CSC2421 - Approximation Algorithms
Lecture #9

Lecture: Charles Rackoff
Scribe Notes by: Qingqing Zhou, Zheng Zeng

11th March 2003

1 The L^3 Algorithm

Algorithm 1 The over-simplified version of L^3 algorithm from the text (corrected)

```
Given basis b_1, b_2, ..., b_n
while basis b_1, b_2, ..., b_n is not Gauss reduced do
(1) for 1 ≤ i ≤ n
      if |u_{i+1,i}| > 1/2 then
         b_{i+1} ← b_{i+1} − m b_i, where
         m = round(u_{i+1,i});
(2) Pick an i (if one exists) such that
      ||b_i(i)|| > (2/√3) ||b_{i+1}(i)||, and interchange (b_i, b_{i+1});
```

For $j \leq i$ we denote by $\mathbf{b}_i(j)$ the component of $\mathbf{b}_i$ orthogonal to $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{j-1}\}$, and thus we have $\mathbf{b}_i(i) = \mathbf{b}_i^*$. (Refer to the notes of lecture#8 for the details about these notations.)

In the last lecture, we showed that if and when the L^3 algorithm halts:

1. The fact that Step (1) of the algorithm makes no changes implies that $||\mathbf{b}_1|| \leq 2^{\frac{n-1}{2}} \cdot OPT$;

2. Step (1) doesn't change $\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*$;

Now we show that the simplified L^3 algorithm halts in a polynomial number of stages. A more complicated version is needed to prove that L^3 is a truly polynomial time algorithm – that is, that in addition the integers don't get too big. For example, see the algorithm in the L^3 paper [LLL82].

Claim 1.1 *The L^3 Algorithm halts in poly(N) number of rounds where N is the original input size in bits.*

To show this, we will prove that step (2) of the algorithm can't be performed too often.

Now we define the "potential" function

$$pot(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) \stackrel{def}{=} \Pi_{i=1}^{n} \|\mathbf{b}_i^*\|^{n-i}$$

We have the following lemma which establishes a lower bound on function $pot(\cdot)$.

Lemma 1.2 $pot(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) \geq 1$

In fact, if we can show that $\|\mathbf{b}_1^*\|\|\mathbf{b}_2^*\| \ldots \|\mathbf{b}_k^*\| \geq 1$ for all k, then by expanding the expression of function $pot(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ we can immediately get the lemma.

Proof Let B_k be the $k \times k$ matrix where row i is $(\|u_{i,1}\mathbf{b}_1^*\|, \|u_{i,2}\mathbf{b}_2^*\|, \ldots, \|u_{i,i}\mathbf{b}_i^*\|, 0, 0, \ldots, 0)$. Since $u_{i,i} = 1$, row i is actually $(\|u_{i,1}\mathbf{b}_1^*\|, \|u_{i,2}\mathbf{b}_2^*\|, \ldots, \|\mathbf{b}_i^*\|, 0, 0, \ldots, 0)$. Notice that $\mathbf{b}_i = \sum_{j=1}^{i} u_{ij}\mathbf{b}_j^*$, so row i is just the coordinates of $\mathbf{b}_i$ in the orthogonal coordinate system determined by $\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_k^*$.

By this construction, matrix B_k is lower diagonal and $det(B_k) = \|\mathbf{b}_1^*\|\|\mathbf{b}_2^*\| \ldots \|\mathbf{b}_k^*\|$. And we have $det(B_k^\top) = \|\mathbf{b}_1^*\|\|\mathbf{b}_2^*\| \ldots \|\mathbf{b}_k^*\|$. This gives us $det(B_k B_k^\top) = \|\mathbf{b}_1^*\|^2\|\mathbf{b}_2^*\|^2 \ldots \|\mathbf{b}_k^*\|^2 > 0$.

Notice that (i, j) entry of matrix $B_k B_k^\top = \mathbf{b}_i \cdot \mathbf{b}_j \in \mathbf{Z}$. So $det(B_k B_k^\top) \in \mathbf{Z}$. Thus $\|\mathbf{b}_1^*\|^2\|\mathbf{b}_2^*\|^2 \ldots \|\mathbf{b}_k^*\|^2 \geq 1$ which implies $\|\mathbf{b}_1^*\|\|\mathbf{b}_2^*\| \ldots \|\mathbf{b}_k^*\| \geq 1$. ▌

Now let's see what happens to function $pot(\cdot)$ when we interchange $\mathbf{b}_i, \mathbf{b}_{i+1}$.

