Vol.2014-AL-148 No.19
2014/6/14

ZDDを用いた極大集合列挙アルゴリズム

菅谷　輝治[1,a)]

概要：本稿では, データ構造 ZDD を用いたシンプルな極大集合列挙アルゴリズムについて紹介する. ZDD を用いた極大集合列挙問題のアルゴリズムの先行研究としては,Knuth が教科書で紹介したものがある. また, 実動作による評価まで行った論文としては Bayardo によるものなどが報告されている. 本稿では,ZDD による新たな極大集合列挙アルゴリズムを紹介する一方, これらの先行研究と実動作で比較し検討を行う.

キーワード：極大集合, 列挙アルゴリズム,ZDD

New algorithm for finding extremal sets with ZDD

Abstract: In this paper we propose a simple algorithm for finding maximal sets. Among the preceding algorithms for identifying maximal sets which make use of ZDD, that of Knuth is known which is written in his text book. Other than that, several practical insights are known, for example, that of Bayrardo or Eén. In this paper, we propose a new algorithm for finding maximal sets then conduct computational experiments in comparison with the preceding algorithms.

Keywords: Extremal sets, Enumeration, ZDD

1. はじめに

本研究では,ZDD というデータ構造を用いたハイパーグラフ・アルゴリズム研究の一例として, 極大集合列挙問題を扱う.

ZDD は湊 [1] によって提案されたデータ構造である. 当初は LSI CAD の分野で用いられるものであったが, 近年ではグラフ理論への応用なども検討されている.

一方, 極大集合列挙問題は充足可能性問題（Satisfiability Problem, SAT）などへの応用が知られている. また, 実動作まで含めたアルゴリズムとしては Bayardo や Eén によるものなどが報告されている. 特に ZDD を用いた極大集合列挙アルゴリズムの先行研究としては,Knuth [2] によるものがある.

本稿では, グラフ理論の分野では比較的新しいデータ構造である ZDD を用い, 極大集合列挙問題を対象に新規手法を提案することを試みる.

2 章では,ZDD と極大集合列挙問題に関する先行研究について記述する. 3 章では, 本稿で用いる概念と定義について説明し, ついで提案手法とその元となった HTCBDD アルゴリズムについて記述する. 4 章では, 評価結果について論じ,5 章で結論を述べる.

2. 先行研究

2.1 BDD と ZDD

2.1.1 真理値表と集合族・二分決定図

集合は, 論理演算で表すことができる. 例えば “0” を集合の要素が “ない”,“1” を “ある” と考えると, **図 1** の真理値表は $\{\{Q\},\{P,Q\}\}$ という集合族を表すとみなすことができるからである.

また, 真理値表は**図 2** のように二分決定図で表現することが可能である. ここで破線は “0” を, 実線は “1” を表す. つまり集合族との関係では破線は “ない” に, 実線は “ある” に相当している. 出口節点は

1 放送大学大学院
The Open University of Japan, School of Graduate Studies
a) 1228226851@campus.ouj.ac.jp

ⓒ 2014 Information Processing Society of Japan

演算	説明
Empty()	$\emptyset(\bot)$ を返す
Base()	$\{0\}(\top)$ を返す
Subset1(P,var)	$var = 1$ であるような枝を返す
Subset0(P.var)	$var = 0$ であるような枝を返す
Change(P,Var)	var を反転させた P を返す
Union(P,Q)	$(P \cup Q)$ を返す
IntSec(P,Q)	$(P \cap Q)$ を返す
Diff(P,Q)	$(P - Q)$ を返す
Count(P)	P の要素数 $\|P\|$ を返す

表 1 ZDD の基本演算
Table 1 Primitive ZDD Operations

False ⊥ と True ⊤ からなる. 真理値表での組み合わせが存在しない場合は ⊥ に, 存在する場合は ⊤ に接続させる.

二分決定図は扱う台集合のサイズの二乗に比例して大きくなるという性質を持っているため, 実際に計算に用いるには圧縮する必要がある. そのためにいくつかの圧縮手法が知られているが, ここでは BDD とその発展形である ZDD について述べる.

2.1.2 BDD

Binary Decision Diagrams（以下 BDD）は [3] で提案され, [4] で演算方法が整理された.

BDD は二分決定図を圧縮するために共有化と縮約という 2 つの技法を用いる.（**図 3・図 4**）

共有化は重複したノードをまとめる手法である. 図 3 の左側の二分決定図では Q を表すノードが 2 つあるが, それらはともに破線が True, 実線が False に接地している. そのため, これらのノードは図の右側のように共有化することができる.

