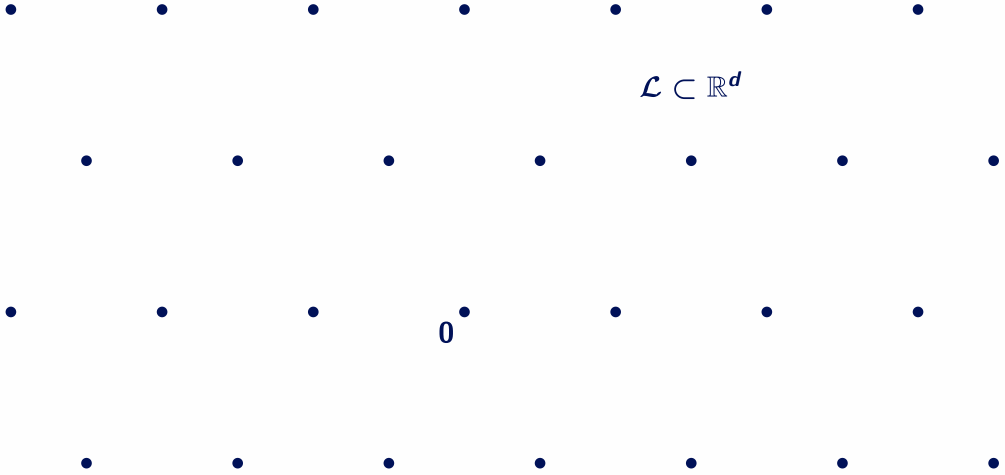
The randomized slicer for CVPP: sharper, faster, smaller, batchier

