Fastest algorithm for the Shortest Vector Problem

Oded Regev

Courant Institute, NYU

(joint with Aggarwal, Dadush, and Stephens-Davidowitz)

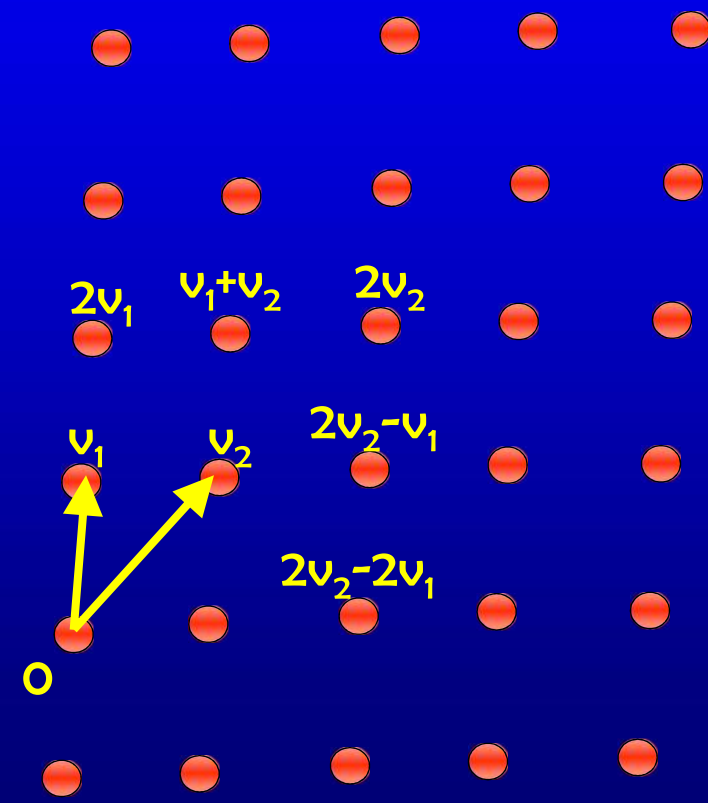
Lattices

- A lattice is a set of points

  $$L=\{a_1v_1+\ldots+a_nv_n \mid a_i \text{ integers}\}$$

  for some linearly independent vectors $v_1,\ldots,v_n$ in R^n