縮約は, 破線・実線が同じノードとつながったら削除するという手法である. 図 4 の左側の二分決定図では P から派生する破線・実線がともに Q につながっている. そのため, これらのノードは図の右側のように縮約することができる.

BDD では, 圧縮したまま演算処理を行うことができるという特長がある. [4]

2.1.3 ZDD

Zero-suppressed Binary Decision Diagrams（以下 ZDD）は [1] で提案された.

ZDD は BDD と同様に共有化の手法を用いるが, 縮約の方法が異なり, 実線がゼロに直結するノードは削除する. **図 5** の左側の二分決定図では,Q のノードはいずれも 1 枝が False に設置しているため, 右のように P ノードのみ記述すれば足りるという手法である. この図 5 からもわかるように,BDD の縮約規則は ZDD では適用されない.

ZDD でも BDD 同様, 圧縮したまま演算を行うことができる. それらを**表 1** に示す. [1]

2.2 極大集合列挙問題

ここでは先行研究について簡単に記述する.

2.2.1 一般的な比較によるもの

最もナイーブに集合同士を比較した場合, 集合の数を N とすると $O(N^2)$ の比較が必要となってしまうため, 探索空間を縮小する必要がある. 探索空間を狭めるための工夫は Prichard の初期の論文に既に明らかにされている. [5], [6], [7].

本章では以下, 台集合 $\{1, ..., M\}$ と, その上にある集合族 $\{X_1, ..., X_N\}$ について解説する.

(1) 含まれる要素ごとに集合を分類

２つの集合間に包含関係がある場合は, かならず両者は共通の要素を持つ. このことを利用して集合を分類しなおし, 探索空間を小さくする.

(2) 集合の大きさに注目

集合同士が包含関係を持つときは,“包含される集合” は “包含する集合” よりも大きさが小さい. このことを利用して比較回数を減らす.

(3) 辞書順を利用する

集合族を辞書順に並べると, 集合に真の包含関係がある場合 $X \subset Y$ には, 次のいずれかが成り立つ. [5]

- X は Y の真の接頭辞となる.
- X は Y より辞書順で後となる.

このことを利用するため, 前処理として集合を辞書順にソートしておいてから探索する.

なお (1)・(2), ないし (1)・(3) は, それぞれ同時に組み合わせて適用することが可能である.

Bayardo らは前述の (1)〜(3) の手法にさらに独自のヒューリスティックスを加え, 各手法を比較している. 最も高速なものは辞書順を利用した AMS-Lex であると報告されている. 処理時間は入力集合中の総要素数を m として $O(m^2/log(m))$ とされる [8]. すなわち, 極大集合列挙問題は多項式時間アルゴリズムの存在する P 問題に属する.

実装も公開されている [9].

2.2.2 ZDD によるもの

ZDD による極大（小）集合列挙問題のアルゴリズムとしては,Knuth によるものがあり, [2] の演習問題 237 に記述されている.

アルゴリズム 1 に記述する.

ここで $NONSUB$ というのは,$NONSUB(p, q) = \{\alpha \in p | \beta \in q$ は　$\alpha \not\subseteq \beta$ を含意する $\}$ という処理である. すなわち,q の要素集合のうち,p の要素集合を含まない集合族 $nonsubset$ を q から抽出して求める

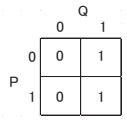


図 1 真理値表と集合
Fig. 1 Truth table and Sets

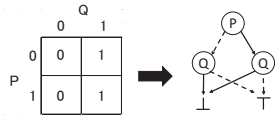


図 2 真理値表と二分決定図
Fig. 2 Truth table and Binary Decision Diagram

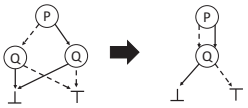


図 3 共有化
Fig. 3 Node Sharing

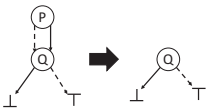


図 4 縮約
Fig. 4 Node Elimination

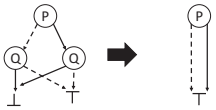


図 5 縮約 (ZDD)
Fig. 5 Node Elimination(ZDD)

アルゴリズム名	探索手法
AMS-SAT	(1) のみ
AMS-Card	(1)・(2)
AMS-Lex	(1)・(3)

表 2 Bayardo らのアルゴリズム
Table 2 Algorithm of Bayardo et al

処理である.

MAX_K の再帰呼び出しの各段で $NONSUB$ を求めるための処理が呼び出されるため, 処理性能には限界がある.

Algorithm 1 MAX_K(p) ZDD p を入力として,p の極大集合全体を計算する.

