Massively parallel methods in computer algebra

Janko Boehm
joint with Wolfram Decker, Anne Frühbis-Krüger,
Franz-Josef Pfreundt, Mirko Rahn, Lukas Ristau

Technische Universität Kaiserslautern

April 03, 2018

supported by

TR 195
SYMBOLIC TOOLS

Outline

- Determining smoothness of algebraic varieties

Outline

- Determining smoothness of algebraic varieties

- Infrastructure for massively parallel computations: GPI-Space
- SINGULAR and GPI-Space

Outline

- Determining smoothness of algebraic varieties

- Infrastructure for massively parallel computations: GPI-Space
- SINGULAR and GPI-Space

- Timings

Outline

- Determining smoothness of algebraic varieties

- Infrastructure for massively parallel computations: GPI-Space
- Singular and GPI-Space

- Timings

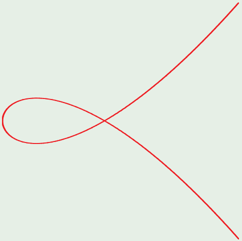
- More applications in
  - geometric invariant theory
  - tropical geometry.

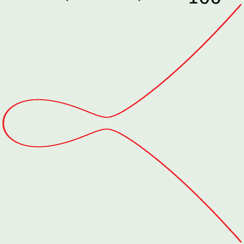
Determining smoothness of algebraic varieties is a key task when constructing new varieties (e.g. to study moduli spaces), since singularities can change important invariants:

Determining smoothness of algebraic varieties is a key task when constructing new varieties (e.g. to study moduli spaces), since singularities can change important invariants:

Example (Toy example)

$$y^2 - x^2(x+1) = 0 \qquad y^2 - x^2(x+1) + \frac{1}{100} = 0$$





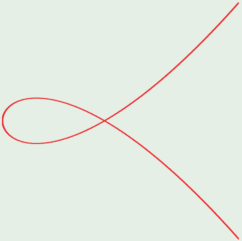
$$p_g(C) = 0 \qquad p_g(C) = 1$$

Smoothness of algebraic varieties

Determining smoothness of algebraic varieties is a key task when constructing new varieties (e.g. to study moduli spaces), since singularities can change important invariants:

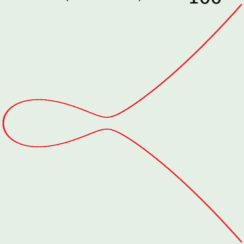
Example (Toy example)

$$y^2 - x^2(x+1) = 0$$



$$p_g(C) = 0$$

$$y^2 - x^2(x+1) + \frac{1}{100} = 0$$



$$p_g(C) = 1$$

according to the genus formula $p_g(C) = \frac{(d-1)(d-2)}{2} - \sum_{P \in C} \delta_P(C)$.

Theorem (Jacobian criterion)

Let K be perfect field, $S = K[x_1, \ldots, x_n]$, $I = \langle f_1, \ldots, f_r \rangle$

$$R = S/I$$

Smoothness of algebraic varieties

Theorem (Jacobian criterion)

Let K be perfect field, $S = K[x_1, \ldots, x_n]$, $I = \langle f_1, \ldots, f_r \rangle$

$$R = S/I$$

and $P \supset I$ a prime ideal of S,

Theorem (Jacobian criterion)

Let K be perfect field, $S = K[x_1, \ldots, x_n]$, $I = \langle f_1, \ldots, f_r \rangle$

$$R = S/I$$

and $P \supset I$ a prime ideal of S, let c be the codimension of $I_P \subset S_P$,

Smoothness of algebraic varieties

Theorem (Jacobian criterion)

Let K be perfect field, $S = K[x_1, \ldots, x_n]$, $I = \langle f_1, \ldots, f_r \rangle$

$$R = S/I$$

and $P \supset I$ a prime ideal of S, let c be the codimension of $I_P \subset S_P$, and let

$$\mathcal{J} = \begin{pmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_1/\partial x_n \\ \vdots & & \vdots \\ \partial f_r/\partial x_1 & \cdots & \partial f_r/\partial x_n \end{pmatrix}$$

be the **Jacobian matrix***.*

Smoothness of algebraic varieties

Theorem (Jacobian criterion)

Let K be perfect field, $S = K[x_1, \ldots, x_n]$, $I = \langle f_1, \ldots, f_r \rangle$

$$R = S/I$$

and $P \supset I$ a prime ideal of S, let c be the codimension of $I_P \subset S_P$, and let

$$\mathcal{J} = \begin{pmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_1/\partial x_n \\ \vdots & & \vdots \\ \partial f_r/\partial x_1 & \cdots & \partial f_r/\partial x_n \end{pmatrix}$$

be the **Jacobian matrix**. *Then* $\operatorname{rank}(\mathcal{J} \bmod P) \leq c$

Theorem (Jacobian criterion)

Let K be perfect field, $S = K[x_1, \ldots, x_n]$, $I = \langle f_1, \ldots, f_r \rangle$

$$R = S/I$$

and $P \supset I$ a prime ideal of S, let c be the codimension of $I_P \subset S_P$, and let

$$\mathcal{J} = \begin{pmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_1/\partial x_n \\ \vdots & & \vdots \\ \partial f_r/\partial x_1 & \cdots & \partial f_r/\partial x_n \end{pmatrix}$$

be the **Jacobian matrix***. Then* $\operatorname{rank}(\mathcal{J} \bmod P) \leq c$ *and R_P is a regular local ring iff*

$$\operatorname{rank}(\mathcal{J} \bmod P) = c.$$

Corollary

Let R be equidimensional,

Corollary

Let R be equidimensional,

$$\operatorname{Sing}(R) = \{P \in \operatorname{Spec}(R) \mid R_P \textit{ not regular}\}$$

its **singular locus**,

Corollary

Let R be equidimensional,

$$\operatorname{Sing}(R) = \{P \in \operatorname{Spec}(R) \mid R_P \textit{ not regular}\}$$

its **singular locus**, *and* $J = \langle \operatorname{minors}_c \mathcal{J} \rangle \subset R$ *the* **Jacobian ideal**.

Smoothness of algebraic varieties

Corollary

Let R be equidimensional,

$$\mathrm{Sing}(R) = \{P \in \mathrm{Spec}(R) \mid R_P \textit{ not regular}\}$$

its **singular locus**, *and $J = \langle \mathrm{minors}_c \, \mathcal{J} \rangle \subset R$ the* **Jacobian ideal**. *Then*

$$\mathrm{Sing}(R) = V(J)$$

Smoothness of algebraic varieties

Corollary

Let R be equidimensional,

$$\mathrm{Sing}(R) = \{P \in \mathrm{Spec}(R) \mid R_P \text{ not regular}\}$$

its **singular locus**, *and* $J = \langle \mathrm{minors}_c\, \mathcal{J} \rangle \subset R$ *the* **Jacobian ideal**. *Then*

$$\mathrm{Sing}(R) = V(J).$$

Example

$$I = \langle y^2 - x^2(x+1) \rangle \qquad I = \langle y^2 - x^2(x+1) + \tfrac{1}{100} \rangle$$

$$\mathrm{Sing}(R) = V(\langle x, y \rangle) \qquad \mathrm{Sing}(R) = V(\langle 1 \rangle) = \emptyset$$

- Jacobian criterion aims at computing the singular locus.

