Computing Tropical Prevarieties in Parallel

Jan Verschelde[1]
joint work with Anders Jensen[2] and Jeff Sommars[1]

[1]University of Illinois at Chicago
[2]Aarhus University

The 8th International Workshop on Parallel Symbolic Computation,
23–24 July 2017, Kaiserslautern, Germany.

Outline

1. introduction
   - problem statement
   - software contributions
2. algorithms
   - half open cones
   - static enumeration
   - dynamic enumeration
3. software
   - parallel algorithms
   - computational results

power series as solutions of polynomial systems

The Newton-Puiseux algorithm on a polynomial f in two variables computes power series expansions for the plane algebraic curve.

The leading powers of the expansions are computed as vectors perpendicular to the edges of the Newton polyhedron P of f.

Generalize the Newton-Puiseux algorithm:

- Each face of P has a normal cone C: all nonzero vectors which make the same minimal inner product with all points in the face.
- The tropical hypersurface T of f is the set of all normal cones that are not maximal, i.e.: no cones normal to vertices.
- Given a tuple $\mathbf{f}$ of polynomials, the tropical prevariety is the intersection of all tropical hypersurfaces of $f \in \mathbf{f}$.

Our problem: compute the tropical prevariety in parallel.

background literature

Two key publications:

- T. Bogart, A.N. Jensen, D. Speyer, B. Sturmfels, and R.R. Thomas: **Computing tropical varieties.** *Journal of Symbolic Computation* 42(1): 54–73, 2007.
- Anders Nedergaard Jensen: **Algorithmic Aspects of Gröbner Fans and Tropical Varieties.** Ph.D. Dissertation, University of Aarhus, 2007.

software

The mixed volume of the Newton polytopes is a generically sharp upper bound on the number of isolated solutions in $(\mathbb{C} \setminus \{0\})^n$.

Implementations of dedicated algorithms for mixed volumes:

- Mixvol, Ioannis Emiris, 1993
- PHCpack, Jan Verschelde, 1999
- MixedVol, Tangan Gao, Tien-Yien Li, and Mengnien Wu, 2005
- DEMiCs, Tomohiko Mizutani and Akiko Takeda, 2008
  DEMiCs = Dynamic Enumeration of Mixed Cells
- pss 5, Gregorio Malajovich, 2015
- Gfan 0.6, Anders Jensen, 2016

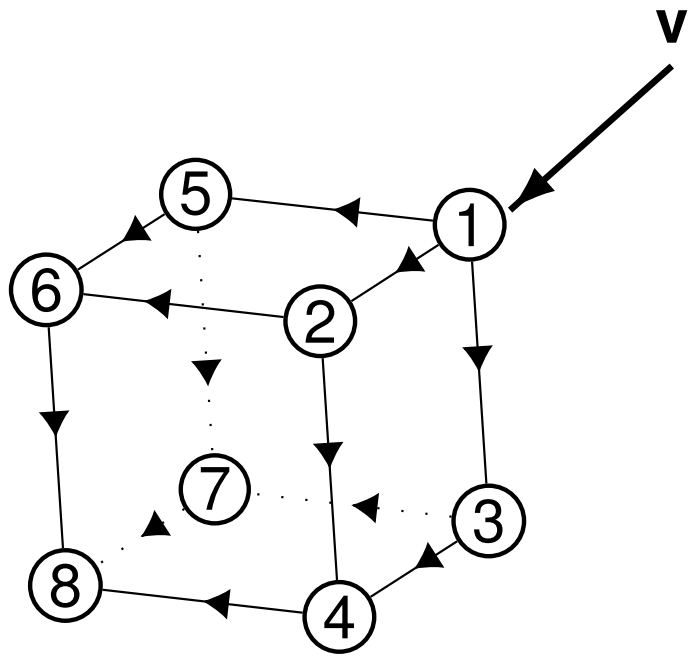
Software for tropical computations:

