Technische Universiteit **Eindhoven** University of Technology

Lattice algorithms for the closest vector problem with preprocessing

Thijs Laarhoven

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

RISC seminar, Amsterdam, The Netherlands
(May 3, 2019)

Lattices

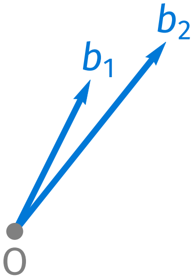
Basics

Lattices

Basics

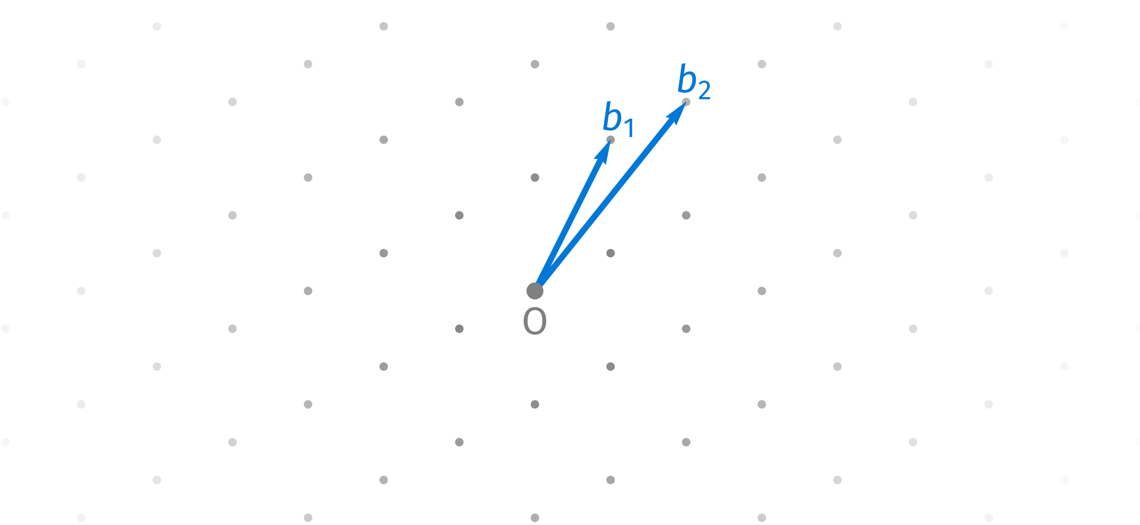
Lattices

Basics



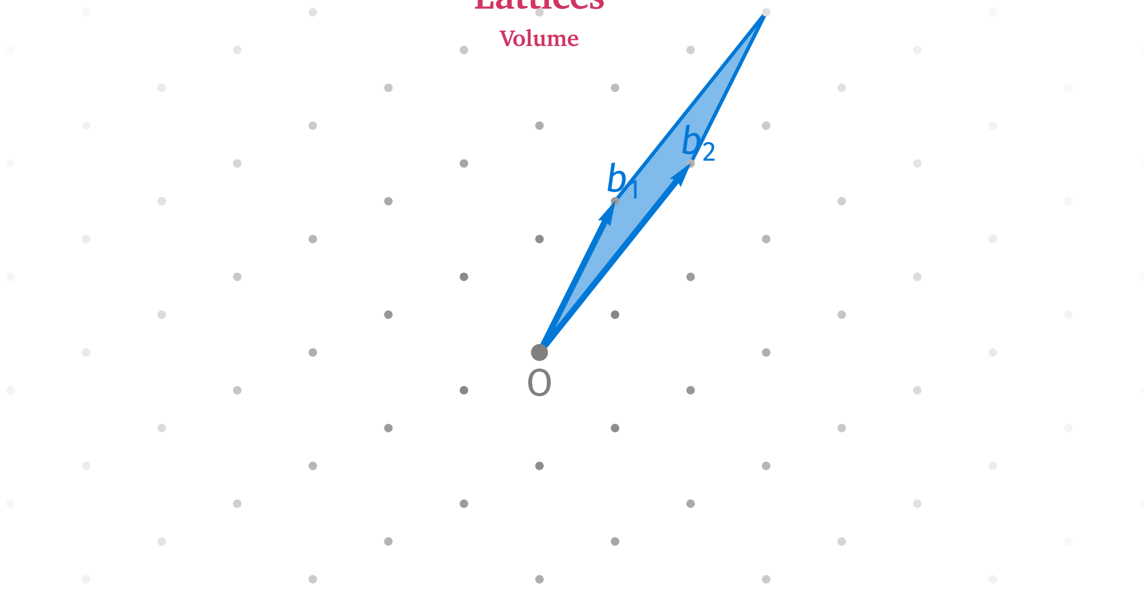
Lattices

Basics



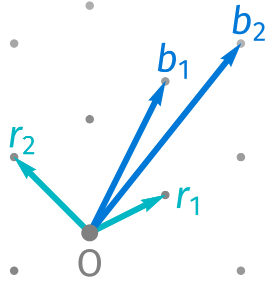
Lattices

Volume



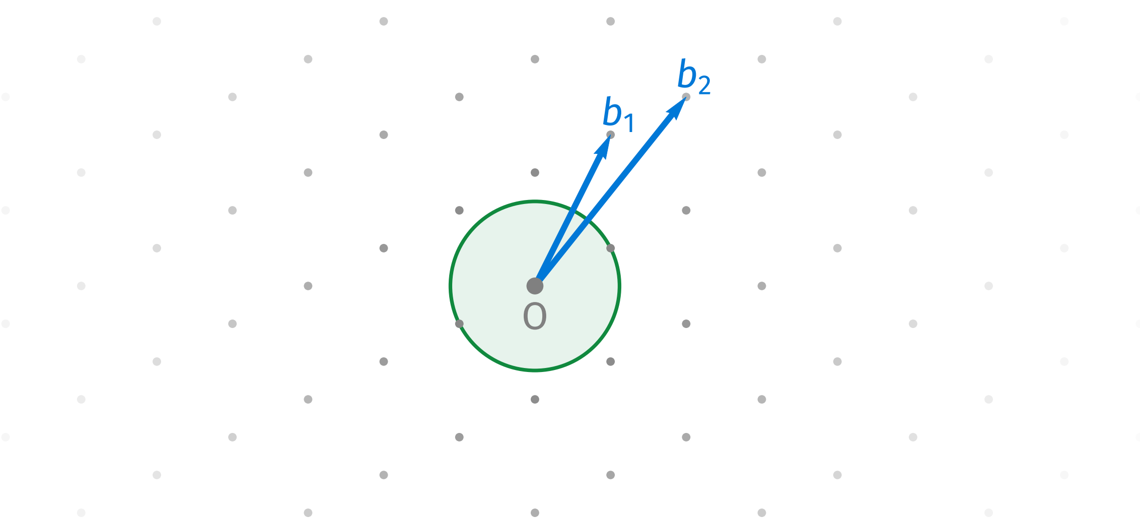
Lattices

Lattice basis reduction



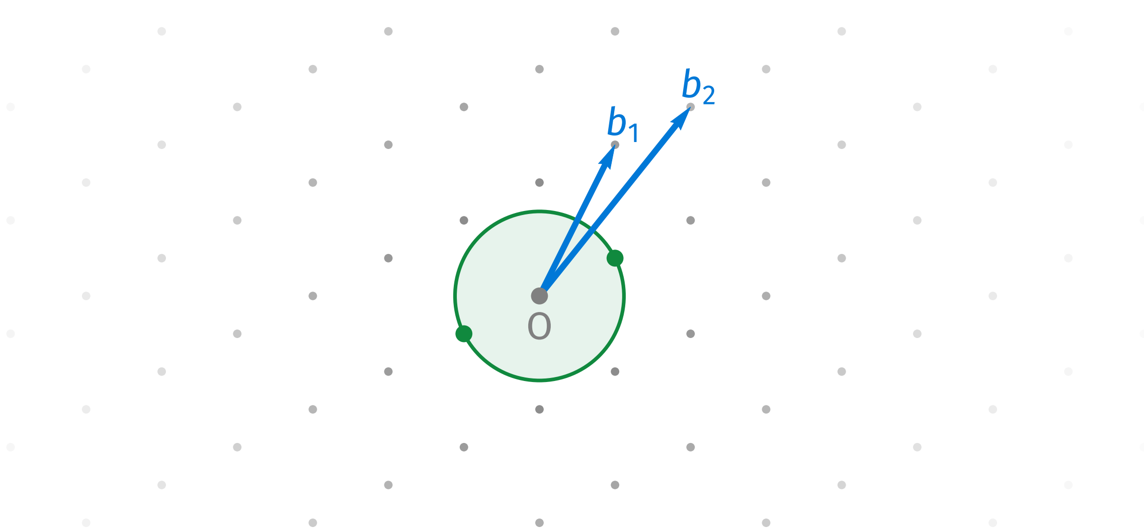
Lattice problems

Shortest Vector Problem (SVP)



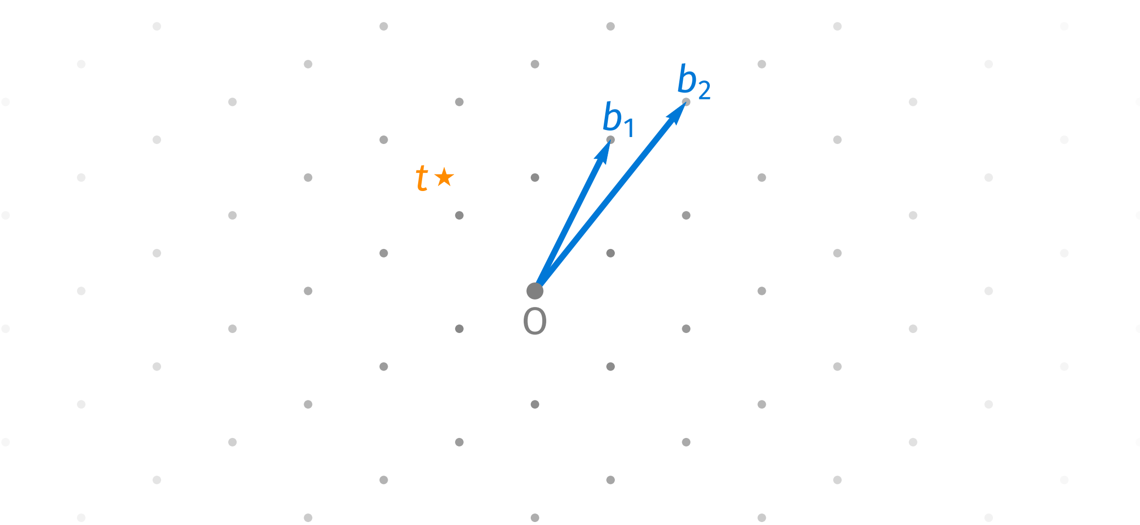
Lattice problems

Shortest Vector Problem (SVP)



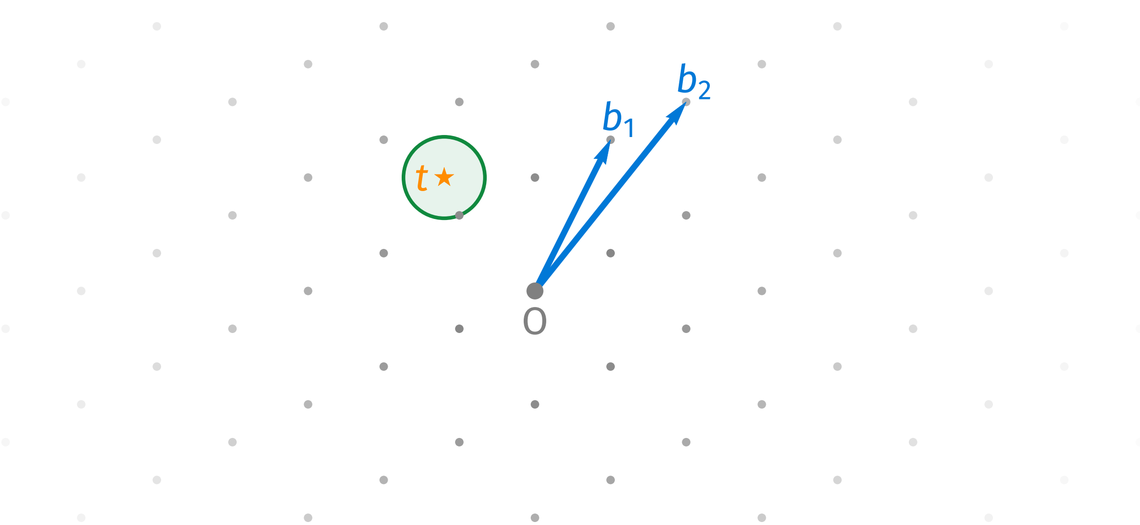
Lattice problems

Closest Vector Problem (CVP)



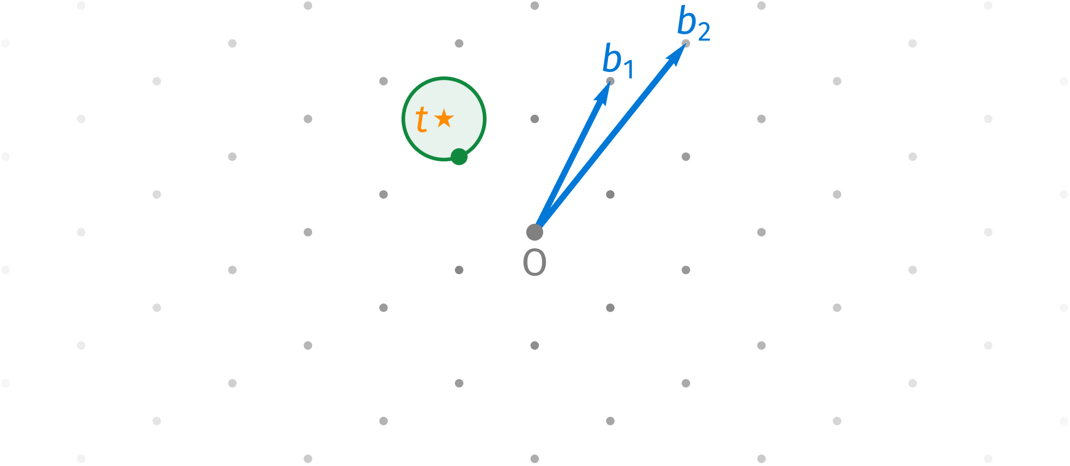
Lattice problems

Closest Vector Problem (CVP)



Lattice problems

Closest Vector Problem (CVP)



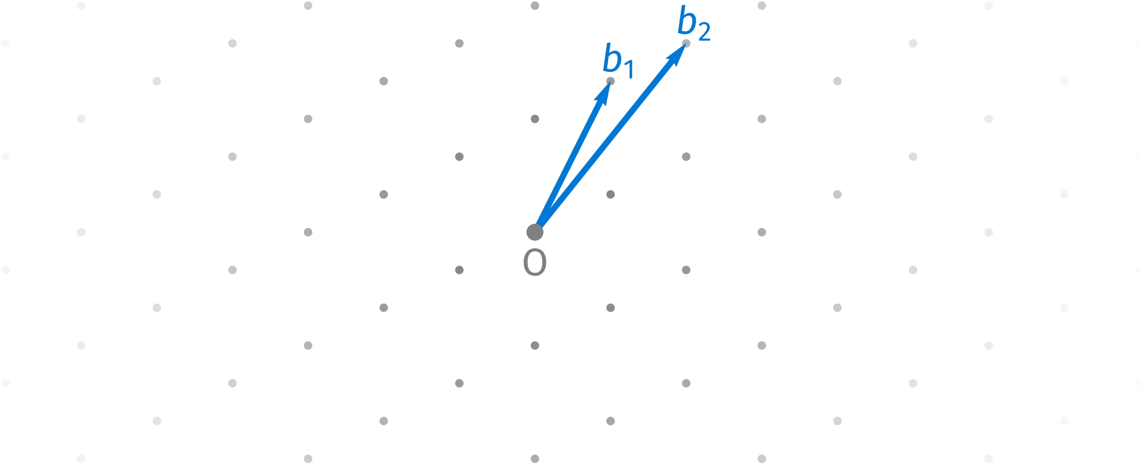
Lattice problems

SVP/CVP asymptotics

	Algorithm	$\log_2(\text{Time})$	$\log_2(\text{Space})$	Experiments
Worst-case SVP	Enumeration [Poh81, Kan83, ..., MW15, AN17]	$O(n \log n)$	$O(\log n)$	**152**
	AKS-sieve [AKS01, NV08, MV10, HPS11]	$3.398n$	$1.985n$	–
	Birthday sieves [PS09, HPS11]	$2.465n$	$1.233n$	–
	Enumeration/DGS hybrid [CCL17]	$2.048n$	$0.500n$	–
	Voronoi cell algorithm [AEVZ02, MV10b, BD15]	$2.000n$	$1.000n$	40
	Quantum sieve [LMP13, LMP15]	$1.799n$	$1.286n$	–
	Quantum enum/DGS [CCL17]	$1.256n$	**0.500n**	–
	Discrete Gaussian sampling [ADRS15, ADS15, AS18]	**1.000n**	$1.000n$	–
Average-case SVP	The Nguyen–Vidick sieve [NV08]	$0.415n$	$0.208n$	50
	GaussSieve [MV10, ..., IKMT14, BNvdP16, YKYC17]	$0.415n$	$0.208n$	130*
	Triple sieve [BLS16, HK17]	$0.396n$	$0.189n$	80
	Kleinjung sieve [Kle14]	$0.379n$	$0.189n$	116
	Leveled sieving [WLTB11, ZPH13]	$0.378n$	$0.283n$	–
	Overlattice sieve [BGJ14]	$0.377n$	$0.293n$	90
	Triple sieve with NNS [HK17, HKL18]	$0.359n$	**0.189n**	76
	Single filters [DL17, ADH+19]	$0.349n$	$0.246n$	**155**
	Hyperplane LSH [Cha02, FBB+14, Laa15, ..., LM18]	$0.337n$	$0.337n$	107
	Hypercube LSH [TT07, Laa17]	$0.322n$	$0.322n$	–
	May–Ozerov NNS [MO15, BGJ15]	$0.311n$	$0.311n$	–
	Quantum sieve [LMP13]	$0.311n$	$0.208n$	–
	Spherical LSH [AINR14, LdW15]	$0.297n$	$0.297n$	–
	Cross-polytope LSH [TT07, AILRS15, BL16, KW17]	$0.297n$	$0.297n$	80
	Spherical LSF [BDGL16, MLB17, ALRW17, Chr17]	**0.292n**	$0.292n$	92
	Quantum NNS sieve [LMP15, Laa16]	**0.265n**	$0.265n$	–

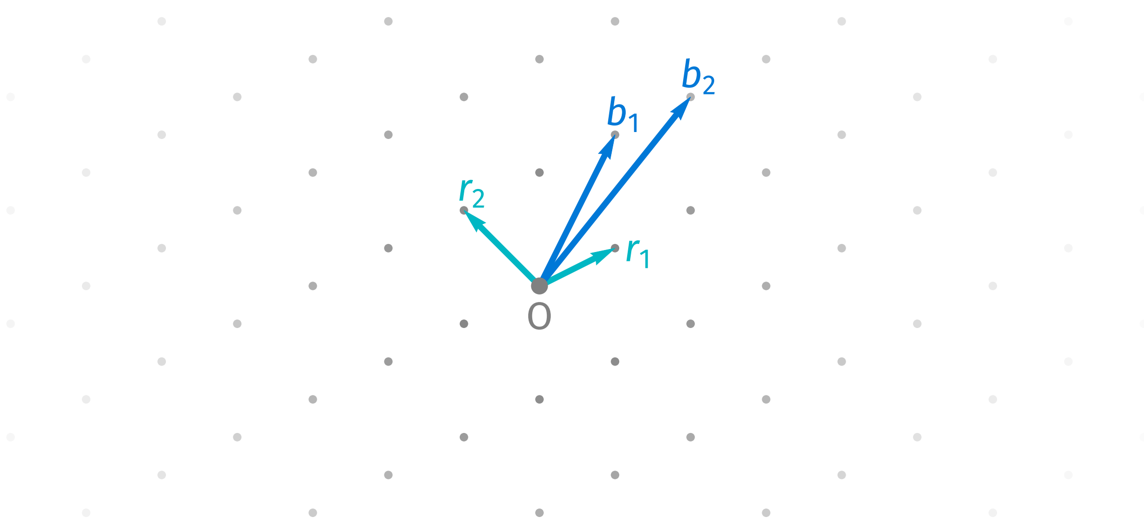
Lattice problems

Closest Vector Problem with Preprocessing (CVPP)



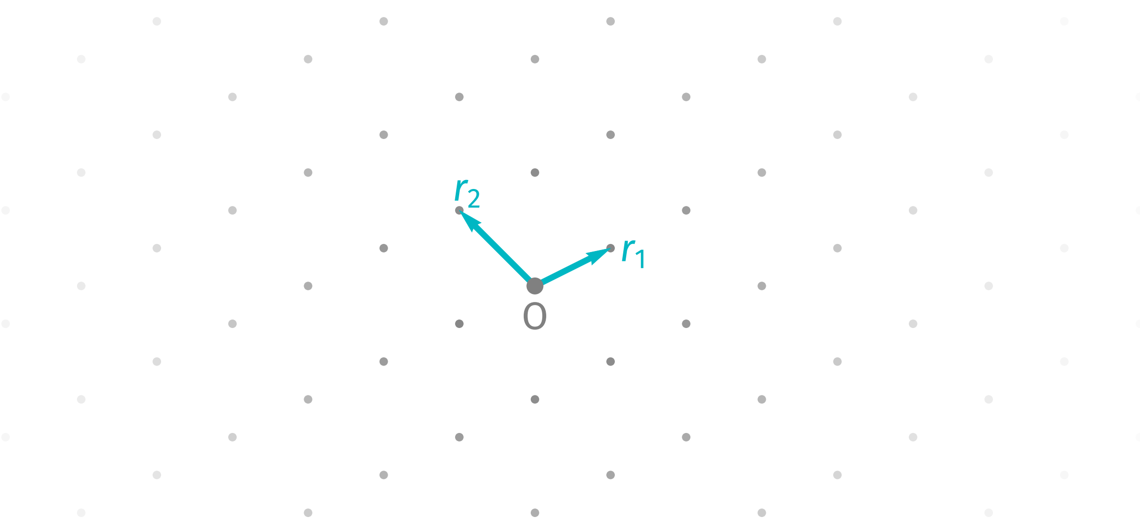
Lattice problems

Closest Vector Problem with Preprocessing (CVPP)



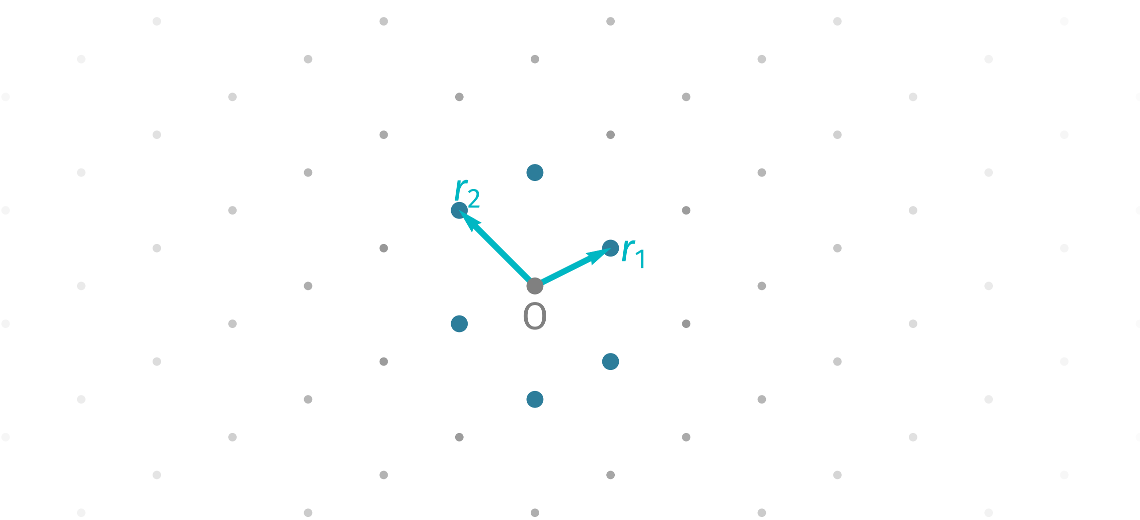
Lattice problems

Closest Vector Problem with Preprocessing (CVPP)