- Jacobian criterion aims at computing the singular locus.
- Expensive, especially if the size c of the minors is large.

Smoothness of algebraic varieties

- Jacobian criterion aims at computing the singular locus.
- Expensive, especially if the size c of the minors is large.
- Determine non/smoothness without computing the Jacobian ideal?

Smoothness of algebraic varieties

- Jacobian criterion aims at computing the singular locus.
- Expensive, especially if the size c of the minors is large.
- Determine non/smoothness without computing the Jacobian ideal?

Let K be a perfect field, $X = \operatorname{Spec}(A) \subset \mathbb{A}^n$, $A = K[x_1 \ldots, x_n]/I$ an affine scheme. If $p \in X$ the (vanishing-)order of $f \in A$ in p is

$$\operatorname{ord}_{\mathfrak{m}_p}(f) := \sup\{t \in \mathbb{N}_0 \mid f \in \mathfrak{m}_p^t\}$$

Smoothness of algebraic varieties

- Jacobian criterion aims at computing the singular locus.
- Expensive, especially if the size c of the minors is large.
- Determine non/smoothness without computing the Jacobian ideal?

Let K be a perfect field, $X = \mathrm{Spec}(A) \subset \mathbb{A}^n$, $A = K[x_1 \ldots, x_n]/I$ an affine scheme. If $p \in X$ the (vanishing-)order of $f \in A$ in p is

$$\mathrm{ord}_{\mathfrak{m}_p}(f) := \sup\{t \in \mathbb{N}_0 \mid f \in \mathfrak{m}_p^t\}$$

Lemma (Hironaka, 1964)

If $f_1, \ldots, f_s$ is a minimal standard basis of $I\mathcal{O}_{\mathbb{A}^n,p}$, sorted by increasing order, then

Smoothness of algebraic varieties

- Jacobian criterion aims at computing the singular locus.
- Expensive, especially if the size c of the minors is large.
- Determine non/smoothness without computing the Jacobian ideal?

Let K be a perfect field, $X = \mathrm{Spec}(A) \subset \mathbb{A}^n$, $A = K[x_1 \ldots, x_n]/I$ an affine scheme. If $p \in X$ the (vanishing-)order of $f \in A$ in p is

$$\mathrm{ord}_{\mathfrak{m}_p}(f) := \sup\{t \in \mathbb{N}_0 \mid f \in \mathfrak{m}_p^t\}$$

Lemma (Hironaka, 1964)

If $f_1, \ldots, f_s$ is a minimal standard basis of $I\mathcal{O}_{\mathbb{A}^n,p}$, sorted by increasing order, then

$$\nu^*(X,p) = (\mathrm{ord}_{\mathfrak{m}_p}(f_1), \ldots, \mathrm{ord}_{\mathfrak{m}_p}(f_s))$$

depends only on $\mathcal{O}_{X,p}$, and X is singular at p if and only if

$$\nu^*(X,p) >_{lex} (\underbrace{1, \ldots, 1}_{\mathrm{codim}(X)}).$$

Algorithmic smoothness criterion

Definition

For $W \subseteq \mathbb{A}^n$, $I \subset K[x_1, \ldots, x_n]/I_W$, and $p \in W$ define

Algorithmic smoothness criterion

Definition

For $W \subseteq \mathbb{A}^n$, $I \subset K[x_1, \ldots, x_n]/I_W$, and $p \in W$ define

$$\operatorname{ord}_p(I) := \sup\{t \in \mathbb{N} \mid I \subseteq \mathfrak{m}_{W,p}^t\}.$$

Algorithmic smoothness criterion

Definition

For $W \subseteq \mathbb{A}^n$, $I \subset K[x_1, \ldots, x_n]/I_W$, and $p \in W$ define

$$\operatorname{ord}_p(I) := \sup\{t \in \mathbb{N} \mid I \subseteq \mathfrak{m}_{W,p}^t\}.$$

Lemma

If W is a smooth complete intersection of codim s, $X = V(I)$ and $\operatorname{ord}_p(I) \geq 2$ then

$$\nu^*(X, p) = (\underbrace{1, \ldots, 1}_{s}, \geq 2, \ldots)$$

hence $X = V(I)$ is not smooth at p.

Algorithmic smoothness criterion

Definition

For $W \subseteq \mathbb{A}^n$, $I \subset K[x_1, \ldots, x_n]/I_W$, and $p \in W$ define

$$\mathrm{ord}_p(I) := \sup\{t \in \mathbb{N} \mid I \subseteq \mathfrak{m}_{W,p}^t\}.$$

Lemma

If W is a smooth complete intersection of codim s, $X = V(I)$ and $\mathrm{ord}_p(I) \geq 2$ then

$$\nu^*(X, p) = (\underbrace{1, \ldots, 1}_{s}, \geq 2, \ldots)$$

hence $X = V(I)$ is not smooth at p.

Lemma

If $\{p \in X \mid \mathrm{ord}_p(I) \geq 2\} = \emptyset$ then there is $f \in I$ defining locally in a Zariski neighborhood of p a smooth hypersurface $X \subset Z \subset W$.

Algorithmic smoothness criterion

Lemma

If M is a $s \times s$-submatrix of the Jacobian matrix of W with $\det(M) \neq 0$, *then the x_i not used for differentiation in M induce by translation a local system of parameters $X_{p,j}$ at every point of $p \in W \cap D(\det(M))$. Write $\partial f_i / \partial X_{p,j}$ for the partial derivatives defined in terms of the Cohen structure theorem isomorphism*

$$K[[y_1, \ldots, y_{n-s}]] \cong \widehat{\mathcal{O}_{W,p}}$$

The partials $\partial f_i / \partial X_{p,j}$ can be represented for all $p \in D(\det(M))$ by fixed elements $H_{i,j} \in \mathcal{O}_W(D(\det(M)))$, and we write

$$\partial f_i / \partial X_j := H_{i,j}$$

.

