プログラミング-言語・基礎・実践- 11－16
(1993. 3. 11)

二分決定グラフの最適変数順序付け問題の計算複雑さ

谷 誠一郎　　濱口 清治　　矢島 脩三
京都大学工学部情報工学教室
〒606-01 京都市左京区吉田本町

あらまし

論理関数の効率的表現方法の一つである二分決定グラフの節点数は、変数の順序付けに大きな影響を受ける。本稿では、共有二分決定グラフ最適変数順序付け問題、すなわち節点数 n の共有二分決定グラフ SB と自然数 $k\,(<n)$ が与えられたとき SB と同じ論理関数の集合を表す節点数 k 以下の共有二分決定グラフを実現するような変数の順序付けが存在するかという問題について、これが NP完全であることの証明を行う。

和文キーワード　二分決定グラフ、ブール関数、変数の順序付け、NP完全

The Complexity of the Variable Ordering Problems of Binary Decision Diagrams

Seiichiro TANI　　Kiyoharu HAMAGUCHI　　Shuzo YAJIMA
Department of Information Science, Faculty of Engineering, Kyoto University
Yoshida-honmachi Sakyou-ku Kyoto, 606-01

Abstract

A binary decision diagram (BDD) is used as an efficient way of the representation of a Boolean function. The size of BDD's is highly sensitive to the variable ordering. In this paper, it is proved that the optimal variable ordering problem of shared binary decision diagrams (SBDD's) (OVO) is NP-complete. The problem is to decide whether, for a given SBDD with n nodes and a positive integer $k\,(<n)$, there exists a variable ordering for some graph whose nodes are less than or equal to k and which represents the same Boolean functions as are represented by the given SBDD.

英文 key words　binary decision diagram, Boolean function, variable ordering, NP-complete

1 緒論

近年の集積回路技術の発展にともない設計対象となる回路は大規模化、複雑化し、このため計算機による設計支援が不可欠になっている。集積回路の設計支援システムでは論理関数の操作を行う必要があるが、演算操作に必要な時間や領域は論理関数の表現方法に大きく依存する。このため、論理関数を表現するための種々の方法が提案されている。 Akers や Bryant によって提案された二分決定グラフ [1, 3] を用いた論理関数処理はその効率の良さから、近年、広く用いられるようになっている。

二分決定グラフは非巡回有向グラフによる論理関数の表現であり、実用的な論理関数の多くは比較的少ない記憶容量で表現できることが知られている。また、二分決定グラフは変数の順序を固定することにより一意に論理関数を表すことができ、節点数に対して多項式時間で論理演算可能である。このため論理関数処理や論理関数の等価性判定などを効率良く行うことができ、計算機上における論理関数の表現として有効なものと考えられている。

一つのグラフで一つの論理関数を表す二分決定グラフに対して、複数の論理関数を同時に一つのグラフで表現するものは共有二分決定グラフと呼ばれている [8]。共有二分決定グラフは二分決定グラフの拡張であり、複数の根を持つ非巡回有向グラフである。また、最近では属性枝の導入 [8] などの改良手法が提案されており、論理設計検証、論理照合、テスト生成、論理合成等さまざまな分野において共有二分決定グラフが利用されつつある。

二分決定グラフや共有二分決定グラフを計算機上で扱う場合、演算操作に必要な時間と領域はその節点数に大きく依存するため、より少ない数の節点によって論理関数を表現することが重要である。一般に二分決定グラフは変数の順序付けを変えると同じ論理関数を表すものでも異なるグラフとなり、節点数が変化する。したがって節点数が少なくなるような変数の順序付けを見つけることは重要な問題である。節点数が最小になるような変数の順序付けを見つけるアルゴリズムとしては、真理値表を入力とするもの [5] 、二分決定グラフを入力とするもの [10] 等が提案されている、いずれのアルゴリズムも節点数に対し指数時間の計算量を持つものであり、変数の数が多い場合は、動的重み付け法、部分決定グラフによる方法等の近似解法 [4, 8, 9] が用いられている。節点数が最小になるような変数の順序付けを見つける問題の難しさに対する理論的研究は、論理式を入力として節点数を最小にするような変数の順序付けを求める問題は co-NP完全があることが Bryant[3] によって示されている他は、これまでほとんどなされていない。

本稿では、与えられた二分決定グラフに対し、節点数が最小になるような変数の順序付けを見つける方法 [10] に関連して、共有二分決定グラフ最適変数順序付け問題、すなわち節点数 n の共有二分決定グラフ SB と自然数 $k\,(<n)$ が与えられたとき SB と同じ論理関数の集合を表すノード数 k 以下の共有二分決定グラフを実現するような変数の順序付けが存在するかという問題について、まずこれが (1)NPに属していることを示し、次にこれが (2)NP困難であることを示すことにより、NP完全であることの証明を行う。(1) では、共有二分決定グラフの最適変数順序付け問題を解く非決定性多項式時間アルゴリズムを示し、(2) では、NP完全性が証明された最適線形配置問題 [7] において与えられる、グラフ G のコストを反映する共有二分決定グラフを構成し、最適線形配置問題を共有二分決定グラフの最適変数順序付け問題に多項式帰着可能であることを示す。

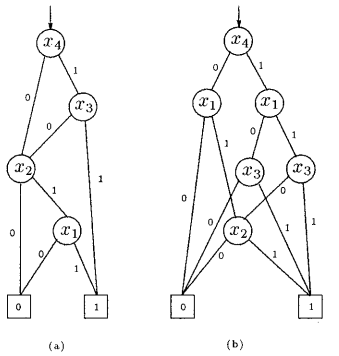


図 1: 論理関数 $f = x_1x_2 + x_3x_4$ を表す二分決定グラフ

2 準備

2.1 二分決定グラフ

二分決定グラフ (**B**inary **D**ecision **D**iagram, **BDD**) は論理関数を表す非巡回有向グラフである [3] 。節点の集合は変数節点と定数節点の二つの集合に分けることができる。

各変数節点はその節点の表す変数と、0 枝、1 枝と呼ばれる二本の有向枝を持つ。v の表す変数を $index(v)$ で示し、0 枝の指す節点を $low(v)$ 、1 枝の指す節点を $high(v)$ で示すことにする。一方、定数節点は 0 または 1 を値として持つ節点で $index(v)$, $low(v)$, $high(v)$ は定義されない。

