Sieving for Shortest Vectors in Ideal Lattices: a Practical Perspective

Joppe W. Bos

NXP

LACAL@RISC Seminar on Cryptologic Algorithms
CWI, Amsterdam, Netherlands

Joint work with Michael Naehrig and Joop van de Pol

Microsoft Research

University of BRISTOL

Sieving for Shortest Vectors in Ideal Lattices: a Practical Perspective

Joppe W. Bos

(Ex-LACAL, April 2007 - February 2012)

LACAL@RISC Seminar on Cryptologic Algorithms
CWI, Amsterdam, Netherlands

Joint work with Michael Naehrig and Joop van de Pol

Microsoft Research

University of BRISTOL

Motivation

- Shortest Vector Problem (SVP) used as a theoretical foundation in many PQ-crypto schemes
  - Lattice based encryption / signature schemes, fully homomorphic encryption
  - Often compute in an ideal lattice for performance reasons

$$R = \mathbb{Z}[X]/(X^n + 1)$$

- Exact SVP is known to be NP-complete
  (In most applications approximations are enough)

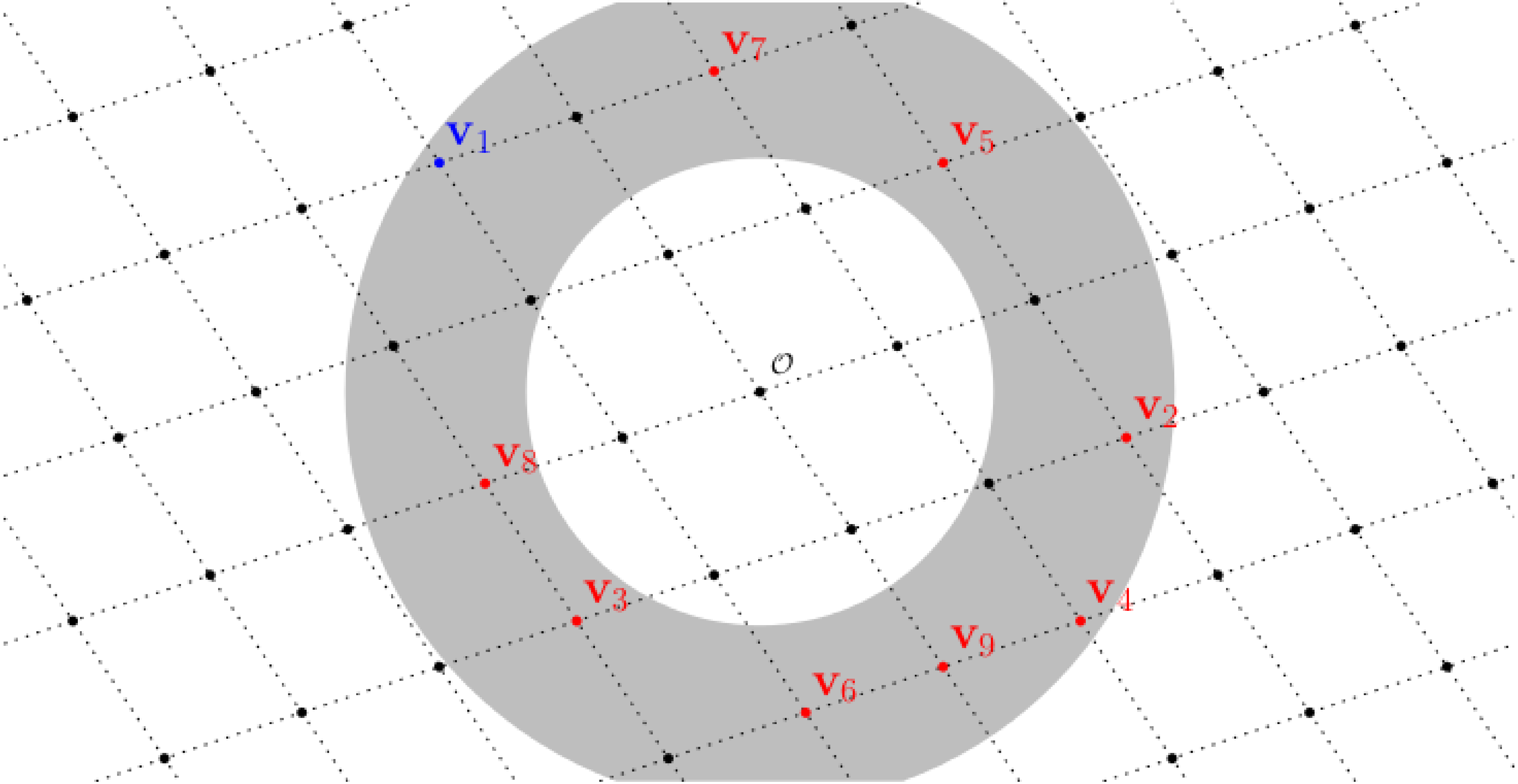
- How efficient can we find short vectors in ideal lattices?

SVP solvers

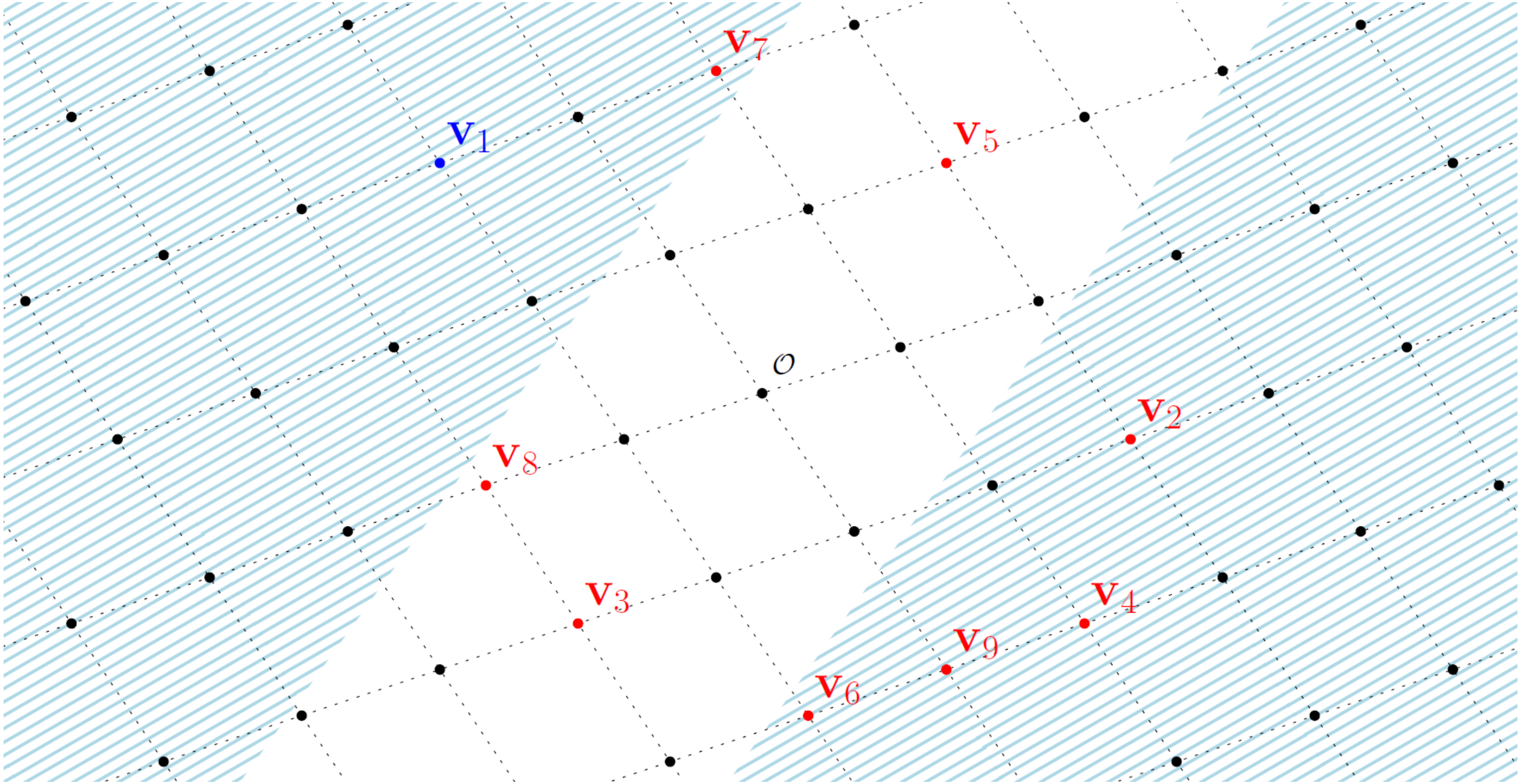
Asymptotic rigorous proven runtimes (ignoring poly-log factors in the exponent)		
	Time	Memory
Voronoi	2^{2n}	2^{n}
List Sieve	$2^{2.465n}$	$2^{1.233n}$
Enumeration	$2^{O(n\log(n))}$	$\mathrm{poly}(n)$

