Vol.2017-AL-162 No.3
2017/3/13

グラフの連結成分の大きさを考慮した連結成分分割の高速な列挙

中畑 裕[1]　川原 純[1]　笠原 正治[1,a)]

概要：本研究ではグラフの連結成分への分割，特に避難所の地区への割当ての問題に取り組む．実用上は連結成分の大きさに上限を設ける等，制約条件を加えて分割を列挙できれば有用である．しかし連結成分への分割の方法は膨大であり，その中から所望の性質を満たす分割を見つけることは一般に困難なことが多い．本研究では，自然な形状の割り当てを得るための制約として最短距離に関する凸制約を導入する．また，この制約を満たす割り当てを列挙するためのアルゴリズムを提案する．

Fast Enumeration of Graph Partition subject to Constraints of Size of Connected Components

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

1. はじめに

グラフ分割問題とは，与えられたグラフに対し，ある制約を満たす連結成分への分割を求める問題である．この問題は様々な応用を持つ．例えば，選挙区割り，コミュニティ検出，フロアプランなどが挙げられる．それらの中でも，避難所割り当て問題は重要な応用の 1 つである．避難所割り当て問題では，入力としてグラフが与えられる．グラフの頂点は地区を，辺は地区の隣接関係を表す．また，辺には距離を表す重みが与えられる．頂点のうち r 個は避難所を含む特別な頂点である．避難所割り当て問題の出力は，与えられたグラフの r 個の連結成分への分割であって，(a) 各連結成分のうちちょうど 1 つの頂点が避難所を含んでおり，かつ (b) 各連結成分における各地区から避難所までの最短距離が D 以下となるようなものである．特に，(b) の条件を**距離の上限制約**と呼ぶことにする．単に各地区を自身に最も近い避難所の連結成分に含めるだけでも制約 (a), (b) は満たされるが，これでは，避難所の人数制約，地理的な条件など様々な条件を満たさない可能性がある．よって実用上は，基本的な条件を満たす解を複数列挙し，そこから更に新たな制約を満たす解を取り出せるようにしたい．しかし連結成分分割のパターンの数は膨大であり，それらの中から所望の解を得ることは困難である．

分割パターンの膨大さに立ち向かうため，Takizawa ら [1] は ZDD（Zero-suppressed Binary Decision Diagram: ゼロサプレス型二分決定グラフ） [2] と呼ばれるデータ構造を用いて，連結成分分割の全パターンを圧縮された状態で求める手法を提案している．ZDD は，集合族を圧縮された形で表現するデータ構造である．Takizawa らの手法では，地図をグリッドで表現し，避難所までの距離や領域の凸性を考慮したグリッドの分割を求めている．Kawahara ら [3] は一票の格差を小さくする選挙区割を ZDD を用いて列挙する手法を提案している．この手法では，一般のグラフにおいて根が存在しない場合に，グラフを指定された個数の連結成分に分割するパターンを列挙する．

本研究では，一般のグラフに対し，距離の上限制約に加え，**最短距離に関する凸制約**の導入を提案する．これは避難者の動線が交差しないような連結成分分割を得るための条件である．本研究では距離の上限制約と最短距離に関する凸制約の両方を満たす連結成分分割を列挙するためのアルゴリズムを提案する．

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

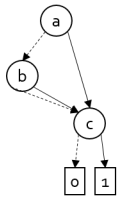

図 1　ZDD
Fig. 1　ZDD.

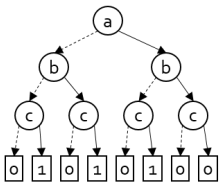

図 2　場合分け二分木
Fig. 2　Perfect binary decision tree.

本研究の構成は以下のとおりである．1 章の残りで関連研究を説明する．2 章では準備として，ZDD とフロンティア法について説明する．3 章で最短距離に関する凸制約の導入，並びに問題の定式化を行う．4 章で提案手法について述べる．5 章で実験結果を示し，6 章で本研究をまとめる．

1.1　関連研究

避難計画問題はネットワークフロー問題の観点から盛んに研究が行われている．最速避難計画問題は，ネットワーク上の全避難者が避難所のいずれかに到達するまでの時間を最小化する問題であり，Ford and Fulkerson [4], [5] によって時間拡大ネットワークを用いたアルゴリズムが提案されている．時間拡大ネットワークを用いずに高速に近似解を得る手法も提案されている [6]．最速輸送問題は，複数の需要と供給が指定されている場合の問題であり，多項式時間アルゴリズムが存在する [7]．

前述した Takizawa ら [1] や Kawahara ら [3] が用いた ZDD 構築のアルゴリズムはフロンティア法 [8], [9], [10] と呼ばれる．フロンティア法は，条件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である．Inoue ら [11] はフロンティア法によって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 [12] 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 [13], [14] や Web の影響拡散の厳密計算 [15] などの応用が存在する．