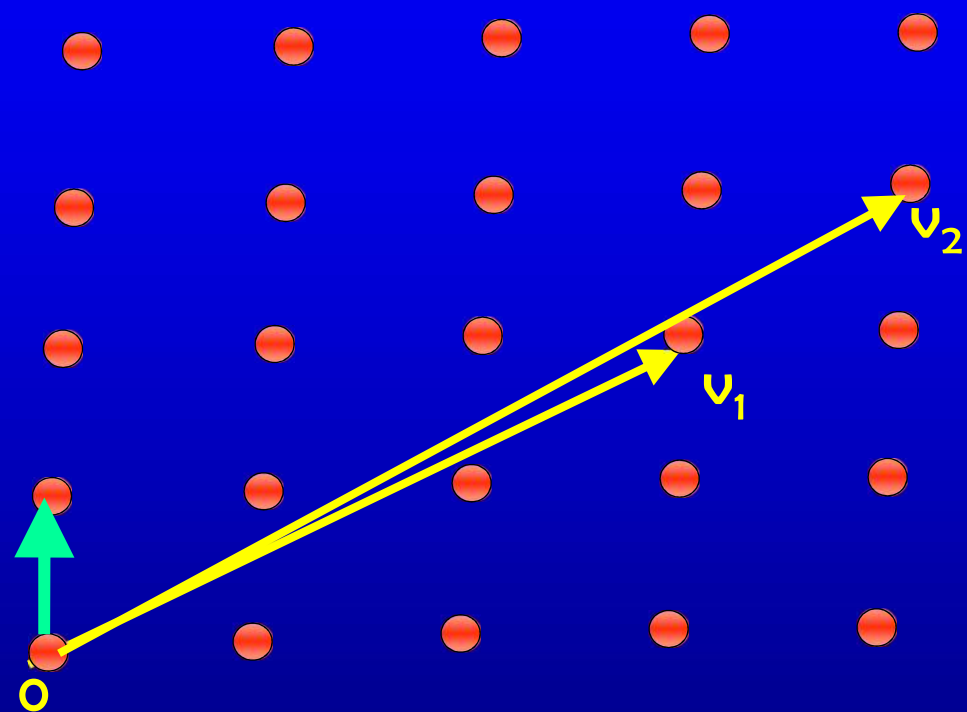
- We call $v_1,\ldots,v_n$ a basis of L



History

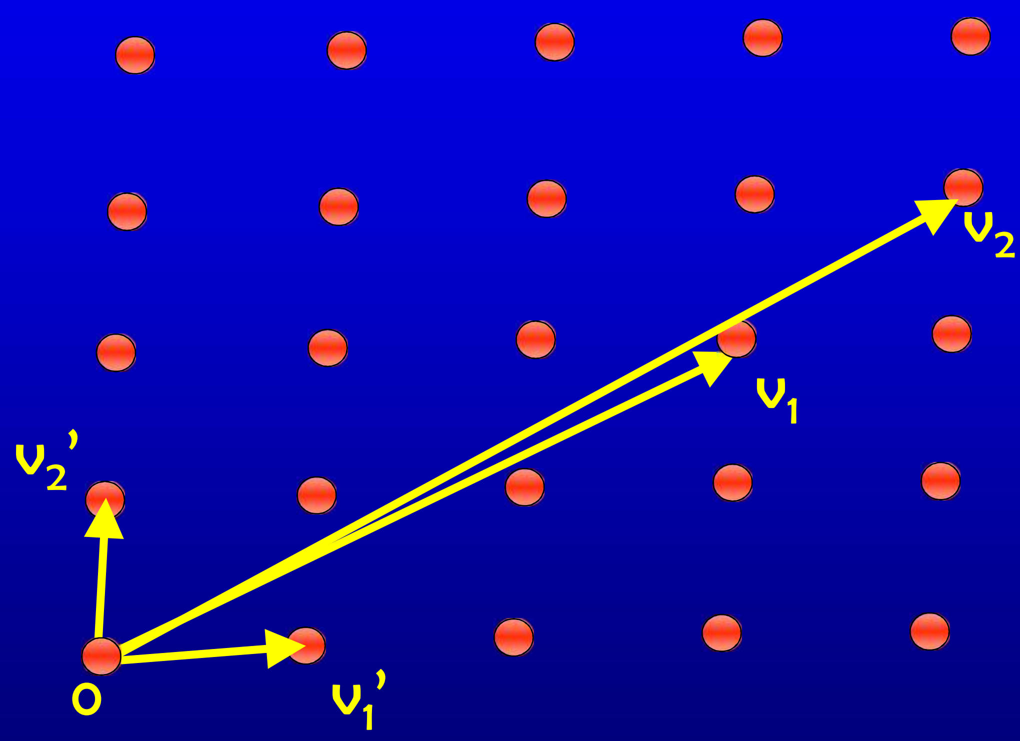
- Geometric objects with rich mathematical structure
- Considerable mathematical interest, starting from early work by Lagrange 1770, Gauss 1801, Hermite 1850, and Minkowski 1896.

Shortest Vector Problem (SVP)