Leo Ducas, Thijs Laarhoven and **Wessel van Woerden.**

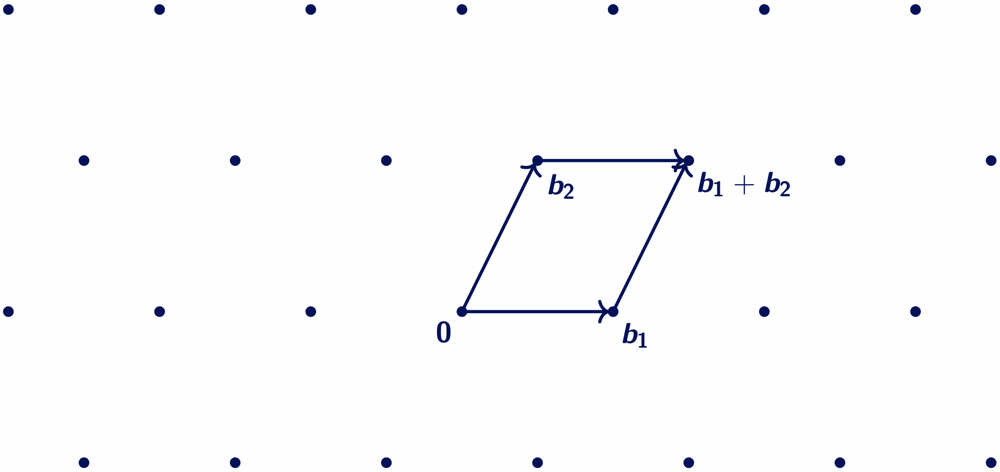
Centrum Wiskunde & Informatica

Technische Universiteit Eindhoven
University of Technology

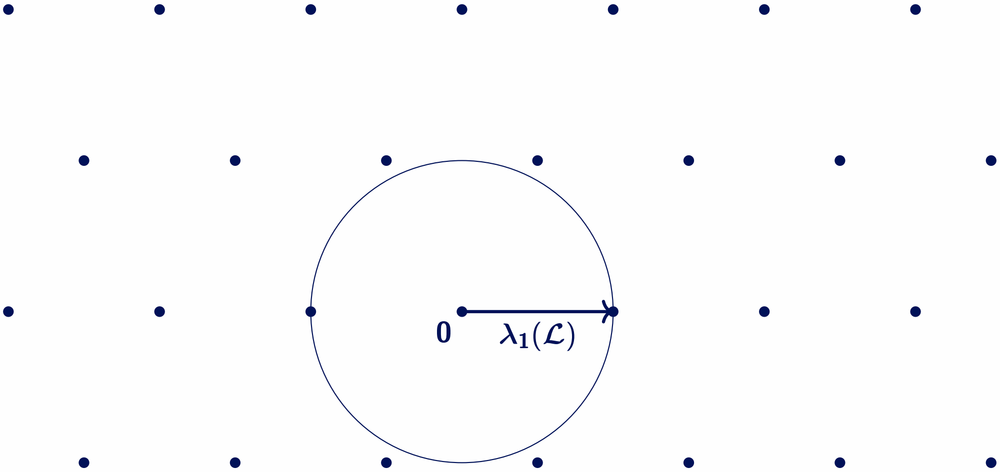
Lattice



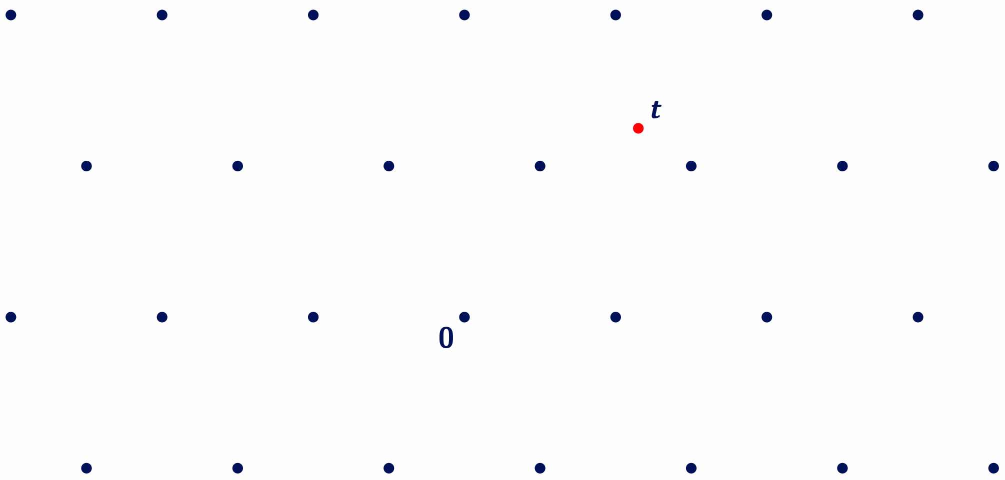
Lattice



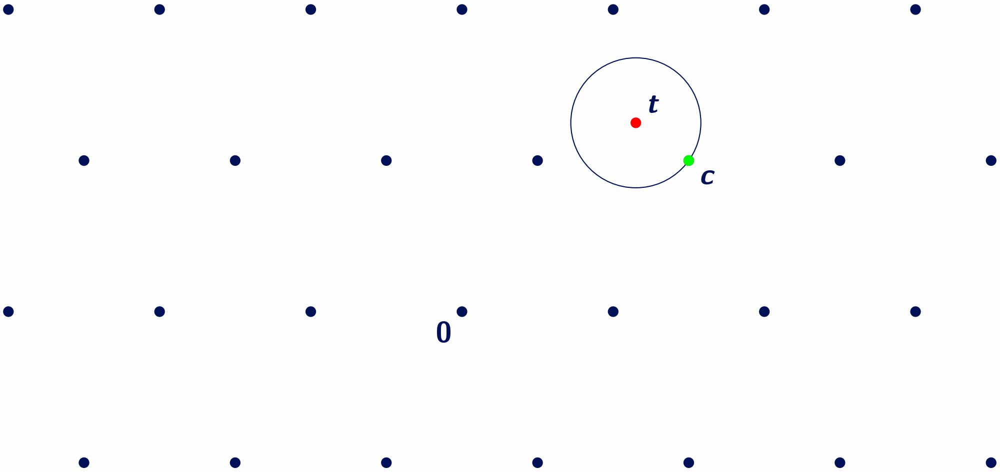
Shortest Vector Problem



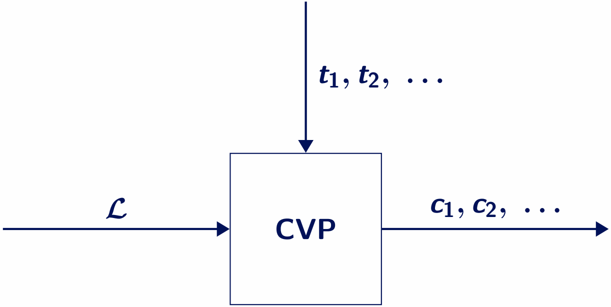
Closest Vector Problem



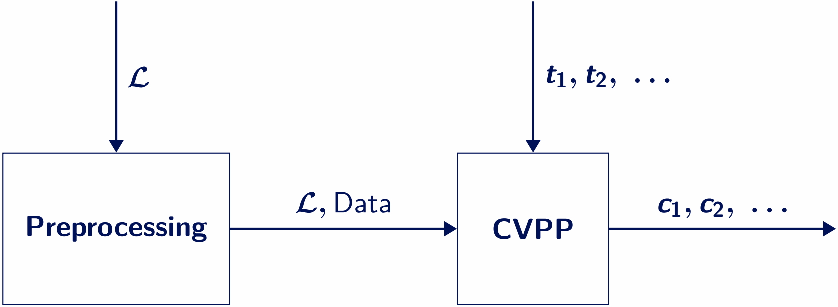
Closest Vector Problem



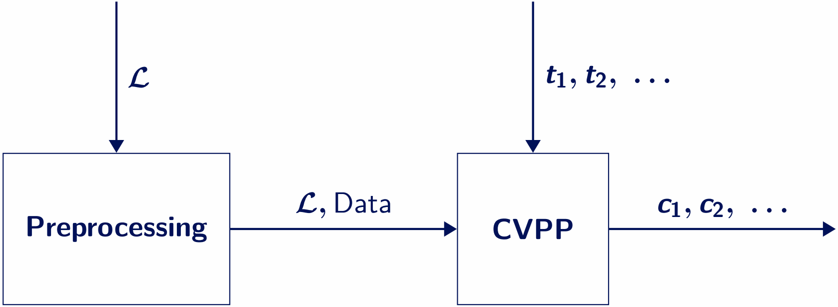
Closest Vector Problem



Closest Vector Problem with Preprocessing

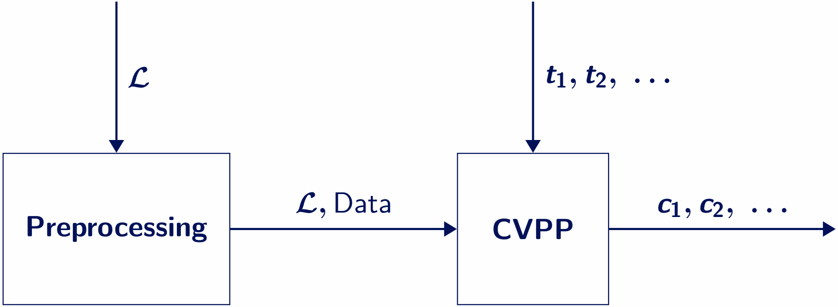


Closest Vector Problem with Preprocessing



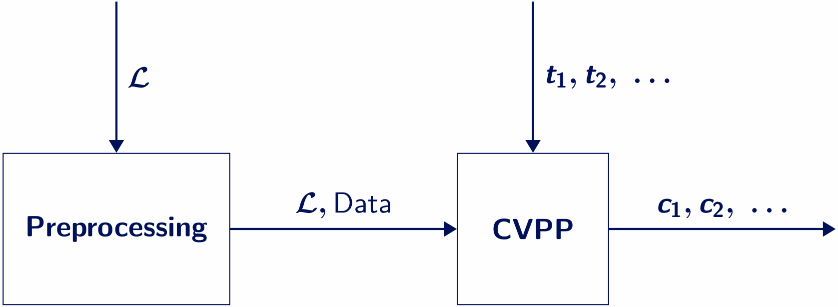
- Improved complexity per target.

Closest Vector Problem with Preprocessing



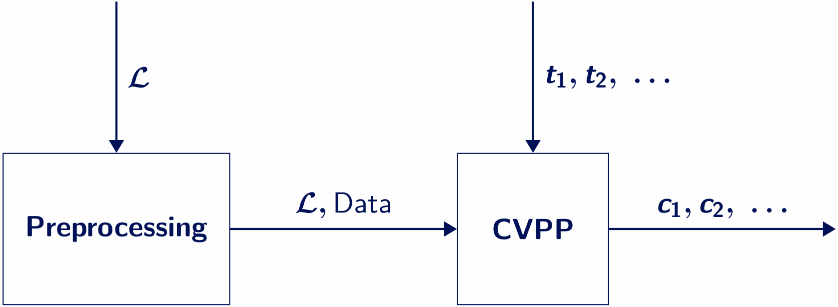
- Improved complexity per target.
- Trade-off between |Data| and CVPP time complexity.

Closest Vector Problem with Preprocessing



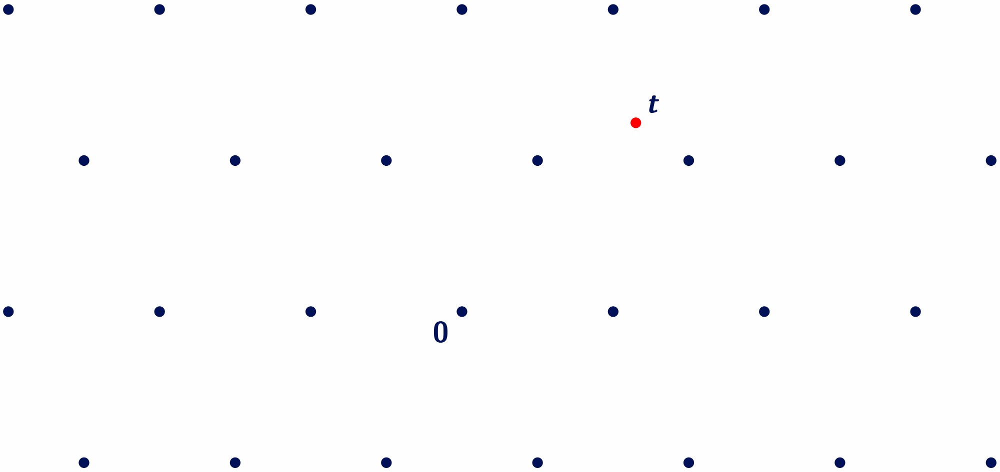
- Improved complexity per target.
- Trade-off between |Data| and CVPP time complexity.
- Ideal-SVP, Enumeration hybrid, computing Class Group actions…

Closest Vector Problem with Preprocessing

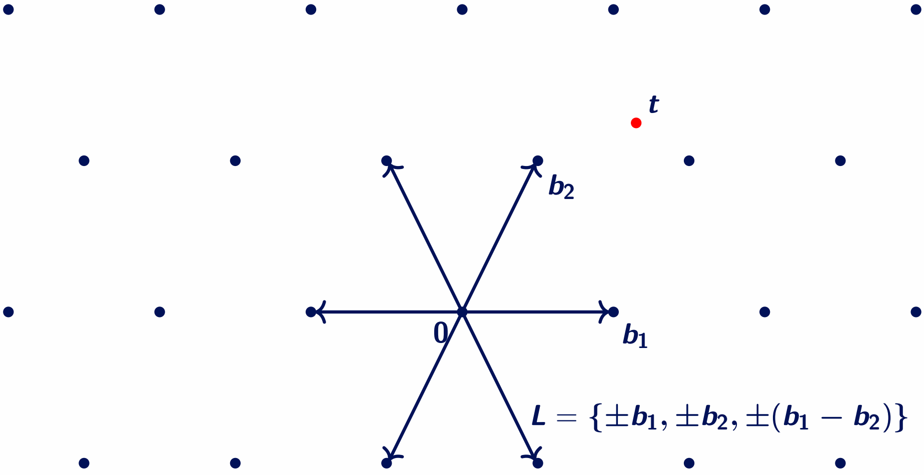


- Improved complexity per target.
- Trade-off between |Data| and CVPP time complexity.
- Ideal-SVP, Enumeration hybrid, computing Class Group actions...
- Preprocessing can be started before any target is known.

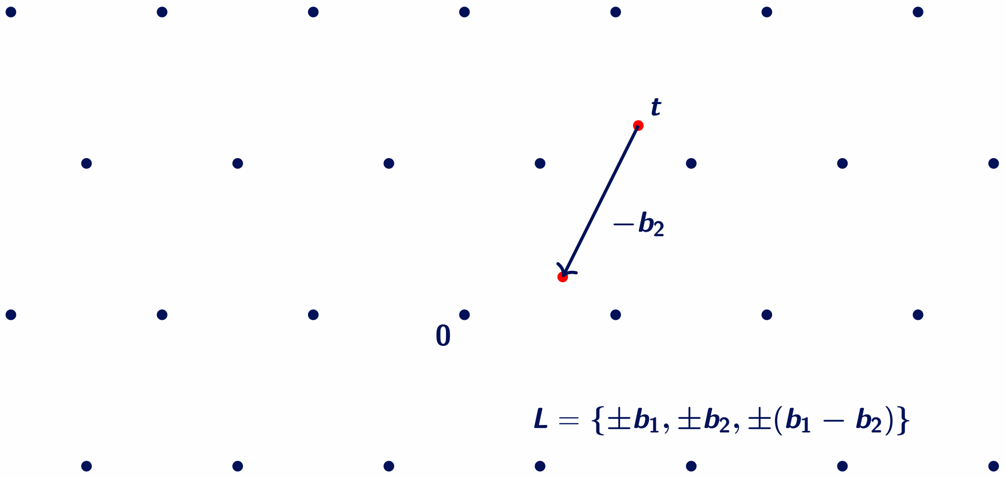
Iterative Slicer [SFS09]



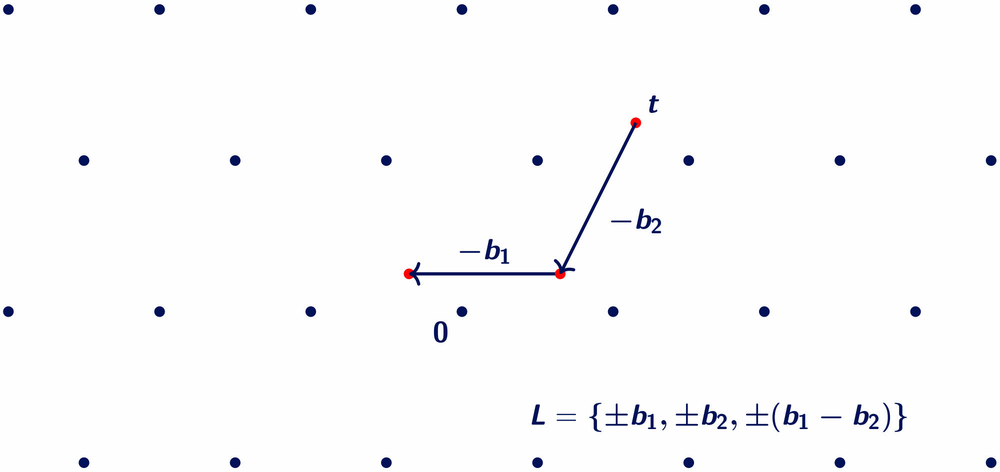
Iterative Slicer [SFS09]



Iterative Slicer [SFS09]

