Vol.2014-AL-149 No.9
2014/9/12

k-IBDD充足可能性問題に対する厳密アルゴリズム

脊戸 和寿[1,a)]　照山 順一[2,3,b)]　長尾 篤樹[4,5,c)]

概要：k-IBDD は，k 段のレイヤーを持つ分岐プログラムであり，各レイヤーが OBDD(順序付き二分決定図) となっている．本稿では，k-IBDD 充足可能性問題（以下，k-IBDD SAT）を考える．k-IBDD SAT とは，入力として k-IBDD が与えられ，シンク 1 に到達する変数への割当が存在するかどうかを判定する問題である．本稿では，n 変数，$poly(n)$ ノードの 2-IBDD SAT を高々 $poly(n) \cdot 2^{n-\sqrt{n}}$ 時間で解く多項式領域アルゴリズムを与える． $poly(n)$ は n の多項式を表す．さらに，このアルゴリズムを拡張することにより， n 変数，$poly(n)$ ノードの k-IBDD SAT を高々，$poly(n) \cdot 2^{n-n^{1/2^{k-1}}}$ 時間で解く多項式領域アルゴリズムを与える．

1. はじめに

k-IBDD 充足可能性問題（以下，k-IBDD SAT）とは，与えられた k-IBDD において，シンク 1 に到達する変数割り当てが存在するかどうかを判定する問題である．k-IBDD とは k 段のレイヤーから構成され，各レイヤーは OBDD（順序付き二分決定図）となっている．k-IBDD SAT は，$k \geq 2$ において，NP 完全であることが知られている [2]. n 変数 m ノードの k-IBDD SAT は変数への全割り当てをチェックすることで，$O(m2^n)$ 時間で解くことができる．k-IBDD SAT において，全探索より超多項式的に高速な $O(m2^{n-\omega(\log n)})$ 時間のアルゴリズムの設計は目標の 1 つである．既に k-IBDD SAT を含む，一般の分岐プログラムに対し，$m = O(n^{2-\epsilon})$（ϵ は 0 より大きい任意に小さい定数）であれば，$O(2^{n-\omega(\log n)})$ 時間の決定性アルゴリズムで解けることが知られている [1]．しかし，このアルゴリズムは指数領域を必要とする上に，$m = \omega(n^2)$ ノードの k-IBDD SAT に対しては動作しない．本研究では，n 変数，$m = O(n^c)$（c は任意の定数）ノードの k-IBDD SAT を $O(m2^{n-\omega(\log n)})$ 時間で解く決定性多項式領域アルゴリズムの設計を目標とし，以下の結果を得た．

定理 1. n 変数，$m = O(n^c)$（c は任意の定数）ノードの *k-IBDD SAT* に対して，計算時間 $poly(n) \cdot 2^{n-n^{\alpha}}$ で解く決定性多項式領域アルゴリズムが存在する．ただし，$\alpha = \frac{1}{2^{k-1}}$ であり，$poly(n)$ は n の多項式を表す．

1.1 関連研究

Chen らは，n 変数，$m = O(n^{2-\epsilon})$ ノードの分岐プログラムにおける充足可能性問題に対し，$O(2^{n-n^{\delta}})$ 時間アルゴリズムを設計した [1]．ここで，ϵ は 0 より大きい任意に小さい定数であり，δ は $0 < \delta < 1$ を満たす ϵ に依存する定数である．Bollig らは k-IBDD SAT の特別な場合でもある k-OBDD SAT について，多項式時間アルゴリズムを考案した [2]．k-IBDD SAT に対しては，実験的に高速なアルゴリズムが Jain らによって提案されている [5].

2. 準備

$X = \{x_1, \ldots, x_n\}$ を論理変数の集合とする．本稿では，1 を論理値の真 (true)，0 を論理値の偽 (false) に対応させる．$x \in X$ について，論理変数 x の否定を $\overline{x}$ で表す．

非決定性分岐プログラム $B = (G, \phi_V, \phi_E)$ は根付有向多重グラフ $G = (V, E)$，ノードラベル関数 $\phi_V : V \to X \cup \{\mathbf{0}, \mathbf{1}\}$，枝ラベル関数 $\phi_E : E \to \{0, 1\}$ で構成される．G はルートノード r とシンクノード t_0, t_1 を持ち，$\phi(t_0) = \mathbf{0}$ かつ $\phi(t_1) = \mathbf{1}$ を満たす．t_0（または t_1）を 0-sink（または 1-sink）と呼ぶ．シンク以外の全てのノード $v \in V$ に対して，$\psi_V(v) \in X$ である．ϕ_E により G のすべての枝は 0 または，1 をラベルとして持つ．

