A-006

ゼロサプレス型二分決定グラフを用いた集合分割の族の表現とその演算アルゴリズムの評価
Representation and Manipulation Algorithms for Family of Set Partitions Using Zero-Suppressed Binary Decision Diagrams and Their Evaluation

奥田 諒平[1)]　川原 純[1)]　湊 真一[1)]
Okuda Ryohei　Kawahara Jun　Minato Shin-ichi

1　はじめに

集合分割とは，ある台集合の要素をいくつかの空でないグループに分ける分け方のことである．集合分割を扱う組合せ最適化には選挙区割問題 [1] や有限オートマトンの状態符号割当 [2] などがある．集合分割の総数は台集合のサイズに対して指数的に増加するため，愚直に制約充足解を列挙しようとするとその計算時間は入力のサイズに対して爆発的に増加する．そこで，集合分割の族を高速に列挙索引化する手法としてゼロサプレス型二分決定グラフ（Zero-Suppressed Binary Decision Diagram; ZDD）[3] と呼ばれるデータ構造を用いたものがいくつか提案されている．高橋ら [6] は，各要素とそれが属する分割グループの組の組合せで集合分割を表現する手法を示しており，本稿ではこの手法をセルラベリング符号化と呼ぶ。川原ら [7] は，隣接グラフを入力として入力グラフの辺集合を用いて各グループが地続きになるような集合分割を表現する手法を示しており，本稿ではこの手法を隣接グラフ符号化と呼ぶ。本研究では，集合分割どうしの細分の関係に基づく半順序から考えられる集合演算をセルラベリング符号化 ZDD と隣接グラフ符号化 ZDD に適用する手法について考察し，計算機実験による性能評価を行う．

2　準備

2.1　集合分割

ある台集合 X に対して，以下の 3 つの条件を満たす $C_1, \ldots, C_k$ を X の集合分割または単に分割とよぶ．また，各分割グループ C_i のことをセル，セルの総数 k を分割数とよぶ．

1. $C_i \subseteq X, C_i \neq \emptyset$
2. $\bigcup_{i=1}^{k} C_i = X$
3. $C_i \cap C_j = \emptyset \ (i \neq j)$

同一の台集合 X に対して，集合分割 $P_1 = \{C_1^1, \ldots, C_l^1\}$, $P_2 = \{C_1^2, \ldots, C_m^2\}$ について，P_1 の任意のセル C_i^1 に対して，$C_i^1 \subseteq C_k^2$ となる C_k^2 が存在するとき，$P_1 \leq_p P_2$ であるとする。$\leq_p$ で定義される半順序は細分の関係を表す．本稿では，このような P_1, P_2 について「P_1 は P_2 の細分である」「P_1 は P_2 と同等かそれより細かい」「P_2 は P_1 と同等かそれより粗い」と表現する．また，ある分割 A, B について $(P \leq_p A) \wedge (P \leq_p B)$ を満たす分割 P の中で最も粗い分割を $A \wedge_p B$ とし，「A と B の交わり」とよぶ，さらに，$(P \geq_p A) \wedge (P \geq_p B)$ を満たす P の中で最も細かい分割を $A \vee_p B$ とし，「A と B の結び」とよぶ，集合分割の族は細分の半順序を元に束をなすことが知られており，$A \wedge_p B, A \vee_p B$ は必ず存在する．

