

# Approximate Voronoi cells for lattices, revisited

Thijs Laarhoven

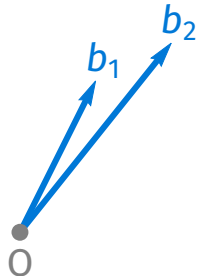
`mail@thijs.com`

`http://www.thijs.com/`

MathCrypt 2019, Santa Barbara, USA  
(August 18, 2019)

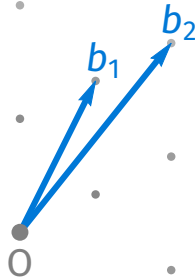
# Lattices

## Basics



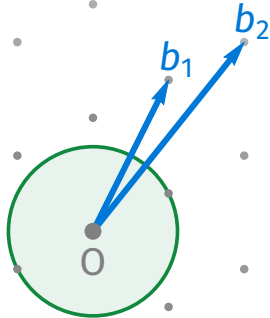
# Lattices

## Basics



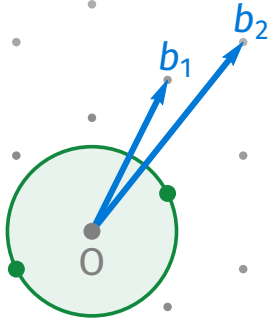
# Lattice problems

## Shortest Vector Problem (SVP)



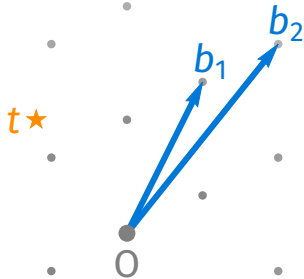
# Lattice problems

## Shortest Vector Problem (SVP)



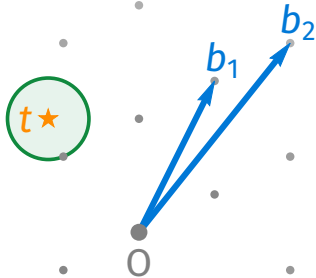
# Lattice problems

## Closest Vector Problem (CVP)



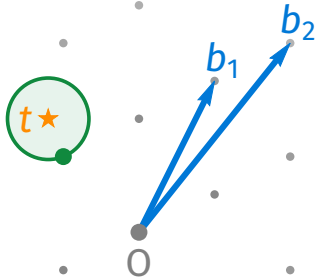
# Lattice problems

## Closest Vector Problem (CVP)



# Lattice problems

## Closest Vector Problem (CVP)





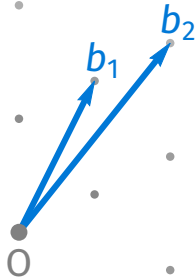
# Lattice problems

## Asymptotics for SVP and CVP

	Algorithm	$\log_2(\text{Time})$	$\log_2(\text{Space})$	Experiments
Worst-case SVP	Enumeration [Poh81, Kan83, ..., MW15, AN17]	$O(n \log n)$	$O(\log n)$	<b>152</b>
	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]	$1.256n$	<b>0.500n</b>	–
	Discrete Gaussian sampling [ADRS15, ADS15, AS18]	<b>1.000n</b>	$1.000n$	–
Average-case SVP	The Nguyen–Vidick sieve [NV08]	$0.415n$	$0.208n$	50
	GaussSieve [MV10, ..., IKMT14, BNvdP16, KYC17]	$0.415n$	$0.208n$	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]	$0.359n$	<b>0.189n</b>	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, DSvW19]	<b>0.292n</b>	$0.292n$	<b>157</b>
	Quantum NNS sieve [LMP15, Laa16]	<b>0.265n</b>	$0.265n$	–

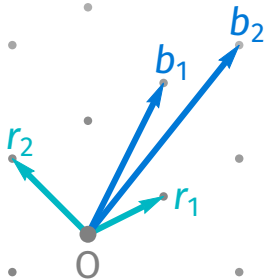
# Lattice problems

## Closest Vector Problem with Preprocessing (CVPP)



# Lattice problems

## Closest Vector Problem with Preprocessing (CVPP)



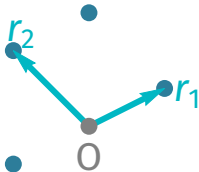
# Lattice problems

## Closest Vector Problem with Preprocessing (CVPP)



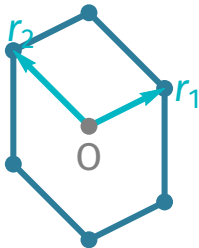
# Lattice problems

## Closest Vector Problem with Preprocessing (CVPP)



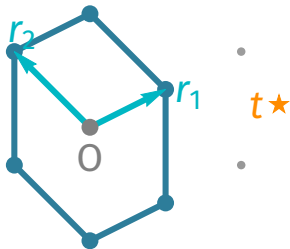
# Lattice problems

## Closest Vector Problem with Preprocessing (CVPP)



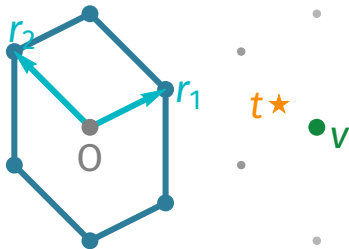
# Lattice problems

## Closest Vector Problem with Preprocessing (CVPP)



# Lattice problems

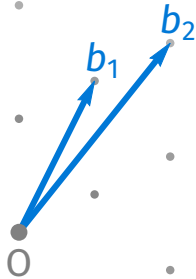
## Closest Vector Problem with Preprocessing (CVPP)





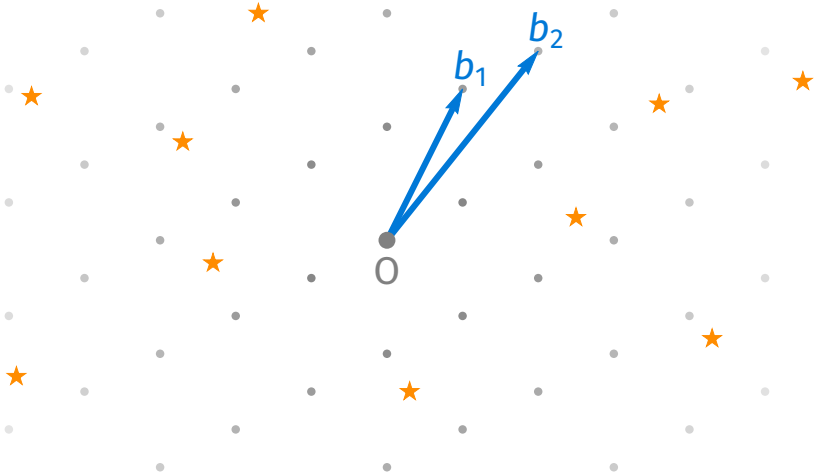
# Lattice problems

## Batch Closest Vector Problem



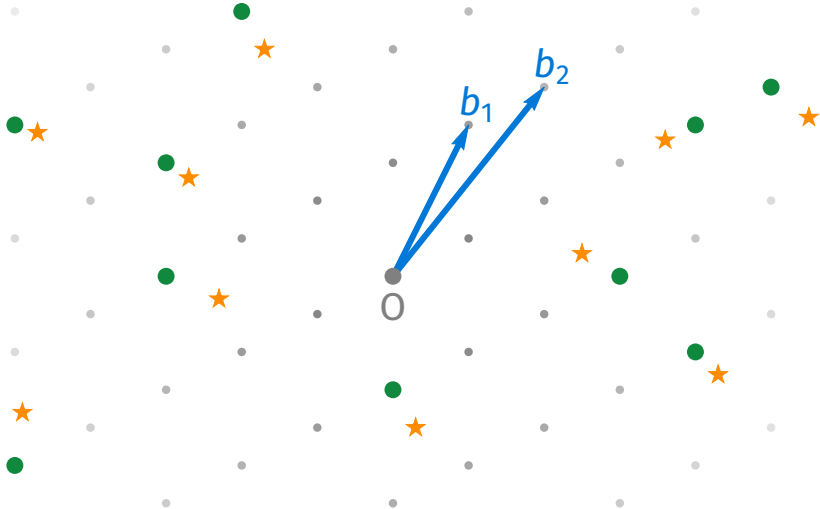
# Lattice problems

## Batch Closest Vector Problem



# Lattice problems

## Batch Closest Vector Problem



# Lattice problems

Why study CVPP?

## Concrete applications

- Speeding up lattice enumeration for SVP or CVP [GNR10]
- Solving approximate SVP on ideal lattices [PHS19]
- Computing class group actions in a relation lattice [BKV19]

Commonly a lattice basis (public key) is known long before the target vectors (encryptions, signatures)

