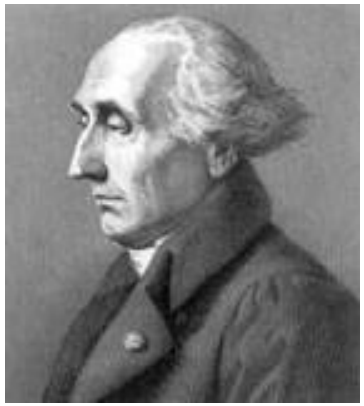
Algorithms for the Densest Sublattice Problem

Daniele Micciancio (UCSD)
(Joint work with D. Dadush – SODA 2013)

January 2013

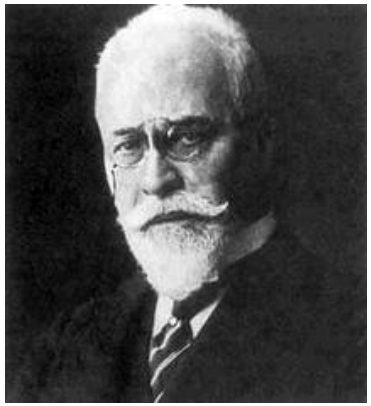
(Point) Lattices

- Traditional area of mathematics



Lagrange

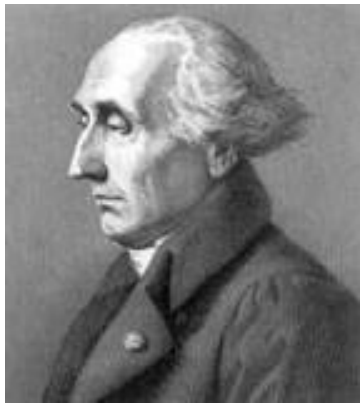
Gauss



Minkowski

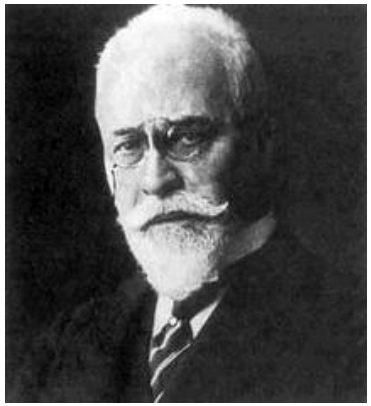
(Point) Lattices

- Traditional area of mathematics



Lagrange

Gauss

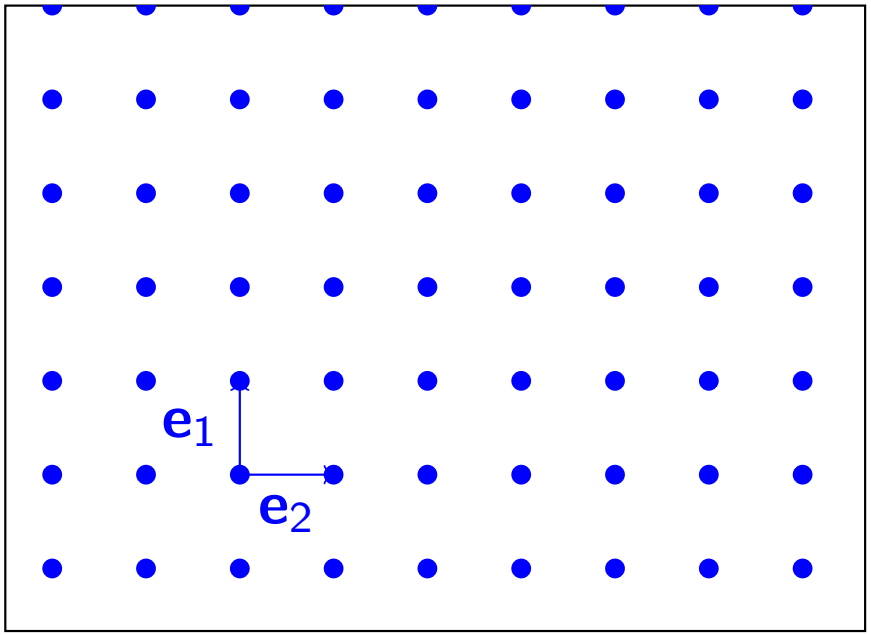


Minkowski

- Key to many algorithmic applications
  - Cryptanalysis (e.g., breaking low-exponent RSA)
  - Coding Theory (e.g., wireless communications)
  - Optimization (e.g., Integer Programming with fixed number of variables)
  - Cryptography (e.g., Cryptographic functions from worst-case complexity assumptions, Fully Homomorphic Encryption)

