A Deterministic Single Exponential Time Algorithm for Most Lattice Problems based on Voronoi Cell Computations

Panagiotis Voulgaris    Daniele Micciancio

University of California, San Diego

June 7, 2010
Applications of lattice algorithms:

- Combinatorial problems:
  - Knapsack problems, Integer Programming, . . .

- Algebraic Number Theory:
  - Factoring polynomials with rational coefficients, . . .

- Geometry of numbers:
  - Packings, Coverings, . . .

- Coding Theory
Cryptanalysis applications:
- Special cases of RSA, DSA
- NtruSign, Knapsack based crypto, …

Lattice based cryptography:
- Average case-worst case connection [Ajtai, '96]
- LWE - Connection with learning problems [Regev, '05]
- Fully Homomorphic Encryption [Gentry, '09]
Advances in lattice algorithms?

Research on lattice algorithms:
- Not as fast moving as Lattice cryptography
- Heuristics preferable over provable algorithms
- Insufficient understanding, . . .

This work:
- Exact solution of SVP, CVP in \((2^{2n}, 2^n)\)
1 Introduction
   - Definitions
     - Known algorithms

2 Contribution
   - Overview
   - CVP given Voronoi Cell

3 Final Remarks
   - Conclusions, Future Work
Lattices:

- Let a linearly indep. basis: \( \mathbf{B} = \{ \vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n \} \)
Lattices:

- Let a linearly indep. basis: \( \mathbf{B} = \{ \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \} \)
- *Lattice* is the closure of \( \mathbf{B} \) under \((+,-)\):
  \[
  \Lambda(\mathbf{B}) = \left\{ \sum a_i \cdot \mathbf{b}_i, a_i \in \mathbb{Z} \right\}
  \]
Lattices:

- Let a linearly indep. basis: \( \mathbf{B} = \{ \vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n \} \)
- Lattice is the closure of \( \mathbf{B} \) under \((+,-)\):
  \( \Lambda(\mathbf{B}) = \{ \sum a_i \cdot \vec{b}_i, a_i \in \mathbb{Z} \} \)
- Lower rank sublattices:
  \( \Lambda_k = \Lambda(\vec{b}_1, \ldots, \vec{b}_k), k \leq n \)
Lattices:

- Let a linearly indep. basis: $B = \{ \vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n \}$
- Lattice is the closure of $\mathcal{B}$ under $(+, -)$:
  \[ \Lambda(\mathcal{B}) = \{ \sum a_i \cdot \vec{b}_i, a_i \in \mathbb{Z} \} \]
- Lower rank sublattices: $\Lambda_k = \Lambda(\vec{b}_1, \ldots, \vec{b}_k), k \leq n$
- Notice that the basis is not unique
Shortest Vector Problem (SVP):

- **Input:** A lattice $\Lambda(B)$

- **Output:** A shortest nonzero vector $\vec{s} \in \Lambda$
Shortest Vector Problem (SVP):  
- Input: A lattice $\Lambda(B)$  
- Output: A shortest nonzero vector $\vec{s} \in \Lambda$
Shortest Vector Problem (SVP):
- Input: A lattice $\Lambda(\mathbf{B})$
- Output: A shortest nonzero vector $\mathbf{s} \in \Lambda$
- Basis vectors can be exponentially larger than $\mathbf{s}$
Closest Vector Problem (CVP):

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

Panagiotis Voulgaris, Daniele Micciancio

SVP, CVP in deterministic $2^{\tilde{c}n}$ time
Closest Vector Problem (CVP):
- Input: A lattice $\Lambda(\mathbf{B})$, and a target vector $\vec{t}$
- Output: A closest lattice point $\vec{c} \in \Lambda$
1 Introduction
   - Definitions
   - Known algorithms

2 Contribution
   - Overview
   - CVP given Voronoi Cell

3 Final Remarks
   - Conclusions, Future Work
Rank reduction: CVP in $\Lambda_n \Rightarrow 2^n$ CVPs in $\Lambda_{n-1}$

Rank reduction for CVP:

- Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + c\vec{b}_n$, $c \in \mathbb{Z}$
Rank reduction: CVP in \( \Lambda_n \Rightarrow 2^n \) CVPs in \( \Lambda_{n-1} \)