# Voronoi cells

## Voronoi tiling



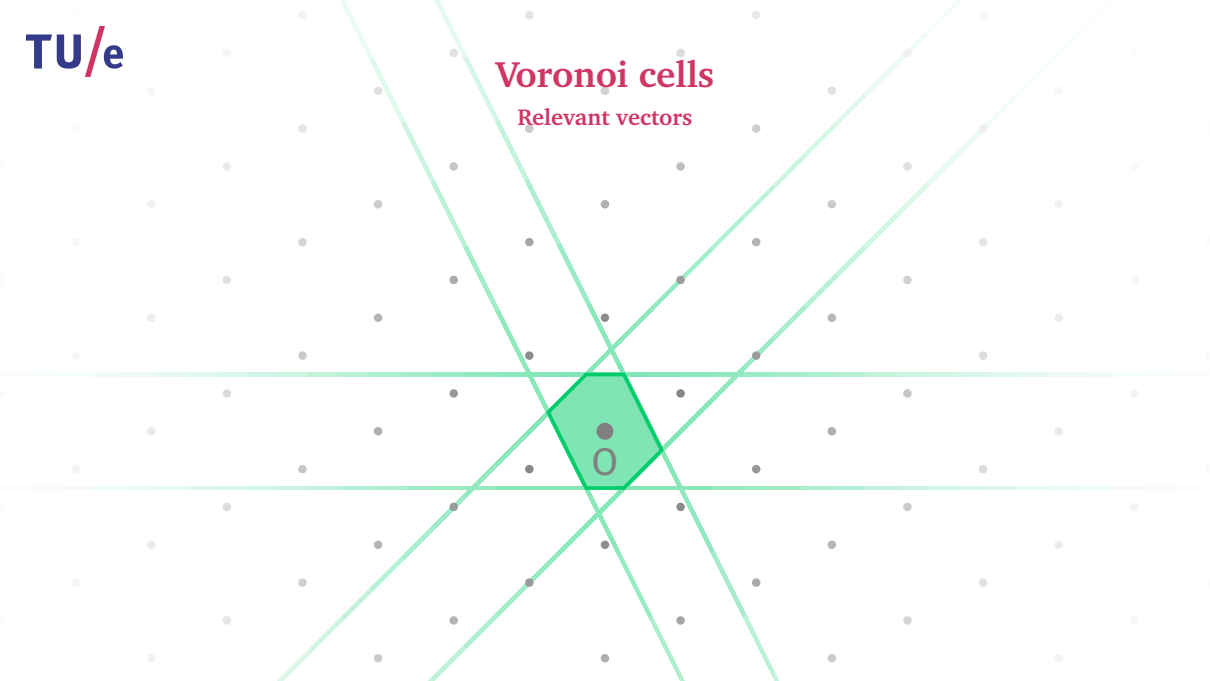
0

Voronoi cells

Voronoi tiling

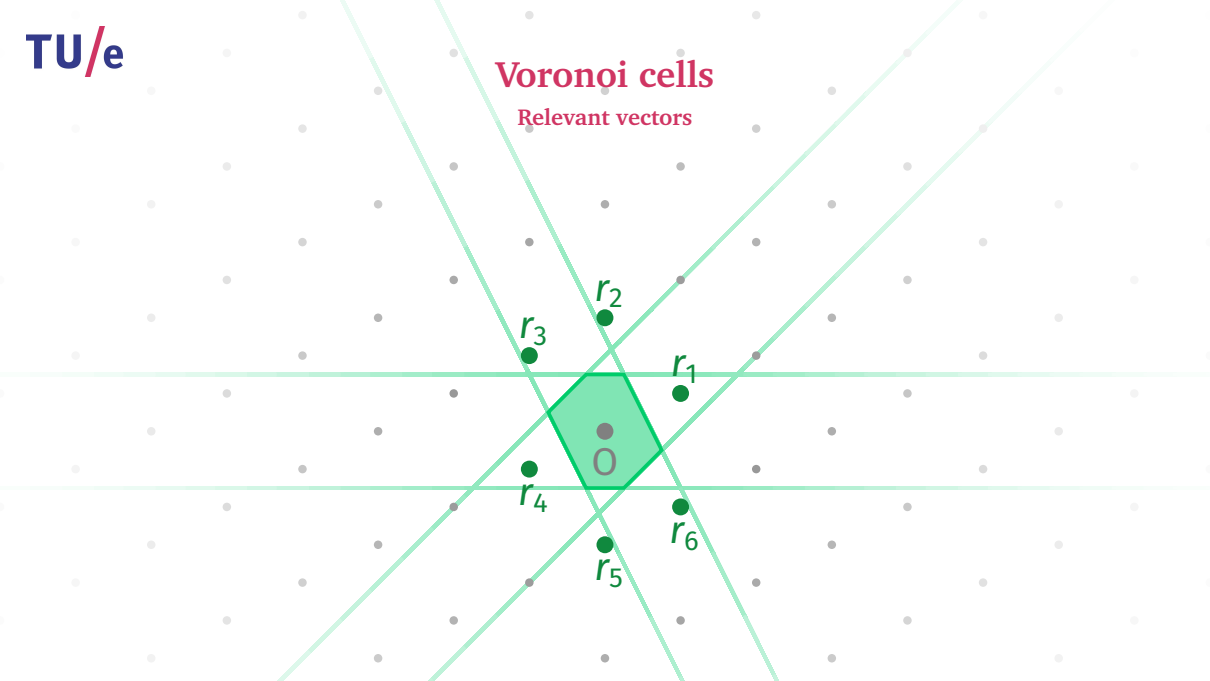


Voronoi cells  
Relevant vectors



# Voronoi cells

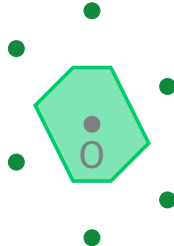
Relevant vectors



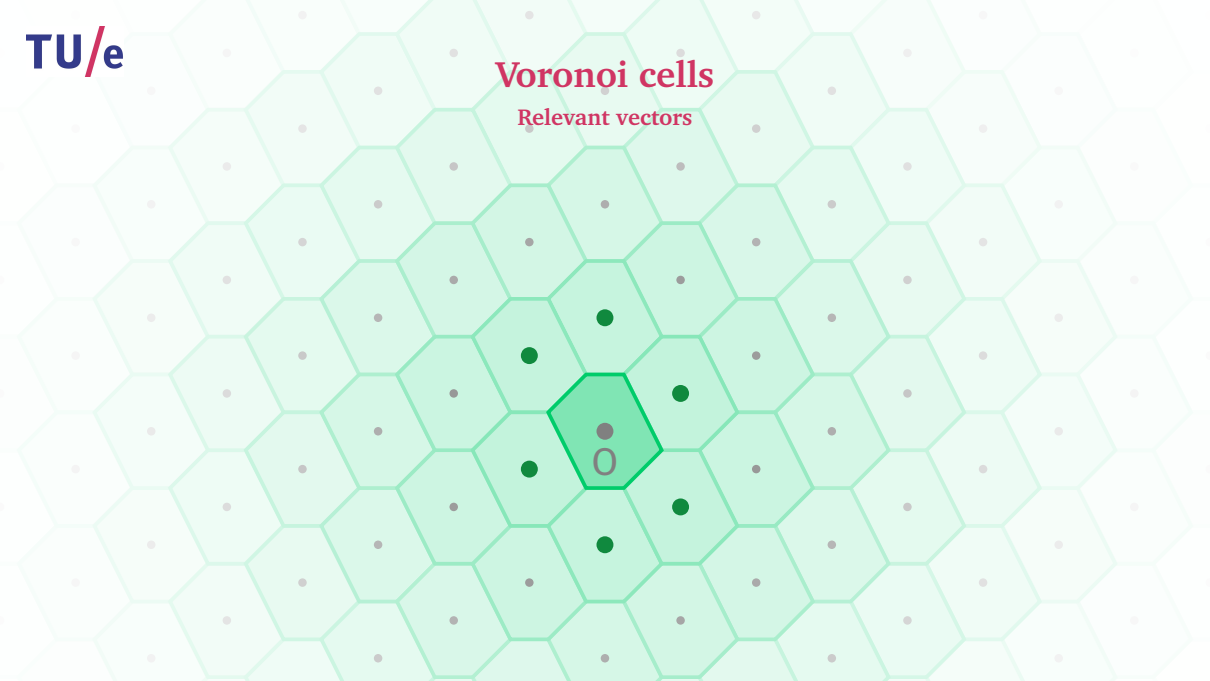


# Voronoi cells

Relevant vectors

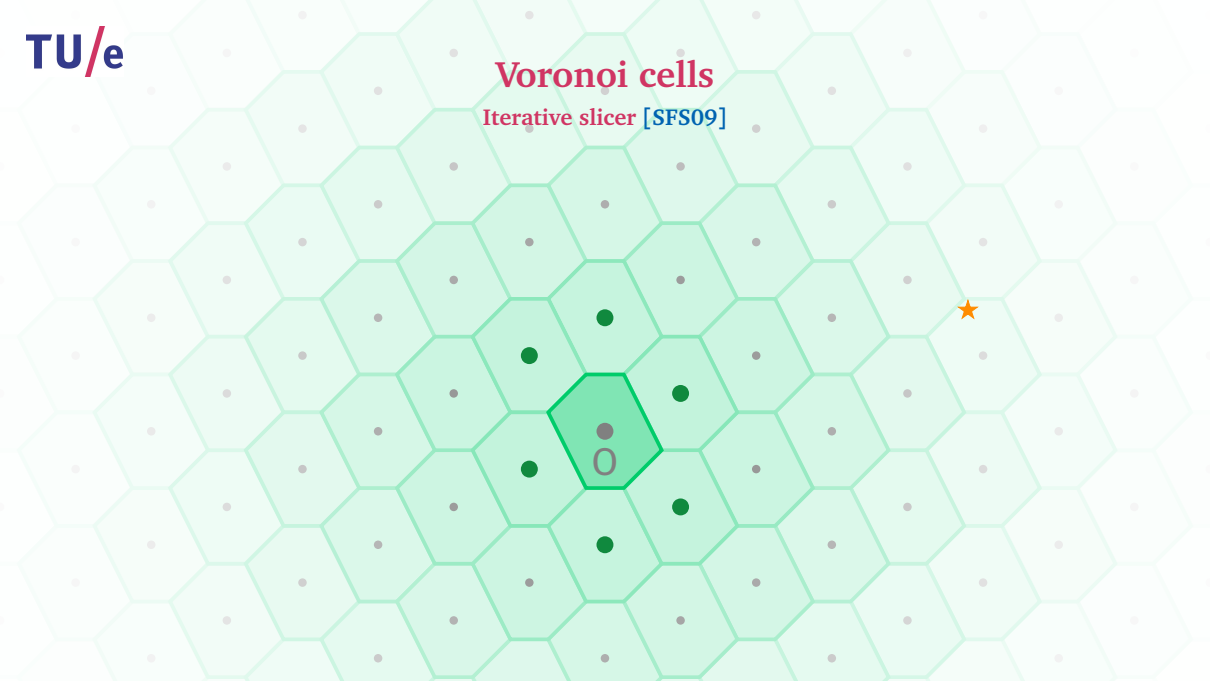


Voronoi cells  
Relevant vectors



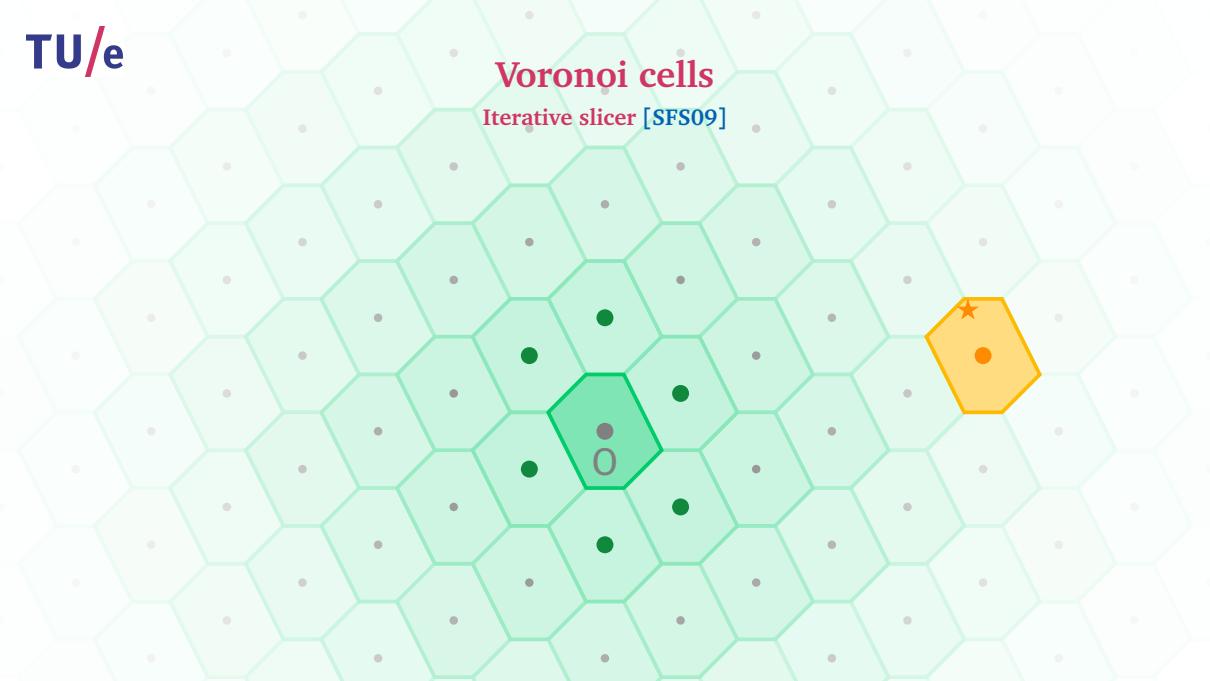
