Experimental evaluation and cross-benchmarking of univariate real solvers

Ioannis Z. Emiris, Michael Hemmer, Menelaos Karavelas, Bernard Mourrain, Elias P. P. Tsigaridas, Zafeirakis Zafeirakopoulos

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Experimental evaluation and cross-benchmarking of univariate real solvers

Ioannis Emiris — Michael Hemmer — Menelaos Karavelas — Bernard Mourrain — Elias P. Tsigaridas — Zafeirakis Zafeirakopoulos

N° 1234

Octomber 2008

Thème SYM

ISSN 0249-6399

Experimental evaluation and cross-benchmarking of univariate real solvers

Ioannis Emiris* , Michael Hemmer† , Menelaos Karavelas‡ , Bernard Mourrain§ , Elias P. Tsigaridas§ , Zafeirakis Zafeirakopoulos*

Thème SYM — Systèmes symboliques
Projet GALAAD

Rapport de recherche n° 1234 — Octomber 2008 — 17 pages

Abstract: Real solving of univariate polynomials is a fundamental problem with several important applications. This paper focuses on the efficient and generic black-box implementations of state-of-the-art algorithms for isolating all real roots of polynomials with integer coefficients, motivated by geometric applications and the recent need to develop software that handles exactly complex geometric objects, particularly in the case of the `CGAL` library. We summarize a large set of experimental results from cross-benchmarking 3 univariate algebraic kernels developed at the GALAAD group at INRIA, MPI-Saarbrücken, and the VEGAS group at LORIA. We have tested 6 solvers from the INRIA kernel, which are based on Sturm sequences, symbolic-numeric methods, and Continued Fractions (CF); two solvers from the MPI kernel, namely `Descartes` and `Bitstream Descartes`; and one solver from the LORIA kernel, relying on the Descartes-based `RS` solver developed at the SALSA group of INRIA. We used a total of 5000 polynomials of 5 types and various degrees and bitsizes, distributed in 150 datasets. The CF family of solvers, `Descartes`, `Bitstream Descartes`, and `RS` are numerically and combinatorially robust, i.e., they always provide the correct results. With respect to speed, the results are not decisive in most cases; overall, two CF methods seem faster, even though currently they do not use symbolic-numeric techniques, while for very large bitsizes `Bitstream Descartes` and `RS` seem more efficient, provided the degree is not very high. For polynomials of degree up to 4 and moderate bitsize, special algorithms give better results. It is important to note that the implementations of the theoretically exact methods are complete, efficient and they never provided wrong results throughout this extensive benchmarking procedure. Lastly, we comment on the different

Partially supported by contract ANR-06-BLAN-0074 "Decotes".

* National Kapodistrian University of Athens, Athens, Greece {emiris, grad0783}@di.uoa.gr
† Max-Planck-Institut für Informatik, Saarbrücken, Germany, hemmer@mpi-inf.mpg.de
‡ University of Crete & FO.R.T.H., Heraklion, Greece, mkaravel@tem.uoc.gr
§ INRIA, Sophia-Antipolis, France, {Bernard.Mourrain, Elias.Tsigaridas}@sophia.inria.fr

programming design approaches adopted by the different kernels, including the degree of interoperability between different algorithmic components, as well as arithmetic data types.

Key-words: univariate polynomials, real solving, algebraic kernel, exact computation, robust implementation

Comparaison experimentale d' outil de résolution d' équations polynomiales en une variable

Résumé :

Mots-clés :

1 Introduction

Real solving of univariate polynomials is a fundamental algebraic question with several important applications. This paper focuses on the efficient and generic implementations of state-of-the-art algorithms for isolating all real roots of polynomials with integer coefficients, motivated by geometric applications and the recent need to develop software that handles exactly complex geometric objects, particularly in the case of the `CGAL`[1] library. We summarize the results of extensive cross-benchmarks for 3 univariate algebraic kernels (see Section 3, [9, 12, 14, 16]) developed at the GALAAD group at INRIA, MPI-Saarbrücken, and the VEGAS group at LORIA. All tables and graphs produced during the benchmarks are available through
`http://acs.cs.rug.nl/index.php?page=p_benchmarks`.

Previous work includes [13] which describes benchmarks between a Sturm-sequence based solver developed by NUA2, and MPI's solvers `Descartes`and `Bitstream Descartes`. The major conclusion for real root isolation is that for degrees ≥ 20 the latter are faster or much faster. For smaller degrees, all methods are comparable.

