Algorithms for hard lattice problems

Thijs Laarhoven

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

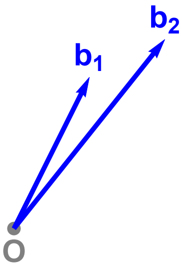
Tenerife PQCrypto Conference
(January 31, 2018)

Lattices

What is a lattice?

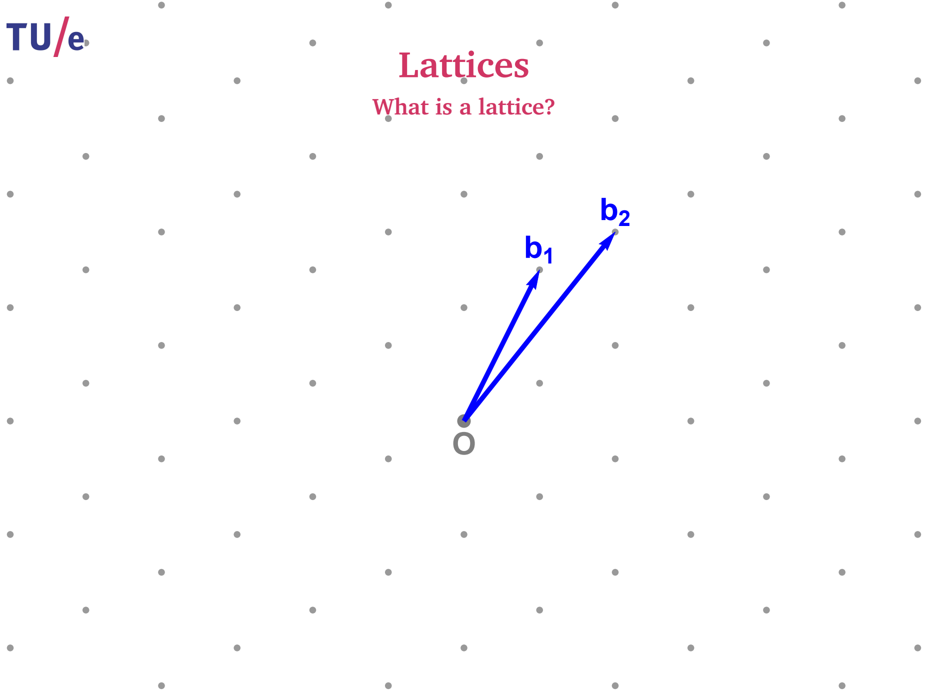
Lattices

What is a lattice?



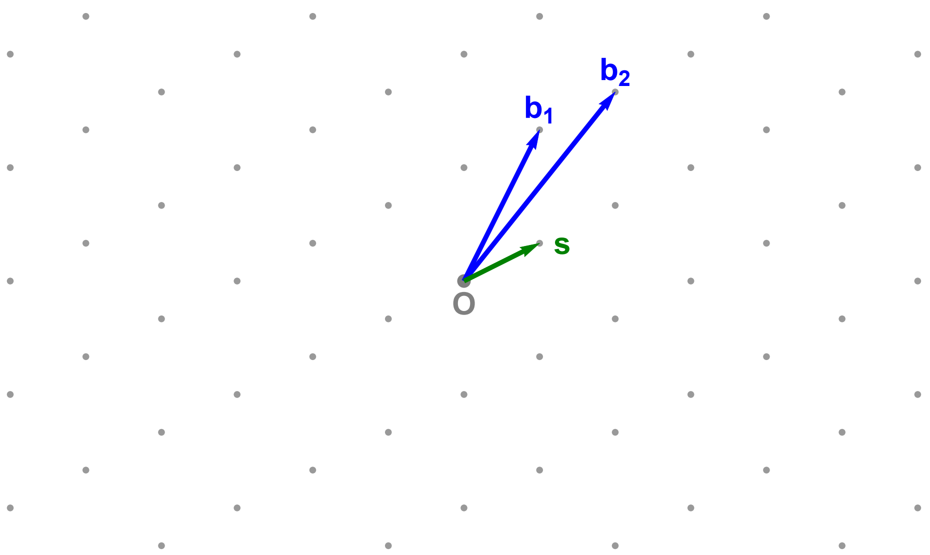
Lattices

What is a lattice?



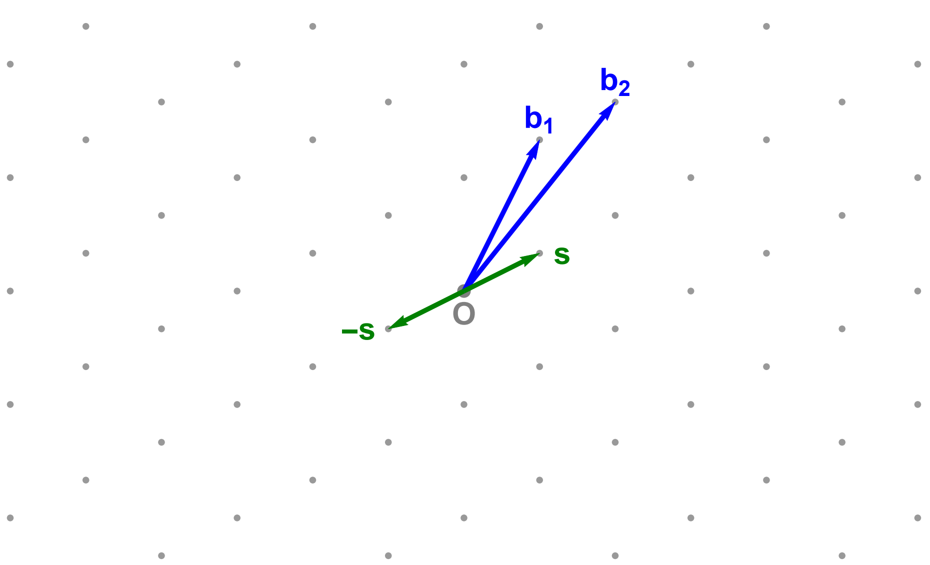
Lattices

Shortest Vector Problem (SVP)



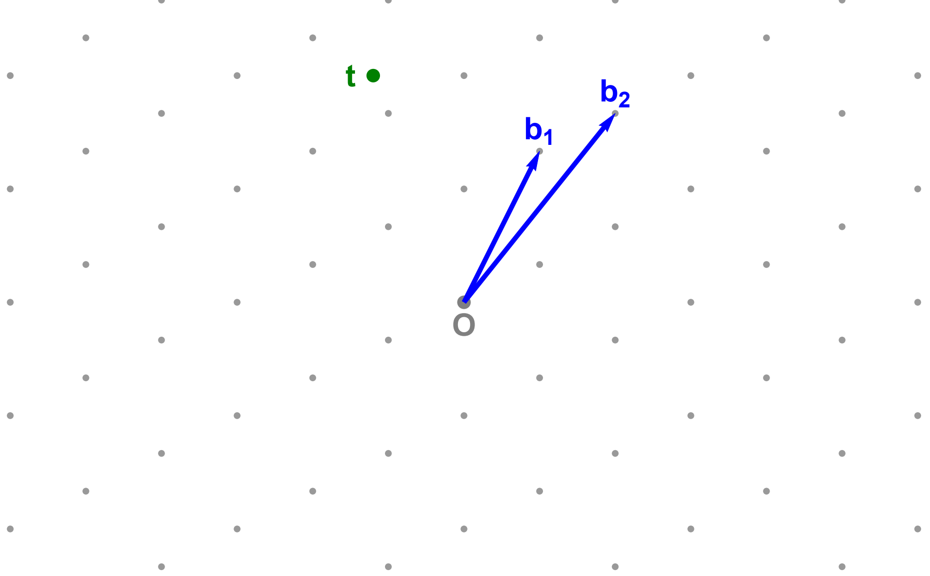
Lattices

Shortest Vector Problem (SVP)



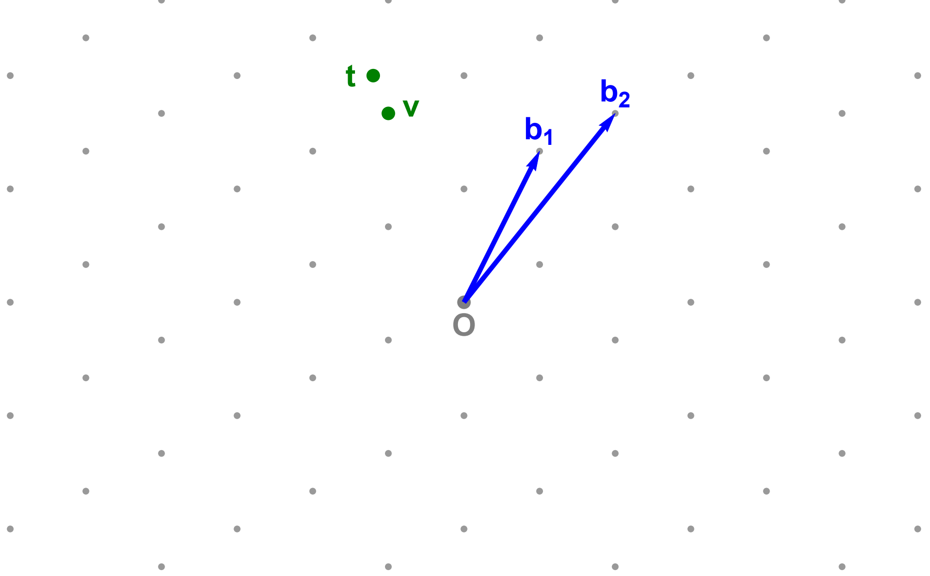
Lattices

Closest Vector Problem (CVP)



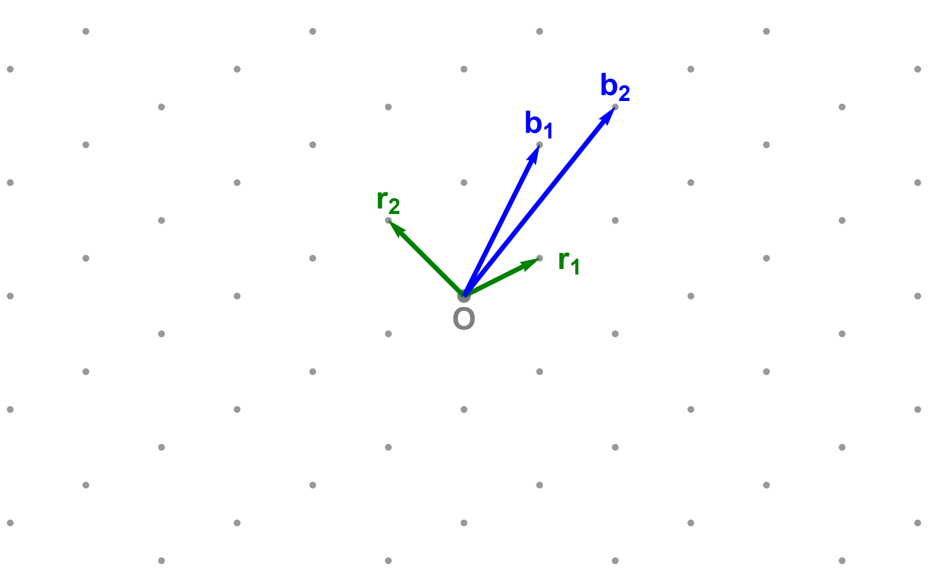
Lattices

Closest Vector Problem (CVP)



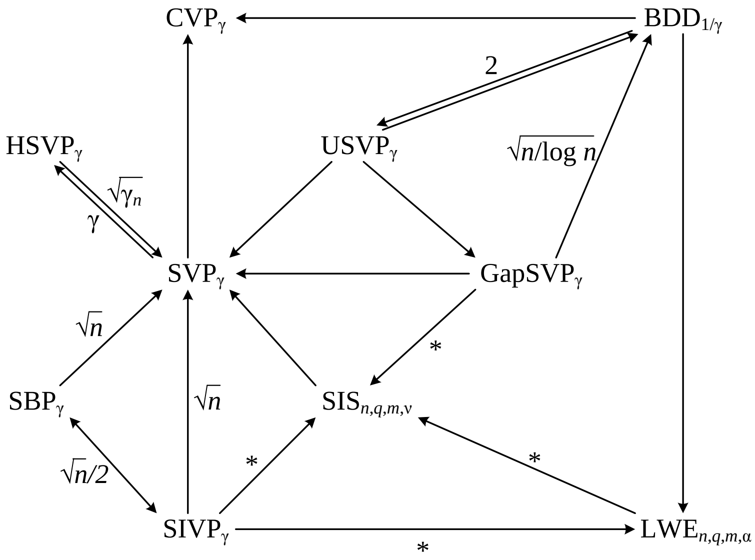
Lattices

Lattice basis reduction



Lattices

Hard lattice problems [LvdPdW12]



Lattices

Lattice-based cryptography

Problem: Security of lattice-based cryptographic primitives

- Lattice-based crypto relies on hardness of lattice problems
- Most lattice problems reducible to (approximate) SVP
- State-of-the-art: BKZ basis reduction [Sch87, SE94, ...]
  - BKZ uses exact SVP algorithm as subroutine
  - Complexity of BKZ dominated by *exact* SVP calls

SVP costs $\Longrightarrow$ BKZ costs $\Longrightarrow$ Security estimates $\Longrightarrow$ Parameters

Problem: How hard is SVP in high dimensions?