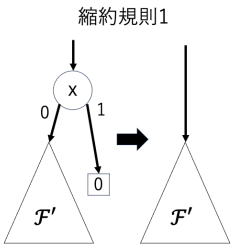

図 1　ZDD の縮約規則 1

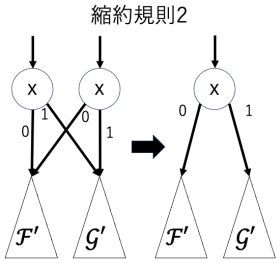

図 2　ZDD の縮約規則 2

表 1　ZDD 上の基本演算

演算名	内容
$\emptyset$	空集合の ZDD を返す
$\{\emptyset\}$	空集合のみを要素とする ZDD を返す
$\mathcal{F} \cap \mathcal{G}$	$\mathcal{F}$ と $\mathcal{G}$ の共通集合を表す ZDD を返す
$\mathcal{F} \cup \mathcal{G}$	$\mathcal{F}$ と $\mathcal{G}$ の和集合を表す ZDD を返す
$\mathcal{F} \setminus \mathcal{G}$	$\mathcal{F}$ と $\mathcal{G}$ の差集合を表す ZDD を返す
$\mathcal{F} \sqcap \mathcal{G}$	$\mathcal{F}$ に属する組合せと $\mathcal{G}$ に属する組合せの集合積を列挙する (Meet 演算)
$\mathcal{F} \sqcup \mathcal{G}$	$\mathcal{F}$ に属する組合せと $\mathcal{G}$ に属する組合せの集合和を列挙する (Join 演算)
$\mathcal{F}.onset(x)$	$\mathcal{F}$ に属する組合せの中で x を含むものを抽出する
$\mathcal{F}.offset(x)$	$\mathcal{F}$ に属する組合せの中で x を含まないものを抽出する
$\mathcal{F}.change(x)$	$\mathcal{F}$ に属する全ての組合せについて，x の有無を反転させる
$\mathcal{F}.top$	$\mathcal{F}$ の根ノードのラベルを返す
$\mathcal{F}.count$	$\mathcal{F}$ が持つ組合せの数を返す
$\mathcal{F}.permit(\mathcal{G})$	$\mathcal{F}$ に属する組合せの中で，$\mathcal{G}$ に属するある組合せに包含されているもののみを抽出する
$\mathcal{F}.restrict(\mathcal{G})$	$\mathcal{F}$ に属する組合せの中で，$\mathcal{G}$ に属するある組合せを包含しているもののみを抽出する

2.2　ゼロサプレス型二分決定グラフ（ZDD）

ZDD[3] とは，組合せ集合を圧縮して表現する有向非巡回グラフのデータ構造である．ZDD は二分決定木を独自の縮約規則を用いて既約になるまで縮約した形で定義される（図 1, 2）．ZDD で表現された組合せ集合は，ZDD に対する演算アルゴリズムを用いて高速に集合演算できることが知られており，表 1 に示す [3, 4, 5].

3　既存手法

3.1　セルラベリング符号化 ZDD

高橋ら [6] は，台集合の要素に全順序を導入してセルに番号を付け，台集合の要素とそれが属するセルの番号のペアをアイテムとする組合せで集合分割を表現する手法を提案している．この手法では，まず集合分割の各セルで最小の要素が小さい順になるように $C_1, C_2, \ldots$

1)　京都大学大学院情報学研究科

を割り当てる．そして，台集合での要素 x について，それぞれ x が $C_i(i \neq 1)$ に属するならば，x_i を組合せに採用する．例えば，台集合 $X = \{a, b, c, d\}$ に対して $a < b < c < d$ とすると，集合分割 $P = \{\{c\}, \{b\}, \{a, d\}\}$ は $C_1 = \{a, d\}, C_2 = \{b\}, C_3 = \{c\}$ となり，$\{b_2, c_3\}$ という組合せで表現される．この表現は任意の集合分割と一対一に対応しており，この組合せを ZDD に格納することで任意の集合分割の族を表現できる．この手法を用いて集合分割の族を格納した ZDD を本稿ではセルラベリング符号化 ZDD とよぶ．

3.2 隣接グラフ符号化 ZDD

川原ら [7] は，隣接グラフを入力として同じセルに属する頂点集合がグラフの辺集合を用いて地続きになるような頂点分割を ZDD 上に列挙する手法を示している．例えば，図 3 に示した頂点分割について，同じセルに属する頂点どうしを結ぶ辺（グラフの太線部分）を全て採用することで頂点分割に一対一対応する辺部分集合を得る．本稿では，この辺部分集合を用いて各セルが同一の連結成分に入っているような頂点分割を可能な頂点分割，可能な頂点分割に対応する辺部分集合を正規の辺集合とよぶ．川原らは可能な頂点分割に対応する正規の辺集合を ZDD に格納することで可能な頂点分割を列挙索引化する手法を提案している．この手法で可能な頂点分割の集合を表現した ZDD を，本稿では隣接グラフ符号化 ZDD とよぶ．

