FPLLL: a lattice reduction library

Shi Bai $\in$ FPLLL development team

Workshop HEAT, 2016

FPLLL: overview

What is FPLLL?

- An implementation of **F**loating-**P**oint **LLL** (and BKZ) reduction algorithms.
- A C++ library, under GNU LGPL v2.1 or any later version.
- Created by Damien Stehlé in 2005.
- Rolling release mode.
- Templated and fairly compact $\approx 17,000$ lines.
- Used by Sage.

`https://github.com/fplll/fplll`

Goal: provides benchmarks for lattice reduction algorithms in practice; and lattice reduction for everyone.

FPLLL: features

LLL reduction:

- FP LLL using Cholesky's factorization (Nguyen-Stehlé).
- FP arithmetic:
  - double: 53-bits, fastest.
  - dpe: exponent in an int (Pelissier-Zimmermann).
  - dd/qd: double-double (106 bits) (Bailey).
  - mpfr: arbitrary precision.
- wrapper outputs a provable result using progressively increased precision.
- Integer arithmetic:
  - long int, double, gmp

FPLLL: features and new features

BKZ reduction:

- Floating-point (double) enumeration (Kannan-Fincke-Pohst).
- BKZ reduction algorithm (Schnorr-Euchner).
- Linear pruning.

FPLLL days (June 20 - 24, ENS Lyon):

- BKZ-2.0 (Chen-Nguyen)
  - extreme pruning.
  - recursive pre-processing of blocks.
  - early termination.
  - faster enumeration.
  - various improvements from Albrecht-Ducas-Stevens.
- Slide reduction (Gama-Nguyen).
- Self-dual BKZ (Micciancio-Walter).
- HPLLL: companion to FPLLL (Gilles Villard).

SVP and CVP: HKZ or enumeration or GaussSieve (Micciancio-Voulgaris).

For users

Binaries:

- fplll: main function for LLL, BKZ, HKZ or SVP.
- latticegen: tool for generating random matrices of various types.

```
./fplll [options]
  -a [lll|svp|bkz|sld|sdbkz]
  -m [proved|heuristic|fast|wrapper]
  -z [int|mpz|double]
  -f [mpfr|qd|dd|dpe|double]
  -p <precision>
  -d <delta> and -e <eta>
  -b <blocksize>
  -s <filename.json> load BKZ strategies
  -bkzautoabort
```

Also easy to call libfplll in your program,

```
g++ -I IDIR -L LDIR -lfplll -lmpfr -lgmp
```

For users: examples

LLL

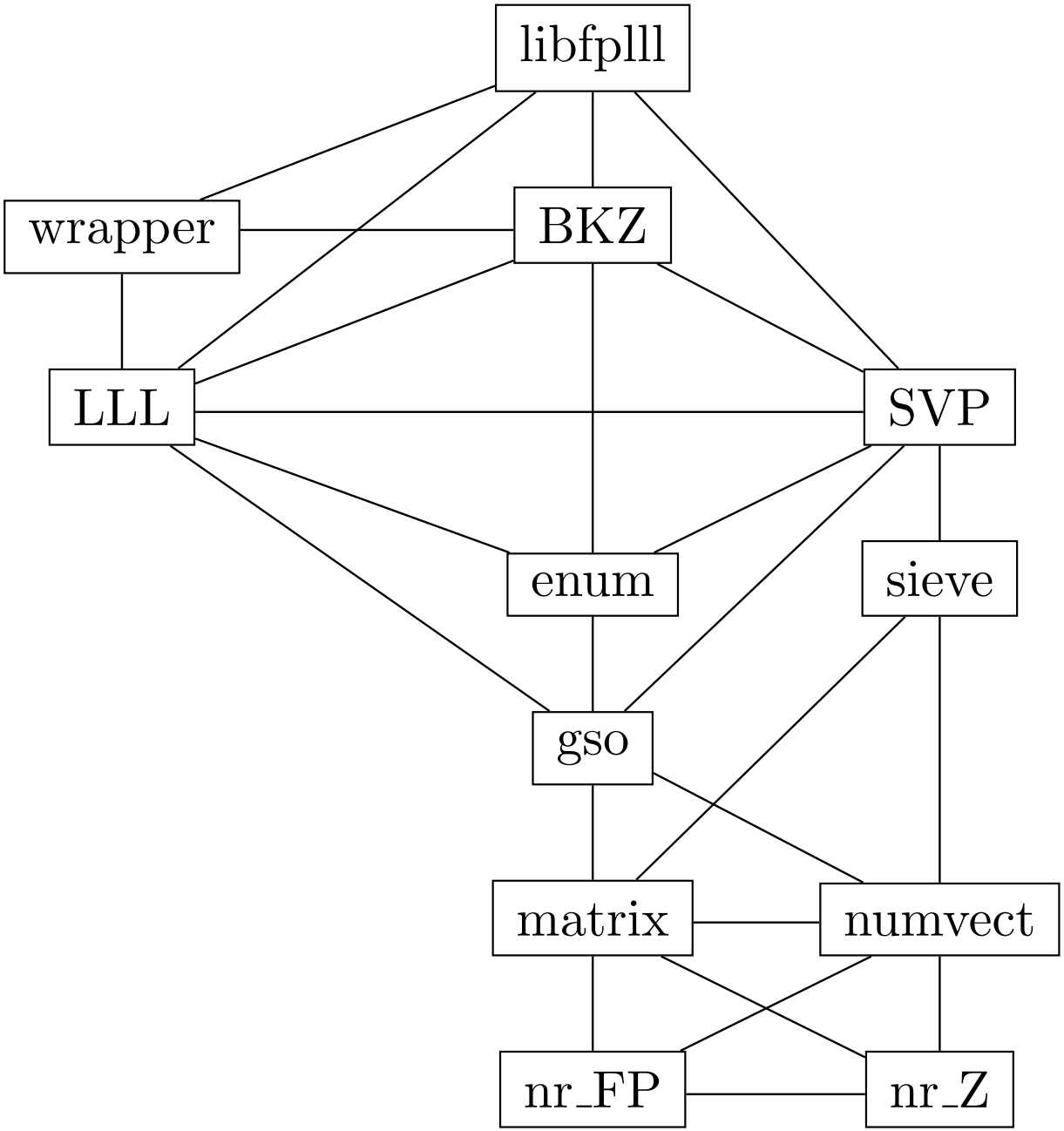
```
./fplll -a lll INPUT -d delta -e eta
```