Outline

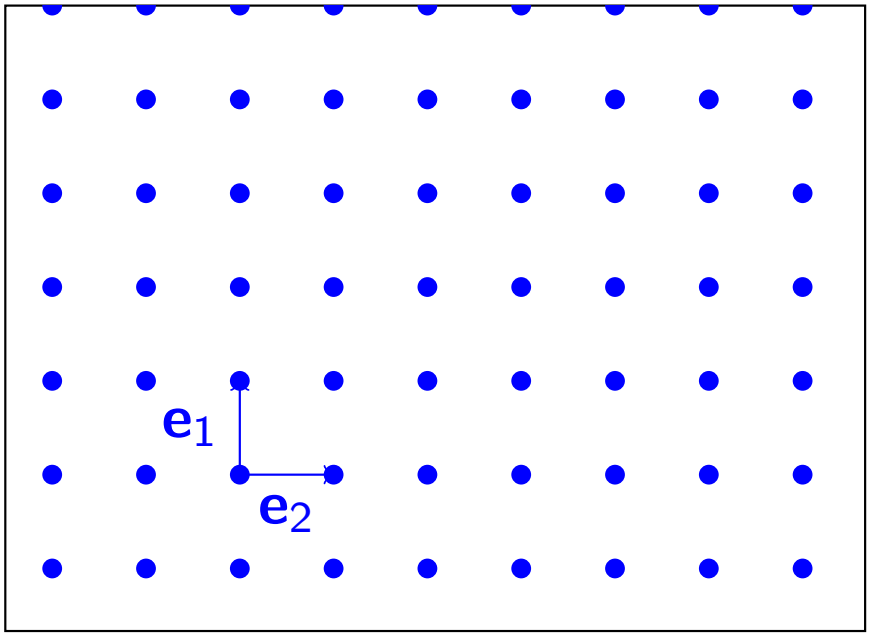
Lattices: Definition



The simplest lattice in n-dimensional space is the integer lattice

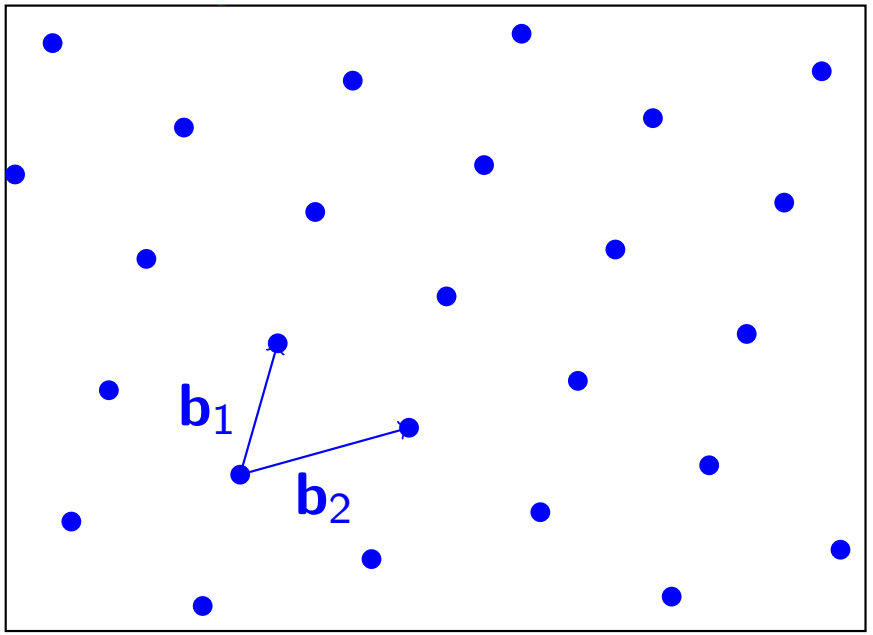
$$\Lambda = \mathbb{Z}^n$$

Lattices: Definition



The simplest lattice in n-dimensional space is the integer lattice

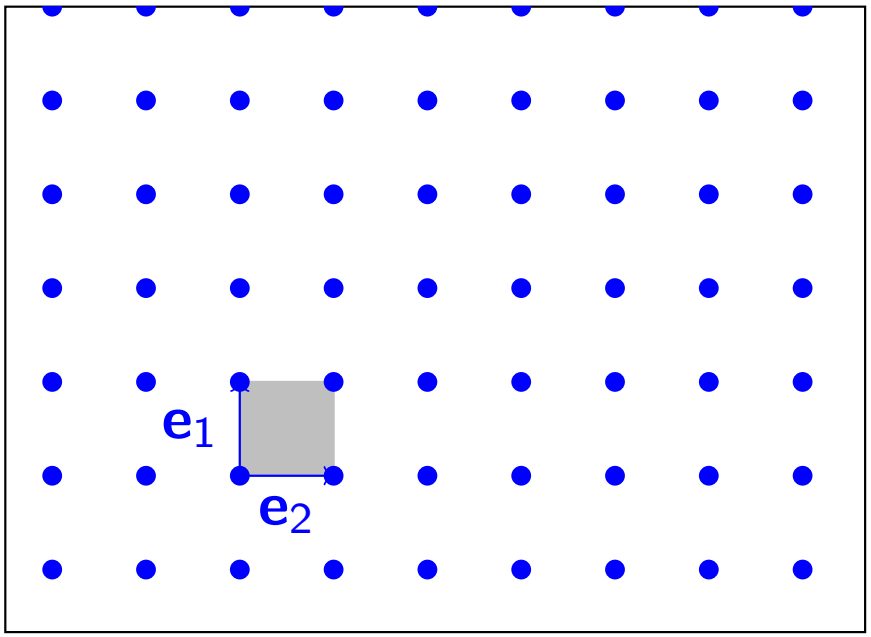
$$\Lambda = \mathbb{Z}^n$$

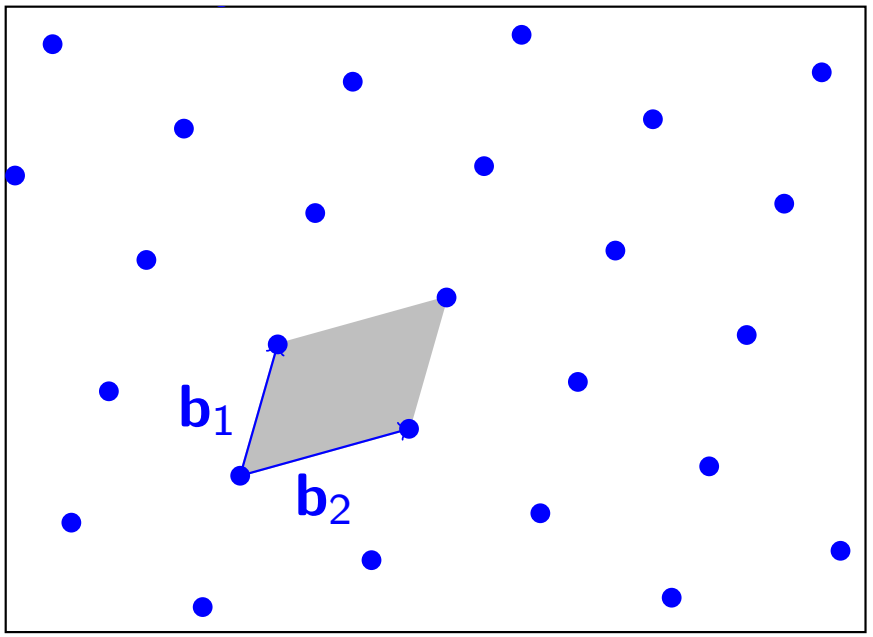


Other lattices are obtained by applying a linear transformation

$$\Lambda = \mathbf{B}\mathbb{Z}^n \qquad (\mathbf{B} \in \mathbb{R}^{d \times n})$$

Lattice Determinant / Density





Definition (Determinant)

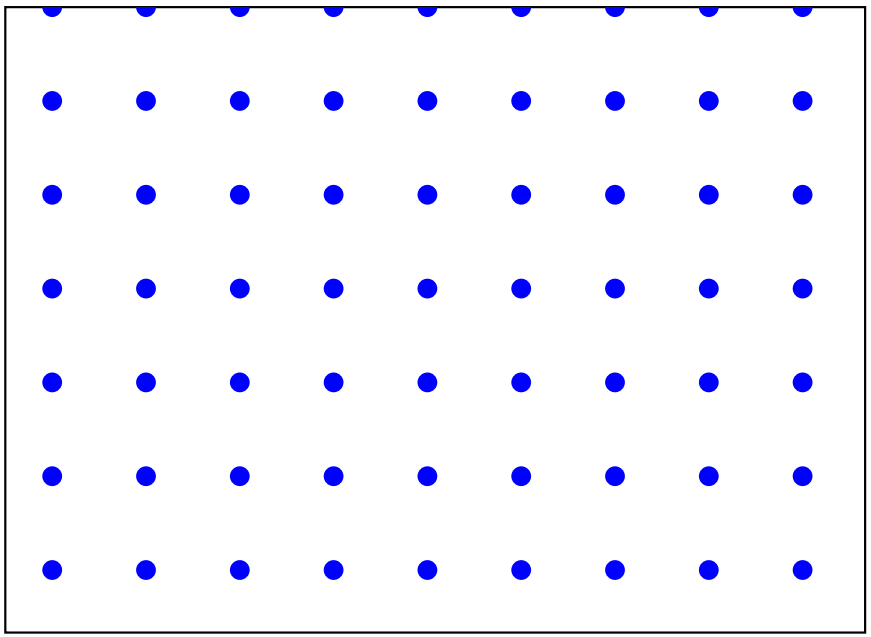
The determinant of a lattice is the volume of a fundamental region

$$\det(\mathbf{B}\mathbb{Z}^n) = \mathrm{vol}_n\left(\mathbf{B}[0,1)^n\right) = \frac{1}{\mathrm{density}(\Lambda)}$$

The Densest Sublattice Problem (DSP)

Definition (Densest Sublattice Problem (k-DSP))

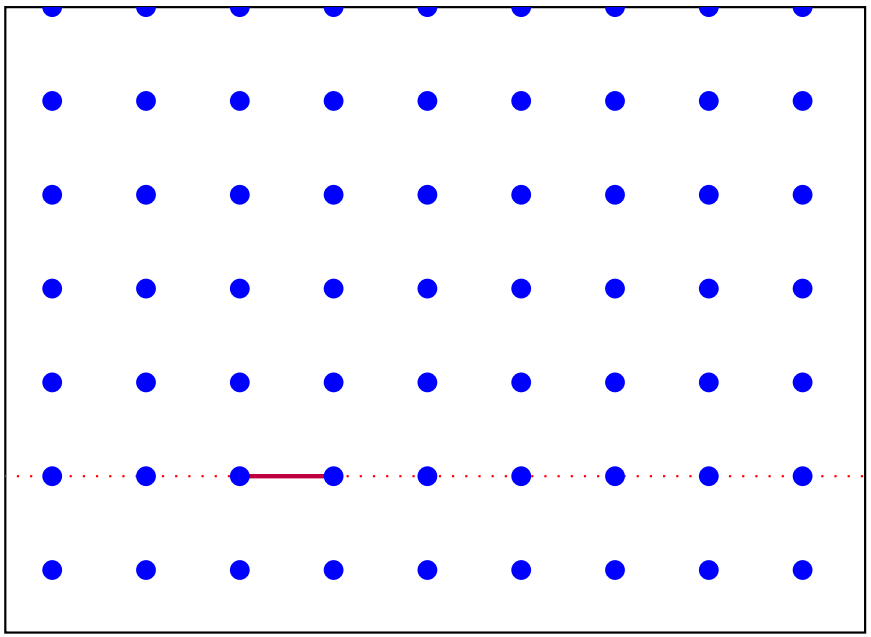
Given a lattice Λ, find a k-dimensional sublattice $\Lambda' \subseteq \Lambda$ that minimizes $\det(\Lambda')$.



The Densest Sublattice Problem (DSP)

Definition (Densest Sublattice Problem (k-DSP))

Given a lattice Λ, find a k-dimensional sublattice $\Lambda' \subseteq \Lambda$ that minimizes $\det(\Lambda')$.



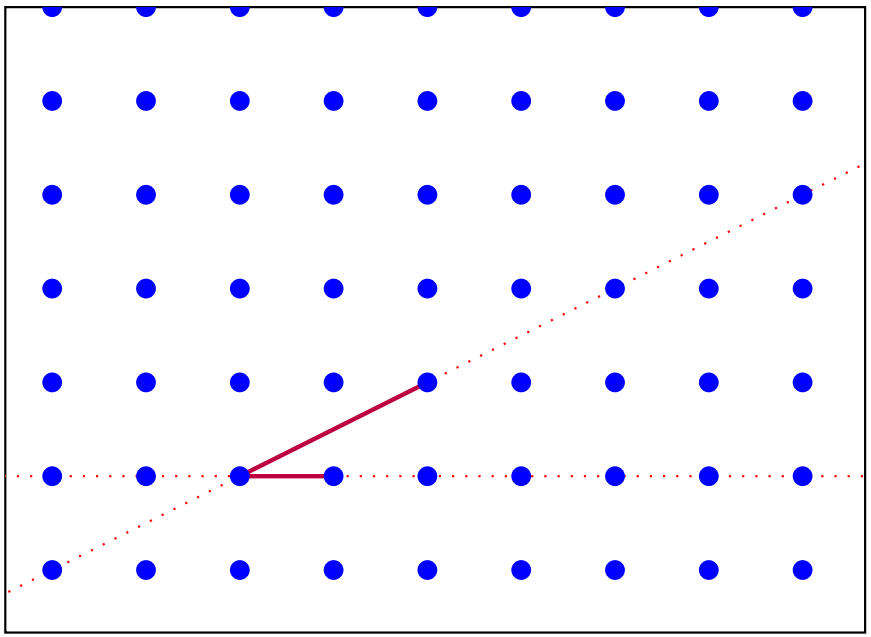
- $\Lambda' = \Lambda \cap S$, $\dim(S) = k$

$$\Lambda' = \mathbf{b}\mathbb{Z} \text{ and } \det(\Lambda') = \|\mathbf{b}\|$$

The Densest Sublattice Problem (DSP)

Definition (Densest Sublattice Problem (k-DSP))

Given a lattice Λ, find a k-dimensional sublattice $\Lambda' \subseteq \Lambda$ that minimizes $\det(\Lambda')$.