# Voronoi cells

Iterative slicer [SFS09]



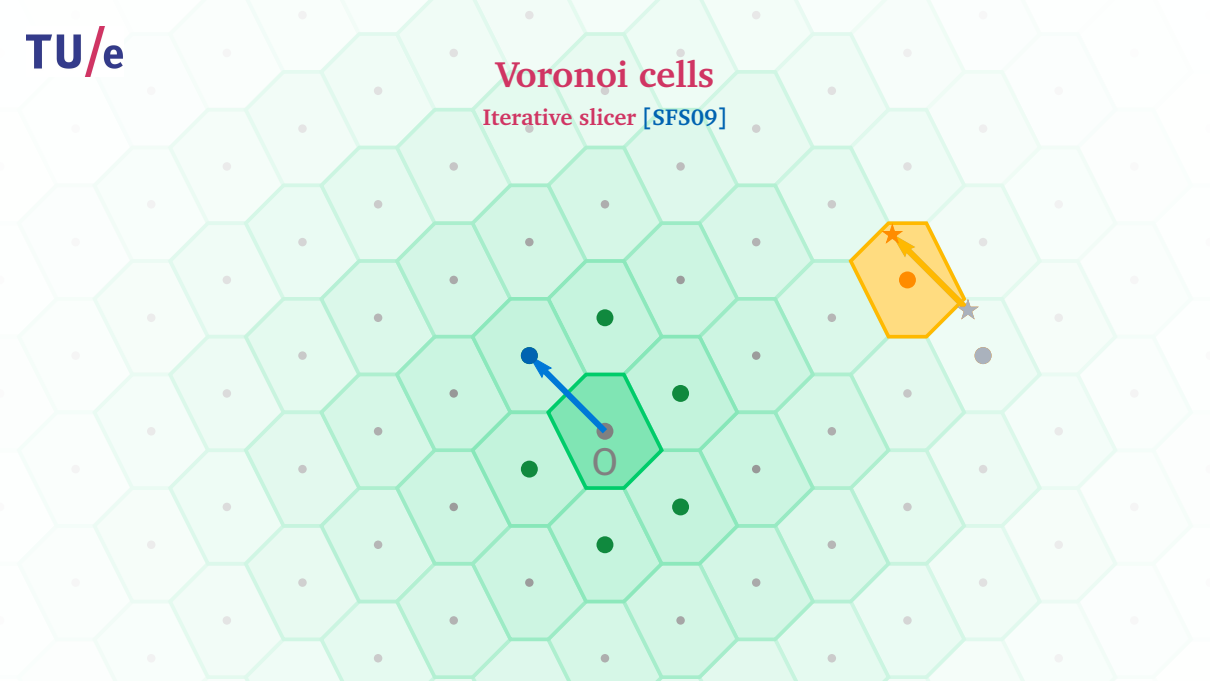
# Voronoi cells

Iterative slicer [SFS09]



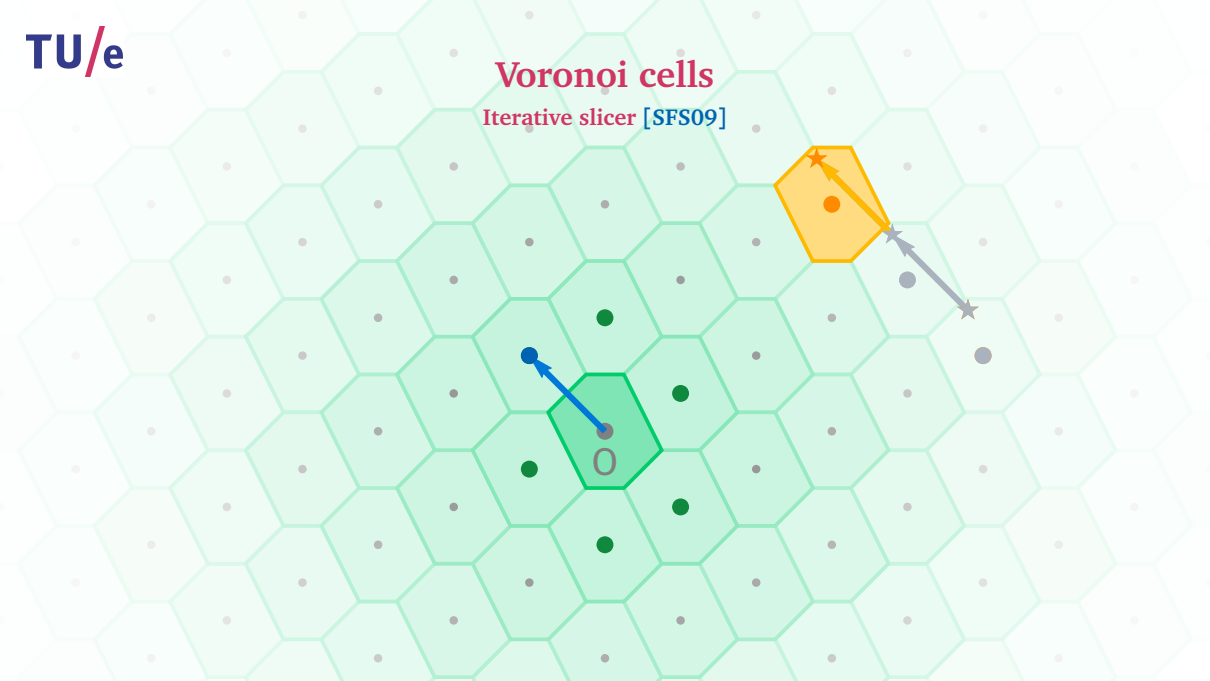
# Voronoi cells

Iterative slicer [SFS09]



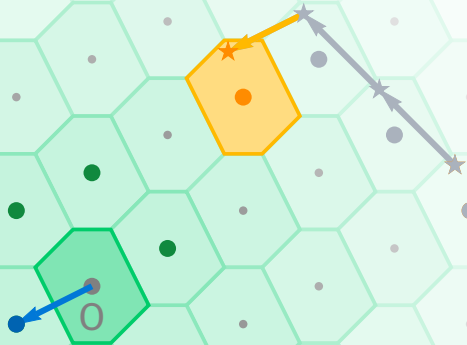
# Voronoi cells

Iterative slicer [SFS09]



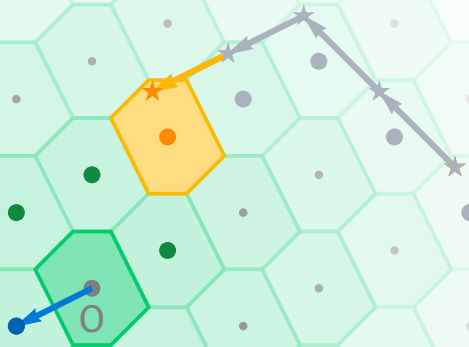
# Voronoi cells

Iterative slicer [SFS09]



# Voronoi cells

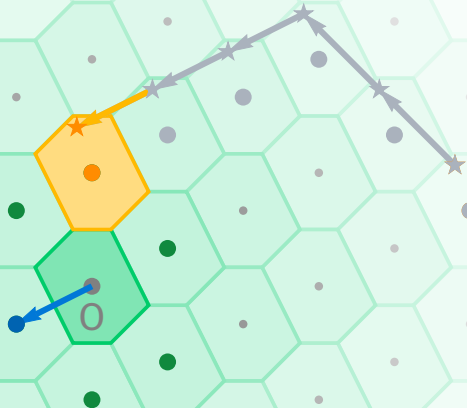
Iterative slicer [SFS09]