We tested 9 methods on 5000 polynomials distributed in 150 datasets (250000 solve calls, over 50h total runtime). We used 5 different types of polynomials of degrees between 3 and 184 and bitsizes between 10 and 8000, see Section 4. To the best of our knowledge, this is the largest number of tests for univariate real solving up to date.

The CF family of solvers, `Descartes`, `Bitstream Descartes`, and numerically and combinatorially robust, i.e., they always provide the correct results. With respect to speed, the results are not decisive in most cases. Overall, two CF methods, namely `CF` and `NCF2` seem faster, while for very large bitsizes `Bitstream Descartes` and `RS` seem more efficient, provided the degree is not very high (see Section 5, Figure 3). For polynomials of degree up to 4 and moderate bitsize, special algorithms give better results as expected from their theoretical compexity, see Figure 4. It is important to note that the implementations of the theoretically exact methods are complete, efficient and they never provided wrong results throughout this extensive benchmarking procedure. We comment on the different programming design approaches adopted by the different kernels, including the degree of interoperability between different algorithmic components, as well as arithmetic data types, see Section 3.

Our results are useful for validating various algorithmic approaches to the fundamental problem of real solving, as well as for evaluating different implementation strategies of algebraic operations aimed at supporting robust and efficient computing in a wide range of applications.

In what follows O_B means bit complexity and the $\widetilde{O}_B$-notation means that we are ignoring logarithmic factors. Moreover, d will be the degree of the polynomials and τ the maximum coefficient bitsize.

The rest of the paper is structured as follows. In Section 2 we describe the setup of the benchmarks. Section 3 describes the methods we tested. Section 4 describes the datasets

[1] `http://www.cgal.org/`

we used. Section 5 presents the results of the benchmarks. In Section 6 we present the conclusions of the cross-benchmarking process for the univariate solvers. Finally, Appendix A contains additional plots produced by the benchmarks and in Appendix B we provide tables showing the best method for various combinations of polynomial type, degree and bisize.

2 Benchmark setup

We used for the benchmarks a 32-bit Pentium III with 256MB RAM. All compilations were done using the optimization flags `-O3` and `-DNDEBUG`. We used the internal release 271 of `CGAL` (released 03-Apr-2008), as well as the software libraries GMP[2] (version 4.2)and NTL[3] (version 5.4.1). The compiler used was `g++` version 4.1.2, on a Debian etch Linux platform.

We measured the time needed by a method to isolate all the real roots of a polynomial, using the clock() function of the `ctime` library (measurements are in msec). In case a method takes more than 30 seconds for some instance, we ignore the measurement. The average over all polynomials in each dataset is reported. For instances that took less than 1 msec, we reported the average time over a number of iterations to exceed 1 msec total time in order to reduce noise.

3 Real Root Isolation methods

An algebraic kernel, see [2], consists of three major components.

1. Polynomials and basic operations. (gcd, square free factorization etc.)
2. Isolation which is the core of the functor Solve_1.
3. Representation of algebraic real roots, comparison of algebraic reals, approximation refinement.

Although the benchmarking was only concerned for univariate real root isolation, we implicitly tested the implementation of square free factorization and real algebraic numbers. The real root isolation methods we tested are part of the Algebraic Kernels developed in INRIA, MPI and LORIA. The INRIA Algebraic Kernel is available through `mathemagix`[4]. The MPI and LORIA Algebraic Kernels are available through the `CGAL`'s svn server.

The algebraic kernel by INRIA relies on SYNAPS[5], which in turn is now been integrated in `mathemagix`. It is an open source effort that provides fundamental algebraic operations such as algebraic number manipulation tools, different types of univariate and multivariate polynomial root solvers, resultant and gcd computations, etc. The main motivation behind this project, is the need to combine symbolic and numeric computations, which is ubiquitous

[2] `http://gmplib.org/`
[3] `http://www.shoup.net/ntl`
[4] `http://www-sop.inria.fr/galaad/mathemagix/`
[5] `http://www-sop.inria.fr/galaad/logiciels/synaps/`

in many problems. We refer the reader to [15] for more details. A notable comment about the polynomial solvers in this library is that they exploit specialized algorithms for polynomials of degree up to 4, that have constant arithmetic complexity, i.e. $O(1)$, and bit-complexity $\widetilde{O}_B(\tau)$, where τ is the coefficient bitsize. The algorithms are based on pre-computed Sturm sequences and they compute rational points that isolate the real roots, as functions in the coefficients of the polynomial. We refer the reader to [7, 17] for more details. In particular, the `Sturm`, `Sleeve`, `Symbnum` and `CF`solvers (see below), use the specialized algorithms for real solving polynomials of degree ≤ 4. Hereafter, we refer to this algorithm as IDS (Isolation and Discrimination Systems).

