Parallel Implementation of a Subsystem-by-Subsystem Solver

Yun Guan Jan Verschelde

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
http://www.math.uic.edu/~guan http://www.math.uic.edu/~jan
guan@math.uic.edu jan@math.uic.edu

MACIS 2007 – International Conference on Mathematical Aspects of Computer and Information Sciences
Paris, France, December 5-7, 2007

Outline

Problems we want to solve

Homotopy methods to solve polynomial systems
are "pleasingly parallel":

- the solution paths can be tracked independently;
- scale very well for a large number of processors.

$\rightarrow$ enumerate solutions one after the other

What are we solving?

- large systems in families of benchmark problems such as katsura, economics, adjacent minors;
- systems with more than 100,000 solutions;
- optimal case (no diverging paths).

Problems we want to solve

Homotopy methods to solve polynomial systems
are “pleasingly parallel”:

- the solution paths can be tracked independently;
- scale very well for a large number of processors.

$\rightarrow$ enumerate solutions one after the other

What are we solving?

- large systems in families of benchmark problems such as katsura, economics, adjacent minors;
- systems with more than 100,000 solutions;
- optimal case (no diverging paths).

The Total Degree Homotopy

Solve by considering a simpler system in a homotopy

$$\underbrace{\left(\left\{ \begin{array}{r} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125 x_2^2 - 1.5 = 0 \end{array} \right. \right)}_{\text{target system}} t + \gamma \underbrace{\left(\left\{ \begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array} \right. \right)}_{\text{start system}} (1-t) = \mathbf{0}$$

where t goes from 0 to 1, and $\gamma \in \mathbb{C}$ is a random constant.

For almost all choices of $\gamma \in \mathbb{C}$, every isolated solution of multiplicity *m* is reached by exactly *m* solution paths.

also called "the gamma trick"

If we take $\gamma = 1$, then at $t \approx 0.92$ singular solutions occur.

The Total Degree Homotopy

Solve by considering a simpler system in a homotopy

$$\underbrace{\left(\left\{ \begin{array}{r} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125 x_2^2 - 1.5 = 0 \end{array} \right. \right)}_{\text{target system}} t + \gamma \underbrace{\left(\left\{ \begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array} \right. \right)}_{\text{start system}} (1 - t) = \mathbf{0}$$

where t goes from 0 to 1, and $\gamma \in \mathbb{C}$ is a random constant.

For almost all choices of $\gamma \in \mathbb{C}$, every isolated solution of multiplicity *m* is reached by exactly *m* solution paths.

also called "the gamma trick"

If we take $\gamma = 1$, then at $t \approx 0.92$ singular solutions occur.

The Total Degree Homotopy

Solve by considering a simpler system in a homotopy

$$\underbrace{\left(\left\{\begin{array}{r} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125 x_2^2 - 1.5 = 0 \end{array}\right.\right)}_{\text{target system}} t + \gamma \underbrace{\left(\left\{\begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array}\right.\right)}_{\textbf{start system}} (1 - t) = \mathbf{0}$$

where t goes from 0 to 1, and $\gamma \in \mathbb{C}$ is a random constant.

For almost all choices of $\gamma \in \mathbb{C}$, every isolated solution of multiplicity m is reached by exactly m solution paths.

also called "the gamma trick"

If we take $\gamma = 1$, then at $t \approx 0.92$ singular solutions occur.

The Total Degree Homotopy

Solve by considering a simpler system in a homotopy

$$\underbrace{\left(\left\{ \begin{array}{r} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125 x_2^2 - 1.5 = 0 \end{array} \right. \right)}_{\textbf{target system}} t + \gamma \underbrace{\left(\left\{ \begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array} \right. \right)}_{\textbf{start system}} (1 - t) = \mathbf{0}$$

where t goes from 0 to 1, and $\gamma \in \mathbb{C}$ is a random constant.

For almost all choices of $\gamma \in \mathbb{C}$, every isolated solution of multiplicity *m* is reached by exactly *m* solution paths.

also called "the gamma trick"

If we take $\gamma = 1$, then at $t \approx 0.92$ singular solutions occur.