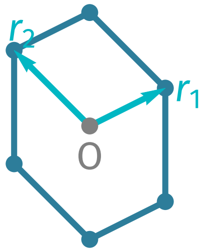
Lattice problems

Closest Vector Problem with Preprocessing (CVPP)



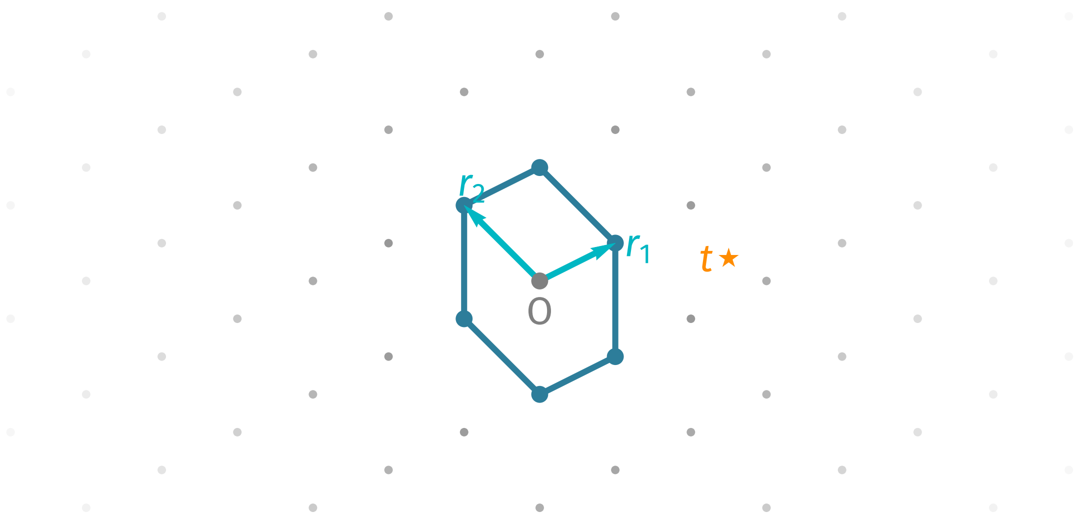
Lattice problems

Closest Vector Problem with Preprocessing (CVPP)



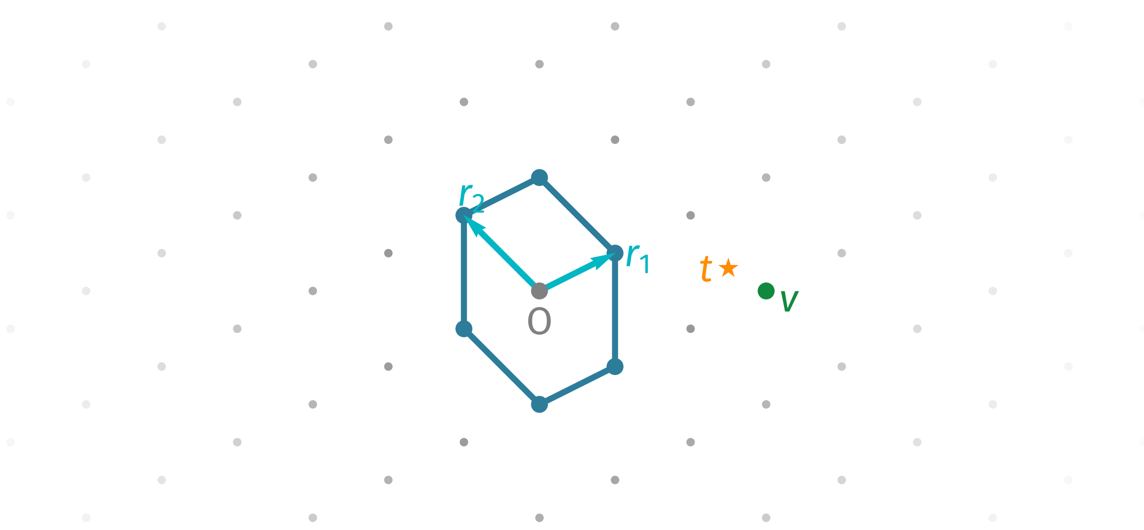
Lattice problems

Closest Vector Problem with Preprocessing (CVPP)



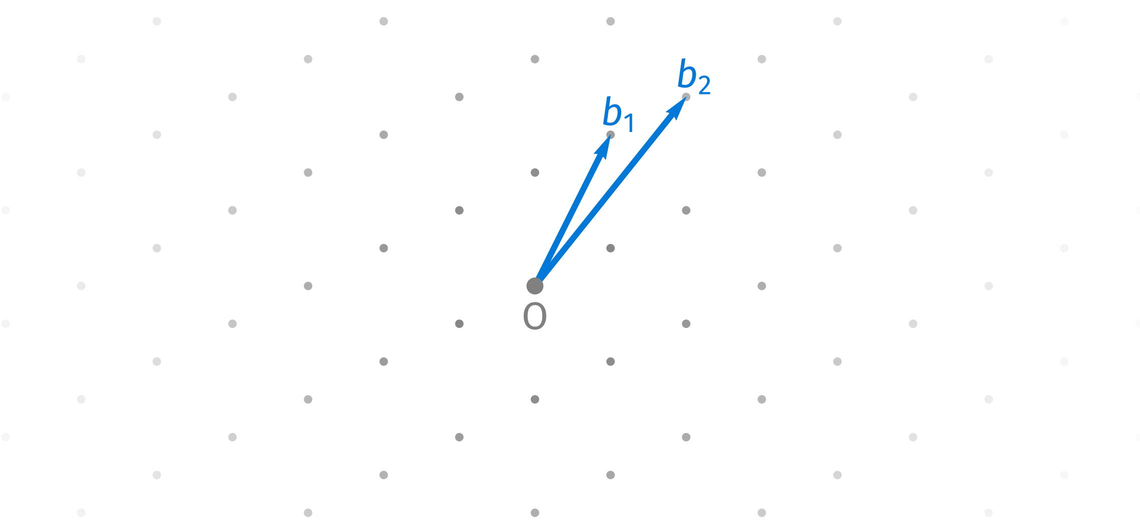
Lattice problems

Closest Vector Problem with Preprocessing (CVPP)



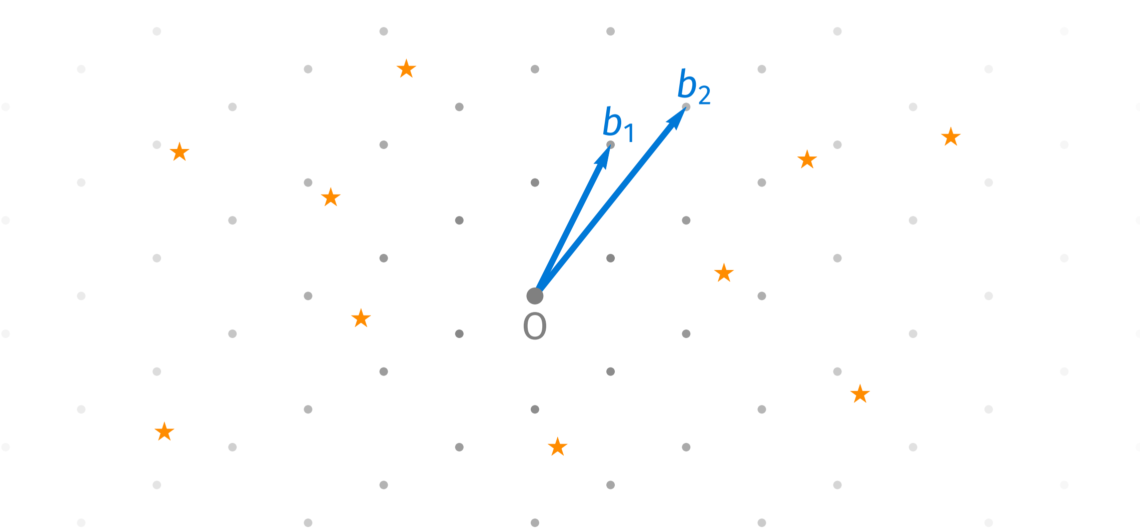
Lattice problems

Batch Closest Vector Problem



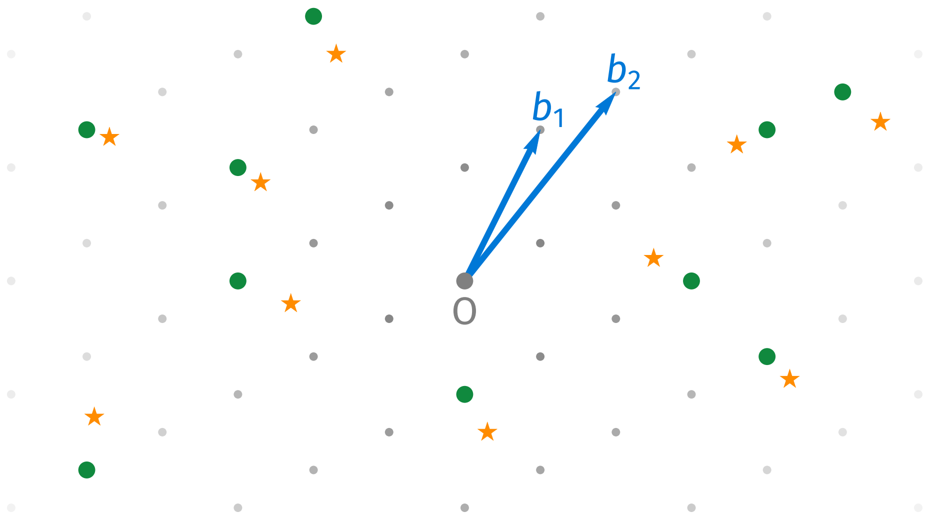
Lattice problems

Batch Closest Vector Problem



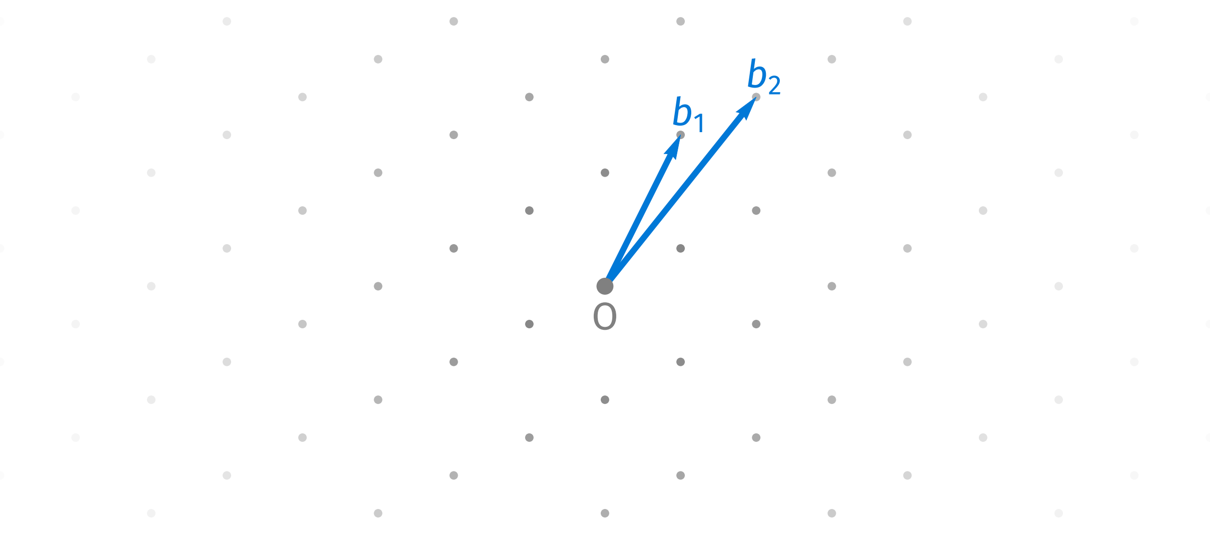
Lattice problems

Batch Closest Vector Problem



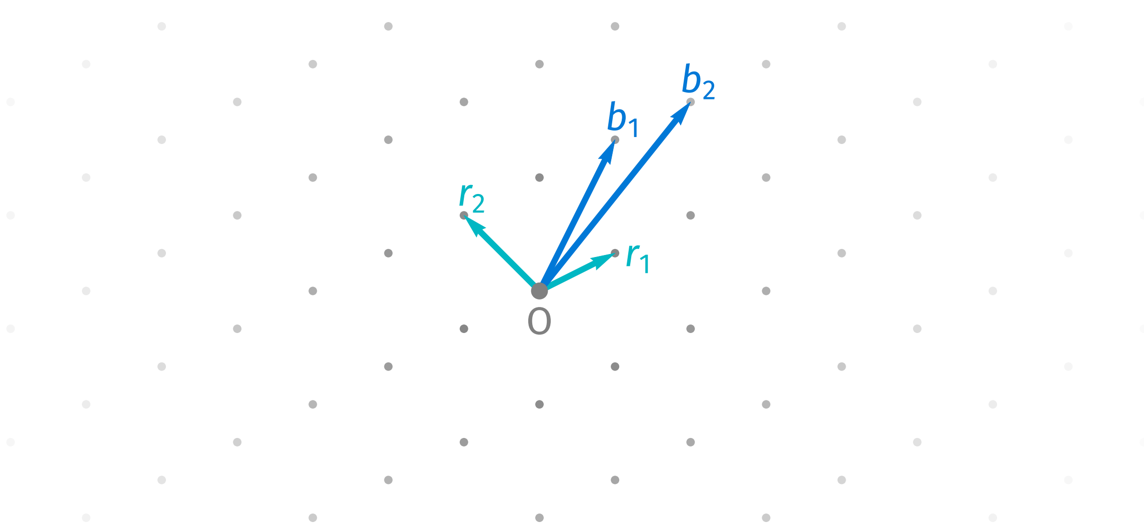
Babai's algorithms

Rounding algorithm [Len84, Bab86]



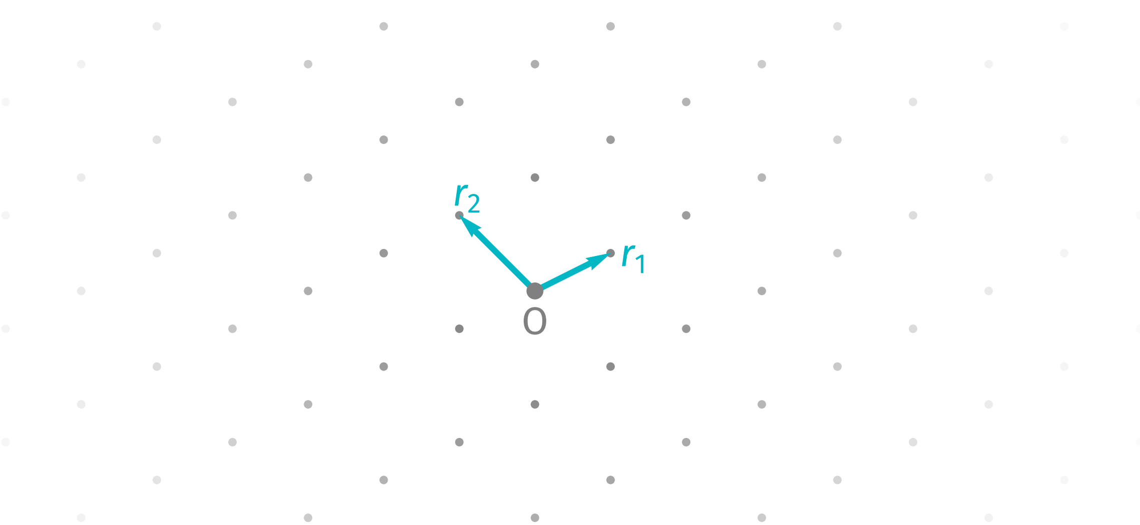
Babai's algorithms

Rounding algorithm [Len84, Bab86]



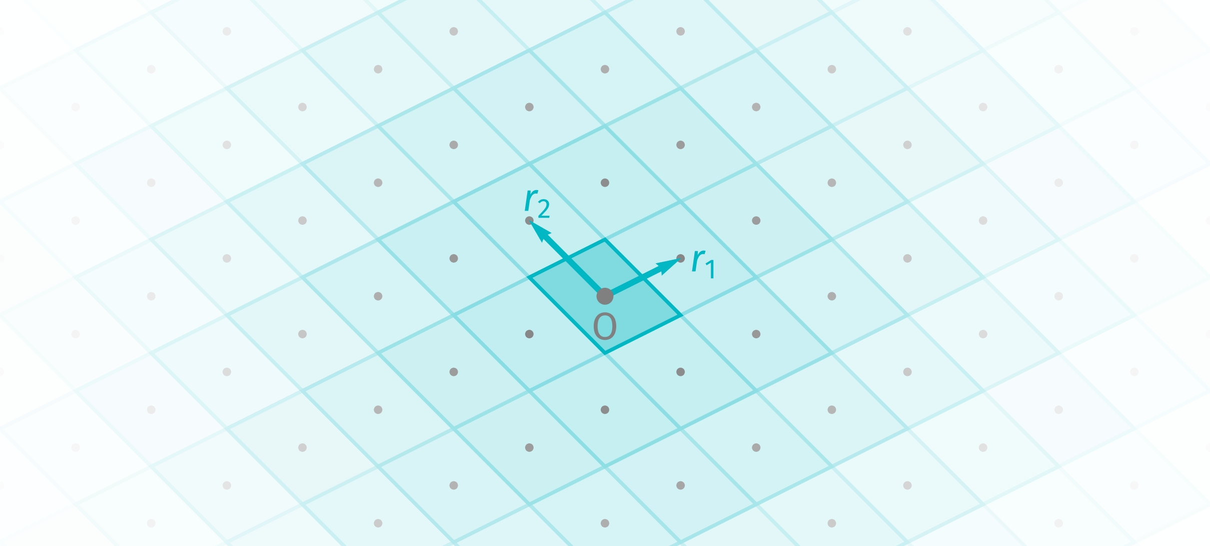
Babai's algorithms

Rounding algorithm [Len84, Bab86]



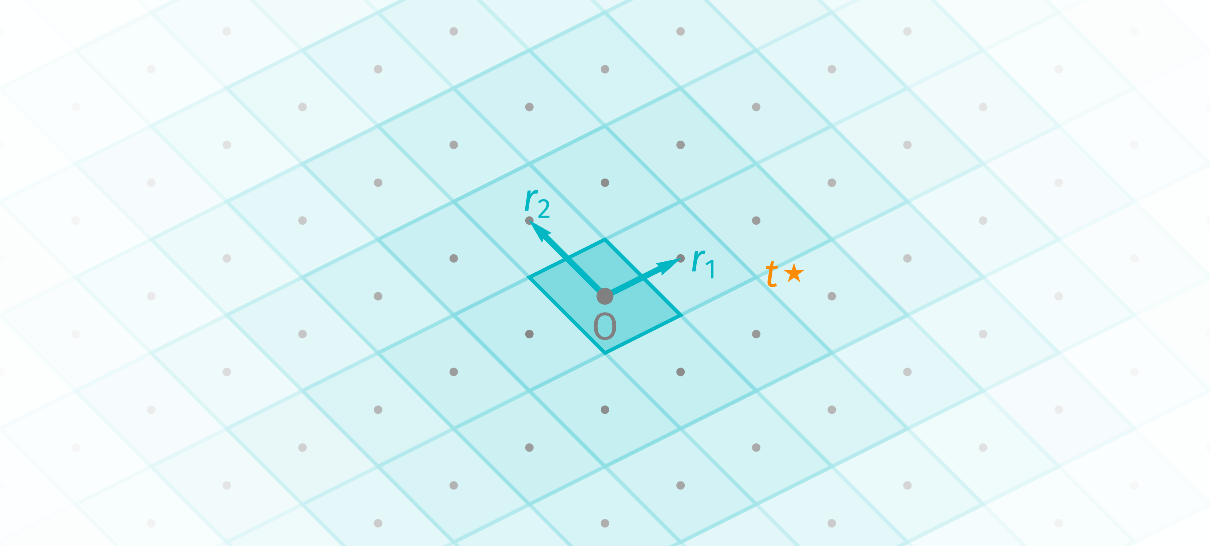
Babai's algorithms

Rounding algorithm [Len84, Bab86]



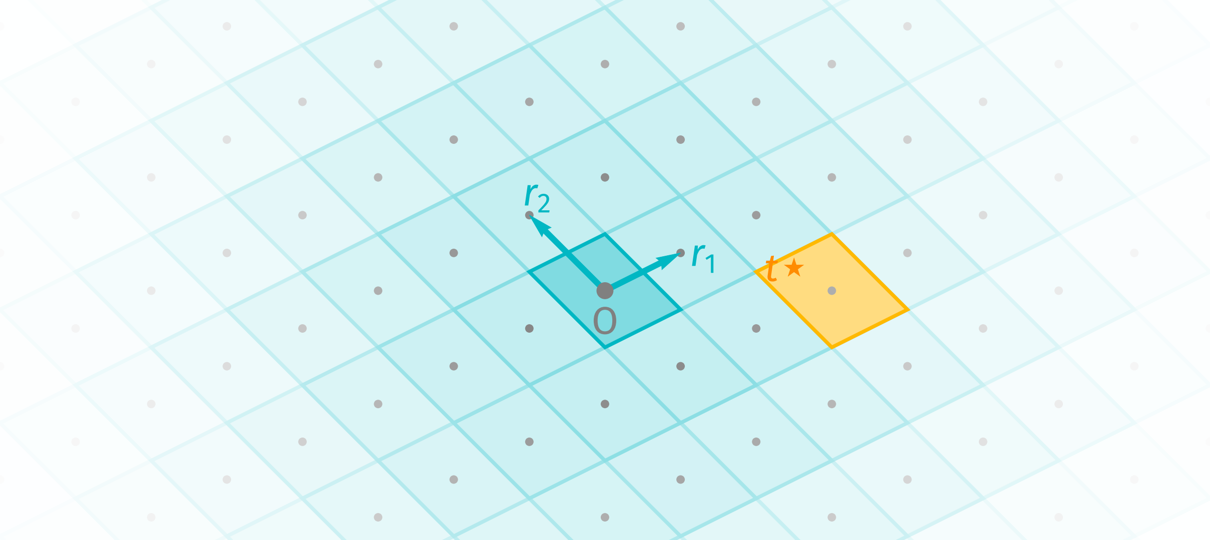
Babai's algorithms

Rounding algorithm [Len84, Bab86]



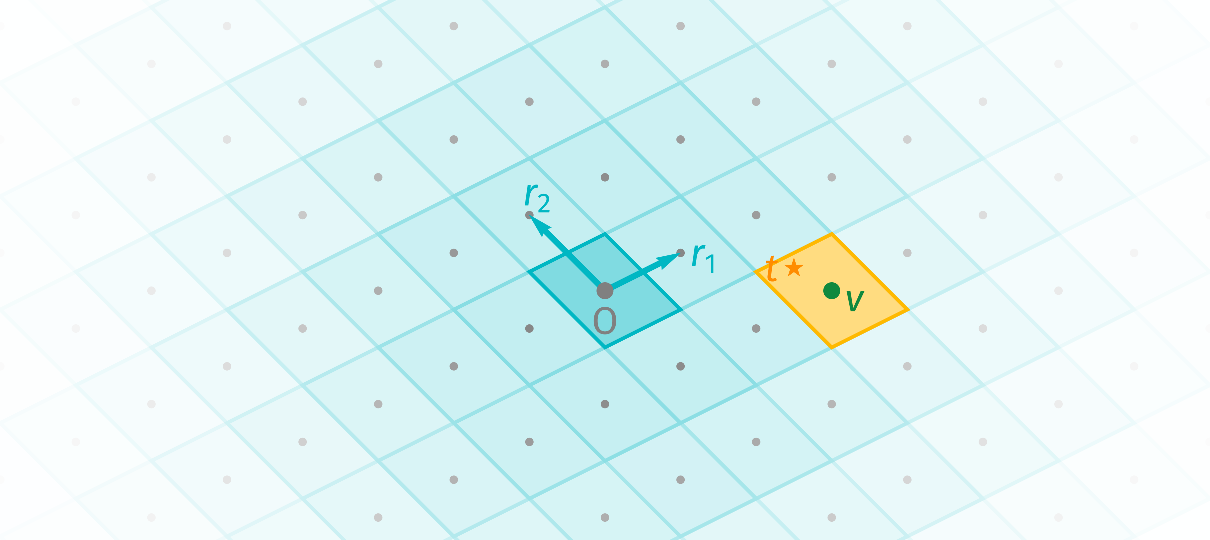
Babai's algorithms

Rounding algorithm [Len84, Bab86]