The algebraic kernel provided by the MPI has been designed such that exploits genericity for real root isolation methods and the way the algebraic real roots are represented, that is, the kernel has two template arguments. This is the principal advantage of the MPI kernel, since it can potentially combine the best root isolator with the best representation of algebraic reals due to its generic design. In the context of this paper the most important template argument is the first, namely the root isolator. For more details as well as the exact definition of the root isolator concept see [12]. Up to date the MPI has provided two classes that are models of the root isolator concept, the Descartes and the `Bitstream Descartes`. However, in principal the MPI kernel could use any root isolation method presented here as long as it provides the proper interface. Moreover, it is not restricted to a certain coefficient type since this is implicitly defined by the given root isolator.

Description of the methods

`Sturm` A subdivision method based on Sturm's theorem. In order to isolate the real roots, the algorithm employs polynomial remainder sequences as a real root counting query. The polynomial remainder sequence is evaluated at the endpoints of the interval at question, to determine if a root is isolated in the given interval. The complexity of the algorithm is $\widetilde{O}_B(d^6 + d^4\tau^2)$. See [4, 9] and references therein. In general, it is expected to be the slowest method. We use it more as a reference, than actually comparing it in a competitive way.

`Sleeve` An approximate real root isolation method that is based on computing an "upper" and "lower" (i.e. a sleeve) approximation of the polynomial. A Descartes-like subdivision algorithm is applied to the sleeve. The approximation is refined, if possible, until the machine precision is reached. The isolating intervals, that the algorithm returns, may contain more than one real root. In some cases we can certify the result using a sign test with the first derivative and interval arithmetic.

`CF` **family** There are three variants of the Continued Fractions (CF) algorithms tested. The algorithm computes the continued fractions expansion of the real roots of the polynomial and then computes an isolating interval with rational end-points for each root, as described in [18]. The algorithm's complexity is $\widetilde{O}_B(d^6 + d^4\tau^2)$.

`NCF` is based on the same algorithm, but the implementation employs the NTL library for square free factorization and factorization over the integers. Integer arithmetic in NTL is based on GMP, which is used through a SYNAPS-NTL conversions interface, see [11].

The difference between `NCF` and `NCF2` is that `NCF2` employs square free factorization, but not factorization over the integers. We have to note that the current implementation of these methods uses only exact arithmetic and does not exploit approximation techniques.

`Symbnum` It is an experimental symbolic-numeric algorithm, which is a combination of the Sleeve and the Continued Fraction algorithm. Initially the Sleeve algorithm runs and produces some intervals, that may contain more than one real root. In the sequel, using exact arithmetic, we compute the continued fractions expansion of the real root(s) that are in the interval, the resulting interval is transformed to (0, 1) and the Sleeve algorithm is applied again (possible after scaling the coefficients).

`Descartes` A real root isolation method based on Descartes's rule of sign. This algorithm has been integrated in `CGAL` from the NumeriX library of EXACUS and was implemented according to [3]. Given the new tree bound in [5] the complexity of the algorithm is $\widetilde{O}_B(d^5\tau^2)$.

`Bitstream Descartes` The coefficients of the polynomial are converted to (potentially infinite) bitstreams. A variant of Descartes's method is used to find isolating intervals for the real roots. The worst case complexity is $\widetilde{O}_B(d^5\tau^2)$. The advantage of this method is that it is adaptive in the number of bits used, see also [5, 6]. A major advantage of both Descartes implementations is that they are generic with respect to the coefficient type. In particular the `Bitstream Descartes` is intended to work over algebraic extensions, as it is reported in [5, 6, 12], for example, it can isolate roots of $f(x, y)|x = \alpha$, where α is an algebraic number. This is the actual strength of the `Bitstream Descartes`.

