1, \ldots, q_{n-1})$*, let* $g(q)$ *be the induced drift vector. If one GBKZ tour transforms* B *into* C*, then*

$$R(C) \le R(B)E^{-1}(A_\beta - Q)E + g(q)E.$$

Moreover, suppose that the same sequence q *gives valid local quality bounds at the corresponding positions in every tour under consideration. If* r^* *is the fixed point satisfying* $r^* = r^*E^{-1}(A_\beta - Q)E + g(q)E$*, then after* k *tours,*

$$R(B_k) \le (R(B_0) - r^*)E^{-1}(A_\beta - Q)^k E + r^*.$$

Thus, the GBKZ dynamic system determines the tour-level propagation relation from the outer blocksize and the local quality sequence $(q_1, \ldots, q_{n-1})$.

2.3 Lattice Basis Quality Evaluation

We now state the worst-case and practical fixed-point quality baselines for standard BKZ, followed by the RHF measure used later in the paper.

Lemma 3 (Worst-case first-vector quality of standard BKZ [15]) *Let* γ_β *be Hermite's constant in rank* β*, and let* $E(n,\beta) = (n-1)/(2(\beta-1)) + \beta(\beta-2)/(2n(\beta-1))$*. The standard BKZ worst-case first-vector bound for a fully* β*-BKZ-reduced basis is*

$$Q_{\mathrm{BKZ,wc}}(n,\beta) = \gamma_\beta^{E(n,\beta)}.$$

Equivalently, for such a basis $B = (b_1, \ldots, b_n)$*,* $\|b_1\| / \operatorname{vol}(L)^{1/n} \le \gamma_\beta^{E(n,\beta)}$*.*

The practical baseline below uses a fixed point of the modeled recurrence. Here, a fixed point represents a profile that is reproduced by the recurrence, so it provides a self-consistent profile from which the first-vector quality is obtained.

Definition 3 (Practical fixed-point baseline for standard BKZ [4, 15]) Let

$$h(\kappa) = \frac{\Gamma(\kappa/2+1)^{1/\kappa}}{\sqrt{\pi}}, \qquad \widetilde{\gamma}_\kappa = \begin{cases} \gamma_\kappa, & \kappa \le 50, \\ h(\kappa)^2, & \kappa > 50 \end{cases}.$$

We omit the auxiliary $(1+\varepsilon)^2$ factor and use the Gaussian heuristic value in this practical baseline. For $\kappa_j = \min\{\beta, n-j+1\}$ and $n_j = j + \kappa_j - 1$, define $x = (x_1, \ldots, x_n)$ by $x_n = 0$ and, for $1 \le j \le n-1$,

$$x_j = \frac{\kappa_j}{\kappa_j - 1} \cdot \frac{1}{2} \log \widetilde{\gamma}_{\kappa_j} + \frac{1}{\kappa_j - 1} \sum_{i=j+1}^{n_j} x_i.$$

Let $w = x(I - Q)$, where Q is the orthogonal projection onto the constant direction. The practical first-vector quality of standard BKZ is then defined as $Q_{\mathrm{BKZ,pr}}(n, \beta) = 2^{w_1}$.

The quantity $Q_{\mathrm{BKZ,pr}}(n, \beta)$ should be interpreted as a practical fixed-point estimate, rather than as a rigorous upper bound on the actual output quality.

Definition 4 (Root-Hermite factor (RHF)) For $Q(B) = \|b_1\|/\operatorname{vol}(L)^{1/n}$, the corresponding root-Hermite factor is

$$\delta_0 = Q(B)^{1/(n-1)} = \left(\frac{\|b_1\|}{\operatorname{vol}(L)^{1/n}}\right)^{1/(n-1)}.$$

Since $G_1(B) = \log Q(B)$, the RHF can equivalently be written as $\delta_0 = 2^{G_1(B)/(n-1)}$. Smaller values of the local quality parameters q_j, the first-vector quality $Q(B)$, and the RHF δ_0 correspond to better reduction quality.

2.4 Computational Estimates

We now state the enumeration cost estimates and tour count heuristic used later in the analysis.

Worst-case enumeration cost. In the worst-case branch, we use the full enumeration estimate $C^{\mathrm{wc}}_{\mathrm{enum}}(\beta) = \beta^{\beta/(2e)}$ [26–28], or equivalently, $\log C^{\mathrm{wc}}_{\mathrm{enum}}(\beta) = \beta \log \beta/(2e)$.

Practical enumeration cost. In the practical branch, we use the empirical enumeration-node estimate $\log C^{\mathrm{pr}}_{\mathrm{enum}}(\beta) = 0.18728\,\beta \log \beta - 1.0192\,\beta + 16.10$, as documented in the Lattice Estimator [32].

Tour count heuristic. We use $N_{\mathrm{tour}}(n, \beta) \asymp (n^2/\beta^2) \log n$ [15] as the tour count heuristic for standard BKZ. For numerical evaluation, we set $N_{\mathrm{tour}}(n, \beta) = (n^2/\beta^2) \log n$.

3 Parameter Separation in BKZ

This section develops a BKZ framework with separate parameters for outer propagation and the dominant local computation. We first describe the corresponding algorithmic framework, then establish its connection with the GBKZ dynamic analysis framework, and finally use the GBKZ tour-level relation to identify the respective effects of the two parameters.

We use *parameter separation* to describe the case where outer propagation and the dominant local computation do not share the same parameter. We denote the two parameters by β_1 and β_2, respectively. This separation does not require every aspect of local processing to be independent of β_1. It requires only that the parameter controlling the dominant local computation need not coincide with the parameter controlling outer propagation.

3.1 A Parameter-Separated BKZ Framework

We preserve the outer tour structure of BKZ and replace its local processing step by a procedure $\mathcal{P}$ using parameter β_2. At outer position j, the parameter β_1 determines a projected block $B[j, n_j]$ of rank κ_j and the corresponding projected local lattice $\Lambda_j = L(B[j, n_j])$. The procedure $\mathcal{P}$ then returns $v_j = \mathcal{P}(\Lambda_j; \beta_2) \in \Lambda_j \setminus \{0\}$. Algorithm 2 summarizes the framework.

Algorithm 2 Parameter-Separated BKZ Framework

Require: A basis $B = (b_1, \ldots, b_n)$ of rank n, parameters β_1 and β_2, and a procedure $\mathcal{P}$ for local processing
Ensure: An updated basis B

1: **for** each outer position $j = 1, \ldots, n-1$ **do**
2: Determine κ_j and n_j according to β_1 and the outer tour structure
3: Set $\Lambda_j \leftarrow L(B[j, n_j])$
4: Compute $v_j \leftarrow \mathcal{P}(\Lambda_j; \beta_2)$
5: Insert/update the outer basis according to the local output v_j
6: **end for**
7: **return** B

3.2 GBKZ Analysis of the Framework

To analyze the parameter-separated framework using GBKZ, we first represent the output of $\mathcal{P}$ by a local quality parameter. We then verify that the local output can be lifted to the current outer lattice and that the outer update is GBKZ-compatible. These are the conditions needed to use the GBKZ propagation relation.

Definition 5 (Effective local quality) Let Λ be a projected local lattice of rank κ, and let $v = \mathcal{P}(\Lambda; \beta_2) \in \Lambda \setminus \{0\}$ be the vector returned by $\mathcal{P}$. We define the effective local quality of this output by

$$q_{\mathrm{eff}}(\Lambda; \beta_2) := \frac{\|v\|}{\mathrm{vol}(\Lambda)^{1/\kappa}}.$$

At outer position j, we write $q_{\mathrm{eff},j} := q_{\mathrm{eff}}(\Lambda_j; \beta_2)$.

At outer position j, let $\Lambda_j = L(B[j, n_j])$, $v_j = \mathcal{P}(\Lambda_j; \beta_2)$, and $q_j = q_{\mathrm{eff}}(\Lambda_j; \beta_2)$. By Definition 5, $\|v_j\| = q_j \,\mathrm{vol}(\Lambda_j)^{1/\kappa_j}$, which gives the local quality condition used in the GBKZ analysis.

Lemma 4 (GBKZ compatibility of the parameter-separated framework) *Consider Algorithm 2. Suppose that, at each outer position j, the local output $v_j = \mathcal{P}(\Lambda_j; \beta_2)$ is liftable to a vector $b^{(j)}$ of the current outer lattice with $\pi_j(b^{(j)}) = v_j$, and that the outer update is GBKZ-compatible. Then every outer step is a valid GBKZ step with local quality parameter $q_j = q_{\mathrm{eff},j}$, and the outer layer is analyzable as a GBKZ process.*

Proof Fix an outer position j. Let $B[j, n_j]$ be the projected block determined by the outer tour, let κ_j be its rank, and let $\Lambda_j = L(B[j, n_j])$ be the corresponding projected local lattice. By Algorithm 2, the local processing at this position returns $v_j = \mathcal{P}(\Lambda_j; \beta_2) \in \Lambda_j$.

