TU/e Technische Universiteit Eindhoven University of Technology

Sieving for shortest lattice vectors using fast search algorithms

Thijs Laarhoven

(joint work with Michele Mosca, Joop van de Pol, Benne de Weger)

mail@thijs.com
http://www.thijs.com/

Rutgers University, New Brunswick, USA
(January 13, 2015)

TU/e

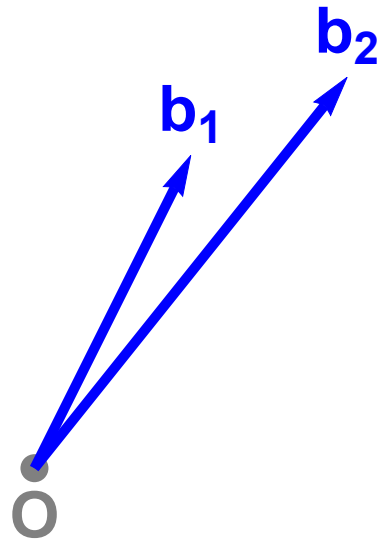
Lattices

What is a lattice?

TU/e

Lattices

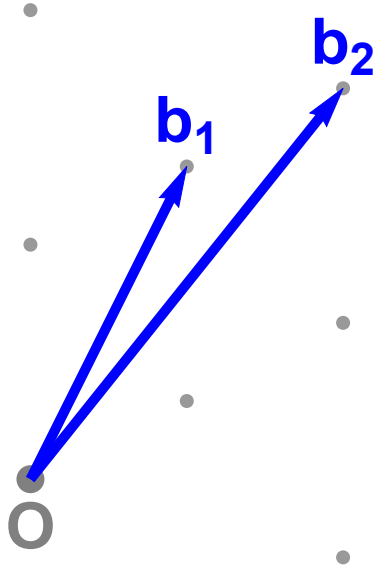
What is a lattice?



TU/e

Lattices

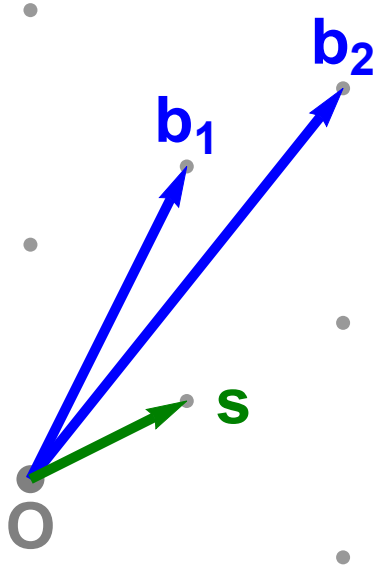
What is a lattice?



TU/e

Lattices

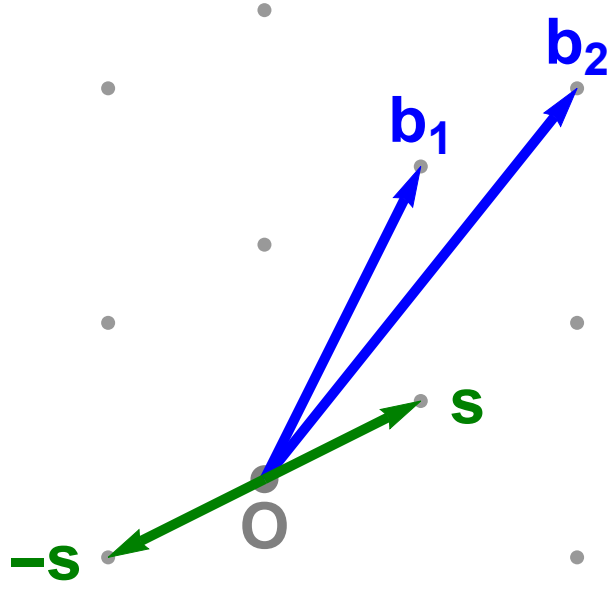
Shortest Vector Problem (SVP)



TU/e

Lattices

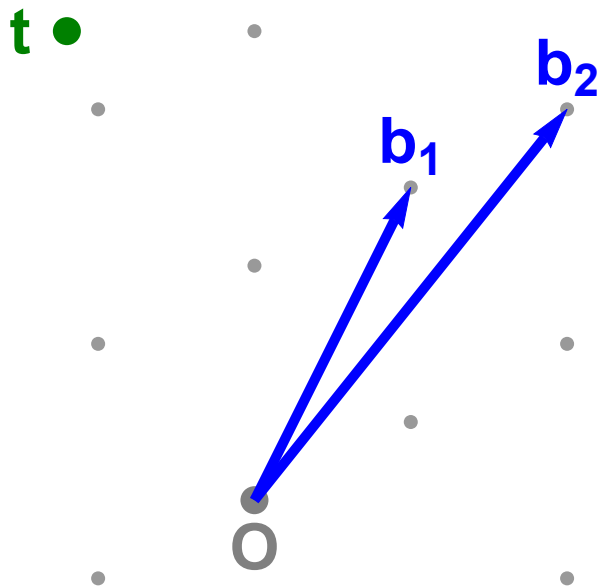
Shortest Vector Problem (SVP)



TU/e

Lattices

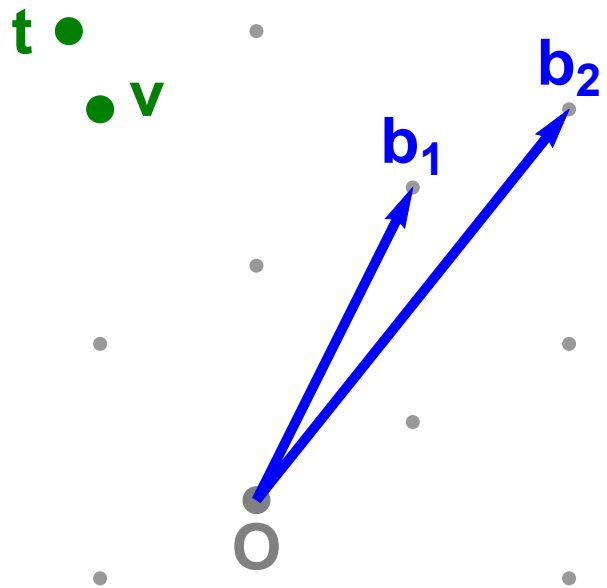
Closest Vector Problem (CVP)



TU/e

Lattices

Closest Vector Problem (CVP)



TU/e

Lattices

Applications

- "Constructive cryptography": Lattice-based cryptosystems
  - Based on hard lattice problems (SVP, CVP, LWE, SIS)
  - NTRU cryptosystem [HPS98, ...]
  - Fully Homomorphic Encryption [Gen09, ...]
  - Candidate for post-quantum cryptography

TU/e

Lattices

Applications

- "Constructive cryptography": Lattice-based cryptosystems
  - Based on hard lattice problems (SVP, CVP, LWE, SIS)
  - NTRU cryptosystem [HPS98, ...]
  - Fully Homomorphic Encryption [Gen09, ...]
  - Candidate for post-quantum cryptography
- "Destructive cryptography": Lattice cryptanalysis
  - Attack knapsack-based cryptosystems [Sha82, LO85]
  - Attack RSA with Coppersmith's method [Cop97]
  - Attack lattice-based cryptosystems [Ngu99, JJ00]

TU/e

Lattices

Applications

- "Constructive cryptography": Lattice-based cryptosystems
  - Based on hard lattice problems (SVP, CVP, LWE, SIS)
  - NTRU cryptosystem [HPS98, ...]
  - Fully Homomorphic Encryption [Gen09, ...]
  - Candidate for post-quantum cryptography
- "Destructive cryptography": Lattice cryptanalysis
  - Attack knapsack-based cryptosystems [Sha82, LO85]
  - Attack RSA with Coppersmith's method [Cop97]
  - Attack lattice-based cryptosystems [Ngu99, JJ00]

How hard are hard lattice problems such as SVP?

TU/e

Nguyen-Vidick sieve

TU/e

Nguyen-Vidick sieve

1. Sample a list L of random lattice vectors

TU/e

Nguyen-Vidick sieve

1. Sample a list L of random lattice vectors

TU/e

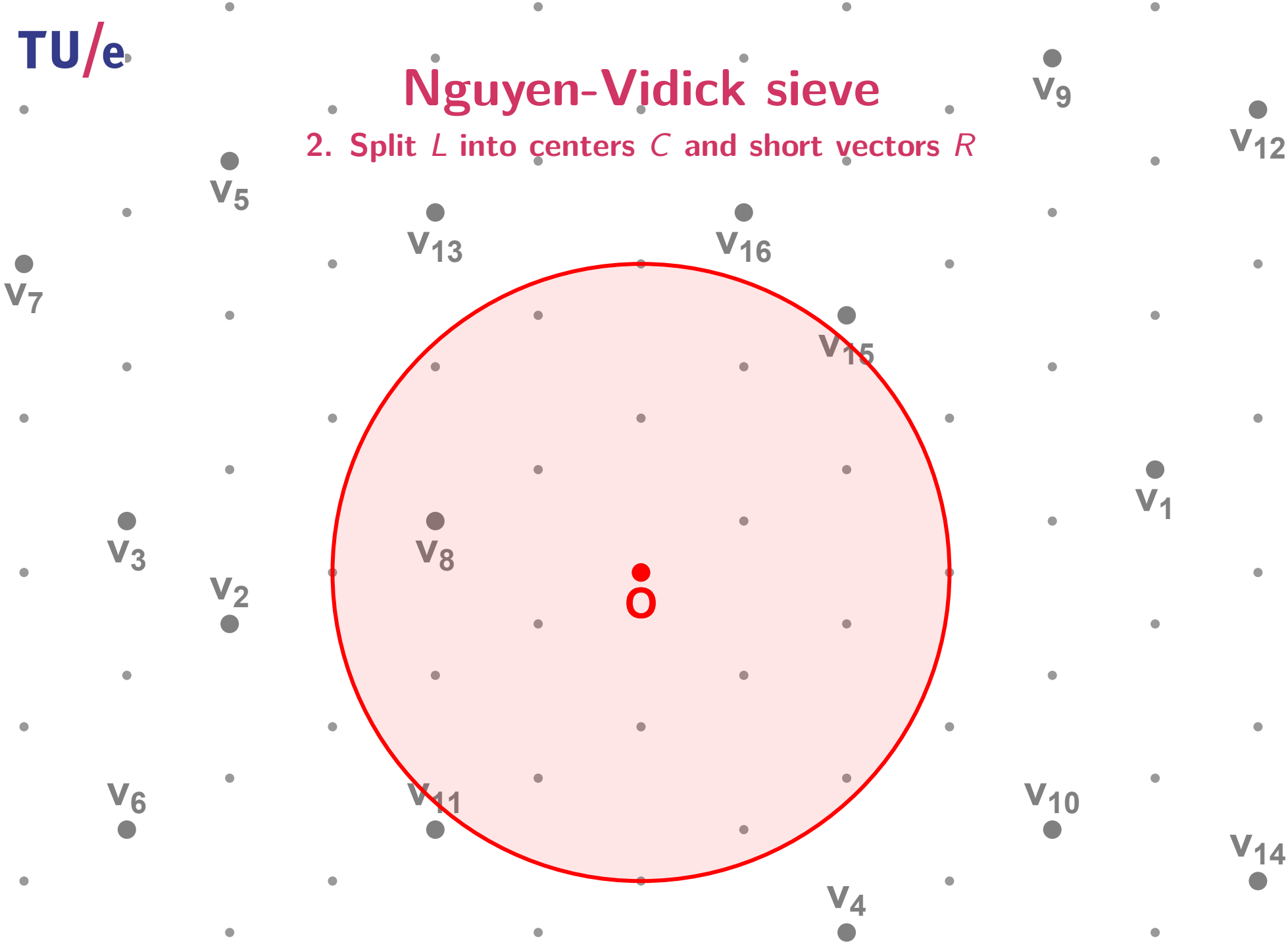
Nguyen-Vidick sieve

2. Split L into centers C and short vectors R

TU/e

Nguyen-Vidick sieve

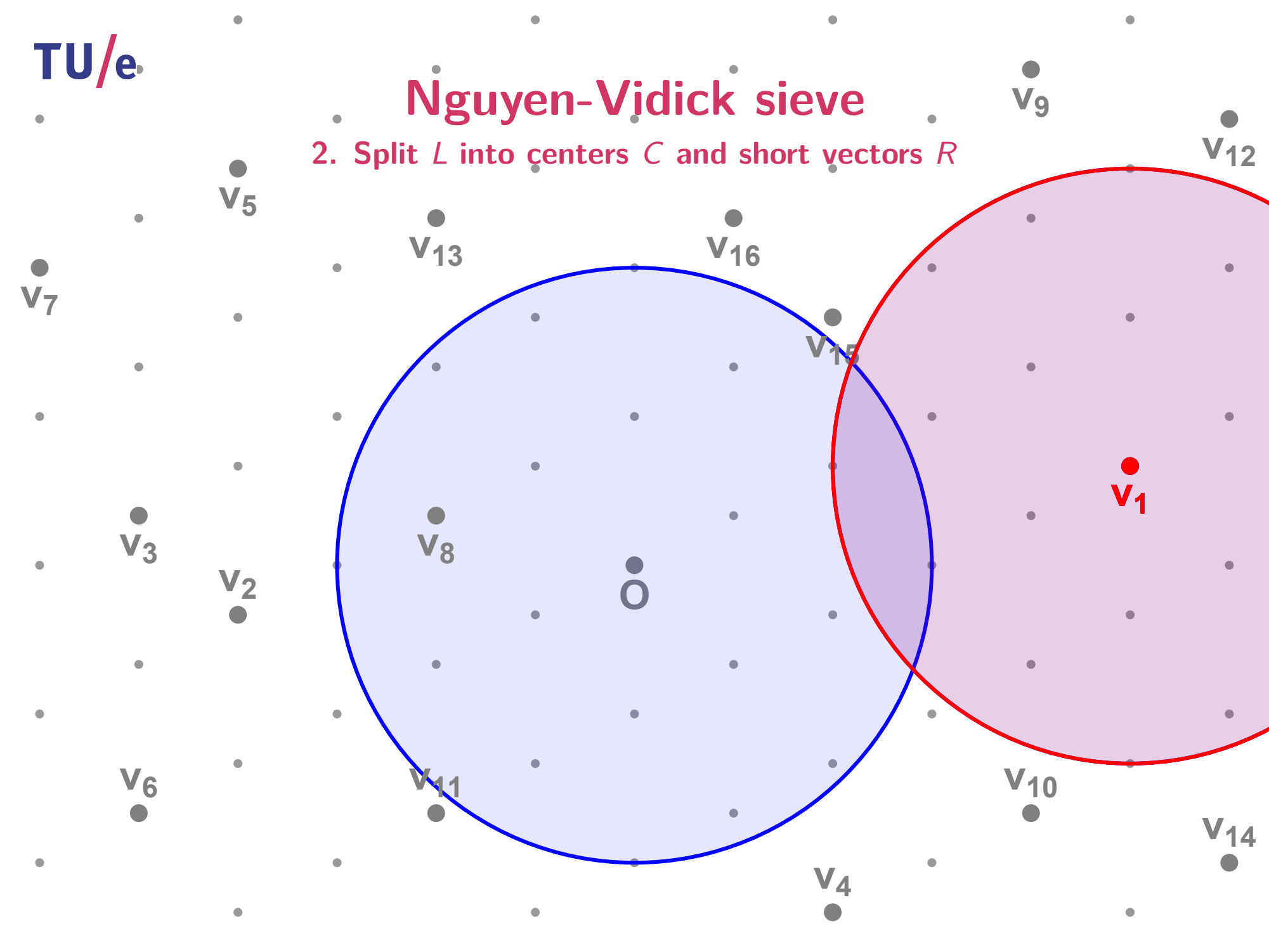
2. Split L into centers C and short vectors R



TU/e

Nguyen-Vidick sieve

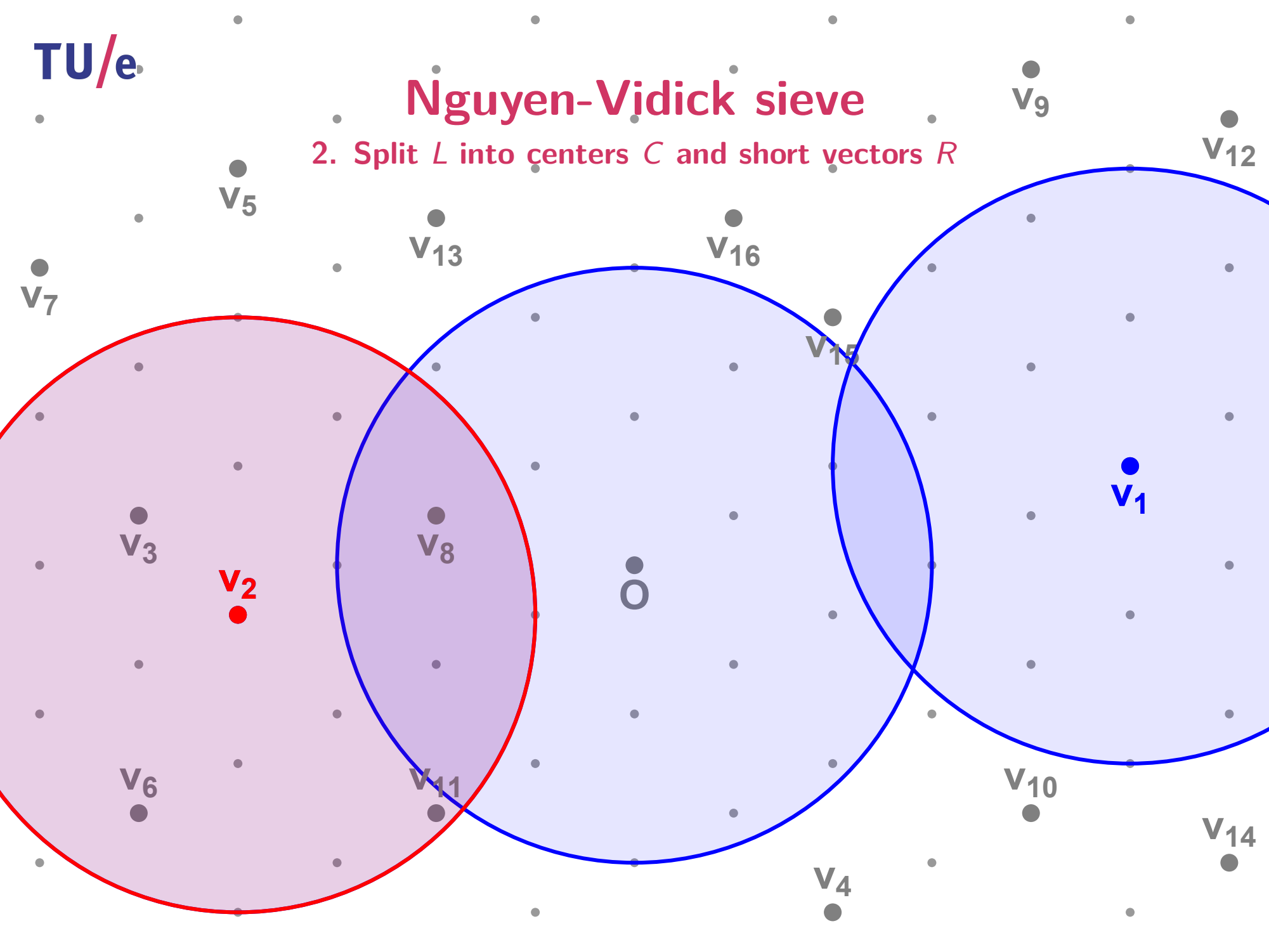
2. Split L into centers C and short vectors R



TU/e

Nguyen-Vidick sieve

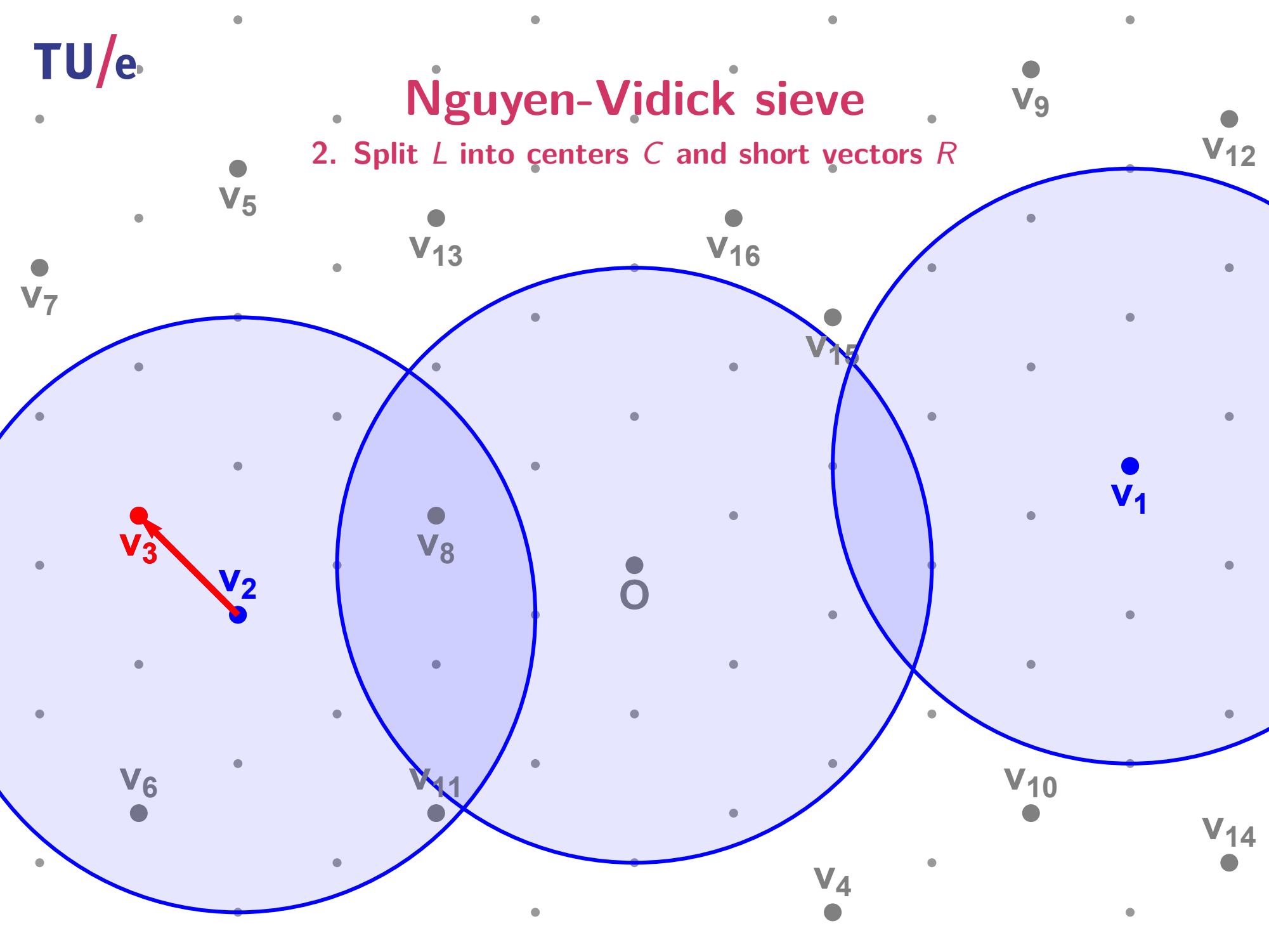
2. Split L into centers C and short vectors R



TU/e

Nguyen-Vidick sieve

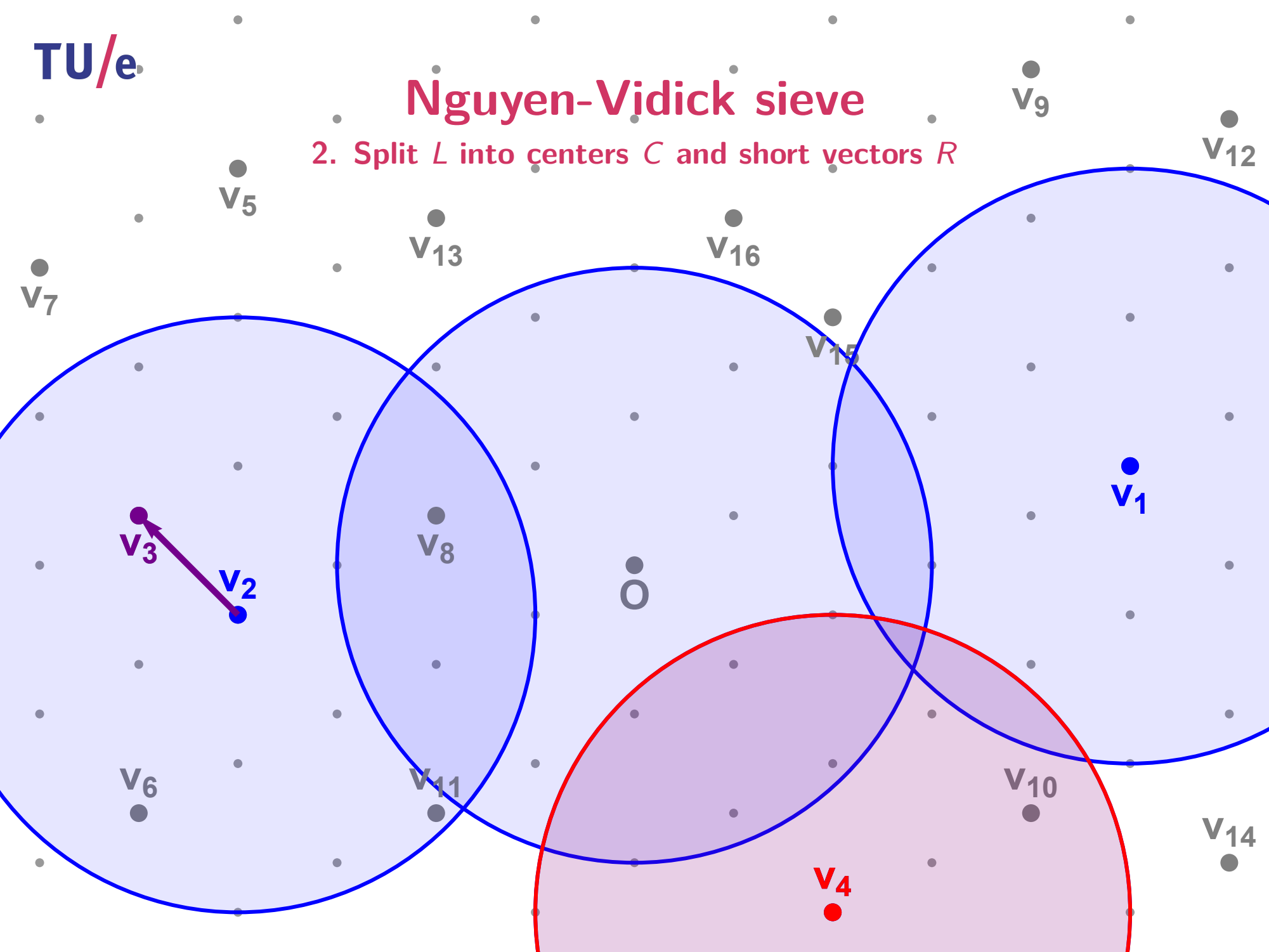
2. Split L into centers C and short vectors R



TU/e

Nguyen-Vidick sieve

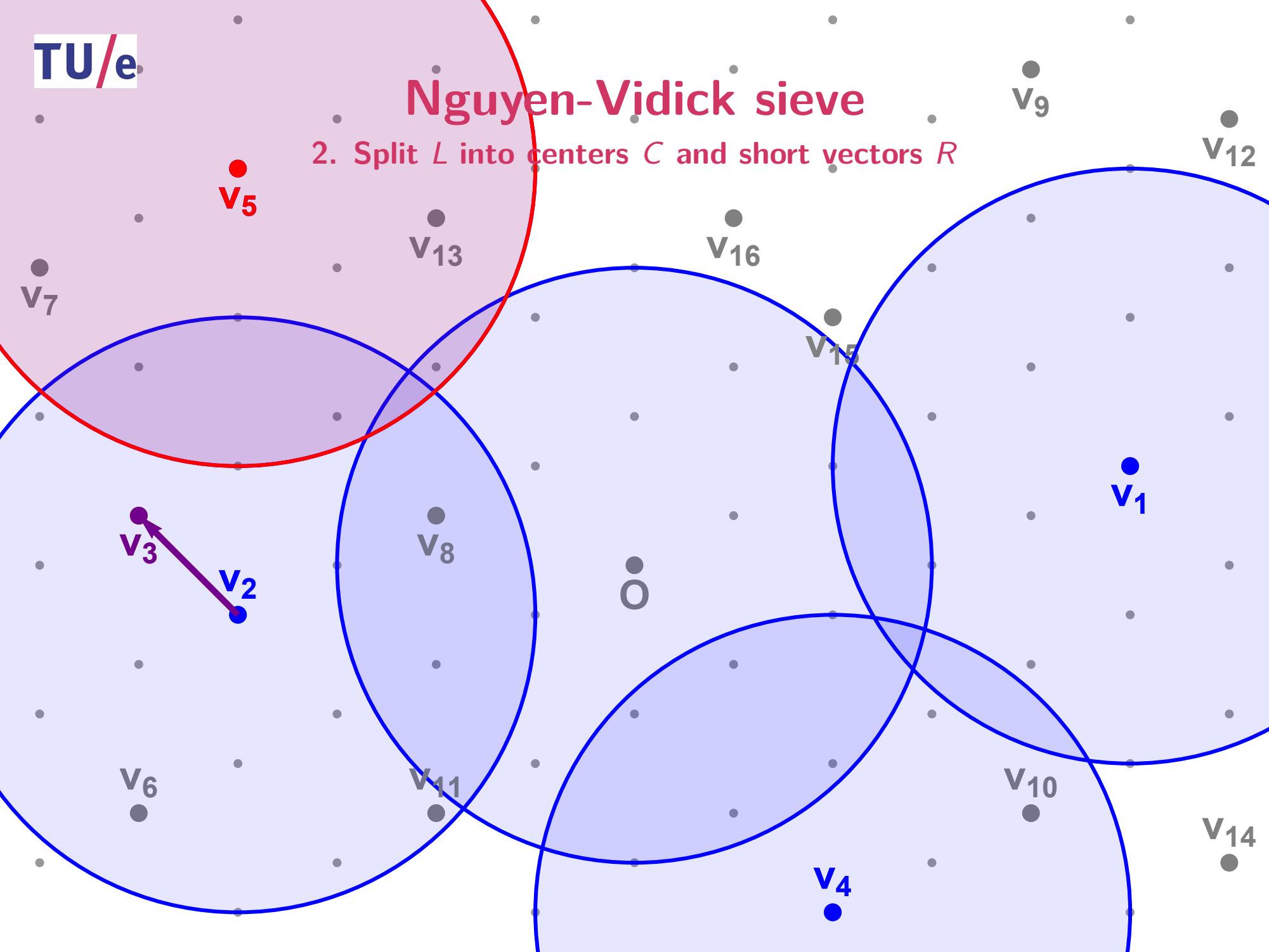
2. Split L into centers C and short vectors R



TU/e

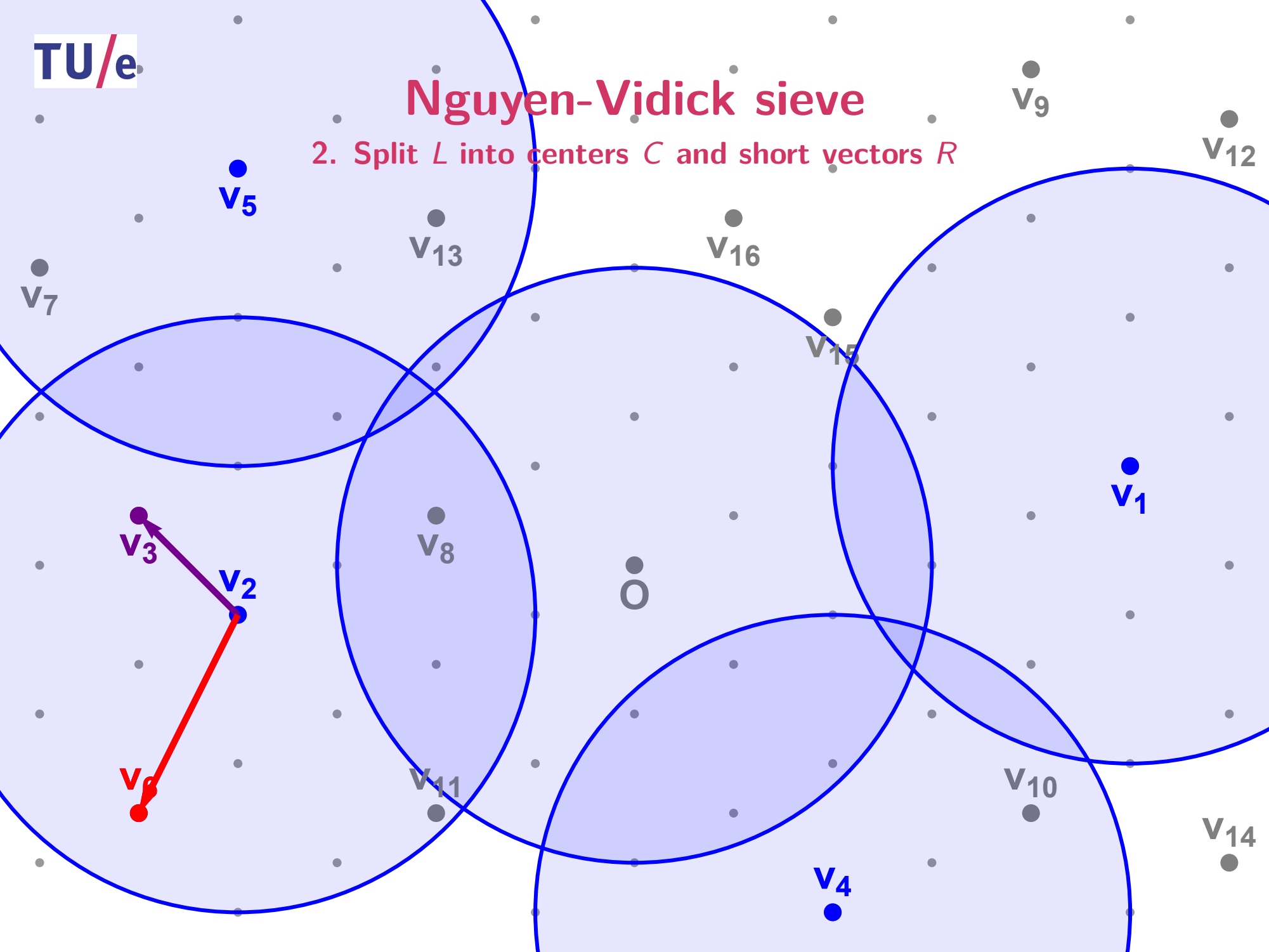
Nguyen-Vidick sieve

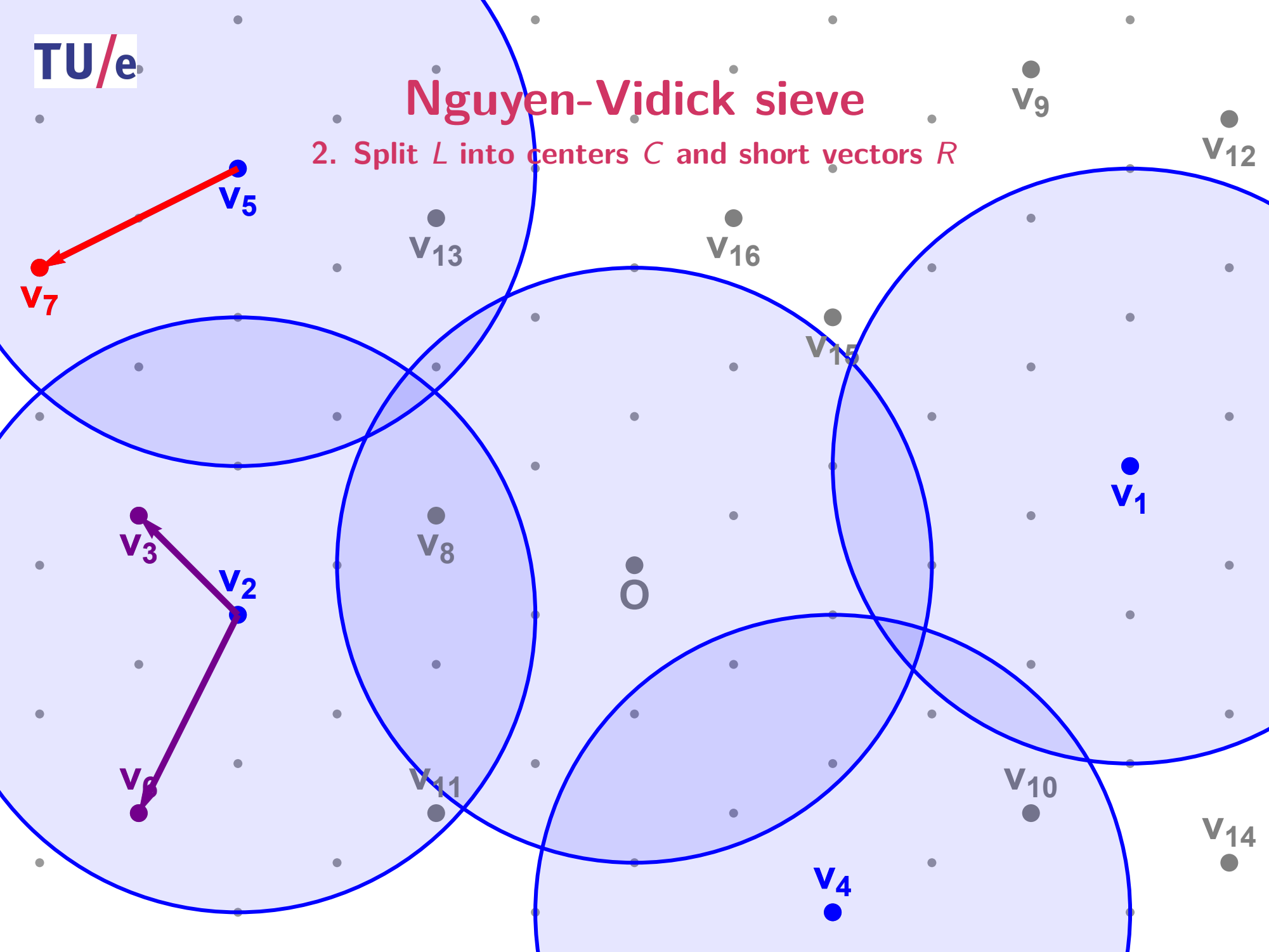
2. Split L into centers C and short vectors R