Input: $ZDDp$
1: **if** $q = \bot_{ZDD}$ *or* $q = \top_{ZDD}$ **then**
2: **return** q;
3: **end if**
4: $hl \leftarrow MAX_K(LO(p))$; $h \leftarrow MAX_K(HI(p))$ $hh \leftarrow NONSUB(h, hl)$;
5: $q \leftarrow ZDD_UNIQUE(V(p), hl, hh)$;
6: **return** q;

3. 提案手法

ここでは, ハイパーグラフの極大集合を列挙する ZDD を用いた新しい手法を提案する. まず問題の数学的な定義を整理する. ついで提案手法の元となる戸田によるヒッティング集合列挙アルゴリズムを紹介する. 以上を踏まえた上で, 本稿の提案手法の説明を行う.

3.1 概念と定義

まず, 本稿で取り扱う問題を扱うために必要な概念と定義を導入する.

3.1.1 BDD と ZDD

BDD と ZDD は, 根と呼ばれる一つの節点より派生する 2 分木である. 各節点は (V, LO, HI) とよばれる 3 つのデータからなる.[*1]

V は変数のインデックス,LO と HI は他の節点へのポインタである. LO と HI の示す先がそれぞれ LO 節点・HI 節点と呼ばれ,LO 節点への有向辺を LO 枝と呼び,HI 節点への有向辺を HI 枝と呼ぶ. 図示する際には,LO 枝を破線で,HI 枝を実線で表す.

終端節点には $\bot$ と $\top$ がある.BDD と ZDD を区別するためにそれぞれ $\bot_{BDD}$ と $\top_{BDD}$,$\bot_{ZDD}$ と $\top_{ZDD}$ と表すこともある.

BDD 及び ZDD は次の 2 つの条件を満たしている必要がある.

(1) 順序付きであること
(2) 既約であること

順序付きであるとは,u が v を差す場合には $V(u) < V(V)$ を満たさなければならないということである.

既約であることとは, これ以上適用できなくなるまで BDD・ZDD の各圧縮規則が適用されているということである. すなわち, 共有化と縮約を適用する余地がなくなっていることである.

BDD 及び ZDD では, 節点 p に対して 3 つ組 $V(p), LO(p), HI(p)$ が一意に定まるようにハッシュ表で管理する. そして関数 BDD_UNIQUE ない

*1 この章の記述は [10] を参考にした

ⓒ 2014 Information Processing Society of Japan

し ZDD_UNIQUE を用意し, インデックス・LO 節点・HI 節点からなる 3 つ組 (k, l, h) が入力として与えられると, ハッシュ表に登録されている場合はその節点を返し, 登録されていない場合は登録して $V(p) = k, LO(p) = l, HI(p) = h$ を満たす新たな節点を返すようにする. このことにより, 等価な節点の重複を避ける一方で節点の等価性判定が定数時間で実行できるようになる.

BDD ないし ZDD の任意の節点に対して, それを根とする部分グラフも BDD ないし ZDD となる. 節点 $\{x\}$ を根とする場合は $BDDx$・$ZDDx$ と表す. 集合族を表す場合は,$\mathcal{S}(x)$ が $BDDx$・$ZDDx$ に対応する集合族となる.

3.1.2 ハイパーグラフと極大（小）集合列挙

ハイパーグラフとは台集合上の集合の集まりであり, 台集合 V と集合族 $\mathcal{E}$ の対 $(V, \mathcal{E})$ として与えられる. $\mathcal{E}$ の元は V の部分集合であり, ハイパーエッジと呼ばれる. 冪集合とは, 台集合によって作られるすべての集合を含んだ集合族である.

集合族に含まれる集合の数を集合数, 各集合の中の要素数をすべて合計したものを総要素数と呼ぶことにする.

集合族 $\mathcal{A}$ の要素となる集合のうち, 他の集合を包含することのない集合のみ集めたものを極小集合と呼ぶ. 同様に, 集合族 $\mathcal{A}$ の要素となる集合のうち, 他の集合に包含されることのない集合のみ集めたものを極大集合と呼ぶ.

本稿で研究対象とする極大集合列挙アルゴリズムとは, 与えられた集合族のうち極大部分のみを抽出するものである.

台集合 V 上の集合族 $\mathcal{A}$ において,$A \subset B$ かつ $A \in \mathcal{A}$ ならば $B \in \mathcal{A}$ がなりたつとき,$\mathcal{A}$ を “上に閉じた” 集合 Upset という.

同様に, 台集合 V 上の集合族 $\mathcal{A}$ において,$B \subset A$ かつ $A \in \mathcal{A}$ ならば $B \in \mathcal{A}$ がなりたつとき,$\mathcal{A}$ を “下に閉じた” 集合 Downset という [11].