Outline

SVP algorithms

- Enumeration
- Sieving

SVP hardness

- Theory
- Practice
- NIST submissions

Conclusion

Outline

SVP algorithms

- Enumeration
- Sieving

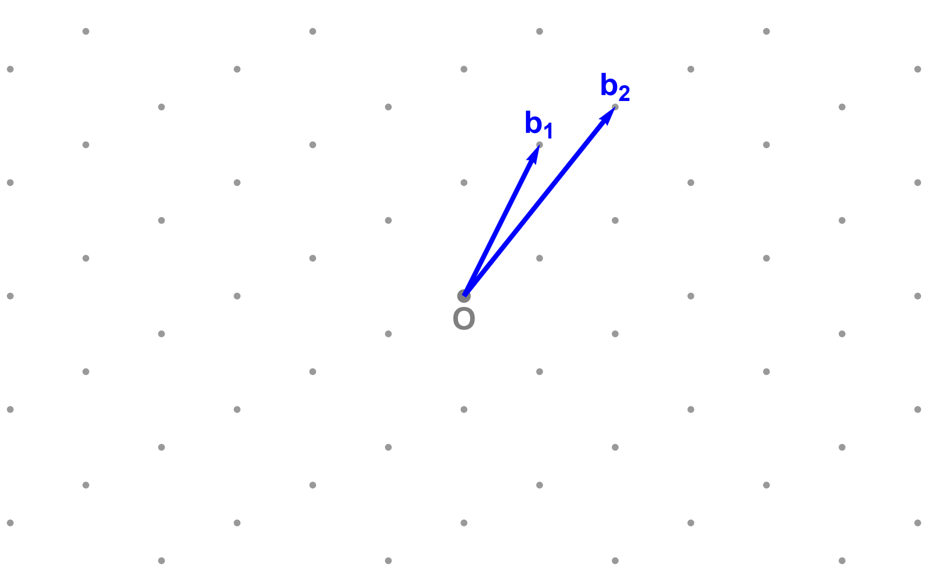
SVP hardness

- Theory
- Practice
- NIST submissions

Conclusion

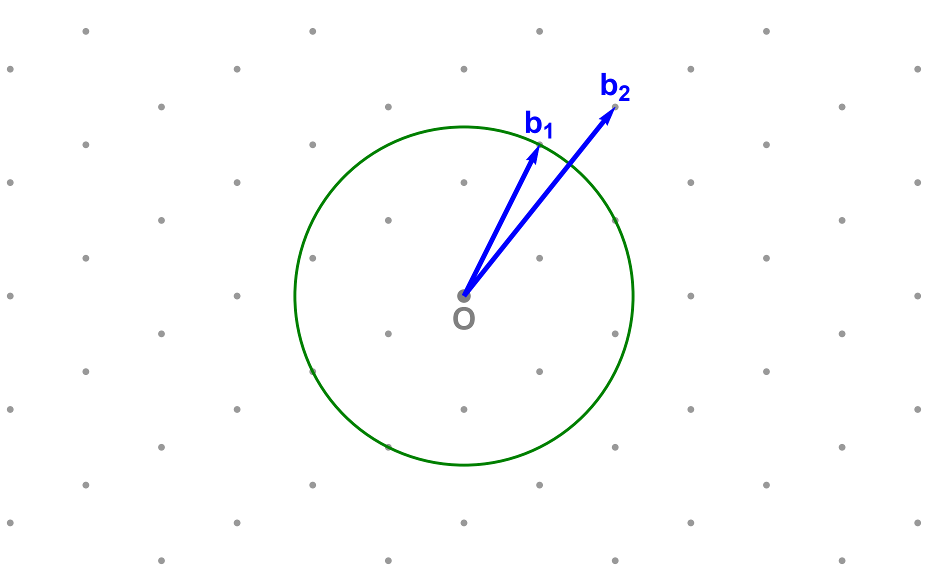
Enumeration

1. Determine possible coefficients of b_2



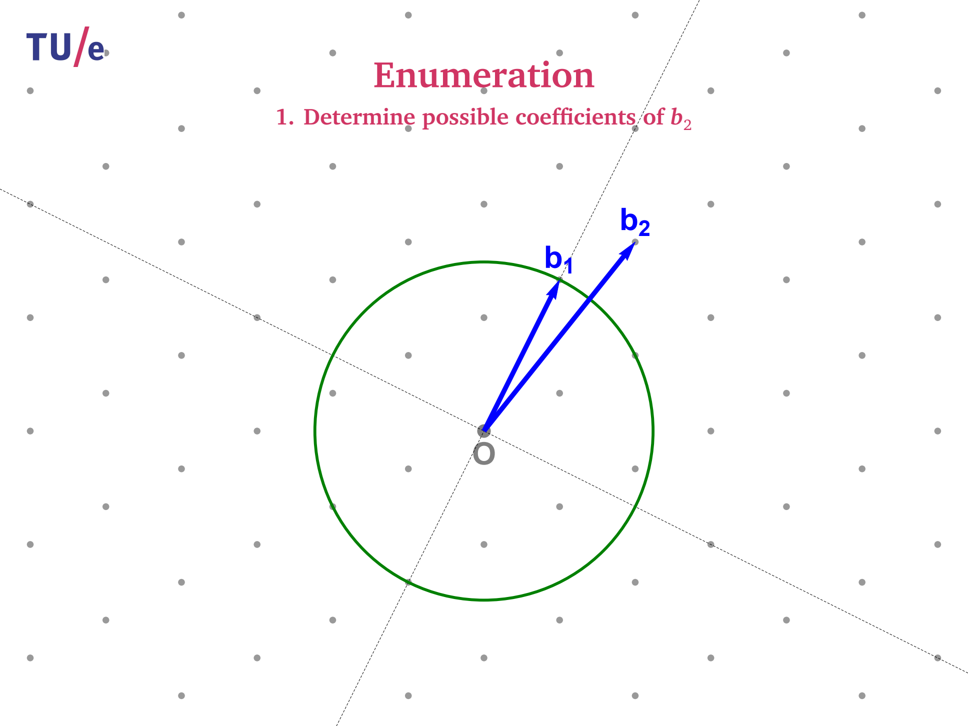
Enumeration

1. Determine possible coefficients of b_2



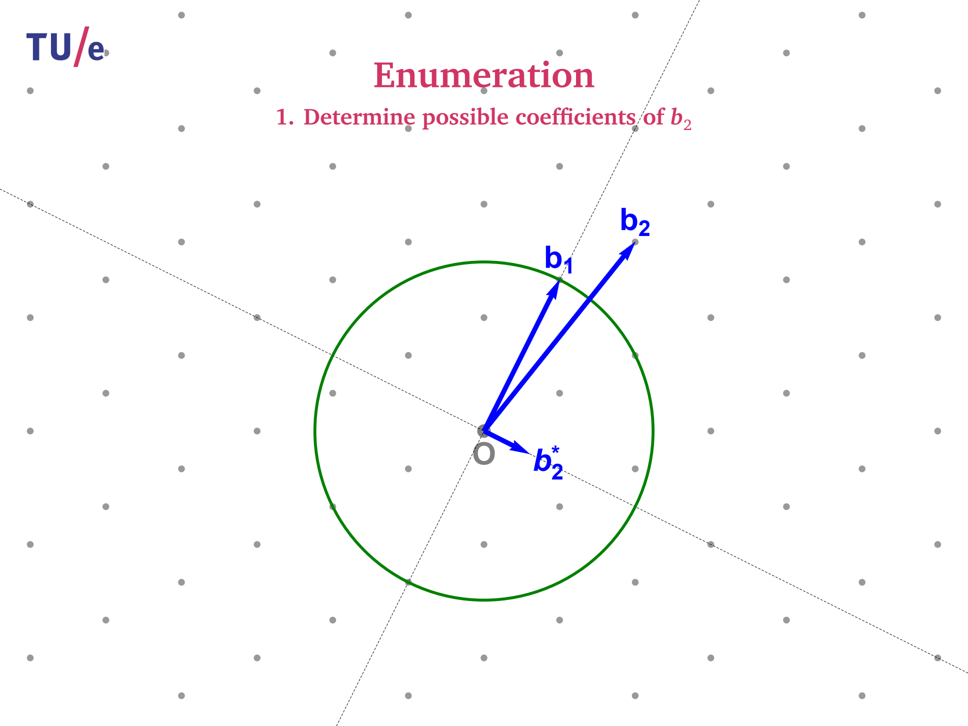
Enumeration

1. Determine possible coefficients of b_2



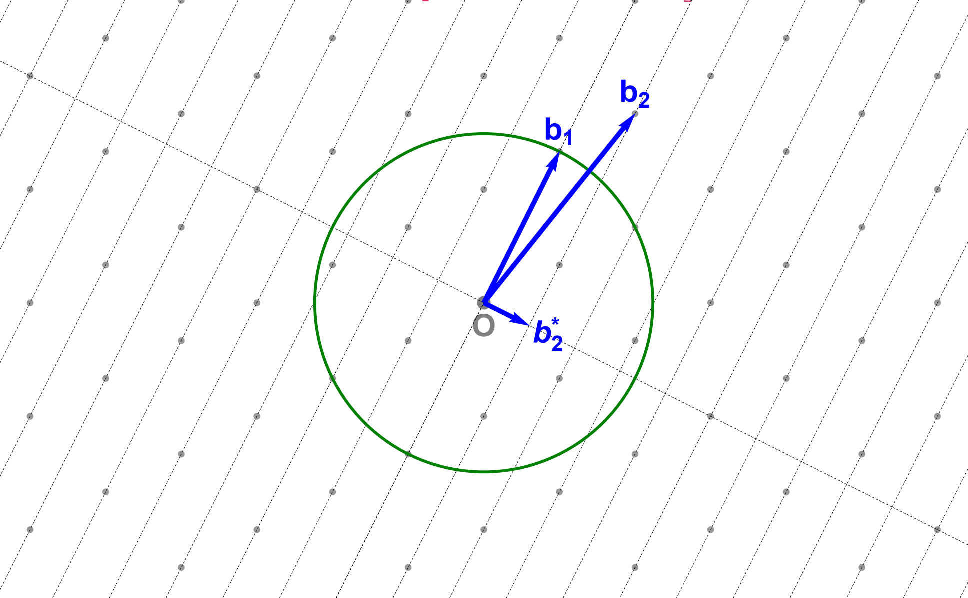
Enumeration

1. Determine possible coefficients of b_2



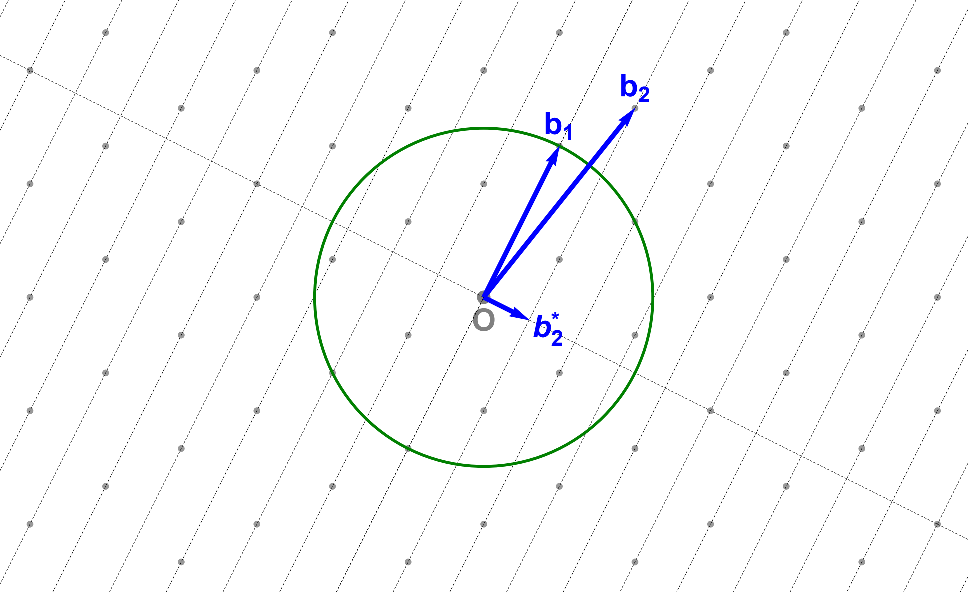
Enumeration

1. Determine possible coefficients of b_2



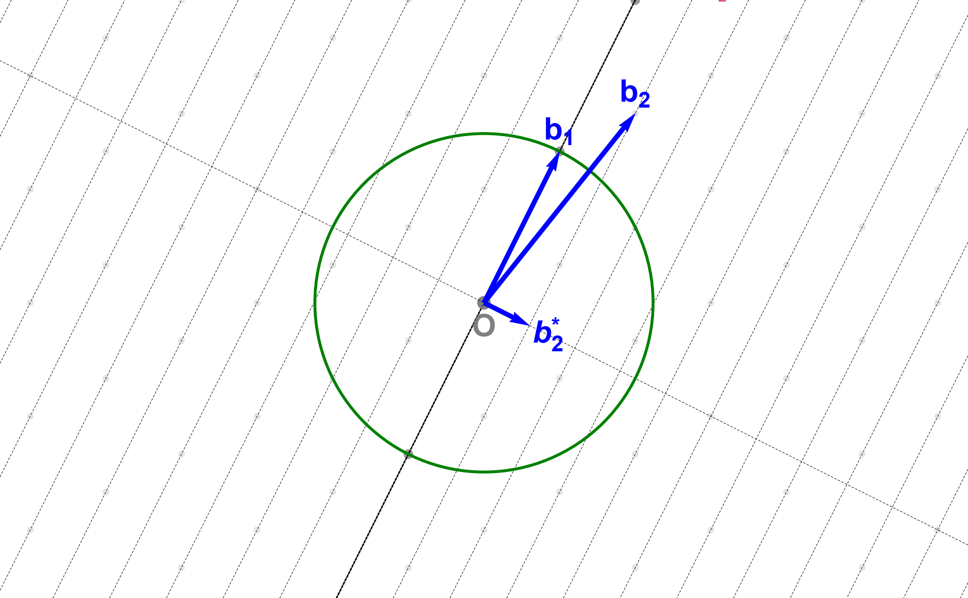
Enumeration

2. Find short vectors for each coefficient of b_2



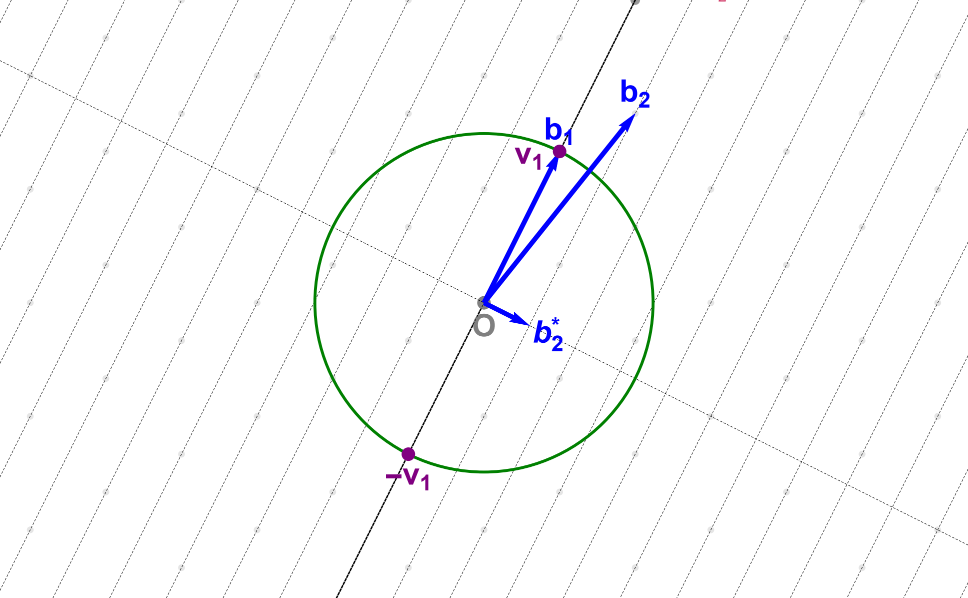
Enumeration

2. Find short vectors for each coefficient of b_2



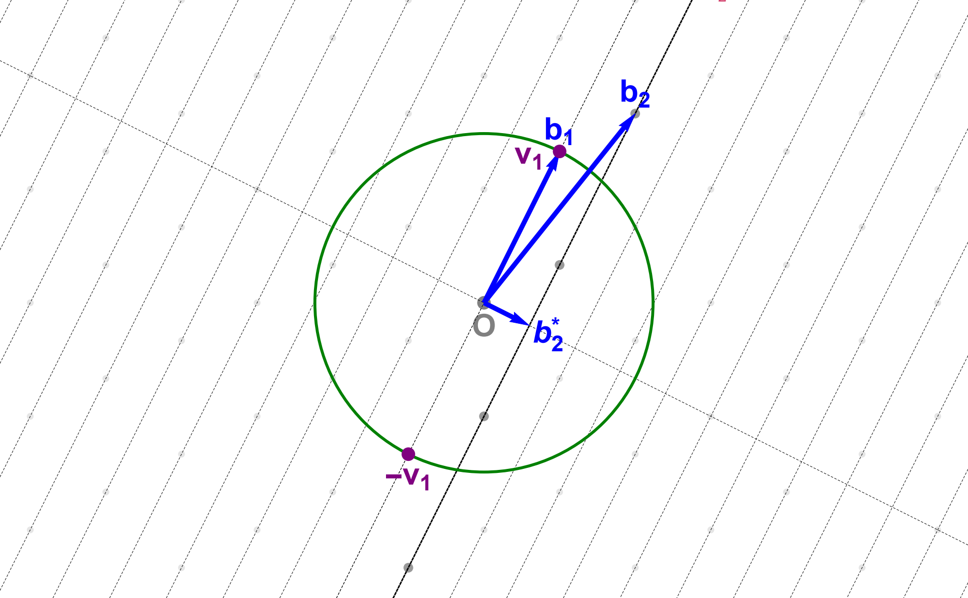
Enumeration

2. Find short vectors for each coefficient of b_2



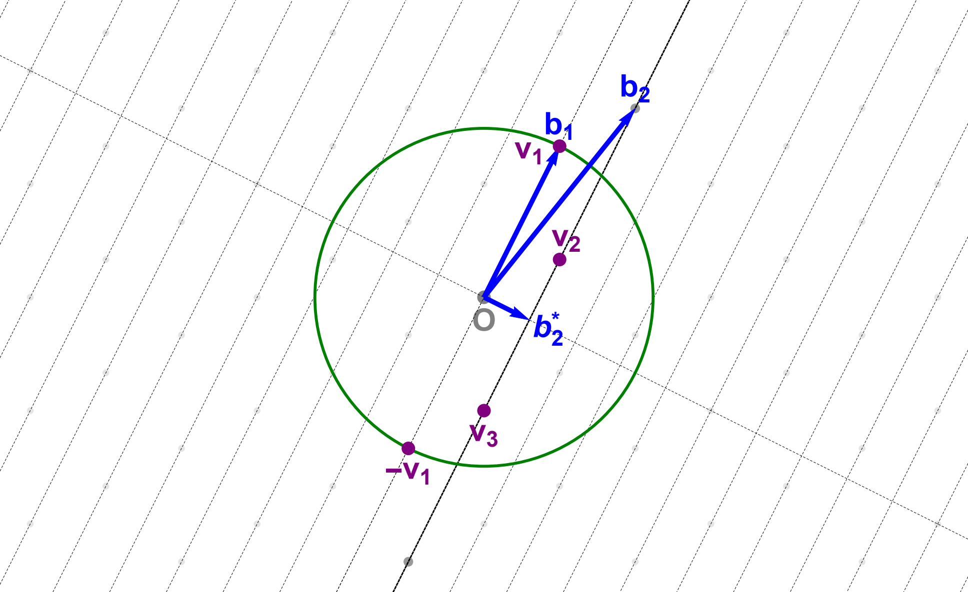
Enumeration

2. Find short vectors for each coefficient of b_2



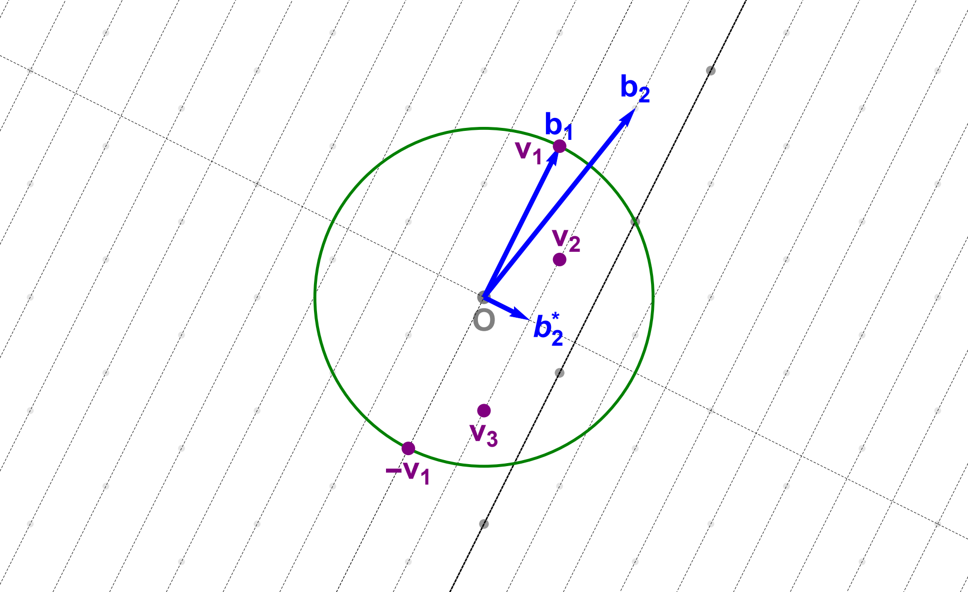
Enumeration

2. Find short vectors for each coefficient of b_2



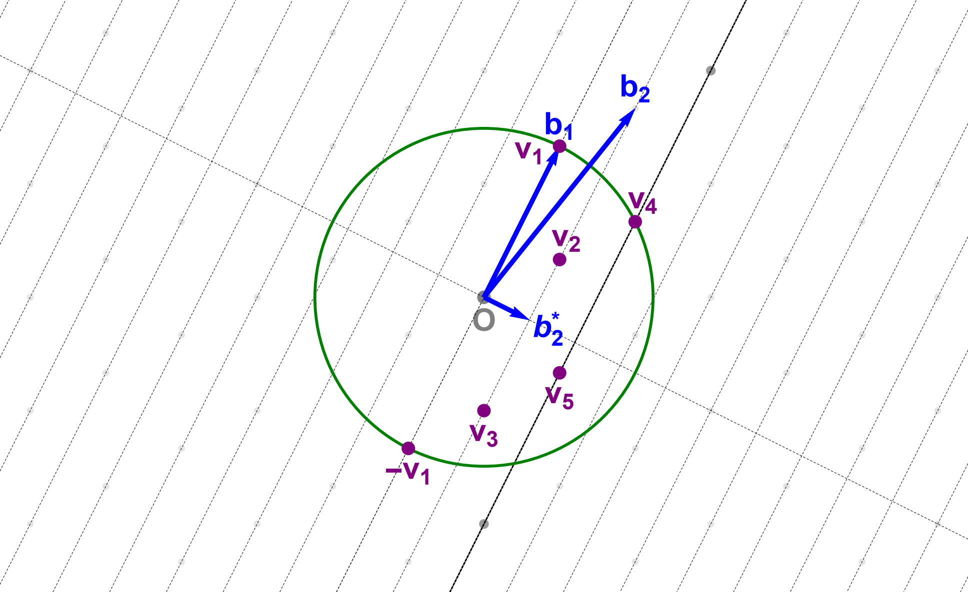
Enumeration

2. Find short vectors for each coefficient of b_2



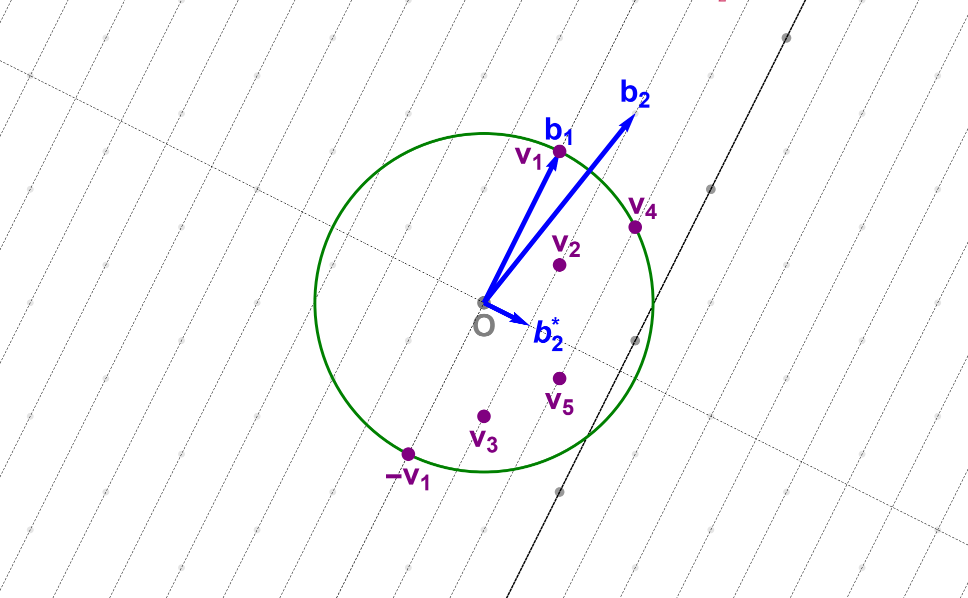
Enumeration

2. Find short vectors for each coefficient of b_2



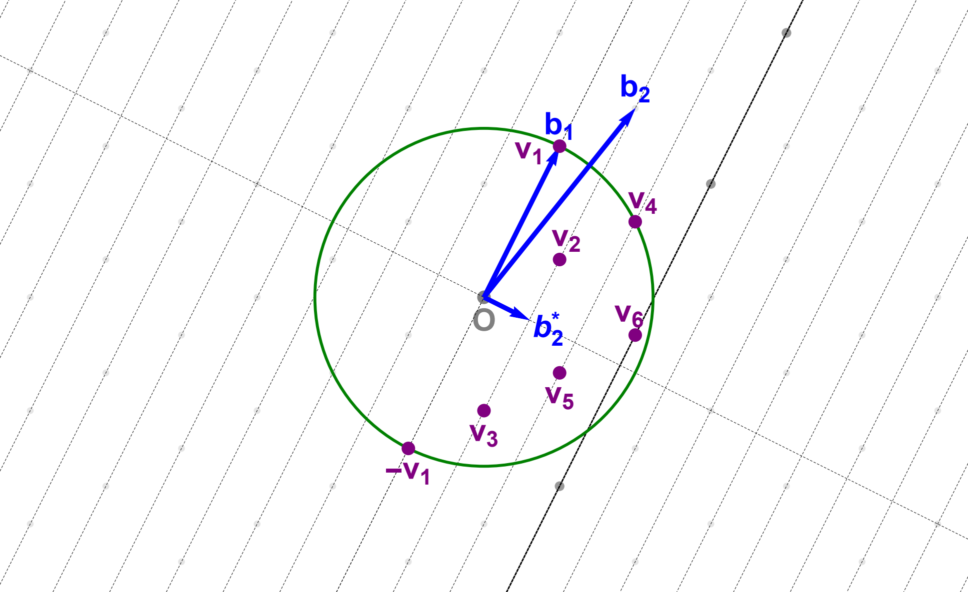
Enumeration

2. Find short vectors for each coefficient of b_2



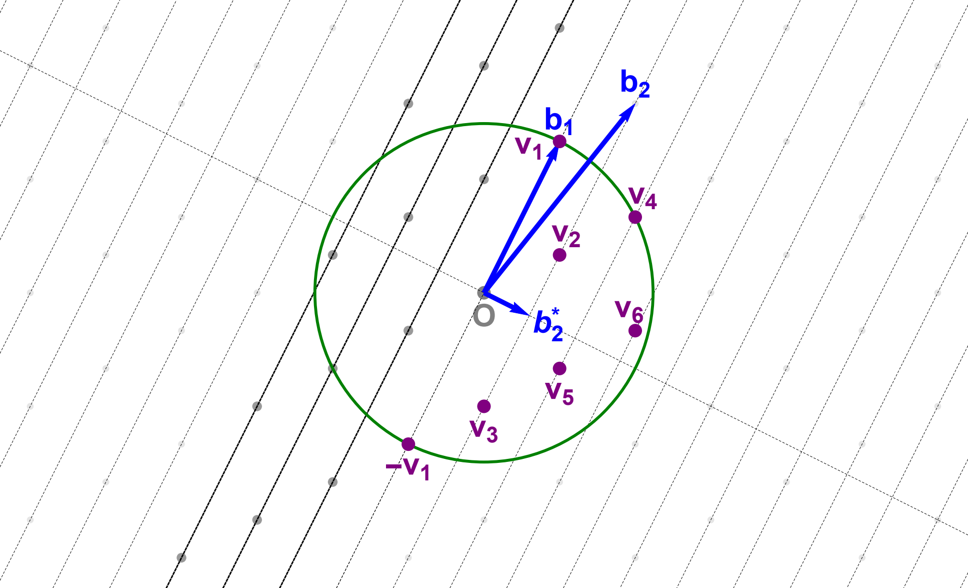
Enumeration

2. Find short vectors for each coefficient of b_2



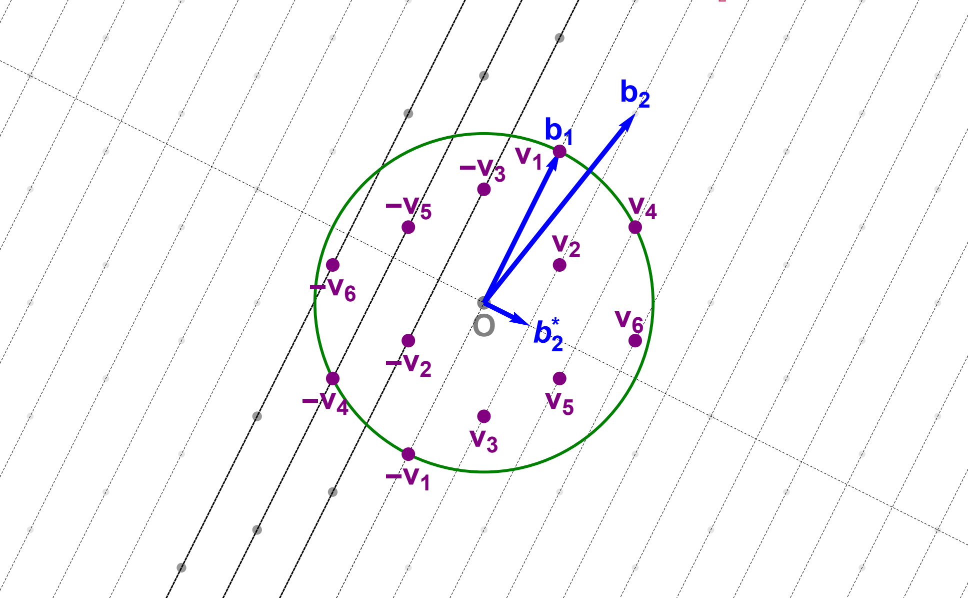
Enumeration

2. Find short vectors for each coefficient of b_2



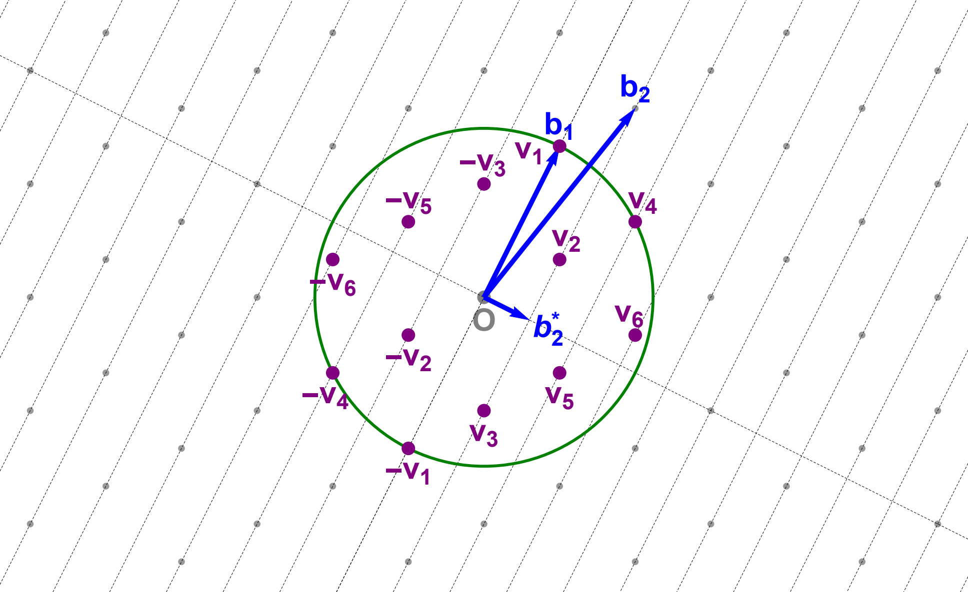
Enumeration

2. Find short vectors for each coefficient of b_2



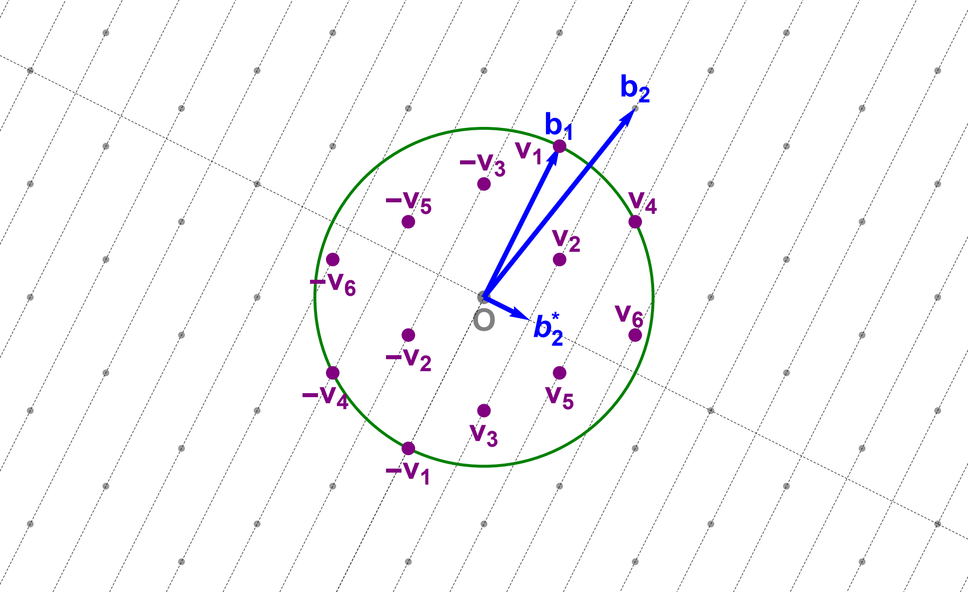
Enumeration

2. Find short vectors for each coefficient of b_2



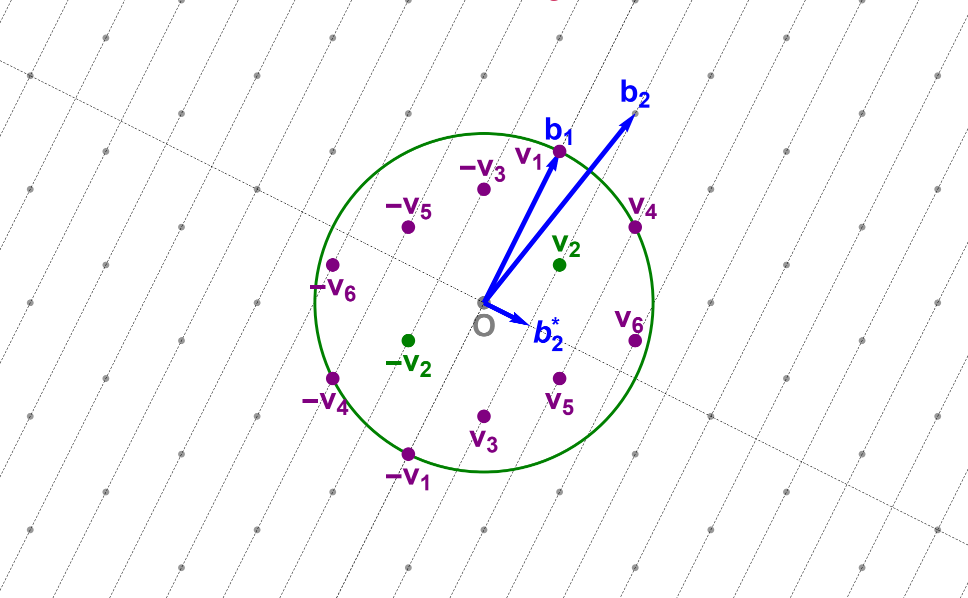
Enumeration

3. Find a shortest vector among all found vectors



Enumeration

3. Find a shortest vector among all found vectors

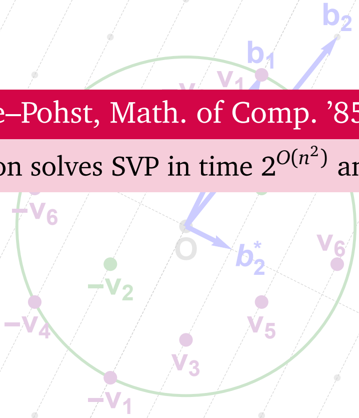


Enumeration

Overview

Enumeration

Overview



Theorem (Fincke–Pohst, Math. of Comp. ’85)

Lattice enumeration solves SVP in time $2^{O(n^2)}$ and space poly(n).

Enumeration

Overview

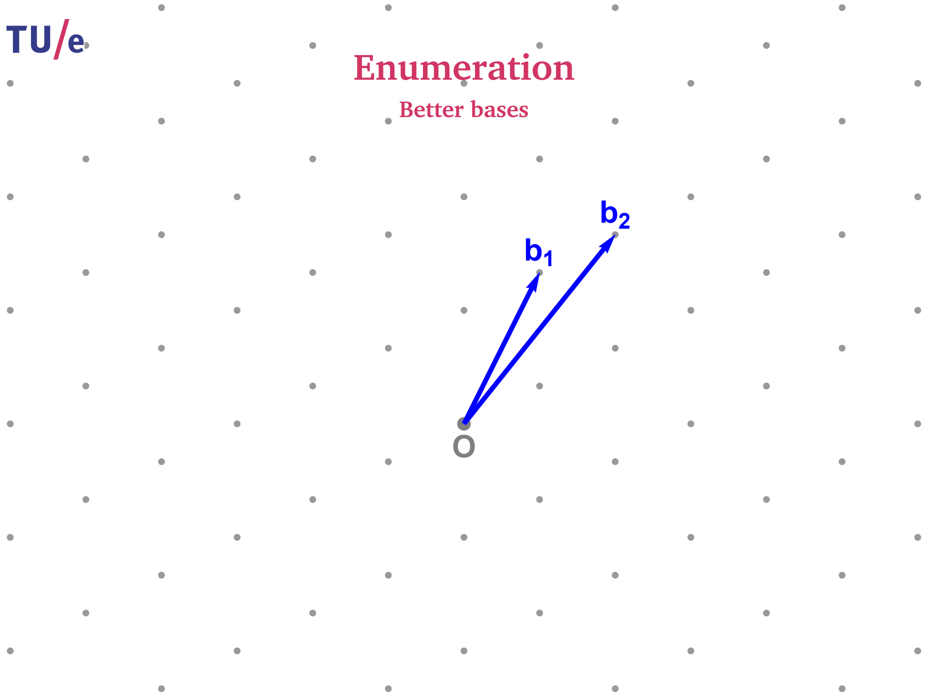
Theorem (Fincke–Pohst, Math. of Comp. ’85)

Lattice enumeration solves SVP in time $2^{O(n^2)}$ and space poly(n).

Essentially reduces SVP_n (CVP_n) to $2^{O(n)}$ instances of CVP_{n-1}.

