TU/e Technische Universiteit Eindhoven University of Technology

Sieving for shortest lattice vectors using locality-sensitive hashing

Thijs Laarhoven

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

DIAMANT-symposium, Veenendaal, The Netherlands
(May 29, 2015)

TU/e

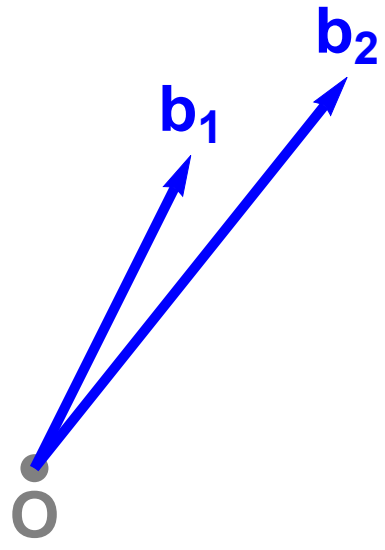
Lattices

What is a lattice?

TU/e

Lattices

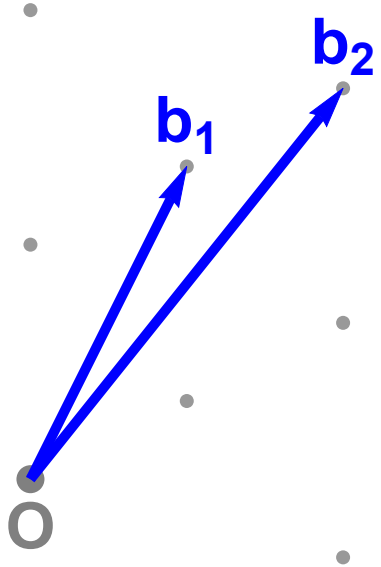
What is a lattice?



TU/e

Lattices

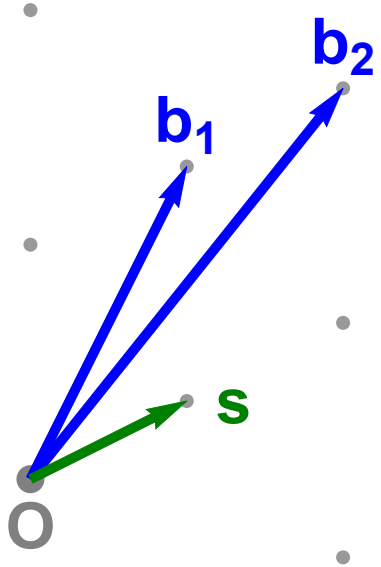
What is a lattice?



TU/e

Lattices

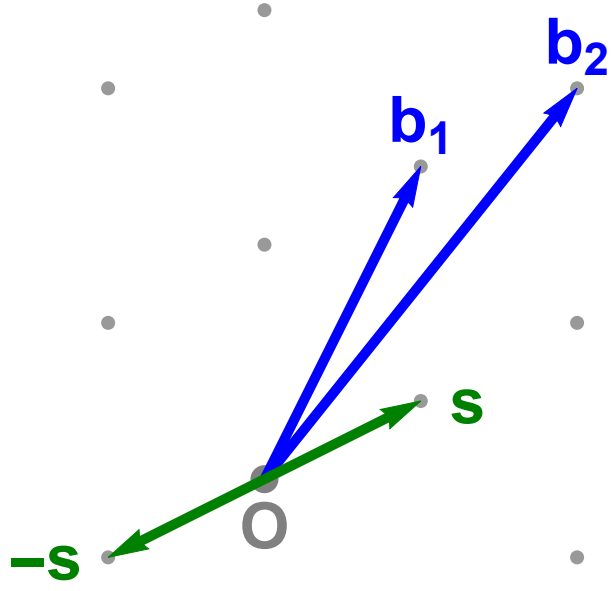
Shortest Vector Problem (SVP)



TU/e

Lattices

Shortest Vector Problem (SVP)



TU/e

Lattices

SVP algorithms

	Algorithm	$\log_2$(Time)	$\log_2$(Space)
Provable SVP	Enumeration [Poh81, Kan83, ..., GNR10]	$\Omega(n \log n)$	$O(\log n)$
	AKS-sieve [AKS01, NV08, MV10, HPS11]	$3.398n$	$1.985n$
	ListSieve [MV10, MDB14]	$3.199n$	$1.327n$
	AKS-sieve-birthday [PS09, HPS11]	$2.648n$	$1.324n$
	ListSieve-birthday [PS09]	$2.465n$	$1.233n$
	Voronoi cell algorithm [MV10b]	$2.000n$	$1.000n$
	Discrete Gaussian sampling [ADRS15, ADS15]	$1.000n$	$1.000n$
Heuristic SVP	NV-sieve [NV08]	$0.415n$	$0.208n$
	GaussSieve [MV10, ..., IKMT14, BNvdP14]	$0.415n$?	$0.208n$
	Two-level sieve [WLTB11]	$0.384n$	$0.256n$
	Three-level sieve [ZPH14]	$0.3778n$	$0.283n$
	Decomposition approach [BGJ14]	$0.3774n$	$0.293n$
	HashSieve [Laa15, MLB15]	$0.337n$	$0.337n$
	SphereSieve [LdW15]	$0.298n$	$0.298n$
	CrossPolytopeSieve [BL15]	$0.298n$	$0.298n$

TU/e

Lattices

SVP algorithms

	Algorithm	$\log_2$(Time)	$\log_2$(Space)
Provable SVP	Enumeration [Poh81, Kan83, ..., GNR10]	$\Omega(n \log n)$	$O(\log n)$
	AKS-sieve [AKS01, NV08, MV10, HPS11]	$3.398n$	$1.985n$
	ListSieve [MV10, MDB14]	$3.199n$	$1.327n$
	AKS-sieve-birthday [PS09, HPS11]	$2.648n$	$1.324n$
	ListSieve-birthday [PS09]	$2.465n$	$1.233n$
	Voronoi cell algorithm [MV10b]	$2.000n$	$1.000n$
	Discrete Gaussian sampling [ADRS15, ADS15]	$1.000n$	$1.000n$
Heuristic SVP	**NV-sieve** [NV08]	$0.415n$	$0.208n$
	GaussSieve [MV10, ..., IKMT14, BNvdP14]	$0.415n$?	$0.208n$
	Two-level sieve [WLTB11]	$0.384n$	$0.256n$
	Three-level sieve [ZPH14]	$0.3778n$	$0.283n$
	Decomposition approach [BGJ14]	$0.3774n$	$0.293n$
	HashSieve [Laa15, MLB15]	$0.337n$	$0.337n$
	SphereSieve [LdW15]	$0.298n$	$0.298n$
	CrossPolytopeSieve [BL15]	$0.298n$	$0.298n$

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

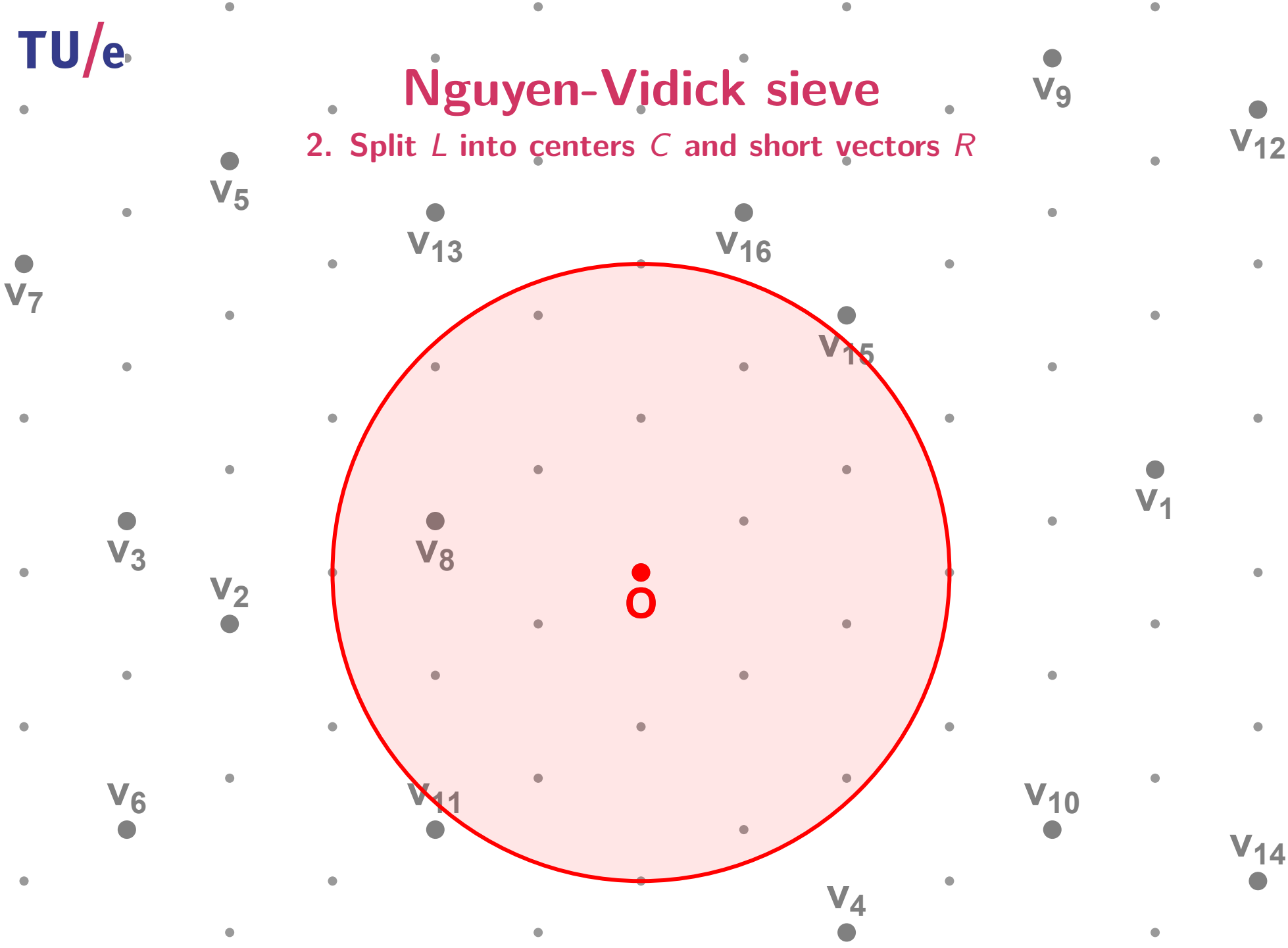
Nguyen-Vidick sieve

2. Split L into centers C and short vectors R

TU/e

Nguyen-Vidick sieve

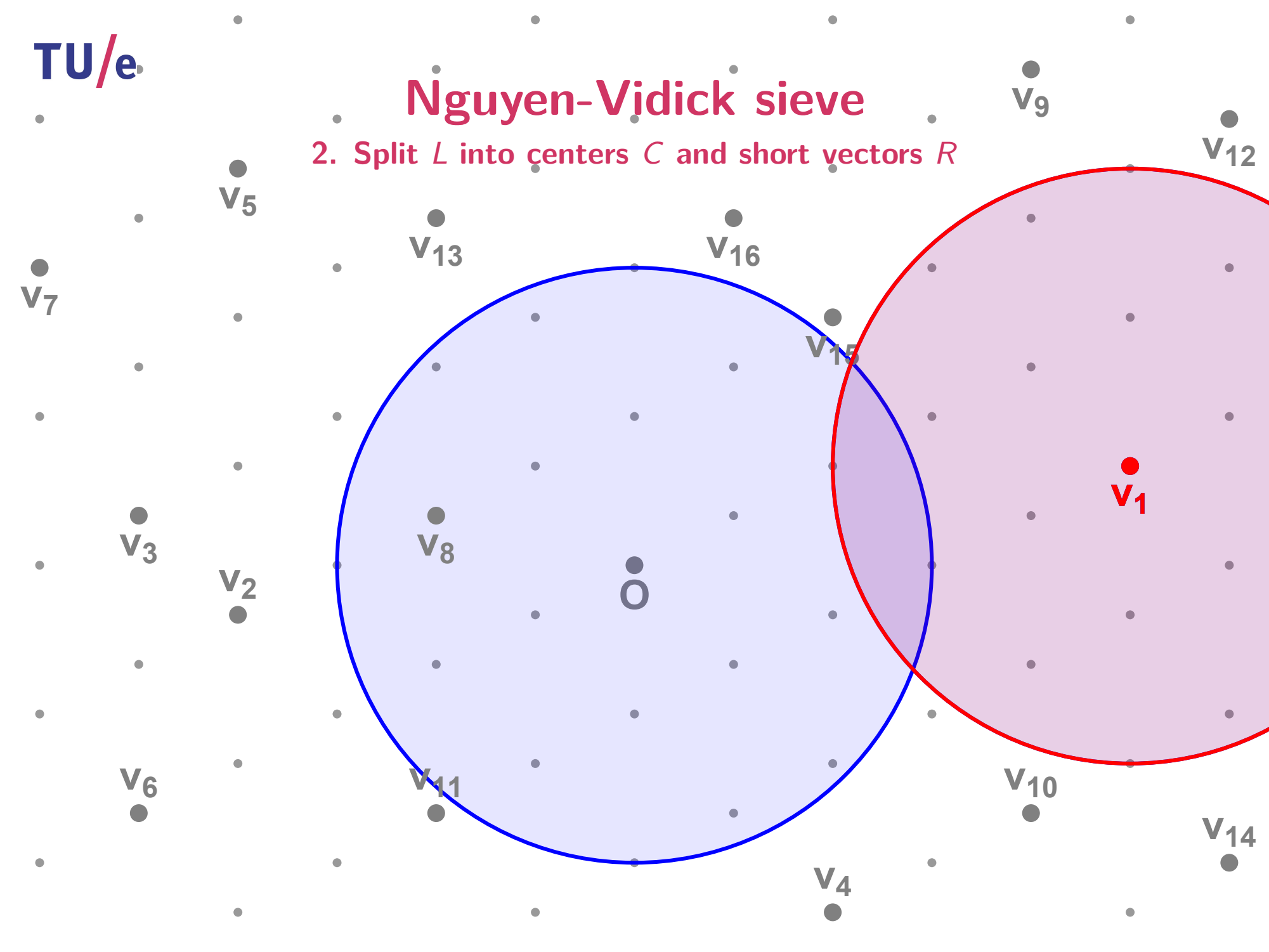
2. Split L into centers C and short vectors R



TU/e

Nguyen-Vidick sieve

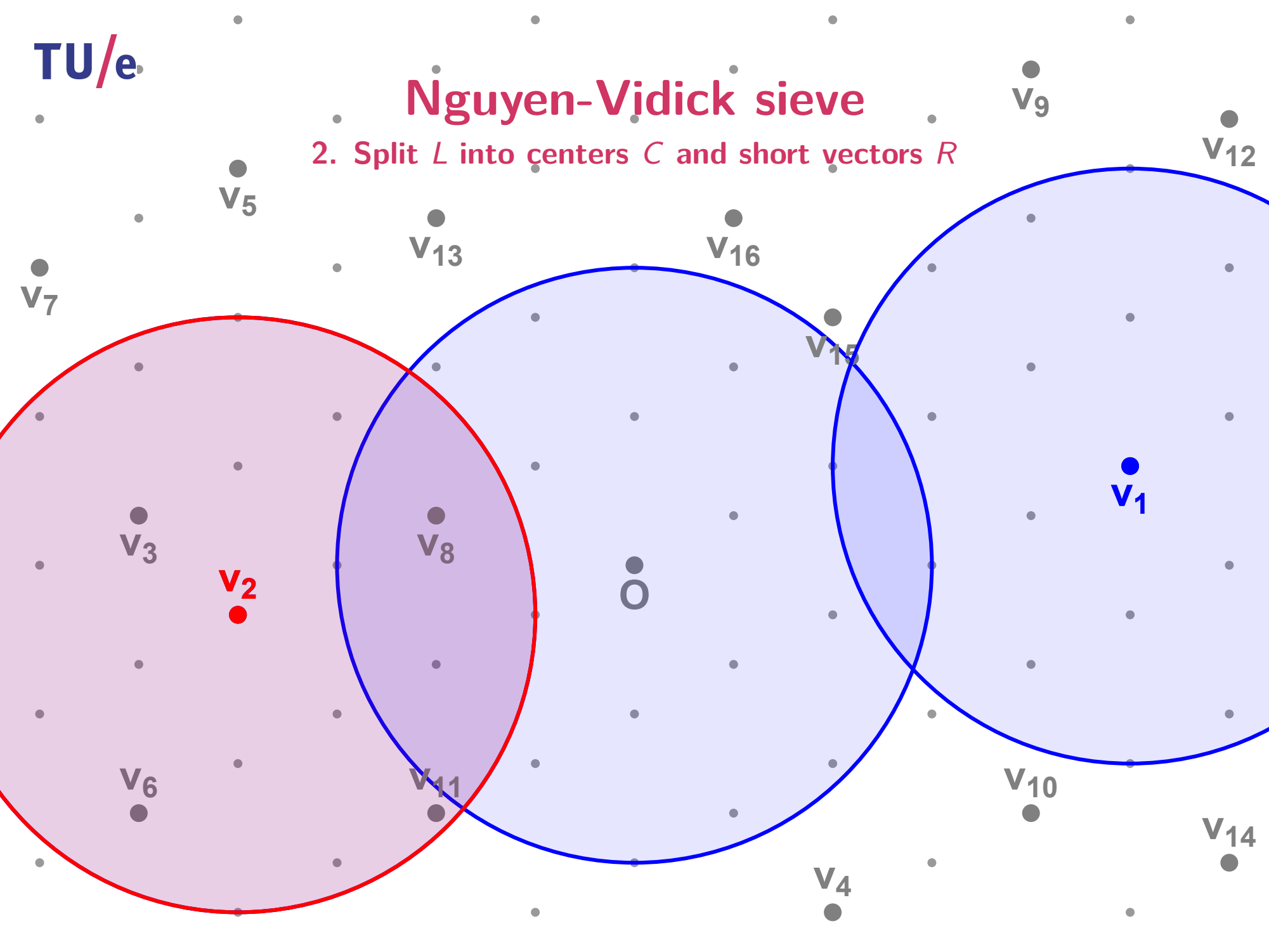
2. Split L into centers C and short vectors R



TU/e

Nguyen-Vidick sieve

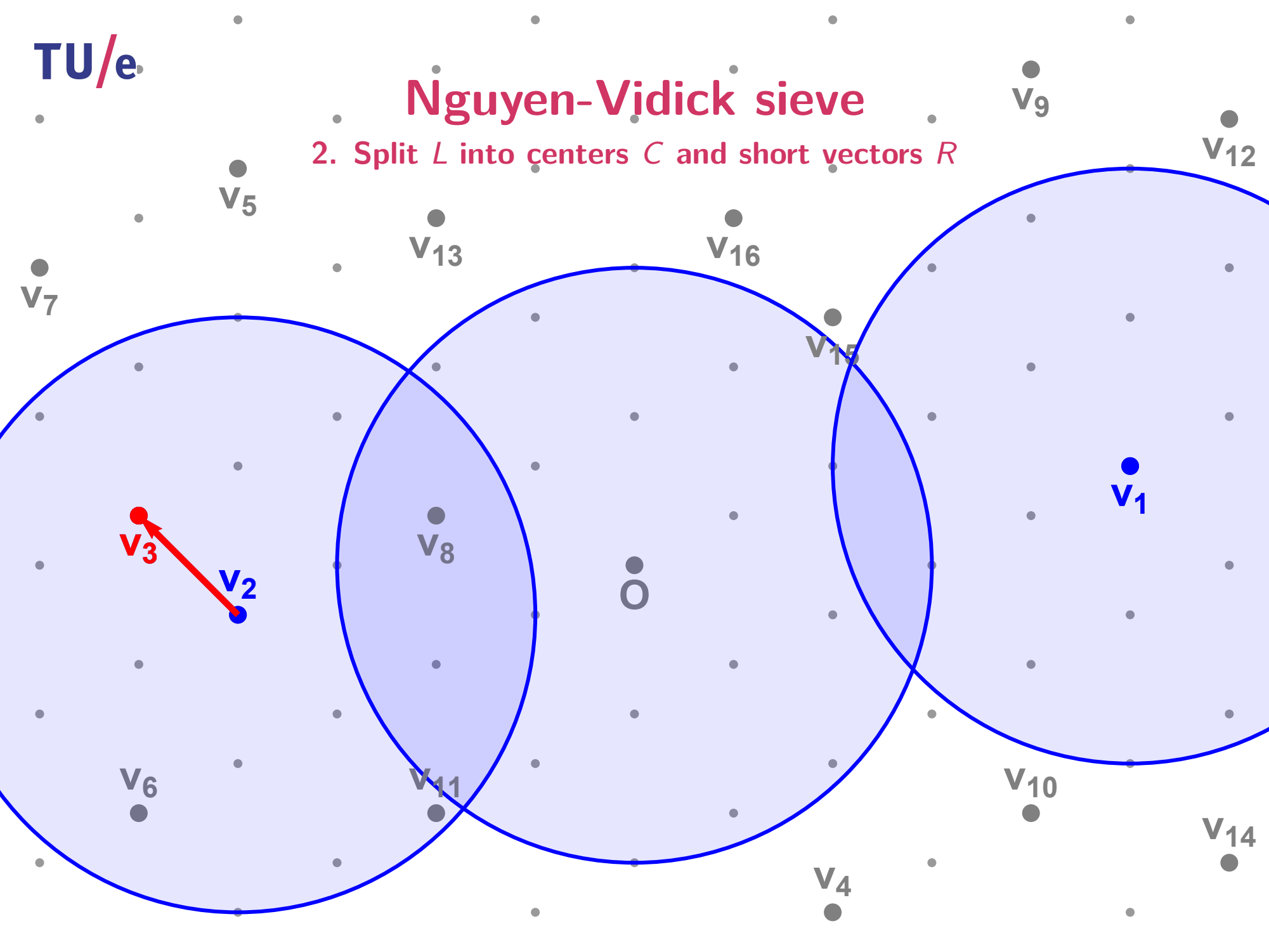
2. Split L into centers C and short vectors R



TU/e

Nguyen-Vidick sieve

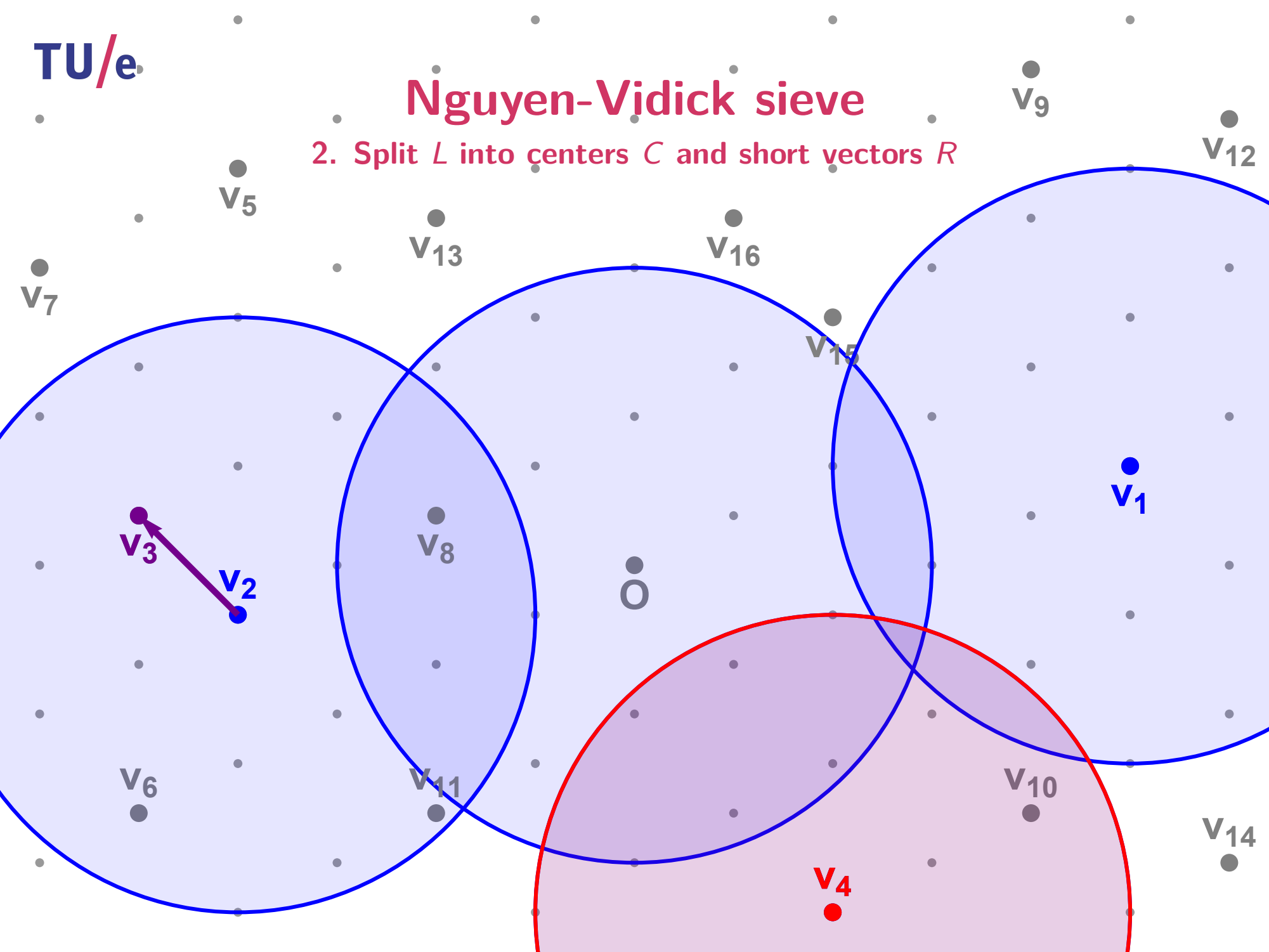
2. Split L into centers C and short vectors R



TU/e

Nguyen-Vidick sieve

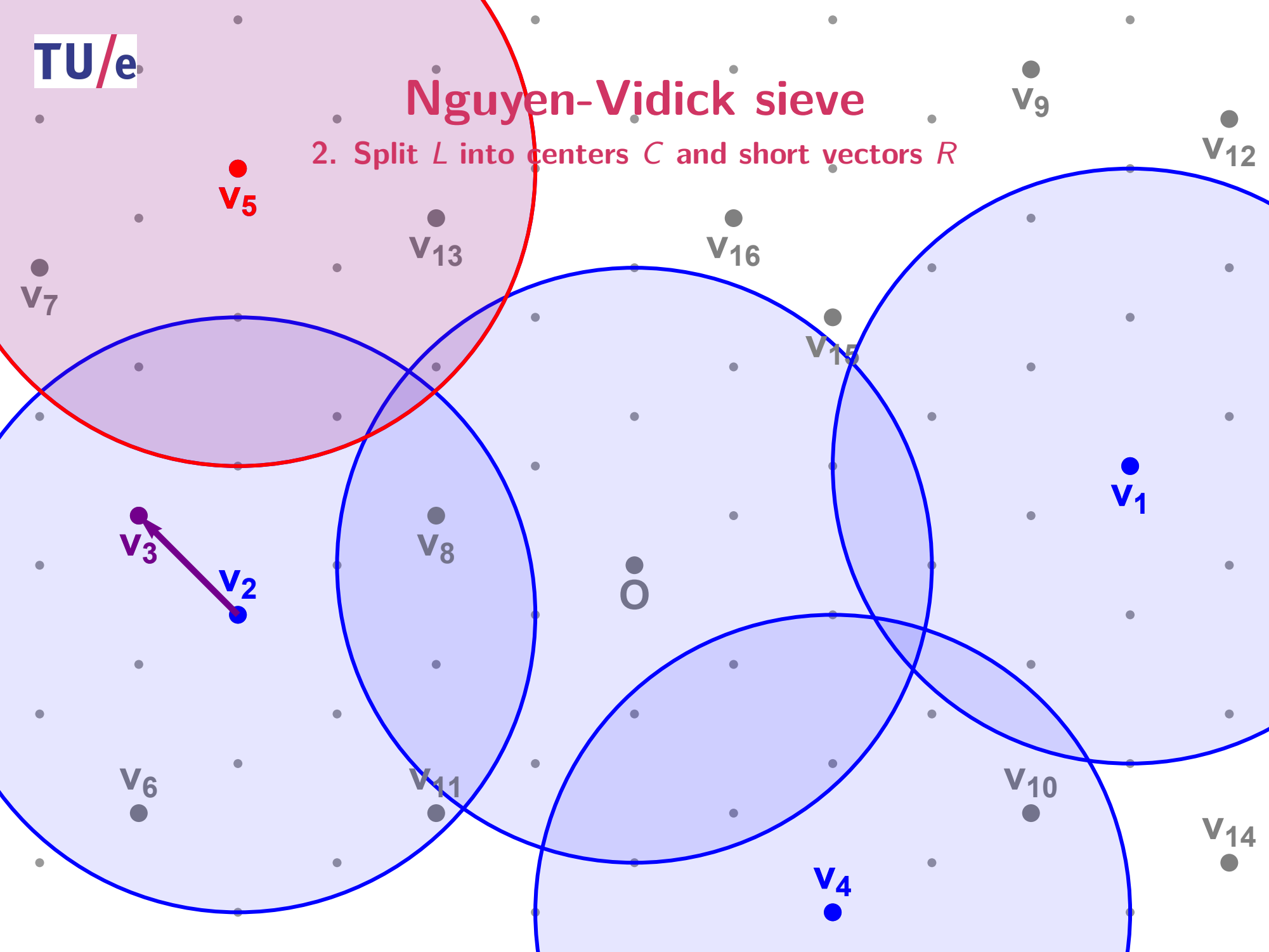
2. Split L into centers C and short vectors R



