On Parallelizing a Software Library for Algebraic Geometry

Wolfgang Schreiner, Christian Mittermaier, Franz Winkler
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, A-4040 Linz, Austria

FirstName.LastName@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at

Algebraic Geometry

Distributed Maple

Parallel Algebraic Geometry

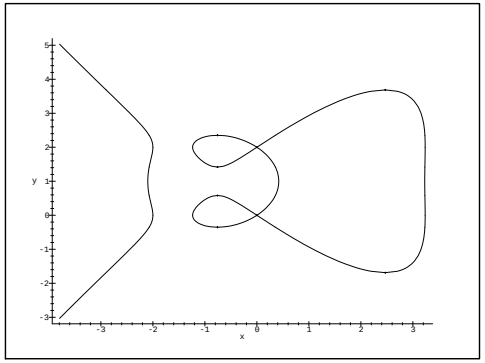
Algebraic Geometry

Algebraic Geometry

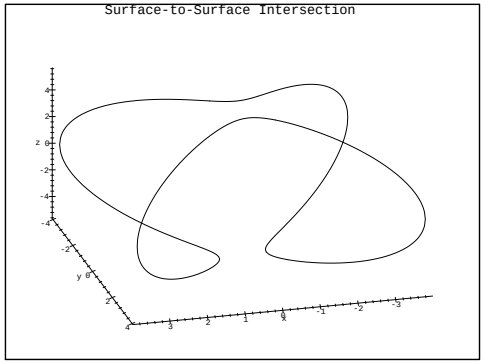
- Problem Area
  - Computing with and reasoning about geometric objects.
  - Representation by *algebraic sets*.
  - Manipulation by computer algebra methods.
- Software: CASA
  - *Computer Algebra Software for Constructive Algebraic Geometry*.
  - Implementation in the computer algebra system *Maple*.
  - Franz Winkler et al., 1990–
- One Application: Scientific Visualization.
  - Plane curves.
  - Surface-to-surface intersections.

Scientific Visualization

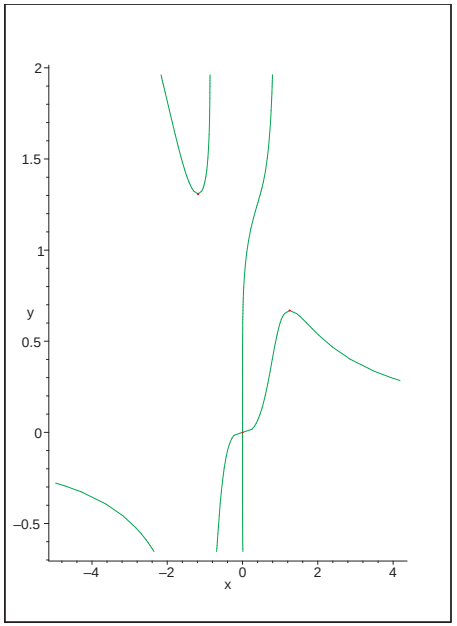
$$y^4 - 96a^2y^2 + 100a^2x^2 - x^4$$

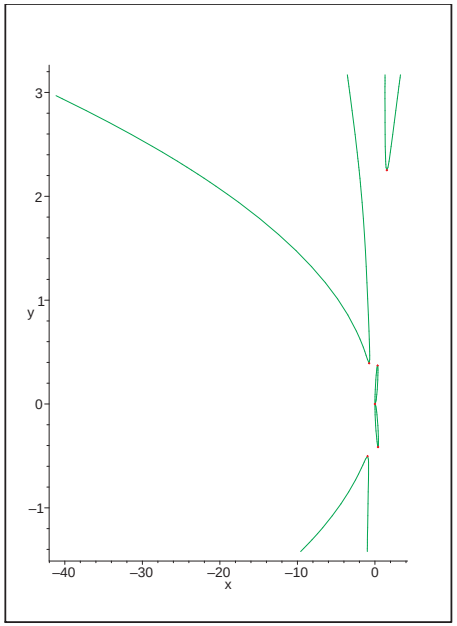


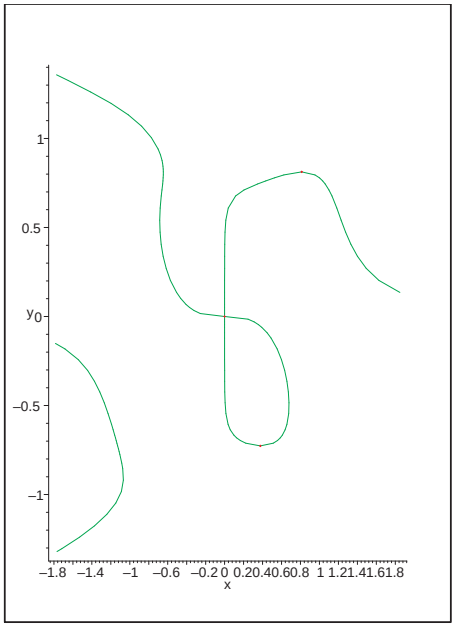
$$y^4 - 384y^2 + 400x^2 - x^4 + z^3,$$
$$x^2 + y^2 + z^2 - 30$$