`RS`-solver The real root isolation is based on the `RS` solver[6] from the SALSA group at INRIA-Rocquencourt, see [14, 16]. The `RS` solver is based on interval Descartes algorithm, approximating the transformation of $[a, b]$ to $[0, \text{inf}]$, when this is sufficient for sign determination. The order of the transformations is important for the memory consumption of the method.

4 Data description

We used datasets of polynomials with different characteristics, in order to have a picture of the advantages or disadvantages of the solvers in various cases. This work was done within the ACS project, thus an increased interest in polynomials derived from geometric problems is evident.

[6] available at `http://fgbrs.lip6.fr/salsa/Software/`

`[rnd]` are datasets of random polynomials of degree between 3 and 100 and bitsize from 10 to 50. These polynomials are the easiest to solve among the ones tested.

`[bts]` Random polynomials are important, since they occur naturally in many problems. Using code for generating polynomials for test files with `CGAL`, we created the `[bts]` datasets of degree between 3 and 100 and bitsize in from 2000 to 8000.

`[mgn]` datasets contain polynomials of the form $x^d - 2(kx-1)^2$ and degree between 3 and 100 and bitsize from 10 to 50. These polynomials have two real root very close each other, and are the most "difficult" polynomials for subdivision-based algorithms.

`[int]`/`[rat]` datasets contain polynomials with integer/rational roots. While the choice of roots for the construction of the polynomials is random, there are cases with multiple selections of the same root. (e.g. small bitsize, high degree). The datasets consist of combinations of degree from 3 to 100 and bitsize from 10 to 50.

`[3darr]` include univariate polynomials in bff format as described in [1]. The polynomials are motivated by 3D-arrangement problems. Each dataset contains 100 univariate polynomials of bitsize 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800 or 2000 bits and degree 12. The polynomials have several real roots within the interval $[-10, 10]$. Each polynomial is the product of six parabolas, where each parabola has at least one root within the interval $[-10, 10]$. Consequently, all polynomials have 12 real roots.

`[vorell]` contains 10 polynomials of degree 184 and bitsize of approximately 3500 bits. These polynomials are obtained from resultants and describe all tritangent circles to three ellipses, see [8, 10]. The ellipses are given randomly through a Graphical User Interface. The input bitsize of the ellipses is from 11 to 13 bits, while the bitsize of the resultant polynomial ranges from 3100 to 3800 bits.

The `[3darr]` datasets are available through `CGAL`'s svn server. The `[rnd]`, `[int]`,`[rat]` and `[mgn]` datasets are available at `http://synaps.inria.fr/bench/upol_root.tgz`.

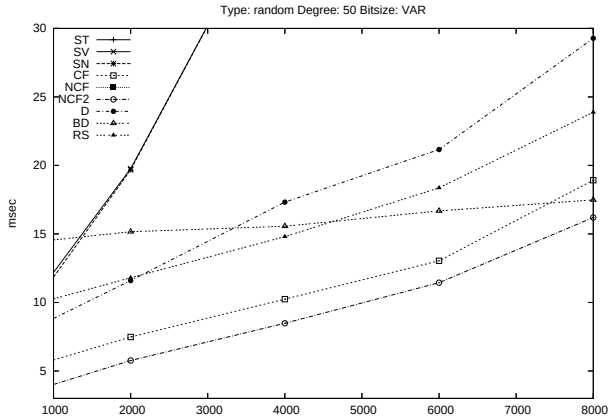
5 Results

Isolation and Discrimination Systems Methods using Isolation and Discrimination Systems have an advantage in cases of degree up to 4 and small to moderate bitsize, but are outperformed by `Descartes` and `Bitstream Descartes` for bitsizes over a threshold (2000 bits). This is expected, since the Isolation and Discrimination Systems are bitsize sensitive. The low theoretical complexity of special algorithms for polynomials of degree up to 4 is consistent with the benchmark results, see Figure 4.

Random polynomials

`[rnd]` `Sleeve` and `Symbnum` are close to `NCF2` and faster than all other methods on polynomials of degree 100 and bitsize up to 50. The randomness of the coefficients is in favor of these approximation methods. `NCF2` is the most effective method. `RS`, `Descartes` and `Bitstream Descartes` are outperformed by `NCF2`, `CF` and the two approximation methods. Due to factorization over the integers, `NCF` is affected by the bitsize increase.