例えば, ハイパーグラフ $(V, \mathcal{E})$ が $V = \{1, ..., 3\}$, $\mathcal{E} = \{1, 2\}, \{2, 3\}, \{2\}$ とする. ここで, 集合数は 3, 総要素数は 5 となる. この時 $\mathcal{E}$ に適当な集合を加えて, 下に閉じた集合にすると $\mathcal{E}' = \{1, 2\}, \{2, 3\}, \{1\}, \{2\}, \{3\}$ となる. 極大集合族は $\{1, 2\}, \{2, 3\}$ である. また同じく適当な集合を加えて上に閉じた集合にすると $\mathcal{E}' = \{1, 2, 3\}, \{1, 2\}, \{2, 3\}, \{2\}$ となり, 極小集合族は $\{2\}$ である.

$\mathcal{E}$ のヒッティング集合とは, V の部分集合であり, すべてのハイパーエッジと “交差する” ものである. すなわち, すべての $U \in \mathcal{E}$ に対して $U \cap T \neq 0$ を満たす集合 T である. ヒッティング集合を包含する集合は明らかにヒッティング集合であるが, ヒッティング集合の部分集合はヒッティング集合とは限らない. ヒッティング集合が他のヒッティング集合を包含しないとき, 極小ヒッティング集合と呼ぶ.

例えば, ハイパーグラフ $(V, \mathcal{E})$ が台集合 $V = \{1, ..., 6\}$, 集合族 $\mathcal{E} = \{1, 2, 3\}, \{3, 4\}, \{5\}, \{5, 6\}$ のとき, 極小ヒッティング集合は $\{1, 4, 5\}, \{2, 4, 5\}, \{3, 5\}$ の 3 つである.

3.2 ヒッティング集合列挙アルゴリズム HTCBDD

戸田は [12], [10] において,ZDD を用いヒッティング集合列挙アルゴリズム Hypergraph Transversal Computation with Binary Decision Diagrams を提案した. このアルゴリズムを以下 HTCBDD と呼ぶ.

この手法は次の 4 段階に分かれる.

(1) 入力の集合族 $\mathcal{E}$ を圧縮して, 対応する ZDD p を求める.
(2) $\mathcal{S}(p)$ のヒッティング集合全体からなる BDD q を計算する.
(3) $\mathcal{S}(q)$ における極小集合全体からなる ZDD r を計算する.
(4) ZDD r を解凍して, 対応する集合族を $\mathcal{E}^*$ 出力する.

このうち 1. と 4. については [13], [14] で提案された圧縮・解凍手法を用いている.

1. の圧縮では, まず入力となる集合族にマルチキー・クイックソート [15] を昇順で施し, その後に二分割法アルゴリズムを実行する.

マルチキー・クイックソート部分は $cn(logn + k)$ 回の比較で行えることが知られている.

次の二分割法部分の計算量が $O(Nlog^2(\Sigma_{U \in \mathcal{E}}|U|/N))$ となる.

2. によって “上方に閉じた” ヒッティング集合全体を得た後, 後段の 3. で極小部分のみを抽出する. 一旦 “上方に閉じた” 集合族にすることで, 極小部分の抽出を行うことが可能となる.

このように HTCBDD の特徴は,p の極小ヒッティング集合を求めるために一旦すべてのヒッティング集合を列挙する点にある.

4. の伸長は,ZDD の根から終端節点までの道をたどりながら出力する. なぜなら, これらの道は各々 ZDD に圧縮された集合に対応するからである. したがって 4. の計算量は $O(|V|\dot{|}\mathcal{E}^*|)$ となる. ただし $\mathcal{E}^*$ は出力集合族を表す.

戸田はこのアルゴリズムが, 先行する [16] や Knuth

Algorithm 2 DOWNSET(p) 入力集合の ZDD p をもとに下方に閉じた集合族 ZDD q を生成する

Input: 入力集合族の $ZDDp$
Output: 下方に閉じた集合族 ZDDq
1: **if** $p = \bot_{ZDD}$ **then**
2: **return** $\bot_{ZDD}$;
3: **end if**
4: **if** $p = \top_{ZDD}$ **then**
5: **return** $\top_{ZDD}$;
6: **end if**
7: $mh \leftarrow DOWNSET(HI(p))$; $ml \leftarrow DOWNSET(LO(p))$;
8: $t \leftarrow UNION(mh, ml)$;
9: $ZDD_UNIQUE(V(q), t, mh)$;
10: **return** r;

