アルゴリズム 55－4
（1997. 1. 23）

充足解探索問題の難しい例題生成手法について

堀江 聡

東京工業大学 情報理工学研究科 計算工学専攻
〒152 東京都目黒区大岡山 2-12-1
horie@cs.titech.ac.jp

あらまし 充足解探索問題に対し、難しい例題をランダムに生成する問題を考える。例えば素因数分解問題に対する難しい例題は簡単に生成することができるが、充足解探索問題に対する難しい例題を生成することは容易ではない。本論文では、素因数分解問題を還元することにより、充足解探索問題に対する難しい例題の生成手法を考えた。剰余計算を利用することで、生成される論理式の変数の個数が少なくよう改良した。

キーワード 充足解探索問題，例題生成手法，難しい例題

Hard Instance Generation for the Satisfying Assignment Search Problem

Akira Horie

Dept. of Computer Science, Tokyo Institute of Technology
Meguro-ku Ookayama 2-12-1, Tokyo 152, Japan
horie@cs.titech.ac.jp

Abstract. For the Satisfying Assignment Search Problem(in short, SAT), we consider a problem of generating its hard instance randomly. It is difficult to generate hard instances for SAT, while generating hard instances for the Factorization Problem, for example, is not so difficult. In this paper, we propose one instance generation algorithm that uses instances for the Factorization Problem to generating relatively hard instances for SAT. Using the modulus operation, we modify the algorithm so that generated instances have smaller number of variables.

Key Words satisfying assignment search problem, instance generation algorithm, hard instance

1 はじめに

本論文では、例題生成アルゴリズムについて考える。具体的には、充足解探索問題に対する難しい例題を生成するアルゴリズムを取り上げる。

アルゴリズムの性能を評価する方法として、数学的に解析する方法、例題を与え、実験的に評価する方法などが考えられる。このうち、実験的に評価する場合には、例題生成アルゴリズムが必要となる。例題生成アルゴリズムには、一様分布な例題を生成するもの、ある性質を持った例題のみを生成するものが考えられる。本論文では、ある性質を持った例題の一例として、難しい例題の生成について議論する。

例えば、素因数分解問題に対する例題として、大きな2つの素数の積が一般に難しいとされ、暗号の安全性の基にもなっている。この例題は、素数判定が比較的易しいことから、大きな数をランダムに生成し、素数判定をし、「素数」と判定された2数を掛け合わせることで、容易に生成できる。一方、充足解探索問題に対する例題として、どのような例題が一般に難しいのかわかっておらず、難しい例題を容易に生成しにくい。

そこで、本論文では、素因数分解問題から充足解探索問題への還元を考え、素因数分解問題に対する難しい例題から、充足解探索問題に対する難しい例題を生成することを考える。

最も素直な方法として、次のような還元が考えられる。入力 x に対し、2数 p,q を推測し、実際に $p\times q$ を計算し、それが x と等しいかを判定する論理回路を考え、その論理回路を論理式に変換する。

ところが、この還元によって生成される論理式は、乗算の計算に手間がかかるため、n ビットの入力 x に対して、$O(n^2)$ 個の変数が必要となってしまう。そこで、剰余計算を利用することにより、被乗数、乗数の桁数を小さくすることで、手間を軽減する。また、mod の計算が効率よくできるように、法となる数を工夫する。このような改良により、本論文では、変数の個数がより少ない論理式を生成する還元を得た。具体的には、変数の個数が $O(n^{3/2}(\ln n)^{1/2})$ に改良し、さらにそれを再帰的に用いることで $O(n\ln n)$ に減らすことができた。

2 準備

ここでは、基本的な用語や記号の定義を行なう.

n 変数の論理式 f に対して、長さ n の 0,1-列 $t\in 0,1^n$ を、f に対する (真偽値)割当という。割当 t に対する $f(t)$ の値は、f の各変数 x_i に対して、t の i ビット目 (t_i) を代入して得られる値である。$f(t)=1$ となるとき、t を f の充足解という。与えられた論理式に対して、充足解を (あれば) 求める問題を、充足解探索問題と呼ぶ。

自然数 x を2進表記したとき、そのビット数を長さとして $|x|$ と表す。

与えられた自然数 x に対して、trivial でない2つの因数を求める問題を、素因数分解問題と呼ぶ。つまり、$x=p\times q$ かつ $p,q\geq 2$ となる p,q を求める問題である。ここでは、$|p|=|q|$ となるような x のみを考える。

