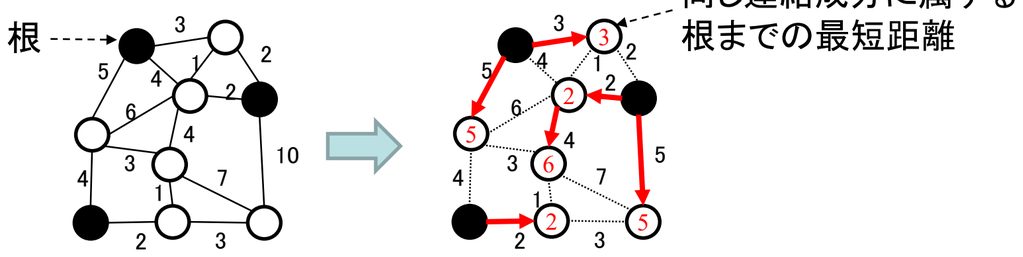
部分最短経路木分割の列挙

奈良先端科学技術大学院大学　情報科学研究科
大規模システム管理研究室　中畑裕、川原純、笠原正治

部分最短経路木分割

辺の重みはすべて正とする

入力：辺重み付き無向グラフ $G = (V, E, w)$，根の集合 $S \subset V$
出力：G の根付き全域森であって、各連結成分が最短経路木の一部であるもの



応用：避難所割り当て（各頂点から根=避難所まで、木の辺のみを通って最短距離で行ける→避難者の動線の交差を避ける）

部分最短経路木分割の集合を表すZDDを構築するための
フロンティア法のアルゴリズムを設計

既存手法：根付き無向全域森のフロンティア法

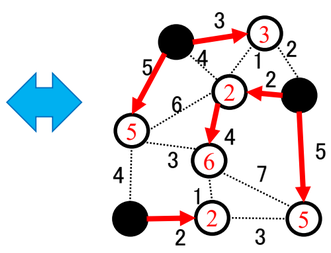
根付き無向全域森に対するフロンティア法のアルゴリズムは知られている[1]
ZDD のノードに以下を記憶させることで可能

連結成分が根を含むか否かを区別

$cmp[v]$ =（フロンティア上の頂点 v の連結成分番号）

辺を処理するとき、以下の場合は枝刈り
- 閉路が発生
- 異なる根を含む連結成分を併合
- 根を含まない連結成分が孤立するとき

（辺を採用するとき）



部分最短経路木分割において、
各連結成分は有向木
- 辺は根からの距離が増える向きに向きづけられる
- 根の入次数が0, 他の頂点の入次数が1

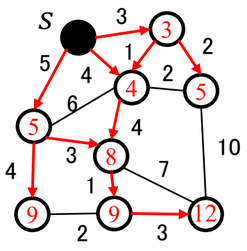
→根ごとに辺の向きを考慮する必要がある

提案手法

根が1つの場合から説明し、次に根が複数の場合を説明する

根が1つの場合 ➡ $cmp[v]$ を記憶するだけで可能

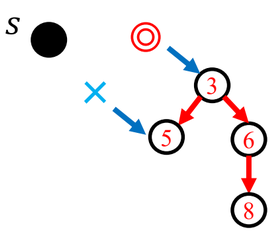
アイデア：各連結成分が常に有向木であるように更新を行う



辺 $e = \{u, v\}$ の向きは式(1), (2)に対応
- $d(s,u) + w(e) = d(s,v) \cdots (1)$　$u \to v$
- $d(s,v) + w(e) = d(s,u) \cdots (2)$　$u \leftarrow v$

どちらも満たされない場合、辺 e は最短経路木に含まれえない

$d(a,b)$: 頂点 a, b 間の最短距離



有向木において、入次数0の頂点は1つのみ

根から最も近い頂点 ➡ 根から各頂点への最短距離を事前に求めておけば判定可能

辺 $e = \{u, v\}$ を処理するとき、以下の場合は枝刈り
- （無向）閉路が発生
- e が式(1), (2)のどちらも満たさない
- 式(1)が成り立つとき、v が入次数0の頂点でない
- 式(2)が成り立つとき、u が入次数0の頂点でない
- 根を含まない連結成分が孤立するとき
- 根以外の頂点が入次数 0 のまま孤立するとき

