

Quantum Cryptanalysis of Post-Quantum Cryptography

Thijs Laarhoven, Michele Mosca, Joop van de Pol

t.m.m.laarhoven@tue.nl
<http://www.thijs.com/>

PQCrypto 2013, Limoges, France
(June 6, 2013)

Solving the Shortest Vector Problem in Lattices Faster Using Quantum Search

Thijs Laarhoven, Michele Mosca, Joop van de Pol

t.m.m.laarhoven@tue.nl
<http://www.thijs.com/>

PQCrypto 2013, Limoges, France
(June 6, 2013)

Outline

Introduction

Lattices

Quantum Search

Applications

SVP Algorithms

Sieving

Saturation

Enumeration

Overview

Conclusion

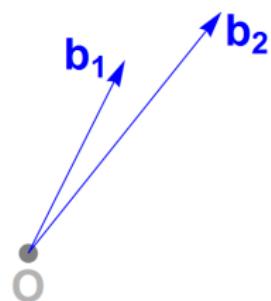
Lattices

What is a lattice?



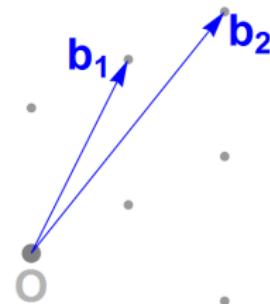
Lattices

What is a lattice?



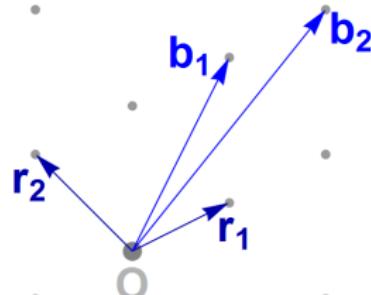
Lattices

What is a lattice?



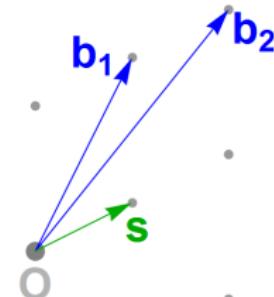
Lattices

Lattice Basis Reduction



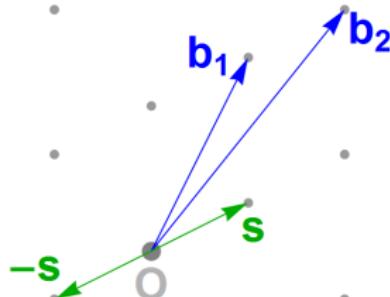
Lattices

Shortest Vector Problem (SVP)



Lattices

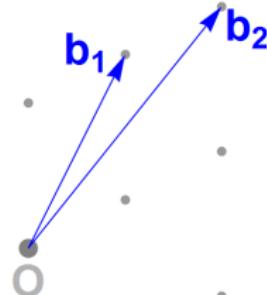
Shortest Vector Problem (SVP)



Lattices

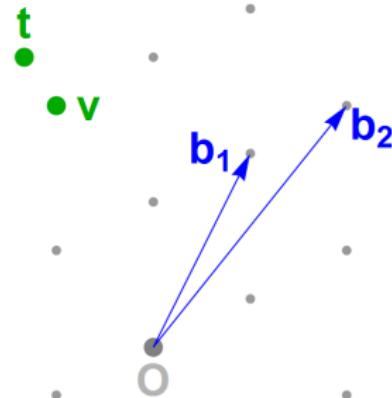
Closest Vector Problem (CVP)

t



Lattices

Closest Vector Problem (CVP)



Quantum Search

Classical form

Problem: Given a list L of size N , and a function $f : L \rightarrow \{0, 1\}$ such that there is exactly one $e \in L$ with $f(e) = 1$. Find e .

Quantum Search

Classical form

Problem: Given a list L of size N , and a function $f : L \rightarrow \{0, 1\}$ such that there is exactly one $e \in L$ with $f(e) = 1$. Find e .

- Classical search: $\Theta(N)$ time

Quantum Search

Classical form

Problem: Given a list L of size N , and a function $f : L \rightarrow \{0, 1\}$ such that there is exactly one $e \in L$ with $f(e) = 1$. Find e .

- Classical search: $\Theta(N)$ time
- Quantum search: $\Theta(\sqrt{N})$ time [Gro96]

Quantum Search

General form

Problem: Given a list L of size N , and a function $f : L \rightarrow \{0, 1\}$ such that there are $c = O(1)$ elements $e \in L$ with $f(e) = 1$. Find one such e .

Quantum Search

General form

Problem: Given a list L of size N , and a function $f : L \rightarrow \{0, 1\}$ such that there are $c = O(1)$ elements $e \in L$ with $f(e) = 1$. Find one such e .

- Classical search: $\Theta(N/c)$ time

Quantum Search

General form

Problem: Given a list L of size N , and a function $f : L \rightarrow \{0, 1\}$ such that there are $c = O(1)$ elements $e \in L$ with $f(e) = 1$. Find one such e .

- Classical search: $\Theta(N/c)$ time
- Quantum search: $\Theta(\sqrt{N/c})$ time [Gro96]

Applications

(Why do we care?)

- “Constructive cryptography”: Lattice-based cryptosystems
 - ▶ Based on hard lattice problems (SVP, CVP)
 - ▶ NTRU cryptosystem [HPS98]
 - ▶ Fully Homomorphic Encryption [Gen09]
 - ▶ Candidate for “Post-Quantum” cryptography

Applications

(Why do we care?)

- “Constructive cryptography”: Lattice-based cryptosystems
 - ▶ Based on hard lattice problems (SVP, CVP)
 - ▶ NTRU cryptosystem [HPS98]
 - ▶ Fully Homomorphic Encryption [Gen09]
 - ▶ Candidate for “Post-Quantum” cryptography
- “Destructive cryptography”: Cryptanalysis
 - ▶ Attack knapsack-based cryptosystems [Sha82, LO85]
 - ▶ Attack variants of RSA [Cop96]
 - ▶ Attack DSA and ECDSA [NS02, NS03]
 - ▶ Attack lattice-based cryptosystems [Ngu99, JJ00]

Applications

(Why do we care?)

- “Constructive cryptography”: Lattice-based cryptosystems
 - ▶ Based on hard lattice problems (SVP, CVP)
 - ▶ NTRU cryptosystem [HPS98]
 - ▶ Fully Homomorphic Encryption [Gen09]
 - ▶ Candidate for “Post-Quantum” cryptography
- “Destructive cryptography”: Cryptanalysis
 - ▶ Attack knapsack-based cryptosystems [Sha82, LO85]
 - ▶ Attack variants of RSA [Cop96]
 - ▶ Attack DSA and ECDSA [NS02, NS03]
 - ▶ Attack lattice-based cryptosystems [Ngu99, JJ00]

How (quantum-)hard are hard lattice problems such as SVP?

