

# Parallel Implementation of a Subsystem-by-Subsystem Solver

Yun Guan    Jan Verschelde

Department of Math, Stat, & CS  
University of Illinois at Chicago  
<http://www.math.uic.edu/~guan>  
[guan@math.uic.edu](mailto:guan@math.uic.edu)

AMS session "Numerical and Symbolic Techniques in  
Algebraic Geometry and Its Applications"  
Oct. 5, 2007

# Outline

- 1 Problem Statement
- 2 Introduction
  - Witness Sets
  - Diagonal Homotopy
  - Parallel Diagonal Homotopy Implemented in PHCpack
  - Subsystem-by-Subsystem Solver
- 3 Parallel Implementation of Subsystem-by-Subsystem Solver
  - Algorithms
  - Equipment and Software
  - Experimental Results

# Problem we want to solve

Homotopy methods to solve polynomial systems is "pleasingly parallel".

- The solution path can be tracked independently.
- scale very well for a large number of processors.

What are we solving?

- large system of family problems such as katsura, economics, adjacent minors;
- systems with more than 100,000 solutions;
- Optimal case (no diverging paths)

# Problem we want to solve

Homotopy methods to solve polynomial systems is "pleasingly parallel".

- The solution path can be tracked independently.
- scale very well for a large number of processors.

What are we solving?

- large system of family problems such as katsura, economics, adjacent minors;
- systems with more than 100,000 solutions;
- Optimal case (no diverging paths)

# Outline

- 1 Problem Statement
- 2 **Introduction**
  - **Witness Sets**
  - Diagonal Homotopy
  - Parallel Diagonal Homotopy Implemented in PHCpack
  - Subsystem-by-Subsystem Solver
- 3 Parallel Implementation of Subsystem-by-Subsystem Solver
  - Algorithms
  - Equipment and Software
  - Experimental Results

# About witness sets

- *Numerical representation* of positive  $d$ -dimensional solution sets.
- Consists of a set of  $k$  general hyperplanes and  $d$  isolated points.
- Compute witness supersets with *a cascade of homotopies*.
- *Filtering* "junk points" to obtain witness sets for all pure  $d$ -dimensional solution sets.

# References

- A.J. Sommese and C.W. Wampler: **Numerical algebraic geometry**. In *The mathematics of numerical analysis (Park City, UT, 1995)*, Vol. 32 of *Lectures in Appl. Math.* (pp.749-763). Providence, RI:Amer. Math. Soc.
- A.J. Sommese and J. Verschelde: **Numerical Homotopies to compute Generic Points on Positive Dimensional Algebraic Sets**. *Journal of Complexity* 16(3):572-602, 2000.
- A.J. Sommese and C.W. Wampler: **The Numerical Solution of Systems of Polynomials Arising in Engineering and Science**. World Scientific Press, 2005.

# Outline

- 1 Problem Statement
- 2 **Introduction**
  - Witness Sets
  - **Diagonal Homotopy**
  - Parallel Diagonal Homotopy Implemented in PHCpack
  - Subsystem-by-Subsystem Solver
- 3 Parallel Implementation of Subsystem-by-Subsystem Solver
  - Algorithms
  - Equipment and Software
  - Experimental Results

## Related Papers

- A.J. Sommese, J. Verschelde, and C.W. Wampler: **Numerical Decomposition of the Solution Sets of Polynomial Systems into Irreducible Components.** *SIAM J. Numer. Anal.* 38(6):2022-2046, 2001.
- A.J. Sommese, J. Verschelde, and C.W. Wampler: **Homotopies for intersecting solution components of polynomial systems.** *SIAM J. Numerical Anal.* 42(4):1552-1571, 2004.
- A.J. Sommese, J. Verschelde, and C.W. Wampler: **An intrinsic homotopy for intersecting algebraic varieties.** *J. Complexity*, 21(3):593-608, 2005.

# What does diagonal homotopy do?

Input: two irreducible components  $A$  and  $B$  given by two witness sets.

Witness Set for $A$	Witness Set for $B$
$\begin{cases} f_A(x) = 0 \\ L_A(x) = 0 \end{cases}$	$\begin{cases} f_B(x) = 0 \\ L_B(x) = 0 \end{cases}$
$\#L_A = \dim(A) = a$	$\#L_B = \dim(B) = b$
$\{\alpha_1, \alpha_2, \dots, \alpha_{\deg(A)}\}$	$\{\beta_1, \beta_2, \dots, \beta_{\deg(B)}\}$

Output: witness sets for all pure dimensional components of  $A \cap B$

# What does diagonal homotopy do? continue...

- 1 Solution pairs are taken to build the start cascade homotopy.
- 2 Embedded hyperplanes are removed one by one to find solutions at the intersection.