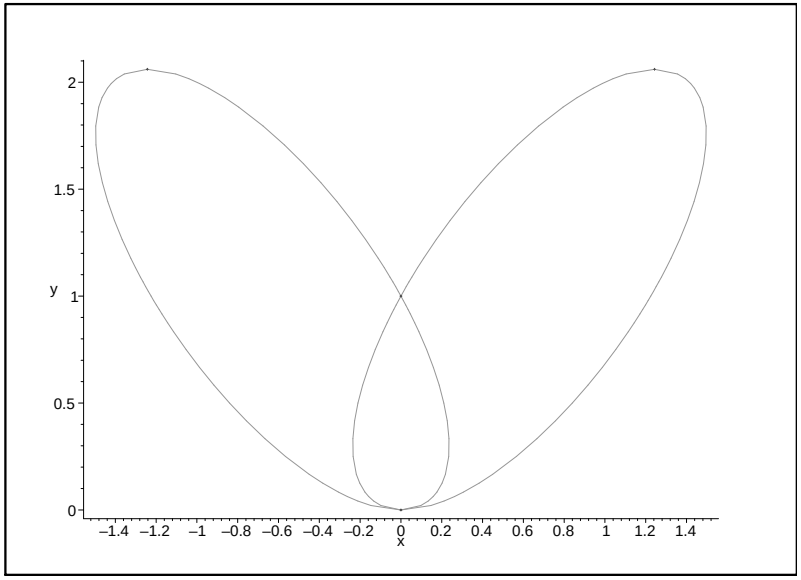
High Degree Curves



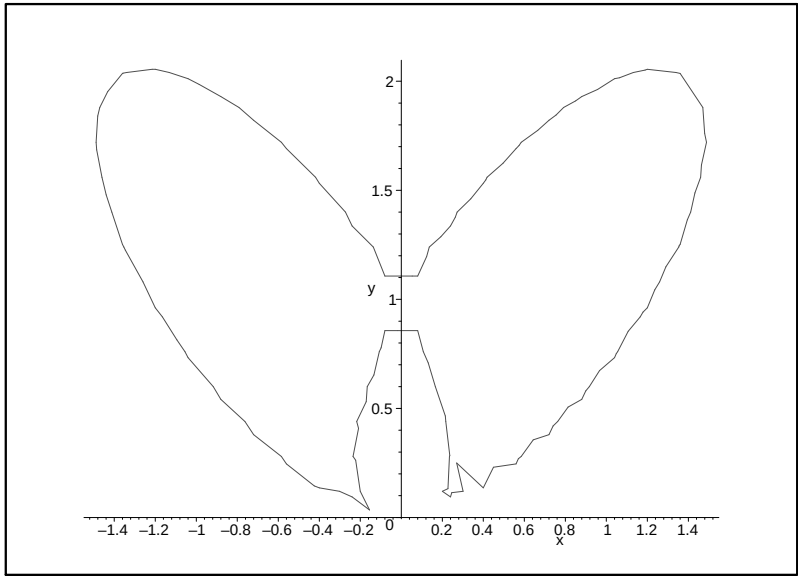




Symbolic versus Numeric

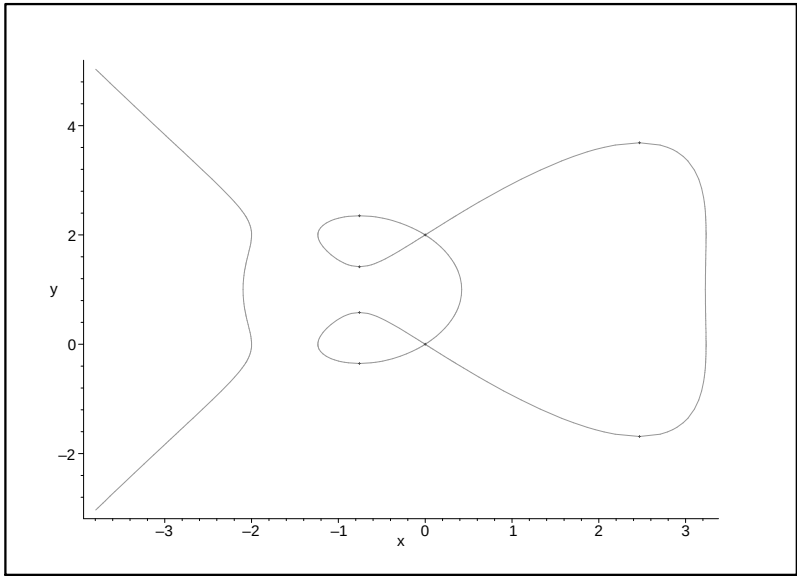


CASA `pacPlot`

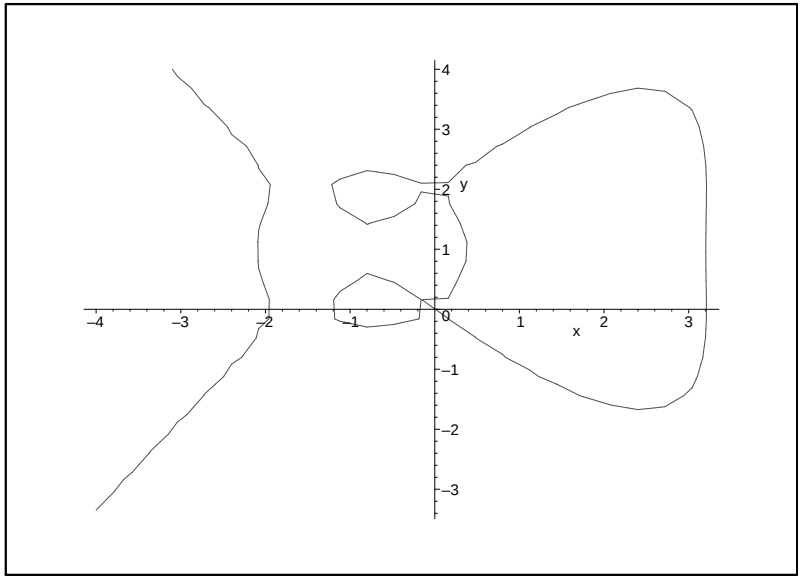


Maple `implicitPlot`

Symbolic versus Numeric



CASA `pacPlot`



Maple `implicitPlot`

Scientific Visualization

- Numerical methods: fast but unreliable.
  - Qualitatively wrong results possible.
  - E.g: missing *critical points*.
- Symbolic methods: reliable but slow.
  - Exponential in degree of polynomial.
  - Up to degree 5 or so.
- Hybrid symbolic-numerical methods: reliable and faster.
  - Still exponential complexity.
  - Up to degree 10 or so.

Many algorithmic improvements during the last 10 years.

Hybrid Algorithm

CASA `pacPlot`: plotting of plain algebraic curves (Tran-Quoc Nam, 1995)

Input: $p \in \mathbb{Q}[x, y]$

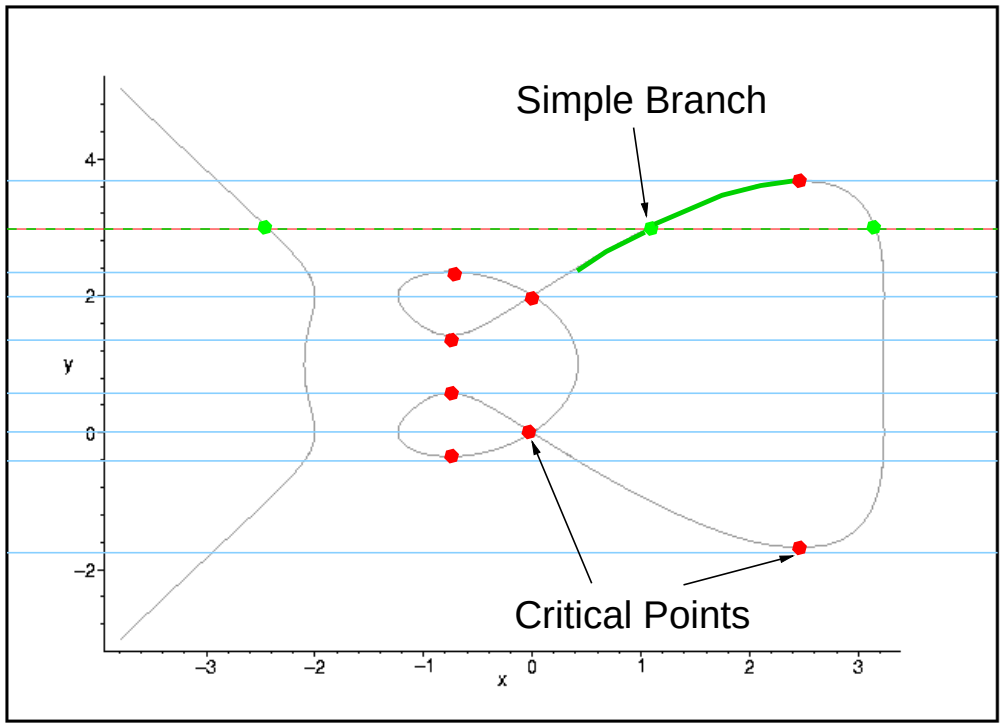
The representation of an algebraic curve in the plane.

Output: a *reliable* plot of p

All relevant information is preserved:

- All critical points are plotted.
- All simple branches are correctly plotted.

Hybrid Algorithm



Hybrid Algorithm

1. Compute all critical points in y-direction
   $\Rightarrow$ set of points (x_i, y_i)

   Extremal points in y-direction and singular points; number of intersections with a vertical line changes (solutions emerge or disappear).

2. Sort critical points by y-coordinate
   $\Rightarrow$ set of intervals $[y_i, y_{i+1}]$

   Each interval has a constant number of "simple" curve paths.

3. Intersect p with each horizontal line at $m_i := (y_i + y_{i+1})/2$
   $\Rightarrow$ set of points (x_i, m_i)

   Each point determines a simple curve branch.

4. (Numerical) apply *Newton Iteration*.

   Trace each simple path from its initial point in both directions towards the extremal points.

Computing the Critical Points

(Finitely many) solutions of system

$$\begin{aligned} p(x,y) &= 0 \\ \frac{\partial p}{\partial x}(x,y) &= 0 \end{aligned}$$

1. Compute $\frac{\partial p}{\partial x}(x,y)$.
2. Triangulize system $\Rightarrow$ set of $(p_i(y), q_i(x,y))$.

   Same roots as original system.
3. Solve each $(p_i(y), q_i(x,y)) \Rightarrow$ set of intervals $((x'_k, x''_k), (y'_k, y''_k))$.
   (a) Factorize $p_i(y) \in \mathbb{Q}[y] \Rightarrow$ set of $f_{ij}(y)$.
   (b) Solve each $(f_{ij}(y), q_i(x,y))$.
4. Take $(\frac{x'_k + x''_k}{2}, \frac{y'_k + y''_k}{2}) \Rightarrow$ set of points (x_k, y_k).

Solving $(p(y), q(x, y))$

1. Isolate the real roots of $p(y) \Rightarrow$ set of $(y_i', y_i'') \in \mathbb{Q} \times \mathbb{Q}$.

   Interval (y_i', y_i'') contains exactly one root $y \in \mathbb{R}$ of $p(y)$.

2. Make $q(x, \alpha)$ squarefree for every root α of $p(y)$.

   Computation in algebraic extension field $\mathbb{Q}(\alpha) := \mathbb{Q}[y]/p$.

   (a) $q' :=$ greatest squarefree divisor of $q(x, \alpha) \in \mathbb{Q}(\alpha)[x]$.

   (b) $q'' := q'[\alpha \leftarrow y] \in \mathbb{Q}[x, y]$.

3. Find all candidates for isolating x-intervals of the roots.

   (a) Compute the resultant $R(x)$ of $(p(y), q''(x, y))$.

   (b) Isolate the real roots of $R(x) \Rightarrow$ set of $(c_k', c_k'') \in \mathbb{Q} \times \mathbb{Q}$.

4. Find the x-intervals for each (y_i', y_i''), $\Rightarrow$ set of $((y_i', y_i''), (x_{ij}', x_{ij}''))$

   Select the isolating intervals for the real roots of $q' \in \mathbb{Q}(\alpha)[x]$ from the candidates (c_k', c_k'').