The Total Degree Homotopy

Solve by considering a simpler system in a homotopy

$$\underbrace{\left(\left\{\begin{array}{r} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125x_2^2 - 1.5 = 0 \end{array}\right.\right)}_{\textbf{target system}} t + \gamma \underbrace{\left(\left\{\begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array}\right.\right)}_{\textbf{start system}} (1-t) = \mathbf{0}$$

where t goes from 0 to 1, and $\gamma \in \mathbb{C}$ is a random constant.

For almost all choices of $\gamma \in \mathbb{C}$, every isolated solution of multiplicity *m* is reached by exactly *m* solution paths.

also called “the gamma trick”

If we take $\gamma = 1$, then at $t \approx 0.92$ singular solutions occur.

The Total Degree Homotopy

Solve by considering a simpler system in a homotopy

$$\underbrace{\left(\left\{ \begin{array}{r} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125 x_2^2 - 1.5 = 0 \end{array} \right. \right)}_{\textbf{target system}} t + \gamma \underbrace{\left(\left\{ \begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array} \right. \right)}_{\textbf{start system}} (1-t) = \mathbf{0}$$

where t goes from 0 to 1, and $\gamma \in \mathbb{C}$ is a random constant.

For almost all choices of $\gamma \in \mathbb{C}$, every isolated solution of multiplicity *m* is reached by exactly *m* solution paths.

also called "the gamma trick"

If we take $\gamma = 1$, then at $t \approx 0.92$ singular solutions occur.

The Total Degree Homotopy

Solve by considering a simpler system in a homotopy

$$\underbrace{\left(\left\{ \begin{array}{r} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125 x_2^2 - 1.5 = 0 \end{array} \right. \right)}_{\text{target system}} t + \gamma \underbrace{\left(\left\{ \begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array} \right. \right)}_{\text{start system}} (1 - t) = \mathbf{0}$$

where t goes from 0 to 1, and $\gamma \in \mathbb{C}$ is a random constant.

For almost all choices of $\gamma \in \mathbb{C}$, every isolated solution of multiplicity *m* is reached by exactly *m* solution paths.

also called "the gamma trick"

If we take $\gamma = 1$, then at $t \approx 0.92$ singular solutions occur.

running total degree homotpies

Katsura systems: n quadrics and one linear equation, #solutions is 2^n. Running on the NCSA machine Tungsten, using p processors, for $n = 20$: 1,048,576 paths:

p	time	min	max
16	22h47m	68,088	70,966
32	9h22m	33,329	34,113
64	4h44m	16,269	16,917
128	2h25m	7,199	8,639
256	1h16m	4,525	3,731

The table lists the total time and the minimum and maximum number of paths tracked by each worker node.
Different homotopy constants cause fluctuations.

running total degree homotpies

Katsura systems: n quadrics and one linear equation, #solutions is 2^n. Running on the NCSA machine Tungsten, using p processors, for $n = 20$: 1,048,576 paths:

p	time	min	max
16	22h47m	68,088	70,966
32	9h22m	33,329	34,113
64	4h44m	16,269	16,917
128	2h25m	7,199	8,639
256	1h16m	4,525	3,731

The table lists the total time and the minimum and maximum number of paths tracked by each worker node.
Different homotopy constants cause fluctuations.

running total degree homotpies

Katsura systems: n quadrics and one linear equation, #solutions is 2^n. Running on the NCSA machine Tungsten, using p processors, for $n = 20$: 1,048,576 paths:

p	time	min	max
16	22h47m	68,088	70,966
32	9h22m	33,329	34,113
64	4h44m	16,269	16,917
128	2h25m	7,199	8,639
256	1h16m	4,525	3,731

The table lists the total time and the minimum and maximum number of paths tracked by each worker node.
Different homotopy constants cause fluctuations.

Witness Sets

- *Numerical representations* of positive dimensional solution sets of polynomial systems.
- A k-dimensional solution set of degree d is represented by
  1. k general hyperplanes; and
  2. d isolated solutions on those k hyperplanes.
- Witness sets are computed either
  1. *top down*: via a cascade of homotopies; or
  2. *bottom up*: diagonal homotopies intersect witness sets.
- Once solution sets of different dimensions are separated as different witness sets, with monodromy and traces we compute ***a numerical irreducible decomposition***.

