Component-level Parallelization of Triangular Decompositions

Marc Moreno Maza and Yuzhen Xie

University of Western Ontario, Canada

August 16, 2008

for

Parallel Symbolic Computation Workshop, 2007

Solving polynomial systems symbolically . . .

- Polynomial systems :
  - systems of non-linear algebraic (or differential) equations,
  - solving them is a fundamental problem in mathematical sciences,
  - which is hard for both numerical and symbolic approaches.
- Symbolic solving :
  - provides exact answers,
  - but suffers from expression swell.
- Applications of symbolic solving :
  - increasing number of applications (cryptology, robotics, geometric modeling, dynamical systems in biology, . . .)
  - can now compete with numerical solving (real solving)
  - sometimes, this is the only way to go (parametric solving, solving over finite fields).

Why solving non-linear systems is much more difficult?

Let $F \subset \mathbb{K}[X]$ with $X = x_1 < \cdots < x_n$ and a coefficient field $\mathbb{K}$. Let d be the maximum (total) degree of a monomial in F.

Let $V(F) \subset \overline{\mathbb{K}}^n$ be the zero set of F, where $\overline{\mathbb{K}}$ is an algebraically closed field containing $\mathbb{K}$. For instance $\mathbb{K} = \mathbb{Q}$ and $\overline{\mathbb{K}} = \mathbb{C}$.

- $V(F)$ may consist of components of **different dimension**: points, curves, surfaces, $\ldots$,
- Even if $V(F)$ is finite, it may contain $O(d^n)$ points,
- The idea of *substitution* or *simplification* is much **more complicated** than in the linear case and leads to the notion of a *Gröbner basis*,
- **Large intermediate data**.

Solving polynomial systems symbolically

$$\left\{\begin{array}{l} x^2+y+z=1 \\ x+y^2+z=1 \\ x+y+z^2=1 \end{array}\right.$$ <u>has Gröbner basis</u> :

$$\left\{\begin{array}{r} z^6-4z^4+4z^3-z^2=0 \\ 2z^2y+z^4-z^2=0 \\ y^2-y-z^2+z=0 \\ x+y+z^2-1=0 \end{array}\right.$$ <u>and triangular decomposition</u> :

$$\left\{\begin{array}{l} z=1 \\ y=0 \\ x=0 \end{array}\right. \bigcup \left\{\begin{array}{l} z=0 \\ y=1 \\ x=0 \end{array}\right. \bigcup \left\{\begin{array}{l} z=0 \\ y=0 \\ x=1 \end{array}\right. \bigcup \left\{\begin{array}{r} z^2+2z-1=0 \\ y=z \\ x=z \end{array}\right.$$

Solving polynomial systems symbolically and in parallel: related work

- Parallelizing the computation of Gröbner bases (R. Bündgen, M. Göbel & W. Küchlin, 1994) (S. Chakrabarti & K. Yelick, 1993 - 1994) (J.-C. Faugère, 1994) (G. Attardi & C. Traverso, 1996) (A. Leykin, 2004)

- Parallelizing the computation of characteristic sets (D.M. Wang, 1994) (I.A. Ajwa, 1998), (Y.W. Wu, W.D. Liao, D.D. Liu & P.S. Wang, 2003) (Y.W. Wu, G.W. Yang, H. Yang, H.M. Zheng & D.D. Liu, 2005)

Parallelizing the computation of Gröbner bases

Input: $F \subset \mathbb{K}[X]$ and an admissible monomial ordering $\leq$.

Output: G a reduced Gröbner basis w.r.t. $\leq$ of the ideal $\langle F \rangle$ generated by F.

repeat

(S) $B :=$ MinimalAutoreducedSubset(F, $\leq$)

(R) $A :=$ S_Polynomials(B) $\cup F$;

$R :=$ Reduce(A, B, $\leq$)

(U) $R := R \setminus \{0\}$; $F := F \cup R$

until $R = \emptyset$

return B

The characteristic set method

Input: $F \subset \mathbb{K}[X]$.