By Definition 5, $\|v_j\| = q_{\mathrm{eff},j} \operatorname{vol}(B[j, n_j])^{1/\kappa_j}$. Let $b^{(j)}$ be the vector used by the outer update, chosen so that its projection equals the local output, namely $\pi_j(b^{(j)}) = v_j$. Then

$$\|\pi_j(b^{(j)})\| \le q_{\mathrm{eff},j} \operatorname{vol}(B[j, n_j])^{1/\kappa_j}.$$

Thus the local quality condition in Lemma 1 holds with $q_j = q_{\mathrm{eff},j}$.

Since the outer update is assumed to be GBKZ-compatible, all hypotheses of Lemma 1 are satisfied. Hence the j-th outer step satisfies the GBKZ one-step propagation inequality with the above q_j. As j is arbitrary, every outer step is a valid GBKZ step, and the outer layer is analyzable as a GBKZ process with the stated local quality parameters. This completes the proof. □

Lemma 4 shows that the outer layer of the framework can be analyzed within GBKZ. We can therefore apply the GBKZ tour-level relation for the Rankin profile $R(B)$ to identify how β_1 and β_2 enter the reduction process.

3.3 Parameter Separation in the Reduction Process

We first prove that changing local processing leaves the outer propagation matrix unchanged for a fixed outer tour structure. We then use this result to analyze the respective roles of β_1 and β_2.

Theorem 1 (Parameter separation) *Fix an input basis B and an outer tour structure determined by β_1. All choices of the local procedure $\mathcal{P}$ and parameter β_2 satisfying the assumptions of Lemma 4 share the same outer propagation matrix*

$$E^{-1}(A_{\beta_1} - Q)E.$$

If two such choices produce the same effective local quality sequence q, they have the same GBKZ tour-level upper bound.

Proof By Lemma 4, the outer layer of Algorithm 2 is a GBKZ process. Following Lemma 2, let A_{β_1} denote the tour aggregation matrix determined by the outer tour with parameter β_1. We write $g(q)$ for the drift vector induced by the local quality sequence $q = (q_1, \ldots, q_{n-1})$, where $q_j = q_{\mathrm{eff}}(\Lambda_j; \beta_2)$.

Let $B^{(j-1)}$ and $B^{(j)}$ denote the bases immediately before and after the local step at position j, respectively. By the GBKZ one-step propagation lemma,

$$R_j\left(B^{(j)}\right) \le \frac{\kappa_j - 1}{\kappa_j} R_{j-1}\left(B^{(j-1)}\right) + \frac{1}{\kappa_j} R_{n_j}\left(B^{(j-1)}\right) + \log q_j.$$

For a fixed outer tour structure, the visited positions, κ_j, and n_j are fixed. Hence the coefficients of the profile terms in the one-step propagation inequality are independent of $\mathcal{P}$ and β_2, and the local output enters the inequality through $\log q_j$.

Applying Lemma 2 to this fixed sequence of steps gives

$$R(C) \le R(B)E^{-1}(A_{\beta_1} - Q)E + g(q)E.$$

In this relation, the linear term $E^{-1}(A_{\beta_1} - Q)E$ is determined by the outer tour structure, whereas the drift term $g(q)E$ is determined by the local quality parameters. Since E and Q depend only on the dimension, all admissible choices of $\mathcal{P}$ and β_2 share the same outer propagation matrix. If two such choices produce the same sequence q, their drift terms are also identical. For the same input basis B, the resulting tour-level upper bounds are therefore equal. □

The theorem allows different local processing choices to be analyzed within the same GBKZ relation. With the outer tour structure fixed, changing $\mathcal{P}$ or β_2 does not require a new derivation of the outer propagation relation. The corresponding local quality sequence is substituted into the same upper bound. Together with the computational cost of each local procedure, this allows the quality–cost tradeoff to be studied under a common propagation relation. The roles of the two parameters are detailed below.

Outer propagation. The parameter β_1 determines the outer tour structure, including the projected blocks visited by the tour. It therefore determines the tour aggregation matrix A_{β_1} and, through $E^{-1}(A_{\beta_1} - Q)E$, the form in which existing local improvements are propagated through the outer tour.

Local processing. The procedure $\mathcal{P}$ uses parameter β_2. For a fixed projected local lattice Λ_j, changing β_2 can change the returned vector v_j and hence the effective local quality $q_j = q_{\mathrm{eff}}(\Lambda_j; \beta_2)$. The resulting sequence $(q_1, \ldots, q_{n-1})$ determines the drift vector $g(q)$ and therefore the local quality term $g(q)E$ in the tour-level relation.

Fig. 1 summarizes these two parameter pathways and their roles in the GBKZ tour-level relation.

$$
\begin{array}{ccccccc}
\beta_1 & \longrightarrow & \text{outer tour structure} & \longrightarrow & A_{\beta_1} & \longrightarrow & E^{-1}(A_{\beta_1} - Q)E \\
 & & \downarrow & & & & \\
 & & \Lambda_j & & & & \\
 & & \downarrow & & & & \\
\beta_2 & \longrightarrow & \mathcal{P}(\Lambda_j; \beta_2) & \longrightarrow & q_j & \longrightarrow & g(q)E
\end{array}
$$

$$
R(C) \le R(B)\ \underbrace{E^{-1}(A_{\beta_1} - Q)E}_{\text{outer propagation term}}\ +\ \underbrace{g(q)E}_{\text{local quality term}}\ .
$$

Fig. 1 The roles of β_1 and β_2 in the GBKZ tour-level relation. The parameter β_1 determines the outer tour structure and the projected local lattice Λ_j. The procedure $\mathcal{P}$ uses parameter β_2, and its output determines the local quality parameters.

Changing β_1 affects both the propagation of local improvements and the projected inputs supplied to $\mathcal{P}$. Thus, even with β_2 fixed, the local quality sequence may change. This input dependence motivates the two analyses developed below for local processing based on inner BKZ: worst-case bounds that are uniform over outer tours and practical estimates that use the projected input profile.

4 Concrete Construction of the Parameter-Separated BKZ Framework

We first give an idealized construction by setting $\mathcal{P}(\Lambda;\beta_2) = \mathrm{BKZ}(\Lambda;\beta_2)$ in the framework of Section 3. This construction is used for the analysis in Section 5. We then develop an executable construction for the experimental evaluation in Section 6.

The executable construction uses two mechanisms to connect the outer tour with inner BKZ. First, we form an explicit matrix representing the projected local lattice. This matrix is used to evaluate the projected lengths required by the outer procedure. Since this matrix generally has real entries and the BKZ implementation used here accepts only integer matrices, we round it to obtain an integer working matrix for inner BKZ. The rounded matrix is used to obtain the coefficients of the inner output, whereas the outer procedure evaluates the corresponding vector on the explicit projected matrix. The second mechanism passes the inner result to the outer tour. We retain the transformation matrix produced by inner BKZ and apply the coefficients of the inner output to the explicit projected matrix. This recovers the projected vector used by the outer tour. Applying the same coefficients to the current outer basis produces the corresponding lattice vector used by the outer update.

In both constructions, β_1 is the outer blocksize, which determines the projected local lattices in the outer tour and their overlap structure, while β_2 is the inner blocksize, which controls the dominant local computation of inner BKZ. Throughout this section, the blocksizes are integers satisfying $2 \le \beta_2 \le \beta_1 \le n$.

4.1 Overall Structure of the Executable Construction

At outer position j, let the current basis be $B = (b_1, \ldots, b_n)$, and let the outer blocksize be β_1. Define $\kappa_j = \min\{\beta_1, n-j+1\}$ and $n_j = j + \kappa_j - 1$. The outer layer considers the projected block $B[j, n_j]$ and the projected local lattice $L(B[j, n_j])$.

Local processing at position j is performed by running inner BKZ with blocksize $\min\{\beta_2, \kappa_j\}$. The local output of inner BKZ is then passed through the vector recovery and lifting procedure, converted into a vector usable by the outer tour, and inserted at position j when it satisfies the outer improvement condition.

One outer tour scans all outer positions. At each position, the algorithm performs standard BKZ preprocessing, runs inner BKZ, and applies the liftable vector insertion and update. Algorithm 3 summarizes the overall construction.

4.2 Running Inner BKZ on an Integer Working Matrix

Ordinary local preprocessing may improve the local shape using a GSO representation or a floating-point projected structure, while a subsequent local short-vector procedure still produces the vector to be inserted. In the present construction, inner BKZ directly performs the local processing whose output is returned to the outer layer. The algorithm therefore retains the information needed to evaluate the projected length with respect to the current GSO representation and to convert a valid inner output into an update of the current outer basis.