各節点にはその節点が表す論理関数が定義される。論理関数 f を表す二分決定グラフの節点 v が表す論理関数を $F(v)$ とし、また根となる節点を $root$ で表すことにすると定義は以下のようになる。

$$F(root) = f$$
$$F(low(v)) = F(v)|_{index(v)=0}$$
$$F(high(v)) = F(v)|_{index(v)=1}$$

ただし、$F(v)|_{x_1=b_1,x_2=b_2,\cdots,x_m=b_m}$ 、及び $F(v)|_{\bigwedge_{i=1}^{m} x_i=b_i}$ は、$F(v)$ の変数 x_i $(i = 1,2,\cdots,m)$ に b_i $(b_i \in \{0,1\},\ i = 1,2,\cdots,m)$ を代入して得られる論理関数を表す。

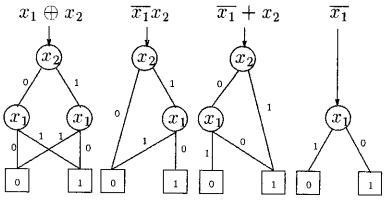
また、次のような変換が施された二分決定グラフを既約な二分決定グラフという [3] 。

1. $low(v) = high(v)$ であるような節点 v が存在する場合、v を削除し v を指していたすべての枝が $low(v)\,(= high(v))$ を指すように変更する。
2. 等価な節点 u, v が存在する場合、v を削除し v を指していたすべての枝が u を指すように変更する。ただし、等価な節点とは次のような場合をいう。
   - 定数節点の場合、u, v の値が等しい。
   - 変数節点の場合、$index(u) = index(v)$ かつ $low(u) = low(v)$ かつ $high(u) = high(v)$

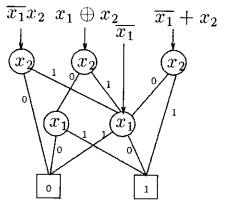
以下本報告書では既約な二分決定グラフを単に二分決定グラフ (BDD) と呼ぶことにする。図 1に論理関数 $f = x_1x_2 + x_3x_4$ を表す二分決定グラフを示す。ただし、本報告書では論理式を以下のように表記するものとする。

論理変数 $x, y, z_i\,(i = 1,2,\cdots,n)$ に対して、

1. 論理和 $\cdots x, y$ の論理和は $x + y$ 。また多変数の場合、$\sum$ を用いて例えば $\sum_{i=1}^{n} z_i$ とも表記する。
2. 論理積 $\cdots x, y$ の論理積は $x \cdot y$ 。また、しばしば xy も同じ意味で用いる。多変数の場合、$\prod$ を用いて例えば $\prod_{i=1}^{n} z_i$ とも表記する。



(a) 二分決定グラフ



(b) 共有二分決定グラフ

図 2: 二分決定グラフと共有二分決定グラフ

3. 否定 $\cdots x$ の否定は $\bar{x}$ 。
4. 排他的論理和 $\cdots x, y$ の排他的論理和は $x \oplus y$ 。多変数の場合、$\oplus$ を用いて例えば $\bigoplus_{i=1}^{n} z_i$ とも表記する。
5. 恒真関数は 1 、恒偽関数は 0 で表すものとする。

2.2 共有二分決定グラフ

複数の二分決定グラフの間で 2.1節で示した変換を施したものを共有二分決定グラフ (**S**hared **B**inary **D**ecision **D**iagram, **SBDD**) と呼ぶ [8] 。論理関数の集合を表現するのに共有二分決定グラフを用いることで必要な記憶容量を減らすことができるばかりでなく、等しい論理関数は同一節点によって表されるので論理関数の等価性判定が容易となる。このため、実際に計算機上では二分決定グラフよりも共有二分決定グラフが用いられている。図 2(a) の四つの二分決定グラフを共有二分決定グラフで表したものが図 2(b) である

2.3 計算複雑さ

言語のクラス NP とは非決定性チューリング機械によって多項式時間で受理される言語のクラスをいい、クラス NP に属するすべての言語 L から多項式帰着可能であるような言語 L_0 を NP**困難**であるという [2] 。ただし言語 L が言語 L_0 に多項式帰着可能とは次の条件を満たす決定性の多項式時間限定チューリング機械 M が存在することである。

条件 1 : M は L 上の任意の文字列 w を L_0 上の任意の文字列 w_0 に変換し、かつ w が L に属するための必要十分条件は w_0 が L_0 に属することである。 □

また、言語 L_0 が次の条件を満たすとき L_0 は NP**完全**であるという [2] 。

条件 2 : もし L_0 を認識する時間計算量 $T(n) \geq n$ の決定性アルゴリズムがあるとすれば、NP に属する任意の言語 L に対して、それを認識する時間計算量 $T(p_L(n))$ の決定性アルゴリズムが存在する。ただし、$p_L(n)$ は L に依存する多項式を表す。 □

NP に属する言語で NP**困難**である言語は NP**完全**であることが知られている [2] 。また、NP**困難**であることの証明は、一般に、すでに証明されている NP**完全**問題から多項式帰着することで行なわれる。本報告書でもこの手法を用いる。

3 共有二分決定グラフの最適変数順序付け問題とその計算複雑さ

3.1 共有二分決定グラフの最適変数順序付け問題

共有二分決定グラフのサイズと変数の順序付けに関して、次の問題を考える。

[SBDD の最適変数順序付け問題]
(the **O**ptimal **V**ariable **O**rdering problem of SBDD's, **OVO**)
節点数 n の既約な共有二分決定グラフ SB と自然数 $k(< n)$ が与えられたとき、SB と同じ論理関数の集合を表す節点数が k 以下の共有二分決定グラフを実現するような変数の順序付けが存在するか。

3.2 共有二分決定グラフの最適変数順序付け問題の計算複雑さ

定義 1 (変数の順序付け π) 変数の数を m としたとき、変数の順序付けを $\pi = (\pi[1], \pi[2], \cdots, \pi[m])$ と記す。ただし、$i \neq j$ なら $\pi[i] \neq \pi[j]$, $\pi[i] \in \{x_1, x_2, \cdots, x_m\}$ 、また $index(v) = \pi[i]$, $index(low(v)) = \pi[j]$ ならば $i > j$ とする ($high(v)$ についても同様)。 □