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

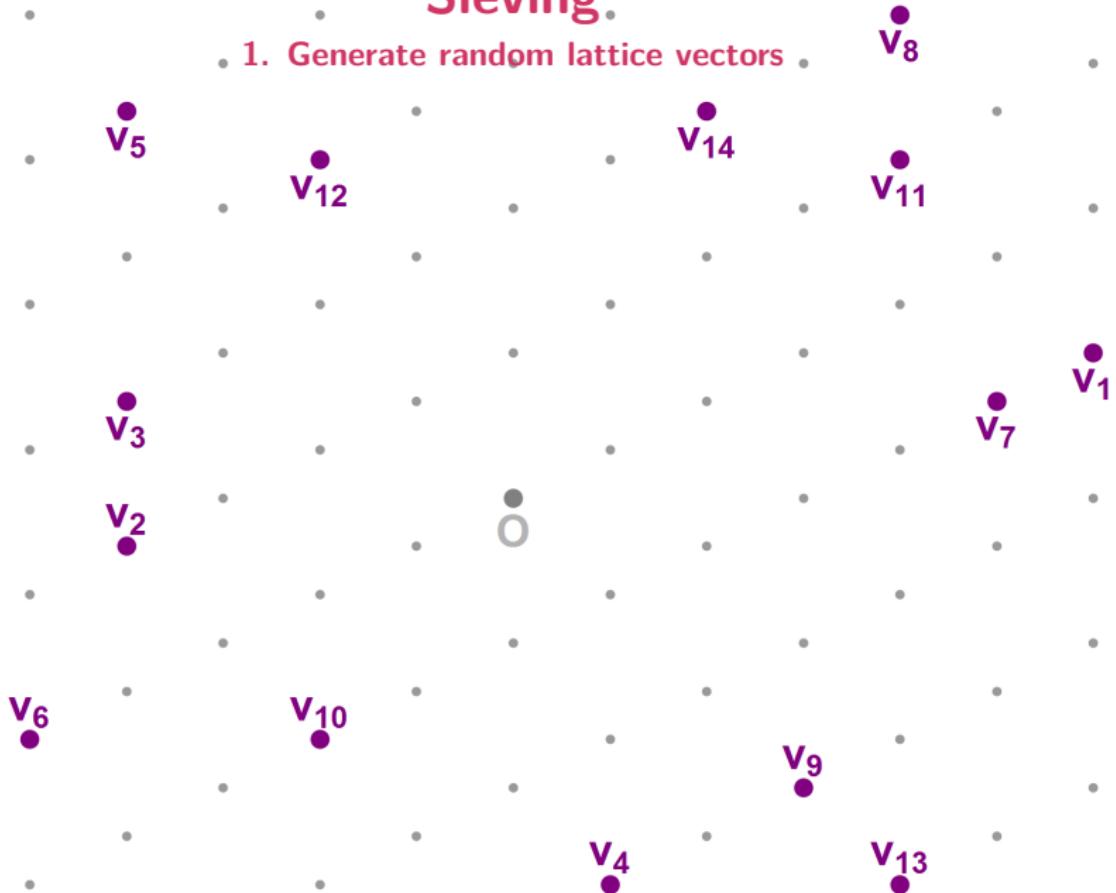
Sieving

1. Generate random lattice vectors



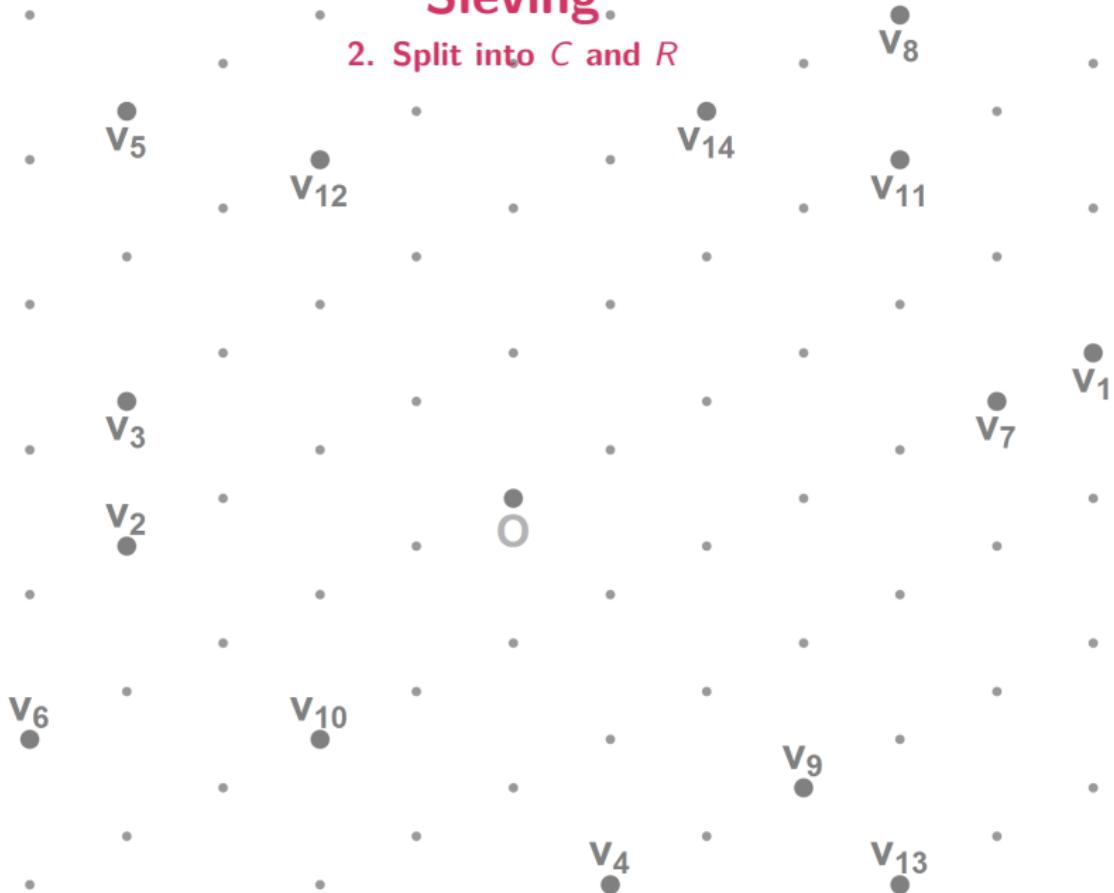
Sieving

1. Generate random lattice vectors



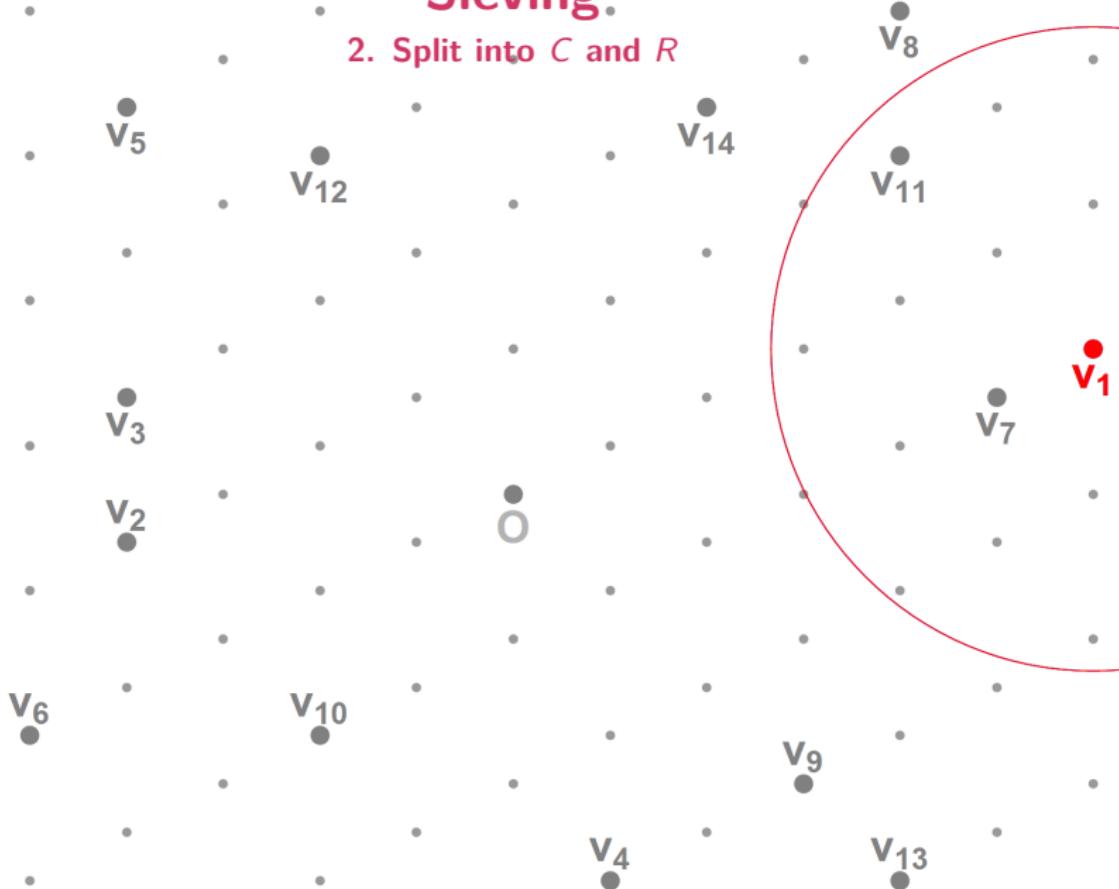
Sieving

2. Split into C and R



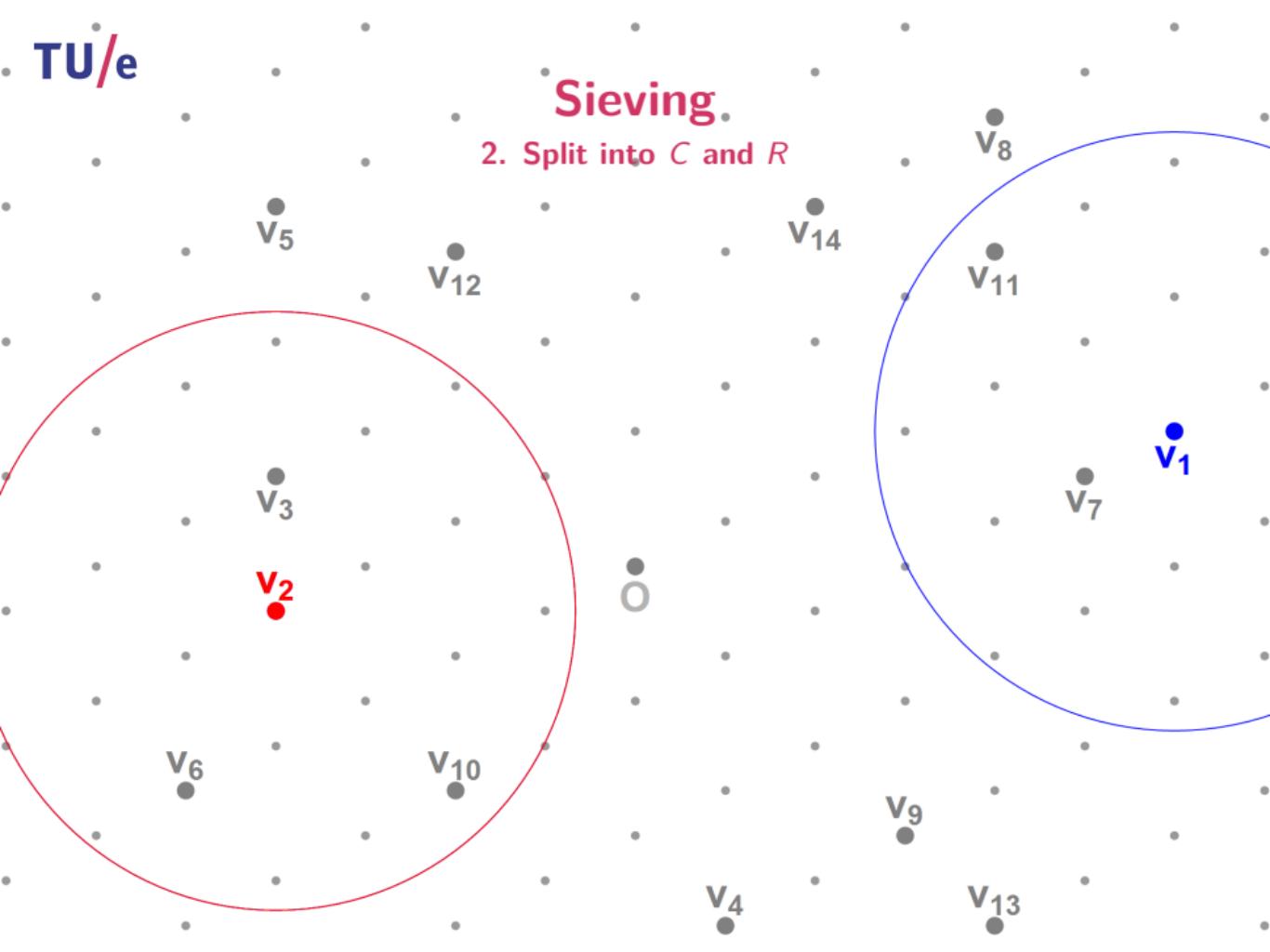
Sieving

2. Split into C and R

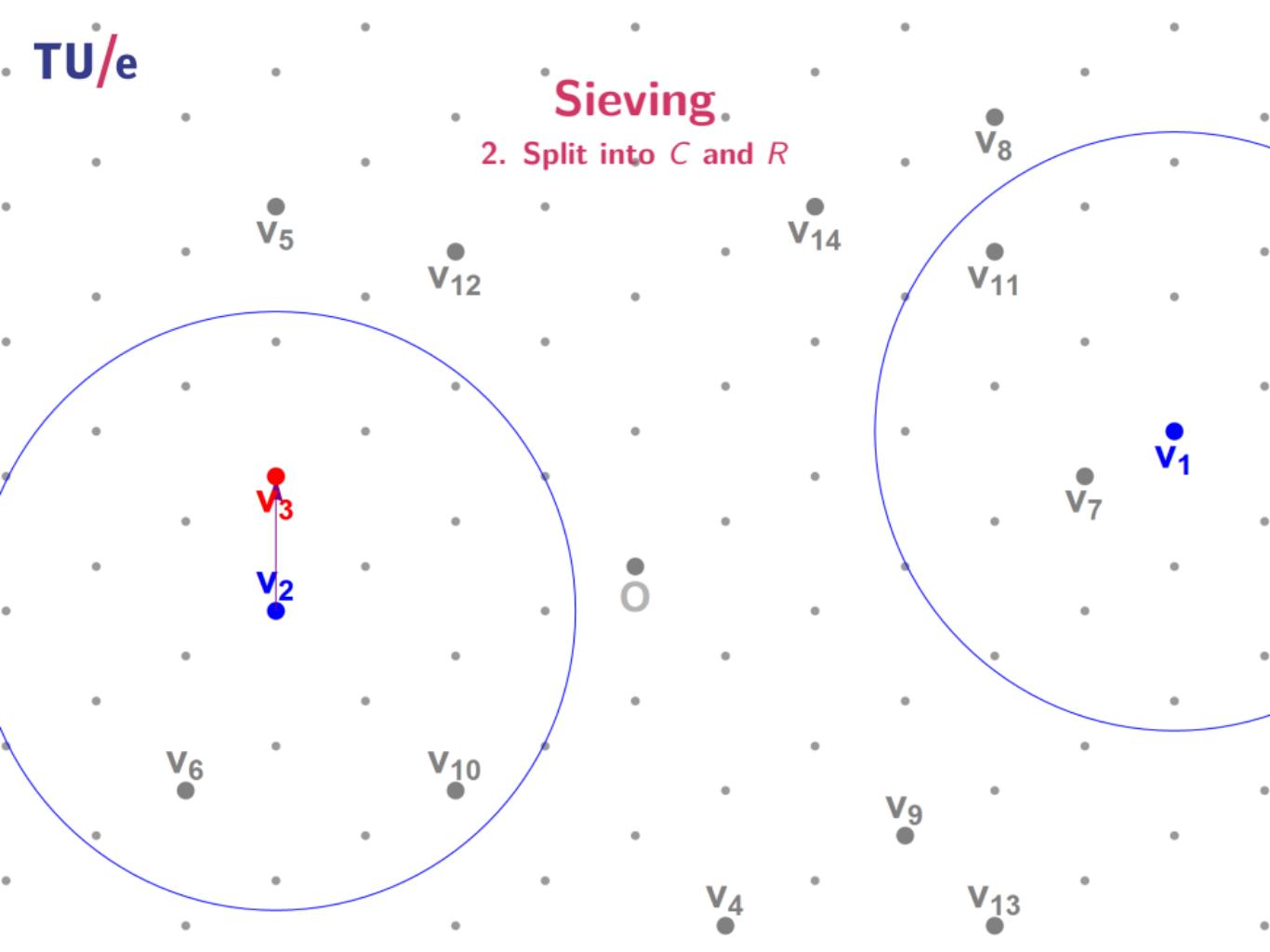


Sieving

2. Split into C and R

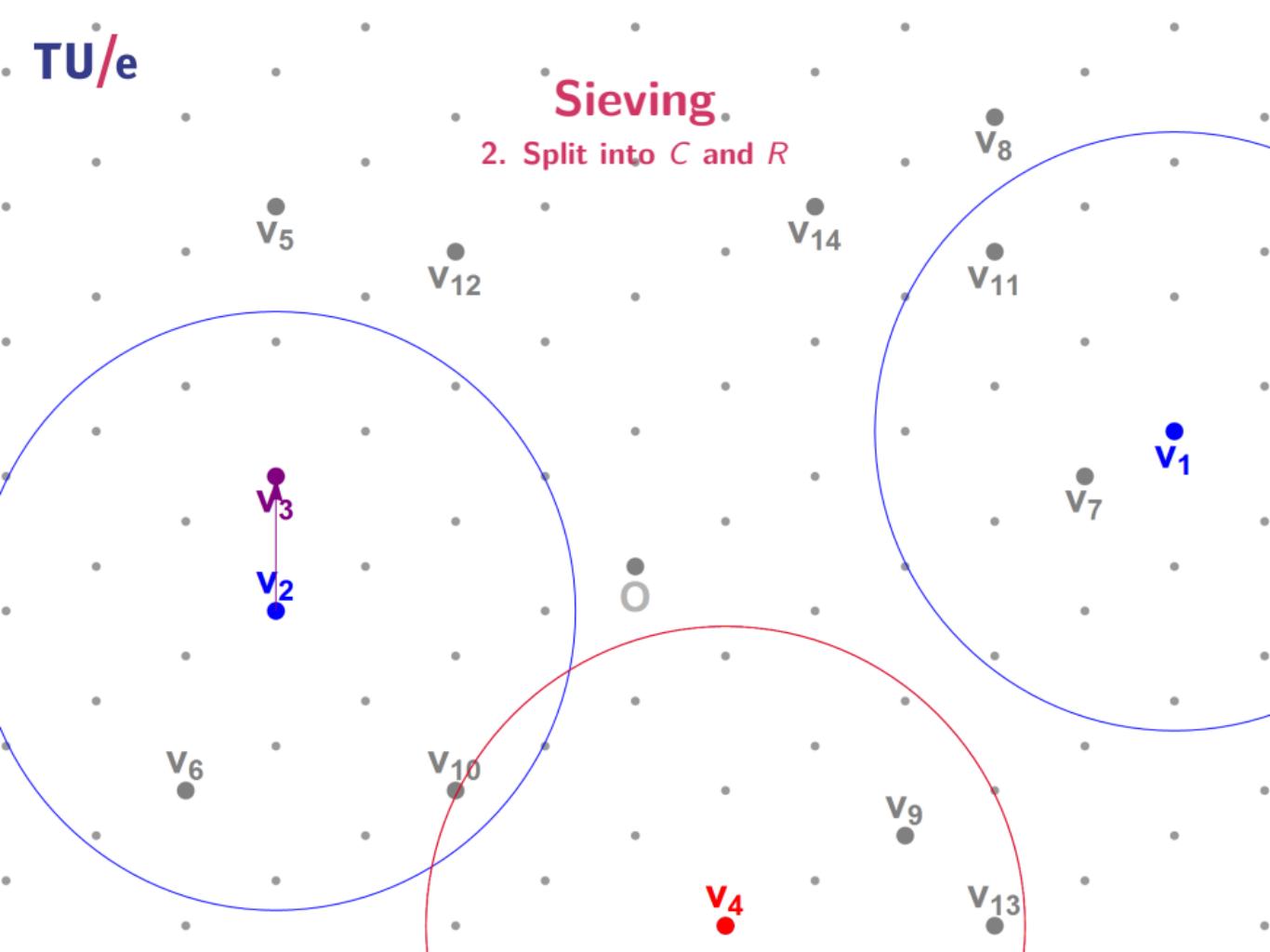


Sieving

2. Split into C and R 

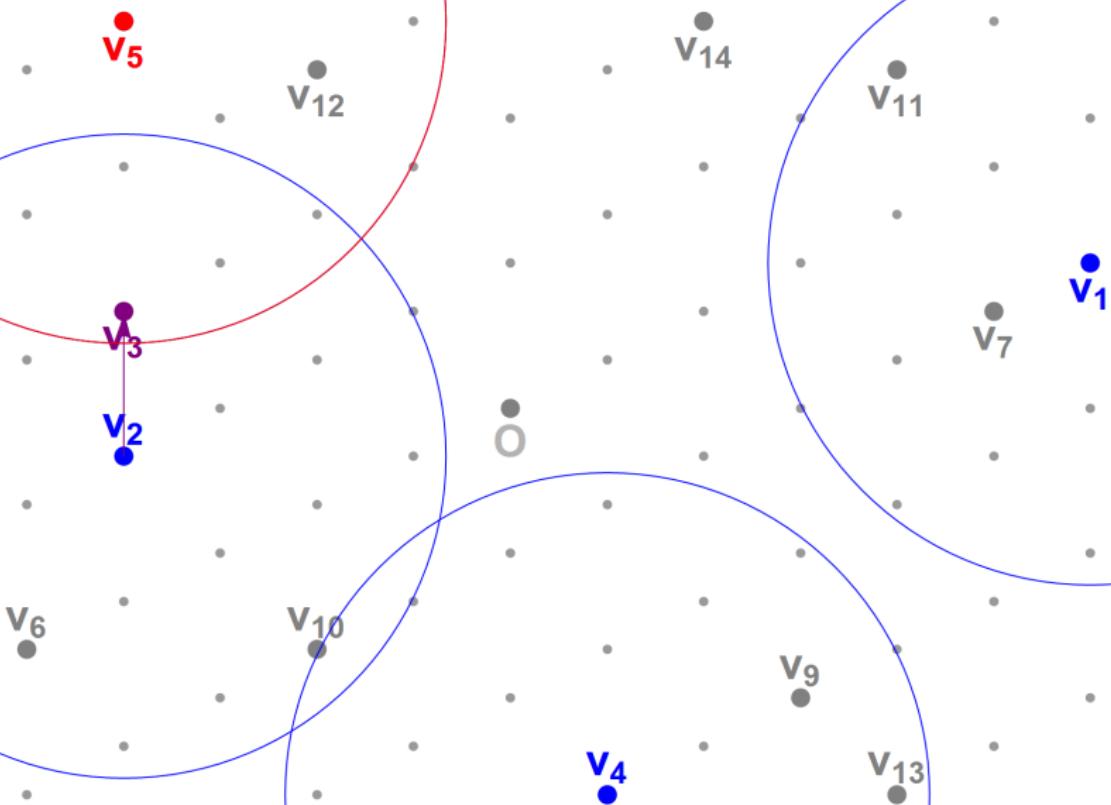
Sieving

2. Split into C and R



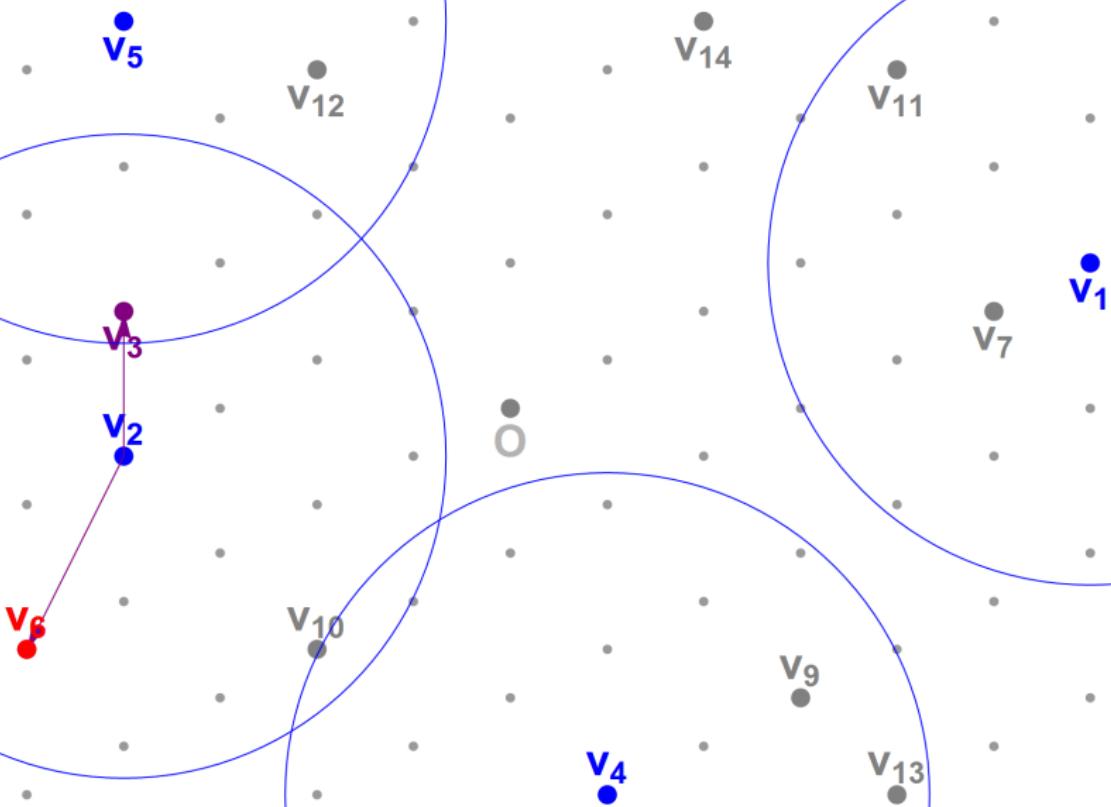
Sieving

2. Split into C and R



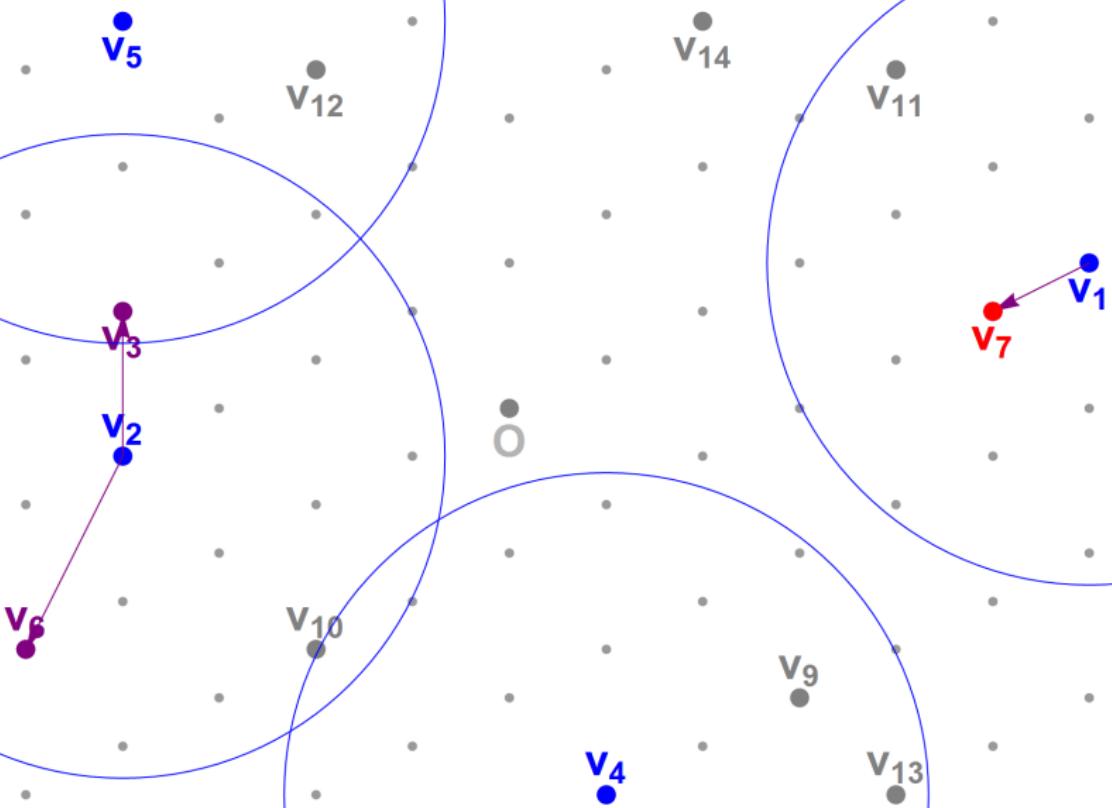
Sieving

2. Split into C and R



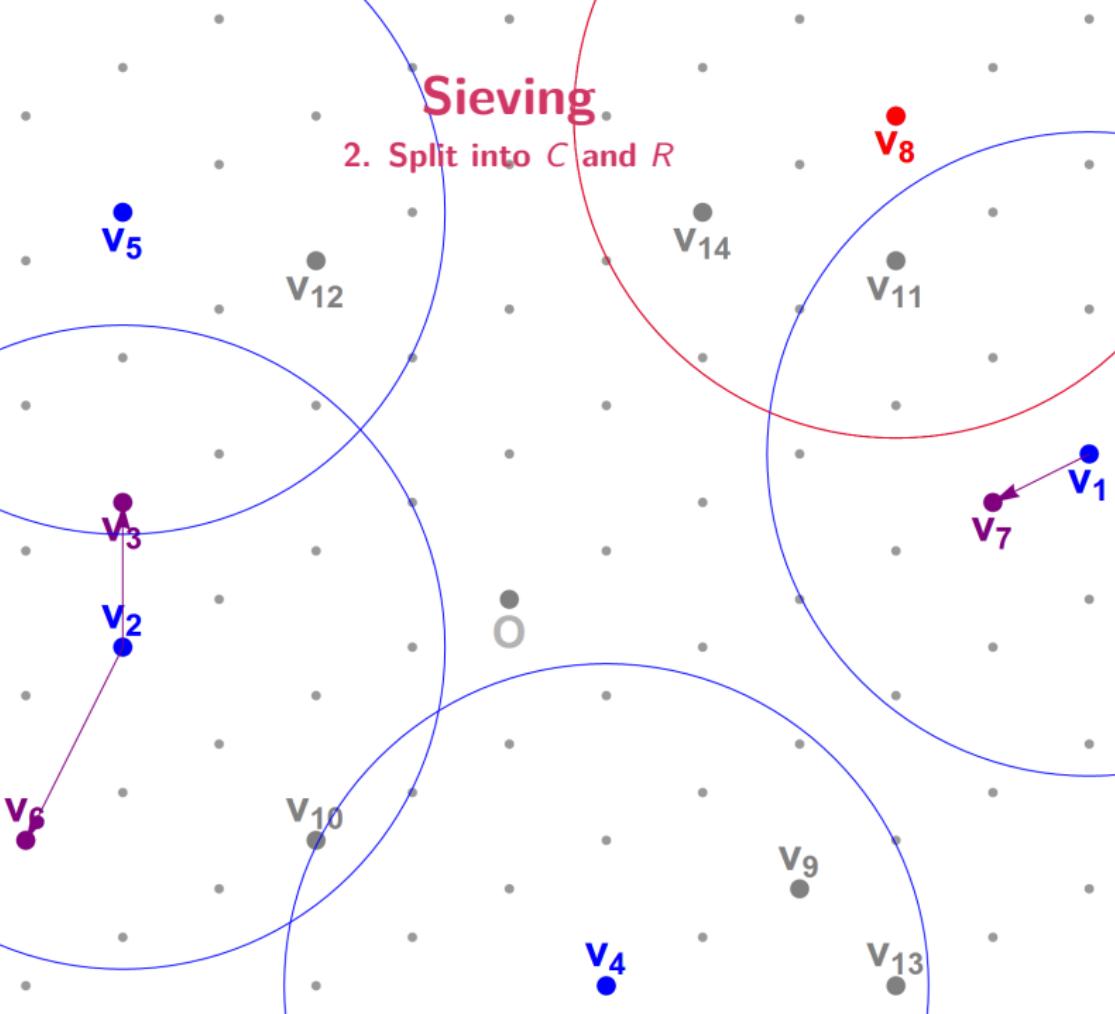
Sieving

2. Split into C and R



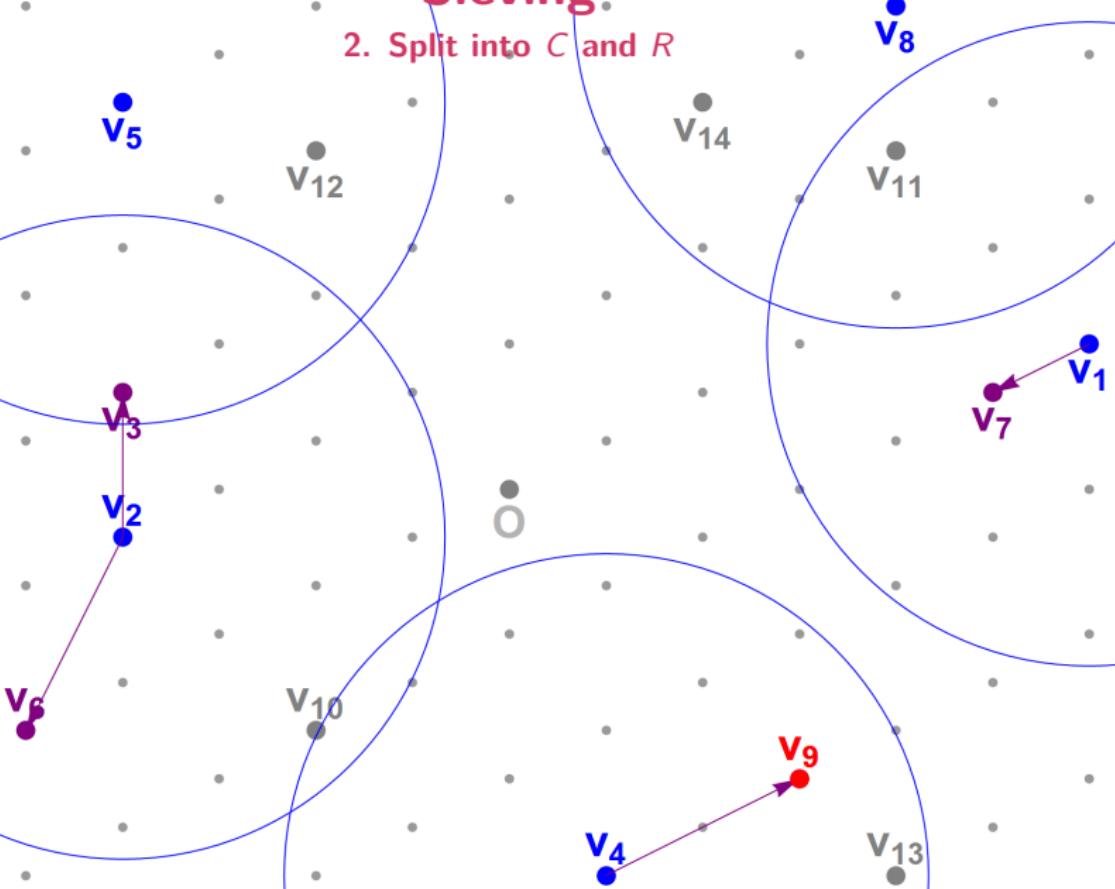
Sieving

2. Split into C and R



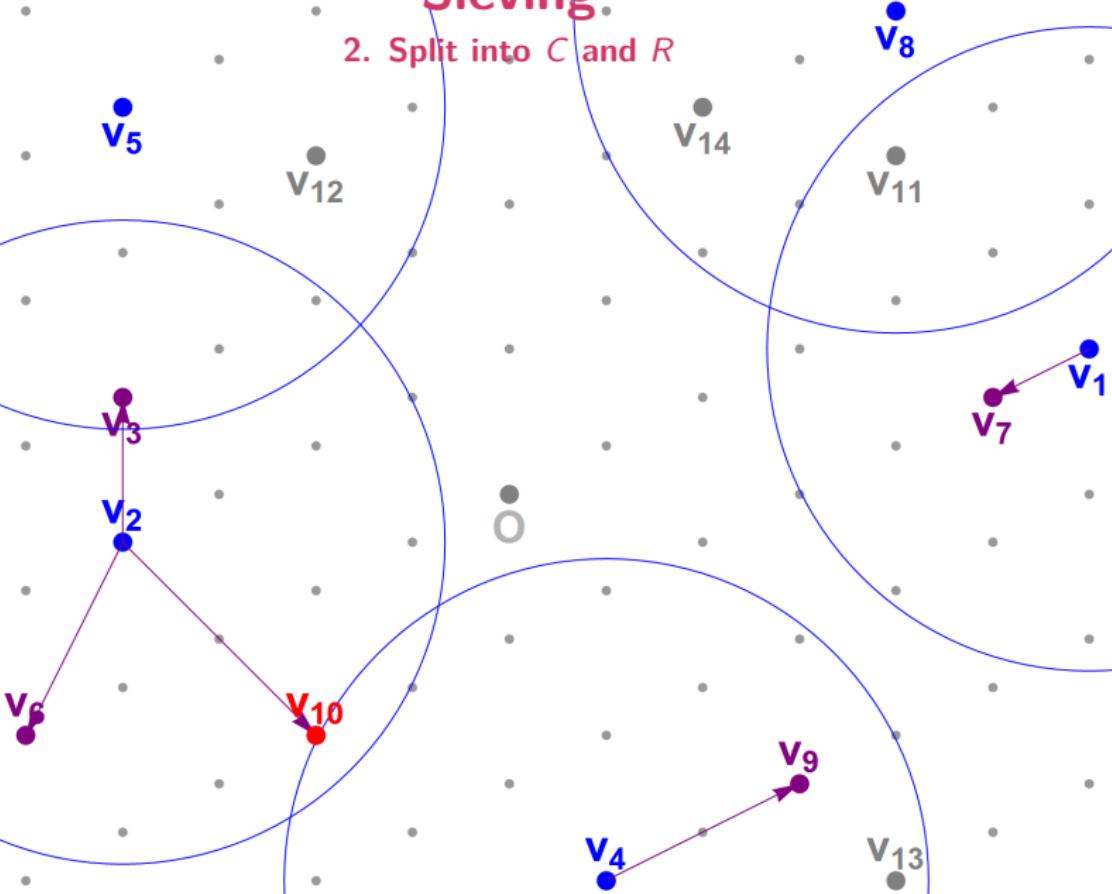
Sieving

2. Split into C and R



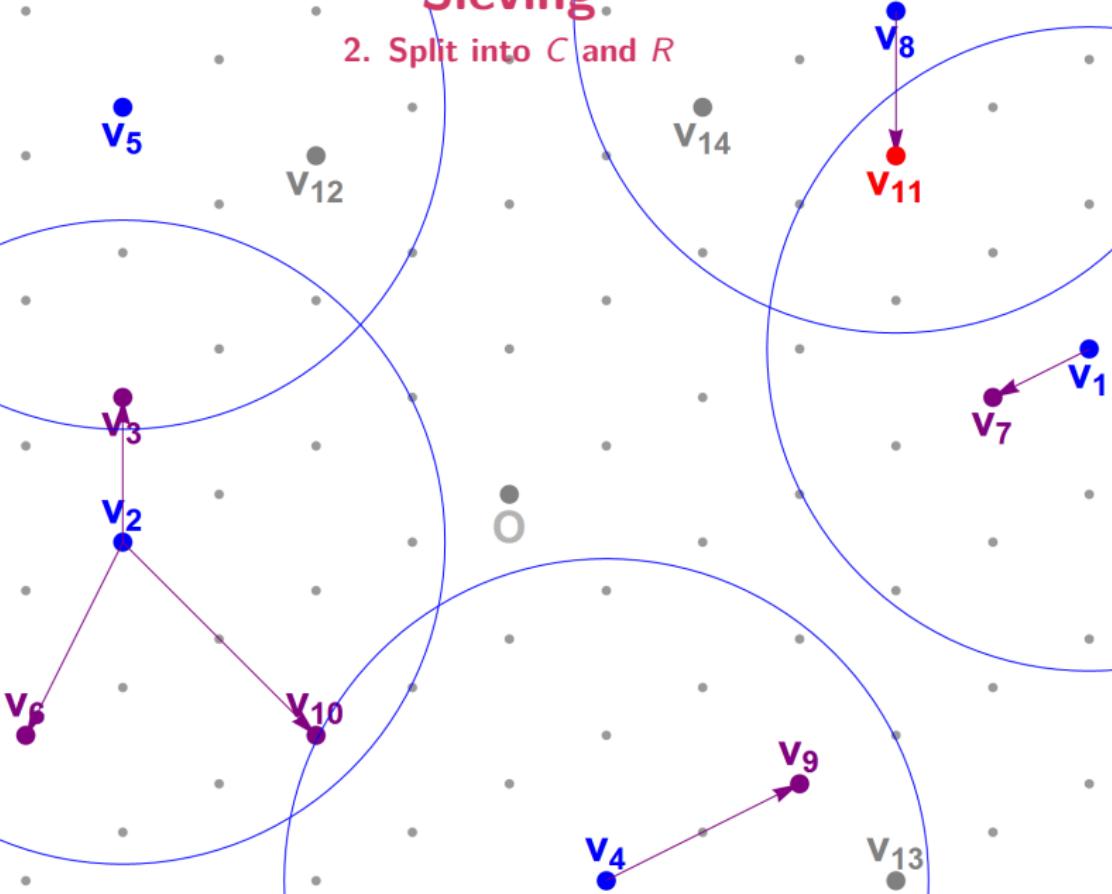
Sieving

2. Split into C and R



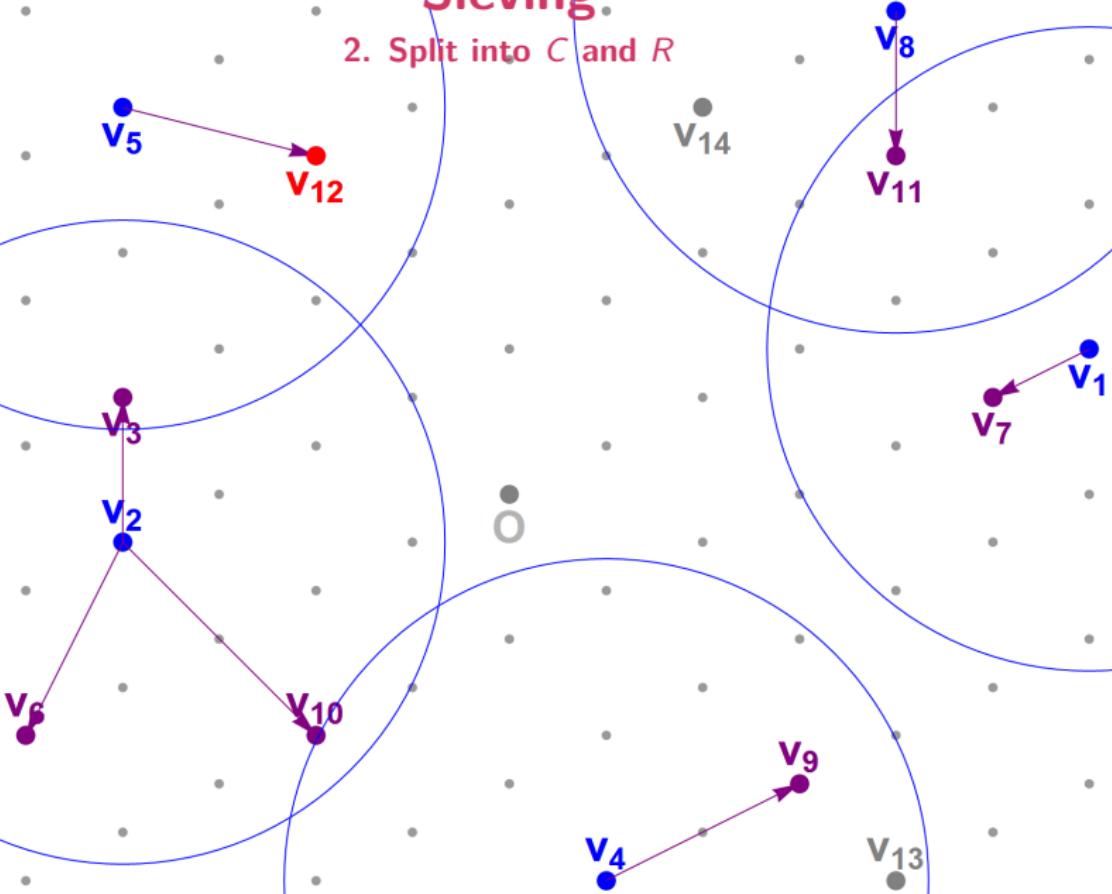
Sieving

2. Split into C and R



