1 The Shortest and Closest Vector Problems

Recall the definition of the approximate Shortest Vector Problem. (The exact version is obtained by taking $\gamma = 1$, which is implicit when γ is omitted.)

Definition 1.1. For $\gamma = \gamma(n) \geq 1$, the γ-approximate Shortest Vector Problem SVP_γ is: given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$, find some nonzero $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \gamma(n) \cdot \lambda_1(\mathcal{L})$.

A closely related inhomogeneous variant is the approximate Closest Vector Problem.

Definition 1.2. For $\gamma = \gamma(n) \geq 1$, the γ-approximate Closest Vector Problem CVP_γ is: given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ and a point $\mathbf{t} \in \mathbb{R}^n$, find some $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{v}\| \leq \gamma(n) \cdot \operatorname{dist}(\mathbf{t}, \mathcal{L})$. Equivalently, find an element of the lattice coset $\mathbf{t} + \mathcal{L}$ having norm at most $\gamma(n) \cdot \lambda(\mathbf{t} + \mathcal{L})$, where $\lambda(\mathbf{t} + \mathcal{L}) := \min_{\mathbf{x} \in \mathbf{t}+\mathcal{L}} \|\mathbf{x}\| = \operatorname{dist}(\mathbf{t}, \mathcal{L})$.

Above we have used the fact that $\operatorname{dist}(\mathbf{t}, \mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L}} \|\mathbf{t} - \mathbf{v}\| = \min_{\mathbf{x} \in \mathbf{t}+\mathcal{L}} \|\mathbf{x}\|$, because $\mathcal{L} = -\mathcal{L}$. The two versions of CVP are equivalent by associating each $\mathbf{v} \in \mathcal{L}$ with $\mathbf{t} - \mathbf{v} \in \mathbf{t} + \mathcal{L}$, and vice versa. Although the former version of the problem is the more "obvious" formulation, the latter version is often more convenient in algorithmic settings, so we will use it throughout these notes.

We first show that SVP_γ is no harder than CVP_γ; more specifically, given an oracle for CVP_γ we can solve SVP_γ efficiently.

Theorem 1.3. *For any $\gamma \geq 1$, we have $\mathsf{SVP}_\gamma \leq \mathsf{CVP}_\gamma$ via a Cook reduction.*

Proof. Consider the following algorithm that, given a lattice basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, and CVP oracle $\mathcal{O}$, outputs some $\mathbf{v} \in \mathcal{L} = \mathcal{L}(\mathbf{B})$:

- For each $i = 1, \ldots, n$, compute basis $\mathbf{B}_i = (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, 2\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n)$ and let $\mathbf{v}_i = \mathcal{O}(\mathbf{B}_i, \mathbf{b}_i)$.
- Output one of the $\mathbf{v}_i$ that has minimal length $\|\mathbf{v}_i\|$.

We claim that this algorithm solves SVP_γ, i.e., it returns some nonzero lattice vector of length at most $\gamma \cdot \lambda_1(\mathcal{L})$.

First observe that for each i, the lattice $\mathcal{L}_i = \mathcal{L}(\mathbf{B}_i) \subset \mathcal{L}$ consists of all those vectors in $\mathcal{L}$ whose $\mathbf{b}_i$-coordinate is even, whereas the coset $\mathbf{b}_i + \mathcal{L}_i \subset \mathcal{L}$ consists of all those whose $\mathbf{b}_i$-coordinate is odd. Therefore, $\bigcap_{i=1}^n \mathcal{L}_i = 2\mathcal{L}$ and $\mathbf{0} \notin \mathbf{b}_i + \mathcal{L}_i$ for all i, so $\lambda(\mathbf{b}_i + \mathcal{L}_i) \geq \lambda_1(\mathcal{L})$. Moreover, if $\mathbf{v} \in \mathcal{L}$ is any shortest nonzero lattice vector, then it cannot be the case that $\mathbf{v} \in \bigcap_{i=1}^n \mathcal{L}_i = 2\mathcal{L}$, otherwise $\mathbf{v}/2 \in \mathcal{L}$ would be a shorter nonzero lattice vector. Therefore, $\mathbf{v} \in \mathbf{b}_i + \mathcal{L}_i$ for at least one i, and so $\lambda(\mathbf{b}_i + \mathcal{L}_i) = \lambda_1(\mathcal{L})$ for all such i.