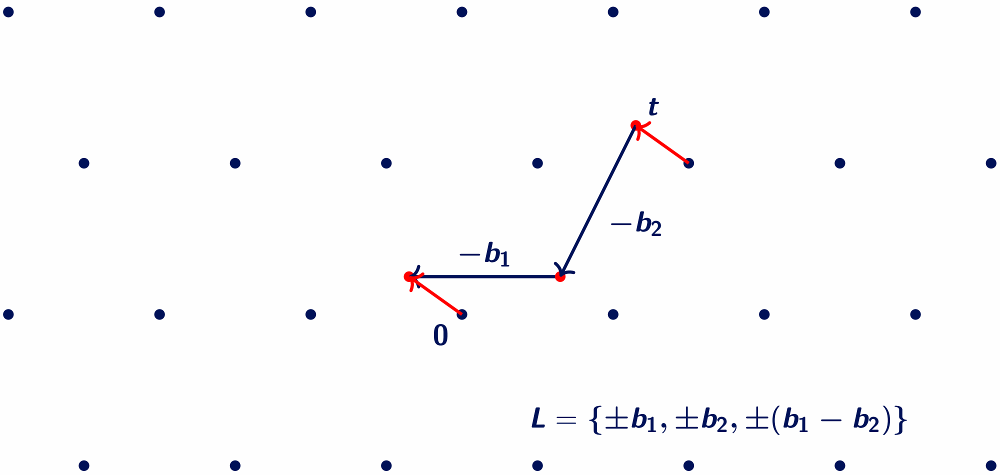


Iterative Slicer [SFS09]



$$L = \{\pm b_1, \pm b_2, \pm(b_1 - b_2)\}$$

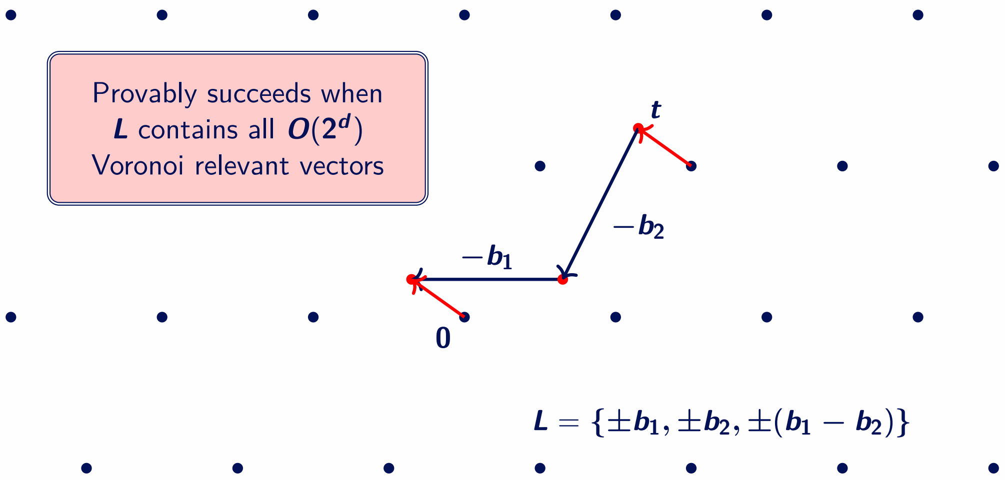
Iterative Slicer [SFS09]



$L = \{\pm b_1, \pm b_2, \pm(b_1 - b_2)\}$

Iterative Slicer [SFS09]

Provably succeeds when
L contains all $O(2^d)$
Voronoi relevant vectors

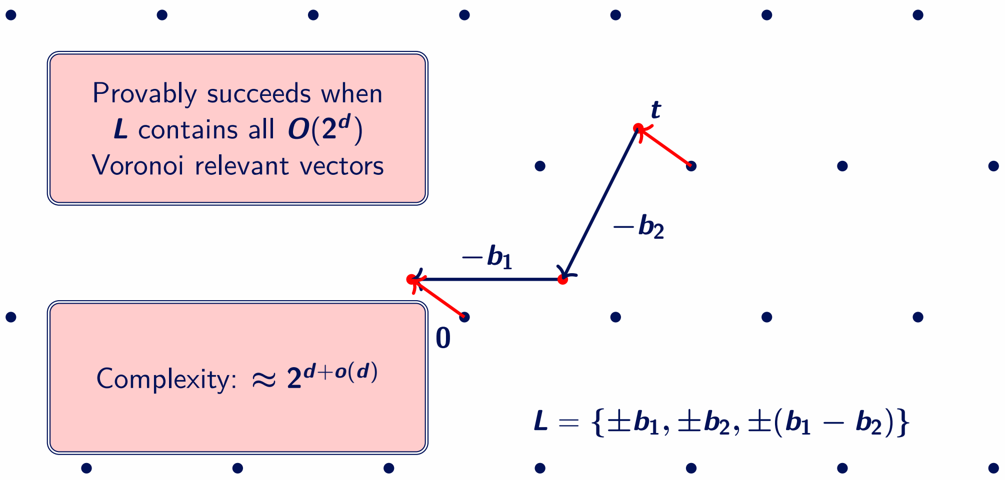


$L = \{\pm b_1, \pm b_2, \pm(b_1 - b_2)\}$

Iterative Slicer [SFS09]

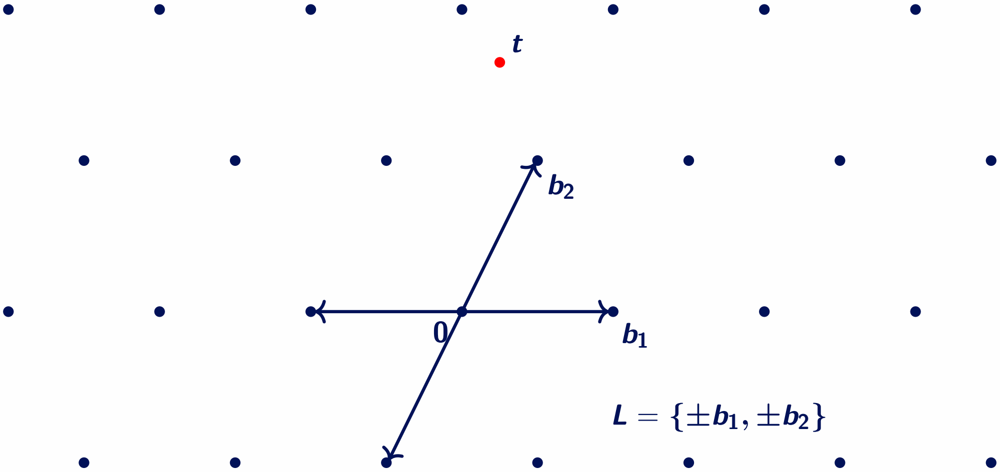
Provably succeeds when $\boldsymbol{L}$ contains all $O(2^d)$ Voronoi relevant vectors

Complexity: $\approx 2^{d+o(d)}$



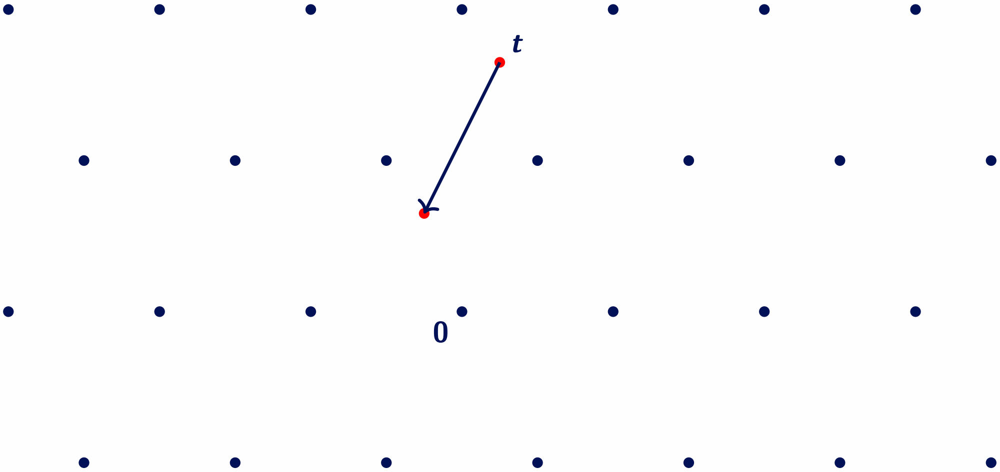
$\boldsymbol{L} = \{\pm b_1, \pm b_2, \pm(b_1 - b_2)\}$

Randomized Iterative Slicer [Laa'16]