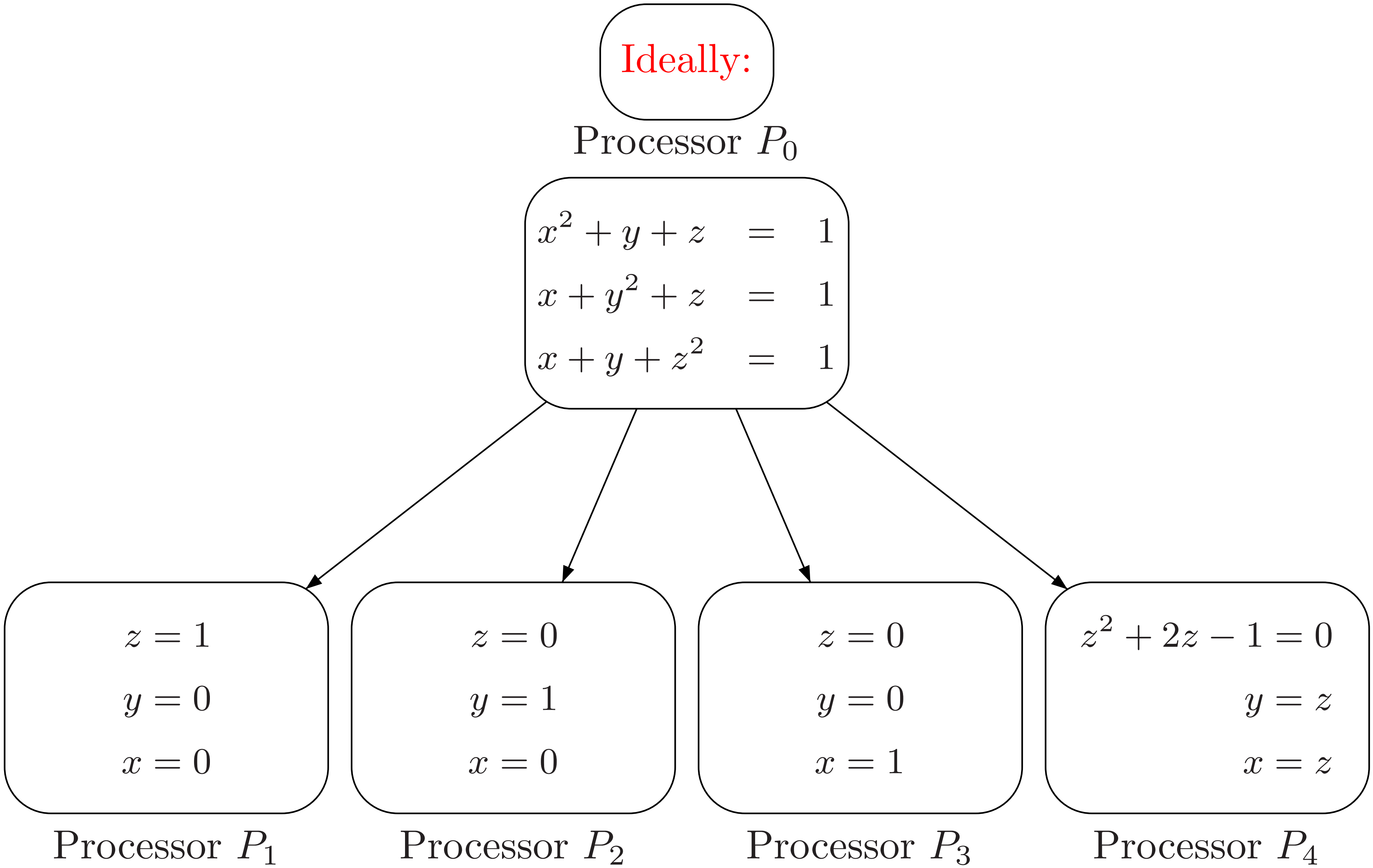
Output: C an autoreduced characteristic set of F (in the sense of Wu).

repeat
(S) $B :=$ MinimalAutoreducedSubset(F, $\leq$)
(R) $A := F \setminus B$;
$R :=$ PseudoReduce(A, B, $\leq$)
(U) $R := R \setminus \{0\}$; $F := F \cup R$
until $R = \emptyset$
return B

- Repeated calls to this procedure computes a decomposition of $V(F)$.
- Cannot start computing the 2nd component before the 1st is completed.

