SHORT VECTOR PROBLEMS AND SIMULTANEOUS APPROXIMATION

DANIEL E. MARTIN

Abstract. In 1982, Lagarias showed that solving the approximate Shortest Vector Problem also solves the problem of finding simultaneous Diophantine approximations [17]. Here we provide the reverse reduction with polynomial complexity. It is gap-preserving provided a consistent norm is used to measure approximation quality in both problems. We also give an alternative to the Lagarias algorithm by first reducing the version of simultaneous approximation in [17] to one with no explicit range in which a solution is sought.

1. Introduction

Our primary result is to show that under the $\ell_1$, $\ell_2$, or $\ell_\infty$-norm, a short vector problem reduces with polynomial complexity to a single simultaneous approximation problem as presented in the definitions below. We use $\min^*$ to mean nonzero minimum, $\{x\}$ to denote the fractional part of $x \in \mathbb{R}^n$, and $[x]$ to denote the set $\{1, \ldots, \lfloor x\rfloor\}$ for $x \in \mathbb{R}$.

Definition 1.1. A short vector problem takes input $\alpha \in [1, \infty)$ and nonsingular $M \in M_n(\mathbb{Z})$. A valid output is $q_0 \in \mathbb{Z}^n$ with $0 < \|Mq_0\| \leq \alpha \min^*_q \|Mq\|$. Let $\text{svp}$ denote an oracle for such a problem.

Definition 1.2. A good Diophantine approximation problem takes input $\alpha, N \in \mathbb{Z}$ and $x \in \mathbb{Q}^n$. A valid output is $q_0 \in [\alpha N]$ with $\|\{q_0x\}\| \leq \alpha \min_{q \in [N]} \|\{qx\}\|$. Let $\text{gda}$ denote an oracle for such a problem.

Literature more commonly refers to a short vector problem as a Shortest Vector Problem when $\alpha = 1$ and an approximate Shortest Vector Problem otherwise (often unrestricted to sublattices of $\mathbb{Z}^n$, though we have lost no generality). A brief exposition can be found in [20]. See [12] or [18] for a more comprehensive overview, [21] for a focus on cryptographic applications, [15] for a summary of hardness results, and [10] for applications to quantum cryptography.

Regarding simultaneous approximation, Brentjes highlights several algorithms in [6]. For a sample of applications to attacking clique and knapsack-type problems see [11], [16], and [24]. Examples of cryptosystems built on the hardness of simultaneous approximation are [2], [4], and [14]. This version is taken from [7] and [22].

The reduction, given in Algorithm 3, preserves the gap $\alpha$ when either the $\ell_1$, $\ell_2$, or $\ell_\infty$-norm is used for both problems. This means the short vector problem defined by $\alpha$ and $M$ is solved by calling $\text{gda}(\alpha, x)$ for some $x \in \mathbb{Q}^n$. In Subsection 4.3 we
note how permitting any amount of gap inflation modifies the algorithm to function under an arbitrary $\ell_p$-norm. In either case, the reduction requires $O(n^3 + n^2 \log mn)$ operations on integers of length $O(n^2 \log mn)$, where $m$ is the maximum magnitude among the integers used by the input. This is proved in Theorem 4.4.

Algorithm 3 is the reverse of Lagarias’ 1982 reduction from good Diophantine approximation to SVP. (See Theorem B in [17], which refers to the problem as good simultaneous approximation. We borrow its name from [7] and [22].) An important contextual distinction is that all reductions in this paper require a consistent norm, while [17] relates simultaneous approximation under the $\ell_\infty$-norm to lattice reduction under the $\ell_2$-norm.

With this specific setup—the $\ell_\infty$-norm for GDA and the $\ell_2$-norm for SVP—ours is not the first to reverse Lagarias’ work. In a seminar posted online from July 1, 2019, Agrawal presented an algorithm achieving this reduction which was complete less some minor details [1]. Tersely stated, he takes an upper triangular basis for a sublattice of $\mathbb{Z}^n$ and transforms it inductively, using integer combinations and rigid rotations with two basis vectors at time, into a lattice (a rotated copy of the original) whose short vectors can be found via simultaneous approximation. The short vector problem defined by $\alpha$ and $M$ gets reduced to $\text{GDA}(\alpha/\sqrt{2n}, x)$, called multiple times in order to account for the unknown minimal vector length which is used to determine $x$.

In contrast, the reduction here takes a completely different approach. It finds a sublattice which is nearly scaled orthonormal, so that only one additional vector is needed to generate the original lattice. This extra vector is the input for GDA. We note that when switching between norms, our reduction is also not gap-preserving. To use Algorithm 3 to solve a short vector problem with respect to the $\ell_2$-norm via GDA with respect to the $\ell_\infty$-norm, the latter must be executed with the parameter $\alpha/\sqrt{n}$ to account for the maximum ratio of nonzero norms $\|q\|_2/\|q\|_\infty$.

The relationship between the two problems in Definitions 1.1 and 1.2 will be studied through the following intermediary.

**Definition 1.3.** A simultaneous approximation problem takes input $\alpha \in [1, \infty)$ and $x \in \mathbb{Q}^n$. A valid output is $q_0 \in \mathbb{Z}$ with $0 < \|\{q_0 x\}\| \leq \alpha \min_{q \in \mathbb{Z}} \|q\|_\infty$. Let $\text{SAP}$ denote an oracle for such a problem.

Section 2 explores the relationship between the two versions of simultaneous approximation given in Definitions 1.2 and 1.3. Among the results, only Proposition 2.1 in Subsection 2.1 is required to verify the final reduction of a short vector problem to either version of simultaneous approximation. Subsection 2.2 contains Algorithm 1. It reduces a good Diophantine approximation problem to polynomially many SAP calls, each executed with the parameter $\alpha/3.06$. So while this reduction is not gap-preserving, the inflation is independent of the input.

Section 3 reduces both versions of simultaneous approximation to SVP. It begins with Algorithm 2, which solves Definition 1.3’s version. We remark at the end of Subsection 3.1 how this reduction adapts without effort to the inhomogeneous forms of these problems—the search for $q_0 \in \mathbb{Z}$ or $q_0 \in \mathbb{Z}^n$ that makes $q_0 x - y$ or $Mq_0 - y$ small for some $y \in \mathbb{Q}^n$. Then Algorithms 1 and 2 are combined in Subsection 3.2 to solve Definition 1.2’s version of simultaneous approximation using SVP. This is our alternative to the Lagarias reduction from [17].