図 1 は非決定性分岐プログラムの例である．各ノードは丸で表し，丸の中の記号はノードラベルを表している．ノードをつなぐ矢印は有向枝を表し，枝の近くにある 0 ま

1 成蹊大学, Seikei University
2 国立情報学研究所, National Institute of Informatics
3 JST, ERATO, 河原林巨大グラフプロジェクト，JST, ERATO, Kawarabayashi Large Graph Project
4 京都大学, Kyoto Uniersity
5 日本学術振興会特別研究員 DC, Research Fellow of Japan Society for the Promotion of Science
a) seto@st.seikei.ac.jp
b) teruyama@nii.ac.jp
c) a-nagao@kuis.kyoto-u.ac.jp

たは 1 は枝ラベルを表す．

入力 $a=(a_1,\ldots,a_n)\in\{0,1\}^n$ に対して，分岐プログラム B はルートノードから出発して以下のように枝をたどる．ノードラベルが x_i の時，a_i と等しいラベルを持つ枝を選択し次のノードに進む．最終的に t_1 に到達するパスが少なくともひとつある時，分岐プログラム B は入力 a に対し 1 を出力し，それ以外の場合 0 を出力する．

2 値関数 $f:\{0,1\}^n\to\{0,1\}$ に対して，すべての $a\in\{0,1\}^n$ に対して $f(a)$ と B の出力が等しいならば，**B は関数 f を表現する**という．B のサイズは，G の枝数とし，$|B|$ で表す．

非決定性分岐プログラムのうち，シンクノードでないすべてのノードからちょうど 2 本の枝が出て，それぞれにラベル 0,1 が 1 つずつ貼られているものを**決定性分岐プログラム**という．以下では，特に記述しない限り分岐プログラムは決定性分岐プログラムであるとする．

順列 $\pi=(\pi(1),\pi(2),\ldots,\pi(n))$ は 1 から n の数字を 1 つずつ含む任意の並びをもつ列である．また，$i\in\{1,\ldots,n\}$ に対して，$\pi^{-1}(i)$ を $\pi(j)=i$ を満たす j とする．

定義 1. OBDD とは，固定された順列 π に従う分岐プログラムである．つまり，ラベル x_i のノードからラベル x_j のノードに向かう枝があるならば，$\pi^{-1}(i)<\pi^{-1}(j)$ を満たす．

定義 2. k-IBDD とは，以下のような分岐プログラムである．k 個のレイヤーに分割でき，i 番目のレイヤーは順列 π_i に従う *OBDD* である．また，i 番目のレイヤーから出る枝は $j>i$ 番目のレイヤーのノードまたはシンクノードに到達する．特に，すべての π_i が等しいものを **k-OBDD** と呼ぶ．

図 1 の分岐プログラムは $\pi=(1,2,3)$ に従う非決定性 OBDD である．図 2 の分岐プログラムは $\pi_1=(1,2,3),\pi_2=(2,3,1)$ に従う 2-IBDD である．

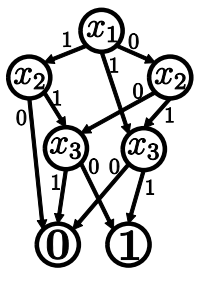

図 1　非決定性 OBDD

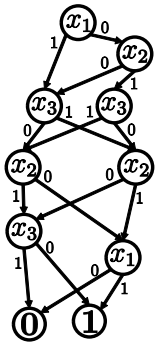

図 2　2-IBDD

与えられた k-IBDD B に対して，B が 1 を出力する入力 $a\in\{0,1\}^n$ があるかどうかを判定する問題を，k-IBDD に対する充足可能性問題，または k-IBDD SAT と呼ぶ．本問題は，$k=1$ の場合つまり OBDD が入力の場合ルートノードから 1-sink へのパスがあるかどうかは到達可能性判定を行えばよく，B のサイズの線形時間で解くことができる．$k=2$ の場合，NP 完全であることが知られている． [2]

順列 $\pi=(\pi(1),\pi(2),\ldots,\pi(n))$ の**逆順**を $\pi^R=(\pi^R(1),\pi^R(2),\ldots,\pi^R(n))=(\pi(n),\pi(n-1),\ldots,\pi(1))$ とする．順列 π における長さ m の部分列 π' とは，任意の $1\le i_1<i_2<\cdots<i_m\le n$ に対し，$\pi'=(\pi'(1),\pi'(2),\ldots,\pi'(m))=(\pi(i_1),\pi(i_2),\ldots,\pi(i_m))$ である．

順列 π の**最長増加部分列** σ_{inc} とは，π における部分列 π' のうち，すべての $1\le i<j\le m$ に対して $\pi'(i)<\pi'(j)$ を満たすもので，m が最大のものである．

順列 π の**最長減少部分列** σ_{dec} とは，π における部分列 π' のうち，すべての $1\le i<j\le m$ に対して $\pi'(i)>\pi'(j)$ を満たすもので，m が最大のものである．

最長増加部分列と最長減少部分列について，以下の定理が知られている．

定理 2 (The Erdős-Szekeres theorem [4])**.** 長さ n の実数列には，長さ $m\ge\sqrt{n}$ の最長増加部分列，または最長減少部分列が存在する．

