实对称双线性函数与多精度整数的快速乘法*)

王小非 洪 帆 汤学明 崔国华
(华中科技大学计算机学院 武汉 430074)

摘 要 多精度整数乘法运算的效率对公钥密码系统中的模乘、模幂的运算效率起着决定性的作用。Toom-Cook 算法是一类应用广泛的多精度整数的快速乘法算法,目前主要的研究方法是插值理论。本文利用实对称双线性函数和二次型的方法研究多精度整数的乘法和平方的快速计算,给出了 Toom-Cook 算法参数的所有代数表现形式和搜索快速算法的基本方法,提出了一些在实际应用中与目前已知结果相同或优于目前已知结果的快速乘法和平方算法。研究结果表明,利用实对称双线性函数和二次型表示 Toom-Cook 算法,更有利于判断算法的优劣程度和得到最优算法。

关键词 实对称双线性函数,二次型,多精度整数乘法,Toom-Cook 算法

Real Symmetric Bilinear Functions and Fast Multiplication of Multi-precision Integers

WANG Xiao-Fei HONG Fan TANG Xue-Ming CUI Guo-Hua
(College of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

Abstract The efficiency of multiplication of multi-precision integers determines that of modular multiplication and modular exponential algorithms in public key cryptographic systems. Toom-Cook algorithm is a kind of widely used fast multiplication algorithm for multi-precision integers, and current primary research method of this algorithm is the theory of interpolation. With real symmetric bilinear functions and quadratic form, the algebraic representation form of all Toom-Cook parameters and how to search a fast algorithm are provided in this paper, some practical multiplication and squaring algorithms are prior to or same as current best results. The research results show that it's more advantageous to analyzing the efficiency of algorithms and get the optimal algorithms with real symmetric bilinear functions and quadratic form than with interpolation.

Keywords Real symmetric bilinear function, Quadratic form, Multiplication of multi-precision integers, Toom-Cook algorithm

1 引言

多精度整数运算普遍使用于计算数论、密码学和高精度计算等领域。在多精度整数的四则运算中,加法和减法算法比较明了,效率也很高,而乘法和除法效率较低。如何优化多精度整数的乘、除法,一直是人们研究的热点。公钥密码学中最常用到的公钥密码算法,例如 Diffie-Hellman 密钥交换协议、RSA 公钥密码体制、有限域(特征为多精度素数)上的椭圆曲线密码体制等涉及大量的多精度整数运算,其中比较耗时的基本运算是乘法和求模。求模可以通过除法来实现,但除法效率较低。当一个 2048 比特的整数对一个 1024 比特的整数求模时,其运行效率仅为两个 1024 比特整数相乘运行效率的 10% 左右。1985 年,Montgomery[1] 提出了著名的 Montgomery 模乘算法,使得可以不用除法运算直接实现模乘。该算法将模运算转化成了乘法运算,模乘的效率大致相当于两次同数量级的整数乘法运算[2]。模乘(包括模平方)运算不仅本身在各领域的计算中广泛使用,而且是模幂运算的基础,因此快速计算模乘是提高这些算法运算速度的关键。而根据 Montgomery 模乘算法我们可以看出,提高模乘运算效率的关键是提高多精度整数的乘法运算效率。

当整数的位数为 n 时,传统的基于进制的乘法的计算复杂度为 $O(n^2)$。分治算法[3,4](也称 Karatsuba 算法)将双精度整数的乘法转化成三次一位整数乘法、少量的移位和加、减法,通过递归调用可以实现复杂度为 $O(n\log_2^3)$ 的乘法算法,计算机实现较为简单,是几千比特以内整数相乘最常用的方法。分治算法的更一般的情况是 Toom-Cook[4] 算法,如果每次将多精度整数分成 r+1 部分,利用插值原理,可以实现复杂度为 $O(n\log_{2^r+1}^{2^r+1})$ 的乘法算法。Toom-Cook 算法中用得比较多的是递归调用 $r=2$ 的情况,其算法复杂度为 $O(n\log_3^5)$,当 r 增大的时候,只有在 n 很大的情况下,算法才会体现出优势。文[5]专门研究了 $r=4,5,6$ 的情况下如何实现一些实用的乘法算法。Schönhage[4] 首先利用中国剩余定理来计算多精度整数的乘法,后来 Schönhage 和 Strassen[4,6] 利用环 $Z/(2^m+1)Z$(m 为 2 的幂)上的快速傅立叶变换(快速数论变换),实现目前已知的最好的复杂度为 $O(n\log_2 n\log_2\log_2 n)$ 的乘法算法。

本文利用实对称双线性函数来研究多精度整数的快速乘法算法。实对称双线性函数的一种特殊情形是二次型,二次型的理论适合研究多精度整数的快速平方算法。本文给出了 Toom-Cook 算法参数之间的矩阵关系,通过实对称矩阵的秩、行列式的值以及惯性指数等基本量,可以得到算法的构造条件以及如何判断算法的优劣。本文所给出的一些实际算法的例子,有的和目前已知的最好结果相当,有的优于目前已知的结果,表明新的研究方法更易于构造高效的乘法算法。

本文第 2 节介绍整数和实对称双线性函数以及二次型的一些记号和定义;第 3 节利用实对称双线性函数研究快速乘

*)本课题得到国家自然科学基金(60403027)、湖北省自然科学基金(2005ABA243)资助。王小非 博士研究生,主要研究方向为密码学和计算机安全;洪 帆 教授,博士生导师,主要研究方向为密码学、信息安全、访问控制、网络安全等;汤学明 博士研究生,主要研究方向为数论、密码学和计算机安全;崔国华 教授,博士生导师,主要研究方向为公钥密码学、数据库加密、信息安全。