定義 2.1. 素因数分解問題から充足解探索問題への還元 $h(x)$ を次のように定義する。

1. h は自然数から論理式への関数。
2. x が素数のとき、論理式 $h(x)$ は充足不可能。
3. x が合成数のとき、論理式 $h(x)$ は充足解 t を持ち、その t から因数 p,q を構成できる。

ここでは、入力 x の長さ $|x|$ を n ビットとし、還元 $h(x)$ は、n ごとに考える。

この還元により、論理式 $h(x)$ の充足解を求めることは、少なくとも x を素因数分解すること以上の難しさを持つ。

3 単純な還元

まず、最も単純な還元方法として、$p \times q$ を計算し、それが x と等しいかを判定する論理回路を構成し、論理式に変換することを考える。この方法を単純な還元と呼ぶ。

入力 x の長さを n ビットとする。このとき、$|p| = |q|$ より, p, q の長さは $\frac{n}{2}$ ビットである。$\frac{n}{2}$ ビット全加算器を $\frac{n}{2} - 1$ 段用いて、筆算の要領で、$p \times q$ を計算する回路を構成する。

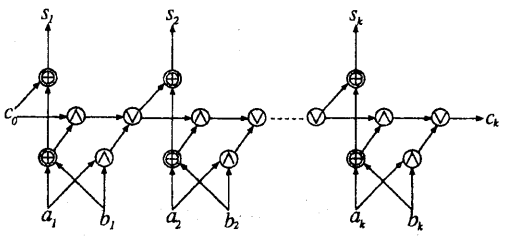

図 1: k-bit 全加算器

k ビット全加算器は (図 1) に示すように、$5k$ ゲートで構成できる。$p \times q$ を計算する回路および乗算の結果が x と等しいかどうかを判定をする回路は (図 2) に示すように、$\frac{3}{2}n^2 - \frac{1}{2}n$ ゲートで構成できる。

論理回路から論理式への変換は以下のように行なう。

各ゲートに、ユニークな変数を割り当てる。全ゲートに対し、以下のような項をすべて論理積でつなぐ。

- ゲート g が、入力が h である否定のゲートのとき $(\overline{g} \vee h)(g \vee \overline{h})$
- ゲート g が、入力が h_1, h_2 である論理和のゲートのとき $(\overline{g} \vee h_1 \vee h_2)(g \vee \overline{h_1})(g \vee \overline{h_2})$
- ゲート g が、入力が h_1, h_2 である論理積のゲートのとき $(g \vee \overline{h_1} \vee \overline{h_2})(\overline{g} \vee h_1)(\overline{g} \vee h_2)$
- ゲート g が、入力が h_1, h_2 である排他的論理和のゲートのとき $(\overline{g} \vee h_1 \vee h_2)(\overline{g} \vee \overline{h_1} \vee \overline{h_2})(g \vee \overline{h_1} \vee h_2)(g \vee h_1 \vee \overline{h_2})$

したがって、生成される論理式の変数の個数は、回路のゲート数に等しい。

定理 3.1. 単純な還元により。n ビットの自然数 x に対して、変数の個数 $\left(\frac{3}{2}n^2 - \frac{1}{2}n\right)$ の論理式が生成される。

4 剰余を利用した還元

4.1 剰余の利用

定理 4.1. (中国人の剰余定理)
$m_1, m_2, \ldots, m_k$ を互いに素とし、$m = m_1 m_2 \cdots m_k$ とする。$u_1, u_2, \ldots, u_k$ を整数とする。このとき、mod m のもとで、次の条件を満たす唯一の整数 u が存在する。

$$u \equiv u_j \pmod{m_j} \quad \text{for } 1 \leq j \leq k$$

この中国人の剰余定理を利用し、$\frac{n}{2}$ビット同士の乗算 $p \times q$ に代わって、ビット数のより小さいビット同士の乗算 $p_j \times q_j$ ($p \equiv p_j$, $q \equiv q_j \pmod{m_j}$) を計算し、$x \equiv x_j$ と等しいかを判定させることとする。この方法を、剰余を利用した還元と呼ぶ。ここで、$m_1, \ldots, m_k$ は、$|m| > n$ となるように選べばよい。