Algorithm 3 Executable Two-Parameter BKZ Construction

Require: A basis $B = (b_1, \ldots, b_n)$, an outer blocksize β_1, and an inner blocksize β_2
Ensure: An updated basis B

Apply initial LLL reduction to B
repeat
  Set *clean* $\leftarrow$ true
  for each outer position $j = 1, \ldots, n-1$ **do**
    Set $\kappa_j \leftarrow \min\{\beta_1, n-j+1\}$ and $n_j \leftarrow j + \kappa_j - 1$
    Perform standard BKZ preprocessing at the current position
    $\mathcal{I}_j \leftarrow$ Algorithm 4$(B, j, \kappa_j, \beta_2)$
    $(B, \mathit{updated}) \leftarrow$ Algorithm 5$(B, j, \kappa_j, \mathcal{I}_j)$
    if *updated* **then**
      Set *clean* $\leftarrow$ false
    end if
  end for
until *clean* or a stopping condition is reached
return B

Concretely, the algorithm forms the floating-point projected representation $\widehat{B}_j = \big(\pi_j(b_j), \pi_j(b_{j+1}), \ldots, \pi_j(b_{n_j})\big)$ using the current GSO data. Since the BKZ implementation used for local processing takes an integer matrix as input, the entries of $\widehat{B}_j$ are rounded componentwise to the nearest integers to obtain an integer working matrix $\widehat{B}_j^{\mathbb{Z}}$. Inner BKZ is then run on $\widehat{B}_j^{\mathbb{Z}}$.

The length of the inner BKZ output on $\widehat{B}_j^{\mathbb{Z}}$ need not equal the projected length of the vector recovered by applying its coefficients to $\widehat{B}_j$. The algorithm therefore retains the transformation matrix produced by inner BKZ. Writing $\widehat{B}_{j,\mathrm{red}}^{\mathbb{Z}}$ for the reduced integer working basis, we have $\widehat{B}_{j,\mathrm{red}}^{\mathbb{Z}} = U_j \widehat{B}_j^{\mathbb{Z}}$. This information is used by the vector recovery and lifting procedure to recover the corresponding vector on $\widehat{B}_j$ and evaluate its projected length. Algorithm 4 summarizes the inner BKZ call and the information passed to the vector recovery and lifting step.

Algorithm 4 Running Inner BKZ on an Integer Working Matrix

Require: The current basis B, an outer position j, the local rank κ_j, and the inner blocksize β_2
Ensure: Local output information $\mathcal{I}_j$

Set $n_j \leftarrow j + \kappa_j - 1$
Compute $\widehat{B}_j = (\pi_j(b_j), \ldots, \pi_j(b_{n_j}))$ using the current GSO data
Round $\widehat{B}_j$ to obtain the integer working matrix $\widehat{B}_j^{\mathbb{Z}}$
Run inner BKZ with blocksize $\min\{\beta_2, \kappa_j\}$ on $\widehat{B}_j^{\mathbb{Z}}$
return $\mathcal{I}_j = (\widehat{B}_j, U_j)$, where U_j is the transformation matrix produced by inner BKZ

Algorithm 5 Vector Recovery and Lifting

Require: The current basis B, an outer position j, the local rank κ_j, and $\mathcal{I}_j = (\widehat{B}_j, U_j)$

Ensure: An updated basis B and a Boolean flag *updated*

1: Extract the first row $u_j = (u_{j,0}, \ldots, u_{j,\kappa_j-1})$ of U_j
2: Recover $v_j = \sum_{t=0}^{\kappa_j-1} u_{j,t}\pi_j(b_{j+t})$ using the floating-point projected representation
3: **if** v_j does not satisfy the outer improvement condition **then**
4: **return** (B, false)
5: **end if**
6: Construct $\widetilde{b}_j = \sum_{t=0}^{\kappa_j-1} u_{j,t}b_{j+t}$
7: Apply standard BKZ insertion and a GBKZ-compatible update to $\widetilde{b}_j$
8: **return** (B, true)

4.3 Vector Recovery and Lifting

Inner BKZ returns a reduced basis of the lattice represented by the integer working matrix. The construction uses the first vector of the inner BKZ output basis as the local short-vector candidate for the current outer position. This vector is represented by the first row $u_j = (u_{j,0}, \ldots, u_{j,\kappa_j-1})$ of the transformation matrix U_j. It records the integer coefficients of the first vector with respect to the rows of the integer working matrix $\widehat{B}_j^{\mathbb{Z}}$.

Applying the same coefficients to the floating-point projected representation $\widehat{B}_j$, the algorithm recovers the corresponding projected vector with respect to the current GSO representation: $v_j = \sum_{t=0}^{\kappa_j-1} u_{j,t}\pi_j(b_{j+t})$. The value $\|v_j\|^2$ is used in the standard BKZ improvement test at the current outer position, with the threshold specified in Section 4.4. If the projected vector does not satisfy the improvement condition, no insertion is performed in this local step.

If v_j satisfies the improvement condition, the same coefficient vector defines a vector in the current outer lattice: $\widetilde{b}_j = \sum_{t=0}^{\kappa_j-1} u_{j,t}b_{j+t}$. By linearity of projection, $\pi_j(\widetilde{b}_j) = \sum_{t=0}^{\kappa_j-1} u_{j,t}\pi_j(b_{j+t}) = v_j$. Thus, the projection of the lifted outer vector $\widetilde{b}_j$ is exactly the projected vector v_j used for local quality verification. Inserting $\widetilde{b}_j$ therefore converts the projected short vector into a local improvement for the outer tour.

After $\widetilde{b}_j$ is constructed, the algorithm applies the standard BKZ insertion and update. We require this update to be GBKZ-compatible so that it satisfies the update condition used in the analysis of Section 3. Algorithm 5 summarizes the vector recovery and lifting procedure.

4.4 GBKZ Compatibility of the Executable Construction

The executable construction follows the outer tour structure of the framework in Section 3 and satisfies its lifting condition. The outer layer preserves the BKZ tour structure with blocksize β_1. At each outer position j, the local procedure constructs an integer working matrix by rounding the explicit projected matrix, runs inner

BKZ with blocksize $\min\{\beta_2, \kappa_j\}$ on this matrix, and applies the resulting coefficients to the explicit projected matrix to recover a vector $v_j \in L(B[j, n_j])$. The vector recovery and lifting procedure constructs a vector $\widetilde{b}_j$ of the current outer lattice satisfying $\pi_j(\widetilde{b}_j) = v_j$. The subsequent insertion and update follow the standard GBKZ-compatible BKZ update procedure [15]. The implementation accepts a candidate when $\|v_j\|^2 < \theta\|b_j^*\|^2$, where $\theta \in (0, 1]$ is the improvement threshold. Setting $\overline{q}_j = \theta^{-1/2} q_{\mathrm{eff}}(\Lambda_j; \beta_2)$ bounds the normalized length of the inserted candidate when insertion is performed and that of the current first projected vector otherwise. Thus the GBKZ tour-level relation applies with local quality parameters $(\overline{q}_1, \ldots, \overline{q}_{n-1})$, while the outer propagation matrix remains unchanged.

5 Quality–Cost Analysis of the Idealized Construction

This section studies the quality–cost behavior of the idealized construction $\mathcal{P}(\Lambda; \beta_2) = \mathrm{BKZ}(\Lambda; \beta_2)$. We derive a worst-case quality bound and develop a practical quality model. Using the practical model and the cost estimates, we numerically compare the construction with standard BKZ.

We analyze reduction quality in two branches. The worst-case branch uses the standard BKZ worst-case bound for the quality supplied by inner BKZ and combines it with the GBKZ dynamic analysis to obtain a closed-form quality bound. The practical branch accounts for the dependence of inner BKZ quality on the current projected local lattice and leads to a practical fixed-point model. Both quality branches use the common cost model below.

Common cost model. For $\star \in \{\mathrm{wc}, \mathrm{pr}\}$, let $C^{\star}_{\mathrm{enum}}(\beta)$ denote the corresponding enumeration cost estimate, and let $N_{\mathrm{tour}}(n, \beta)$ denote the tour count estimate. We estimate the dominant cost of standard BKZ as $C^{\star}_{\mathrm{BKZ}}(n, \beta) = N_{\mathrm{tour}}(n, \beta)(n - \beta + 1)C^{\star}_{\mathrm{enum}}(\beta)$.

For the idealized construction, β_1 determines the outer tour structure, so we use $N_{\mathrm{tour}}(n, \beta_1)$ as a heuristic estimate of the outer-tour count. At positions with $\kappa_j = \beta_1$, inner BKZ uses blocksize β_2. We model the dominant cost by $C^{\star}_{\mathrm{sep}}(n, \beta_1, \beta_2) = N_{\mathrm{tour}}(n, \beta_1)(n - \beta_1 + 1)N_{\mathrm{tour}}(\beta_1, \beta_2)(\beta_1 - \beta_2 + 1)C^{\star}_{\mathrm{enum}}(\beta_2)$. The first two factors describe the outer tour, while the remaining factors describe the dominant inner BKZ computation. Both estimates omit the costs associated with shorter tail positions. Including the omitted tail positions changes some individual comparisons but preserves the trends reported below. Lower-order costs from other parts are also omitted.

We use these cost estimates with the corresponding quality model in each of the two analyses below.

5.1 Worst-Case Quality–Cost Analysis

In this worst-case analysis, we use fully BKZ-reduced inner outputs and set the outer improvement threshold to $\theta = 1$.

Since our construction allows $\beta_2 < \beta_1$, the standard BKZ quality guarantee for blocksize β_1 does not directly apply. We therefore derive a global worst-case quality bound to establish the overall guarantee provided by our construction and its dependence on both blocksizes.

