A Blackbox Polynomial System Solver on Parallel Shared Memory Computers*

Jan Verschelde
University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
851 S. Morgan Street (m/c 249), Chicago, IL 60607-7045, USA
janv@uic.edu, http://www.math.uic.edu/~jan

June 17, 2018

Abstract

A numerical irreducible decomposition for a polynomial system provides representations for the irreducible factors of all positive dimensional solution sets of the system, separated from its isolated solutions. Homotopy continuation methods are applied to compute a numerical irreducible decomposition. Load balancing and pipelining are techniques in a parallel implementation on a computer with multicore processors. The application of the parallel algorithms is illustrated on solving the cyclic n-roots problems, in particular for $n = 8, 9$, and 12.

Keywords and phrases. Homotopy continuation, numerical irreducible decomposition, mathematical software, multitasking, pipelining, polyhedral homotopies, polynomial system, shared memory parallel computing.

1 Introduction

Almost all computers have multicore processors enabling the simultaneous execution of instructions in an algorithm. The algorithms considered in this paper are applied to solve a polynomial system. Parallel algorithms can often deliver significant speedups on computers with multicore processors.

A blackbox solver implies a fixed selection of algorithms, run with default settings of options and tolerances. The selected methods are homotopy continuation methods to compute a numerical irreducible decomposition of the solution set of a polynomial system. As the solution paths defined by a polynomial homotopy can be tracked independently from each other, there is no communication and no synchronization overhead. Therefore, one may hope that with p threads, the speedup will be close to p.

*This material is based upon work supported by the National Science Foundation under Grant No. 1440534.

The number of paths that needs to be tracked to compute a numerical irreducible decomposition can be a multiple of the number of paths defined by a homotopy to approximate all isolated solutions. Nevertheless, in order to properly distinguish the isolated singular solutions (which occur with multiplicity two or higher) from the solutions on positive dimensional solutions, one needs a representation for the positive dimensional solution sets.

On parallel shared memory computers, the work crew model is applied. In this model, threads are collaborating to complete a queue of jobs. The pointer to the next job in the queue is guarded by a semaphore so only one thread can access the next job and move the pointer to the next job forwards. The design of multithreaded software is described in [17].

The development of the blackbox solver was targeted at the cyclic n-roots systems. Backelin's Lemma [2] states that, if n has a quadratic divisor, then there are infinitely many cyclic n-roots. Interesting values for n are thus 8, 9, and 12, respectively considered in [4], [7], and [16].

Problem Statement. The top down computation of a numerical irreducible decomposition requires first the solving of a system augmented with as many general linear equations as the expected top dimension of the solution set. This first stage is then followed by a cascade of homotopies to compute candidate generic points on lower dimensional solution sets. In the third stage, the output of the cascades is filtered and generic points are classified along their irreducible components. In the application of the work crew model with p threads, the problem is to study if the speedup will converge to p, asymptotically for sufficiently large problems. Another interesting question concerns *quality up*: if we can afford the same computational time as on one thread, then by how much can we improve the quality of the computed results with p threads?

Prior Work. The software used in this paper is PHCpack [20], which provides a numerical irreducible decomposition [18]. For the mixed volume computation, MixedVol [8] and DEMiCs [14] are used. An introduction to the homotopy continuation methods for computing positive dimensional solution sets is described in [19]. The overhead of double double and quad double precision [9] in path trackers can be compensated on multicore workstations by parallel algorithms [21]. The factorization of a pure dimensional solution set on a distributed memory computer with message passing was described in [10].

Related Work. A numerical irreducible decomposition can be computed by a program described in [3], but that program lacks polyhedral homotopies, needed to efficiently solve sparse polynomial systems such as the cyclic n-roots problems. Parallel algorithms for mixed volumes and polyhedral homotopies were presented in [5, 6]. The computation of the positive dimensional solutions for the cyclic 12-roots problem was reported first in [16]. A recent parallel implementation of polyhedral homotopies was announced in [13].

Contributions and Organization. The next section proposes the application of pipelining to interleave the computation of mixed cells with the tracking of solution paths to solve a random coefficient system. The production rate of mixed cells relative to the cost of path tracking is related to the pipeline latency. The third section describes the second stage in the solver and exam-

ines the speedup for tracking paths defined by sequences of homotopies. In section four, the speedup of the application of the homotopy membership test is defined. One outcome of this research is free and open software to compute a numerical irreducible decomposition on parallel shared memory computers. Computational experiments with the software are presented in section five.

2 Solving the Top Dimensional System

There is only one input to the blackbox solver: the expected top dimension of the solution set. This input may be replaced by the number of variables minus one. However, entering an expected top dimension that is too high may lead to a significant computational overhead.