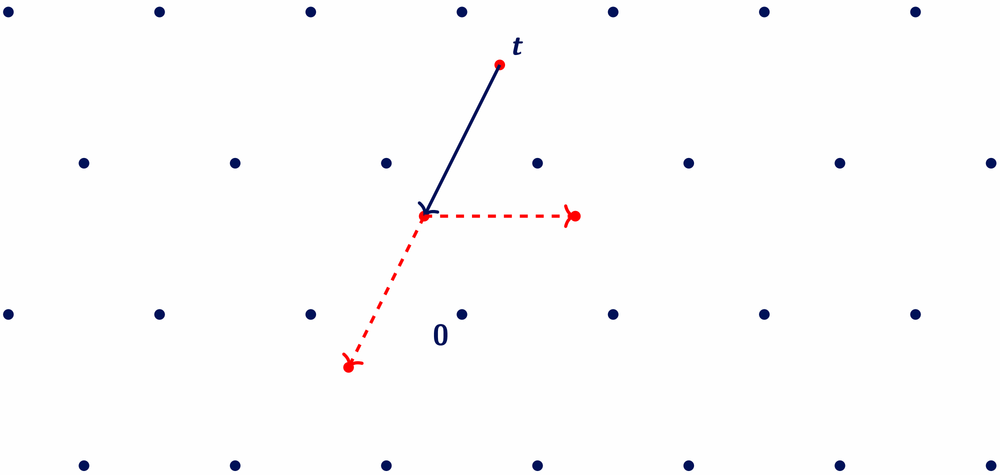


$L = \{\pm b_1, \pm b_2\}$

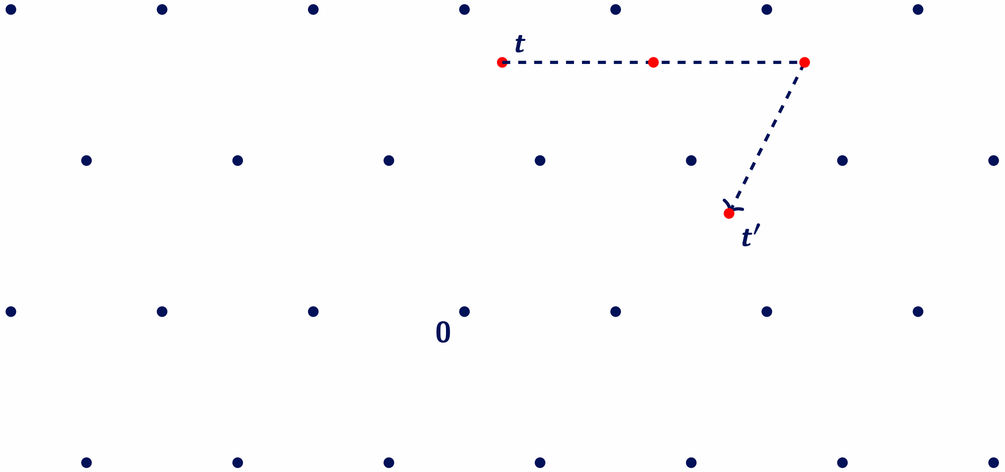
Randomized Iterative Slicer [Laa'16]



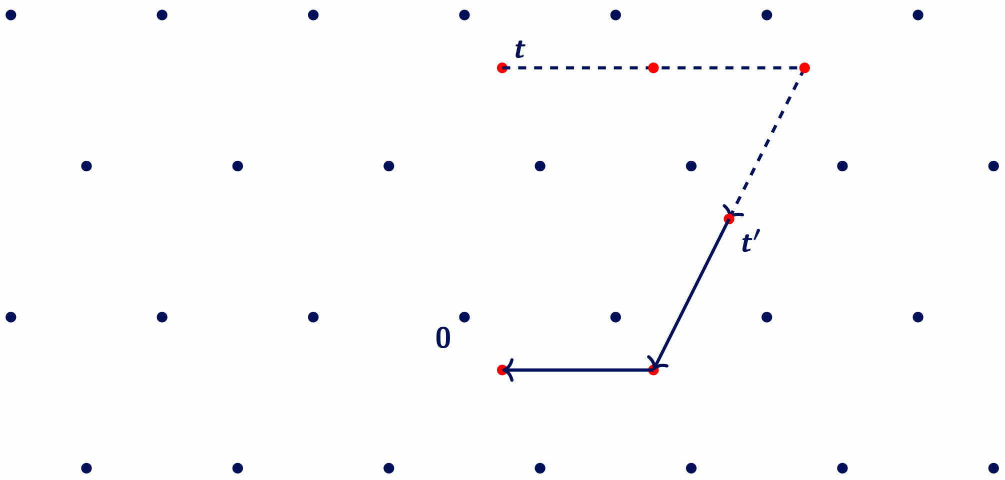
Randomized Iterative Slicer [Laa'16]



Randomized Iterative Slicer [Laa'16]

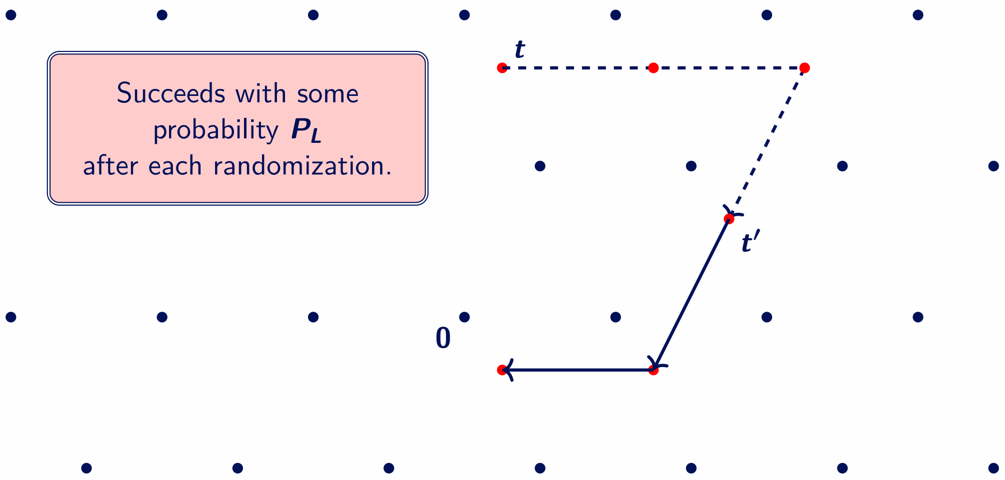


Randomized Iterative Slicer [Laa'16]



Randomized Iterative Slicer [Laa'16]

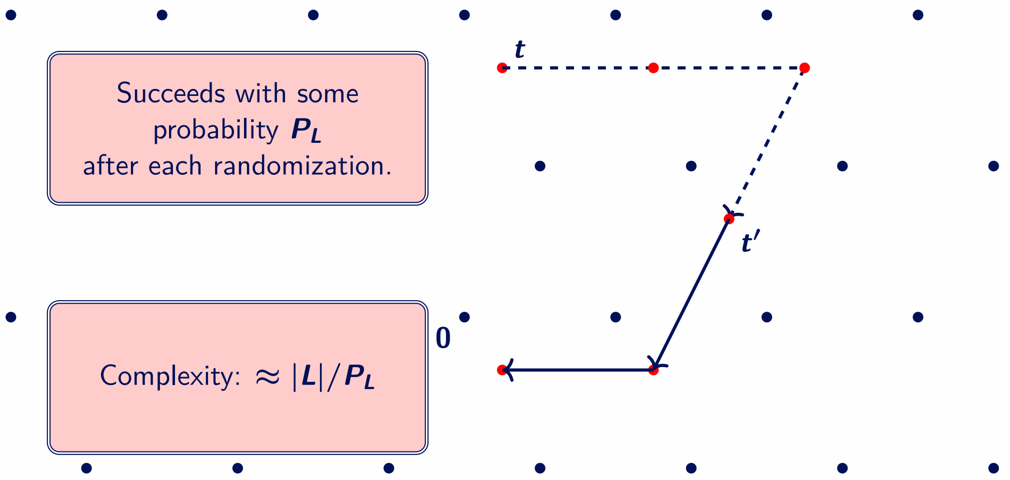
Succeeds with some probability P_L after each randomization.



Randomized Iterative Slicer [Laa'16]

Succeeds with some probability P_L after each randomization.

Complexity: $\approx |L|/P_L$



Preprocessing

- $\boldsymbol{L}_\alpha :=$ " the $\alpha^{\boldsymbol{d}+\boldsymbol{o}(\boldsymbol{d})}$ shortest nonzero vectors of $\boldsymbol{\mathcal{L}}$"

Preprocessing

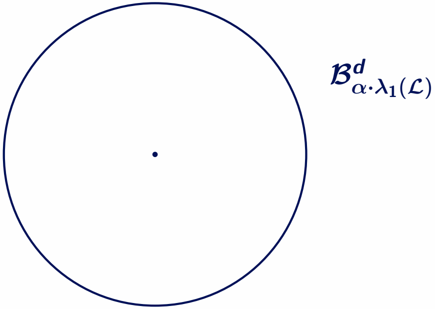
- L_α := " the $\alpha^{d+o(d)}$ shortest nonzero vectors of $\mathcal{L}$"
- Under the Gaussian Heuristic:

$$L_\alpha = \{x \in \mathcal{L} : \|x\| \leq \alpha \cdot \lambda_1(\mathcal{L})\} \setminus \{0\}.$$

Preprocessing

- $L_\alpha :=$ " the $\alpha^{d+o(d)}$ shortest nonzero vectors of $\mathcal{L}$"
- Under the Gaussian Heuristic:

$$L_\alpha = \{x \in \mathcal{L} : \|x\| \leq \alpha \cdot \lambda_1(\mathcal{L})\} \setminus \{0\}.$$

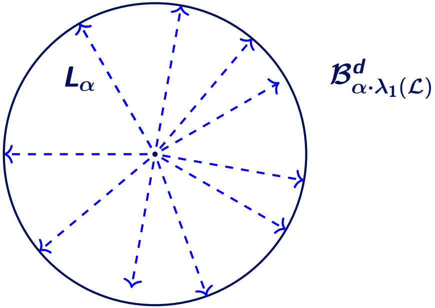


Preprocessing

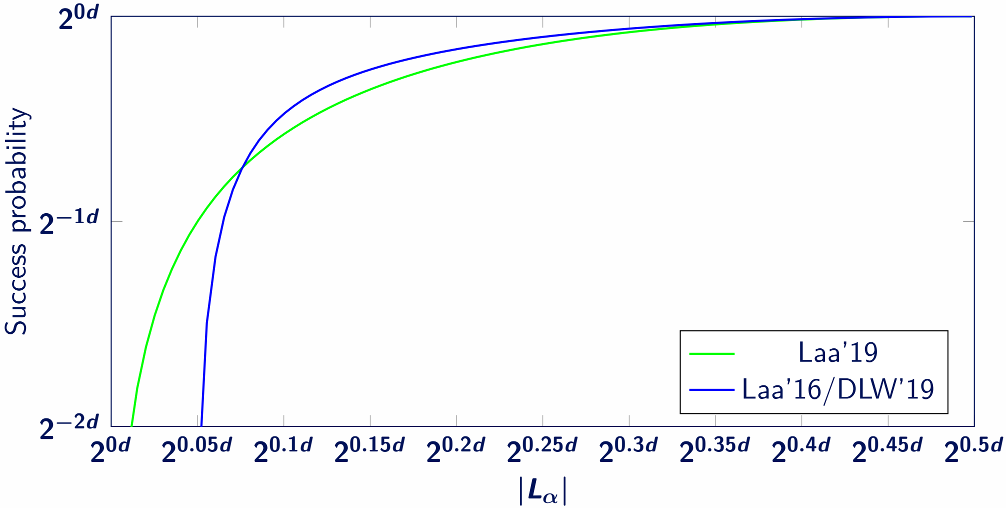
- $L_\alpha :=$ " the $\alpha^{d+o(d)}$ shortest nonzero vectors of $\mathcal{L}$"
- Under the Gaussian Heuristic:

$$L_\alpha = \{x \in \mathcal{L} : \|x\| \leq \alpha \cdot \lambda_1(\mathcal{L})\} \setminus \{0\}.$$

