# Faster Sieving Algorithm for Approximate SVP with Constant Approximation Factors

Divesh Aggarwal[1], Bogdan Ursu[1], and Serge Vaudenay[1]

EPFL
divesh@comp.nus.edu.sg bogdan.ursu@kit.edu serge.vaudenay@epfl.ch

**Abstract.** There is a large gap between theory and practice in the complexities of sieving algorithms for solving the shortest vector problem in an arbitrary Euclidean lattice. In this paper, we work towards reducing this gap, providing theoretical refinements of the time and space complexity bounds in the context of the approximate shortest vector problem. This is achieved by relaxing the requirements on the AKS algorithm, rather than on the ListSieve, resulting in exponentially smaller bounds starting from $\mu \approx 2$, for constant values of $\mu$. We also explain why these improvements carry over to also give the fastest quantum algorithms for the approximate shortest vector problem.

## 1 Introduction

A lattice $\mathcal{L}$ is defined as the set of all integer combinations of some linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$. The matrix $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called a basis of $\mathcal{L}$, and we write $\mathcal{L}(\mathbf{B})$ for the lattice generated by $\mathbf{B}$.

Starting in the '80s, the use of approximate and exact solvers for SVP (and other lattice problems) gained prominence for their applications in algorithmic number theory [LLL82], coding over Gaussian channels [dB89], cryptanalysis [Sha84, Bri85, LO85], combinatorial optimization and integer programming [Len83, Kan87, FT87]. Starting with the breakthrough result of Ajtai [Ajt96], lattices began to be used in constructive cryptography. Ajtai showed that lattice problems have a very desirable property for cryptography: a worst case to average case reduction. This property yields one-way functions and collision resistant hash functions, based on the *worst case* hardness of lattice problems. This is in a stark contrast to the traditional number theoretic constructions which are based on the average-case hardness e.g., factoring, discrete logarithms. Many powerful cryptographic primitives, such as fully homomorphic encryption [Gen09, BV11, BV14], now have their security based on the *worst-case* hardness of approximating the decision version of SVP (and other lattice problems) to within polynomial factors [Ajt04, MR07, Reg09, BLP$^+$13].

Perhaps the most central computational problem on lattices is the Shortest Vector Problem (SVP). Given a basis for a lattice $\mathcal{L} \subseteq \mathbb{R}^n$, the SVP is to compute a non-zero vector in $\mathcal{L}$ of minimum Euclidean norm. We call this length $\lambda_1(\mathcal{L})$. For the purpose of breaking cryptosystems, we are more interested in solving ApproxSVP$_\mu$, where the goal is to find a lattice vector of length at most $\mu \cdot \lambda_1(\mathcal{L})$.

From the computational complexity perspective, much is known about SVP in both its exact and approximate versions. On the hardness side, SVP was shown to be NP-hard to approximate within any constant factor (under randomized reductions) and hard to approximate to within $n^{c/\log\log n}$ for some constant $c > 0$ under reasonable complexity assumptions [Mic01, Kho05, HR12].

Thus, we do not expect a polynomial time algorithm for solving exact SVP. The fastest known algorithm for exact SVP runs in time $2^{n+o(n)}$ and is due to [ADRS15].

The type of algorithms that have seen the most progress in the literature and is also the topic of this work is the so called sieving algorithms. AKS [AKS01] devised a method based on "randomized sieving", whereby exponentially many randomly generated lattice vectors are iteratively combined to create shorter and shorter vectors, to give the first $2^{O(n)}$-time (and space) randomized algorithm for SVP. Many extensions and improvements of their sieving technique have been proposed, both provable [AKS02, MV10, PS09] and heuristic [NV08, WLTB11, ZPH14, BGJ14, Laa14], where the fastest provable sieving algorithm [PS09] for exact SVP requires $2^{2.465n+o(n)}$ time. The gap between the complexity of the fastest provable algorithm and that of the fastest heuristic algorithm that works well in practice is huge, and it is desirable to understand this gap better, and attempt to close the gap.

For the purpose of cryptanalysis of cryptosystems based on ApproxSVP, it is reasonable to ask whether one can come up with faster algorithms if we relax the goal to output only an approximation of the shortest vector for a small enough approximation factor $\mu$.

In particular, it was shown in [LWXZ11, WLW15] that one can modify the ListSieve algorithm from [MV10, PS09] to obtain a faster algorithm for approximate SVP. Additionally, there is a $2^{n/2+o(n)}$ time and space algorithm [ADRS15] for GapSVP (which is a decision variant of approx SVP)

In this work we show that if we apply a similar modification as [LWXZ11, WLW15] to [AKS01], then we obtain asymptotically faster classical and quantum algorithms for small enough approximation factors. This is particularly surprising since the exact SVP algorithm based on ListSieve in [PS09] is significantly faster than the exact algorithm in [AKS01].

*Organization of the paper.* In Section 2, we give the mathematical preliminaries, in Section 3, we recall the AKS algorithm for solving exact SVP, and then in Section 4, we show our main result, i.e., an algorithm for solving approximate SVP. In Section 5, we conclude with some open questions.

## 2 Background and Notation

**Lemma 1 (A corollary of the Chernoff Bound).** *Let $X_1, ..., X_n$ be independent random Bernoulli variables which take values in $\{0, 1\}$. We define*

$X \stackrel{\text{def}}{=} \sum_{i=1}^{n} X_i$ *and* $0 < \delta < 1$. *Then*

$$P(X \geq (1-\delta)E[X]) \geq 1 - \frac{1}{e^{\delta^2 E[X]/2}} .$$

*Hyperspheres and Hyperspherical caps.* By $B_n(\mathbf{x}, R)$ we denote the $n$-dimensional hypersphere of radius $R$ centered in $\mathbf{x}$, i.e.

$$B_n(\mathbf{x}, R) = \{\mathbf{y} \in \mathbb{R}^n \ : \ \|\mathbf{x} - \mathbf{y}\| \leq R\} .$$

Let $B_n(R)$ be the shorthand notation for $B_n(\mathbf{0}, R)$.

From now on, we denote by $\theta(\mathbf{x}, \mathbf{y})$ the common angle between the vectors $\mathbf{x}$ and $\mathbf{y}$, which is always between 0 and $\pi$. For any $R, h, \mathbf{x}$ such that $\|\mathbf{x}\| = R$, we define a hyperspherical cap $S_n(\mathbf{x}, h, R)$ of dimension $n$ and height $h$ at $\mathbf{x}$ as:

$$S_n(\mathbf{x}, h, R) \stackrel{\text{def}}{=} \Big\{\mathbf{y} \in B_n(R) : \|\mathbf{y}\| = R \text{ and } \theta(\mathbf{x}, \mathbf{y}) \leq \arccos\Big(1 - \frac{h}{R}\Big)\Big\}.$$

The next lemma provides an estimate on the relative volume of a hyperspherical cap with respect to the volume of the corresponding hypersphere.

**Lemma 2.** *[BDGL, Lemma 2.1][MV10, see also Lemma 4.1]: Let* $\mathbf{x}$ *be a unit vector and consider the hyperspherical cap* $S_n(\mathbf{x}, h, R)$ *of height* $h$ *of a hyperball of radius* $R$. *Then the ratio of its volume with respect to the volume of the* $B_n(R)$ *hyperball satisfies:*

$$\frac{\text{vol}(S_n(\mathbf{x}, h, R))}{\text{vol}(B_n(R))} = \text{poly(n)}\Big(1 - \Big(1 - \frac{h}{R}\Big)^2\Big)^{\frac{n}{2}}.$$

*Lattices.* A rank $d$ lattice $\mathcal{L} \subset \mathbb{R}^n$ is the set of all integer linear combinations of $d$ linearly independent vectors $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$. $\mathbf{B}$ is called a basis of the lattice and $n$ is the dimension of the lattice. Formally, a lattice is represented by a basis $\mathbf{B}$ for computational purposes, though for simplicity we often do not make this explicit. If $n = d$, we say that the lattice has full rank, and we assume this for the rest of the paper as results for full-rank lattices naturally imply results for arbitrary lattices.

