グラフ列挙索引化技法の種々の問題への適用

川原　純，湊　真一

本記事では，フロンティア法による s–t パス列挙の技法を，s–t パスだけではなく，全域木やマッチング，集合被覆などにも適用できることを示す．最初に，フロンティア法を用いて，与えられたグラフのすべての全域木の集合や，すべての根付き全域森の集合を表現する ZDD を構築する手法について述べる．次に，s–t パスや全域木の場合を一般化する形で，フロンティア法が適用可能な種々の問題を分類，整理して，統一的な視点から述べる．

キーワード：アルゴリズム，ZDD，全域木，根付き全域森，列挙アルゴリズム

1. はじめに

前記事 [5] では，フロンティア法（Simpath アルゴリズム）と呼ばれる，グラフ上の指定された 2 点間の s–t パス全体を表現する ZDD の構築アルゴリズムについて述べられている．フロンティア法では，グラフの各辺を 0,1 変数とする ZDD を，頂上から下に向かって直接的に構築する．その際に，ZDD の各ノードに，自身より下位の部分グラフの構築に必要な情報を記憶しておき，記憶した情報を基に，ノードの等価性判定を行い，ノードの共有を可能な限り行う．また，早い段階での枝刈り（0,1–終端への接続）を行う．ZDD 構築において，s–t パス列挙に特有の箇所は，ノードに記憶させる情報と，ノードの等価性判定，枝刈り判定のみであり，これらの部分をサブルーチン的に入れ替えることによって，全域木やマッチングなど，さまざまな部分グラフの集合を表現する ZDD の構築が可能となる．本記事では，フロンティア法をさまざまな対象へ適用する手法について，前記事の続きとして述べる．

2. 全域木の集合を表現する ZDD

フロンティア法の一般化について述べる前に，別の例として，全域木の集合を表現する ZDD を構築する手法について述べる．以下では，全域木の集合を表現する ZDD を全域木 ZDD，s–t パスの集合を表現する ZDD を s–t パス ZDD などと呼ぶことにする．s–t パスの場合と全域木の場合を比較することで，フロンティア法の本質について理解がより深まるであろう．関根らは，Tutte 多項式の計算 [6] において，全域木の集合を表現する BDD の構築を行っている．この計算で用いられている全域木 BDD の構築法は，s–t パス ZDD を構築する Knuth の Simpath アルゴリズムと極めて類似している．ここで，BDD/ZDD の違いは（BDD/ZDD 構築の最中は）本質的ではないので，統一のため本記事ではすべて ZDD と呼ぶ．本節では，フロンティア法の枠組みに沿って，関根らの手法を紹介する．

2.1 全域木 ZDD の構築

グラフ $G = (V, E)$ が入力として与えられたとする．ここで，V は G の頂点集合 $\{v_1, \ldots, v_n\}$，E は G の辺集合 $\{e_1, \ldots, e_m\}$ である．辺には変数順 $e_1, \ldots, e_m$ が定められているとする．変数順については議論の余地があるが，ここでは，V から任意に頂点を 1 つ選び，その頂点から幅優先探索的に辺に順 $e_1, \ldots, e_m$ が付けられたと仮定する．G 上の全域木とは，サイクルを持たない G の連結な部分グラフ $G' = (V, H)$，$H \subseteq E$ である．

ZDD の構築手順は基本的には前記事で述べた s–t パスの場合と同じである．ここでは，疑似コードの形で Algorithm 1 に示す．

Algorithm 1 の 2 行目が ZDD の各段についての処理，3 行目がその段の各ノードについての処理である．4 行目が x–枝 $(x = 0, 1)$ の先のノードを作成する処理である．$x = 1$ $(x = 0)$ が，e_i を全域木の辺として採用する場合（採用しない場合）に対応する．5 行目では，終端条件の判定を行う．例えば，s–t パス ZDD の場合は，パスに分岐が生じるなど，s–t パスが完成できないと判定したときに，n の x–枝の先を 0–終端に接続する（詳しくは前記事参照 [5]）．終端に接続する場合は，6–11 行目は実行しない．7 行目ではノードに