Asymptotic heuristic runtimes		
BKZ 2.0	$n \cdot N \cdot \mathrm{svp}(k)$	$\mathrm{poly}(n)$
+ Enumeration with extreme pruning	$n \cdot N \cdot 2^{O(k^2)}$	$\mathrm{poly}(n)$
Gauss Sieve	"$2^{0.48n}$"	$2^{0.2075n}$
Decomposition	$2^{0.3374n}$	$2^{0.2925n}$
Voronoi	"up to dimension 8"	

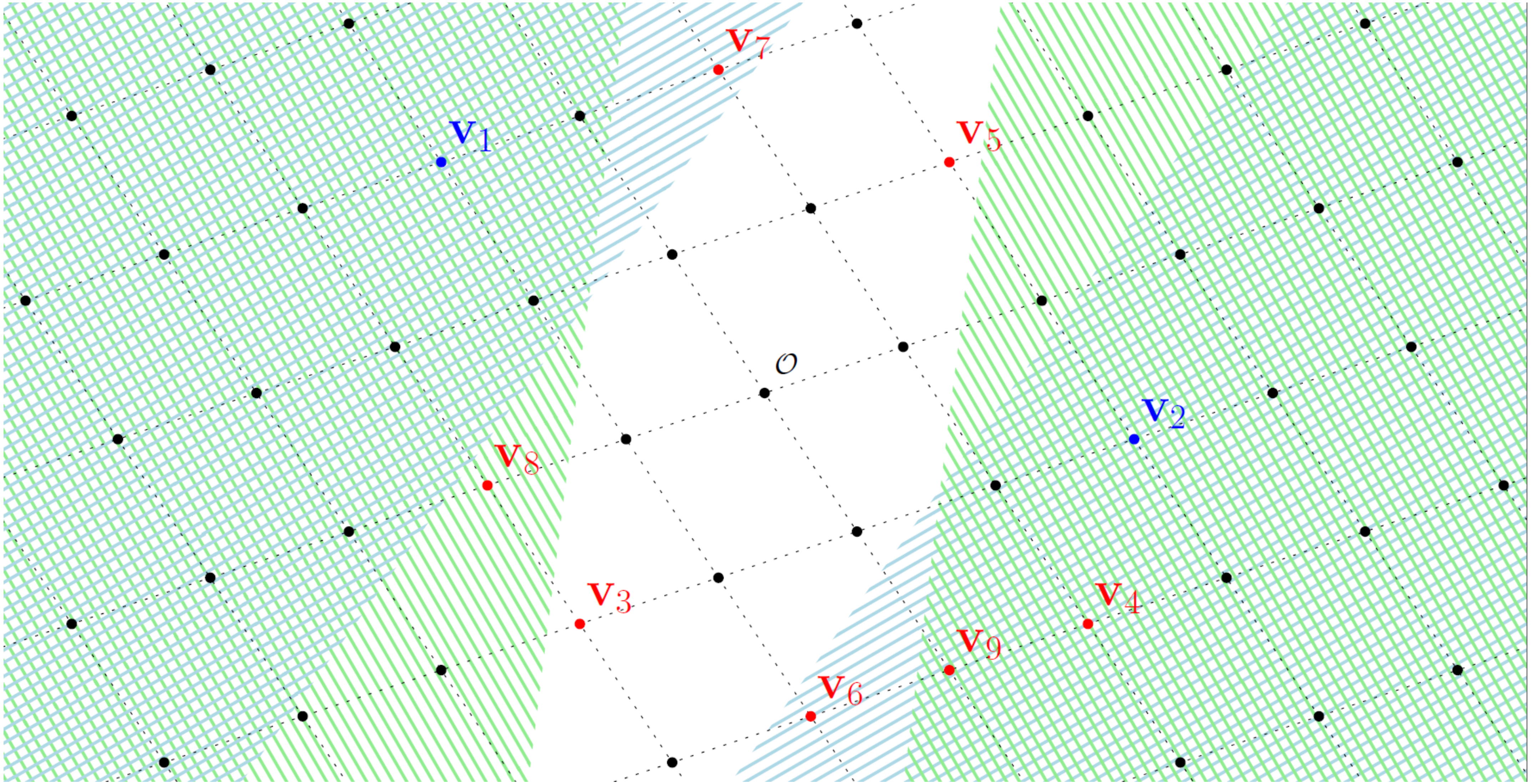
Only sieving algorithms take advantage of the ideal lattice structure



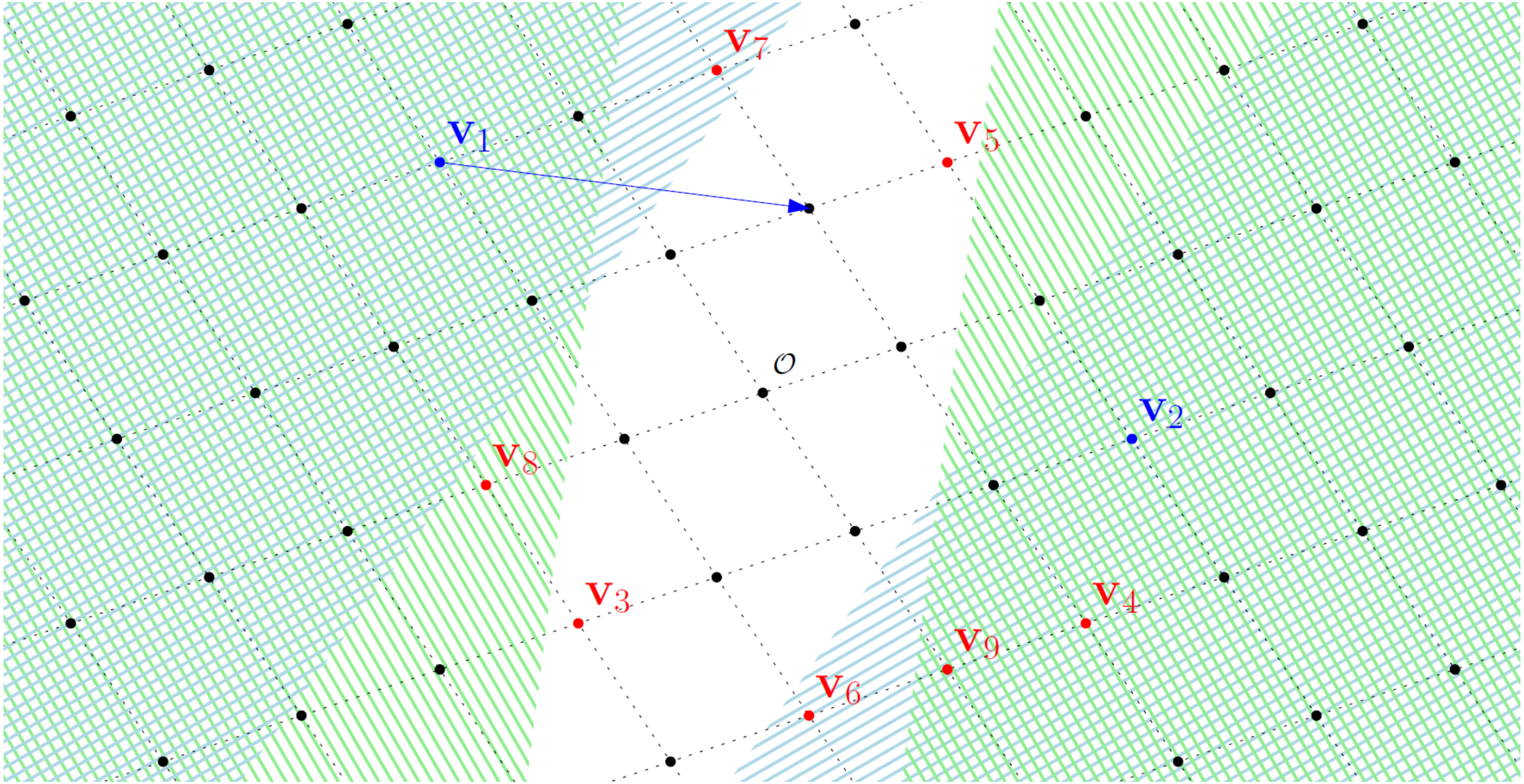
Sample a list of vectors and Gauss reduce all vectors with respect to each other



