数理解析研究所講究録 1325 巻 2003 年 45-50

節点重み最大クリーク抽出に基づく量子回路の深さ最小化

名久井 行秀 (Yukihide Nakui)* 西野 哲朗 (Tetsuro Nishino)†
富田 悦次 (Etsuji Tomita)† 中村 知倫 (Tomonori Nakumura)†
* 電気通信大学大学院 電気通信学研究科 電子情報学専攻
(The Graduate School of Electro-Communications)
† 電気通信大学 電気通信学部 情報通信工学科 情報メディア工学講座
(The University of Electro-Communications)

1 はじめに

現在, 我々が使用しているコンピュータは, 基本的に論理回路から構成されている. これは, 計算可能な関数が, すべて論理関数として表現可能であることに基づいている. つまり, 論理関数の計算は, すべての演算の基礎であり, これを効率的に計算することは, 本質的に重要である.

量子力学の概念に基づく量子コンピュータにおいても, 論理関数の計算は, 非常に基本的なタスクである. したがって, 量子回路上で, 論理関数を構成するための方法や, その最小化, 簡単化の方法を確立することは大変重要である.

一方, 最大クリーク抽出問題に対する厳密解を高速に得るためのアルゴリズムが提案されている [4]. そして, そのアルゴリズムを利用して節点重み最大クリーク抽出アルゴリズムが提案された [6]. 本稿で取り扱う量子論理回路の深さ最小化問題は, 入力サイズに対する多項式時間ではその解を求めることができないと強く予想される. このため量子論理回路の最小化問題をクリーク抽出問題に還元することで, 高速にその解を求めることができると期待される.

本研究では, 量子論理回路の深さ最小化問題を, 節点重み最大クリーク抽出問題に還元させて解く方法について述べる. また, その還元アルゴリズムの実働化も行ったので, そのことについても報告する.

2 量子計算

まず, 量子計算について簡単に説明する.

量子コンピュータのシステムの状態は, 2^n 次元複素ベクトル空間 V のベクトルで表される. 特に計算基底状態 $|0\rangle, |1\rangle$ を

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1}$$

ととる.

システムの状態は, システムに固有な, $2^n \times 2^n$ のユニタリ変換 U によって推移すると考える:

$$U|x\rangle = |y\rangle, \quad (UU^* = U^*U = I). \tag{2}$$

複合的な量子システムの状態空間は, その空間のテンソル積ベクトル空間で与えられる.

状態 $|x\rangle$ と $|x\rangle$ の線形結合もまた, その空間 V の状態である. すなわち, $\alpha, \beta \in \boldsymbol{C}$ としたとき,

$$\alpha|x\rangle + \beta|y\rangle \in V, \quad |\alpha|^2 + \beta|^2 = 1 \tag{3}$$

となる. (**重ね合わせの原理**)

この重ねあわせ状態を観測すると,

- 確率 $|\alpha|^2$ で, 状態 $|x\rangle$
- 確率 $|\beta|^2$ で, 状態 $|y\rangle$

が, 観測される.

観測を行うと重ねあわせ状態は失われ, 観測された状態へ推移する. (**波束の収縮**)

3 量子回路, 量子ゲート

この節では, 基本的な量子ゲートを定義する.

定義 3.1 (NOT ゲート)
論理否定 NOT ($\bar{x}$ で表記する.) を計算する量子ゲートは, ユニタリ作用素

$$N \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4}$$

で与えられる. すなわち

$$\forall x \in \{0,1\}, \quad N|x\rangle = |\bar{x}\rangle = |\mathrm{NOT}(x)\rangle \tag{5}$$

が成り立つ. このユニタリ作用素を **NOT ゲート**と呼び, 図 1 のように表す.