かわはら　じゅん
JST ERATO/北海道大学
〒 060–0814 北海道札幌市北区北 14 条西 9 丁目　北海道大学情報科学研究科 C306 号室
みなと　しんいち
北海道大学/JST ERATO

Algorithm 1　ZDD 構築アルゴリズム

1: 1 段目に根ノードを作成
2: **for** $i = 1$ to m **do** // e_i の処理
3:　**for** すでに作成済みの i 段目の各ノード n について **do**
4:　　**for** $x = 0, 1$ **do** // 0–枝，1–枝の処理
5:　　終端条件の判定 (a)
6:　　新しいノード n' を作成（$i+1$ 段目とする）
7:　　n' の情報を更新 (b)
8:　　**if** n' と等価なノード n'' がすでに存在 (c) **then**
9:　　$n' \leftarrow n''$
10:　　**end if**
11:　　n の x–枝の先を n' とする．
12:　**end for**
13:　**end for**
14: **end for**

記憶させる情報（前記事で mate と呼んでいた配列）を更新する．8 行目でノードが共有可能であるか判定を行う．

全域木と s–t パスの違いは (a) 終端条件の判定（5 行目），(b) ノードに記憶させる情報の更新（7 行目），(c) ノードの等価性判定（8 行目）に現れる．以下ではこの 3 点をどのように行うかを見ていく．

G の部分グラフ $G' = (V, H)$ が全域木になるには，以下の (i)，(ii) を満たしていればよい．(i) G' はサイクルを持たない，(ii) G' は 2 つ以上の連結成分を持たない．(a) の終端条件の判定では，各ノードの x–枝 $(x = 0, 1)$ の先を作成する最中に，(i), (ii) をそれぞれ判定し，いずれかを満たさないと判定した場合は，そのノードの x–枝の先を 0–終端に接続する．最終段のノード $(i = m)$ の処理で，(i), (ii) を満たすと判定した場合，全域木になることが確定するので，ノードの x–枝の先を 1–終端に接続する．(i)，(ii) の状況の例を図 1 に示す．図 1 の左では，e を全域木の辺として採用するとき，サイクルが生じるので，(i) が満たされない．図 1 の右では，e を全域木の辺として採用しないとき，左側の連結成分は右側とはつながらないことが確定し，(ii) が満たされないことがわかる．このことを「連結成分が切り離される」と言うことにする．

(i) と (ii) の判定のために，各頂点がどの連結成分に含まれているかを記憶する．具体的には，すべての頂点対 v, w について，v と w が同じ連結成分に含まれるか否かを記憶する．$e = \{v, w\}$ を全域木の辺として採用する場合，v と w が同じ連結成分に含まれているなら，v と w を接続するとサイクルが生じるので，(i) の条件が満たされなくなることが判定できる．(ii) の条件判定も可能であるが，これに関しては後述する．

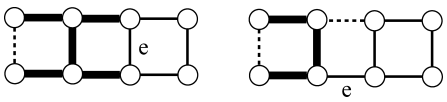

図 1　(i), (ii) が満たされなくなる瞬間の例．太線は全域木の辺として採用された辺，点線は全域木の辺として採用されなかった辺，細線は未処理の辺を表す．

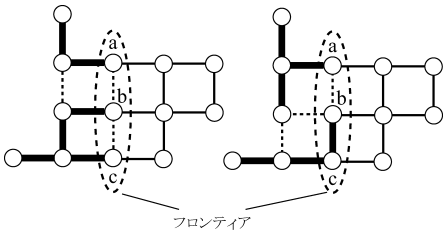

図 2　等価な途中状態とフロンティアの例．太線，点線，細線の用法は前の図と同様．

v と w が異なる連結成分に属するとき，v と w を接続すると，2 つの連結成分は 1 つに結合され，それ以外の連結成分は変化しない．このことから，ある段階における連結成分の情報だけから，$e = \{v, w\}$ を加えた後の連結成分の情報を更新できる．以上の更新が Algorithm 1 の (b) で行われる．