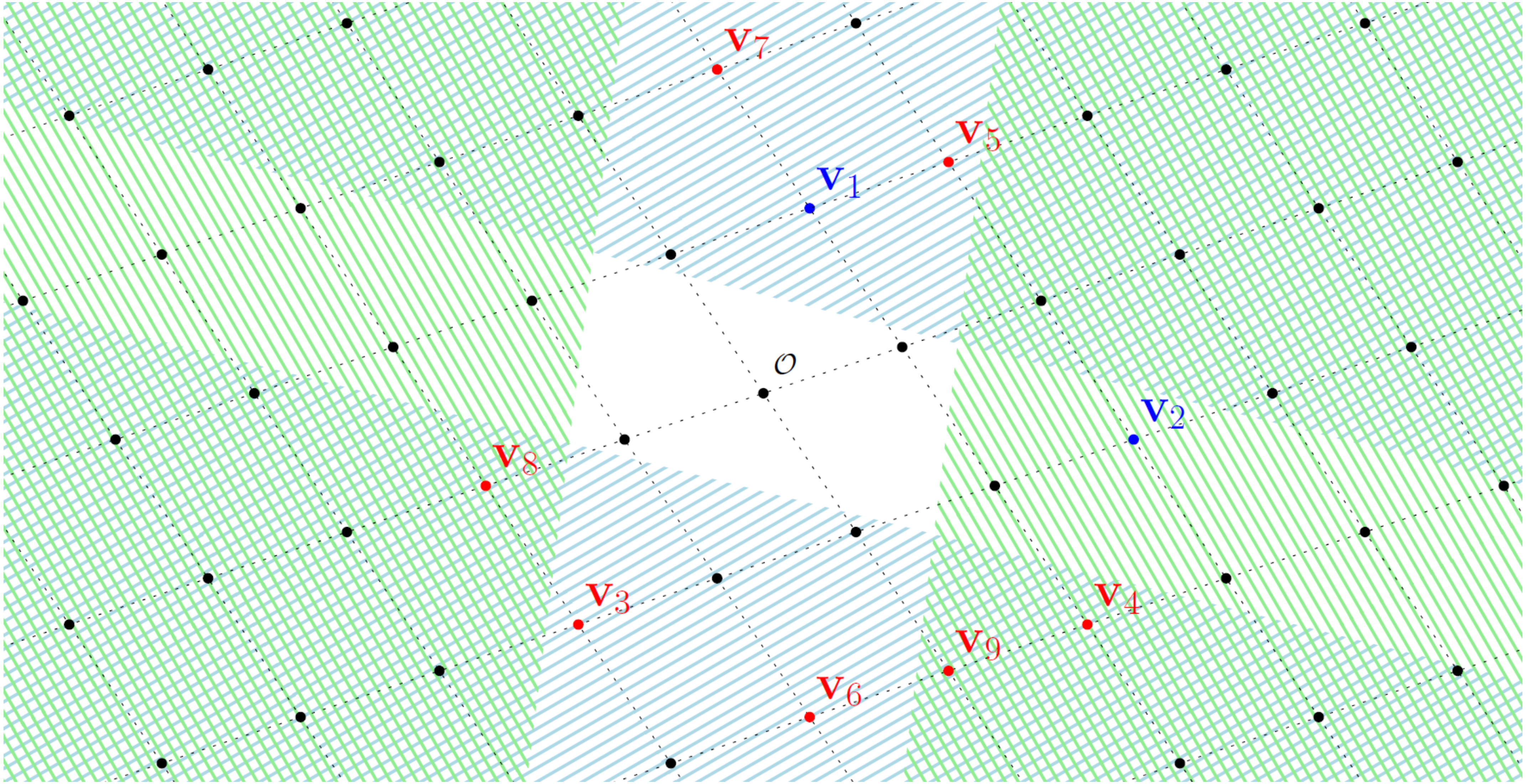
Each vector corresponds to two half spaces.
If a vector is in half-space of another previous vector, it can be reduced.



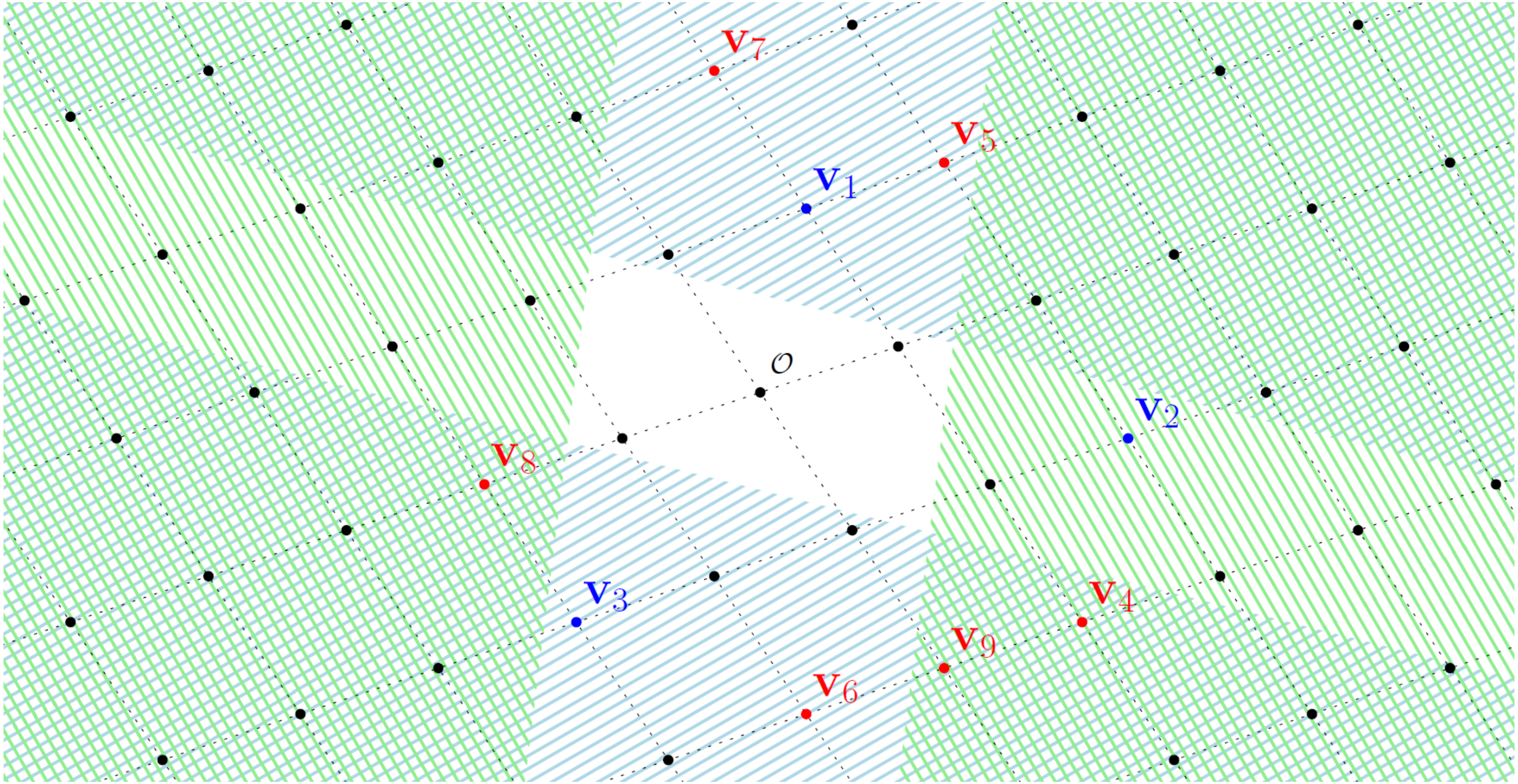
Each vector corresponds to two half spaces.
If a vector is in a half-space of another previous vector, it can be reduced.



