Vol.2016-AL-159 No.2
2016/9/23

グラフ部分構造列挙のための ゼロサプレス型項分岐決定図の効率的な構築法

西野 正彬[1,a)]　安田 宜仁[1]　湊 真一[2]　永田 昌明[1]

概要：本稿ではグラフ部分構造の集合を表現するゼロサプレス型項分岐決定図 (Zero-suppressed Sentential Decision Diagram, ZSDD) の効率的な構築法を提案する．グラフ部分構造とは，マッチングや点素パスの集合などの，特定の性質を満たす辺の集合のことを表す．ZSDD は近年提案された集合族を表現するためのデータ構造であり，ZSDD を用いてグラフ部分構造の集合を表現することで，簡潔な表現となり様々な演算を効率的に実行することが可能である．本稿ではグラフ部分構造の集合を表現する ZSDD を高速に構築する方法を提案する．また，提案するトップダウン構築法の副作用として，グラフの Branch Decomposition から ZSDD の頂点数が見積もれることも示す．

1. はじめに

二分決定図 (Binary Decision Diagram, BDD) [1] は論理関数を圧縮して表現可能なデータ構造である．BDD を用いて論理関数を表現することで，BDD のサイズに比例する計算時間で様々な計算を効率的に実行することができる．その論理関数操作における効率性のため，BDD にはいくつもの変種が存在する．ゼロサプレス型二分決定図 (Zero-suppressed Binary Decision Diagram, ZDD) [8], 項分岐決定図 (Sentential Decision Diagram, SDD) [3]，ゼロサプレス型項分岐決定図 (Zero-suppressed Sentential Decision Diagram, ZSDD) [10] などが代表的な BDD の変種である．これらのうち，SDD と ZSDD は，二つの決定図に対して論理演算を実行して実行結果の決定図を生成する Apply 演算をサポートしていることと，それぞれ BDD と ZDD を包含して BDD, ZDD よりも簡潔となることが知られているため，近年注目を集めている [3], [10].

決定図の重要な応用事例として，グラフ部分構造集合の表現が挙げられる．ここでグラフの部分構造とは，パスやサイクル，マッチング，全域木といった，グラフの辺の集合として表現できるもののことをいう．これら辺の集合の集合は集合族として表現でき，集合族は論理関数を用いて表現できるため，決定図を用いて部分構造の集合を表現することができる．さらに決定図はグラフの部分構造の集合をコンパクトに表現できることが知られている．例えば，Knuth は 10^{10} 通り以上あるグラフの全ての連結成分の集合が，数百頂点からなる BDD によって表現できることを示している [7]. 決定図を用いてグラフの部分構造を表現することで，部分構造の数え上げやランダムサンプリングなどの様々な演算を高速に実行することが可能となる．こうした利便性のため，決定図はネットワークの信頼性計算 [4]，グラフ彩色問題の求解 [9]，配電網の損失最小化問題の求解 [6] などのタスクにおいて，グラフの部分構造を扱うための重要な道具として利用されている．

本稿ではグラフの部分構造の集合を表現する ZSDD を高速に構築するためのトップダウン構築アルゴリズムを提案する．グラフ部分構造の集合を表現する BDD, ZDD を高速に構築するアルゴリズムとして SimPath アルゴリズム [7] が知られているが，本稿で提案するアルゴリズムは SimPath を ZSDD 構築にも適用できるよう拡張したものである．前述した通り，ZSDD は ZDD を包含し，ZDD よりも簡潔な表現であることが知られているため，提案法でこれまで ZDD では扱うことができなかった巨大なグラフ，複雑なグラフの部分構造をより効率的に扱うことが可能となる．検証では，提案法によって ZDD より頂点数が少ない ZSDD を高速に作成できたことを示す．また，提案する ZSDD 構築方法の副作用として，部分構造の集合を表現する ZSDD の頂点数の上限をグラフの Branch Decomposition より与えることができることも示す．

2. 準備

$G = (V, E)$ を V を頂点の集合，E を辺の集合とするグラフとする．頂点の数，辺の数をそれぞれ $|V|, |E|$ とする．

[1] 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
[2] 北海道大学 大学院情報科学研究科
[a)] nishino.masaaki@lab.ntt.co.jp

マッチングやパスなどのグラフ部分構造は E の部分集合によって表現される．そのため，これらの部分構造の集合は E の部分集合からなる集合族として表現される．集合族は論理関数と等価であるため，以下では決定図は集合族を表現するものとして説明を進める．

■ $(\mathbf{X},\mathbf{Y})$-**分割**

$(\mathbf{X},\mathbf{Y})$-**分解**は与えられた集合族を小さな集合族に分割する手法であり，ZSDD を構成する重要な技術である．f を集合族，$\mathbf{X}$, $\mathbf{Y}$ はそれぞれ要素の集合であり，全体集合の分割となっているとする．集合族 f は，$\mathbf{X}$, $\mathbf{Y}$ を全体集合とする集合族 $p_i(\mathbf{X})$, $s_i(\mathbf{Y})$ によって，

$$f = [p_1(\mathbf{X}) \sqcup s_1(\mathbf{Y})] \cup \cdots \cup [p_n(\mathbf{X}) \sqcup s_n(\mathbf{Y})] \ ,$$