全域木 ZDD の場合も，s–t パスの場合と同様にフロンティアを考えることができる．図 2 の 2 つのグラフにおいて，頂点 b と c は同じ連結成分に属し，頂点 a とは異なる連結成分に属する．詳細は省略するが，フロンティア上以外の頂点については，どの連結成分に属するか記憶しなくてよい．この 2 つのグラフに対応する 2 つのノードは等価である（2 つのノードの下位の部分グラフが一致する）ので，ノードの共有を行う．以上の判定が Algorithm 1 の (c) で行われる．

2.2　全域木 ZDD 構築の実装

以上を踏まえて，ノードに記憶させる情報について，実装レベルの話を述べる．連結成分を記憶するための，part 配列を導入する．part 配列は，（フロンティア上の）各頂点が属する連結成分の番号を記憶する．連結成分の番号は $1, \ldots, n$ の値を取る．最初はすべての頂点は孤立しているので，$j = 1, \ldots, n$ について，$part[v_j] = j$ と設定する．

すべての頂点 $v, w \in V$ について，v と w は同じ連結成分に属する，また，そのときに限り $part[v] = part[w]$ が成り立つ．すなわち，2 つの頂点 v, w が同じ連結成

分に属するなら part 配列の値は同じであり，異なる連結成分に属するなら，part 配列の値は異なる．

フロンティア法が構築する ZDD の各ノードには，part 配列を記憶させる．ノード n の part 配列を $n.part$ と表記する．ノード n の 0–枝，1–枝が指すノードをそれぞれ n_0，n_1 とする．Algorithm 1 の (b) では，$n_0.part$ と $n_1.part$ を $n.part$ から計算している．この詳細について述べる．

n において，辺 $e = \{v, w\}$ を全域木の辺として採用しない場合は，part 配列は変化しない．したがって，すべての頂点 u について，$n_0.part[u] \leftarrow n.part[u]$ を実行すればよい．

n において，辺 $e = \{v, w\}$ を全域木の辺として採用する場合は，$n.part[v] = n.part[w]$ のとき，v と w はすでに同じ連結成分に属し，サイクルが生じるので，n の 1–枝の先を 0–終端に接続する（以下，このことを「0–終端へ遷移する」と言うことにする．1–終端の場合も同様）．$n.part[v] \neq n.part[w]$ のとき，v と w は異なる連結成分に属するが，それらが結合される．$a = \min\{n.part[v], n.part[w]\}$，$b = \max\{n.part[v], n.part[w]\}$ とすると，各頂点 v について，以下を実行する．

$$n_1.part[v] \leftarrow \begin{cases} a & \text{if } n.part[v] = b \\ n.part[v] & \text{otherwise.} \end{cases}$$

すなわち，大きいほうの連結成分の番号の頂点を，すべて小さいほうの連結成分の番号に置き換える．

同じ段の 2 つのノード n, n' に割り当てられた part 配列について，フロンティア上のすべての頂点の part 配列値が一致するとき，n と n' は等価であると定義する．(c) の等価性判定はこの定義に基づいて行う．実際には，part 配列値を比較するために，値の「ソート」（連結成分の番号が小さい順に現れるように番号を付け替える）が必要になるが，詳しくは省略する．

(ii) の条件判定について具体的に述べる．フロンティア上の頂点 u が，フロンティアから外れる場合，u 以外のフロンティア上の任意の頂点 v について，$part[u] \neq part[v]$ であるならば，u が属する連結成分は切り離される．この場合，0–終端に遷移する．

3. 根付き全域森の集合を表現する ZDD

本特集の「フロンティア法による電力網構成制御」では，電力網の設計のため，根付き全域森の集合を表現する ZDD を構築している．本節では，根付き全域森に対するフロンティア法について述べる．グラフ $G = (V, E)$ と，根と呼ばれる，V 上の頂点 $r_1, \ldots, r_\ell$ が与えられたとき，G 上の根付き全域森とは，以下の条件を満たす G の部分グラフ $G' = (V, E')$，$E' \subseteq E$ である．(i) G' はサイクルを持たない，(ii) G' の各連結成分は，ちょうど 1 つの根を含む．G 上のすべての根付き全域森 ZDD を構築する問題を考える．