法算法；第4节利用实二次型理论研究快速平方算法；第5节对所构造的算法和目前的一些结果做对比分析；最后对全文的主要结论做简单的回顾。

2 记号和定义

设 $U=(u_{n-1}\,u_{n-2},\cdots,u_1u_0)_B$，$V=(v_{n-1}\,v_{n-2},\cdots,v_1v_0)_B$ 表示两个无符号多精度整数，其中 B 为大于1的正整数，$0\leqslant u_i,v_i<B$，$U=\sum_{i=0}^{n-1}v_iB^i$，$V=\sum_{i=0}^{n-1}v_iB^i$ 当 u_{n-1} 和 v_{n-1} 不为0时，我们称它们为 B 进制 n 位整数，它们的乘积为

$$UV=\sum_{i=0}^{2n-2}w_iB^i \quad \text{其中，} w_i=\sum_{s+t=i}u_sv_t \tag{1}$$

当 $u_i=v_i$ 的时候，(1)式就表示整数的平方。乘法和平方运算的目的就是求出所有 w_i。

设 S 是实数域 R 上的 n 维线性空间，$f(\alpha,\beta)$ 是 S 上双线性函数，给定 S 的任意一组基，若 α,β 在这组基下的坐标为 $x_i,y_j(0\leqslant i,j\leqslant n-1)$，则 $f(\alpha,\beta)$ 可以表示为 $f(\alpha,\beta)=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}a_{i,j}x_iy_j=\alpha'A\beta$。其中，系数矩阵 A 为

$$A=\begin{pmatrix}a_{0,0}&a_{0,1}&\cdots&a_{0,n1}\\a_{1,0}&a_{1,1}&\cdots&a_{1,n1}\\\cdots&\cdots&\cdots&\cdots\\a_{n-1,0}&a_{n-1,1}&\cdots&a_{n-1,n-1}\end{pmatrix}$$

A 的秩 $r(A)$ 也称为双线性函数的秩。若 $f(\alpha,\beta)=f(\beta,\alpha)$，则称 $f(\alpha,\beta)$ 为 S 上的实对称双线性函数。进一步，若 $\alpha=\beta$，则称实对称双线性函数 $f(\alpha,\beta)$ 或其系数矩阵为一个 n 元实二次型。式(1)中的 U 和 V 的乘积的系数 w_i 可以看成是关于变量 $(u_{n-1},u_{n-2},\cdots,u_1,u_0)$ 和 $(v_{n-1},v_{n-2},\cdots,v_1,v_0)$ 的实对称双线性函数，它的系数矩阵是副对角线上元素全为1的实对称矩阵。当 $u_i=v_i$ 的时候，系数 w_i 就是一个 n 元实二次型。关于实对称双线性函数和一次齐式有如下的关系：

引理1[7] 实对称双线性函数可以表示成两个一次齐式的乘积的充要条件是它的秩为1；实二次型可以表示成两个一次齐式的乘积的充要条件是它的秩为1，或者秩为2且正惯性指数为1。

假设 $w'_i,0\leqslant i\leqslant k-1$ 是 k 个关于 $u_{n-1},u_{n-2},\cdots,u_1,u_0$ 和 $v_{n-1},v_{n-2},\cdots,v_1,v_0$ 的一次齐式的乘积，即 $w'_i=(\sum_{s=0}^{n-1}a_su_s)(\sum_{t=0}^{n-1}b_tv_t)$，$a_s,b_t\in R$，如果存在实线性变换矩阵 T 使得：

$$\begin{pmatrix}w_0\\w_1\\\cdots\\w_{2n-2}\end{pmatrix}=T\begin{pmatrix}w'_0\\w'_1\\\cdots\\w'_{k-1}\end{pmatrix} \tag{2}$$

则我们称在计算两个 n 位整数乘积的过程中共用了 k 次1位整数的乘法。这时，所有的 w_i 都可以通过这 k 个一位整数的乘积线性表示出来。本文的主要工作就是寻找个数尽量少、形式尽量简单的 w'_i，使得它们可以线性表示 w_i，并且变换矩阵 T 尽可能简化(例如，非零元素尽可能地少，而且最好为1或者−1)。

3 多精度整数的乘法

考虑实数域 R 上以 $\{u_iv_j,0\leqslant i,j\leqslant n-1\}$ 为基的线性空间 $G=\{x\mid x=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}a_{i,j}u_iv_j,a_{i,j}\in R\}$、以 $\{w_i,0\leqslant i\leqslant 2n-2\}$ 为基的线性空间 $G_0=\{x\mid x=\sum_{i=0}^{2n-2}a_iw_i,a_i\in R\}$ 和由 $\{w'_i,0\leqslant i\leqslant k-1\}$ 张成的线性空间 $G_1=\{x\mid x=\sum_{i=0}^{k-1}a_iw'_i,a_i\in R\}$。由于 $G_0\subseteq G_1\subseteq G$，所以，$k$ 至少为 $2n-1$。也就是说，用线性表示的方法来计算 n 位整数的乘积，至少需要 $2n-1$ 次一位整数的乘法。此时，$\{w'_i,0\leqslant i\leqslant 2n-2\}$ 之间线性无关，T 是满秩矩阵，$G_0=G_1$，T 是两组基 $\{w_i,0\leqslant i\leqslant 2n-2\}$ 和 $\{w'_i,0\leqslant i\leqslant 2n-2\}$ 之间的线性变换，所以每一个 w'_i 也可以通过 $\{w_i,0\leqslant i\leqslant 2n-2\}$ 线性表示出来。