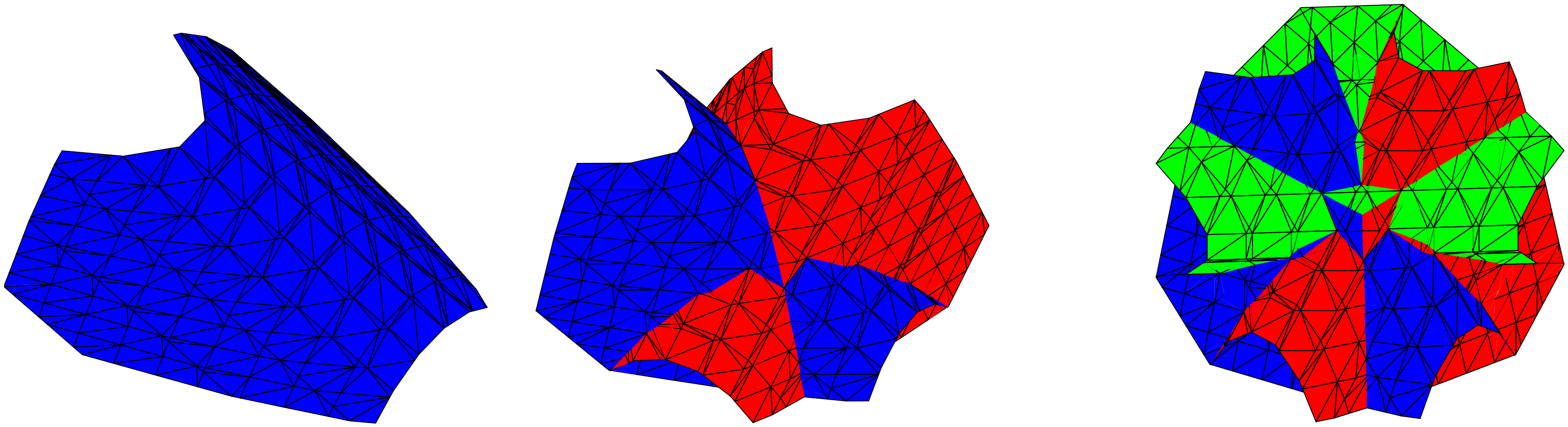
Solving polynomial systems symbolically and in parallel: the context of our work

- **New motivations**:
  - renaissance of parallelism,
  - new algorithms, modular triangular decompositions, offering better opportunities for parallel execution.
- **Our goal**:
  - multi-level parallelism:
    * coarse grained "component-level" for tasks computing geometric objects,
    * medium/fine grained level for polynomial arithmetic within each task.
  - In component-level, the number of processes in use depends on the geometry of the solution set


Ideally:
Processor P_0
$x^2 + y + z = 1$
$x + y^2 + z = 1$
$x + y + z^2 = 1$
$z = 1$
$y = 0$
$x = 0$
Processor P_1
$z = 0$
$y = 1$
$x = 0$
Processor P_2
$z = 0$
$y = 0$
$x = 1$
Processor P_3
$z^2 + 2z - 1 = 0$
$y = z$
$x = z$
Processor P_4

An algorithm for triangular decomposition

Incremental solving: by solving one equation after the other, lead to a more geometric approach.



$$\left\{ \; x^2 + y + z = 1 \right.$$

$$\left\{ \begin{array}{ccl} x & + & y^2 + z = 1 \\ y^4 & + & (2z-2)y^2 \\ & + & y - z + z^2 = 0 \end{array} \right.$$

$$\left\{ \begin{array}{r} x + y = 1 \\ y^2 - y = 0 \\ z = 0 \end{array} \right.$$

$$\left\{ \begin{array}{r} 2x + z^2 = 1 \\ 2y + z^2 = 1 \\ z^3 + z^2 - 3z = -1 \end{array} \right.$$