Witness Sets

- *Numerical representations* of positive dimensional solution sets of polynomial systems.
- A k-dimensional solution set of degree d is represented by
  1. k general hyperplanes; and
  2. d isolated solutions on those k hyperplanes.
- Witness sets are computed either
  1. *top down*: via a cascade of homotopies; or
  2. *bottom up*: diagonal homotopies intersect witness sets.
- Once solution sets of different dimensions are separated as different witness sets, with monodromy and traces we compute ***a numerical irreducible decomposition***.

Witness Sets

- *Numerical representations* of positive dimensional solution sets of polynomial systems.
- A k-dimensional solution set of degree d is represented by
  1. k general hyperplanes; and
  2. d isolated solutions on those k hyperplanes.
- Witness sets are computed either
  1. *top down*: via a cascade of homotopies; or
  2. *bottom up*: diagonal homotopies intersect witness sets.
- Once solution sets of different dimensions are separated as different witness sets, with monodromy and traces we compute ***a numerical irreducible decomposition***.

Witness Sets

- *Numerical representations* of positive dimensional solution sets of polynomial systems.
- A k-dimensional solution set of degree d is represented by
  1. k general hyperplanes; and
  2. d isolated solutions on those k hyperplanes.
- Witness sets are computed either
  1. *top down*: via a cascade of homotopies; or
  2. *bottom up*: diagonal homotopies intersect witness sets.
- Once solution sets of different dimensions are separated as different witness sets, with monodromy and traces we compute ***a numerical irreducible decomposition***.

References for Witness Sets

- A.J. Sommese and C.W. Wampler:
  Numerical algebraic geometry. In *The Mathematics of Numerical Analysis*, pages 749–763, AMS 1996.
- 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, 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.

References for Diagonal Homotopies

- 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.
- A.J. Sommese and C.W. Wampler: **The Numerical Solution of Systems of Polynomials Arising in Engineering and Science**. World Scientific Press, 2005.

What does a diagonal homotopy do?

input/output specification

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

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

Output: witness sets for all pure dimensional components of A∩B

What does a diagonal homotopy do?

input/output specification

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

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

Output: witness sets for all pure dimensional components of A∩B

What does a diagonal homotopy do?

input/output specification

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

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

Output: witness sets for all pure dimensional components of A∩B

What does a diagonal homotopy do?
input/output specification

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

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

Output: witness sets for all pure dimensional components of A∩B

What does diagonal homotopy do?

a special case

1. Solution pairs start a cascade of homotopies.
2. Hyperplanes are removed one by one in the cascade.

Special case: A and B are complete intersections, stored as $\sharp\{\mathbf{x} \in \mathbb{C}^n | f_A(\mathbf{x}) = 0, L_A(\mathbf{x}) = 0\} = \deg(A)$, $\sharp\{\mathbf{y} \in \mathbb{C}^n | f_B(\mathbf{y}) = 0, L_B(\mathbf{y}) = 0\} = \deg(B)$, and $\dim(A \cap B) = 0$, then the diagonal homotopy is

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

starting at the $\deg(A) \times \deg(B)$ solutions in $A \times B \in \mathbb{C}^{n+n}$.

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

What does diagonal homotopy do?

a special case

1. Solution pairs start a cascade of homotopies.
2. Hyperplanes are removed one by one in the cascade.

Special case: A and B are complete intersections, stored as
$\sharp\{\mathbf{x} \in \mathbb{C}^n | f_A(\mathbf{x}) = 0, L_A(\mathbf{x}) = 0\} = \deg(A)$,
$\sharp\{\mathbf{y} \in \mathbb{C}^n | f_B(\mathbf{y}) = 0, L_B(\mathbf{y}) = 0\} = \deg(B)$, and $\dim(A \cap B) = 0$,
then the diagonal homotopy is

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

starting at the $\deg(A) \times \deg(B)$ solutions in $A \times B \in \mathbb{C}^{n+n}$.

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

What does diagonal homotopy do?

a special case

1. Solution pairs start a cascade of homotopies.
2. Hyperplanes are removed one by one in the cascade.

Special case: A and B are complete intersections, stored as
$\sharp\{\mathbf{x} \in \mathbb{C}^n | f_A(\mathbf{x}) = 0, L_A(\mathbf{x}) = 0\} = \deg(A)$,
$\sharp\{\mathbf{y} \in \mathbb{C}^n | f_B(\mathbf{y}) = 0, L_B(\mathbf{y}) = 0\} = \deg(B)$, and $\dim(A \cap B) = 0$,
then the diagonal homotopy is

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

starting at the $\deg(A) \times \deg(B)$ solutions in $A \times B \in \mathbb{C}^{n+n}$.