として表現される．記号 $\cup$, $\sqcup$ は集合族に対する和，結合演算を表し，それぞれ $f \cup g = \{a \mid a \in f$ かつ $a \in g\}$, $f \sqcup g = \{a \cup b \mid a \in f$ かつ $b \in g\}$ として定義される．$p_1,\ldots,p_n$ を $(\mathbf{X},\mathbf{Y})$-分解における**主部**，$s_1,\ldots,s_n$ を **副部**とよぶ．もし主部が排他的 (すべての $i \neq j$ について $p_i \cap p_j = \emptyset$) かつ網羅的 ($\bigcup_{i=1}^{n} p_i$ が全体集合のべき集合と等しい) かつ無矛盾 (すべての i について $p_i \neq \emptyset$) であるならば，その $(\mathbf{X},\mathbf{Y})$-分解を $(\mathbf{X},\mathbf{Y})$-**分割**とよび，$\{(p_1,s_1),\ldots,(p_n,s_n)\}$ として表現する．さらに，すべての $i \neq j$ について $s_i \neq s_j$ を満たすならば $(\mathbf{X},\mathbf{Y})$-分割は圧縮されているとよぶ．例えば，集合族 $\{\{A,B\},\{B\},\{B,C\},\{C,D\}\}$ の $\mathbf{X} = \{A,B\}$, $\mathbf{Y} = \{C,D\}$ としたときの $(\mathbf{X},\mathbf{Y})$-分割は

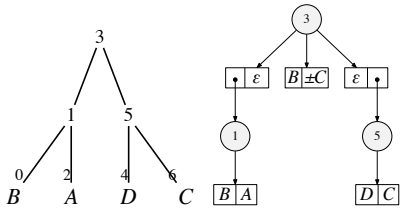
$$[\{\{A,B\}\} \sqcup \{\emptyset\}] \cup [\{\{B\}\} \sqcup \{\emptyset,\{C\}\}] \cup [\{\emptyset\} \sqcup \{\{C,D\}\}] \ ,$$

である．ここで $\{\{A,B\}\}$, $\{\{B\}\}$, $\{\emptyset\}$ が主部であり，$\{\emptyset\}$, $\{\emptyset,\{C\}\}$, $\{\{C,D\}\}$ が副部である．

■ **vtree**

$(\mathbf{X},\mathbf{Y})$-分割と並んで ZSDD を構成する重要な概念である *vtree* を導入する．ZSDD は集合族に対して再帰的に $(\mathbf{X},\mathbf{Y})$-分割を適用することで有向非巡回グラフの形に集合族を分解して表現する手法である．すなわち，与えられた集合族を，$(\mathbf{X},\mathbf{Y})$-分割によって $\mathbf{X}$, $\mathbf{Y}$ を全体集合とする集合族 $p_1,\ldots,p_n$, $s_1,\ldots,s_n$ に分解し，さらにそれらを $(\mathbf{X},\mathbf{Y})$-分割によって部分集合族に分解する…という手続きを表現したものが ZSDD である．vtree は再帰的な $(\mathbf{X},\mathbf{Y})$-分割の順序を与えるものであり，ある vtree に沿った $(\mathbf{X},\mathbf{Y})$-分割を表現する ZSDD を作成すると，一意な ZSDD を構成できる．vtree は各節が必ず 2 つの子をもつような根付き二分木であり，vtree の各葉が全体集合に含まれる各要素に対応している．図. 1 (a) に vtree の例を示す．図中の葉は対応する変数を表現しており，節の数字は ID を表している．各節には一意な ID が付与されている．

vtree の節は，全体集合に含まれる要素を，左側の子を根とする木に含まれる要素の集合と右側の子を根とする木に含まれる要素の集合の 2 つの集合に分割しているとみなすことができる．図中の根 v_3 は，$\mathbf{X} = \{A,B\}$ かつ $\mathbf{Y} = \{C,D\}$ であるような $(\mathbf{X},\mathbf{Y})$-分割を表している．同様に v_3 の左側の子 v_1 も，v_1 を根とする部分木において $\mathbf{X} = \{B\}$ かつ $\mathbf{Y} = \{A\}$ であるような $(\mathbf{X},\mathbf{Y})$-分割を表している．以下では v^l, v^r がそれぞれ v の左側の子，右側の子を表しているものとする．また，それぞれの頂点を根とする部分木を表すためにも v^l，v^r を用いる．もし v^l が葉であるとき，v を**シャノンノード**とよび，そうでない場合に**分解ノード**とよぶ. 図 1 (a) の根は分解ノードであり，その子はどちらもシャノンノードである．



(a) A vtree (b) A ZSDD

図 1: (a) vtree の例, (b) (a) の vtree を参照し，集合族 $\{\{A,B\},\{B\},\{B,C\},\{C,D\}\}$ を表現する ZSDD

なお，以下では入力グラフ，vtree, ZSDD の 3 種類のグラフを扱うため，混乱を避けるために vtree の頂点を vnode とよび，$v, v^l, v^r, v_1, v_2, \ldots$ などと表す．同様に，入力グラフの頂点は頂点あるいは gnode とよび，$u, u_1, u_2, \ldots$ などと表す．後述する ZSDD の頂点は znode とよび，$z_1,\ldots,z_n$ などと表す．

3. ゼロサプレス型項分岐決定図 （ZSDD）

ZSDD を以下のように再帰的に定義する．なお，α を ZSDD とし，α が表現する集合族を $\langle\alpha\rangle$ とする．

定義 1. α は以下のいずれかの条件を満たすとき， *vnode* v を参照する *ZSDD* とよぶ．

- $\alpha = \varepsilon$ または $\alpha = \bot$. *(解釈: それぞれ* $\langle\varepsilon\rangle = \{\emptyset\}$, $\langle\bot\rangle = \emptyset$*)*
- $\alpha = X$ または $\alpha = \pm X$ かつ v が要素 X に対応する *vtree* の葉．(解釈: それぞれ $\langle X\rangle = \{\{X\}\}$, $\langle\pm X\rangle = \{\{X\},\emptyset\}$)
- $\alpha = \{(p_1,s_1),\ldots,(p_n,s_n)\}$, v は *vnode,* $(p_1,\ldots,p_n)$ がそれぞれ部分 *vnode* v^l を参照する *ZSDD,* $s_1,\ldots,s_n$ *vnode* v^r を参照する *ZSDD,* かつ $\langle p_1\rangle,\ldots,\langle p_n\rangle$ が $(\mathbf{X},\mathbf{Y})$-分割の条件を満たす集合族に対応している．(解釈: $\langle\alpha\rangle = \bigcup_{i=1}^{n} \langle p_i\rangle \sqcup \langle s_i\rangle$)

