Fast Practical Lattice Reduction through Iterated Compression

Keegan Ryan and Nadia Heninger

University of California, San Diego
kryan@ucsd.edu,nadiah@cs.ucsd.edu

Abstract. We introduce a new lattice basis reduction algorithm with approximation guarantees analogous to the LLL algorithm and practical performance that far exceeds the current state of the art. We achieve these results by iteratively applying precision management techniques within a recursive algorithm structure and show the stability of this approach. We analyze the asymptotic behavior of our algorithm, and show that the heuristic running time is $O(n^{\omega}(C+n)^{1+\varepsilon})$ for lattices of dimension n, $\omega \in (2,3]$ bounding the cost of size reduction, matrix multiplication, and QR factorization, and C bounding the log of the condition number of the input basis B. This yields a running time of $O\left(n^{\omega}(p+n)^{1+\varepsilon}\right)$ for precision $p = O(\log \|B\|_{max})$ in common applications. Our algorithm is fully practical, and we have published our implementation. We experimentally validate our heuristic, give extensive benchmarks against numerous classes of cryptographic lattices, and show that our algorithm significantly outperforms existing implementations.

1 Introduction

Lattice basis reduction is a fundamental technique in cryptanalysis. The celebrated LLL algorithm [40] achieves a $2^{O(n)}$ approximation factor for lattice reduction in time $O(n^{5+\varepsilon}(p+\log n)^{2+\varepsilon})$ for a lattice of dimension n with entries of size $p = O(\log \|B\|_{max})$. This is polynomial time, but the large exponents ensure that this algorithm quickly becomes infeasible in practice for even moderately sized lattices.

The current gold standard lattice basis reduction algorithm used in practice is the L^2 algorithm [49] implemented in fpLLL [57], which improves the dependence on p by carefully managing precision for a runtime of $O(n^{4+\varepsilon}(p+\log n)(p+n))$. Current implementations take advantage of hardware floating-point support and are fast up to a few hundred dimensions, but the running time again becomes an obstacle beyond this point.

A separate line of work reduces the running time dependence on n through developing reduction algorithms with a recursive structure [36,45]. These algorithms have impressive performance, but have practical drawbacks in only outputting a single short vector, and a lack of accessible implementations.

Kirchner, Espitau, and Fouque [33] combined these approaches by giving an algorithm with a recursive structure that decreases the working precision as the lattice basis is reduced. They report impressive performance numbers on lattice bases of high dimension and precision, and claim a running time of $\tilde{O}(n^\omega C)$ for $C > \log(\|B\|\|B^{-1}\|)$ and ω equaling the matrix multiplication exponent. These results are based on a very strong heuristic assumption about linear regressions of log-Gram-Schmidt norms related to the Geometric Series Assumption (GSA). They use this assumption to argue that the required precision decreases exponentially throughout execution of their algorithm.

Heuristic precision issues. Unfortunately, the heuristic of [33] fails for large classes of cryptanalytically relevant lattices that include some of the most canonical applications of the LLL algorithm in cryptography: NTRU [16], Coppersmith lattices for solving low-degree polynomials modulo integers [14,29], and factoring with partial information [30]. These deviations from the GSA result in either unfavorable running times or computational errors due to insufficient precision. In practice, their implementation simply fails to reduce many of these lattices.

Our contribution. In this work, we give a new recursive variant of the LLL algorithm with a novel iterative strategy for managing precision, and our variant is extremely fast in practice. In addition, our algorithm parallelizes naturally. We benchmark the performance of our algorithm against a diverse collection of families of lattices of theoretical and practical interest to showcase how the wildly differing structures of these lattices exhibit differing behaviors during lattice reduction. We find that our algorithm significantly outperforms fpLLL in every test case, and the algorithm of [33] on almost all families of lattices. Our test cases include lattice bases of dimension up to 4889 with entries of size 8.5Mb, which we were able to reduce in 46 core-hours.

Existing analysis tools are incompatible with our algorithm, so we develop new theoretical results to explain the behavior of and increase confidence in the stability and correctness of our approach. Using significantly weaker heuristic assumptions, our asymptotic running time parallels the claimed running time of [33], and the approximation guarantees for the reduced basis match those of the LLL algorithm.

We are making our implementation[1] available to the community with the aim for it to be practical drop-in replacement for fpLLL.

1.1 Overview of Techniques

Iterated Compression. We develop several new tools to support the implementation and analysis of our algorithm. Our algorithm uses a new metric for lattice reduction distinct from prior work, which we call the *drop* of a lattice basis. Our new definition is used analogously to the Lovász condition of traditional LLL reduction. We say a basis B of rank n is α-lattice-reduced if it is size-reduced and $\mathrm{drop}(B) \leq \alpha n$.

[1] Our implementation is available at https://github.com/keeganryan/flatter

In addition, we apply a type of lattice basis "compression", which transforms a lattice basis into one with similar geometric properties but with smaller entries and superior numerical stability. This compression is largely akin to [54], although our analysis is new.

Our algorithm is summarized by the pseudocode in Algorithm 1.

Algorithm 1: `Reduce` (sketch)

```
  Input  : Lattice basis $B$ of rank $n$ and reduction quality $\alpha$
  Output: Unimodular matrix $U$ such that $BU$ is $\alpha$-lattice-reduced
1 while $drop(B) > \alpha n$ do
2     for Proj. sublattice indices $[i:j]$ in $\{[\frac{n}{4} : \frac{3n}{4}], [0 : \frac{n}{2}], [\frac{n}{2} : n]\}$ do
3         $U_k \leftarrow$ Reduce($B_{[i:j]}, \alpha'$) for improved reduction quality $\alpha' < \alpha$.
4         $B \leftarrow$ Compress($BU_k$) to decrease entry size
5 return $U \leftarrow$ accumulation of all previous unimodular transformations
```

While the general description of the algorithm is simple and reminiscent of prior algorithms, the details and its analysis are not. The design of the compression function is critical to achieving numerical stability in practice, and, like [54], it is more sophisticated than simply taking the most significant bits. Unlike [54], our algorithm iterates the compression process, so extra care is needed to ensure accumulating rounding errors remain manageable. We develop first-order and asymptotic results using perturbation theory to bound the entry size of compressed bases and the accuracy of iterated compression.

Using heuristic assumptions, we prove the following running time for our algorithm, given here in a simplified form.

Theorem 1 (Simplified). *Let B be an integer lattice basis of dimension n and let $C > \log(\|B\|\|B^{-1}\|)$. If the running time of size reduction, matrix multiplication, and QR factorization has a $O(n^\omega)$ dependence on the dimension for some $\omega \in (2, 3]$, and our heuristic assumptions are true, then the running time of our algorithm is*

$$O\left(n^\omega (C + n)^{1+\varepsilon}\right).$$

For B that are also upper triangular and size-reduced, as is common in cryptanalytic attacks, this is $O\left(n^\omega (\log \|B\|_{max} + n)^{1+\varepsilon}\right)$.

A deep dive into profiles. We analyze the behavior of our algorithm without relying upon the Geometric Series Assumption (GSA), which is clearly false for major classes of lattices of cryptanalytic importance, including Coppersmith, NTRU, LWE, and hidden number problem lattices. We do this by studying the evolution of the *profile*, or the log-norms of the Gram-Schmidt vectors, during lattice reduction. Unlike [33], which assumed via heuristic that the profile closely followed a linear trend of known slope, our analysis considers all possible profile shapes and their evolutions. We use the basis drop, as computed from the profile, to relate the change in lattice potential to the change in required precision.

In addition to its utility in analyzing the behavior of our algorithm, our redefinition of reduction quality in terms of the basis drop leads to bases with the same properties as LLL-reduced bases, and we achieve results analogous to [47, Theorem 9]. In practice, our algorithm returns approximations of equivalent quality to LLL.

Theorem 2. *Let B be a α-lattice-reduced rank-n basis satisfying our new definition of reduction quality. Let $\vec{b}_i^*$ denote the i^{th} Gram-Schmidt vector and $\lambda_i(B)$ denote the i^{th} successive minimum of the lattice spanned by B. Then B satisfies*

1. $\|\vec{b}_1\| \leq 2^{\alpha n}(\det B)^{1/n}$.
2. $\|\vec{b}_n^*\| \geq 2^{-\alpha n}(\det B)^{1/n}$.
3. *For all* $i \in \{1, \ldots, n\}$, $\|\vec{b}_i\| \leq 2^{\alpha n + O(n)}\lambda_i(B)$.
4. $\|\vec{b}_1\| \times \cdots \times \|\vec{b}_n\| \leq 2^{\alpha n^2 + O(n^2)} \det B$.

A focus on applications. Finally, we benchmark our algorithm against a wide variety of lattice families. We sourced lattice constructions from numerous cryptanalysis papers to comprehensively analyze the behavior of our implementation on lattices of research interest. Our selected test cases exhibit profiles that evolve in wildly different ways, and our implementation significantly outperforms existing tools on these lattices.

The main goal of this work is to produce a fully practical, implementable algorithm that outperforms existing lattice reduction implementations, and is intended to be used in practice. To that end, our theoretical results characterize the behavior of the algorithm and give confidence in the numerical stability of our approach with the help of weaker heuristic assumptions than prior work. We experimentally justify our heuristics.

2 Background

2.1 Notation

In this work, we represent bases in column notation, with the columns of B being basis vectors of the lattice. We index the vectors from 1 to n and denote the i^{th} vector by $\vec{b}_i$. We use $\|\cdot\|$ to refer to the Euclidean norm for vectors and spectral norm for matrices, unless otherwise specified. We use log to denote the base-2 logarithm. We use $\kappa(B) = \|B\|\|B^{-1}\|$ to represent the condition number of matrix B. We use $B_{[i:j]}$ for $0 \leq i < j \leq n$ to denote the projected sublattice of rank $j - i$ formed by taking the lattice generated by the first j vectors projected orthogonally to the first i vectors. For brevity, we may use the term "sublattice" when we refer to a projected sublattice, as our algorithm does not consider sublattices in the non-projected sense.

2.2 History of Lattice Reduction Algorithms

The running time to reduce a lattice basis B is typically given in terms of the dimension n of the lattice and the size of the largest basis vector $\beta = \log \max_i \|\vec{b}_i\| = O(\log \|B\|_{max} + \log n)$. The goal of lattice reduction is to find a reduced basis where the n output vectors approximate the optimally smallest basis by an exponential factor $2^{O(n)}$. The original LLL algorithm [40] terminates in $O(n^{5+\varepsilon}\beta^{2+\varepsilon})$ bit operations where $\varepsilon > 0$ allows fast integer arithmetic. Two main lines of work have made progress towards reducing the running time of lattice reduction.

One line of work reduces the dependence on the dimension by using a recursive algorithm structure. Koy and Schnorr [36] proposed minimizing the cost by iteratively reducing overlapping sublattices, and the resulting algorithm required $O(n^{3+\varepsilon}\beta^{2+\varepsilon})$ bit operations. However, the output of the algorithm only bounded the length of the first vector by a factor of $2^{O(n \log n)}$ and made no guarantees on the other vectors. While their reported performance is impressive, to our knowledge the algorithm has not been implemented since. The recursive reduction idea was improved upon by Neumaier and Stehlé [45], achieving a proven runtime of $O(n^{4+\varepsilon}\beta^{1+\varepsilon})$ and bounding the first vector by a factor of $2^{O(n)}$. This algorithm uses exact precision, so the algorithm may need to compute with integers of bit-size $O(n\beta)$. For this reason, the algorithm is not considered practical, and to our knowledge, it has never been implemented.

A second line of work reduces the running time by carefully managing precision [10,43]. The L^2 algorithm of Nguyen and Stehlé [49] takes time $O(n^{4+\varepsilon}\beta(n+\beta))$ and returns a basis of essentially the same quality as LLL. Essentially, the L^2 algorithm progressively reduces the first k basis vectors and observes that $O(k)$ bits suffice to represent the reduced partial basis. This algorithm is implemented in fpLLL, is fast, and is the tool most commonly used in practice. The $\tilde{L}^1$ algorithm [51] uses numerical stability results to reduce the runtime to $O(n^{5+\varepsilon}\beta + n^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$ and achieves essentially the same reduction quality as LLL. Here, ω is an exponent for matrix multiplication. However, this algorithm is also considered impractical and to our knowledge has not been implemented. An alternative approach of Bi et al. [5] and Saruchi et al. [54] improves lattice reduction running time in practice by reducing the bit sizes of entries to form an approximate basis with smaller vector precision $\beta' < \beta$, then applying lattice reduction algorithms to reduce the smaller basis.

It has long been a goal to unify these two lines of research, but there are many challenges to doing so. Kirchner, Espitau, and Fouque [33] recently published an algorithm that has a recursive structure and also decreases the working precision as the lattice basis becomes more reduced. Their algorithm is fast in practice for certain classes of bases, and they claim a running time of $\tilde{O}(n^\omega C)$ for $C > \log(\|B\|\|B^{-1}\|)$ by using strong heuristics to bound the necessary precision at each step. Unfortunately, their heuristic assumptions do not hold for important classes of lattice bases like NTRU and Coppersmith lattices, and their implementation does not work on these lattices in practice.

2.3 Lattice Reduction Basics

Many algorithms based on LLL operate on the Gram-Schmidt orthogonalization (GSO) of a lattice basis B, or the closely related QR-factorization which represents $B = QR$ as the product of an orthogonal matrix Q and upper triangular R. For a Gram-Schmidt vector $\vec{b}_i^*$, note that we have $\|\vec{b}_i^*\| = |R_{i,i}|$. Recursive lattice reduction algorithms frequently consider projected sublattices; that is, the lattice generated by vectors $\vec{b}_{i+1}, \dots, \vec{b}_j$ projected onto the vector space orthogonal to $\vec{b}_1, \dots, \vec{b}_i$. Using the upper triangular Gram-Schmidt coefficient matrix or R-factor, such a projected sublattice basis is easily computed from a block matrix of dimension n_2 along the diagonal.

The basic steps of the LLL algorithm are a swap operation on neighboring vectors $\vec{b}_i$ and $\vec{b}_{i+1}$ followed by a size-reduction step to ensure the new vector at index $i+1$ is small relative to indices 1 through i. This size reduction step ensures the GSO coefficients are not too large. The process repeats until the reduction criteria are satisfied and the algorithm terminates.

Variants of the LLL algorithm have a similar structure. Recursive algorithms determine a projected sublattice basis B_{sub} of dimension n_2 from R, and use a lattice reduction algorithm to find unimodular $U \in \mathbb{Z}^{n_2 \times n_2}$ such that $B_{sub}U$ is reduced. This U is applied to n_2 columns of the lattice of dimension n, and the Gram-Schmidt norms $\|\vec{b}_k^*\|, \dots, \|\vec{b}_{k+n_2}^*\|$ are changed in the exact same way as the Gram-Schmidt norms of B_{sub}. In essence, recursive algorithms are more efficient because they avoid having to update the full basis, and instead batch together many swapping and size-reduction operations in dimension n_2 into a single efficient update U to apply to dimension n. We say an algorithm is LLL-like if it generates update matrices U through a combination of neighbor swaps and size reductions.

There are multiple definitions of size reduction in the literature. We will not restrict ourselves to a particular one, and instead allow any definition that satisfies the following property:

Definition 1 (Size reduction). *Let B be a basis of rank n and let $B = QDM$ be the Gram-Schmidt orthogonalization where Q is orthogonal, $D = diag(\|\vec{b}_1^*\|, \dots, \|\vec{b}_n^*\|)$, and M is unitriangular. B is* size-reduced *if $\|M\| = 2^{O(n)}$ and $\|M^{-1}\| = 2^{O(n)}$.*

Note that then $\|B\| = (\max_i \|\vec{b}_i^*\|)2^{O(n)}$ and $\|B^{-1}\| = (\min_i \|\vec{b}_i^*\|)^{-1}2^{O(n)}$. This property is satisfied by the η size reduction of L^2 (see [54, Lemma 12]) and the Seysen block size reduction of [33] (see Theorem 2). In fact, Seysen size reduction gives a better bound, substituting $O(\log^2(n))$ for $O(n)$. In practice, however, we prefer to compute η size reductions, since they are easy to compute, and have the convenient property that a projected sublattice of an η-size-reduced basis is η-size-reduced. This definition of size reduction is important because it allows us to bound matrix condition numbers and norms easily. For examples of this, see Appendix B.2.

2.4 Heuristic Assumptions

Heuristic assumptions are frequently used to understand the empirical behavior of lattice algorithms.

The geometric series assumption (GSA) states that reduced lattice bases heuristically have $\|\vec{b}_i^*\|/\|\vec{b}_{i+1}^*\|$ constant. While this holds for some applications, it is false generally. Our results do not depend on the GSA.

The heuristic assumption of [33] states that the slope of a linear regression of the $\log \|\vec{b}_i^*\|$ values decreases exponentially quickly throughout lattice reduction, and in its limit it approximates a known, small value. This assumption is used recursively to bound the maximum and minimum Gram-Schmidt norm, which is in turn used to conclude that an exponentially decreasing working precision suffices for their algorithm. Their assumption is validated against Knapsack and NTRU-like lattices.

This assumption is violated when the final profile does not closely follow the GSA. For example, consider overstretched NTRU. We note that NTRU-like [57] bases are not generated in the same way as genuine NTRU bases, and their empirical behavior during lattice reduction is different. Kirchner and Fouque [34] use a lemma by Pataki and Tural [52] to prove by contradiction that genuine NTRU bases do not follow the GSA, and the Gram-Schmidt norms are shorter than the GSA predicts. As a result, the required working precision does not decrease exponentially.

It is also claimed in [33] that QR factorization (Cholesky decomposition to be precise) and size reduction can heuristically be performed with the same asymptotic running time as matrix multiplication. We do not evaluate this claim in our work, but we observe that if it is true, it lowers the asymptotic running time of our algorithm as well.

3 Lattice Profiles and Their Application

Define the profile of a lattice basis as a vector $\vec{\ell}$ with $\ell_i = \log \|\vec{b}_i^*\|$. The origin of this name is unclear; we note the term has been used in various forms by Kirchner, Espitau, and Fouque [31,32], and Ducas and van Woerden [22,23]. The profile of a lattice basis encodes a wealth of information about its properties and its behavior under lattice reduction, and it can vary significantly across different problems. Works by Kirchner and Fouque [35], Albrecht and Ducas [1], and Ducas and van Woerden [23] have used the behavior of the lattice profile to analyze specific problems, for example to identify dense sublattices and uniquely short vectors for NTRU and LWE lattices that allow solving these problems more efficiently. In our case, we will use the profile to characterize the behavior of lattice reduction algorithms on general lattices. This section recounts some properties of lattice profiles found in prior work, and also develops our new concept of lattice reduction.

3.1 Example Profiles

The cryptanalytic applications of lattice reduction vary significantly, and so too do the input and output profiles of the bases involved in solving these problems, computed before and after performing lattice reduction. Example input and output pairs are given in Figure 1.

We can observe that there are several common input shapes for these profiles. Input bases for q-ary lattices, LWE lattices, and NTRU lattices [42] all resemble a step function, where the first half of the profile is large ($\log q$), and the second half is small (0). We loosely classify these input bases as being "balanced." On the other hand, input profiles for Goldstein-Mayer lattices [27], Gentry-Halevi lattices [26], and knapsack-lattices [38] consist of a single large entry, and the remaining ones are small. Input bases with such profiles were named "knapsack-like" by [33]. Finally, we note several cases where the input profile is neither balanced or knapsack-like, such as the Coppersmith lattices for RSA [41], Elliptic Curve Hidden Number Problem (ECHNP) [59], and Modular Inversion Hidden Number Problem (MIHNP) [9], or Ajtai bases [48], or Goldreich-Goldwasser-Halevi encryption lattices [46]. We call such profiles irregular. These categories are not formal or strict; rather they aid in understanding the spectrum of profiles that appear in lattice cryptanalysis.

The output profiles for these cryptanalytic problems are similarly diverse. One class of output profiles seems to follow the GSA. We use the term "GSA-like" to describe the output profiles of lattice problems which follow such a pattern. The output profiles for random q-ary lattices, Goldstein-Mayer lattices, and Ajtai lattices are all GSA-like. Another well-represented class of output profiles consists of a single small element followed by a GSA-like decay. Output profiles of this shape imply that lattice reduction has found a single short vector $\vec{b}_0$ with small norm, and all other vectors are significantly longer. This context appears when solving the unique SVP problem, so we call such output profiles "uSVP-like." Finally, we observe that several output bases are neither GSA-like nor uSVP-like. Instead, we observe the presence of two or more levels where the profile is basically flat within a level before increasing to the start of the next level. We call these output profiles "structured" because their shape implies the presence of unexpectedly dense sublattices within a lattice, demonstrating additional structure when compared to random lattices. Output profiles for NTRU lattices, NTRU-like lattices [57], and Coppersmith lattices are all structured.

One important observation is that different lattice problems may have identical input profiles, but completely different output profiles. Random q-ary, LWE, and NTRU lattices all have balanced input profiles, but their outputs are GSA-like, uSVP-like, and structured respectively. Representing a lattice basis by its profile therefore discards information about the lattice and makes it impossible to exactly predict the properties and behavior of the lattice profile during lattice reduction. Although the lattice profile does not encode every property about a lattice basis, we show through our analysis that the lattice profile is a powerful tool for analyzing general lattice problems.

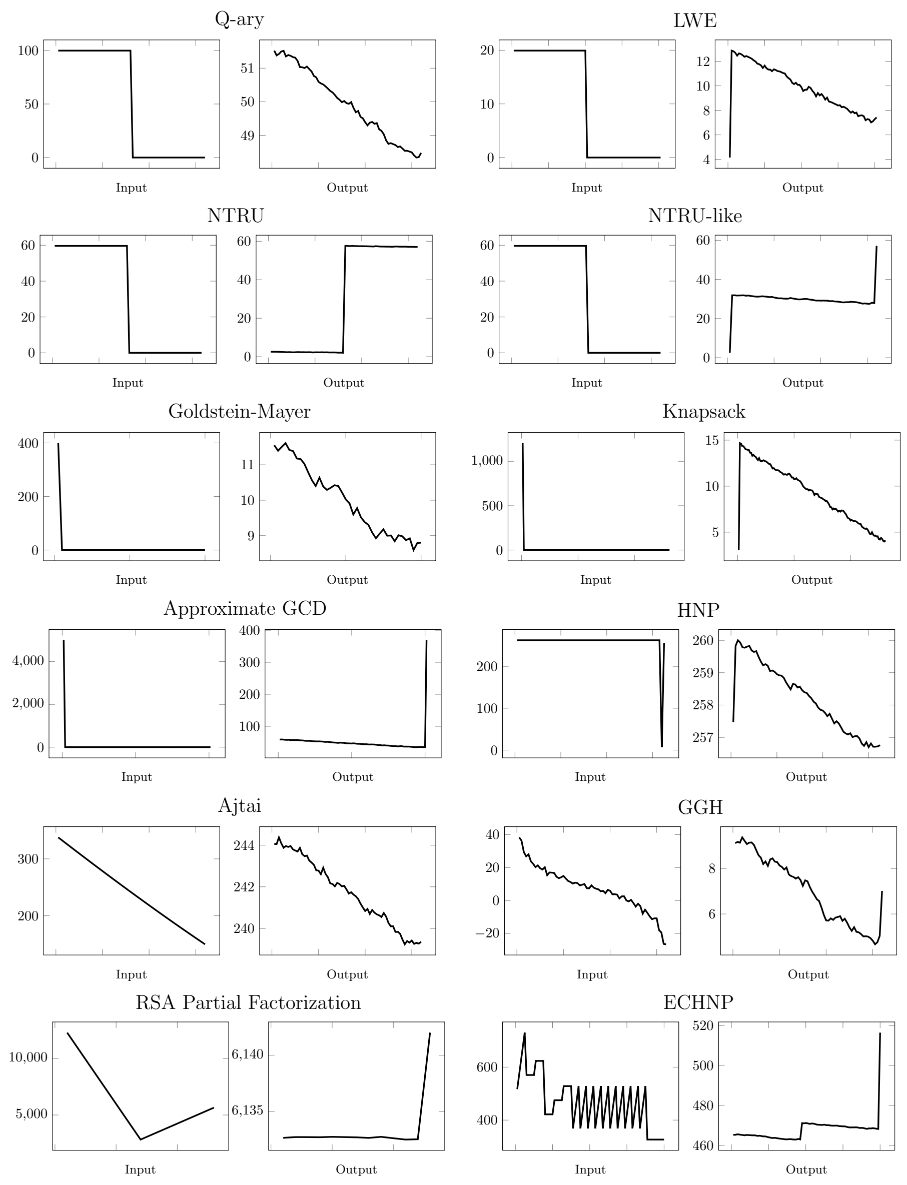

Fig. 1. Sample Lattice Input and Output Profiles. We generated sample lattices for a number of cryptanalytic problems, then LLL-reduced them. There is a wide variety of input types and output types. Note that the scales on the y-axes vary enormously across different problem instances.