定理1 $w=\sum_{i=0}^{2n-2}a_iw_i,a_i\in R$ 能表示成关于关于 $u_{n-1},u_{n-2},\cdots,u_1,u_0$ 和 $v_{n-1},v_{n-2},\cdots,v_1,v_0$ 的一次齐式的乘积，当且仅当以下两个条件之一成立：

(a) $w=a_{2n-2}w_{2n-2},a_{2n-2}\neq 0$；

(b) $a_0\neq 0$ 而且存在实数 x 使得对于所有 $0\leqslant i\leqslant 2n-3$ 的都有 $a_{i+1}=xa_i$。

证明：$n=1$ 时结论显然成立。当 $n\geqslant 2$ 时，如第2节所述，$w=\sum_{i=0}^{2n-2}a_iw_i,a_i\in R$ 是一个实对称双线性函数；由引理1，它能表示成关于关于 $u_{n-1},u_{n-2},\cdots,u_1,u_0$ 和 $v_{n-1},v_{n-2},\cdots,v_1,v_0$ 的一次齐式的乘积，当且仅当其系数矩阵 A 的秩为1。

$$A=\begin{pmatrix}a_0&a_1&a_2&\cdots&a_{n-1}\\a_1&a_2&a_3&\cdots&a_n\\a_2&a_3&a_4&\cdots&a_{n+1}\\\cdots&\cdots&\cdots&\cdots&\cdots\\a_{n-1}&a_n&a_{n+1}&\cdots&a_{2n-2}\end{pmatrix}$$

先证必要性。如果 A 的秩为1，A 的任意两个行向量之间一定是线性相关的，所以对于每个 $0\leqslant i\leqslant 2n-2$，都存在不全为0的实数 k_{i1},k_{i2}，使得 $k_{i1}(a_i,a_{i+1},\cdots,a_{i+n-1})+k_{i2}(a_{i+1},a_{i+2},\cdots,a_{i+n})=0$。

如果 k_{i1},k_{i2} 中有一个为0，不妨设 $k_{i1}=0,k_{i2}\neq 0$，那么，由于 $k_{i2}(a_{i+1},a_{i+2},\cdots,a_{i+n})=0$，所以 $a_{i+1},a_{i+2},\cdots,a_{i+n}$ 均为0。如果 $a_i\neq 0$，那么在矩阵 A 中第 $i+1$ 行的上方存在一个副对角线上元素全为 a_i 的 $i+1$ 阶三角子矩阵，它的秩为 $i+1$，所以 i 最多可能为0，也就是说第 $i+1$ 行的上方如果有非零元素，只可能是 a_0；同样的道理，第 $i+1$ 行的下方如果有非零元素，只可能是 a_{2n-2}。由于矩阵 A 的秩为1，$a_0\neq 0$ 和 $a_{2n-2}\neq 0$ 两种情况还不能同时出现。

如果对于所有 $0\leqslant i\leqslant 2n-2$，$k_{i1}$ 和 k_{i2} 都不等于0，令 $x_i=-k_{i1}/k_{i2}\neq 0$，那么，对于所有的 $1\leqslant j\leqslant n$，都有 $a_{i+j}=x_ia_{i+j-1}$，一旦 $a_{i+1},\cdots,a_{i+n}$ 中有一个元素 a_{i+j} 为0，那么由 $a_i=a_{i+1}/x_i,a_{i+1}=a_{i+2}/x_i,\cdots,a_{i+j-1}=a_{i+j}/x_i$ 可知 $a_i=a_{i+1}=\cdots=a_{i+j-1}=a_{i+j}=0$，由 $a_{i+n}=x_ia_{i+n-1},a_{i+n-1}=x_ia_{i+n-2},\cdots,a_{i+j+1}=x_ia_{i+j}$ 可知 $a_{i+n}=a_{i+n-1}=\cdots=a_{i+j+1}=a_{i+j}=0$，由前面的证明可知仅有 $a_0\neq 0$ 或者 $a_{2n-2}\neq 0$。而如果所有 $a_{i+1},\cdots,a_{i+n}$ 都不等于0，那么让 i 取遍0到 $n-2$，由递推关系 $a_{i+j}=x_ia_{i+j-1}$ 可以得到所有 $x_i(0\leqslant i\neq n-2)$ 都相等，不妨记为 x，所有 $a_i(0\leqslant i\leqslant 2n-2)$ 均不等于0，而且 $a_{i+1}=xa_i(0\leqslant i\leqslant 2n-3)$。

综合以上证明的情况，如果仅有 $a_{2n-2}\neq 0$，就是情况(a)，如果仅有 $a_0\neq 0$ 或者所有元素都不为0，就是情况(b)。这样必要性得证。

反之，如果系数满足(a)或者(b)的情况之一，矩阵 A 的秩一定为1，充分性得证。证毕

推论1 存在求n位整数的乘积，只需要 $2n-1$ 次一位整数乘法的线性表示算法。如果 T 如等式(2)中将 $\{w'_i,0\leqslant i$

$\leqslant 2n-2\}$变换到$\{w_i, 0\leqslant i\leqslant 2n-2\}$，那么 T^{-1}的行向量只可能取如下两种值：

(a)$l(0,0,\cdots,0,1), l\neq 0$；

(b)$l(1,x,x^2,\cdots,x^{2n-2}), l\neq 0$.