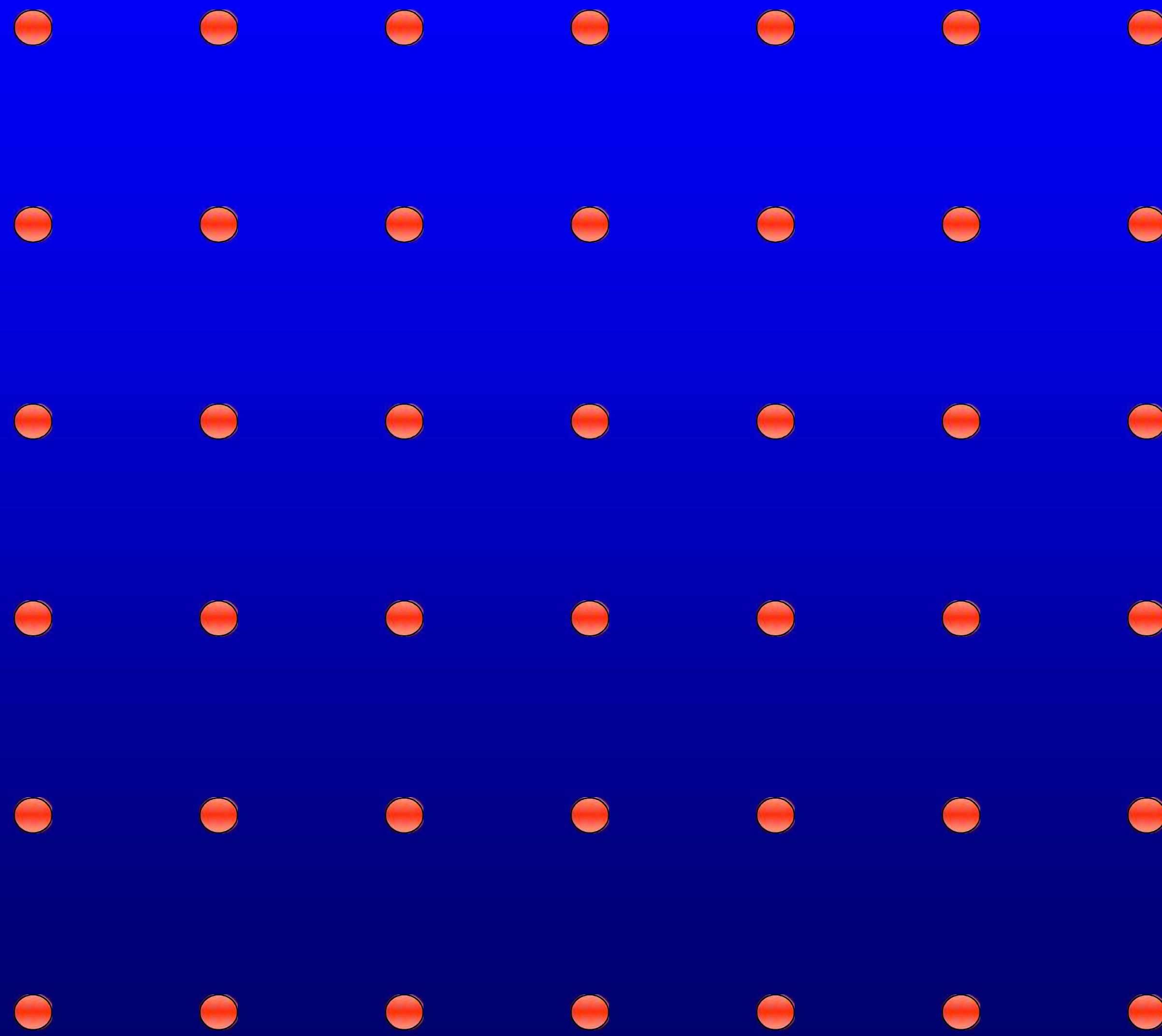


- SVP: Given a lattice, find the shortest vector
- Best known algorithm runs in time $2^{O(n)}$ [AjtaiKumarSivakumar01,...]
  - No better quantum algorithm known

Basis is not Unique



Even Rotations of Z^n seem hard!



The LLL Algorithm

[LenstraLenstraLovász82]

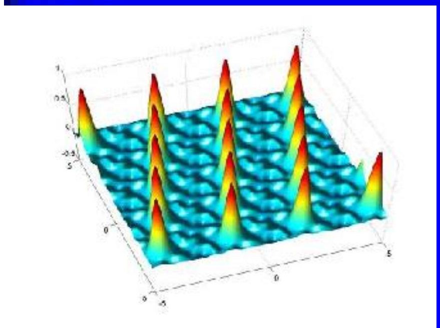
- An efficient algorithm that outputs a “somewhat short” vector in a lattice
- Applications include:
  - Solving integer programs in a fixed dimension,
  - Factoring polynomials over rationals,
  - Finding integer relations:

$$5.709975946676696\ldots \stackrel{?}{=} 4+\sqrt[3]{5}$$

  - Attacking knapsack-based cryptosystems [LagariasOdlyzko’85] and variants of RSA [Håstad’85, Coppersmith’01]