2.1 Random Hyperplanes and Slack Variables

A system is called *square* if it has as many equations as unknowns. A system is *underdetermined* if it has fewer equations than unknowns. An underdetermined system can be turned into square system by adding as many linear equations with randomly generated complex coefficients as the difference between the number of unknowns and equations. A system is *overdetermined* if there are more equations than unknowns. To turn an overdetermined system into a square one, add repeatedly to every equation in the overdetermined system a random complex constant multiplied by a new slack variable, repeatedly until the total number of variables equals the number of equations.

The top dimensional system is the given polynomial system, augmented with as many linear equations with randomly generated complex coefficients as the expected top dimension. To the augmented system as many slack variables are added as the expected top dimension. The result of adding random linear equations and slack variables is called an *embedded* system. Solutions of the embedded system with zero slack variables are generic points on the top dimensional solution set. Solutions of the embedded system with nonzero slack variables are start solutions in cascades of homotopies to compute generic points on lower dimensional solution sets.

Example 2.1. *(embedding a system)* The equations for the cyclic 4-roots problem are

$$\mathbf{f}(\mathbf{x}) = \begin{cases} x_1 + x_2 + x_3 + x_4 = 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 = 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_1 + x_4x_1x_2 = 0 \\ x_1x_2x_3x_4 - 1 = 0. \end{cases} \tag{1}$$

The expected top dimension equals one. The system is augmented by one linear

equation and one slack variable z_1. The embedded system is then the following:

$$E_1(\mathbf{f}(\mathbf{x}), z_1) = \left\{ \begin{array}{rcl} x_1 + x_2 + x_3 + x_4 + \gamma_1 z_1 & = & 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 + \gamma_2 z_1 & = & 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_1 + x_4x_1x_2 + \gamma_3 z_1 & = & 0 \\ x_1x_2x_3x_4 - 1 + \gamma_4 z_1 & = & 0 \\ c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + z_1 & = & 0. \end{array} \right. \quad (2)$$

The constants γ_1, γ_2, γ_3, γ_4 and c_0, c_1, c_2, c_3, c_4 are randomly generated complex numbers.

The system $E_1(\mathbf{f}(\mathbf{x}), z_1) = \mathbf{0}$ has 20 solutions. Four of those 20 solutions have a zero value for the slack variable z_1. Those four solutions satisfy thus the system

$$E_1(\mathbf{f}(\mathbf{x}), 0) = \left\{ \begin{array}{c} x_1 + x_2 + x_3 + x_4 = 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 = 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_1 + x_4x_1x_2 = 0 \\ x_1x_2x_3x_4 - 1 = 0 \\ c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 = 0. \end{array} \right. \quad (3)$$

By the random choice of the constants c_0, c_1, c_2, c_3, and c_4, the four solutions are generic points on the one dimensional solution set. Four equals the degree of the one dimensional solution set of the cyclic 4-roots problem.

For systems with sufficiently general coefficients, polyhedral homotopies are generically optimal in the sense that no solution path diverges. Therefore, the default choice to solve the top dimensional system is the computation of a mixed cell configuration and the solving of a random coefficient start system. Tracking the paths to solve the random coefficient start system is a pleasingly parallel computation, which with dynamic load balancing will lead to a close to optimal speedup.

2.2 Pipelined Polyhedral Homotopies

The computation of all mixed cells is harder to run in parallel, but fortunately the mixed volume computation takes in general less time than the tracking of all solution paths and, more importantly, the mixed cells are not obtained all at once at the end, but are produced in sequence, one after the other. As soon as a cell is available, the tracking of as many solution paths as the volume of the cell can start. Figure 1 illustrates a 2-stage pipeline with p threads.

Figure 2 illustrates the application of pipelining to the solving of a random coefficient system where the subdivision of the Newton polytopes has six cells. The six cells are computed by the first thread. The other three threads take the cells and run polyhedral homotopies to compute as many solutions as the volume of the corresponding cell.

Counting the horizontal span of time units in Figure 2, the total time equals 9 units. In the corresponding sequential process, it takes 24 time units. This particular pipeline with 4 threads gives a speedup of $24/9 \approx 2.67$.

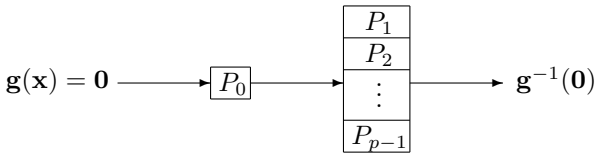

Figure 1: A 2-stage pipeline with thread P_0 in the first stage to compute the cells to solve the start systems with paths to be tracked in the second stage by $p-1$ threads P_1, P_2, ..., P_{p-1}. The input to the pipeline is a random coefficient system $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ and the output are its solutions in the set $\mathbf{g}^{-1}(\mathbf{0})$.

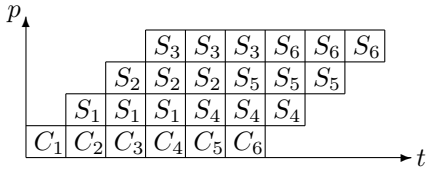

Figure 2: A space time diagram for a 2-stage pipeline with one thread to produce 6 cells C_1, C_2, ..., C_6 and 3 threads to solve the corresponding 6 start systems S_1, S_2, ..., S_6. For regularity, it is assumed that solving one start system takes three times as many time units as it takes to produce one cell.