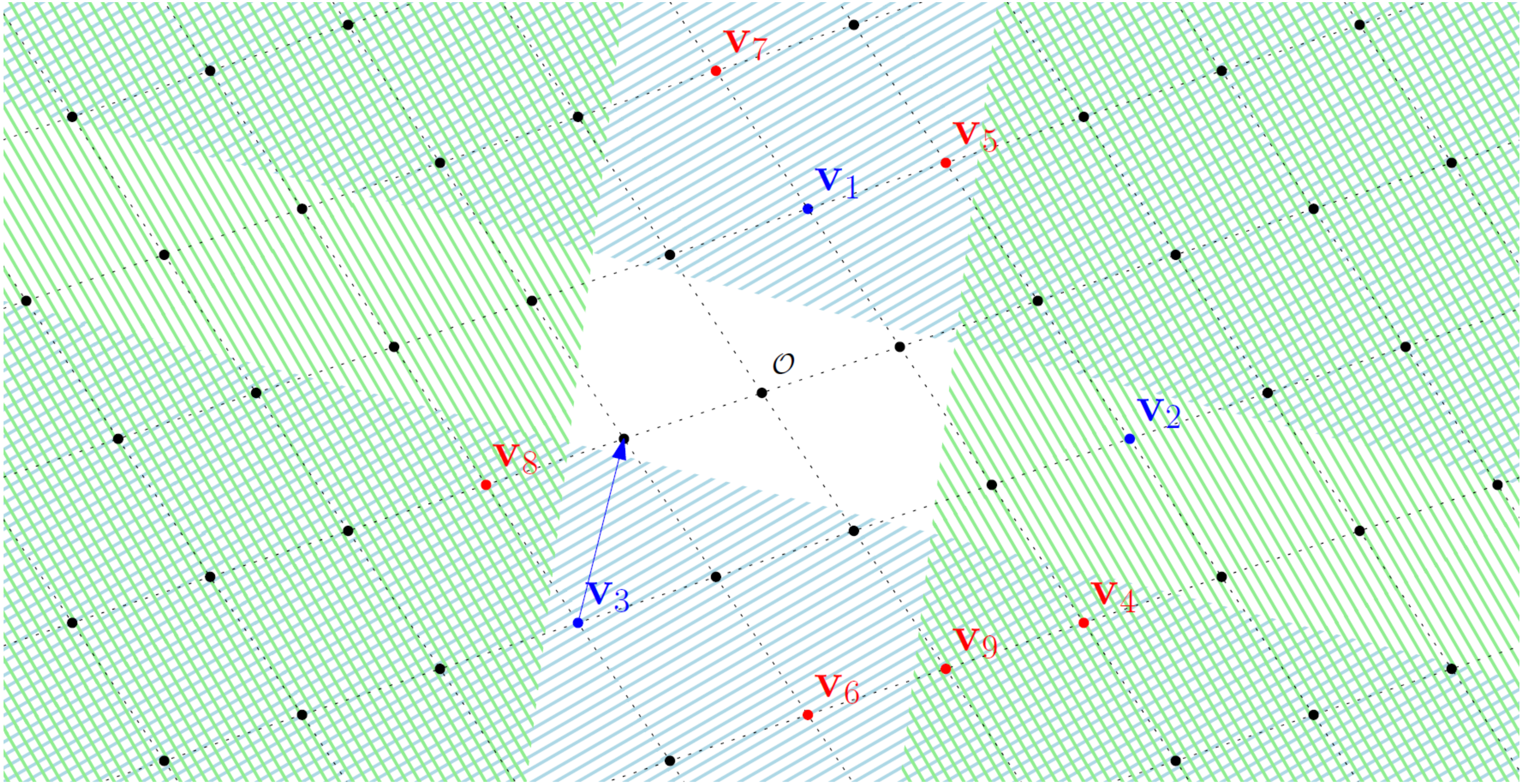
When two vectors can reduce each other, the shorter one reduces the longer one.



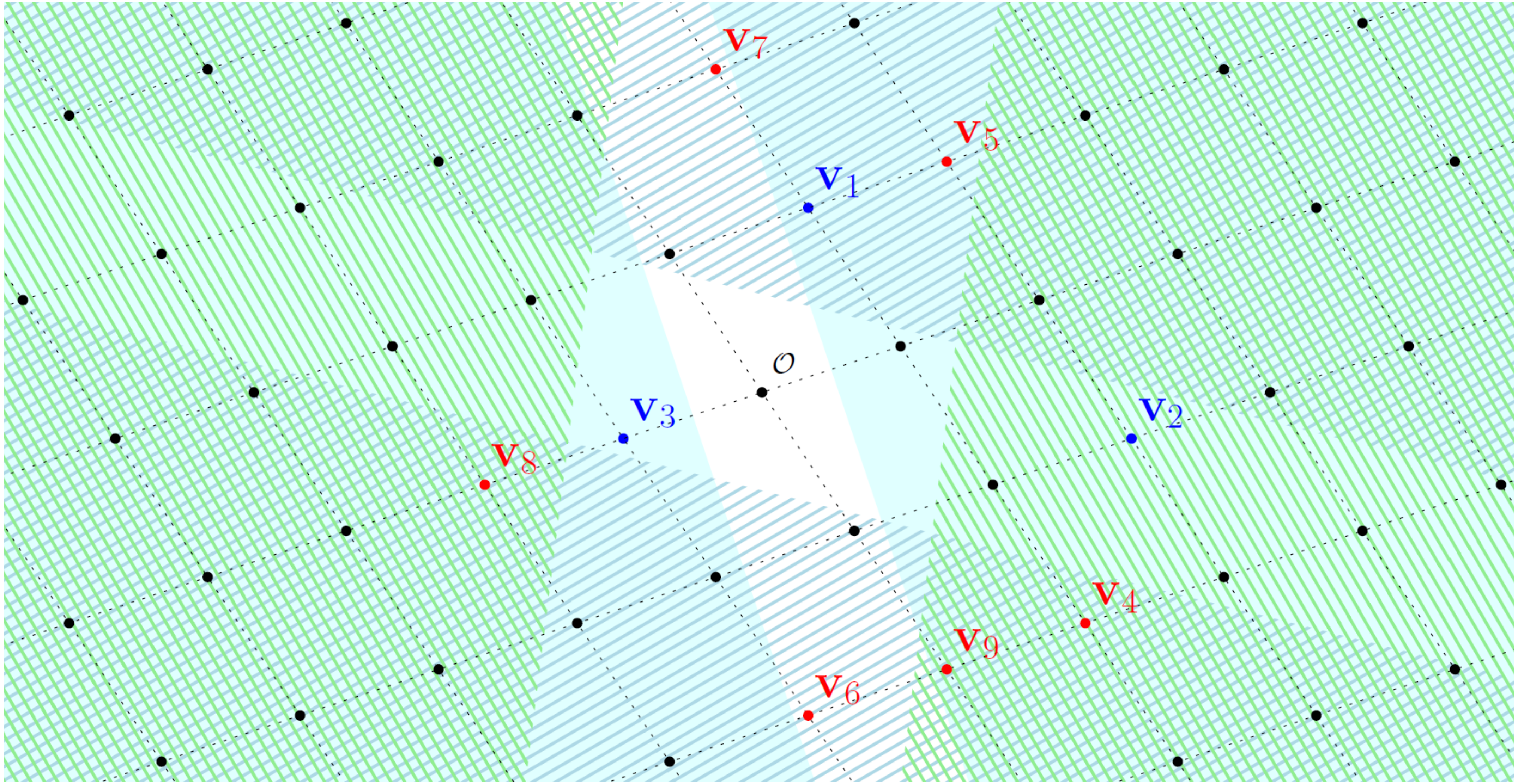
When two vectors can reduce each other, the shorter one reduces the longer one. The half-spaces increasingly cover more space.