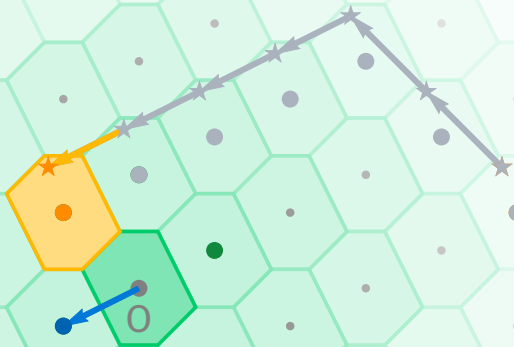
# Voronoi cells

Iterative slicer [SFS09]



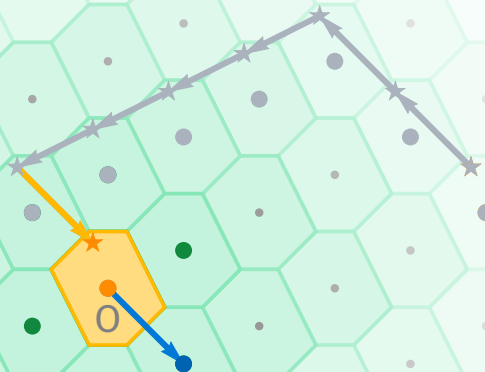
# Voronoi cells

Iterative slicer [SFS09]



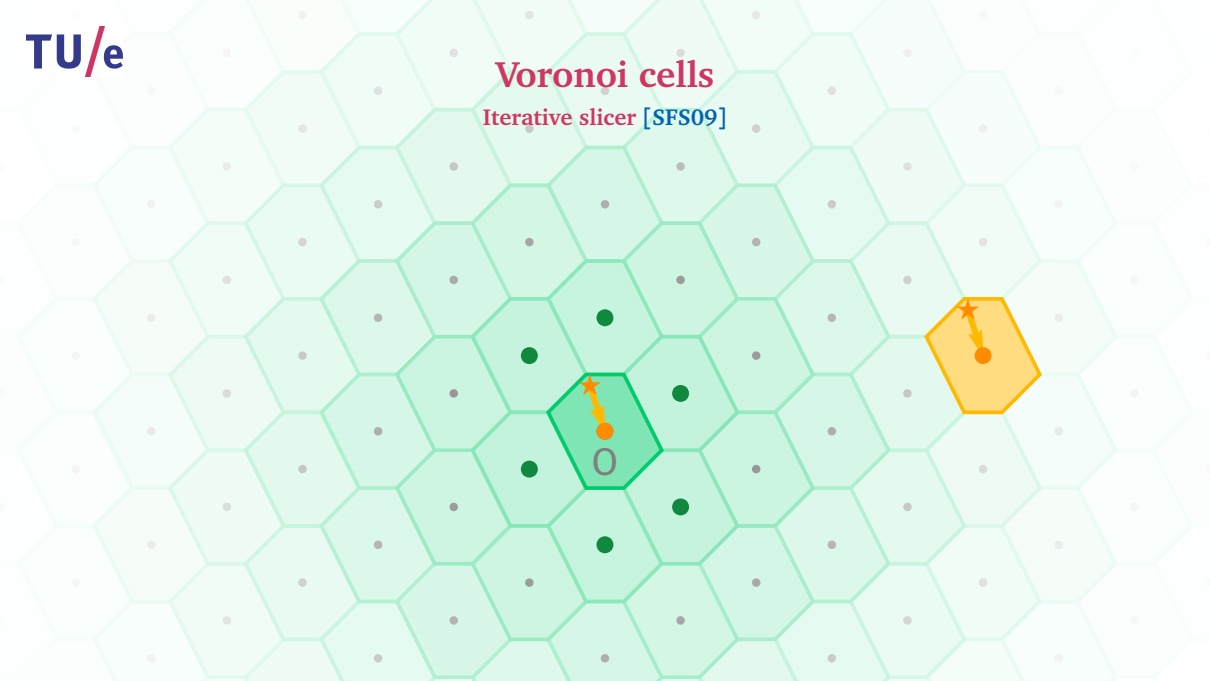
# Voronoi cells

Iterative slicer [SFS09]



# Voronoi cells

Iterative slicer [SFS09]



# Approximate Voronoi cells

Decrease list size

0



The diagram shows a 2D plane populated with numerous small gray dots representing points. A single, larger black dot is positioned near the center of the grid. Directly below this black dot is the number '0'. The text 'Approximate Voronoi cells' is at the top, and 'Decrease list size' is just below it. The TU/e logo is in the top-left corner.

# Approximate Voronoi cells

Decrease list size

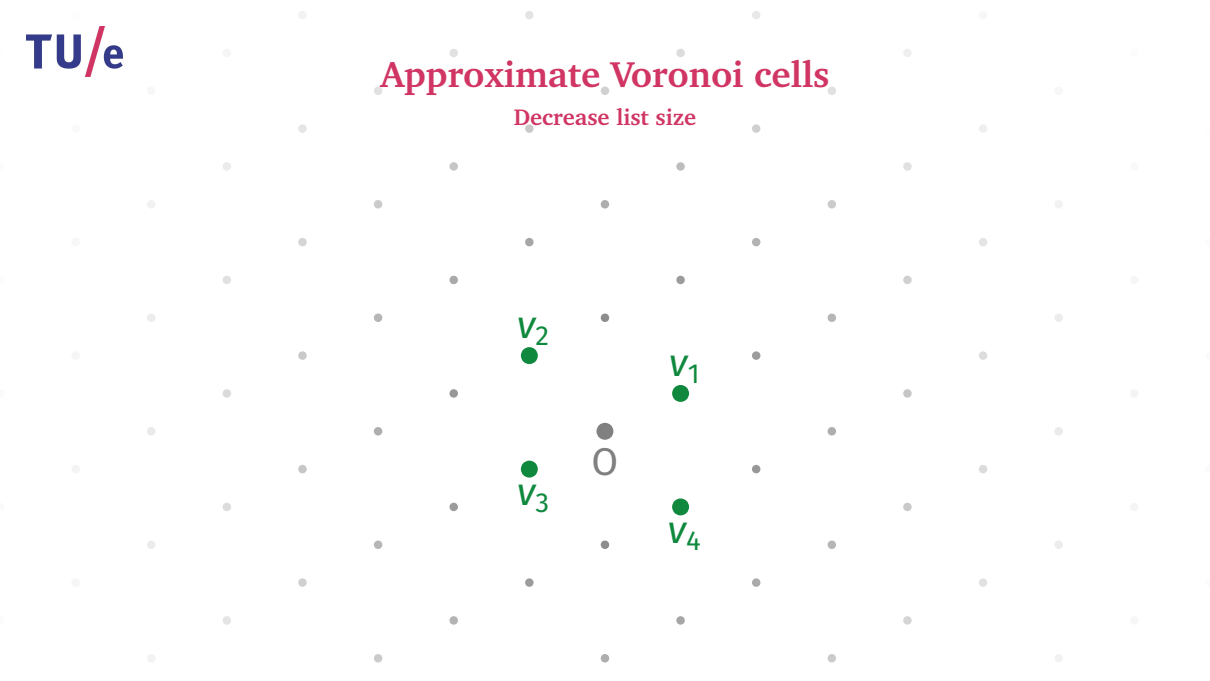
$V_2$

$V_1$

0

$V_3$

$V_4$



# Approximate Voronoi cells

Decrease list size

$V_2$

$V_1$

$V_3$

$V_4$

0



The diagram illustrates the process of decreasing the list size in a Voronoi diagram algorithm. It features a light gray background with a grid of small gray dots representing points. Four green lines intersect to form four regions, each labeled with a green dot and a label:  $V_1$ ,  $V_2$ ,  $V_3$ , and  $V_4$ . A central gray dot is labeled '0'. The green lines are slightly curved, suggesting they are approximations of the true Voronoi boundaries.

# Approximate Voronoi cells

Decrease list size

$V_2$

$V_1$

$V_3$

$V_4$

0

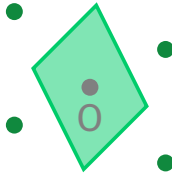


The diagram illustrates the process of decreasing the list size in a Voronoi diagram. A central point, labeled '0', is surrounded by four green lines that intersect at it, forming a green quadrilateral region. This region is labeled '0' in the center. Four points, labeled  $V_1$ ,  $V_2$ ,  $V_3$ , and  $V_4$ , are marked with green dots on the lines.  $V_1$  is on the top-right line,  $V_2$  is on the top-left line,  $V_3$  is on the bottom-left line, and  $V_4$  is on the bottom-right line. The background is filled with many small gray dots representing the point set. The text 'Approximate Voronoi cells' is written in red at the top, and 'Decrease list size' is written in red below it.