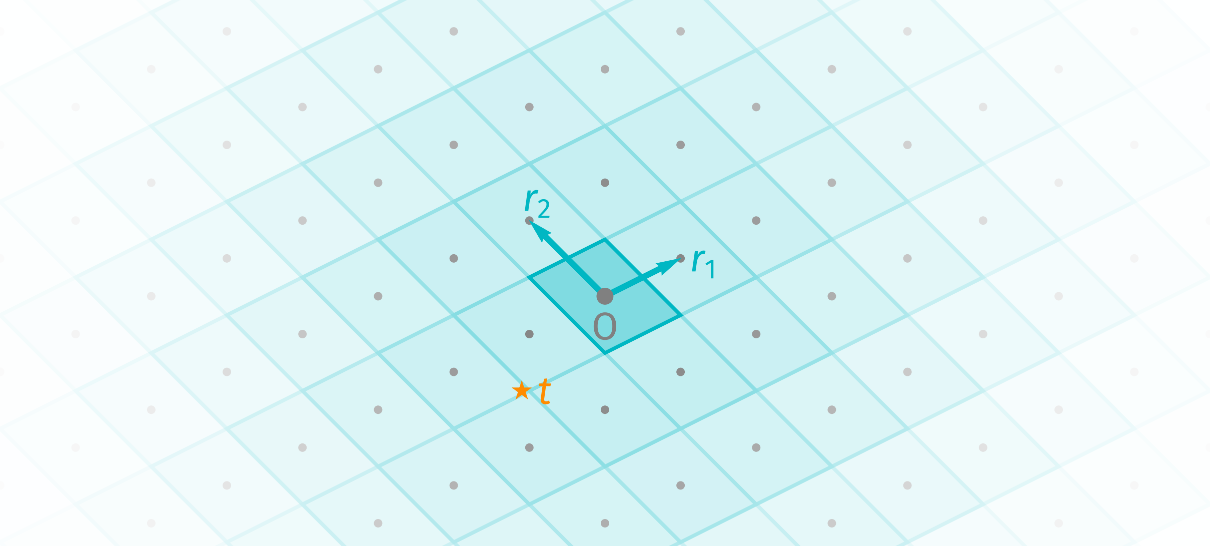
Babai's algorithms

Rounding algorithm [Len84, Bab86]



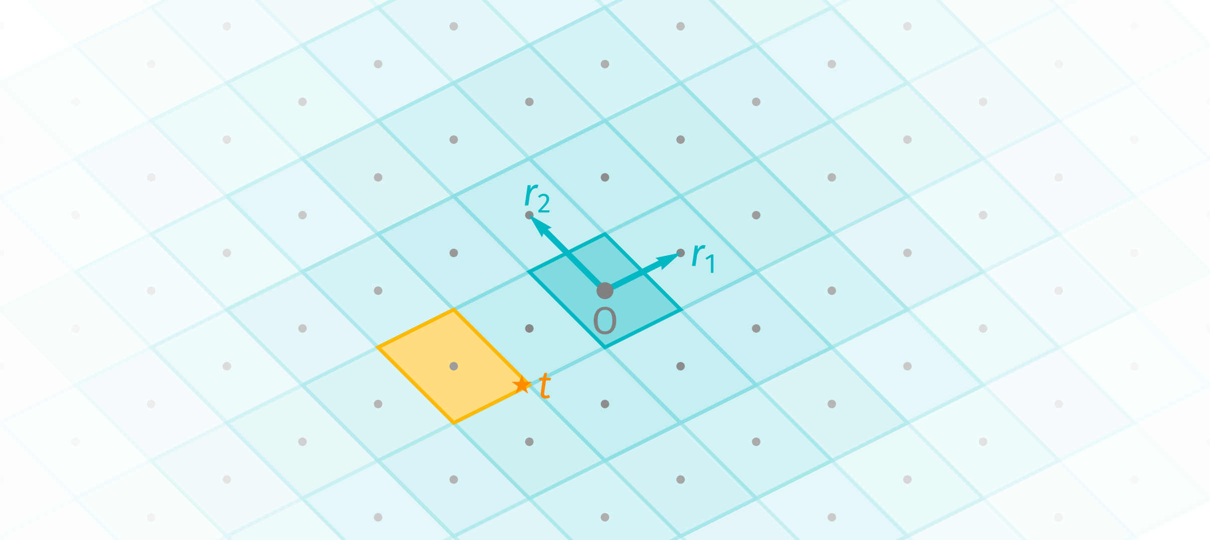
Babai's algorithms

Rounding algorithm [Len84, Bab86]



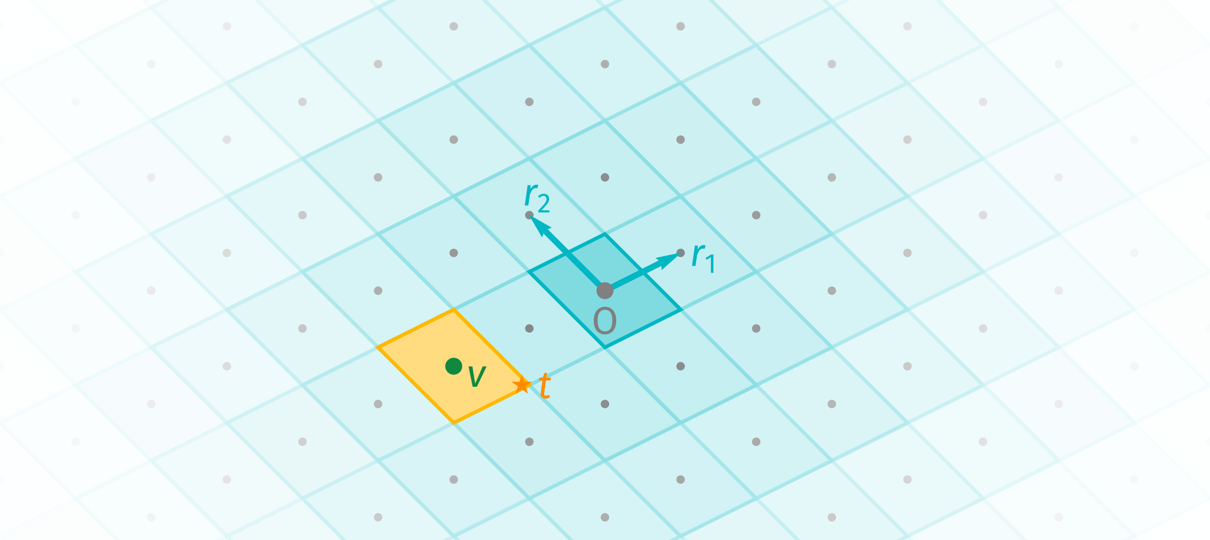
Babai's algorithms

Rounding algorithm [Len84, Bab86]



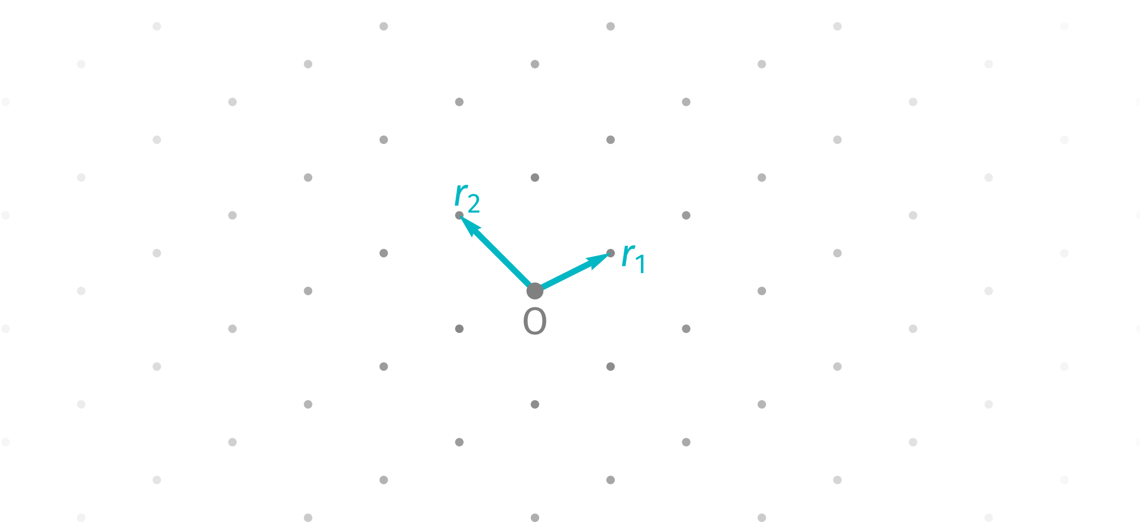
Babai's algorithms

Rounding algorithm [Len84, Bab86]



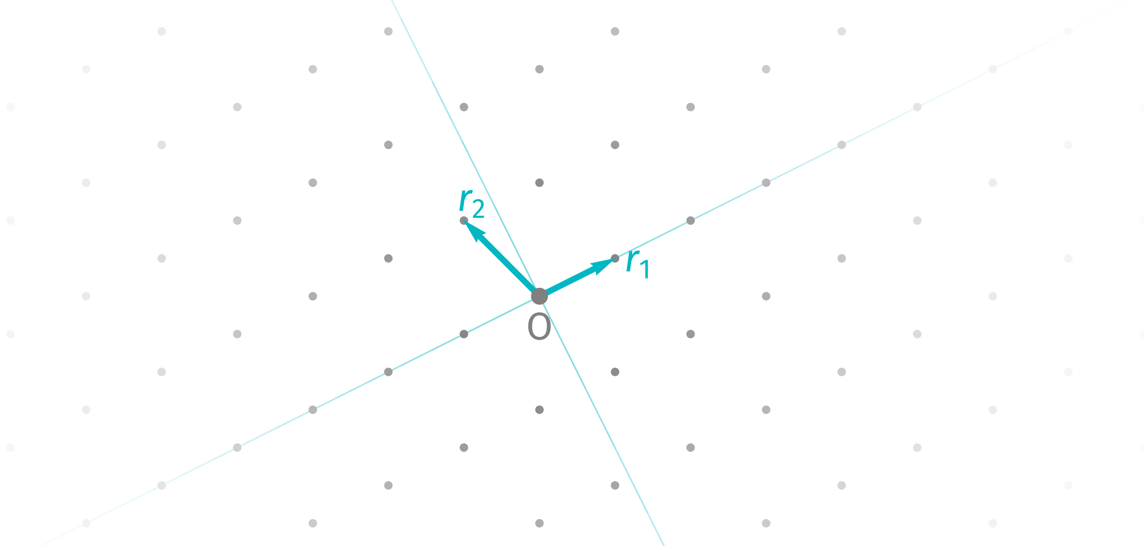
Babai's algorithms

Gram-Schmidt orthogonalization



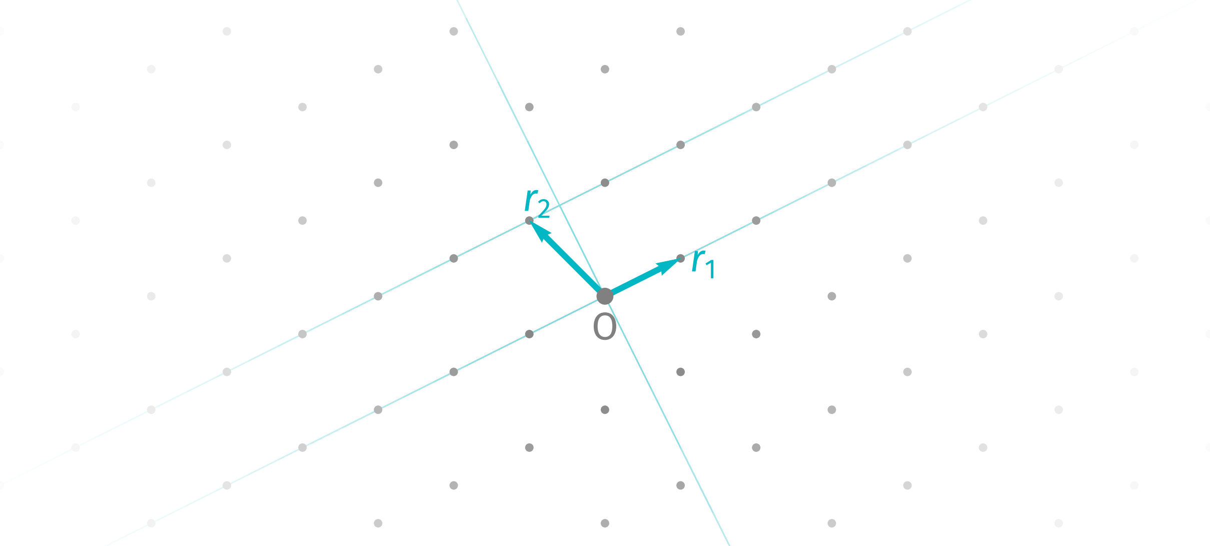
Babai's algorithms

Gram-Schmidt orthogonalization



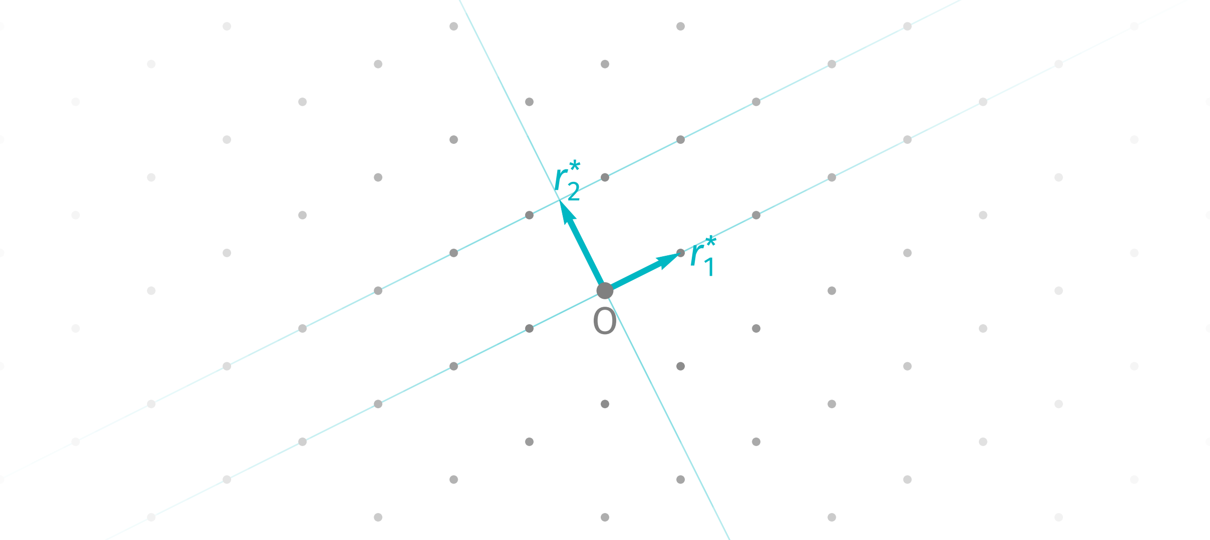
Babai's algorithms

Gram-Schmidt orthogonalization



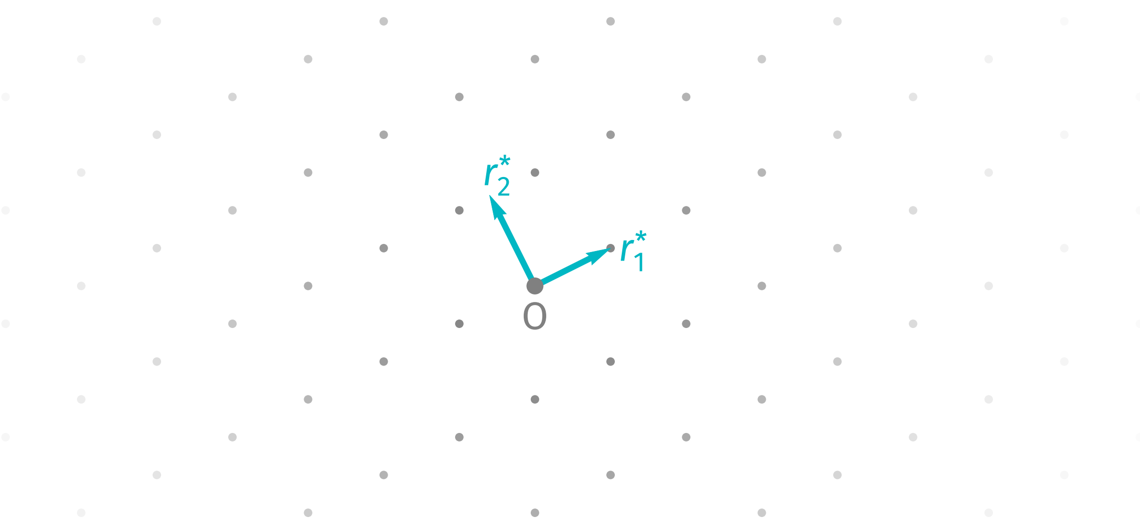
Babai's algorithms

Gram-Schmidt orthogonalization



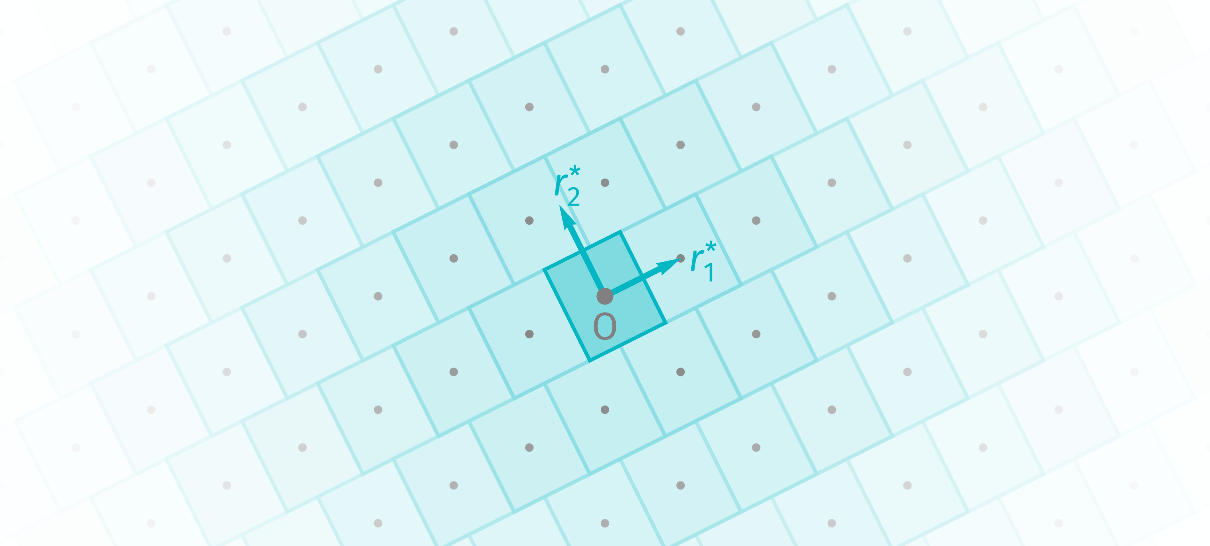
Babai's algorithms

Nearest plane algorithm [Bab86]



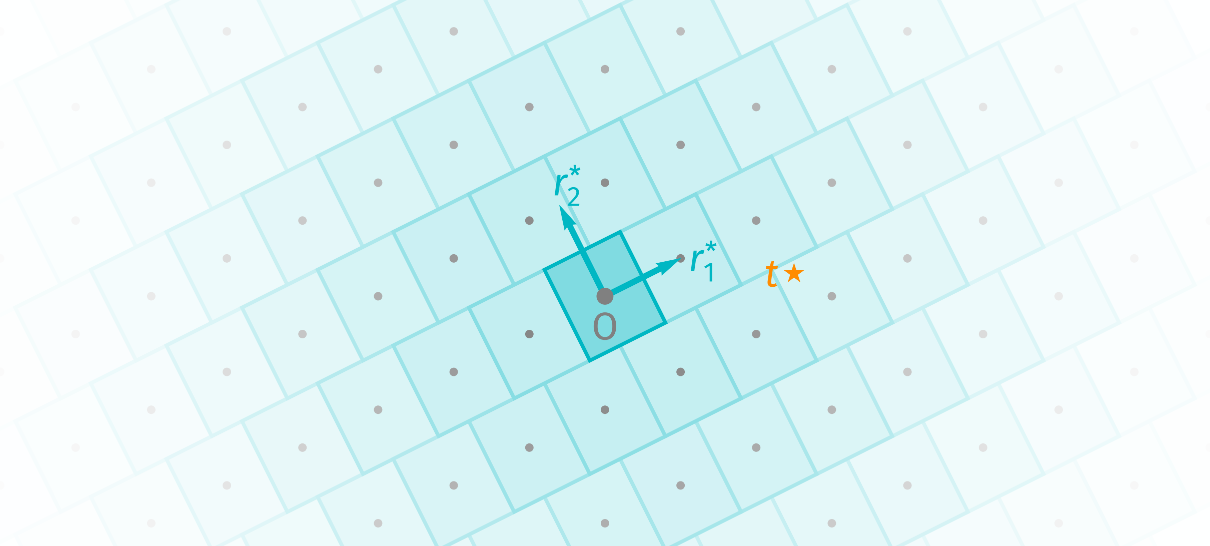
Babai's algorithms

Nearest plane algorithm [Bab86]



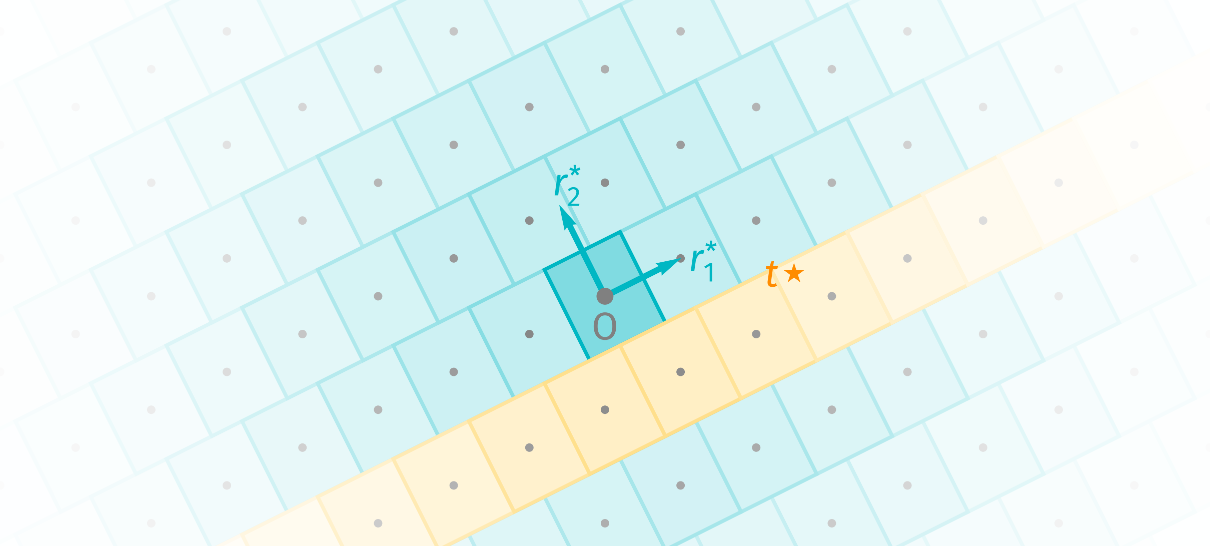
Babai's algorithms

Nearest plane algorithm [Bab86]