Lattices and Cryptography

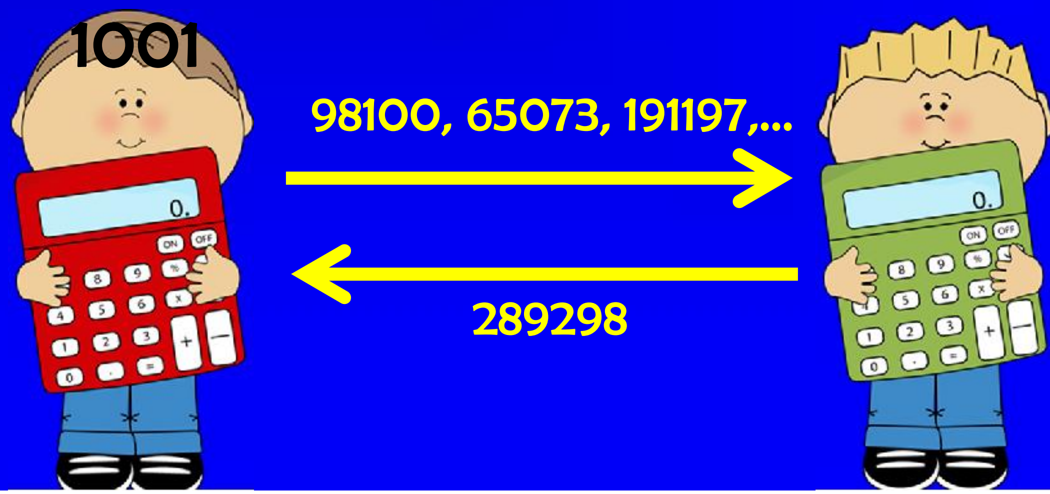
- Lattices can also be used to create cryptography
- This started with a breakthrough of Ajtai in 1996
- Cryptography based on lattices has many advantages compared with 'traditional' cryptography like RSA:
  - It has strong, mathematically proven, security
  - It is resistant to quantum computers
  - In some cases, it is much faster
  - It can do more: fully homomorphic encryption!



A Really Simple Public Key Cryptosystem

[Ajtai-Dwork 1997, Cohen 2000, R03, Levieil-Naccache 2008]

- Private key: a random odd integer s
- Public key: a list of random multiples of s plus a small positive even random number
- Encrypt 0: add random set of half of them
- Encrypt 1: same, but add 1
- Decrypt: compute remainder under division by s and check if it's odd