（辺を採用するとき）

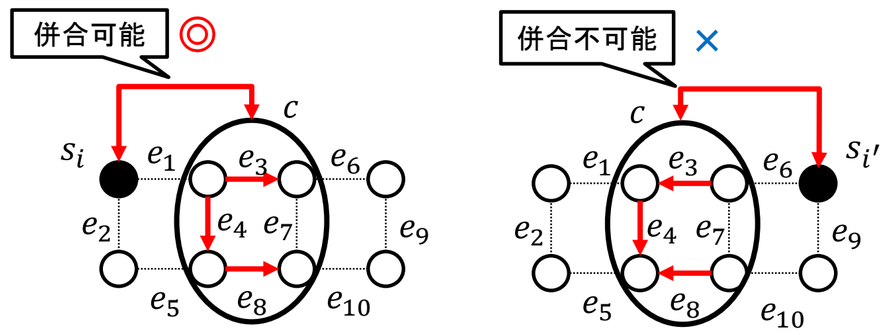
[1] J. Kawahara, T. Inoue, H. Iwashita, and S. Minato. "Frontier-based search for enumerating all constrained subgraphs with compressed representation," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E100-A, No.9, 2017 (to appear).

根が複数の場合 ➡ 各（連結成分, 根）のペアについて、併合できるか否かを管理

$S = \{s_1, s_2, \ldots, s_r\}$

$cmp[v]$ に加えて次の状態を持つ

$valid[i][c]$ =（根 s_i とフロンティア上に存在する連結成分 c が併合できるか）



$valid[i][c] = \text{true}$　　$valid[i'][c] = \text{false}$

- 辺 $e = \{u, v\}$ を処理するとき、以下の場合は枝刈り
  - （無向）閉路が発生
  - 異なる根を含む連結成分を併合
  - 根を s_i を含む連結成分と、$valid[i][c] = \text{false}$ なる連結成分 c を併合するとき

（辺を採用するとき）

- 各根 s_i について、辺の向きに関する判定を行う
  - 式(1)が成立 かつ v が入次数0でないならば、$valid[i][cmp[v]] \leftarrow \text{false}$
  - 式(2)が成立する場合も同様
  - 式(1)も式(2)も成立しないならば $valid[i][c] \leftarrow \text{false}$　（c: $cmp[u], cmp[v]$）
- 連結成分 c と c' を併合するとき、新たな連結成分 c'' の $valid$ を AND ルールで更新
  - $valid[j][c''] \leftarrow valid[j][c] \wedge valid[j][c'] \; (1 \le j \le r)$
- 根を含まない連結成分が孤立するとき、枝刈り
- 根以外の頂点 v が根 s_i から見て入次数 0 のまま孤立するとき、$valid[i][cmp[v]] \leftarrow \text{false}$
  - $valid[j][cmp[v]] = \text{false} \; (1 \le j \le r)$ となれば枝刈り

実験結果

- 用いたグラフ：
  - $G_1 \sim G_5$: 格子グラフ(6x6～10x10)
  - $G_6 \sim G_8$: グラフ描画のベンチマークグラフ[2]
  - $G_9 \sim G_{11}$: 地図グラフ
- データはすべて重みなしグラフであったため、辺の重みは1–10の整数をランダムに決定
- 根は各グラフにおいて3個ランダムに配置

Graph Name	n	m	f	Time (sec.)	# of ZDD Nodes	Memory (MB)	# of Solutions
G_1	36	60	8	0.02	3789	2	11712
G_2	49	84	9	0.24	5559	3	233186
G_3	64	112	10	0.04	3917	2	146484
G_4	81	144	11	0.51	24499	4	148250828
G_5	100	180	12	0.56	3045	4	803968
G_6	78	108	12	4.91	367612	49	3003348
G_7	92	120	13	9.01	680761	58	10429360
G_8	100	158	20	DNF	—	—	—
G_9	47	92	6	0.00	458	1	11360
G_{10}	100	246	11	0.16	11621	3	94454784
G_{11}	175	451	20	3.70	177480	45	92385124810752

最大 175 頂点 451 辺のグラフに対し、3.7秒 45 MB でZDDを構築完了
→提案手法の有効性を示す

G_8に対しては1日以内に求解できず
→より広くグラフを扱うにはアルゴリズムの効率化が必要

[2] http://www.graphdrawing.org/data.html

本研究は，JSPS科研費基盤研究(B)15H04008の助成を受けて実施している．

NAIST