TU/e

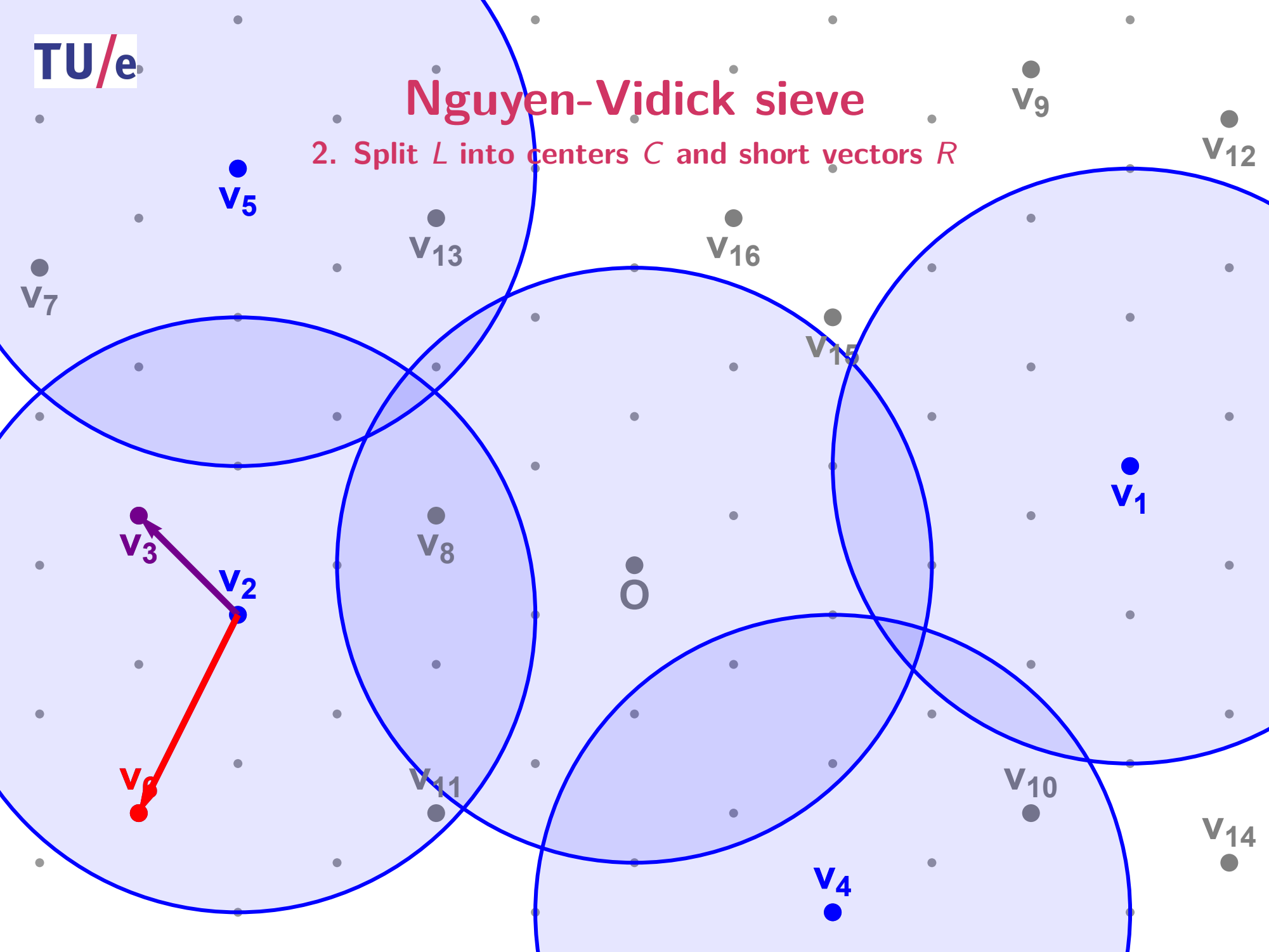
Nguyen-Vidick sieve

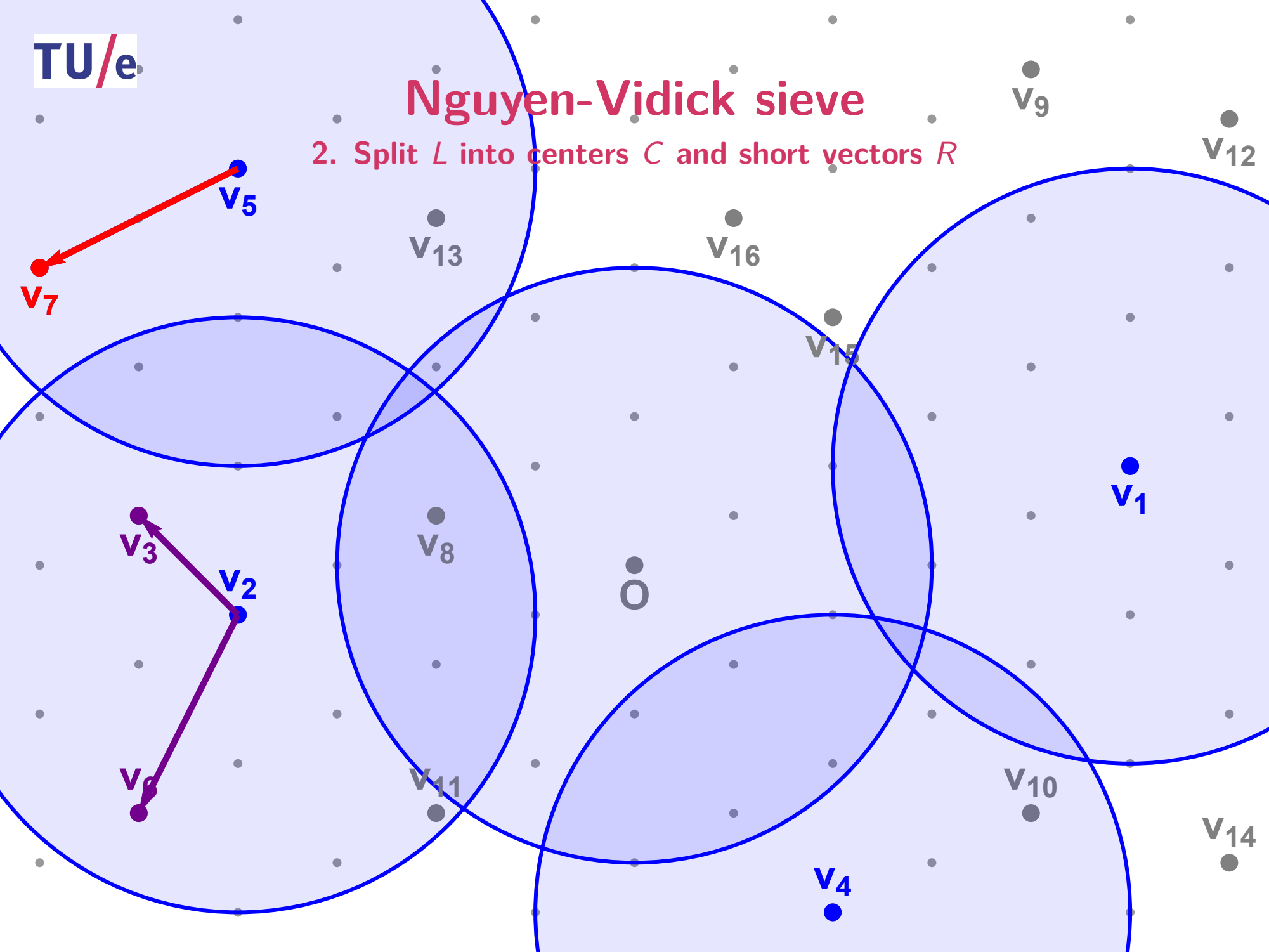
2. Split L into centers C and short vectors R



Nguyen-Vidick sieve

2. Split L into centers C and short vectors R

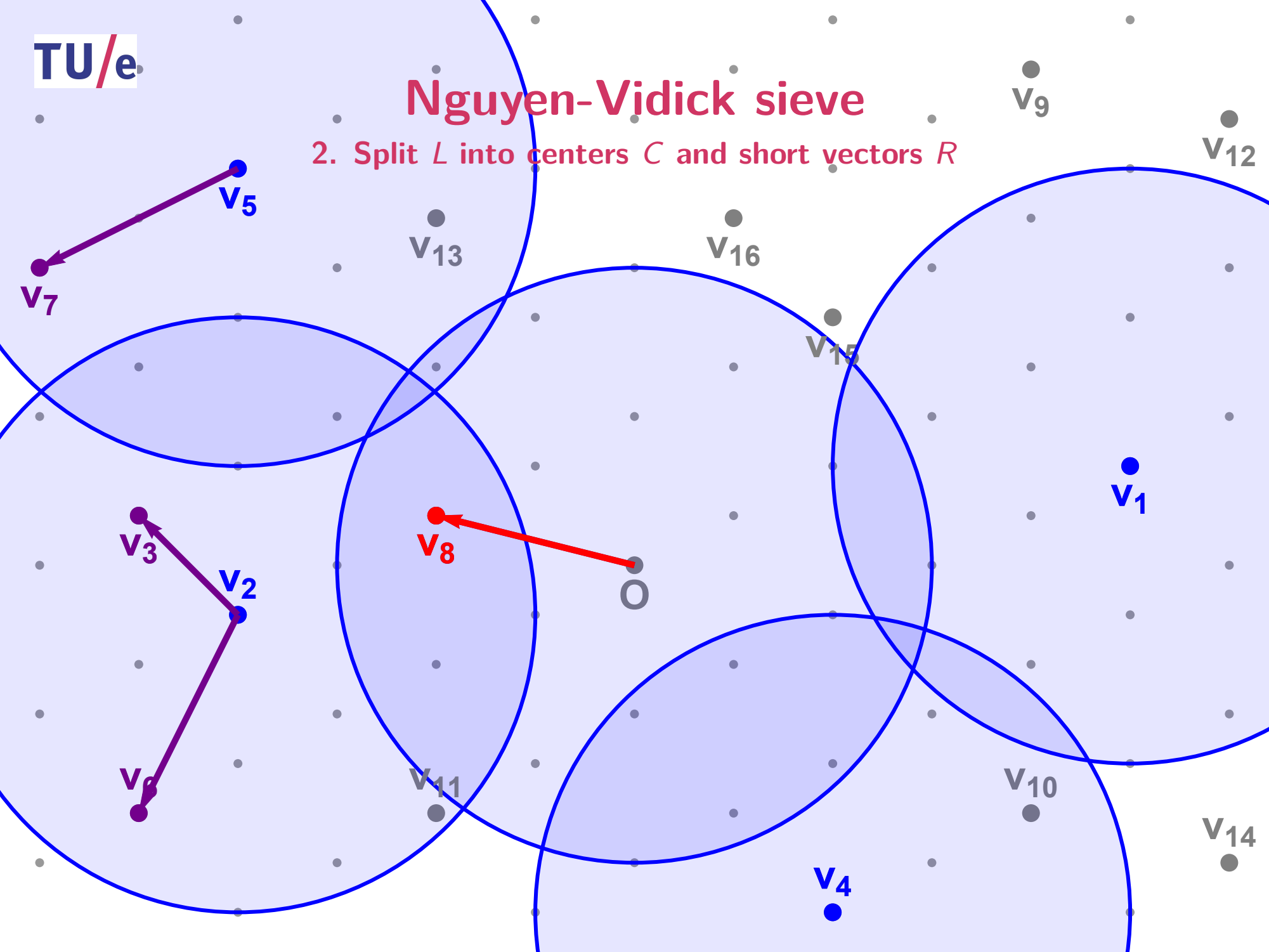


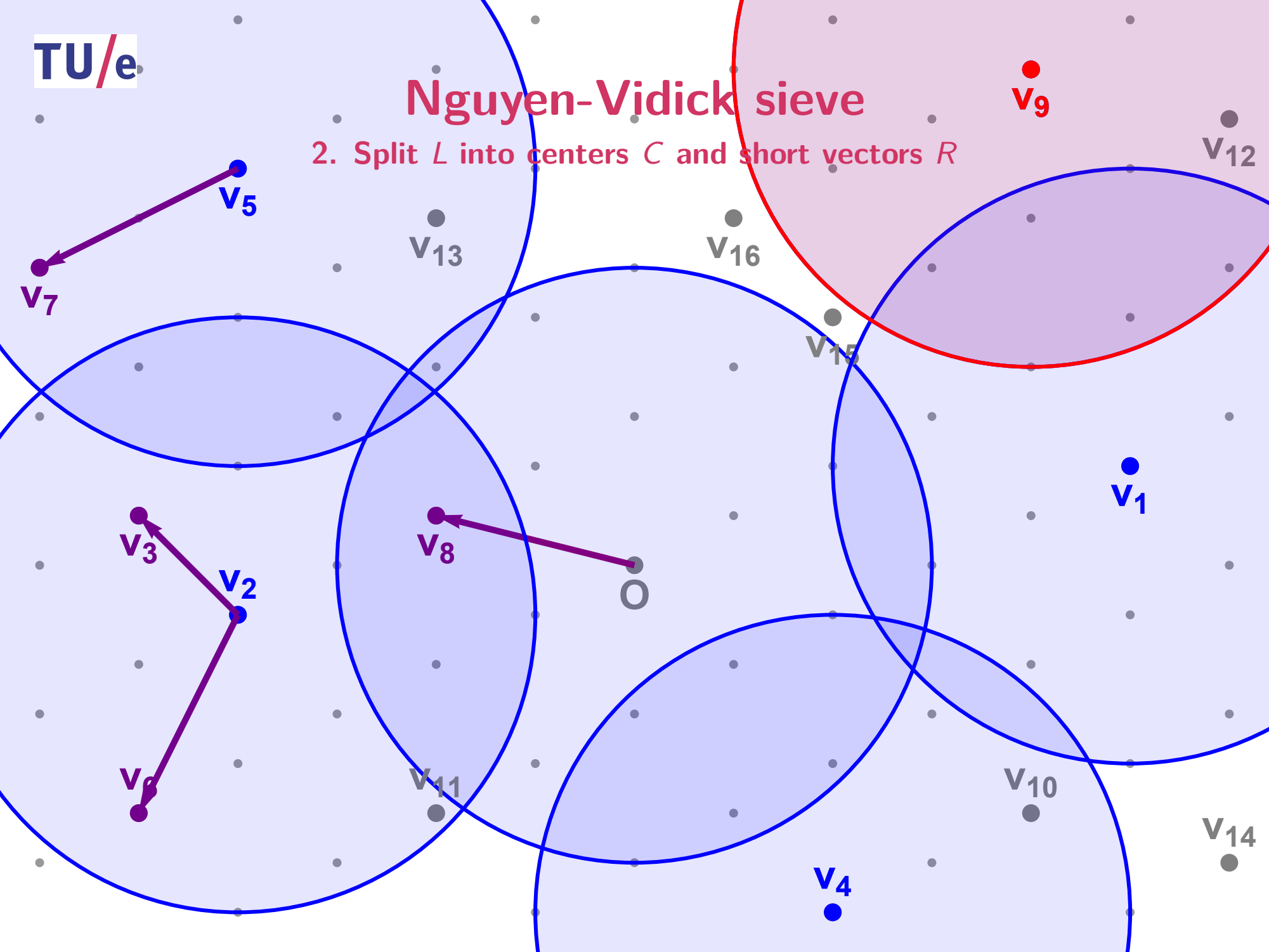

TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_5
v_7
v_{13}
v_{16}
v_9
v_{12}
v_{15}
v_1
v_3
v_2
v_8
O
v_6
v_{11}
v_{10}
v_{14}
v_4

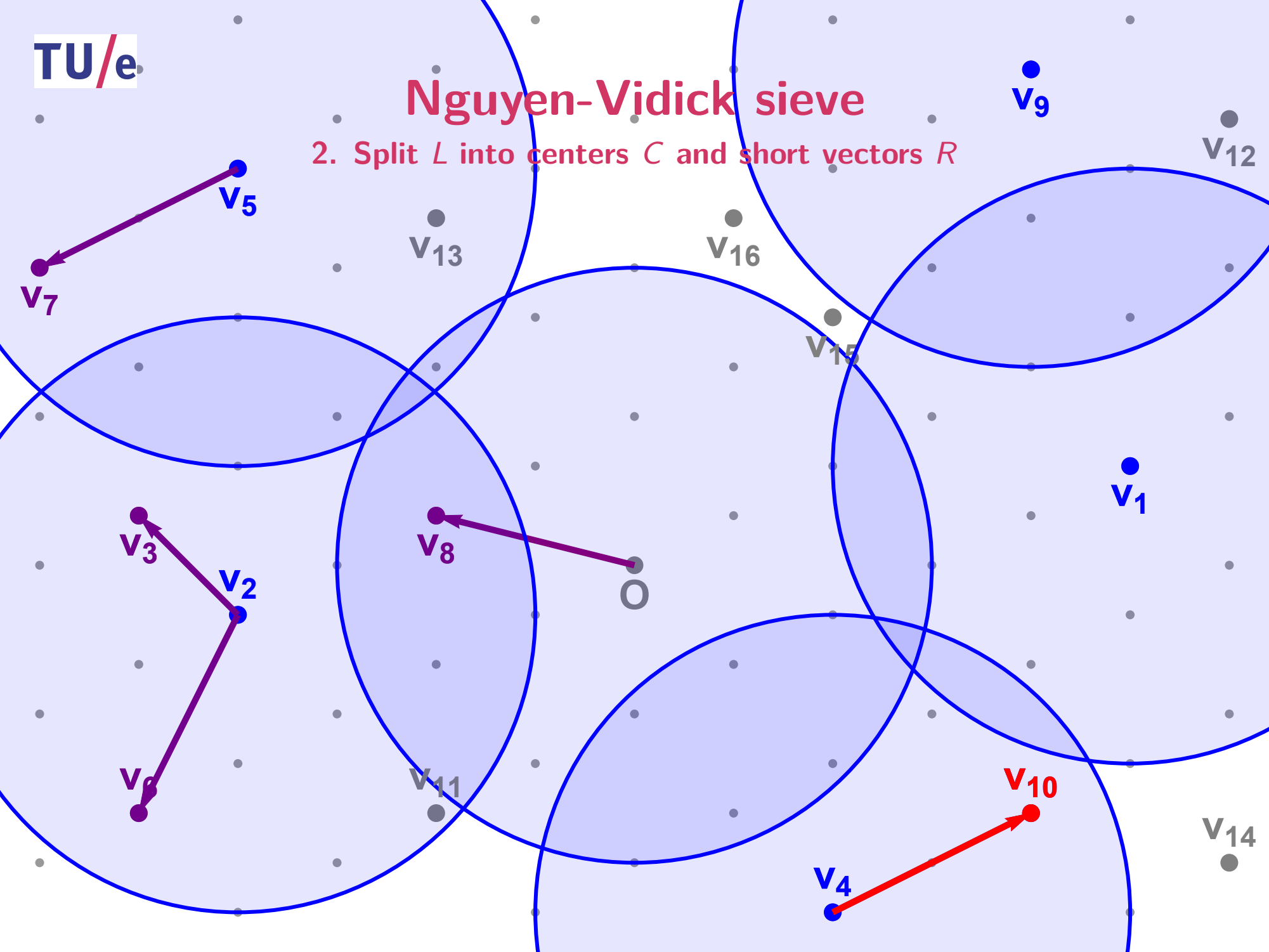
TU/e

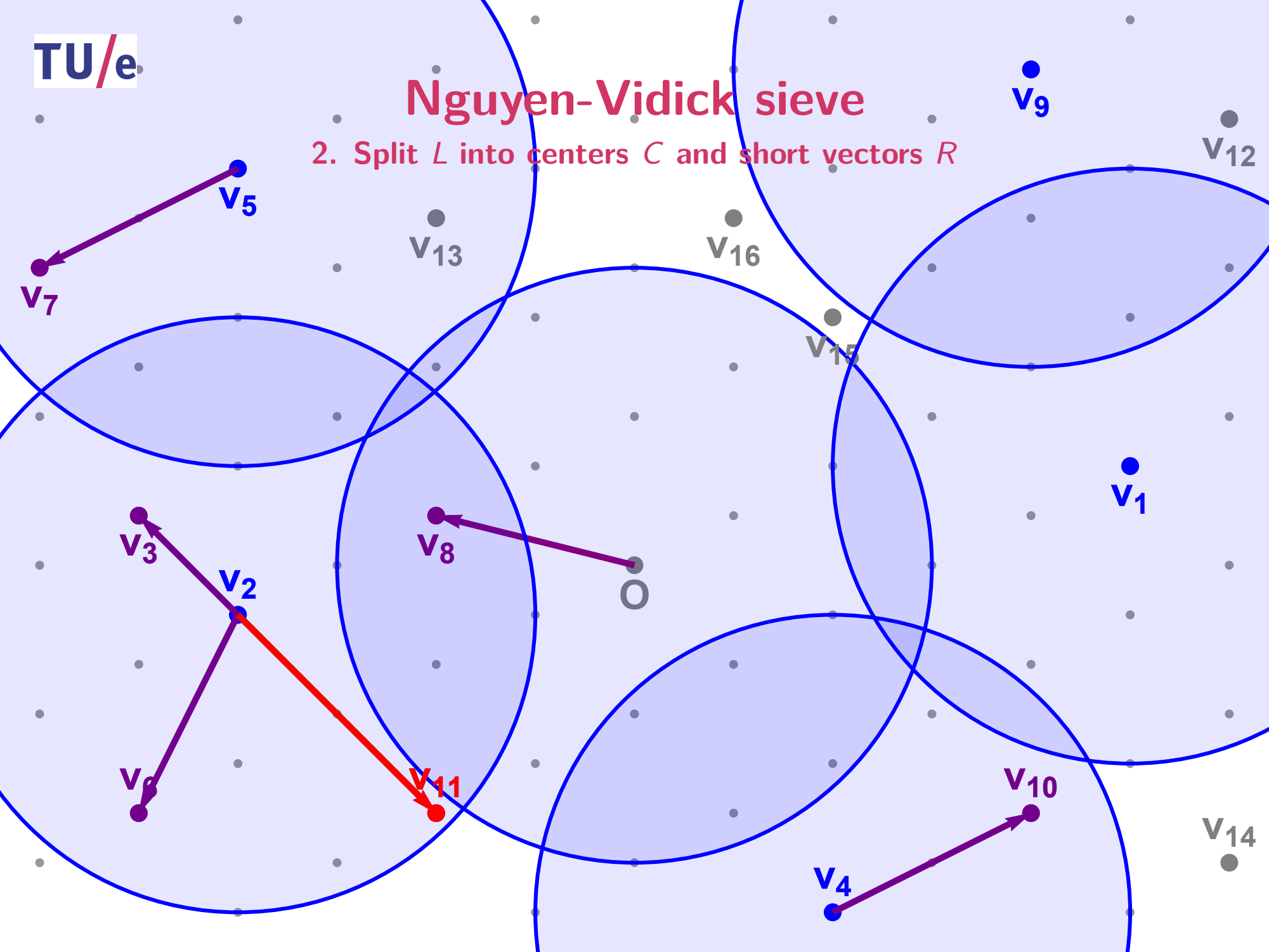
Nguyen-Vidick sieve

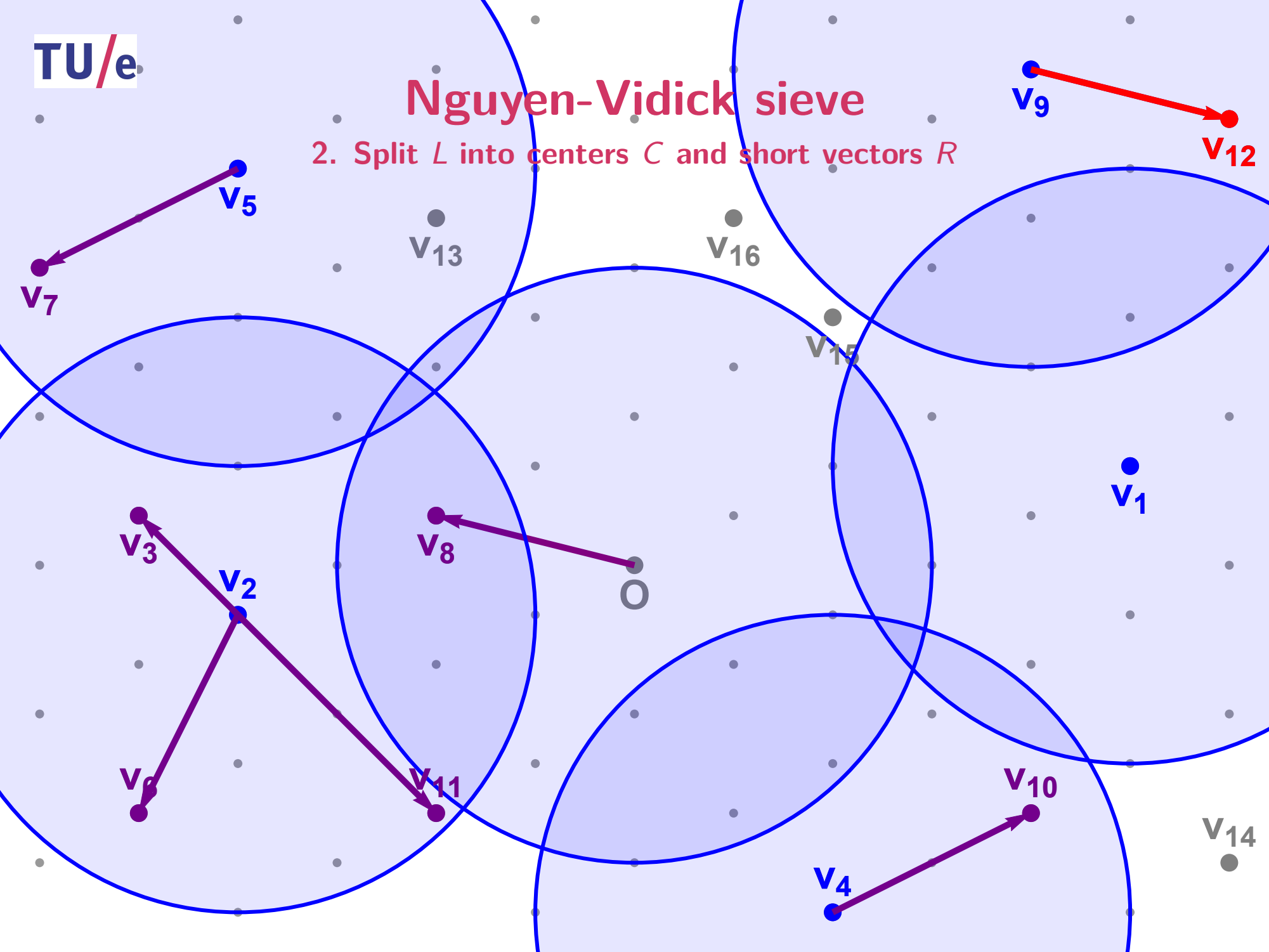
2. Split L into centers C and short vectors R