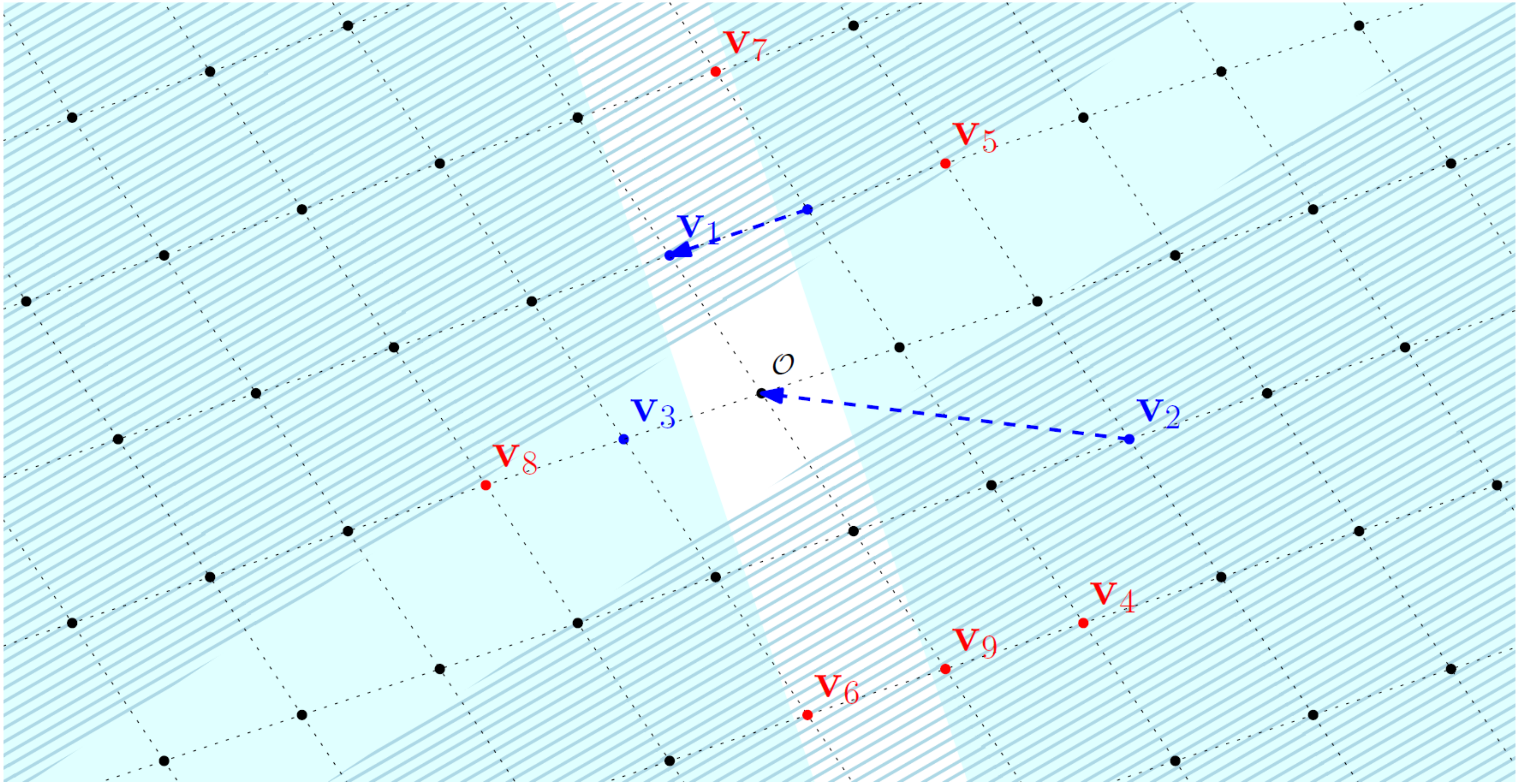
All vectors become pairwise Gauss reduced.



All vectors become pairwise Gauss reduced and the list consists of shorter and shorter vectors.

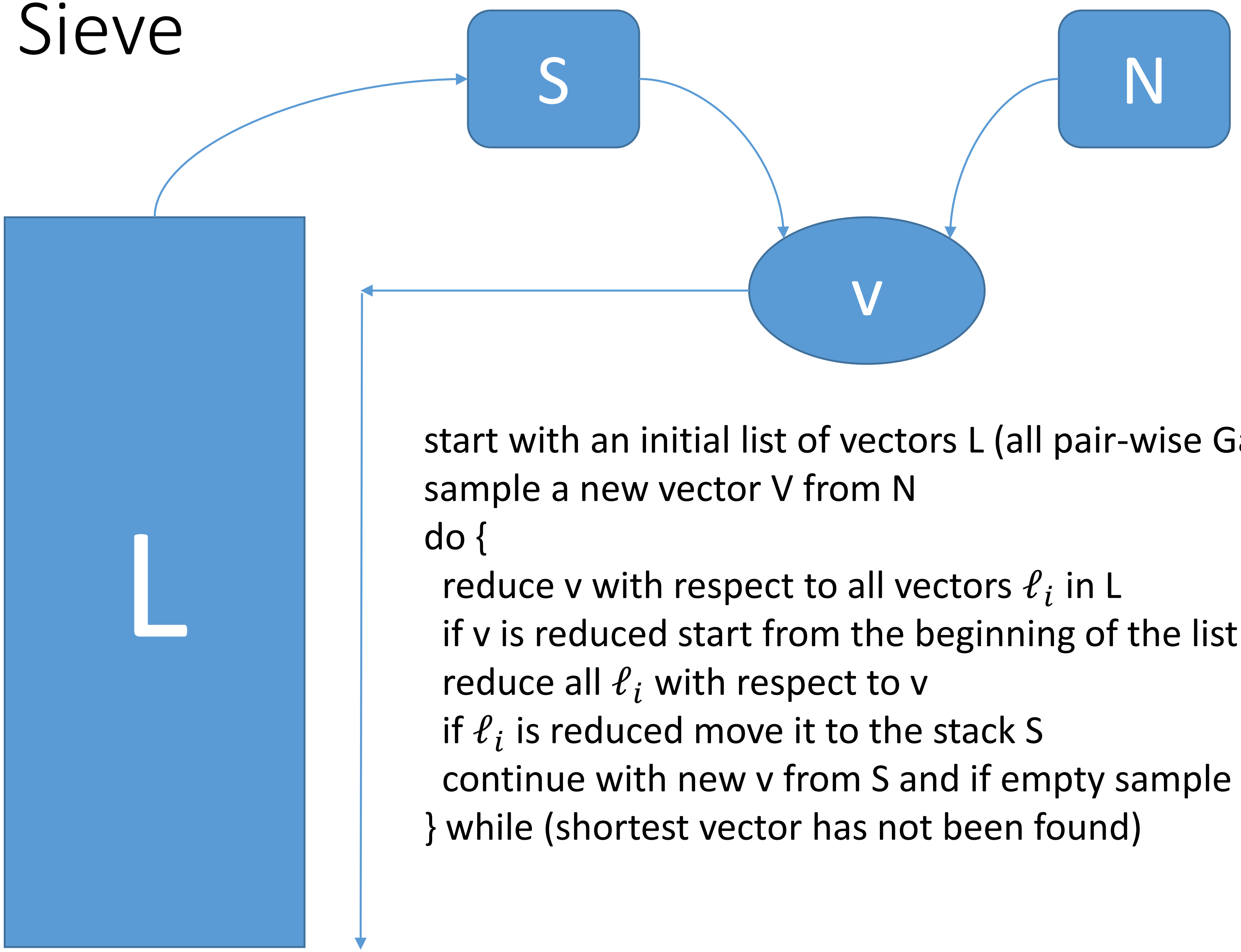


Repeat until we find a short vector or enough collisions.



Repeat until we find a short vector or enough collisions.
Nothing can be proven about the collisions.

