Journal of Discrete Mathematical Sciences and Cryptography

ISSN: 0972-0529 (Print) 2169-0065 (Online) Journal homepage: https://www.tandfonline.com/loi/tdmc20

Meta-heuristic approaches to solve shortest lattice vector problem

V. Dinesh Reddy & G.S.V.R.K. Rao

To cite this article: V. Dinesh Reddy & G.S.V.R.K. Rao (2019): Meta-heuristic approaches to solve shortest lattice vector problem, Journal of Discrete Mathematical Sciences and Cryptography, DOI: 10.1080/09720529.2019.1675300

To link to this article: https://doi.org/10.1080/09720529.2019.1675300

Published online: 17 Nov 2019.

Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
DOI : 10.1080/09720529.2019.1675300

Taylor & Francis
Taylor & Francis Group

Meta-heuristic approaches to solve shortest lattice vector problem

V. Dinesh Reddy *
Global Technology Office- Fundational Research
Cognizant Technology Solutions
Hyderabad 500032
Telangana
India

G.S.V.R.K. Rao †
Global Technology Office- Fundational Research
Cognizant Technology Solutions
Chennai
Tamil Nadu
India

Abstract

We present the aptness of population based meta-heuristic approaches to compute a shortest non-zero vector in a lattice for solving the Shortest lattice Vector Problem (SVP). This problem has a great many applications such as optimization, communication theory, cryptography, etc. At the same time, SVP is notoriously hard to predict, both in terms of running time and output quality.

The SVP is known to be NP-hard under randomized reduction and there is no polynomial time solution for this problem. Though LLL algorithm is a polynomial time algorithm, it does not give the optimal solutions. In this paper, we present the application of Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) to solve the SVP on an appropriate search space. We have implemented the PSO, GA and LLL algorithm for the SVP. The comparison results shows that our algorithm works for all instances.

Subject Classification: *(2000) cc:68T20*

Keywords: *Shortest Vector Problem, Lattice Problems, PSO, GA*

*E-mail: dineshvemula@gmail.com (Corresponding Author)
†E-mail: subrahmanyavrk.rao@cognizant.com

1. Introduction

Most crypto-system depends on the discrete logarithm problem in multiplicative cyclic groups or the guesstimated hardness of integer factorization. However, Shor [1] gave efficient quantum algorithms for integer factorization, which would break public-key cryptography schemes if large-scale quantum computers can be built. But, lattice based cryptography would be resistant to attacks by quantum computers. A thorough study on the rich structure of lattices reveals that they have numerous applications in mathematics, cryptography and computer engineering. In particular, polynomials factorization [2, 3], pure integer programming in high dimension spaces [4, 5], and many other lattice problems [6–10].

Crypto-systems based on Lattices are often simple to implement and have parallelizable structure that makes them faster in certain contexts. Moreover, crypto system based on algebraic lattices and non-associative structures over certain rings can even outperform traditional systems in some cases [11]. Cryptography fundamentally need average-case complexity, i.e., random instances are hard to solve. This is qualitatively different from the worst-case notion of hardness, where breaking the cryptographic construction implies an efficient algorithm for solving any instance of some underlying lattice problem in the worst case.

In this paper, we present the application of Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) to solve the SVP for various lattices. The remainder of this paper is organized as follows. Section 2 describes related work in solving shortest vector problem and other lattice problems which are basis for powerful cryptography. Section 3 presents some preliminaries that will be required to understand the lattices and shortest vector problem. Section 4 describes the proposed algorithms. Section 5 presents the experimental results and comparison with the state-of-the-art algorithms on benchmark instances followed by conclusions and future directions in Section 6.

