Solving All Lattice Problems in Deterministic Single Exponential Time

Daniele Micciancio (UCSD)
(Joint work with P. Voulgaris, STOC 2010)

Barriers II Workshop, Princeton

August 27, 2010
Lattices

- Traditional area of mathematics
  - Bridge between number theory and geometry
  - Studied by Lagrange, Gauss, ..., Minkowski, ...
- Key to many algorithmic applications
  - Cryptanalysis, Coding Theory, Integer Programming
- Foundation of Lattice based Cryptography
  - Exponentially hard to break, even by quantum adversary
  - Asymptotically fast and easily parallelizable cryptographic functions
  - Secure based on conjectured hardness of worst-case problems
  - Extremely versatile: CPA/CCA encryption, digital signature, ... ring signatures, threshold encryption, IBE, ..., HIBE, ..., fully homomorphic encryption

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Finding exact solutions
- Best known algorithms run in exponential time
- NP-hard: no subexponential time solution is expected

Finding good ($n^{O(1)}$) approximations
- Foundation of lattice based cryptography
- Not known how to solve substantially faster than exact version

Finding exponential ($2^{O(n)}$) approximations
- Extensively used in cryptanalysis
- Polynomial time algorithms, based on exact solution of small dimensional subproblems

Daniele Micciancio
CVP in deterministic $2^{O(n)}$ time
Finding *exact* solutions
- Best known algorithms run in *exponential* time
- NP-hard: no subexponential time solution is expected

Finding good ($n^{O(1)}$) approximations
- Foundation of lattice based cryptography
- Not known how to solve substantially faster than exact version

Finding exponential ($2^{O(n)}$) approximations
- Extensively used in cryptanalysis
- Polynomial time algorithms, *based on exact* solution of small dimensional subproblems
1 Introduction Lattices
   • Lattice Problems
     • Algorithmic Techniques

2 New Algorithm
   • Overview
   • Voronoi Cell
   • CVPP Algorithm

3 Final Remarks and Open Problems

Daniele Micciancio
CVP in deterministic $2^{O(n)}$ time
A lattice is the set of all integer linear combinations of (linearly independent) basis vectors \( \mathbf{B} = \{ \vec{b}_1, \ldots, \vec{b}_n \} \subset \mathbb{R}^n: \)

\[
\Lambda = \sum_{i=1}^{n} \vec{b}_i \cdot \mathbb{Z}
\]
A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{ \mathbf{b}_1, \ldots, \mathbf{b}_n \} \subset \mathbb{R}^n$:

$$\Lambda = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{ \mathbf{B} \mathbf{x} : \mathbf{x} \in \mathbb{Z}^n \}$$
Point Lattices

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors \( \mathbf{B} = \{ \mathbf{b}_1, \ldots, \mathbf{b}_n \} \subset \mathbb{R}^n: \\
\Lambda = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{ \mathbf{B} \bar{x} : \bar{x} \in \mathbb{Z}^n \}

The same lattice has many bases

\[ \Lambda = \sum_{i=1}^{n} \mathbf{c}_i \cdot \mathbb{Z} \]
A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\vec{b}_1, \ldots, \vec{b}_n\} \subset \mathbb{R}^n$:

$$\Lambda = \sum_{i=1}^{n} \vec{b}_i \cdot \mathbb{Z} = \{\mathbf{B} \vec{x} : \vec{x} \in \mathbb{Z}^n\}$$

The same lattice has many bases

$$\Lambda = \sum_{i=1}^{n} \vec{c}_i \cdot \mathbb{Z}$$

**Definition (Lattice)**

Discrete additive subgroup of $\mathbb{R}^n$
Shortest Vector Problem (SVP)

Definition (SVP)
Given a lattice $\mathcal{L}(B)$, find a (nonzero) lattice vector $B\vec{x}$ (with $\vec{x} \in \mathbb{Z}^k$) of minimal length $\|B\vec{x}\|

Input: A lattice basis $B$

Daniele Micciancio
CVP in deterministic $2^{O(n)}$ time
Definition (SVP)

Given a lattice $\mathcal{L}(B)$, find a (nonzero) lattice vector $B\vec{x}$ (with $\vec{x} \in \mathbb{Z}^k$) of minimal length $\|B\vec{x}\|

- Input: A lattice basis $B$
- Output: A shortest nonzero vector $\vec{s} \in \Lambda$

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Shortest Vector Problem (SVP)

Definition (SVP)
Given a lattice $\mathcal{L}(B)$, find a (nonzero) lattice vector $B\vec{x}$ (with $\vec{x} \in \mathbb{Z}^k$) of minimal length $\|B\vec{x}\|

- Input: A lattice basis $B$
- Output: A shortest nonzero vector $\vec{s} \in \Lambda$
- The problem is hard when dimension $n$ is high and basis is skewed

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Shortest Vector Problem (SVP)

Definition (SVP)
Given a lattice $\mathcal{L}(B)$, find a (nonzero) lattice vector $B\vec{x}$ (with $\vec{x} \in \mathbb{Z}^k$) of minimal length $\|B\vec{x}\|$

- **Input**: A lattice basis $B$
- **Output**: A shortest nonzero vector $\vec{s} \in \Lambda$
- The problem is hard when dimension $n$ is high and basis is skewed
- Shortest vector can be much shorter than basis vectors

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Definition (SIVP)

Given a lattice \( \mathcal{L}(B) \), find \( n \) linearly independent lattice vectors \( \vec{s}_1, \ldots, \vec{s}_n \) of minimal length \( \max_i \| \vec{s}_i \| \).