Lemma

If M is a $s \times s$-submatrix of the Jacobian matrix of W with $\det(M) \neq 0$, then the x_i not used for differentiation in M induce by translation a local system of parameters $X_{p,j}$ at every point of $p \in W \cap D(\det(M))$. Write $\partial f_i / \partial X_{p,j}$ for the partial derivatives defined in terms of the Cohen structure theorem isomorphism

$$K[[y_1, \ldots, y_{n-s}]] \cong \widehat{\mathcal{O}_{W,p}}$$

The partials $\partial f_i / \partial X_{p,j}$ can be represented for all $p \in D(\det(M))$ by fixed elements $H_{i,j} \in \mathcal{O}_W(D(\det(M)))$, and we write

$$\partial f_i / \partial X_j := H_{i,j}$$

Lemma

$\{p \in X \mid \mathrm{ord}_p(I) \geq 2\}$ *is defined in* $X \cap D(\det(M))$ *by*

$$I + \langle \partial f_i / \partial X_j \mid i, j \rangle$$

Algorithmic smoothness criterion

This allows us to iteratively describe an equidimensional $X \subset \mathbb{A}^n$ locally as a smooth complete intersection or recognize that X is not smooth:

This allows us to iteratively describe an equidimensional $X \subset \mathbb{A}^n$ locally as a smooth complete intersection or recognize that X is not smooth: Let $X = V(f_1, \ldots, f_r) \subset V(f_1 \ldots, f_s) = W \subset \mathbb{A}^n$ where W is a complete intersection, smooth in $D(q) = \{q \neq 0\}$.

Algorithmic smoothness criterion

This allows us to iteratively describe an equidimensional $X \subset \mathbb{A}^n$ locally as a smooth complete intersection or recognize that X is not smooth: Let $X = V(f_1, \ldots, f_r) \subset V(f_1 \ldots, f_s) = W \subset \mathbb{A}^n$ where W is a complete intersection, smooth in $D(q) = \{q \neq 0\}$.

- Find a set L of $s \times s$ submatrices M of $\mathcal{J}(W)$ with $\det(M) \neq 0$ and

$$\langle f_1, \ldots, f_r \rangle + \langle \det(M) \mid M \in L \rangle = \langle 1 \rangle$$

Algorithmic smoothness criterion

This allows us to iteratively describe an equidimensional $X \subset \mathbb{A}^n$ locally as a smooth complete intersection or recognize that X is not smooth: Let $X = V(f_1, \ldots, f_r) \subset V(f_1 \ldots, f_s) = W \subset \mathbb{A}^n$ where W is a complete intersection, smooth in $D(q) = \{q \neq 0\}$.

- Find a set L of $s \times s$ submatrices M of $\mathcal{J}(W)$ with $\det(M) \neq 0$ and

$$\langle f_1, \ldots, f_r \rangle + \langle \det(M) \mid M \in L \rangle = \langle 1 \rangle$$

  W.l.o.g. consider the first minor.

Algorithmic smoothness criterion

This allows us to iteratively describe an equidimensional $X \subset \mathbb{A}^n$ locally as a smooth complete intersection or recognize that X is not smooth: Let $X = V(f_1, \dots, f_r) \subset V(f_1 \dots, f_s) = W \subset \mathbb{A}^n$ where W is a complete intersection, smooth in $D(q) = \{q \neq 0\}$.

- Find a set L of $s \times s$ submatrices M of $\mathcal{J}(W)$ with $\det(M) \neq 0$ and

$$\langle f_1, \dots, f_r \rangle + \langle \det(M) \mid M \in L \rangle = \langle 1 \rangle$$

  W.l.o.g. consider the first minor.

- Find A such that $A \cdot M = \det(M) \cdot E_s$ and let

$$F := \begin{pmatrix} A & 0 \\ 0 & \det(M) \cdot E_{r-s} \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ \vdots \\ f_r \end{pmatrix}$$

Algorithmic smoothness criterion

- Then the locus of order ≥ 2 in $D(q)$ is empty iff

$$q \in \sqrt{\langle f_1, \ldots, f_r, \partial F_i / \partial X_j \mid i, j > s \rangle}$$

where the $\partial F_i / \partial X_j$ can be computed as the entries of the right lower block after the row reduction

$$\mathcal{J}(F) = \left(\begin{array}{ccc|c} \det(M) & & 0 & \\ & \ddots & & * \\ 0 & & \det(M) & \\ \hline & * & & * \end{array} \right) \mapsto \left(\begin{array}{ccc|c} \det(M) & & 0 & \\ & \ddots & & * \\ 0 & & \det(M) & \\ \hline & 0 & & * \end{array} \right)$$

Algorithmic smoothness criterion

- Then the locus of order ≥ 2 in $D(q)$ is empty iff

$$q \in \sqrt{\langle f_1, \ldots, f_r, \partial F_i / \partial X_j \mid i, j > s \rangle}$$

where the $\partial F_i / \partial X_j$ can be computed as the entries of the right lower block after the row reduction

$$\mathcal{J}(F) = \left(\begin{array}{ccc|c} \det(M) & & 0 & \\ & \ddots & & * \\ 0 & & \det(M) & \\ \hline & * & & * \end{array} \right) \mapsto \left(\begin{array}{ccc|c} \det(M) & & 0 & \\ & \ddots & & * \\ 0 & & \det(M) & \\ \hline & 0 & & * \end{array} \right)$$

- If so, consider a representation

$$q^m = \sum \alpha_{i,j} \cdot \partial F_i / \partial X_j \bmod \langle f_1, \ldots, f_r \rangle$$

Algorithmic smoothness criterion

- Then the locus of order ≥ 2 in $D(q)$ is empty iff

$$q \in \sqrt{\langle f_1, \ldots, f_r, \partial F_i/\partial X_j \mid i,j > s\rangle}$$

where the $\partial F_i/\partial X_j$ can be computed as the entries of the right lower block after the row reduction

$$\mathcal{J}(F) = \left(\begin{array}{ccc|c} \det(M) & & 0 & \\ & \ddots & & * \\ 0 & & \det(M) & \\ \hline & * & & * \end{array}\right) \mapsto \left(\begin{array}{ccc|c} \det(M) & & 0 & \\ & \ddots & & * \\ 0 & & \det(M) & \\ \hline & 0 & & * \end{array}\right)$$

- If so, consider a representation

$$q^m = \sum \alpha_{i,j} \cdot \partial F_i/\partial X_j \bmod \langle f_1, \ldots, f_r\rangle$$

- If $\alpha_{i,j} \neq 0$, on $D(q \cdot \partial F_i/\partial X_j)$ we can replace

$$(f_1, \ldots, f_s) \mapsto (f_1, \ldots, f_s, F_i)$$

defining smooth c.i. $W' \supset X$ with $\operatorname{codim}(W') = \operatorname{codim}(W) + 1$.