Algorithm 3 MAX(q) $\mathcal{S}(q)$ が下方に閉じた ZDD q をもとに, 極大集合全体からなる ZDD r を計算する

Input: 下方に閉じた ZDD q
Output: 極大集合全体 ZDD r
1: **if** $q = \bot_{ZDD}$ **then**
2: **return** $\bot_{ZDD}$;
3: **end if**
4: **if** $q = \top_{ZDD}$ **then**
5: **return** $\top_{ZDD}$;
6: **end if**
7: $mh \leftarrow MAX(HI(q))$; $ml \leftarrow MAX(LO(q))$;
8: $t \leftarrow DIFF(ml, mh)$;
9: $r \leftarrow ZDD_UNIQUE(V(q), t, mh)$;
10: **return** r;

によるヒッティング集合列挙アルゴリズム [2] と比較して, 特に疎なデータで HTCBDD が高速に動作することを示している [12].

3.3 提案手法

3.3.1 極大集合の列挙

提案手法では,HTCBDD を変形させて, 極大集合列挙問題に用いることを試みる.

HTCBDD は前述のように, 上方に閉じたヒッティング集合全体をまず列挙した後, その中から極小部分のみを抽出することで極小ヒッティング集合の列挙を実現した.

提案手法では, まず入力となる集合族に新たな集合を追加することで下方に閉じた集合族に変え, ついで極大部分抽出処理に通すことで極大集合を列挙する.

つまり, 提案手法による極大集合列挙アルゴリズムは次のような手順になる. 2 及び 3 の部分が HTCBDD との差分となる.

(1) 入力の集合族 $\mathcal{E}$ を圧縮して, 対応する ZDD p を求める.
(2) **$\mathcal{S}(p)$ の集合族に必要な集合を加えて, 下に閉じた集合族 ZDD q を計算する.**
(3) **$\mathcal{S}(q)$ における極大集合全体からなる ZDD r を計算する.**
(4) ZDD r を解凍して, 対応する集合族を $\mathcal{E}^*$ 出力する.

2 番めの, 下に閉じた集合族をつくるアルゴリズムを Algorithm2 に示す.

3 番めの, 極大集合を抽出するアルゴリズムを Algorithym3 に示す.

3.3.2 極大集合列挙アルゴリズムの正当性・性能

提案手法の手順のうち, 1・4 の計算量については 3.2 で記述した.

手順の 2・3 にあたるアルゴリズム 2,3 について述べる.

定理 3.1. アルゴリズム *2* は入力となる集合族を含む下に閉じた集合族を出力する

証明. p が終端節点の場合は $p = q$ とする. ここで,“$B \subset A$ かつ $A \in S(q)$ ならば $B \in S(q)$” を証明する必要がある.

入力 $ZDDp$ の構造に関する帰納法で示す.

p が終端節点の場合はただちに確認できる.

終端節点以外の場合について,pl := LO(p) ; ph := HI (p) と表す.

$DOWNSET(q)$ は $DOWNSET(pl) \cup DOWNSET(ph)$ と $\{\{V(p)\} \cup \mathcal{U} : \mathcal{U} \in DOWNSET(ph)\}$ の直和であるため, A が含まれる部分で場合分けすると以下の 3 つに分かれる.

(1) $A \in DOWNSET(pl)$ の場合

帰納法の前提より,$B \subset A$ ならば $B \in DOWNSET(pl)$. したがって,$B \in (DOWNSET(pl) \cup DOWNSET(ph))$.

(2) $A \in DOWNSET(ph)$ の場合

(1) と同様に,$B \in (DOWNSET(pl) \cup DOWNSET(ph))$ が示される.

(3) $A \in (\{\{V(p)\} \cup \mathcal{U} : \mathcal{U} \in DOWNSET(ph)\})$

これはさらに 2 つに場合分けされる.

(3-1) A, B ともに $V(p)$ を含む場合

$A' = A - \{V(p)\}$, $B' = B - \{V(p)\}$ とする.

$B' \subset A'$ であり, 当然 $A' \in DOWNSET(ph)$ であることから, 帰納法の前提より $B' \in DOWNSET(ph)$. よって $B \in (\{\{V(p)\} \cup \mathcal{U} : \mathcal{U} \in DOWNSET(ph)\})$

(3-2) A は $V(p)$ を含み,B は $V(p)$ を含まない場合

$A' = A - \{V(p)\}$ とする. 当然 $A' \in DOWNSET(ph)$ である.

$B \subset A$ かつ $V(p) \notin B$ より $B \subset A'$. よって帰納法の前提より $B \subset DOWNSET(ph)$ となるため,

$B \in (\{\{V(p)\} \cup \mathcal{U} : \mathcal{U} \in DOWNSET(ph)\})$ となる.