- Input: A lattice basis \( B \)

Daniele Micciancio  
CVP in deterministic \( 2^{O(n)} \) time
**Definition (SIVP)**

Given a lattice $\mathcal{L}(\mathbf{B})$, find $n$ linearly independent lattice vectors $\mathbf{s}_1, \ldots, \mathbf{s}_n$ of minimal length $\max_i \|\mathbf{s}_i\|$

- **Input:** A lattice basis $\mathbf{B}$
- **Output:** $n$ shortest linearly independent lattice vectors $\mathbf{s}_1, \ldots, \mathbf{s}_n \in \Lambda$

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Shortest Independent Vectors Problem (SIVP)

**Definition (SIVP)**

Given a lattice $\mathcal{L}(B)$, find $n$ linearly independent lattice vectors $\vec{s}_1, \ldots, \vec{s}_n$ of minimal length $\max_i \|\vec{s}_i\|$

- **Input:** A lattice basis $B$
- **Output:** $n$ shortest linearly independent lattice vectors $\vec{s}_1, \ldots, \vec{s}_n \in \Lambda$
- The problem is hard when dimension $n$ is high and basis is skewed
Closest Vector Point (CVP)

Inhomogeneous version of SVP

**Definition (CVP)**

Given a lattice \( \mathcal{L}(B) \) and a target point \( \vec{t} \), find a lattice vector \( B\vec{x} \) which minimizes the distance \( \|B\vec{x} - \vec{t}\| \)

- Input: A lattice \( \Lambda(B) \), and a target vector \( \vec{t} \)

\[2^{O(n)} \text{ time}\]
**Definition (CVP)**

Given a lattice \( \mathcal{L}(B) \) and a target point \( \vec{t} \), find a lattice vector \( B\vec{x} \) which minimizes the distance \( \|B\vec{x} - \vec{t}\| \).

- Input: A lattice \( \Lambda(B) \), and a target vector \( \vec{t} \)
- Output: A closest lattice point \( \vec{c} \in \Lambda \)

Inhomogeneous version of SVP
Inhomogeneous version of SVP

**Definition (CVP)**

Given a lattice \( \mathcal{L}(\mathbf{B}) \) and a target point \( \vec{t} \), find a lattice vector \( \mathbf{B}\vec{x} \) which minimizes the distance \( \| \mathbf{B}\vec{x} - \vec{t} \| \)

- **Input:** A lattice \( \Lambda(\mathbf{B}) \), and a target vector \( \vec{t} \)
- **Output:** A closest lattice point \( \vec{c} \in \Lambda \)
- **NP-hard [vEB’81], even for fixed lattice [M’01]
Efficient (dimension preserving) reductions
- SVP, SIVP \( \leq \) CVP [GMSS’99, M’08]

Fastest previous algorithm
- SVP, SIVP, CVP: [Kannan’87] runs in \( n^{O(n)} \) time
- SVP: [AKS’01] runs in randomized \( 2^{O(n)} \) time and space
- Algorithms work in any \( \ell_p \) norm [BN’07]
Complexity of SVP, SIVP, CVP

- Efficient (dimension preserving) reductions
  - SVP, SIVP $\leq$ CVP [GMSS’99, M’08]

- Fastest previous algorithm
  - SVP,SIVP,CVP : [Kannan’87] runs in $n^{O(n)}$ time
  - SVP: [AKS’01] runs in randomized $2^{O(n)}$ time and space
  - Algorithms work in any $\ell_p$ norm [BN’07]

- Barriers
  - Can CVP, SIVP also be solved in $2^{c\cdot n}$ time?
  - What is the smallest constant $c$? [NV’09,MP’10,PS’10]: $c < 2.5$ for SVP in $\ell_2$.
  - Is randomization and exponential space useful/necessary?
Complexity of SVP, SIVP, CVP

- Efficient (dimension preserving) reductions
  - SVP, SIVP \(\leq\) CVP [GMSS’99, M’08]
- Fastest previous algorithm
  - SVP,SIVP,CVP : [Kannan’87] runs in \(n^{O(n)}\) time
  - SVP: [AKS’01] runs in randomized \(2^{O(n)}\) time and space
  - Algorithms work in any \(\ell_p\) norm [BN’07]
- Barriers
  - Can CVP, SIVP also be solved in \(2^{c\cdot n}\) time? Yes! (for \(\ell_2\))
  - What is the smallest constant \(c\)? [NV’09,MP’10,PS’10]: \(c < 2.5\) for SVP in \(\ell_2\). \(c \leq 2\) for SVP,SIVP,CVP!
  - Is randomization and exponential space useful/necessary? Randomization is not!
Efficient (dimension preserving) reductions
- \( \text{SVP, SIVP} \leq \text{CVP} \) [GMSS’99, M’08]

Fastest previous algorithm
- \( \text{SVP, SIVP, CVP, IP} \): [Kannan’87] runs in \( n^{O(n)} \) time
- \( \text{SVP} \): [AKS’01] runs in randomized \( 2^{O(n)} \) time and space
- Algorithms work in any \( \ell_p \) norm [BN’07]

Barriers
- Can \( \text{CVP, SIVP} \) also be solved in \( 2^{c\cdot n} \) time? Yes! (for \( \ell_2 \))
- What is the smallest constant \( c \)? [NV’09, MP’10, PS’10]: \( c < 2.5 \) for SVP in \( \ell_2 \). \( c \leq 2 \) for SVP, SIVP, CVP!
- Is randomization and exponential space useful/necessary? Randomization is not!
- What about other norms and Integer Programming (IP)?
1 Introduction Lattices
   - Lattice Problems
   - Algorithmic Techniques

2 New Algorithm
   - Overview
   - Voronoi Cell
   - CVPP Algorithm

3 Final Remarks and Open Problems
Size Reduction

- $\vec{b}$: (short) lattice vector
- $\vec{c}$: arbitrary point

Can make $\vec{c}$ shorter by subtracting $\vec{b}$ from it. Repeat until $\vec{c}$ closer to $\vec{0}$ than to $\vec{b}$.

Remarks

$\vec{c} - \vec{c}' \in \Lambda$

Key step in \[LLL'82\] basis reduction algorithm.

Technique is used in most other lattice algorithms.

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Size Reduction

- $\vec{b}$: (short) lattice vector
- $\vec{c}$: arbitrary point
- Can make $\vec{c}$ shorter by subtracting $\vec{b}$ from it

Remarks

$\vec{c} - \vec{c}' \in \Lambda$

Key step in [LLL'82] basis reduction algorithm

Technique is used in most other lattice algorithms

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Size Reduction

- \( \vec{b} \): (short) lattice vector
- \( \vec{c} \): arbitrary point

Can make \( \vec{c} \) shorter by subtracting \( \vec{b} \) from it

Repeat until \( \vec{c} \) closer to \( \vec{0} \) than to \( \vec{b} \)

Remarks

\( \vec{c} - \vec{c}' \in \Lambda \)

Key step in [LLL'82] basis reduction algorithm

Technique is used in most other lattice algorithms

Daniele Micciancio

CVP in deterministic \( 2^{O(n)} \) time
Size Reduction

- $\vec{b}$: (short) lattice vector
- $\vec{c}$: arbitrary point
- Can make $\vec{c}$ shorter by subtracting $\vec{b}$ from it
- Repeat until $\vec{c}$ closer to $\vec{0}$ than to $\vec{b}$ or $-\vec{b}$