2. Related Work

During the 80's, cryptographers who solved the shortest vector problem turned many heads for their accomplishments in the number theory. Recent discoveries in the cryptography lead to the study of lattice based problems for the past decade and a half. The shortest vector problem and other lattice problems are now the basis for many powerful cryptographies such as homomorphic encryption in security as SVP is capable of handling worst-case hardness of approximating the decision

version [12]. SVP is known to be both exact and approximate of its version in the computational complexity point of view [13]. In the hardness perspective, SVP is shown to be both NP-hard and hard on to approximate within random constant factor and within nc/log log n for some specific non zero and non-negative constant under reasonable complexity respectively. In Polynomial-time algorithms, LLL basis reduction gives a $2^{O(n)}$ approximation algorithm for SVP and a approximation factor of $r^{n/r}$ in $2^{O(r)}poly(n)$ with Schnorrs block reduction algorithm [14, 15]. This gives a promising time and better approximation.

Algorithms used for solving the exact SVP requires exponential time and space even for the polynomial approximation factors sometime. This applies also to the algorithm which we represent in here. The above represented algorithms are used for assessing the security of the lattice based cryptography [16]. But the algorithms are also applied as the subroutines for the current approximation algorithms. They are applied to some applications with naturally arising low dimensional lattices. Therefore, the importance of these algorithms are for both theoretical and practical implementation despite of the exponential running time. The approximation of lattice problems is static over two decades but the exact algorithms are significantly dynamic. For SVP, three different classes of algorithms are developed. The first is based on the combination of strong basis reduction and the exhaustive enumeration inside Euclidean balls which was developed by Kannan [5]. The same was refined by many others. The current fastest algorithm in this class can solve SVP in $n^{n/(2e)}$ time while using poly(n) space. Later, a sub-exponential approximation factor of $2^{O(n(loglogn)^2/\log n)}$ was obtained by improved algorithms [17].

Lattice basis reduction algorithm such as LLL and BKZ are known for finding short vectors with large approximation factors. Particularly in BKZ we achieve the time complexity and the output quality by tuning the block-size parameter β. Higher β produces short vectors in output basis taking long time for the algorithm [18]. β used as a subroutine, helps BKZ solve the exact shortest vector problem. Here the runtime depends on the β. By forth, complexity of solving exact SVP directly impacts the estimated hardness of solving approximate SVP with BKZ. The algorithm AKS named after Ajtai, Kumar and Sivakumar is similar to this work. This algorithm constructed a method which creates shorted vectors iteratively combining the randomly generated lattice vectors. This gives the first ** -time and space randomized algorithm for SVP. This method is based on Randomized Sieving [14].

3. Preliminaries of Lattices and SVP

Definition (Lattices). Given *n linearly independent* vectors $\mathbf{b}_1,..., \mathbf{b}_n \in \mathbb{R}^m$, a lattice L of dimension n is a discrete additive subgroup of $\mathbb{R}^m$ and described as the set of all integer combinations of n linearly independent vectors b_i of $\mathbb{R}^m$. the lattice generated by them is defined as

$$\mathcal{L}(v_i) \overset{def}{=} \left\{ \sum_{i=1}^{n} \alpha_i b_i \mid \alpha_i \in \mathbb{Z} \right\}$$

Where $v_i = \mathbf{b}_1, ..., \mathbf{b}_n$ is a *basis* of the lattice($\mathcal{L}$). Note that the definition requires $\mathbf{b}_1, ..., \mathbf{b}_n$ to be linearly independent over $\mathbb{R}^m$ where $m \geq n$. The integer n is the dimension of the lattice L. If m = n, the lattice is called as full dimension lattice. Given a lattice with basis v_i, the shortest vector problem is to find the non-zero vector $\vec{u}$ that minimizes the $|\vec{u}|$ (or) Given a basis for a lattice $\mathcal{L} \subseteq \mathbb{R}^m$, SVP is to compute a non-zero vector in $\mathcal{L}$ of minimum Euclidean norm.

