Available online at www.sciencedirect.com

ScienceDirect

Applied Mathematics and Computation 190 (2007) 1848–1854

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

Computational efficiency analysis of Wu et al.'s fast modular multi-exponentiation algorithm

Da-Zhi Sun [a,*], Jin-Peng Huai [b], Ji-Zhou Sun [a], Jia-Wan Zhang [a]

[a] *School of Computer Science and Technology, Tianjin University, No. 92 Weijin Road, Nankai District, Tianjin 300072, PR China*
[b] *School of Computer, Beihang University, Beijing 100083, PR China*

Abstract

Very recently, for speeding up the computation of modular multi-exponentiation, Wu et al. presented a fast algorithm combining the complement recoding method and the minimal weight binary signed-digit representation technique. They claimed that the proposed algorithm reduced the number of modular multiplications from $1.503k$ to $1.306k$ on average, where the value k is the maximum bit-length of two exponents. However, in this paper, we show that their claim is unwarranted. We analyze the computational efficiency of Wu et al.'s algorithm by modeling it as a Markov chain. Our main result is that Wu et al.'s algorithm requires $1.471k$ modular multiplications on average.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Modular multi-exponentiation; Complement recoding; Minimal weight binary signed-digit (BSD) representation; Computational efficiency; Markov chain

1. Introduction

The computation of modular multi-exponentiation for large positive integer operands is required in many important applications in computer science and engineering. However, it is also a very time-consuming arithmetic operation, which requires a great deal of processing steps. Therefore, a significant problem is how to reduce the time needed to perform a modular multi-exponentiation operation. Many researchers have addressed this problem. For modular multi-exponentiation $A^X B^Y (\bmod N)$, assume the value k is the maximum bit-length of two exponents X and Y, i.e. $k = \lceil \log_2 \max(X, Y) \rceil$. Consider the average case. Pekmestzi's algorithm [1] uses $1.75k$ modular multiplications. Directly employing the minimal weight binary signed-digit (BSD) representation technique, Dimitrov et al.'s algorithm [2] requires $1.556k$ modular multiplications. Since there are many optimal BSD representations between X and Y, Dimitrov et al.'s algorithm can further reduce

* Corresponding author.
E-mail addresses: sundazhi1977@126.com, sundazhi@tju.edu.cn (D.-Z. Sun).

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.02.066

to $1.534k$ modular multiplications with eight reduction rules [2]. Solinas's algorithm [3] using the joint sparse form needs $1.503k$ modular multiplications.

Very recently, for speeding up the computation of modular multi-exponentiation, Wu et al. [4] presented a fast algorithm combining the complement recoding method and the minimal weight BSD representation technique. They claimed that their algorithm required only $1.306k$ modular multiplications on average. However, in this paper, we show that their claim is unwarranted. We analyze the computational efficiency of Wu et al.'s algorithm by modeling it as a Markov chain. Our main result is that Wu et al.'s algorithm requires $1.471k$ modular multiplications on average.

The remainder paper is organized as follows. We introduce some background knowledge in Section 2. In Section 3, we review the related modular multi-exponentiation algorithms. In Section 4, we analyze the computational efficiency of Wu et al.'s algorithm using a Markov chain. Finally, brief conclusions are given in Section 5.

2. Preliminaries

2.1. Minimal weight BSD representation

Definition 1. If $E = \sum_{i=0}^{k} d_i 2^i$ where $d_i \in \{0, 1, -1\}$ for $i = 0, 1, \ldots, k$, then $(d_k \ldots d_1 d_0)_{\mathrm{BSD}}$ is called a BSD representation for the integer E. $(d_k \ldots d_1 d_0)_{\mathrm{BSD}}$ is said to the minimal weight BSD representation $(d_k \ldots d_1 d_0)_{\mathrm{MWBSD}}$ if the representation has the smallest number of non-zero bits among all BSD representations.

Let E be positive integer whose binary representation is $(e_{k+1} e_k \ldots e_1 e_0)_2$ with $e_{k+1} = e_k = 0$. Menezes et al. [5] summarized an algorithm to recode an exponent to the minimal weight BSD representation, as depicted in Fig. 1.

2.2. Complement recoding method

In 2003, Chang et al. [6] proposed a method to reduce the Hamming weight of the exponent $E = (e_{k-1} \ldots e_1 e_0)_2$. Performing complement is perhaps advantageous in the speedup of the exponential computation. The equation is shown as follows:

$$E = \sum_{i=0}^{k-1} e_i 2^i = 2^k - \overline{E} - 1, \tag{1}$$

where $\overline{E} = (\overline{e_{k-1}} \cdots \overline{e_1}\,\overline{e_0})_2$ and $\begin{cases} \overline{e_i} = 0, & \text{if } e_i = 1 \\ \overline{e_i} = 1, & \text{if } e_i = 0 \end{cases}$ for $i = k-1, \ldots, 1, 0$.