2.3 Speedup

As in Figure 1, consider a scenario with p threads:

- the first thread produces n cells; and
- the other $p-1$ threads track all paths corresponding to the cells.

Assume that tracking all paths for one cell costs F times the amount of time it takes to produce that one cell. In this scenario, the sequential time T_1, the parallel time T_p, and the speedup S_p are defined by the following formulas:

$$T_1 = n + Fn, \quad T_p = p - 1 + \frac{Fn}{p-1}, \quad S_p = \frac{T_1}{T_p} = \frac{n(1+F)}{p-1+\frac{Fn}{p-1}}. \tag{4}$$

The term $p-1$ in T_p is *the pipeline latency*, the time it takes to fill up the pipeline with jobs. After this latency, the pipeline works at full speed.

The formula for the speedup S_p in (4) is rather too complicated for direct interpretation. Let us consider a special case. For large problems, the number n of cells is larger than the number p of threads, $n \gg p$. For a fixed number p of threads, let n approach infinity. Then an optimal speedup is achieved, if the pipeline latency $p-1$ equals the multiplier factor F in the tracking of all paths relative to the time to produce one cell. This observation is formalized in the following theorem.

Theorem 2.2. *If $F = p - 1$, then $S_p = p$ for $n \to \infty$.*

Proof. For $F = p - 1$, $T_1 = np$ and $T_p = n + p - 1$. Then, letting $n \to \infty$,

$$\lim_{n\to\infty} S_p = \lim_{n\to\infty} \frac{T_1}{T_p} = \lim_{n\to\infty} \frac{np}{n+p-1} = p. \qquad \square \tag{5}$$

□

In case the multiplier factor is larger than the pipeline latency, if $F > p-1$, then the first thread will finish sooner with its production of cells and remains idle for some time. If $p \gg 1$, then having one thread out of many idle is not bad. The other case, if tracking all paths for one cell is smaller than the pipeline latency, if $F < p - 1$, is worse as many threads will be idle waiting for cells to process.

The above analysis applies to pipelined polyhedral homotopies to solve a random coefficient system. Consider the solving of the top dimensional system.

Corollary 2.3. *Let F be the multiplier factor in the cost of tracking the paths to solve the start system, relative to the cost of computing the cells. If the pipeline latency equals F, then the speedup to solve the top dimensional system with p threads will asymptotically converge to p, as the number of cells goes to infinity.*

Proof. Solving the top dimensional system consists in two stages. The first stage, solving a random coefficient system, is covered by Theorem 2.2. In the second stage, the solutions of the random coefficient system are the start solutions in a homotopy to solve the top dimensional system. This second stage is a pleasingly parallel computation as the paths can be tracked independently from each and for which the speedup is close to optimal for sufficiently large problems. □ □

3 Computing Lower Dimensional Solution Sets

The solution of the top dimensional system is an important first stage, which leads to the top dimensional solution set, provided the given dimension on input equals the top dimension. This section describes the second stage in a numerical irreducible decomposition: the computation of candidate generic points on the lower dimensional solution sets.

3.1 Cascades of Homotopies

The solutions of an embedded system with nonzero slack variables are regular solutions and serve as start solutions to compute sufficiently many generic points on the lower dimensional solution sets. The sufficiently many in the sentence above means that there will be at least as many generic points as the degrees of the lower dimensional solution sets.

Example 3.1. *(a system with a 3-stage cascade of homotopies)* Consider the following system:

$$\mathbf{f}(\mathbf{x}) = \begin{cases} (x_1-1)(x_1-2)(x_1-3)(x_1-4) = 0 \\ (x_1-1)(x_2-1)(x_2-2)(x_2-3) = 0 \\ (x_1-1)(x_1-2)(x_3-1)(x_3-2) = 0 \\ (x_1-1)(x_2-1)(x_3-1)(x_4-1) = 0. \end{cases} \tag{6}$$

In its factored form, the numerical irreducible decomposition is apparent. First, there is the three dimensional solution set defined by $x_1 = 1$. Second, for $x_1 = 2$, observe that $x_2 = 1$ defines a two dimensional solution set and four lines: $(2, 2, x_3, 1)$, $(2, 2, 1, x_4)$, $(2, 3, 1, x_4)$, and $(2, 3, x_3, 1)$. Third, for $x_1 = 3$, there are four lines: $(3, 1, 1, x_4)$, $(3, 1, 2, x_4)$, $(3, 2, 1, x_4)$, $(3, 3, 1, x_4)$, and two isolated points $(3, 2, 2, 1)$ and $(3, 3, 2, 1)$. Fourth, for $x_1 = 4$, there are four lines: $(4, 1, 1, x_4)$, $(4, 1, 2, x_4)$, $(4, 2, 1, x_4)$, $(4, 3, 1, x_4)$, and two additional isolated solutions $(4, 3, 2, 1)$ and $(4, 2, 2, 1)$.

