Predicting Lattice reduction

Nicolas Gama, Phong Nguyen

Ecole Normale Supérieure,CNRS,INRIA, France.

April 14, 2008

Central Questions

- What is the concrete hardness of Lattice problems?
- How to select security parameters for a lattice-based cryptosystems?
- What is the best practical lattice reduction algorithm?
- Can we predict the output of lattice reduction algorithms without running them?

Part 1: Introduction

Part 1:

- Introduction

Historically... two extreme views on lattice problems

Everything is easy (1982-)

- Many cryptosystems broken by LLL.
- "If the cryptosystem is related to lattices, then it must be insecure"

Historically... two extreme views on lattice problems

Everything is easy (1982-)

- Many cryptosystems broken by LLL.
- "If the cryptosystem is related to lattices, then it must be insecure"

Everything is hard (1996-)

- "Lattice problems are NP-hard, even with small approximation factor"
- Worst-case to average-case reductions

Historically... two extreme views on lattice problems

Everything is easy (1982-)

- Many cryptosystems broken by LLL.
- "If the cryptosystem is related to lattices, then it must be insecure"

Everything is hard (1996-)

- "Lattice problems are NP-hard, even with small approximation factor"
- Worst-case to average-case reductions

Finding the truth...

- Is the goal of this article.

Cryptographic interests

Question

I have a cryptosystem, whose security is related to lattices.

Cryptographic interests

Question

I have a cryptosystem, whose security is related to lattices.

Cryptography

Choose parameters for the cryptosystem. (rarely done in papers presenting new lattice cryptosystems)

Cryptographic interests

Question

I have a cryptosystem, whose security is related to lattices.

Cryptography

Choose parameters for the cryptosystem. (rarely done in papers presenting new lattice cryptosystems)

Cryptanalysis

Predict whether an attack will work without implementing it, without being too optimistic, nor too pessimistic.

Limits of theory

Status of Lattice theory:

- Many theoretical articles.
- Theoretical bounds far from reality
  - "*Experiments perform always better*"
- Few concrete bounds
  - *Or which do not match experiments*
- Several unimplemented attacks turned out to fail.

Limits of theory

Status of Lattice theory:

- Many theoretical articles.
- Theoretical bounds far from reality
  - "*Experiments perform always better*"
- Few concrete bounds
  - *Or which do not match experiments*
- Several unimplemented attacks turned out to fail.

Solution

Limits of theory

Status of Lattice theory:

- Many theoretical articles.
- Theoretical bounds far from reality
  - "*Experiments perform always better*"
- Few concrete bounds
  - *Or which do not match experiments*
- Several unimplemented attacks turned out to fail.

Solution

- Only practice can reveal limits of lattice reduction.
- Computer science is also about computers.

Our Results

Results

- Prediction of the output quality of most lattice reduction algorithms.
- Precise numerical bounds on the output quality.
- Describe the domain of lattice problems solvable in practice.
- A better understanding of limitations of lattice reduction algorithms

Our Results

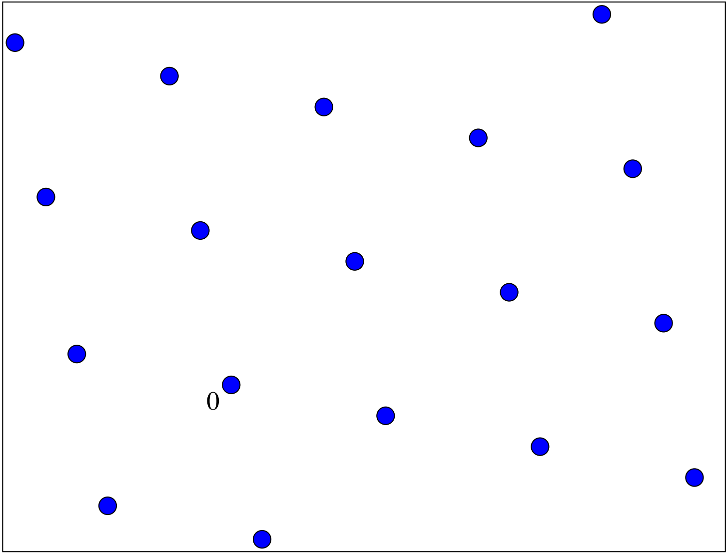
Results

- Prediction of the output quality of most lattice reduction algorithms.
- Precise numerical bounds on the output quality.
- Describe the domain of lattice problems solvable in practice.
- A better understanding of limitations of lattice reduction algorithms

