Decomposing Solution Sets of Polynomial Systems: A New Parallel Monodromy Breakup Algorithm[*]

Anton Leykin[†] **and Jan Verschelde**[‡]
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago, 851 South Morgan (M/C 249)
Chicago, IL 60607-7045, USA.

Abstract: Our problem is to decompose a positive dimensional solution set of a polynomial system into irreducible components. This solution set is represented by a witness set, obtained by intersecting the set with random linear slices of complementary dimension. Points on the same irreducible components are connected by path tracking techniques applying the idea of monodromy. The computation of a linear trace for each component certifies the decomposition. This decomposition method exhibits a good practical performance on solution sets of relatively high degrees defined by systems of low degree polynomials.

Using the same concepts of monodromy and linear trace, we present a new monodromy breakup algorithm. On multiple processors, we solve the synchronization issues which resulted in a performance loss of the straightforward parallel version of the original algorithm. Our new algorithm performs also better on a single processor: by exploiting several random slices and choosing pairs of slices according to accumulated statistics, the tracking of many redundant paths are avoided and new paths connecting points of the witness set are discovered with a higher probability.

2000 Mathematics Subject Classification. Primary 65H10. Secondary 14Q99, 68W30.
Key words and phrases. Homotopy, irreducible components, linear trace, monodromy, numerical algebraic geometry, parallel path tracking, polynomial systems.

1 INTRODUCTION

1.1 Problem Statement

As polynomial equations emerge more and more often in various fields of science and engineering, the question of simplification of polynomials and polynomial systems becomes of the most importance. How can we simplify? One way to understand better the solution set of a polynomial is to factor it; equivalently, in case of a polynomial system we talk about finding an *irreducible decomposition* of its solution set, a central problem in *numerical algebraic geometry* [35, 36]. For symbolic methods to deal with positive dimensional solution sets, see for example [21] or [40]. We refer to [37], for a description of the algorithms used in computer algebra systems to factor multivariate polynomials.

A widely known family of polynomial systems used for benchmarking is "cyclic n-roots", which arose in Fourier analysis [3, 4]. The case $n = 4$ is our running example:

$$\left\{ \begin{array}{rcl} x_1 + x_2 + x_3 + x_4 & = & 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 & = & 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_1 + x_4x_1x_2 & = & 0 \\ x_1x_2x_3x_4 - 1 & = & 0, \end{array} \right. \quad (1)$$

which is the so-called cyclic 4-roots problem, abbreviated as `cyclic4`. This system has a one dimensional solution component of degree four, which becomes obvious by the substitution $x_3 = -x_1$ and $x_4 = -x_2$. After this substitution, the first three equations vanish and the last equation simplifies to $x_1^2x_2^2-1$. Since $x_1^2x_2^2-1 = (x_1x_2-1)(x_1x_2+1)$, the curve of degree four factors in two irreducible quadrics.

We denote systems of polynomial equations by $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) = \mathbf{0}$, where $f_i \in \mathbb{C}[\mathbf{x}] = \mathbb{C}[x_1, \ldots, x_n]$ for all i. Very often, the coefficients are known with

[*]This material is based upon work supported by the National Science Foundation under Grant No. 0134611 and Grant No. 0410036.

[†]leykin@math.uic.edu. http://www.math.uic.edu/~leykin.
[‡]jan@math.uic.edu. http://www.math.uic.edu/~jan.

limited accuracy. The solution set V to $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ is naturally organized into pure dimensional solution sets $V = [V_0, V_1, \ldots, V_n]$, where $\dim(V_k) = k$. A numerical representation of a pure dimensional solution set V_k is a *witness set* [30] [36], which consists of

1. the polynomials f_i $(i = 1, \ldots, m)$;
2. k linear equations $\mathbf{L}(\mathbf{x}) = (L_1(\mathbf{x}), \ldots, L_k(\mathbf{x})) = \mathbf{0}$ with generic coefficients describing k generic hyperplane slices;
3. a list W of $\deg(V_k)$ isolated solutions to the system $\mathbf{f}(\mathbf{x}) = \mathbf{L}(\mathbf{x}) = \mathbf{0}$.

By the generic choice of the coefficients of the $\mathbf{L}(\mathbf{x}) = \mathbf{0}$, the k hyperplanes defined by $\mathbf{L}$ cut out exactly as many isolated regular solutions on V_k as $\deg(V_k)$.

