Ruyong Feng · Wen-shin Lee
Yosuke Sato *Editors*

Computer Mathematics

9th Asian Symposium (ASCM2009), Fukuoka, December 2009, 10th Asian Symposium (ASCM2012), Beijing, October 2012, Contributed Papers and Invited Talks

Springer

Computer Mathematics

Ruyong Feng · Wen-shin Lee
Yosuke Sato
Editors

Computer Mathematics

9th Asian Symposium (ASCM2009), Fukuoka, December 2009, 10th Asian Symposium (ASCM2012), Beijing, October 2012, Contributed Papers and Invited Talks

Springer

Editors
Ruyong Feng
Academy of Mathematics
and Systems Science
Beijing
China

Wen-shin Lee
Department of Mathematics
and Computer Science
University of Antwerp
Antwerp
Belgium

Yosuke Sato
Department of Mathematical
Information Science
Tokyo University of Science
Tokyo
Japan

ISBN 978-3-662-43798-8 ISBN 978-3-662-43799-5 (eBook)
DOI 10.1007/978-3-662-43799-5

Library of Congress Control Number: 2014949487

Mathematics Subject Classification: 68W30, 65Y20, 68U05, 68P05

Springer Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the proceedings of the Ninth Asian Symposium on Computer Mathematics (ASCM 2009), held at the JAL Resort Sea Hawk Hotel, Fukuoka, Japan, December 14–17, 2009, and the proceedings of the Tenth Asian Symposium on Computer Mathematics (ASCM 2012), held at the Chinese Academy of Sciences, Beijing, China, October 26–28, 2012. In both conferences, the contributed papers were selected by the Program Committee for presentation at the symposium and went through a standard refereeing process after the symposium. Both Program Committees had strong Asian participation, and the reviewing process was aided by reviewers from around the world.

The ASCM 2009 was jointly held with the Third International Conference on Mathematical Aspects of Computer and Information Sciences (MACIS 2009). The invited speakers at the joint conference were Markus Rosenkranz, Toshinori Oaku, Kokichi Sugihara, and Lihong Zhi. The ASCM 2012 had three excellent plenary talks delivered by Erich Kaltofen, Markus Püschel, and Josef Schicho. This volume also contains the extended abstracts provided by Erich Kaltofen and Josef Schicho after the symposium.

In addition to the regular sessions, both ASCM 2009 and ASCM 2012 had organized sessions presenting research of selected topics. The three organized sessions of ASCM 2009 were "Digitizing Mathematics: From Pen and Paper to Digital Content" organized by Volker Sorge and Alan P. Sexton, "Validated Numerical Computation" by Mitsuhiro Nakao, and "Computational Algebraic Number Theory" by Guenael Renault. The four organized sessions of ASCM 2012 were "On the Latest Progress in Verified Computation" organized by Lihong Zhi, "Computational Geometry" by Jin-San Cheng, "Parametric Polynomial Computations" by Yosuke Sato, and "Differential and Difference Algebra" by Chun-Ming Yuan.

We would like to express our gratitude to all those who have contributed to the present volume and the organization of ASCM 2009 and ASCM 2012. We thank the authors of the papers for contributing their work and the conference participants for their presence. We also thank the organizers of the organized sessions for presenting selected research topics and the invited speakers for accepting our

invitations. We are very grateful to the Program Committee members and the reviewers for their time and efforts in evaluating the submissions before and after each conference. We especially thank both teams for the local arrangements, who made the conferences successful and enjoyable. Last but not least, we thank Ziming Li for his valuable assistance in the conference organization and the publication of the proceedings. ASCM 2009 thanks Math-for-industry and Mathematical Research Center for Industrial Technology at Kyushu University, Ehime Campus Information Service, Cybernet, Japan Society for Symbolic and Algebraic Computation, Maplesoft, and JAL Hotels for their financial support. ACSM 2012 gratefully acknowledges the generous support of the Academy of Mathematics and Systems Science and the Key Laboratory of Mathematics Mechanization at the Chinese Academy of Sciences, the National Natural Science Foundation of China, Maplesoft, Cybernet Systems China.

Previous ASCM meetings were held in Beijing, China (1995), Kobe, Japan (1996), Lanzhou, China (1998), Chiang Mai, Thailand (2000), Matsuyama, Japan (2001), Beijing, China (2003), Seoul, Korea (2005), and Singapore, Singapore (2007). We hope that the ASCM continues to serve as a forum for participants to present original research, learn of research progress and developments, and exchange ideas and views on doing mathematics using computers.

