On the complexity of the extended Euclidean algorithm

(Extended Abstract)

George Havas
Centre for Discrete Mathematics and Computing
School of Information Technology and Electrical Engineering
The University of Queensland, Queensland 4072, Australia

Euclid's algorithm for computing the greatest common divisor of 2 numbers is considered to be the oldest proper algorithm known ([10]). This algorithm can be amplified naturally in various ways. The GCD problem for more than two numbers is interesting in its own right. Thus, we can use Euclid's algorithm recursively to compute the GCD of more than two numbers. Also, we can do a constructive computation, the so-called extended GCD, which expresses the GCD as a linear combination of the input numbers.

Extended GCD computation is of particular interest in number theory (see [1, chapters 2 and 3]) and in computational linear algebra ([3, 4, 9]), in both of which it takes a basic role in fundamental algorithms. An overview of some of the earlier history of the extended GCD is given in [1], showing that it dates back to at least Euler.

Motivated by many efforts to find good algorithms for extended GCD computation, Majewski and Havas ([12]) showed that this is genuinely difficult. There are a number of problems for which efficient solutions are not readily available.

Theorem 1.
Given set A of positive integers and positive integer K, the problem "Does A contain a subset R such that $|R| \leq K$ and $\gcd(r \in R) = \gcd(a \in A)$?" is NP*-complete.*

Theorem 2.
Given set A of positive integers and positive integer K, the problem "Does

there exist a set of multipliers $\{x_i\}$ *such that no* x_i *is larger in magnitude than* K *and* $\sum_{a_i \in A} x_i a_i = \gcd(a \in A)$ *?" is* NP*-complete.*

On the other hand, if we are willing to accept an upper bound on the quality of the solution which depends on the size of the input numbers instead of on the size of the optimal solution we observe the following result.

Theorem 3.
There is an optimal time, optimal space algorithm for computing the extended GCD of n *integers* $\{a_1, \ldots, a_n\}$*, which guarantees that no multiplier is larger than the largest of the numbers divided by 2.*

Even though Theorem 1 tells us that finding a sparsest possible solution to the extended GCD problem is NP-complete, the problem is almost always easy ([5]). There exists a class of problems, AP, which, although NP-complete, have average polynomial time complexity. This problem is in AP.

Theorem 3 can be improved as follows [2].

Theorem 4.
Let $\{a_1, \ldots, a_n\}$ *be a multiset of positive integers.*
Let $m = \max(a_1, \ldots, a_{n-1})$ *and let* $g_k = \gcd(a_k, \ldots, a_n)$ *for* $1 \leq k \leq n$*.*
Then there exists an integer solution to the equation

$$x_1 a_1 + \cdots + x_n a_n = g_1$$

which satisfies

$$-\frac{g_{j+1}}{2g_j} < x_j \leq \frac{g_{j+1}}{2g_j} \text{ for } 1 \leq j \leq n-1, \tag{1}$$

$$|x_n| \leq \max\left(\frac{m}{2g_1}, 1\right). \tag{2}$$

Theorem 4 is realized by a practical algorithm. Furthermore, it is optimal in the sense that the bounds given in the Theorem for the multipliers can be approached as closely as desired by any individual multiplier. It is not worse than about the square of the best possible bound for distinct numbers, since a general lower bound for the Euclidean norm of the multiplier vector in terms of the initial numbers a_i must be at least $O(\sqrt{\max\{a_i\}})$ (see [6]).

Let us return to the hard problems. One way of addressing hard problems is to seek approximately optimal solutions instead of optimal solutions (see [8]). Consider the ℓ_p-norm shortest extended GCD multiplier problem [7]:

SHORTEST GCD MULTIPLIER in ℓ_p-norm (SGCDM$_p$)
Instance: n numbers $a_1, \ldots, a_n \in \mathbb{Z}$
Objective: Find a vector $\mathbf{x} \in \mathbb{Z}^n$ such that $\sum_{i=1}^{n} x_i a_i = \gcd(a_1, \ldots, a_n)$ and such that the ℓ_p-norm $\|\mathbf{x}\|_p := (\sum_{1 \le i \le n} |x_i|^p)^{1/p}$ of the vector $\mathbf{x}$ is minimized.

We have the following results:

Theorem 5.
1. *Unless* NP $\subseteq$ P, *there exists no polynomial-time algorithm which approximates the* SHORTEST GCD MULTIPLIER *problem in* ℓ_p*-norm within a factor of* k, *where* $k \ge 1$ *is an arbitrary constant.*
2. *Unless* NP $\subseteq$ DTIME$(n^{\mathrm{poly}(\log n)})$, *there exists no polynomial-time algorithm which approximates the* SHORTEST GCD MULTIPLIER *problem in* ℓ_p*-norm within a factor of* $n^{1/(p \log^{\gamma} n)}$, *for* γ *an arbitrary small positive constant.*

Corollary 6.
For every ℓ_p*-norm, the Extended GCD problem is* NP*-complete (in its feasibility recognition form).*

How well can we approximate optimal solutions in polynomial time? Approximating the solution to the ℓ_2-norm problem within a factor of $\sqrt{n/O(\log n)}$ is very unlikely NP-hard.

