Tailoring a Genetic Algorithm for searching Shortest Lattice Vector in NTRU Lattices

Raul Caram de Assis [1], Thiago do Rêgo Sousa[2]

[1 2] Centro de Pesquisa e Desenvolvimento para a Segurança das Comunicações (Cepesc)

raulcaram21@gmail.com, thiagodoregosousa@gmail.com

Abstract. *The NTRU cryptosystem relies on the hardness of the Shortest Vector Problem (SVP), a central challenge in lattice-based cryptography. While genetic algorithms (GAs) have shown promise in solving SVP on generic lattices, little attention has been given to applying it to cryptographic structures like NTRU. In this work, we investigate the suitability of a GA specifically tailored to NTRU lattices. Our approach involves searching outside the lattice and analyzing a cost function whose behavior appears smoother when using BKZ-reduced bases. We present initial observations and outline directions for future research.*

1. Introduction

The NTRU system, proposed in [Hoffstein et al. 1998], is one of the most popular post-quantum cryptosystems whose security depends on the difficulty of solving a shortest vector problem (SVP) which is considered a classical lattice problem for which no efficient quantum or classical algorithm has been developed so far for the general case.

Many different methods have been developed to solve, approximately or exactly, the SVP and another highly related problem: the closest vector problem (CVP). Most notably, basis reduction algorithms LLL and BKZ and their variants, as well as sieving, prunning, random sampling and a series of different stochastic-search based algorithms or a combination of those.

Genetic algorithms (GAs) have proven effective in navigating large combinatorial search spaces and finding near-optimal solutions to hard problems, including the SVP. The starting point for a successful application of GAs to solve SVP was in the article [Ding et al. 2015] in which the authors improved upon some former solutions submitted to the Darmstadt challenges (see [Darmstadt lattice challenge]).

In the context of GAs, one of the first works was [Ding and Zhu 2014] that proposed a random sampling method based on y-sparse representations. [Ding et al. 2015] introduced a coordinate system based on the Gram-Schmidt orthogonalized basis along with binary encoding. [Luan et al. 2020] extended this idea using the Natural Number Representation introduced by [Fukase and Kashiwabara 2015]. [Reddy and Rao 2021] and [Fukase and Kaminaga 2022] further improved using evolutionary approaches. More recently, [Moghissi and Payandeh 2019] explored parallelization for SVP using evolutionary strategies.

Despite these advances, most research has focused on generic lattice instances from SVP challenge sets, with limited attention to cryptographic systems like NTRU. In this work, we investigate the potential of tailoring genetic algorithms specifically to attack the NTRU cryptosystem. Rather than applying general-purpose GA techniques, we

propose a domain-specific adaptation and examine its effectiveness. This paper presents preliminary ideas, outlines the design of our customized algorithm, and identifies open research questions that must be addressed to assess the practicality and scalability of such approaches in cryptanalytic contexts. The rest of the paper is organized as follows: In Section 2 we describe the NTRU system, the SVP problem that is associated to recovering its private key and the whole of genetic algorithms in solving SVP for general lattices. Section 3 presents our idea to tailor GAs to NTRU by searching for points outside the lattice and using an appropriate cost function. Section 4 concludes and gives further research directions.

2. SVP, NTRU and Genetic algorithms

2.1. Lattices and SVP

Given a basis $B = (\vec{b}_0, \ldots, \vec{b}_{n-1})$ containing linearly independent vectors in $\mathbb{R}^n$, to be treated as the $n \times n$ matrix having $\vec{b_i}$ as its $(i+1)$-th row; we define $L(B) = \{\vec{x}B | \vec{x} \in \mathbb{Z}^n\}$ as the lattice generated by B, and if B is clear from the context or not relevant we may simply write it as L. This is the set of all points in $\mathbb{R}^n$ that can be reached by starting from the origin $\vec{0}$ and through finitely many operations of adding or subtracting vectors in B. The basis is never unique and in fact for every lattice of dimension $n \geq 2$ there is an infinite number of different basis. The SVP for L is the problem of finding one $\vec{v} \in L^* = L \setminus \vec{0}$ such that for any $\vec{w} \in L^*$, $\|\vec{v}\| \leq \|\vec{w}\|$.