また，最長増加部分列と最長減少部分列は $O(n^2)$ 時間で求めることができる．

部分割り当てとは $a=(a_1,\ldots,a_n)\in\{0,1,*\}^n$ で表し，a_i が 0 または 1 であることは，変数 x_i の値が a_i に固定されることを意味する．部分割り当て $a\in\{0,1,*\}^n$ に対して，$S(a):=\{x_i\mid\alpha_i\neq*\}$ を a **の台**と呼ぶ．$B|_a$ を部分割り当て a に対する分岐プログラム B の部分分岐プログラムとし，以下の操作により構成されるものとする．

(1) ラベル $x_i\in S(a)$ を持つノードから出る枝でラベル $\overline{a_i}$ を持つものをすべて削除する．

(2) 入次数が 0 でありルートノードでないノード v が存在する限り以下の操作を行う．

・ノード v と v から出る枝をすべて削除する．

(3) ラベル $x_i\in S(a)$ を持つルートノードでないノード v がある限り以下の操作を行う．

・集合 U を $\{u\mid(u,v)\in E\}$ とする．また，ラベル a_i を持つ枝 (v,w) が存在する．

・すべての $u\in U$ について (u,v) と同じラベルを持つ枝 (u,w) を追加し，(u,v) を削除する．

・枝 (v,w) を削除する．

(4) ルートノード r のラベル x_i が $x_i\in S(a)$ を満たすならば，枝 (r,v) を削除し，ノード v をルートノードとする．

図 3 は図 2 の 2-IBDD を B とした時，部分割り当て $a=(1,*,*)$ に対する $B|_a$ を表す．

3. OBDD の変換アルゴリズム

補題 3. 共通の順列 π に従う 2 つの非決定性 *OBDD* B_1 と B_2 が 2 値関数 f_1，f_2 を表現する時，関数 $f_1\wedge f_2$ を表現

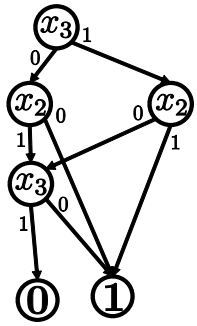

図 3　部分割り当てに対する 2-IBDD

Algorithm 1 Conjunction(B_1, B_2)

Require:
f_1 を表現する π に従う OBDD $B_1 = (G_1(V_1, E_1), \phi_{V_1}, \phi_{E_1})$,
f_2 を表現する π に従う OBDD $B_2 = (G_2(V_2, E_2), \phi_{V_2}, \phi_{E_2})$
Ensure:
$f_1 \wedge f_2$ を表現する π に従う OBDD $B = (G(V, E), \phi_V, \phi_E)$
1: $V := V_1 \times V_2$
2: $E := \emptyset$
3: **for all** $v := (v_1, v_2) \in V$ **do**
4: **if** $\phi_{V_1}(v_1) = \mathbf{0}$ **or** $\phi_{V_2}(v_2) = \mathbf{0}$ **then**
5: $\phi_V(v) = \mathbf{0}$
6: **else if** $\phi_{V_1}(v_1) = \mathbf{1}$ **and** $\phi_{V_2}(v_2) = \mathbf{1}$ **then**
7: $\phi_V(v) = \mathbf{1}$
8: **else**
9: $\phi_V(v) = x_\ell$, ただし $\phi_{V_1}(v_1) = x_i, \phi_{V_2}(v_2) = x_j$ に対して $\pi^{-1}(\ell) = \min\{\pi^{-1}(i), \pi^{-1}(j)\}$ を満たす.
10: **end if**
11: **end for**
12: **for all** $v := (v_1, v_2) \in V$ **do**
13: $E'_i := \{(v_i, u_i) \mid (v_i, u_i) \in E_i\}, i \in \{1, 2\}$
14: **if** $\phi_{V_1}(v_1) = \phi_{V_2}(v_2)$ **then**
15: **for all** $e_1 = (v_1, u_1) \in E'_1$ **do**
16: **for all** $e_2 = (v_2, u_2) \in E'_2$ **do**
17: **if** $\phi_{E_1}(e_1) = \phi_{E_2}(e_2)$ **then**
18: $e := ((v_1, v_2), (u_1, u_2))$
19: $E := E \cup \{e\}$
20: $\phi_E(e) = \phi_{E_1}(e_1)$
21: **end if**
22: **end for**
23: **end for**
24: **else if** $\phi_V(v) = \phi_{V_2}(v_2)$ **then**
25: **for all** $e = (v_2, u_2) \in E'_2$ **do**
26: $e := ((v_1, v_2), (v_1, u_2))$
27: $E := E \cup \{e\}$
28: $\phi_E(e) = \phi_{E_2}(e_2)$
29: **end for**
30: **else**
31: **for all** $e = (v_1, u_1) \in E'_1$ **do**
32: $e := ((v_1, v_2), (u_1, v_2))$
33: $E := E \cup \{e\}$
34: $\phi_E(e) = \phi_{E_1}(e_1)$
35: **end for**
36: **end if**
37: **end for**
38: **return** $B = (G(V, E), \phi_V, \phi_E)$