2.　準備

2.1　ZDD

ZDD（Zero-suppressed Binary Decision Diagram: ゼロサプレス型二分決定グラフ） [2] は，集合族をコンパクトに表すデータ構造である．ZDD は非巡回有向グラフであり，ノードとして 0-終端，1-終端，根ノード，その他のノードを持つ．また，枝として 0-枝と 1-枝を持つ．0-終端，1-終端以外のノードは，(a) アイテムのラベル，(b)0-枝の指すノードへのポインタ，(c)1-枝の指すノードへのポインタを持つ．あるノードから出る 0(1)-枝は，そのノードのラベルが表すアイテムを取らない（取る）ことに対応している．根ノードから 1-終端に至る経路の集合がもとの集合族に対応している．また，ZDD 上のすべてのパスにおいて，アイテムのラベルの順序は固定されており，パス上に同じラベルが複数回現れることはない．例として，集合族 $\{\{a,c\},\{b,c\},\{c\}\}$ を表す ZDD を図 **1** に示す．図には 0-枝を点線，1-枝を実線で示している．この例では，ラベルの順序は a,b,c である．ZDD の利点は集合族をコンパクトに表現できること，また，ZDD のコンパクトな形のまま様々な集合演算（union, intersect 等）を行うことが可能である点である [2]．

ZDD は図 **2** に示す場合分け二分木に対し，(a) 冗長なノードの削除と (b) 等価なノードの共有を行うことによって得られる．まず，(a) 冗長なノードの削除とは，ノード N の 1-枝の先が 0-終端なら，N を削除し，N の親からの枝を N の 0-枝の先につなぐという規則である．図 **3** に，図 2 の場合分け二分木に対して冗長なノードの削除を適用した様子を示す．まず左上の図において，右下の c のラベルを持つノード N は 1-枝の先が 0-終端なので削除され，N の親である N' からの 1-枝が，N の 0-枝が指していた先につながれる．すると右上の図でノード N' の 1-枝が新たに 0-終端を指すようになるので，N' も削除される．以上により下の図の二分木が得られる．次に，(b) 等価なノードの共有とは，同じラベルを持つ 2 つのノード N, N' の部分木が同じなら N, N' を共有するという規則である．図 **4** に示すのは，図 3 の最後の二分木に等価なノードの共有を適用した様子である．左の図において，c のラベルを持つノードはすべて，0-枝の指す先が 0-終端，1-枝の指す先が 1-終端という構造をしている．よってこれらのノードは共有される．以上の操作により図 1 と同じ ZDD が得られる．

以上の手順 (a), (b) により，ZDD は集合族をコンパクトに表現することができる．しかし，場合分け二分木を圧縮する方法では，ZDD を構築するために少なくとも場合分け二分木のノード数に比例する時間がかかる．場合分け二分木のノード数は，アイテム数を n として $\mathrm{O}(2^n)$ となるため，より効率的な ZDD の構築方法を考える必要がある．

2.2　フロンティア法

フロンティア法 [8], [9], [10] は，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である．ここで部分グラフとは，もとのグラフを $G=(V,E)$ として，$G'=(V,E'), E'\subset E$ のことである．このように定義すると，部分グラフは辺の集合とみなすことができる．ゆえに部分グラフの集合は辺の集合の集合となり，これを ZDD で表現する．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部分グラフの集合を表す．1-終端が表す部分グラフの集合が，制約を満たす部分グラフの集合である．フロンティア法では，以下のように根ノードか

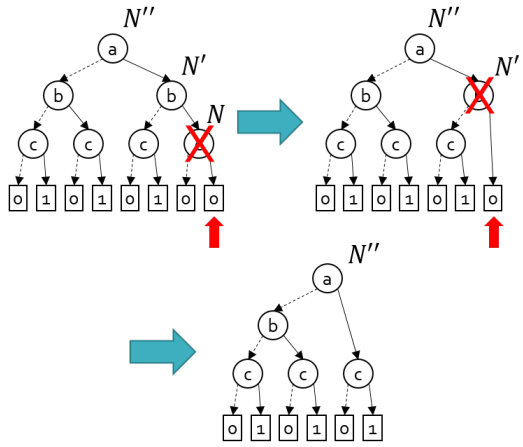

図 3　冗長なノードの削除
Fig. 3　Deletion of redundant nodes.

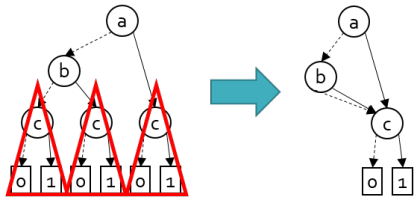

図 4　等価なノードの共有
Fig. 4　Sharing equivalent nodes.