根付き全域森 ZDD の構築は，全域木の場合とほとんど同じである．各頂点が属する連結成分がどの根を含んでいるか，またはいずれの根も含まないかを記憶すればよい．ある頂点 $v \in V$ と，正の整数 a について，v と r_a が同じ連結成分に属するならば，$part[v] = -a$ とする．$part[v]$ が正の値の場合は，全域木の場合の定義と同じである．すなわち，part 配列値の符号によって，その頂点の属する連結成分が根を含むかどうかを示す．

$part[v]$ の初期値は，$v = r_a$ のときは $part[v] = -a$，そうでないときは，$v = v_i$ であるような v について，$part[v] = i$ とする．part 配列の更新の手順を以下のように変更する．辺 $e = \{v, w\}$ を加えるとき，$part[v] = part[w]$ ならば，v と w はすでに同じ連結成分に属し，サイクルが生じるため，0–終端に遷移する．$part[v] < 0$ かつ $part[w] < 0$ かつ $part[v] \neq part[w]$ ならば，v, w は異なる連結成分に属し，それぞれは異なる根を含み，2 つの連結成分が結合されるので，根を 2 つ含むようになるため，根付き全域森ではなくなる．この場合は 0–終端に遷移する．それ以外の場合は，part 配列の更新式は全域木の場合と同様である．ここで，part 配列の更新式では，辺 e の両端の 2 つの連結成分のうち，part 配列値の大きいほうを小さいほうで置き換えられることに注意されたい．$part[v] < 0$ かつ $part[w] > 0$ のとき，w を含む連結成分の part 配列値は $part[v]$ に置き換えられる．これは，元は w を含む連結成分は根を含まず，e を加えたあとは根を含むことを意味する．

切り離された連結成分の扱いは，part 配列値が正である連結成分のときは，全域木の場合と同様である（0–終端に遷移する）．part 配列値が負である連結成分のとき，その連結成分は根を 1 つ含むので，制約に違反することはないので，0–終端に遷移しない．

4. フロンティア法の汎用化

前節で述べた Algorithm 1 は，s–t パスや全域木に限らず，さまざまな種類の部分グラフを列挙できる．具体的には，Algorithm 1 の (a), (b), (c) を，対象に応じて設計すればよい．本節では，フロンティア法で構

築可能な ZDD の種類を分類しながら述べる．

(c) で ZDD のノードの等価性判定を行う際，判定に必要な情報を，ZDD の各ノードに記憶させることによって，子ノードを作成することなく，また，親ノードをたどることなく判定が可能となる．この情報を一般に *configuration*（様相）と呼ぶことにする．s–t パス ZDD では，*configuration* は mate 配列，すなわち端点対の情報であり，全域木 ZDD では *configuration* は part 配列，すなわち頂点分割の情報である．その他の情報を *configuration* に記憶させることもある．

4.1 パス系問題

s–t パスに類似した問題について考える．これは，*configuration* として mate 配列を記憶する問題であると分類できる．s–t パス以外にも以下の部分グラフ ZDD の構築が可能である．

(1) 複数端点対パス
(2) サイクル
(3) 複数サイクル
(4) ハミルトン s–t パス，サイクル
(5) 1 対多点間パス
(6) 全点間対パス
(7) 有向グラフ上の有向 s–t パス

これらの問題では *configuration* として mate 配列と，個々の問題に応じた情報を記憶する．終端判定条件と，mate 配列の更新は問題に応じて細部が異なる．それぞれの概略についてだけ述べ，詳細は省略する．

(1) 複数端点対パス [7]

複数端点対パスは，グラフ $G=(V,E)$ と，G 上の l 個の端点対 $(s_1,t_1),(s_2,t_2),\ldots,(s_l,t_l)$ が与えられたとき，互いに点素な l 個の s_i–t_i パスを求める問題である．s–t パス ZDD の場合と比べて，終端判定の条件が異なる．以下の (i), (ii), (iii) のいずれかが成り立つときは 0–終端に遷移する．(i) 部分パスの端点となるべき頂点 s_i や t_i が部分パスの中点となる，(ii) $i\neq j$ のとき，s_i と t_j が同一部分パスの端点となる．(iii) 頂点 s_i または t_i がフロンティアから除かれるとき，それが部分パスの端点となっていない．これらの条件は mate 配列を用いて簡単に判定することができる．

