Vol.2015-AL-152 No.1
2015/3/3

線形サイズ 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，与えられた k-IBDD が 1 を出力するような変数割当が存在するかどうかを判定する問題である．本問題に対して，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が知られている．ここで，$poly(n)$ は n の多項式を表す．本稿では，n 変数，cn ノードの k-IBDD SAT を高々 $poly(n) \cdot 2^{(1-\mu(c))n}$ 時間で解く指数領域アルゴリズムを与える．ここで，$\mu(c) = \Omega\left(\frac{1}{(\log c)^{2^{k-1}-1}}\right)$ である．我々のアルゴリズムは既存のアルゴリズムを拡張することで線形サイズの k-IBDD に対して 2^n 時間より指数的な高速化を達成している．

1. はじめに

本稿では, k-IBDD 充足可能性問題（以下, k-IBDD SAT）について研究する．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，$k = 1$ ならば線形時間で解くことができる．また，$k \geq 2$ 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，$O(m \cdot 2^n)$ 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 $O(m \cdot 2^{n-\omega(\log n)})$ 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．

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

本問題に対して，[6] によって n 変数，$m = O(n^c)$（c は任意の定数）ノードの k-IBDD SAT を $O(m \cdot 2^{n-\omega(\log n)})$ 時間で動作する決定性多項式領域アルゴリズムで解けることが示されている．

定理 1 (定理 1 [6]). 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．

多項式サイズの k-IBDD であれば超多項式的に高速なアルゴリズムとなっている．しかしながら，ノード数を n の線形に制限しても指数的に高速なアルゴリズム，つまり定数 $\epsilon > 0$ に対して $2^{(1-\epsilon)n}$ 時間で解くアルゴリズムはまだ知られていない．本研究では，サイズが n の線形である k-IBDD SAT を全探索より指数的に高速なアルゴリズム設計を目標とし以下の結果を得た．

定理 2. 任意の定数 $c > 0$ に対して, $\mu(c) > 0$ を満たす関数 $\mu(c)$ が存在して, n 変数, $m = cn$ ノードの *k-IBDD SAT* を $poly(n) \cdot 2^{(1-\mu(c))n}$ 時間，$poly(n) \cdot 2^{\nu(c)n}$ 領域で解く決定性アルゴリズムが存在する．ただし, $\nu(c) = O\left(\frac{1}{(\log c)^{2^{k-1}-1}}\right)$ であり，$poly(n)$ は n の多項式を表す．また，十分大きな c について $\mu(c) = \Omega\left(\frac{1}{(\log c)^{2^{k-1}-1}}\right)$ である．

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，$O\left(2^{n-n^{\delta}}\right)$ 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

2.1 諸定義

$X = \{x_1, \ldots, x_n\}$ 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 $\bar{x}$ 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．

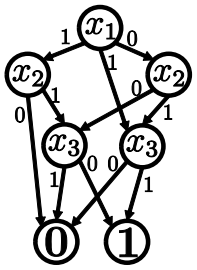

図 1 非決定性 OBDD

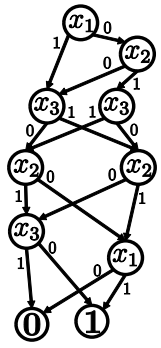

図 2 2-IBDD

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 3 (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，$S(a) := \{x_i \mid \alpha_i \ne *\}$ 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．

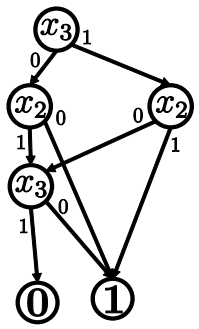

図 3 部分割り当てに対する 2-IBDD

2.2 既存手法

本節では我々のアルゴリズムを構成する上でサブルーチンとなる既存手法と，本手法に使用されるいくつかの補題を [6] より引用する．

補題 4 (補題 3 [6])．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．

補題 5 (補題 4 [6])．B を順列 π に従う *OBDD* とする．このとき，$O(|B|)$ 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．