Given a basis, $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, we write $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ to denote the lattice with basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$. The length of a shortest non-zero vector in the lattice is written $\lambda_1(\mathcal{L})$. The fundamental parallelepiped $\mathcal{P}(\mathbf{B})$ of the lattice is defined as the set of all vectors that can be written as $\sum_{i=1}^{n} \alpha_i \mathbf{b}_i$ where $0 \leq \alpha_i < 1$ for $1 \leq i \leq n$. For any vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} \mod \mathcal{P}(\mathbf{B})$ denotes the unique vector $\mathbf{y}$ in $\mathcal{P}(\mathbf{B})$ such that $\mathbf{x} - \mathbf{y}$ is a lattice vector. There is an efficient algorithm to compute $\mathbf{y} = \mathbf{x} \mod \mathcal{P}(\mathbf{B})$ given $\mathbf{x}$ and $\mathbf{B}$ since we can express $\mathbf{x}$ as $\sum_{i=1}^{n} \beta_i \mathbf{b}_i$ for some $\beta_i \in \mathbb{R}$, and then $\mathbf{y} = \sum_{i=1}^{n} (\beta_i - \lfloor \beta_i \rfloor)\mathbf{b}_i$.

We next define an LLL-reduced basis [LLL82].

**Definition 3.** *Given a basis* $\mathbf{B} = [\mathbf{b}_1 \ \mathbf{b}_2 \ \ldots \ \mathbf{b}_n]$, *the Gram-Schmidt orthogonalization of* $\mathbf{B}$ *is defined by* $\widetilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \widetilde{\mathbf{b}}_j$, *where* $\mu_{ij} = \frac{\langle \mathbf{b}_i, \widetilde{\mathbf{b}}_j \rangle}{\langle \widetilde{\mathbf{b}}_j, \widetilde{\mathbf{b}}_j \rangle}$.

Note that the Gram-Schmidt orthogonal basis satisfies $\langle \widetilde{\mathbf{b}}_i, \widetilde{\mathbf{b}}_j \rangle = 0$, for all $i \neq j$.

**Definition 4.** *A basis* $\mathbf{B} = [\mathbf{b}_1 \ \dots \ \mathbf{b}_n\}$ *with Gram-Schmidt orthogonal basis* $[\widetilde{\mathbf{b}}_1 \ \dots \ \widetilde{\mathbf{b}}_n]$ *is LLL reduced if for all* $1 \leq i < n$, $\|\widetilde{\mathbf{b}}_i\|^2 \leq 2\|\widetilde{\mathbf{b}}_{i+1}\|^2$ *and for all* $1 \leq j < i \leq n$, $\mu_{ij} = \frac{\langle \mathbf{b}_i, \widetilde{\mathbf{b}}_j \rangle}{\langle \widetilde{\mathbf{b}}_j, \widetilde{\mathbf{b}}_j \rangle} \leq \frac{1}{2}$.

The LLL reduced basis property can be interpreted as "$\widetilde{\mathbf{b}}_{i+1}$ is not much shorter than $\widetilde{\mathbf{b}}_i$". It is possible to LLL reduce any basis in polynomial time, and in particular, this implies that $\widetilde{\mathbf{b}}_1 = \mathbf{b}_1$ is a lattice vector of length at most $2^{(n-1)/2} \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$ [LLL82].

Now, we define the main computational problems that we study in this paper.

**Definition 5 (Shortest vector problem - SVP).** *Given a basis* $\mathbf{b_1}, ..., \mathbf{b_n}$ *of a lattice* $\mathcal{L}$*, find a shortest non-zero vector of* $\mathcal{L}$*.*

**Definition 6 ($\mu$-Approximate SVP - $\mu$SVP).** *Given a basis* $\mathbf{b_1}, ..., \mathbf{b_n}$ *of a lattice* $\mathcal{L}$*, find a non-zero lattice vector of length at most* $\mu\lambda_1(\mathcal{L})$*.*

The following result shows that in order to solve $\mu$SVP it is sufficient to find an algorithm that solves $\mu$SVP assuming the knowledge of a very good approximation of the length of the shortest vector.

**Lemma 7.** *[Reg04, HPS11] For any* $\mu \geq 1$*, let* $\mathcal{A}$ *be an algorithm that given as input a basis* $\mathbf{B}$ *of an* $n$*-dimensional lattice and a hint* $\lambda$ *such that* $\lambda_1(\mathcal{L}(\mathbf{B})) \leq \lambda \leq (1 + \frac{1}{n})\lambda_1(\mathcal{L}(\mathbf{B}))$ *computes a non-zero vector in* $\mathcal{L}(\mathbf{B})$ *of length at most* $\mu \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$ *and runs in time at most* $T(n)$*. Then there is an efficient algorithm* $\mathcal{A}'$ *which uses* $\mathcal{A}$ *as a subroutine and solves* $\mu$*SVP in time* $T(n) \cdot n^2$*, where* $n$ *is the dimension of the lattice.*

The algorithm $\mathcal{A}'$ first runs the LLL algorithm to obtain $\nu$ such that

$$\lambda_1(\mathcal{L}(\mathbf{B})) \leq \nu \leq 2^{(n-1)/2} \cdot \lambda_1(\mathcal{L}(\mathbf{B}) \ .$$

Then, it guesses a value $i$ in the set $\{0, ..., \lceil \frac{n-1}{2} \log_{(1+\frac{1}{n})} 2 \rceil\}$ and calls $\mathcal{A}$ with $\lambda = \lambda(i) = \nu(1 + \frac{1}{n})^{-i}$. As there are at most $n^2$ possible values for $i$, the running time of the algorithm is upper bounded by $T(n) \cdot n^2$ and the success of the algorithm follows from the fact that for at least one value of $i$, we have that $\lambda_1(\mathcal{L}(\mathbf{B})) \leq \lambda(i) \leq (1 + \frac{1}{n})\lambda_1(\mathcal{L}(\mathbf{B}))$. For a more precise explanation, we direct the reader to [Reg04, HPS11].

The following result shows that the basis vectors of an LLL-reduced basis that are significantly larger than the shortest vector do not contribute to the shortest vector.

**Lemma 8.** *[NV08, Lemma 3.3] Let* $(\mathbf{b_1}, ..., \mathbf{b_n})$ *be an LLL-reduced basis of a lattice* $\mathcal{L}$*. If* $\mathbf{s}$ *is a shortest vector of* $\mathcal{L}$*, then there exists an index* $i \in \{1, .., n\}$ *such that* $\mathbf{s}$ *belongs to the lattice spanned by* $(\mathbf{b_1}, ..., \mathbf{b_i})$ *and* $\|\mathbf{b_j}\| \leq 2^{3n}\lambda_1(\mathcal{L})$ *for every* $j = \overline{1, i}$*.*

Therefore, upon applying the LLL algorithm it is possible to safely remove all the basis vectors of norm bigger than $2^{3n}\lambda_1(\mathcal{L})$ without losing the shortest vector in the lattice. Thus, from now on we assume without loss of generality that the input of the algorithm is an LLL-reduced basis of a lattice $\mathcal{L}$ with all basis vectors of norm less than $2^{3n}\lambda_1(\mathcal{L})$. Additionally, we assume that we are given $\lambda$ such that $\lambda_1(\mathcal{L}) \leq \lambda \leq (1+1/n)\lambda_1(\mathcal{L})$.

We will also need the following result that bounds the number of vectors in a ball of radius $R$ that are pairwise at least $\gamma R$ distance apart from each other for some $\gamma \in (0,1)$. The result is obtained using bounds from [KL78a].

**Lemma 9.** *[HPS11, Lemma 7.1] Let $S \subseteq B_n(R)$ such that the distance between any two vectors of $S$ is at least $\gamma R$, with $0 < \gamma < 1$. Then $|S| \leq N_T \stackrel{\text{def}}{=} 2^{c_t n + o(n)}$, where:*

$$c_t = -\log_2(\gamma) + 0.401.$$

## 3 The AKS Sieving Algorithm

The first provable lattice sieving algorithm has been proposed by [AKS01], and has been subsequently improved in a number of papers [MV10, NV08, HPS11]. Since our $\mu$SVP algorithm is obtained by a modification of the AKS sieving algorithm [AKS01], we describe the AKS sieving algorithm here for completeness. Our presentation closely follows the exposition in [HPS11].