- Gfan, Anders Jensen, 2006

our contributions

1. Application of dynamic enumeration to the computation of the half open cones in the tropical prevariety.
2. A parallel shared memory implementation with work stealing.
3. Computed the tropical prevariety of cyclic 16-roots.
4. GPL licensed software `DynamicPrevariety` at `https://github.com/sommars/DynamicPrevariety`.

orienting a cube by a vector **v**



A half open cone at a vertex **a** of P is defined as follows.
For each edge **e** incident to **a**, construct an inequality:

- if **e** is outgoing, then the inequality is strict,
- if **e** is ingoing, then the inequality is not strict.

constructing half open cones

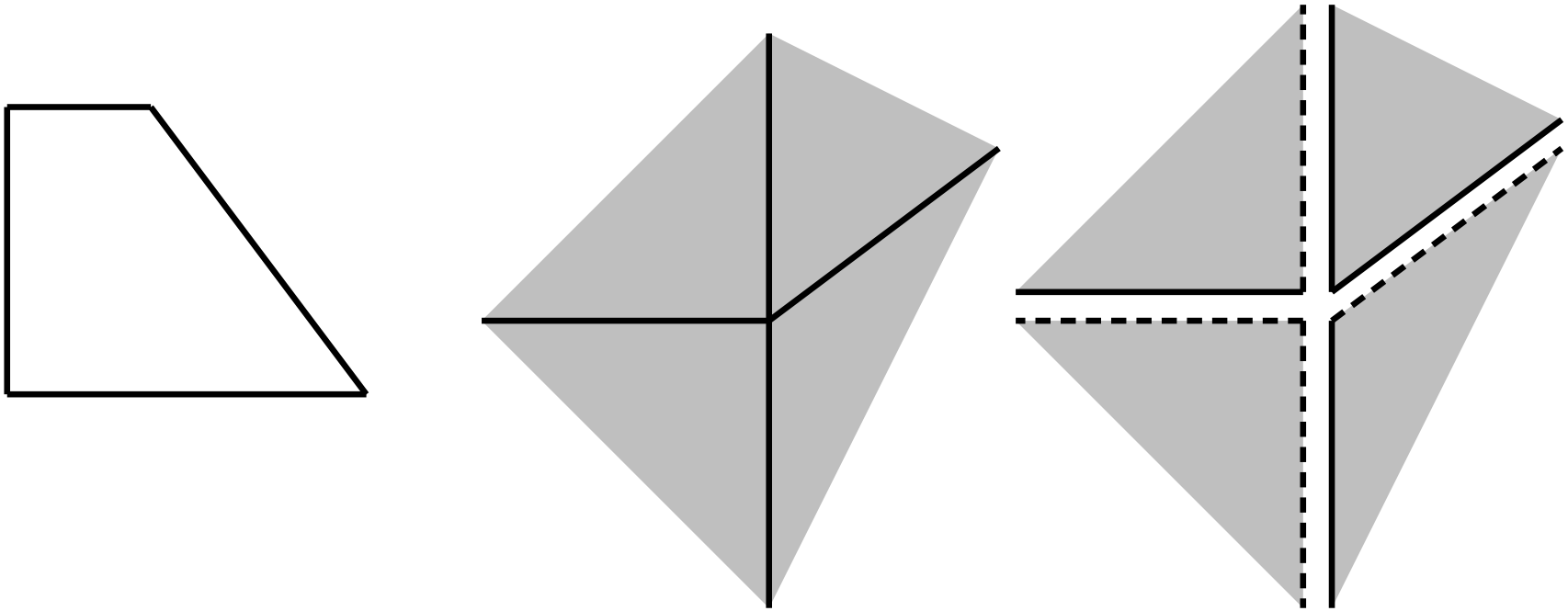
Algorithm 1 Partition a full dimensional cone in half open cones

Input: An inequality description of a full dimensional half open cone C.