Figure 1: [bts]Random polynomials of degree 50 and big bitsize



[bts] When the bitsize is over 4000, `Descartes` and `Bitstream Descartes` are performing better than the rest of the methods provided that the degree is up to 10. While the degree increases, `NCF2` and `RS` are faster. For bitsize 8000 bits and degree 50, `Bitstream Descartes` is equivalent to `CF`, while for bitsize 2000 bits `CF` is faster than `Bitstream Descartes`. See Figure 1.

In any case, `NCF2` and `RS` are faster than `Bitstream Descartes` for degrees higher than 10. The break-even point for `RS` and `NCF2` for degree 50 is 6000 bits. For bitsizes bigger than that, `RS` is faster than `NCF2`.

`Bitstream Descartes` is not affected by bitsize increase. For polynomials of degree 100 and bitsize from 2000 to 8000 bits, `Bitstream Descartes` shows no change in performance. While the degree of the polynomial increases, `Bitstream Descartes` requires more bits due to internal bounds that depend on the degree. The constant overhead is big in the case of high degrees, thus up to 8000 bits `Bitstream Descartes` is not better than `NCF2` or `RS`. This result is consistent with the results in [14].

Polynomials with integer or rational roots All methods accept a flag to determine whether a polynomial is known to be square-free. The `[int]` and `[rat]` sets, contain both square-free and non square free polynomials, thus the methods were called asking for multiplicities. This means that implicitly, we are timing square-free factorization as well. The degree increase is affecting the square free factorization and influences the time measured. By the time of the benchmarks the MPI kernel did not employ modular arithmetic to compute the GCD during square-free factorization[7], this caused a serious drawback. Hence, it was not possible to compare `Descartes` and `Bitstream Descartes` with respect

[7] Modular GCD is now being integrated in `CGAL`. The timings will be updated for the final version of this paper.

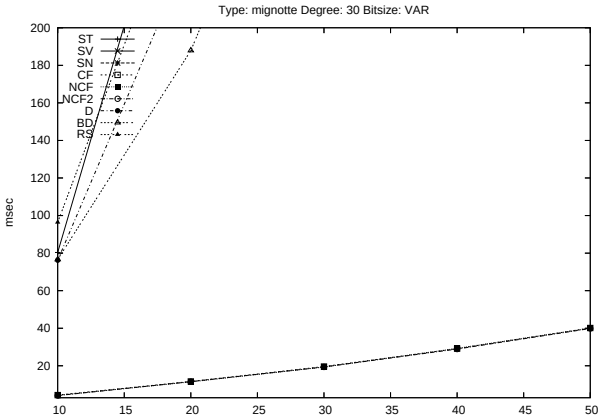
to the other methods based on these results. The fact that these sets (especially `[int]`) have many polynomials with multiple roots gives an advantage to the `mathemagix` methods, which use the NTL library for factorization. Therefore, `NCF` and `NCF2` are equivalent and clearly the fastest methods. The LORIA kernel implements its own modular gcd. This may explain why RS is in all cases in between of the two groups, slower than `NCF` and `NCF2`, but faster than the rest of the methods, see Figure 5.

Every method has similar behavior for `[int]` and the respective `[rat]` cases.

Mignotte polynomials The `[mgn]` polynomials are the most difficult to solve among the polynomials tested. In some cases, especially when the bitsize was big, approximation methods (`Sleeve` and `Symbnum`), failed to isolate the roots correctly, providing only 3 roots instead of 4. Even for moderate degrees, only methods based on Continued Fractions are capable of solving Mignotte polynomials. For example, the times are not even comparable for `[mgn]` of degree 30, as shown in Figure 2.

`RS` is considerably slow in `[mgn]`. The problem is due to the high memory consumption of the algorithm. In cases of degree higher than 30 the method would run out of memory.

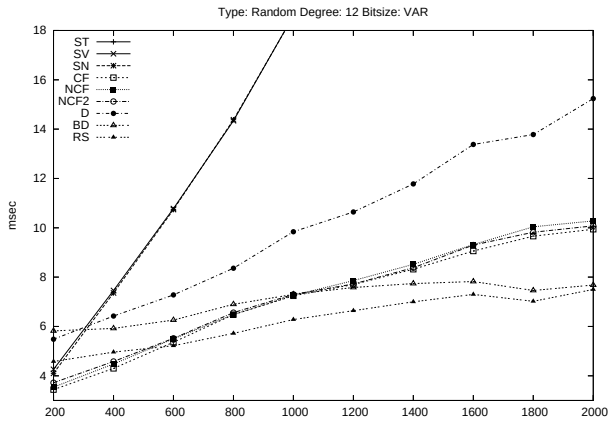
Figure 2: `[mgn]` Minotte polynomials of degree 30