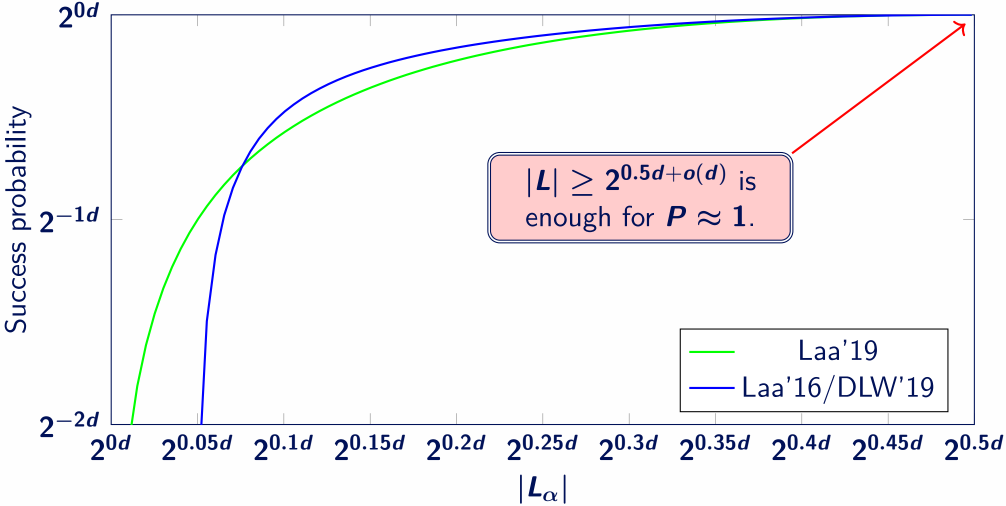
Uniformly distributed



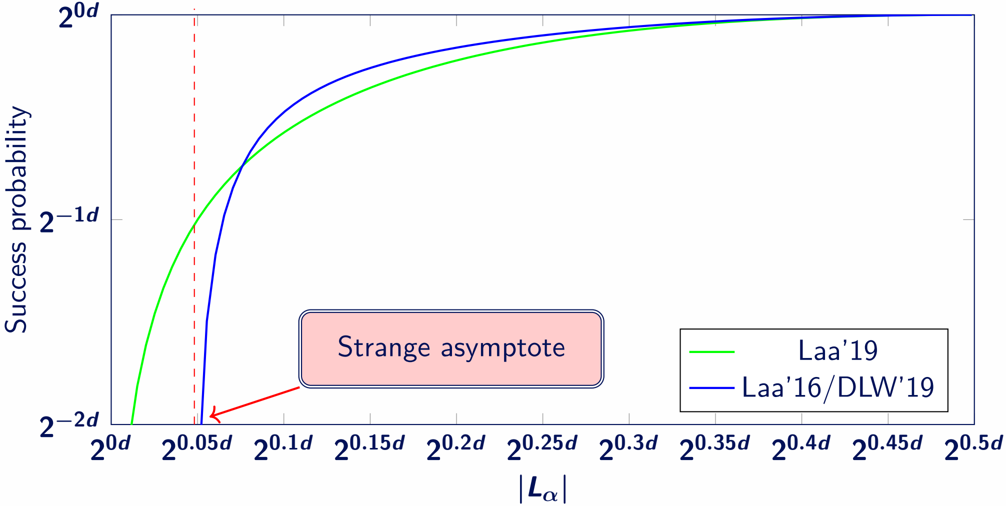
Success Probability - Lower bounds



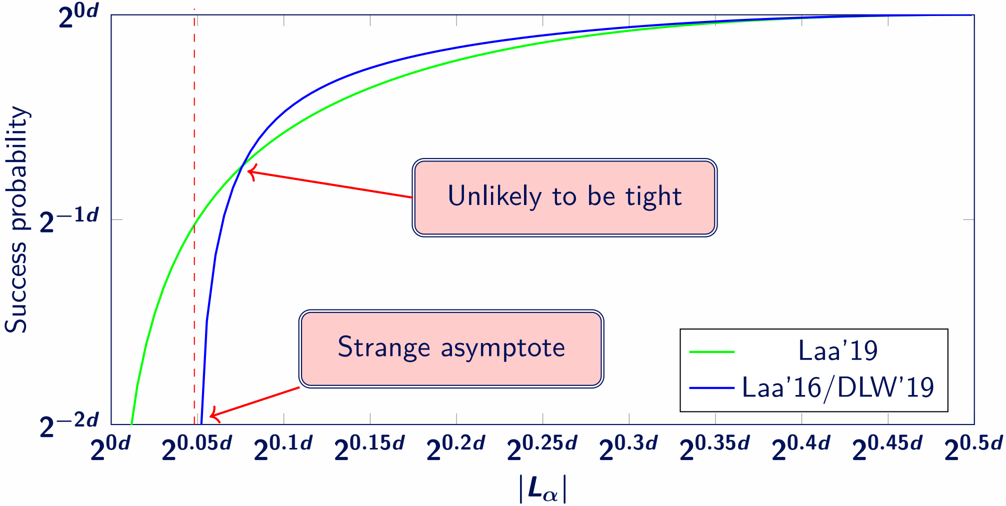
Success Probability - Lower bounds



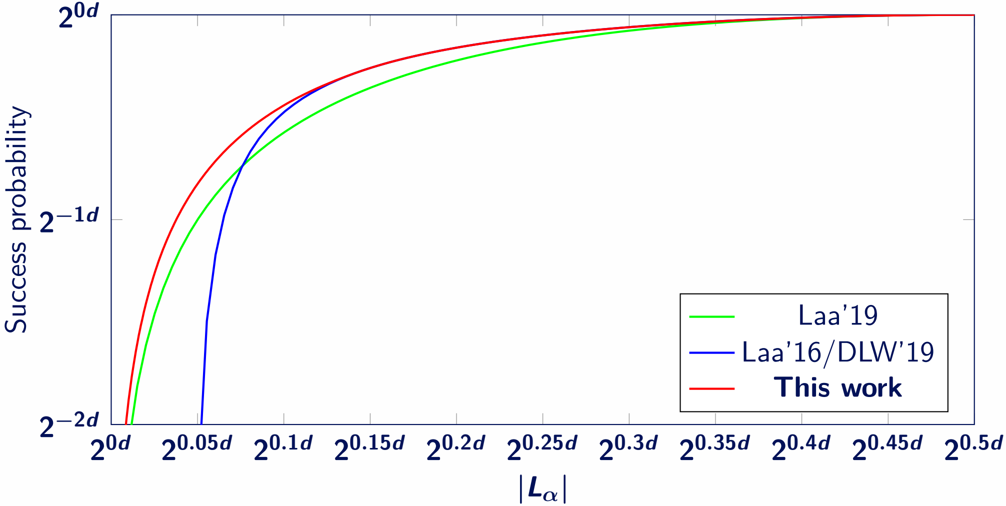
Success Probability - Lower bounds



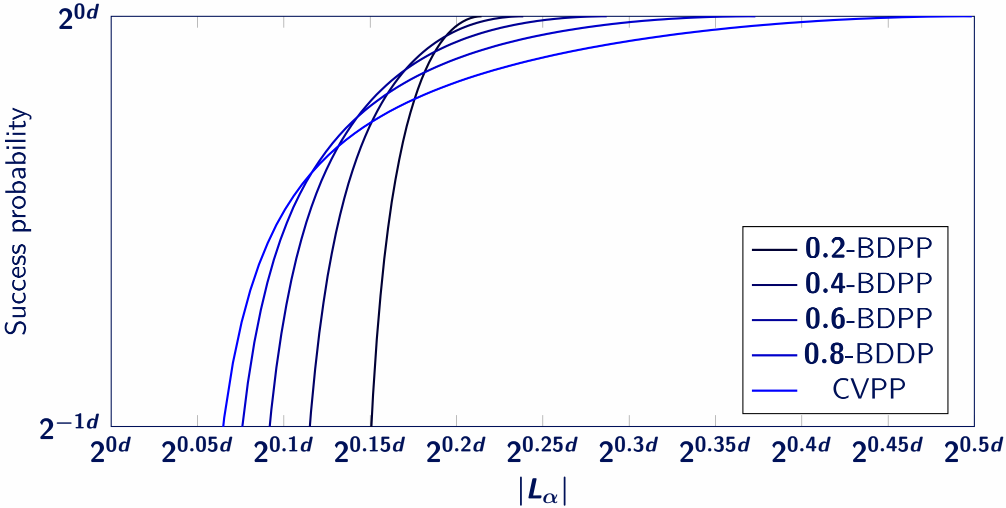
Success Probability - Lower bounds



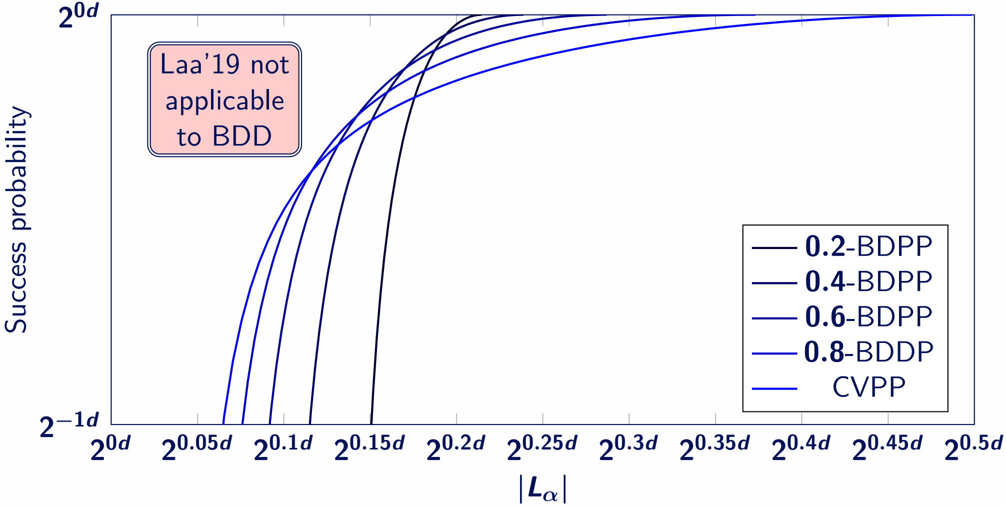
Success Probability - Lower bounds



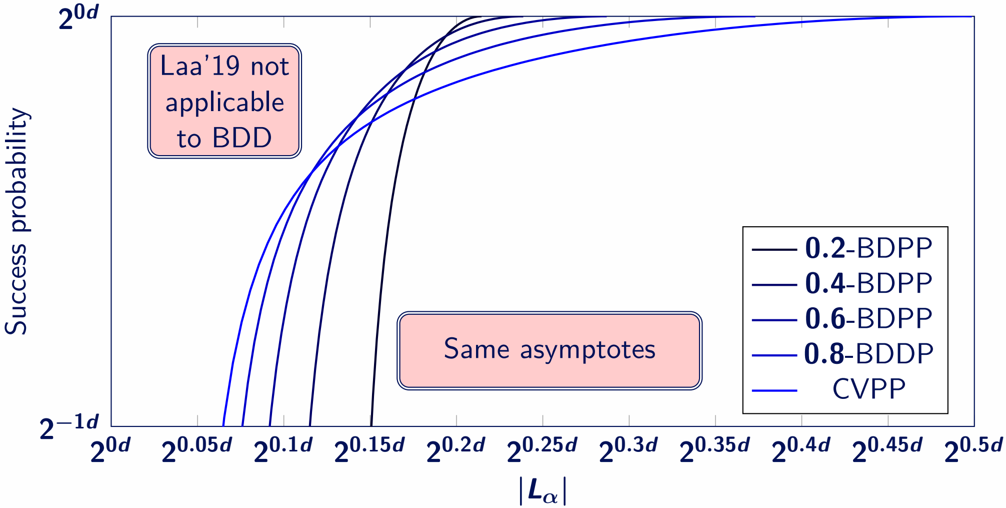
Bounded Distance Decoding [Laa'19]



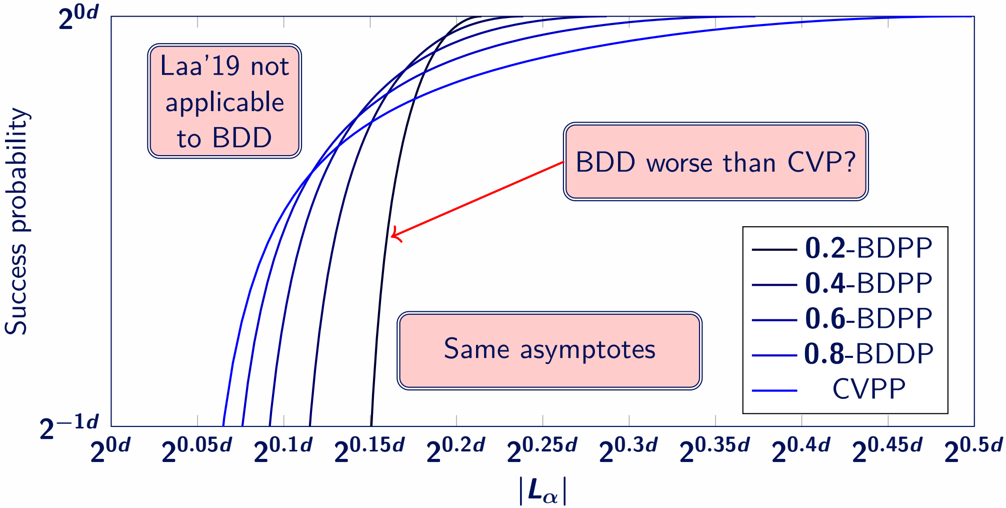
Bounded Distance Decoding [Laa'19]



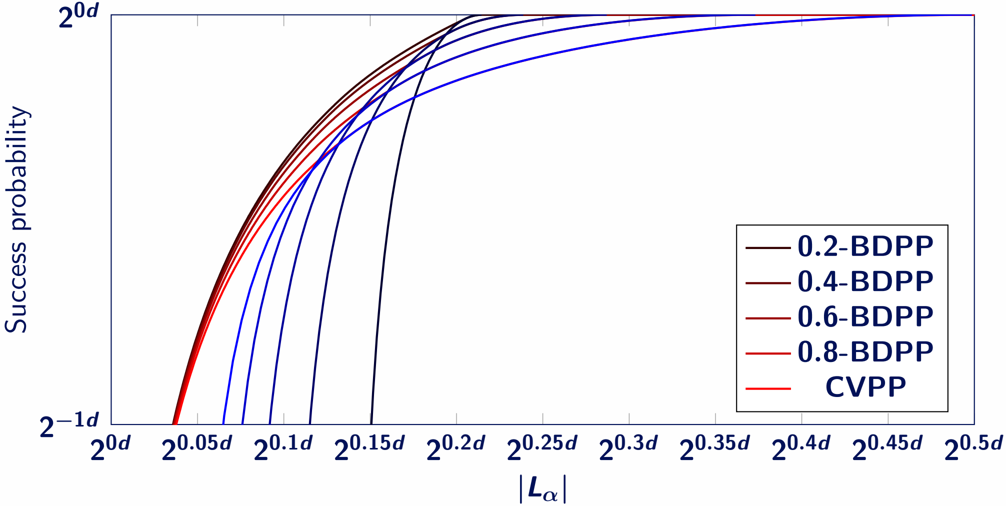
Bounded Distance Decoding [Laa'19]



Bounded Distance Decoding [Laa'19]



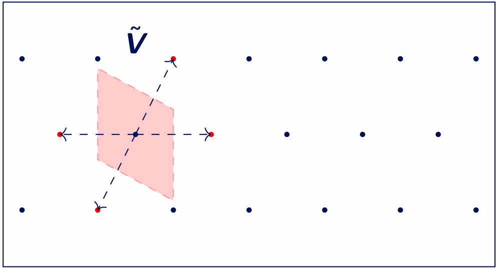
Bounded Distance Decoding [This work]



Known lower bounds

