フロンティア法入門講座

基盤S　RA　鈴木　浩史

もくじ

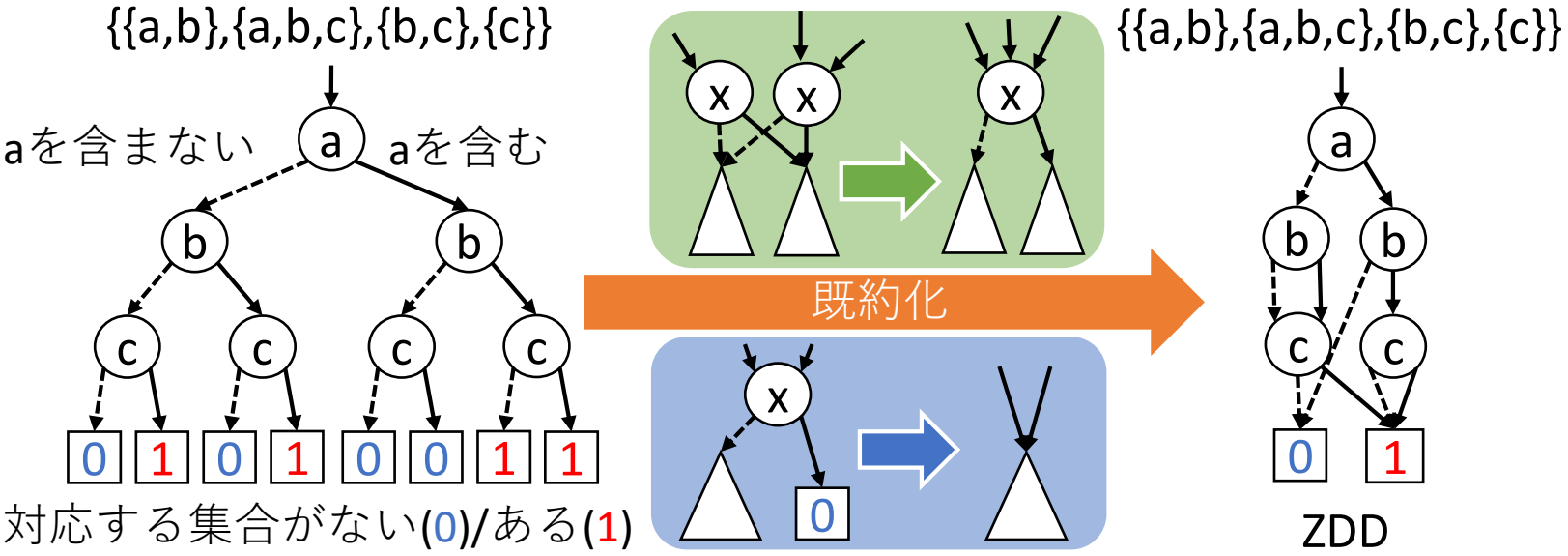
- ZDD
- グラフ
- フロンティア法
- 諸々のライブラリ
- まとめ

ZDD [minato 1993]