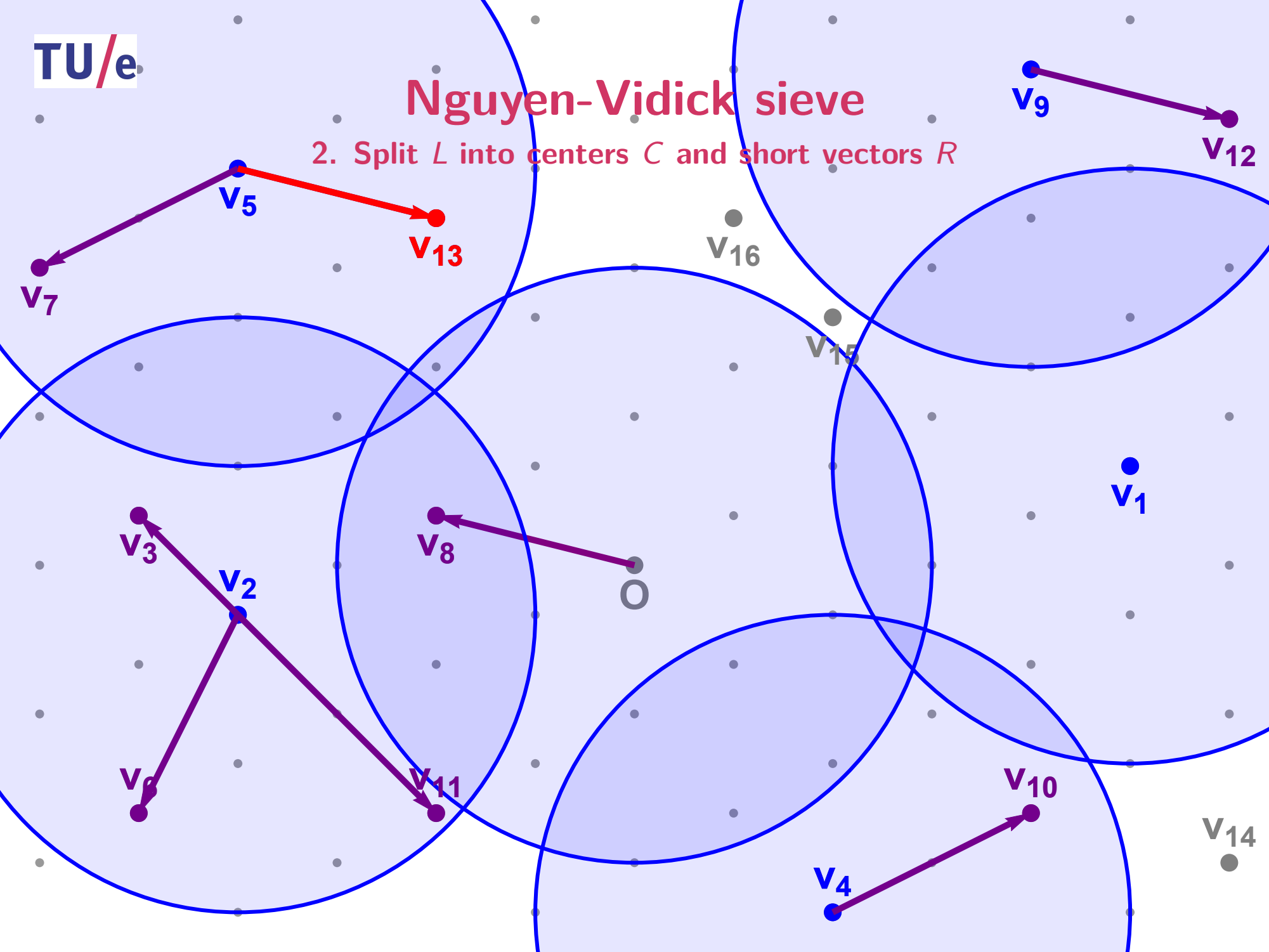



TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4


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

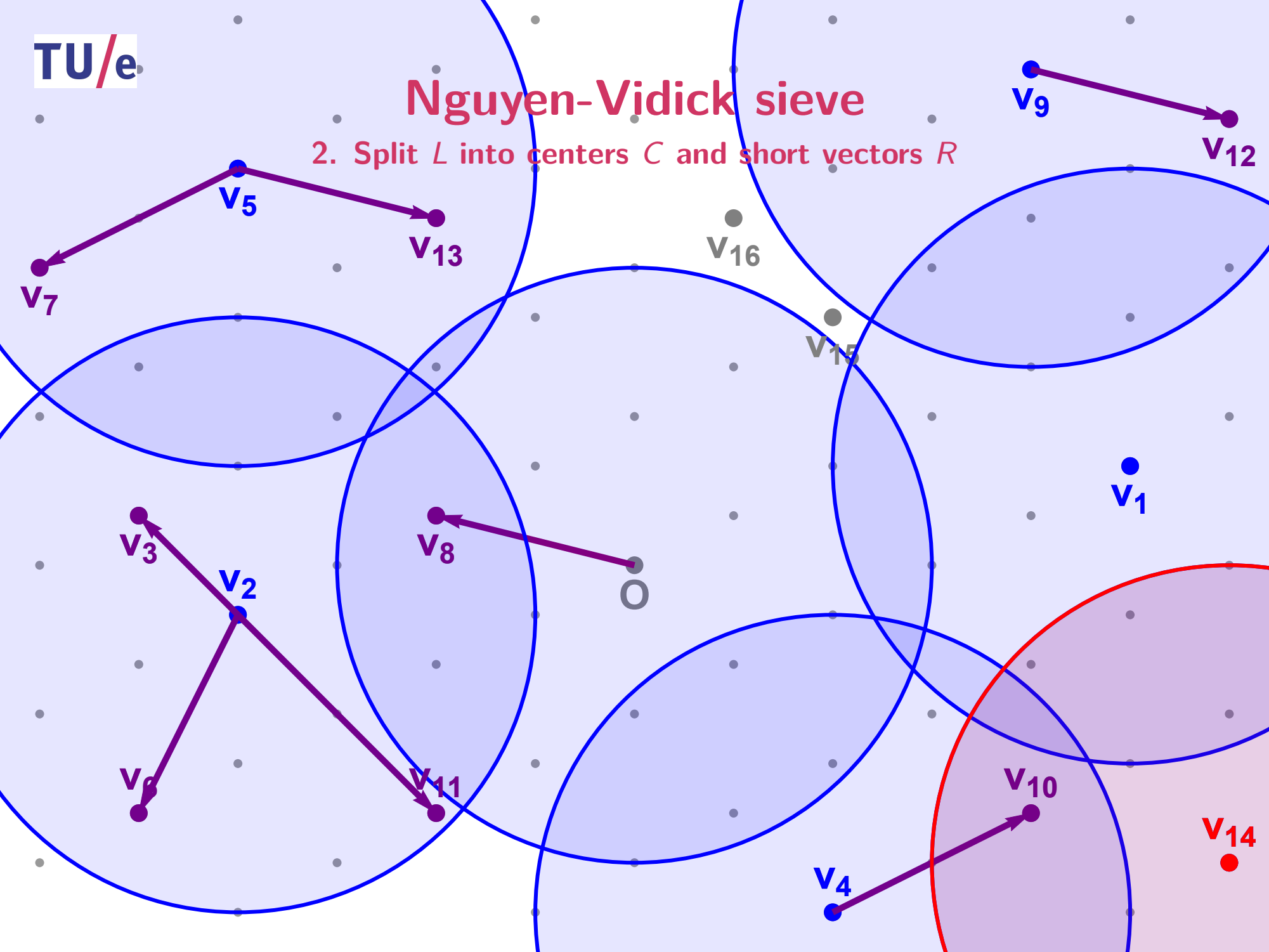

TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4

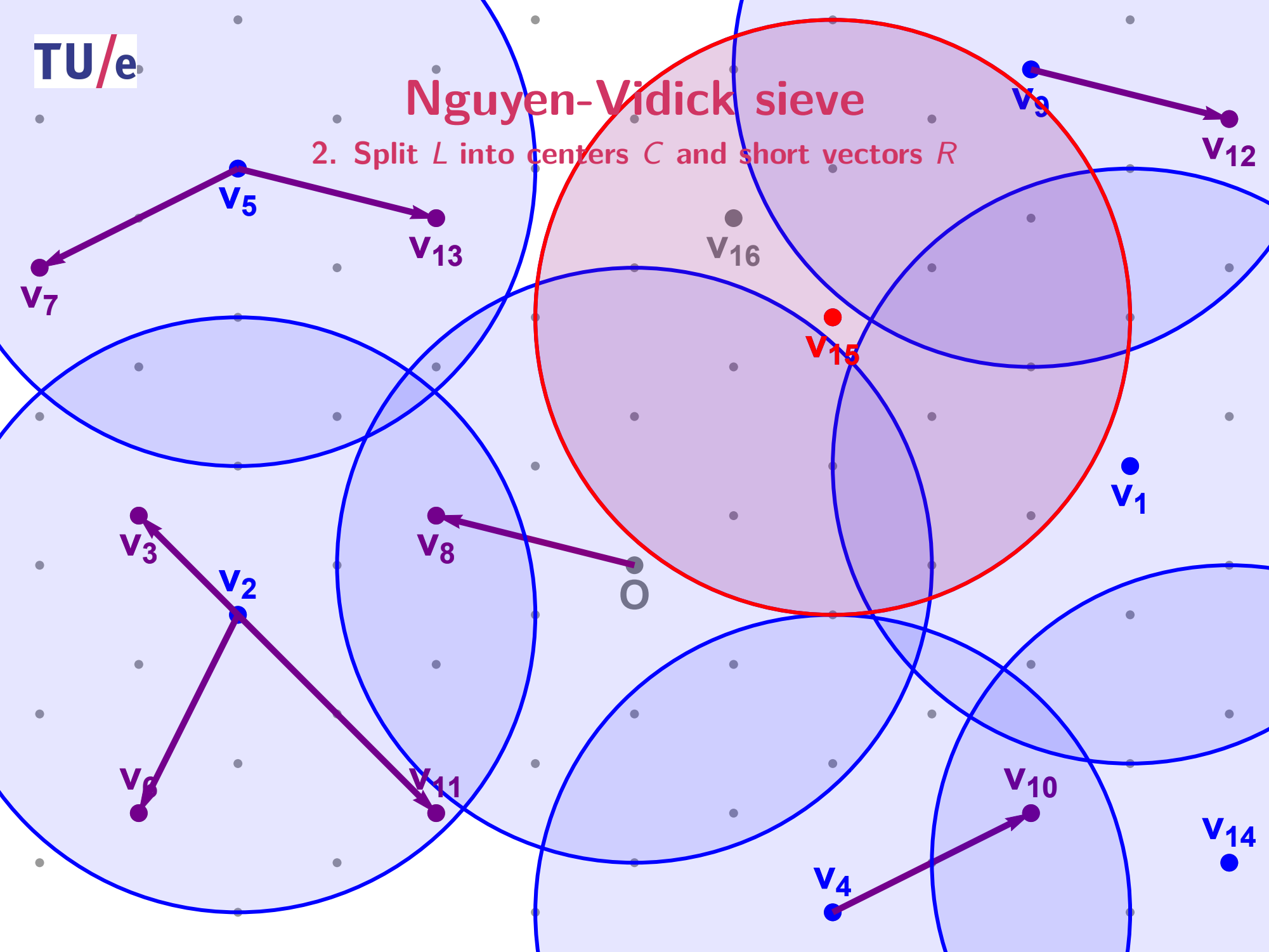

TU/e
Nguyen-Vidick sieve
2. Split L into centers C and short vectors R
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4

TU/e

Nguyen-Vidick sieve

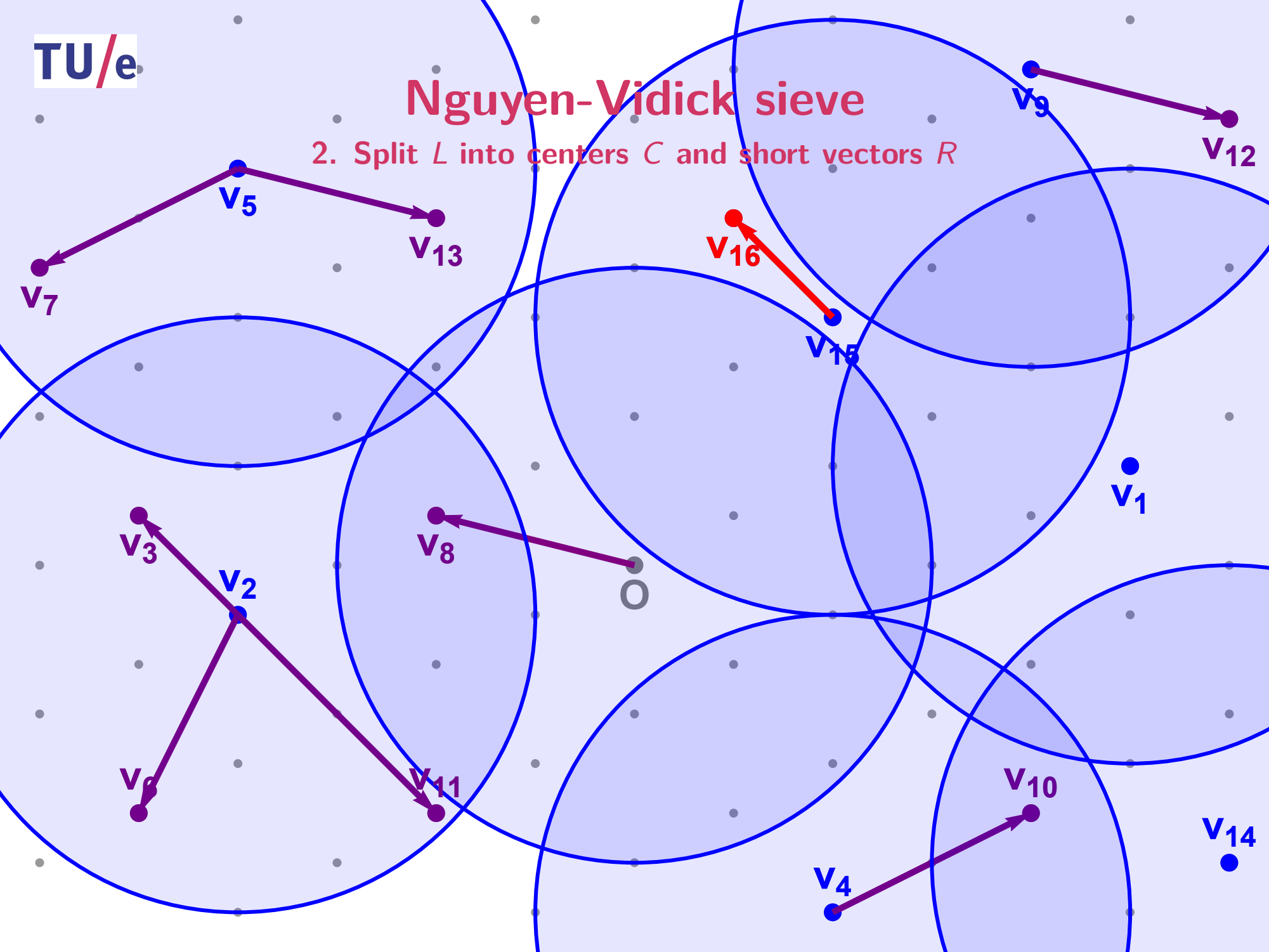
2. Split L into centers C and short vectors R




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

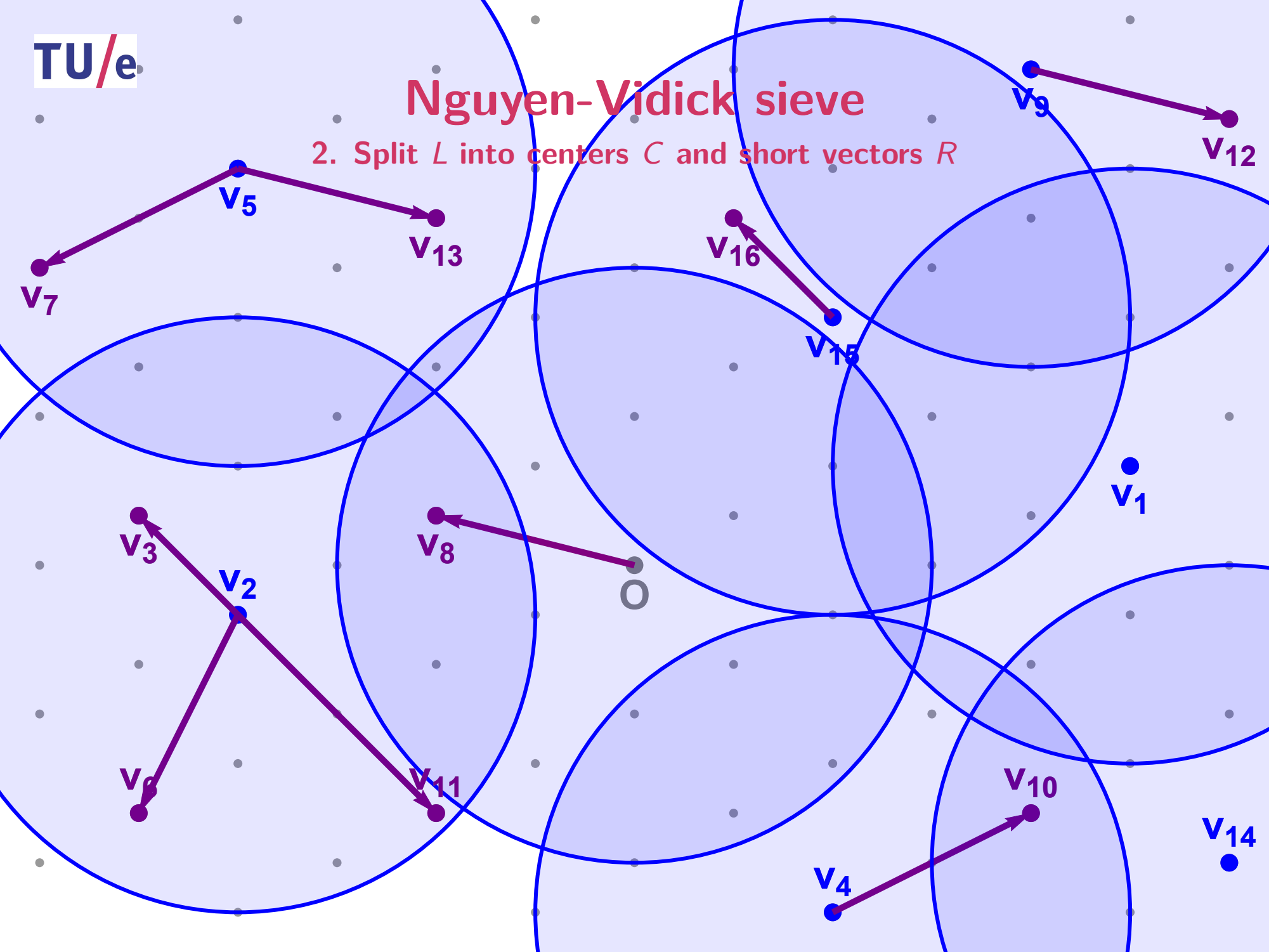
2. Split L into centers C and short vectors R



TU/e

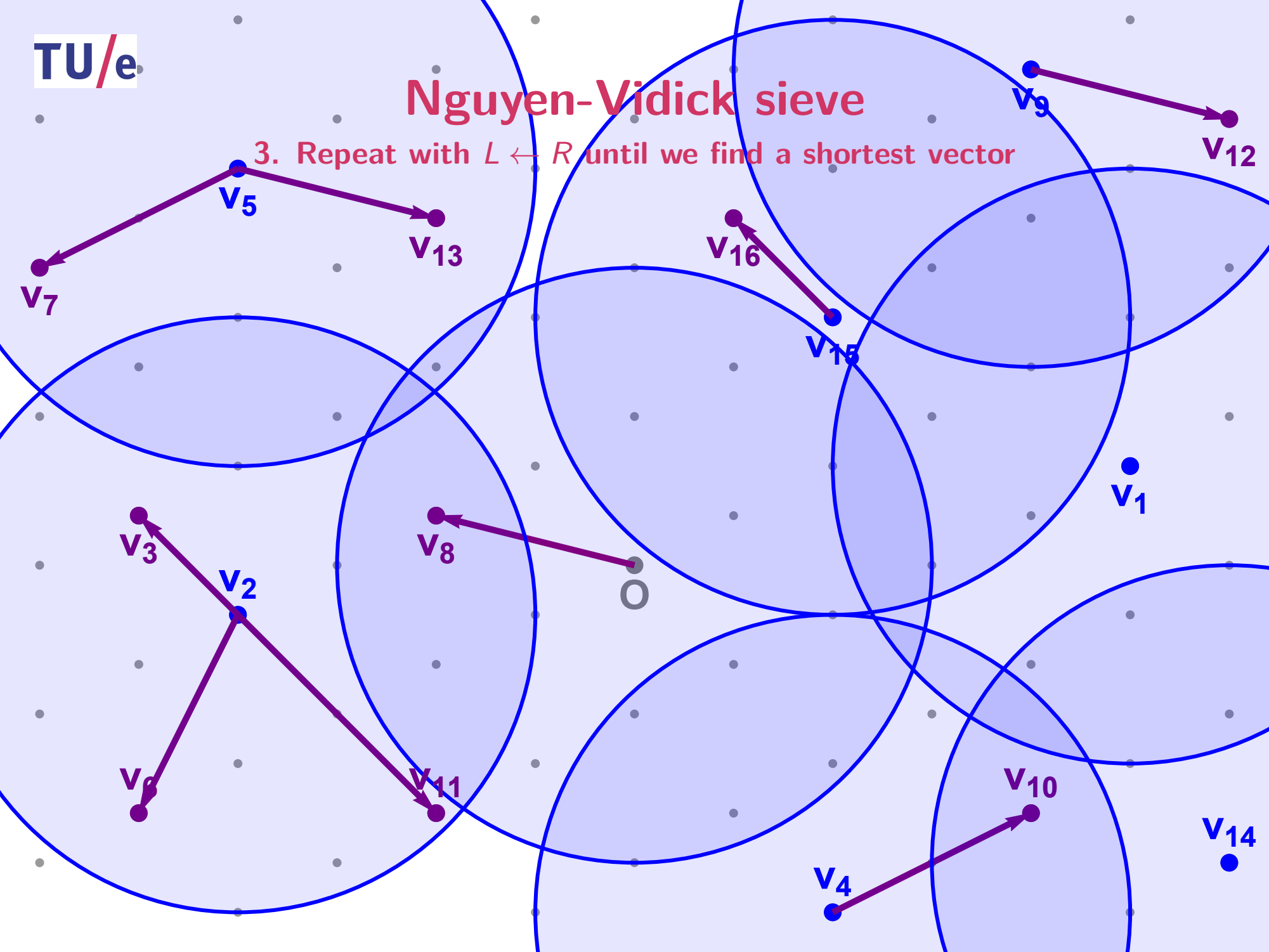
Nguyen-Vidick sieve

2. Split L into centers C and short vectors R



Nguyen-Vidick sieve

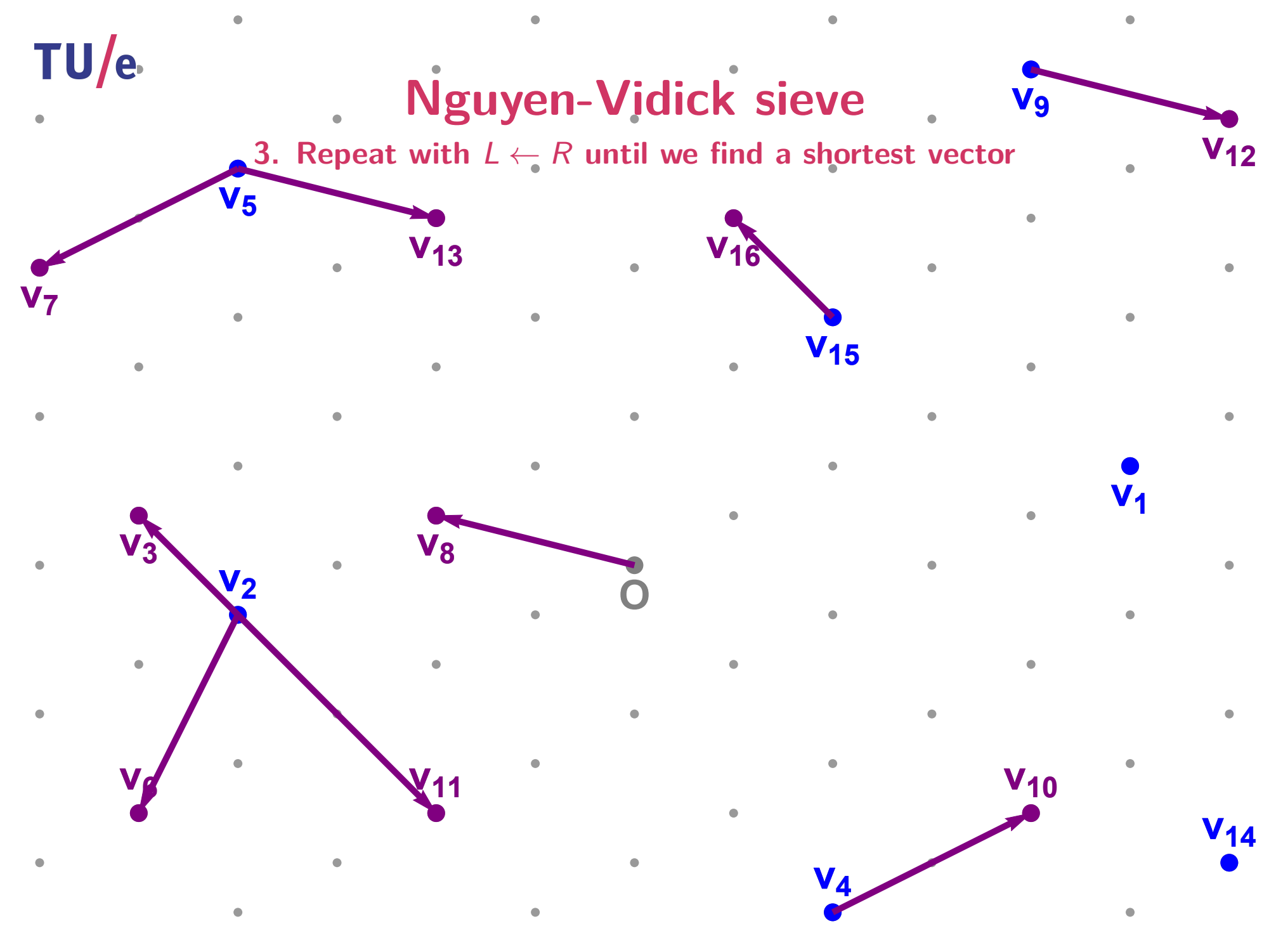
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve

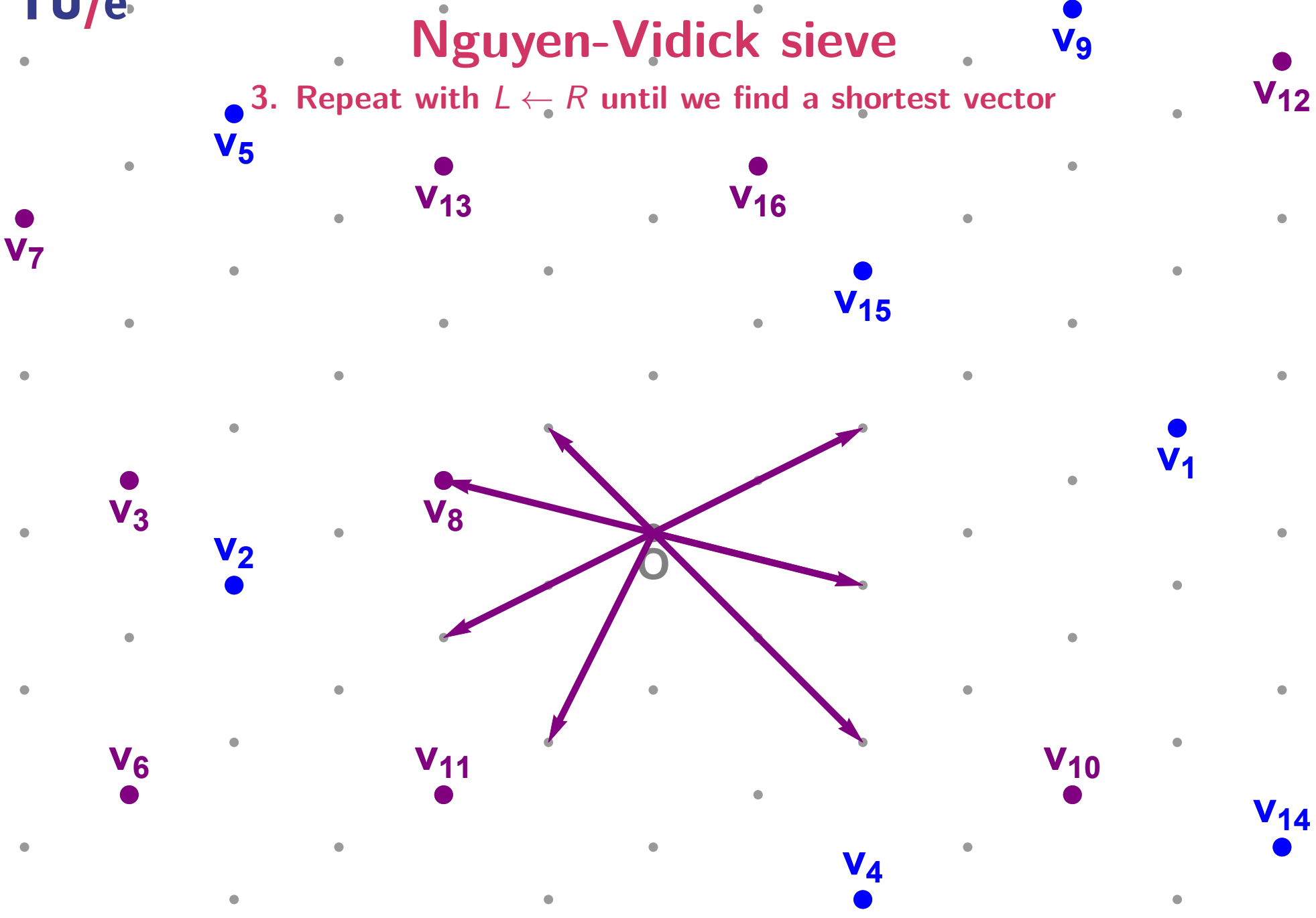
3. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

Nguyen-Vidick sieve

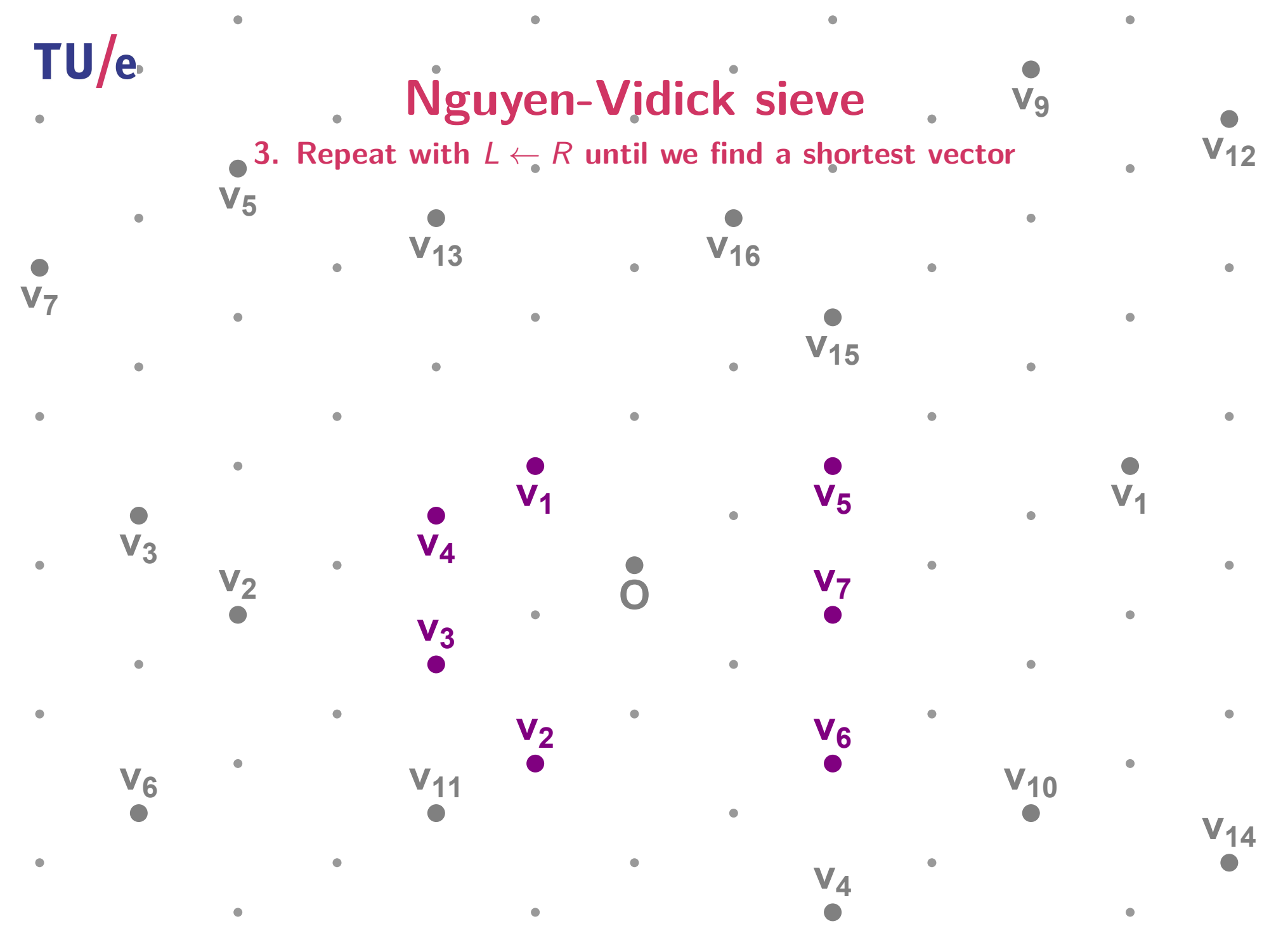
3. Repeat with $L \leftarrow R$ until we find a shortest vector



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

- Space complexity: $\sqrt{4/3}^n \approx 2^{0.21n+o(n)}$ vectors
  - Need $\sqrt{4/3}^n$ vectors to cover all corners of $\mathbb{R}^n$