Nguyen-Vidick sieve

2. Split L into centers C and short vectors R

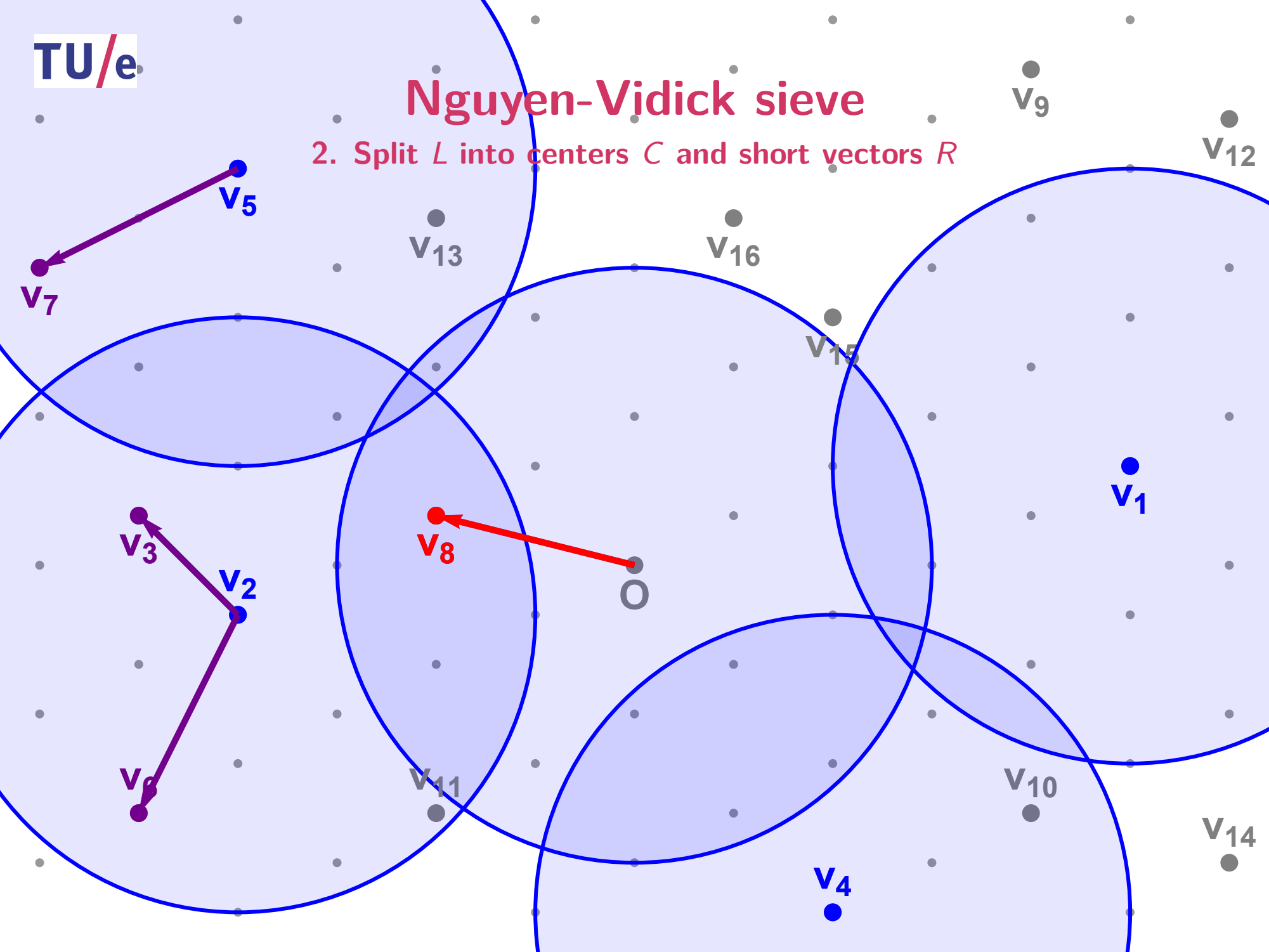


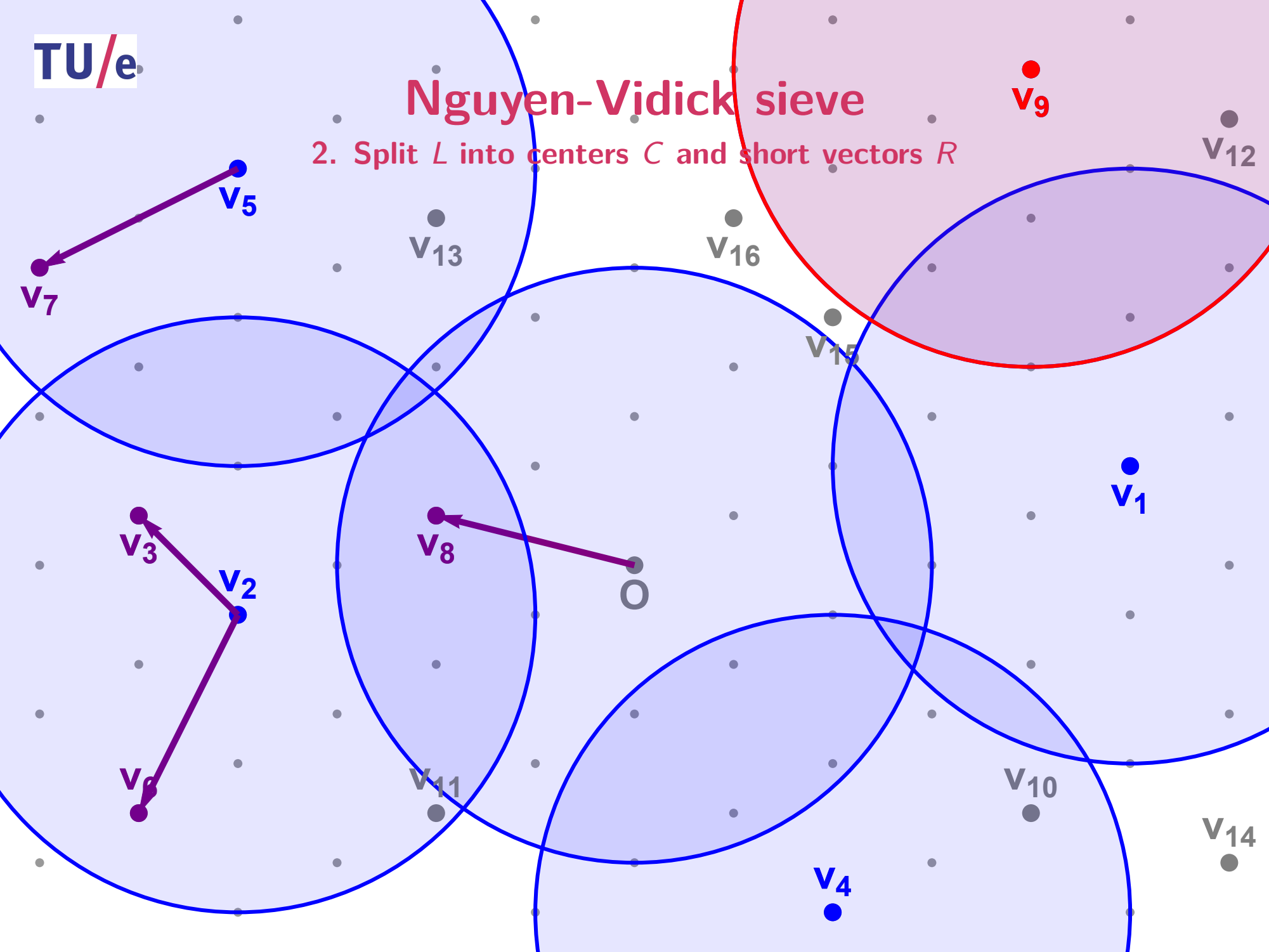

TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_5
v_7
v_{13}
v_{16}
v_9
v_{12}
v_{15}
v_1
v_3
v_2
v_8
O
v_6
v_{11}
v_{10}
v_{14}
v_4

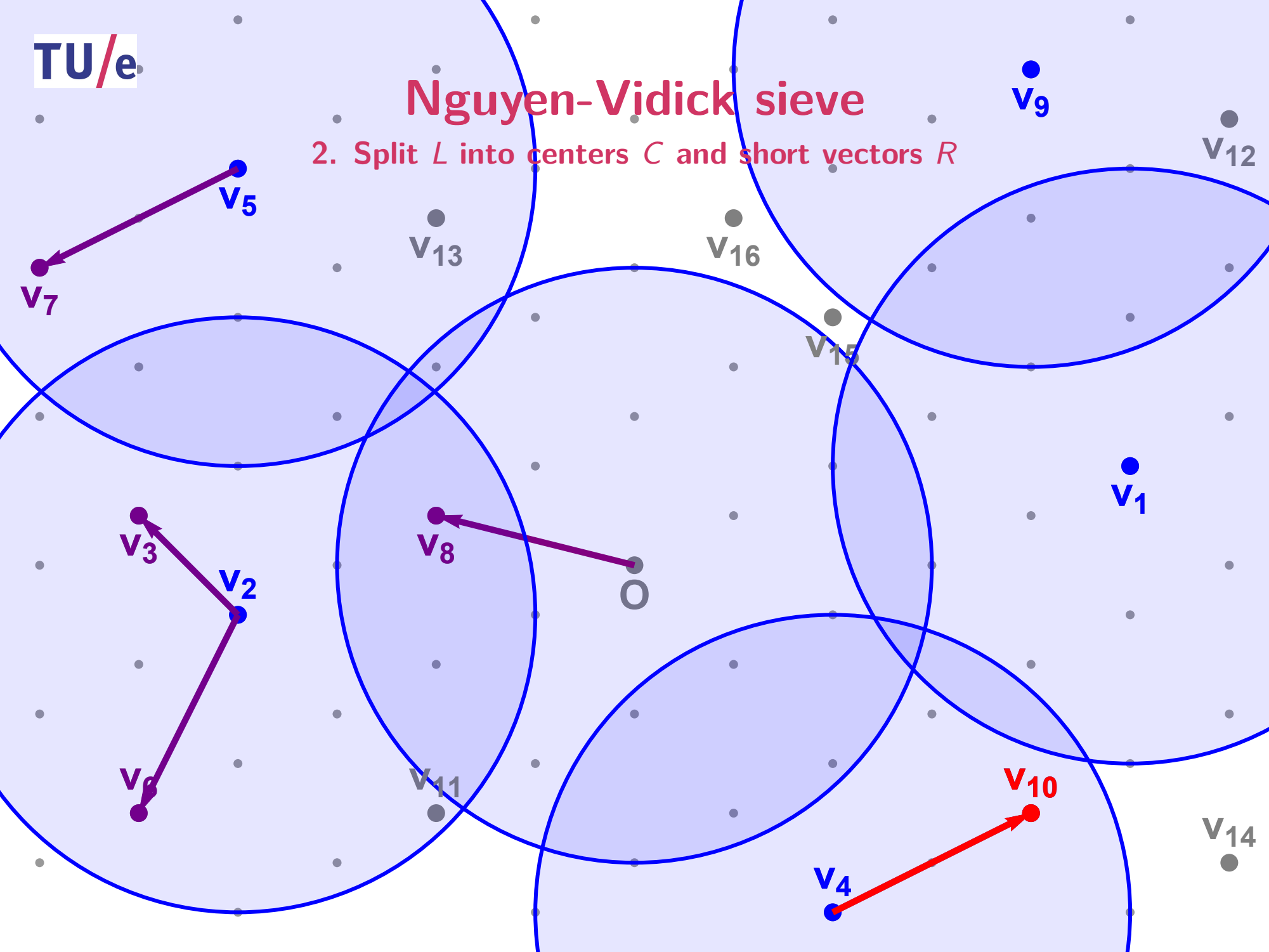
TU/e

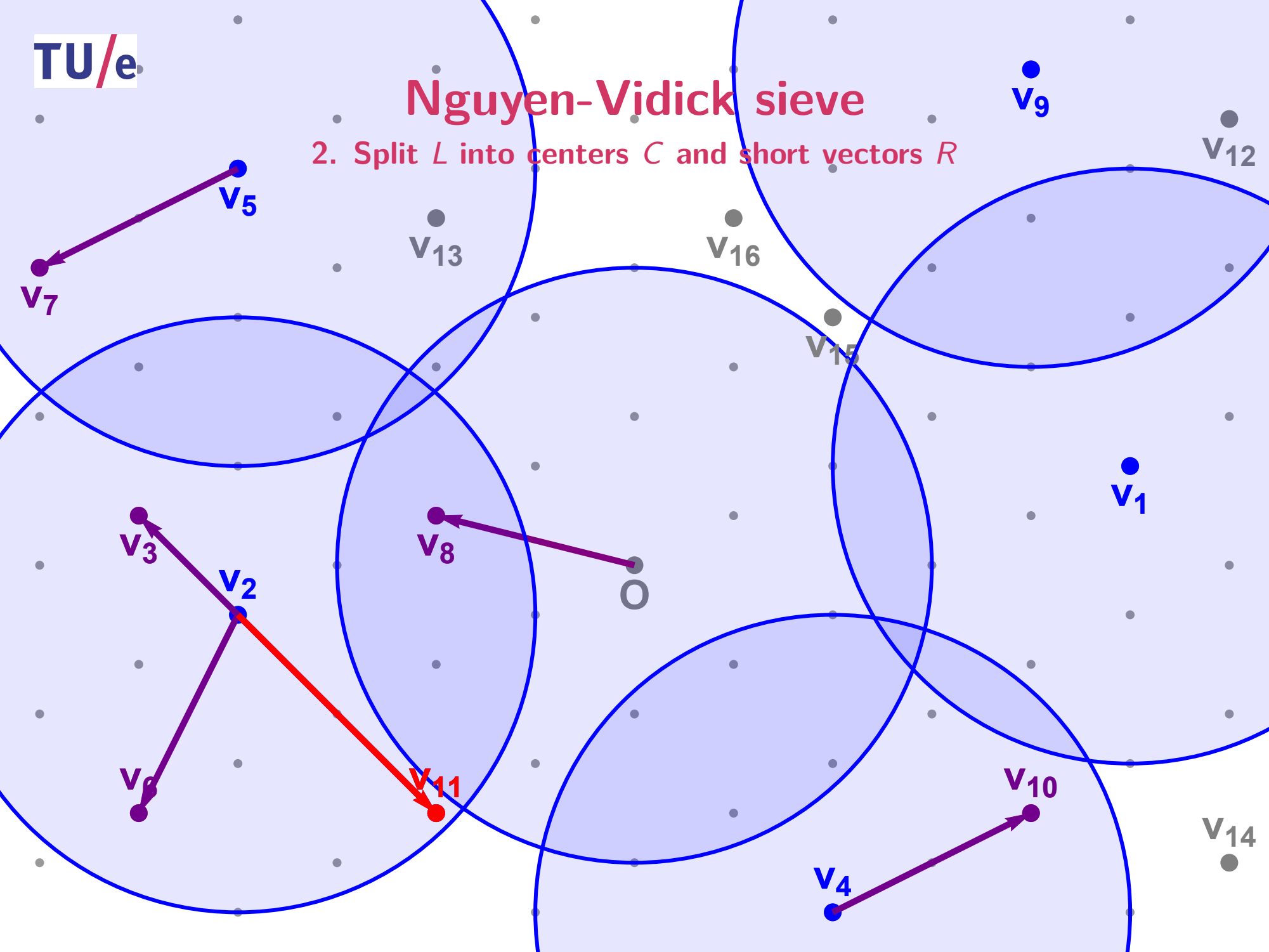
Nguyen-Vidick sieve

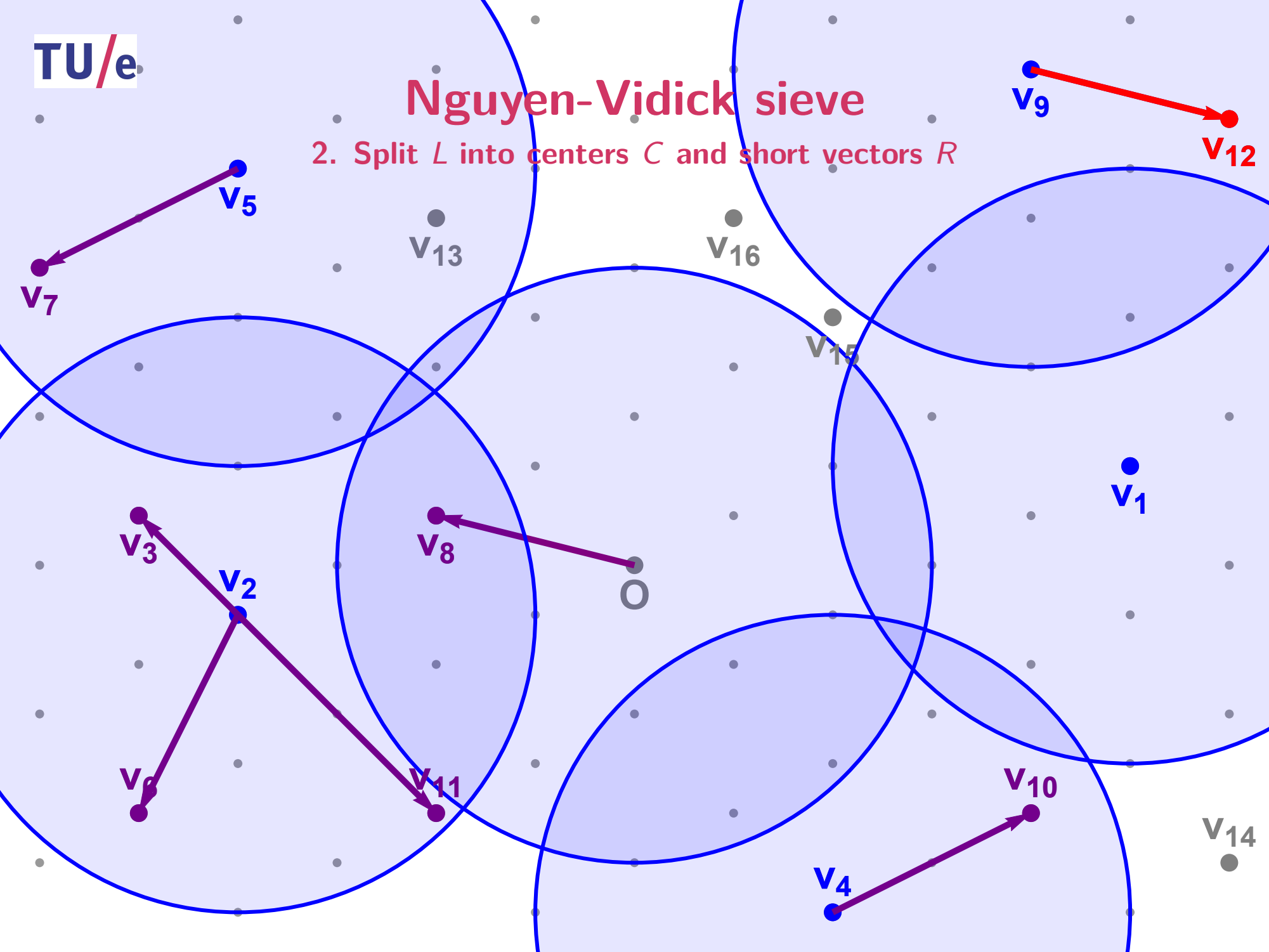
2. Split L into centers C and short vectors R

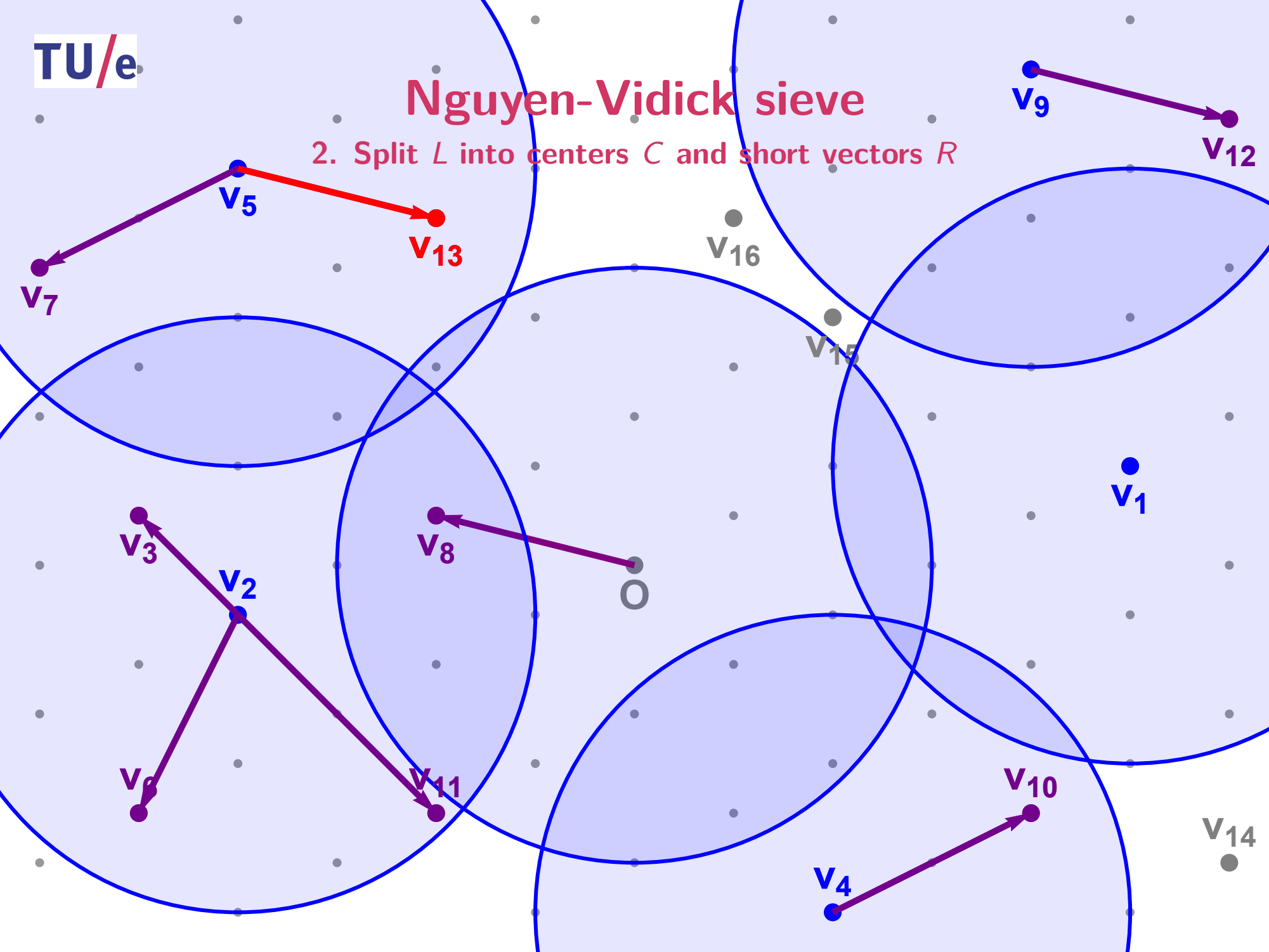



TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4


TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_{11}
v_{10}
v_{14}
v_4


TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_{11}
v_{10}
v_{14}
v_4

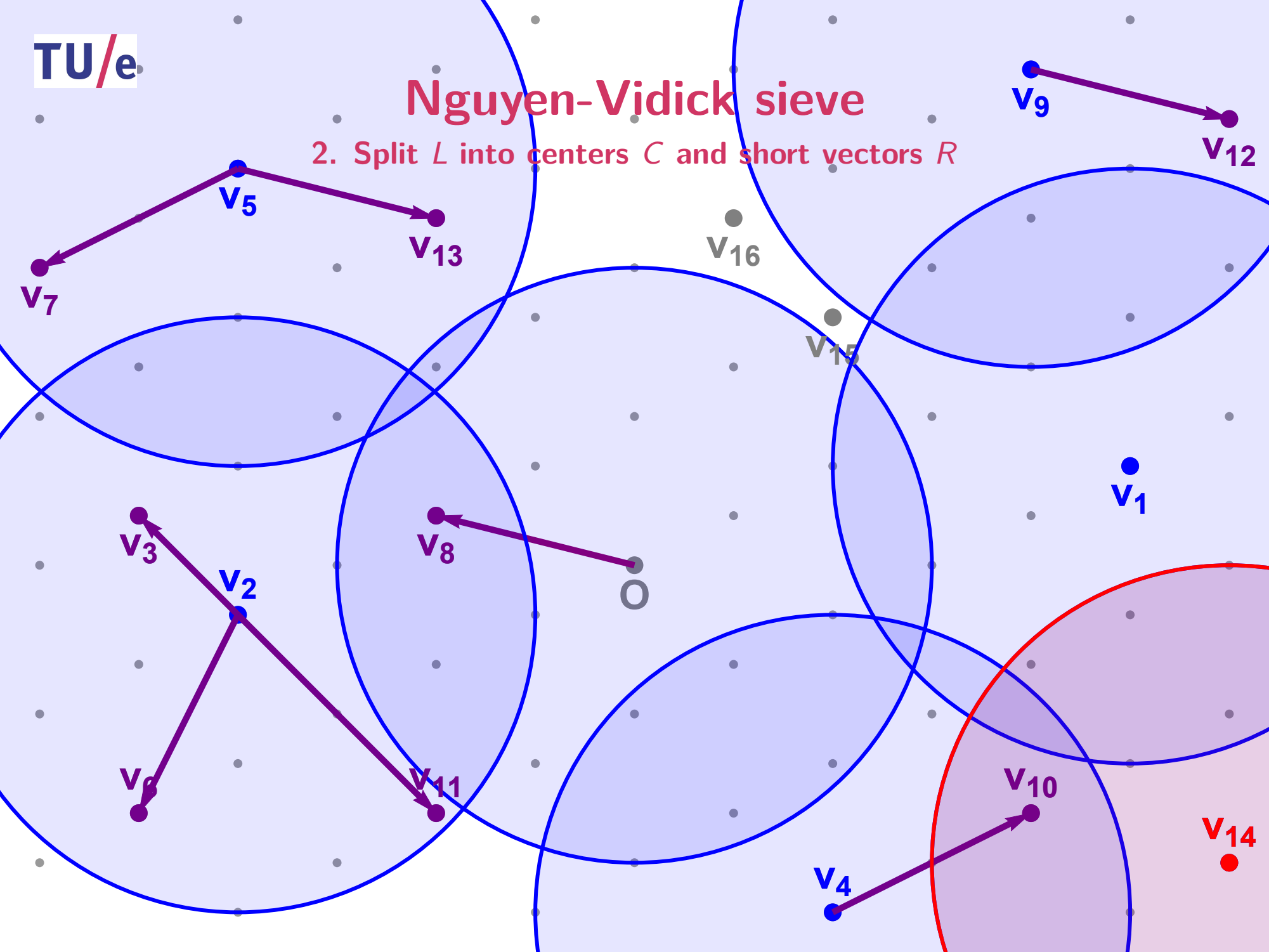

TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4

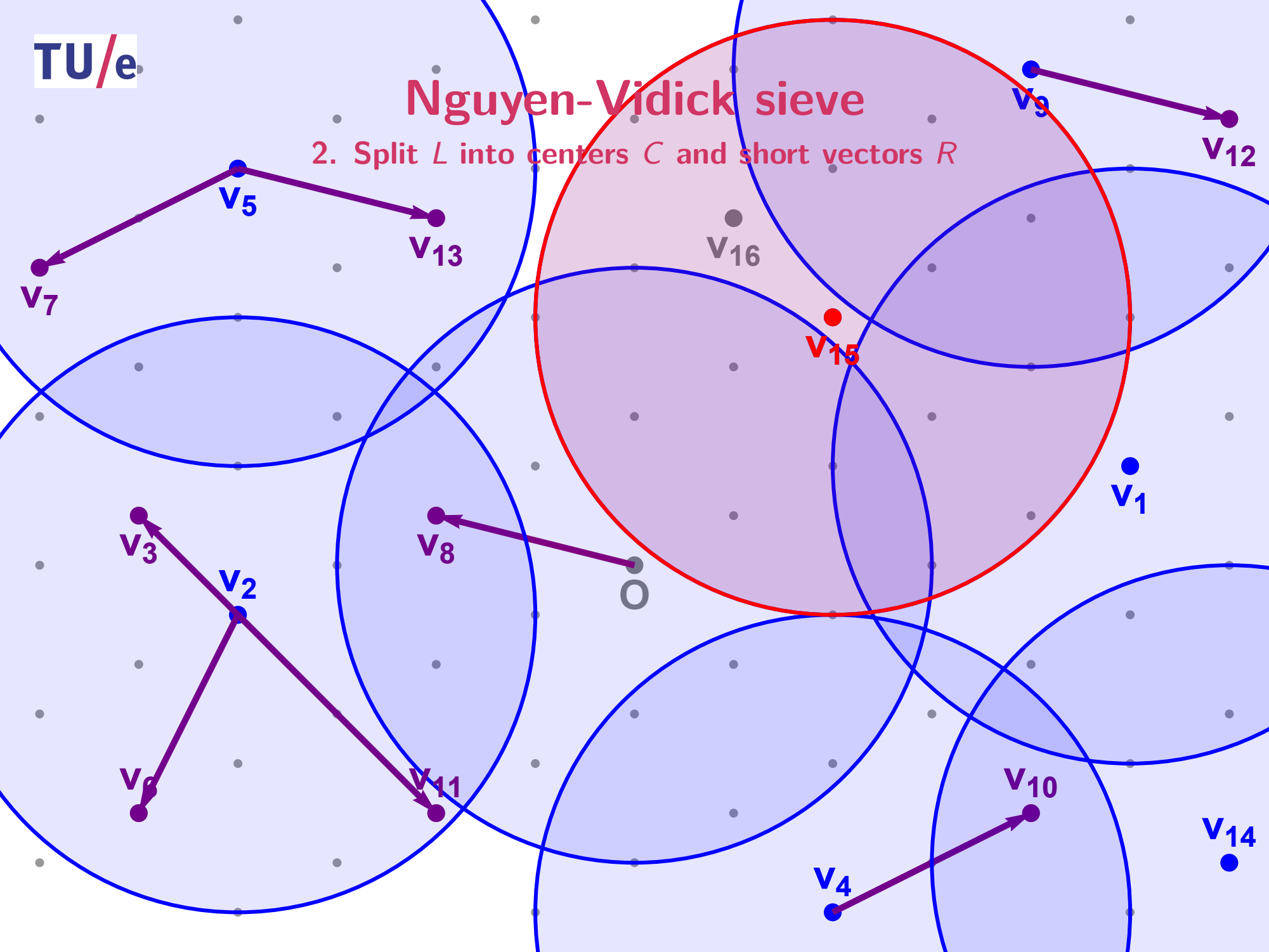

TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4

TU/e

Nguyen-Vidick sieve

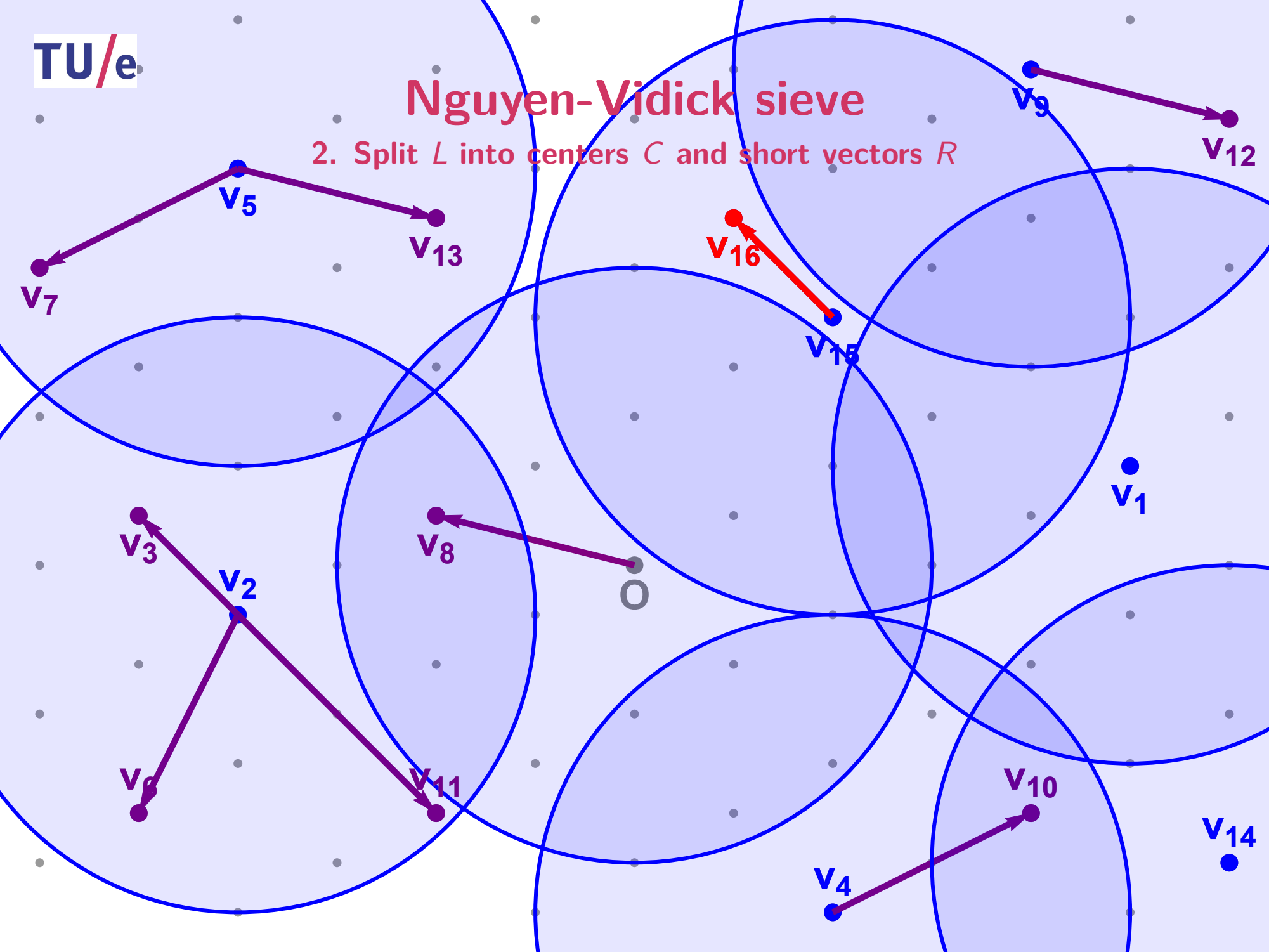
2. Split L into centers C and short vectors R




TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_5
v_{13}
v_7
v_{16}
v_9
v_{12}
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4

Nguyen-Vidick sieve

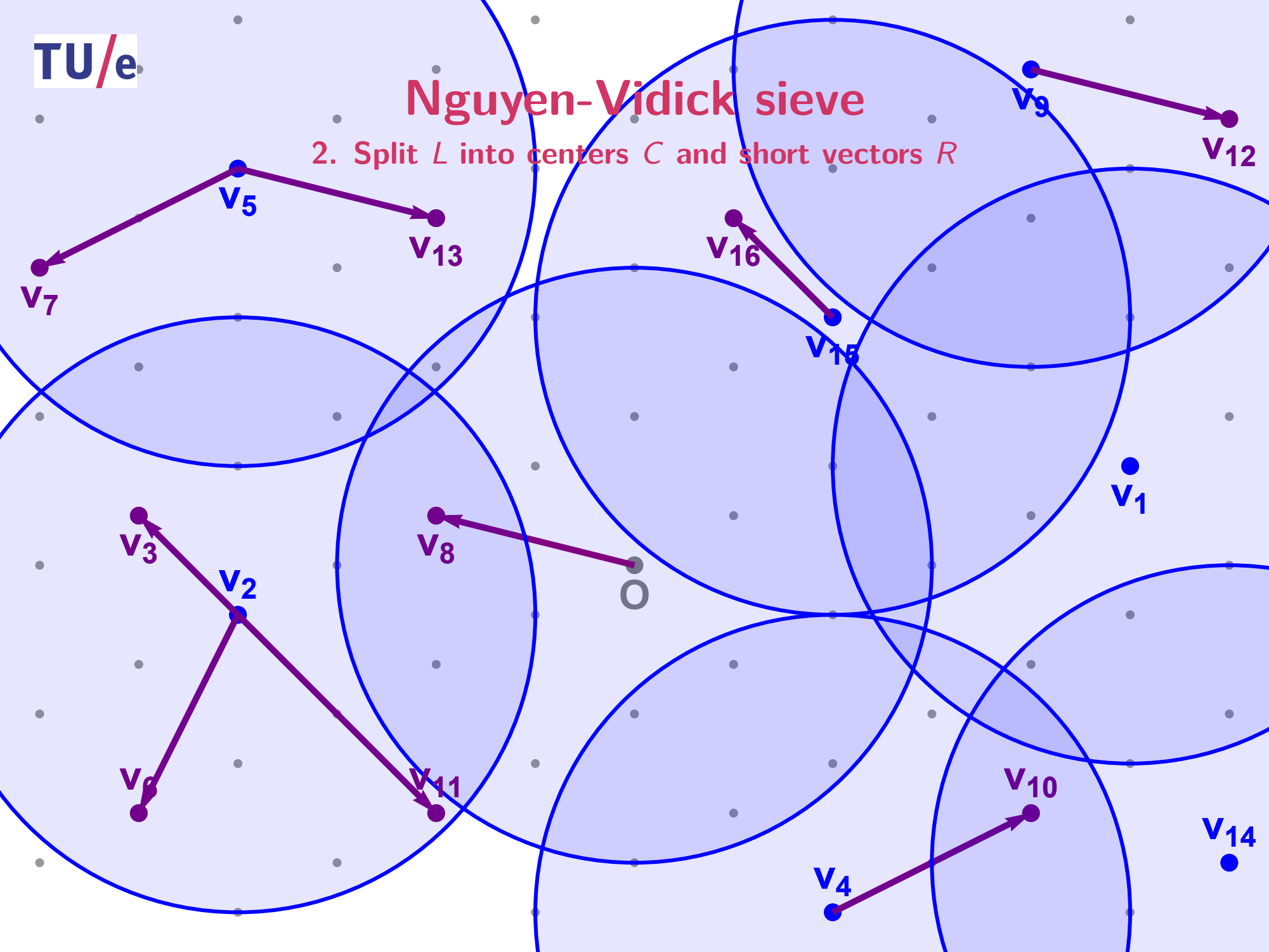
2. Split L into centers C and short vectors R



