The BKZ algorithm

Joop van de Pol

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB
United Kingdom.

May 9th, 2014

University of BRISTOL

Outline

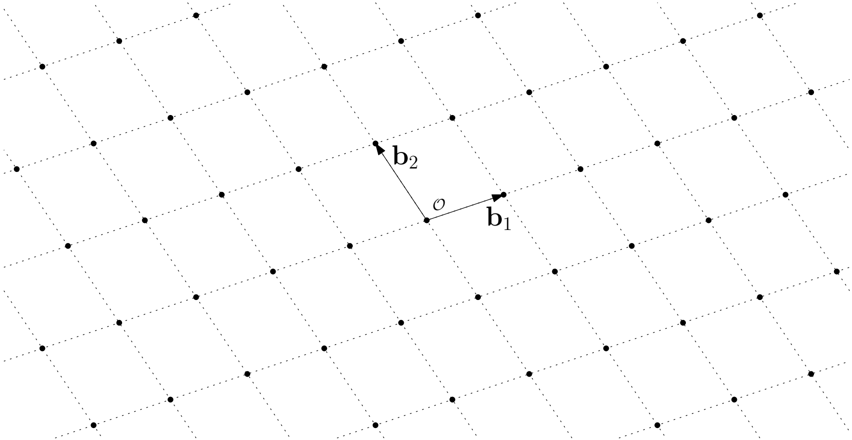
Lattices

Basis Reduction

- LLL
- BKZ
- Enumeration
- BKZ 2.0

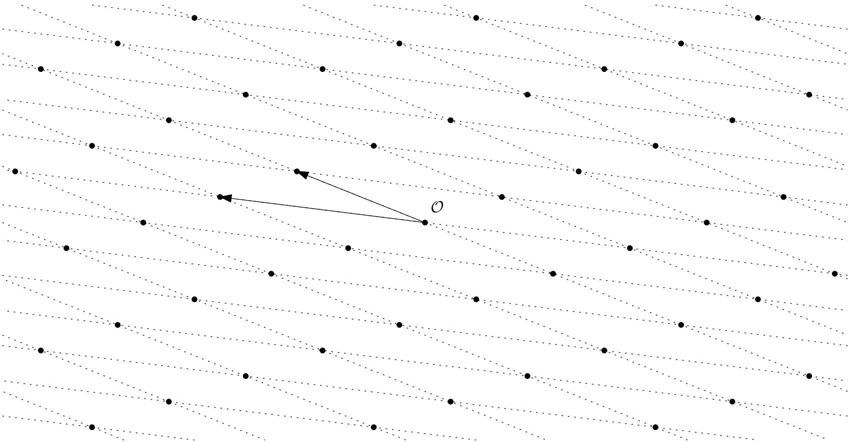
Open questions

Lattices



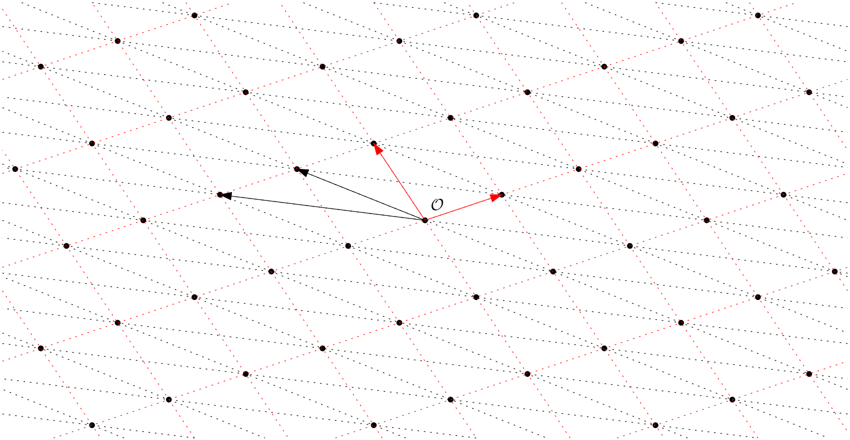
Lattices are represented by a basis.

Lattices



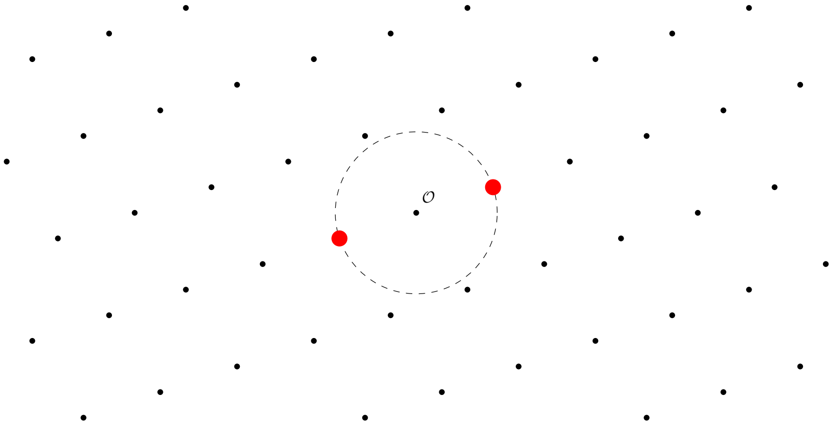
Lattices are represented by a basis. This basis is not unique.