TU/e

Nguyen-Vidick sieve

Overview

- Space complexity: $\sqrt{4/3}^n \approx 2^{0.21n+o(n)}$ vectors
  - Need $\sqrt{4/3}^n$ vectors to cover all corners of $\mathbb{R}^n$
- Time complexity: $(4/3)^n \approx 2^{0.42n+o(n)}$
  - Comparing a target vector to all centers: $2^{0.21n+o(n)}$
  - Repeating this for each list vector: $2^{0.21n+o(n)}$
  - Repeating the whole sieving procedure: $\mathrm{poly}(n)$

TU/e

Nguyen-Vidick sieve

Overview

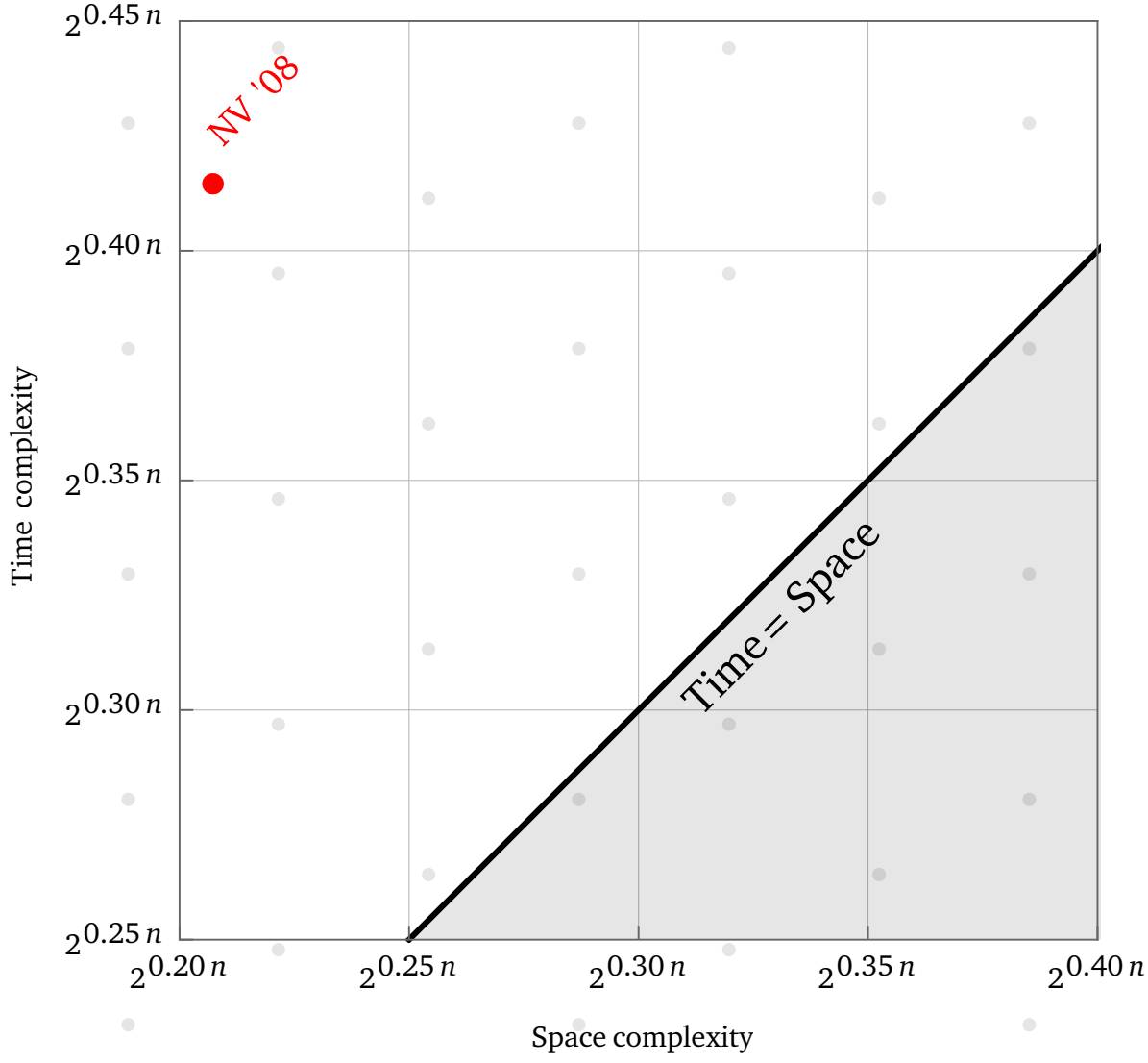
- Space complexity: $\sqrt{4/3}^n \approx 2^{0.21n+o(n)}$ vectors
  - Need $\sqrt{4/3}^n$ vectors to cover all corners of $\mathbb{R}^n$
- Time complexity: $(4/3)^n \approx 2^{0.42n+o(n)}$
  - Comparing a target vector to all centers: $2^{0.21n+o(n)}$
  - Repeating this for each list vector: $2^{0.21n+o(n)}$
  - Repeating the whole sieving procedure: $\mathrm{poly}(n)$

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

The NV-sieve runs in time $2^{0.42n+o(n)}$ and space $2^{0.21n+o(n)}$.

Nguyen-Vidick sieve

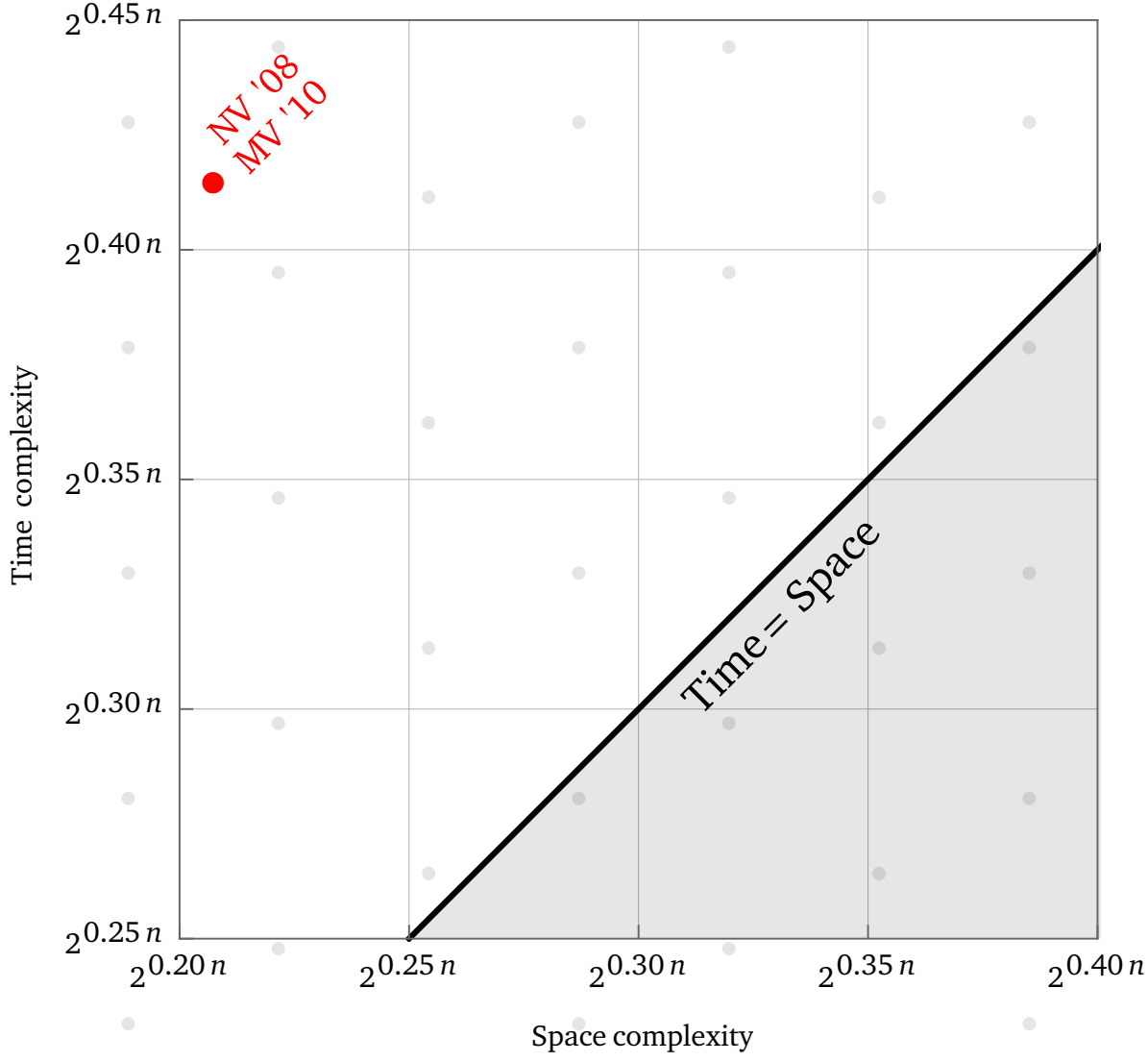
Space/time trade-off



TU/e

GaussSieve

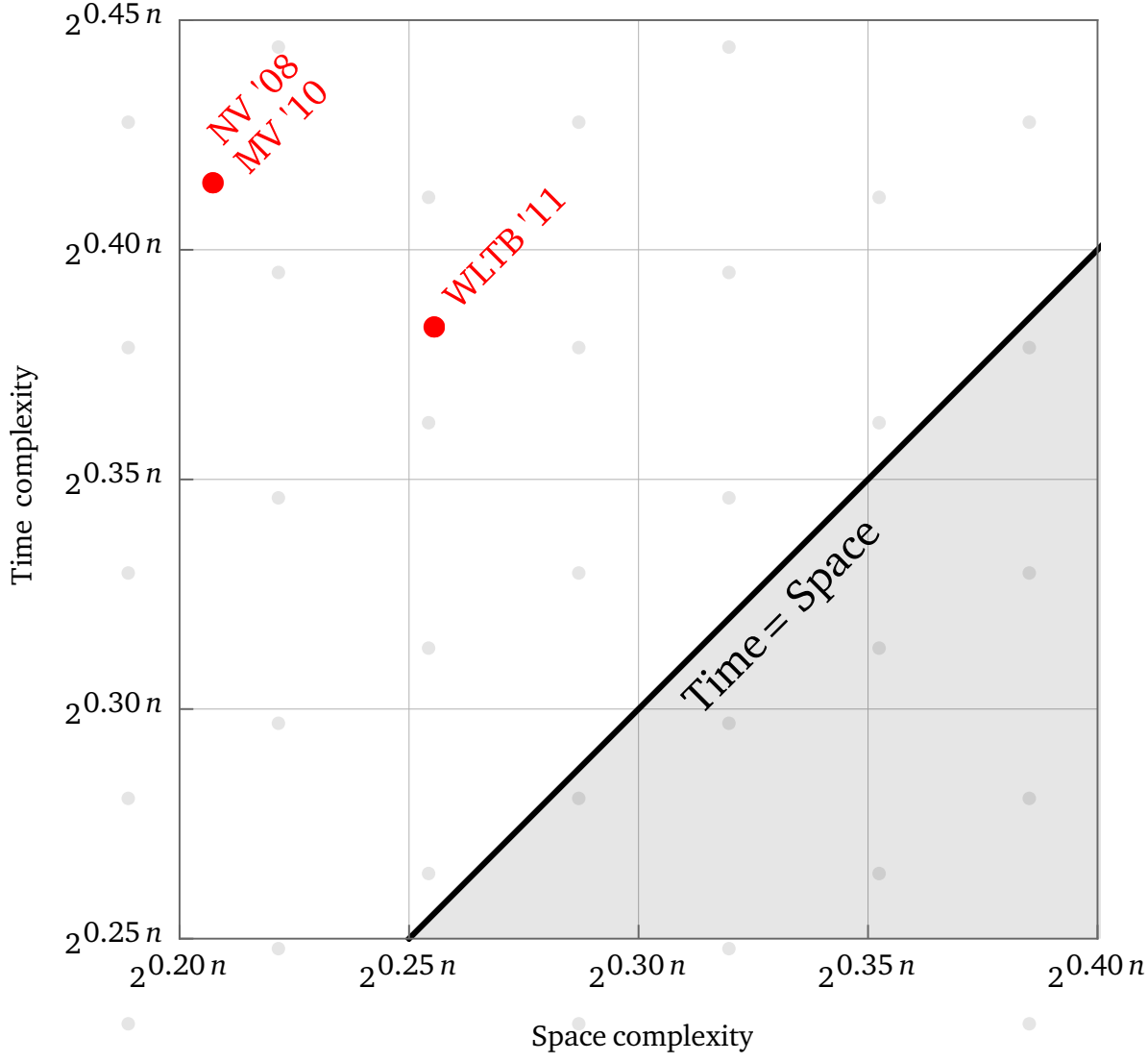
Space/time trade-off



TU/e

Two-level sieving

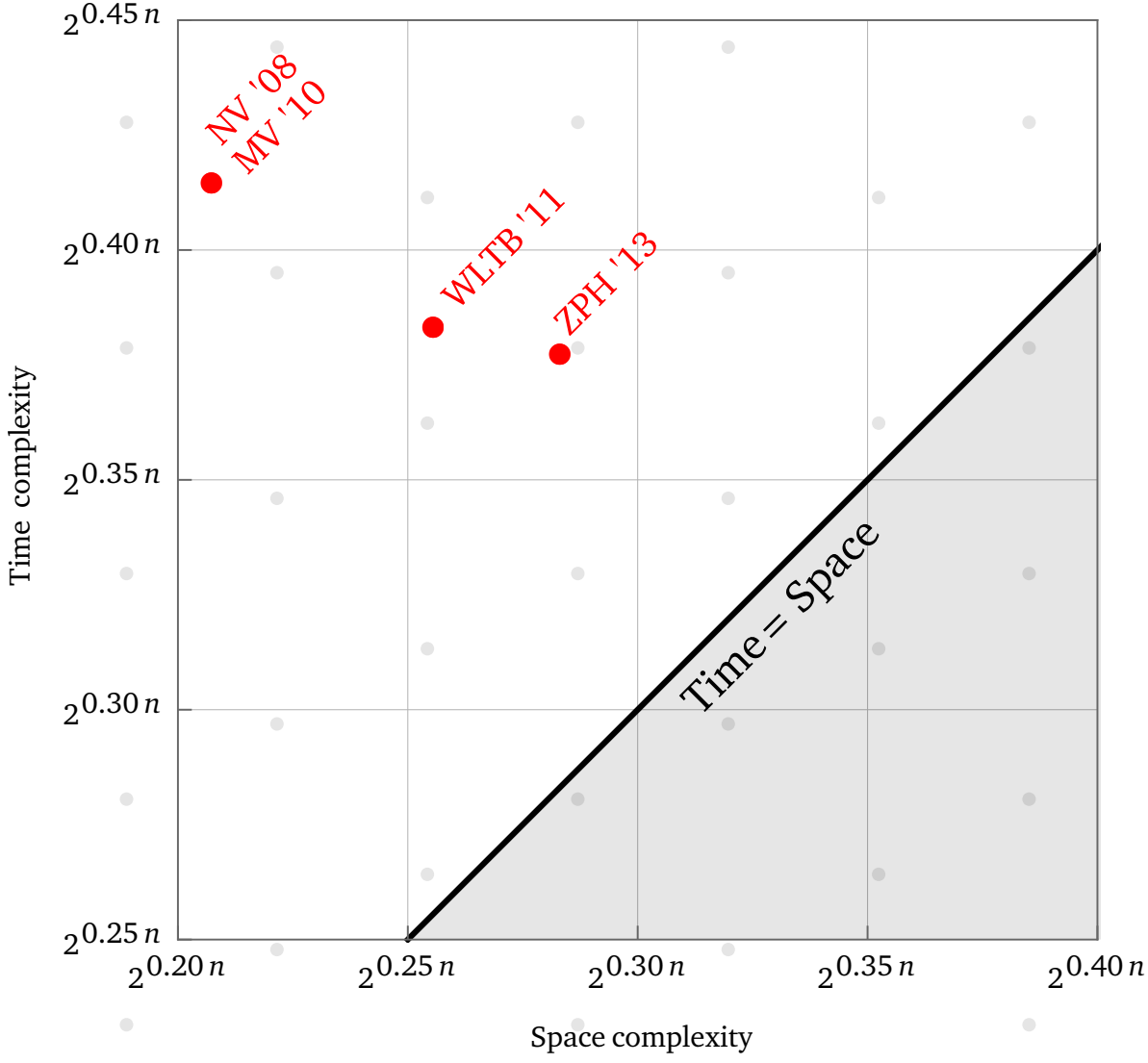
Space/time trade-off



TU/e

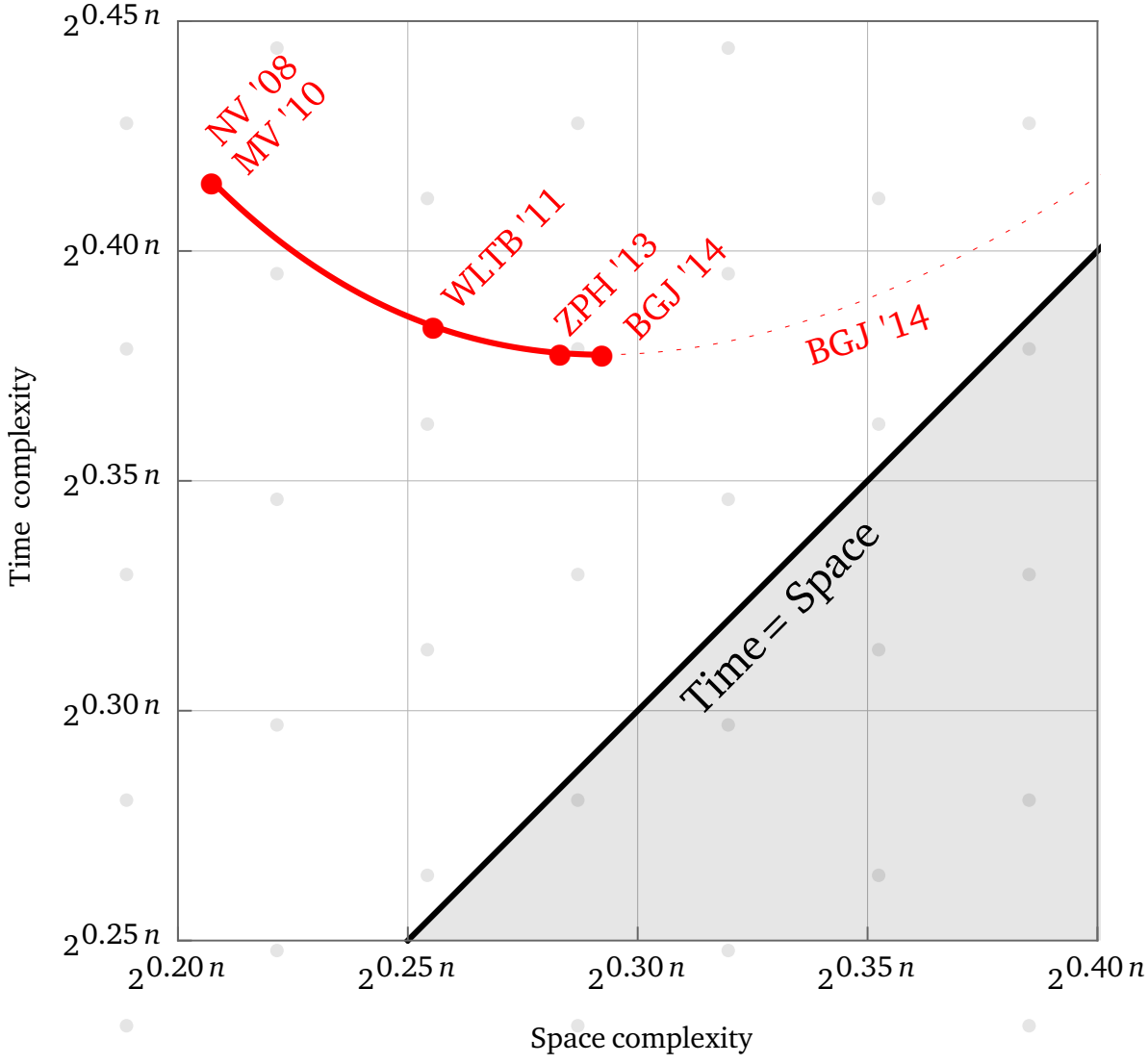
Three-level sieving

Space/time trade-off



Decomposition approach

Space/time trade-off



TU/e

HashSieve

1. Sample a list L of random lattice vectors

TU/e

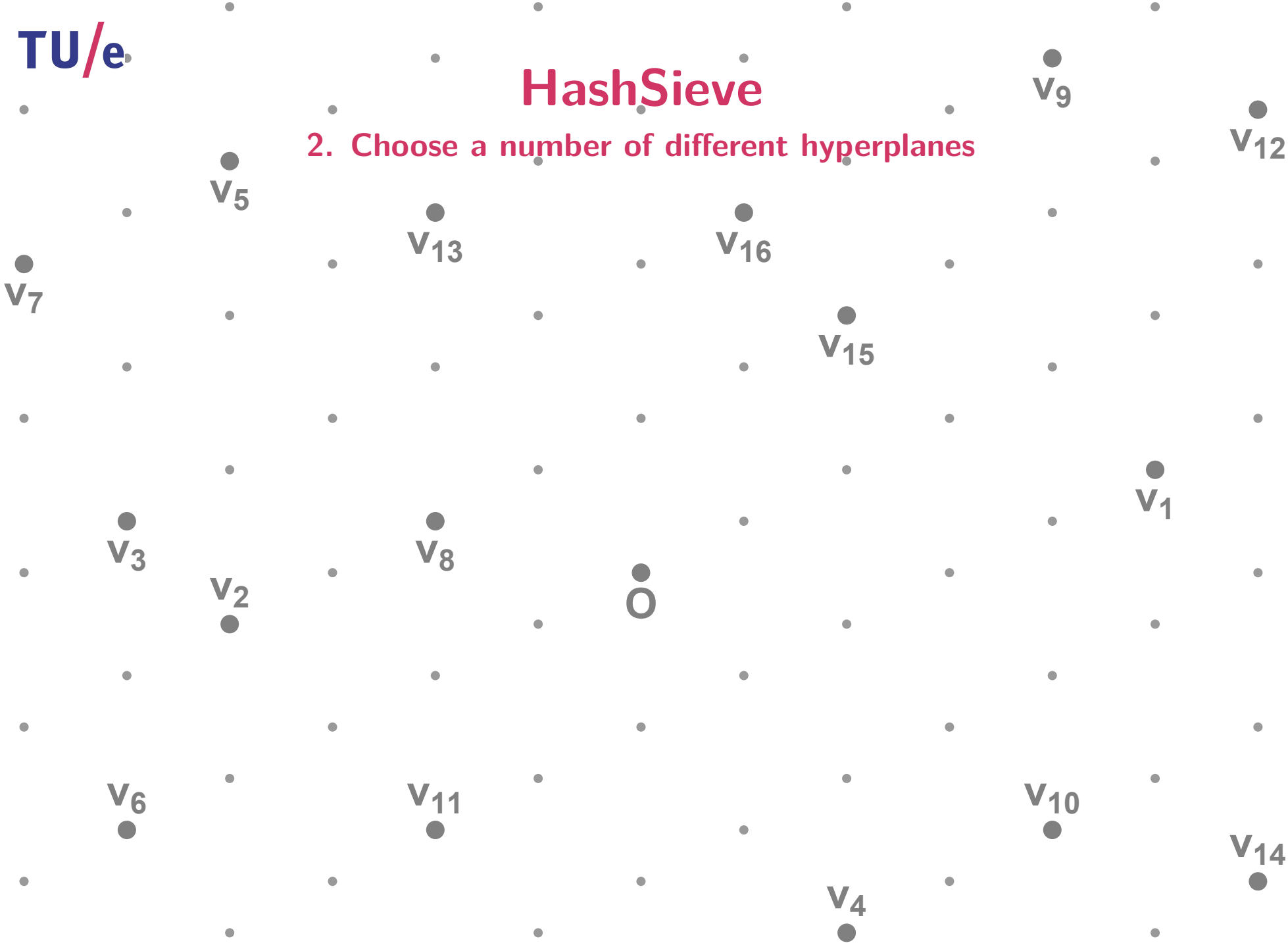
HashSieve

1. Sample a list L of random lattice vectors

TU/e

HashSieve

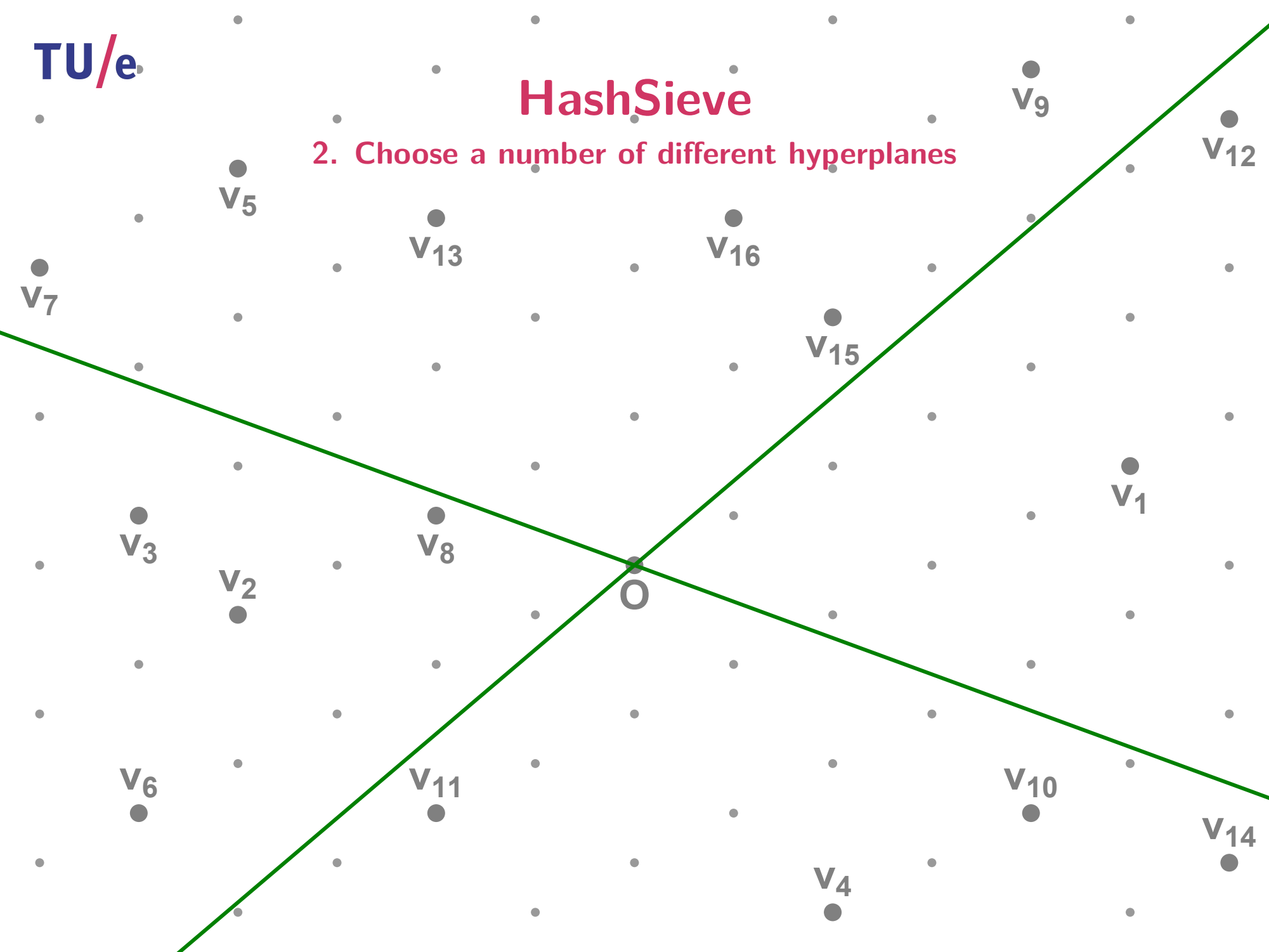
2. Choose a number of different hyperplanes



TU/e

HashSieve

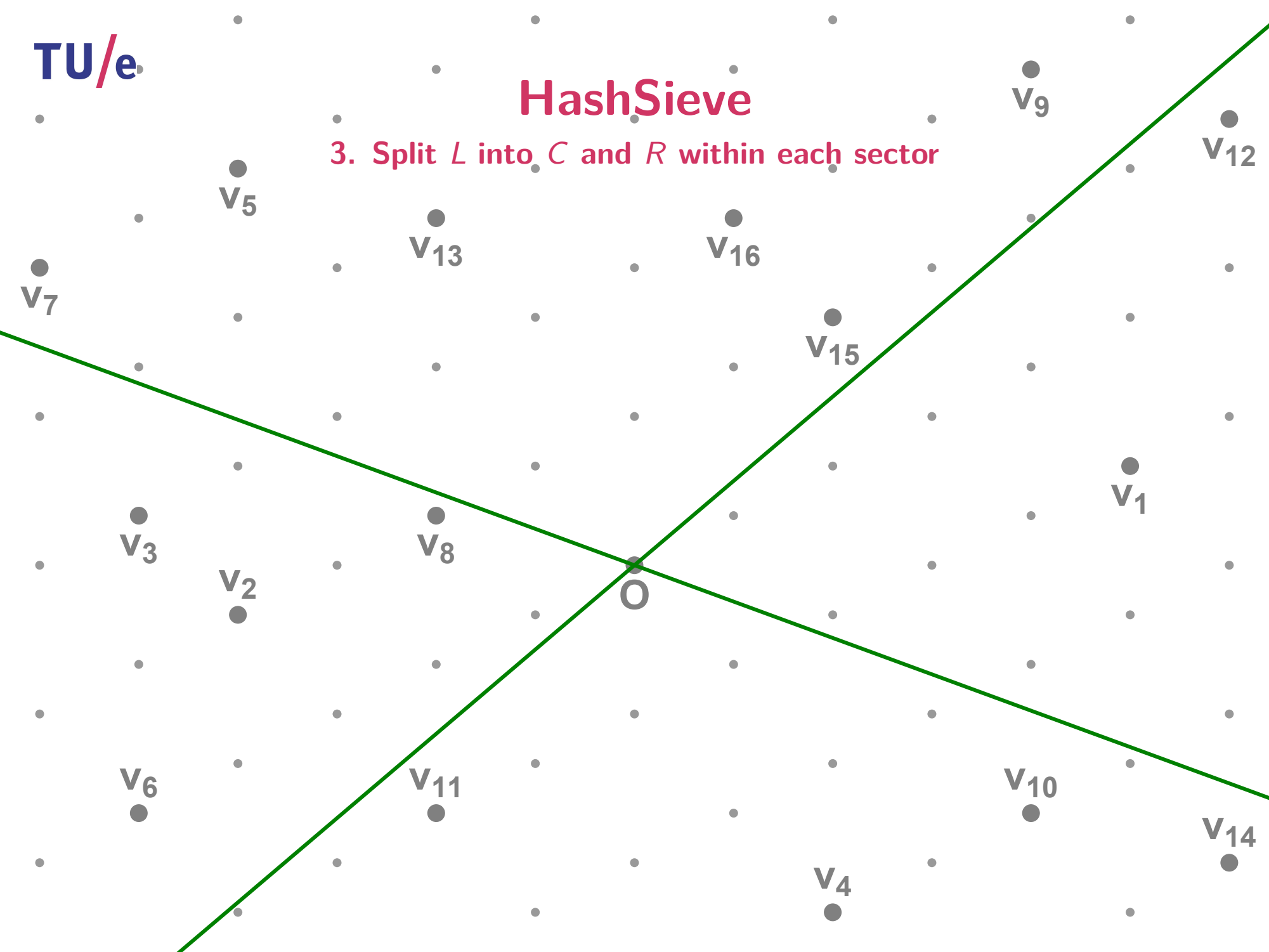
2. Choose a number of different hyperplanes