Selecting the x-intervals of $q \in \mathbb{Q}(\alpha)[x]$

(c'_k, c''_k) is an isolating x-interval for the real root whose y-coordinate is isolated by (y'_i, y''_i) if and only if

1. $q'(c'_k) = 0$ or
2. $q'(c''_k) = 0$ or
3. $\text{sign}(q'(c'_k)) \neq \text{sign}(q'(c''_k))$.

Computations in $\mathbb{Q}(\alpha)[x]$ where α is determined by p and (y'_i, y''_i).

Critical Point Computation

P := criticalPoints(A):
  $P := \emptyset$
  $S :=$ triangulize(A)
  for $(p, q) \in S$ **do**
    $Y :=$ realroot(p)
    $q' :=$ squarefree$_\alpha(q, p)$
    $q'' :=$ substitute$_\alpha(q')$
    $R :=$ resultant(p, q'')
    $C :=$ realroot(R)
    for $y \in Y$ **do**
      for $c \in C$ **do**
        if check$_\alpha(c, y, p, q')$ **then** $P := P \cup \{(y, c)\}$
      end
    end
  end

Execution Times of Hybrid Algorithm

1. Compute critical points.
2. Sort critical points and determine interval lines.
3. Intersect curve with lines.
4. Apply Newton Iteration.

- Steps 2-4: neglible.
- Step 1: $\geq$ 95% of computation time.
  - Resultants in $\mathbb{Q}[x_1, x_2]$.
  - Real root isolation in $\mathbb{Q}[x]$.
  - Gcd in $\mathbb{Q}(\alpha)[x]$ (squarefree$_\alpha$, check$_\alpha$).

$\Rightarrow$ *Parallelization and/or algorithmic improvements*.

Distributed Maple

Parallelization of CASA

- Large existing code base in Maple.
  - Porting to other system unrealistic.
- Maple is *sequential*.
  - No multi-threading.
  - No message passing.
- Maple is *closed*.
  - No access to internal kernel structures.
- Only external interface: input/output.
  - Linearization of internal data structures.
  - Reading/writing from/to files or pipes.

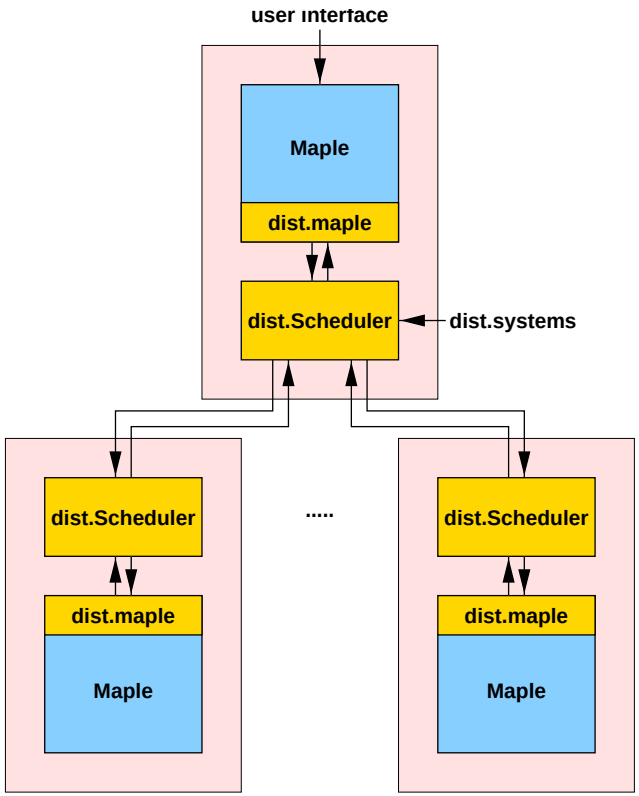
Constraints for integration into a parallel environment.

Distributed Maple

Wolfgang Schreiner, 1998.

- Coordination Layer.
  - Distributed interaction and task scheduling.
  - Java program *dist.Scheduler*.
- Interface Layer.
  - Connecting Maple kernel to scheduler.
  - Maple package *dist.maple*.
- Computing Layer.
  - Maple kernels as computational engines.

No modification of Maple!



Example

```
    |\^/|     Maple V Release 4 (Johannes Kepler University Linz)
._|\|   |/|_. Copyright (c) 1981-1996 by Waterloo Maple Inc. All rights
 \  MAPLE  /  reserved. Maple and Maple V are registered trademarks of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> read `dist.maple`;
Distributed Maple V1.0 (c) 1998 Wolfgang Schreiner (RISC-Linz)
See http://www.risc.uni-linz.ac.at/software/distmaple
> dist[initialize]([[deneb,solaris], [iris,irix]]);
connecting deneb...
connecting iris...
                                    okay
> t1 := dist[start](int, x^n, x):
> t2 := dist[start](int, x^n, n):
> dist[wait](t1) + dist[wait](t2);
                                (n + 1)     n
                               x           x
                               -------- + -----
                                n + 1     ln(x)
```

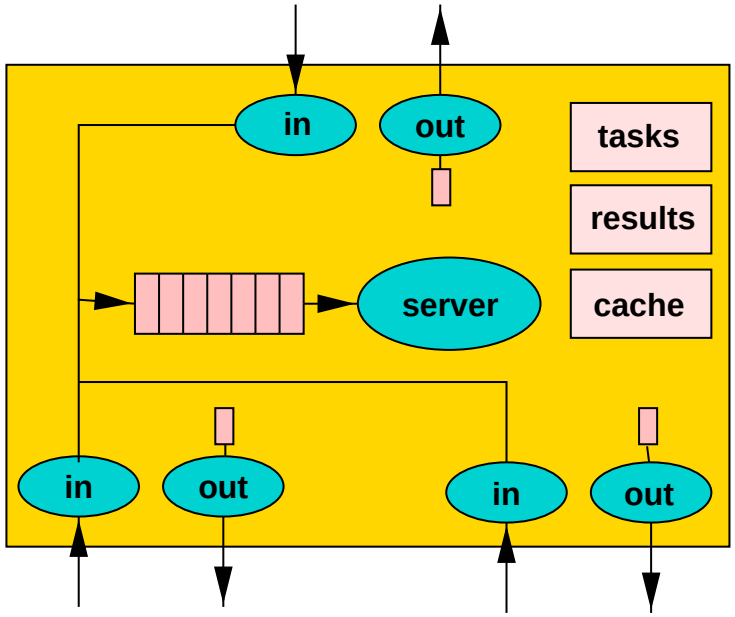
Programming Model

- Functional Parallelism
  - Task Creation: `t = dist[start](f, ...)`.
  - Synchronization: `r = dist[wait](t)`.
- Non-Determinism
  - Selection: `(i, r) = dist[select](tlist)`.
  - Abortion: `dist[delete](t)`.
- Shared Data Objects
  - Create empty object: `d = dist[data]()`.
  - Write data: `dist[put](d, v)`.
  - Wait for data: `v = dist[get](d)`.

Derived from functional/logic/dataflow models.

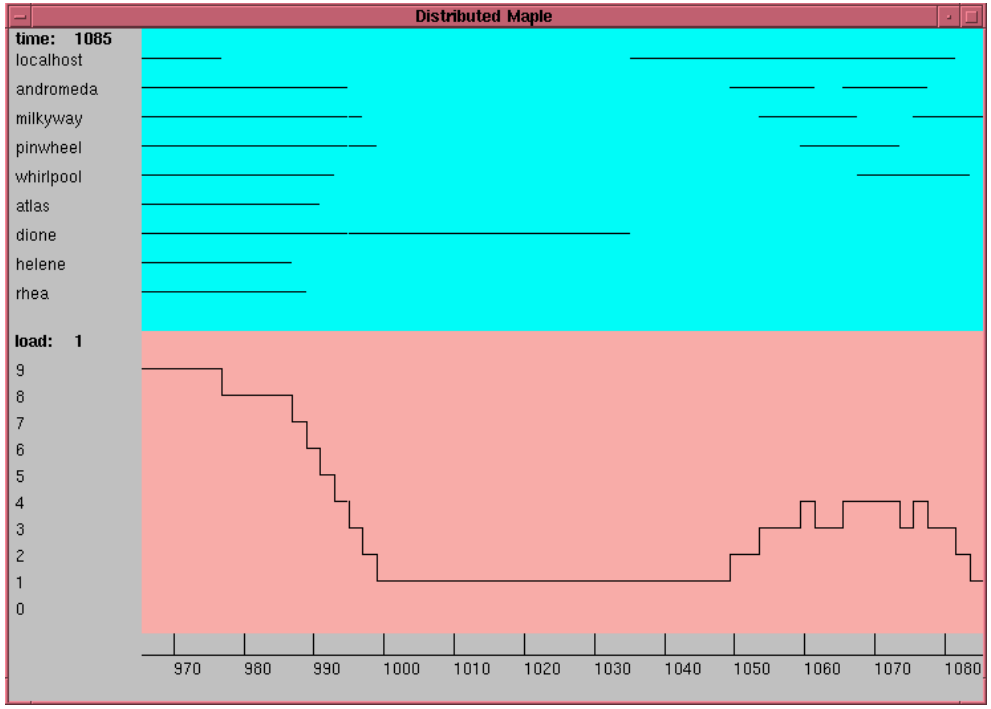
Scheduler

- Task Management.
  - Tasks to be scheduled for execution.
  - Task results and result caches.
- Distributed Operation.
  - Remote system startup.
  - Multi-threaded service handling.
  - Watchdog threads monitor connectivity.
- Visualization
  - Online visualization.
  - Generation of trace files.