Lattices



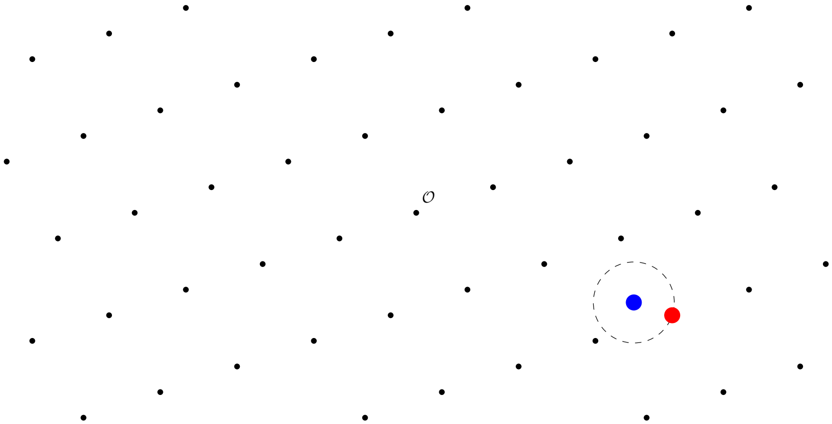
Lattices are represented by a basis. This basis is not unique.
Many bases span the same lattice. Some are ‘better’ than others.

Lattices



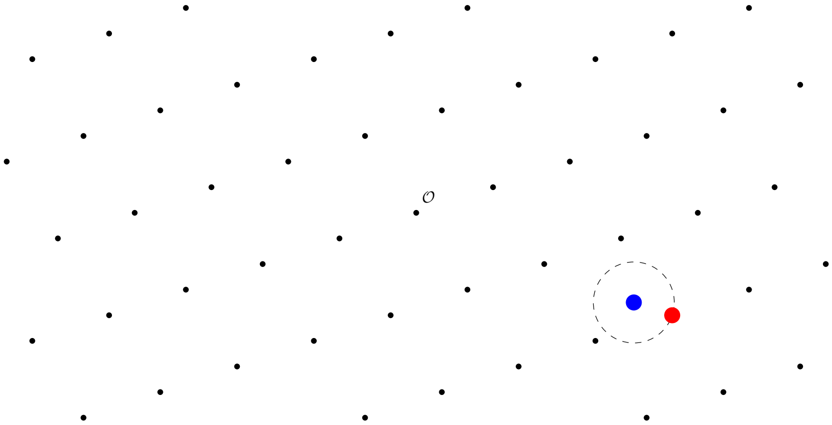
Lattice problems are about finding short and close vectors.

Lattices



Lattice problems are about finding short and close vectors.

Lattices



Lattice problems are about finding short and close vectors.
In practice it suffices to find short and orthogonal basis vectors.

Gram-Schmidt

Iterative process to orthonormalize a set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$:

$$\mathbf{b}_1^* := \mathbf{b}_1$$

$$\mathbf{b}_i^* := \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^*, \quad \text{where } \mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2} \text{ for all } 1 \leq j < i \leq d.$$

Result: vectors $\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*$ that are pairwise orthogonal.
They span the same *space* as $\mathbf{b}_1, \ldots, \mathbf{b}_d$.

Gram-Schmidt

Iterative process to orthonormalize a set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$:

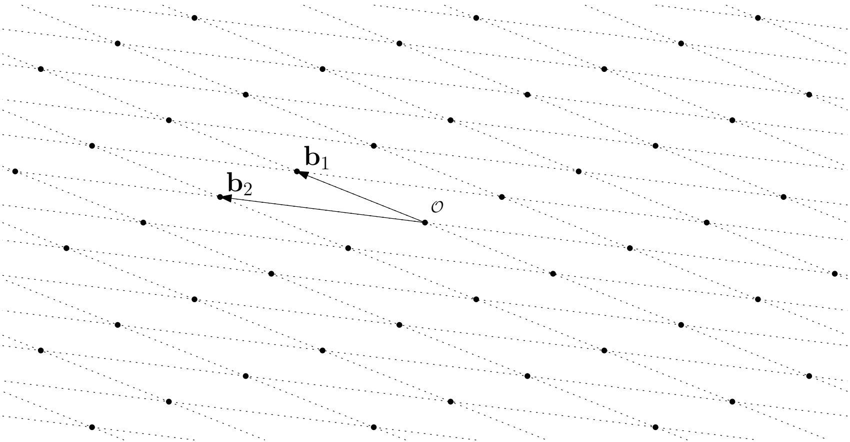
$$\mathbf{b}_1^* := \mathbf{b}_1$$

$$\mathbf{b}_i^* := \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^*, \quad \text{where } \mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2} \text{ for all } 1 \leq j < i \leq d.$$