Notice how the treatment of positive dimensional solution sets is reduced to dealing with collections of generic points. Using slack variables we reduce overdetermined polynomial systems to systems with as many variables as unknowns [29]. For a system like cyclic 4-roots in (1), we add one slack variable z to the system:

$$\left\{ \begin{array}{r} x_1 + x_2 + x_3 + x_4 + b_1 z = 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 + b_2 z = 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_1 + x_4x_1x_2 + b_3 z = 0 \\ x_1x_2x_3x_4 - 1 + b_4 z = 0 \\ c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + z = 0, \end{array} \right. \qquad (2)$$

where the coefficients $b_1, \ldots, b_4, c_0, c_1, \ldots, c_4$ are randomly chosen complex numbers. Observe that elimination of z corresponds to adding a random multiple of the last equation to the first four equations (a common trick in commutative algebra to deal with overdetermined polynomial systems). The extra linear equation reduces the dimension of the solution set by one and we find generic points on the curve as regular solutions with $z = 0$. For the cyclic 4-roots system the witness set contains four generic points, as the degree of the curve equals four.

The main question now is: given a positive dimensional solution set V, can we find its decomposition into the irreducible components? In the language of witness sets this interprets as: given a witness set $(\mathbf{f}, \mathbf{L}, W)$ of V, can we find a decomposition $W = W^{(1)} \sqcup \ldots \sqcup W^{(r)}$ such that for all i the witness set $(\mathbf{f}, \mathbf{L}, W^{(i)})$ represents an irreducible component of V?

Finding polynomial time algorithms for the factorization of multivariate polynomials with approximate coefficients was posed in [20] as one of the challenges in symbolic computation. This challenge received a lot of attention [6, 8, 9, 13, 14, 15, 19, 24, 25, 34]; see [7] for a nice description of recent methods.

1.2 Numerical Homotopies define Loops around Singularities

In [31], a new numerical algorithm using homotopy continuation methods was proposed to decompose a positive dimensional solution set into irreducible factors. Linear traces were proposed in [32] to certify a numerical irreducible decomposition. The implementation [33] was adjusted to the important special case of factoring one single multivariate complex polynomial in [34], see also [8] and [9].

In this section we outline the idea of exploiting *monodromy* using homotopies as in [31] to define loops around singularities. Assume two witness sets $(\mathbf{f}, \mathbf{L}_1, W_1)$ and $(\mathbf{f}, \mathbf{L}_2, W_2)$ represent the same positive dimensional irreducible component V. Consider the system $\mathbf{H}_{\gamma,\mathbf{L}_1,\mathbf{L}_2}(\mathbf{x}, t)$:

$$\left\{ \begin{array}{rcl} \mathbf{f}(\mathbf{x}) & = & \mathbf{0}; \\ (1-t)\mathbf{L}_1(\mathbf{x}) + \gamma t \mathbf{L}_2(\mathbf{x}) & = & \mathbf{0}. \end{array} \right. \quad t \in [0, 1] \qquad (3)$$

where γ is a generic nonzero complex number. Then, due to the generic choice of γ, for a fixed value of t all isolated solutions to $\mathbf{H}_{\gamma,\mathbf{L}_1,\mathbf{L}_2}(\mathbf{x}, t)$ are all regular. In particular, these are W_1 for $t = 0$ and W_2 for $t = 1$.

Tracking solutions of $\mathbf{H}_{\gamma,\mathbf{L}_1,\mathbf{L}_2}(\mathbf{x}, t)$ as t varies from 0 to 1 defines a 1-to-1 map $\phi_{\gamma,\mathbf{L}_1,\mathbf{L}_2} : W_1 \rightarrow W_2$. The composition of two such maps defines a permutation of the points of a witness set. In particular,

$$\pi_{\gamma,\mathbf{L}_1,\mathbf{L}_2} = \phi_{\gamma_2,\mathbf{L}_2,\mathbf{L}_1} \circ \phi_{\gamma_1,\mathbf{L}_1,\mathbf{L}_2} \qquad (4)$$

is a permutation of W_1. All permutations that arise in this fashion form a subgroup of the symmetry group acting on W_1 and the orbits of this action are witness sets that represent irreducible components.

The idea to exploit monodromy first appeared in a theoretical complexity study [2]. Although our approach does not need to know the precise location of the singularities, one could as in [11] compute those for algebraic curves, see also the command `algcurves[monodromy]` in Maple.

The algorithm in [31] collects points connected by loops into the same witness sets which converge to numerical representations of the irreducible components. In [32], a stop criterion for this algorithm was presented, using the linear trace. We explain this trace test on a system like cyclic 4-roots. Note that our program works only with generic points obtained as solutions of (2) and does not deal with a symbolic polynomial of degree four. Via a generic projection we map the points in 4-space down to the plane. If two of the four points belong to the same irreducible component, there must exist a quadratic polynomial $p(x, y)$ vanishing at those two points and at any point of the quadratic irreducible factor. The linear trace is then defined by rewriting $p(x, y)$ as $p(x, y(x))$:

$$\begin{array}{rcll} p(x, y(x)) & = & (y - y_1(x))(y - y_2(x)) & (5) \\ & = & y^2 - (y_1(x) + y_2(x))y + y_1(x)y_2(x) & (6) \\ & = & y^2 - t_1(x)y + t_2(x), & (7) \end{array}$$

where $t_1(x)$ is the linear trace. If t_1 was not linear, then $\deg(p) > 2$. So $t_1(x) = ax + b$, for some a and b to be determined by interpolation at $x = x_0$ and $x = x_1$, with corresponding y-values in (x_0, y_{01}), (x_0, y_{02}), (x_1, y_{11}), and