Bollig らによって，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．さらに，Bollig らの手法を拡張することで，すべての i に対して $\pi_i = \pi_1$ または $\pi_i = \pi_1^R$ を満たすならば多項式時間で解けることが知られている [6]．アルゴリズムを Algorithm 1 に示す．

補題 6 (補題 5 [6])．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* B に対して，*k-IBDD SAT* は n の多項式時間で解くことができる．

補題 6 の証明における計算時間と計算領域の解析を厳密にすることにより，サイズ m とレイヤー数 k を任意の値とした以下の補題を得られる．

補題 7. k を正の整数とする．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 ノードの *k-IBDD* B に対して，*k-IBDD SAT* は $O((m/k)^{2k})$ 時間，$O((m/k)^k)$ 領域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 $(i \leq \ell < j)$ 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 $(2k)^{k-1} \prod_{l=2}^{k} |B_i|$ 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 $(1 \leq i \leq k)$ 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

Algorithm 1 (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) { 補題 5 の適用 }
    end if
    $B'_1 :=$Conjunction(B'_1, B'_i) { 補題 4 の適用 }
  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"

$e \in \{(u, v) \mid (u, v) \in E$ 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，$e' := (u, t_{i,1})$.
- $v \neq r_{i+1}$ 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 5 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 4 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 $O(\prod_{i=1}^k |B_i|)$ 時間で構成することができる．また，$|B^*| = O(\prod_{i=1}^k |B_i|)$ である．OBDD B^* の充足可能性判定は到達可能性判定により $O(|B^*|) = O(\prod_{i=1}^k |B_i|)$ 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 $O(\prod_{i=1}^k |B_i| \cdot \prod_{i=2}^k |B_i|)$ 時間で解くことができる．また，$\sum_{i=1}^k |B_i| = m$ と相加相乗平均から $\prod_{i=1}^k |B_i| \le (m/k)^k$ となる．よって，全体の計算時間は $O(\prod_{i=1}^k |B_i| \cdot \prod_{i=2}^k |B_i|) = O((m/k)^{2k})$ となり，計算領域 $O(\prod_{i=1}^k |B_i|) = O((m/k)^k)$ を必要とする． □

3. 充足可能性判定アルゴリズム

本節では入力が線形サイズである k-IBDD SAT が 2^n 時間より指数的に高速に解くことができることを示す．アルゴリズムのアイデアをわかりやすくするために，まず入力が線形サイズである 2-IBDD SAT が 2^n 時間より指数的に高速に解くことができることを示す．2-IBDD SAT を解く決定性アルゴリズムは Algorithm 2 である．

Algorithm 2 2-IBDD SAT(B, L)

Require: 順列 π_1, π_2 に従う 2-IBDD $B = (G, \phi_V, \phi_E)$,
ただし，$\pi_1 = (1, \ldots, n)$，整数 L
Ensure: B が充足可能であれば "Yes"，そうでなければ "No".
for all $\ell \in \{1, \ldots, L\}$ **do**
  $\pi_2^\ell := (\pi_2(\lceil n/L \rceil (\ell - 1) + 1), \ldots, \pi_2(\lceil n/L \rceil \ell))$
  π_2^ℓ の最長増加部分列 σ^ℓ_{inc}，最長減少部分列 σ^ℓ_{dec} を計算
  if $|\sigma^\ell_{inc}| \ge |\sigma^\ell_{dec}|$ **then**
    $\sigma^\ell := \sigma_{inc}$
  else
    $\sigma^\ell := \sigma_{dec}$
  end if
  $I^\ell := \{\sigma^\ell(i) \mid 1 \le i \le |\sigma^\ell|\}$
end for
$I := \cup_{\ell \in \{1, \ldots, L\}} I^\ell$
$Y := \{x_i \mid i \in I\}$
$\pi := sort(I)$ {I の要素を昇順に並び替え }
for all $a \in \{0, 1, *\}^n$，ただし $S(a) = X \backslash Y$ **do**
  if (π, π^R)-k-IBDD SAT$(B|_a)$ = "Yes" **then**
    return "Yes"
  end if
end for
return "No"