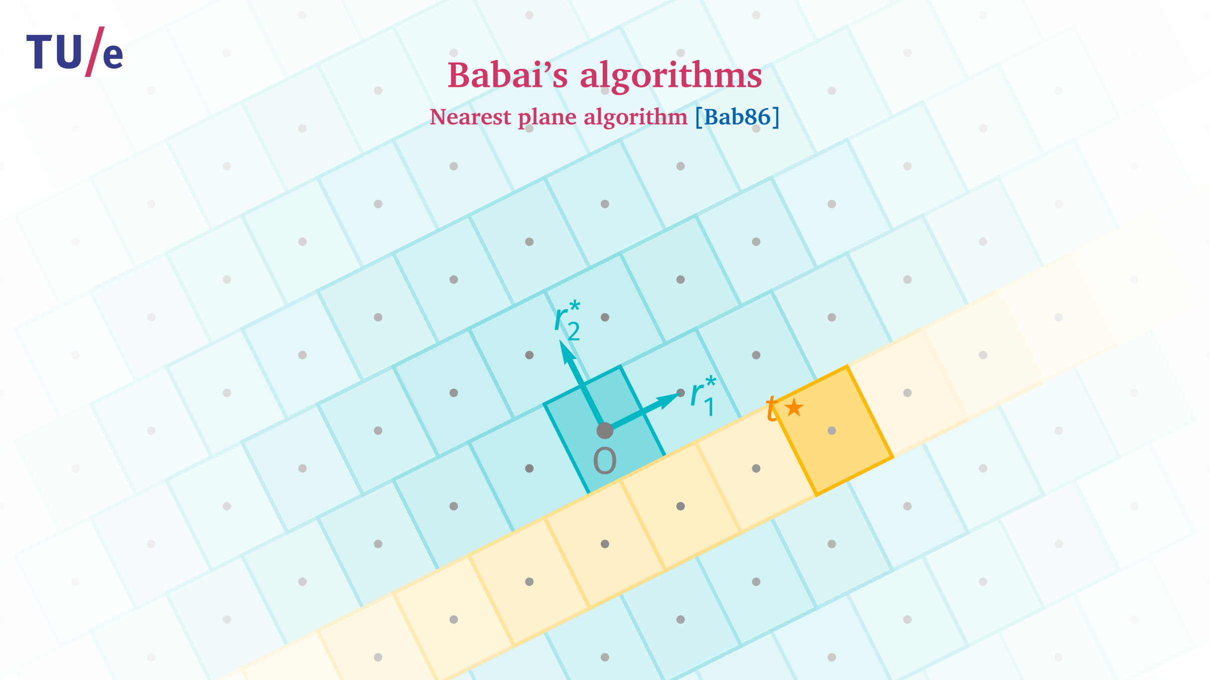
Babai's algorithms

Nearest plane algorithm [Bab86]



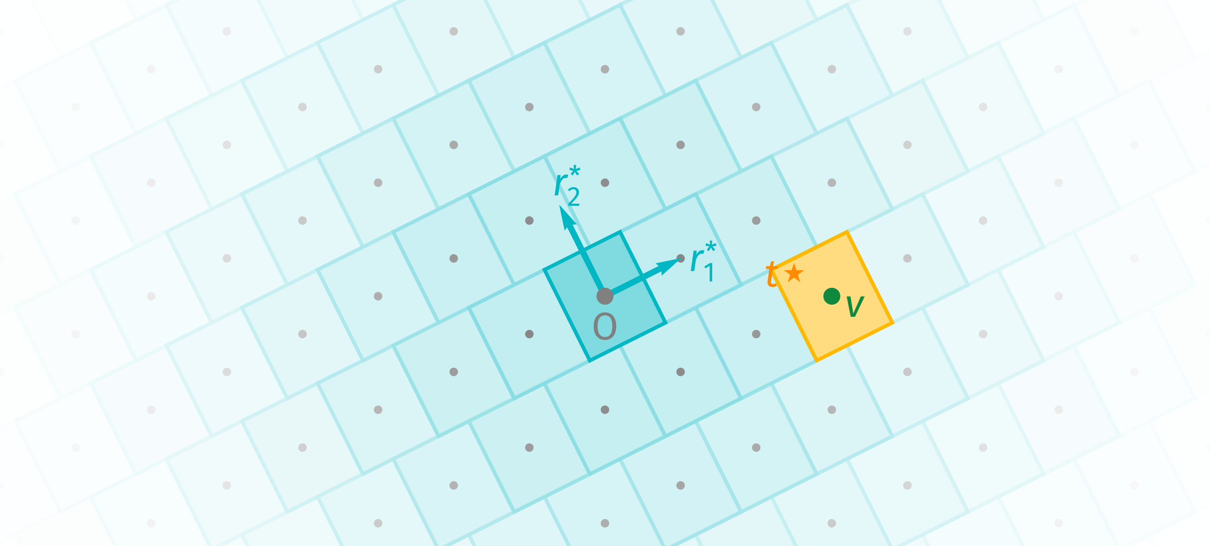
Babai's algorithms

Nearest plane algorithm [Bab86]



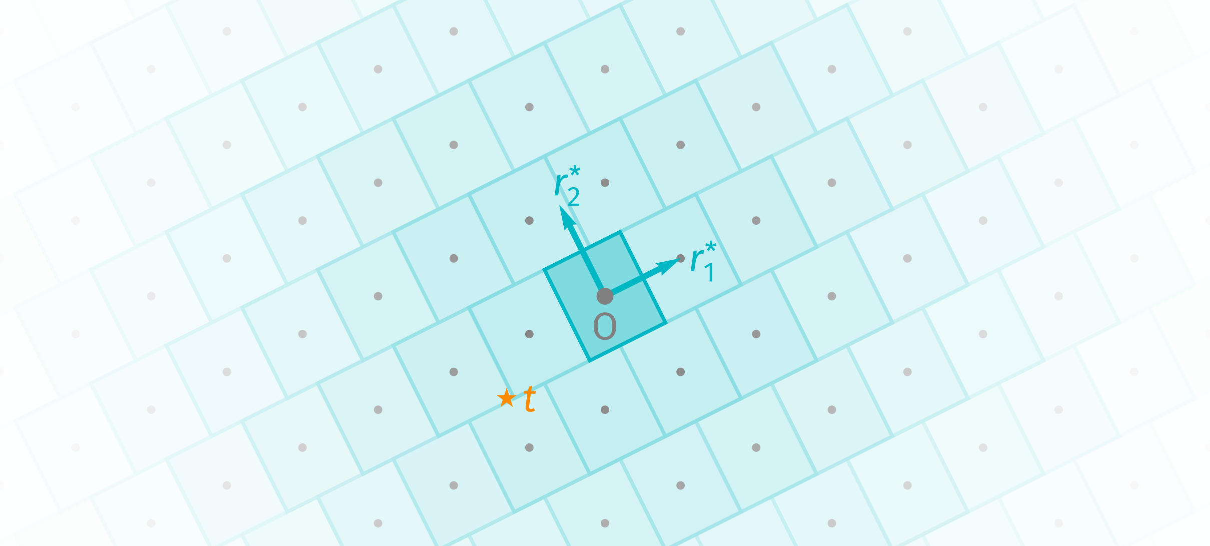
Babai's algorithms

Nearest plane algorithm [Bab86]



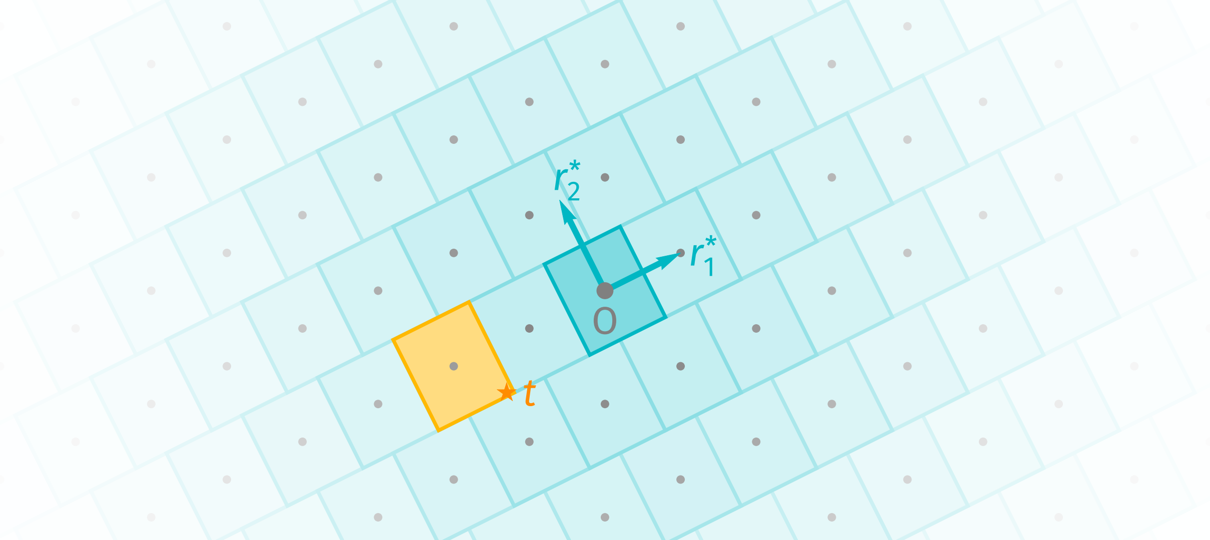
Babai's algorithms

Nearest plane algorithm [Bab86]



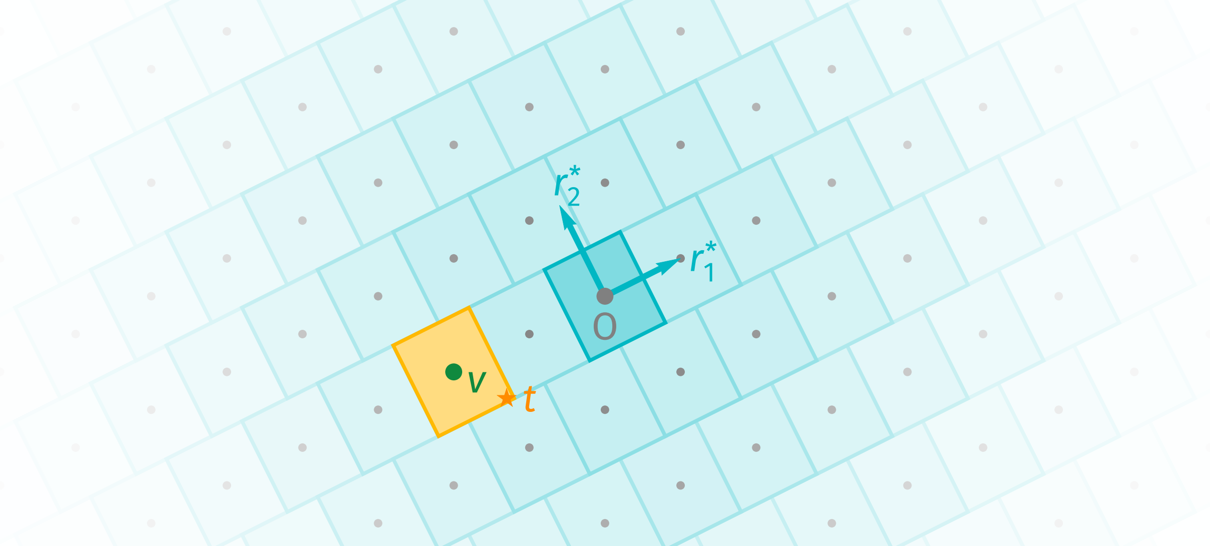
Babai's algorithms

Nearest plane algorithm [Bab86]



Babai's algorithms

Nearest plane algorithm [Bab86]



Babai's algorithms

Overview

- *Preprocessing*: find a short basis ($2^{O(n)}$ time, poly(n) space)

Babai's algorithms

Overview

- *Preprocessing*: find a short basis ($2^{O(n)}$ time, poly(n) space)
- *Query*: round-off or nearest-planes (poly(n) time)

Babai's algorithms

Overview

- *Preprocessing*: find a short basis ($2^{O(n)}$ time, poly(n) space)
- *Query*: round-off or nearest-planes (poly(n) time)
- *Strengths*: fast and simple algorithms

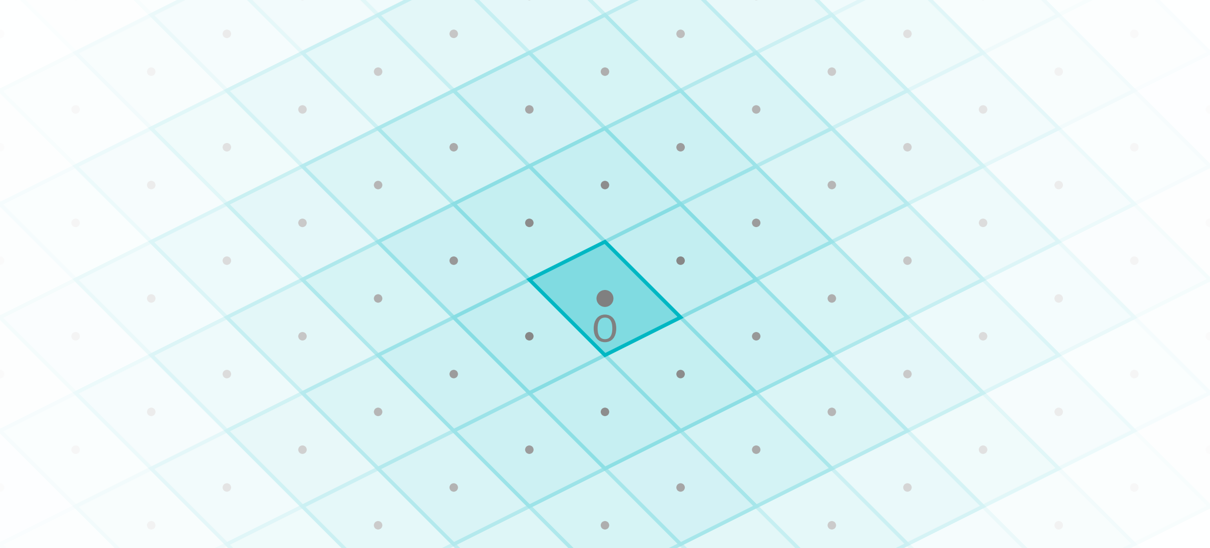
Babai's algorithms

Overview

- *Preprocessing*: find a short basis ($2^{O(n)}$ time, poly(n) space)
- *Query*: round-off or nearest-planes (poly(n) time)
- *Strengths*: fast and simple algorithms
- *Limitations*: does not always solve CVPP