Beijing, May 2014 — Ruyong Feng
Antwerp — Wen-shin Lee
Tokyo — Yosuke Sato

Conference Organization

Organizing Committee of ASCM2009

General Chair
Masakazu Suzuki, Kyushu University, Japan

Program Committee
Xavier Dahan, Kyushu University, Japan
Xiao-Shan Gao, Chinese Academy of Sciences, China
Deepak Kapur, University of New Mexico, USA
Ziming Li, Academy of Mathematics and Systems Science, China
Hirokazu Murao, The University of Electro Communications, Japan
Mitsuhiro Nakao, Kyushu University, Japan
Hyungju Park, KIAS, Korea
Guenael Renault, UPMC, INRIA, France
Ko Sakai, Tsukuba University, Japan
Yosuke Sato (co-chair), Tokyo University of Science, Japan
Alan P. Sexton, University of Birmingham, UK
Volker Sorge, University of Birmingham, UK
Gert Vegter, Groningen University, The Netherlands
Chee Yap (co-chair), New York University, USA

Local Arrangements
Tatsuyoshi Hamada, Fukuoka University/JST CREST, Japan
Koji Nakagawa, Kyushu University, Japan
Hiroshi Yoshida, Kyushu University, Japan

Organizing Committee of ASCM2012

General Chair
Ziming Li, Chinese Academy of Sciences, China

Program Committee
Shaoshi Chen, North Carolina State University, USA
Howard Cheng, University of Lethbridge, Canada
Ruyong Feng (co-chair), Chinese Academy of Sciences, China
Hoon Hong, North Carolina State University, USA
Wen-shin Lee (co-chair), University of Antwerp, Belgium
Kosaku Nagasaka, Kobe University, Japan
Chau Ngo, Quy Nhon University, Vietnam
Tateaki Sasaki, University of Tsukuba, Japan
Yosuke Sato, Tokyo University of Science, Japan
Luu Ba Thang, Hanoi National University of Education, Vietnam
Irem Yaman, Gebze Institute of Technology, Turkey
Zhengfeng Yang, East China Normal University, China
Chee K. Yap, New York University, USA
Jian Zhang, Chinese Academy of Sciences, China
Zhifang Zhang, Chinese Academy of Sciences, China

Local Arrangements
Jinsan Cheng, Chinese Academy of Sciences, China
Lei Huang, Chinese Academy of Sciences, China
Chunming Yuan (chair), Chinese Academy of Sciences, China
Daizhen Zhou, Chinese Academy of Sciences, China

Contents

Contributors

Yoshinori Aono NICT, Koganei, Japan

François Boulier LIFL, Université de Lille 1, Villeneuve D'Ascq Cedex, France

Brice Boyer Department of Mathematics, North Carolina State University, Raleigh, NC, USA

Changbo Chen ORCCA, University of Western Ontario (UWO), London, ON, Canada; Chongqing Key Laboratory of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China

Liangyu Chen Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China

Howard Cheng Department of Mathematics and Computer Science, University of Lethbridge, Lethbridge, Canada

Matthew T. Comer Department of Mathematics, North Carolina State University, Raleigh, NC, USA

Xiao-Shan Gao KLMM, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, People's Republic of China

Mark Giesbrecht Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada

Jingjun Han Beijing International Center for Mathematical Research and School of Mathematical Sciences, Peking University, Beijing, China

Hoon Hong Department of Mathematics, North Carolina State University, Raleigh, USA

Akinari Hoshi Department of Mathematics, Rikkyo University, Toshima-ku, Tokyo, Japan

Daiju Inaba Mathematics Certification Institute of Japan, Katsushika-ku, Japan

Shutaro Inoue Tokyo University of Science, Tokyo, Japan

Qingfang Jin Key Laboratory of Mathematics Mechanization, AMSS, Chinese Academy of Science, Beijing, China

Erich L. Kaltofen Department of Mathematics, North Carolina State University, Raleigh, NC, USA

T.W. Kelsey School of Computer Science, University of St Andrews, St Andrews, UK

Heinz Kredel IT-Center, University of Mannheim, Mannheim, Germany

George Labahn Symbolic Computation Group, David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

Na Lei School of Mathematics, Jilin University, Changchun, China

François Lemaire LIFL, Université de Lille 1, Villeneuve D'Ascq Cedex, France

Tuo Leng School of Computer Engineering and Science, Shanghai University, Shanghai, China

U. Leonhardt School of Physics and Astronomy, University of St Andrews, St Andrews, UK

Yaohui Li Department of Computer Science, Tianjin University of Technology and Education, Tianjin, People's Republic of China

S.A. Linton School of Computer Science, University of St Andrews, St Andrews, UK

Xuefeng Liu Research Institute for Science and Engineering, Waseda University, Shinjuku-ku, Tokyo, Japan

Zhuojun Liu Key Laboratory of Mathematics Mechanization, AMSS, Chinese Academy of Science, Beijing, China

Michael Monagan Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada

Marc Moreno Maza ORCCA, University of Western Ontario (UWO), London, ON, Canada

Akira Nagai NTT Information Sharing Platform Laboratories, Tokyo, Japan

Shin'ichi Oishi Faculty of Science and Engineering, Waseda University, Shinjuku-ku, Tokyo, Japan; CREST/JST, Saitama, Japan

Tomoaki Okayama Graduate School of Economics, Hitotsubashi University, Kunitachi, Tokyo, Japan

Roman Pearce Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada

Nam Pham Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada

Sanzheng Qiao Shanghai Key Laboratory of Contemporary Applied Mathematics, Department of Computing and Software, McMaster University, Hamilton, ON, Canada

Yuxue Ren School of Mathematics, Jilin University, Changchun, China

Tateaki Sasaki Institute of Mathematics, University of Tsukuba, Tsukuba, Japan

Josef Schicho RICAM Linz, Austrian Academy of Sciences, Vienna, Austria

Hiroshi Sekigawa Tokyo University of Science, Tokyo, Japan

Fei Shen LMAM and School of Mathematical Sciences, Peking University, Beijing, China