定義 3.2 (C-NOT ゲート)
次のような論理関数

$$\forall x, y \in \{0,1\},$$
$$f(x,y) \equiv (x, x \oplus y) = (x, \mathrm{EXOR}(x,y)) \tag{6}$$

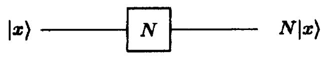

図 1: NOT ゲート

を計算する量子ゲートは,

$$C_y^x \equiv |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes N \tag{7}$$

で与えられる. すなわち

$$\forall x, y \in \{0,1\}, \quad C_y^x|x,y\rangle = |x, x \oplus y\rangle \tag{8}$$

が成り立つ. ここで N は式 (4) で定義されたものであり, I は, 恒等作用素である. また, $x \oplus y = \mathrm{EXOR}(x,y)$ は, x と y の排他的論理和を表す. この量子ゲートを**制御 NOT ゲート**(controlled NOT gate, あるいは, **C-NOT gate**) と呼ぶ. これを, 図 2 のように表す.

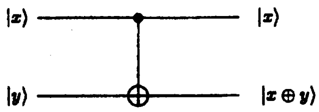

図 2: 制御 NOT ゲート

このダイアグラムは,

- $x = 0$ のときは, N が無効になり, 入力 $|x,y\rangle$ がそのまま出力される. ($0 \oplus y = y$ であるため)
- $x = 1$ のときは, N が有効になり, 出力は $|x,\bar{y}\rangle$ となる. ($1 \oplus y = \bar{y}$ であるため)

とみることができる. この意味で $|x\rangle$ を**制御ビット**(control bit), $|y\rangle$ を**目標ビット**(target bit) を呼ぶ.

定義 3.3 (Toffoli ゲート)
U を 2×2 のユニタリ行列とする. $n = 1, 2, \ldots$ に対して, $2^{n+1} \times 2^{n+1}$ のユニタリ行列 $\Lambda_n U$ を次のように定め, これを $(n+1)$ 量子ビットの**一般 Toffoli ゲート**と呼ぶ. すなわち,

$$\begin{aligned} &\forall x_1, \ldots, x_n, y \in \{0,1\}, \\ &\Lambda_n U|x_1, \ldots, x_n, y\rangle \\ &\equiv \begin{cases} |x_1, \ldots, x_n\rangle \otimes U|y\rangle, & \text{if} \quad x_1 \wedge \cdots \wedge x_n = 1 \\ |x_1, \ldots, x_n, y\rangle, & \text{if} \quad x_1 \wedge \cdots \wedge x_n = 0 \end{cases} \end{aligned} \tag{9}$$

と定める. ここで, $a \wedge b$ は a, b の論理積を表す.

特に U が $U = N$ であるとき, $\Lambda_n N$ を $(n+1)$ 量子ビットの **Toffoli ゲート** T_y^x と呼ぶ. そして, これを図 3 のように表すことにする.

$n = 0$ のときの Toffoli ゲートは, NOT ゲート N であるとする. また, $n = 1$ のときは, C-NOT ゲート (controlled not ゲート) に相当する.

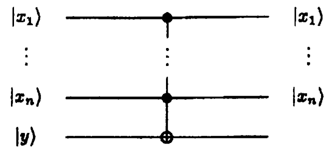

図 3: Toffoli ゲート

4 量子論理回路の深さ最小化

定義 4.1 (量子論理回路)
本稿では, ユニタリ変換 $U_{f\text{-C-NOT}}$

$$U_{f\text{-C-NOT}}|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle \tag{10}$$

を計算する回路 (**f-C-NOT 回路** [1]) が**量子論理回路**であるとする. f-C-NOT は, function-controlled-NOT をあらわす.

さらに, 本稿ではこの量子論理回路 (f-C-NOT 回路) に対して次の制約を与える:

- $|x\rangle$ は, $n(\geq 1)$ 量子ビットで構成され, 制御ビットとしてのみに用いられる.
- $|y\rangle$ は, 1 量子ビットで構成され, 目標ビットとしてのみに用いられる.
- 補助ビットは, 用いない.
- 構成要素は, NOT ゲートと Toffoli ゲートのみである.

f-C-NOT 回路は, 最小項に対応するユニタリ変換である**最小項ゲート**で簡単に構成されうる. 最小項ゲート M_y^x は,

$$M_y^x \equiv \left(\prod N_{x_i}\right) T_y^x \left(\prod N_{x_i}\right) \tag{11}$$

のように NOT ゲートで挟まれた 1 つの $(n+1)$ ビット Toffoli ゲートによって実現されうる. ここで N_{x_i} と T_y^x は, それぞれ, 制御レジスタの第 i qubit の状態だけを反転する NOT ゲートと, n 個の制御ビット x と 1 個の目標ビット y を持つ Toffoli ゲートを表す. そして, f-C-NOT 回路は,

$$U_{f\text{-C-NOT}} = \prod M_y^x \tag{12}$$

のように, その最小項ゲートすべての積で得られる. ここで, この積は, 最小項 $m(x)$ について $m(x) = 1$ を満たす x に関する (すなわち, true miniterm に関する) ものである. また, どの最小項ゲートも, 入力の 1 つの状態だけに対して動作するので, 最小項ゲートの積の順序は可換である.