ら幅優先的に ZDD を構築していく．まず，与えられたグラフの辺の本数を m として，各辺に $e_1, e_2, \ldots, e_m$ とラベルを付け，順序を固定する．次にラベル e_1 を持つ根ノードを作成する．そして $k = 1$ から $k = m - 1$ の順に，ラベル e_k を持つすべてのノードについて，0-枝と 1-枝が指す先の，ラベル e_{k+1} を持つノードを作成する．フロンティア法は以上の過程で (a) 枝刈りと (b) ノードの共有という 2 つの処理を行うことにより ZDD の構築を効率化する．(a) 枝刈りとは，ZDD のノードの 0-枝または 1-枝の指す子ノードを作成する際，今後制約を満たす部分グラフが得られる見込みがないとわかった時点で枝を 0-終端に接続し，それ以降の子ノードの作成は行わないという処理である．また，(b) ノードの共有とは，あるノード N の子 N' を作成するとき，同じラベルを持つ等価なノード N'' がすでに作られていれば，N' を N'' と共有するという処理である．処理 (a), (b) を行うために，フロンティア法では ZDD の各ノードに，自身が表す部分グラフの集合に対応する情報を持たせる．例として，与えられたグラフのすべての全域木の集合を表す ZDD を構築する場合を考える．ある辺を採用することによってサイクルが発生するときは，今後全域木が完成する見込みがなくなるので枝刈りを行う．この判定のために，ZDD の各ノードに $cmp[v] =$（頂点 v の連結成分番号）という配列を持たせる．実は $cmp[v]$ の定義域は頂点全体とする必要はない．フロンティア法において，未処理の辺と処理済みの辺の両方が接続する頂点の集合を

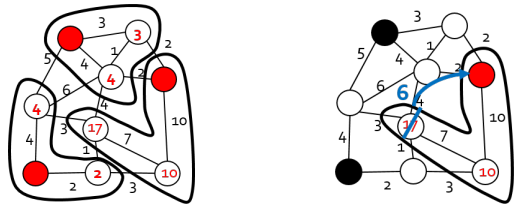

図 5　最短距離に関する凸制約を満たさない連結成分分割の例
Fig. 5　An example of a partition not satisfying the convexity of a shortest distance.

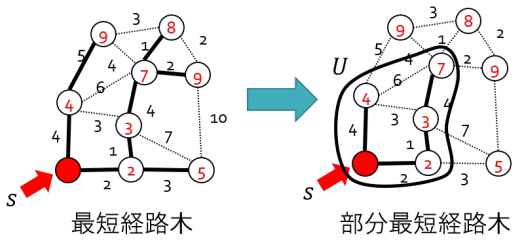

図 6　最短経路木と部分最短経路木の例
Fig. 6　An example of a shortest path tree and a partial shortest path tree.

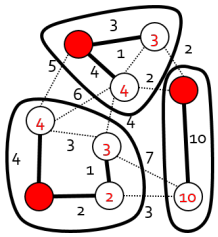

図 7　部分最短経路木分割の例
Fig. 7　An example of a partition of partial shortest path trees.

フロンティアと呼ぶ．ラベル e_k を持つノード N', N'' に対し，辺を e_k まで処理したときのフロンティアに含まれるすべての頂点 v について cmp の値が等しければ，N' と N'' の部分木は一致することが言える．よって cmp の定義域はフロンティアに限ってよい．これによりメモリが削減できるとともにノードの共有が促進され，より効率的な ZDD の構築が可能となる．

3.　問題設定

本研究では，避難所割り当て問題に対し，新たに**最短距離に関する凸制約**を導入することを提案する．例えば $D = 20$ のとき，**図 5** 左のような連結成分分割が得られたとする．色を塗られた頂点が避難所を含む頂点を表す．辺のそばに書かれた数字は辺の重みである．各頂点に書かれた値は，自身に割り当てられた避難所までの，連結成分内を通ったときの最短距離を表す．確かにこの割り当ては距離の上限制約を満たす．しかし，中央付近の 17 と書かれた頂点に注目すると，この頂点からは，自身を含む連結成分の外を通ることで避難所までの距離を短縮できてしまう

(図 5 右)．このような区割は除外したい．そこで新たな制約として，各頂点から避難所までの連結成分内における最短距離が，もとのグラフにおける最短距離に一致していなければならないという制約を課す．これが最短距離に関する凸制約である．

