CSE 206A: Lattice Algorithms and Applications Spring 2007

Lecture 7: SVP, CVP and minimum distance

Lecturer: Daniele Micciancio *Scribe: Stephen Checkoway*

In this lecture, we begin studying the relationships between lattice problems. The two we will focus on are the Shortest Vector Problem, and the related inhomogeneous Closest Vector Problem, both in their *search* and *optimization* versions. We begin with the search versions.

Definition 1 *The approximate Shortest Vector Problem* SVP_γ *is, on input a basis* $\mathbf{B}$ *with rank*$\mathbf{B} = n$*, to find a vector* $\mathbf{x} \in \mathcal{L}(\mathbf{B}) - \{0\}$ *such that* $\|\mathbf{x}\| \leq \gamma(n)\lambda_1(\mathbf{B})$.

Using LLL, we can solve SVP_γ for $\gamma = 2^{\Theta(n)}$.

Definition 2 *The approximate Closest Vector Problem* CVP_γ *is, on input a basis* $\mathbf{B}$ *with rank*$\mathbf{B} = n$ *and* $\mathbf{t} \in \mathbb{R}^m$*, to find a vector* $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{x} - \mathbf{t}\| \leq \gamma(n) \min_{\mathbf{y} \in \mathcal{L}(\mathbf{B})} \|\mathbf{y} - \mathbf{t}\|$.

Note that if $\mathbf{t}$ is not in the span of $\mathbf{B}$, then we can project $\mathbf{t}$ onto span($\mathbf{B}$). Call the projection $\mathbf{t}'$. If $\mathbf{y} \in \mathcal{L}(\mathbf{B})$ is the vector closest to $\mathbf{t}'$, then $\mathbf{y}$ is the vector closest to $\mathbf{t}$. If $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ is the vector given by CVP_γ, then $\|\mathbf{t}' - \mathbf{x}\|^2 \leq \gamma^2 \|\mathbf{t}' - \mathbf{y}\|^2$ and since $\mathbf{t}'$ is the orthogonal projection of $\mathbf{t}$, $\|\mathbf{t} - \mathbf{x}\|^2 = \|\mathbf{t} - \mathbf{t}'\|^2 + \|\mathbf{t}' - \mathbf{x}\|^2$ and $\|\mathbf{t} - \mathbf{y}\|^2 = \|\mathbf{t} - \mathbf{t}'\|^2 + \|\mathbf{t}' - \mathbf{y}\|^2$. Then,

$$\begin{aligned} \|\mathbf{x} - \mathbf{t}\|^2 &= \|\mathbf{t} - \mathbf{t}'\|^2 + \|\mathbf{t}' - \mathbf{x}\|^2 \\ &\leq \gamma^2 \|\mathbf{t}' - \mathbf{y}\|^2 + \|\mathbf{t} - \mathbf{t}'\|^2 \\ &\leq \gamma^2 (\|\mathbf{t}' - \mathbf{y}\|^2 + \|\mathbf{t} - \mathbf{t}'\|^2) \\ &= \gamma^2 \|\mathbf{t} - \mathbf{y}\|^2. \end{aligned}$$

In addition to the search problems, we can consider the corresponding optimization problems.

Definition 3 *The approximate optimization Shortest Vector Problem* OPTSVP_γ *is given a basis* $\mathbf{B}$*, output a value in the range* $[\lambda_1, \gamma\lambda_1]$.

Definition 4 *The approximate optimization Closest Vector Problem* OPTCVP_γ *is given a basis* $\mathbf{B}$ *and a target vector* $\mathbf{t} \in \mathbb{R}^m$*, output a value in the range* $[dist(t, \mathcal{L}(\mathbf{B})), \gamma \cdot dist(t, \mathcal{L}(\mathbf{B}))]$.

Note that we can change the ranges in the two optimization problems by a multiplicative factor without altering the problem.

We can now consider complexity-theoretic relationships between these problems. We first consider two easy reductions, given in the following two theorems.