For example, $E = 249 = (11111001)_2 = 2^8 - (00000110)_2 - 1 = 256 - 6 - 1$.

```
Algorithm A
Input: a positive integer $E = (e_{k+1} e_k e_{k-1} \cdots e_1 e_0)_2$ with $e_{k+1} = e_k = 0$.
Output: a minimal weight BSD representation $(d_k \cdots d_1 d_0)_{MWBSD}$ for $E$.
1. $c_0 = 0$;
2. For $i$ from 0 to $k$ do the following:
    2.1 $c_{i+1} = \lfloor (e_i + e_{i+1} + c_i)/2 \rfloor$;
    2.2 $d_i = e_i + c_i - 2c_{i+1}$;
3. Return $(d_k \cdots d_1 d_0)_{MWBSD}$.
```

Fig. 1. Minimal weight BSD exponent recoding.

3. Review of two modular multi-exponentiation algorithms

3.1. Dimitrov et al.'s algorithm without reduction rules

To compute modular multi-exponentiation $A^X B^Y (\text{mod} N)$, Dimitrov et al.'s algorithm without reduction rules can be described as Fig. 2. It is the basic modular multi-exponentiation algorithm using the minimal weight BSD representation technique.

Assume two exponents X and Y with uniform distribution. The expected average number of modular multiplications is $14k/9 \approx 1.556k$ [2].

3.2. Wu et al.'s algorithm

Wu et al. proposed a new modular multi-exponentiation algorithm. The main idea is to integrate the complement recoding method into Dimitrov et al.'s algorithm without reduction rules. Wu et al.'s algorithm can be depicted in Fig. 3. In [4], the original description of the transforming operations is $X = -\overline{(X)}_{\text{MWBSD}}$ and $Y = -\overline{(Y)}_{\text{MWBSD}}$ for the complement recoding method. But the representations of the values $-\overline{(X)}_{\text{MWBSD}}$ and $-\overline{(Y)}_{\text{MWBSD}}$ seem to be obscure. According to their design purpose, we make slight modifications in the corresponding steps. It has no any effect on the following computational efficiency analysis of Wu et al.'s algorithm, since the cost of those alterations is negligible.

4. Analysis of Wu et al.'s algorithm using Markov chain

We focus on the computational efficiency of Wu et al.'s algorithm, which combines the complement recoding method and the minimal weight BSD representation technique. Similarly to [4], assume all computations of values $A^{2^k}(\text{mod}N), B^{2^k}(\text{mod}N), A^{2^k}B^{2^k}(\text{mod}N)$ in Step 2 of **Algorithm C** are free. As a matter of fact, this requirement on Wu et al.'s algorithm restricts the application environments and increases extra overheads, compared with Dimitrov et al.'s algorithm without reduction rules.

Since two exponents are independently processed by Wu et al.'s combination strategy, we firstly discuss an arbitrary exponent E passing from Step 3 to Step 5 in **Algorithm C**. Let $\text{Ham}(E)$ be the Hamming weight of $E = (e_{k-1} \dots e_1 e_0)_2$. Clearly, the function of the complement recoding method, i.e. Step 4 of **Algorithm C**, can be regarded as

$$\text{Ham}(E) = \begin{cases} k - \text{Ham}(E), & \text{if } \text{Ham}(E) > \frac{k}{2}, \\ \text{Ham}(E), & \text{if } \text{Ham}(E) \leqslant \frac{k}{2}. \end{cases} \tag{2}$$

Algorithm B

```
Input: the parameters $A, B, N, X, Y$.
Output: the result $C = A^X B^Y (\text{mod } N)$.
1. $k = \lceil \log_2 \max(X, Y) \rceil$;
2. Recode $X, Y$ to minimal weight BSD representations
   $X = (x_k x_{k-1} \cdots x_1 x_0)_{MWBSD}, Y = (y_k y_{k-1} \cdots y_1 y_0)_{MWBSD}$;
3. Compute and store $A^{-1}(\text{mod } N), B^{-1}(\text{mod } N), A^{-1}B(\text{mod } N), AB^{-1}(\text{mod } N)$,
   $A^{-1}B^{-1}(\text{mod } N), AB(\text{mod } N)$;
4. $C = A^{x_k} B^{y_k} (\text{mod } N)$;
5. For $i$ from $k-1$ to $0$ do the following:
   5.1 $C = CC(\text{mod } N)$;
   5.2 If $(x_i, y_i) \neq (0,0)$ then $C = CA^{x_i} B^{y_i} (\text{mod } N)$;
6. Return $C$.
```