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

Parallel Diagonal Homotopy

- Runs in various stages: every stage removes one hyperplane in the cascade of homotopies.
- Currently we use the extrinsic version of the diagonal homotopy.
- For memory efficiency, *jumpstarting* homotopy:
  1. The manager computes a start solution or reads it from file "just in time" whenever a worker needs a path tracking job.
  2. As soon as a worker finishes tracking a path, the solution is written to file.

Parallel Diagonal Homotopy

- Runs in various stages: every stage removes one hyperplane in the cascade of homotopies.
- Currently we use the extrinsic version of the diagonal homotopy.
- For memory efficiency, *jumpstarting* homotopy:
  1. The manager computes a start solution or reads it from file "just in time" whenever a worker needs a path tracking job.
  2. As soon as a worker finishes tracking a path, the solution is written to file.

Parallel Diagonal Homotopy

- Runs in various stages: every stage removes one hyperplane in the cascade of homotopies.
- Currently we use the extrinsic version of the diagonal homotopy.
- For memory efficiency, *jumpstarting* homotopy:
  1. The manager computes a start solution or reads it from file "just in time" whenever a worker needs a path tracking job.
  2. As soon as a worker finishes tracking a path, the solution is written to file.

Parallel Diagonal Homotopy

- Runs in various stages: every stage removes one hyperplane in the cascade of homotopies.
- Currently we use the extrinsic version of the diagonal homotopy.
- For memory efficiency, *jumpstarting* homotopy:
  1. The manager computes a start solution or reads it from file "just in time" whenever a worker needs a path tracking job.
  2. As soon as a worker finishes tracking a path, the solution is written to file.

Parallel Diagonal Homotopy

- Runs in various stages: every stage removes one hyperplane in the cascade of homotopies.
- Currently we use the extrinsic version of the diagonal homotopy.
- For memory efficiency, *jumpstarting* homotopy:
  1. The manager computes a start solution or reads it from file "just in time" whenever a worker needs a path tracking job.
  2. As soon as a worker finishes tracking a path, the solution is written to file.

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

An Illustration

Assume two witness sets are completed, each has degree 4.
Using 5 workers:

manager		**workers**
path 1 to node 1	*(1,1)*	node 1 receives path 1
path 2 to node 2	*(1,2)*	node 2 receives path 2
path 3 to node 3	*(1,3)*	node 3 receives path 3
path 4 to node 4	*(1,4)*	node 4 receives path 4
resetting file for witness set 2		
path 5 to node 1	*(2,1)*	node 1 receives path 5
path 6 to node 2	*(2,2)*	node 2 receives path 6
path 7 to node 3	*(2,3)*	node 3 receives path 7
path 8 to node 4	*(2,4)*	node 4 receives path 8

Extension of Previous Work

- A.J. Sommese, J. Verschelde, and C.W. Wampler: **Solving Polynomial Systems Equation by Equation**. To appear in the IMA Volume 146 on *Algorithms in Algebraic Geometry.* Springer, 2007.
- The equation-by-equation solver is a limiting case of the subsystem-by-subsystem approach.
- Here we apply the diagonal homotopy in a more flexible way.

Extension of Previous Work

- A.J. Sommese, J. Verschelde, and C.W. Wampler: **Solving Polynomial Systems Equation by Equation**. To appear in the IMA Volume 146 on *Algorithms in Algebraic Geometry.* Springer, 2007.
- The equation-by-equation solver is a limiting case of the subsystem-by-subsystem approach.
- Here we apply the diagonal homotopy in a more flexible way.

Extension of Previous Work

- A.J. Sommese, J. Verschelde, and C.W. Wampler: **Solving Polynomial Systems Equation by Equation**. To appear in the IMA Volume 146 on *Algorithms in Algebraic Geometry.* Springer, 2007.
- The equation-by-equation solver is a limiting case of the subsystem-by-subsystem approach.
- Here we apply the diagonal homotopy in a more flexible way.

Divide and Conquer

Schematic overview of solving a system of eight quadrics.