Output: A collection of disjoint half open cones with union equal to the boundary of C.

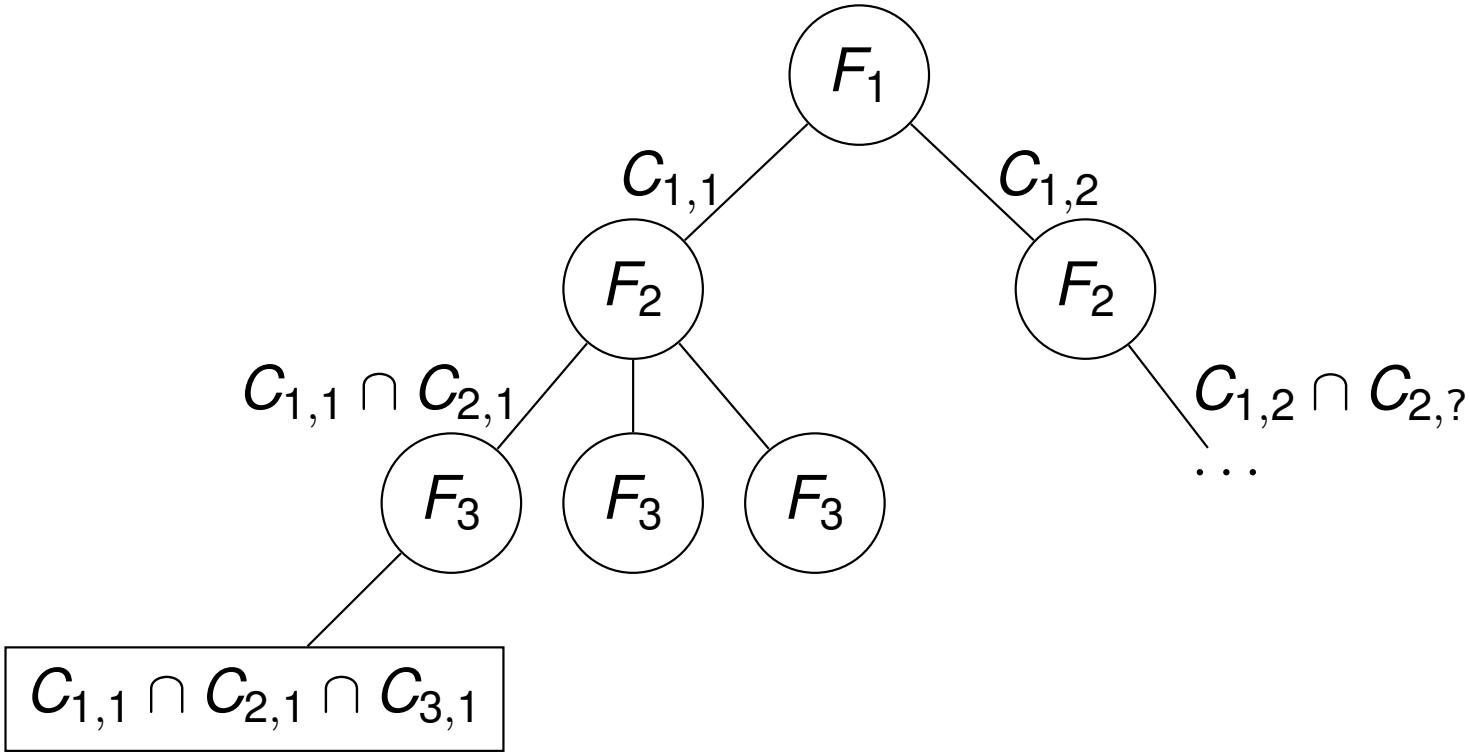
```
function CreateHalfOpenCones(C)
    if C has only strict constraints then return ∅
    else
        Choose a non-strict constraint c of C
        C_< := C but with c being strict
        C_= := C but with c being an equation
        return C_= ∪ CreateHalfOpenCones(C_<)
    end if
end function
```

a partition into four half open cones



static enumeration

F_i is a list of normal cones $C_{i,j}$ of the ith Newton polytope.