3.2 Functions of the Lattice Profile

The profile of a lattice basis encodes significant information about how the profile might evolve during reduction by an LLL-like algorithm. While in some applications it is justified to use the geometric series assumption to predict the results of lattice reduction, it is clear from Figure 1 that this assumption is false for many lattices of interest. For our algorithm to perform well on generic lattices, we must consider all possible ways the profile can change during lattice reduction.

One useful quantity is the *spread* of a profile, which we define in terms of its profile $\vec{\ell}$ as $\mathrm{spread}(B) = \max_i \ell_i - \min_i \ell_i$. Kirchner et al. [33] use the spread to set the working precision of their algorithm, and they use their heuristic assumption to argue that the spread decreases exponentially quickly.

Neumaier and Stehlé give a key result about the spread [45, Lemma 2]. During every profile-altering step in an LLL-like algorithm that updates $\vec{\ell} \to \vec{\ell}'$, we have $\max_i \ell'_i \leq \max_i \ell_i$. Similarly, $\min_i \ell'_i \geq \min_i \ell_i$. That is, $\mathrm{spread}(B)$ never increases. Unfortunately, the spread is not guaranteed to decrease: consider the basis $\begin{bmatrix} 2^p & 0 \\ 0 & 1 \end{bmatrix}$. The spread is p, and the spread of reduced basis $\begin{bmatrix} 0 & 2^p \\ 1 & 0 \end{bmatrix}$ is also p.

In this work, we consider the lattice potential $\mathit{\Pi}(B)$ of a basis in the logarithmic domain

$$\text{Potential: } \mathit{\Pi}(B) = \sum_{i=1}^{n} (n - i + 1)\ell_i.$$

With every swap in an LLL-like algorithm, the potential decreases. Additionally, if we have a sublattice B_{sub} and reduce to B'_{sub}, the change in potential of the sublattice equals the change in potential of the updated full lattice: $\mathit{\Pi}(B_{sub}) - \mathit{\Pi}(B'_{sub}) = \mathit{\Pi}(B) - \mathit{\Pi}(B')$.

Although not directly used in this work, many useful properties of the profile are found in the work of Ducas and van Woerden [22].

3.3 Profile Compression and Profile Drop

The profile of a lattice basis also encodes information about how individual vectors of that basis may be scaled without interfering with the behavior of an LLL-like algorithm. This scaling was first described by Saruchi et al. [54] and plays a central role in our ability to bound the precision required of our lattice reduction algorithm. We recount their technique here, and this motivates the definition of a new function of the lattice profile we call the *drop*.

At a high level, the number of bits of precision required to reduce an arbitrary basis B depends on the log-condition number of B. If B is size-reduced, the log-condition number depends on the spread. If B is also compressed by the scaling method, the spread depends on the drop. We show that the drop decreases during reduction, bounding the precision necessary at each step.

The key observation from the work of Saruchi et al. [54] is that some lattice bases can naturally be split into contiguous blocks where no LLL swaps are ever performed between neighboring blocks. Whenever the profile is known, such as is the case for upper-triangular bases, these blocks can be detected and exploited to reduce the number of bits of precision needed to represent the lattice.

Consider a profile $\vec{\ell}$ of dimension n where there exists index k satisfying

$$\max_{1\le i\le k} \ell_i < \min_{k+1\le i\le n} \ell_i.$$

This naturally splits the lattice into one projected sublattice of dimension k and one of dimension $n-k$. Because the profile-altering swaps of an LLL-like algorithm will never increase the maximum profile value of the first sublattice and will never decrease the minimum of the second, there will never be a point at which an LLL swap occurs at indices $(k, k+1)$.

We can make changes to the basis without interfering with this property. For any $d < \min_{k+1\le i\le n} \ell_i - \max_{1\le i\le k} \ell_i$, we may scale vectors $\vec{b}_{k+1}, \ldots, \vec{b}_n$ by 2^{-d}, and the resulting basis still has the property that no LLL swaps occur at $(k, k+1)$. Since the vectors reduced within a block are scaled uniformly, this operation does not interfere with the order of LLL swaps within a projected sublattice. We call any scaling that preserves this block structure and the relative differences within a block a *valid* scaling. This operation is depicted in Figure 2, and Saruchi et al. give an example method for computing valid scalings [54, Algorithm 2].

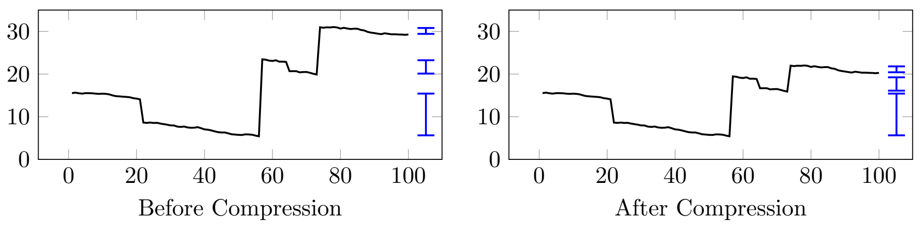

Fig. 2. Example scaling operation. The profile on the left is scaled with integer scaling factors to create the profile on the right. These profiles can be divided into three blocks: 1 to 55, 56 to 72, and 73 to 100. These three blocks can be scaled independently, so the second and third blocks are scaled down so that the gaps between blocks are small. The blue bars on the right depict the different components of the drop. Although the spread decreases with compression, the drop is unchanged.

Similar to how the Lovász condition ensures the decrease between ℓ_i and ℓ_{i+1} is bounded, the drop is used to ensure that the decrease within each disjoint region of the lattice profile is bounded. The drop is the infimum of the spread of a valid scaling of the profile, or equivalently it is the sum of the spreads of each independent block in the profile. A formal definition is given below.

Definition 2 (Profile drop). *Let $\vec{\ell}$ be the profile of a lattice of dimension n, and define the set*

$$D = \cup_{1\le i\le n-1, \ell_{i+1}<\ell_i} [\ell_{i+1}, \ell_i].$$

Then the drop of the profile is the volume of this set.

A few properties are apparent. First, the drop is invariant under valid scaling. Second, by rescaling the regions so the gap between each region is $O(1)$, there is a way to create a scaled lattice with profile $\vec{\ell}' = \vec{\ell} - \vec{d}$ such that spread$(\vec{\ell}') =$ drop$(\vec{\ell}) + O(n)$. Finally, just as Neumeier and Stehlé show the spread never increases during LLL swaps, the same is true of the drop.

3.4 Lattice Reduction Condition

We use the newly defined drop of a lattice profile to specify our reduction condition. The Lovász condition of LLL reduction is unsuitable for our application because it bounds the decrease between all neighboring profile elements by the same amount. For our recursive algorithm, reducing a projected sublattice results in two large decreases between the profile elements directly neighboring the sublattice. Although possible, it is prohibitively expensive to iterate the algorithm until all decreases between neighbors are small. By weakening the definition of a reduced lattice, we obtain an algorithm that is asymptotically and practically more powerful.

Definition 3 (α lattice reduction). *A basis B is α-lattice-reduced for $\alpha > 0$ if it is size-reduced and if drop$(B) \leq \alpha n$.*

Theorem 2 shows that bases reduced in our sense are just as useful as bases reduced in the traditional sense, and we find the same to be true in practice.

Proof of Theorem 2. The proof of this theorem proceeds in much the same way as that in [47, Theorem 9]. One key observation that makes our proof different is that drop$(B) \leq \alpha n \Rightarrow \max_{i<j}(\ell_i - \ell_j) \leq \alpha n$. The other is that $\log \|\vec{b}_i\| \leq \max_{j \leq i} \ell_j + O(n)$ by size reducedness. Full details are in Appendix B.3. $\square$

4 Improved Lattice Reduction Algorithm

A crucial component of our algorithm is how we manage the growth of numerical error as we repeatedly scale, rotate, and round lattice bases. This is analogous to tracking floating point error in a computation, except we apply the concept to entire lattices. While many works [43,51,54] rely on the rigorous numerical bounds of Chang et al. [10] to analyze the effects of small perturbations, we unfortunately cannot rely on their results here. They require an analog of the Lovász condition which is prohibitively strict for our algorithm, and there seems to be no easy way to adapt their results to support our weaker condition on reduced bases. As a consequence, our results are asymptotic and to first order; we leave the important goal of developing rigorous bounds to future work.

Our approach to relating two lattice bases comes from normwise perturbation analysis [56], which uses an exponentially small parameter $\gamma = 2^{-u}$ to quantify the maximum deviation between two values. Frequently analysis is done up to first order in γ, and terms involving γ^2 are assumed to be negligibly small.

Definition 4 (Similar Lattice Bases). *Let B and $\hat{B}$ be two bases of the same rank and dimension, and let γ be a small parameter. We say B and $\hat{B}$ are* similar *if there exists orthogonal Q such that for all nonzero $\vec{x}$, up to first order in γ,*

$$\frac{\|B\vec{x} - Q\hat{B}\vec{x}\|}{\|B\vec{x}\|} \leq \gamma.$$

For simplicity, we write $B \approx_\gamma \hat{B}$.

For integer values of $\vec{x}$, note that $B\vec{x}$ and $Q\hat{B}\vec{x}$ are lattice vectors; the definition therefore states that every vector in lattice $\mathcal{L}(B)$ is normwise close to a particular (rotated) vector in the lattice $\mathcal{L}(\hat{B})$. This is different from [10], which considers matrix perturbations and basis vector perturbations, but our definition implies that if U is invertible, $B \approx_\gamma \hat{B} \Leftrightarrow BU \approx_\gamma \hat{B}U$. The above definition leads to several other useful results.

Lemma 1. *Let $B_1, B_2, B_3 \in \mathbb{R}^{n\times n}$, and let $\gamma, \gamma' \geq 0$. Then*

1. *$B_1 \approx_\gamma B_2 \Leftrightarrow B_2 \approx_\gamma B_1$*
2. *$B_1 \approx_\gamma B_2$ and $B_2 \approx_{\gamma'} B_3 \Rightarrow B_1 \approx_{\gamma+\gamma'} B_3$.*
3. *If $B_1 \approx_\gamma B_2$, then to first order*

$$\|B_2\| \leq (1+\gamma)\,\|B_1\| \quad \text{and} \quad \|B_2^{-1}\| \leq (1+\kappa(B_1)\gamma)\,\|B_1^{-1}\|$$

4. *Let $\vec{\ell_1}$ and $\vec{\ell_2}$ be the profiles of B_1 and B_2, and let $B_1 \approx_\gamma B_2$. Then*

$$|\ell_{1,i} - \ell_{2,i}| < \sqrt{2n^3}\kappa(B_1)\gamma \text{ for all } i \in \{1,\ldots,n\}.$$

The proofs are not particularly enlightening, and they involve routine applications of perturbation theory. They are found in Appendix B.4.

There are a few important things to note with this lemma. First, lattice basis similarity is symmetric, and error grows slowly when considering transitive similarity. Second, for $\gamma \ll 1/\kappa(B)$, we see that lattice similarity implies that the condition number and profiles of similar bases are provably close. This shows how the condition number $\kappa(B)$ plays an important role in similarity, and it justifies our focus on constraining the condition number during lattice reduction.

4.1 Basis Compression

The central object in our algorithm is a *compressed* lattice basis which plays an analogous role to the size reduction and Gram-Schmidt orthogonalization in the original LLL algorithm. In particular, we use lattice compression to convert an arbitrary basis B into a new basis $\hat{B}$ that captures the same geometric properties with respect to lattice reduction, except $\hat{B}$ is upper-triangular with small integer entries and has a small condition number, making it easier to work with.

Definition 5 (γ-Compressed Lattice Basis). *Let $B \in \mathbb{R}^{n\times n}$ be a basis of rank n. We say $\hat{B} \in \mathbb{Z}^{n\times n}$ is a γ-compressed basis of B if it satisfies the following:*

- *$\hat{B} \approx_\gamma BUD$ for some unimodular U and diagonal D.*
- *$\hat{B} \in \mathbb{Z}^{n\times n}$ is upper-triangular with nonzero diagonal.*
- *$\hat{B}$ is size-reduced: $\log \kappa(\hat{B}) = drop(\hat{B}) + O(n)$.*
- *Entries are small: $\log \|\hat{B}\|_{max} = O(drop(B) - \log\gamma + n)$*
- *If U_2 is a unimodular matrix obtained by performing LLL swaps or size-reduction operations on $\hat{B}$, then DU_2D^{-1} is unimodular.*

The last property is related to the scaling technique of Saruchi et al. [54, Theorem 2], and it is due to U_2 having a block upper triangular structure that respects the independent blocks described in Section 3.3. It is possible to efficiently compute a γ-compressed basis given an estimate of $\kappa(B)$. We outline the basic compression operation in Algorithm 2, and give the full details in Appendix B.5.

Algorithm 2: `CompressLattice` (Simplified)

```
  Input  : $B \in \mathbb{Z}^{n\times n}$, $\gamma \le 1/2$, $C > \log\kappa(B)$
  Output: $\hat{B}$ compressed, $U \in \mathbb{Z}^{n\times n}$, $D = diag(2^{d_1}, \dots, 2^{d_n})$ with $d_i \in \mathbb{Z}$
          satisfying $\hat{B} \approx_{2^{-O(\mathrm{drop}(B)+n)}\gamma} BUD$. The profile of $\hat{B}$ is close to the
          ($D$-scaled) profile of $B$ with absolute error $\gamma$.
1 QR-Factorize $B$ to get an upper-triangular, floating point, $\gamma$-similar basis.
2 Compute profile $\vec{\ell}$ from the diagonal of the R-factor.
3 Compute integer scaling vector $\vec{d}$ using the profile and the technique of [54].
4 Scale and round the entries to integer values.
5 Size reduce to get basis $\hat{B}$ and unimodular $U$.
6 return $\hat{B}, U, diag(2^{d_1}, \dots, 2^{d_n})$
```

Lemma 2. *Let $\omega \in (2, 3]$ and ε be global parameters bounding the complexity of algorithms as follows. We assume there exists algorithm* `QR` *that on input B, $C > \log\kappa(B)$, returns a γ-similar basis R with $\kappa(R) = 2^{O(C)}$ in time $O(n^\omega(C - \log\gamma + \log n)^{1+\varepsilon})$. We also assume that there exists algorithm* `SizeReduce` *that size reduces an integer, upper-triangular basis B (with $C > \log\kappa(B)$) in time $O(n^\omega(C + \log\|B\|_{max} + n)^{1+\varepsilon})$. Finally, we assume there exists a matrix multiplication algorithm which computes product A_1A_2 of two $n \times n$ matrices in time $O(n^\omega(\log\|A_1\|_{max} + \log\|A_2\|_{max})^{1+\varepsilon})$.*

Algorithm `CompressLattice` *returns a compressed basis $\hat{B}$, diagonal $D = diag(2^{d_1}, \dots, 2^{d_n})$, and unimodular U satisfying $\hat{B} \approx_{2^{-O(drop(B)+n)}\gamma} BUD$. In addition, if $\vec{\ell}_B$ is the profile of B and $\vec{\ell}_{\hat{B}}$ is the profile of $\hat{B}$, then we have $\left|\vec{\ell}_{\hat{B},i} - (\vec{\ell}_{B,i} + d_i)\right| \le \gamma$. This algorithm takes time $O(n^\omega(C - \log\gamma + n)^{1+\varepsilon})$.*

Example QR factorization and size reduction algorithms satisfying the condition with $\omega = 3$ are provided in Appendix B.5, and heuristic algorithms for $\omega < 3$ are suggested in [33].

The proof of this lemma involves tracking how the condition number changes after each operation, showing how each operation results in a similar lattice. The resulting lattice has small condition number and is upper-triangular as a result of the size-reduction and QR operations. The scaling operation ensures that $\mathrm{spread}(\hat{\ell}_i) = \mathrm{drop}(B) + O(n)$, which in turn bounds the resulting precision.

4.2 Reducing Sublattices

An important feature of recursive lattice reduction algorithms is the ability to compute projected sublattices B_{sub}, reduce them to obtain unimodular U_{sub}, then use U_{sub} to apply the same transformation to the original lattice basis [36,45]. For completeness, we include a description of this operation using our notation of compression and γ-similarity.

Algorithm 3: `ReduceSublattice`

```
Input : Compressed basis $B^{(k)} \in \mathbb{Z}^{n\times n}$, sublattice index $1 \le i < j \le n$,
        reduction quality $\boldsymbol{\alpha}(\cdot)$, approximation quality $\gamma \le 2^{-1}$, reduction
        function LatRed
Output: Compressed basis $B^{(k+1)}$, unimodular $U$, and diagonal $D$ such that
        $B^{(k+1)} \approx_{\gamma'} B^{(k)}UD$ for $\gamma' = 2^{-O(\mathrm{drop}(B^{(k+1)})+n)}\gamma$ and
        $\mathrm{drop}(B^{(k+1)}_{[i:j]}) \le (\boldsymbol{\alpha}(j-i)+\gamma)(j-i)$.
1 $B'_{sub}, U'_{sub}, D_{sub} \leftarrow$ CompressLattice($B^{(k)}_{[i:j]}, \gamma$)
2 $U''_{sub} \leftarrow$ LatRed($B'_{sub}, \boldsymbol{\alpha}, \gamma$)
3 $U' \leftarrow \mathrm{diag}(I_{i-1}, U'_{sub}D_{sub}U''_{sub}D^{-1}_{sub}, I_{n-j})$
4 $B^{(k+1)}, U'', D \leftarrow$ CompressLattice($B^{(k)}U', \gamma$)
5 $U \leftarrow U'U''$
6 return $B^{(k+1)}, U, D$
```

Lemma 3 (Correctness of Algorithm 3 (`ReduceSublattice`)). *Consider compressed basis $B^{(k)}$, sublattice index $[i : j]$, approximation quality γ, lattice reduction function `LatRed`, and lattice reduction quality $\boldsymbol{\alpha}$. Algorithm 3 returns compressed basis $B^{(k+1)}$, unimodular U, and diagonal D such that for $\gamma' = 2^{-O(drop(B^{(k+1)})+n)}\gamma$, we have $B^{(k+1)} \approx_{\gamma'} B^{(k)}UD$. In addition, the profile of $B^{(k+1)}$ matches the (D-scaled) profile of $B^{(k)}$ outside of index $[i : j]$ to absolute error γ and has bounded drop on index $[i : j]$: $drop(B^{(k+1)}_{[i:j]}) \le (\boldsymbol{\alpha}(j-i)+\gamma)(j-i)$. The running time of this algorithm, excluding the call to `LatRed`, is*

$$O\left(n^{\omega}(drop(B^{(k)}) - \log\gamma + n)^{1+\varepsilon}\right).$$

The Global Profile. In both the practical implementation and theoretical analysis of our algorithm, it is helpful to make use of a concept we call the "global profile." As explored in Lemma 3, sublattice reduction returns a new basis whose (scaled) profile is almost unchanged outside of the sublattice index, and the drop is almost bounded within the sublattice index, where the absolute error scales with γ. We note that the cost of this algorithm scales with $-\log \gamma$, so it is exceedingly cheap to trade off running time for accuracy.

The closeness of these profiles means that we can relate the explicitly computed profile of the current (local) sublattice to the (global) profile of the original input had we applied the same unimodular transformations to the input basis. If we are currently reducing the sublattice B_{sub} of input B at global index $[i : j]$ with profiles $\vec{\ell}_{sub}$ and $\vec{\ell}$ respectively, then there exist unimodular U, $D = diag(2^{d_1}, \ldots, 2^{d_n})$, and similarity γ such that

$$|\bar{\ell}_{i+k} - (\ell_{sub,k} - d_{i+k})| \leq \gamma \text{ for } k \in \{1, \ldots, j-i\}.$$

For the purpose of our practical implementation and asymptotic analysis, we assume γ is parameterized so that the absolute error in the profile is negligibly small. This makes the notation much cleaner and more intuitive. We revisit this assumption later when we use our heuristic assumptions to select γ, but we note that the accumulated error is swallowed by the γ' term in Algorithm 3, is offset by the strictly decreasing $\boldsymbol{\alpha}$ in Lemma 4, and does not affect the reducedness of bases, since similarity to a basis with drop $O(\alpha n)$ implies a drop of $O((\alpha+\gamma)n) = O(\alpha n)$. We leave rigorous analyses to future work.

This definition is practically useful, since $\vec{d}$ can be efficiently computed from by tracking how the bases have been scaled during compression. Therefore at any point in the computation, the global profile is known to high accuracy. We use $\bar{\Pi}$ to refer to the potential computed from the global profile $\vec{\bar{\ell}}$, and we observe that this is a useful tool for understanding the remaining work to be done during lattice reduction. For more information, we refer to Appendix B.1.

4.3 Lattice Reduction of Partially Reduced Bases

So far, we have demonstrated how to take a specified sublattice and reduce it, all while efficiently maintaining a "compressed" representation whose size depends on the current profile drop. Depending on the profile of the compressed basis, the choice of sublattice can have a large impact on the running time. It is important to select these sublattices carefully, and in this section we develop a particular strategy that is efficient for input bases whose profiles are slightly constrained. Section 4.5 uses this subroutine to reduce arbitrary input bases.

Our constraints enable us to construct a simple recursive algorithm that maintains these properties, and we use this invariance to argue that our algorithm makes progress no matter the particular evolution of the profile. This definition and recursive subroutine is the core of our algorithm, and we show how it is used to reduce generic lattice bases in Section 4.5. We call such constrained bases *LR-reduced*, since we require that the *Left* ($B_{[0:\frac{n}{2}]}$) and *Right* ($B_{[\frac{n}{2}:n]}$) projected sublattices each have bounded drop.

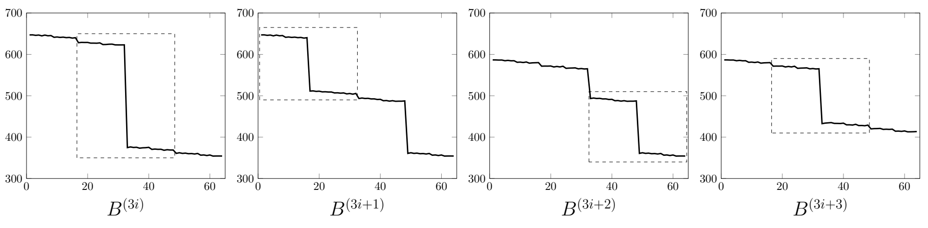

Fig. 3. Changes in Profile during LR Reduction. We logged the profile (without compression scalings) at different stages while running Algorithm 4 on a q-ary lattice basis. The dashed box indicates the profile of the sublattice reduce in each step, corresponding to the Middle, Left, and Right sublattices. Observe that $B^{(3i)}$ and $B^{(3i+3)}$ are LR-reduced, and all sublattices are LR-reduced.