Fig. 2. Dimitrov et al.'s algorithm without reduction rules.

```
Algorithm C
Input : the parameters $A, B, N, X, Y$.
Output : the result $C = A^X B^Y \pmod N$.
1. $k = \lceil \log_2 \max(X, Y) \rceil$;
2. Precompute and store $A^{2^k} \pmod N, B^{2^k} \pmod N, A^{2^k} B^{2^k} \pmod N$;
3. Count the Hamming weights of $X$ and $Y$, denote as Ham($X$) and Ham($Y$);
4. If Ham($X$) $> \frac{k}{2}$ then $X = \overline{X}, A = A^{-1} \pmod N$, $HW_x = 1$ else $HW_x = 0$;
   If Ham($Y$) $> \frac{k}{2}$ then $Y = \overline{Y}, B = B^{-1} \pmod N$, $HW_y = 1$ else $HW_y = 0$;
5. Recode $X, Y$ to minimal weight BSD representations
   $X = (x_k \cdots x_1 x_0)_{MWBSD}, Y = (y_k \cdots y_1 y_0)_{MWBSD}$;
6. Compute and store $A^{-1} \pmod N, B^{-1} \pmod N, A^{-1}B \pmod N, AB^{-1} \pmod N$,
   $A^{-1}B^{-1} \pmod N$, $AB \pmod N$;
7. $C = A^{x_k} B^{y_k} \pmod N$;
8. For $i$ from $k-1$ to 0 do the following :
   8.1 $C = CC \pmod N$;
   8.2 If $(x_i, y_i) \neq (0,0)$ then $C = CA^{x_i} B^{y_i} \pmod N$;
9. If $(HW_x, HW_y) \neq (0,0)$ then $C = A^{2^k HW_x} B^{2^k HW_y} C A^{HW_x} B^{HW_y} \pmod N$;
10. Return $C$.
```

Fig. 3. Wu et al.'s algorithm.

Let $P(EV)$ denote the probability that the event EV occurs. Before the complement recoding, each bit of $E = (e_{k-1} \ldots e_1 e_0)_2$ assumes a value of 0 or 1 with equal probability, i.e. $P(e_i = 0) = P(e_i = 1) = 1/2$ for $0 \leqslant i \leqslant k-1$, and there is no dependency between any two bits. After the complement recoding, it should be a value of 0 or 1 with unequal probability, i.e. $P(e_i = 0) = 3/4$ and $P(e_i = 1) = 1/4$ for $0 \leqslant i \leqslant k-1$. Certainly, there is still no dependency between any two bits.

Consider the minimal weight BSD representation of $E = (e_{k-1} \ldots e_1 e_0)_2$, which has been transformed by the complement recoding method. According to **Algorithm A**, Table 1 [5] can list all possible inputs to the ith iteration of Step 2, and the corresponding outputs.

We can get the following simple facts:

$$P((e_i, c_i, e_{i+1}, c_{i+1}) = (0,0,0,0) | d_i = 0) = P((e_i, e_{i+1}) = (0,0)) = \frac{3}{4}\frac{3}{4} = \frac{9}{16}, \tag{3}$$

$$P((e_i, c_i, e_{i+1}, c_{i+1}) = (0,0,1,0) | d_i = 0) = P((e_i, e_{i+1}) = (0,1)) = \frac{3}{4}\frac{1}{4} = \frac{3}{16}, \tag{4}$$

$$P((e_i, c_i, e_{i+1}, c_{i+1}) = (1,1,0,1) | d_i = 0) = P((e_i, e_{i+1}) = (1,0)) = \frac{1}{4}\frac{3}{4} = \frac{3}{16}, \tag{5}$$

Table 1
Minimal weight BSD exponent recoding

Inputs								
e_i	0	0	0	0	1	1	1	1
c_i	0	0	1	1	0	0	1	1
e_{i+1}	0	1	0	1	0	1	0	1
Outputs								
c_{i+1}	0	0	0	1	0	1	1	1
d_i	0	0	1	−1	1	−1	0	0

$$P((e_i,c_i,e_{i+1},c_{i+1})=(1,1,1,1)|d_i=0)=P((e_i,e_{i+1})=(1,1))=\frac{1}{4}\frac{1}{4}=\frac{1}{16}, \tag{6}$$

$$\begin{aligned}P((e_{i+1},c_{i+1})&=(0,0)|d_i=1)\\ &=P(((e_i,c_i,e_{i+1},c_{i+1})=(0,1,0,0)\cup(1,0,0,0))|d_i=1)=1,\end{aligned} \tag{7}$$

$$\begin{aligned}P((e_{i+1},c_{i+1})&=(1,1)|d_i=-1)\\ &=P(((e_i,c_i,e_{i+1},c_{i+1})=(0,1,1,1)\cup(1,0,1,1))|d_i=-1)=1.\end{aligned} \tag{8}$$