Theorem 5 *Given an oracle for* SVP_γ*, we can solve* OPTSVP_γ*.*

Proof On input $\mathbf{B}$, run $\text{SVP}_\gamma(\mathbf{B})$ and get a vector $\mathbf{x}$. We know that $\lambda_1 \leq \|\mathbf{x}\| \leq \gamma\lambda_1$ so output $\|\mathbf{x}\|$. ■

Theorem 6 *Given an oracle for* CVP_γ*, we can solve* OPTCVP_γ*.*

Proof On input $(\mathbf{B}, \mathbf{t})$, run $\text{CVP}_\gamma(\mathbf{B}, \mathbf{t})$ and get a lattice vector $\mathbf{x}$. Since $\mathbf{x}$ is within $\gamma \cdot \text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$ of $\mathbf{t}$, output $\|\mathbf{x} - \mathbf{t}\|$. ■

Next, we consider the relationship between SVP and CVP. The obvious mapping $\mathbf{B} \mapsto (\mathbf{B}, \mathbf{0})$ reducing SVP to CVP does not work since CVP will return $\mathbf{0}$ which is not a solution to SVP. Nevertheless, we can perform the reduction.

Theorem 7 *Given an oracle for* CVP_γ*, we can solve* SVP_γ*. Given an oracle for* OPTCVP_γ*, we can solve* OPTSVP_γ*.*

Proof If $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$, let $\mathbf{B}^{(i)} = [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, 2\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n]$. For each $1 \leq i \leq n$, call $\text{CVP}_\gamma(\mathbf{B}^{(i)}, \mathbf{b}_i)$ and let $\mathbf{x}_i$ be the returned vector. Select from $\{\mathbf{x}_i - \mathbf{b}_i \mid 1 \leq i \leq n\}$ the vector with minimum length and output it as the shortest vector.

For each i, $\mathbf{x}_i - \mathbf{b}_i \in \mathcal{L}(\mathbf{B})$ because $\mathbf{b}_i$ is a lattice vector—in fact, it is a basis vector—and $\mathbf{x}_i$ is in the sublattice $\mathcal{L}(\mathbf{B}^{(i)}) \subset \mathcal{L}(\mathbf{B})$. In addition $\mathbf{x}_i - \mathbf{b}_i \neq \mathbf{0}$ because $\mathbf{b}_i \notin \mathcal{L}(\mathbf{B}^{(i)})$.

Let $\mathbf{v} = \sum_{i=1}^n a_i\mathbf{b}_i$ be the shortest vector in $\mathcal{L}(\mathbf{b})$—i.e., $\|\mathbf{v}\| = \lambda_1$. One of the a_i is odd since otherwise $\frac{1}{2}\mathbf{v} = \sum_{i=1}^n \frac{1}{2}a_i\mathbf{b}_i \in \mathcal{L}(\mathbf{B})$ and $\|\frac{1}{2}\mathbf{v}\| = \lambda_1/2$. Fix i such that a_i is odd and write $a_i = 2a_i' - 1$. Then, $\|a_1\mathbf{b}_1 + \cdots + 2a_i'\mathbf{b}_i - \mathbf{b}_i + \cdots + a_n\mathbf{b}_n\| = \|\mathbf{v}\| = \lambda_1$. Since $a_1\mathbf{b}_1 + \cdots + 2a_i'\mathbf{b}_i + \cdots + a_n\mathbf{b}_n \in \mathcal{L}(\mathbf{B}^{(i)})$, $\text{dist}(\mathbf{b}_i, \mathcal{L}(\mathbf{B}^{(i)})) = \lambda_1$. Therefore, $\|\mathbf{x}_i - \mathbf{b}_i\| \leq \gamma\lambda_1$.

Using the same method, we get a reduction $\text{OPTSVP}_\gamma \rightarrow \text{OPTCVP}_\gamma$. The difference is that instead of getting vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, we get distances $d_1, d_2, \ldots, d_n$ and we output the minimum distance. The analysis is the same. ■

The next theorem shows that we can reduce CVP, in an exact sense to OPTCVP which is a partial converse to Theorem 6.

Theorem 8 *Given an oracle for* OPTCVP*, we can solve* CVP*.*

Proof The main idea behind this reduction is to modify the basis for the lattice to produce sublattices, making the lattice very sparse while preserving the distance of the target from the lattice. We can assume, without loss of generality, that both the input lattice $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$ and target vector $\mathbf{t}$ have integer entries. (The general case where rational numbers are used easily reduces to the integer case by scaling both the lattice and the target by the least common multiple of all denominators occurring in the original input.)