する順列 π に従う *OBDD* B を計算時間 $O(|B|)$ で構成することができる．また，$|B| \leq 2|B_1| \cdot |B_2|$ である．

Proof. Brayant [3] の手法を非決定性 OBDD に対して拡張した Algorithm 1 により補題を満たす OBDD が構成されることを示す．

構成される OBDD B は 2 つの OBDD B_1，B_2 を同時に模倣する．r_1, r_2 を B_1, B_2 のルートノードとする．B はノード (r_1, r_2) から計算を始める．B のノード (v_1, v_2) にいる時，B_1 のノード v_1，B_2 のノード v_2 にいることを意味する．入力 $a \in \{0,1\}^n$ に対する B の計算は以下の通りである．v_1 のラベルを x_i，v_2 のラベルを x_j とする．また，v_1 からラベル a_i を持つ枝で遷移した先のノードを u_1 とし，v_2 からラベル a_j を持つ枝で遷移した先のノードを u_2 とする．

v_1 と v_2 のラベルが共に x_i であれば，行 14–23 により B はラベル a_i を持つ枝により (v_1, v_2) から (u_1, u_2) に遷移する．B_1，B_2 において変数 x_i に対する計算を同時に行うことに対応する．v_1 と v_2 のラベルが異なる時，順列 π において $\pi^{-1}(i) < \pi^{-1}(j)$ であるとすると，行 24–36 によりラベル a_i を持つ枝により (v_1, v_2) から (u_1, v_2) に遷移する．B_1 において変数 x_i に対する遷移を行い，B_2 は v_2 に留まることに対応する．また，$\pi^{-1}(i) > \pi^{-1}(j)$ である場合は，ラベル a_j を持つ枝により (v_1, v_2) から (v_1, u_2) に遷移する．B_2 において変数 x_j に対する遷移を行い，B_1 は v_1 に留まることに対応する．以上から，B は順列 π に従うことがわかる．

行 4–5 により一方の OBDD で 0-sink に到達する入力に対して，B では 0-sink に到達する．行 6–7 により両方の OBDD で 1-sink に到達する入力に対して，B は 1-sink に到達する．よって，構成された OBDD は $f_1 \wedge f_2$ を表現する．

B において B_1 での遷移を表現する枝は高々 $|B_1| \cdot |B_2|$ であり，同様に B_2 での遷移を表現する枝は高々 $|B_1| \cdot |B_2|$ である．よって，B の枝数は高々 $2|B_1| \cdot |B_2|$ となる． □

補題 4. B を順列 π に従う *OBDD* とする．この時，$O(|B|)$ 時間で順列 π^R に従う非決定性 *OBDD* B^R を構成することができる．また，$|B^R| = O(|B|)$ である．

Proof. Algorithm 2 により補題の B^R が構成されることを示す．

まず，行 1–5 により 0-sink に入る枝をすべて削除する．行 6–19 ではどのノードに関しても，入る枝の始点のラベルが同じになるように与えられた B を変更している．ノード v に関して，ノード集合 $U = \{u \mid (u, v) \in E\}$ のラベルが異なる場合，U のノードのラベル x_i のうち $\pi^{-1}(i)$ が最大となるものを x_ℓ とする．$r(u) \neq x_\ell$ なるノード $u \in U$ に対して，以下の操作を行う．

(1) x_ℓ をラベルに持つノード w を追加する．
(2) 枝 (u, v) と同じラベルを持つ枝 (u, w) を追加する．
(3) ラベル 0, 1 を持つ枝 (w, v) を 1 つずつ追加する．

Algorithm 2 Reverse(B)

