Simultaneous Diophantine approximation (SDA): Lagarias '82

* Diophantine Approx: Given a real $x$, approximate it as $a/b$ (with integers $a, b$), such that $|x - a/b| \leq \frac{1}{2b}$

* Simultaneous D.A: Given many $x_i$'s, approximate all with the same approximate common denominator $\varepsilon$ s.t $\frac{a_i}{b} - x_i | \leq \frac{\varepsilon}{2b}$

* If the $x_i$'s are rational then there is an exact solution $x_i = \frac{a_i}{b}$ ($b$ is the LCM of all the denominators) but we often want to approximate with denominator $\leq b$.

* Definition: Let $\varepsilon \sum_i x_i = \frac{a_i}{b}$, $a_i, b \in \mathbb{Z}$, $i = 1, 2, ..., n$ be an instance of SDA. We say that an approximate common denominator $q$ is of quality $(\varepsilon, \delta)$ is $q \neq 0$, $q \in \mathbb{Z}$ and the following conditions hold:
  1. $q \leq \varepsilon b$
  2. $q x_i$ is within $\frac{\delta}{q}$ of an integer for all $i = 1, 2, ..., n$

(namely $\exists \varepsilon_i$ such that $|\frac{a_i}{b} - \frac{\varepsilon_i}{q}| \leq \frac{\delta}{q}$)

* Consider the lattice spanned by columns of $B(n \times n)$ (where $c$ is a parameter).

\[ B = \begin{pmatrix} a_1 & b \\ a_2 & b \\ \vdots \\ a_n & b \end{pmatrix} \]

Note: $\det(B) = b^n \cdot c$, so by Minkowsky we know that $\Lambda(B)$ has nonzero vectors of length

\[ \leq \sqrt{n+1} \cdot \det(B)^{\frac{1}{n+1}} = \sqrt{n+1} \cdot b \cdot \left(\frac{c}{b}\right)^{\frac{1}{n+1}} \]

Heuristically we expect $\Lambda(B)$ to have exponentially many vectors (in $n$) of length $\leq \text{poly}(n) \cdot b \cdot \left(\frac{c}{b}\right)^{\frac{1}{n+1}}$, and for "random" SDA instance we expect no vectors of length $\leq b \cdot \left(\frac{c}{b}\right)^{\frac{1}{n+1}} / \text{poly}(n)$

* Claim: From any vector in $\Lambda(B)$ of length $0 \leq \ell \leq b$ we can compute efficiently an approx common denominator $q$ of quality $(\varepsilon, \delta)$ with $\varepsilon \leq \ell / \text{poly}(n)$, $\delta \leq 2\ell / b$.
Proof: Let \( \mathbf{x} \in \Lambda(B) \), \( \mathbf{0} \neq \| \mathbf{x} \| \leq b \), and we write \( \mathbf{x} = B \mathbf{z} \) with
\[
\mathbf{z} = \begin{pmatrix}
c \\
a_1 & b \\
\vdots & \ddots \\
a_n & \cdots & b
\end{pmatrix} = \begin{pmatrix}
\mathbf{e}_q \\
q_{a_1} - P_1 \\
\vdots \\
q_{a_n} - P_n, b
\end{pmatrix} = b \begin{pmatrix}
c_q/b \\
q_{a_1}/b - P_1 \\
\vdots \\
q_{a_n}/b - P_n
\end{pmatrix}.
\]

Note that we cannot have \( q = 0 \), or else we get \( \| \mathbf{x} \| \geq \max \{ \| \mathbf{p} \| : \mathbf{p} \in B \} b \) and since the \( p_i \)'s are integers, and \( \| \mathbf{x} \| \leq b \) then it must be that all the \( p_i \)'s are zero (which is a contradiction).

Hence we have \( c \leq \ell \) so \( \epsilon = \frac{c}{b} \leq \frac{\ell}{b} \).

Also, for all \( i \), the distance from \( \frac{a_i}{b} \) to the nearest integer is at most \( |\frac{a_i}{b} - P_i| \leq \frac{\ell}{b} \), namely \( \delta \leq \frac{\ell}{b} \).

**Note:** If we want to set \( \epsilon = \delta \) then we need \( c = \frac{\ell}{2} \), but usually \( (\epsilon, \delta) \) come from the application and then we set \( c = \frac{\ell}{2} \).

**Claim:** From any approx.-common-denominator which is \( (\epsilon, \delta) \)-good, we can compute a vector \( B \mathbf{z} \in \Lambda(B) \) of length at most
\[
\| \mathbf{z} \| \leq 2 \cdot \frac{\epsilon}{\delta} + \sqrt{(c - \epsilon)^2 + \lambda \cdot \delta^2} \leq b \cdot (c - \epsilon + \sqrt{n} \cdot \delta).
\]

Proof is essentially the same as above.

* Hence there is basically 1-1 correspondence between "good" approx.-common-denominators and "short" vectors in \( \Lambda(B) \).