定理 1 (OVO$\in NP$) **OVO** は NP に属する。

証明
図 3に、**OVO** を解く非決定性多項式時間アルゴリズム MAKESBDD を示す。このアルゴリズムは以下のデータ構造と関数を仮定している。

- 与えられた共有二分決定グラフの表す論理関数を f_i $(i = 1, 2, \cdots p)$ とする。
- $bddtable$ はすでに生成した節点とその節点が表す論理関数の対の集合とする。また、$(v, f_i) \in bddtable$ であるとき f_i は $bddtable$ に登録されているという。ただし、節点 v は論理関数 f_i を表す節点である。
- $F(v)$ は $bddtable$ を探索して節点 v の表す論理関数を返す関数とする。
- $var(F, \pi)$ は論理関数 F が依存する変数で i が最大となる変数 $\pi[i]$ を表す。

MAKESBDD は、節点数 n の共有二分決定グラフ SB と自然数 $k(< n)$ を入力とし、受理の場合は'yes' 、非受理の場合は'no' を出力する。MAKESBDD は、変数の順序付け π を非決定的に選択し、これにしたがって SB と同じ論理関数の集合を表す共有二分決定グラフを、手続き $bddcreate$ を呼び出すことにより、生成する。手続き $bddcreate$ は再帰的に二分決定グラフを生成するが、生成途中に節点数が k を越える場合は、その時点で直ちに生成を中止し、非受理とするものである。従って、MAKESBDD を実行することによって生成される節点数は高々 k 個である。図 4に手続き $bddcreate$ を示す。ただし、4,16 行目の $\exists u\ (u, tmp) \in bddtable$ は、$bddtable$ 中に tmp が登録されている場合は真でかつ tmp を表す節点を u に代入し、それ以外は偽となる条件式を表す。

手続き $bddcreate$ の呼びだし回数は変数節点数に等しいがこれは高々 k 個しか生成されないので、$bddcreate$ の呼びだし回数も高々 k 回である。また、$bddtable$ に登録されているすべての論理関数を、二分決定グラフによって表現すると、tmp の計算 (3 行目) 及び $bddcreate$ 中の (u, tmp) の探索 (4,16 行目) は、n に対する多項式時間で計算可能である [3] 。よって、決定性部分全体の計算量は $O(n^{O(1)})$ である。また、$bddtable$ には高々 k $(\leq n)$ 個の要素が登録されるだけであり、要素一つ当たり $O(n)$ の領域さえあればいいので、$bddtable$ のために必要な領域は $O(n^2)$ である。 *Q.E.D.*

アルゴリズム
入力：節点数 n の共有二分決定グラフ SB、自然数 k $(< n)$ 。
出力：受理のとき"yes" 、非受理のとき"no" 。
方法：以下の手続き MAKESBDD による。

```
1 procedure MAKESBDD(SB,k)
2 begin
3    非決定的に変数の順序 π を選択する;
4    bddtable := φ;
5    while(SB の表す論理関数で
           bddtable に登録されていない f に対して)do
6       begin
7          f を表す節点を生成する;
8          v := 今までに登録した節点数 +1;
9          bddtable = bddtable∪{ (v, f)};
10         call bddcreate(v);
11      end;
12   if (生成した節点数 ≤ k)
13      then 受理
14      else 非受理;
15 end.
```

図 3: OVO を解く非決定性多項式時間アルゴリズム

定義 2 (二分決定グラフの最適変数順序付け問題) 次に示す問題を二分決定グラフの最適変数順序付け問題という。
「節点数 n の既約な二分決定グラフ B と自然数 $k(< n)$ が与えられたとき、B と同じ論理関数を表す節点数 k 以下の二分決定グラフを実現するような変数の順序付けが存在するか。」 □

系 1 二分決定グラフの最適変数順序付け問題は NP に属する。

証明
OVO において、一つの論理関数を表す SBDD を考えれば良い。 *Q.E.D.*

以下、共有二分決定グラフの最適変数順序付け問題が NP困難であることを証明するための準備として、諸定義を行う。次に定義する最適線形配置問題は文献 [6] による。

定義 3 [最適線形配置問題]
(the **O**ptimal **L**inear **A**rrangement problem, **OLA**)
OLA とは次の問題をいう *[6, 7]*。
「グラフ $G = (V, E)$ と正整数 K が与えられた時、$\Sigma_{\{u,v\}\in E}|\psi(u) - \psi(v)| \leq K$ を満たす 1 対 1 写像 $\psi : V \longrightarrow \{1, 2, 3, \cdots, |V|\}$ が存在するか。」
また、$|\psi(u) - \psi(v)|$ を枝 (u, v) のコストといい、

$$\Sigma_{(u,v)\in E}|\psi(u) - \psi(v)|$$

をグラフ G のコストという。 □

命題 1 (OLA の NP完全性) **OLA** は NP完全である *[7]*。

定義 4 (BDD のサイズ) 論理関数 f の表す BDD で変数の順序付けが π であるものの節点数を $size(f, \pi)$ で表す。ただし、混乱のない場合は $size(f)$ と略記する。 □

```
1 procedure bddcreate(v)
2 begin
3    tmp := F(v)|var(F(v),π)=0;
4    if (∃u (u, tmp) ∈ bddtable)
5       then low(v) := u ;
6       else if (今までつくった節点数 ≥ k)
7          then 非受理
8          else begin
9             tmp を表す節点 v' を新しくつくる;
10            bddtable := bddtable ∪ {(v' , tmp)};
11            low(v) := v';
12            if (tmp ≠ 0 and tmp ≠ 1)
13               then bddcreate(low(v));
14            end;
15   tmp := F(v)|var(F(v),π)=1;
16   if (∃u (u, tmp) ∈ bddtable)
17      then high(v) := u ;
18      else if (今までつくった節点数 ≥ k)
19         then 非受理
20         else begin
21            tmp を表す節点 v' を新しくつくる;
22            bddtable := bddtable ∪ {(v' , tmp)};
23            high(v) := v';
24            if (tmp ≠ 0 and tmp ≠ 1)
25               then bddcreate(high(v));
26            end;
27 end.
```

図 4: 再帰的に二分決定グラフを生成する手続き *bddcreate*

定義 5 (T- ファージ関数)
$f(x_1, x_2, \cdots, x_n) = (x_{i_1} \oplus x_{i_2}) \prod_{k\neq i_1, i_2} x_k$ を n 変数 T- ファージ関数という。ただし、$(i_1, i_2, \cdots, i_n)$ は $(1, 2, \cdots, n)$ の置換を表す。 □