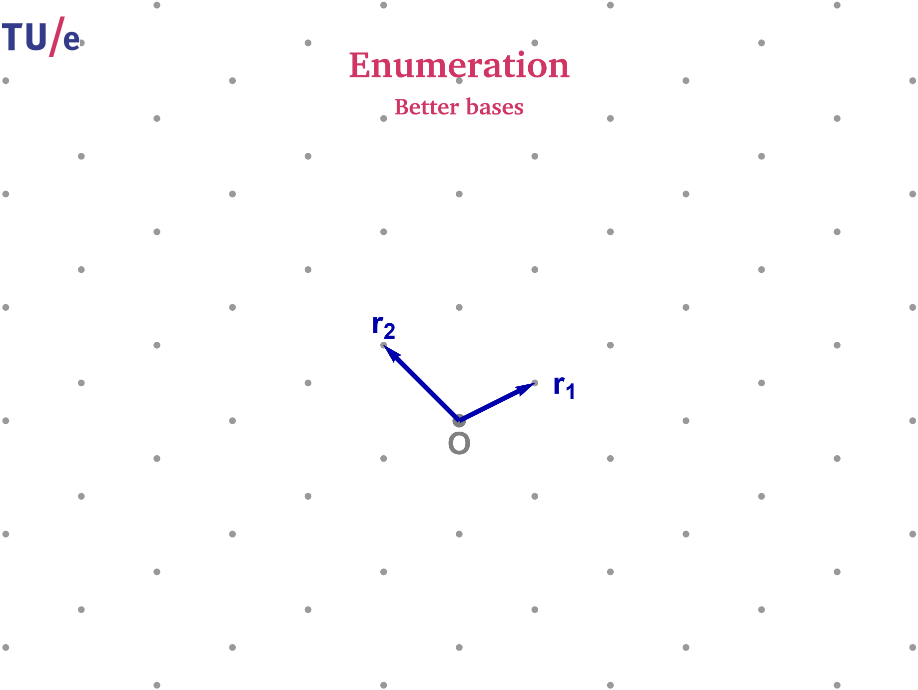
Enumeration

Better bases



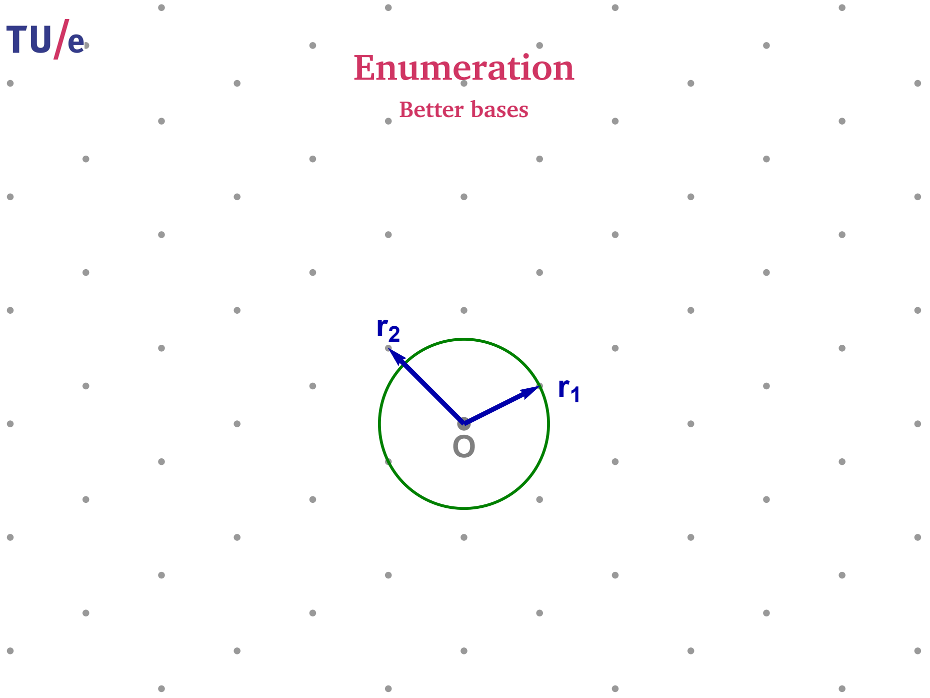
Enumeration

Better bases



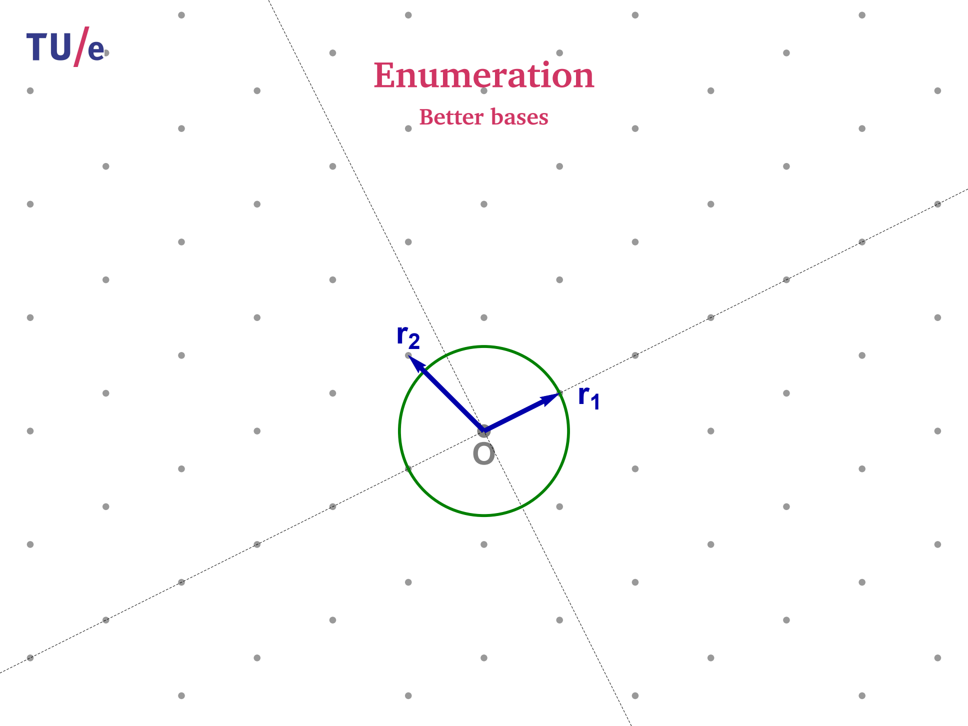
Enumeration

Better bases



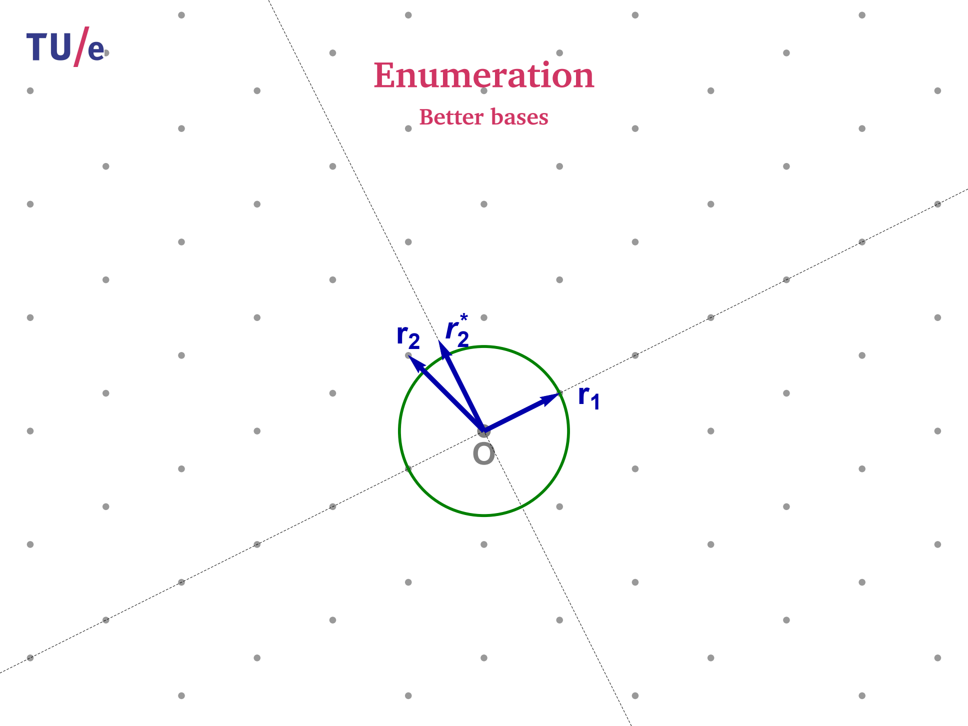
Enumeration

Better bases



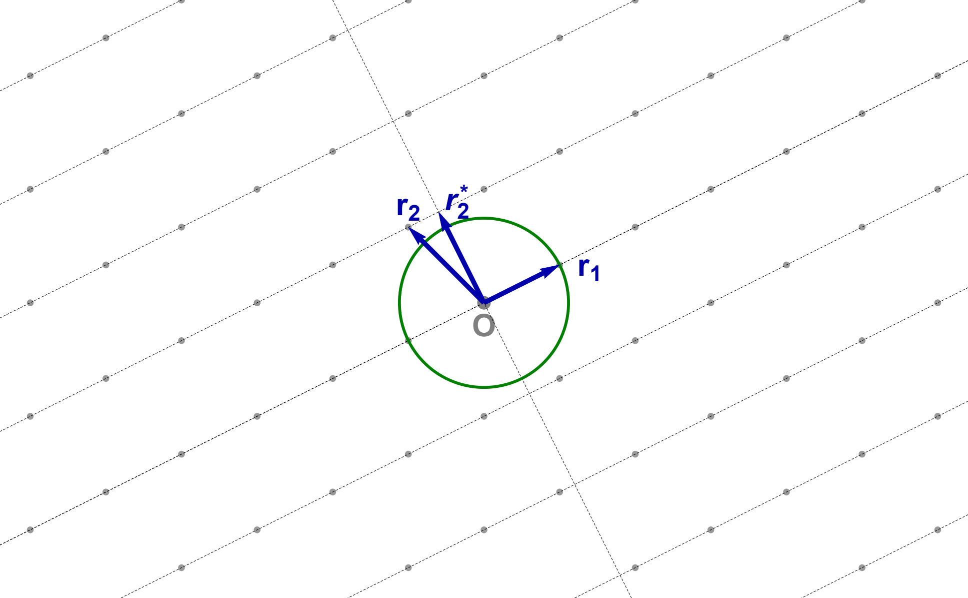
Enumeration

Better bases



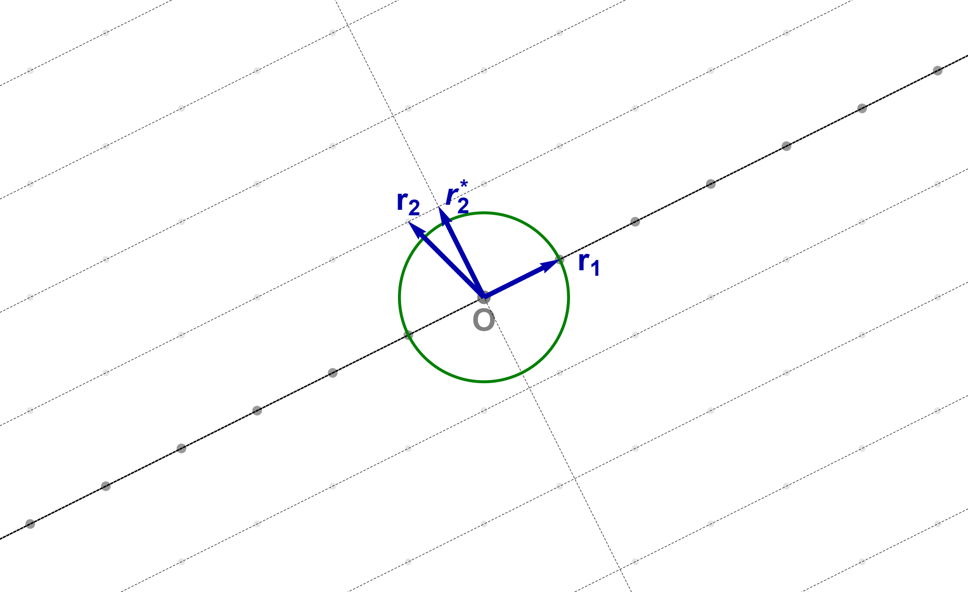
Enumeration

Better bases



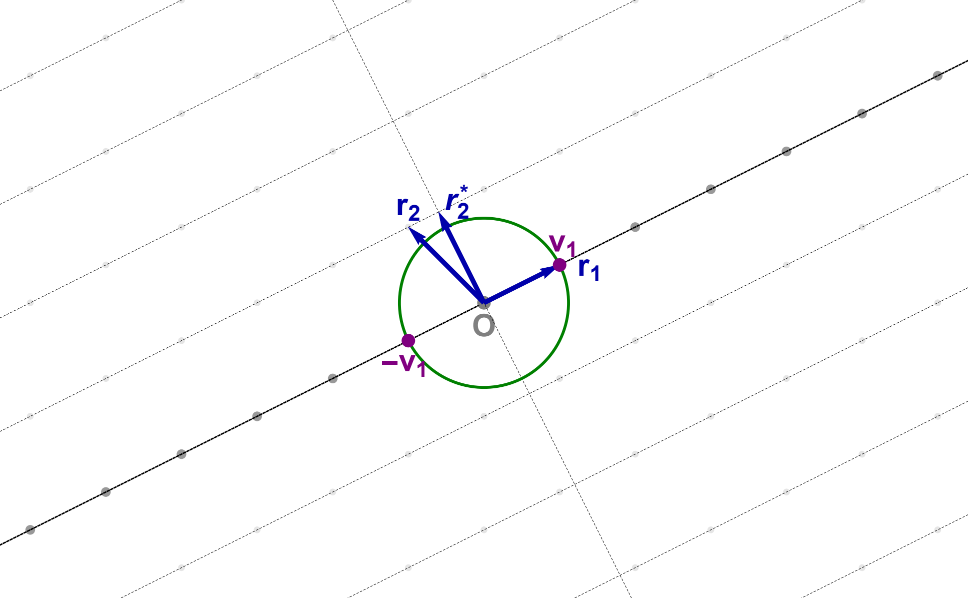
Enumeration

Better bases



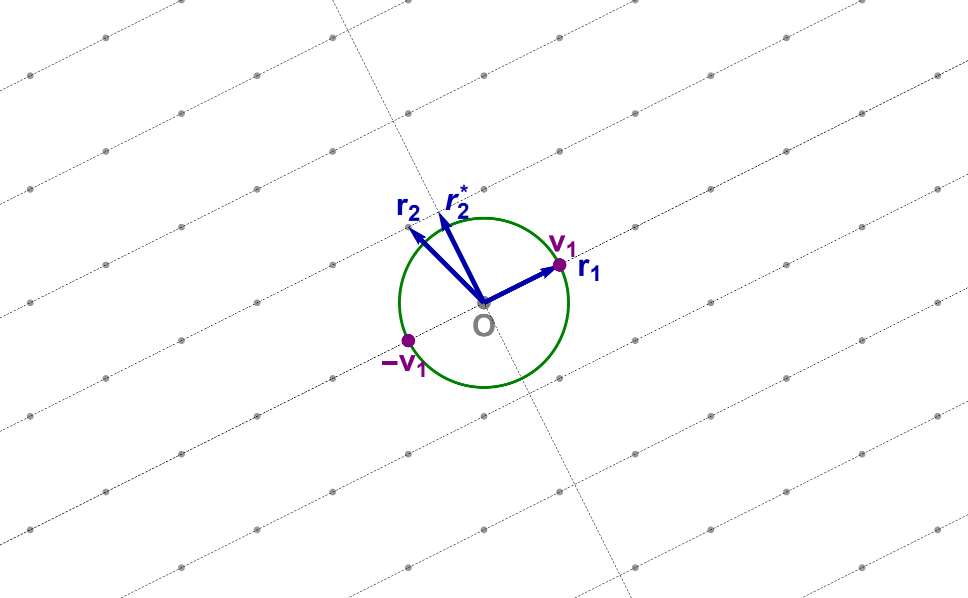
Enumeration

Better bases



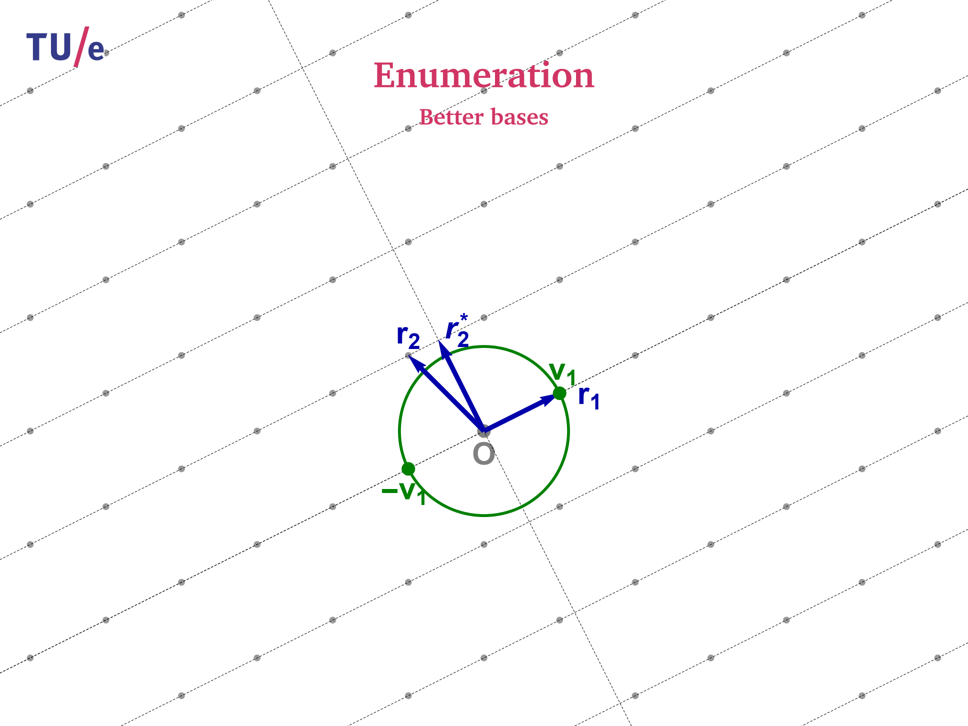
Enumeration

Better bases



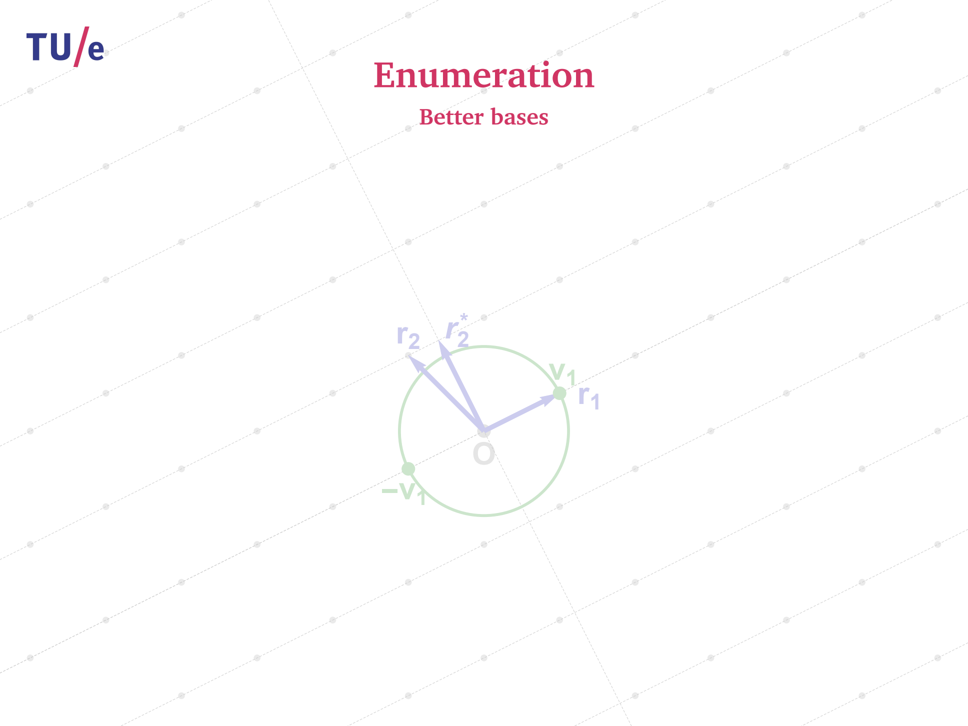
Enumeration

Better bases



Enumeration

Better bases



Enumeration

Better bases

Theorem (Kannan, STOC'83)

Combining enumeration with stronger basis reduction, one can solve SVP in time $2^{O(n \log n)}$ and space $\mathrm{poly}(n)$.

Enumeration

Better bases

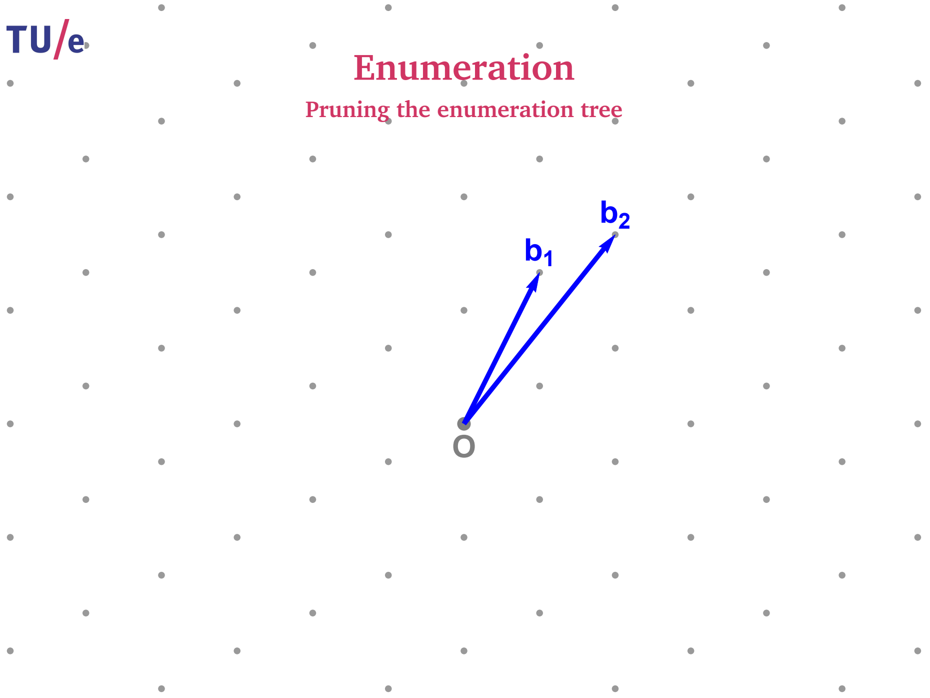
Theorem (Kannan, STOC'83)

Combining enumeration with stronger basis reduction, one can solve SVP in time $2^{O(n \log n)}$ and space $\mathrm{poly}(n)$.

> *"Our algorithm reduces an n-dimensional problem to polynomially many (instead of $2^{O(n)}$) $(n-1)$-dimensional problems. [...] The algorithm we propose, first finds a more orthogonal basis for a lattice in time $2^{O(n \log n)}$."*
>
> – Kannan, *STOC'83*

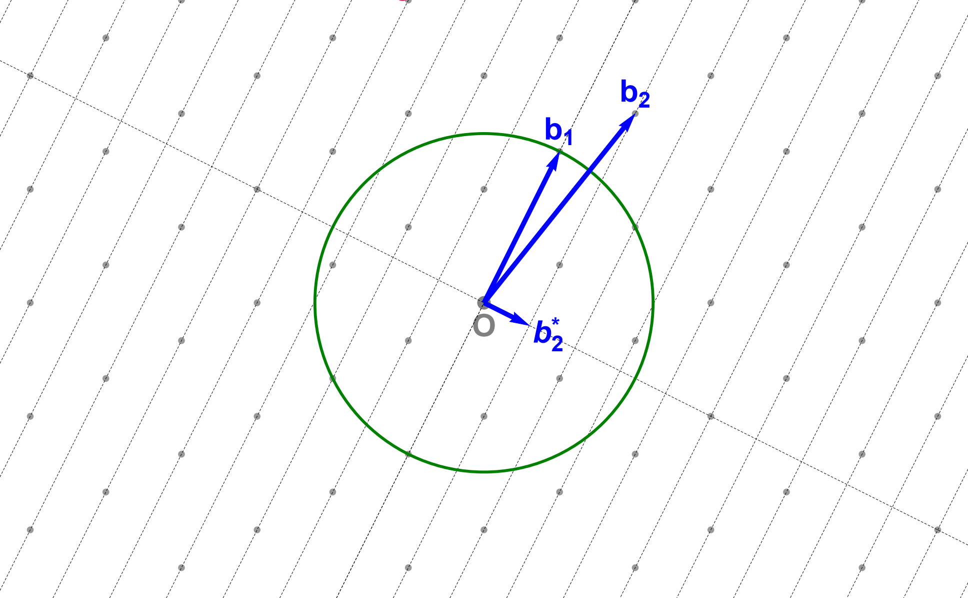
Enumeration

Pruning the enumeration tree



Enumeration

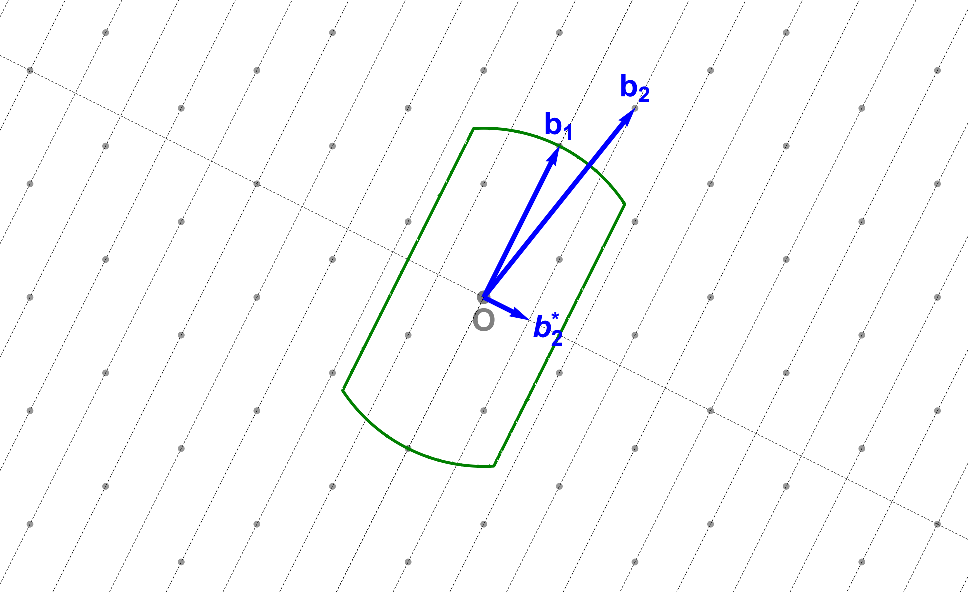
Pruning the enumeration tree



TU/e

Enumeration

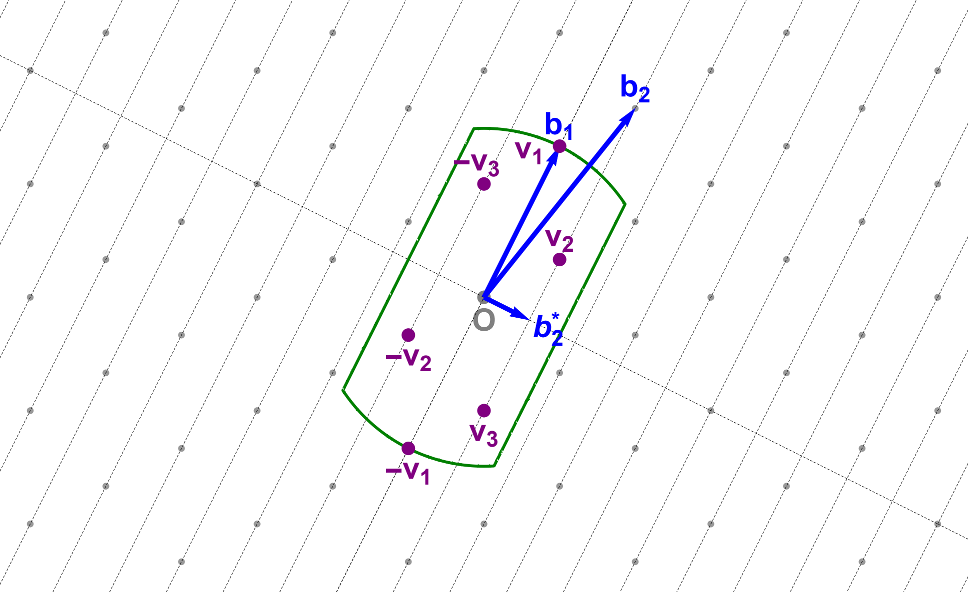
Pruning the enumeration tree



TU/e

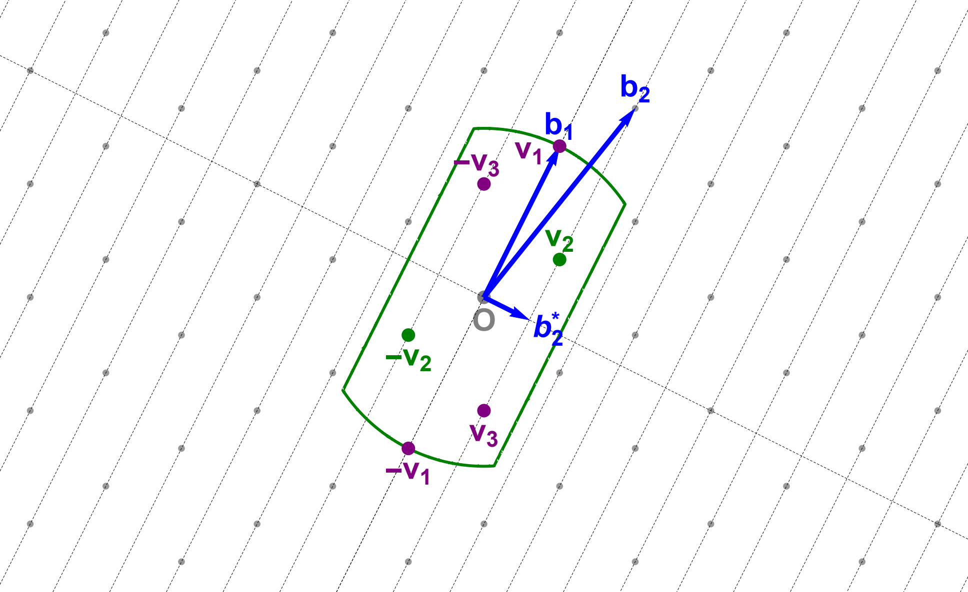
Enumeration

Pruning the enumeration tree



Enumeration

Pruning the enumeration tree



Enumeration

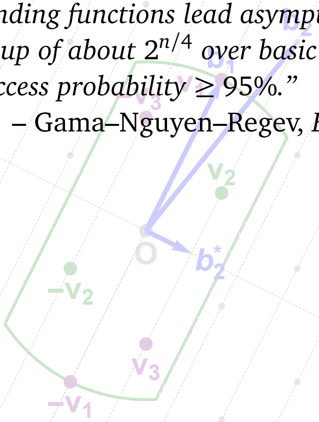
Pruning the enumeration tree

Enumeration

Pruning the enumeration tree

"Well-chosen bounding functions lead asymptotically to an exponential speedup of about $2^{n/4}$ *over basic enumeration, maintaining a success probability* $\geq 95\%$*."*

– Gama–Nguyen–Regev, *EUROCRYPT'10*



Enumeration

Pruning the enumeration tree

"Well-chosen bounding functions lead asymptotically to an exponential speedup of about $2^{n/4}$ over basic enumeration, maintaining a success probability $\geq 95\%$."

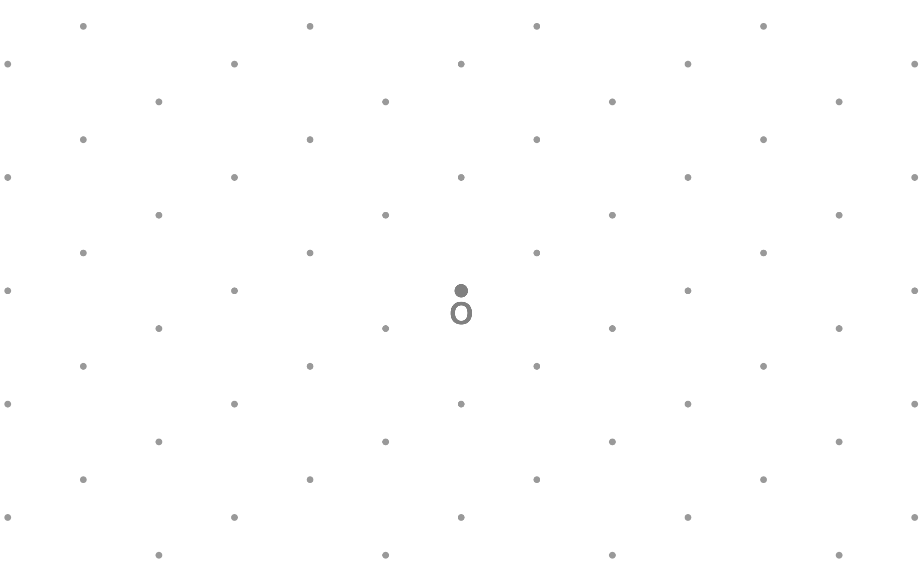
– Gama–Nguyen–Regev, *EUROCRYPT'10*

"With extreme pruning, the probability of finding the desired vector is actually rather low (say, 0.1%), but surprisingly, the running time of the enumeration is reduced by a much more significant factor (say, much more than 1000)."