(2) サイクル [4, Exercise 226]

サイクル ZDD の構築は，s–t パス ZDD の場合から，0–終端，1–終端に遷移する条件を変更するだけである．2 つの頂点 v,w について，v,w を部分パスの両端とし，辺 $e_i=\{v,w\}$ を採用する状況を考える．このとき，サイクルが完成する．v,w 以外のフロンティア上のある頂点 u が別の部分パスの端点になっているならば，サイクル以外にも余分なパスが存在するので，0–終端に遷移する．そうでなければ，e_{i+1} 以降の辺を加えなければ 1 つのサイクルが完成するので，1–終端に遷移する．

(3) 複数サイクル

(2) と考え方は同様である．ただし，$e_i=\{v,w\}$ を追加して，サイクルが完成する場合でも，0–終端または 1–終端に遷移しない．e_i の処理後，フロンティア上の頂点 u が，フロンティアから除かれる場合，u がある部分パスの端点になっているならば，今後，u から辺を伸ばしていく余地がなくなることになり，サイクルは完成し得ないので，この場合は 0–終端に遷移する．辺を最後まで処理し終えて，0–終端に遷移しなければ，1–終端に遷移する．

(4) ハミルトン s–t パス，サイクル [4, Exercise 227]

すべての頂点を使用した s–t パスをハミルトン s–t パス，すべての頂点を使用したサイクルをハミルトンサイクルという．ハミルトン s–t パス，サイクルの ZDD の構築には，s–t パス ZDD やサイクル ZDD の方法に，ハミルトン性の判定を加えればよい．フロンティア上の頂点 u が，フロンティアから外れる場合，u が孤立した頂点（いずれの部分パスにも含まれない頂点）であるならば，今後，u はパスまたはサイクルに含まれる可能性がなくなるので，0–終端に遷移する．

(5) 1 対多点間パス [4, Exercise 228]，(6) 全点間対パス

G 上の頂点 s が与えられたとき，s の 1 対多点間パスとは，s から任意の点の間のパスである．全点間対パスとは，G 上の任意の 2 点間のパスである．1 対多点間パス ZDD を構築する場合，ダミーの終点 u を用意し，$V\setminus\{u\}$ 上の任意の点から u への辺を張ったグラフ G' について，s–u パス ZDD を構築すればよい．全点間対パス ZDD を構築する場合，2 つのダミーの点 u,v を用意し，V 上の任意の点から u,v へそれぞれ辺を張ったグラフ G'' について，u–v パス ZDD を構築すればよい．

(7) 有向グラフ上の有向 s–t パス [4, Exercise 233]

有向グラフについても，フロンティア法の枠組みはそのまま適用できる．mate 配列として，部分パスの両端だけでなく，方向も記憶する．詳しくは [4, Exercise 233] を参照せよ．

4.2 森系問題

全域木，根付き全域森以外にも，フロンティア法によってさまざまな部分グラフ ZDD を構築できる．これは，*configuration* として part 配列を記憶するタイ

プの問題であると分類できる．

(1) 森

(2) 連結部分グラフ，m–連結成分

(3) カット，s–t カット，k–分割カット

(1) 森

全域木の場合から連結性判定（2 節の (ii)）を除くと森 ZDD になる．

(2) 連結部分グラフ [4, Exercise 55]，m–連結成分

全域木の場合からサイクル判定（2 節の (i)）を除くと連結部分グラフ ZDD になる．m–連結成分 ZDD を構築する場合は，切り離された連結成分の数を数えるカウンタを用意し，*configuration* として記憶すればよい．ある辺の処理中に連結成分が切り離されたとき，全域木 ZDD の場合は 0–終端へ遷移したが，G に対する m–連結成分とは，m 個の連結成分をもつ G の部分グラフである．m–連結成分 ZDD の場合は，0–終端へ遷移せずに，カウンタを 1 増やす．等価性判定では，part 配列の判定に加えて，カウンタの一致も調べる．すべての辺を処理した後，連結成分の数が m となれば 1–終端へ，そうでなければ 0–終端へ遷移する．容易にわかるように，少なくとも m 個の連結成分や，高々 m 個の連結成分が存在するような部分グラフに対しても ZDD 構築が可能である．