TU/e

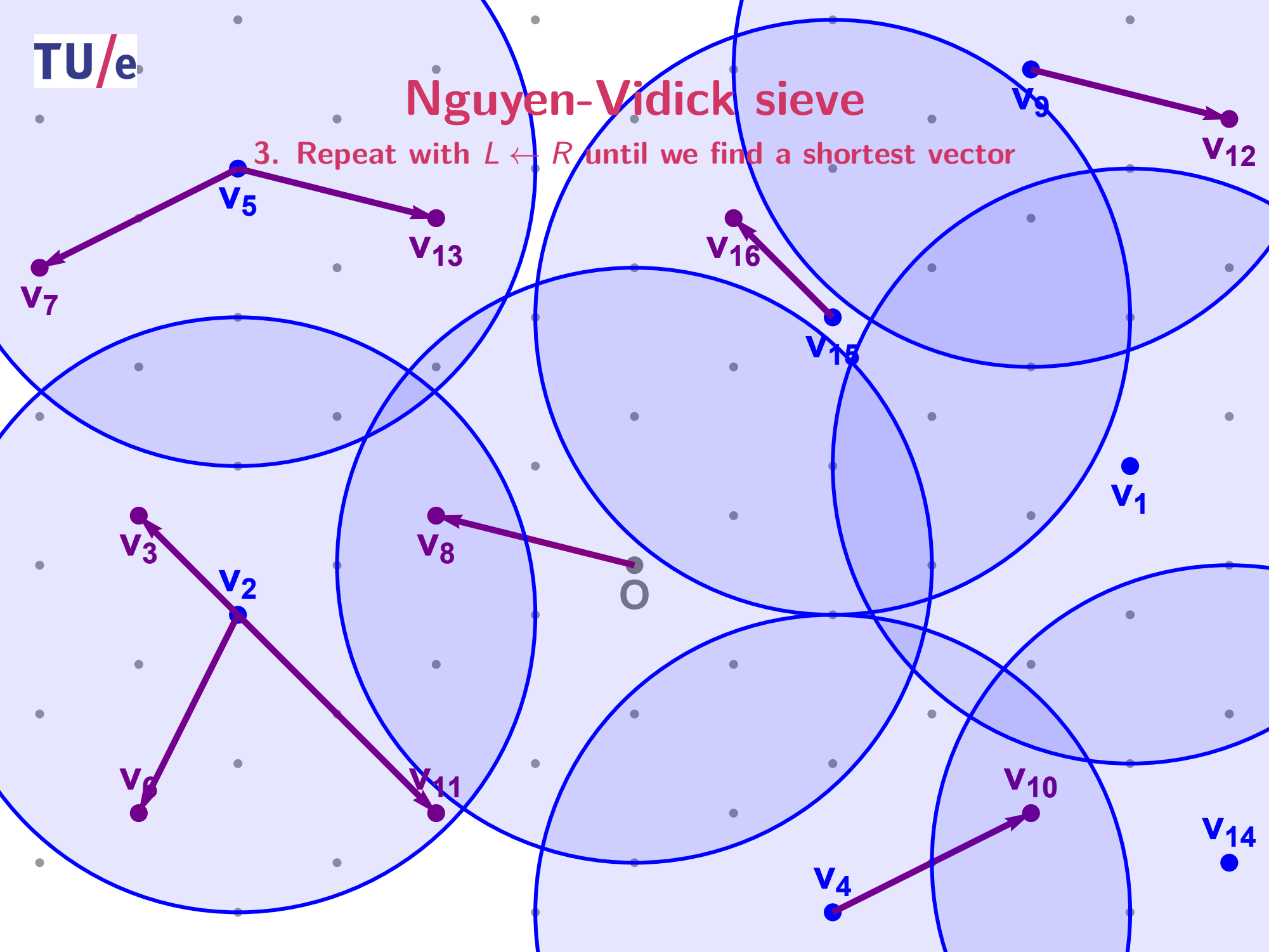
Nguyen-Vidick sieve

2. Split L into centers C and short vectors R



Nguyen-Vidick sieve

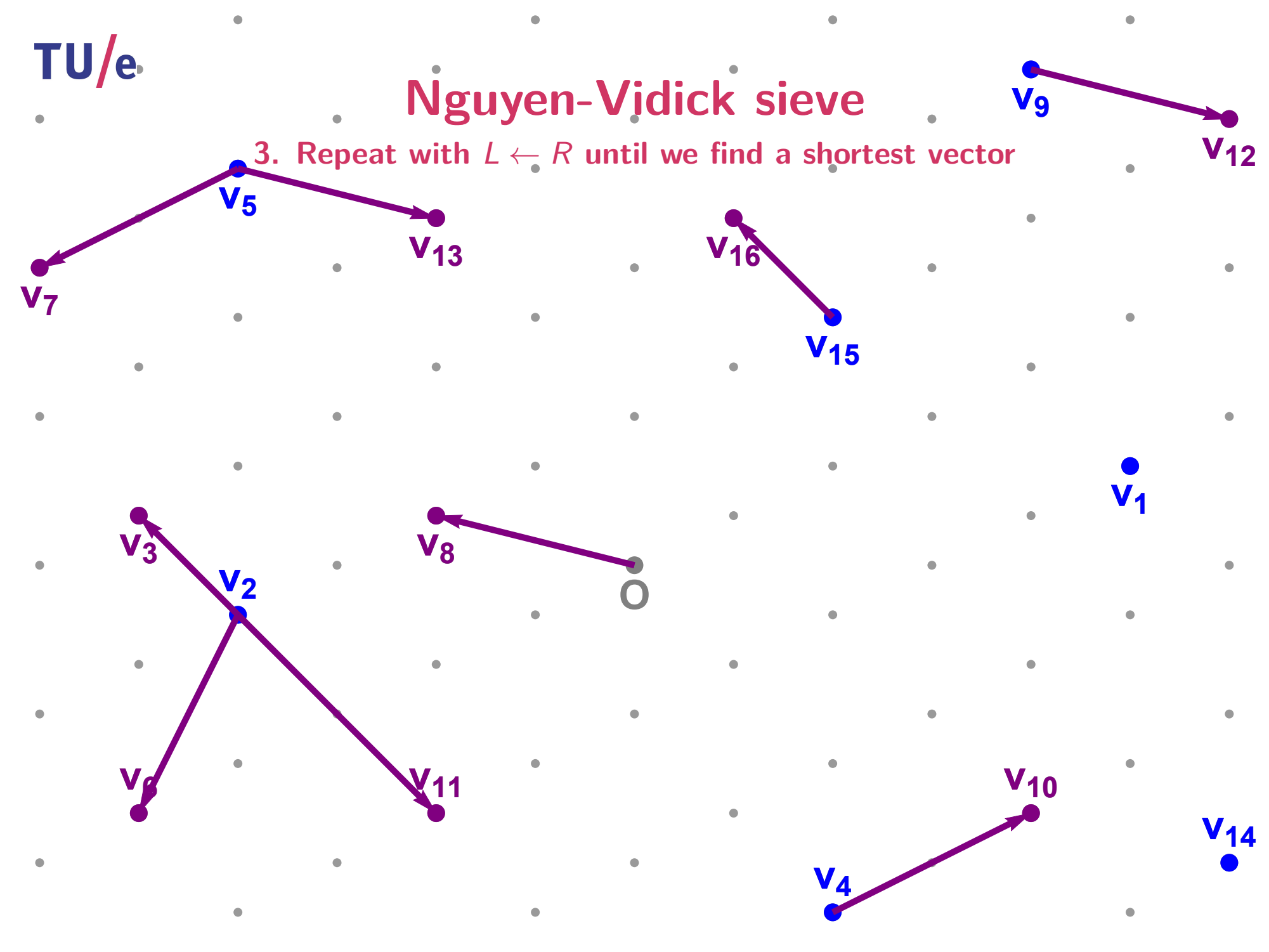
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve

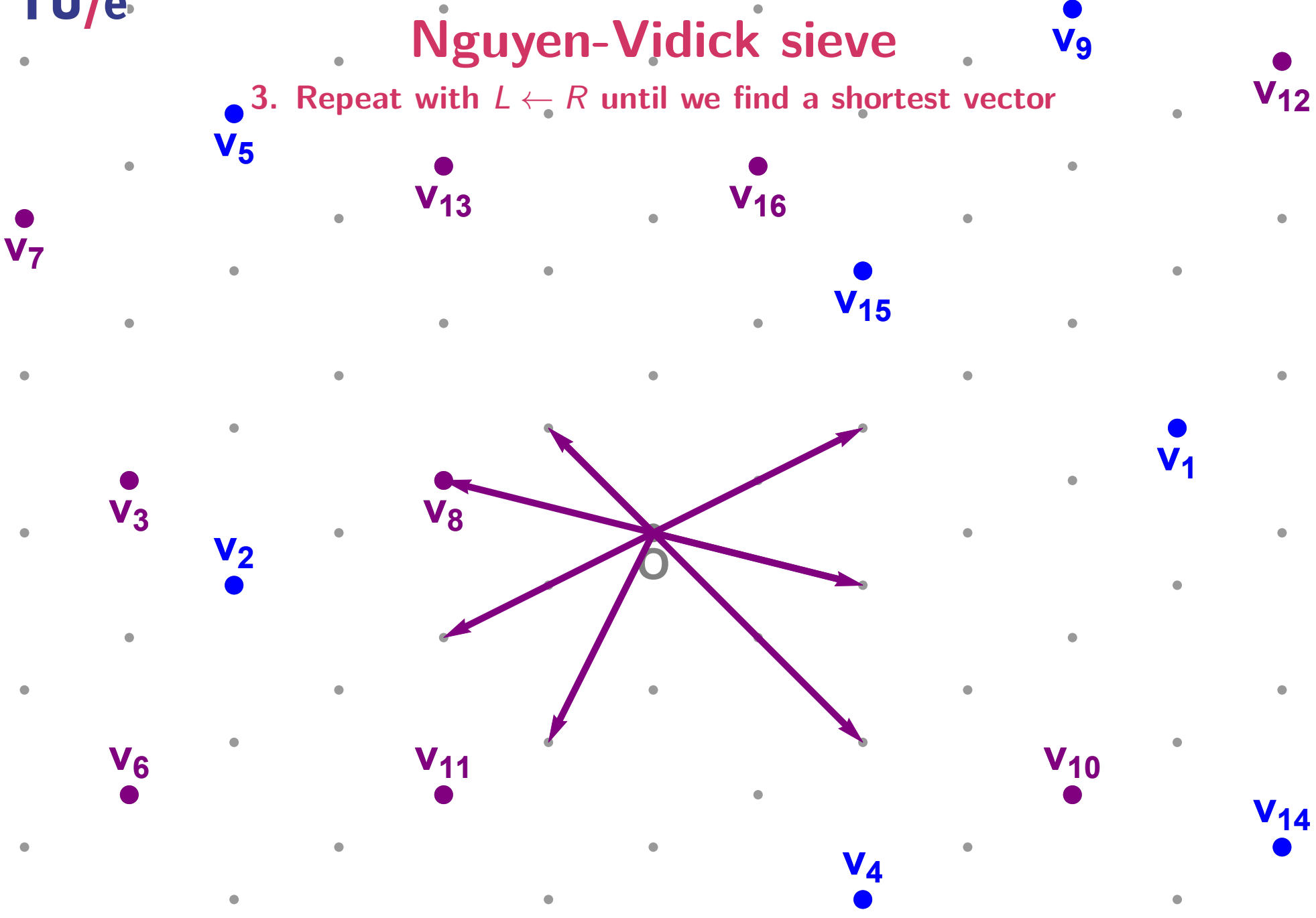
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve

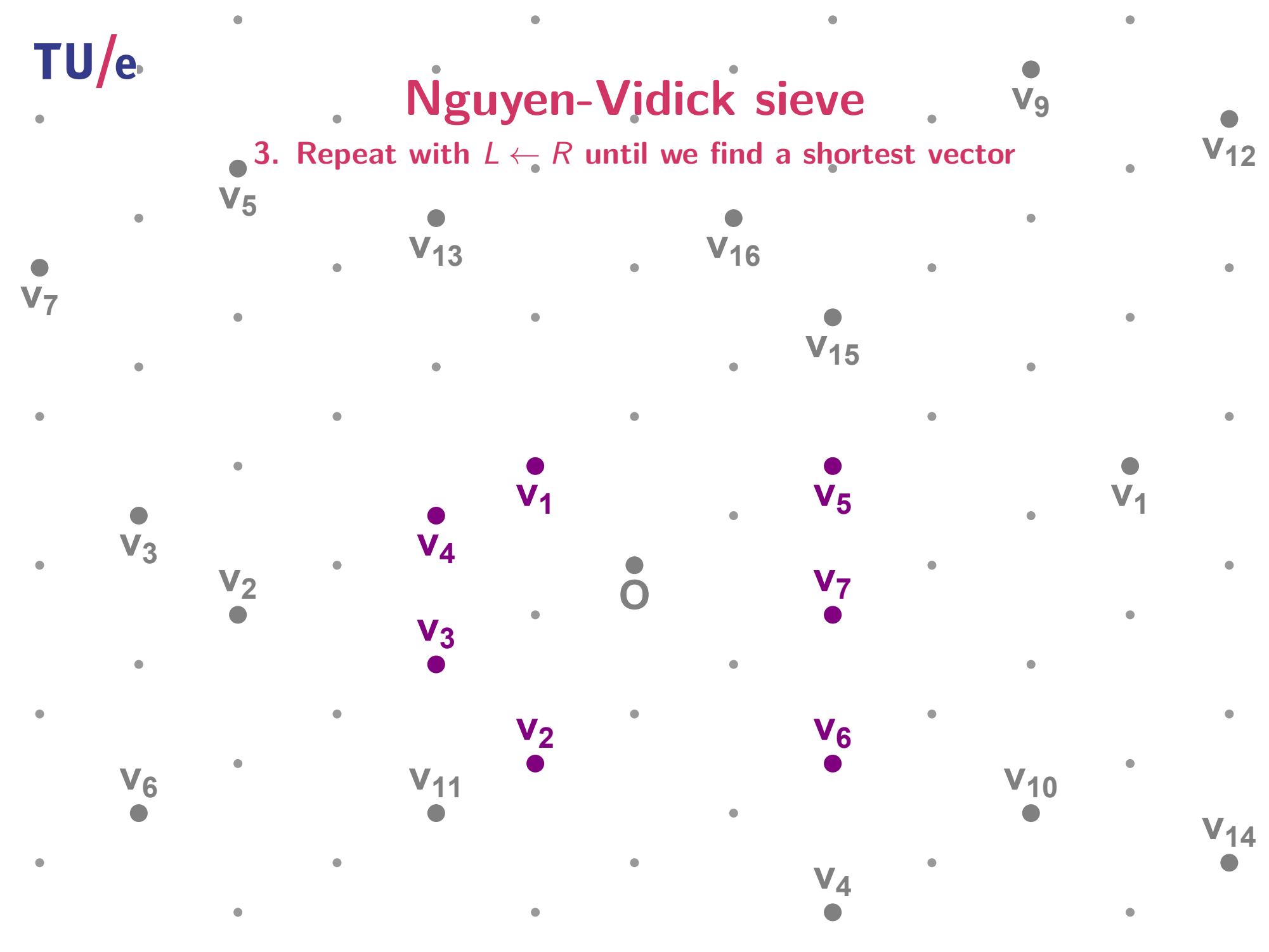
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve

3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve

Overview

TU/e

Nguyen-Vidick sieve

Overview

Heuristic (Nguyen and Vidick, J. Math. Crypt. '08)

The Nguyen-Vidick sieve runs in time $(4/3)^n$ and space $\sqrt{4/3}^n$.

TU/e

Nguyen-Vidick sieve

Overview

Heuristic (Nguyen and Vidick, J. Math. Crypt. '08)

The Nguyen-Vidick sieve runs in time $2^{0.415n}$ and space $2^{0.208n}$.

TU/e

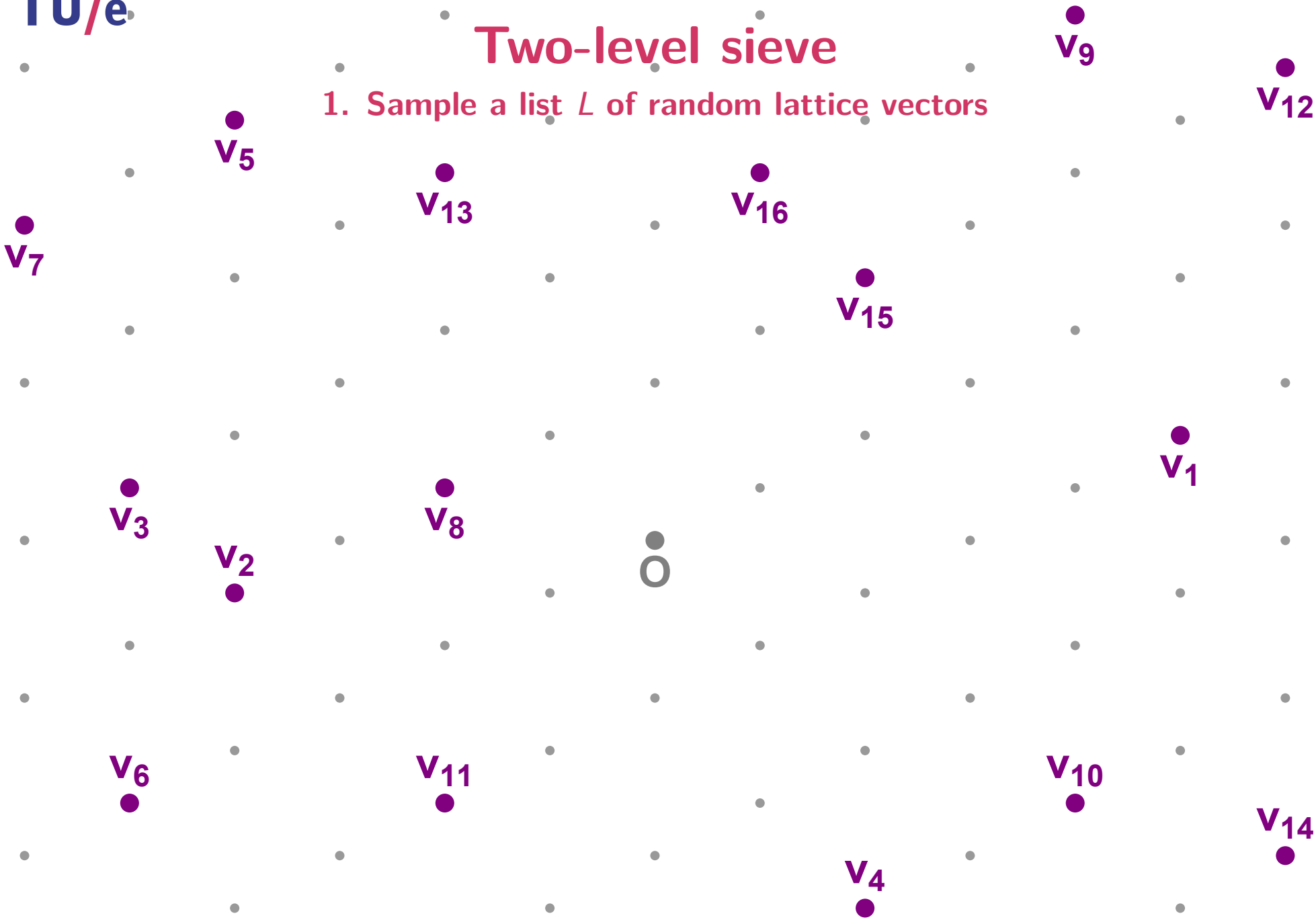
Two-level sieve

1. Sample a list L of random lattice vectors

TU/e

Two-level sieve

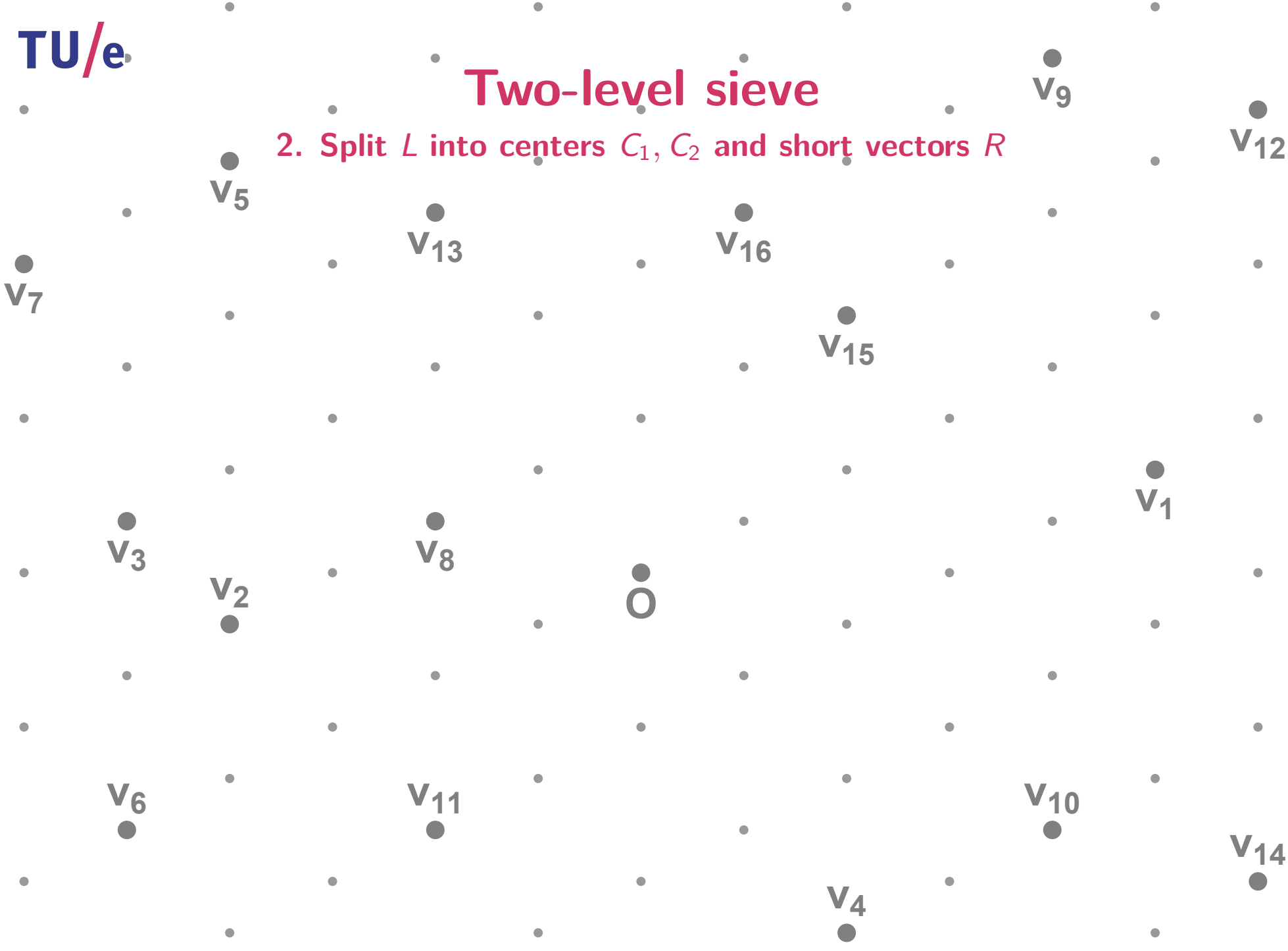
1. Sample a list L of random lattice vectors



TU/e

Two-level sieve

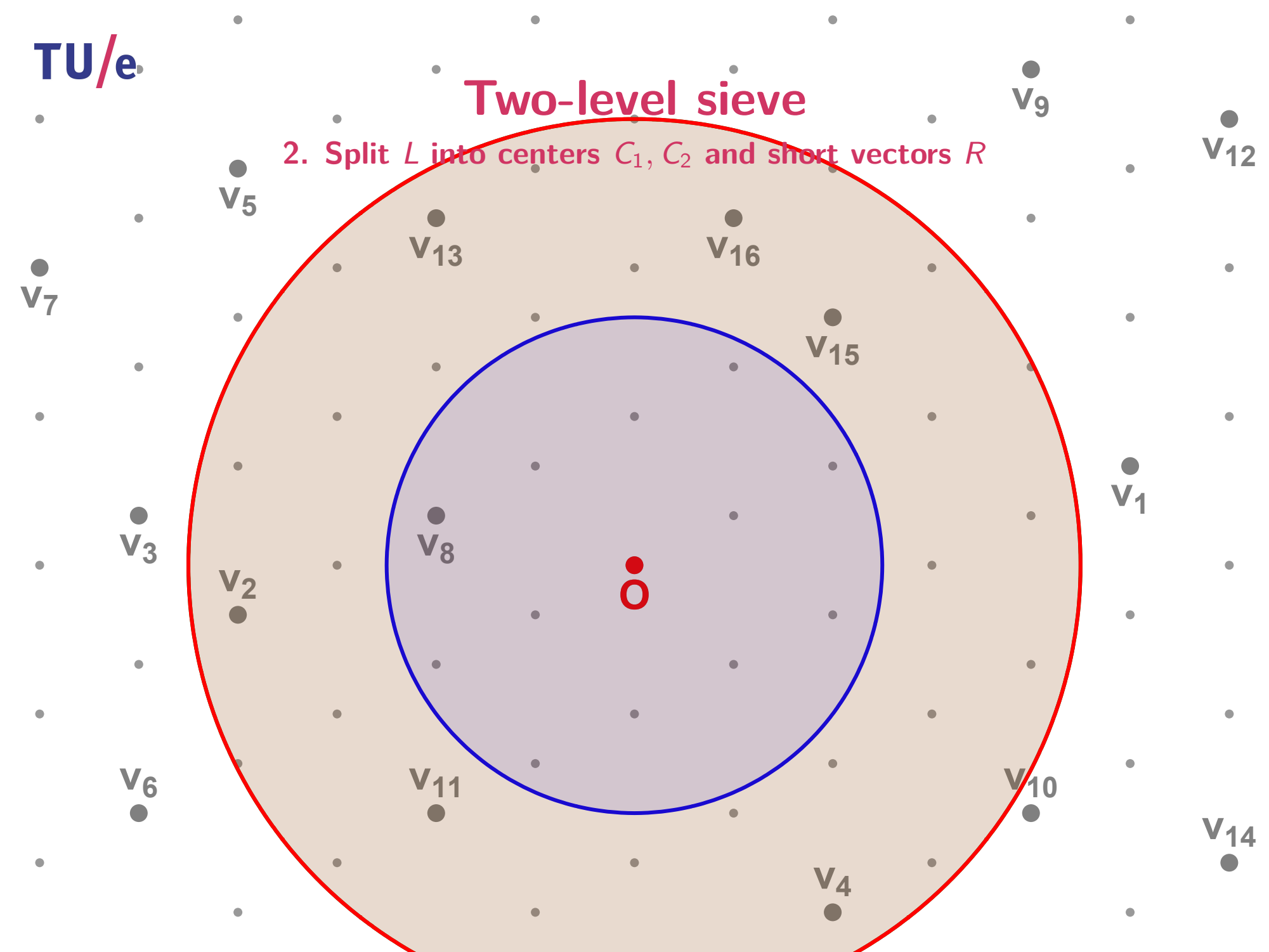
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

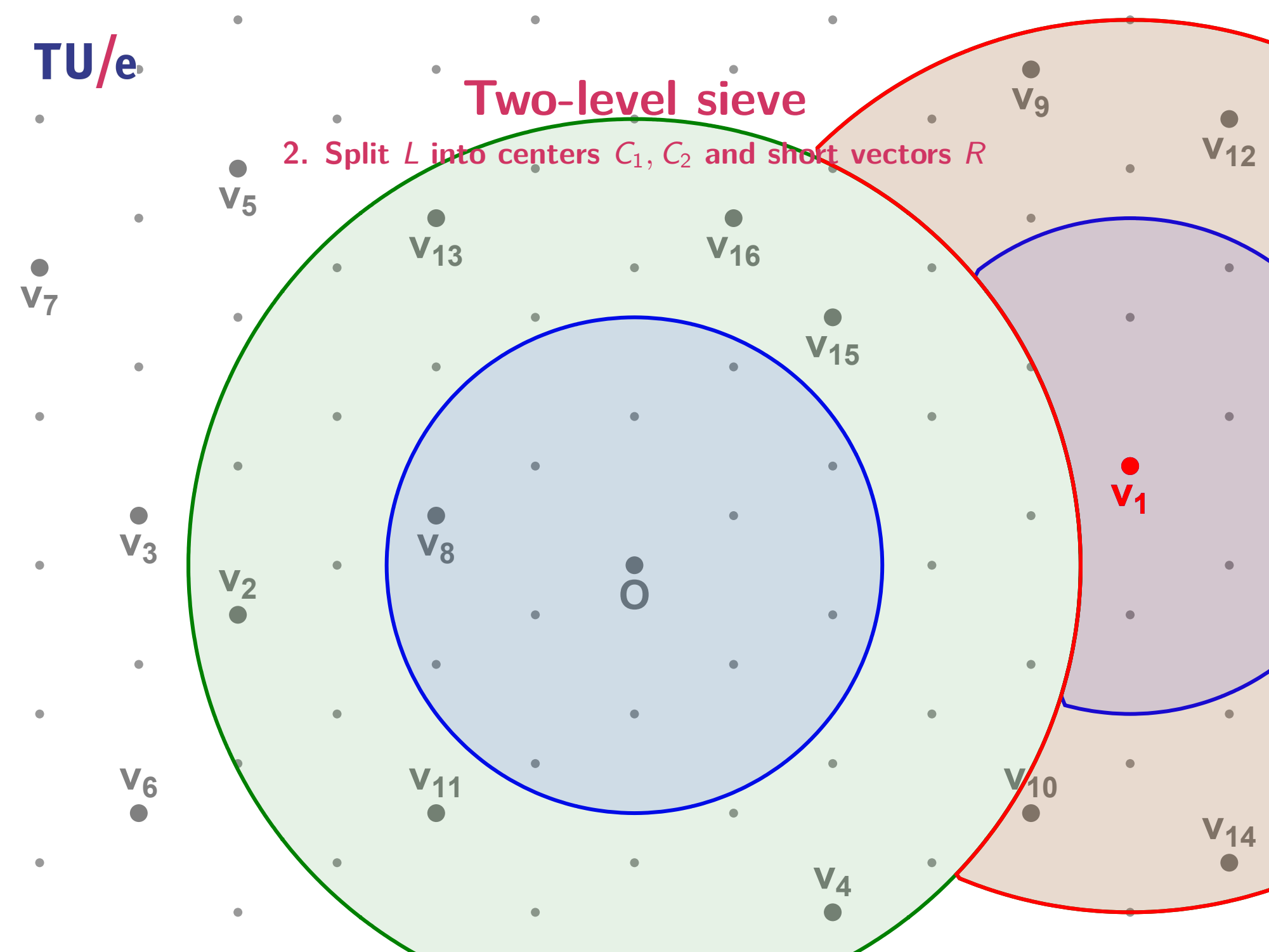
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

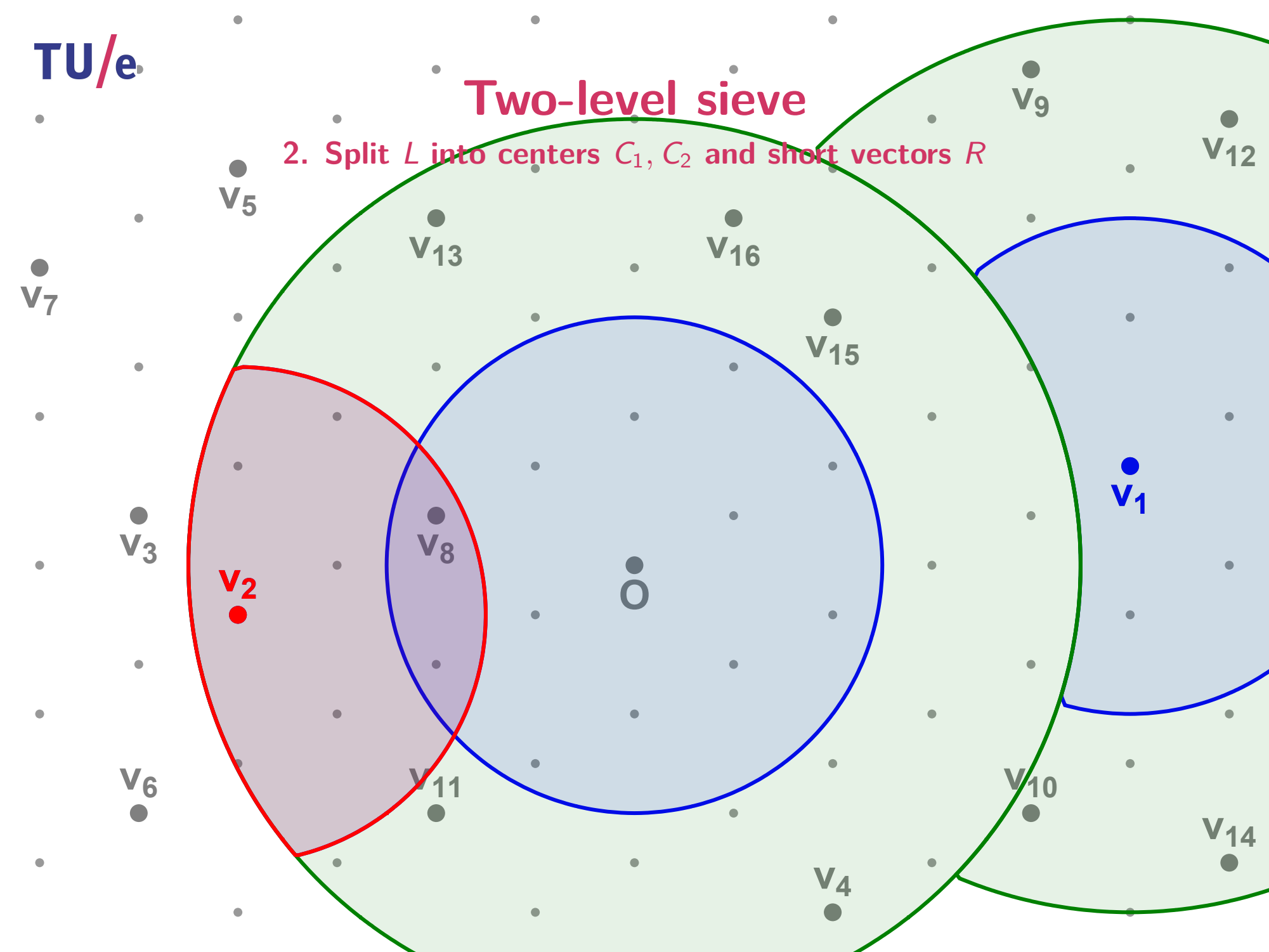
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

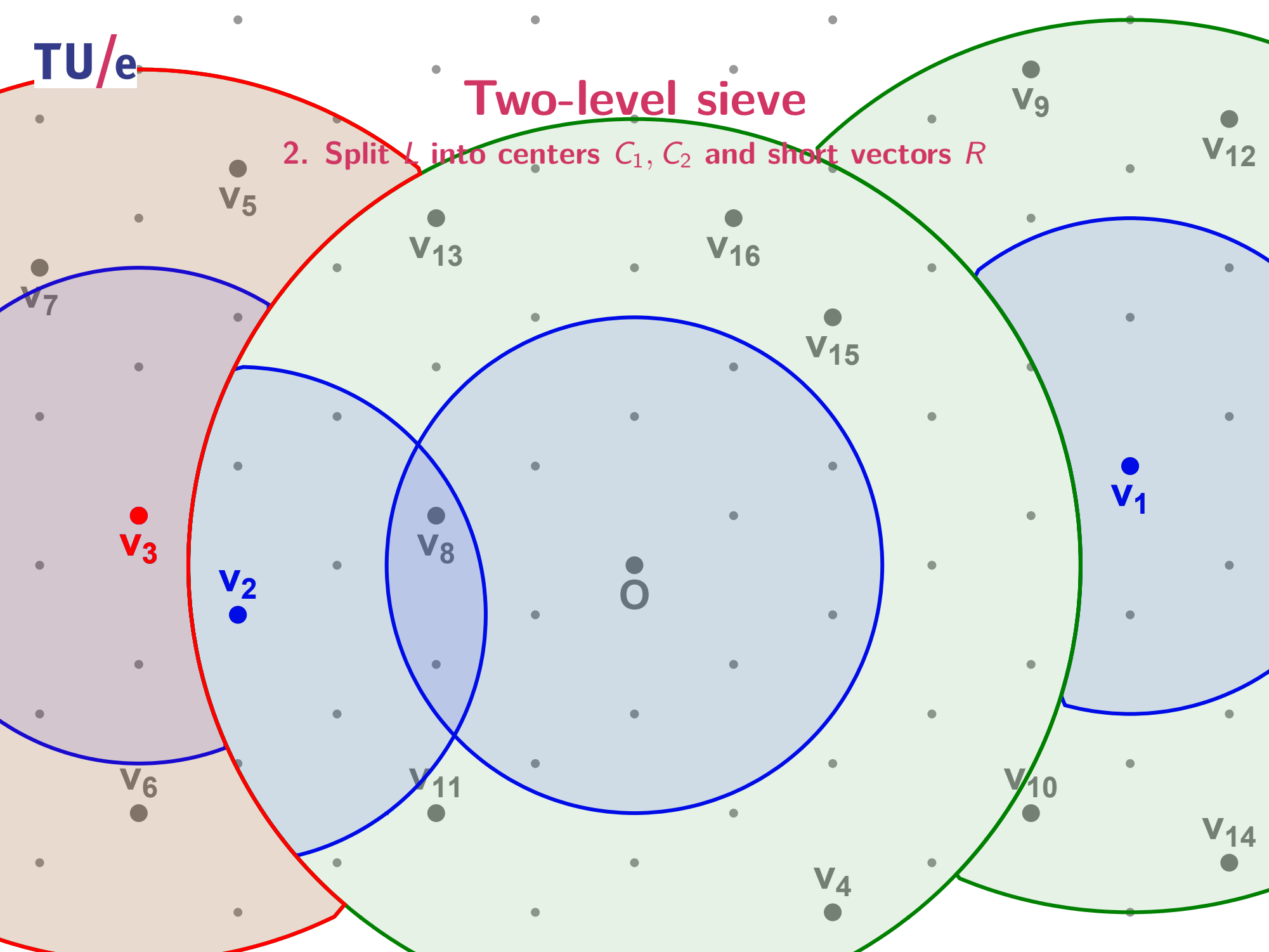
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

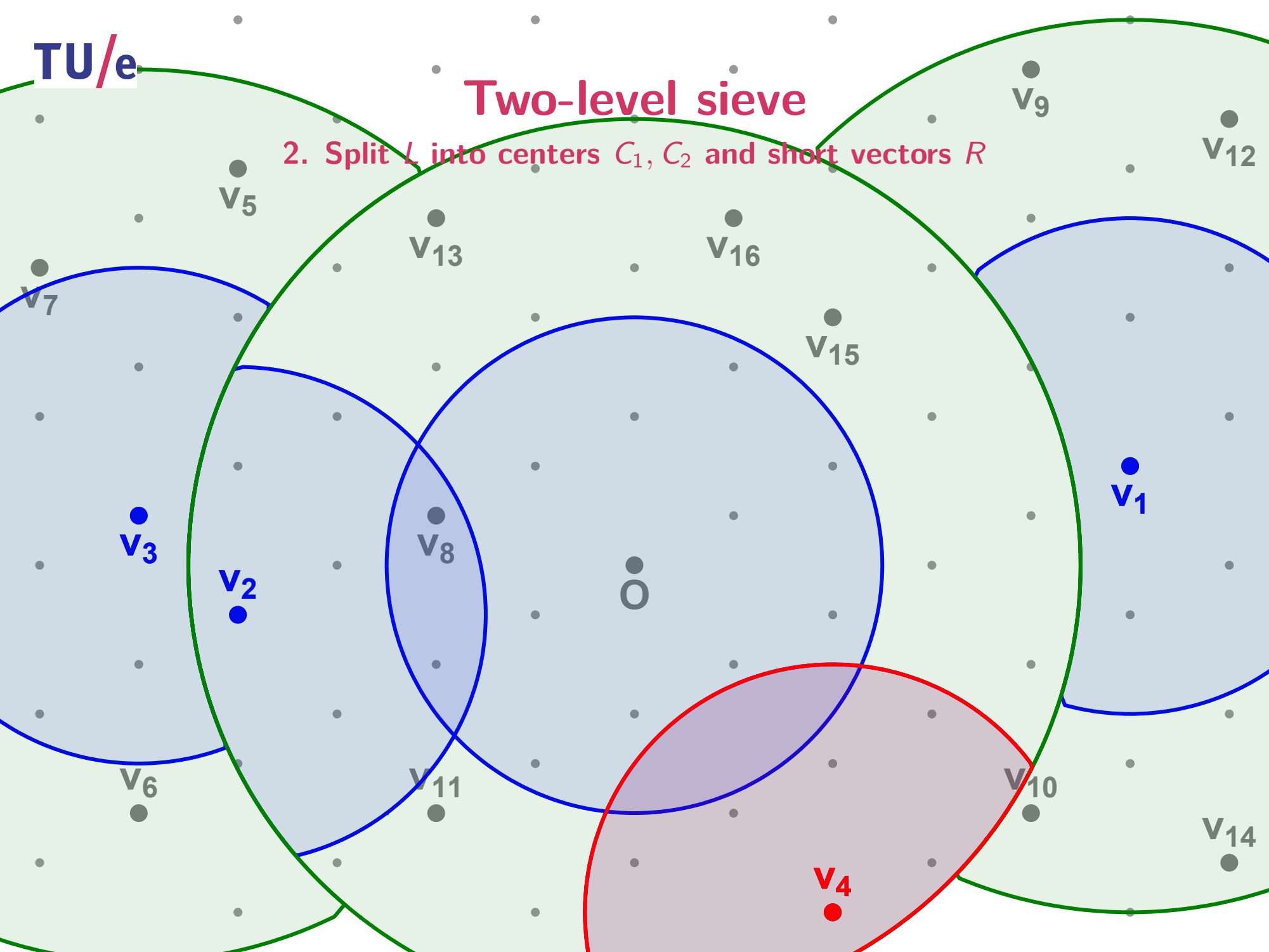
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