$$\#\{x \in \mathbb{C}^n \mid f_A(x) = 0, L_A(x) = 0\} = \alpha$$

$$\#\{y \in \mathbb{C}^n \mid f_B(y) = 0, L_B(y) = 0\} = \beta$$

Assume  $A$  and  $B$  are complete intersections,  $\dim(A \cap B) = 0$

$$h(x, y, t) = \begin{cases} f_A(x) = 0 \\ f_B(y) = 0 \\ (1-t) \begin{pmatrix} L_A(x) \\ L_B(y) \end{pmatrix} + t(x-y) = 0 \end{cases}$$

starts at the  $\alpha \times \beta$  solutions in  $A \times B \in \mathbb{C}^{n+n}$ .

At  $t=1$ , we find solutions at the diagonal  $x=y$ , in  $A \cap B$ .

# What does diagonal homotopy do? continue...

- 1 Solution pairs are taken to build the start cascade homotopy.
- 2 Embedded hyperplanes are removed one by one to find solutions at the intersection.

$$\#\{x \in \mathbb{C}^n \mid f_A(x) = 0, L_A(x) = 0\} = \alpha$$

$$\#\{y \in \mathbb{C}^n \mid f_B(y) = 0, L_B(y) = 0\} = \beta$$

Assume  $A$  and  $B$  are complete intersections,  $\dim(A \cap B) = 0$

$$h(x, y, t) = \begin{cases} f_A(x) = 0 \\ f_B(y) = 0 \\ (1-t) \begin{pmatrix} L_A(x) \\ L_B(y) \end{pmatrix} + t(x-y) = 0 \end{cases}$$

starts at the  $\alpha \times \beta$  solutions in  $A \times B \in \mathbb{C}^{n+n}$ .

At  $t=1$ , we find solutions at the diagonal  $x=y$ , in  $A \cap B$ .

# Outline

- 1 Problem Statement
- 2 **Introduction**
  - Witness Sets
  - Diagonal Homotopy
  - **Parallel Diagonal Homotopy Implemented in PHCpack**
  - Subsystem-by-Subsystem Solver
- 3 Parallel Implementation of Subsystem-by-Subsystem Solver
  - Algorithms
  - Equipment and Software
  - Experimental Results

# Parallel diagonal homotopy

- Implemented in stages.
- Using extrinsic version of diagonal homotopy.
- Takes pairs of solutions, such as  $(1,3)$ .
- jumpstarting mechanism:
  - 1 The manager compute start solution or reads it from file "just in time" whenever a worker needs a path tracking job.
  - 2 As soon as worker finish tracking a path, the solution is written to file.

# Outline

- 1 Problem Statement
- 2 **Introduction**
  - Witness Sets
  - Diagonal Homotopy
  - Parallel Diagonal Homotopy Implemented in PHCpack
  - **Subsystem-by-Subsystem Solver**
- 3 Parallel Implementation of Subsystem-by-Subsystem Solver
  - Algorithms
  - Equipment and Software
  - Experimental Results

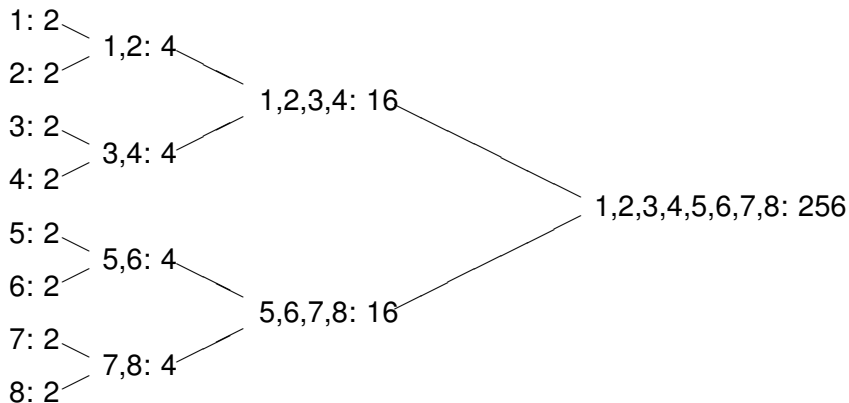
## Previous work

- A.J. Sommese, J. Verschelde, and C.W. Wampler: **Solving Polynomial Systems Equation by Equation**. Accepted for publication in the IMA Volume on Algorithms in Algebraic Geometry.
- Application of diagonal homotopy.
- Equation-by-Equation solver is a limiting case of the subsystem-by-subsystem approach .