Suppose before the interchange, the basis is $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ and after the interchange, the basis becomes $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n$.

It is easy to see $\mathbf{c}_1^* = \mathbf{b}_1^*, \ldots, \mathbf{c}_{i-1}^* = \mathbf{b}_{i-1}^*, \mathbf{c}_{i+2}^* = \mathbf{b}_{i+2}^*, \ldots, \mathbf{c}_n^* = \mathbf{b}_n^*$. Note that $\mathbf{b}_i^* = \mathbf{b}_i(i) = \mathbf{c}_{i+1}(i)$ and $\mathbf{c}_i^* = \mathbf{c}_i(i) = \mathbf{b}_{i+1}(i)$.

We name the function $pot(\cdot)$ before the interchange as "old pot" and "new pot" after the interchange. We have

$$\frac{\text{old pot}}{\text{new pot}} = \frac{||\mathbf{b}_i^*||^{n-i}||\mathbf{b}_{i+1}^*||^{n-i-1}}{||\mathbf{c}_i^*||^{n-i}||\mathbf{c}_{i+1}^*||^{n-i-1}} \tag{1}$$

Notice that for the angle θ between vector $\mathbf{c}_i^*$ and $\mathbf{b}_i^*$,

$$\sin\theta = \frac{||\mathbf{b}_{i+1}^*||}{||\mathbf{c}_i^*||} = \frac{||\mathbf{c}_{i+1}^*||}{||\mathbf{b}_i^*||} \tag{2}$$

Substitute equation (2) into equation (1), and we have:

$$\begin{aligned}
\frac{\text{old pot}}{\text{new pot}} &= \left(\frac{||\mathbf{b}_{i+1}^*||}{||\mathbf{c}_i^*||}\right)^{n-i-1} \left(\frac{||\mathbf{b}_i^*||}{||\mathbf{c}_{i+1}^*||}\right)^{n-i-1} \frac{||\mathbf{b}_i^*||}{||\mathbf{c}_i^*||} \\
&= \frac{||\mathbf{b}_i(i)||}{||\mathbf{b}_{i+1}(i)||} \\
&> \frac{2}{\sqrt{3}}
\end{aligned}$$

So we have

1. Original potential value is at most exponential size in N (original input size in bits);
2. The new potential function is at most $\frac{\sqrt{3}}{2}(< 1)$ of the old potential function after step (2) in each round;
3. The potential function is always at least 1. (Lemma 1.2)

This shows that the number of step (2) performed by the algorithm is bounded above by a polynomial in N. Thus in this sense the algorithm is in polynomial time. Claim 1.1 stands.

Note:

1. The L^3 algorithm usually works better than our worst-case theory predicts;
2. For any fixed dimensionality $n = c$, there exists a polynomial algorithm that solves SVP.

2 Applications

There are many applications of the L^3 algorithm. For instance, factorization of polynomials with rational coefficients [LLL82]. We will now see how to use L^3 to solve certain types of average case subset-sum problem(SS).

2.1 Subset-sum problem(SS)

Definition The subset-sum problem is:
given non-negative integers $a_1, a_2, \ldots, a_n$ and t , find a subset $S \subseteq \{1, 2, \ldots, n\}$, such that $t = \sum_{i \in S} a_i$, if one exists.

Theorem 2.1 *The SS problem is NP-hard. (cc. [GJ79])*

An average case version of the SS problem is: Given instances from a particular distribution where there is guaranteed to be a solution, find a solution.

Let parameter m be the number of bits in each a_i. Consider the distribution where we choose $X_1, X_2, \ldots, X_n$ randomly from $\{0, 1\}$ and $a_1, a_2, \ldots, a_n$ randomly from $\{0, 1, \ldots, 2^m - 1\}$, and output $a_1, a_2, \ldots, a_n, \sum_{i=1}^{n} X_i a_i = t$. We ask if given such a randomly generated $a_1, a_2, \ldots, a_n, t$, is there a probabilistic polynomial time algorithm which finds $X_1^{'}, X_2^{'}, \ldots, X_n^{'} \in \{0, 1\}$ such that $\sum_{i=1}^{n} X_i^{'} a_i = t$, with significant probability?