T- ファージ関数の例を図 5に示す。ただし 0 を値として持つ定数節点への枝は省略した。以下の図でも同様に省略する。

補題 1 (T- ファージ関数の性質)
n 変数 T- ファージ関数 $f = (x_{i_1} \oplus x_{i_2}) \prod_{k\neq i_1, i_2} x_k$ の順序付け π による BDD のサイズ $size(f, \pi)$ は、$\pi[l] = x_{i_1}$, $\pi[m] = x_{i_2}$ ならば $|l - m| + n + 2$ である。

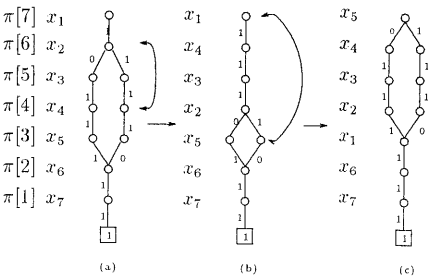
証明
一般性を失うことなく、$\pi[l] = x_{i_1}$, $\pi[m] = x_{i_2}$, $l > m$ と仮定することが出来る。n 変数論理関数 f を表す BDD の変数の順序付け π に対するサイズ $size(f, \pi)$ は、

$$\mathcal{F}_p = \left\{ f|_{\bigwedge_{t=n-p+1}^{n} \pi[t]=b_t} \;\middle|\; \begin{array}{l} b_t \in \{0, 1\} \\ t = n-p+1, n-p+2, \cdots, n \end{array} \right\}$$

に対し、

$$size(f, \pi) = |\bigcup_{p=1}^{n} \mathcal{F}_p|$$

が成り立つ



$$f = (x_2 \oplus x_5)x_1x_3x_4x_6x_7$$

図 5: T- ファージ関数の性質

また、$n-i+1 > l$ に対し、

$$\mathcal{F}_i = \{0,\ (\pi[l] \oplus \pi[m]) \prod_{\substack{t=1 \\ t\neq l,m}}^{n-i} \pi[t]\}$$

$m < n-i+1 \leq l$ に対し、

$$\mathcal{F}_i = \{0,\ \prod_{t=1}^{n-i} \pi[t],\ \overline{\pi[m]} \prod_{\substack{t=1 \\ t\neq m}}^{n-i} \pi[t]\}$$

$1 < n-i+1 \leq m$ に対し、

$$\mathcal{F}_i = \{0,\ \prod_{t=1}^{n-i} \pi[t]\}$$

$n-i+1 = 1$ に対し、

$$\mathcal{F}_i = \{0,1\}$$

が成り立つ。よって、

$$|\bigcup_{p=1}^{n} \mathcal{F}_p| = |l-m|+n+2$$

Q.E.D.

定義 6 (TSB) 次のような関数の集合を表現する *SBDD* を **TSB** と呼ぶ。ただし、$(i_1, i_2, \cdots, i_n)$ は $(1,2,\cdots,n)$ の置換を表し、また $j = 1,2,\cdots,m$ である。

$$\begin{cases} f_j = (x_{i_1 j} \oplus x_{i_2 j})x_{i_3 j} \cdots x_{i_n j} \\ g_j = (y_{i_1 j} \oplus y_{i_2 j})y_{i_3 j} \cdots y_{i_n j} \\ h_i = \bigoplus_{j=1}^{m}(x_{ij} \oplus y_{ij}) \prod_{j,k\neq i} \overline{x_{kj}} \cdot \overline{y_{kj}} \end{cases} \quad (1)$$

□

定義 7 (変数ブロック) 各 i $(i = 1,2,\cdots n)$ に対して、変数の集合 $\{x_{ij}, y_{ij} | 1 \leq j \leq m\}$ をブロック i と呼び、B_i で表す。また、$B = \{B_1, B_2, \cdots, B_n\}$ とする。 □

定義 8 (well-ordered,wo) 各 i $(i = 1,2,\cdots n)$ に対して、変数の集合 $\{x_{ij}, y_{ij} | 1 \leq j \leq m\}$ が連続している順序付けによって構成されている *SBDD* を *well-ordered* であるという。また、*well-ordered* でない *SBDD* を *non-well-ordered* であるという。*well-ordered*、*non-well-ordered* はそれぞれ *wo*、*nwo* とも記す。 □

定義 9 (変数ブロックの順序付け φ) *wo-SBDD* に対して変数ブロックの順序付けを

$$\varphi = (\varphi[1], \varphi[2], \cdots, \varphi[n])$$

とする。ただし、$\varphi[l] \in \{B_1, B_2, \cdots, B_n\}$、$l_1 \neq l_2$ ならば $\varphi[l_1] \neq \varphi[l_2]$、また $\pi[i_1] \in \varphi[l_1]$、$\pi[i_2] \in \varphi[l_2]$ なる任意の $\pi[i_1]$、$\pi[i_2]$ に対して、$i_1 > i_2$ なら $l_1 > l_2$ である。 □

補題 2 (OLA の枝と T- ファージ関数) OLA において与えられるグラフ $G = (V, E)$ の節点に i $(i = 1,2,\cdots,|V|)$、枝に j $(j = 1,2,\cdots,|E|)$ と名前をつける。次に節点 i_1, i_2 を両端に持つ枝 j に対して、次のように T- ファージ関数を二つずつを対応させる。

$$f_j = (x_{i_1 j} \oplus x_{i_2 j}) \prod_{k\neq i_1, i_2} x_{kj}$$

$$g_j = (y_{i_1 j} \oplus y_{i_2 j}) \prod_{k\neq i_1, i_2} y_{kj}$$

このとき $i \in \{1,2,\cdots,|V|\}$, $l \in \{1,2,\cdots,|V|\}$ を満たす i, l に対し 1 対 1 写像 $F : B \longrightarrow \{\varphi_1, \varphi_2, \cdots, \varphi_{|V|}\}$ を

$$\psi : i \mapsto l \Longleftrightarrow F : B_i \mapsto \varphi[l]$$

を満たすように定めれば、次が成り立つ。

$$\textbf{OLA の枝 } j \textbf{ のコスト} = \frac{1}{2}(size(f_j)+size(g_j))-|V|-2 \quad (2)$$