The main idea of the algorithm is to randomly sample a large enough set of lattice vectors in a ball of radius $R$ such that there are many pairs of vectors in this set which are at most $\gamma R$ apart. This results in obtaining many lattice vectors of length at most $\gamma R$, and as long as we have enough vectors to start with, we should be able to recursively get shorter vectors until we get a shortest vector with good probability. However, one of the main difficulties in this approach lies in understanding the distribution of lattice vectors. Instead of trying to understand this distribution precisely, the algorithm "blurs" the perspective of the lattice by adding to each lattice vector $\mathbf{v}$ a small real perturbation $\mathbf{x}$, which will mean that depending on the size of $\mathbf{x}$, the result $\mathbf{v}+\mathbf{x}$ could have been obtained from other lattice vectors close to $\mathbf{v}$ as well. The algorithm keeps pairs of lattice vectors and their perturbed variants and sieves only based on information from the latter. Then, once it finishes sieving, it uses the former in order to recover the shortest vector.

### Sampling

Rather than first sampling lattice vectors and then adding the perturbation, the sampling procedure generates pairs by first choosing a perturbation $\mathbf{x}$ of small norm. The description of the sampling algorithm as used in [AKS01, NV08] is given in Algorithm 1.

**Algorithm 1** [AKS01]:Sample

**Input:** $B = \{\mathbf{b_1}, ..., \mathbf{b_n}\}$ and perturbation $\mathbf{x}$
**Output:** A lattice vector $\mathbf{v}$ and its perturbed variant $\mathbf{v} + \mathbf{x}$.
1: Compute $\mathbf{v} = -\mathbf{x} + (\mathbf{x} \mod P(B))$.
2: **return** pair $(\mathbf{v}, \mathbf{v} + \mathbf{x})$.

### Sieving

The sieving procedure receives as input a list of pairs in $(\mathbf{v}, \mathbf{v} + \mathbf{x}) \in \mathcal{L} \cap B_n(R)$. Then, it performs reductions so that for the output pairs, the perturbed vector is in a ball of a smaller radius $\gamma R + \epsilon$, where $\gamma$ is a positive sub-unitary real and $\epsilon$ is a parameter which is linear in $\lambda_1(\mathcal{L})$ and is an upper bound on the magnitude of perturbations. During the process, a certain number of pairs will not be reducible but will be used instead for the reduction of other pairs. These pairs are called centers.

Before the start of the procedure, none of the pairs are designated as centers. The initial list is processed sequentially: if a pair $(\mathbf{v}, \mathbf{v} + \mathbf{x})$ is geometrically close to a pair $(\mathbf{c}, \mathbf{c} + \mathbf{x}')$, that has been labelled as a center, the pair is reduced by computing two vector differences and outputting the pair $(\mathbf{v} - \mathbf{c}, (\mathbf{v} + \mathbf{x}) - \mathbf{c})$. Otherwise, if $(\mathbf{v}, \mathbf{v} + \mathbf{x})$ is not close to any center, it will be designated as a center itself. Geometrically close here means that the difference between the second component of the two pairs is smaller than $\gamma R$.

Notice that the sieving algorithm preserves the initial perturbations. This is a technical detail needed for proving that we output a shortest non-zero lattice vector with high probability. Also, the sieving routine only considers the second component of each pair, thus it only concerns itself with the perturbed lattice vectors and not the lattice vectors themselves. The formal description is given in Algorithm 2.

**Algorithm 2** [AKS01]:Sieving

**Input:** $R$, $0 < \gamma < 1$ and a list *List* of vector pairs $(\mathbf{v}, \mathbf{v} + \mathbf{x})$ in $\mathcal{L} \times B_n(R)$.
**Output:** Another list $List'$ of vector pairs
1: $Centers \leftarrow \emptyset$.
2: $List' \leftarrow \emptyset$.
3: **for** $(\mathbf{v}, \mathbf{v} + \mathbf{x}) \in List$ **do**
4: **if** there is no pair $(\mathbf{v}', \mathbf{v}' + \mathbf{x}') \in Centers$ such that $\|\mathbf{v}' + \mathbf{x}' - (\mathbf{v} + \mathbf{x})\| \leq \gamma R$ **then**:
5: Add $(\mathbf{v}, \mathbf{v} + \mathbf{x})$ to $Centers$.
6: **else** add $(\mathbf{v} - \mathbf{v}', (\mathbf{v} + \mathbf{x}) - \mathbf{v}')$ to $List'$.
7: **end if**
8: **end for**
9: **return** $List'$.

### The Main Algorithm

The algorithm is given in Algorithm 3.

**Algorithm 3** [AKS01]:AKS

**Input:** Integer Lattice Basis $B = \{\mathbf{b_1}, ..., \mathbf{b_n}\}$, $\epsilon_0, \gamma$, $N$ and hint $\lambda$.
**Output:** A shortest non-zero vector of $B$.

$\epsilon \leftarrow \epsilon_0 \times \lambda$.
$List_0 \leftarrow \emptyset$.
**for** $i = 1$ to $N$ **do**
    Draw $\mathbf{x}$ uniformly at random from $B_n(\epsilon)$.
    Add Algorithm $1(B, \mathbf{x})$ to $List_0$.
**end for**
$R \leftarrow n \times \max_i \|\mathbf{b_i}\|$.
$j \leftarrow 1$.
**while** $R > (1 + \frac{1}{n})\frac{\epsilon}{1-\gamma}$ **do**
    $List_j \leftarrow$ *Algorithm* $2(List_{j-1}, R, \gamma)$.
    $R \leftarrow \gamma R + \epsilon$.
    $j \leftarrow j + 1$
**end while**
Consider all pairs $(\mathbf{v}, \mathbf{v} + \mathbf{x_1}), (\mathbf{w}, \mathbf{w} + \mathbf{x_2})$ in $List_k$ and return minimal non-zero difference $\mathbf{v} - \mathbf{w}$.

### Correctness of the AKS Algorithm

**Theorem 10.** *[HPS11] The AKS algorithm succeeds in finding a shortest non-zero vector in the lattice $\mathcal{L}(\mathbf{B})$ in time $2^{3.397n}$ with probability at least $\frac{1}{2} - \frac{1}{2^{\Theta(n)}}$, when run with $N = 2^{1.984n}$, with $\gamma = 0.496$ and $\epsilon_0 = 0.676$.*

For brevity, we will not detail the proof of correctness for the AKS algorithm. Nevertheless, the main outline and ideas of this proof will appear in our proof for the $\mu$AKS algorithm from the following section.

## 4 Adapting the AKS for Solving $\mu$SVP

In this section we describe a natural adaptation of the AKS sieve for finding approximations of the non-zero shortest vector. This course of action has first been explored for modifications of ListSieve, as proposed in [LWXZ11] and [WLW15]. The ListSieve is another sieving algorithm which has lower time and space complexity compared to the AKS, and its fastest variant uses the birthday paradox and has been introduced in [PS09] and [HPS11]. Nevertheless, we show that when the goal is to recover only an approximation of the shortest vector up to a factor $\mu$, adapting the AKS is more natural and results in an algorithm that has both lower time and lower space complexity than adaptations of the ListSieve. Current findings suggest that designing a provable sieving algorithm

for finding an approximation of the shortest vector results in time and space complexities only limited by the known bounds [KL78a] on the kissing number. In particular, an improvement in the result of Lemma 9 will immediately imply an improvement in all sieving based algorithms including ours.

