多数桁数因数分解における 0-1 行列の疎行列直接解法

渡邊 裕介、 後 保範 （早稲田大学）

Direct Sparse Solver with 0-1 Matrix for Multiple Precision Integer Factorizations

Yusuke Watanabe, Yasunori Ushiro (Waseda University)

1. はじめに

RSA 暗号は多数桁の因数分解の困難性を利用している。現在、1024 ビットの RSA 暗号の解読はスーパーコンで数千年かかると言われている。RSA 暗号に使用される因数分解には篩系の解法が使用される。篩系の解法として MPQS(複数多項式 2 次篩法)、GNFS(一般数体篩法)があり、更に多重基底多項式篩法(MBPS)を提案している。いずれの方法でも篩に使用した基底の 2 乗化が必要である。このため、篩で得たデータを掛け合せて素数、一次式及び素イデアルのベキを mod 2 の基で一次従属にするデータの組み合わせを抽出する必要がある。これは、篩法で得られたデータに対して、0-1 を係数とする行列の一次従属行を求めることに帰着する。その方法として、ブロック･ランチョス法が用いられているが、我々は疎行列直接解法を使用する方法を検討している。

2. 密行列のガウス消去法

0-1 を係数とする N×M 次元(N>M)の行列を A とし、N×N 次元の単位行列を E と表示する。A と E を合わせた N×(M+N)次元の行列を G とすると、行列 G の A に対応する部分のガウス消去を mod 2 の基で、G 全体で行い、A の要素が総てゼロになる行に対応する E の部分に一次従属関係ができる。

これを利用して、行列 G の要素を $G_{i,j}$ とすると消去計算を下記で行う。必ず 1 になるように行交換して得られたピボット $G_{s,t}$ を使用した消去計算は下記のようになる。

$$G_{i,j} = G_{i,j} - G_{i,t} \cdot G_{s,j} \pmod 2, \qquad (i=s+1,\cdots,N,\quad j=t+1,\cdots,N+M)$$

これは行列 A を(mod 2)の基で LU 分解し、$L^{-1}E$ (mod 2)を計算したものに帰着する。

3. ガウス消去法の工夫

しかし、行列 G の作成には、大きなメモリ量が必要になる。このため、行列 A だけを使用して計算できるように工夫する。これには、行列 A を(mod 2)の基で LU 分解し、$L^{-1}E$ (mod2)を計算すれば良いことに着目して行う。このとき重要なのは、上三角行列 U は作成する必要がないことである。行列 A の要素を $A_{i,j}$ とし、必ず 1 になるように行交換して得られた消去ピボット $A_{s,t}$ を使用した消去計算は下記のようになる。

$$A_{i,j} = A_{i,j} - A_{i,t} \cdot A_{s,j} \pmod 2, \qquad (i=s+1,\cdots,N,\quad j=1,\cdots,t-1)$$

$$A_{i,j} = A_{i,j} - A_{i,t} \cdot A_{s,j} \pmod 2, \qquad (i=s+1,\cdots,N,\quad j=t+1,\cdots,M)$$

$$A_{i,t} = A_{i,t} \qquad (i=s+1,\cdots,N)$$

この場合は、交換した行番号の情報が必要なため、長さ M と N のベクトルを用いる。

7×5 次元の行列での具体的な例を示す。表 1 は入力行列 A と、表 2 は消去後の行列 A と、それぞれ行交換の情報を蓄えた 2 つのベクトル(行番号と消去行)である。行番号のベクトルは行番号を示すもので、最初は 1,2,3, …,N の順に入れておき、行交換に合わせて入れ替える。消去行はゼロクリアしておき、消去ピボット(表 2 の太字の **1**)となった行番号を入れる。

表 1. 入力行列 A

行番号	入力				
1	1	0	1	0	0
2	1	1	1	0	1
3	1	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	1
6	1	0	0	1	1
7	1	1	1	0	0
消去行	0	0	0	0	0