- $\Lambda' = \Lambda \cap S$, $\dim(S) = k$

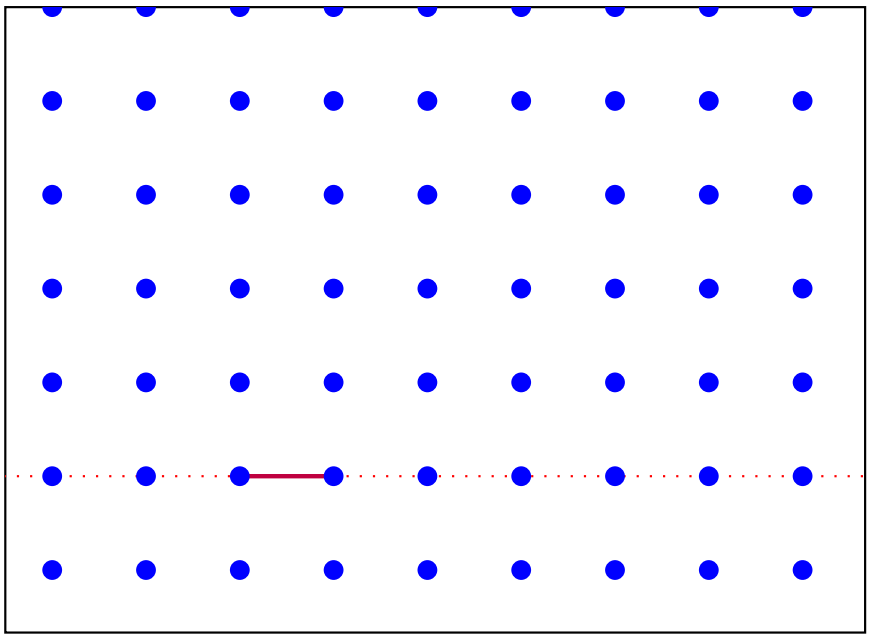
  $\Lambda' = \mathbf{b}\mathbb{Z}$ and $\det(\Lambda') = \|\mathbf{b}\|$

- Small det $\Leftrightarrow$ High density

The Densest Sublattice Problem (DSP)

Definition (Densest Sublattice Problem (k-DSP))

Given a lattice Λ, find a k-dimensional sublattice $\Lambda' \subseteq \Lambda$ that minimizes $\det(\Lambda')$.



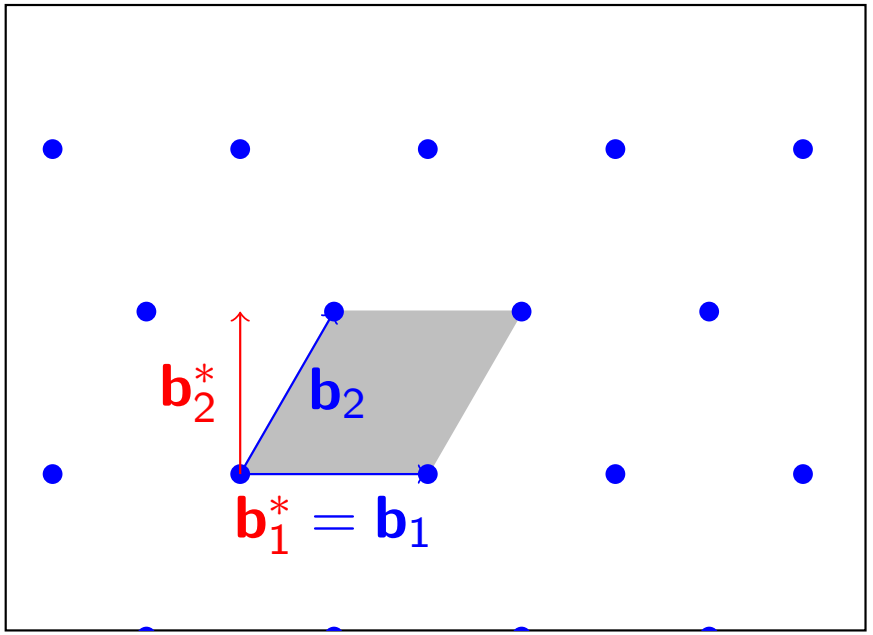
- $\Lambda' = \Lambda \cap S$, $\dim(S) = k$

  $\Lambda' = \mathbf{b}\mathbb{Z}$ and $\det(\Lambda') = \|\mathbf{b}\|$

- Small det $\Leftrightarrow$ High density
- 1-DSP = SVP
  (Shortest Vector Problem)

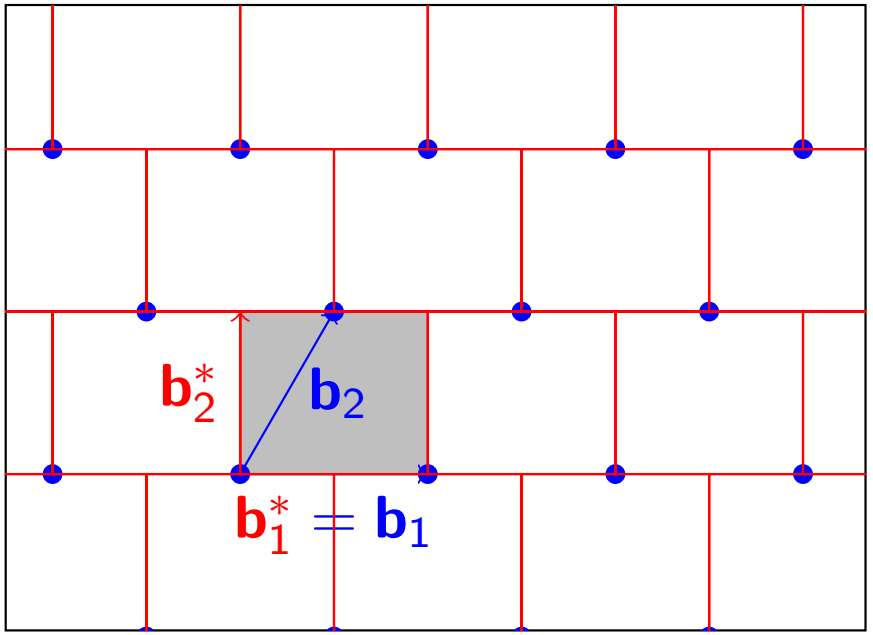
Lattice rounding

- Gram-Schmidt orthogonalization $\mathbf{B}^*[0,1)^n$



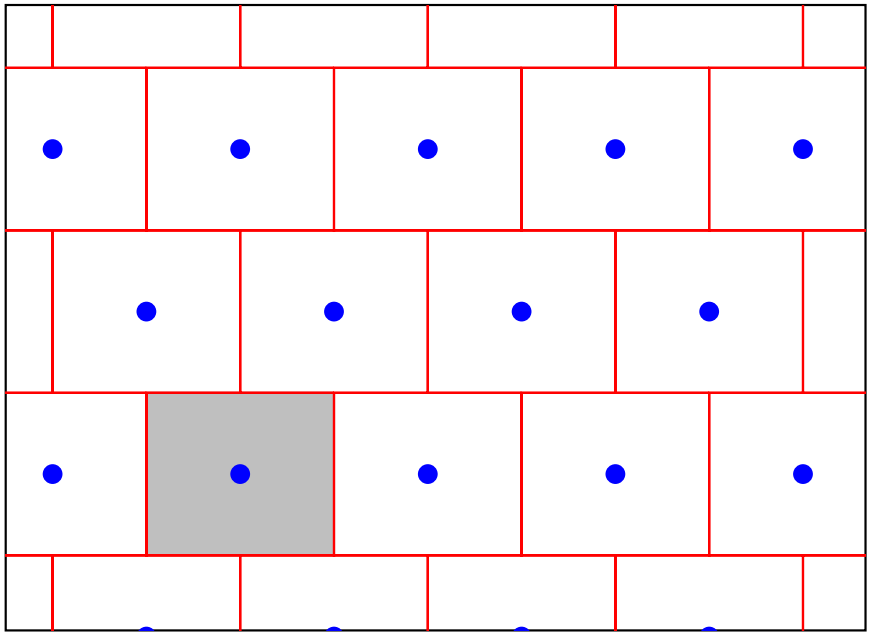
Lattice rounding

- Gram-Schmidt orthogonalization $\mathbf{B}^*[0,1)^n$ is also a fundamental region for Λ



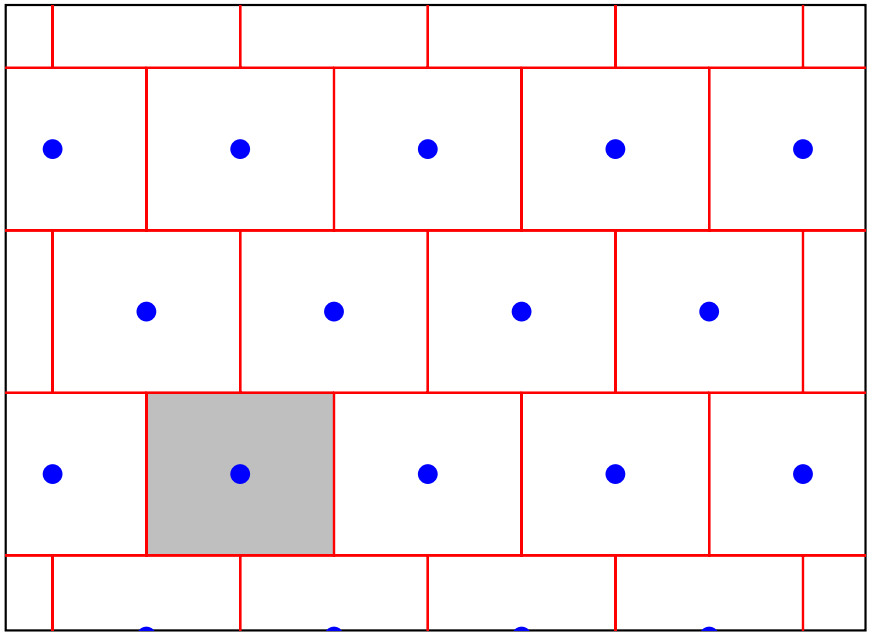
Lattice rounding

- Gram-Schmidt orthogonalization $\mathbf{B}^*[0,1)^n$ is also a fundamental region for Λ