Liyong Shen School of Mathematical Science, University of Chinese Academy of Sciences, Beijing, China

Petr Sojka Faculty of Informatics, Masaryk University, Brno, Czech Republic

Yuqing Song Department of Computer Science, Tianjin University of Technology and Education, Tianjin, People's Republic of China

Yao Sun SKLOIS, Institute of Information Engineering, CAS, Beijing, China

Joris van der Hoeven LIX, CNRS, École polytechnique, Palaiseau Cedex, France

Dingkang Wang KLMM, Academy of Mathematics and Systems Science, CAS, Beijing, China

Dongming Wang Laboratoire D'Informatique de Paris 6, CNRS—Université Pierre et Marie Curie, Paris, France

Yimin Wei Shanghai Key Laboratory of Contemporary Applied Mathematics, School of Mathematical Sciences, Fudan University, Shanghai, People's Republic of China

Baofeng Wu Key Laboratory of Mathematics Mechanization, AMSS, Chinese Academy of Science, Beijing, China

Min Wu Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China

Wenyuan Wu Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China

Xiaoli Wu The School of Science, Hangzhou Dianzi University, Hangzhou, Zhejiang, China; Key Laboratory of Mathematics Mechanization, Chinese Academy of Sciences, Beijing, China

Zhifeng Wu Department of Computer Science, Tianjin University of Technology and Education, Tianjin, People's Republic of China

Bican Xia LMAM and School of Mathematical Sciences, Peking University, Beijing, China

Rong Xiao University of Western Ontario, London, ON, Canada

C. Xiong School of Computer Science, University of St Andrews, St Andrews, UK

Jing Yang LMIB and School of Mathematics and Systems Science, Beihang University, Beijing, China

Zhengfeng Yang Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China

Chun-Ming Yuan KLMM, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, People's Republic of China

Zhenbing Zeng Department of Mathematics, Shanghai University, Shanghai, China

Wen Zhang School of Mathematics and Physics, Qingdao University of Science and Technology, Qingdao, People's Republic of China

Xiaoming Zhang Key Laboratory of Mathematics Mechanization, AMSS, Chinese Academy of Science, Beijing, China

Yang Zhang Department of Mathematics, University of Manitoba, Winnipeg MB, Canada

Zhi-Yong Zhang KLMM, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, People's Republic of China

Xiaopeng Zheng School of Mathematics, Jilin University, Changchun, China

Part I
Invited Talks of ASCM2012

Symbolic Computation and Complexity Theory Transcript of My Talk

Erich L. Kaltofen

Keywords Computational complexity · Exponential-time algorithms · Practicality

1 The Setting

I gave talks at the conference *Alan Turing's Heritage: Logic, Computation & Complexity* in Lyon, France on July 3, 2012, at the Pierre and Marie Curie University (UPMC) Paris 6, France on July 17, 2012, and at the Tenth Asian Symposium on Computer Mathematics (ASCM) in Beijing, China, on October 26, 2012 on the complexity theoretic hardness of many problems that the discipline of symbolic computation tackles. Here is a brief transcript of part of those talks.

1.1 NP-Completeness and Beyond

A fundamental problem of symbolic computation, that of solving systems of polynomial equations, is easily shown to be NP-hard: $x \vee \neg y \quad \equiv \quad (1-x)\,y = 0, x(x-1) = 0, y(y-1) = 0$, which shows how to encode a clause in a satisfiability problem as polynomial equations.

This material is based on work supported in part by the National Science Foundation under Grants CCF-1115772.

E.L. Kaltofen (✉)
Department of Mathematics, North Carolina State University,
Raleigh, NC 27695-8205, USA
e-mail: kaltofen@math.ncsu.edu

R. Feng et al. (eds.), *Computer Mathematics*, DOI 10.1007/978-3-662-43799-5_1

Real geometry, when the solutions of the polynomial systems are restricted to real numbers, is by Tarski's algorithm decidable. However, Fischer and Rabin [1] have shown that the problem requires exponential space, $2^{\Omega(n)}$, where n is the number of variables in the polynomials. Furthermore, Mayr and Meyer [2] have extended the result to Polynomial Ideal Membership over the rationals, that is, they show $2^{\Omega(n)}$-space hardness.

Finally, Fröhlich and Shepherdson [3] shows that there are fields K in which the five arithmetic operations, namely addition, negation, multiplication, division, and equality testing are computable but where factorization in $\mathsf{K}[x]$ is undecidable ("unentscheidbar"). The proof is based on the infinite tower of extensions by squareroots of prime integers and the fact that $\sqrt{2} \notin \mathbb{Q}(\sqrt{3}, \sqrt{5}, \sqrt{7}, \sqrt{11}, \ldots)$, for instance.

These are indeed formidable computational complexity theoretic barriers to the discipline of symbolic computation.

1.2 Early Symbolic Computation Algorithms

Buchberger's famous 1965 Gröbner basis algorithm tackles exactly these hard problems: it decides ideal membership and computes solutions to polynomial systems. Berlekamp's and Zassenhaus's 1968 polynomial factorization algorithms work for coefficients in finite fields and the rational numbers. Collins's 1974 cylindrical algebraic decomposition algorithm performs Tarski's quantifier elimination.

The pursuit of symbolic computation algorithms that solved these computational hard problems in the early 1980s was ridiculed by some theorists as hopeless.