Algorithmic smoothness criterion

- Then the locus of order ≥ 2 in $D(q)$ is empty iff
$$q \in \sqrt{\langle f_1, \ldots, f_r, \partial F_i/\partial X_j \mid i,j > s\rangle}$$
where the $\partial F_i/\partial X_j$ can be computed as the entries of the right lower block after the row reduction
$$\mathcal{J}(F) = \left(\begin{array}{ccc|c} \det(M) & & 0 & \\ & \ddots & & * \\ 0 & & \det(M) & \\ \hline & * & & * \end{array}\right) \mapsto \left(\begin{array}{ccc|c} \det(M) & & 0 & \\ & \ddots & & * \\ 0 & & \det(M) & \\ \hline & 0 & & * \end{array}\right)$$
- If so, consider a representation
$$q^m = \sum \alpha_{i,j} \cdot \partial F_i/\partial X_j \bmod \langle f_1, \ldots, f_r\rangle$$
- If $\alpha_{i,j} \neq 0$, on $D(q \cdot \partial F_i/\partial X_j)$ we can replace
$$(f_1, \ldots, f_s) \mapsto (f_1, \ldots, f_s, F_i)$$
defining smooth c.i. $W' \supset X$ with $\operatorname{codim}(W') = \operatorname{codim}(W) + 1$.
- Iteratively obtain tree of charts.

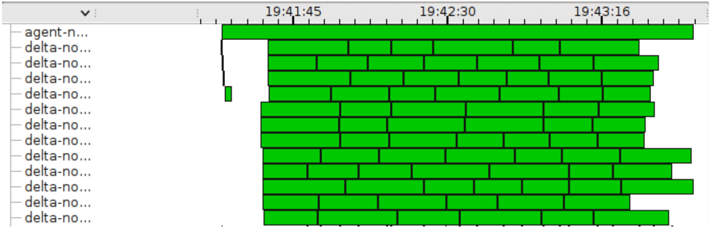
GPI-Space

- developed by Fraunhofer Institute for Industrial Mathematics ITWM

GPI-Space

- developed by Fraunhofer Institute for Industrial Mathematics ITWM

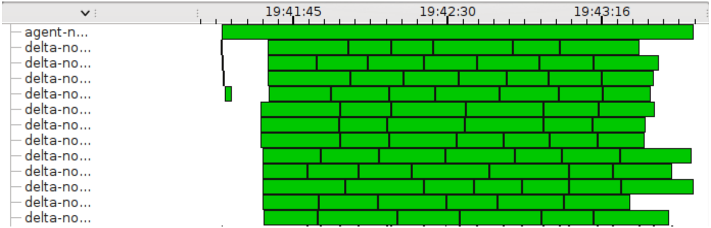
- task based workflow management system for massively parallel computations



GPI-Space

- developed by Fraunhofer Institute for Industrial Mathematics ITWM

- task based workflow management system for massively parallel computations



- based on idea of separating coordination and computation.

- Separate, specialized language for coordination layer (Petri nets).

Philosophy and Features of GPI-Space

- Separate, specialized language for coordination layer (Petri nets).
- Implementations and optimizations in the coordination layer and computational can be done by the respective experts.

- Separate, specialized language for coordination layer (Petri nets).
- Implementations and optimizations in the coordination layer and computational can be done by the respective experts.
- Complex coordination hidden from domain experts: automatic parallelization, cost optimized data transfers hiding latency, adaptation to dynamic changes in the computing environment.

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Virtual memory layer allows

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Virtual memory layer allows

- to scale computations beyond limitations of single machine.

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Virtual memory layer allows

- to scale computations beyond limitations of single machine.
- legacy applications to interoperate in an efficient way.

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Virtual memory layer allows

- to scale computations beyond limitations of single machine.
- legacy applications to interoperate in an efficient way.
- to switch between different types of memory without changing the implementation.

Structure and Implementation of GPI-Space

Three main components:

- Distributed, scalable, resilient **runtime system** for dynamic computational environments (from small to huge): manages computational resources and memory. Scheduler assigns activities to resources.

GPI-Space: Scheduler

Three main components:

- Distributed, scalable, resilient **runtime system** for huge dynamic environments: manages memory and computational resources. Scheduler assigns activities to resources w.r.t. both the needs of the current computations and the overall optimization goals.

Structure and Implementation of GPI-Space

Three main components:

- Distributed, scalable, resilient **runtime system** for huge dynamic environments: manages memory and computational resources. Scheduler assigns activities to resources w.r.t. both the needs of the current computations and the overall optimization goals.
- **Virtual memory manager**: allows algorithmic building blocks to communicate, share partial results. Communication managed by the runtime system rather than the domain applications.

Three main components:

- Distributed, scalable, resilient **runtime system** for huge dynamic environments: manages memory and computational resources. Scheduler assigns activities to resources w.r.t. both the needs of the current computations and the overall optimization goals.
- **Virtual memory manager**: allows algorithmic building blocks to communicate, share partial results. Communication managed by the runtime system rather than the domain applications.
- Petri net based **workflow engine**: manages the full application state and is responsible for automatic parallelization and dependency tracking.

Petri nets

Introduced by Carl Adam Petri (1926–2010) in 1962 to describe concurrent asynchronous systems (this is how real world physics works!). In fact he invented them already much earlier to remember chemical reactions in school.

Petri nets

Petri nets are a graphical way to model algorithms: Consist out of **places** and **transitions**. By a marking of places a state is described, if the conditions of a transition are satisfied, it changes the state.

Petri nets

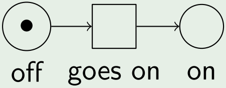
Petri nets are a graphical way to model algorithms: Consist out of **places** and **transitions**. By a marking of places a state is described, if the conditions of a transition are satisfied, it changes the state.

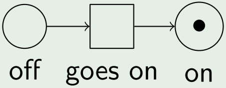
Example

Petri nets

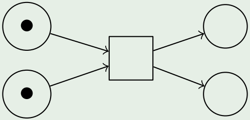
Petri nets are a graphical way to model algorithms: Consist out of **places** and **transitions**. By a marking of places a state is described, if the conditions of a transition are satisfied, it changes the state.