An algorithm for triangular decomposition

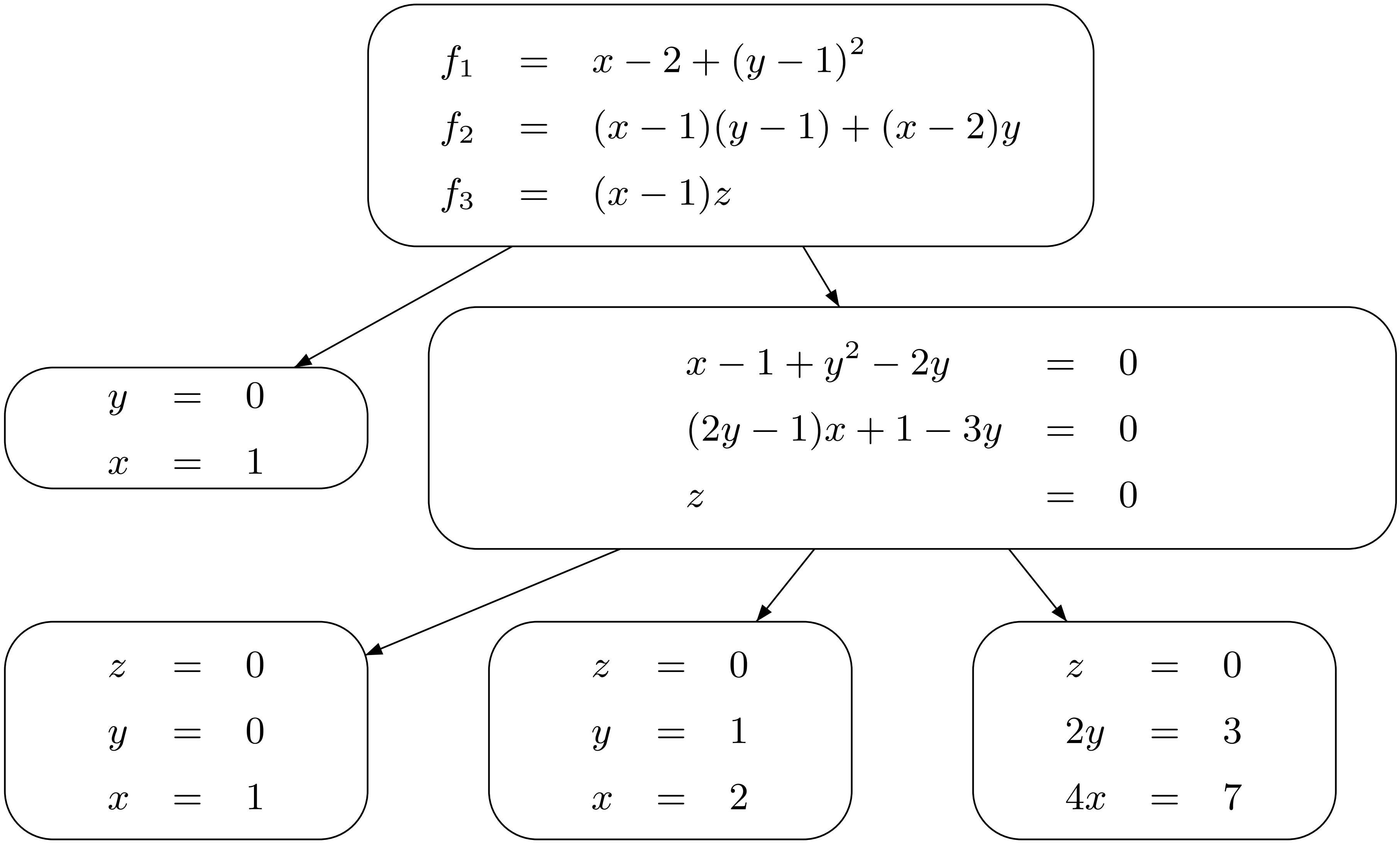
A task manager scheme: Triade (M. Moreno Maza, 2000)

- A *task* is any couple $[F, T]$ where $F \subset \mathbb{K}[X]$ and $T \subset \mathbb{K}[X]$ is a triangular system, more precisely a regular chain.
  - if $F = \emptyset$, the task is *solved*,
  - otherwise, *solving* $[F, T]$ means to compute triangular systems $T_1, \ldots, T_\ell$ representing $Z(F, T)$, the common zeros of F and T.

Lazy evaluation and solving by decreasing order of dimension: computing tasks $[F_1, T_1], \ldots, [F_\ell, T_\ell]$ s.t

- each $[F_i, T_i]$ is closer to be solved than $[F, T]$,
- $Z(F_1, T_1) \cup \cdots \cup Z(F_\ell, T_\ell)$ represents $Z(F, T)$,
- for all i we have $F_i = \emptyset$ whenever T_i has maximum dimension.