Require: 順列 π に従う OBDD $B = (G(V, E), \phi_V, \phi_E)$
Ensure: B と等価で順列 π^R に従う非決定性 OBDD $B^R = (G^R(V^R, E^R), \phi_{V^R}, \phi_{E^R})$
{ 前処理 }
1: **for all** $e = (u, v) \in E$ **do**
2: **if** $\phi_V(v) = \mathbf{0}$ **then**
3: $E := E \backslash \{e\}$
4: **end if**
5: **end for**
6: **for all** $v \in V$ **do**
7: $L_{(*,v)} := \{i \mid x_i = \phi_V(u)$ **and** $(u, v) \in E\}$
8: **if** $|L_{(*,v)}| > 1$ **then**
9: すべての $i \in L_{(*,v)}$ について $\pi^{-1}(\ell) \geq \pi^{-1}(i)$ を満たす $L_{(*,v)}$ の要素 ℓ を選択
10: **for all** $e = (u, v) \in E$ **do**
11: **if** $\phi(u) \neq x_\ell$ **then**
12: $V := V \cup \{w\}$ **and** $\phi_V(w) = x_\ell$
13: $E := E \cup \{e_1 := (w, v), e_2 := (w, v), e' := (u, w)\}$
14: $\phi_E(e_1) := 0, \phi_E(e_2) := 1, \phi_E(e') := \phi_E(e)$
15: $E := E \backslash \{e\}$
16: **end if**
17: **end for**
18: **end if**
19: **end for**
{ $G^R(V^R, E^R)$ の構成 }
20: $V^R := V$, $E^R := E$
21: **for all** $e = (u, v) \in E$ **do**
22: $\phi_{V^R}(v) := \phi_V(u)$
23: **end for**
24: $\phi_{V^R}(r) := \mathbf{1}$，ただし r は G のルートノード
25: **for all** $e = (u, v) \in E$ **do**
26: $e^R := (v, u)$
27: $\phi_{E^R}(e^R) = \phi_E(e)$
28: $E^R = E^R \cup \{e^R\} \backslash \{e\}$
29: **end for**
30: **for all** $v \in V^R$ **do**
31: **if** ラベル $b \in \{0, 1\}$ を持つ枝 (v, w) を持たない **then**
32: $e^R := (v, t_0)$
33: $E^R = E^R \cup \{e^R\}$
34: $\phi_{E^R}(e^R) = b$
35: **end if**
36: **end for**
37: **return** $B = (G^R(V^R, E^R), \phi_{V^R}, \phi_{E^R})$

(4) 枝 (u, v) を削除する．

以上の操作により，元の OBDD と等価で，どのノードも入る枝の始点のラベルがすべて同じである OBDD を構成することができる．上の操作で新たに追加されるノードは入次数が 1 であるから操作の対象になることはない．よって上の操作は高々 $|E|$ 回行われ，サイズは高々 3 倍になる．図 4 は図 1 の OBDD に対して以上の操作を行った後の状態を表している．

行 20–36 では，以下の操作で B^R を構成する．

(1) 各ノードのラベルを，入る枝の始点のラベルに置き換える．ルートノードは 1-sink に書き換える．

(2) 枝を逆向きにする．

(3) 各ノード v に対してラベル $b \in \{0, 1\}$ を持つ枝 $(v, w), w \in V$ が存在しない場合，ラベル b を持つ枝 (v, t_0) を追加する．

この操作により得られる B^R の各計算経路は，もとの OBDD の計算経路を逆順に辿ったものと対応する．ただし，元の OBDD で同じラベルを持った枝が合流した場合，B^R では非決定的遷移をすることに注意する．図 5 は図 1 の OBDD に対して出力される非決定性 OBDD を表している．

以上から B と等価で π^R に従う非決定 OBDD B^R が構成され，$|B^R| = O(|B|)$ である． □

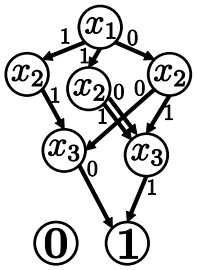

図 4 前処理

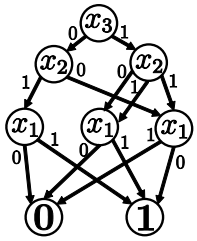

図 5 B^R

4. 充足可能性判定アルゴリズム

定数 k に対して，k-OBDD SAT は多項式時間で解けることが知られている [2]，つまり，入力の k-IBDD において $\pi_1 = \pi_2 = \cdots = \pi_k$ ならば多項式時間で解くことができる．

まず，k-OBDD を含む特別な k-IBDD SAT が多項式時間で解くことができることを示す．アルゴリズムを Algorithm 3 に示す．

補題 5. k を定数とする．順列 $\pi_1, \pi_2, \ldots, \pi_k$ が，すべての i $(2 \leq i \leq k)$ について $\pi_i = \pi_1$，または $\pi_i = \pi_1^R$ を満たすとする．順列 $\pi_1, \pi_2, \ldots, \pi_k$ に従う n 変数，$m = O(n^c)$（c は任意の定数）ノードの *k-IBDD* G に対して，*k-IBDD SAT* は n の多項式時間で解くことができる．

Proof. [2] の定理 2 の証明を非決定性 k-OBDD に拡張する．B を補題の条件を満たす非決定性 k-OBDD とし，各レイヤーを $B_1, B_2, \ldots, B_k$，各レイヤーのノード集合を $V_1, \ldots, V_k$ とする．

行 1–16 ではすべての枝 (u, v) に対して，ある i が存在して，$u, v \in V_i$ または $u \in V_i$ かつ $v \in V_{i+1}$ を満たすように B を変更している．ノード $u \in V_i$ とノード $v \in V_j$ に対して，$(u, v) \in E$ かつ $j - i > 1$ である場合に以下の操作を行う．