ここで，ε, $\bot$, X, $\pm X$ を**終端** *ZSDD* とよぶ．終端 ZSDD でない ZSDD は，vnode v によって表されている $(\mathbf{X},\mathbf{Y})$-分割 $\{(p_1,s_1),\ldots,(p_n,s_n)\}$ を表しており，**決定** *ZSDD* とよぶ．図. 1 (b) は図 1 (a) の vtree を参照し，集合族

$\{\{A,B\},\{B\},\{B,C\},\{C,D\}\}$ を表す ZSDD である．図中の図中の円ノードと，その子ノードである四角いノードとあわせて決定 ZSDD を表現している．円ノード中の数字はその決定 ZSDD が参照している vnode の ID である．四角いノード $\boxed{p}\boxed{s}$ は $(\mathbf{X},\mathbf{Y})$-分割に含まれる主部，副部のペアを表現している．左側が主部，右側が副部である．以下では決定 ZSDD の根を表す丸いノードを決定 znode, 四角いノードを要素 znode とよぶ．

もし ZSDD が $\{(\varepsilon,\alpha),(\bar{\varepsilon},\bot)\}$, $\{(\alpha,\varepsilon),(\bar{\alpha},\bot)\}$ もしくは $\{(\mathfrak{p},\bot)\}$ といった形の $(\mathbf{X},\mathbf{Y})$-分割を内部に持たないならば，その ZSDD は**削減されている**とよぶ．ここで $\bar{\alpha}$ は α が表す集合族の補集合を表す．削減されていない ZSDD は，上記のいずれかに該当する $(\mathbf{X},\mathbf{Y})$-分割をすべて α に置き換えることで削減されたな ZSDD に変換することができる．もし ZSDD α 中の全ての分割に $(\beta,\bot)$ の形の要素が含まれていないなら，α は**暗黙的な分割**を採用しているとよぶ．暗黙的な分割は副部が $\bot$ であるような要素を明示的に表現しないことによってで，ZSDD の頂点数を削減する．暗黙的な分割をによって削除された要素の主部は，$(\mathbf{X},\mathbf{Y})$-分割の他の主部をもとに復元可能である．図 1 (b) は暗黙的な分割を採用している．以下では暗黙的な分割を採用した ZSDD を構築する．

4. トップダウン構築アルゴリズム

提案する ZSDD 構築アルゴリズムは，後述するフロンティア頂点の情報を用いた構築が適用可能な，複数の異なるグラフ部分構造の集合を表す ZSDD を構築できるという意味で汎用的なものである．紙面の都合上，本稿ではマッチングと点素パスの集合を表す ZSDD を構築する方法のみを示す．特にマッチングの集合の構築法は比較的単純なため，以下ではマッチングのためのトップダウン構築を例に説明を行う．

■ フロンティア

提案するトップダウン構築アルゴリズムは，vtree とグラフを入力として受け取り，決定 znode を根から順に構築していく．すなわち，まず根 vnode v を参照する決定 znode を作成し，次に v を参照している全ての znode について，その子である要素 znode を作成する．この手続きを生成した子 znode に対しても再帰的に繰り返すことによって所望の ZSDD を構築する．ただし，素朴に znode を上位から順に生成していくと，仮に各決定 znode の子頂点の数が高々 2 つにおさまっていたとしても，根からの高さ h の子 znode の個数は 2^h 個となり，指数的に znode の数が増加してしまう．そこで，決定 znode を生成する際に等価な znode をマージすることによって急激な頂点数の増加を避けながら構築を行う．

znode z,z' は，参照する vnode が等しくかつそれらが表す集合族が等しいときに等価な頂点となる．vnode v を参照する znode が表す集合族を f, v を根とする vtree の葉頂点に対応する辺の集合を $E_v \subseteq E$ とする．f はある $S \subseteq (E \setminus E_v)$ に対して，$f \sqcup \{S\}$ がグラフの部分構造集合となっているような集合族である．S が変化すると条件を満たす f も変化するが，異なる S,S' に対して同じ集合族 f が対応することもある．この判定を行うために**フロンティア頂点**とよばれるグラフ中の頂点集合が利用できる．いま，E_v によって誘導される G の部分グラフを G_1, $E \setminus E_v$ によって誘導される部分グラフを G_2 とする．このときに G_1 と G_2 の両方に出現しているグラフの頂点をフロンティア頂点とよぶ．ある種のグラフ部分構造に対しては，取り得る集合族 f は $\mathbf{X}$ によって G_2 中でフロンティア頂点にどのように辺が接続しているかにのみ影響を受ける．すなわち，異なる S,S' であっても，フロンティア頂点の状態が等しければ，対応する集合族は等価となる．

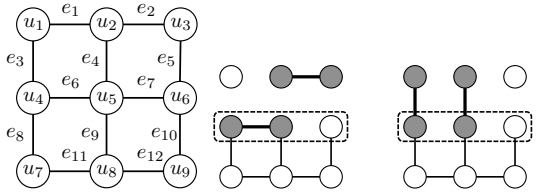


図 2: (a) グラフの例と (b) フロンティア頂点 u_4,u_5,u_6 に等しい接続状況を与える $e_1,\ldots,e_7$ の選択の例.