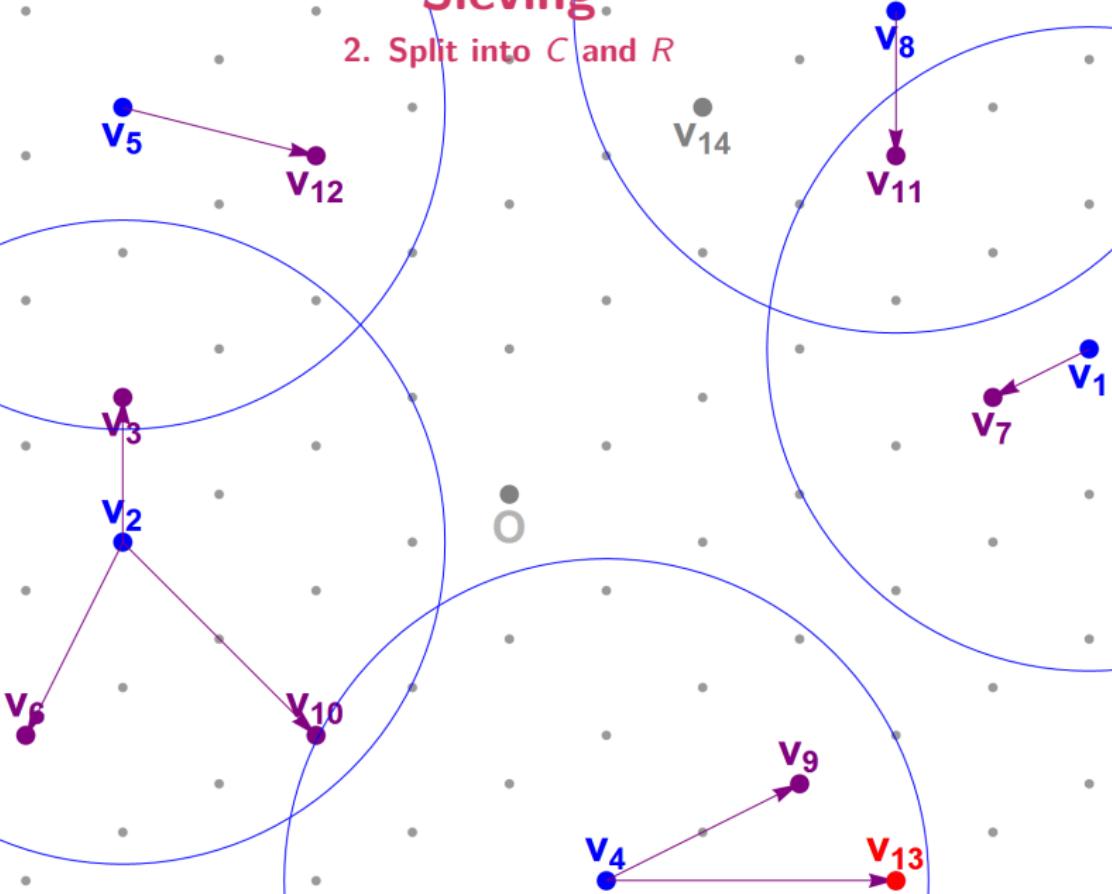
Sieving

2. Split into C and R



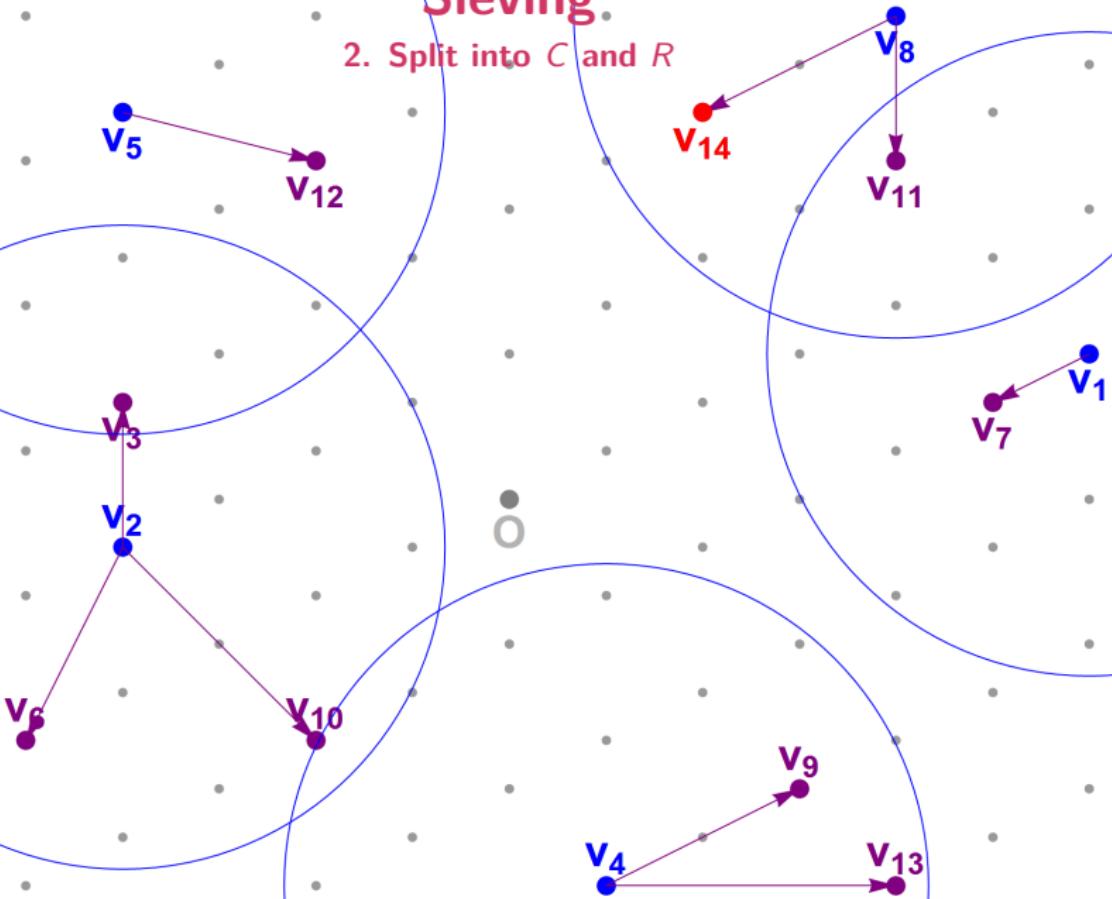
Sieving

2. Split into C and R



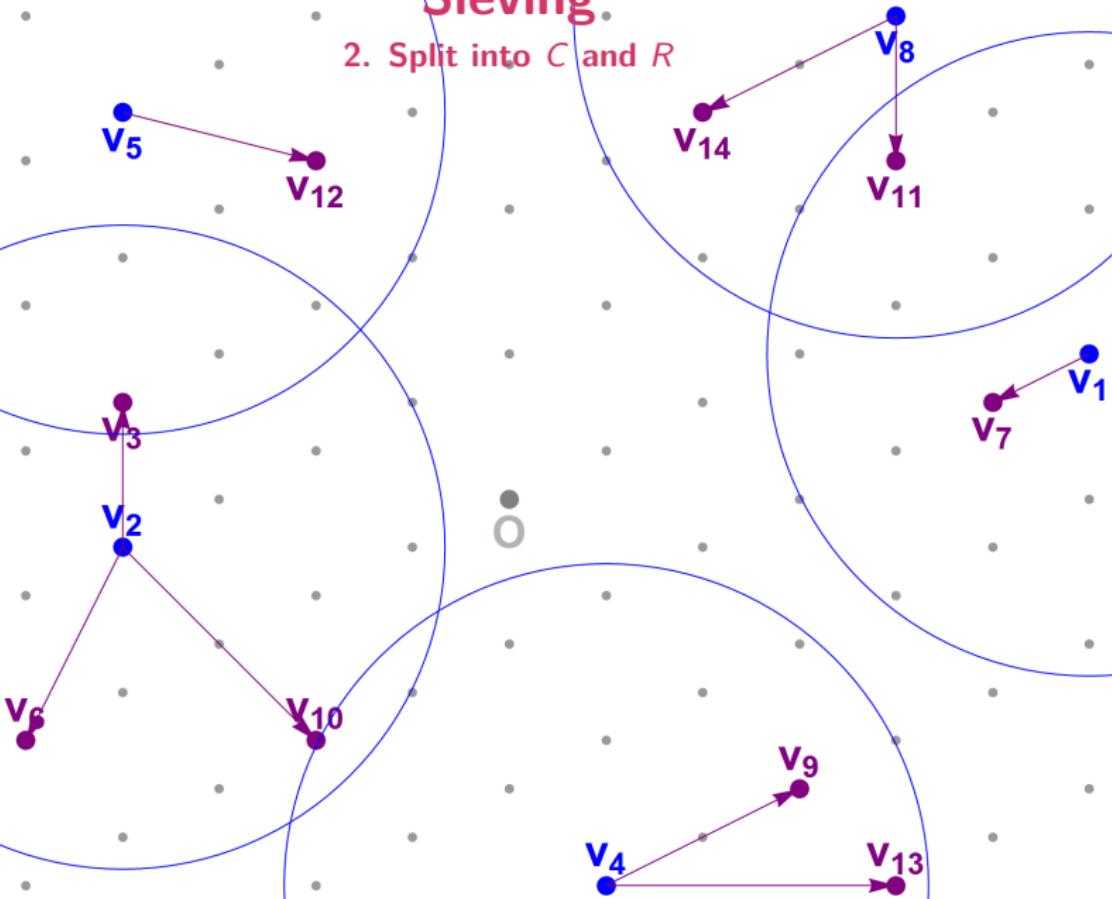
Sieving

2. Split into C and R



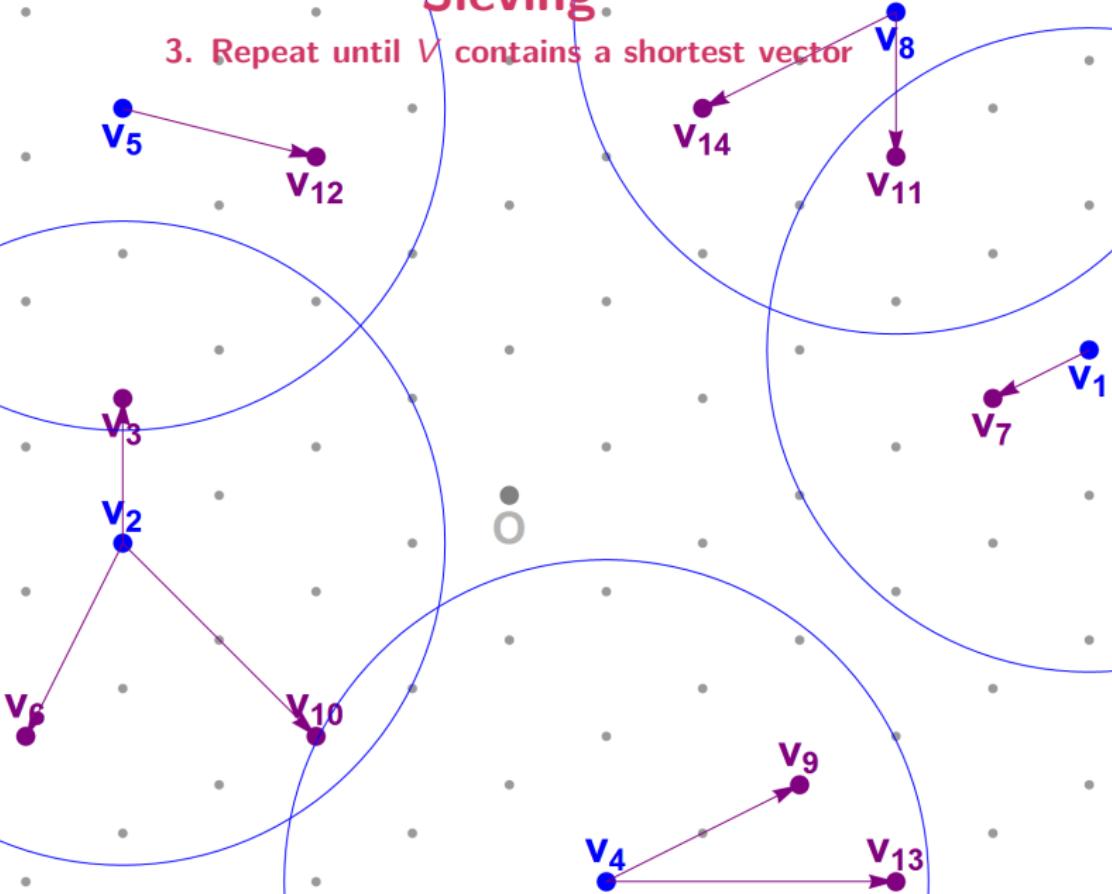
Sieving

2. Split into C and R



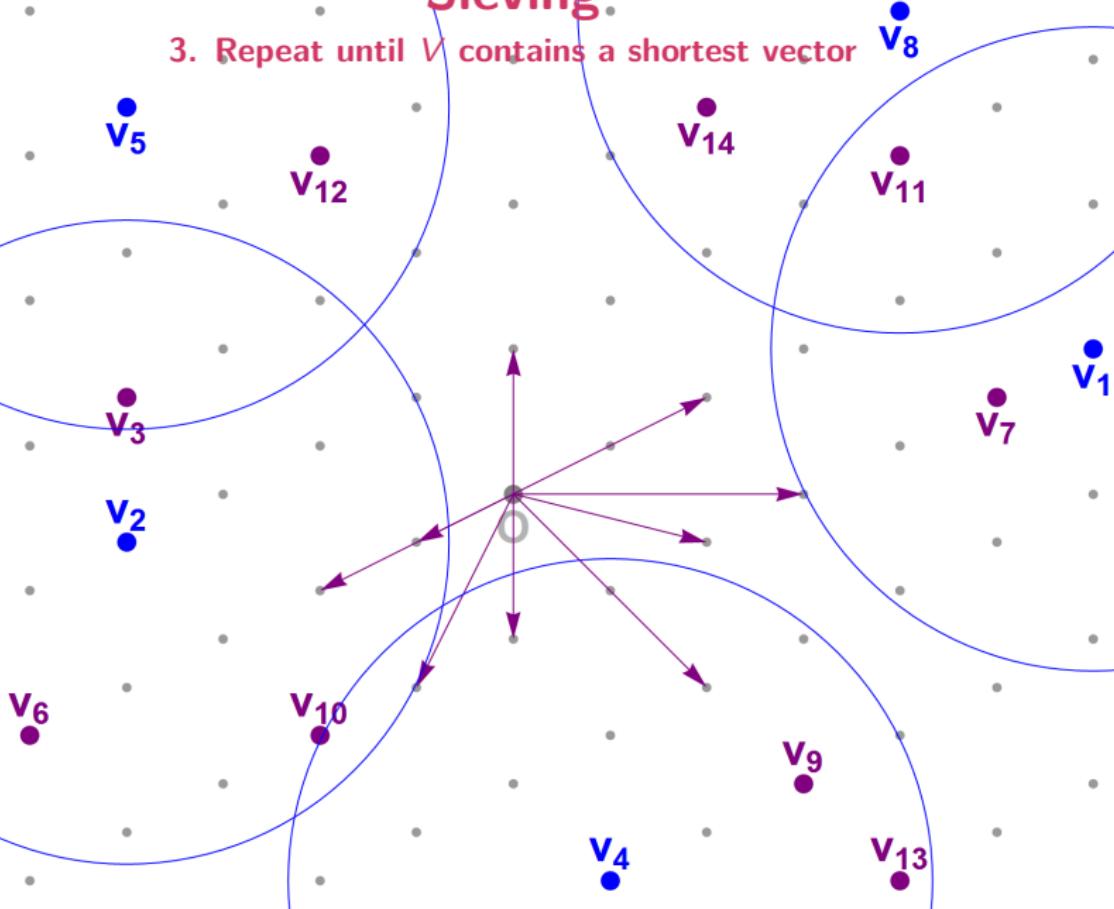
Sieving

3. Repeat until V contains a shortest vector



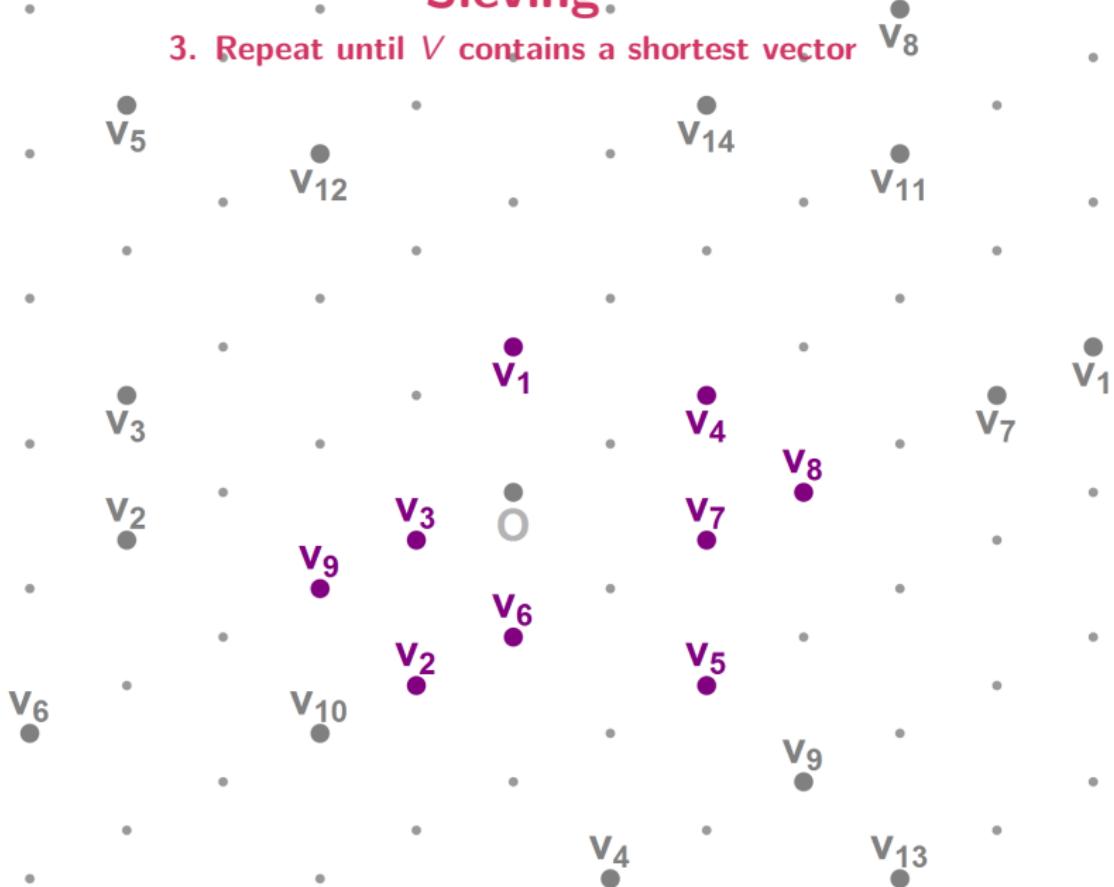
Sieving

3. Repeat until V contains a shortest vector



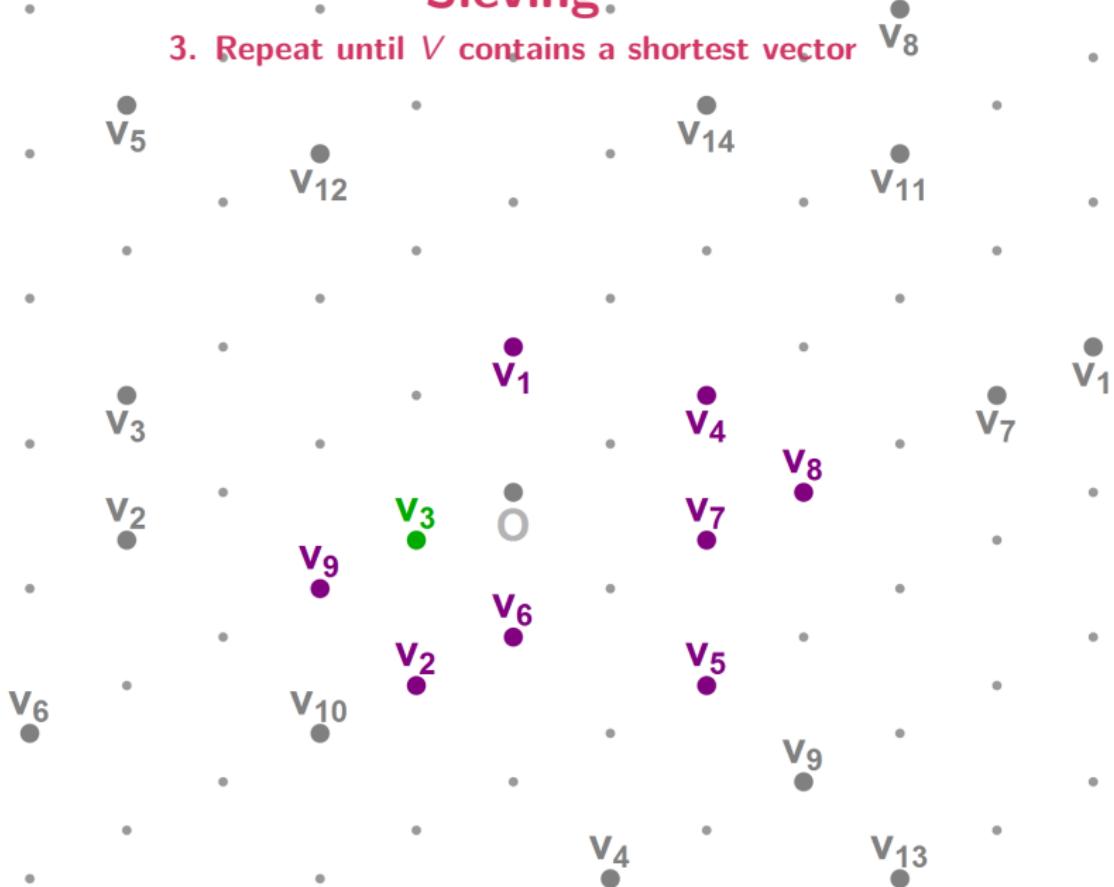
Sieving

3. Repeat until V contains a shortest vector



Sieving

3. Repeat until V contains a shortest vector



Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

Complexity?

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α
- Classical Time:
- Quantum Time:

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α
- Classical Time: $\approx 2^{\alpha n} \cdot 2^{\alpha n} = 2^{2\alpha n}$
- Quantum Time:

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α
- Classical Time: $\approx 2^{\alpha n} \cdot 2^{\alpha n} = 2^{2\alpha n}$
- Quantum Time: $\approx 2^{\alpha n} \cdot \sqrt{2^{\alpha n}} = 2^{\frac{3}{2}\alpha n}$