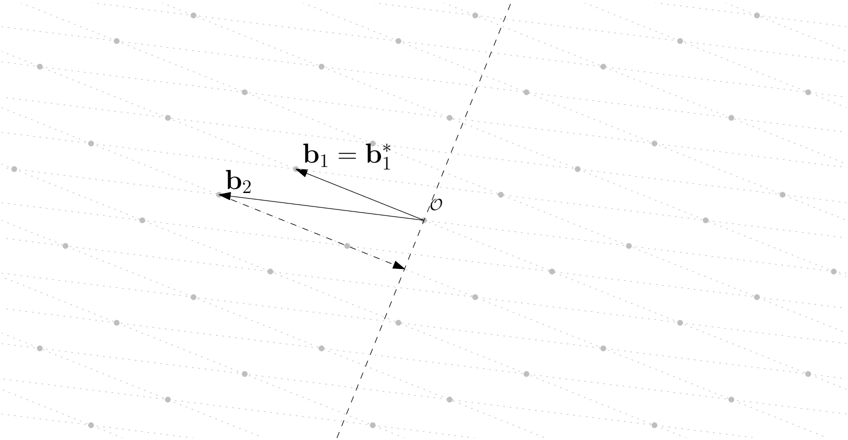
Result: vectors $\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*$ that are pairwise orthogonal.
They span the same *space* as $\mathbf{b}_1, \ldots, \mathbf{b}_d$.

In lattices: only integral combinations are allowed, $\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*$ will not span the same lattice!

Gram-Schmidt

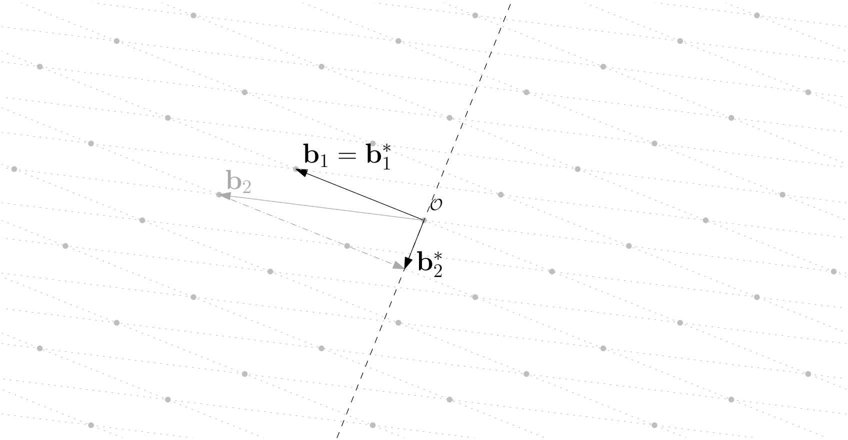


Gram-Schmidt



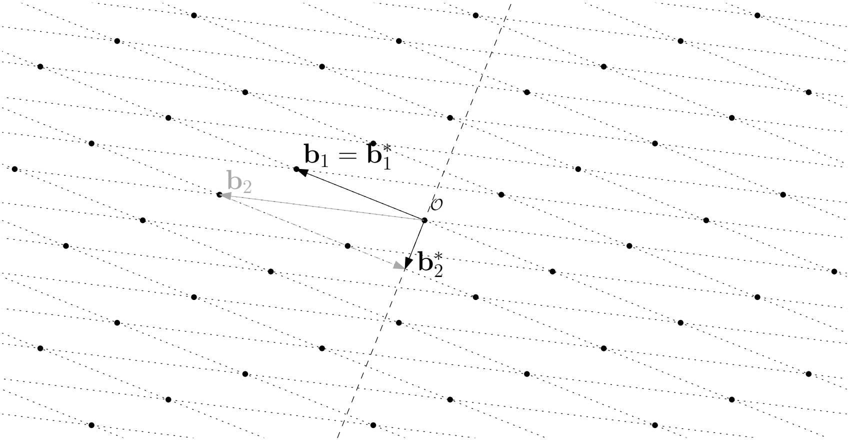
Forget that we are in a lattice.

Gram-Schmidt



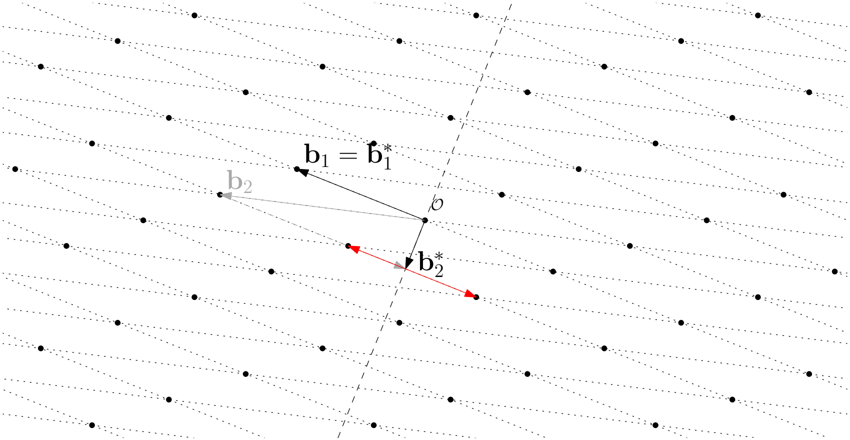
Projecting $\mathbf{b}_2$ gives $\mathbf{b}_2^*$.