マッチングはフロンティア頂点の状態で等価性が判定できる部分構造の一例である．いま，図 2 (a) のグラフにおいて，ある vnode v を根とする vtree に含まれる辺の集合 E_v が $\{e_8,e_9,e_{10},e_{11},e_{12}\}$ であったとする．このときフロンティア頂点は u_4,u_5,u_6 である．図 2 (b) は $E \setminus E_v$ から辺がいくつか選択されたときのフロンティア頂点の様子を表している．これらの辺集合のいずれにおいても u_4,u_5 は既に辺に接続されており，u_6 は接続されていない．この場合の E_v を全体集合とするマッチングを構成する集合族は，どちらの例でも $\{\emptyset,\{e_{10}\},\{e_{10},e_{11}\},\{e_{11}\},\{e_{12}\},\}$ であり，等価である．このように，マッチングの集合を構築する際には，フロンティアに含まれる頂点が辺に接続されているかどうかを利用することで，構築される集合族の等価性，すなわち決定 znode の等価性を判定することができる．

以下に示すトップダウン構築アルゴリズムでは，フロンティア頂点の状態は生成された znode の状態を表現するラベルとして利用される．ラベルは要素数 $|V|$ の配列 m によって表現される．フロンティア頂点 $u_i \in V$ の状態は $m[i]$ に格納される．マッチング集合構築時には，$m[u]$ には U, C, R, F の 4 種類のシンボルを保持する．それぞれ，u がいずれの辺とも接していない場合 (U)，u が単一の辺と接している場合 (C), u が以降の処理で必ず単一の辺と接する必要が有る場合 (R), u が以降フロンティア頂点として

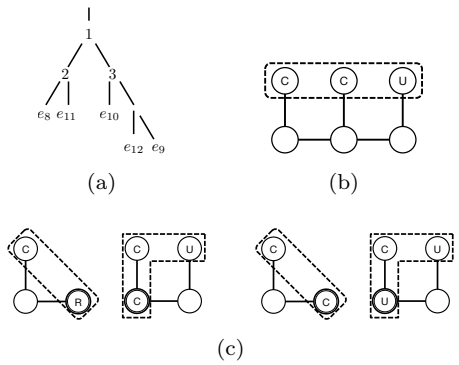


図 3: 主部と副部の状態の組合せの例．(a) vtree，(b) 親 znode の状態 (c) 可能な主部・副部の状態の例.

Algorithm 1: トップダウン構築アルゴリズム

Input: グラフ $G = (V, E)$, vtree の根 v
Output: G の部分構造の集合を表現する ZSDD Z

```
1  v ← root vtree node
2  Z[v] ← rootState()
3  recursiveConstruct(v, Z)
4  return reduce(Z)
```

Algorithm 2: recursiveConstruct(v, Z)

```
1  if v がシャノンノード then
2      for x ∈ Z[v] do
3          elems ← ∅
4          m_1 ← shannonChild(v, x, False)
5          if m_1 ≠ ⊥ then
6              y_1 ← uniqueNode(m_1)
7              elems ← elems ∪{(ε, y_1)}
8          m_2 ← shannonChild(v, x, True)
9          if m_2 ≠ ⊥ then
10             m_2 ← uniqueNode(m_2)
11             elems ← elems ∪ {(X, m_2)}
12         elems を x の子に設定する
13     if v^r が節 then recursiveConstruct(v^r, Z)
14 else                                   // 分解ノード
15     for x ∈ Z[v] do
16         elems ← ∅
17         for (m_p, m_s) ∈ decompChild(v, x) do
18             y_p ← uniqueNode(m_p),  y_s ← uniqueNode(m_s)
19             elems ← elems ∪ {(y_p, y_s)}
20         elems を x の子ノードに設定する
21     recursiveConstruct(v^l, Z),  recursiveConstruct(v^r, Z)
```

出現しない場合 (F) を表す.

■ アルゴリズム（トップダウン構築共通の処理）

トップダウン構築アルゴリズムを Algorithm 1 に示す．アルゴリズムはグラフ $G = (V, E)$ と vtree の根 v を入力として受け取り，特定のグラフ部分構造の集合を表す ZSDD を出力する．ここで $Z[v]$ は，vnode v を参照する決定 znode の集合を保持するテーブルである．ZSDD は決定 znode の集合として表現されるため，$Z[v]$ をすべての vnode v について集めたものが一つの ZSDD となる．アルゴリズムは，まず rootState() を実行し，ZSDD の根となる znode と，そのラベルを設定する．rootState() の処理内容については後述する．次に再帰的に vnode v を参照する znode を構築する手続きである recursiveConstruct(v, Z) を実行することで，再帰的に zsdd を作成する．その後．reduce(Z) は Z に格納された znode のうち冗長なものを vtree の葉から根の順番に削除することで，削減されかつ暗黙的な分割を適用した ZSDD を構築する手続きである．紙面の都合上 reduce の詳細は割愛する．

recussiveConstruction(v, Z) の手続きについてアルゴリズム 2 を用いて説明する．v がシャノンノードであるならば，手続き shannonChild(v, z, t) を $t \in \{\mathsf{true}, \mathsf{false}\}$ の 2 種類の引数でそれぞれ実行する．shannonChild(v, z, t) は，決定 znode z の子要素 znode の副部を返す関数であり，副部が終端 znode ならば該当の終端頂点を，そうでないならば v^r を参照する決定 znode の状態に付与するラベル m を返す．m_1 が $\bot$ でないならば，y_1 に uniqueNode(m_1) を代入する（6 行目）．uniqueNode(m) は，m が終端 znode であるならばその頂点を，そうでないならば $Z[v^r]$ 中にラベル m をもつ znode が既に存在するかどうかを調べ，もし存在するならばそのアドレスを，存在しないならば新たに状態 m をもつ znode を $Z[v^r]$ に追加し，そのアドレスを出力する．もし v が分解 vtree ならば，手続き decompChild(v, z) を各 $x \in Z[v]$ に対して実行する．decompChild(v, x) は，x の子である主部，副部のペア (m_p, m_s) の集合を出力する．ここで m_p, m_s は終端 znode もしくは決定 znode のラベルである．各 m_p, m_s について手続き uniqueNode を実行して，その結果を用いて z の子である要素 znode を設定する．この手続きを再帰的に v^l, v^r に対して繰り返すことで，znode をすべて生成する．こうして生成された zsdd は削減されておらず，また暗黙的な分解も含むため，最後に reduce(Z) を適用する．