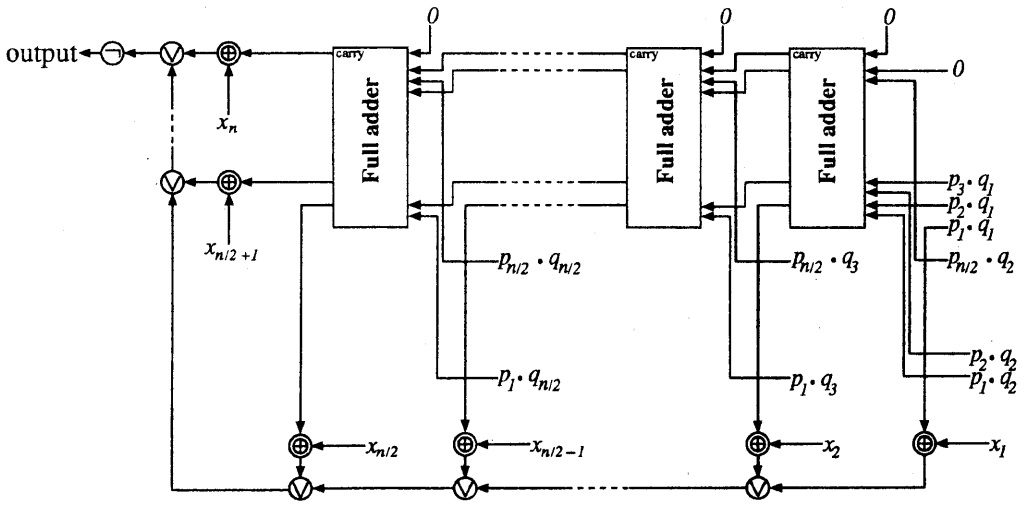

図 2: $p \times q$を計算し, x と比較する回路

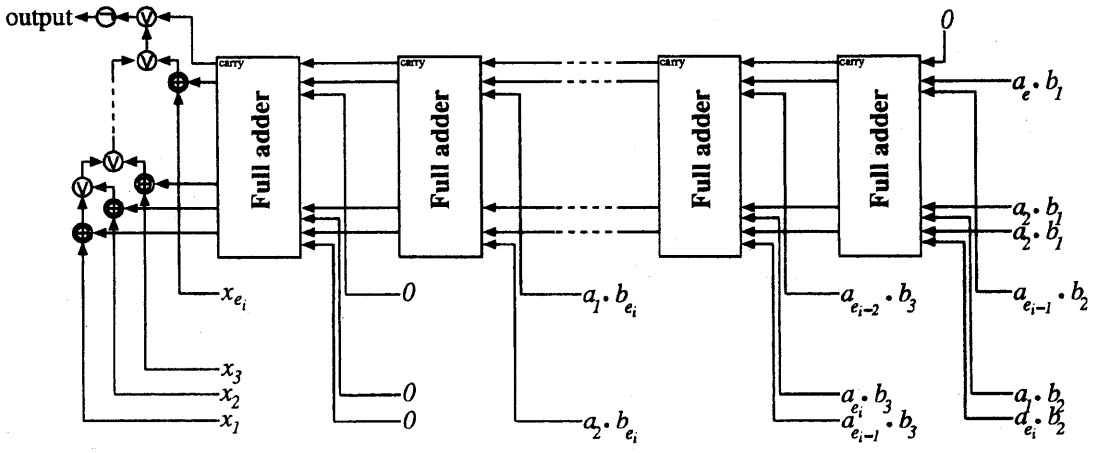

図 3: $\pmod{m_i}$ のもとで, $p_i \times q_i$を計算し, x_i と比較する回路

さらに、法として、$m_j = 2^{e_j} - 1$ を選ぶと、$\bmod m_j$ の計算を効率的に行なうことができる。

補題 4.2. n ビットの自然数 x に対して、$x \equiv x' \pmod{2^e - 1}$ となる x' は、ゲート数 $5e\left(\left\lfloor \frac{n}{e} \right\rfloor + 2\right)$ の論理回路で計算できる。