**Note:** This is an easy example of how lattice-based algorithms work: We look for ways to cast the problem at hand as consisting of linear relations with integer coefficients and finding small solutions.
Using SOA to solve approximate GCD

We have parameters $p < n < \gamma$. We are given as input

\[ \mathcal{E}(w) = \{ \mathbf{w} : \mathbf{p}, \mathbf{r}, \mathbf{i} \} \mid i = 0, 1, \ldots, n \rceil \wedge \mathbf{p} \in [2^{n+1}, 2^{\gamma-1}], \mathbf{p} \text{ odd} \]

\[ \mathbf{q} \in [2^{\gamma-1}, 2^{\gamma-1}], \mathbf{r} \in [-2^p, 2^p-1]. \]

We can assume wlog. that $w_0 > w_i$ for all $i \Rightarrow q_0 > q_i$.

**Claim:** $q_0$ is an approximate common denominator of quality $(E, 0)$ with $E \leq 2^{-n+1}$ and $d \leq 2^{p-n+3}$

**Proof:**

\[ E = \frac{q_0}{w_0} = \frac{q_0}{q_0, p, r_0} = \frac{q_0}{q_0, p, r_0} \leq \frac{1}{p - 1} \leq 2^{-n+1}. \]

To bound $d$, note that

\[ q_0, w_i = \frac{q_0, p, r_0}{q_0, p, r_0} = \frac{q_0, p, r_0}{p, r_0} = \frac{q_0, p, r_0}{p, r_0} = \frac{r_i - \frac{q_0}{w_0} r_0}{p, r_0} \]

hence the distance between $q_0, \frac{w_i}{w_0}$ and the nearest integer is

\[ d = \frac{|r_i - \frac{q_0}{w_0} r_0|}{p, r_0} \leq \frac{|r_i| + |r_0|}{p - 1} \leq \frac{2^{p+1}}{2^{n-1}} = 2^{p-n+2} \]

for the lattice $B = \left( \begin{array}{c} 2^{p+1} \\ \vdots \\ 2^{p+1} \end{array} \right)$.

We therefore use parameter $E = \frac{E}{2^p} = \frac{2^{p-n+3}}{2^{n+1}} = 2^{p-n+1}$ if this was a "random instance" then we expect to find in $\Lambda(B)$ vectors of size

\[ \sim w_0 \left( \frac{2^{p+1}}{w_0} \right) \sqrt{n+1}. \]

However, the vector corresponding to $q_0$ has size $\sim \sqrt{n+1} \cdot q_0 \cdot 2^{p+1}$.

When do we expect that the vector corresponding to $q_0$ be the shortest nonzero vector in $\Lambda(B)$?

**Answer:** When $n$ is large enough so that $q_0 \cdot 2^{p+1} \ll w_0 \cdot \left( \frac{2^{p+1}}{w_0} \right) \sqrt{n+1}$.

Recall that $q_0 \sim 2^p$, $w_0 \sim 2^{\gamma-n}$, this means that we need

\[ 2^{p+1} \ll 2^{\gamma-n} \left( \frac{p+1}{\gamma-1} \right) \]

\[ \iff \gamma + p + 1 \ll \gamma + n + \frac{p+1}{\gamma-1} \]

\[ \iff n+1 \gg \frac{\gamma - p - 1}{n-\gamma} \approx \frac{\gamma}{n} \]
If we have enough samples \( n \gg \frac{\lambda}{\mu} \) the the vector corresponding to \( \eta_0 \) will be the shortest nonzero vector in \( \Lambda(\mathcal{B}) \). We can then hope that running LLL on \( \mathcal{B} \) will recover this vector, and thereby also \( \eta_0 \) (and the secret \( \rho \)).

Q2: Will this attack work?
A: Depends on the sizes of \( n \) and \( \lambda \) (recall \( \rho \approx 2^n, \lambda \approx 2^\lambda \)).

Example 1: Assume that we set \( \lambda = n^2 \), and \( n = \frac{2^\lambda}{\mu} = 2n^2 \).

- The smallest vector in \( \Lambda(\mathcal{B}) \) is the one corresponding to \( \eta_0 \) of size \( \approx \sqrt{2 \lambda - \mu}_n \cdot 2^{\lambda+n} \).
- All the vectors in \( \Lambda(\mathcal{B}) \) that are not multiples of the shortest are of size \( \approx \sqrt{2 \lambda - \mu}_n \cdot 2^{\lambda+n+(\lambda-\delta)\mu_n} \).

- Using LLL-like algorithms with approximation factor \( \approx 2^{\lambda/8} = 2^{\lambda/4} \) we can find a vector in \( \Lambda(\mathcal{B}) \) of size at most
  \[
  \sqrt{2 \lambda - \mu}_n \cdot 2^{\lambda+n} \leq \sqrt{2 \lambda - \mu}_n \cdot 2^{\lambda+n/2}
  \]
  Hence this must be a multiple of the vector corresponding to \( \eta_0 \).
  Then we can find the vector itself, and therefore \( \eta_0 \) and \( \rho \).

Example 2: Set \( \lambda = n^2 \), and still \( n = \frac{2^\lambda}{\mu} = 2n^2 \).

- Now any algorithm with approximation factor \( 2^{\lambda/8} = 2^{\lambda/4} \) will only be able to find vectors of size
  \[
  \sqrt{2 \lambda - \mu}_n \cdot 2^{\lambda+2n^2+\mu} \gg \sqrt{2 \lambda - \mu}_n \cdot 2^{\lambda+n/2}
  \]
  Hence it will almost surely find \( \eta_0 \) as the exponentially many auxiliary vectors, and not the one corresponding to \( \eta_0 \).

Tracing through the parameters, the "safe region" is \( \lambda = \omega(n^2) \).