■ アルゴリズム（マッチング集合構築固有の処理）

提案法は 3 つの手続き rootNode(), shannonChild(v, x, t), decompChild(v, x) を構築したい対象の部分構造にあわせて適切に設計することで，様々な部分構造集合を表す ZSDD のトップダウン構築を行うことができる．以下では全てのマッチングの集合を表現する ZSDD の構築を例にとって，それぞれの関数の具体的な動作を説明する．なお，以下では vnode v のフロンティア頂点である gnode の集合を $F(v)$ と表す.

まず rootNode() は, 全ての $u_i \in V$ に対して $m[i] = \mathsf{U}$ とした配列 m を出力する．shannonChild(v, z, t)，decompChild(v, t) はそれぞれ Algorithm 3 にしめす．shannonChild(v, z, t) は，z の状態を更新して，z の子の状態を作成する．もし $t = \mathsf{true}$ ならば，葉 v^l に対応する要素 e を追加してもよいか，すなわち e の両端点を u_a と u_b とすると，

u_a, u_b ともにまだ辺が接続されていないかを判定する．もし既に辺が接続されているならば $\perp$ を出力し処理を終了する．そうでないならば，$m[a], m[a]$ を C に設定し，e の追加によって接する辺の状態が変化したことを表す．その後，全ての $u_i \in F(v^r) \setminus F(v)$ に対して $m[i] \leftarrow \mathsf{F}$ とする．このとき，もし $m[i] = \mathsf{R}$ であったならば，u に以後の処理で辺が接続されることがないため，$\perp$ を出力して処理を終了する．もし v^r が葉でなかった場合には，更新した m を出力する．もし v^r が葉であった場合には適切な終端 ZSDD を出力する．

手続き decompChild() は，もし $F(v^l)$ と $F(v^r)$ 共通の頂点がなかった場合には，単に m の各フロンティア頂点の要素を m_p と m_s にコピーして出力するのみである（25 行目）．もし共通の頂点が存在するならば，主部と副部とで整合性がとれるような共通の設定の値の組合せを全て列挙し，可能な組合せの数だけ要素 znode を作成する．主部と副部とで共通のフロンティア頂点を u_i とすると，$m[i] = \mathsf{C}$ であった場合には，主部と副部のどちら側でも u_i にこれ以上辺を接続させられないため，$(m_p[i], m_s[i]) = (\mathsf{C}, \mathsf{C})$ とする．$m[i] = \mathsf{U}$ であった場合には，主部か副部のいずれか片方においてのみ u_i に辺が接続されてよいため，主部で接続を許す場合とそうでない場合とで，$(m_p[i], m_s[i])$ を (R, C) または (C, U) に設定する．ここで前者において主部の状態を U ではなく R に設定するのは，C としたときと R に設定したときのそれぞれに対応する集合族を排他とするためである．$m[i] = \mathsf{R}$ であった場合には，主部側で u_i に辺を接続させるか，副部側で接続させるかの違いによって $(m_p[i], m_s[i])$ を (R, C) または (C, R) に設定する．このように，m_p, m_s の共通のフロンティア頂点への可能な割り当て方をすべて列挙し，全ての可能な割り当て方について m_p と m_s のペアを作成し，それらを z の子の要素として elems に格納して出力する．

マッチング集合の構築における decompChild(v, z) の動作例を図 3 に示す．この図は親のラベルからどのように子のラベルが生成されるかを示したものである．いま，vnode v_1 を参照している znode z の子 znode を構築しているものとする．フロンティア頂点は u_4, u_5, u_6 であり，それらに対応する値は C, C, U であったとする．v^l は e_8, e_{11} を含んでおり，v^r は e_9, e_{10}, e_{12} を含んでいる．そのため，$F(v^l) = \{u_4, u_8\}$, $F(v^r) = \{u_6, u_8\}$ であり，u_8 が共通のフロンティア頂点となっている．$m[8] = \mathsf{U}$ であることから，主部と副部に対して (R, C) または (C, U) の 2 種類の $m_p[8]$ への割り当てパターンが存在する．共有の頂点は u_8 のみなので，この頂点の子である要素頂点は 2 種類存在する．

5. znode 数の上界

提案法では，構築される ZSDD の頂点数に，入力グラフ由来の上界を与えることができる．

Algorithm 3: マッチング集合構築のための処理

```
1  function shannonChild(v, z, t):
2      m ← z のラベルをコピー
3      e ← v^l に相当するグラフの辺
4      (u_a, u_b) ← e が対応する辺に接する頂点
5      if t = True then
6          if m[a] = C or m[b] = C then return ⊥
7          m[a] ← C,   m[b] ← C
8      for u_i ∈ F(v) \ F(v^r) do
9          if m[i] = R then return ⊥
10         else m[i] ← N
11     if v^r が葉 vnode でない then return m
12     else
13         e ← v^r に相当するグラフの辺
14         (u_a, u_b) ← e に接する頂点
15         if (m[a], m[b]) = (C, R) or (R, C) then return ⊥
16         else if m[a] = C or m[b] = C then return ε
17         else if m[a] = R or m[b] = R then return e
18         else return ±e
19 function decompChild(v, z):
20     elems ← ∅
21     common ← F(v^l) ∪ F(v^r)
22     m_p, m_s に x のラベルをコピー
23     for u_i ∈ F(v^l) \ F(v) do m_p[i] ← U
24     for u_i ∈ F(v^r) \ F(v) do m_s[i] ← U
25     if common = ∅ then return {(m_p, m_s)}
26     for u_i ∈ common do
27         if m_p[i] = C then
28             Combs[i] ← {(C, C)}
29         else if m_p[i] = U then
30             Combs[i] ← {(R, C), (C, U)}
31         else if m_p[i] = R then
32             Combs[i] ← {(R, C), (C, R)}
33     for vals ∈ enumerateCombination(Combs) do
34         m'_p ← copy of m_p,   m'_s ← copy of m_s
35         for u_i ∈ Common do (m'_p[i], m'_s[i]) ← vals[i]
36         elems ← elems ∪ {(m'_p, m'_s)}
37     return elems
```