- Each lower bound looks at a different event.

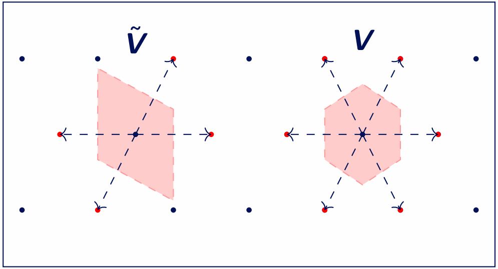
Approximate Voronoi Cell [Laa'19]



Known lower bounds

- Each lower bound looks at a different event.

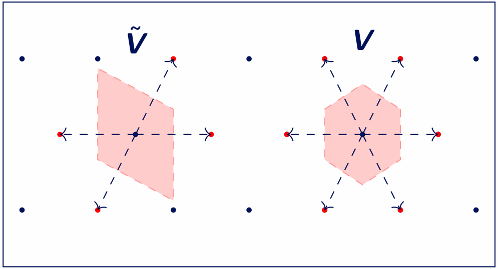
Approximate Voronoi Cell [Laa'19]



Known lower bounds

- Each lower bound looks at a different event.

Approximate Voronoi Cell [Laa'19]

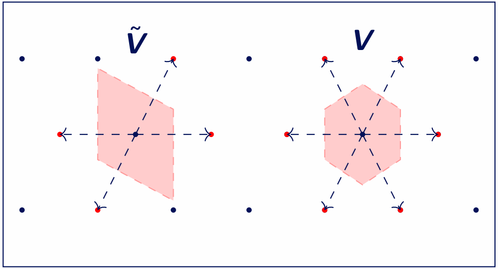


$$P_\alpha \geq \mathrm{Vol}(V)/\mathrm{Vol}(\tilde{V})$$

Known lower bounds

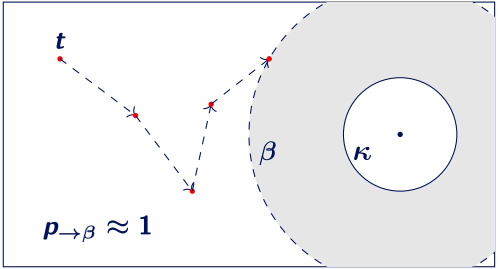
- Each lower bound looks at a different event.

Approximate Voronoi Cell [Laa'19]



$$P_\alpha \geq \mathrm{Vol}(V)/\mathrm{Vol}(\tilde{V})$$

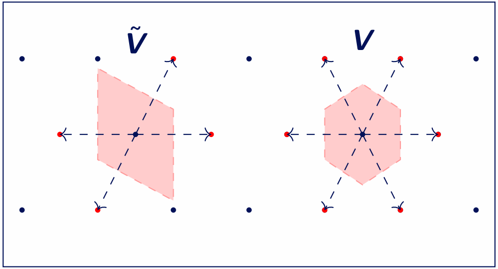
"**1**-step" analysis [DLW'19]



Known lower bounds

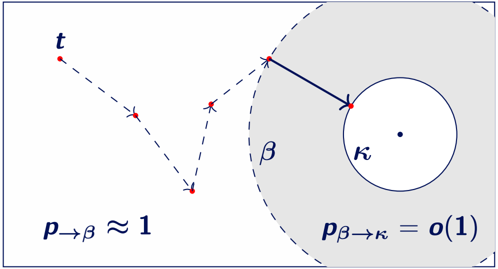
- Each lower bound looks at a different event.