– Gama–Nguyen–Regev, *EUROCRYPT'10*

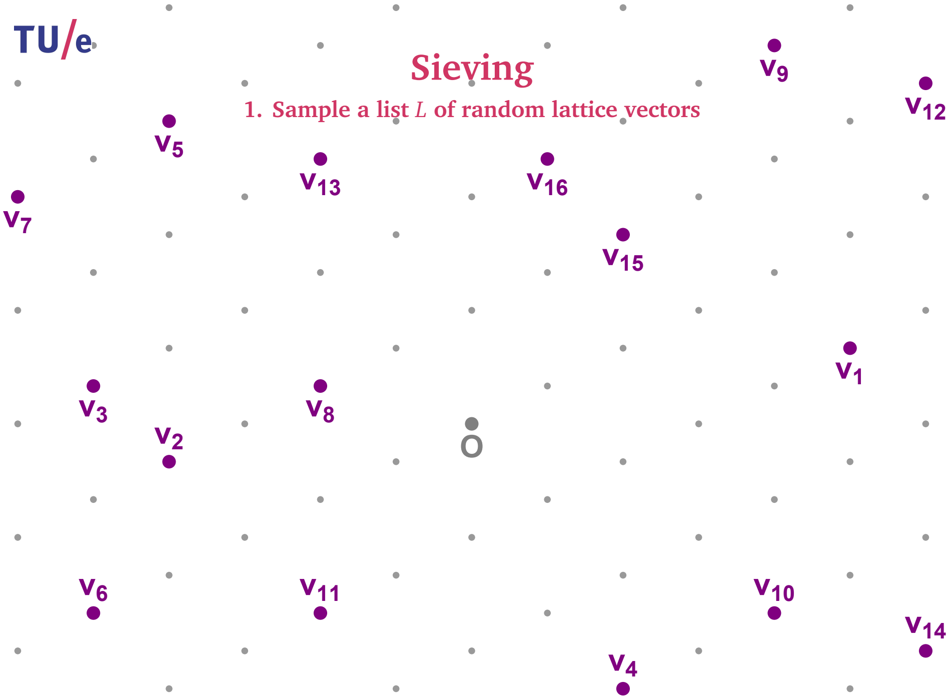
Sieving

1. Sample a list L of random lattice vectors



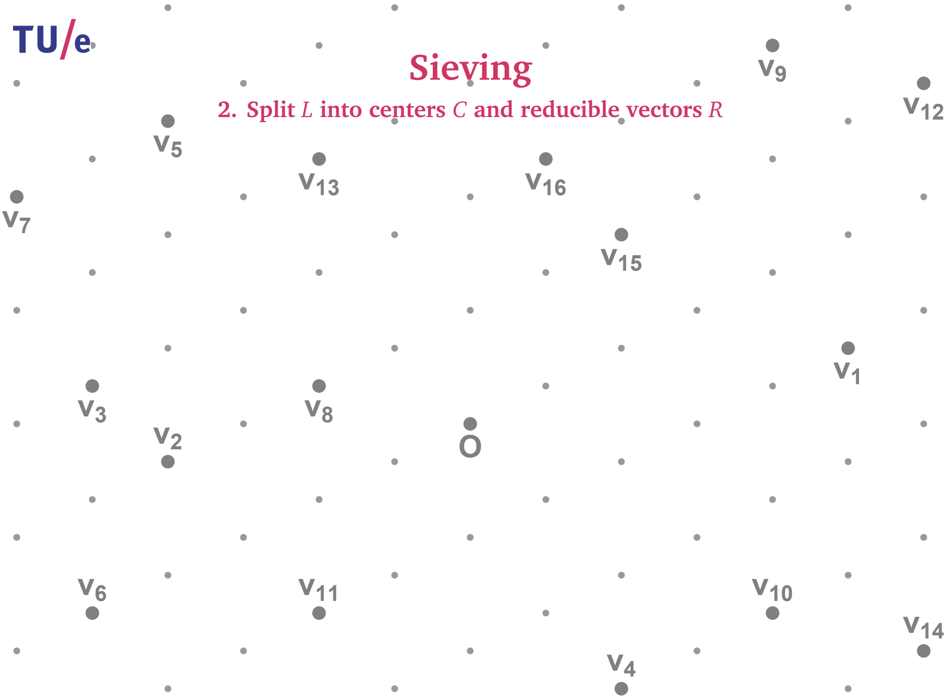
Sieving

1. Sample a list L of random lattice vectors



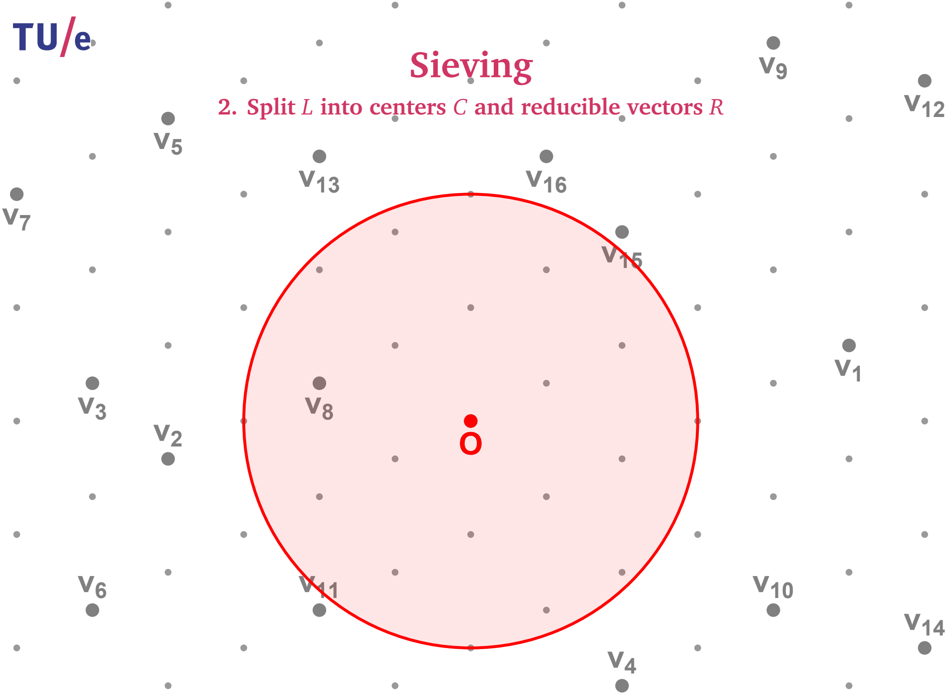
Sieving

2. Split L into centers C and reducible vectors R



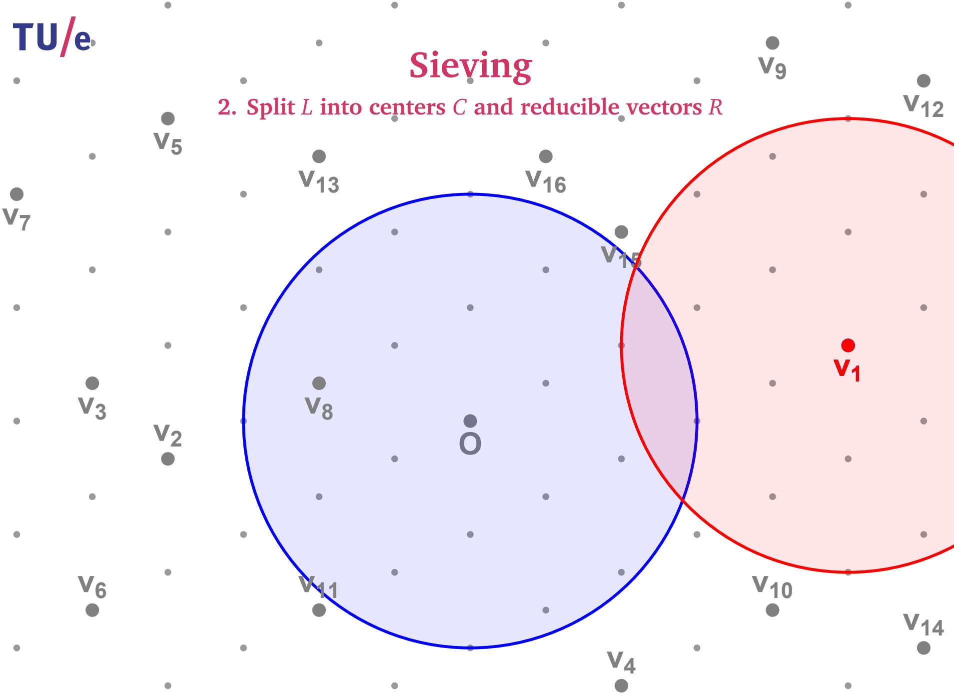
Sieving

2. Split L into centers C and reducible vectors R



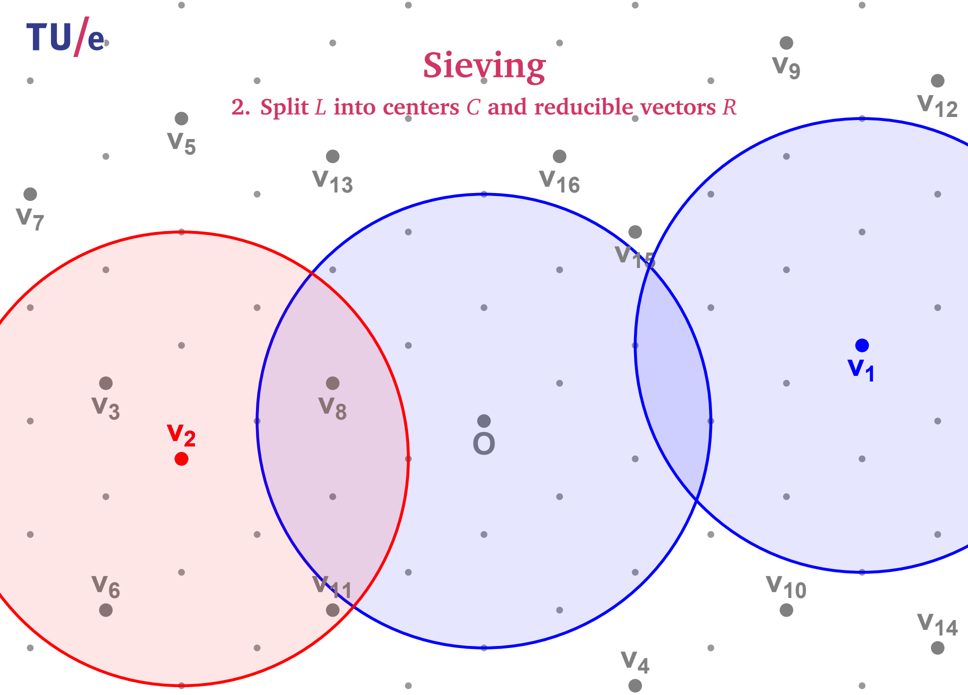
Sieving

2. Split L into centers C and reducible vectors R



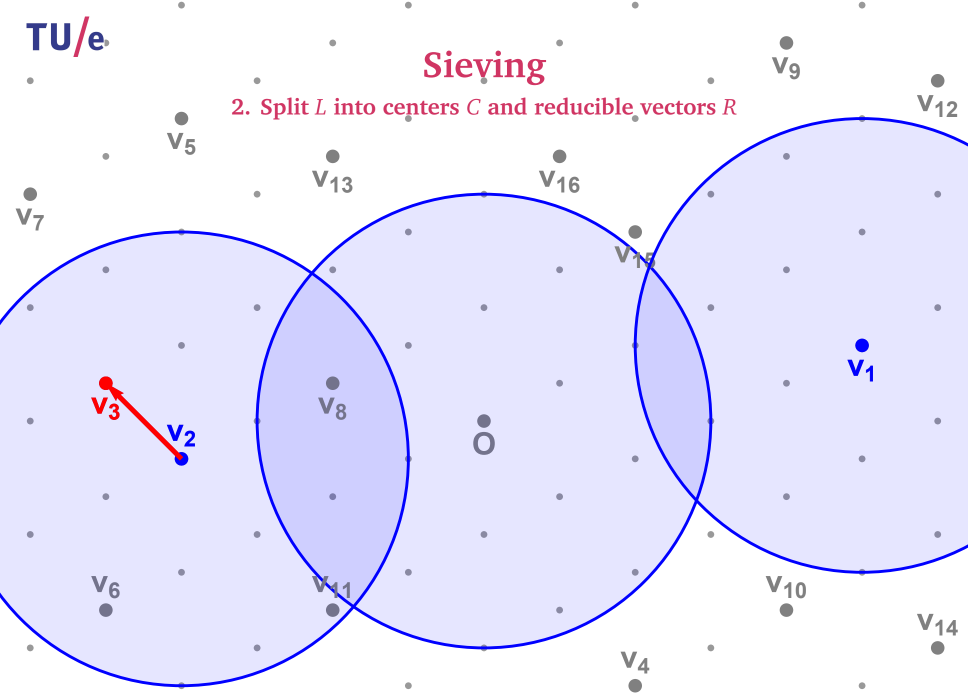
Sieving

2. Split L into centers C and reducible vectors R



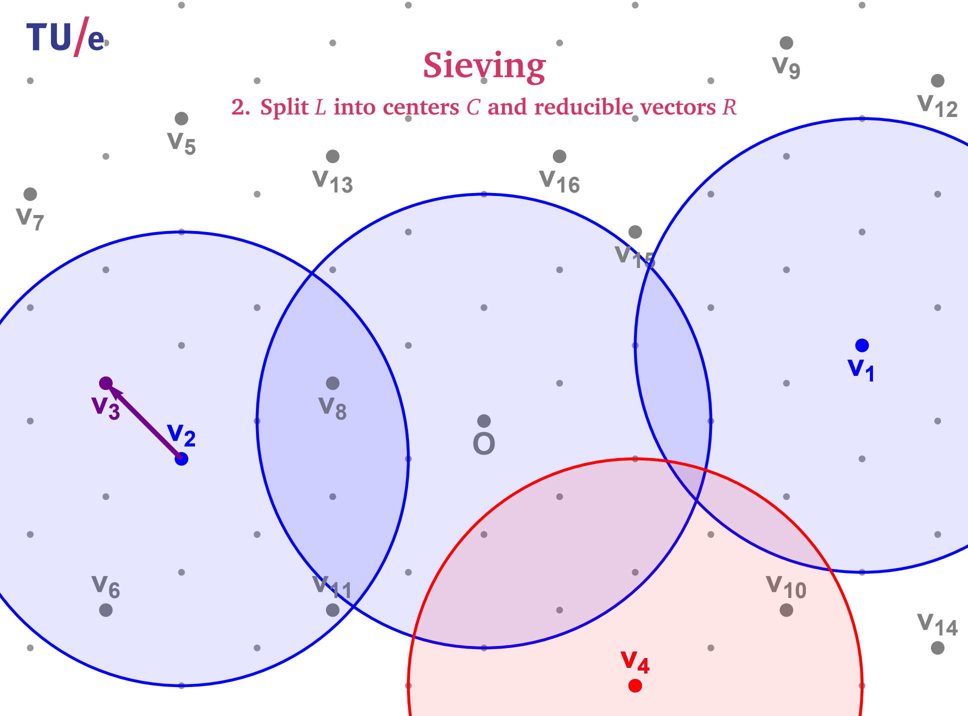
Sieving

2. Split L into centers C and reducible vectors R



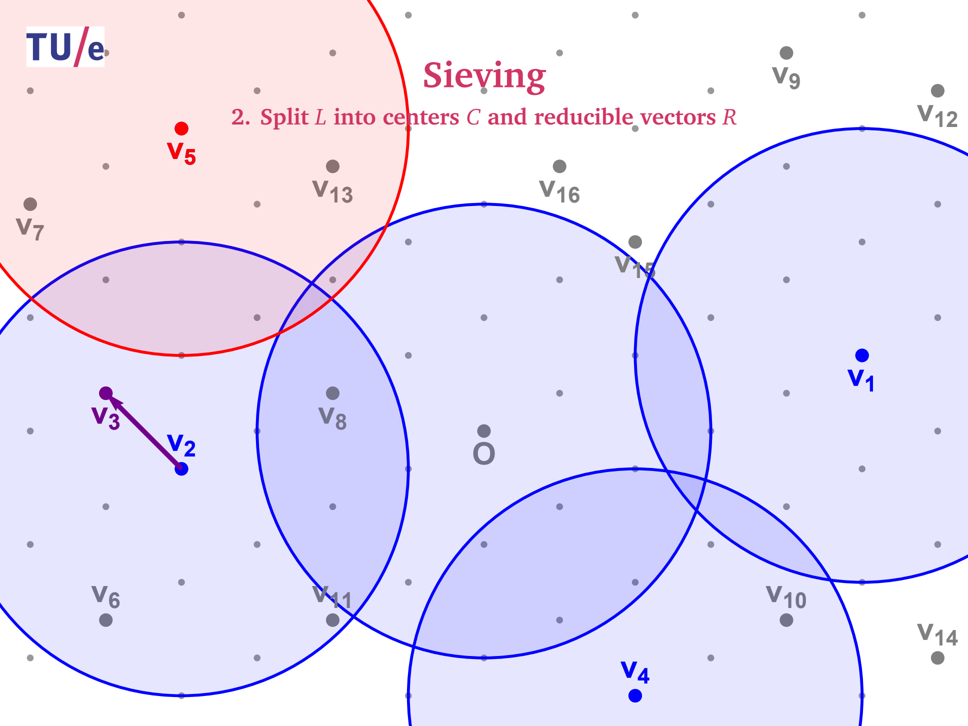
Sieving

2. Split L into centers C and reducible vectors R



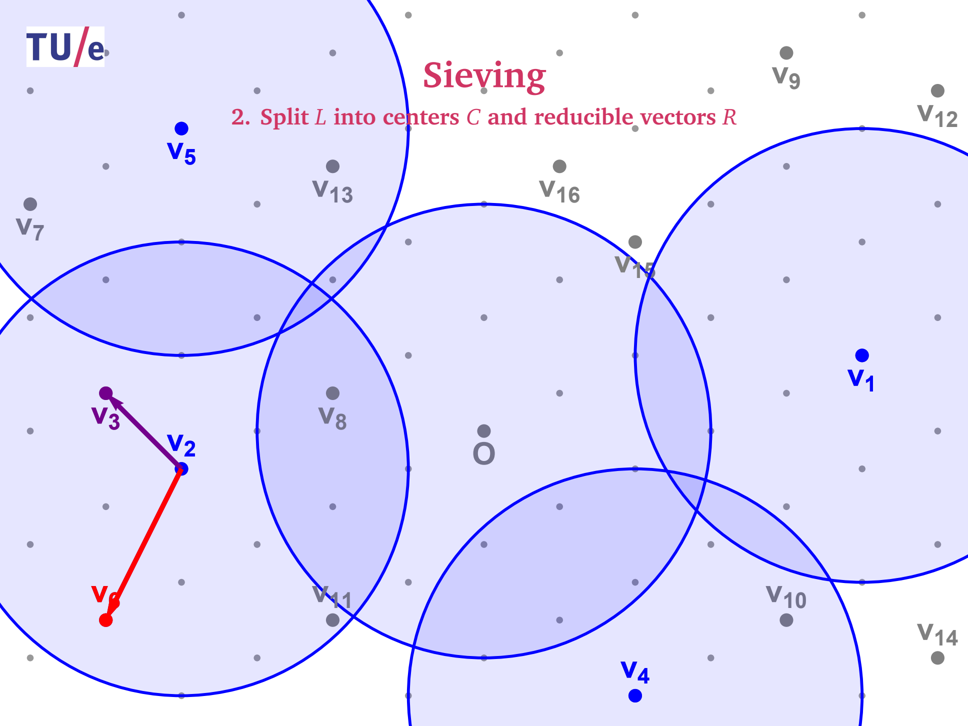
Sieving

2. Split L into centers C and reducible vectors R



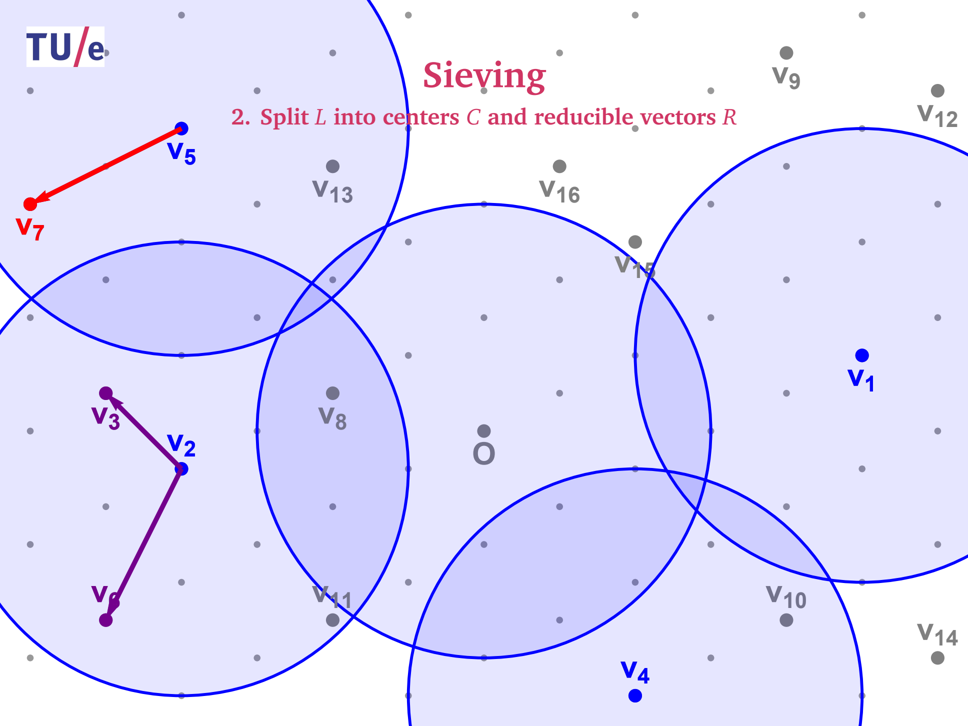
Sieving

2. Split L into centers C and reducible vectors R



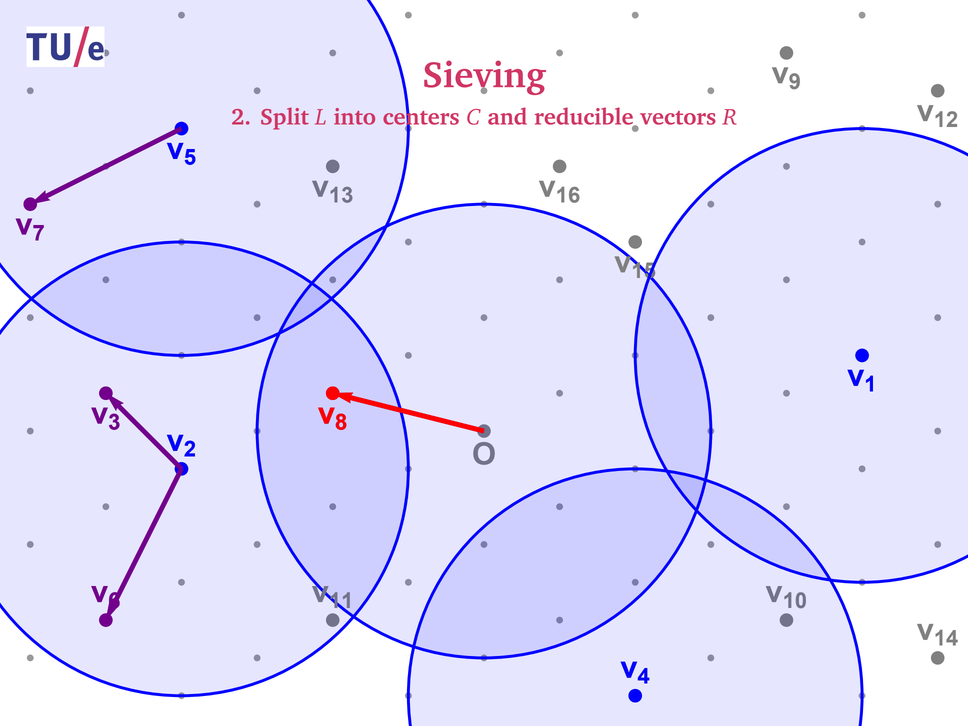
Sieving

2. Split L into centers C and reducible vectors R



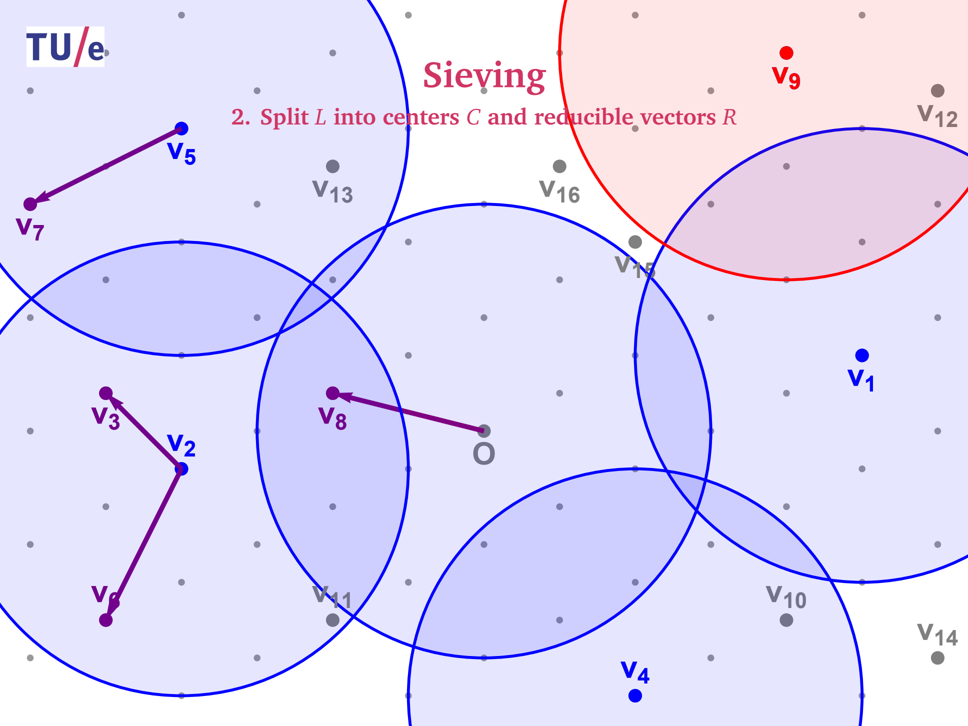
Sieving

2. Split L into centers C and reducible vectors R



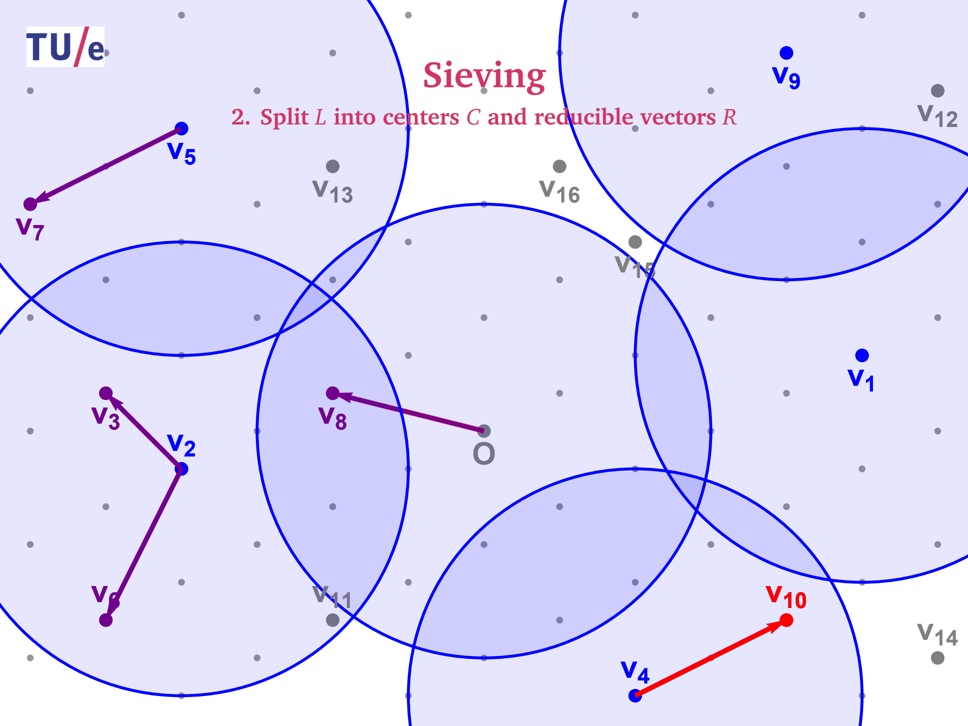
Sieving

2. Split L into centers C and reducible vectors R



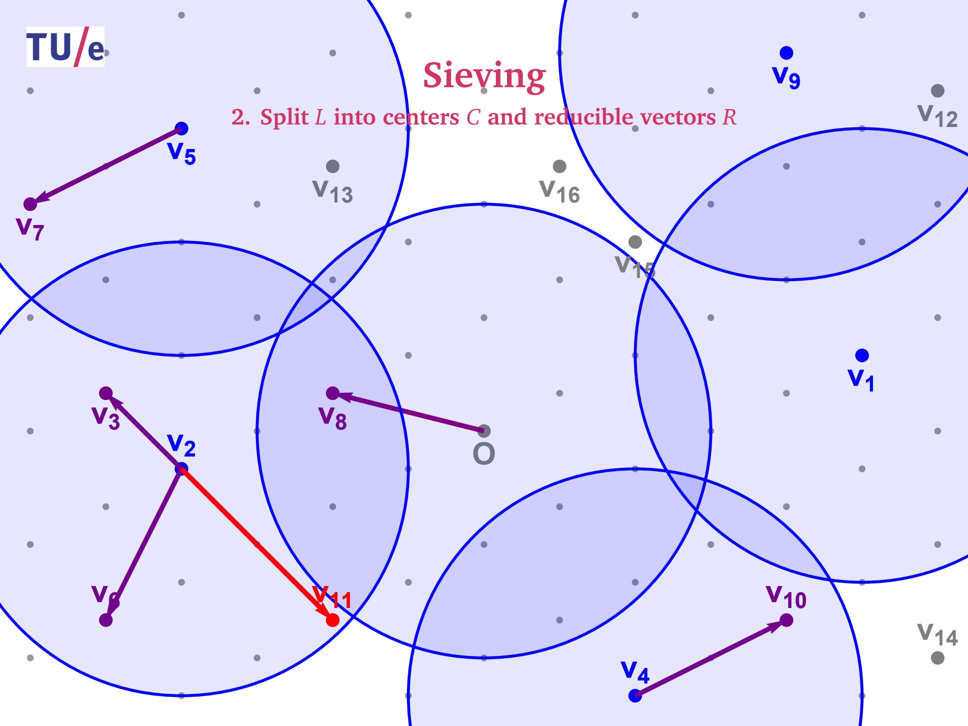
Sieving

2. Split L into centers C and reducible vectors R



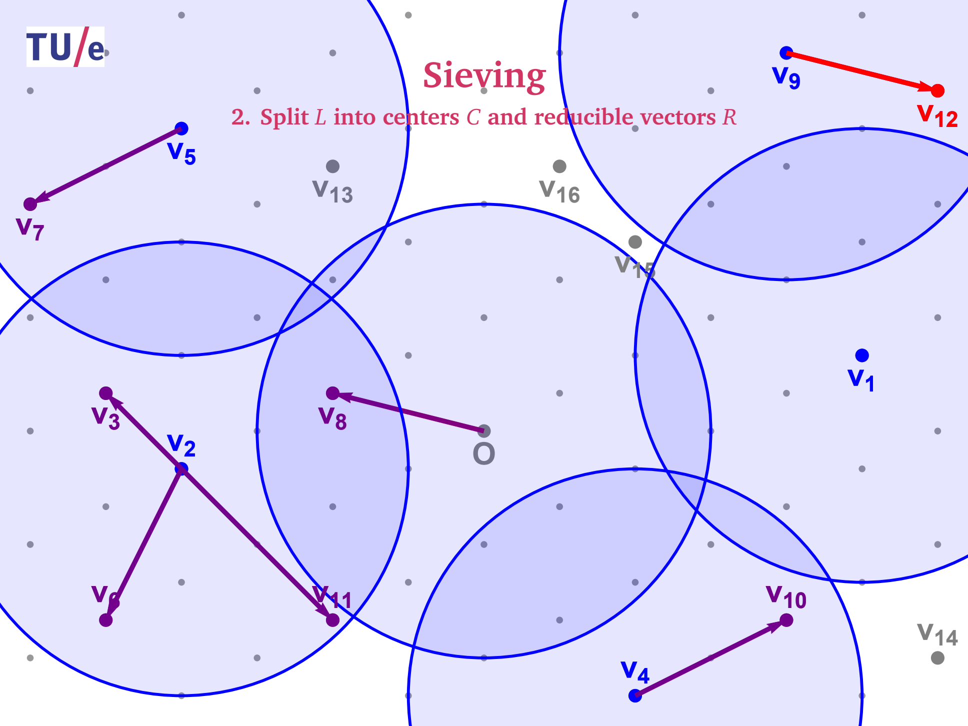
Sieving

