Vol.2018-AL-166 No.3
2018/1/28

グラフ分割集合を表すZDDに対する連結成分重み下限制約

中畑 裕[1]　川原 純[1]　笠原 正治[1,a)]

概要：ゼロサプレス型二分決定グラフ（ZDD）は集合族を圧縮して表現するデータ構造である．ZDD は豊富な演算体系を持ち，集合族に対する様々なクエリが ZDD を解凍することなく行える．また，グラフが与えられたとき，指定した制約を満たす部分グラフの集合を表す ZDD を効率的に構築する手法が存在し，フロンティア法と呼ばれている．本稿では，ZDD やフロンティア法の価値をより高めるための新たな操作を提案する．それは，グラフ分割集合を表す ZDD に対し，禁止したい複数の連結な部分グラフを指定し，それらのうちのどれも連結成分として含まないようなグラフ分割すべての集合を表す ZDD を得る手法である．また，本手法を応用することで，グラフ分割集合を表す ZDD から連結成分の重みが一定値以上であるようなグラフ分割のみを取り出し，それらすべての集合を表す ZDD を構築することが可能であることを示す．計算機実験により提案手法の性能を評価する．

キーワード：グラフ分割，グラフアルゴリズム，ゼロサプレス型二分決定グラフ (ZDD)，フロンティア法

Extracting Graph Partitions Without Too Small Connected Components from a ZDD

Yu Nakahata[1]　Jun Kawahara[1]　Shoji Kasahara[1,a)]

1. はじめに

ゼロサプレス型二分決定グラフ（ZDD） [1] は集合族を圧縮して表現するデータ構造である．ZDD は集合族に対する様々なクエリを処理できる．例えば，集合族に属する集合の数を求める，要素の重みの線形和が最大・最小となるものを取り出す，ランダムにサンプリングを行うといった操作が ZDD の大きさに対する線形時間で行える．また，2 つの ZDD が与えられたとき，それらが表す集合族の和集合，積集合，差集合を表す ZDD の構築が，与えられた 2 つの ZDD の大きさの積に比例する時間で行える．

グラフが与えられたとき，制約を満たす部分グラフの集合を表す ZDD を効率よく構築する枠組みが存在し，フロンティア法 [2], [3], [4] と呼ばれている．フロンティア法では，パス，サイクル，木など様々な構造の制約を扱える．この枠組みを用いて，Inoue ら [5] は根付き全域森を表す ZDD を構築し，それを配電網の電力ロス最適化問題に応用する手法を提案した．Yoshinaka ら [6] はいくつかのペンシルパズルの求解を，フロンティア法によって ZDD を構築することで行う手法を提案した．また，パズル問題を生成するアルゴリズムも提案している．他にも，信頼性評価 [7], [8] や Web の影響拡散の厳密計算 [9]，一票の格差の小さい選挙区割りの列挙 [10] などの応用が存在する．

本稿ではフロンティア法や ZDD の価値を高めるための新たな操作を提案する．それは，グラフ分割集合を表す ZDD に対し，禁止したい複数の連結な部分グラフを指定し，それらのうちのどれも連結成分として含まないようなグラフ分割すべての集合を表す ZDD を得る手法である．また，本手法を応用することで，グラフ分割集合を表す ZDD から連結成分の重みが一定値以上であるようなグラ

[1] 奈良先端科学技術大学院大学 情報科学研究科
Nara Institute of Science and Technology, Graduate School of Information Science
[a)] kasahara@is.naist.jp

フ分割のみを取り出し，それらすべての集合を表す ZDD を構築することが可能であることを示す．計算機実験により提案手法の性能を評価する．

2. 準備

本節では準備として，まず諸定義を与える．次に，本稿で用いるデータ構造であるゼロサプレス型二分決定グラフ（ZDD）と，ゼロサプレス型三分決定グラフ（ZTDD）について説明する．また，これらを効率的に構築する枠組みであるフロンティア法について説明する．

2.1 諸定義

正の整数からなる集合を $\mathbb{Z}^+$ とする．正の整数 k に対し $[k] = \{1, 2, \ldots, k\}$ とする．本稿では頂点重み付き無向グラフ $G = (V, E, p)$ を考える．$V = [n]$ は頂点集合，$E = \{e_1, e_2, \ldots, e_m\} \subseteq \{\{u, v\} \mid u, v \in V\}$ は辺集合を表し，関数 $p\colon V \to \mathbb{Z}^+$ は頂点の重みを与える．本稿では「部分グラフ」と言うとき，$E' \subseteq E$ に対して (V, E', p) のことを指し，E' と (V, E', p) を同一視する．また，部分グラフが複数の連結成分を持ちうることを強調したい場合，本稿では部分グラフのことを「グラフ分割」と呼ぶ．部分グラフ $E' \subseteq E$ における頂点 v の近傍を $N(E', v) = \{u \mid \{u, v\} \in E'\}$ で定める．また，添字が i 以下，i 未満，i 以上，i より大きい辺の集合をそれぞれ $E^{\leq i}$, $E^{<i}$, $E^{\geq i}$, $E^{>i}$ と書く．

集合 U に対し，$U^+ = \{+e \mid e \in U\}, U^- = \{-e \mid e \in U\}, U^\pm = U^+ \cup U^-$ とする．**符号付き集合** [11] とは，$U^\pm$ の部分集合であって，任意の $e \in U$ に対して $+e$ と $-e$ のうち高々 1 つを含むものである．例えば $U = [3]$ のとき $\{+1, -2\}, \{-3\}$ はそれぞれ符号付き集合であるが，$\{+1, -1, +3\}$ は符号付き集合でない．**符号付き集合族** [11] とは，符号付き集合からなる集合である．特に $U = E$ のとき，本稿では符号付き集合を**符号付き部分グラフ**，符号付き集合族を**符号付き部分グラフ集合**ということがある．また，符号付き集合 $S^\pm$ に対し，要素の符号を無視した集合を $\mathrm{abs}(S^\pm) = \{e \mid +e \in S^\pm \vee -e \in S^\pm\}$ とする．

