The General Sieve Kernel and New Records in Lattice Reduction

Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova,
Eamonn W. Postlethwaite, Marc Stevens

(Lattice) Sieving: What and Why?

Sieves:

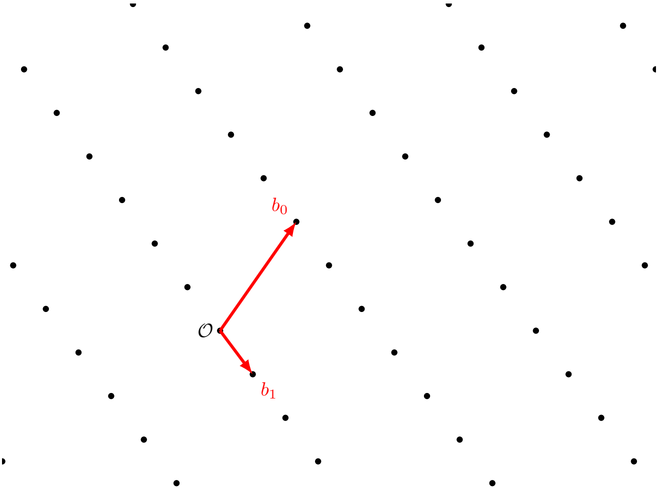
- take as input a description of a lattice (a “basis”),
- output a database of short vectors in that lattice.

Short vectors are critical in lattice reduction and in more general cryptanalysis.

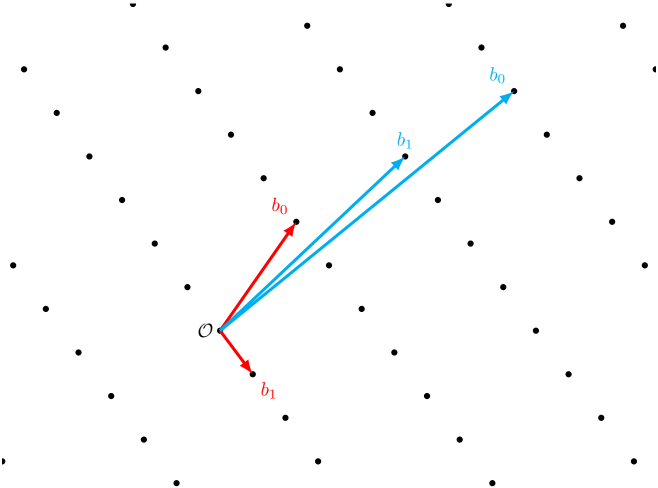
(Lattice) Sieving: What and Why?

A single exponential, in lattice dimension d, time and memory short vector finder.

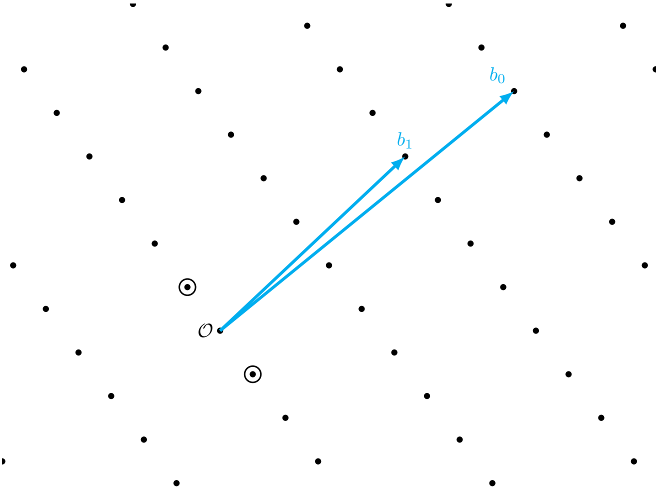
algorithm	time	memory
sieving	$\exp(\Theta(d))$	$\exp(\Theta(d))$
enumeration	$\exp(\Theta(d \log d))$	$\mathrm{poly}(d)$