2 Cook's Thesis

In his plenary talk at the ICM in Kyoto Cook [4], three function classes were introduced:
polytime, the functions computable by polynomial-time algorithms,
NAT, functions arising from natural computational problems,
PracSolv, functions computable on an actual computer on **all inputs** of 10,000 bits or less.

Stephen Cook then formalizes his thesis:

Thesis: $PracSolv \cap NAT = polytime \cap NAT$.

The notion that *polytime*captures the domain of efficiently computable functions is ingrained in theoretical computer science. Many reductions in modern theoretical cryptography make use of the device. Stephen Cook refines the notion to natural functions. As an unnatural function he gives the example of $2^{\lceil \log n \rceil^{1000}} \in polytime$, that with high exponent of 1,000, and which he excludes from *NAT*.

As evidence in 1990 there were polynomial-time algorithms for linear programming and for polynomial factorization over rational numbers. Polynomial identity testing via the Zippel-Schwartz lemma was and is random polynomial time, and at that time it was hypothesized that with randomization in polynomial-time algorithms one could not reach beyond the class *polytime*. Today, results of Impagliazio and Kabanets raise more doubts. Indeed, already in 1990 polynomial factorization, even for polynomials in many variables, was known to be in randomized *polytime* Kaltofen and Trager [5]. Supersparse polynomial factorization algorithms would follow 15 years later Kaltofen and Koiran [6].

Richard D. Jenks had expressed some doubt in the thesis at that time, telling me as PhD student: *You prove that problems are hard and I write computer programs that solve them.* In the following I will attempt to challenge Cook's Thesis.

The Thesis can fail in two directions. There may exist an $f \in \mathit{PracSolv} \cap \mathit{NAT}$ but $f \notin \mathit{polytime}$. Many programs in symbolic computation can produce outputs to problems that are, in the worst case, hard.

1. Proofs that a positive semidefinite polynomial is not a sum-of-squares: 462-dimensional linear matrix inequalities (LMI) with 7,546 variables Guo, Kaltofen and Zhi [7]. Semidefinite programming constitutes a far-reaching generalization to linear programming with a limited nonlinearity in its control parameters: the solution must remain a definite matrix.
2. Large Gröbner basis problems, e.g., compact McEliece crypto system: 115 variables, 193,584 equations, degrees $= 2, 3, \ldots, 256$ Faugére et al. [8].
3. Proofs that certain nonlinear polynomial equation problems (LMIs) do not have a rational solution, while they have a real solution Guo, Din and Zhi [9]. I have substituted this diophantine problem to my list for this paper; in my talk I listed a problem in real algebraic geometry with 4 variables.

The above examples do not violate Cook's Thesis. Clearly, a superpolynomial-time algorithm can have polynomial-time running time on a subset of inputs. Many algorithms in symbolic computation, such as Buchberger's algorithm and its modern variant *FGb*, have unpredictable running time. Cook's *NAT* is the class of "natural" functions, not "natural" inputs. It is important for algorithmic infrastructure to know the worst-case behavior of an algorithm: Google returns a list of ranked pages for all queries. Nonetheless, it is the hallmark of the symbolic computation discipline to have greatly enlarged the domain of natural and solvable inputs to hard problems.

The Thesis can also fail in the opposite: We may have an $f \in \mathit{polytime} \cap \mathit{NAT}$ but $f \notin \mathit{PracSolv}$. It is actually not so easy to find a natural problem in symbolic computation that is in *polytime* but whose worst-case complexity is super-quadratic in its input size. I offer three examples:

1. The characteristic polynomial of a sparse matrix $\in \mathbb{Z}_p^{n\times n}$ with $O(n)$ nonzero entries is notoriously difficult to compute with $O(n)$ auxiliary space in the worst case. The best algorithm is of $n^{2+1/2+o(1)}$-time, $O(n)$-space Villard [10]. For $n = 10^6$ this is, ignoring the implied $n^{o(1)}$, $\geq 10^9\times$ input size. We restrict to

$O(n)$ space because by using quadratic space one has $O(n^{2.38})$ time with fast matrix multiplication, although the solution is quite impractical.
2. The Sylvester resultant in x of $f(x, y), g(x, y)$ can be computed by the half-GCD algorithm in $\max\{\deg_x(f), \deg_x(g)\}^{2+o(1)} \times \max\{\deg_y(f), \deg_y(g)\}^{1+o(1)}$ scalar operations.
3. Lattice basis reduction is polynomial-time, but the dependency on the dimension may be superquadratic. I have substituted this diophantine problem to my list for this paper; in my talk I listed sharp estimates for the structured condition numbers of Hankel matrices.

In sparse/structured linear algebra, $O(n \log(n))$ versus $O(n^2)$ running time makes all the difference, for example in discrete Fourier transform algorithms. Polynomial factorization is again a forerunner: polynomials modulo 2 can be factored in sub-quadratic time since Kaltofen and Shoup [11].

Shaoshi Chen mentioned while he was at NCSU that some of the algorithms for symbolic summation have worst-case performance beyond quadratic. Those and the above are all candidates for polynomial-time problems that are not practically solvable for large inputs, although one may require much more than Cook's original 10,000 bits for the inputs. It is my conclusion that the Thesis fails on that side: $\mathit{PracSolv} \cap \mathit{NAT} \subsetneq \mathit{polytime} \cap \mathit{NAT}$.