Algorithm 2 Static enumeration

Input: A list F of fans $F_1, \ldots F_N$ in $\mathbb{R}^n$ where each F_i is represented by a list of cones covering the support of F_i.

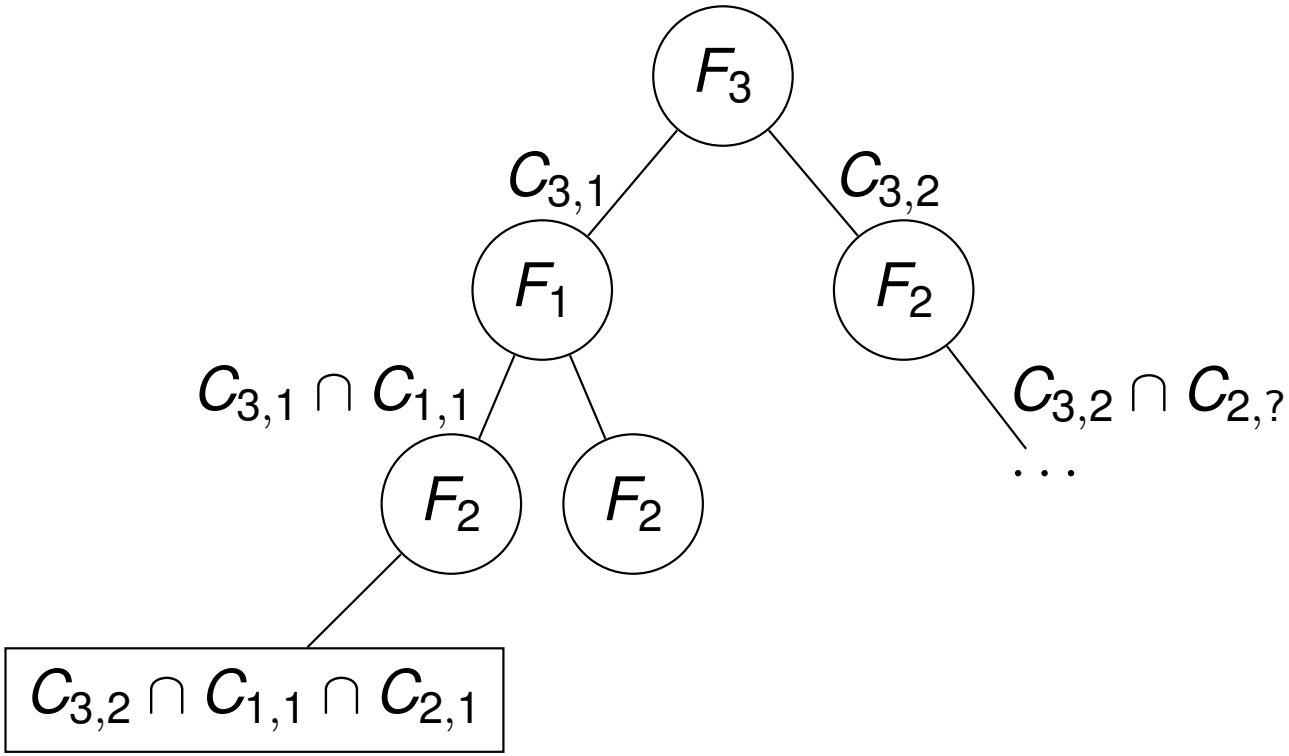
Output: A list of cones covering the support of $F_1 \wedge \cdots \wedge F_N$.

```
    procedure StaticEnumeration(Cone C, Index i)
        if C ≠ ∅ then
            if i > |F| then
                Output C
 5:         else
                for each cone D in F_i do
                    StaticEnumeration(C ∩ D, i + 1)
                end for
            end if
10:     end if
    end procedure
    StaticEnumeration(ℝ^n, 1)
```

dynamic enumeration

Apply a greedy search to minimize the number of intersections:

1. Select the fan with the fewest cones at the start.
2. Use information of pairwise cone intersections.



Algorithm 3 Dynamic enumeration

Input: A list F of fans $F_1, \ldots F_N$ in $\mathbb{R}^n$ where each F_i is represented by a list of cones covering the support of F_i.