Definition 6 (LR-reduced basis). *We consider a basis B of rank $n = 2^k$ and a reduction parameter $\boldsymbol{\alpha}(\cdot)$. B is LR-reduced if $k = 0$ or:*

- *$B_{[0:\frac{n}{2}]}$ and $B_{[\frac{n}{2}:n]}$ have bounded drop $\leq \boldsymbol{\alpha}(n/2)n/2$, and*
- *$B_{[0:\frac{n}{2}]}$ and $B_{[\frac{n}{2}:n]}$ are LR-reduced with parameter $\boldsymbol{\alpha}$.*

Algorithm 4: `ReduceLR`

```
Input  : Compressed LR-reduced basis $B^{(3i)}$ of rank $n \geq 2$ a power of 2,
         reduction quality $\boldsymbol{\alpha}$, similarity quality $\gamma$
Output: Unimodular $U$ such that $B^{(3i)}UD \approx_{2^{-O(\alpha n)}\gamma} B^{(3r)}$ for diagonal $D$
         and number of rounds $r - i$. $B^{(3r)}$ is LR-reduced and
         $\boldsymbol{\alpha}(n)$-lattice-reduced.
1  if $n = 2$ then
2      return $U \leftarrow$ output of Schönhage's reduction algorithm [55]
3  $B^{(3i+1)}, U_M, D_M \leftarrow$ ReduceSublattice($B^{(3i)}, \frac{n}{4}, \frac{3n}{4}, \boldsymbol{\alpha}, \gamma,$ ReduceLR)
4  $B^{(3i+2)}, U_L, D_L \leftarrow$ ReduceSublattice($B^{(3i+1)}, 0, \frac{n}{2}, \boldsymbol{\alpha}, \gamma,$ ReduceLR)
5  $B^{(3i+3)}, U_R, D_R \leftarrow$ ReduceSublattice($B^{(3i+2)}, \frac{n}{2}, n, \boldsymbol{\alpha}, \gamma,$ ReduceLR)
6  if $drop(B^{(3i+3)}) > \boldsymbol{\alpha}(n)n$ then
7      $U_2 \leftarrow$ ReduceLR($B^{(3i+3)}, \boldsymbol{\alpha}, \gamma$)
8  else
9      $U_2 \leftarrow I_n$
10 $U \leftarrow U_M D_M U_L (D_L U_R (D_R U_2 D_R^{-1}) D_L^{-1}) D_M^{-1}$
11 return $U$
```

We describe a method to fully reduce LR-reduced bases in Algorithm 4, and we depict the behavior in Figure 3. The main appeal of this construction is how the inputs to each recursive call are LR-reduced, so we can temporarily avoid some of the complexity of analyzing the behavior of our algorithm on completely arbitrary lattice bases. We conclude this section by exploring how to set some of the input arguments for this algorithm, and investigate the behavior in Section 4.4.

Setting Reduction Parameter $\boldsymbol{\alpha}$. We have so far used $\boldsymbol{\alpha}(\cdot)$ to represent the approximation factors used to bound the quality of reduced sublattices of different sizes. The particular choice of $\boldsymbol{\alpha}$ has significant impact on the behavior of the algorithm. If $\boldsymbol{\alpha}(n/2) \approx \boldsymbol{\alpha}(n)$, then the process depicted in Figure 3 takes too many rounds to converge, since we require the overall slope in this example to match the slope in both the left and right halves of the profile. Ideally, we want the sublattices to be reduced to higher quality, so we want $\boldsymbol{\alpha}(n/2) < \boldsymbol{\alpha}(n)$. Fewer rounds are needed to reduce a basis of rank n, but if $\boldsymbol{\alpha}(n/2)$ is too small, the high-quality sublattice reductions become prohibitively expensive. In addition, we require $\boldsymbol{\alpha}(2) \geq \alpha^* = \log(4/3)$, a bound determined by Hermite's constant for rank-2 lattices.

A similar phenomenon was observed by Kirchner, Espitau, and Fouque [33], and they suggest asymptotic ranges for determining the reduction quality of sublattices. Since our main goal is to have a practical algorithm that concretely achieves comparable reduction quality to LLL, we develop an entirely new, concrete method for setting this parameter. In particular, if our end goal is reducing a lattice of rank N to quality α, we reduce sublattices of rank n to quality

$$\boldsymbol{\alpha}(n) = \alpha^* + \left(\frac{n}{N}\right)^{\log g} (\alpha - \alpha^*)$$

for some fixed parameter $1 < g < 2^{\omega-2}$. This choice of $\boldsymbol{\alpha}$ geometrically interpolates between α^* and α, and it guarantees $\boldsymbol{\alpha}(2) \geq \alpha^*$ and $\boldsymbol{\alpha}(N) = \alpha$.

Bounding the Number of Rounds. The description of Algorithm 4 references the number of rounds r, which counts how many iterations the cycle of three sublattice reduction steps are applied to input basis $B^{(0)}$. Understanding r is crucial to bounding the running time and numerical error of the algorithm, but although r is small in practice, it is challenging to bound this value for all possible lattice bases without making heuristic assumptions.

Koy and Schnorr [36, Theorem 3] bound the number of iterations of their recursive algorithm with a block variant of the potential function, but they find it is "impractical" to bound the spread of Gram-Schmidt norms between blocks, so their approach fails to bound the profile drop and is incompatible with our algorithm. Neumaier and Stehle [45, Lemma 5] use dynamical systems to bound the number of iterations, but the resulting bases are similarly ill-behaved. Kirchner et al. [33, Section 4.2] suggest using six iterations in practice.

We make the following heuristic assumptions involving r. These assumptions hold for the diverse suite of lattice families we tested our algorithm on, and we were unable to create any pathological counterexamples.

Heuristic 1. The potential reduced in the current round is bounded by the potential reduced in all remaining rounds. In particular,

$$\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3(i+1))}) = \Omega\left(\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3r)})\right).$$

Heuristic 2. The change in potential in the current round is bounded in all except the last two rounds. That is, for all $i < r - 2$,

$$\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3(i+1))}) = \Omega\left(n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))\right).$$

Recalling Figure 3, the first heuristic assumption captures the idea that the bulk of the remaining reduction work happens in the next three sublattice reductions. Intuitively, since the left and right sublattices are already reduced, all of the remaining potential must be between these two halves; reducing the middle sublattice decreases much of this potential. The second heuristic is designed to handle cases where the change in potential is small, like the counterexample described at the beginning of this section. While we know we cannot eliminate these cases entirely, this heuristic states that if no sublattice reduction in a round removes much potential, the profile is very nearly reduced.

As can be found in Appendix B.6, it is possible to show by induction that our heuristic assumptions imply that the number of rounds r is

$$O\left(\log\left(\frac{\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3r)})}{n^3(\alpha - \alpha^*)}\right)\right) = O\left(\log(\mathrm{drop}(B^{(3i)})/n) + \log\left(\frac{1}{\alpha - \alpha^*}\right)\right).$$

Setting the Similarity Parameter $\boldsymbol{\gamma}$. Algorithm 3 returns a γ'-similar basis where $\gamma' = 2^{-O(\mathrm{drop}(B^{(k+1)})+n)}\gamma$; since Algorithm 4 calls this subroutine a total of $3r$ times within the r rounds of reduction, this creates a similarity relationship with parameter $\sum_{i=0}^{r-1} 2^{-O(\mathrm{drop}(B^{(i+1)})+n)}\gamma \leq 3r2^{-O(\mathrm{drop}(B^{(3r)})+n)}\gamma$ $= 3r2^{-O(\alpha n)}\gamma$, and this achieves that the profiles of $B^{(0)}$ and $B^{(3r)}$ are the same up to scaling and absolute error $3r\gamma$.

This suggests setting the working value of γ at each round i to $-\log(\gamma) = \Theta(\log r)$. The heuristic contribution of $\log r = O(\log(\log \mathrm{drop}(B^{(3i)}) + \log n))$ is small, so we infer that the heuristic assumptions about the number of rounds does not matter much when it comes to evaluating the numerical stability. In fact, any choice of γ satisfying

$$\gamma = 2^{-O\left(\mathrm{drop}(B^{(3i)})+n\right)}.$$

does not affect the $O(\mathrm{drop}(B^{(k)}) - \log\gamma + n)$ term in the running time.

4.4 Analyzing the Behavior of Left-Right Reduction

Relating the Potential to the Profile Drop. The lattice potential is a useful tool for bounding the remaining work, but the cost of performing basis compression and matrix multiplication depends on the profile drop. We wish to relate these two quantities, but if a lattice does not follow the GSA, large changes in potential do not necessarily imply large changes in the drop, and vice versa.

As one example, consider the case where an exceptionally short vector is found when reducing the middle sublattice. This means the updated profile has a large, downward spike at index $n/4+1$, corresponding to the short projected vector. The sublattice reduction may have caused a large decrease in potential, but the presence of the large, downward spike can mean there was little change in the drop. Since the large spike is in the center of the left sublattice, reducing that sublattice eliminates that drop. However, it is possible that this sublattice reduction involved only a small decrease in lattice potential, despite the cost of the update step with many bits of precision.

While [33, Section 4.2] bounds the cost complexity per unit reduction in potential, it does so using the heuristic assumption which predicts exponentially decreasing spread. This fails to consider the common cases where sublattice reduction finds vectors shorter than predicted by the GSA. We require a new approach to relate the change in lattice potential to the cost of future updates.

The following lemma shows that the three sublattice reductions in Algorithm 4 either decrease the drop (and therefore future update cost) by a large amount or decrease the potential by a large amount. We prove this by using the LR-reduction property to bound the maximum and minimum values of the profile in sublattices of rank $n/4$ after each of the three sublattice reductions.

Lemma 4. *Let $B^{(3i)}$ be the compressed LR-reduced basis of rank n at round i used as input to Algorithm 4, and let $B^{(3i+3)}$ be the basis after the three sublattice reductions. The drop of the input basis for the next round is bounded by the current round's change in global potential $\bar{\Pi}$:*

$$n^2\left(\mathit{drop}(B^{(3i+3)}) - 5\boldsymbol{\alpha}(n)n/2\right) = O\left(\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3i+3)})\right).$$

We will use this lemma to amortize the cost of expensive update steps with the overall change in lattice potential.

Bounding the Running Time. There are many factors that go into bounding the running time of Algorithm 4. Our goal in analyzing this algorithm is to find a cost function T that can bound the running time at all levels. One piece of the running time on input B is the cost of the nonrecursive steps, which by Lemma 3 and our choice of γ is $O(n^\omega(\mathrm{drop}(B)+n)^{1+\varepsilon})$. We write this as $O(n^\omega p^{1+\varepsilon})$ for $p = \Theta(\mathrm{drop}(B)+n)$. Note that $\log(\|B\|_{max}) = O(p)$ and $\log(\|U\|_{max}) = O(p)$, so it is natural to think of p as the working precision. Note also that in the base case, the cost of Schönhage reduction [55] takes time $O(p^{1+\varepsilon}) = O(n^\omega p^{1+\varepsilon})$.

The cost function T also contains the recursive cost of Algorithm 4, which calls itself three times on sublattices of rank $n/2$, and once on a lattice of rank n. A cost function T is therefore a valid bound on the running time if

$$T(B^{(3i)}) \geq T(B_M) + T(B_L) + T(B_R) + C_u n^\omega p^{1+\varepsilon} + T(B^{(3i+3)}) \geq 0 \qquad (1)$$

for some constant $C_u > 0$ bounding the cost of the base case and nonrecursive steps. We have found that the cost function naturally depends on three values.

1. **Potential Change.** Like in the original LLL algorithm, the potential of a lattice basis conveys important information about how much work remains. Including this term allows us to distinguish between lattice bases that require the same amount of precision but different amounts of remaining potential. We set this term to
$$T_{\bar{\Pi}}(B^{(3i)}) = n^{\omega-2}\left(\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3r)})\right)p^{\varepsilon}.$$
2. **Precision.** Lattice basis potential does not capture the entire picture. A lattice basis may have asymptotically small potential, but require arbitrarily large precision. Since the nonrecursive steps depend on the precision, this term must be present. We set this term to
$$T_{prec}(B^{(3i)}) = n^{\omega}p^{1+\varepsilon}.$$
3. **Approximation.** We require at least one term to depend on the approximation factor of lattice reduction. Otherwise, this would imply that it is possible to achieve lattice bases with arbitrarily good approximation. We set this term to
$$T_A(B^{(3i)}) = \left(\frac{\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3r)})}{n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))} + c_1\right) n^{\omega}(n + \boldsymbol{\alpha}(n)n)p^{\varepsilon}$$
for $i < r$ for some $c_1 > 0$ and $T_A(B^{(3r)}) = 0$.

We define function $T(B^{(3i)}) = C_{\bar{\Pi}}T_{\bar{\Pi}}(B^{(3i)}) + C_{prec}T_{prec}(B^{(3i)}) + C_A T_A(B^{(3i)})$ for some constants $C_{\bar{\Pi}}, C_{prec}, C_A$. While the first two terms make intuitive sense, the third is more difficult to explain. As the basis becomes more reduced, the approximation factor of sublattice reduction remains a fixed cost that always contributes to the drop. To bound the contribution of this fixed cost in each node of the recursion tree, we use the heuristic assumptions to bound the recursive behavior and show that this choice of $T_A(\cdot)$ satisfies the requirements. Our conclusion does not rule out the possibility of simpler constructions; we only aim to give a reasonable construction that works.

Lemma 5. *Let $\omega \in (2, 3]$ be a parameter bounding the complexity of matrix multiplication, size reduction, and QR factorization as described in Lemma 2. If the heuristic assumptions 1 and 2 are correct, then there exists appropriate choice of constants $C_{\bar{\Pi}}, C_{prec}, C_A > 0$ such that $T(\cdot)$ satisfies Equation 1.*

As a corollary, this means that for input B and reduction goal α, it is possible to instantiate $\boldsymbol{\alpha}$ and γ such that the running time of Algorithm 4 on these inputs is $O(T(B)) =$
$$O\left(\left(\frac{\alpha}{\alpha - \alpha^*}\right) n^{\omega}p^{1+\varepsilon} + \alpha n^{\omega+1}p^{\varepsilon}\right).$$
where $p = O(drop(B) + n)$.

4.5 Reducing Generic Lattice Bases

Algorithm 4 provides a method for α-reducing bases with rank $n = 2^k$ that are LR-reduced and compressed, but most lattice bases do not satisfy this condition. Fortunately, it is comparatively easy to reduce generic bases using this method.

We begin by *padding* the lattice with a number of large, orthogonal vectors in higher-dimensional space until the rank of the padded lattice is a power of 2. These extraneous vectors are chosen so that they are never modified by lattice reduction. In particular, for an input basis B of rank n with $\kappa(B) < 2^C$, we generate the padded basis of dimension $N = 2^{\lceil \log(n) \rceil}$

$$B_{pad} = \begin{bmatrix} B & 0 \\ 0 & \lceil 2n\|B\|_{max} \rceil I_{N-n} \end{bmatrix}.$$

We have $\mathrm{drop}(B) = \mathrm{drop}(B_{pad})$ and $\kappa(B_{pad}) = 2^{C+O(\log n)}$. The latter statement is because the singular values of B_{pad} are the singular values of B and $\lceil 2n\|B\|_{max} \rceil$. The largest singular value of B_{pad} is only up to $O(n)$ times larger than the largest of B, and the smallest singular value is unchanged.

We next compress B_{pad} to obtain a compressed basis for input to Algorithm 3. This basis is not LR-reduced, however. We recursively reduce the left sublattice to quality $\boldsymbol{\alpha}$, and then do the same with the right. Now that the basis is LR-reduced and compressed, we use Algorithm 4 to reduce it the rest of the way. Although this algorithm is also recursive, its recursive structure is simply a binary tree, and it is fully analyzable.

Lattice reduction of B_{pad} returns U_{pad} such that, for some scaling D_{pad}, $\mathrm{drop}(B_{pad}U_{pad}) = \mathrm{drop}(B_{pad}U_{pad}D_{pad}) \leq \alpha N/2$. In addition, the padding scheme yields the special structure

$$U_{pad} = \begin{bmatrix} U' & 0 \\ 0 & I_{N-n} \end{bmatrix},$$

so $\mathrm{drop}(BU') \leq \alpha n$. We finally size-reduce BU' to obtain U which α-lattice-reduces B. These operations are described in detail in Appendix B.7. We arrive at the following result.

Theorem 1 (Full)**.** *Let $B \in \mathbb{Z}^{n\times n}$ be a lattice basis and $C > \log(\kappa(B))$ a bound on its condition number. Let α^* be a constant determined by the Hermite constant in small dimension, and let $\alpha > 2\alpha^*$ be the desired reduction quality. Finally, let $\omega \in (2,3]$ and $\varepsilon > 0$ be parameters bounding the runtime of size reduction, matrix multiplication, and QR factorization as described in Lemma 2. If the heuristic assumptions 1 and 2 are correct, then our algorithm returns unimodular U such that BU is $O(\alpha)$-lattice-reduced. The running time of our reduction algorithm is*

$$O\left(\left(\frac{\alpha}{\alpha-\alpha^*}\right) n^{\omega}(C+n)^{1+\varepsilon}\right).$$

5 Implementation Details

We implemented our algorithm in C++. Our implementation is currently about 14,000 lines of code. We have two versions of our algorithm: the "provable" version, where the padding scheme, recursion decisions, and reduction parameters match those described in our proofs, and the "heuristic" version, which we use to evaluate the practical performance in our running time experiments.

Heuristic and optimization improvements. Our heuristic implementation does not implement the padding scheme from Section 4.5. Rather than limit `ReduceLR` to operate on bases of rank 2^k, we simply round when subdividing the lattice. In addition, we also use an approach similar to [33] to reduce sublattices to quality $\alpha \gg \boldsymbol{\alpha}(n/2)$ in early rounds, avoiding expensive reductions of sublattices that have little effect on global potential. Until the drop is sufficiently small, we reduce sublattices to quality $\Theta(\mathrm{drop}(B)/n)$, which improves running time considerably.

We also do not always recurse down to rank 2. Instead, if the dimension is ≤ 32 and required precision is at most 128 bits, we use fpLLL to reduce that basis using LLL or BKZ, depending on the desired reduction quality. This allows us to achieve α better than $\log(4/3)$ in practice.

Typically, we perform compression so the resulting size is $2 \cdot \mathrm{drop}(B) + 3n + 30$ bits and find that our algorithm is stable for all of our test cases. For some of the ultra-large test cases, we aggressively set the precision to $\mathrm{drop}(B) + 30$ bits. This is more memory efficient, but it is not stable for some of the lattice bases with structured output, so we do not do this by default.

Many optimizations are found in the compression function, as this is a significant cost in our algorithm. We note that $B^{(k)}U'$ in Algorithm 3 is block-upper-triangular, with $B_{2,2}$ corresponding to $B^{(k)}_{[i:j]}$ in the following example:

$$B^{(k)}U' = \begin{bmatrix} B_{1,1} & B_{1,2} & B_{1,3} \\ 0 & B_{2,2} & B_{2,3} \\ 0 & 0 & B_{3,3} \end{bmatrix}.$$

The only non-upper-triangular block on the diagonal is $B_{2,2}$, so $\begin{bmatrix} B_{2,2} & B_{2,3} \end{bmatrix}$ can be QR-factorized on its own. If $B^{(k)}$ was η-size-reduced before sublattice reduction, then $B_{1,1}$ and $B_{3,3}$ remain size-reduced. $B_{1,2}$ changed with multiplication by U', but only $B_{1,1}$ is needed to size-reduce $B_{1,2}$. This is independent of the QR-factorization in the second row, so we are able to exploit the data dependencies to do multiple operations in parallel, including reducing multiple sublattices simultaneously. If we use do a variant of Seysen size reduction, we avoid a potentially unstable inversion of $B_{1,1}$ by exploiting its triangularity [28, Chapter 8] to size-reduce $B_{1,2}$.

We additionally adopt a version of [33, Algorithm 8] to make a basis LR-reduced, rather than exactly implement the method described in Section 4.5. This is because in practice, knapsack-like lattices often behave like random, GSA-like lattices at the beginning of computation. The first sublattice reduction of rank 2 often decreases the drop by half, and it is cheaper to recompress the entire rank-n basis now since only two rows are not η-size-reduced. However, it is easy to construct knapsack-like counterexamples that do not behave like random lattices (a diagonal knapsack-like basis is one example), so while this is a worthwhile case to optimize for, and there is no downside to performing early reduction in this way, it is not possible to prove that our algorithm runs faster on knapsack-like input bases.

6 Experimental Evaluation

We conducted a series of experiments to demonstrate the performance of our algorithm. All of our experiments were carried out on a single core of a 2.20GHz Intel Xeon E5-2699v4 processor of a machine with 512GB of RAM unless noted otherwise. This machine was built in 2016. For fpLLL, we used version 5.4.2 available on GitHub, compiled and run on this same machine. For the Kirchner, Espitau, and Fouque (KEF) algorithm, the authors kindly shared their source code with us, which we also compiled and ran on this same machine.

Although the specific details of a problem determine what approximation factor is required to solve the problem, we configure all three implementations to heuristically obtain equivalently short vectors. Typically, this corresponds to a root Hermite factor of 1.02 or 1.03, as is achieved by default with fpLLL. Each measurement is collected over a single instance of the respective problem, since the running time does not vary significantly across multiple trials. Additional experiments are found in Appendix A.

6.1 Knapsack Lattices

Knapsack lattices are one of the canonical applications of lattice reduction in cryptography and cryptanalysis, and one of the most common classes of lattices encountered in lattice reduction problems. One of the first applications of lattice reduction in cryptography was to solve the low-density subset sum problem to attack knapsack public-key cryptography. We use the lattice basis construction of [38], which is knapsack-like in input and uSVP-like in output, since the solution to the subset sum problem corresponds to a uniquely short vector.

We constructed the lattice bases for the knapsack instances in [33] and reduced them using our implementation, fpLLL, and the implementation of [33]. Because their implementation crashes partway through the reduction process, we present their reported values. All examples were run single-threaded.

We conjecture that the presence of this uniquely short vector, which means that the heuristic assumption of [33] does not hold at all recursion levels, leads to their algorithm requiring more bits of precision than ours, and therefore a longer runtime.

6.2 Gentry-Halevi Fully Homomorphic Encryption

Gentry proposed a Fully Homomorphic Encryption (FHE) scheme based on ideal lattices in 2009 [25] with concrete parameters suggested by Gentry and Halevi

Table 1. Performance on knapsack lattices.

Dimension	Bit size	fpLLL (s)	[33, reported] (s)	Ours (s)
128	100000	3831	400	69
256	10000	2764	200	83
384	10000	10855	780	246

in [26]. The scheme relied on two hardness assumptions. First, the difficulty of breaking the underlying somewhat homomorphic encryption scheme was based on ideal lattice problems, and the hardness of the transformation that made it bootstrappable was based on the sparse subset sum problem (SSSP). Gentry and Halevi have created several public keys for this scheme as a cryptanalytic key recovery challenge.[2]

Due to a polynomial-time quantum algorithm [6] and subexponential-time classical algorithms [8,7] solving the principal ideal problem, FHE based on ideals is no longer considered secure. The SSSP challenges were solved in practice by [39], breaking the bootstrapping argument of Gentry and Halevi's scheme. We are unaware of a practical break of the underlying ideal lattice problems for the proposed small, medium, and large parameters in the main challenges.

Gentry and Halevi propose different security levels s that naturally lead to lattices in Hermite Normal Form of dimension $n = 2^{s+1}$ and bit size $380(2^{s+1})$. The goal of lattice reduction is finding a basis with $\|\vec{b}_n^*\|$ sufficiently large. Gentry-Halevi lattices are knapsack-like in input and GSA-like in output.

Chen and Nguyen reduced the toy parameters (dimension 512) in 30 core-days in 2011 using generic lattice reduction algorithms [11]. They estimated that the small parameters (dimension 2048) would take 45 core-years. Plantard, Susilo, and Zhang exploited extra algebraic structure to solve the toy challenge in 24 core-days in 2015, with an estimate of 15.7 years for the small parameters. [53]

Our implementation solves the toy parameters in 15 core-minutes and the small parameters in under 31 core-hours running single-threaded on the 2.2GHz processors. We additionally broke the small parameters in 4 hours 10 minutes wall time running in multithreaded mode on an AMD EPYC 72F3 8-core processor running at up to 4.1GHz. Gentry and Halevi's small main challenge has solution `201 216 186 242 353 55 335 420 104 13 299 262 510 414 239`.

The KEF implementation [33] ran on lattices up to dimension 128; beyond this point none of the experiments terminated. Because the input and output profiles of these lattices aligns with the heuristic behavior of the "knapsack-like" case in [33], we expect our implementation to match the asymptotic behavior of theirs. This is what we observe in Figure 4 for the small dimensions where their implementation terminated.

[2] https://shaih.github.io/pubs/FHE-challenge-2010.html

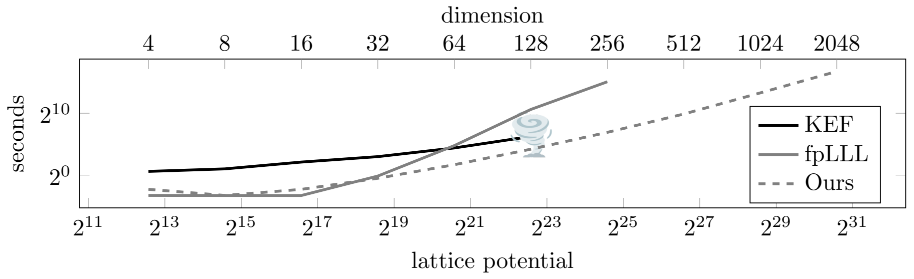

Fig. 4. Gentry-Halevi Ideal FHE. We compare the performance of our lattice reduction algorithm on Gentry-Halevi ideal lattices. The 512-dimensional parameters correspond to the "toy" parameter settings in the original scheme, and 2048 dimensions were the "small" parameters. The KEF algorithm did not terminate on these lattices above dimension 128.