証明. x の2進表記において、e ビットごとまとめて考えると、$A = 2^e$, $t = \left\lfloor \frac{n}{e} \right\rfloor$, $0 \le a_k < 2^e$ for $0 \le k \le t$ として、

$$x = a_t A^t + a_{t-1} A^{t-1} + \cdots + a_1 A + a_0$$

$A \equiv 1 \pmod{2^e - 1}$ であるから、

$$x \equiv a_t + a_{t-1} + \cdots + a_0 \pmod{2^e - 1}$$

となる。

$(\bmod 2^e - 1)$ のもとでの加算を考える。u, v の和 w は、

$$w = \begin{cases} u + v & \text{if } u + v < 2^e \\ ((u + v) \bmod 2^e) + 1 & \text{if } u + v \ge 2^e \end{cases}$$

となる。つまり、桁上げを最下位ビットに加えればよい。

$$u + v \ge 2^e \Leftrightarrow (u + v) \text{ の } e \text{ ビット目が } 1$$

であるから、

$w = ((u + v) \bmod 2^e) + (u + v)_e$.

ただし、$(u + v)_e$ とは、$(u + v)$ の e ビット目。これより、$u + v$ は、全加算器2つで実現できる。

ここで、$a_t + a_{t-1} + \cdots + a_0$ の繰り返し加算において、桁上げの処理を次の加算に持ち越す桁上げ保存加算を用いる。加算において発生した桁上げをすぐに最下位ビットに加えるのではなく、次の加算において桁上げ入力として扱う。つまり、各加算では中間和と桁上げを求め、最後に中間和と桁上げを加算して最終的な和を求める。すなわち、

$$\begin{aligned} a_0 + a_1 &= c_1 \cdot 2^e + s_1, \\ s_1 + a_2 + c_1 &= c_2 \cdot 2^e + s_2, \\ &\vdots \\ s_{t-1} + a_t + c_{t-1} &= c_t \cdot 2^e + s_t, \\ s_t + c_t &= c' \cdot 2^e + s', \\ s' + c' &= s. \end{aligned}$$

このうち最後の2つの加算, $s_t + c_t$, $s' + c'$ は、次のような場合があるために必要である。$s_{t-1} = 2^e - 1, a_t = 2^e - 1, c_{t-1} = 1$ のとき、$s_{t-1} + a_t + c_{t-1} = 1 \cdot 2^e + (2^e - 1)$ となる。このとき、$s_t + c_t = 2^e - 1 + 1 = 1 \cdot 2^e + 0$ と桁上げが発生する。この処理のために最後の加算が必要である。

以上、加算は全部で、$\left\lfloor \frac{n}{e} \right\rfloor + 2$ 回である。e ビット全加算器のゲート数は $5e$ であるから、補題が証明された。

(証明終)

$2^e - 1$ の形の整数で、互いに素なものが十分存在することを、次の補題で示す。

補題 4.3. $\gcd(2^e - 1, 2^f - 1) = 2^{\gcd(e,f)} - 1$

証明. まず、

$2^e \equiv 2^f \pmod{2^g - 1} \Leftrightarrow e \equiv f \pmod{g}$

を証明する。

$(\Leftarrow)$ $e = f + kg$ とすると、$2^e = 2^f (2^g)^k \equiv 2^f 1^k \pmod{2^g - 1}$ である。

$(\Rightarrow)$ $2^e \equiv 2^f \pmod{2^g - 1}$ とすると、$2^g \equiv 1 \pmod{2^g - 1}$ であるから、$2^{e \bmod g} \equiv 2^{f \bmod g} \pmod{2^g - 1}$。$2^{e \bmod g}, 2^{f \bmod g} < 2^g$ より、$2^{e \bmod g} = 2^{f \bmod g}$。したがって、$e \equiv f \pmod{g}$ である。

これより、$(2^e - 1) \bmod (2^f - 1) = 2^{e \bmod f} - 1$ となる。したがって、ユークリッドの互除法を

用いて、

$$\begin{aligned}\gcd(2^e-1,2^f-1) &= \gcd(2^f-1,2^{e \bmod f}-1)\\ &\vdots\\ &= 2^{\gcd(e,f)}-1\end{aligned}$$

(証明終)

この補題は、2^e-1 と 2^f-1 が互いに素となるのは、e と f が互いに素のときであり、またその場合に限ることを示している。

定理 4.4. 剰余を利用した還元により、n ビットの自然数 x に対して、次の個数以下の変数を持つ論理式が生成される。

$$\sum_{i=1}^{k}\left(6e_i^2+32e_i+5e_i\left(\left\lfloor\frac{n}{e_i}\right\rfloor+2\left\lfloor\frac{n}{2e_i}\right\rfloor\right)+2\right)$$