It turns out that the hardest case for m is $m = n$.

A reasonable **conjecture:** if $n < m \leq 2n$, then for every probabilistic polynomial time algorithm, the success probability $< \frac{1}{n^c}$ for each c (and sufficiently large n) .

We now see an application of the subset-sum problem to constructing a pseudo-random number generator.

Theorem 2.2 *[IN96] If the above conjecture is true for m, then the following is a pseudo-random generator.*

$$\begin{array}{rcl} \text{input} & & \text{output} \\ a_1, \ldots, a_n, X_1, \ldots, X_n & \vdash & a_1, \ldots, a_n, (\sum_{i=1}^{n} X_i a_i) \bmod 2^m \\ \text{length} : mn + n \text{ bits} & & mn + m \text{ bits} \end{array}$$

For $m = n^2$, we can use the L^3 algorithm to break average-case subset sum [LO83]. (This can be improved to $m = cn^2$ for any fixed $c > 0$.) From now on, assume $m = n^2$.

Given $a_1, a_2, \ldots, a_n$ and t, consider the $n+1$ dimensional lattice with a basis that consists of the rows of

$$B = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \\ \mathbf{b}_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & -a_1 \\ 0 & 1 & & 0 & 0 & -a_2 \\ \vdots & & & & & \\ 0 & 0 & & 0 & 1 & -a_n \\ 0 & 0 & \ldots & 0 & 0 & t \end{pmatrix}$$

where $a_1, a_2, \ldots, a_n$ are chosen randomly from $\{0, 1, \ldots, 2^m - 1\}$, $X_1, X_2, \ldots, X_n$ are chosen randomly from $\{0, 1\}$, and we let $t = \sum_{i=1}^n X_i a_i$.

Let $\mathcal{L} = \mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{n+1})$. A short vector $\mathbf{w}$ in $\mathcal{L}$ is $\sum_{i=1}^n X_i \mathbf{b}_i + \mathbf{b}_{n+1} = (X_1, X_2, \ldots, X_n, 0)$ which has 0 in the last position and 0 or 1 in each of other positions. So $\|\mathbf{w}\| \leq \sqrt{n}$. Other short vectors have the form $d \cdot \mathbf{w}$ for d a small integer in absolute value.

Claim 2.3 *With high probability, the L^3 algorithm will return a multiple of $\mathbf{w}$: $(\lambda X_1, \lambda X_2, \ldots, \lambda X_n, 0)$ for some λ.*
Note that from such a vector, we can easily learn $(X_1, \ldots, X_n)$.

This follows from Claim 2.4, since any output of the L^3 algorithm is a vector with length $\leq 2^{((n+1)-1)/2} \cdot OPT \leq \|\mathbf{w}\| \cdot 2^{n/2} \leq \sqrt{n} \cdot 2^{n/2}$.

Claim 2.4 *For $m = n^2$, with high probability, all vectors in $\mathcal{L}$ of length $\leq \sqrt{n} \cdot 2^{n/2}$ are multiples of $\mathbf{w}$.*

Proof Fix $X_1, X_2, \ldots, X_n$ to be, for example,

$$\underbrace{11\ldots1}_{u}\underbrace{00\ldots0}_{n-u}$$

The a_i are chosen randomly, and we have $t = a_1 + a_2 + \ldots + a_u$.

Consider a lattice vector $(k_1, k_2, \ldots, k_n, \alpha)$ with length$\leq \sqrt{n} \cdot 2^{n/2}$.

Let p be the probability there exists $k_1, k_2, \ldots, k_n, \lambda$ such that $|k_i| \leq \sqrt{n} \cdot 2^{n/2}$ for all i and $(k_1, k_2, \ldots, k_n) \neq (\lambda X_1, \lambda X_2, \ldots, \lambda X_n)$ and $|\alpha| \leq \sqrt{n} \cdot 2^{n/2}$ where

$$\begin{aligned} \alpha &= \lambda t - \sum_{i=1}^n k_i a_i. \\ &= (\lambda - k_1)a_1 + (\lambda - k_2)a_2 + \ldots + (\lambda - k_u)a_u - k_{u+1}a_{u+1} - \ldots - k_n a_n \end{aligned}$$

We now calculate an upper bound on the probability p.