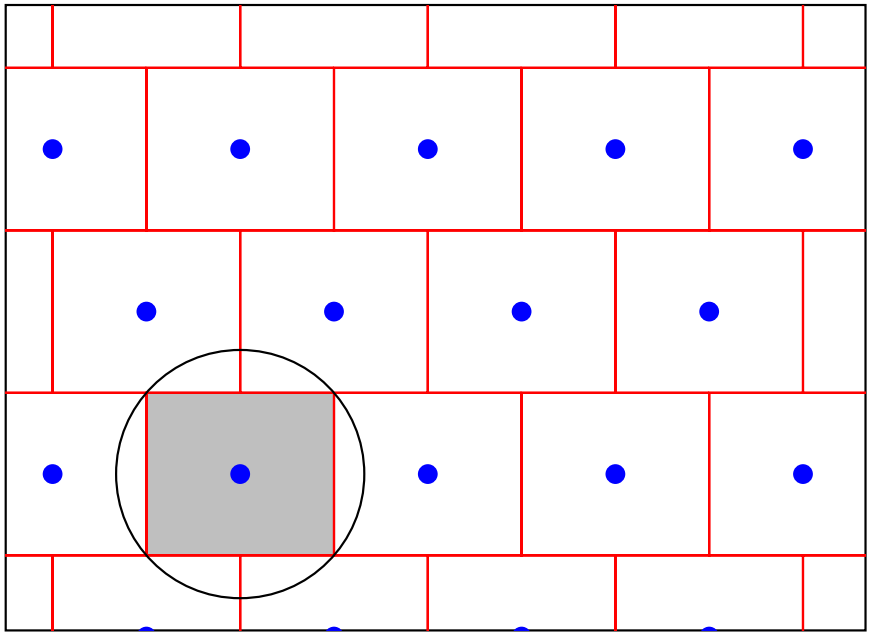
Lattice rounding

- Gram-Schmidt orthogonalization $\mathbf{B}^*[0,1)^n$ is also a fundamental region for Λ
- Any $\mathbf{t}$ can be efficiently rounded to $\mathbf{v} \in \Lambda$



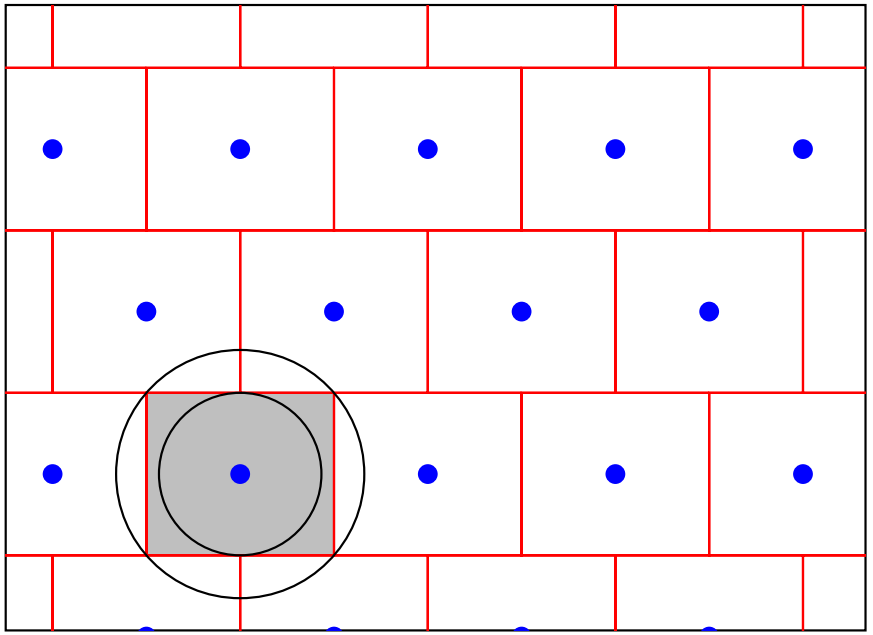
Lattice rounding

- Gram-Schmidt orthogonalization $\mathbf{B}^*[0,1)^n$ is also a fundamental region for Λ
- Any $\mathbf{t}$ can be efficiently rounded to $\mathbf{v} \in \Lambda$
- $\|\mathbf{t} - \mathbf{v}\| \leq \frac{1}{2}\sqrt{\sum_i \|\mathbf{b}_i^*\|^2}$



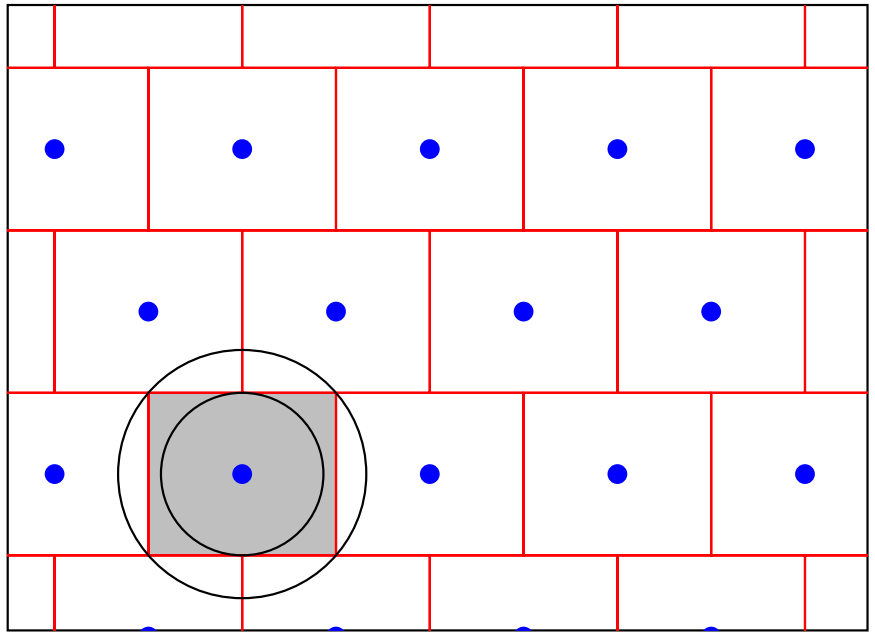
Lattice rounding

- Gram-Schmidt orthogonalization $\mathbf{B}^*[0,1)^n$ is also a fundamental region for Λ
- Any $\mathbf{t}$ can be efficiently rounded to $\mathbf{v} \in \Lambda$
- $\|\mathbf{t} - \mathbf{v}\| \leq \frac{1}{2}\sqrt{\sum_i \|\mathbf{b}_i^*\|^2}$
- $\mathbf{v}$ solves CVP when $\|\mathbf{t} - \mathbf{v}\| \leq \min \|\mathbf{b}_i^*\|/2$



Lattice rounding

- Gram-Schmidt orthogonalization $\mathbf{B}^*[0,1)^n$ is also a fundamental region for Λ
- Any $\mathbf{t}$ can be efficiently rounded to $\mathbf{v} \in \Lambda$
- $\|\mathbf{t} - \mathbf{v}\| \leq \frac{1}{2}\sqrt{\sum_i \|\mathbf{b}_i^*\|^2}$
- $\mathbf{v}$ solves CVP when $\|\mathbf{t} - \mathbf{v}\| \leq \min \|\mathbf{b}_i^*\|/2$



Lemma (Nearest Plane Algorithm [Babai 1986])

Rounding w.r.t $\mathbf{B}^*$ *approximates CVP within* $\sqrt{n} \cdot \frac{\max_i \|\mathbf{b}_i^*\|}{\min_i \|\mathbf{b}_i^*\|}$

Basis reduction

Definition (Basis reduction problem)

Given a lattice, find a basis such that $\|\mathbf{b}_i^*\| \approx \det(\Lambda)^{1/n}$, or, more generally, the $\|\mathbf{b}_i^*\|$ do not decrease too quickly.

Basis reduction

Definition (Basis reduction problem)

Given a lattice, find a basis such that $\|\mathbf{b}_i^*\| \approx \det(\Lambda)^{1/n}$, or, more generally, the $\|\mathbf{b}_i^*\|$ do not decrease too quickly.

- Sort $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \ldots \leq \|\mathbf{b}_n\|$

Basis reduction

Definition (Basis reduction problem)

Given a lattice, find a basis such that $\|\mathbf{b}_i^*\| \approx \det(\Lambda)^{1/n}$, or, more generally, the $\|\mathbf{b}_i^*\|$ do not decrease too quickly.

- Sort $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \ldots \leq \|\mathbf{b}_n\|$
- Still, typically $\|\mathbf{b}_1^*\| > \|\mathbf{b}_2^*\| > \ldots > \|\mathbf{b}_n^*\|$

Basis reduction

Definition (Basis reduction problem)

Given a lattice, find a basis such that $\|\mathbf{b}_i^*\| \approx \det(\Lambda)^{1/n}$, or, more generally, the $\|\mathbf{b}_i^*\|$ do not decrease too quickly.

- Sort $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \ldots \leq \|\mathbf{b}_n\|$
- Still, typically $\|\mathbf{b}_1^*\| > \|\mathbf{b}_2^*\| > \ldots > \|\mathbf{b}_n^*\|$
- This is unavoidable, even for $k = 2$, e.g., for "exagonal" lattice

$$\frac{\|\mathbf{b}_1\|}{\|\mathbf{b}_2^*\|} = \frac{\|\mathbf{b}_1\| \cdot \|\mathbf{b}_1\|}{\|\mathbf{b}_1\| \cdot \|\mathbf{b}_2^*\|} = \frac{\|\mathbf{b}_1\|^2}{\det(\Lambda)} \leq \gamma_2 = \frac{2}{\sqrt{3}} \approx 1.1547$$

Basis reduction

Definition (Basis reduction problem)

Given a lattice, find a basis such that $\|\mathbf{b}_i^*\| \approx \det(\Lambda)^{1/n}$, or, more generally, the $\|\mathbf{b}_i^*\|$ do not decrease too quickly.