1: 2
1,2: 4
2: 2
1,2,3,4: 16
3: 2
3,4: 4
4: 2
1,2,3,4,5,6,7,8: 256
5: 2
5,6: 4
6: 2
5,6,7,8: 16
7: 2
7,8: 4
8: 2

Assume homotopy is optimal: no diverging paths.

Divide and Conquer

Schematic overview of solving a system of eight quadrics.

1: 2
2: 2
1,2: 4

3: 2
4: 2
3,4: 4

1,2,3,4: 16

5: 2
6: 2
5,6: 4

7: 2
8: 2
7,8: 4

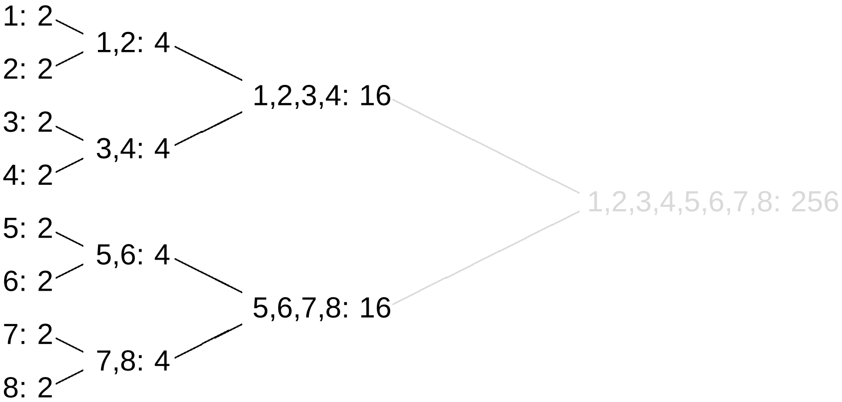
5,6,7,8: 16

1,2,3,4,5,6,7,8: 256

Assume homotopy is optimal: no diverging paths.

Divide and Conquer

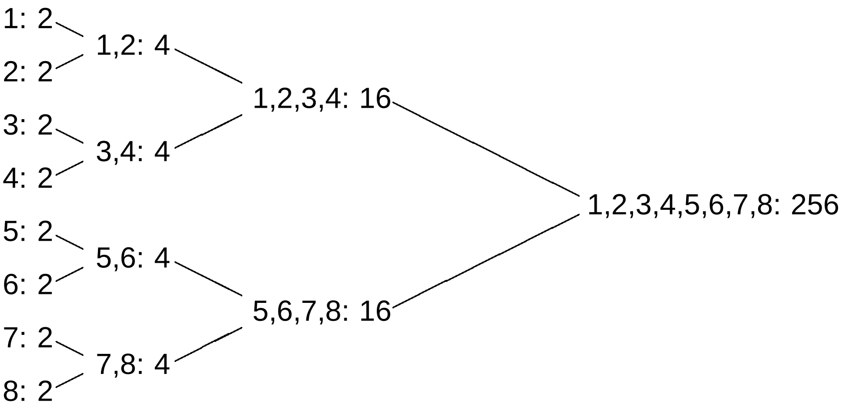
Schematic overview of solving a system of eight quadrics.



Assume homotopy is optimal: no diverging paths.

Divide and Conquer

Schematic overview of solving a system of eight quadrics.



Assume homotopy is optimal: no diverging paths.

Data Structures

The triangular state table

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

of completed jobs

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

Data Structures

The triangular state table

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

of completed jobs

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

Data Structures

The triangular state table

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

of completed jobs

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

Data Structures

The triangular state table

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

of completed jobs

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		**worker**	
broadcast file name	$\rightarrow$	receive file name	*file with equations*
send data	$\rightarrow$	receive data solve equation write to file	*data =* *equation indices* *terminated by 0*
receive data	$\leftarrow$	send data	*synchronization*

Initial Job Distribution

manager		**worker**	
broadcast file name	$\rightarrow$	receive file name	*file with equations*
send data	$\rightarrow$	receive data solve equation write to file	*data =* *equation indices* *terminated by 0*
receive data	$\leftarrow$	send data	*synchronization*

Initial Job Distribution

manager		**worker**	
broadcast file name	$\rightarrow$	receive file name	*file with equations*
send data	$\rightarrow$	receive data solve equation write to file	*data =* *equation indices* *terminated by 0*
receive data	$\leftarrow$	send data	*synchronization*