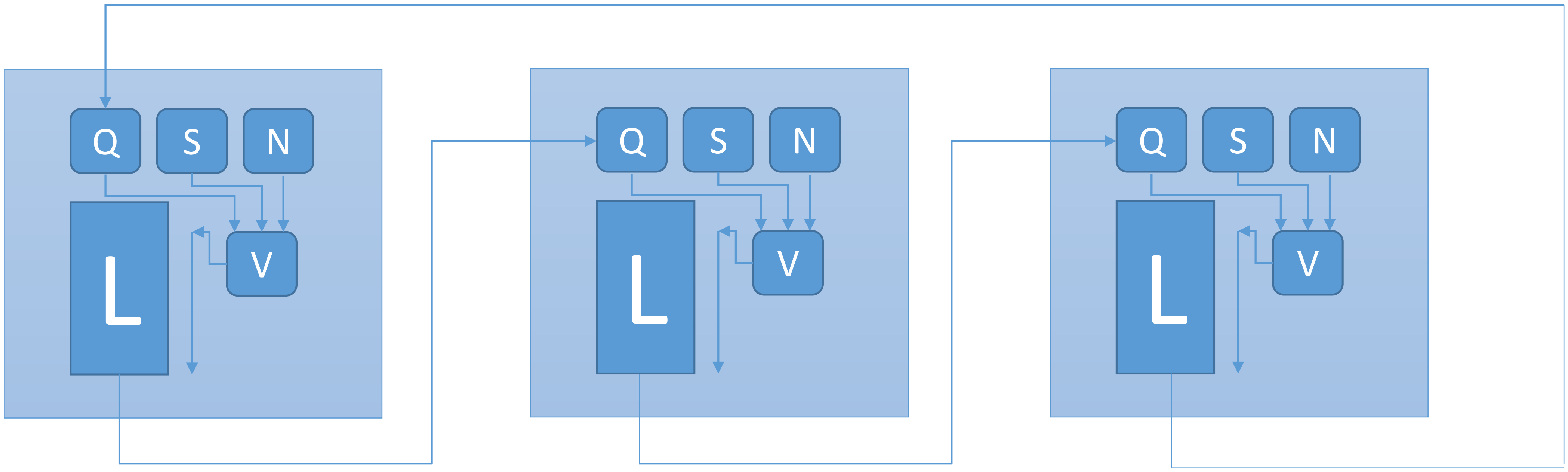
Gauss Sieve



start with an initial list of vectors L (all pair-wise Gauss reduced)
sample a new vector V from N
do {
  reduce v with respect to all vectors ℓ_i in L
  if v is reduced start from the beginning of the list L
  reduce all ℓ_i with respect to v
  if ℓ_i is reduced move it to the stack S
  continue with new v from S and if empty sample a new one from N
} while (shortest vector has not been found)

P. Voulgaris and D. Micciancio. Faster exponential time algorithms for the shortest vector problem. Electronic Colloquium on Computational Complexity, 2009

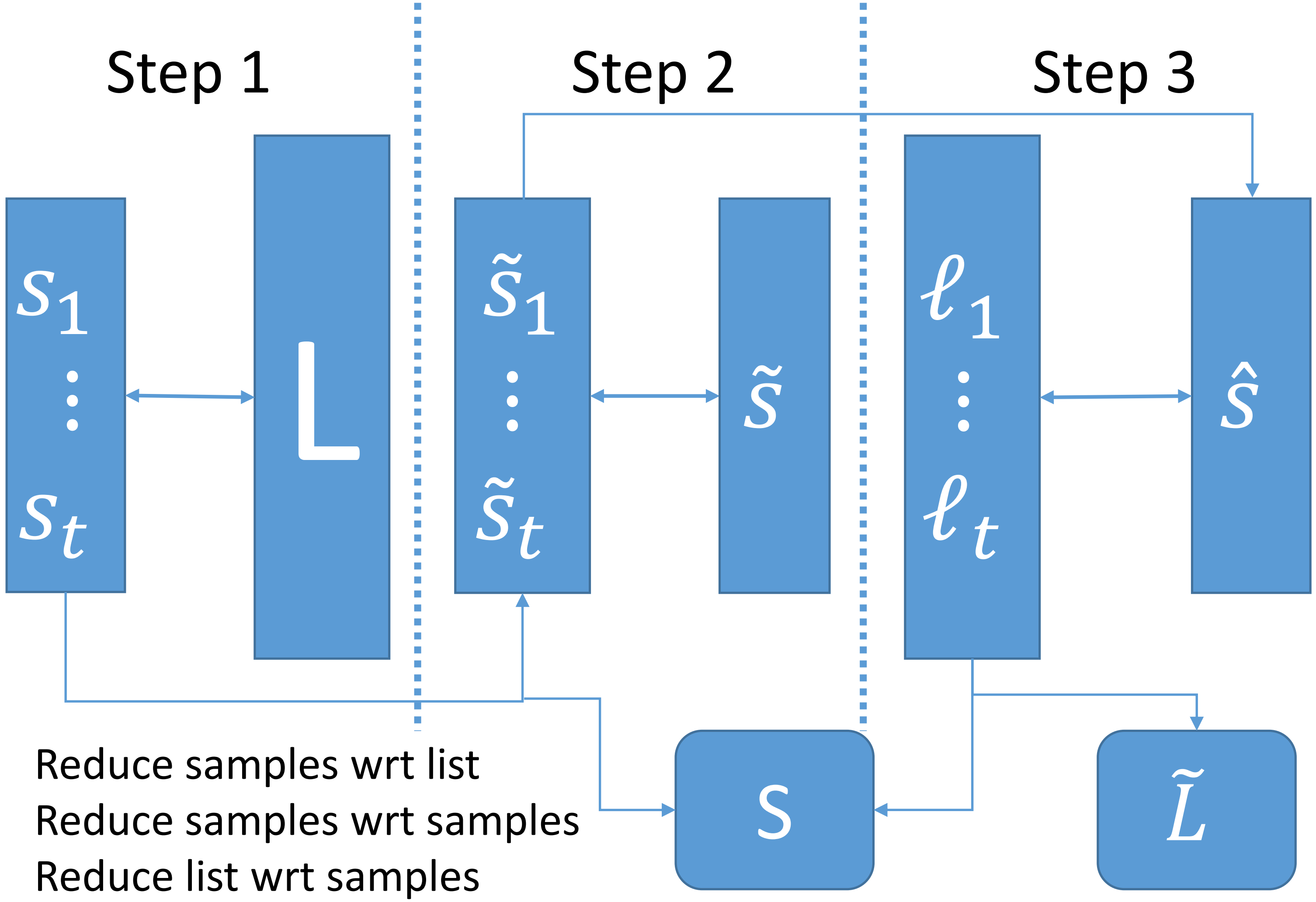
Parallel Gauss Sieve



Pros	Cons
Easy parallel algorithm	$\bigcup_i L_i$ are **not** necessarily pair-wise Gauss reduced
Total list size $(\bigcup_i L_i)$ is distributed among nodes	One node might sample a lot of new vectors: "traffic jams" + idle nodes
	Suggested solution: skip jams → more vectors in $(\bigcup_i L_i)$ are not pair-wise Gauss reduced → increased list size → increased running time

B. Milde and M. Schneider. A parallel implementation of Gauss Sieve for the shortest vector problem in lattices. In Parallel Computing Technologies, 2011

Parallel Gauss Sieve – another approach



- Reduce samples wrt list
- Reduce samples wrt samples
- Reduce list wrt samples
- Use S as new vectors and $\tilde{L}$ as the new list