次に, 積項ゲートを定義する. 各変数に対して高々 1 つのリテラル $(x, \bar{x})$ を論理積で結合した式を**積項**と呼ぶことにする. すなわち, 最小項に対して don't care bit を許したものが積項である. そして, Toffoli

ゲートを NOT ゲートで挟み，この積項に対応する状態で制御されるようにしたゲートを**積項ゲート**と呼ぶことにする．

これらの積項ゲートに関して次の命題が示される [5].

【命題】 4.1
積項ゲートの積は，対応する論理表現においては，その積項の排他的論理和 (EXOR) で表すことができる．

ここで，任意の積項を排他的論理和で結合した AND-EXOR 論理式を **ESOP(Exclusive-or Sum Of Product)** と定義する．また，論理関数 f を ESOP で表現したときに積項数が最小になる ESOP を論理関数 f の**最小 ESOP** と定義する．

そして，これまでの命題から，次のことが示される [5].

定理 4.1 (量子論理回路の深さ最小化)
論理関数 f の最小 ESOP は，f-C-NOT 回路に対する深さ最小の量子回路を与える．

5 ESOP 最小化問題の定式化

ここまでの議論で，f-C-NOT 回路の深さ最小化問題が，ESOP の最小化問題に深く関係があることがわかった．この節では，ESOP の最小化問題について述べる．

ESOP における積項による最小項の被覆関係は，偶奇性を持つカバー問題であり，以下の **Helliwell 方程式** で表される．[2]

$$H(g) = H(g_0, g_1, \ldots, g_{\xi-1}) = \bigwedge_{i=0}^{N-1} S_i = 1 \quad (13)$$

ここで，$S_i = \left(\bigoplus_{g_j \in T_i} g_j \right) \oplus f(a_i) \oplus 1$ であり，N は，最小項の個数 $(= 2^n)$，ξ は，すべての可能な積項の数 $(= 3^n)$ である．そして，T_i は，i 番目の最小項を被覆する積項の集合であるとする．$f(a_i)$ は，i 番目の最小項を 1 にするような入力が与えられたときの f の値である．$(f(a_i) \in \{0,1\})$ g_j は，$g_j = 1$ のとき，j 番目の積項 g_j が，ESOP に含まれることを意味する変数とする．

すなわち，S_i は，i 番目の最小項が，偶奇性 (つまり，true minterm ならば積項によって奇数回被覆され，false minterm ならば偶数回被覆される) を満たすための条件をあらわす．そして，$H(g_0, g_1, \ldots, g_{\xi-1}) = 1$ は，すべての最小項において，偶奇性が満たされていることを意味する．

したがって，ESOP の最小化問題は，$H(g) = 1$ を満たす g の中で，最小の割り当てのものを求める問題といえる．ここで g の最小の割り当ては，$(\sum_{i=0}^{\xi-1} g_i)$ を最小にするものをさすものとする．

例 1
2 変数関数の場合 Helliwell 方程式は以下のようになる．

$$\begin{aligned} H(g) &= H(g_0, g_1, \ldots, g_8) \\ &= (g_0 \oplus g_4 \oplus g_6 \oplus g_8 \oplus f(0,0) \oplus 1) \\ &\quad \cdot (g_1 \oplus g_4 \oplus g_7 \oplus g_8 \oplus f(0,1) \oplus 1) \\ &\quad \cdot (g_2 \oplus g_5 \oplus g_6 \oplus g_8 \oplus f(1,0) \oplus 1) \\ &\quad \cdot (g_3 \oplus g_5 \oplus g_7 \oplus g_8 \oplus f(1,1) \oplus 1) = 1 \quad (14) \end{aligned}$$