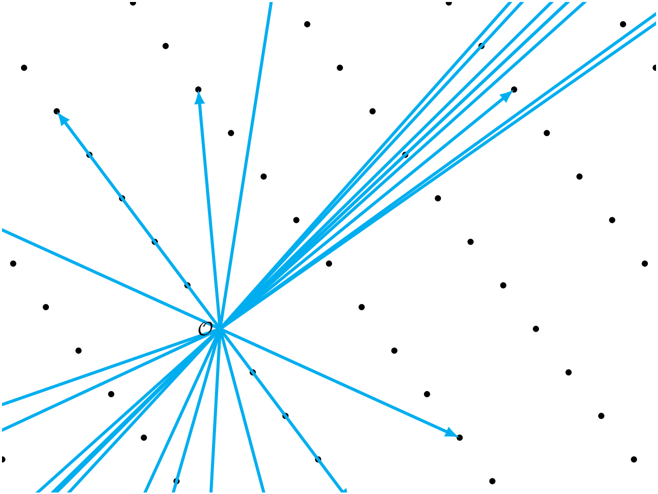
$\Lambda = \mathrm{Span}_{\mathbb{Z}}(b_0, \ldots, b_{d-1})$, $B = \{b_0, \ldots, b_{d-1}\} \subset \mathbb{R}^d$ basis



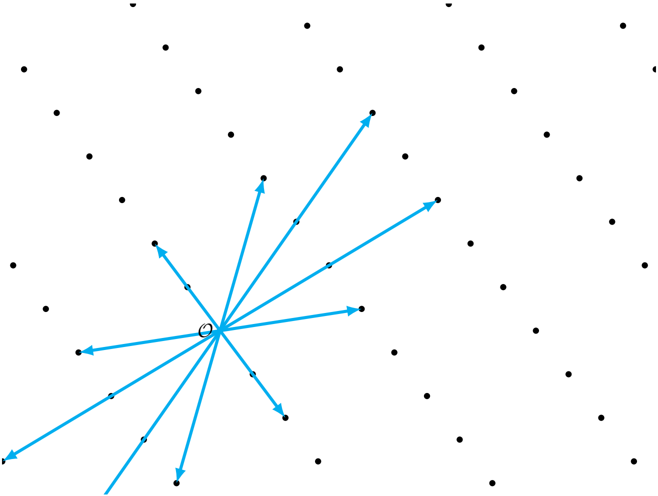
Good basis B, bad basis B



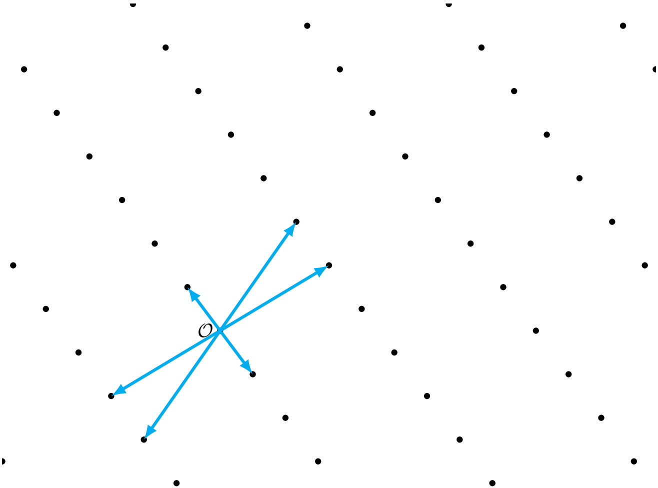
SVP: find $v \in \Lambda \setminus \{0\}$ such that $\|v\| \leq \|w\|$ for all $w \in \Lambda \setminus \{0\}$



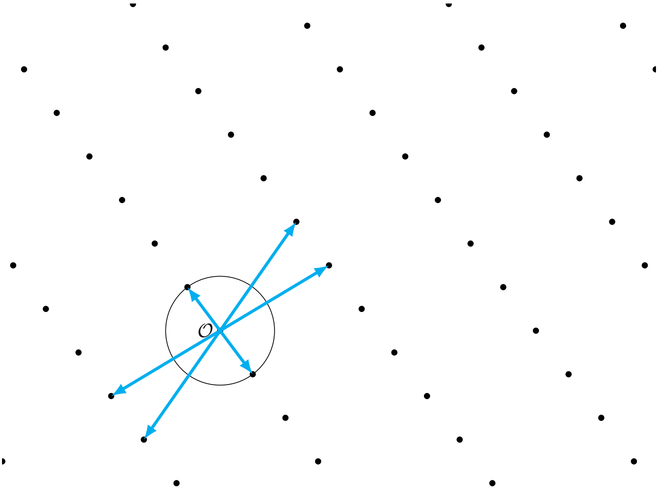
Sieve: $v, w \in L \subset \Lambda$, if $\|v \pm w\| \leq \|v\|$, $v \leftarrow v \pm w$



Sieve: $v, w \in L \subset \Lambda$, if $\|v \pm w\| \leq \|v\|$, $v \leftarrow v \pm w$



Sieve: $v, w \in L \subset \Lambda$, if $\|v \pm w\| \leq \|v\|$, $v \leftarrow v \pm w$



Sieve: $v, w \in L \subset \Lambda$, if $\|v \pm w\| \leq \|v\|$, $v \leftarrow v \pm w$

Intuition

A sieve outputs many short vectors, but only the shortest is used.