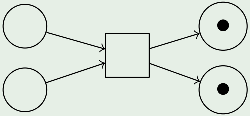
Example

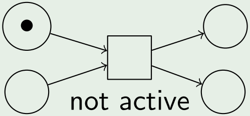




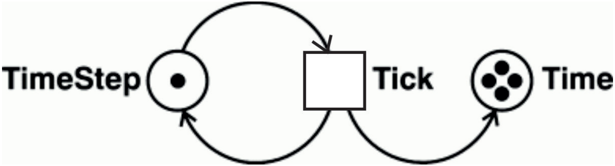
Example



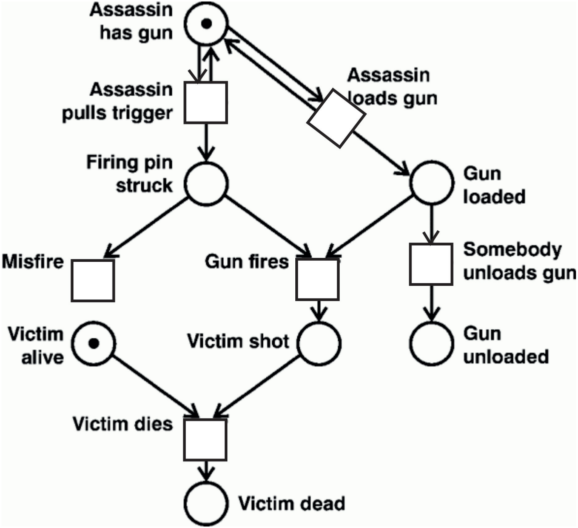




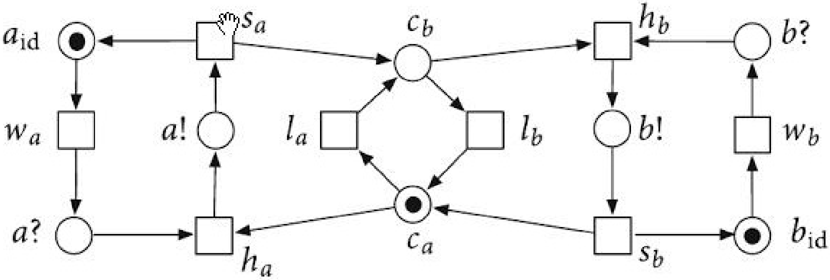
Clock at time $t = 4$:



Real world examples



Assigning a printer to two users:



Features of Petri nets

- Graphical, so accessible not only to experts in programming.

Features of Petri nets

- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.

Features of Petri nets

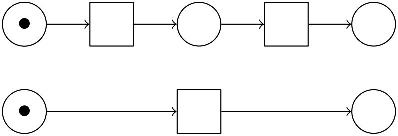
- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:

Features of Petri nets

- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies (we know that idea?), no global events.

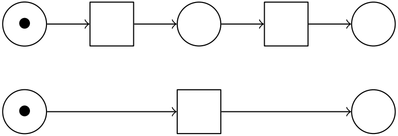
Features of Petri nets

- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies (we know that idea?), no global events.



Features of Petri nets

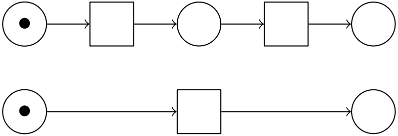
- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies (we know that idea?), no global events.



  - Reversible, can compute backwards (a key idea in physics), can recompute in case of a loss of a result.

Features of Petri nets

- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
    - Locality of dependencies (we know that idea?), no global events.



    - Reversible, can compute backwards (a key idea in physics), can recompute in case of a loss of a result.
    - Can add resources to running computations without any synchronization.

Features of Petri nets

Definition

A **Petri net** is a triple (P, T, F) with finite, disjoint sets **places** P and **transitions** T. The **flow function** $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ describes interaction of places and transitions. We say that p is a **predecessor** of t if $F(p, t) > 0$, and that p is a **sucessor** of t if $F(t, p) > 0$.

Petri nets

Definition

A **Petri net** is a triple (P, T, F) with finite, disjoint sets **places** P and **transitions** T. The **flow function** $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ describes interaction of places and transitions. We say that p is a **predecessor** of t if $F(p, t) > 0$, and that p is a **sucessor** of t if $F(t, p) > 0$.
The state of a Petri net is described by a so called **marking** $M : P \to \mathbb{N}$.
If $M(p) = k$, we say p **has** k **tokens under** M.

Definition

A **Petri net** is a triple (P, T, F) with finite, disjoint sets **places** P and **transitions** T. The **flow function** $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ describes interaction of places and transitions. We say that p is a **predecessor** of t if $F(p, t) > 0$, and that p is a **sucessor** of t if $F(t, p) > 0$.
The state of a Petri net is described by a so called **marking** $M : P \to \mathbb{N}$.
If $M(p) = k$, we say p **has** k **tokens under** M.
We say that a marking M **enables** a transition t if the places have enough tokens required by F, that is $\forall p \in P : M(p) \geq F(p, t)$.

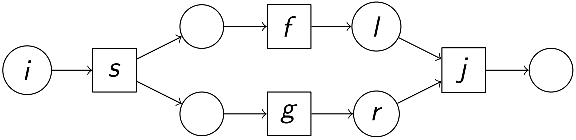
Petri nets

Definition

A **Petri net** is a triple (P, T, F) with finite, disjoint sets **places** P and **transitions** T. The **flow function** $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ describes interaction of places and transitions. We say that p is a **predecessor** of t if $F(p, t) > 0$, and that p is a **sucessor** of t if $F(t, p) > 0$.
The state of a Petri net is described by a so called **marking** $M : P \to \mathbb{N}$.
If $M(p) = k$, we say p **has** k **tokens under** M.
We say that a marking M **enables** a transition t if the places have enough tokens required by F, that is $\forall p \in P : M(p) \geq F(p, t)$.
A transition enabled by M can **fire** leading to the new marking
$M'(p) = M(p) - F(p, t) + F(t, p)$.

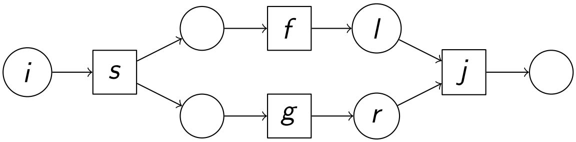
To fire t consumes $F(p, t)$ tokens from each predecessor p of t and produces $F(q, t)$ tokens on each sucessor q of t.
To execute a Petri net we randomly fire enabled transitions.

- Task parallelism:
  Transitions f and g can fire in parallel:

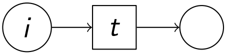


Parallel computations

- Task parallelism:
  Transitions f and g can fire in parallel:



- Data parallelism:
  If i holds multiple tokens, t can fire in parallel:



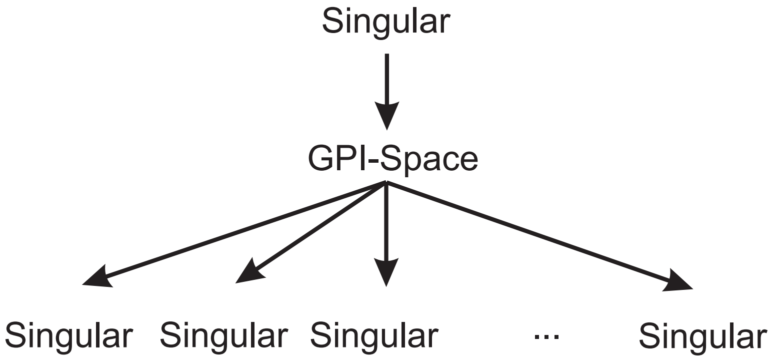
Note:

- Real world transitions take time.
- Tokens can be complex data structures.