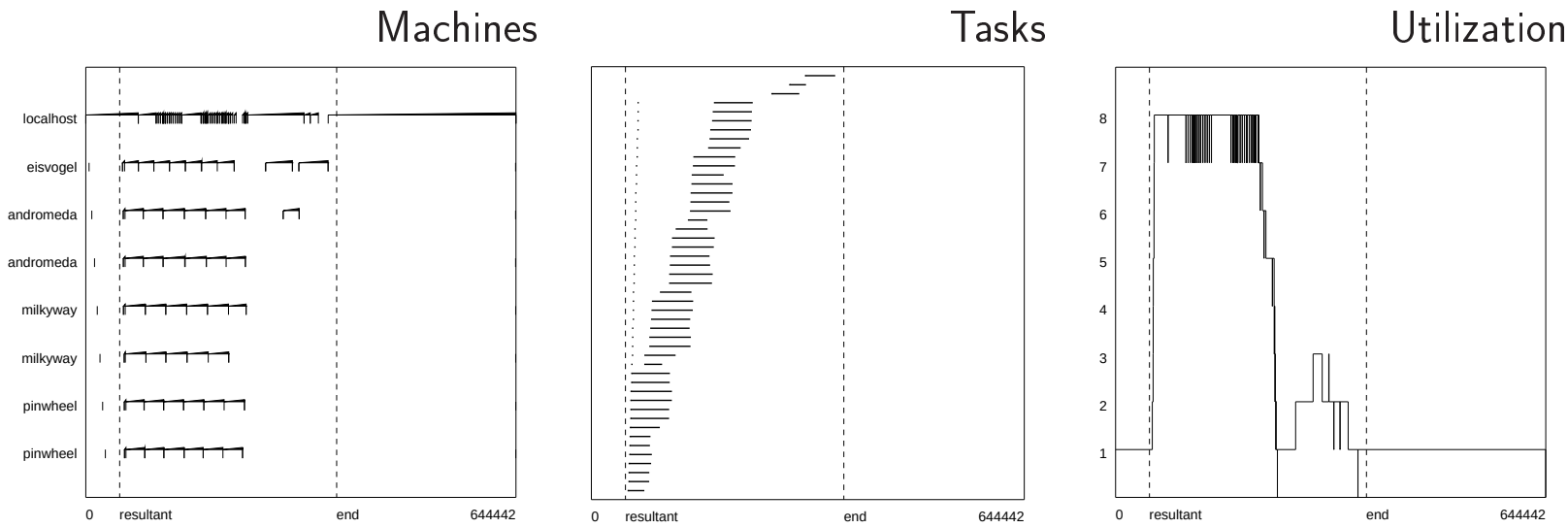
Can be combined with any computational engine.

Online Visualization



Current system status.

Post-Execution Visualization



Analysis of trace files.

Parallel Algebraic Geometry

Speeding up the Algorithm

1. Parallelization of critical point computation.
2. Parallel resultant computation.
3. Improved and parallel real root isolation.
4. Algorithmic improvements.

Speedups on multiple layers possible.

Parallelization of Critical Point Computation

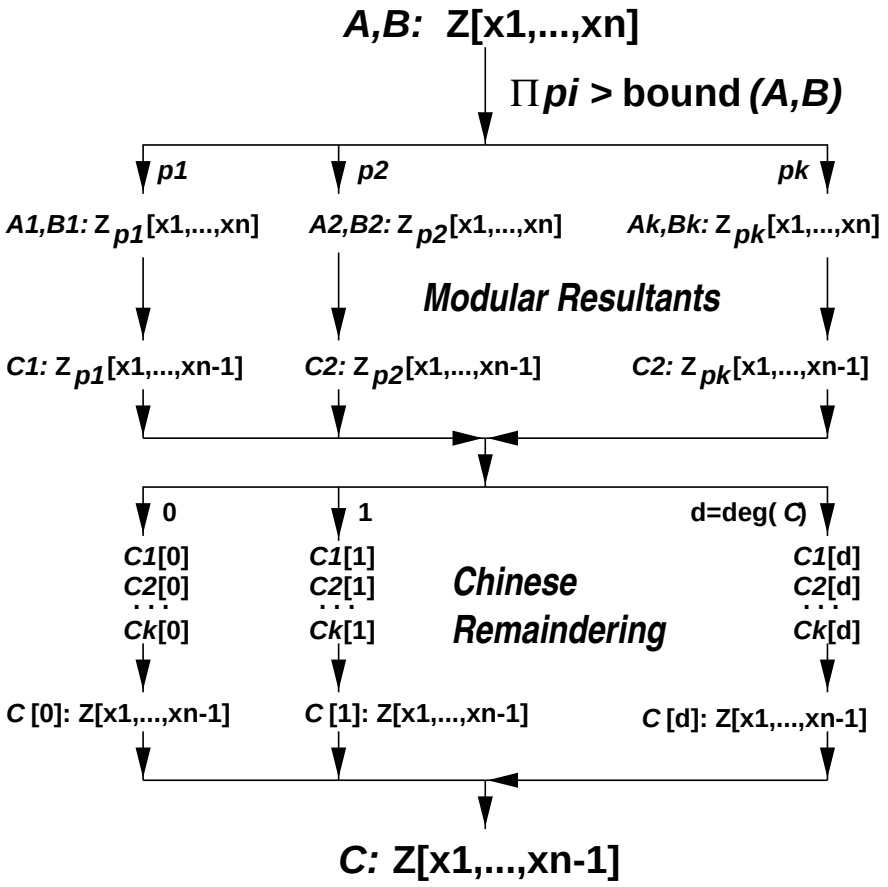
P := criticalPoints(x):

$P := \emptyset$
$S :=$ triangulize(x)
for $(p, q) \in S$ **do** *// parallel loop*
  $Y :=$ realroot(p) *// parallel realroot*
  $q' :=$ squarefree$_\alpha(q, p)$
  $q'' :=$ substitute$_\alpha(q')$
  $R :=$ resultant(p, q'') *// parallel resultant*
  $C :=$ realroot(R) *// parallel realroot*
  for $y \in Y$ **do** *// parallel loop*
    for $c \in C$ **do**
      if check$_\alpha(c, y, p, q')$ **then** $P := P \cup \{(y, c)\}$
    end
  end
end

Parallel Resultant Computation

- Polynomial Resultants
  - Given: $A, B \in \mathbb{Z}[x_1, \ldots, x_n]$
  - Wanted: $C \in \mathbb{Z}[x_1, \ldots, x_{n-1}]$, the resultant of A and B.
- Modular Algorithm
  - Solve problem in multiple finite precision integer domains.
  - Lift result back to the domain of (arbitrary precision) integers.

Result coefficients become extremely large!



Distributed Maple Implementation

Wolfgang Schreiner, 1999.

- Modification of Maple `resultant`.
  - Replacement of `bivariate/modular`.
  - Called for bivariate polynomials of high degree.
  - Other cases solved faster by sequential methods.
- Performance Tuning
  - M images per modular task.
  - C/M images per CRA task.
  - $\Rightarrow$ proportional task granularities.
  - $\approx$ 10–30s per task.

Parallelization based on functional model.