Initial task $[\{f_1, f_2, f_3\}, \emptyset]$



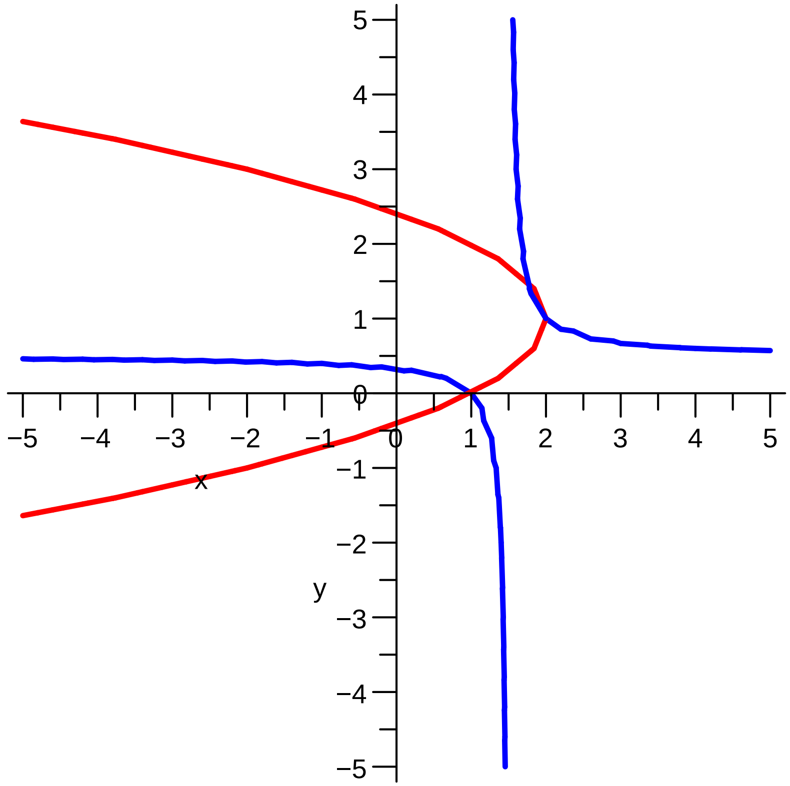
Triade top level

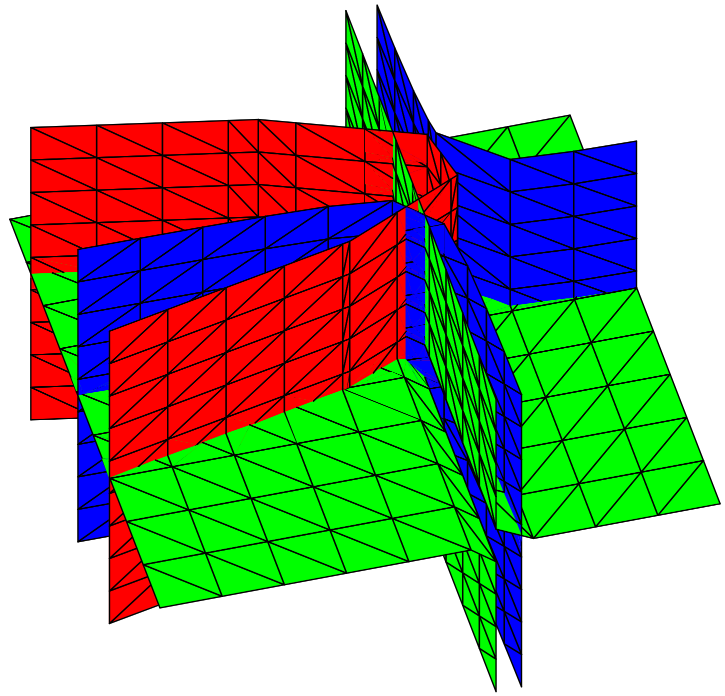
Input: $F \subset \mathbb{K}[X]$.

Output: $\mathcal{T}$ a triangular decomposition of $V(F)$.

$ToDo := [F, \emptyset]$; $\mathcal{T} := [\,]$
repeat
(S) $Tasks := \mathrm{Select}(ToDo)$
(R) $Results := \mathrm{LazySolve}(Tasks)$
(U) $(ToDo, \mathcal{T}) := \mathrm{Update}(Results, ToDo, \mathcal{T})$
until $ToDo = \emptyset$
return $\mathcal{T}$