What changes with respect to the original outline of the AKS is the introduction of a different sieving procedure based on the description of the AKS-Birthday from [HPS11]. The idea of this modification is to preselect the pairs that are to be used as centers right after the sampling phase, and to keep two separate lists of vectors, as in the ListSieve-Birthday algorithm. One list will only be used for reductions, while the other will provide a sufficiently large set of reduced pairs at the end of the sieving steps, so that using Lemma 13 we obtain at least one good pair. The new idea here is that, by separating the pairs used for reductions from the pairs among which we search for the approximate shortest vector, it is not necessary anymore to require that the pairs used as centers contain good perturbations. This idea could also be used to improve the AKS-Birthday algorithm for exact SVP described in [HPS11], but unfortunately the resulting improved algorithm does not have a better time complexity than the ListSieve-Birthday. We proceed by giving a detailed description of the modified sieving procedure used first in AKS-Birthday, followed by a description of the modified AKS, its proof of correctness and a comparison with previous results.

### Modified Sieving Procedure

In Algorithm 4, we give a full description of the sieving procedure described in [HPS11] for the AKS-Birthday algorithm. In particular, throughout our algorithm we maintain two lists of vectors instead of just one. List $C$ will provide the centers which will help reduce another list $S$, along with the rest of list $C$.

For each call to the sieving procedure, we first preselect $N_C$ pairs (the number $N_C$ will be determined later) from a list $C$ and we add them to a new set, call it Centers. For each $(\mathbf{v}, \mathbf{v}+\mathbf{x}) \in (C \cup S) \setminus \text{Centers}$, we attempt to reduce the pair by finding a center pair $(\mathbf{c}, \mathbf{c}+\mathbf{x}')$ such that the difference $(\mathbf{v}+\mathbf{x}) - (\mathbf{c}+\mathbf{x}')$ is smaller than $\gamma R$. We keep these small differences for the next iteration of the sieve.

### The Main Algorithm

For recovering the shortest vector in the lattice up to an approximation factor $\mu$, we use the outline of the original AKS algorithm and we modify it in Algorithm 5 to allow for the incorporation of the modified sieving procedure explained in the previous paragraph. The algorithm starts with a great number of pairs $(\mathbf{v}, \mathbf{v}+\mathbf{x}) \in \mathcal{L} \cap B_n(R)$, where the second component is bounded by an initial radius $R$. Then, it reduces the norm of all elements repeatedly until it is left with many pairs in the hyperball of small radius $R_0$, which will have to be close to $\mu$. At each step $j$, the set $C_j$ will provide the centers which will be used to reduce both itself and the set $S_j$, obtaining sets $C_{j+1}$ and $S_{j+1}$. At the end of all the

**Algorithm 4** (Adapted from [HPS11]):Sieving

**Input:** Lists $C$, $S$, $Centers$ included in $\mathcal{L} \times B_n(R)$, along with $R$, $0 < \gamma < 1$.
**Output:** Lists $C'$ and $S'$ of vector pairs

```
C' = S' = ∅.
for (v, v + x) ∈ C do
    if there is a pair (v', v' + x') ∈ Centers such that ‖v' + x' − (v + x)‖ ≤ γR then
        Add (v − v', (v + x) − v') to C'.
    end if
end for
for (v, v + x) ∈ S do
    if there exists (v', v' + x') ∈ Centers such that ‖v' + x' − (v + x)‖ ≤ γR then
        Add (v − v', (v + x) − v') to S'.
    end if
end for
return C', S'.
```

runs of the sieving procedure, we only look at vectors in the last set $S_{j-1}$ and output the shortest non-zero one.

### 4.1 Correctness of the $\mu$AKS

As mentioned before, the proof of correctness for the $\mu$AKS algorithm follows the same outline as the proof given in [HPS11], with the exception that in this case we analyse how to find approximations of the shortest non-zero vector rather than the exact solution.

**Lemma 11.** *Define* $R_{\text{end}} \stackrel{\text{def}}{=} \frac{\epsilon}{1-\gamma}$. *When run with* $\epsilon_0 > 0$, $0 < \gamma < 1$, $\lambda_1(\mathcal{L}(\mathbf{B})) \leq \lambda \leq \lambda_1(\mathcal{L}(\mathbf{B}))(1 + \frac{1}{n})$ *and* $R_0 \geq R_{\text{end}}(1 + \frac{1}{n})$, *the $\mu$AKS algorithm reaches radius $R_0$ after $O(n)$ radius reductions.*

*Proof.* Denote the number of sieving steps by a variable $k$, which will be upper-bounded below. Let $R$ denote the initial radius from which the AKS algorithm starts sieving. We will prove that we can reach radius $R_{\text{end}}(1 + \frac{1}{n})$ after $O(n)$ runs of the sieve, which will imply the same for $R_0$. In fact, when radius $R_{\text{end}}$ is reached, it is not possible to reduce the radius anymore. Nonetheless, $R_{\text{end}}$ is achieved only as the number of sieving executions reaches infinity, since after $k$ iterations the reached radius is $\gamma^k R + \frac{1-\gamma^k}{1-\gamma}\epsilon = R_{\text{end}} + \gamma^k(R - R_{\text{end}})$. This quantity is equal to $R_{\text{end}}$ only when $k \to \infty$. Instead, consider the number of steps needed to obtain radius $R_{\text{end}}(1 + \frac{1}{n})$ and ask that $R_{\text{end}} + \gamma^k(R - R_{\text{end}}) = (1 + \frac{1}{n})R_{\text{end}}$. Then $k = \lceil \log_\gamma(\frac{R_{\text{end}}}{n(R-R_{\text{end}})}) \rceil$.

Parameter $\epsilon_0$ is not dependent on $n$, which along with the fact that $\lambda_1(\mathcal{L}) \leq \lambda \leq (1 + \frac{1}{n})\lambda_1(\mathcal{L})$ implies that $\epsilon = \epsilon_0\lambda$ is in $\Theta(\lambda_1(\mathcal{L}))$. Since $\gamma$ is not dependent on $n$, this means that $R_{\text{end}} = \Theta(\lambda_1(\mathcal{L}))$ as well. Now, it is known that using Lemma 8 it is guaranteed to obtain $R = 2^{O(n)}\lambda_1(\mathcal{L})$, which means that $k = \log_{1/\gamma}(n2^{O(n)}) = O(n)$. □

**Algorithm 5** $\mu$AKS

**Input:** Integer Lattice Basis $B = \{\mathbf{b_1}, ..., \mathbf{b_n}\}$, $\epsilon_0, 0 < \gamma < 1$, $N_1$, $N_2$, $\mu$, $R_0$ and $\lambda$.
**Output:** A $\mu-$approximation of the shortest non-zero vector of $\mathcal{L}(B)$.

$C_0 = S_0 = \emptyset$.
$\epsilon \leftarrow \epsilon_0 \times \lambda$
**for** $i = 1$ to $N_1$ **do**
    Draw $\mathbf{x}$ uniformly at random from $B_n(\epsilon)$.
    Add Algorithm 1$(B, \mathbf{x})$ to $C_0$.
**end for**
**for** $i = 1$ to $N_2$ **do**
    Draw $\mathbf{x}$ uniformly at random from $B_n(\epsilon)$.
    Add Algorithm 1$(B, \mathbf{x})$ to $S_0$.
**end for**
$R \leftarrow n \times \max_i \|\mathbf{b_i}\| + \epsilon$.
$j \leftarrow 1$.
**while** $R > R_0$ **do**
    $Centers \leftarrow \emptyset$.
    Add first $N_C$ pairs of $C_{j-1}$ to $Centers$.
    $(C_j, S_j) \leftarrow$Algorithm 4$(C_{j-1} \setminus Centers, S_{j-1}, Centers, R, \gamma)$.
    $R \leftarrow \gamma R + \epsilon$.
    $j \leftarrow j + 1$.
**end while**
Consider all $(\mathbf{v}, \mathbf{v} + \mathbf{x}) \in S_{j-1}$ and output shortest non-zero $\mathbf{v}$.

We now return to the description of the proof of correctness. Some explanations regarding the last steps of the algorithm are also necessary. Let $r_0$ be the last radius for the corresponding lattice vectors obtained. If the perturbed vectors output at the end of the algorithm reside in $B_n(R_0)$, it follows that $r_0 = R_0 + \epsilon$. Once radius $R_0$ has been reached, the goal is to obtain sufficient lattice vectors so as to ensure that with high probability at least one lattice vector is a non-zero vector.