Stephen Cook at the Turing Centennial celebration in San Francisco in June 2012 suggested to me to consider in place of *polytime* the class of *logarithmic space* as the practical one (in my talk I stated it as *poly-logarithmic space*).

3 Faugère's Question

After my talk in Paris, Jean-Charles Faugère asked me the following question: Does it make sense to study **and implement** algorithms that have exponential running time? My answer was "no," with some disapproval from the audience. I clarified that one may study algorithms that are worst-case exponential, but that run polynomial-time on the inputs studied. Executing an exponential-time algorithm, say a combinatorial search, constitutes a single computation, not providing an algorithm for general use.

I should add Ludovic Perret's comment to me at NCSU in October 2012: One studies exponential algorithms to know their run times, for example when choosing size of a key in a crypto scheme.

References

1. Fischer, M.J., Rabin, M.O.: Super-exponential complexity of presburger arithmetic. In: R. M. Karp, editor, Complexity of Computation, pp. 27–41. Amer. Math. Soc. (1974)
2. Mayr, E.W., Meyer, A.R.: The complexity of the word problem for commutative semigroups and polynomial ideals. Advances Math. **46**, 305–329 (1982)

3. Fröhlich, A., Shepherdson, J.C.: Effective procedures in field theory. Phil. Trans. Roy. Soc. Ser. A 248, 407–432 (1955/1956)
4. Cook, S.A.: Computational complexity of higher type functions. In: Proceedings of the ICM Kyoto Japan, ICM Series, pp. 55–69 (1990)
5. Kaltofen, E., Trager, B.: Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. J. Symbolic Comput., 9(3):301–320 (1990) URL: http://www.math.ncsu.edu/kaltofen/bibliography/90/KaTr90.pdf
6. Kaltofen, E., Koiran, P.: Finding small degree factors of multivariate supersparse (lacunary) polynomials over algebraic number fields. In: Jean-Guillaume Dumas, editor, ISSAC MMVI Proceedings of the 2006 International Symposium Symbolic Algebraic Computation, pp. 162–168, New York, N. Y., 2006. ACM Press. ISBN 1-59593-276-3. URL: http://www.math.ncsu.edu/kaltofen/bibliography/06/KaKoi06.pdf
7. Guo, F., Kaltofen, E.L., Zhi, L.: Certificates of impossibility of Hilbert-Artin representations of a given degree for definite polynomials and functions. In: Joris van der Hoeven and Mark van Hoeij, editors, ISSAC 2012 Proceedings of the 37th International Symposium Symbolic Algebraic Computation, pp. 195–202, New York, N. Y., July 2012. Association for Computing Machinery. ISBN 978-1-4503-1269. URL: http://www.math.ncsu.edu/kaltofen/bibliography/12/GKZ12.pdf;URL: http://arxiv.org/abs/1203.0253
8. Faugère, J.-C., Otmani, A., Perret, L., Tillich, J.-P.: Algebraic cryptanalysis of McEliece variants with compact keys. In: Proceedings of the Eurocrypt 2010, volume 6110 of Lecture Notes in Computer Science, pp. 279–298, Heidelberg, Germany, 2010. Springer Verlag
9. Guo, Q., Safey El Din, M., Zhi, L.: Computing rational solutions of linear matrix inequalities. In: Manuel Kauers, editor, ISSAC 2013 Proceedings of the 38th International Symposium Symbolic Algebraic Computation, New York, N. Y., 2013. Association for Computing Machinery
10. Villard, G.: Computing the Frobenius normal form of a sparse matrix. In: Proceedings of the Third International Workshop on Computer Algebra in Scientific Computing, pp. 395–407, Heidelberg, Germany, 2000. Springer Verlag
11. Kaltofen, E., Shoup, V.: Subquadratic-time factoring of polynomials over finite fields. Math. Comput., 67(223):1179–1197, July 1998. URL: http://www.math.ncsu.edu/kaltofen/bibliography/98/KaSh98.pdf

Factorization of Motions

Josef Schicho

1 Extended Abstract

We define *motion polynomials* as polynomials with coefficients in the dual quaternions and study their factorizations. The motion polynomials correspond to motions in 3D space, and factoring into linear factors means to compose the motion into translations and rotations. This allows to realize the motion by a linkage with revolute or prismatic joints. This is joint work with G. Hegedüs (Univ. Oboda), Z. Li (RICAM), and H.-P. Schröcker (Univ. Innsbruck). The results are published in [1]. This research has been supported by the Austrian Science Fund (FWF): DK W 1214-N15.

Let $\mathbb{H} = \langle 1, \mathbf{i}, \mathbf{j}, \mathbf{k} \rangle_{\mathbb{R}}$ be the skew field of quaternions. It is well known [2] that $\mathbb{H}$ is algebraically closed, in the sense that every univariate left polynomial $P \in \mathbb{H}[t]$ can be written as a product of linear polynomials. Here, the variable t is supposed to commute with the coefficients. To decompose P, one looks for right zeroes in $\mathbb{H}$: if $P(q) = 0$, then $(t - q)$ is a right factor of P, and the polynomial quotient has degree one less.