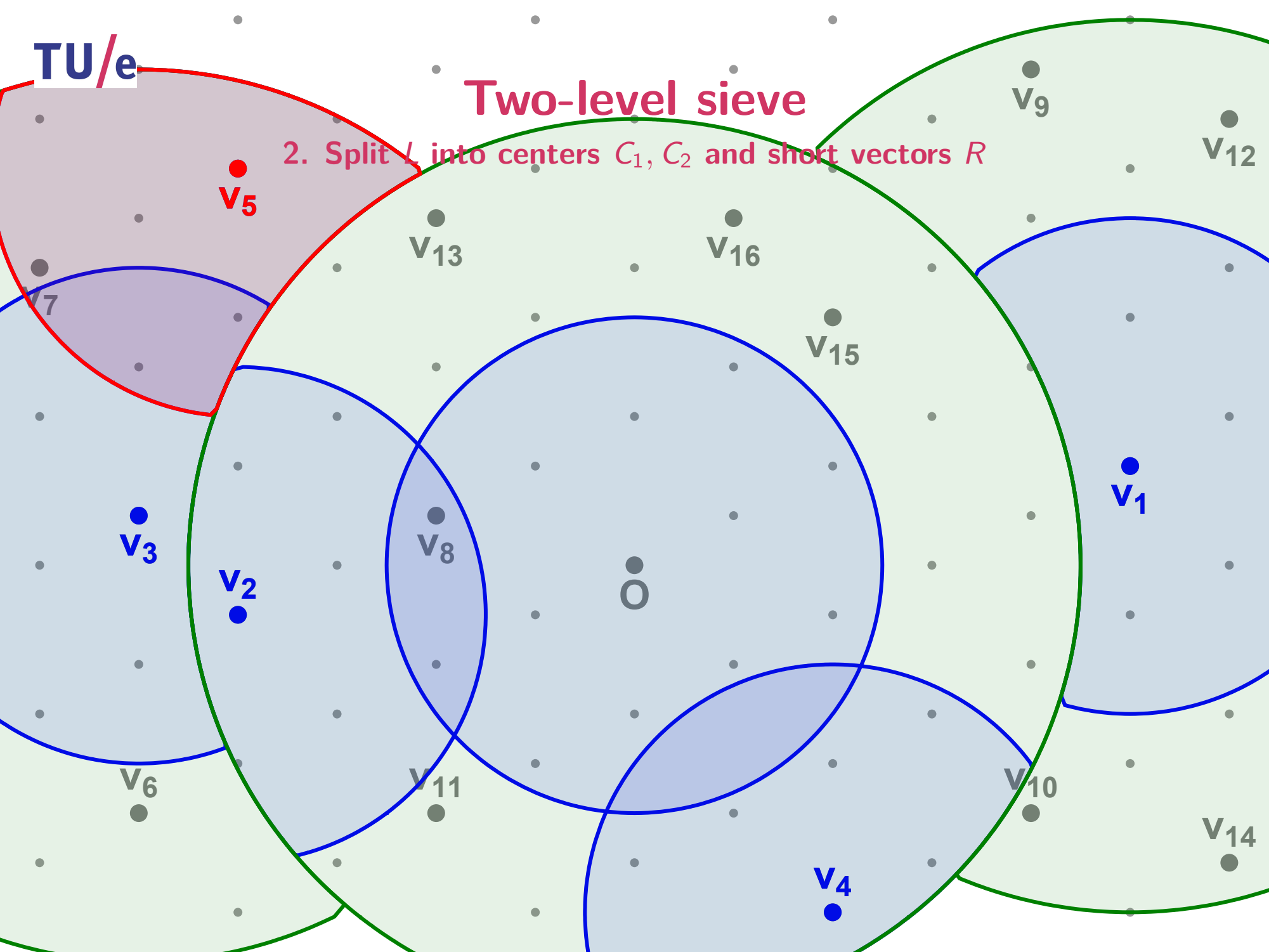
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

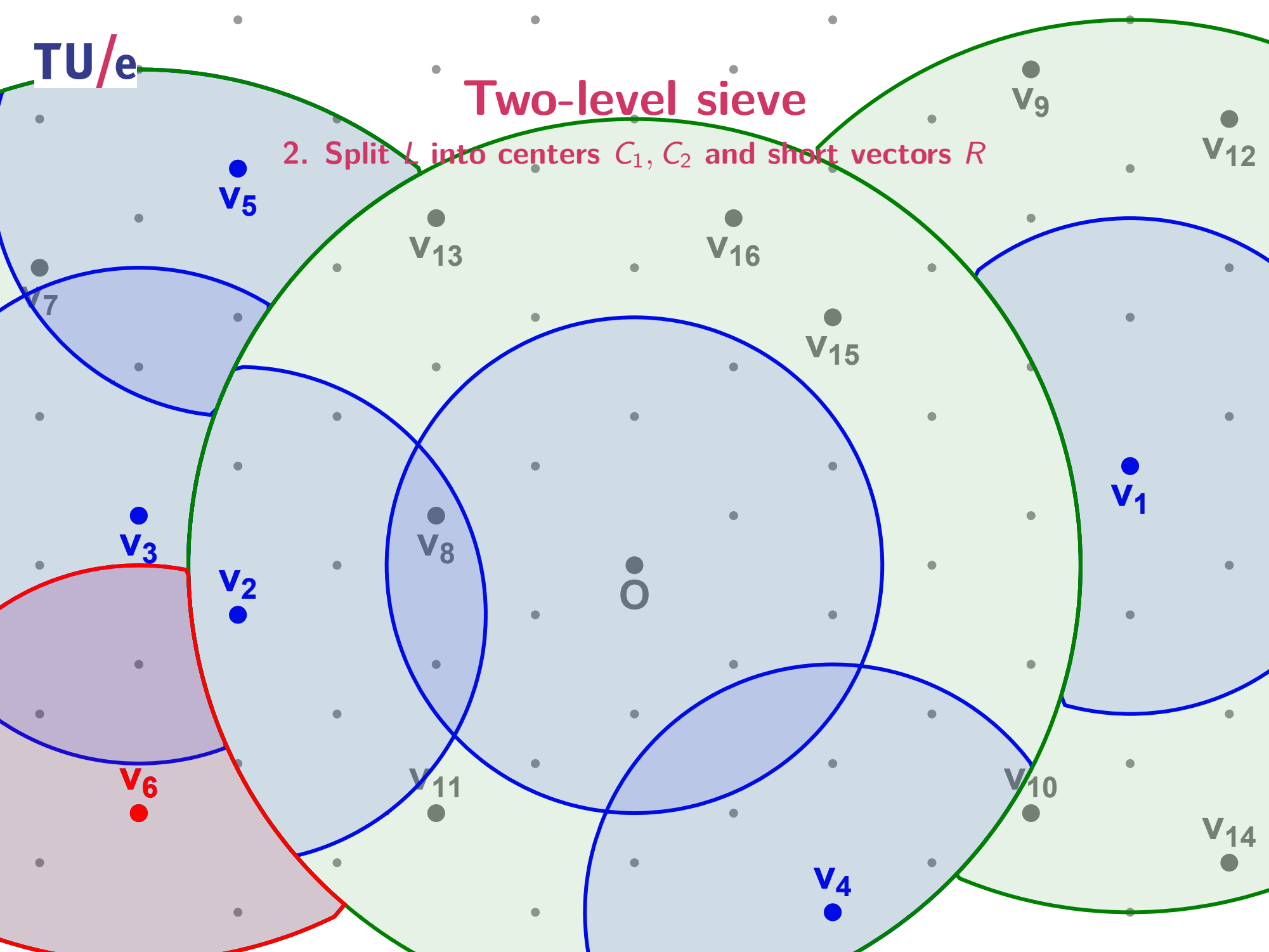
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

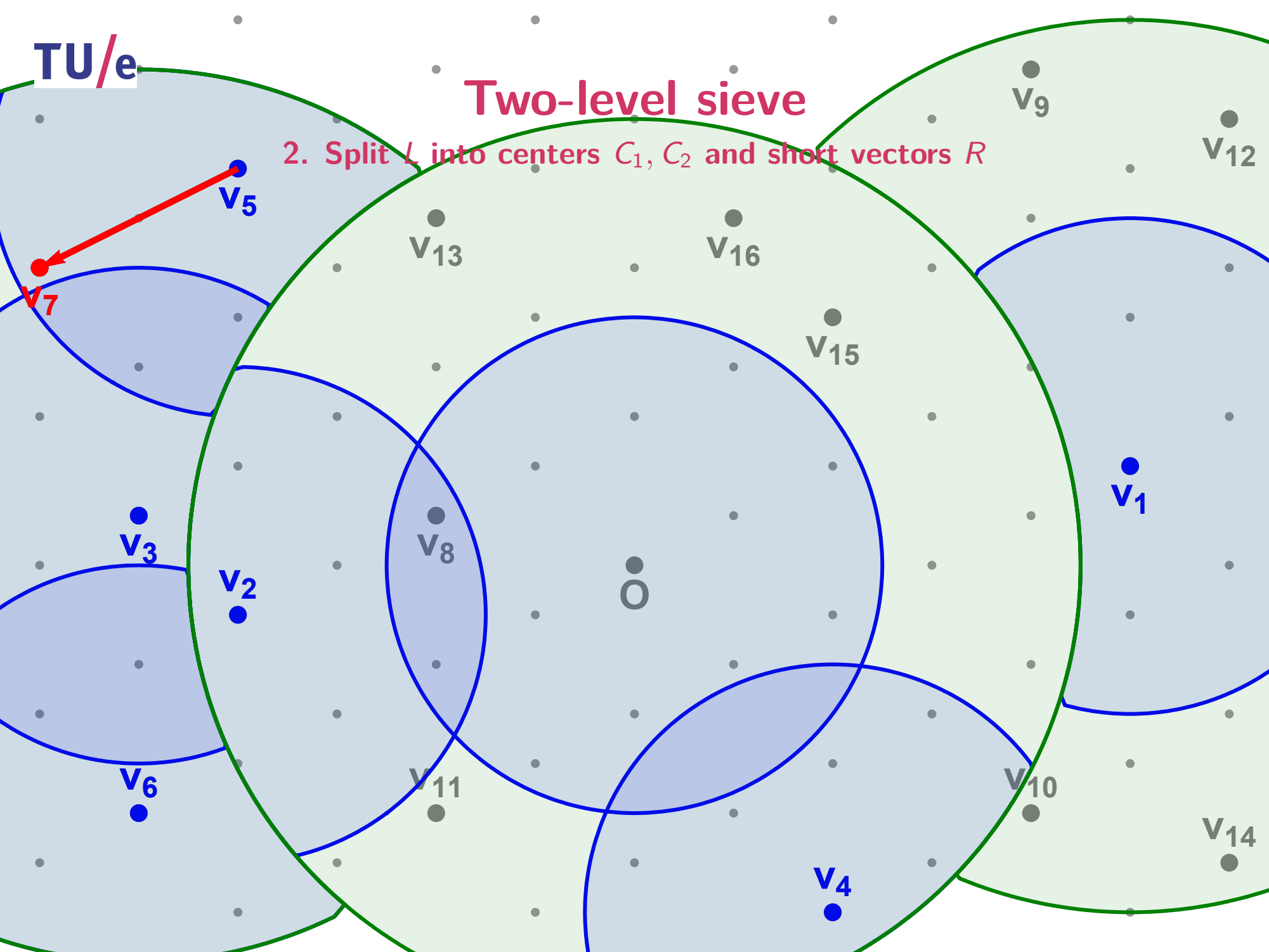
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

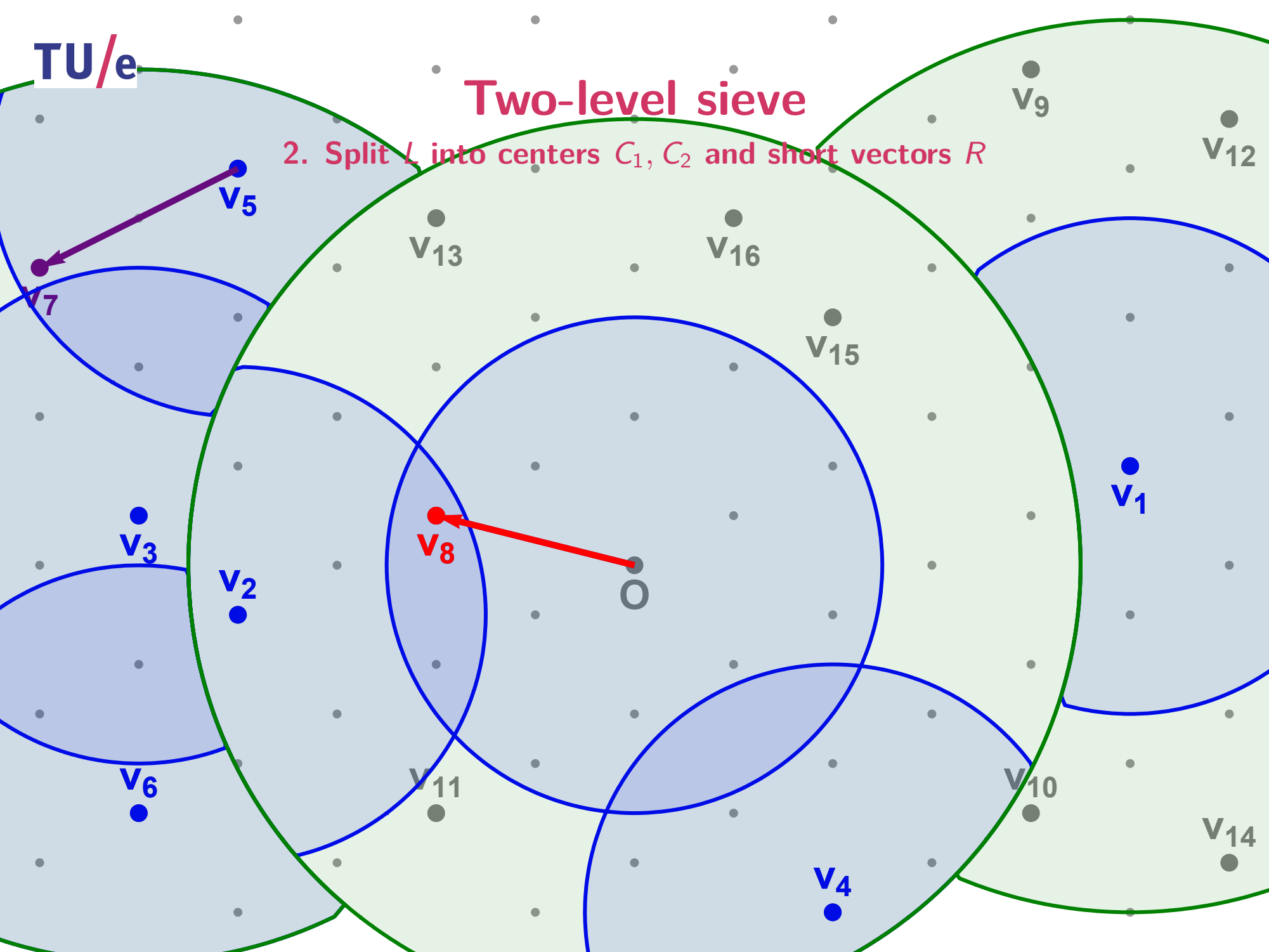
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

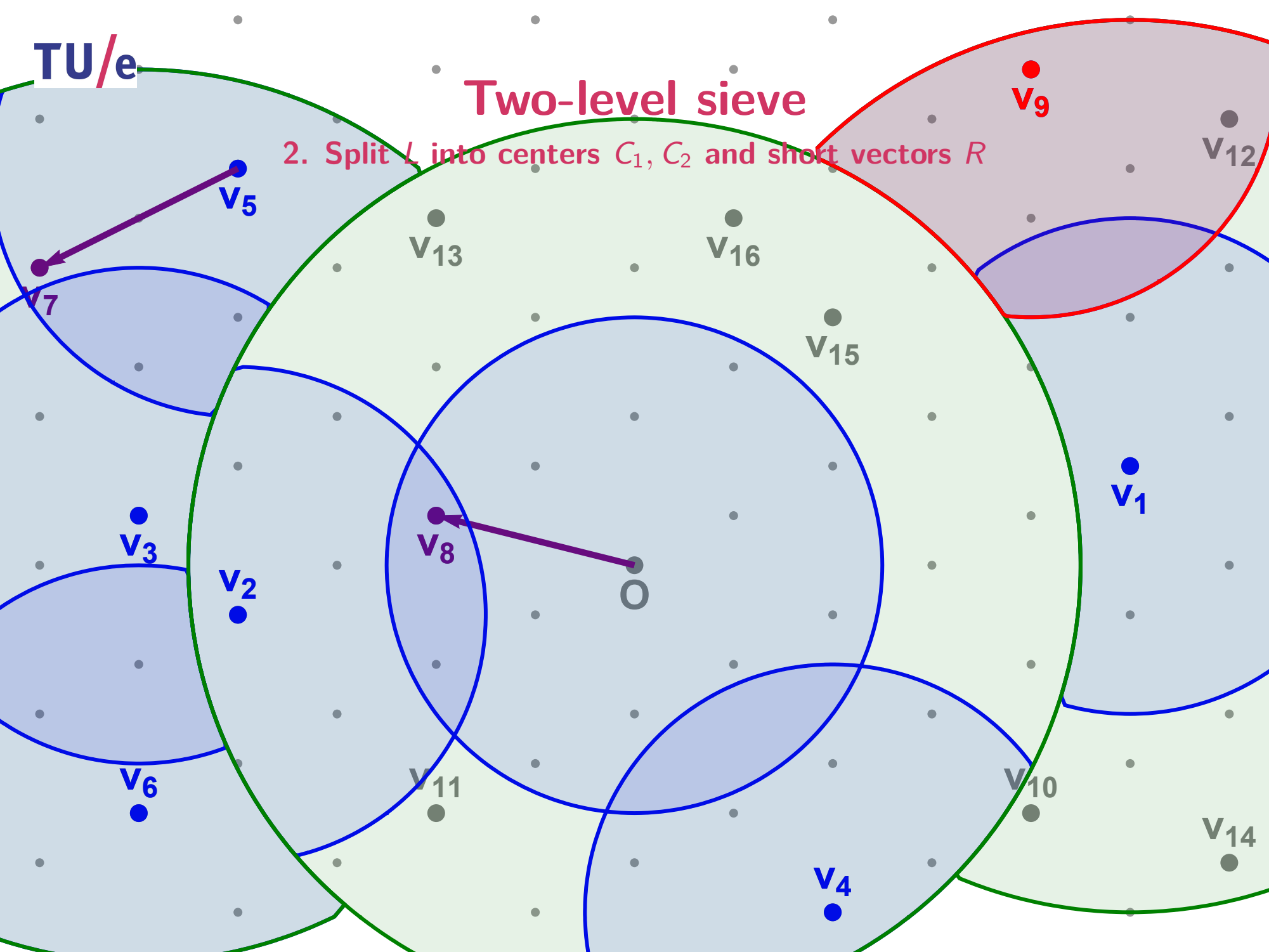
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

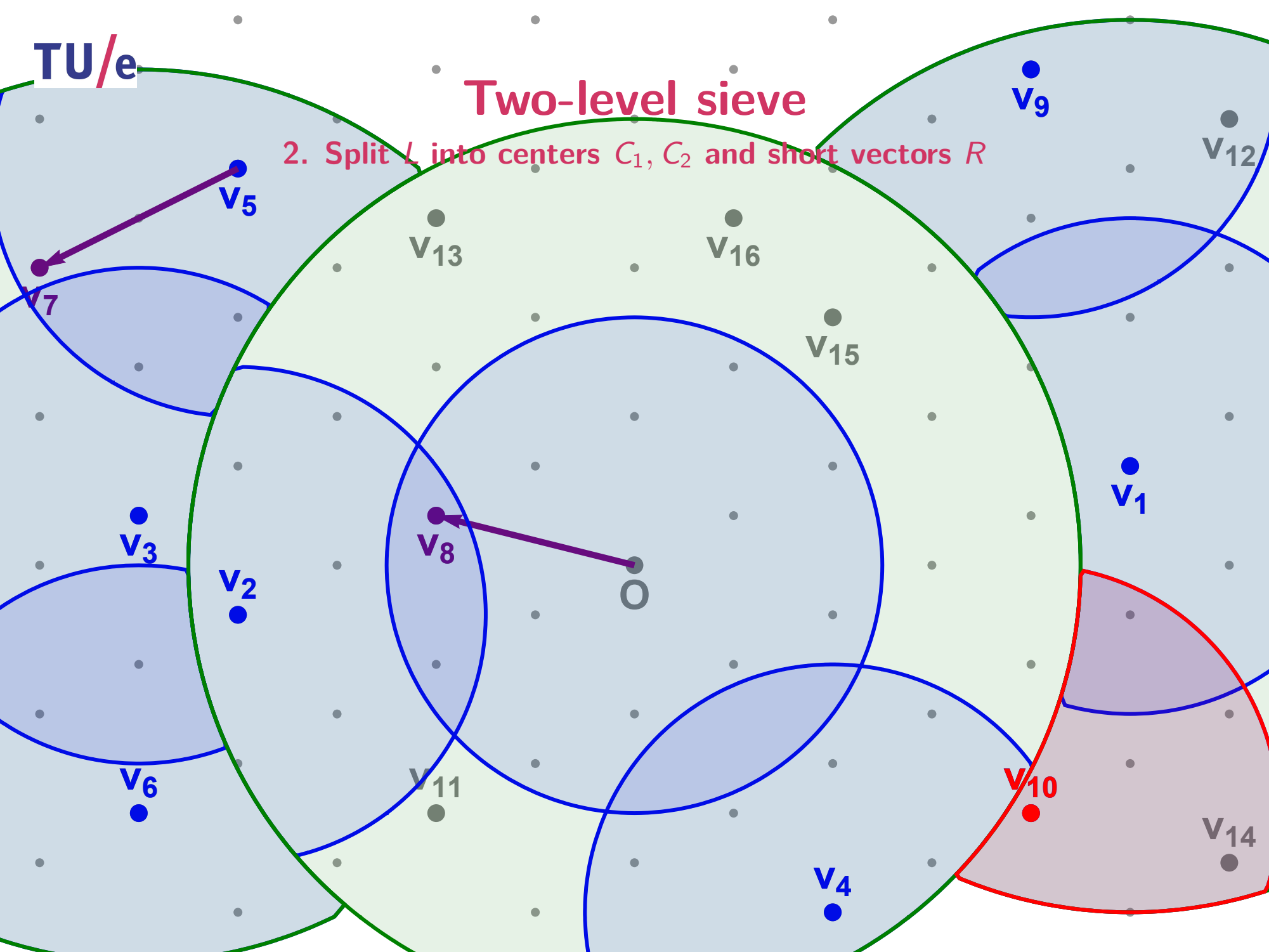
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

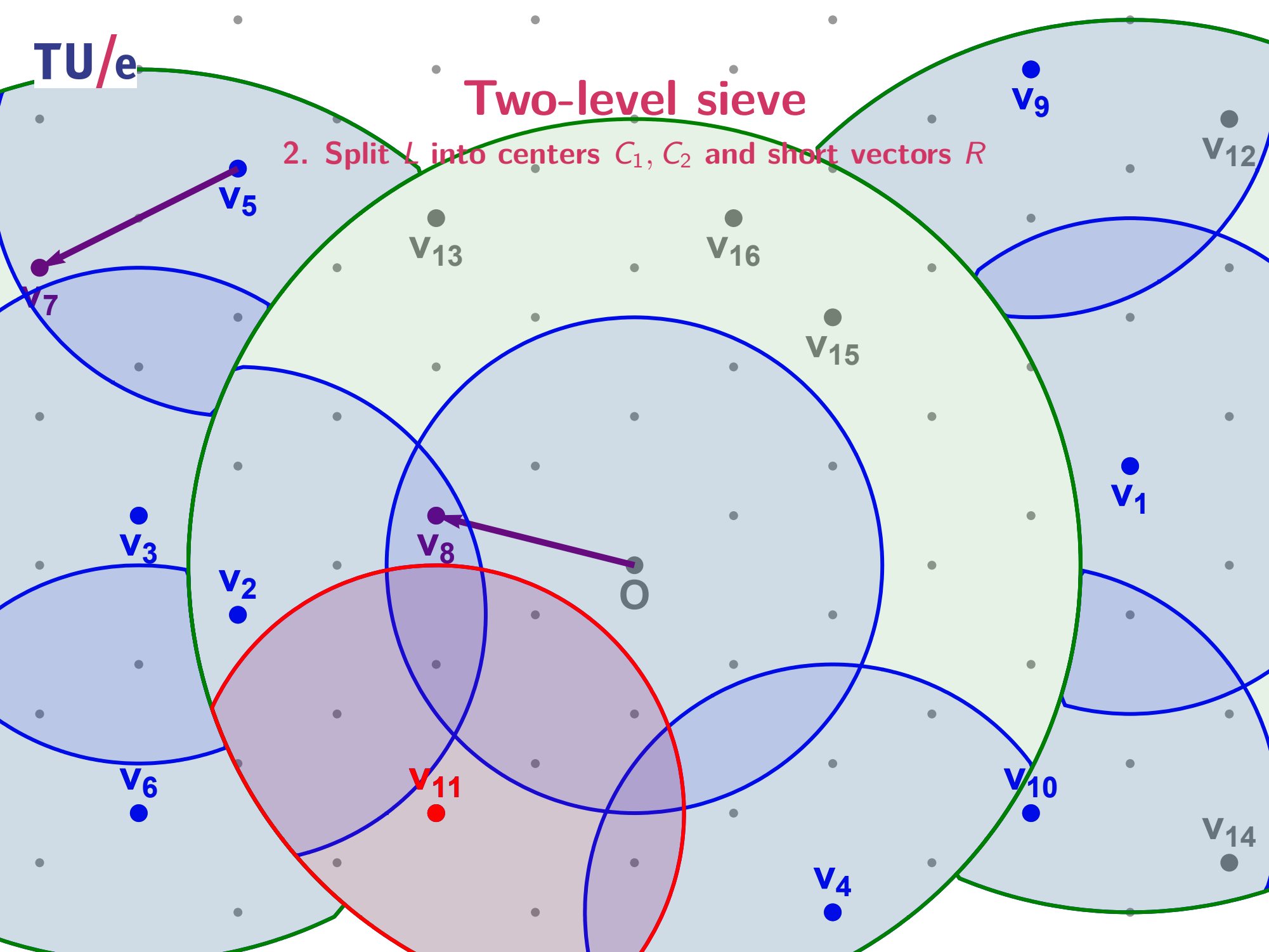
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

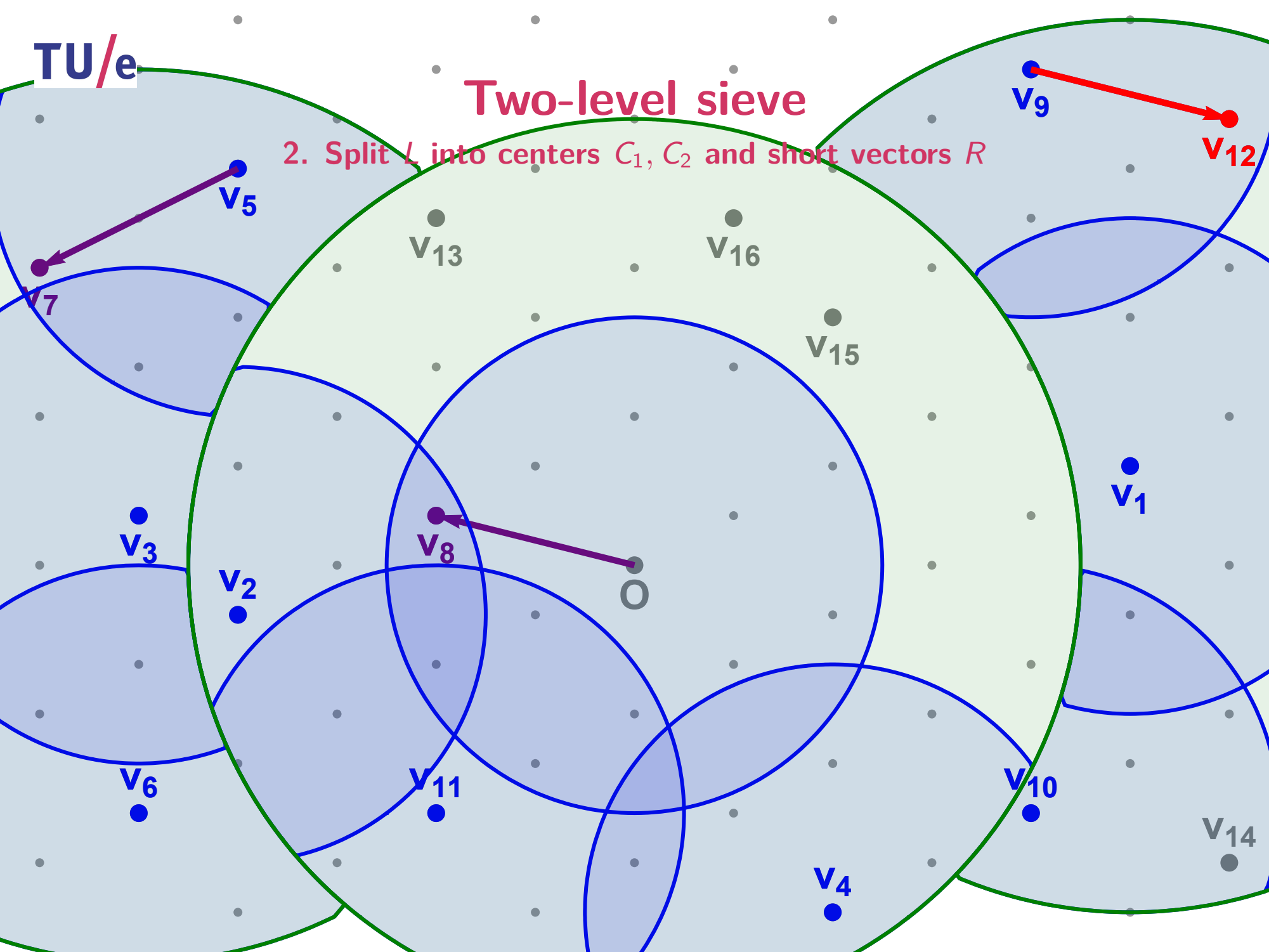
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

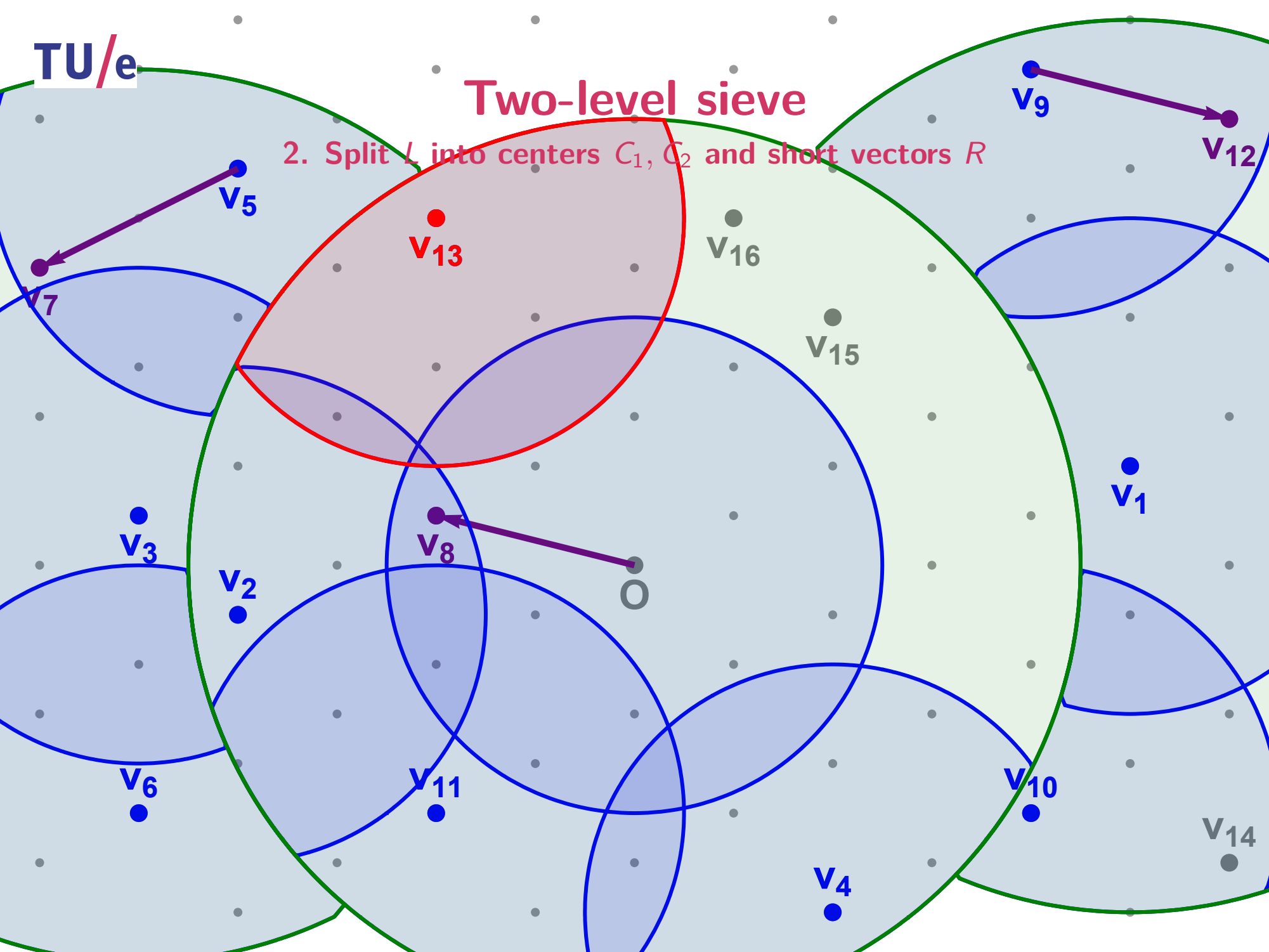
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

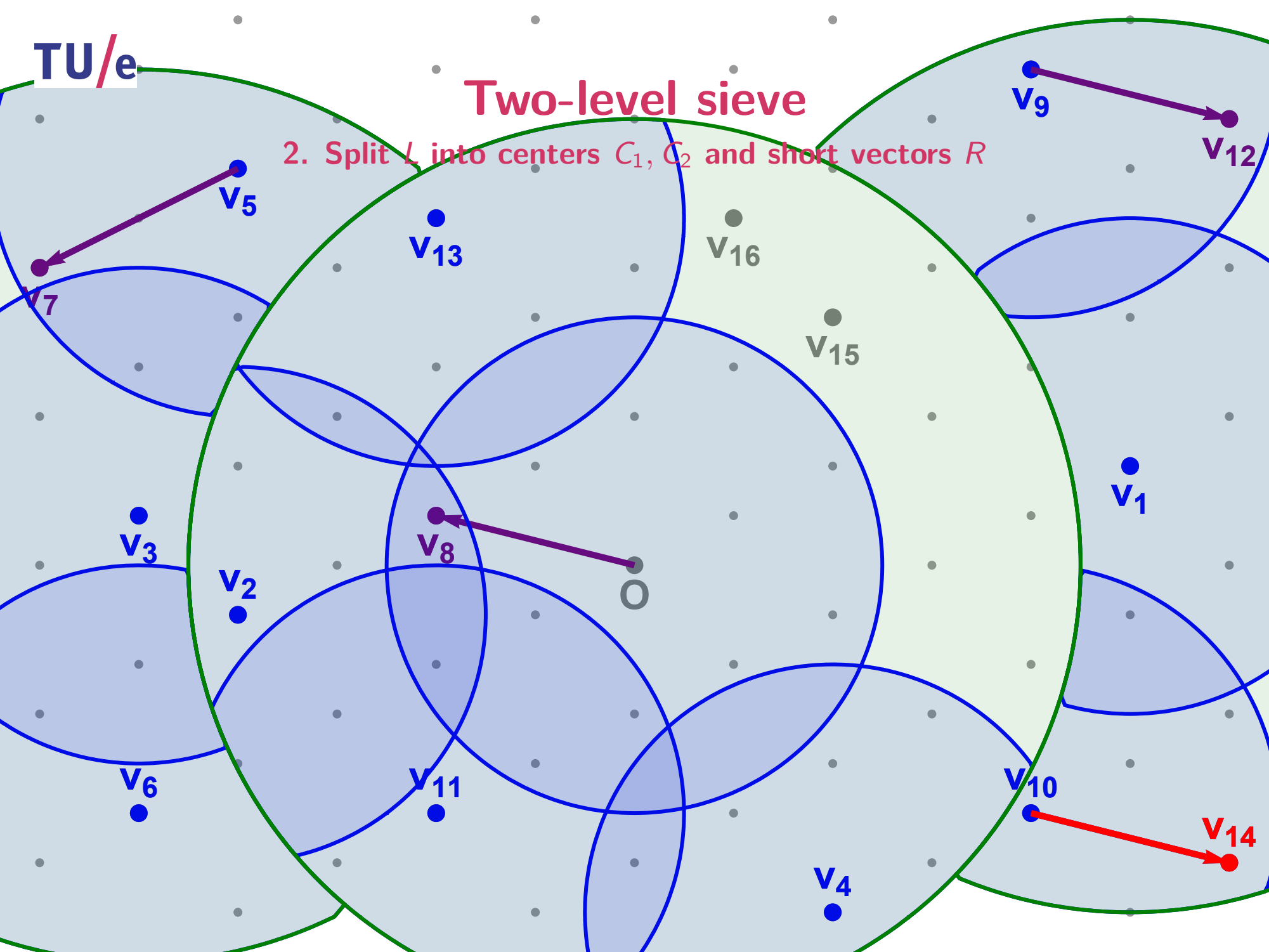
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

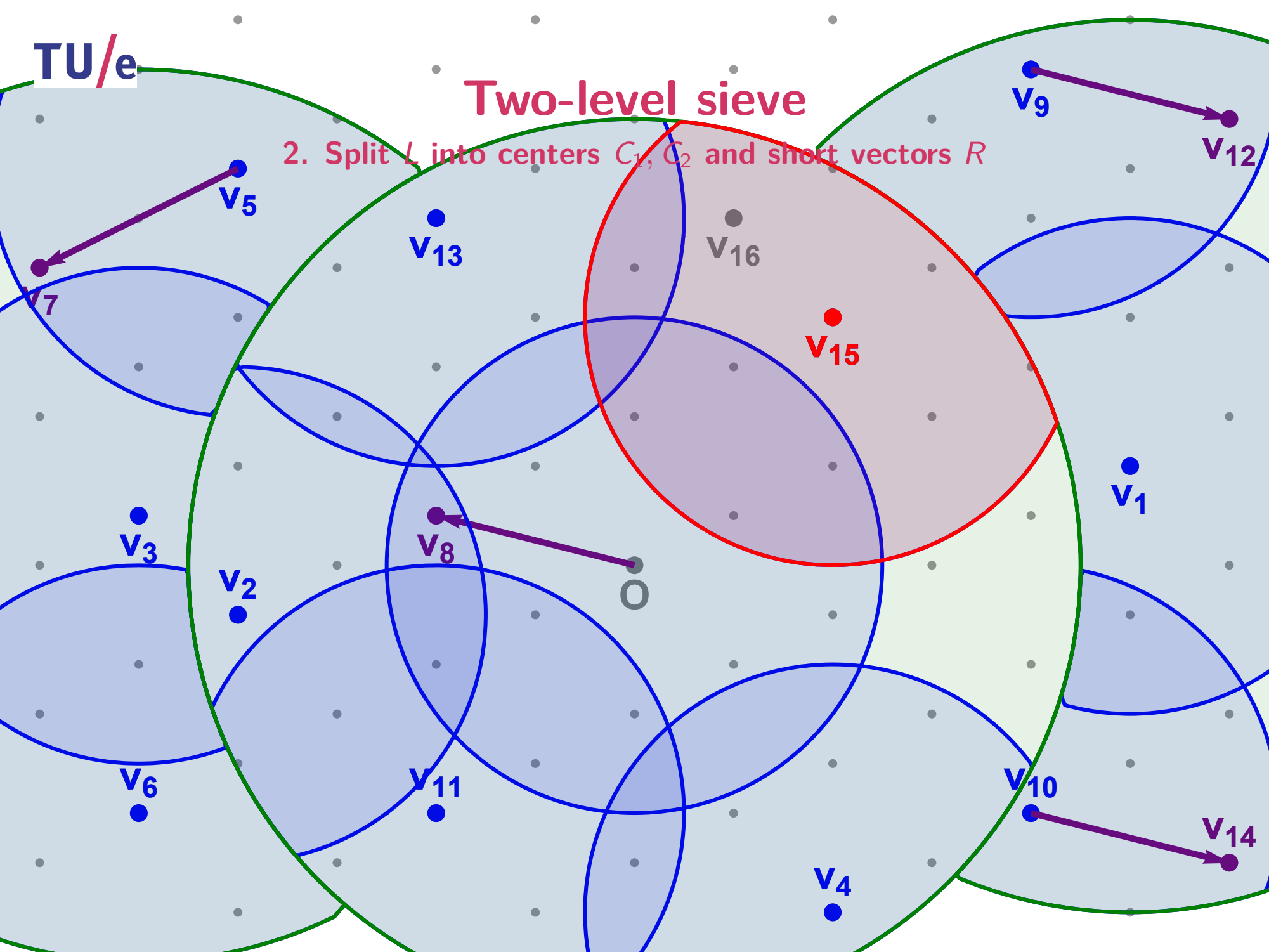
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