- Sort $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \ldots \leq \|\mathbf{b}_n\|$
- Still, typically $\|\mathbf{b}_1^*\| > \|\mathbf{b}_2^*\| > \ldots > \|\mathbf{b}_n^*\|$
- This is unavoidable, even for $k = 2$, e.g., for "exagonal" lattice

$$\frac{\|\mathbf{b}_1\|}{\|\mathbf{b}_2^*\|} = \frac{\|\mathbf{b}_1\| \cdot \|\mathbf{b}_1\|}{\|\mathbf{b}_1\| \cdot \|\mathbf{b}_2^*\|} = \frac{\|\mathbf{b}_1\|^2}{\det(\Lambda)} \leq \gamma_2 = \frac{2}{\sqrt{3}} \approx 1.1547$$

- Minimizing $\|\mathbf{b}_1\|/\|\mathbf{b}_2^*\|$ is equivalent to SVP

Definition (Basis reduction problem)

Given a lattice, find a basis such that $\|\mathbf{b}_i^*\| \approx \det(\Lambda)^{1/n}$, or, more generally, the $\|\mathbf{b}_i^*\|$ do not decrease too quickly.

- Sort $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \ldots \leq \|\mathbf{b}_n\|$
- Still, typically $\|\mathbf{b}_1^*\| > \|\mathbf{b}_2^*\| > \ldots > \|\mathbf{b}_n^*\|$
- This is unavoidable, even for $k = 2$, e.g., for "exagonal" lattice

$$\frac{\|\mathbf{b}_1\|}{\|\mathbf{b}_2^*\|} = \frac{\|\mathbf{b}_1\| \cdot \|\mathbf{b}_1\|}{\|\mathbf{b}_1\| \cdot \|\mathbf{b}_2^*\|} = \frac{\|\mathbf{b}_1\|^2}{\det(\Lambda)} \leq \gamma_2 = \frac{2}{\sqrt{3}} \approx 1.1547$$

- Minimizing $\|\mathbf{b}_1\|/\|\mathbf{b}_2^*\|$ is equivalent to SVP
- Hemite constant:

$$\gamma_n = \sup_{\Lambda} \inf_{\mathbf{B}} \left(\frac{\|\mathbf{b}_1\|}{\det(\Lambda)^{1/n}} \right)^2 = \Theta(n)$$

LLL basis reduction algorithm

Theorem (Lenstra, Lenstra, Lovasz (LLL) 1982)

Every lattice has an efficiently computable basis such that $\|\mathbf{b}_{i+1}^*\| \geq \tilde{\gamma}_2 \cdot \|\mathbf{b}_i^*\|$ *for all* i*, and* $\frac{\max_i \|\mathbf{b}_i^*\|}{\min_i \|\mathbf{b}_i^*\|} = 2^{O(n)}$

LLL basis reduction algorithm

Theorem (Lenstra, Lenstra, Lovasz (LLL) 1982)

Every lattice has an efficiently computable basis such that $\|\mathbf{b}_{i+1}^*\| \geq \tilde{\gamma}_2 \cdot \|\mathbf{b}_i^*\|$ *for all* i*, and* $\frac{\max_i \|\mathbf{b}_i^*\|}{\min_i \|\mathbf{b}_i^*\|} = 2^{O(n)}$

- $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$

LLL basis reduction algorithm

Theorem (Lenstra, Lenstra, Lovasz (LLL) 1982)

Every lattice has an efficiently computable basis such that $\|\mathbf{b}_{i+1}^*\| \geq \tilde{\gamma}_2 \cdot \|\mathbf{b}_i^*\|$ *for all* i*, and* $\frac{\max_i \|\mathbf{b}_i^*\|}{\min_i \|\mathbf{b}_i^*\|} = 2^{O(n)}$

- $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$
- Locally modify each 2-dim sublattice $[\mathbf{b}_i, \mathbf{b}_{i+1}]$ so $\|\mathbf{b}_i^*\|$ is (almost) minimal

LLL basis reduction algorithm

Theorem (Lenstra, Lenstra, Lovasz (LLL) 1982)

Every lattice has an efficiently computable basis such that $\|\mathbf{b}_{i+1}^*\| \geq \tilde{\gamma}_2 \cdot \|\mathbf{b}_i^*\|$ *for all* i*, and* $\frac{\max_i \|\mathbf{b}_i^*\|}{\min_i \|\mathbf{b}_i^*\|} = 2^{O(n)}$

- $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$
- Locally modify each 2-dim sublattice $[\mathbf{b}_i, \mathbf{b}_{i+1}]$ so $\|\mathbf{b}_i^*\|$ is (almost) minimal
- LLL terminates because each local modification makes "progress" towards reducing the basis

LLL basis reduction algorithm

Theorem (Lenstra, Lenstra, Lovasz (LLL) 1982)

Every lattice has an efficiently computable basis such that $\|\mathbf{b}_{i+1}^*\| \geq \tilde{\gamma}_2 \cdot \|\mathbf{b}_i^*\|$ *for all* i*, and* $\frac{\max_i \|\mathbf{b}_i^*\|}{\min_i \|\mathbf{b}_i^*\|} = 2^{O(n)}$

- $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$
- Locally modify each 2-dim sublattice $[\mathbf{b}_i, \mathbf{b}_{i+1}]$ so $\|\mathbf{b}_i^*\|$ is (almost) minimal
- LLL terminates because each local modification makes "progress" towards reducing the basis
- Polynomial time termination for any $\tilde{\gamma}_2 > \gamma_2$

Generalized basis reduction algorithm

- Partition basis vectors into blocks $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_m]$

Generalized basis reduction algorithm

- Partition basis vectors into blocks $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_m]$

Block Basis Reduction Algorithm

Locally modify each pair of blocks $[\mathbf{B}_i, \mathbf{B}_{i+1}]$ so to minimize the product $\|\mathbf{b}_j^*\| \cdots \|\mathbf{b}_k^*\|$ corresponding to $\mathbf{B}_i = [\mathbf{b}_j, \ldots, \mathbf{b}_k]$

Generalized basis reduction algorithm

- Partition basis vectors into blocks $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_m]$

Block Basis Reduction Algorithm

Locally modify each pair of blocks $[\mathbf{B}_i, \mathbf{B}_{i+1}]$ so to minimize the product $\|\mathbf{b}_j^*\| \cdots \|\mathbf{b}_k^*\|$ corresponding to $\mathbf{B}_i = [\mathbf{b}_j, \ldots, \mathbf{b}_k]$

- Local modification is a $\dim(\mathbf{B}_i)$-DSP instance in dimension $\dim(\mathbf{B}_i) + \dim(\mathbf{B}_{i+1})$

Generalized basis reduction algorithm

- Partition basis vectors into blocks $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_m]$

Block Basis Reduction Algorithm

Locally modify each pair of blocks $[\mathbf{B}_i, \mathbf{B}_{i+1}]$ so to minimize the product $\|\mathbf{b}_j^*\| \cdots \|\mathbf{b}_k^*\|$ corresponding to $\mathbf{B}_i = [\mathbf{b}_j, \ldots, \mathbf{b}_k]$

- Local modification is a $\dim(\mathbf{B}_i)$-DSP instance in dimension $\dim(\mathbf{B}_i) + \dim(\mathbf{B}_{i+1})$
- LLL is a special case where $\dim(\mathbf{B}_i) = 1$, and 1-DSP $=$ SVP

Generalized basis reduction algorithm

- Partition basis vectors into blocks $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_m]$

Block Basis Reduction Algorithm

Locally modify each pair of blocks $[\mathbf{B}_i, \mathbf{B}_{i+1}]$ so to minimize the product $\|\mathbf{b}_j^*\| \cdots \|\mathbf{b}_k^*\|$ corresponding to $\mathbf{B}_i = [\mathbf{b}_j, \ldots, \mathbf{b}_k]$

- Local modification is a $\dim(\mathbf{B}_i)$-DSP instance in dimension $\dim(\mathbf{B}_i) + \dim(\mathbf{B}_{i+1})$
- LLL is a special case where $\dim(\mathbf{B}_i) = 1$, and 1-DSP $=$ SVP
- Sliding Reduction [Gama, Nguyen 2008] corresponds to $\dim(\mathbf{B}_{2i}) = 1$ and $\dim(\mathbf{B}_{2i+1}) = k$. Still, 1-DSP=SVP

Generalized basis reduction algorithm

- Partition basis vectors into blocks $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_m]$

Block Basis Reduction Algorithm

Locally modify each pair of blocks $[\mathbf{B}_i, \mathbf{B}_{i+1}]$ so to minimize the product $\|\mathbf{b}_j^*\| \cdots \|\mathbf{b}_k^*\|$ corresponding to $\mathbf{B}_i = [\mathbf{b}_j, \ldots, \mathbf{b}_k]$

- Local modification is a $\dim(\mathbf{B}_i)$-DSP instance in dimension $\dim(\mathbf{B}_i) + \dim(\mathbf{B}_{i+1})$
- LLL is a special case where $\dim(\mathbf{B}_i) = 1$, and 1-DSP $=$ SVP
- Sliding Reduction [Gama, Nguyen 2008] corresponds to $\dim(\mathbf{B}_{2i}) = 1$ and $\dim(\mathbf{B}_{2i+1}) = k$. Still, 1-DSP=SVP
- [Gama,Howgrave-Graham,Koy,Nguyen 2006]: $\dim(B_i) = k$

DSP and LLL [GHKN'06]