基盤S離散構造処理系プロジェクトの代名詞！

- 集合族を効率良く表現するデータ構造
  - 場合分け二分木の圧縮形
  - アイテムの出現順序を固定 ⇒ 一意な表現

※集合族：集合を要素とする集合

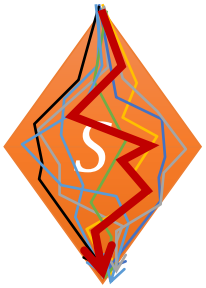


ZDDができることの一部

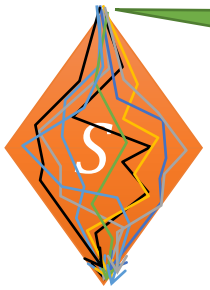
- 圧縮したまま集合族の演算ができる！

- 高速に最適解探索ができる！

$$maximize\ f(s) = \sum_{i \in s} c_i \quad (s \in S)$$



- 一様サンプリングができる！



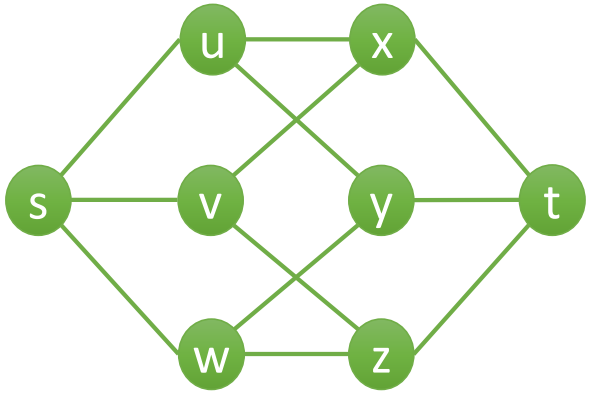
全て等確率で取り出せる

多くの操作が，アイテム数やZDDのノード数に依存した計算時間で実行可能

（無向）グラフ

- グラフ$G = (V, E)$
  - V：頂点集合，$E \subseteq [V]^2$：辺集合

※$[X]^k$：Xの部分集合でk要素からなるもの

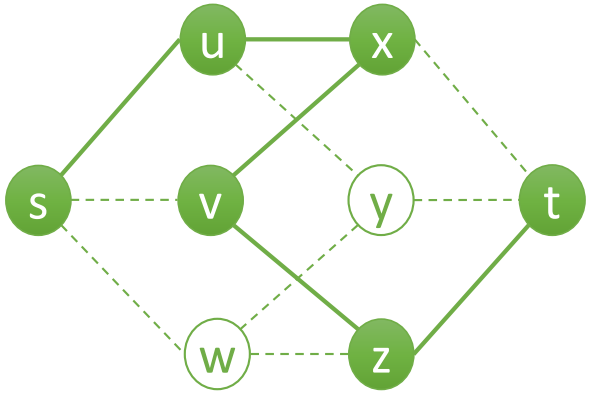


$$V = \{s, t, u, v, w, x, y, z\}$$
$$E = \{\{s, u, \}, \{s, v\}, \{s, w\},$$
$$\{t, x\}, \{t, y\}, \{t, z\},$$
$$\{u, x\}, \{u, y\}, \{v, x\},$$
$$\{v, z\}, \{w, y\}, \{w, z\}\}$$

※図と数式でフォント揃ってないのはお許しを…

部分グラフ

- グラフ$G = (V, E)$の部分グラフ$H = (V', E')$
  - $V' \subseteq V$，$E' \subseteq E$，$\bigcup_{e \in E'} e = V'$
  - いわゆる孤立点はないと仮定します

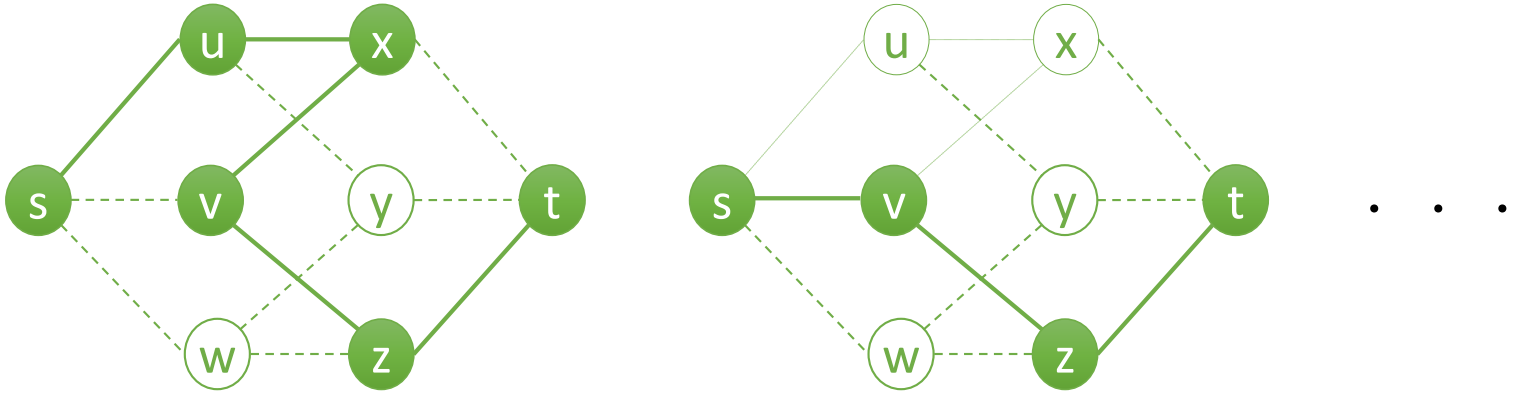


$$V' = \{s, t, u, v, x, z\}$$
$$E' = \{\{s, u,\}, \{t, z\}, \{u, x\}, \{v, x\}, \{v, z\}\}$$