2.2 ゼロサプレス型二分決定グラフ（ZDD）

ゼロサプレス型二分決定グラフ（Zero-suppressed binary decision diagram, ZDD）[1] は，集合族を表現する非巡回有向グラフ $Z = (N_Z, A_Z)$ である．ここで，N_Z は ZDD ノードの集合，A_Z は有向枝の集合である[*1]．N_Z は 2 つの終端ノード $\top, \bot$ と，それ以外の 0 個以上の非終端ノードからなる．非終端ノードはいずれも，自身から出て他のノードに入る枝である 0-枝，1-枝を 1 本ずつ持ち，台集合の要素 1 つに対応するラベルを持つ．ノード α から出る 0-枝，1-枝が入るノードをそれぞれ α の 0-子，1-子と呼び，

[*1] 混乱を避けるため，入力グラフの頂点，辺は「頂点」「辺」と呼び，ZDD, ZTDD の頂点と辺は「ノード」「枝」と呼ぶことにする．

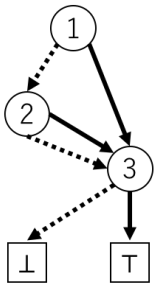

図 1 集合族 $\{\{1,3\},\{2,3\},\{3\}\}$ を表す ZDD．四角は終端ノードを表す．丸は非終端ノードを表し，中の数はラベルを示す．実線は 1-枝，点線は 0-枝を表す．

α のラベルを $l(\alpha)$ と書く．また，Z の 0-枝，1-枝の集合をそれぞれ $A_{Z,0}, A_{Z,1}$ と書く．すなわち $A = A_{Z,0} \cup A_{Z,1}$ である．任意の $(\alpha, \beta) \in A$ に対して，$l(\alpha) < l(\beta)$ が成り立つ．ただし，$l(\top) = l(\bot) = \infty$ とする．非終端ノードのうち入次数 0 のノードが高々 1 つ存在する．これを根ノードと呼び，r_Z と書く．また，Z の非終端ノードの個数を Z のノード数と言い，$|Z|$ と書く．

ZDD Z は次のように集合族を表現する．r_Z から $\top$ に至る有向パスの集合を $\mathcal{P}_Z$ とする．$p = (n_1, a_1, n_2, a_2, \ldots, n_k, a_k, \top) \in \mathcal{P}_Z, n_1 = r_Z$ に対し，$S_p = \{l(n_i) \mid a_i \in A_{Z,1}, i \in [k]\}$ とする．このとき，$\mathcal{S}_Z = \{S_p \mid p \in \mathcal{P}_Z\}$ が Z の表す集合族である．すなわち，r_Z から $\top$ に至る有向パスの 1 つ 1 つが，Z が表す集合族に含まれる集合の 1 つ 1 つに対応する．例として，集合族 $\{\{1,3\},\{2,3\},\{3\}\}$ を表す ZDD を **図 1** に示す．図 1 では根ノードから $\top$ に至る有向パスが 3 つあり，$1 \to 3 \to \top, 1 \dashrightarrow 2 \to 3 \to \top, 1 \dashrightarrow 2 \dashrightarrow 3 \to \top$ である．それぞれが $\{1,3\}, \{2,3\}, \{3\}$ に対応している．

2.3 ゼロサプレス型三分決定グラフ（ZTDD）

ゼロサプレス型三分決定グラフ（Zero-suppressed ternary decision diagram, ZTDD）[12] は，符号付き集合族を表現する非巡回有向グラフ $T = (N_T, A_T)$ である．ここで，N_T は ZTDD ノードの集合，A_T は有向枝の集合である．ZTDD は ZDD と多くの定義を共有しているので，基本的には 2.2 節と同じ記法を ZTDD に対しても用いる．ZTDD が ZDD と異なる点は，ZTDD の各非終端ノードが ZERO 枝, POS 枝, NEG 枝の 3 種類の枝を 1 つずつ持つことである．T の ZERO 枝, POS 枝, NEG 枝の集合をそれぞれ $A_{T,0}, A_{T,+}, A_{T,-}$ と書く．すなわち $A_T = A_{T,0} \cup A_{T,+} \cup A_{T,-}$ である．

ZTDD T は次のように符号付き集合族を表現する．$p = (n_1, a_1, n_2, a_2, \ldots, n_k, a_k, \top) \in \mathcal{P}_T, n_1 = r_T$ に対し $S_p^\pm = \{+l(n_i) \mid a_i \in A_{T,+}, i \in [k]\} \cup \{-l(n_i) \mid a_i \in A_{T,-}, i \in [k]\}$ とする．このとき，$\mathcal{S}_T^\pm = \{S_p^\pm \mid p \in \mathcal{P}_T\}$ が T の表す符号付き集合族である．例として，符号付き集合族 $\{\{+1,-2\},\{+1,-3\},\{-2,+3\}\}$ を表す ZTDD を **図 2**

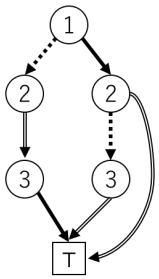

図 2 符号付き集合族 $\{\{+1,-2\},\{+1,-3\},\{-2,+3\}\}$ を表す ZTDD．点線は ZERO 枝，一重線は POS 枝，二重線は NEG 枝を表す．見やすさのため，$\bot$ とそれに向かう枝は省略している．

に示す．図 2 では根ノードから $\top$ への有向パスが 3 つあり，$1 \to 2 \Rightarrow \top$, $1 \to 2 \dashrightarrow 3 \Rightarrow \top$, $1 \dashrightarrow 2 \Rightarrow 3 \to \top$ である．それぞれが $\{+1,-2\},\{+1,-3\},\{-2,+3\}$ に対応している．

2.4 フロンティア法

フロンティア法 [2], [3], [4] は，与えられたグラフに対する制約つき部分グラフの集合を表す決定グラフを効率的に構築する手法である．ここでは ZDD を構築するフロンティア法を例にとり説明する．フロンティア法では台集合を入力グラフの辺集合 E とし，部分グラフを辺集合の部分集合で表す．すると部分グラフの集合は E を台集合とする集合族となる．フロンティア法は，求めたい部分グラフの集合（辺の集合族）を $\mathcal{M}$ として $\mathcal{M}$ を表現する ZDD を構築する．