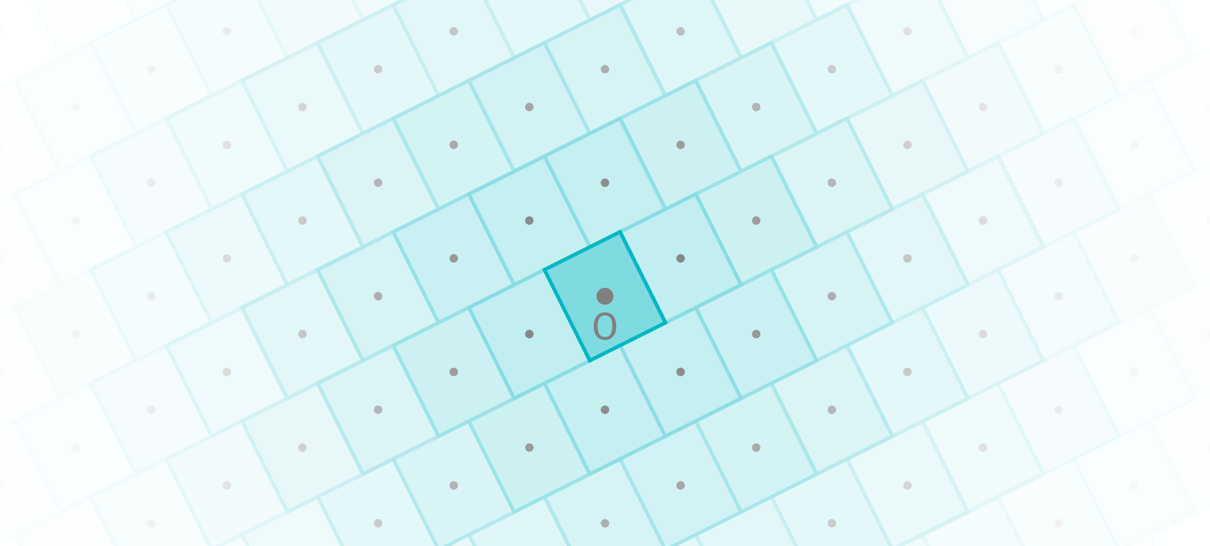
Voronoi cells

Round-off tiling



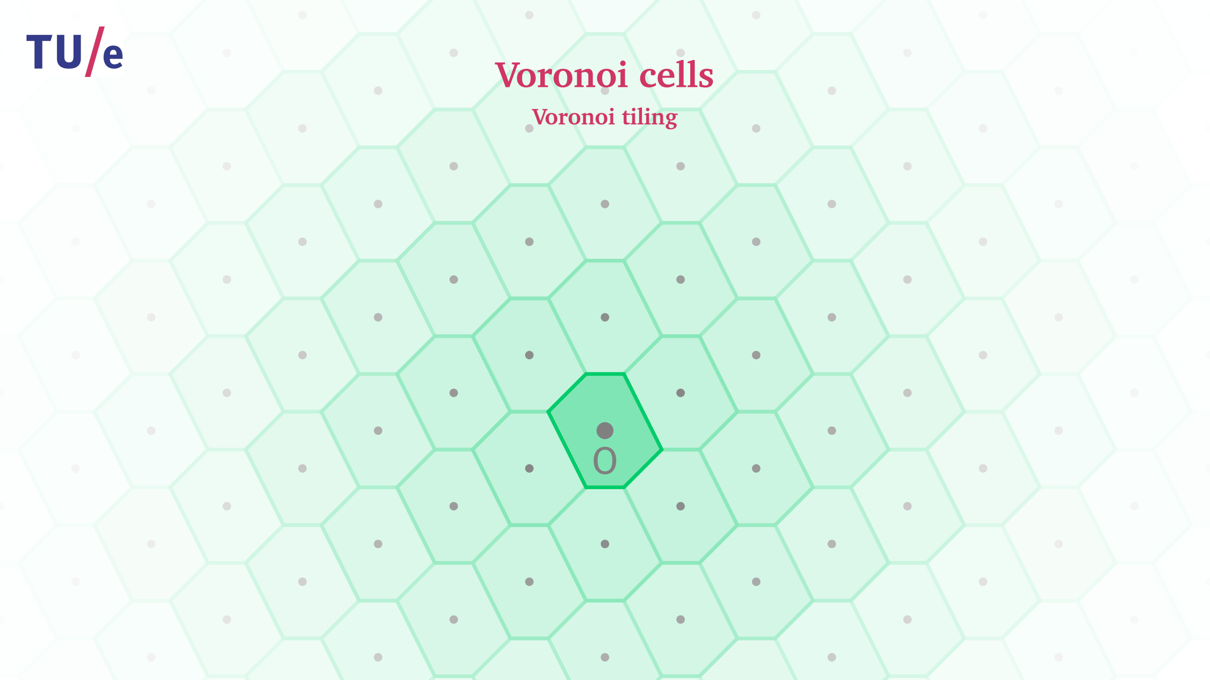
Voronoi cells

Nearest-plane tiling



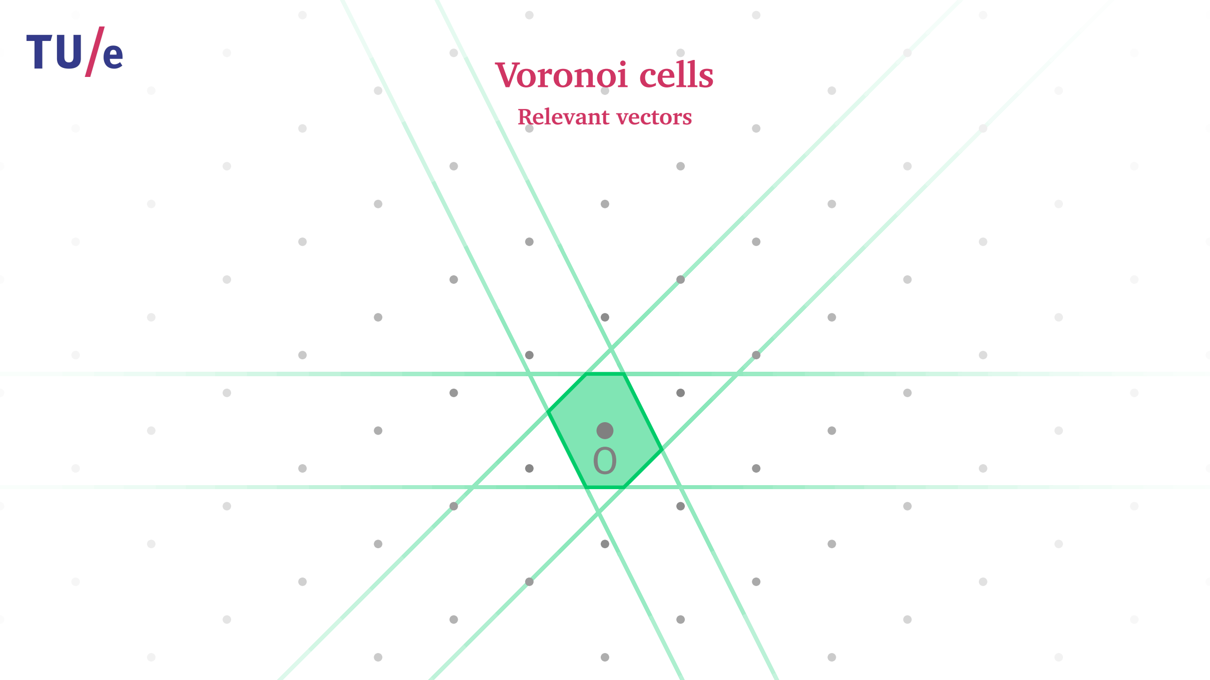
Voronoi cells

Voronoi tiling



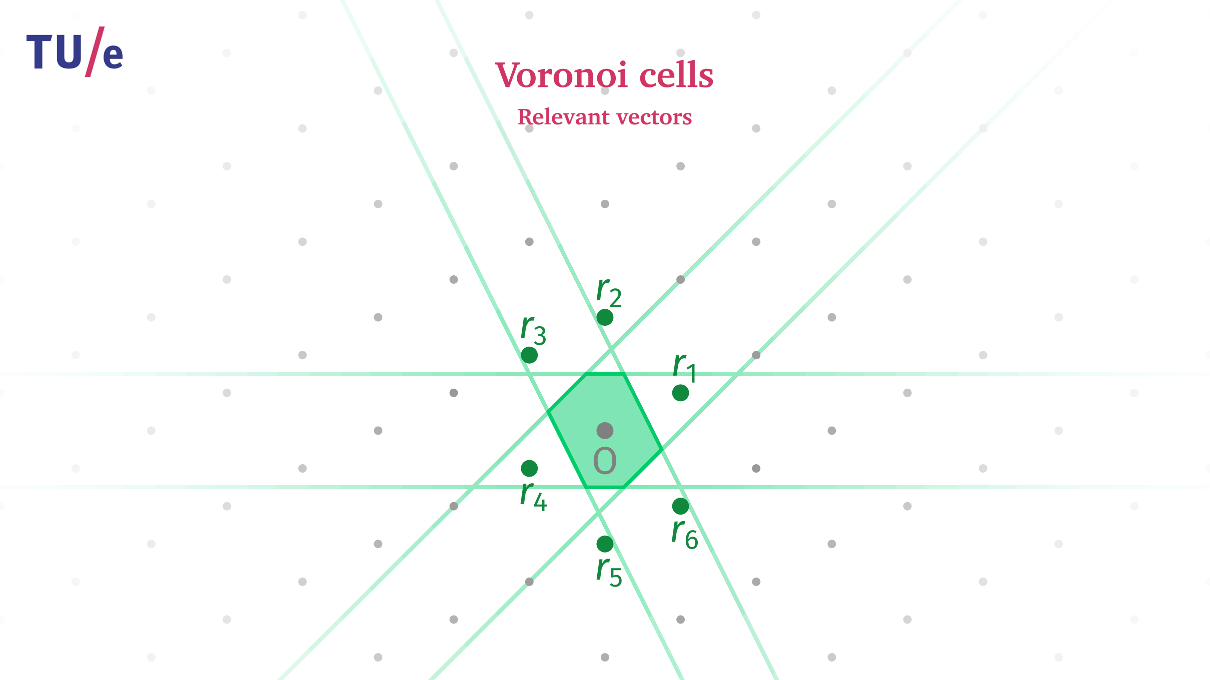
Voronoi cells

Relevant vectors



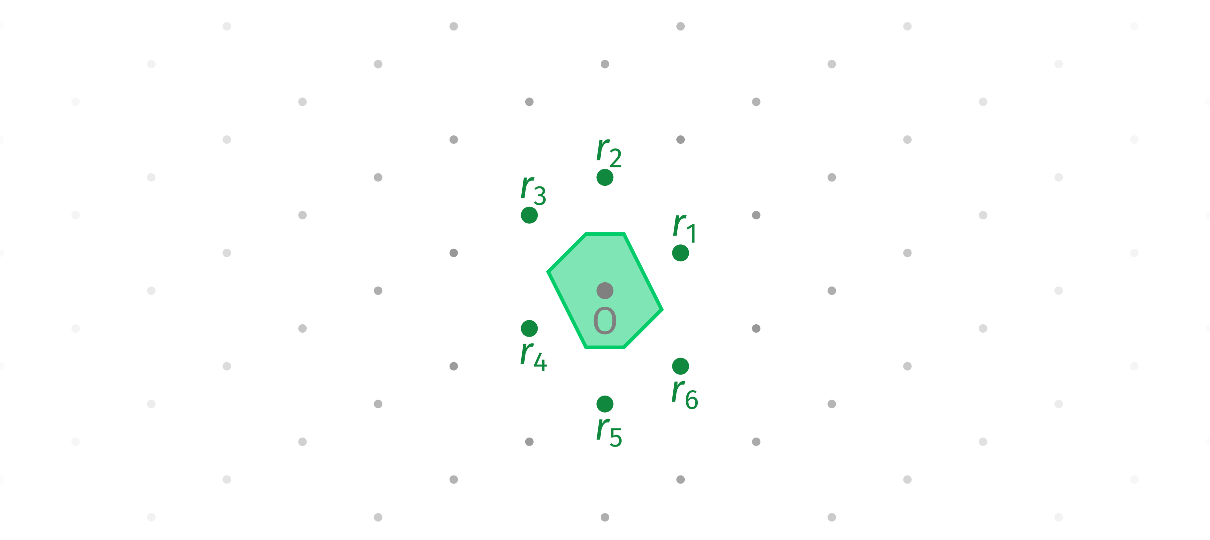
Voronoi cells

Relevant vectors



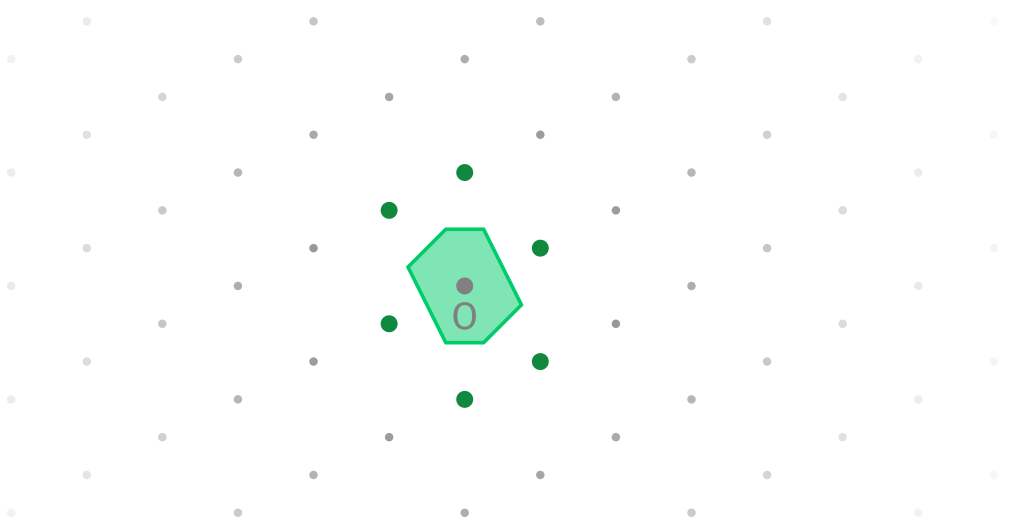
Voronoi cells

Relevant vectors



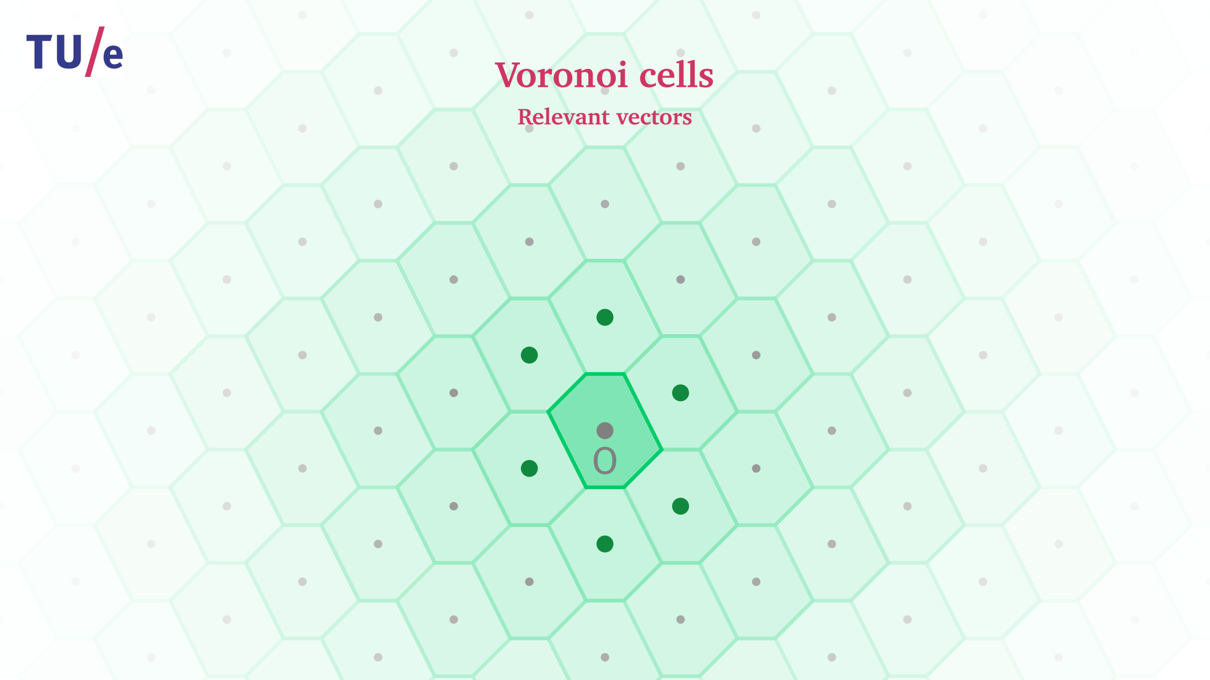
Voronoi cells

Relevant vectors



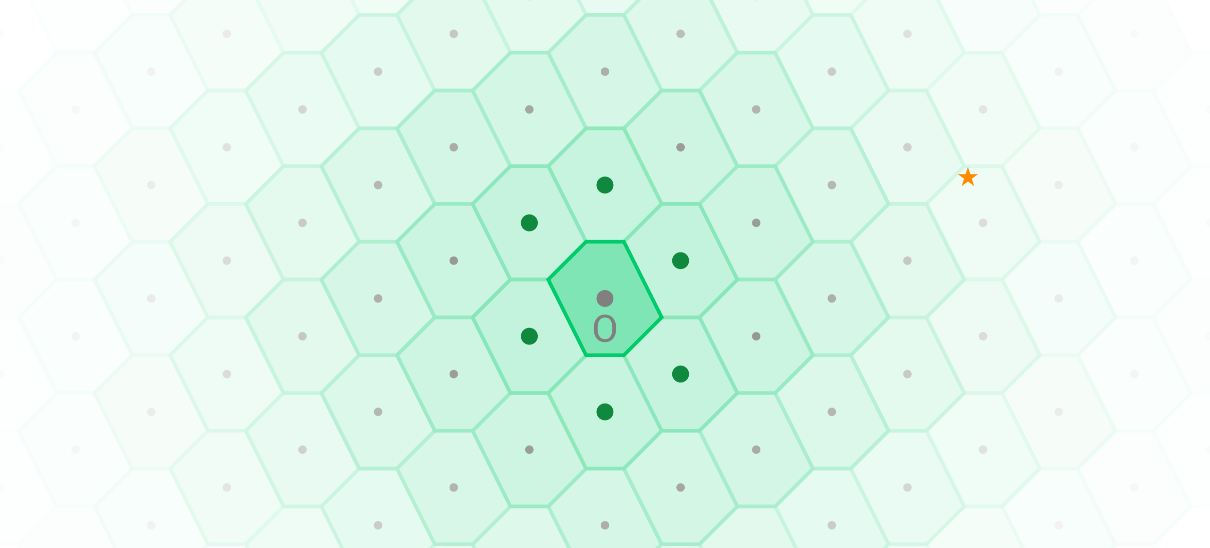
Voronoi cells

Relevant vectors



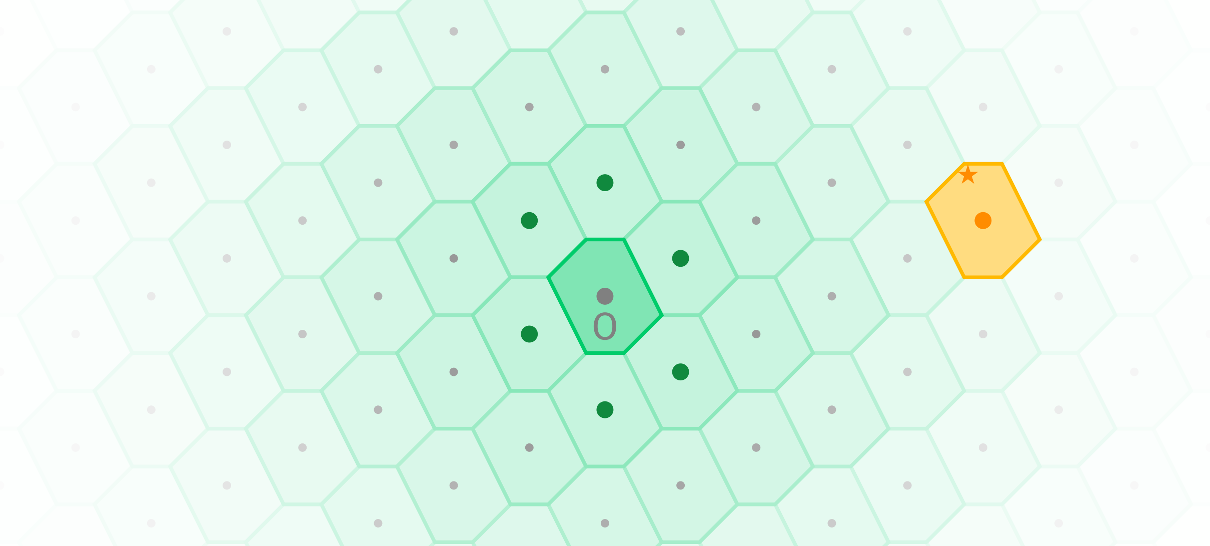
Voronoi cells

Iterative slicer [SFS09]



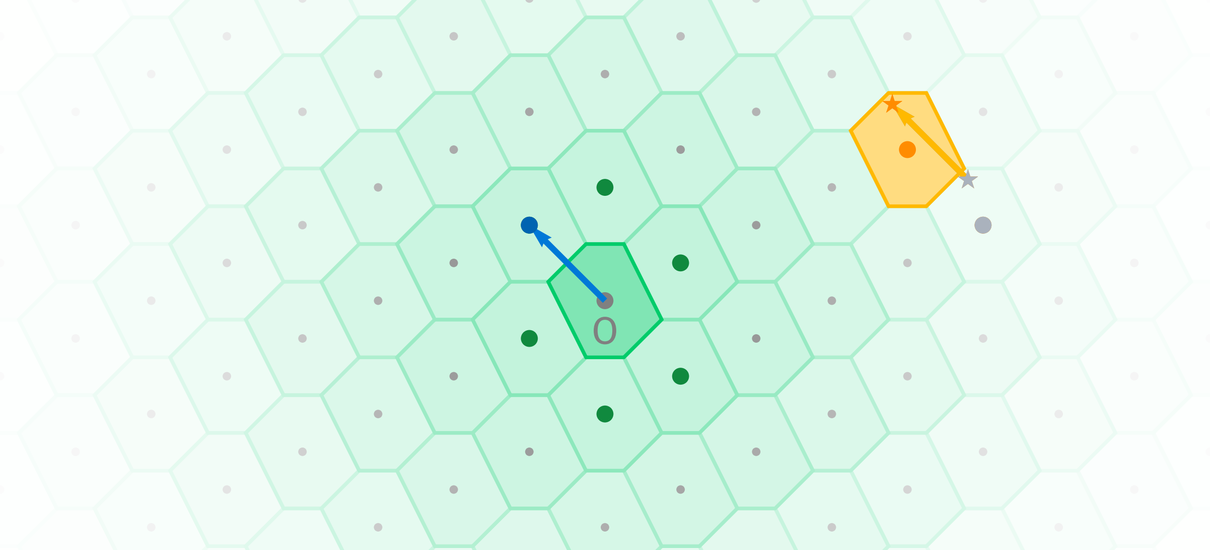
Voronoi cells

Iterative slicer [SFS09]



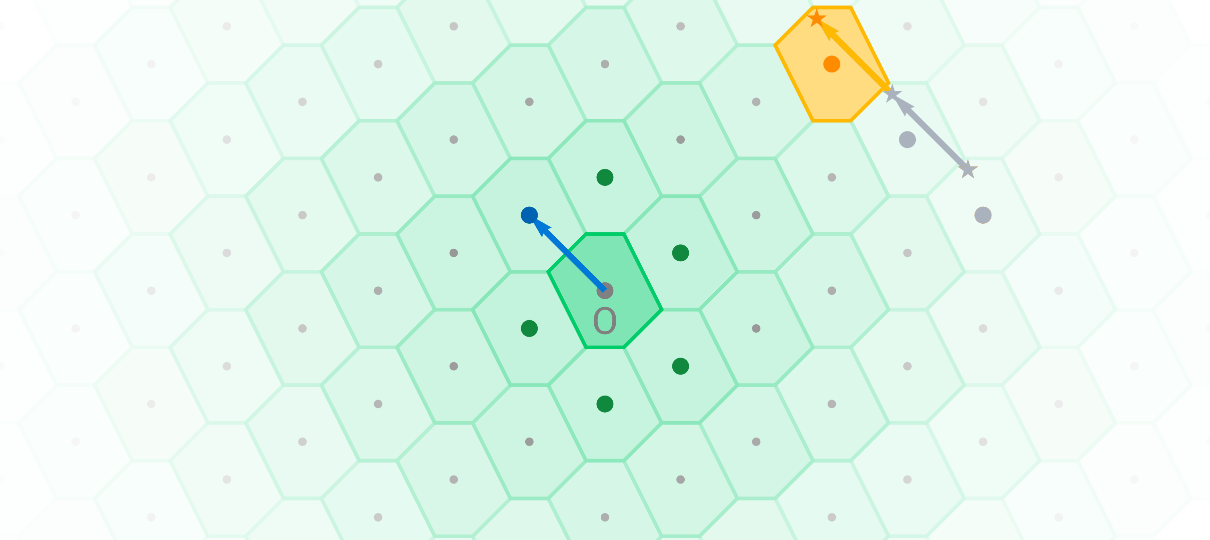
Voronoi cells

Iterative slicer [SFS09]



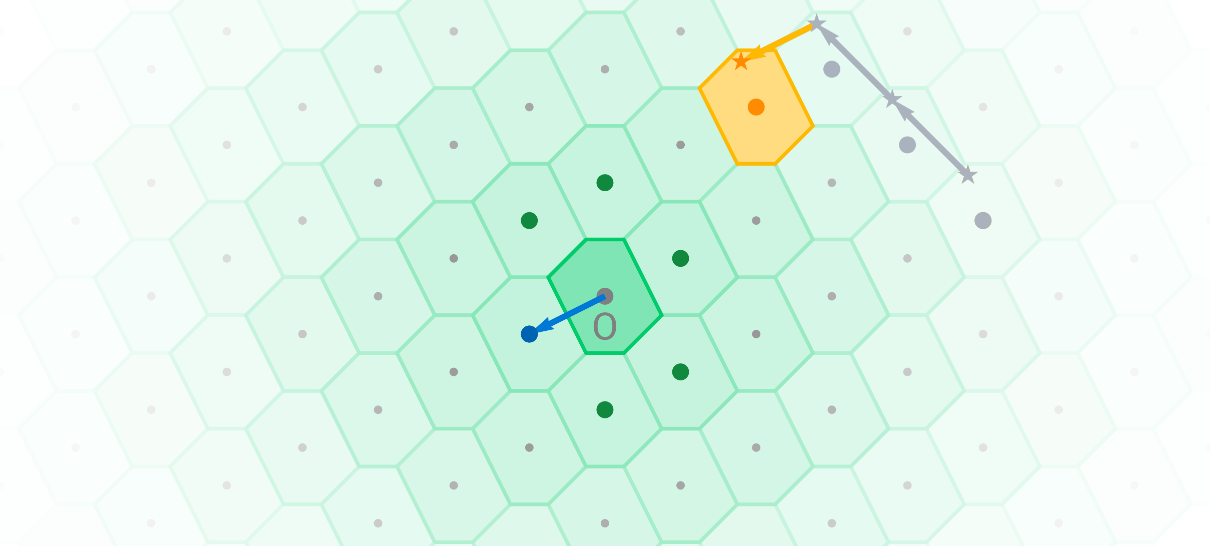
Voronoi cells

Iterative slicer [SFS09]



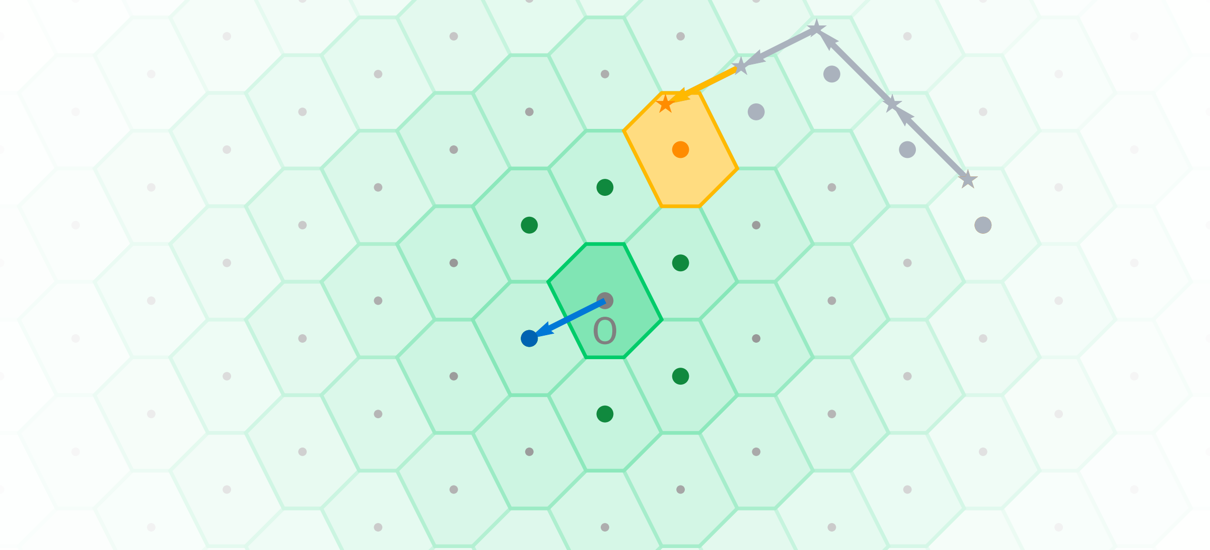
Voronoi cells

Iterative slicer [SFS09]



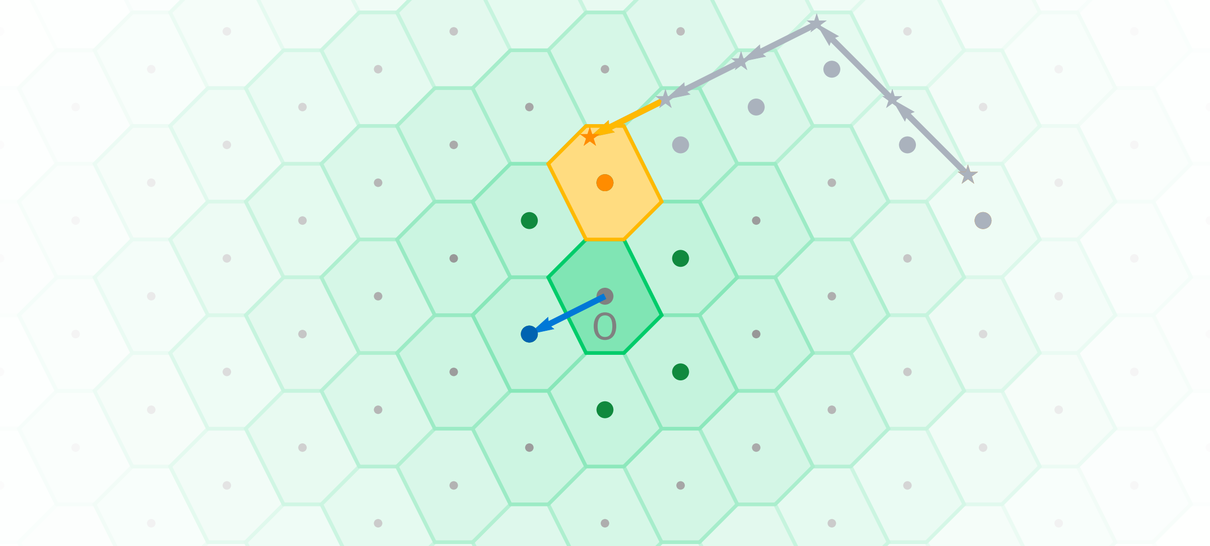
Voronoi cells

Iterative slicer [SFS09]



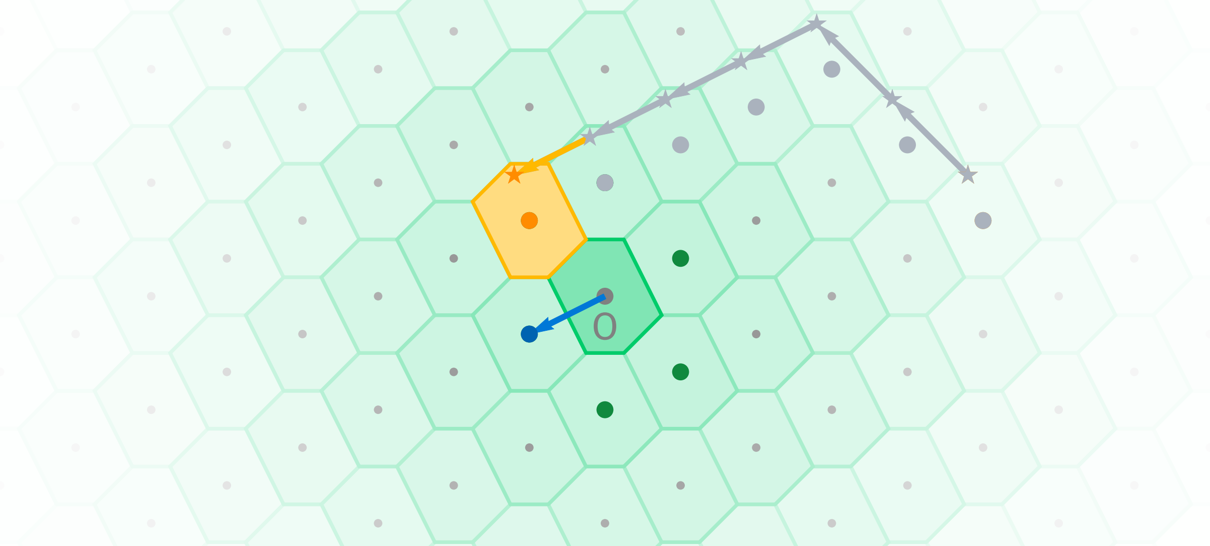
Voronoi cells

Iterative slicer [SFS09]