- Assume $[\mathbf{b}_1, \ldots, \mathbf{b}_k]$ generate the densest k-dimensional sublattice of $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n]$

DSP and LLL [GHKN'06]

- Assume $[\mathbf{b}_1, \ldots, \mathbf{b}_k]$ generate the densest k-dimensional sublattice of $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n]$
- Applying LLL to $\mathbf{B}$ does not modify $\Lambda' = [\mathbf{b}_1, \ldots, \mathbf{b}_k]\mathbb{Z}^k$

DSP and LLL [GHKN'06]

- Assume $[\mathbf{b}_1, \ldots, \mathbf{b}_k]$ generate the densest k-dimensional sublattice of $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n]$
- Applying LLL to $\mathbf{B}$ does not modify $\Lambda' = [\mathbf{b}_1, \ldots, \mathbf{b}_k]\mathbb{Z}^k$
- Algorithm for k-DSP: enumerate all LLL reduced bases for the input lattice, and select the smallest $\det([\mathbf{b}_1, \ldots, \mathbf{b}_k])$

DSP and LLL [GHKN'06]

- Assume $[\mathbf{b}_1, \ldots, \mathbf{b}_k]$ generate the densest k-dimensional sublattice of $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n]$
- Applying LLL to $\mathbf{B}$ does not modify $\Lambda' = [\mathbf{b}_1, \ldots, \mathbf{b}_k]\mathbb{Z}^k$
- Algorithm for k-DSP: enumerate all LLL reduced bases for the input lattice, and select the smallest $\det([\mathbf{b}_1, \ldots, \mathbf{b}_k])$
- A lattice can have as many as $2^{O(n^3)}$ reduced bases!

Better solutions for DSP?

- Recursive approach to find solution M to k-DSP
  - Minimize $\det(M) = \|\mathbf{b}_1^*\| \cdots \|\mathbf{b}_k^*\|$ where $M = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$
  - First find some $\mathbf{b}_1 \in M$, somehow.
  - Then minimize $\|\mathbf{b}_2^*\| \cdots \|\mathbf{b}_k^*\|$, i.e. solve $(k-1)$-DSP in projection of Λ orthogonal to $\mathbf{b}_1$.

Better solutions for DSP?

- Recursive approach to find solution M to k-DSP
  - Minimize $\det(M) = \|\mathbf{b}_1^*\| \cdots \|\mathbf{b}_k^*\|$ where $M = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$
  - First find some $\mathbf{b}_1 \in M$, somehow.
  - Then minimize $\|\mathbf{b}_2^*\| \cdots \|\mathbf{b}_k^*\|$, i.e. solve $(k-1)$-DSP in projection of Λ orthogonal to $\mathbf{b}_1$.
- How can we find $\mathbf{b}_1 \in M$?
  - Greedily set $\mathbf{b}_1$ to shortest lattice vector

Better solutions for DSP?

- Recursive approach to find solution M to k-DSP
  - Minimize $\det(M) = \|\mathbf{b}_1^*\| \cdots \|\mathbf{b}_k^*\|$ where $M = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$
  - First find some $\mathbf{b}_1 \in M$, somehow.
  - Then minimize $\|\mathbf{b}_2^*\| \cdots \|\mathbf{b}_k^*\|$, i.e. solve $(k-1)$-DSP in projection of Λ orthogonal to $\mathbf{b}_1$.
- How can we find $\mathbf{b}_1 \in M$?
  - Greedily set $\mathbf{b}_1$ to shortest lattice vector
  - Does not work! E.g., k-DSP for $k = 2, n = 3$:

SVP	$\mathbf{b}_1$	$=$	(a, a, a)	$\sqrt{1/2} < a < \sqrt{2/3}$
2-DSP	$\mathbf{b}_2$	$=$	$(1, -1, 0)$	
	$\mathbf{b}_3$	$=$	$(0, -1, 1)$	

Main Lemma and Algorithm

Lemma

For any k-DSP solution $M \subset \Lambda$, either M contains (every) shortest lattice vector $\mathbf{v} \in \Lambda$, or M contains k linearly independent vectors of length $\leq k\|\mathbf{v}\|$

Main Lemma and Algorithm

Lemma

For any k-DSP solution $M \subset \Lambda$, either M contains (every) shortest lattice vector $\mathbf{v} \in \Lambda$, or M contains k linearly independent vectors of length $\leq k\|\mathbf{v}\|$

Algorithm (by cases):

- Find shortest lattice vector $\mathbf{v}$, and recurtively solve $(k-1)$-DSP in $\Lambda \perp \mathbf{v}$

Main Lemma and Algorithm

Lemma

For any k-DSP solution $M \subset \Lambda$, either M contains (every) shortest lattice vector $\mathbf{v} \in \Lambda$, or M contains k linearly independent vectors of length $\leq k\|\mathbf{v}\|$

Algorithm (by cases):

- Find shortest lattice vector $\mathbf{v}$, and recurtively solve $(k-1)$-DSP in $\Lambda \perp \mathbf{v}$
- List all (L) vectors of length $\leq k\|\mathbf{v}\|$, and consider all subsets of L of size k

Main Lemma and Algorithm

Lemma

For any k-DSP solution $M \subset \Lambda$, either M contains (every) shortest lattice vector $\mathbf{v} \in \Lambda$, or M contains k linearly independent vectors of length $\leq k\|\mathbf{v}\|$

Algorithm (by cases):

- Find shortest lattice vector $\mathbf{v}$, and recurtively solve $(k-1)$-DSP in $\Lambda \perp \mathbf{v}$
- List all (L) vectors of length $\leq k\|\mathbf{v}\|$, and consider all subsets of L of size k
- Select the best solution found

Main Lemma and Algorithm

Lemma

For any k-DSP solution $M \subset \Lambda$, either M contains (every) shortest lattice vector $\mathbf{v} \in \Lambda$, or M contains k linearly independent vectors of length $\leq k\|\mathbf{v}\|$

Algorithm (by cases):

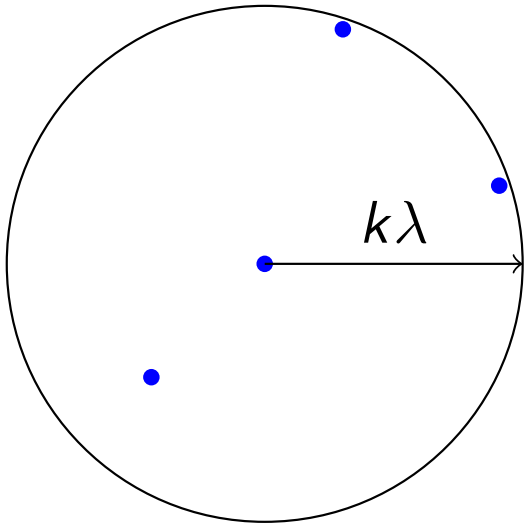
- Find shortest lattice vector $\mathbf{v}$, and recurtively solve $(k-1)$-DSP in $\Lambda \perp \mathbf{v}$
- List all (L) vectors of length $\leq k\|\mathbf{v}\|$, and consider all subsets of L of size k
- Select the best solution found

Running time is $T \approx 2^{O(n)} + \binom{L}{k}$

Bounding the list size

- Put a sphere of radius $\lambda/2$ around every point in L
- Spheres are disjoint
- All spheres belong to larger sphere of radius $(k + \frac{1}{2})\lambda$
- Volume bound:

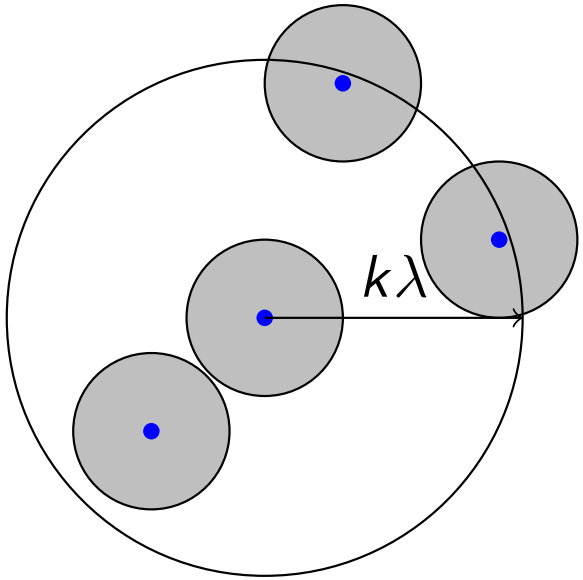
$$|L| \cdot \left(\frac{1}{2}\right)^n \leq \left(k + \frac{1}{2}\right)^n$$



Bounding the list size

- Put a sphere of radius $\lambda/2$ around every point in L
- Spheres are disjoint
- All spheres belong to larger sphere of radius $(k + \frac{1}{2})\lambda$
- Volume bound:

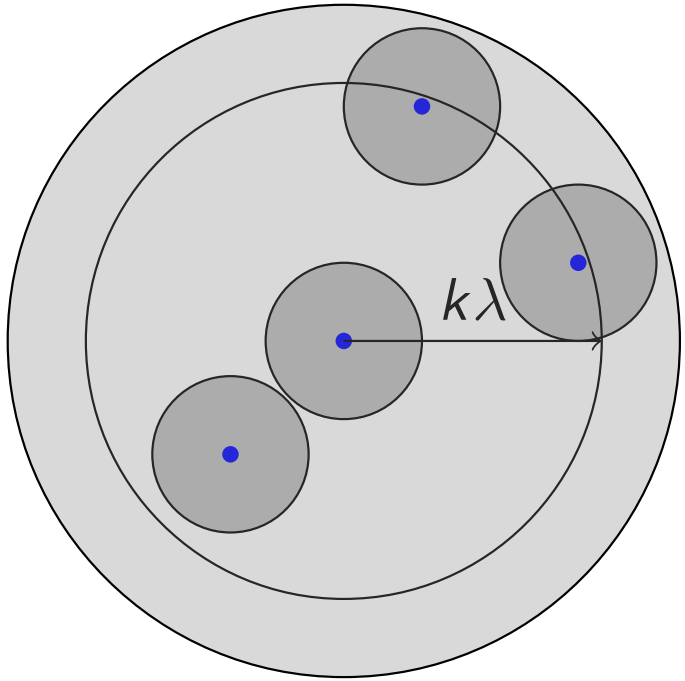
$$|L| \cdot \left(\frac{1}{2}\right)^n \leq \left(k + \frac{1}{2}\right)^n$$