Sorted then by dimension, there is one three dimensional solution set, one two dimensional solution set, twelve lines, and four isolated solutions.

The top dimensional system has three random linear equations and three slack variables z_1, z_2, and z_3. The mixed volume of the top dimensional system equals 61 and this is the number of paths tracked in its solution. Of those 61 paths, 6 diverge to infinity and the cascade of homotopies starts with 55 paths. The number of paths tracked in the cascade is summarized at the right in Figure 3.

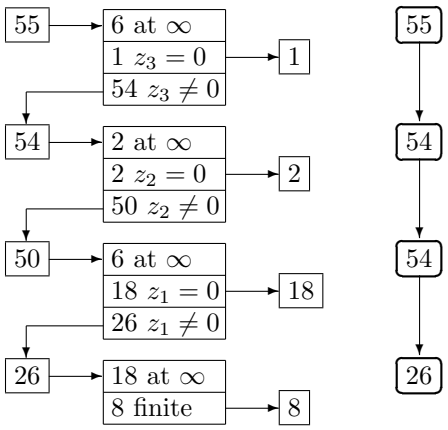

Figure 3: At the left are the numbers of paths tracked in each stage of the computation of a numerical irreducible decomposition of $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ in (6). The numbers at the right are the *candidate* generic points on each positive dimensional solution set, or in case of the rightmost 8 at the bottom, the number of *candidate* isolated solutions. Shown at the farthest right is the summary of the number of paths tracked in each stage of the cascade.

The number of solutions with nonzero slack variables remains constant in each run, because those solutions are regular. Except for the top dimensional system, the number of solutions with slack variables equal to zero fluctuates, each time different random constants are generated in the embedding, because such solutions are highly singular.

The right of Figure 3 shows the order of computation of the path tracking jobs, in four stages, for each dimension of the solution set. The obvious parallel implementation is to have p threads collaborate to track all paths in that stage.

3.2 Speedup

The following analysis assumes that every path has the same difficulty and requires the same amount of time to track.

Theorem 3.2. *Let T_p be the time it takes to track n paths with p threads. Then, the optimal speedup S_p is*

$$S_p = p - \frac{p-r}{T_p}, \quad r = n \bmod p. \tag{7}$$

If $n < p$, then $S_p = n$.

Proof. Assume it takes one time unit to track one path. The time on one thread is then $T_1 = n = qp + r$, $q = \lfloor n/p \rfloor$ and $r = n \bmod p$. As $r < p$, the tracking of r paths with p threads takes one time unit, so $T_p = q + 1$. Then the speedup is

$$S_p = \frac{T_1}{T_p} = \frac{qp+r}{q+1} = \frac{qp+p-p+r}{q+1} = \frac{qp+p}{q+1} - \frac{p-r}{q+1} = p - \frac{p-r}{T_p}. \tag{8}$$

If $n < p$, then $q = 0$ and $r = n$, which leads to $S_p = n$. □ □

In the limit, as $n \to \infty$, also $T_p \to \infty$, then $(p-r)/T_p \to 0$ and so $S_p \to p$. For a cascade with $D+1$ stages, Theorem 3.2 can be generalized as follows.

Corollary 3.3. *Let T_p be the time it takes to track with p threads a sequence of n_0, n_1, …, n_D paths. Then, the optimal speedup S_p is*

$$S_p = p - \frac{dp - r_0 - r_1 - \cdots - r_D}{T_p}, \quad r_k = n_k \bmod p, k = 0, 1, \ldots D. \tag{9}$$

Proof. Assume it takes one time unit to track one path. The time on one thread is then

$$T_1 = n_0 + n_1 + \cdots + n_D = q_0 p + r_0 + q_1 p + r_1 + \cdots + q_D p + r_D, \tag{10}$$

where $q_k = \lfloor n_k/p \rfloor$ and $r_k = n_k \bmod p$, for $k = 0, 1, \ldots, D$. As $r_k < p$, the tracking of r_k paths with p threads takes $D+1$ time units, so the time on p threads is

$$T_p = q_0 + q_1 + \cdots + q_D + D + 1. \tag{11}$$

Then the speedup is

$$S_p = \frac{T_1}{T_p} = \frac{pT_p - dp + r_0 + r_1 + \cdots + r_D}{T_p} \quad (12)$$

$$= p - \frac{dp - r_0 - r_1 - \cdots - r_D}{T_p}. \quad \square \quad (13)$$

$\square$

If the length $D+1$ of the sequence of paths is long and the number of paths in each stage is less than p, then the speedup will be limited.