Algorithm 1.1 Monodromy Breakup certified by Linear Trace: $\mathcal{P} = \text{Breakup}(W_L, d, N)$

```
Input: $W_L$, $d$, $N$                                          witness set, degree, max #loops
Output: $\mathcal{P}$                                          partitioned witness set
0. initialize $\mathcal{P}$ with $d$ singletons;                done by manager node
1. generate two slices $L'$ and $L''$ parallel to the given $L$;   broadcast data to all nodes
2. track $d$ paths for witness set with $L'$;                  executed in parallel by workers
3. track $d$ paths for witness set with $L''$;                 executed in parallel by workers
4. for $k$ from 1 to $N$ do
   4.1 generate new slices $K$ and a random $\alpha$;          broadcast $K$ and $\alpha$ to all nodes
   4.2 track $d$ paths defined by $\mathbf{H}_{\alpha,L,K}$, see (3);   executed in parallel by workers
   4.3 generate a random $\beta$;                              broadcast $\beta$ to all nodes
   4.4 track $d$ paths defined by $\mathbf{H}_{\beta,K,L}$, see (3);    executed in parallel by workers
   4.5 compute the permutation and update $\mathcal{P}$;        done by manager node
   4.6 if linear trace test certifies $\mathcal{P}$
          then leave the loop;
       end if;
   end for.
```

(x_1, y_{12}). If for an additional sample, at $x = x_2$ with corresponding y-values (x_2, y_{21}), (x_2, y_{22}), we have $t_1(x_2) = y_{21} + y_{22}$, then we have an irreducible quadratic factor, otherwise the two points do not lie on the same factor.

The linear trace test, called zero-sum relations, was first introduced in [28] and further developed in [26, 27]. For factors of small to moderate degree, the linear trace test can be applied in an exhaustive combinatorial enumeration as was proposed in [13, 14, 24], [25] and improved in [6].

1.3 Parallel Algorithms

Homotopy continuation methods are very well suited for parallel processing as after distributing the path tracking jobs among the computers in the network, no further communications are needed, see [1, 5, 17] for granularity issues. For computational algebraic geometry, this implies that homotopy methods can solve much larger polynomial systems than methods in computer algebra which are harder to adapt to parallel computers [22]. One recent example is the solution of the cyclic 13-roots problems with PHoM [10, 16] for which 2,704,156 paths were tracked.

Modern homotopies in numerical algebraic geometry often appear in a sequence like the Pieri homotopies [39] where the start solutions of one homotopy lie at the end of paths defined by another homotopy. The homotopies to factor positive dimensional solution sets raise job scheduling issues as the decision to certain track paths depends on the outcome of other paths. This paper can be regarded as a solution to the job scheduling problems raised in [23].

The parallel algorithm proposed in [23] (described in pseudo code by Algorithm 1.1) exhibited a good speedup in the path tracking jobs, but the certification with linear traces executed only by the manager node before the scheduling of new path tracking jobs diminished the overall performance as all nodes were idling waiting for the assignment of new path tracking jobs. At the end of [23] we outlined a probabilistic complexity study simulated in Maple, suggesting various job scheduling techniques. In this paper we report on its parallel implementation confirming the efficiency of the approach.

2 USING MONODROMY MORE EFFICIENTLY

The monodromy breakup algorithm of [31, 32] is sketched by Algorithm 1.1. On input is a witness set W_L and on output a partition of W_L, corresponding to the irreducible decomposition.

Our first parallel implementation of this algorithm [23], described in the right column of Algorithm 1.1, uses a straightforward manager/worker model in which the manager node distributes the paths among the available processors in the network. According to our experimental results, a sizeable speedup is achieved by distributing the routine path-tracking jobs to different nodes. However, a probabilistic study in [23] suggested that we can save some work by taking a smaller one-path-one-point tracking job as an atomic task.

Following the previous discussion, we take two generic slices $\mathbf{L}_1$ and $\mathbf{L}_2$ (in this case these are hyperplanes) and look at the witness points W_1 and W_2. Consider the bipartite graph with vertices W_1 on one side and W_2 on the other.

One atomic step of the monodromy breakup algorithm consists of creating a map like $\phi_{\gamma,\mathbf{L}_1,\mathbf{L}_2}$. We can visualize such a map by connecting the points of W_1 and W_2 that map into each other with an edge, as shown in Figure 1.

As one may see in our example, in order to create one permutation we need to construct 8 edges in the graph. If one is lucky then it may take the algorithm only one permutation to decompose `cyclic4`, as shown in Figure 2.