Geometric problems The `[3darr]` polynomials have relatively big bitsize and we can observe the break-even point of `Bitstream Descartes`. For small bitsizes the CF family methods perform considerably better. `Bitstream Descartes` outperforms `Descartes` and the CF-family for bitsizes bigger than 1200. This observation is consistent with the results presented in [12]. `RS` is the most effective method for bitsizes over 600 bits. The behavior of the method is very good in this set, since it is fast enough for polynomials of bitsize lower than 600, and the fastest one for bitsizes over this threshold. The bitsize increase has less effect on `Bitstream Descartes` and `RS` than on `CF`, `NCF2` and `Descartes`. See Figure 3. The

low degree (in this case 12) is in favor of `Bitstream Descartes` for high bitsizes. Moreover, `NCF` due to the factorization over the integers is slower than `CF` and `NCF2` in the datasets of big bitsizes.

Figure 3: `[3darr]` Polynomials of degree 12, derived from 3D arrangement problems



The `[vorell]` dataset contains polynomials having the highest degree of all the polynomials tested. Moreover, the bitsize is considerably high (approx. 3500 bits). The fastest methods in this case are `Sleeve` and `Symbnum`. Since the polynomials are random, the approximation suffices for the root isolation. `NCF2` is very close to these methods. `NCF` and `Descartes` are equivalent, while `Bitstream Descartes` is not competitive. The degree of these polynomials slows down `Bitstream Descartes`, while `NCF`'s factorization is slow due to the bitsize. `RS` is equivalent to `CF`, which are better than `NCF` and `Descartes`, but worse than `NCF2` and the approximation methods.

6 Conclusions

From the general picture of the benchmarks, it is clear that there is no "best method" overall. Depending on the characteristics of the problem the performance of any method may change. A remarkable observation is that the implementations of the theoretically exact methods are complete, efficient and they never provided wrong results throughout this extensive benchmarking procedure.

In this paper we summarized the performance of the univariate solvers in several typical problem cases. Tables showing the method with the best performance for `[rnd]`/`[bts]`, `[int]` and `[mgn]` are presented in Figure 6, Figure 7 and Figure 8 respectively.

Robustness In terms of robustness, the CF-family, `Descartes`, `Bitstream Descartes` and `RS` never failed to provide the correct result (considering the number of roots returned).

In some cases though, `CF`, `Descartes` or `Bitstream Descartes` may need too much time to finish. Nevertheless, given the time, they would finish by isolating all real roots. The implementations of these methods comply with their theoretic exactness.

The sets that produced the most problems in approximation methods are `[mgn]`. The usage of inexact types leads to false results, since the roots are too close to be isolated correctly.

Performance The `Sleeve` method is very effective in small problems, where an approximation is able to isolate the roots correctly. Even for `[mgn]`, which is the most difficult case for approximation methods. When it comes to higher degrees and bitsizes, it is either much slower than the other methods or fails to isolate all the real roots correctly. Nevertheless, for random polynomials of moderate bitsize approximation methods perform well.

In the case of `[3darr]` and `[bts]` not even approximation methods failed to isolate the roots. As long as the degree is low and the roots are well separated, the bitsize does not cause approximation methods to fail in isolating all roots, or exact methods to exceed the time limit.

The difficulty of root isolation, as shown in `[mgn]`, is due to the separation bound and not to the coefficient bitsize. It is clear that Continued Fractions based methods always provide the correct result much faster than any other method in cases of small separation bound.

Summary Overall one can say that for nearly all instances the `CF` and the `NCF2` solvers are among the best methods, even though the current implementation uses only exact arithmetic and does not exploit symbolic-numeric techniques that would speed up their performance even more. The only exceptions are cases of polynomials having simultaneously low degree and high bitsize, as shown in the tables in Appendix B. In these case Bitstream Descartes or RS show a better performance, where RS is slightly more effected by the bitsize than the Bitstream Descartes. `Bitstream Descartes` is able to handle coefficient types other than integers (rationals), e.g. coefficients from algebraic extensions. Currently is the only method recommended for non rational coefficient types.