# Outline

- 1 Problem Statement
- 2 Introduction
  - Witness Sets
  - Diagonal Homotopy
  - Parallel Diagonal Homotopy Implemented in PHCpack
  - Subsystem-by-Subsystem Solver
- 3 **Parallel Implementation of Subsystem-by-Subsystem Solver**
  - **Algorithms**
  - Equipment and Software
  - Experimental Results

# Divide and Conquer



**Figure:** Schematic overview of solving a system of eight quadrics with an optimal homotopy.

# Data Structure

The triangular state table

1	.....	.....	.....
2	.....	.....	
3	.....		
4	.....		
5			
6			
7			
8			

Queue of jobs

...	job 4	job 3	job 2	job 1
-----	-------	-------	-------	-------

Queue of idle workers

...	worker 3	worker 2	worker 1
-----	----------	----------	----------

One job

2	1	2	1	2	3	4	2
---	---	---	---	---	---	---	---

# Initial Job Distribution

## Manager

broadcast filename →

send data →

receive data ←

## Workers

receive filename

receive data  
 solve equation  
 write to file

send data

*file contains equations*

*data = equation indices  
 terminated with 0*

*synchronization*

# Job Scheduling: main loop

- runs in  $\lceil \log_2^n \rceil$  stages.
- Homotopies in stage  $k$  involve  $2^k$  equations.
- The manager maintains the state table, the job queue, and the queue of idle workers.

# Job Scheduling: main loop, continue...

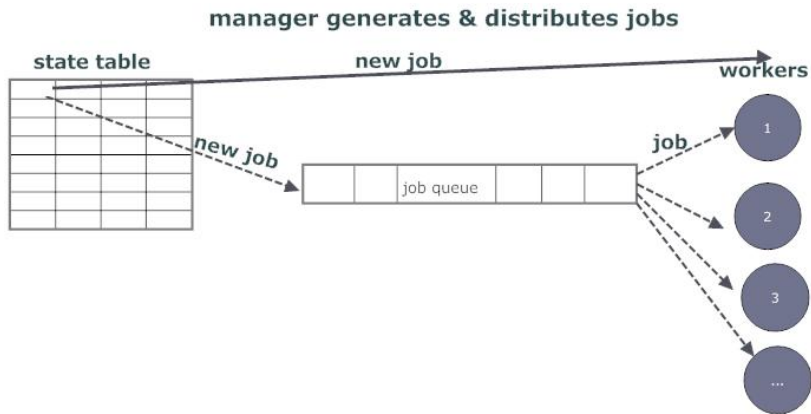


Figure: main loop part 1

# Job Scheduling: main loop, continue...

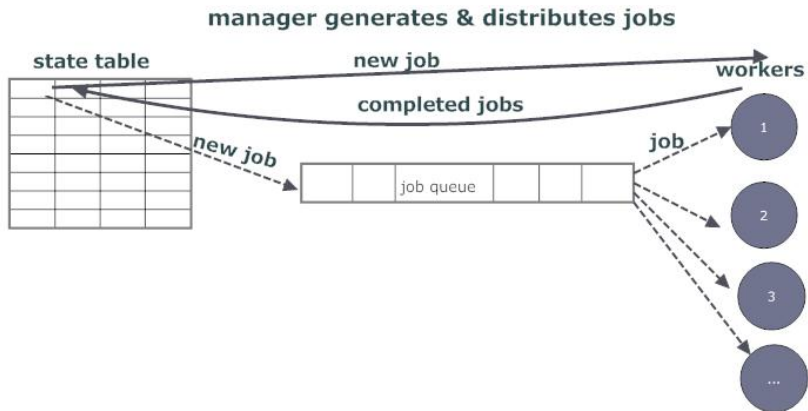


Figure: main loop part 2

# Job Scheduling: main loop, continue...

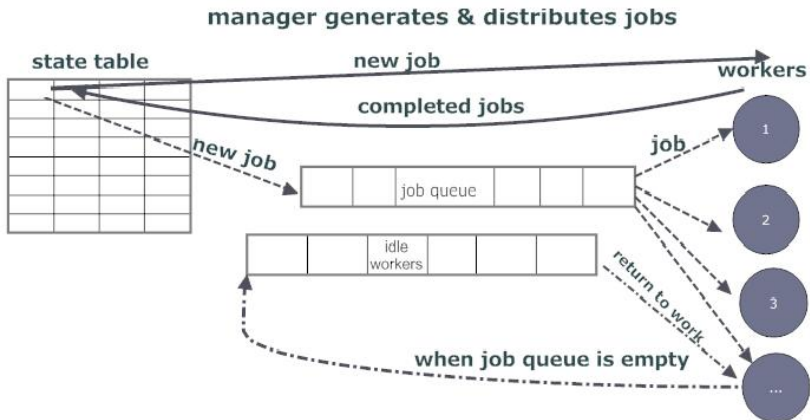


Figure: main loop

# Outline

- 1 Problem Statement
- 2 Introduction
  - Witness Sets
  - Diagonal Homotopy
  - Parallel Diagonal Homotopy Implemented in PHCpack
  - Subsystem-by-Subsystem Solver
- 3 Parallel Implementation of Subsystem-by-Subsystem Solver
  - Algorithms
  - **Equipment and Software**
  - Experimental Results