The connected components of the graph in Figure 2 correspond to the two witness sets that, in turn, represent two irreducible components of the solution set of `cyclic4`: two quadric curves.

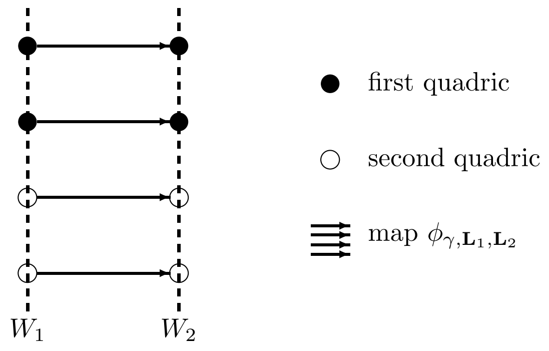
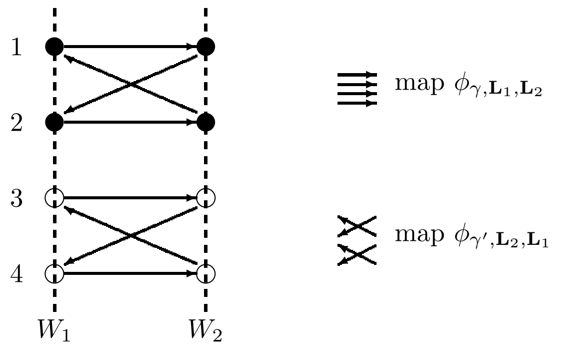


Figure 1: The bipartite graph $W_1 \leftrightarrow W_2$ for `cyclic4`



1
2
3
4
map $\phi_{\gamma,\mathbf{L}_1,\mathbf{L}_2}$
map $\phi_{\gamma',\mathbf{L}_2,\mathbf{L}_1}$
W_1
W_2

Figure 2: Permutation (12)(34) for `cyclic4`

In fact 2 out of 8 edges in Figure 2 can be removed keeping the connected components still connected, see Figure 3. Since each edge can be created by tracking only one point of a witness set, we may avoid doing extra work by trying to create as few edges as possible.

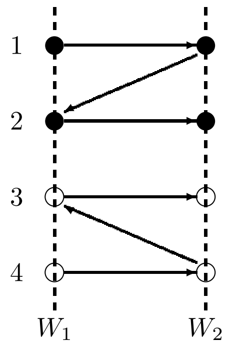


Figure 3: A 6-edge graph for `cyclic4`

3 A NEW ALGORITHM

In this section we first describe the sequential version of our new monodromy breakup algorithm before addressing its parallel execution.

3.1 Serial Version

The flow chart of our new algorithm is shown in Figure 4. As in Algorithm 1.1, we also have the initialization of the "trace grid", which are the two witness sets on two parallel slices needed to certify the irreducible decomposition using linear traces. While Algorithm 1.1 first completes all the loops for all points in a witness set before proceeding to the next level, our new approach initializes s new witness sets which are available for generating loops.

The main loop of the new algorithm shown in Figure 4 leaves much freedom to complete loops between any two slices. For every slice, the algorithm keeps track of the number of loops that did not yield a permutation, stored in N_{dis}. Based on these statistics, the algorithm can discriminate against slices which were not productive in the past and select those slices which led to more new permutations.

As explained in the previous section, the algorithm typically returns with a certified decomposition before all loops are completed. This is the main reason why on single processors, our new algorithm outperforms Algorithm 1.1. At the same time, the new algorithm is more suitable for parallel execution, as we will explain next.

3.2 Parallel Version

The parallel version of our new algorithm runs in a manager/worker model and is presented in Algorithm 3.1.

The initialization phase is very similar to the initialization of the parallel version of Algorithm 1.1, with the manager distributing path tracking jobs evenly among all nodes. After the initialization, the manager node keeps looking for available path tracking nodes to assign paths and the other nodes are either busy tracking paths or ready to start new path tracking jobs.

Compared to our previous parallel algorithm described in [23], the computation of the linear trace by the manager node is now interleaved by path tracking jobs performed on the other nodes.

4 EXPERIMENTAL RESULTS

Our algorithms are implemented using the path tracking routines in PHCpack [38], extended in [33] with facilities for a numerical irreducible decomposition. As in [39], we apply MPI for message passing. Our main program is written in C and links with the interface of PHCpack.

Our equipment consists of two personal cluster machines purchased from Rocketcalc (`www.rocketcalc.com`) for a total of 12 2.4Ghz CPUs, served by a Dell workstation with two dual 2.4Ghz processors. So in total we have 14 processors at our disposal.