ここで，$g_0, g_1, \ldots, g_8$ については，図 4 に示す．

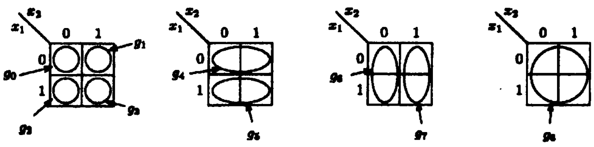

図 4: 2 変数の積項の対応図

しかしながら，効率的に g の最小の割り当てを求めるアルゴリズムは，知られていない．そこで，この問題を CLIQUE 問題に還元させることを考える．これは，最大クリーク抽出問題の厳密解を高速に求めるアルゴリズムが提案されており [4],[6]，この還元により，比較的高速にその最小割り当てを求めることができると期待されるためである．

6 CLIQUE への還元

この節において，Helliwell 方程式の解法を CLIQUE 問題へと還元する．まず，CLIQUE 問題について述べる．CLIQUE 問題は，次のように定義される．[3]

CLIQUE

INSTANCE:

グラフ $G = (V, E)$ と正整数 $K \le |V|$.

QUESTION:

G は，サイズ K 以上のクリークを持つか?

6.1 Helliwell 方程式から 3-ESOP へ

次に，Helliwell 方程式の解法を CLIQUE 問題へと還元する方法について述べていくが，理解を助けるために簡単な具体例を用いて説明していく．例えば，2 変数論理関数に対する Helliwell 方程式 (14) から，以下の連立方程式が導き出せる．(ただし，$f(0,0)$ 等は given とする．)

$$\begin{cases} g_0 \oplus g_4 \oplus g_6 \oplus g_8 = f(0,0) \\ g_1 \oplus g_4 \oplus g_7 \oplus g_8 = f(0,1) \\ g_2 \oplus g_5 \oplus g_6 \oplus g_8 = f(1,0) \\ g_3 \oplus g_5 \oplus g_7 \oplus g_8 = f(1,1) \end{cases} \quad (15)$$

まず，この連立方程式のうちの 1 つに着目する．ここでは，

$$g_0 \oplus g_4 \oplus g_6 \oplus g_8 = f(0,0) \quad (16)$$

に着目することにする．この方程式の左辺の積項数は $4(=N=2^n)$ である．これを N-ESOP と呼ぶことにする．この 1 つの N-ESOP を $N-1$ 個の 3-ESOP に変換することを考える．

この 1 つの N-ESOP は，次のように $N-1$ 個の 3-ESOP に変換できる．(図 5 参照)．

$$\begin{cases} \overline{f(0,0)} \oplus y_{01} \oplus y_{02} = 1 \\ \overline{y_{01}} \oplus g_0 \oplus g_4 = 1 \\ \overline{y_{02}} \oplus g_6 \oplus g_8 = 1 \end{cases} \tag{17}$$

ここで，$1 \oplus x = \bar{x}$ を利用した．

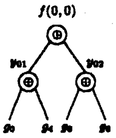

図 5: 式 (16) の構文木

同様な方法で，式 (15) は，すべて，3-ESOP による連立方程式に変換可能である．ただし，$f(0,0)$ 等の値は，given であるとしているので，$f(0,0)$ 等を含む式に関しては，2-ESOP であるといえる．

以上で述べた方法によって，一般に，1 つの N-ESOP を $N-1$ 個の 3-ESOP に変換するには，構文木の内部節点の数に比例するステップで変換でき，要する時間計算量は $O(N)$ ステップである．

【命題】 6.1
N-ESOP は，$N-1$ 個の 3-ESOP に変換できる．

6.2 充足列の定義

次に，1 つの 3-ESOP に着目する．例えば，式 (17) における

$$\overline{y_{01}} \oplus g_0 \oplus g_4 = 1 \tag{18}$$

に着目することにする．

ここで，ある 3-ESOP を満たす割り当てに対し，**充足列**を定義する．充足列は，記号列 $a'_1a'_2\cdots a'_m (a'_i \in \{0,1,X\})$ で表される $(1 \le i \le m)$:

$$\begin{cases} a'_i = 0(\text{or } 1), & \text{if } a_i \text{ に } 0(\text{または}, 1) \text{ が割り当てられる}. \\ a'_i = X, & \text{if } a_i \text{ はその 3-ESOP では，特定されない}. \end{cases} \tag{19}$$