Remarks
- $\vec{c} - \vec{c}' \in \Lambda$
- Key step in [LLL'82] basis reduction algorithm
- Technique is used in most other lattice algorithms
Goal: Solve $CVP(\Lambda_n, \vec{t})$

Partition $\Lambda_n$ into layers of the form: $\Lambda_n - 1 + c \vec{b}_n$, $c = 2, 1, 3, 0, ...$

Find lattice point $\vec{v}_i$ in each layer closest to (the projection of) $\vec{t}$

Only need to consider nearby layers

Dual LLL: $2^n$ layers
Dual SVP: $n$ layers

Select the best solution $\vec{v}_1$

Notice: All layers contain same lattice $\Lambda_n$
Goal: Solve $CVP(\Lambda_n, \vec{t})$
Partition $\Lambda_n$ into *layers* of the form: $\Lambda_{n-1} + c\vec{b}_n$, $c = 2, 1, 3, 0, \ldots$
Goal: Solve $CVP(\Lambda_n, \vec{t})$
Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + c\vec{b_n}$, $c = 2, 1, 3, 0, \ldots$
Find lattice point $\vec{v_i}$ in each layer closest to (the projection of) $\vec{t}$

Notice: All layers contain the same lattice $\Lambda_{n-1}$

Daniele Micciancio  CVP in deterministic $2^{O(n)}$ time
Goal: Solve $\text{CVP}(\Lambda_n, \vec{t})$

Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + c \vec{b}_n$, $c = 2, 1, 3, 0, \ldots$

Find lattice point $\vec{v}_i$ in each layer closest to (the projection of) $\vec{t}$

Notice: All layers contain the same lattice $\Lambda_{n-1}$
Goal: Solve $CVP(\Lambda_n, \vec{t})$

Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + c\vec{b}_n$,
$c = 2, 1, 3, 0, \ldots$

Find lattice point $\vec{v}_i$ in each layer closest to (the projection of) $\vec{t}$

Notice: All layers contain the same lattice $\Lambda_{n-1}$

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Goal: Solve $CVP(\Lambda_n, \vec{t})$

Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + c\vec{b}_n$, $c = 2, 1, 3, 0, \ldots$

Find lattice point $\vec{v}_i$ in each layer closest to (the projection of) $\vec{t}$

Select the best solution $\vec{v}_1$ Notice: All layers contain same lattice $\Lambda_{n-1}$
Goal: Solve $CVP(\Lambda_n, \vec{t})$
Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + c\vec{b}_n$, $c = 2, 1, 3, 0, \ldots$
Find lattice point $\vec{v}_i$ in each layer closest to (the projection of) $\vec{t}$
Only need to consider nearby layers
- Dual LLL: $2^n$ layers
- Dual SVP: $n$ layers
Goal: Solve $CVP(\Lambda_n, \vec{t})$

Partition $\Lambda_n$ into layers of the form: $\Lambda_n - 1 + c\vec{b}_n$, $c = 2, 1, 3, 0, \ldots$

Find lattice point $\vec{v}_i$ in each layer closest to (the projection of) $\vec{t}$

Only need to consider nearby layers

- Dual LLL: $2^n$ layers
- Dual SVP: $n$ layers

Select the best solution $\vec{v}_1$
Rank reduction: \( CVP(\Lambda_n) \leq 2^n \cdot CVP(\Lambda_{n-1}) \)

- **Goal:** Solve \( CVP(\Lambda_n, \vec{t}) \)
- Partition \( \Lambda_n \) into layers of the form: \( \Lambda_{n-1} + c \vec{b}_n \), \( c = 2, 1, 3, 0, \ldots \)
- Find lattice point \( \vec{v}_i \) in each layer closest to (the projection of) \( \vec{t} \)
- Only need to consider nearby layers
  - Dual LLL: \( 2^n \) layers
  - Dual SVP: \( n \) layers
- Select the best solution \( \vec{v}_1 \)
- Notice: All layers contain same lattice \( \Lambda_{n-1} \)

Daniele Micciancio  
CVP in deterministic \( 2^{O(n)} \) time
1 Introduction Lattices
   - Lattice Problems
   - Algorithmic Techniques

2 New Algorithm
   - Overview
   - Voronoi Cell
   - CVPP Algorithm

3 Final Remarks and Open Problems
Solving CVP by rank reduction

- Rank reduction \( \text{CVP}(\Lambda_n) \leq k \cdot \text{CVP}(\Lambda_{n-1}) \)
  - LLL: \( k = 2^n \),
  - SVP: \( k = n \),
Solving CVP by rank reduction

- Rank reduction $CVP(\Lambda_n) \leq k \cdot CVP(\Lambda_{n-1})$
  - LLL: $k = 2^n$, $T = 2^{n^2}$
  - SVP: $k = n$, $T = n^n$
- Iterate: $CVP(\Lambda_n) \leq k \cdot CVP(\Lambda_{n-1}) \leq \cdots \leq k^n CVP(\Lambda_1) = k^n$
Solving CVP by rank reduction

- Rank reduction: $CVP(\Lambda_n) \leq k \cdot CVP(\Lambda_{n-1})$
  - LLL: $k = 2^n, \ T = 2^{n^2}$
  - SVP: $k = n, \ T = n^n$

- Iterate: $CVP(\Lambda_n) \leq k \cdot CVP(\Lambda_{n-1}) \leq \cdots \leq k^n CVP(\Lambda_1) = k^n$

- Our approach
  - Exploit the fact that recursive calls use the same lower dimensional sublattices
  - Preprocess the lattice to speed up the solution of many CVP instances
Problem (CVPP)

Find a function \( \pi \) and an efficient algorithm CVPP such that
\[
CVPP(\pi(\Lambda), \vec{t}) = CVP(\Lambda, \vec{t})
\]

- Only the running time of CVPP counts. The function \( \pi \) is arbitrary.
CVP with Preprocessing (CVPP)

Problem (CVPP)

Find a function $\pi$ and an efficient algorithm CVPP such that

$$CVPP(\pi(\Lambda), \vec{t}) = CVP(\Lambda, \vec{t})$$

- Only the running time of CVPP counts. The function $\pi$ is arbitrary.
- Complexity
  - Still NP-hard [M’01]!
  - [LLS’93,AR’04] approximates within $n^{O(1)}$ in polynomial time
  - Polynomial time solutions require $|\pi(\Lambda)| \leq n^{O(1)}$
Problem (CVPP)