(1) すべての ℓ $(i < \ell < j)$ に対して，ラベル $x_{\pi_\ell(1)}$ を持つノード w_ℓ を B_ℓ に追加する．

(2) 枝 (u, v) と同じラベルを持つ枝 (u, w_{i+1}) を追加する．

(3) すべての ℓ $(i < \ell < j - 1)$ に対して，ラベル 0, 1 を

Algorithm 3 (π, π^R)-k-IBDD SAT(B)

Require: 順列 $\pi_1, \ldots, \pi_k$ に従う k-OBDD $B = (G, \phi_V, \phi_E)$,
ただし $\pi_i = \pi_1$ または $\pi_i = \pi_1^R$ $(2 \le i \le k)$
Ensure: B が充足可能であれば "Yes"，そうでなければ "No".

for all $e = (u, v) \in E$ **do**
  if $j - i > 1$，ただし $u \in V_i, v \in V_j$ **then**
    for $\ell = i+1, \ldots, j-1$ **do**
      $V_\ell := V_\ell \cup \{w_i\}$ **and** $\phi_V(w_i) := x_{\pi_\ell(1)}$
    end for
    $E := E \cup \{e' := (u, w_{i+1})\}$
    $\phi_E(w_i) := x_{\pi_\ell(1)}$
    for $\ell = i+1, \ldots, j-2$ **do**
      $E := E \cup \{e'_0 := (w_\ell, w_{\ell+1}), e'_1 := (w_\ell, w_{\ell+1})\}$
      $\phi_E(e'_0) := 0, \phi_E(e'_0) := 1$
    end for
    $E := E \cup \{e'_0 := (w_{j-1}, v), e'_1 := (w_{j-1}, v)\}$
    $\phi_E(e'_0) := 0, \phi_E(e'_0) := 1$
    $E := E \backslash \{e\}$
  end if
end for
for all $(r_2, \ldots, r_k) \in V_2 \times \cdots \times V_k$ **do**
  for $i = 1, \ldots, k$ **do**
    $V'_i := \{v \mid v \in V_i$かつ r_iから到達可能 $\} \cup \{t^i_0, t^i_1\}$
    $\phi_{V'_i}(t^i_0) := \mathbf{0}, \phi_{V'_i}(t^i_1) := \mathbf{1}$
    $E'_i := \emptyset$
    for all $e = (u, v) \in E$，ただし $u \in V'_i$ **do**
      if $v \in V'_i$ **then**
        $E'_i := E'_i \cup \{e' := (u, v)\}$
      else if $v = r_{i+1}$ **then**
        $E'_i := E'_i \cup \{e' := (u, t^i_1)\}$
      else
        $E'_i := E'_i \cup \{e' := (u, t^i_0)\}$
      end if
      $\phi_{E'_i}(e') := \phi_E(e)$
    end for
    $B'_i := (G'_i(V'_i, E'_i), \phi_{V'_i}, \phi_{E'_i})$
  end for
  for $i = 2, \ldots, k$ **do**
    if $\pi_i = \pi_1^R$ **then**
      $B'_i :=$Reverse(B'_i)
    end if
    $B'_1 :=$Conjunction(B'_1, B'_i)
  end for
  if B'_1 がルートノードから 1-sink までのパスを持つ **then**
    return "Yes"
  end if
end for
return "No"

持つ枝 $(w_\ell, w_{\ell+1})$ を 1 つずつ追加する．

(4) ラベル 0, 1 を持つ枝 (w_{j-1}, v) を 1 つずつ追加する．

(5) 枝 (u, v) を削除する．

上の操作で新たに追加される枝は第 ℓ レイヤーのノードと第 $\ell+1$ レイヤーのノード $(i \le \ell < j)$ を結ぶため操作の対象とならない．よって上の操作は高々 $|E|$ 回行われ，サイズは高々 $2k$ 倍になる．

B に対して充足する入力の計算経路は，$V_2, V_3, \ldots, V_k$ のあるノード $r_2, \ldots, r_k$ を経由して 1-sink へ到達する．経由ノードの可能性は高々 $(2k|B|)^{k-1}$ 通りである．各可能性に対して，そのノードを経由して 1-sink に到達する入力があるかを判定する．

$k-1$ 個のノード $r_2, \ldots, r_k$ $(r_i \in V_i)$ を選択し，$B_1, B_2, \ldots, B_k$ をもとにして k 個の OBDD $B'_1, B'_2, \ldots, B'_k$ を構成する．また，各 OBDD B'_i は順列 π_i に従う．簡単のため，r_1 を B のルートノード，r_{k+1} を 1-sink とする．各 i $(1 \le i \le k)$ について B_i から r_i をソースノードとする OBDD B'_i を以下のように構成する．まず，r_i から到達可能である V_i のノード集合およびシンクノード $t_{i,0}$，$t_{i,1}$ を B'_i のノード集合 V'_i とする．B'_i の枝集合は，すべての枝 $e \in \{(u, v) \mid (u, v) \in E$ かつ $u \in V'_i\}$ に対して，以下の条件を満たしラベル $\phi_E(u, v)$ を持つ枝 e' で構成される．