Finally, Algorithm 3 in Section 4 reduces a short vector problem to either GDA or SAP. It also adapts to the inhomogeneous versions of SVP and SAP (though not
SHORT VECTOR PROBLEMS AND SIMULTANEOUS APPROXIMATION

In Corollary 4.5 we observe that Algorithm 3 allows for a simpler proof of the NP-hardness of GDA under an appropriate bound on \( \alpha \) (a result first obtained in [7]). We also combine Algorithms 2 and 3 in Subsection 4.2 to solve a simultaneous approximation problem with GDA. In particular, we give all six reductions among the defined problems, as shown in the diagram below.

The two reductions in Figure 1 without algorithm numbers are achieved by following the two arrows that combine to give the same source and target. Dashed arrows indicate a norm restriction: each must be executed under either the \( \ell_1 \), \( \ell_2 \), or \( \ell_\infty \)-norm to be gap-preserving. However, we point out in Subsection 4.3 how the restriction can be alleviated to any \( \ell_p \)-norm provided we accept gap inflation by a constant arbitrarily close to 1.

The results are summarized by the following table. It uses \( m \) and \( d \) to denote the maximal magnitude among input integers and the least common denominator of the input vector, respectively. As always, \( n \) denotes the dimension of the input matrix or vector. Trivial cases that cause logarithms to equal 0 are ignored. Column descriptions follow.

| Reduction     | Operations | Integers | Inflation         | Calls          |
|---------------|------------|----------|-------------------|----------------|
| GDA \( \rightarrow \) SAP | \( n \log m \) | \( n \log m \) | 3.06 \( \log_2 d/\alpha N \) |                |
| SAP \( \rightarrow \) SVP | \( (n + \log m)^2 \) | \( n \log m \) | 1 | 1 |
| GDA \( \rightarrow \) SVP | \( (n + \log m)^2 \) | \( n \log m \) | 3.06 \( \log_2 d/\alpha N \) |                |
| SVP \( \rightarrow \) GDA | \( n^3 + n^2 \log mn \) | \( n^2 \log mn \) | 1 | 1 |
| SVP \( \rightarrow \) SAP | \( n^3 + n^2 \log mn \) | \( n^2 \log mn \) | 1 | 1 |
| SAP \( \rightarrow \) GDA | \( n^3 \log m \) | \( n^3 \log m \) | 1 | 1 |

Table 1. Summary of reduction complexities and gap inflations.

**Operations:** Big-O bound on the number of arithmetic operations per oracle call.

**Integers:** Big-O bound on the length of integers used throughout the reduction.

**Inflation:** Maximum gap inflation. For example, to solve a good Diophantine approximation problem with some \( \alpha \) using Algorithm 1, SAP is called with \( \alpha/3.06 \).

**Calls:** Upper bound on the number of required calls to the oracle.

2. VERSIONS OF SIMULTANEOUS APPROXIMATION

2.1. **SAP to GDA.** Rather than give a complete reduction from a simultaneous approximation problem to GDA, which is postponed until the end of Subsection 4.2, the purpose of this subsection is to observe a condition on the input that makes these two versions of simultaneous approximation equivalent.

**Proposition 2.1.** Suppose the \( i \)th coordinate of \( \mathbf{x} \) is of the form \( x_i = 1/d \), where \( d \in \mathbb{N} \) makes \( d\mathbf{x} \in \mathbb{Z}^n \). Under an \( \ell_p \)-norm, GDA(\( \alpha, \mathbf{x}, N \)) solves the simultaneous approximation problem defined by \( \alpha \) and \( \mathbf{x} \) with \( N = d/2\alpha \).

**Proof.** Let \( q_{\min} \in [d/2] \) be such that \( \|q_{\min}\mathbf{x}\| \) is the nonzero minimum. If \( \|q_{\min}\mathbf{x}\| \geq 1/2\alpha \) then every integer in \( [N] = [d/2\alpha] \) solves the simultaneous approximation problem defined by \( \alpha \) and \( \mathbf{x} \), so assume this is not the case.
Since we are working under an $\ell_p$-norm, $\|\{q_{\text{min}}x]\|$ is an upper bound for its $i^{th}$ coordinate, $q_{\text{min}}/d$. Combining this with our assumption that $\|\{q_{\text{min}}x]\| < 1/2\alpha$ gives $q_{\text{min}} \in [d/2\alpha] = [N]$, implying $\min_{q \in [N]} \|\{qx]\| = \min_{q \in \mathbb{Z}} \|\{qx]\|$. And because $\alpha N < d$, it is guaranteed that GDA($\alpha$, $x$, $N$) is not a multiple of $d$. □

Note that without an assumption on $x$ like the one used in this proposition, there is no natural choice for $N$, the parameter that differentiates these two problems. If we set $N$ too small, say with $N < d/2$, then $\min_{q \in [N]} \|\{qx]\|$ may be unacceptably larger than $\min_{q \in \mathbb{Z}} \|\{qx]\|$, potentially making GDA’s approximation poor. If we set $N$ too large, say with $N \geq d/\alpha$ (giving $d \in [\alpha N]$), then GDA may return $d$, which is not a valid output for the initial simultaneous approximation problem. So for $\alpha \geq 2$, then $d/2 \geq d/\alpha$ means there is no choice for $N$ that avoids both inequalities and guarantees that GDA($\alpha$, $x$, $N$) is a solution.

To get around this, our strategy is to first reduce a simultaneous approximation problem to SVP (done in Algorithm 2). Then, in Algorithm 3, which reduces a short vector problem to SAP, we are careful to produce an input vector for the oracle that satisfies the hypothesis of Proposition 2.1, making GDA an equally valid option.

2.2. GDA to SAP. By reducing modulo the least common denominator of $x$, call it $d$ again, SAP can be thought of as having an implicit bound of $d/2$ on its output. The problem faced in this reduction is that outputs for a good Diophantine approximation problem are bounded by $d/\alpha \geq \alpha N$ because $\alpha N < d/2$. Then, $\min_{q \in [N]} \|\{qx]\|$ means there is no choice for $d/\alpha \geq \alpha N$ that avoids both inequalities and guarantees that GDA($\alpha$, $x$, $N$) is a solution.

To get around this, our strategy is to first reduce a simultaneous approximation problem to SVP (done in Algorithm 2). Then, in Algorithm 3, which reduces a short vector problem to SAP, we are careful to produce an input vector for the oracle that satisfies the hypothesis of Proposition 2.1, making GDA an equally valid option.

Algorithm 1: A reduction from a good Diophantine approximation problem to multiple calls to SAP under a consistent norm.

| input: $\alpha, N \in [1, \infty)$, $x = (x_1, \ldots, x_n) \in \mathbb{Q}^n$ |
| output: $q_0 \in [\alpha N]$ with $\|\{q_0x]\| \leq \alpha \min_{q \in [N]} \|\{qx]\|$ |
| 1: $d \leftarrow \text{lcm}(x_1, \ldots, x_n) > 0$ |
| 2: while $d > \alpha N$ do |
| 3: $d \leftarrow \text{SAP}((\alpha/3.06, x) \bmod d)$ ▶ good, but large approximation |
| 4: $x \leftarrow x - \{dx\}/d$ ▶ now lcm($x) = d$, at most half of the previous iteration’s lcm |
| 5: return $d$ |

Proposition 2.2. The output of Algorithm 1 solves the initial good Diophantine approximation problem.