証明

OLA において節点 i_1, i_2 が 1 対 1 写像 f によってそれぞれ l_1, l_2 に移されたとすると枝 j のコストは $|l_1 - l_2|$ である。一方、これを TSB について考えてみると、$x_{i_1 j} \in B_{i_1}, x_{i_2 j} \in B_{i_2}$ であり、また B_{i_1}, B_{i_2} は F によってそれぞれ $\varphi[l_1], \varphi[l_2]$ に移される。ここで f_j が依存する変数は各変数ブロック内に一つであること、及び補題 1 から、

$$size(f_j) = |l_1 - l_2| + |V| + 2 \quad (3)$$

が成り立つ。g_j についても同様である。よって式 (2) が成り立つ。 *Q.E.D.*

定義 10 (wo-TSB のタイプ) *wo-TSB* において、$\pi[1]$ が $x_{i_1 j}$, $x_{i_2 j}$, $y_{i_1 j}$, $y_{i_2 j}$ のいずれでもないときその *wo-TSB* をタイプ 1 といい、それ以外のものをタイプ 2 という。$x_{i_1 j}$, $x_{i_2 j}$, $y_{i_1 j}$, $y_{i_2 j}$ は各 j $(j = 1,\cdots,m)$ に対し式 *(1)* を満たす。 □

補題 3 (変数ブロック内での変数の順序入れ換え)
TSB が *well-ordered* であるとき、$\varphi[1]$ の変数ブロック内の変数順序の変化による節点数の変化は高々 1 である。また、$\varphi[1]$ 以外の変数ブロック内では変数順序を変えても、節点数は変化しない。

証明

TSB において各 f_j, g_j はそれぞれ独立な変数に依存しており互いに共有される変数節点は存在しないが、TSB が well-ordered であるとき f_j, g_j 部分の変数節点 v のうち $index(v) = \pi[1]$ を満たすものは h_{i_1} $(B_{i_1} \in \varphi[1])$ に共有される。このとき共有される変数節点数はタイプ 1 では 1 個、タイプ 2 では 2 個である。図 6 にその例を示す。このため、$\varphi[1]$ の変数ブロック内で変数の順序を変えた場合、タイプ 1、タイプ 2 の一方から他方に変化し wo-TSB の節点数が 1 だけ変化する場合がある。一方、$\varphi[1]$ 以外のブロックについては f_j, g_j に対応する変数節点は共有されないので、仮にある f_j に注目すると f_j が依存する変数は各変数ブロック

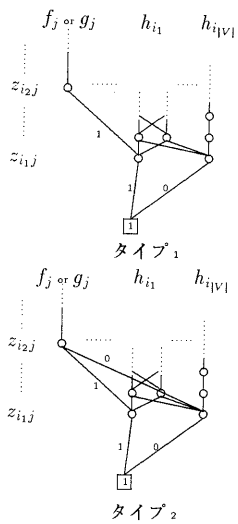


図 6: wo-TSB の 2 つのタイプ ($z_{ij} \in \{x_i, y_i\}$)

にただ一つ存在するのでブロック内で変数の順序を変えても f_j が依存する変数の順序は変化せず、従って変数節点数は変化しない。g_j についても同様である。h_i についてはブロック内の変数について対称なのでやはり変数節点数は変化しない。また、定数節点数は常に 2 であることから、題意は証明された。 *Q.E.D.*

以下の証明では補題 3 の結果を前提とする。

補題 4 (OLA と OVO の対応)
OLA において与えられるコスト K のグラフ G に対する *TSB* を補題 2 の f_j $(j = 1,2,\cdots,|E|)$、g_j $(i = 1,2,\cdots,|E|)$、及び h_i $(i = 1,2,\cdots,|V|)$ より構成する。この *TSB* が *well-ordered* であるとき、タイプ 1、タイプ 2 のサイズをそれぞれ $k_1(K), k_2(K)$ とすると次のようになる。

$$k_1(K) = 2\{K + 1 - |V| + |E||V|(|V|+2) - |E|\sum_{i=1}^{|V|-2} i\} \quad (4)$$

$$k_2(K) = 2\{K - |V| + |E||V|(|V|+2) - |E|\sum_{i=1}^{|V|-2} i\} + 1 \quad (5)$$

証明
補題 2 の f_j, g_j, 及び h_i より wo-TSB を構成する。以下、タイプ 1 の場合に限って話を進める。
各 f_j, g_j は 2 つの定数節点及び $\pi[1]$ に属する節点 1 個だけが必ず $h_{i_{|V|}}$ $(B_{i_{|V|}} = \varphi[|V|])$ に共有されるので wo-TSB の f_j, g_j 部分の変数節点数は全体として **OLA** のグラフ G のコストを反映する。よって **OLA** のコスト K のグラフ G に対応する wo-TSB の f_j、g_j 部分の変数節点数は $\sum_{j=1}^{|E|} size(f_j) + \sum_{j=1}^{|E|} size(g_j) - 4|E| - 1$ に等しい。また、$x_{i_1 j} \in \varphi[l_{j1}], x_{i_2 j} \in \varphi[l_{j2}]$ とすると、式 (3) より

$$\begin{aligned}\sum_{j=1}^{|E|} size(f_j) &= \sum_{j=1}^{|E|}(|l_{j1} - l_{j2}| + |V| + 2) \\ &= K + (|V|+2)|E|\end{aligned}$$

となる。$\sum_{j=1}^{|E|} size(g_j) = \sum_{j=1}^{|E|} size(f_j)$ より

$$\sum_{j=1}^{|E|} size(f_j) + \sum_{j=1}^{|E|} size(g_j) - 4|E| - 1 = 2(K + |V||E|) - 1 \quad (6)$$

次に wo-TSB の h_i 部分について考える。

$$t_{max} = \max\{\, t \mid \pi[t] \in B_i\}$$

$$t_{min} = \min\{\, t \mid \pi[t] \in B_i\}$$

とすると、

$$\mathcal{H}_p = \left\{ h_i|_{\bigwedge_{t=2|V||E|-p+1}^{2|V||E|} \pi[t]=b_t} \;\middle|\; \begin{array}{l} b_t \in \{0,1\} \\ t = 2|V||E| - p + 1, 2|V||E| - p + 2, \cdots, 2|V||E| \end{array} \right\}$$

に対し、

$$size(h_i) = |\bigcup_{p=1}^{2|V||E|} \mathcal{H}_p|$$

が成り立つ。また、
$t_{max} < 2|V||E| - j + 1 \le 2|V||E|$ のとき、

$$\mathcal{H}_j = \{0, \prod_{t=t_{max}+1}^{2|V||E|-j} \overline{\pi[t]} \bigoplus_{t=t_{min}}^{t_{max}} \pi[t] \prod_{t=1}^{t_{min}-1} \overline{\pi[t]}\}$$