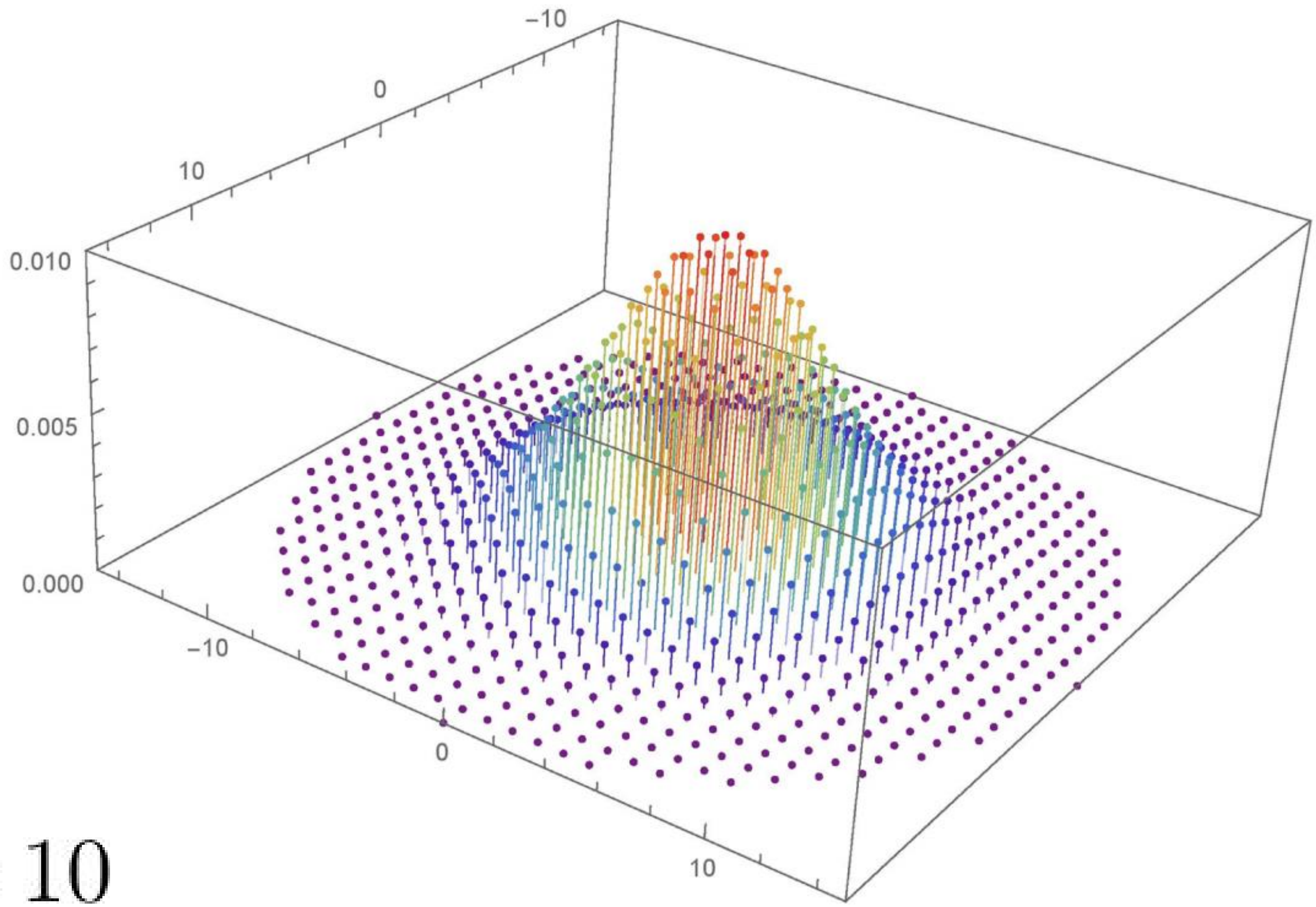


Progress on provable SVP algs

	Time
[Kan86]	$n^{O(n)}$
[AKS01]	$2^{O(n)}$
[NV08, PS09, MV10a]…	$2^{2.465n+o(n)}$
