CSC 2414 Lattices in Computer Science September 27, 2011

Lecture 4

Lecturer: Vinod Vaikuntanathan *Scribe: Wesley George*

Topics covered this lecture:

- $SVP \leq CVP$
- Approximating CVP: Babai's Nearest Plane Algorithm
- An Efficient Algorithm for Integer Programming in constant dimensions

1 SVP is no harder than CVP

Definition 1. *(The Closest Vector Problem) Let CVP be the problem of on input $B \in \mathbb{Z}^{m \times n}$ and $t \in \mathbb{Z}^m$ returning an $x \in \mathcal{L}(B)$ such that*

$$\forall y \in \mathcal{L}(B) \|x - t\| \leq \|y - t\|$$

Note that if $t \in \mathcal{L}(B)$, the correct answer to the CVP instance (B, t) is t. There is a natural approximation version of CVP:

Definition 2. *Let CVP_γ be the problem of on input B, t returning an $x \in \mathcal{L}(B)$ such that*

$$\forall y \in \mathcal{L}(B) \|x - t\| \leq \gamma \|y - t\|$$

This is related to (but different from) the shortest vector problem. SVP asks, given a lattice B, return $x \in \mathcal{L}(B)$ such that $\|x = Bz\|$ is minimized. CVP is the problem of given B and t, return $x \in \mathcal{L}(B)$ that minimizes $\|x - t\|$. Both languages are Turing reducible to each other (though the known reductions have different properties). We give the easier reduction:

Theorem 3 ([GMSS99])**.** $\forall \gamma \quad SVP_\gamma \leq_T CVP_\gamma$

We write $A \leq_T B$ to denote that A is *Turing-reducible* to B; A is Turing-reducible to a problem B if given an oracle for B, one can compute answers to A. In this case we will show taht if we can compute correct answers for CVP_γ, we can use this to compute the γ-shortest vector of some fixed lattice.

Note that the decision formulation of CVP is NP-hard while (the natural decision version of) SVP is in NP; thus we have a generic reduction of SVP to CVP (through, say, 3-SAT). However this reduction is inefficient in that it requires solutions to large instances of CVP in order to compute the corresponding SVP instance. In particular, this reduction gives no sense of the relationship between approximating CVP and approximating SVP. When we prove Threorem 3, we give a reduction that is dimension-preserving, approximation-preserving and deterministic. The known converse to Theorem 3 has none of these three (desirable) properties.

First we give a sense of the reduction: The shortest vector of a lattice B is the closest *non-zero* vector to the origin. As $CVP(B, 0) = 0$, the trivial reduction won't work. Instead, we will generate sublattices of B by "knocking out" certain vectors and ask for the closest vector in this sublattice to the "hole" in the lattice we've created. Let B be a lattice basis; consider sublattices spanned by bases $B^{(1)}, \ldots, B^{(n)}$ where the ith basis $B^{(i)}$ is $\{b_1, \ldots, b_{i-1}, 2b_i, b_{i+1}, \ldots, b_n\}$.

Claim 4. $\mathbf{b_i} \notin \mathcal{L}(B^{(i)})$

Proof. Were this the case, we would have $\{\alpha_i\}_{i=1}^n$ such that

$$\alpha_1(2\mathbf{b_1}) + \alpha_2\mathbf{b_2} + \ldots \alpha_n\mathbf{b_n} = \mathbf{b_1}$$

i.e. that

$$(2\alpha_1 - 1)\mathbf{b_1} + \alpha_2\mathbf{b_2} + \ldots \alpha_n\mathbf{b_n} = \mathbf{0}$$

contradicting the linear independence of $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$. □

Having noted this fact, consider the following correspondence between short vectors in $\mathcal{L}(B)$, and vectors close to $\mathbf{b_i}$ in $\mathcal{L}(B^{(i)})$:

Lemma 5. *1. Let $\mathbf{u} \in \mathcal{L}(B^{(i)})$, then $\mathbf{v} = \mathbf{u} - \mathbf{b_i} \in \mathcal{L}(B)$.*

2. Let $\mathbf{v} = \sum_{i=1}^n \alpha_i \cdot \mathbf{b_i}$ such that α_j is odd for some fixed j. Then $\mathbf{u} = \mathbf{v} - \mathbf{b_j} \in \mathcal{L}(B^{(j)})$.

This lemma follows easily the definition of $\mathcal{L}(B^{(i)})$.