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α
- Classical Time: $\approx 2^{\alpha n} \cdot 2^{\alpha n} = 2^{2\alpha n}$
- Quantum Time: $\approx 2^{\alpha n} \cdot \sqrt{2^{\alpha n}} = 2^{\frac{3}{2}\alpha n}$
- Improvement: 25% in the exponent

Sieving

Invented in 2001 [AKS01]

Procedure:

1. Generate a long list V of random lattice vectors
2. Split V into two sets C (centers, cover) and R (rest):
 - ▶ Set $C = \emptyset$ and $R = \emptyset$
 - ▶ For each $v \in V$, find the closest $c \in C$
 - ▶ If $\|v - c\|$ is “large”, add v to C
 - ▶ If $\|v - c\|$ is “small”, add $v - c$ to R
3. Set $V = R$ and repeat until V contains a shortest vector

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some $\alpha \approx 0.21$
- Classical Time: $\approx 2^{\alpha n} \cdot 2^{\alpha n} = 2^{2\alpha n} \approx 2^{0.42n+o(n)}$ [NV08]
- Quantum Time: $\approx 2^{\alpha n} \cdot \sqrt{2^{\alpha n}} = 2^{\frac{3}{2}\alpha n} \approx 2^{0.31n+o(n)}$ [LMP13]
- Improvement: 25% in the exponent

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Search C for a shortest vector