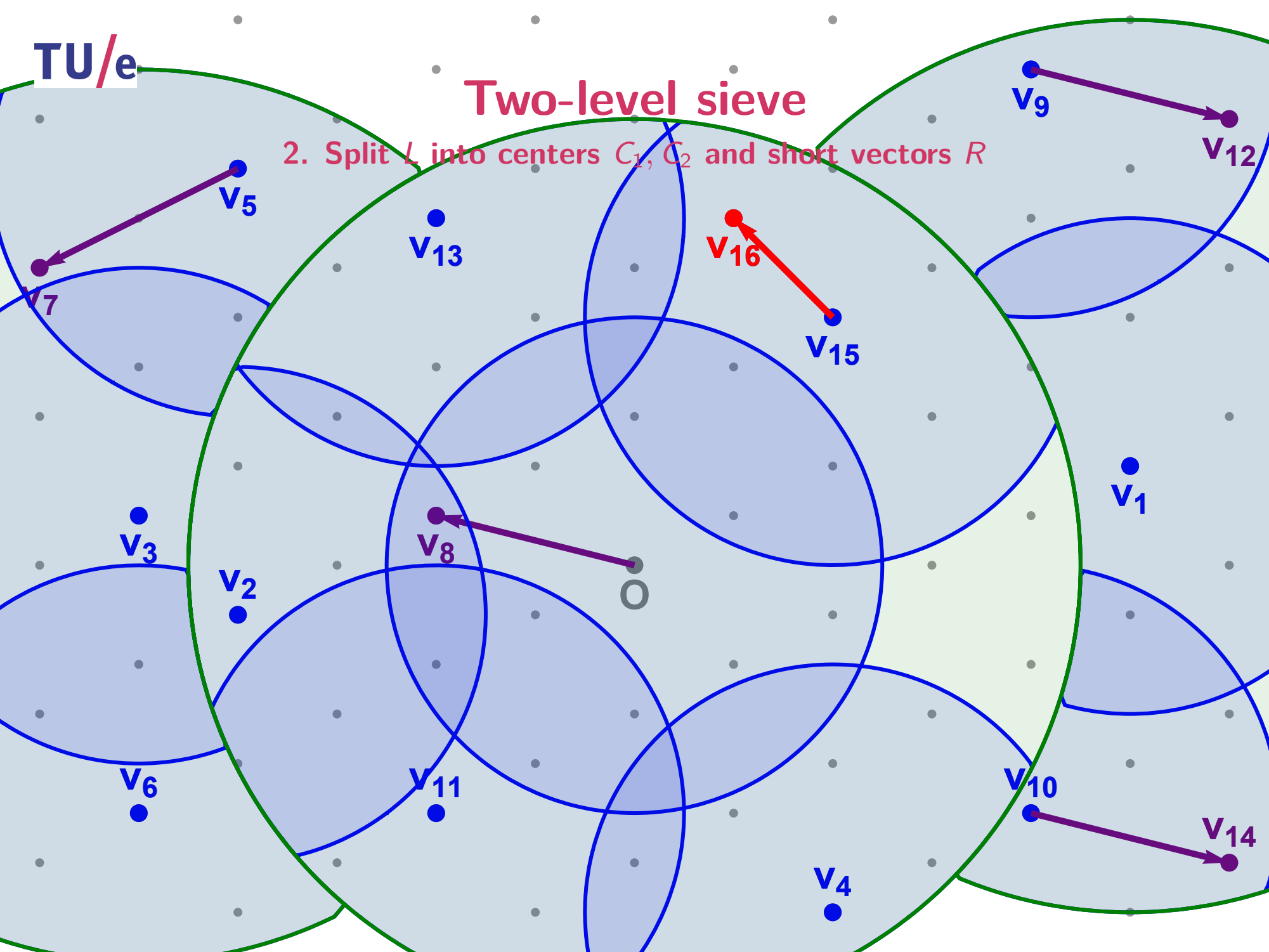
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

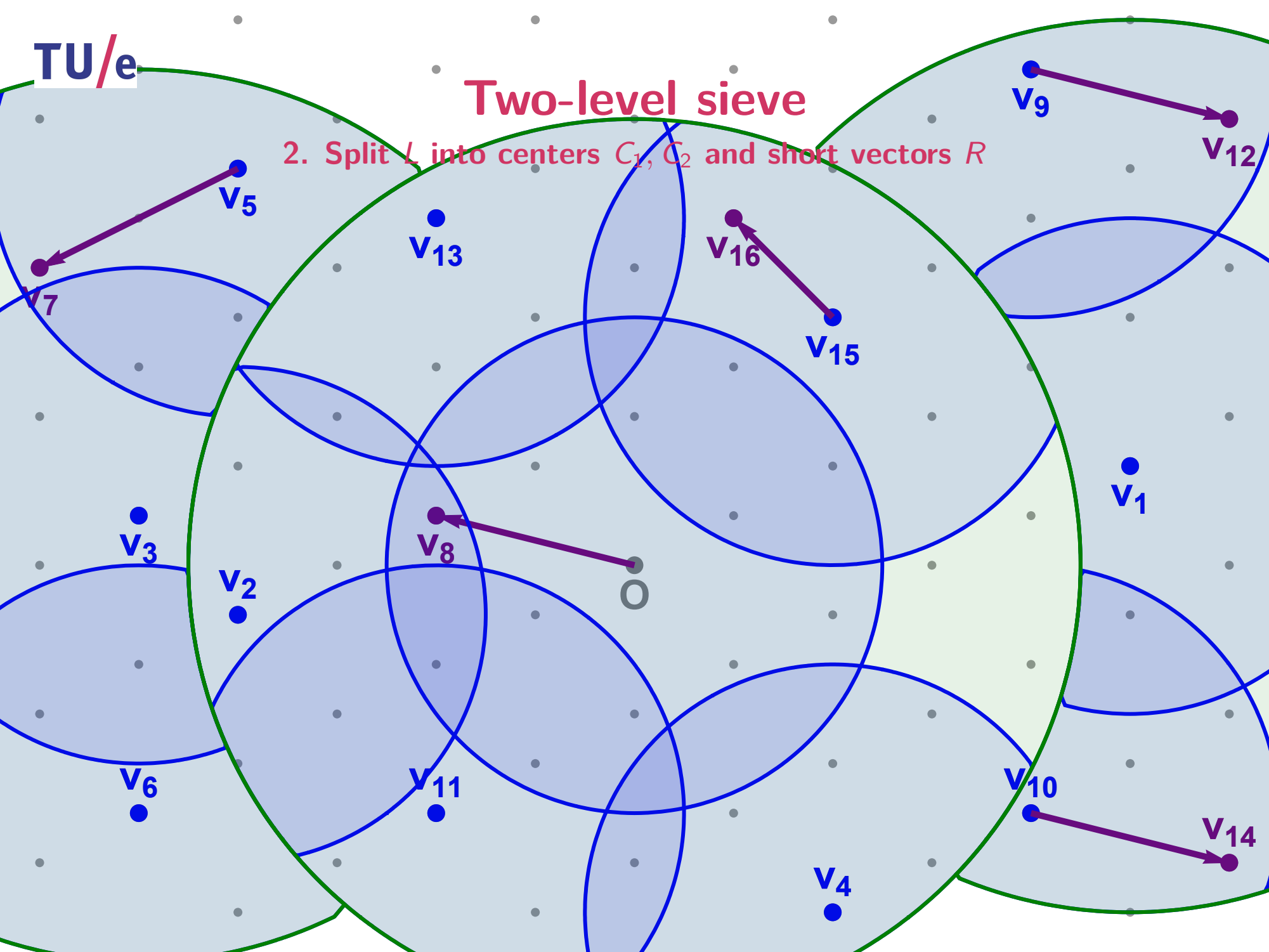
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

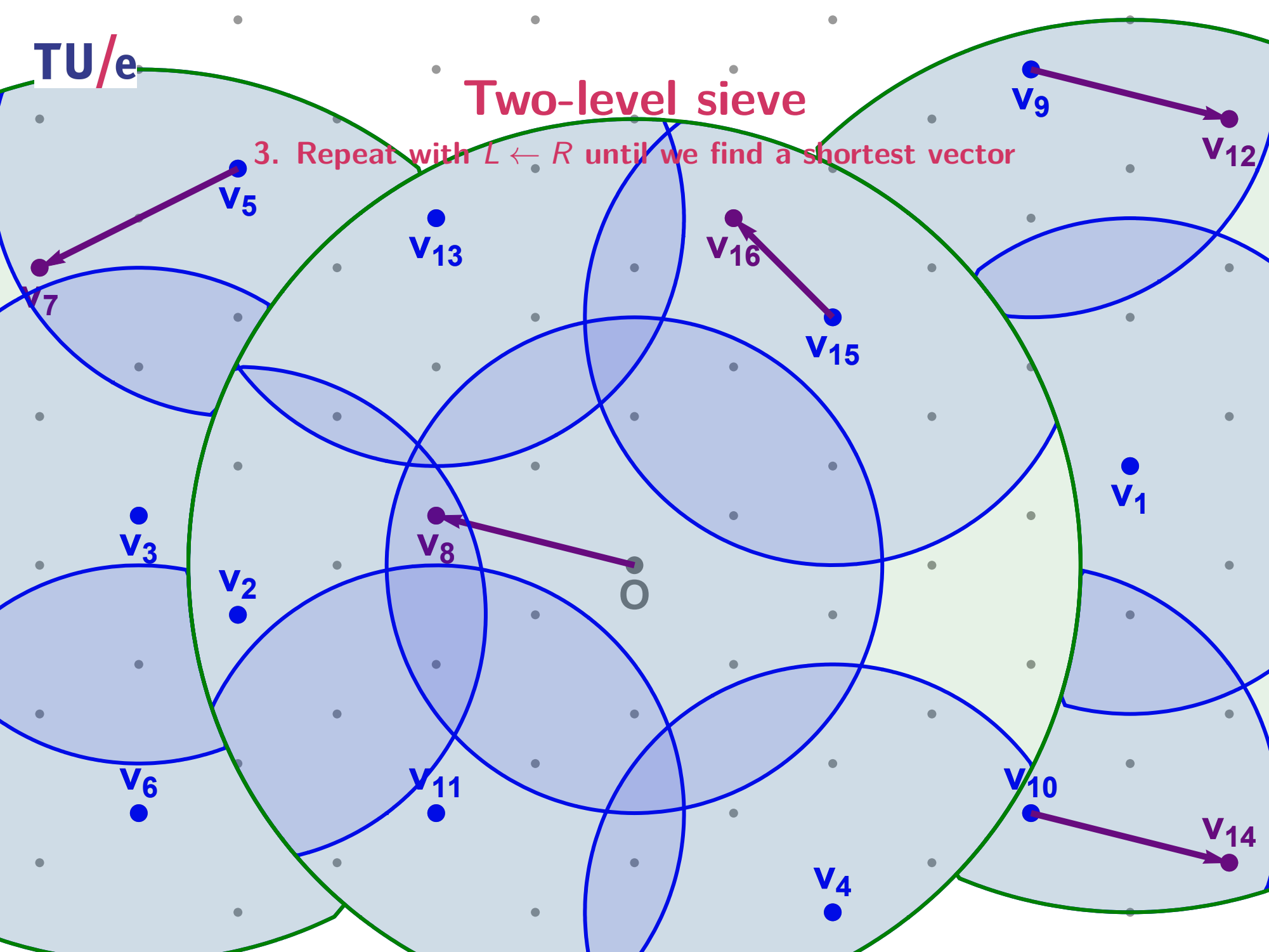
2. Split L into centers C_1, C_2 and short vectors R



TU/e

Two-level sieve

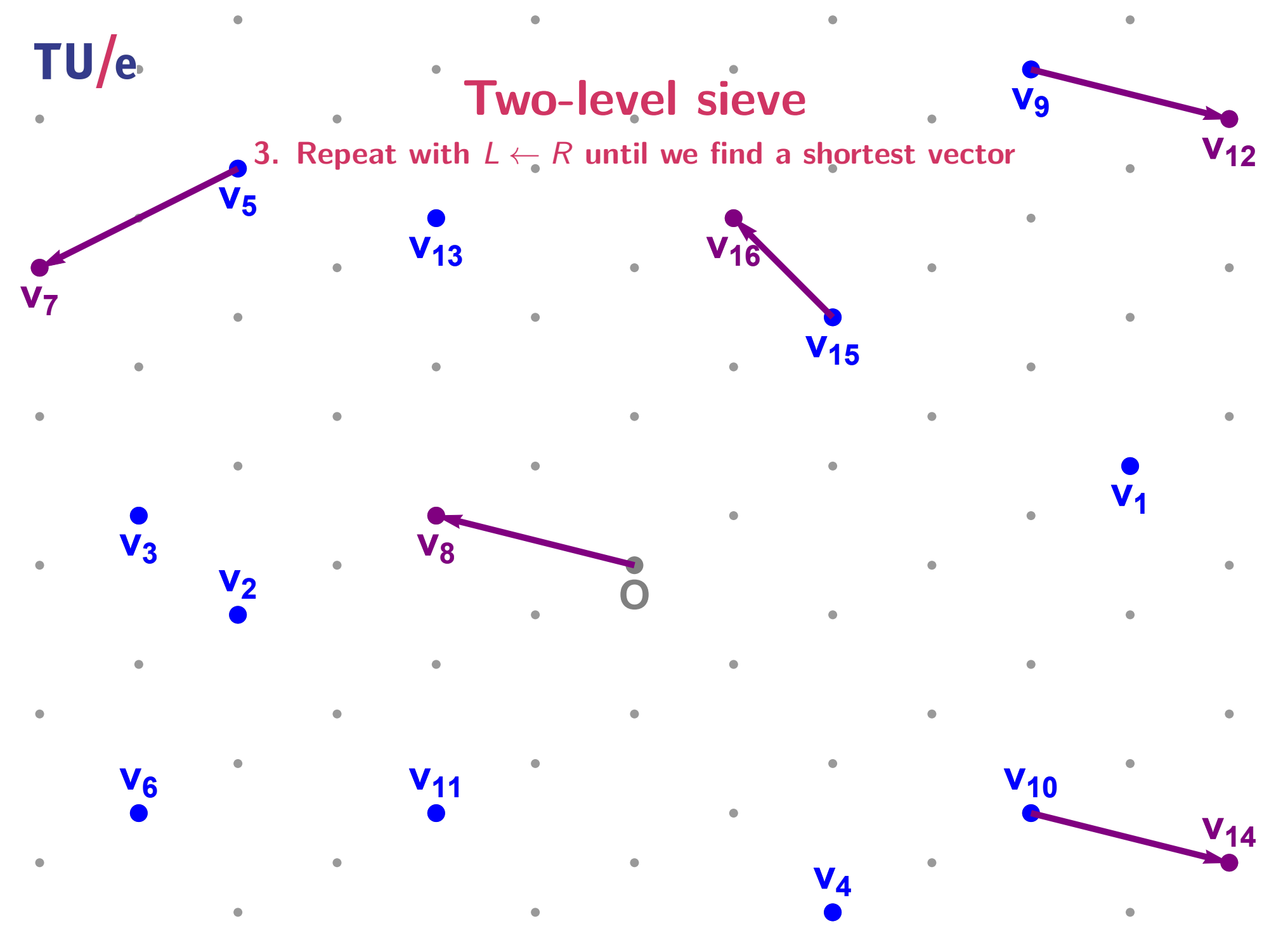
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Two-level sieve

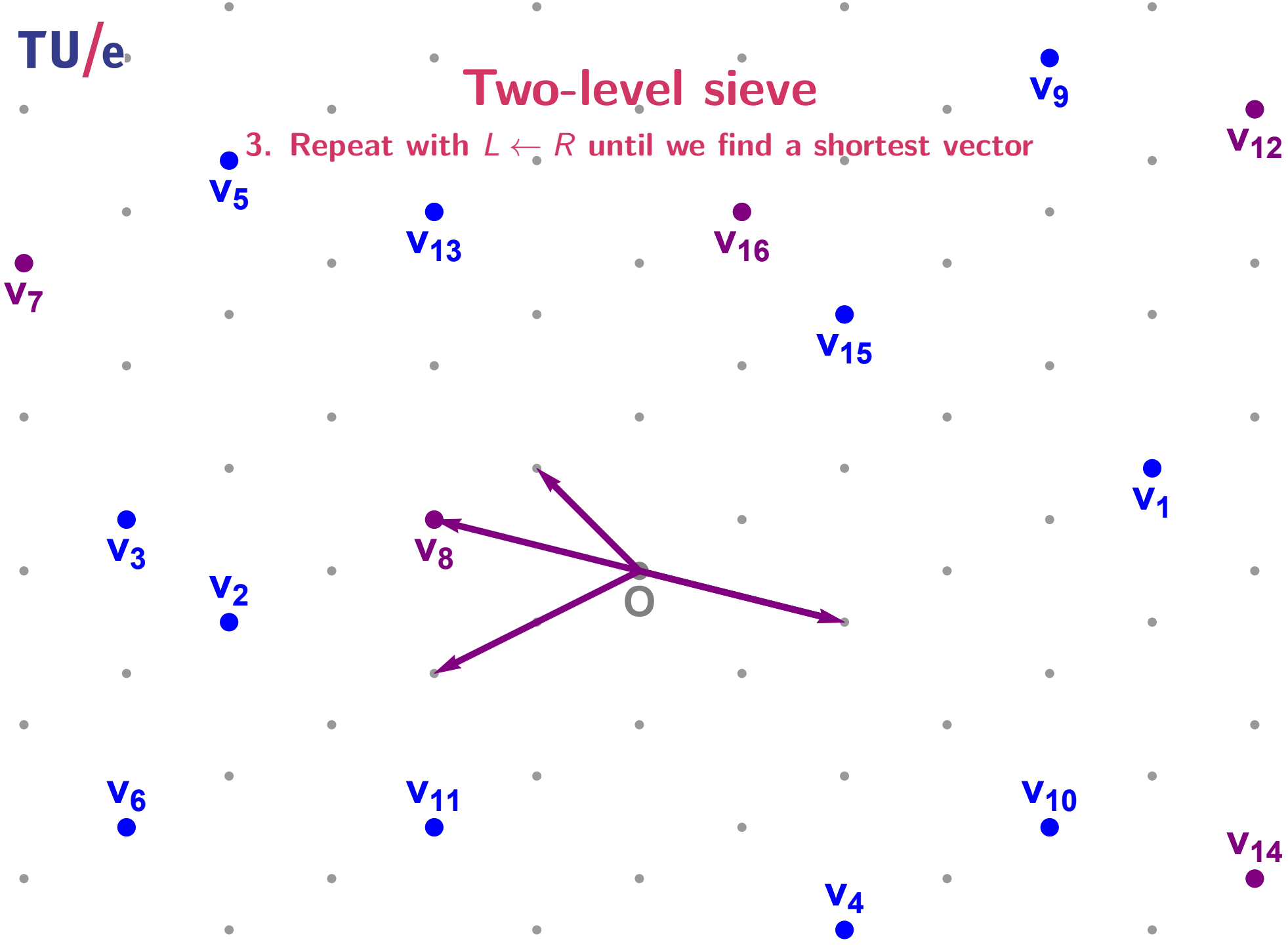
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Two-level sieve

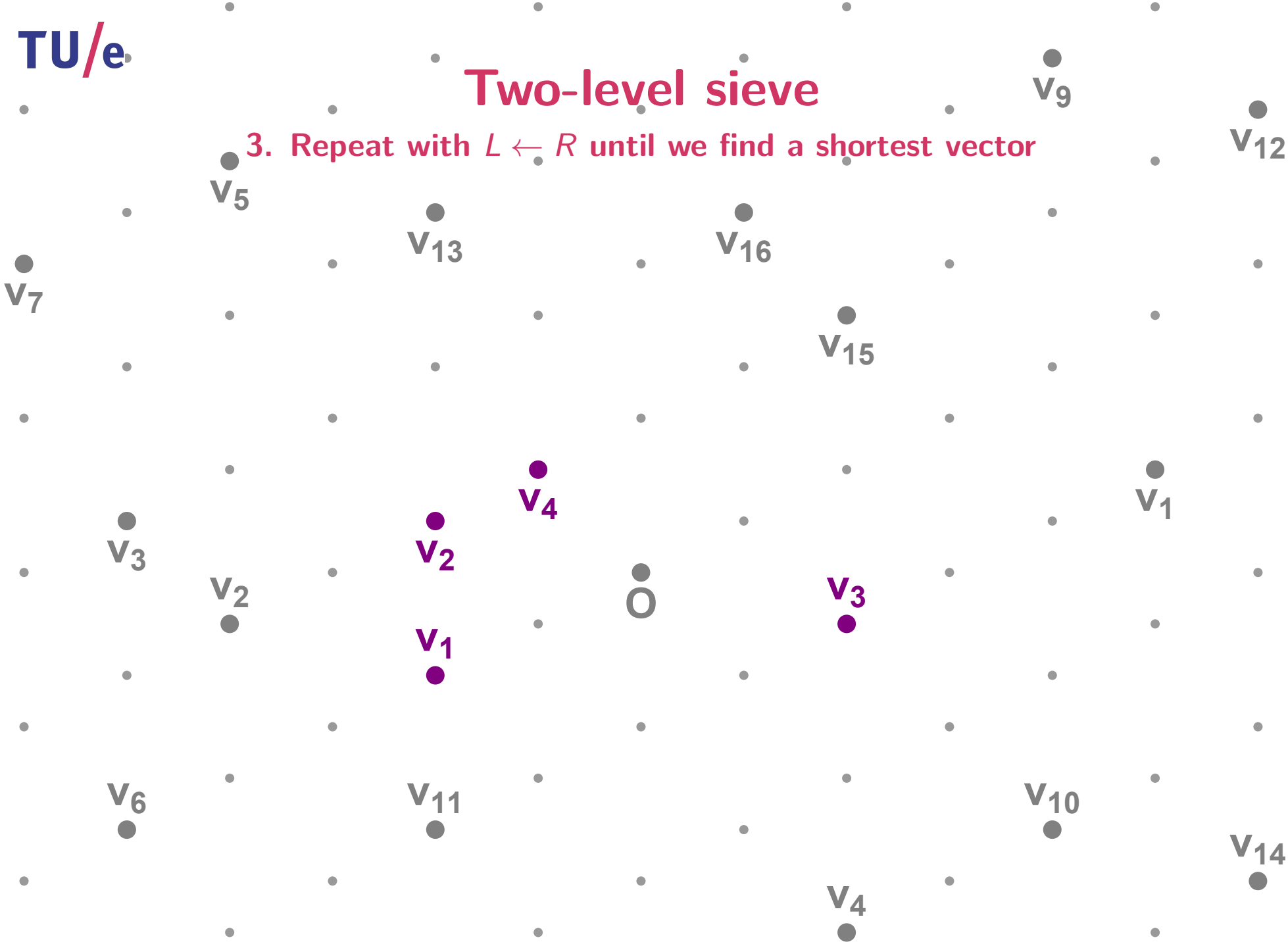
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Two-level sieve

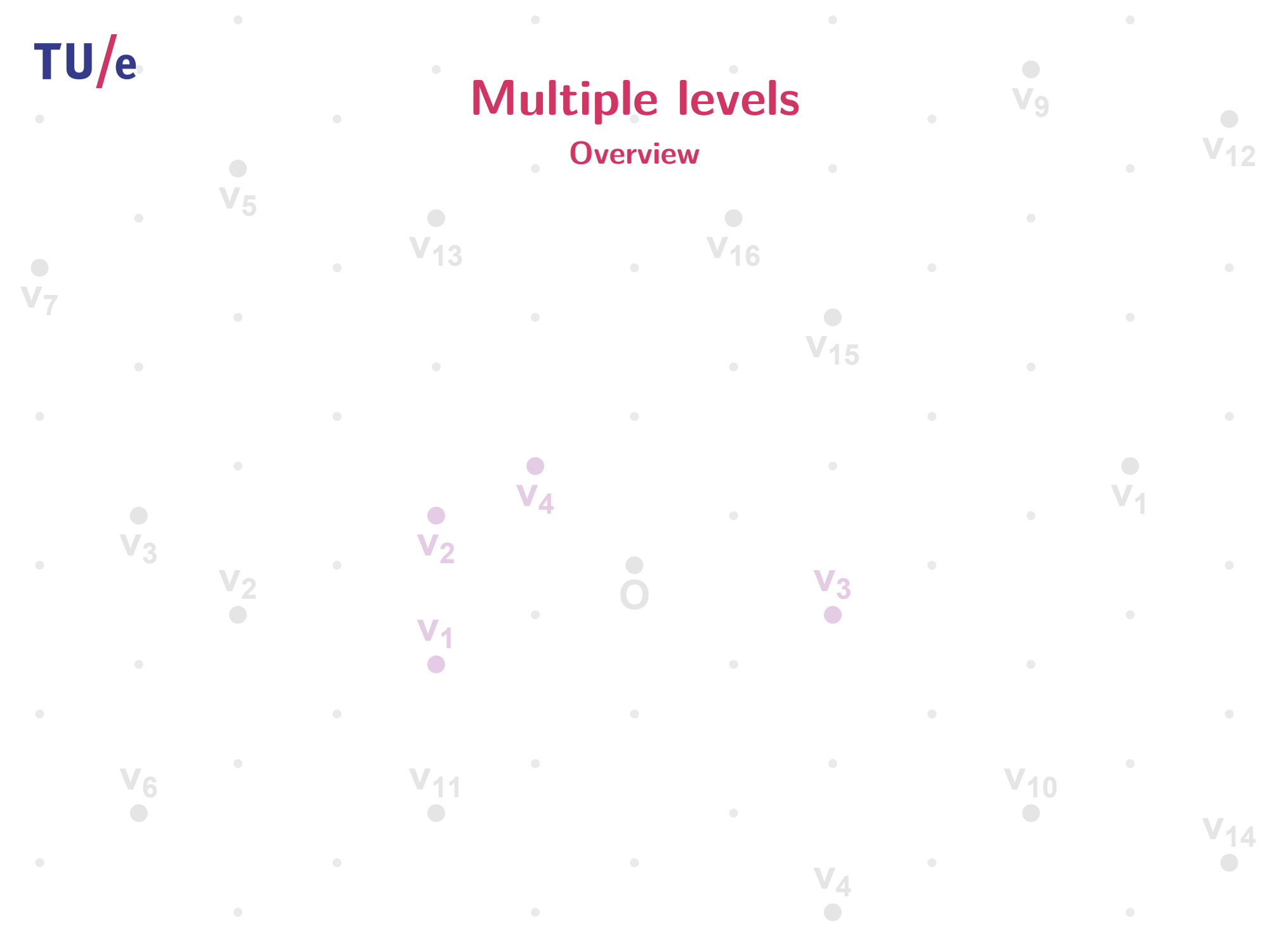
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Multiple levels

Overview



TU/e

Multiple levels

Overview

Heuristic (Nguyen and Vidick, J. Math. Crypt. '08)

The one-level sieve runs in time $2^{0.4150n}$ and space $2^{0.2075n}$.

TU/e

Multiple levels

Overview

Heuristic (Nguyen and Vidick, J. Math. Crypt. '08)

The one-level sieve runs in time $2^{0.4150n}$ and space $2^{0.2075n}$.

Heuristic (Wang et al., ASIACCS'11)

The two-level sieve runs in time $2^{0.3836n}$ and space $2^{0.2557n}$.

TU/e

Multiple levels

Overview

Heuristic (Nguyen and Vidick, J. Math. Crypt. '08)

The one-level sieve runs in time $2^{0.4150n}$ and space $2^{0.2075n}$.

Heuristic (Wang et al., ASIACCS'11)

The two-level sieve runs in time $2^{0.3836n}$ and space $2^{0.2557n}$.

Heuristic (Zhang et al., SAC'13)

The three-level sieve runs in time $2^{0.3778n}$ and space $2^{0.2833n}$.

TU/e

Multiple levels

Overview

Heuristic (Nguyen and Vidick, J. Math. Crypt. '08)

The one-level sieve runs in time $2^{0.4150n}$ and space $2^{0.2075n}$.

Heuristic (Wang et al., ASIACCS'11)

The two-level sieve runs in time $2^{0.3836n}$ and space $2^{0.2557n}$.

Heuristic (Zhang et al., SAC'13)

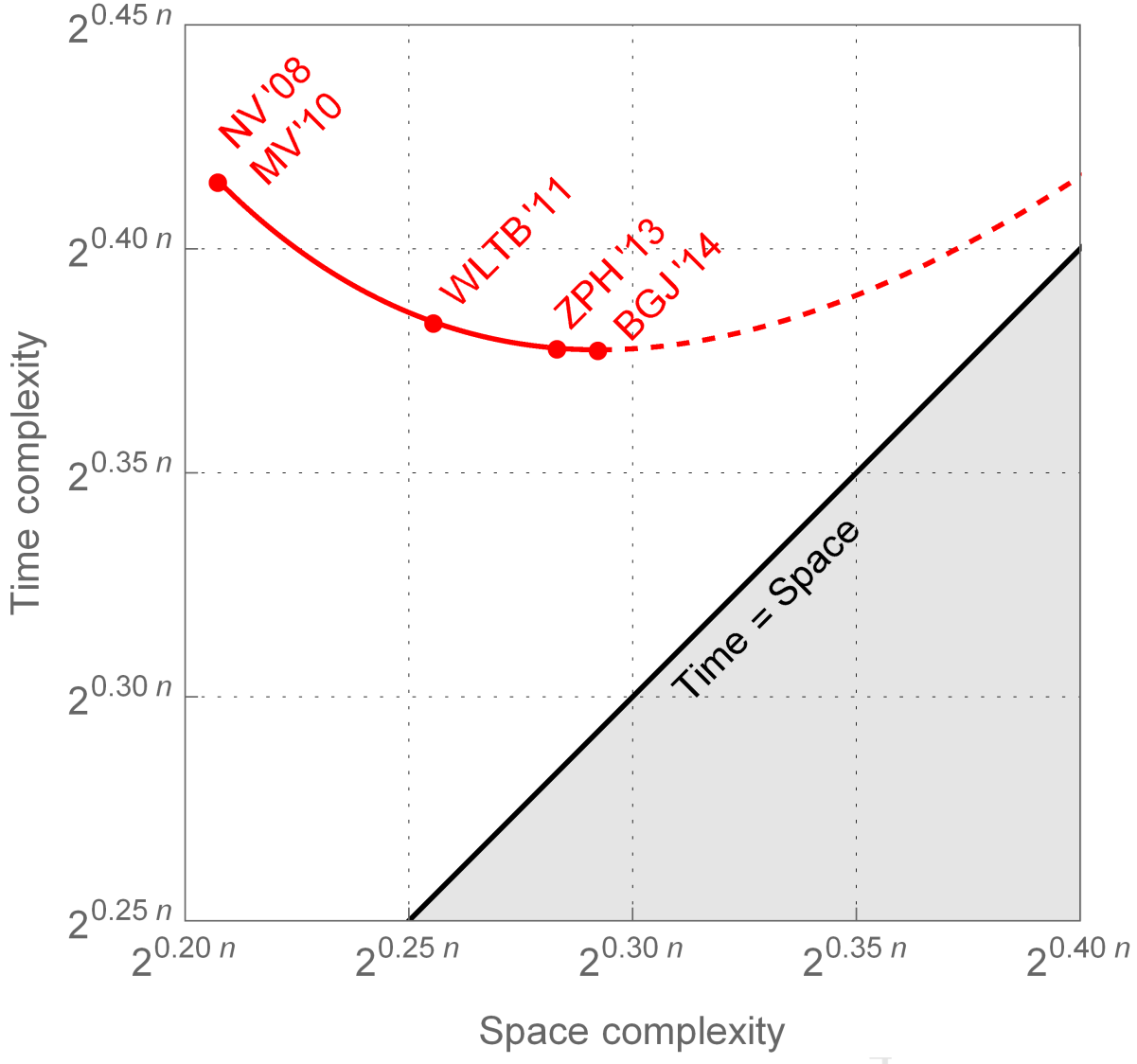
The three-level sieve runs in time $2^{0.3778n}$ and space $2^{0.2833n}$.

Conjecture

The four-level sieve runs in time $2^{0.3774n}$ and space $2^{0.2925n}$, and higher-level sieves are not faster than this.

Sieving

Space/time trade-off



TU/e

Quantum Search

Classical form

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

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

TU/e

Quantum Search

General form

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

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

TU/e

Quantum Search

General form

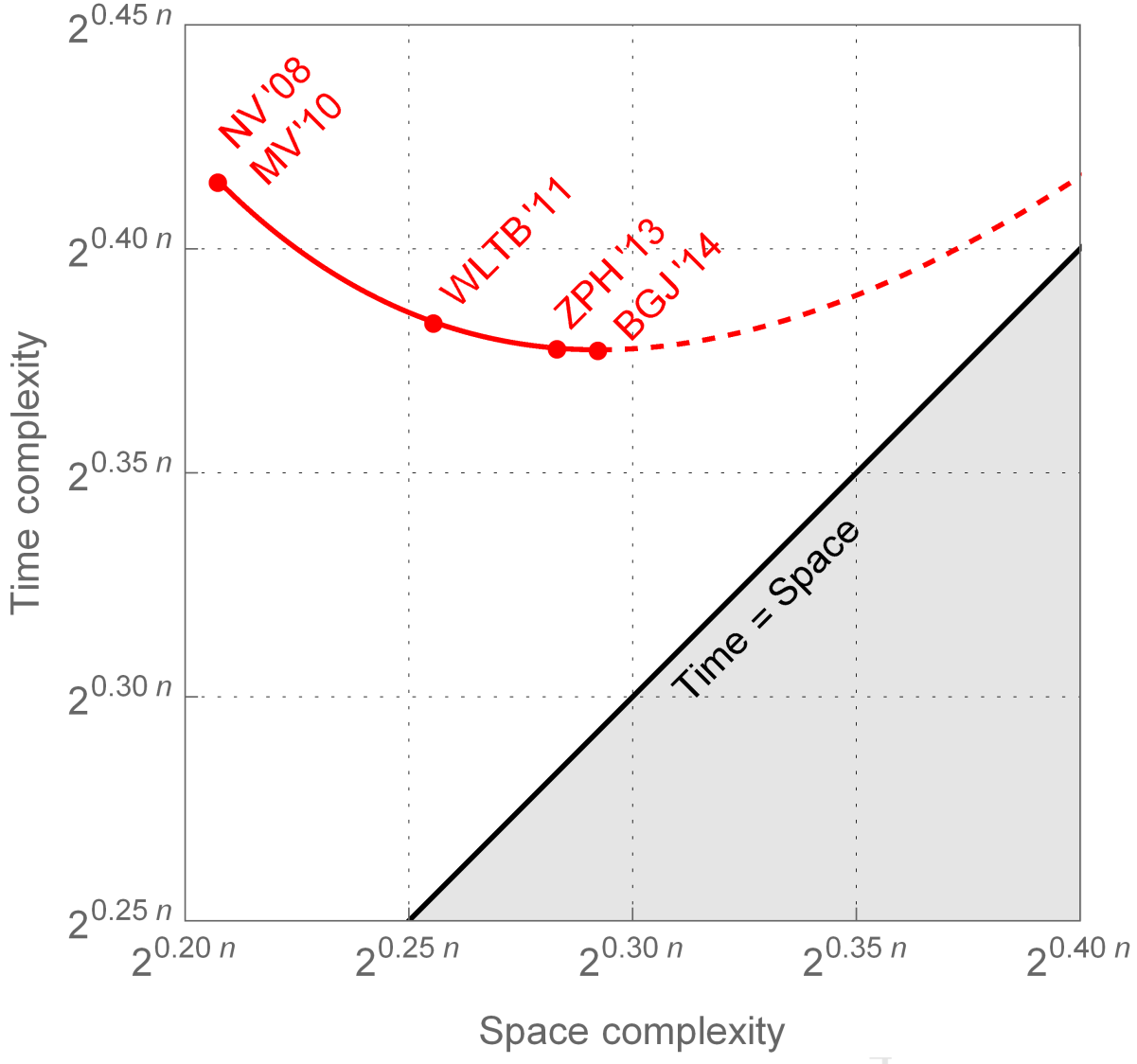
Problem: Given a list L of size N, and a function $f : L \to \{0, 1\}$ such that there are $c = O(1)$ elements $e \in L$ with $f(e) = 1$. Find one such element e.