证明：由定理1，推论的后一部分是显然的。下面我们证明前一部分。考虑线性变换

$$\begin{pmatrix} 1 & a_0 & a_0^2 & \cdots & a_0^{2n-2} \\ 1 & a_1 & a_1^2 & \cdots & a_1^{2n-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & a_{2n-2} & a_{2n-2}^2 & \cdots & a_{2n-2}^{2n-2} \end{pmatrix}$$，其中 $a_i\neq a_j$，若 $i\neq j$。

它的行列式是 Vandermonde 行列式的转置，所以不等于0，将它的逆记为 T。由定理1，T 可以将 $2n-1$ 个一次齐式的乘积$\{w'_i, 0\leqslant i\leqslant 2n-2\}$线性变换成$\{w_i, 0\leqslant i\leqslant 2n-2\}$。这样的一个乘法算法只需要 $2n-1$ 次一位整数乘法。

由推论1，我们有下面的结论：

推论 2[4] 对于任意给定的 $\varepsilon>0$，存在独立于 n 的常数 $C(\varepsilon)$，使得两个长度为 n 的无符号整数相乘所需的基本操作 $t(n)<C(\varepsilon)n^{1+\varepsilon}$。

这一结论的证明可以参考文[4]对于 Toom-Cook 算法的复杂性证明。

从推论1，我们就可以构造所有的 Toom-Cook 算法。为了运算简单起见，我们通常将推论1中的 l 取为1或者-1，并且总是将第(a)类向量加入到中 T^{-1}中。为了讨论方便，以后我们将向量$(1,x,x^2,\cdots,x^{2n-2})$记为 $g(x,n)$。

首先考虑 $n=2$ 的情况。取 $g(0,2)$，$-g(-1,2)$和第(a)类向量，我们得到

$$T^{-1}=\begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$，此时，

$$\begin{pmatrix} w'_0 \\ w'_1 \\ w'_2 \end{pmatrix}=T^{-1}\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}=\begin{pmatrix} u_0v_0 \\ (u_1-u_0)(v_0-v_1) \\ u_1v_1 \end{pmatrix}$$

由计算可得

$$T=\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}=T\begin{pmatrix} w'_0 \\ w'_i \\ w'_2 \end{pmatrix}=\begin{pmatrix} w'_0 \\ w'_1+w'_2+w'_0 \\ w'_2 \end{pmatrix}$$

这个算法就是我们熟知的分治算法，共用了3次1位数的乘法，利用递归的方法可以得到一个复杂度为 $O(n\log_2^3)$的乘法算法。

再考虑 $n=3$ 的情况。取 $g(0,3)$，$g(1,3)$，$g(-1,3)$，$g(2,3)$和第(a)类向量，我们得到

$$T^{-1}=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 2 & 4 & 8 & 16 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

此时，

$$\begin{pmatrix} w'_0 \\ w'_1 \\ w'_2 \\ w'_3 \\ w'_4 \end{pmatrix}=T^{-1}\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}=\begin{pmatrix} u_0v_0 \\ (u_0+u_1+u_2)(v_0+v_1+v_2) \\ (u_0-u_1+u_2)(v_0-v_1+v_2) \\ (u_0+2u_1+4u_2)(v_0+2v_1+4v_2) \\ u_2v_2 \end{pmatrix}$$

由计算可得

$$T=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1/2 & 1 & -1/3 & -1/6 & 2 \\ -1 & 1/2 & 1/2 & 0 & -1 \\ 1/2 & -1/2 & -1/6 & 1/6 & -2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}=\begin{pmatrix} w'_0 \\ w'_1+2w'_4-\frac{1}{6}(3w'_0+2w'_2+w'_3) \\ \frac{1}{2}(w'_1+w'_2)-w'_0-w'_4 \\ \frac{1}{6}(3w'_0-3w'_1-w'_2+w'_3)-2w'_4 \\ w'_4 \end{pmatrix} \quad (3)$$

这个算法在 GNU GMP[8] 多精度运算库中由 Zimmerman 实现，是目前最好的 $n=3$ 的一个 Toom-Cook 实例。

在实际应用中，分治算法($n=2$)是用得最多的乘法算法，主要原因是实际应用中一般都只涉及几千比特的整数相乘(例如 RSA 目前到1024比特或者2048比特就足够安全了)。当 n 增大时，虽然减少了乘法运算的次数，但是增加了加、减法和一些小系数的处理。这些增加的运算如果过于复杂，那么对于几千比特的整数而言就会削弱因乘法运算减少而带来的加速效果。分治算法的变换矩阵 T 及其逆矩 T^{-1}阵中的元素都为0，1或者-1，处理起来非常方便。相比较而言，当 $n=3$ 时，虽然 T^{-1}中的元素 2^i 在计算机中很容易用移位来实现，但是 T 中出现了除以3的运算。文[9]和[10]研究了如何快速除以小常数的算法。从研究结果来看，除以小常数处理起来还是需要较多的操作。因此，尽管 $n=3$ 可以实现复杂度为 $O(n\log_3^5)$的乘法算法，但处理几千比特的整数时，人们仍然宁可选择分治算法。

处理几千比特的整数相乘除了采用分治算法之外，还有一种方法就是选择不可逆的矩阵 T，这样可以简化变换矩阵。

当 $n=3$ 时，考虑

$$\begin{pmatrix} w'_0 \\ w'_1 \\ w'_2 \\ w'_3 \\ w'_4 \\ w'_5 \end{pmatrix}=\begin{pmatrix} u_0v_0 \\ u_1v_1 \\ u_2v_2 \\ (u_0+u_1)(v_0+v_1) \\ (u_1+u_2)(v_1+v_2) \\ (u_0+u_2)(v_0+v_2) \end{pmatrix}$$，则

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 1 \\ 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} w'_0 \\ w'_1 \\ w'_2 \\ w'_3 \\ w'_4 \\ w'_5 \end{pmatrix} \quad (4)$$

