tropical_computations

January 31, 2018

1 Tropical Computations in polymake

Based on - Hampe & Joswig: Tropical computations in polymake in: Algorithmic and Experimental Methods in Algebra, Geometry, and Number Theory (Böckle, Decker & Malle, eds.), Springer 2018, to appear; preprint **arXiv:1612.02581**.

This document is available as **Jupyter notebook** and **pdf**.

We start out by switching to the application for tropical computations.

```
In [1]: application "tropical";
```

1.1 Arithmetic and Linear Algebra

The arithmetic requires to choose between min and max as the tropical addition. There is no default!

```
In [2]: $a = new TropicalNumber<Min>(3);
        $b = new TropicalNumber<Min>(5);
        $c = new TropicalNumber<Min>(8);
        print "a * c = ", $a*$c, ", b + c = ", $b+$c;
```

```
Out[2]: a * c = 11, b + c = 5
```

Computations with matrices work as usual. Here we see how determining all shortest paths in a finite digraph can be reduced to tropical matrix computation.

```
In [3]: $A = new Matrix<TropicalNumber<Min>>([[1,2,3],[1,2,4],[1,0,1]]);
        $I = new Matrix<TropicalNumber<Min>>([[0,"inf","inf"],["inf",0,"inf"],["inf","inf",0]])
        $B = $I + $A + $A*$A;
        print $B, "\n";
        print "idempotent? ", $B*$B == $B;
```

```
Out[3]: 0 2 3
        1 0 4
        1 0 0

        idempotent? 1
```

Computing the tropical determinant is equivalent to solving a linear assignment problem.

```
In [4]: $A = new Matrix<TropicalNumber<Min>>([[1,0,1],[1,1,0],[1,10,"inf"]]);
        print tdet($A);
```

```
Out[4]: 1
```

This is how to find an optimal assignment.

```
In [5]: print tdet_and_perm($A),"\n";
        $sigma = tdet_and_perm($A)->second;
        print $A->elem(0,$sigma->[0]) * $A->elem(1,$sigma->[1]) * $A->elem(2,$sigma->[2]);
```

```
Out[5]: 1 <1 2 0>
        1
```

1.2 Hypersurfaces

Let's look at a cubic surface in the tropical 3-torus. Throughout we employ homogeneous coordinates.

```
In [6]: $F = toTropicalPolynomial("min(12+3*x0,-131+2*x0+x1,
        -67+2*x0+x2,-9+2*x0+x3,-131+x0+2*x1,-129+x0+x1+x2,
        -131+x0+x1+x3,-116+x0+2*x2,-76+x0+x2+x3,-24+x0+2*x3,-95+3*x1,
        -108+2*x1+x2,-92+2*x1+x3,-115+x1+2*x2,-117+x1+x2+x3,
        -83+x1+2*x3,-119+3*x2,-119+2*x2+x3,-82+x2+2*x3,-36+3*x3)");
        $S = new Hypersurface<Min>(POLYNOMIAL=>$F);
        print $S->AMBIENT_DIM, " ", $S->DEGREE;
```

```
Out[6]: 4 3
```

```
In [7]: $S->VISUAL(FacetColor=>"orange",FacetTransparency=>0.5);
```

1.3 Moduli of Tropical Plane Curves

We now switch to the application for polyhedral fans.

```
In [8]: application "fan";
```

On the 15 lattice points in four times the unit triangle we define a triangulation.

```
In [9]: $points = [ [0,0,4],[1,0,3],[0,1,3],[2,0,2],[1,1,2],
        [0,2,2], [3,0,1],[2,1,1],[1,2,1], [0,3,1],[4,0,0],[3,1,0],
        [2,2,0],[1,3,0],[0,4,0] ];
        $triangulation = [[0,1,2],[9,11,12],[9,12,13],
        [9,13,14],[1,2,5],[6,10,11],[3,6,11],[1,5,9],[1,3,11],
        [8,9,11],[1,4,9],[1,7,11],[7,8,11],[4,8,9],[1,4,7],[4,7,8]];
        $pointMatrix = (ones_vector<Rational>(15)) | (new Matrix<Rational>($points));
        $Sigma = new SubdivisionOfPoints(POINTS=>$pointMatrix, MAXIMAL_CELLS=>$triangulation);
        $Sigma->VISUAL;
```