---

[1] L. Babai, On lovász' lattice reduction and the nearest lattice point problem, Combinatorica 6 (1986), no. 1, 1–13.

[2] Léo Ducas, Shortest vector from lattice sieving: A few dimensions for free, EUROCRYPT 2018, Part I (Jesper Buus Nielsen and Vincent Rijmen, eds.), LNCS, vol. 10820, Springer, Heidelberg, April / May 2018, pp. 125–145.

[3] Thijs Laarhoven and Artur Mariano, Progressive lattice sieving, Post-Quantum Cryptography – 9th International Conference, PQCrypto 2018 (Tanja Lange and Rainer Steinwandt, eds.), Springer, Heidelberg, 2018, pp. 292–311.

Intuition

A sieve outputs many short vectors, but only the shortest is used. Instead

- sieve in (projected) sublattices and lift[1] to the full lattice[2]
- sieve in sublattices to "seed" higher dimensional sieves[3]

[1] L. Babai, On lovász' lattice reduction and the nearest lattice point problem, Combinatorica 6 (1986), no. 1, 1–13.

[2] Léo Ducas, Shortest vector from lattice sieving: A few dimensions for free, EUROCRYPT 2018, Part I (Jesper Buus Nielsen and Vincent Rijmen, eds.), LNCS, vol. 10820, Springer, Heidelberg, April / May 2018, pp. 125–145.

[3] Thijs Laarhoven and Artur Mariano, Progressive lattice sieving, Post-Quantum Cryptography – 9th International Conference, PQCrypto 2018 (Tanja Lange and Rainer Steinwandt, eds.), Springer, Heidelberg, 2018, pp. 292–311.

Intuition II

Recycle information between related lattices

and

Go beyond sieving as black box oracle for shortest vectors

Implicitly,

sieve $\leftrightarrow$ stateful machine

Contributions

In our paper we give

- a framework for treating sieves as stateful machines,
- an open source, $\{$documented, optimised, tweakable$\}$ implementation, `https://github.com/fplll/g6k`,
- a variety of new strategies for lattice reduction tasks.

Contributions

In our paper we give

- a framework for treating sieves as stateful machines,
- an open source, {documented, optimised, tweakable} implementation, `https://github.com/fplll/g6k`,
- a variety of new strategies for lattice reduction tasks.

We are therefore able to

- show sieving outperforms enumeration by low dimensions,
- show that SVP can be (slightly) amortised within BKZ,
- break a number of lattice challenge records.

The General Sieve Kernel, or
G6K (pronounced ʒe.si.ka)

A Simple Sieve

We use a grammar to define our sieving operations.

A Simple Sieve

We use a grammar to define our sieving operations.

$\text{Reset}_{0,d}$ S I_0

Key: sieve, lift, c_i = insert

A Simple Sieve

We use a grammar to define our sieving operations.

$\mathrm{Reset}_{0,d}$ S I_0

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Reset!

Key: sieve, lift, c_i = insert

A Simple Sieve

We use a grammar to define our sieving operations.

$\text{Reset}_{0,d}$ S I_0

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Simple Sieve

We use a grammar to define our sieving operations.

Reset$_{0,d}$ S I$_0$

$$(c_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Insert!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\mathrm{Reset}_{0,1}\ (\mathrm{ER\ S})^{d-1}\ \mathrm{I}_0$

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\text{Reset}_{0,1}$ $(\text{ER S})^{d-1}$ I_0

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Reset!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\text{Reset}_{0,1}\ (\text{ER S})^{d-1}\ \text{I}_0$

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Extend Right!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\mathrm{Reset}_{0,1}\ (\mathrm{ER\ S})^{d-1}\ \mathrm{I}_0$

$$({\color{red}b_0}, {\color{red}b_1}, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\text{Reset}_{0,1}\ (\text{ER S})^{d-1}\ \text{I}_0$