Proof. Let $d_i$ and $x_i$ denote the values of “$d$” and “$x$” after $i$ while loops have been completed. In particular, $d_0$ and $x_0$ are defined by the input. Also let $I + 1$ be the total number of while loops executed so that the output is $d_{I+1}$.

The triangle inequality gives

$$(2.1) \quad \|\{d_{I+1}x]\| \leq \|\{d_Ix_i]\| + d_{I+1} \sum_{i=1}^I \|x_i - x_{i-1}\|. $$
With \( \lambda_i = \min_{q \in [N]} \|q \mathbf{x}_i\| \), the choice of \( d_{i+1} \) bounds the first summand by \( \alpha \lambda_I/c \), where \( c = 3.06 \) in Algorithm 1 but is left undetermined for now. Similarly, the choice of \( d_i = \text{SAP}(\alpha/c, \mathbf{x}_{i-1}) \) and the fact that \( d_i > \alpha N \geq \overline{N} \) make
\[
\| \mathbf{x}_i - \mathbf{x}_{i-1} \| = \| \frac{d_i}{d_i} \mathbf{x}_{i-1} \| \leq \frac{\alpha \min_{q \in \mathbb{Z}} \|q \mathbf{x}_{i-1}\|}{cd_i} \leq \frac{\alpha \lambda_{i-1}}{cd_i}.
\]
So to bound (2.1) it must be checked that the \( \lambda_i \)'s are not too large. To this end, fix some \( i \leq I \) and let \( q_{\min} \in [N] \) satisfy \( \|q_{\min} \mathbf{x}_{i-1}\| = \lambda_{i-1} \). Then we have the following upper bound on \( \lambda_i \), where the three inequalities are due to the triangle inequality, inequality (2.2), and \( q_{\min} \leq N < d_i/\alpha \leq d_i/2^{I-i} c \), respectively.
\[
\|q_{\min} \mathbf{x}_i\| \leq \lambda_{i-1} + q_{\min} \|\mathbf{x}_i - \mathbf{x}_{i-1}\| \leq \lambda_{i-1} \left(1 + \frac{\alpha q_{\min}}{cd_i}\right) < \lambda_{i-1} \left(1 + \frac{1}{2^{I-i} c}\right).
\]
Inductively, this gives
\[
\lambda_i < \lambda_0 \prod_{j=1}^i \left(1 + \frac{1}{2^{I-j} c}\right).
\]
Now the three numbered inequalities above can be combined to get
\[
\|d_{I+1} \mathbf{x}\| \leq \frac{\alpha d_{I+1}}{c} \sum_{i=0}^I \frac{\lambda_i}{d_{i+1}} \leq \frac{\alpha}{c} \sum_{i=0}^I \frac{\lambda_i}{2^{I-i}} \leq \frac{\alpha \lambda_0}{c} \sum_{i=0}^I \frac{1}{2^{I-i}} \prod_{j=1}^i \left(1 + \frac{1}{2^{I-j} c}\right).
\]
Thus the output approximation quality, \( \|d_{I+1} \mathbf{x}\| \), is bounded by \( \alpha \min_{q \in [N]} \|q \mathbf{x}\| = \alpha \lambda_0 \) provided \( c \) satisfies
\[
1 \geq \frac{1}{c} \sum_{i=0}^\infty \frac{1}{2^i} \prod_{j=1}^\infty \left(1 + \frac{1}{2^j c}\right).
\]
This justifies our choice of \( c = 3.06 \) in line 4. \( \square \)

**Proposition 2.3.** Let \( m > 1 \) be the maximum magnitude among integers defining \( \mathbf{x} \), and let \( d > 1 \) be its least common denominator. The reduction in Algorithm 1 requires an initial \( O(n \log m) \) operations plus \( O(n) \) operations for each call to SAP, of which there are at most \( \lceil \log_2 (d/\alpha N) \rceil \), on integers of length \( O(n \log m) \).

**Proof.** Repeated applications of the Euclidean algorithm computes the least common denominator of the entries of \( \mathbf{x} \) with \( O(n \log m) \) operations on integers of length \( O(n \log m) \). Reducing \( \text{SAP}(\alpha/3.06, \mathbf{x}) \) modulo \( d \) in line 3 decreases the least common denominator of each successive value of \( \mathbf{x} \) by at least a factor of 1/2. This gives a bound on the number of \textbf{while} loops of \( \lceil \log_2 (d/\alpha N) \rceil \). New numerators for \( \mathbf{x} \) are bounded by \( d \), so the integer length bound does not change. \( \square \)

We remark that this reduction does not adapt to inhomogeneous forms of these problems. In seeking \( q_0 \in [\alpha N] \) with \( \|q_0 \mathbf{x} - \mathbf{y}\| \leq \alpha \min_{q \in [N]} \|q \mathbf{x} - \mathbf{y}\| \), we might consider running the \textbf{while} loop in Algorithm 1 (still with our usual homogeneous version of SAP to replace \( \mathbf{x} \) with something nearby of smaller denominator) to reduce \( d \) for all but the last iteration, where an inhomogeneous form of SAP could be executed. This fails in general. The inequality in (2.3) is still true, but now \( \lambda_0 = \min_{q \in [N]} \|q \mathbf{x}\| \) is not relevant. The minimum that matters is \( \min_{q \in [N]} \|q \mathbf{x} - \mathbf{y}\| \), and this can be arbitrarily smaller than \( \lambda_0 \).
3. Reducing to \text{svp}

First we restrict our attention to Definition 1.3’s version of simultaneous approximation (\text{SAP}) in Algorithm 2. Then, in Subsection 3.2, we will compare the combination with Algorithm 1 to Lagarias’ reduction in [17] from a good Diophantine approximation problem.

The remainder of our work uses the following.

**Definition 3.1.** Let \text{B{\v{e}}ZOUT} execute the Euclidean algorithm on \((a_1, a_2) \in \mathbb{Z}^2\) to get \((b_1, b_2) \in \mathbb{Z}^2\) satisfying \(a_1 b_1 + a_2 b_2 = |\gcd(a_1, a_2)|\) with \(|b_1| \leq |a_2/2 \gcd(a_1, a_2)|\).

**Notation 3.2.** Given a matrix \(M\), \(M_{i,j}\) is the entry in the \(i\)th row and \(j\)th column. An asterisk, like \(M_{*,*}\), is used to denote an entire row (as in this case) or column.