# Equipment & Software

- Equipment:
  - Personal cluster:
    - One workstation with two dual 2.4Ghz processors, running Linux.
    - Two Rocketcalc clusters: one with four and an other with eight 2.4Ghz processors.
  - NCSA "tungsten" cluster:
    - Running Linux.
    - 1280 3.2GHz processors.
- Software:
  - The main parallel program is written in C, using MPI. Also all job scheduling routines are written in C.
  - The main program is linked to PHC path tracking program.
  - The parallel path trackers follow manager-worker protocol.

# Equipment & Software

- Equipment:
  - Personal cluster:
    - One workstation with two dual 2.4Ghz processors, running Linux.
    - Two Rocketcalc clusters: one with four and an other with eight 2.4Ghz processors.
  - NCSA "tungsten" cluster:
    - Running Linux.
    - 1280 3.2GHz processors.
- Software:
  - The main parallel program is written in C, using MPI. Also all job scheduling routines are written in C.
  - The main program is linked to PHC path tracking program.
  - The parallel path trackers follow manager-worker protocol.

# Outline

- 1 Problem Statement
- 2 Introduction
  - Witness Sets
  - Diagonal Homotopy
  - Parallel Diagonal Homotopy Implemented in PHCpack
  - Subsystem-by-Subsystem Solver
- 3 **Parallel Implementation of Subsystem-by-Subsystem Solver**
  - Algorithms
  - Equipment and Software
  - **Experimental Results**

# Current Results

The complexity of job scheduling is increased for following reasons:

- There must be sufficient points in both witness sets in order to intersect a pair of witness sets.
- New job can be formed only when a pair of witness points are completed.
- The solutions for the second witness set are arriving much slower than those for the first witness set.

# Current Results, continue...

Assume:

- Two witness sets are completed, each is degree of 4.
- Using 5 workers.

## manager

path 1 to node 1           (1,1)  
 path 2 to node 2           (1,2)  
 path 3 to node 3           (1,3)  
 path 4 to node 4           (1,4)  
 resetting 2                 *reset the file 2*  
 path 5 to node 5           (2,1)  
 path 6 to node 2           (2,2)  
 path 7 to node 4           (2,3)  
 path 8 to node 2           (2,4)  
 resetting 2                 *reset the file 2*

## workers

Node 1 receives path 1  
 Node 5 receives path 5  
 Node 2 receives path 2  
 Node 3 receives path 3  
  
 Node 4 receives path 4  
 Node 2 receives path 6  
 Node 4 receives path 7  
 Node 2 receives path 8

## Current Results, continue...

Assume:

- Two witness sets are completed, each is degree of 4.
- Using 5 workers.

### manager

path 1 to node 1             $(1,1)$   
 path 2 to node 2             $(1,2)$   
 path 3 to node 3             $(1,3)$   
 path 4 to node 4             $(1,4)$   
 resetting 2                  *reset the file 2*  
 path 5 to node 5             $(2,1)$   
 path 6 to node 2             $(2,2)$   
 path 7 to node 4             $(2,3)$   
 path 8 to node 2             $(2,4)$   
 resetting 2                  *reset the file 2*

### workers

Node 1 receives path 1  
 Node 5 receives path 5  
 Node 2 receives path 2  
 Node 3 receives path 3  
  
 Node 4 receives path 4  
 Node 2 receives path 6  
 Node 4 receives path 7  
 Node 2 receives path 8

# Conclusion and Future Work

## 1 Conclusion:

- Scheduling of path tracking jobs leads to an almost optimal speedup, using dynamic load balancing.
- jumpstarting mechanism makes a better memory management.

## 2 Future Work:

- Polynomial systems with diverging paths.
- Interlace with parallel monodromy implementation for factoring.
- Taking the components of solutions discovered early in computation.
- Automatic quality control.

# Conclusion and Future Work

## 1 Conclusion:

- Scheduling of path tracking jobs leads to an almost optimal speedup, using dynamic load balancing.
- jumpstarting mechanism makes a better memory management.

## 2 Future Work:

- Polynomial systems with diverging paths.
- Interlace with parallel monodromy implementation for factoring.
- Taking the components of solutions discovered early in computation.
- Automatic quality control.