表 2. 消去後の行列 A

	行番号	消去後				
	1	**1**	0	1	0	0
	2	1	**1**	0	0	1
	4	0	0	**1**	1	1
	3	1	0	0	0	**1**
従属行 →	5	1	1	0	0	0
従属行 →	6	1	0	1	0	0
従属行 →	7	1	1	0	0	1
	消去行	1	2	4	0	3

消去は 4 行目で終わり、行番号が 5,6,7 の行は一次従属を示す。一次従属は 1 が立っている列に対応する消去行と、行番号そのもので成り立つ。そのため、この場合の一次従属は 3 組あり、それぞれ(1,2,5),(1,4,6),(1,2,3,7)が得られる。

4. ブロック化によるガウス消去法

疎行列に対して効率の良い解法にするため、ブロック化してガウス消去法を使用する。連立一次方程式の場合のブロックガウス消去法(LU 分解)と異なり、ブロック化の有無に関わらず、行交換は行全体で行う必要がある。このため、消去の最終結果はブロック無しの場合と同一になる。ここでは、ブロック化のガウス消去法の計算手順と具体例を示す。

[計算手順]

0-1 行列のブロック化によるガウス消去法の計算手順を図1に示す。

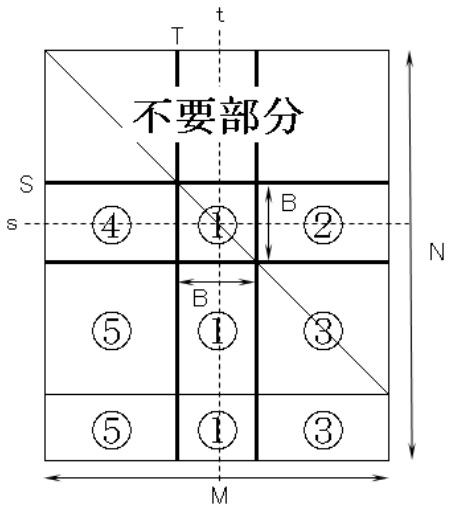

図 1. ブロックガウスの消去手順

連立一次方程式の場合

1. ①の部分を消去する
2. ②の部分を消去する
3. ③の部分を消去する

0-1 行列の場合

1. ①の部分を消去する
2. ③の部分を消去する
3. ⑤の部分を消去する

このため、連立一次方程式の場合と異なるのは

(1) ⑤を消去する必要がある

(2) ②を消去する必要がない

の 2 点である。

そこで、具体的にブロック化サイズ B でブロック化することを考える。図 1 に示す $A_{S,T}$ のブロック内の消去ピボット $A_{s,t}$ を使用した消去計算は、以下の 2 ステップで行われる。

– Step 1 –

(s=S,…,S+B-1, t=T,…,T+B-1)

$A_{i,j} = A_{i,j} - A_{i,t} \cdot A_{s,j} \pmod 2$, (i=s+1,…,N, j=T,…,t-1)

$A_{i,j} = A_{i,j} - A_{i,t} \cdot A_{s,j} \pmod 2$, (i=s+1,…,N, j=t+1,…, T+B-1)

$A_{i,t} = A_{i,t}$ (i=s+1,…,N)

– Step 2 –

(s=S,…,S+B-1, t=T, ,…,T+B-1)

$A_{i,j} = A_{i,j} - A_{i,t} \cdot A_{s,j} \pmod 2$, (i=S+B,…,N, j=1,…,T-1)

$A_{i,j} = A_{i,j} - A_{i,t} \cdot A_{s,j} \pmod 2$, (i=S+B,…,N, j=T+B,…,M)

$A_{i,t} = A_{i,t}$ (i=S+B,…,N)

[具体例]

表 1 と同じく表 3 の入力行列を利用して、ブロック消去する場合を示す。表 5 はブロックサイズ 2 でブロック化したときの消去結果である。ここで、表 5 は表 2 と、図 1 の②および④の部分を除いて等しい行列となる。なお、表 4 の消去中の行列は表 3 を最初のブロック(S=T=1)で消去した途中経過の行列を示す。