Approximate Voronoi Cell [Laa'19]



$P_\alpha \geq \mathrm{Vol}(V)/\mathrm{Vol}(\tilde{V})$

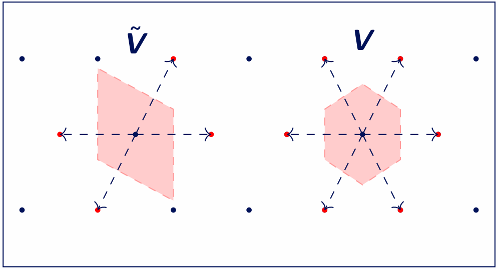
"1-step" analysis [DLW'19]



Known lower bounds

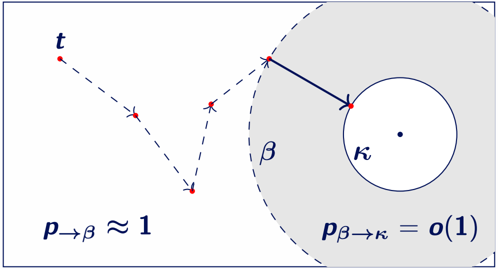
- Each lower bound looks at a different event.

Approximate Voronoi Cell [Laa'19]



$$P_\alpha \geq \mathrm{Vol}(V)/\mathrm{Vol}(\tilde{V})$$

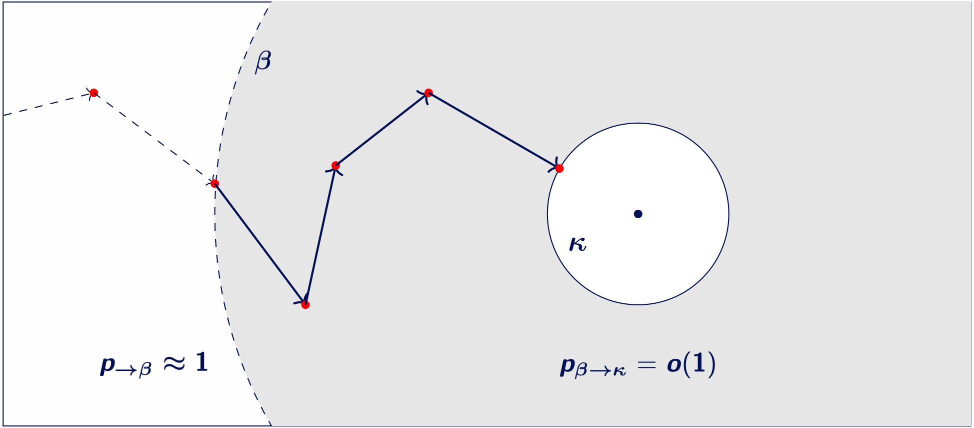
"**1**-step" analysis [DLW'19]



$$P_\alpha \geq p_{\to\beta} \cdot p_{\beta\to\kappa} \approx p_{\beta\to\kappa}$$

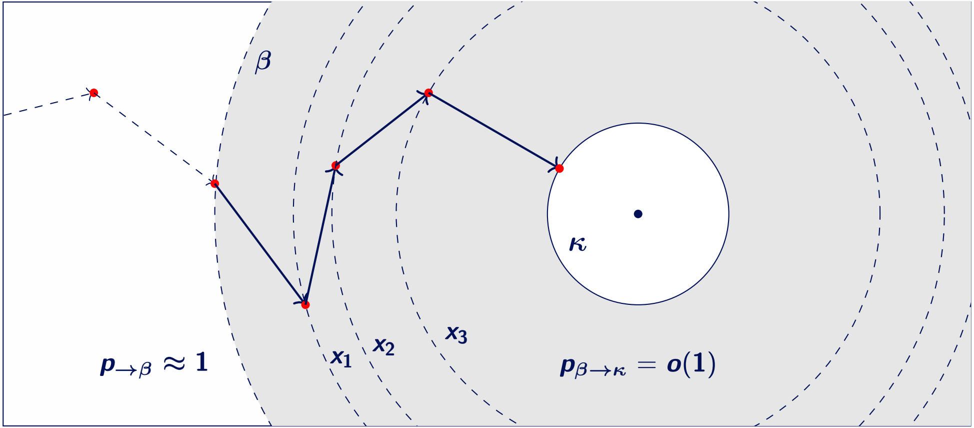
Multi-step analysis [This work]

- Model "all" events $\implies$ Random Walk.



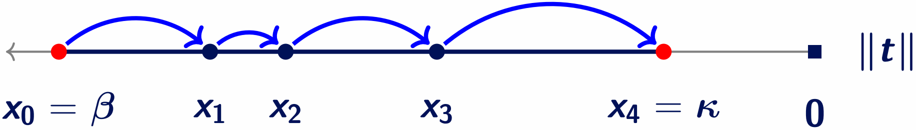
Multi-step analysis [This work]

- Model “all” events $\Longrightarrow$ Random Walk.



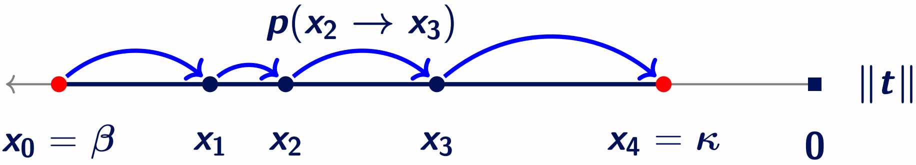
Multi-step analysis [This work]

- Model “all” events $\Longrightarrow$ Random Walk.
- Simplify by looking at the norm.



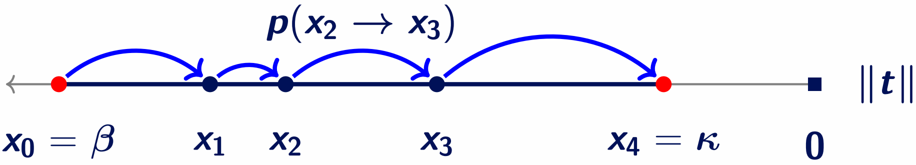
Multi-step analysis [This work]

- Model "all" events $\implies$ Random Walk.
- Simplify by looking at the norm.



Multi-step analysis [This work]

- Model "all" events $\implies$ Random Walk.
- Simplify by looking at the norm.



- This gives a lower bound

$$P \geq \prod_{i=1}^{4} p(x_{i-1} \to x_i).$$

Optimization Problem

- Each path $\boldsymbol{X}$: $\beta = \boldsymbol{x}_0 \rightarrow \boldsymbol{x}_1 \rightarrow \ldots \rightarrow \boldsymbol{x}_s = \boldsymbol{\kappa}$ gives a lower bound

$$\boldsymbol{P} \geq \boldsymbol{P}(\boldsymbol{X}) := \prod_{i=1}^{s} \boldsymbol{p}(\boldsymbol{x}_{i-1} \rightarrow \boldsymbol{x}_i).$$

Optimization Problem

- Each path $\boldsymbol{X}$: $\beta = x_0 \to x_1 \to \ldots \to x_s = \kappa$ gives a lower bound

$$P \geq P(X) := \prod_{i=1}^{s} p(x_{i-1} \to x_i).$$

- Each transition probability is of the form

$$p(y \to z) = 2^{-c(y \to z) \cdot d + o(d)}.$$

for some constant $c(y \to z)$ only depending on α, y and z.

Optimization Problem

- Each path $\boldsymbol{X}$: $\beta = x_0 \to x_1 \to \ldots \to x_s = \kappa$ gives a lower bound

$$P \geq P(X) := \prod_{i=1}^{s} p(x_{i-1} \to x_i).$$

- Each transition probability is of the form

$$p(y \to z) = 2^{-c(y \to z) \cdot d + o(d)}.$$

  for some constant $c(y \to z)$ only depending on α, y and z.

- To find the best path we have to solve

$$\min_{\text{path } X} C(X) := \sum c(x_{i-1} \to x_i).$$

Optimization Problem

- Each path $\boldsymbol{X}$: $\beta = x_0 \to x_1 \to \ldots \to x_s = \kappa$ gives a lower bound

$$P \geq P(X) := \prod_{i=1}^{s} p(x_{i-1} \to x_i).$$

- Each transition probability is of the form

$$p(y \to z) = 2^{-c(y \to z) \cdot d + o(d)}.$$

  for some constant $c(y \to z)$ only depending on α, y and z.

- To find the best path we have to solve

$$\min_{\text{path } X} C(X) := \sum c(x_{i-1} \to x_i).$$

- Formal analysis using densities makes this bound tight (up to $2^{o(d)}$)

Convex optimization

- $\boldsymbol{C}(\boldsymbol{X})$ is a strictly convex function for paths of fixed length.

Convex optimization

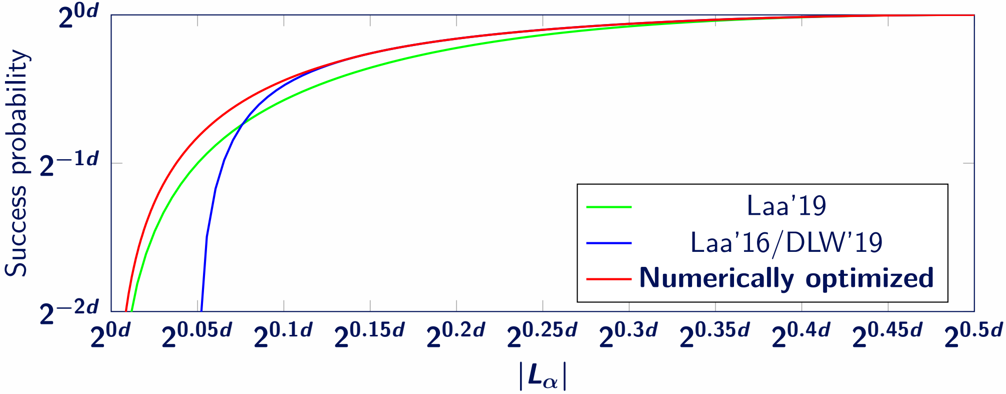
- $\boldsymbol{C}(\boldsymbol{X})$ is a strictly convex function for paths of fixed length.
- Unique local and global minimum.