Integration by Lukas Ristau:

Integration by Lukas Ristau:

- SINGULAR calls GPI-Space.
- GPI-Space uses LIBSINGULAR on the workers.



Modelling the smoothness test

Petri net implementation of the smoothness test (L. Ristau) in short:

- Run all charts occuring in the descent in codimension in parallel.

Petri net implementation of the smoothness test (L. Ristau) in short:

- Run all charts occuring in the descent in codimension in parallel.
- Charts leading to non/smoothness quickly win.

Modelling the smoothness test

Petri net implementation of the smoothness test (L. Ristau) in short:

- Run all charts occuring in the descent in codimension in parallel.
- Charts leading to non/smoothness quickly win.
- If one chart shows not smooth, return **false**.

Modelling the smoothness test

Petri net implementation of the smoothness test (L. Ristau) in short:

- Run all charts occuring in the descent in codimension in parallel.
- Charts leading to non/smoothness quickly win.
- If one chart shows not smooth, return **false**.
- Once X is completely covered with charts $U_i \subset W$ s.t. $X \cap U_i$ is a smooth complete intersection, return **true**.

Application to Algorithms in Algebraic Geometry

- Key concept in algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by charts.

- Key concept in algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by charts.
  - Global properties are related to local ones in the individual charts.

- Key concept in algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by charts.
  - Global properties are related to local ones in the individual charts.
- Use this natural parallel structure algorithmically?

Application to Algorithms in Algebraic Geometry

- Key concept in algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by charts.
  - Global properties are related to local ones in the individual charts.
- Use this natural parallel structure algorithmically?
- Computational basis of most algorithms is Buchberger's Algorithm.

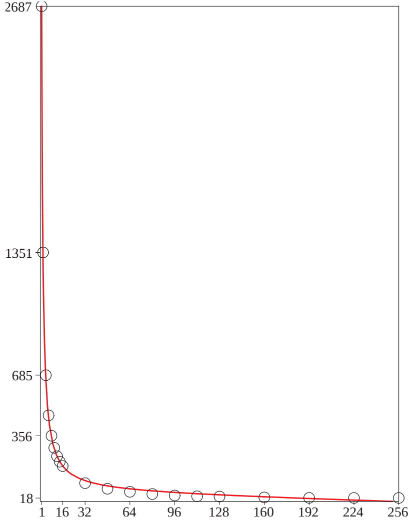
Application to Algorithms in Algebraic Geometry

- Key concept in algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by charts.
  - Global properties are related to local ones in the individual charts.
- Use this natural parallel structure algorithmically?
- Computational basis of most algorithms is Buchberger's Algorithm.
- Buchberger's Algorithm has doubly exponential worst case complexity [Mayr, Meyer 1982], much faster in many practical examples of interest $\rightarrow$ unpredictable for parallelization (can parallelize *individual* computations via modular and linear algebra methods).
- $\rightarrow$ Single chart may dominate the run-time.
- Solution: Model algorithm in a parallel way s.t. it automatically finds a good cover.

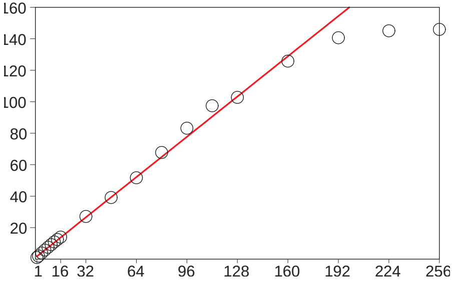
Campedelli surface of codim 5 with algebraic fundamental group $\mathbb{Z}/6$:

Timings for Campedelli surface

Campedelli surface of codim 5 with algebraic fundamental group $\mathbb{Z}/6$:



sec/cores



1/cores

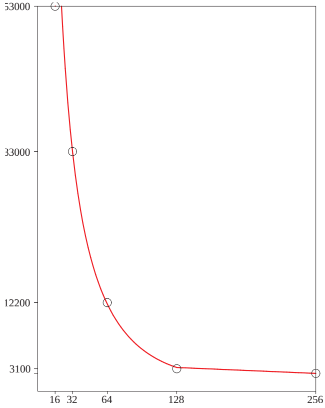
Jacobian criterion takes 461sec.

Timings for Godeaux surface of codim 11

General type surface by Frank-Olaf Schreyer, Isabel Stenger.

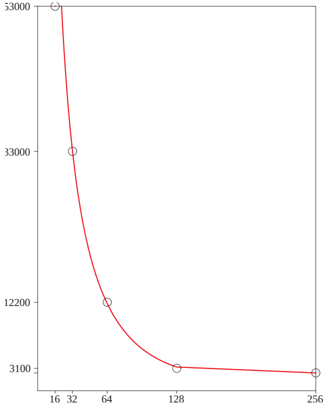
Timings for Godeaux surface of codim 11

General type surface by Frank-Olaf Schreyer, Isabel Stenger.
Superlinear speedup for small number of cores.



Timings for Godeaux surface of codim 11

General type surface by Frank-Olaf Schreyer, Isabel Stenger.
Superlinear speedup for small number of cores.



Jacobian criterion does not finish.

Similar approach of choice of good cover for algorithmic resolution of singularities.

Similar approach of choice of good cover for algorithmic resolution of singularities.

General framework applicable to many more algorithmic problems in computer algebra.

- Computation of integration-by-parts identities for Feynman integrals

More Algorithms in Computer Algebra

Similar approach of choice of good cover for algorithmic resolution of singularities.

General framework applicable to many more algorithmic problems in computer algebra.

- Computation of integration-by-parts identities for Feynman integrals
- Computation of generating functions for Hurwitz numbers via Feynman integrals to test quasimodularity

Similar approach of choice of good cover for algorithmic resolution of singularities.

General framework applicable to many more algorithmic problems in computer algebra.

- Computation of integration-by-parts identities for Feynman integrals
- Computation of generating functions for Hurwitz numbers via Feynman integrals to test quasimodularity
- Computation of
  - GIT-fans,
  - Gröbner fans, and
  - tropical varieties

  with symmetry via a parallel fan traversal.

Similar approach of choice of good cover for algorithmic resolution of singularities.

General framework applicable to many more algorithmic problems in computer algebra.

- Computation of integration-by-parts identities for Feynman integrals
- Computation of generating functions for Hurwitz numbers via Feynman integrals to test quasimodularity
- Computation of
  - GIT-fans,
  - Gröbner fans, and
  - tropical varieties

  with symmetry via a parallel fan traversal.
- Computation of Gröbner bases and syzygies.

