Lattice Enumeration using Extreme Pruning

Nicolas Gama, Phong Nguyen, Oded Regev

June, 2010

This is a story of...

Something which would have taken 1.3 billion years...

...can now be done in 61 days

...at home!

Outline

1. Introduction
2. SVP, Enumeration and Pruning
3. Sketch of the analysis

Analogy

Treasure Hunt

There are 10101 doors, a Treasure is hidden according to the distribution

- 25%: behind door number 1
- 65%: behind a uniformly chosen door between 2 and 101
- 10%: behind a uniformly chosen door between 102 and 10101

Strategies

Full enumeration: open all the doors
Time required 10101, always succeeds

Analogy

Treasure Hunt

There are 10101 doors, a Treasure is hidden according to the distribution

- 25%: behind door number 1
- 65%: behind a uniformly chosen door between 2 and 101
- 10%: behind a uniformly chosen door between 102 and 10101

Strategies

Full enumeration: open all the doors
Time required 10101, always succeeds

Pruned enumeration: just go over first 101 doors.
Time required: 101; success probability 90%

Analogy

Treasure Hunt

There are 10101 doors, a Treasure is hidden according to the distribution

- 25%: behind door number 1
- 65%: behind a uniformly chosen door between 2 and 101
- 10%: behind a uniformly chosen door between 102 and 10101

Strategies

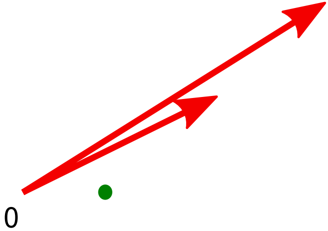
Full enumeration: open all the doors
Time required 10101, always succeeds

Pruned enumeration: just go over first 101 doors.
Time required: 101; success probability 90%

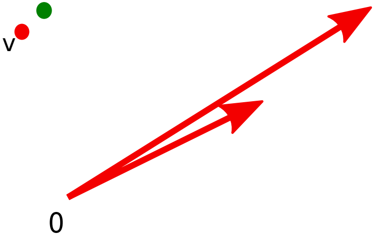
Extreme pruning: just try the first door. If not there, restart game.
Expect time to find treasure: 4

The problems

SVP: Given a lattice basis B, find the shortest non-zero vector of $L(B)$

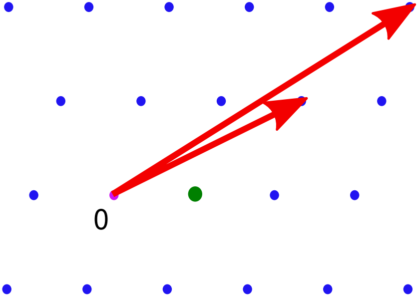


CVP: Given a lattice basis B and a target vector $\vec{v} \in \mathbb{R}^m$, find the lattice vector of $L(B)$ closest to $\vec{v}$

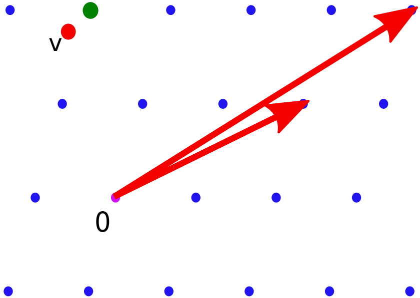


The problems

SVP: Given a lattice basis B, find the shortest non-zero vector of $L(B)$



CVP: Given a lattice basis B and a target vector $\vec{v} \in \mathbb{R}^m$, find the lattice vector of $L(B)$ closest to $\vec{v}$



Background on exact algorithms for SVP/CVP

SVP
CVP

low storage
mem: Poly
time: $2^{\Theta(n^2)}$
or $2^{\Theta(n \log n)}$

Enumeration
[Ka83]
[ScEu94]
[ScHo95]

Widely used
Subroutine for solving lattice challenges

large storage
mem: $2^{\Theta(n)}$
time: $2^{\Theta(n)}$

Sieving
[AKS01]

Voronoi
[MiVo10]

Impractical in high dimension

Our results

Exponential speed-up against enumeration

Pruning ($2^{\Theta(n^2)}$ time, negligible memory)

1. First sound analysis of pruned enumeration
2. Prove that asymptotically pruning gives exponential speedup of $2^{n/4}$
3. Main contribution: *Extreme pruning*
   Further speed-up $\approx 2^{n/4}$ vs. Basic Pruning
   Leading to an overall $\approx 2^{n/2}$ vs. Full enumeration