At this point, it is certainly possible to choose the initial $N_1 = n^2 N_T$ (where $N_T$ is as defined in Lemma 9), and $N_2 = 1$ to ensure that after all the sieving steps at least one lattice vector in $B_n(r_0)$ is obtained, but there are no guarantees that this lattice vector is non-zero. To formally prove that this is the case, it is necessary to use some special types of vector pairs which are denoted as good. These are those pairs whose perturbation satisfies the following condition:

**Definition 12 (Good perturbations and good pairs).** *Let us define* $T_s \stackrel{\text{def}}{=} B_n(0, \epsilon) \cap B_n(-\mathbf{s}, \epsilon)$ *and* $T_{-s} \stackrel{\text{def}}{=} B_n(0, \epsilon) \cap B_n(\mathbf{s}, \epsilon)$*, where* $\mathbf{s}$ *is a shortest vector in* $L$*. Then a pair* $(\mathbf{v}, \mathbf{v} + \mathbf{x})$ *is defined to be good when* $\mathbf{x} \in (T_s \cup T_{-s}) \setminus (T_s \cap T_{-s})$.

When the perturbations are drawn uniformly at random from the $n$-dimensional ball of radius $\epsilon = \epsilon_0 \lambda$, the following lemma provides a lower bound on the probability that a perturbation and by extension a pair is good. The formulation for the case when $\epsilon_0 \leq 1$ can be found in [NV08], whilst the case $\epsilon_0 \leq \frac{1}{2}$ is not

interesting, as there would be no good perturbations. Values of $\epsilon_0 > 1$ have been used before in [LWXZ11, WLW15], what we do in addition is to motivate that this case is identical to the case of $\frac{1}{2} \leq \epsilon < 1$.

**Lemma 13.** *[NV08, Adapted from Lemma 3.4]: Consider $\epsilon_0 > \frac{1}{2}$ and $\lambda$, where $\lambda_1(\mathcal{L}) \leq \lambda \leq (1 + \frac{1}{n})\lambda_1(\mathcal{L})$. For any perturbation $\mathbf{x}$ drawn uniformly at random from $B_n(\epsilon_0\lambda)$, the probability that $\mathbf{x}$ is good is given by:*

$$P_{x \xleftarrow{\$} B_n(\epsilon_0\lambda)}(\mathbf{x} \in (T_s \cup T_{-s}) \setminus (T_s \cap T_{-s})) \geq \frac{1}{N_G},$$

$$\text{where } N_G \stackrel{\text{def}}{=} 2^{-c_g n + o(n)} \text{ and } c_g = -\tfrac{1}{2}\log_2(1 - \tfrac{1}{4\epsilon_0^2}).$$

*Proof.* First recall that from Lemma 7, the hint $\lambda$ satisfies $\lambda_1(\mathcal{L}) \leq \lambda \leq (1 + \frac{1}{n})(\lambda_1(\mathcal{L}))$. Consider the case $\lambda = \lambda_1(\mathcal{L})$. When $\epsilon_0 \leq 1$, it is sufficient to use Lemma 2 for height $h = (\epsilon_0 - \frac{1}{2})\lambda$ and radius $\epsilon = \epsilon_0\lambda$, as the two spherical caps corresponding to $T_s$ and $T_{-s}$ do not intersect, as in Figure 1. We obtain that the relative volume is:

$$\text{poly(n)}\Big(1-\Big(1-\frac{(\epsilon_0 - \frac{1}{2})\lambda_1(\mathcal{L})}{\epsilon_0\lambda_1(\mathcal{L})}\Big)^2\Big)^{\frac{n}{2}} = \text{poly(n)}\Big(1-\frac{1}{4\epsilon_0^2}\Big)^{\frac{n}{2}} = 2^{\frac{n}{2}\log_2\left(1-\frac{1}{4\epsilon_0^2}\right)+o(n)}.$$

This is the expression we wanted.

Nevertheless, for $\epsilon_0 > 1$, it is necessary to take into account the intersection which is composed of two spherical caps of height $(\epsilon_0 - 1)\lambda$, and we are left with:

$$\text{poly(n)}\Big(\Big(1 - \frac{1}{4\epsilon_0^2}\Big)^{\frac{n}{2}} - \Big(1 - \frac{1}{\epsilon_0^2}\Big)^{\frac{n}{2}}\Big).$$

Using the fact that $\log(a+b) = \log(a) + \log(1 + \frac{b}{a})$, we obtain the relative volume $2^{c'_g n + o(n)}$, where:

$$c'_g = -\frac{1}{2}\log_2\left(1 - \frac{1}{4\epsilon_0^2}\right) - \frac{1}{n}\log_2\left(1 - \left(\frac{\epsilon_0^2 - 1}{4\epsilon_0^2 - 1}\right)^{\frac{n}{2}}\right).$$

For asymptotically large n, the second term vanishes and we obtain $c_g \approx_{n\to\infty} c'_g$.

Now, let us see what happens when $\lambda_1(\mathcal{L}) \leq \lambda$. The error is at most $\frac{1}{n}$, which results in hyperspheres $B_n(\epsilon_0\lambda)$ of volumes larger than the volume of $B_n(\epsilon_0\lambda_1(\mathcal{L}))$ by a factor $(1+\frac{1}{n})^n$. Thus, this difference belongs to $2^{o(n)}$ and will be disregarded.

Because the expression $\frac{1}{n}\log_2\left(1-\left(\frac{\epsilon_0^2-1}{4\epsilon_0^2-1}\right)^{\frac{n}{2}}\right)$ is in $o(1)$ as $n$ goes to infinity, it could be put aside as well and it is possible to use the formulation for $\epsilon_0\lambda \leq \lambda_1(\mathcal{L})$ in the case when $\epsilon_0\lambda > \lambda_1(\mathcal{L})$. □

First, it is important to choose $N_C$ appropriately such that the number of lost pairs is not too large. The main difference with the original AKS will be that now we require that at the end of the sieving steps we obtain at least one good pair. We are ready now to state our main result, the proof of which is very similar to the proof for the AKS-Birthday algorithm as described in [HPS11].

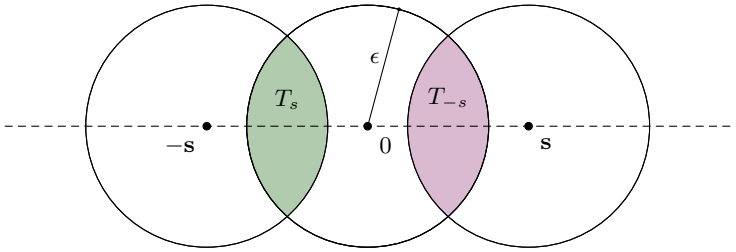


**Fig. 1.** Good perturbations in the case where $\epsilon < \lambda_1(\mathcal{L})$ [Reg04]

**Theorem 14.** *We consider the $\mu$SVP with parameters $n$ and $\mu$. We let parameters $\epsilon_0 > \frac{1}{2}$, $0 < \gamma < 1$ be such that $\mu = \frac{\epsilon_0}{1-\gamma}(1+\frac{1}{n}) + \epsilon_0$. We define $N_T$ and $N_G$ as in Lemma 9 and Lemma 13, respectively. For parameters $N_1 = n^4 N_T$, $N_2 = (\frac{n}{n-1})^2 \times n^4 \times N_G$, $N_C = n^2 N_T$, $\lambda_1(\mathcal{L}(\mathbf{B})) \leq \lambda \leq \lambda_1(\mathcal{L}(\mathbf{B}))(1+\frac{1}{n})$ and $R_0 = \frac{\epsilon_0 \lambda}{1-\gamma}(1+\frac{1}{n})$, the $\mu$AKS algorithm succeeds in finding a $\mu-$approximation of the shortest non-zero vector with probability greater than $\frac{1}{2} - \frac{1}{2^{\theta(n)}}$.*

*Proof.* First remark that according to Lemma 7, we can assume we are given $\lambda_1(\mathcal{L}(\mathbf{B})) \leq \lambda \leq \lambda_1(\mathcal{L}(\mathbf{B}))(1+\frac{1}{n})$ as an input to the algorithm, incurring only a polynomial time overhead. From *Lemma* 11, the radius for the perturbed vectors converges towards $\frac{\epsilon}{1-\gamma}$. In order to ensure that the number of sieve operations is in $O(n)$, we ask that the last radius $R_0$ is greater than $\frac{\epsilon}{1-\gamma}$ by at least $\frac{1}{n}$. The norm of the lattice vectors obtained will therefore be bounded by $R_0 + \epsilon$, which is why we ask that $\mu\lambda \geq R_0 + \epsilon$. This later condition is equivalent to $\mu \geq \frac{\epsilon_0}{1-\gamma}(1+\frac{1}{n}) + \epsilon_0$.