2. Split L into centers C and reducible vectors R



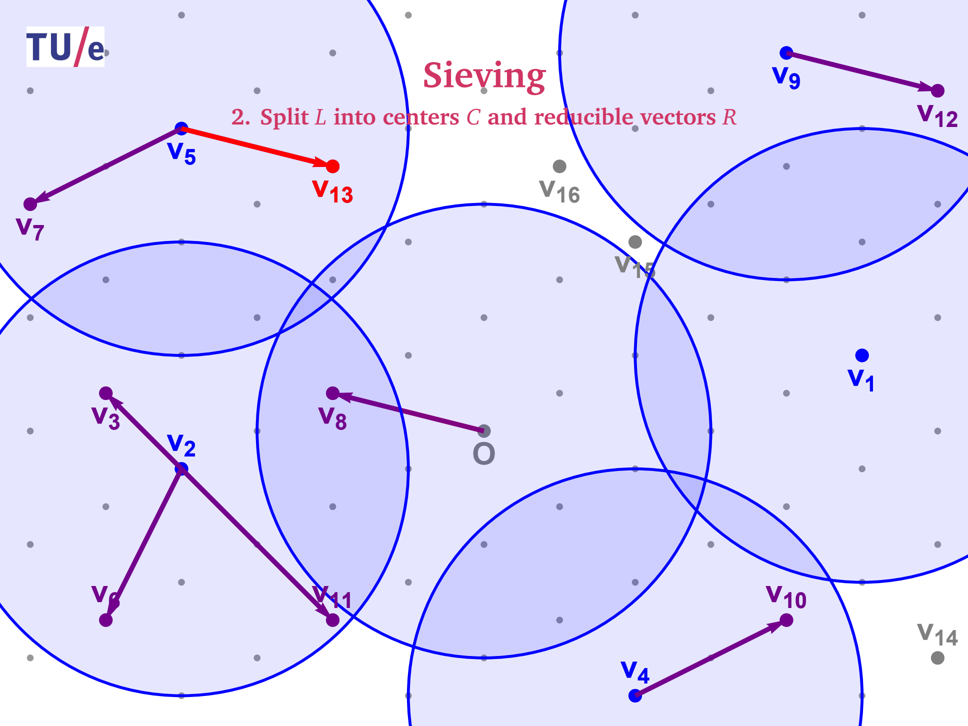
Sieving

2. Split L into centers C and reducible vectors R



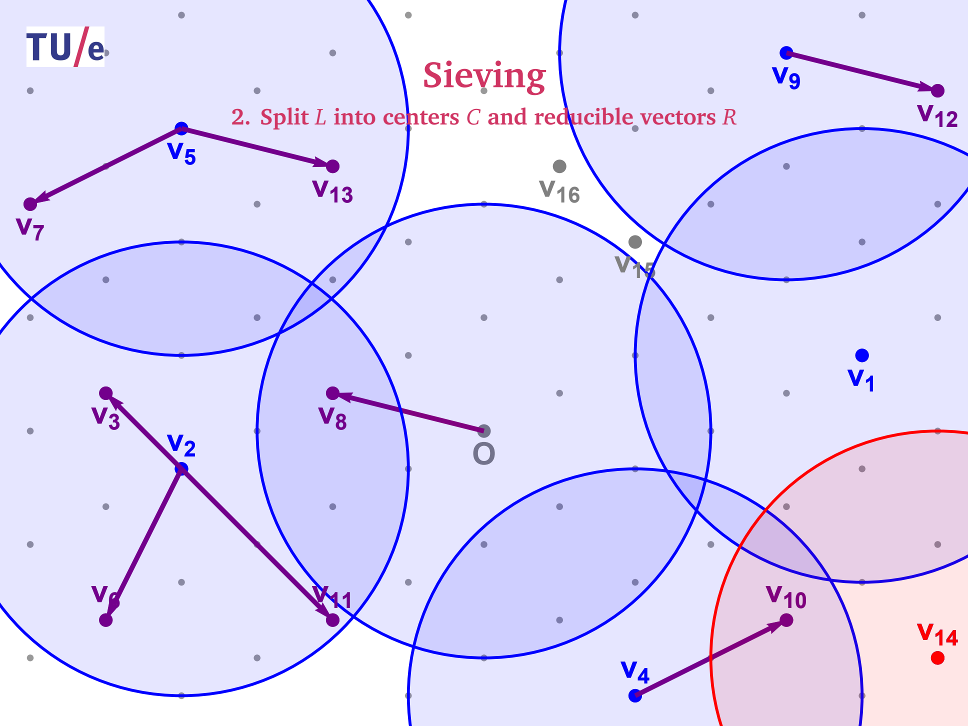
Sieving

2. Split L into centers C and reducible vectors R



Sieving

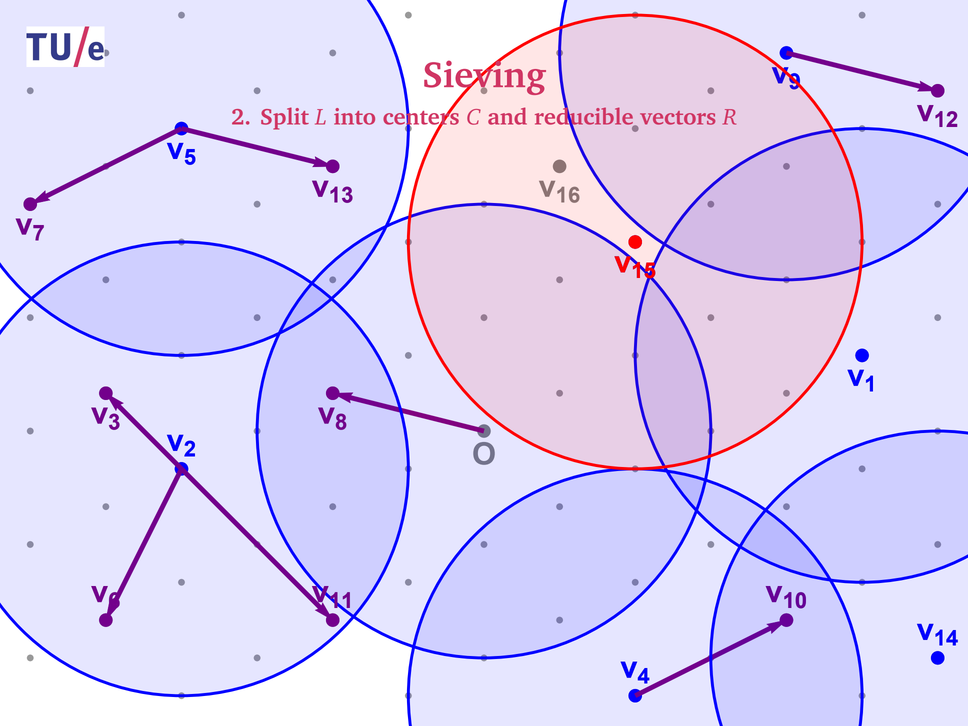
2. Split L into centers C and reducible vectors R



TU/e

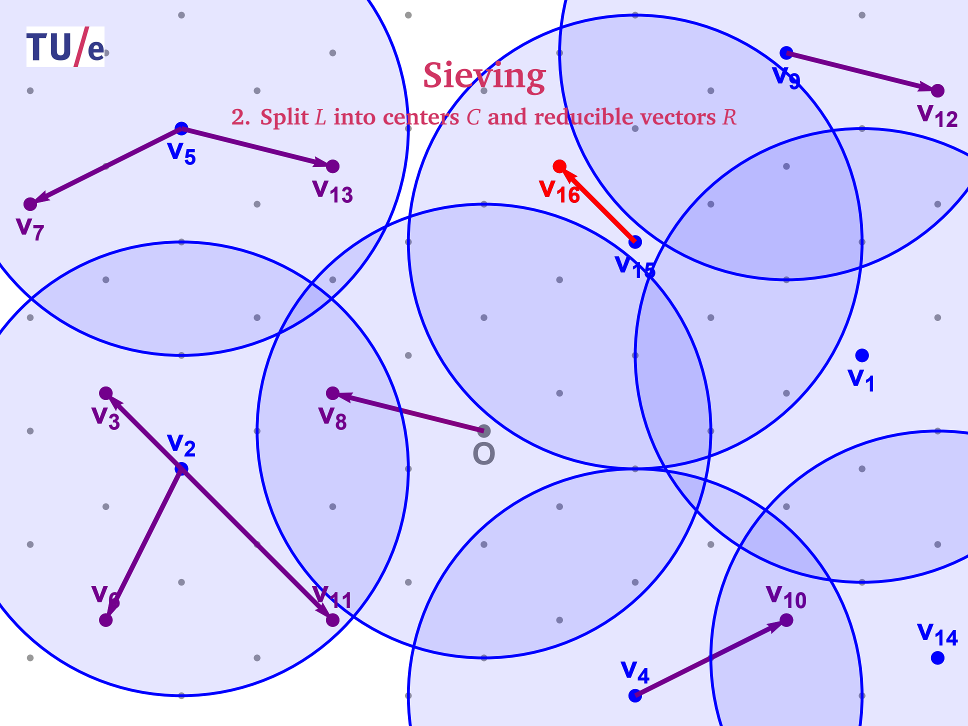
Sieving

2. Split L into centers C and reducible vectors R



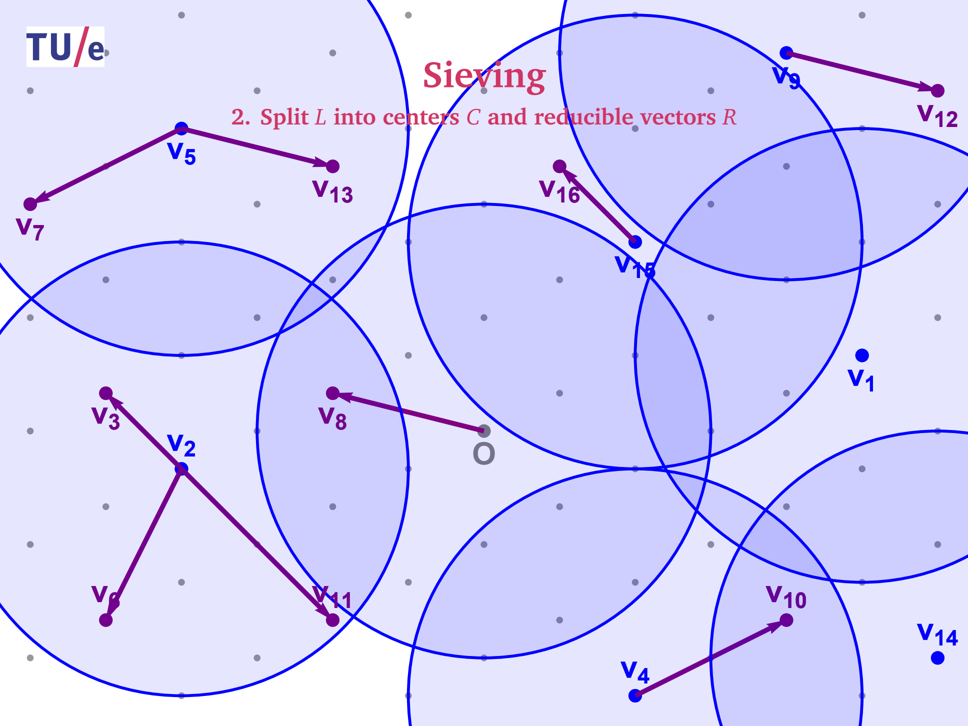
Sieving

2. Split L into centers C and reducible vectors R



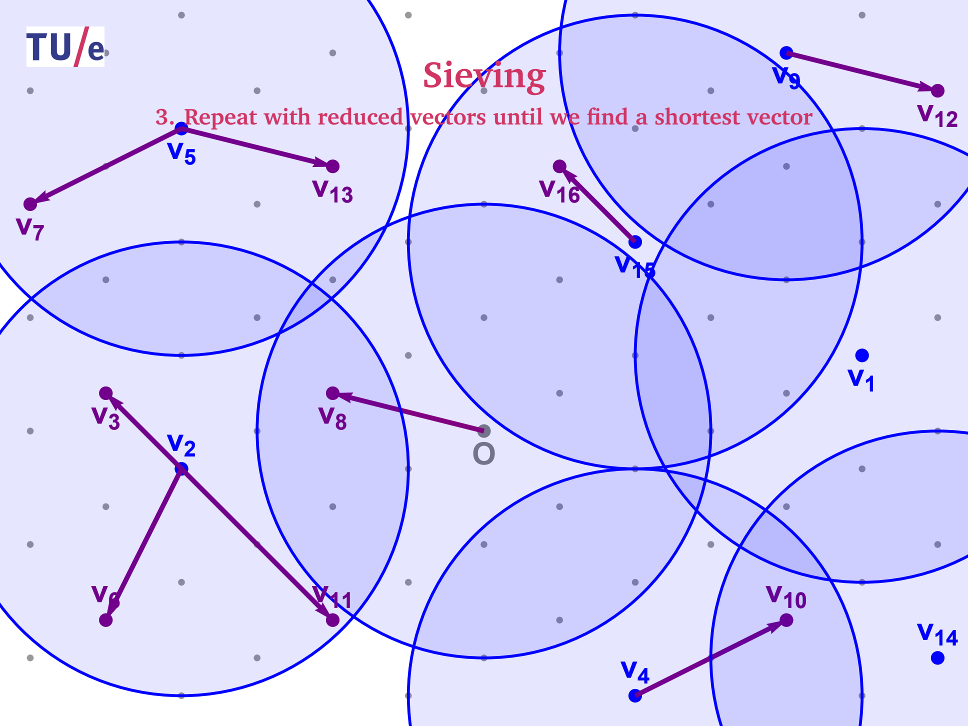
Sieving

2. Split L into centers C and reducible vectors R



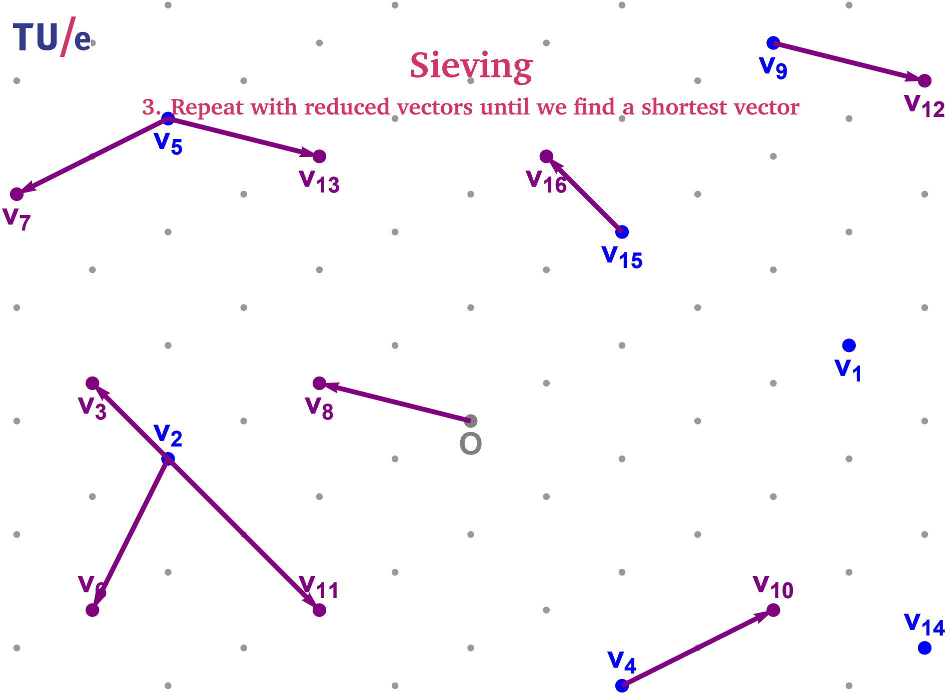
Sieving

3. Repeat with reduced vectors until we find a shortest vector



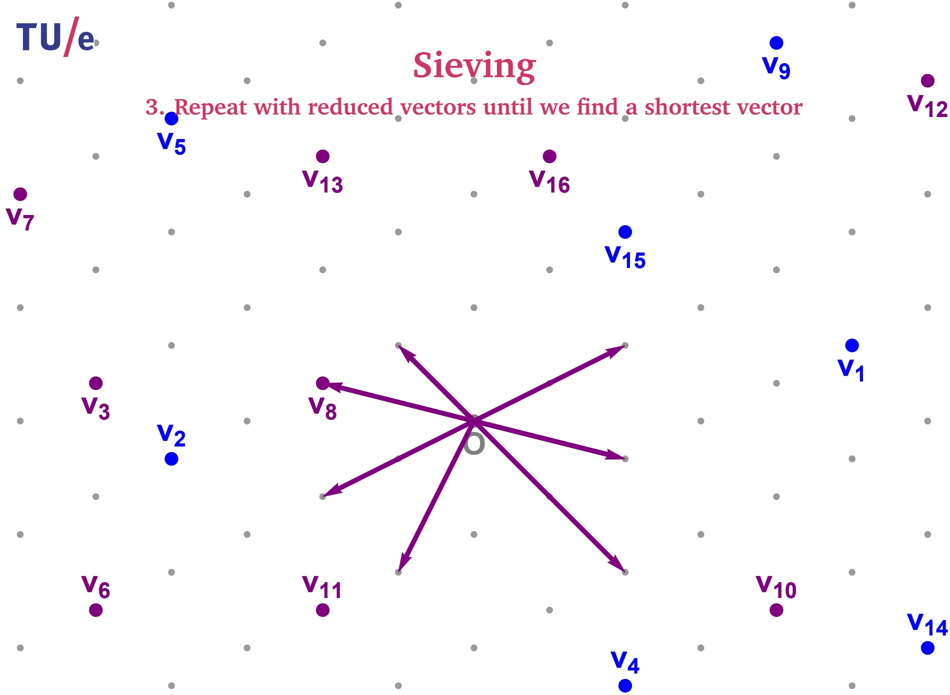
Sieving

3. Repeat with reduced vectors until we find a shortest vector



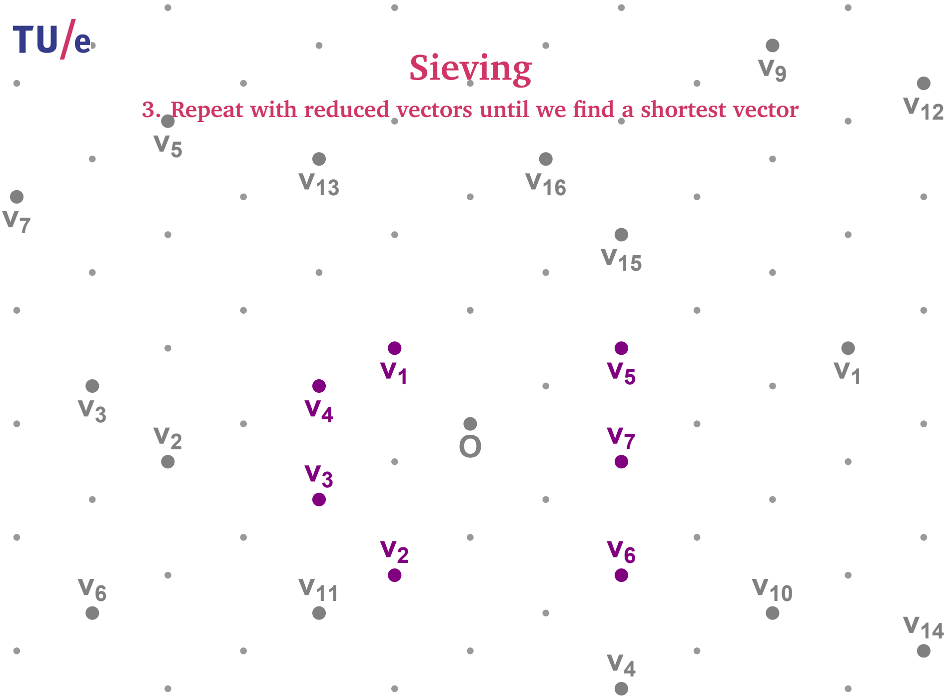
Sieving

3. Repeat with reduced vectors until we find a shortest vector



Sieving

3. Repeat with reduced vectors until we find a shortest vector



Sieving

Overview

Sieving

Overview

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

Sieving solves SVP in time $(4/3)^{n+o(n)}$ and space $(4/3)^{n/2+o(n)}$.

Sieving

Overview

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

Sieving solves SVP in time $(4/3)^{n+o(n)}$ and space $(4/3)^{n/2+o(n)}$.

The list size comes from heuristic packing/saturation arguments, the time complexity is quadratic in the list size.