Now by hypothesis on $\mathcal{O}$, for every i we have $\mathbf{v}_i \in \mathbf{b}_i + \mathcal{L}_i \subset \mathcal{L}$, so $\mathbf{v}_i \neq \mathbf{0}$, and $\|\mathbf{v}_i\| \leq \gamma \cdot \lambda(\mathbf{b}_i + \mathcal{L}_i)$. Since $\lambda(\mathbf{b}_i + \mathcal{L}_i) = \lambda_1(\mathcal{L})$ for at least one i, some $\|\mathbf{v}_i\| \leq \gamma \cdot \lambda_1(\mathcal{L})$, and correctness follows. $\square$

We note that essentially the same reduction works for the *decisional* variants GapSVP_γ and GapCVP_γ of the problems, where instead of returning vectors $\mathbf{v}_i$, the oracle $\mathcal{O}$ returns yes/no answers, and the reduction outputs the logical OR of all the answers.

1.1 Algorithms for SVP and CVP

The following are some historical milestones in algorithms for SVP and CVP:

- In 1983, Kannan gave a deterministic algorithm that solves n-dimensional SVP and CVP in $n^{O(n)} = 2^{O(n \log n)}$ time and $\mathrm{poly}(n)$ space.
- In 2001, Ajtai, Kumar, and Sivakumar (AKS) gave a *randomized* "sieve" algorithm that solves SVP and $\mathsf{CVP}_{1+\varepsilon}$ (for any constant $\varepsilon > 0$) in singly exponential $2^{O(n)}$ time *and* space.
- In 2010, Micciancio and Voulgaris (MV) gave a *deterministic* algorithm that solves CVP (and hence SVP and other problems) in $2^{O(n)}$ time and space.

It is an important open question whether there exists a singly exponential-time (or better) algorithm that uses only polynomial space, or even subexponential space.

2 The Micciancio-Voulgaris Algorithm for CVP

The MV algorithm solves CVP in any n-dimensional lattice in $2^{O(n)}$ time (for simplicity, we ignore polynomial factors in the input length). It is based around the (closed) *Voronoi* cell of the lattice, which, to recall, is the set of all points in $\mathbb{R}^n$ that are as close or closer to the origin than to any other lattice point:

$$\bar{\mathcal{V}}(\mathcal{L}) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq \|\mathbf{x} - \mathbf{v}\| \ \forall\ \mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}\}.$$

We often omit the argument $\mathcal{L}$ when it is clear from context. From the definition it can be seen that for any coset $\mathbf{t} + \mathcal{L}$, the set $(\mathbf{t} + \mathcal{L}) \cap \bar{\mathcal{V}}$ consists exactly of all the shortest elements of $\mathbf{t} + \mathcal{L}$. For any lattice point $\mathbf{v}$, define the halfspace

$$\begin{aligned} H_{\mathbf{v}} &= \{\mathbf{x} : \|\mathbf{x}\| \leq \|\mathbf{x} - \mathbf{v}\|\} \\ &= \{\mathbf{x} : 2\langle \mathbf{x}, \mathbf{v}\rangle \leq \langle \mathbf{v}, \mathbf{v}\rangle\}. \end{aligned}$$

It is easy to see that $\bar{\mathcal{V}}$ is the intersection of $H_{\mathbf{v}}$ over all $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$. The minimal set V of lattice vectors such that $\bar{\mathcal{V}} = \bigcap_{\mathbf{v} \in V} H_{\mathbf{v}}$ is called the set of *Voronoi-relevant* vectors; we call them *relevant* vectors for short. It can be proved that an n-dimensional lattice has at most $2(2^n - 1) \leq 2^{n+1}$ relevant vectors.