Find a function $\pi$ and an efficient algorithm CVPP such that
$CVPP(\pi(\Lambda), \vec{t}) = CVP(\Lambda, \vec{t})$

- Only the running time of CVPP counts. The function $\pi$ is arbitrary.
- Complexity
  - Still NP-hard [M’01]!
  - [LLS’93,AR’04] approximates within $n^{O(1)}$ in polynomial time
  - Polynomial time solutions require $|\pi(\Lambda)| \leq n^{O(1)}$
- Our work:
  - $CVPP(\pi(\Lambda), \vec{t})$ runs in $2^{O(n)}$ time
CVP with Preprocessing (CVPP)

Problem (CVPP)

Find a function $\pi$ and an efficient algorithm CVPP such that

$$CVPP(\pi(\Lambda), \vec{t}) = CVP(\Lambda, \vec{t})$$

- Only the running time of CVPP counts. The function $\pi$ is arbitrary.
- Complexity
  - Still NP-hard [M’01]!
  - [LLS’93,AR’04] approximates within $n^{O(1)}$ in polynomial time
  - Polynomial time solutions require $|\pi(\Lambda)| \leq n^{O(1)}$
- Our work:
  - $CVPP(\pi(\Lambda), \vec{t})$ runs in $2^{O(n)}$ time
  - $\pi(\Lambda)$ has size $2^{O(n)}$

Daniele Micciancio
CVP in deterministic $2^{O(n)}$ time
CVP with Preprocessing (CVPP)

Problem (CVPP)

Find a function $\pi$ and an efficient algorithm CVPP such that
$\text{CVPP}(\pi(\Lambda), \vec{t}) = \text{CVP}(\Lambda, \vec{t})$

- Only the running time of CVPP counts. The function $\pi$ is arbitrary.

Complexity

- Still NP-hard [M’01]!
- [LLS’93,AR’04] approximates within $n^{O(1)}$ in polynomial time
- Polynomial time solutions require $|\pi(\Lambda)| \leq n^{O(1)}$

Our work:

- $\text{CVPP}(\pi(\Lambda), \vec{t})$ runs in $2^{O(n)}$ time
- $\pi(\Lambda)$ has size $2^{O(n)}$
- $\pi(\Lambda)$ can also be computed in time $2^{O(n)}$

Daniele Micciancio  CVP in deterministic $2^{O(n)}$ time
Overview of CVP algorithm

Building blocks:
- \( \pi(\Lambda) = \mathcal{V}(\Lambda) \): Voronoi cell of the lattice
Overview of CVP algorithm

Building blocks:
- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $CVP(\Lambda_n) \leq CVPP(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
Overview of CVP algorithm

Building blocks:

- \( \pi(\Lambda) = \mathcal{V}(\Lambda) \): Voronoi cell of the lattice
- Our approach: \( \text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n) \)
- \( \text{CVPP}(\mathcal{V}(\Lambda_n)) \) algorithm with running time \( 2^n \)
Overview of CVP algorithm

Building blocks:

- \( \pi(\Lambda) = \mathcal{V}(\Lambda) \): Voronoi cell of the lattice
- Our approach: \( \text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n) \)
- \( \text{CVPP}(\mathcal{V}(\Lambda_n)) \) algorithm with running time \( 2^n \)
- Voronoi cell computation \( \mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n) \)

Daniele Micciancio  CVP in deterministic \( 2^{O(n)} \) time
Overview of CVP algorithm

Building blocks:
- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $CVP(\Lambda_n) \leq CVPP(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
- $CVPP(\mathcal{V}(\Lambda_n))$ algorithm with running time $2^n$
- Voronoi cell computation $\mathcal{V}(\Lambda_n) \leq 2^n CVP(\Lambda_n)$
- Dimension reduction $CVP(\Lambda_n) \leq 2^n \cdot CVP(\Lambda_{n-1})$

Daniele Micciancio  CVP in deterministic $2^{O(n)}$ time
Overview of CVP algorithm

Building blocks:

- \( \pi(\Lambda) = \mathcal{V}(\Lambda) \): Voronoi cell of the lattice
- Our approach: \( \text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n) \)
- \( \text{CVPP}(\mathcal{V}(\Lambda_n)) \) algorithm with running time \( 2^n \)
- Voronoi cell computation \( \mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n) \)
- Dimension reduction \( \text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1}) \)

Computing the Voronoi cell of a lattice:

\[ \mathcal{V}(\Lambda_n) \]
Overview of CVP algorithm

Building blocks:
- \( \pi(\Lambda) = \mathcal{V}(\Lambda) \): Voronoi cell of the lattice
- Our approach: \( \text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n) \)
- \( \text{CVPP}(\mathcal{V}(\Lambda_n)) \) algorithm with running time \( 2^n \)
- Voronoi cell computation \( \mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n) \)
- Dimension reduction \( \text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1}) \)

Computing the Voronoi cell of a lattice:

\( \mathcal{V}(\Lambda_n) \)
Overview of CVP algorithm

Building blocks:
- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $\text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
- $\text{CVPP}(\mathcal{V}(\Lambda_n))$ algorithm with running time $2^n$
- Voronoi cell computation $\mathcal{V}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_n)$
- Dimension reduction $\text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1})$

Computing the Voronoi cell of a lattice:

$$\mathcal{V}(\Lambda_n) \leq 2^{O(n)} \cdot \text{CVP}(\Lambda_n)$$
Overview of CVP algorithm

Building blocks:
- \( \pi(\Lambda) = \mathcal{V}(\Lambda) \): Voronoi cell of the lattice
- Our approach: \( CVP(\Lambda_n) \leq CVPP(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n) \)
- \( CVPP(\mathcal{V}(\Lambda_n)) \) algorithm with running time \( 2^n \)
- Voronoi cell computation \( \mathcal{V}(\Lambda_n) \leq 2^n CVP(\Lambda_n) \)
- Dimension reduction \( CVP(\Lambda_n) \leq 2^n \cdot CVP(\Lambda_{n-1}) \)

Computing the Voronoi cell of a lattice:

\[ \mathcal{V}(\Lambda_n) \leq 2^{O(n)} CVP(\Lambda_n) \]
Overview of CVP algorithm