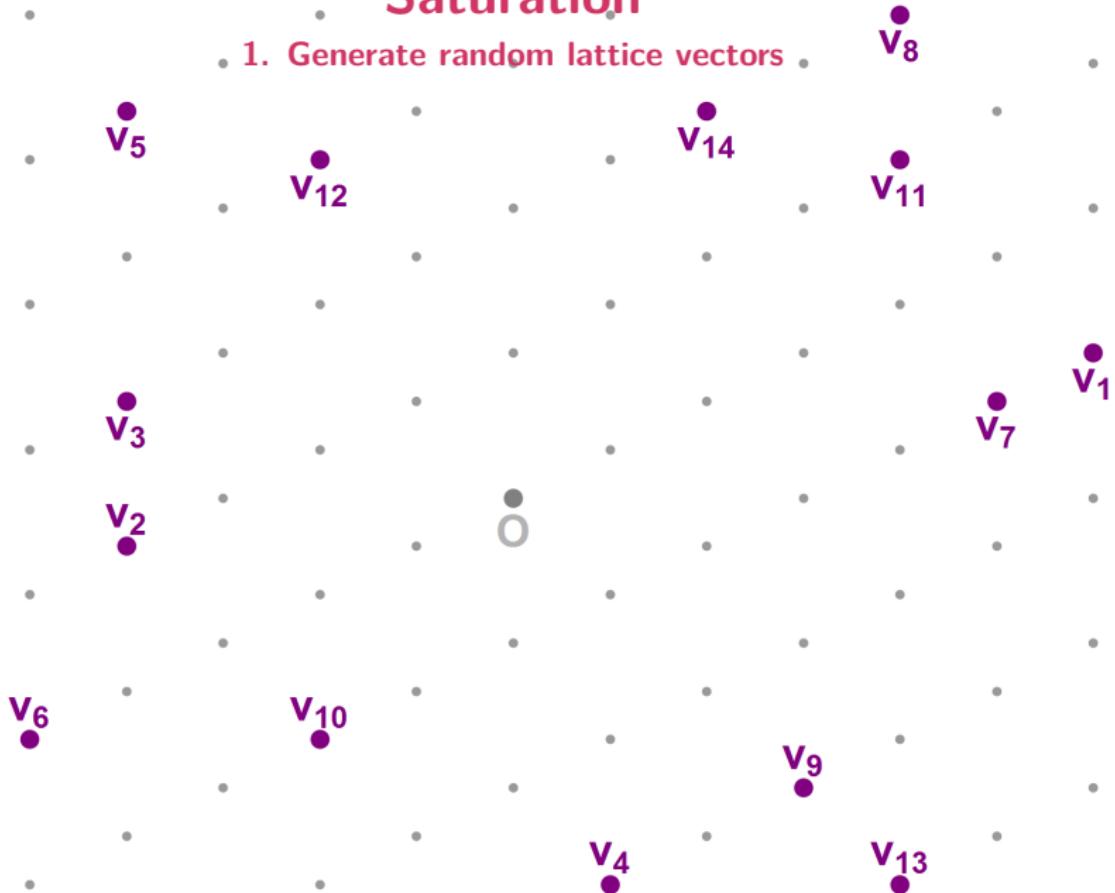
Saturation

1. Generate random lattice vectors



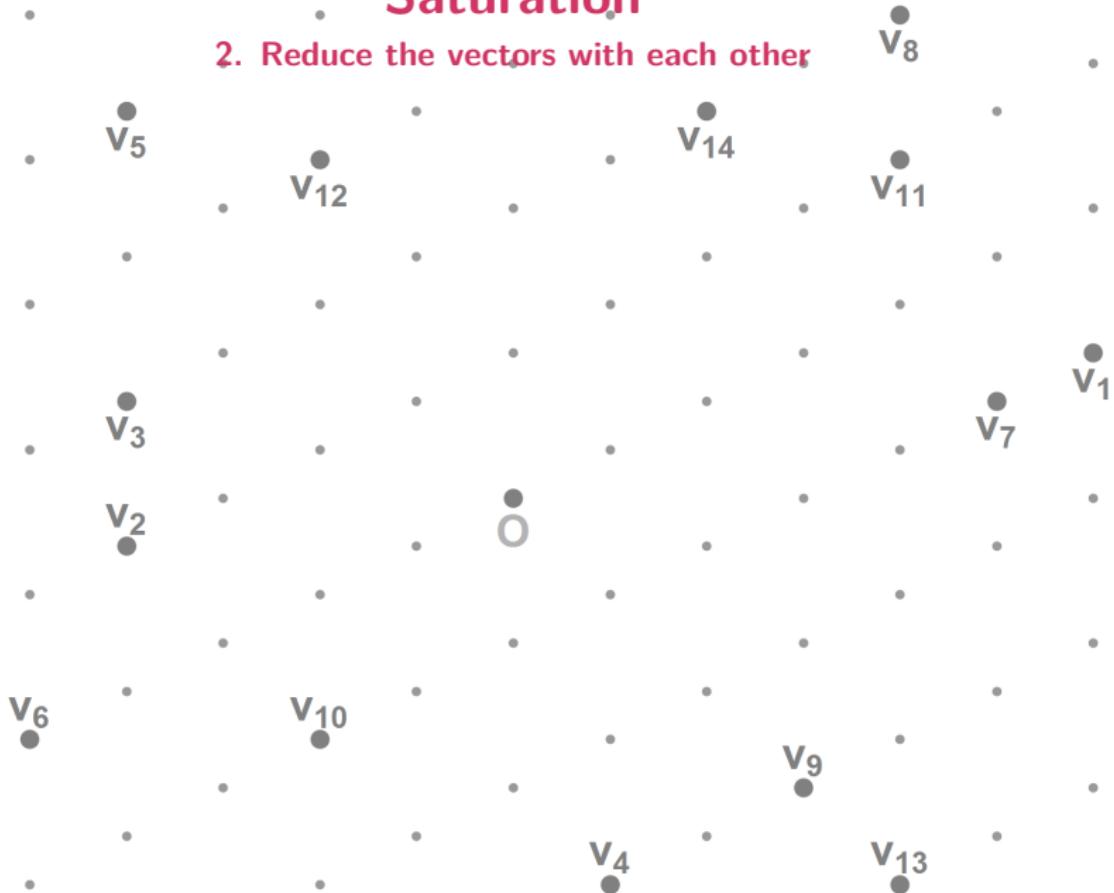
Saturation

1. Generate random lattice vectors



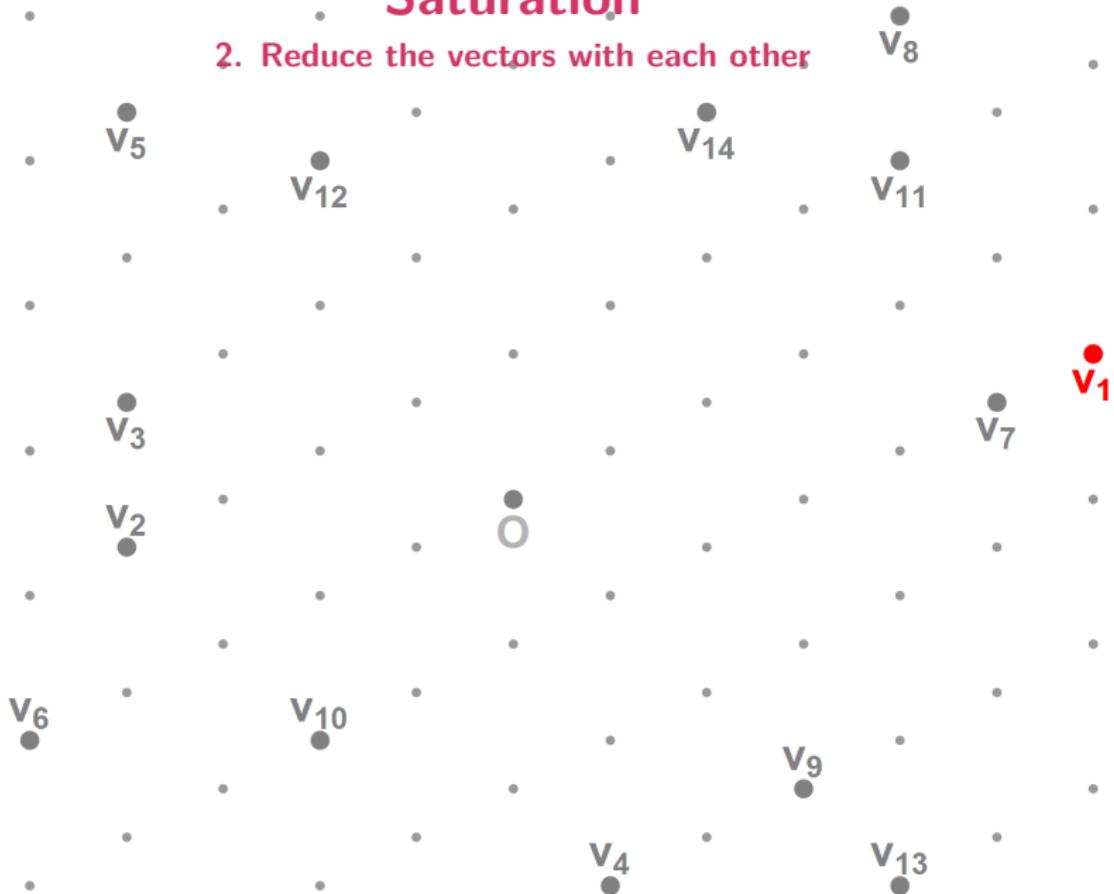
Saturation

2. Reduce the vectors with each other



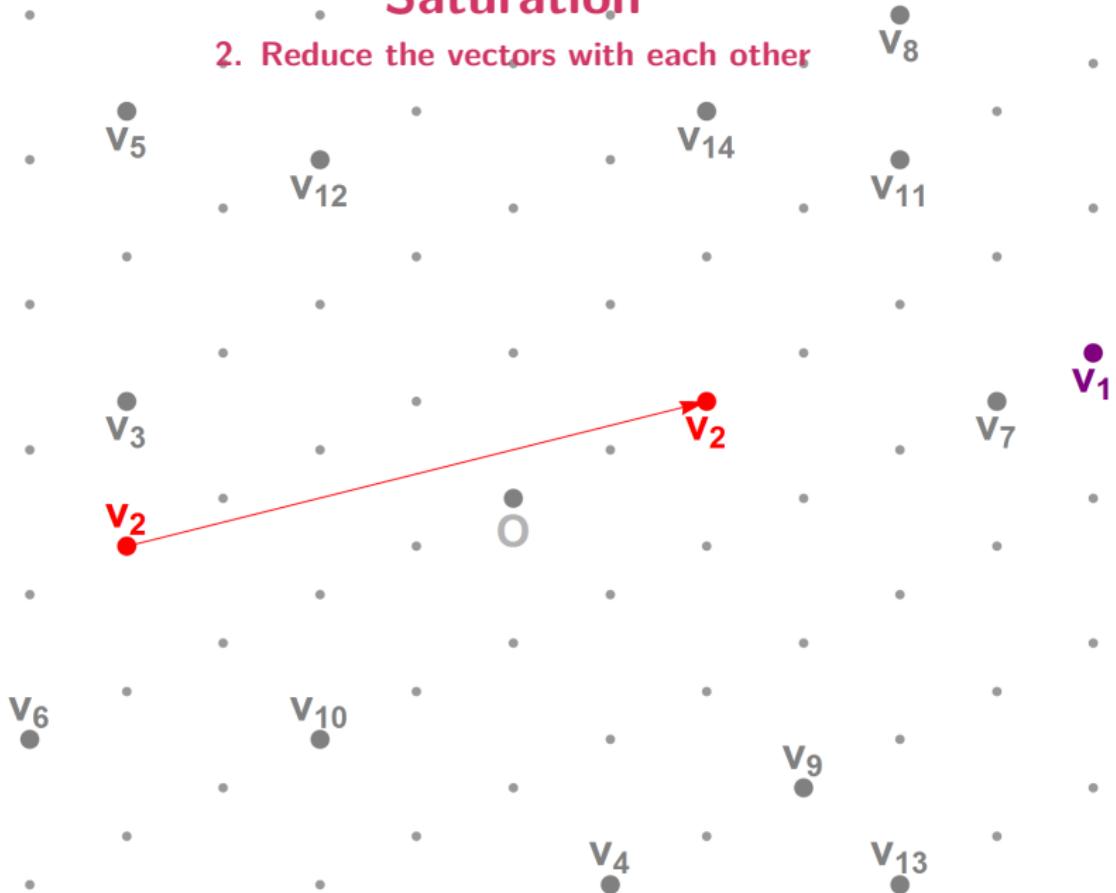
Saturation

2. Reduce the vectors with each other



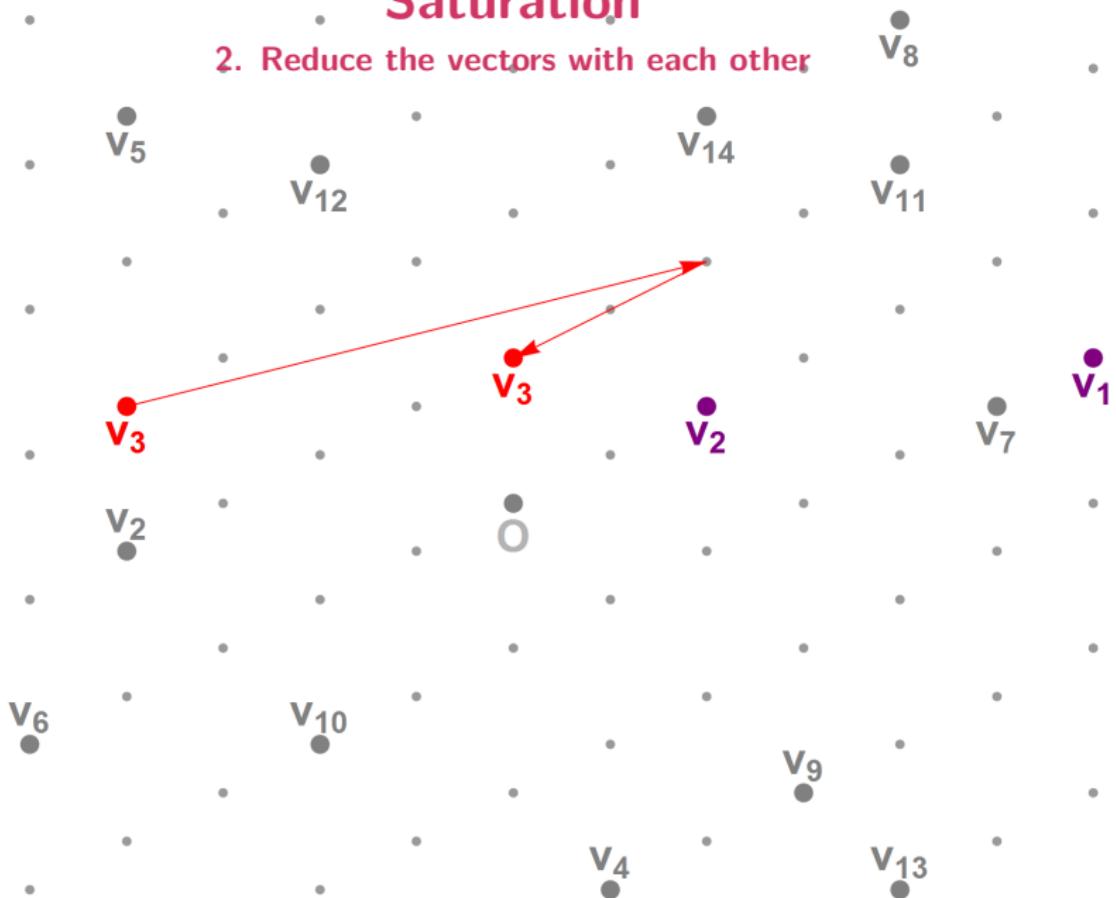
Saturation

2. Reduce the vectors with each other



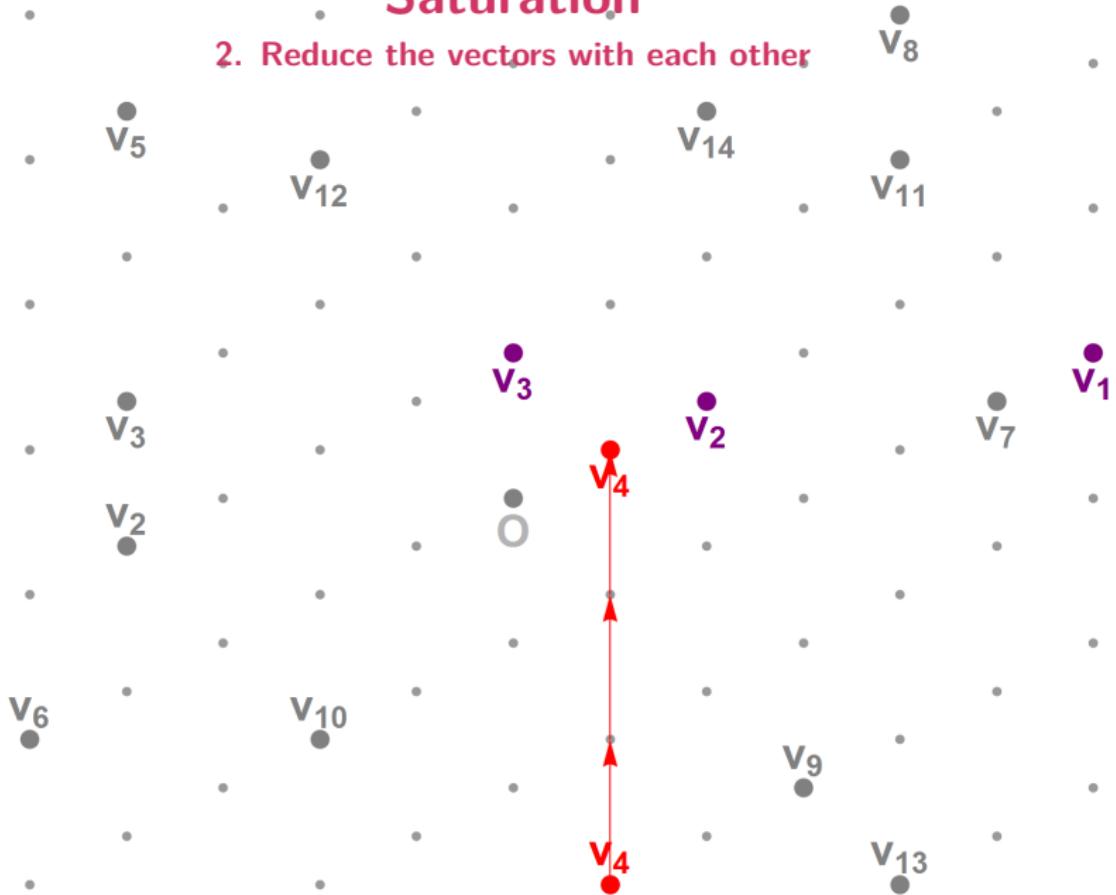
Saturation

2. Reduce the vectors with each other



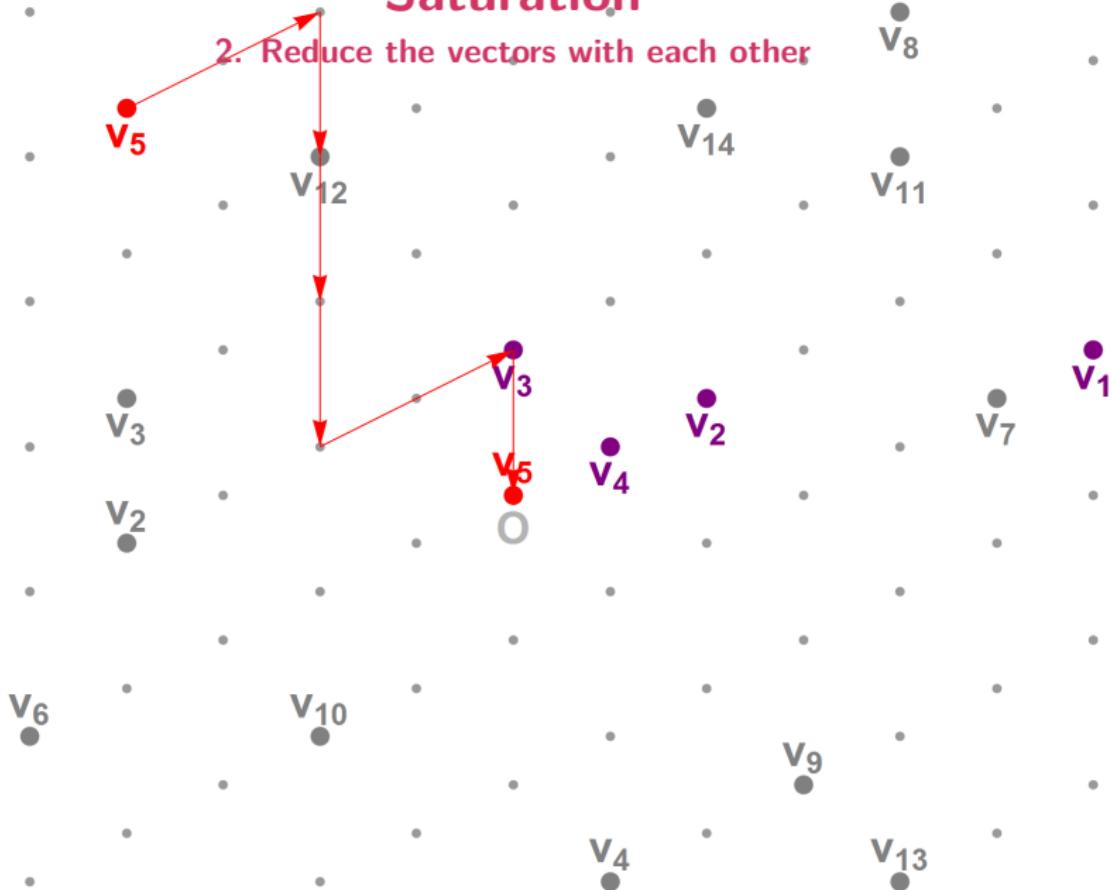
Saturation

2. Reduce the vectors with each other



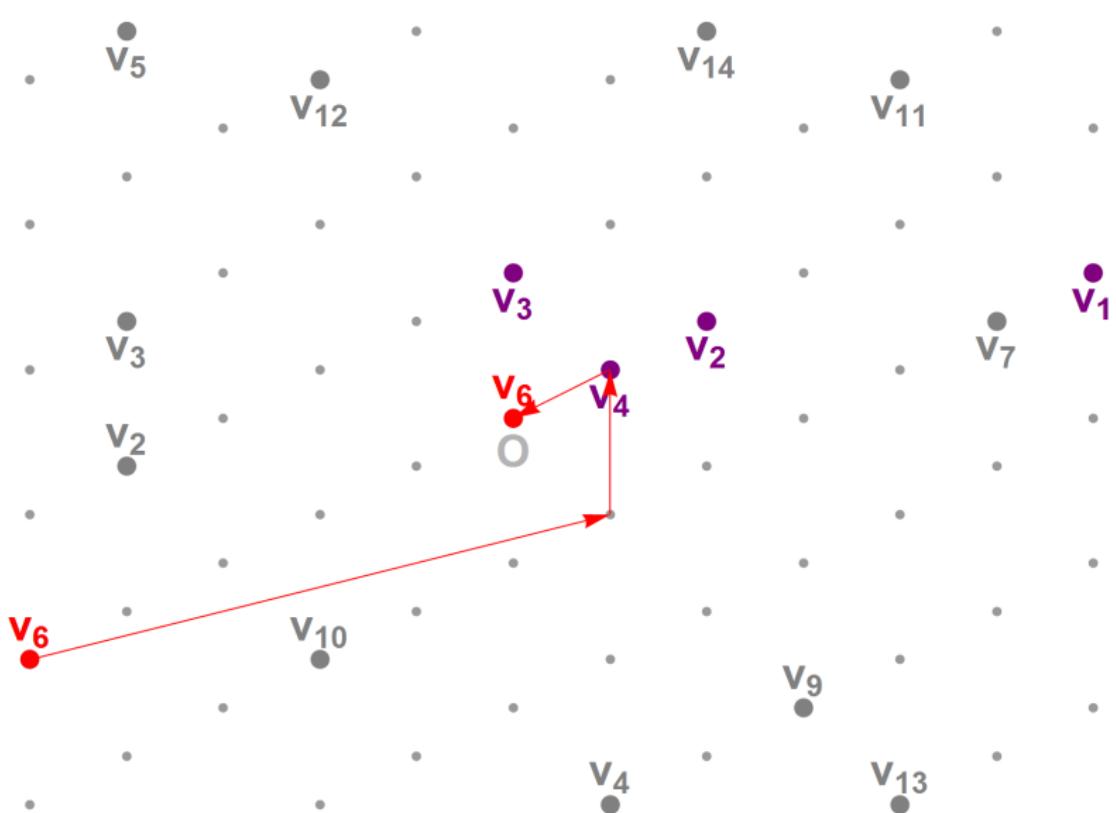
Saturation

2. Reduce the vectors with each other



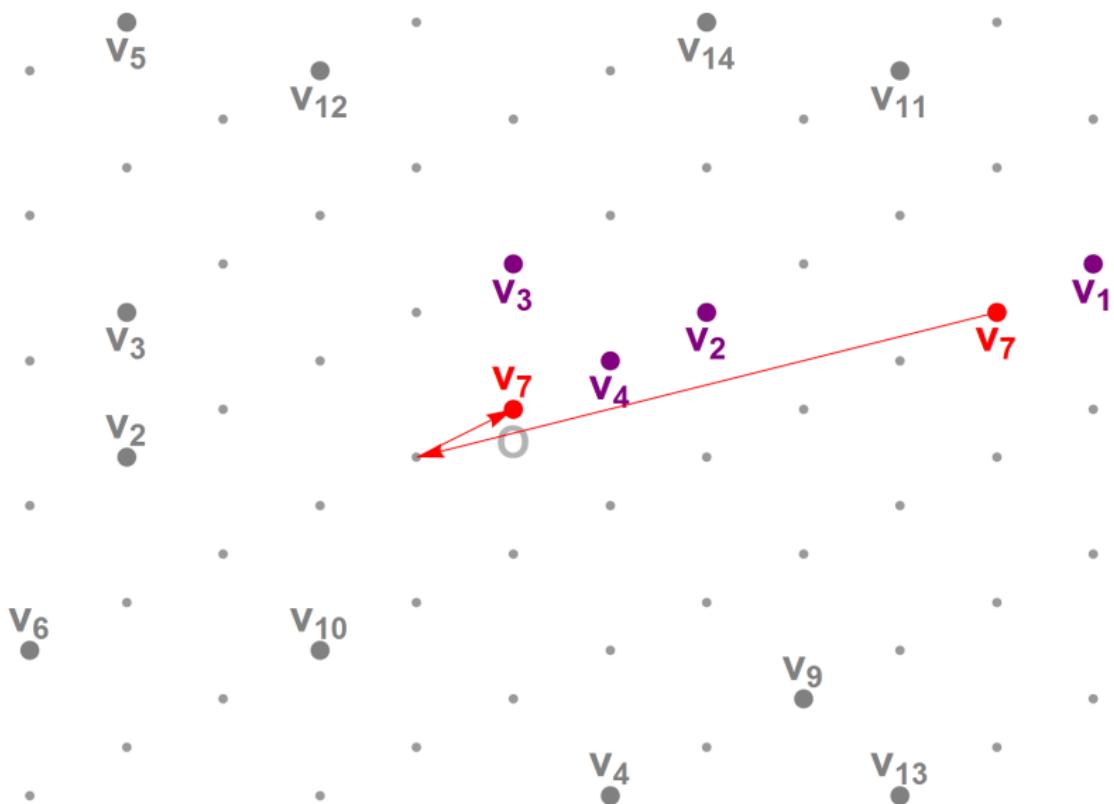
Saturation

2. Reduce the vectors with each other



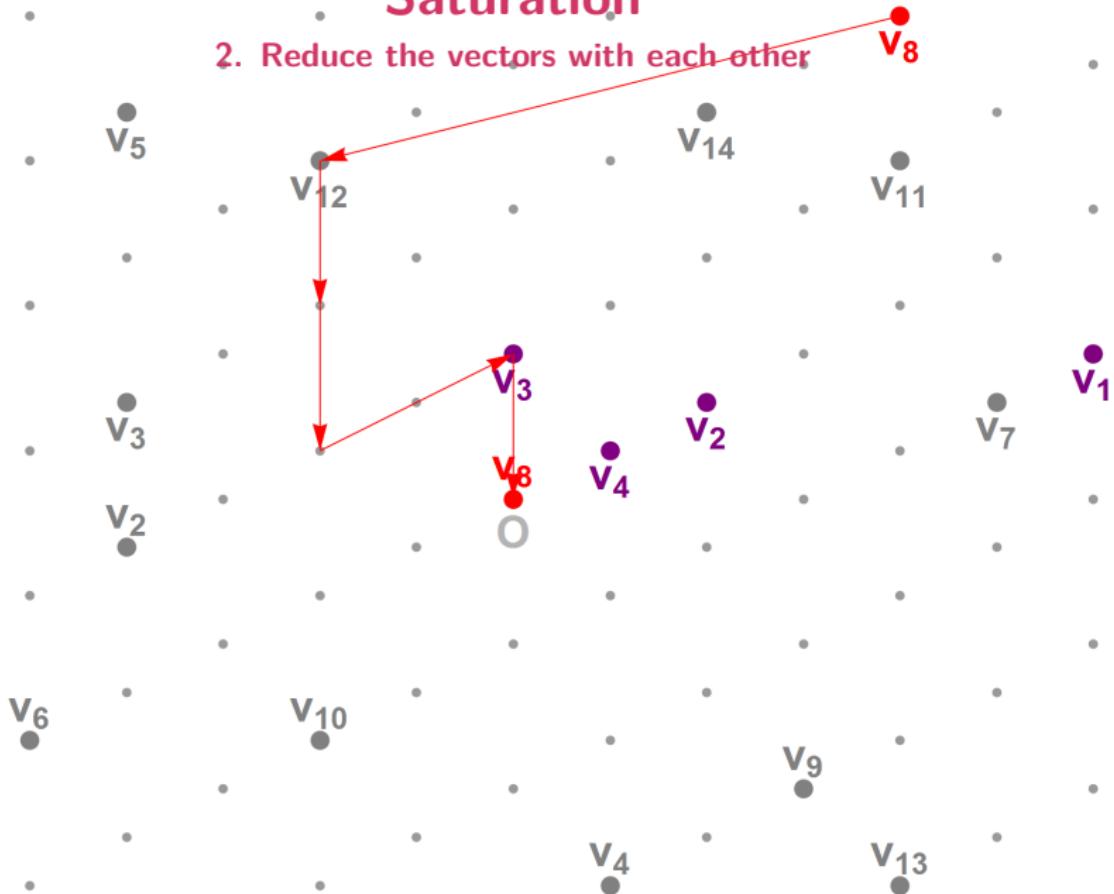
Saturation

2. Reduce the vectors with each other



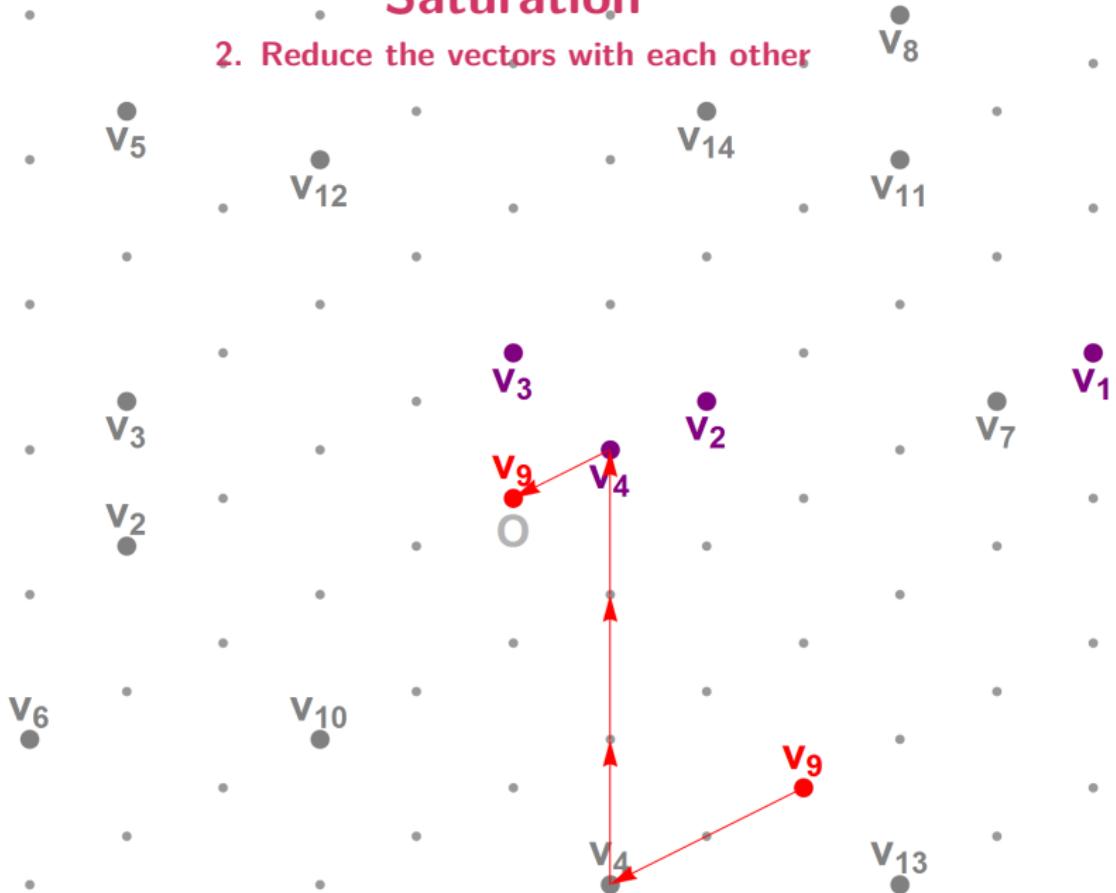
Saturation

2. Reduce the vectors with each other



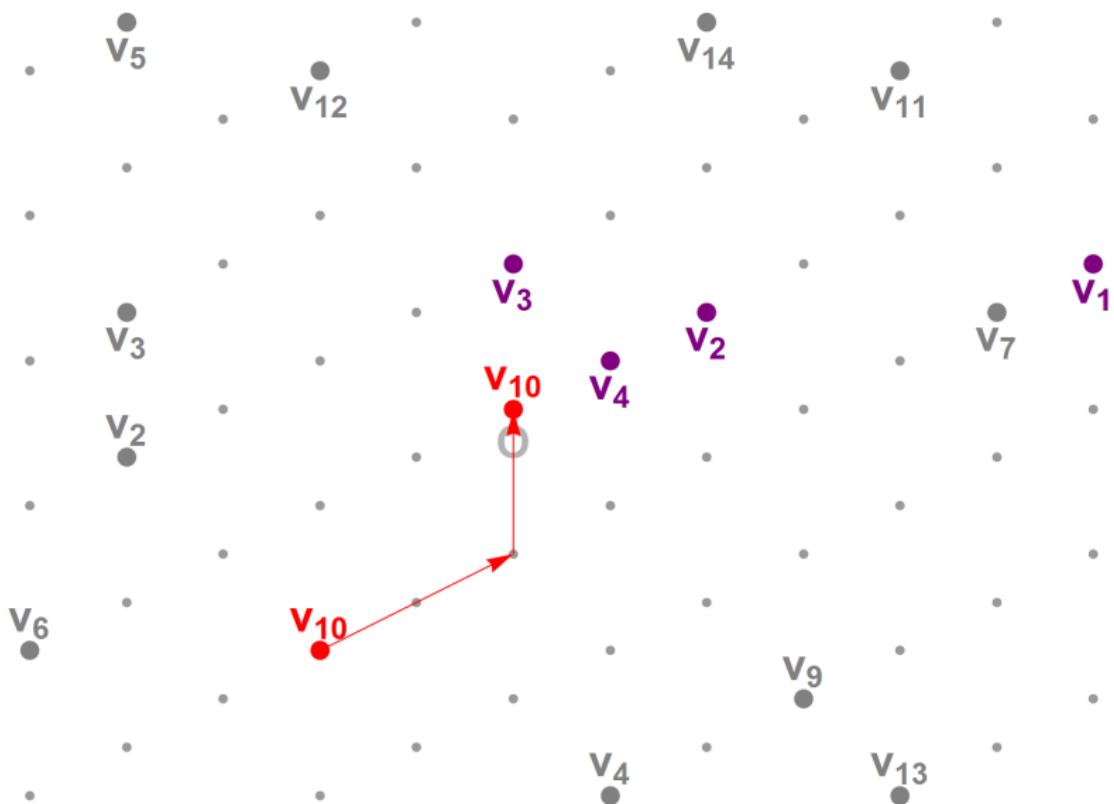
Saturation

2. Reduce the vectors with each other



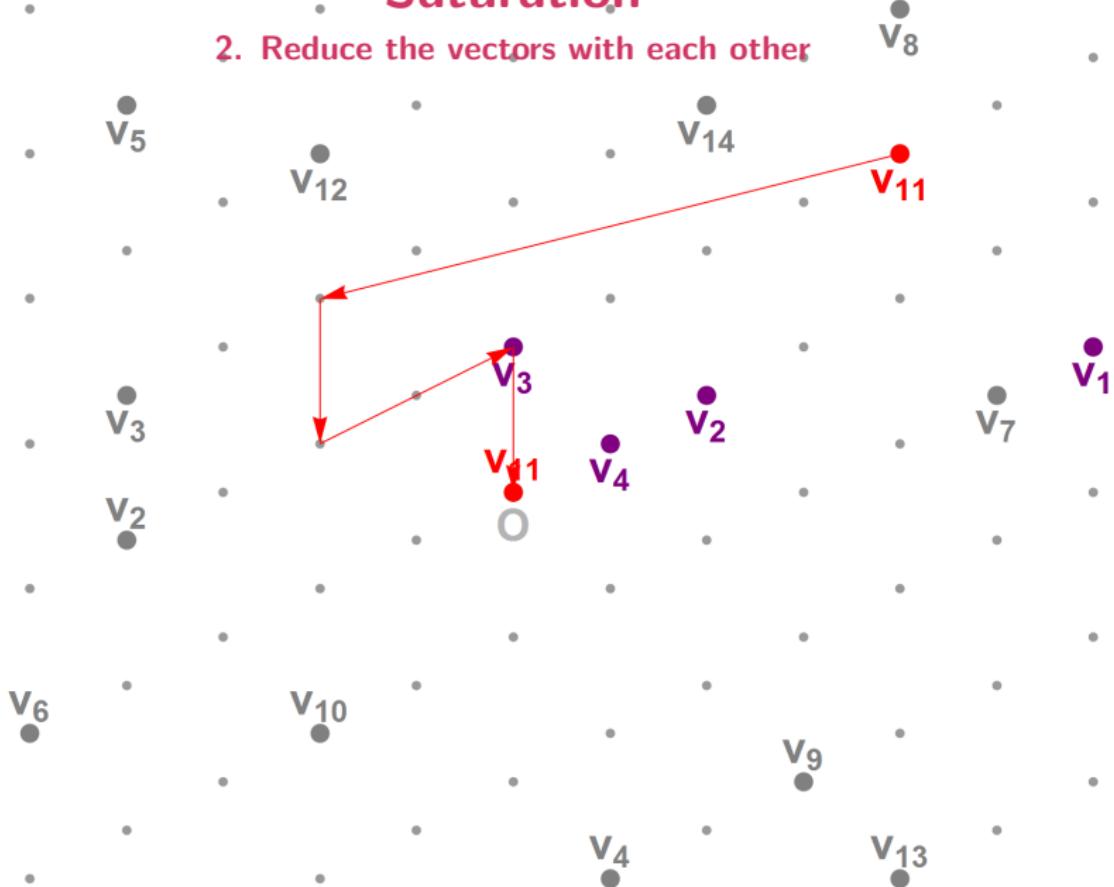
Saturation

2. Reduce the vectors with each other



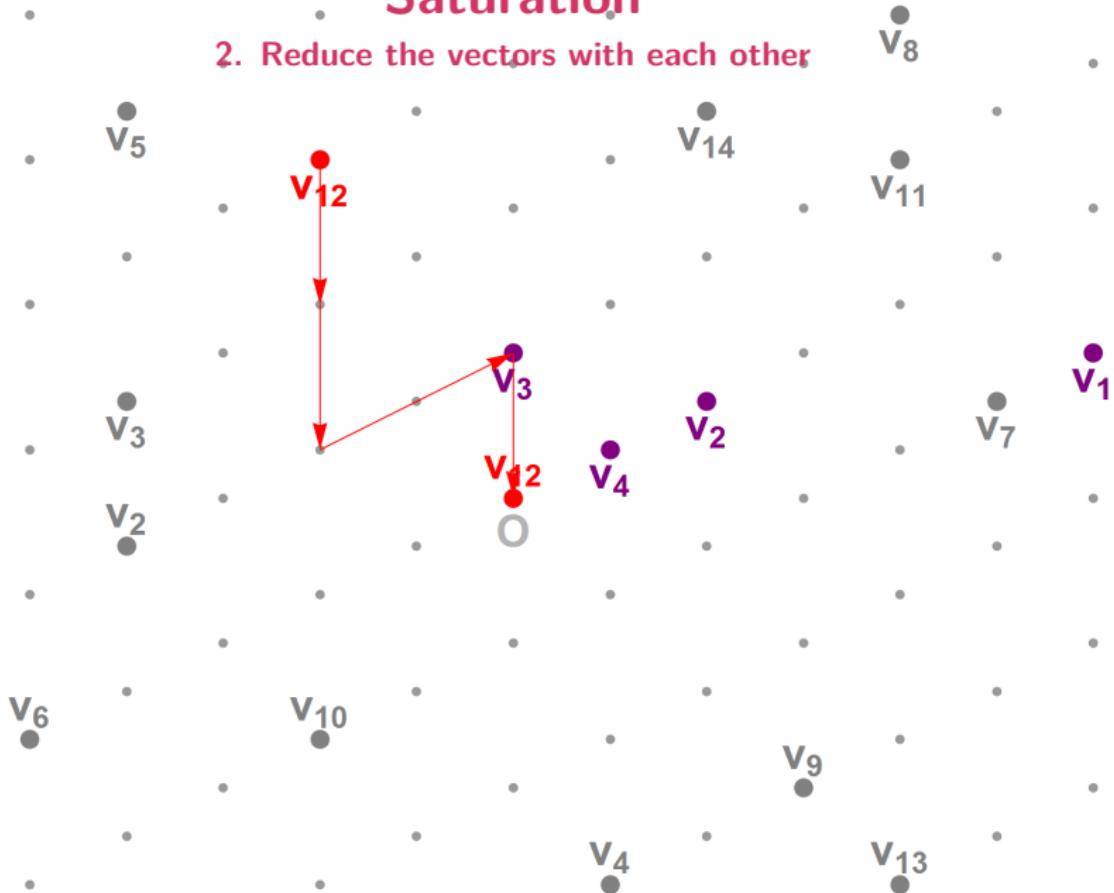
Saturation

2. Reduce the vectors with each other



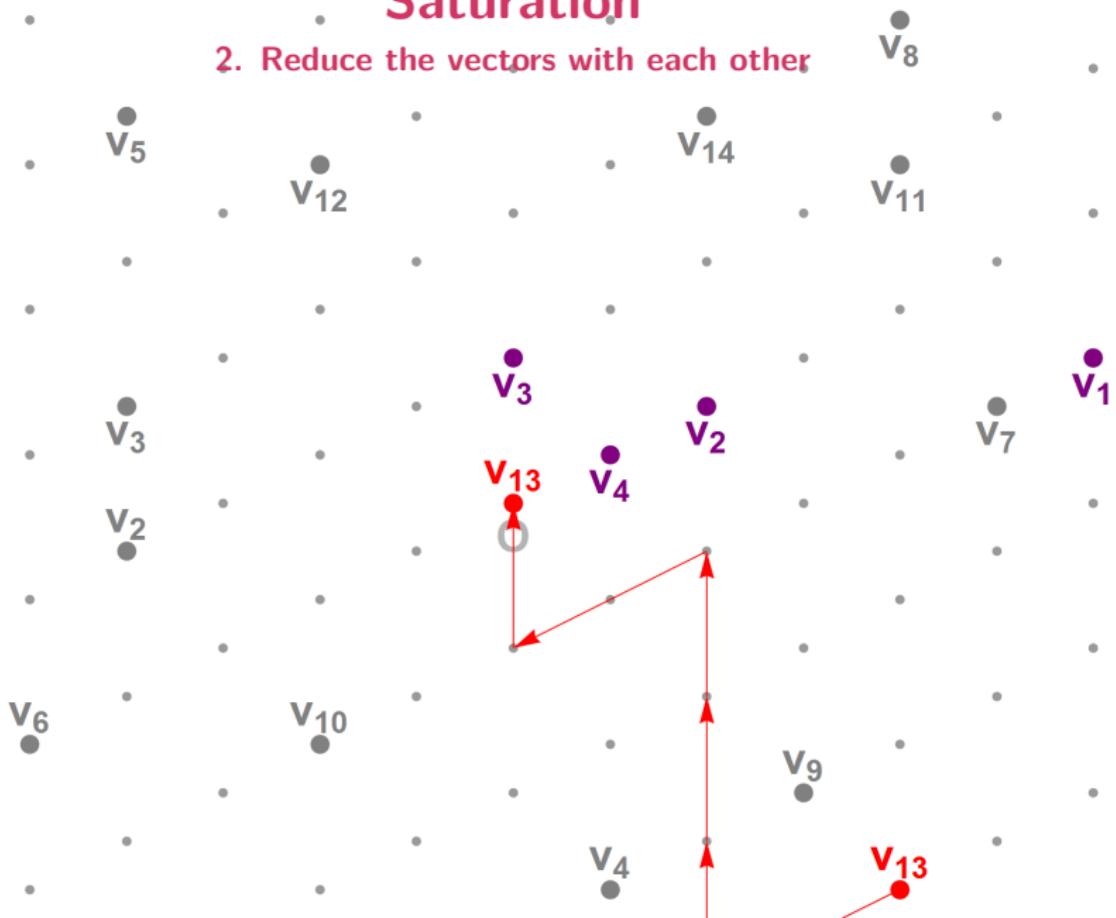
Saturation

2. Reduce the vectors with each other



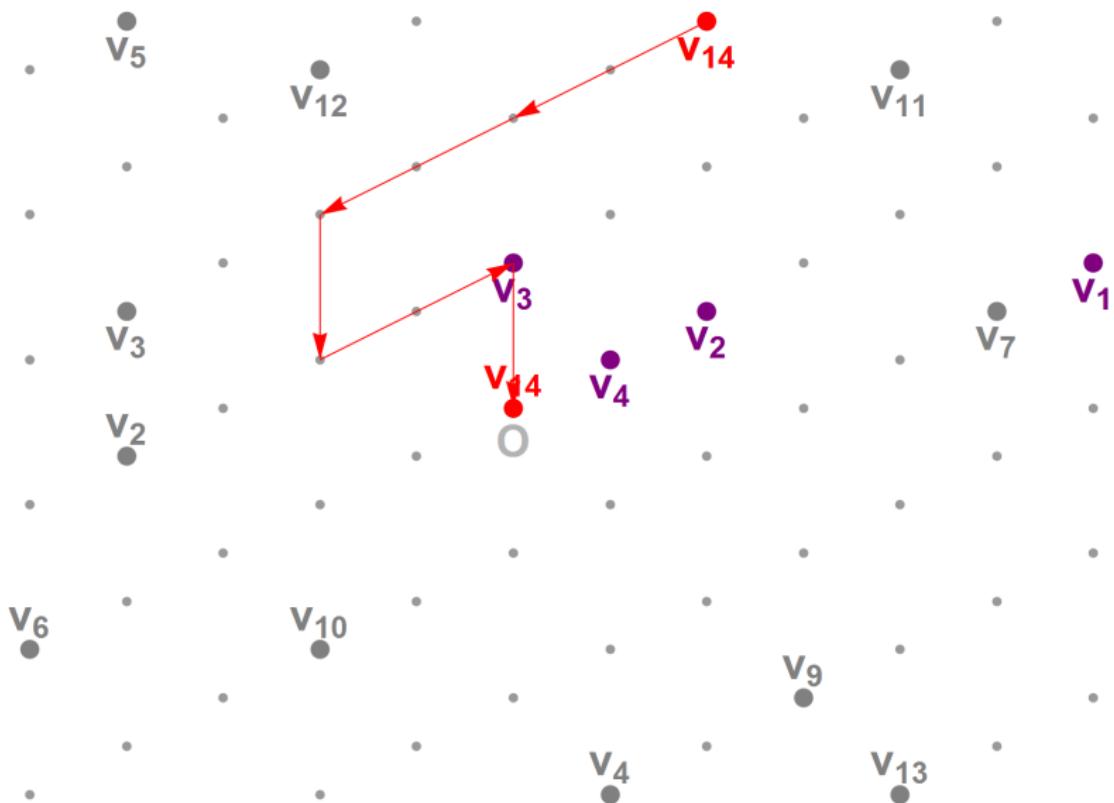
Saturation

2. Reduce the vectors with each other



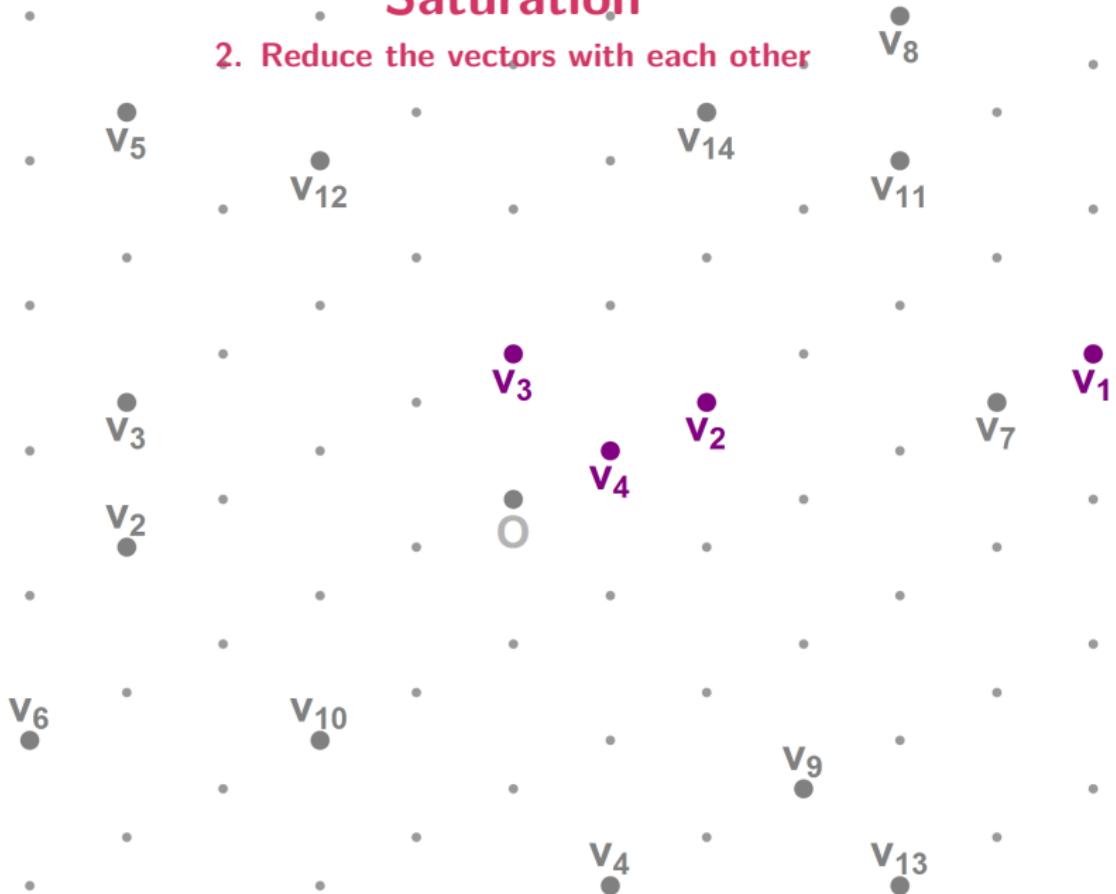
Saturation

2. Reduce the vectors with each other



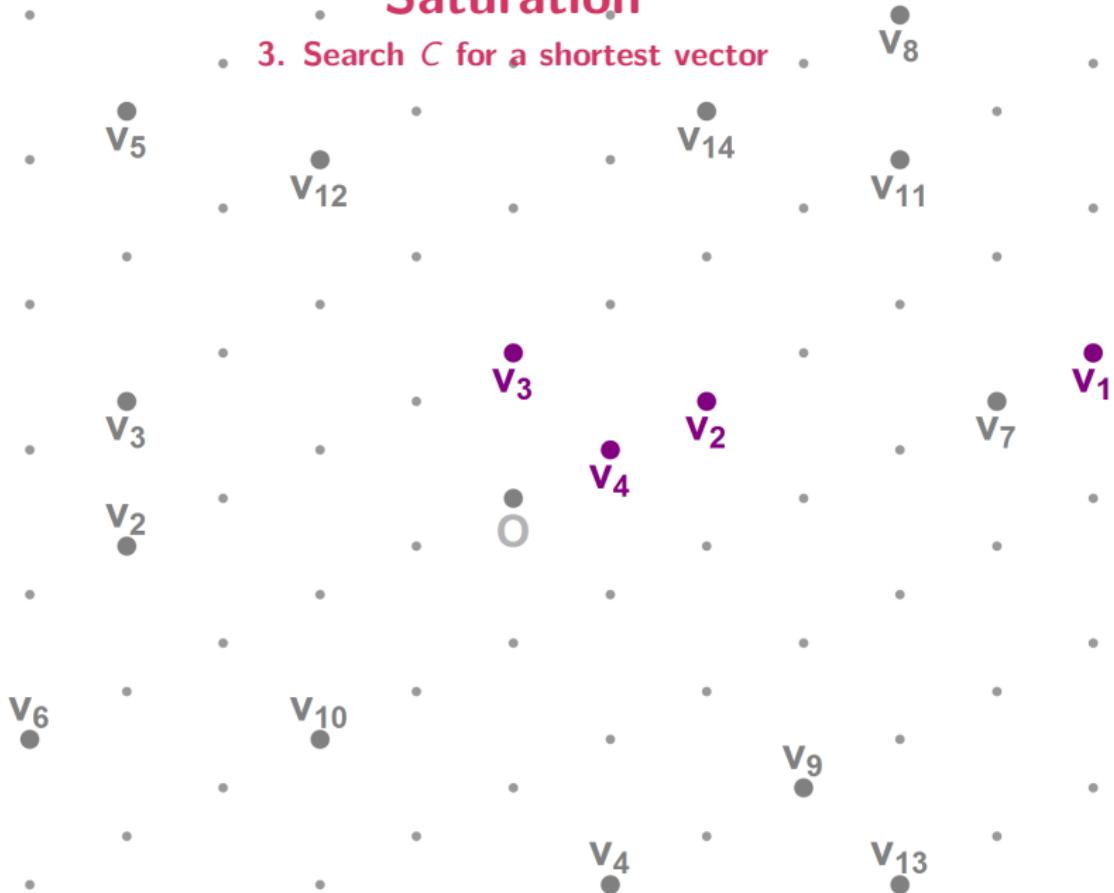