Future Work Given the advantages of the bitstream method for huge bit sizes as well as the advantages of the methods based on Continued Fractions for small separation bound, it might be worth trying to exploit the bitstream idea in the context of CF methods. Moreover, it should be possible to improve the performance of the symbolic-numeric techniques which can serve as a filter for all exact methods presented here.

7 Acknowledgments

All authors acknowledge partial support by IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex

Shapes). B. Mourrain and E. Tsigaridas also acknowledge partial support by contract ANR-06-BLAN-0074 "Decotes". The authors are greatful to Luis Peñaranda for his help with the LORIA kernel and to Sebastian Limbach for his help with the MPI kernel.

References

[1] Eric Berberich, Franziska Ebert, and Lutz Kettner. Definition of file format for benchmark instances for arrangements of quadrics. Technical Report ACS-TR-123109-01, MPI, Month 12 2006.

[2] Eric Berberich, Michael Hemmer, Menelaos I. Karavelas, and Monique Teillaud. Revision of the interface specification of algebraic kernel. Technical Report ACS-TR-243301-01, INRIA Sophia-Antipolis, Max Planck Institut für Informatik, National University of Athens, 2007.

[3] G. E. Collins and A. G. Akritas. Polynomial real root isolation using descarte's rule of signs. In *Proc. 3^{rd} ACM Symp. on Symbolic and Algebraic Comp.*, pages 272–275, NY, USA, 1976. doi: http://doi.acm.org/10.1145/800205.806346.

[4] Z. Du, V. Sharma, and C. K. Yap. Amortized bound for root isolation via Sturm sequences. In D. Wang and L. Zhi, editors, *Int. Workshop on Symbolic Numeric Computing*, pages 113–129, School of Science, Beihang University, Beijing, China, 2005. Birkhauser.

[5] A. Eigenwillig. *Real Root Isolation for Exact and Approximate Polynomials Using Descartes' Rule of Signs*. PhD thesis, Universität des Saarlandes, Saarbrücken, 2008.

[6] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert. A Descartes algorithm for polynomials with bit-stream coefficients. In *Proc. 8th Int. Workshop on Computer Algebra in Scient. Comput. (CASC)*, LNCS. Springer, 2005. to appear.

[7] I. Z. Emiris and E. P. Tsigaridas. Computing with real algebraic numbers of small degree. In S. Albers and T. Radzik, editors, *Proc. 12th European Symp. of Algorithms (ESA)*, volume 3221 of *LNCS*, pages 652–663, Bergen, Norway, Sep 14–17 2004. Springer Verlag.

[8] I. Z. Emiris, E. P. Tsigaridas, and G. M. Tzoumas. Predicates for the exact Voronoi diagram of ellipses under the euclidean metric. *Intern. J. Computational Geometry & Applications, Special Issue*. Accepted. Available from http://www.di.uoa.gr/~geotz/.

[9] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real Algebraic Numbers: Complexity Analysis and Experimentation. In P. Hertling, C. Hoffmann, W. Luther, and N. Revol, editors, *Reliable Implementations of Real Number Algorithms: Theory and Practice*, volume 5045 of *LNCS*, pages 57–82. Springer Verlag, 2008. also available in www.inria.fr/rrrt/rr-5897.html.

[10] Ioannis Z. Emiris and George M. Tzoumas. A real-time and exact implementation of the predicates for the Voronoi diagram of parametric ellipses. In *Proc. of the 2007 ACM symposium on Solid and physical modeling (SPM)*, pages 133–142, Beijing, China, June 2007. ACM Press. ISBN 978-1-59593-666-0. doi: http://doi.acm.org/10.1145/1236246.1236266.

[11] I.Z. Emiris, E.P Tsigaridas, and G.M. Tzoumas. Voronoi diagram of ellipses in CGAL. In *Proc. Europ. Works. Comp. Geom.*, pages 87–90, Nancy, France, 2008.

[12] Michael Hemmer and Sebastian Limbach. Benchmakrs on a generic univariate algebraic kernel. Technical Report ACS-TR-243306-03, Algorithms for Complex Shapes with certified topology and numerics, Max Planck Institut fÃ¼r Informatik, SaarbrÃ¼cken, GERMANY, 2007.

[13] Menelaos I. Karavelas. Preliminary cross-benchmarks of the MPI and NUA univariate algebraic kernels. Technical Report ACS-TR-243306-05, National University of Athens, 2008.