Gram-Schmidt



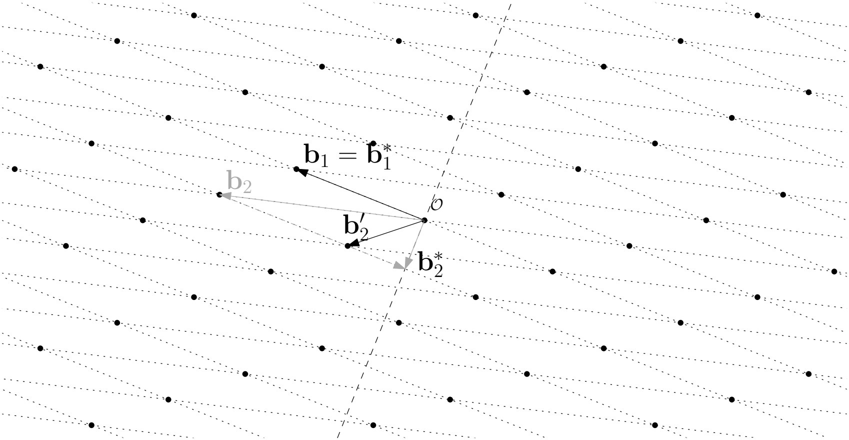
$\mathbf{b}_2^*$ is not a lattice vector.

Gram-Schmidt



But there is a lattice vector within $\frac{1}{2}\|\mathbf{b}_1^*\|$ from $\mathbf{b}_2^*$:
$\mathbf{b}_2' := \mathbf{b}_2 - \lceil \mu_{2,1} \rfloor \cdot \mathbf{b}_1$.

Gram-Schmidt



It is always possible to choose a basis close to the Gram Schmidt vectors. This basis is called size-reduced.

LLL (1982)

First polynomial-time basis reduction algorithm.
Ideas:

- Always take the basis ‘closest’ to Gram-Schmidt.

LLL (1982)

First polynomial-time basis reduction algorithm.
Ideas:

- Always take the basis ‘closest’ to Gram-Schmidt.
- Improve Gram-Schmidt vectors by changing their order.

LLL (1982)

First polynomial-time basis reduction algorithm.
Ideas:

- Always take the basis 'closest' to Gram-Schmidt.
- Improve Gram-Schmidt vectors by changing their order.
- Being greedy when ordering basis vectors is bad for the complexity.

$$\mathbf{b}_1, \ldots, \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_d$$
$$\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_d^*$$

LLL (1982)

First polynomial-time basis reduction algorithm.
Ideas:

- Always take the basis 'closest' to Gram-Schmidt.
- Improve Gram-Schmidt vectors by changing their order.
- Being greedy when ordering basis vectors is bad for the complexity.

$$\mathbf{b}_1, \ldots, \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_d$$
$$\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_d^*$$

What happens when $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ are swapped?
Only $\mathbf{b}_i^*$ and $\mathbf{b}_{i+1}^*$ change. New $\mathbf{b}_i^*$ becomes $\mathbf{b}_{i+1}^* + \mu_{i+1,i}\mathbf{b}_i^*$.

LLL (1982)

First polynomial-time basis reduction algorithm.
Ideas:

- Always take the basis 'closest' to Gram-Schmidt.
- Improve Gram-Schmidt vectors by changing their order.
- Being greedy when ordering basis vectors is bad for the complexity.

$$\mathbf{b}_1, \ldots, \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_d$$
$$\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_d^*$$

What happens when $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ are swapped?
Only $\mathbf{b}_i^*$ and $\mathbf{b}_{i+1}^*$ change. New $\mathbf{b}_i^*$ becomes $\mathbf{b}_{i+1}^* + \mu_{i+1,i}\mathbf{b}_i^*$.
Swap when $\|\mathbf{b}_{i+1}^* + \mu_{i+1,i}\mathbf{b}_i^*\|^2 < \delta\|\mathbf{b}_i^*\|^2$, for $\delta \in (1/4, 1)$.

BKZ (1987, 1994)

Trade-off between basis quality and time.

$$\mathbf{b}_1, \ldots, \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}, \ldots, \mathbf{b}_d$$
$$\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_{i+\beta-1}^*, \ldots, \mathbf{b}_d^*$$

BKZ (1987, 1994)

Trade-off between basis quality and time.

$$\mathbf{b}_1, \ldots, \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}, \ldots, \mathbf{b}_d$$
$$\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_{i+\beta-1}^*, \ldots, \mathbf{b}_d^*$$

Compute $\mathbf{b}_{\mathrm{new}}$, a combination of vectors $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}$ such that it becomes the shortest possible i'th Gram-Schmidt vector.

BKZ (1987, 1994)

Trade-off between basis quality and time.

$$\mathbf{b}_1, \ldots, \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}, \ldots, \mathbf{b}_d$$
$$\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_{i+\beta-1}^*, \ldots, \mathbf{b}_d^*$$

Compute $\mathbf{b}_{\text{new}}$, a combination of vectors $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}$ such that it becomes the shortest possible i'th Gram-Schmidt vector.
If $\|\mathbf{b}_{\text{new}}\|^2 < \delta \|\mathbf{b}_i^*\|^2$, insert $\mathbf{b}_{\text{new}}$ into the basis:

$$\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_{\text{new}}, \mathbf{b}_i, \ldots, \mathbf{b}_d$$

BKZ (1987, 1994)

Trade-off between basis quality and time.

$$\mathbf{b}_1, \ldots, \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}, \ldots, \mathbf{b}_d$$
$$\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_{i+\beta-1}^*, \ldots, \mathbf{b}_d^*$$

Compute $\mathbf{b}_{\mathrm{new}}$, a combination of vectors $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}$ such that it becomes the shortest possible i'th Gram-Schmidt vector. If $\|\mathbf{b}_{\mathrm{new}}\|^2 < \delta \|\mathbf{b}_i^*\|^2$, insert $\mathbf{b}_{\mathrm{new}}$ into the basis:

$$\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_{\mathrm{new}}, \mathbf{b}_i, \ldots, \mathbf{b}_d$$

Now LLL is used to remove the linear dependency created by the extra vector. BKZ moves cyclically through the basis indices i.