Output: A list of cones covering the support of $F_1 \wedge \cdots \wedge F_N$.

```
    procedure DynamicEnumeration(Cone C, Set I)
        if C ≠ ∅ then
            if I = ∅ then
                Output C
 5:         else
                Greedily choose index i ∈ I.
                for each cone D in F_i do
                    DynamicEnumeration(C ∩ D, I \ {i})
                end for
10:         end if
        end if
    end procedure
    DynamicEnumeration(ℝ^n, {1, ..., |F|})
```

relation tables

To implement the "Greedily choose index" we apply relation tables:

- introduced by T. Gao and T.Y. Li in 2003;
- store whether or not pairs of cones could intersect.

Denote $C_{i,j}$ the jth cone of fan $T(P_i)$.
For a cone C of fan $T(P)$, *the relation table* $R(i,j)$ is a boolean array

$$R(i,j) = \begin{cases} 1, & \text{if } C \cap C_{i,j} \neq \emptyset, \\ 0, & \text{if } C \cap C_{i,j} = \emptyset, \\ 0, & \text{if } P = P_i. \end{cases}$$

where $1 \leq i \leq N$ and $1 \leq j \leq \#\text{Edges}(P_i)$.

Algorithm 4 Greedy choice of index

procedure DYNAMICENUMERATION(Cone C, Set I)
  . . . omitted code is the same as before . . .
  Choose index $i \in I$ such that F_i has fewest
    cones which C could intersect.
  for each cone D in F_i **do**
    if C's relation table allows $C \cap D \neq \emptyset$ **then**
      Intersect C's relation table with D's
        relation table, and store on $C \cap D$
      DYNAMICENUMERATION($C \cap D$, $I \setminus \{i\}$)
    end if
  end for
end procedure
Compute relation tables for $\mathbb{R}^n$ and the cones in F
DYNAMICENUMERATION($\mathbb{R}^n$, $\{1, \ldots, |F|\}$)

the Parma Polyhedral Library (PPL)

Our first algorithms (in CASC 2016) are implemented in Sage.

- T. Bogart, M. Hampton, and W. Stein: `groebner_fan` module of Sage, 2008.
- V. Braun and M. Hampton: `polyhedra` module of Sage, 2011.
- A.Y. Novoseltsev: `lattice_polytope` module of Sage, 2011.
- R. Bagnara, P.M. Hill, and E. Zaffanella. **The Parma Polyhedral Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.** *Science of Computer Programming*, 2008.

Enea Zaffanella developed a thread safe version of PPL.

- Exact arithmetic with arbitrary precision integers (GMP).
- Improved speedups are achieved using the allocator TCMalloc.

design of a parallel algorithm

We distinguish three stages:

1. Compute all vertex points of all Newton polytopes.
   For small Newton polytopes spanned by relatively few monomials, this stage takes less than a second for a single thread.
2. Parallel computation of the relation tables.
   We intersect all pairs of cones, processing a job queue.
   One job is the intersection of two polyhedral cones.
3. Parallel dynamic enumeration.
   - Coarse grained by forking processes, dividing cones of the first fan, is inefficient due to the difference in work loads.
   - The application of work stealing gives good results.

iterative version of dynamic enumeration

A parallel implementation needs a job queue,
provided by an iterative version of the algorithm.

Algorithm 5 Iterative version of dynamic enumeration — initialization

---

Input: A list of fans $F_1, \ldots, F_N$ in $\mathbb{R}^n$ where each F_i is represented by a list of cones covering the support of F_i.