- $v \in V'_i$ ならば，$e' := (u, v)$.
- $v = r_{i+1}$ ならば，$e' := (u, t_{i,1})$.
- $v \ne r_{i+1}$ ならば，$e' := (u, t_{i,0})$.

k-IBDD B が $r_2, \ldots, r_k$ を経由する充足入力を持つならばそのときに限り，$B'_1, B'_2, \ldots, B'_k$ は共通の充足入力を持つ．補題 4 より，$\pi_i = \pi_1^R$ に従う決定性 OBDD B'_i と等価な $\pi_i^R = \pi_1$ に従う非決定性 OBDD B'^R_i が構成でき，これを B'_i と置き換える．以上から順列 π_1 に従う k 個の非決定性 OBDD $B'_1, B'_2, \ldots, B'_k$ を同時に充足する入力を持つかどうかを判定すればよい．補題 3 を $k-1$ 回適用することにより，k 個の OBDD が共通の充足入力を持つ時に，またその時に限り 1 を出力する OBDD B^* を $O(|B|^k)$ 時間で構成することができる．また，$|B^*| = O(|B|^k)$ である．OBDD B^* の充足可能性判定は到達可能性判定により $O(|B^*|) = O(|B|^k)$ 時間で解くことができる．

以上から，B に対する充足可能性問題は $O(|B|^{2k-1})$ 時間で解くことができる． □

定理 1 の前に，2-IBDD SAT が 2^n 時間より超多項式的に高速に解くことができることを示す．2-IBDD SAT を解く決定性多項式領域アルゴリズムは Algorithm 4 である．

Algorithm 4 2-IBDD SAT

Require: 順列 π_1, π_2 に従う 2-IBDD $B = (G, \phi_V, \phi_E)$,
ただし，$\pi_1 = (1, \ldots, n)$
Ensure: B が充足可能であれば "Yes"，そうでなければ "No".

π_2 の最長増加部分列 σ_{inc}，最長減少部分列 σ_{dec} を計算
if $|\sigma_{inc}| \ge |\sigma_{dec}|$ **then**
  $\sigma = \sigma_{inc}$
else
  $\sigma = \sigma_{dec}$
end if
$Y := \{\sigma(i) \mid 1 \le i \le |\sigma|\}$
for all $a \in \{0, 1, *\}^n$，ただし $S(a) = X \backslash Y$ **do**
  if (σ, σ^R)-k-IBDD SAT$(B|_a)$ = "Yes" **then**
    return "Yes"
  end if
end for
return "No"

定理 6. n 変数，$m = O(n^c)$（c は任意の定数）ノードの 2-*IBDD SAT* に対して，計算時間 $poly(n) \cdot 2^{n-\sqrt{n}}$ で解く決定性多項式領域アルゴリズムが存在する．ただし，$poly(n)$ は n の多項式を表す．

Proof. 入力の 2-IBDD を B とし，順列 π_1, π_2 に従うとする．まず，一般性を失うことなく $\pi_1 = (1, 2, \ldots, n)$ としてよい．以下ではアルゴリズムの概要を述べる．

π_2 の最長増加列を σ_{inc}，最長減少列を σ_{dec} とする．このうち長いほうを σ とし，その長さを $|\sigma|$ とする．$Y := \{\sigma(i) \mid 1 \leq i \leq |\sigma|\}$ とする．変数集合 $X \backslash Y$ を台としたすべての部分割り当てを行う．各部分割り当て a について，$B|_a$ が充足可能かどうかを判定する．$B|_a$ が充足可能となる部分割り当て a が存在すれば，B も充足可能である．また，すべての部分割り当てにおいて $B|_a$ が充足不可能であれば B も充足不可能である．

$\sigma = \sigma_{inc}$ とすると $B|_a$ は順列 σ に従う 2-OBDD となり，一方，$\sigma = \sigma_{dec}$ とすると順列 σ^R, σ に従う 2-IBDD となる．補題 5 および [2] によりいかなる場合でも，$B|_a$ に対する充足可能性問題は多項式時間で解ける．定理 2 より $|Y| \geq \sqrt{n}$，つまり $|X \backslash Y| \leq n - \sqrt{n}$ となり，全体の計算時間は高々 $poly(n) \cdot 2^{n-\sqrt{n}}$ となる． □

最後に，主定理の k-IBDD SAT を決定性多項式領域で解くアルゴリズムを Algorithm 5 に示す．

Algorithm 5 k-IBDD SAT

Require: 順列 $\pi_1, \ldots, \pi_k$ に従う k-IBDD $B = (G, \phi_V, \phi_E)$，ただし，$\pi_1 = (1, \ldots, n)$
Ensure: B が充足可能であれば “Yes”，そうでなければ “No”．

```
σ_1 := π_1
for i = 2, ..., k do
    π'_i := (π_i(j_1), π_i(j_2), ..., π_i(j_{|σ_{i-1}|})),
    ただし j_1 < j_2 < ··· < j_{|σ_{i-1}|} かつ ∀p, ∃q, π'_i(p) = σ_{i-1}(q)
    π'_i の最長増加部分列 σ_inc，最長減少部分列 σ_dec を計算
    if |σ_inc| ≥ |σ_dec| then
        σ_i := σ_inc
    else
        σ_i := σ_dec
    end if
end for
Y := {σ_k(i) | 1 ≤ i ≤ |σ_k|}
for all a ∈ {0, 1, *}^n, ただし S(a) = X\Y do
    if (σ_k, σ_k^R)-k-IBDD SAT(B|_a) = "Yes" then
        return "Yes"
    end if
end for
return "No"
```