フロンティア法ではまず，辺の順序を $e_1 < e_2 < \cdots < e_m$ と固定する．次にラベル e_1 を持つ根ノードを作成し，以下，幅優先的に ZDD を構築する．フロンティア法によって構築される ZDD の各ノードは，根ノードから自身に至るパスの集合に対応する集合族，すなわち部分グラフ集合に対応付けられる．ラベル e_i を持つ ZDD ノード n_i が表す部分グラフ集合を $\mathcal{G}(n_i)$ とすると，$\mathcal{G}(n_i)$ は $E^{<i}$ の辺のみを用いた部分グラフ集合を表す．$\mathcal{G}(\top)$ が ZDD の表す部分グラフ集合である．

フロンティア法に基づくアルゴリズムは，ConstructDD 関数と MakeNewNode 関数の組で定義される [4]．前者は ZDD 全体の幅優先的な構築手順を定める関数であり，フロンティア法に基づくアルゴリズムの多くに共通する．一方後者は，ラベル e_i を持つ ZDD ノード n_i と $x \in \{0,1\}$ が与えられたとき，n_i の x-子を返す関数であり，この関数が行うべき計算は $\mathcal{M}$ により異なる．MakeNewNode 関数は，ZDD 全体を走査することなく計算を行う．これを実現するため，各ノード n_i は $\mathcal{G}(n_i)$ に属するすべての部分グラフに共通する情報 n_i.conf を持つ．n_i.conf の定義もまた，$\mathcal{M}$ により異なる．フロンティア法に基づくアルゴリズムは，問題ごとに適切な n_i.conf と MakeNewNode 関数を設計することで実現される．MakeNewNode 関数は，n_i の構築中の子を n_{new} として，以下の条件を満たすように設計される．

(1) 任意の $S^{\le i} \in \mathcal{G}(n_{\mathrm{new}})$ と任意の $S^{>i} \subseteq E^{>i}$ に対して $S^{\le i} \cup S^{>i} \notin \mathcal{M}$ であるならば，n_{new} を作成せず，n_i の x-子を $\bot$ とする．これを**枝刈り**と呼ぶ．
(2) 枝刈りされずにすべての辺を処理し終えたならば，$\top$ を返す．
(3) その他の場合，n_i.conf から n_{new}.conf を計算し，n_{new} を返す．

多くの $\mathcal{M}$ に対し，ラベル e_i を持つノード n_i の n_i.conf として，$E^{<i}$ の辺と $E^{\ge i}$ の辺の両方に接続する頂点についての状態のみを管理すれば十分であることが知られている．このような頂点集合を**フロンティア**と呼ぶ．厳密には，$F_i = (\bigcup_{j=1}^{i-1} e_j) \cap (\bigcup_{k=i}^{m} e_k)$ を i **番目のフロンティア**と定める．ただし，$F_0 = \emptyset$ とする．n_i は F_{i-1} に属する頂点に関する情報を持つ．

3. 提案手法

本節では，グラフ分割集合 $\mathcal{A}$ を表す ZDD $Z_{\mathcal{A}}$ が与えられたとき，$\mathcal{A}$ に属するグラフ分割のうち下限制約を満たすもの，すなわち各連結成分の重みが L 以上であるグラフ分割をすべて集めた集合 $\mathcal{B}$ を表す ZDD $Z_{\mathcal{B}}$ を構築する手法を提案する．$\mathcal{A}$ に対して特に制約は設けない．すなわち，$\mathcal{A}$ は考えうるすべてのグラフ分割の集合でもよいし，連結成分の個数を限定したグラフ分割の集合でもよい．$\mathcal{A}$ から下限制約を満たすグラフ分割を取り出すには，$\mathcal{A}$ の要素のうち，重み L 未満の部分グラフを連結成分として含むものを除外すれば良い．例えば $L = 5$ のとき，**図 3** 左の部分グラフ A を考える．A は連結成分として図 3 右の連結な部分グラフ S を含む．S の重みは $1+3=4<5$ であるから，A は下限制約に反する．以後，$\mathcal{A}$ に属するグラフ分割のうち，重み L 未満の連結成分を「含む」グラフ分割を取り出す操作を考える．これが実現できれば，後で $Z_{\mathcal{A}}$ との差集合演算により，重み L 未満の連結成分を含まないグラフ分割，すなわち下限制約を満たすグラフ分割の集合を表す ZDD が得られる．

提案手法は，連結成分に関する以下の補題を用いる．

補題 3.1. 部分グラフ A と 連結な部分グラフ S に対し，A が S を連結成分として含むための必要十分条件は，以下のすべてが成り立つことである．

(1) A は S の辺をすべて含む．
(2) A は $\mathrm{dom}(S)$ に隣接する S 外の辺をどれも含まない．

補題 3.1 より，連結な部分グラフ S に対し，符号付き部分グラフ $S^{\pm}$ を次のように対応させることができる．

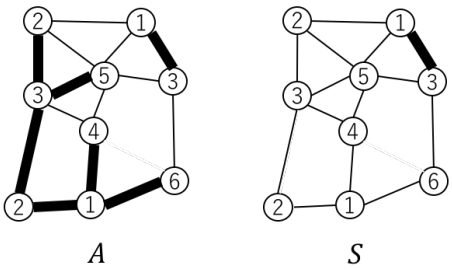

図 3 部分グラフ A と連結な部分グラフ S．太線が各部分グラフに含まれる辺を表し，各頂点に書かれた数は頂点の重みを表す．A は S を連結成分として含む．S の重みは $1+3=4<5$ であるから，$L=5$ のとき，A は下限制約に反する．

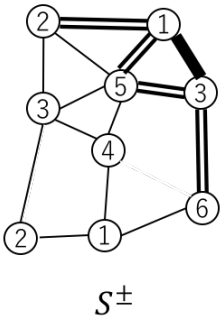

図 4 図 3 の S に対応する符号付き部分グラフ（極小カット付き部分グラフ） $S^{\pm}$．太線が正の辺，二重線が負の辺を表す．

$$S^{\pm} = S^{+} \cup S^{-} \tag{1}$$

$$S^{+} = \{+e \mid e \in S\} \tag{2}$$

$$S^{-} = \{-e \mid e \in E \setminus S \wedge e \cap \mathrm{dom}(S) \neq \emptyset\} \tag{3}$$