Experimental results

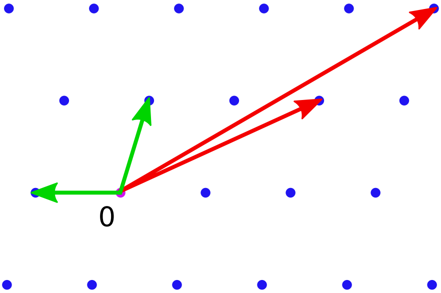
Find the shortest vector of a dense 110-dimensional lattice

Full enumeration: 1.3 billion years (estimated)

Basic pruning: 320 years (estimated)

Extreme pruning: 61 days

Enumerating vectors

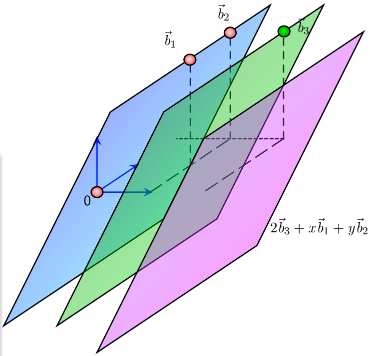


Definitions

- Lattice
- Basis
- Dimension
- Volume $(\mathrm{Volume}(L))$
- Shortest vector $(\lambda_1(L))$

Enumerate all points of a given 3D lattice of norm $\leq \sqrt{12}$

$$B = \begin{pmatrix} 1 & 1 & 2 \\ \hline 1 & 2 & 2 \\ \hline 2 & 2 & 2 \end{pmatrix}$$



Solution

- Find all $\vec{v} = u_1\vec{b}_1 + u_2\vec{b}_2 + u_3\vec{b}_3$:
  - $(u_1, u_2, u_3) \in \mathbb{Z}^3$
  - $\|\vec{v}\| \leq \sqrt{12}$
- For each "possible" u_3
  - make a recursive call

The quality of the input basis

$$B = \begin{pmatrix} 1 & 1 & 2 \\ \hline 1 & 2 & 2 \\ \hline 2 & 2 & 2 \end{pmatrix} \text{ or } C = \begin{pmatrix} 144 & 172 & 184 \\ \hline 100 & 120 & 128 \\ \hline 36 & 44 & 48 \end{pmatrix}?$$

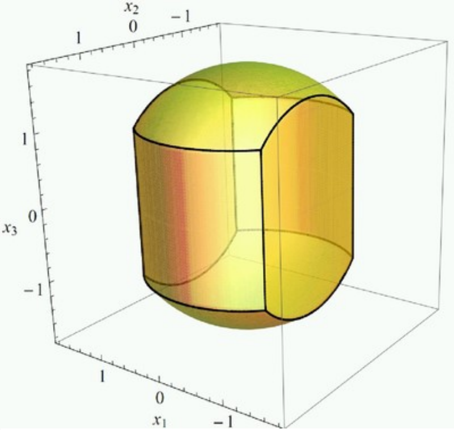
Basis reduction

The running time depends on the quality of the input basis

Enumerating in a cylinder intersection

Pruning

- Pruned enumeration puts a different norm bound for each level of the recursion
- This effectively replaces searching in a ball with searching in a *cylinder intersection*
  - $x_1^2 \le \alpha_1$
  - $x_1^2 + x_2^2 \le \alpha_2$
  - $x_1^2 + x_2^2 + x_3^2 \le \alpha_3$



The algorithm may miss the shortest vector

Caveat

- We do not explore all the possibilities any more
- On some bases, it may miss the shortest vector
- Hence success probability is lower than 1

Basis randomization

Dealing with the success probability

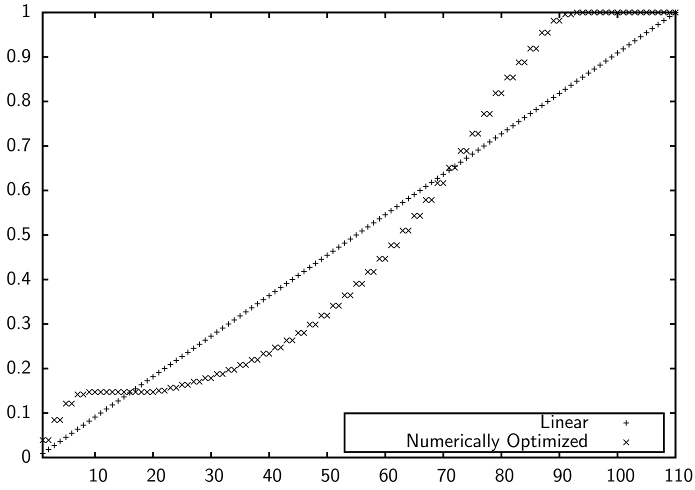
- We search the shortest vector of the lattice.
- A lattice contains a lot of "reduced" bases
- Their directions are well distributed
  - Pruning will succeed on some of them

Algorithm

Repeat the following:

1. Generate a reduced basis
2. Do pruned enumeration

The bounding function



Experimental evidence

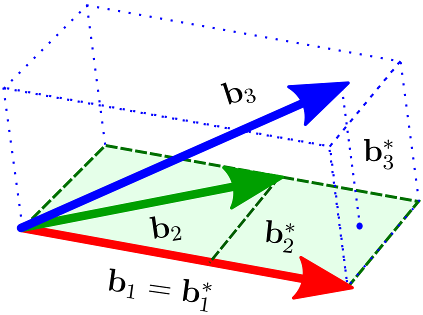
Experimental result

- 61 sequential CPU-Days to solve a 110-dim CJLOSS problem
- ≈500 **independent** runs of ≈3h
  - (45 min reduction time included)

Analysis

1. All the above running-times are predictable
2. The best bounding function can be numerically obtained

Triangular isometric representation

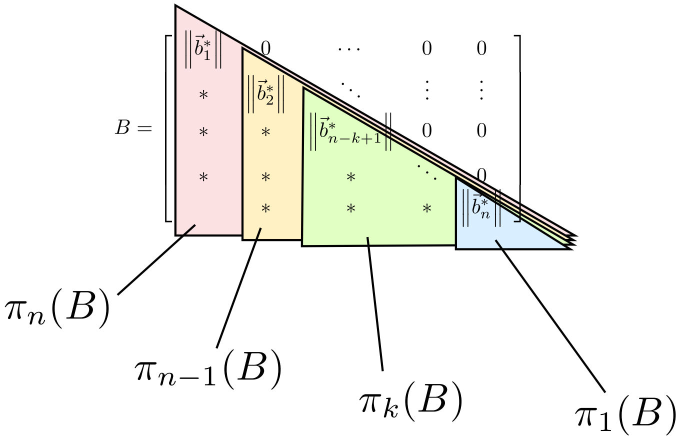


Bases are viewed up to an isometry

- (Gram Schmidt)

$$B = \begin{bmatrix} \left\|\vec{b}_1^*\right\| & 0 & \cdots & 0 \\ ? & \left\|\vec{b}_2^*\right\| & \ddots & \vdots \\ ? & ? & \ddots & 0 \\ ? & ? & ? & \left\|\vec{b}_n^*\right\| \end{bmatrix} \cdot Q$$

Projected lattices/Partial norms



Complexity analysis - Full enumeration

Complexity of depth k:

N_k = number of lattice points of $\pi_k(L)$ in $\mathrm{Ball}(\mathrm{target}, R)$

$$N_k = \mathrm{Volume}(\mathrm{Ball}_k(\cdot, R)) \cap \pi_k(L)$$

Complexity analysis - Full enumeration

Complexity of depth k:

N_k = number of lattice points of $\pi_k(L)$ in $\mathrm{Ball}(\mathrm{target}, R)$

$$N_k = \frac{\mathrm{Volume}(\mathrm{Ball}_k(\cdot, R))}{\mathrm{Volume}(\pi_k(L))}$$

(Gaussian Heuristic)

Complexity analysis - Full enumeration

Complexity of depth k:

N_k = number of lattice points of $\pi_k(L)$ in $\mathrm{Ball}(\mathrm{target}, R)$

$$N_k = \frac{\mathrm{Volume}(\mathrm{Ball}_k(\cdot, R))}{\mathrm{Volume}(\pi_k(L))}$$

(Gaussian Heuristic)

Total running time:

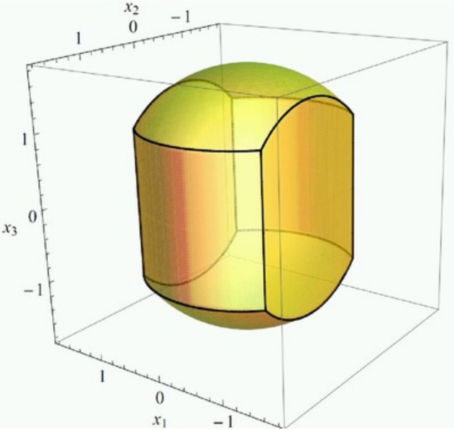
$$\text{running time} = t_{\text{reduction}} + t_{\text{node}} \cdot \sum_{k=1}^{n} N_k$$

Pruned running time

$$\text{Running time} = \frac{t_{\text{reduction}} + t_{\text{node}} \sum_{k=1}^{n} \frac{\text{Volume}(\alpha_1,\ldots,\alpha_k)}{\text{Volume}(\pi_k(L))}}{\text{Proba}_{\text{success}}}$$