4 Filtering Lower Dimensional Solution Sets

Even if one is interested only in the isolated solutions of a polynomial system, one would need to be able to distinguish the isolated multiple solutions from solutions on a positive dimensional solution set. Without additional information, both an isolated multiple solution and a solution on a positive dimensional set appear numerically as singular solutions, that is: as solutions where the Jacobian matrix does not have full rank. A homotopy membership test makes this distinction.

4.1 Homotopy Membership Tests

Example 4.1. *(homotopy membership test)* Consider the following system:

$$\mathbf{f}(\mathbf{x}) = \begin{cases} (x_1 - 1)(x_1 - 2) = 0 \\ (x_1 - 1)x_2^2 = 0. \end{cases} \quad (14)$$

The solution consists of the line $x_1 = 1$ and the isolated point $(2, 0)$ which occurs with multiplicity two. The line $x_1 = 1$ is represented by one generic point as the solution of the embedded system

$$E(\mathbf{f}(\mathbf{x}), z_1) = \begin{cases} (x_1 - 1)(x_1 - 2) + \gamma_1 z_1 = 0 \\ (x_1 - 1)x_2^2 + \gamma_2 z_1 = 0 \\ c_0 + c_1 x_1 + c_2 x_2 + z_1 = 0, \end{cases} \quad (15)$$

where the constants γ_1, γ_2, c_0, c_1, and c_2 are randomly generated complex numbers. Replacing the constant c_0 by $c_3 = -2c_1$ makes that the point $(2, 0, 0)$ satisfies the system $E(\mathbf{f}(\mathbf{x}), z_1) = \mathbf{0}$. Consider the homotopy

$$\mathbf{h}(\mathbf{x}, z_1, t) = \begin{cases} (x_1 - 1)(x_1 - 2) + \gamma_1 z_1 = 0 \\ (x_1 - 1)x_2^2 + \gamma_2 z_1 = 0 \\ (1 - t)c_0 + tc_3 + c_1 x_1 + c_2 x_2 + z_1 = 0. \end{cases} \quad (16)$$

For $t = 0$, there is the generic point on the line $x_1 = 1$ as a solution of the system (15). Tracking one path starting at the generic point to $t = 1$ moves the generic point to another generic point on $x_1 = 1$. If that other generic point at $t = 1$ coincides with the point $(2, 0, 0)$, then the point $(2, 0)$ belongs to the line. Otherwise, as is the case in this example, it does not.

In running the homotopy membership test, a number of paths need to be tracked. To identify the bottlenecks in a parallel version, consider the output of Figure 3 in the continuation of the example on the system in 6.

Example 4.2. *(Example 3.1 continued)* Assume the spurious points on the higher dimensional solution sets have already been removed so there is one generic point on the three dimensional solution set, one generic point on the two dimensional solution set, and twelve generic points on the one dimensional solution set.

At the end of the cascade, there are eight candidate isolated solutions. Four of those eight are regular solutions and are thus isolated. The other four solutions are singular. Singular solutions may be isolated multiple solutions, but could also belong to the higher dimensional solution sets. Consider Figure 4.

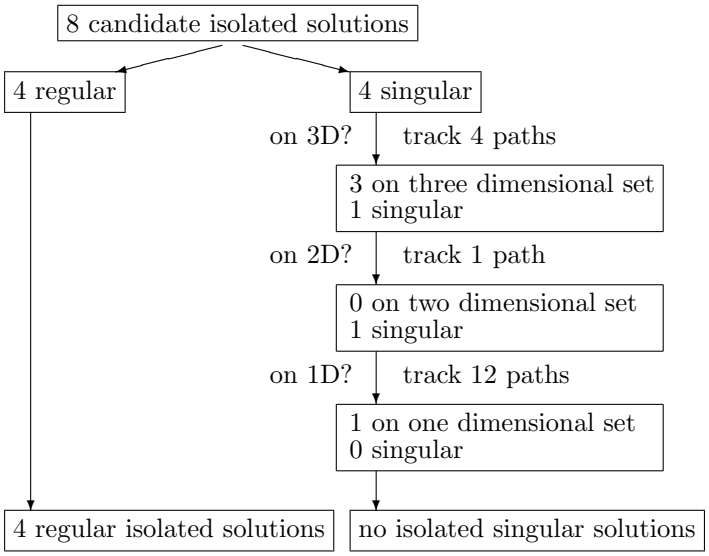

Figure 4: Stages in testing whether the singular candidate isolated points belong to the higher dimensional solution sets.

Executing the homotopy membership tests as in in Figure 4, first on 3D, then on 2D, and finally on 1D, the bottleneck occurs in the middle, where there is only one path to track.

Figure 5 is the continuation of Figure 3: the output of the cascade shown in Figure 3 is the input of the filtering in Figure 5. Figure 4 explains the last stage in Figure 5.

4.2 Speedup

The analysis of the speedup is another consequence of Theorem 3.2.