✓ After step 3 all vectors in $\tilde{L}$ are pairwise Gauss reduced

✓ Avoids the traffic jam problem

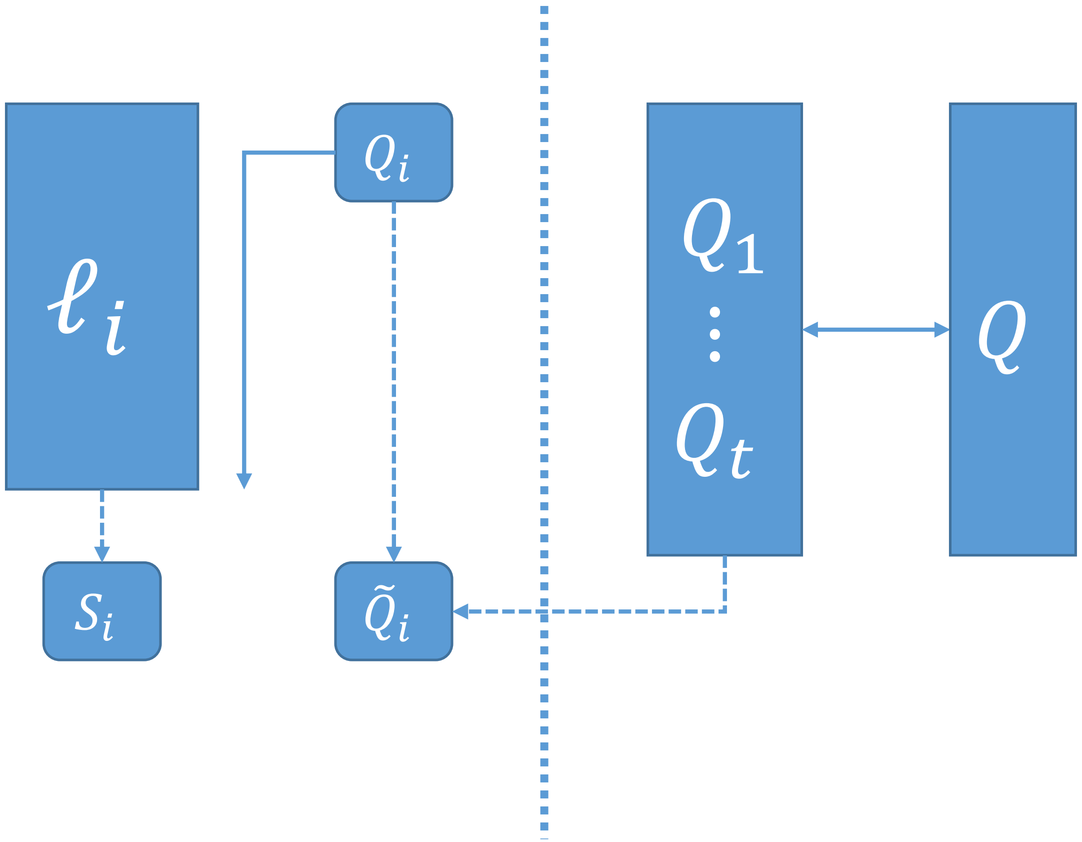
✓ Every node requires the complete list L and all samples S

✓ Conservative estimated max. list size for (non-ideal) dim. 128 is $2^{28} \rightarrow 64$ GB

✓ Used to solve ideal lattice challenge of dim. 128 in
  $\approx$ 15 days on 1344 CPUs
  $\approx$ 55 CPU years

T. Ishiguro, S. Kiyomoto, Y. Miyake, and T. Takagi. Parallel Gauss sieve algorithm: Solving the SVP challenge over a 128-dimensional ideal lattice. In PKC, 2014

Parallel Gauss Sieve – combining both approaches



- Collectively obtain new batch Q_i
- Reduce vectors from Q_i wrt ℓ_i and vice-versa
- Reduced vectors from ℓ_i go to s_i
- Reduced vectors from Q_i go to $\tilde{Q}_i$
- Reduce Q wrt to Q

- Locally ℓ_i is replaced by $\ell_i \backslash S_i$
- Compute j s. t. $|\ell_j|$ is minimal and update ℓ_j as $\ell_j \cup \bigcap_i Q_i$

- This avoids traffic jams
- Total list size $(\bigcup_i L_i)$ is distributed among nodes
- All vectors are pairwise Gauss reduced

- The same vector $v \in Q$ might be reduced by different ℓ_i at different nodes $\rightarrow$ collisions
- Propagate the vector with minimal norm

Ideal lattice

✓ Ideal lattice: additional structure → also ideals in a ring R

✓Most crypto settings restrict to

$$R = \mathbb{Z}[X]/(\Phi_m(X)),$$

where $m = 2n, n = 2^{\ell}, \ell > 0$ s.t. $\Phi_m(X) = X^n + 1$

- If $a(X)$ belongs to an ideal then $X^i a$ for $i \in \mathbb{Z}$ also belongs to the ideal
- Negative exponents: $X^{-1} = -X^{n-1}$

Notation: An element $a \in R$ is of the form

$$a(X) = \sum_{i=0}^{n-1} a_i X^i$$

and given by the coefficient vector

$$\boldsymbol{a} = (a_0, a_1, \dots, a_{n-1})$$

Ideal lattice

Previous work: store one vector, represent n vectors.

Observation 1: Checking if all n^2 pairs of rotations of a vector $\boldsymbol{a}$ with a vector $\boldsymbol{b}$ are Gauss reduced can be done with only n comparisons and n scalar products.

Lemma 1.
Let $a, b \in R = R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of 2 and $i, j \in \mathbb{Z}$. Then we have:

$$X^i \cdot \left(X^j \cdot \boldsymbol{a}\right) = X^{i+j} \cdot \boldsymbol{a}\,, \quad X^i \cdot (\boldsymbol{a} \cdot \boldsymbol{b}) = X^i \cdot \boldsymbol{a} + X^i \cdot \mathbf{b}\,, \quad X^n \cdot \boldsymbol{a} = -\boldsymbol{a},$$

$$\left\langle X^i \cdot \boldsymbol{a}, X^i \cdot \boldsymbol{b} \right\rangle = \langle \boldsymbol{a}, \boldsymbol{b} \rangle, \quad \left\langle X^i \cdot \boldsymbol{a}, X^j \cdot \boldsymbol{b} \right\rangle = \left\langle \boldsymbol{a}, -X^{n-i+j} \cdot \boldsymbol{b} \right\rangle.$$

Lemma 2.
Let $a, b \in R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of 2 and $i, j \in \mathbb{Z}$.

If $2\left|\left\langle \boldsymbol{a}, X^\ell \cdot \boldsymbol{b} \right\rangle\right| \leq \min\{\langle \boldsymbol{a}, \boldsymbol{a} \rangle, \langle \boldsymbol{b}, \boldsymbol{b} \rangle\}$ for all $0 \leq \ell < n$, then $X^i \cdot \boldsymbol{a}$ and $X^j \cdot \boldsymbol{b}$ are Gauss reduced for all $i, j \in \mathbb{Z}$.

Ideal lattice

Observation 1. Checking if all n^2 pairs of rotations of a vector $\boldsymbol{a}$ with a vector $\boldsymbol{b}$ are Gauss reduced can be done with only n comparisons and n scalar products.

Observation 2. The n scalar products can be computed using a single ring product.

Define the reflex polynomial $b^{(R)}(X)$ as

$$b^{(R)}(X) = X^{n-1} \cdot b(X^{-1}) \text{ such that } \boldsymbol{b}^{(R)} = (b_{n-1}, b_{n-2}, \ldots, b_0)$$

Lemma 3. Let

$$c(X) = a(X) \cdot \left(-X \cdot b^{(R)}(X)\right) \bmod (X^n + 1)$$

And let $c = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{Z}^n$ be its coefficient vector. Then $c_i = \left\langle a, X^i \cdot b \right\rangle$ for $0 \leq i < n$.

Ideal lattice

Observation 1. Checking if all n^2 pairs of rotations of a vector $\boldsymbol{a}$ with a vector $\boldsymbol{b}$ are Gauss reduced can be done with only n comparisons and n scalar products.

Observation 2. The n scalar products can be computed using a single ring product.

Observation 3. Since the ring product is a negacyclic convolution we can use a (symbolic) FFT

Nussbaumer's symbolic FFT

Decompose $\mathbb{Z}[X]/(X^n + 1)$ into two extensions. Let $n = 2^k = s \cdot r$ such that $s|r$. Then