Let $(\mathbf{B}, \mathbf{t})$ be the input CVP instance, where $\mathbf{B} \in \mathbb{Z}^{d \times n}$ and $\mathbf{t} \in \mathbb{Z}^d$. Assume also, without loss of generality, that $\mathbf{t}$ is in the linear span of $\mathcal{L}(\mathbf{B})$, so that $\text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq \sum_i \|\mathbf{b}_i\|$. Make a first call OPTCVP$(\mathbf{B}, \mathbf{t})$ to determine the distance $\mu = \text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$ of the optimal solution. If $\mu = 0$, then $\mathbf{t}$ is a lattice vector, and we can immediately return $\mathbf{t}$ as the solution to CVP instance $(\mathbf{B}, \mathbf{t})$. So, assume $\mu \neq 0$.

We build a sequence of equivalent CVP instances $(\mathbf{B}_i, \mathbf{t}_i)$, where $\mathbf{B}_i = 2^i\mathbf{B}$, $\mathbf{t}_i - \mathbf{t} \in \mathcal{L}(\mathbf{B})$, and $\text{dist}(\mathbf{t}_i, \mathcal{L}(\mathbf{B}_i)) = \mu$. In particular, if $\mathbf{x}_i \in \mathcal{L}(\mathbf{B}_i)$ is the solution to CVP instance $(\mathbf{B}_i, \mathbf{t}_i)$, then $\mathbf{x} = \mathbf{x}_i - (\mathbf{t}_i - \mathbf{t})$ is a solution to the original input instance, because $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ and $\|\mathbf{x} - \mathbf{t}\| = \|\mathbf{x}_i - \mathbf{t}_i\| = \mu$.

Notice that if $i > 1 + n \log_2 \|\mathbf{b}_i\| \geq 1 + \log_2 \mu$, then CVP instance $(\mathbf{B}_i, \mathbf{t}_i)$ can be easily solved by rounding $\mathbf{t}_i$ to the closest lattice point in $2^i\mathbb{Z}^n$. This necessarily gives the right answer because $\mathcal{L}(\mathbf{B}_i) \subseteq 2^i \cdot \mathbb{Z}^d$, and all other points in $2^i \cdot \mathbb{Z}^d$ are at distance at least $2^{i-1} > \mu$ from $\mathbf{t}_i$. So, the rounding of $\mathbf{t}_i$ to the closest point in $2^i\mathbb{Z}^d$ is the only lattice point within distance μ from $\mathbf{t}_i$.

All that is left to do is to show how to compute the sequence of target vectors $\mathbf{t}_i$ (starting from $\mathbf{t}_0 = \mathbf{t}$) using the OPTCVP oracle. We show how to compute $\mathbf{t}_1$ from $\mathbf{B}_0 = \mathbf{B}$ and $\mathbf{t}_0 = \mathbf{t}$. The computation of $\mathbf{t}_{i+1}$ from $\mathbf{B}_i$ and $\mathbf{t}_i$ is analogous. We want to change $\mathbf{B}$ into $\mathbf{B}_1 = 2\mathbf{B}$, while preserving the distance of the target from the lattice. We double the basis vectors one at a time, possibly modifying the target vector to preserve the distance from the lattice. Assume we have already found a target $\mathbf{t}^i \in \mathbf{t} + \mathcal{L}(\mathbf{B})$ within distance μ from $\mathbf{B}^i = [2\mathbf{b}_1, \ldots, 2\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n]$. We want to find a target $\mathbf{t}^{i+1} \in \mathbf{t} + \mathcal{L}(\mathbf{B})$ within distance μ from $\mathbf{B}^{i+1} = [2\mathbf{b}_1, \ldots, 2\mathbf{b}_{i+1}, \mathbf{b}_{i+2}, \ldots, \mathbf{b}_n]$. Notice that at least one of the two vectors $\mathbf{t}^i$ and $\mathbf{t}^i + \mathbf{b}_{i+1}$ is at distance μ from $\mathcal{L}(\mathbf{B}^{i+1})$. Perform the oracle call OPTCVP$(\mathbf{B}^{i+1}, \mathbf{t}^i)$. If the oracle returns μ, then we leave $\mathbf{t}^{i+1} = \mathbf{t}^i$ unchanged, otherwise, we replace $\mathbf{t}^i$ with $\mathbf{t}^{i+1} = \mathbf{t}^i + \mathbf{b}_{i+1}$. It is easy to see that in both cases the distance of the new target $\mathbf{t}^{i+1}$ from $\mathcal{L}(\mathbf{B}^{i+1})$ is μ.