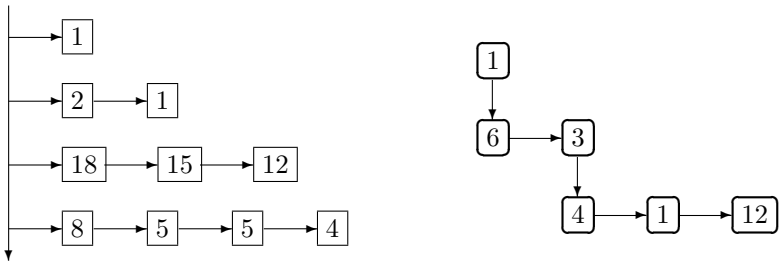

Figure 5: On input are the candidate generic points shown as output in Figure 3: 1 point at dimension three, 2 points at dimension two, 18 points at dimension one, and 8 candidate isolated points. Points on higher dimensional solution sets are removed by homotopy membership filters. The numbers at the right equal the number of paths in each stage of the filters. The sequence 4, 1, 12 at the bottom is explained in Figure 4.

Corollary 4.3. *Let T_p be the time it takes to filter n_D, n_{D-1}, ..., $n_{\ell+1}$ singular points on components respectively of dimensions D, $D-1$, ..., $\ell+1$ and degrees d_D, d_{D-1}, ..., $d_{\ell+1}$. Then, the optimal speedup is*

$$S_p = p - \frac{(D-\ell)p - r_D - r_{D-1} - \cdots - r_{\ell+1}}{T_p}, \quad r_k = (n_k d_k) \bmod p, \qquad (17)$$

for $k = \ell+1, \ldots, D-1, D$.

Proof. For a component of degree d_k, it takes $n_k d_k$ paths to filter n_k singular points. The statement in (17) follows from replacing n_k by $n_k d_k$ in the statement in (9) of Corollary 3.3. □ □

Although the example shown in Figure 5 is too small for parallel computation, it illustrates the law of diminishing returns in introducing parallelisms. There are two reasons for a reduced parallelism:

1. The number of singular solutions and the degrees of the solution sets could be smaller than the number of available cores.

2. In a cascade of homotopies, there are as many steps as $D+1$, where D is the expected top dimension. To filter the output of the cascade, there are $D(D+1)/2$ stages, so longer sequences of homotopies are considered.

Singular solutions that do not lie on any higher positive dimensional solution set need to be processed further by deflation [11, 12], not available yet in a multithreaded implementation. Parallel algorithms to factor the positive dimensional solutions into irreducible factors are described in [10].

5 Computational Experiments

The software was developed on a Mac OS X laptop and Linux workstations. The executable for Windows also supports multithreading. All times reported below are on a CentOS Linux 7 computer with two Intel Xeon E5-2699v4 Broadwell-EP 2.20 GHz processors, which each have 22 cores, 256 KB L2 cache and 55 MB L3 cache. The memory is 256 MB, in 8 banks of 32 MB at 2400 MHz. As the processors support hyperthreading, speedups of more than 44 are possible.

On Linux, the executable `phc` is compiled with the GNAT GPL 2016 edition of the gnu-ada compiler. The thread model is posix, in gcc version 4.9.4. The code in PHCpack contains an Ada translation of the MixedVol Algorithm [8], The source code for the software is at github, licensed under GNU GPL version 3. The blackbox solver for a numerical irreducible decomposition is called as `phc -B` and with `p` threads: as `phc -B -tp`. With `phc -B2` and `phc -B4`, computations happen respectively in double double and quad double arithmetic [9].

5.1 Solving Cyclic 8 and Cyclic 9-Roots

Both cyclic 8 and cyclic 9-roots are relatively small problems, relative compared to the cyclic 12-roots problem. Table 1 summarizes wall clock times and speedups for runs on the cyclic 8 and 9-roots systems. The wall clock time is the real time, elapsed since the start and the end of each run. This includes the CPU time, system time, and is also influenced by other jobs the operating system is running.

	cyclic 8-roots		cyclic 9-roots	
p	seconds	speedup	seconds	speedup
1	181.765	1.00	2598.435	1.00
2	167.871	1.08	1779.939	1.46
4	89.713	2.03	901.424	2.88
8	47.644	3.82	427.800	6.07
16	32.215	5.65	267.838	9.70
32	22.182	8.19	153.353	16.94
64	20.103	9.04	150.734	17.24

Table 1: Wall clock times in seconds with `phc -B -tp` for `p` threads.

With 64 threads the time for cyclic 8-roots reduces from 3 minutes to 20 seconds and for cyclic 9-roots from 43 minutes to 2 minutes and 30 seconds. Table 2 summarizes the wall clock times with 64 threads in higher precision.

5.2 Solving Cyclic 12-Roots on One Thread

The classical Bézout bound for the system is 479,001,600. This is lowered to 342,875,319 with the application of a linear-product start system. In contrast, the mixed volume of the embedded cyclic 12-roots system equals 983,952.

	cyclic 8-roots			cyclic 9-roots		
	seconds	=	hms format	seconds	=	hms format
dd	53.042	=	53s	498.805	=	8m19s
qd	916.020	=	15m16s	4761.258	=	1h19m21s

Table 2: Wall clock times with 64 threads in double double and quad double precision.