Saturation

2. Reduce the vectors with each other



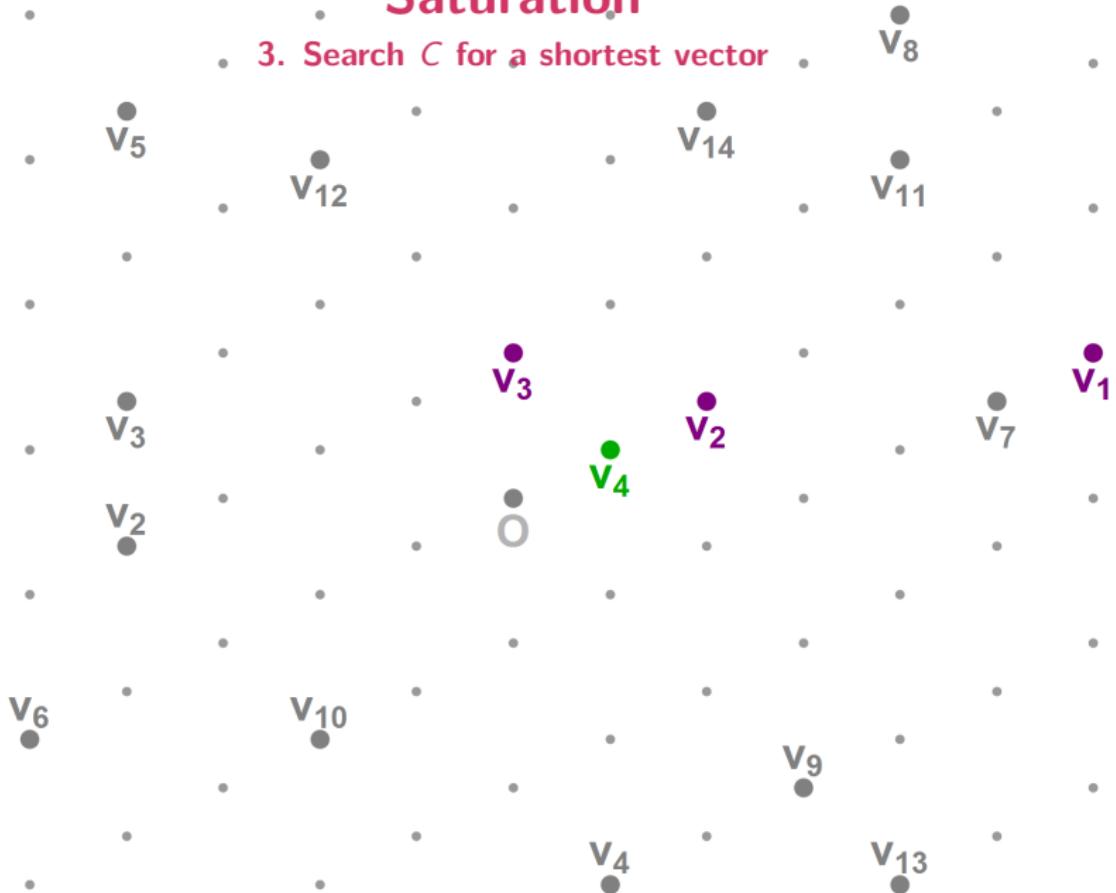
Saturation

3. Search C for a shortest vector



Saturation

3. Search C for a shortest vector



Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Find a shortest vector among the reduced vectors

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Find a shortest vector among the reduced vectors

Complexity?

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Find a shortest vector among the reduced vectors

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Find a shortest vector among the reduced vectors

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α
- Classical Time:
- Quantum Time:

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Find a shortest vector among the reduced vectors

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α
- Classical Time: $\approx 2^{\alpha n} \cdot 2^{\alpha n} = 2^{2\alpha n}$
- Quantum Time:

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Find a shortest vector among the reduced vectors

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α
- Classical Time: $\approx 2^{\alpha n} \cdot 2^{\alpha n} = 2^{2\alpha n}$
- Quantum Time: $\approx 2^{\alpha n} \cdot \sqrt{2^{\alpha n}} = 2^{\frac{3}{2}\alpha n}$

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Find a shortest vector among the reduced vectors

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some α
- Classical Time: $\approx 2^{\alpha n} \cdot 2^{\alpha n} = 2^{2\alpha n}$
- Quantum Time: $\approx 2^{\alpha n} \cdot \sqrt{2^{\alpha n}} = 2^{\frac{3}{2}\alpha n}$
- Improvement: $\approx 25\%$ in the exponent