アルゴリズムの概要を述べる．我々のアルゴリズムの主なアイデアは，部分割り当てにより補題 7 が適用できる k-IBDD を得ることである．まず，パラメータ L を用いて第 2 レイヤーを L 分割する．以下では $L = 2$ の場合を考え，簡単のため n は偶数とする．列 π'_2, π''_2 を $\pi'_2 = (\pi_2(1), \ldots, \pi_2(n/2))$，$\pi''_2 = (\pi_2(n/2+1), \ldots, \pi_2(n))$ と定義する．2 分割した第 2 レイヤーのうち列 π'_2 に従う

レイヤーを第 2-1 レイヤー，列 π_2'' に従うレイヤーを第 2-2 レイヤーと呼ぶこととする．

ここで，σ' として π_2' の最長増加部分列または最長減少部分列の長いほうを選択する．同様に σ'' として π_2'' の最長増加部分列または最長減少部分列の長いほうを選択する．例として，σ' を π_2' の最長増加部分列，σ'' を π_2'' の最長減少部分列とする．列 π を σ' と σ'' の要素を昇順に並べた列とする．このとき，列 σ' は π の部分列であり，列 σ'' は π^R の部分列である．変数集合 Y を $\{x_i \mid \exists j, i = \pi(j)\}$ とする．つまり，集合 Y は π の要素をインデックスに持つ変数の集合である．さて，変数集合 $X\backslash Y$ を台とした部分割り当てから得られる部分分岐プログラムについて考える．変数 $x_i \in X\backslash Y$ をラベルに持つノードは削除され，変数 $x_i \in Y$ をラベルに持つノードのみで構成される．また，各レイヤーの順序は保存されるので，第 1 レイヤーは順序 π に従う．第 2-1 レイヤーは列 σ' に従うが，σ' は π の増加部分列であるから第 2-1 レイヤーは列 π に従うと考えられる．同様に第 2-2 レイヤーは列 π^R に従う．よって，変数集合 $X\backslash Y$ を台とした部分割り当てをして得られる部分分岐プログラムは順序 π 及び π^R に従う 3-IBDD となる．以上の操作により補題 7 が適用できる．

計算時間の最適化のため，分割数 L を最適に選択することにより指数的に高速なアルゴリズムを構成することができる．

定理 8. 任意の定数 $c > 0$ に対して，$\mu(c) > 0$ を満たす関数 $\mu(c)$ が存在して，n 変数，$m = cn$ ノードの 2-*IBDD SAT* を $poly(n) \cdot 2^{(1-\mu(c))n}$ 時間，$poly(n) \cdot 2^{\nu(c)n}$ 領域で解く決定性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．また，十分大きな c について $\mu(c) = \Omega\left(\frac{1}{\log c}\right)$ であり，$\nu(c) = O\left(\frac{1}{\log c}\right)$ である．

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 2-IBDD SAT(B, L) の動作及び計算量を解析する．まず，順列 π_2 を L 分割し，$\pi_2^1, \ldots, \pi_2^L$ とする．つまり，各 $\ell \in \{1, \ldots, L\}$ に対して $\pi_2^\ell := (\pi_2(\lceil N/L \rceil(\ell-1)+1), \ldots, \pi_2(\lceil N/L \rceil \ell))$ とする．π_2^ℓ の最長増加部分列を σ_{inc}^ℓ，最長減少部分列を σ_{dec}^ℓ とする．これらのうち長いほうを σ^ℓ とし，その長さを $|\sigma^\ell|$ で表す．各 $\ell \in \{1, \ldots, L\}$ に対して，集合 I^ℓ を $I^\ell = \{\sigma^\ell(i) \mid 1 \le i \le |\sigma^\ell|\}$ と定義する．また，$I = \cup_{\ell=1}^L I^\ell$ とする．