At an outer position with $\kappa_j = \beta_1$, inner BKZ acts on a rank-β_1 projected local lattice with blocksize β_2. We use the standard BKZ worst-case first-vector bound as an upper bound on the effective local quality. This standard BKZ bound depends only on β_1 and β_2.

Definition 6 (Worst-case local quality bound for inner BKZ) Define $q_{\mathrm{inner,wc}}(\beta_1, \beta_2) := Q_{\mathrm{BKZ,wc}}(\beta_1, \beta_2) = \gamma_{\beta_2}^{E(\beta_1,\beta_2)}$. For a rank-$\beta_1$ projected local lattice Λ, the standard BKZ worst-case bound gives $q_{\mathrm{eff}}(\Lambda; \beta_2) \le q_{\mathrm{inner,wc}}(\beta_1, \beta_2)$.

Theorem 2 (Worst-case quality under inner BKZ local bounds) *Under the worst-case local quality bounds induced by inner BKZ, for every $\varepsilon > 0$, the first-vector quality bound after sufficient outer tours is $(1+\varepsilon)Q_{\mathrm{sep,wc}}(n, \beta_1, \beta_2)$, where*

$$Q_{\mathrm{sep,wc}}(n, \beta_1, \beta_2) = \gamma_{\beta_2}^{2E(\beta_1,\beta_2)E(n,\beta_1)}.$$

Proof If $\beta_1 = n$, the first local call acts on the whole lattice and gives first-vector quality at most $q_{\mathrm{inner,wc}}(n, \beta_2)$. Subsequent steps do not increase the first-vector length. Since $q_{\mathrm{inner,wc}}(n, \beta_2) \ge 1$ and $2E(n, n) \ge 1$, the stated bound follows. We therefore assume $\beta_1 < n$ below.

At an outer position j, let $\kappa_j = \min\{\beta_1, n-j+1\}$, $n_j = j+\kappa_j-1$, and $\beta_{2,j} = \min\{\beta_2, \kappa_j\}$. By Lemma 3, inner BKZ gives $\|v_j\| \le q_j \,\mathrm{vol}(B[j, n_j])^{1/\kappa_j}$, where $q_j := Q_{\mathrm{BKZ,wc}}(\kappa_j, \beta_{2,j}) = \gamma_{\beta_{2,j}}^{E(\kappa_j,\beta_{2,j})}$.

If $\kappa_j = \beta_1$, then $\beta_{2,j} = \beta_2$ and $q_j = q_{\mathrm{inner,wc}}(\beta_1, \beta_2)$. Thus

$$q_j^{\kappa_j} = q_{\mathrm{inner,wc}}(\beta_1, \beta_2)^{\beta_1}.$$

For the tail positions with $\kappa_j < \beta_1$, the corresponding tail analysis in [15] gives the following conclusion. If the local quality parameter at rank β_1 is q and every shorter rank κ_j has a local quality parameter q_j satisfying $q_j^{\kappa_j} \le q^{\beta_1}$, then the corresponding GBKZ fixed-point bound on the first-vector quality is $q^{2E(n,\beta_1)}$. Taking $q = q_{\mathrm{inner,wc}}(\beta_1, \beta_2)$, it is therefore sufficient to prove

$$q_j^{\kappa_j} \le q_{\mathrm{inner,wc}}(\beta_1, \beta_2)^{\beta_1}, \qquad 2 \le \kappa_j < \beta_1. \tag{1}$$

If $\beta_2 \le \kappa_j < \beta_1$, then $\beta_{2,j} = \beta_2$. By the definition of E in Lemma 3, $rE(r, \beta_2) = [r(r-1) + \beta_2(\beta_2-2)]/[2(\beta_2-1)]$, which is increasing in r. Hence

$$q_j^{\kappa_j} = \gamma_{\beta_2}^{\kappa_j E(\kappa_j,\beta_2)} \le \gamma_{\beta_2}^{\beta_1 E(\beta_1,\beta_2)} = q_{\mathrm{inner,wc}}(\beta_1, \beta_2)^{\beta_1}.$$

If $2 \le \kappa_j < \beta_2$, then $\beta_{2,j} = \kappa_j$. Newman's inequality, Eq. (33) in [15], gives $\gamma_{\kappa_j}^{\kappa_j} \le \gamma_{\beta_2}^{\beta_2}$. By the definition of E in Lemma 3, $E(r, r) = 1 - 1/[2(r-1)]$ is increasing in r, while $rE(r, \beta_2)$ is increasing for $r \ge \beta_2$. Hence

$$q_j^{\kappa_j} = (\gamma_{\kappa_j}^{\kappa_j})^{E(\kappa_j,\kappa_j)} \le \gamma_{\beta_2}^{\beta_2 E(\kappa_j,\kappa_j)} \le \gamma_{\beta_2}^{\beta_2 E(\beta_2,\beta_2)} \le \gamma_{\beta_2}^{\beta_1 E(\beta_1,\beta_2)} = q_{\mathrm{inner,wc}}(\beta_1, \beta_2)^{\beta_1}.$$

Equation (1) now holds for all tail positions. Together with the case $\kappa_j = \beta_1$ above, this bounds the first-vector quality at the GBKZ fixed point by $q_{\mathrm{inner,wc}}(\beta_1, \beta_2)^{2E(n,\beta_1)}$. These local quality bounds apply in every outer tour. By Lemma 2, the contribution from the initial basis tends to zero because $\|A_{\beta_1} - Q\|_2 < 1$ [15]. Thus, for every $\varepsilon > 0$, after sufficient outer tours, the first-vector quality is at most $(1+\varepsilon)q_{\mathrm{inner,wc}}(\beta_1, \beta_2)^{2E(n,\beta_1)}$. Substituting Definition 6 gives the stated bound. □

Theorem 2 gives a closed-form worst-case quality bound for the idealized construction. However, the worst-case local quality bound does not capture how the current projected local lattice affects the output of inner BKZ. The practical analysis below models this input dependence.

5.2 Practical Quality–Cost Analysis

The local quality is affected by the current projected local lattice even if the local processing is implemented by enumeration or sieving. The heuristic GBKZ analysis [15] does not model this dependence explicitly, using the Gaussian heuristic to express local quality as a function of the local rank alone. Our practical analysis cannot directly use this compression because the local processing has its own parameter and its output cannot be described by the local rank alone. We therefore introduce a practical local quality estimate to model this dependence.

The estimate combines information about the current projected input with the reduction available from inner BKZ. A local quality floor prevents overly optimistic estimates. At outer position j, define $\kappa_j = \min\{\beta_1, n-j+1\}$ and $n_j = j + \kappa_j - 1$. Let $w = (w_1, \ldots, w_n)$ denote the normalized log-GSO profile used as the state of the practical fixed-point recurrence, set $w_0 = 0$, and write $\Delta w_i := w_i - w_{i-1}$. We define the following three quantities for the practical local quality estimate.

Current profile estimate. For the projected block $B[j, n_j]$, define $s_j(w) = \Delta w_j - (1/\kappa_j)\sum_{i=j}^{n_j} \Delta w_i$ and $q_{\mathrm{state,pr}}(w; j) = 2^{s_j(w)}$. Here, $s_j(w)$ measures the deviation of the first profile increment from the block average. Smaller values represent a more favorable local state in the model.

Generic inner BKZ estimate. Let $\beta_{2,j} = \min\{\kappa_j, \beta_2\}$. Using $Q_{\mathrm{BKZ,pr}}$ from Definition 3, define $q_{\mathrm{gen,pr}}(\kappa_j, \beta_2) = Q_{\mathrm{BKZ,pr}}(\kappa_j, \beta_{2,j})$. This estimate does not depend on the current input state.

Local quality floor. Using $\sqrt{\widetilde{\gamma}_{\kappa_j}}$ from Definition 3, define $q_{\mathrm{floor,pr}}(\kappa_j) = \sqrt{\widetilde{\gamma}_{\kappa_j}}$.

We combine the state and generic estimates through a heuristic max–min rule. The minimum selects the more favorable estimate of local quality, and the maximum with $q_{\mathrm{floor,pr}}$ prevents overly optimistic estimates. This gives the following practical estimate.

Definition 7 (Practical local quality estimate) The practical local quality estimate for the effective local quality at position j is

$$\widehat{q}^{\mathrm{pr}}_{\mathrm{eff},j}(w) = \max\bigl\{q_{\mathrm{floor,pr}}(\kappa_j), \min\{q_{\mathrm{state,pr}}(w; j), q_{\mathrm{gen,pr}}(\kappa_j, \beta_2)\}\bigr\}.$$

Let $\ell_j(w) = \log \widehat{q}^{\mathrm{pr}}_{\mathrm{eff},j}(w)$. Given w, define $x = (x_1, \ldots, x_n)$ by $x_n = 0$ and, for $1 \le j \le n-1$,

$$x_j = \frac{\kappa_j}{\kappa_j - 1}\ell_j(w) + \frac{1}{\kappa_j - 1}\sum_{i=j+1}^{n_j} x_i.$$

Define $T_{\beta_1,\beta_2}(w) = x(I - Q)$. Since the local quality terms $\widehat{q}^{\mathrm{pr}}_{\mathrm{eff},j}(w)$ now depend on w, the GBKZ analysis based on fixed local quality parameters cannot be applied directly. We therefore establish the existence of a fixed point for T_{β_1,β_2} below.