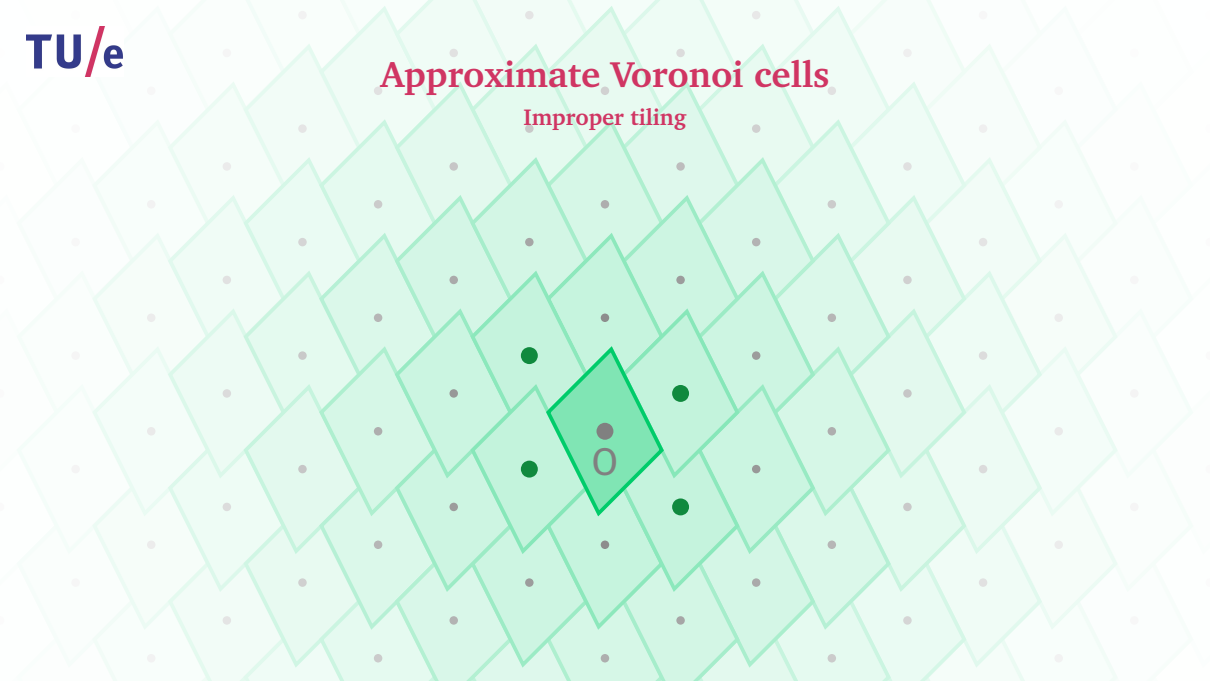
# Approximate Voronoi cells

Decrease list size



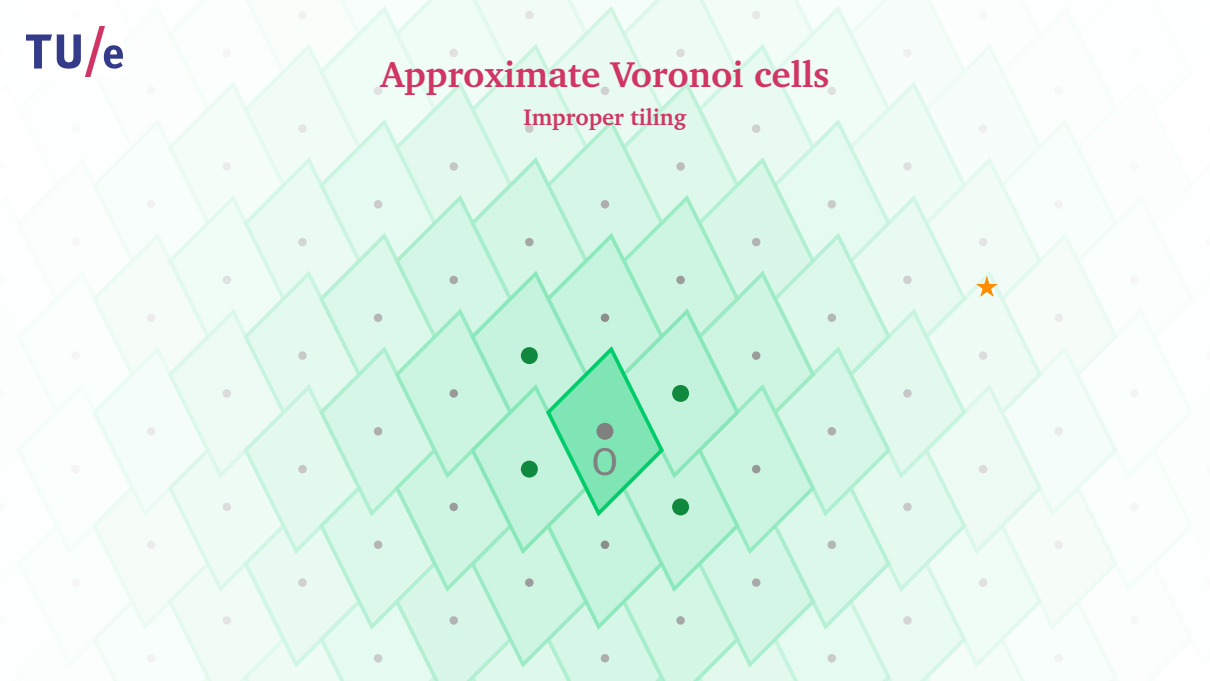
# Approximate Voronoi cells

Improper tiling



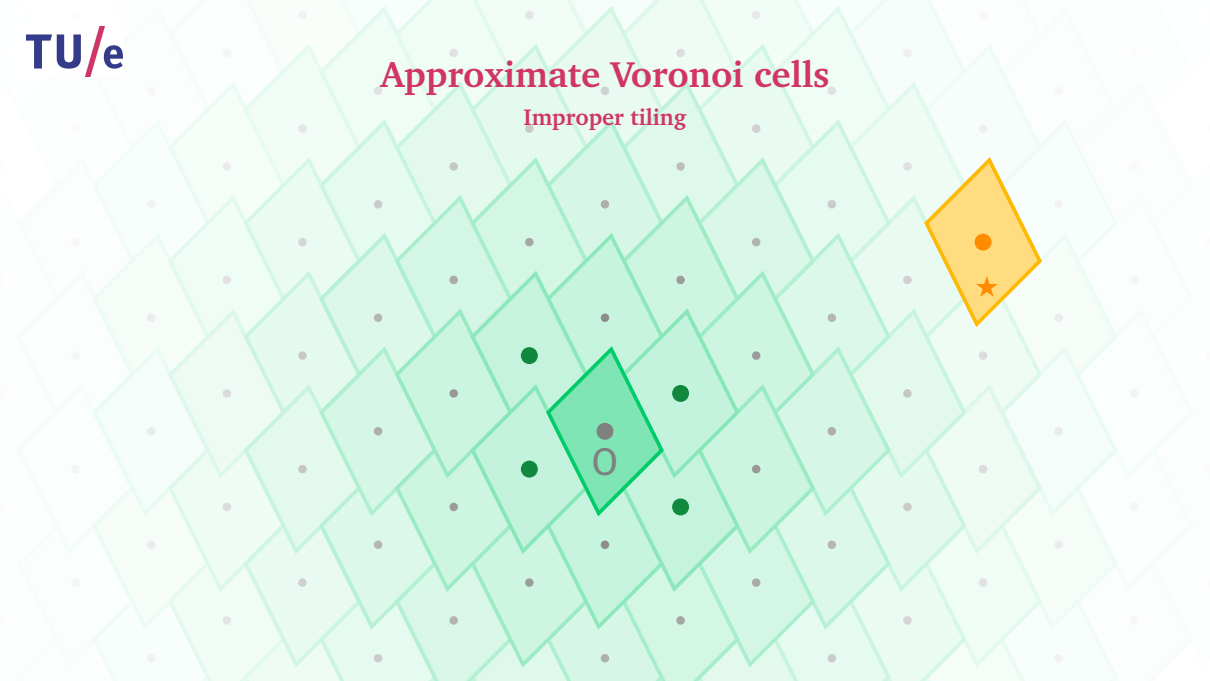
# Approximate Voronoi cells

Improper tiling



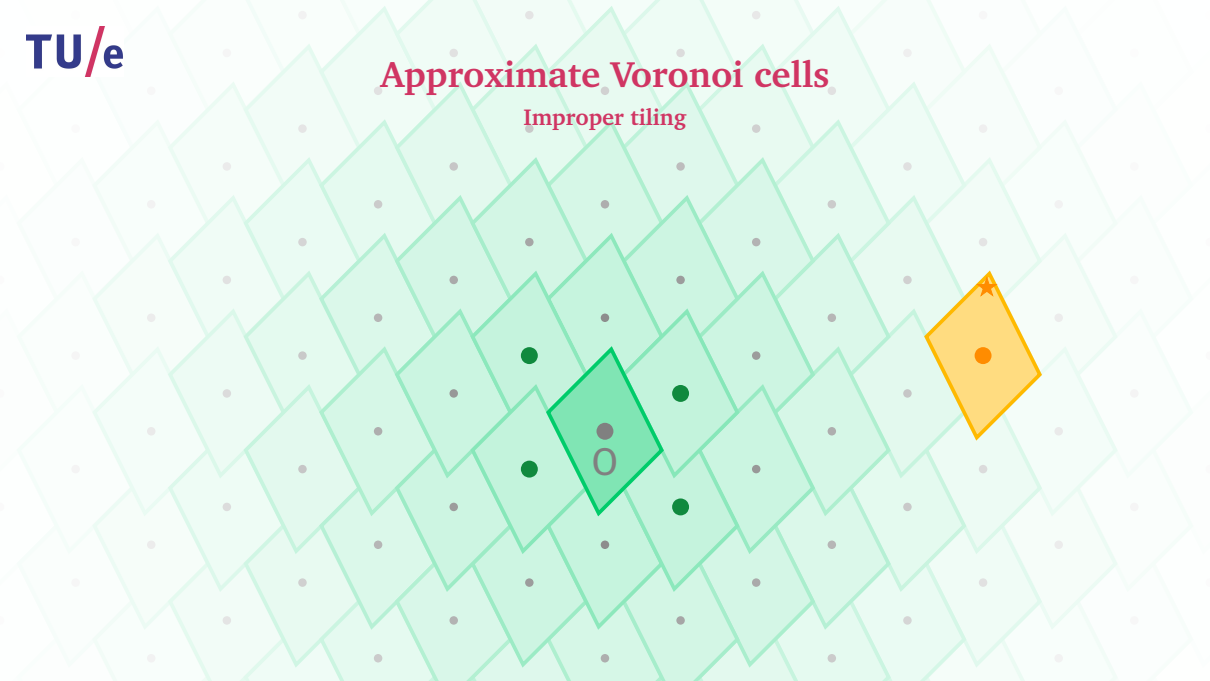
# Approximate Voronoi cells

Improper tiling