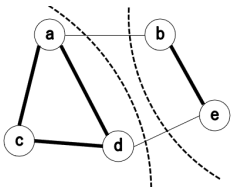

図 3　可能な頂点分割と正規の辺集合の例

4　細分の関係に基づく集合分割の族の演算

細分の関係に基づく半順序 $\leq_p$ を利用した集合演算として以下の 2 系統の演算を考える．

FilterSup/Sub　集合分割の族 A とある単一の分割 P を受け取り，A 中で P と同等かそれより粗い/細かい分割のみを抽出する．

P-Meet/Join　集合分割の族 A, B を受け取り，A, B のそれぞれに属する分割どうしの交わり $\wedge_p$ / 結び $\vee_p$ を列挙する．

以上の演算は ZDD の形で圧縮保持された集合分割に対して，適切な条件をもとに集合分割の族を抽出または列挙する演算であり，これらを高速に実行できるアルゴリズムには応用性が高いと考えられる．本研究では以上の集合演算を既存手法の ZDD 上で実行する手法について考察する．

5　セルラベリング符号化 ZDD への拡張演算

本章では台集合 $X = \{x^1, \ldots, x^n\}(x^1 < \cdots < x^n)$ の集合分割の族を表現したセルラベリング符号化 ZDD $\mathcal{Z}$ と X 上の分割 P を用いて FilterSup/Sub 演算を行うアルゴリズムを検討する．下付き添え字はセルラベリング法におけるセルの位置を表すために用いるので，ここでは要素 x の順を上付き添え字で表す．台集合を X とするセルラベリング符号化 ZDD では，X の各アイテム x^i について，セル番号 j への割り当てに応じて x^i_j のように異なるラベル番号に符号化して ZDD のアイテム変数にする．

まず，X の各要素を順にセルに割り振ることによって集合分割を生成することを考える．$x^1, \ldots, x^{i-1}$ までの要素の任意の分割 $\{C_1, \ldots, C_m\}$ に対して x^i をセル $C_l(1 \leq l \leq m)$ に入れるか，$C_{m+1} = \{x^i\}$ となるセル C_{m+1} を追加することによって i 要素の任意の集合分割を生成できる．

この再帰的な生成法に対して，FilterSup 演算では，P 上で x^i と同じセルに属する要素 $x^w(w < i)$ が存在して，x^w がセル C_l に属するならば x^i を C_l に入れる，という条件を付加することで P と同等かそれより粗い集合分割を再帰的に生成できる．また，FilterSub 演算では，P 上で x^i と異なるセルに属する要素 $x^w(w < i)$ が存在して，x^w がセル C_l に属するならば x^i を C_l に入れない，という条件を付加することで P と同等かそれより細かい集合分割を再帰的に生成できる．これらの再帰的生成法に従い，X に対する集合分割で P と同等かそれよりも粗い分割を列挙した ZDD を出力する演算アルゴリズムを UpperBound 構成（図 4），P と同等かそれよりも細かい分割を列挙した ZDD を出力する演算アルゴリズムを LowerBound 構成とする．本稿ではページ数の都合で LowerBound 構成のアルゴリズムを省略する．これらのアルゴリズムでは分割 P を表現する整数配列 $Parr$ を入力として与えている．$Parr$ は台集合の要素数と同じ長さの整数配列であり，$Parr[i] = k$ であるならば，台集合上で i 番目の要素が分割 P において k 番目のセルに属していることを表す．これらのアルゴリズムでは $i-1$ 番目までの要素の分割のうち，分割数が k であるものを $\mathcal{H}_k$ に保存しておき，i 番目までの要素の分割のうち，分割数が k であるものを $\mathcal{G}_k$ に順次加えていくことによって構成している．