6.3 Univariate Coppersmith

Coppersmith's method can be used to find small roots of low-degree polynomials modulo integers of possibly unknown factorization. This method is particularly interesting in cryptanalysis because it is fully provable using only the approximation guarantees of LLL, and is one of the canonical applications for LLL in cryptanalysis. However, the dimension and parameters increase quickly as the size of the problem approaches the asymptotic limits.

Coppersmith's original papers [14] described a different lattice construction than the one we implement; we implement the version that is due to Howgrave-Graham [29], which is a dual formulation of Coppersmith's original lattice. Coppersmith surveys a number of interesting optimizations in a later paper. [15]

We implemented Coppersmith's method to solve the problem of decrypting low public exponent RSA with stereotyped messages. We set $e = 3$ for a 2048-bit modulus N, and varied the number of unknown bits of the message from 400 (solvable with a dimension 5 lattice) and 678 (solvable with a 382-dimensional lattice with 430,000 bit entries). The asymptotic limit for the method with these parameters (without additional brute forcing) would be expected to be $\approx \lfloor(\log N)/3\rfloor = 682$ bits.

These lattices violate the GSA, and the heuristic assumption in [33] does not hold for them. These lattices have irregular input and structured output. We note that this application only requires finding a single short vector, so the provable guarantees of some of the recursive algorithms would suffice to solve it.

We compare our algorithm to fpLLL, the fpLLL with rounding approach of Bi et al. [5] which achieves an asymptotic improvement over LLL for this lattice construction, and the KEF algorithm [33].

The experimental results are shown in Figure 5. We can see that the LLL with rounding technique achieves an asymptotic improvement over plain LLL, and our algorithm improves further. The implementation of [33] crashed on all instances after dimension 20, a lattice with 20,000 bit entries.

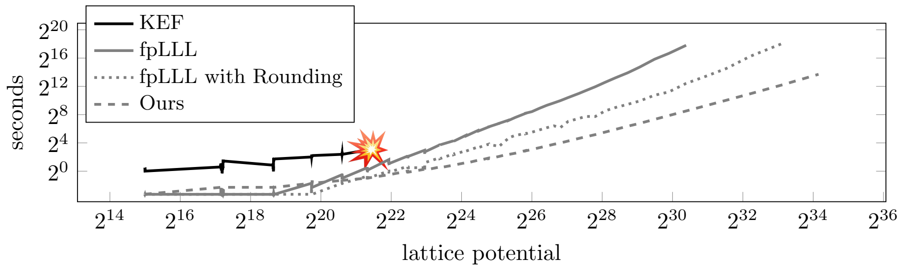

Fig. 5. RSA with Stereotyped Messages. We generated Coppersmith/Howgrave-Graham lattices for 2048-bit RSA plaintext recovery from stereotyped messages, and compare our running time to the KEF algorithm, fpLLL, and fpLLL with the rounding approach of Bi et al. [5]. The KEF algorithm crashed on all instances above dimension 20.

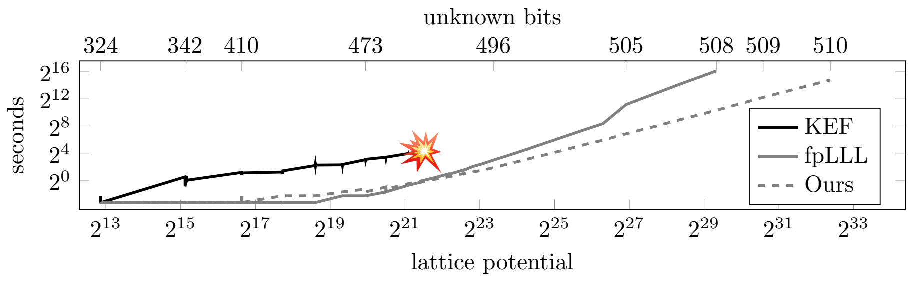

Fig. 6. RSA Partial Factorization. We generated Howgrave-Graham lattices for 2048-bit RSA factorization with high bits known. The dimension and potential grow quickly as the number of unknown bits approaches the asymptotic limit of 512. The KEF code crashed after 487 unknown bits, solvable with a 19-dimensional lattice. The largest parameters we were able to solve required a 265-dimensional lattice with 270,000-bit entries.

We attempted to determine why their implementation crashed to rule out simple programming errors. The most common cause of failure occurred while performing size reduction; an assertion failed, indicating that the function failed to converge on a stable solution within 1000 iterations. We hypothesize that the lack of stability is due to incorrectly setting the working precision of their algorithm. Occasionally, their implementation exhausted the 512GB RAM on our test machines, and the process was killed by the operating system. These failures appear to stem from weaknesses in the algorithm rather than the programming of the implementation.

6.4 RSA Factorization with High Bits Known

In [14], Coppersmith gave a polynomial-time algorithm for factoring an RSA modulus from half of the most or least significant bits of one of the factors, assuming only the approximation guarantee of LLL.

Later, Howgrave-Graham gave an alternative formulation of factoring with partial information as an instance of the problem of computing approximate common divisors [30], with an alternative lattice construction that has become the preferred method of solving this problem in practice. This method has seen several real-world applications, including cryptanalysis of broken random number generation in Taiwanese smartcards [4], cryptanalyzing broken Infineon prime generation [44], and cryptanalyzing Mega encryption [3].

In theory, using this method it is possible to factor an RSA modulus in polynomial time with knowledge of half of the most or least significant bits of one of the factors. In practice, this method can be remarkably efficient when relatively few bits need to be recovered, but the parameters required to solve the problem increase quickly as the number of bits to solve for approaches half the bits of a factor. To be concrete, for a 2048-bit RSA modulus, a 3-dimensional lattice with 2048-bit entries suffices to solve for 341 unknown bits of one of the 1024-bit factors, but if one wishes to solve for 511 unknown bits, the minimum parameters that result in a solvable problem instance dictate that one needs to reduce a 545-dimensional lattice with 557,000-bit entries, assuming that the reduction achieves an approximation factor of $1.02^{\dim L}$.

Figure 6 shows experimental results comparing algorithm performance on Howgrave-Graham lattices of minimal parameters to solve each problem size. The largest problem instance we were able to solve recovered 510 bits of a 1024-bit factor; this required reducing a 265-dimensional lattice with 270,000-bit entries in under 8 hours of computation time. The largest instance fpLLL was able to solve recovered 508 unknown bits by reducing a lattice of dimension 135 with 133,000-bit entries. For this problem size, fpLLL took around 20 hours of computational time; our implementation reduced the same lattice in 20 minutes. The implementation of [33] crashed on all instances larger than the 19-dimensional lattice that solves the problem for 487 unknown bits.

In practice, one can brute force some of these unknown bits to avoid prohibitively expensive lattice reduction times, and apply the "chaining" approach of Bi et al. [5] to the successive lattice reductions. Improving the runtime of the lattice reduction step is an important step in reducing the total runtime of this process. The lattice for this problem has irregular input and structured output, violates the GSA, and falsifies the heuristic of [33].

6.5 *q*-ary Lattice Reduction

Lattice-reduction algorithms are frequently run on q-ary lattices of the form

$$B = \begin{bmatrix} qI & A \\ 0 & I \end{bmatrix}.$$

It is easy to generate lattices of this form uniformly at random by sampling A from $\mathbb{Z}_q^{m\times n}$. Input bases like this are balanced, but their outputs types can vary depending on the problem. Although the profiles for LWE and NTRU lattices behave in the same way as for a random q-ary lattice, there is a "phase transition" where the extra structure in these lattices becomes apparent [1,23]. For LWE and NTRU, this divergence from randomness occurs when the secret vector is found, and lattice reduction can terminate early. It is therefore interesting to consider the performance of our algorithm on random q-ary lattice bases, and the output of these bases is GSA-like.

We ran our lattice reduction implementation on the same parameter sizes as reported in [33] and to approximation quality equivalent to root Hermite factor 1.02. We reduced the same lattices using fpLLL and the implementation from [33]. All experiments were single-threaded.

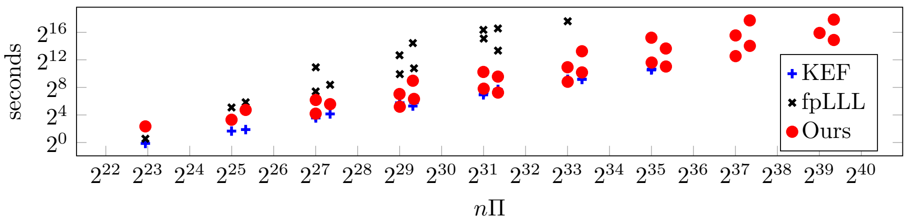

Fig. 7. ***q*-ary.** The algorithm of [33] has been optimized for the case of q-ary lattices, and performs quite well on them. Our implementation shows similar asymptotic behavior. [33] did not terminate for instance sizes above 1024 dimensions.

Our implementation is significantly faster than fpLLL, but slower than [33]. The implementation of [33] did not terminate for instance sizes above dimension 1024 with precision 512, dimension 512 with precision 4096. The largest basis successfully reduced by fpLLL had dimension 256 with 2048-bit entries. The implementation of [33] has been aggressively optimized for this lattice profile, and such optimizations could likely be used to speed up our own implementation even further. We also note that our implementation was capable of running on even larger instances. The largest q-ary basis we successfully reduced had dimension 1024 with 2048-bit entries.

6.6 Approximate GCD

Numerous FHE and multilinear map schemes [12,18,19,20,21,58] have been proposed which rely on the hardness of the approximate greatest common divisor (AGCD) [30] problem. There are many different lattice constructions for solving AGCD (see [58, Appendix B] and [13,24,33]), and the solvability depends on both the number of samples and the approximation factor. While Kirchner et al. mention that they vary the approximation factor and dimension out of

Table 2. Performance on AGCD lattices. We perform a lattice attack on FHE and multilinear map schemes by solving instances of the AGCD problem. The reported size is the maximum size of entries in the basis in millions of bits. Kirchner et al. also performed experiments on CCK, but we do not list their running time because our AGCD instances were instantiated from [17] instead of the revised parameters in [12].

Scheme	Sec. Lvl.	Size (Mb)	Our dim.	Our α	KEF (s)	Our time (s)
CMNT [20]	42	0.16	152	0.220	300	59
	52	0.86	556	0.210	3300	1406
	62	4.2	2156	0.179		52016
CNT [21]	42	0.06	204	0.199	200	54
	52	0.27	876	0.135	1700	1673
	62	1.02	3422	0.065		531005
CLT [19]	42	0.27	144	0.314	780	85
	52	1.1	600	0.213	10560	1946
	62	4.2	2486	0.167		68940
CLT [18]	52	0.99	601	0.210	4000	1779
	62	4.26	2514	0.167	79320	67599
CCK* [17]	52	0.29	313	0.209		271
	62	1.6	1138	0.190		7758
	72	8.5	4889	0.135		526391

memory concerns, they do not document the final parameters for each test case. We derive and run experiments for our own construction and parameters.

We build and reduce the dual of the lattice construction (3.1) presented in van Dijk et al. [58] to solve AGCD, which we note is related to the multiplicity-one Coppersmith construction [13]. We focus on the dual because the resulting unimodular update matrices are smaller in practice, and computing the update matrices for the primal is prohibitively expensive. It is efficient to compute the dual of the reduced dual and recover the secret short vector in the primal.

Like [33], we observe a tradeoff between approximation quality α, dimension, and attack time. As predicted by Theorem 1, we experimentally observe that both high dimension and small α can lead to long running times. We find a curve that fits the behavior on small instances to predict parameter settings that lead to fast reduction times for large instances. Our choice of parameters may not be optimal, but our strategy is effective for solving AGCD in practice.

We generated lattice bases for the same schemes described in [33] and attempted to reduce them using both their implementation and our implementation in multithreaded mode to match the original evaluation. Their implementation crashed partway through reduction on all instances, so we present their running times as originally reported. We present the results in Table 2.

We see that our implementation is faster in all cases. However, since we do not have the full details of their construction, we are unsure if the improved performance is due to the nature of our algorithm, the details of our implementation, or the parametrization of our attack. Regardless of the reason, we also note that our implementation successfully broke larger instances of the AGCD problem than previously reported for three of the schemes, corresponding to a 62-bit security level. One of the longest running attacks we performed reduced a basis of rank 4889 and took around 6 days to complete.

Acknowledgments

We are grateful to Thomas Espitau for helpful discussions and sharing source code, we thank Jonathan M. Smith for graciously providing computing time on different, modern processors for our experiments, and we thank Daniel J. Bernstein for numerous discussions and grantwriting. We also thank Léo Ducas for informative discussions and Shai Halevi for finding and reuploading an offline copy of the Gentry and Halevi FHE challenges. Finally, we thank the anonymous reviewers for their extensive comments on an earlier version of this work. This work was supported by NSF grant no. 1913210.

References

1. Albrecht, M., Ducas, L.: Lattice attacks on NTRU and LWE: A history of refinements. Cryptology ePrint Archive, Report 2021/799 (2021), https://eprint.iacr.org/2021/799
2. Albrecht, M.R., Bai, S., Ducas, L.: A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 153–178. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_6
3. Backendal, M., Haller, M., Paterson, K.G.: MEGA: Malleable Encryption Goes Awry. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 450–467 (May 2023). https://doi.org/10.1109/SP46215.2023.00026
4. Bernstein, D.J., Chang, Y.A., Cheng, C.M., Chou, L.P., Heninger, N., Lange, T., van Someren, N.: Factoring RSA keys from certified smart cards: Coppersmith in the wild. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 341–360. Springer, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42045-0_18
5. Bi, J., Coron, J.S., Faugère, J.C., Nguyen, P.Q., Renault, G., Zeitoun, R.: Rounding and chaining LLL: Finding faster small roots of univariate polynomial congruences. In: Krawczyk [37], pp. 185–202. https://doi.org/10.1007/978-3-642-54631-0_11
6. Biasse, J.F., Song, F.: Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In: Krauthgamer, R. (ed.) 27th SODA. pp. 893–902. ACM-SIAM (Jan 2016). https://doi.org/10.1137/1.9781611974331.ch64
7. Biasse, J.F.: Subexponential time relations in the class group of large degree number fields (2014). https://doi.org/10.3934/amc.2014.8.407
8. Biasse, J.F., Fieker, C.: Subexponential class group and unit group computation in large degree number fields. LMS Journal of Computation and Mathematics **17**(A), 385–403 (2014). https://doi.org/10.1112/S1461157014000345
9. Boneh, D., Halevi, S., Howgrave-Graham, N.: The modular inversion hidden number problem. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 36–51. Springer, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_3

10. Chang, X.W., Stehlé, D., Villard, G.: Perturbation analysis of the QR factor R in the context of LLL lattice basis reduction. Mathematics of Computation **81**(279), 1487–1511 (2012), https://hal-ens-lyon.archives-ouvertes.fr/ensl-00529425
11. 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 (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_1
12. Cheon, J.H., Coron, J.S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_20
13. Cohn, H., Heninger, N.: Approximate common divisors via lattices. ANTS X p. 271 (2012). https://doi.org/10.2140/obs.2013.1.271
14. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology **10**(4), 233–260 (Sep 1997). https://doi.org/10.1007/s001459900030
15. Coppersmith, D.: Finding small solutions to small degree polynomials. In: Silverman, J.H. (ed.) Cryptography and Lattices. pp. 20–31. Springer, Heidelberg (2001)
16. Coppersmith, D., Shamir, A.: Lattice attacks on NTRU. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 52–61. Springer, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_5
17. Coron, J.S., Lepoint, T., Tibouchi, M.: Batch fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/036 (2013), https://eprint.iacr.org/2013/036
18. Coron, J.S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_26
19. Coron, J.S., Lepoint, T., Tibouchi, M.: Scale-invariant fully homomorphic encryption over the integers. In: Krawczyk [37], pp. 311–328. https://doi.org/10.1007/978-3-642-54631-0_18
20. Coron, J.S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-22792-9_28
21. Coron, J.S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 446–464. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_27
22. Ducas, L., van Woerden, W.: A note on a claim of Eldar & Hallgren: LLL already solves it. Cryptology ePrint Archive, Report 2021/1391 (2021), https://eprint.iacr.org/2021/1391
23. Ducas, L., van Woerden, W.P.J.: NTRU fatigue: How stretched is overstretched? In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 3–32. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_1
24. Galbraith, S.D., Gebregiyorgis, S.W., Murphy, S.: Algorithms for the approximate common divisor problem. LMS Journal of Computation and Mathematics **19**(A), 58–72 (2016). https://doi.org/10.1112/S1461157016000218
25. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC. pp. 169–178. ACM Press (May / Jun 2009). https://doi.org/10.1145/1536414.1536440

26. Gentry, C., Halevi, S.: Implementing Gentry's fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_9
27. Goldstein, D., Mayer, A.: On the equidistribution of Hecke points **15**(2), 165–189 (2003). https://doi.org/10.1515/form.2003.009
28. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edn. (2002)
29. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M. (ed.) 6th IMA International Conference on Cryptography and Coding. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (Dec 1997)
30. Howgrave-Graham, N.: Approximate integer common divisors. In: Silverman, J.H. (ed.) Cryptography and Lattices. pp. 51–66. Springer, Heidelberg (2001)
31. Kirchner, P., Espitau, T., Fouque, P.A.: Algebraic and euclidean lattices: Optimal lattice reduction and beyond. Cryptology ePrint Archive, Report 2019/1436 (2019), https://eprint.iacr.org/2019/1436
32. Kirchner, P., Espitau, T., Fouque, P.A.: Fast reduction of algebraic lattices over cyclotomic fields. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 155–185. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_6
33. Kirchner, P., Espitau, T., Fouque, P.A.: Towards faster polynomial-time lattice reduction. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 760–790. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_26
34. Kirchner, P., Fouque, P.A.: Revisiting lattice attacks on overstretched NTRU parameters. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 3–26. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_1
35. Kirchner, P., Fouque, P.A.: Revisiting lattice attacks on overstretched NTRU parameters. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 3–26. Springer International Publishing, Cham (2017)
36. Koy, H., Schnorr, C.P.: Segment LLL-reduction of lattice bases. In: Silverman, J.H. (ed.) Cryptography and Lattices. pp. 67–80. Springer, Heidelberg (2001)
37. Krawczyk, H. (ed.): PKC 2014, LNCS, vol. 8383. Springer, Heidelberg (Mar 2014)
38. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. In: 24th FOCS. pp. 1–10. IEEE Computer Society Press (Nov 1983). https://doi.org/10.1109/SFCS.1983.70
39. Lee, M.S.: On the sparse subset sum problem from gentry-halevi's implementation of fully homomorphic encryption. Cryptology ePrint Archive, Report 2011/567 (2011), https://eprint.iacr.org/2011/567
40. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4), 515–534 (Dec 1982). https://doi.org/10.1007/BF01457454
41. May, A.: Using LLL-reduction for solving RSA and factorization problems. In: Nguyen and Vallée [50], pp. 315–348. https://doi.org/10.1007/978-3-642-02295-1
42. Micciancio, D., Regev, O.: Lattice-based Cryptography, pp. 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_5
43. Morel, I., Stehlé, D., Villard, G.: H-LLL: Using householder inside LLL. In: Proceedings of the 2009 International Symposium on Symbolic and Algebraic Compu-

tation. p. 271–278. ISSAC '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1576702.1576740

44. Nemec, M., Sýs, M., Svenda, P., Klinec, D., Matyas, V.: The return of coppersmith's attack: Practical factorization of widely used RSA moduli. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1631–1648. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3133969
45. Neumaier, A., Stehlé, D.: Faster LLL-type reduction of lattice bases. In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation. p. 373–380. ISSAC '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2930889.2930917
46. Nguyen, P.Q.: The two faces of lattices in cryptology (invited talk). In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, p. 313. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-45537-X_24
47. Nguyen, P.Q.: Hermite's constant and lattice algorithms. In: Nguyen and Vallée [50], pp. 19–69. https://doi.org/10.1007/978-3-642-02295-1
48. Nguyen, P.Q., Stehlé, D.: LLL on the average. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. pp. 238–256. Springer, Heidelberg (2006)
49. Nguyen, P.Q., Stehlé, D.: An LLL algorithm with quadratic complexity. SIAM Journal on Computing **39**(3), 874–903 (2009). https://doi.org/10.1137/070705702
50. Nguyen, P.Q., Vallée, B. (eds.): The LLL Algorithm - Survey and Applications. ISC, Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-02295-1
51. Novocin, A., Stehlé, D., Villard, G.: An LLL-reduction algorithm with quasi-linear time complexity: Extended abstract. In: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing. p. 403–412. STOC '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/1993636.1993691
52. Pataki, G., Tural, M.: On sublattice determinants in reduced bases (2008). https://doi.org/10.48550/ARXIV.0804.4014
53. Plantard, T., Susilo, W., Zhang, Z.: LLL for ideal lattices: re-evaluation of the security of Gentry-Halevi's FHE scheme. Designs, Codes and Cryptography **76**(2), 325–344 (Aug 2015). https://doi.org/10.1007/s10623-014-9957-1
54. Saruchi, Morel, I., Stehlé, D., Villard, G.: LLL reducing with the most significant bits. In: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation. p. 367–374. ISSAC '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2608628.2608645
55. Schönhage, A.: Fast reduction and composition of binary quadratic forms. In: Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation. p. 128–133. ISSAC '91, Association for Computing Machinery, New York, NY, USA (1991). https://doi.org/10.1145/120694.120711
56. Stewart, G.W., Sun, J.g.: Matrix perturbation theory (1990)
57. The FPLLL development team: fplll, a lattice reduction library, Version: 5.4.2 (2022), https://github.com/fplll/fplll, available at https://github.com/fplll/fplll
58. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_2
59. Xu, J., Hu, L., Sarkar, S.: Cryptanalysis of elliptic curve hidden number problem from PKC 2017. Designs, Codes and Cryptography **88**(2), 341–361 (Feb 2020). https://doi.org/10.1007/s10623-019-00685-y

A Extra Experiments

A.1 Validation of Heuristics

We perform two experiments to validate Heuristics 1 and Heuristic 2. We ran Algorithm 4 on a diverse set of test cases exhibiting a wide variety of reduction behavior. We logged the profiles at every iteration and computed global profiles $\bar{\Pi}(B^{(3i)}), \bar{\Pi}(B^{(3i+3)})$, and $\bar{\Pi}(B^{(3r)})$ for close to 100000 samples. Heuristic 1 predicts $\frac{\bar{\Pi}(B^{(3i)})-\bar{\Pi}(B^{(3(i+1))})}{\bar{\Pi}(B^{(3i)})-\bar{\Pi}(B^{(3r)})} = \Omega(1)$, so we plot the distribution of these values in Figure 8. At every single step, at least 27% of the total remaining potential was eliminated before the next round, supporting the $\Omega(1)$ claim.

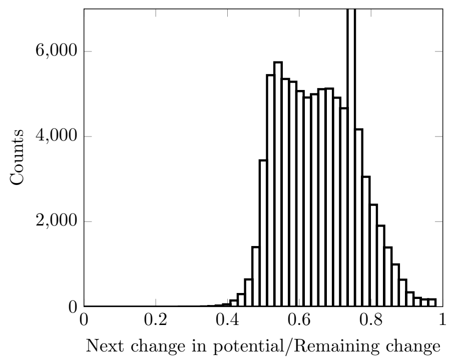

Fig. 8. Distribution of ratios of potential change. We experimentally examined the change in potential in each round of sublattice reductions and found that at least 27% and typically 50% or more of the remaining potential is eliminated with each round. There is a significant peak at 75% which corresponds to the dense sublattice structure particular to the reduction of NTRU lattices.

To validate Heuristic 2, we compared the change in potential in each round to $n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))$ and identified the minimum ratio over all of our samples. We find that the behavior is exponential in the final rounds as predicted by Heuristic 1, but in the last round $r-1$, the change in potential no longer appears to be bounded below. This matches and exceeds our Heuristic 2. In addition, the changes in potential are the same order of magnitude as the $\frac{1}{12}(\boldsymbol{\alpha}(n)-\boldsymbol{\alpha}(n/2))n^3$ change in potential when the slope goes from $\boldsymbol{\alpha}(n)$ to $\boldsymbol{\alpha}(n/2)$, so it is reasonable for these minima to be in this range.