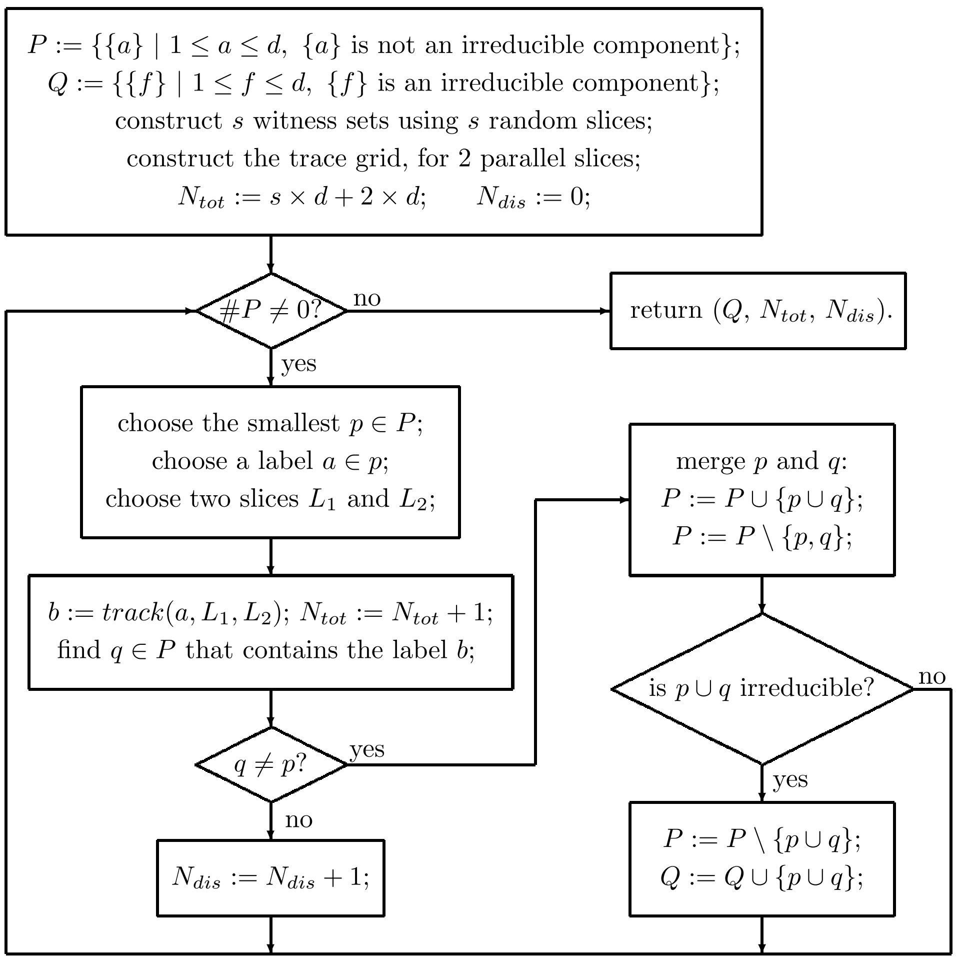

$P := \{\{a\} \mid 1 \le a \le d,\ \{a\}$ is not an irreducible component$\}$;
$Q := \{\{f\} \mid 1 \le f \le d,\ \{f\}$ is an irreducible component$\}$;
construct s witness sets using s random slices;
construct the trace grid, for 2 parallel slices;
$N_{tot} := s \times d + 2 \times d$; $N_{dis} := 0$;
$\#P \neq 0$?
no
return $(Q,\ N_{tot},\ N_{dis})$.
yes
choose the smallest $p \in P$;
choose a label $a \in p$;
choose two slices L_1 and L_2;
merge p and q:
$P := P \cup \{p \cup q\}$;
$P := P \setminus \{p, q\}$;
$b := track(a, L_1, L_2)$; $N_{tot} := N_{tot} + 1$;
find $q \in P$ that contains the label b;
is $p \cup q$ irreducible?
no
$q \neq p$?
yes
yes
no
$P := P \setminus \{p \cup q\}$;
$Q := Q \cup \{p \cup q\}$;
$N_{dis} := N_{dis} + 1$;