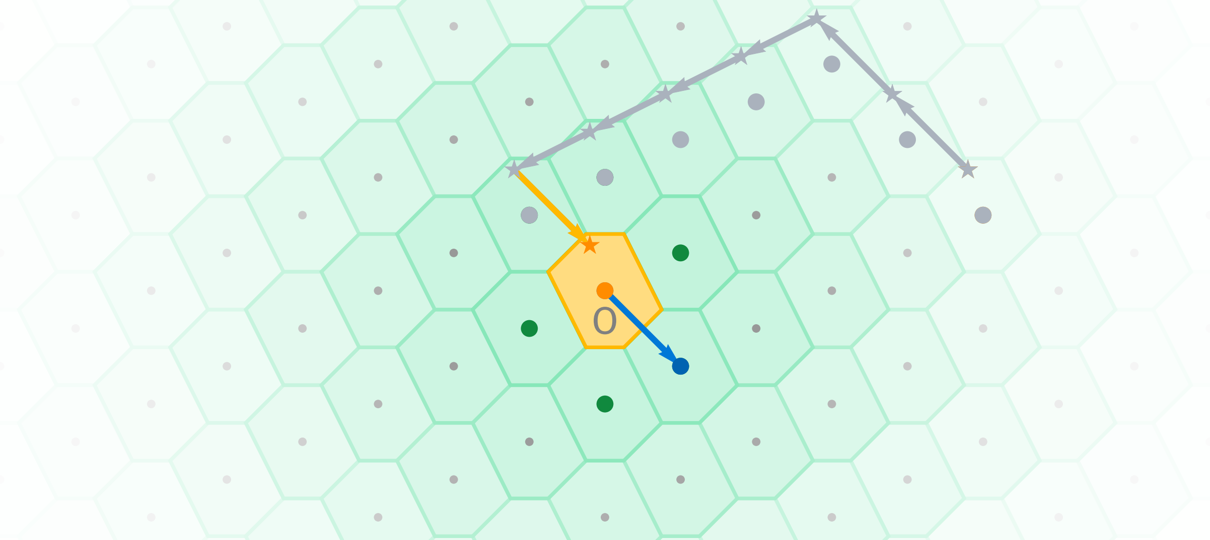
Voronoi cells

Iterative slicer [SFS09]



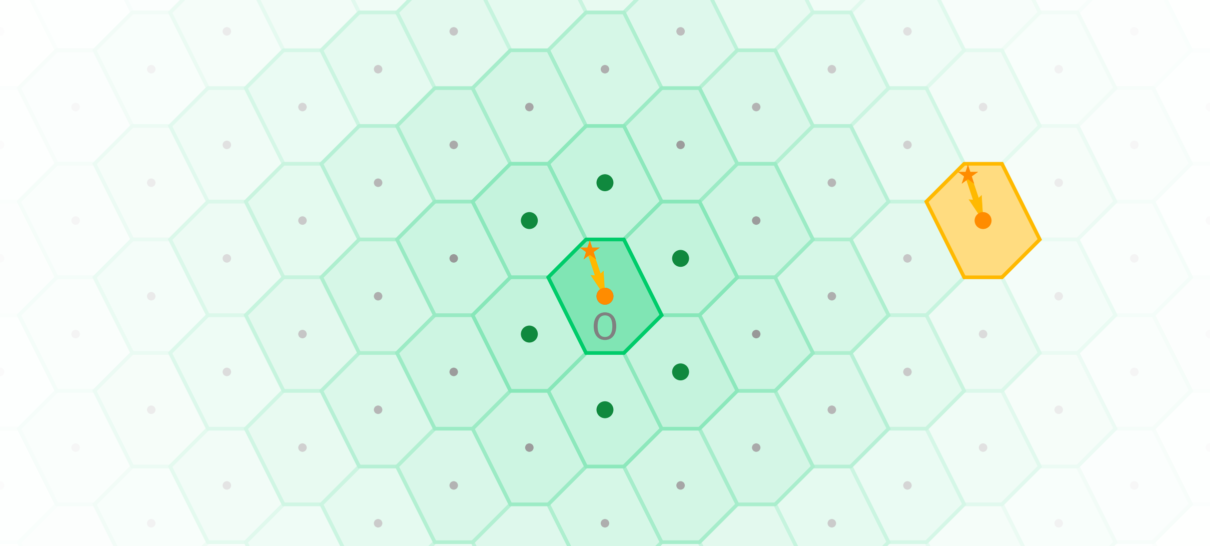
Voronoi cells

Iterative slicer [SFS09]



Voronoi cells

Iterative slicer [SFS09]



Voronoi cells

Overview

- *Preprocessing*: find the relevant vectors ($2^{2n+o(n)}$ time, $2^{n+o(n)}$ space [MV10])

Voronoi cells

Overview

- *Preprocessing*: find the relevant vectors ($2^{2n+o(n)}$ time, $2^{n+o(n)}$ space [MV10])
- *Query*: reduce with the relevant vectors ($2^{n+o(n)}$ time [BD15])

Voronoi cells

Overview

- *Preprocessing*: find the relevant vectors ($2^{2n+o(n)}$ time, $2^{n+o(n)}$ space [MV10])
- *Query*: reduce with the relevant vectors ($2^{n+o(n)}$ time [BD15])
- *Strengths*: provably solves CVPP for arbitrary targets and lattices

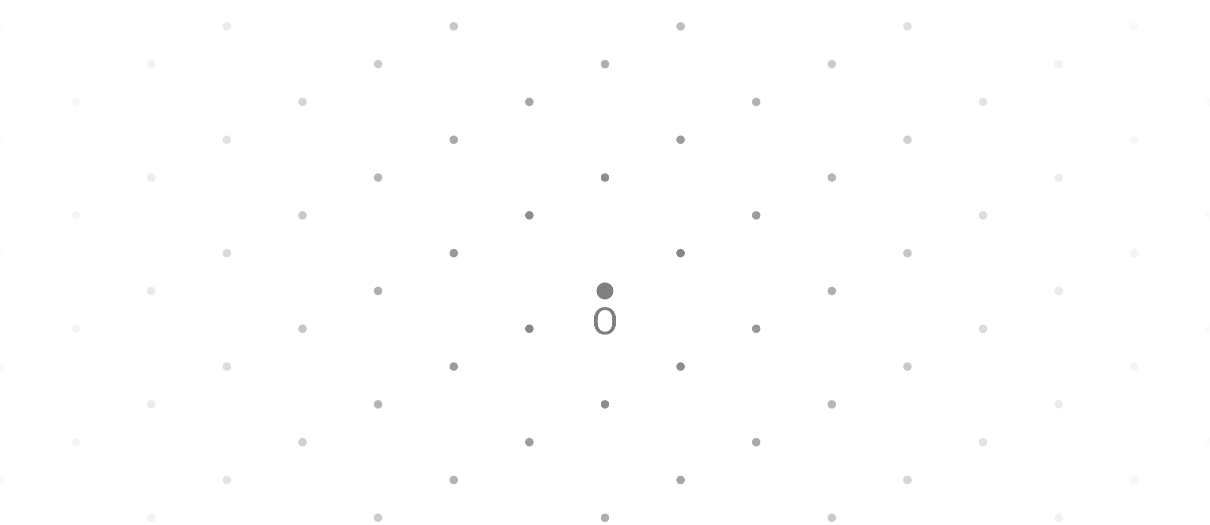
Voronoi cells

Overview

- *Preprocessing*: find the relevant vectors ($2^{2n+o(n)}$ time, $2^{n+o(n)}$ space [MV10])
- *Query*: reduce with the relevant vectors ($2^{n+o(n)}$ time [BD15])
- *Strengths*: provably solves CVPP for arbitrary targets and lattices
- *Limitations*: large time and memory requirements