[MV10b] Det	$2^{2n+o(n)}$

Discrete Gaussian Distribution

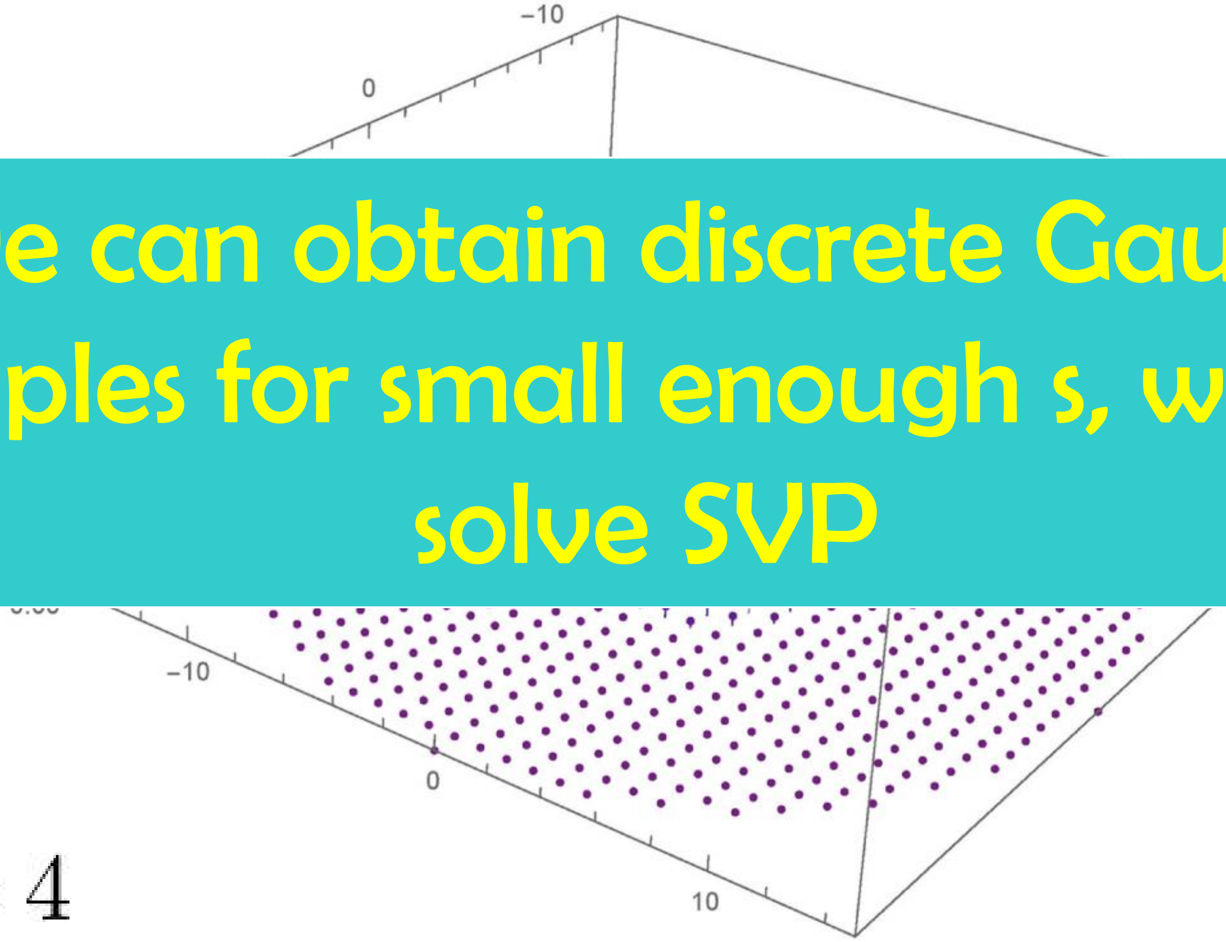
$$D_{\mathcal{L},s} := \Pr[\mathbf{y}] \propto e^{-\|\mathbf{y}\|^2/s^2}$$