Saturation

Invented in 2009 [MV09]

Procedure:

1. Generate a long list V of random lattice vectors
2. “Reduce the vectors with each other”:
 - ▶ Set $C = \emptyset$
 - ▶ For each $v \in V$, find the closest vector $c \in C$
 - ▶ If $\|v - c\| < \|v\|$, set $v \leftarrow v - c$ and find new closest $c \in C$
 - ▶ If $\|v - c\| \geq \|v\|$, add v to C
3. Find a shortest vector among the reduced vectors

Complexity?

- Space: $|V|, |C|, |R| \leq 2^{\alpha n}$ for some $\alpha \approx 0.21$
- Classical Time: $\approx 2^{\alpha n} \cdot 2^{\alpha n} = 2^{2\alpha n} \approx 2^{0.52n+o(n)}$ [MV09]
- Quantum Time: $\approx 2^{\alpha n} \cdot \sqrt{2^{\alpha n}} = 2^{\frac{3}{2}\alpha n} \approx 2^{0.39n+o(n)}$ [LMP13]
- Improvement: $\approx 25\%$ in the exponent

Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

Procedure:

1. “Guess” the n th coordinate (coefficient of basis vector b_n)

Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

Procedure:

1. “Guess” the n th coordinate (coefficient of basis vector b_n)
2. Find a shortest vector, given the n th coordinate

Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

Procedure:

1. “Guess” the n th coordinate (coefficient of basis vector b_n)
2. Find a shortest vector, given the n th coordinate
3. Search for a shortest vector among all of these vectors

Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

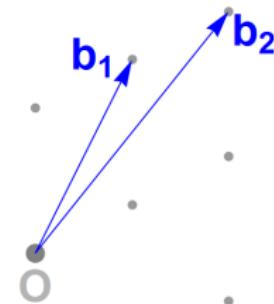
Procedure:

1. “Guess” the n th coordinate (coefficient of basis vector b_n)
2. Find a shortest vector, given the n th coordinate
3. Search for a shortest vector among all of these vectors

