demo

August 13, 2020

1 polymake: Software for Polyhedral Geometry

CO@Work: 15 Sep 2020

1.0.1 Michael Joswig (TU Berlin & MPI for Mathematics in the Sciences)

This notebook/presentation is based on polymake 4.1.

polymake is available from https://polymake.org (where you also find documentation, tutorials, …), or from the squashed mirror at https://github.com/polymake or via the package manager of many Linux distributions.

If you need help browse/post to https://forum.polymake.org/

1.1 Basic Computations With Polytopes

Let us define a rational polytope, given in terms of linear inequalities.

[1]:
```
$k = fractional_knapsack([40, -2,-3,-5]);
```

Non-trivial inequality:

$$40 - 2x - 3y - 5z \geq 0$$

which is the same as

$$2x + 3y + 5z \leq 40$$

This is how we can see what properties are currently known.

[2]:
```
$k->properties();
```

[2]:
```
type: Polytope<Rational>
description: knapsack 40 -2 -3 -5


BOUNDED
true

CONE_AMBIENT_DIM
4

INEQUALITIES
```

```
40 -2 -3 -5
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0
```

A dual convex hull computation gives the vertices.

[3]:
```
print $k->VERTICES;
```

[3]:
```
1 20 0 0
1 0 40/3 0
1 0 0 0
1 0 0 8
```

[4]:
```
$k->properties();
```

[4]:
```
type: Polytope<Rational>
description: knapsack 40 -2 -3 -5


BOUNDED
true

CONE_AMBIENT_DIM
4

FEASIBLE
true

INEQUALITIES
40 -2 -3 -5
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0


LINEALITY_DIM
0

LINEALITY_SPACE
```

```
POINTED
true

VERTICES
1 20 0 0
1 0 40/3 0
1 0 0 0
1 0 0 8
```

Solving a linear program.

[5]:
```
$lp = $k->LP(LINEAR_OBJECTIVE=>new Vector([0,1,2]));
print $lp->MAXIMAL_VERTEX, "\n", $lp->MAXIMAL_VALUE;
```

[5]:
```
1 0 40/3 0
80/3
```

Solving an integer linear program goes through the explicit construction of the integer hull. This requires to enumerate all lattice points in the feasible region.

[6]:
```
$ik = integer_hull($k);
print $ik->VERTICES;
```

[6]:
```
1 0 0 0
1 0 0 8
1 0 10 2
1 0 13 0
1 2 12 0
1 20 0 0
```

Now the ILP becomes an LP.

[7]:
```
$ilp = $ik->LP(LINEAR_OBJECTIVE=>new Vector([0,1,2]));
print $ilp->MAXIMAL_VERTEX, "\n", $ilp->MAXIMAL_VALUE;
```

[7]:
```
1 2 12 0
26
```

polymake knows several ways to visualize polytopes and other objects. For instance, you can try to omit "svg(…)" in the command below.

[8]:
```
svg($k->VISUAL->LATTICE_COLORED);
```

By the way, there are many algorithms for and implementations of methods for computing convex hulls (many of which are available in polymake).

A recent survey with extensive computational experiments is: Benjamin Assarf, Ewgenij Gawrilow, Katrin Herr, Michael Joswig, Benjamin Lorenz, Andreas Paffenholz and Thomas Rehn: Computing convex hulls and counting integer points with polymake. Math. Program. Comput. 9 (2017), no. 1, 1-38. https://link.springer.com/article/10.1007/s12532-016-0104-z

1.2 Other Things To Check Out

Originally, polymake was only about polytopes, polyhedra and linear programs. Over the years many other objects from other areas of discrete mathematics were added. Below you see a tiny fraction only.

1.2.1 Combinatorial topology

[9]:
```
application "topaz";
```

[10]:
```
$p = real_projective_plane();
print rows_labeled($p->HOMOLOGY);
```

[10]:
```
0:{} 0
1:{(2 1)} 0
2:{} 0
```

1.2.2 Tropical geometry builds bridges between optimization and algebraic geometry

[11]:
```
application "tropical";
```

[12]:
```
$x = new TropicalNumber<Min>(3);
$y = new TropicalNumber<Min>(5);
print "tropical sum = ", $x+$y, ", tropical product = ", $x*$y;
```

[12]:
```
tropical sum = 3, tropical product = 8
```

[13]:
```
$M = new Matrix<TropicalNumber<Min>>([[0,1,2],[5,0,1],["inf",4,0]]);
print $M + $M*$M;
```

[13]:
```
0 1 2
5 0 1
9 4 0
```

[14]:
```
print tdet($M);
```

[14]:
```
0
```

In case you find to find out more about tropical geometry and how this is relevant for optimization, check out my book project (that I hope to complete soon): Essentials of Tropical Combinatorics http://page.math.tu-berlin.de/~joswig/etc/index.html

1.3 Polytopes and Linear Programs Over Fields of Rational Functions

Polytopes are defined over any ordered field.

[15]:
```
application "polytope";
```

Produce polytopes, depending on a parameter t, which may seen as infinitesimally small.

[16]:
```
$monomial=monomials<Rational,Rational>(1);
local_var_names<UniPolynomial<Rational,Rational>>(qw(t));
$t = new PuiseuxFraction<Min>($monomial);
$km = klee_minty_cube(3,$t);
print $km->VERTICES;
```

[16]:
```
(1) (0) (0) (0)
(1) (1) (t) (t^2)
(1) (0) (1) (t)
(1) (1) (1 - t) (t - t^2)
(1) (0) (0) (1)
(1) (1) (t) (1 - t^2)
(1) (0) (1) (1 - t)
(1) (1) (1 - t) (1 - t + t^2)
```

Convex hull computations, linear programs etc just work the same as over the rationals.

[17]:
```
print $km->FACETS;
```

[17]:
```
(0) (1) (0) (0)
(1) (- 1) (0) (0)
(0) (- t) (1) (0)
(1) (- t) (- 1) (0)
(0) (0) (- t) (1)
(1) (0) (- t) (- 1)
```

[18]:
```
print $km->VOLUME;
```

[18]:
```
(1 -2*t + t^2)
```

[19]:
```
print evaluate($km->VOLUME,1/100);
```

[19]:
```
9801/10000
```

[20]:
```
print evaluate_float($km->VOLUME,0.0053526);
```

[20]:
```
0.98932345032676
```

polymake has an online help system.

Use “apropos” for searching (i.e., to find the name of a function you do not know exactly).

Use “help” for a specific function whose name you know.

[21]:
```
help "long_and_winding";
```

[21]:
```
polytope/functions/Producing a polytope from scratch/long_and_winding:
long_and_winding(r; Options) -> Polytope<PuiseuxFraction<Max, Rational,
Rational> >

 Produce polytope in dimension 2r with 3r+2 facets such that the total curvature
 of the central path is at least Omega(2^r); see
 Allamigeon, Benchimol, Gaubert and Joswig, SIAM J. Appl. Algebra Geom. (2018).
 See also perturbed_long_and_winding.

Arguments:
  Int r defining parameter

Options:
  eval_ratio => Rational parameter for evaluating the puiseux rational
functions
  eval_exp => Int to evaluate at eval_ratio^eval_exp, default: 1
  eval_float => Float parameter for evaluating the puiseux rational
functions

Returns Polytope<PuiseuxFraction<Max, Rational, Rational> >

Examples:

*) This yields a 4-polytope over the field of Puiseux fractions.
   > $p = long_and_winding(2);
*) This yields a rational 4-polytope with the same combinatorics.
   > $p = long_and_winding(2,eval_ratio=>2);
```

2 This is the END.