Theorem 3 (Practical fixed-point quality model) *For fixed n, β_1, β_2, the map T_{β_1,β_2} has at least one fixed point. If w^* is such a fixed point, the corresponding modeled first-vector quality is $2^{w_1^*}$.*

Proof For each position j, the quantity $s_j(w)$ is affine in w. Hence $q_{\mathrm{state,pr}}(w; j) = 2^{s_j(w)}$ is continuous. The quantities $q_{\mathrm{gen,pr}}(\kappa_j, \beta_2)$ and $q_{\mathrm{floor,pr}}(\kappa_j)$ are independent of w. Therefore $\widehat{q}^{\mathrm{pr}}_{\mathrm{eff},j}(w)$ is continuous. Hence $\ell_j(w)$, the vector x defined by the recurrence, and $T_{\beta_1,\beta_2}(w)$ are continuous functions of w.

For fixed n, β_1, β_2, each $\ell_j(w)$ lies between $\log q_{\mathrm{floor,pr}}(\kappa_j)$ and $\log \max\{q_{\mathrm{floor,pr}}(\kappa_j), q_{\mathrm{gen,pr}}(\kappa_j, \beta_2)\}$. Hence there is a finite constant M, independent of w, that bounds all $\ell_j(w)$. Since the recurrence for x contains finitely many linear combinations of these values, $x(I - Q)$ is also bounded by a finite constant independent of w.

Moreover, the image of T_{β_1,β_2} lies in $\mathrm{span}(\mathbf{1})^{\perp}$ and is bounded. Choose a closed ball $K \subseteq \mathrm{span}(\mathbf{1})^{\perp}$ large enough to contain this image. Then K is nonempty, compact, and convex, and $T_{\beta_1,\beta_2}(K) \subseteq K$. By Brouwer's fixed-point theorem, T_{β_1,β_2} has a fixed point $w^* \in K$.

At a fixed point w^*, the local qualities used in the recurrence agree with those determined by the resulting state. As in the standard practical fixed-point baseline, w_1^* is the logarithm of the normalized first-vector quality. Hence the corresponding modeled first-vector quality is $2^{w_1^*}$. □

We evaluate the practical model for $n \in \{250, 260, \ldots, 350\}$. For $\kappa \leq 50$, we use known exact values of γ_κ where available and Blichfeldt's upper bounds otherwise. Following the blocksize range used by the heuristic GBKZ analysis [15], we require the standard BKZ blocksize and the inner blocksize to be at least 69. Thus the standard BKZ model uses $69 \leq \beta \leq n$, and the two-parameter model uses $69 \leq \beta_2 \leq \beta_1 \leq n$. We initialize $w^{(0)}$ with the standard BKZ practical fixed-point profile for (n, β_1) and apply the direct iteration $w^{(t+1)} = T_{\beta_1,\beta_2}(w^{(t)})$. The iteration stops when $\max_i |w_i^{(t+1)} - w_i^{(t)}| < 10^{-12}$ and is limited to 300 iterations. The iteration converged under this criterion for all parameter settings used in the comparisons.

For each standard BKZ blocksize β, we use its modeled cost $C^{\mathrm{pr}}_{\mathrm{BKZ}}(n, \beta)$ as a budget. Among all two-parameter choices satisfying $C^{\mathrm{pr}}_{\mathrm{sep}}(n, \beta_1, \beta_2) \leq C^{\mathrm{pr}}_{\mathrm{BKZ}}(n, \beta)$, we record the smallest modeled RHF. Thus "without additional modeled cost" means that the modeled two-parameter cost does not exceed the corresponding BKZ budget.

Both model figures use the common cutoff $\log_2 C \leq 100$, since 2^{100} already represents a large modeled cost. The same cutoff is applied to standard BKZ and the idealized construction at every dimension to display their quality–cost differences over a common cost range.

The fraction of standard BKZ blocksize settings admitting a two-parameter choice with lower modeled RHF and no additional modeled cost increases with dimension, reaching about 56% at $n = 350$ (Fig. 2). The best modeled RHF improvement also increases with dimension. The comparison includes standard BKZ blocksizes satisfying $\log_2 C \leq 100$.

Fig. 3 shows how the modeled RHF difference changes across the cost range. In the displayed results, the best modeled RHF improvement occurs in an intermediate cost range. At low modeled costs, the cost of the inner BKZ calls leaves little room to increase β_1, so there is little room to benefit from separating the two blocksizes. In the

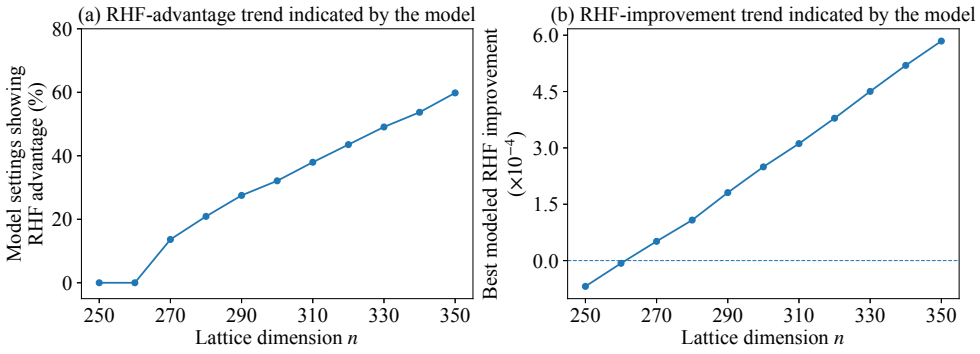

Fig. 2 Quality–cost behavior across dimensions indicated by the practical model. In Panel (a), the model settings correspond to the standard BKZ blocksize settings considered in the comparison. Panel (a) reports the fraction of these settings for which there exists a two-parameter choice with lower modeled RHF and no larger modeled cost than the corresponding BKZ setting. Panel (b) reports the best modeled RHF improvement under the same modeled cost budget. Both panels use $\log_2 C \leq 100$.

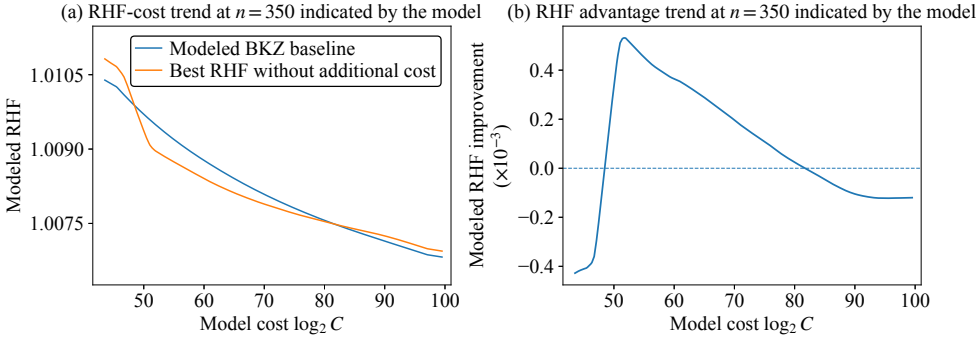

Fig. 3 Modeled quality–cost behavior across the cost range at $n = 350$. Panel (a) compares modeled RHF as a function of $\log_2 C$. Panel (b) shows the modeled RHF of standard BKZ minus that of the best two-parameter point within the same modeled cost budget. A positive value favors the two-parameter choice. The same $\log_2 C \leq 100$ cutoff is used. A light smoothing is applied only to the displayed curves; all parameter comparisons and selections use the unsmoothed model outputs.

intermediate range, a larger β_1 can extend the outer propagation range while a smaller β_2 limits the growth of the dominant local enumeration cost. At higher costs, standard BKZ can itself use larger blocksizes, and the modeled RHF difference narrows.

6 Experimental Evaluation

We now turn to experimental evaluation. The controlled experiment studies the role of the outer blocksize described in Section 3. The RHF–time evaluation studies whether the quality–cost behavior analyzed in Section 5 is also observed for the executable construction.

All experiments were conducted on a CPU server equipped with two Intel Xeon E5-2696 v4 processors at 2.20 GHz, with 44 physical cores, 88 logical cores, and 251 GiB of RAM. The server runs Ubuntu 22.04.5 LTS with Linux kernel 5.15.0. The implementation was written in Python 3.10.19 and used `fpylll` 0.6.4 [33] with the underlying `fplll` library [34]. All runs were performed on CPU only. The outer GSO computations used MPFR arithmetic with a precision of 212 bits. The explicit projected matrices used double precision, while the inner BKZ calls retained the library's default floating-point settings. We use RHF as the common measure of reduction quality. In the controlled experiment, we additionally record the log-GSO profile $G(B)$ described in Section 2. Reported running times exclude profile logging overhead.