列 π を I の要素を昇順に並べた列とする．明らかに π は π_1 の部分列である．各 $\ell \in \{1, \ldots, L\}$ に対して $I \subset I^\ell$ であり，列 σ^ℓ は I^ℓ の要素からなる昇順列または降順列であるから，σ^ℓ は π または π^R の部分列である．

変数集合を $Y = \{x_i \mid i \in I\}$ とし，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．すべての部分割り当て a に対して，部分分岐プログラム $B|_a$ 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．

部分分岐プログラム $B|_a$ は π 及び π^R に従う $(L+1)$-IBDD と考えることができる．補題 7 より $B|_a$ に対する充足可能性問題は $O(poly(n) \cdot (cn/L)^{2L+2})$ 時間，$O(poly(n) \cdot (cn/L)^{L+1})$ 領域で解ける．定理 3 より各 $\ell \in \{1, \ldots, L\}$ に対して，$|I^\ell| \ge \sqrt{n/L}$ である．よって $|I| = \sum_{\ell=1}^L |I^\ell| \ge \sqrt{Ln}$．つまり，$|X\backslash Y|$ は $n - \sqrt{Ln}$ 以下となり，部分割り当て a のパターン数は高々 $2^{n-\sqrt{Ln}}$ である．以上から全体の計算時間は高々 $poly(n) \cdot (cn/L)^{2L+2} \cdot 2^{n-\sqrt{Ln}}$ となる．

以下では定理を満たすためのパラメータ L を与える．新たなパラメータ δ を用いて $L = \delta n$ $(0 < \delta < 1)$ とすると，

$$\begin{aligned}(cn/L)^{2L+1} \cdot 2^{n-\sqrt{Ln}} &= (c/\delta)^{2\delta n+1} \cdot 2^{(1-\sqrt{\delta})n} \\ &= (c/\delta) \cdot 2^{(1-\sqrt{\delta}+2\delta\log(c/\delta))n}(1)\end{aligned}$$

である．$\mu(c) = \sqrt{\delta} - 2\delta\log(c/\delta)$ とすると，$\log\delta + \frac{1}{2\sqrt{\delta}} > \log c$ を満たす定数 δ を選択すれば，$\mu(c)$ は正の定数となる．

$f(\delta) = \log\delta + \frac{1}{2\sqrt{\delta}}$ とすると，

$$\frac{d}{d\delta}f(\delta) = \frac{\log e}{\delta} - \frac{1}{4\delta^{\frac{3}{2}}} = \frac{1}{\delta^{\frac{3}{2}}}\left(\log e\sqrt{\delta} - \frac{1}{4}\right)$$

となる．e は自然対数の底である．$0 < \delta < 1/4\log^2 e$ なる定数 δ に対して，

$$\frac{d}{d\delta}f(\delta) < -2\log^3 e$$

となる．よって，δ を十分小さくとれば $f(\delta)$ を十分大きくとることができ $\mu(c) > 0$ を満たす．

また，$\delta = \frac{1}{16\log^2 c}$ とおくと，十分大きな c について，

$$\mu(c) = \frac{1}{8\log c} - \frac{\log\log c + 2}{4\log^2 c} = \Omega(1/\log c) \qquad (2)$$

となる．($c \ge 2^{16}$ ならば $\mu(c) > 0$ を満たす．) 全体の計算時間は高々 $poly(n) \cdot 2^{(1-\mu(c))n}$ となり，2^n より指数的に高速である．

このとき，計算領域は

$$\begin{aligned}&O\left(poly(n) \cdot (m/L)^{L+1}\right) \\ &= O\left(poly(n) \cdot (16c\log^2 c)^{\frac{n}{16\log^2 c}}\right) \\ &= O\left(poly(n) \cdot 2^{O(n/\log c)}\right)\end{aligned}$$