グラフ列挙問題

- グラフ$G = (V, E)$と性質Pを入力として，Pを満たす部分グラフを全て見つける

$P = s$からtへの経路を成す



グラフ列挙問題を どのように応用したい？

- さらに他の性質で絞り込みたい！
  - カップルA-dとB-cが必ずマッチングする
  - 一部の道路が通行止め
- 最適な計画を立てたい！
  - 切符1枚で電車の長旅
  - 最小電力ロスの配電網
- 統計的な検査をしたい！
  - ネットワークの対故障性
  - SNSユーザーの中心性を推定

ZDDがあれば…

すべての解を保持するZDDの構築

- 既存の列挙アルゴリズムの出力をZDDに変換
  - 最も単純と思われる手法
  - 性質Pによってアルゴリズムが大幅に変化する
    - 分割法，逆探索，…
  - 解の数に依存した計算時間
  - ZDDの部分以外は省メモリ

- フロンティア法
  - グラフ列挙問題におけるZDD構築といえばこれ
  - 性質Pによらず枠組みが統一されている
  - グラフのパラメータに依存した計算時間
  - 消費メモリもグラフのパラメータ依存

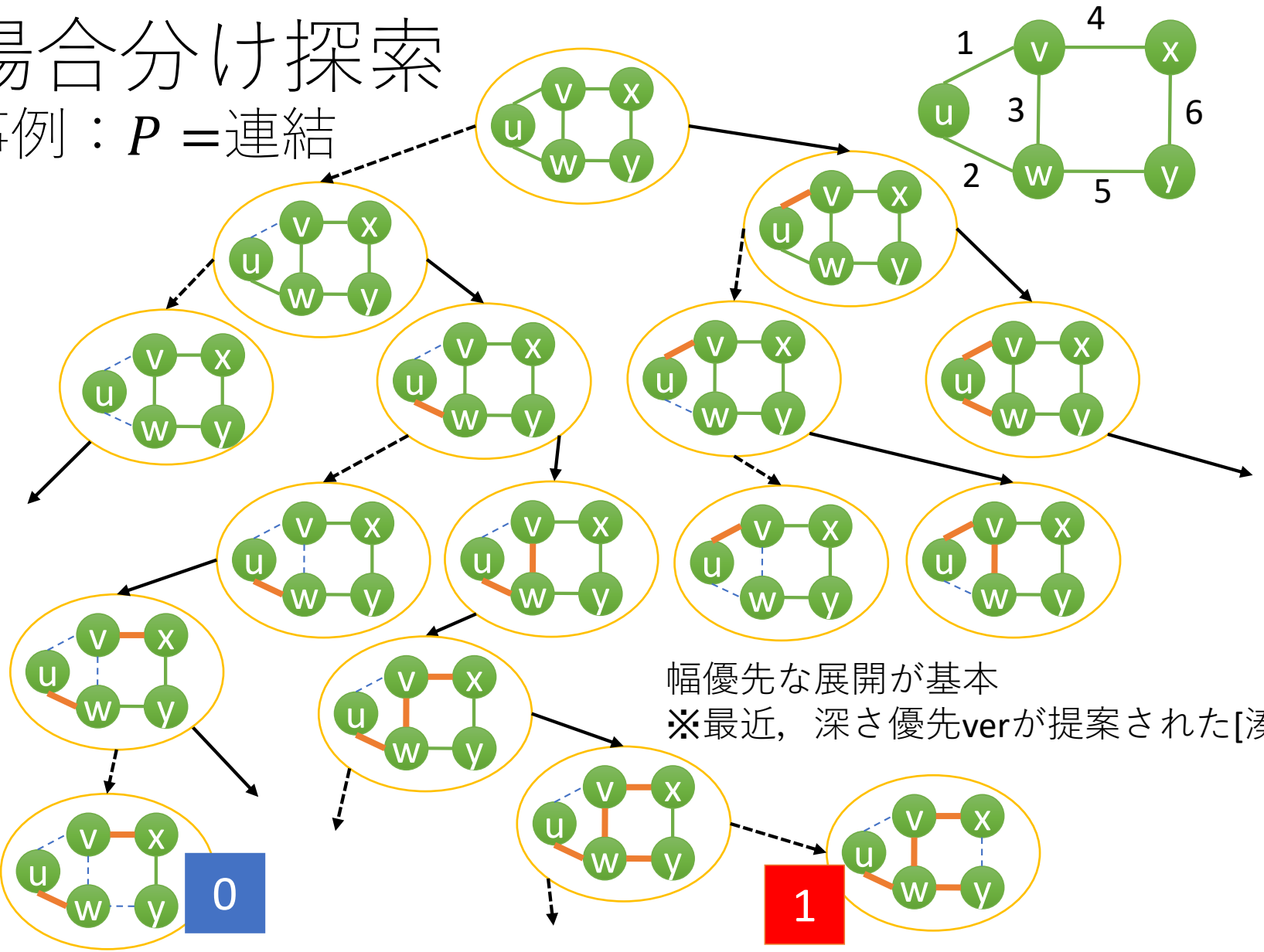
- メモリと要相談だけど、フロンティア法は桁違いの数の解を持つZDDを高速に構築できる可能性を持っている