6.1 Controlled Experiment

The purpose of this experiment is to examine the effect of the outer blocksize β_1 with the inner blocksize β_2 and the rank of the projected local lattice passed to inner BKZ fixed. Changing β_1 changes the range of basis positions involved in preprocessing and insertion updates, which can affect the input profiles at subsequent positions. This allows us to observe how these effects accumulate over successive outer tours and influence the resulting RHF.

We use random q-ary lattice bases with $n \in \{150, 200, 250, 300\}$, generated with `bits`=20 and $k = n/2$. For each dimension, we use three base seed values $\{123, 234, 345\}$, with the actual `FPLLL` seed equal to the base seed value plus n. The inner blocksize and the rank of the projected local lattice are fixed at $\beta_2 = 30$ and $m = 40$, while the outer blocksize ranges over $\beta_1 \in \{40, 45, 50, 55, 60, 65, 70, 75, 80\}$. This gives $4 \times 3 \times 9 = 108$ runs. For a fixed dimension and base seed value, the same input basis and algorithmic random seed are used across the different values of β_1.

The main change from the executable construction is that the rank of the projected local lattice is fixed to $m = 40$ instead of increasing with β_1. To accommodate this fixed rank, the transformation matrix used for the inner output is sized to the actual projected dimension, and the returned coefficient vector is padded with zeros before the unchanged outer postprocessing. The preprocessing, outer tour order, candidate acceptance test, lifting, insertion, postprocessing, and tail block handling are otherwise unchanged.

The inner BKZ calls use `BKZ.DEFAULT_STRATEGY`, including the default pruning, and are run without `AUTO_ABORT`, `MAX_TIME`, or `MAX_LOOPS`. The outer procedure also disables these stopping conditions and performs exactly three complete outer tours. We record the initial state and the state after each completed tour.

For the profile comparison, Fig. 4 uses the log-GSO profile $G(B)$. For tour t, define the profile change relative to $\beta_1 = 40$ by $\Delta G^{(t)}(\beta_1) = G^{(t)}(\beta_1) - G^{(t)}(40)$.

In Fig. 4(a), we plot $\Delta G^{(3)}(\beta_1)$ for $\beta_1 \in \{50, 60, 70, 80\}$ after the third outer tour. Since the four dimensions have different profile lengths, each profile is mapped to the normalized basis position $(i-1)/(n-1)$ and interpolated to a common grid before averaging. Each displayed curve is therefore averaged over four dimensions and three input bases per dimension. In Fig. 4(b), we fix $\beta_1 = 80$ and compare $\Delta G^{(1)}(80)$ and $\Delta G^{(3)}(80)$. A light smoothing is applied after averaging only for visualization. All RHF values and comparisons use the unsmoothed experimental data.

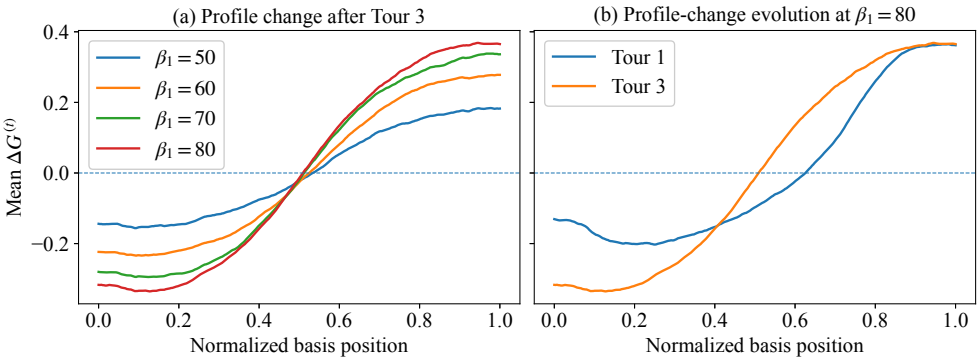

Fig. 4 Log-GSO profile changes in the controlled experiment with fixed $\beta_2 = 30$ and fixed projected local lattice rank $m = 40$. Panel (a) shows the mean $\Delta G^{(3)}(\beta_1)$ for $\beta_1 \in \{50, 60, 70, 80\}$. Panel (b) shows the mean $\Delta G^{(t)}(80)$ after Tours 1 and 3. The curves are averaged across four dimensions and three input bases per dimension. Light smoothing is applied after averaging only for visualization.

Fig. 4(a) shows that the profile difference relative to $\beta_1 = 40$ generally becomes more pronounced as β_1 increases. The leading part of the profile decreases while the trailing part increases, showing a redistribution of the profile. Fig. 4(b) shows that the profile change is already visible after the first outer tour and continues to evolve through the third tour, where it affects more leading positions. Thus, changing β_1 changes the profile evolution even though the inner blocksize and projected local lattice rank are fixed.

We next examine whether this profile change leads to a difference in reduction quality over successive outer tours. For tour t, define the RHF improvement relative to $\beta_1 = 40$ by $\Delta\mathrm{RHF}_t(\beta_1) = \mathrm{RHF}_t(40) - \mathrm{RHF}_t(\beta_1)$. A positive value therefore means that the corresponding choice of β_1 yields a lower RHF. Fig. 5 reports this quantity after tours 1 and 3 for each dimension, with every point averaged over the three input bases.

After the first tour, the RHF differences are small even though Fig. 4 already shows a visible profile change. After three tours, positive RHF improvements appear clearly, and larger values of β_1 generally produce larger improvements, although the relation is not strictly monotone. Together, the two figures show that, with the inner blocksize and the rank of the projected local lattice fixed, changing the outer blocksize changes the profile evolution and that this effect can accumulate over successive outer tours into an RHF improvement. Thus, the controlled experiment provides experimental support for the conclusion in Section 3 that β_1 and β_2 play different roles.

6.2 RHF–Time Evaluation of the Executable Construction

We now evaluate the executable construction of Section 4. We use BKZ 2.0 as the practical baseline. BKZ 2.0 is the standard optimized BKZ-type reduction procedure implemented in `fplll` and exposed through `fpylll`. Our implementation is built on the same `fplll`/`fpylll` infrastructure and uses the same enumeration-based local computation, BKZ strategies, GSO maintenance, and compatible insertion and update

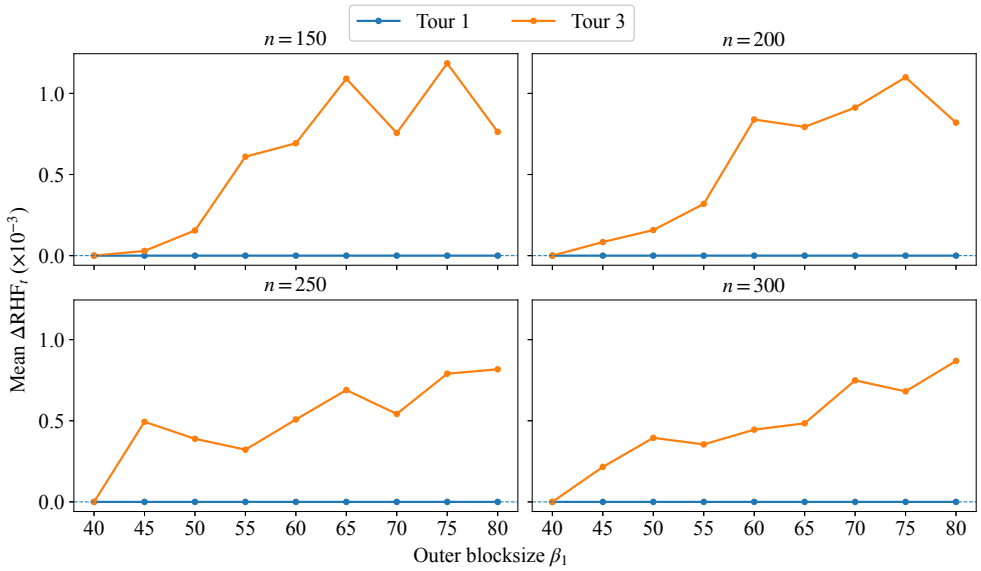

Fig. 5 Mean $\Delta\mathrm{RHF}_t(\beta_1)$ in the controlled experiment. The four panels correspond to $n \in \{150, 200, 250, 300\}$. Each panel reports the result after Tours 1 and 3 for $\beta_1 \in \{40, 45, 50, 55, 60, 65, 70, 75, 80\}$, averaged over three input bases. Positive values indicate a lower RHF than the $\beta_1 = 40$ reference.

routines. This provides a common implementation environment for comparing BKZ 2.0 with the executable construction.

We use two types of input bases. The first consists of random q-ary lattice bases with dimensions $n \in \{150, 200, 250, 300\}$, using the same `bits` $= 20$, $k = n/2$, and three base seed values $\{123, 234, 345\}$ as above. The second consists of SVP Challenge bases [25] with dimensions $n \in \{150, 180, 210, 240\}$. For each SVP Challenge dimension, we use the three challenge bases with seeds $\{0, 1, 2\}$.