2.2. NTRU cryptosystem

NTRU is an asymmetric cryptographic system designed by [Hoffstein et al. 1998] to be quantum-safe. The system is based upon operations in a given convolutional polynomial ring $R_q = (\mathbb{Z}[x]/q\mathbb{Z})/(x^N - 1)$, for given $q, N \in \mathbb{N}_{>0}$ co-primes, with a pair of operations $(+, *)$ defined. In the context of NTRU, the reduction modulo q is 'center-lifted', *i.e.* it takes any integer z and returns the unique $z' \in C_q := \mathbb{Z} \cap (-q/2, q/2]$ such that $q|(z - z')$.

The polynomials in R_q have degree at most $N-1$ and its coefficients are members of C_q. The operations $+$ and $*$ (sum and convolution, respectively) of two polynomials in R_q are to be done as the ordinary sum/product of two polynomials followed by the application of two reductions: the integer coefficients are reduced modulo q and the exponents of the variables $x^0, x^1, \ldots$ are reduced modulo N.

In NTRU the private keys $\vec{f} = (f_0, \ldots, f_{N-1})$ and $\vec{g} = (g_0, \ldots, g_{N-1})$ are vectors with entries in $\{-1, 0, 1\}$ (more precisely they have a non-zero, fixed or random, quantity of each of these possible values). The parameters $\vec{f}$ and $\vec{g}$ are generated independently and from those the public key $\vec{h}$ is defined through a computation that is hard-to-invert without knowledge of $\vec{f}$ or $\vec{g}$, so that the public key does not reveal the private keys.

Precisely, $\vec{h} = (h_0, \ldots, h_{N-1})$ is defined by the relation $h(x) = p \cdot f^{-1}(x) * g(x)$ in the ring R_q (for some p co-prime with q and much smaller than it), thus is a vector with entries in C_q for a given $q \in \mathbb{N}_{>0}$. Except for $\vec{f}$ and $\vec{g}$, the other parameters are considered public. In matrix form we may express the relation between the keys by saying $\vec{f} \cdot (p^{-1}\mathbf{H}) = \vec{g} \mod q$, where $\mathbf{H}$ is defined as the circulant matrix containing the coefficients of the polynomial h and its rotations (see [Silverman et al. 2008, Section 6.11.1]).

The reduction modulo q implies the existence of $\vec{u} \in \mathbb{Z}^N$ satisfying $\vec{f} \cdot (p^{-1}\mathbf{H}) - q\vec{u} = \vec{g}$. Fixing $n = 2N$ and given $\vec{h}$ and q, we consider the following as a basis, represented here as a block of four $N \times N$ square matrices, of an NTRU lattice L of dimension n:

$$B = \begin{pmatrix} q\mathbf{I} & \mathbf{0} \\ p^{-1}\mathbf{H} & \mathbf{I} \end{pmatrix},$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{0}$ is the matrix with all entries '0'. And thus we have $(-\vec{u}, \vec{f}) \cdot B = (\vec{g}, \vec{f})$.

Consequently $(\vec{g}, \vec{f})$ is a point of L, of coordinates $(-\vec{u}, \vec{f})$ in basis B, and, considering its entries are all in the set $\{-1, 0, 1\}$, it is likely a shortest non-zero vector of L. Also, it is established in [Hoffstein 2008, Section 7.10.2] that if you pick $\vec{g}$ and $\vec{f}$, rotate their entries, each by the same amount and in the same direction, concatenate them into a new $(\vec{g'}, \vec{f'})$ pair, then this vector is also in L. In addition, by the defining properties of a lattice, so is the negative of each of these rotated vectors. And each of these n non-zero vectors in $\{-1, 0, 1\}^n \cap L^*$ will serve as a decryption key.

This way, discovering $\vec{f}$ or $\vec{g}$ from the public parameters can be reduced to solving the SVP for a particular lattice of dimension n. The search for a short non-zero vector in a lattice $L(B)$ can be understood as finding the set of coordinates $\vec{x} \in \mathbb{Z}^n \setminus \vec{0}$ that minimizes $\|\vec{x}B\|$ (as any lattice vector is an integer linear combination of the basis vectors and vice-versa) and thus SVP is inherently an optimization problem. In this model, when working with any directed search-based algorithm, the natural approach is to make the quality value of a lattice point related to how short it is.