### 3.1. \text{sap} to \text{svp}

For a simultaneous approximation problem, the output \(q_0\) makes \(\|q_0 x\|\) small, where the fractional part is \(q_0\) shifted by something in \(\mathbb{Z}^n\). So we are finding short vectors in the lattice generated by \(x\) and the standard basis for \(\mathbb{Z}^n\). The only thing that must be adjusted to make this an input for \text{svp} is the number of vectors—the matrix \(M\) in Definition 1.1 is square. The next algorithm simply replaces these \(n + 1\) vectors with a basis for the same lattice.

**Algorithm 2:** A gap-preserving reduction from a simultaneous approximation problem to one call to \text{svp} under a consistent norm.

```
input: \(\alpha \in [1, \infty), x = (x_1, \ldots, x_n) \in \mathbb{Q}^n\)
output: \(q_0 \in \mathbb{Z}\) with \(0 < \|q_0 x\| \leq \alpha \min_{q \in \mathbb{Z}} \|qx\|\)

1: \(d \leftarrow \text{lcd}(x_1, \ldots, x_n)\)
2: \(M \leftarrow 0 \in M_n(\mathbb{Z})\) \quad \triangleright \text{M is to have first column } dx \text{ and generate } d\mathbb{Z}^n
3: \(a, M_{1,1} \leftarrow dx_1\)
4: for \(i \leftarrow 2\) to \(n\) do
5: \hspace{1em} if \(i = n\) then \quad \triangleright \text{may be that } \gcd(dx_1, \ldots, dx_n) \neq 1
6: \hspace{2em} \(b \leftarrow a/\gcd(a, dx_n)\)
7: \hspace{2em} while \(\gcd(b, dx_n) \neq 1\) do
8: \hspace{3em} \(b \leftarrow b/\gcd(b, dx_n)\)
9: \hspace{2em} \(x_n \leftarrow x_n + b\) \quad \triangleright \text{so shift } x_n \text{ by } b \text{ to fix this}
10: \hspace{1em} \((b_1, b_2) \leftarrow \text{B{\v{e}}ZOUT}(a, dx_i)\)
11: \hspace{1em} \(M_{*,i} \leftarrow (-1)^{i-1} db_2 M_{*,1}/a\) \quad \triangleright \text{determinant of columns } 2, \ldots, i \text{ is}
12: \hspace{1em} \(M_{1,i} \leftarrow dx_i\) \quad \(-1)^{i-1} db_2/a \text{ times determinant}
13: \hspace{1em} \(M_{i,i} \leftarrow db_1\) \quad \text{of columns } 1, \ldots, i-1
14: \hspace{1em} \(a \leftarrow ab_1 + dx_i b_2\) \quad \triangleright \text{top, left determinant is } d^{-1} a
15: \text{return } \text{svp}(a, M)_{1}\) \quad \triangleright \text{first coordinate is a solution}
```

**Proposition 3.3.** The output of Algorithm 2 solves the initial simultaneous approximation problem.

**Proof.** The claim follows if we can show that the \text{for} loop ends with a matrix \(M \in M_n(\mathbb{Z})\) whose columns generate the same lattice as \(dx\) and \(d\mathbb{Z}^n\), where \(d\) is the least common denominator of the entries of \(x\).

Fix some \(i \geq 1\) and let \(a, (b_1, b_2), \text{ and } M\) be as they are immediately after line 13 is executed for the \((i-1)\)th time. Assume for induction that \(a = \gcd(dx_1, \ldots, dx_{i-1})\) and that this is the determinant of the top, left \((i-1)\times(i-1)\) minor of \(M\) after...
scaling columns 2 through $i - 1$ by $1/d$. Scaling column $i$ by $1/d$ as well, lines 11, 12, and 13 give the top, left $i \times i$ minor of $M$ the following form (entries marked with a * are irrelevant):

$$
\begin{bmatrix}
  dx_1 & \cdots & * & (−1)^{i−1}db_2x_1/a \\
  \vdots & \vdots & \vdots & \vdots \\
  dx_{i−1} & \cdots & * & (−1)^{i−1}db_2x_{i−1}/a \\
  dx_i & 0 & \cdots & 0 \ b_1
\end{bmatrix}
$$

From our induction hypothesis and cofactor expansion along the bottom row, this determinant is computed to be $ab_1 + dx_ib_2$. By line 14 this is the next value of “$a$,” which is then $\gcd(a, dx_i) = \gcd(dx_1, \ldots, dx_i)$ by line 10.

By induction, then, once the for loop is completed, scaling all but the first column of $M$ by $1/d$ gives a matrix with determinant $\gcd(dx_1, \ldots, dx_n + db)$, where $b$ is the integer produced in the if block and $x_n$ is its initial value. Letting $a = \gcd(dx_1, \ldots, dx_{n−1})$ (its value in line 6), since $d$ is the least common denominator of the entries of $x$ we have $\gcd(a, dx_n, d) = 1$. In particular, because $a/b$ and $b$ split $a$ into those primes which divide $dx_n$ and those that do not by the while loop, we get $\gcd(a, dx_n + db) = 1$. Thus our scaled copy of $M$ is in $\text{SL}_n(\mathbb{Z})$, meaning unscaling the last $n − 1$ columns leaves a matrix that generates $d\mathbb{Z}^n$. Since these unscaled columns are now multiples of $d$, the lattice defined by the final value of $M$ is seen to equal that generated by $dx$ and $d\mathbb{Z}^n$. The first column is $dx$ with the last entry shifted by a multiple of $d$, so the first coordinate of $\text{sVP}(\alpha, M)$ solves the original simultaneous approximation problem.

\[\square\]

**Proposition 3.4.** Let $m > 1$ be the maximum magnitude among integers defining $x$. The reduction in Algorithm 2 requires $O((n + \log m)^2)$ operations on integers of length $O(n \log m)$.

**Proof.** As with Algorithm 1, line 1 requires $O(n \log m)$ operations on and resulting in integers of length $O(n \log m)$. Within the for loop, line 10 requires $O(\log m)$ operations and line 11 requires $O(n)$ operations. To see the former, first recall from the previous proof that $a = \gcd(dx_1, \ldots, dx_i)$ after the $(i − 1)^{\text{th}}$ while loop iteration. So running the Euclidean algorithm again in the next iteration on the pair $(\gcd(dx_1, \ldots, dx_i), dx_{i+1})$ requires the same number of steps as with the pair $(\gcd(d_{i+1}x_1, \ldots, d_{i+1}x_i), d_{i+1}x_{i+1})$, where $d_{i+1} = \text{lcm}(x_1, \ldots, x_{i+1})$. But note that $\gcd(d_{i+1}x_1, \ldots, d_{i+1}x_i) \leq m \gcd(dx_1, \ldots, dx_i)$ and that $\gcd(d_{i+1}x_1, \ldots, d_{i+1}x_i)$ divides the greatest common divisor of the (reduced) numerators of $x_1, \ldots, x_i$, which is bounded by $m$.