Note that identification of $\mathbf{v}$ and $\mathbf{u} - \mathbf{b_i}$ means $\|\mathbf{v}\| = \|\mathbf{u} - \mathbf{b_i}\|$, i.e. that the length of $\mathbf{v}$ is equal to the distance between $\mathbf{u}$ and $\mathbf{b_i}$. Now we are prepared to prove 3:

Theorem 3. Consider the following algorithm for that on input a lattice basis $B = \{b_1, \ldots, b_n\}$ and given an oracle for CVP_γ does the following:

- Compute n bases $B^{(1)}, \ldots, B^{(n)}$ where the ith basis $B^{(i)}$ is $\{b_1, \ldots, b_{i-1}, 2b_i, b_{i+1}, \ldots, b_n\}$
- For each $i \in [n]$, let $v_i = CVP_\gamma(B^{(i)}, b_i)$
- outputs $\arg\min_{v_i}\{\|v_i + b_i\|\}$

It is clear, by Lemma 5.1 that each of $\mathbf{v_i} - \mathbf{b_i} \in \mathcal{L}(B)$. It remains to show that one of them is the shortest vector.

Now suppose $\mathbf{v}$ is the shortest vector of $\mathcal{L}(B)$. Note that if $\mathbf{v} = \sum \alpha_i \mathbf{b_i}$, one of the α_i's must be odd (otherwise $\mathbf{v}' = \mathbf{v}/2 \in \mathcal{L}(B)$, and $\|\mathbf{v}'\| = 1/2^n\|\mathbf{v}\|$, contradicting that $\mathbf{v}$ is the shortest vector of $\mathcal{L}(B)$.)

Now assume, without loss of generality, that α_1 is odd. By Lemma 2, $\mathbf{v} + \mathbf{b_1}$ is the closest vector to $\mathbf{b_1}$ in $\mathcal{L}(B^{(1)}$. So this is recovered in the oracle call $CVP(B^{(1)}, \mathbf{b_1})$, and output by our algorithm above. □

2 Approximating CVP: Babai's Nearest Plane Algorithm

Consider the following algorithm for $CVP_{\gamma=2^{n/2}}(B, \mathbf{t})$:

Begin by replacing B with a $\delta = 3/4$-L^3 reduced basis (obtained by running LLL on B).

- Project $\mathbf{t}$ to $\mathcal{H}_n = \text{Span}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$; call this projection $\mathbf{s}$
- Find $c \in \mathbb{Z}$ such that the hyperplane $c \cdot \tilde{\mathbf{b}}_n + \text{Span}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$ is closest to $\mathbf{s}$[1] [2].
- Let $\mathbf{s}' = \mathbf{s} - c \cdot \mathbf{b_n}$; call NearestPlane on $(\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}), \mathbf{s}')$ to get $\mathbf{x}'$.
- return $\mathbf{x} = \mathbf{x}' + c\mathbf{b_n}$.

There is a natural intuition to what the algorithm is doing. At the first level, the space is divided into (countably) infinite hyperplanes, each integer-valued translations of $\mathcal{H}_{n_1} = \text{Span}\{\mathbf{b}_1, ..., \mathbf{b}_{n-1}\}$. We locate the closest hyperplane to $\mathbf{s}$ to determine the $\mathbf{b}_n$th coordinate of $\mathbf{s}$, then recurse on the projection of $\mathbf{s}$ onto $\mathcal{H}_{n-1}$.

[1] i.e. c is the closest integer to $\langle \mathbf{s}, \tilde{\mathbf{b}}_n \rangle / \langle \tilde{\mathbf{b}}_n, \tilde{\mathbf{b}}_n \rangle$

[2] As usual, we use $\tilde{\mathbf{b}}_n$ to denote the nth vector in the gram-schmidt orthogonalization of $\{\mathbf{b}_i\}_{i=1}^n$

Theorem 6 ((Babai)). *The algorithm NearestPlane($B, \mathbf{t}$) is a $2^{n/2}$-approximation algorithm for CVP (i.e. a polytime algorithm for $CVP_{2^{n/2}}$.*

