MAP-SVP

MAP-SVP: Massively Parallel Solver for Shortest Vector Problem

***Yuji Shinano**, Zuse Institute Berlin*

In Short

- MAP-SVP is a massively parallel solver for Shortest Vector Problem (SVP).
- Our experimental implemenation of MAP-SVP is the world first practical asynchronous distributed-memory solver of SVP.
- The experimental implemenation of MAP-SVP achieved new records for 104, 111, 121 and 127 dimensions in SVP Challenge.

A *lattice* is a discrete subgroup of the Euclidean space, $\mathbb{R}^n$. A lattice L of dimension n is spanned by a *basis* $\mathbf{B}$ consisting of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$, and any vector in L can be represented as a linear combination of the $\mathbf{b}_i$s with integer coefficients. In the past few years, lattices have attracted considerable interest in cryptography. In particular, with the recent development of quantum computers, since 2015, the US National Institute of Standards and Technology (NIST) started developing new standards for *post-quantum cryptography (PQC)* and called for proposals to prepare information security systems that can resist quantum computers [1]. (cf., The most popular cryptographic systems, such as RSA, DSA, and ECDSA, could be broken by Shor's algorithms with the use of large-scale quantum computers.) In 2019, NIST allowed 26 proposals for the second round of the NIST PQC Standardization Process, among which 12 were based on lattices.

The most famous computational problem in lattices is the *shortest vector problem (SVP)* that asks us to find a non-zero shortest vector in a given lattice. Its hardness ensures the security of lattice-based cryptography. The Darmstadt SVP Challenge [2] is a recognized venue for testing algorithms for solving SVPs; it publicly lists sample bases of dimensions from 40 up to 200. It is a contest of finding shorter vectors, not necessarily the shortest one. Specifically, any non-zero lattice vector whose length is shorter than $(1.05 \cdot \mathrm{GH}(L))$ can be submitted for each lattice L, where $\mathrm{GH}(L)$ is the expectation of the length of the non-zero shortest vector in L, which is given by

$$\mathrm{GH}(L) := \nu_n^{-\frac{1}{n}} \mathrm{vol}(L)^{\frac{1}{n}} \sim \sqrt{\frac{n}{2\pi e}} \mathrm{vol}(L)^{\frac{1}{n}}, \qquad (0.1)$$

where ν_n denotes the volume of the unit ball in $\mathbb{R}^n$.

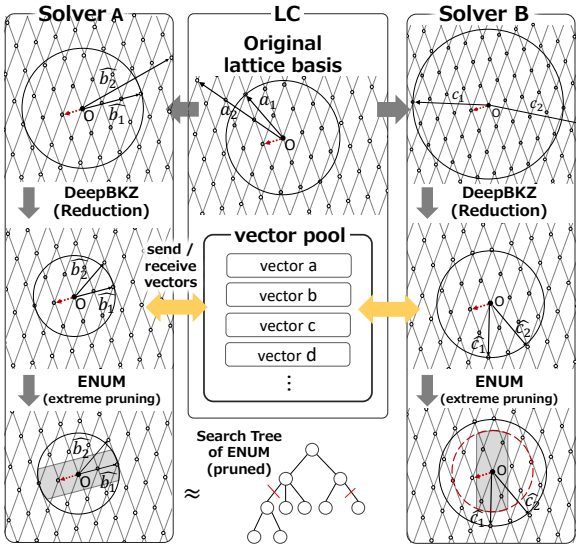

***Figure 1:** Overview of our SVP solver*

There are two main algorithms to find a non-zero shortest lattice vector; *Sieve* and *ENUM*. Both of the algorithms perform an exhaustive search of all the short lattice vectors, whose number is exponential in the lattice dimension. In the SVP Challenge, most of the high-dimensional records have been achieved by a variant of the sieve algorithm. The sieve algorithm searches for the shortest vector by repeatedly storing short differences between the short lattice vectors. A high-dimensional SVP instance requires numerous vectors to be stocked. Specifically, it requires a memory that is exponential in the dimension of the input lattice. Hence, it is highly difficult to satisfy memory requirements for solving high-dimensional SVPs using the sieve algorithm, even by increasing the number of processes. In contrast, ENUM is asymptotically slower than the sieve algorithm, but its space-complexity is polynomial in the lattice dimension. Therefore, ENUM is more suited for solving SVP in higher dimensions. In particular, it is essential for cryptanalysis of lattice-based cryptography, since it uses very high dimensions such as 256 and 512 for cryptographic security. In this project, we develop a new SVP solver called the ***Ma**ssively **P**arallel Solver for **SVP*** (MAP-SVP), which is suitable for large-scale parallelization.