Convex optimization

- $\boldsymbol{C}(\boldsymbol{X})$ is a strictly convex function for paths of fixed length.
- Unique local and global minimum.
- Easy to optimize numerically.

Convex optimization

- $\boldsymbol{C}(\boldsymbol{X})$ is a strictly convex function for paths of fixed length.
- Unique local and global minimum.
- Easy to optimize numerically.



A (surprising) analytic Solution

- Construct local optimal s-step path analytically.

A (surprising) analytic Solution

- Construct local optimal $\boldsymbol{s}$-step path analytically.
- Use known constraints:

$$\frac{\partial}{\partial \boldsymbol{x_i}} \boldsymbol{C}(\boldsymbol{X}) = \mathbf{0} \text{ for all } \mathbf{1} \leq \boldsymbol{i} < \boldsymbol{s},$$

$$\boldsymbol{x_0} = \boldsymbol{\beta}, \boldsymbol{x_s} = \boldsymbol{\kappa}.$$

A (surprising) analytic Solution

- Construct local optimal $\boldsymbol{s}$-step path analytically.
- Use known constraints:

$$\frac{\partial}{\partial \boldsymbol{x_i}} \boldsymbol{C}(\boldsymbol{X}) = \mathbf{0} \text{ for all } \mathbf{1} \leq \boldsymbol{i} < \boldsymbol{s},$$

$$\boldsymbol{x_0} = \beta, \boldsymbol{x_s} = \boldsymbol{\kappa}.$$

- Express solution in terms of $\alpha, \boldsymbol{\kappa}$ using symbolic algebra.

A (surprising) analytic Solution

- Construct local optimal s-step path analytically.
- Use known constraints:

$$\frac{\partial}{\partial x_i} C(X) = 0 \text{ for all } 1 \leq i < s,$$

$$x_0 = \beta, x_s = \kappa.$$

- Express solution in terms of α, κ using symbolic algebra.
- Show which path length s is optimal.

A (surprising) analytic Solution

Theorem (Optimal path)

The path $\boldsymbol{X} : \beta \to x_1 \to \ldots \to x_s = \kappa$ *that minimizes* $\boldsymbol{C}(\boldsymbol{X})$ *consists of*

$$s = \left\lceil -\frac{1}{2} + \frac{1}{2\alpha^2}\sqrt{(4\beta^2 - \alpha^2)^2 - 8(2\beta^2 - \alpha^2)\kappa^2} \right\rceil$$

steps, and is for $s > 1$ *given by* $x_i = \sqrt{u \cdot i^2 + v \cdot i + \beta^2}$, *with*

$$u := \frac{(\beta^2 + \kappa^2 - \alpha^2)s - \sqrt{(\alpha^2 s^2 - (\beta^2 + \kappa^2)) + 4\beta^2\kappa^2(s^2 - 1)}}{s^3 - s}$$

$$v := \frac{(\alpha^2 - 2\beta^2)s^2 + (\beta^2 - \kappa^2) + \sqrt{(\alpha^2 s^2 - (\beta^2 + \kappa^2)) + 4\beta^2\kappa^2(s^2 - 1)}s}{s^3 - s}.$$

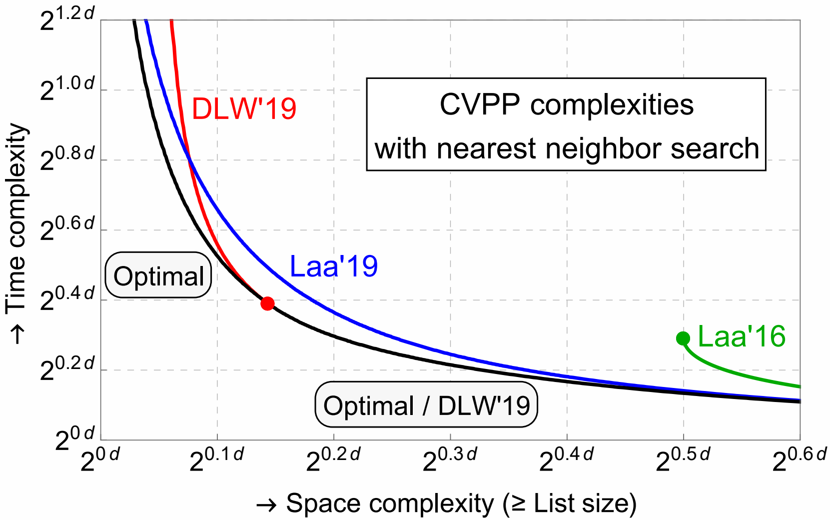
Nearest Neighbor Search (NNS)

- We still need to iterate over $|\boldsymbol{L}|$ vectors per reduction step, cost $\tilde{\boldsymbol{O}}(|\boldsymbol{L}|)$.

Nearest Neighbor Search (NNS)

- We still need to iterate over $|\boldsymbol{L}|$ vectors per reduction step, cost $\tilde{\boldsymbol{O}}(|\boldsymbol{L}|)$.
- NNS data structures reduce this, at the cost of more memory.

New CVPP time-memory trade-off



Memoryless NNS for batch CVPP

- Memoryless NNS: No memory overhead if we process a batch of at least $|\boldsymbol{L}|$ targets.

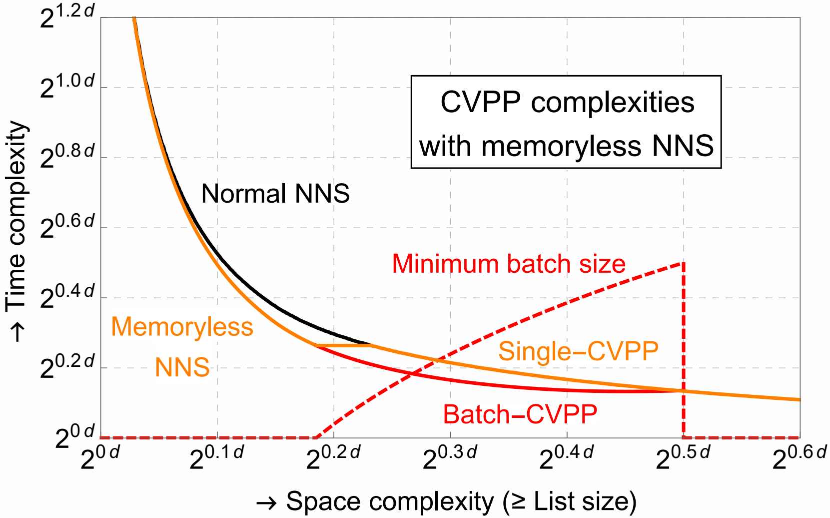
Memoryless NNS for batch CVPP

- Memoryless NNS: No memory overhead if we process a batch of at least $|\boldsymbol{L}|$ targets.
- Each CVPP target already gives us $\approx \boldsymbol{1/P}$ rerandomized targets.

Memoryless NNS for batch CVPP

- Memoryless NNS: No memory overhead if we process a batch of at least $|L|$ targets.
- Each CVPP target already gives us $\approx 1/P$ rerandomized targets.
- Batches of size $\min\{1, P \cdot |L|\}$ are enough.

Further improvements using memoryless NNS



Conclusion

"The randomized slicer for CVPP: sharper, faster, smaller, batchier"

Conclusion

"The randomized slicer for CVPP: sharper, faster, smaller, batchier"

- **Sharper:** Full understanding of the asymptotic behaviour of the Iterative Slicer, leading to a tight bound on the success probability.

Conclusion

"The randomized slicer for CVPP: sharper, faster, smaller, batchier"

- **Sharper:** Full understanding of the asymptotic behaviour of the Iterative Slicer, leading to a tight bound on the success probability.
- **Faster:** We obtain better time-memory trade-offs for CVPP.

Conclusion

"The randomized slicer for CVPP: sharper, faster, smaller, batchier"

- **Sharper:** Full understanding of the asymptotic behaviour of the Iterative Slicer, leading to a tight bound on the success probability.
- **Faster:** We obtain better time-memory trade-offs for CVPP.
- **Smaller:** We decrease the memory requirement for NNS, even for a single CVPP instance.

Conclusion

"The randomized slicer for CVPP: sharper, faster, smaller, batchier"

- **Sharper:** Full understanding of the asymptotic behaviour of the Iterative Slicer, leading to a tight bound on the success probability.
- **Faster:** We obtain better time-memory trade-offs for CVPP.
- **Smaller:** We decrease the memory requirement for NNS, even for a single CVPP instance.
- **Batchier:** We significantly improve on the per-target time complexities for batch-CVPP.

Bibliography

- N. Sommer, M. Feder and O. Shalvi, 2009. Finding the closest lattice point by iterative slicing. SIAM Journal on Discrete Mathematics, 23(2), pp.715-731.
- T. Laarhoven, 2016, August. Sieving for closest lattice vectors (with preprocessing). In International Conference on Selected Areas in Cryptography (pp. 523-542). Springer, Cham.
- E. Doulgerakis, T. Laarhoven and B. de Weger, 2019. Finding closest lattice vectors using approximate Voronoi cells. PQCRYPTO.
- T. Laarhoven, 2019. Approximate Voronoi cells for lattices, revisited. In: Proceedings of the 1st MATHCRYPT.

Thank you!