定理 7 (定理 1 の再掲)．n 変数, $m = O(n^c)$ (c は任意の定数) ノードの *k-IBDD SAT* に対して，計算時間 $poly(n) \cdot 2^{n-n^\alpha}$ で解く決定性多項式領域アルゴリズムが存在する．ただし，$\alpha = \frac{1}{2^{k-1}}$ であり，$poly(n)$ は n の多項式を表す．

Proof. 入力の k-IBDD を B とし，順列 $\pi_1, \pi_2, \ldots, \pi_k$ に従うとする．一般性を失うことなく $\pi_1 = (1, 2, \ldots, n)$ としてよい．以下ではアルゴリズムの概要を述べる．

まず，π_2 の最長増加部分列と最長減少部分列を求め，長いほうを σ_2 とし，その長さを $|\sigma_2|$ とする．定理 2 より $|\sigma_2| \geq \sqrt{n}$ である．行 3 により π_3 から σ_2 の要素だけを抽出した部分列 π'_3 を得る．次に，π'_3 の最長増加部分列と最長減少部分列を求め，長いほうを σ_3 とし，その長さを $|\sigma_3|$ とする．σ_3 は σ_2 の要素からなる増加列または減少列である．σ_2 自身も増加列または減少列であることから，σ_3 は σ_2 または σ_2^R の部分列である．よって，π_1, π_2, π_3 は σ_3 または σ_3^R を部分列として持つ．また，$|\pi'_3| = |\sigma_2|$ であることに注意すると，定理 2 より $|\sigma_3| \geq n^{1/4}$ である．以下同様に，$2 \leq i \leq k$ に対して，π_i から σ_{i-1} の要素だけを抽出した部分列を π'_i とする．π'_i の最長増加部分列，最長減少部分列のうち長いほうを σ_i とし，その長さを $|\sigma_i|$ とする．$\pi_1, \pi_2, \ldots, \pi_i$ は σ_i または σ_i^R を部分列として持ち，帰納的に $|\sigma_i| \geq n^{1/2^{i-1}}$ となる．また，各 σ_i は多項式時間で求めることができる．

$Y := \{\sigma_k(i) \mid 1 \leq i \leq |\sigma_k|\}$ とする．定理 6 の証明と同様に，変数集合 $X \backslash Y$ を台としたすべての部分割り当てを行う．各部分割り当て a について，$B|_a$ が充足可能かどうかを判定する．$\pi_1, \pi_2, \ldots, \pi_k$ が σ_k または σ_k^R を部分列として持つことから，各割り当てに対して k-IBDD $B|_a$ は，各レイヤーが順列 σ_k または σ_k^R に従う k-IBDD となる．補題 5 より $B|_a$ の充足可能性判定は多項式時間でできる．

$\alpha = \frac{1}{2^{k-1}}$ とすると，$|X \backslash Y| = n - |\sigma_k| \leq n - n^\alpha$ となる．以上から，全体の計算時間は高々 $poly(n) \cdot 2^{n-n^\alpha}$ となる． □

謝辞 本研究は MEXT 科研費 24106003，JSPS 科研費 26730007 および JST ERATO 河原林巨大グラフプロジェクトの助成を受けたものです。

参考文献

[1] R. Chen, V. Kabanets, A. Kolokolova, R. Shaltiel and D. Zuckerman. Mining Circuit Lower Bound Proofs for Meta-Algorithms. In Proceedings of the 29th Annual IEEE Conference on Computational Complexity (CCC), 2014.

[2] B. Bollig, M. Sauerhoff, D Sieling, I Wegener. On The Power Of Different Types Of Restricted Branching Programs. Electronic Colloquium on Computational Complexity (ECCC), vol.1, no.26, 1994.

[3] R. E. Bryant. Graph-based algorithm for Boolean function manipulation. IEEE Trans. on Computers 35, pp.677–691, 1986.

[4] P. Erdos, G. Szekeres. A combinatorial problem in geometry. Compositio Mathematica, 2, pp.463–470, 1935.

[5] J. Jain, J. Bitner, M. S. Abadir, J. A. Abraham and D. S. Fussell. Indexed BDDs : Algorithmic Advances in Techniques to Represent and Verify Boolean Functions. IEEE Transaction on Computers, vol. 46(11), pp.1230–1245, 1997.