The wall clock time on the blackbox solver on one thread is about 95 hours (almost 4 days). This run includes the computation of the linear-product bound which takes about 3 hours. This computation is excluded in the parallel version because the multithreaded version overlaps the mixed volume computation with polyhedral homotopies. While a speedup of about 30 is not optimal, the time reduces from 4 days to less than 3 hours with 64 threads, see Table 3.

The blackbox solver does not exploit symmetry, see [1] for such exploitation.

5.3 Pipelined Polyhedral Homotopies

This section concerns the computation of a random coefficient start system used in a homotopy to solve the top dimensional system, to start the cascade homotopies for the cyclic 12-roots system. Table 3 summarizes the wall clock times to solve a random coefficient start system to solve the top dimensional system.

p	seconds	hms format	speedup	total seconds	hms format	percentage
2	62812.764	17h26m52s	1.00	157517.816	43h45m18s	39.88%
4	21181.058	5h53m01s	2.97	73088.635	20h18m09s	28.98%
8	8932.512	2h28m53s	7.03	38384.005	10h39m44s	23.27%
16	4656.478	1h17m36s	13.49	19657.329	5h27m37s	23.69%
32	4200.362	1h10m01s	14.95	12154.088	3h22m34s	34.56%
64	4422.220	1h13m42s	14.20	9808.424	2h43m28s	45.08%

Table 3: Times of the pipelined polyhedral homotopies versus the total time in the solver `phc -B -tp`, for increasing values 2, 4, 8, 16, 32, 64 of the tasks `p`.

For pipelining, we need at least 2 tasks: one to produce the mixed cells and another to track the paths. The speedup of `p` tasks is computed over 2 tasks. With 16 threads, the time to solve a random coefficient system is reduced from 17.43 hours to 1.17 hour. The second part of Table 3 lists the time of solving the random coefficient system relative to the total time of the solver. For 2 threads, solving the random coefficient system takes almost 40% of the total time and then decreases to less than 24% of the total time with 16 threads. Already for 16 threads, the speedup of 13.49 indicates that the production of mixed cells cannot keep up with the pace of tracking the paths.

Dynamic enumeration [15] applies a greedy algorithm to compute all mixed cells and its implementation in DEMiCs [14] produces the mixed cells at a faster pace than MixedVol [8]. Table 4 shows times for the mixed volume computation with DEMiCs [14] in a pipelined version of the polyhedral homotopies.

p	seconds	=	hms format	speedup
2	56614	=	15h43m34s	1.00
4	21224	=	5h53m44s	2.67
8	9182	=	2h23m44s	6.17
16	4627	=	1h17m07s	12.24
32	2171	=	36m11s	26.08
64	1989	=	33m09s	28.46

Table 4: Times of the pipelined polyhedral homotopies with DEMiCs, for increasing values 2, 4, 8, 16, 32, 64 of tasks p. The last time is an average over 13 runs. With 64 threads the times ranged between 23 minutes and 47 minutes.

5.4 Solving the Cyclic 12-Roots System in Parallel

As already shown in Table 3, the total time with 2 threads goes down from more than 43 hours to less than 3 hours, with 64 threads. Table 5 provides a detailed breakup of the wall clock times for each stage in the solver.

	solving top system			cascade and filter			grand	
p	start	contin	total	cascade	filter	total	total	speedup
2	62813	47667	110803	44383	2331	46714	157518	1.00
4	21181	25105	46617	24913	1558	26471	73089	2.16
8	8933	14632	23896	13542	946	14488	38384	4.10
16	4656	7178	12129	6853	676	7529	19657	8.01
32	4200	3663	8094	3415	645	4060	12154	12.96
64	4422	2240	7003	2228	557	2805	9808	16.06

Table 5: Wall clock times in seconds for all stages of the solver on cyclic 12-roots. The solving of the top dimension system breaks up in two stages: the solving of a start system (start) and the continuation to the solutions of the the top dimensional system (contin). Speedups are good in the cascade stage, but the filter stage contains also the factorization in irreducible components, which does not run in parallel.

A run in double double precision with 64 threads ends after 7 hours and 37 minutes. This time lies between the times in double precision with 8 threads, 10 hours and 39 minutes, and with 16 threads, 5 hours and 27 minutes (Table 3). Confusing quality with precision, from 8 to 64 threads, the working precision can be doubled with a reduction in time by 3 hours, from 10.5 hours to 7.5 hours.

References

[1] D. Adrovic and J. Verschelde. Polyhedral methods for space curves exploiting symmetry applied to the cyclic n-roots problem. In V.P. Gerdt, W. Koepf, E.W. Mayr, and E.V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing, 15th International Workshop (CASC 2013)*, volume 8136 of *Lecture Notes in Computer Science*, pages 10–29. Springer-Verlag, 2013.