UpperBound 構成のコードを説明する。10 行目の $\mathcal{H}_{j-1}.change(x^i_j)$ は，$\mathcal{H}_{j-1}$ の各分割（のセルラベリング符号化）に x^i_j を追加するという意味になる．$Parr[i] > length(Rep)$ の場合を考えているので，これは $C_j = \{x^i\}$ となるセルを追加することに相当する．分割数 $j-1$ の各分割に $C_j = \{x^i\}$ となるセルを加えて分割数 j の分割となるため，それらを $\mathcal{G}_j$ に加える．12 行目は $\mathcal{H}_k$ の各分割に x^i_j を追加したものを $\mathcal{G}_k$ に加えるという意味である．17 行目から 20 行目では，$\mathcal{H}_j$ のうち，要素 $x^s_2, x^s_3, \ldots$ のいずれも含まない分割のみを $\mathcal{G}_j$ に格納する．すなわち，要素 x^s が 1 番目のセルに入っている分割のみ抽出するという意味になる．23 行目の $\mathcal{H}_k.change(x^i_j).onset(x^s_j)$ は $\mathcal{H}_k$ の各分割に x^i_j を加えて，x^s_j を含むもののみを抽出するという意味になる．すなわち，x^i_j と x^s_j は同じセルに入っているという意味になる．

UpperBound 構成によって得られた ZDD $\mathcal{F}$ は P と同等かそれよりも粗い集合分割の族を表現するため，$\mathcal{F}$ と入力 ZDD $\mathcal{Z}$ について，ZDD の基本演算を用いて共通集合をとる（表 1）ことによって，FilterSup 演算での出力となる集合分割の族を表現した ZDD が得られる．LowerBound 構成についても同様に，入力 ZDD との共通集合をとることでで FilterSub 演算の出力を表現する ZDD が得られる．

Algorithm 1 UpperBound 構成

Require: 台集合 $X = \{x^1, \ldots, x^{|X|}\}$, X 上での分割の整数配列表現 $Parr$
Ensure: X に対する集合分割の中で $Parr$ の表す分割と同等かそれより粗い分割全てを列挙したセルラベリング符号化 ZDD $\mathcal{F}$

$\mathcal{G}_1 \leftarrow \{\emptyset\}$
$\mathcal{H}_1 \leftarrow \{\emptyset\}$
$Rep \leftarrow [1]$
for $i = 2, \ldots, |X|$ **do**
  $\mathcal{G}_i \leftarrow \emptyset$
  $\mathcal{H}_i \leftarrow \emptyset$
for $i = 2, \ldots, |X|$ **do**
  if $Parr[i] > length(Rep)$ **then**
    for $j = 2, \ldots, length(Rep) + 1$ **do**
      $\mathcal{G}_j \leftarrow \mathcal{G}_j \cup (\mathcal{H}_{j-1}.change(x^i_j))$
      for $k = j, \ldots, length(Rep)$ **do**
        $\mathcal{G}_k \leftarrow \mathcal{G}_k \cup (\mathcal{H}_k.change(x^i_j))$
    $Rep.append(i)$
  else if $Parr[i] > 1$ **then**
    $s \leftarrow Rep[Parr[i]]$
    $\mathcal{I} \leftarrow \emptyset$
    for $j = 2, \ldots, length(Rep)$ **do**
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{\emptyset\}.change(x^s_j)$
    for $j = 2, \ldots, length(Rep)$ **do**
      $\mathcal{G}_j \leftarrow \mathcal{H}_j \backslash (\mathcal{H}_j.restrict(\mathcal{I}))$
    for $j = 2, \ldots, length(Rep)$ **do**
      for $k = j, \ldots, length(Rep)$ **do**
        $\mathcal{G}_k \leftarrow \mathcal{G}_k \cup (\mathcal{H}_k.change(x^i_j).onset(x^s_j))$
  for $j = 2, \ldots, length(Rep)$ **do**
    $\mathcal{H}_j \leftarrow \mathcal{G}_j$
$\mathcal{F} \leftarrow \emptyset$
for $i = 1, \ldots, |X|$ **do**
  $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{G}_i$
return $\mathcal{F}$