Figure 4: The flow chart of our new Monodromy Breakup Algorithm

Algorithm 3.1 Parallel Monodromy Breakup certified by Linear Trace:
$[Q, N_{tot}, N_{dis}] = \text{Parallel\_Breakup}(W_L, s)$

```
                              manager                                                 worker

input:
   $W_L$      a witness set;
   $s$        #new slices.
output:
   $Q$        partition of $\{1,\ldots,d = |W_L|\}$, $f \in Q$ is an irreducible component;
   $N_{tot}$  total #paths tracked;
   $N_{dis}$  #paths discarded (produced no new information).

broadcast $W_L$;                                                                   store $W_L$;
broadcast 2 parallel slices (for linear trace test);                               store 2 parallel slices;
broadcast $s$ random slices (for monodromy);                                       store $s$ random slices;
for $s+2$ newly created slices do
    scatter $d$ start points                                                       receive start points;
        of solution paths;                                                         track solution paths;
    gather $d$ end points; (these provide labels to the witness points on new slices)   send end points of paths;
end for;
$P := \{\{a\} \mid 1 \le a \le d,\ \{a\}$ is not an irreducible component$\}$;
$Q := \{\{f\} \mid 1 \le f \le d,\ \{f\}$ is an irreducible component$\}$;
$N_{tot} := s \times d + 2 \times d; N_{dis} := 0$;
while $\#P \neq 0$ do
    if a worker is idle then
        $[a, L_1, L_2] :=$ **Choose**$(P)$;
        send $(a, L_1, L_2)$ to worker;                                            receive $(a, L_1, L_2)$;
    end if;                                                                        $b :=$ **Track**$(a, L_1, L_2)$;
    if a worker is done then
        receive $b$ from worker;                                                   send $b$ to manager;
        **Update**$(a, b, P, Q, N_{tot}, N_{dis})$;
    end if;
end while;
return $(Q, N_{tot}, N_{dis})$.
```

Subroutines (see also the flow chart in Figure 4):

- $[a, L_1, L_2]$ **:= Choose**(P) picks a label a and two slices from the set of $s+1$ slices used for computing monodromy. Label a has to be in a subset $p \in P$ of the smallest size and such that the number of workers employed on tracking points with labels in p is minimal. The manager stores this number for every $p \in P$. The pair of slices L_1,L_2 is chosen to maximize the probability of discovering new information. This is done according to statistics collected and stored by the manager – we record the number of discarded paths amongst the fixed number of recent paths tracked between the pair.

- b **:= Track**(a, L_1, L_2) is the main atomic routine executed by workers which track a continuation path from a witness point labelled with a in W_{L_1} to produce a witness point in W_{L_2} labelled with b.

- **Update**$(a, b, P, Q, N_{tot}, N_{dis})$ increments N_{tot}. If a and b are in the same $p \in P$, then N_{dis} is incremented. Otherwise, if it finds $q \in P$ that contains b, then p and q are merged in the partition P, and the linear trace test will tell if $p \cup q$ is irreducible. If the linear trace test is successful, then $p \cup q$ is moved from P into the set of irreducible components Q.

4.1 Plain Parallel Path Tracking

In Table 1 we show with three runs the main defect of our first parallel implementation presented in [23]. While we have no real control over the number of loops it takes to complete the factorization, we observe from the data in Table 1 that as the total number of loops increases, the work done by the manager node increases.

#loops	4	6	9
manager	1.8	3.8	7.6
min track	8.0	10.9	18.5
max track	10.8	15.7	21.8
total	12.6	19.5	29.4

Table 1: Three runs with the first parallel monodromy breakup algorithm, executing respectively 4, 6, and 9 loops to factor a curve of degree 144 defined by the cyclic 8-roots problem on 14 processors. Times are reported in seconds: the time spent by the manager certifying the decomposition and scheduling the jobs; the minimal and maximal time the nodes spent tracking paths.

Although the cyclic 8-roots system is a problem of modest size, we already observe in Table 1 that for 9 loops, more than 25% of the time is spent by the manager node, while all the other nodes are idling. For larger problems and more processors, the poor performance of this first implementation will become even more apparent.

4.2 Performance of the new Algorithm

Table 2 reports on five runs done on 14 processors to decompose the curve of degree 144 defined by the cyclic 8-roots system. For three of the five runs we used three new slices. In the last two runs we see that the total time decreases if we use only two new slices.

	3 new slices			2 new slices	
#runs	1	2	3	4	5
initial	8.73	9.01	8.89	6.54	6.98
manager	6.06	6.22	6.18	6.67	7.10
min track	5.96	6.16	6.07	6.60	7.02
max track	6.06	6.24	6.23	6.11	7.15
total	14.9	15.4	15.3	13.4	14.2

Table 2: Five runs with our new parallel monodromy breakup algorithm, three times with 3 new slices and two times with 2 new slices, to factor the same curve of degree 144 defined by the cyclic 8-roots problem. We report the time used for initialization, the time spent by the manager node, the minimal and maximal time for the nodes spent tracking paths, and the total time. All reported times are expressed in seconds.

In Table 2 we see an even distribution of the time spent by the nodes. Using fewer slices reduces the initialization time at the expense of a slightly higher running time in the main loop.

Compared to the timings in Table 1 we do not notice such a wide fluctuation in the total execution time between different numbers of loops. The total execution time of the most favorable situation reported in Table 1 is only slightly lower than the best total time in Table 2.