Bounding the list size

- Put a sphere of radius $\lambda/2$ around every point in L
- Spheres are disjoint
- All spheres belong to larger sphere of radius $(k + \frac{1}{2})\lambda$
- Volume bound:

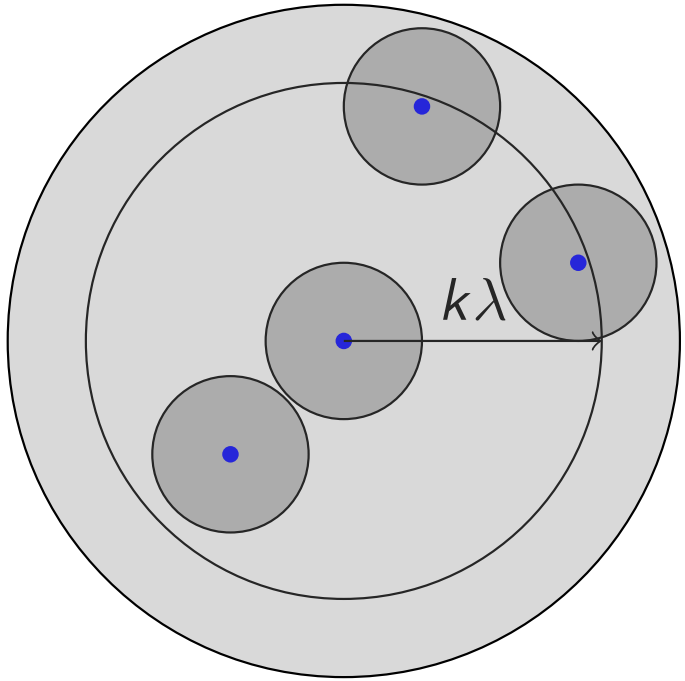
$$|L| \cdot \left(\frac{1}{2}\right)^n \leq \left(k + \frac{1}{2}\right)^n$$



Bounding the list size

- Put a sphere of radius $\lambda/2$ around every point in L
- Spheres are disjoint
- All spheres belong to larger sphere of radius $(k + \frac{1}{2})\lambda$
- Volume bound:

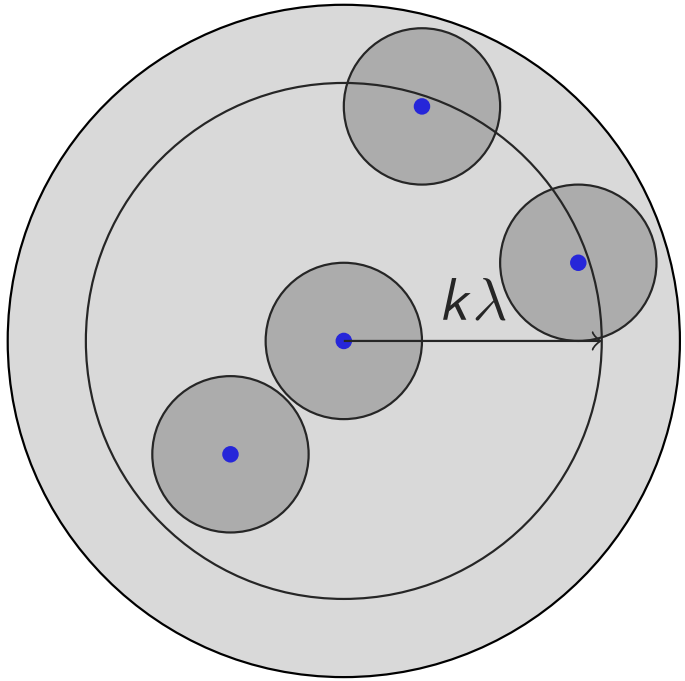
$$|L| \cdot \left(\frac{1}{2}\right)^n \leq \left(k + \frac{1}{2}\right)^n$$



Bounding the list size

- Put a sphere of radius $\lambda/2$ around every point in L
- Spheres are disjoint
- All spheres belong to larger sphere of radius $(k + \frac{1}{2})\lambda$
- Volume bound:

$$|L| \cdot \left(\frac{1}{2}\right)^n \leq \left(k + \frac{1}{2}\right)^n$$



Theorem (Main)

k-DSP can be solved in time

$$T \approx |L|^k = (2k+1)^{kn} = k^{O(kn)}$$

Conclusion

- First shot at algorithmic solution of Densest Sublattice Problem
- Problem/Solution extends to arbitrary norms, achieving similar asymptotic running times

Conclusion

- First shot at algorithmic solution of Densest Sublattice Problem
- Problem/Solution extends to arbitrary norms, achieving similar asymptotic running times
- Potential applications to lattice basis reduction

Conclusion

- First shot at algorithmic solution of Densest Sublattice Problem
- Problem/Solution extends to arbitrary norms, achieving similar asymptotic running times
- Potential applications to lattice basis reduction
- Plenty of open problems!
  - Basic algorithm is far from practical. Develop fast heuristic implementations
  - Evaluate performance in practice when run on random lattices within block basis reduction
  - Reduce time dependency on k (currently $k^{O(k \cdot n)}$: can k-DSP be solved in time $2^{O(n)}$
  - Explore applications

Conclusion

- First shot at algorithmic solution of Densest Sublattice Problem
- Problem/Solution extends to arbitrary norms, achieving similar asymptotic running times
- Potential applications to lattice basis reduction
- Plenty of open problems!
  - Basic algorithm is far from practical. Develop fast heuristic implementations
  - Evaluate performance in practice when run on random lattices within block basis reduction
  - Reduce time dependency on k (currently $k^{O(k \cdot n)}$: can k-DSP be solved in time $2^{O(n)}$
  - Explore applications

Thanks! Questions?

Optimizations

k-DSP solution M satisfies other useful properties

Lemma

Any k-DSP solution M always contains a vector of length $\leq \gamma_k \lambda$

Optimizations

k-DSP solution M satisfies other useful properties

Lemma

Any k-DSP solution M always contains a vector of length $\leq \gamma_k \lambda$

- $\gamma_k = \Theta(n)$, but $\gamma_k \ll k$ (e.g., $\gamma_k \leq 2$ for all $k = 1, \ldots, 8$)

Optimizations

k-DSP solution M satisfies other useful properties

Lemma

Any k-DSP solution M always contains a vector of length $\leq \gamma_k \lambda$

- $\gamma_k = \Theta(n)$, but $\gamma_k \ll k$ (e.g., $\gamma_k \leq 2$ for all $k = 1, \ldots, 8$)
- Notice: M may not contain k linearly independent vectors of length $\leq \gamma_k \|\mathbf{v}\|$

Optimizations

k-DSP solution M satisfies other useful properties

Lemma

Any k-DSP solution M always contains a vector of length $\leq \gamma_k \lambda$

- $\gamma_k = \Theta(n)$, but $\gamma_k \ll k$ (e.g., $\gamma_k \leq 2$ for all $k = 1, \ldots, 8$)
- Notice: M may not contain k linearly independent vectors of length $\leq \gamma_k \|\mathbf{v}\|$
- Still, lemma is useful to restrict search for $\mathbf{b}_1$ to subset of L

Optimizations

k-DSP solution M satisfies other useful properties

Lemma

Any k-DSP solution M always contains a vector of length $\leq \gamma_k \lambda$

- $\gamma_k = \Theta(n)$, but $\gamma_k \ll k$ (e.g., $\gamma_k \leq 2$ for all $k = 1, \ldots, 8$)
- Notice: M may not contain k linearly independent vectors of length $\leq \gamma_k \|\mathbf{v}\|$
- Still, lemma is useful to restrict search for $\mathbf{b}_1$ to subset of L
- One can project and recurse for $\mathbf{b}_2, \ldots, \mathbf{b}_k$

Optimizations

k-DSP solution M satisfies other useful properties

Lemma

Any k-DSP solution M always contains a vector of length $\leq \gamma_k \lambda$

- $\gamma_k = \Theta(n)$, but $\gamma_k \ll k$ (e.g., $\gamma_k \leq 2$ for all $k = 1, \ldots, 8$)
- Notice: M may not contain k linearly independent vectors of length $\leq \gamma_k \|\mathbf{v}\|$
- Still, lemma is useful to restrict search for $\mathbf{b}_1$ to subset of L
- One can project and recurse for $\mathbf{b}_2, \ldots, \mathbf{b}_k$
- Asymptotic running time is still $k^{O(kn)}$

Noneuclidean norms

- ℓ_1 and ℓ_∞ norms naturally arise in many combinatorial applications, e.g., integer programming

Noneuclidean norms

- ℓ_1 and ℓ_∞ norms naturally arise in many combinatorial applications, e.g., integer programming
- Non ℓ_p norm naturally arise when "projecting" ℓ_1 or ℓ_∞ norms to lower dimensional subspaces

Noneuclidean norms

- ℓ_1 and ℓ_∞ norms naturally arise in many combinatorial applications, e.g., integer programming
- Non ℓ_p norm naturally arise when "projecting" ℓ_1 or ℓ_∞ norms to lower dimensional subspaces
- Our algorithm can be adapted to arbitrary norms, achieving similar running time