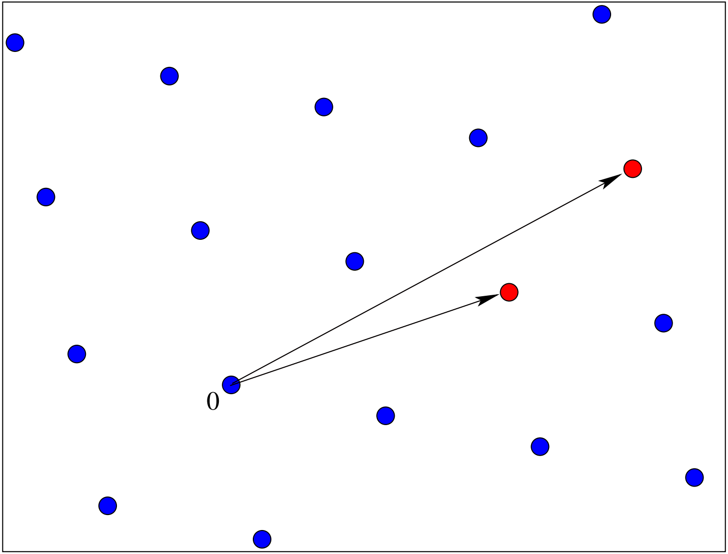
Method

- We launched lattice reduction on several processors during 1 year.
- We draw heuristics from all these simulations.
- We have validated these heuristics using past attacks.

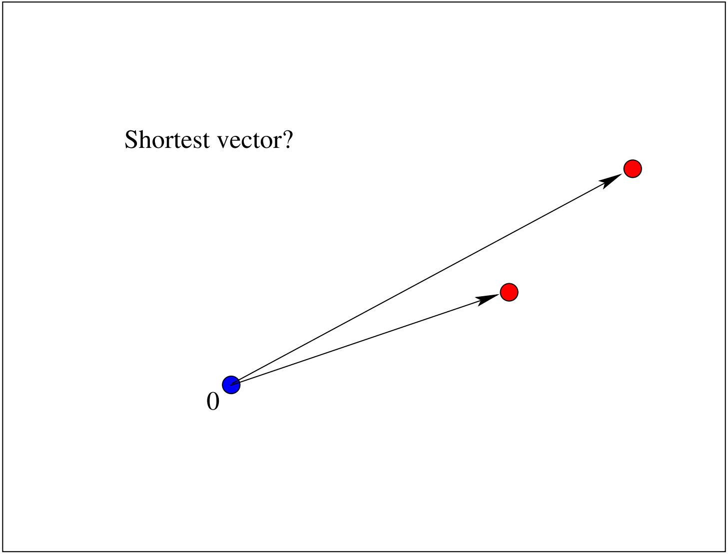
Lattice



Lattice



Lattice



Lattice problems - Approximations

Difficult lattice problems

CVP: Closest vector problem
SVP: Shortest vector problem
Hard problem (even NP-Hard)

Lattice problems - Approximations

Difficult lattice problems

CVP: Closest vector problem
SVP: Shortest vector problem
Hard problem (even NP-Hard)

Approximations problems:

- Approx-SVP: Instead of finding the shortest vector, find a vector at most α times bigger.
- Very different than in Discrete Log, or Factorization.

Lattice problems - Approximations

Difficult lattice problems

CVP: Closest vector problem
SVP: Shortest vector problem
Hard problem (even NP-Hard)

Approximations problems:

- Approx-SVP: Instead of finding the shortest vector, find a vector at most α times bigger.
- Very different than in Discrete Log, or Factorization.

Mixed problems

- Unique-SVP: If there is a vector α-times smaller than any other, find-it.