There is a crucial difference between the sieving procedure of the AKS and the modified sieving procedure of the $\mu$AKS. In the first case all the pairs not labelled as centers will be reduced and there will be at most $N_T$ centers. In the second case however, since the centers are preselected, there might be pairs which are at a distance greater than $\gamma R$ from all the centers, and are lost in subsequent rounds. Let us call these pairs exterior pairs. In the following, we estimate the number of exterior pairs using the approach used in [HPS11] for AKS-Birthday.

Assume for the moment that we are applying the sieving procedure of the $\mu$AKS on the $N_C$ pairs that have been set aside in line 15. We want to upper bound the probability that a pair is at a distance greater than $\gamma R$ from all these $N_C$ pairs. Let this probability be $p$. Let $p_i$ be the probability that the $i^{th}$ pair is far from all previous $(i-1)$ pairs. From Lemma 9, we know that the number of pairs far from each other is smaller than $N_T$, which means that $\sum_{i=1}^{N_C} p_i \leq N_T$. Also, it is easy to see that $p_i$ is monotonically decreasing, and $p_{N_C} \geq p$. Thus we have that $N_C \times p \leq N_T$ which implies $p \leq \frac{1}{n^2}$.

Let us return to the $\mu$AKS algorithm and the modified sieving procedure. Now we look at the pairs which we want to reduce and which will not be among the first $n^2 N_C$ pairs set aside as centers. The probability $p'$ of a pair being exterior is smaller than $p_{N_C+1}$, as there are more centers than in the scenario

mentioned in the previous paragraph. It is worth pointing out an observation from [HPS11], that all the vectors from the last list $S$ are independently and identically distributed, which would allow us to use Lemma 1 to bound the number of exterior pairs.

For simplicity, in the following we assume that the number of sieving steps (number of entries in the while loop of line 13) is exactly $n$ rather than $O(n)$, as proven in Lemma 11. The probability that a pair is not exterior after all the executions of the sieve is greater than $(1 - \frac{1}{n^2})^n$. Therefore, the probability $p''$ that a pair is exterior when sieving is completed is less than $1 - (1 - \frac{1}{n^2})^n \approx \frac{1}{n}$.

Let us consider the worst case scenario in which $p'' = \frac{1}{n}$. Consider $X_i$ to be a random variable equal to 1 if pair $i$ is not exterior and to 0 if pair $i$ is exterior. Also, let $X \stackrel{\text{def}}{=} \sum_{i=1}^{N_2} X_i$, $N_2' \stackrel{\text{def}}{=} (\frac{n}{n-1})^2 \times n^3 \times N_G$ and $N_2'' = \frac{n^3}{1-\frac{1}{n}} N_G$. In fact, $N_2' = E[X]$ and $N_2'' = (1 - \frac{1}{n})E[X]$. From Lemma 1 with $\delta = \frac{1}{n}$, it follows that:

$$P(X \geq N_2'') \geq 1 - \frac{1}{e^{\frac{N_2'}{2n^2}}} \geq 1 - \frac{1}{e^n}.$$

This means that with probability exponentially close to 1, the algorithm will obtain after all the sieving steps at least $N_2''$ pairs. In the following we are interested in how many pairs out of $N_2''$ are actually good, in the sense of Definition 12. For this, first let $Y$ be a random variable denoting the number of good pairs out of the $N_2''$ pairs obtained after sieving. Using Lemma 13 and Lemma 1 with $\delta = \frac{1}{n}$, we have that:

$$P(Y \geq n^3) \geq 1 - \frac{1}{e^{\frac{N_2''}{2N_G n^2}}} = 1 - \frac{1}{e^{\frac{n^2}{2(n-1)}}}.$$

So we obtain at least one good pair with probability exponentially close to 1. To conclude, we use the tossing argument first introduced to prove the correctness of the AKS by [AKS01].

*Tossing argument* We first show that the probability to obtain zero vectors in $B_n(r_0)$ is at most double the probability of obtaining the shortest vector by using a slight modification of the $\mu$AKS algorithm. The idea is to consider $\mu\text{AKS}_2$, an identical algorithm with the exception that in Algorithm 1 the algorithm applies only once, with probability $\frac{1}{2}$, a function $\tau$ on all perturbations $\mathbf{x} \in B_n(\mathbf{0}, \epsilon)$. The function $\tau$ is defined as:

$$\tau(\mathbf{x}) = \begin{cases} \mathbf{x} + \mathbf{s}, \text{if } \mathbf{x} \in T_s \setminus (T_s \cap T_{-s}), \text{where } \mathbf{s} \text{ denotes the shortest vector in } \mathcal{L} \\ \mathbf{x} - \mathbf{s}, \text{if } \mathbf{x} \in T_{-s} \setminus (T_s \cap T_{-s}) \\ \mathbf{x}, \text{otherwise.} \end{cases}$$

Unsurprisingly, this function looks strange and might actually be uncomputable in polynomial time. Also, it might appear that the proof employs a circular argument. In fact, it is irrelevant what is the complexity of $\tau$, since in the

following it will be shown that the probability of a lattice vector being outputted by $\mu\text{AKS}_2$ or $\mu$AKS is the same. There are several properties to consider at this point:

1. In Algorithm 1, pairs $(\mathbf{v}, \mathbf{v}+\mathbf{x})$ are generated by sampling $\mathbf{x}$ uniformly from a small hyperball. Then, the lattice vector $\mathbf{v}$ is computed as $\mathbf{v} = -\mathbf{x} + (\mathbf{x} \bmod P(B))$. Recall that when sieving, the sub-procedure makes its decisions only based on perturbed vectors. In fact, these remain the same even if function $\tau$ is applied, $\mathbf{v} + \mathbf{x} = \mathbf{v} + \tau(\mathbf{x})$ for all $\mathbf{x} \in B_n(\mathbf{0}, \epsilon)$. Therefore, the $\tau$ function has no impact on the perturbed vectors obtained after sampling.
2. Perturbations are being maintained throughout the algorithm. Moreover, as the sets $T_s$ and $T_{-s}$ have equal volumes, it follows that the $\tau$ function maintains the same distribution of perturbations as in the original $\mu$AKS, that is to say that $\tau(\mathbf{x})$ is also uniformly distributed in $B_n(\epsilon)$. This along with the fact that sieving does not use lattice vectors translates to the same distribution of the output pairs. Nevertheless, it cannot be said that the initial $\mu$AKS and the modified variant have the same control flow, this is true for the first execution of the sieve, but not for the others.

To conclude, first recall that with probability almost 1, at the end of the algorithm there is at least one pair left for which the perturbation is good. This means that in $\mu\text{AKS}_2$, with probability $\frac{1}{2}$, this good pair had an application of $\tau$ on it. Namely, what is obtained is that with probability $\frac{1}{2}$, any zero lattice vector is taken by the $\tau$ function into $\mathbf{s}$. Since $\mu\text{AKS}_2$ has the same output distributions as $\mu\text{AKS}_1$, this means that the $\mu$AKS algorithm also recovers the shortest non-zero vector $\mathbf{s}$ with probability at least half the probability of obtaining a zero vector. Otherwise, if the good pair does not correspond to the zero vector we still have an $r_0$ approximation of $\mathbf{s}$ and we are done.