The MV algorithm has the following high-level structure. Given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t}$ (defining a coset $\mathbf{t} + \mathcal{L}$), the algorithm works as follows:

1. Compute a description of the Voronoi cell $\bar{\mathcal{V}}(\mathcal{L})$, as a list containing all the relevant vectors $\mathbf{v} \in \mathcal{L}$, of which there are at most $2^{n+1} = 2^{O(n)}$. (The possibly exponential number of relevant vectors is the sole reason for the algorithm's exponential space complexity.)

   This "preprocessing" phase depends only on the lattice $\mathcal{L}$, not the target $\mathbf{t}$, and the result can later be reused for additional targets.

2. Use the relevant vectors to "walk," starting from $\mathbf{t}$, through a sequence of elements of the coset $\mathbf{t} + \mathcal{L}$ (by adding certain lattice vectors to the target), finally terminating with a point in $(\mathbf{t} + \mathcal{L}) \cap \bar{\mathcal{V}}$, which is a solution to the CVP instance.

   The walk proceeds in phases, where each phase starts with some $\mathbf{t}_k \in (\mathbf{t} + \mathcal{L}) \cap 2^k \cdot \bar{\mathcal{V}}$, and outputs some $\mathbf{t}_{k-1} \in (\mathbf{t} + \mathcal{L}) \cap 2^{k-1} \cdot \bar{\mathcal{V}}$. We show below that each phase takes $2^{O(n)}$ time, and by using LLL we can ensure that the initial target point $\mathbf{t}$ is in $2^{O(n)} \cdot \bar{\mathcal{V}}$, so the total number of phases is only $O(n)$. Therefore, the overall runtime of this step is $2^{O(n)}$.

2.1 The Walk

We first describe how Step 2, the "walk," is performed. The first observation is that by a scaling argument, it suffices to show how to perform the final phase of the walk, which takes some $\mathbf{t}_2 \in (\mathbf{t} + \mathcal{L}) \cap 2\bar{\mathcal{V}}$ and outputs some $\mathbf{t}_1 \in (\mathbf{t} + \mathcal{L}) \cap \bar{\mathcal{V}}$. Then all prior phases can be performed using this procedure with a suitable scaling of the lattice: since $2^k \cdot \bar{\mathcal{V}}(\mathcal{L}) = 2 \cdot \bar{\mathcal{V}}(2^{k-1}\mathcal{L})$, we can use the procedure on the lattice $2^{k-1}\mathcal{L}$ (whose relevant vectors are just scalings of $\mathcal{L}$'s relevant vectors by a 2^{k-1} factor) to go from some $\mathbf{t}_k \in (\mathbf{t} + \mathcal{L}) \cap 2^k \cdot \bar{\mathcal{V}}(\mathcal{L})$ to some $\mathbf{t}_{k-1} \in (\mathbf{t} + \mathcal{L}) \cap 2^{k-1} \cdot \bar{\mathcal{V}}(\mathcal{L})$.

The walk from $2\bar{\mathcal{V}}$ to $\bar{\mathcal{V}}$ works as follows: if our current target $\mathbf{t} \in \bar{\mathcal{V}}$ (which can be checked by testing if $\mathbf{t} \in H_{\mathbf{v}}$ for all $\mathbf{v} \in V$), then we output $\mathbf{t}$ and are done. Otherwise, we add some relevant vector to $\mathbf{t}$ and loop. The only question is, which relevant vector should be added to ensure that we make progress, and terminate within $2^{O(n)}$ iterations? Recall that if $\mathbf{t} \notin \bar{\mathcal{V}}(\mathcal{L})$, then it lies outside some halfspace $H_{\mathbf{v}}$, i.e., it violates the inequality $2\langle \mathbf{t}, \mathbf{v} \rangle \leq \langle \mathbf{v}, \mathbf{v} \rangle$. The MV algorithm greedily chooses a relevant vector $\mathbf{v}$ whose inequality is "most violated," i.e., it maximizes the ratio $2\langle \mathbf{t}, \mathbf{v} \rangle / \langle \mathbf{v}, \mathbf{v} \rangle$, and subtracts $\mathbf{v}$ from $\mathbf{t}$. Observe that for such $\mathbf{v}$, if we let $\alpha = 2\langle \mathbf{t}, \mathbf{v} \rangle / \langle \mathbf{v}, \mathbf{v} \rangle$, then α is the smallest positive real number such that $\mathbf{t} \in \alpha\bar{\mathcal{V}}(\mathcal{L})$; by assumption, we start with $\alpha \leq 2$.

