In this lecture we present some basic computational complexity results related to lattice problems. We focus mainly on the closest lattice vector problem (CVP) and its variants.

1 Decision versus Search

Recall that in the closest vector problem we are given a lattice and a target vector (which is usually not in the lattice) and we are supposed to find the lattice point that is closest to the target point. More precisely, one can consider three variants of the CVP, depending on whether we have to actually find the closest vector, find its distance, or only decide if it is closer than some given number:

- **Decisional** CVP: Given a lattice basis $B \in \mathbb{Z}^{m\times n}$, a target vector $t \in \mathbb{Z}^m$, and a rational $r \in \mathbb{Q}$, determine whether $\mathrm{dist}(t, \mathcal{L}(B)) \leq r$ or not.
- **Optimization** CVP: Given a lattice basis $B \in \mathbb{Z}^{m\times n}$ and a target vector $t \in \mathbb{Z}^m$, find $\mathrm{dist}(t, \mathcal{L}(B))$.
- **Search** CVP: Given a lattice basis $B \in \mathbb{Z}^{m\times n}$ and a target vector $t \in \mathbb{Z}^m$, find $x \in \mathbb{Z}^n$ that minimizes $\|Bx - t\|$.

It is not difficult to see that a solution to the search variant immediately implies a solution to the optimization variant; furthermore, a solution to the optimization variant implies a solution to the decisional variant. The next lemma shows that the converse also holds, and hence all three variants are in fact polynomially equivalent.

LEMMA 1 *Search* CVP *can be solved in polynomial time given an oracle that solves decisional* CVP*.*

PROOF: Given a basis $B = (b_1, \ldots, b_n)$ and a target t, our first goal is to determine $r = \mathrm{dist}(t, \mathcal{L}(B))$ (in other words, we first solve the optimization variant of CVP). The idea is to use binary search. More precisely, define $R = \sum_{i=1}^{n} \|b_i\|$ and notice that $0 \leq r \leq R$, so R is a (rough) upper bound on the distance. Moreover, notice that r, as the l_2 distance between two integer points, must be the square root of an integer. Hence, there are only R^2 possible values for r. Therefore, a binary search for $\mathrm{dist}(t, \mathcal{L}(B))$ using the decisional CVP oracle needs at most $2 \log R$ steps, which is polynomial in the input size.

Now that we found $r = \mathrm{dist}(t, \mathcal{L}(B))$, our goal is to find the closest vector to t. Our first observation is that it is enough to find the closest lattice vector to any point of the form $t + v$ where $v \in \mathcal{L}(B)$ (since we can then easily subtract v from the answer to obtain the closest vector to t). In order to do this, we apply an iterative procedure that makes the lattice sparser and sparser. Eventually, the lattice is so sparse that we can compute the closest vector to t directly in polynomial time.

We now describe one iterative step in detail. Its input is a pair (B', t') satisfying that $\mathcal{L}(B')$ is a sublattice of the original lattice $\mathcal{L}(B)$, t' is of the form $t + v$ for some $v \in \mathcal{L}(B)$, and $\mathrm{dist}(t', \mathcal{L}(B')) = r$. Define $B'' = \{2b'_1, b'_2, \ldots, b'_n\}$ and notice that $\mathcal{L}(B'')$ is a sublattice of $\mathcal{L}(B')$ (containing 'half the points'). Call the decisional CVP oracle with the input (B'', t', r) to see if $\mathrm{dist}(t', \mathcal{L}(B'')) \leq r$. If the oracle returns YES, we continue in the next iteration with the pair $(B'', t'' = t')$. Otherwise, we continue with $(B'', t'' = t' - b'_1)$.