定理 2. α を 提案法で構築された，グラフの全てのマッチングの集合を表現する *ZSDD* であるとする．このときに α に含まれる決定 *znode* の個数は，$|E|2^W$ 個以下となる．ここで $|E|$ はグラフの辺の数，W は $F(v)$ のサイズの最大値である．

証明. (Sketch) 提案アルゴリズムで生成される ZSDD 中では，同一の vnode を参照しつつ，かつ同一のラベルをもつ znode は常に 1 つしか存在しないため，vnode v を参照する決定 znode の数は，v を参照する頂点を構築する際に用いられたラベルの異なり数以下となる．マッチング集合の構築においては，フロンティアに含まれる頂点に対しては必ず C, U, R のいずれかが割り当てられるため，ラベルの異なり数は $3^{|F(v)|}$ 以下となる．この上界は，頂点が取り得る値が常に 2 種類であることを利用することでさらに

$2^{|F(v)|}$ まで改善することができる．W の定義および葉でない vnode の数は $|E|-1$ となることより，決定 znode の数は $|E|2^W$ 以下となる． □

W の値は利用される vtree によって異なり，また W を最小にするような vtree を見つける問題は，BDD において最適な変数順序を見つける問題を含むため NP 困難である．しかしながら，構築される ZSDD の頂点数を小さくする vtree を近似的に求める効果的な方法として，グラフの Branch Decomposition (BD) が利用できることを示す．無向グラフ $G=(V,E)$ の BD とは，$2|E|-1$ 頂点をもち，各節の次数が 3 であるような木 T のことである [11]．T の各葉は E の各辺に一対一で対応している．T の各辺 e に対して，e の幅を以下のようにして定義する．e を T から除くと，T は 2 つの部分グラフに分割され，それに従い T の各葉ノードに対応していた G の辺も 2 つの集合に分割される．こうして分割された辺の集合によって誘導される G の部分グラフを G_1, G_2 としたときに，G_1 と G_2 の共通頂点の集合のサイズを e の幅とする．また，T の幅をその辺の幅の最大値として定義する．次にグラフの BD と vtree との関係を示す．

定理 3. 幅が W であるようなグラフの *BD* T が与えられたとき，各 *vnode* v について $F(v)$ の大きさが W 以下であるような *vtree* を構築できる．

T から vtree を構築する手続きを示すことで定理を証明する．T は根なしの木 vtree は根付き木かつ順序木であるため，まず T の葉を一つ選択し，その葉と隣接する節の間に頂点を一つ挿入する．そして挿入した頂点を根とし，各子に対して順序をつけていくことによって，vtree を構築することができる．Branch decomposition の幅の定義は，vtree のある vnode におけるフロンティアサイズの定義と等しいため，フロンティアサイズの最大値は W と一致する．

一般のグラフに対して幅最小の BD を見つける問題は求解が困難である．しかし，例えば平面グラフに関しては幅最小の BD を見つける問題が多項式時間で解けることがや，よい BD を発見するための効果的なヒューリスティクス [2] が知られている．

6. 点素パス集合を表す ZSDD の構築

本節ではマッチングのほかの部分構造の例として，点素パス集合を表現する ZSDD のトップダウン構築法の概要を示す．点素パスとは，ある 2 つの端点 $s,t \in V$ が与えられたときに，s を始点，t を終点とするようなパスのうち，同じ頂点が経路中で高々 1 度しか出現しないものを指す．

■ フロンティア

マッチング集合の場合と同様に，点素パス集合を表す ZSDD のトップダウン構築においてもフロンティア頂点のパターンを znode の状態として用いる．マッチングの場合にはフロンティア頂点への辺の接続数を状態として用いていたが，点素パスの列挙においては辺の接続数とともに，どのフロンティア頂点同士が部分パスの端点となっているかを情報として用いる．図 4 は点素パス構築における等価なフロンティアパターンの例である．なお，$s=u_1$, $t=u_9$ とする．フロンティア頂点は u_4, u_5, u_6 であり，それぞれ u_4 は u_1 と，u_5 は u_6 とパスによって接続されている．残りの辺 $e_8, \ldots, e_{12}$ の選択方法のうち，$e_1, \ldots, e_7$ から選択された辺とあわせて点素パスを構成するためには，e_8, e_9, e_{10}, e_{11} を選択することが唯一の選択肢である．この選択はフロンティア頂点がどの頂点とパスによって接続されているかにのみ依存し，パスの形状には依存しない．この例では u_5, u_6 を接続するパスは異なるが，フロンティア頂点の接続状況は同一であるため等価な状態となる．