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

Potentially speed up search subroutines in sieving

Sieving

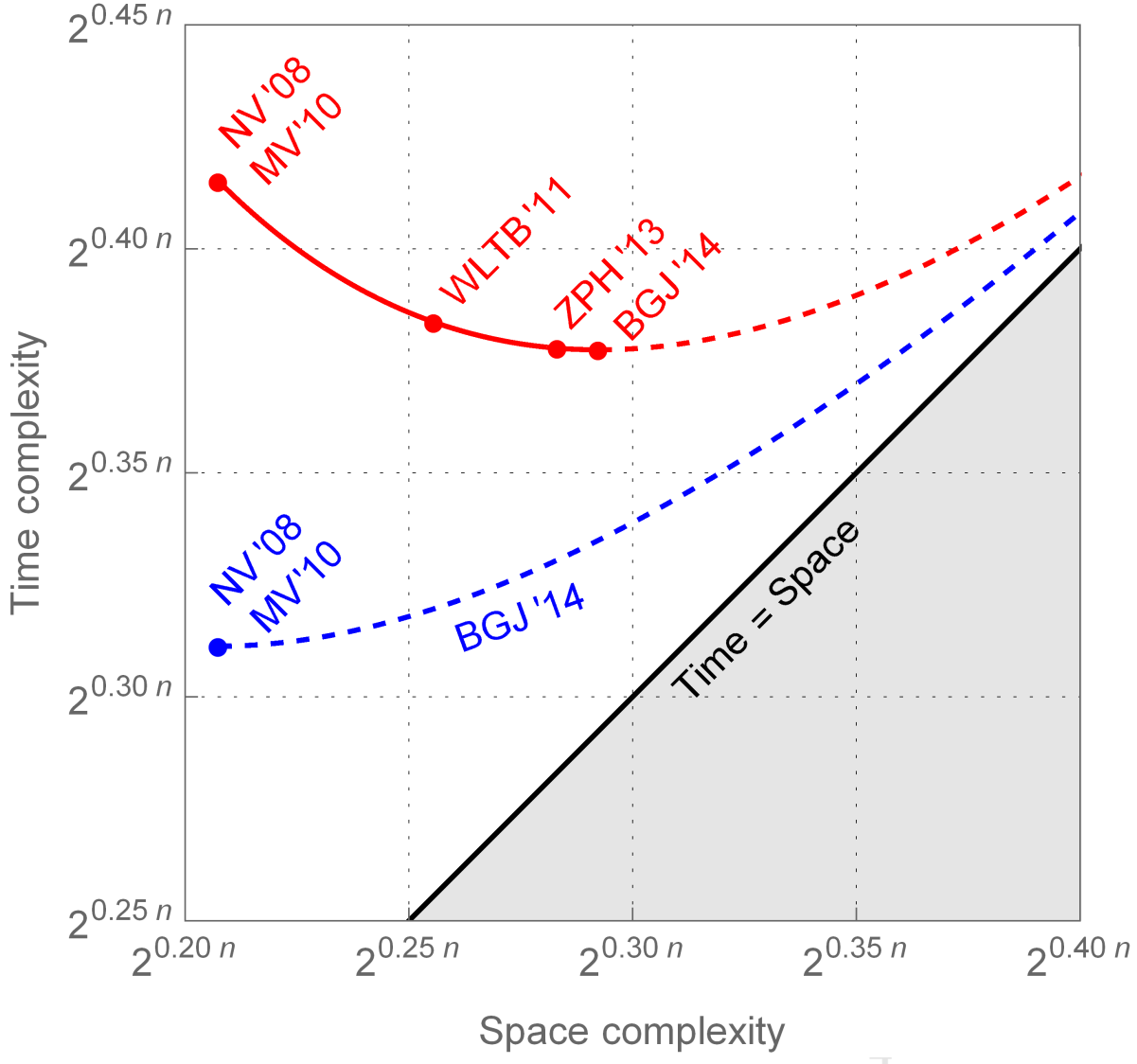
Space/time trade-off



TU/e

Sieving

Space/time trade-off



TU/e

Nguyen-Vidick sieve with LSH

1. Sample a list L of random lattice vectors

TU/e

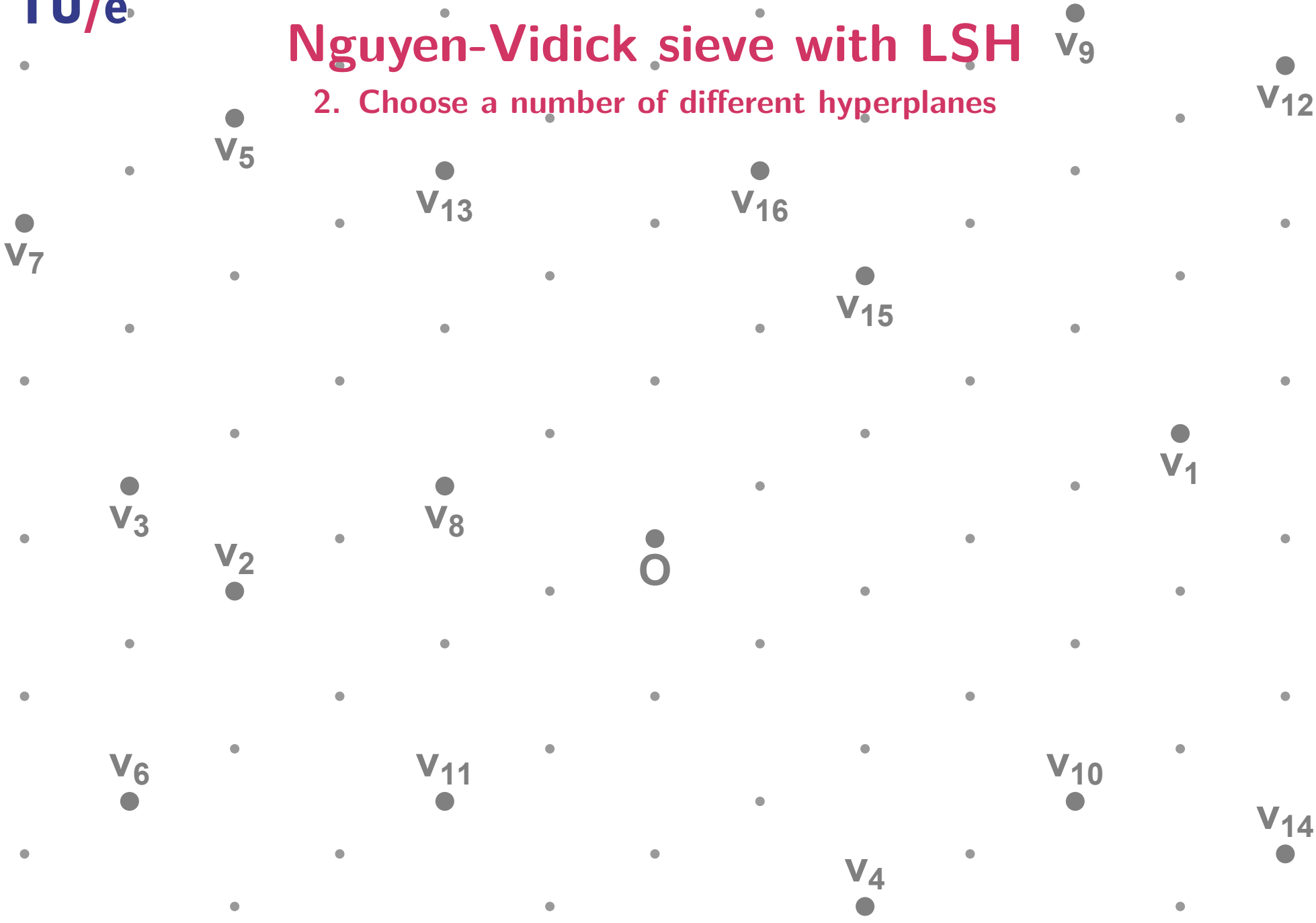
Nguyen-Vidick sieve with LSH

1. Sample a list L of random lattice vectors

TU/e

Nguyen-Vidick sieve with LSH

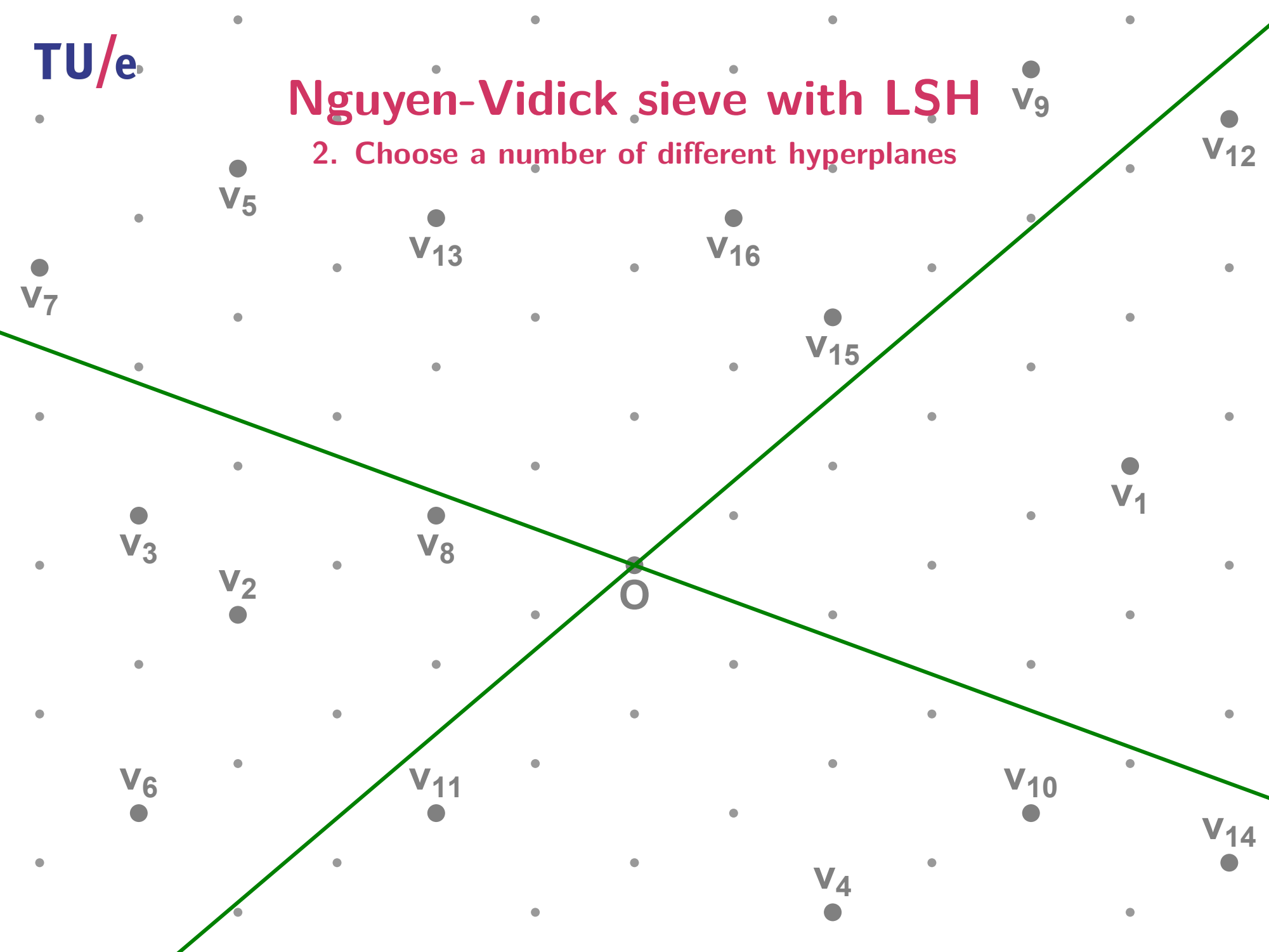
2. Choose a number of different hyperplanes



TU/e

Nguyen-Vidick sieve with LSH

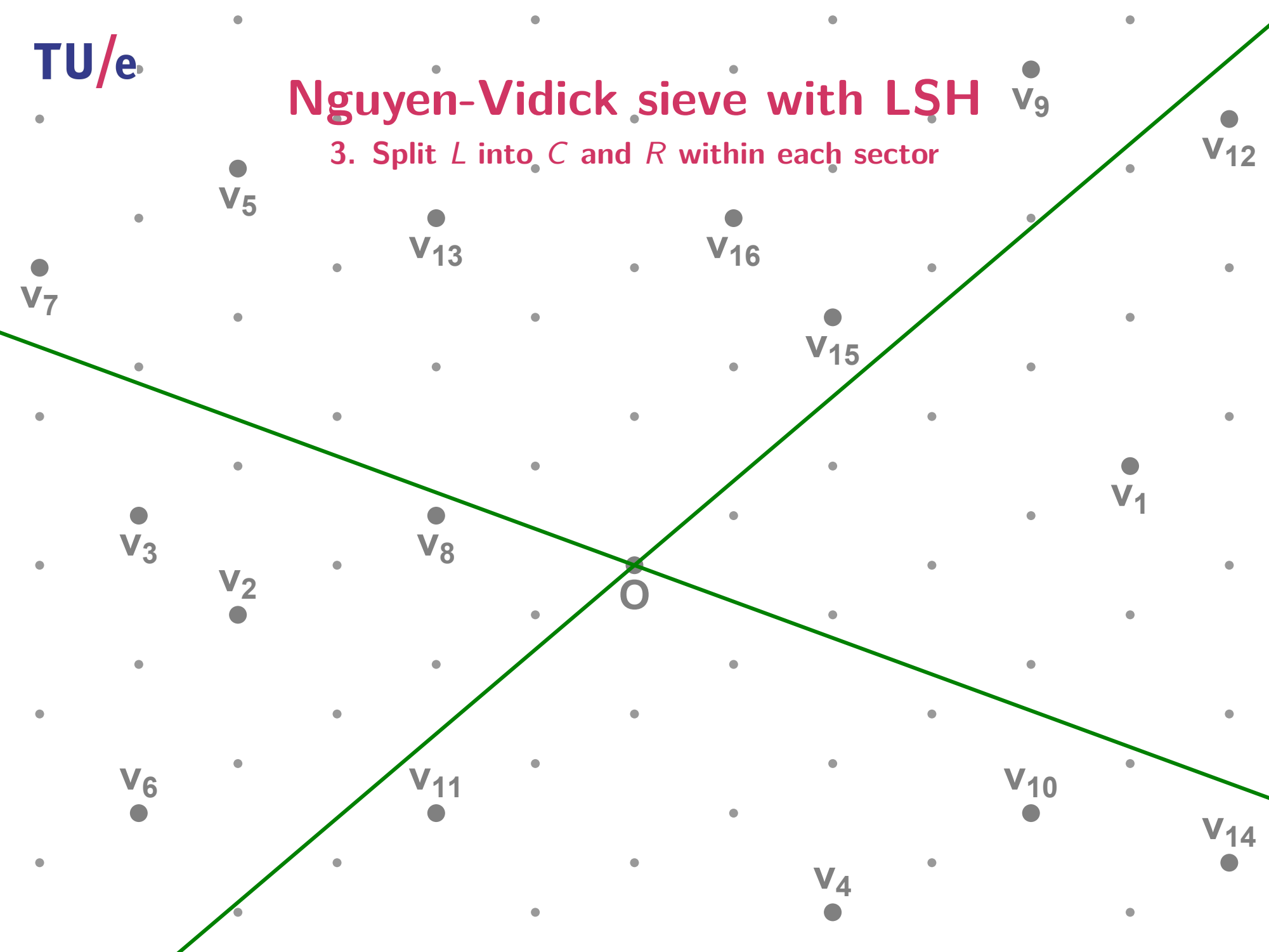
2. Choose a number of different hyperplanes



TU/e

Nguyen-Vidick sieve with LSH

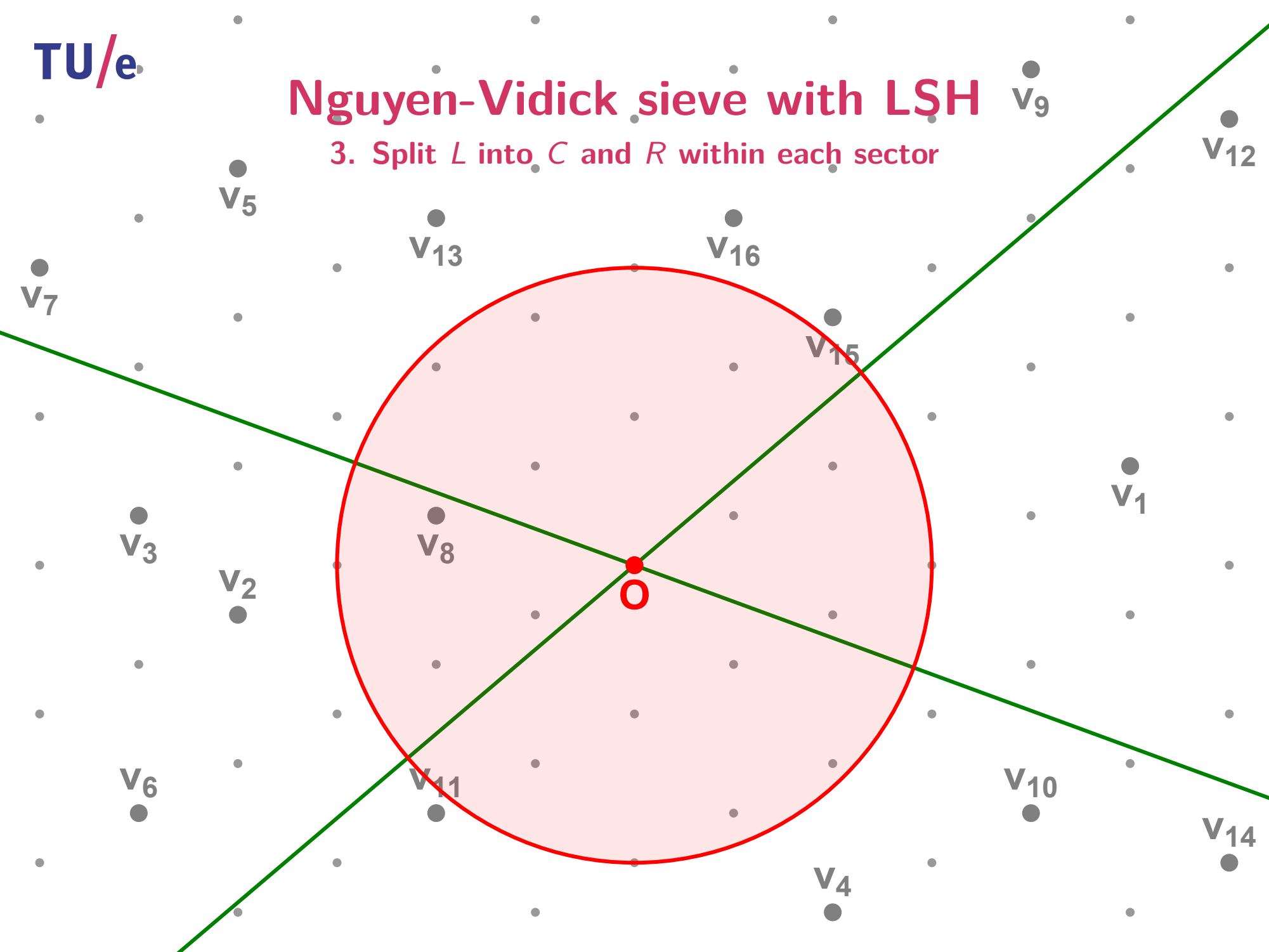
3. Split L into C and R within each sector



TU/e

Nguyen-Vidick sieve with LSH

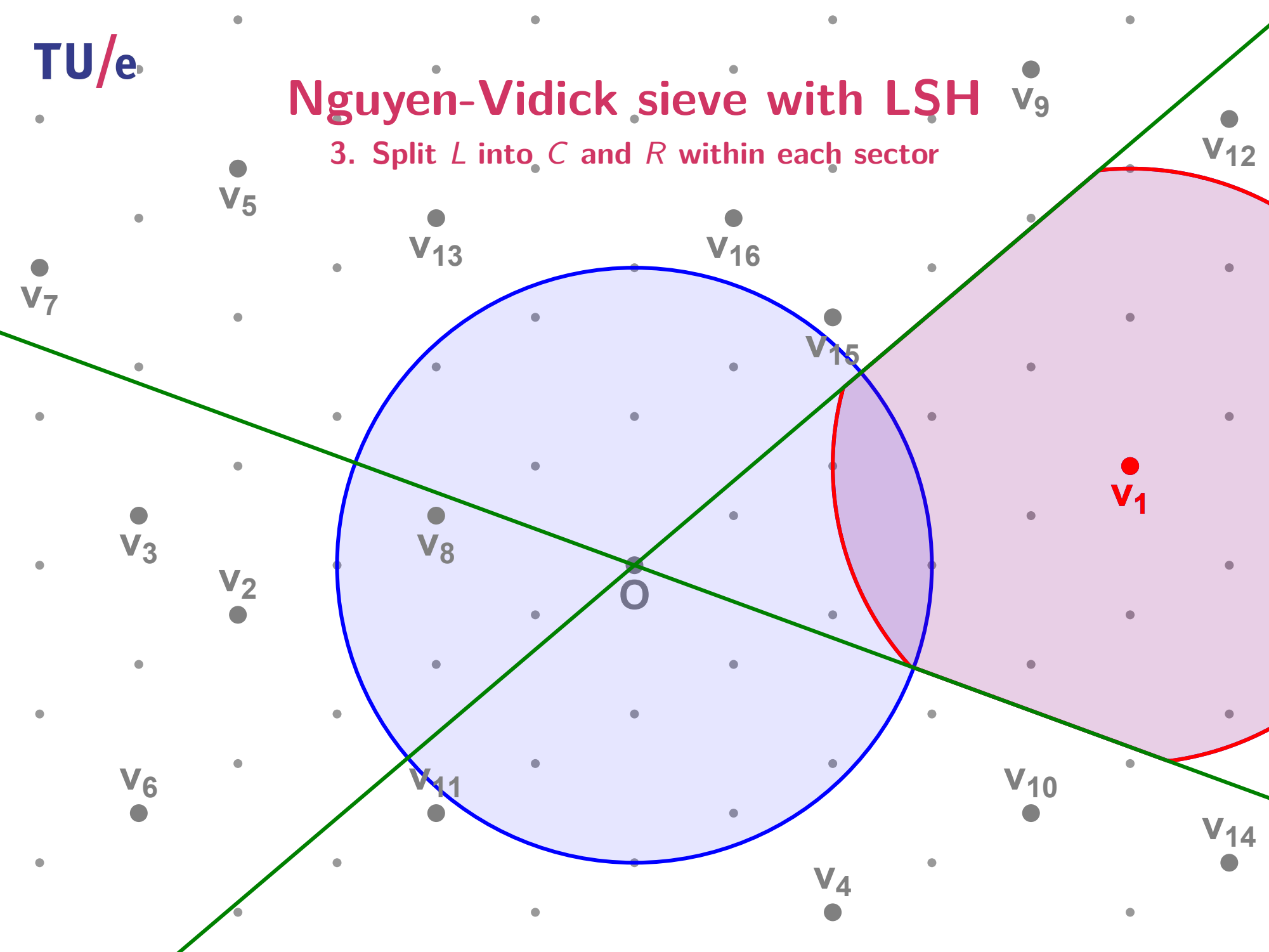
3. Split L into C and R within each sector



TU/e

Nguyen-Vidick sieve with LSH

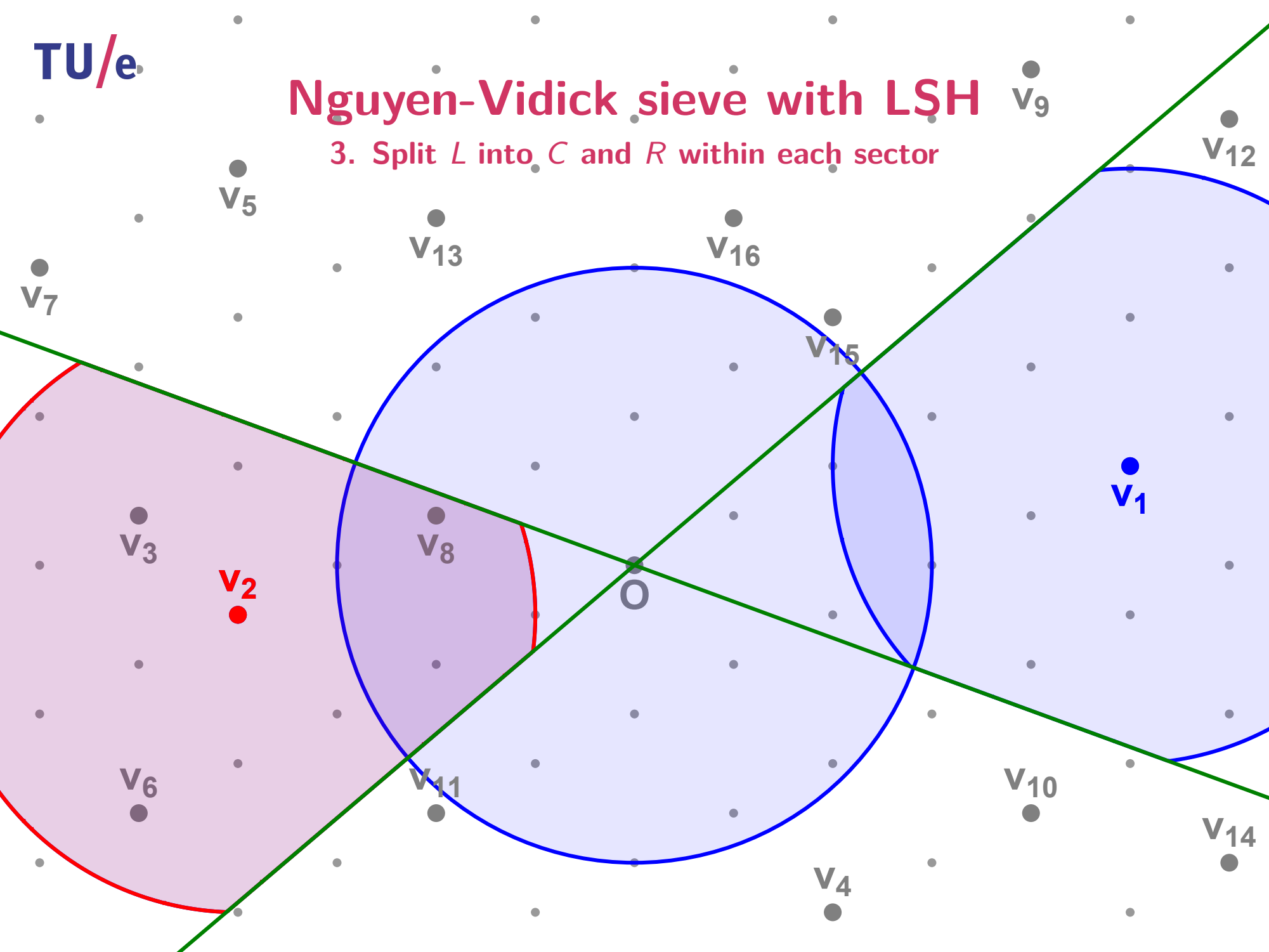
3. Split L into C and R within each sector



TU/e

Nguyen-Vidick sieve with LSH

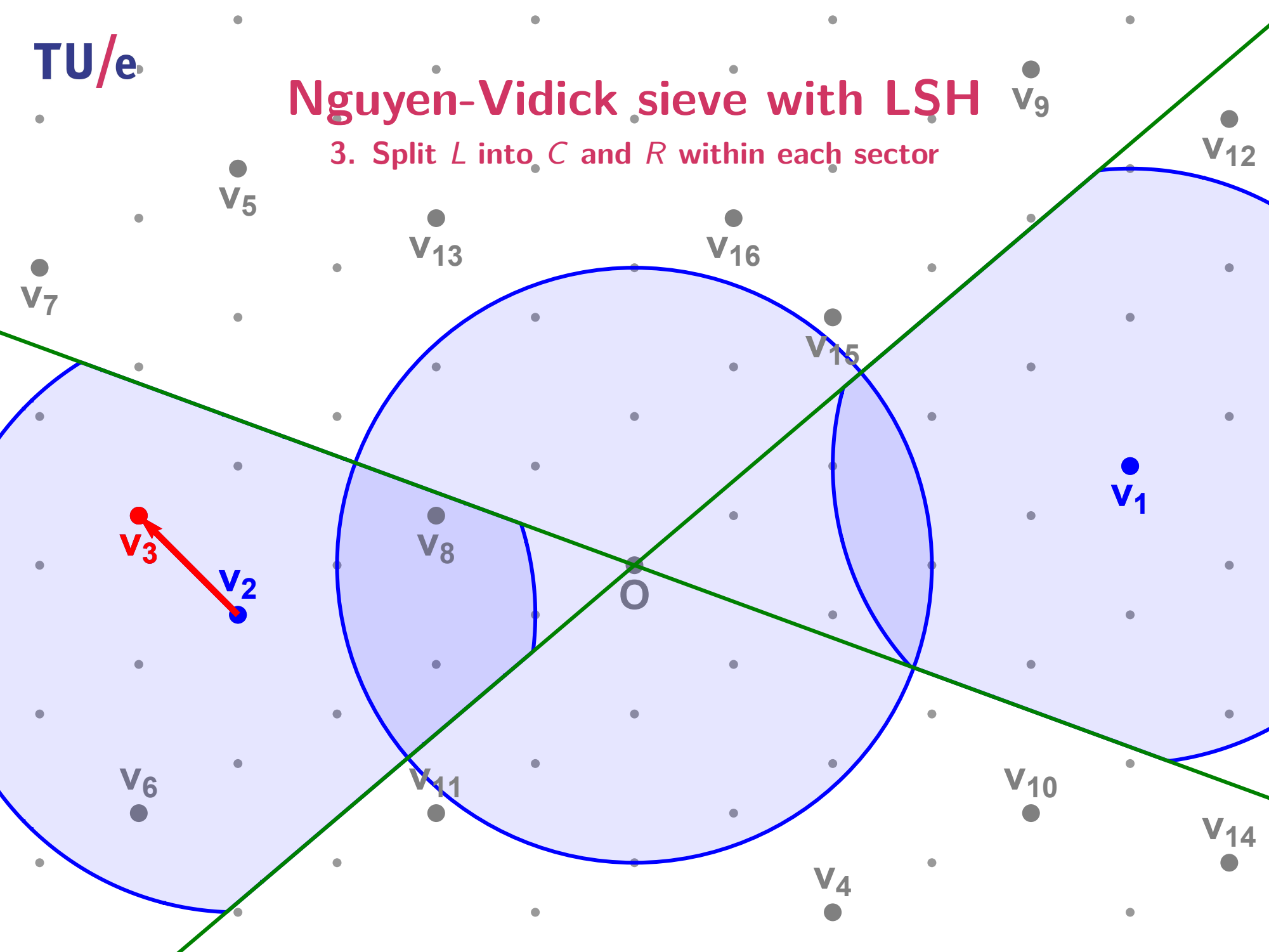
3. Split L into C and R within each sector



TU/e

Nguyen-Vidick sieve with LSH

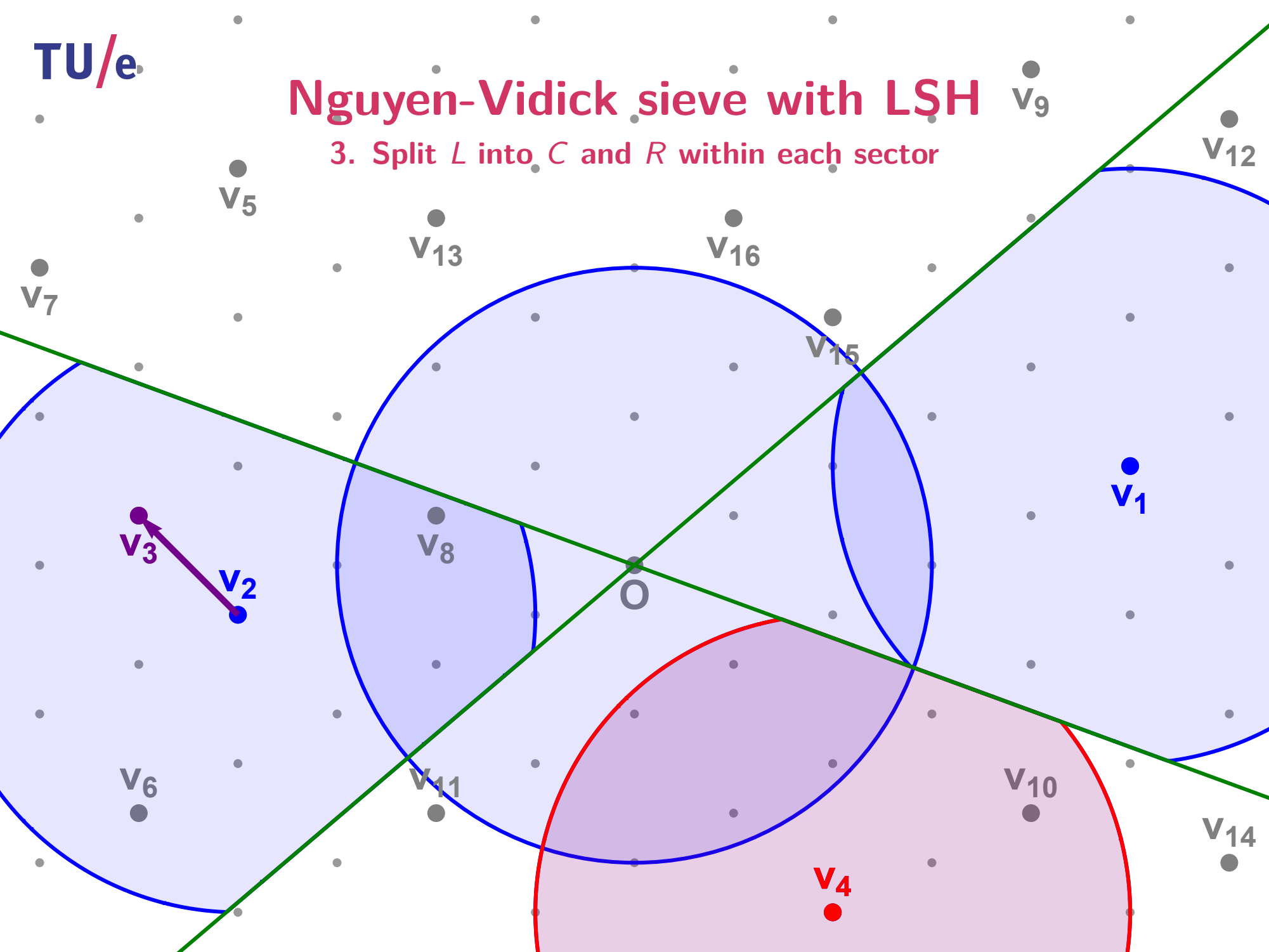
3. Split L into C and R within each sector

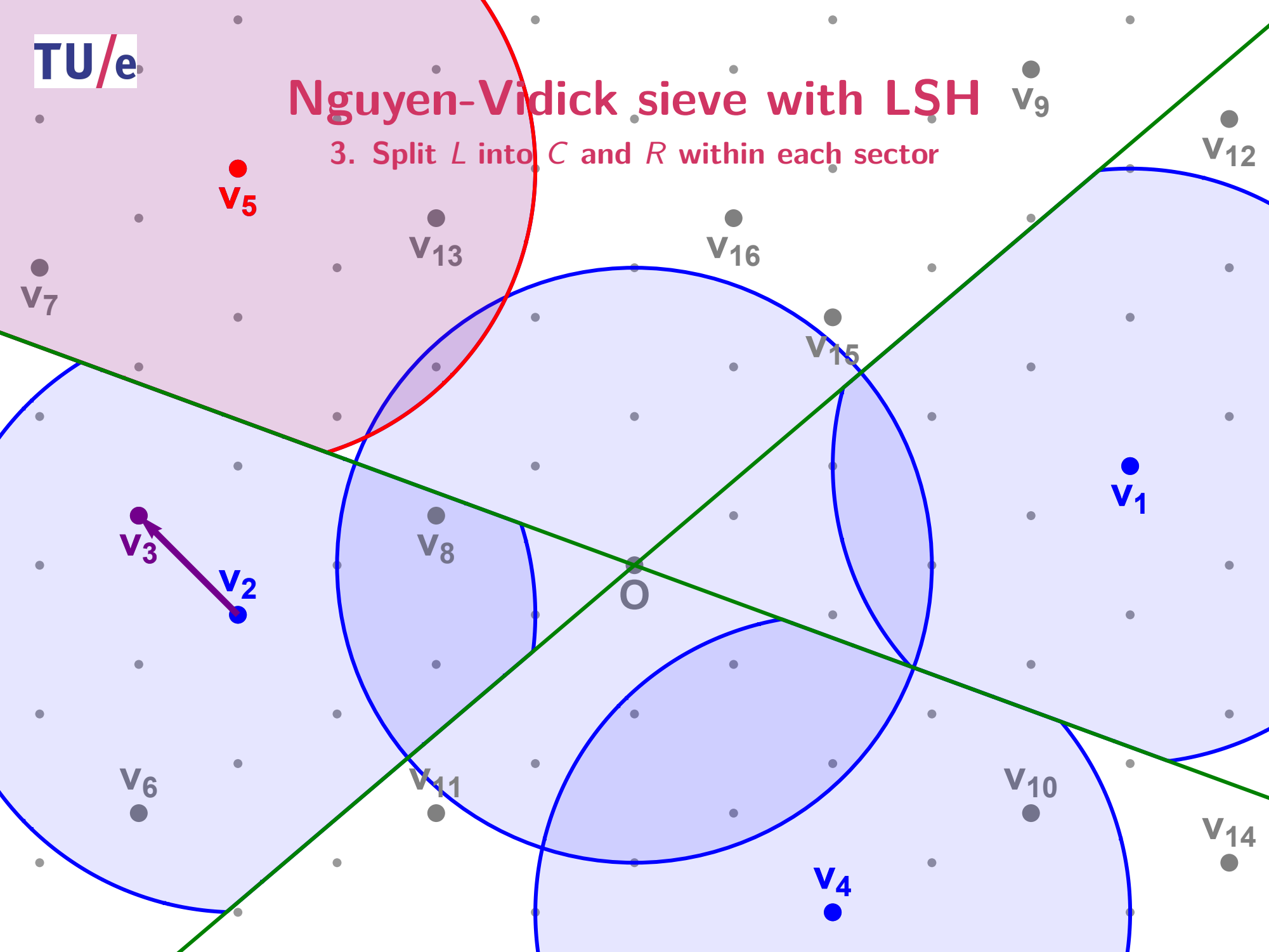


TU/e

Nguyen-Vidick sieve with LSH

3. Split L into C and R within each sector

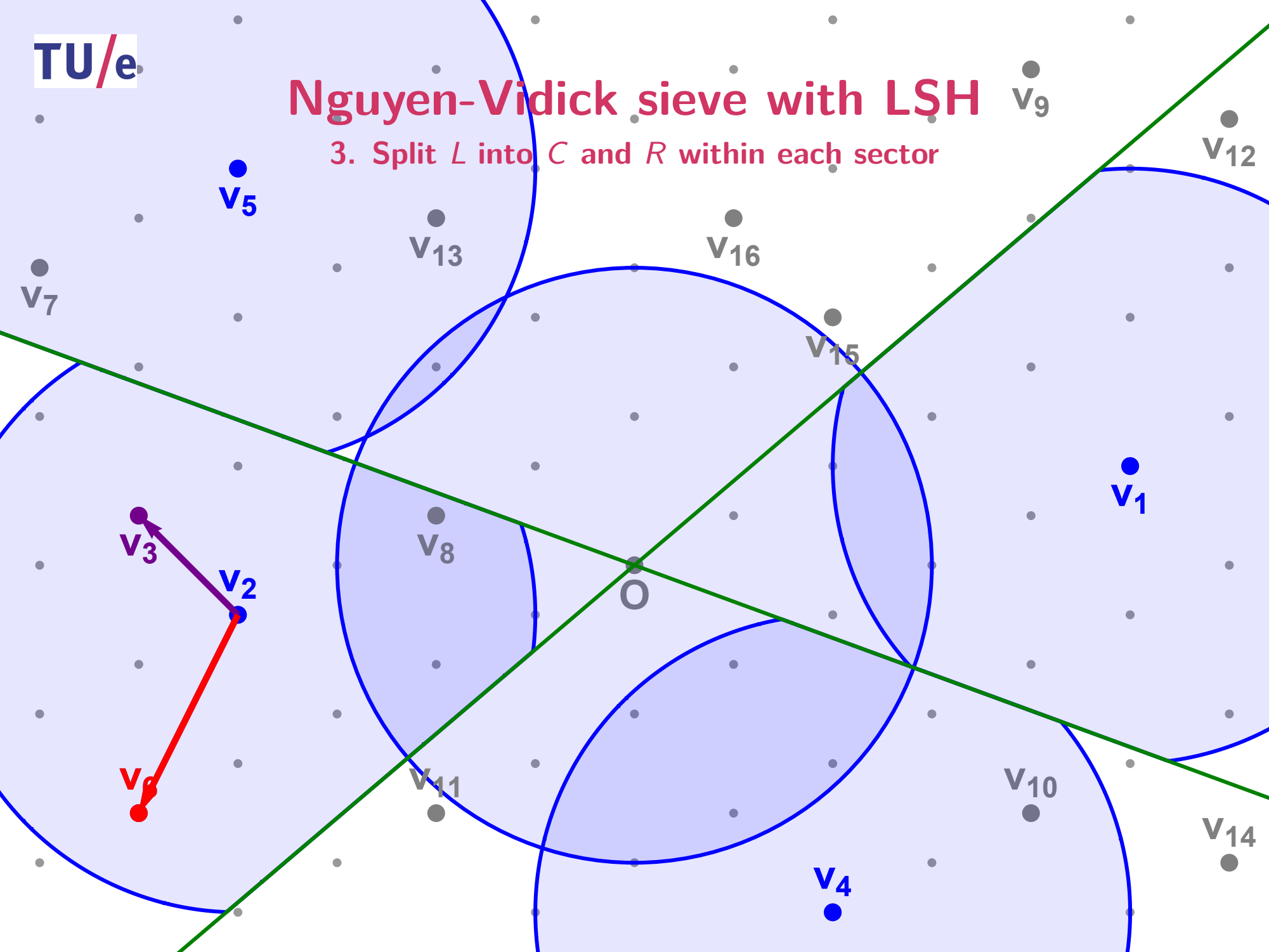



TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
v_5
v_{13}
v_{16}
v_9
v_{12}
v_7
v_{15}
v_1
v_3
v_2
v_8
O
v_6
v_{11}
v_{10}
v_{14}
v_4