以上により,“$B \subset A$ かつ $A \in S(q)$ ならば $B \in S(q)$” が示された.

□

定理 3.2. アルゴリズム *2* は $O(|p| \cdot N(p)^2)$ 時間で計算できる. ここで,$N(p) := max\{|DOWNSET(p')| : p'$ は p の部分 $ZDD\}$

証明は, [10] の定理 1. のそれと同等であるため, 省略する.

定理 3.3. アルゴリズム *3* は入力となる集合族の極大集合を抽出する.

証明. q が終端節点の場合は $q = r$ とする.

入力 $ZDDq$ の構造に関する帰納法で示す.

q が終端節点の場合はただちに確認できる.

終端節点以外の場合について,ql := LO(q) ; qh := HI (q) と表す.

全ての $A \in qh$ は $\{\{V(p)\} \cup A\}$ の部分集合である. 入力集合は下に閉じた集合であることから,$A \in ql$ となる. 従って $ql \supset qh$.

帰納法の前提より,$MAX(ql)$ 及び $MAX(qh)$ にはそれぞれ ql 及び qh の極大集合が抽出される.

よって,$MAX(ql) \supset MAX(qh)$ となる. 従って,$DIFF(MAX(ql), MAX(qh))$ には,$\mathcal{S}(q)$ の極大集合のうち $V(q)$ を含まないものが抽出されている.

また,$MAX(qh)$ には qh の極大集合が抽出されることから, $\{\{V(q)\} \cup \mathcal{U} : \mathcal{U} \in MAX(qh)\}$ には $\mathcal{S}(q)$ の極大集合のうち $V(q)$ を含むものが抽出されている.

r は $MAX(ql) \setminus MAX(qh)$ と $\{\{V(q)\} \cup \mathcal{U} : \mathcal{U} \in MAX(qh)\}$ の直和であることから, r には $\mathcal{S}(q)$ の極大集合が全て抽出される.

□

定理 3.4. アルゴリズム *3* は $O(|q| \cdot L(q)^2)$ 時間で計算できる. ここで,$L(q) := max\{|MAX(q')| : q'$ は p の部分 $ZDD\}$

この証明も, [10] の定理 1. のそれと同等であるため, 省略する.

4. 評価

本章では, 実動作によるアルゴリズムの評価を行う. 以下では提案手法を DownSet Differentiation と名付け, 略して DSD と呼ぶことにする. DSD と Bayardo らの AMS-Lexcographic 及び Knuth の手法を様々なデータセットで動作比較する.

4.1 実装と環境

提案アルゴリズムと Knuth のアルゴリズムは C 言語で実装した. 本実験では, 湊により開発された BDD パッケージ (BDD Package SAPPORO-Edition-1.0) を使用した. 戸田による HTCBDD の実装 [17] 及び ZCOMP の実装 [18] をベースに提案手法を実装している. Bayardo の実装は google-extremal-sets [9] から入手した. 使用した計算機環境は,Intel(R) Core(TM)2 Duo CPU T9550 @2.66GHz,2GB RAM,Ubuntu 14.04,gcc version 4.8.2 である.

なお,Bayardo のオリジナルの実装は極大集合数を数上げるのみで集合を列挙しないが, 提案手法と出力を比較するために答えの列挙を行うように改造した. また,HTCBDD や ZCOMP のオリジナルではマルチキーソート処理がアルゴリズム内に含まれているが, Bayardo の実装ではアルゴリズムの外でソートしてから入力するため, 提案手法の実装でも入力はあらかじめソートするようにして提案アルゴリズム内ではソートしないようにした. その他 BDD パッケージの初期化処理で, 台集合のサイズの 2 乗に比例して処理時間のかかる部分があったため, サイズに比例した処理時間となるような改造を行っている.

4.2 データセット

本稿では, 次のようなデータを用いて評価を行う. データセット 1 及び 2 は, 後述するデータ仕様で自作した. また, データセット 3 及び 4 は, 公開サイトから入手したものから生成した.

4.2.1 データセット 1

台集合 $\{1, ..., n\}$ よりなる冪集合のうち一定割合を抽出して得られる集合族を入力とする. ここでは, 台集合を $\{1, ..., 23\}$ に固定し, 冪集合のうちそれぞれ $25\%, 50\%, 75\%, 100\%$ を取り出してデータセットを構成した（**表 3**）.

集合数に比べて ZDD のノード数が少ないデータセットであり,ZDD としては圧縮効率がよいといえる. なおノード数は ZCOMP [18] を用いて得られる ZDD 形式をファイル出力した時の行数で数えており, 以下のデータセットでも同様である.