TU/e

HashSieve

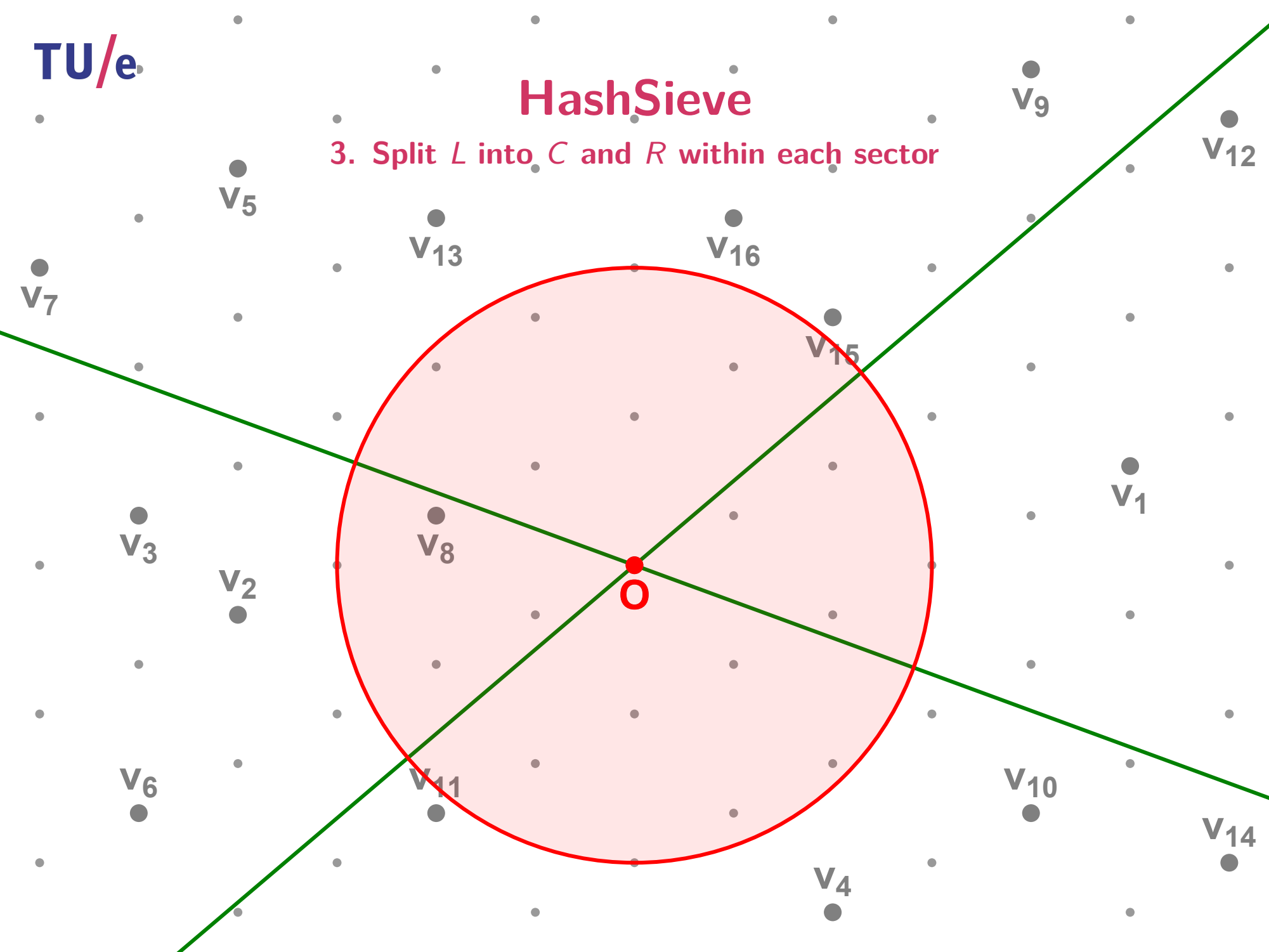
3. Split L into C and R within each sector



TU/e

HashSieve

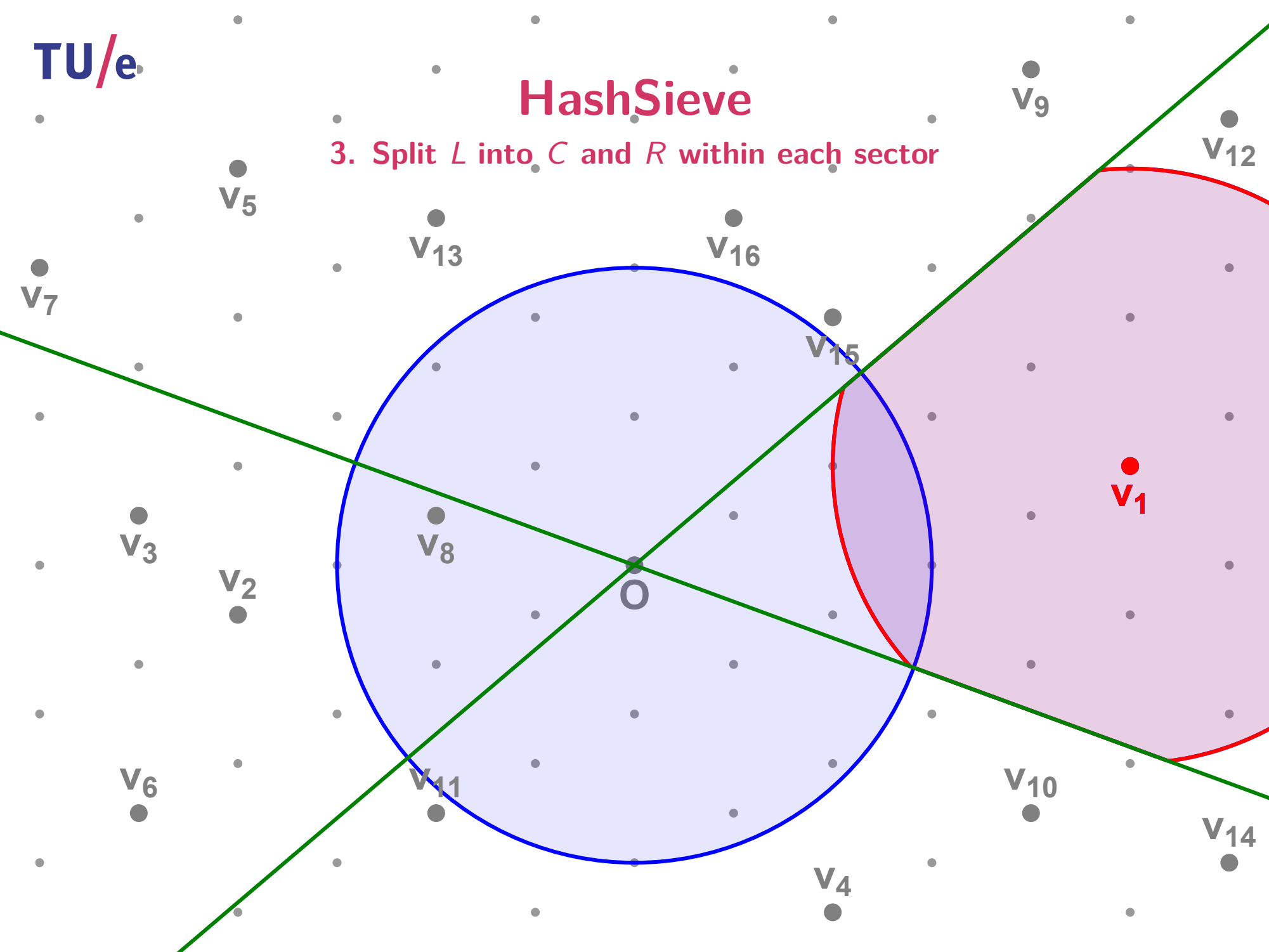
3. Split L into C and R within each sector



TU/e

HashSieve

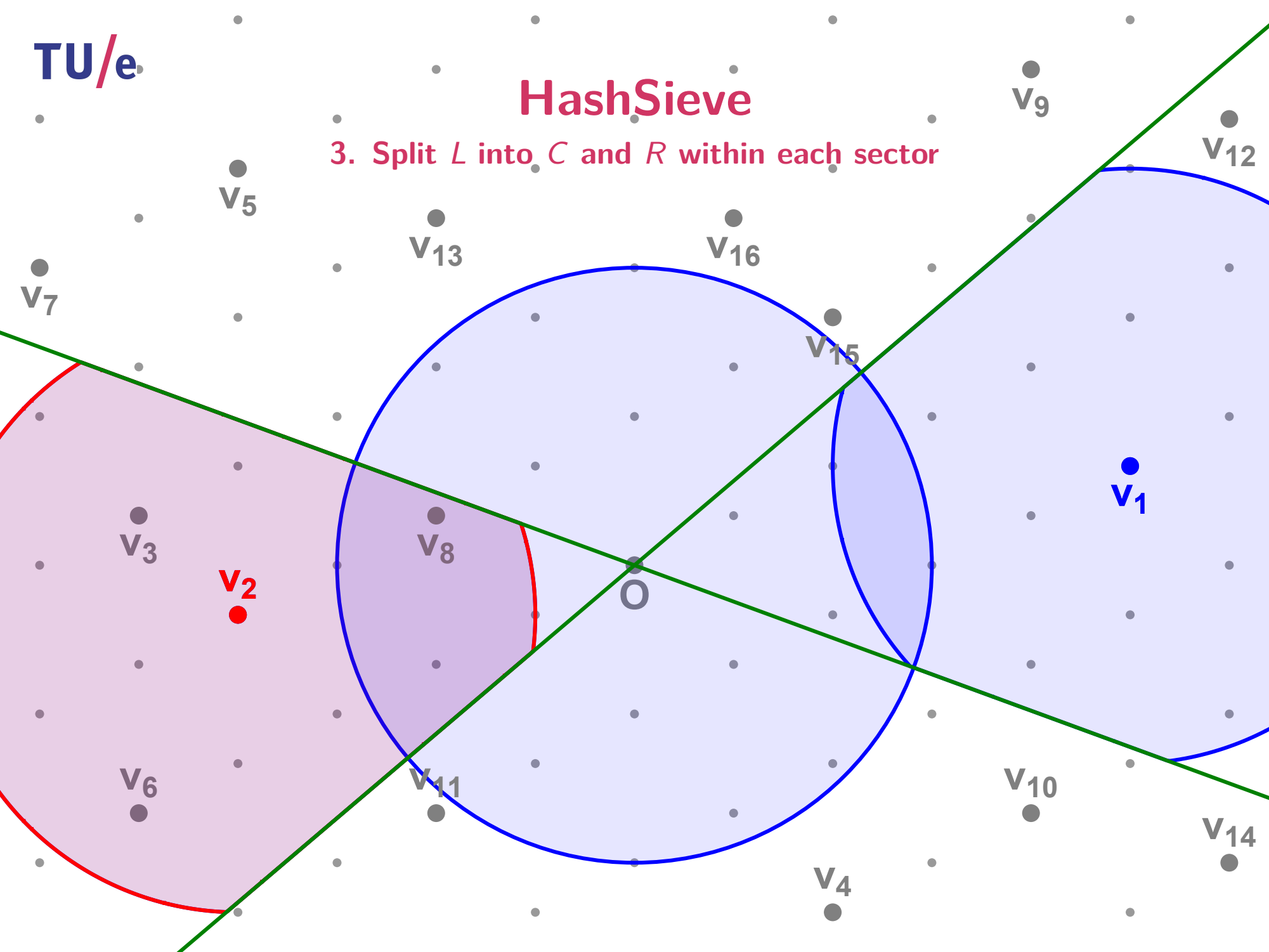
3. Split L into C and R within each sector



TU/e

HashSieve

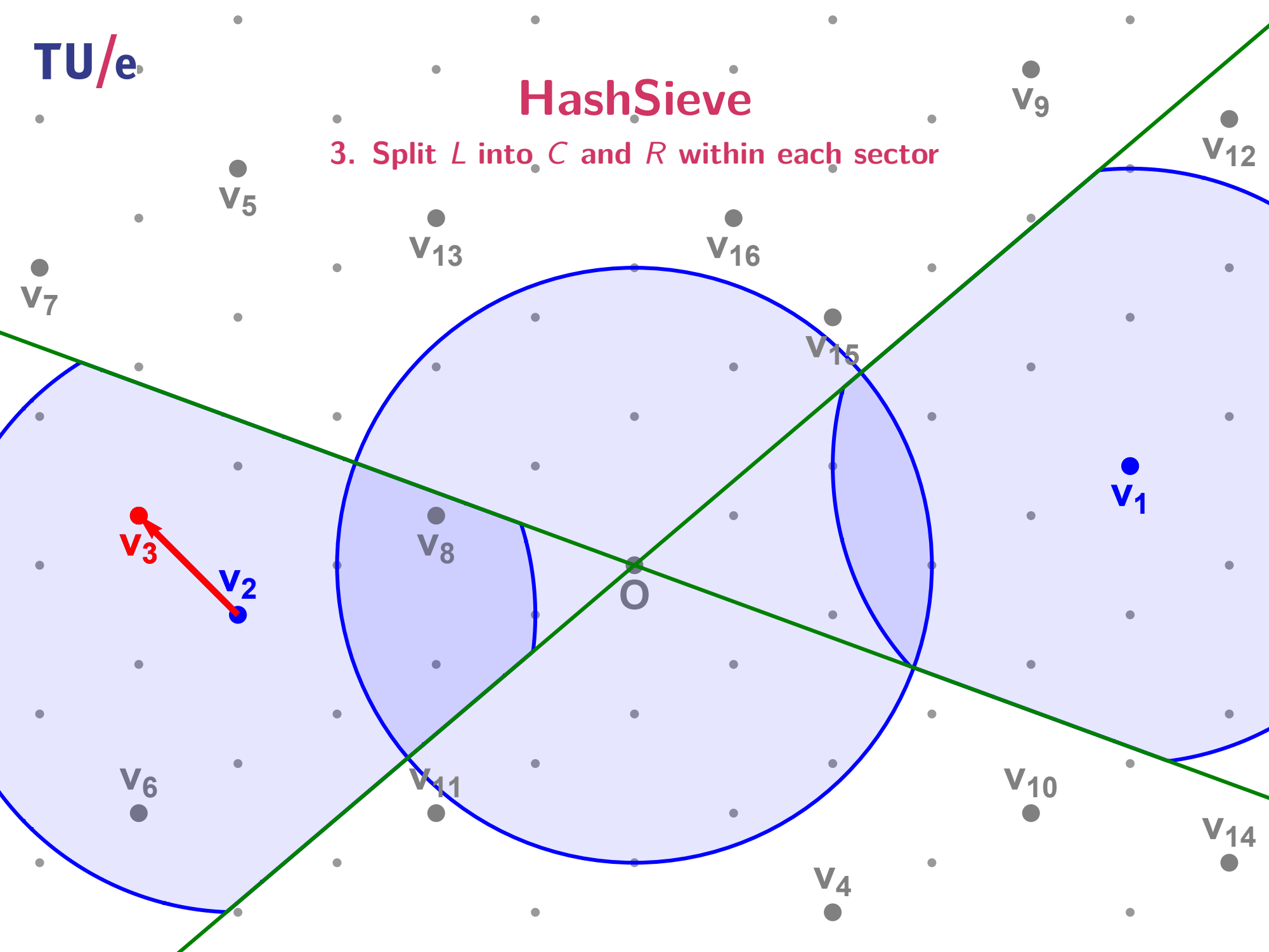
3. Split L into C and R within each sector



TU/e

HashSieve

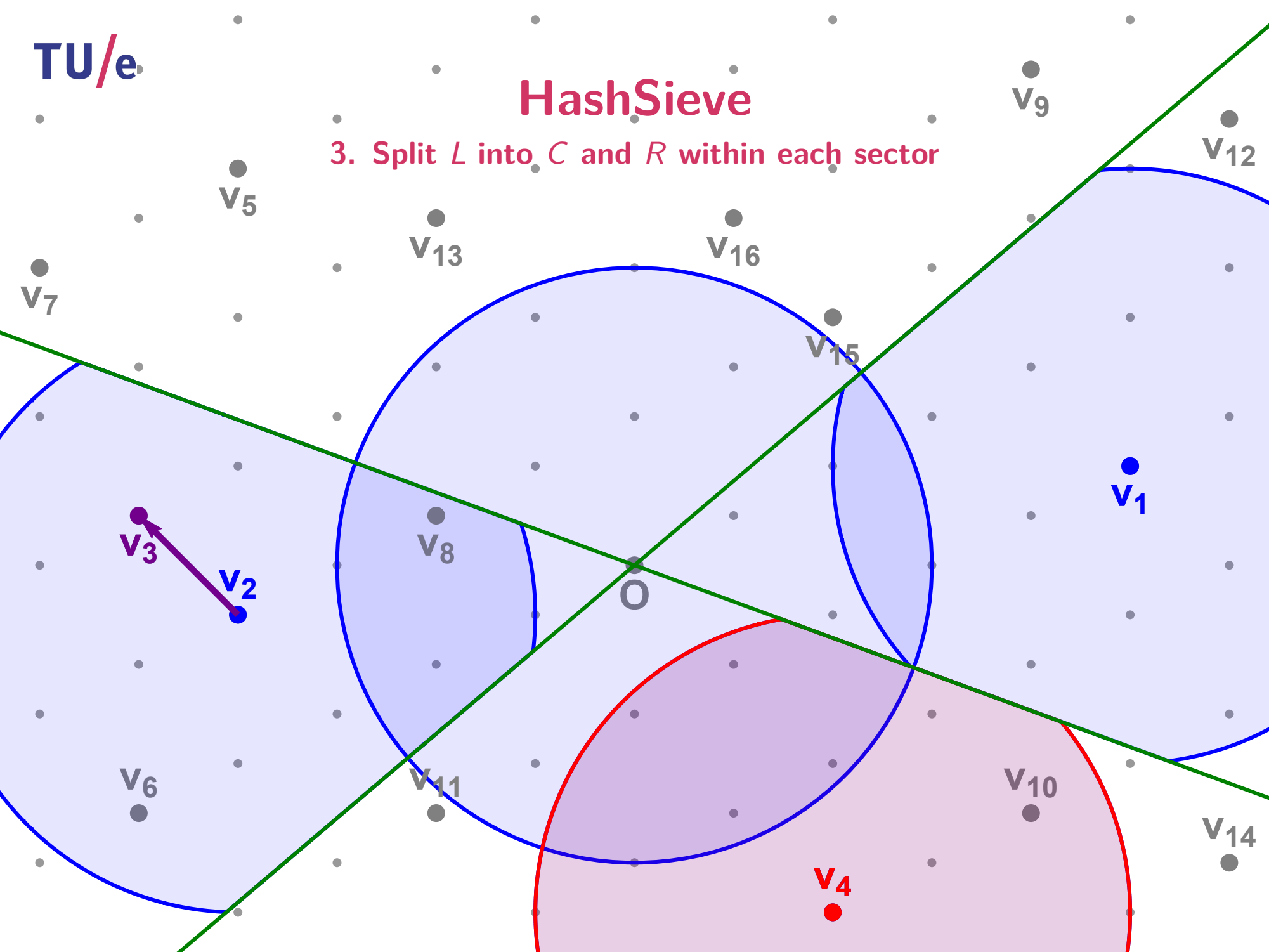
3. Split L into C and R within each sector

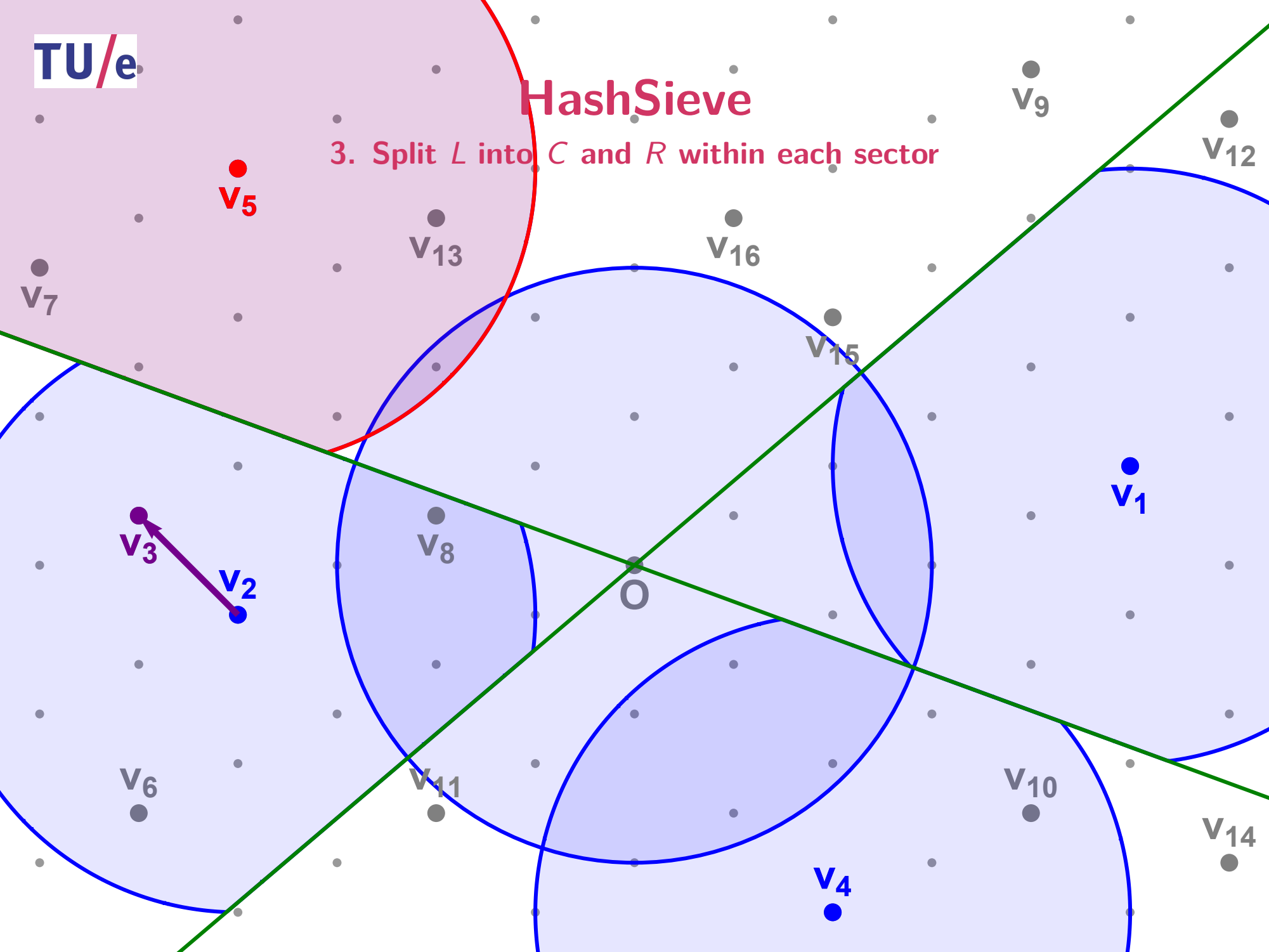


TU/e

HashSieve

3. Split L into C and R within each sector

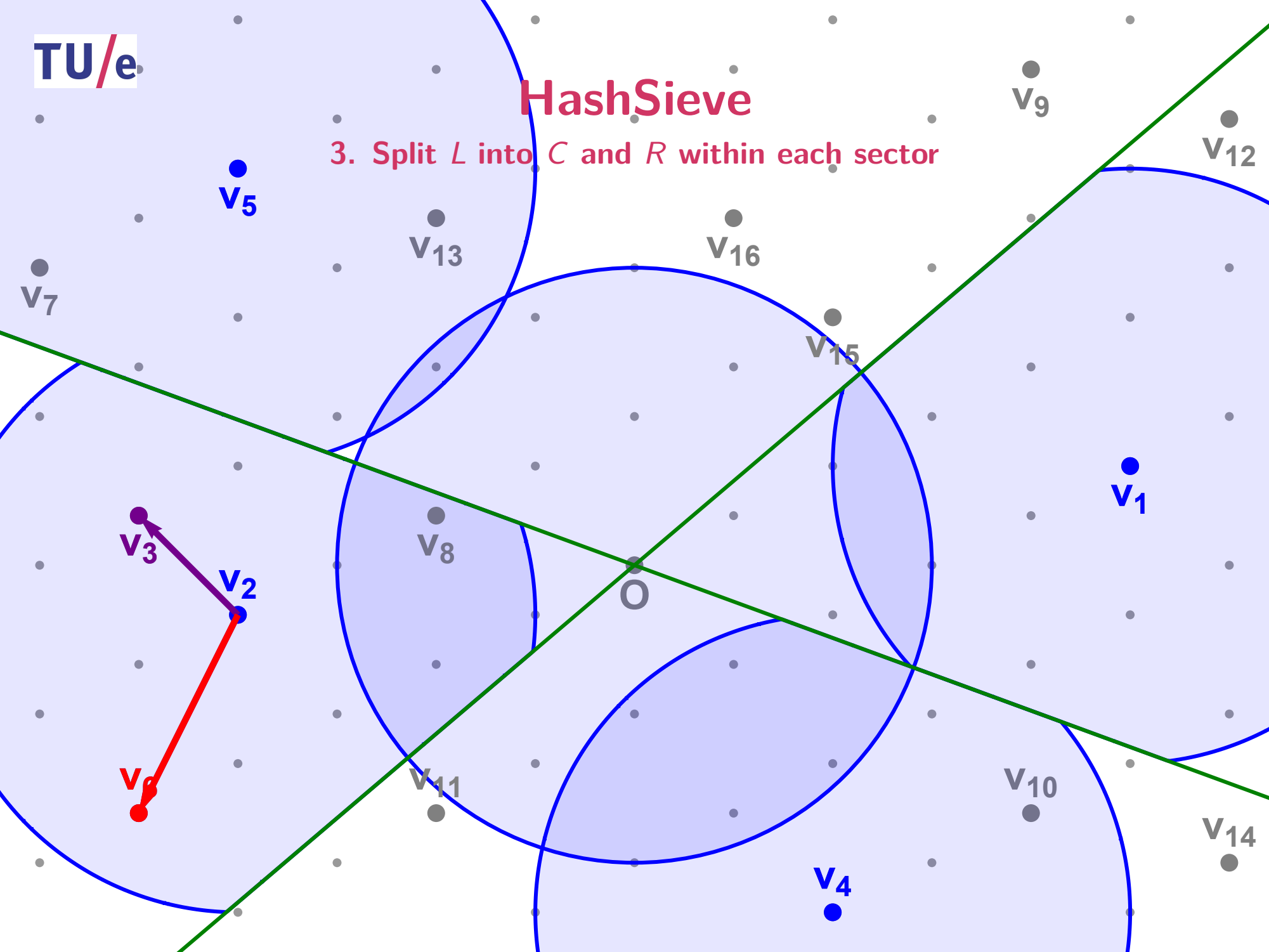



TU/e
HashSieve
3. Split L into C and R within each sector
v_5
v_{13}
v_{16}
v_9
v_{12}
v_7
v_{15}
v_1
v_3
v_2
v_8
O
v_6
v_{11}
v_{10}
v_{14}
v_4

TU/e

HashSieve

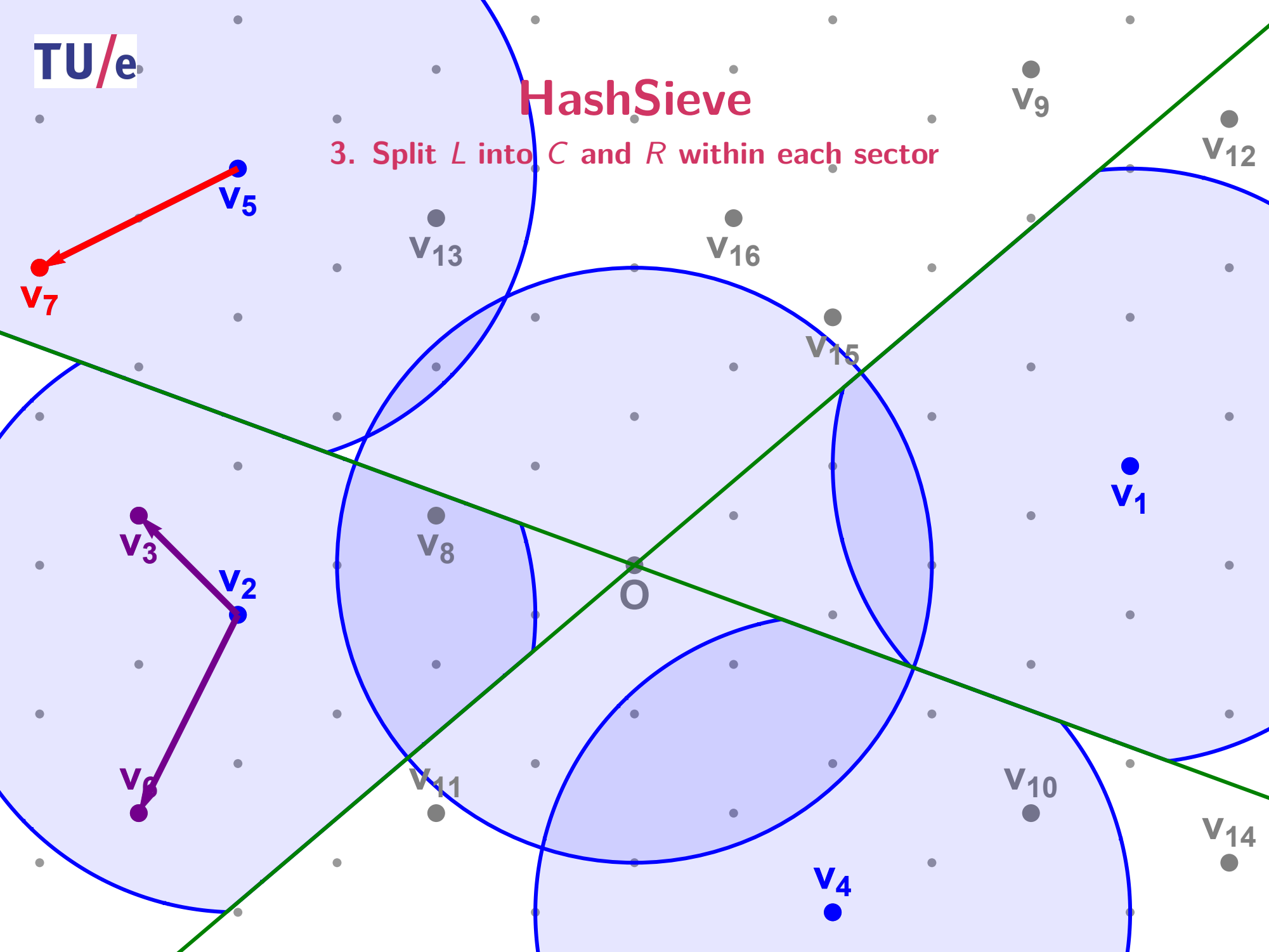
3. Split L into C and R within each sector