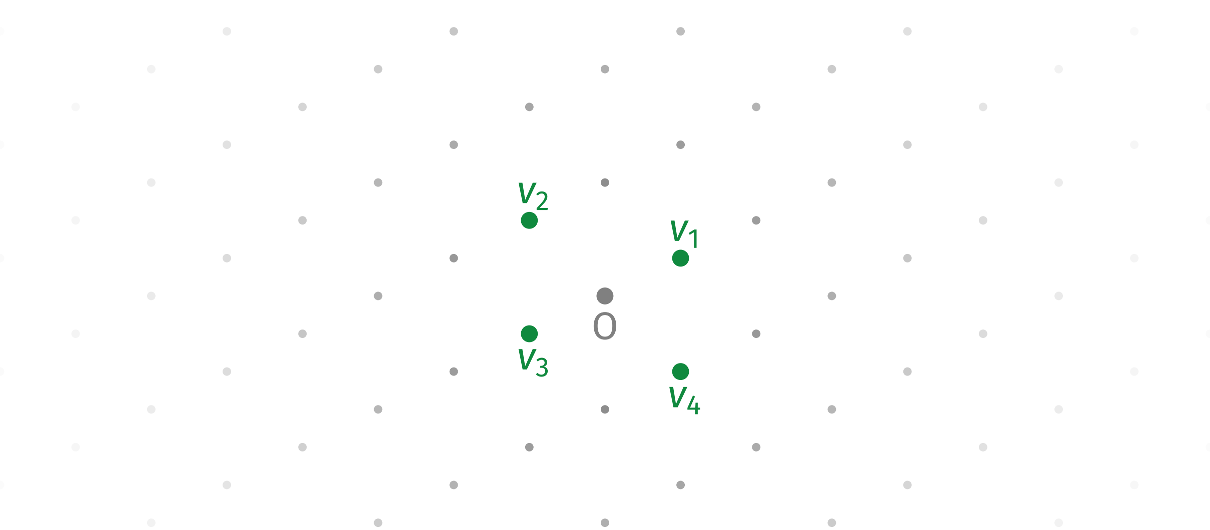
Approximate Voronoi cells

Decrease list size



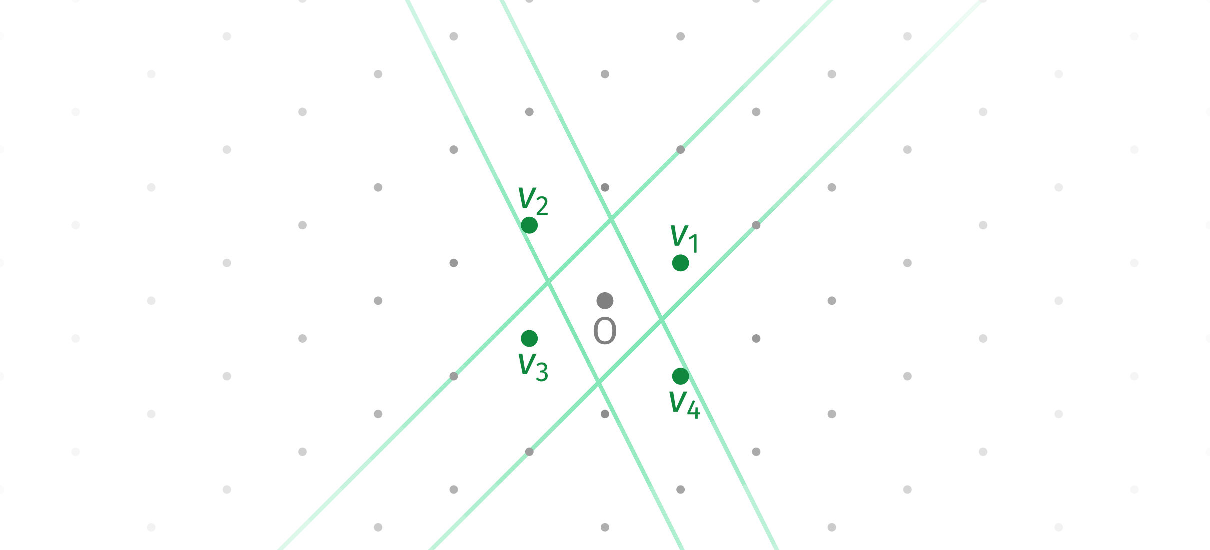
Approximate Voronoi cells

Decrease list size



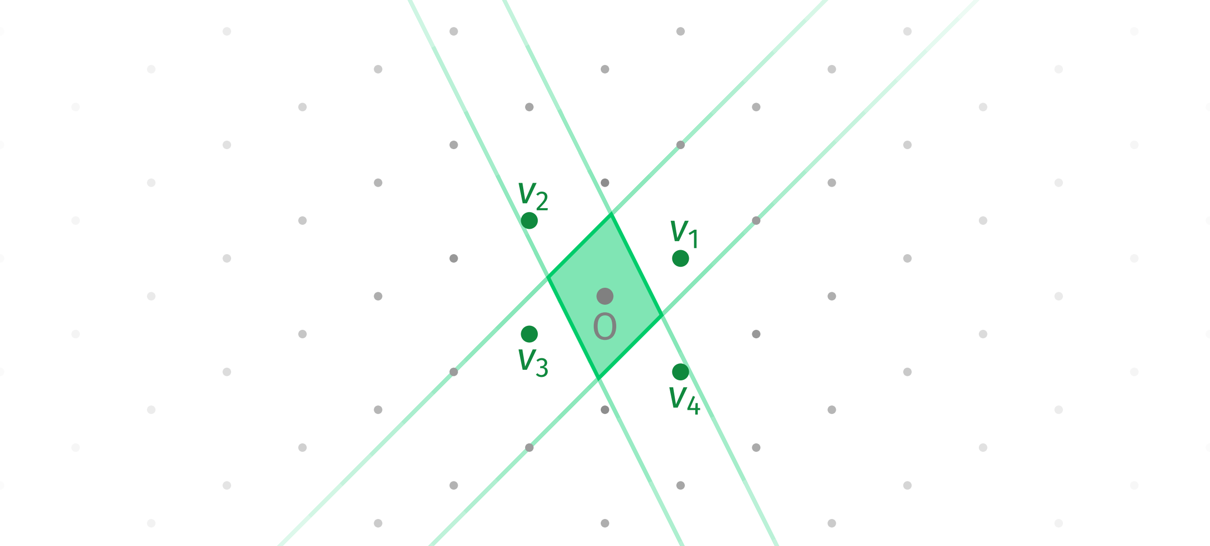
Approximate Voronoi cells

Decrease list size



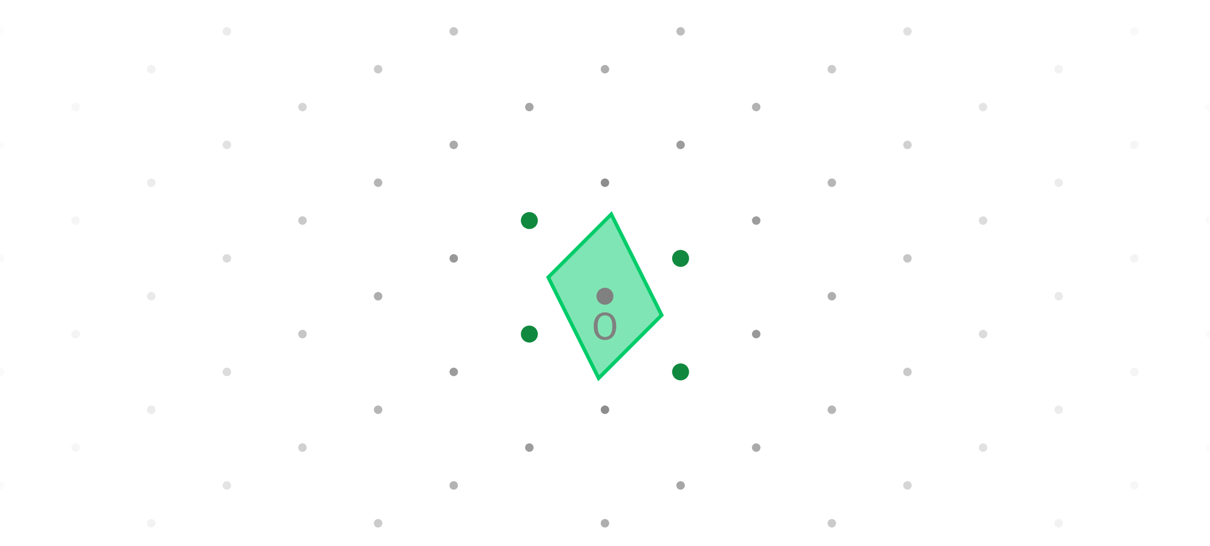
Approximate Voronoi cells

Decrease list size



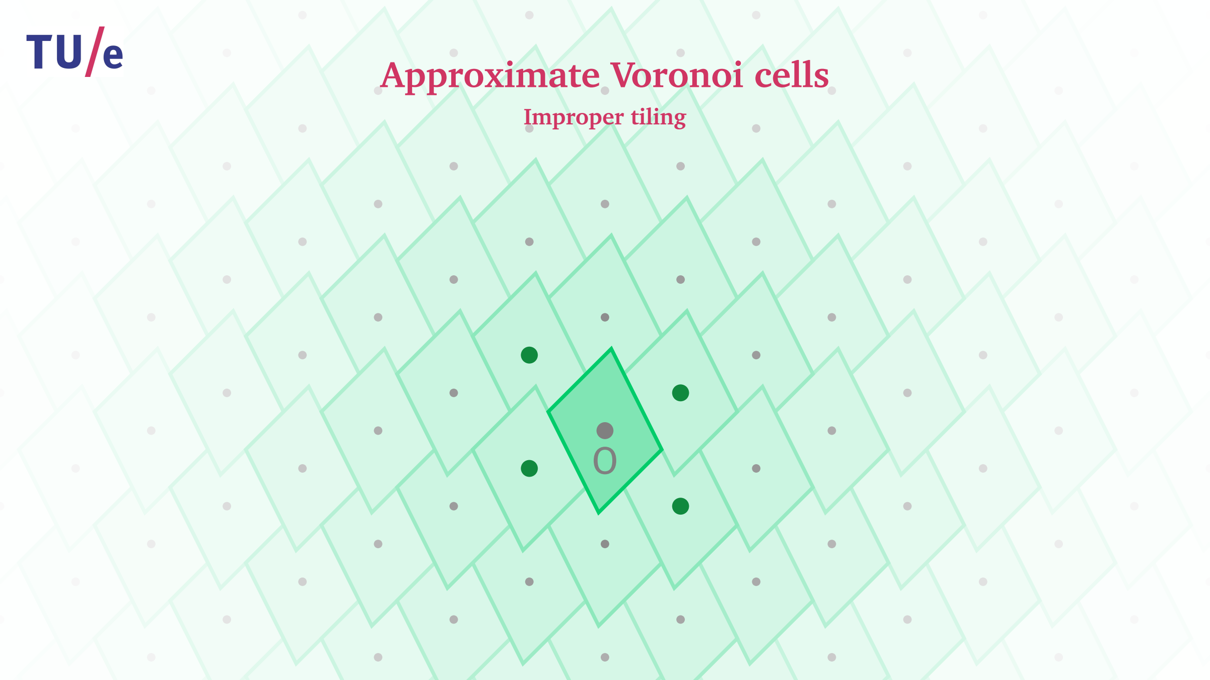
Approximate Voronoi cells

Decrease list size



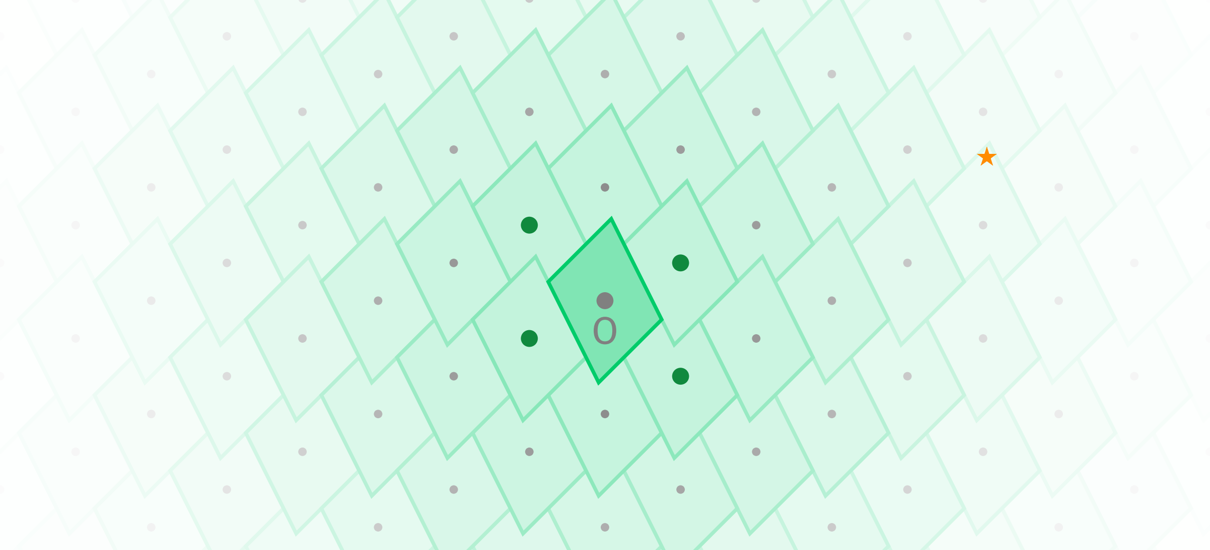
Approximate Voronoi cells

Improper tiling



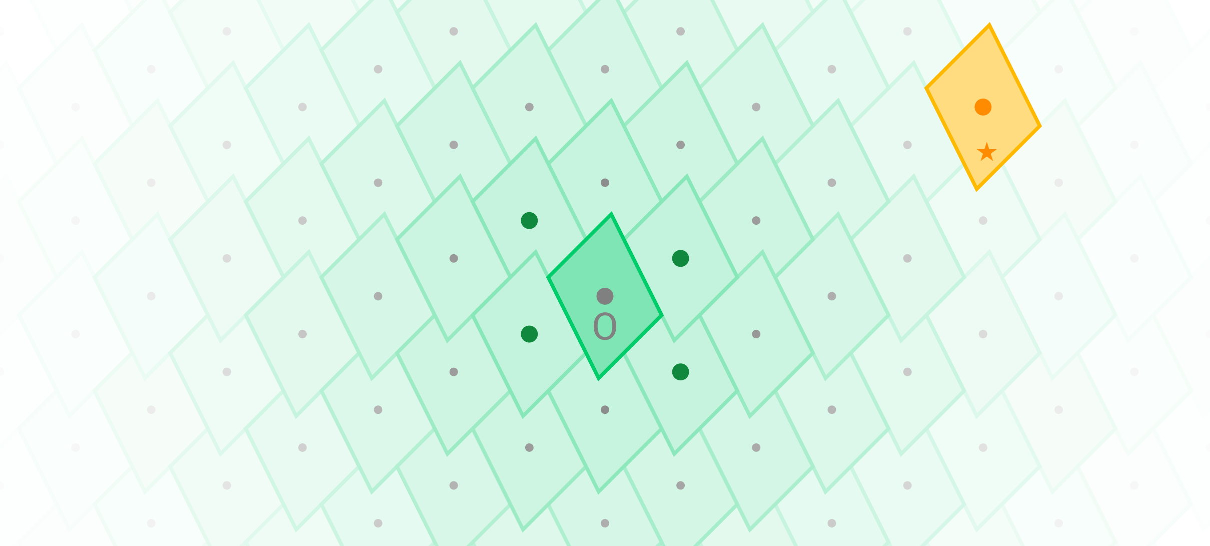
Approximate Voronoi cells

Improper tiling



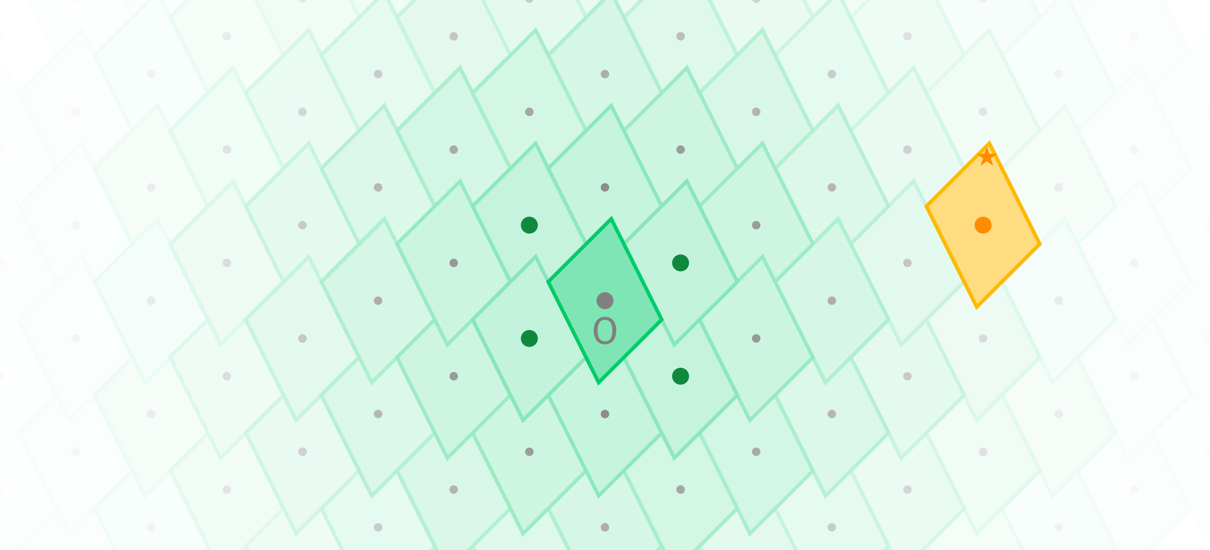
Approximate Voronoi cells

Improper tiling



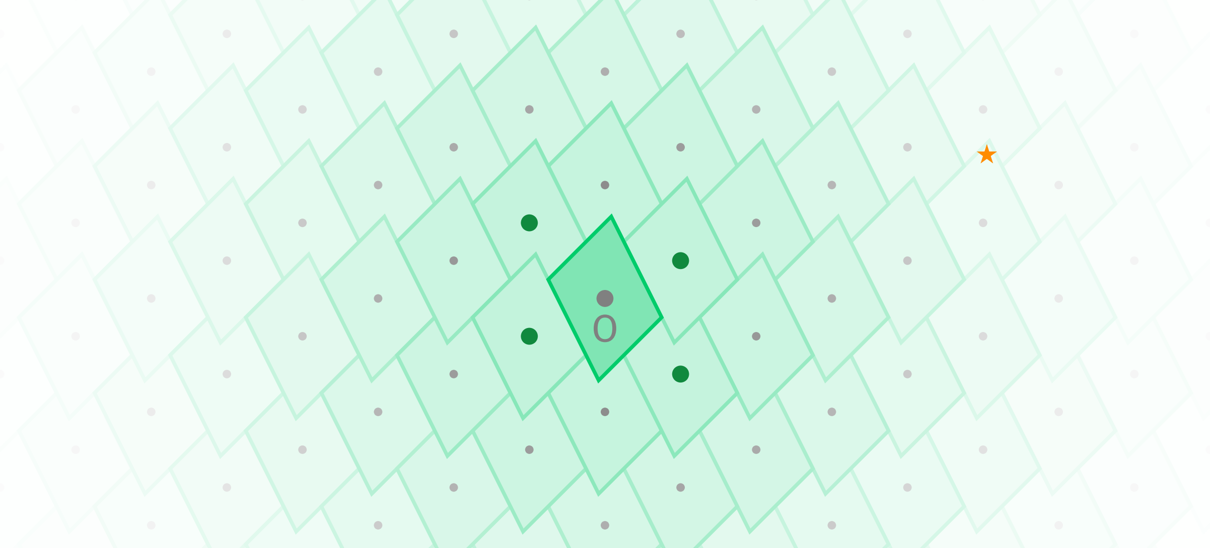
Approximate Voronoi cells

Improper tiling



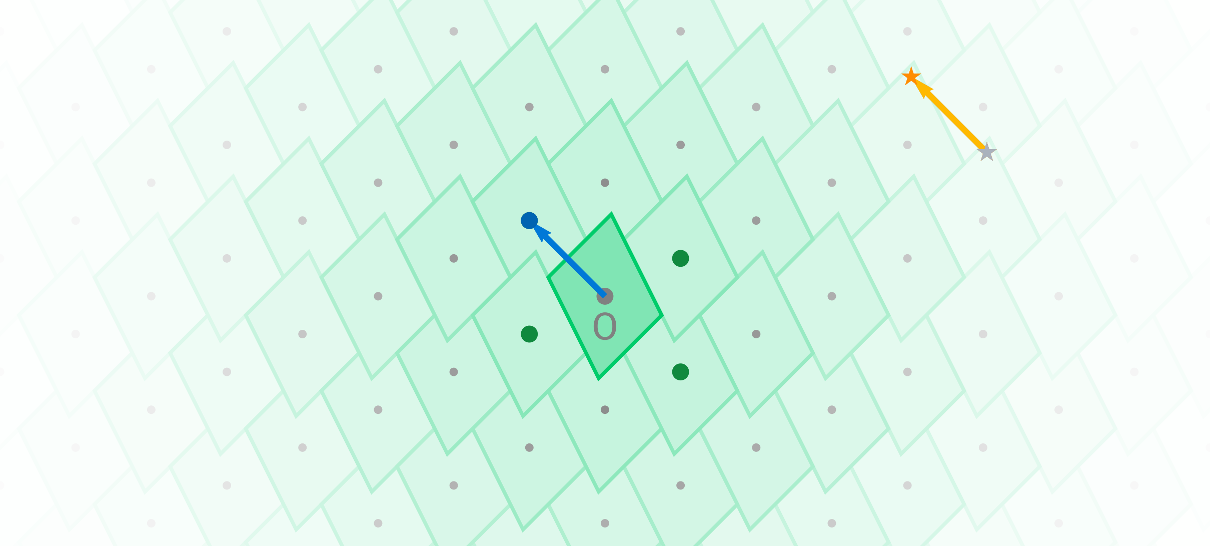
Approximate Voronoi cells

Iterative slicer [SFS09]



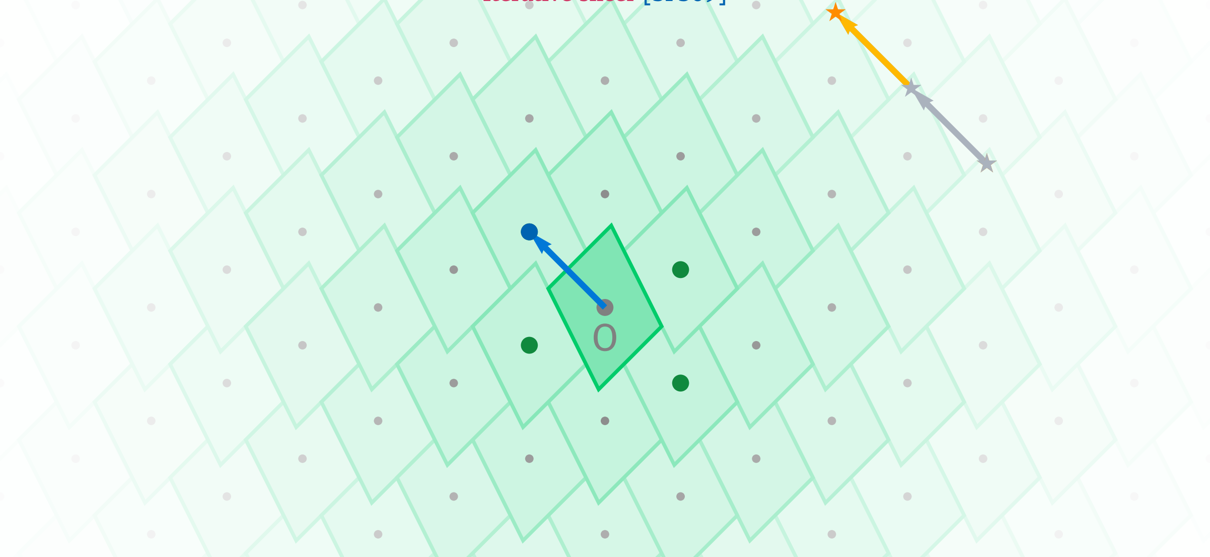
Approximate Voronoi cells

Iterative slicer [SFS09]



Approximate Voronoi cells

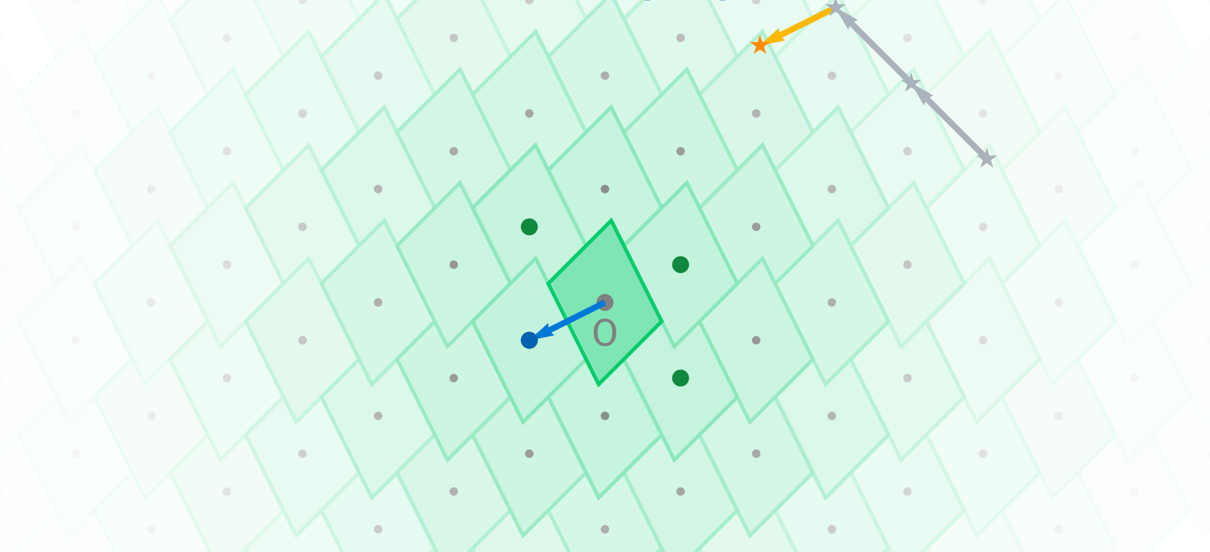
Iterative slicer [SFS09]



TU/e

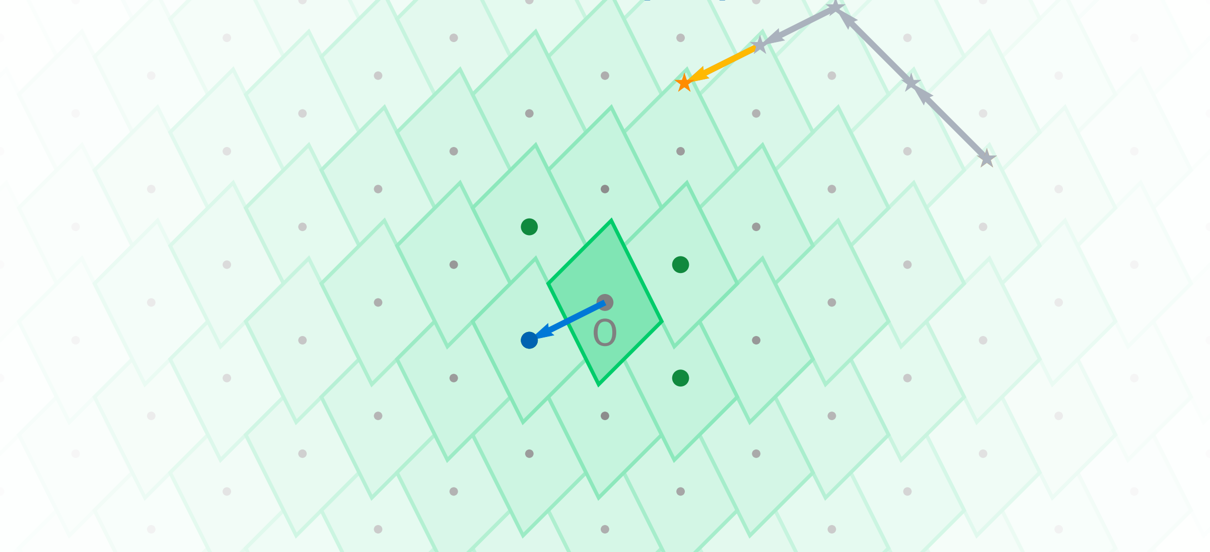
Approximate Voronoi cells

Iterative slicer [SFS09]



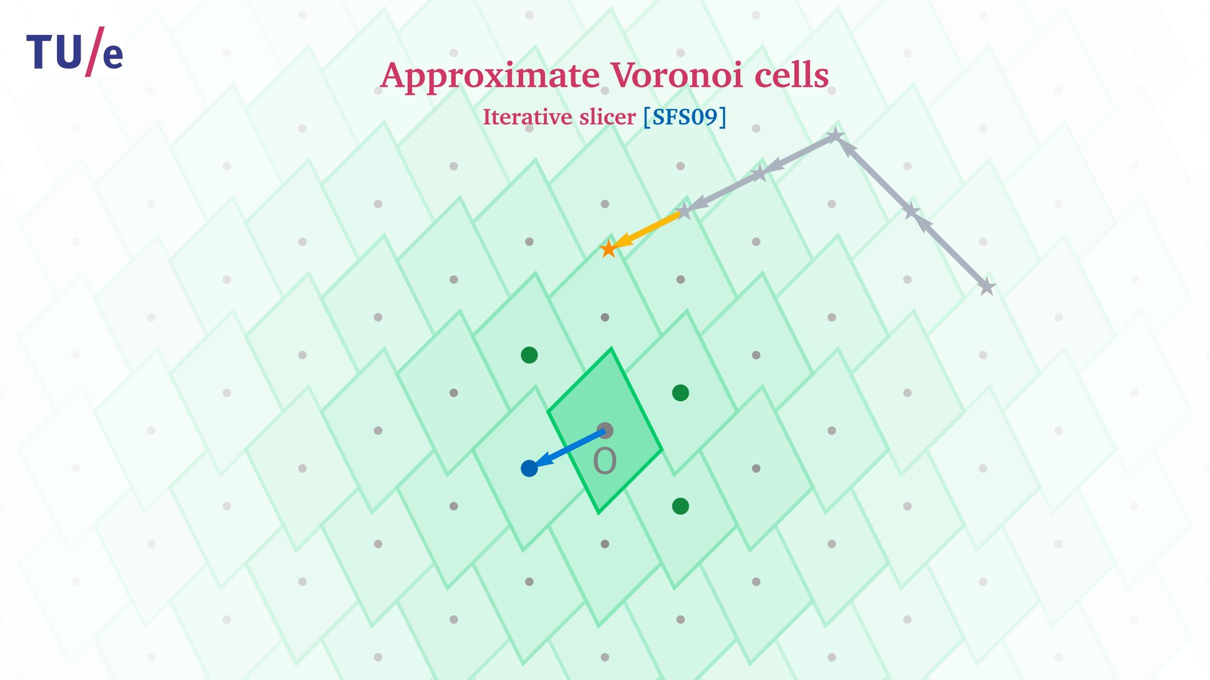
Approximate Voronoi cells

Iterative slicer [SFS09]



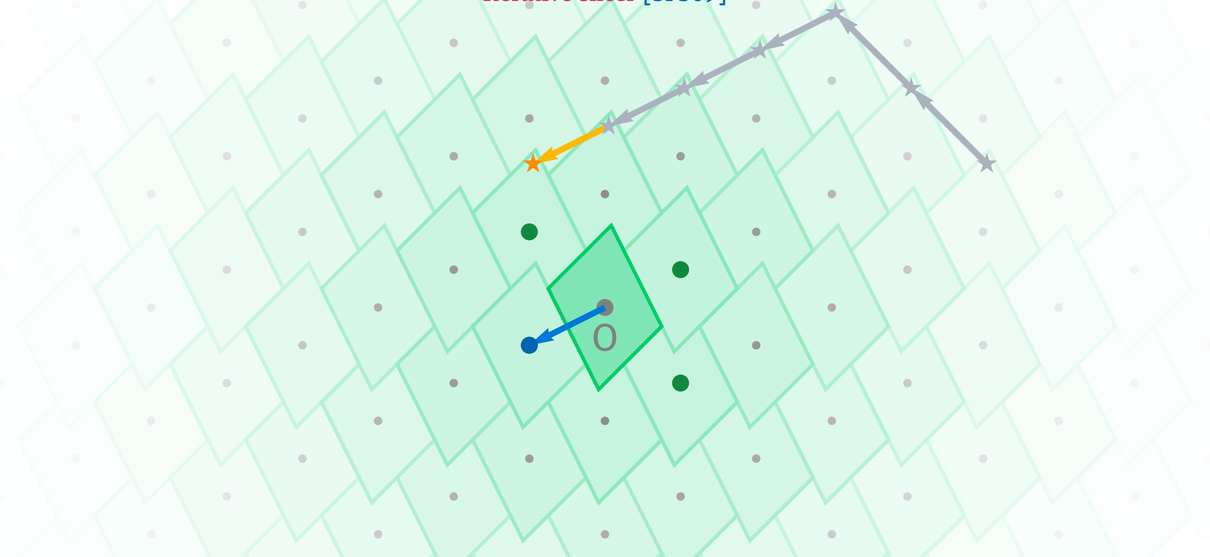
Approximate Voronoi cells

Iterative slicer [SFS09]



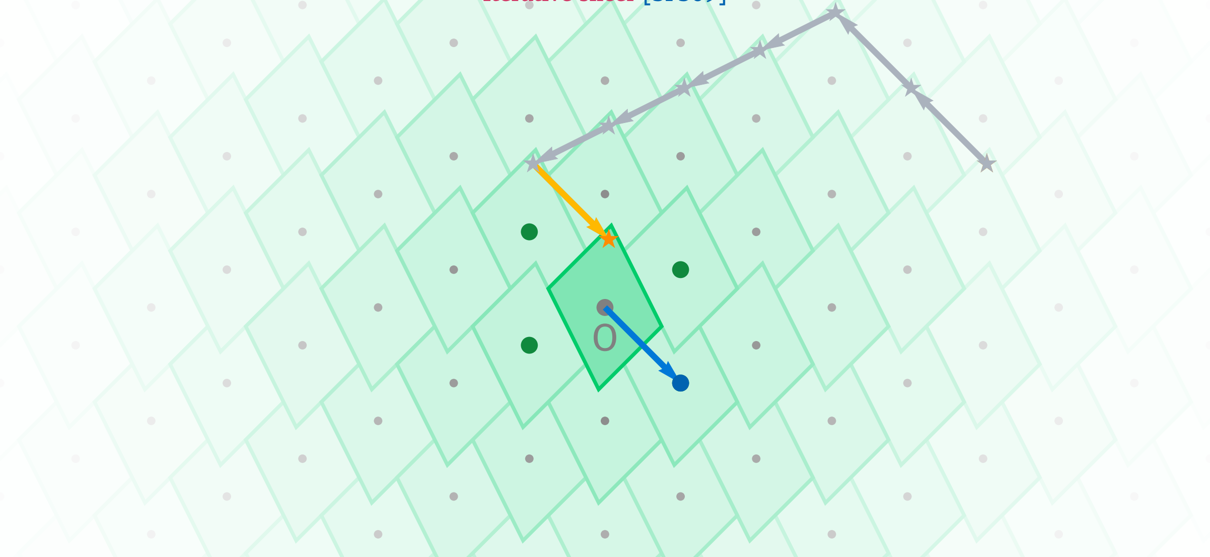
Approximate Voronoi cells

Iterative slicer [SFS09]



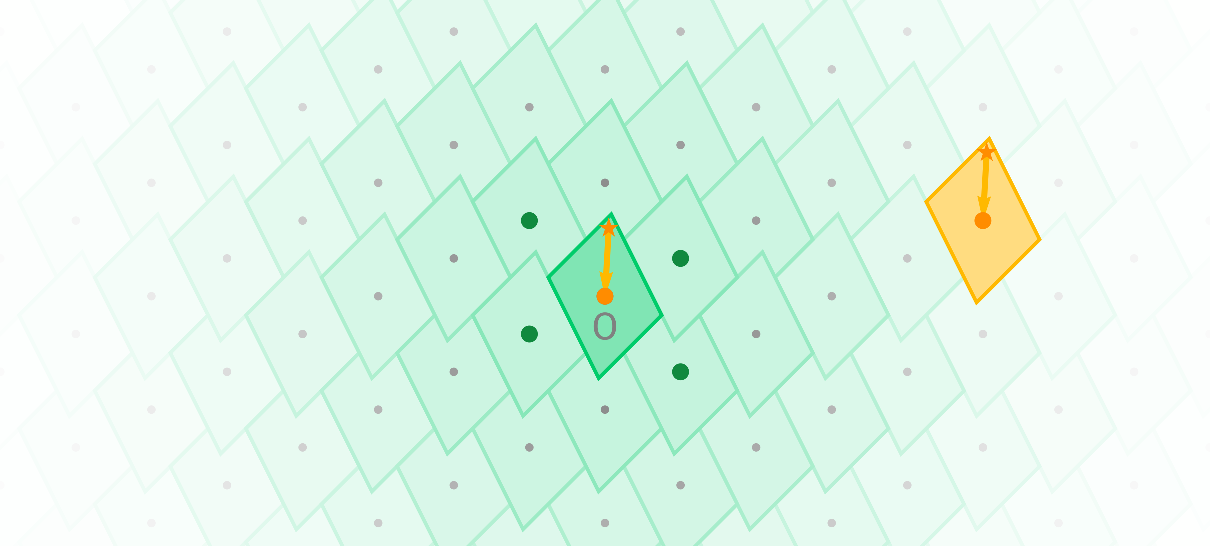
Approximate Voronoi cells

Iterative slicer [SFS09]



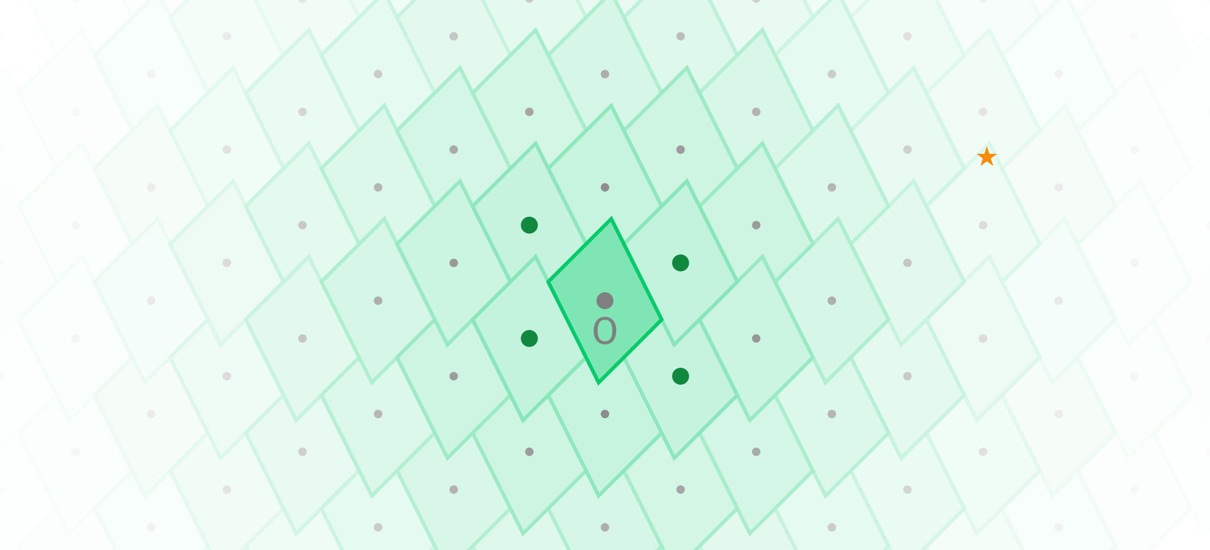
Approximate Voronoi cells

Iterative slicer [SFS09]