Approximations

Theory

- Lattice problems can all be approximated within an at most exponential factor by polynomial algorithms.
- Approximating SVP (or CVP) within a polynomial factor is very hard

Approximations

Theory

- Lattice problems can all be approximated within an at most exponential factor by polynomial algorithms.
- Approximating SVP (or CVP) within a polynomial factor is very hard

Practice

- The output of Lattice reduction algorithm is indeed simply exponential
- But the constant of the exponential is very close to 1.
- So close that :
    - *Approx:* It is easy to approximate SVP to a factor n^2 in very high dimensions ($n \leq 1200$).
    - *Exact:* Lattice problems are solvable exactly up to dimension 70-80, and sometimes up to dimension 300, depending on the structure of the lattice problem.

Lattice reduction algorithms

Basics on Lattice reduction algorithms

Algorithms considered:

- LLL (1982),
- semi-2k (Schnorr, 1987),
- slide (Gama-Nguyen, 2008),
- BKZ, Deep (Schnorr-Euchner, 1994).

Classification of Reduction algorithms

Basics

- Every lattice reduction algorithm uses an exhaustive search subroutine.

Classification of Reduction algorithms

Basics

- Every lattice reduction algorithm uses an exhaustive search subroutine.
- Except LLL and Deep.

Basics

- Every lattice reduction algorithm uses an exhaustive search subroutine.
- Except LLL and Deep.

Classification of Lattice reduction algorithms

Classification of Reduction algorithms

Basics

- Every lattice reduction algorithm uses an exhaustive search subroutine.
- Except LLL and Deep.

Classification of Lattice reduction algorithms

1. Theoretical algorithms
   - Proved polynomials, proved output quality.
   - In particular, number of calls to exhaustive search polynomially bounded.
   - *Concerns:* LLL - semi-block-$2k$ reduction - Slide reduction - ...

Classification of Reduction algorithms

Basics

- Every lattice reduction algorithm uses an exhaustive search subroutine.
- Except LLL and Deep.

Classification of Lattice reduction algorithms

1. Theoretical algorithms
   - Proved polynomials, proved output quality.
   - In particular, number of calls to exhaustive search polynomially bounded.
   - *Concerns:* LLL - semi-block-2k reduction - Slide reduction - ...
2. Practical algorithms
   - No bound on complexity.
   - Sometimes no bound on quality either.
   - *Concerns:* BKZ - Deep

Part 2:

- Approximation algorithms

Quality factors

Quality of Lattice reduction:

- The length of the first vector (normalized):
  - **Hermite factor: (HF)** Compared to the n^{th}-root of the volume $\left\|\vec{b}_1\right\| / \mathrm{vol}(L)^{1/n}$.
  - **Approx factor: (AF)** Compared to the shortest vector $\left\|\vec{b}_1\right\| / \lambda_1(L)$.
  - Dual factor:$\max_k \left\|\vec{b}_1\right\| / \left\|\vec{b}_k^*\right\|$ where $\left\|\vec{b}_k^*\right\| = \mathrm{distance}(\vec{b}_k,\ \mathrm{span}(\vec{b}_1, \ldots, \vec{b}_{k-1}))$.

Quality of LLL.

Theory

- Hermite Factor $\leq \sqrt{\gamma_2}^{n-1} \approx 1.07^n$
- Approx Factor $\leq \gamma_2^{n-2} \approx 1.15^n$
- There are worst case bases reaching both bounds.

How to deal with worst case Bases

Facts.

- A lattice contains more than one basis!
- If a particular basis can not be reduced, it does not mean that the lattice is hard to reduce.
- Theory doesn't give accurate results on the "average" among bases of the same lattice.

How to deal with worst case Bases

Facts.