Theorem 7.
The $\sqrt{n/O(\log n)}$-SHORTEST GCD MULTIPLIER *in* ℓ_2*-norm is not* NP*-hard unless the Polynomial-Time Hierarchy collapses to its second level.*

Using lattice basis reduction [11] an algorithm HMM has been developed and analyzed [6, 13]. It is polynomial time and achieves a factor of $2^{(n-1)/2}$.

Theorem 8.
On inputs $a_1, \ldots, a_n \in \mathbb{Z}$ *algorithm* HMM *computes in polynomial time a vector* $\mathbf{x}$ *with* $\langle \mathbf{x}, \mathbf{a} \rangle = \gcd(\mathbf{a})$ *satisfying*

$$\|\mathbf{x}\| \le 2^{(n-1)/2} \cdot opt_{\mathrm{SGCDM}_2}(a_1, \ldots, a_n).$$

It remains to be seen whether this substantial gap between around $\sqrt{n}$ and $2^{(n-1)/2}$ can be reduced.

Acknowledgments

The author was supported by the Australian Research Council.

References

[1] A.J. Brentjes, *Multi–dimensional continued fraction algorithms*, Mathematisch Centrum, Amsterdam 1981.

[2] G. Havas and D. Ford, *A new algorithm and refined bounds for extended GCD computation*, Algorithmic number theory, Lecture Notes in Computer Science 1122, 145–150, 1996.

[3] G. Havas and B.S. Majewski. *Integer matrix diagonalization*, Journal of Symbolic Computation, 24:399–408, 1997.

[4] G. Havas and B.S. Majewski, *Hermite normal form computation for integer matrices*, Congressus Numerantium, 105:184–193, 1994.

[5] G. Havas and B.S. Majewski. *A hard problem which is almost always easy*, Algorithms and Computation, Lecture Notes in Computer Science 1004, 216–223, 1995.

[6] G. Havas, B.S. Majewski and K.R. Matthews. *Extended GCD and Hermite normal form algorithms via lattice basis reduction*. Experimental Mathematics, 7:125–136, 1998; 8:205, 1999.

[7] G. Havas and J-P. Seifert. *The complexity of the extended GCD problem*, Mathematical foundations of computer science, Lecture Notes in Computer Science 1672, 103–113, 1999.

[8] D.S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, PWS Publishing, Boston, Mass., 1996.

[9] C.S. Iliopoulos. *Worst case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix*, SIAM Journal on Computing, 18:658–669, 1989.

[10] D.E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 3rd edition, 1998.

[11] A.K. Lenstra, H.W. Lenstra Jr., and L. Lovász. *Factoring polynomials with rational coefficients*, Math. Ann., 261:515–534, 1982.

[12] B.S. Majewski and G. Havas, *The complexity of greatest common divisor computations*, Algorithmic Number Theory, Lecture Notes in Computer Science 877, 184–193, 1994.

[13] W. Van Der Kallen, *Complexity of the Havas, Majewski, Matthews LLL Hermite normal form algorithm*, Journal of Symbolic Computation, 30:329–337, 2000.

In arithmetic and computer programming, the extended Euclidean algorithm is an extension to the Euclidean algorithm, and computes, in addition to the greatest common divisor (gcd) of integers a and b, also the coefficients of BÃ©zout's identity, which are integers x and y such that. This is a certifying algorithm, because the gcd is the only number that can simultaneously satisfy this equation and divide the inputs. It allows one to compute also, with almost no extra cost, the quotients of a and b by... Worst case complexity bounds on algorithms for computing the. canonical structure of ï¬nite abelian groups and the Hermite and Smith normal. forms of an integer matrix, SIAM Journal on Computing, 18:658â€“669, 1989.Â . The candidate set of for the th term of (12) is given by (28) Although the extended Euclidean algorithm is NP-complete [25], can be computed before detection. Furthermore, (28) is a one-to-one mapping. Bound-Intersection Detection for Multiple-Symbol Differential Unitary Spaceâ€“Time Modulation. What is the bit-complexity invloved in calculating the greatest common divisor of two n-bit values x and y using Euclids Extended algorithm i.e. the complexity in terms of n. I observed the following pattern while calculating the GCD using a standard Extended Euclid Algorithm in the worst cases for different bit size. The complexity in terms of the magnitude of the two values x and y is close to the value: Source.Â If a and b are N bits long, then in the worst case (Fibonacci pairs), the extended Euclidean algorithm will take O(N) iterations. Let f(N) be the cost of a single iteration. Certainly f(N) will be at least linear, but still polynomial, and nearly half of the iterations in each case will involve arguments at least N/2 bits long, so the total complexity will be in O(N * f(N)).