□

In order to ensure a higher success probability, it is possible to run the $\mu$AKS algorithm a polynomial number of times $p(n)$, resulting in a success probability larger than a threshold which is roughly $1 - \frac{1}{2^{p(n)}}$, a probability exponentially close to 1. The complexity of the algorithm remains the same, as the polynomial number of rounds is absorbed into the $o(n)$ term in the exponent.

Our most important observation in the construction of our algorithm has been that, since the perturbations associated with centers are irrevocably lost, it is not absolutely necessary that we use only good pairs as centers. To our knowledge, while center preselection has been described before in [HPS11], the idea of not restricting ourselves to good center pairs is novel. The proof of correctness is dependent on the pre-selection of the pairs we use for reduction and unlike previous approaches to the AKS [HPS11, MV10] we process them separately from the pairs on which we apply the tossing argument.

### Complexity Analysis

Using the correctness analysis, it is sufficient to obtain just one good pair at the end of the sieving steps. Therefore, set $N_1 = n^4 N_T = 2^{c_t n + o(n)}$, $N_2 = (\frac{n}{n-1})^3 \times$

$n^3 \times N_G = 2^{c_g n+o(n)}$ and $N_C = 2^{c_t n+o(n)}$. Let us denote the space complexity by $N_{\text{space}} = 2^{\text{c}_{\text{space}} n+o(n)}$ and the time complexity by $N_{\text{time}} = 2^{\text{c}_{\text{time}} n+o(n)}$. The space complexity is $N_{\text{space}} = N_1 + N_2$, which means that $\text{c}_{\text{space}} = \max(c_t, c_g)$. The most expensive operation is the sieving procedure, which requires time $N_{\text{space}} N_C$. Therefore, we obtain $\text{c}_{\text{time}} = \text{c}_{\text{space}} + c_t$. To obtain the expression of the overall complexity we use the fact that the smallest possible radius for lattice vectors is $\epsilon(1 + \frac{1}{1-\gamma})$, which means that $\mu$ must be at least $\epsilon_0(1 + \frac{1}{1-\gamma})$. The function $\epsilon_0(1 + \frac{1}{1-\gamma})$ is increasing with respect to both its arguments and $\gamma$ and $\epsilon_0$ must also be as large as possible to have a small number of centers and a big probability of sampling good pairs. Therefore, we set $\mu = \epsilon_0(1 + \frac{1}{1-\gamma})$. Plugging the expressions from Lemma 9 and Lemma 13 we obtain the following result:

**Lemma 15.** *The $\mu$AKS algorithm with parameters as in Theorem 14 solves the approximate shortest vector problem in time complexity bounded by $2^{\text{c}_{\text{time}} n+o(n)}$ and space complexity bounded by $2^{\text{c}_{\text{space}} n+o(n)}$, where:*

$$\text{c}_{\text{time}} = \max\Big[-\frac{1}{2}\log_2\Big(1-\frac{1}{4\epsilon_0^2}\Big), \log_2\Big(\frac{\mu-\epsilon_0}{\mu-2\epsilon_0}\Big)+0.401\Big]+ \\ +\log_2\Big(\frac{\mu-\epsilon_0}{\mu-2\epsilon_0}\Big)+0.401.$$

$$\text{c}_{\text{space}} = \max\Big[-\frac{1}{2}\log_2\Big(1-\frac{1}{4\epsilon_0^2}\Big), \log_2\Big(\frac{\mu-\epsilon_0}{\mu-2\epsilon_0}\Big)+0.401\Big], \\ \textit{where } \mu \geq \frac{\epsilon_0}{1-\gamma}\Big(1+\frac{1}{n}\Big)+\epsilon_0 \textit{ and } \epsilon_0 > \frac{1}{2}.$$

Note that since $\mu \geq \frac{\epsilon_0}{1-\gamma}\big(1+\frac{1}{n}\big)+\epsilon_0$, it follows that $\mu > 2\epsilon_0$ and the logarithms are well defined. In the expressions above, term $\log_2\big(\frac{\mu-\epsilon_0}{\mu-2\epsilon_0}\big)$ corresponds to the exponent describing the number of centers $c_t$ and thus to $\log(\frac{1}{\gamma})$. The minimal time complexity is reached when we have the following equality:

$$-\frac{1}{2}\log_2\Big(1-\frac{1}{4\epsilon_0^2}\Big) = \log_2\Big(\frac{\mu-\epsilon_0}{\mu-2\epsilon_0}\Big)+0.401.$$

Although this last equation is solvable in the general case by treating either a quadratic or a cubic equation (using for the cubic the Cardano's method), we will omit giving a verbose expression just in terms of $\epsilon_0$. Another remark is that for asymptotically large $\mu$, an optimal value for $\epsilon_0$ is $\epsilon_0 \approx 0.765663$, while $\text{c}_{\text{time}} \approx 0.802$.

### Comparison with Previous Results

The effect of relaxing the search for a minimal difference of lattice vectors has been applied before on modifications of the ListSieve-Birthday algorithm. A first proposal has been put forward in [LWXZ11, WLW15], of complexity:

**Lemma 16.** ([WLW15, Theorem 6]) *The optimal time complexity of the $\mu$ListSieve-Birthday (introduced in [WLW15] as Algorithm 2) is $2^{c_{\text{time}}n+o(n)}$, where $c_{\text{time}} = 0.802 - 1.5\log_2(1-\mu^{-2/3})$. The corresponding space complexity is $2^{c_{\text{space}}n+o(n)}$, where $c_{\text{space}} = 0.401 - 0.5\log_2(1-\mu^{-2/3})$ and $\epsilon_0 = \frac{\sqrt[3]{\mu}}{2}$.*

A second proposal appears in [LWXZ11], where the authors impose an additional restriction on the lattice vectors in the list constructed by the ListSieve algorithm. Namely, they ask that the distance between every two lattice vectors be greater than $\mu\lambda$. The complexity of this adaptation of the ListSieve is given by the following result:

**Lemma 17.** ([LWXZ11]) *The time and space complexities of the algorithm referred to as Modified Algorithm in [LWXZ11] are $2^{c_{\text{time}}n+o(n)}$ and $2^{c_{\text{space}}n+o(n)}$, respectively. Expressions $c_{\text{time}}$ and $c_{\text{space}}$ are defined as $c_{\text{time}} = \max(2c_l + c_g, 2c_g)$, $c_{\text{space}} = \max(c_l, c_g)$, with $c_g$ as defined in Lemma 13 and $c_l = \log_2(\frac{\sqrt{\epsilon^2+\mu^2}+\epsilon}{\mu}) + 0.401$.*

We have numerically verified up to approximation factor $2^{16}$ that this adaptation of the AKS algorithm has a lower time complexity than both approximation variants of the ListSieve-Birthday, in Table 1 we compare the first values, keeping the same approximation factors as given in [LWXZ11, WLW15]. It should be noted that a trivial adaptation of the AKS does not perform better compared to the ListSieve-Birthday. For approximation factors in $\omega(1)$, even taking $\epsilon_0 = 1 - \frac{1}{2^\mu}$ will yield that the complexity of the ListSieve-Birthday adaptation is in $o$(complexity of $\mu$AKS). Our algorithm starts providing better time complexity bounds starting from $\mu \approx 2$.