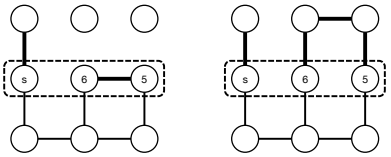


図 4: 点素パス構築における等価なフロンティア頂点の状態の例．

■ アルゴリズム

点素パス集合を表す ZSDD のトップダウン構築におけるサブルーチン $\mathsf{shannonChild}(v,x,t)$, $\mathsf{decompChild}(v,x)$ の概要を説明する．$\mathsf{shannonChild}(v,x,t)$ は，vnode v^l が対応するグラフの辺を選択する・あるいは選択しないことによるフロンティア頂点の状態の変化を計算して，子 znode のラベルを作成する．この処理は点素パスの集合を表す ZDD を構築するための SimPath アルゴリズムにおける ZSDD 頂点作成のための手続きとほぼ等しい．

$\mathsf{decompChild}(v,x)$ はマッチング集合の場合と同様に，可能な主部と副部のフロンティア頂点の値の組合せを全て列挙して子要素 znode 群を生成する．可能なパターンの列挙は，まず (a)$F(v^l)$ と $F(v^r)$ に共通して含まれる頂点の整合性のとれる主部と副部の組合せをすべて列挙し，さらに (b) それぞれの主部と副部の組合せについて，v^l, v^r にそれぞれ端点をもつようなパスの接続方法を全て列挙することで行う．(a) はマッチングの場合と同様に，各共通フロンティア節点 u_i について，$m[i]$ の値にもとづいて子の主部，副部のフロンティア $m_p[i]$, $m_s[i]$ の可能な値を定め，その組合せをすべて列挙する処理である．

(b) の処理については例を用いて説明する．いま，図 5 (a) の破線で囲まれているフロンティア頂点 $u_1, \ldots, u_7$ があり，u_1 は s と，u_2, u_3, u_4 はそれぞれ u_7, u_6, u_5 とパスで結ばれていたとする．このフロンティアを主部 $u_1, \ldots, u_4$, 副部 $u_4, \ldots, u_7$ に分割すると，(u_2, u_7), (u_3, u_6), (u_4, u_5) の間で主部と副部とをつなぐパスが存在することになる．

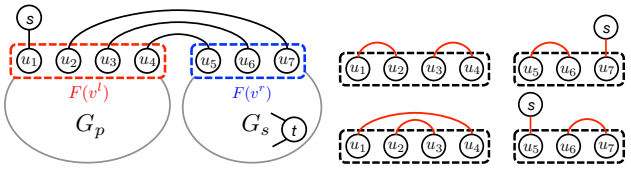


図 5: decompChild() における，接続パターンを考慮したラベル更新の例．(a) 親 znode におけるラベル (b) 可能な主部・副部の状態の組合せ．

このパスが存在する限り，v^l に対応する部分グラフ G_p での辺の選択が G_s での辺の選択に影響を与えることになるため，それぞれの部分 ZSDD を独立に構築することができない．そこで，主部の端点間での可能な接続パターンを列挙し，そのパターンの数だけ子を生成することにする．主部では，(u_1, u_2) と (u_3, u_4) を接続するパターンと (u_1, u_4) と (u_2, u_3) をそれぞれ接続するパターンとが考えられる．それぞれのパターンにもとづいて副部のフロンティアを計算したものが図 5 (b) になる．たとえば，上の例では主部で (u_1, u_2) が接続されることで u_7 に s が接続されることになるため，副部のフロンティア頂点の値に反映している．主部では以降 (u_1, u_2) と (u_3, u_4) 間に点素パスが存在するように処理を継続し，副部でも同様にフロンティアパターンに合致した集合族を見つけるよう独立して処理を進める．

7. 検証

いくつかのベンチマークデータセットを用いてマッチング集合，点素パス集合を表す決定図構築における提案法の性能を検証する．ベースラインとして，Apply 演算に基づく ZSDD のボトムアップ構築法および ZDD を構築する SimPath アルゴリズムを用いる．ただし，ボトムアップ構築法は，入力となる CNF が現実的なサイズになるマッチング集合の構築でのみ利用した．vtree は [2] で提案された，クラスタリングに基づくヒューリスティクスによって得られた Branch decomposition から生成した．SimPath アルゴリズムで採用する変数順序として，ZDD のトップダウン構築アルゴリズムを実装したライブラリ Graphillion [5] で採用されている幅優先探索による順序づけと，ZSDD 構築のための vtree を行きがけ順かつ部分木のサイズが小さい順に深さ優先探索を行うことによって得られた変数順序の 2 種類を利用した．

検証用のデータセットしては，文献 [2] で利用された TSPLIB のデータセットをドロネー分割することによって得られた無向グラフと，RomeGraph [*1] データセットを用いた．なお，RomeGraph データセットには膨大な数のデータセットが含まれているため，頂点数が 100 頂点のデータのうち，辞書順で最初に来る 10 サンプルを利用した．なお，全ての手法で 10 分以内に処理が終わらなかったデータセットは除外してある．全ての手法は C++によって実装され，Xeon E5-2687W 3.10 Ghz，128GB RAM を搭載した Linux 計算機上で実験を行った．

検証結果を表に示す．ここで BU は Apply 演算による ZSDD のボトムアップ構築法，TD は提案法，Z(b), Z(v) はそれぞれ ZDD のトップダウン構築法であり，それぞれ幅優先探索由来の変数順序を採用したものと，vtree の深さ優先探索による変数順序を採用したものの結果となる．表より，マッチングの集合の構築においては提案法によるトップダウン構築が最も高速であり，かつ ZDD と比べると常に小さな ZSDD を得ることができている．なお，ボトムアップ構築とトップダウン構築とを比較すると常にトップダウン構築のほうが高速であった．構築された ZSDD のサイズはトップダウン構築のほうがやや大きい．これは，ボトムアップ構築では常に圧縮された ZSDD が出力されているのに対し，トップダウン構築では ZSDD の圧縮は行われていないためである．Apply 演算を用いることでトップダウン構築された ZSDD も圧縮可能であり，圧縮後はボトムアップ構築とサイズが一致する．圧縮にかかる時間は ZSDD のサイズに依存する．しかし今回はいずれの場合でも BU と TD でサイズに大きな差は出なかったことから効率的に圧縮可能であると考えられる．点素パスについても，ほぼ全てのインスタンスで提案法のほうが高速かつ構築された決定図のサイズが小さくなった．