(3) カット，s–t カット，k–分割カット

グラフ $G=(V,E)$ において，辺集合 $H \subseteq E$ を G から除いたときに，G が連結でなくなるとき，H をカット（セット）と言う．辺集合 $H \subseteq E$ を G から除いたときに，G の 2 点 s,t が分断される（異なる連結成分に属する）とき，H を s–t カット（セット）と言う．辺集合 $H \subseteq E$ を G から除いたときに，連結成分が k 個生じるとき，H を k–分割カット（セット）と言う．カット ZDD の構築は，(2) の連結部分グラフ ZDD の場合の 0–枝と 1–枝の役割を変えたものと本質的に同じである．s–t カット ZDD の場合は，s と t が同じ連結成分に含まれているかどうかを判定すればよい．そのために s や t を含む連結成分の番号をそれぞれ記憶すればよい．k–分割カット ZDD の場合は m–連結成分の場合と同様に，切り離された連結成分の個数を記憶する．

5. ハイパーグラフに対するフロンティア法

ハイパーグラフは，グラフを拡張したもので，各辺 e_i が V 上の（2 個とは限らず）任意個の頂点に接続する．すなわち，頂点集合 $V=\{v_1,\ldots,v_n\}$，辺集合 $E=\{e_1,\ldots,e_m\}$，$e_i \in 2^V$ であるとき，$H=(V,E)$ をハイパーグラフと言う．本稿では，E の要素を（ハイパー辺とは呼ばず）単に辺と呼ぶ．

ハイパーグラフに対しても，フロンティア法と同じ枠組みで，ある性質をもった H の部分ハイパーグラフ $H'=(V,E')$, $E' \subseteq E$ の ZDD を構築できる．以下の問題が考えられる [8].

(1) 集合被覆

(2) 集合パッキング

(3) 集合分割

ハイパーグラフ $H=(V,E)$ に対する集合被覆とは $\bigcup_{e\in E'} e = V$ となるような辺の部分集合 $E' \subseteq E$ のことである．ハイパーグラフ $H=(V,E)$ に対する集合パッキングとは任意の $e,e' \in E'$ について，$e \cap e' = \emptyset$ となるような辺の部分集合 $E' \subseteq E$ のことである．ハイパーグラフ $H=(V,E)$ に対する集合分割とは集合被覆かつ集合パッキングとなるような $E' \subseteq E$ のことである．

ハイパーグラフにおける集合被覆 ZDD，集合パッキング ZDD，集合分割 ZDD をフロンティア法で構築する方法について述べる．グラフのフロンティア法と同様に，各辺 $e_1,\ldots,e_m$ について，辺を E' に加えるかどうか取捨選択すればよい．ハイパーグラフのフロンティアについては，グラフのフロンティアと同様に定義できる．*configuration* として，各フロンティア上の頂点 v について，v がすでにある辺によって被覆されているかどうかを記憶すればよい．それ以外は，対象によって動作が異なる．

(1) 集合被覆

フロンティア上の頂点 u が，フロンティアから外れる場合，u がいずれの辺によっても被覆されていない場合，0–終端に遷移する．

(2) 集合パッキング

e_i を採用するとき，e_i に含まれるある頂点がすでに被覆されているならば 0–終端に遷移する．

(3) 集合分割

(1) と (2) の条件判定を両方行えばよい．

以下の (1′), (2′), (3′) は，(1), (2), (3) の（通常の）グラフ版である．

(1′) 辺被覆

(2′) マッチング

(3′) 完全マッチング

6. グラフ以外の問題に対する ZDD 構築

フロンティア法のように，0, 1 変数の取捨選択を分岐して，等価な状態を共有して ZDD を構築する手法は，

与えられたグラフの部分グラフを表現する ZDD を構築する問題だけでなく，さまざまな問題に応用できる可能性をもつ．単純な例の 1 つとして，0–1 ナップザック問題が挙げられる．アイテム $e_1, \ldots, e_m$ と，それぞれの重さ $w_1, \ldots, w_m$，制限容量 W が与えられたとき，重さの総和が W を越えないアイテムの組合せを全列挙する問題である（一般には選択したアイテムの総和を最大化する問題を考えることが多い）．*configuration* として，それまで取得したアイテムの重みの総和を記憶し，総和が等しいノードは共有する．総和が W を越えると 0–終端に遷移し，アイテムをすべて取捨選択し終えると 1–終端に遷移する．