2.3. Genetic Algorithms

Genetic algorithms (GA) are a class of population-based optimization algorithms inspired by the mechanism of evolution, reproduction, and survival of the fittest individuals. GAs perform iterations, evolving a set of approximate solutions — called individuals — until the exact (or a good enough) solution has been found. It is an essential part of genetic algorithms to associate to each individual a fitness value used to quantify the quality of the solutions and to make the current set of solutions interact with one another to derive a new set of solutions. The fitness function must be appropriately defined in that: (i) The individuals within the search space will maximize the fitness if, and only if, they are exact solutions to the underlying problem, and (ii) there is a general trend that the better the approximation to the exact solution the greater the fitness value.

Underlying any GA, there are (typically random) sub-procedures of crossover and mutation. Informally speaking, crossover is about generating a third individual from a pair of previous individuals in a way the new individual bears resemblance to its parents. Mutation is about replacing an individual by another that is different although expected to be similar. Local search procedures were developed independently from GAs, but can be integrated into it. A local search procedure consists of repeatedly replacing a given solution by another, in the neighborhood of the current one (to be conveniently defined in the next section), that maximizes the fitness value until this procedure is no longer able to improve the current solution.

The work of [Luan et al. 2020] deviates a bit from [Ding et al. 2015] for not including a local search step. On the other hand, it does selection through a deterministic process and includes a random restart step to avoid stagnation. To a more detailed description we refer to their articles in the references section.

3. Tailoring a GA to attack NTRU

Our approach follows the idea of using a GA to find SVP solutions but differs from the previously developed algorithms in that our goal is only to solve NTRU lattices. That allow us to operate on the additional assumption that the exact solutions belong to the set $S = \{-1, 0, 1\}^n \setminus \vec{0}$ which we can use as our search space, that is, the set of individuals we consider as approximate or exact solutions. This forces us to modify the definition of fitness as, unlike the others, we do not operate by iteratively finding shorter vectors in L but by minimizing, within S, the distance to L. Thus we shift the paradigm from searching a short vector in L^* to searching in S a vector in L, since the set $S \cap L$ must contain all SVP solutions. This is a core difference in the structure of our GA algorithm from the previous ones and as a consequence our search space is mostly comprised of vectors outside L.

While we could directly apply [Ding et al. 2015] or [Luan et al. 2020]'s algorithm to find short vectors in any NTRU lattice, we want to investigate if its possible to take any advantage on the specificity of the problem and come up with a new algorithm that is more efficient for the subclass of SVP problems in NTRU lattices.

Given a basis B of an NTRU lattice L, if we consider the vector space E spanned by the vectors in B, each point $\vec{v} \in E$ has coordinates $\vec{x} = \vec{v}B^{-1} \in \mathbb{R}^n$, the subset of E with all integer coordinates represent the lattice L. Let the fractional coordinates of any $\vec{v} \in E$ be the vector $\zeta_B(\vec{v}) = \vec{v}B^{-1} - \lfloor \vec{v}B^{-1} \rceil$, where $\lfloor \vec{x} \rceil$ refers to the vector one gets by rounding to the nearest integer every entry of $\vec{x}$. Then the values of $\zeta_B(\vec{v})$ describe what we call a path from $\vec{v}$ to some point of L. More precisely, if $\zeta_B(\vec{v}) = (\zeta_B(\vec{v})_1, \ldots, \zeta_B(\vec{v})_N)$ then starting from $\vec{v}$ if one moves exactly $\zeta_B(\vec{v})_i \cdot \|\vec{b_i}\|$ in the direction of $\vec{b_i}$, for every $i \in \{0, \ldots, N-1\}$, the final destination is the point $\lfloor \vec{v}B^{-1} \rceil B \in L$.

We conclude $\|\zeta_B(\vec{v})\|$ is a proxy measure of the distance from $\vec{v}$ to L and the length of the path they describe is an upper limit for the distance from $\vec{v}$ to L. Note that if the basis vectors are closer to being orthogonal this measure becomes more accurate because, in the degree to which they are parallel, moving in one direction can undo moving in another direction. Due to the fractional coordinates dependency on the choice of basis B this proxy measure can be improved by the beforehand application to B of a basis reduction algorithm such as LLL or BKZ.

We define the functions $cost_B(\vec{v}) = \|\zeta_B(\vec{v})\|^2$ and $fitness_B(\vec{v}) = cost_B(\vec{v})^{-1}$. One may consider $fitness_B(\vec{v}) = \infty$ if $cost_B(\vec{v}) = 0$ but this is not strictly necessary as our algorithm terminates once it finds a zero cost vector.