最短距離に関する凸制約は，**最短経路木**を用いて表現できる．$E' \subset E$ がグラフ $G = (V, E)$ の $s \in V$ を根とする最短経路木であるとは，E' が G の全域木かつ，$\forall v \in V, d_{(V,E')}(s, v) = d_G(s, v)$ となることである．ただし，$d_G(a, b)$ はグラフ G における頂点 a, b 間の最短距離とする．最短距離に関する凸制約を満たす連結成分を作るには，最短経路木の一部分を考えればよい．ここでは最短経路木の一部分を**部分最短経路木**として以下で定義する．$E' \subset E$ がグラフ G の $s \in V$ を根とする $U \subset V, s \in U$ に関する部分最短経路木であるとは，E' が $G[U]$ の全域木かつ，$\forall v \in U, d_{(U,E')}(s, v) = d_G(s, v)$ となることである．ただし，$G[U]$ は G の U による誘導部分グラフである．**図 6** に最短経路木と部分最短経路木の例を示す．色を塗られた頂点が根 s，太線で示した辺が（部分）最短経路木を構成する辺である．図のように，部分最短経路木は最短経路木の一部分を刈り取ったものであると言うことができる．

部分最短経路木を用いると，各連結成分が部分最短経路木であるような連結成分分割を求めれば，各連結成分が最短距離に関する凸制約を満たすことになる．よって，我々が求めたいのはグラフの部分最短経路木への分割である．これをグラフの**部分最短経路木分割**と呼ぶことにする．部分最短経路木分割の例を**図 7** に示す．色を塗られた頂点が各部分最短経路木の根である．また，各頂点に書かれた値は，自身が含まれる部分最短経路木の根までの最短距離である．

以上より，問題は以下のように定式化される．ただし以下では避難所を含む頂点を（部分最短経路木の）「根」と呼んでいる．

入力

- 単純連結無向グラフ $G = (V, E, w)$
  - 頂点集合 $V = \{v_1, v_2, \ldots, v_n\}$
  - 辺集合 $E = \{e_1, e_2, \ldots, e_m\}$
  - 辺の重み関数 $w : E \to \{x \in \mathbb{R} \mid x > 0\}$
- 根の集合 $S = \{s_1, s_2, \ldots, s_r\}, S \subset V$
- 距離の上限値 $D \in \mathbb{R}$

出力

出力は，以下のすべての条件を満たすグラフ G の連結成分分割の集合である．

(1) どの頂点もちょうど 1 つの連結成分に含まれる
(2) 各連結成分に根がちょうど 1 つ含まれる
(3) どの頂点からも，同じ連結成分に属する根に距離 D 以内で到達可能（距離の上限制約）
(4) 各連結成分が部分最短経路木（最短距離に関する凸制約）

提案手法では，条件を満たす連結成分分割の集合を，ZDD の形で出力する．ZDD が得られれば，そこから連結成分分割 1 つ 1 つを陽に出力することは容易である [2]．

Algorithm 1 getChild($N, k, take$)

```
Let e_k = {u, v}.
Copy N to N'.
if take = 1 then
    if cmp[u] = cmp[v] then
        return 0 // 閉路ができる
    else if cmp[u] ≤ r and cmp[v] ≤ r then
        return 0 // s_i, s_j (i ≠ j) が連結となる
    else if cmp[u] ≤ r and r < cmp[v] ≤ r + f and valid[cmp[u]][cmp[v]] = false then
        // cmp[v] は u と連結な根と連結にしてはならない
        return 0
    else if cmp[v] ≤ r and r < cmp[u] ≤ r + f and valid[cmp[v]][cmp[u]] = false then
        // cmp[u] は v と連結な根と連結にしてはならない
        return 0
    end if
    if updateState(N', k) が false を返す then
        return 0
    end if
end if
if e_k が 連結成分 c に接続する最後の辺である and c が根を含まない then
    // どの根とも連結でない連結成分が発生
    return 0
end if
if k = m then
    return 1
end if
return N'
```

Algorithm 2 updateState(N', k)

```
// 辺 e_k を採用するとき，ノード N' に情報の更新を反映
if updateValid(N', k) が false を返す then
    return false
end if
// cmp の更新（小さい方に揃える）
c_min ← min{cmp[u], cmp[v]}
c_max ← max{cmp[u], cmp[v]}
for x ∈ F_{k-1} ∪ e_k such that cmp[x] = c_max do
    cmp[x] ← c_min
end for
if checkDirection(node, c_min) が false を返す then
    return false
end if
return true
```

4. 提案手法

本章では，与えられたグラフの，距離の上限制約を満たす部分最短経路木への分割を列挙するためのフロンティア法のアルゴリズムを提案する．まず，部分最短経路木の性

Algorithm 3 updateValid(N', k)

```
// 辺 e_k を採用するとき，ノード N' における新たな連結成分の valid を更新
c_min ← min{cmp[u], cmp[v]}
c_max ← max{cmp[u], cmp[v]}
if c_min > r then
    // まだ根と連結でない成分どうしを併合する
    for i = 1 to r do
        valid[i][c_min] ← valid[i][c_min] and valid[i][c_max]
    end for
    if valid[i][c_min] = false for all i then
        // 連結成分 c_min はどの根とも連結にできなくなった
        return false
    end if
end if
return true
```