Lemma 2.1. *The walk from $\mathbf{t}_2 \in (\mathbf{t} + \mathcal{L}) \cap 2\bar{\mathcal{V}}$ to $\mathbf{t}_1 \in (\mathbf{t} + \mathcal{L}) \cap \bar{\mathcal{V}}$ terminates within at most 2^n iterations.*

The above lemma follows by combining two lemmas that we state and prove next. The first shows that subtracting the chosen $\mathbf{v}$ brings the target closer to the origin, while staying within the same multiple of the Voronoi cell.

Lemma 2.2. *For any $\mathbf{t} \notin \bar{\mathcal{V}}$, if $\mathbf{v} \in \mathcal{L}$ is a relevant vector maximizing $\alpha = 2\langle \mathbf{t}, \mathbf{v} \rangle / \langle \mathbf{v}, \mathbf{v} \rangle$, then $\mathbf{t} - \mathbf{v} \in \alpha\bar{\mathcal{V}}$ and $\|\mathbf{t} - \mathbf{v}\| < \|\mathbf{t}\|$.*

Proof. As already argued, α is the smallest positive real such that $\mathbf{t} \in \alpha \cdot \bar{\mathcal{V}}$, so $\alpha > 1$ (otherwise, $\mathbf{t} \in \bar{\mathcal{V}}$). Since $\alpha = 2\langle \mathbf{t}, \mathbf{v} \rangle / \langle \mathbf{v}, \mathbf{v} \rangle$, we have

$$\|\mathbf{t}\|^2 = \langle \mathbf{t}, \mathbf{t} \rangle = \langle \mathbf{t}, \mathbf{t} \rangle - 2\langle \mathbf{t}, \mathbf{v} \rangle + \langle \mathbf{v}, \mathbf{v} \rangle = \|\mathbf{t} - \alpha\mathbf{v}\|^2.$$

Now because $\mathbf{t} \in \alpha\bar{\mathcal{V}}(\mathcal{L}) = \bar{\mathcal{V}}(\alpha\mathcal{L})$, it must be that $\mathbf{t}$ is a shortest element of $\mathbf{t} + \alpha\mathcal{L}$. Since $\alpha\mathbf{v} \in \alpha\mathcal{L}$, we also have $\mathbf{t} - \alpha\mathbf{v} \in \mathbf{t} + \alpha\mathcal{L}$. Then because $\|\mathbf{t}\| = \|\mathbf{t} - \alpha\mathbf{v}\|$, we conclude that $\mathbf{t} - \alpha\mathbf{v}$ is also a shortest element in $\mathbf{t} + \alpha\mathcal{L}$, and so $\mathbf{t} - \alpha\mathbf{v} \in \alpha\bar{\mathcal{V}}(\mathcal{L})$. Finally, by convexity of $\bar{\mathcal{V}}$ (which is an intersection of halfspaces) and the fact that $\alpha > 1$, we have $\mathbf{t} - \mathbf{v} \in \alpha \cdot \bar{\mathcal{V}}(\mathcal{L})$.

For the second claim, since $\alpha > 1$ and $\mathbf{v} \neq \mathbf{0}$ we have

$$\|\mathbf{t} - \mathbf{v}\| = \|\mathbf{t}\|^2 + \|\mathbf{v}\|^2 - 2\langle \mathbf{t}, \mathbf{v} \rangle = \|\mathbf{t}\|^2 - (\alpha - 1)\|\mathbf{v}\|^2 < \|\mathbf{t}\|^2. \qquad \square$$

The second lemma bounds the number of distinct lengths our intermediate target vectors can have.