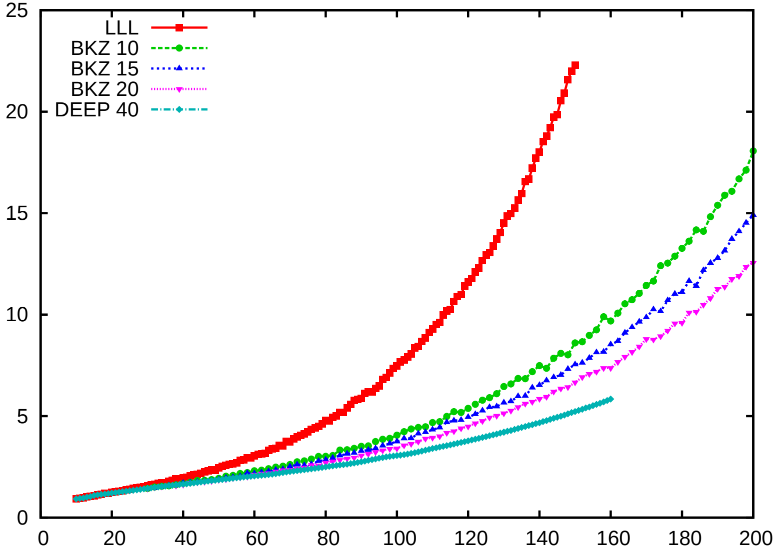
- A lattice contains more than one basis!
- If a particular basis can not be reduced, it does not mean that the lattice is hard to reduce.
- Theory doesn't give accurate results on the "average" among bases of the same lattice.

Practice

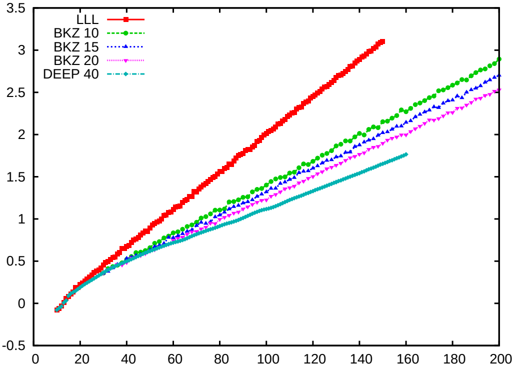
For any lattice, by randomizing the input basis, you get with LLL

- Hermite Factor $\leq 1.022^n$ (compare with 1.07^n)
- Approx Factor $\leq 1.043^n$ (compare with 1.15^n)

Hermite factors of different algorithms



Same in log scale



Quality of Deep, BKZ $\approx \sqrt{\text{Quality of LLL}}$.

Differences between theory and practice: Deep

Theory: (Deep)

- No exhaustive search
- Simplistic svp oracle: take the smallest vector of the basis (or a projection)
- Quality: Same worst-case bases than LLL (Same theoretical upper-bounds as LLL)
- Complexity: No bound.

Practice:

- Beats BKZ in very high dimension
- Quality: 1.011^n

Differences between theory and practice: Slide

Theory

- Hermite Factor $\leq \sqrt{\gamma_k}^{(n-1)/(k-1)}$ (better than BKZ)
- Approx Factor $\leq \gamma_k^{(n-k)/(k-1)}$ (better than BKZ)
- Polynomial (quadratic) number of calls to exhaustive search (much better than BKZ)
- Every provable indicators are better for Slide reduction than BKZ.

Differences between theory and practice: Slide

Theory

- Hermite Factor $\leq \sqrt{\gamma_k}^{(n-1)/(k-1)}$ (better than BKZ)
- Approx Factor $\leq \gamma_k^{(n-k)/(k-1)}$ (better than BKZ)
- Polynomial (quadratic) number of calls to exhaustive search (much better than BKZ)
- Every provable indicators are better for Slide reduction than BKZ.

Practice

- Practical up to blocksize ≈ 60, and "quality better than proved"
- faster than BKZ for blocksizes ≥ 20
- but...

Differences between theory and practice: Slide

Theory

- Hermite Factor $\leq \sqrt{\gamma_k}^{(n-1)/(k-1)}$ (better than BKZ)
- Approx Factor $\leq \gamma_k^{(n-k)/(k-1)}$ (better than BKZ)
- Polynomial (quadratic) number of calls to exhaustive search (much better than BKZ)
- Every provable indicators are better for Slide reduction than BKZ.