4. Proposed Method

4.1 *Particle Swarm Optimization (PSO)*

PSO is a population-based stochastic search algorithm based on simulation of the social interaction of animals [19, 20]. PSO is a suitable to find optimal value of a numerical function in a continuous domain. PSO is a collective, anarchic, iterative method with emphasis on cooperation. Each particle in the swarm is able to communicate to some other in the position and quality of the best site it knows. The algorithm is initialized with a population of solutions/particles. These particles move stochastically determine the moving direction in the search space. Each particle moves in a multi-dimensional space according to its own velocity, particle's best performance, and the best performance of its best informant. PSO is a collective, anarchic, iterative method with emphasis on cooperation [21]. Each particle in the swarm is able to communicate to some other in the position and quality of the best site it knows. The algorithm is initialized with a swarm of random solutions. Each particle moves in a multi dimensional space according to its own velocity, particle's best performance, and the best performance of its best informant. For each iteration PSO tries to find the global best and local best solutions among the generated neighborhood structures. Once we find the global best and local best, it tries to update all the particles in the population. Thus we get a new set of neighborhoods and looks for new optimum point. This

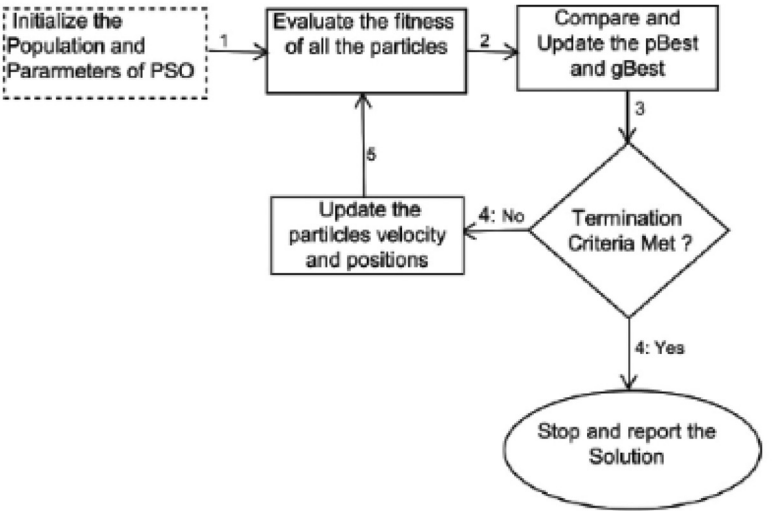

Fig 1

Flow chart of PSO

process continues until we get continuous improvements in the updated particle or up to max number iterations.

Particle Design :

Let B be an n x n non-singular matrix and let u be a n x 1 vector, $\|u\| \leq k$, k being a non-zero constant. We need to find an integer vector w such that Bw = u and u is the shortest vector in $\mathcal{L}$. We define a particle as a list of 'n' integer values. Then each particle position corresponds to an integer vector W_i, represented as follows.

$$W_i(t) = (2, -5, 35, ..., -62, ..., -20)$$

For each particle PSO calculates the $BW_i(t)$. PSO has the ability to keep track the particle that gives the shortest vector in the lattice $(\hat{S}V_j)$ and this is global best. When a particle takes part of the population, best experience it had till that point of time is called local best (SV_i). The particle velocity is updated using these best values according to the Equation 1. Further, each particle updates its position using the updated velocity, and its current position as shown in Equation 2.

$$V_i(t+1) = wV_i(t) + k_1 r_1(t)(SV_i(t) - W_i(t)) + k_2 r_2(t) (\hat{S}V_j(t) - W_i(t)) \quad (1)$$

$$W_i(t+1) = W_i(t) + V_i(t+1) \quad (2)$$

Here, k1 and k2 are the learning factors which determines the convergence properties of the algorithm, w is the inertia weight coefficient that determines how the previous velocity of the particle influences the velocity in the next iteration, and r1, r2 are the random number between (0,1).

4.2 *Genetic Algorithm (GA)*

A genetic algorithm is an adaptive heuristic search algorithm which can be applied to both constrained and unconstrained optimization problems [22–24]. GA is a meta-heuristic built on the trans-formative process of natural selection, mating, and reproduction that mimics biological evolution. A GA simulates this process with natural populations of individuals that evolve according to the principles of natural selection and survival of the fittest. Each individual of the population is coded to make a chromosome that represents a integer vector used to calculate the shortest vector for a given basis.