To form the first generation, we first generate $2n$ vectors drawn from S according to a uniform distribution. The procedures of selection, crossover, mutation and local search are used (in this order) to generate the next generation individuals that will replace the current ones.

When it comes to selecting a pair $(\vec{s}, \vec{t})$ for the crossover operation, we choose two

distinct elements of the population with probability proportional to their fitness.

The crossover operation is defined as follows: Given $\vec{s} = (s_0, \ldots, s_{n-1})$ and $\vec{t} = (t_0, \ldots, t_{n-1})$, both in S, crossover$(\vec{s}, \vec{t}) = \vec{u} = (u_0, \ldots, u_{n-1})$ where each u_i is calculated independently as the realization of a uniform random variable in $\{s_i, t_i\}$. So the amount of genes that $\vec{u}$ inherits from each parent is a binomial variable $Bin(n, 0.5)$.

Let they symbol $\boxplus$ stand for the operation on the set $\{-1, 0, 1\}$: $x \boxplus y = (x+y+1 \mod (3)) - 1$. Once we have the crossover output, the mutation operation is applied to $\vec{u}$ according to the rule: For each u_i, take only one of these actions: with probability $1/n$ make $u_i \leftarrow u_i \boxplus 1$, with probability $1/n$ make $u_i \leftarrow u_i \boxplus 2$, with probability $1 - 2/n$ leave it unchanged. If it happens that, after mutation, the resulting vector is not in S then using the same selected pair, redo the crossover and mutation operations. This ensures the output after mutation is in S.

Now, taking this updated $\vec{u}$ we improve it through a local search on S. We define a neighborhood of $\vec{u}$, $N(\vec{u})$, as the set containing only $\vec{u}$ and the points in S that differ from $\vec{u}$ on a single coordinate, that is, the points within a Hamming distance of 1 from $\vec{u}$. The local search consists of repeatedly updating $\vec{u}$ to one point in $N(\vec{u})$ with the smallest cost until this step can no longer improve the solution.

This process of selection of pair, applying crossover, mutation and then local search are used to generate the majority of individuals that will compose the new population. Keeping one or more of the fittest of one generation to the next is known as an "elitist strategy" and has been used in GAs for many different applications. This strategy guarantees that the maximum fitness observed in the current generation is no lower than that of the previous and has been used, in a different way in both [Ding et al. 2015] and [Luan et al. 2020]. In our case, following the standard of Ding et al., we always keep only the best of each generation.

In this way, we generate population after population until we meet the stopping criteria: A population having an individual with cost zero, that is, a point in L. To the degree that lower cost values correlate with proximity to L, our algorithm tends to replace current solutions for newer solutions that tend to be closer to optimal ones.

3.1. Experimental evaluation of the cost function

Our initial investigation of the cost function examined its behavior near the key $(\vec{g}, \vec{f})$ and observed whether the resulting cost varied only slightly — a characteristic desirable for optimization methods.

We also observed that this local behavior differs depending on whether the lattice basis is BKZ-reduced. Preliminary results show that with a BKZ-reduced basis, the cost function becomes smoother around the optimum. This smoothness may benefit optimization algorithms by providing more consistent gradients or search directions. We have seen this behavior when increasing the NTRU lattice dimension and we report in Figure 1 the results for a lattice of dimension $n = 200$, as $N = 100$ for NTRU. It is worth observing that when the basis is not BKZ-reduced, altering coordinates corresponding to the coefficients of the g polynomial changes slightly the cost function, but altering the coefficients of the f polynomial changes significantly (see the Blue clusters of small and big values in Figure 1(a)).

(a) Histogram (Full Range) **(b) Histogram (Filtered [0, 1])**

Figure 1. Cost values for points in the neighborhood of $(\vec{f}, \vec{g})$, i.e., vectors that differ at only one coordinate. Using the original non-BKZ-reduced basis VS using a BKZ-reduced basis (block size = 10) of dimension $n = 200$.

When expanding our analysis to a range of Hamming distances (Figure 2) we observe that without any basis reduction the correlation between Hamming distance from the secret key and the cost value of a point is weak and the distribution basically stagnates after Hamming distance 5. With LLL or BKZ-reduction, however, if we interpolate the medians of every box-plot we would reach something close to a concave log-like function that slowly increases the median cost values with the Hamming distance. For a lattice of dimension 100, the block size parameter of the BKZ-reduction or whether we use LLL instead did not seem to strongly affect the distribution of cost values by their Hamming distance from the secret key.