BKZ 2.0

```
./fplll -a bkz -b 60 svp-challenge-148
 -s ../strategies/default.json -bkzautoabort -v
```

```
0, time = 63s, r_0 = 1.79e8, slope = -0.066553, log2(nodes) = 30.0
1, time = 119s, r_0 = 6.81e7, slope = -0.056137, log2(nodes) = 30.9
2, time = 169s, r_0 = 4.93e7, slope = -0.051181, log2(nodes) = 31.4
3, time = 220s, r_0 = 3.78e7, slope = -0.049026, log2(nodes) = 31.8
4, time = 270s, r_0 = 3.62e7, slope = -0.048141, log2(nodes) = 32.0
5, time = 315s, r_0 = 3.23e7, slope = -0.047673, log2(nodes) = 32.3
6, time = 359s, r_0 = 3.23e7, slope = -0.047619, log2(nodes) = 32.5
7, time = 400s, r_0 = 3.23e7, slope = -0.047489, log2(nodes) = 32.6
8, time = 447s, r_0 = 3.23e7, slope = -0.047557, log2(nodes) = 32.7
9, time = 489s, r_0 = 2.97e7, slope = -0.047375, log2(nodes) = 32.9
...
18, time = 889s, r_0 = 2.93e7, slope = -0.047315, log2(nodes) = 33.7
...
real 16m12.092s
```

For potential contributors



For potential contributors

Developing new lattice reduction algorithms?

- Implementing and testing new lattice-reduction strategies in C++ might be time-consuming.
- Implementing and testing new lattice-reduction strategies in Python might be faster, yet inefficient for LLL/SVP coded in Python.
- Combining them: FPyLLL.

FPyLLL: overview

What is FPyLLL?

- A Python library for performing lattice reduction on integral lattices based on the FPLLL. It's a thin wrapper around fplll.
- implements a few algorithms beyond fplll and provides some convenient interface.
- License GPLv2+.
- Created by Martin R. Albrecht in 2015.

`https://github.com/fplll/fpylll`

Goal: provides a convenient interface for experimenting, development and extension of lattice reduction algorithms.

For users

Example: BKZ algorithm in 70 lines of Python code (copied from Martin's github).

Imports:

```
from fpylll import IntegerMatrix, LLL, GSO
from fpylll import Enumeration as Enum
```

BKZ tour:

```
def bkz_tour(self, block_size, min_row, max_row):
  clean = True
  for kappa in range(min_row, max_row-1):
    bs = min(block_size, max_row - kappa)
    clean &= self.svp_reduction(kappa, bs)
  return clean
```

For users

SVP reduction

```
def svp_reduction(self, kappa, block_size):
  lll_obj(0, kappa, kappa + block_size)
  [setup max_dist, expo]
  solution = Enum.enum(M, max_dist, expo, kappa, \
                       kappa + block_size)
  [concatenate solution]
```

Combine all:

```
while True:
  clean = self.bkz_tour(block_size, 0, A.nrows)
  if clean:
    break
```

Call for testing/contribution

A planned release after the BKZ 2.0 stabilised/polished. Please help for the testing/debugging/benchmarking.

FPLLL and FPyLLL welcome contributions, e.g. the list of open issues.

To contribute,

- clone the github repo;
- commit your code on a separate branch;
- preferably with tests;
- send a pull request.

Contributors of FPLLL:

> Martin R. Albrecht, Shi Bai, Guillaume Bonnoron, David Cade, Léo Ducas, Joop van de Pol, Xavier Pujol, Damien Stehlé, Marc Stevens, Gilles Villard and Michael Walter.

URLs

FPLLL

`https://github.com/fplll/fplll`

FPLLL mailing list: fplll-devel

`fplll-devel@googlegroups.com`

FPYLLL

`https://github.com/fplll/fpylll`