$S^{\pm}$ は，S について，補題 3.1 の (1) に該当する辺を正で，(2) に該当する辺を負で表現した符号付き部分グラフである．**図 4** に，図 3 の S に対する $S^{\pm}$ を示す．図 3 の A は，確かに S^{+} の辺をすべて含み，S^{-} の辺をどれも含んでいないことがわかる．ここで，S^{-} は G のカットであり，$E \setminus S^{-}$ は S^{+} を辺集合とする連結成分を持つ．また，S^{-} はそのようなカットの中で極小なものである．そこで，S に対する $S^{\pm}$ を**極小カット付き部分グラフ**と呼ぶことにする．

以下に，提案手法の手順を示す．

(1) 重みが L 未満の連結な部分グラフの集合 $\mathcal{S}$ を表す ZDD $Z_{\mathcal{S}}$ を構築する．

(2) $Z_{\mathcal{S}}$ を元に，$\mathcal{S}$ の各要素を極小カット付き部分グラフに変換した符号付き部分グラフ集合 $\mathcal{S}^{\pm}$ を表す ZTDD $T_{\mathcal{S}}$ を構築する．

(3) $T_{\mathcal{S}}$ を元に，$\mathcal{S}$ に属する少なくとも 1 つの要素を連結成分として含む部分グラフの集合を表す ZDD $Z^{\uparrow}$ を構築する．

(4) $Z_{\mathcal{A}} - Z^{\uparrow}$ により $Z_{\mathcal{B}}$ を得る．

(4) は ZDD の差集合演算 [1] により実現可能である．以下では (1)–(3) の各ステップについて説明する．

3.1 $Z_{\mathcal{S}}$ の構築

本節では，重みが L 未満の連結な部分グラフの集合 $\mathcal{S}$ を表す ZDD $Z_{\mathcal{S}}$ を構築する手法を説明する．本節では辺を 1 本以上含む部分グラフのみを考える．この仮定の元では重み L 未満の頂点 1 つのみからなる部分グラフを表現することができないが，この問題については 3.3 節で解決策を示す．

$Z_{\mathcal{S}}$ はフロンティア法に基づくアルゴリズムにより構築可能である．$Z_{\mathcal{S}}$ を構築するには，フロンティア法の過程において，ある連結成分が完成するとき，フロンティア上に存在する連結成分がただ 1 つであり，かつその連結成分の重みが L 未満であることを担保すれば良い．前者は各頂点の連結関係 `comp` を管理することで対処できる [4]．後者は，自身に接続する辺を少なくとも 1 つ採用した頂点についての重みの和 `weight` を管理することで実現できる．

3.2 $T_{\mathcal{S}}$ の構築

本節ではまず，入力グラフの連結とは限らない極小カット付き部分グラフ集合を表す ZTDD を構築する手法を提案する．次に，その手法を拡張し，3.1 節で構築された ZDD $Z_{\mathcal{S}}$ から ZTDD $T_{\mathcal{S}}$ を構築する手法を示す．

提案手法は，極小カット付き部分グラフに関する以下の補題を用いる．

補題 3.2. 符号付き部分グラフ $S^{\pm} = S^{+} \cup S^{-} (S^{+} \subseteq E^{+}, S^{-} \subseteq E^{-})$ を考える．S^{+} に属する辺を正の辺，S^{-} に属する辺を負の辺，どちらでもない辺を 0 の辺と呼ぶことにする．$S^{\pm}$ が極小カット付き部分グラフであるための必要十分条件は，以下の全てが成り立つことである．

(1) 任意の頂点 v に対し，v には 0 の辺と正の辺のうち高々 1 種類が接続する．

(2) 任意の負の辺 $\{u, v\}$ に対し，u, v の少なくとも一方には正の辺が接続する．

補題 3.2 の条件のうち，前者は S^{-} がカットであり $E \setminus \mathrm{abs}(S^{-})$ が $\mathrm{abs}(S^{+})$ を辺集合とする連結成分を持つことを保証する．後者は S^{-} の極小性を担保する．このことから補題 3.2 の正当性が言える．

補題 3.2 の条件 (1), (2) を満たす符号付き部分グラフを索引化するためのフロンティア法を設計する．まず，条件 (1) について考える．条件 (1) を担保するには，各頂点について，0，正，負のどの辺が接続しているかを長さ n の配列 $\texttt{colors} : V \to 2^{\{0,+,-\}}$ で管理すればよい．配列 `colors` を用いると，条件 (1) に違反する場合に枝刈りを行うことで条件 (1) を担保できる．

次に，条件 (2) について考える．今，負として採用された辺を $\{u, v\}$ とする．u, v が同時にフロンティアを去るときには，条件 (2) が成り立っているかを $\texttt{colors}[u], \texttt{colors}[v]$ より判定し，成り立っていない場合には枝刈りを行えば良い．一方で，u, v のうち片方のみ（一般性を失わず u とす

Algorithm 1: 極小カット付き部分グラフ集合を表す ZTDD 構築のための MakeNewNode(n_i, i, s)