Initial Job Distribution

manager		**worker**	
broadcast file name	$\rightarrow$	receive file name	*file with equations*
send data	$\rightarrow$	receive data solve equation write to file	*data =* *equation indices* *terminated by 0*
receive data	$\leftarrow$	send data	*synchronization*

Job Scheduling: the main loop

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

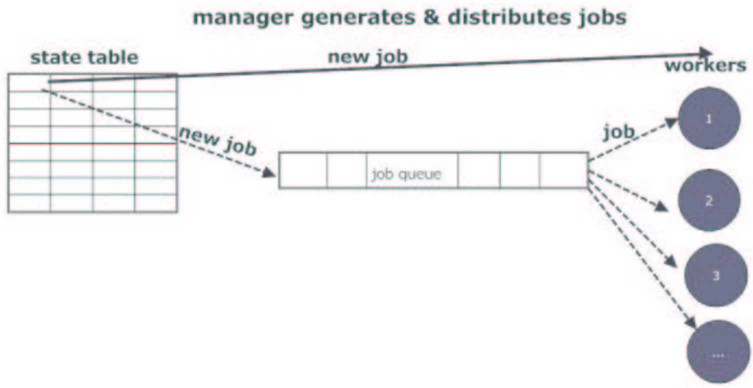
Job Scheduling: the main loop

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

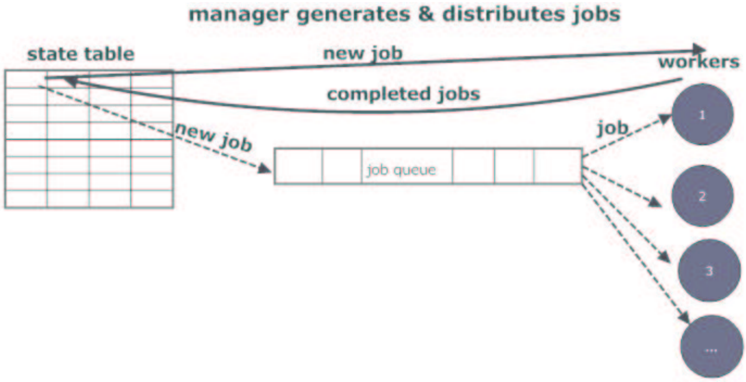
Job Scheduling: the main loop

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

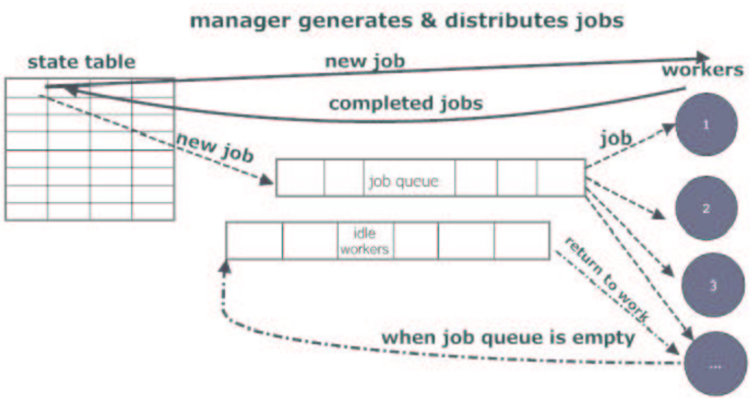
Job Scheduling: main loop, a picture



Job Scheduling: main loop, continued



Job Scheduling: main loop, finally



Software & Equipment

Diagonal homotopies are available in PHCpack.
`http://www.math.uic.edu/~jan/download.html`

1. Parallel code uses and improves sequential versions.
2. `PHClib` forms interface with PHCpack as library.
3. Main parallel programs use MPI for communication.

Computers used:

- Software development on personal cluster:
    1. One workstation with two dual 2.4Ghz processors.
    2. Two Rocketcalc clusters: one with four and an other with eight 2.4Ghz processors.
- NCSA Tungsten cluster is a supercomputer: 1280 3.2GHz processors, running Linux.

Software & Equipment

Diagonal homotopies are available in PHCpack.
`http://www.math.uic.edu/~jan/download.html`

1. Parallel code uses and improves sequential versions.
2. `PHClib` forms interface with PHCpack as library.
3. Main parallel programs use MPI for communication.