Let's see how many rays the sceondary fan has.

```
In [10]: $sc=$Sigma->secondary_cone();
         print $sc->N_RAYS;

Out[10]: 12
```

This is like the "factorization" of the secondary cone into "primes".

```
In [11]: for (my $i=0; $i<$sc->N_RAYS; ++$i) {
           my $c=new SubdivisionOfPoints(POINTS=>$pointMatrix,WEIGHTS=>$sc->RAYS->[$i]);
           print $i, ":", $c->N_MAXIMAL_CELLS, " ";
         }

Out[11]: 0:2 1:2 2:3 3:2 4:2 5:3 6:2 7:2 8:2 9:2 10:2 11:3
```

Switching back.

```
In [12]: application "tropical";
```

Now let's make a tropical curve which is dual to the triangulation above.

```
In [13]: $C = new Hypersurface<Min>(MONOMIALS=>$points, COEFFICIENTS=>[6, 0,3, 1,-1/3,1, 3,-1/3
         $ratCoeff = new Vector<Rational>($C->COEFFICIENTS);
         print $sc->FACETS * $ratCoeff;
         print $sc->LINEAR_SPAN * $ratCoeff;

Out[13]: 4 4 8/3 4 4 4 4 8/3 8/3 4/3 4/3 4/3

In [14]: print $C->DEGREE, " ", $C->GENUS;

Out[14]: 4 3

In [15]: $C->VISUAL;
```

1.4 Tropical Grassmannians

A tropical linear (or valuated matroid) space comes about from a lifting function on the vertices of a hypersimplex.

```
In [16]: application "polytope";

In [17]: $Delta=hypersimplex(3,6);
         $p=new Vector<Int>([0,0,3,1,2,1,0,1,0,2,2,0,3,0,4,1,2,2,0,0]);

In [18]: application "fan";

In [19]: $tlinear=new SubdivisionOfPoints(POINTS=>$Delta->VERTICES,WEIGHTS=>$p);
         print $tlinear->MAXIMAL_CELLS;
```

```
Out[19]: {0 1 3 5 6 7 8 9 11 13 15 18 19}
         {0 1 3 6 8 10 11 12 13 15 17 18 19}
         {0 1 4 5 6 7 8 11 13 16 18 19}
         {0 1 4 6 8 10 11 13 16 17 18 19}
         {1 3 8 10 12 13 14 15 17 19}
         {0 1 2 3 5 7 9 11 13 15}
```

Every cell should be the matroid polytope of some matroid. How can we check this?

```
In [20]: application "matroid";

In [21]: $bases = new IncidenceMatrix(uniform_matroid(3,6)->BASES);
         @subdiv_bases = map { new Array<Set>(rows($bases->minor($_,All))) } @{$tlinear->MAXIMA
         print join(",", map { check_basis_exchange_axiom($_) } @subdiv_bases );

Out[21]: 1,1,1,1,1,1
```

Is this tropical linear space realizable?

```
In [22]: application "ideal";

In [23]: $I=pluecker_ideal(3,6);
         $pp=new Vector<Int>(5*ones_vector(20)-$p); # Singular works with max only, and it neec
         $J=new Ideal(GENERATORS=>$I->GROEBNER(ORDER_VECTOR=>$pp)->INITIAL_FORMS);
         print $J->contains_monomial();

Out[23]: 0
```

1.5 Tropical Convexity

```
In [24]: application "tropical";

In [25]: $C = cyclic<Min>(2,6);
         print $C->POINTS;
         $C->VISUAL;

Out[25]: 0 0 0
         0 1 2
         0 2 4
         0 3 6
         0 4 8
         0 5 10

In [26]: print $C->POLYTOPE_MAXIMAL_COVECTORS;

Out[26]: <{5}
         {5}
         {0 1 2 3 4}
         >
```

```
<{1 2 3 4 5}
{0}
{0}
>
<{5}
{4}
{0 1 2 3}
>
<{5}
{3 4}
{0 1 2}
>
<{4 5}
{3}
{0 1 2}
>
<{5}
{2 3 4}
{0 1}
>
<{4 5}
{2 3}
{0 1}
>
<{4 5}
{1 2 3}
{0}
>
<{5}
{1 2 3 4}
{0}
>
<{3 4 5}
{1 2}
{0}
>
<{3 4 5}
{2}
{0 1}
>
<{2 3 4 5}
{1}
{0}
>
```