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Extend Right!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\mathrm{Reset}_{0,1}\ (\mathrm{ER\ S})^{d-1}\ \mathrm{I}_0$

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\mathrm{Reset}_{0,1}$ $(\mathrm{ER\ S})^{d-1}$ I_0

$$({\color{red}b_0}, {\color{red}b_1}, {\color{red}b_2}, \ldots, {\color{red}b_{d-3}}, b_{d-2}, b_{d-1})$$

Extend Right!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\text{Reset}_{0,1}\ (\text{ER S})^{d-1}\ \text{I}_0$

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\text{Reset}_{0,1}$ $(\text{ER S})^{d-1}$ I_0

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Extend Right!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\text{Reset}_{0,1}$ (ER S)$^{d-1}$ I_0

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\text{Reset}_{0,1}$ $(\text{ER S})^{d-1}$ I_0

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Extend Right!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\text{Reset}_{0,1}\ (\text{ER S})^{d-1}\ \text{I}_0$

$$(b_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Progressive Sieve [LM18]

$\mathrm{Reset}_{0,1}\ (\mathrm{ER\ S})^{d-1}\ \mathrm{I}_0$

$$(c_0, b_1, b_2, \ldots, b_{d-3}, b_{d-2}, b_{d-1})$$

Insert!

Key: sieve, lift, c_i = insert

A "Dimensions for Free" Sieve [Duc18]

$\mathrm{Reset}_{f,f+1}$ $(\mathrm{E}\,\mathrm{R}\,\mathrm{S})^{d-f-1}$ I_0 I_1 $\ldots$

Key: sieve, lift, c_i = insert

A "Dimensions for Free" Sieve [Duc18]

$\text{Reset}_{f,f+1}$ $(\text{ER S})^{d-f-1}$ I_0 I_1 $\ldots$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, \ldots, b_{d-1})$$

Reset!

Key: sieve, lift, c_i = insert

A “Dimensions for Free” Sieve [Duc18]

$\text{Reset}_{f,f+1}\ (\text{ER S})^{d-f-1}\ \text{I}_0\ \text{I}_1\ \ldots$

$$(b_0, b_1, \ldots, b_{f-1}, {\color{red}b_f}, {\color{red}b_{f+1}}, \ldots, \ldots, b_{d-1})$$

Extend Right!

Key: sieve, lift, c_i = insert

A "Dimensions for Free" Sieve [Duc18]

$\mathrm{Reset}_{f,f+1}$ $(\mathrm{ER}\ \mathrm{S})^{d-f-1}$ I_0 I_1 $\ldots$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, \ldots, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A “Dimensions for Free” Sieve [Duc18]

$\text{Reset}_{f,f+1}$ $(\text{ER S})^{d-f-1}$ I_0 I_1 $\ldots$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, \ldots, b_{d-1})$$

Extend Right!

Key: sieve, lift, c_i = insert

A “Dimensions for Free” Sieve [Duc18]

$\text{Reset}_{f,f+1}\ (\text{ER S})^{d-f-1}\ \text{I}_0\ \text{I}_1\ \ldots$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, \ldots, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A “Dimensions for Free” Sieve [Duc18]

$\text{Reset}_{f,f+1}\ (\text{ER S})^{d-f-1}\ \text{I}_0\ \text{I}_1\ \ldots$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, \ldots, b_{d-1})$$

Key: sieve, lift, c_i = insert

A "Dimensions for Free" Sieve [Duc18]

$\text{Reset}_{f,f+1}\ (\text{ER S})^{d-f-1}\ \text{I}_0\ \text{I}_1\ \ldots$

$$(c_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, \ldots, b_{d-1})$$

Insert!

Key: sieve, lift, c_i = insert

A "Dimensions for Free" Sieve [Duc18]

$\mathrm{Reset}_{f,f+1}\ (\mathrm{ER\ S})^{d-f-1}\ \mathrm{I}_0\ \mathrm{I}_1\ \ldots$

$$(c_0, c_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, \ldots, b_{d-1})$$

Insert!

Key: sieve, lift, c_i = insert

A Pump

$$\text{Pump}_{f,r}\text{: Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

Key: sieve, lift, c_i = insert

A Pump

$$\text{Pump}_{f,r}\text{: } \text{Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

$$\left(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, \color{red}{b_{d-1}}\right)$$

Reset!