フロンティア法によるZDDの構築

- 「辺を使う/使わない」で場合分け探索を行う
- 枝刈り，探索空間の共有による効率化
- 場合分け構造が（既約化途中の）ZDDを成す
  - 部分グラフ集合 = 辺をアイテムとする集合族
- 既約化して完成

場合分け探索

事例：P ＝連結

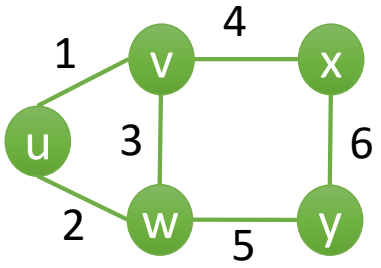


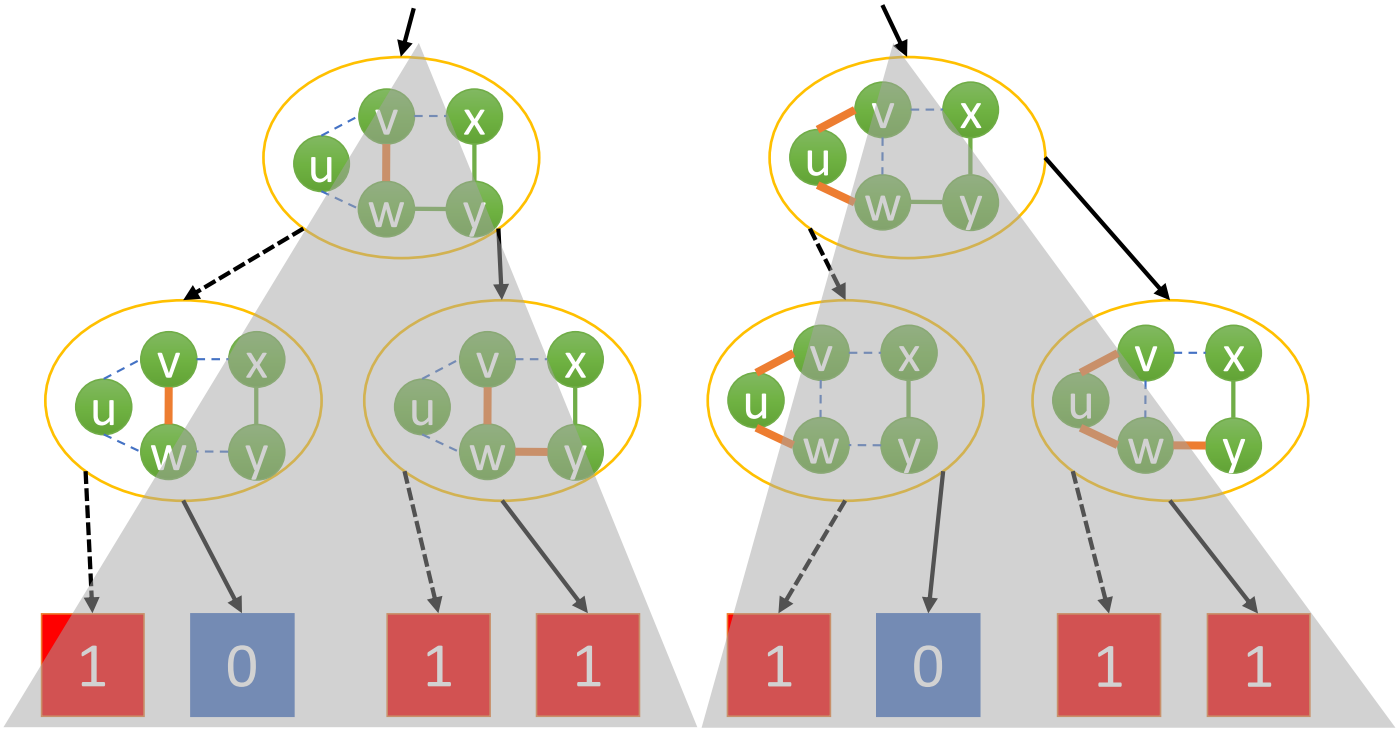
幅優先な展開が基本

※最近，深さ優先verが提案された[湊]