For the last while loop iteration, when $i = n$, the if block requires the Euclidean algorithm at most $\log_2 a = \log_2 \left| \gcd(dx_1, \ldots, dx_{n−1}) \right|$ times. We have just this shown to be $O(\log m)$. This gives a bound of $O(\log^2 m)$ on the number of operations to compute the value of $b$ used in line 9.

\[\square\]

We remark that this algorithm does adapt to inhomogeneous forms of these problems. In other words, to find $q_0 \in \mathbb{Z}$ with $0 \leq \|q_0x - y\| \leq \min_{q \in \mathbb{Z}} \|qx - y\|$ we can perform the same reduction, but finish by executing a function which solves the close vector problem (see chapter 18 of [12], for example) defined by $\alpha, M$, and $dy$ (provided the problem is genuinely inhomogeneous—that $qx - y \in \mathbb{Z}^n$ has no solution $q \in \mathbb{Z}$—otherwise stick with $\text{sVP}$).
3.2. **GDA to SVP.** Combining Algorithms 1 and 2 gives an alternative to the Lagarias reduction from good Diophantine approximation to SVP in [17]. Specifically, we execute Algorithm 1 but replace $\text{sap}(1/3.06, x)$ in line 3 with the output of Algorithm 2 (and ignore the now redundant computation of $d$ in line 1 of Algorithm 2). By Proposition 2.3, this requires at most $\lceil \log_2(d/\sqrt{\alpha}) \rceil$ calls to SVP. And Proposition 3.4 states that each call requires $O((n + \log m)^2)$ operations on integers of length $O(n \log m)$. We have proved the following.

Recall that switching from $\ell_2$ to $\ell_\infty$ decreases a nonzero norm by at most a factor of $1/\sqrt{n}$. In particular, by executing this combination of Algorithms 1 and 2 with respect to the $\ell_\infty$ norm we get an $\ell_\infty$ solution to the initial good Diophantine approximation problem provided we execute $\text{svp}(1/3.06\sqrt{n}, M)$. In [17], Lagarias achieves this reduction with exactly $[n + \log_d dN]$ instances of $\text{svp}(1/\sqrt{\alpha}N, M)$, where $d$ still denotes the least common denominator of the entries of $x$. The reduction requires an initial $O(n \log m)$ arithmetic operations (to compute $d$), then only one additional operation per instance. The integers involved have input length $O(\log m^n N)$.

Whether the benefit of fewer instances of SVP outweighs the increased operations per instance depends on the complexity of SVP. For example, since the number of calls to the oracle is strictly fewer here, ours is an asymptotic improvement over Lagarias’ reduction when the time complexity of SVP exceeds $O((n + \log m)^2)$.

4. **Reducing to GDA or SAP.**

We begin again by focusing first on the reduction to SAP.

4.1. **Intuition.** Consider an input matrix $M \in \mathbb{M}_n(\mathbb{Z})$ for a short vector problem. Let $d = \det M$, and let $e_1, ..., e_n$ denote the standard basis vectors for $\mathbb{Z}^n$. If there was one vector, call it $b \in \mathbb{Z}^n$, for which the set $\{Mb, de_1, ..., de_n\}$ generated the columns of $M$, our reduction would just amount to finding it. This is exactly the setup for simultaneous approximation: $n + 1$ vectors, $n$ of which are scaled orthonormal. A solution could be obtained by doing simultaneous approximation on $Mb/d$, scaling the resulting short vector by $d$, and applying $M^{-1}$ (to comply with Definition 1.1). Unfortunately, unless $n \leq 2$ or $d = \pm 1$, such a $b$ does not exist. Indeed, the adjugate matrix, $\text{adj} M$, has at most rank 1 over $\mathbb{Z}/p\mathbb{Z}$ for a prime $p$ dividing $d$. So at least $n - 1$ additional vectors are required to have full rank modulo $p$, which is a prerequisite to having full rank over $\mathbb{Q}$. But asking that $Mb$ generate the columns of $M$ alongside $de_1, ..., de_n$ is equivalent to asking that $b$ generate $\mathbb{Z}^n$ alongside the columns of $\text{adj} M$.

What mattered is the matrix with columns $de_1, ..., de_n$ being scaled orthonormal. As such, multiplying by it or its inverse has no affect on a vector’s relative length. So we plan to find a different set of $n$ column vectors—a set for which just one additional $Mb$ is needed to generate the original lattice—which is nearly scaled orthonormal, making the effect of the corresponding matrix multiplication on the gap, $\alpha$, negligible. The initial short vector problem becomes a search for an integer combination of $Mb$ and these columns, say $c_1, ..., c_n$. Then we can solve the simultaneous approximation problem defined by $\alpha$ and $[c_1 \cdots c_n]^{-1}Mb$. This works as long as multiplying by $[c_1 \cdots c_n]$ changes the ratio between the lengths of the shortest vector and our output by less than whatever is afforded by the fact that lattice norms form a discrete set.
An arbitrary lattice may have all of its scaled orthonormal sublattices contained in $d\mathbb{Z}^n$. So as candidates for the matrix $[a_1 \cdots a_n]$ we look for something of the form $cd \cdot \text{Id} + M = M(c \cdot \text{adj} M + A)$ for some $c \in \mathbb{Z}$ and $A \in M_n(\mathbb{Z})$. If the entries of $A$ are sufficiently small then multiplication by this matrix has a similar effect on relative vector norms as multiplying by $cd \cdot \text{Id}$, which is to say very little.

The author has found it simpler to choose $A$ first (the identity always works), then pick $c$ under both a size constraint (to make the presence of $A$ insignificant) and a congruence constraint (to guarantee that $b$ exists). In contrast, it is also possible to choose $c$ sufficiently large first then find $A$ so that $b$ exists. This works since the smallest entries of an admissible $A$ grow more slowly than $c$. In any case, we should tailor our choice so that one of the coordinates of $(c \cdot \text{adj} M + A)^{-1}b$ is of the form $1/\det(c \cdot \text{adj} M + A)$ to admit Proposition 2.1 and hence GDA.

4.2. SVP TO GDA OR SAP. In addition to Definition 3.1 and Notation 3.2 from Section 3, the following will be used.

Notation 4.1. Given a permutation, $\pi \in S_n$, and an $n \times n$ matrix $M$, let $\pi M$ and $M\pi$ denote the resulting matrix after permuting rows and columns, respectively.