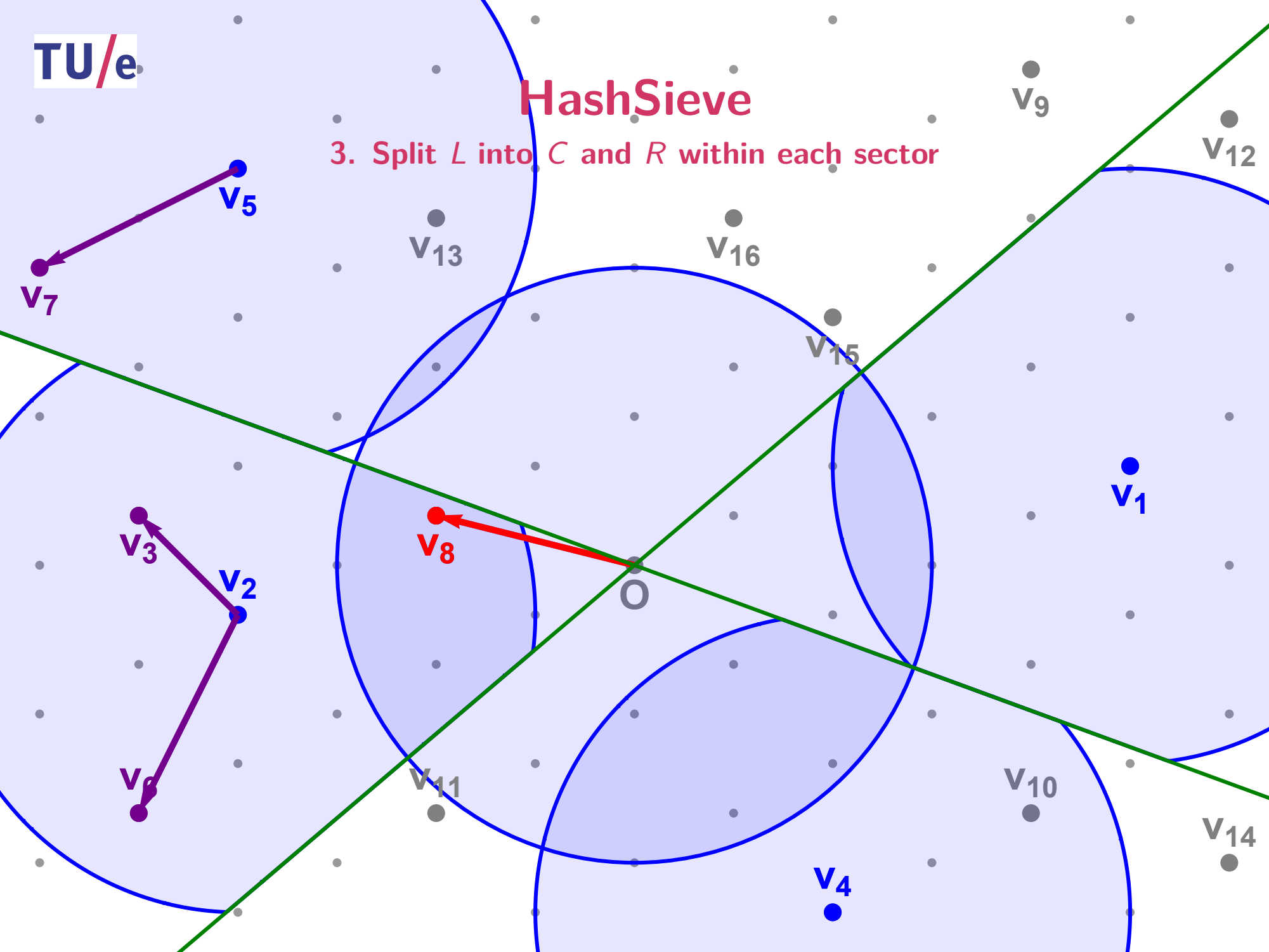


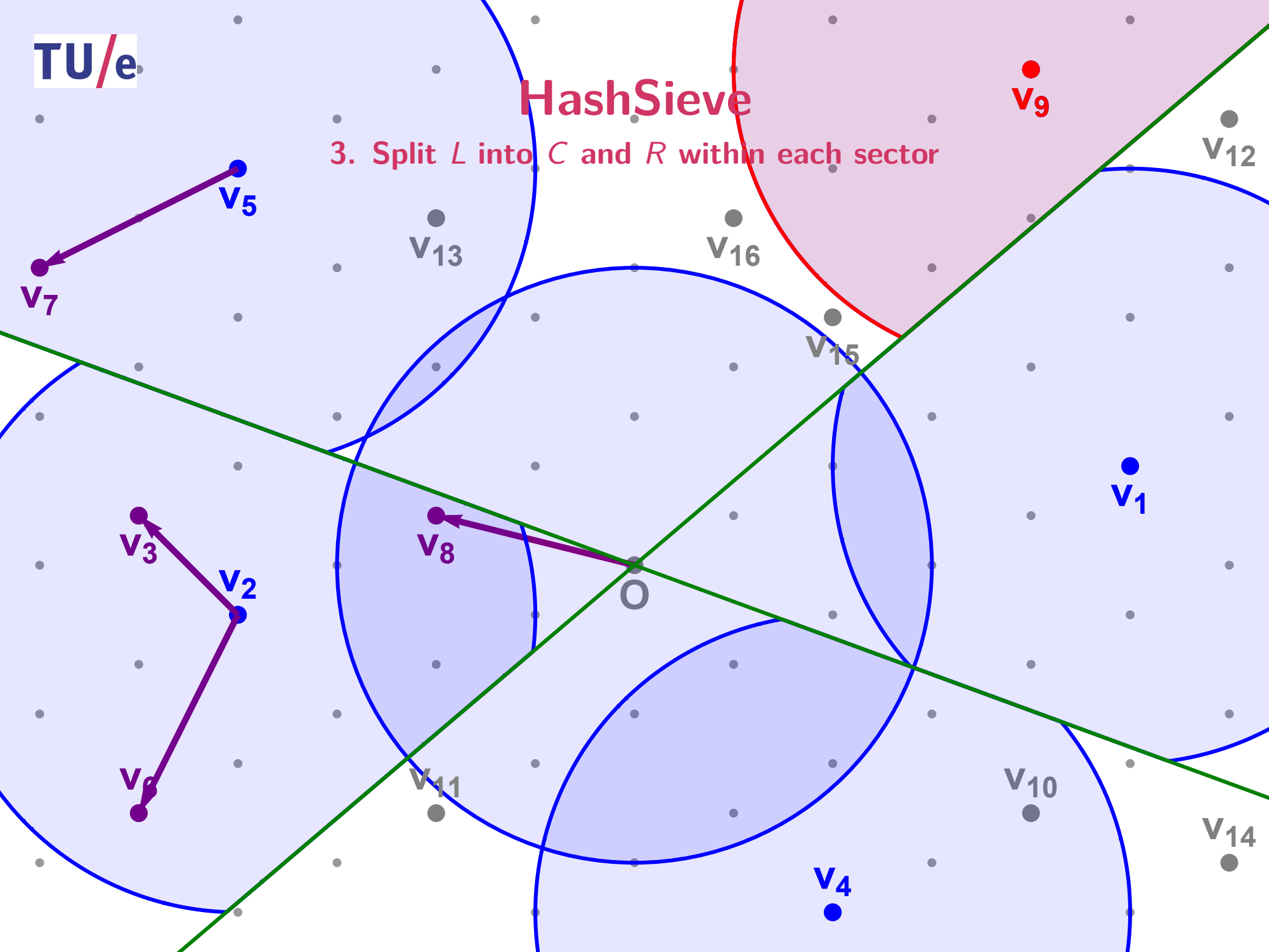
TU/e

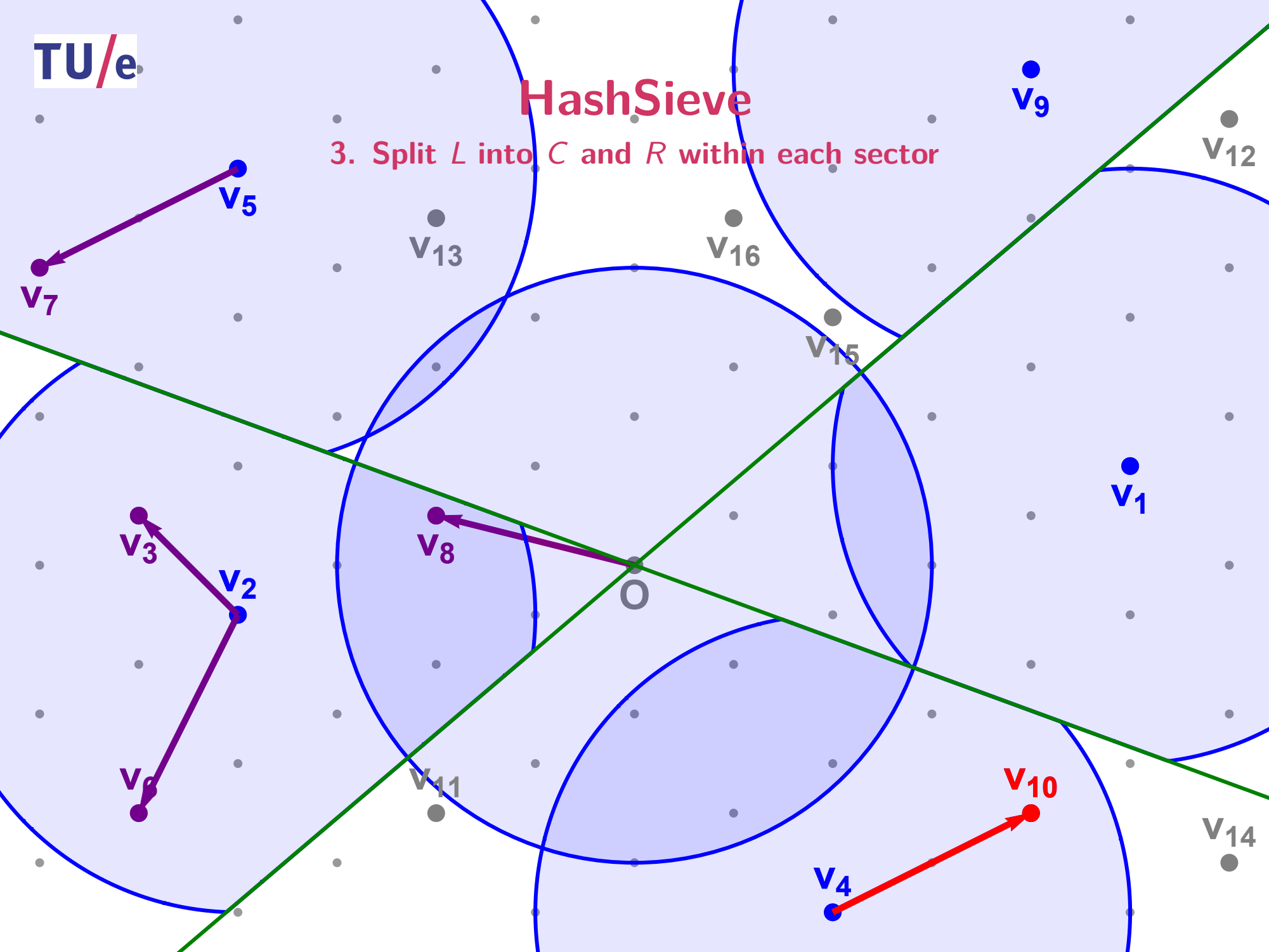
HashSieve

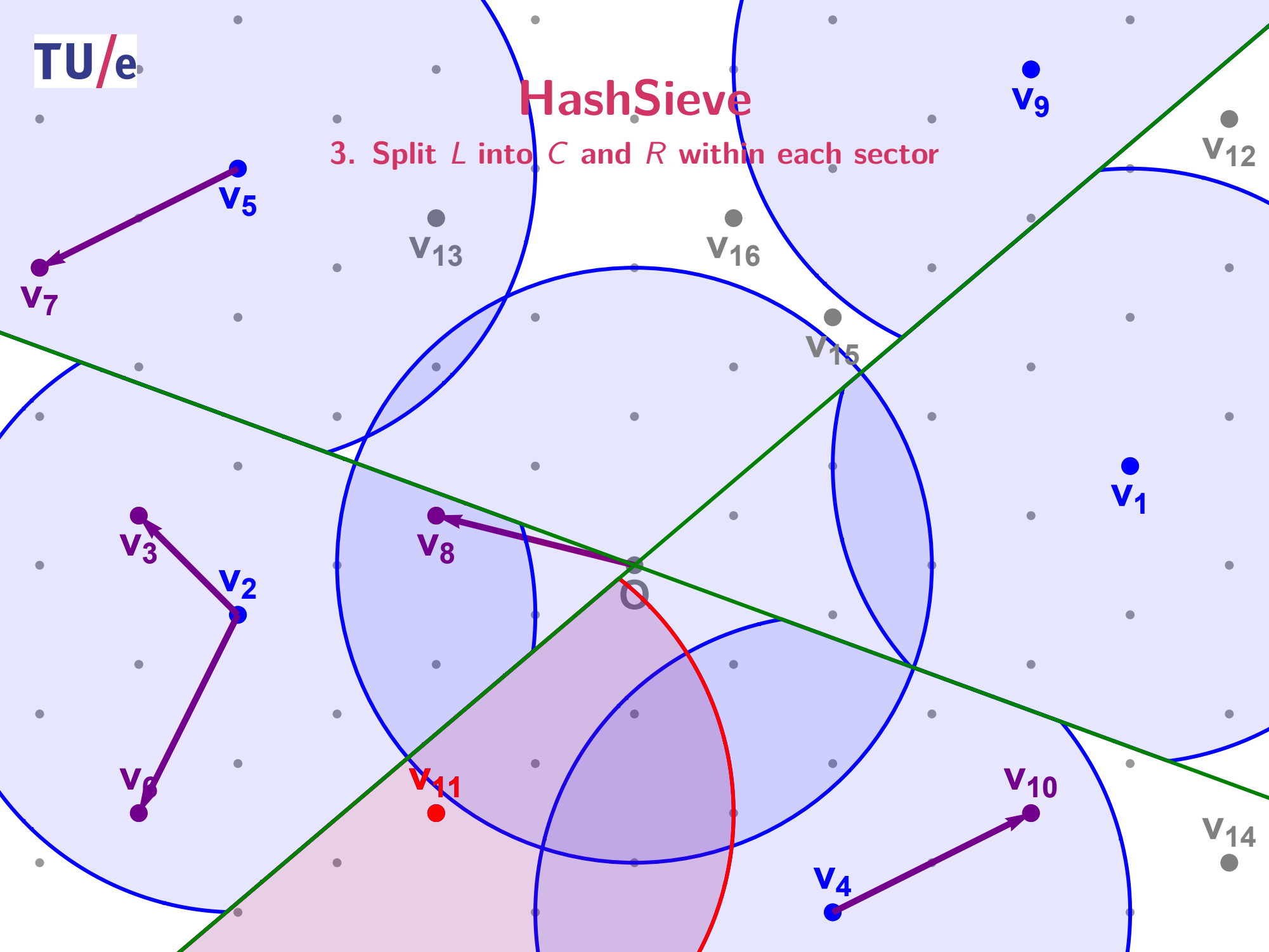
3. Split L into C and R within each sector

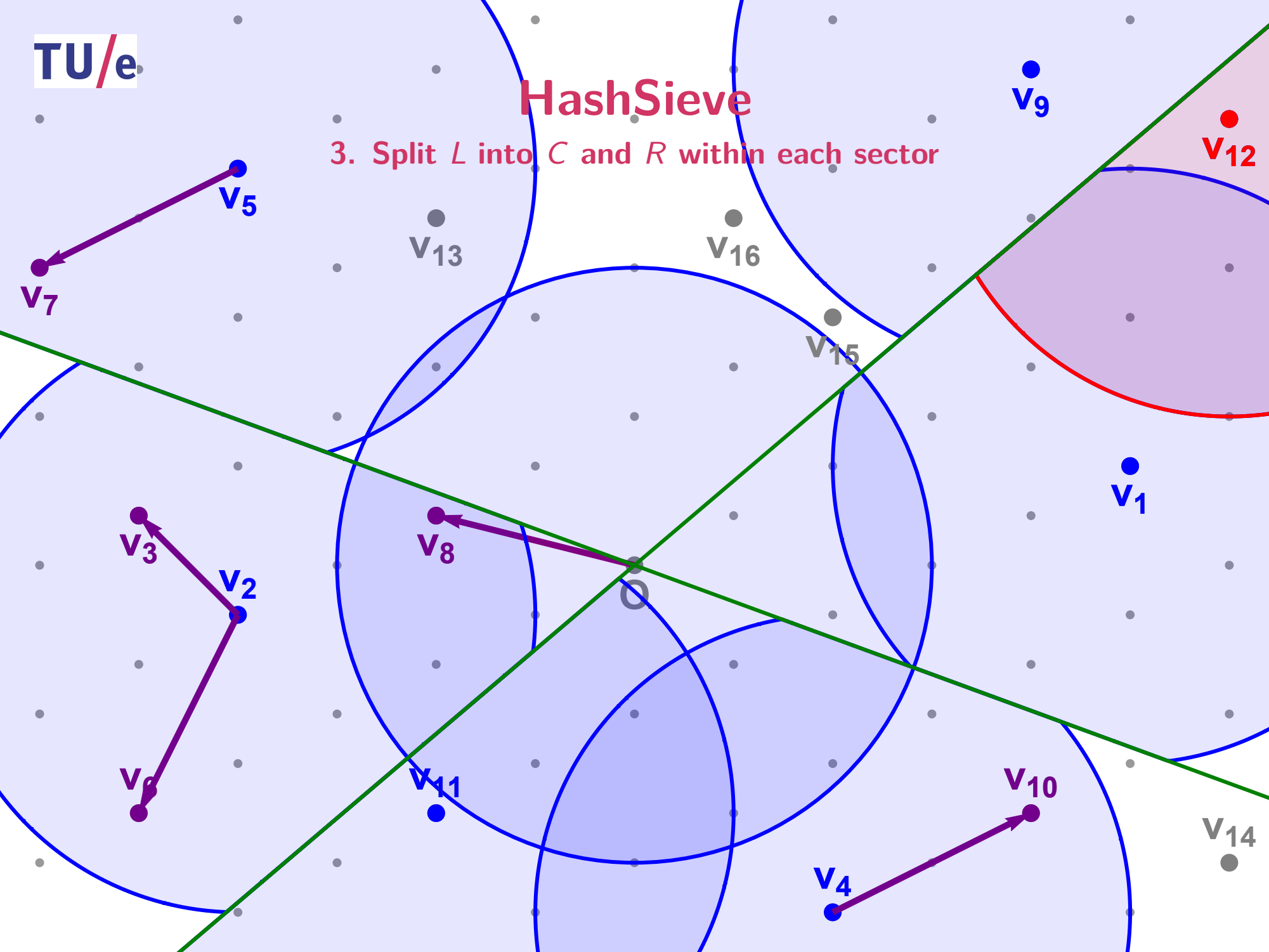


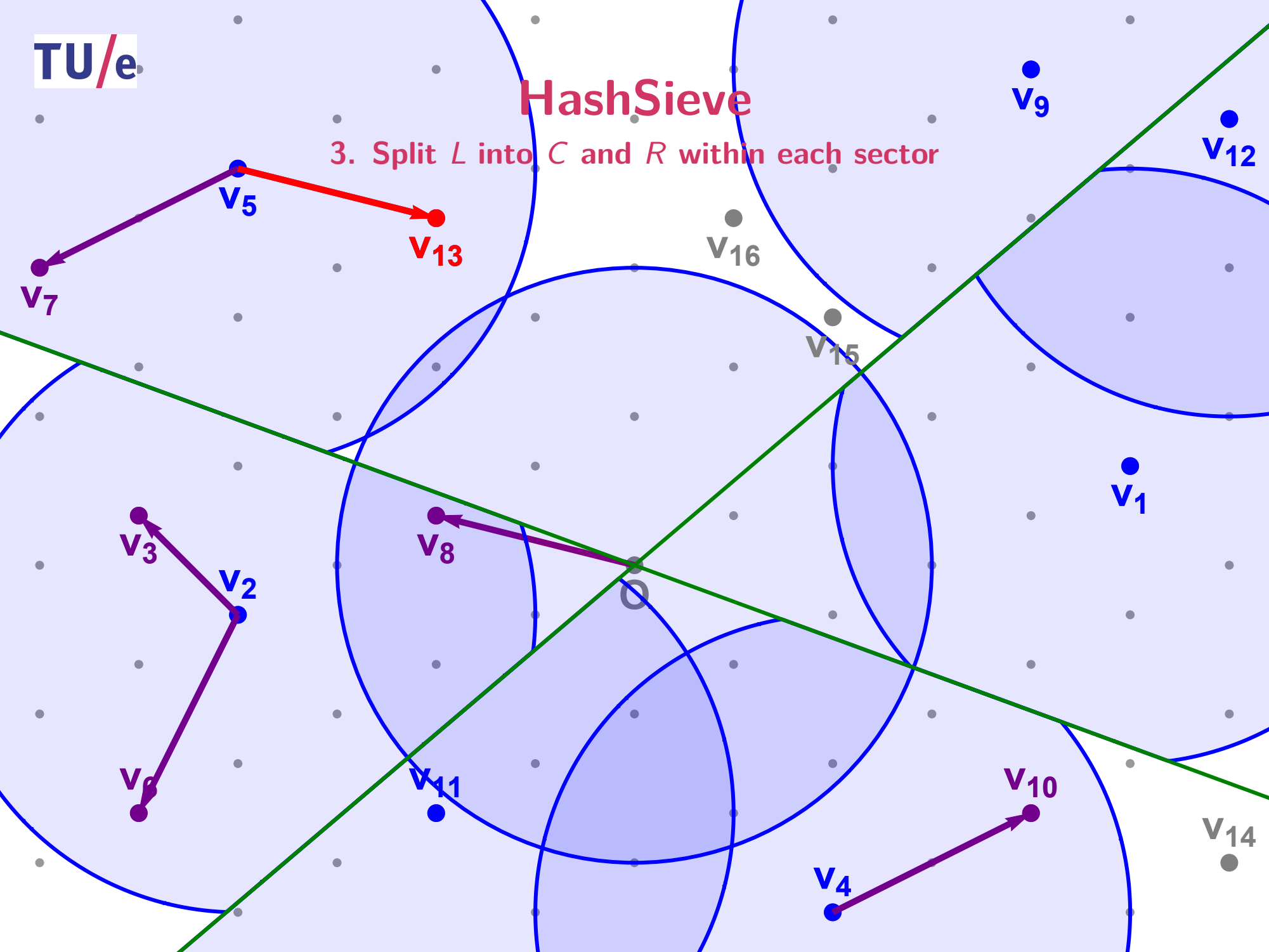

TU/e
HashSieve
3. Split L into C and R within each sector
v_5
v_7
v_{13}
v_{16}
v_9
v_{12}
v_{15}
v_1
v_3
v_2
v_8
O
v_6
v_{11}
v_{10}
v_{14}
v_4

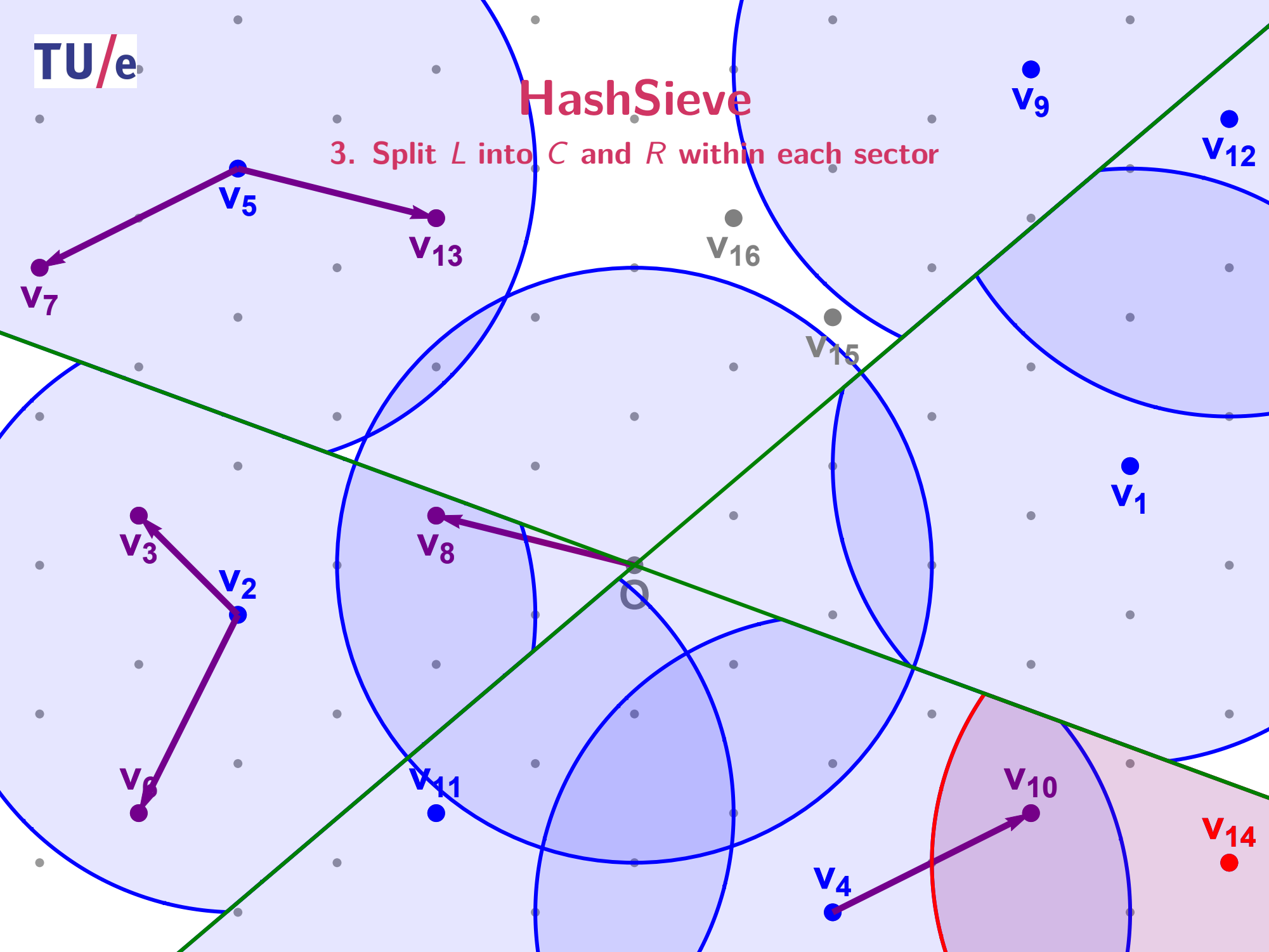

TU/e
HashSieve
3. Split L into C and R within each sector
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4

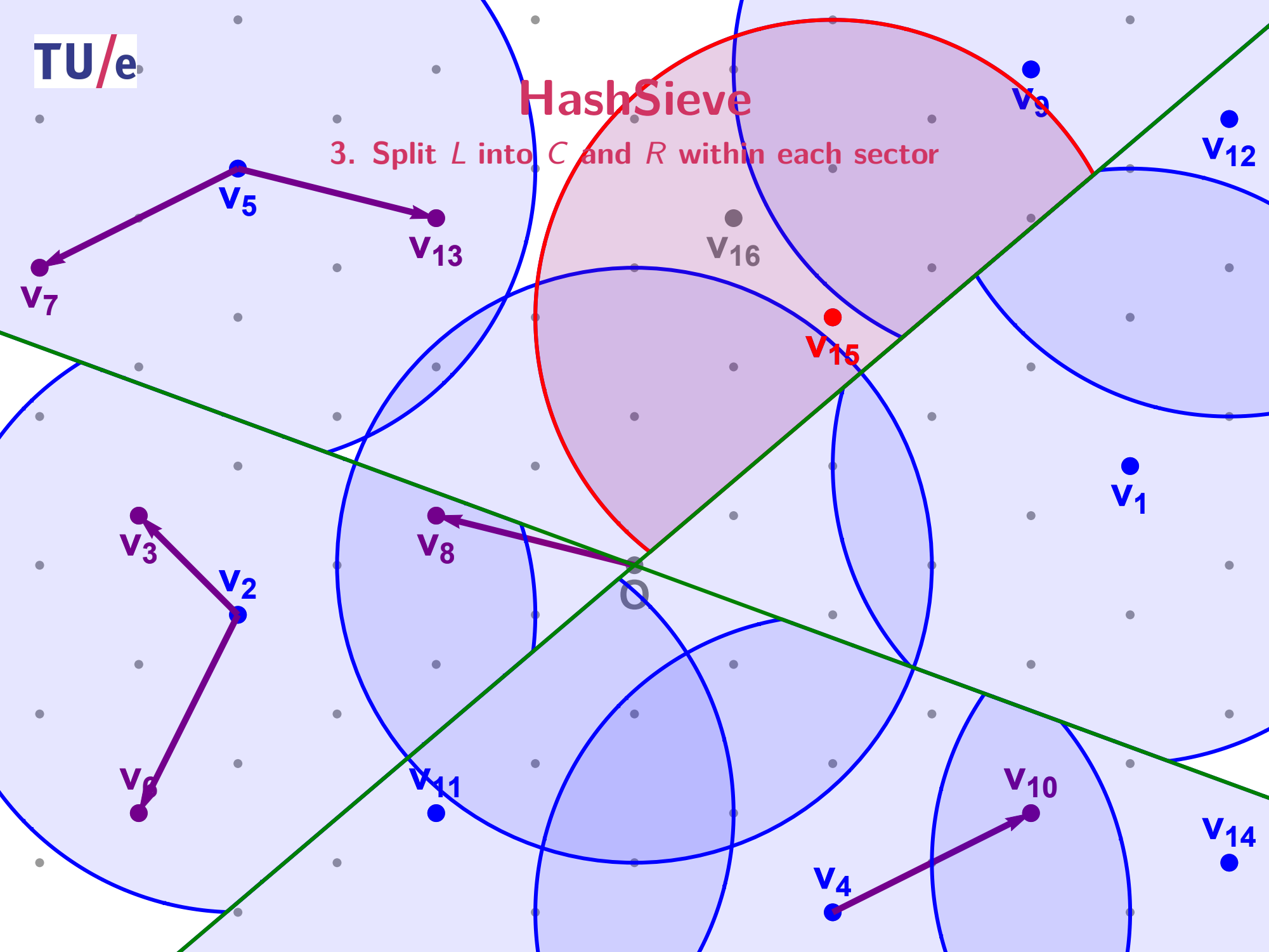

TU/e
HashSieve
3. Split L into C and R within each sector
v_9
v_{12}
v_5
v_{13}
v_{16}
v_7
v_{15}
v_1
v_3
v_8
v_2
O
v_6
v_{11}
v_{10}
v_{14}
v_4