Building blocks:
- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $\text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
- $\text{CVPP}(\mathcal{V}(\Lambda_n))$ algorithm with running time $2^n$
- Voronoi cell computation $\mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n)$
- Dimension reduction $\text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1})$

Computing the Voronoi cell of a lattice:

$$\mathcal{V}(\Lambda_n) \leq 2^{O(n)} \text{CVP}(\Lambda_n)$$
$$\leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVP}(\Lambda_{n-1})$$
Overview of CVP algorithm

Building blocks:
- \(\pi(\Lambda) = \mathcal{V}(\Lambda)\): Voronoi cell of the lattice
- Our approach: \(\text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)\)
- \(\text{CVPP}(\mathcal{V}(\Lambda_n))\) algorithm with running time \(2^n\)
- Voronoi cell computation \(\mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n)\)
- Dimension reduction \(\text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1})\)

Computing the Voronoi cell of a lattice:

\[
\mathcal{V}(\Lambda_n) \leq 2^{O(n)} \text{CVP}(\Lambda_n) \\
\leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVP}(\Lambda_{n-1})
\]
Overview of CVP algorithm

Building blocks:
- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $\text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
- $\text{CVPP}(\mathcal{V}(\Lambda_n))$ algorithm with running time $2^n$
- Voronoi cell computation $\mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n)$
- Dimension reduction $\text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1})$

Computing the Voronoi cell of a lattice:

$$\mathcal{V}(\Lambda_n) \leq 2^{O(n)} \text{CVP}(\Lambda_n)$$
$$\leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVP}(\Lambda_{n-1})$$
$$\leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVPP}(\mathcal{V}(\Lambda_{n-1})) + \mathcal{V}(\Lambda_{n-1})$$
Overview of CVP algorithm

Building blocks:
- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $\text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
- $\text{CVPP}(\mathcal{V}(\Lambda_n))$ algorithm with running time $2^n$
- Voronoi cell computation $\mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n)$
- Dimension reduction $\text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1})$

Computing the Voronoi cell of a lattice:

$$\mathcal{V}(\Lambda_n) \leq 2^{O(n)} \text{CVP}(\Lambda_n) \leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVP}(\Lambda_{n-1}) \leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVPP}(\mathcal{V}(\Lambda_{n-1})) + \mathcal{V}(\Lambda_{n-1})$$
Overview of CVP algorithm

Building blocks:
- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $\text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
- $\text{CVPP}(\mathcal{V}(\Lambda_n))$ algorithm with running time $2^n$
- Voronoi cell computation $\mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n)$
- Dimension reduction $\text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1})$

Computing the Voronoi cell of a lattice:

$$\mathcal{V}(\Lambda_n) \leq 2^{O(n)} \text{CVP}(\Lambda_n)$$
$$\leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVP}(\Lambda_{n-1})$$
$$\leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVPP}(\mathcal{V}(\Lambda_{n-1})) + \mathcal{V}(\Lambda_{n-1})$$
$$\leq 2^{O(n)}2^{O(n)}2^{O(n)} + \mathcal{V}(\Lambda_{n-1})$$
Overview of CVP algorithm

Building blocks:

- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $\text{CVP}(\Lambda_n) \leq \text{CVPP}(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
- $\text{CVPP}(\mathcal{V}(\Lambda_n))$ algorithm with running time $2^n$
- Voronoi cell computation $\mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n)$
- Dimension reduction $\text{CVP}(\Lambda_n) \leq 2^n \cdot \text{CVP}(\Lambda_{n-1})$

Computing the Voronoi cell of a lattice:

\[
\begin{align*}
\mathcal{V}(\Lambda_n) & \leq 2^{O(n)} \text{CVP}(\Lambda_n) \\
& \leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVP}(\Lambda_{n-1}) \\
& \leq 2^{O(n)} \cdot 2^{O(n)} \cdot \text{CVPP}(\mathcal{V}(\Lambda_{n-1})) + \mathcal{V}(\Lambda_{n-1}) \\
& \leq 2^{O(n)} 2^{O(n)} 2^{O(n)} + \mathcal{V}(\Lambda_{n-1}) \\
& = 2^{O(n)} + \mathcal{V}(\Lambda_{n-1})
\end{align*}
\]
Overview of CVP algorithm

Building blocks:

- $\pi(\Lambda) = \mathcal{V}(\Lambda)$: Voronoi cell of the lattice
- Our approach: $CVP(\Lambda_n) \leq CVPP(\mathcal{V}(\Lambda_n)) + \mathcal{V}(\Lambda_n)$
- $CVPP(\mathcal{V}(\Lambda_n))$ algorithm with running time $2^n$
- Voronoi cell computation $\mathcal{V}(\Lambda_n) \leq 2^n CVP(\Lambda_n)$
- Dimension reduction $CVP(\Lambda_n) \leq 2^n \cdot CVP(\Lambda_{n-1})$

Computing the Voronoi cell of a lattice:

\[
\mathcal{V}(\Lambda_n) \leq 2^{O(n)} CVP(\Lambda_n) \\
\leq 2^{O(n)} \cdot 2^{O(n)} \cdot CVP(\Lambda_{n-1}) \\
\leq 2^{O(n)} \cdot 2^{O(n)} \cdot CVPP(\mathcal{V}(\Lambda_{n-1})) + \mathcal{V}(\Lambda_{n-1}) \\
\leq 2^{O(n)}2^{O(n)}2^{O(n)} + \mathcal{V}(\Lambda_{n-1}) \\
= 2^{O(n)} + \mathcal{V}(\Lambda_{n-1}) \\
\leq 2^{O(n)} + 2^{O(n)} + \mathcal{V}(\Lambda_{n-2}) \leq \ldots \leq 2^{O(n)}
\]
1 Introduction Lattices
   - Lattice Problems
   - Algorithmic Techniques

2 New Algorithm
   - Overview
   - Voronoi Cell
   - CVPP Algorithm

3 Final Remarks and Open Problems
Voronoi Cell

**Definition (Voronoit Cell)**

Set of points in $\mathbb{R}^n$ closer to 0 than to any other lattice point

$$\mathcal{V}(\Lambda) = \{ \vec{x} : \forall \vec{v} \in \Lambda, \| \vec{x} \| \leq \| \vec{x} - \vec{v} \| \}$$
Representing the Voronoi cell