A.2 NTRU

When the modulus of an NTRU instance is too large, it is called *overstretched* and can be solved efficiently by using reduction quality equivalent to LLL [2,23,35]. We run similar tests of Overstretched NTRU as those reported in [33]. However,

Table 3. Behavior in final rounds. We compare the smallest ratio of change in potential to $n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))$ in each round, and also compare the 1st and 5th percentile to show that the minimum is not an extreme outlier. There is an approximately exponential pattern up until round $r-1$, at which point the change in potential can become extremely small. This is stronger than our heuristic, which predicted exponential behavior up until round $r-2$.

Round	Minimum ratio	1%	5%
$r-5$	0.61490	0.80577	1.74633
$r-4$	0.17845	0.44872	1.38575
$r-3$	0.09072	0.40902	1.29064
$r-2$	0.04880	0.52662	3.75014
$r-1$	0.00001	0.08200	1.02669

we are not certain we have replicated their test cases properly. Based on the reported running time of fpLLL, it appears that the table in their Section 6.1 under-reports the dimension of NTRU lattices by a factor of 2. Assuming this was the case for these experiments, we perform the experiments with the reported dimension doubled.

Table 4. Performance on NTRU lattices.

Dimension	Bit size	fpLLL	KEF reported (s)	Ours (s)
512	80	2042	200	147
768	70	11173	600	558
1024	70	35300	2000	1204

For larger instances, Table 2 of [33] reduces sublattices of NTRU lattices. We are uncertain we have properly replicated their construction, as it appears that in one case their construction suggests taking a lattice of dimension 1024 and extracting a sublattice of dimension 1648.

Table 5. Performance on overstretched NTRU lattices.

Dimension	Bit size	KEF	Ours
2560	883	15800	8752
3086	11	839000	Memory error

A.3 Multivariate Coppersmith

There are a number of heuristic multivariate extensions to Coppersmith methods, which quickly produce large lattices with very high dimension. For our evaluation, we look at the Elliptic Curve Hidden Number Problem (ECHNP) as presented in [59]. In this problem, an oracle reveals a fraction of the bits of the x-coordinate of an elliptic curve point, and Coppersmith's method provides a way to recover the remainder. As the fraction of revealed bits becomes smaller, the dimension and entry size of the corresponding lattice basis quickly grows larger. We classify these bases as having irregular input and structured output.

We consider a 256-bit curve and leakage varying from 80% to 70% of the bits. The largest lattice we successfully reduced had rank 891 and 1794-bit entries. We reduced the same lattices using fpLLL and the implementation of [33]; the latter implementation crashed on the smallest instance and did not terminate on any larger test cases.

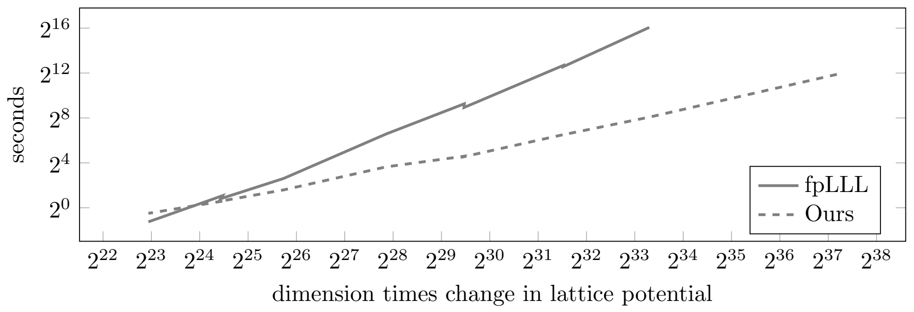

Fig. 9. Multivariate Coppersmith ECHNP lattices. The KEF [33] implementation did not terminate on any test cases so we omit it from the plot.

B Proofs of Theorems and Lemmas

B.1 Bounding the Decrease in Global Potential

As mentioned in Section 6, the global profile can be used to bound the remaining change in potential before the lattice is fully reduced. This can be used as an estimate of how much work remains in lattice reduction and the progress made so far. Although the original LLL paper bounds the change in log-potential by $\tilde{O}(n^2 \max_i \|\vec{b}_i\|)$ [40, Proposition 1.26], our bound is significantly stronger in several common cases. On top of this, our bound is easy to compute, and it is the tightest bound that can be computed from the profile.

Lemma 6 (Bounding the change in potential). *Let B be a lattice basis of rank n, and let B' be any other basis of the same lattice with smaller potential, for example the end result of lattice reducing B. Let $\ell_i = \log \|\vec{b}_i^*\|$ be the profile of B, and similarly for $\ell_{end,i}$ and B'. We can compute a new vector $\vec{\ell}_\nearrow$ which is the entries of $\vec{\ell}$ sorted in ascending order. Then the maximum change in potential $\Pi(\vec{\ell}) = \sum_{i=1}^{n}(n-i+1)\ell_i$ is bounded by*

$$\Pi(\vec{\ell}) - \Pi(\vec{\ell}_{end}) \leq \Pi(\vec{\ell}) - \Pi(\vec{\ell}_\nearrow).$$

This bound is easily computed from $\vec{\ell}$ alone. Furthermore, this bound is optimally tight among all bounds computable from $\vec{\ell}$ alone.

Our proof makes use of the following result from Pataki and Tural [52], which we rewrite in our notation.

Lemma 7 (Pataki-Tural). *Let B_1 and B_2 be bases for a lattice of rank n, and let $\vec{\ell}_1$ and $\vec{\ell}_2$ be their respective profiles. For any $r \leq n$ we have*

$$\sum_{i=1}^{r} \ell_{2,i} \geq \min_{I \subset \{1,\ldots,n\}, |I|=r} \sum_{i \in I} \ell_{1,i}.$$

Proof of Lemma 6. To find the upper bound, we would like to calculate the maximum possible change in potential

$$\max_{\vec{\ell}^* \in S} \left(\Pi(\vec{\ell}) - \Pi(\vec{\ell}^*) \right)$$

where S is the set of lattice profiles that could be reached from $\vec{\ell}$ via a LLL-like algorithm. We first prove a lower bound for $\min_{\vec{\ell}^*} \Pi(\vec{\ell}^*)$ and then show this bound is achieved by $\vec{\ell}^* = \vec{\ell}_\nearrow$.

We use the partial sums $s_k^* = \sum_{j=1}^{k} \ell_j^*$, observing that $\Pi(\vec{\ell}^*) = \sum_{k=1}^{n} s_k^*$. Lemma 7 gives a lower bound for each k which is achieved by $\vec{\ell}_\nearrow$:

$$s_k^* \geq \min_{I \subset \{1,\ldots,n\}, |I|=k} \sum_{i \in I} \ell_i = \sum_{i=1}^{k} \ell_{\nearrow,i}.$$

Therefore $\Pi(\vec{\ell}) \geq \Pi(\vec{\ell}_{end}) \geq \Pi(\vec{\ell}_{\nearrow})$, proving the bound.

We now show the bound is optimally tight for bounds solely computed from the profile. Given profile $\vec{\ell}$, we will give an example basis where the bound is achieved. Consider the lattice basis $B = \operatorname{diag}(2^{\ell_1}, \ldots, 2^{\ell_n})$ which has profile $\vec{\ell}$. Let B' be the basis which has the columns of B sorted by size. Clearly this is a basis of the same lattice, and the profile of B' is $\vec{\ell}_{\nearrow}$. Because we have constructed an example where the change in potential of a basis with profile $\vec{\ell}$ is $\Pi(\vec{\ell}) - \Pi(\vec{\ell}_{\nearrow})$, our bound is tight. □

B.2 Properties of Size-Reduced Lattices

Our definition of size reduction is asymptotic in nature, but despite the lack of additional specificity in Definition 1, this is sufficient to conclude several easy but interesting things about size-reduced bases.

Lemma 8. *Let B be a size-reduced basis of rank n, and let $\vec{\ell}$ be the profile. Then*

$$\log \kappa(B) = \mathit{spread}(B) + O(n).$$

Proof. Let $B = QDM$ be the decomposition in Definition 1. We have $\kappa(B) = \|B\|\|B^{-1}\| = \|QDM\|\|(QDM)^{-1}\| \leq \|D\|\|M\|\|M^{-1}\|\|D^{-1}\|$. Thus $\log \kappa(B) = \max_i \ell_i - \min_i \ell_i + O(n) = \operatorname{spread}(B) + O(n)$. □

Another useful property of size-reduced bases is that they allow us to bound condition numbers of the update matrices.

Lemma 9. *Let B be a basis of rank n and let U be a unimodular matrix such that BU is size-reduced. We bound the size of the largest element of U by*

$$\log \|U\|_{max} \leq \log \|U\| = O(\log \kappa(B) + n).$$

The same bound holds for U^{-1}.

Proof. First, note that $\|U\| \leq \|B^{-1}\|\|BU\|$ and $\|U^{-1}\| \leq \|B\|\|(BU)^{-1}\|$. This means $\log \kappa(U) \leq \log \kappa(B) + \log \kappa(BU)$, and

$$\log \kappa(U) = \log \kappa(B) + \operatorname{spread}(BU) + O(n) = O(\log \kappa(B) + n)$$

by Lemma 8. Since U is unimodular and has integer entries, $1 \leq \|U\|$ and $1 \leq \|U^{-1}\|$. Thus

$$\log \|U\| = O(\log \kappa(B) + n).$$

□

Lemma 10. *Let B be a size-reduced basis of rank n. The size of each vector is bounded by the profile $\vec{\ell}$. That is, for all $i \in \{1, \ldots, n\}$*

$$\log \|\vec{b}_i\| = \max_{1 \leq j \leq i} \ell_j + O(n).$$

Proof. We note that it is possible to decompose $B = QDM$ as the product of an orthogonal matrix, diagonal matrix, and unitriangular matrix. For any index i, $\|\vec{b}_i\| = \|(DM)_i\|$, and since M is upper-triangular, only the first i entries along the diagonal of D affect vector $(DM)_i$. This property implies $\|\vec{b}_i\| \leq \max_{1 \leq j \leq i} \|\vec{b}_j^*\|\|M_i\| \leq 2^{\max_{1 \leq j \leq i} \ell_j}\|M\|$. Thus $\log \|\vec{b}_i\| = \max_{1 \leq j \leq i} \ell_j + O(n)$. □

B.3 Properties of Reduced Lattices

We recall the statement of Theorem 2.

Theorem 2. *Let B be a α-lattice-reduced rank-n basis satisfying our new definition of reduction quality. Let $\vec{b}_i^*$ denote the i^{th} Gram-Schmidt vector and $\lambda_i(B)$ denote the i^{th} successive minimum of the lattice spanned by B. Then B satisfies*

1. $\|\vec{b}_1\| \leq 2^{\alpha n}(\det B)^{1/n}$.
2. $\|\vec{b}_n^*\| \geq 2^{-\alpha n}(\det B)^{1/n}$.
3. *For all* $i \in \{1, \ldots, n\}, \|\vec{b}_i\| \leq 2^{\alpha n + O(n)}\lambda_i(B)$.
4. $\|\vec{b}_1\| \times \cdots \times \|\vec{b}_n\| \leq 2^{\alpha n^2 + O(n^2)} \det B$.

Proof of Theorem 2. Let $\vec{\ell}$ be the profile or B.

To prove 1, we have $\ell_1 - \min_i \ell_i \leq \text{drop}(B) \leq \alpha n$ and $\frac{1}{n}\log(\det B) = \frac{1}{n}\sum_{i=1}^n \ell_i \geq \min_i \ell_i \geq \ell_1 - \alpha n$. Exponentiating gives $\det(B)^{1/n} \geq \|\vec{b}_1\| 2^{-\alpha n} \Rightarrow \|\vec{b}_1\| \leq 2^{\alpha n}\det(B)^{1/n}$.

The proof of 2 follows in the same way: $\max_i \ell_i - \ell_n \leq \text{drop}(B) \leq \alpha n$ and $\frac{1}{n}\log(\det B) = \frac{1}{n}\sum_{i=1}^n \ell_i \leq \max_i \ell_i \leq \ell_n + \alpha n$. Exponentiating gives $\|\vec{b}_n^*\| \leq 2^{-\alpha n}\det(B)^{1/n}$.

To prove 3, we first have $\min_{i \leq j \leq n} \ell_j \leq \log \lambda_i(B)$ by [47, Lemma 7]. Because B is α-lattice-reduced, it is size-reduced. By Lemma 10,

$$\begin{aligned}\log \|\vec{b}_i\| &= \max_{1 \leq j \leq i} \ell_j + O(n) \\ &= (\max_{1 \leq j \leq i} \ell_j - \min_{i \leq j \leq n} \ell_j) + \min_{i \leq j \leq n} \ell_j + O(n) \\ &= \alpha n + \log \lambda_i(B) + O(n).\end{aligned}$$

Exponentiating gives $\|\vec{b}_i\| \leq 2^{\alpha n + O(n)}\lambda_i(L)$.

To prove 4, note that we have $\log \|\vec{b}_i\| \leq \max_{1 \leq j \leq i} \ell_j + O(n) \leq \ell_i + \alpha n + O(n)$. Summing over i gives

$$\sum_{i=1}^n \log \|\vec{b}_i\| \leq O(n^2) + \alpha n^2 + \sum_{i=1}^n \ell_i$$

$$\Rightarrow \|\vec{b}_1\| \times \cdots \times \|\vec{b}_n\| \leq 2^{\alpha n^2 + O(n^2)} \det B.$$

□

B.4 Properties of Similar Lattices

We recall the statement of Lemma 1.

Lemma 1. *Let $B_1, B_2, B_3 \in \mathbb{R}^{n\times n}$, and let $\gamma, \gamma' \geq 0$. Then*

1. *$B_1 \approx_\gamma B_2 \Leftrightarrow B_2 \approx_\gamma B_1$*
2. *$B_1 \approx_\gamma B_2$ and $B_2 \approx_{\gamma'} B_3 \Rightarrow B_1 \approx_{\gamma+\gamma'} B_3$.*
3. *If $B_1 \approx_\gamma B_2$, then to first order*

$$\|B_2\| \leq (1+\gamma)\,\|B_1\| \quad \text{and} \quad \|B_2^{-1}\| \leq (1+\kappa(B_1)\gamma)\,\|B_1^{-1}\|$$

4. *Let $\vec{\ell}_1$ and $\vec{\ell}_2$ be the profiles of B_1 and B_2, and let $B_1 \approx_\gamma B_2$. Then*

$$|\ell_{1,i} - \ell_{2,i}| < \sqrt{2n^3}\kappa(B_1)\gamma \text{ for all } i \in \{1, \ldots, n\}.$$

Proof.

Item 1. By assumption in the forward direction, there exists orthogonal Q such that for all nonzero $\vec{x}$, $\|B_1\vec{x} - QB_2\vec{x}\| \leq \gamma\|B_1\vec{x}\|$. Using the triangle inequality, $\|B_1\vec{x}\| - \|QB_2\vec{x}\| \leq \|B_1\vec{x} - QB_2\vec{x}\|$. Combining these, we get $\|B_1\vec{x}\| \leq \frac{1}{1-\gamma}\|B_2\vec{x}\|$ To first order in γ, this is $(1+\gamma)\|B_2\vec{x}\|$. Finally, $\|B_1\vec{x} - QB_2\vec{x}\| \leq \gamma(1+\gamma)\|B_2\vec{x}\|$ gives the required bound to first order. The reverse direction is identical.

Item 2. We have by assumption the existence of Q, Q' such that for nonzero $\vec{x}$,

$$\|B_1\vec{x} - QB_2\vec{x}\| \leq \gamma\|B_1\vec{x}\|$$
$$\|B_2\vec{x} - Q'B_3\vec{x}\| \leq \gamma'\|B_2\vec{x}\|.$$

Thus,

$$\begin{aligned}
&\|B_1\vec{x} - QQ'B_3\vec{x}\| \\
&\leq \|B_1\vec{x} - QB_2\vec{x}\| + \|QB_2\vec{x} - QQ'B_3\vec{x}\| \\
&= \|B_1\vec{x} - QB_2\vec{x}\| + \|B_2\vec{x} - Q'B_3\vec{x}\| \\
&\leq \|B_1\vec{x} - QB_2\vec{x}\| + \gamma'\|B_2\vec{x}\| \\
&\leq \|B_1\vec{x} - QB_2\vec{x}\| + \gamma'(\|B_1\vec{x} - QB_2\vec{x}\| + \|B_1\vec{x}\|) \\
&\leq (1+\gamma')\gamma\|B_1\vec{x}\| + \gamma'\|B_1\vec{x}\| \\
&\leq (\gamma+\gamma')\|B_1\vec{x}\|
\end{aligned}$$

after discarding the higher-order γ terms.

Item 3. We have by assumption the existence of Q such that for nonzero $\vec{x}$,

$$\|B_1\vec{x} - QB_2\vec{x}\| \leq \gamma\|B_1\vec{x}\|.$$

By definition of the spectral norm, $\|B_1 - QB_2\| = \max_{\vec{x}\neq 0}\|B_1\vec{x} - QB_2\vec{x}\|/\|\vec{x}\| \leq \max_{\vec{x}\neq 0}\gamma\|B_1\vec{x}\|/\|\vec{x}\| = \gamma\|B_1\|$. This means

$$\|B_2\| = \|QB_2\| \leq \|B_1 - QB_2\| + \|B_1\| \leq (1+\gamma)\|B_1\|.$$

For the inverse,

$$\|(QB_2)^{-1} - B_1^{-1}\| \le \frac{\kappa(B_1)\frac{\|B_1 - QB_2\|}{\|B_1\|}}{1 - \kappa(B_1)\frac{\|B_1 - QB_2\|}{\|B_1\|}} \|B_1^{-1}\|$$

by Stewart and Sun [56, Corollary III.2.7]. Therefore

$$\|B_2^{-1}\| \le \|B_1^{-1}\| + \|(QB_2)^{-1} - B_1^{-1}\| < \left(1 + \frac{\kappa(B_1)\gamma}{1 - \kappa(B_1)\gamma}\right) \|B_1^{-1}\|$$

and the condition holds to first order.

Item 4. We have $\|B_1\vec{x} - QB_2\vec{x}\| \le \gamma\|B_1\vec{x}\|$. We set $B_1 + \Delta B_1 = QB_2$ and use standard results from [56]. We have

$$\|\Delta B_1\| = \max_{\vec{x}\neq 0} \frac{\|\Delta B_1\vec{x}\|}{\|\vec{x}\|} = \max_{\vec{x}\neq 0} \frac{\|B_1\vec{x} - QB_2\vec{x}\|}{\|B_1\vec{x}\|} \frac{\|B_1\vec{x}\|}{\|\vec{x}\|} \le \gamma\|B_1\|.$$

Therefore, $\|\Delta B_1\|_F \le \gamma\sqrt{n}\|B_1\|$. If we let R and $\hat{R}$ be the R-factors of B_1 and QB_2D, then

$$\begin{aligned} |R_{ii} - \hat{R}_{ii}|/|R_{ii}| &\le n(|R_{ii} - \hat{R}_{ii}|)/\|R\| \\ &\le n\|R - \hat{R}\|/\|R\| \\ &\le \sqrt{2}n\|B_1\|\|B_1^{-1}\|\|\Delta B_1\|_F/\|R\| \\ &\le \gamma\sqrt{2n^3}\|B_1\|\|B_1^{-1}\|. \end{aligned}$$

This can be used to relate the profiles $\vec{\ell_1}$ and $\vec{\ell_2}$ of B_1 and B_2 in terms of γ and the condition number of B_1. Thus, to first order,

$$\begin{aligned} \ell_{2,i} - \ell_{1,i} &= \log|\hat{R}_{ii}| - \log|R_{ii}| \\ &\le \log((|R_{ii} - \hat{R}_{ii}| + |R_{ii}|)/|R_{ii}|) \\ &= \log(1 + |R_{ii} - \hat{R}_{ii}|/|R_{ii}|) \\ &\le |R_{ii} - \hat{R}_{ii}|/|R_{ii}| \\ &\le \gamma\sqrt{2n^3}\kappa(B_1). \end{aligned}$$

A similar bound applies to $\ell_{1,i} - \ell_{2,i}$. □

We also develop the following results, which are helpful for analyzing our algorithm.

Lemma 11 (Lattice Similarity via Matrix Perturbation). *Let $B, \hat{B} \in \mathbb{R}^{n\times n}$ be bases of rank n and γ such that $B \approx_\gamma \hat{B}$. Then there exists Q such that*

$$\|B - Q\hat{B}\| \le \gamma\|B\|.$$

Conversely, let $B, \hat{B} \in \mathbb{R}^{n\times n}$ be bases of rank n and Q an orthogonal matrix. For $\gamma \ge \|B - Q\hat{B}\|\|B^{-1}\|$ we have

$$B \approx_\gamma \hat{B}.$$

Proof. For the first claim, the definition of lattice similarity establishes the existence of Q such that $\|B\vec{x} - Q\hat{B}\vec{x}\| \le \gamma\|B\vec{x}\|$ for all nonzero $\vec{x}$. Thus $\|B - Q\hat{B}\| \le \max_{\vec{x}\neq 0} \|(B - Q\hat{B})\vec{x}\|/\|\vec{x}\| \le \max_{\vec{x}\neq 0} \gamma\|B\vec{x}\|/\|\vec{x}\| = \gamma\|B\|$.

For the second claim, we wish to bound $\|B\vec{x} - Q\hat{B}\vec{x}\|/\|B\vec{x}\|$ by γ for all nonzero $\vec{x}$. We have

$$\max_{\vec{x}\neq 0} \frac{\|B\vec{x} - Q\hat{B}\vec{x}\|}{\|B\vec{x}\|} \le \|B - Q\hat{B}\| \max_{\vec{x}\neq 0} \frac{\|\vec{x}\|}{\|B\vec{x}\|} = \|B - Q\hat{B}\|\|B^{-1}\| \le \gamma.$$

□

Lemma 12 (Lattice Similarity after Rounding). *Let B be a basis of rank n with $\kappa(B) < 2^C$, and let $\lfloor B\rceil \in \mathbb{Z}^{n\times n}$ be the same basis with each entry rounded to the closest integer. For $\gamma \ge n2^C/\|B\|_{max}$, $B \approx_\gamma \lfloor B\rceil$.*

Proof. Rounding gives $\|B - \lfloor B\rceil\| \le n\|B - \lfloor B\rceil\|_{max} = n/2$. By Lemma 11, we therefore have $B \approx_\gamma \lfloor B\rceil$ for $\gamma \ge n\|B^{-1}\|/2$. Since $\|B^{-1}\| < 2^C/\|B\| \le 2^C/\|B\|_{max}$, the condition holds for $\gamma \ge n2^{C-1}/\|B\|_{max}$. □

B.5 The Compression Algorithm

In this section, we analyze the stability and running time of algorithms involved in lattice basis compression. We begin by giving example methods to perform QR factorization and size reduction, and we analyze their behavior under our framework of lattice basis similarity.

Lemma 13 (Running time of QR factorization). *Let $B \in \mathbb{Z}^{n\times n}$ be a basis of rank n where $\kappa(B) < 2^C$. For any γ, there exists an algorithm that computes upper-triangular R in floating point representation such that*

$$R \approx_\gamma B \text{ and } \kappa(R) = 2^{O(C)}$$

This operation takes time $O(n^3(C - \log(\gamma) + \log(n))^{1+\varepsilon})$.

Proof. Our algorithm performs Householder QR factorization of B with floating point precision p', to be specified later. Let R be the output of this algorithm. By [28, Theorem 19.4], there exists orthogonal Q such that $B + \Delta B = QR$ where $\|\Delta\vec{b}_j\| \le \tilde{\gamma}'_{n^2}\|\vec{b}_j\|$ for $j \in \{1, \ldots, n\}$ and $\tilde{\gamma}'_{n^2}$ as defined in [28], noting $-\log(\tilde{\gamma}'_{n^2}) + \log(n^2) = \Theta(p')$ to first order. We have $\|B - QRI\| = \|\Delta B\| \le \sum_{j=1}^{n} \|\Delta\vec{b}_j\| \le n\tilde{\gamma}'_{n^2} \max_j \|\vec{b}_j\| \le n\tilde{\gamma}'_{n^2}\|B\|$. By Lemma 11, $B \approx_{\gamma'} R$ for $\gamma' \ge n\tilde{\gamma}'_{n^2}2^C > n\tilde{\gamma}'_{n^2}\kappa(B)$. What remains is choosing γ' based on γ and choosing p' such that $\gamma' \ge n\tilde{\gamma}'_{n^2}2^C$.