$s = 10$

Discrete Gaussian Distribution

$$D_{\mathcal{L},s} := \Pr[\mathbf{y}] \propto e^{-\|\mathbf{y}\|^2/s^2}$$



If we can obtain discrete Gaussian samples for small enough s, we can solve SVP

$s = 4$

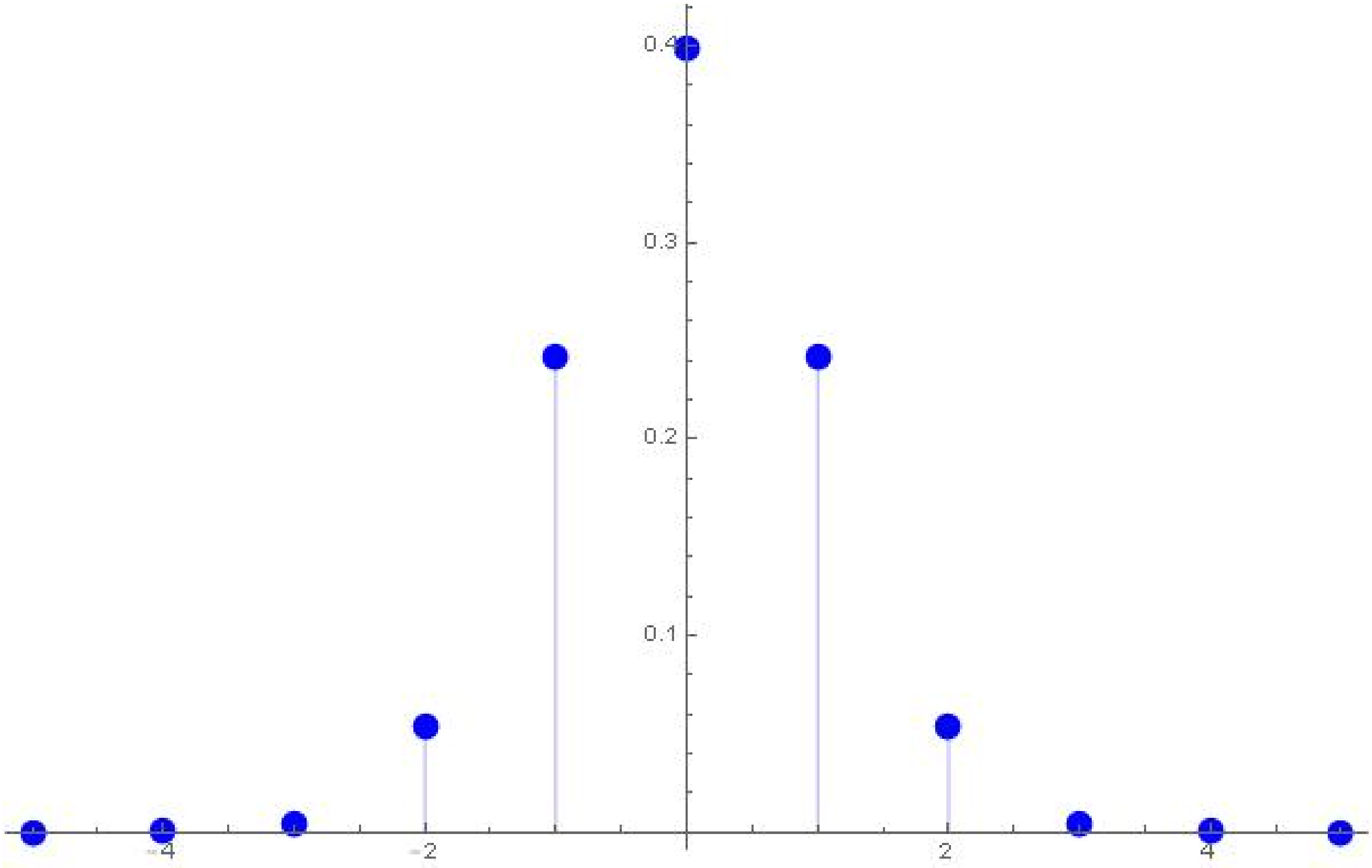
Obtaining discrete Gaussian samples

- It is easy to obtain samples for large s [GPV08]
- Our goal: take samples of width s and output samples with smaller width, say, s/Ě2
  - Then we can simply repeat
- Naïve attempt: given x output x/2
  - Problem: x/2 is not in the lattice!
- Second naïve attempt: only take x in 2L, and then output x/2
  - Correct output distribution, but we keep only 2^{-n} of the samples

Obtaining discrete Gaussian samples

- A better attempt: partition the samples according to their coset of 2L
- Then take two samples from a coset and output their average
  - Notice that if x,y are in the same coset of 2L, then x+y is in 2L, and so (x+y)/2 is in L
- Intuitively, since x and y are Gaussian with s, then x+y is Gaussian with Ĕ2·s, and (x+y)/2 is Gaussian with s/Ĕ2
- But is it distributed correctly?