More Algorithms in Computer Algebra

Similar approach of choice of good cover for algorithmic resolution of singularities.

General framework applicable to many more algorithmic problems in computer algebra.

- Computation of integration-by-parts identities for Feynman integrals
- Computation of generating functions for Hurwitz numbers via Feynman integrals to test quasimodularity
- Computation of
  - GIT-fans,
  - Gröbner fans, and
  - tropical varieties

  with symmetry via a parallel fan traversal.
- Computation of Gröbner bases and syzygies.
- Do you have more ideas?

Quotients

Quotients of algebraic varieties by algebraic groups play an important role in constructing moduli spaces.

Quotients

Quotients of algebraic varieties by algebraic groups play an important role in constructing moduli spaces.

Example

$$\mathbb{C}^* \times \mathbb{C}^2 \to \mathbb{C}^2, \qquad t \cdot (x, y) = (tx, ty)$$

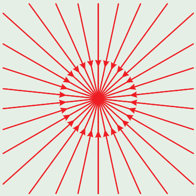
Quotients

Quotients of algebraic varieties by algebraic groups play an important role in constructing moduli spaces.

Example

$$\mathbb{C}^* \times \mathbb{C}^2 \to \mathbb{C}^2, \qquad t \cdot (x, y) = (tx, ty)$$

$$U = \mathbb{C}^2$$



$$U//\mathbb{C}^* = \{pt\}$$

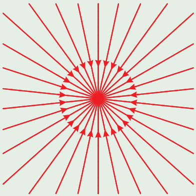
Quotients

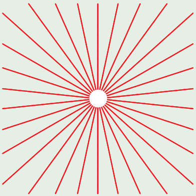
Quotients of algebraic varieties by algebraic groups play an important role in constructing moduli spaces.

Example

$$\mathbb{C}^* \times \mathbb{C}^2 \to \mathbb{C}^2, \qquad t \cdot (x, y) = (tx, ty)$$

$$U = \mathbb{C}^2 \qquad\qquad U = \mathbb{C}^2 \backslash \{0\}$$





$$U//\mathbb{C}^* = \{pt\} \qquad\qquad U//\mathbb{C}^* = \mathbb{P}^1$$

GIT-Fan

For torus actions on affine varieties $V(\mathfrak{a})$, classify all possible quotients (choices of open sets) in terms of a polyhedral fan, the **GIT-fan**.

For torus actions on affine varieties $V(\mathfrak{a})$, classify all possible quotients (choices of open sets) in terms of a polyhedral fan, the **GIT-fan**.

Example

For $\mathbb{C}^* \times \mathbb{C}^2 \to \mathbb{C}^2, \quad t \cdot (x, y) = (tx, ty), \quad Q = (1, 1), \quad \mathfrak{a} = 0$

$$U_1 = \mathbb{C}^2$$
$$U_2 = \mathbb{C}^2 \backslash \{0\}$$

$\Lambda(\mathfrak{a}, Q) =$ 

Symmetries

Example

$$\mathfrak{a} = \langle T_1 T_3 - T_2 T_4 \rangle \subset \mathbb{K}[T_1, \ldots, T_4] \qquad \deg(T_j) = q_j$$

$(\mathbb{C}^*)^2$-action on $V(\mathfrak{a})$ given by

$$Q = (q_1, \ldots, q_4) = \begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \end{pmatrix}$$

Symmetries

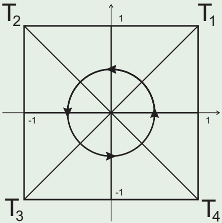
Example

$$\mathfrak{a} = \langle T_1 T_3 - T_2 T_4 \rangle \subset \mathbb{K}[T_1, \ldots, T_4] \qquad \deg(T_j) = q_j$$

$(\mathbb{C}^*)^2$-action on $V(\mathfrak{a})$ given by

$$Q = (q_1, \ldots, q_4) = \begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \end{pmatrix}$$

$$G = D_4 = \langle (1,2)(3,4), (1,2,3,4) \rangle \subset S_4$$



Symmetric GIT-Algorithm

Algorithm to compute GIT-fans with symmetries (B., Keicher, Ren, 2016) via a fan traversal, combining Gröbner bases with computations in polyhedral geometry and group theory.

Symmetric GIT-Algorithm

Algorithm to compute GIT-fans with symmetries (B., Keicher, Ren, 2016) via a fan traversal, combining Gröbner bases with computations in polyhedral geometry and group theory.

- Each GIT-cone is an intersection of orbit cones.

Algorithm to compute GIT-fans with symmetries (B., Keicher, Ren, 2016) via a fan traversal, combining Gröbner bases with computations in polyhedral geometry and group theory.

- Each GIT-cone is an intersection of orbit cones.
- Determine all orbit cones via monomial containment tests.

Symmetric GIT-Algorithm

Algorithm to compute GIT-fans with symmetries (B., Keicher, Ren, 2016) via a fan traversal, combining Gröbner bases with computations in polyhedral geometry and group theory.

- Each GIT-cone is an intersection of orbit cones.
- Determine all orbit cones via monomial containment tests.
- Traverse fan by passing through codim 1 faces to neighbours.

Symmetric GIT-Algorithm

Algorithm to compute GIT-fans with symmetries (B., Keicher, Ren, 2016) via a fan traversal, combining Gröbner bases with computations in polyhedral geometry and group theory.

- Each GIT-cone is an intersection of orbit cones.
- Determine all orbit cones via monomial containment tests.
- Traverse fan by passing through codim 1 faces to neighbours.
- Hash GIT-cones via the binary vector encoding which orbit cones occur in the corresponding intersection. Hash interacts well with symmetry group action.
- Compute in each orbit only a single representative.

Cox ring of the moduli space of stable genus zero curves with 6 marked points $\overline{M}_{0,6}$ is $\mathbb{Z}^{16}$-graded, has 40 generators (Castravet, 2009),

GIT-fans of torus actions on affine varieties

Cox ring of the moduli space of stable genus zero curves with 6 marked points $\overline{M}_{0,6}$ is $\mathbb{Z}^{16}$-graded, has 40 generators (Castravet, 2009), 225 relations (Bernal Guillen, Maclagan, 2012), Cox ring of the moduli space of stable genus zero curves with 6 marked points $\overline{M}_{0,6}$ is $\mathbb{Z}^{16}$-graded, has 40 generators (Castravet, 2009), 225 relations (Bernal Guillen, Maclagan, 2012), and a natural S_6-action.