Practice

- Practical up to blocksize ≈ 60, and "quality better than proved"
- faster than BKZ for blocksizes ≥ 20
- but...
- Hermite Factor of Slide-60$\leq 1.013^n$

Differences between theory and practice: Slide

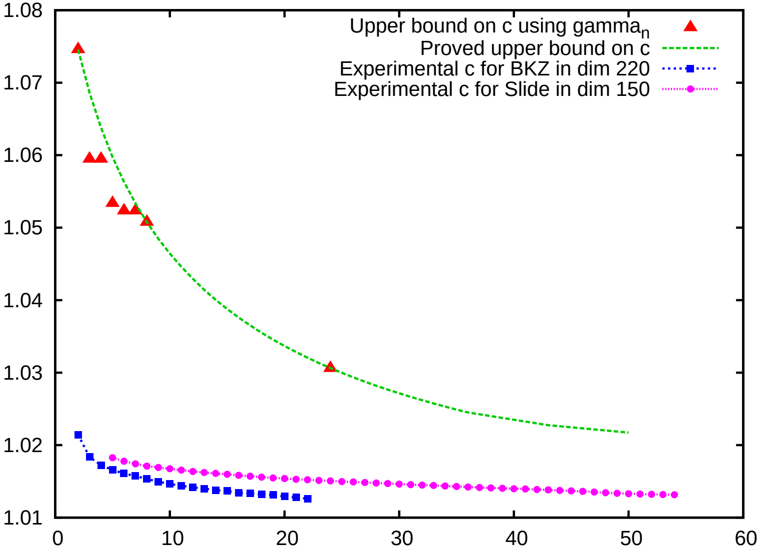
Theory

- Hermite Factor $\leq \sqrt{\gamma_k}^{(n-1)/(k-1)}$ (better than BKZ)
- Approx Factor $\leq \gamma_k^{(n-k)/(k-1)}$ (better than BKZ)
- Polynomial (quadratic) number of calls to exhaustive search (much better than BKZ)
- Every provable indicators are better for Slide reduction than BKZ.

Practice

- Practical up to blocksize ≈ 60, and "quality better than proved"
- faster than BKZ for blocksizes ≥ 20
- but...
- Hermite Factor of Slide-60$\leq 1.013^n$
- Slide-60 beaten by BKZ-20

Comparison Slide, BKZ, proved upper-bounds



Exceptional Results on Specific Lattices.

1. LO Knapsack Lattice, Orthogonal lattices (possibly mod(q))
   - From the "standard" basis, LLL provides a HF in $2^{O(\sqrt{n})}$.
2. Ajtai's worst-case to average-case lattice
   - Sub-exponential HF with LLL
   - *Note*: They are not worst case lattices for Hermite-SVP or Aprox-SVP.
3. NTRU Lattices
   - The q-vectors are small by definition!

In any case:

- One can extract a sublattice (or block) which satisfies the prediction.

Summary

1. The Hermite factor of a reduction algorithm is always smaller than in a random lattice.
2. Random lattices are worst case lattices (for Hermite factor).
3. The final Hermite factor depends on the input basis
4. If the basis is randomized, then it matches exactly the random lattice case:
   - HF $= \left\|\vec{b}_1\right\| / \mathrm{vol}(L)^{1/n}$ is at most simply exponential in n
   - The constant of the exponential is very small (1.021 downto 1.01)

From Hermite-SVP to Approx-SVP

Background

- AF can be made $\leq \mathrm{HF}^2$

Practice

- One can effectively build worst case lattices with AF $=$ HF^2

From Hermite-SVP to Approx-SVP

Background

- AF can be made $\leq \mathrm{HF}^2$
- No better bound knowm for cryptographic lattices
- But if $\lambda_1(L) \geq \mathrm{vol}(L)^{1/n}$ then AF is already $\leq \mathrm{HF}$