BKZ (1987, 1994)

Trade-off between basis quality and time.

$$\mathbf{b}_1, \ldots, \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}, \ldots, \mathbf{b}_d$$
$$\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_{i+\beta-1}^*, \ldots, \mathbf{b}_d^*$$

Compute $\mathbf{b}_{\text{new}}$, a combination of vectors $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}$ such that it becomes the shortest possible i'th Gram-Schmidt vector. If $\|\mathbf{b}_{\text{new}}\|^2 < \delta\|\mathbf{b}_i^*\|^2$, insert $\mathbf{b}_{\text{new}}$ into the basis:

$$\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_{\text{new}}, \mathbf{b}_i, \ldots, \mathbf{b}_d$$

Now LLL is used to remove the linear dependency created by the extra vector. BKZ moves cyclically through the basis indices i.

Note: we do not have a good bound on the time complexity of BKZ.

BKZ (1987, 1994)

"Compute $\mathbf{b}_{\text{new}}$, a combination of vectors $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}$ such that it becomes the shortest possible *i*'th Gram-Schmidt vector."

BKZ (1987, 1994)

"Compute $\mathbf{b}_{\text{new}}$, a combination of vectors $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}$ such that it becomes the shortest possible i'th Gram-Schmidt vector."

This is equivalent to solving SVP (!) in a (projected) lattice of dimension β. BKZ uses an SVP-oracle for lower dimensions to find this vector $\mathbf{b}_{\text{new}}$.

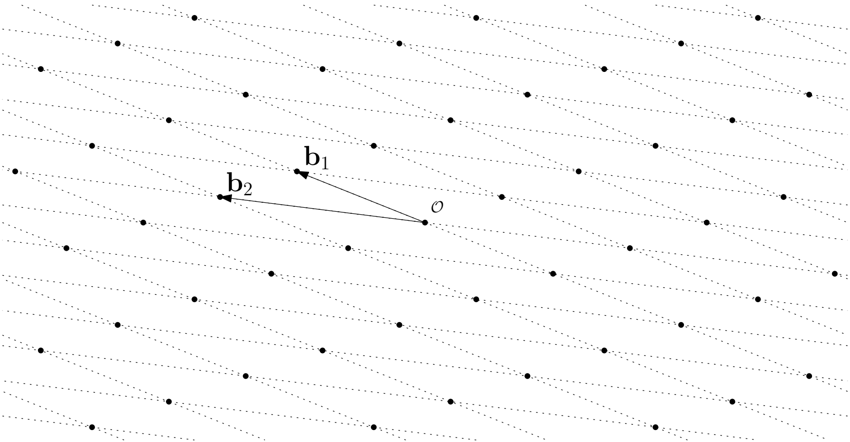
BKZ (1987, 1994)

"Compute $\mathbf{b}_{\text{new}}$, a combination of vectors $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}$ such that it becomes the shortest possible *i*'th Gram-Schmidt vector."

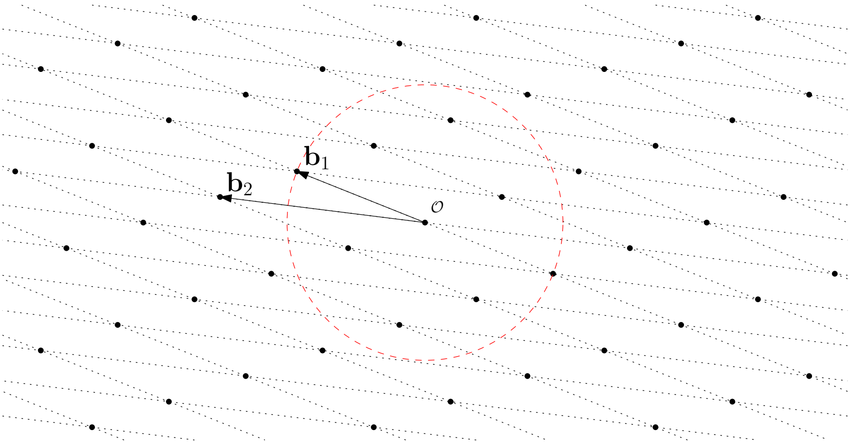
This is equivalent to solving SVP (!) in a (projected) lattice of dimension β. BKZ uses an SVP-oracle for lower dimensions to find this vector $\mathbf{b}_{\text{new}}$.

In practice enumeration is used: enumerate all lattice points within a certain radius around the origin.