Sieving

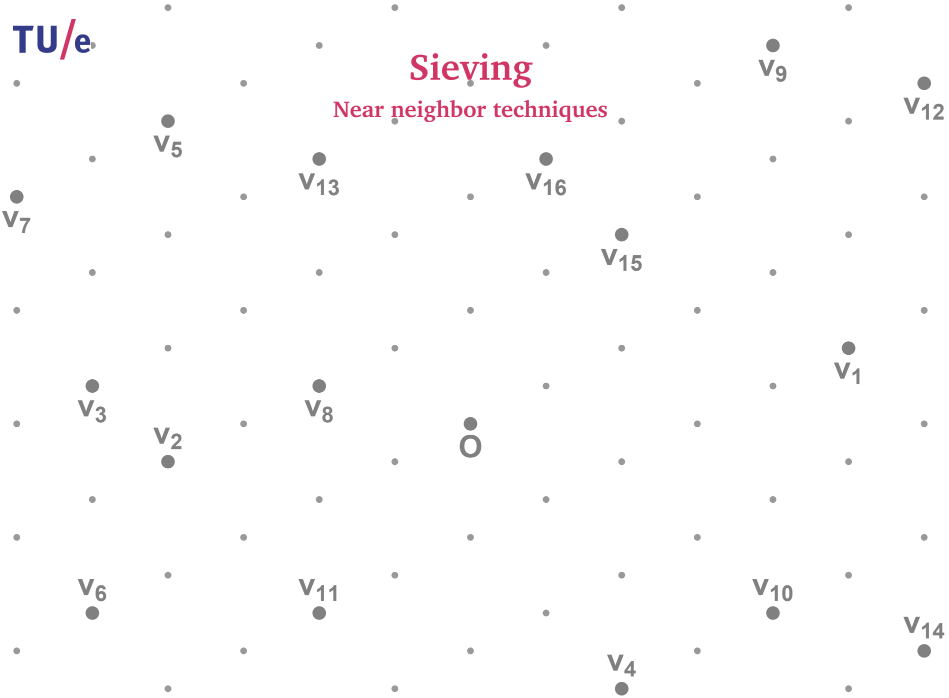
Near neighbor techniques

Sieving

Near neighbor techniques

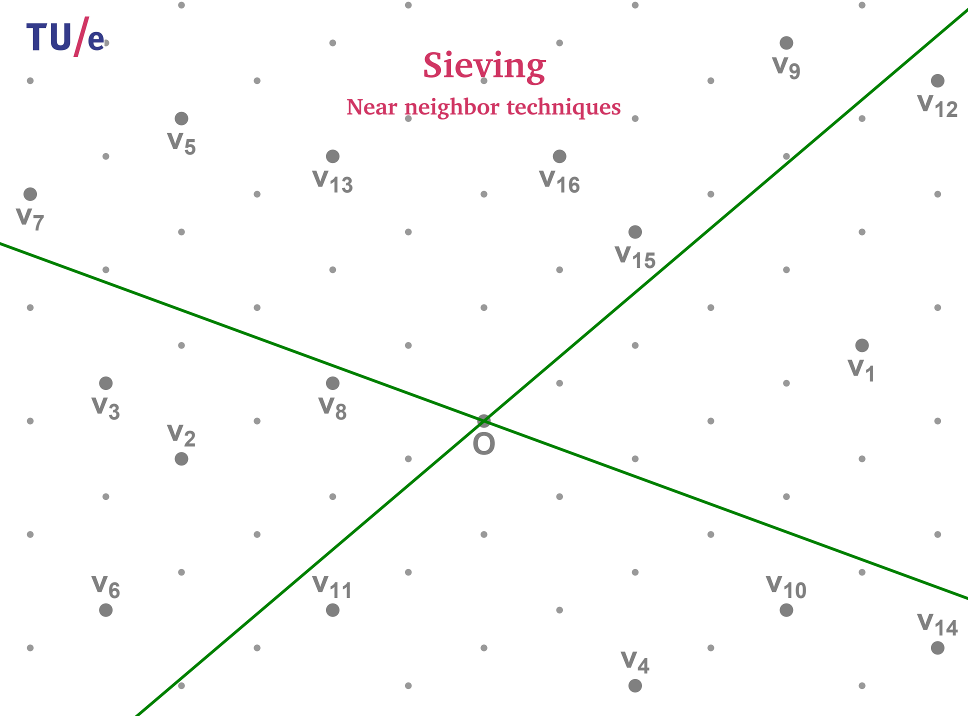
Sieving

Near neighbor techniques



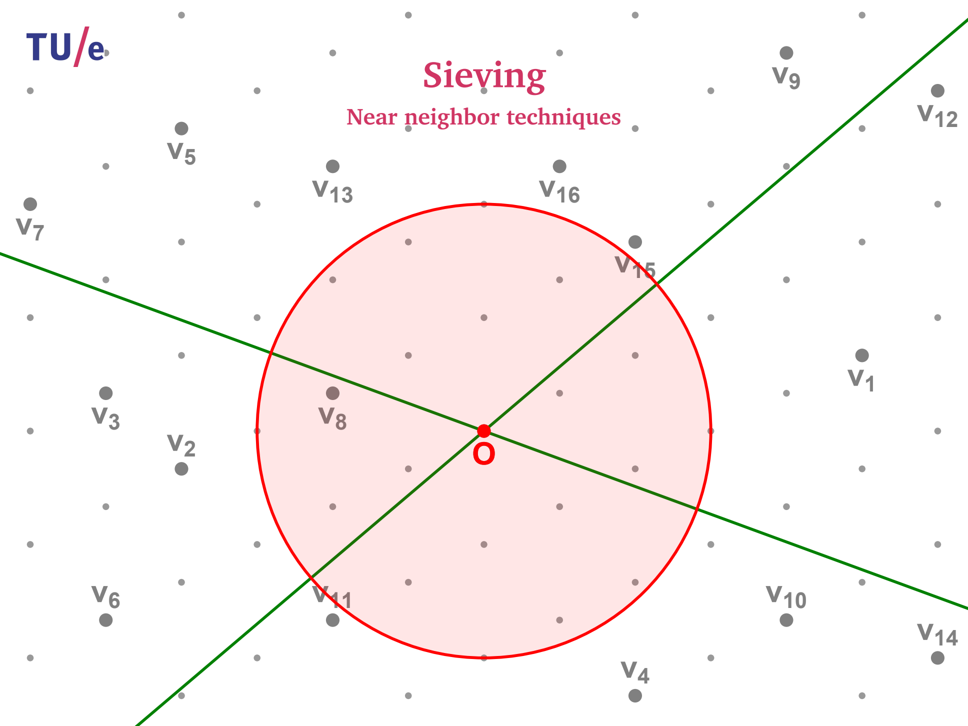
Sieving

Near neighbor techniques



Sieving

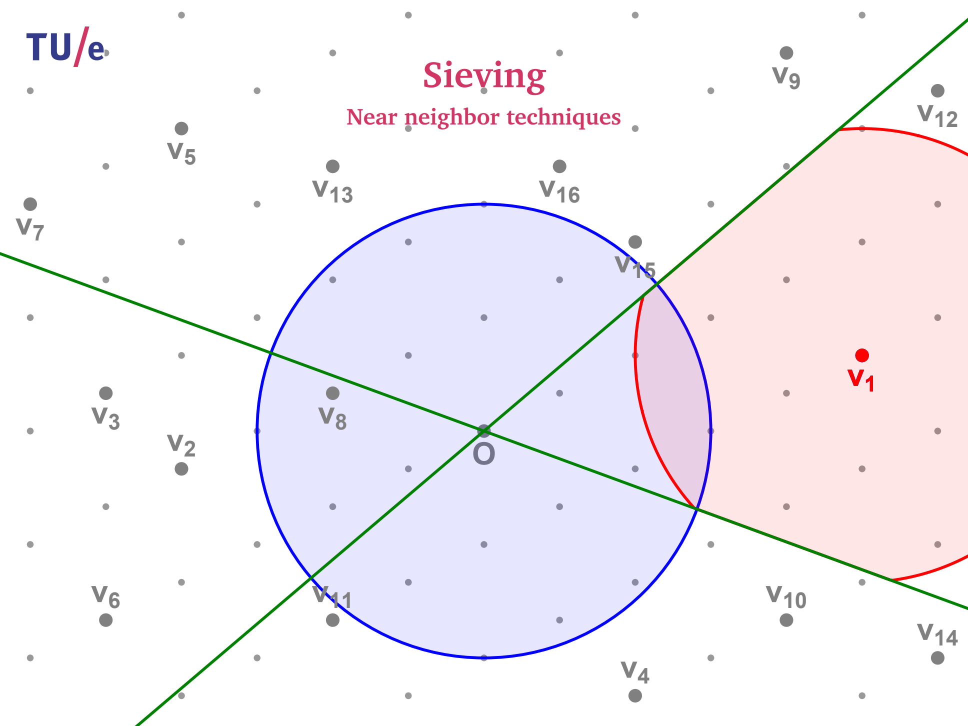
Near neighbor techniques



TU/e

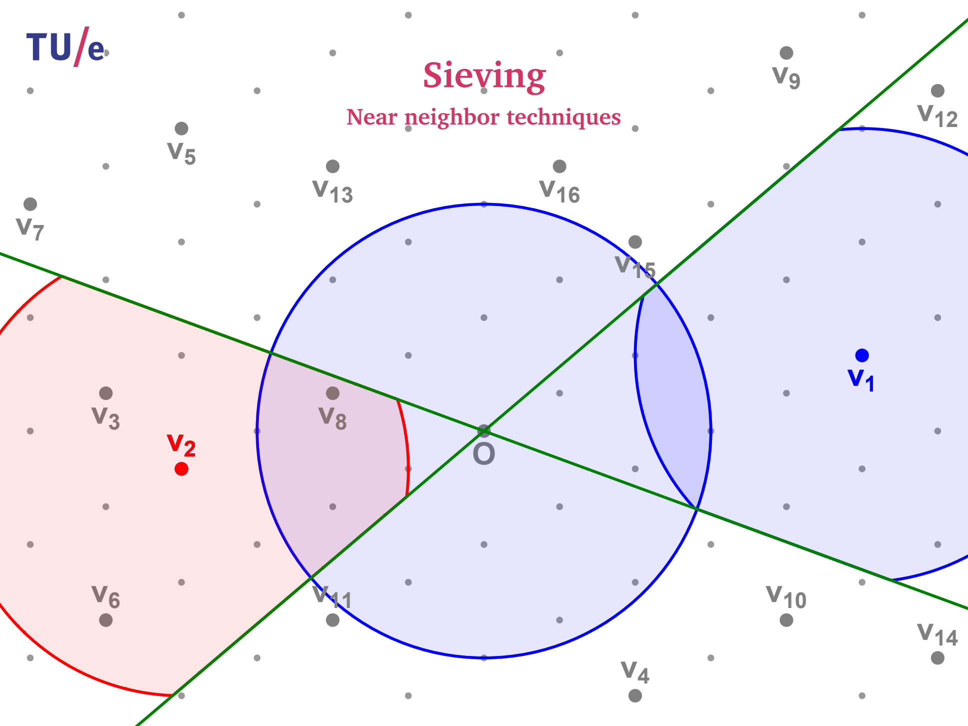
Sieving

Near neighbor techniques



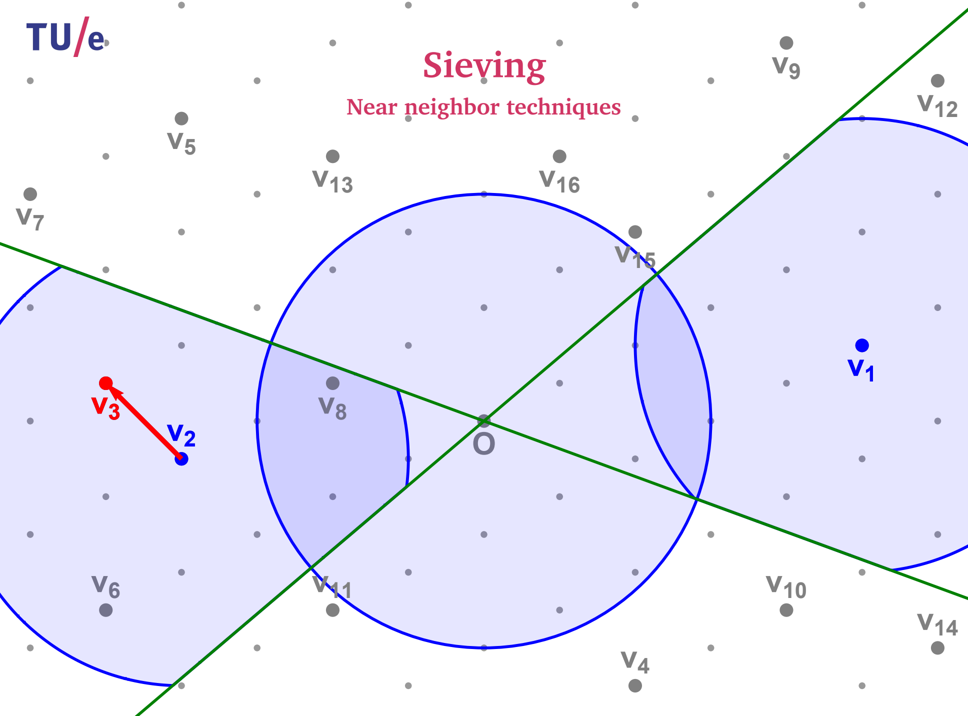
Sieving

Near neighbor techniques



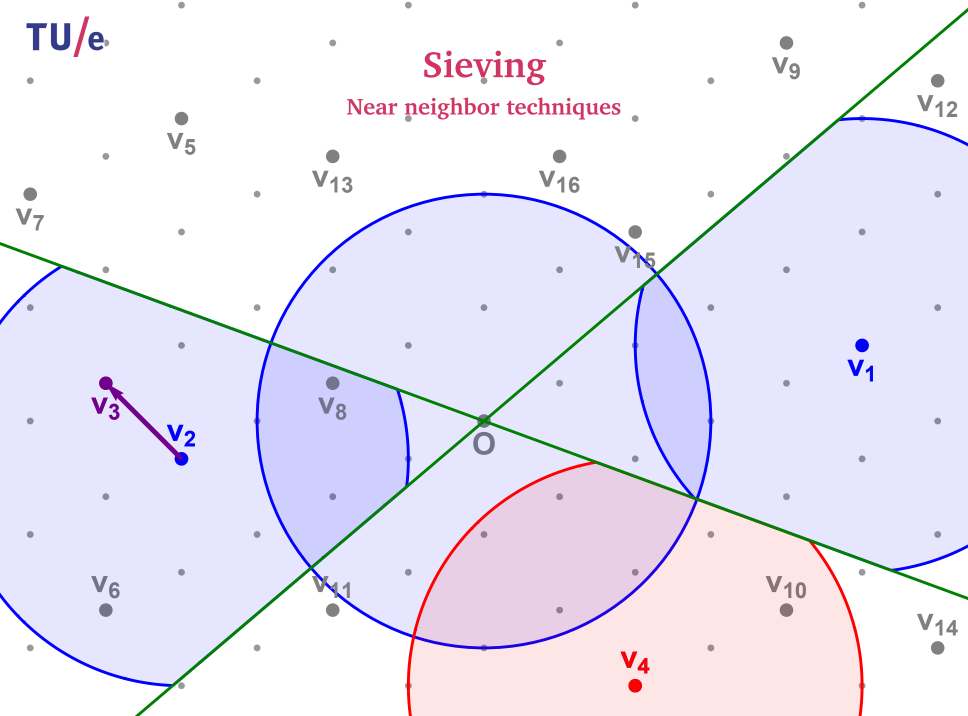
Sieving

Near neighbor techniques



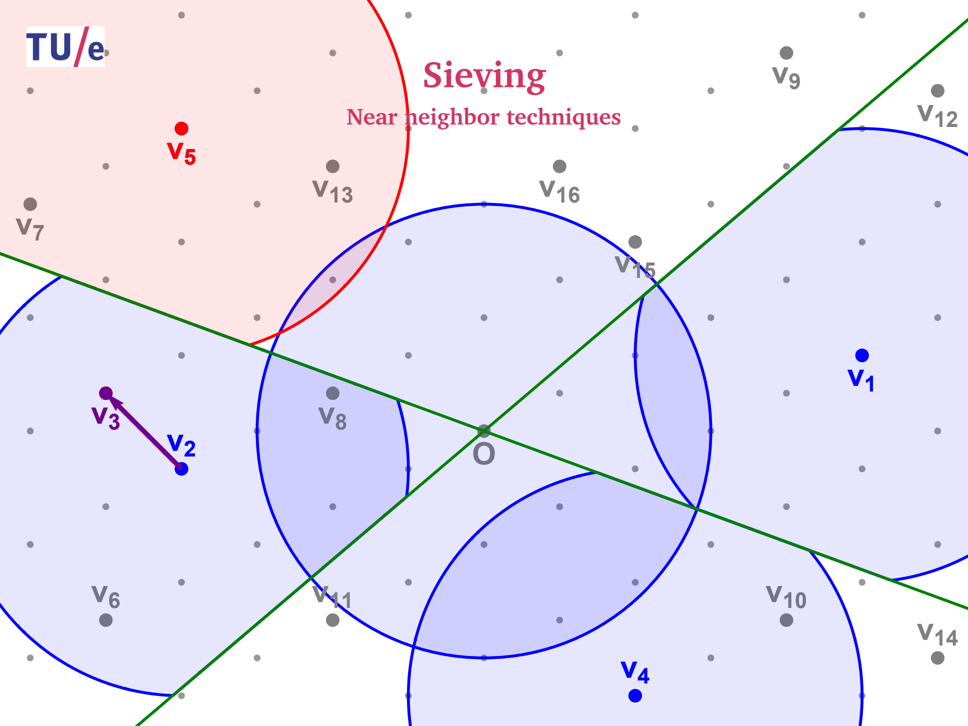
Sieving

Near neighbor techniques



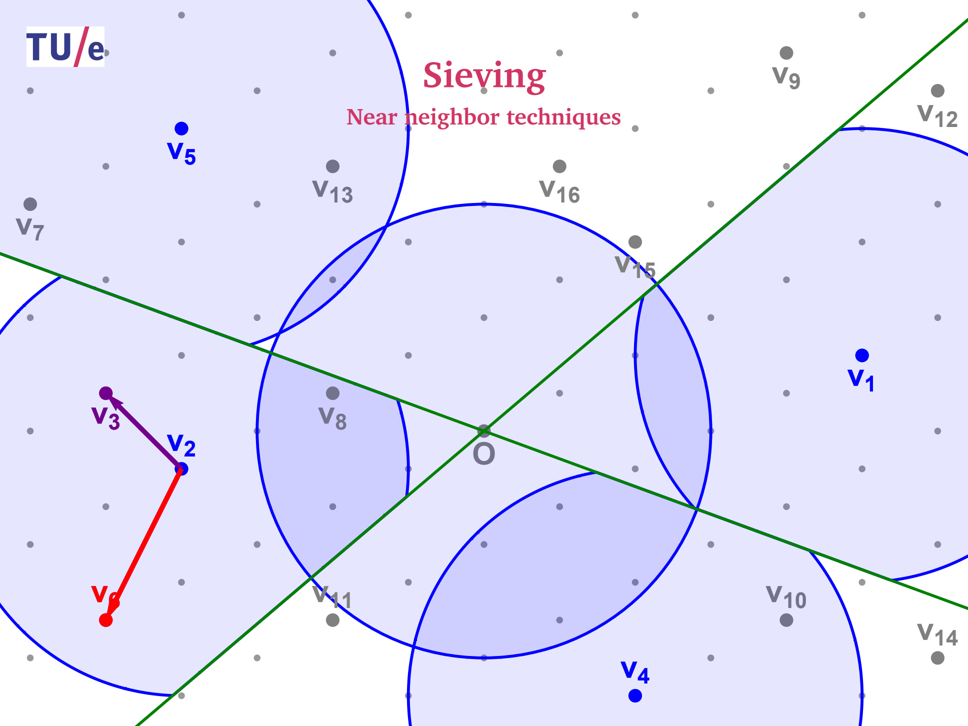
Sieving

Near neighbor techniques



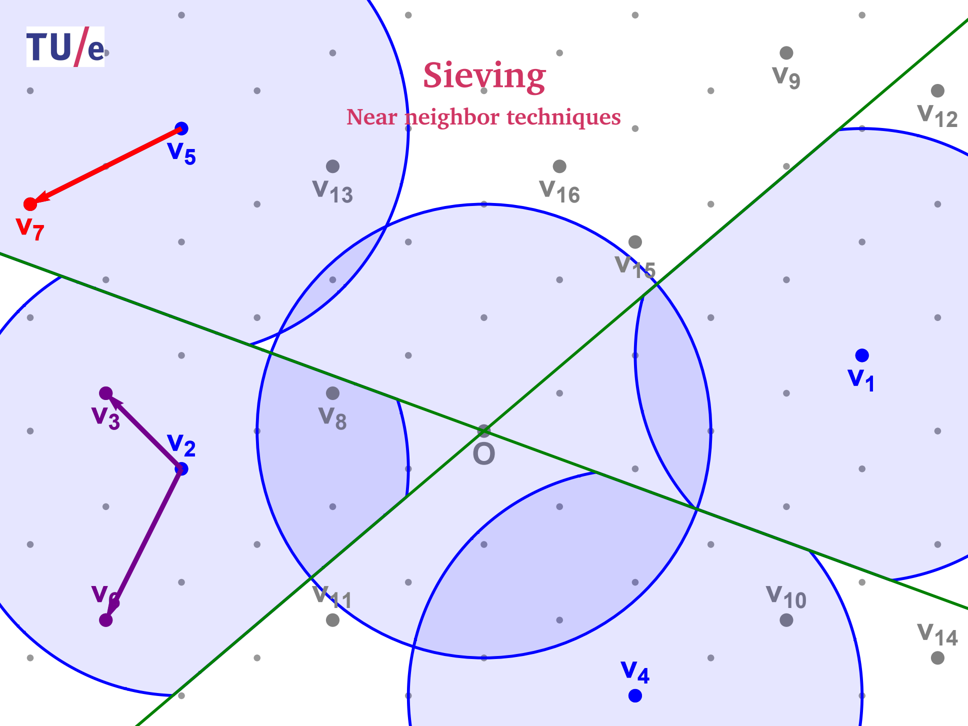
Sieving

Near neighbor techniques



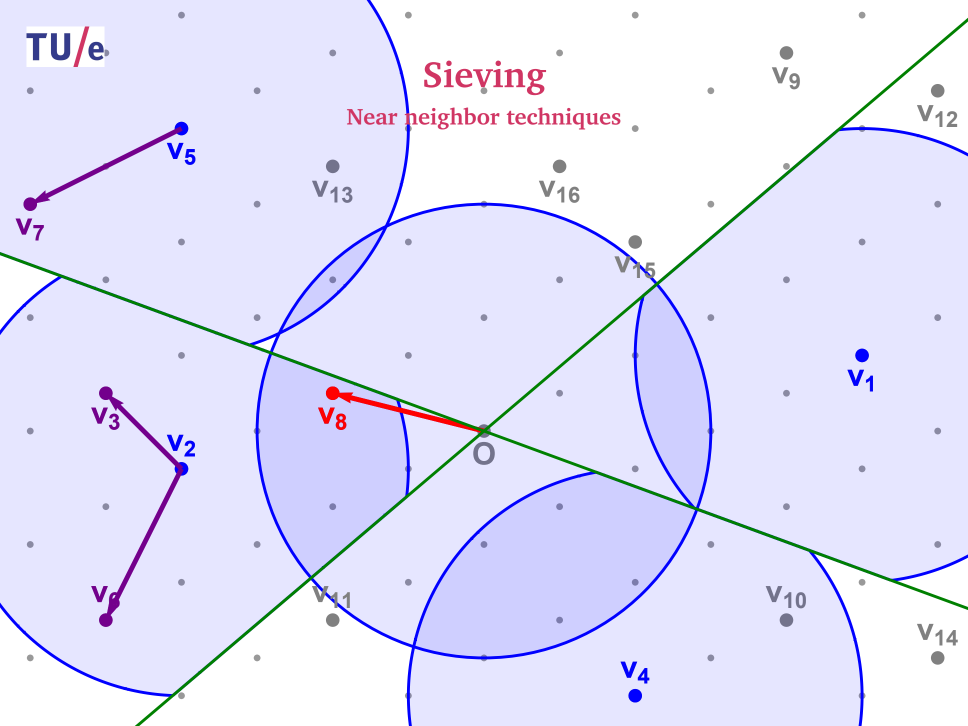
Sieving

Near neighbor techniques



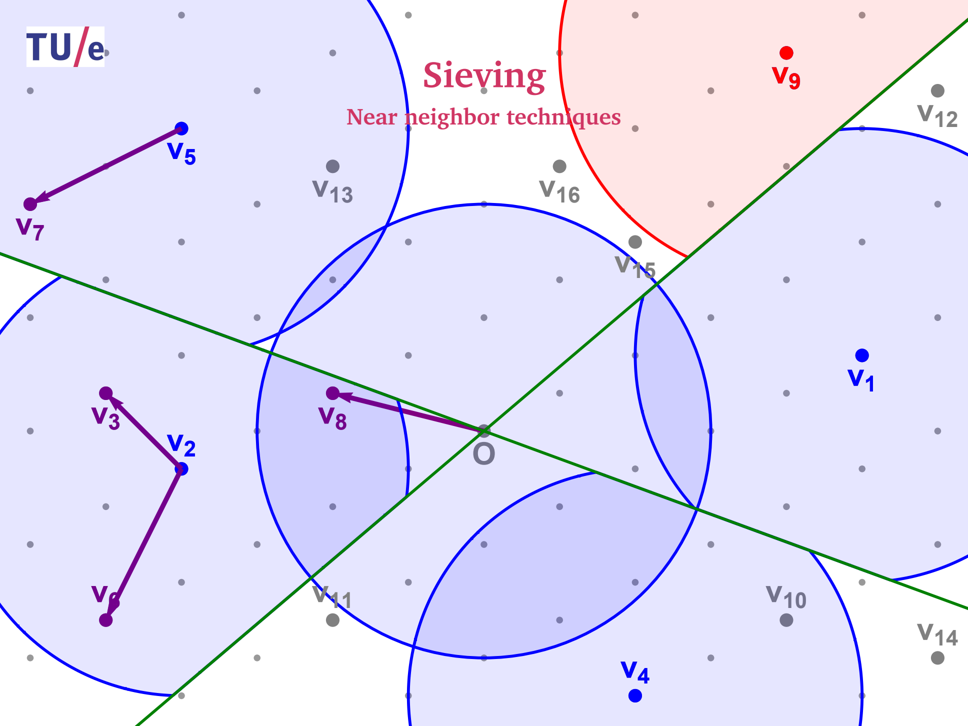
Sieving

Near neighbor techniques



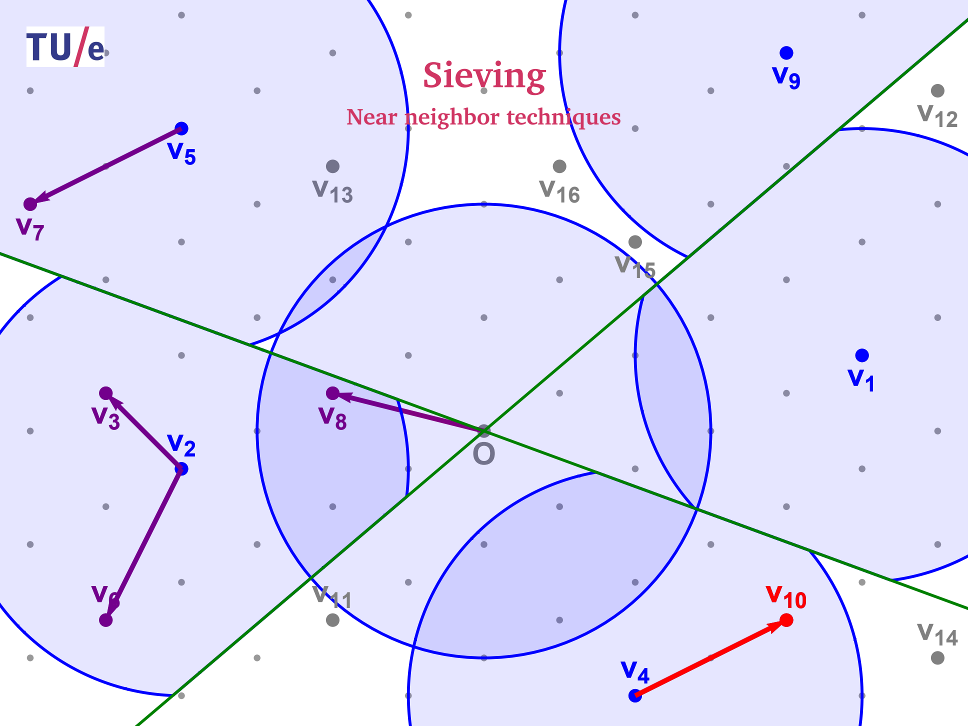
Sieving

Near neighbor techniques



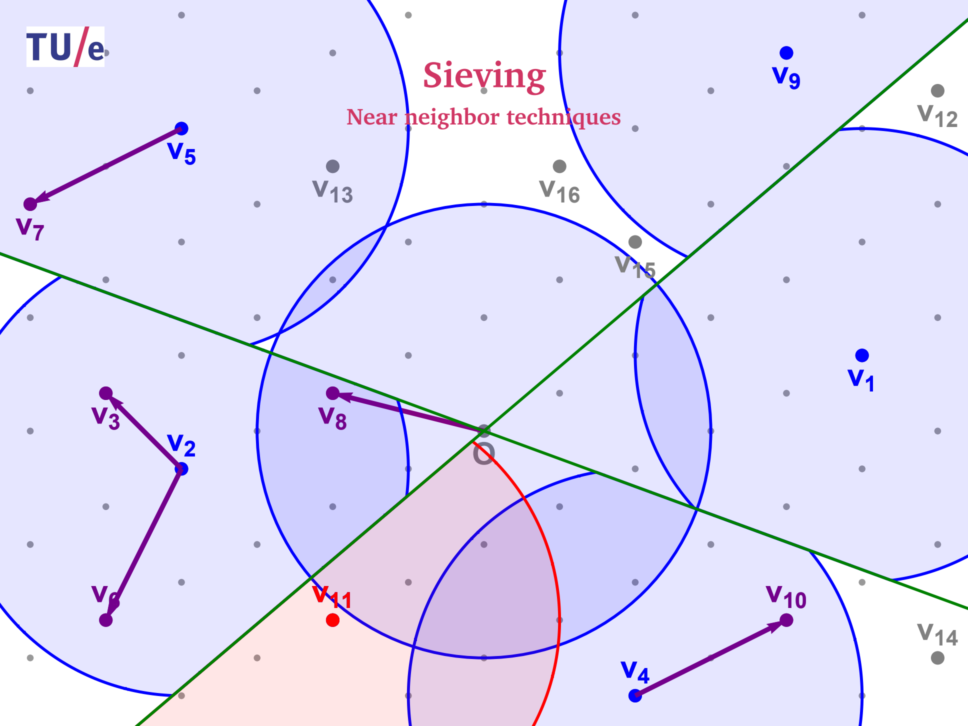
Sieving

Near neighbor techniques



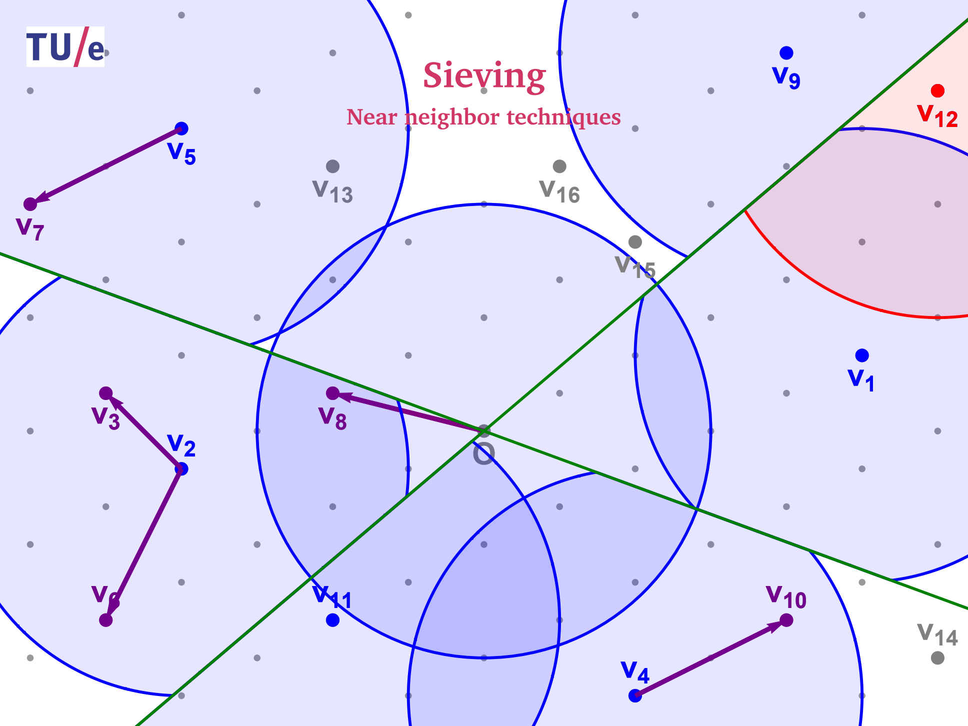
Sieving

Near neighbor techniques



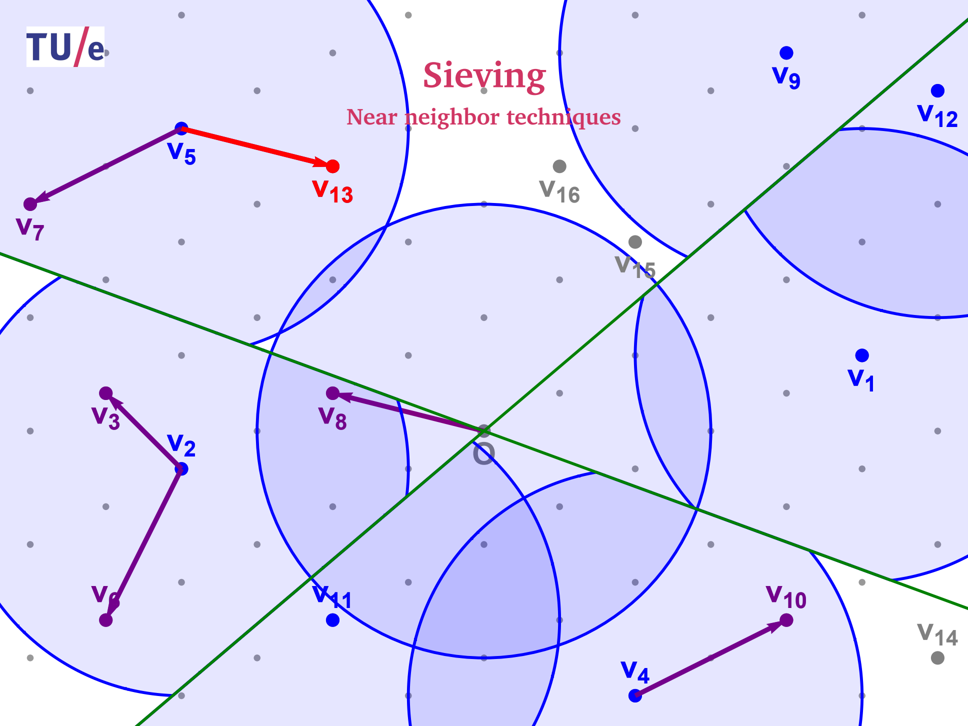
Sieving

Near neighbor techniques



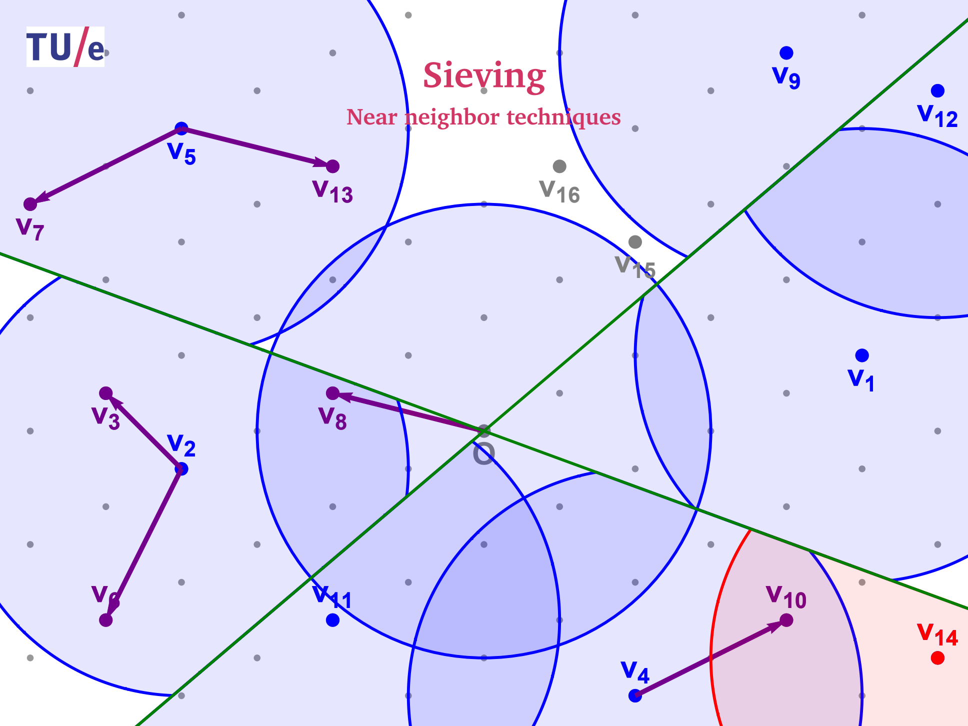
Sieving

Near neighbor techniques



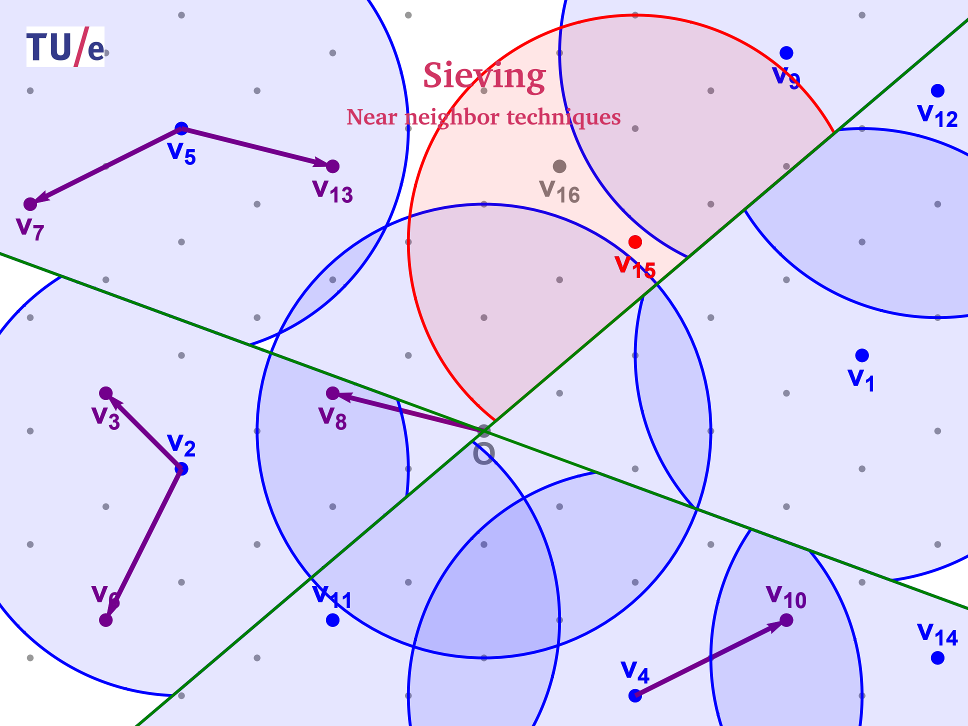
Sieving

Near neighbor techniques



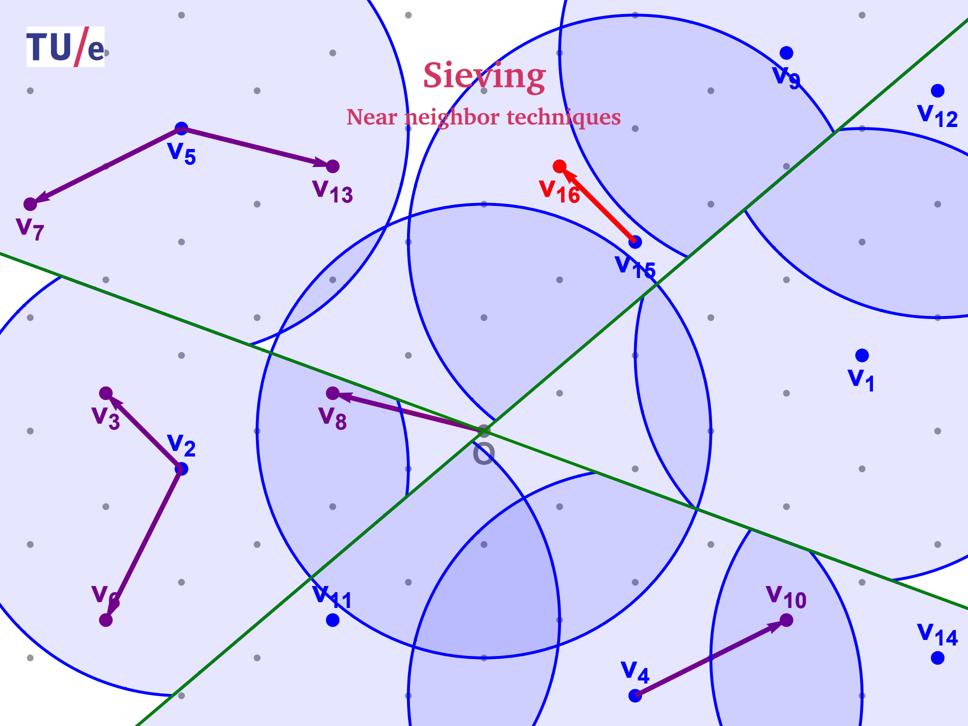
Sieving

Near neighbor techniques



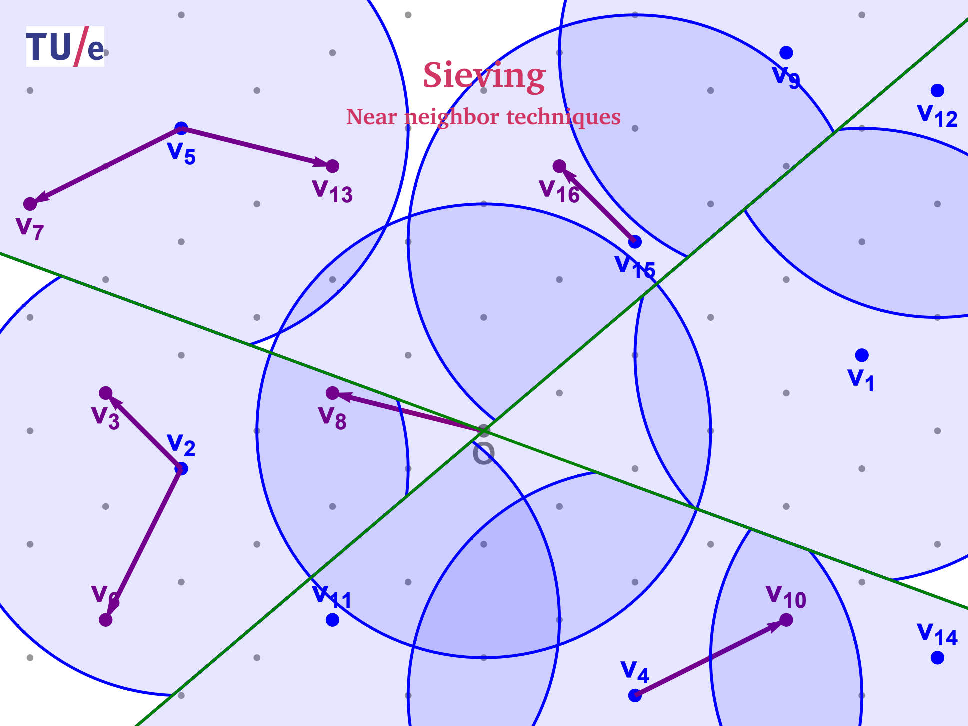
Sieving

Near neighbor techniques



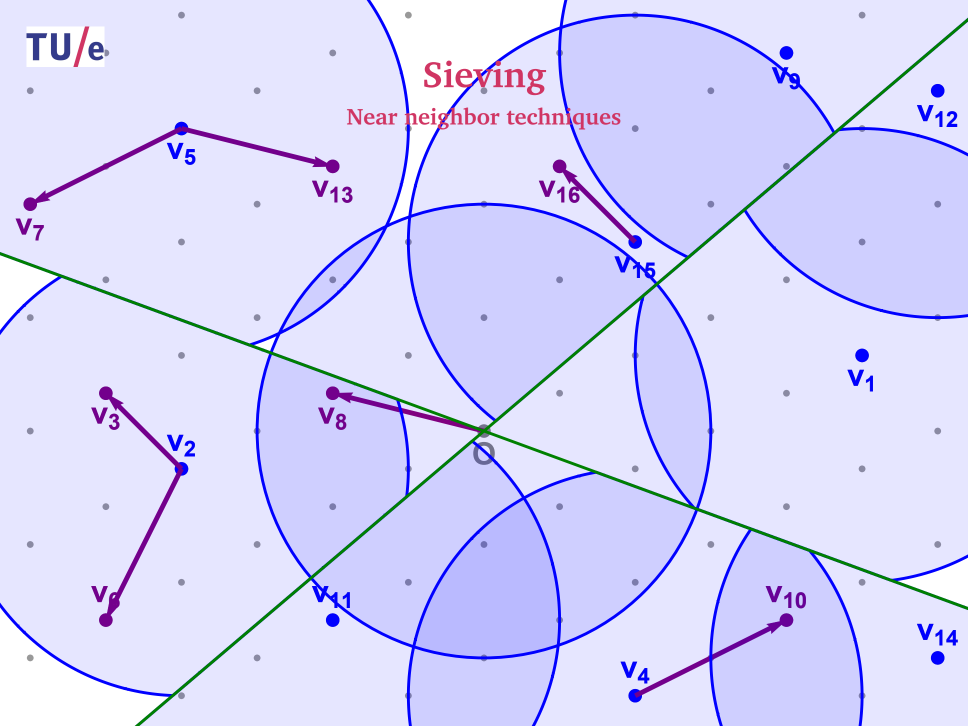
Sieving

Near neighbor techniques



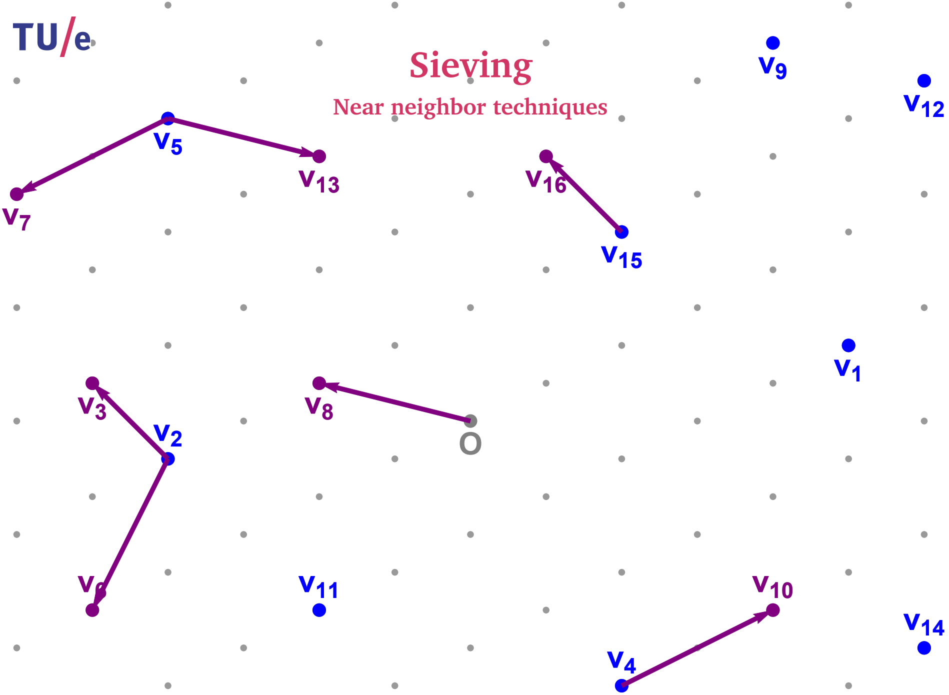
Sieving

Near neighbor techniques



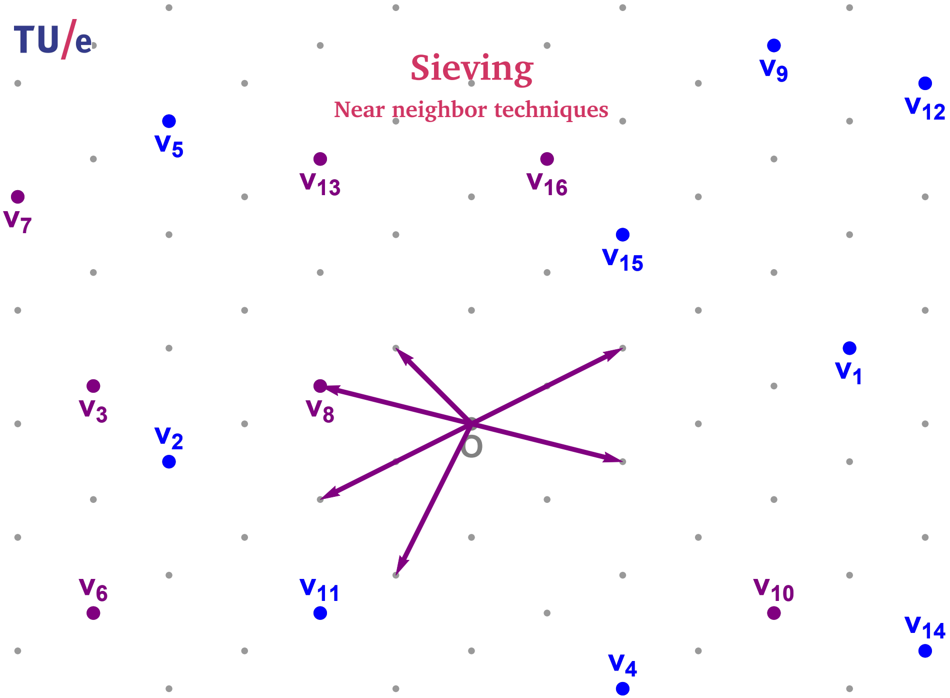
Sieving

Near neighbor techniques



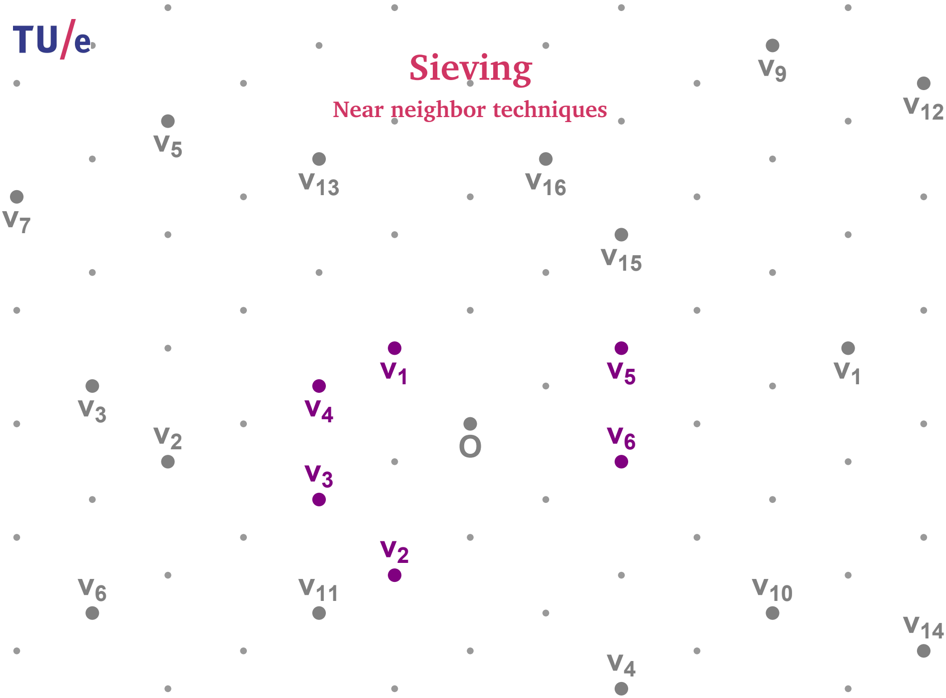
Sieving

Near neighbor techniques



Sieving

Near neighbor techniques

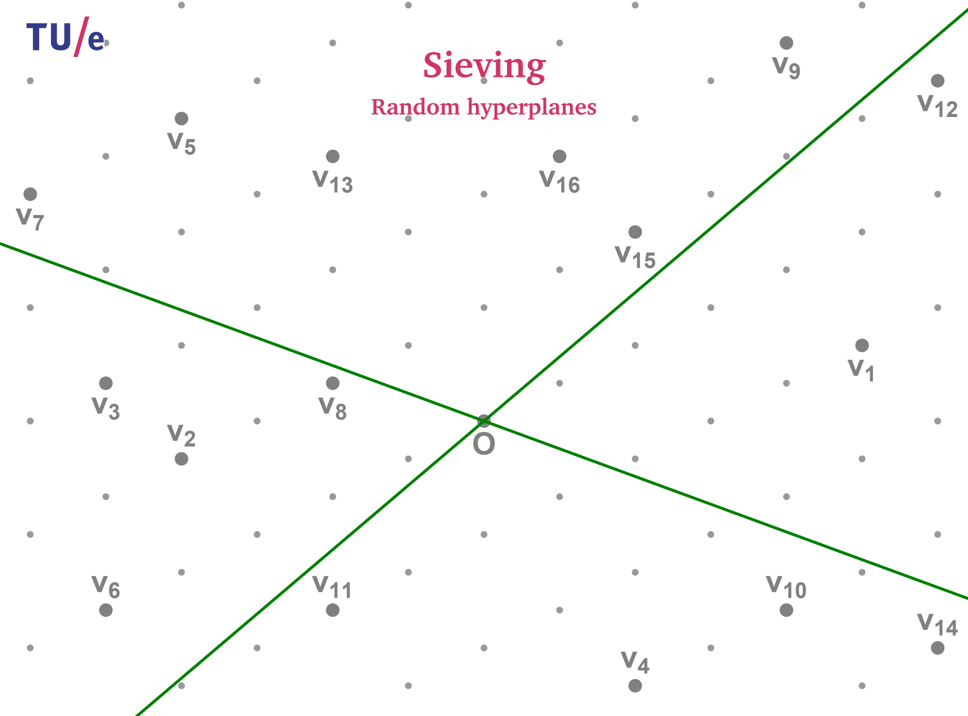


Sieving

Random hyperplanes

Sieving

Random hyperplanes



Sieving

Random hypercones

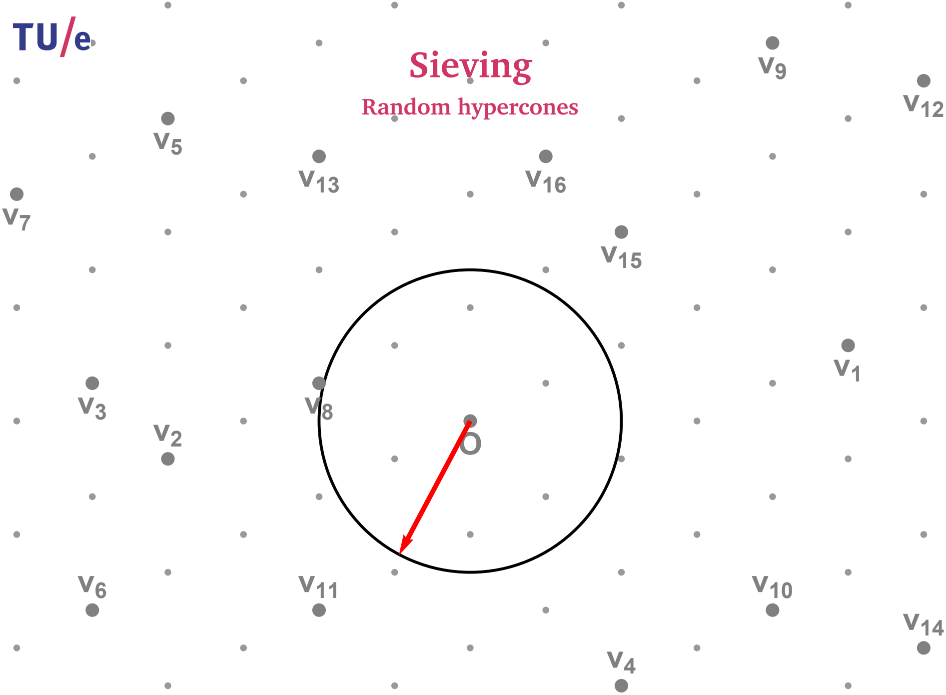
TU/e

Sieving

Random hypercones

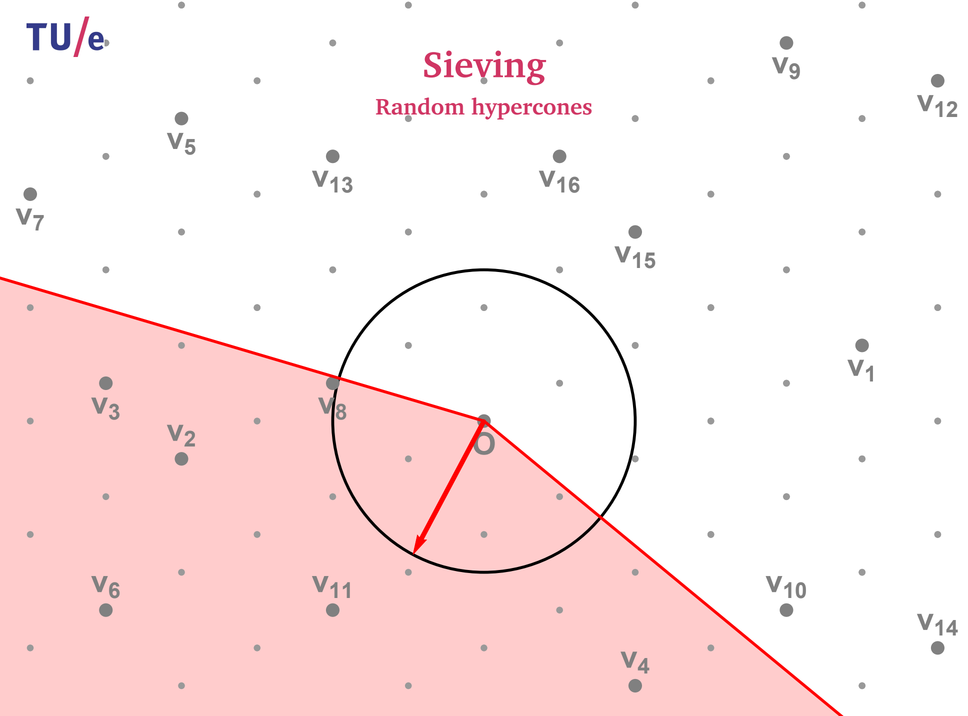
Sieving

Random hypercones



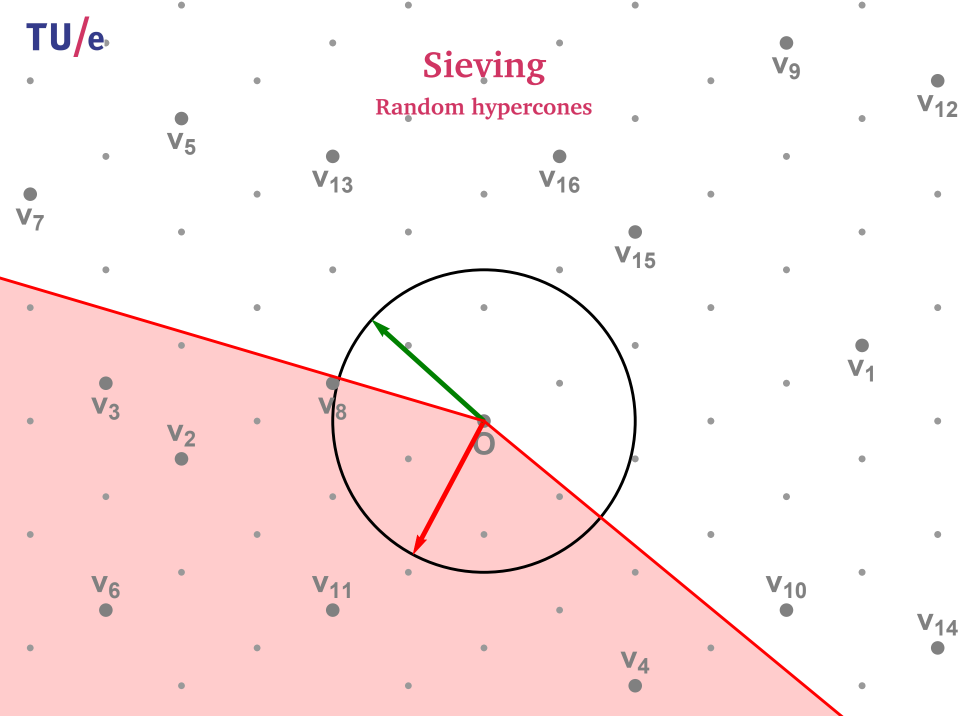
Sieving

Random hypercones



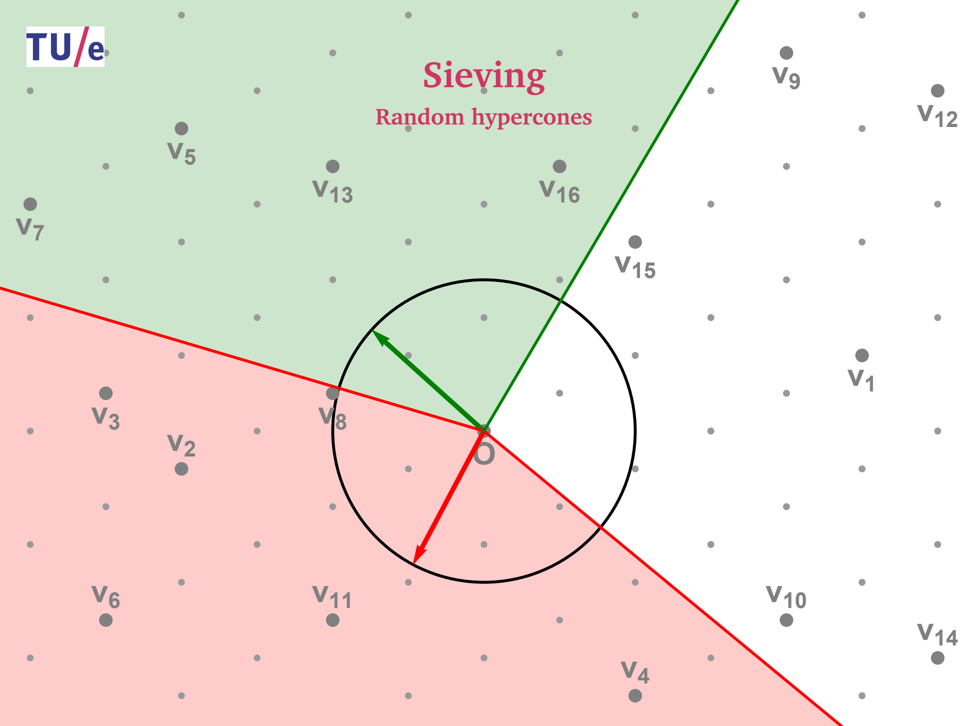
Sieving

Random hypercones



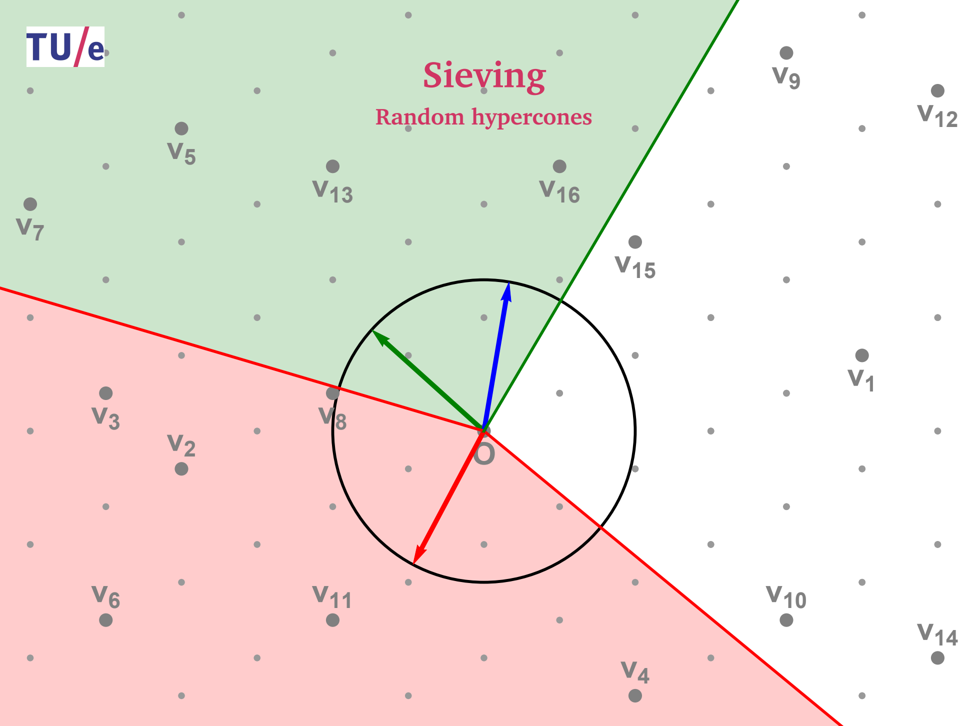
Sieving

Random hypercones



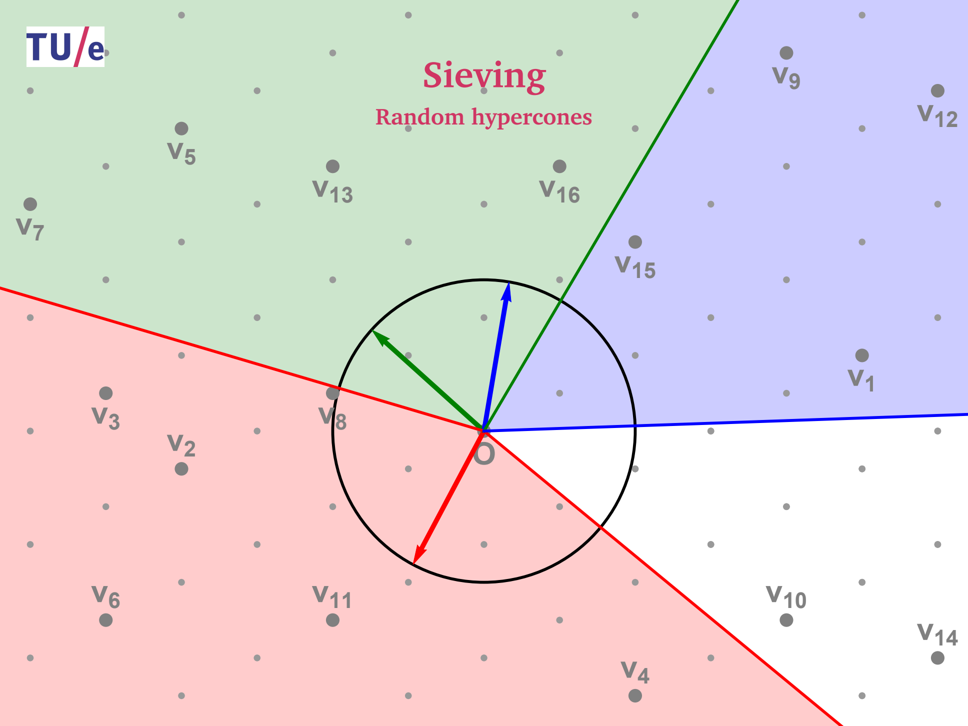
Sieving

Random hypercones



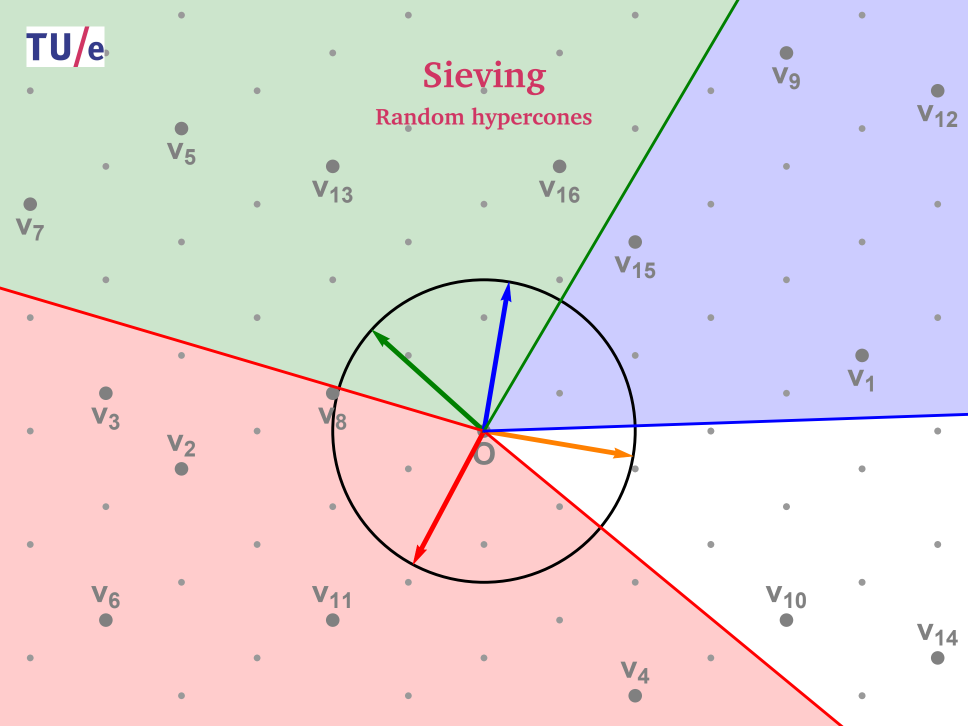
Sieving

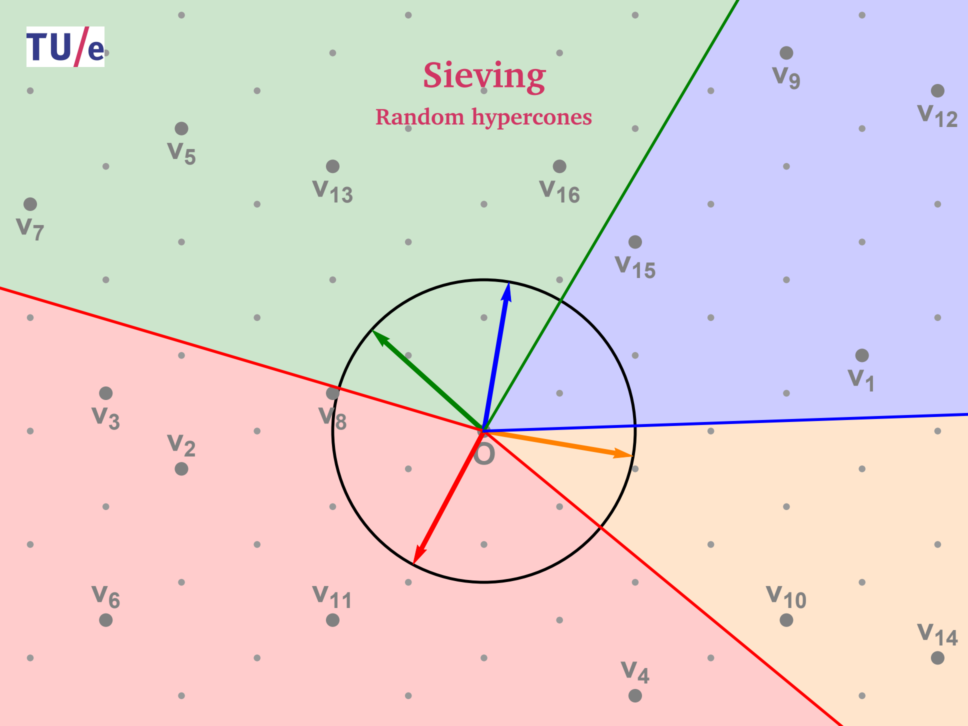
Random hypercones



Sieving

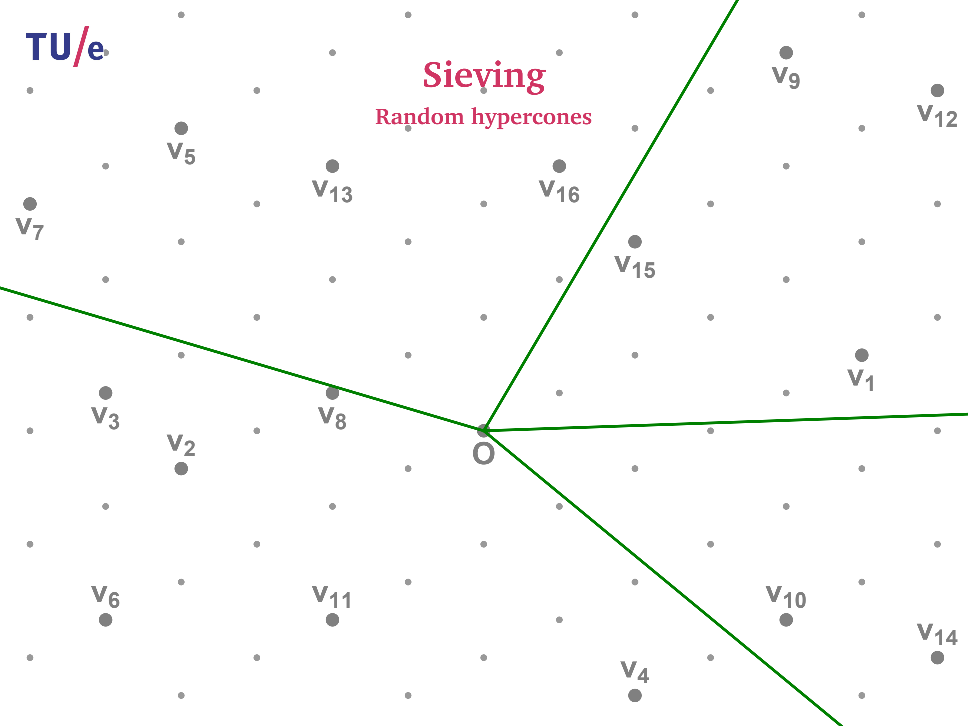
Random hypercones