// ラベル e_i を持つ ZTDD ノード n_i の $s \in \{0, +, -\}$ 枝の指す子ノードを返す関数
Let $e_i = \{u, v\}$.
Copy n_i to n'_i.
foreach $x \in \{u, v\}$ **do**
  // 補題 3.2 の条件 (1) に違反
  if $0 \in n'_i$.colors$[x]$ **and** $s = +$ **then return** $\bot$
  if $+ \in n'_i$.colors$[x]$ **and** $s = 0$ **then return** $\bot$
  if n'_i.colors$[x] = \{-\}$ **and** $s = 0$ **then**
    // x の処理済みの辺による近傍かつフロンティア上の頂点を予約
    $n'_i \leftarrow$ Reserve$(n'_i, N(E^{<i}, x) \cap (F_{i-1} \cup F_i))$
    if $n'_i = \bot$ **then return** $\bot$
  if $0 \in n'_i$.colors$[x]$ **and** $s = -$ **then**
    // 負の辺 e_i の端点 x に既に 0 の辺が接続している．よって e_i の他方の端点を予約
    $n'_i \leftarrow$ Reserve$(n'_i, e \setminus \{x\})$
    if $n'_i = \bot$ **then return** $\bot$
  if n'_i.reserved$[x] = 1$ **and** $s = 0$ **then**
    // x は予約されているが，x に 0 の辺が接続すると x に今後正の辺が接続できないため条件 (1) に違反
    return $\bot$
  if n'_i.reserved$[x] = 1$ **and** $s = +$ **then**
    n'_i.reserved$[x] \leftarrow 0$ // 予約は達成された
  n'_i.colors$[x] \leftarrow n'_i$.colors$[x] \cup \{s\}$
foreach $x \in \{u, v\}$ **do**
  if $x \notin F_i$ **then**
    // x はフロンティアを去る
    if n'_i.reserved$[x] = 1$ **and** $+ \notin n'_i$.colors$[x]$ **then**
      // x は予約されているが x に正の辺が接続していない
      return $\bot$
    if n'_i.colors$[x] = \{-\}$ **then**
      // x の処理済みの辺による近傍かつフロンティア上の頂点を予約
      $n'_i \leftarrow$ Reserve$(n'_i, N(E^{\le i}, x) \cap (F_{i-1} \cup F_i))$
      if $n'_i = \bot$ **then return** $\bot$
    // フロンティアから去る頂点の状態を忘却
    n'_i.colors$[x] \leftarrow \{\}$
    n'_i.reserved$[x] \leftarrow 0$
if $i = m$ **then**
  return $\top$ // すべての条件を満たす
return n'_i

る）がフロンティアを去るとき，u に正の辺が接続していなかった場合，今後 v に正の辺が接続することを担保しなければならない．この状況に対処するため，各頂点について，今後自身に正の辺が接続しなければならないかどうか

Algorithm 2: Reserve(n', X)

// ZTDD ノード n' において X に属する頂点を予約し，状態を更新したノード n'' を返す
Copy n' to n''.
for $x \in X$ **do**
  // x に既に 0 の辺が接続していたら予約不可
  if $0 \in n''$.colors$[x]$ **then return** $\bot$
  // x にまだ正の辺が接続していなければ予約
  if $+ \notin n''$.colors$[x]$ **then** n''.reserved$[x] \leftarrow 1$
return n''

という情報を長さ n の配列 reserved : $V \to \{0, 1\}$ で管理する．各頂点 v に対し，今後 v に正の辺が接続しなければならないとき，またそのときのみ reserved$[v] = 1$ であるように reserved を管理する．配列 reserved を用いると，頂点 v がフロンティアから去るとき，reserved$[v] = 1$ かつ $+ \notin$ colors[v] ならば枝刈りを行うことで条件 (2) を担保できる．以上のアルゴリズムにより，補題 3.2 の条件 (1), (2) をともに満たす符号付き部分グラフの集合，すなわち（連結とは限らない）極小カット付き部分グラフの集合を表す ZTDD を構築することができる．Algorithm 1 に本アルゴリズムの MakeNewNode 関数を示す．また，そのサブルーチンである Reserve 関数を Algorithm 2 に示す．

続いて，3.1 節で構築された ZDD $Z_{\mathcal{S}}$ から ZTDD $T_{\mathcal{S}}$ を構築する手法を示す．これは上記の手法にサブセッティング法 [13] の考えを取り入れることにより可能である．サブセッティング法は，ある決定グラフを参照しながら，対応する新たな決定グラフを構築するアルゴリズムの枠組みである．$Z_{\mathcal{S}}$ が表す集合族 $\mathcal{S}$ と $T_{\mathcal{S}}$ が表す符号付き集合族との間には以下の関係が成り立つ．

$$\mathcal{S}^{\pm} = \{S^+ \cup S^- \mid \exists S \in \mathcal{S}\} \tag{4}$$

ただし S^+, S^- は式 (2), (3) で定義される．式 (4) に基づき，サブセッティング法により $T_{\mathcal{S}}$ を構築可能である．すなわち，$T_{\mathcal{S}}$ の要素 $S^{\pm} = S^+ \cup S^-$ が $Z_{\mathcal{S}}$ の要素 S に対して $\mathrm{abs}(S^+) = S$ を満たすように $Z_{\mathcal{S}}$ を参照する．

3.3 $Z^{\uparrow}$ の構築

本節では，極小カット付き部分グラフ集合を表す ZTDD $T_{\mathcal{S}}$ が与えられたとき，$\mathcal{S}$ に属するいずれかの部分グラフを連結成分として含む部分グラフの集合を表す ZDD $Z^{\uparrow}$ を構築する手法を提案する．そのために，符号付き集合族に対する符号制約付き上位集合族の概念を定義する．次に，ZTDD が与えられたとき，それが表す符号付き集合族に対する符号制約付き上位集合族を表す ZDD を構築するアルゴリズムを提案する．そして，そのアルゴリズムを用いて $T_{\mathcal{S}}$ から $Z^{\uparrow}$ が構築可能であることを示す．

まず，符号付き集合と集合との間の包含関係を定義する．

定義 3.1. 台集合を同じくする集合 S と符号付き集合 T に対し，以下が成り立つとき，S は T を包含するといい，$S \supseteq T$ と書く．

$$\forall + e \in T, e \in S \wedge \forall - e \in T, e \notin S \tag{5}$$

つまり，T に正として含まれる要素を S がすべて含み，T に負として含まれる要素を S がどれも含まないとき，S は T を包含するという．これを用いて，符号付き集合族に対する符号制約付き上位集合族を次で定義する．

定義 3.2. 台集合を U とする符号付き集合族 $\mathcal{T}$ に対し，$\mathcal{T}$ の符号付き上位集合族 $f(\mathcal{T})$ を以下で定義する．

$$f(\mathcal{T}) = \{S \subseteq U \mid \exists T \in \mathcal{T}, S \supseteq T\} \tag{6}$$

すなわち，$f(\mathcal{T})$ は $\mathcal{T}$ に属する少なくとも 1 つの符号付き集合を包含するような集合の族である．$f(\mathcal{T})$ について，次の補題が成り立つ．