Output: A list of cones covering the support of $F_1 \wedge \cdots \wedge F_N$.

```
    Compute relation tables
    F := fan with fewest cones
    Cones := Cones from F
    while Cones≠ ∅ do
5:      ... code executed by the main loop ...
    end while
```

The "Cones" is the job queue.

Algorithm 6 Iterative version of dynamic enumeration —- main loop

```
    while Cones≠ ∅ do
        C := remove an element from Cones
        Choose fan F′ not used to produce C such that F′
            has fewest cones with which C could intersect.
 5:     for each cone D in F′ do
            if C's relation table allows C ∩ D ≠ ∅ then
                Compute C ∩ D
                if C ∩ D ≠ ∅ then
                    if C ∩ D used all fans then
10:                     Output C ∩ D
                    else
                        Intersect C's relation table with D's
                            relation table, and store on C ∩ D
                        Add C ∩ D to Cones
15:                 end if
                end if
            end if
        end for
    end while
```

work stealing

We use the parallel runtime library provided by PPL.

- Each thread has its own job queue.
- If there are p threads, numbered from 1 to p,
  then the ith thread looks to steal from the threads in the order:

$$i+1, i+2, \ldots, p, 1, 2, \ldots, i-1.$$

hardware and software

We compare against Gfan which contains an implementation of a variant of the dynamic enumeration algorithm.

Gfan timings are for a single thread running on an Intel Xeon E2670. SoPlex (Wunderling, 1996) was enabled in Gfan, providing a speed up of roughly a factor 3.

Except for the Gfan timings, all computations were done on a 2.2 GHz Intel Xeon E5-2699 processor in a CentOS Linux workstation with 256 GB RAM using varying numbers of threads.

the cyclic 16-roots problem

The cyclic 16-roots problem is an academic benchmark problem.

$$\begin{cases} x_0 + x_1 + \cdots + x_{n-1} = 0 \\ i = 2, 3, \ldots, n-1 : \sum_{j=0}^{n-1} \prod_{k=j}^{j+i-1} x_{k \bmod n} = 0 \\ x_0 x_1 x_2 \cdots x_{n-1} - 1 = 0. \end{cases}$$

Specifics for $n = 16$:

- By Backelin's lemma, there is a 3-dimensional solution set.
- DEMiCs computed the mixed volume 135,555,072.
  This result was published in 2008.

number of cone intersections

n	static enumeration	dynamic enumeration	number of rays
4	114	114	2
5	682	676	0
6	2,286	2,254	8
7	7,397	7,163	28
8	19,619	18,315	94
9	63,109	50,584	276
10	269,223	160,203	712
11	1,625,520	827,469	2,244
12	11,040,912	5,044,441	5,582
13		36,633,391	14,872
14		264,463,730	49,114
15		1,852,158,881	145,276
16		13,715,434,028	527,126

timings

n	Gfan	1 thread	10 threads	20 threads
4	0.020s	0.008s	0.017s	0.028s
5	0.058s	0.036s	0.053s	0.073s
6	0.22s	0.10s	0.11s	0.16s
7	0.64s	0.29s	0.26s	0.37s
8	2.87s	0.79s	0.49s	0.70s
9	13.0s	2.8s	1.2s	1.4s
10	1m22s	9.8s	4.4s	3.7s
11	9m17s	50s	16.8s	20.3s
12	82m33s	5m2s	1m5s	1m3s
13		46m59s	8m30s	6m20s
14		6h22m56s	67m31s	46m37s
15			10h25m45s	7h43m57s
16			84h20m37s	62h36m31s

the number of maximal cones

A maximal cone is not contained in any other cone.

The number of maximal cones by dimension of cyclic 16-roots:

dim	#cones
1	0
2	768
3	114,432
4	1,169,792
5	1,007,616
6	2,443,136
7	4,743,904
8	109,920