Finally, we report on a calculation of a larger example, the ideal of adjacent 2-by-2 minors of a general 2-by-9 matrix of 18 unknowns, see [12] and [18]. This system in 18 variables defines a 10-dimensional surface of degree 256 which factors in 34 irreducible components. The total execution time of our new monodromy breakup algorithm on 14 processors is 97.1 seconds, of which 62.7 are spent on the initialization, 33.8 seconds by the manager node in the main loop while the time path tracking on the other nodes fluctuated between 32.9 and 35.6 seconds.

The performance of our first parallel algorithm on this system is even more erratic. The very best complete run of 3 monodromy loops took 122.9 seconds on 14 processors, where the path tracking time ranged between 75.9 and 104.4 seconds. Even on this very best run, our new algorithm still takes only 80% of the time spent by the first parallel monodromy breakup algorithm of [23].

5 CONCLUSIONS

In this paper we report on the development and performance of a new monodromy breakup algorithm. Experimental results show a more predictable and regular performance than our first parallel implementation of [23].

Due to its finer granularity and the absence of need for synchronization beyond the initialization stage the new algorithm exhibits much more even load distribution between the worker nodes. This together with avoiding tracking many redundant paths results in a performance superior to that of the previously used approach.

Acknowledgments. The authors thank the referees for their helpful and constructive comments.

REFERENCES

[1] D.C.S. Allison, A. Chakraborty, and L.T. Watson. Granularity issues for solving polynomial systems via globally convergent algorithms on a hypercube. *J. of Supercomputing*, 3:5–20, 1989.

[2] C. Bajaj, J. Canny, T. Garrity, and J. Warren. Factoring rational polynomials over the complex numbers. *SIAM J. Comput.*, 22(2):318–331, 1993.

[3] G. Björck. Functions of modulus one on Z_p whose Fourier transforms have constant modulus. In J. Szabados and K. Tandori, editors, *Proceedings of the Alfred Haar Memorial Conference, Budapest*, volume 49

of *Colloquia Mathematica Societatis János Bolyai*, pages 193–197. North Holland, 1985.

[4] G. Björck. Functions of modulus one on Z_n whose Fourier transforms have constant modulus, and "cyclic n-roots". In J.S. Byrnes and J.F. Byrnes, editors, *Recent Advances in Fourier Analysis and its Applications*, volume 315 of *NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci.*, pages 131–140. Kluwer, 1989.

[5] A. Chakraborty, D.C.S. Allison, C.J. Ribbens, and L.T. Watson. The parallel complexity of embedding algorithms for the solution of systems of nonlinear equations. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):458–465, 1993.

[6] G. Chèze. Absolute polynomial factorization in two variables and the knapsack problem. In J. Gutierrez, editor, *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation (ISSAC 2004)*, pages 87–94. ACM, 2004.

[7] G. Chèze and A. Galligo. Four lectures on polynomial absolute factorization. In A. Dickenstein and I.Z. Emiris, editors, *Solving Polynomial Equations: Foundations, Algorithms, and Applications*, volume 14 of *Algorithms and Computation in Mathematics*, pages 339–392. Springer–Verlag, 2005.

[8] R.M. Corless, A. Galligo, I.S. Kotsireas, and S.M. Watt. A geometric-numeric algorithm for factoring multivariate polynomials. In T. Mora, editor, *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation (ISSAC 2002)*, pages 37–45. ACM, 2002.

[9] R.M. Corless, M.W. Giesbrecht, M. van Hoeij, I.S. Kotsireas, and S.M. Watt. Towards factoring bivariate approximate polynomials. In B. Mourrain, editor, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC 2001)*, pages 85–92. ACM, 2001.

[10] Y. Dai, S. Kim, and M. Kojima. Computing all nonsingular solutions of cyclic-n polynomial using polyhedral homotopy continuation methods. *J. Comput. Appl. Math.*, 152(1-2):83–97, 2003.

[11] B. Deconinck and M. van Hoeij. Computing Riemann matrices of algebraic curves. *Physica D*, 152:28–46, 2001.

[12] P. Diaconis, D. Eisenbud, and B. Sturmfels. Lattice walks and primary decomposition. In B.E. Sagan and R.P. Stanley, editors, *Mathematical Essays in Honor of Gian-Carlo Rota*, volume 161 of *Progress in Mathematics*, pages 173–193. Birkhäuser, 1998.

[13] A. Galligo and D. Rupprecht. Semi-numerical determination of irreducible branches of a reduced space curve. In B. Mourrain, editor, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC 2001)*, pages 137–142. ACM, 2001.

[14] A. Galligo and D. Rupprecht. Irreducible decomposition of curves. *J. Symbolic Computation*, 33(5):661–677, 2002.

[15] X.-S. Gao, E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials via differential equations. In J. Gutierrez, editor, *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation (ISSAC 2004)*, pages 167–174. ACM, 2004.

[16] T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa, and T. Mizutani. PHoM – a polyhedral homotopy continuation method for polynomial systems. *Computing*, 73:55–77, 2004.