図 4　UpperBound 構成

FilterSup 演算は，高橋らが提案した指定要素が同じセルに含まれる集合分割を ZDD から抽出する演算（本稿では SameGroup 演算とよぶ）[6] を利用することによっても実装できる．この演算を用いて P 中の全てのセルのそれぞれについてセル中の要素が同じセルに入るような集合分割を $\mathcal{Z}$ から抽出することによって P より粗い分割を $\mathcal{Z}$ から抽出することが可能である（図 5）．

6　隣接グラフ符号化への集合演算の検討

隣接グラフ $G = (V, E)$ について，可能な頂点分割 P_1, P_2 と対応する正規の辺集合 E_1, E_2 を考える．このとき，$P_1 \leq_p P_2$ と $E_1 \subseteq E_2$ は同値である．これは C^1_i 中の任意の頂点ペアが P_2 上でも同一のセルに入ることと，部分グラフ $G'_1 = (V, E_1)$ 上で頂点ペアを結ぶ任意のパスが E_2 に包含されることが対応するためである．ZDD どうしの集合演算 Restrict($\mathcal{F}, \mathcal{G}$) (表 1) は $\mathcal{F}$ に属する要素の中で $\mathcal{G}$ に属する要素のいずれかを包含する分割を全て抽出した ZDD を出力する．従って，フィルタリングを受ける集合分割の族を表現した隣接グラフ ZDD $\mathcal{F}$ と単一の分割 P のみを表現した隣接グラフ符号化 ZDD$\mathcal{G}$ を構成すれば，ZDD の演算 Restrict($\mathcal{F}, \mathcal{G}$) は P よりも粗い集合分割の族を表現し，これは FilterSup 演算の出力と一致する．また，ZDD の集合演算 Permit($\mathcal{F}, \mathcal{G}$) (表 1) は $\mathcal{F}$ に属する要素の中で $\mathcal{G}$ に属する要素のいずれかに包含されるもののみを抽出した ZDD を出力する．FilterSub 演算も同様の考え方で Permit 演算を利用することで実現できる．

Algorithm 2 SameGroup 演算を利用する FilterSup 演算

Require: 台集合 $X = \{x^1, \ldots, x^{|X|}\}$, X 上の集合分割を集めたセルラベリング符号化 ZDD $\mathcal{Z}$, X 上での分割 $P = \{C_1, \ldots, C_{|P|}\}$
Ensure: $\mathcal{Z}$ に属する集合分割の中で P と同等かそれより粗い分割全てを列挙したセルラベリング符号化 ZDD $\mathcal{F}$

$\mathcal{F} \leftarrow \mathcal{Z}$
for each item C_i in P **do**
  if $|C_i| > 1$ **then**
    $\mathcal{F} \leftarrow SameGroup(\mathcal{F}, C_i)$
    //$SameGroup(\mathcal{F}, C_i)$ はセルラベリング符号化 ZDD $\mathcal{F}$ と集合 C_i を受け取り，C_i 中の要素が全て同じセルに属する分割を $\mathcal{F}$ から抽出した ZDD を出力する．
return $\mathcal{F}$

図 5　SameGroup 演算を利用する FilterSup 演算

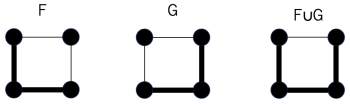

図 6　集合和が非正規の辺集合になる例

また，正規の辺集合を全て集めた集合族は交わり $\cap$ について閉じているため，$P_1 \wedge_p P_2$ と $E_1 \cap E_2$ は同値である．ZDD の基本演算 Meet($\mathcal{F}, \mathcal{G}$) (表 1) は $\mathcal{F}$ と $\mathcal{G}$ に属する要素どうしの交わり $\cap$ を列挙するため，隣接グラフ符号化 ZDD $\mathcal{F}, \mathcal{G}$ について，Meet($\mathcal{F}, \mathcal{G}$) の出力 ZDD は P-Meet 演算の出力である集合分割の族を過不足なく表現する．正規の辺集合を全て集めた集合族は結び $\cup$ について閉じていない（図 6）が，辺集合どうしの結びをとることで得られる辺集合その連結成分によって $P_1 \leq_p P_2$ を表現する．従って，Join($\mathcal{F}, \mathcal{G}$) (表 1) は P-Join 演算の出力を非正規形の形で表現する．