Practice

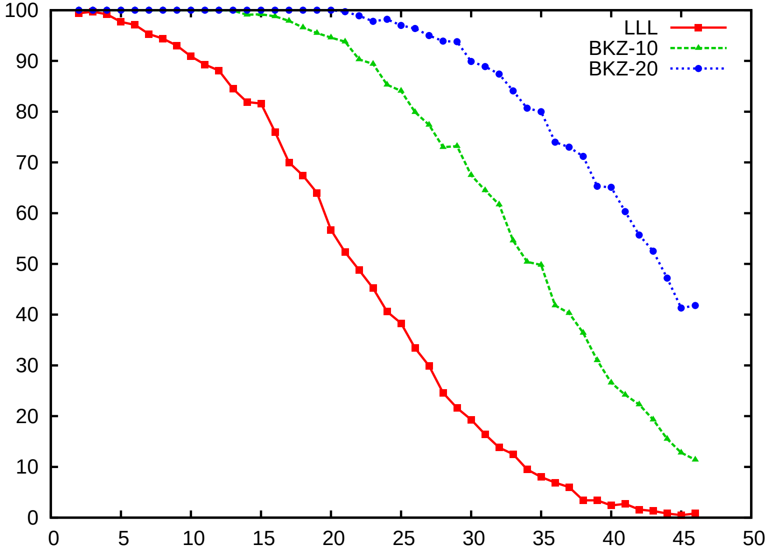
- One can effectively build worst case lattices with $\mathrm{AF} = \mathrm{HF}^2$

Part 3: Exact Algorithms

Part 3:

- Exact algorithms (very useful in cryptography)

Lattice reduction algorithms as SVP oracles



Summary

What one would expect:

- LLL is a SVP oracle up to dimension $n = 2$.

The reality:

- LLL is a randomized SVP oracle up to dimension 30-35 (on all lattices)

Approx SVP to Unique SVP

Theory

If Approx-SVP can be approximated to a factor 1.011^{2n} then Unique SVP with gap 1.011^{2n} can be solved.

Approx SVP to Unique SVP

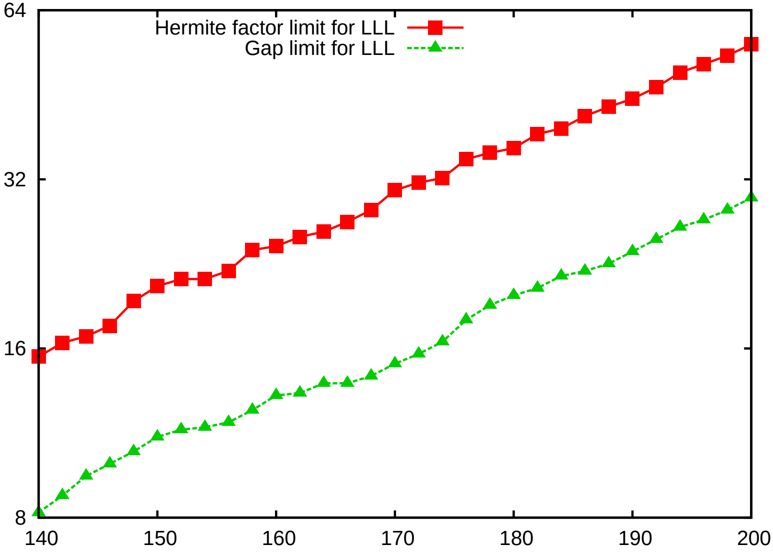
Theory

If Approx-SVP can be approximated to a factor 1.011^{2n}
then Unique SVP with gap 1.011^{2n} can be solved.