$t_{min} < 2|V||E| - j + 1 \le t_{max}$ のとき、

$$\mathcal{H}_j = \{0, \bigoplus_{t=t_{min}}^{2|V||E|-j} \pi[t] \prod_{t=1}^{t_{min}-1} \overline{\pi[t]}, \overline{\bigoplus_{t=t_{min}}^{2|V||E|-j} \pi[t]} \prod_{t=1}^{t_{min}-1} \overline{\pi[t]}\}$$

$1 < 2|V||E| - j + 1 \le t_{min}$ のとき、

$$\mathcal{H}_j = \{0, \prod_{t=1}^{2|V||E|-j} \overline{\pi[t]}\}$$

$2|V||E| - j + 1 = 1$ のとき、

$$\mathcal{H}_j = \{0,\ 1\}$$

よって次式が成り立つ。

$$\begin{aligned} size(h_i) &= |\bigcup_{p=1}^{2|V||E|} \mathcal{H}_p| \\ &= 2|V||E| + 2|E| + 1 \end{aligned} \quad (7)$$

また、各 h_i $(B_i \neq \varphi[|V|])$ について、

$$F(v) = \prod_{t=1}^{t_{min}} \overline{\pi[t]}$$

または、$1 < 2|V||E| - j + 1 \le t_{min}$ なる j に対し

$$F(v) = \prod_{t=1}^{2|V||E|-j} \overline{\pi[t]}$$

を満たす節点 v 及び 2 つの定数節点は $h_{i_{|V|}}(B_{i_{|V|}} = \varphi[|V|])$ に共有される。よって、各 $h_i(B_i \neq \varphi[|V|])$ の節点で $h_{i_{|V|}}(B_{i_{|V|}} = \varphi[|V|])$ に共有されるものの個数は

$$2(l-1)|E| + 3 \ (\varphi[l] = B_i) \quad (8)$$

```
1  begin
2     k := 1;
3     while(k < 2mn) do
4        begin
5           π[k] ∈ B_i を満たす i を u に代入する;
6           k := k + 1;
7           while(∃s := min{ t | π[t] ∈ B_u, k ≤ t ≤ 2mn}) do
8              begin
9                 for ct := s downto k + 1 do
10                   π[ct] と π[ct − 1] を交換;
11                k := k + 1;
12             end;
13       end;
14 end.
```

図 7: nwo-TSB から wo-TSB への変換アルゴリズム

wo-TSB の h_i 部分の節点数は、式 (7),(8) より

$$\sum_{i=1}^{|V|} size(h_i) - (\text{共有される節点数})$$
$$= (2|V||E| + 2|E| + 1)|V| - \sum_{l=1}^{|V|-1} \{2(l-1)|E| + 3\}$$
$$= 2|E|(|V| + 1)|V| - 2|V| - 2|E| \sum_{i=1}^{|V|-2} i + 3 \qquad (9)$$

となる。式 (6) と式 (9) の和をとることにより式 (4) が得られる。また、タイプ 2 についても、$\pi[1]$ に対応する変数節点がタイプ 1 より一つ多く共有される以外は同様なことがいえるので式 (5) が得られる。 *Q.E.D.*

補題 5 (TSB のサイズの上限) 補題 4で構成した *TSB* のサイズの上限は **OLA** において与えられるグラフ G の節点数 n に対して $O(n^4)$ である。

証明
$|V| = n$ とすると、$|E|$ はグラフ G が完全グラフのとき最大値 $\frac{n(n+1)}{2}$ をとるから、

$$|E| \leq \frac{n(n+1)}{2} \qquad (10)$$

また、グラフ G のコスト K はグラフ G が完全グラフのとき最大値をとる。また、各枝のコストは $n-1$ 以下であるから、次式が成り立つ。

$$K < \frac{n(n+1)}{2}(n-1) \qquad (11)$$

$|V| = n$ 及び式 (10),(11) より、式 (4),(5) の k_1, k_2 の上限は $O(n^4)$ である。 *Q.E.D.*

補題 6 (wo-TSB への変換) 任意の *nwo-TSB* はそれより節点数が少ない関数的に等価な *wo-TSB* に変換できる。

証明
以下、式 (1) によって構成される wo-TSB に対し、図 7に示す変換アルゴリズムが題意を満たすことを証明する。
ただし、変換アルゴリズムの 7 行目の $\exists s := \min\{t \cdots\}$ は、$\min\{t \cdots\}$ が存在すれば真でありかつその値を s に代入し、存在しない場合は偽となる条件式を表す。
まず最初にある一つのブロックの変数を連続させるための手続き、7 〜 12 によって節点数が増加しないことを証明する。
7 〜 12 行目による変換は次に定義する TSB 1 から TSB 2 への変換として表すことができる。

TSB1

- $p := 2m \times c$ (c は非負整数) とする。さらに $p > 0$ のときには次の条件を満たす。
  条件 C(p): 各 t $(t = 1, \cdots, \frac{p}{2m})$ に対し、$\pi[2m(t-1)+1], \cdots, \pi[2tm]$ は同じ変数ブロックに属する。
- ある u $(1 \leq u \leq n)$ に対し $\pi[p+1] \in B_u$
- ある $q (0 \leq q \leq 2mn - 2m)$ に対し、$p + m + q = \max\{l | \pi[l] \in B_u\}$

□

TSB2

- $p > 0$ のとき**条件 C(p)** を満たす。
- $\pi[p+t] \in B_u$ $(t = 1, 2, \cdots, 2m)$

(すなわち、**条件 C(p+2m)** を満たす) □

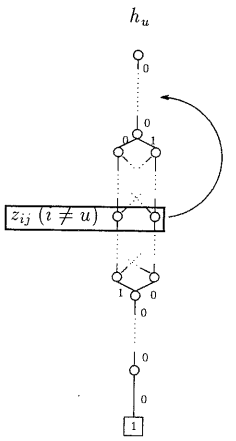


図 8: nwo-TSB の h_u に対応する部分 ($z_{ij} \in \{x_{ij}, y_{ij}\}$)