Key: sieve, lift, c_i = insert

A Pump

$$\mathrm{Pump}_{f,r}\colon\ \mathrm{Reset}_{r-1,r}\ \overbrace{(\mathrm{EL\ S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\mathrm{I\ S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, \color{red}{b_{d-2}}, \color{red}{b_{d-1}})$$

Extend Left!

Key: sieve, lift, c_i = insert

A Pump

$$\mathrm{Pump}_{f,r}\text{: } \mathrm{Reset}_{r-1,r}\ \overbrace{(\mathrm{EL\ S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\mathrm{I\ S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Pump

$$\text{Pump}_{f,r}\colon\ \text{Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Extend Left!

Key: sieve, lift, c_i = insert

A Pump

$$\mathrm{Pump}_{f,r}\colon\ \mathrm{Reset}_{r-1,r}\ \overbrace{(\mathrm{EL\ S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\mathrm{I\ S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Pump

$$\mathrm{Pump}_{f,r}\text{: } \mathrm{Reset}_{r-1,r}\ \overbrace{(\mathrm{EL\ S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\mathrm{I\ S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Extend Left!

Key: sieve, lift, c_i = insert

A Pump

$$\text{Pump}_{f,r}\text{: Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Pump

$$\text{Pump}_{f,r}\text{: Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

$$(c_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Insert!

Key: sieve, lift, c_i = insert

A Pump

$$\text{Pump}_{f,r}\text{: Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

A Pump

$$\text{Pump}_{f,r}\text{: Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, c_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Insert!

Key: sieve, lift, c_i = insert

$$\text{Pump}_{f,r}\text{: Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

$$(b_0, b_1, \ldots, b_{f-1}, b_f, b_{f+1}, \ldots, b_{d-2}, b_{d-1})$$

Sieve!

Key: sieve, lift, c_i = insert

$$\text{Pump}_{f,r}\text{: Reset}_{r-1,r}\ \overbrace{(\text{EL S})^{r-f-1}}^{\text{pump-up}}\ \overbrace{(\text{I S})^{r-f-1}}^{\text{pump-down}}$$

$$(c_0, c_1, \ldots, c_{f-1}, c_f, c_{f+1}, \ldots, c_{d-2}, b_{d-1})$$

Insert!

Key: sieve, lift, c_i = insert

Principles, Sieves and Tweaks

G6K has three high level design principles, to

- recycle (short) vectors between lattices,
- lift vectors, on the fly, to higher dimensional lattices,
- decide the insertion position only *after* sieving.

We implement

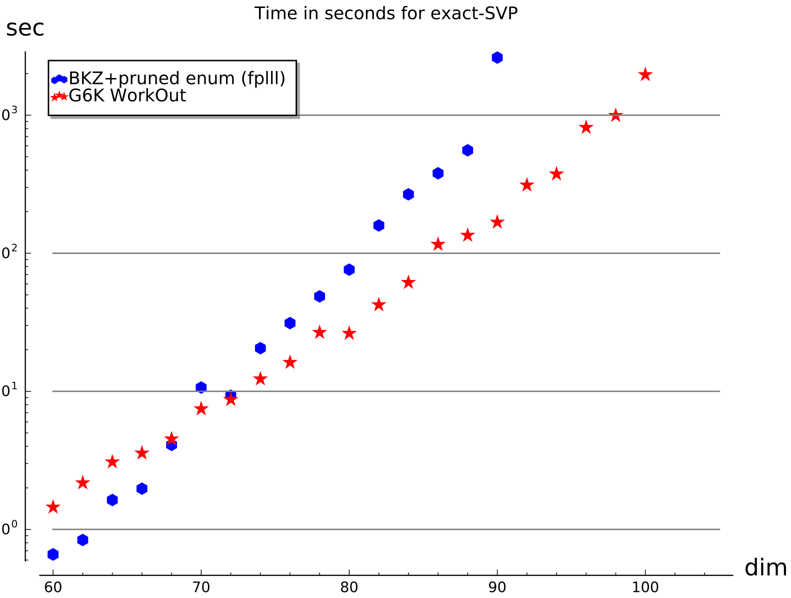
- `bgj1` [BGJ15] and `triple_sieve` [BLS16, HK17],