となる． □

Algorithm 2 のアイデアを拡張することにより，入力が

線形サイズである k-IBDD SAT が 2^n 時間より指数的に高速に解くことができることを示す．k-IBDD SAT を解く決定性アルゴリズムは Algorithm 3 である．また，Algorithm 3 で使用されるサブルーチン $Extraction(\pi, I)$ を Algorithm 4 で与える．$Extraction(\pi, I)$ は列 π と集合 I を引数とし，集合 I の要素のみからなる π の最長部分列 π' を出力する．

Algorithm 3 k-IBDD SAT(B, L)

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

$I_1 := \{1, 2, \ldots, n\}$
for all $h = 2, \ldots, k$ **do**
  π'_h :=Extraction(π_h, I_{h-1})
  for all $\ell \in \{1, \ldots, L\}$ **do**
    $\pi_h^\ell := (\pi'_h(\lceil |I_{h-1}|/L \rceil (\ell-1)+1), \ldots, \pi'_h(\lceil |I_{h-1}|/L \rceil \ell))$ { 列 π'_h を L 分割 }
    π_h^ℓ の最長増加部分列 $\sigma_{h,inc}^\ell$，最長減少部分列 $\sigma_{h,dec}^\ell$ を計算
    if $|\sigma_{h,inc}^\ell| \geq |\sigma_{h,dec}^\ell|$ **then**
      $\sigma_h^\ell := \sigma_{h,inc}^\ell$
    else
      $\sigma_h^\ell := \sigma_{h,dec}^\ell$
    end if
    $I_h^\ell := \{\sigma_h(j) \mid 1 \leq j \leq |\sigma_h|\}$
  end for
  $I_h := \cup_{\ell \in \{1,\ldots,L\}} I_h^\ell$
end for
$\pi := sort(I_k)$ {I_k の要素の昇順列 }
$Y := \{x_i \mid i \in I_k\}$
for all $a \in \{0, 1, *\}^n$，ただし $S(a) = X \backslash Y$ **do**
  if (π, π^R)-k-IBDD SAT($B|_a$) = “Yes” **then**
    return “Yes”
  end if
end for
return “No”

Algorithm 4 Extraction(π, I)

Require: 列 π, 集合 I
Ensure: I の要素からなる順列 π の部分列 π'

$j = 1$
for $i = 1, \ldots, |\pi|$ **do**
  if $\pi(i) \in I$ **then**
    $\pi'(j) := \pi(i)$
    $j := j + 1$
  end if
end for
return π'

定理 9 (再掲：定理 2)．任意の定数 $c > 0$ に対して，$\mu(c) > 0$ を満たす関数 $\mu(c)$ が存在して，n 変数，$m = cn$ ノードの *k-IBDD SAT* を $poly(n) \cdot 2^{(1-\mu(c))n}$ 時間，$poly(n) \cdot 2^{\nu(c)n}$ 領域で解く決定性アルゴリズムが存在する．ただし，$\nu(c) = O\left(\frac{1}{(\log c)^{2^{k-1}-1}}\right)$ であり，$poly(n)$ は n の多項式を表す．また，十分大きな c について $\mu(c) = \Omega\left(\frac{1}{(\log c)^{2^{k-1}-1}}\right)$ である．

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．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 を L_2 分割し，$\pi_2^1, \ldots, \pi_2^L$ とする．つまり，各 $\ell \in \{1, \ldots, L\}$ に対して $\pi_2^\ell := (\pi_2(\lceil n/L \rceil(\ell-1)+1), \ldots, \pi_2(\lceil n/L \rceil \ell))$ とする．列 π_2^ℓ の最長増加部分列を $\sigma_{2,inc}^\ell$，最長減少部分列を $\sigma_{2,dec}^\ell$ とする．このうち長いほうを σ_2^ℓ とし，その長さを $|\sigma_2^\ell|$ とする．$I_2^\ell := \{\sigma_2^\ell(i) \mid 1 \leq i \leq |\sigma_2^\ell|\}$ とし，$I_2 := \cup_{\ell=1}^{L_2} I_2^\ell$ とする．