Rank reduction for CVP:
- Partition \( \Lambda_n \) into layers of the form: \( \Lambda_{n-1} + c \vec{b}_n, c \in \mathbb{Z} \)
- The CVP of \( \vec{t} \) is in one of \( 2^n \) nearby layers
Rank reduction: CVP in $\Lambda_n \Rightarrow 2^n$ CVPs in $\Lambda_{n-1}$

Rank reduction for CVP:

- Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + cb_n$, $c \in \mathbb{Z}$
- The CVP of $\vec{t}$ is in one of $2^n$ nearby layers
- For each layer solve CVP in $\Lambda_{n-1}$ for the projection of $\vec{t}$
Rank reduction: CVP in $\Lambda_n \Rightarrow 2^n$ CVPs in $\Lambda_{n-1}$

**Rank reduction for CVP:**

- Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + c\vec{b}_n$, $c \in \mathbb{Z}$
- The CVP of $\vec{t}$ is in one of $2^n$ nearby layers
- For each layer solve CVP in $\Lambda_{n-1}$ for the projection of $\vec{t}$
Rank reduction: CVP in $\Lambda_n \Rightarrow 2^n$ CVPs in $\Lambda_{n-1}$

Rank reduction for CVP:
- Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + cb_n$, $c \in \mathbb{Z}$
- The CVP of $\vec{t}$ is in one of $2^n$ nearby layers
- For each layer solve CVP in $\Lambda_{n-1}$ for the projection of $\vec{t}$
- Keep the solution closer to $\vec{t}$
**Rank reduction: CVP in $\Lambda_n \Rightarrow 2^n$ CVPs in $\Lambda_{n-1}$**

Rank reduction for CVP:

- Partition $\Lambda_n$ into layers of the form: $\Lambda_{n-1} + c\mathbf{b}_n$, $c \in \mathbb{Z}$
- The CVP of $\mathbf{t}$ is in one of $2^n$ nearby layers
- For each layer solve CVP in $\Lambda_{n-1}$ for the projection of $\mathbf{t}$
- Keep the solution closer to $\mathbf{t}$

We have the reduction:

\[ \text{CVP in } \Lambda_n \rightarrow 2^n \cdot \text{CVPs in } \Lambda_{n-1} \]
2\(\tilde{c}n \log n\) time algorithm for SVP, CVP [Kannan]

Using rank reduction:

- Recursion gives a \(2^{\tilde{c}n^2}\) algorithm for CVP

\[\text{CVP in } \Lambda_n \rightarrow 2^n \rightarrow \text{CVPs in } \Lambda_{n-1} \rightarrow \ldots \rightarrow \prod_{k=1}^{n} 2^k \rightarrow \text{CVPs in } \Lambda_1\]

- Improved to \(2^{\tilde{c}n \log n}\) [Kannan]
- Similar technique solves SVP
- Best result for solving CVP, SVP in poly space
With $2^{\tilde{c}n}$ space [Ajtai, Kumar, Sivakumar '01]:

- First $(2^{\tilde{c}n}, 2^{\tilde{c}n})$ time algorithm for SVP
  - Best analysis $(2^{2.46n}, 2^{1.23n})$ [PS], [MV], [NV]

Notice that:

- Does not solve CVP
- Non-deterministic
1 Introduction
   • Definitions
   • Known algorithms

2 Contribution
   • Overview
   • CVP given Voronoi Cell

3 Final Remarks
   • Conclusions, Future Work
The main contributions of this work:

- Solves deterministically SVP, CVP in $(2^{2n}, 2^n)$
- SVP: Faster than theoretical AKS
  was $(2^{2.46n}, 2^{1.23n})$ [PS]
- CVP: First $2^{\tilde{c}n}$ time algorithm
  was $2^{\tilde{c}n \log n}$ [Kannan]
- Introduces a novel way to use Voronoi cells
Revisiting rank reduction

Recall rank reduction:

$$\text{CVP in } \Lambda_n \rightarrow 2^n \cdot \text{CVPs in } \Lambda_{n-1}$$

The $2^n$ CVP instances are on the same lattice $\Lambda_{n-1}$

Idea: Can we compute a hint for $\Lambda_{n-1}$ that allows faster CVP computations?