補題 3.3. $\mathcal{T}$ を符号付き集合族とし，$\mathcal{T}$ に属する要素の絶対値の最小値を e とする．また，

$$\mathcal{T}_0 = \{T \mid T \in \mathcal{T} \wedge +e \notin T \wedge -e \notin T\} \tag{7}$$

$$\mathcal{T}_+ = \{T \setminus \{+e\} \mid T \in \mathcal{T} \wedge +e \in T\} \tag{8}$$

$$\mathcal{T}_- = \{T \setminus \{-e\} \mid T \in \mathcal{T} \wedge -e \in T\} \tag{9}$$

とする．$g(\mathcal{T})$ を，台集合を $\{e, e+1, \ldots, n\}$ とする $\mathcal{T}$ の符号制約付き上位集合族とする．ただし，$\emptyset, \{\emptyset\}$ は任意の台集合に対する集合族とする．このとき，$f(\mathcal{T}) = g(\mathcal{T})$ である．g に関して，

$$g(\mathcal{T}) = \begin{cases} \emptyset & (\mathcal{T} = \emptyset) \\ \{\emptyset\} & (\mathcal{T} = \{\emptyset\}) \\ ((g(\mathcal{T}_0) \cup g(\mathcal{T}_+)) \bowtie \{\{e\}\}) & \\ \cup (g(\mathcal{T}_0) \cup g(\mathcal{T}_-)) & (\text{otherwise}) \end{cases} \tag{10}$$

が成り立つ．ただし，集合族 $\mathcal{A}, \mathcal{B}$ に対し，$\mathcal{A} \bowtie \mathcal{B} = \{A \cup B \mid A \in \mathcal{A} \wedge B \in \mathcal{B}\}$ とする．

Proof. $\mathcal{T} = \emptyset, \{\emptyset\}$ の場合は $g(\mathcal{T})$ の定義より明らかである．それ以外の場合を考える．$\mathcal{T} = \mathcal{T}_0 \cup (\mathcal{T}_+ \bowtie \{\{+e\}\}) \cup (\mathcal{T}_- \bowtie \{\{-e\}\})$ より，任意の $A \in g(\mathcal{T})$ は，$g(\mathcal{T}_0)$, $g(\mathcal{T}_+ \bowtie \{\{+e\}\})$, または $g(\mathcal{T}_- \bowtie \{\{-e\}\})$ に属する．$A \in g(\mathcal{T}_0)$ のとき，A は e を含んでも含まなくても良い．$A \in g(\mathcal{T}_+ \bowtie \{\{+e\}\})$ のとき，A は e を含まなければならず，かつ $A \setminus \{e\} \in g(\mathcal{T}_+)$ が成り立たなければならない．$A \in g(\mathcal{T}_- \bowtie \{\{-e\}\})$ のとき，A は e を含まず，$A \in g(\mathcal{T}_-)$ となる．以上より，

$$\begin{aligned} g(\mathcal{T}) &= g(\mathcal{T}_0) \cup (g(\mathcal{T}_0) \bowtie \{\{e\}\}) \cup (g(\mathcal{T}_+) \bowtie \{\{e\}\}) \cup g(\mathcal{T}_-) \\ &= ((g(\mathcal{T}_0) \cup g(\mathcal{T}_+)) \bowtie \{\{e\}\}) \cup (g(\mathcal{T}_0) \cup g(\mathcal{T}_-)) \end{aligned}$$

□

表 1 入力の情報．
Table 1 Property of the inputs.

グラフ名	n	m	k	$\lvert Z_{\mathcal{A}}\rvert$	$\lvert\mathcal{A}\rvert$
G_1（群馬）	37	80	4	10236	1.25×10^8
G_2（茨城）	44	95	7	17107	6.38×10^{13}
G_3（千葉）	60	134	14	301946	6.69×10^{22}
G_4（愛知）	69	173	17	1598213	9.26×10^{29}

表 2 G_1（群馬）の実験結果．
Table 2 Experimantal results for G_1 (Gunma).

r	$L(r,k)$	時間（秒）	$\lvert Z_{\mathcal{B}}\rvert$	$\lvert\mathcal{B}\rvert$
1.1	458947	6.71	17017	9.46×10^5
1.2	429016	3.01	14158	9.98×10^5
1.3	402750	1.47	12384	1.02×10^6
1.4	379514	1.53	12915	1.13×10^6
1.5	358813	1.28	13884	1.14×10^6

表 3 G_2（茨城）の実験結果．
Table 3 Experimantal results for G_2 (Ibaraki).

r	$L(r,k)$	時間（秒）	$\lvert Z_{\mathcal{B}}\rvert$	$\lvert\mathcal{B}\rvert$
1.1	383928	6.91	811805	8.40×10^{10}
1.2	355836	5.45	755527	1.12×10^{11}
1.3	331574	3.36	533722	1.52×10^{11}
1.4	310410	2.52	499241	1.98×10^{11}
1.5	291785	1.55	373024	2.42×10^{11}

補題 3.3 より，再帰演算に基づいて ZTDD から，それが表す符号付き集合族の符号制約付き上位集合族を表す ZDD を構築することができる．$Z^{\uparrow}$ が表す集合族は $T_{\mathcal{S}}$ が表す符号付き集合族に対する符号制約付き上位集合族そのものである．従って，上記のアルゴリズムを用いて $T_{\mathcal{S}}$ から $Z^{\uparrow}$ が構築可能である．

最後に，3.1 節の末尾で述べた，重み L 未満の頂点 v を孤立点として含む部分グラフに対処する方法を示す．部分グラフが頂点 v を孤立点として含むための必要十分条件は，v に接続する辺をどれも含まないことである（その他の辺は含んでも含まなくても良い）．これを用いると，頂点 v を孤立点として含む部分グラフの集合を表す ZDD を $O(m)$ 時間で構築可能である．そこで，重み L 未満の各頂点 v に対して，頂点 v を孤立点として含む部分グラフの集合を表す ZDD を構築し，$Z^{\uparrow}$ との和集合をとり，これを新たな $Z^{\uparrow}$ とする．これにより重み L 未満の頂点を孤立点として含む部分グラフにも対処できる．

4. 実験

本節では計算機実験の結果を示す．用いた計算機は Intel Xeon Processor E5-2690v2 (3.00 GHz) の CPU と 64 GB のメモリ容量を備え，OS は Oracle Linux 6 である．提案