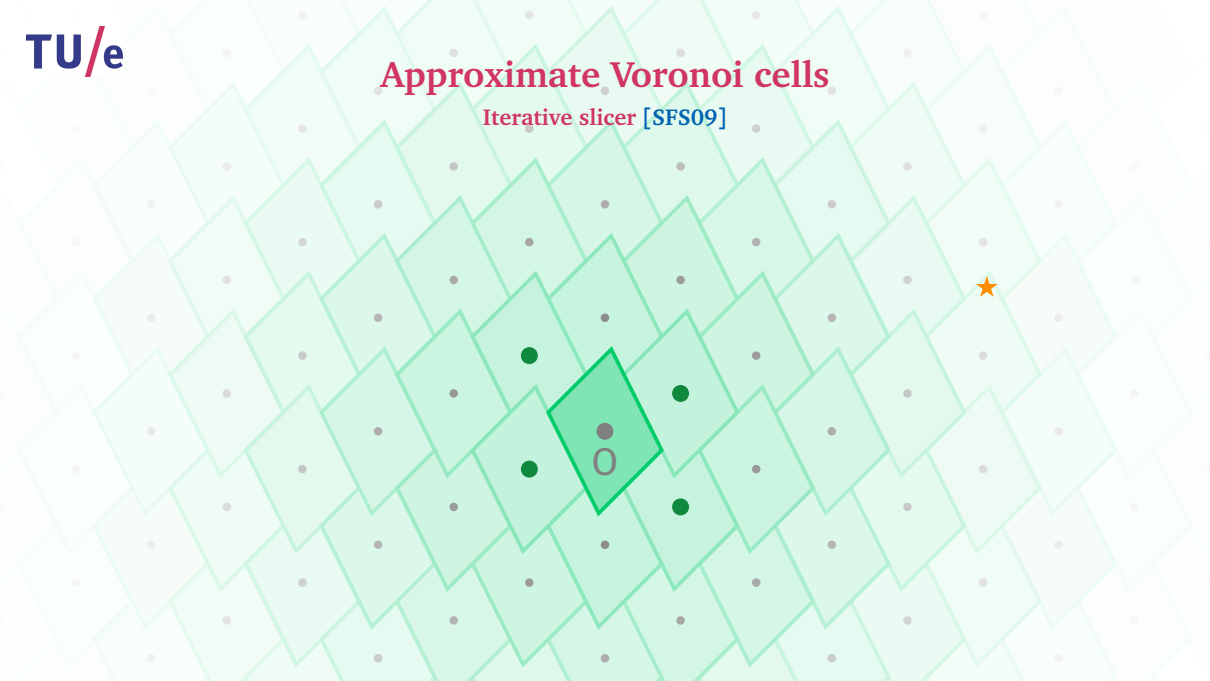
# Approximate Voronoi cells

Improper tiling



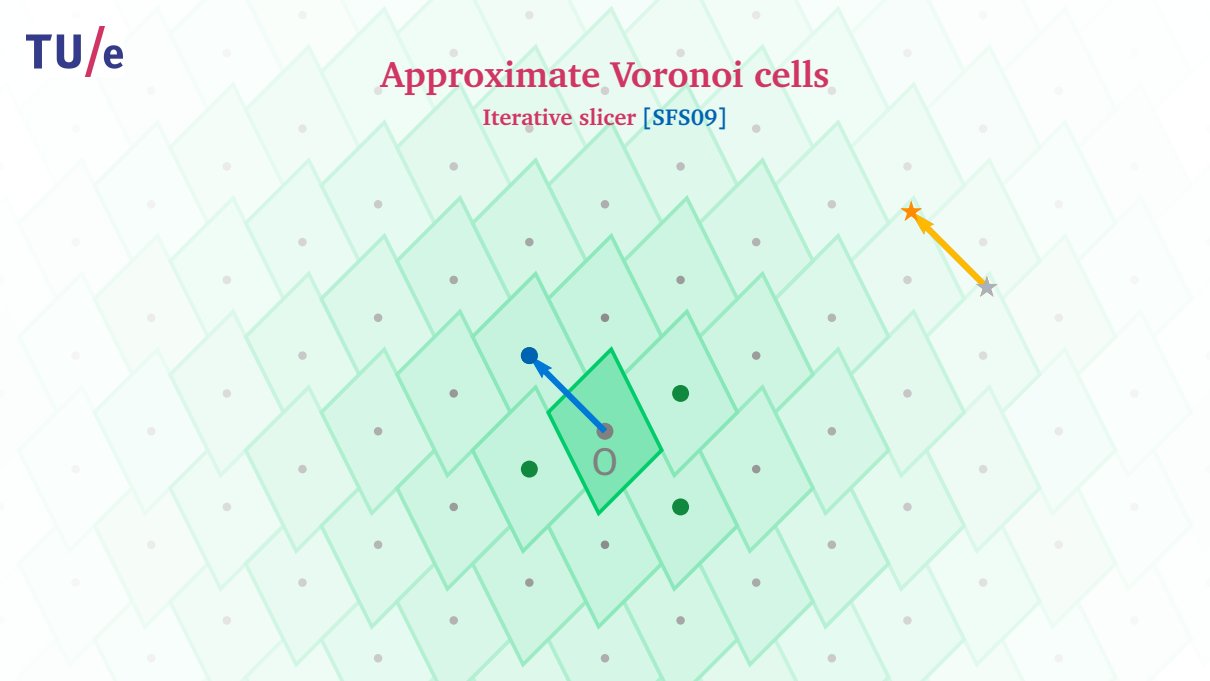
# Approximate Voronoi cells

Iterative slicer [SFS09]



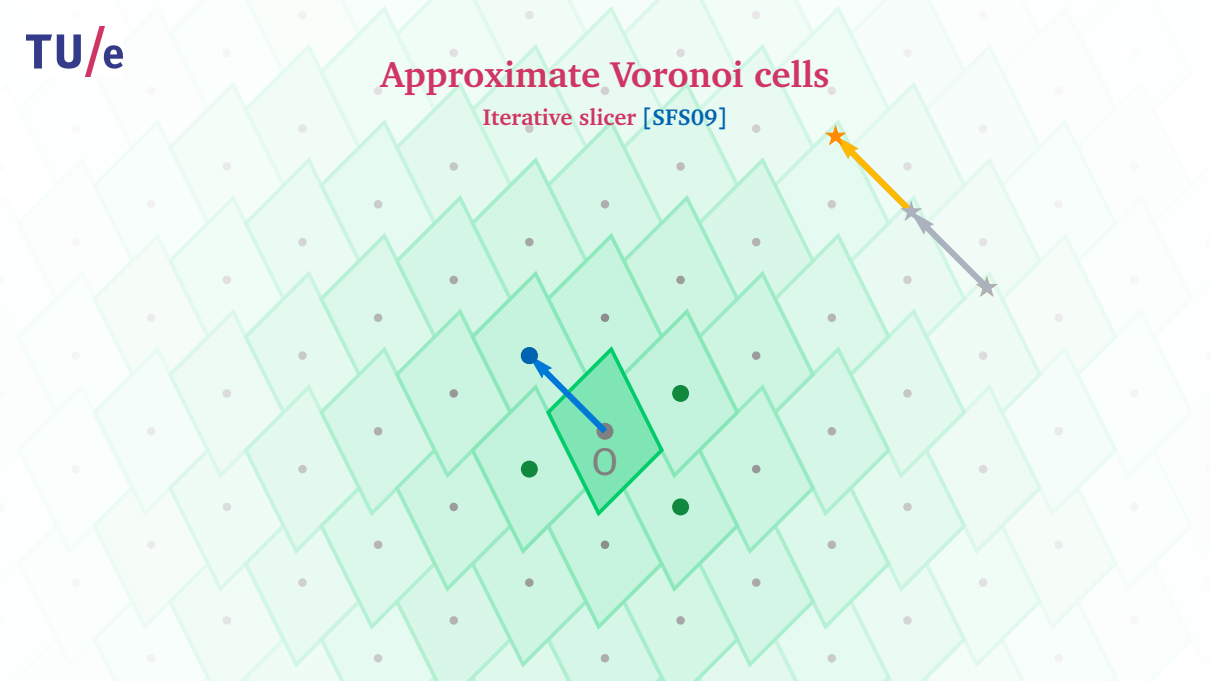
# Approximate Voronoi cells

Iterative slicer [SFS09]



# Approximate Voronoi cells

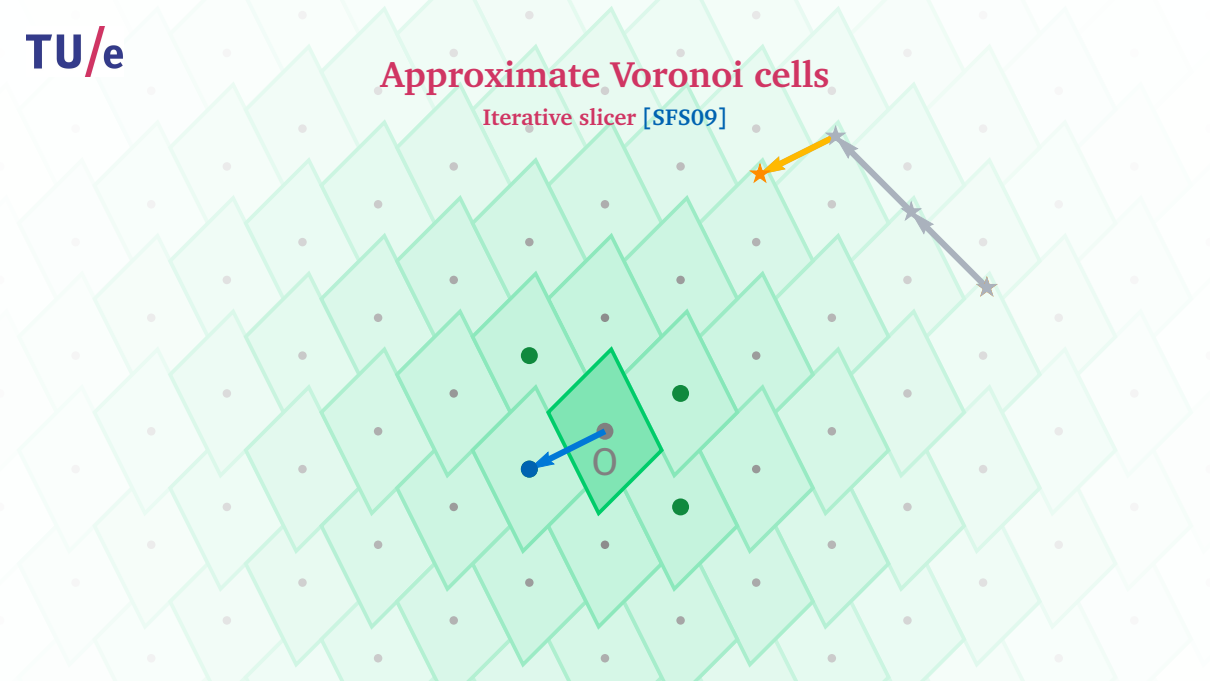
Iterative slicer [SFS09]