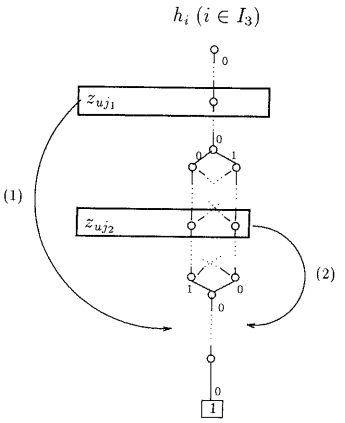


図 9: nwo-TSB の h_i $(i \in I_3)$ に対応する部分 ($z_{ij} \in \{x_{ij}, y_{ij}\}$)

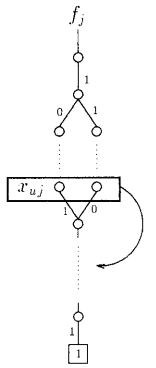


図 10: f_j の節点がもっとも増加する場合

$q = 0$ のときは 7 〜 12 によって変数の順序が変えられることはないので、節点数は変化しない。以下、$q > 0$ の場合について考える。

まず h_i の節点数の変化を以下の三通りの場合に分けて考える。

1. h_i $(\pi[t] \in B_i, t < t_{min})$
2. h_u
3. h_i $(\pi[t] \in B_i, t > t_{min}, i \neq u)$

ただし

$$t_{min} = \min\{\, t \mid \pi[t] \in B_u\} \tag{12}$$

とする。また、以下では h_i の節点の減少数を $-\Delta h_i$ $(-\Delta h_i \geq 0)$ で表し、

$$\begin{aligned} I_1 &= \{\, i \mid \pi[t] \in B_i, t < t_{min}\} \\ I_2 &= \{u\} \\ I_3 &= \{\, i \mid \pi[t] \in B_i, t > t_{min}, i \neq u\} \end{aligned}$$

とする。

1. h_i $(i \in I_1)$ について

$\pi[t]$ $(t < t_{min})$ に対応する変数の順序は変化しないことから次式が成り立つ。

$$-\Delta h_i = 0 \; (i \in I_1) \tag{13}$$

2. h_i $(i \in I_2)$ について

次に示すの変数の移動 (1)(図 8) が q 回起き、h_i の節点は q 個減少する。

変数の移動 (1) : 次のように t_{max} を定める。

$$t_{max} = \max\{\, t \mid \pi[t] \in B_u\} \tag{14}$$

このとき、

$$z_{ij} = \pi[t_1] \, (z_{ij} \in \{x_{ij}, y_{ij}\}, i \neq u, t_{min} < t_1 < t_{max})$$

を満たす変数 z_{ij} の $\pi[t_2]$ $(t_2 > t_{max})$ への移動。 □

よって、次式が成り立つ。

$$-\Delta h_i = q \tag{15}$$

3. h_i $(i \in I_3)$ について

新たに共有される部分が増えることによる節点の減少分の各 h_i $(i \in I_3)$ についての和を α 、また各 h_i $(i \in I_3)$ 自体の変形による節点数の減少分の各 h_i についての和を β とすると、次に示す変数の移動 (2) または変数の移動 (3) が、$q > 0$ ならば、各 h_i $(i \in I_3)$ について少なくとも 1 回起こるので $\alpha > 0$ である。また変数の移動 (3) は 0 回以上起こるので $\beta \geq 0$ である。

変数の移動 (2) : 以下のように t'_{max}, t'_{min} を定める。

$$t'_{max} = \max\{\, t \mid \pi(t) \in B_i\} \tag{16}$$

$$t'_{min} = \min\{\, t \mid \pi(t) \in B_i\} \tag{17}$$

このとき、

$$z_{uj} = \pi[t_1] \, (z_{uj} \in \{x_{uj}, y_{uj}\}, t_1 > t'_{max})$$

を満たす z_{uj} の $\pi[t_2]$ $(t_2 < t'_{min})$ への移動。 □

変数の移動 (3) : t'_{max}, t'_{min} を式 (17),(16) により定める。このとき、

$$z_{uj} = \pi[t_1] \, (z_{uj} \in \{x_{uj}, y_{uj}\}, t'_{min} < t_1 < t'_{max})$$

を満たす z_{uj} の $\pi[t_2]$ $(t_2 < t'_{min})$ への移動。 □

変数の移動 (2)、変数の移動 (3) をそれぞれ図 9の (1),(2) に示す。

よって、次式が成り立つ。

$$\sum_{i \in I_3} (-\Delta h_i) = \alpha + \beta > 0 \; (\alpha > 0 \, , \; \beta \geq 0) \tag{18}$$

式 (13)、(15)、(18) より次式が成り立つ。

$$\sum_{i=1}^{n} (-\Delta h_i) = q + \alpha + \beta > 0 \tag{19}$$

次に、f_j, g_j の節点数の増加数をそれぞれ $\Delta f_j, \Delta g_j$ で表すことにする。

x_{uj} と変数の順序が逆転した $x_{ij} (i \neq u)$ の数を $r1_j$ 、y_{uj} と変数の順序が逆転した $y_{ij} (i \neq u)$ の数を $r2_j$ とする。ただし、変数 z_1, z_2 の順序が逆転するとは $\pi[t_1] = z_1$ 、$\pi[t_2] = z_2$ であるとき t_1 と t_2 の大小関係が逆転することという。ここで、t_{max}, t_{min} を式 (12),(14) により定めると、x_{uj} と変数の順序が逆転する x_{ij} $(i \neq u)$ の集合 X_j は各 j に対し、次のように表される。

$$X_j = \{\, x_{ij} \mid \pi[t] = x_{ij}, t_{min} < t < t_{max}, \; 1 \leq i \leq n, \; i \neq u\}$$

同様に y_{uj} と変数の順序が逆転する y_{ij} の集合 Y_j も各 j に対し、次のように表される。

$$Y_j = \{\, y_{ij} \mid \pi[t] = y_{ij}, t_{min} < t < t_{max}, \; 1 \leq i \leq n, \; i \neq u\}$$