Yes, the voronoi cell of $\Lambda_{n-1}$ is such a hint
Voronoi Cell $\mathcal{V}$ is the set of points closer to 0 than any other lattice point:

$$\{ \vec{p} : \forall \vec{u} \in \Lambda, \| \vec{p} \| \leq \| \vec{u} - \vec{p} \| \}$$
Voronoi Cell \( \mathcal{V} \) is the set of points closer to 0 than any other lattice point:
\[
\{ \vec{p} : \forall \vec{u} \in \Lambda, \|\vec{p}\| \leq \|\vec{u} - \vec{p}\| \}
\]

How do we describe \( \mathcal{V} \)?
Given $\vec{u}_1 \in \Lambda$ consider the halfspace closer to $\vec{0}$. The minimum subset $R$ of $\Lambda$ that defines $V$ is the set of relevant points. Each relevant point defines an $n-1$ dimensional facet of $V$. Let $\text{Vor}(\Lambda)$ the problem of finding the set $R$. Panagiotis Voulgaris, Daniele Micciancio

SVP, CVP in deterministic $2^{\tilde{c}n}$ time
Given $\vec{u}_1 \in \Lambda$ consider the halfspace closer to $\vec{0}$

$\cal{V}$ is the intersection of all these halfspaces
Given \( \vec{u}_1 \in \Lambda \) consider the halfspace closer to \( \vec{0} \).

\( \mathcal{V} \) is the intersection of all these halfspaces.
Given $\vec{u}_1 \in \Lambda$ consider the halfspace closer to $\vec{0}$.

$\mathcal{V}$ is the intersection of all these halfspaces.
Given $\vec{u}_1 \in \Lambda$ consider the halfspace closer to $\vec{0}$

$\mathcal{V}$ is the intersection of all these halfspaces
Given $\vec{u}_1 \in \Lambda$ consider the halfspace closer to $\vec{0}$.

$\mathcal{V}$ is the intersection of all these halfspaces.

The minimum subset $R$ of $\Lambda$ that defines $\mathcal{V}$ is the set of relevant points.
Given $\vec{u}_1 \in \Lambda$ consider the halfspace closer to $\vec{0}$

$\mathcal{V}$ is the intersection of all these halfspaces

The minimum subset $R$ of $\Lambda$ that defines $\mathcal{V}$ is the set of relevant points

Each relevant point defines a $n-1$ dimensional facet of $\mathcal{V}$
• Given $\vec{u}_1 \in \Lambda$ consider the halfspace closer to $\vec{0}$

• $\mathcal{V}$ is the intersection of all these halfspaces

• The minimum subset $R$ of $\Lambda$ that defines $\mathcal{V}$ is the set of **relevant points**

• Each relevant point defines a n-1 dimensional facet of $\mathcal{V}$

• Let $\text{Vor}(\Lambda)$ the problem of finding the set $R$
\[ \text{Vor}(\Lambda_n) \rightarrow 2^n \text{ CVPs in } \Lambda_n \]

- \( R \) has at most \( 2^n - 1 \) pairs of points [Voronoi]

Panagiotis Voulgaris, Daniele Micciancio

SVP, CVP in deterministic \( 2^{\tilde{c}n} \) time
$\text{Vor}(\Lambda_n) \rightarrow 2^n \text{ CVPs in } \Lambda_n$

- $R$ has at most $2^n - 1$ pairs of points [Voronoi]
- Each pair can be found with a CVP computation in $\Lambda_n$
\( \text{Vor}(\Lambda_n) \rightarrow 2^n \text{ CVPs in } \Lambda_n \)

- \( R \) has at most \( 2^n - 1 \) pairs of points [Voronoï]
- Each pair can be found with a CVP computation in \( \Lambda_n \)
- We have the reduction:

\[
\text{Vor}(\Lambda_n) \rightarrow 2^n \cdot \text{CVPs in } \Lambda_n
\]
We will show that given $\mathcal{V}$, CVP can be computed in $2^{2n}$:

$$\text{CVP in } \Lambda_n \rightarrow \left( \text{Vor}(\Lambda_n) + 2^{2n} \right)$$
Outline of our result

Recall the reductions:

1. $\text{Vor}(\Lambda_n) \rightarrow 2^n$.
2. $\text{CVPs in } \Lambda_n \rightarrow 2^n$.
3. $\text{CVP in } \Lambda_n \rightarrow (\text{Vor}(\Lambda_n) + 2^n)$.