次に，Extraction(π_3, I_2) により I_2 の要素のみを用いた π_3 の部分列を π'_3 を得る．π'_3 を L 分割し，$\pi_3^1, \ldots, \pi_3^L$ とする．ここで各 $\ell \in \{1, \ldots, L\}$ に対して $\pi_3^\ell := (\pi'_3(\lceil |I_2|/L \rceil(\ell-1)+1), \ldots, \pi'_3(\lceil |I_2|/L \rceil \ell))$ とする．$\pi_3'^\ell$ の最長増加 b 部分列を $\sigma_{3,inc}^\ell$，最長減少部分列を $\sigma_{3,dec}^\ell$ とする．このうち長いほうを σ_3^ℓ とし，その長さを $|\sigma_3^\ell|$ とする．$I_3^\ell := \{\sigma_3^\ell(i) \mid 1 \leq i \leq |\sigma_3^\ell|\}$ とし，$I_3 := \cup_{\ell=1}^{L_3} I_3^\ell$ とする．

同様に各 h $(2 \leq h \leq k)$ に対して，Extraction(π_h, I_{h-1}) により I_{h-1} の要素のみを用いた π_h の部分列を π'_h を得る．π'_h を L 分割し，$\pi_h^1, \ldots, \pi_h^L$ とする．ここで各 $\ell \in \{1, \ldots, L\}$ に対して $\pi_h^\ell := (\pi'_h(\lceil |I_{h-1}|/L \rceil(\ell-1)+1), \ldots, \pi'_h(\lceil |I_{h-1}|/L \rceil \ell))$ とする．$\pi_h'^\ell$ の最長増加部分列を $\sigma_{h,inc}^\ell$，最長減少部分列を $\sigma_{h,dec}^\ell$ とする．このうち長いほうを σ_h^ℓ とし，その長さを $|\sigma_h^\ell|$ とする．$I_h^\ell := \{\sigma_h^\ell(i) \mid 1 \leq i \leq |\sigma_h^\ell|\}$ とし，$I_h := \cup_{\ell=1}^{L_h} I_h^\ell$ とする．

変数集合を $Y = \{x_i \mid i \in I_k\}$ とし，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．

列 π を I_k の要素を昇順に並べた列とする．明らかに π は π_1 の部分列である．各 $h \in \{2, \ldots, k\}, \ell \in \{1, \ldots, L\}$ に対して分割されたレイヤーをそれぞれ第 h-ℓ レイヤーと呼ぶことにする．$I_k \subseteq I_h^\ell$ と σ_i^ℓ が昇順列または降順列であることから，$B|_a$ の第 h-ℓ レイヤーは列 π または列 π^R に従う．よって部分分岐プログラム $B|_a$ は π または π^R に従う $((k-1)L+1)$-IBDD である．補題 7 より $B|_a$ に対する充足可能性問題は $poly(n) \cdot (cn/(k-1)L)^{2(k-1)L}$ 時間，$poly(n) \cdot (cn/(k-1)L)^{(k-1)L}$ 領域で解ける．

よって，全体の計算時間は高々 $poly(n) \cdot (cn/(k-1)L)^{2(k-1)L} \cdot 2^{n-|I_k|}$ である．また，計算領域は $poly(n) \cdot (cn/(k-1)L)^{(k-1)L}$ である．