TU/e

Nguyen-Vidick sieve with LSH

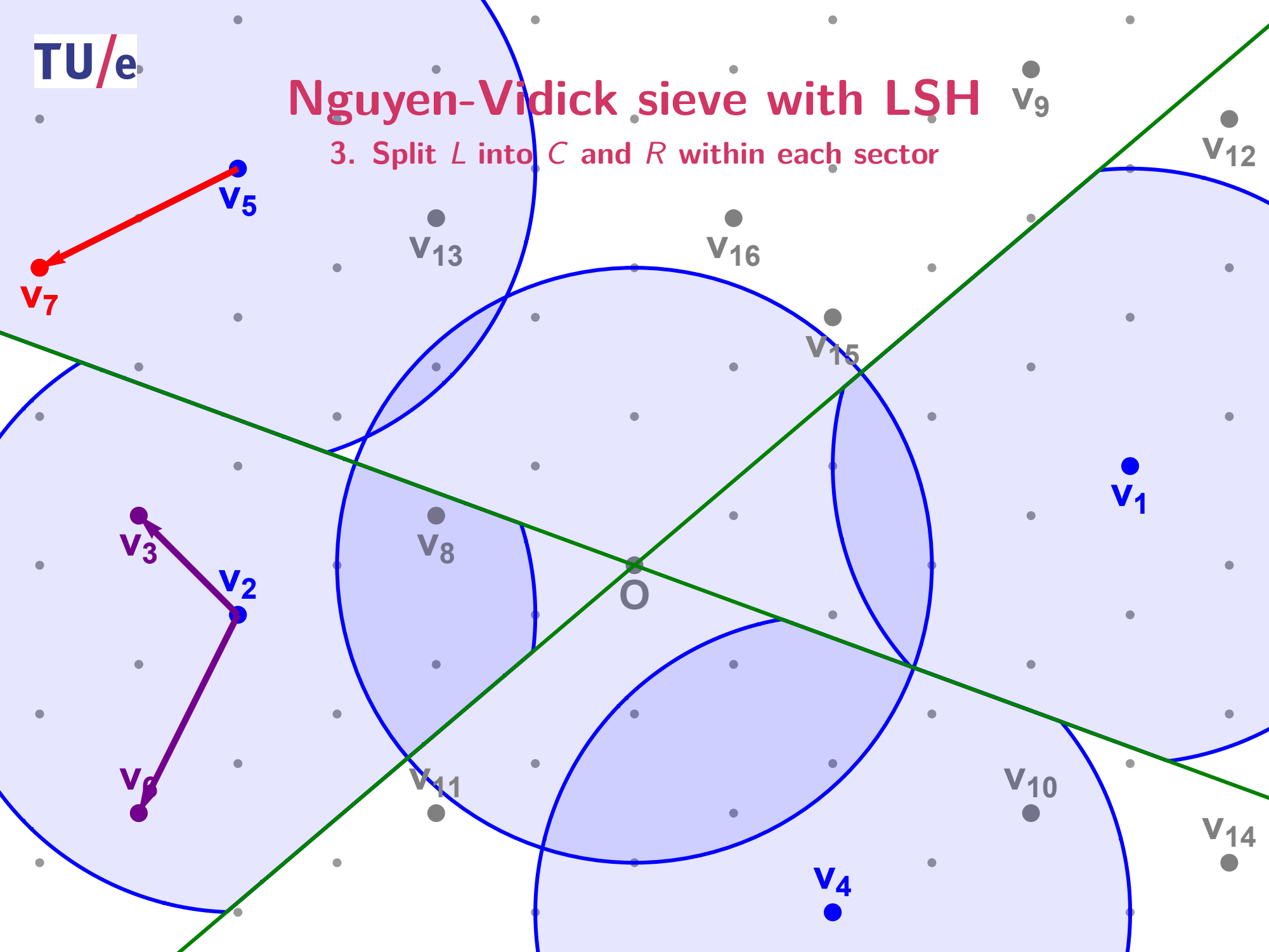
3. Split L into C and R within each sector



TU/e

Nguyen-Vidick sieve with LSH

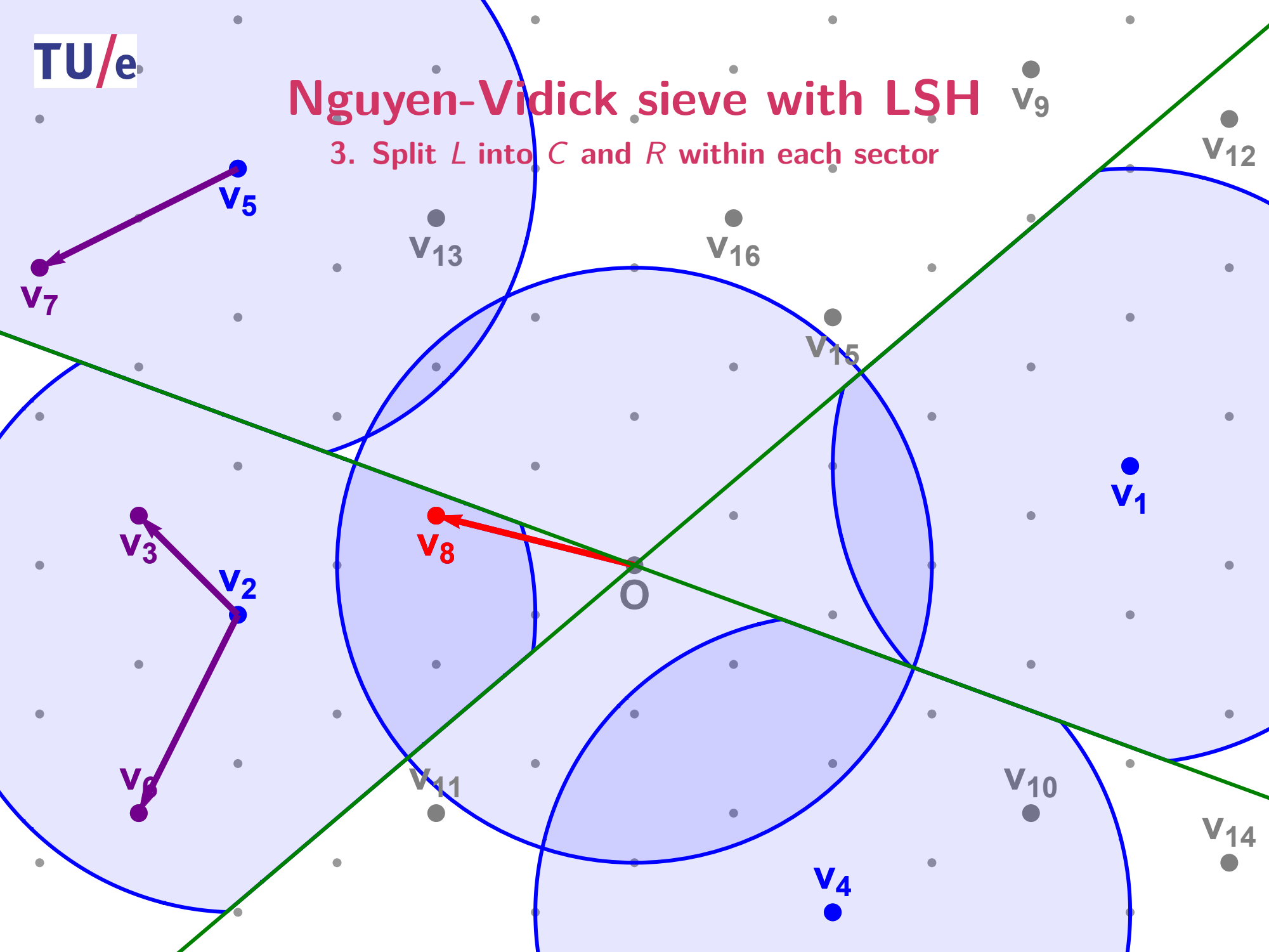
3. Split L into C and R within each sector

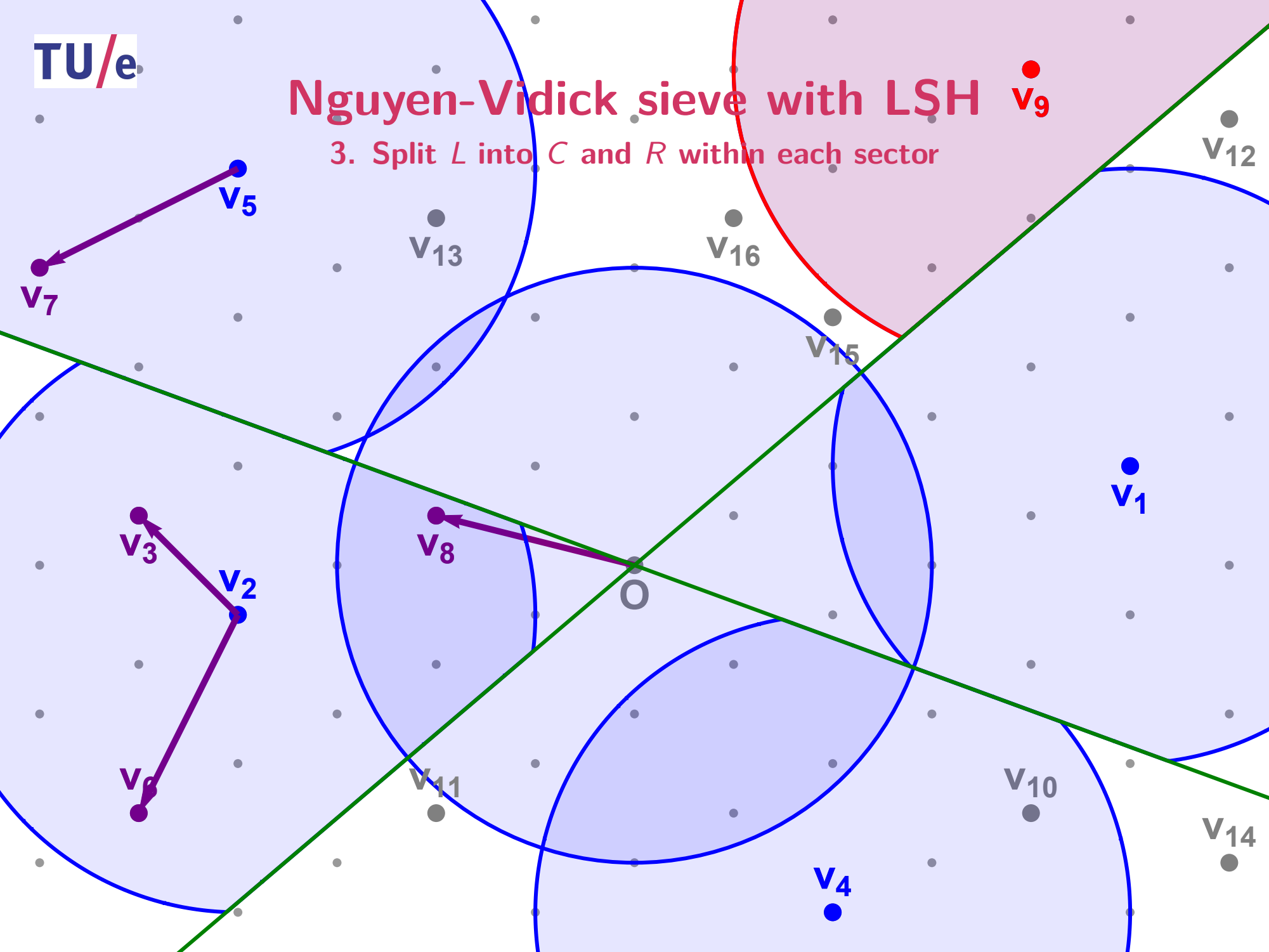


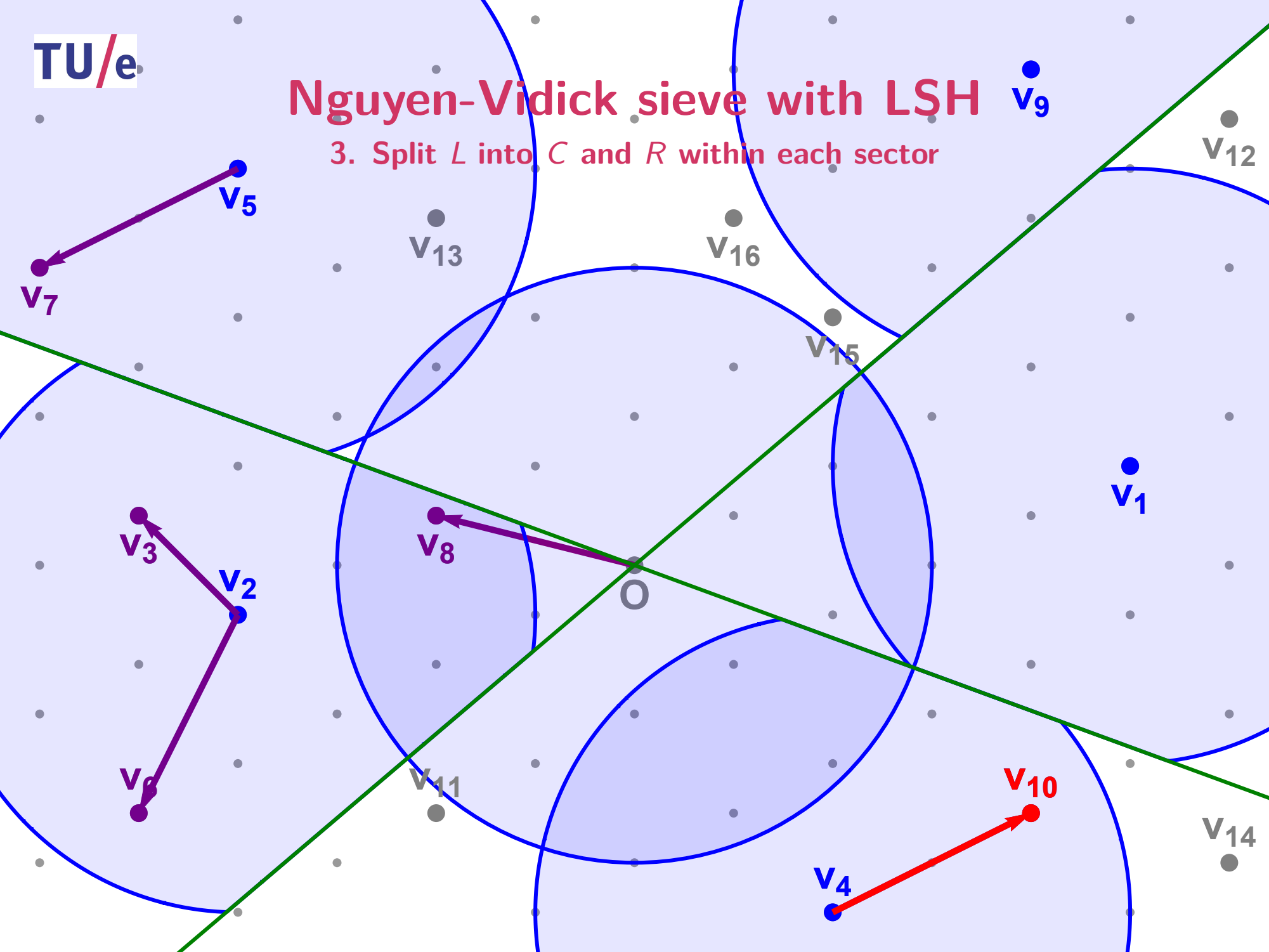
TU/e

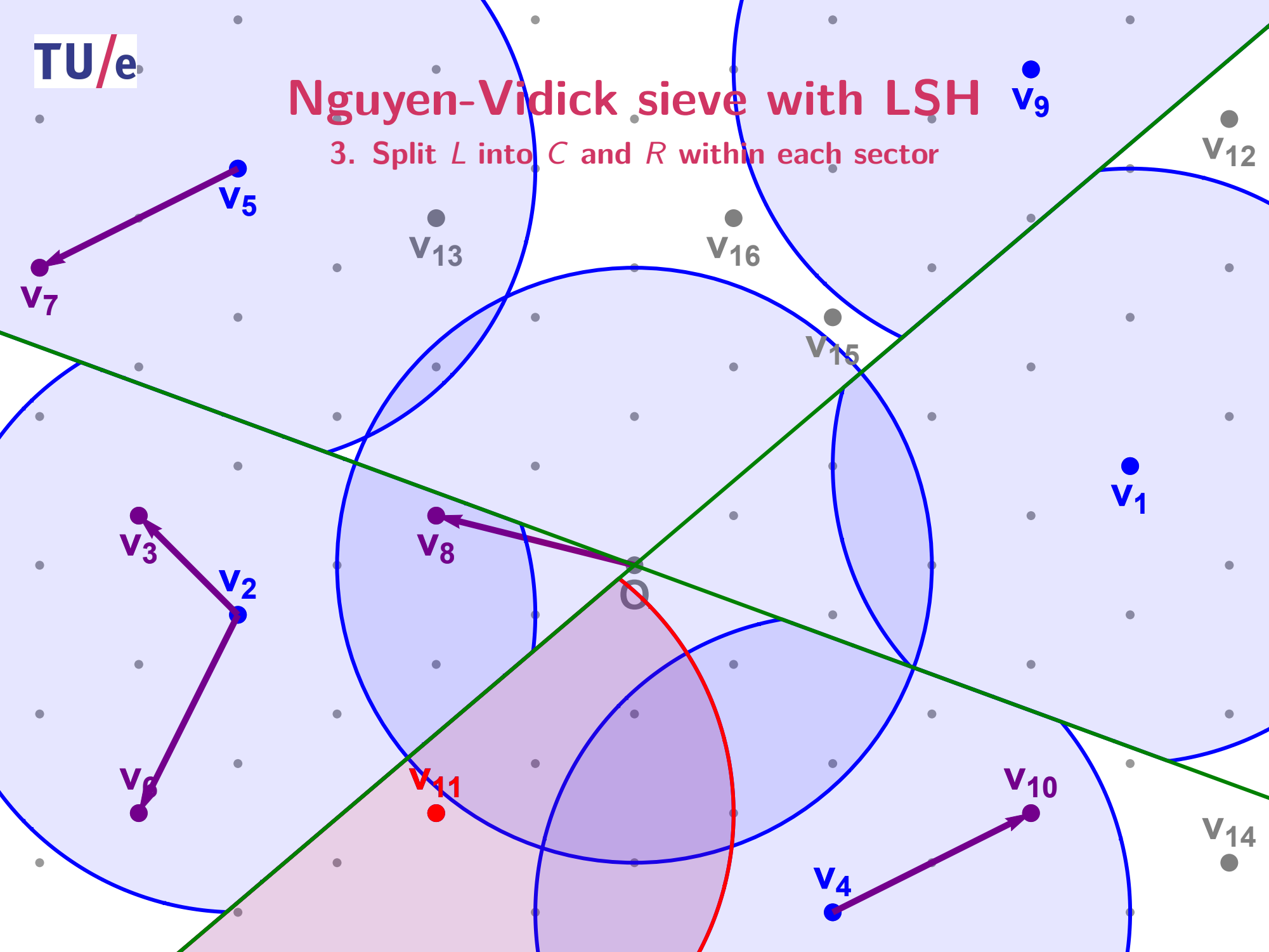
Nguyen-Vidick sieve with LSH

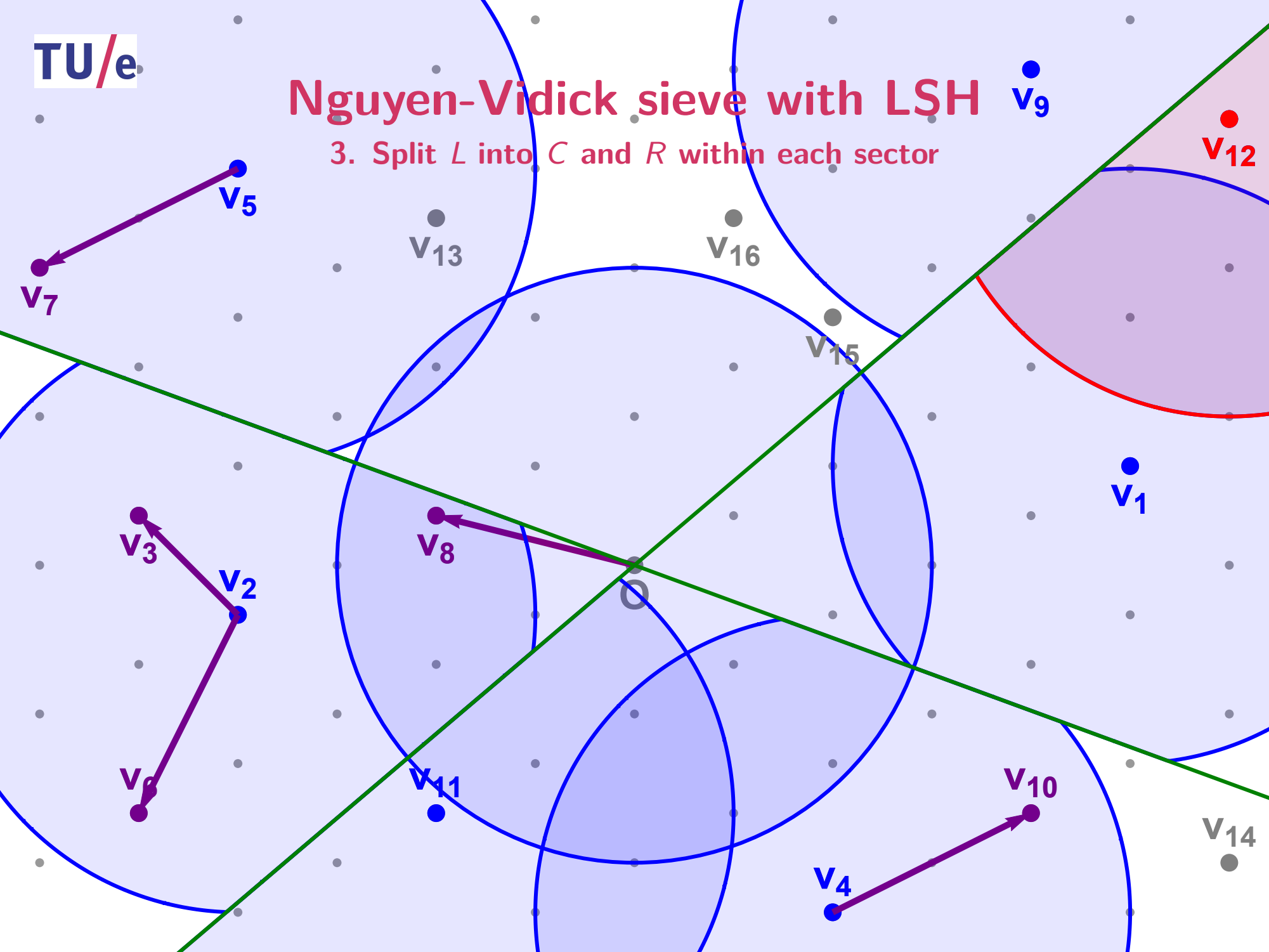
3. Split L into C and R within each sector

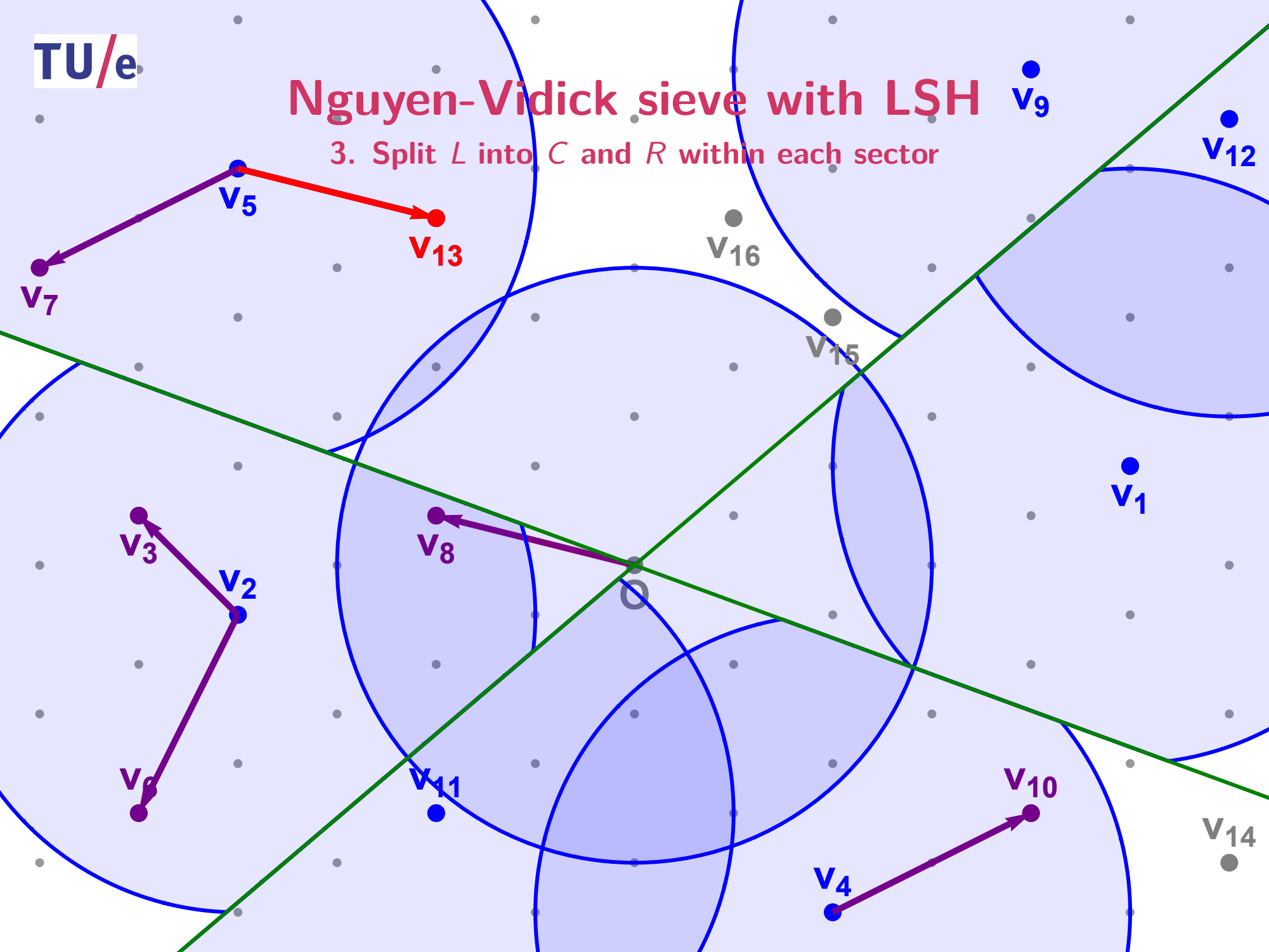


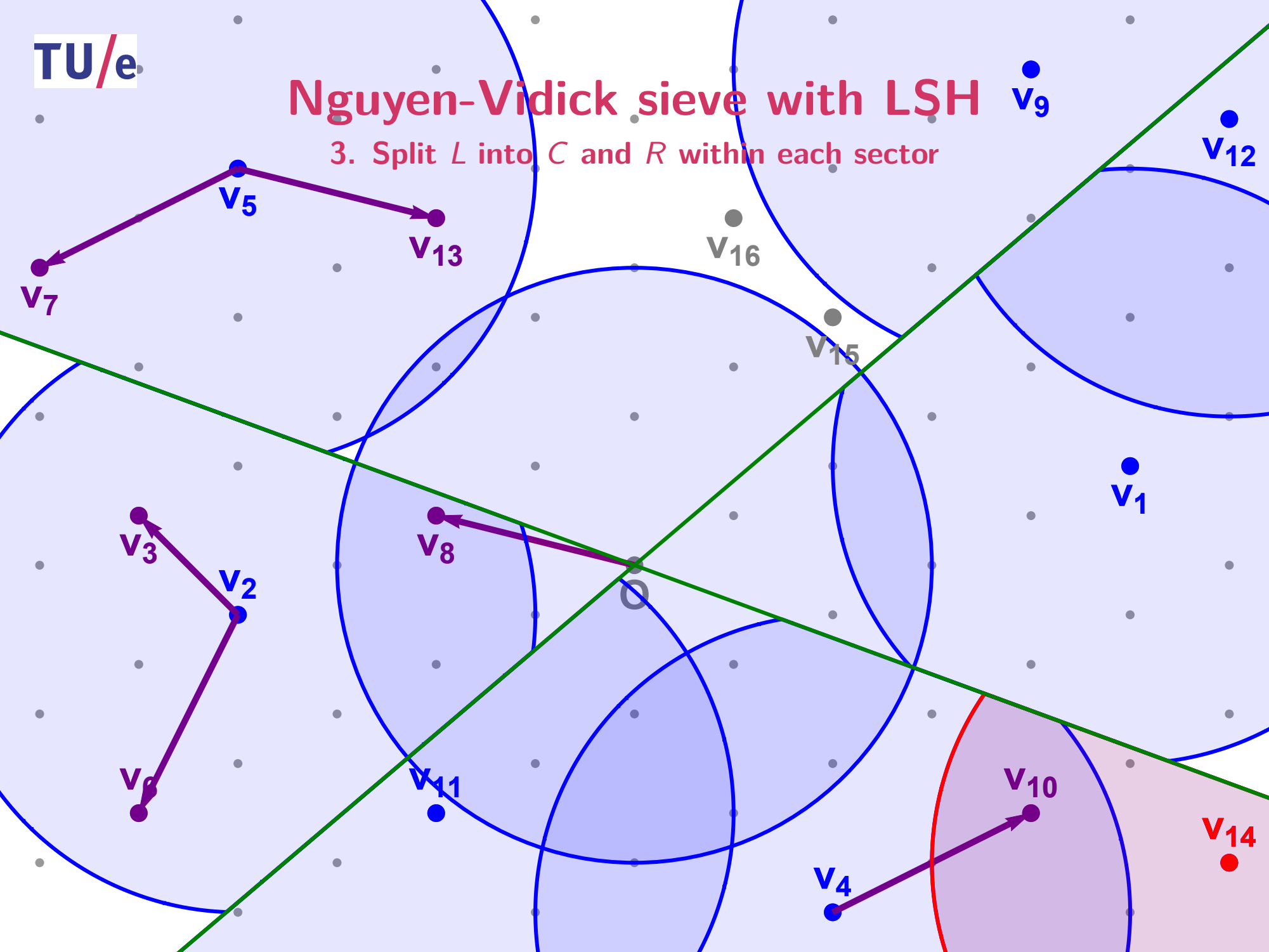

TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4

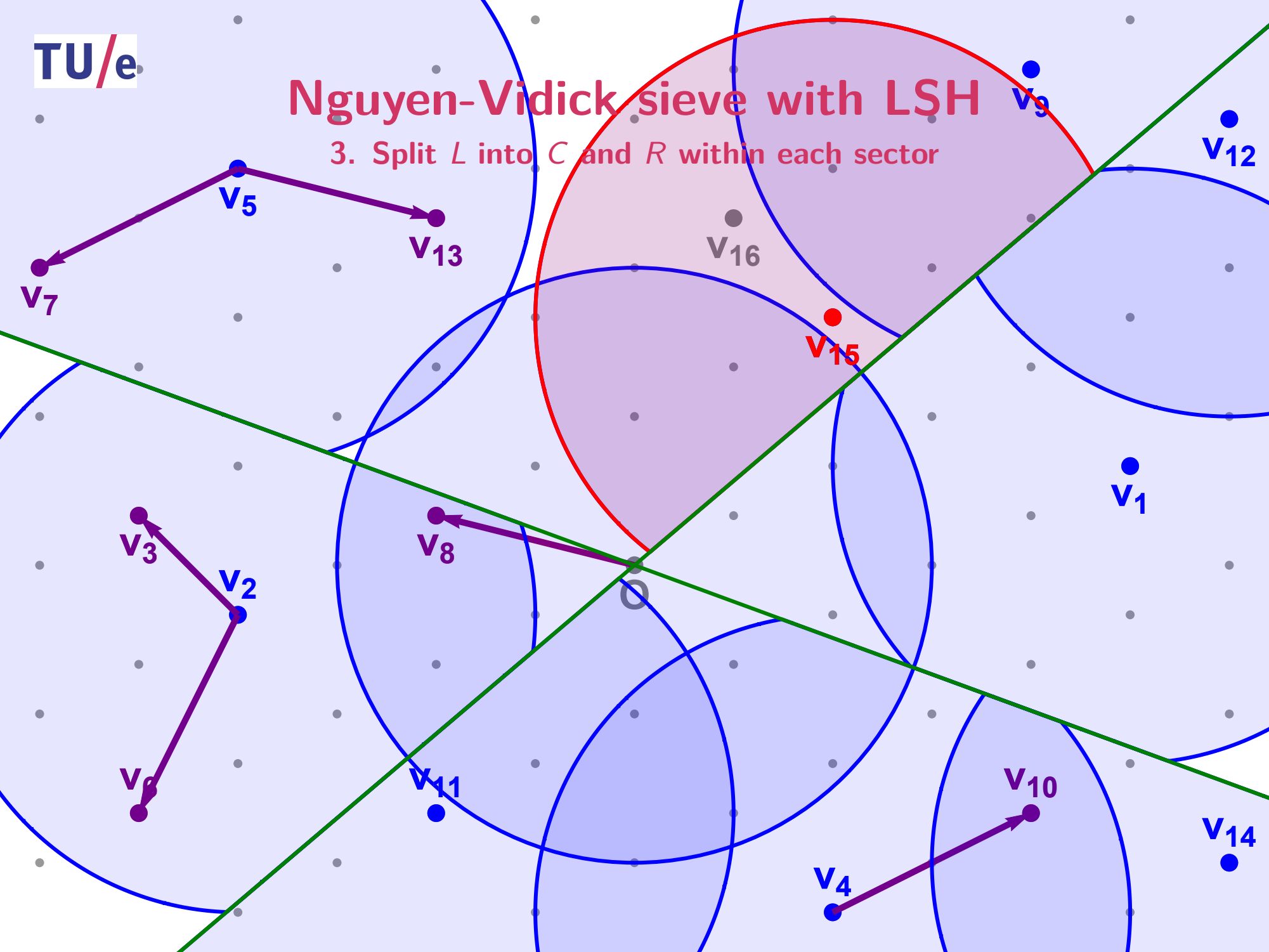

TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_{11}
v_{10}
v_{14}
v_4

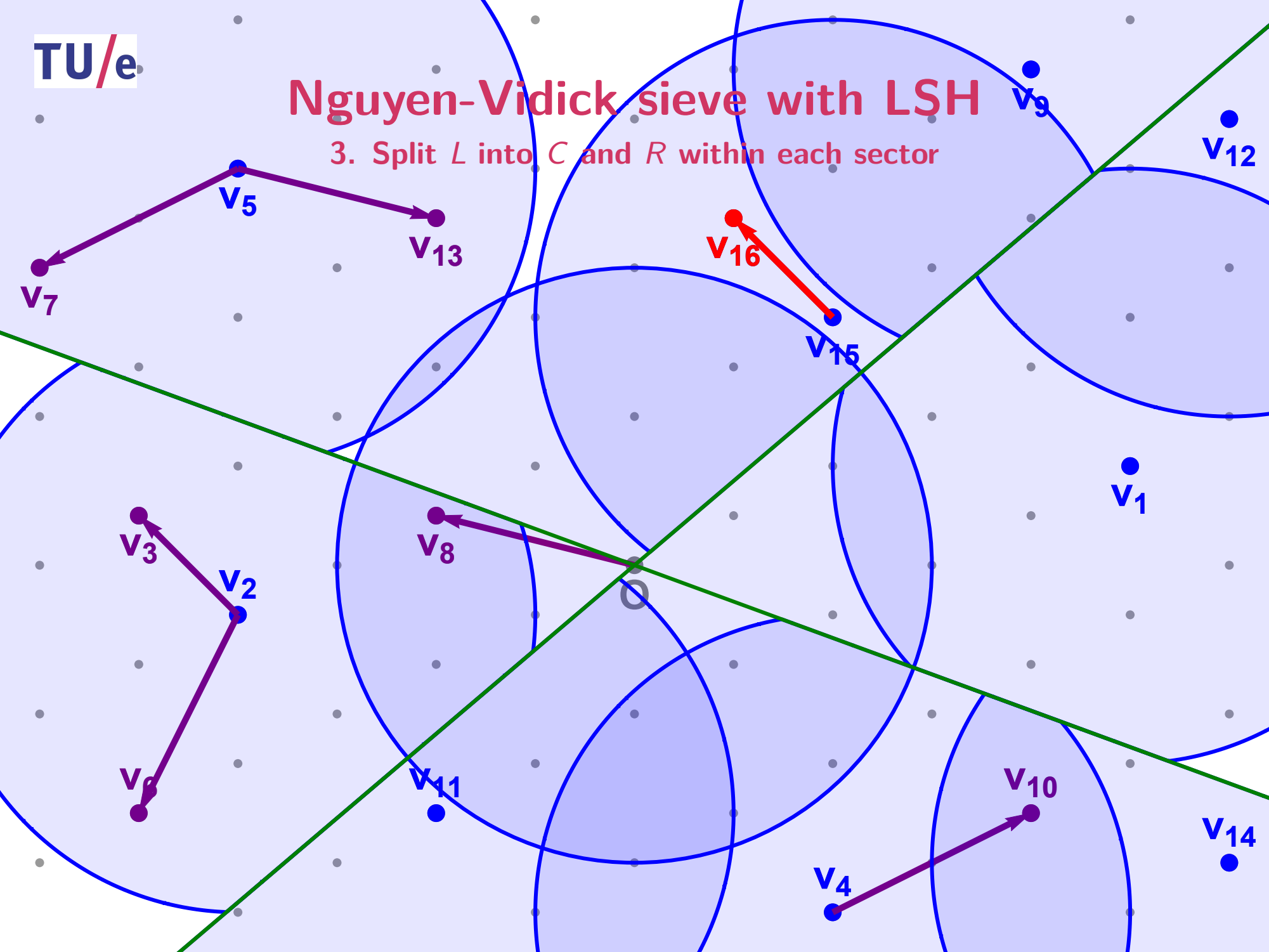

TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
$\mathbf{v}_9$
$\mathbf{v}_{12}$
$\mathbf{v}_5$
$\mathbf{v}_{13}$
$\mathbf{v}_{16}$
$\mathbf{v}_7$
$\mathbf{v}_{15}$
$\mathbf{v}_1$
$\mathbf{v}_3$
$\mathbf{v}_8$
$\mathbf{v}_2$
0
$\mathbf{v}_{11}$
$\mathbf{v}_{10}$
$\mathbf{v}_{14}$
$\mathbf{v}_4$


TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
$\mathbf{v}_9$
$\mathbf{v}_{12}$
$\mathbf{v}_5$
$\mathbf{v}_{13}$
$\mathbf{v}_{16}$
$\mathbf{v}_7$
$\mathbf{v}_{15}$
$\mathbf{v}_1$
$\mathbf{v}_3$
$\mathbf{v}_8$
$\mathbf{v}_2$
0
$\mathbf{v}_{11}$
$\mathbf{v}_{10}$
$\mathbf{v}_{14}$
$\mathbf{v}_4$


TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
0
v_{11}
v_{10}
v_{14}
v_4