Algorithm 4 checkDirection(N', k, c)

```
// N' において辺 e_k を連結成分 c に加えることができるか
if c ≤ r then
    if d(s_c, u) + w(e_k) = d(s_c, v) then
        if indeg[c][v] = 1 then
            return false
        end if
        indeg[c][v] ← indeg[c][v] + 1
    else if d(s_c, v) + w(e_k) = d(s_c, u) then
        if indeg[c][u] = 1 then
            return false
        end if
        indeg[c][u] ← indeg[c][u] + 1
    else
        // 辺 e_k は s_c の連結成分に加えることができない
        return false
    end if
else
    for i = 1 to r do
        if d(s_i, u) + w(e_k) = d(s_i, v) then
            if indeg[i][v] = 1 then
                valid[i][c] ← false
            else
                indeg[i][v] ← indeg[i][v] + 1
            end if
        else if d(s_i, u) + w(e_k) = d(s_i, v) then
            if indeg[i][u] = 1 then
                valid[i][c] ← false
            else
                indeg[i][u] ← indeg[i][u] + 1
            end if
        else
            valid[i][c] ← false
        end if
    end for
    if valid[i][c] = false for all i then
        // 連結成分 c はどの根とも連結にできなくなった
        return false
    end if
end if
return true // ∃i valid[i][c] = true
```

質について考察する．ある辺 $e = \{u, v\}$ が s を根とする部分最短経路木に含まれうるのは次の式 (1) または式 (2) が成り立つときである．

$$d_G(s, u) + w(e) = d_G(s, v) \tag{1}$$

$$d_G(s, v) + w(e) = d_G(s, u) \tag{2}$$

$w(e) > 0$ より，式 (1), (2) が同時に成り立つことはない．そこで，式 (1) が成り立つとき辺 e を $u \to v$ の向きに，式 (2) が成り立つとき $v \to u$ の向きに向きづけることにする．すると部分最短経路木は有向木とみなすことができる．このとき s の入次数は 0，他のすべての頂点の入次数は 1 である．

以上を踏まえて，ZDD のノードに以下の情報を持たせる．まず，連結成分を管理するために，全域木の例と同様に，フロンティア上の頂点 v に対して $cmp[v] =$（v の連結成分番号）という配列を持つ．ただし，連結成分番号から各連結成分が根を含んでいるか否かがわかるようにする．ここでは，頂点 v の属する連結成分に s_i が含まれていれば $cmp[v] = i$，そうでなければ $r < cmp[v] \leq r + f$ が成り立つようにする．ここで，f はフロンティアに含まれる頂点数の最大値である．また，入次数を管理するため，フロンティア上の頂点 v に対して $indeg[i][v] =$（v の s_i を根としたときの入次数）というデータを持つ．また，どの根とも連結でないある連結成分に採用されている辺 e が，根 s_i について式 (1), (2) のどちらも満たさない場合，この連結成分と s_i の属する連結成分を併合することはできない．そこで，どの根とも連結でない連結成分については各根と連結にしてよいかどうかのフラグを持つ必要がある．これを $valid[i][c] =$（連結成分 c と s_i の属する連結成分が併合できるか）とする．c の範囲は $r < c \leq r + f$ である．距離の上限制約は，新たな頂点がフロンティアに表れたとき，各根に対する最短距離と D の関係から $valid$ を初期化することで考慮できる．

アルゴリズムの流れを説明する．ある辺 $e_k = \{u, v\}$ を採用するとき，以下の場合は枝刈りを行う．

- 閉路が発生する
- 異なる根を含む連結成分どうしを併合する
- 根 s_i を含む連結成分と根を含まない連結成分 c を併合するとき，$valid[i][c] = \text{false}$

枝刈りを通過した場合，併合される 2 つの連結成分を c, c' とする．このとき，一般性を失わず，状況は次の 2 つの場合に限られるとしてよい．

(A) c は根を含まず，c' は根 s_p を含む

(B) c も c' も根を含まない

以後，c と c' を併合することによって生まれる新たな連結成分を c'' とする．

辺 e_k を採用することによるノードの情報の更新を考える．まずは辺 e_k の向きを考慮しない場合を考え，

のちに向きを考慮した場合を考える．上記の (A) の場合，枝刈りを通過したことで既に $valid[i][c] = \text{true}$ が保証されているので，何もすることはない．(B) の場合，$valid[1][c''], valid[2][c''], \ldots, valid[r][c'']$ を更新する必要がある．$valid[i][c'']$ は，$valid[i][c] = \text{true}$ かつ $valid[i][c'] = \text{true}$ のときのみ true, そうでなければ false である．この規則に従い，すべての $1 \leq j \leq r$ に対して $valid[j][c'']$ を求める．もしすべての j に対して $valid[j][c''] = \text{false}$ となったならば，連結成分 c'' は今後どの根を含む連結成分とも併合できなくなるため枝刈りを行う．