表 4　G_3（千葉）の実験結果.
Table 4　Experimantal results for G_3 (Chiba).

r	$L(r,k)$	時間（秒）	$\lvert Z_{\mathcal{B}}\rvert$	$\lvert\mathcal{B}\rvert$
1.1	377742	347.34	22148157	1.08×10^{16}
1.2	348159	92.72	13705748	1.58×10^{16}
1.3	322874	47.44	10262756	3.14×10^{16}
1.4	301013	21.29	8074232	4.03×10^{16}
1.5	281924	15.64	7448386	5.12×10^{16}

表 5　G_4（愛知）の実験結果.
Table 5　Experimantal results for G_4 (Aichi).

r	$L(r,k)$	時間（秒）	$\lvert Z_{\mathcal{B}}\rvert$	$\lvert\mathcal{B}\rvert$
1.1	402370	314.12	62071331	1.18×10^{23}
1.2	370499	169.57	47097411	1.79×10^{23}
1.3	343307	112.85	40032786	2.45×10^{23}
1.4	319833	461.29	36686888	3.35×10^{23}
1.5	299363	4934.24	27591614	5.28×10^{23}

手法の実装には C++ 言語を用い，すべてのプログラムを g++ で `-O3` 最適化オプション付きでコンパイルした．また，提案手法の実装においては `TdZdd` ライブラリ [13]，`SAPPORO_BDD` ライブラリ[*2]を用いた．

実験で用いたグラフは，日本の各都道府県を表すものである．頂点が市区町村，辺が市区町村の隣接関係を表す．頂点の重みは，該当する市区町村の人口である．下限制約を入れる前のグラフ分割の集合 $\mathcal{A}$ を求めるに当たっては，Kawahara ら [10] の手法を用いた．Kawahara らの手法は，グラフを指定された個数の誘導部分グラフに分割する手法である．Kawahara らは連結成分の重みの最大値と最小値の比が r 以下であるような k 個の連結成分からなるグラフ分割を索引化している．r と k を決めると，ここから必要条件として各連結成分の重みが $L(k,r)=S/(r(k-1)+1)$ 以上でなければならないという条件を導くことができる [10]．実験ではこの $L(k,r)$ を連結成分の重みの下限として用いた．各グラフに対し $r=1.1,1.2,1.3,1.4,1.5$ で実験を行った．実験に用いたグラフの情報を**表 1** に示す．表にはグラフ名（括弧内は都道府県名），頂点数，辺数，連結成分の個数，$\mathcal{A}$ を表す ZDD のノード数，$\mathcal{A}$ の要素数を示している．なお，表において，集合族の要素数に対しては指数表記を用い，仮数部は小数第 3 位切り捨てで表記している．断りのない限り以後同様である．

表 2-5 に各グラフに対する実験結果を示す．各表には r の値，$L(k,r)$ の値，提案手法全体にかかった時間，$Z_{\mathcal{B}}$ のノード数，$\mathcal{B}$ の要素数を示している．どのグラフにおいても，r の値が大きくなるほど計算時間と $\lvert Z_{\mathcal{B}}\rvert$ が減少する傾向が見られる．一方，r の値が大きくなるほど $L(k,r)$ は小さくなるため，$\lvert\mathcal{B}\rvert$ は r に対して単調非減少である．このように r の増加に対して $\lvert Z_{\mathcal{B}}\rvert$ と $\lvert\mathcal{B}\rvert$ が逆の変化を示す理由は，ZDD の性質にあると考えられる．ZDD は，自身が表す集合族に「規則性」があればあるほど，等価なノードが共有されノード数が少なくなるという性質を持つ．r を小さくすればするほど，重みの下限 $L(k,r)$ が小さくなり解の規則性が失われるため，ZDD ノードの共有が効きにくくなり，結果として ZDD のノード数が増加しているのだと考えられる．また，異なるグラフ間の結果を比較すると，$\lvert Z_{\mathcal{A}}\rvert$ が大きいほど計算時間や $\lvert Z_{\mathcal{B}}\rvert$ が大きくなる傾向が見られる．

より詳細な分析のため，提案手法の各段階における決定グラフの構築時間，ノード数等を調べた．G_3 と G_4 についてその結果を**表 6-7** に示す．これらの表には，r の値，$Z_{\mathcal{S}}$ の構築時間，ノード数，要素数，$T_{\mathcal{S}}$ の構築時間，ノード数，$Z^{\uparrow}$ の構築時間，ノード数，要素数，$Z_{\mathcal{A}}-Z^{\uparrow}$ の計算時間を示している．時間の単位はどれも秒である．なお，$T_{\mathcal{S}}$ の要素数は $Z_{\mathcal{S}}$ の要素数と一致するため省略している．$Z^{\uparrow}$ の要素数は，仮数部を小数第 3 位で切り捨てると r の値ごとの違いが見られなかったため，小数第 6 位切り捨てとしている．また，$Z_{\mathcal{A}}-Z^{\uparrow}$ の結果として構築された ZDD が $Z_{\mathcal{B}}$ であり，そのノード数と要素数は表 4, 5 に示されている．

表を見ると，G_3, G_4 のどちらにおいても，$Z_{\mathcal{S}}, T_{\mathcal{S}}$ の構築は 1 秒程度で終了しており，ボトルネックは $Z^{\uparrow}$ の構築または $Z_{\mathcal{A}}-Z^{\uparrow}$ の計算であることがわかる．$Z^{\uparrow}$ の構築時間と $Z_{\mathcal{A}}-Z^{\uparrow}$ の計算時間を比べると，G_3 においてはどの r の値に対しても $Z^{\uparrow}$ の構築の方が時間がかかっている．一方で，G_4 では $r=1.1,1.2,1.3$ のときは $Z^{\uparrow}$ の構築時間の方が大きいが，$r=1.4,1.5$ では $Z_{\mathcal{A}}-Z^{\uparrow}$ の計算時間の方が大きくなっている．特に，G_4 で $r=1.5$ の場合に計算時間が全体で 1 時間以上かかっていたのは $Z_{\mathcal{A}}-Z^{\uparrow}$ の計算に時間がかかっていたことが原因であったとわかる．