In order to find right zeroes, we compute the norm polynomial $N(t) = P(t)\overline{P}(t)$, where $\overline{P}$ is obtained by conjugating all coefficients. It is a real polynomial that does not assume negative values when evaluated at real numbers. Generically, it has no real zeroes, so that it can be written as a product of irreducible quadratic factors. For any such factor Q, there is a unique common right zero of P and Q in $\mathbb{H}$, and this common right zero can be computed by polynomial division: the polynomial remainder of P mod Q is linear.

The factorization algorithm can be extended to skew ring $\mathbb{DH} = \langle 1, \mathbf{i}, \mathbf{j}, \mathbf{k} \rangle_{\mathbb{D}}$ of dual quaternions, where $\mathbb{D} = \mathbb{R} \oplus \mathbb{R}\epsilon$ is the two-dimensional $\mathbb{R}$-algebra generated by $\mathbb{R}$ and ϵ with $\epsilon^2 = 0$. We are especially interested in polynomials with real

J. Schicho (✉)
RICAM Linz, Austrian Academy of Sciences, Vienna, Austria
e-mail: josef.schicho@ricam.oeaw.ac.at

R. Feng et al. (eds.), *Computer Mathematics*, DOI 10.1007/978-3-662-43799-5_2

norm polynomials; these polynomials are called *motion polynomials*. Since $\mathbb{D}$ is not a field, the algorithm may sometimes fail, so that there exist polynomials in $\mathbb{DH}[t]$ without factorization into linear ones. For generic motion polynomials of degree d, the algorithm works, and one gets $d!$ different factorizations into linear motion polynomials. (In contrast to the commutative case, it is not allowed to permute the factors.)

The special interest in motion polynomials comes from a well-known isomorphism of the six-dimensional Lie group SE_3 of Euclidean displacements and the multiplicative group of dual quaternions with nonzero real norm modulo multiplication by nonzero real scalars. Motions are curves in SE_3, and in this sense motion polynomials parameterize motions. Conversely, every motion that has a parameterization by rational functions can also be parameterized by a motion polynomial.

Linear motion polynomials parameterize revolutions around a fixed axis or translational pushes in fixed directions. Hence the factorization into linear motion polynomials decomposes the parameterized motion into revolutions or pushes, and the motion can be realized by a chain of revolute or prismatic joints.

A generic quadratic motion has two factorizations into two revolutions. The two chains of revolute joints can be combined to a movable closed chain with four links and four revolute joints. This linkage is called the Bennett linkage after its discoverer Bennett [3].

For $d > 2$, a generic motion of degree d can be decomposed into d revolutions in $d!$ different ways. Again, it is possible to combine the corresponding chains into one linkage. For instance, for $d = 3$ we obtain a movable linkage with 8 links connected by 12 revolute joints. Since the 6 decompositions are in relation to the permutations of the 3 irreducible factors of the norm polynomial of the parameterizing cubic motion polynomial, and the group of permutations is generated by transpositions, one can construct all decompositions by composing two neighboring revolutions and decomposing in the second way, as above. We call this operation *Bennett flip*.

By multiplying linear motion polynomials and applying Bennett flips, one can construct various families of closed overconstrained linkages. For instance, let us multiply two linear motion polynomials parameterizing revolutions around the same axes and a third linear motion polynomial; then we do two Bennett flips and construct a closed 5R linkage. This linkage is called the Goldberg 5R linkage after its discoverer [4]. It was shown in [5] that the Goldberg 5R linkage is the only movable 5R linkage with all 5 joints actually moving that is neither planar nor spherical.

Similar constructions lead to various families of 6R linkages, some well known, and some new. It should be mentioned that the classification of closed 6R linkages is a famous open problem in kinematics. Not all known families have a motion that can be rationally parameterized. An upper bound for the genus of the motion of a 6R linkage is given in [6].

References

1. Hegedüs, G., Schicho, J., Schröcker, H.-P.: Factorization of rational curves in the study quadric and revolute linkages. Mech. Mach. Theory **69**(1), 142–152 (2013)
2. Gordon, B., Motzkin, T.S.: On the zeros of polynomials over division rings. Trans. Amer. Math. Soc. **116**, 218–226 (1965)
3. Bennett, G.T.: The skew isogramm-mechanism. Proc. London Math. Soc. (2nd Series) 13, 151–173 (1913–1914).
4. Goldberg, M.: New five-bar and six-bar linkages in three dimensions. Trans. ASME **65**, 649–656 (1943)
5. Karger, A.: Classification of 5r closed kinematic chains with self mobility. Mech. Mach. Th. **33**, 213–222 (1998)
6. Hegedüs, G., Schicho, J., Schröcker, H.-P.: The theory of bonds II: Closed 6R linkages with maximal genus. Technical Report, arxiv 1309.6558 (2013)

Part II
Contributed Papers of ASCM2009

Simplification of the Lattice Based Attack of Boneh and Durfee for RSA Cryptoanalysis

Yoshinori Aono

Abstract We present a new formulation and its simpler analysis of the lattice-based attack of Boneh and Durfee for the RSA cryptography [1]. We follow the same approach as theirs, however, we propose a new way of defining a lattice with which we can achieve the same solvable key bound $d < N^{0.292}$. Our lattice is represented as a lower triangle matrix, which makes its analysis much simpler than that of [1]. We think that our analysis technique would be useful for considering applications/generalizations of this approach.