它在乘法上比(3)式多用了一个，但是加、减法和系数的处理却大为简化。

4 多精度整数的平方

我们将等式(1)重写为

$$U^2=\sum_{i=0}^{2n-2}w_iB^i$$，其中，$w_i=\sum_{s+t=i}u_su_t \quad (5)$

它的系数 w_i 可以看成是关于变量$(u_{n-1},u_{n-2},\cdots,u_1,u_0)$的一个 n 元实二次型。同样，$w=\sum_{i=0}^{2n-2}a_iw_i$，$a_i\in R$ 也是一个 n 元实二次型，根据引理1，w 可以表示成两个一次齐式的乘积的

充要条件是它的秩为1，或者秩为2且正惯性指数为1。由于w的系数矩阵和乘法的情形是一样的，因此定理1的充分性条件对于平方也是成立的。而且推论2对于平方也是成立的。

和普通乘法比较起来，除了推论1中的两类向量可以作为T^{-1}的行向量，还有更多的更简单的向量可以使$w=\sum_{i=0}^{2n-2}a_iw_i, a_i\in R$表示成一次齐式的乘积。我们限制$a_i\in\{-1,0,1\}$，利用计算机搜索到一些易于计算的形式。

当$n=2$时，$w_0=u_0^2, w_1=2u_0u_1, w_2=u_1^2$，本身都是一次齐式的乘积，可以直接计算。

当$n=3,4,5,6$时，我们采用如下的搜索规则搜索向量$(a_0,a_1,\cdots,a_{2n-2})$，使得$w=\sum_{i=0}^{2n-2}a_iw_i$可以表示成一次齐式的乘积。

搜索规则1：

(a)$a_i\in\{-1,0,1\}$；

(b)w的系数矩阵A的秩为1或者秩为2且正惯性指数为1；

(c)如果$(a_0,a_1,\cdots,a_{2n-2})$满足条件(a)和(b)，那么它的负向量$-(a_0,a_1,\cdots,a_{2n-2})$也满足条件(1)和(2)，我们不再将负向量记入搜索结果。

当$n=3$时，按照搜索规则1，共搜索到32个向量，见表1。

表1　$n=3$时搜索的32各向量

(0,1,1,0,−1)	(0,1,1,1,1)	(0,1,0,−1,0)	(0,1,0,0,0)	(0,1,0,1,0)	(0,1,−1,0,1)	(0,1,−1,1,−1)	(0,0,0, 1,1)
(0,0,0,1,0)	(0,0,0,−1,1)	(0,0,0,0,1)	(1,1,1,1,−1)	(1,1,1,1,0)	(1,1,1,1,1)	(1,1,0,−1,−1)	(1,1,0,0,0)
(1,1,0,1,−1)	(1,1,−1,−1,1)	(1,1,−1,1,−1)	(1,0,0,0,−1)	(1,0,0,0,0)	(1,0,−1,−1,0)	(1,0,−1,0,1)	(1,0,−1,1,0)
(1,−1,1,−1,−1)	(1,−1,1,−1,0)	(1,−1,1,−1,1)	(1,−1,0,−1,−1)	(1,−1,0,0,0)	(1,−1,0,1,−1)	(1,−1,−1,−1,−1)	(1,−1,−1,1,1)

当$n=4$时，按照搜索规则1，共搜索到30个向量，见表2。

表2　$n=4$时搜索的30个向量

(0,1,1,0,−1,−1,0)	(0,1,1,1,1,1,1)	(0,1,0,−1,0,1,0)	(0,1,0,0,0,0,0)	(0,1,0,1,0,1,0)	(0,1,−1,0,1,−1,0)
(0,0,0,0,0,1,1)	(0,0,0,0,0,1,0)	(0,0,0,0,0,−1,1)	(0,0,0,0,0,0,1)	(0,1,−1,1,−1,1,−1)	(1,1,1,1,1,1,−1)
(1,1,1,1,1,1,0)	(1,1,1,1,1,1,1)	(1,1,0,−1,−1,0,1)	(1,1,0,0,0,0,0)	(1,1,−1,−1,1,1,−1)	(1,1,−1,1,−1,1,−1)
(1,0,0,0,0,0,−1)	(1,0,0,0,0,0,0)	(1,0,−1,−1,0,1,1)	(1,0,−1,0,1,0,−1)	(1,0,−1,1,0,−1,1)	(1,−1,1,−1,1,−1,−1)
(1,−1,1,−1,1,−1,0)	(1,−1,1,−1,1,−1,1)	(1,−1,0,0,0,0,0)	(1,−1,0,1,−1,0,1)	(1,−1,−1,−1,−1,−1,−1)	(1,−1,−1,1,1,−1,−1)

当$n=5$时，按照搜索规则1，共搜索到30个向量，见表3。

表3　$n=5$时搜索的30个向量

(0,1,1,0,−1,−1,0,1,1)	(0,1,1,1,1,1,1,1,1)	(0,1,0,−1,0,1,0,−1,0)	(0,1,0,0,0,0,0,0,0)	(0,1,0,1,0,1,0,1,0)
(0,1,−1,0,1,−1,0,1,−1)	(0,0,0,0,0,0,0,1,1)	(0,0,0,0,0,0,0,1,0)	(0,0,0,0,0,0,0,−1,1)	(0,0,0,0,0,0,0,0,1)
(0,1,−1,1,−1,1,−1,1,−1)	(1,1,1,1,1,1,1,1,−1)	(1,1,1,1,1,1,1,1,0)	(1,1,1,1,1,1,1,1,1)	(1,1,0,−1,−1,0,1,1,0)
(1,1,0,0,0,0,0,0,0)	(1,1,−1,−1,1,1,−1,−1,1)	(1,1,−1,1,−1,1,−1,1,−1)	(1,0,0,0,0,0,0,0,−1)	(1,0,0,0,0,0,0,0,0)
(1,0,−1,−1,0,1,1,0,−1)	(1,0,−1,0,1,0,−1,0,1)	(1,0,−1,1,0,−1,1,0,−1)	(1,−1,1,−1,1,−1,1,−1,−1)	(1,−1,1,−1,1,−1,1,−1,0)
(1,−1,1,−1,1,−1,1,−1,1)	(1,−1,0,0,0,0,0,0,0)	(1,−1,0,1,−1,0,1,−1,0)	(1,−1,−1,−1,−1,−1,−1,−1,−1)	(1,−1,−1,1,1,−1,−1,1,1)