Computers used:

- Software development on personal cluster:
  1. One workstation with two dual 2.4Ghz processors.
  2. Two Rocketcalc clusters: one with four and an other with eight 2.4Ghz processors.
- NCSA Tungsten cluster is a supercomputer: 1280 3.2GHz processors, running Linux.

Software & Equipment

Diagonal homotopies are available in PHCpack.
`http://www.math.uic.edu/~jan/download.html`

1. Parallel code uses and improves sequential versions.
2. `PHClib` forms interface with PHCpack as library.
3. Main parallel programs use MPI for communication.

Computers used:

- Software development on personal cluster:
  1. One workstation with two dual 2.4Ghz processors.
  2. Two Rocketcalc clusters: one with four and an other with eight 2.4Ghz processors.
- NCSA Tungsten cluster is a supercomputer: 1280 3.2GHz processors, running Linux.

Software & Equipment

Diagonal homotopies are available in PHCpack.
`http://www.math.uic.edu/~jan/download.html`

1. Parallel code uses and improves sequential versions.
2. `PHClib` forms interface with PHCpack as library.
3. Main parallel programs use MPI for communication.

Computers used:

- Software development on personal cluster:
  1. One workstation with two dual 2.4Ghz processors.
  2. Two Rocketcalc clusters: one with four and an other with eight 2.4Ghz processors.
- NCSA Tungsten cluster is a supercomputer: 1280 3.2GHz processors, running Linux.

Complexity of Job Scheduling

Job scheduling uses dynamic load balancing.
Some additional concerns:

- There must be sufficient points in both witness sets in order to intersect a pair of witness sets.
- New jobs 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.

Synchronization currently prevents optimal speedup.

Complexity of Job Scheduling

Job scheduling uses dynamic load balancing.
Some additional concerns:

- There must be sufficient points in both witness sets in order to intersect a pair of witness sets.
- New jobs 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.

Synchronization currently prevents optimal speedup.

Complexity of Job Scheduling

Job scheduling uses dynamic load balancing.
Some additional concerns:

- There must be sufficient points in both witness sets in order to intersect a pair of witness sets.
- New jobs 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.

Synchronization currently prevents optimal speedup.

Solving katsura8

on a 2.4Ghz Rocketcalc personal cluster

p	time	max	min
2	459s	1,408	1,408
3	277s	787	621
4	175s	514	391
5	140s	375	289
6	104s	307	240
7	98s	251	207
8	86s	218	173
9	85s	193	147
10	81s	167	132
11	72s	152	124
12	68s	147	110

$p = 2$: 1 worker

double #workers (1,2,4,8):
$p = 2 \rightarrow 3 \rightarrow 5 \rightarrow 9$

time: 459s
$\rightarrow$ 277s
$\rightarrow$ 140s
$\rightarrow$ 85s

Solving katsura8
on a 2.4Ghz Rocketcalc personal cluster

p	time	max	min
2	459s	1,408	1,408
3	277s	787	621
4	175s	514	391
5	140s	375	289
6	104s	307	240
7	98s	251	207
8	86s	218	173
9	85s	193	147
10	81s	167	132
11	72s	152	124
12	68s	147	110

$p = 2$: 1 worker

double #workers (1,2,4,8):
$p = 2 \rightarrow 3 \rightarrow 5 \rightarrow 9$

time: 459s
$\rightarrow$ 277s
$\rightarrow$ 140s
$\rightarrow$ 85s

Solving katsura8

on a 2.4Ghz Rocketcalc personal cluster

p	time	max	min
2	459s	1,408	1,408
3	277s	787	621
4	175s	514	391
5	140s	375	289
6	104s	307	240
7	98s	251	207
8	86s	218	173
9	85s	193	147
10	81s	167	132
11	72s	152	124
12	68s	147	110

$p = 2$: 1 worker

double #workers (1,2,4,8):

$p = 2 \rightarrow 3 \rightarrow 5 \rightarrow 9$

time: 459s

$\rightarrow$ 277s

$\rightarrow$ 140s

$\rightarrow$ 85s

Summary

- parallel diagonal homotopy allows jumpstarting for efficient memory management
- dynamic load balancing leads to acceptable speedup
- synchronization along stages gives overhead