Execution Environment

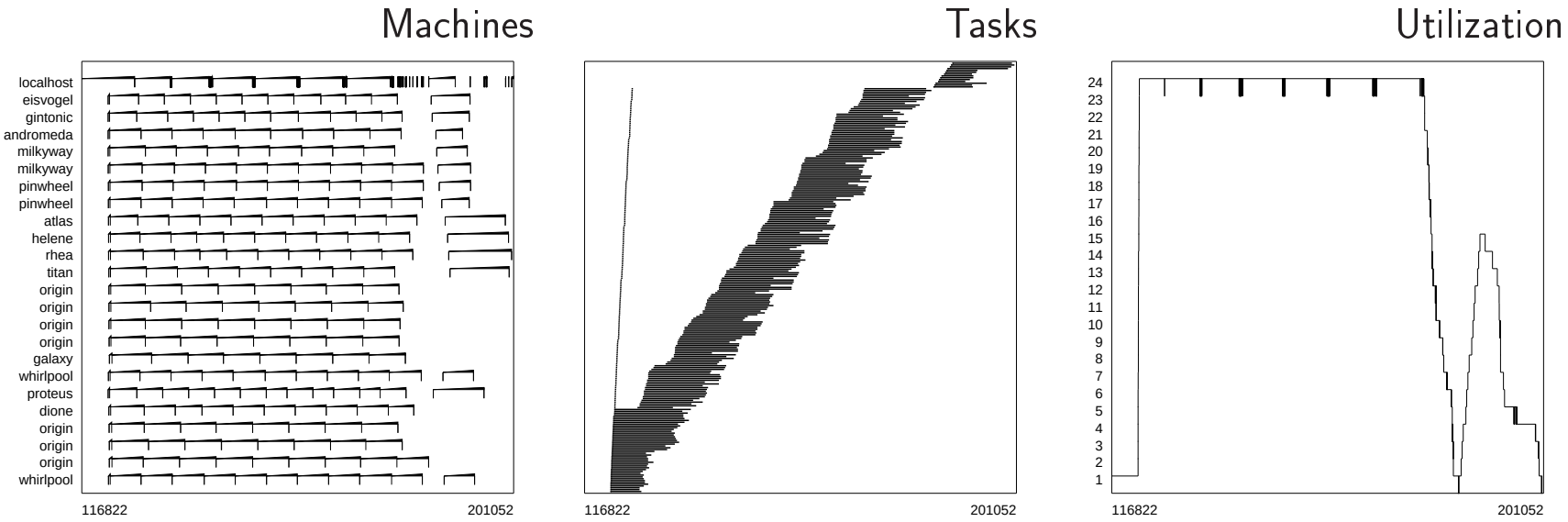
- Sequential Execution.
  - 1 Linux PC (1 Pentium@333MHz).
- Parallel Execution
  - 3 Linux PCs (1 Pentium@333MHz, local Ethernet).
  - 4 SGI Octanes (2 R10000@250MHz, local Ethernet).
  - 5 Linux PCs (1 Pentium@266MHz, local Ethernet).
  - 1 Sun UltraSparc (1 Sparc@296MHz, local Ethernet).
  - 1 SGI Origin (32 R10000@195MHz, 7 used, wide area ATM).
- Raw Performance.
  - Processor Number: 24.
  - Performance Factor: 17.99

Heterogeneous network of PCs, workstations, multiprocessors.

Benchmark Curves

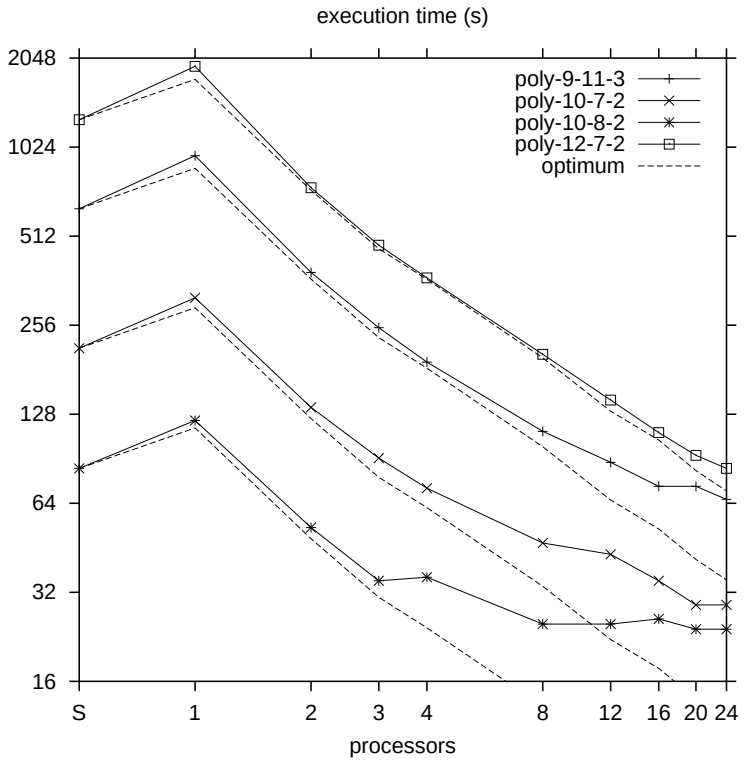
Name	Tot.Deg.	Value
9-11-3	14	$23x^4y^8 - 17x^6y^6 + 22x^8 + 84x^7y^7 + 20x^4y^6 -$ $49x^3y^8 - 80x^4y^2 - 5x^3y - 92x^3y^2 - 47y^8 + 5xy^5$
10-7-2	13	$55x^8y^2 + 66x^9y^4 + 27y^3 + 31x^2y^5 - 75x^2y^6 - 70y^8 - 97x^2y^9$
10-8-2	18	$-22x^9y^9 - 26x^2y^5 + 85x^8y - 50x^2y^6 - 74x^6y^4 - 31x^8y^7 - 172x^7y^7$
12-7-2	20	$99xy^9 - 12x^6y^2 - 26x^2y^{11} - 47x^8y^{11} - 61x^5y^8 - 90x^9y^{11} + 94x^2y$

Visualization



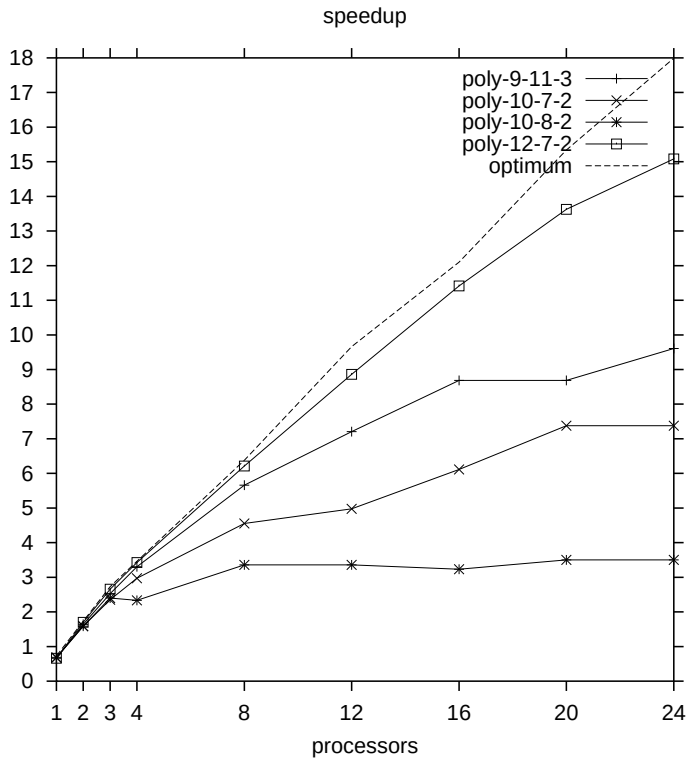
Chinese remaindering becomes bottleneck.

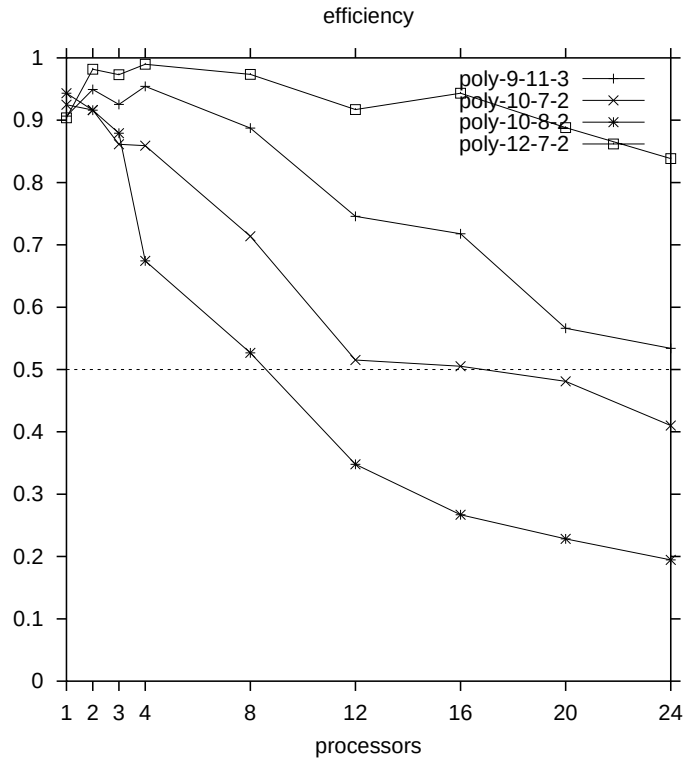
Execution Time



From 1267s (fastest single processor) to 84s (network).