We claim that the output (B'', t'') satisfies the three invariants mentioned above. First, $\mathcal{L}(B'')$ is a sublattice of $\mathcal{L}(B')$ and hence also of $\mathcal{L}(B)$. Second, t'' is either t' or $t' - b'_1$, and both are of the form $t + v$ for some $v \in \mathcal{L}(B)$. Third, let us show that $\mathrm{dist}(t'', \mathcal{L}(B'')) = r$. If the oracle returns YES this is obvious. Otherwise, notice that $\mathcal{L}(B') = \mathcal{L}(B'') \cup (\mathcal{L}(B'') + b'_1)$ so it must be the case that $\mathrm{dist}(t', \mathcal{L}(B'') + b'_1) = r$. But this is equivalent to $\mathrm{dist}(t' - b'_1, \mathcal{L}(B'')) = r$ and we are done.

We now continue with the description of the algorithm. We apply the above iterative step $k = n + \log r$ times starting with (B, t). We then apply a similar process to each of the other $n - 1$ basis vectors (that is, instead of b_1). Eventually, after nk steps, we obtain a pair (B^*, t^*) where $B^* = (2^k b_1, \ldots, 2^k b_n)$. By the

invariants above, we know that $\mathrm{dist}(t^*, \mathcal{L}(B^*)) = r$ and that t^* is of the form $t + v$,$v \in \mathcal{L}(B)$. Therefore, it is enough to find the closest vector to t^*.

In order to do this, notice that

$$\lambda_1(\mathcal{L}(B^*)) \geq 2^k = 2^n \cdot r$$

since each vector in $\mathcal{L}(B^*)$ is an integer vector all of whose coordinates are a multiple of 2^k. In other words, the distance between any two vectors in $\mathcal{L}(B^*)$ is at least $2^n \cdot r$. Therefore, the second closest vector to t^* in $\mathcal{L}(B^*)$ must be of distance at least

$$2^n \cdot r - r \geq 2^{n-1} \cdot r.$$

Finally, we apply Babai's CVP approximation algorithm to t^* and B^* (see Lecture 3). Since the approximation factor is better than 2^{n-1}, this approximately closest vector must in fact be the closest vector to t^* and we are done. □

Interestingly, it is not known how to generalize the above reduction to the approximation version of CVP. To demonstrate one main difficulty, assume we are given an oracle that gives a 2-approximation to decisional CVP (more precisely, the oracle solves the promise problem described later in this lecture). In one of the iterative steps, the distance between t' and B'' becomes $2r$ but the oracle claims that this distance is still r (as it is allowed to do). We are therefore led to believe that we can continue with the pair (B'', t'). After several more iterations, the oracle 'suddenly' decides to tell us the truth: the distance between the point and the lattice is in fact $2r$. At this point, we realize we did something wrong, but it seems impossible to correct things as we don't know which step is the one that caused the distance to increase from r to $2r$. As we continue, the same thing can happen again and again, as the distance increases to $4r$, $8r$ and so on without us being able to do anything about it.

In the next section we show that decisional CVP is an **NP**-complete problem. Given this fact, the reader might now wonder: isn't the above lemma obvious? Indeed, using an oracle to decisional CVP we can solve any **NP** problem, and from there it is not difficult to solve search CVP. We argue that the above reduction has its merits. For instance, assume we had an algorithm that solves decisional CVP in time $2^{O(\sqrt{n})}$ (the best known algorithm requires $2^{O(n)}$ time). Then, since the above reduction performs queries whose dimension is the same as that of the input lattice, we would obtain an algorithm for search CVP running in time $2^{O(\sqrt{n})}$. On the other hand, a reduction among **NP** problems is likely to involve a polynomial blowup in the dimension, hence leading to an algorithm running in time 2^{n^c}.

2 Decisional CVP is NP-complete

In this part of the lecture we show that decisional CVP is **NP**-complete.

THEOREM 2 *Decisional* CVP *is* **NP***-complete.*

PROOF: First, we need to show that decisional CVP is in **NP**. For any instance (B, t, r) such that $\mathrm{dist}(t, \mathcal{L}(B)) \leq r$, let $x \in \mathcal{L}(B)$ be a vector that satisfies $\|x - t\| \leq r$. We claim that x can serve as an **NP** witness. Indeed, given x it is easy to verify that $\mathrm{dist}(t, \mathcal{L}(B)) \leq r$ (by checking that $x \in \mathcal{L}(B)$ and $\|x - t\| \leq r$). Moreover, we can represent x in a polynomial number of bits because it is an integer vector and all its entries are at most $\|t\| + r$ in absolute value.