and make use of algorithmic tweaks

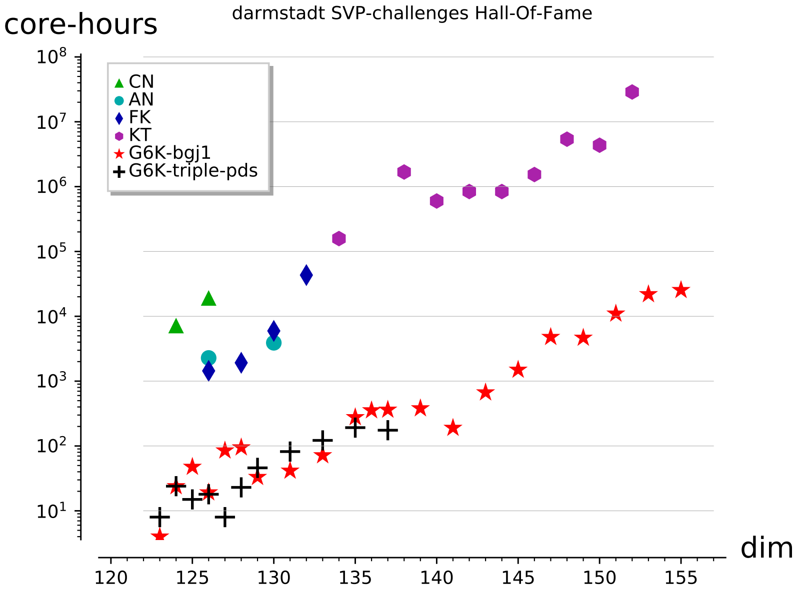
- `XOR-POPCNT`, non terminal insertion, opportunistic dimensions for free, a new database replacement condition...

Records

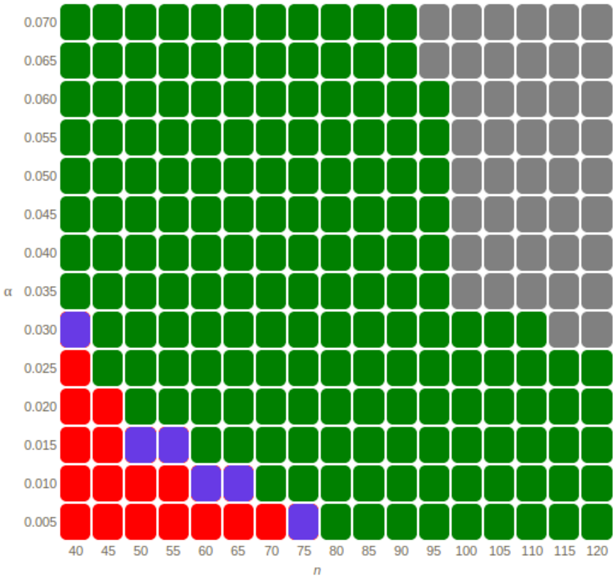
Exact SVP, `Workout` vs. Enumeration (FPLLL) [dt16]



Hermite SVP (Darmstadt Challenges), with `Workout`



LWE (Darmstadt Challenges), with `Pump&JumpBKZ`



Implementation

We have three layers

- c++: multithreaded, heavy operations (sieves, *db* updates)
- cython: middleware, basis maintainance
- python: control, tuning, monitoring

Thanks!

Questions?

L. Babai, *On lovász' lattice reduction and the nearest lattice point problem*, Combinatorica **6** (1986), no. 1, 1–13.

Anja Becker, Nicolas Gama, and Antoine Joux, *Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search*, Cryptology ePrint Archive, Report 2015/522, 2015, `http://eprint.iacr.org/2015/522`.

Shi Bai, Thijs Laarhoven, and Damien Stehlé, *Tuple lattice sieving*, LMS Journal of Computation and Mathematics **19** (2016), no. A, 146—162.

The FPLLL development team, *fplll, a lattice reduction library*, Available at `https://github.com/fplll/fplll`, 2016.

Léo Ducas, *Shortest vector from lattice sieving: A few dimensions for free*, EUROCRYPT 2018, Part I (Jesper Buus

Nielsen and Vincent Rijmen, eds.), LNCS, vol. 10820, Springer, Heidelberg, April / May 2018, pp. 125–145.

Gottfried Herold and Elena Kirshanova, *Improved algorithms for the approximate k-list problem in euclidean norm*, PKC 2017, Part I (Serge Fehr, ed.), LNCS, vol. 10174, Springer, Heidelberg, March 2017, pp. 16–40.

Thijs Laarhoven and Artur Mariano, *Progressive lattice sieving*, Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018 (Tanja Lange and Rainer Steinwandt, eds.), Springer, Heidelberg, 2018, pp. 292–311.