我们从$n=4$和$n=5$的情况可以看出明显的规律性：搜索的结果集中都有30个向量，而且向量的模式都是相同的。更一般地，我们有下面的结论：

命题1　当$n\geqslant 4$时，按照搜索规则1，搜索的结果集中共有30个向量，每个向量的分量，或者除了首尾的1到2个固定分量以外的其它分量，都按照周期1,2,3,4或者6重复出现，固定分量与周期性出现的分量的搭配是固定不变的。

例如：(0,1,1,0,−1,−1,0,1,1)的分量出现的周期为6,(0,0,0,0,0,0,0,−1,−1)除了最后两个分量外的其它分量出现的周期为1。

命题1的证明需要较为繁杂的计算来验证，限于篇幅，我们在此仅对其作一说明。首先，$n=4$和$n=5$的情况是对命题1的一个验证，当$n>5$时，任何向量如果满足搜索规则1，根据搜索中关于秩和正惯性指数的条件，它的前9个分量必须是$n=5$的30种情况中的一种，或者是这30个向量的负向量中的一种，或者全部为0。以向量(0,1,1,1,1,1,1,1,1)为例，当它演变到$n=6$中的向量的时候，可能有9种情况，分别是(0,1,1,1,1,1,1,1,1,0,0),(0,1,1,1,1,1,1,1,1,0,1),…,(0,1,1,1,1,1,1,1,1,−1,−1)等，所有这些情况中只有(0,1,1,1,1,1,1,1,1,1,1)满足搜索规则1的第二个条件。另一个例子是向量(0,0,0,0,0,0,0,−1,−1)，它不能直接通过增加两个元素来演变到$n=6$的情况，(0,0,0,0,0,0,0,0,0,−1,−1)是通过在(0,0,0,0,0,0,0,0,0)后面添加两个−1得到的。

定理2　命题1中的30个向量，当$n=4,5,6$和$n\geqslant 7$时的最大无关组中的向量个数分别为7,9,11和12。

证明：我们选取如下12个模式的向量。

周期为1的：(0,1,1,1,…,1),(0,1,0,0,…,0),(1,0,0,0,…,0),(0,0,0,…,0, 1, 1),(0,0,0,…,0,1,0)；

周期为2的：(0,1,0,1,…,0,1,0)；

周期为3的：(0,1,−1,0,1,−1,…),(1,0,−1,1,0,−1…)；

周期为4的：(0,1,0,−1,0,1,0,…),(1,0,−1,0,1,0,−1,…)；

周期为6的：(0,1,1,0,−1,−1,0,…),(1,0,−1,−1,0,1,1,…)。

容易验证所有其它的18个向量可以通过这12个向量线性表示出来。

通过计算，当$n=4,5,6,7$时，以这12个向量为行向量的矩阵的秩分别为7,9,11和12。所以当$n=4,5,6$和7时，30

个向量的最大无关组中的向量个数分别为 7,9,11 和 12。当 $n>7$ 时,由于以这 12 个向量为行向量的矩阵中的前 13 列所形成的矩阵在 $n=7$ 时被验证秩为 12,所以整个矩阵的秩也为 12。由此 $n>7$ 时,30 个向量的最大无关组中的向量个数为 12。证毕。

从搜索结果集中选择 $2n-1$ 个不同的向量,以它们为行向量,形成一个矩阵,这个矩阵就是等式 2 中变换矩阵 T 的逆。

为了使变换矩阵 T 的计算简单,所选矩阵一般满足下面的条件:

(a)矩阵可逆;

(b)判别式为 ±1 或者 ±2;

(c) T 的非零元很少。

根据定理 2,当 $n\geqslant7$ 时,30 个向量的最大无关组中的向量个数为 12,也就是说,无论如何选择,形成的变换矩阵的秩都不会超过 12,而此时的变换矩阵至少有 13 行,所以一定是不可逆矩阵。我们的讨论仅限于 $n=3,4,5,6$ 的情况,这对于一般的应用已经足够了。

当 $n=3$ 的时候,除了可以得到文[11]的两个解外,我们还可以得到如下一个行列式为 1 的矩阵:

$$T_3^{-1}=\begin{pmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&0&1&0\\0&0&0&0&1\\0&1&1&0&-1\end{pmatrix},$$

$$T_3=\begin{pmatrix}1&0&0&0&0\\0&1&0&0&0\\0&-1&0&1&\\0&0&1&0&0\\0&0&0&1&0\end{pmatrix},\begin{pmatrix}w'_0\\w'_1\\w'_2\\w'_3\\w'4\end{pmatrix}=T_3^{-1}\begin{pmatrix}w_0\\w_1\\w_2\\w_3\\w_4\end{pmatrix}$$

$$=\begin{pmatrix}u_0^2\\2u_0u_1\\2u_1u_2\\u_2^2\\(u_1+u_2)(2u_0+u_1-u_2)\end{pmatrix}$$