7　性能評価

セルラベリング符号化 ZDD での FilterSup, FilterSub 演算について，提案した手法の性能を評価するため，計算機実験を行った．実験には MacBook Air Retina, 13inch, 2020, 1.1 GHz クアッドコア Intel Core i5, メモリ 8 GB 3733 MHz LPDDR4X を用いた．

セルラベリング符号化 ZDD での FilterSup 演算として，本稿では UpperBound 構成を用いた手法と SameGroup 演算を用いた手法の二つを提案した．これらをそれぞれ手法 1, 手法 2 として，これら 2 つの手法を比較する実験を行った．パラメータとして，台集合 $X = \{x^1, \ldots, x^n\}$ の要素数 n とフィルタリングに用いる分割の分割数 k を設定し，それを元に生成した ZDD $\mathcal{F}$ と分割 P について FilterSup($\mathcal{Z}, P$) を実行した．$\mathcal{Z}$ には，高橋らの手法 [6] を参考に n 要素の台集合における集合分割を全て列挙した ZDD を設定した．分割 P は分割数が k である必要があるため，まず $x^1, \ldots, x^k$ をそれぞれ $C_1, \ldots, C_k$ に加えてから $x^{k+1}, \ldots, x^n$ を $C_1, \ldots, C_k$ のいずれかのセルにそれぞれランダムに加えることによって生成した．結果が表 2 である．この表において，ZDD size は ZDD の接点

表 2　FilterSup($\mathcal{Z}, P$) の実行結果

台集合の要素数 n	P の分割数 k	実行時間 手法 1(ms)	実行時間 手法 2(ms)	ZDD size	Cardinality
10	2	0	0	4	2
10	5	0	0	37	52
10	8	0	1	239	4140
20	4	0	15	108	15
20	10	303	329	76806	115975
20	16	217	175	115968	1.05E+10
30	6	19	174	3045	203
30	10	807	934	148975	115975
30	12	31637	14874	2633507	4213597
30	15	TO	TO		
30	25	84062	89731	16945800	4.64E+18
40	4	1	186	213	15
40	8	466	1294	78956	4140
40	12	196010	TO	41304689	4213597
40	20	TO	TO	-	-
40	35	TO	TO	-	-
40	37	47289	11886	9383041	5.29E+31
50	5	7	509	1242	52
50	11	95212	35051	8969707	678570
50	45	TO	TO	-	-
50	48	24427	3330	2805502	6.29E+44

数，Cardinality は，各 ZDD が格納する集合分割の個数を表す．TO は制限時間の 5 分が経過しても実行終了しなかったことを表す．

手法 1 と手法 2 では，同一のデータ構造を利用して同一の集合分割の族を出力するため，出力 ZDD の接点数およびその集合分割の個数は全て一致した．台集合の要素数に対して分割数が小さいインスタンスでは手法 1 の方が高速であり，分割数が大きくなるにつれて手法 2 の方が高速になっている．また，台集合の要素数が 30 以上のインスタンスでは，分割数が小さいインスタンスや分割数が大きいインスタンスではタイムアウトせずに実行できている一方で台集合の要素数に対して分割数が中程度のインスタンスではタイムアウトが発生している．分割数が大きなインスタンスにおいて手法 2 の方が高速であることの原因は，Restrict 演算の実行回数であると考えられる．X の要素数を n，P の分割数を k とすると，実装手法 1 での Restrict 演算の呼び出し回数は高々 $n \times (n-k)$ 回であるのに対し，実装手法 2 での呼び出し回数は，SameGroup 演算の一回の呼び出しあたり Restrict 演算を $(n+1)$ 回呼び出すため [6]，$(n+1) \times$ (サイズが 2 以上のセルの数) 回である．要素数 n に対して分割数 k が大きくなるほど，サイズが 1 のセルの数が増加し，実装手法 2 の方が高速に動作すると考えられる．