In order to compute the averages of non-zero and zero bits, which potentially represent the number of modular multiplications, we can model the minimal weight BSD representation method using a Markov chain. In this situation, we define state variable S of the Markov chain is 0, 1, 2, when a bit 0, 1, -1 is formed using **Algorithm A**. The probability that state $S=i$ for $i=0,1,2$ succeeds state $S=j$ for $j=0,1,2$ is denoted by p_{ij}. We have:

$$\begin{aligned}p_{00}&=P(d_{i+1}=0|d_i=0)\\ &=P(((e_{i+1},c_{i+1})=(0,0)|d_i=0)\cap e_{i+2}=0)\\ &\quad+P(((e_{i+1},c_{i+1})=(0,0)|d_i=0)\cap e_{i+2}=1)\\ &\quad+P(((e_{i+1},c_{i+1})=(1,1)|d_i=0)\cap e_{i+2}=1)\\ &\quad+P(((e_{i+1},c_{i+1})=(1,1)|d_i=0)\cap e_{i+2}=0)\\ &=P((e_i,c_i,e_{i+1},c_{i+1})=(0,0,0,0)|d_i=0)P(e_{i+2}=0)\\ &\quad+P((e_i,c_i,e_{i+1},c_{i+1})=(0,0,0,0)|d_i=0)P(e_{i+2}=1)\\ &\quad+P((e_i,c_i,e_{i+1},c_{i+1})=(1,1,1,1)|d_i=0)P(e_{i+2}=1)\\ &\quad+P((e_i,c_i,e_{i+1},c_{i+1})=(1,1,1,1)|d_i=0)P(e_{i+2}=0)\\ &=\frac{9}{16}\frac{3}{4}+\frac{9}{16}\frac{1}{4}+\frac{1}{16}\frac{3}{4}+\frac{1}{16}\frac{1}{4}=\frac{5}{8},\end{aligned} \tag{9}$$

$$\begin{aligned}p_{01}&=P(d_{i+1}=1|d_i=0)\\ &=P(((e_{i+1},c_{i+1})=(1,0)|d_i=0)\cap e_{i+2}=0)\\ &\quad+P(((e_{i+1},c_{i+1})=(0,1)|d_i=0)\cap e_{i+2}=0)\\ &=P((e_i,c_i,e_{i+1},c_{i+1})=(0,0,1,0)|d_i=0)P(e_{i+2}=0)\\ &\quad+P((e_i,c_i,e_{i+1},c_{i+1})=(1,1,0,1)|d_i=0)P(e_{i+2}=0)\\ &=\frac{3}{16}\frac{3}{4}+\frac{3}{16}\frac{3}{4}=\frac{9}{32},\end{aligned} \tag{10}$$

$$\begin{aligned}p_{02}&=P(d_{i+1}=-1|d_i=0)\\ &=P(((e_{i+1},c_{i+1})=(1,0)|d_i=0)\cap e_{i+2}=1)\\ &\quad+P(((e_{i+1},c_{i+1})=(0,1)|d_i=0)\cap e_{i+2}=1)\\ &=P((e_i,c_i,e_{i+1},c_{i+1})=(0,0,1,0)|d_i=0)P(e_{i+2}=1)\\ &\quad+P((e_i,c_i,e_{i+1},c_{i+1})=(1,1,0,1)|d_i=0)P(e_{i+2}=1)\\ &=\frac{3}{16}\frac{1}{4}+\frac{3}{16}\frac{1}{4}=\frac{3}{32},\end{aligned} \tag{11}$$

$$\begin{aligned}p_{10}&=P(d_{i+1}=0|d_i=1)\\ &=P(((e_{i+1},c_{i+1})=(0,0)|d_i=1)\cap e_{i+2}=0)\\ &\quad+P(((e_{i+1},c_{i+1})=(0,0)|d_i=1)\cap e_{i+2}=1)\\ &=P((e_{i+1},c_{i+1})=(0,0)|d_i=1)P(e_{i+2}=0)\end{aligned} \tag{12}$$