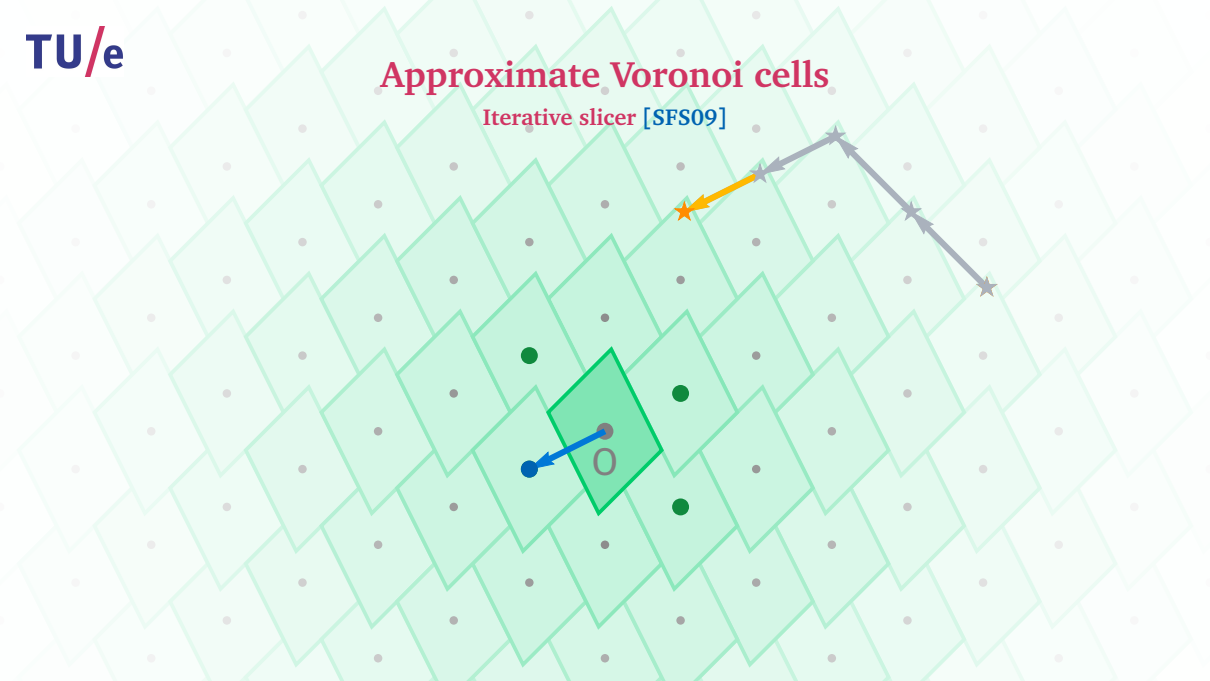
# Approximate Voronoi cells

Iterative slicer [SFS09]



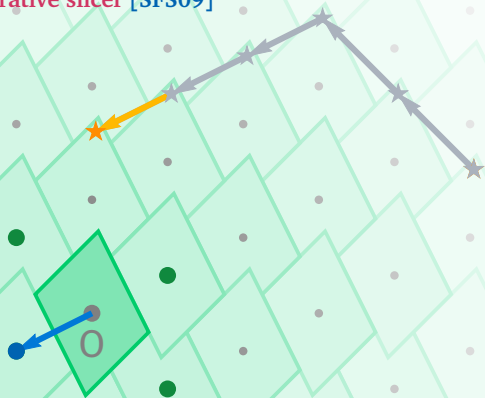
# Approximate Voronoi cells

Iterative slicer [SFS09]



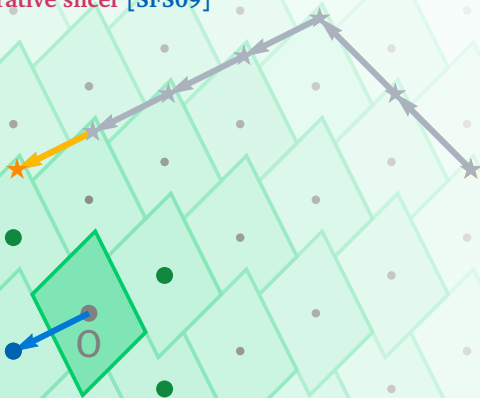
# Approximate Voronoi cells

Iterative slicer [SFS09]



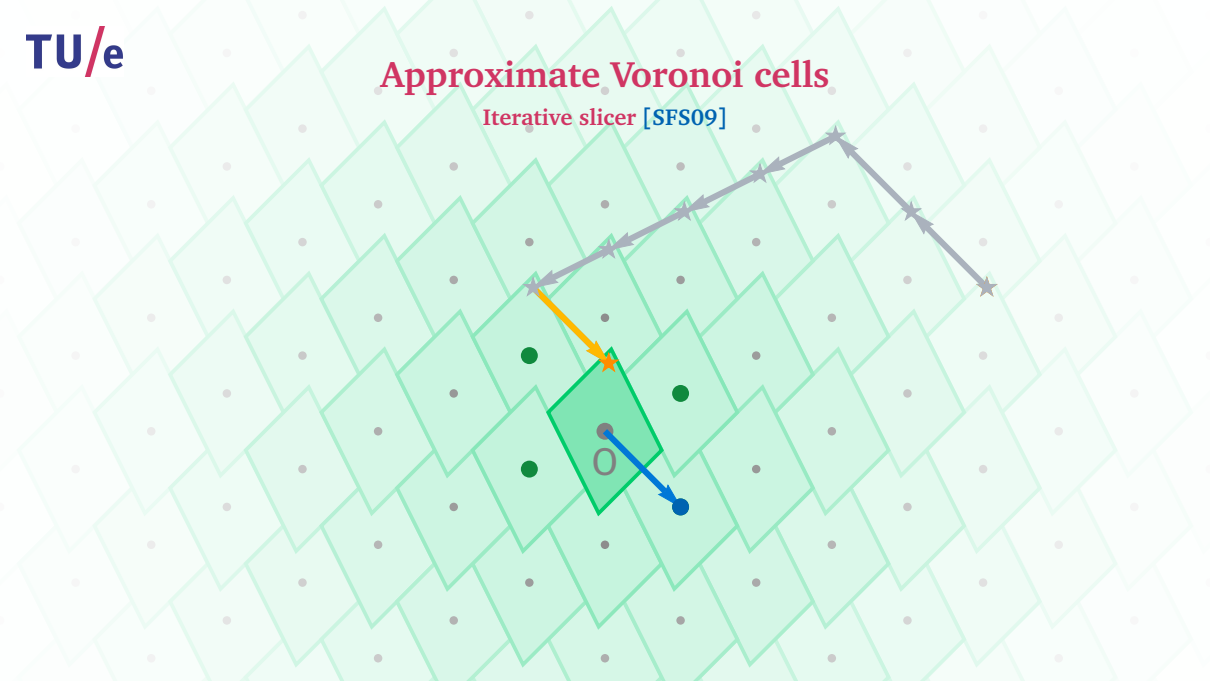
# Approximate Voronoi cells

Iterative slicer [SFS09]



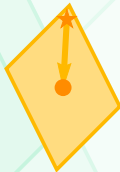
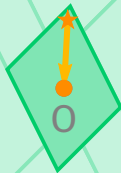
# Approximate Voronoi cells

Iterative slicer [SFS09]



# Approximate Voronoi cells

Iterative slicer [SFS09]



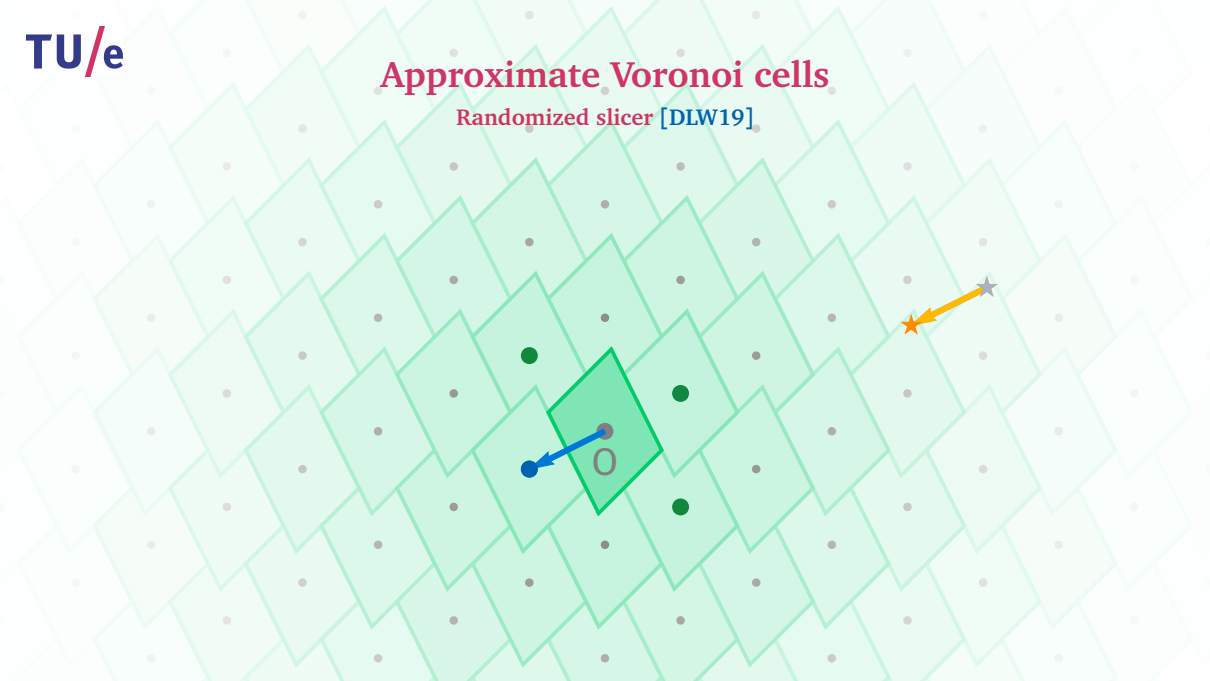
# Approximate Voronoi cells

Randomized slicer [DLW19]