ただし、$e_1,e_2,\ldots,e_k$ は、$e_1+e_2+\cdots+e_k>n$ となる、互いに素な整数とする。

証明. 各 $i(1\le i\le k)$ に対して、$x\equiv x_i, p\equiv p_i, q\equiv q_i \pmod{2^{e_i}-1}$ を計算するのに必要なゲート数は $|x|=n, |p|=|q|=\dfrac{n}{2}$ であるから,

$$5e_i\left(\left\lfloor\frac{n}{e_i}\right\rfloor+2\left\lfloor\frac{n}{2e_i}\right\rfloor\right)+30e_i.$$

次に、$p_i\times q_i \pmod{2^{e_i}-1}$ の計算およびその結果が x_i と等しいかの判定は、前節で示した単純な還元と同様にで行なうが、$\pmod{2^{e_i}-1}$ のもとでの計算であることに留意して、以下のように行なう。

今、p_i, q_i をそれぞれ $p_i=\sum_{j=0}^{e_i-1}a_j2^j, q_i=\sum_{j=0}^{e_i-1}b_j2^j$ とする。$2^{e_i}\equiv 1 \pmod{2^{e_i}-1}$ を考慮すると,

	a_{e_i}	a_{e_i-1}	$\cdots$	$\cdots$	a_1
$\times$	b_{e_i}	b_{e_i-1}	$\cdots$	$\cdots$	b_1
	$b_1a_{e_i}$	$b_1a_{e_i-1}$	$\cdots$	$\cdots$	b_1a_1
	$b_2a_{e_i-1}$	$\cdots$	$\cdots$	b_2a_1	$b_2a_{e_i}$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$+$	$b_{e_i}a_1$	$b_{e_i}a_{e_i}$	$\cdots$	$\cdots$	$b_{e_i}a_2$
	t_{e_i}	t_{e_i-1}	$\cdots$	$\cdots$	t_1

のようになる。繰り返し加算については、桁上げ保存加算を用いる。この方法で、乗算を計算し、x_i と等しいかどうか判定する回路は (図 3) で示すように、$6e_i^2+2e_i+1$ ゲートで構成できる。

これらを加えて

$$6e_i^2+32e_i+5e_i\left(\left\lfloor\frac{n}{e_i}\right\rfloor+2\left\lfloor\frac{n}{2e_i}\right\rfloor\right)+1$$

すべての i について、論理回路から論理式を構成し、論理積で結ぶので、生成される論理式の変数の個数は

$$\sum_{i=1}^{k}\left(6e_i^2+32e_i+5e_i\left(\left\lfloor\frac{n}{e_i}\right\rfloor+2\left\lfloor\frac{n}{2e_i}\right\rfloor\right)+2\right)$$

(証明終)

例 4.1. 入力 x が $n=500$ ビットの場合を考える。次のような m_i を選ぶ。

$m_1=2^{53}-1,\ m_2=2^{51}-1,\ m_3=2^{49}-1$
$m_4=2^{47}-1,\ m_5=2^{43}-1,\ m_6=2^{41}-1$
$m_7=2^{37}-1,\ m_8=2^{32}-1,\ m_9=2^{31}-1$
$m_{10}=2^{29}-1,\ m_{11}=2^{25}-1,\ m_{12}=2^{23}-1$
$m_{13}=2^{19}-1,\ m_{14}=2^{13}-1,\ m_{15}=2^{11}-1$
$m=m_1\cdots m_{15}>2^{504}.$

この還元では、$n=100$ のとき、約 20 万変数の論理式が生成される。これに対し、単純な還元では、約 37 万変数の論理式となる。

4.2 評価

剰余を利用した還元により、n ビットの自然数のとき、変数の個数がどのくらいの論理式が生成されるか、上限について考える。

$m=m_1\cdots m_k>2^n$ となるように、互いに素な $m_1,\ldots,m_k$ を選ぶことになる。一方、$m_j=2^{e_j}-1$ であるから、$m<2^{e_1+\cdots+e_k}$。さらに、補題 4.3 を考え合わせると、$e_1+\cdots+e_k>n$ となるような、互いに素な $e_1,\ldots,e_k$ を選ぶことになる。