当 $n=4$ 的时候,搜索到的一个行列式为 1 的矩阵为:

$$T_4^{-1}=\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&1&1\\0&1&0&-1&0&1&0\\1&1&-1&-1&1&1&-1\\1&-1&0&1&-1&0&1\end{pmatrix},$$

$$T_4=\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\2&0&1&0&0&-1&-1\\0&1&1&0&-1&0&0\\1&0&0&1&-1&0&-1\\0&0&1&0&0&0&0\\0&0&-1&1&0&0&0\end{pmatrix}$$

当 $n=5$ 的时候,搜索到的一个行列式为 2 的矩阵为:

$$T_5^{-1}=\begin{pmatrix}1&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&1\\0&1&0&-1&0&1&0&-1&0\\1&1&0&-1&-1&0&1&1&0\\0&1&-1&0&1&-1&0&1&-1\\1&-1&0&1&-1&0&1&-1&0\\1&-1&-1&1&1&1&-1&-1&1\end{pmatrix},$$

$$T_5=\begin{pmatrix}1&0&0&0&0&0&0&0&\\0&1&0&0&0&0&0&0&0\\2&0&0&1&-1&-1/2&0&-1/2&-1\\2&0&0&1&-1&-1/2&0&1/2&0\\0&1&1&0&0&-1/2&0&1/2&0\\2&-1&1&2&0&-1&1&0&-1\\0&0&2&0&1&-1/2&0&1/2&0\\1&-1&1&2&0&-1/2&0&1/2&0\\1&-1&1&2&0&-1/2&1&1/2&-1\\0&0&1&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0\end{pmatrix}$$

当 $n=6$ 的时候,所搜索到的矩阵的行列式的绝对值最少的为 24。因此,T_6 中除了加减法和移位操作外,一定还包含有被 3 除的操作,在通常几千比特的乘法运算中,已经不能显出优势了。

5 算法对比分析

利用实对称双线性函数,我们很容易地给出了分治算法和 Zimmerman 算法,这两个算法是目前应用比较普遍的 Toom-Cook 快速乘法算法。对于 $n=3$ 的情况,(4)式给出了 3 位整数相乘需要 6 次一位整数乘法的快速算法,采用递归的方法可以得到一个复杂度 $O(n\log_3^6)$ 的快速算法。和 Zimmerman 算法比较起来,它的变换矩阵中的元素都是 -1,0 或者 1,大大简化了系数的处理,对于几千比特的整数运算具有一定的优势。

利用二次型的方法,我们搜索到了 $n=3,4,5$ 时的快速算法,T_3 和文[11]中平方的效率相当,均需要 5 次乘法和少量的移位及加减法。但文[11]中给出的 $n=4$ 时的变换矩阵的行列式的值为 1/2,$n=5$ 时的行列式的值为 1/8,本文所搜索到的变换矩阵 T_4 和 T_5 的行列式的值分别为 1 和 1/2,而且矩阵中的非零元素比文[11]少,说明平均运算效率优于文[11]。

结论 本文基于实对称双线性函数和实二次型研究了多精度整数的快速乘法和平方理论。定理 1 给出了 Toom-Cook 算法的所有表现形式,推论 1 进一步明确了寻找 Toom-Cook 算法的基本方法。在第 3 节,以 n=3 为例,我们还给出了 Toom-Cook 算法的一个推广。

对于平方而言,本文给出了 Toom-Cook 平方算法中矩阵的约束条件,即秩为 1,或者秩为 2 且正惯性指数为 1。在进一步限制变换矩阵中的元素为 -1,0 和 1 之后,我们给出了所有的这些矩阵的行向量以及最大无关组向量的个数,从而证明了当 n≥7 时,限制条件下的快速平方算法是不存在的。

本文的研究方法和结果对提高数论计算、公钥密码计算和高精度数值计算的运算效率具有一定的参考价值。

致谢 作者所在的信息安全课题组的老师和博士研究生组织讨论班对多精度整数的计算理论与密码学应用进行了深

入的讨论，并在本文的写作过程中提出了许多宝贵的意见和建议，在此一并表示感谢。

参 考 文 献

1 Montgomery PL. Modular Multiplication Without Trial Division. Mathematics of Computation, 1985, 44(170): 519～521
2 Acar T, BS Kaliski Jr, C Ko C. Analyzing and Comparing Montgomery Multiplication Algorithms. IEEE Micro, 1995, 16(3): 26～33
3 Karatsuba A, Ofman Y. Multiplication of multidigit numbers on automata. Soviet Physics Doklady (English translation), 1963, 7(7): 595～596
4 Knuth DE. The Art of Comuter Programming. Vol. 2, Seminumerical Algorithms. 2nd edition. Addison-Wesley, 1981
5 Montgomery PL. Five, six, and seven-term Karatsuba-like formulae. IEEE Transaction on Computers, 2005, 54(3): 362～369
6 Schönhage A, Strassen V. Schnelle mutiplikation grosser zahlen. Computing, 1971, 7: 281～292
7 熊全淹. 线性代数. 第三版. 高等教育出版社, 1985. 164～172
8 GNU. GNU multiple precision arithmetic library. http://www.swox.com/gmp. 2005
9 Jebelean T. An algorithm for exact division. Journal of Symbolic Computation, 1993, 15: 169～180
10 Krandick W, Jebelean T. Bidirectional exact integer division. Journal of Symbolic Computation, 1996, 21: 441～455
11 Chung J, Hasan MA. http://www.cacr.math.uwaterloo.ca/techreports/2006/cacr2006-24.pdf, 2006