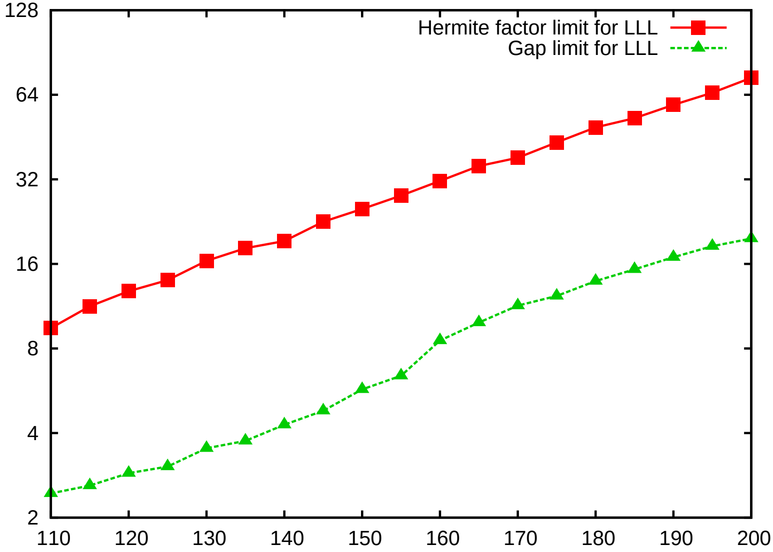
Question

- Is it possible to solve Unique-SVP when the GAP is smaller than HF^2?

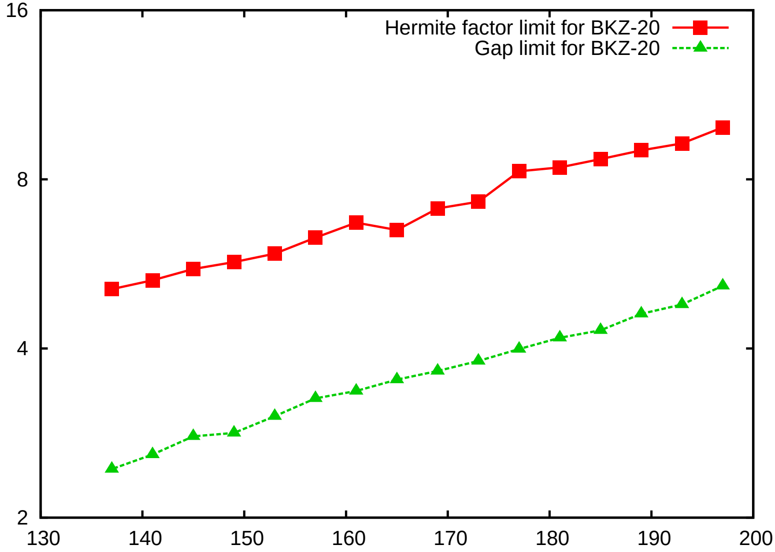
Gap vs HF on semi-orthogonal lattices



Same figure compared to LO-Lattices



Same figure for BKZ-20



Summary

Question

- Is it possible to solve Unique-SVP when the GAP is smaller than HF^2?

Summary

Question

- Is it possible to solve Unique-SVP when the GAP is smaller than HF^2?

Experimental result

1. The Gap needs to be exponential in n order to retrieve the shortest vector
2. But its order is the Hermite Factor, and not its square!

Part 4: Running Time

Part 4:

- Running Times

Questions

1. What is the actual running-time of BKZ (function of n, k)?

Running time of BKZ

Questions

1. What is the actual running-time of BKZ (function of n, k)?
2. What is the limitation in BKZ?
    - The exhaustive search?
    - The number of calls to exhaustive search?

Running time of BKZ

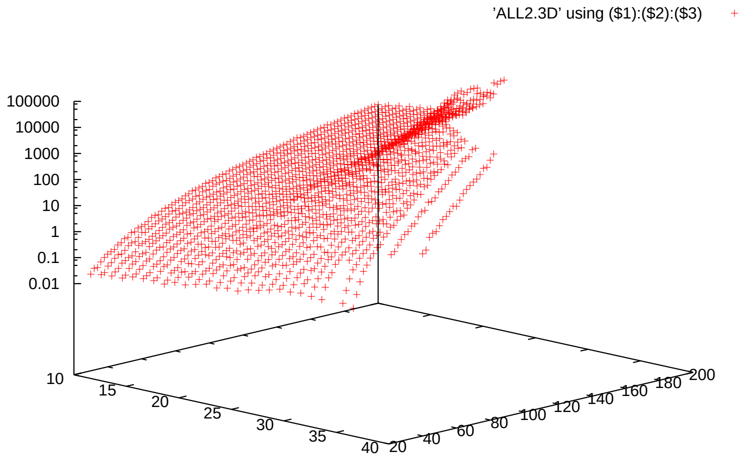
Questions

1. What is the actual running-time of BKZ (function of n, k)?
2. What is the limitation in BKZ?
    - The exhaustive search?
    - The number of calls to exhaustive search?