0–1 ナップザック問題の場合，各アイテム変数 e_i は，他のアイテム変数と関連をもたない．グラフ上の辺と頂点のように，辺を取捨選択する際，頂点に情報を記憶するというフロンティア法の考えとは異なるが，ここでは広義のフロンティア法として扱うことにする．

この方法による ZDD の構築は，動的計画法によってナップザック問題を解く場合に現れる表を構築することとみなすことができる．

この種類の問題として，以下の問題が挙げられる．詳細は省略する．

(1) 独立集合 [2]
(2) 支配集合
(3) クリーク
(4) 頂点彩色
(5) 部分和問題
(6) 行列積問題
(7) マトロイド [3]

(1)–(4) はグラフ上の問題であるが，グラフの辺ではなくて頂点を変数とする ZDD を構築する．前節までで述べたグラフ上のフロンティア法では，辺を変数として，辺を取捨選択したときの情報を頂点に記憶しておくことで，ZDD ノードを多く共有できるようになるが，(1)–(4) の問題は，頂点の取捨選択の情報を頂点に記憶することになるので，ZDD ノードの共有は効率良く行われるようになるとは限らない．例えば (3) のクリーク列挙では，Apply 演算を用いた手法 [1] と，フロンティア法を比べると，構築にかかる時間は実験的にそれほど差はない．[1] のアルゴリズムは，再帰を用いて，既存の ZDD パッケージの演算をいくつか組み合わせるだけで実現しており，フロンティア法よりも容易に実装ができる．

7. おわりに

フロンティア法によって，さまざまな組合せ問題に対して，全解を表現する ZDD が構築可能であることを見てきた．フロンティア法は，*configuration* とノードの等価性判定，枝刈り判定を少し変えるだけで，さまざまな問題に対応できることがわかる．目的に応じて細かな条件を加えて，柔軟にカスタマイズできる点が長所である．グラフ以外のフロンティア法については効率性や有効性の面で未解明の部分も多い．構築した ZDD の活用については，本特集の他の記事を参照されたい．

参考文献

[1] O. Coudert, Solving graph optimization problems with ZBDDs, In *Proceedings of the 1997 European conference on Design and Test*, *EDTC '97*, pp. 224–228, IEEE Computer Society, Washington, DC, USA, 1997.

[2] K. Hayase, K. Sadakane and S. Tani, Output-size sensitiveness of OBDD construction through maximal independent set problem. In *Lecture Notes in Computer Science*, 959, pp. 229–234, 1995.

[3] H. Imai, S. Iwata, K. Sekine and K. Yoshida, Combinatorial and geometric approaches to counting problems on linear matroids, graphic arrangements and partial orders. In *Lecture Notes in Computer Science*, 1090, pp. 68–80. Springer, 1996.

[4] D. E. Knuth, *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms, Part 1*, Addison-Wesley Professional, 1st ed., March 2011.

[5] 湊真一，BDD/ZDD を用いたグラフ列挙索引化技法，オペレーションズ・リサーチ，Vol. 57, No. 11, pp. 597–603, 2012.

[6] K. Sekine, H. Imai and S. Tani, Computing the Tutte polynomial of a graph of moderate size, In *Proceedings of the 6th International Symposium on Algorithms and Computation* (*ISAAC*), pp. 224–233, 1995.

[7] R. Yoshinaka, T. Saitoh, J. Kawahara, K. Tsuruma, H. Iwashita, and S. Minato, Finding all solutions and instances of numberlink and slitherlink by ZDDs, *Algorithms*, Vol. 5, No. 2, pp. 176–213, 2012.

[8] 今井浩，今井桂子，BDD による計算代数・計算幾何の不変量計算（アルゴリズムと計算の理論），数理解析研究所講究録，Vol. 1041, pp. 12–18, 1998–04.