In our paper[3], we demonstrated the scalability and performance of an experimental implementation of MAP-SVP through numerical experiments on SVP instances of up to 127 dimensions. The MAP-SVP is the first practical asynchronous distributed-memory

solver of SVP. We succeed in employing $103,680$ cores in our experiment. To the best of our knowledge, this is the largest massive-scale experiment for solving SVP. In MAP-SVP, multiple processes independently execute two SVP solving algorithms, *ENUM* and *DeepBKZ*, which have a small memory footprint. The memory complexity of each process is $O(n^2)$ with respect to dimension n of SVP, and in our numerical experiments, the memory usage is less than 0.013 GB per process for even a 155-dimensional SVP instance. Therefore, we can execute the MAP-SVP for high-dimensional SVPs and obtain the shortest vector. The experimental MAP-SVP is implemented by the specialized Ubiquity Generator (UG) framework [4] with the *parallelDispatch* function. The UG is a generic framework to parallelize branch-and-bound based solvers and has achieved large-scale MPI parallelism with 80,000 cores [5]. Owing to parallelDispatch of the UG, we can stably run a massive number of processes sharing information asynchronously with low communication overhead. In addition, we extend parallelDispatch by implementing the vector pooling feature, which allows each process to receive short vectors found by other processes as needed. We tackled instances of the SVP Challenge using our experimental parallel application and achieved new records for 104, 111, 121 and 127 dimensions.

The overview of the MAP-SVP for a two-dimensional SVP is shown in Fig. 1. Our system is composed of a management process, called as LoadCoordinator (abbreviated to LC throughout this document), and multiple Solvers. We provide a lattice basis B as an SVP instance to the LC, which then randomizes and distributes it to each Solver. In Fig. 1, the solid arrows, points, and circles represent the lattice basis, lattice, and depth-first search space of ENUM, respectively. The radius of the circle is the length of the shortest vector in the current bases. Each Solver executes the DeepBKZ and ENUM algorithms while sharing short vectors via the *vector pool* managed by the LC. The DeepBKZ algorithm modifies the basis vector so that each vector is shortened. We apply DeepBKZ as a preprocessing for ENUM, to reduce its search space. We can set the radius of the search space of each Solver as that of the smallest solver, as depicted by the dashed red circle for Solver B in Fig. 1. This operation does not impair the optimality of the entire system. Subsequently, each Solver uses a technique called *extreme pruning*, to further reduce the search space in ENUM. Because the input lattice basis is randomized, the reduced search spaces for all the Solvers are also fundamentally different. In Fig. 1, the gray area and the dashed red arrow represent the reduced search space and the shortest vector, respectively. The MAP-SVP prunes the search tree of each Solver according to the theoretically computed probability of the shortest vector lying within a search sub-tree. Therefore, as the number of Solvers increases, a more aggressive pruning can be applied to reduce the computation time of each Solver.

The UG framework is a software framework to parallelize branch-and-bound based solvers. However, the algorithm used to solve SVP in this project is **not branch-and-bound**. This means that the experimental implementation uses work-arounds. For a clean implementation, the UG framework itself needs to be refactored so that it can handle non branch-and-bound based solvers. We call the refactored UG, *the generalized UG*. Here, the branch-and-based is separated within the framework. Based on the knowledge obtained by the experimental implementation, we will redesign MAP-SVP and build a new solver on top of the generalized UG.

WWW

https://www.zib.de/members/shinano

More Information

[1] The National Institute of Standards and Technology (NIST), *Post-quantum cryptography*.

[2] M. Schneider, N. Gama, P. Baumann, and L. Nobach, *SVP challenge (2010)*, http://latticechallenge.org/svp-challenge.

[3] N. Tateiwa, Y. Shinano, , S. Nakamura, A. Yoshida, S. Kaji, M. Yasuda, and K. Fujisawa, *Massive parallelization for finding shortest lattice vectors based on ubiquity generator framework*, in the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC20) to be held from 15-20 November 2020 in Atlanta, GA, USA, (2020).

[4] *UG: Ubiquity Generator framework*, http://ug.zib.de/.

[5] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, T. Koch, and M. Winkler, *Solving open mip instances with parascip on supercomputers using up to 80,000 cores*, in 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2016, pp. 770–779.

Project Partners

IMI(Kyushu University), RIKEN R-CCS

Funding

Forschungscampus Modal