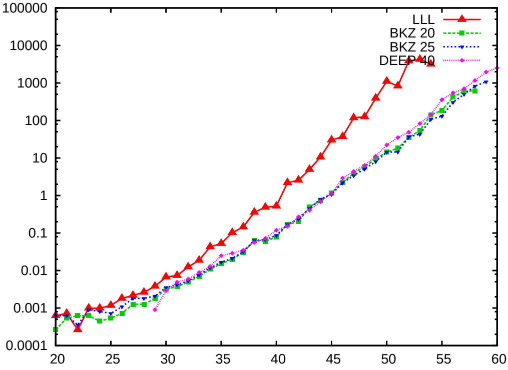
Proved bounds:

- At most doubly exponential in n and k?
- Doesn't help very much!

Overview of Experiments

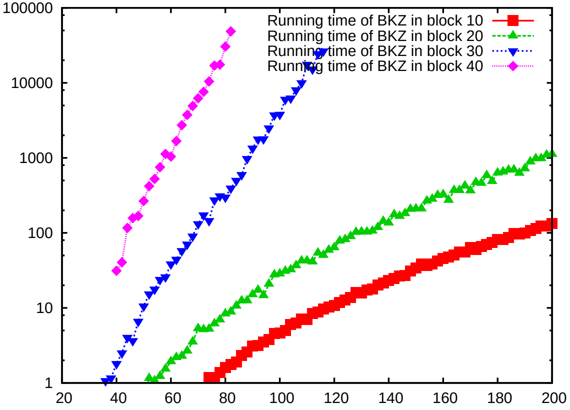


Experimental running time of exhaustive search



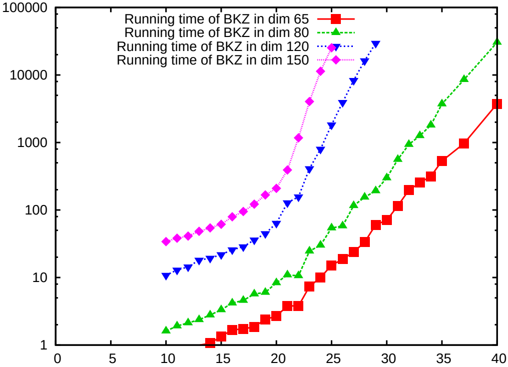
- Practical up to dimension 70
- Not enough to explain the complexity of BKZ

Experimental running time of BKZ



- Sub-exponential in n for fixed $k \lessapprox 24$
- looks exponential in n for fixed $k \geq 25$

Experimental running time of BKZ (2)



- super-exponential in k for fixed n
- High increase from $k \geq 20$

Deduction

- If one wants a higher k in BKZ, then one must reduce the dimension.
- Reducing the time of the exhaustive search (pruning) is not enough
- Experiments: from a BKZ-20 reduced basis of NTRU107 (parameters of 1998)
  - Ex1: perform BKZ-42 in a projected block of NTRU107
  - Ex2: perform BKZ-30 in projected blocks of NTRU107 of dimension 70, starting a positions multiple of 35.
- Both retrieve the private key in 1 day.
- In comparison, BKZ-25 does not end, and does not seem to retrieve the key in its temporary variables

Conclusion

- We now have a clear view of the gap between theory and practice.
- We can predict the results of Lattice reduction algorithms in most cases.

A few miracles with practical lattice reduction

- The exhaustive search, although in $2^{O(n^2)}$, works in dimension 70.
- LLL is a randomized SVP oracle (with non-negligible probability) up to dimension 30.
- Unique-SVP is solvable when the gap is linear in the Hermite-factor, not quadratic

Open problems

- Explain the values of the various experimental constants
- Find better algorithms:
  - better trade-offs between the number of calls, the cost of the subroutine, and the quality
  - Is it possible to reach an HF of 1.005^n in practice?
- Explain the Unique-SVP phenomenon.