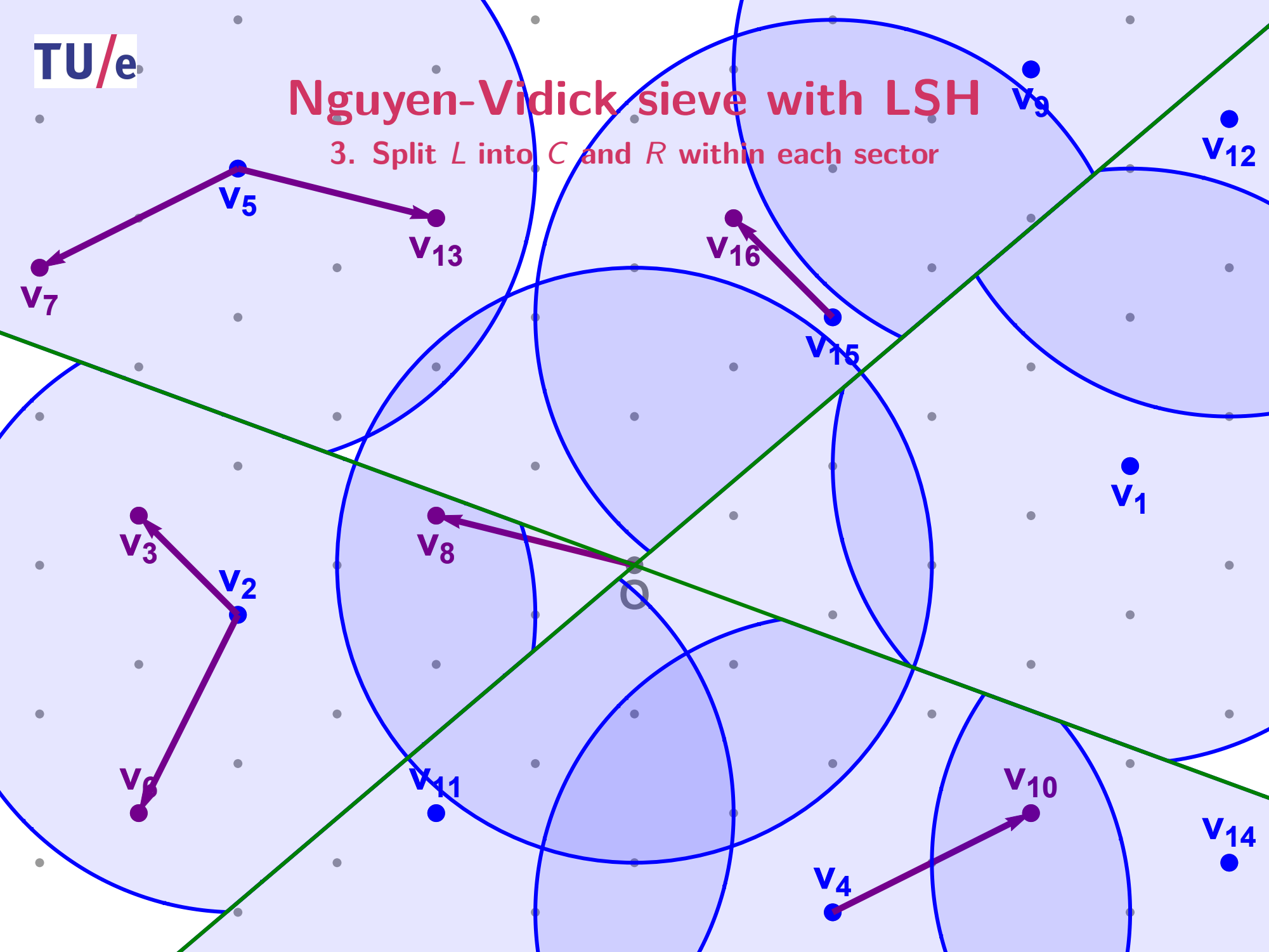

TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
$\mathbf{v}_5$
$\mathbf{v}_{13}$
$\mathbf{v}_7$
$\mathbf{v}_{16}$
$\mathbf{v}_9$
$\mathbf{v}_{12}$
$\mathbf{v}_{15}$
$\mathbf{v}_1$
$\mathbf{v}_3$
$\mathbf{v}_2$
$\mathbf{v}_8$
$\mathbf{v}_6$
$\mathbf{v}_{11}$
$\mathbf{v}_{10}$
$\mathbf{v}_{14}$
$\mathbf{v}_4$


TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
$\mathbf{v}_5$
$\mathbf{v}_{13}$
$\mathbf{v}_7$
$\mathbf{v}_{16}$
$\mathbf{v}_9$
$\mathbf{v}_{12}$
$\mathbf{v}_{15}$
$\mathbf{v}_1$
$\mathbf{v}_3$
$\mathbf{v}_8$
$\mathbf{v}_2$
0
$\mathbf{v}_6$
$\mathbf{v}_{11}$
$\mathbf{v}_{10}$
$\mathbf{v}_{14}$
$\mathbf{v}_4$


TU/e
Nguyen-Vidick sieve with LSH
3. Split L into C and R within each sector
$\mathbf{v}_5$
$\mathbf{v}_{13}$
$\mathbf{v}_7$
$\mathbf{v}_{16}$
$\mathbf{v}_{15}$
$\mathbf{v}_9$
$\mathbf{v}_{12}$
$\mathbf{v}_1$
$\mathbf{v}_3$
$\mathbf{v}_2$
$\mathbf{v}_8$
0
$\mathbf{v}_6$
$\mathbf{v}_{11}$
$\mathbf{v}_{10}$
$\mathbf{v}_{14}$
$\mathbf{v}_4$

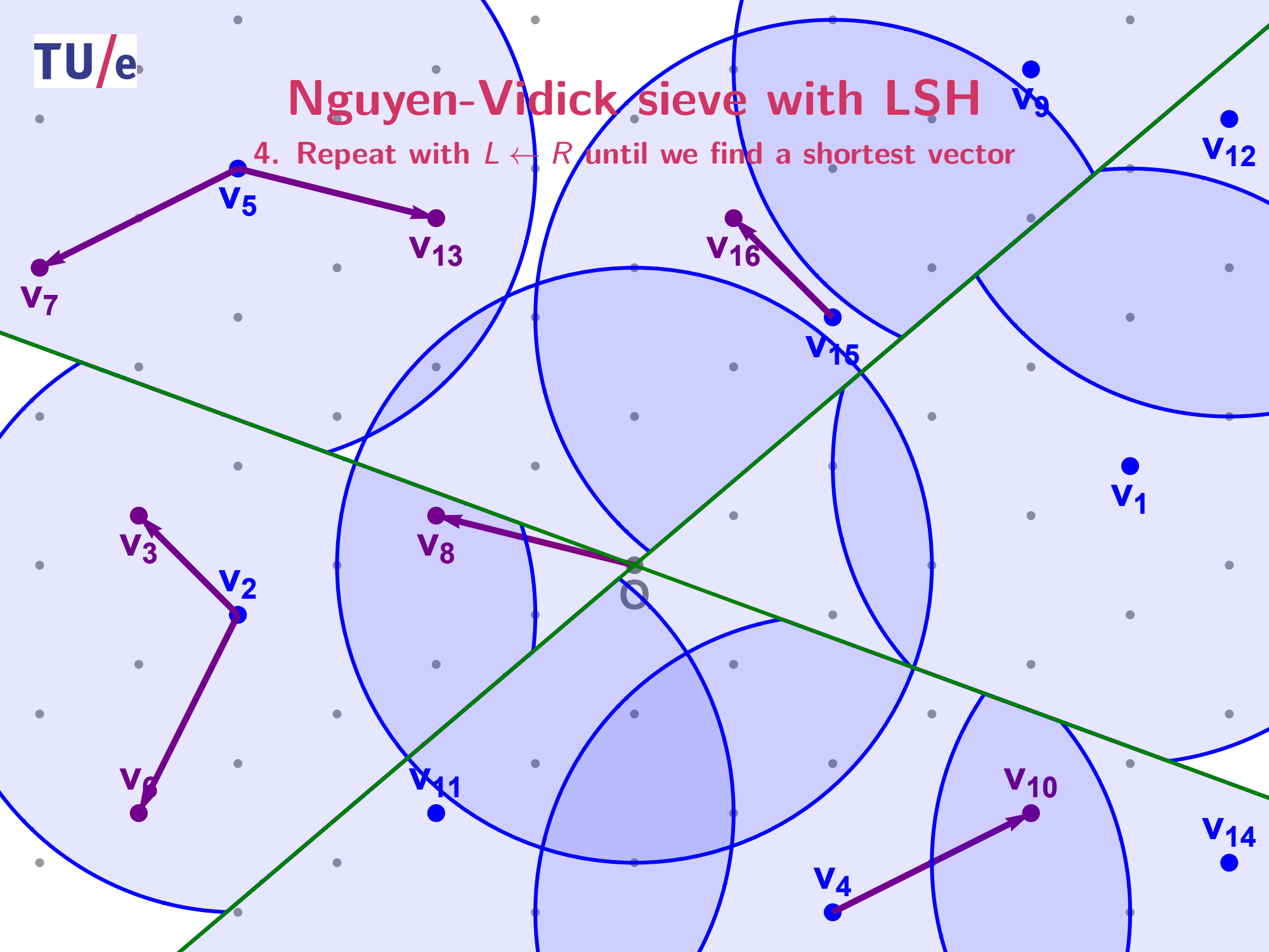
Nguyen-Vidick sieve with LSH

3. Split L into C and R within each sector



Nguyen-Vidick sieve with LSH

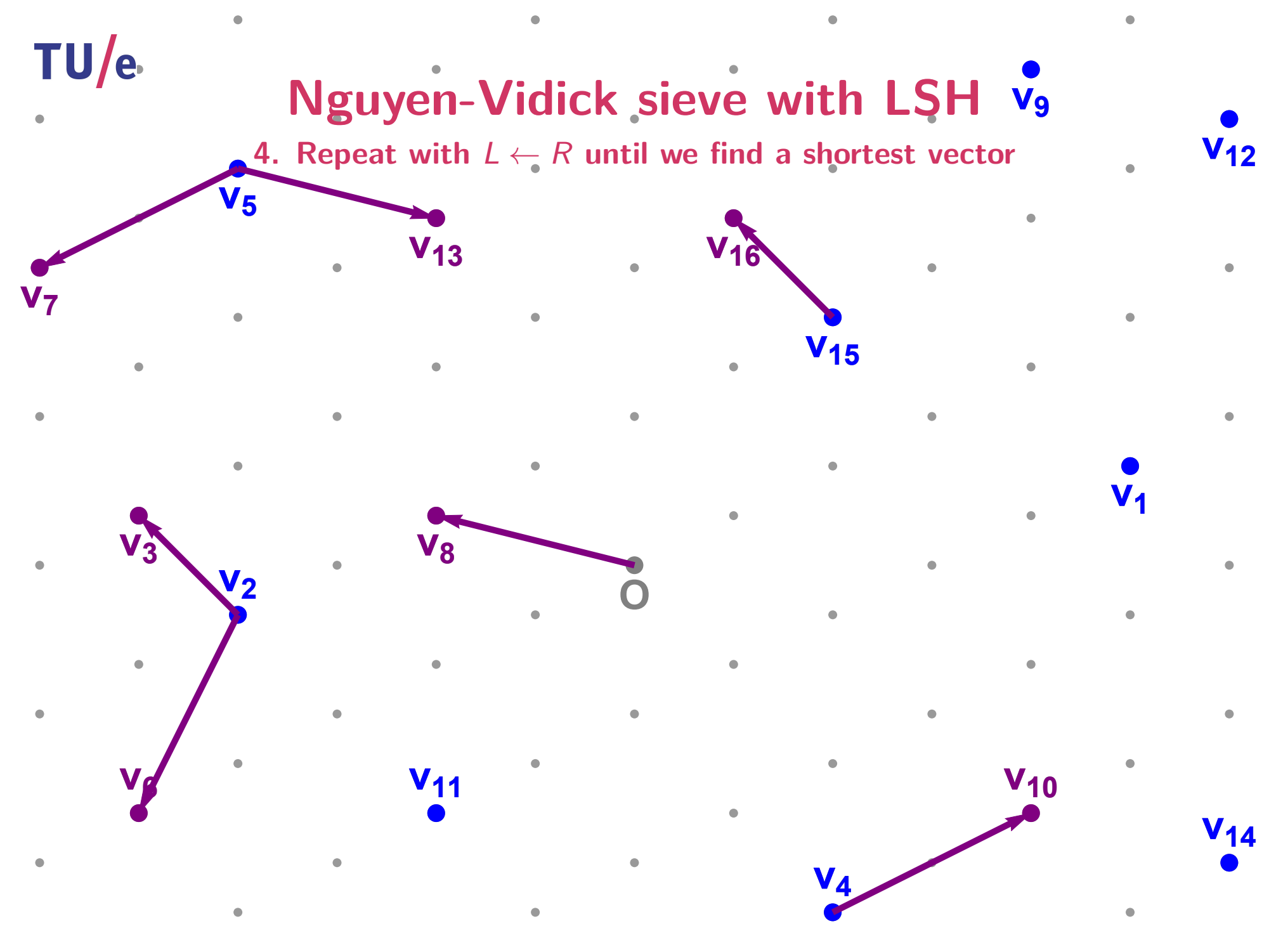
4. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve with LSH

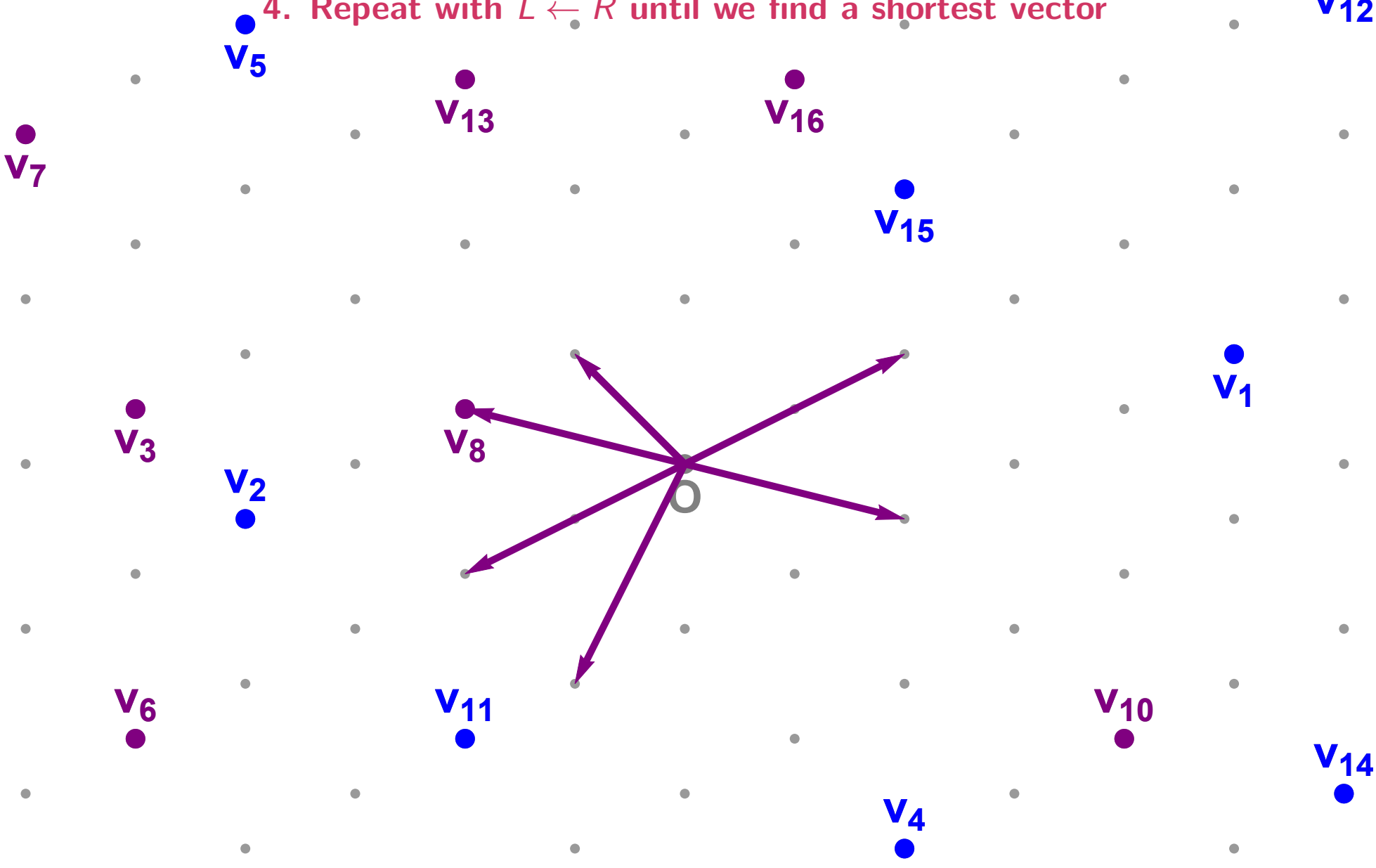
4. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve with LSH

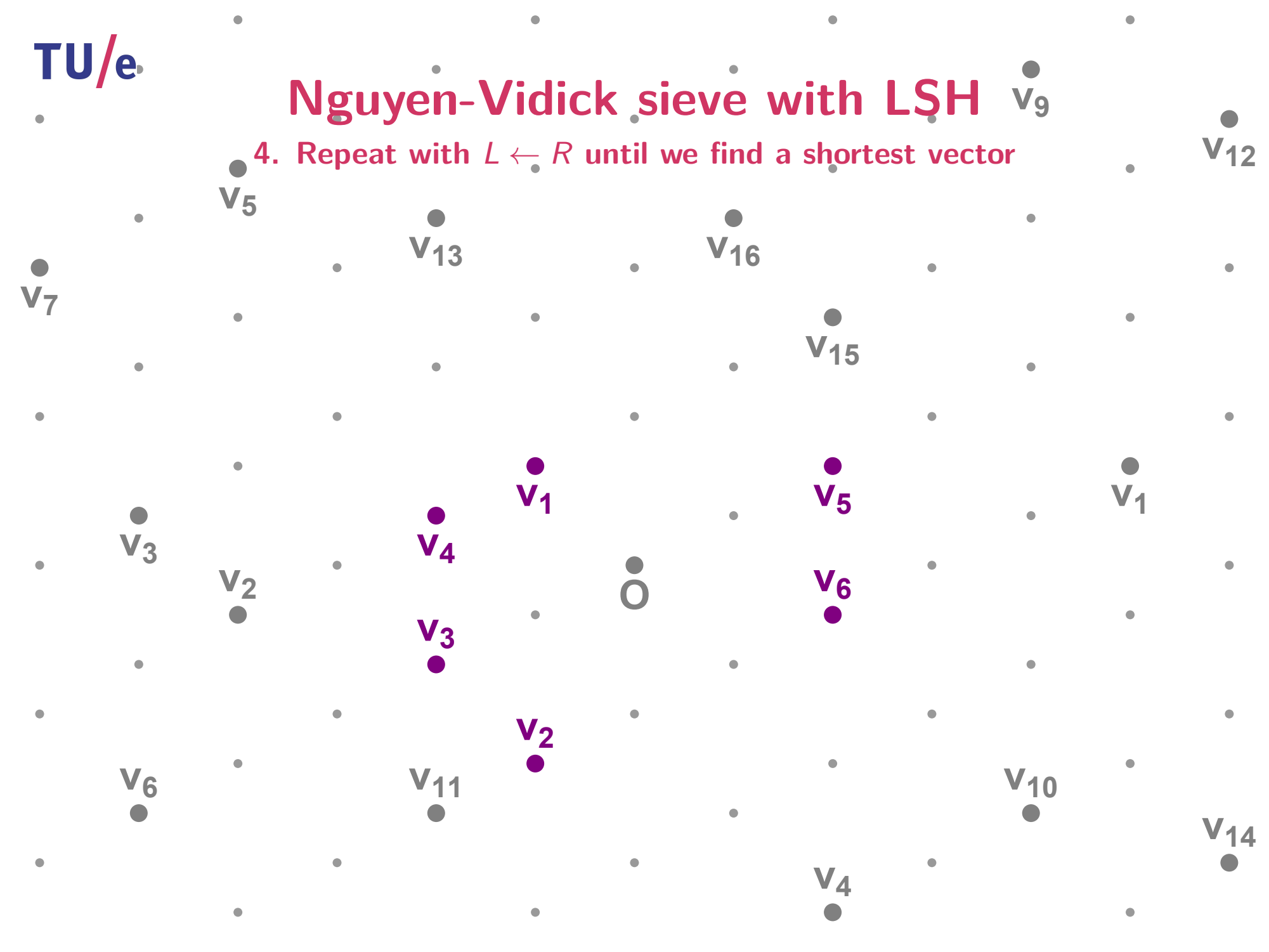
4. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve with LSH

4. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve with LSH

Overview

TU/e

Nguyen-Vidick sieve with LSH

Overview

- Two parameters to tune
  - $k = O(n)$: Number of hyperplanes, leading to 2^k regions
  - $t = 2^{O(n)}$: Number of different, independent "hash tables"

TU/e

Nguyen-Vidick sieve with LSH

Overview

- Two parameters to tune
  - $k = O(n)$: Number of hyperplanes, leading to 2^k regions
  - $t = 2^{O(n)}$: Number of different, independent "hash tables"
- Space complexity: $2^{0.34n+o(n)}$
  - Store $2^{0.13n}$ hash tables, each containing all $2^{0.21n}$ vectors

TU/e

Nguyen-Vidick sieve with LSH

Overview

- Two parameters to tune
  - $k = O(n)$: Number of hyperplanes, leading to 2^k regions
  - $t = 2^{O(n)}$: Number of different, independent "hash tables"
- Space complexity: $2^{0.34n+o(n)}$
  - Store $2^{0.13n}$ hash tables, each containing all $2^{0.21n}$ vectors
- Time complexity: $2^{0.34n+o(n)}$
  - Compute $2^{0.13n}$ hashes, and go through $2^{0.13n}$ vectors
  - Repeat this for each of $2^{0.21n}$ vectors

TU/e

Nguyen-Vidick sieve with LSH

Overview

- Two parameters to tune
  - $k = O(n)$: Number of hyperplanes, leading to 2^k regions
  - $t = 2^{O(n)}$: Number of different, independent "hash tables"
- Space complexity: $2^{0.34n+o(n)}$
  - Store $2^{0.13n}$ hash tables, each containing all $2^{0.21n}$ vectors
- Time complexity: $2^{0.34n+o(n)}$
  - Compute $2^{0.13n}$ hashes, and go through $2^{0.13n}$ vectors
  - Repeat this for each of $2^{0.21n}$ vectors

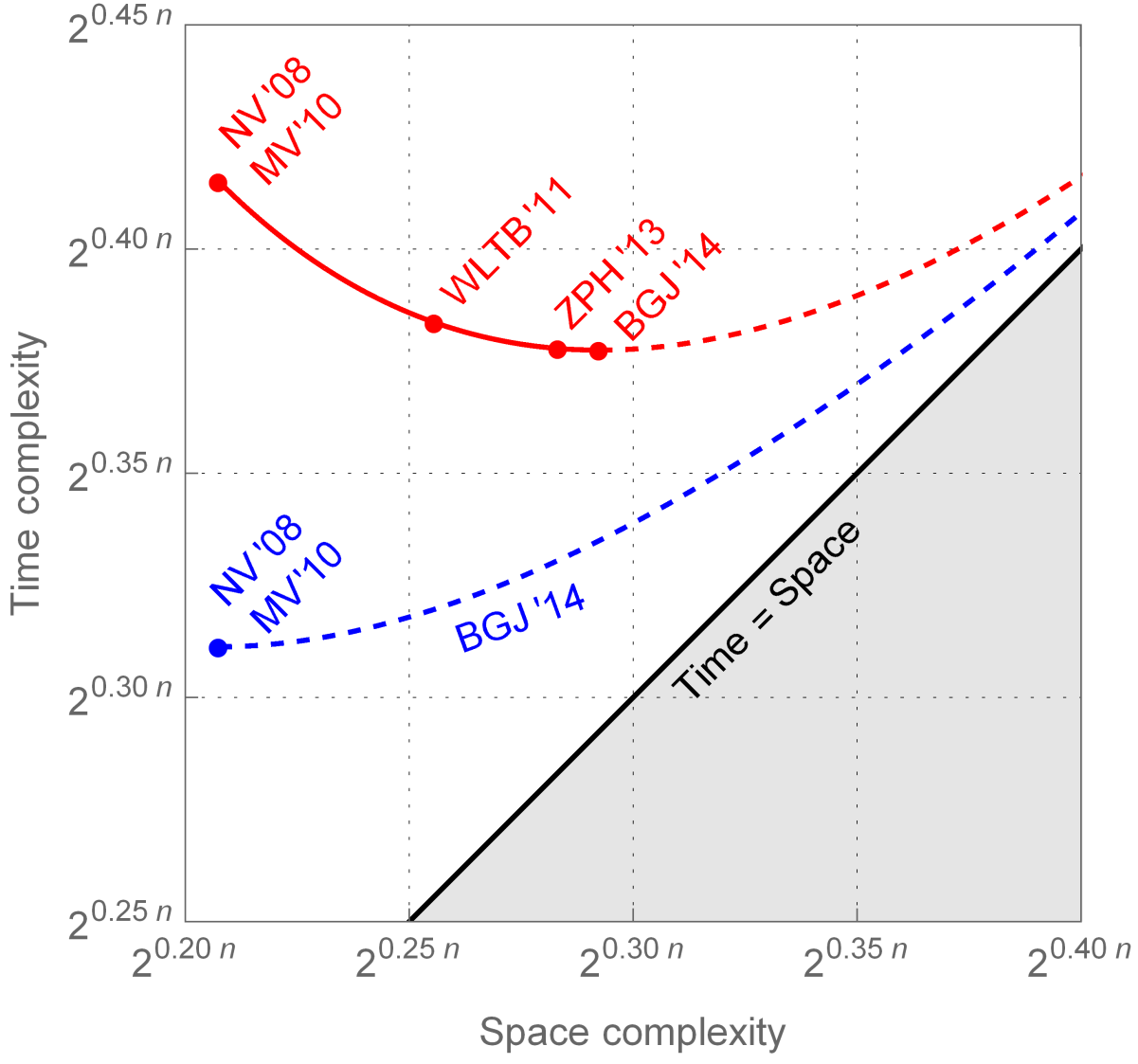
Heuristic

Sieving with LSH runs in time $2^{0.34n+o(n)}$ and space $2^{0.34n+o(n)}$.

TU/e

Sieving

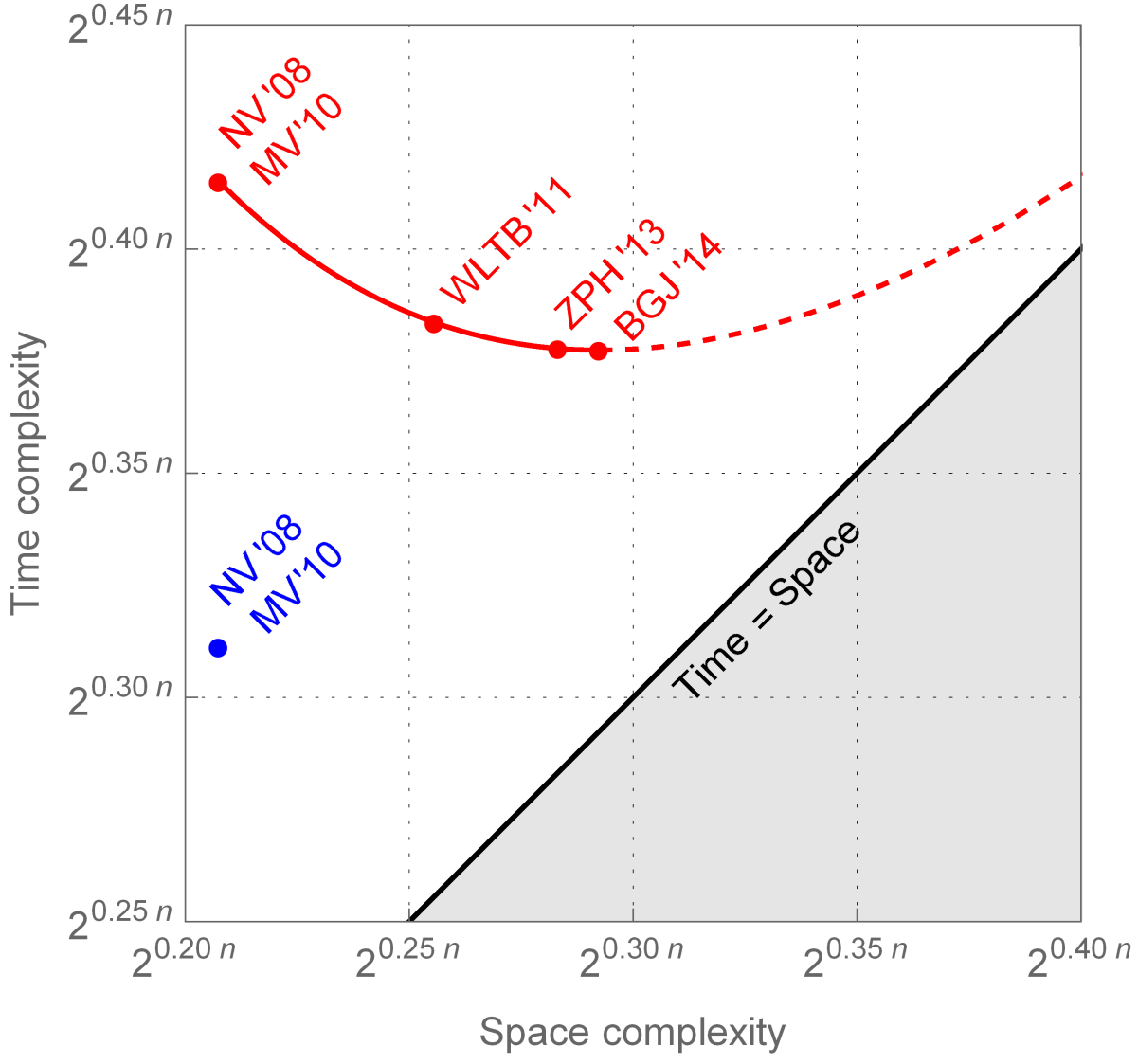
Space/time trade-off



TU/e

Sieving

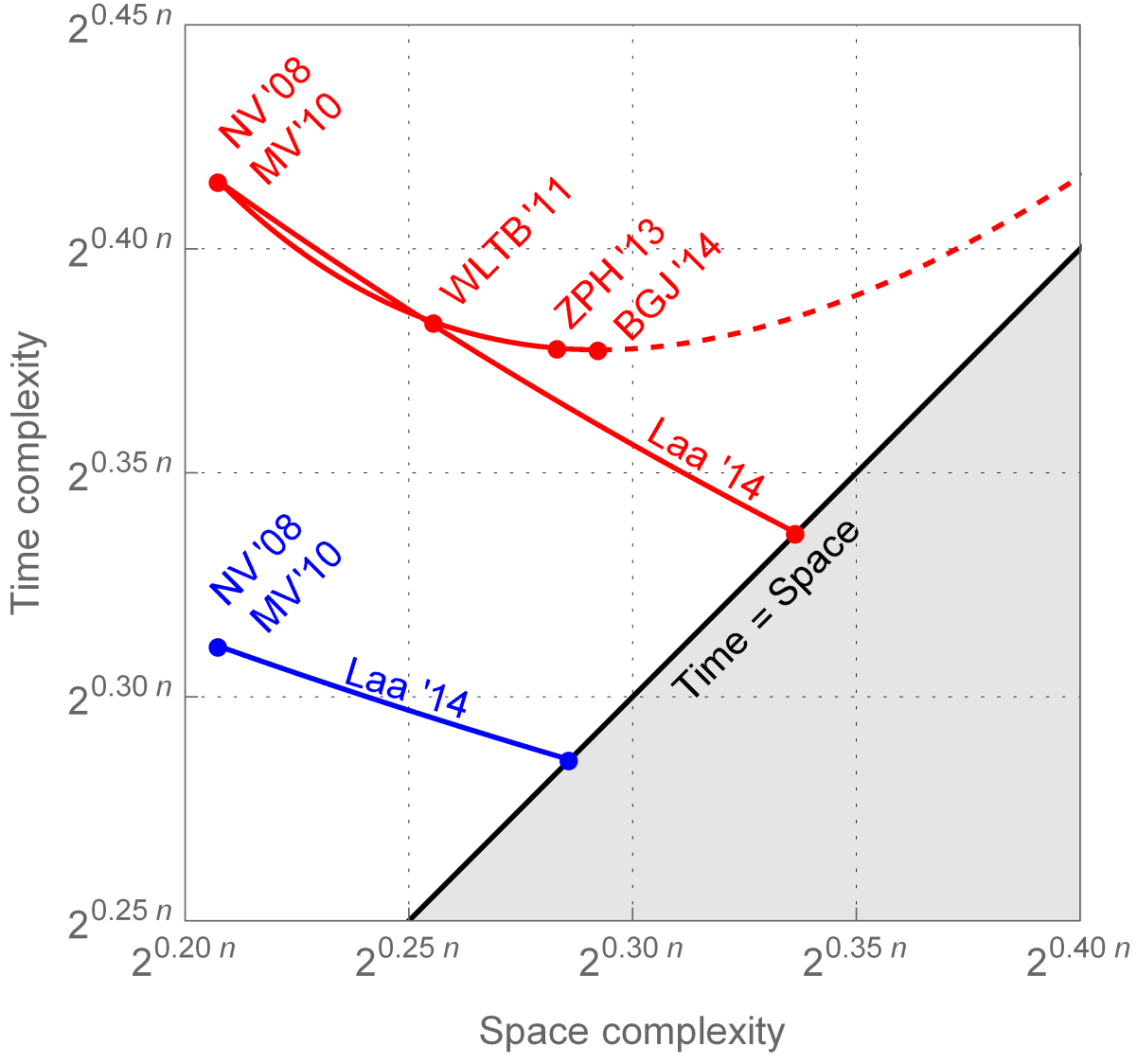
Space/time trade-off



TU/e

Sieving with LSH

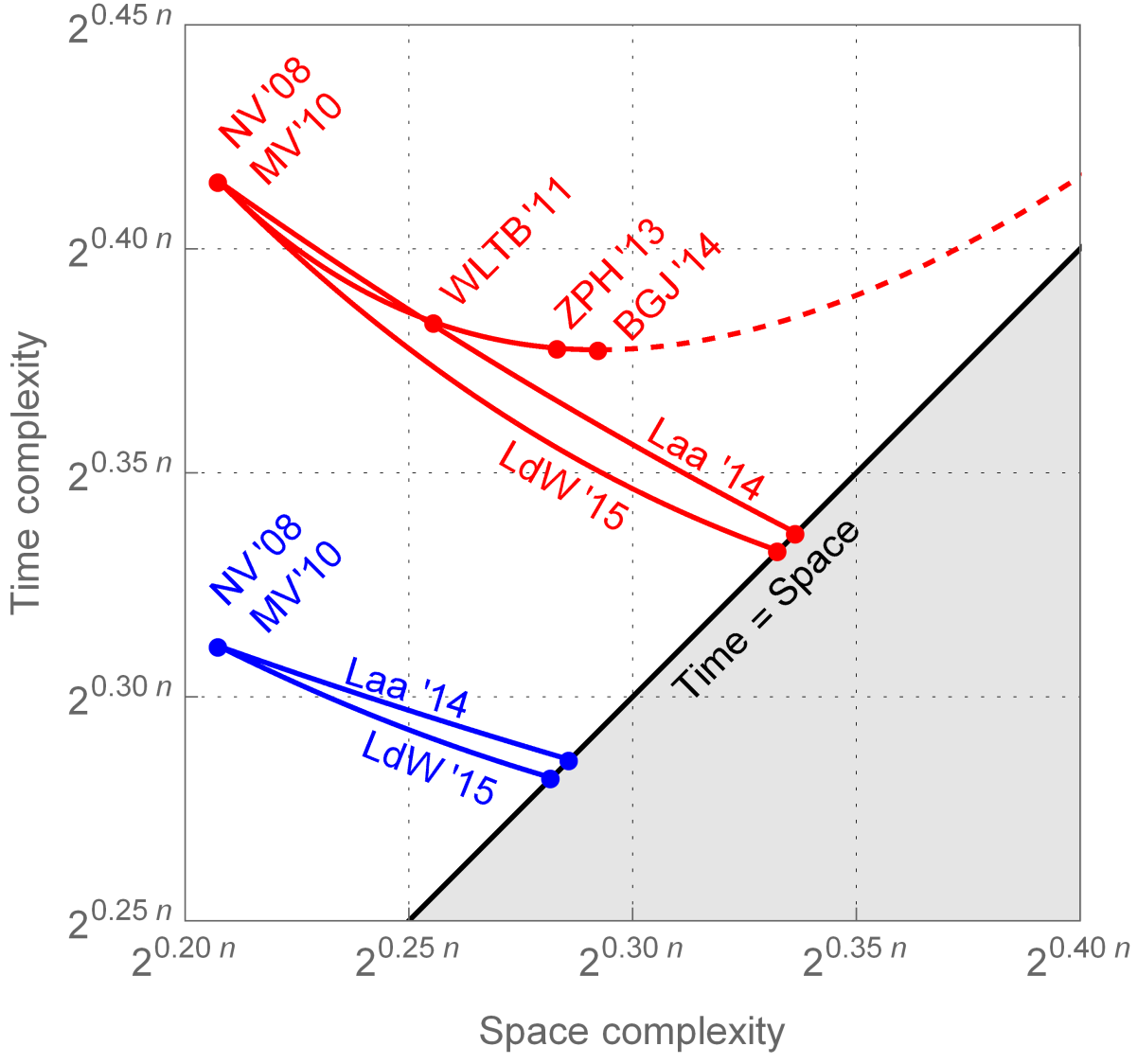
Space/time trade-off



TU/e

Sieving with LSH

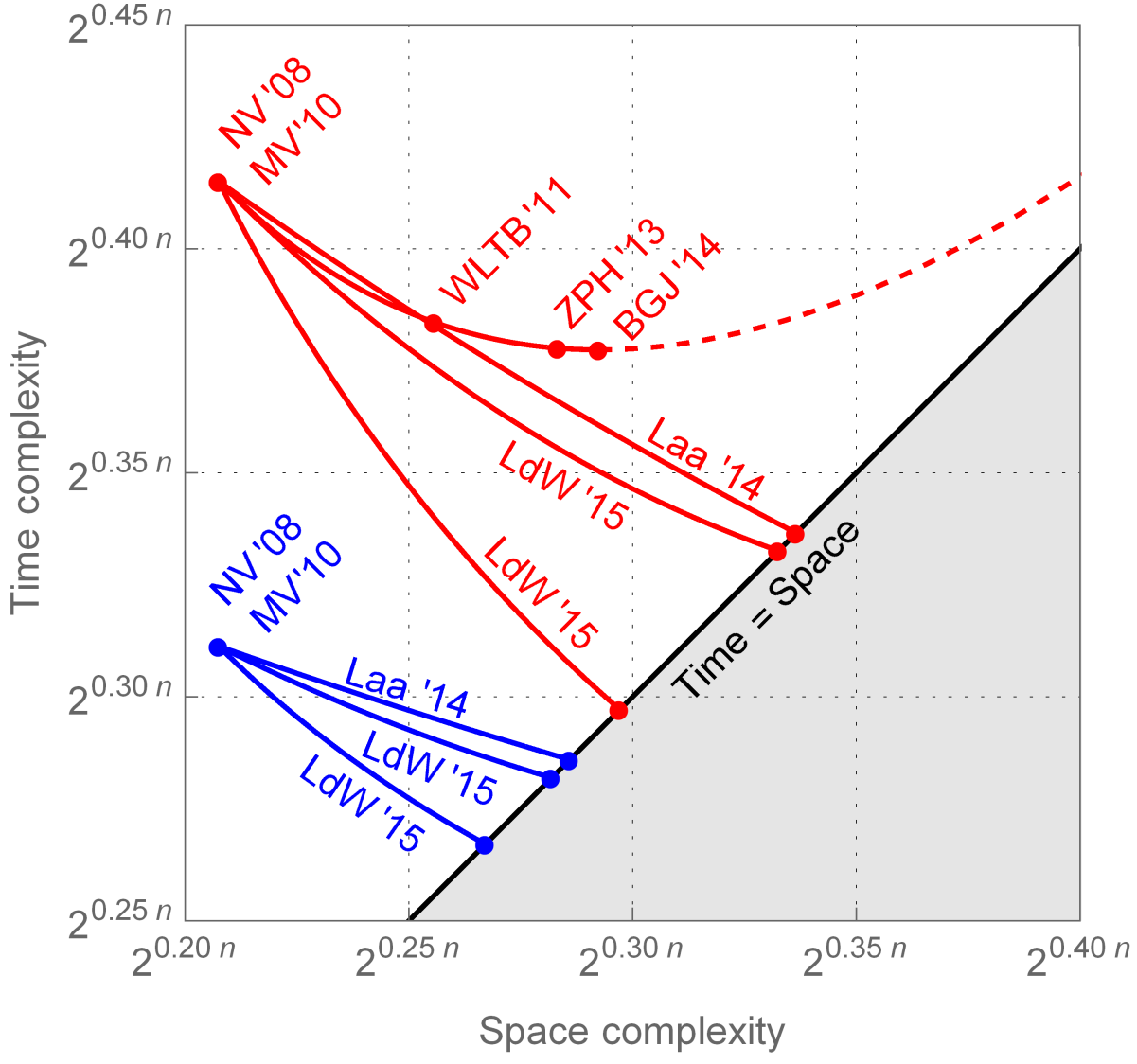
Space/time trade-off



TU/e

Sieving with LSH

Space/time trade-off



TU/e

Questions