定理 8 の証明と同様に $|I_2| = \sum_{\ell=1}^{L} |I_2^\ell| \geq \sqrt{Ln}$ である．また同様にして，すべての $2 \leq h \leq k$ に対して $|I_h^\ell| \geq \sqrt{|I_{h-1}|/L}$ を満たし，$|I_h| = \sum_{\ell=1}^{L} |I_h^\ell| \geq \sqrt{L|I_{h-1}|}$ である．よって，帰納的に

$$|I_k| \geq L^{1-\frac{1}{2^{k-1}}} \times n^{\frac{1}{2^{k-1}}}$$

となる．

ここで，$L = \delta n$ $(0 < \delta < 1)$ とすると，

$$\begin{aligned}
&\left(\frac{cn}{(k-1)L}\right)^{2(k-1)L} \cdot 2^{n-|I_k|} \\
&\leq \left(\frac{c}{(k-1)\delta}\right)^{2(k-1)\delta n} \cdot 2^{\left(1-\delta^{1-\frac{1}{2^{k-1}}}\right)n} \\
&\leq 2^{\left(1-\delta^{1-\frac{1}{2^{k-1}}}+2(k-1)\delta\log\frac{c}{(k-1)\delta}\right)n} \qquad (3)
\end{aligned}$$

である．

$$\mu(c) = \delta^{1-\frac{1}{2^{k-1}}} - 2(k-1)\delta\log\frac{c}{(k-1)\delta}$$

とすると，

$$\log\left((k-1)\delta\right) + \frac{1}{2(k-1)\cdot\delta^{\frac{1}{2^{k-1}}}} > \log c$$

を満たす定数 δ を選択すれば，$\mu(c)$ は正の定数となる．$f(\delta) = \log\left((k-1)\delta\right) + \frac{1}{2(k-1)\cdot\delta^{\frac{1}{2^{k-1}}}}$ とすると，

$$\begin{aligned}
\frac{d}{d\delta}f(\delta) &= \frac{\log e}{\delta} - \frac{1}{2^k(k-1)\cdot\delta^{1+\frac{1}{2^{k-1}}}} \\
&= \frac{1}{\delta^{1+\frac{1}{2^{k-1}}}}\left(\log e\cdot\delta^{\frac{1}{2^{k-1}}} - \frac{1}{2^k(k-1)}\right) \qquad (4)
\end{aligned}$$

となる．e は自然対数の底である．$0 < \delta < (1/2^k(k-1)\log e)^{2^{k-1}}$ なる δ に対して $\frac{d}{d\delta}f(\delta) < 0$ であり，δ を十分小さくとれば $f(\delta)$ を十分大きくなり $\mu(c) > 0$ を満たす．

また，$\delta = \frac{1}{\beta\cdot(\log c)^{2^{k-1}}}$ とおくと，

$$\begin{aligned}
\mu(c) &= \frac{(\beta^{\frac{1}{2^{k-1}}} - 2(k-1))}{\beta\cdot(\log c)^{2^{k-1}-1}} \\
&\quad + \frac{2(k-1)}{\beta}\cdot\frac{\log\frac{k-1}{\beta} - 2^{k-1}\log\log c}{(\log c)^{2^{k-1}}} \qquad (5)
\end{aligned}$$

となる．$\beta > (2(k-1))^{2^{k-1}}$ を満たす β を選択することにより，十分大きな c について，$\mu(c) = \Omega\left(\frac{1}{(\log c)^{2^{k-1}-1}}\right)$ となる．このとき，計算領域は $poly(n)\cdot 2^{\nu(c)n}$ となる．ただし，$\nu(c) = O\left(\frac{1}{(\log c)^{2^{k-1}}}\right)$ である．

以上から定理は示された． □

謝辞 本研究は MEXT 科研費 24106003，JSPS 科研費 26730007, JSPS 特別研究員奨励費および 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.

[6] 脊戸和寿, 照山順一, 長尾篤樹. k-IBDD 充足可能性問題に対する厳密アルゴリズム, 情報処理学会研究報告, Vol.2014-AL-149, No.9, pp.1–6, 2014.