Speedup and Efficiency





80% efficiency with 24 processors in a heterogeneous network.

Real Root Isolation

Given:	$p \in \mathbb{Z}[x]$, squarefree
Find:	set of isolating intervals (x_i', x_i'') for the roots of p.
Algorithm:	Uspensky(A) $\cup$ Uspensky'(A)

Isolation of the positive real roots of A of degree n:

R := Uspensky(A):
Let 2^k be a root bound for A.
if $k \geq 0$
then $L :=$ ZeroOne($A(2^k x)$)
else $L :=$ ZeroOne($2^{-kn} A(2^k x)$)
$R := \{(2^k a, 2^k b) | (a, b) \in L\}$.

Solve problem for transformed polynomial with roots in $[0, 1]$.

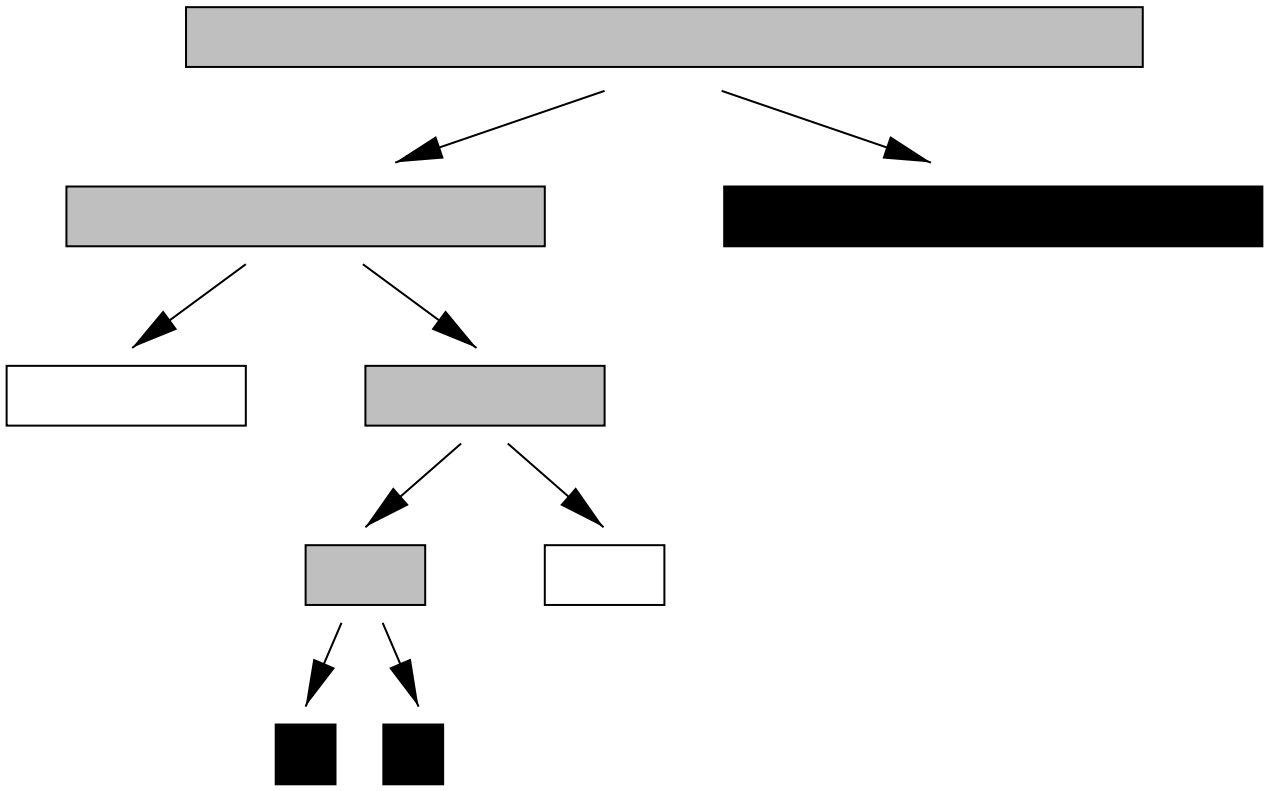
Recursive Algorithm

R := ZeroOne(A):

$A^* := (x+1)^n A(1/(x+1))$;
if signvar$(A^*) = 0$ **then** $R := \emptyset$; **return**
if signvar$(A^*) = 1$ **then** $R := \{(0,1)\}$; **return**
if $A(1/2) = 0$
  then $R := \{(1/2, 1/2)\}$; $A' := A/(2x-1)$
  else $R := \{\}$; $A' := A$
$A'' := 2^n A'(x/2)$
$R := R \cup$
  $\{(a/2, b/2) | (a,b) \in \mathsf{ZeroOne}(A'')\} \cup$
  $\{((a+1)/2, (b+1)/2) | (a,b) \in \mathsf{ZeroOne}(A''(x+1))\}$

Refine intervals until we know that they contain no root or one root.

Recursive Algorithm



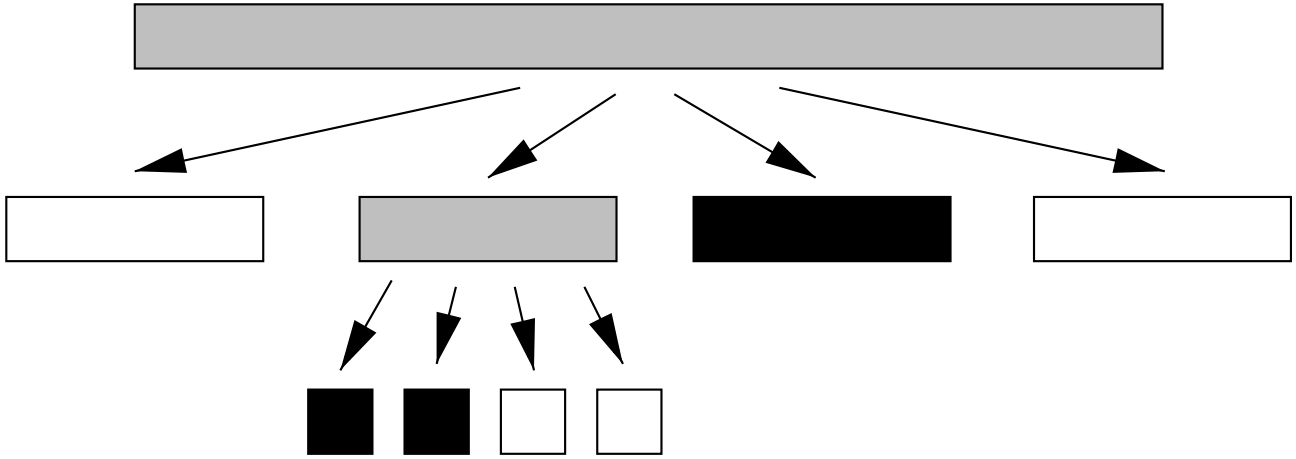
Highly unbalanced binary trees with few long paths.

Parallelization Approaches

1. Positive and negative roots in parallel.
2. Parallelization of recursion tree.
   - Deep and narrow trees $\Rightarrow$ only 1-2 processors active.
3. Parallelization of polynomial transformations.
   - Fine grained parallelism $\Rightarrow$ shared memory multiprocessors only.
4. Widen and flatten the recursion tree
   - Split interval into 2^k subintervals.
   - More processors can be active at same time.
   - More work is done $\Rightarrow$ *speculative parallelism*.

Tradeoff between parallelism and amount of work.

Speculative Parallelism



Add some speculation to keep more processors busy.

Real Root Isolation with p Processors

Christian Mittermaier, 1999.