Each $\vec{v} \in \Lambda$ defines

$$\mathcal{H}_\vec{v} = \{ \vec{x} : \|\vec{x}\| \leq \|\vec{x} - \vec{v}\| \}$$

Theorem (Voronoi)

The number of relevant points is at most $|R| \leq 2 \cdot (2^n - 1)$
Representing the Voronoi cell

- Each $\vec{v} \in \Lambda$ defines
  \[ \mathcal{H}_\vec{v} = \{ \vec{x} : \|\vec{x}\| \leq \|\vec{x} - \vec{v}\| \} \]

- $\mathcal{V}$ is the intersection
  \[ \mathcal{V} = \bigcap_{\vec{v} \in \Lambda} \mathcal{H}_\vec{v} \]
Representing the Voronoi cell

- Each $\vec{v} \in \Lambda$ defines $H_{\vec{v}} = \{ \vec{x} : \|\vec{x}\| \leq \|\vec{x} - \vec{v}\| \}$
- $\mathcal{V}$ is the intersection $\mathcal{V} = \bigcap_{\vec{v} \in \Lambda} H_{\vec{v}}$

Daniele Micciancio  CVP in deterministic $2^{O(n)}$ time
Representing the Voronoi cell

- Each $\vec{v} \in \Lambda$ defines
  \[ \mathcal{H}_{\vec{v}} = \{ \vec{x} : \|\vec{x}\| \leq \|\vec{x} - \vec{v}\| \} \]
- $\mathcal{V}$ is the intersection
  \[ \mathcal{V} = \bigcap_{\vec{v} \in \Lambda} \mathcal{H}_{\vec{v}} \]

Daniele Micciancio  CVP in deterministic $2^{O(n)}$ time
Each $\vec{v} \in \Lambda$ defines

$$\mathcal{H}_\vec{v} = \{ \vec{x} : \|\vec{x}\| \leq \|\vec{x} - \vec{v}\| \}$$

$\mathcal{V}$ is the intersection

$$\mathcal{V} = \bigcap_{\vec{v} \in \Lambda} \mathcal{H}_\vec{v}$$
Representing the Voronoi cell

- Each $\vec{v} \in \Lambda$ defines
  $$\mathcal{H}_\vec{v} = \{\vec{x} : \|\vec{x}\| \leq \|\vec{x} - \vec{v}\|\}$$

- $\mathcal{V}$ is the intersection
  $$\mathcal{V} = \bigcap_{\vec{v} \in \Lambda} \mathcal{H}_\vec{v}$$
Representing the Voronoi cell

- Each \( \vec{v} \in \Lambda \) defines
  \[ \mathcal{H}_{\vec{v}} = \{ \vec{x} : \|\vec{x}\| \leq \|\vec{x} - \vec{v}\| \} \]

- \( \mathcal{V} \) is the intersection
  \[ \mathcal{V} = \bigcap_{\vec{v} \in R} \mathcal{H}_{\vec{v}}, \ R \subset \Lambda \]

- Not all \( \vec{v} \in \Lambda \) are needed

Daniele Micciancio        CVP in deterministic \( 2^{O(n)} \) time
Representing the Voronoi cell

Each \( \vec{v} \in \Lambda \) defines

\[ \mathcal{H}_\vec{v} = \{ \vec{x} : \|\vec{x}\| \leq \|\vec{x} - \vec{v}\| \} \]

\( \mathcal{V} \) is the intersection

\[ \mathcal{V} = \bigcap_{\vec{v} \in R} \mathcal{H}_\vec{v}, \quad R \subset \Lambda \]

Not all \( \vec{v} \in \Lambda \) are needed

**Theorem (Voronoi)**

*The number of relevant points is at most* \( |R| \leq 2 \cdot (2^n - 1) \)
Computing $\mathcal{V}(\Lambda_n)$

- Why is $|R| \leq 2 \cdot (2^n - 1)$?

Partition $\Lambda$ into cosets modulo $2\Lambda$

There are $2^n - 1$ nonzero cosets

From each coset, select the pair $\vec{v}, -\vec{v}$ closest to $\vec{0}$

$R$ is the set of all such pairs

Each pair is found by a CVP computation in lattice $2\Lambda$

CVP$(2\Lambda)$ is equivalent to CVP$(\Lambda)$

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Why is $|R| \leq 2 \cdot (2^n - 1)$?

Partition $\Lambda$ into cosets modulo $2\Lambda$
Computing $V(\Lambda_n)$

- Why is $|R| \leq 2 \cdot (2^n - 1)$?
- Partition $\Lambda$ into cosets modulo $2\Lambda$
- There are $2^n - 1$ nonzero cosets

Daniele Micciancio  
CVP in deterministic $2^{O(n)}$ time
Computing $V(\Lambda_n)$

- Why is $|R| \leq 2 \cdot (2^n - 1)$?
- Partition $\Lambda$ into cosets modulo $2\Lambda$
- There are $2^n - 1$ nonzero cosets
- From each coset, select the pair $\vec{v}, -\vec{v}$ closest to $\vec{0}$

Daniele Micciancio
CVP in deterministic $2^{O(n)}$ time
Computing $\mathcal{V}(\Lambda_n)$

Why is $|R| \leq 2 \cdot (2^n - 1)$?

Partition $\Lambda$ into cosets modulo $2\Lambda$

There are $2^n - 1$ nonzero cosets

From each coset, select the pair $\vec{v}, -\vec{v}$ closest to $\vec{0}$

$R$ is the set of all such pairs

Each pair is found by a CVP computation in lattice $2\Lambda$

CVP $(2\Lambda)$ is equivalent to CVP $(\Lambda)$

Daniele Micciancio
CVP in deterministic $2^{O(n)}$ time
Computing $\mathcal{V}(\Lambda_n)$

- Why is $|R| \leq 2 \cdot (2^n - 1)$?
- Partition $\Lambda$ into cosets modulo $2\Lambda$
- There are $2^n - 1$ nonzero cosets
- From each coset, select the pair $\vec{v}, -\vec{v}$ closest to $\vec{0}$

Each pair is found by a CVP computation in lattice $2\Lambda$

CVP($2\Lambda$) is equivalent to CVP($\Lambda$)

CVP in deterministic $2^{O(n)}$ time
Why is $|R| \leq 2 \cdot (2^n - 1)$?

Partition $\Lambda$ into cosets modulo $2\Lambda$.

There are $2^n - 1$ nonzero cosets.