For the evaluation reported here, we use BKZ 2.0 blocksizes $\beta \in \{45, 50, 55\}$ and a two-parameter grid with $\beta_1 \in \{60, 65, 70, 75, 80\}$ and $\beta_2 \in \{40, 45, 50\}$. The grid is chosen so that the inner blocksize range overlaps the BKZ 2.0 baseline range and extends below it, while the outer blocksize range lies above it. We set the largest BKZ 2.0 baseline blocksize to $\beta = 55$ because the running time is already substantial at this blocksize for the larger tested dimensions, making a complete extension to larger blocksizes considerably more expensive. All 15 two-parameter choices satisfy $\beta_2 < \beta_1$. For each input family, the reported grid therefore contains $4 \times 3 \times (3 + 15) = 216$ runs, for a total of 432 reported runs.

Both BKZ 2.0 and the outer layer of the executable construction use `BKZ.DEFAULT_STRATEGY` and the standard `AUTO_ABORT` criterion. The inner BKZ calls use the default strategy and retain their standard `AUTO_ABORT` behavior. To allow each reduction to proceed as far as possible under these automatic stopping conditions, we do not use the `MAX_TIME` or `MAX_LOOPS` options. A run terminates after a clean outer

tour or when the standard `AUTO_ABORT` criterion is met. All runs begin with the same initial LLL step for the corresponding implementation. The comparison below uses the final RHF and total running time.

For a fixed input basis and a BKZ 2.0 blocksize β, we compare each of the 15 two-parameter runs with the corresponding BKZ 2.0 run. We count a comparison as favorable when $\mathrm{RHF}_{\mathrm{sep}} < \mathrm{RHF}_{\mathrm{BKZ}}$ and $T_{\mathrm{sep}} < T_{\mathrm{BKZ}}$. For every fixed pair (n, β), the reported fraction is computed over all 15 two-parameter choices and all three input bases, giving 45 comparisons per point.

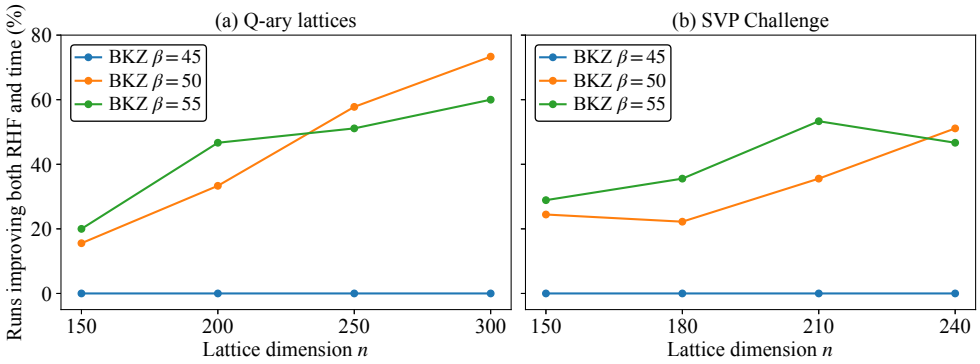

Fig. 6 Fraction of two-parameter runs satisfying both $\mathrm{RHF}_{\mathrm{sep}} < \mathrm{RHF}_{\mathrm{BKZ}}$ and $T_{\mathrm{sep}} < T_{\mathrm{BKZ}}$ for the corresponding BKZ 2.0 baseline. Panel (a) shows the random q-ary bases and panel (b) the SVP Challenge bases. Each point represents 15 two-parameter choices for each of three input bases, giving 45 comparisons in total.

Fig. 6 shows a clear difference between the three BKZ 2.0 baseline blocksizes. For $\beta = 45$, the fraction is 0% in every displayed dimension for both input families. No comparison satisfies both conditions at this baseline blocksize. For $\beta = 50$ and $\beta = 55$, favorable comparisons occur in every displayed dimension, and the fraction is generally larger at higher dimensions. On the q-ary bases, the fractions for $\beta = 50$ and $\beta = 55$ increase from 15.56% and 20.00% at $n = 150$ to 73.33% and 60.00% at $n = 300$, respectively. On the SVP Challenge bases, the trend is not strictly monotone, but the fractions reach 51.11% for $\beta = 50$ at $n = 240$ and 53.33% for $\beta = 55$ at $n = 210$.

Fig. 6 summarizes the full reported parameter grid, but it does not show the magnitude of the differences in RHF and running time. Table 2 therefore reports two representative choices of (β_1, β_2) using the mean RHF and running time over the three input bases at each dimension. The first compares BKZ 2.0 with $\beta = 50$ against $(\beta_1, \beta_2) = (75, 40)$. The relation $40 < 50 < 75$ makes the separation between the inner and outer blocksizes explicit and provides a direct example of reducing the inner blocksize while increasing the outer blocksize. The second compares BKZ 2.0 with $\beta = 55$ against $(\beta_1, \beta_2) = (60, 45)$. This choice directly illustrates the observed time saving.

The choice $(75, 40)$ achieves a lower mean RHF in every reported case and, except for the q-ary case at $n = 150$, also gives a lower mean running time than BKZ 2.0 with

$\beta = 50$. The choice $(60, 45)$ reduces the mean running time in every reported case. Its mean RHF is lower in all rows except the q-ary case at $n = 150$, where the two values differ by about 10^{-6}. These representative results also reflect the dimension trend shown in Fig. 6. At the larger displayed dimensions, both selected choices improve RHF and running time over their respective BKZ 2.0 baselines.

Table 2 Representative choices of (β_1, β_2). RHF and running time are means over the three input bases at each dimension. In the RHF and Time columns, each entry is reported as BKZ 2.0 / two-parameter BKZ. The time ratio is the mean running time of the two-parameter BKZ divided by the mean running time of BKZ 2.0, so a value below one means that the two-parameter BKZ is faster. In the Input column, SVP denotes SVP Challenge bases.

Input	n	BKZ β	(β_1, β_2)	RHF	Time [s]	Time ratio
q-ary	150	50	$(75, 40)$	1.011907 / 1.011529	2733.5 / 3546.1	1.297
q-ary	150	55	$(60, 45)$	1.011406 / 1.011407	5119.0 / 3434.1	0.671
q-ary	200	50	$(75, 40)$	1.011963 / 1.011624	9870.4 / 8481.4	0.859
q-ary	200	55	$(60, 45)$	1.011753 / 1.011553	13911.0 / 7742.3	0.557
q-ary	250	50	$(75, 40)$	1.012084 / 1.011732	30901.1 / 22087.1	0.715
q-ary	250	55	$(60, 45)$	1.011718 / 1.011647	37632.9 / 17573.6	0.467
q-ary	300	50	$(75, 40)$	1.012143 / 1.011826	65016.4 / 36348.8	0.559
q-ary	300	55	$(60, 45)$	1.011804 / 1.011658	96530.1 / 39297.9	0.407
SVP	150	50	$(75, 40)$	1.011842 / 1.011752	3439.8 / 3322.0	0.966
SVP	150	55	$(60, 45)$	1.011654 / 1.011418	4557.7 / 3085.1	0.677
SVP	180	50	$(75, 40)$	1.011862 / 1.011687	5961.8 / 4555.4	0.764
SVP	180	55	$(60, 45)$	1.011604 / 1.011561	9031.2 / 5792.2	0.641
SVP	210	50	$(75, 40)$	1.012044 / 1.011614	12306.2 / 9624.5	0.782
SVP	210	55	$(60, 45)$	1.011616 / 1.011477	25962.5 / 12118.1	0.467
SVP	240	50	$(75, 40)$	1.012095 / 1.011840	25474.6 / 16883.8	0.663
SVP	240	55	$(60, 45)$	1.011743 / 1.011526	35080.6 / 17275.1	0.492

The practical branch in Section 5 analyzes the idealized construction using modeled RHF and modeled cost. The experiment above evaluates the executable construction using measured RHF and actual running time. The model and experiments show qualitatively similar trends with dimension.

6.3 Comparison with Other BKZ Variants

In addition to the detailed comparison with BKZ 2.0 above, we make supplementary comparisons based on published results for the BKZ variants introduced in 2020 [16] and 2021 [17]. The two works of Albrecht et al. represent a leading line of enumeration-based BKZ, with cost estimates validated through simulation and experiment.