[17] S. Harimoto and L.T. Watson. The granularity of homotopy algorithms for polynomial systems of equations. In G. Rodrigue, editor, *Parallel processing for scientific computing*, pages 115–120. SIAM, 1989.

[18] S. Hoşten and J. Shapiro. Primary decomposition of lattice basis ideals. *Journal of Symbolic Computation*, 29(4&5):625–639, 2000.

[19] Y. Huang, W. Wu, H.J. Stetter, and L. Zhi. Pseudofactors of multivariate polynomials. In C. Traverso, editor, *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation (ISSAC 2000)*, pages 161–168. ACM, 2000.

[20] E. Kaltofen. Challenges of symbolic computation: my favorite open problems. *J. Symbolic Computation*, 29(6):891–919, 2000.

[21] D. Lazard. A new method for solving algebraic systems of positive dimension. *Discrete Applied Mathematics*, 33(1–3):147–160, 1991.

[22] A. Leykin. On parallel computation of Gröbner bases. In Y. Yang, editor, *Proceedings of the 2004 International Conference on Parallel Processing Workshops, 15-18 August 2004, Montreal, Quebec, Canada. High Performance Scientific and Engineering Computing*, pages 160–164. IEEE Computer Society, 2004.

[23] A. Leykin and J. Verschelde. Factoring solution sets of polynomial systems in parallel. In Tor Skeie and Chu-Sing Yang, editors, *Proceedings of the 2005 International Conference on Parallel Processing Workshops. 14-17 June 2005. Oslo, Norway. High Performance Scientific and Engineering Computing*, pages 173–180. IEEE Computer Society, 2005.

[24] D. Rupprecht. Semi-numerical absolute factorization of polynomials with integer coefficients. *J. Symbolic Computation*, 37(5), 2004.

[25] T. Sasaki. Approximate multivariate polynomial factorization based on zero-sum relations. In B. Mourrain, editor, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC 2001)*, pages 284–291. ACM, 2001.

[26] T. Sasaki, T. Saito, and T. T. Hilano. Analysis of approximate factorization algorithm i. *Japan J. of Industrial and Applied Math.*, 9:351–368, 1992.

[27] T. Sasaki and M. Sasaki. A unified method for multivariate polynomial factorizations. *Japan J. of Industrial and Applied Math.*, 10:21–39, 1993.

[28] T. Sasaki, M. Suzuki, M. Kolár, and M. Sasaki. Approximate factorization of multivariate polynomials and absolute irreducibility testing. *Japan J. of Industrial and Applied Math.*, 8:357–375, 1991.

[29] A.J. Sommese and J. Verschelde. Numerical homotopies to compute generic points on positive dimensional algebraic sets. *J. of Complexity*, 16(3):572–602, 2000.

[30] 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.

[31] A.J. Sommese, J. Verschelde, and C.W. Wampler. Using monodromy to decompose solution sets of polynomial systems into irreducible components. In C. Ciliberto, F. Hirzebruch, R. Miranda, and M. Teicher, editors, *Application of Algebraic Geometry to Coding Theory, Physics and Computation*, pages 297–315. Kluwer Academic Publishers, 2001. Proceedings of a NATO Conference, February 25 - March 1, 2001, Eilat, Israel.

[32] A.J. Sommese, J. Verschelde, and C.W. Wampler. Symmetric functions applied to decomposing solution sets of polynomial systems. *SIAM J. Numer. Anal.*, 40(6):2026–2046, 2002.

[33] 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.

[34] A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical factorization of multivariate complex polynomials. *Theoretical Computer Science*, 315(2-3):651–669, 2004. Special Issue on Algebraic and Numerical Algorithms edited by I.Z. Emiris, B. Mourrain, and V.Y. Pan.

[35] A.J. Sommese and C.W. Wampler. Numerical algebraic geometry. In J. Renegar, M. Shub, and S. Smale, editors, *The Mathematics of Numerical Analysis*, volume 32 of *Lectures in Applied Mathematics*, pages 749–763. AMS, 1996. Proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics. Park City, Utah, July 17-August 11, 1995, Park City, Utah.

[36] A.J. Sommese and C.W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005.

[37] M. van Hoeij. Factoring polynomials and the knapsack problem. *Journal of Number Theory*, 95:167–189, 2002.

[38] 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 at `http://www.math.uic.edu/~jan`.

[39] J. Verschelde and Y. Wang. Computing feedback laws for linear systems with a parallel Pieri homotopy. In Y. Yang, editor, *Proceedings of the 2004 International Conference on Parallel Processing Workshops, 15-18 August 2004, Montreal, Quebec, Canada. High Performance Scientific and Engineering Computing*, pages 222–229. IEEE Computer Society, 2004.

[40] W.T. Wu. Basic principles of mechanical theorem proving in elementary geometries. *Journal of Automated Reasoning*, 2(3):221–252, 1986.