Sieving
Random hypercones
v9
v12
v5
v13
v16
v7
v15
v1
v3
v8
v2
O
v6
v11
v10
v14
v4

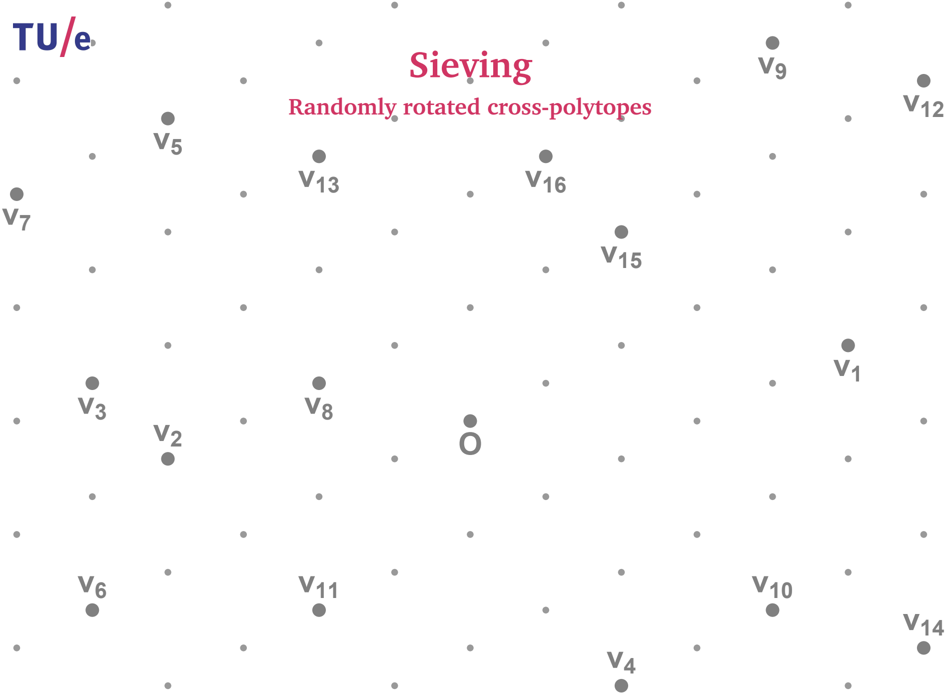
Sieving

Random hypercones



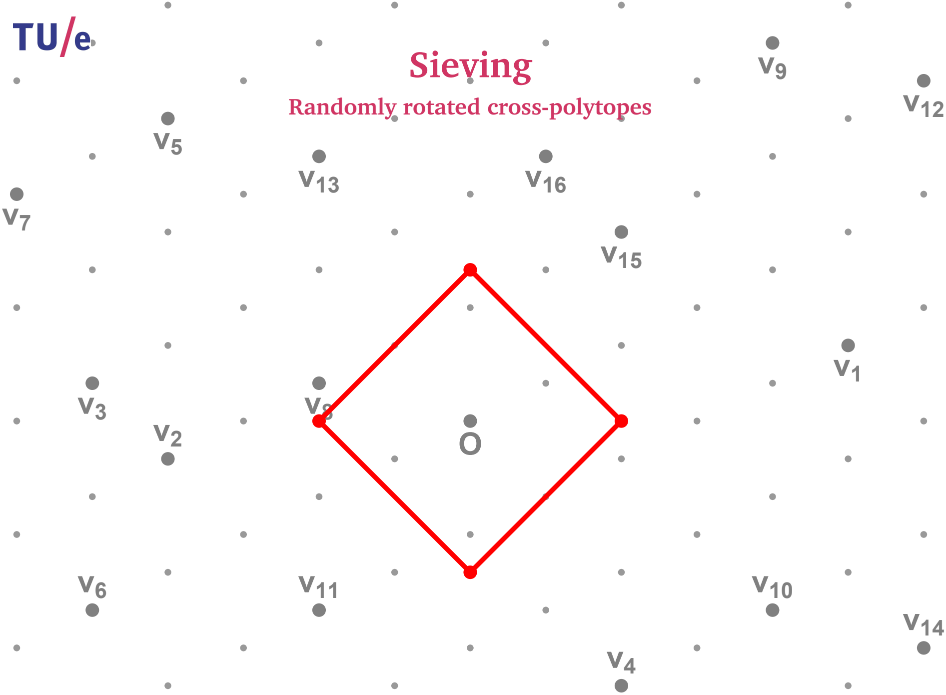
Sieving

Randomly rotated cross-polytopes



Sieving

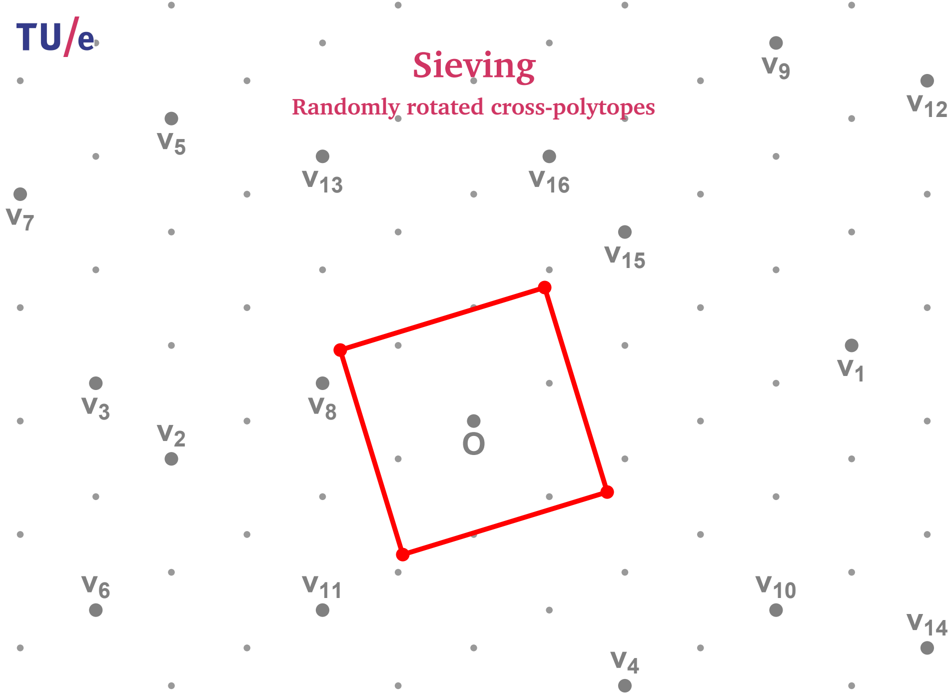
Randomly rotated cross-polytopes



TU/e

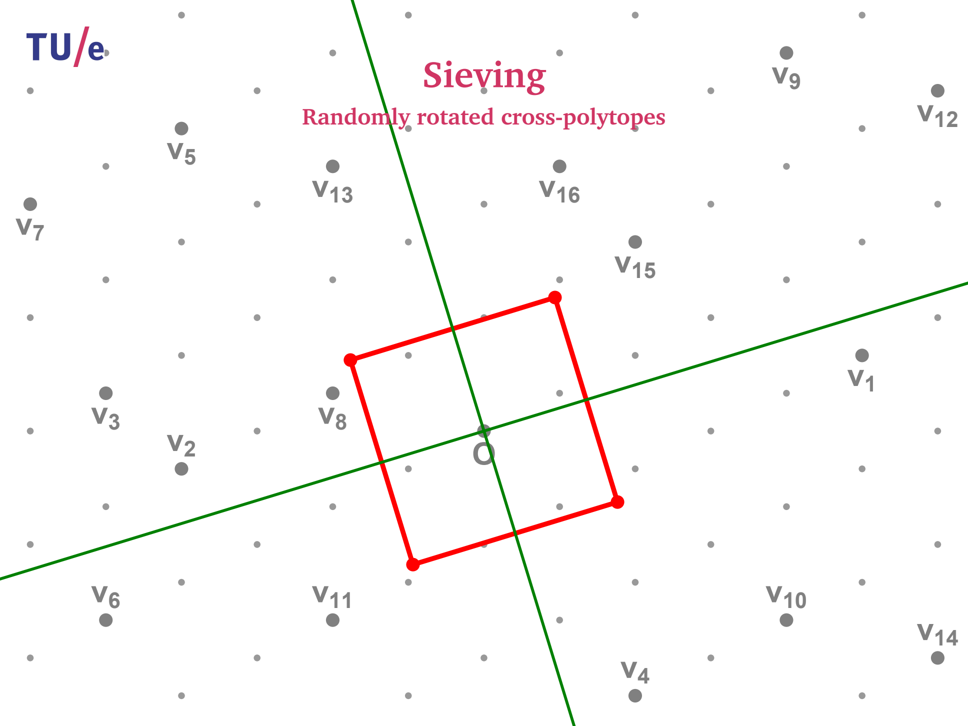
Sieving

Randomly rotated cross-polytopes



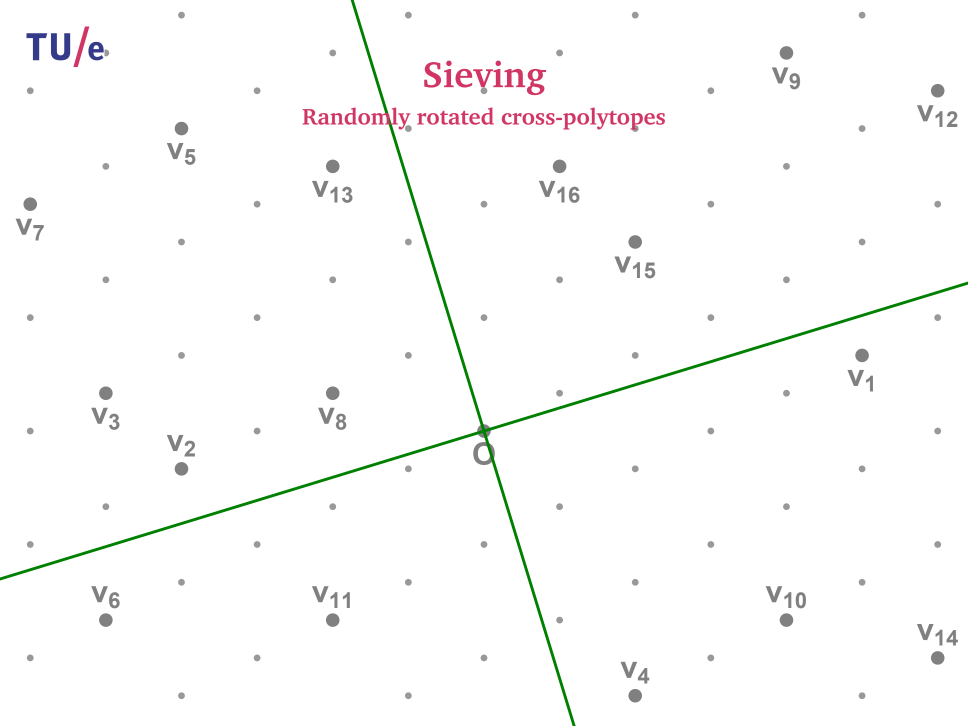
Sieving

Randomly rotated cross-polytopes



Sieving

Randomly rotated cross-polytopes



Outline

SVP algorithms

- Enumeration
- Sieving

SVP hardness

- Theory
- Practice
- NIST submissions

Conclusion

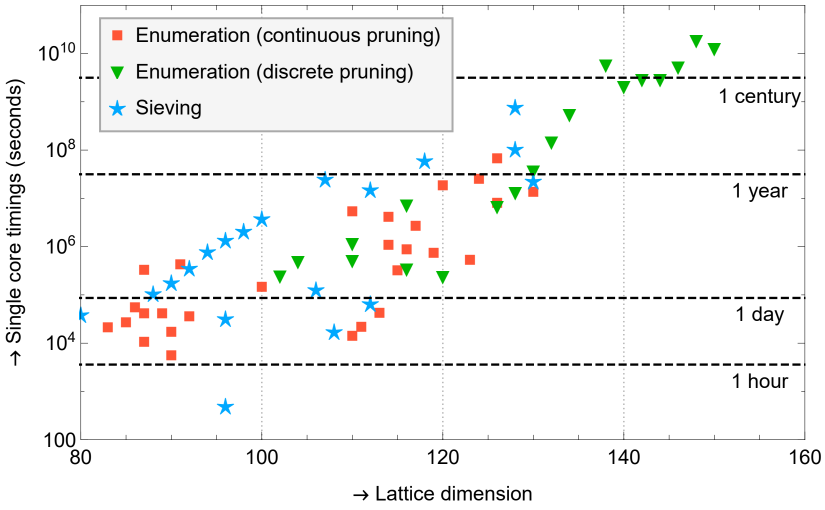
SVP hardness

Theory

	Algorithm	$\log_2$(Time)	$\log_2$(Space)
Proven SVP	Enumeration [Poh81, Kan83, ..., MW15, AN17]	$O(n \log n)$	$O(\log n)$
	AKS-sieve [AKS01, NV08, MV10, HPS11]	$3.398n$	$1.985n$
	ListSieve [MV10, MDB14]	$3.199n$	$1.327n$
	Birthday sieves [PS09, HPS11]	$2.465n$	$1.233n$
	Enumeration/DGS hybrid [CCL17]	$2.048n$	$0.500n$
	Voronoi cell algorithm [AEVZ02, MV10b]	$2.000n$	$1.000n$
	Quantum sieve [LMP13, LMP15]	$1.799n$	$1.286n$
	Quantum enum/DGS [CCL17]	$1.256n$	**0.500n**
	Discrete Gaussian sampling [ADRS15, ADS15, AS18]	**1.000n**	$1.000n$
Sieving	The Nguyen–Vidick sieve [NV08]	$0.415n$	$0.208n$
	GaussSieve [MV10, ..., IKMT14, BNvdP16, YKYC17]	$0.415n$	$0.208n$
	Triple sieve [BLS16, HK17]	$0.396n$	$0.189n$
	Leveled sieving [WLTB11, ZPH13]	$0.3778n$	$0.283n$
	Overlattice sieve [BGJ14]	$0.3774n$	$0.293n$
	Quantum sieve [LMP13]	$0.312n$	$0.208n$
Sieving + NNS	Triple sieve with NNS [HK17, HKL18]	$0.359n$	**0.189n**
	Hyperplane LSH [Cha02, Laa15, ..., LM18, Duc18]	$0.337n$	$0.337n$
	Graph-based NNS [EPY99, DCL11, MPLK14, Laa18]	$0.327n$	$0.282n$
	Hypercube LSH [TT07, Laa17]	$0.322n$	$0.322n$
	May–Ozerov NNS [MO15, BGJ15]	$0.311n$	$0.311n$
	Spherical LSH [AINR14, LdW15]	$0.298n$	$0.298n$
	Cross-polytope LSH [TT07, AILRS15, BL16, KW17]	$0.298n$	$0.298n$
	Spherical LSF [BDGL16, MLB17, ALRW17, Chr17]	**0.293n**	$0.293n$
	Quantum NNS sieve [LMP15, Laa16]	**0.265n**	$0.265n$

SVP hardness

Practice



SVP hardness

NIST submissions

Title	S	E	O	Submitters
CRYSTALS–Dilithium	•			**Lyubashevsky**, Ducas, Kiltz, Lepoint, Schwabe, Seiler, Stehlé
CRYSTALS–Kyber	•			**Schwabe**, Avanzi, Bos, Ducas, Kiltz, Lepoint, Lyubashevsky, Schanck, ...
Ding Key Exchange	•			**Ding**, Takagi, Gao, Wang
DRS			•	**Plantard**, Sipasseuth, Dumondelle, Susilo
(R.)EMBLEM	•			**Seo**, Park, Lee, Kim, Lee
FALCON	•			**Prest**, Fouque, Hoffstein, Kirchner, Lyubashevsky, Pornin, Ricosset, ...
FrodoKEM	•			**Naehrig**, Alkim, Bos, Ducas, Easterbrook, LaMacchia, Longa, Mironov, ...