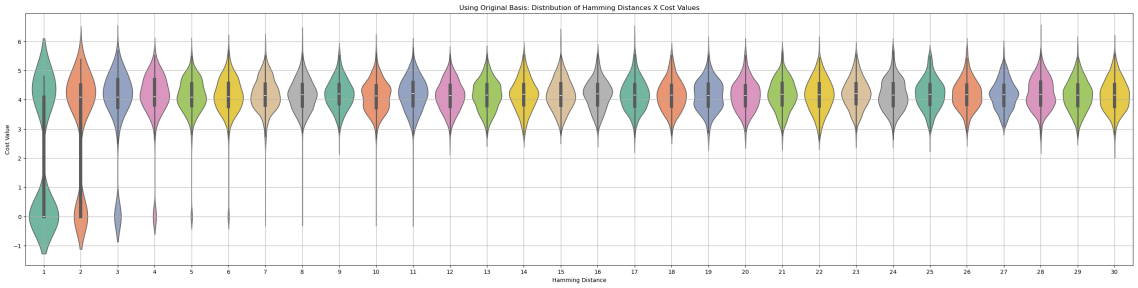
4. Conclusion

The use of genetic algorithms, as well as other metaheuristic methods, have proven to be useful in solving problems of combinatorial nature, including SVPs. The properties of an NTRU lattice allow a redesign of the previously published GAs that solve this problem.

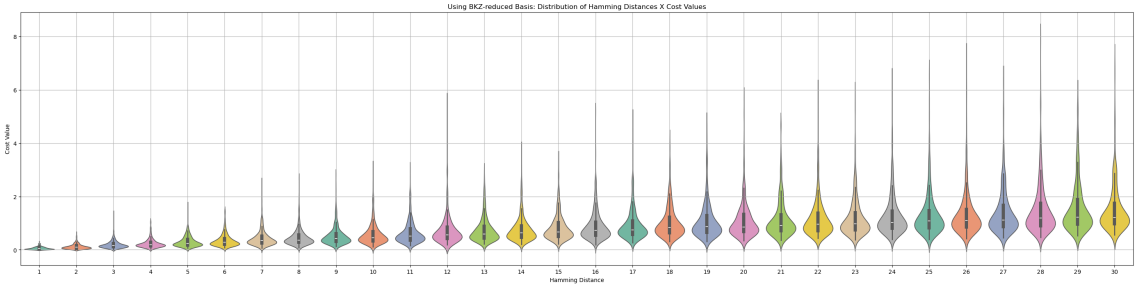
We have shown it is possible to use a search space that is not solely made of lattice points to find the shortest vector in a lattice by constructing an algorithm that operates that way. Although in this article we only focus on NTRU lattices, the core ideas behind our algorithm could in the future extend to solving other types of lattices $L \subset \mathbb{Z}^n$, for some $n \in \mathbb{N}$, by conveniently modifying the definition of the search space, S, to be a subset of $\mathbb{Z}^n$ that contains SVP solutions. That is, as long as we can define the search space S to contain the SVP solutions, and preferably no other lattice vectors and make it as small as possible, we can apply the exact same principles.

NTRU lattices are good candidates to test our ideas as we know that the SVP solutions have entries in $\{-1, 0, 1\}$. So it allows for a straightforward definition of S. Future research questions for this work include: Are there better cost functions to use? Is it practical to reduce the search space to just the coordinates of f, by computing a candidate g using the public key h? Can we beat other GA algorithms when it comes to NTRU lattices SVPs? If we get a competitive GA for solving SVP in NTRU lattices, will a generalization of this algorithm for general lattices remain competitive?