$$R \cong S = T[X]/(X^s - Z), \text{ where } T = \mathbb{Z}[Z]/(Z^r + 1)$$

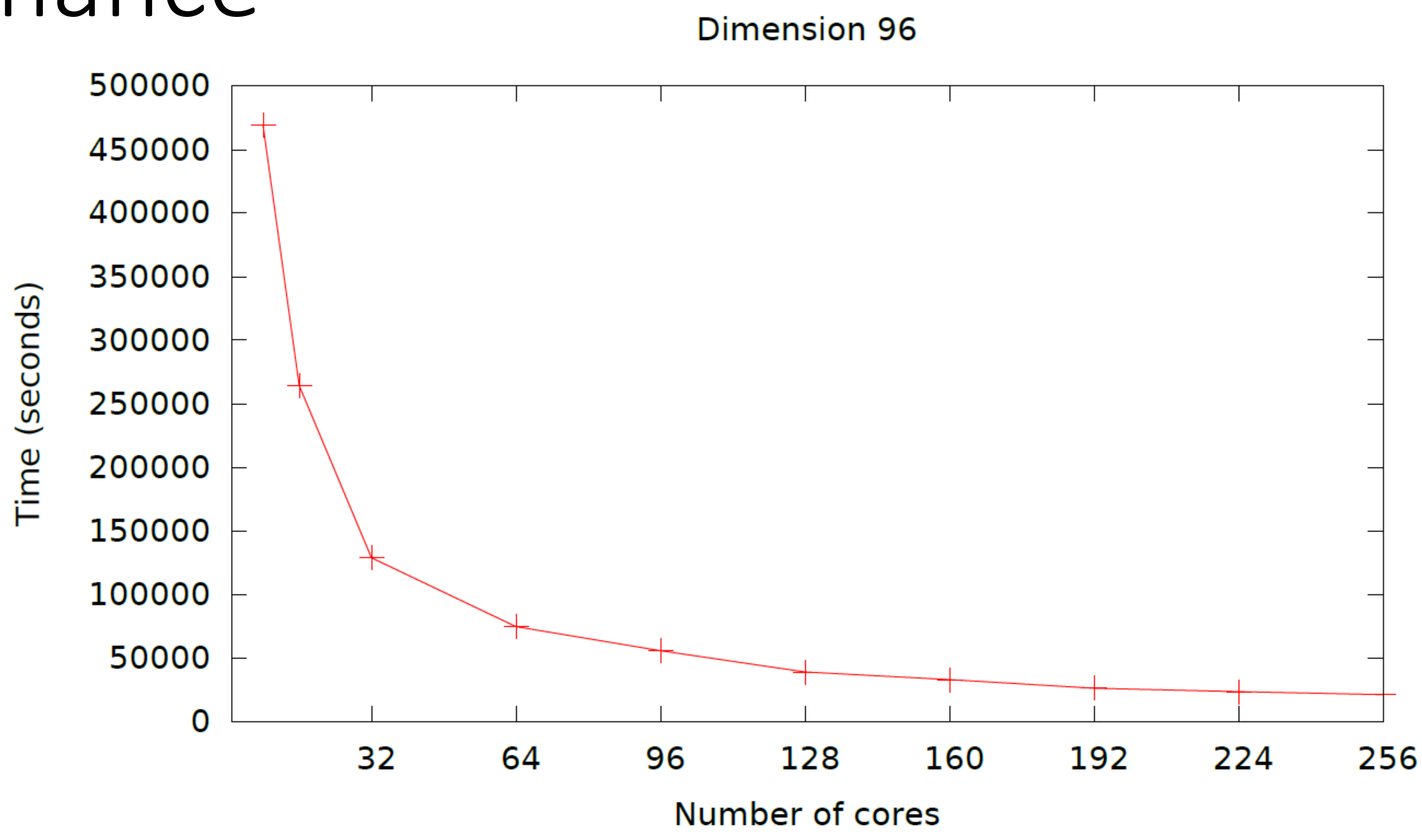
Note: $Z^{r/s}$ is an s^{th} root of -1 in T and $X^s = Z$ in S

Allows to compute the DFT symbolically in T

Use $\mathcal{O}(n \ln n)$ instead of $\mathcal{O}(n^2)$ arithmetic operations

H. J. Nussbaumer. Fast polynomial transform algorithms for digital convolution. Acoustics, Speech and Signal Processing, IEEE Transactions on, 1980

Performance

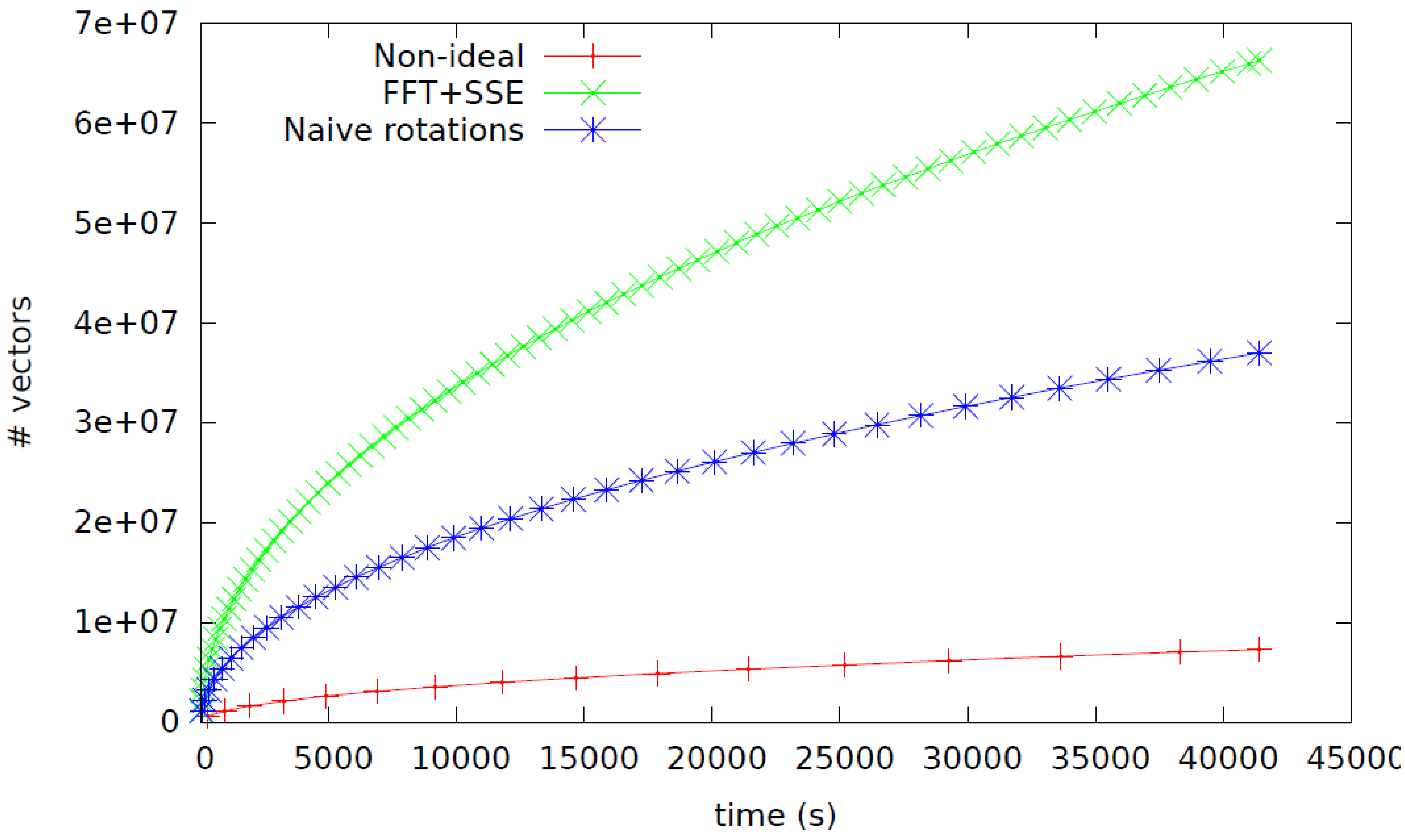


Speedup	
8 CPU versus 32 CPU	3.6
8 CPU versus 256 CPU	22.1

Lattices obtained from the SVP challenge, preprocess with BKZ with blocksize 30.

Experiments run on the BlueCrystal Phase 2 cluster of the Advanced Computing Research Centre at the University of Bristol

Performance



- Ishiguro et al. found a short vector in a dim. 128 ideal lattice in 14.88 days on 1334 CPUs ≈ 55 CPU years
- Our algorithm using FFT on the same lattice challenge on the same hardware (Bristol cluster) on 8.69 days on 1024 CPUs ≈ 25 CPU years
- More than twice as efficient
- Running challenge again with better load balancing, expect better results soon

Conclusions, Remarks & Future Work

- Better algorithms for the Gauss Sieve approach
- Symbolic FFT approach for Gauss Sieve approach in ideal lattices
- However, BKZ and variants of enumeration techniques appear to give best results in currently tractable dimensions (up to dim. 140)
- Unlike BKZ, enumeration etc. with sieving we can take advantage of the ideal lattice structure
- What about larger dimensions? Sieving algorithms seem to have asymptotically better run-time. Pinpointing this cross-over point is an important question for the security assessment of lattice-based crypto systems

- For more information see our paper: http://eprint.iacr.org/2014/880
- Source code will be made available in the upcoming weeks

Possible future work

Recent paper: locality-sensitive hash-sieve, try similar approach to divide list size

T. Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. Cryptology ePrint Archive, Report 2014/744