Noneuclidean norms

- ℓ_1 and ℓ_∞ norms naturally arise in many combinatorial applications, e.g., integer programming
- Non ℓ_p norm naturally arise when "projecting" ℓ_1 or ℓ_∞ norms to lower dimensional subspaces
- Our algorithm can be adapted to arbitrary norms, achieving similar running time
- Generalization is conceptually simple, but technically involved, using methods from high dimensional convex geometry

Noneuclidean norms

- ℓ_1 and ℓ_∞ norms naturally arise in many combinatorial applications, e.g., integer programming
- Non ℓ_p norm naturally arise when "projecting" ℓ_1 or ℓ_∞ norms to lower dimensional subspaces
- Our algorithm can be adapted to arbitrary norms, achieving similar running time
- Generalization is conceptually simple, but technically involved, using methods from high dimensional convex geometry
- Even the definition of DSP is not completely obvious

Performance in practice

- Strightforward implementation of DSP algorithm unlikely to be practical
- Work in progress: implementation of optimized versions of algorithm for small values of k

Performance in practice

- Strightforward implementation of DSP algorithm unlikely to be practical
- Work in progress: implementation of optimized versions of algorithm for small values of k
- [Schnorr,Euchner 1996] . . . [Gama, Nguyen, Regev 2010]: heuristic pruning methods for SVP/CVP ($n \approx 100$)
- [Nguyen,Stehle 2006] . . . [Chen,Nguyen 2012] LLL-like algoriths much better in practice than γ_2^n worst-case bound

Performance in practice

- Strightforward implementation of DSP algorithm unlikely to be practical
- Work in progress: implementation of optimized versions of algorithm for small values of k
- [Schnorr,Euchner 1996] ... [Gama, Nguyen, Regev 2010]: heuristic pruning methods for SVP/CVP ($n \approx 100$)
- [Nguyen,Stehle 2006] ... [Chen,Nguyen 2012] LLL-like algoriths much better in practice than γ_2^n worst-case bound

Open Problem

Develop effective pruning strategies/heuristics to speed up DSP computation, and evaluate performance in practice on random lattices.

Single exponential time?

- Our algorithm solves k-DSP in $TIME(2^{O(n)})$ when $k = O(1)$

Open Problem

Can DSP be solved in time $2^{O(n)}$ for arbitrary k?

Single exponential time?

- Our algorithm solves k-DSP in $TIME(2^{O(n)})$ when $k = O(1)$

Open Problem

Can DSP be solved in time $2^{O(n)}$ for arbitrary k?

- It may be tempting to assume that k-DSP must be exponential in k because solution consists of k vectors

Single exponential time?

- Our algorithm solves k-DSP in $TIME(2^{O(n)})$ when $k = O(1)$

Open Problem

Can DSP be solved in time $2^{O(n)}$ for arbitrary k?

- It may be tempting to assume that k-DSP must be exponential in k because solution consists of k vectors
- But there is not reason for this to be so:
  - Shortest Independent Vectors Problem (k-SIVP): Find k linearly independent vectors such that $\max\{\|\mathbf{v}_1\|, \ldots, \|\mathbf{v}_k\|\}$
  - k-DSP similar to k-SIVP, with "det" instead of "max"

Single exponential time?

- Our algorithm solves k-DSP in $TIME(2^{O(n)})$ when $k = O(1)$

Open Problem

Can DSP be solved in time $2^{O(n)}$ for arbitrary k?

- It may be tempting to assume that k-DSP must be exponential in k because solution consists of k vectors
- But there is not reason for this to be so:
  - Shortest Independent Vectors Problem (k-SIVP): Find k linearly independent vectors such that $\max\{\|\mathbf{v}_1\|, \ldots, \|\mathbf{v}_k\|\}$
  - k-DSP similar to k-SIVP, with "det" instead of "max"
  - SIVP reduces to CVP [Micciancio 2008], and therefore it can be solved in time $2^{O(n)}$ [Micciancio, Voulgaris 2010].

Single exponential time?

- Our algorithm solves k-DSP in $TIME(2^{O(n)})$ when $k = O(1)$

Open Problem

Can DSP be solved in time $2^{O(n)}$ for arbitrary k?

- It may be tempting to assume that k-DSP must be exponential in k because solution consists of k vectors
- But there is not reason for this to be so:
  - Shortest Independent Vectors Problem (k-SIVP): Find k linearly independent vectors such that $\max\{\|\mathbf{v}_1\|, \ldots, \|\mathbf{v}_k\|\}$
  - k-DSP similar to k-SIVP, with "det" instead of "max"
  - SIVP reduces to CVP [Micciancio 2008], and therefore it can be solved in time $2^{O(n)}$ [Micciancio, Voulgaris 2010].
- Alternatively, offer evidence that k-DSP cannot be solved in $2^{O(n)}$, e.g., reduce higher dimensional CVP problem to DSP

Rankin constant (Generalizing Hermite $\gamma_n = \gamma_{n,1}$)

$$\gamma_{n,k} = \sup_{\Lambda} \inf_{\mathbf{B}} \left(\frac{\det(\mathbf{b}_1, \ldots, \mathbf{b}_k)^{1/k}}{\det(\Lambda)^{1/n}} \right)^{2k} \qquad \text{(Rankin 1955)}$$

1	1								
2	$\frac{2}{\sqrt{3}}$	1							
3	$\sqrt[3]{2}$	$\sqrt[3]{2}$	1						
4	$\sqrt{2}$	$3/2$	$\sqrt{2}$	1					
5	$2^{3/5}$			$2^{3/5}$	1				
6	$\sqrt[6]{64/3}$	$3^{2/3}$		$3^{2/3}$	$\sqrt[6]{64/3}$	1			
7	$2^{6/7}$					$2^{6/7}$	1		
8	2	3	4	4	4	3	2	1	
24	4							4	1
n/k	1	2	3	4	5	6	7	23	24

DSP algorithm can be used to obtain lower bounds on $\gamma_{n,k}$

Rankin constant (Generalizing Hermite $\gamma_n = \gamma_{n,1}$)

$$\gamma_{n,k} = \sup_{\Lambda} \inf_{\mathbf{B}} \left(\frac{\det(\mathbf{b}_1, \ldots, \mathbf{b}_k)^{1/k}}{\det(\Lambda)^{1/n}} \right)^{2k} \qquad \text{(Rankin 1955)}$$

1	1								
2	$\frac{2}{\sqrt{3}}$	1							
3	$\sqrt[3]{2}$	$\sqrt[3]{2}$	1						
4	$\sqrt{2}$	3/2	$\sqrt{2}$	1					
5	$2^{3/5}$			$2^{3/5}$	1				
6	$\sqrt[6]{64/3}$	$3^{2/3}$		$3^{2/3}$	$\sqrt[6]{64/3}$	1			
7	$2^{6/7}$					$2^{6/7}$	1		
8	2	3	4	4	4	3	2	1	
24	4							4	1
n/k	1	2	3	4	5	6	7	23	24

DSP algorithm can be used to obtain lower bounds on $\gamma_{n,k}$

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$
  - SVP approximation factor $\approx \gamma_{h,k}^{n/k(h-k)}$

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$
  - SVP approximation factor $\approx \gamma_{h,k}^{n/k(h-k)}$
  - For $h = 2, k = 1$, we get LLL factor γ_2^n

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$
  - SVP approximation factor $\approx \gamma_{h,k}^{n/k(h-k)}$
  - For $h = 2, k = 1$, we get LLL factor γ_2^n
- Tradeoff between quality and running time

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$
  - SVP approximation factor $\approx \gamma_{h,k}^{n/k(h-k)}$
  - For $h = 2, k = 1$, we get LLL factor γ_2^n
- Tradeoff between quality and running time
- Open Problem: Determine optimal values of k, h in theory and/or practice

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$
  - SVP approximation factor $\approx \gamma_{h,k}^{n/k(h-k)}$
  - For $h = 2, k = 1$, we get LLL factor γ_2^n
- Tradeoff between quality and running time
- Open Problem: Determine optimal values of k, h in theory and/or practice
- [Gama,Nguyen 2008] suggests that $k = 1$ offers better tradeoff than $k = h/2$

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$
  - SVP approximation factor $\approx \gamma_{h,k}^{n/k(h-k)}$
  - For $h = 2, k = 1$, we get LLL factor γ_2^n
- Tradeoff between quality and running time
- Open Problem: Determine optimal values of k, h in theory and/or practice
- [Gama,Nguyen 2008] suggests that $k = 1$ offers better tradeoff than $k = h/2$
- But $k = 2$ produces better bases than $k = 1$!

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$
  - SVP approximation factor $\approx \gamma_{h,k}^{n/k(h-k)}$
  - For $h = 2, k = 1$, we get LLL factor γ_2^n
- Tradeoff between quality and running time
- Open Problem: Determine optimal values of k, h in theory and/or practice
- [Gama,Nguyen 2008] suggests that $k = 1$ offers better tradeoff than $k = h/2$
- But $k = 2$ produces better bases than $k = 1$!
- CVP with preprocessing: once a reduced basis is found, it can be used multiple times with different targets

Performance of DSP Basis Reduction

- Using k-DSP algorithm on h-dimensional blocks
  - Running time $k^{O(kh)}$
  - SVP approximation factor $\approx \gamma_{h,k}^{n/k(h-k)}$
  - For $h = 2, k = 1$, we get LLL factor γ_2^n
- Tradeoff between quality and running time
- Open Problem: Determine optimal values of k, h in theory and/or practice
- [Gama,Nguyen 2008] suggests that $k = 1$ offers better tradeoff than $k = h/2$
- But $k = 2$ produces better bases than $k = 1$!
- CVP with preprocessing: once a reduced basis is found, it can be used multiple times with different targets
- Using "dynamic" h, k may produce even better bases