$$
\begin{aligned}
&\quad + P((e_{i+1}, c_{i+1}) = (0,0) | d_i = 1) P(e_{i+2} = 1) \\
&= \frac{3}{4} + \frac{1}{4} = 1, \\
p_{20} &= P(d_{i+1} = 0 | d_i = -1) \\
&= P(((e_{i+1}, c_{i+1}) = (1,1) | d_i = -1) \cap e_{i+2} = 0) \\
&\quad + P(((e_{i+1}, c_{i+1}) = (1,1) | d_i = -1) \cap e_{i+2} = 1) \\
&= P((e_{i+1}, c_{i+1}) = (1,1) | d_i = -1) P(e_{i+2} = 0) \qquad (13) \\
&\quad + P((e_{i+1}, c_{i+1}) = (1,1) | d_i = -1) P(e_{i+2} = 1) \\
&= \frac{3}{4} + \frac{1}{4} = 1, \\
p_{11} &= p_{12} = p_{21} = p_{22} = 0. \qquad (14)
\end{aligned}
$$

Hence, the one step transition probability matrix of Wu et al.'s combination strategy is given as

$$
\mathrm{TPM}_{\text{Wu et al.'s strategy}} = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{bmatrix} = \begin{bmatrix} \frac{5}{8} & \frac{9}{32} & \frac{3}{32} \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \qquad (15)
$$

Let $p_j(k)$ for $j = 0, 1, 2$ denote the k-step transition probability of state j. When $k \to \infty$, the laws of the Markov chain imply the following equations:

$$
p_0(\infty) + p_1(\infty) + p_2(\infty) = 1, \qquad (16)
$$

$$
p_{00}p_0(\infty) + p_{10}p_1(\infty) + p_{20}p_2(\infty) = \frac{5}{8}p_0(\infty) + p_1(\infty) + p_2(\infty) = p_0(\infty), \qquad (17)
$$

$$
p_{01}p_0(\infty) + p_{11}p_1(\infty) + p_{21}p_2(\infty) = \frac{9}{32}p_0(\infty) = p_1(\infty). \qquad (18)
$$

We can obtain the solution $p_0(\infty) = 8/11$, $p_1(\infty) = 9/44$, $p_2(\infty) = 3/44$ from Eqs. (16)–(18). It means that the averages of non-zero and zero bits using Wu et al.'s combination strategy can be counted by $(p_1(\infty) + p_2(\infty))k = (9/44 + 3/44)k = 3k/11$ and $p_0(\infty)k = 8k/11$.

For modular multi-exponentiation $A^X B^Y (\mathrm{mod} N)$ in **Algorithm C**, we also assume two exponents X and Y with uniform distribution. Since $(8/11)^2 k$ of the pairs $\{(x_i, y_i), i = 0, 1, \ldots, k-1\}$ after Wu et al.'s combination strategy are expected to be (0, 0) on average, their algorithm requires $2k - (8/11)^2 k = 178k/121 \approx 1.471k$ modular multiplications.

5. Conclusions

When the computational cost related the complement recoding method is omitted, we analyze the computational efficiency of Wu et al.'s fast modular multi-exponentiation algorithm using a Markov chain. Although it combines the complement recoding method and the minimal weight BSD representation technique, Wu et al.'s algorithm at most requires up to $(14k/9 - 178k/121)/(14k/9) \approx 5.4\%$ fewer modular multiplications than Dimitrov et al.'s algorithm, which merely uses the minimal weight BSD representation technique.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 60673196 and the Applied Foundation Research Project of Tianjin under Grant No. 60673196.

References

[1] K. Pekmestzi, Complex number multipliers, IEE Proceeding-Computers and Digital Techniques 136 (1) (1989) 70–75.

[2] V.S. Dimitrov, G.A. Jullien, W.C. Miller, Complexity and fast algorithms for multiexponentiations, IEEE Transactions on Computers 49 (2) (2000) 141–147.

[3] J.A. Solinas, Low-weight binary representations for pairs of integers. Available from: <http://www.cacr.math.uwaterloo.ca/techreports/2001/tech_reports2001.html>.

[4] C.L. Wu, D.C. Lou, J.C. Lai, T.J. Chang, Fast modular multi-exponentiation using modified complex arithmetic, Applied Mathematics and Computation, in press, doi: 10.1016/j.amc.2006.08.051.

[5] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, 1997, p. 628.

[6] C.C. Chang, Y.T. Kuo, C.H. Lin, Fast algorithms for common multiplicand multiplication and exponentiation by performing complements, in: Proceedings of 17th International Conference on Advanced Information Networking and Applications, IEEE Computer Society, 2003, pp. 807–811.