Enumeration

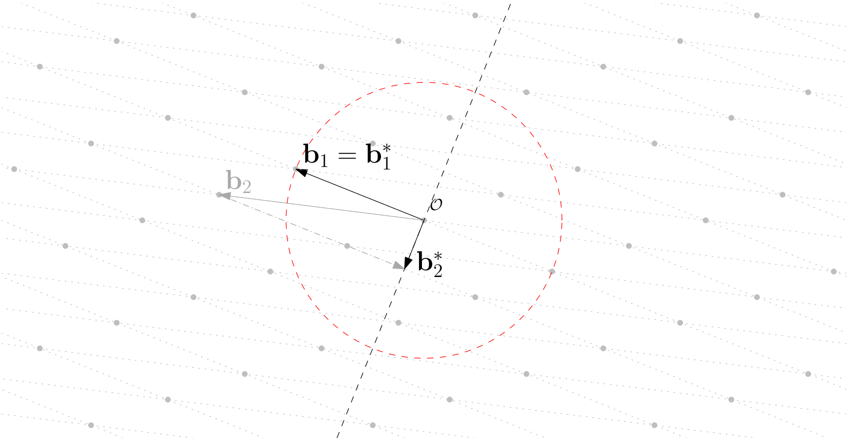


Enumeration



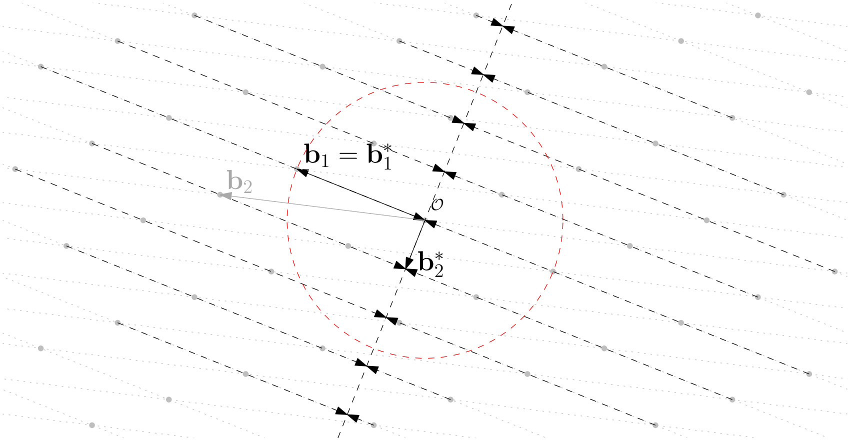
1) Choose a bound.

Enumeration



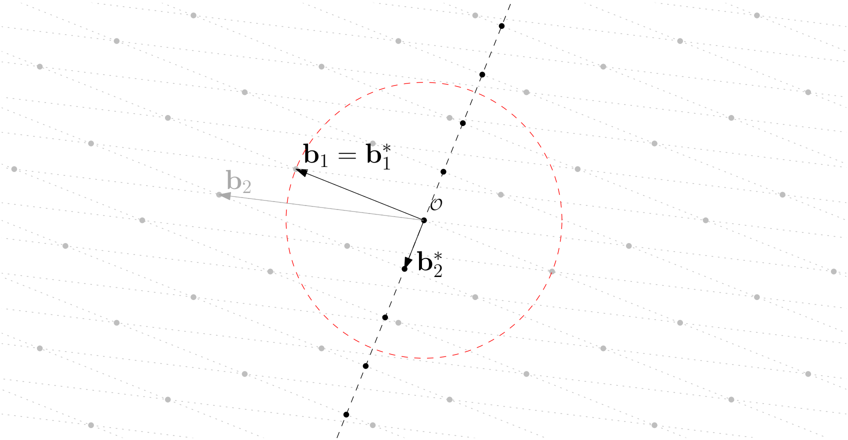
2) Do the Gram-Schmidt.

Enumeration



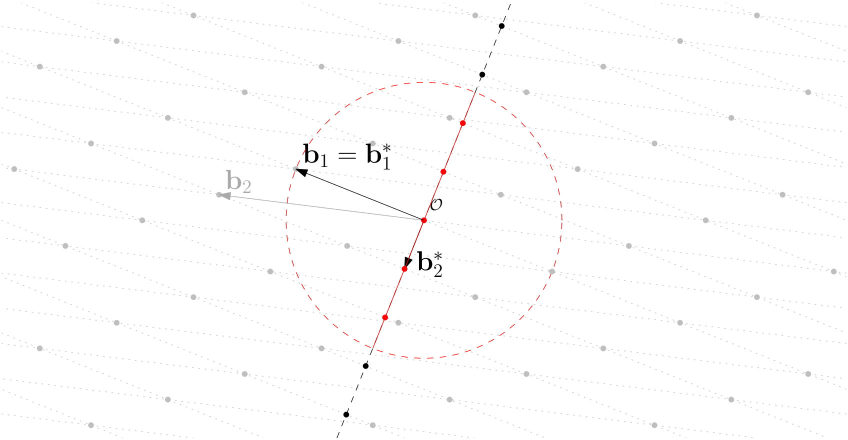
3) ‘Project’ whole lattice.