We choose $\gamma' = 2^{-\Theta(C - \log(\gamma))}$. This is done so that $B \approx_{\gamma'} R$ implies $\kappa(R) = O(\kappa(B)\gamma') = 2^{O(C)}$. It then suffices to take $p' = \Theta(C - \log(\gamma) + \log(n))$. Householder factorization involves $O(n^3)$ floating point operations, so the overall running time is is $O(n^3(C - \log(\gamma) + \log(n))^{1+\varepsilon})$. □

Algorithm 5: `SizeReduce` adapted from [45,47]

```
Input  : Upper-triangular basis $B \in \mathbb{Z}^{n\times n}$ and bound $C > \log(\kappa(B))$
Output: Basis $B'$ and unimodular $U$ such that $B' = BU$ is size-reduced
1  $b_{i,j,1} \leftarrow B_{i,j}$ for $i, j \in \{1, \ldots, n\}$
2  $U \leftarrow I_n$
3  $p' \leftarrow \lceil C + \log(n) \rceil + 2$
4  for $j \leftarrow 1$ to $n$ do
5      for $i \leftarrow j-1$ to $1$ do
6          $q_{i,j} \leftarrow \lfloor b_{i,j,(j-i)}/b_{i,i,1} \rceil$
7          for $k \leftarrow 1$ to $i$ do
8              $b_{k,j,(j-i+1)} \leftarrow b_{k,j,(j-i)} - q_{i,j} b_{k,i,(i-k+1)}$
9              $U_{k,j} \leftarrow U_{k,j} - q_{i,j} U_{k,i} \pmod{2^{p'}}$
10 $B'_{i,j} \leftarrow b_{i,j,(j-i+1)}$
11 return $B', U$
```

We also analyze the size-reduction operation in depth. We consider size-reduction on integer, upper-triangular lattices, which means we do not need to worry about numerical precision or representing large rational numbers in the Gram-Schmidt decomposition. While our results are not difficult to prove, we are not aware of a good reference for the running time in this context, so we present our analysis of Algorithm 5 here.

Lemma 14 (Running Time of Size Reduction). *Let $B \in \mathbb{Z}^{n\times n}$ be an upper-triangular basis of rank n and $C > \log(\kappa(B))$ be an upper bound on the condition number. Let $p > \log \|B\|_{max}$ bound the input size. On this input, Algorithm 5 takes time $O(n^3(C + p + n)^{1+\varepsilon})$ to size-reduce B.*

Proof. Clearly, there are $O(n^3)$ arithmetic operations, so we wish to bound the sizes of integers encountered during computation. We first bound the q and b variables by tracking how each is updated throughout the course of the algorithm, and then separately bound the size of entries of U.

Unsurprisingly, we use the size-reduction property to show that intermediate values of $b_{k,j,(j-i+1)}$ are have reduced size. Note that the (i, j) entry in the basis B is updated $j - i$ times before the value in B' is computed. We have $|q_{i,j}| = \lfloor b_{i,j,(j-i)}/b_{i,i,1} \rceil \leq |b_{i,j,(j-i)}|/|b_{i,i,1}|+1$, so it is helpful to have a bound on $|b_{k,j,(j-k)}|$. Since $b_{k,j,(j-i+1)} = b_{k,j,(j-i)} - q_{i,j} b_{k,i,(i-k+1)}$, we have $|b_{k,j,(j-i+1)}| \leq |b_{k,j,(j-i)}| + |q_{i,j}||b_{k,i,(i-k+1)}|$, and size-reducedness gives $|b_{k,i,(i-k+1)}| < |b_{k,k,1}|$.

We therefore have

$$\begin{aligned}
|b_{k,j,(j-k)}| &< |q_{(k+1),j}||b_{k,k,1}| + |b_{k,j,(j-k-1)}| \\
&< \sum_{i=k+1}^{j-1} |q_{i,j}||b_{k,k,1}| + |b_{k,j,1}| \\
&< |b_{k,k,1}| \left(\sum_{i=k+1}^{j-1} \frac{|b_{i,j,(j-i)}|}{|b_{i,i,1}|} + 1 \right) + |b_{k,j,1}| \\
&< |b_{k,k,1}| \left(\sum_{i=k+1}^{j-1} \frac{|b_{i,j,(j-i)}|}{|b_{i,i,1}|} \right) + (j-k)2^p \\
\Rightarrow \frac{|b_{k,j,(j-k)}|}{|b_{k,k,1}|} &< \left(\sum_{i=k+1}^{j-1} \frac{|b_{i,j,(j-i)}|}{|b_{i,i,1}|} \right) + (j-k)2^p / |b_{k,k,1}| \\
&< \left(\sum_{i=k+1}^{j-1} \frac{|b_{i,j,(j-i)}|}{|b_{i,i,1}|} \right) + (j-k)2^p.
\end{aligned}$$

This gives a recursive relation that is satisfied by $\frac{|b_{i,j,(j-i)}|}{|b_{i,i,1}|} < (2^{j-i} - 1)2^p$ for $1 \leq i < j \leq n$. Thus we have $|b_{k,j,(j-k)}| < 2^{n+2p}$ and $|b_{k,j,(j-i+1)}| < |b_{k,j,j}|$. This means the largest value of $b_{i,j,k}$ (and $q_{i,j}$) can be stored with $\lceil n + 2p \rceil + 1$ bits.

To bound the precision needed to calculate the final U, it suffices to determine $p' > \log \|U\|_{max}$ and perform all updates to U modulo $2^{p'}$. We could use Lemma 9 to do this, but because additional information about BU is known, we can derive an exact bound. We have $\|U\|_{max} \leq \|U\| \leq \|B^{-1}\|\|BU\|$ and $\|BU\|^2 \leq \|BU\|_F^2 \leq n \sum_{i=1}^n B_{i,i}^2 \leq n^2 \|B\|_{max}^2 \leq n^2 \|B\|^2$. Because $\|U\|_{max} \leq n2^C$, $p' = \lceil C + \log(n) \rceil + 2$ bits suffice to calculate each entry of U.

Combining these, we see that the largest value encountered during execution has $O(C+p+n)$ bits, so the running time of Algorithm 5 is $O(n^3(C+p+n)^{1+\varepsilon})$.
□

Algorithm 6: `ScaleAndRound`

```
  Input  : $B \in \mathbb{R}^{n \times n}$ in floating point representation, $\gamma \leq 2^{-1}$, $C > \log \kappa(B)$
  Output: $B' \in \mathbb{Z}^{n \times n}$ in integer representation, $d \in \mathbb{Z}$ with $B \approx_\gamma B'2^d$
1 $\gamma' \leftarrow 2^{-C}\gamma$
2 $d \leftarrow \lceil C + \log(n) - \log(\gamma') - \log(\|B\|_{max}) \rceil + 1$
3 $B_1 \leftarrow 2^d B$
4 $B' \leftarrow \lfloor B_1 \rceil$
5 return $B', d$
```

Next, we examine an auxiliary method that aids in the compression of lattice bases. Our compression algorithm frequently needs to convert a basis from floating point representation to integer representation, and abstracting out this method gives useful stability bounds.

Lemma 15. *On input B, γ, and $C > \log \kappa(B)$, Algorithm 6 (`ScaleAndRound`) returns an integer basis B' and integer scaling factor d satisfying $B \approx_\gamma B'2^d$. The resulting basis B' has bounded condition number and entry size, satisfying $\kappa(B') = 2^{O(C)}$ and $\|B'\|_{max} = 2^{O(C+\log(n)-\log(\gamma))}$. The running time on this input is $O(n^2(C + \log(n) - \log(\gamma))$.*

Proof. After computing γ' and d, Algorithm 6 scales the floating point values in B by 2^d, which can be done exactly. This gives $B \approx_0 2^{-d}B_1$. We have $\|B_1\|_{max} > n2^C/\gamma'$ and $\|B_1\|_{max} < n2^{C+2}/\gamma'$.

After rounding the values in B_1 to obtain B', Lemma 12 establishes that since $\gamma' > n2^C/\|B_1\|_{max}$, $B_1 \approx_{\gamma'} B'$. We bound $\kappa(B') < 2^{C'}$ using Lemma 1. We have

$$\kappa(B') \leq (1 + \kappa(B_1)\gamma')\,\kappa(B_1) < (1+\gamma)\,\kappa(B_1)$$

so $\kappa(B') \approx \kappa(B_1) = \kappa(B) < 2^C$ up to first order in γ, and we can compute bound $\log \kappa(B') = C + \log(1+\gamma)$. Since $\gamma \leq 2^{-1}$ and condition numbers are always at least 1, $C_1 = O(C)$.

We bound $\|B'\|_{max} < 2^{p'}$ for $p' = \Theta(C+\log(n)-\log(\gamma))$ by noting $\|B'\|_{max} \leq \|B_1\|_{max} + 1 \leq n2^{C+2}/\gamma' + 1 \leq n2^{2C+3}/\gamma$.

The running time is dominated by the cost of outputting the n^2 values of size $O(C + \log(n) - \log(\gamma))$ bits. □

These tools allow us to construct the full compression algorithm. We describe the full proven algorithm in Algorithm 7 and prove the claims made in Lemma 2, which we recall here.

Lemma 2. *Let $\omega \in (2, 3]$ and ε be global parameters bounding the complexity of algorithms as follows. We assume there exists algorithm `QR` that on input B, $C > \log \kappa(B)$, returns a γ-similar basis R with $\kappa(R) = 2^{O(C)}$ in time $O(n^\omega(C - \log\gamma + \log n)^{1+\varepsilon})$. We also assume that there exists algorithm `SizeReduce` that size reduces an integer, upper-triangular basis B (with $C > \log \kappa(B)$) in time $O(n^\omega(C + \log\|B\|_{max} + n)^{1+\varepsilon})$. Finally, we assume there exists a matrix multiplication algorithm which computes product A_1A_2 of two $n \times n$ matrices in time $O(n^\omega(\log\|A_1\|_{max} + \log\|A_2\|_{max})^{1+\varepsilon})$.*

Algorithm `CompressLattice` returns a compressed basis $\hat{B}$, diagonal $D = diag(2^{d_1}, \ldots, 2^{d_n})$, and unimodular U satisfying $\hat{B} \approx_{2^{-O(drop(B)+n)}\gamma} BUD$. In addition, if $\vec{\ell}_B$ is the profile of B and $\vec{\ell}_{\hat{B}}$ is the profile of $\hat{B}$, then we have $\left|\vec{\ell}_{\hat{B},i} - (\vec{\ell}_{B,i} + d_i)\right| \leq \gamma$. This algorithm takes time $O(n^\omega(C - \log\gamma + n)^{1+\varepsilon})$.

Proof. Algorithm 7 performs a number of manipulations of basis B. Throughout the execution of the algorithm, we must show that the condition number and size of entries remain bounded, and we must also establish that output lattice basis is similar to the input basis.

Algorithm 7: `CompressLattice`

```
Input  : $B \in \mathbb{Z}^{n\times n}$, $\gamma \le 1/2$, $C > \log\kappa(B)$
Output: $\hat{B}$ compressed, $U \in \mathbb{Z}^{n\times n}$, $D = diag(2^{d_1}, \ldots, 2^{d_n})$ with $d_i \in \mathbb{Z}$
        satisfying $\hat{B} \approx_{2^{-O(\mathrm{drop}(B)+n)}\gamma} BUD$. The profile of $\hat{B}$ is close to the
        ($D$-scaled) profile of $B$ with absolute error $\gamma$.
1  $\gamma' \leftarrow \gamma 2^{-C-1}/\sqrt{2n^3}$
2  $B_1 \leftarrow \mathtt{QR}(B, \gamma'/3)$ ;                                  // $\kappa(B) = 2^{O(C)}$
3  $B_2, s' \leftarrow \mathtt{ScaleAndRound}(B_1, \gamma'/3)$ ;                  // $\kappa(B_1) = 2^{O(C)}$
4  $B_3, U' \leftarrow \mathtt{SizeReduce}(B_2)$ ;                               // $\kappa(B_2) = 2^{O(C)}$
5  Compute profile $\ell_i \leftarrow \log(|B_3|_{i,i})$ for $i \in \{1, \ldots, n\}$
6  $\vec{d} \leftarrow 0, \vec{\ell'} \leftarrow \vec{\ell}$
7  for $k \leftarrow 1$ to $n$ do
8      if $\max_{1\le i\le k} \ell_i + 1 < \min_{k+1\le i\le n} \ell_i$ then
9          $t \leftarrow \lfloor \min_{k+1\le i\le n} \ell_i - \max_{1\le i\le k} \ell_i \rfloor$
10         $d_i \leftarrow d_i - t$ for $k+1 \le i \le n$
11         $\ell'_i \leftarrow \ell'_i - t$ for $k+1 \le i \le n$
12 $D' \leftarrow diag(2^{d_1}, \ldots, 2^{d_n})$
13 $B_4 \leftarrow B_3 D'$ ;                                   // $\kappa(B_3) = 2^{O(\mathtt{spread}(\vec{\ell})+n)}$
14 $B_5, s'' \leftarrow \mathtt{ScaleAndRound}(B_4, \gamma'/3)$ ;      // $\kappa(B_4) = 2^{O(\mathtt{spread}(\vec{\ell})+n)}$
15 $B_6, U'' \leftarrow \mathtt{SizeReduce}(B_5)$ ;                   // $\kappa(B_5) = 2^{O(\mathtt{spread}(\vec{\ell})+n)}$
16 $\gamma'' \leftarrow \gamma 2^{-O(\mathrm{spread}(\vec{\ell'})+n)}$
17 $B_7, s''' \leftarrow \mathtt{ScaleAndRound}(B_6, \gamma'')$ ;      // $\kappa(B_6) = 2^{O(\mathtt{spread}(\vec{\ell'})+n)}$
18 $B_8, U''' \leftarrow \mathtt{SizeReduce}(B_7)$ ;                  // $\kappa(B_7) = 2^{O(\mathtt{spread}(\vec{\ell'})+n)}$
19 $U \leftarrow U'D'U''U'''D'^{-1}$
20 $D \leftarrow 2^{-(s'+s''+s''')}D'$
21 $\hat{B} \leftarrow B_8$
22 return $\hat{B}, U, D$
```

We begin by QR-factorizing B to quality $\gamma'/3$. Since $\kappa(B) = 2^{O(C)}$, the assumption implies that $B_1 \approx_{\gamma'/3} B$ and $\kappa(B_1) = 2^{O(C)}$. The operation takes time $O(n^\omega(C - \log\gamma' + \log n)^{1+\varepsilon})$.

Scaling and rounding returns B_2 and s' satisfying $B_1 \approx_{\gamma'/3} 2^{s'} B_2$, $\kappa(B_2) = 2^{O(c)}$, and $\|B_2\|_{max} = 2^{O(C+\log n - \log\gamma')}$. The running time of scaling and rounding is negligible compared to QR-factorization and size reduction.

Size reduction of B_2 takes time $O(n^\omega(C - \log\gamma' + n)^{1+\varepsilon})$ and returns B_3, U'. Because B_3 is size-reduced, $\kappa(B_3) = 2^{O(\text{spread}(\vec{\ell})+n)}$ and $\kappa(U') = 2^{O(C+n)}$.

Computing the profile and scaling coefficients $\vec{d}$ as described takes $O(n^2)$ operations on logarithmically small values, so it does not contribute to the overall running time. We note it is possible to do this in $O(n)$ operations by considering the cumulative maximum from the left and cumulative minimum from the right. We have $B_4 \approx_0 B_3D'$. Since $\|B_4\| \leq \|B_3\|\|D'\|$ and $\|B_4^{-1}\| \leq \|B_3^{-1}\|\|D'^{-1}\|$, $\kappa(B_4) = \kappa(B_3)2^{O(\max_i d_i - \min_i d_i)} = 2^{O(\text{spread}(\vec{\ell}))+n}$.

Scaling and rounding of B_4 returns B_5 and s'' satisfying $B_4 \approx_{\gamma'/3} 2^{s''} B_5$. We have $\kappa(B_5) = 2^{O(\text{spread}(\vec{\ell})+n)}$ and $\|B_5\|_{max} = 2^{O(\text{spread}(\vec{\ell})+n-\log\gamma')}$. The running time is again negligible.

Size reduction of B_5 takes time $O(n^\omega(\text{spread}(\vec{\ell}) - \log\gamma' + n)^{1+\varepsilon})$ and returns B_6, U''. Because B_6 is size-reduced, $\kappa(B_6) = 2^{O(\text{spread}(\vec{\ell}')+n)}$ and $\kappa(U'') = 2^{O(\text{spread}(\vec{\ell})+n)}$.

Scaling and rounding of B_6 returns B_7 and s''' satisfying $B_6 \approx_{\gamma''} 2^{s'''} B_7$. We have $\kappa(B_7) = 2^{O(\text{spread}(\vec{\ell}')+n)}$ and $\|B_7\|_{max} = 2^{O(\text{spread}(\vec{\ell}')-\log\gamma''+n)}$.

The final size reduction of B_7 takes time $O(n^\omega(\text{spread}(\vec{\ell}') - \log\gamma'' + n)^{1+\varepsilon})$ and returns B_8, U'''. Because B_8 is size-reduced, $\kappa(B_8) = 2^{O(\text{spread}(\vec{\ell}')+n)}$ and $\kappa(U''') = 2^{O(\text{spread}(\vec{\ell}')+n)}$.

It remains to show that U and D satisfy $\hat{B} \approx_\gamma BUD$, $\hat{B} = B_8$ is compressed, the profiles are close to absolute error γ, and the overall running time is bounded. We have $B \approx_{\gamma'/3} B_1$, $B_1 \approx_{\gamma'/3} 2^{s'} B_2$, $B_2U' = B_3$, $B_3 = B_4D'^{-1}$, $B_4 \approx_{\gamma'/3} 2^{s''} B_5$, $B_5U'' = B_6$, $B_6 \approx_{\gamma''} 2^{s'''} B_7$, and $B_7U''' = B_8$. Combining these, we get $BU'D'U''D'^{-1} \approx_{\gamma'} 2^{s'+s''} B_6 D'^{-1}$ and $B_6U''' \approx_{\gamma''} 2^{s'''} B_8$. Altogether, this is $BU'D'U''U'''D'^{-1} \approx_{\gamma'+\gamma''} B_8 2^{s'+s''+s'''} D'^{-1}$, simplifying to $\hat{B} \approx_{\gamma'+\gamma''} BUD$. Since $\gamma' + \gamma'' \leq 2^{-O(\text{spread}(\vec{\ell}')+n)}\gamma = 2^{-O(\text{drop}(B)+n)}\gamma$, this is $\hat{B} \approx_{2^{-O(\text{drop}(B)+n)}\gamma} BUD$.

Next, let $\vec{\ell}_B$ be the profile of B, let $\vec{\ell}_{B_6}$ be the profile of B_6, and let and let $\vec{\ell}_{\hat{B}}$ be the profile of $\hat{B}$. First, we have $BU'D'U''D'^{-1} \approx_{\gamma'} 2^{s'+s''} B_6 D'^{-1}$. Observe that the profile of the left hand side is $\vec{\ell}_B$, because size reduction operations do not alter the profile. The profile of the right hand side is $\vec{\ell}_{B_6} + s' + s'' - \vec{d}'$. By Lemma 1, $|\ell_{B,i} - (\ell_{B_6,i} + s' + s'' - d_i')| \leq \sqrt{2n^3}\kappa(B)\gamma' \leq \gamma/2$. By the same logic, $B_6U''' \approx_{\gamma''} 2^{s'''} B_8$ implies $|\ell_{B_6,i} - (\ell_{\hat{B},i} + s''')| \leq \sqrt{2n^3}\kappa(B_6)\gamma'' \leq \gamma/2$. Thus $|\ell_{B,i} - (\ell_{\hat{B},i} - d_i)| \leq \gamma$, and the profile of $\hat{B}$ is close to the (D-scaled) profile of B with absolute error γ.

We note that $\hat{B}$ has integer entries, is upper-triangular, is non-singular, and is size-reduced. The size of entries in $\hat{B}$ is $\log \|\hat{B}\|_{max} = O(\text{spread}(\vec{\ell}') - \log\gamma + n)$. By the similarity of the bases, $O(\text{spread}(\vec{\ell}') - \log\gamma + n) = O(\text{drop}(\hat{B}) + n - \log\gamma)$. This means that $\hat{B}$ is compressed.

The running time of the algorithm is dominated by the calls to QR factorization and size reduction. The total running time is $O(n^\omega(C - \log\gamma + n)^{1+\varepsilon})$. $\square$

B.6 The LR Reduction Algorithm

With the tools in place to evaluate lattice similarity and efficiently perform lattice basis compression, we now turn to understanding the subroutines involving lattice reduction. We begin by proving Lemma 3, which we recall here.

Lemma 3. *Consider compressed basis $B^{(k)}$, sublattice index $[i : j]$, approximation quality γ, lattice reduction function* `LatRed`*, and lattice reduction quality $\boldsymbol{\alpha}$. Algorithm 3 returns compressed basis $B^{(k+1)}$, unimodular U, and diagonal D such that for $\gamma' = 2^{-O(drop(B^{(k+1)})+n)}\gamma$, we have $B^{(k+1)} \approx_{\gamma'} B^{(k)}UD$. In addition, the profile of $B^{(k+1)}$ matches the (D-scaled) profile of $B^{(k)}$ outside of index $[i : j]$ to absolute error γ and has bounded drop on index $[i : j]$: $drop(B^{(k+1)}_{[i:j]}) \leq (\boldsymbol{\alpha}(j-i)+\gamma)(j-i)$. The running time of this algorithm, excluding the call to* `LatRed`*, is*

$$O\left(n^\omega(drop(B^{(k)}) - \log\gamma + n)^{1+\varepsilon}\right).$$

Proof. By the properties of `CompressLattice`, we have $B'_{sub} \approx_{\gamma_1} B_{sub}U'_{sub}D_{sub}$ for $\gamma_1 = 2^{-O(\text{drop}(B_{sub})+n)}\gamma$. Lattice reduction of B'_{sub} generates U''_{sub} such that there exists B^*_{sub} that is $\boldsymbol{\alpha}(j-i)$-lattice-reduced and $B^*_{sub} \approx_{\gamma_2} B'_{sub}U''_{sub}D_{sub}$ for $\gamma_2 = 2^{-O(\text{drop}(B^*_{sub})+n)}\gamma$ and some D_{sub}. This means that the profile of $B'_{sub}U''_{sub}$ matches the profile of B^*_{sub} (up to scaling by D) with absolute error γ.

By the second lattice compression, the profile of $B^{(k+1)}$ matches the profile of $B^{(k)}U'$ (up to scaling) with absolute error γ, and by construction of U', this matches the profile of $B^{(k)}$ outside of $[i : j]$ and matches the profile of $B'_{sub}U''_{sub}$ inside. Thus, since B^*_{sub} is $\boldsymbol{\alpha}(j-i)$-lattice-reduced, $\text{drop}(B^*_{sub}) \leq \boldsymbol{\alpha}(j-i)(j-i)$. The absolute error of γ leads to at most $\gamma(j-i)$ error in the drop of $B^{(k+1)}_{[i:j]}$, concluding the bounds about the profile error.

Clearly the returned basis $B^{(k+1)}$ is compressed and D is diagonal as a result of the second call to `CompressLattice`. Because D_{sub} is a valid scaling, $U'_{sub}D_{sub}U''_{sub}D^{-1}_{sub}$ is unimodular, so $U'U''$ is unimodular.

The running time of `ReduceSublattice` involves calls to `CompressLattice`, calls to matrix multiplication, and potentially recursive calls to `LatRed`. We can bound the cost of `CompressLattice` by Lemma 2 and we can bound the condition number and entry size of all matrices involved by $2^{O(\text{drop}(B^{(k)})+n)}$, which bounds the cost of matrix multiplication. Excluding the call to `LatRed`, the running time is $O(n^\omega(\text{drop}(B^{(k)}) - \log\gamma + n)^{1+\varepsilon})$. $\square$

We wish to bound the number of rounds r required in the execution of Algorithm 4. To prove the finiteness of the number of rounds, we introduce $\bar{\mathit{\Pi}}^{(end)}$ as a formality, defining $\bar{\mathit{\Pi}}^{(end)} = \bar{\mathit{\Pi}}(B^{(3r)})$ if Algorithm 4 terminates in r rounds and $\bar{\mathit{\Pi}}^{(end)} = \lim_{i\to\infty} \bar{\mathit{\Pi}}(B^{(3i)})$ if it never terminates. Since $\bar{\mathit{\Pi}}(B^{(3i)})$ is monotonically decreasing and bounded below by Lemma 6, the limit exists.