Algorithm 3: A gap-preserving reduction from a short vector problem with $n \geq 2$ to one call to GDA or SAP under a consistent $\ell_p$-norm with $p \in \{1, 2, \infty\}$.

```
input: a \geq b \in \mathbb{N} (\alpha = a/b), M \in M_n(\mathbb{Z}) with \det M \neq 0
output: q_0 \in \mathbb{Z}^n with 0 < \|Mq_0\| \leq \alpha \min_{q \in \mathbb{Z}^n} \|Mq\|

1. |M_{i_0,j_0}| \leftarrow \max_{i,j} |M_{i,j}|
2. c \leftarrow M_{i_0,j_0} \det M[5\alpha^2 n^3/det M]
3. M \leftarrow c \cdot (j_0 2) \text{adj} M(i_0 1) + \text{Id}
4. (b_1, b_2) \leftarrow \text{BEZOUT}((\text{adj} M)_{1,1}, (\text{adj} M)_{1,2})
5. x \leftarrow M^{-1}(b_1, b_2, 0, 0, ..., 0)
6. q_0 \leftarrow \text{GDA}(\alpha, x, \text{det} M/2\alpha) or \text{SAP}(\alpha, x)
7. return \((j_0 2)M\{q_0 x\})
```

Lemma 4.2. Let $M'$ denote the value of “$M$” after line 3 of Algorithm 3. Then $\gcd((\text{adj} M')_{1,1}, (\text{adj} M')_{1,2}) = 1$.

Proof. Let $f(x) = \sum_j f_j x_j$ and $\tilde{f}(x) = \sum_j \tilde{f}_j x_j$ be the determinants of the $(n - 1) \times (n - 1)$ minors of $x \cdot (j_0 2)\text{adj} M(i_0 1) + \text{Id}$ obtained by removing the first column and either the first or second row, respectively. Then $f_0 = 1$ and $\tilde{f}_0 = 0$. This makes the determinant of the matrix below $-\tilde{f}_{n-1}^{-1}$, so the system of equations,

\[
\begin{bmatrix}
    f_{n-1} & 0 & \cdots & 0 \\
    \vdots & \ddots & \ddots & \vdots \\
    f_1 & f_{n-1} & \cdots & f_{n-1} \\
    f_0 & f_{n-2} & \cdots & f_{n-2} \\
    0 & f_0 & \cdots & \tilde{f}_0
\end{bmatrix}
\begin{bmatrix}
    g_{n-2} \\
    \vdots \\
    g_0
\end{bmatrix}
= 
\begin{bmatrix}
    0 \\
    \vdots \\
    0
\end{bmatrix},
\]
has a solution in integers. That is, \( f(x)g(x) + \tilde{f}(x)\tilde{g}(x) = \tilde{f}_{n-1} \), where \( g(x), \tilde{g}(x) \in \mathbb{Z}[x] \) have coefficients from the column vector above. But
\[
\tilde{f}_{n-1} = -(\text{adj}(j_0 2)\text{adj} M(i_0 1)))_{1,2} = -M_{i_0,j_0} \det M^{n-2}.
\]
This implies \( \gcd(f(c), \tilde{f}(c)) \) divides a power of \( M_{i_0,j_0} \det M \) for any \( c \in \mathbb{Z} \). Recalling that \( f_0 = 1 \), we have \( \gcd(f(c), c) = 1 \). Thus \( \gcd(f(c), \tilde{f}(c)) = 1 \) whenever \( M_{i_0,j_0} \det M \mid c \), as is the case for our choice in line 2.

\[\text{Theorem 4.3.} \quad \text{Under the } \ell_1, \ell_2, \text{ or } \ell_\infty\text{-norm, the output of Algorithm 3 solves the initial short vector problem.}\]

\[\text{Proof.} \quad \text{There are two parts to the proof: 1) showing that the algorithm replaces the columns of } M \text{ with } n + 1 \text{ vectors that define the same lattice, } n \text{ of them being “nearly” scaled orthonormal, and 2) showing that nearly scaled orthonormal is as good as being scaled orthonormal.}
\]

For part 1), let \( M' = c \cdot (j_0 2)\text{adj} M(i_0 1) + \text{Id} \) (the value of “\( M' \)” after line 3), and let \( (b_1, b_2) \) be as in line 4. With \( b = (b_1, b_2, 0, ..., 0) \), Lemma 4.2 gives the following \( x_1 \) coordinate:
\[
x = M'^{-1}b = \frac{(1, x_2, ..., x_n)}{\det M'}.
\]

By Cramer’s rule [8], the 1 in the first coordinate is the determinant after replacing the first column of \( M' \) by \( b \), so that these \( n \) columns generate \( \mathbb{Z}^n \). This in turn shows that the columns of \( M(j_0 2)M' \) combined with \( M(j_0 2)b \) generate the same lattice as the columns of \( M \). Also note by Proposition 2.1, as the comment in line 6 highlights, that the 1 in the first coordinate of \( \det M' \cdot x \) makes GDA and SAP interchangeable with the parameter \( N \) set to \( \det M'/2\alpha \).

Instead of finding a short integer combination of \( M(j_0 2)b \) and the columns of \( M'(j_0 2) = c \det M \cdot \text{Id}(i_0 1) + M(j_0 2) \) (which is “nearly” scaled orthonormal), Algorithm 3 uses \( (M(j_0 2)M')^{-1}(M(j_0 2)b) = x \) and the columns of \( (M(j_0 2)M')^{-1}(M(j_0 2)M') = \text{Id} \). Part 2) of the proof is to make precise the insignificance of the second matrix summand, \( M(j_0 2) \), in (4.1). We begin by computing how much multiplication by the full matrix in (4.1) is allowed to inflate the gap without invalidating the output of GDA or SAP.

By Minkowski’s theorem [19], the magnitude of the shortest vector in the original lattice with respect to the \( \ell_\infty \)-norm is not more than \( |\det M|^{1/\alpha} \). So under an \( \ell_p \)-norm with \( p \in \mathbb{N} \), the shortest vector has some magnitude, say \( \lambda \), with \( (n^{1/p})|\det M|^{1/\alpha} \geq \lambda^p \in \mathbb{Z} \). In particular, \( n|\det M|^{2/n} \geq \lambda^2 \in \mathbb{Z} \) when \( p \in \{1, 2, \infty\} \). Now, if \( q \in \mathbb{Z}^n \) is such that \( |Mq|^2 < (a^2\lambda^2 + 1)/b^2 \), then it must be that \( |Mq| \leq a\lambda/b \) since there are no integers strictly between \( (a\lambda/b)^2 \) and \( (a^2\lambda^2 + 1)/b^2 \). So multiplication by \( M(j_0 2)M' \) must inflate the gap between the norms of our output vector and the shortest vector by less than
\[
(4.2) \quad \frac{\sqrt{a^2\lambda^2 + 1}}{b\alpha\lambda} = \frac{\sqrt{a^2\lambda^2 + 1}}{a\lambda} \geq \frac{\sqrt{a^2n} |\det M|^{2/n}}{a\sqrt{n} |\det M|^{1/n}}.
\]

Since scaling does not affect the ratio of vector norms, to determine the effect of multiplication by (4.1) it suffices to consider the matrix
\[
(4.3) \quad \text{Id}(i_0 1) + M(j_0 2)/c \det M
\]
instead. If \( q_{\text{min}} \) was a shortest nonzero vector in the simultaneous approximation lattice generated by \( Z^n \) and \( x \), then the shortest vector after applying (4.3) to this lattice has norm at least \((1 - \|M\|_{\text{op}}/c \det M)\|q_{\text{min}}\|\), where \( \|M\|_{\text{op}} \) is the operator norm. Similarly, the vector \( \{q \} \) obtained using \( q_0 \) from line 6 increases in norm by at most a factor of \((1 + \|M\|_{\text{op}}/c \det M)\).