[14] S. Lazard, L Peñaranda, and E Tsigaridas. A CGAL-based univariate algebraic kernel and application to arrangements. In *Proc. Europ. Works. Comp. Geom.*, pages 91–94, 2008.

[15] B. Mourrain, P. Pavone, P. Trébuchet, E. P. Tsigaridas, and J. Wintz. SYNAPS, a library for dedicated applications in symbolic numeric computations. In M. Stillman, N. Takayama, and J. Verschelde, editors, *IMA Vol. in Math. and its Applications*, pages 81–110. Springer, 2007.

[16] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial real roots. *J. of Computational and Applied Mathematics*, 162(1):33–50, 2003.

[17] E. P. Tsigaridas. *Algebraic algorithms and applications to geometry.* PhD thesis, National Kapodistrian University of Athens, Aug 2006. (http://www-sop.inria.fr/galaad/elias).

[18] Elias P. Tsigaridas and Ioannis Z. Emiris. On the complexity of real root isolation using continued fractions. *Theor. Comput. Sci.*, 392(1-3):158–173, 2008. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/j.tcs.2007.10.010.

A Plots

Figure 4: `[rnd]`Random polynomials of degree 3.We observe the IDS behaviour.

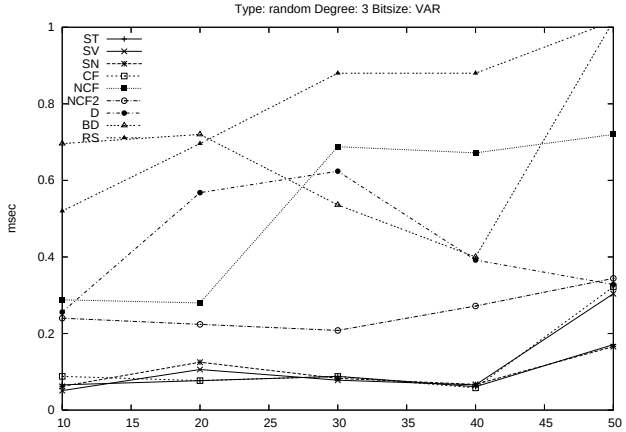
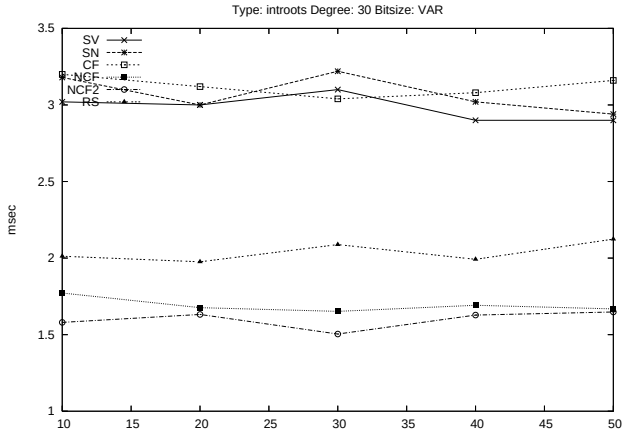


Figure 5: `[int]`Polynomials of degree 30 with integer roots.



B Best performance tables

In the following tables we present the fastest solver for each combination of polynomial type, degree and bitsize tested. The legend gives the correspondance of the abbreviations used in the tables and the names of the methods as described in Section 3.

Table 1: Legend

SN	`Symbnum`
CF	`CF`
NCF	`NCF`
NCF2	`NCF2`
CFF	`CF` Family
IDS	Isolation and Discrimination Systems
BD	`Bitstream Descartes`
RS	`RS` Solver
n/a	Not decisive

Figure 6: Best methods for random polynomials

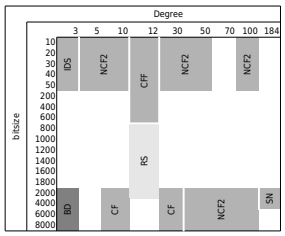


Figure 7: Best methods for polynomials with integer roots

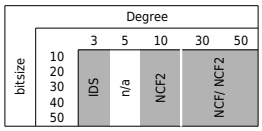
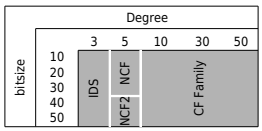


Figure 8: Best methods for Mignotte polynomials



Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399