ただし，$A=\{a_1,a_2,\cdots,a_m\}$ は，着目している N-ESOP に対する構文木の根を除く節点のラベルに対応している．例えば，式 (16) では，$A=\{y_{01},y_{02},g_0,g_1,\ldots,g_8\}$ である．そして，式 (18) に関する充足列は，

y_{01}	y_{02}	g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
1	X	0	X	X	X	1	X	X	X	X
1	X	1	X	X	X	0	X	X	X	X
0	X	0	X	X	X	0	X	X	X	X
0	X	1	X	X	X	1	X	X	X	X

(20)

となる．これらは，それぞれ，式 (18) の充足割り当て $(y_{01},g_0,g_4)=(1,0,1),(1,1,0),(0,0,0),(0,1,1)$ に対応している．すなわち，充足列は，充足割り当てに冗長性を持たせたものである．1 つの 3-ESOP に対しては，この充足列は，4 つのみであることに注意されたい．(2-ESOP に関しては，充足列は 2 つのみである)．そして，前節で 3-ESOP にした方程式すべてに対して，その充足列が求められる．

また，ある 2 つの充足列について，その割り当てに矛盾が生じない場合その 2 つの充足列は**両立可能**であると呼ぶことにする．そうでない場合，その 2 つの充足列は，**両立不可能**であると呼ぶことにする．すなわち，充足列 $a'_1a'_2\cdots a'_m$ と $a''_1a''_2\cdots a''_m$ に対し，すべての l, $(1 \le l \le m)$ について，$a'_l \ne X$ かつ $a''_l \ne X$ ならば，$a'_l = a''_l$ が成り立つとき，その充足列 $a'_1a'_2\cdots a'_m$ と $a''_1a''_2\cdots a''_m$ は，両立可能である．1 つの 3-ESOP から派生した充足列は，常に互いに両立不可能であることに注意されたい．

1 つの 3-ESOP から 4 つの充足列を生成するための時間計算量は，$O(1)$ ステップである．

6.3 CLIQUE への還元

次に，前節で求めた充足列に対して，その充足列をノードラベルとするようなグラフを描く．そして，そのグラフに対し，辺は，そのラベルに関して，両立可能なノード間に張られるとする．(図 6 参照)

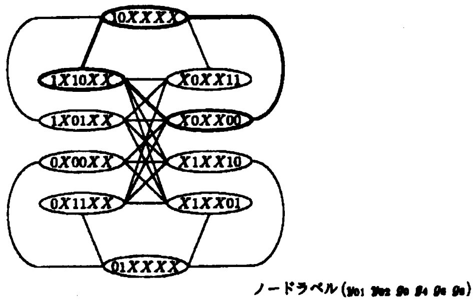

図 6: 式 (17) に対応するグラフ (ただし，$f(0,0)=1$ とする)

ここで，連立している 3-ESOP 式の総数を M とする．そのようなグラフに対して，M-クリークが存在すると，その連立 3-ESOP 方程式を充足する解が存在することになる．例えば，図 6 については，式 (17) は，3 つの式からなるので，図 6 において，3-クリークを見つければ，そのノードラベルから解がわかる．図 6 に 3-クリークのうちの 1 つを示したが，これは，式 (17) を満たす解のうちの 1 つが，$(y_{01},y_{02},g_0,g_4,g_6,g_8)=(1,0,1,0,0,0)$ であることを示している．

前節までの議論によって，Helliwell 方程式から導かれる 3-ESOP と 2-ESOP の総数 M は，$N(N-1)$ 個である．そして，その充足列から CLIQUE 問題のインスタンスとしてのグラフの節点数は，$2N(2N-3)$ となる．したがって，グラフを構成するための時間計算量は，$O((2N(2N-3))^2)=O(N^4)$ ステッ

プである．本研究では，n 変数論理関数として真理値表が入力されるとしているので，そのサイズは $O(2^n) = O(N)$ であり，この還元は，入力サイズに関して多項式である．