We consider the case when $a_1, a_2, \ldots, a_n$ are chosen randomly from $\{0, 1, \ldots, 2^{n^2} - 1\}$.

Assume $a_1 \geq 2^{n^2}/n$; this happens with probability $\geq 1 - \frac{1}{n}$.

Imagine we had $\lambda - k_1 > \sqrt{n} \cdot 2^{n/2} \cdot n \cdot n$. Then we have $(\lambda - k_1)a_1 > \sqrt{n} \cdot 2^{n/2} \cdot 2^{n^2} \cdot n$ and $(\lambda - k_i) > 0$ for $2 \leq i \leq u$. Notice that $\sum_{i=u+1}^{n} k_i a_i \leq \sqrt{n} \cdot 2^{n/2} \cdot 2^{n^2} \cdot (n-1) < (\lambda - k_1)a_1$, so

$$\begin{aligned}
\alpha &\geq (\lambda - k_1)a_1 - \sum_{i=u+1}^{n} k_i a_i \\
&> \sqrt{n} \cdot 2^{n/2} \cdot 2^{n^2} \cdot (n - (n-1)) \\
&> \sqrt{n} \cdot 2^{n/2}
\end{aligned}$$

Contradiction. So $\lambda \leq \sqrt{n} \cdot 2^{n/2} \cdot (n^2 + 1)$. Similarly, $|\lambda| \leq \sqrt{n} \cdot 2^{n/2} \cdot (n^2 + 1)$.

Fix a particular $k_1, k_2, \ldots, k_n, \lambda$, such that $(k_1, k_2, \ldots, k_u) \neq (\lambda, \lambda, \ldots, \lambda)$ and fix β. What is the probability that $\alpha = \beta$, over the choice of $a_1, a_2, \ldots, a_n$?

Say for example $\lambda \neq k_2$. Choose first $a_1, a_3, \ldots, a_n$ and β. The probability over choice of a_2 that $(\lambda - k_2)a_2 = \beta - (\lambda - k_1)a_1 - (\lambda - k_3)a_3 - \ldots - (\lambda - k_u)a_u + k_{u+1}a_{u+1} + \ldots + k_n a_n$ is $\leq 1/2^m = 1/2^{n^2}$.

How many $k_1, k_2, \ldots, k_n$, λ and β are there that are small enough?

Notice that $|k_i| \leq \sqrt{n} \cdot 2^{n/2}$ for all i, $(1 \leq i \leq n)$, $|\lambda| \leq \sqrt{n} \cdot 2^{n/2} \cdot (n^2 + 1)$ and $|\beta| \leq \sqrt{n} \cdot 2^{n/2}$. The number of combinations is at most $((\sqrt{n} \cdot 2^{n/2} \cdot (n^2 + 1))^{(n+2)} \leq 2^{n^2(1/2+\varepsilon)}$. So we have

$$\begin{aligned}
p &\leq \frac{2^{n^2(1/2+\varepsilon)}}{2^{n^2}} \cdot (1 - \frac{1}{n}) + \frac{1}{n} \\
&\leq \frac{2^{n^2(1/2+\varepsilon)}}{2^{n^2}} + \frac{1}{n} \\
&\approx \frac{1}{n}
\end{aligned}$$

This completes the proof. ▌

The density of SS is defined as $\frac{n}{m}$. So these are "low density" SS problems.

References

[LLL82] A. K. Lenstra, H. W. Lenstra, J. L. Lovasz, "Factoring polynomials with rational coefficients". *Mathematische Ann.*, 261:pp513-534, 1982.

[GJ79] M. R. Garey and D. S. Johnson. "Computers and Intractability: A Guide to the Theory of NP Completeness". *Freeman*, 1979.

[IN96] R. Impagliazzo and M. Naor. "Efficient cryptographic schemes provably as secure as subset-sum". *Journal of Cryptology*, 9:pp199-216, 1996.

[LO83] J. C. Lagarias and A. M. Odlyzko. "Solving Low-Density Subset Sum Problems". *FOCS*, pp.1-10, 1983.