等価な探索空間

事例：P ＝連結

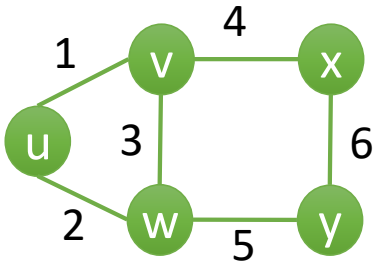


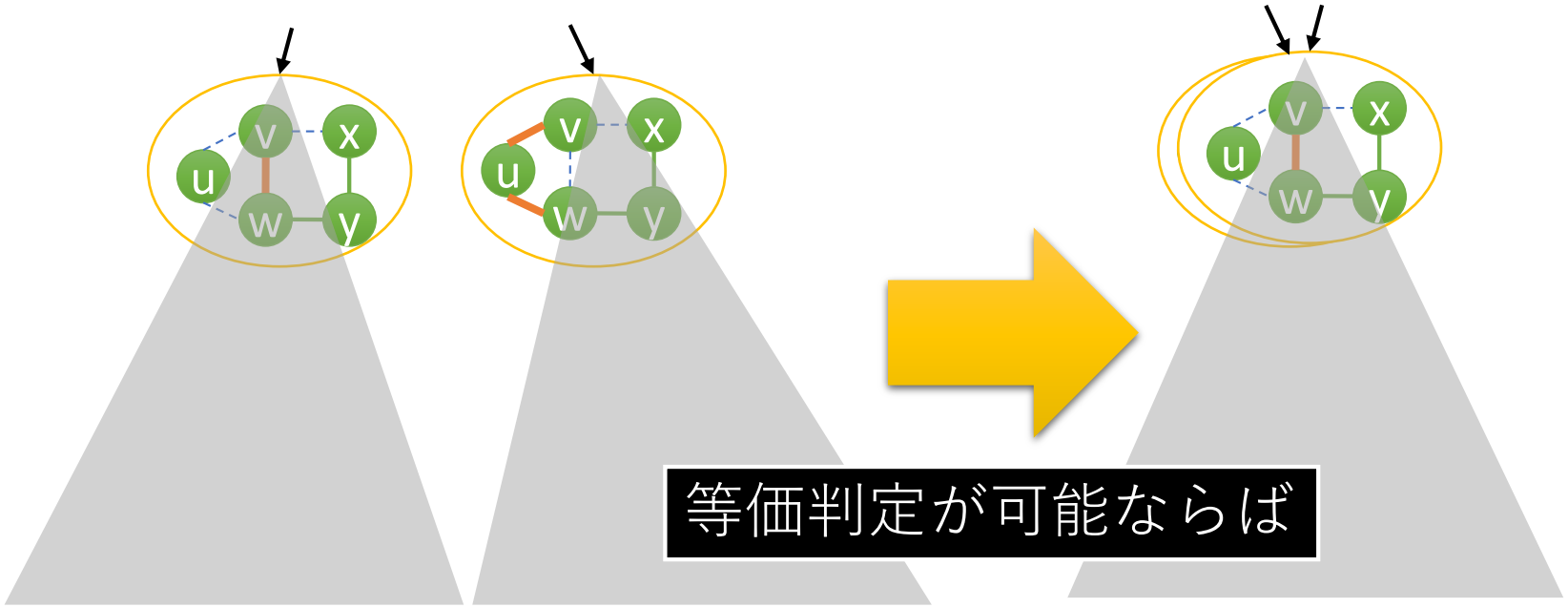


場合分け構造が同じ

等価な探索空間の共有

事例：P ＝連結





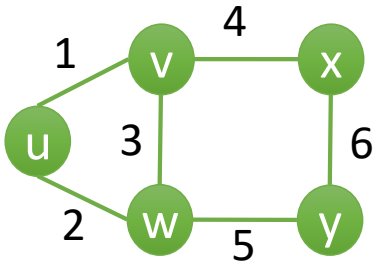
等価判定が可能ならば

場合分け構造が同じ

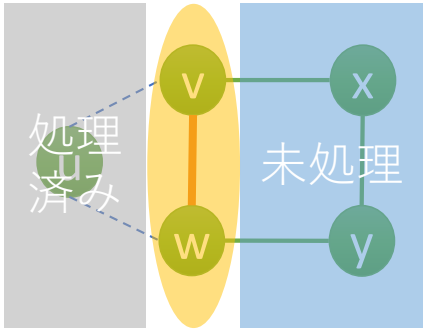
共有して良い

無駄な計算が減る！

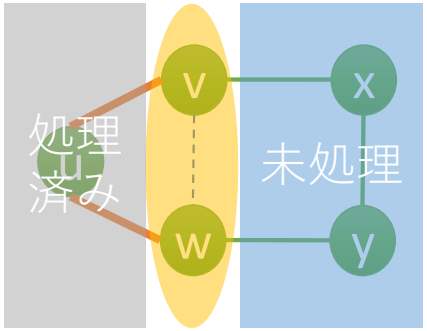
フロンティア－定義－



- 処理済み辺と未処理辺の境界にある頂点集合

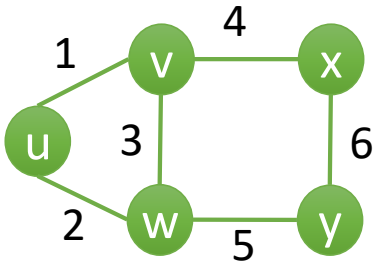


フロンティア
{v,w}

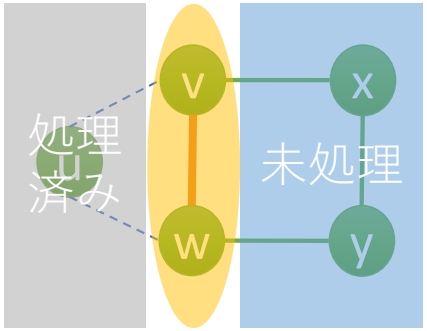


- i番目の辺e_iを処理するときのフロンティアをF_iとすると
  - $F_i = \left(\mathrm{U}_{j=1}^{i-1} e_j\right) \cap \left(\mathrm{U}_{j=i}^{|E|} e_j\right)$