4.2.2 データセット 2

集合 $\{1, ..., n\}$ 及びサイズ $\lfloor n/2 \rfloor$ の部分集合からなる集合族を入力とする. ここでは,n として $\{21, 22, 23, 24, 25\}$ を選び, サイズ $\lfloor n/2 \rfloor$ の集合族については 99999 個固定とした（**表 4**）.

データセット 1 とは逆に, 集合数に比べて ZDD ノード数が多いデータセットであり,ZDD としては圧縮効率が悪いといえる.

4.2.3 データセット 3 及び 4

Frequent Itemset Mining Dataset Repository [19]

割合	集合数	ノード数	極大集合数
25%	2097157	712	1
50%	4194307	761	1
75%	6291457	728	1
100%	8388607	45	1

表 3 データセット 1
Table 3 Dataset1

n	集合数	ノード数	極大集合数
21	100000	37771	1
22	100000	51010	1
23	100000	62460	1
24	100000	75378	1
25	100000	85995	1

表 4 データセット 2
Table 4 Dataset2

頻出集合数	集合数	ノード数	極大集合数
1200	9759966	23889	52056
1600	1261227	5514	11209
2000	166580	1324	2419

表 5 データセット 3/chess
Table 5 Dataset3/chess

頻出集合数	集合数	ノード数	極大集合数
800	478374	21028	27066
1000	65236	13636	21692
1200	19412	7460	9642

表 6 データセット 4/T40I10D100K
Table 6 Dataset4/T40I10D100K

から入手した実データを用い, 頻出集合を列挙したものを入力とする. ここでは,chess 及び T40I10D100K を用い, **表 5・表 6** に示した頻出集合を発生させて入力とした. 頻出集合列挙には LCM（Linear time Closed itemset Miner） ver.2 [20] を用いた.

ZDD のノード数を集合数を比較すると, データセット 3 はデータセット 4 に比べて ZDD の圧縮効率がよいことが分かる.

4.3 結果

図 6 及び**図 7** に評価結果を示す. すべてのデータについて, 提案手法・AMS・Knuth アルゴリズムの出力が等しくなることを確認した.

4.3.1 データセット 1

データセット 1 では実行時間は提案手法が最も短い. メモリ使用量では提案手法と AMS はほぼ同等で,Knuth の手法が大きくなっている. 提案手法は指数時間アルゴリズムであるが, 圧縮効率のよいデータセット 1 のような入力の場合にはその欠点にも関わらず提案手法よりも高速に動作できることが分かる.

4.3.2 データセット 2

データ・セット 2 では実行時間・メモリ使用量とも AMS が最も効率がよい. [8] に示されているように,AMS は集合数が一定である場合は, 処理時間が一定となる. ついで提案手法・Knuth のアルゴリズムの順となる. データセット 2 のような圧縮効率の悪いアルゴリズムの場合には, 処理に指数時間がかかってしまうことが明らかである.

4.3.3 データセット 3 及び 4

データセット 3 及び 4 の実行時間は, データ量が大きい入力ほど提案手法のほうが AMS よりも短い. Knuth の手法は実行時間・メモリ使用量とも最も効率がよくない. データセット 3 及び 4 を比較するとデータセット 3 でより顕著に提案手法が高速に動作できていることから, やはり ZDD の圧縮効率のよいデータセットにおいて提案手法は高速に動作することが分かる.

5. まとめ

本稿では, 極大集合列挙のための新しいアルゴリズムを提案した. そして, 先行研究と実動作での比較評価を行った. 極大集合列挙問題は多項式時間アルゴリズムが知られていることから, 最悪で指数時間を要する提案手法は改善の余地が大きい. しかし, 集合数に比べて ZDD にした時のノード数が少なくてすむ圧縮効率のよいデータセットでは, 多項式時間アルゴリズムを実装した従来手法に比べても高速な動作が可

ⓒ 2014 Information Processing Society of Japan

図 6 実行時間
Fig. 6 CPU Time

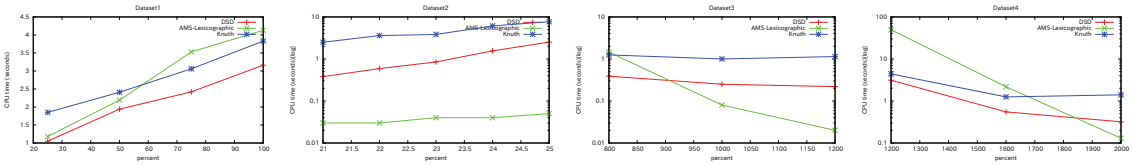
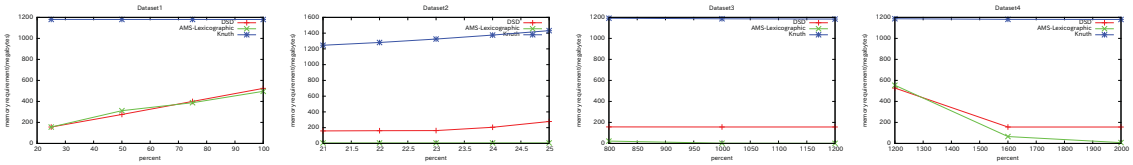


図 7 メモリ使用量
Fig. 7 Memory Requirement



能であることが示された.