次に，辺 e_k の向きを考慮したノードの情報の更新を考える．辺 e_k の向きは，式 (1), (2) のいずれが成り立つかによって決まる．(A) のとき，次の 3 通りがある．

(A1)式 (1) が成り立つ
(A2)式 (2) が成り立つ
(A3)式 (1) も式 (2) も成り立たない

(A1) のとき，$indeg[p][v]$ が既に 1 であれば辺 e_k を $u \to v$ の向きに採用することはできないので，枝刈りを行う．そうでないときは $indeg[p][v]$ を 1 とし，処理を続行する．(A2) のときも同様である．(A3) のときは，辺 e_k を採用することができないので枝刈りを行う．

(B) のときは，各 $1 \leq i \leq r$ に対して更新を行う．ある i について，上記の (A1)-(A3) に対応する場合を (B1)-(B3) とする．(B1) のとき，$indeg[i][v]$ が既に 1 であれば，$valid[i][c'']$ を false とする．そうでなければ $indeg[i][v]$ を 1 とする．(B2) のときも同様である．(B3) の場合，s_i を根としたとき辺 e_k は採用できないので $valid[i][c'']$ を false とする．(A) のときとの違いは，即座に枝刈りを行うのではなく $valid$ を更新している点である．辺の向きを考慮しない場合と同様に，もしすべての i に対して $valid[i][c''] = \text{false}$ となったならば枝刈りを行う．

アルゴリズムの疑似コードを Algorithm 1-4 に示す．Algorithm 1 は，ZDD のノード N において辺のラベルが e_k であるとき，$take$-枝の指すノード N' を作成し返す関数である．Algorithm 2 はそのサブルーチンであり，辺 e_k を採用するとき，ノード N' に情報の更新を反映する．さらにそのサブルーチンとして Algorithm 3, 4 がある．前者は辺 e_k を採用するとき，ノード N' における新たな連結成分の $valid$ を更新する．後者は N' において辺 e_k を連結成分 c に加えることができるかを判定する．疑似コード中では 0-終端を **0**, 1-終端を **1** で示している．

5. 実験結果

本章では，提案手法をいくつかのグラフデータに適用した実験結果を示し，提案手法の性能を評価する．用いた計算機は Intel Xeon Processor E7-8870 (2.4GHz) の CPU と 2TB のメモリ容量を備え，OS は Oracle Linux 6.7 である．提案手法の実装には C++言語と TdZdd ライブラリ [16] を用いた．すべてのケースにおいて，根は 3 個ランダムに配置した．また，どのデータも辺の重みがつけられていなかったため，1 から 10 の整数をランダムに割り当てた．

入力グラフの情報と $D = \infty$ のときの実験結果を**表 1** に示す．入力グラフの情報としては，データ名，グラフ名，頂点数 n, 辺数 m, フロンティアに含まれる頂点数の最大値 f を示している．G_1 から G_5 は 6×6 から 10×10 の格子グラフである．G_6 から G_8 は graphdrawing.org [17] で公開されている “Undirected graphs” から採用した．G_9 から G_{11} は地図を表すグラフである．G_9 は 47 都道府県，G_{10} はノースカロライナ州の主要都市，G_{11} は北海道の主要都市を表す．実験結果としては，ZDD の構築に要した時間 [sec.]，構築された ZDD のノード数，使用メモリ [MB]，得られた解の個数を示している．表中の “DNF” は 1 日以内に結果が得られなかったことを示す．

結果を見ると，提案手法は G_8 を除くすべてのグラフに対して数分以内で ZDD の構築に終了している．特に G_{11} は 頂点数 177，辺数 451 と他のグラフに比べて規模が大きいが，計算時間は 2 分 40 秒程度であった．さらに G_{11} では，$407006389846016 \simeq 4.07 \times 10^{14}$ 個の解を 1787 MB のメモリで表現している．これらの結果は，提案手法が膨大な数の分割に現実的な時間とメモリで対処できることを示している．一方 G_8 については，1 日以内に ZDD の構築が完了しなかった．この理由としては，G_8 は G_{11} よりも解の個数が多くなるために，メモリ不足に陥り計算が終了しなかったという可能性がある．また，G_2 と G_3, G_4 と G_5 のように，頂点数・辺数の大小と解の個数の大小が逆転しているケースが見られる．これらは，解の個数が各ケースごとの根の配置や辺の重みのつけ方に依存するためだと考えられる．