TU/e
HashSieve
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
HashSieve
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
HashSieve
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
v_{11}
v_{10}
v_{14}
v_4

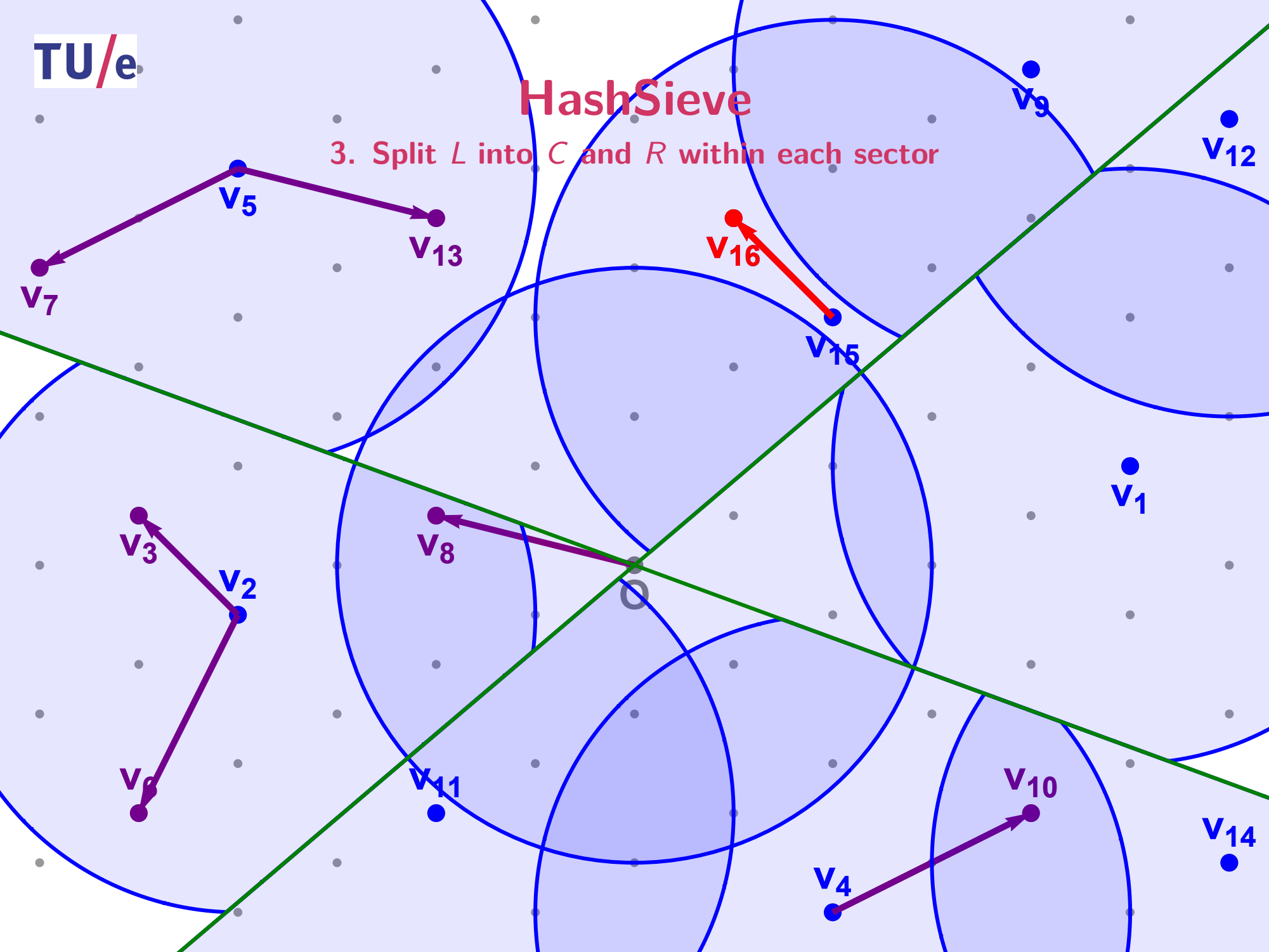

TU/e
HashSieve
3. Split L into C and R within each sector
v_5
v_{13}
v_7
v_{16}
v_{15}
v_9
v_{12}
v_1
v_3
v_2
v_8
v_6
v_{11}
v_{10}
v_{14}
v_4

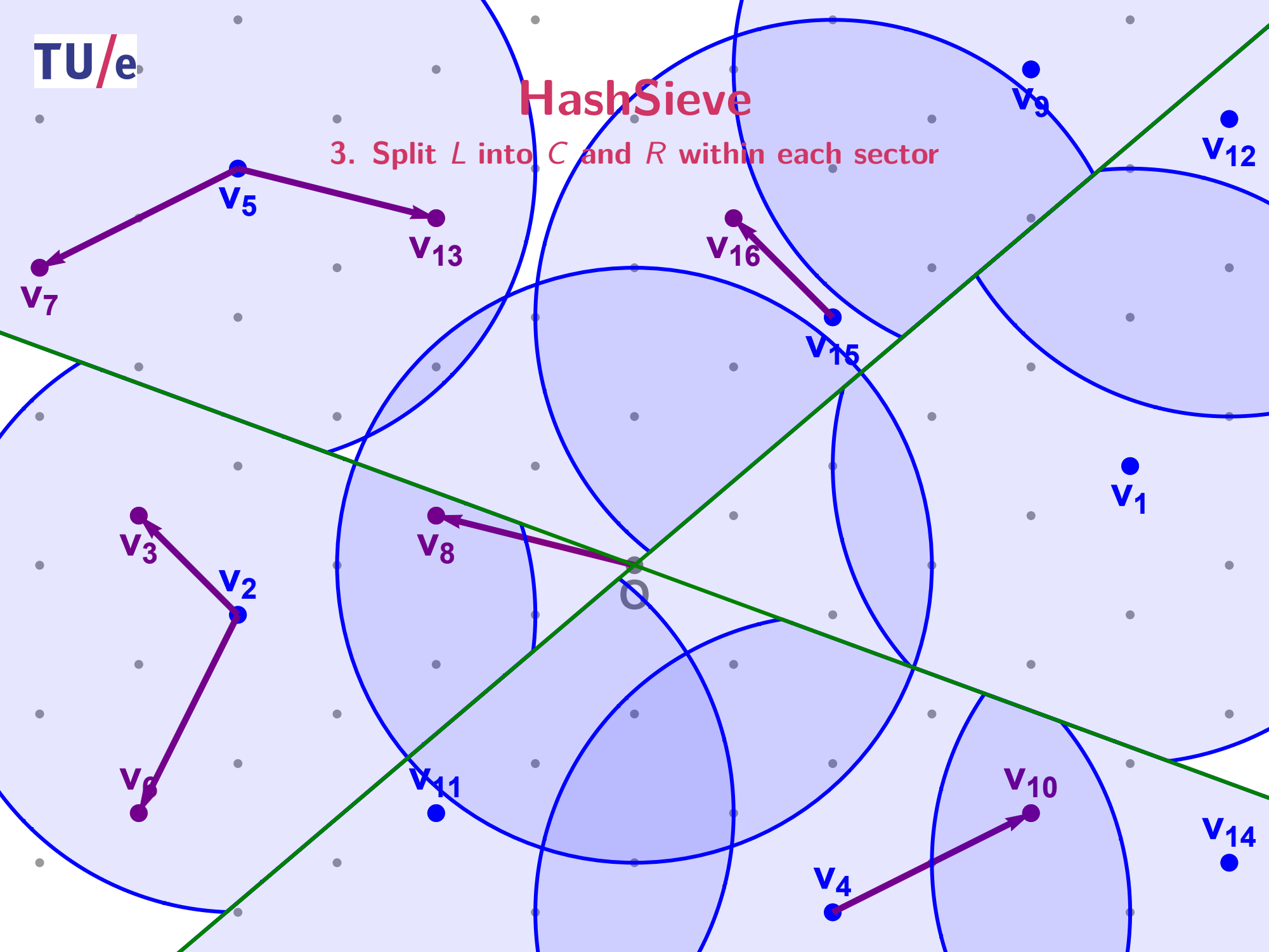

TU/e
HashSieve
3. Split L into C and R within each sector
$\mathbf{v}_5$
$\mathbf{v}_{13}$
$\mathbf{v}_7$
$\mathbf{v}_9$
$\mathbf{v}_{12}$
$\mathbf{v}_{16}$
$\mathbf{v}_{15}$
$\mathbf{v}_1$
$\mathbf{v}_3$
$\mathbf{v}_2$
$\mathbf{v}_8$
$\mathbf{v}_6$
$\mathbf{v}_{11}$
$\mathbf{v}_{10}$
$\mathbf{v}_{14}$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

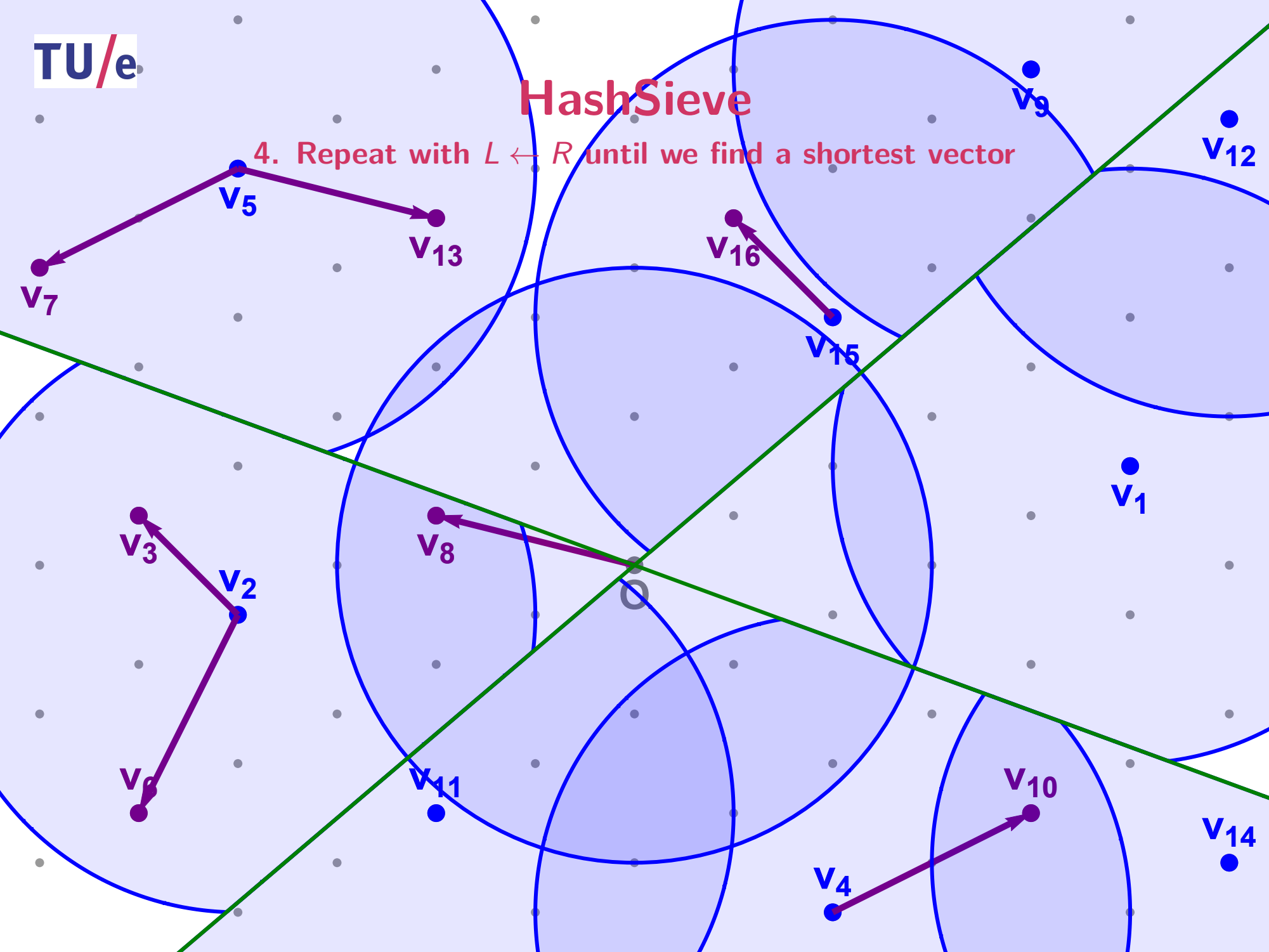
TU/e

HashSieve

3. Split L into C and R within each sector



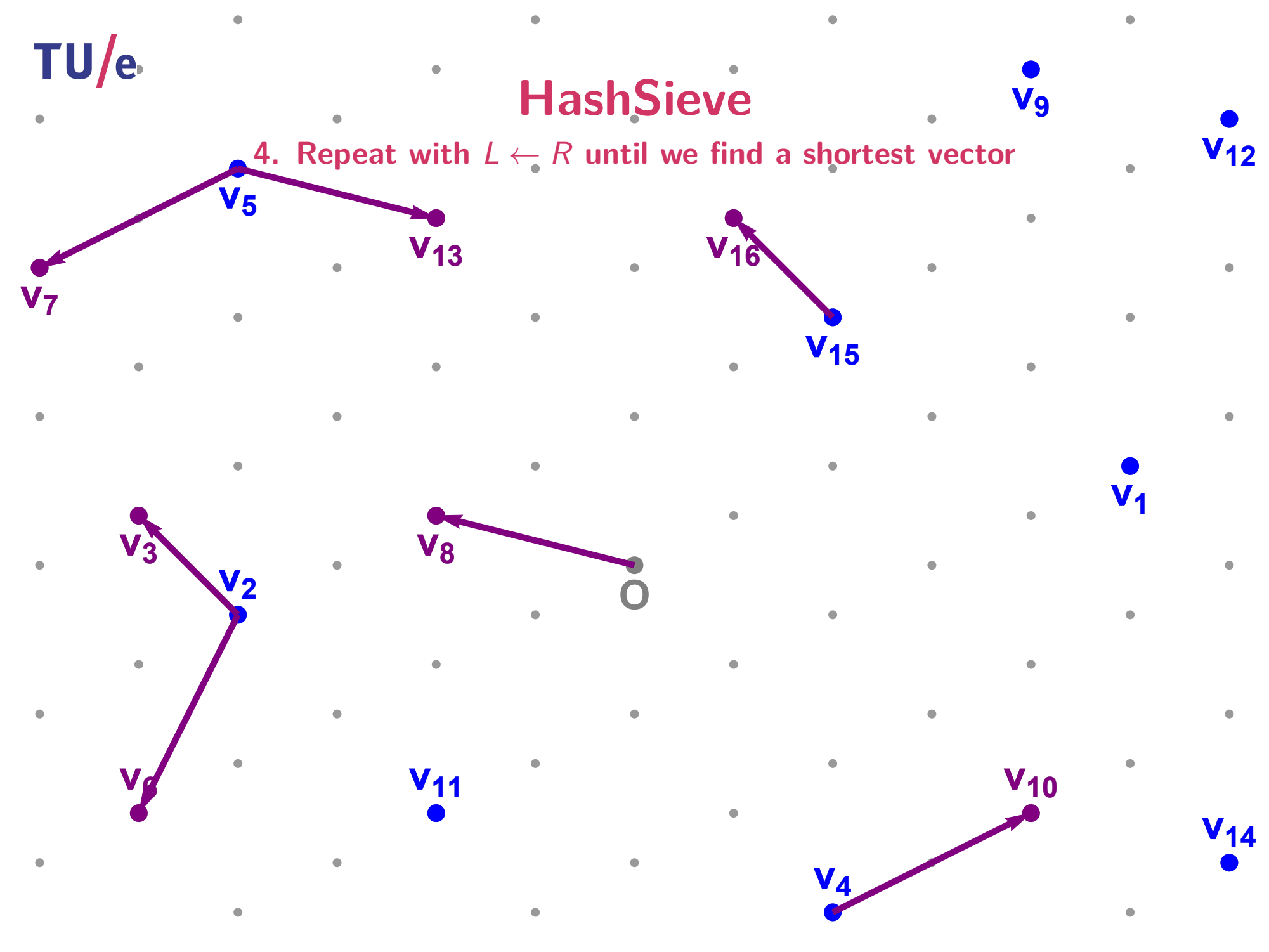

TU/e
HashSieve
3. Split L into C and R within each sector
v_5
v_{13}
v_7
v_{16}
v_{15}
v_9
v_{12}
v_1
v_3
v_2
v_8
v_6
v_{11}
v_{10}
v_4
v_{14}
0


TU/e
HashSieve
4. Repeat with $L \leftarrow R$ until we find a shortest vector
v_5
v_{13}
v_7
v_{16}
v_{15}
v_9
v_{12}
v_1
v_3
v_2
v_8
0
v_6
v_{11}
v_{10}
v_{14}
v_4

TU/e

HashSieve

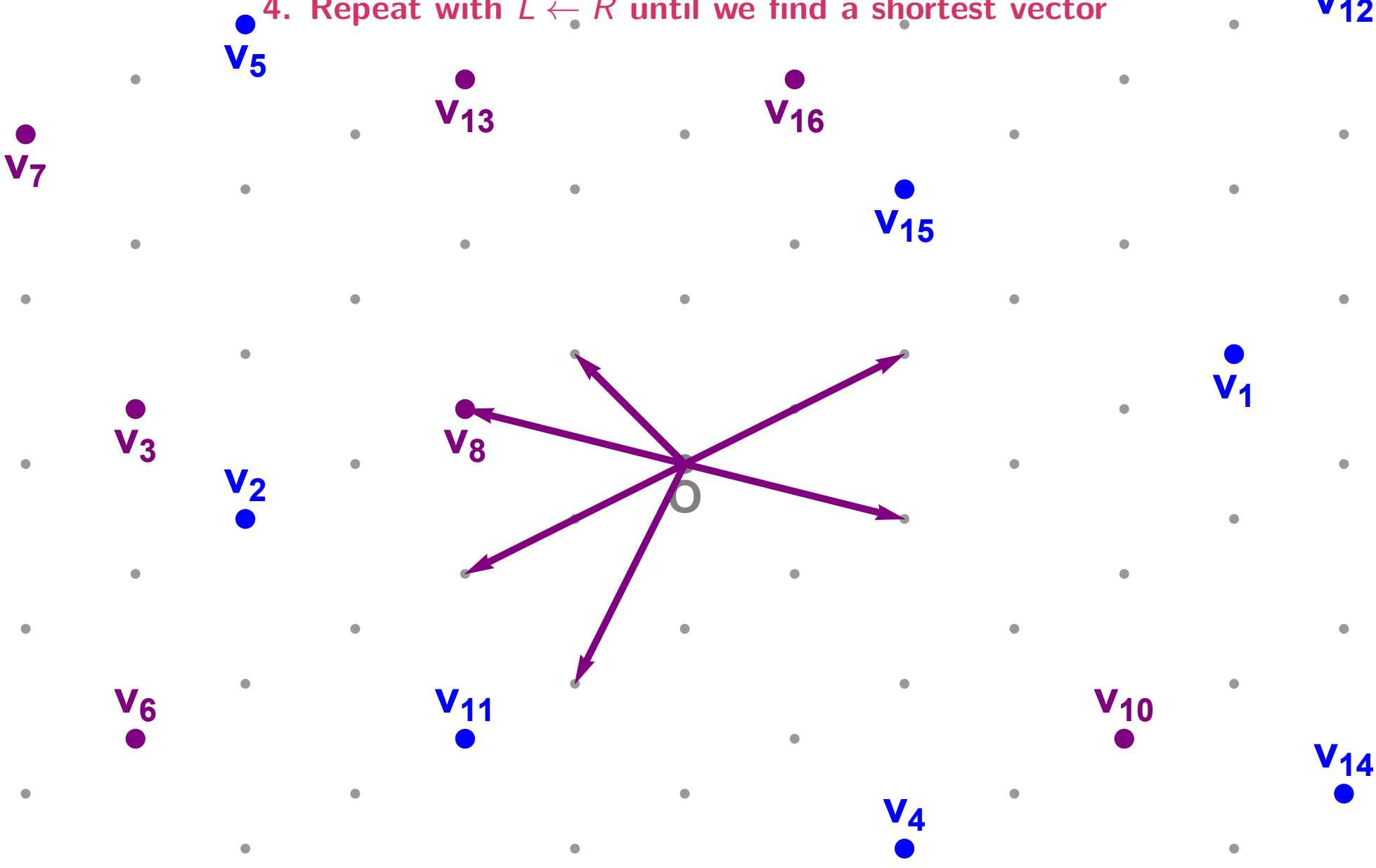
4. Repeat with $L \leftarrow R$ until we find a shortest vector



TU/e

HashSieve

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



TU/e

HashSieve

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

TU/e

HashSieve

Overview

TU/e

HashSieve

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

HashSieve

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

HashSieve

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

HashSieve

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

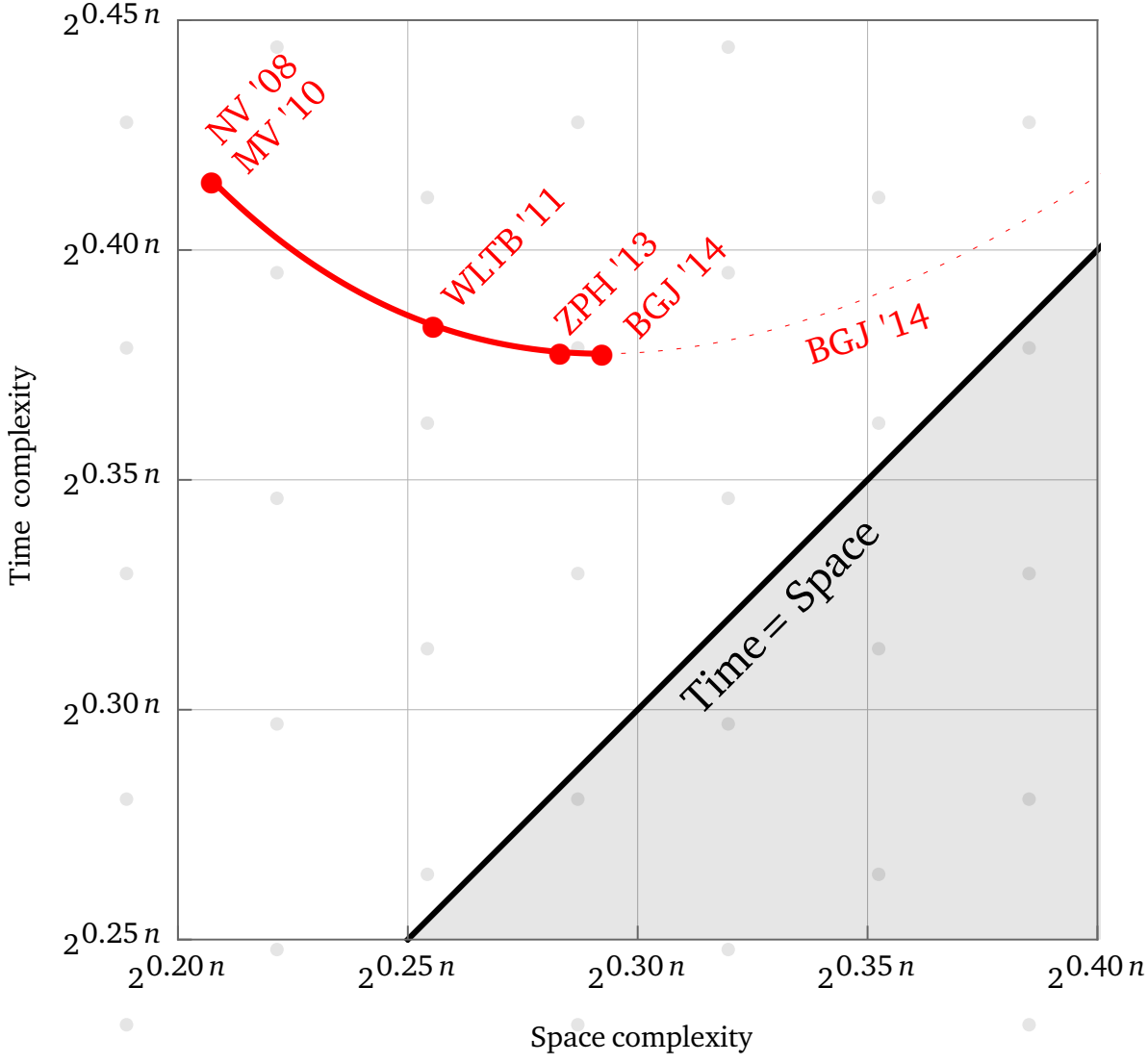
Heuristic

The HashSieve runs in time and space $2^{0.34n+o(n)}$.