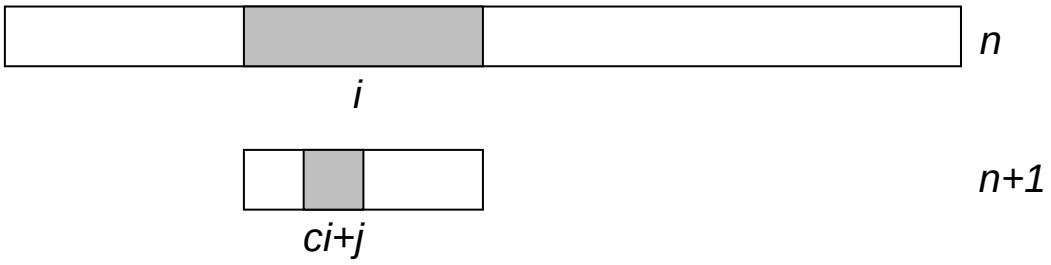
R := realroot(A):

$I := \{[0,1]\}$
$W := \{\}$
$R := \{\}$
while $I \cup W \neq \{\}$
  Split each of the n intervals in I into 2^d pieces
    such that $|W| + n * 2^d \leq p < |W| + n * 2^{d+1}$ ($d := 1$, if $p \leq |W|$)
  $W := W \cup$ the split intervals
  Compute the sign transformations of A on p of the intervals in W in parallel
  $W := W -$ the processed intervals
  $R := R \cup$ the intervals with one root
  $I :=$ the intervals with unknown number of roots
end
Refine intervals in R to desired accuracy in parallel

Representation

- Polynomial Transformation
  - The i-th of 2^n intervals of $[0,1]$ is represented by a polynomial $A_{n,i}$.
  - An interval i on level n is split into $c = 2^d$ pieces on level $n+1$ numbered $ic \ldots (i+1)c-1$:

$$A_{n+1,i*c+j} := (c^n A_{n,i}(x/c))(x+j)$$



Each task returns the computed $A_{n,i}$ (together with sign variation) for further reuse.

Computational Improvements

- Use primitive part of polynomial

  Limit size of coefficients; done in CASA `realroot` but not in Maple `realroot`.

$$A := \text{primpart}(A)$$

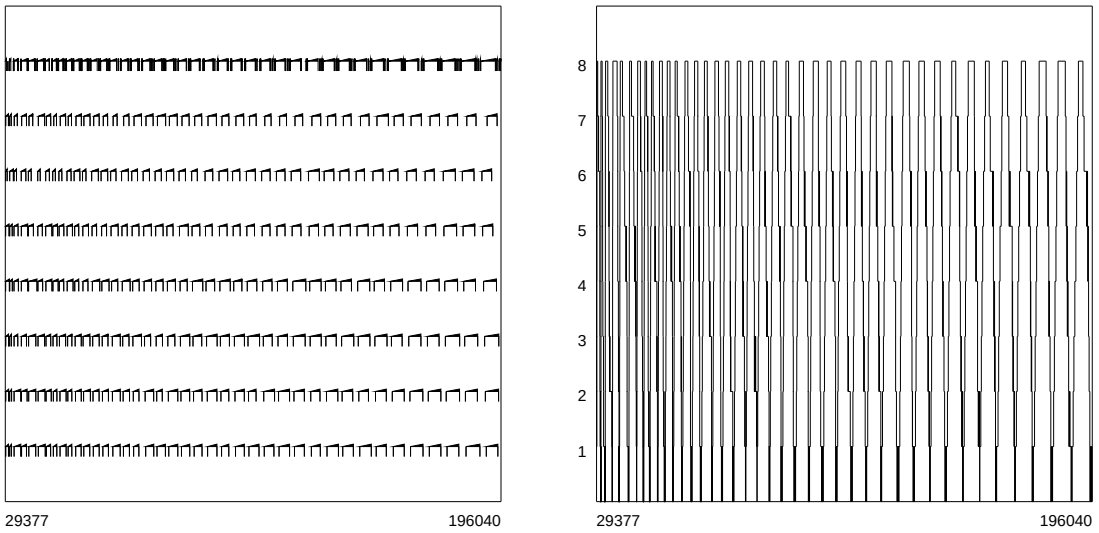
- Use Horner Schema for Transformations:

  Neither done in CASA `realroot` nor in Maple `realroot`.

$$\begin{aligned} \text{Old:} \quad & A^* := (x+1)^n A(1/(x+1)) \\ \text{New:} \quad & A^* := \text{expand}(\text{horner}(x^n A(1/x))(x+1)) \end{aligned}$$

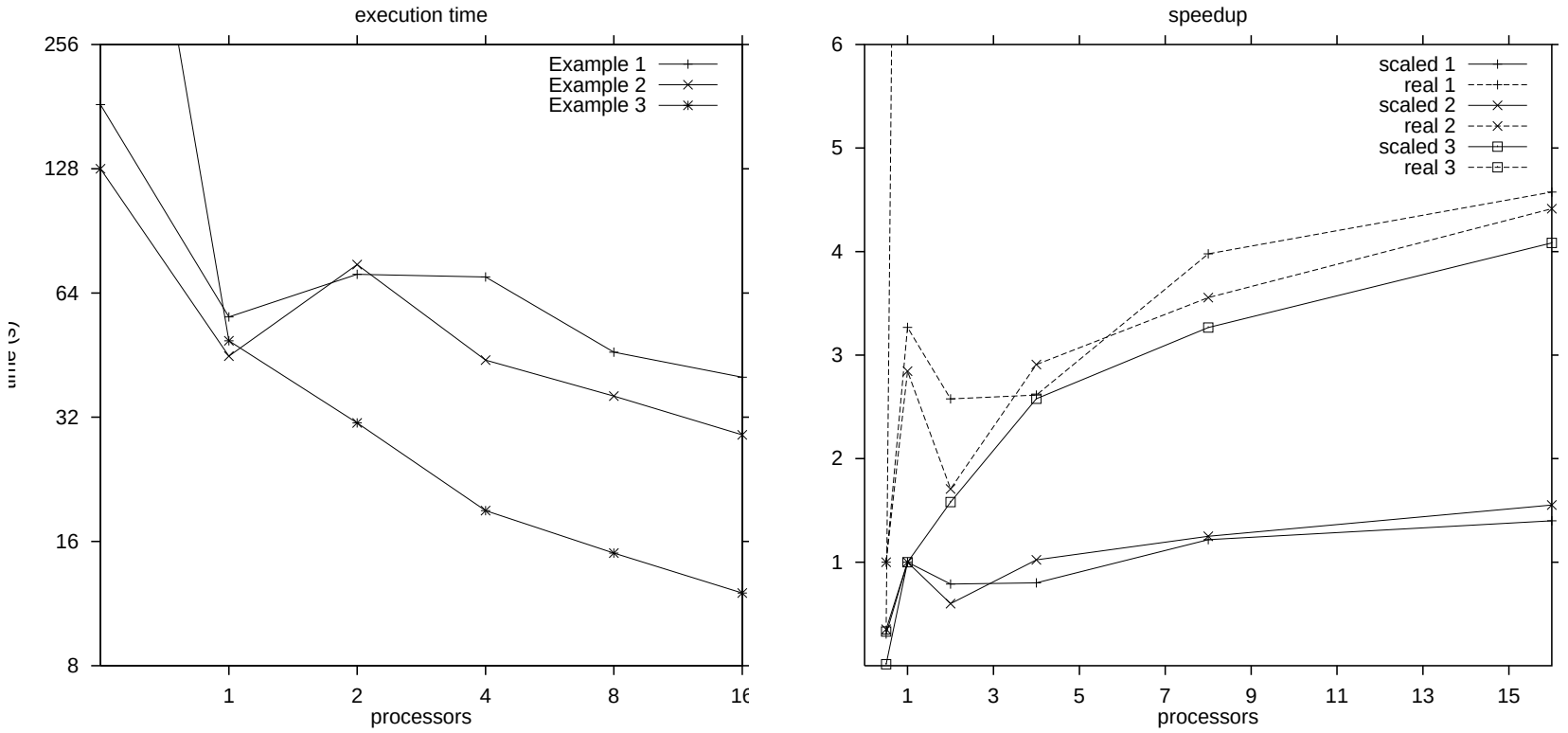
	Maple	CASA	New
Example 1	124s	164s	68s
Example 2	85s	109s	40s
Example 3	1306s	174s	47s

Parallel Behavior



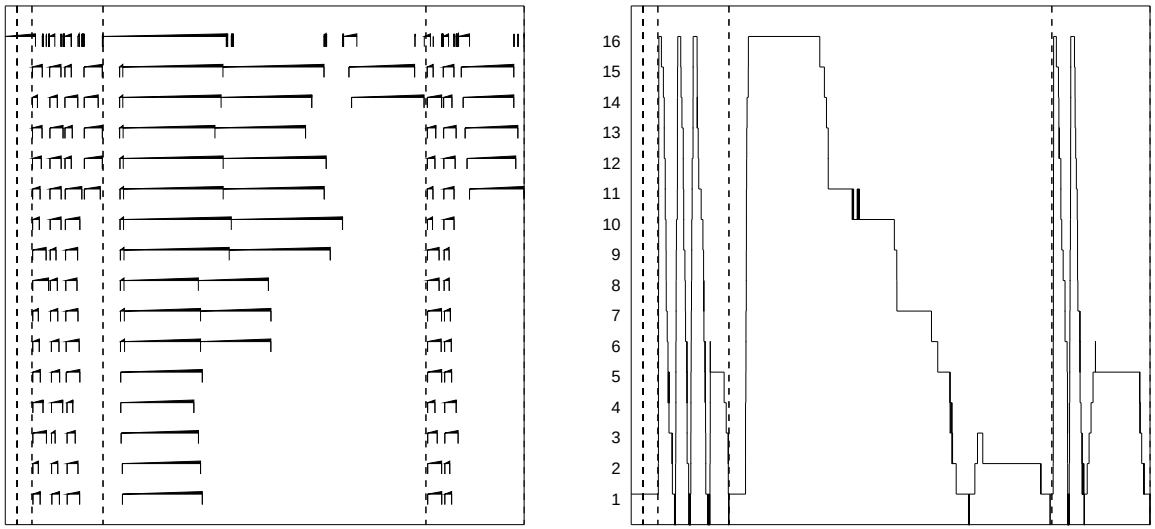
Large number of short parallel phases.