Enumeration



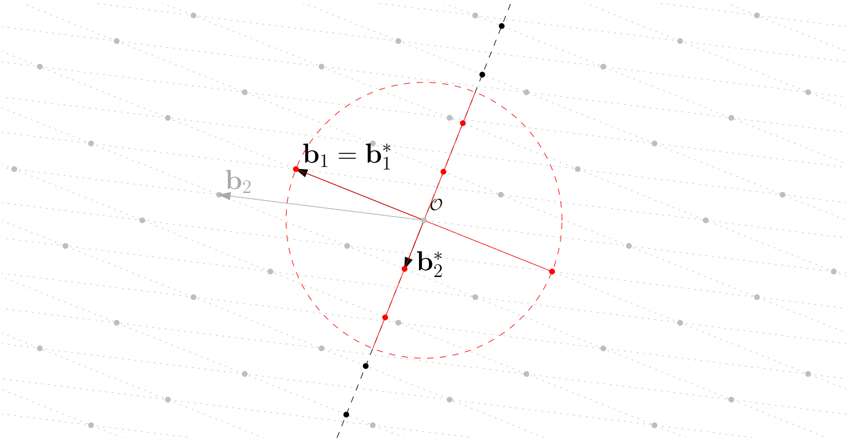
Lattice vector within bound ⇒ its projection within bound.

Enumeration



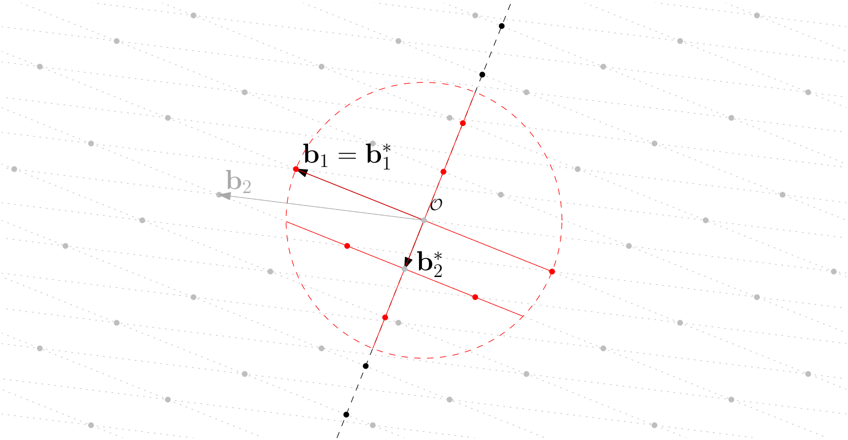
4) Enumerate all vectors in projected lattice within bound.

Enumeration



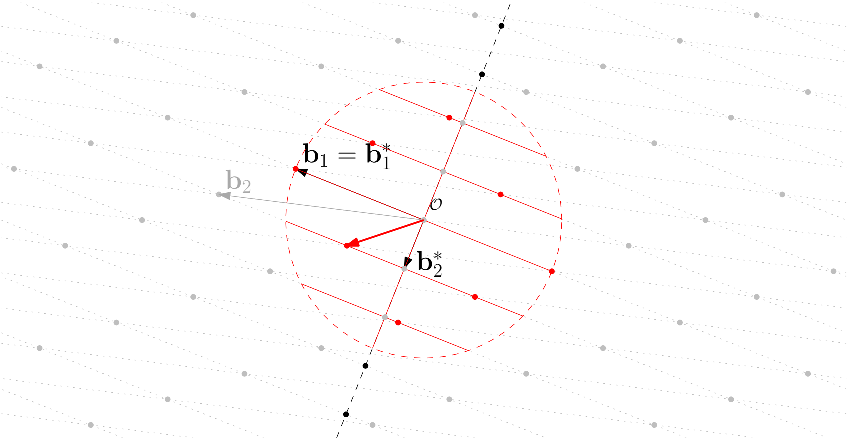
5) For each vector in projected lattice, enumerate all lattice vectors.

Enumeration



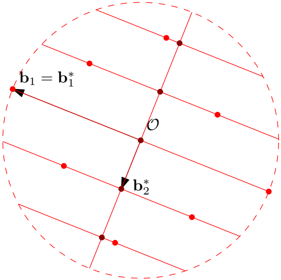
5) For each vector in projected lattice, enumerate all lattice vectors.

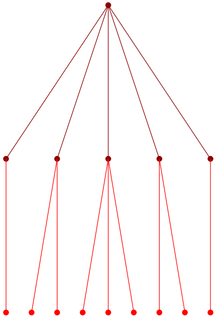
Enumeration



6) Pick the shortest vector.

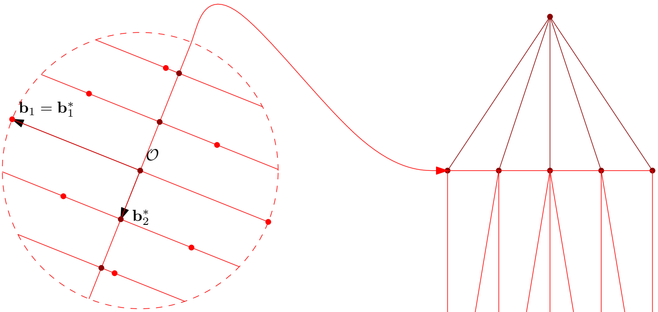
Enumeration as a tree





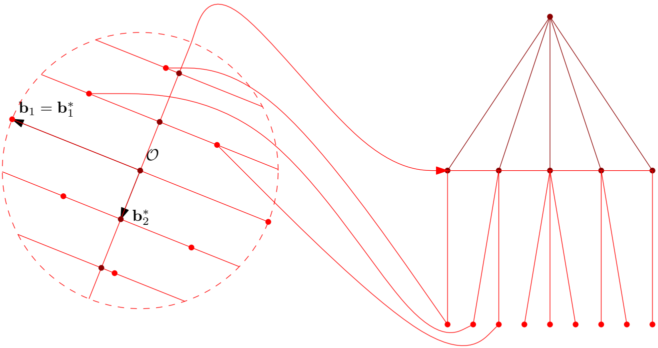
Enumeration is like a tree search.