続いて FilterSub 演算について，提案手法の効率性を確認する計算機実験を行った．FilterSup 演算の実験と同様の実験環境とパラメータ設定を用い，提案手法に基づいて FilterSub($\mathcal{Z}, P$) を求める演算を行った結果が表 3 である．台集合の要素数に対して P の分割数が小さくなるほどに列挙対象となる集合分割のパタン数は大きくなるため，台集合の要素数が同じであれば，P の分割数が小さくなるほどに実行時間や ZDD サイズが大きくなっている．台集合の要素数が 30 程度であればどのインスタンスについても問題なく実行できるが，台集合の要素数が 40 を超える場合，小さい分割数のインスタンスにタイムアウトがみられた．

表 3　FilterSub($\mathcal{Z}, P$) の実行結果

台集合の要素数 n	P の分割数 k	実行時間 (ms)	ZDD size	Cardinality
10	2	1	368	3045
10	5	0	51	60
10	8	0	13	4
20	4	161	38180	16146000
20	10	2	577	3600
20	16	0	39	16
30	6	24831	3930570	70259934150
30	10	1120	143127	118395000
30	15	17	2888	96000
30	25	1	63	32
40	8	TO	-	-
40	10	32741	4461069	5.78861E+12
40	20	132	6898	3840000
40	35	1	73	32
50	10	TO	-	-
50	20	6876	243429	33750000000
50	25	4181	322999	3648320000
50	40	7	275	1280
50	45	2	83	32

8　おわりに

本稿では，細分に基づく半順序を利用してセルラベリング符号化 ZDD へのフィルタリングを行うアルゴリズムと隣接グラフ符号化 ZDD に対して提案した集合演算を行う手法を示した．セルラベリング符号化 ZDD については，FilterSup/Sub 演算の実装により，分割数 20 程度のインスタンスであれば提案手法によってタイムアウトせずに実行できることと提案手法間で得意なインスタンスの考察を示した．隣接グラフ符号化 ZDD については，P-Meet 演算が ZDD の基本演算 Meet によって実現できること，P-Join 演算の演算結果が基本演算 Join を用いて非正規形の ZDD で表現できること，FilterSup 演算，FilterSub 演算がそれぞれ ZDD の基本演算 Restrict, Permit を用いて実現できることを示した．今後の課題として，隣接グラフ符号化 ZDD での P-Join を正規形で手に入れる手法を検討すること，集合分割の族を表現する手法としてセルラベリング符号化と隣接グラフ符号化の性能を比較することが考えられる．

謝辞　本研究の一部は，JSPS 科研費 JP20H00605, JP20H05964，JP23H04383 の助成を受けたものである．

参考文献

[1] 坂口利裕 and 和田淳一郎．選挙区割り問題．オペレーションズ・リサーチ 48 巻第 1 号，p30-35,2003.

[2] Giovanni De Micheli, Robert K. Brayton and Alberto L. Sangiovanni-Vincentelli. Optimal state assignment for finite state machines. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 4: 269-285, 1985.

[3] Shin-ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems, 30th ACM/IEEE Design Automation Conference, pages 272-277, 1993.

[4] Donald E. Knuth. The Art of Computer Programming, Volume 4A, Combinatorial Algorithms, Part 1. Addison Wesley Professional, 2011.

[5] Hiroshi G. Okuno and Shin-ichi Minato and Hideki Isozaki. On the properties of combination set operations. Inf. Process. Lett. 66: 165-199, 1998.

[6] 高橋翔哉 and 吉岡真治. ZDD を用いた集合分割の列挙索引化手法の提案と実験的評価. 人工知能学会研究会資料 人工知能基本問題研究会 112 回 (2020/3), page 8. 一般社団法人人工知能学会, 2020.

[7] Jun Kawahara, Takashi Horiyama, Keisuke Hotta and Shin-ichi Minato. Generating all patterns of graph partitions within a disparity bound. Workshop on Algorithms and Computation, pages 119-131, 2017.