Lemma 2.3. *For any $\mathbf{t}$, let $U = (\mathbf{t} + \mathcal{L}) \cap 2\bar{\mathcal{V}}(\mathcal{L})$. Then $|\{\|\mathbf{u}\| : \mathbf{u} \in U\}| \leq 2^n$.*

Proof. For any $\mathbf{t}' \in \mathbb{R}^n$, the points in $(\mathbf{t}' + 2\mathcal{L}) \cap 2\bar{\mathcal{V}}(\mathcal{L})$ are the shortest vectors in the coset $\mathbf{t}' + 2\mathcal{L}$, and therefore all have the same length. Since $\mathcal{L}$ is the union of 2^n distinct cosets $\mathbf{t}' + 2\mathcal{L}$ (because the quotient group $\mathcal{L}/2\mathcal{L}$ has size $\det(2\mathcal{L})/\det(\mathcal{L}) = 2^n$), we see that $\mathbf{t} + \mathcal{L}$ is also the union of 2^n cosets of $2\mathcal{L}$. By partitioning the vectors in U according to these cosets, we conclude that these vectors have at most 2^n distinct lengths overall. $\square$

We can now prove Lemma 2.1: since each step strictly decreases the length of the target while keeping it in $2\bar{\mathcal{V}}$, and the intermediate targets can take on at most 2^n distinct lengths overall, the walk must terminate within 2^n steps. This completes the analysis of the "walk" step.

2.2 Computing the Voronoi Cell

We only summarize the main ideas behind the computation of the Voronoi cell of the lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, or more precisely, of all its relevant vectors. The basic idea is the compute them in a "bottom-up" fashion, by iteratively computing the relevant vectors of the lower-rank lattices $\mathcal{L}_1 = \mathcal{L}(\mathbf{b}_1)$, $\mathcal{L}_2 = \mathcal{L}(\mathbf{b}_1, \mathbf{b}_2)$, $\mathcal{L}_3 = \mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$, etc. Clearly, the relevant vectors of $\mathcal{L}_1 = \mathcal{L}(\mathbf{b}_1)$ are trivially just $\{\pm\mathbf{b}_1\}$.

To iteratively compute the relevant vectors of $\mathcal{L}_i$, we actually use a CVP oracle for $\mathcal{L}_{i-1}$, which we can implement using the "walk" step with the already-computed relevant vectors of $\mathcal{L}_{i-1}$. The key fact we use is a characterization (due to Voronoi) of relevant vectors: $\mathbf{v} \in \mathcal{L}$ is a relevant vector of a lattice $\mathcal{L}$ if and only if $\pm\mathbf{v}$ are the *only* shortest elements of the coset $(\mathbf{v} + 2\mathcal{L})$. In particular, every relevant vector $\mathbf{v} \in \mathcal{L}$ is the (unique, up to sign) shortest element of some coset $\mathbf{t} + 2\mathcal{L}$ for $\mathbf{t} \in \mathcal{L}$. So if we find a shortest element of each of the 2^n such cosets $\mathbf{t} + 2\mathcal{L}$, we will find every relevant vector (up to sign). We might find other non-relevant vectors as well, but these do not interfere with the "walk" step and so they can be retained. (In fact, there is a way to check for relevance if desired.)

For each coset $\mathbf{t} + 2\mathcal{L}_i$, we find a shortest element using our CVP oracle for $\mathcal{L}_{i-1}$. This is done by partitioning the coset $\mathbf{t}+2\mathcal{L}_i$ according to the $\mathbf{b}_i$ coefficient, which yields several "slices" that each correspond to some coset of $2\mathcal{L}_{i-1}$. By using an LLL-reduced basis $\mathbf{B}$ we can ensure that the number of slices we need to inspect is only $2^{O(n)}$. For each of the slices we find a shortest element in the corresponding coset, and then take a shortest one overall to get a shortest element of $\mathbf{t} + 2\mathcal{L}_i$.

Overall, to find the relevant vectors of $\mathcal{L}_i$ (given those of $\mathcal{L}_{i-1}$) we need to solve CVP on 2^n cosets of $2\mathcal{L}_i$, each of which reduces to solving CVP on $2^{O(n)}$ cosets of $2\mathcal{L}_{i-1}$, each of which takes $2^{O(n)}$ time using the "walk" step. So the relevant vectors of $\mathcal{L}_i$ can be computed in $2^{O(n)}$ time overall, and hence so can the relevant vectors of $\mathcal{L} = \mathcal{L}(\mathbf{B})$.