# Approximate Voronoi cells

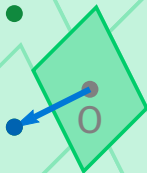
Randomized slicer [DLW19]





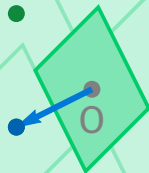
# Approximate Voronoi cells

Randomized slicer [DLW19]



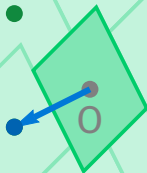
# Approximate Voronoi cells

Randomized slicer [DLW19]



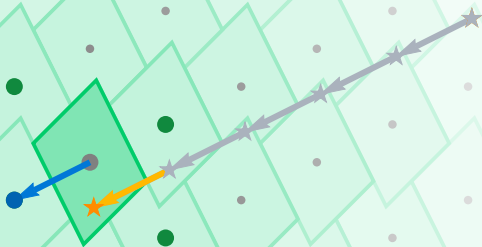
# Approximate Voronoi cells

Randomized slicer [DLW19]



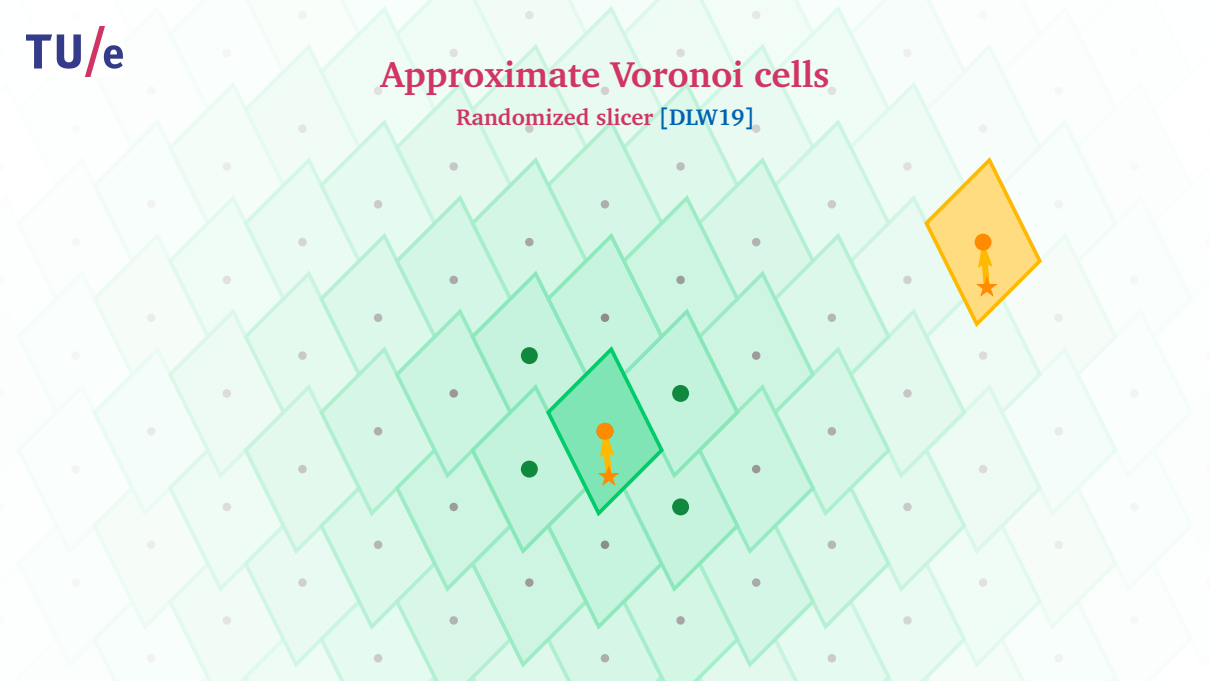
# Approximate Voronoi cells

Randomized slicer [DLW19]



# Approximate Voronoi cells

Randomized slicer [DLW19]



## Approximate Voronoi cells

Success probability estimation

**Main problem:** Success probability  $p$  of the iterative slicer?

# Approximate Voronoi cells

## Success probability estimation

**Main problem:** Success probability  $p$  of the iterative slicer?

**Previous results:** [DLW19]

- Directly obtained a lower bound on  $p$  via the slicer
- Conjectured that  $p$  is exactly proportional to  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$
- Open problem: obtain a tight analysis, perhaps via  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$

# Approximate Voronoi cells

## Success probability estimation

**Main problem:** Success probability  $p$  of the iterative slicer?

**Previous results:** [DLW19]

- Directly obtained a lower bound on  $p$  via the slicer
- Conjectured that  $p$  is exactly proportional to  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$
- Open problem: obtain a tight analysis, perhaps via  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$

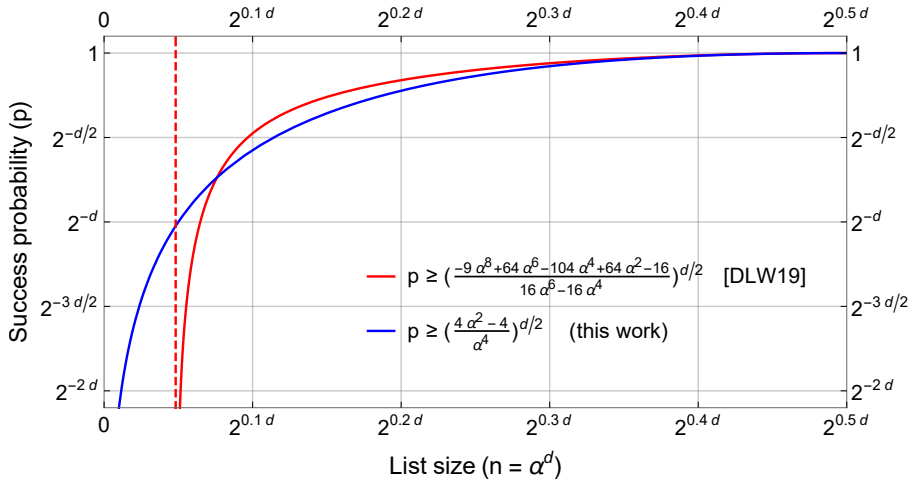
**This work:**

- Proved tight bounds on  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$  under the Gaussian heuristic
- Results show that  $p$  **cannot** be (exactly) proportional to  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$
- From  $p \geq \text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$  we obtain new lower bounds on  $p$
- No nonsensical asymptote at  $2^{0.05d}$  memory anymore



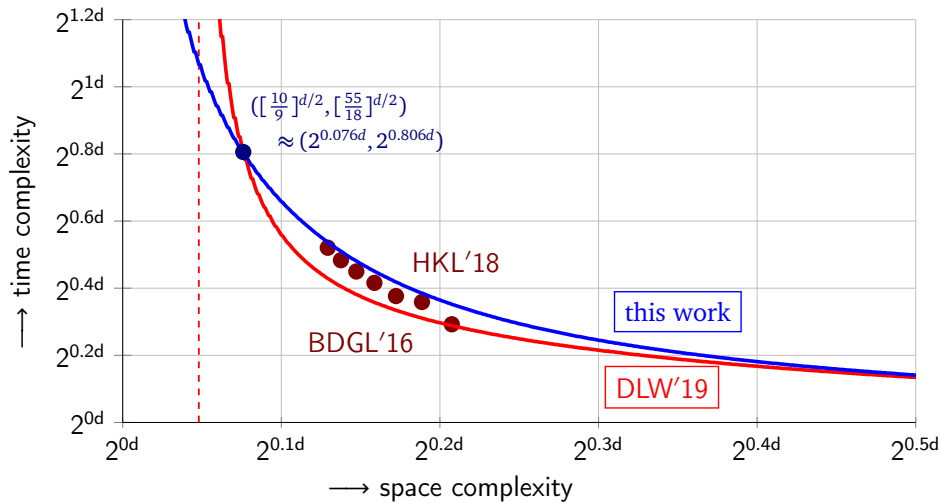
# Approximate Voronoi cells

Lower bounds on success probability



## Approximate Voronoi cells

Time-space trade-offs for CVPP



## Conclusion

### Voronoi cells

- Solves CVPP exactly in the worst case for all lattices
- Requires too much space (and time) to be useful

**Voronoi cells**

- Solves CVPP exactly in the worst case for all lattices
- Requires too much space (and time) to be useful

**Approximate Voronoi cells**

- Offers heuristic alternative to exact Voronoi cells
- Success probability analysis:
  - ▶ Original analysis did not appear to be tight
  - ▶ Conjectured that tighter bounds may be obtained via  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$
  - ▶ This work: obtained tight bounds on the ratio  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$
  - ▶ Results in better CVPP complexities for low-memory regime
  - ▶ Unfortunately, approach via  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$  is not tight either

## Conclusion

### Voronoi cells

- Solves CVPP exactly in the worst case for all lattices
- Requires too much space (and time) to be useful

### Approximate Voronoi cells

- Offers heuristic alternative to exact Voronoi cells
- Success probability analysis:
  - ▶ Original analysis did not appear to be tight
  - ▶ Conjectured that tighter bounds may be obtained via  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$
  - ▶ This work: obtained tight bounds on the ratio  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$
  - ▶ Results in better CVPP complexities for low-memory regime
  - ▶ Unfortunately, approach via  $\text{vol}(\mathcal{V})/\text{vol}(\mathcal{V}_L)$  is not tight either

### Open problems

- Obtain truly tight bounds (ongoing work with Leo Ducas, Wessel van Woerden)
- Find an efficient BDDP-version of this CVPP algorithm