（上接第 43 页）

的更新，在进行关键字查找时必然影响查询的正确性并降低查找效率，因此 *successor* 对保证查询的正确性起着非常重要的作用。环中节点定期触发算法 stabilize（如图 5 所示）实现 *successor* 的及时更新，更改节点的 *successor* 的指针，更新 *finger* 表，既保证了查找的正确，又保证了查找的快速。

5 性能分析

我们在 CPU 为 AMD700MHz、内存为 256M 的机器上使用 SFS Chord 模拟器[13]进行了模拟测试。SFS Chord 模拟器的结构由两部分组成：协议实现部分（sim）和事件产生部分（traffic_gen）。协议实现部分用来实现各种 P2P 协议；事件产生部分由事件产生器产生各种事件，包括节点加入，文档插入，文档查找等事件。模拟实验中网络节点间的延时采用了 P2PSim[14]项目中测量的大约 2000 个网络 DNS 之间的 RTTs 矩阵，通过统计事件执行结果获取实验数据。

模拟实验主要验证 DeChord 相对于 Chord 所带来的查找效率的改进。在模拟过程中，提交相同数量的节点加入事件和文档查询事件，统计两种算法下每个文档的平均查询时间。这里文档的查询时间等于查询文档事件中路由路径所经过的节点之间的延时之和。

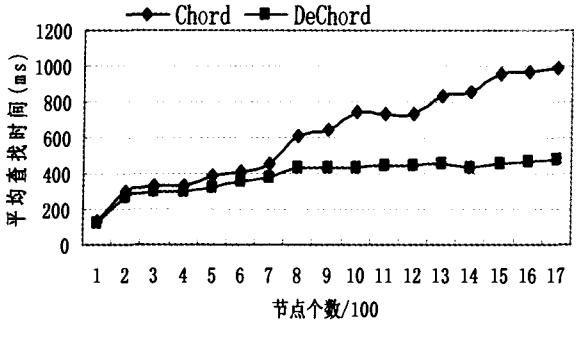

图 6 DeChord 与 Chord 的平均查找时间分布图

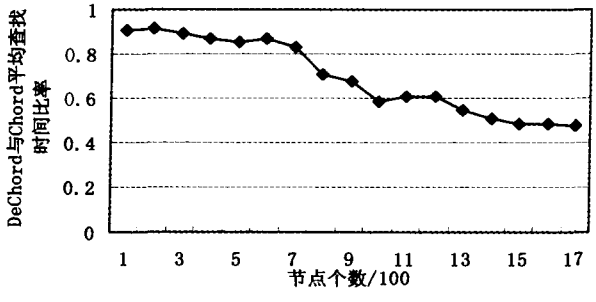

图 7 DeChord 与 Chord 的平均查找时间比率图

图 6 给出了 DeChord 与 Chord 在不同规模的环境下平均查找时间的分布情况，其中横坐标是节点个数，纵坐标是查找时间（ms）。从模拟试验结果可以看出来，网络节点数越多，DeChord 相对于 Chord 带来的改进就越明显。这是因为系统中加入的节点数越多，节点的 *finger* 表更新的次数就越多。由于每次更新都是淘汰 *finger* 表中距离较远的节点，导致 *fniger* 表中表项的后继节点中近距离节点占节点总数的比例越高，因此查找过程中节点都是选择距离较近的节点进行路由，减少了路由过程中的网络延时。

图 7 给出了 DeChord 与 Chord 的平均查找时间的比率图，其中横坐标代表网络节点个数，纵坐标代表 DeChord 与 Chord 的平均查找时间比率。从图中可以看出，比率曲线成一个逐渐下降的趋势，这也说明了与 Chord 相比，网络节点数越多，DeChord 改进的效果越明显，这也符合我们的算法设计目标。

结论 本文提出一种改进的 Chord 构建算法 DeChord，DeChord 考虑到节点之间的物理距离，系统节点总是选择距离自己物理距离较近的节点作为邻居节点，这种邻居节点的选择方式可以降低消息路由过程中每一跳的网络延时，从而降低了路由的总延时。模拟实验结果表明，与 Chord 相比，DeChord 算法能大幅度缩短数据定位延时，且网络节点越多，效果越明显。下一步的工作将针对节点失效处理进行进一步的研究，提出更优化的失效处理策略并进行模拟。

参 考 文 献

1 Napster. http://www.napster.com
2 Gnutella. http://gnutella.wego.com
3 FreeNet. http://freenet.sourceforge.net
4 Stoica I, Morris R, Karger D, Kaashoek M F, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of SIGCOMM 2001, Aug. 2001
5 Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A Scalable Content-Addressable Network. In: Proceedings of SIGCOMM 2001, Aug. 2001
6 Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. Available at: http://research.microsoft.com/antr/PAST/, 2001
7 FIPS 180-1. Secure Hash Standard. U. S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, Apr. 1995
8 Dabek F, Kaashoek M F, Karger D, Morris R, Stoica I. Wide-area cooperative storage with CFS. SOSP'01, Banff, Canada, October 2001
9 Castro M, Druschel P, Hu Y C. Topology-aware routing in structured peer-to-peer overlay networks, Microsoft research technical report msr-tr-2002-82
10 Szymaniak M, Pierre G, van Steen M. Scalable Cooperative Latency Estimation
11 Ng T S E, Zhang Hui. A Network Positioning System for the Internet
12 Castro M, Druschel P, Hu Y C. Topology-aware routing in structured peer-to-peer overlay networks, Microsoft research technical report msr-tr-2002-82
13 SFS Chord simulator. http://pdos.lcs.mit.edu/cgi-bin/cvsweb.cgi/sfsnet/simulator
14 The p2psim project. http://www.pdos.lcs.mit.edu/p2p sim