1 Introduction

Boneh and Durfee [1] proposed a polynomial time attack by which we can recover the RSA secret key d from the public information (e, N) when $d < N^{0.292}$; which we will refer as the *Boneh-Durfee* bound. The basic idea of the attack is based on the Coppersmith technique [3] by which we can obtain small solutions of a modular equation such as $f(x_1, x_2, \ldots, x_n) \equiv 0 \pmod{W}$. The technique converts the problem of finding small solutions of the equation into the problem of solving an algebraic equation by a lattice reduction algorithm.

Here is a detailed explanation of their approach. The goal is to obtain a small solution (x_0, y_0) of the following target equation to recover the secret key:

$$f_{\mathrm{BD}}(x, y) = -1 + x(y + A) \equiv 1 \pmod{e} \tag{1}$$

Here $A = N + 1$. From this, the following bivariate polynomials are defined:

$$g_{i,j}(x, y) = \begin{cases} x^{i-j}(f_{\mathrm{BD}}(x, y))^j e^{m-j} & \text{for} \quad i \geq j \\ y^{j-i}(f_{\mathrm{BD}}(x, y))^i e^{m-i} & \text{for} \quad i < j \end{cases} \tag{2}$$

for a certain range of (i, j) and a fixed integer m. These polynomials are converted to a lattice represented by a row echelon matrix L_{BD} defined by using the coefficients

Y. Aono (✉)
NICT, Koganei, Japan
e-mail: aono@nict.go.jp

R. Feng et al. (eds.), *Computer Mathematics*, DOI 10.1007/978-3-662-43799-5_3

of $g_{i,j}(x, y)$ with some parameters. Then by using a lattice reduction algorithm, we obtain a system of polynomial equations from which we can compute polynomial number of candidates of the solution (x_0, y_0) numerically.

In this approach a technically crucial point is to design a matrix for a lattice with a small determinant. They showed that their matrix has a sufficiently small determinant; however, its analysis is complicated since the technique is of geometrically progressive matrices, and it seems hard to apply for the other situations. The purpose of this paper is to give a new way to construct a lattice with asymptotically the same determinant that is much simpler to analyze.

Since Boneh and Durfee's work, variants of their technique have been proposed. Blömer and May [2] proposed a new lattice-based algorithm for attacking RSA with a short secret key. They constructed a lower triangle lattice by eliminating some columns from the original lattice; this simplifies the determinant analysis. In [7], Jochemsz and May gave an algorithm for finding small roots of a multivariate modular/integer equation based on a generalized lattice construction strategy. Note that both algorithms achieve a slightly weaker solvable key bound than the Boneh–Durfee bound.

In this paper we follow the strategy of Boneh and Durfee to give a new variation of the lattice-based attack with a simpler analysis. We propose a conversion from the polynomials (2) to three-variable polynomials $G_{i,j}(x, y, z)$ when we construct lattice; on the other hand, Boneh and Durfee directly constructed the lattice from $g_{i,j}(x, y)$. Since we obtain a lower triangle matrix representation of our lattice, we can easily compute its determinant. Therefore, we give a new simple algorithm to achieve the Boneh–Durfee bound. We remark that the same idea was independently found by Herrmann and May [6]; they referred their technique as "llinearization" and applied it for analyzing an attack for RSA-CRT.

We carry out our computer experiments to compare the qualities of our lattice and that of Boneh and Durfee. We check the solvable key ranges, the determinants, and the length of obtained vectors by L^2 algorithm [9] on lattice generated by these two algorithms. As shown in Sect. 5, we confirm that the qualities of the two lattice series are equivalent for various parameters in practice. We find the computational time of the L^2 algorithm is reduced by about 30 % from the original attack of Boneh and Durfee.

This paper is organized as follows: In Sect. 2, we give basic symbols, notations, lemmas. We give our formulation of the lattice-based attack in Sect. 3 and its detailed analysis is explained in Sect. 4. The computer experiments to compare our lattice construction and that in [1] are described in Sect. 5.

2 Preliminaries

In this section, for the following discussions, we introduce some notations and state some known facts and key technical lemmas.

We use the standard RSA notations throughout this paper. A given RSA instance is defined by p, q, e, and d, where p and q are large primes, e is a public key, and d is the

corresponding secret key. Let $N = p \times q$, and assume that $\gcd(e, (p-1)(q-1)) = 1$. The key relation between e and d is

$$ed \equiv 1 \pmod{(p-1)(q-1)} \tag{3}$$

from which we derive our target Eq. (1) by following the argument in [1].

The basic strategy of the lattice-based attack is to convert the problem of recovering RSA key into the problem of finding small solution of a modular equation within the certain range. In general, solving modular equation is not easy, whereas there are some cases where we may be able to use the standard numerical method for solving this problem. The Howgrave–Graham lemma [5] provides us with one of such cases.