To bound \( \|M\|_{\text{op}} \), recall that \( |M_{i_o,j_0}| = \max_{i,j} |M_{i,j}| \). For a unit vector, \( u \in \mathbb{R}^n \), we then have

\[
\|Mu\| \leq n\|Mu\|_{\infty} < n^2|M_{i_o,j_0}|\|u\|_{\infty} \leq n^2|M_{i_o,j_0}|
\]

(Note the waste under the \( \ell_2 \) or \( \ell_{\infty} \)-norm.) This is an upper bound for \( \|M\|_{\text{op}} \).

Combining it with our conclusion regarding (4.2) shows that it suffices to verify that the following inequality holds:

\[
\frac{1 + n^2|M_{i_o,j_0}/c \det M|}{1 - n^2|M_{i_o,j_0}/c \det M|} \leq \frac{\sqrt{a^2n|\det M|^{2/n} + 1}}{a\sqrt{n}|\det M|^{1/n}}.
\]

The choice of \( c \) in line 2 is \( M_{i_o,j_0} \det M [5a^2n^3/\det M] \), making the denominator above positive. So we can solve for \( |c| \) in (4.5) to get a desired lower bound of

\[
\frac{\sqrt{a^2n|\det M|^{2/n} + 1} + a\sqrt{n}|\det M|^{1/n} - n^2|M_{i_o,j_0}|}{\sqrt{a^2n|\det M|^{2/n} + 1} - a\sqrt{n}|\det M|^{1/n}} < \frac{(5a^2n|\det M|^{2/n}n^2|M_{i_o,j_0}|)}{|\det M|}.
\]

This is satisfied since Algorithm 3 assumes \( n \geq 2 \).

**Theorem 4.4.** Let \( n = \max(a^{1/n}, |M_{i_o,j_0}|) \). The reduction in Algorithm 3 requires \( O(n^3 + n^2 \log mn) \) operations on integers of length \( O(n^2 \log mn) \).

**Proof.** Computing determinants, adjugates, and inverses requires \( O(n^3) \) operations. See the Bareiss algorithm in [5], for example, which has polynomial bit complexity.

We also need an expression in terms of \( a \cdot |M_{i_o,j_0}| \), and \( n \) for the number of operations required by BÉZOUT in line 4. With the Euclidean algorithm this is \( O(\log |(\text{adj } M')_{1,1}|) \), where \( M' \) is the value of “\( M' \)” after line 3. By Hadamard’s inequality [13], \( |\det M| \) and the magnitudes of the entries in \( \text{adj } M \) are bounded by \((\sqrt{n}|M_{i_o,j_0}|)^n\). In particular, the entries in \( M' \) have length \( O(\log a(n|M_{i_o,j_0}|)) \).

The entries in \( \text{adj } M' \) are \((n - 1) \times (n - 1)\) minor determinants, so we apply Hadamard’s inequality again to get \( \log |(\text{adj } M')_{1,1}| = O(\log a(n|M_{i_o,j_0}|)) = O(n^2 \log mn) \) as claimed.

In [9], Dinur proves the NP-hardness of short vector problems under the \( \ell_{\infty} \)-norm when \( \alpha = n^{c/\log \log n} \) for some \( c > 0 \) by giving a direct reduction from the Boolean satisfiability problem (SAT). As a consequence, Theorems 4.3 and 4.4 prove the same for both good Diophantine approximation and simultaneous approximation problems.

**Corollary 4.5.** Good Diophantine approximation and simultaneous approximation problems are NP-hard with \( \alpha = n^{c/\log \log n} \) for some \( c > 0 \).

This result is already known for good Diophantine approximation [7], though the reduction \( \text{SAT} \rightarrow \text{SVP} \rightarrow \text{GDA} \) completed here is simpler. In [7], Chen and Meng adapt the work of Dinur as well as Rössner and Seifert [23] to reduce SAT to the problem of finding short integer vectors that solve a homogeneous system of linear equations (HLS). It works by first reducing to the problem of finding pseudo-labels for a regular bipartite graph (PSL) with an algorithm in [3]. From HLS, [23] is further
employed to reduce to the same problem but consisting of only one linear equation (SIR), which is then reduced to GDA in a different paper of Rössner and Seifert [22]. Each of the reductions, SAT → PSL → HLS → SIR → GDA, is gap-preserving and under the \( \ell_\infty \)-norm.

Currently, short vector problems are only known to be NP-hard under the \( \ell_p \)-norm. But there are other hardness results under a general \( \ell_p \)-norm for which Theorems 4.3 and 4.4 can be considered complementary. See [15] for an exposition.

Another corollary is the reduction from a simultaneous approximation problem to GDA, giving the final row of Table 1. By Proposition 3.4, Algorithm 2 results in one call to SVP with integers of length \( O(n \log m) \), where \( m \) is the maximum magnitude among integers defining the input vector, \( x \). This remains true if we replace \( m \) with \( a^{1/n^2} \) if it happens to be larger, where \( a = a/b \). Then Theorem 4.4 can be used. It implies the reduction to SAP requires \( O(n^3 + n^3 \log mn^{1/n}) = O(n^3 \log m) \) (absorbing the number of operations required by Algorithm 2) on integers of length \( O(n^3 \log mn^{1/n}) = O(n^3 \log m) \).

4.3. Further discussion. The last algorithm restricted our typical use of the \( \ell_p \)-norm to \( p \in \{1, 2, \infty\} \), so we will discuss what happens when attempting a more general approach.