We compute $\text{Vor}(\Lambda_n)$:
Outline of our result

Recall the reductions:

1. $\text{Vor}(\Lambda_n) \rightarrow 2^n \cdot \text{CVPs in } \Lambda_n$

2. $\text{CVP in } \Lambda_n \rightarrow 2^n \cdot \text{CVPs in } \Lambda_{n-1}$

3. $\text{CVP in } \Lambda_n \rightarrow (\text{Vor}(\Lambda_n) + 2^n)$

We compute $\text{Vor}(\Lambda_n)$:

$\text{Vor}(\Lambda_n) \rightarrow 2^n \cdot \text{CVPs in } \Lambda_n$
Recall the reductions:

1. $\text{Vor}(\Lambda_n)$
   $\xrightarrow{2^n} \text{CVPs in } \Lambda_n$

2. $\text{CVP in } \Lambda_n$
   $\xrightarrow{2^n} \text{CVPs in } \Lambda_{n-1}$

3. $\text{CVP in } \Lambda_n$
   $(\text{Vor}(\Lambda_n) + 2^n)$

We compute $\text{Vor}(\Lambda_n)$:

$\text{Vor}(\Lambda_n)$
$\xrightarrow{2^n} \text{CVPs in } \Lambda_n$
$\xrightarrow{2^{2n}} \text{CVPs in } \Lambda_{n-1}$
Outline of our result

Recall the reductions:

1. $\text{Vor}(\Lambda_n)$ \rightarrow $2^n$
2. CVPs in $\Lambda_n$ \rightarrow $2^n$
3. CVP in $\Lambda_n$ \rightarrow (CVP in $\Lambda_n$) + $2^n$

We compute $\text{Vor}(\Lambda_n)$:

1. $\text{Vor}(\Lambda_n)$ \rightarrow $2^n$
2. CVPs in $\Lambda_n$ \rightarrow $2^{2n}$
3. CVPs in $\Lambda_{n-1}$ \rightarrow $2^{2n}$
4. (Vor ($\Lambda_{n-1}$) + $2^n$)
Outline of our result

Recall the reductions:

1. \( \text{Vor}(\Lambda_n) \rightarrow 2^n \cdot \text{CVPs in } \Lambda_n \)
2. \( \text{CVP in } \Lambda_n \rightarrow 2^n \cdot \text{CVPs in } \Lambda_{n-1} \)
3. \( \text{CVP in } \Lambda_n \rightarrow (\text{Vor}(\Lambda_n) + 2^n) \)

We compute \( \text{Vor}(\Lambda_n) \):

\( \text{Vor}(\Lambda_n) \rightarrow 2^n \cdot \text{CVPs in } \Lambda_n \rightarrow 2^{2n} \cdot \text{CVPs in } \Lambda_{n-1} \rightarrow 2^{2n} \cdot (\text{Vor}(\Lambda_{n-1}) + 2^n) \)

So we have:

\( \text{Vor}(\Lambda_n) \rightarrow (\text{Vor}(\Lambda_{n-1}) + 2^{4n}) \)
Finally solve $\text{Vor}(\Lambda_n)$ in $2^{4n}$:

\[
\text{Vor}(\Lambda_n) \rightarrow \left(\text{Vor}(\Lambda_{n-1}) + 2^{4n}\right) \rightarrow \cdots \rightarrow \left(\text{Vor}(\Lambda_1) + \sum_{k=1}^{n} 2^{4k}\right)
\]

Previously we had $2^{\tilde{c}n^2}$:

\[
\text{CVP in } \Lambda_n \rightarrow 2^n \cdot \text{CVPs in } \Lambda_{n-1} \rightarrow \cdots \rightarrow \prod_{k=1}^{n} 2^k \cdot \text{CVPs in } \Lambda_1
\]
1 Introduction
   • Definitions
   • Known algorithms

2 Contribution
   • Overview
     • CVP given Voronoi Cell

3 Final Remarks
   • Conclusions, Future Work
Given a target $\vec{t}$, find $\vec{c} \in \Lambda$ such that: $\vec{t} - \vec{c} \in \mathcal{V}$
Given a target $\vec{t}$, find $\vec{c} \in \Lambda$ such that: $\vec{t} - \vec{c} \in \mathcal{V}$