次に，同じグラフに対して $D = 40, 50, 60$ の 3 つの値で実験を行った結果を**表 2**, **表 3**, **表 4** に示す．用いたグラフは G_5, G_7, G_{11} であり，時間と使用メモリの単位は表 1 と同じである．どのグラフにおいても，D の値が大きくなるほど解の個数も大きくなっていることがわかる．G_5 は $D = 50$ の時点で，G_7 は $D = 60$ の時点で既に $D = \infty$ のときと解の個数が一致している．G_{11} は $D = 40$ では解が得られなかった．$D = 50$ のときと $D = 60$ のときでは解の個数が一致している．これらは $D = \infty$ のときの解の個数よりは小さい．より詳細な分析として，G_{11} に対して二分探索により解が 1 個以上見つかる最小の D を求めた．その結果，G_{11} が 1 個以上の解を持つ最小の D は $D = 47$ であり，そのときの解の個数は $D = 50, 60$ のときと一致していた．このように解の個数の増加が階段的な挙動を示すことは興味深い．

構築された ZDD を用いると，ある条件を満たす 1 つ 1 つの解を取り出すことも可能である．部分最短経路木分割

表 1　入力グラフの情報と $D = \infty$ のときの実験結果．DNF は 1 日以内に解が得られなかったことを示す．

Table 1　Property of input graphs and experimental results when $D = \infty$. DNF means we could not obtain solutions within one day.

データ名	グラフ名	n	m	f	時間 [sec.]	ZDD のノード数	使用メモリ [MB]	解の個数
grid6×6	G_1	36	60	8	0.23	8705	3	11712
grid7×7	G_2	49	84	9	6.31	233186	50	233186
grid8×8	G_3	64	112	10	0.50	31040	5	146484
grid9×9	G_4	81	144	11	25.66	914174	134	148250828
grid10×10	G_5	100	180	12	41.57	1728982	200	803968
grafo4129.78	G_6	78	108	12	246.85	22064839	4172	3003348
grafo7499.92	G_7	92	120	13	399.61	33212174	3513	10429360
grafo8882.100	G_8	100	158	20	DNF	—	—	—
47 都道府県	G_9	47	92	6	0.01	611	1	11360
ノースカロライナ州	G_{10}	100	246	11	2.39	143670	20	94454784
北海道	G_{11}	177	451	20	160.63	3417594	1787	407006389846016

表 2　$D = 40$ のときの実験結果

Table 2　An experimantal result when $D = 40$.

	時間	ZDD のノード数	メモリ	解の個数
G_5	36.58	1568269	182	630872
G_7	132.77	9060037	1243	513704
G_{11}	0.00	0	1	0

表 3　$D = 50$ のときの実験結果

Table 3　An experimantal result when $D = 50$.

	時間	ZDD のノード数	メモリ	解の個数
G_5	47.23	1728982	200	803968
G_7	271.31	19726707	2118	701008
G_{11}	157.95	3413652	1786	401721333235712

表 4　$D = 60$ のときの実験結果

Table 4　An experimantal result when $D = 60$.

	時間	ZDD のノード数	メモリ	解の個数
G_5	46.41	1728982	200	803968
G_7	437.92	33212174	3513	10429360
G_{11}	184.06	3413652	1786	401721333235712

においては，採用された辺が各避難者が最短距離で自身に割り当てられた避難所へ向かうときに通る可能性が高いため，採用された辺の重みの和ができるだけ小さい分割が望ましいと言える．そこで，採用された辺の重みの総和が最小となるような連結成分分割を取り出した例を図 **8** に示す．用いたグラフは G_6 である．色付きの頂点が根を，太線で示した辺が採用された辺を表している．これは確かに 3 つの根付き木からなる森になっている．

6. まとめ

本研究では，避難所割り当て問題に対して最短距離に関する凸制約を導入することを提案した．また，この制約を満たすグラフの連結成分分割として，部分最短経路木分割のためのフロンティア法のアルゴリズムを提案した．そして，いくつかのグラフデータに対し実験を行った．今後の課題として，提案手法の計算量的評価を行うことや，避難所の人数制約，災害による道路の閉塞などの新たな条件を考慮に入れることが挙げられる．また，他手法との比較を行うことを検討している．

参考文献

[1] Takizawa, A., Takechi, Y., Ohta, A., Katoh, N., Inoue, T., Horiyama, T., Kawahara, J. and Minato, S.: Enumeration of region partitioning for evacuation planning based on ZDD (2013).

[2] Minato, S.: Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems, *the 30th ACM/IEEE design automation conference*, pp. 272–277 (online), DOI: 10.1145/157485.164890 (1993).

[3] Kawahara, J., Horiyama, T., Hotta, K. and Minato, S.: Generating All Patterns of Graph Partitions within a Disparity Bound, *In Proc. of the 11th International Conference and Workshops on Algorithms and Computation (WALCOM 2017)* (2017 (to appear)).