Second, we prove that CVP is **NP**-hard by a reduction from the subset-sum problem. Recall that an instance of the subset-sum problem is given by $n + 1$ integers $a_1, a_2, \ldots, a_n, S$ and our goal is decide whether there is a set $A \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in A} a_i = S$. Given a subset-sum instance, we reduce it to the decisional CVP instance (B, t, r) given by

$$B = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ 2 & 0 & \cdots & 0 \\ 0 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 2 \end{pmatrix}_{(n+1)\times n} \qquad t = \begin{pmatrix} S \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{(n+1)\times 1} \qquad r = \sqrt{n}.$$

Clearly, this reduction can be done in polynomial time.

If $(a_1, a_2, \ldots, a_n, S)$ is a YES instance of subset-sum, then there is a set $A \subseteq \{1, \ldots, n\}$, such that $\sum_{i\in A} a_i = S$. Consider the lattice vector obtained by summing the columns of B corresponding to A. Its first coordinate is exactly S and all remaining coordinates are either 0 or 2. Hence, its distance from t is exactly $\sqrt{n}$. Conversely, assume there is a lattice point $x \in \mathcal{L}(B)$, such that $\|t - x\| \leq \sqrt{n}$. By our construction, the last n coordinates of x must be even. This already means that $\|t - x\|$ must be at least $\sqrt{n}$. In order for it to be exactly $\sqrt{n}$, it must be the case that the first coordinate of x is S, and all other coordinates are either 0 or 2. Hence, x is a sum of a subset of the columns of B, and the corresponding set of indices A satisfies $\sum_{i\in A} a_i = S$. □

The above theorem is only presented for the ℓ_2 norm. It is not difficult to generalize it to the ℓ_p norm for any $p \geq 1$, including $p = \infty$.

One might wonder whether similar methods can be used to prove the **NP**-completeness of the Shortest Vector Problem (SVP). It turns out that in the ℓ_∞ norm, similar methods work [5]. On the other hand, showing that SVP in **NP**-complete in other norms, and in particular the ℓ_2 norm, requires substantially more work [3, 4, 1].

3 SVP **versus** CVP

Our aim in this section is to show that for any $\gamma \geq 1$, finding γ-approximate solutions to SVP is not harder than finding γ-approximate solutions to CVP. At first, this might seem trivial: given an SVP instance, we can apply a CVP procedure to it with the target vector taken to be the vector 0. A moment's thought reveals that this does not work, since 0 is part of any lattice and hence the closest vector to 0 is 0 itself! Instead, we could try to make a 'hole' in the lattice and then use the CVP procedure with the hole as the target point. This hole need not be the origin; any lattice point is equally good. There is still a problem with this approach: the set of points obtained by removing one point from a lattice is not a lattice. So, instead of removing just one point, we are forced to remove a whole set of points so that the remaining set of points forms a *sublattice* of our lattice. This introduces the possibility that we remove too many points and miss some close-by vectors. As we will show below, this can be solved by trying several different sublattices, one of which is guaranteed to reveal the desired short vector.

Before proving this result, let us recall some definitions. All three variants of CVP mentioned above can be generalized to the approximation setting, but for our purposes, it is enough to consider the generalization of decisional CVP. This generalization is a promise problem known as GapCVP_γ where $\gamma \geq 1$ is some approximation factor. Recall that a promise problem is a pair $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$. We say that an algorithm solves the promise problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ if for any input instance $I \in \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$, it decides correctly whether $I \in \Pi_{\text{YES}}$ or $I \in \Pi_{\text{NO}}$. Unlike (total) decision problems, $\Pi_{\text{YES}} \cup \Pi_{\text{NO}}$ need not include all possible input strings.