$e_1 > e_2 > \cdots > e_k$ とする。ここで、互いに素な $e_1,\ldots,e_k$ の代わりに、素数の組 $e'_1,\ldots,e'_k (e'_1 > e'_2 > \ldots > e'_k,\ e'_1 + \cdots + e'_k > n)$ を考えると、明らかに $e_1 < e'_1$ となる。以下では、この e'_1 を使って評価する。

e'_1 以下の素数の和を考える。素数定理 $\pi(x) \sim \frac{x}{\ln x}$ を用いると、

$$\sum_{p \le e'_1} p \sim \frac{e'^2_1}{2 \ln e'_1}$$

よって、$\frac{e'^2_1}{2 \ln e'_1} \ge n$ より、$e'_1 = O((n \ln n)^{1/2})$ を得る。

次に、$\sum_{i=1}^{k} e_i^2 < \sum_{i=1}^{k} e'^2_i < \sum_{e'_1 \le p} p^2$ を評価する。ここで、$\sum_{i=1}^{n} a_i b_i$ を考える。$A(j) = \sum_{i=1}^{j} a_i$ とおくと、$\sum_{i=1}^{n} a_i b_i = \sum_{j=1}^{n-1} A(j)(b_j - b_{j+1}) + A(n) b_n$ となる。a_i を i が素数のとき 1、それ以外のとき 0 とし、b_i を i^2 とすると、$A(k) = \pi(k)$ であり、$\sum_{e'_1 \le p} p^2 = \sum_{i=1}^{e'_1} a_i b_j$ となる。よって、

$$\begin{aligned} \sum_{e'_1 \le p} p^2 &= \sum_{j=1}^{e'_1 - 1} (-2j-1)\pi(j) + \pi(e'_1) e'^2_1 \\ &= O\left(\frac{e'^3_1}{\ln e'_1}\right) \\ &= O(n^{3/2} (\ln n)^{1/2}) \end{aligned}$$

これより、

$$\sum_{i=1}^{k} \left(6e_i^2 + 32e_i + 5e_i \left(\left\lfloor \frac{n}{e_i} \right\rfloor + 2 \left\lfloor \frac{n}{2e_i} \right\rfloor\right) + 2\right)$$

$$= O(n^{3/2} (\ln n)^{1/2})$$

定理 4.5. 剰余を利用した還元により、n ビットの自然数 x に対して、変数の個数 $O(n^{3/2}(\ln n)^{1/2})$ の論理式が生成される。

ここで、$p_i \times q_i \pmod{2^{e_i} - 1}$ と x_i が等しいかどうかの判定を単純な還元でなく、剰余を利用した還元を再帰的に利用することを考える。再帰の深さを i とするとき、生成される論理式の変数の個数は、$O(n^{1+1/2^i} (\ln n)^{1-1/2^n})$ となることがわかる。$i > \log \log n$ とすると、これは $O(n \ln n)$ になる。

定理 4.6. 剰余を利用した還元を再帰的に用いることにより、n ビットの自然数 x に対して、変数の個数 $O(n \ln n)$ の論理式が生成される。

この結果は、単純な還元において、乗算を現在最も効率のよい乗算法と考えられている Schönhage and Strassen 法 [4] を用いた場合の変数の個数 $O(n \log n \log \log n)$ よりも効率がよい。

5 おわりに

充足解探索問題に対する難しい例題の生成に対し、素因数分解問題からの還元を考えた。n ビットの自然数に対して、生成される論理式の変数の個数を単純な方法による $O(n^2)$ から、$O(n^{3/2}(\ln n)^{1/2})$ に、さらに再帰的に利用することで $O(n \ln n)$ にまで改良した。今後の課題としては、さらに変数の個数を減らすことができないか、また他の問題からの還元などが考えられる。

参考文献

[1] D. E. Knuth, *Seminumerical Alogorithms*, volume 2 of *The Art of computer programming*. Addison-Wesley, 1969. Second edition 1981.

[2] 元木 光雄, 充足不可能な論理式例題の生成に関する研究, 卒業論文, 東京工業大学 1996.

[3] 高木 直史, 算術演算回路のアルゴリズム 1. 加算回路のアルゴリズム, 情報処理, Vol.37, No.1, pp.80-85 (1996).

[4] Schönhage and Strassen, *Schnelle Multiplikation großer Zalhen*, Computing 7, pp.281-292, (1971).