5. Results

A particle swarm like meta-heuristic algorithm works for a combinatorial optimization problem on a given search space associated with the instance. The search space for SVP problem can be a directed graph, where each vertex of the graph represents a lattice point of the given basis. We will have an edge from a lattice point u to other lattice point v, if v can be generated from u through some inexpensive computation. Each lattice point has a value associated with it which represents the vector magnitude (euclidean norm). The goal of the proposed algorithm is to

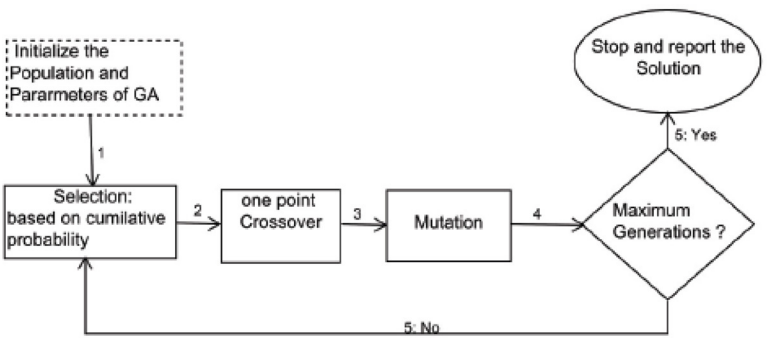

Fig 2

Flow chart of Genetic Algorithm

Table 1

Magnitude of shortest vector in the lattice for each iteration

Iteration	PSO	RANDOM	GA
1	857.7873	14265.02	8330.71
2	860.5022	11352.96	8929.55
3	782.1419	13253.36	7920.44
4	861.1016	13362.64	8013.06
5	861.1016	14495.91	8097.29
6	880.8059	12607.21	7379.19
7	955.4413	15161.54	8305.3
8	891.7404	14130.98	8022.04
9	862.3526	12668.39	8316.05
10	737.2082	15004.69	8927.03
11	785.3572	15005.19	7926.71
12	880.8059	15031.09	7481.671
13	841.0987	14360.35	8026.28
14	727.6723	18565.61	8072.72
15	726.011	11352.96	7122.17
16	761.3192	13946.68	8400.01
17	726.011	12858.79	7788.18
18	951.812	13349.58	8493.59
19	841.0987	16278.24	8757.32
20	727.6723	12980.41	8271.16
21	951.812	12865.23	7122.17
22	891.7404	13104.85	7417.85
23	834.4993	12514.68	9013.19
24	834.4993	14365.68	8255.18
25	891.7404	13554.81	8293.25
26	891.7404	11990.01	8038.68
27	861.1016	13244.97	8397.51
28	737.2082	14710.35	8622.32
29	726.011	11562.45	8261.22
30	785.3572	13221.59	7799.61

Table 2
Magnitude of shortest vector for given instance

Type	Dimension	PSO	GA	RANDOM
Dual Modular	10	4.5825	7.4833	10.9544
Modular	10	2.4491	2.6457	2.6457
Random	10	2.645	5.291502	6.2449
Swift	8	4.1232	4.2426	4.2426
NTRU	8	3.6055	4.2426	5.9160

reach to a point that is the shortest vector of the lattice for a given basis. The PSO algorithm generates a finite neighborhood structures for the given basis randomly and searches for the shortest vector among those solutions. For each iteration PSO tries to find the global best and local best solutions among the generated neighborhood structures. Once we find the global best and local best, it tries to update all the particles in the population. Thus we get a new set of neighborhoods and looks for new optimum point. This process continues until we get continuous improvements in the updated particle or up to max number iterations.

We compared the proposed algorithm with the randomized algorithm and genetic algorithm on benchmark instance SVPs. The randomized algorithm updates the solutions each time with some random weights. A genetic algorithm is an adaptive heuristic search algorithm which can be applied to both constrained and unconstrained optimization problems [23, 24]. Genetic algorithms are built based on natural evolution. Each individual of the population is coded to make a chromosome that represents a integer vector to calculate the shortest vector for a given basis. We have used a benchmark lattice from [25]. We have tested our algorithm for this instance and compared with GA and random algorithms. Further, we have also tested our algorithms for Random, ntru, Modular / Dual Modular , and SWIFT lattices given in [25]. We repeated the experiments upto 30 times for the said approaches and presented the magnitude of the shortest vector in Table 1. We found that PSO works faster and gives the optimal value compared to the other algorithms as shown in the Table 2.