8. おわりに

本稿ではグラフ部分構造の集合を表現する ZSDD のトップダウン構築法を示した．提案法はグラフ部分構造の集合を表す ZDD を構築するための SimPath アルゴリズムを拡張したものである．既存の Branch Decomposion ヒューリスティック由来の vtree を用いることで，ZDD よりも小さな ZSDD をより高速に構築できることを示した．SimPath アルゴリズムが様々なグラフ部分構造を表現する ZDD の構築に用いることができるのと同様に，提案するトップダウン構築法も今回紹介したマッチングや点素パス以外の部分構造にも適用可能である．今後はこれらの変種に対応するための拡張および実問題への応用を検討したい．

[*1] http://www.graphdrawing.org/download/rome-graphml.tgz

参考文献

[1] Bryant, R. E.: Graph-based algorithms for boolean function manipulation, *Computers, IEEE Transactions on*, Vol. C-35, No. 8, pp. 677–691 (1986).
[2] Cook, W. and Seymour, P.: Tour Merging via Branch-Decomposition, *INFORMS J. on Computing*, Vol. 15, No. 3, pp. 233–248 (2003).
[3] Darwiche, A.: SDD: A New Canonical Representation

表 1: マッチング集合を表す決定図の構築実験結果.

			構築時間 (ミリ秒)				サイズ			
Instance	$\|V\|$	$\|E\|$	BU	TD	Z (b)	Z (v)	BU	TD	Z (b)	Z (v)
att48	48	130	95	9	132	35	7,420	7,550	40,370	20,438
berlin52	52	145	542	17	7,676	158	16,043	16,380	520,466	79,726
eil51	51	142	443	16	1,371	217	16,303	16,882	366,273	101,306
eil76	76	215	1,888	105	72,474	11,019	103,317	103,915	8,253,872	1,152,860
eil101	101	290	17,095	239	–	66,436	177,932	183,123	–	10,027,975
pr226	226	660	19,637	71	–	155,318	26,832	26,938	–	16,027,488
rat99	99	280	2,147	57	–	18,920	37,421	39,143	–	2,157,450
st70	70	197	1,418	45	62,559	70,336	46,288	47,275	3,861,677	7,244,529
grafo10106.100	100	119	94	1	371	4	885	890	141,278	2,690
grafo10116.100	100	149	481	195	–	5,590	108,637	112,709	–	853,204
grafo10124.100	100	139	469	89	59,927	385	71,210	71,819	5,664,264	187,536
grafo10153.100	100	136	210	28	29,943	119	21,164	21,332	4,462,425	55,647
grafo10183.100	100	132	108	8	29,508	658	6,094	6,191	3,513,816	213,194

表 2: 点素パス集合を表す決定図の構築実験結果.

			構築時間 (ミリ秒)			サイズ		
Instance	$\|V\|$	$\|E\|$	TD	Z (b)	Z (v)	TD	Z (b)	Z (v)
att48	48	130	544	20,679	1,149	118,718	991,456	240,068
berlin52	52	145	1,892	–	16,225	450,849	–	1,305,617
eil51	51	142	1,556	–	27,873	254,678	–	2,742,091
ulysses22	22	57	2	183	11	1,117	15,649	5,377
grafo10106.100	100	119	2	1,914	4	466	7,861	1,056
grafo10124.100	100	139	34,915	–	60,099	2,531,651	–	4,694,606
grafo10153.100	100	136	14,442	–	3124	995,739	–	432,263
grafo10183.100	100	132	239	–	140,348	35,364	–	364,206
grafo10184.100	100	140	14,503	–	113,433	878,067	–	2,205,368

of Propositional Knowledge Bases, *IJCAI*, pp. 819–826 (2011).

[4] Hardy, G., Lucet, C. and Limnios, N.: K-terminal network reliability measures with binary decision diagrams, *Reliability, IEEE Transactions on*, Vol. 56, No. 3, pp. 506–515 (2007).

[5] Inoue, T., Iwashita, H., Kawahara, J. and Minato, S.-i.: Graphillion: software library for very large sets of labeled graphs, *International Journal on Software Tools for Technology Transfer*, Vol. 18, No. 1, pp. 57–66 (2016).

[6] Inoue, T., Takano, K., Watanabe, T., Kawahara, J., Yoshinaka, R., Kishimoto, A., Tsuda, K., Minato, S. and Hayashi, Y.: Distribution loss minimization with guaranteed error bound, *Smart Grid, IEEE Transactions on*, Vol. 5, No. 1, pp. 102–111 (2014).

[7] Knuth, D. E.: *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*, Addison-Wesley (2011).

[8] Minato, S.: Zero-suppressed BDDs for Set Manipulation in Combinatorial Problems, *DAC*, pp. 272–277 (1993).

[9] Morrison, D. R., Sewell, E. C. and Jacobson, S. H.: Solving the Pricing Problem in a Branch-and-Price Algorithm for Graph Coloring Using Zero-Suppressed Binary Decision Diagrams, *INFORMS Journal on Computing*, Vol. 28, No. 1, pp. 67–82 (2016).

[10] Nishino, M., Yasuda, N., Minato, S. and Nagata, M.: Zero-suppressed Sentential Decision Diagrams, *AAAI*, pp. 1058–1066 (2016).

[11] Robertson, N. and Seymour, P. D.: Graph minors. X. Obstructions to tree-decomposition, *Journal of Combinatorial Theory, Series B*, Vol. 52, No. 2, pp. 153–190 (1991).