Recursive: Reduces SVP_n (CVP_n) to several instances of CVP_{n-1}

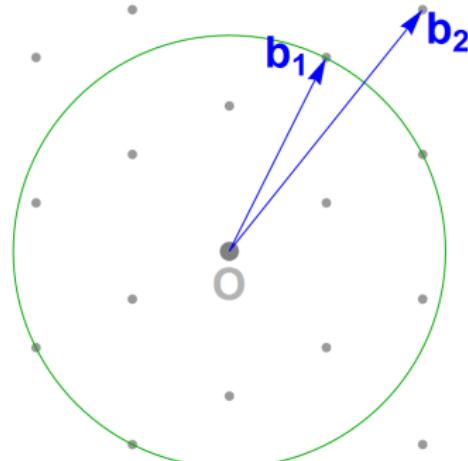
Enumeration

Possible coefficients of b_2



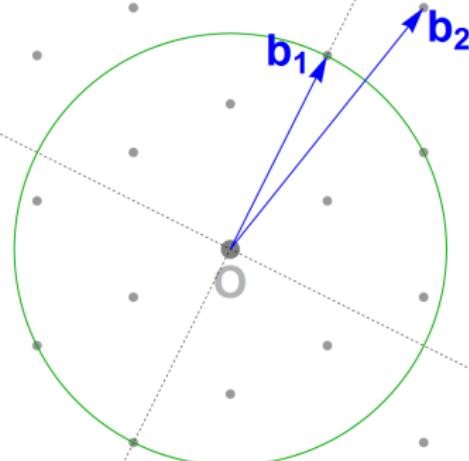
Enumeration

Possible coefficients of b_2



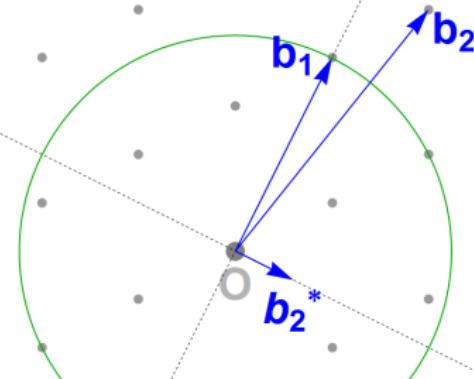
Enumeration

Possible coefficients of b_2



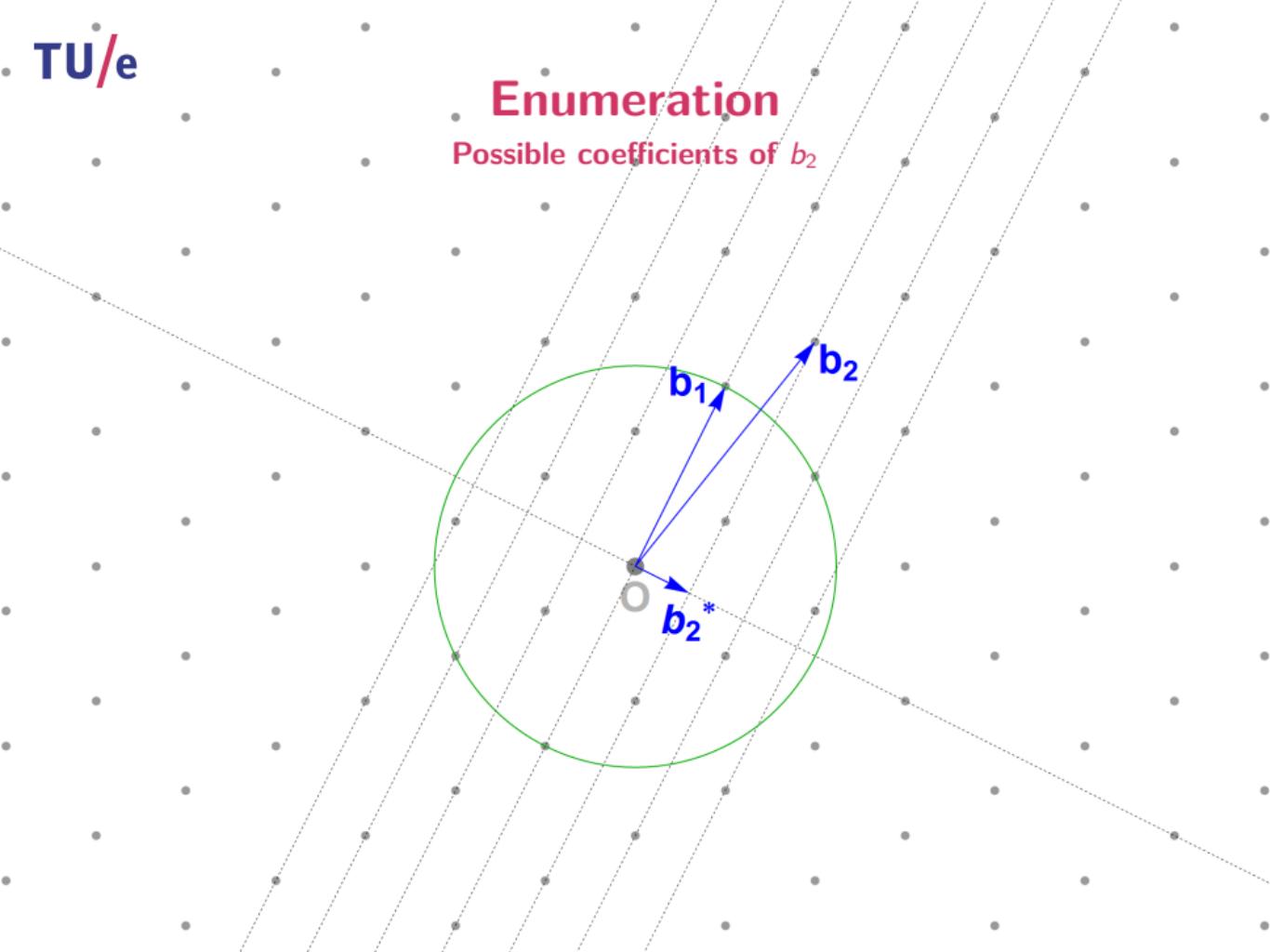
Enumeration

Possible coefficients of b_2



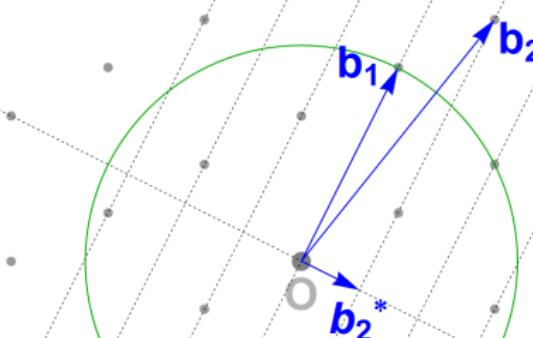
Enumeration

Possible coefficients of b_2



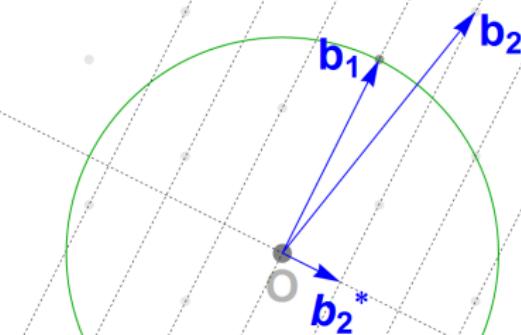
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP_1



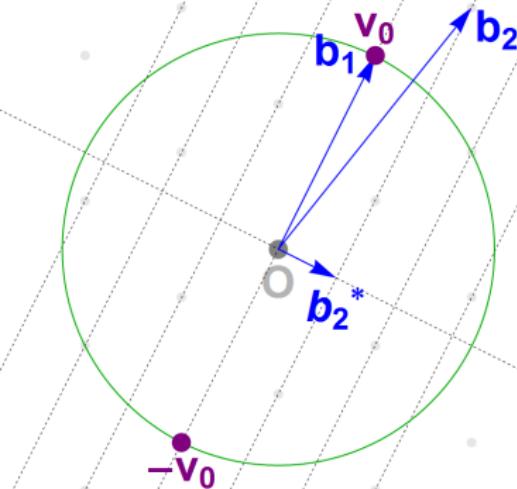
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP₁



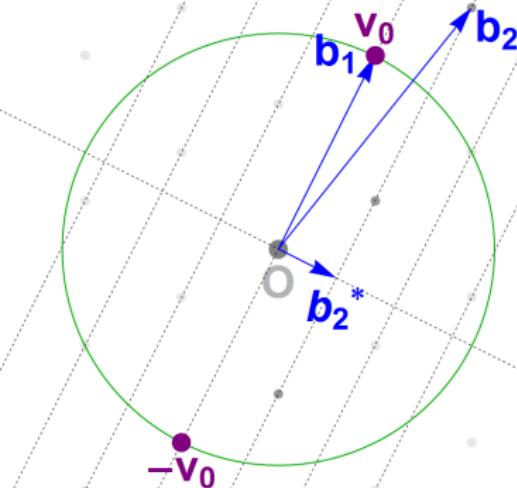
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP₁



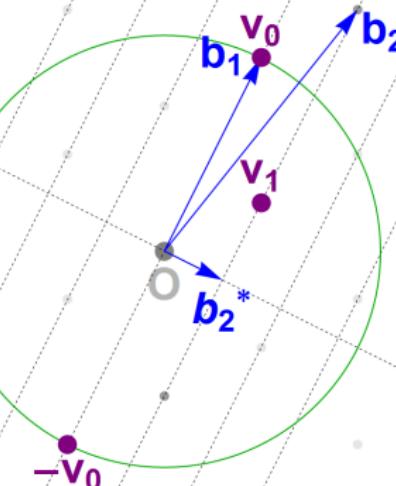
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP_1



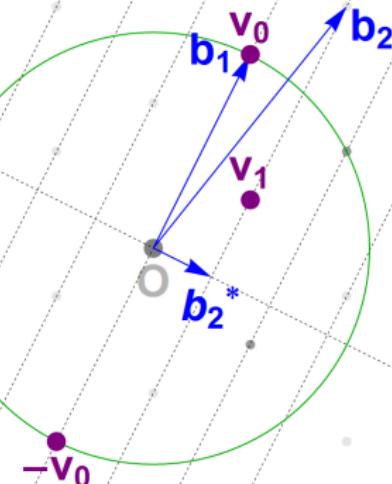
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP_1



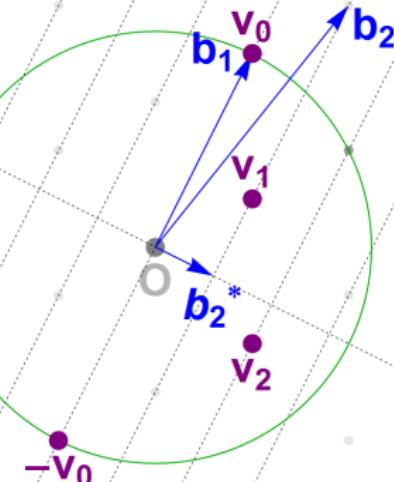
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP_1



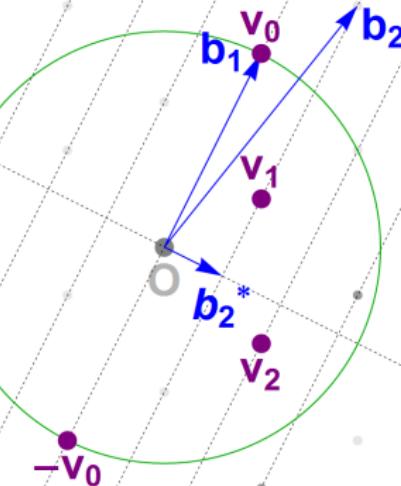
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP_1



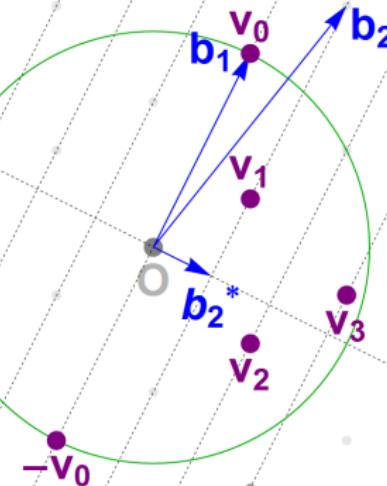
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP₁



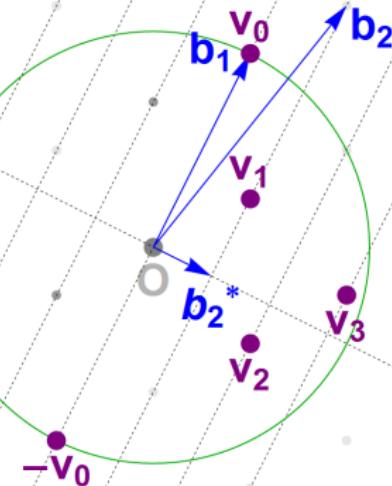
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP_1



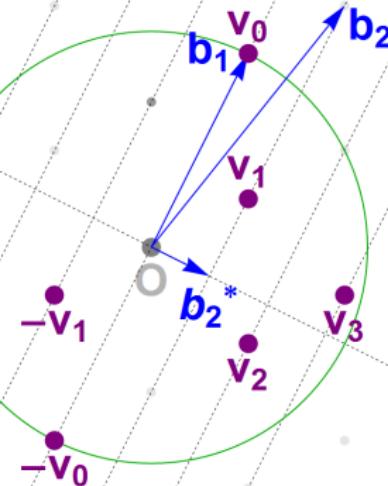
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP_1



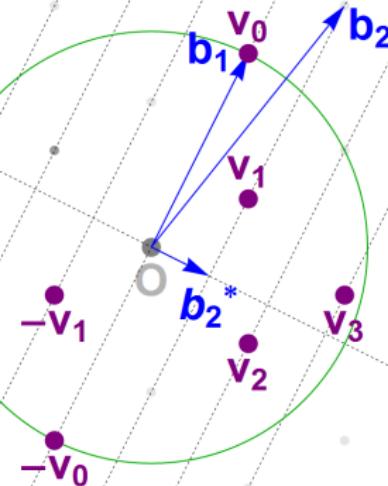
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP₁



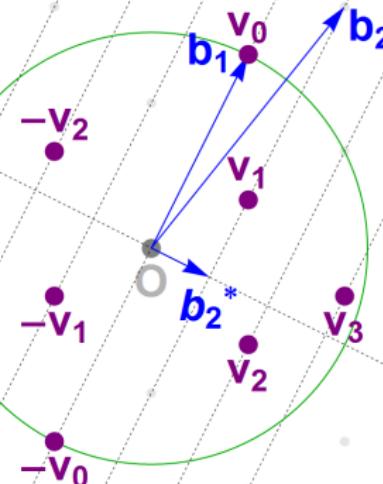
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP_1



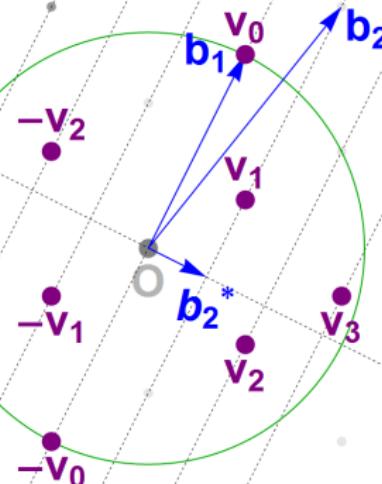
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP₁



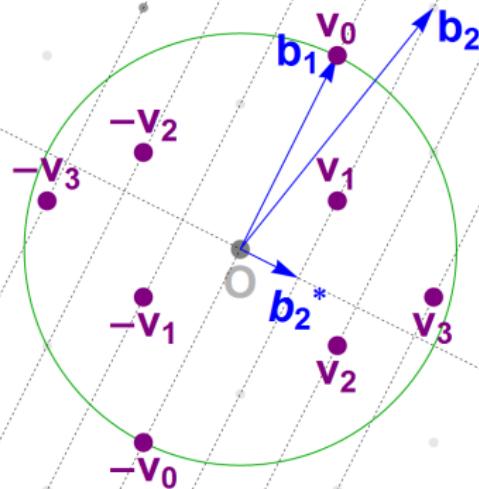
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP₁



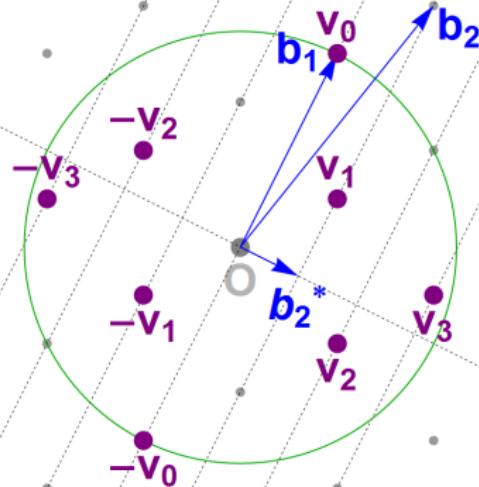
Enumeration

1-2. Guess the coefficient of b_2 and solve CVP₁



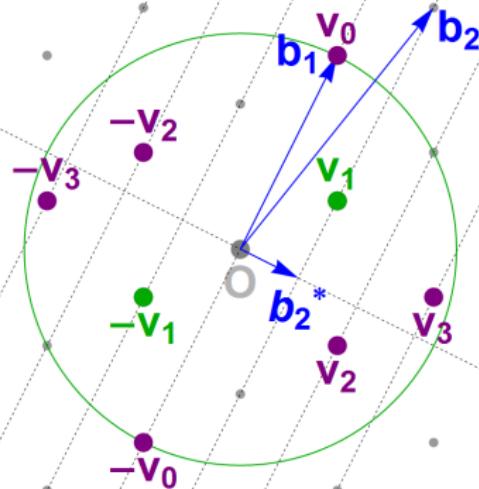
Enumeration

3. Find a shortest vector among all of them



Enumeration

3. Find a shortest vector among all of them



Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

Procedure:

1. Guess the n th coordinate (coefficient of basis vector b_n)
2. Find a shortest vector, given the n th coordinate
3. Search for a shortest vector among all of these vectors

Recursive: Reduces SVP_n (CVP_n) to several instances of CVP_{n-1}

Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

Procedure:

1. Guess the n th coordinate (coefficient of basis vector b_n)
2. Find a shortest vector, given the n th coordinate
3. Search for a shortest vector among all of these vectors

Recursive: Reduces SVP_n (CVP_n) to several instances of CVP_{n-1}

Complexity?

Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

Procedure:

1. Guess the n th coordinate (coefficient of basis vector b_n)
2. Find a shortest vector, given the n th coordinate
3. Search for a shortest vector among all of these vectors

Recursive: Reduces SVP_n (CVP_n) to several instances of CVP_{n-1}

Complexity?

- Space: (polynomial)

Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

Procedure:

1. Guess the n th coordinate (coefficient of basis vector b_n)
2. Find a shortest vector, given the n th coordinate
3. Search for a shortest vector among all of these vectors

Recursive: Reduces SVP_n (CVP_n) to several instances of CVP_{n-1}

Complexity?

- Space: (polynomial)
- Classical Time: $2^{O(n \log n)}$ [Kan83]

Enumeration

Invented in the early '80s [Poh81, Kan83, FP85]

Procedure:

1. Guess the n th coordinate (coefficient of basis vector b_n)
2. Find a shortest vector, given the n th coordinate
3. Search for a shortest vector among all of these vectors

Recursive: Reduces SVP_n (CVP_n) to several instances of CVP_{n-1}

Complexity?

- Space: (polynomial)
- Classical Time: $2^{O(n \log n)}$ [Kan83]
- Quantum Time: $2^{O(n \log n)}$?

Overview

Theoretical results (large n)

Table: Complexities of SVP algorithms in logarithmic leading order terms, ordered by their time complexities (descending).

Algorithm	Classical		Quantum	
	Time	Space	Time	Space
Enum. [Kan83]	$O(n \log n)$	$O(\log n)$	$O(n \log n)$	$O(\log n)$
Sieving [PS09]	$2.65n$	$1.33n$	$2.65n$	$1.33n$
Saturation [PS09]	$2.47n$	$1.24n$	$2.47n$	$1.24n$
Voronoi cell [MV10]	$2.00n$	$1.00n$	$2.00n$	$1.00n$

Overview

Theoretical results (large n)

Table: Complexities of SVP algorithms in logarithmic leading order terms, ordered by their time complexities (descending).

Algorithm	Classical		Quantum	
	Time	Space	Time	Space
Enum. [Kan83]	$O(n \log n)$	$O(\log n)$	$O(n \log n)$	$O(\log n)$
Sieving [PS09]	$2.65n$	$1.33n$	$2.65n$	$1.33n$
Saturation [LMP13]	$2.47n$	$1.24n$	$1.80n$	$1.29n$
Voronoi cell [MV10]	$2.00n$	$1.00n$	$2.00n$	$1.00n$

Overview

Theoretical results (large n)

Table: Complexities of SVP algorithms in logarithmic leading order terms, ordered by their time complexities (descending).

Algorithm	Classical		Quantum	
	Time	Space	Time	Space
Enum. [Kan83]	$O(n \log n)$	$O(\log n)$	$O(n \log n)$	$O(\log n)$
Sieving [PS09]	$2.65n$	$1.33n$	$2.65n$	$1.33n$
Voronoi cell [MV10]	$2.00n$	$1.00n$	$2.00n$	$1.00n$
Saturation [LMP13]	$2.47n$	$1.24n$	$1.80n$	$1.29n$

Overview

Heuristic/Experimental results ($n \approx 100$)

Table: Complexities of SVP algorithms in logarithmic leading order terms, ordered by their time complexities (descending).

Algorithm	Classical		Quantum	
	Time	Space	Time	Space
Voronoi cell [MV10]	$2.00n$	$1.00n$	$2.00n$	$1.00n$
Sieving [NV08]	$0.42n$	$0.21n$	$0.42n$	$0.21n$
Saturation [MV09]	$0.52n$	$0.21n$	$0.52n$	$0.21n$
Enum. [GNR10]	$O(n \log n)$	$O(\log n)$	$O(n \log n)$	$O(\log n)$

Overview

Heuristic/Experimental results ($n \approx 100$)

Table: Complexities of SVP algorithms in logarithmic leading order terms, ordered by their time complexities (descending).

Algorithm	Classical		Quantum	
	Time	Space	Time	Space
Voronoi cell [MV10]	$2.00n$	$1.00n$	$2.00n$	$1.00n$
Sieving [LMP13]	$0.42n$	$0.21n$	$0.32n$	$0.21n$
Saturation [LMP13]	$0.52n$	$0.21n$	$0.39n$	$0.21n$
Enum. [GNR10]	$O(n \log n)$	$O(\log n)$	$O(n \log n)$	$O(\log n)$

Conclusion

Results

- Faster sieving algorithms (exponent: -25%)
- Faster saturation algorithms (exponent: $\approx -25\%$)

Open problems

- Improve enumeration algorithms?
- Improve Voronoi cell algorithm?
- Use other quantum algorithms?
- Build a quantum computer?

Questions