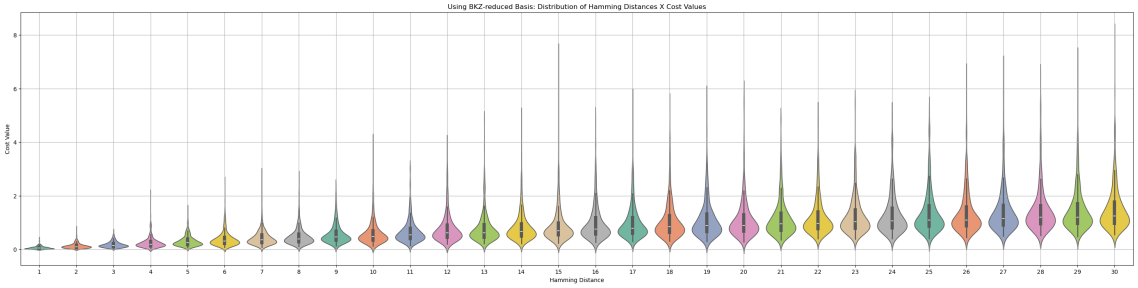
The source code is available at `https://github.com/RaulCA/Genetic-Algorithm-for-SVP-in-NTRU-lattices/`.



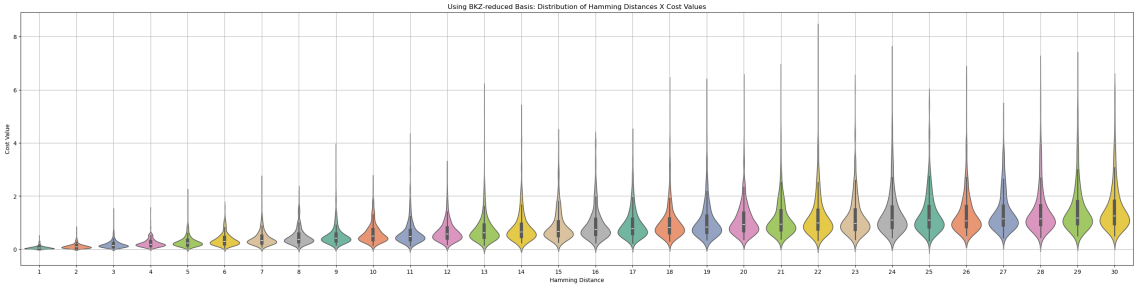
(a) No reduction



(b) With LLL reduction



(c) With BKZ-reduction (block size = 3)



(d) With BKZ-reduction (block size = 50)

Figure 2. Violin plots of how the cost function increases as we move away from the secret pair $(\vec{g}, \vec{f})$ in a lattice of dimension $n = 100$. Applying LLL or BKZ-reduction makes the correlation better between higher costs and greater Hamming distances from the exact solution. For each hamming distance we used 500 samples.

References

Ajitha, S. K., Biswas, S., and Kurur, P. P. (2011). Metropolis algorithm for solving shortest lattice vector problem (svp). In *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, pages 442–447. IEEE.

Darmstadt lattice challenge. Svp challenge hall of fame. `https://www.latticechallenge.org/svp-challenge/halloffame.php`. Accessed: 2025-04-22.

Ding, D. and Zhu, G. (2014). A random sampling algorithm for svp challenge based on y-sparse representations of short lattice vectors. In *2014 Tenth International Conference on Computational Intelligence and Security*, pages 425–429. IEEE.

Ding, D., Zhu, G., and Wang, X. (2015). A genetic algorithm for searching the shortest lattice vector of svp challenge. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation*, pages 823–830.

Fukase, M. and Kaminaga, M. (2022). Improving genetic algorithms for solving the svp: focusing on low memory consumption and high reproducibility. *Iran Journal of Computer Science*, 5(4):359–372.

Fukase, M. and Kashiwabara, K. (2015). An accelerated algorithm for solving svp based on statistical analysis. *Journal of Information Processing*, 23(1):67–80.

Hoffstein, J. (2008). An introduction to cryptography. In *An Introduction to Mathematical Cryptography*, pages 1–58. Springer.

Hoffstein, J., Pipher, J., and Silverman, J. H. (1998). Ntru: A ring-based public key cryptosystem. In *International algorithmic number theory symposium*, pages 267–288. Springer.

Luan, L., Gu, C., and Zheng, Y. (2020). A genetic algorithm with restart strategy for solving approximate shortest vector problem. In *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*, pages 243–250. IEEE.

Moghissi, G. R. and Payandeh, A. (2019). A parallel evolutionary search for shortest vector problem. *International Journal of Information Technology and Computer Science*.

Reddy, V. D. and Rao, G. (2021). Meta-heuristic approaches to solve shortest lattice vector problem. *Journal of Discrete Mathematical Sciences and Cryptography*, 24(1):81–91.

Silverman, J. H., Pipher, J., and Hoffstein, J. (2008). *An introduction to mathematical cryptography*, volume 1. Springer.