【命題】 6.2
$M(= N(N-1))$ を連立方程式に含まれる式の数とする．上で述べた方法で構成したグラフにおいて，M-クリークが存在することと，その連立方程式に，解が存在することは等価である．

6.4 重みつきクリーク問題の利用

ここまでに述べた方法で，Helliwell 方程式の解法を CLIQUE 問題に還元できることがわかった．しかし，Helliwell 方程式のみでは，ESOP に含まれる積項数が，考慮されないことに注意しなければならない．そこで，重み付きクリーク抽出問題を考えることにより，最小 ESOP を求める方法について考える．

まず，ノードに重みを持つグラフとその節点重みが最小な最大クリーク抽出問題について考える．節点重みが最小な最大クリーク抽出問題とは，最大クリークのうちで，節点の重みの総和が，最小となるようなクリークを発見する問題である．そして，これが節点重みが最大な最大クリーク抽出問題 (以降，節点重み最大クリーク抽出問題とする) に還元できることを示す．節点重み最大クリーク抽出問題に対する高速なアルゴリズムについては，文献 [6] で提案されている．

積項数を考慮するための節点の重みは，次のように与える．

- 充足列の g_i に相当する記号が 1 で，その積項 g_i が被覆する最小項の数が l ならば，与える重みは，N/l とする．ここで，$N(= 2^n)$ は，最小項の総数である．
- g_i に相当する記号が 0 か X のときは，重みに貢献しない．また，構文木の内部節点から生成された変数 y_i は，重みに貢献しない．

この重み付けは，Helliwell 方程式からわかるように，その連立方程式に g_i が，それが被覆する最小項の数だけ現れることによる．つまり，各 g_i に関して，その連立方程式 (すなわち，1 つのクリーク) において，出現回数 × 重み が一定となる．

【命題】 6.3
上で述べた方法で得られた節点重みつきグラフにおいて，節点重みが最小な最大クリークを求めることにより，最小 ESOP が得られる．

また，充足列において，1 の数の代わりに，0 の数について考えると，ESOP の最小化問題は，節点重み最大クリーク抽出問題に還元できる．

系 6.1
直前で述べた方法で，得られた節点重みつきグラフにおいて，節点重み最大クリークを求めることにより，最小 ESOP が得られる．

1 つの節点に重みを与えるための時間計算量については，ノードラベルである充足列の長さに比例し，$O(2^n) = O(N)$ ステップである．節点の総数は，前節で述べたように $2N(2N-3)$ であり，グラフすべての節点に重みを与えるのに，$O(N^3)$ ステップかかる．つまり，重みの付与も入力サイズに関して多項式時間の計算量で求めることができる．

7 実働化とその実行時間

ここまでに述べた還元アルゴリズムを C 言語で実働化した．そして，文献 [6] に基づくアルゴリズムを用いて，実際に論理関数の最小 ESOP を求めた結果を表 1, 表 2 に示す．

計算機環境は，以下のとおりである．

- CPU
  Pentium4　2.8GHz
- OS
  Linux
- コンパイラ，およびコンパイルオプション
  gcc -O2

実際のプログラムの実行に要した時間を表 1, 表 2 に示す．ここで用いた節点重み最大クリーク抽出アルゴリズム VWMC 等については，文献 [6] において提案されたものである．詳細は省略するが，簡単に述べておく:

VWMC : 最大クリークを求めながら，節点重み最大クリークを求めるアルゴリズム．

VWMC$_\omega$: 最大クリークのサイズ ω をあらかじめ与え，節点重み最大クリークを求めていくもの．

VWMC$_\omega$+ : VWMC$_\omega$ にクリーク重みの上界による分枝限定を加えたもの．

表中で用いている論理関数の表記法については，以下のとおりである．論理関数 $f(x_1, x_2, \ldots, x_n)$ は，その論理関数の最小項の論理和に対して係数 $m_{2^n-1}m_{2^n-2}\cdots m_1 m_0 (m_i \in \{0,1\})$ を与えることで表現するできる:

$$
\begin{aligned}
f(x_1, x_2, \ldots, x_n) \;=\; & m_0\overline{x_1}\,\overline{x_2}\cdots\overline{x_n} \\
\vee\; & m_1\overline{x_1}\,\overline{x_2}\cdots\overline{x_{n-1}}x_n \\
\vee\; & m_2\overline{x_1}\,\overline{x_2}\cdots\overline{x_{n-1}}x_n \\
\vee\; & \cdots \\
\vee\; & m_{2^n-2}x_1x_2\cdots x_{n-1}\overline{x_n} \\
\vee\; & m_{2^n-1}x_1x_2\cdots x_{n-1}x_n \qquad (21)
\end{aligned}
$$