[4] Ford, L. R. and Fulkerson, D. R.: Constructing maximal dynamic flows from static flows, *Operations research*, Vol. 6, No. 3, pp. 419–433 (1958).

[5] Ford, L. R. and Fulkerson, D. R.: Flows in networks. 1962, *Princeton U. Press, Princeton, NJ* (1962).

[6] 大田章雄，神山直之，瀧澤重志，加藤直樹：最速輸送問題に対する高速近似解法の提案及び避難計画への応用に関する研究，情報処理学会第 76 回全国大会， Vol. 6, p. 6 (2014).

[7] Hoppe, B. and Tardos, É.: The quickest transshipment problem, *Mathematics of Operations Research*, Vol. 25, No. 1, pp. 36–62 (2000).

[8] Sekine, K., Imai, H. and Tani, S.: Computing the Tutte Polynomial of a Graph of Moderate Size, *In Proc. of the 6th International Symposium on Algorithms and Computation (ISAAC)*, pp. 224–233 (1995).

[9] Knuth, D. E.: *The art of computer programming, Vol. 4A, Combinatorial algorithms, Part 1*, Addison-Wesley (2011).

[10] Kawahara, J., Inoue, T., Iwashita, H. and Minato, S.:

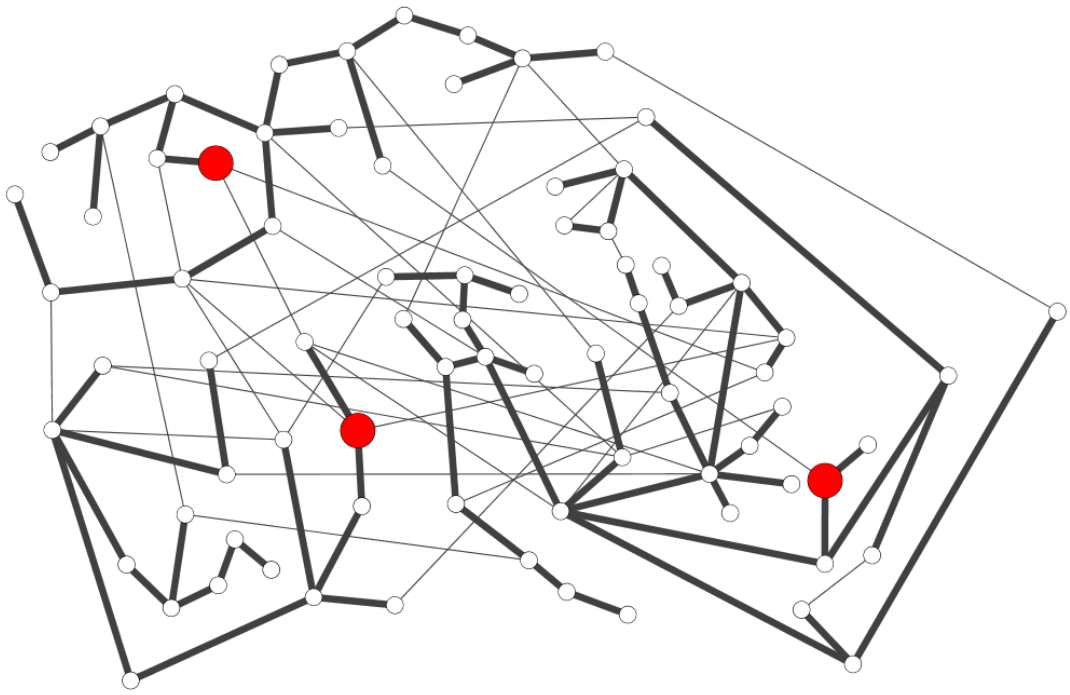

図 8　G_6 の部分最短経路木分割の例
Fig. 8　An example of a partition of partial shortest path trees of G_6.

Frontier-based search for enumerating all constrained subgraphs with compressed representation. Hokkaido University, Division of Computer Science, *TCS Technical reports TCS-TR-A-14-76* (2014).

[11] 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).

[12] 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 (online), DOI: 10.3390/a5020176 (2012).

[13] 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).

[14] 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).

[15] Maehara, T., Suzuki, H. and Ishihata, M.: Exact Computation of Influence Spread by Binary Decision Diagrams, *In Proc. of the 26th International World Wide Conference (WWW)* (2017 (to appear)).

[16] kunisura: TdZdd (online), available from 〈`https://github.com/kunisura/TdZdd`〉 (accessed 2016-02-03).

[17] graphdrawing.org: graphdrawing.org (online), available from 〈`http://www.graphdrawing.org/data.html`〉, (accessed 2017-01-25).