The final thing to check is that the numbers involved do not get too large. This is clear because at each iteration of the algorithm, we are doubling $\mathbf{b}_i$ for some i which increases the bit-length of the numbers by 1 and possibly adding $\mathbf{b}_i$ to $\mathbf{t}$ which increases the bit-length by one, in the worst case. Since all numbers remain small, this takes polynomial time. ■

The question of whether we can give a reduction $\text{CVP}_\gamma \rightarrow \text{OPTCVP}_{\gamma'}$ for $\gamma \geq \gamma' > 1$ is currently open.

In light of Theorem 8, the next logical question is can we do the same thing for SVP $\rightarrow$ OPTSVP? We can almost do it. If $\mathbf{B}'$ is out modified basis—after doubling $\mathbf{b}_1$, say—and $\lambda(\mathbf{B}) = \lambda(\mathbf{B}')$, then everything is fine and we can continue. However, if they are not equal, then we cannot shift our target vector, as we have no target. Still, if $\mathbf{v} = \sum_{i=1}^n a_i\mathbf{b}_i$ is the shortest vector in $\mathcal{L}(\mathbf{B})$, then after doubling $\mathbf{b}_i$, we can tell if a_i is even or odd. This is the idea for the next reduction in the following theorem.

Theorem 9 *Given an oracle for* OPTSVP*, we can solve* SVP*.*

Proof Let $\mathbf{B}_0 = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$ be an LLL-basis and let $\mathbf{v} = \sum_{i=1}^n a_i\mathbf{b}_i$ be the shortest

vector in $\mathcal{L}(\mathbf{B}_0)$. Using OptSvp, find $\lambda_1 = \lambda_1(\mathbf{B}_0)$. Create a new basis $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_{n-1}, 2\mathbf{b}_n]$ and run OptSvp($\mathbf{B}$). If the result from the oracle is λ_1, then a_n must be even in which case let $\mathbf{B}_1 \leftarrow \mathbf{B}$. Otherwise, a_n must be odd. We can repeatedly double b_n by creating bases $\mathbf{B}_2, \mathbf{B}_3, \cdots$ until we either doubled $\mathbf{b}_n$ a maximum number of times, or we found that the corresponding coefficient in the modified basis is odd which is to say that $a_n = 2^k \cdot c$ where c is odd and k is the number of times we doubled $\mathbf{b}_i$ without OptSvp returning something other than λ_1. We want to double $\mathbf{b}_n$ some number of times t which is still to be determined.

If we cannot do so—i.e., $k < t$—then we can perform the same doubling operation on some other basis vector $\mathbf{b}_j$ in the basis $\mathbf{B}_k$ producing new bases $\mathbf{B}_{k+1}, \mathbf{B}_{k+2}, \cdots$ until either we are unable to do so any more because the corresponding coefficient for the modified $\mathbf{b}_j$ is odd or we reach $\mathbf{B}_t$. If at this point we cannot double $\mathbf{b}_j$ any longer, we have the basis $\mathbf{B}_m = [\mathbf{b}'_1, \mathbf{b}'_2, \cdots, \mathbf{b}'_n]$ which contains the shortest vector in $\mathcal{L}(\mathbf{B}_0)$, $\mathbf{v} = \sum_{i=1}^{n} a'_i \mathbf{b}'_i$ with the coefficients a'_j and a'_n being odd. Form a new basis by adding $\mathbf{b}'_n$ to $\mathbf{b}'_j$ to get $\mathbf{B}'_m = [\mathbf{b}'_1, \cdots, \mathbf{b}'_j + \mathbf{b}'_n, \cdots, \mathbf{b}'_n]$. Now, $\mathcal{L}(\mathbf{B}'_m) = \mathcal{L}(\mathbf{B}_m)$ and we can express $\mathbf{v}$ as

$$\mathbf{v} = \sum_{\substack{1 \le i < n \\ i \ne j}} a'_i \mathbf{b}'_i + a'_j(\mathbf{b}_j + \mathbf{b}'_n) + (a'_n - a'_j)\mathbf{b}'_n.$$

Note that in $\mathbf{B}'_m$, the coefficient of the last vector is even. We can resume doubling that vector and adding it to the j-th basis vector as needed to get the sequences of bases $\mathbf{B}_0, \mathbf{B}_1, \cdots, \mathbf{B}_t$, each of which has $\mathbf{v}$ in the lattice it generates.