Difficulty 1: Removing redundant computation



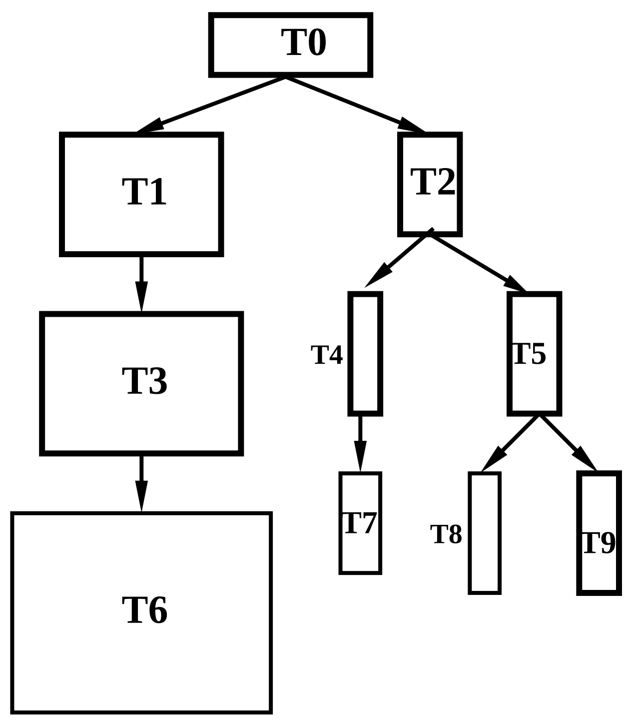


The red and blue surfaces intersect on the line $x - 1 = y = 0$ contained in the green plane $x = 1$. With the other green plane $z = 0$, they intersect at $(2, 1, 0)$, $(\frac{7}{4}, \frac{3}{2}, 0)$ but also at $x - 1 = y = z = 0$, which is redundant.

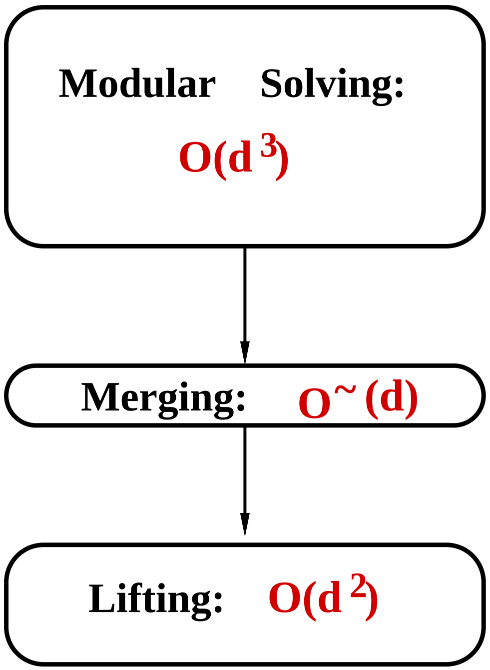
Difficult 2: Dynamic and very irregular computations

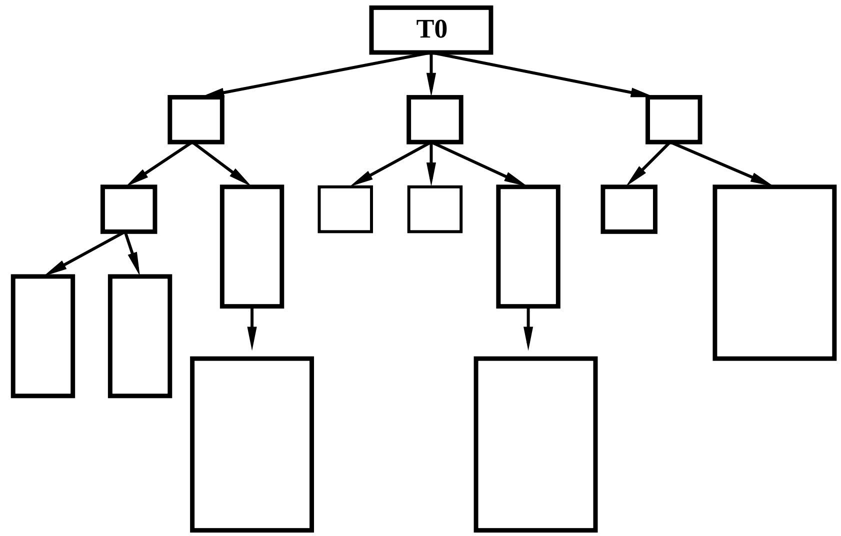
• **Very irregular tasks** (CPU time, memory, data-communication)

• Moreover, most polynomial systems $F \subseteq \mathbb{Q}[X]$ (arising both in practice and in theory) can be represented by a single triangular set.