$\bigcap_{j=1}^{m} (X_j \cap Y_j) = \phi$ であるので、TSB 1 の定義より次式が成り立つ。

$$\sum_{j=1}^{m} (|X_j| + |Y_j|) = q$$

$|X_j| = r1_j$ 、$|Y_j| = r2_j$ であることより、

$$\sum_{j=1}^{m} (r1_j + r2_j) = q \tag{20}$$

また、f_j の節点数がもっとも増加するのは、

$$t_{j-min} = \min\{\, t \mid \pi[t] = x_{i_a j}, a \in \{1, 2\}\}$$

を満たす t_{j-min} に対して、$\pi[t_{j-min}] = x_{uj}$ でかつ x_{uj} が $\pi[t_{j-min} - r1_j]$ に移動するときである (図 10)。このとき、f_j 部分の節点数は $r1_j$ 個増加するから、次式が成り立つ。

$$\Delta f_j \leq r1_j \tag{21}$$

が成り立つ。g_j についても同様なことがいえて、

$$\Delta g_j \leq r2_j \tag{22}$$

式(20),(21),(22)より

$$\sum_{j=1}^{m}(\Delta f_j + \Delta g_j) \leq \sum_{j=1}^{m}(r1_j + r2_j) = q \tag{23}$$

式(19),(23)の和をとることにより次の式が得られる。

$$\sum_{j=1}^{m}(\Delta f_j + \Delta g_j) + \sum_{i=1}^{n}\Delta h_i < 0 \tag{24}$$

以上より、アルゴリズムの7行目から12行目によって $q = 0$ のときは節点数は変わらず、$q > 0$ のときは節点数が減少することが分かった。
アルゴリズムが停止するまでに7行目から12行目は n 回繰り返されるが、仮定より変換前のTSBはnon-well-formedであるので、そのうち少なくとも1回は $q > 0$ の場合がある。よって補題6は証明された。 *Q.E.D.*

定理 2 (OLA から OVO への多項式帰着) OVO は OLA より多項式帰着可能である。

証明
OVO において与えられるSBと自然数 k として、それぞれ補題4で構成したTSB、$k_1(K)$ を与えるものとする。ここで節点数 $k_1(K)$ 以下のTSBが見つかったと仮定する。これがwell-orderedのときは補題4の $k_1(K), k_2(K)$ を表す式より任意の自然数 K に対して $k_2(K) < k_1(K) < k_2(K+1)$ が導かれるので、**OLA** において与えられるグラフ G のコストが K 以下となる1対1写像 ψ が存在する。

一方、non-well-orderedのときは補題6より必ずそれより節点数の少ないwo-TSBに変換できるので $k-1$ 個以下の節点を持つwo-TSBの存在が保証される。従ってやはり **OLA** において与えられるグラフ G のコストが K 以下となる1対1写像 ψ が存在する。以上より、**OVO** は **OLA** から帰着されることが示された。

また、補題5よりこのTSBの節点数は **OLA** の節点数に対する多項式で表されるのでこの帰着は多項式帰着である。 *Q.E.D.*

定理 3 (OVO の *NP*完全性) OVO は *NP*完全である。

証明
命題1、定理1と定理2より題意は証明された。 *Q.E.D.*

4 結論

共有二分決定グラフ及び二分決定グラフは論理設計検証、論理照合、テスト生成、論理合成等さまざまな分野においてその有用性が示されつつある。

計算機上で、共有二分決定グラフや二分決定グラフを用いて効率良く論理関数を処理するためには、節点数を少なくする変数の順序付けを見つけることが非常に重要である。節点数を最小にする方法はこれまでいくつか提案されてきたが[5, 10]、節点数を少なくする変数の順序付けを見つける問題の難しさに対する理論的研究は、これまで、Bryant[3]によって行なわれた以外は、ほとんどなされていない。

本稿では共有二分決定グラフ及び二分決定グラフの節点数と変数の順序付けに関して、共有二分決定グラフの最適変数順序付け問題について、これが *NP*完全であることの証明を与えた。

以下のような点を明らかにすることが今後の課題である。

- 属性枝を用いた共有二分決定グラフの節点数と変数の順序付けに関する問題の計算複雑さ。
- 共有二分決定グラフの最小化問題、すなわち共有二分決定グラフ SB が与えられたとき、SB と同じ論理関数を表す共有二分決定グラフの集合のなかで SB は最小かという問題の計算複雑さ。
- 二分決定グラフの最適変数順序付け問題、すなわち節点数 n の既約な二分決定グラフ B と自然数 $k(<n)$ が与えられたとき、B と同じ論理関数を表す節点数 k 以下の二分決定グラフを実現するような変数の順序付けが存在するかという問題が、B の節点数に対して多項式時間で解けるような、B の表す論理関数の集合。

共有二分決定グラフや二分決定グラフは、組合せ問題等、他の幅広い分野への応用も考えられ、節点数と変数の順序付けの関係など、その性質の理論的な解明が重要になると考えられる。

謝辞

本研究に関して、貴重な御助言、御討論を頂いた高木直史助教授、武永康彦助手をはじめとする本学矢島研究室の諸氏に感謝いたします。

参考文献

[1] S.B.Akers: Binary Decision Diagrams, IEEE Trans. on Computers, C-27, 509-516, (1978).

[2] A.V.Aho, J.E.Hopcroft and J.D.Ullman: The Design and Analysis of Computer Algorithms, Addison-Wesley Reading MA(1974).

[3] R.E.Bryant: Graph Based Algorithms for Boolean Function Manipulation, IEEE Trans. on Computers, C-35, 677-691, (1986).

[4] M.Fujita, H.Fujisawa and N.Kawato: Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams, IEEE ICCAD-88, 2-5, (1988).

[5] S.J.Friedman and K.J.Supowt: Finding the Optimal Variable Ordering for Binary Decision Diagrams, IEEE Trans. on Computers, C-39, 710-713, (1986).

[6] M.R.Garey and D.S.Johnson: Computers and Intractability - A Guide to the Theory of *NP*-Completeness, W.H.FREEMAN AND COMPANY(1979).

[7] M.R.Garey, D.S.Johnson and L.Stockmeyer: Some Simplified NP-complete Graph Problems, Theoretical Computer Science 1, 237-267, (1976).

[8] S.Minato, N.Ishiura and S.Yajima: Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation, Proc. 27th ACM/IEEE DAC, 52-57, (1990).

[9] M.R.Mercer, R.Kapur and D.E.Ross: Functional Approaches to Generating Ordering for Efficient Symbolic Representations, Proc. 29th ACM/IEEE DAC, 624-629, (1992).

[10] N.Ishiura,H.Sawada and S.Yajima: Minimization of Binary Decision Diagrams Based on Exchanges of Variables, IEEE ICCAD-91, 472-475, (1991).