Proof. Let $\mathbf{x}$ be the result of running the nearest-plane algorithm on a fixed lattice with basis B and target vector $\mathbf{t}$. Let $\mathbf{y} \in \mathcal{L}(B)$ such that $\|\mathbf{t} - \mathbf{y}\| = \text{dist}(\mathcal{L}(B), \mathbf{t})$. To prove the theorem, we need to show that

$$\|\mathbf{x} - \mathbf{t}\| \leq 2^{n/2}\|\mathbf{y} - \mathbf{t}\| \tag{1}$$

We will show that

$$\|\mathbf{x} - \mathbf{s}\|^2 \leq 2^n\|\mathbf{y} - \mathbf{s}\|^2 \tag{2}$$

From here, we have (1) as

$$\begin{aligned}
\|\mathbf{x} - \mathbf{t}\|^2 &= \|\mathbf{x} - \mathbf{s}\|^2 + \|\mathbf{s} - \mathbf{t}\|^2 && (\mathbf{s} - \mathbf{t} \in \mathcal{H}_n^\perp) \\
&\leq 2^n\|\mathbf{y} - \mathbf{s}\| + \|\mathbf{s} - \mathbf{t}\|^2 && \text{by (2)} \\
&\leq 2^n(\|\mathbf{y} - \mathbf{s}\| + \|\mathbf{s} - \mathbf{t}\|^2) \\
&= 2^n\|\mathbf{y} - \mathbf{t}\|^2 && (\mathbf{s} - \mathbf{t} \in \mathcal{H}_n^\perp)
\end{aligned}$$

We will prove (2) by induction on n, but we will need the following key lemma:

Lemma 7. *Let $\mathbf{s} \in \mathcal{H}_n$. Then*

$$\|\mathbf{x} - \mathbf{s}\| \leq 2^{n/2} \cdot (1/2\|\tilde{\mathbf{b}}_n\|)$$

Proof. First observe that:

$$\|\mathbf{x} - \mathbf{s}\|^2 \leq 1/4 \sum_{i=1}^{n} \|\tilde{\mathbf{b}}_i\|^2$$

This follows immediately from the fact that the translations of $\mathbf{s}$ at each step are along an orthogonal basis: At each rounding step, $\mathbf{s}$ moves at most $1/2\|\tilde{\mathbf{b}}_i\|$ in direction $\tilde{\mathbf{b}}_i$; since the set $\{\tilde{b}_i\}$ is orthogonal, the square of the total distance translated is bounded by

$$\sum_{i=1}^{n} (1/2\|\tilde{\mathbf{b}}_i\|)^2 = 1/4 \sum_{i=1}^{n} \|\tilde{\mathbf{b}}_i\|^2$$

Now since $\{\mathbf{b}_i\}_{i=1}^n$ is a $\delta - L^3$ reduced basis, we have that

$$\begin{aligned}
\sum_i \|\tilde{\mathbf{b}}_i\|^2 &\leq 2^{(n-i)}\|\tilde{\mathbf{b}}_n\|^2 \\
&\leq 2^n\|\tilde{\mathbf{b}}_n\|^2
\end{aligned}$$

combining this with the previous claim, gives us a bound on $\|\mathbf{x} - \mathbf{s}\|$ in nicer terms:

$$\|\tilde{\mathbf{b}}_i\| \leq 2^{n/2} \cdot (1/2\|\tilde{\mathbf{b}}_n\|)$$

□

Finally we will show (2) by induction on n. For the base case, (2) holds for $n = 1$: it is easily seen that as NearestPlane computes CVP exactly for $n = 1$.

For the inductive step, consider whether $\|\mathbf{s} - \mathbf{y}\| \leq \|\tilde{\mathbf{b}}_n\|/2$. If this is not the case, we have (2) immediately thanks to Lemma 7.

But if $\|\mathbf{s} - \mathbf{y}\| \leq \|\tilde{\mathbf{b}}_n\|/2$, then $\mathbf{y} \in c\mathbf{b}_n + \mathcal{H}_{n-1}$. Specifically, if $\mathbf{y}'$ is the closest vector to $\mathbf{s}'$ in $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$, we have that

$$\mathbf{y}' + c\mathbf{b}_n = \mathbf{y}$$

Since we build $\mathbf{s}$ from $\mathbf{s}'$ by adding the same $c\mathbf{b}_n$ term, we have that

$$\|\mathbf{y}' - \mathbf{s}'\| = \|\mathbf{y} - \mathbf{s}\|$$

This means that we don't introduce any errors in this step of the algorithm. Specifically:

$$\begin{aligned}
\|\mathbf{x} - \mathbf{s}\| &= \|\mathbf{x}' - \mathbf{s}'\| \\
&\leq 2^{(n-1)/2}\|\mathbf{y}' - \mathbf{s}'\| && \text{(inductive hypothesis)} \\
&\leq 2^{(n-1)/2}\|\mathbf{y} - \mathbf{s}\| && \text{(choice of } c\text{)} \\
&\leq 2^{n/2}\|\mathbf{y} - \mathbf{s}\|
\end{aligned}$$

i.e. the (2) holds for this step.