決定グラフのノード数に注目すると，どのケースにおいても $\lvert T_{\mathcal{S}}\rvert$ は $\lvert Z_{\mathcal{S}}\rvert$ の高々 2 倍である．一方で，$\lvert Z^{\uparrow}\rvert$ は $\lvert T_{\mathcal{S}}\rvert$ に対し 10–650 倍となっている．この $T_{\mathcal{S}}$ から $Z^{\uparrow}$ へのノード数の増加が，$Z^{\uparrow}$ の構築に時間がかかる原因だと考えられる．この問題に対しては，$T_{\mathcal{S}}$ から $Z^{\uparrow}$ を陽に構築することなく，$Z_{\mathcal{A}}$ と $T_{\mathcal{S}}$ との間の演算を設計することで計算時間・メモリ使用量の増大を避けることができる可能性がある．

5. まとめ

本稿では，グラフ分割集合を表す ZDD に対し，禁止したい複数の連結な部分グラフを指定し，それらのうちのどれも連結成分として含まないようなグラフ分割すべての集合を表す ZDD を得る手法を提案した．また，その手法を

[*2] `SAPPORO_BDD` ライブラリは一般公開されていないが，`https://github.com/takemaru/graphillion/tree/master/src/SAPPOROBDD` からソースコードを閲覧できる．

表 6 G_3（千葉）の詳細な実験結果.
Table 6 Detailed experimantal results for G_3 (Chiba).

	Z_S			T_S		$Z^{\uparrow}$			$Z_A - Z^{\uparrow}$
r	時間	ノード数	要素数	時間	ノード数	時間	ノード数	要素数	時間
1.1	1.46	96686	8.01×10^8	1.10	150314	306.69	8216593	2.17361×10^{40}	38.09
1.2	0.78	70008	3.15×10^8	0.79	111772	78.92	3278643	2.17357×10^{40}	12.25
1.3	0.47	52896	1.22×10^8	0.61	86251	38.63	1548718	2.17319×10^{40}	7.73
1.4	0.31	41762	5.95×10^7	0.56	68833	13.07	747221	2.17316×10^{40}	7.35
1.5	0.21	34475	2.53×10^7	0.39	56800	9.62	618028	2.17314×10^{40}	5.42

表 7 G_4（愛知）の詳細な実験結果.
Table 7 Detailed experimantal results for G_4 (Aichi).

	Z_S			T_S		$Z^{\uparrow}$			$Z_A - Z^{\uparrow}$
r	時間	ノード数	要素数	時間	ノード数	時間	ノード数	要素数	時間
1.1	0.02	7042	22134	0.29	12026	216.38	7824393	1.18964×10^{52}	97.43
1.2	0.01	5274	10974	0.22	8992	116.52	4900757	1.18931×10^{52}	52.82
1.3	0.01	4042	6437	0.19	7197	67.76	3352106	1.18924×10^{52}	44.89
1.4	0.01	3333	3839	0.16	5973	42.95	2689681	1.18919×10^{52}	418.17
1.5	< 0.01	2736	2802	0.14	5019	21.87	1438723	1.18904×10^{52}	4912.23

応用することで，グラフ分割集合を表す ZDD から連結成分の重みが一定値以上であるようなグラフ分割のみを取り出し，それらすべての集合を表す ZDD を構築することが可能であることを示した．計算機実験により提案手法の性能を評価した．今後の課題として，既存手法との比較が挙げられる．また，提案手法を拡張することで連結成分の重みの上限の制約や，重みの比に関する制約が扱えるか，といった点について検討していきたい．

参考文献

[1] Minato, S.: Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems, *Proc. of the 30th ACM/IEEE design automation conference*, pp. 272–277 (1993).
[2] Knuth, D. E.: *The art of computer programming, Vol. 4A, Combinatorial algorithms, Part 1*, Addison-Wesley (2011).
[3] Sekine, K., Imai, H. and Tani, S.: Computing the Tutte Polynomial of a Graph of Moderate Size, *Proc. of the 6th International Symposium on Algorithms and Computation (ISAAC)*, pp. 224–233 (1995).
[4] Kawahara, J., Inoue, T., Iwashita, H. and Minato, S.: Frontier-based search for enumerating all constrained subgraphs with compressed representation, *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 100, No. 9, pp. 1773–1784 (2017).
[5] 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, *IEEE Transactions on Smart Grid*, Vol. 5, No. 1, pp. 102–111 (2014).
[6] Yoshinaka, R., Saitoh, T., Kawahara, J., Tsuruma, K., Iwashita, H. and Minato, S.: Finding All Solutions and Instances of Numberlink and Slitherlink by ZDDs, *Algorithms*, Vol. 5, No. 2, pp. 176–213 (2012).
[7] Imai, H., Sekine, K. and Imai, K.: Computational Investigations of All-Terminal Network Reliability via BDDs, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E82-A, pp. 714–721 (1999).
[8] Hardy, G., Lucet, C. and Limnios, N.: K-Terminal Network Reliability Measures With Binary Decision Diagrams, *IEEE Transactions on Reliability*, Vol. 56, No. 3, pp. 506–515 (2007).
[9] Maehara, T., Suzuki, H. and Ishihata, M.: Exact Computation of Influence Spread by Binary Decision Diagrams, *Proc. of the 26th International World Wide Conference (WWW)*, pp. 947–956 (2017).
[10] Kawahara, J., Horiyama, T., Hotta, K. and Minato, S.: Generating All Patterns of Graph Partitions within a Disparity Bound, *Proc. of the 11th International Conference and Workshops on Algorithms and Computation (WALCOM)*, pp. 119–131 (2017).
[11] Toda, T.: Efficient Construction of BDDs from CNFs (in Japanese), *IPSJ SIG Technical Report*, Vol. 2013-AL-145, No. 3, pp. 1–8 (2013).
[12] Yasuoka, K.: A new method to represent sets of products: ternary decision diagrams, *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 78, No. 12, pp. 1722–1728 (1995).
[13] Iwashita, H. and Minato, S.: Efficient top-down ZDD construction techniques using recursive specifications, *TCS Technical Reports*, Vol. TCS-TR-A-13-69 (2013).