From each coset, select the pair $\vec{v}, -\vec{v}$ closest to $\vec{0}$.
Computing $\mathcal{V}(\Lambda_n)$

- Why is $|R| \leq 2 \cdot (2^n - 1)$?
- Partition $\Lambda$ into cosets modulo $2\Lambda$
- There are $2^n - 1$ nonzero cosets
- From each coset, select the pair $\vec{v}, -\vec{v}$ closest to $\vec{0}$

$\vec{v}_1$, $\vec{v}_2$, $-\vec{v}_1$, $-\vec{v}_2$
Computing $\mathcal{V}(\Lambda_n)$

- Why is $|R| \leq 2 \cdot (2^n - 1)$?
- Partition $\Lambda$ into cosets modulo $2\Lambda$
- There are $2^n - 1$ nonzero cosets
- From each coset, select the pair $\vec{v}, -\vec{v}$ closest to $\vec{0}$

$R$ is the set of all such pairs. Each pair is found by a CVP computation in lattice $2\Lambda$. CVP $(2\Lambda)$ is equivalent to CVP $(\Lambda)$. 

Daniele Micciancio  
CVP in deterministic $2^{O(n)}$ time
Computing $\mathcal{V}(\Lambda_n) \leq 2^n \text{CVP}(\Lambda_n)$

- Why is $|R| \leq 2 \cdot (2^n - 1)$?
- Partition $\Lambda$ into cosets modulo $2\Lambda$
- There are $2^n - 1$ nonzero cosets
- From each coset, select the pair $\vec{v}, -\vec{v}$ closest to $\vec{0}$
- $R$ is the set of all such pairs
- Each pair is found by a CVP computation in lattice $2\Lambda$
- $\text{CVP}(2\Lambda)$ is equivalent to $\text{CVP}(\Lambda)$

Daniele Micciancio  
CVP in deterministic $2^{O(n)}$ time
1 Introduction Lattices
   • Lattice Problems
   • Algorithmic Techniques

2 New Algorithm
   • Overview
   • Voronoi Cell
   • CVPP Algorithm

3 Final Remarks and Open Problems
Definition (CVP)

Given $\Lambda$ and $\vec{t}$, find $\vec{v} \in \Lambda$ such that $\vec{t} \in \vec{v} + \mathcal{V}$

CVP goal: bring $\vec{t}$ inside $\mathcal{V}$ by shifting it by $\vec{v} \in \Lambda$.
Definition (CVP)

Given \( \Lambda \) and \( \vec{t} \), find \( \vec{v} \in \Lambda \) such that \( \vec{t} \in \vec{v} + \mathcal{V} \)

\[ \vec{t} \in \vec{v} + \mathcal{V} \iff \vec{t} - \vec{v} \in \mathcal{V} \]
Definition (CVP)

Given \( \Lambda \) and \( \vec{t} \), find \( \vec{v} \in \Lambda \) such that \( \vec{t} \in \vec{v} + \mathcal{V} \)

- \( \vec{t} \in \vec{v} + \mathcal{V} \iff \vec{t} - \vec{v} \in \mathcal{V} \)
- CVP goal: bring \( \vec{t} \) inside \( \mathcal{V} \) by shifting it by \( \vec{v} \in \Lambda \)

Daniele Micciancio

CVP in deterministic \( 2^{O(n)} \) time
Definition (CVP)

Given $\Lambda$ and $\vec{t}$, find $\vec{v} \in \Lambda$ such that $\vec{t} \in \vec{v} + \mathcal{V}$.

- $\vec{t} \in \vec{v} + \mathcal{V} \equiv \vec{t} - \vec{v} \in \mathcal{V}$
- CVP goal: bring $\vec{t}$ inside $\mathcal{V}$ by shifting it by $\vec{v} \in \Lambda$
- Algorithm [SFS’09]:
  - While $\vec{t} \notin \mathcal{V}$:
  - Select $\vec{v} \in \mathbb{R}$, $\vec{t} \notin \mathcal{H}_\vec{v}$
  - size reduce $\vec{t}$ using $\vec{v}$
Definition (CVP)

Given \( \Lambda \) and \( \vec{t} \), find \( \vec{v} \in \Lambda \) such that \( \vec{t} \in \vec{v} + \mathcal{V} \)

- \( \vec{t} \in \vec{v} + \mathcal{V} \equiv \vec{t} - \vec{v} \in \mathcal{V} \)
- CVP goal: bring \( \vec{t} \) inside \( \mathcal{V} \) by shifting it by \( \vec{v} \in \Lambda \)
- Algorithm [SFS’09]:
  - While \( \vec{t} \notin \mathcal{V} \):
  - Select \( \vec{v} \in \mathcal{R} \) . \( \vec{t} \notin \mathcal{H}_\vec{v} \)
  - size reduce \( \vec{t} \) using \( \vec{v} \)

[SFS’09] only proves termination
Definition (CVP)

Given $\Lambda$ and $\vec{t}$, find $\vec{v} \in \Lambda$ such that $\vec{t} \in \vec{v} + \mathcal{V}$

- $\vec{t} \in \vec{v} + \mathcal{V} \equiv \vec{t} - \vec{v} \in \mathcal{V}$
- CVP goal: bring $\vec{t}$ inside $\mathcal{V}$ by shifting it by $\vec{v} \in \Lambda$
- Algorithm [SFS’09]:
  - While $\vec{t} \not\in \mathcal{V}$:
  - Select $\vec{v} \in R$ . $\vec{t} \not\in \mathcal{H}_\vec{v}$
  - size reduce $\vec{t}$ using $\vec{v}$

[SFS’09] only proves termination

Question: What is a good selection strategy for $\vec{v} \in R$?
Our selection strategy

Assume $\vec{t} \in 2\mathcal{V}$

Strategy:
Compute smallest $k \in \mathbb{R}$ such that $\vec{t} \in k\mathcal{V}$
Subtract the relevant vector associated to corresponding facet

Why does it work?
The new vector $\vec{t}'$ is shorter than $\vec{t}$ still $\vec{t}' \in 2\mathcal{V} \\ |(\vec{t}' - \Lambda) \cap 2\mathcal{V}| \leq 2^n$

Daniele Micciancio
CVP in deterministic $2^{O(n)}$ time
Our selection strategy

- Assume $\vec{t} \in 2\mathcal{V}$
- Goal: find $\vec{t'} \in \vec{t} - \Lambda \cap \mathcal{V}$:

Why does it work? The new vector $\vec{t'}$ is shorter than $\vec{t}$ still $\vec{t'} \in 2\mathcal{V}$. $|\vec{t} - \Lambda \cap 2\mathcal{V}| \leq 2^n$
Our selection strategy

- Assume $\vec{t} \in 2V$
- Goal: find $\vec{t}' \in \vec{t} - \Lambda \cap V$:
- Strategy:
  - Compute smallest $k \in \mathbb{R}$ such that $\vec{t} \in kV$

Why does it work?
The new vector $\vec{t}'$ is shorter than $\vec{t}$ still $\vec{t}' \in 2V \setminus (\vec{t} - \Lambda \cap V) \subseteq 2V$.
Our selection strategy

- Assume \( \vec{t} \in 2\mathcal{V} \)
- Goal: find \( \vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V} \):
- Strategy:
  - Compute smallest \( k \in \mathbb{R} \) such that \( \vec{t} \in k\mathcal{V} \)
  - Subtract the relevant vector associated to corresponding facet

Daniele Micciancio  
CVP in deterministic \( 2^{O(n)} \) time
Our selection strategy

- Assume $\vec{t} \in 2\mathcal{V}$
- Goal: find $\vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V}$:
- Strategy:
  - Compute smallest $k \in \mathbb{R}$ such that $\vec{t} \in k\mathcal{V}$
  - Subtract the relevant vector associated to corresponding facet
- Why does it work?
Our selection strategy

- Assume $\vec{t} \in 2\mathcal{V}$
- Goal: find $\vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V}$:
- Strategy:
  - Compute smallest $k \in \mathbb{R}$ such that $\vec{t} \in k\mathcal{V}$
  - Subtract the relevant vector associated to corresponding facet
- Why does it work?
  - The new vector $\vec{t}'$ is shorter than $\vec{t}$
Our selection strategy

- Assume $\vec{t} \in 2\mathcal{V}$
- Goal: find $\vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V}$:
- Strategy:
  - Compute smallest $k \in \mathbb{R}$ such that $\vec{t} \in k\mathcal{V}$
  - Subtract the relevant vector associated to corresponding facet
- Why does it work?
  - The new vector $\vec{t}'$ is shorter than $\vec{t}$
  - still $\vec{t}' \in 2\mathcal{V}$
Our selection strategy

- Assume $\vec{t} \in 2V$
- Goal: find $\vec{t}' \in \vec{t} - \Lambda \cap V$
- Strategy:
  - Compute smallest $k \in \mathbb{R}$ such that $\vec{t} \in kV$
  - Subtract the relevant vector associated to corresponding facet
- Why does it work?
  - The new vector $\vec{t}'$ is shorter than $\vec{t}$
  - still $\vec{t}' \in 2V$
  - $| (\vec{t} - \Lambda) \cap 2V | \leq 2^n$
Doubling the Voronoi Cell

Solve CVP for any $\vec{t}$:

- Find $\vec{k} \in \mathbb{Z}$ such that $\vec{t} \in 2^k \mathcal{V}$
- Use CVP$_{2^k \mathcal{V}}$ to go from $2^k \mathcal{V}$ to $2^{k-1} \mathcal{V}$

Daniele Micciancio  CVP in deterministic $2^{O(n)}$ time
Doubling the Voronoi Cell

Solve CVP for any $\vec{t}$:
- Find $\vec{k} \in \mathbb{Z}$ such that $\vec{t} \in 2^k \mathcal{V}$
- Use CVP$_{2\mathcal{V}}$ to go from $2^k \mathcal{V}$ to $2^{k-1} \mathcal{V}$
Doubling the Voronoi Cell

Solve CVP for any $\vec{t}$:

- Find $k \in \mathbb{Z}$ such that $\vec{t} \in 2^k \mathcal{V}$
- Use CVP$_{2V}$ to go from $2^k \mathcal{V}$ to $2^{k-1} \mathcal{V}$
Doubling the Voronoi Cell

Solve CVP for any $\vec{t}$:
- Find $\vec{k} \in \mathbb{Z}$ such that $\vec{t} \in 2^k \mathcal{V}$
- Use CVP$_{2^k \mathcal{V}}$ to go from $2^k \mathcal{V}$ to $2^{k-1} \mathcal{V}$

Daniele Micciancio

CVP in deterministic $2^{O(n)}$ time
Doubling the Voronoi Cell

Solve CVP for any $\vec{t}$:

- Find $\vec{k} \in \mathbb{Z}$ such that $\vec{t} \in 2^k \mathcal{V}$
- Use CVP$_{2\mathcal{V}}$ to go from $2^k \mathcal{V}$ to $2^{k-1} \mathcal{V}$
Doubling the Voronoi Cell

Solve CVP for any $\vec{t}$:

- Find $\vec{k} \in \mathbb{Z}$ such that $\vec{t} \in 2^k \mathcal{V}$
- Use CVP$_{2\mathcal{V}}$ to go from $2^k \mathcal{V}$ to $2^{k-1} \mathcal{V}$
Doubling the Voronoi Cell

Solve CVP for any $\vec{t}$:

- Find $k \in \mathbb{Z}$ such that $\vec{t} \in 2^k \mathcal{V}$
- Use CVP$_{2\mathcal{V}}$ to go from $2^k \mathcal{V}$ to $2^{k-1} \mathcal{V}$
CVP can be solved deterministically in time $2^{c \cdot n}$

Algorithms for SVP, SIVP and many other problems follow by reduction

Question: what is the best possible $c$?
  - Under ETH, $c = \Omega(1)
  - In this talk, we didn’t optimize $c$
  - With some more work, we can reduce $c = 2$

SVP: improves previous $c < 2.5$, deterministically!

CVP: First $2^{O(n)}$ time algorithm, and first asymptotic improvement since [K’87]
Open Problems

- Practical barrier in lattice cryptography:
  - Evaluate appropriate key size to achieve security
  - Current state of the art lattice reduction algorithms are poorly understood
  - Problem: find better, practical lattice algorithms that allow to extrapolate running time/complexity of approximation to very high dimension

- Reduce space complexity to polynomial
- Design polynomial time CVPP approximation algorithms based on approximate Voronoi cell
- Extend to $\ell_\infty$
  - Most relevant norm for cryptanalysis
  - Application to Integer Programming

Question

Is the number of $\ell_\infty$-relevant points still bounded by $2^{O(n)}$