Equivalently:
Given a target $\vec{t}$, find $\vec{c} \in \Lambda$ such that: $\vec{t} - \vec{c} \in \mathcal{V}$

Equivalently:

- From all the points in $\vec{t} - \Lambda$,
Given a target $\vec{t}$, find $\vec{c} \in \Lambda$ such that: $\vec{t} - \vec{c} \in \mathcal{V}$

Equivalently:
- From all the points in $\vec{t} - \Lambda$,
- find a point in $\mathcal{V}$
Voronoi Cell and CVP

Given a target \( \vec{t} \), find \( \vec{c} \in \Lambda \) such that: \( \vec{t} - \vec{c} \in \mathcal{V} \)

Equivalently:

- From all the points in \( \vec{t} - \Lambda \), find a point in \( \mathcal{V} \)

Strategy: Bring \( \vec{t} \) in \( \mathcal{V} \) by subtracting relevant points

- First we solve for \( \vec{t} \in 2\mathcal{V} \),
- and then for any \( \vec{t} \)
Given a target $\vec{t} \in 2\mathcal{V}$, find $\vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V}$:
Given a target $\vec{t} \in 2\mathcal{V}$, find $\vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V}$:

- Scale $\mathcal{V}$ by $k \in \mathbb{R}$ s.t. $\vec{t}$ is on a facet of $k\mathcal{V}$
Given a target \( \vec{t} \in 2\mathcal{V} \), find \( \vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V} \):

- Scale \( \mathcal{V} \) by \( k \in \mathbb{R} \) s.t. \( \vec{t}' \) is on a facet of \( k\mathcal{V} \)
- Subtract the relevant vector corresponding to that facet

\[ \text{Panagiotis Voulgaris, Daniele Micciancio} \]

\[ \text{SVP, CVP in deterministic } 2^{\tilde{c}n} \text{ time} \]
Given a target $\vec{t} \in 2\mathcal{V}$, find $\vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V}$:

- Scale $\mathcal{V}$ by $k \in \mathbb{R}$ s.t. $\vec{t}$ is on a facet of $k\mathcal{V}$
- Subtract the relevant vector corresponding to that facet
- The new vector $\vec{t}'$ is shorter but still in $2\mathcal{V}$
Given a target $\vec{t} \in 2\mathcal{V}$, find $\vec{t}' \in \vec{t} - \Lambda \cap \mathcal{V}$:

- Scale $\mathcal{V}$ by $k \in \mathbb{R}$ s.t. $\vec{t}$ is on a facet of $k\mathcal{V}$
- Subtract the relevant vector corresponding to that facet
- The new vector $\vec{t}'$ is shorter but still in $2\mathcal{V}$
- $| (\vec{t} - \Lambda) \cap 2\mathcal{V} | \leq 2^n$, so after $2^n$ steps CVP is found
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 $2^{k-1} \mathcal{V}$
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$_{2\mathcal{V}}$ 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$_{2^k \mathcal{V}}$ 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\(_{2^k} \) 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\(_{2^k} \) to go from \( 2^k \mathcal{V} \) to \( 2^{k-1} \mathcal{V} \)

Panagiotis Voulgaris, Daniele Micciancio
SVP, CVP in deterministic \( 2^{\tilde{c}n} \) 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 \( \text{CVP}_{2^k} \) to go from \( 2^k \mathcal{V} \) to \( 2^{k-1} \mathcal{V} \)

Panagiotis Voulgaris, Daniele Micciancio

SVP, CVP in deterministic \( 2^{\tilde{c}n} \) time
1. **Introduction**
   - Definitions
   - Known algorithms

2. **Contribution**
   - Overview
   - CVP given Voronoi Cell

3. **Final Remarks**
   - Conclusions, Future Work
Conclusions, Future Work

Conclusions:
- SVP, CVP can be solved deterministicall in \(2^{2n}, 2^n\)
- Connection between voronoi cells and CVP

Open problems:
- Reduce space complexity?
- CVP in other norms \(\ell_1, \ell_\infty\)? (Connection with IP)
- Faster variant only for SVP?
Thank you!

(: Thank you! :)