Create parallelism: using modular methods





For solving $F \subseteq \mathbb{Q}[X]$ we use modular methods. Indeed, for a prime p:

- irreducible polynomials in $\mathbb{Q}[X]$ are likely to factor modulo p,
- for p big enough, the result over $\mathbb{Q}$ can be recovered from the one over $Z/pZ[X]$.

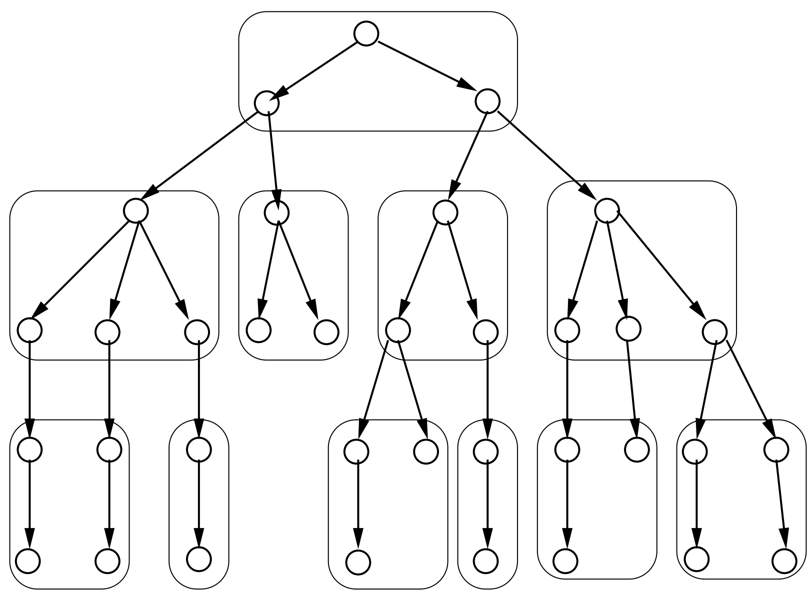
(X. Dahan, M. Moreno Maza, É. Schost, W. Wu & Y. Xie, 2005)

Effect of modular solving

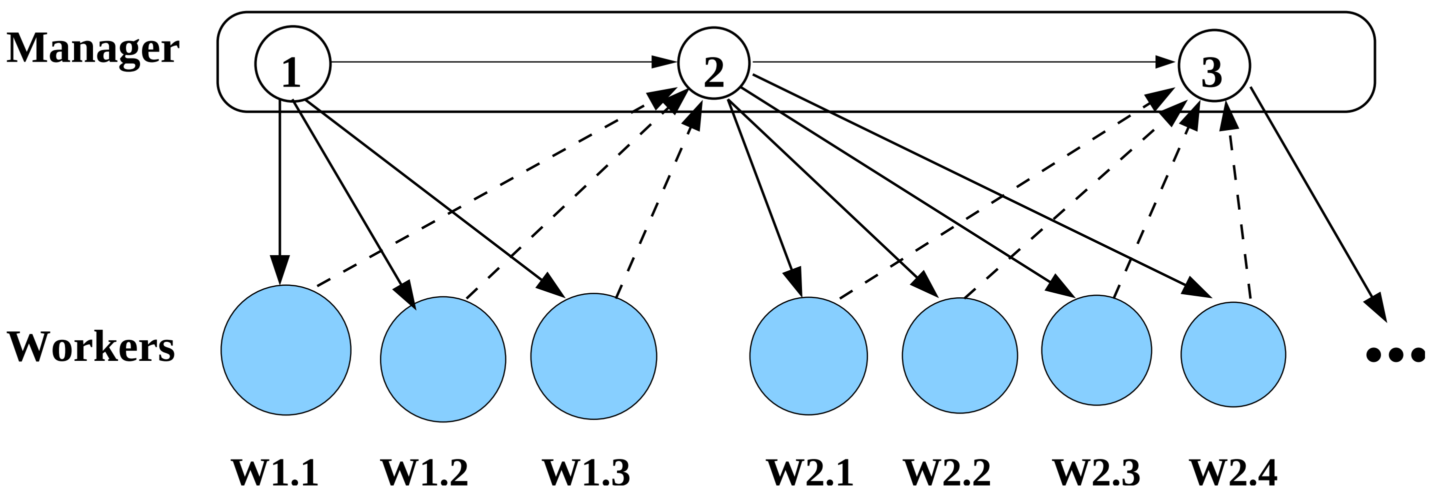
Sys	Name	n	d	p	*Degrees*
1	eco6	6	3	105761	[1,1,2,4,4,4]
2	eco7	7	3	387799	[1,1,1,1,4,2, 4,4,4,4,4,2]
3	CassouNogues2	4	6	155317	[8]
4	CassouNogues	4	8	513899	[8,8]
5	Nooburg4	4	3	7703	[18,6,6,3,3,4,4,4,4,2,2,2, 2,2,2,2,2,1,1,1,1,1]
6	UteshevBikker	4	3	7841	[1,1,1,1,2,30]
7	Cohn2	4	6	188261	[3,5,2,1,2,1,1,16,12,10,8,8, 4,6,4,4,4,4,2,1,1,1,1,1,1,1, 1,1,1,1,1,1,1]