To state the Howgrave–Graham lemma, we introduce the notion of *XY-norm* for a bivariate polynomial $f(x, y) = \sum_{i,j} a_{i,j} x^i y^j$ and integers X and Y by

$$||f(x, y)||_{XY} \stackrel{\text{def}}{=} \sqrt{\sum_{i,j} a_{i,j}^2 X^{2i} Y^{2j}}.$$

Lemma 1 (Howgrave–Graham [5]) *For any positive integers X, Y and W, let $f(x, y)$ be a bivariate polynomial consisting of w terms with integral coefficients such that the following holds:*

$$||f(x, y)||_{XY} < W/\sqrt{w}.$$

Then we have for x and y satisfying $|x| < X$ and $|y| < Y$,

$$f(x, y) \equiv 0 \pmod{W} \Leftrightarrow f(x, y) = 0.$$

Note that $f(x, y) = 0$ clearly implies $f(x, y) \equiv 0 \pmod{W}$. What is important is its converse. This lemma guarantees that we can find all solutions of the modular equation within the range from the integer solutions of $f(x, y) = 0$.

Now we introduce some definitions and lemmas about the lattice; we need to obtain a polynomial having small XY-norm to use Lemma 1, and this problem can be reduced to a problem of finding short vectors in a lattice.

Consider linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$, then the lattice with basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is defined as

$$L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{a_1\mathbf{b}_1 + a_2\mathbf{b}_2 + \cdots + a_n\mathbf{b}_n | a_i \in \mathbb{Z} \quad \text{for} \quad i = 1, \ldots, n\}. \tag{4}$$

That is, the lattice is the set of integral linear combinations of its basis vectors. We often represent the lattice $L(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ by the matrix $\begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{pmatrix}$.

To find short vectors in a lattice, it can be used a lattice reduction algorithm; here we use the LLL algorithm [8] in our analysis. The two short vectors in the LLL reduced basis described in the following theorem are important.

Theorem 1 *[1, Fact 3.3] Let* $\mathbf{b}_1, \ldots, \mathbf{b}_n$ *be a lattice basis. Then the LLL algorithm can find linearly independent lattice vectors* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *such that*

$$|\mathbf{v}_1| \leq 2^{(n-1)/4} |\det(L)|^{1/n} \text{ and } |\mathbf{v}_2| \leq 2^{n/2} |\det(L)|^{1/(n-1)}. \tag{5}$$

Here, $\det(L)$ is the determinant of the lattice which is defined by the determinant of a matrix representation of the lattice; note that suppose we have a lower triangle matrix representation of a lattice, $\det(L)$ is easily calculated by the product of its diagonal elements.

In the lattice-based attack, it needs to convert polynomials to vectors; since a lattice reduction algorithm is designed for vectors, while our targets are polynomials. Then we introduce a mapping.

We divide this mapping into two steps, named a *vectorisation* and an *instantiation* respectively. We introduce a way to map three-variable polynomials to vectors since we consider three-variable polynomials in our construction.

Definition 1 (*Polynomials* $\Rightarrow$ *vectors*)
Let $\mathbf{K}$ be a finite sequence of distinct three-variable monomials; let its order be fixed, and for any t, $x^{i_t} y^{j_t} z^{k_t}$ be the tth monomial in this order. Then for any $f(x, y, z) = \sum_{1 \leq t \leq |\mathbf{K}|} a_t x^{i_t} y^{j_t} z^{k_t}$, we map it to the following vector $\mathbf{b}$, which is called the *vectorization* of $f(x, y, z)$ and is denoted as $\mathscr{V}_{\mathbf{K}}(f)$.

$$\begin{array}{rl} f(x,y,z) = & a_1 x^{i_1} y^{j_1} z^{k_1} + a_2 x^{i_2} y^{j_2} z^{k_2} + \cdots + a_{|\mathbf{K}|} x^{i_{|\mathbf{K}|}} y^{j_{|\mathbf{K}|}} z^{k_{|\mathbf{K}|}} \\ & \downarrow \qquad\qquad \downarrow \qquad\qquad\qquad\qquad \downarrow \\ \mathbf{b} = & (a_1 x^{i_1} y^{j_1} z^{k_1}, \quad a_2 x^{i_2} y^{j_2} z^{k_2} \quad , \ldots, \quad a_{|\mathbf{K}|} x^{i_{|\mathbf{K}|}} y^{j_{|\mathbf{K}|}} z^{k_{|\mathbf{K}|}}). \end{array}$$

For example the polynomial $f(x, y, z) = -3x^3 + 4x^2yz - 2xy^2z^2 + 7xy^3z^3$ is mapped to the vector $(-3x^3, 4x^2yz, -2xy^2z^2, 7xy^3z^2)$.

We introduce a conversion named an instantiation and its inverse; it converts a three-variable monomials to integers by substitution. Our matrix is defined using the vectorizations and hence each element of the matrix is monomial. On the other hand, a lattice reduction algorithm is designed for integer lattices or integer matrices. Thus, for using a lattice reduction algorithm, we need to instantiate our matrix by substituting some integers X, Y and Z to x, y and z, which we call an *instantiation* with X, Y and Z. Conversely, converting an integer vector to a polynomial is called a *deinstantiation*. Note that (since $\mathbf{K}$ and the order of monomials is fixed) we know a monomial $x^{i_t} y^{j_t} z^{k_t}$ corresponding to the tth entry of a given vector; hence, deinstantiation at the tth entry can be achieved by simply dividing its integral value by $X^{i_t} Y^{j_t} Z^{k_t}$.

These vectorization, instantiation, and deinstantiation procedures are essentially the same as those used by some published works such as Boneh and Durfee.