表 3. 入力行列 A

行番号	入力				
1	1	0	1	0	0
2	1	1	1	0	1
3	1	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	1
6	1	0	0	1	1
7	1	1	1	0	0
消去行	0	0	0	0	0

表 4. 消去中の行列 A

行番号	途中経過				
1	1	0	1	0	0
2	1	1	1	0	1
3	1	0	0	0	1
4	0	0	1	1	1
5	1	1	0	0	0
6	1	0	1	1	1
7	0	1	0	0	1
消去行	1	2	0	0	0

表 5. 消去後の行列 A

行番号	消去後				
1	1	0	1	0	0
2	1	1	1	0	1
4	1	0	1	1	1
3	1	0	0	0	1
5	1	1	0	0	0
6	1	0	1	0	0
7	1	1	0	0	1
消去行	1	2	4	0	3

表 5 から、表 2 の時と同様に、一次従属は 3 組で(1,2,5),(1,4,6),(1,2,3,7)が得られる。

5. 疎行列ガウス消去法

(1) 疎行列の持ち方

篩で発生する 0-1 疎行列の持ち方の詳細は現在検討中である。この行列は非常に疎で、値は 32 ビット整数に 32 要素、64 ビット整数に 64 要素詰めて、排他和演算で一括処理できる特徴をもつ。疎行列のデータ圧縮の方法として、2 次元配列にして記憶する方法と、1 次元配列に記憶する方法が考えられる。2 次元配列に持つと、非ゼロ要素数も含めて更新が容易であるが、配列の溢れ対策が重要になる。一方、一次元配列では効率の良い詰め方ができるが、更新の頻度が高いと処理速度を低下させると思われる。疎行列ガウス消去法では、更新

頻度が高いので、2 次元配列の方法を採用することにした。

表 1 の行列 A を例に、現在検討中の疎行列の具体的な持ち方を表 6 に示す。この方法では、非ゼロ列番号を示す数字にゼロが現れると、その行の非ゼロ要素は終わりを示す。

(2) 確保した配列が溢れた場合の対策

非ゼロ列番号の数字のゼロが、その行の非ゼロ要素の終わりを示すので、その自然な拡張として行の最後の非ゼロ列番号がゼロ以外なら、この行は溢れが発生していることを示す。また同時に、どこに続けて記憶するかの情報も示す。表 7 は、表 6 のサイズを小さくしたために、溢れが発生した例を示す。

表 6. 疎行列の持ち方

行番号	非ゼロ列番号				
1	1	3	0		
2	1	2	3	5	0
3	1	3	5	0	
4	3	4	5	0	
5	2	5	0		
6	1	4	5	0	
7	1	2	3	0	

表 7. 配列の溢れ

行番号	非ゼロ列番号			
1	1	3	0	
2	1	2	3	**8**
3	1	3	5	0
4	3	4	5	0
5	2	5	0	
6	1	4	5	0
7	1	2	3	0
8	5	0		
9	0			

ここで、表 7 の太字の数字 **8** はその行にまだ非ゼロ列があることを表しており、この場合、2 行目の **8** はデータが溢れ出たので、その続きが 8 行目にあるということを表している。

6. おわりに

GNFS やMPQS等の篩法を使用した多数桁数の因数分解で発生する、大次元で疎な 0-1 行列の従属行を求める手法として、疎行列直接解法を検討している。まだ、検討を開始して間もないため、計算方法を確立した段階である。今後、疎行列の直接解法のメモリ量と性能を左右する番号変換処理等を組み込み、数値実験を行うつもりである。

謝辞

高速化方法についての貴重なコメントを頂き、研究打ち合わせの場所をご提供頂いた、東京大学金田康正教授に謹んで感謝の意を表します。

参考文献

1) 木田祐司: Number Field Sieve(数体ふるい法)について、木田祐司 HP、
http://www.rkmath.rikkyo.ac.jp/~kida/bunkai.htm (1999).

2) 後 保範: 行列計算、後 保範 HP、
http://www.aoni.waseda.jp/ushiro/info_matrix.htm (2003).

3) 横森 貴: アルゴリズムデータ構造計算論、サイエンス社 (2005).