Enumeration as a tree



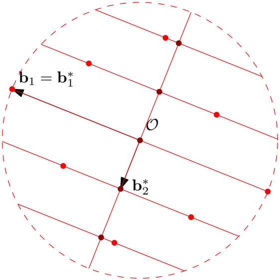
Enumeration is like a tree search. Each level corresponds to a projected lattice.

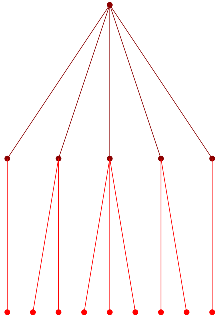
Enumeration as a tree



Enumeration is like a tree search. Each level corresponds to a projected lattice. The leaves correspond to lattice vectors.

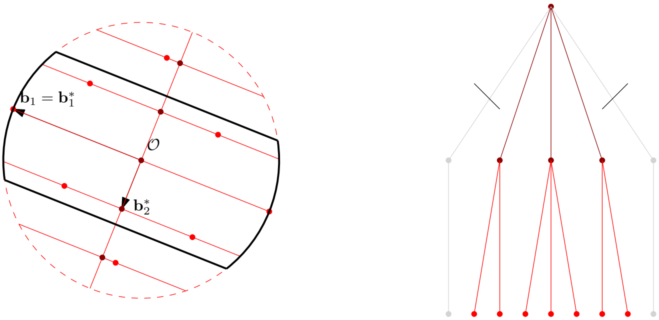
Extreme pruning





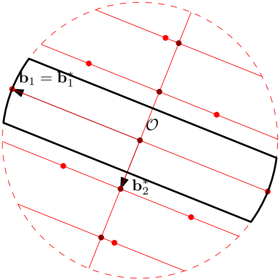
Branches near the edge yield fewer leaves.

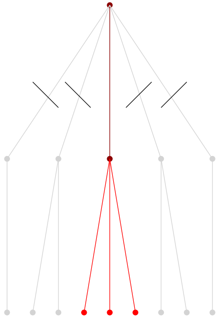
Extreme pruning



Branches near the edge yield fewer leaves. Pruning decreases the size of the tree.

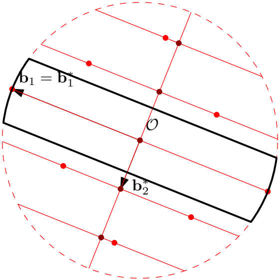
Extreme pruning

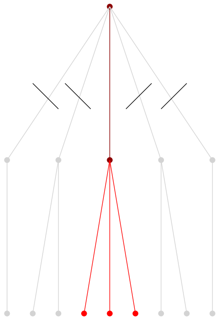




Branches near the edge yield fewer leaves. Pruning decreases the size of the tree. It might also remove the solutions.

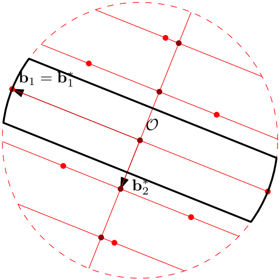
Extreme pruning

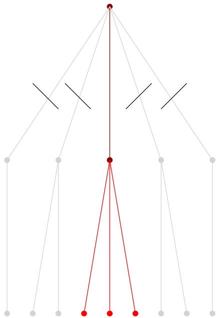




Extreme pruning: probability p of finding the solution, but more than p^{-1} times faster.

Extreme pruning





Extreme pruning: probability p of finding the solution, but more than p^{-1} times faster. This gives a speed-up of $\approx 2^{d/2}$.

BKZ 2.0

Extreme pruning requires a good bound on the length of the shortest vector. For small β the Gaussian Heuristic does not hold.

BKZ 2.0

Extreme pruning requires a good bound on the length of the shortest vector. For small β the Gaussian Heuristic does not hold.

For $\beta > 40$, the projected lattices behave like random lattices. The Gaussian Heuristic gives us a good bound.

BKZ 2.0

Extreme pruning requires a good bound on the length of the shortest vector. For small β the Gaussian Heuristic does not hold.

For $\beta > 40$, the projected lattices behave like random lattices. The Gaussian Heuristic gives us a good bound.

Chen and Nguyen proposed BKZ 2.0 with the following improvements over the original:

- Better enumeration bound
- Extreme pruning
- Aborting BKZ after a fixed number of rounds
- Better preprocessing of the blocks

Open questions

Regarding BKZ (2.0):

- Many heuristics. What can we prove?
- Destroys local structure for global improvement. Can this be done better?
- What about structured (ideal) lattices?
- Can we speed it up using a quantum computer?

In general:

- Are there better classical algorithms?
- What about quantum algorithms?

Questions?