Lemma 16 (The number of rounds is heuristically bounded). *Let $B^{(0)} \in \mathbb{Z}^{n\times n}$ and $\boldsymbol{\alpha}(\cdot)$ be the input to Algorithm 4. Let α be the global reduction parameter. Assuming Heuristics 1 and 2 are true, the number of rounds r is bounded by*

$$r = O\left(\log\left(\frac{\bar{\mathit{\Pi}}(B^{(0)}) - \bar{\mathit{\Pi}}^{(end)}}{n^3(\alpha - \alpha^*)}\right)\right).$$

Proof. Let $\mathit{\Delta}^{(i)} = \bar{\mathit{\Pi}}(B^{(3i)}) - \bar{\mathit{\Pi}}^{(end)}$. Heuristic 2 tells us that there exists c_1 such that for $i < r-2$,

$$\bar{\mathit{\Pi}}(B^{(3i)}) - \bar{\mathit{\Pi}}(B^{(3(i+1))}) \geq c_1 n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2)),$$

and by Heuristic 1, there exists $c_2 \in [0,1)$ such that

$$\bar{\mathit{\Pi}}(B^{(3i)}) - \bar{\mathit{\Pi}}(B^{(3(i+1))}) \geq c_2 \mathit{\Delta}^{(i)}.$$

We make the claim that if $\mathit{\Delta}^{(i)} \leq (1-c_2)^{-k} c_1 n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))$, then $r - i \leq k+2$, and we prove the claim by induction on $k \geq 0$.

Clearly for $k = 0$, then $\mathit{\Delta}^{(i)} \leq c_1 n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))$, so Heuristic 2 tells us that $r - i \leq 2 = k+2$.

If the claim is satisfied for $k-1$ and $\mathit{\Delta}^{(i)} \leq (1-c_2)^{-(k-1)} c_1 n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))$, then $\bar{\mathit{\Pi}}(B^{(3i)}) - \bar{\mathit{\Pi}}(B^{(3(i+1))}) \geq c_2 \mathit{\Delta}^{(i)}$ by Heuristic 1. This implies $\mathit{\Delta}^{(i+1)} = \mathit{\Delta}^{(i)} - (\bar{\mathit{\Pi}}(B^{(3i)}) - \bar{\mathit{\Pi}}(B^{(3(i+1))})) \leq (1-c_2)\mathit{\Delta}^{(0)} \leq (1-c_2)^{-(k-1)} c_1 n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))$. We use the inductive hypothesis to conclude $r - (i+1) \leq (k-1) + 2 \equiv r - i \leq k+2$.

We then have

$$\begin{aligned} r \leq k+2 &= \log\left(\frac{\mathit{\Delta}^{(0)}}{c_1 n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))}\right) / \log(1/(1-c_2)) \\ &= O\left(\log\left(\frac{\bar{\mathit{\Pi}}(B^{(0)}) - \bar{\mathit{\Pi}}^{(end)}}{n^3(\alpha - \alpha^*)}\right)\right). \end{aligned}$$

□

Next, we wish to relate the profile drop to the change in potential. This is expressed through Lemma 4, which we recall here.

Lemma 4. *Let $B^{(3i)}$ be the compressed LR-reduced basis of rank n at round i used as input to Algorithm 4, and let $B^{(3i+3)}$ be the basis after the three sublattice reductions. The drop of the input basis for the next round is bounded by the current round's change in global potential $\bar{\mathit{\Pi}}$:*

$$n^2\left(\mathit{drop}(B^{(3i+3)}) - 5\boldsymbol{\alpha}(n)n/2\right) = O\left(\bar{\mathit{\Pi}}(B^{(3i)}) - \bar{\mathit{\Pi}}(B^{(3i+3)})\right).$$

Proof. At a high level, we want to show that if $\operatorname{drop}(B^{(3i+3)})$ is large, then the change in potential from the first sublattice reduction must have been large.

We introduce the following shorthand to make the proof more concise. Since $n > 2$ is a power of 2, we split the profile into four sections: 1 (referring to the first $n/4$ entries) through 4 (the last $n/4$) entries. We also label the first $n/2$ entries and last $n/2$ entries by L and R respectively, and the middle $n/2$ entries by M. These blocks overlap; block 2 is contained in both block L and block M.

Let $\vec{\ell}$ be the subset of the global profile corresponding to $B^{(3i)}$, and let $\vec{\ell}'$, $\vec{\ell}''$, and $\vec{\ell}'''$ be the subset of the global profile for $B^{(3i+1)}$ through $B^{(3i+3)}$.

Our proof involves the drop, maxima and minima of the blocks of these profiles, so in our shorthand,

$$\begin{aligned}\operatorname{drop} &= \operatorname{drop}(\vec{\ell})\\ \operatorname{drop}''_L &= \operatorname{drop}((\ell''_1, \ldots, \ell''_{n/4}))\\ \max'_2 &= \max_{n/4+1 \le j \le n/2} \ell'_j\\ \min'''_R &= \min_{n/2+1 \le j \le n} \ell'''_j.\end{aligned}$$

Our lattice reduction definition introduces constraints on these values. We let $\alpha = \boldsymbol{\alpha}(n)$ be shorthand. Because blocks L and R have α-bounded drop, we have $\operatorname{drop}_L, \operatorname{drop}_R \le \alpha n/2$. The first sublattice reduction works on block M, so blocks 1 and 4 remain unchanged between $\vec{\ell}$ and $\vec{\ell}'$. Since block M in $\vec{\ell}'$ is reduced, we have $\operatorname{drop}'_M \le \alpha n/2$. Similar constraints are introduced by sublattice reducing blocks L and R, and we use these constraints to bound the change in potential.

At the end of all three sublattice reductions, we have $\operatorname{drop}'''_L, \operatorname{drop}'''_R \le \alpha n/2$. By properties of the profile drop,

$$\operatorname{drop}''' \le \operatorname{drop}'''_L + \operatorname{drop}'''_R + \max'''_L - \min'''_R \le \alpha n + \max'''_L - \min'''_R$$

so the overall profile drop forms a lower bound on $\max'''_L - \min'''_R$.

Since the minimum of a profile cannot decrease during lattice reduction, $\min'''_R \ge \min''_R = \min'_R$. Similarly, $\max'''_L = \max''_L \le \max'_L$. We therefore have

$$\operatorname{drop}''' \le \max'_L - \min'_R + \alpha n.$$

In other words, if the drop after three rounds is large, the drop after one round was also large.

If $\max'_L - \min'_R$ is large, then we have four cases to consider. $\max'_L = \max(\max'_1, \max'_2)$ and $\min'_R = \min(\min'_3, \min'_4)$, so the four cases refer to which part of profile l' value determines $\max'_L$ and which part determines $\min'_R$.

If $\max'_L = \max'_2$ and $\min'_R = \min'_3$, then $\max'_L - \min'_R$ large implies $\max'_2 - \min'_3$ large. However, this contradicts the condition that block M of profile l' is α-reduced, since $\max'_2 - \min'_3 \le \alpha n/2$. This case is therefore impossible for $\max'_L - \min'_R > \alpha n/2 \Leftarrow \operatorname{drop}''' > 3\alpha n/2$.

In the remaining three cases, we therefore have $\max'_L = \max'_1$ or $\min'_R = \min'_4$. We wish to show that either $\min_2 - \max'_2$ or $\min'_3 - \max_3$ is large. This means that either every profile element in block 2 dropped significantly during the first sublattice reduction or every profile element in block 3 increased significantly; in either case, this implies a significant change in potential.

If $\max'_L = \max'_1$ and $\min'_R = \min'_3$, then $\max'_L = \max_1$, and by the reducedness of the L block in l, we have $\min_2 \geq \max_1 - \alpha n/2$. By the reducedness of the M block in l', we have $\max'_2 \leq \min'_3 + \alpha n/2 = \min'_R + \alpha n/2$. Therefore in this case

$$\begin{aligned}\min_2 - \max'_2 &\geq \max_1 - \min'_3 - \alpha n\\ &= \max'_L - \min'_R - \alpha n\\ &\geq \mathrm{drop}''' - 2\alpha n\end{aligned}$$

We therefore have a nontrivial lower bound on $\min_2 - \max'_2$ when $\mathrm{drop}''' > 2\alpha n$.

If $\max'_L = \max'_2$ and $\min'_R = \min'_4$, then similar logic shows that $\min'_3 - \max_3 \geq \mathrm{drop}''' - 2\alpha n$.

If $\max'_L = \max'_1$ and $\min'_R = \min'_4$, then the reducedness of L and R in l implies $\min_2 \geq \max_1 - \alpha n/2 = \max'_L - \alpha n/2$ and $\max_3 \leq \min_4 + \alpha n/2 = \min'_R + \alpha n/2$. Therefore,

$$\begin{aligned}\min_2 - \max_3 &\geq \max'_L - \min'_R - \alpha n\\ &\geq \mathrm{drop}''' - 2\alpha n.\end{aligned}$$

We also have

$$\begin{aligned}\min_2 - \max_3 &= \min_2 - \max'_2 + \max'_2 - \min'_3 + \min'_3 - \max_3\\ &\leq (\min_2 - \max'_2) + (\min'_3 - \max_3) + \alpha n/2,\end{aligned}$$

so combining these gives

$$(\min_2 - \max'_2) + (\min'_3 - \max_3) \geq \mathrm{drop}''' - 5\alpha n/2.$$

One of $\min_2 - \max'_2$ or $\min'_3 - \max_3$ must therefore be at least $(\mathrm{drop}''' - 5\alpha n/2)/2$.

No matter which case we are in, we have shown that either

$$\min_2 - \max'_2 \geq d \text{ or } \min'_3 - \max_3 \geq d \text{ for } d = \mathrm{drop}''' - 5\alpha n/2.$$

With these bounds in mind, we can now bound the change in potential during the first sublattice reduction of the M block. We consider the case where $\min_2 - \max'_2 \geq d$, since the analysis for the other case is similar. Our analysis is based on the partial sums s_k. To recall, we have

$$\begin{aligned}s_k &= \sum_{i=1}^{k} \ell_i & s'_k &= \sum_{i=1}^{k} \ell'_i\\ \bar{\Pi}(\vec{\ell}) &= \sum_{k=1}^{n} s_k & \bar{\Pi}(\vec{\ell}') &= \sum_{k=1}^{n} s'_k\end{aligned}$$

and

$$s'_k \leq s_k \text{ for all } 1 \leq k \leq n.$$

To lower bound the change in potential, we have

$$\begin{aligned}
\bar{\Pi}(\vec{\ell}) - \bar{\Pi}(\vec{\ell}') &= \sum_{k=1}^{n} s_k - \sum_{k=1}^{n} s'_k \\
&= \sum_{k=1}^{n} (s_k - s'_k) \\
&\geq \sum_{k=n/4+1}^{n/2} (s_k - s'_k) \\
&= \sum_{k=n/4+1}^{n/2} \sum_{i=1}^{k} (\ell_k - \ell'_k) \\
&= \sum_{k=n/4+1}^{n/2} \left(\sum_{i=1}^{n/4} (\ell_k - \ell'_k) + \sum_{i=n/4+1}^{k} (\ell_k - \ell'_k) \right) \\
&\geq \sum_{k=n/4+1}^{n/2} \left(0 + \sum_{i=n/4+1}^{k} d \right) = d \left(\frac{n(n+4)}{32} \right).
\end{aligned}$$

Therefore we have $(n^2 + 4n)d \leq \bar{\Pi}(\vec{\ell}) - \bar{\Pi}(\vec{\ell}')$, so $n^2 d = O(\bar{\Pi}(\vec{\ell}) - \bar{\Pi}(\vec{\ell}'))$. when $\min_2 - \max'_2 \geq d$. In the case where $\min'_3 - \max_3 \geq d$, the only trick involves using the identical form $s_k = \sum_{i=1}^{n} \ell_i - \sum_{i=k+1}^{n} \ell_i$ and the observation $\sum_{i=1}^{n} \ell_i = \sum_{i=1}^{n} \ell'_i$. This analysis also gives $n^2 d = O(\bar{\Pi}(\vec{\ell}) - \bar{\Pi}(\vec{\ell}'))$.

Combining this part with the previous part, we have that if $\text{drop}(B^{(3i+3)}) = \text{drop}''' > 5\alpha n/2$, then

$$n^2(\text{drop}(B^{(3i+3)}) - 5\alpha n/2) = O(\bar{\Pi}(\vec{\ell}) - \bar{\Pi}(\vec{\ell}')).$$

□

Analyzing the running time of Algorithm 4 is challenging, mainly due to how the evolution of the profile influences both the necessary precision and recursive behavior of the algorithm. Our proposed cost bound $T(\cdot)$ is complex, but all terms are necessary in our analysis. Before we prove Lemma 5, we first present an intermediate result which relies on Lemma 4 to simplify our argument.

Lemma 17. *Let $B^{(3i)}$ be a lattice basis of rank n input to Algorithm 4, $\boldsymbol{\alpha}$ the reduction quality parameter, and let $B^{(3r)}$ be as defined in the algorithm. Let $\omega \in (2, 3]$ and ε be parameters bounding the complexity of matrix multiplication, size reduction, and QR factorization as described in Lemma 2. We let $p = O(drop(B^{(3i)}) + n)$ represent the working precision which bounds the size of integers that appear in the computation.*

We define the following function $T(B^{(3i)}) =$

$$C_{\bar{\Pi}} n^{\omega-2}\left(\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3r)})\right)p^{\varepsilon} + C_{prec} n^{\omega} p^{1+\varepsilon} + C_A A(B^{(3i)}) n^{\omega}(n+\boldsymbol{\alpha}(n)n)p^{\varepsilon},$$

which has a term involving the potential change, a term involving the precision, and an term $A(\cdot)$ *involving the approximation factor.*

If $A(\cdot)$ *satisfies*

$$A(B^{(3i)}) \geq \frac{1}{2^{\omega+1}}\left(A(B_M) + A(B_L) + A(B_R)\right) + A(B^{(3i+3)}) + c$$

for all $i < r$ *and for some constant* $c > 0$*, then there exist positive constants* $C_{\bar{\Pi}}$*,* C_p*, and* C_A *such that* $T(\cdot)$ *satisfies Equation (1).*

Proof. For the time being, we will introduce the auxiliary $\tilde{A}(B) = C_A A(B) n^{\omega}(n+\boldsymbol{\alpha}(n)n)p^{\varepsilon}$ to make the inequalities more concise. After substituting into Equation (1), we wish to prove the following behemoth:

$$\begin{aligned}
&C_{\bar{\Pi}} n^{\omega-2}\left(\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3r)})\right)p^{\varepsilon} + C_{prec} n^{\omega} p^{1+\varepsilon} + \tilde{A}(B^{(3i)})\\
\geq& C_{\bar{\Pi}}\left(\frac{n}{2}\right)^{\omega-2}\left(\bar{\Pi}(B_{sub,M}) - \bar{\Pi}(B_{sub,M}^{(r_M)})\right){p_M}^{\varepsilon} + C_{prec}\left(\frac{n}{2}\right)^{\omega} p_M^{1+\varepsilon} + \tilde{A}(B_{sub,M})\\
+& C_{\bar{\Pi}}\left(\frac{n}{2}\right)^{\omega-2}\left(\bar{\Pi}(B_{sub,L}) - \bar{\Pi}(B_{sub,L}^{(r_L)})\right){p'_L}^{\varepsilon} + C_{prec}\left(\frac{n}{2}\right)^{\omega} {p'}_L^{1+\varepsilon} + \tilde{A}(B_{sub,L})\\
+& C_{\bar{\Pi}}\left(\frac{n}{2}\right)^{\omega-2}\left(\bar{\Pi}(B_{sub,R}) - \bar{\Pi}(B_{sub,R}^{(r_R)})\right){p''_R}^{\varepsilon} + C_{prec}\left(\frac{n}{2}\right)^{\omega} {p''}_R^{1+\varepsilon} + \tilde{A}(B_{sub,R})\\
+& C_u n^{\omega} p^{1+\varepsilon}\\
+& C_{\bar{\Pi}} n^{\omega-2}\left(\bar{\Pi}(B^{(3i+3)}) - \bar{\Pi}(B^{(3r)})\right){p'''}^{\varepsilon} + C_{prec} n^{\omega} {p'''}^{1+\varepsilon} + \tilde{A}(B^{(3i+3)})
\end{aligned}$$

We work term by term to simplify this inequality. First, we examine the terms involving $C_{\bar{\Pi}}$. Because the change in potential in the sublattices is the same as the change in potential in the original lattice, we have

$$\begin{aligned}
\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3i+3)}) &= \bar{\Pi}(B_{sub,M}) - \bar{\Pi}(B_{sub,M}^{(r_M)})\\
&+ \bar{\Pi}(B_{sub,L}) - \bar{\Pi}(B_{sub,L}^{(r_L)})\\
&+ \bar{\Pi}(B_{sub,R}) - \bar{\Pi}(B_{sub,R}^{(r_R)}).
\end{aligned}$$

We have the condition that the drop always decreases, so the working precision always decreases, and this means $p_M, {p'}_L, {p''}_R, p''' \leq p$. Using this observation, we can substitute the above into the massive inequality and show that it suffices to prove the simplified inequality

$$\begin{aligned}
&C_{\bar{\Pi}}\left(1 - \frac{1}{2^{\omega-2}}\right) n^{\omega-2}\left(\bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3i+3)})\right)p^{\varepsilon} + C_{prec} n^{\omega} p^{1+\varepsilon} + \tilde{A}(B^{(3i)})\\
\geq& 3C_{prec}\left(\frac{n}{2}\right)^{\omega} p^{1+\varepsilon} + \tilde{A}(B_{sub,M}) + \tilde{A}(B_{sub,L}) + \tilde{A}(B_{sub,R})\\
+& C_u n^{\omega} p^{1+\varepsilon}\\
+& C_{prec} n^{\omega} p''' p^{\varepsilon} + \tilde{A}(B^{(3i+3)})
\end{aligned}$$

which we rewrite as

$$
\begin{aligned}
&C_{\bar{\Pi}}\left(1-\frac{1}{2^{\omega-2}}\right) n^{\omega-2}\left(\bar{\Pi}(B^{(3i)})-\bar{\Pi}(B^{(3i+3)})\right) p^{\varepsilon}\\
&+\left(C_{prec}\left(1-\frac{3}{2^{\omega}}\right)-C_u\right) n^{\omega} p^{1+\varepsilon}\\
&+\tilde{A}(B^{(3i)})\\
&\geq C_{prec} n^{\omega} p''' p^{\varepsilon}\\
&+\tilde{A}(B_{sub,M})+\tilde{A}(B_{sub,L})+\tilde{A}(B_{sub,R})+\tilde{A}(B^{(3i+3)}).
\end{aligned}
$$

By Lemma 4, there exists constant C_1 such that $n^2(\text{drop}(B^{(3i+3)})-5\boldsymbol{\alpha}(n)n/2) \leq C_1(\bar{\Pi}(B^{(3i)})-\bar{\Pi}(B^{(3i+3)}))$, so we can simplify further to

$$
\begin{aligned}
&\frac{C_{\bar{\Pi}}}{C_1}\left(1-\frac{1}{2^{\omega-2}}\right) n^{\omega}\left(\text{drop}(B^{(3i+3)})-5\boldsymbol{\alpha}(n)n/2\right) p^{\varepsilon}\\
&+\left(\left(1-\frac{3}{2^{\omega}}\right) C_{prec}-C_u\right) n^{\omega} p^{1+\varepsilon}\\
&+\tilde{A}(B^{(3i)})\\
&\geq C_{prec} n^{\omega} p''' p^{\varepsilon}\\
&+\tilde{A}(B_{sub,M})+\tilde{A}(B_{sub,L})+\tilde{A}(B_{sub,R})+\tilde{A}(B^{(3i+3)}).
\end{aligned}
$$

Since C_u is the non-recursive update cost, we can set C_{prec} large enough such that $\left(1-\frac{3}{2^{\omega}}\right) C_{prec}-C_u \geq 0$. This allows us to eliminate the second term.

Because $B^{(3i+3)}$ is compressed, $p'''=O(\text{drop}(B^{(3i+3)})+n)$, so there exists $C_2 \geq 5\alpha/2$ such that $p''' \leq C_2(\text{drop}(B^{(3i+3)})+n)$. Therefore, in order to prove the original claim, it suffices to prove

$$
\begin{aligned}
&\frac{C_{\bar{\Pi}}}{C_1}\left(1-\frac{1}{2^{\omega-2}}\right) n^{\omega}\left(\text{drop}(B^{(3i+3)})-5\alpha n/2\right) p^{\varepsilon}+\tilde{A}(B^{(3i)})\\
&\geq C_{prec} C_2 n^{\omega}(\text{drop}(B^{(3i+3)})+n) p^{\varepsilon}+\tilde{A}(B_{sub,M})+\\
&\quad \tilde{A}(B_{sub,L})+\tilde{A}(B_{sub,R})+\tilde{A}(B^{(3i+3)}).
\end{aligned}
$$

Rearranging terms,

$$
\begin{aligned}
&\left(\frac{C_{\bar{\Pi}}}{C_1}\left(1-\frac{1}{2^{\omega-2}}\right)-C_{prec}C_2\right) \text{drop}(B^{(3i+3)}) n^{\omega} p^{\varepsilon}+\tilde{A}(B^{(3i)})\\
&\geq\left(C_{prec}C_2 n+\frac{C_{\bar{\Pi}}}{C_1}\left(1-\frac{1}{2^{\omega-2}}\right)(5\boldsymbol{\alpha}(n)n/2)\right) n^{\omega} p^{\varepsilon}\\
&+\tilde{A}(B_{sub,M})+\tilde{A}(B_{sub,L})+\tilde{A}(B_{sub,R})+\tilde{A}(B^{(3i+3)}).
\end{aligned}
$$

Since C_1, C_{prec}, and C_2 are all constants, and since $\omega > 2$, we may set $C_{\bar{\Pi}}$ such that $\left(\frac{C_{\bar{\Pi}}}{C_1}\left(1-\frac{1}{2^{\omega-2}}\right)-C_{prec}C_2\right) \geq 0$.

Substituting back $\tilde{A}(B) = C_A A(B) n^\omega (n + \boldsymbol{\alpha}(n)n) p^\varepsilon$ gives

$$\begin{aligned}
C_A A(B^{(3i)}) n^\omega (n + \boldsymbol{\alpha}(n)n) p^\varepsilon \geq & \left(C_{prec} C_2 n + \frac{C_{\bar{\Pi}}}{C_1} \left(1 - \frac{1}{2^{\omega-2}}\right) \left(\frac{5\boldsymbol{\alpha}(n)n}{2}\right)\right) n^\omega p^\varepsilon \\
& + C_A A(B_{sub,M}) \left(\frac{n}{2}\right)^\omega \left(\frac{n}{2} + \boldsymbol{\alpha}\left(\frac{n}{2}\right)\frac{n}{2}\right) p_M{}^\varepsilon \\
& + C_A A(B_{sub,L}) \left(\frac{n}{2}\right)^\omega \left(\frac{n}{2} + \boldsymbol{\alpha}\left(\frac{n}{2}\right)\frac{n}{2}\right) p'_L{}^\varepsilon \\
& + C_A A(B_{sub,R}) \left(\frac{n}{2}\right)^\omega \left(\frac{n}{2} + \boldsymbol{\alpha}\left(\frac{n}{2}\right)\frac{n}{2}\right) p''_R{}^\varepsilon \\
& + C_A A(B^{(3i+3)}) n^\omega (n + \boldsymbol{\alpha}(n)n) p'''^\varepsilon .
\end{aligned}$$

We use the bound on the precision and $(n + \boldsymbol{\alpha}(n)n) \geq 2\left(\frac{n}{2} + \boldsymbol{\alpha}\left(\frac{n}{2}\right)\frac{n}{2}\right)$ to show it suffices to prove

$$\begin{aligned}
A(B^{(3i)}) \geq & \frac{1}{2^{\omega+1}} \left(A(B_{sub,M}) + A(B_{sub,L}) + A(B_{sub,R})\right) + A(B^{(3i+3)}) \\
& + \frac{1}{C_A} \left(C_{prec} C_2 n + \frac{C_{\bar{\Pi}}}{C_1} \left(1 - \frac{1}{2^{\omega-2}}\right) (5\boldsymbol{\alpha}(n)n/2)\right) (n + \boldsymbol{\alpha}(n)n)^{-1}
\end{aligned}$$

Since $\alpha = \Omega(1)$ and since by assumption about A we have $A(B^{(3i)}) \geq \frac{1}{2^{\omega+1}} \left(A(B_{sub,M}) + A(B_{sub,L}) + A(B_{sub,R})\right) + A(B^{(3i+3)}) + c$, we may set C_A large enough that the inequality is satisfied. This proves the original claim. □

We have transformed the problem of proving something about all terms of cost function $T(\cdot)$ to a question about only the approximation term. We propose a candidate term that satisfies the necessary condition under our heuristic assumptions.