DEFINITION 1 *(GapCVP_γ) The input consists of $B \in \mathbb{Z}^{m\times n}$, $t \in \mathbb{Z}^m$ and $r \in \mathbb{Q}$.*

- *In* YES *inputs, we have* $\operatorname{dist}(t, \mathcal{L}(B)) \leq r$.

- *In* NO *inputs, we have* $\mathrm{dist}(t, \mathcal{L}(B)) > \gamma \cdot r$.

Notice that GapCVP_1 is exactly decisional CVP. Let us also define the corresponding analogue of the shortest vector problem.

DEFINITION 2 *(*GapSVP_γ*) The input consists of* $B \in \mathbb{Z}^{m \times n}$ *and* $r \in \mathbb{Q}$.

- *In* YES *inputs,* $\lambda_1(\mathcal{L}(B)) \leq r$.
- *In* NO *inputs, we have* $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$.

THEOREM 3 *For any* $\gamma \geq 1$*, given access to a* GapCVP_γ *oracle, it is possible to solve* GapSVP_γ *in polynomial time.*

PROOF: We describe an algorithm that solves GapSVP_γ using a GapCVP_γ oracle. Let (B, r) be the given GapSVP_γ instance. We construct the following n GapCVP_γ instances: for $i = 1, \ldots, n$, the ith instance consists of the basis $B_i = (b_1, \ldots, b_{i-1}, 2b_i, b_{i+1}, \ldots, b_n)$, the target vector b_i, and the distance r. The algorithm applies the oracle to each of these instances. It returns YES if the oracle returns YES for at least one of the instances, and NO otherwise.

We now prove the correctness of the algorithm. Assume (B, r) is a NO instance, i.e., $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$. In other words, any nonzero $v \in \mathcal{L}(B)$ satisfies $\|v\| > \gamma \cdot r$. For each $v \in \mathcal{L}(B_i)$, we have $v - b_i \in \mathcal{L}(B)$ and $v - b_i \neq 0$. So for each $v \in \mathcal{L}(B_i)$ we see that $\|v - b_i\| > \gamma \cdot r$. Thus we conclude that the oracle answers NO on each of the n instances.

Now assume that $\lambda_1(\mathcal{L}(B)) \leq r$. Let v be the shortest lattice vector, so $\|v\| \leq r$ and write

$$v = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

for some integers $a_1, \ldots, a_n$. At least one of the a_is is odd, since otherwise $v/2$ would also be in $\mathcal{L}(B)$ in contradiction to the minimality of $\|v\|$. Let k be an index such that a_k is odd. Clearly $b_k + v \in \mathcal{L}(B_k)$ and thus the distance between b_k and $\mathcal{L}(B_k)$ is at most $\|v\| \leq r$. We conclude that there is at least one input on which the oracle returns YES, as desired. □

The above reduction has several desirable features. For instance, it is *gap-preserving*, that is, the approximation gap γ is maintained precisely. Moreover, it is *rank-preserving* in the sense that it calls the oracle with lattices whose rank is the same as that of the input lattice. These advantages and others make it very useful in a variety of other scenarios. One drawback of the reduction is that it is a Cook reduction (as opposed to the more standard Karp reduction) since it needs to call the CVP oracle several times. It turns out that based on similar ideas, one can construct a *randomized* Karp reduction, hence giving a partial answer to this drawback (see the homework). It is an open question whether there exists a *deterministic* Karp reduction from GapSVP_γ to GapCVP_γ.

References

[1] M. Ajtai. The shortest vector problem in l_2 is NP-hard for randomized reductions (extended abstract) 10-19. In *Proc. 30th ACM Symp. on Theory of Computing (STOC)*, pages 10–19. ACM, 1998.

[2] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Inform. Process. Lett.*, 71(2):55–61, 1999.

[3] S. Khot. Hardness of approximating the shortest vector problem in lattices. In *Proc. 45th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 126–135. IEEE, 2004.

[4] D. Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, Mar. 2001. Preliminary version in FOCS 1998.

[5] P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, University of Amsterdam, Department of Mathematics, Netherlands, 1981. Technical Report 8104.