今後の課題は, 極大集合列挙問題でのより効率のよい解法の模索,NP 困難な問題への類似手法の応用である.

参考文献

[1] Minato, S.-i.: Zero-suppressed BDDs for set manipulation in combinatorial problems, *Design Automation, 1993. 30th Conference on*, IEEE, pp. 272–277 (1993).

[2] Knuth, D. E.: *Art of Computer Programming, Volume 4, Fascicle 4, The: Generating All Trees–History of Combinatorial Generation*, Addison-Wesley Professional (2013).

[3] Akers, S. B.: Binary decision diagrams, *Computers, IEEE Transactions on*, Vol. 100, No. 6, pp. 509–516 (1978).

[4] Bryant, R. E.: Graph-based algorithms for boolean function manipulation, *Computers, IEEE Transactions on*, Vol. 100, No. 8, pp. 677–691 (1986).

[5] Pritchard, P.: Opportunistic algorithms for eliminating supersets, *Acta Informatica*, Vol. 28, No. 8, pp. 733–754 (1991).

[6] Pritchard, P.: A simple sub-quadratic algorithm for computing the subset partial order, *Information processing letters*, Vol. 56, No. 6, pp. 337–341 (1995).

[7] Pritchard, P.: An old sub-quadratic algorithm for rinding extremal sets, *Information Processing Letters*, Vol. 62, No. 6, pp. 329–334 (1997).

[8] Bayardo, R. J. and Panda, B.: Fast Algorithms for Finding Extremal Sets., *SDM*, SIAM, pp. 25–34 (2011).

[9] Bayardo, R. J.: google-extremal-sets, Bayardo, Roberto J (online), available from ⟨http://google-extremal-sets.googlecode.com/svn/trunk/ google-extremal-sets-read-only⟩ (accessed 2014-01-01).

[10] 戸田貴久, 湊真一：二分決定グラフに基づく大規模ハイパーグラフの双対化とその応用, 2013 年度人工知能学会全国大会（第 27 回）, Vol. 2013 (2013).

[11] Anderson, I.: *Combinatorics of finite sets*, Courier Dover Publications (1987).

[12] Toda, T.: Hypergraph transversal computation with binary decision diagrams, *Experimental Algorithms*, Springer, pp. 91–102 (2013).

[13] Toda, T.: Fast Construction of ZDDs from Large-scale Hypergraphs, Technical report, Technical Report A-13-62, Division of Computer Science, Hokkaido University (2013).

[14] 戸田貴久：大規模ハイパーグラフから ZDD の高速な構築アルゴリズム, 情報処理学会研究報告. AL, アルゴリズム研究会報告, Vol. 2013, No. 1, pp. 1–6 (2013).

[15] Bentley, J. L. and Sedgewick, R.: Fast algorithms for sorting and searching strings, *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, pp. 360–369 (1997).

[16] Murakami, K. and Uno, T.: Efficient algorithms for dualizing large-scale hypergraphs., *ALENEX*, SIAM, pp. 1–13 (2013).

[17] Toda, T.: HTC-BDD: Hypergraph Transversal Computation with Binary Decision Diagrams, Toda, Takahisa (online), available from ⟨http://kuma-san.net/htcbdd.html⟩ (accessed 2014-01-01).

[18] Toda, T.: COMP: Fast Compression of Hypergraphs into ZDDs, Toda, Takahisa (online), available from ⟨http://kuma-san.net/zcomp.html⟩ (accessed "2014-01-01").

[19] Goethals, B.: "Frequent Itemset Mining Dataset Repository", IEEE (online), available from ⟨"http://fimi.ua.ac.be/data/"⟩ (accessed "2014-04-28").

[20] Uno, T.: "LCM (Linear time Closed itemset Miner) ver.2", National Institute of Informatics (online), available from ⟨"http://research.nii.ac.jp/ uno/code/lcm21.zip"⟩ (accessed "2014-01-1").

ⓒ 2014 Information Processing Society of Japan