Input Distribution

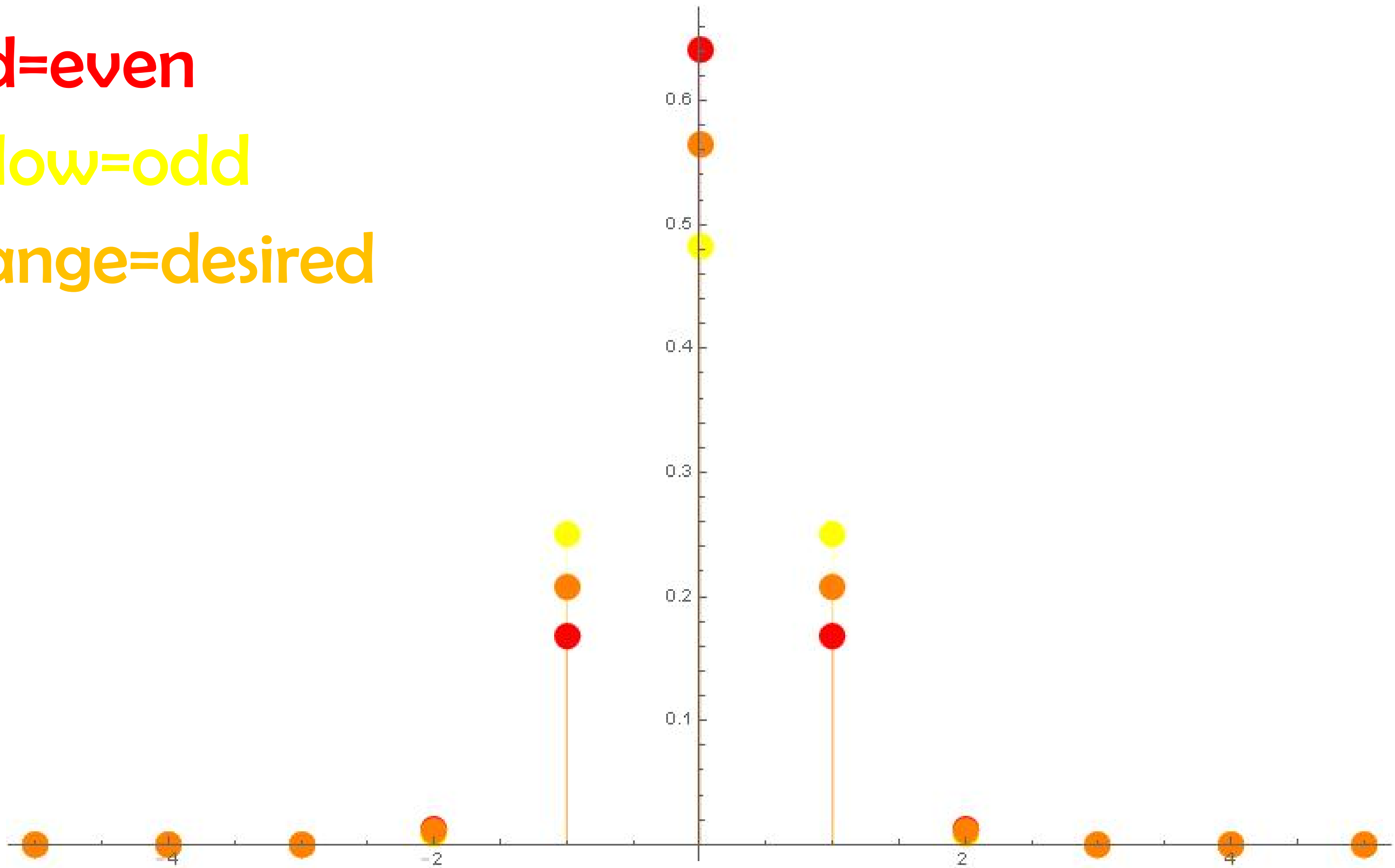


Output Distribution

Red=even

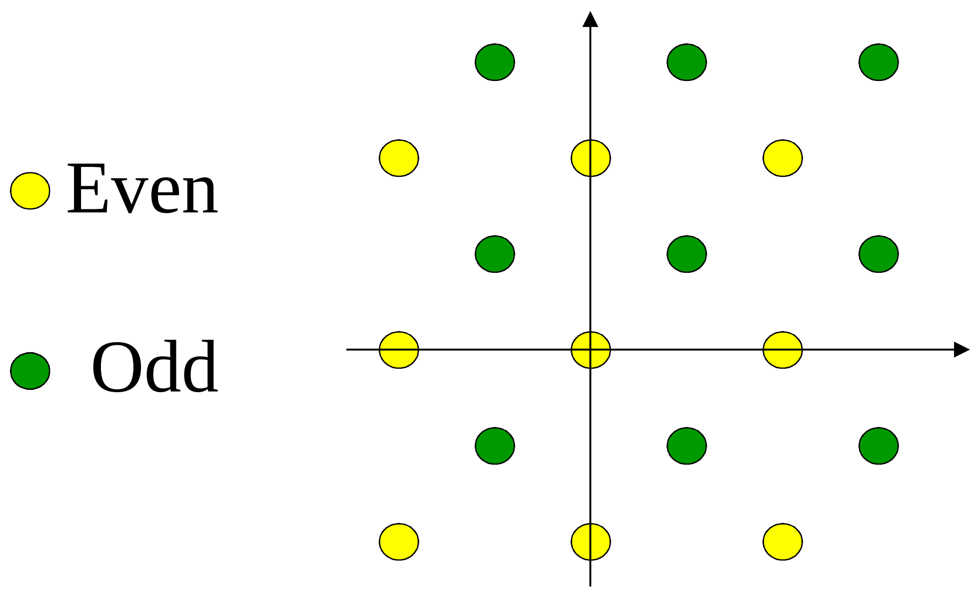
Yellow=odd

Orange=desired



Obtaining discrete Gaussian samples

- It turns out that by taking "even" with probability p^2_{even} and "odd" with probability p^2_{odd} , we get *exactly* the discrete Gaussian distribution
- Proof by picture in the one-dimensional case:



Square Sampling

- Summary so far:
  - Bucket the samples into 2^n buckets, based on their coset of 2L
  - Then pick a bucket with probability proportional to *square* of its probability, and output $(x+y)/2$ for two vectors in the bucket
- For this we use a "square sampling" procedure: given samples from a distribution $(p_1,\ldots,p_N)$, output samples from the distribution $(p_1^2,\ldots,p_N^2)/\sum p_i^2$
  - We do this using rejection sampling
  - The loss rate is $\sum p_i^2/p_{max}$
  - Total loss is $2^{n/2}$ due to magic!

Summary

- In time 2^n we are able to sample from the discrete Gaussian distribution (of any radius)
  - This implies a 2^n time algorithm for SVP
  - Recent work by my coauthors: also CVP in 2^n !
- A close inspection of our algorithm shows that $2^{n/2}$ should be the right answer
  - So far we are only able to achieve that above smoothing
  - This implies $2^{n/2}$ algorithm for O(1)-GapSVP
  - Puzzle: given coin with unknown heads probability p; output a coin with probability Ěp