GIT-fans of torus actions on affine varieties

Cox ring of the moduli space of stable genus zero curves with 6 marked points $\overline{M}_{0,6}$ is $\mathbb{Z}^{16}$-graded, has 40 generators (Castravet, 2009), 225 relations (Bernal Guillen, Maclagan, 2012), and a natural S_6-action.

Example

The GIT-fan decomposition of the moving cone $\mathrm{Mov}(\overline{M}_{0,6})$ classifies all small modifications (rational maps which are isomorphisms on open subsets which have a complement of codimension ≥ 2).
The moving cone $\mathrm{Mov}(\overline{M}_{0,6})$ has

$$176\,512\,225$$

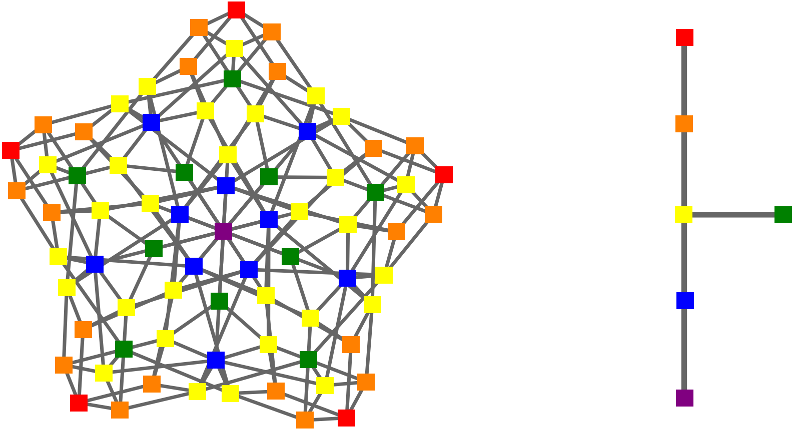
GIT-cones of maximal dimension 16, which decompose into

$$249\,605$$

orbits under the S_6-action. The cone with orbit length one is the semiample cone (dual of Mori cone).

Smaller example: GIT-fan for $\mathbb{G}(2,5)$

Adjacency graph of the maximal-dimensional GIT-cones and their orbits:

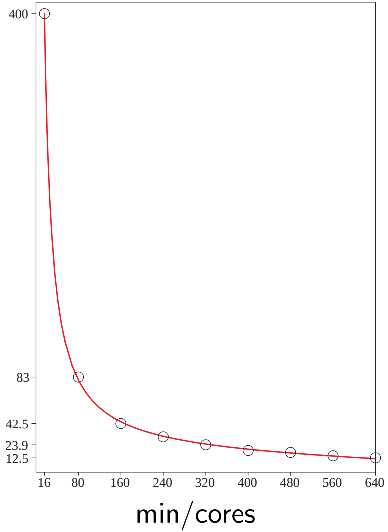


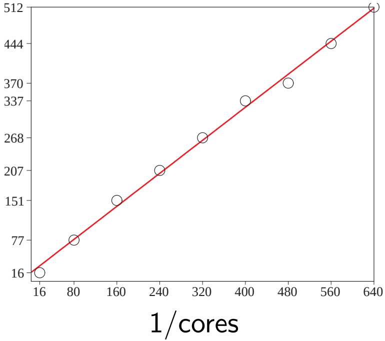
Timings and Scaling in GPI-Space for $\overline{M}_{0,6}$

Using the SINGULAR task model with 1 core 16 days, 16 cores 1 day.

Timings and Scaling in GPI-Space for $\overline{M}_{0,6}$

Using the SINGULAR task model with 1 core 16 days, 16 cores 1 day. Symmetric GIT-fan algorithm implemented by Christian Reinbold:





- Algorithm to compute tropical links by obtaining valuations via Puiseux expansions (Tommy Hofmann, Yue Ren, 2016).

Computing tropical varieties with symmetry

- Algorithm to compute tropical links by obtaining valuations via Puiseux expansions (Tommy Hofmann, Yue Ren, 2016).
- Newton-Puiseux implementation by Santiago Laplagne.

Computing tropical varieties with symmetry

- Algorithm to compute tropical links by obtaining valuations via Puiseux expansions (Tommy Hofmann, Yue Ren, 2016).
- Newton-Puiseux implementation by Santiago Laplagne.
- Use fan traversal by Christian Reinbold.

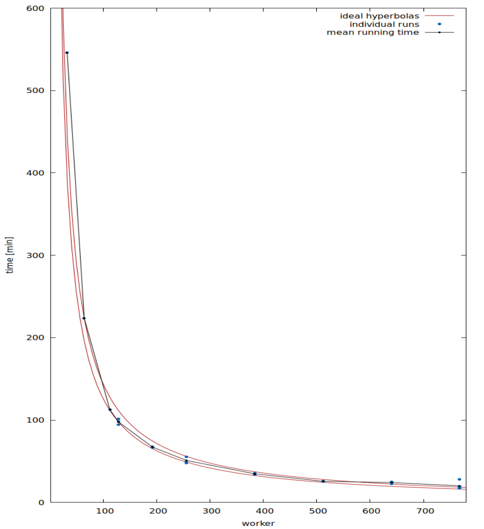
Computing tropical varieties with symmetry

- Algorithm to compute tropical links by obtaining valuations via Puiseux expansions (Tommy Hofmann, Yue Ren, 2016).
- Newton-Puiseux implementation by Santiago Laplagne.
- Use fan traversal by Christian Reinbold.
- Implementation of a parallel and symmetric algorithm for computing tropical varieties by Dominik Bendle.

Computing tropical varieties with symmetry

- Algorithm to compute tropical links by obtaining valuations via Puiseux expansions (Tommy Hofmann, Yue Ren, 2016).
- Newton-Puiseux implementation by Santiago Laplagne.
- Use fan traversal by Christian Reinbold.
- Implementation of a parallel and symmetric algorithm for computing tropical varieties by Dominik Bendle.

Example (Tropicalization of $\mathbb{G}(3,8)$)



References

H. Hironaka. *Resolution of singularities of an algebraic variety over a field of characteristic zero. I, II.* Ann. of Math. (1964).

J. Boehm and A. Frühbis-Krüger. *A smoothness test for higher codimension*. J. Symb. Comput. (2017).

F.-J. Pfreundt and M. Rahn: *GPI-space*, Fraunhofer ITWM Kaiserslautern, http://www.gpi-space.de/.

I. V. Dolgachev and Y. Hu. *Variation of geometric invariant theory quotients*. Publ. Math., Inst. Hautes Etud. Sci. (1998).

J. Boehm, S. Keicher, Y. Ren. *Computing GIT-fans with symmetry and the Mori chamber decomposition of* $\overline{M}_{0,6}$, arXiv:1603.09241 (2016).

T. Hofmann, Y. Ren. Computing tropical points and tropical links. arXiv:1611.02878 (2016).