[2] J. Backelin. Square multiples n give infinitely many cyclic n-roots. Reports, Matematiska Institutionen 8, Stockholms universitet, 1989.

[3] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Software for numerical algebraic geometry: a paradigm and progress towards its implementation. In M.E. Stillman, N. Takayama, and J. Verschelde, editors, *Software for Algebraic Geometry*, volume 148 of *The IMA Volumes in Mathematics and its Applications*, pages 33–46. Springer-Verlag, 2008.

[4] G. Björck and R. Fröberg. Methods to "divide out" certain solutions from systems of algebraic equations, applied to find all cyclic 8-roots. In M. Gyllenberg and L.E. Persson, editors, *Analysis, Algebra and Computers in Math. research*, volume 564 of *Lecture Notes in Mathematics*, pages 57–70. Dekker, 1994.

[5] T. Chen, T.-L. Lee, and T.-Y. Li. Hom4PS-3: a parallel numerical solver for systems of polynomial equations based on polyhedral homotopy continuation methods. In H. Hong and C. Yap, editors, *Mathematical Software – ICMS 2014, 4th International Conference, Proceedings*, volume 8592 of *Lecture Notes in Computer Science*, pages 183–190. Springer-Verlag, 2014.

[6] T. Chen, T.L. Lee, and T.Y. Li. Mixed volume computation in parallel. *Taiwanese Journal of Mathematics*, 18(1):93–114, 2014.

[7] J.C. Faugère. Finding all the solutions of Cyclic 9 using Gröbner basis techniques. In *Computer Mathematics - Proceedings of the Fifth Asian Symposium (ASCM 2001)*, volume 9 of *Lecture Notes Series on Computing*, pages 1–12. World Scientific, 2001.

[8] T. Gao, T.Y. Li, and M. Wu. Algorithm 846: MixedVol: a software package for mixed-volume computation. *ACM Trans. Math. Softw.*, 31(4):555–560, 2005.

[9] Y. Hida, X.S. Li, and D.H. Bailey. Algorithms for quad-double precision floating point arithmetic. In *15th IEEE Symposium on Computer Arithmetic (Arith-15 2001)*, pages 155–162. IEEE Computer Society, 2001.

[10] A. Leykin and J. Verschelde. Decomposing solution sets of polynomial systems: a new parallel monodromy breakup algorithm. *The International Journal of Computational Science and Engineering*, 4(2):94–101, 2009.

[11] A. Leykin, J. Verschelde, and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theor. Comput. Sci.*, 359(1-3):111–122, 2006.

[12] A. Leykin, J. Verschelde, and A. Zhao. Evaluation of Jacobian matrices for Newton's method with deflation to approximate isolated singular solutions of polynomial systems. In D. Wang and L. Zhi, editors, *Symbolic-Numeric Computation*, Trends in Mathematics, pages 269–278. Birkhauser, 2007.

[13] G. Malajovich. Computing mixed volume and all mixed cells in quermass-integral time. *Found. Comput. Math.*, pages 1–42, 2016.

[14] T. Mizutani and A. Takeda. DEMiCs: A software package for computing the mixed volume via dynamic enumeration of all mixed cells. In M.E. Stillman, N. Takayama, and J. Verschelde, editors, *Software for Algebraic Geometry*, volume 148 of *The IMA Volumes in Mathematics and its Applications*, pages 59–79. Springer-Verlag, 2008.

[15] T. Mizutani, A. Takeda, and M. Kojima. Dynamic enumeration of all mixed cells. *Discrete Comput. Geom.*, 37(3):351–367, 2007.

[16] R. Sabeti. Numerical-symbolic exact irreducible decomposition of cyclic-12. *LMS Journal of Computation and Mathematics*, 14:155–172, 2011.

[17] B. I. Sandén. *Design of Multithreaded Software. The Entity-Life Modeling Approach*. IEEE Computer Society, 2011.

[18] A. J. Sommese, J. Verschelde, and C. W. Wampler. Numerical irreducible decomposition using PHCpack. In M. Joswig and N. Takayama, editors, *Algebra, Geometry, and Software Systems*, pages 109–130. Springer-Verlag, 2003.

[19] A.J. Sommese, J. Verschelde, and C.W. Wampler. Introduction to numerical algebraic geometry. In A. Dickenstein and I. Z. Emiris, editors, *Solving Polynomial Equations. Foundations, Algorithms and Applications*, volume 14 of *Algorithms and Computation in Mathematics*, pages 301–337. Springer-Verlag, 2005.

[20] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999. Software available via `http://www.phcpack.org`.

[21] J. Verschelde and G. Yoffe. Polynomial homotopies on multicore workstations. In M.M. Maza and J.-L. Roch, editors, *Proceedings of the 4th International Workshop on Parallel Symbolic Computation (PASCO 2010)*, pages 131–140. ACM, 2010.