6. Conclusion

We have tested the aptness of the Meta-heuristic approaches, and their generalization for solving SVP which is NP-Hard. We have started with a random a search space for the problem, and expanded this search space by

adding the vectors that are smaller than the existing points. We compared the performance of PSO, GA, and RANDOM algorithms for solving the said instances and found that PSO has performed well compared to the other algorithms. In our future work, we will explore our approach for larger instances.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, IEEE, (1994).

[2] H. W. Lenstra Jr, "Integer programming with a fixed number of variables," *Mathematics of operations research*, vol. 8, no. 4, pp. 538–548, (1983).

[3] A. Schönhage, "Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm," *in International Colloquium on Automata, Languages, and Programming*, pp. 436–447, Springer, (1984).

[4] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515-534, (1982).

[5] R. Kannan, "Minkowski's convex body theorem and integer programming," *Mathematics of operations research*, vol. 12, no. 3, pp. 415–440, (1987).

[6] A. Shamir, "A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem," in *Proceedings of 23rd Annual Symposium on Foundations of Computer Science*, pp. 145–152, IEEE, (1982).

[7] J. Hastad, "Solving simultaneous modular equations of low degree," *siam Journal on Computing*, vol. 17, no. 2, pp. 336–341, (1988).

[8] A. M. Frieze, J. Hastad, R. Kannan, J. C. Lagarias, and A. Shamir, "Reconstructing truncated integer variables satisfying linear congruences," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 262–280, (1988).

[9] D. Coppersmith, "Small solutions to polynomial equations, and low exponent rsa vulnerabilities," *Journal of Cryptology*, vol. 10, no. 4, pp. 233-260, (1997).

[10] M. Bellare, S. Goldwasser, and D. Micciancio, "Pseudo-random generators within cryptographic applications: the dss case," in *Advances in CryptologyCRYPTO*, vol. 97, pp. 277–291, (1997).

[11] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium*, pp. 267–288, Springer, (1998).

[12] C. Gentry et al., "Fully homomorphic encryption using ideal lattices.," in *Stoc*, vol. 9, pp. 169–178, (2009).

[13] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximate optima in lattices, codes, and systems of linear equations," *Journal of Computer and System Sciences*, vol. 54, no. 2, pp. 317–331, (1997).

[14] M. Ajtai, R. Kumar, and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 601-610, ACM, (2001).

[15] C.-P. Schnorr, "A hierarchy of polynomial time lattice basis reduction algorithms," *The oretical computer science*, vol. 53, no. 2-3, pp. 201-224, (1987).

[16] P. Q. Nguyen and J. Stern, "The two faces of lattices in cryptology," in *Cryptography and lattices*, pp. 146–180, Springer, (2001).

[17] D. Micciancio, *On the hardness of the shortest vector problem*. PhD thesis, Massachusetts Institute of Technology, (1998).

[18] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM* (*JACM*), vol. 56, no. 6, p. 34, (2009).

[19] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, (2007).

[20] V. D. Reddy, G. R. Gangadharan, and G. S. V. R. K. Rao, "Energy-aware virtual machine allocation and selection in cloud data centers," *Soft Computing*, pp. 1-16, (2017).

[21] C. Jatoth and G. Gangadharan, "Qos-aware web service composition using quantum in spired particle swarm optimization," *in International Conference on Intelligent Decision Technologies*, pp. 255–265, Springer, (2017).

[22] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, (1992).

[23] D. E. Goldberg, *Genetic algorithms*. Pearson Education India, (2006).

[24] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem," *European journal of operational research*, vol. 94, no. 2, pp. 392–404, (1996).

[25] L. Richard and S. Michael, "Hard lattice generator," *http://doc.sage-math.org/html/en /reference/cryptography/sage/crypto/lattice.html*.

Received January, 2019