Comparison with the BKZ variant in [16]. This variant uses extended preprocessing beyond the enumeration range. The public simulation data accompanying the full version of [17] provide cost estimates for this variant with $\alpha = \alpha' = 1$. We select $c = 0.25$, originally recommended in [16], and $c = 0.15$, which was shown in [17, Remark 1 and Fig. C.1] to have a lower cost. We compare average

costs per tour using their published local cost estimates and our measured average tour times. The tabulated costs in these data start at $k = 50$, for which the least expensive strategy is the standard case $(\alpha, c) = (1, 0)$, without relaxed enumeration or extended preprocessing. Since 55 is the only tested BKZ 2.0 baseline blocksize above 50 in our experiments, we compare at $k = 55$ on random q-ary lattices. The target RHF is $\delta_{55} = h(55)^{1/54} \approx 1.01176534$. Our measured cost ratio is $r_{\mathrm{PS}} = \overline{T}_{\mathrm{PS}}/\overline{T}_{\mathrm{BKZ\,2.0},55}$, where $\overline{T}$ averages tour times within each run and then over the three inputs. For this variant, we estimate the cost per tour relative to its standard case as $r_{2020} = (C_{2020}/C_{\mathrm{base}})(n - m + 1)/(n - 55 + 1)$, where m is the window size. Here C_{2020} and C_{base} are the published local costs, including preprocessing, of the selected strategy and the standard case, respectively. Both are estimated at $k = 55$ by linear interpolation on a logarithmic scale between their respective values at $k = 54$ and 56. This approximation counts full windows and omits shorter tail windows. For the representative choice $(\beta_1, \beta_2) = (60, 45)$, our mean RHFs meet the target δ_{55} in all four tested dimensions. At $n = 150, 200, 250, 300$, r_{PS} is 38.7%, 52.5%, 63.1%, and 65.8% lower than r_{2020} for $c = 0.25$, respectively. The corresponding reductions for $c = 0.15$ are 36.6%, 49.8%, 60.6%, and 63.3%.

Comparison with the BKZ variant in [17]. We use the same comparison procedure as for the variant in [16]. At this target, the published strategy with $\alpha = 1.05$, $c = 0$, $\alpha' = 1$, and $m = 68$ has the lowest normalized tour cost among those considered and a modeled RHF of approximately 1.011696. All four mean RHFs of our fixed configuration meet this stronger target. Under the same normalization, our normalized tour cost ratio is lower by 17.2%, 35.6%, and 40.3% at $n = 200, 250, 300$, respectively, and higher by 6.3% at $n = 150$.

7 Conclusion

This paper studied parameter separation in BKZ through a general framework with separate parameters for outer propagation and the dominant local computation. With suitable local processing choices and corresponding adaptations to the outer operations, the framework covers the local parameter settings of existing BKZ variants. We gave idealized and executable constructions using inner BKZ with a smaller blocksize while retaining a larger outer propagation range. For the idealized construction, we derived a worst-case quality bound and developed a practical fixed-point model that accounts for the dependence of inner BKZ quality on its projected input and jointly estimates local quality and the global basis profile.

Experiments on random q-ary bases up to dimension 300 and SVP Challenge bases up to dimension 240 demonstrated simultaneous improvements in RHF and running time over BKZ 2.0 for many tested parameter and input combinations. Joint improvements were generally more frequent at higher dimensions within the tested ranges. Supplementary comparisons using published cost estimates also identified cases with lower normalized tour cost ratios than those of selected strategies for the BKZ variants in [16, 17] at the reported quality targets. Within the tested ranges, the experiments show that weaker local processing can improve the overall quality–cost tradeoff when combined with a larger outer propagation range.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China (62572037), and in part by Beijing Natural Science Foundation (4242022).

References

[1] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, pp. 99–108. ACM, New York (1996). https://doi.org/10.1145/237814.237838

[2] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM **56**(6) (2009) https://doi.org/10.1145/1568318.1568324 . Art. no. 34

[3] Schnorr, C.-P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming **66**, 181–199 (1994) https://doi.org/10.1007/BF01581144

[4] Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_1

[5] Wang, L., Xia, W., Wang, G., Wang, B., Gu, D.: Improved Pump and Jump BKZ by sharp simulator. Cryptology ePrint Archive, Paper 2022/468, https://eprint.iacr.org/2022/468 (2022)

[6] Ducas, L., Pulles, L.N., Stevens, M.: Towards a modern LLL implementation. In: Hanaoka, G., Yang, B.-Y. (eds.) ASIACRYPT 2025. LNCS, vol. 16247, pp. 65–99. Springer, Singapore (2026). https://doi.org/10.1007/978-981-95-5099-9_3

[7] Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_3

[8] Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_21

[9] Albrecht, M.R., Curtis, B.R., Deo, A., Davidson, A., Player, R., Postlethwaite, E.W., Virdia, F., Wunderer, T.: Estimate all the {LWE, NTRU} schemes! In: Catalano, D., De Prisco, R. (eds.) SCN 2018. LNCS, vol. 11035, pp. 351–367. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_19

[10] Ducas, L., Engelberts, L., de Perthuis, P.: Predicting module-lattice reduction. In: Hanaoka, G., Yang, B.-Y. (eds.) ASIACRYPT 2025. LNCS, vol. 16247, pp. 133–166. Springer, Singapore (2026). https://doi.org/10.1007/978-981-95-5099-9_5

[11] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber: Algorithm specifications and supporting documentation, version 3.02. Round-3 submission to the NIST Post-Quantum Cryptography Standardization Project, https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf (2021)

[12] Bai, S., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: Algorithm specifications and supporting documentation, version 3.1. Round-3 submission to the NIST Post-Quantum Cryptography Standardization Project, https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf (2021)

[13] Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J.M., Lichtinger, J., Miller, C.A., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., Liu, Y.-K.: Status report on the third round of the NIST Post-Quantum Cryptography Standardization Process. NIST Internal Report 8413-upd1. National Institute of Standards and Technology (2022). https://doi.org/10.6028/NIST.IR.8413-upd1

[14] Hanrot, G., Pujol, X., Stehlé, D.: Analyzing blockwise lattice algorithms using dynamical systems. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 447–464. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_25

[15] Li, J., Nguyen, P.Q.: A complete analysis of the BKZ lattice reduction algorithm. Journal of Cryptology **38** (2025) https://doi.org/10.1007/s00145-024-09527-0 . Art. no. 12

[16] Albrecht, M.R., Bai, S., Fouque, P.-A., Kirchner, P., Stehlé, D., Wen, W.: Faster enumeration-based lattice reduction: Root Hermite factor $k^{1/(2k)}$ time $k^{k/8+o(k)}$. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12171, pp. 186–212. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_7

[17] Albrecht, M.R., Bai, S., Li, J., Rowell, J.: Lattice reduction with approximate enumeration oracles: Practical algorithms and concrete performance. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12826, pp. 732–759. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84245-1_25

[18] Ducas, L.: Shortest vector from lattice sieving: A few dimensions for free. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 125–145. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_5

[19] Laarhoven, T., Mariano, A.: Progressive lattice sieving. In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018. LNCS, vol. 10786, pp. 292–311. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_14

[20] Albrecht, M.R., Ducas, L., Herold, G., Kirshanova, E., Postlethwaite, E.W.,

Stevens, M.: The general sieve kernel and new records in lattice reduction. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 717–746. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_25

[21] Aono, Y., Wang, Y., Hayashi, T., Takagi, T.: Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 789–819. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_30

[22] Micciancio, D., Walter, M.: Practical, predictable lattice basis reduction. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 820–849. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_31

[23] Yasuda, M., Nakamura, S., Yamaguchi, J.: Analysis of DeepBKZ reduction for finding short lattice vectors. Designs, Codes and Cryptography **88**(10), 2077–2100 (2020) https://doi.org/10.1007/s10623-020-00765-4

[24] Sato, A., Yasuda, M.: A new BKZ-type reduction with provable termination and development of its self-dual variant. Japan Journal of Industrial and Applied Mathematics **43** (2026) https://doi.org/10.1007/s13160-026-00799-6 . Art. no. 42

[25] Schneider, M., Gama, N.: SVP Challenge. TU Darmstadt Lattice Challenge, https://www.latticechallenge.org/svp-challenge/

[26] Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, pp. 193–206. ACM, New York (1983). https://doi.org/10.1145/800061.808749

[27] Fincke, U., Pohst, M.E.: Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Mathematics of Computation **44**(170), 463–471 (1985) https://doi.org/10.1090/S0025-5718-1985-0777278-8

[28] Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_13

[29] Li, J., Nguyen, P.Q.: Approximating the densest sublattice from Rankin's inequality. LMS Journal of Computation and Mathematics **17**(A), 92–111 (2014) https://doi.org/10.1112/S1461157014000333

[30] Aggarwal, D., Espitau, T., Peters, S., Stephens-Davidowitz, N.: Recursive lattice reduction—a framework for finding short lattice vectors. In: Bercea, I.-O., Pagh, R. (eds.) 2025 Symposium on Simplicity in Algorithms (SOSA), pp. 533–573. SIAM, Philadelphia, PA (2025). https://doi.org/10.1137/1.9781611978315.40

[31] Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**, 515–534 (1982) https://doi.org/10.1007/BF01457454

[32] Albrecht, M.R.: Lattice Estimator: CheNgu12 cost model documentation. Online documentation, https://lattice-estimator.readthedocs.io/en/latest/_apidoc/estimator.reduction/estimator.reduction.CheNgu12.html. Accessed 29 October 2026

[33] The FPLLL Development Team: fpylll, a Python wrapper for the fplll lattice reduction library, Version 0.6.4. https://github.com/fplll/fpylll (2025)

[34] The FPLLL Development Team: fplll, a lattice reduction library, Version 5.5.0. https://github.com/fplll/fplll (2023)