This concludes the proof of Theorem 6. □

The lecture notes [KR04] were consulted in the preparation of this section.

3 Integer Programming in Constant Dimensions

We are concerned with the following variant of Integer Programming(IP):

Given $A \in \mathbb{Q}^{m \times n}$, determine if there are any integer points in the set $\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax \geq 0\}$

i.e. compute whether

$$\{x \in \mathbb{R}^n \mid Ax \geq 0\} \cap \mathbb{Z}^n = \emptyset$$

Note that $Ax \geq 0$ is taken to be true if *every entry* of the vector Ax is greater than 0.

IP is NP-complete, though this fact is not obvious (it is not obvious to see that it is even *in* NP). In this section we will prove the following theorem:

Theorem 8. *There is an* $2^{O(n^2)}\text{poly}(m, \|A\|, \|b\|)$ *time algorithm for the Integer Programming problem.*

We take $\|A\|$,$\|b\|$ to be the number of bits needed to describe A and b respectively. The obvious corollary of this theorem is a polytime algorithm for IP when n is constant.

What has this problem to do with lattices? We view this (feasibility version of IP) as a generalization of the CVP problem: in the case of CVP, you are given an arbitrary vector $t \in \mathbb{R}^n$, and asked for the closest lattice point; here, we are given a convex set $\mathcal{X}$ and asked for the closest lattice point. Just as in CVP where the target vector may *be* a lattice point, it may be the case that the convex set in question *contains* a lattice point (and any such lattice points are treated as having the same distance - i.e. zero - from $\mathcal{X}$).

Our starting point is a Theorem (of Loëner and John) which gives tight inner and outer approximations of a convex set (i.e. $\mathcal{X}$) by similar ellipsoids (i.e. each of their axises are in the same proportion; the outer ellipsoid can be obtained by dilating the inner ellipsoid by a factor of $n^{3/2}$ in all dimensions). By applying the right linear transformation, we can map these ellipsoids (and $\mathcal{X}$) to a pair of spheres $B(p; r)$ and $B(p; R)$ such that

$$B(p; r) \subseteq T\mathcal{X} \subseteq B(p; R)$$

Note that T preserves the relative proportions of $B(p; r)$ and $B(p; R)$, so we have the bound on R/r is preserved (i.e. $n^{3/2}$). We will, at each step, compute the exact closest vector $v \in T\mathbb{Z}^n$ to p, the center of the spheres (for which we use an algorithm for exact CVP); if $v \in T\mathcal{X}$, we are done; otherwise if we take the longest vector of $\mathcal{L}$, b_k and partition $\mathcal{L}$ into a sequence of parallel hyperplanes spanned by $\{b_i\}_{i \neq k}$. We recurse for each hyperplane (i.e. asking of the lattice of integer points on this hyperplane intersects $\mathcal{X}$ anywhere). Note that hyperplanes in question have dimension 1 less than $\mathcal{L}$, so we are reducing the size of the problem with each recursive call, to a depth of at most n. One of the recursive calls will return a point if and only if $\mathcal{X} \cap \mathbb{Z}^n \neq \emptyset$.

The point is that if v is not in $\mathcal{X}$, then it is also not in the inner approximating sphere $B(p;r)$ giving us a lower bound on $\|v-p\|$ (i.e. r); this will turn into a certain lower bound on h the distance between our $\mathcal{L}$-partitioning parallel hyperplanes. Since at most $2R/h$ of these hyperplanes intersect our outer approximating sphere (and thus $\mathcal{X}$), we get an $O(R/r)$ bound on the number of recursive calls we need to make. Since we chose $B(p;r)$ and $B(p;R)$ such that R/r is not too large, this gives us an acceptable bound on the number of recursive calls we make (the key factor in the running time of our algorithm).