Execution Times and Speedups



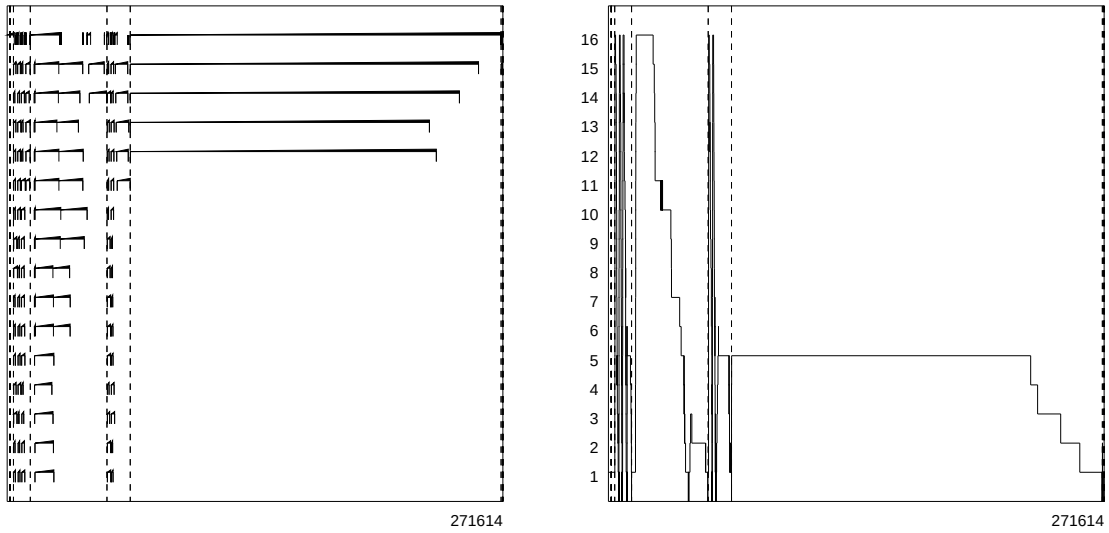
Speedup 3 *by computational improvements, 1.5-4 by parallelization.*

Integration with Parallel Resultant Computation



Real root isolation more fine-grained than resultant computation.

Integration with Parallel Loop



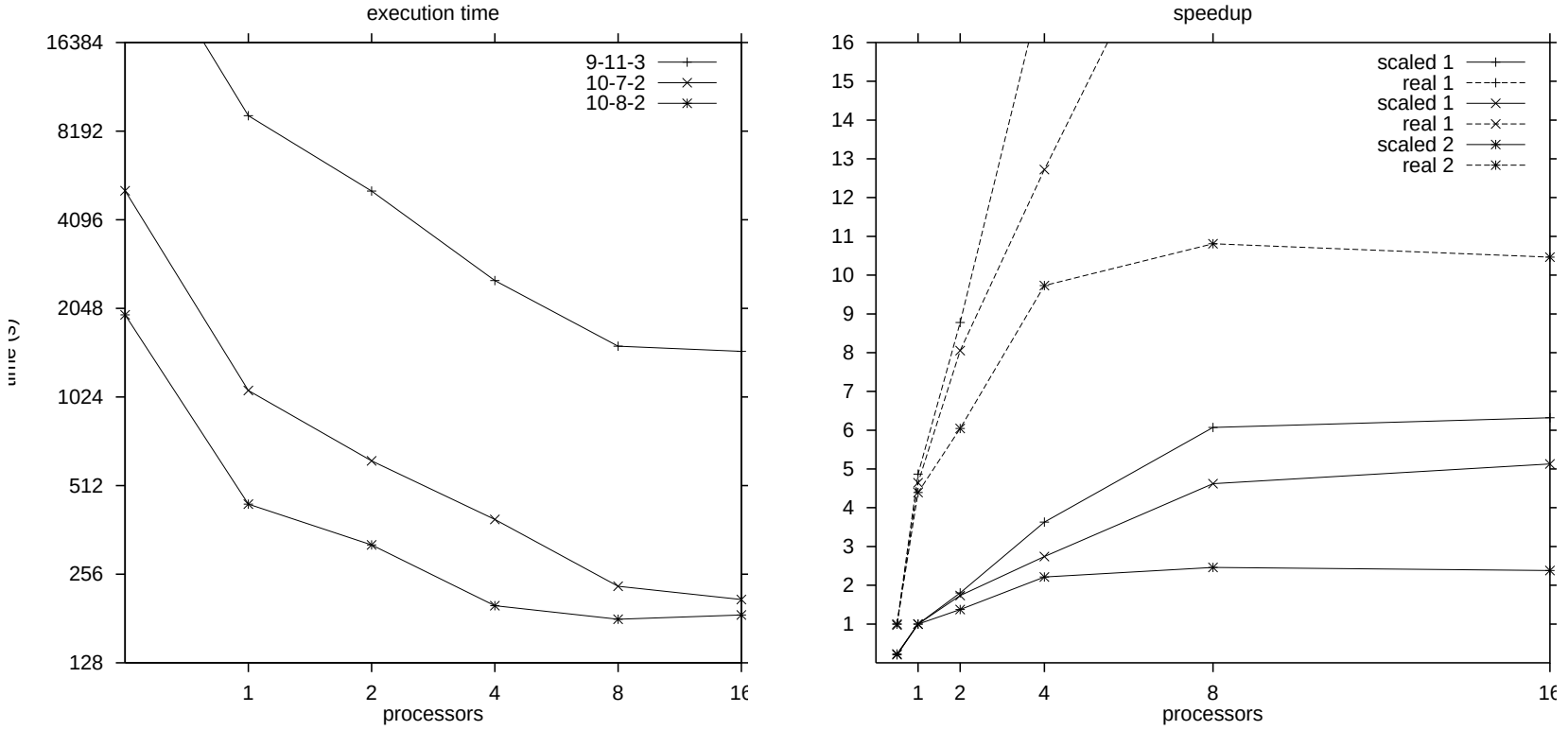
Often too few parallel loop iterations to keep all machines busy.

Algorithmic Improvements

P := criticalPoints(x):
  $P := \emptyset$
  $S :=$ triangulize(x)
  for $(p, q) \in S$ **do**
    $Y :=$ realroot(p)
    $R :=$ resultant(p, q)
    $C :=$ realroot(squarefree(R))
    for $y \in Y$ **do**
      for $(c', c'') \in C$ **do**
        if check'(y, p, squarefree($q(c', \alpha)$), squarefree($q(c'', \alpha)$)) **then** $P := P \cup \{(y, c)\}$
      end
    end
  end

Compute gcd in $\mathbb{Q}[y]$, not in $\mathbb{Q}(\alpha)[y]$!

Execution Times and Speedups



Speedup ≈ 5 *by parallelization,* ≈ 5 *by first algorithmic improvement.*

Conclusions

- Achievements
  - Portable Java-based parallel execution framework.
  - Speedup of up to 15 for resultant computation and up to 4 for real root isolation.
  - Improvements of real root isolation (factor 3) and of plotting algorithm (factor 5).
  - Total improvement by factor of 25 (more expected by second improvement).
- Current Work
  - Integration of parallel and algorithmic improvements.
  - Fine-tuning of performance parameters.
  - More CASA algorithms.

`http://www.risc.uni-linz.ac.at/software/casa`
`http://www.risc.uni-linz.ac.at/software/distmaple`