そして，その係数列 $m_{2^n-1}m_{2^n-2}\cdots m_1 m_0$ を 2^n ビットの 2 進数とみる．さらに，それを $2^n/4$ 桁の 16 進数としたもので論理関数を表現している．

表 1: 2 変数論理関数の場合 (節点数: 40　枝密度: 0.80000)

論理関数	最小 ESOP の積項数	還元に要した時間 [sec]	VWMC[sec]	VWMC$_\omega$[sec]	VWMC$_\omega$+[sec]
0	0	0.00000	0.00010	0.00010	0.00010
1	1	0.00000	0.00010	0.00013	0.00011
6	2	0.00000	0.00011	0.00013	0.00012
F	1	0.00000	0.00010	0.00013	0.00009

表 2: 3 変数論理関数の場合 (節点数: 208　枝密度: 0.946860)

論理関数	最小 ESOP の積項数	還元に要した時間 [sec]	VWMC[sec]	VWMC$_\omega$[sec]	VWMC$_\omega$+[sec]
00	0	0.00000	7.52	7.54	0.04
01	1	0.00000	7.55	7.59	0.10
09	2	0.00000	7.54	7.65	0.09
49	3	0.00000	7.50	7.55	0.05
6C	2	0.00000	7.58	7.54	0.05
FF	1	0.00000	7.61	7.59	0.03

8 考察

1. 3 変数までの規模では, 還元に要する計算時間は無視できる程小さいことがわかった. また, 還元だけに関すれば, 4 変数の場合: 0.00000 秒, 5 変数の場合: 0.05000 秒, 6 変数の場合: 1.00000 秒で計算ができることもわかった.

2. 3 変数において VWMC$_\omega$+ の結果が他に比べて非常に高速であることがわかる.

3. 4 変数以上の論理関数については, 計算時間が 24 時間以上かかることがわかった. これは, 枝密度が高く (4 変数では, 0.984172), 節点数が 928(4 変数の場合) というグラフが生成され, 最大クリークが多数存在することによると考えられる.

   このグラフにおいて重みを無視して, 最大クリークを 1 つ抽出する場合, 枝密度が高い場合に効率化されたアルゴリズム MCQd+[4] による実行結果は, 15 秒以下であり, この基本アルゴリズムの考慮により, 改善が期待される.

4. 少なくとも 3 変数までに限れば, 論理関数の種類による実行時間の差は, ほとんどないことがわかる.

参考文献

[1] Jae-Seung Lee, Yongwook Chung, Jaehyun Kim, and Soonchil Lee: "A Practical Method of Constructing Quantum Combinational Logic Circuits", 1999. LANL quant-ph/9911053

[2] T. Sasao: "An Exact Minimization of AND-EXOR Expressions Using Reduced Covering Functions", Proc. of the Synthesis and Simulation Meeting and International Interchange, October 20-22, 1993, pp. 374-383.

[3] M.R.Garey and D.S.Johnson: "COMPUTERS AND INTRACTABILITY A Guide to the Theory of NP-Completeness", Freeman, San Francisco, 1979.

[4] 関友和, 富田悦次: 「分枝限定法を用いた最大クリーク抽出アルゴリズムの効率化」, 電子情報通信学会コンピュテーション研究会, COMP2001-50, pp.101-108.

[5] 名久井行秀, 西野哲朗: 「量子論理回路の深さ最小化について」, 第 6 回量子技術研究会資料 QIT2002-22, pp.113-118, 2002.

[6] 中村 知倫, 富田　悦次, 西野　哲朗, 名久井行秀: 「節点重み最大クリーク抽出アルゴリズム」, 2002 年度冬のＬＡシンポジウム予稿, pp. 8.1-8.6.