TU/e

HashSieve

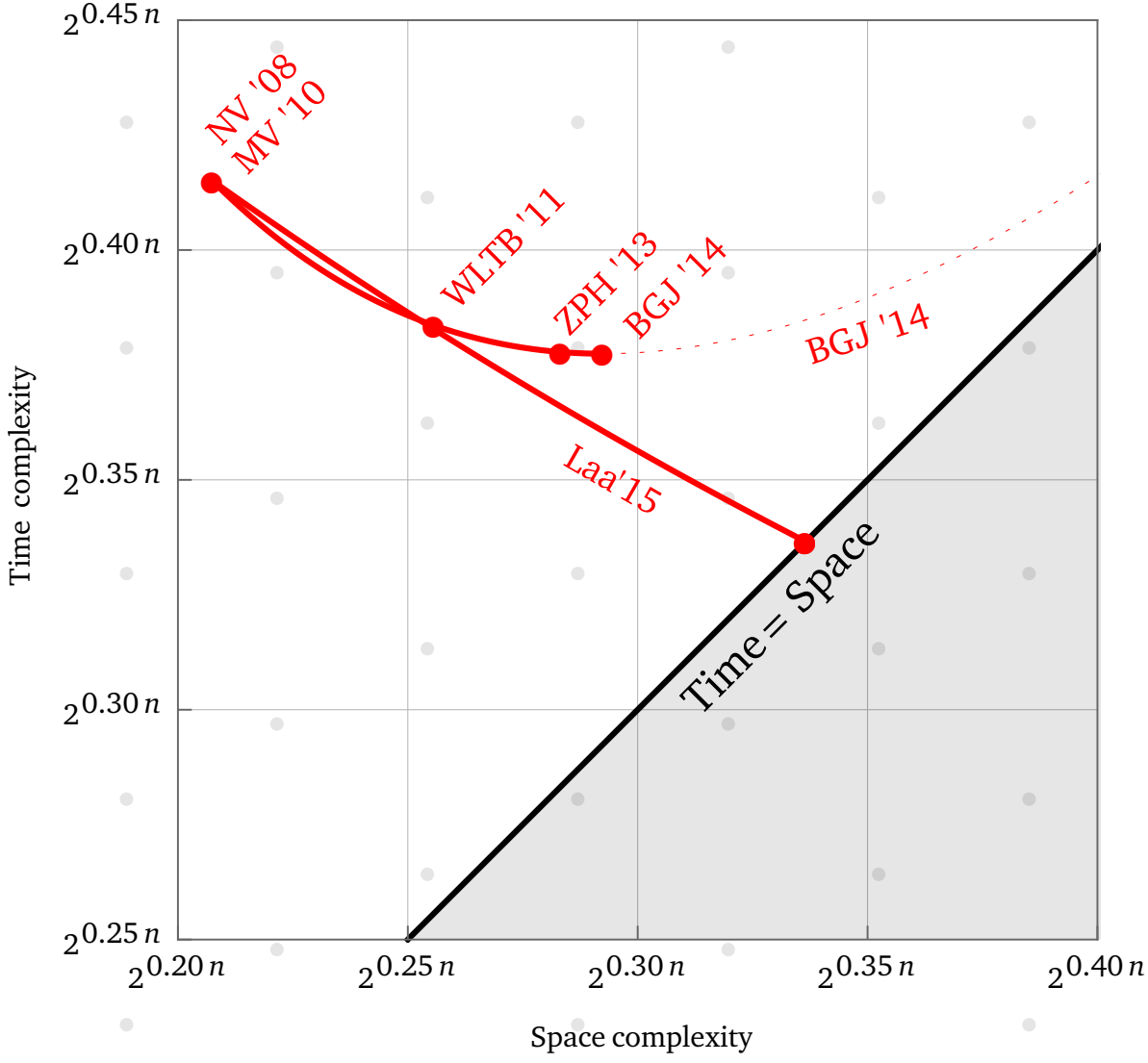
Space/time trade-off



TU/e

HashSieve

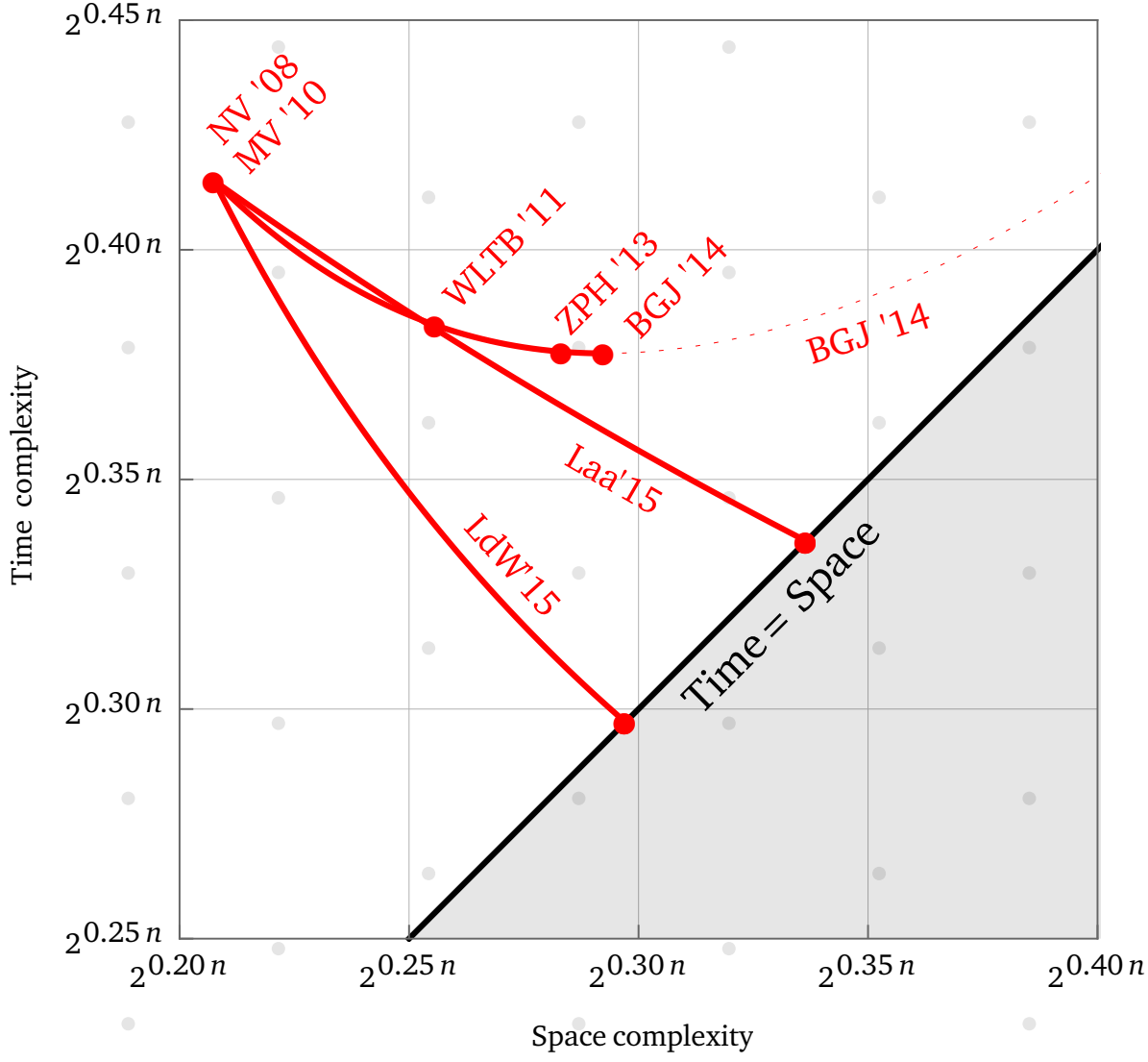
Space/time trade-off



TU/e

SphereSieve

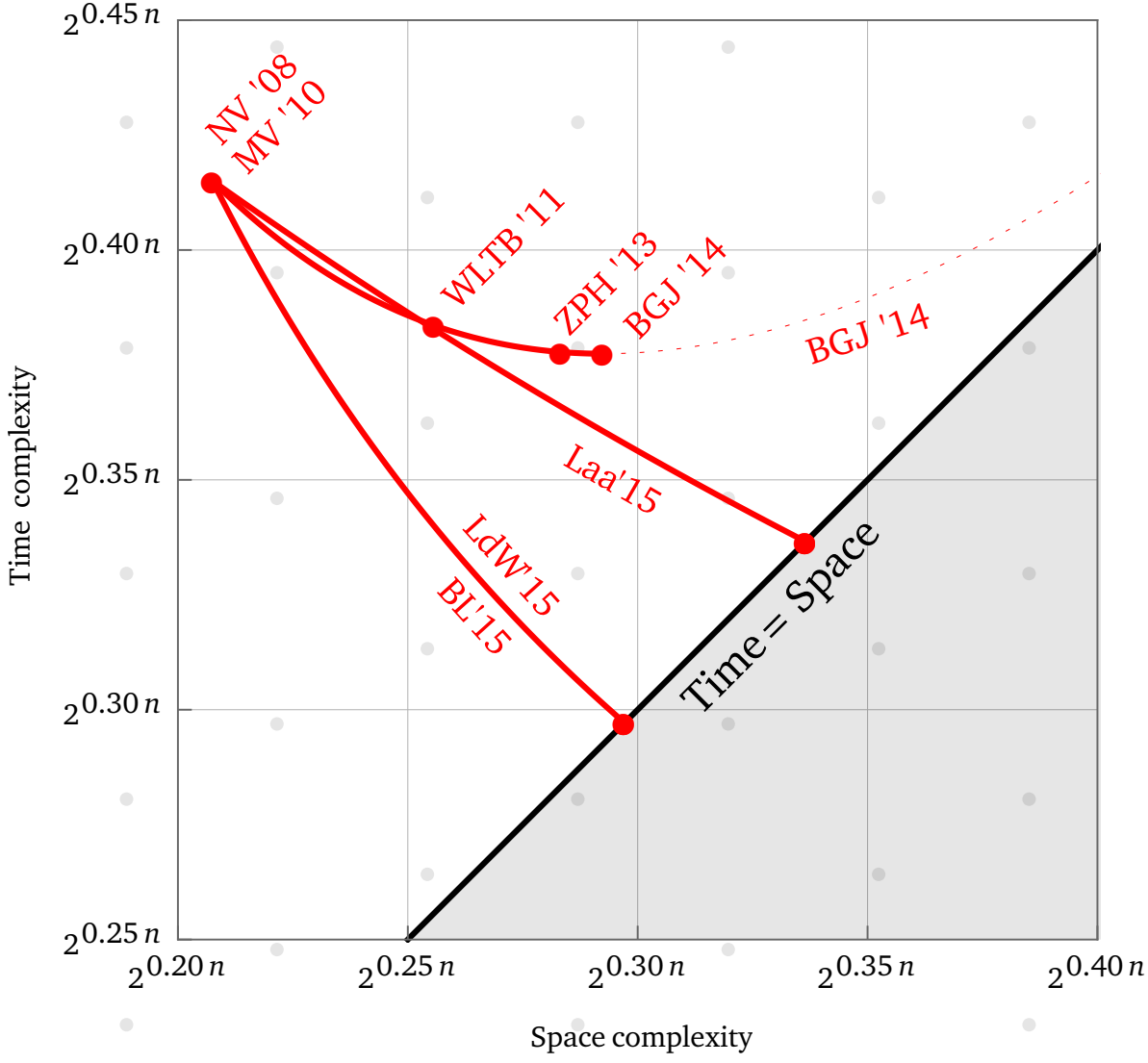
Space/time trade-off



TU/e

CrossPolytopeSieve

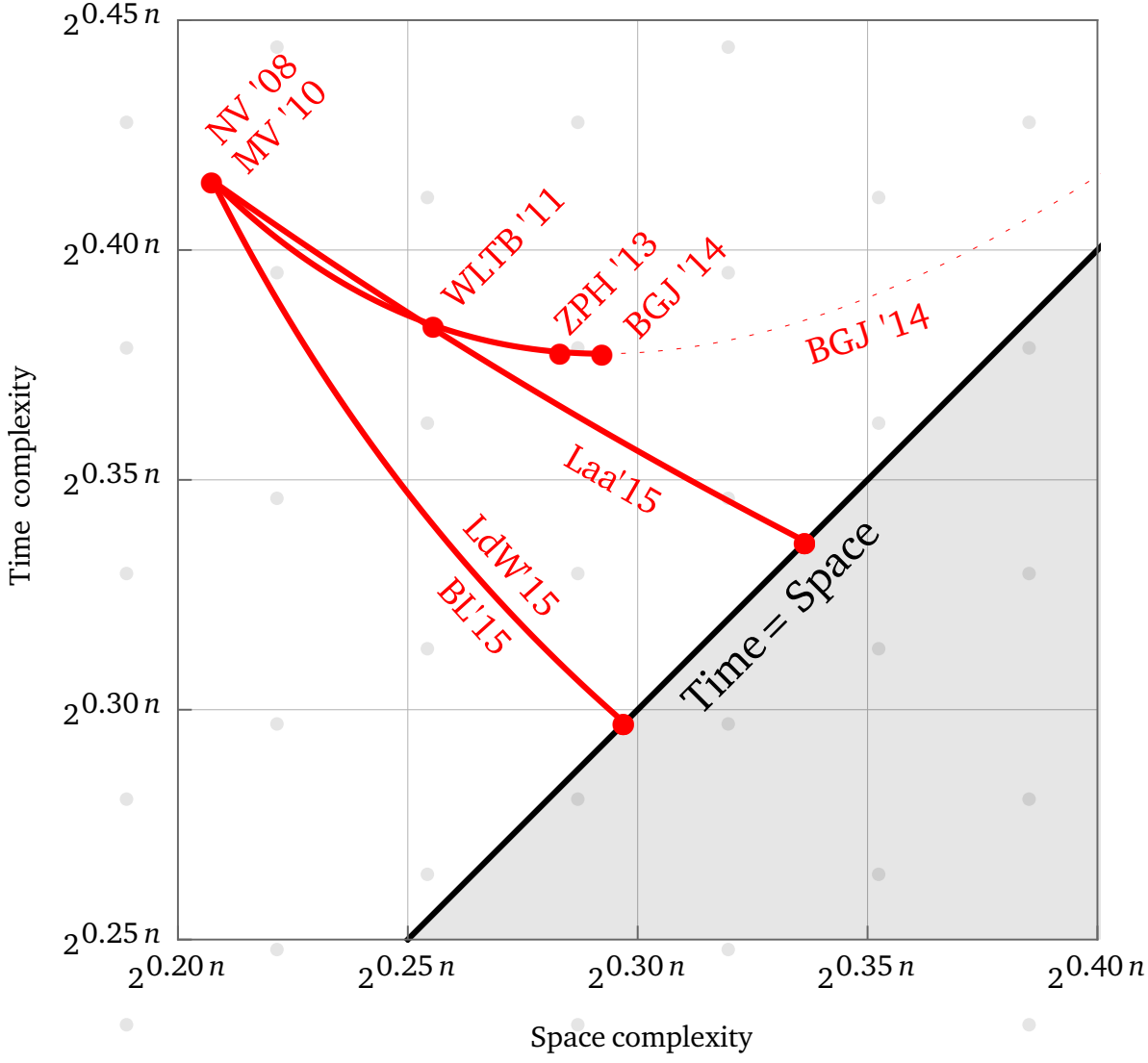
Space/time trade-off



TU/e

“Post-quantum cryptography”

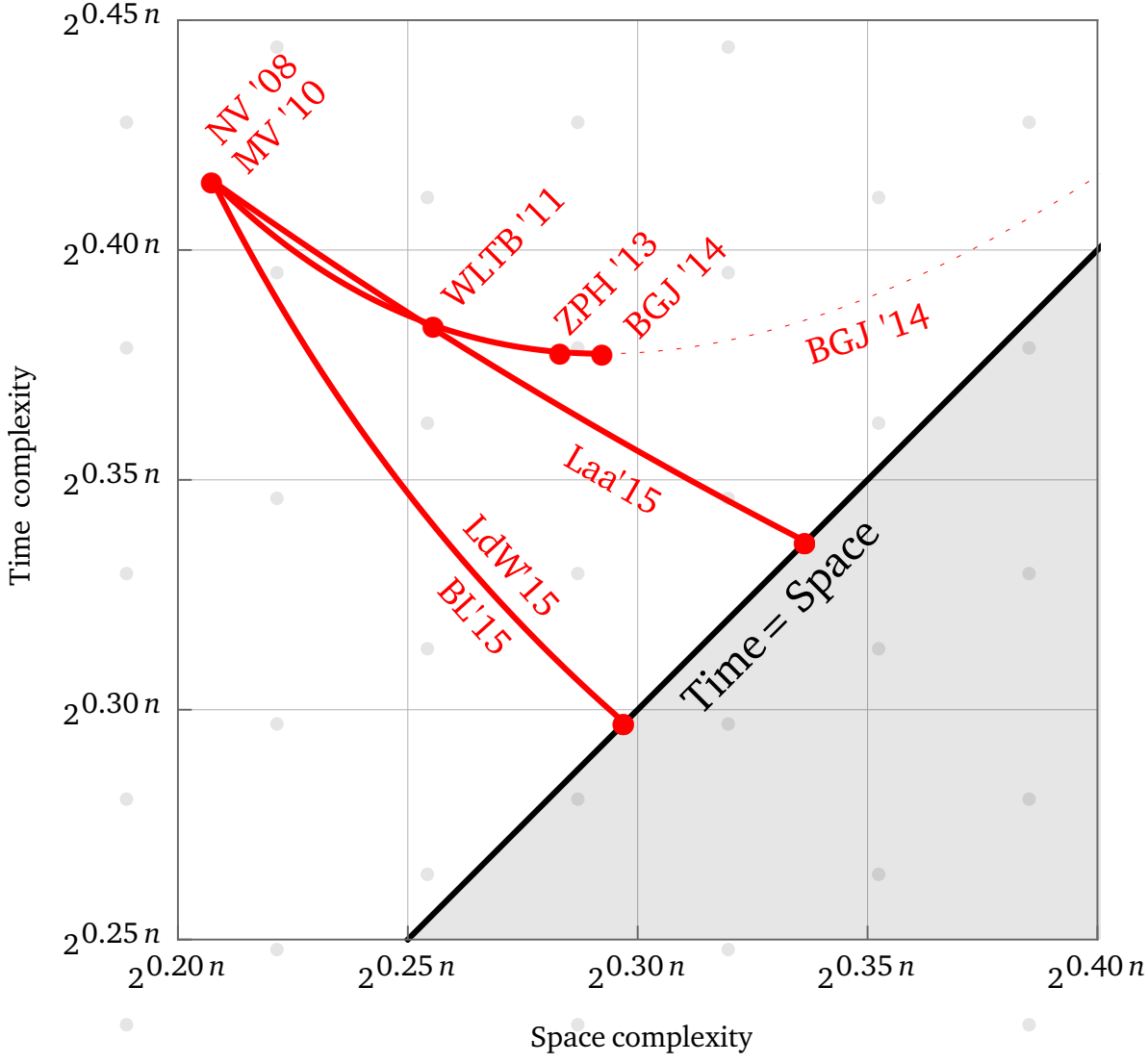
Space/time trade-off



TU/e

"Post-quantum cryptography"

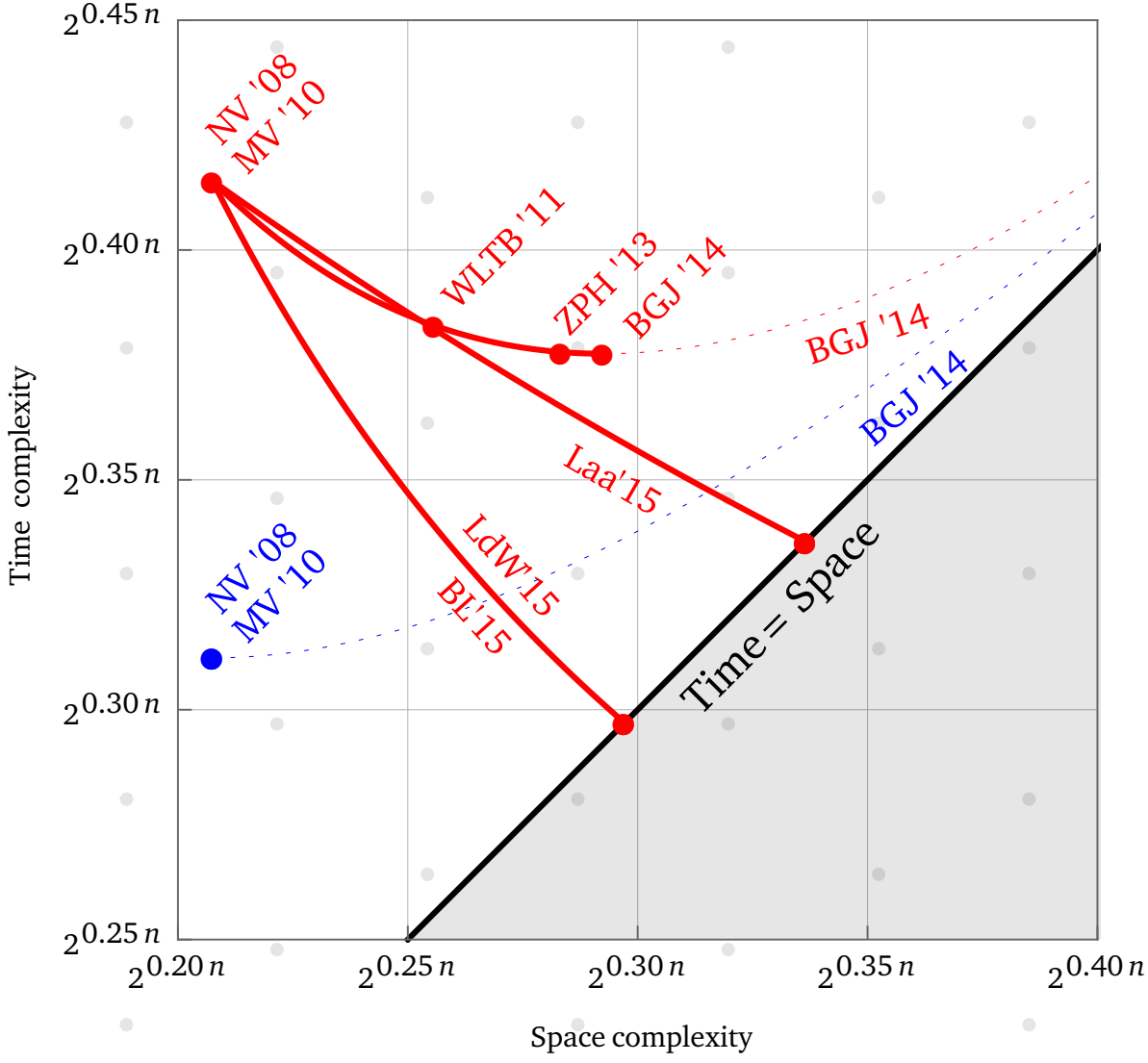
Space/time trade-off (quantum search)



TU/e

"Post-quantum cryptography"

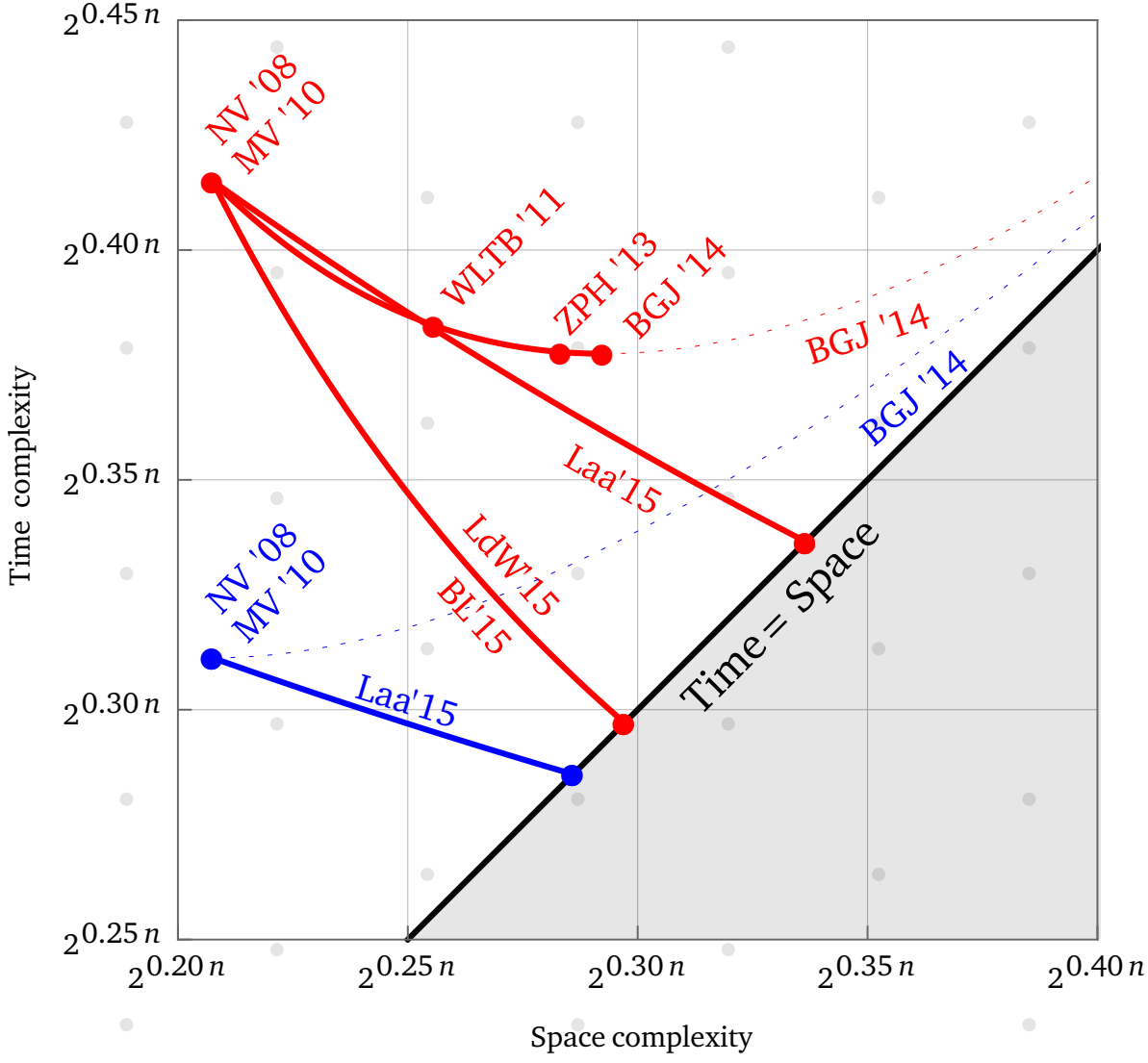
Space/time trade-off (quantum search)



TU/e

“Post-quantum cryptography”

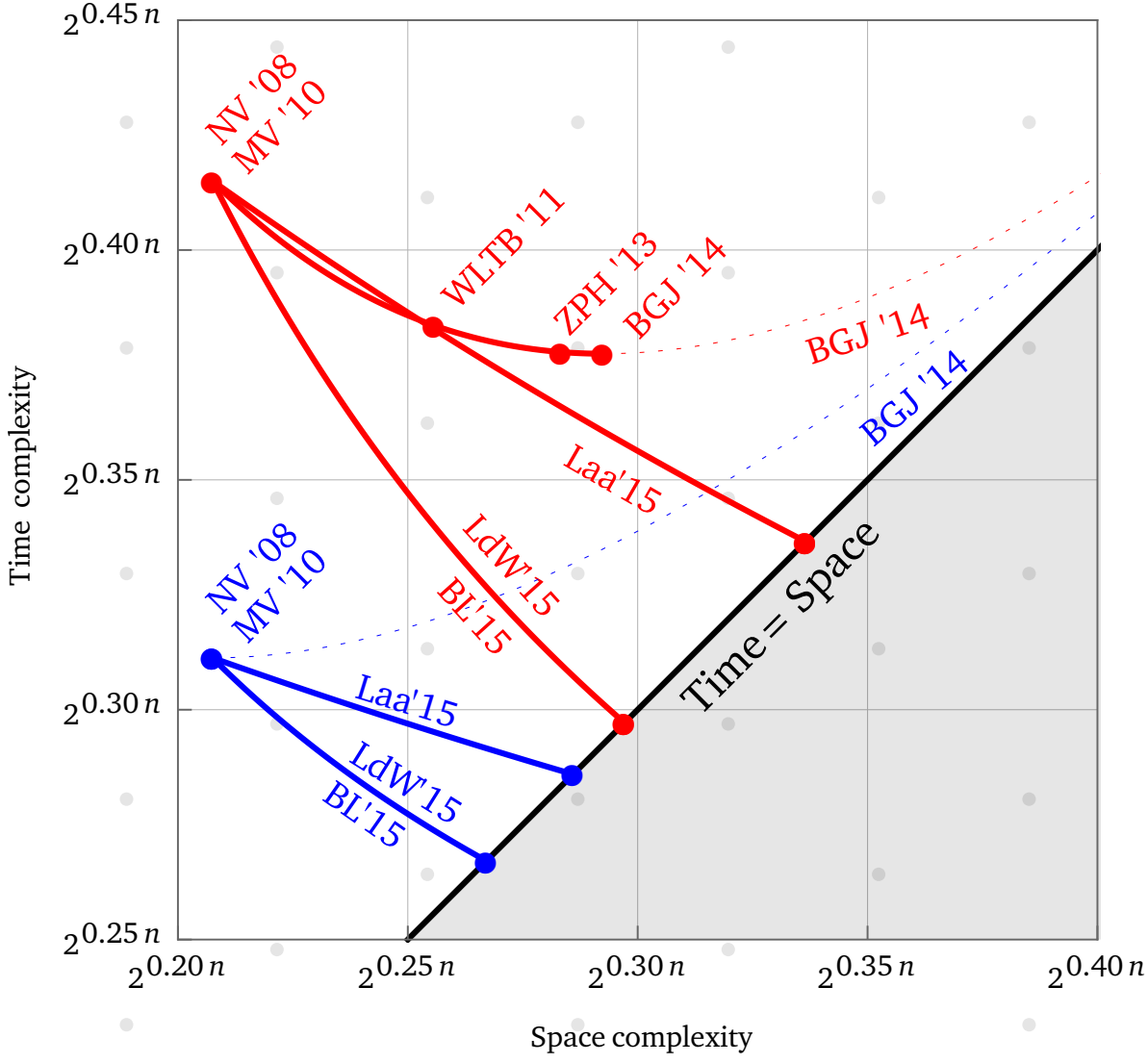
Space/time trade-off (quantum search)



TU/e

"Post-quantum cryptography"

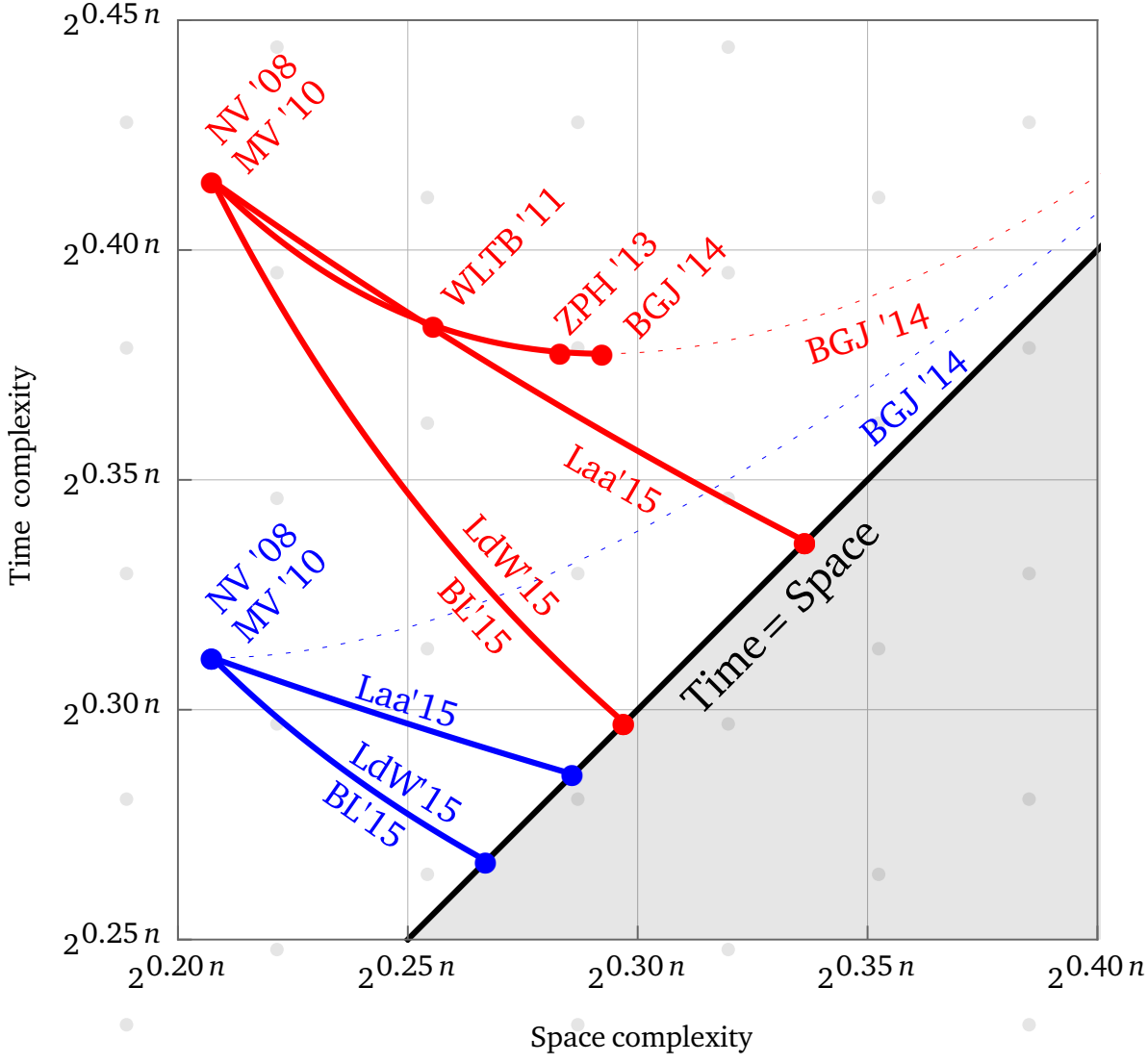
Space/time trade-off (quantum search)



TU/e

"Post-quantum cryptography"

Space/time trade-off (quantum search)



TU/e

Questions