Giophantus	•			**Akiyama**, Goto, Okumura, Takagi, Nuida, Hanaoka, Shimizu, Ikematsu
HILA5	•			**Saarinen**
KCL	•			**Zhao**, Jin, Gong, Sui
KINDI	•			**El Bansarkhani**
LAC	•			**Lu**, Liu, Jia, Xue, He, Zhang
LIMA	•			**Smart**, Albrecht, Lindell, Orsini, Osheter, Paterson, Peer
Lizard	•			**Cheon**, Park, Lee, Kim, Song, Hong, Kim, Kim, Hong, Yun, Kim, Park, ...
LOTUS		•		**Phong**, Hayashi, Aono, Moriai
NewHope	•			**Pöppelmann**, Alkim, Avanzi, Bos, Ducas, De La Piedra, Schwabe, Stebila
NTRUEncrypt	◦	◦		**Zhang**, Chen, Hoffstein, Whyte
NTRU-HRSS-KEM	•			**Schanck**, Hülsing, Rijneveld, Schwabe
NTRU Prime		•		**Bernstein**, Chuengsatiansup, Lange, Van Vredendaal
Odd Manhattan			•	**Plantard**
pqNTRUSign	◦	◦		**Zhang**, Chen, Hoffstein, Whyte
qTESLA	•			**Bindel**, Akleylek, Alkim, Barreto, Buchmann, Eaton, Gutoski, Krämer, ...
Round2	•			**Garcia-Morchon**, Zhang, Bhattacharya, Rietman, Tolhuizen, Torre-Arce
SABER	•			**D'Anvers**, Karmakar, Roy, Vercauteren
Three Bears	•			**Hamburg**
Titanium	•			**Steinfeld**, Sakzad, Zhao
Totals:	**21**	**3**	**2**	**Total: 26 proposals with SVP hardness estimates**

*Not included in the overview: Compact LWE, Mersenne, Ramstake, ...

SVP hardness

NIST submissions

Most common hardness estimates:

- Ignore space complexity, ignore $o(n)$ in time complexity
- Classical sieving: $2^{0.292n}$ time
- Quantum sieving: $2^{0.265n}$ time
- "Paranoid bound": $2^{0.208n}$ time
  - Note: $2^{0.208n}$ is not a "lower bound" (Frodo, ...)
- Complexity of BKZ(β) $\geq$ Complexity of SVP(β)

Conclusion

Summary

- Lattice-based crypto relies on hardness of finding short vectors
- State-of-the-art basis reduction: BKZ with fast SVP subroutine
- Enumeration for SVP:
  - Brute-force all combinations of basis vectors
  - Memory-efficient
  - Fast pruning heuristics
- Sieving for SVP:
  - Consider pairwise combinations of many lattice vectors
  - Large memory requirement
  - Practical near neighbor speed-ups
- Theory: Sieving superior in high dimensions
- Practice: Enumeration superior in low dimensions
- Hardness estimates: Commonly based on sieving

Conclusion

Open problems

Challenges in provable SVP algorithms

- Sieving: Control distribution of lattice vectors
- DGS: Combine vectors more efficiently
- Obtain proven bounds for “random” lattices

Challenges in heuristic SVP algorithms

- Enumeration: Find best pruning methods
- Sieving: Find best near neighbor techniques
- Effective time–memory trade-offs

Challenges in SVP hardness estimation

- Analyze sieving in a more realistic memory model
- Lower bounds (SVP, sieving, enumeration, ...)
- Consensus on attack model/memory costs

Questions?