Exploit parallelism!

- Driving idea: limit the irregularity of tasks. In particular,
  - to avoid inexpensive computations leading to expensive data communication.
  - to balance the work among the workers.
- use regularized initial and split-by-hight
- estimate the cost of a task by its rank and dimension to guide the scheduling.



Task Pool with Dimension and Rank Guided (TPDRG) dynamic scheduling



Challenges in the implementation

- dynamic process creation and management,
- scheduling of highly irregular tasks,
- complex data types, such as the polynomial data type,
- heavy data-communication and synchronization.

Preliminary implementation

- Parallel framework: multi-processed parallelism support in Aldor on SMPs and multicores
  - using shared memory segments for data communication.
  - high-level objects (e.g. sparse multivariate polynomials) are serialized.
- Supported by the `BasicMath` library and the sequential Triade solver in Aldor.
- Machine: Silky in SHARCNET (SGI Altix 3700 Bx2, 128 Itanium2 Processors 1.6GHz SMP).

Sequential timing and overhead of regularized initial

Sys	$Sequential$ (s)	$Seq.(regularized\ initial)$ (s)	$slowBy$ (%)
1	3.63	4.00	0.01
2	707.53	727.95	0.01
3	463.02	476.16	0.01
4	2132.87	2162.40	0.01
5	4.10	4.14	0.01
6	866.27	866.20	-
7	298.33	305.24	0.01

Speedup vs #processor

#P	Sys1	Sys2	Sys3	Sys4	Sys5	Sys6	Sys7
3	1.3	2.1	1.7	1.5	2.0	1.4	2.9
5	2.1	3.2	2.2	2.2	2.0	1.8	3.1
7	2.1	5.1	2.3	2.3	2.2	1.8	3.1
9	2.1	6.1	2.3	2.4	2.3	1.9	3.2
11	2.0	6.1	2.3	2.4	2.6	1.9	3.2
13	-	6.1	2.3	2.4	2.5	1.9	3.2

Best TPDRG timing vs Greedy scheduling (s)

System	#P	*TPDRG* (best) (A)	Greedy (A)	#P (B)	Greedy (B)
1	7	1.91	1.79	9	1.78
2	13	119.09	120.51	15	120.52
3	13	206.38	213.21	15	213.35
4	20	852.49	896.79	22	939.62
5	13	1.61	1.63	15	1.63
6	20	451.36	500.50	22	469.35
7	17	96.20	100.78	19	96.17

Summary

- Created opportunities by using modular methods, for coarse grained component-level parallel solving of polynomial systems in $\mathbb{Q}[X]$
- Exploited these opportunities by transforming the Triade algorithm: strengthen its notion of a task by regularized initial and split-by-height.
- Geometrical information guided scheduling.
- A preliminary implementation using multi-processed parallelism support in Aldor.
- Launched the first step towards multi-level parallelization.
- Expect the speedup in component-level parallelization would add a multiplicative factor to the medium/fine level.
- Limitation of this implementation: memory

Towards efficient multi-level parallelization

- Build Aldor threads to support fine parallelism for symbolic computations targeting SMP and multi-cores. In particular,
  - properly treat parametric types, such as polynomial data types,
  - thread scheduling by *work-stealing* and *work first principle*.
- Investigate multi-level parallelism for triangular decompositions over clusters:
  - coarse grained level (multi-processed) for tasks to compute geometric of the solution sets.
  - medium/fine grained level (multi-threaded) for polynomial arithmetic such as multiplication, GCD/resultant, and factorization.
  - to improve the performance of symbolic solvers on emerging architectures.