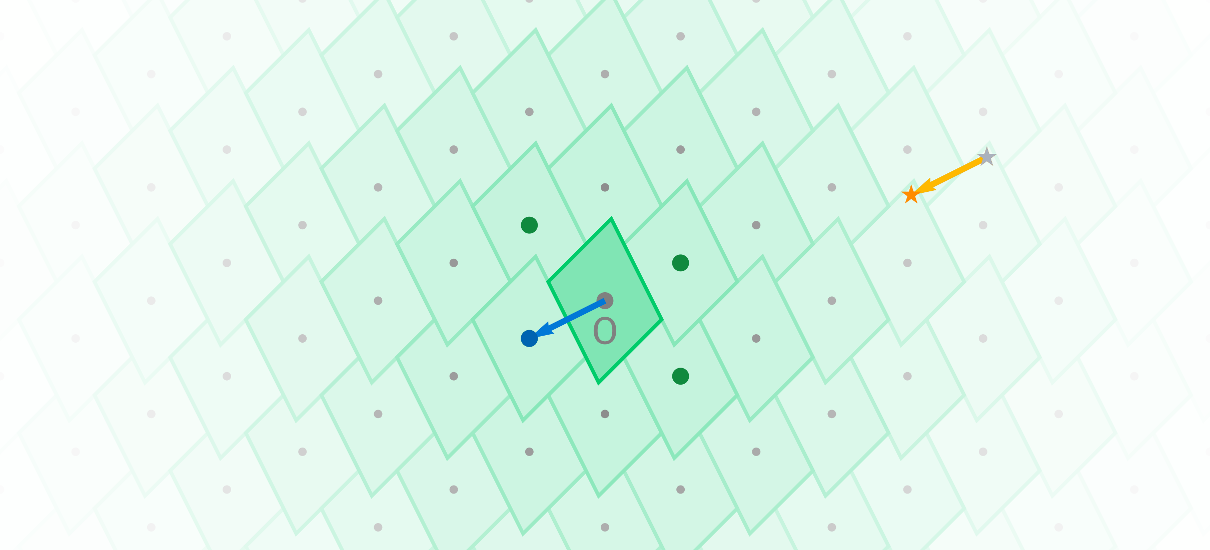
Approximate Voronoi cells

Randomized slicer



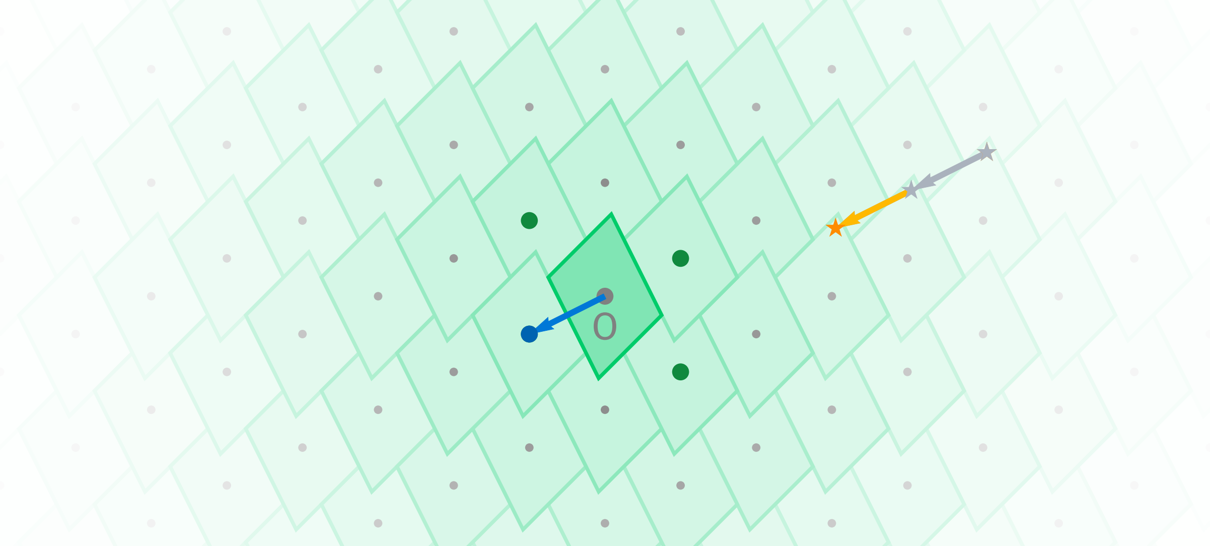
Approximate Voronoi cells

Randomized slicer



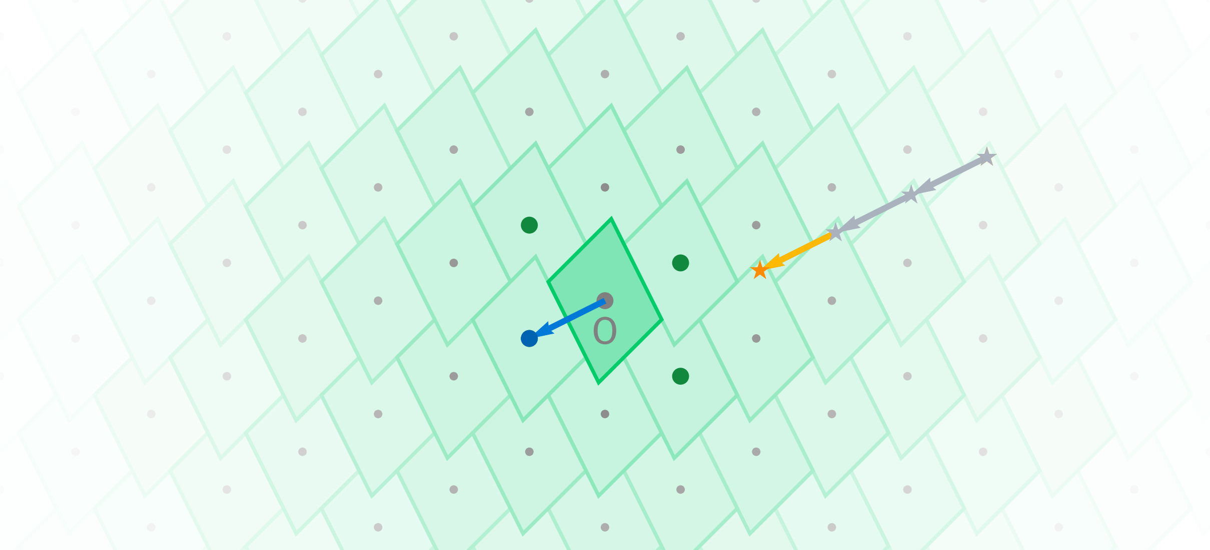
Approximate Voronoi cells

Randomized slicer



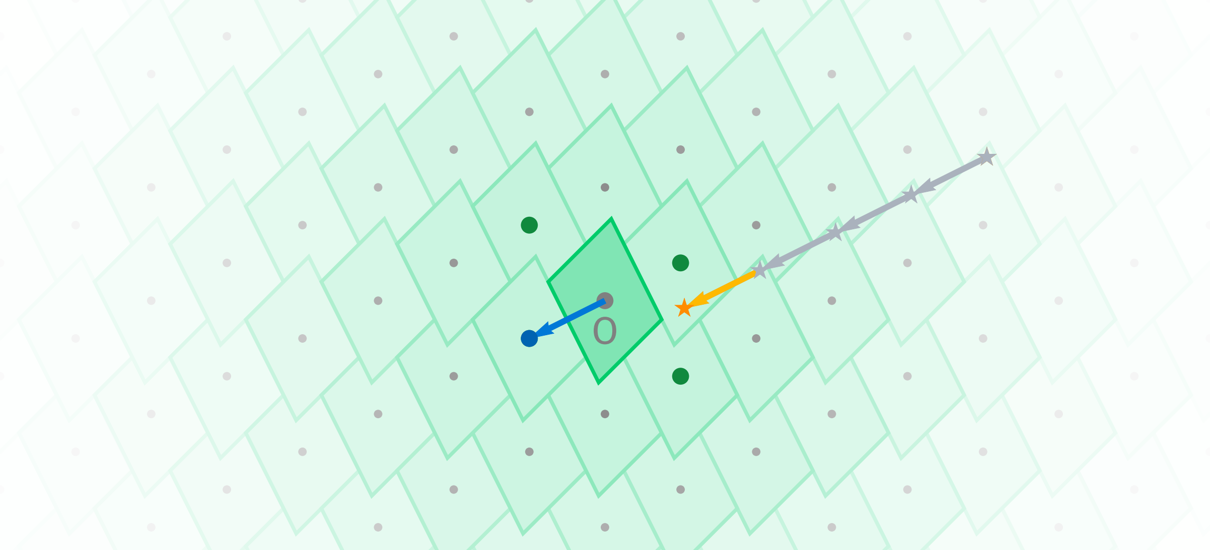
Approximate Voronoi cells

Randomized slicer



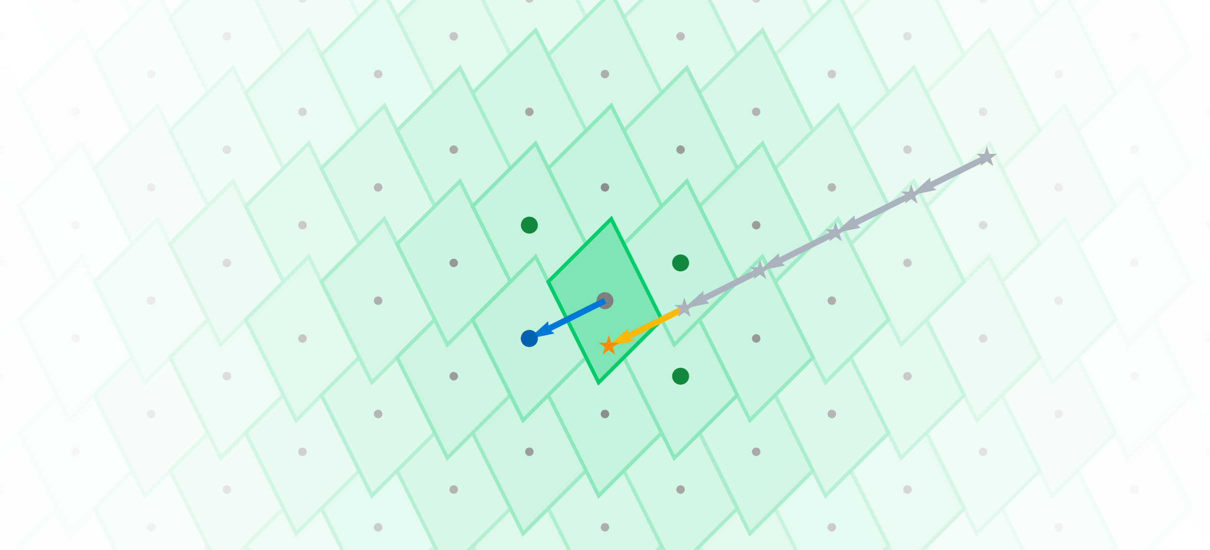
Approximate Voronoi cells

Randomized slicer



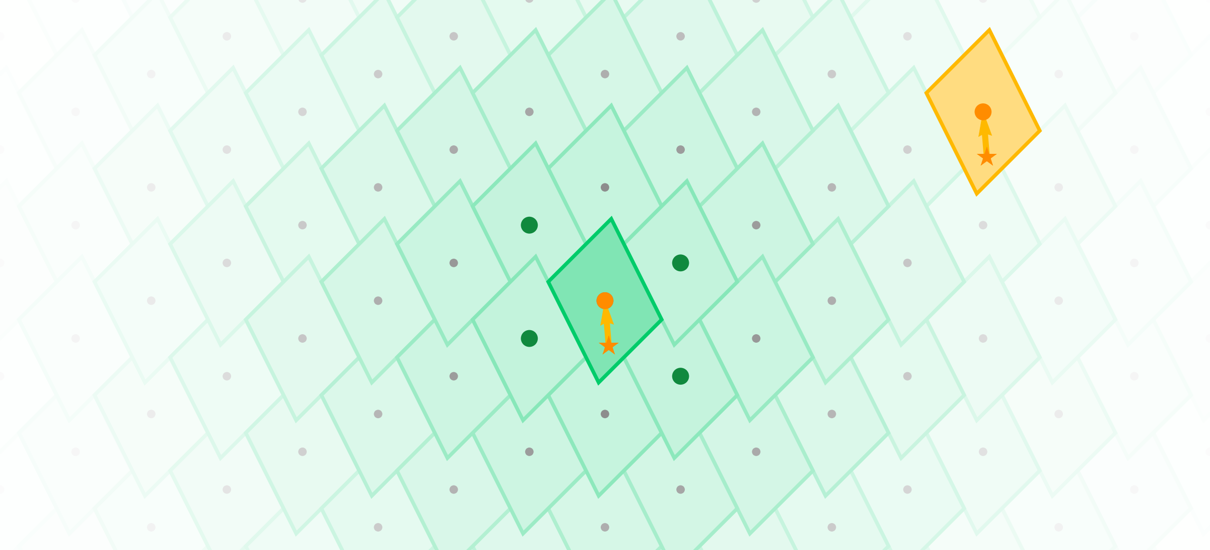
Approximate Voronoi cells

Randomized slicer



Approximate Voronoi cells

Randomized slicer



Approximate Voronoi cells

Estimating the volume [Laa16, DLW19]

Lemma (Good approximations, with heuristics)

Let L consist of the $\alpha^{n+o(n)}$ shortest vectors of a lattice $\mathcal{L}$, with $\alpha \geq \sqrt{2} + o(1)$. Then:

$$\frac{\operatorname{vol}(\mathcal{V}_L)}{\operatorname{vol}(\mathcal{V})} = 1 + o(1). \tag{1}$$

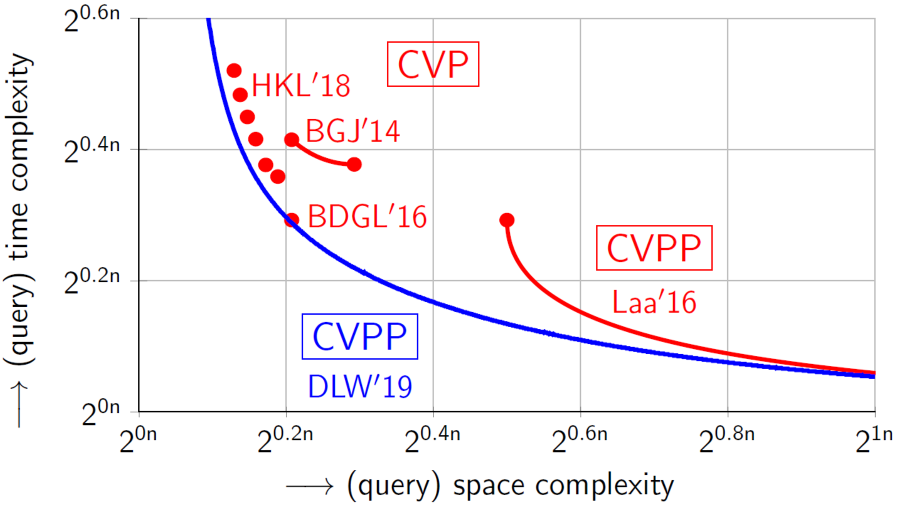
Lemma (Arbitrary approximations, with heuristics)

Let L consist of the $\alpha^{n+o(n)}$ shortest vectors of a lattice $\mathcal{L}$, with $\alpha \in (1.03396, \sqrt{2})$. Then:

$$\frac{\operatorname{vol}(\mathcal{V}_L)}{\operatorname{vol}(\mathcal{V})} \leq \left(\frac{16\alpha^4 \left(\alpha^2 - 1\right)}{-9\alpha^8 + 64\alpha^6 - 104\alpha^4 + 64\alpha^2 - 16} \right)^{n/2+o(n)}. \tag{2}$$

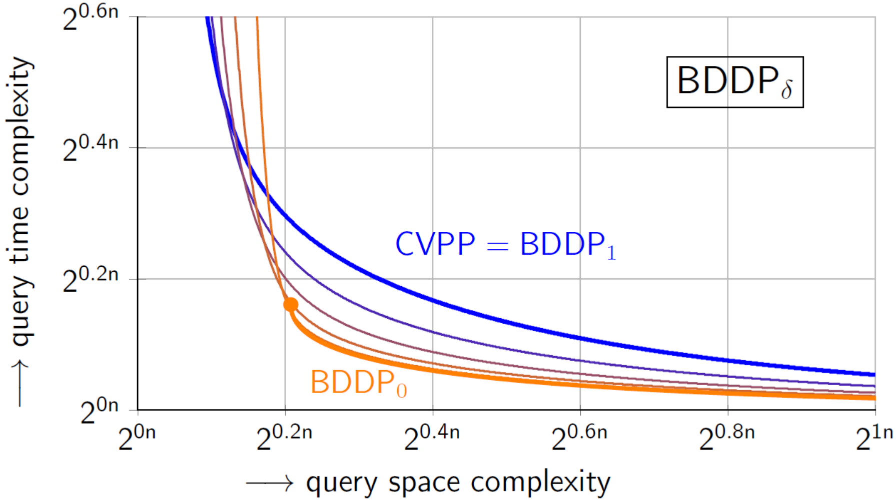
Approximate Voronoi cells

Results for CVPP



Approximate Voronoi cells

Results for BDDP



Approximate Voronoi cells

Overview

- *Preprocessing*: find many short vectors ($2^{O(n)}$ time, $2^{O(n)}$ space)

Approximate Voronoi cells

Overview

- *Preprocessing*: find many short vectors ($2^{O(n)}$ time, $2^{O(n)}$ space)
- *Query*: (randomized) reduction with short vectors ($2^{O(n)}$ time [Laa16, DLW19])

Approximate Voronoi cells

Overview

- *Preprocessing*: find many short vectors ($2^{O(n)}$ time, $2^{O(n)}$ space)
- *Query*: (randomized) reduction with short vectors ($2^{O(n)}$ time [Laa16, DLW19])
- *Strengths*: efficient method for hard CVPP instances

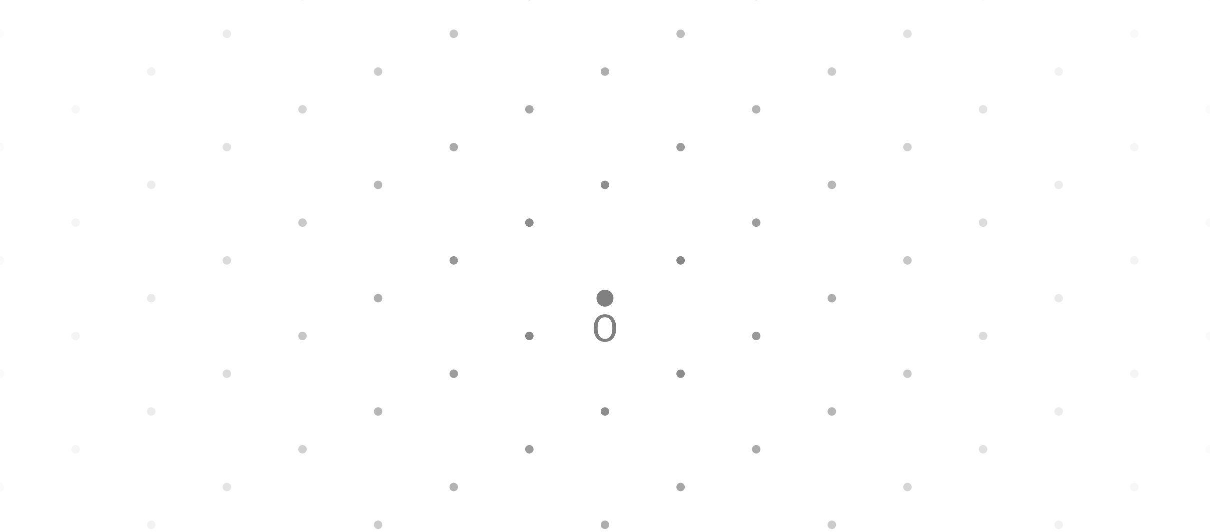
Approximate Voronoi cells

Overview

- *Preprocessing*: find many short vectors ($2^{O(n)}$ time, $2^{O(n)}$ space)
- *Query*: (randomized) reduction with short vectors ($2^{O(n)}$ time [Laa16, DLW19])
- *Strengths*: efficient method for hard CVPP instances
- *Limitations*: does not scale well for BDDP instances

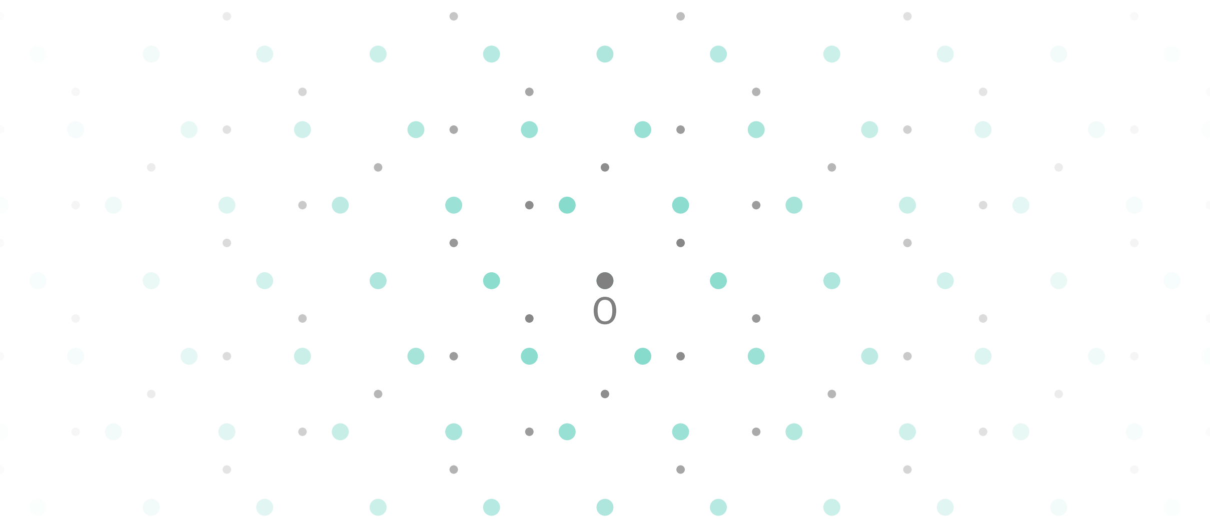
Dual approach

Dual lattices



Dual approach

Dual lattices



Dual approach

Distinguisher

$\mathcal{L}^* = \{\boldsymbol{x} \in \mathbb{R}^n : \langle \boldsymbol{x}, \boldsymbol{v} \rangle \in \mathbb{Z}, \forall\, \boldsymbol{v} \in \mathcal{L}\}$

- Primal target vector $\boldsymbol{t} = \boldsymbol{v} + \boldsymbol{e}$ with $\boldsymbol{v} \in \mathcal{L}$
- Short dual vector $\boldsymbol{v}^* \in \mathcal{L}^*$
- Distinguisher:

$$\begin{cases} \langle \boldsymbol{t}, \boldsymbol{v}^* \rangle \mod 1 = 0 & \text{if } \|\boldsymbol{e}\| = 0; \\ \langle \boldsymbol{t}, \boldsymbol{v}^* \rangle \mod 1 \approx 0 & \text{if } \|\boldsymbol{e}\| \approx 0 \text{ and } \|\boldsymbol{v}^*\| \text{ small}; \\ \langle \boldsymbol{t}, \boldsymbol{v}^* \rangle \mod 1 \sim U(-\frac{1}{2}, \frac{1}{2}) & \text{if } \|\boldsymbol{e}\| \gg 0. \end{cases}$$

Dual approach

Overview

- *Preprocessing*: find many short dual vectors ($2^{O(n)}$ time, $2^{O(n)}$ space)

Dual approach

Overview

- *Preprocessing*: find many short dual vectors ($2^{O(n)}$ time, $2^{O(n)}$ space)
- *Query*: distinguish based on dot products modulo 1

Dual approach

Overview

- *Preprocessing*: find many short dual vectors ($2^{O(n)}$ time, $2^{O(n)}$ space)
- *Query*: distinguish based on dot products modulo 1
- *Strengths*: smooth trade-offs for BDDP

0

Dual approach

Overview

- *Preprocessing*: find many short dual vectors ($2^{O(n)}$ time, $2^{O(n)}$ space)
- *Query*: distinguish based on dot products modulo 1
- *Strengths*: smooth trade-offs for BDDP
- *Limitations*: traditionally only solves decisional BDD(P)

Conclusion

Summary

Babai's algorithms

- Fast and simple algorithms
- Targets must lie close to the lattice

Voronoi cells

- Provable, deterministic algorithm
- Requires $2^{n+o(n)}$ time and space

Approximate Voronoi cells

- Heuristic alternative to exact Voronoi cells
- Nearest neighbor speed-ups
- Does not scale well for BDDP

Dual approach

- Distinguisher using short dual vectors
- Works better when target is somewhat close to lattice
- Traditionally only solves decisional problem

Conclusion

Open problems / Work in progress

Approximate Voronoi cells

- Eliminate lower bound on space complexity
- Improve upper bound on volume ratio
- Apply other nearest neighbor techniques

Dual approach

- Analyze method heuristically
- Efficient conversion to search-CVPP
- Find cross-over point with other methods