Lemma 18. *Let $\{B^{(0)}, \ldots, B^{(3r)}\}$ be bases created during the execution of Algorithm 4. Assume Heuristic 2 holds. Next, define*

$$A(B^{(3i)}) = \frac{\Pi(B^{(3i)}) - \Pi(B^{(3r)})}{n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))} + c_1$$

for $i < r$ and $A(B^{(3r)}) = 0$ otherwise. There exists positive constants c_1, c_2 such that

$$A(B^{(3i)}) \geq 2^{-(\omega+1)}(A(B_M^{(3i)}) + A(B_L^{(3i+1)}) + A(B_R^{(3i+2)})) + A(B^{(3(i+1))}) + c_2$$

for all $i < r$. This choice of A satisfies the conditions of Lemma 17.

Proof. At many points, we will use the observation that the change in potential globally equals the total change in potential in each sublattice. We will use $\Delta^{(i)} = \bar{\Pi}(B^{(3i)}) - \bar{\Pi}(B^{(3r)})$ to denote the total change in potential from the current input, and we use $\Delta_M^{(i)}, \Delta_L^{(i)}, \Delta_R^{(i)}$ to denote the total change in potential from the input to the recursive sublattice calls. In this notation, $\Delta^{(i)} = \Delta_M^{(i)} + \Delta_L^{(i)} + \Delta_R^{(i)} + \Delta^{(i+1)}$. We also denote $\xi_n = n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))$.

We begin by considering the case $i < r-2$. Then our inequality is equivalent to

$$\frac{\Delta^{(i)}}{\xi_n} + c_1 \geq 2^{-(\omega+1)} \left(\frac{\Delta_M^{(i)}}{\xi_{n/2}} + c_1 + \frac{\Delta_L^{(i)}}{\xi_{n/2}} + c_1 + \frac{\Delta_R^{(i)}}{\xi_{n/2}} + c_1 \right) + \frac{\Delta^{(i+1)}}{\xi_n} + c_1 + c_2$$

which is equivalent to

$$\frac{\Delta^{(i)} - \Delta^{(i+1)}}{\xi_n} \geq 2^{-(\omega+1)} \left(\frac{\Delta_M^{(i)} + \Delta_L^{(i)} + \Delta_R^{(i)}}{\xi_{n/2}} + 3c_1 \right) + c_2.$$

This inequality is satisfied when

$$\left(1 - \frac{\xi_n}{\xi_{n/2} 2^{\omega+1}}\right) \frac{\Delta^{(i)} - \Delta^{(i+1)}}{\xi_n} \geq \frac{3c_1}{2^{\omega+1}} + c_2.$$

By Heuristic 2, $\Delta^{(i)} - \Delta^{(i+1)} = \Omega(n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))) = \Omega(\xi_n)$. That means it is possible to select positive c_1, c_2 satisfying this equality so long as $\frac{\xi_n}{\xi_{n/2}} < 2^{\omega+1}$. We have

$$\frac{\xi_n}{\xi_{n/2}} = \frac{n^3(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))}{(n/2)^3(\boldsymbol{\alpha}(n/2) - \boldsymbol{\alpha}(n/4))} = \frac{8\left(\left(\frac{n}{N}\right)^{\log g}(\alpha - \alpha^*) - \left(\frac{n/2}{N}\right)^{\log g}(\alpha - \alpha^*)\right)}{\left(\frac{n/2}{N}\right)^{\log g}(\alpha - \alpha^*) - \left(\frac{n/4}{N}\right)^{\log g}(\alpha - \alpha^*)}$$

$$= \frac{8(1 - (\frac{1}{2})^{\log g})}{(\frac{1}{2})^{\log g} - (\frac{1}{4})^{\log g}} = 8\frac{g-1}{g}\frac{g^2}{g-1} = 8g < 2^{\omega+1}.$$

Next, we consider the case $i = r-1$. Our inequality is equivalent to

$$\frac{\Delta^{(r-1)}}{\xi_n} + c_1 \geq 2^{-(\omega+1)} \left(\frac{\Delta_M^{(r-1)}}{\xi_{n/2}} + c_1 + \frac{\Delta_L^{(r-1)}}{\xi_{n/2}} + c_1 + \frac{\Delta_R^{(r-1)}}{\xi_{n/2}} + c_1 \right) + 0 + c_2$$

which we rewrite as

$$\left(1 - \frac{\xi_n}{\xi_{n/2} 2^{\omega+1}}\right) \frac{\Delta^{(r-1)}}{\xi_n} \geq \left(\frac{3}{2^{\omega+1}} - 1\right) c_1 + c_2.$$

Since the left hand side is nonnegative, it suffices to show that the right hand side is negative. This imposes the condition $c_2 \leq \left(\frac{2^{\omega+1}-3}{2^{\omega+1}}\right) c_1$, which we will show later can be satisfied.

Finally, we consider the case $i = r-2$. We use the bound for $A(B^{(3(r-1))})$ just developed to show that it suffices to prove

$$\left(1 - \frac{\xi_n}{\xi_{n/2}2^{\omega+1}}\right)\frac{\Delta^{(r-2)}}{\xi_n} \geq \left(\frac{6}{2^{\omega+1}} - 1\right)c_1 + 2c_2.$$

By the same logic as before, this imposes the requirement $c_2 \leq \left(\frac{2^{\omega+1}-6}{2^{\omega+2}}\right)c_1$.

We have shown that the inequality is satisfied in all cases if

$$\begin{aligned} c_1\left(\frac{3}{2^{\omega+1}} + \frac{c_2}{c_1}\right) &= O(1) \\ \frac{c_2}{c_1} &\leq \left(\frac{2^{\omega+1}-3}{2^{\omega+1}}\right) \\ \frac{c_2}{c_1} &\leq \left(\frac{2^{\omega+1}-6}{2^{\omega+2}}\right). \end{aligned}$$

Because $\omega > 2$, we can select $\frac{c_2}{c_1} > 0$ first, then select $c_1 > 0$. Therefore there exists $c_1, c_2 > 0$ such that the claim is true. □

Using our heuristic assumptions, we have constructed a candidate function $T(\cdot)$ such that the running time of Algorithm 4 on input $B^{(3i)}$ and $\boldsymbol{\alpha}$ is bounded by $T(B^{(3i)})$. We now recall Lemma 5, which simplifies this asymptotic running time.

Lemma 5. *Let $\omega \in (2,3]$ be a parameter bounding the complexity of matrix multiplication, size reduction, and QR factorization as described in Lemma 2. If the heuristic assumptions 1 and 2 are correct, then there exists appropriate choice of constants $C_{\bar{\Pi}}, C_{prec}, C_A > 0$ such that $T(\cdot)$ satisfies Equation 1.*

As a corollary, this means that for input B and reduction goal α, it is possible to instantiate $\boldsymbol{\alpha}$ and γ such that the running time of Algorithm 4 on these inputs is $O(T(B)) =$

$$O\left(\left(\frac{\alpha}{\alpha - \alpha^*}\right)n^\omega p^{1+\varepsilon} + \alpha n^{\omega+1}p^\varepsilon\right).$$

where $p = O(drop(B) + n)$.

Proof. We let $C_{\bar{\Pi}}, C_{prec}$, C_A, and $T(\cdot)$ be as in Lemmas 17 and 18. We let $\Delta = \bar{\Pi}(B^{(0)}) - \bar{\Pi}(B^{(3r)})$. By Lemmas 17 and 18, our cost function $T(\cdot)$ bounds the execution time at each level; we have $T(B^{(0)})$

$$\begin{aligned} &= O\left(n^{\omega-2}\Delta p^\varepsilon + n^\omega p^{1+\varepsilon} + A(B)n^\omega(n + \boldsymbol{\alpha}(n)n)p^\varepsilon\right) \\ &= O\left(n^{\omega-2}p^\varepsilon\left(\Delta + n^2 p + \frac{\Delta}{n(\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2))}(n + \boldsymbol{\alpha}(n)n) + (1+\boldsymbol{\alpha}(n))n^3\right)\right) \\ &= O\left(n^{\omega-2}p^\varepsilon\left(\Delta\left(1 + \frac{1+\boldsymbol{\alpha}(n)}{\boldsymbol{\alpha}(n) - \boldsymbol{\alpha}(n/2)}\right) + n^2 p + (1+\boldsymbol{\alpha}(n))n^3\right)\right). \end{aligned}$$

By the boundedness of potential for integer bases [40], $\varDelta = O(n^2 p)$, so the runtime is

$$O\left(\left(1+\frac{1+\boldsymbol{\alpha}(n)}{\boldsymbol{\alpha}(n)-\boldsymbol{\alpha}(n/2)}\right)n^{\omega}p^{1+\varepsilon}+(1+\boldsymbol{\alpha}(n))n^{\omega+1}p^{\varepsilon}\right).$$

Finally, using $\boldsymbol{\alpha}(n) = \varOmega(\alpha^*)$, $\alpha^* = \varTheta(1)$, and the definition of $\boldsymbol{\alpha}$,

$$O\left(\left(\frac{\alpha}{\alpha-\alpha^*}\right)n^{\omega}p^{1+\varepsilon}+\alpha n^{\omega+1}p^{\varepsilon}\right).$$

□

B.7 The General Reduction Algorithm

We present two new wrappers around Algorithm 4 that allow us to reduce more general lattice bases. First, we will consider how to reduce compressed bases whose dimension is a power of 2 by making it LR-reduced. This idea is presented in Algorithm 8.

Algorithm 8: `ReducePow2`

```
  Input  : Compressed lattice basis $B$ of rank $n = 2^k$, reduction parameter $\alpha$
  Output: $U \in \mathbb{Z}^{n\times n}$ such that $\mathrm{drop}(BU) \le \alpha n$
1 if $n = 1$ then
2     return $U \leftarrow I_1$
3 $\gamma \leftarrow 2^{-\varTheta(\mathrm{drop}(B)+n)}$
4 $B', U_L, D_L \leftarrow$ ReduceSublattice($B, 0, \frac{n}{2}, \boldsymbol{\alpha}, \gamma$, Reduce)
5 $B'', U_R, D_R \leftarrow$ ReduceSublattice($B', \frac{n}{2}, n, \boldsymbol{\alpha}, \gamma$, Reduce)
6 $U_1 \leftarrow U_L D_L U_R D_L^{-1}$
7 $B_{LR}, U_c, D_c \leftarrow$ CompressLattice($BU_1, \gamma$)
8 $U_2 \leftarrow$ ReduceLR($B_{LR}, \boldsymbol{\alpha}, \gamma$)
9 $U \leftarrow U_1 U_c D_c U_2 D_c^{-1}$
10 return $U$
```

In this algorithm, we begin with a compressed basis that is not LR-reduced. We use `ReducePow2` recursively to first LR-reduce and $\boldsymbol{\alpha}$-reduce the left sublattice, then the same for the right. This new basis is LR-reduced and compressed, so we reduce it the rest of the way with `ReduceLR`. Overall, this method calls `ReduceLR` $n/2$ times on bases of rank 2, $n/4$ times on bases of rank 4, and so on, and it also performs the $O(n^{\omega}p^{1+\varepsilon})$ update steps associated with compression at each level of the recursion tree. This allows us to use Lemma 3 and Lemma 5 to bound the runtime of Algorithm 8.

Lemma 19. *Let B be a basis of rank n input to Algorithm 8 and assume the same conditions as Lemma 5. The running time of the algorithm on this input is*

$$O\left(\frac{\alpha}{\alpha-\alpha^*}n^{\omega}\left(\mathit{drop}(B)+n\right)^{1+\varepsilon}\right)$$

Proof. Let $p = O(\mathrm{drop}(B) + n)$ be the initial working precision. Note that for any sublattice B_{sub}, $n_{sub} < n$ and $\mathrm{drop}(B_{sub}) \leq \mathrm{drop}(B)$, so p bounds the working precision in any called process. At each recursion level, there is a call to `ReduceLR` and the $O(n_{sub}^{\omega} p^{1+\varepsilon})$ update steps for compressing the basis and multiplying matrices. Summing over all nodes in the recursion tree, the running time of the algorithm is therefore

$$\begin{aligned}
\sum_{i=1}^{\log n} \frac{n}{2^i} O\left(\left(\frac{\boldsymbol{\alpha}(2^i)}{(\boldsymbol{\alpha}(2^i) - \alpha^*)}\right)(2^i)^{\omega} p^{1+\varepsilon} + \boldsymbol{\alpha}(2^i)(2^i)^{\omega+1} p^{\varepsilon} + (2^i)^{\omega} p^{1+\varepsilon}\right) \\
= O\left(p^{1+\varepsilon} \sum_{i=1}^{\log n}\left(\frac{\alpha}{\boldsymbol{\alpha}(2^i) - \alpha^*}\right) 2^{\log n - i + i\omega} + \alpha p^{\varepsilon} \sum_{i=1}^{\log n} 2^{\log n + \omega i}\right) \\
= O\left(\alpha n^{\omega+1} p^{\varepsilon} + p^{1+\varepsilon} \sum_{i=1}^{\log n}\left(\frac{\alpha}{\boldsymbol{\alpha}(2^i) - \alpha^*}\right) 2^{\log n - i + i\omega}\right).
\end{aligned}$$

The second term can be bounded as

$$\begin{aligned}
O\left(\frac{\alpha}{\alpha - \alpha^*} p^{1+\varepsilon} \sum_{i=1}^{\log n} 2^{\log g(\log n - i) + \log n - i + i\omega}\right) \\
= O\left(\frac{\alpha}{\alpha - \alpha^*} p^{1+\varepsilon} 2^{\log g \log n + \log n} (2^{\omega - \log g - 1})^{\log n}\right) \\
= O\left(\frac{\alpha}{\alpha - \alpha^*} n^{\omega} p^{1+\varepsilon}\right).
\end{aligned}$$

This makes the running time of the function

$$\begin{aligned}
O\left(\left(\frac{\alpha}{\alpha - \alpha^*}\right) n^{\omega} p^{1+\varepsilon} + \alpha n^{\omega+1} p^{\varepsilon}\right) \\
= O\left(\left(\frac{\alpha}{\alpha - \alpha^*}\right) n^{\omega} (p + \alpha n)^{1+\varepsilon}\right) \\
= O\left(\left(\frac{\alpha}{\alpha - \alpha^*}\right) n^{\omega} \left(\mathrm{drop}(B) + n\right)^{1+\varepsilon}\right)
\end{aligned}$$

using the fact that for B not yet reduced, $\mathrm{drop}(B) = \Omega(\alpha n)$. □

Note that as α approaches α^*, the algorithm becomes more expensive, but as α grows, the bound roughly stays the same.

The last piece in the puzzle is a method to convert an arbitrary basis to one that is compressed with rank a power of 2. We present one such method in Algorithm 9, and we use this construction to prove Theorem 1.

To prove the correctness of the algorithm, we need to do one more trick to show the output is size-reduced. The tools to do this are in the following lemma.

Lemma 20. *Let $B \approx_{\gamma} B'$ be two similar bases of rank n, and assume B' is size-reduced. If $-\log \gamma = \Omega(\mathit{spread}(B) + n)$, then B is also size-reduced.*

Algorithm 9: `Reduce`

```
Input  : Lattice basis $B$ of rank $n$, bound $C > \log(\kappa(B))$, reduction
         parameter $\alpha$
Output: $U \in \mathbb{Z}^{n\times n}$ such that $BU$ is $O(\alpha)$-lattice-reduced
1  $N \leftarrow 2^{\lceil \log(n) \rceil}$
2  $P \leftarrow \lceil 2n\|B\|_{max} \rceil I_{N-n}$
3  $B_{pad} \leftarrow diag(B, P)$
4  $\gamma \leftarrow 2^{-\Theta(C+N)}$
5  $\hat{B}_{pad}, U_1, D_1 \leftarrow$ CompressLattice$(B_{pad}, \gamma)$
6  $U_2 \leftarrow$ ReducePow2$(\hat{B}_{pad}, \alpha)$
7  $U'_{pad} \leftarrow U_1 D_1 U_2 D_1^{-1}$
8  $\gamma_2 \leftarrow 2^{-\Theta(\text{spread}(B)+N)}$
9  $B_2, s \leftarrow$ ScaleAndRound$(B_{pad}, \gamma_2 2^{-O(C+N)})$
10 $B_3 \leftarrow B_2 U'_{pad}$
11 $B_4 \leftarrow$ QR$(B_3, \gamma_2)$
12 $B_5, s' \leftarrow$ ScaleAndRound$(B_4, \gamma_2)$
13 $B_6, U_{sr} \leftarrow$ SizeReduce$(B_5)$
14 $U_{pad} \leftarrow U'_{pad} U_{sr}$
15 $U \leftarrow$ top left $n \times n$ block of $U_{pad}$
16 return $U$
```

Proof. We let $B = QDM$ and $B' = Q'D'M'$ decompose our two bases. By definition of similarity, we have $DM \approx_\gamma D'M'$. Ideally, we would like to show $M \approx_{\gamma_2} M'$ for some γ_2 so we can bound $\kappa(M)$ by $\kappa(M')$. Instead, we show $M \approx_{\gamma_2} D'D^{-1}M'$ and $\kappa(D'D^{-1})$ is small. For any nonzero $\vec{x}$, we have

$$\begin{aligned}
\|DM\vec{x} - D'M'\vec{x}\| &\leq \gamma\|DM\vec{x}\| \\
&\leq \gamma\|D\|\|M\vec{x}\| \\
&= \gamma(\max_i D_{ii})\|M\vec{x}\| \\
\|DM\vec{x} - D'M'\vec{x}\| &\geq \|M\vec{x} - D'D^{-1}M'\|/\|D^{-1}\| \\
&= \|M\vec{x} - D'D^{-1}M'\|(\min_i D_{ii}),
\end{aligned}$$

thus $\|M\vec{x} - D'D^{-1}M'\| \leq (\gamma(\max_i D_{ii})(\min_i D_{ii})^{-1})\|M\vec{x}\|$, so we have $M \approx_{\gamma_2} D'D^{-1}M$ for $\gamma_2 = \gamma(\max_i D_{ii})(\min_i D_{ii})^{-1}) = \gamma 2^{\text{spread}(B)}$. Item 3 of Lemma 1 tells us that $\kappa(M) \leq (1 + \kappa(D'D^{-1}M')\gamma_2)\kappa(D'D^{-1}M')$, so our goal is to bound $\kappa(D'D^{-1}M') \leq \kappa(D'D^{-1})\kappa(M')$.

$D'D^{-1}$ is a diagonal matrix whose entries are $D'_{1,1}/D_{1,1}, \ldots D'_{n,n}/D_{n,n}$. This implies that $\kappa(D'D^{-1}) = \max_i(D'_{ii}/D_{ii})/(\min_i(D'_{ii}/D_{ii}))$. Since D is the diagonal of the R-factor of B, we use the result in the proof of Item 4 of Lemma 1 to bound

$$\begin{aligned} D'_{ii}/D_{ii} &= 1 + (R'_{ii} - R_{ii})/(R_{ii}) \\ &\leq 1 + |R'_{ii} - R_{ii}|/|R_{ii}| \\ &\leq 1 + \gamma\sqrt{2n^3}\kappa(B). \end{aligned}$$

We bound D_{ii}/D'_{ii} in the same way to get $\kappa(D'D^{-1}) \leq (1 + 2\gamma\sqrt{2n^3}\kappa(B))$ to first order in γ. Since $\kappa(M') = \|M\|\|M^{-1}\| = 2^{O(n)}$ by size-reducedness, $\kappa(D'D^{-1}M) = (1 + \gamma\sqrt{2n^3}\kappa(B))2^{O(n)}$. Therefore, we have

$$\kappa(M) = (1 + (1 + \gamma\sqrt{2n^3}\kappa(B))2^{O(n)}\gamma_2)(1 + \gamma\sqrt{2n^3}\kappa(B))2^{O(n)}$$

which implies

$$\kappa(M) = (1 + (1 + \gamma 2^{O(\mathrm{spread}(B)+n)})\gamma 2^{O(\mathrm{spread}(B)+n)})(1 + \gamma 2^{O(\mathrm{spread}(B)+n)})2^{O(n)}.$$

Because $-\log\gamma = \Omega(\mathrm{spread}(B) + n)$, then $\kappa(M) = 2^{O(n)}$. Since M is unitriangular, $\|M\| \geq 1$ and $\|M^{-1}\| \geq 1$, so $\|M\| = 2^{O(n)}$ and $\|M^{-1}\| = 2^{O(n)}$, proving that B is size-reduced. □

Finally, we can prove the correctness and running time of Algorithm 9.

Theorem 1 (Full). *Let $B \in \mathbb{Z}^{n\times n}$ be a lattice basis and $C > \log(\kappa(B))$ a bound on its condition number. Let α^* be a constant determined by the Hermite constant in small dimension, and let $\alpha > 2\alpha^*$ be the desired reduction quality. Finally, let $\omega \in (2, 3]$ and $\varepsilon > 0$ be parameters bounding the runtime of size reduction, matrix multiplication, and QR factorization as described in Lemma 2. If the heuristic assumptions 1 and 2 are correct, then our algorithm returns unimodular U such that BU is $O(\alpha)$-lattice-reduced. The running time of our reduction algorithm is*

$$O\left(\left(\frac{\alpha}{\alpha - \alpha^*}\right) n^{\omega}(C + n)^{1+\varepsilon}\right).$$

Proof. First, we argue that the result of Algorithm 9 is correct. The algorithm begins by padding B to form B_{pad}, a basis of rank N. We then compress B_{pad}, at a cost of $O(n^{\omega}(C+n)^{1+\varepsilon})$ to obtain $\hat{B}_{pad}$ After reduction by `ReducePow2`, we have that $\mathrm{drop}(\hat{B}_{pad}U_1)$ is bounded. We compute U'_{pad} such that $\mathrm{drop}(B_{pad}U'_{pad})$ is bounded. We are close to done; we just need to perform a size reduction operation to ensure the final $B_{pad}U_{pad}$ is size-reduced. We just need to find an upper triangular basis that is suitably similar to $B_{pad}U'_{pad}$, size-reduce it, and apply Lemma 20. This is what the computation of B_2 through B_6 accomplishes. Note that we are careful to avoid computing $B_{pad}U'_{pad}$ directly, because we do not have a bound on $\|B_{pad}\|_{max}$. Once we have size reducing matrix U_{sr}, we compute $U_{pad} \leftarrow U'_{pad}U_{sr}$. Because of the sequence of operations, we have $B_{pad}U_{pad} \approx_{O(\gamma_2)} 2^{s+s'}B_6$ for size-reduced B_6, so with appropriate choice of γ_2, Lemma 20 implies $B_{pad}U_{pad}$ is size-reduced. By the construction of B_{pad}, we have that BU has drop bounded by $\alpha N \leq 2\alpha n$, and by definition of size reduction, BU is size-reduced as well. Therefore BU is $O(\alpha)$-lattice-reduced.

The running time of this algorithm involves a call to `ReducePow2`, which by Lemma 19 costs

$$O\left(\frac{\alpha}{\alpha-\alpha^*}N^\omega\left(\mathrm{drop}(\hat{B}_{pad})+N\right)^{1+\varepsilon}\right)=O\left(\frac{\alpha}{\alpha-\alpha^*}n^\omega\left(\mathrm{drop}(B)+n\right)^{1+\varepsilon}\right).$$

The calls to all the other functions are dominated by operations which involve at least $\log C$ bits. In particular, this is the first lattice compression and the multiplication $B_2 U'_{pad}$. These operations are bounded by $O(n^\omega(C+n))$. Noting that $\mathrm{drop}(B) \leq \mathrm{spread}(B) \leq \log \kappa(B) < C$, the running time of our lattice reduction algorithm is

$$O\left(\frac{\alpha}{\alpha-\alpha^*}n^\omega\left(C+n\right)^{1+\varepsilon}\right).$$

□