Computing Tropical Prevarieties in Parallel

Jan Verschelde\textsuperscript{1}
joint work with Anders Jensen\textsuperscript{2} and Jeff Sommars\textsuperscript{1}

\textsuperscript{1}University of Illinois at Chicago
\textsuperscript{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:


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.
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 CREATE HALF OPEN CONES($C$)

if $C$ has only strict constraints then return $\emptyset$
else
    Choose a non-strict constraint $c$ of $C$
    5: $C_\prec := C$ but with $c$ being strict
    $C_\equiv := C$ but with $c$ being an equation
    return $C_\equiv \cup$ CreateHalfOpenCones($C_\prec$)
end if
end function
a partition into four half open cones
$F_i$ is a list of normal cones $C_{i,j}$ of the $i$th 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 $F_1 \land \cdots \land F_N$.

procedure STATICENUMERATION(Cone $C$, Index $i$)
    if $C \neq \emptyset$ then
        if $i > |F|$ then
            Output $C$
        else
            for each cone $D$ in $F_i$ do
                STATICENUMERATION($C \cap D$, $i + 1$)
            end for
        end if
    end if
end procedure

STATICENUMERATION($\mathbb{R}^n$, 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 DYNAMIC_ENUMERATION(Cone $C$, Set $I$)
    if $C \neq \emptyset$ then
        if $I = \emptyset$ then
            Output $C$
        else
            Greedily choose index $i \in I$.
            for each cone $D$ in $F_i$ do
                DYNAMIC_ENUMERATION($C \cap D$, $I \setminus \{i\}$)
            end for
        end if
    end if
end procedure

DYNAMIC_ENUMERATION($\mathbb{R}^n$, \{1, \ldots, |F|\})
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.
  5: 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($C \cap D$, $I \setminus \{i\}$)
  10: 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.


Enea Zaffanella developed a thread safe version of PPL.

- Exact arithmetic with arbitrary precision integers (GMP).
- Improved speedups are achieved using the allocator TCMalloc.
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 $F_1 \land \cdots \land F_N$.

Compute relation tables
$F :=$ fan with fewest cones
Cones := Cones from $F$
while Cones $\neq \emptyset$ 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

\begin{algorithm}
\While{Cones \neq \emptyset} {
    \State $C :=$ remove an element from Cones
    Choose fan $F'$ not used to produce $C$ such that $F'$ has fewest cones with which $C$ could intersect.
    \For{each cone $D$ in $F'$} {
        \If{$C$'s relation table allows $C \cap D \neq \emptyset$} {
            Compute $C \cap D$
            \If{$C \cap D \neq \emptyset$} {
                \If{$C \cap D$ used all fans} {
                    Output $C \cap D$
                } \Else {
                    Intersect $C$'s relation table with $D$'s relation table, and store on $C \cap D$
                    Add $C \cap D$ to Cones
                }
            }
        }
    }
}
\end{algorithm}
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 $i$th 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 is an academic benchmark problem.

\[
\begin{align*}
    x_0 + x_1 + \cdots + x_{n-1} &= 0 \\
    i = 2, 3, \ldots, n-1 : \sum_{j=0}^{n-1} \prod_{k=j}^{j+i-1} x_k \mod n &= 0 \\
    x_0x_1x_2 \cdots x_{n-1} - 1 &= 0.
\end{align*}
\]

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

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

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

<table>
<thead>
<tr>
<th>dim</th>
<th>#cones</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>768</td>
</tr>
<tr>
<td>3</td>
<td>114,432</td>
</tr>
<tr>
<td>4</td>
<td>1,169,792</td>
</tr>
<tr>
<td>5</td>
<td>1,007,616</td>
</tr>
<tr>
<td>6</td>
<td>2,443,136</td>
</tr>
<tr>
<td>7</td>
<td>4,743,904</td>
</tr>
<tr>
<td>8</td>
<td>109,920</td>
</tr>
</tbody>
</table>