This is the spirit of the theorem; now let us be more precise. The main subroutine of our algorithm, which takes a input a convex body $\mathcal{K}$ defined by a matrix A and vector b (i.e. $\mathcal{K} = \{x \mid Ax \geq b\}$), is the following:

- Compute T, p, r, R such that $B(p;r) \subseteq T\mathcal{K} \subseteq B(p;R)$
- Compute the closest vector in $T\mathbb{Z}^n$ to p; call this vector y. If $y \in T\mathcal{K}$, return $T^{-1}(y)$; else
  - Let t_k be the column of T with the longest ℓ_2 norm. Let T' be the matrix obtained from removing t_k from T. Let $\mathcal{L}' = \mathcal{L}(T)$.
  - For $i \in \{\lceil -R/h \rceil, \ldots, \lfloor R/h \rfloor\}$
    - Recurse on $T\mathcal{K} \cap (\mathcal{L}' + i\mathbf{t}_k)$; if a y in this set is found, return $T^{-1}(y)$.
- report that $\mathcal{K} = \emptyset$

The algorithm for IP is this subroutine invoked on $A, 0^n$ (where A is the set of constraints defining the integer program). We can represent, at each stage of our algorithm, the convex body in question as a (matrix,vector) pair (A, b). Note that it is easy to take intersections of such sets with hyperplanes (if $\mathbf{z}, c$ are the normal and offset of the hyperplane in question, just add the rows $\mathbf{z}, -\mathbf{z}$ to A, and entries $c, -c$ to b).

We will prove this algorithm runs in the stated time bound. We are granted that we can efficiently compute T,p,r and R. Let $\mathcal{L}$ be the lattice spanned by the columns of T. If the CVP call on $p, \mathcal{L}$ returns $y \in T\mathcal{X}$, we're done. We want to argue that if $y \notin T\mathcal{X}$, then $\{b_i\}$, the basis for T, contains a long vector b_k. To this end, we start with the following easy claim:

Claim 9. *Let B be a lattice.* $\sup_{x \in \mathbb{R}^n} \inf_{y \in \mathcal{L}(B)} \|x - y\| \leq 1/2 \sum \|b_i\|$.

Proof. Consider all the points that are closer to the origin than any other $y \in \mathcal{L}(B)$. We claim this is the set

$$\mathcal{S} = \{Bx \| x \in (-1/2, 1/2)^n\}$$

i.e. a translation of the fundamental parallelopiped. So it suffices to consider the maximum length of a vector in $\mathcal{S}$. The result follows immediately by the triangle inequality. □

If the closest vector to p in T is not in $\mathcal{K}$, we know it is not in $B(p;r)$, i.e. the inner approximation of $\mathcal{X}$. And so we have

$$r \leq \inf_{y \in \mathcal{L}(B)} \|p - y\| \leq 1/2 \sum \|b_i\| \leq n/2 \max \|b_i\|$$

If b_k is the vector of maximum length, we have that

$$\|b_k\| > 2r/n \tag{3}$$

Recall the following claim (question 1 of assignment 2 question 1)

Claim 10. *Suppose $\{b_1, \ldots, b_n\}$ is L^3 reduced. Fix k and partition $\mathcal{L} = \mathsf{Span}(b_1, \ldots, b_n)$ into the parallel hyperplanes*

$$\mathcal{L}' + ib_k$$

For $i \in \mathbb{N}$, $\mathcal{L}' = \mathsf{Span}(b_1, \ldots, b_{k-1}, b_{k+1}, \ldots, b_n)$. Let h be the distane between any pair of adjacent hyperplanes. Then

$$h \geq 2^{-n(n-1)/4} \|b_k\|$$

Combining this with (3) gives

$$h \geq 2^{-n(n-1)4+1} r/n \tag{4}$$

Since there are clearly $2R/h$ hyperplanes intersecting $B(p; R)$. We want to show that this number is not too large.

Claim 11.

$$2R/h = 2^{O(n^2)}$$

This is immediate from (4) and our starting guarantee that $R/r = O(n^{3/2})$. So if n is constant, so is $2R/h$, and so is the number of recursive calls we make at any stage. This bounds the total number of nodes in the tree of recursive calls at $2^{O(n^3)}$. What remains is to show that we do no more than polynomial work at each stage. The only thing to really account for is the CVP call we make: there are algorithms that run in $n^{O(n)}$ time, which is constant for our purposes here. This concludes the proof of Theorem 8.

References

[GMSS99] Goldreich, Micciancio, Safra, and Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71:55–61, July 1999.

[KR04] Eyal Kaplan and Oded Regev. Cvp algorithm. http://www.cs.tau.ac.il/~odedr/goto.php?name=ln_babai_cvp&link=teaching/lattices_fall_2004/ln/cvp.pdf, 2004.