Volume of a cylinder intersection

$$\|\pi_1(\vec{x})\|^2 \leq \alpha_1$$
$$\|\pi_2(\vec{x})\|^2 \leq \alpha_2$$
$$\|\pi_3(\vec{x})\|^2 \leq \alpha_3$$
$$\ldots$$
$$\|\pi_k(\vec{x})\|^2 \leq \alpha_k$$



Volume of a cylinder intersection

Closed formula

$$V_{\alpha_1,\ldots,\alpha_k} = 2^n \cdot \int_{x_1=0}^{\sqrt{\alpha_1}} \int_{x_2=0}^{\sqrt{\alpha_2 - x_1^2}} \int_{x_3=0}^{\sqrt{\alpha_3 - x_1^2 - x_2^2}} \cdots \int_{x_n=0}^{\sqrt{\alpha_n - x_1^2 - \cdots - x_{n-1}^2}} dx_1\, dx_2 \ldots dx_n$$

Particular case

- Computing this volume exactly in general seems hard
- Luckily, for bounding functions of the form:

  $$(\alpha_1, \alpha_1, \alpha_3, \alpha_3, \ldots, \alpha_{n-1}, \alpha_{n-1}),$$

  we can compute it exactly using the Dirichlet distribution.
- These exact computations already lead to very good upper and lower bounds

Success probability

What we want:

- the surface of $\text{Cylinder}(\alpha_1, \ldots, \alpha_n) \cap \text{Sphere}(\alpha_n)$

Remark

- We can still compute it precisely and quickly for

$$(\alpha_1, \alpha_1, \alpha_3, \alpha_3, \ldots, \alpha_{n-1}, \alpha_{n-1}).$$

Best bounding function

Optimizing the bounding function

1. start from the linear bounding function
2. apply perturbations, keep the best

Best bounding function

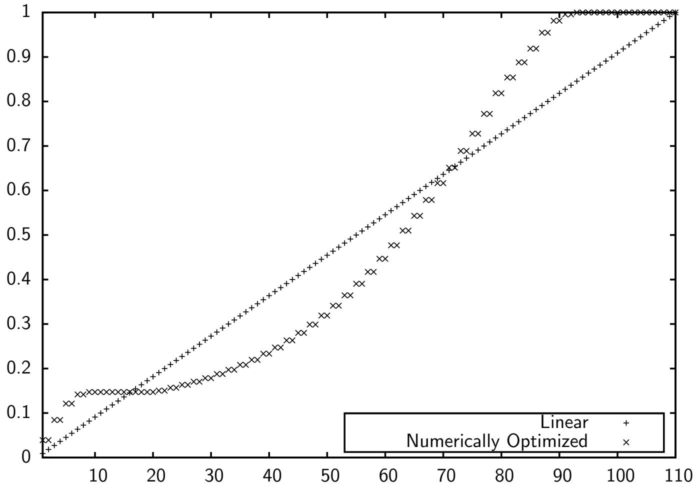
Optimizing the bounding function

1. start from the linear bounding function
2. apply perturbations, keep the best

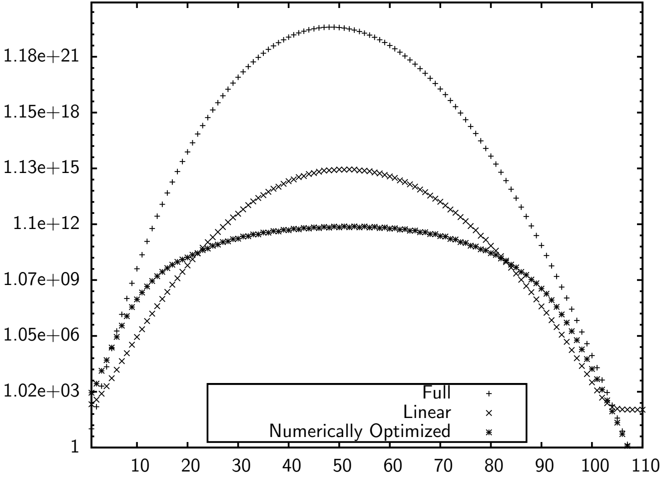
It converges!

- It converges
- The limit seems to be a global optimum
  - Whatever starting point we use!

Best bounding function



Best bounding function



Summary

Extreme Pruning ($2^{\Theta(n^2)}$ time, negligible memory)

- Exponential speed-up:
  $\approx 2^{n/2}$ vs. Full enumeration
  $\approx 2^{n/4}$ vs. all kind of high-probability pruning
- Sound geometric analysis
- Tight running-time predictions (within 1%)
- Massively parallel

Open questions

Main open questions

- Apply these ideas to improve lattice reduction algorithms? (in progress)
- Are there time-memory trade-offs?
  - *I.e.*, use more memory to improve running time

Other open questions

- Design SIMD versions?
- Prove that the numerically optimized bounding function is the best one