**Table 1.** Comparison of time complexity bounds for $\mu$ListSieve-Birthday variants and $\mu$AKS. A table entry $c_{\text{time}}$ indicates a time complexity of $2^{c_{\text{time}}n+o(n)}$, when $n \to \infty$.

| | Modified ListSieve[LWXZ11] | $\mu$ListSieve-Birthday[WLW15] | $\mu$AKS | | |
|---|---|---|---|---|---|
| $\mu$ | $c_{\text{time}}$ | $c_{\text{time}}$ | $\epsilon_0$ | $\gamma$ | $c_{\text{time}}$ |
| 2.71 | 1.99758 | 2.36552 | 0.595838 | 0.718168 | 1.75721 |
| 3.61 | 1.77978 | 1.99933 | 0.624387 | 0.790868 | 1.47898 |
| 8 | 1.36434 | 1.42456 | 0.687324 | 0.90601 | 1.0868 |
| 15 | 1.16723 | 1.19071 | 0.719597 | 0.94961 | 0.951187 |
| 100 | 0.903217 | 0.904852 | 0.757947 | 0.992363 | 0.824121 |

In order to compare the space complexity, first we observe that our algorithm improves on the space complexity of the more efficient version of the ListSieve-Birthday variants at around $\mu \approx 3.37$ and we have numerically verified that this behaviour holds up to a $2^{16}$ approximation factor. In Table 2 we compare the complexities for the same small values of $\mu$.

The ListSieve-Birthday algorithm seems to be more efficient because in the list of vectors we use for reduction we keep only vectors of norm greater than a

**Table 2.** Comparison of corresponding space complexity bounds for $\mu$ListSieve-Birthday variants and $\mu$AKS. A table entry $c_{space}$ indicates a space complexity of $2^{c_{space}n+o(n)}$, when $n \to \infty$.

| | Modified ListSieve[LWXZ11] | $\mu$ListSieve-Birthday[WLW15] | $\mu$AKS |
|---|---|---|---|
| $\mu$ | $c_{space}$ | $c_{space}$ | $c_{space}$ |
| 2.71 | 0.833343 | 0.922173 | 0.878607 |
| 3.61 | 0.749856 | 0.800109 | 0.739491 |
| 8 | 0.59624 | 0.608519 | 0.543402 |
| 15 | 0.526077 | 0.53057 | 0.475594 |
| 100 | 0.435 | 0.435284 | 0.41206 |

threshold $r_0\lambda$, which when optimized is around $3.01\lambda$. Nevertheless, the proof of correctness for the ListSieve-Birthday requires that the shrinking factor has to be chosen as $1 - \frac{1}{n}$. We have no such restrictions on the algorithm we propose, and this is why we achieve a better time complexity when considering the approximate version of the shortest vector problem.

### Quantum Search

Most cryptographic primitives based on lattices are normally regarded as secure with respect to quantum computers. Nevertheless, it remains relevant to see whether quantum computers can employ speed-ups of the known classical algorithms solving the SVP in our case. In particular, the Grover quantum search algorithm can be used to speed up sieving algorithms, as pointed out in [LMP15]. The algorithm considers a list $L$ of size $N$ along with a function $f : L \to \{0, 1\}$ such that the set $f^{-1}(1)$ is small. When the memory model considered is a RAM memory, which is also quantumly addressable, the search requires $O(\sqrt{N})$ operations, as opposed to $O(N)$ in the classical setting. In the quantum setting, reducing our list of centers is done in $2^{c_{space}+c_t/2}$, which unsurprisingly yields, for asymptotically large approximations factors, a time complexity of $2^{0.602n+o(n)}$, just as the adaptations of ListSieve-Birthday. Naturally, as the quantum search algorithm brings only a speedup in the exponent of $\frac{1}{4}$, it follows that $\mu$AKS will still have a smaller time and space complexity when compared to ListSieve.

## 5 Conclusions and Open Questions

In this work, we give a new algorithm for approximate SVP that is slightly faster than previously known algorithms. More importantly, it justifies that the ListSieve algorithm by [MV10] does not outperform [AKS01] in every respect.

There are a number of open questions that still remain. In particular, all known algorithms for the shortest vector problem are constrained by the kissing constant from [KL78b], and it is not completely clear whether this is a fundamental bottleneck or just a shortcoming of the current techniques. Another reason for the gap in theoretical and practical results originates in the current bounds on the kissing constant, which are not presently known to be tight.

# Bibliography


[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in $2^n$ time using discrete gaussian sampling. *Symposium on Theory of Computing (STOC)*, 2015.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108, 1996.

[Ajt04] Miklós Ajtai. Generating hard instances of lattice problems. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 1–32. Dept. Math., Seconda Univ. Napoli, Caserta, 2004. Preliminary version in STOC'96.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.

[AKS02] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *CCC*, pages 41–45, 2002.

[BDGL] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24. `http://epubs.siam.org/doi/abs/10.1137/1.9781611974331.ch2`.

[BGJ14] Anja Becker, Nicolas Gama, and Antoine Joux. A sieve algorithm based on overlattices. *LMS Journal of Computation and Mathematics*, 17(A):49–70, 2014.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013.

[Bri85] Ernest F. Brickell. Breaking iterated knapsacks. In *Advances in cryptology (Santa Barbara, Calif., 1984)*, volume 196 of *Lecture Notes in Comput. Sci.*, pages 342–358. Springer, Berlin, 1985.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12, 2014.

[dB89] R. de Buda. Some optimal codes have structure. *Selected Areas in Communications, IEEE Journal on*, 7(6):893–899, Aug 1989.

[FT87] András Frank and Éva Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, pages 169–178. ACM, New York, 2009.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. *Coding and Cryptology: Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, chapter Algorithms for the Shortest and Closest Lattice Vector Problems. Springer Berlin Heidelberg, 2011. http://dx.doi.org/10.1007/978-3-642-20901-7_10.

[HR12] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. *Theory of Computing*, 8(23):513–531, 2012. Preliminary version in STOC'07.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[Kho05] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM*, 52(5):789–808, September 2005. Preliminary version in FOCS'04.

[KL78a] G. Kabatiansky and V. Levenshtein. On bounds for packings on a sphere and in space. *Probl. Peredachi Inf.*, 14(1):3–25, 1978. http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ppi&paperid=1518&option_lang=eng.

[KL78b] G. A. Kabatjanskiĭ and V. I. Levenšteĭn. Bounds for packings on the sphere and in space. *Problemy Peredači Informacii*, 14(1):3–25, 1978.

[Laa14] Thijs Laarhoven. Sieving for shortest vectors in lattices using angu r locality-sensitive hashing. *IACR Cryptology ePrint Archive*, 2014:744, 2014.

[Len83] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[LMP15] Thijs Laarhoven, Michele Mosca, and Joop Pol. Finding shortest lattice vectors faster using quantum search. *Des. Codes Cryptography*, 77(2-3):375–400, December 2015. http://dx.doi.org/10.1007/s10623-015-0067-5.

[LO85] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *J. Assoc. Comput. Mach.*, 32(1):229–246, 1985.

[LWXZ11] Mingjie Liu, Xiaoyun Wang, Guangwu Xu, and Xuexin Zheng. Shortest lattice vectors in the presence of gaps. *IACR Cryptology ePrint Archive*, 2011:139, 2011.

[Mic01] Daniele Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, March 2001. Preliminary version in FOCS 1998.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302 (electronic), 2007.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *SODA*, pages 1468–1480, 2010.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *J. Math. Cryptol.*, 2(2):181–207, 2008.

[PS09] Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009:605, 2009.

[Reg04] Oded Regev. Lecture 8 - a $2^{O(n)}$ time algorithm for SVP. *Lecture notes on lattices in computer science*, 2004. `http://www.cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/svpalg.pdf`.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):Art. 34, 40, 2009.

[Sha84] Adi Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Trans. Inform. Theory*, 30(5):699–704, 1984.

[WLTB11] Xiaoyun Wang, Mingjie Liu, Chengliang Tian, and Jingguo Bi. Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 1–9, New York, NY, USA, 2011. ACM.

[WLW15] Wei Wei, Mingjie Liu, and Xiaoyun Wang. Finding shortest lattice vectors in the presence of gaps. In *Topics in Cryptology — CT-RSA 2015*, pages 239–257. Springer International Publishing, April 2015. `http://dx.doi.org/10.1007/978-3-319-16715-2_13`.

[ZPH14] Feng Zhang, Yanbin Pan, and Gengran Hu. A three-level sieve algorithm for the shortest vector problem. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography – SAC 2013*, Lecture Notes in Computer Science, pages 29–47. Springer Berlin Heidelberg, 2014.