We now have a chain of lattices $\mathcal{L}(\mathbf{B}_0) \supset \mathcal{L}(\mathbf{B}_1) \supset \cdots \supset \mathcal{L}(\mathbf{B}_t)$ and the corresponding dual lattices $\mathcal{L}(\mathbf{B}_0)^* \subset \mathcal{L}(\mathbf{B}_1)^* \subset \cdots \subset \mathcal{L}(\mathbf{B}_t)^*$ generated by the dual bases $\mathbf{D}_0, \mathbf{D}_1, \cdots, \mathbf{D}_t$. Since doubling a vector in the lattice doubles the determinant and halves the determinant of the dual, we have $\det(\mathbf{B}_t) = 2^t \det(\mathbf{B}_0)$ and $\det(\mathbf{D}_t) = 2^{-t} \det(\mathbf{D}_0)$. We can now run LLL($\mathbf{D}_t$) to get a vector $\mathbf{w} \in \mathbf{D}_t$ such that $\|\mathbf{w}\| \le 2^n \sqrt{n} \det(\mathbf{D}_t)^{1/n} = 2^n \sqrt{n} 2^{-t/n} \det(\mathbf{D}_0)^{1/n}$. If we let $t = 2n^2$, then $\|\mathbf{w}\| \le \sqrt{n} \det(\mathbf{D}_0)^{1/n} / 2^n$.

From Minkowski's bound we have $\lambda_1 \le \sqrt{n} \det(B_0)^{1/n} \le 2^n \det(\mathbf{B}_0)^{1/n} / \sqrt{n} = 1/\|\mathbf{v}\|$. Since $\mathbf{w} \in \mathcal{L}(\mathbf{D}_t)$ separate $\mathcal{L}(\mathbf{B}_t)$ into hyperplanes each a distance $1/\|\mathbf{w}\|$ apart, the shortest vector in $\mathcal{L}(\mathbf{B}_0)$ must necessarily be in the hyperplane orthogonal to $\mathbf{w}$ passing through the origin. Take the $(n-1)$-dimensional sublattice resulting from the intersection of $\mathcal{L}(\mathbf{B}_t)$ and the orthogonal complement[1] of $\{\mathbf{v}\}$—i.e., $\Lambda = \mathcal{L}(\mathbf{B}_t) \cap \{\mathbf{v}\}^\perp$—and recursively apply the algorithm. This clearly terminates when $n = 1$ since the shortest vectors in $\mathcal{L}([\mathbf{b}])$ are $\pm\mathbf{b}$.

At each iteration of the algorithm $O(n^2)$ oracle queries are required and therefore $O(n^3)$ queries are required in total. Building each basis for the query is clearly polynomial time since we are only doubling one vector and possibly adding a vector each time. Thus, this reduction is polynomial time. ■

Notice that in the reduction from Cvp to OptCvp, we didn't have to use LLL or the dual lattice. Question: is there a simpler and more direct reduction from Svp to OptSvp in the style of the reduction given in Theorem 8?

[1] The orthogonal complement of a subset $S \subseteq \mathbb{R}^n$ is the subspace $S^\perp = \{\mathbf{x} \in \mathbb{R}^n \mid \langle x, y \rangle = 0,\ \forall y \in S\}$.

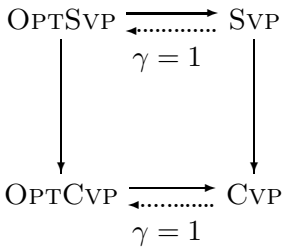

Figure 1: Relations between lattice problems.

The reductions in Theorems 5, 6, 7, 8, and 9 are collected in Figure 1.