Transforming a lattice with a matrix that is not scaled orthonormal, as is the case with (4.1), affects relative vector norms. Multiplication by \( M(j_0/2)M' \) in the previous proof may change the gap between the length of the shortest vector in the simultaneous approximation lattice and that of the vector output by GDA or SAP. That this potential inflation does not invalidate our output relies on the set of vector norms being discrete and \( \alpha \) being rational—facts that were exploited to produce the expression in (4.2). The idea behind that paragraph is to find a nonempty interval \((a\lambda, a'\lambda)\), where \( \lambda = \min_{q \in \mathbb{Z}^n} \|Mq\| \), that contains no norms from the lattice defined by \( M \) (or even \( \mathbb{Z}^n \) for the interval tacitly given in the proof). This creates admissible inflation \( a'/\alpha \), which is (4.2).

The purpose of restricting to \( \ell_1 \), \( \ell_2 \), or \( \ell_\infty \) is to facilitate finding this interval. Knowing that \((b\alpha\lambda)^2 \in \mathbb{Z} \) for some \( b \in \mathbb{Z} \) simplifies the search for \( a' \). The same is true for any \( \ell_p \)-norm with \( p \in \mathbb{N} \). But the immediate analogs of (4.2), (4.4), and (4.5) lead to a replacement for the very last bound used in the proof of the form

\[
\frac{(5pa^n|\det M|p^n/2)mn^2}{|\det M|}.
\]

So the number of operations needed to execute BÉZOUT in line 4 now depends exponentially on the input length \( \log p \). This has not taken into account, however, the possibility of a nontrivial lower bound for the difference between large consecutive integers which are sums of \( n \) perfect \( p^{th} \) powers. Such a bound would allow for a larger interval, \((a\lambda, a'\lambda)\), that provably contains no lattice norms. But this involves number theoretic considerations beyond the scope of our work.

These arguments are all in effort to preserve the gap. When a small amount of inflation is allowed, the situation clarifies. To solve a short vector problem with gap \( \alpha \) using either GDA or SAP with gap \( \alpha' < \alpha \), the inequality (4.5) becomes

\[
\frac{1 + n^2|M_{i_0,j_0}/c \det M|}{1 - n^2|M_{i_0,j_0}/c \det M|} \leq \frac{\alpha}{\alpha'}.
\]
We still need $M_{i_0,j_0} \det M$ to divide $c$ for the purpose of Lemma 4.2. Given these two constraints, we see that with

$$c \leftarrow M_{i_0,j_0} \det M \left\lceil \frac{(\alpha + \alpha')n^2}{(\alpha - \alpha') \det M^2} \right\rceil$$

in line 2 of Algorithm 3, there is no need to insist that $\alpha$ is rational or impose restrictions on $p \in [1, \infty]$ defining the norm.

Finally, we note that the reduction to SAP adapts to the inhomogeneous forms of these problems while the reduction to GDA does not. If $y' \in \mathbb{Z}^n$ (or $y' \in \mathbb{R}^n$ if we do not intend to preserve the gap as discussed above), then the reduction can end by solving the simultaneous approximation problem of finding $q_0 \in \mathbb{Z}$ with

$$\|q_0x - y\| \leq \alpha \min_{q \in \mathbb{Z}} \|qx - y\|,$$

where $y = (M_{j_0}^T M)^{-1} y'$ (using the matrix from (4.1)). But unless we know that the first coordinate of $y$ is an integer, there is no clear modification to Proposition 2.1 that will allow GDA to be applied.

References

[1] Manindra Agrawal. Simultaneous Diophantine approximation and short lattice vectors. https://www.youtube.com/watch?v=7SGCXbim6Ug, 2019. Accessed: 2019-12-01.
[2] Frederik Armknecht, Carsten Elsner, and Martin Schmidt. Using the inhomogeneous simultaneous approximation problem for cryptographic design. In International Conference on Cryptology in Africa, pages 242–259. Springer, 2011.
[3] Sanjeev Arora, László Babai, Jacques Stern, and Z Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. Journal of Computer and System Sciences, 54(2):317–331, 1997.
[4] Wang Baocang and Hu Yupu. Public key cryptosystem based on two cryptographic assumptions. IEE Proceedings-Communications, 152(6):861–865, 2005.
[5] Erwin H. Bareiss. Sylvester’s identity and multistep integer-preserving Gaussian elimination. Mathematics of Computation, 22(103):565–578, 1968.
[6] Arne Johan Breitjes. Multi-dimensional continued fraction algorithms. MC Tracts, 1981.
[7] Wenbin Chen and Jiangtao Meng. An improved lower bound for approximating Shortest Integer Relation in $\ell_\infty$-norm ($SIR_\infty$). Information Processing Letters, 101(4):174–179, 2007.
[8] Gabriel Cramer. Introduction à l’analyse des lignes courbes algébriques. Chez les Frères Cramer & Cl. Philibert, 1750.
[9] Irit Dinur. Approximating $SVP_\infty$ to within almost-polynomial factors is NP-hard. Theoretical Computer Science, 285(1):55–71, 2002.
[10] Léo Ducas. Advances on quantum cryptanalysis of ideal lattices. Nieuw Archief voor Wiskunde, 5:184–189, 2017.
[11] András Frank and Éva Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. Combinatorica, 7(1):49–65, 1987.
[12] Steven D. Galbraith. Mathematics of public key cryptography. Cambridge University Press, 2012.
[13] Jacques Hadamard. Résolution d’une question relative aux determinants. Bulletin des Sciences Mathématiques, 2:240–246, 1893.
[14] H Inoue, Sh Kamada, and K Naito. Simultaneous approximation problems of \( p \)-adic numbers and \( p \)-adic knapsack cryptosystems-alice in \( p \)-adic numberland. *P-Adic Numbers, Ultrametric Analysis, and Applications*, 8(4):312–324, 2016.

[15] R. Kumar and D. Sivakumar. Complexity of SVP—a readers digest. *SIGACT News*, 32(3):40–52, 2001.

[16] Jeffrey C Lagarias. Knapsack public key cryptosystems and Diophantine approximation. In *Advances in cryptology*, pages 3–23. Springer, 1984.

[17] Jeffrey C. Lagarias. The computational complexity of simultaneous Diophantine approximation problems. *SIAM Journal on Computing*, 14(1):196–209, 1985.

[18] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671. Springer Science & Business Media, 2012.

[19] Hermann Minkowski. *Geometrie der zahlen*, volume 40. Chelsea Publishing, 1910.

[20] Phong Q. Nguyen. Lattice reduction algorithms: Theory and practice. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 2–6. Springer, 2011.

[21] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.

[22] Carsten Rössner and Jean-Pierre Seifert. Approximating good simultaneous Diophantine approximations is almost NP-hard. In *International Symposium on Mathematical Foundations of Computer Science*, pages 494–505. Springer, 1996.

[23] Carsten Rössner and Jean-Pierre Seifert. On the hardness of approximating shortest integer relations among rational numbers. *Theoretical Computer Science*, 209(1-2):287–297, 1998.

[24] Adi Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 145–152. IEEE, 1982.