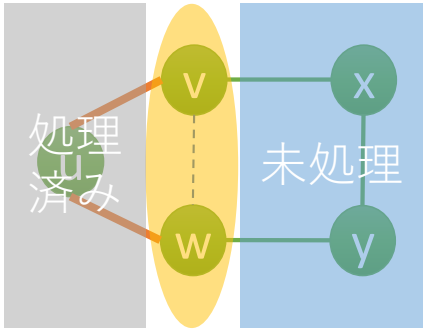
フロンティアと等価性



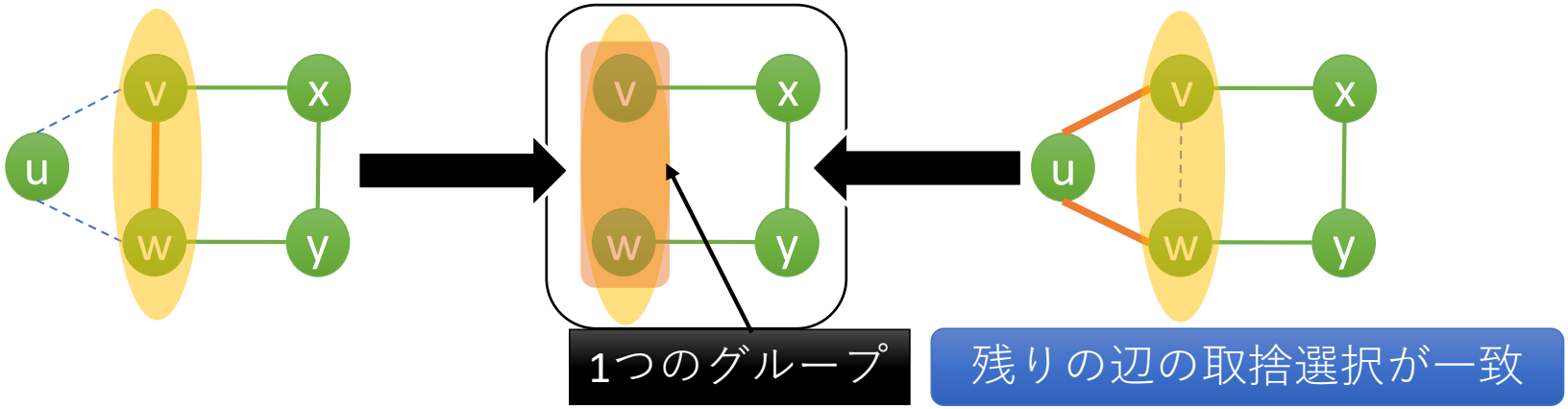
- 処理済み辺と未処理辺の境界にある頂点集合



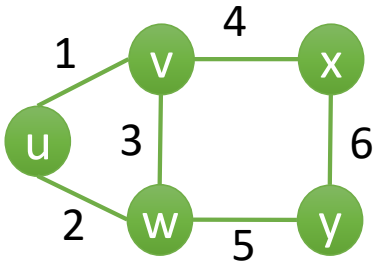
フロンティア
{v,w}



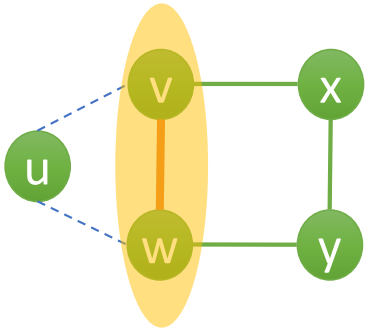
- フロンティアに着目したときの等価性



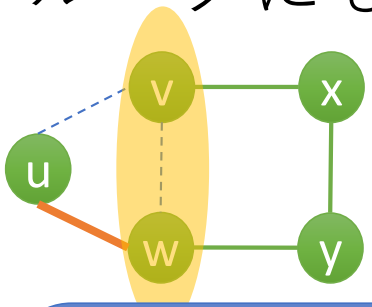
mate配列
事例：P =連結



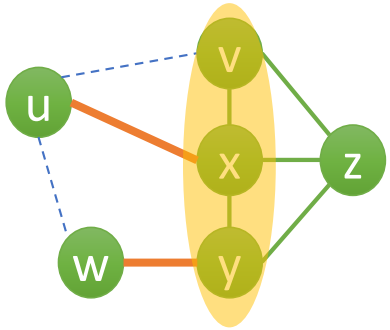
- $\forall v \in F_i$のグループ分けを表す
- $mate[v] = \begin{cases} a \geq 1, & v\text{がグループ}a\text{に属す} \\ 0, & v\text{はどのグループにも属さない} \end{cases}$



$mate[v] = 1$
$mate[w] = 1$



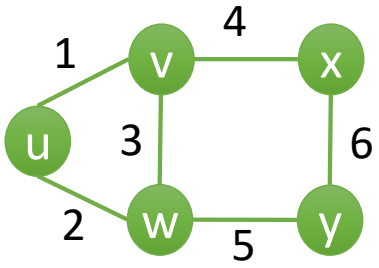
$mate[v] = 0$
$mate[w] = 1$

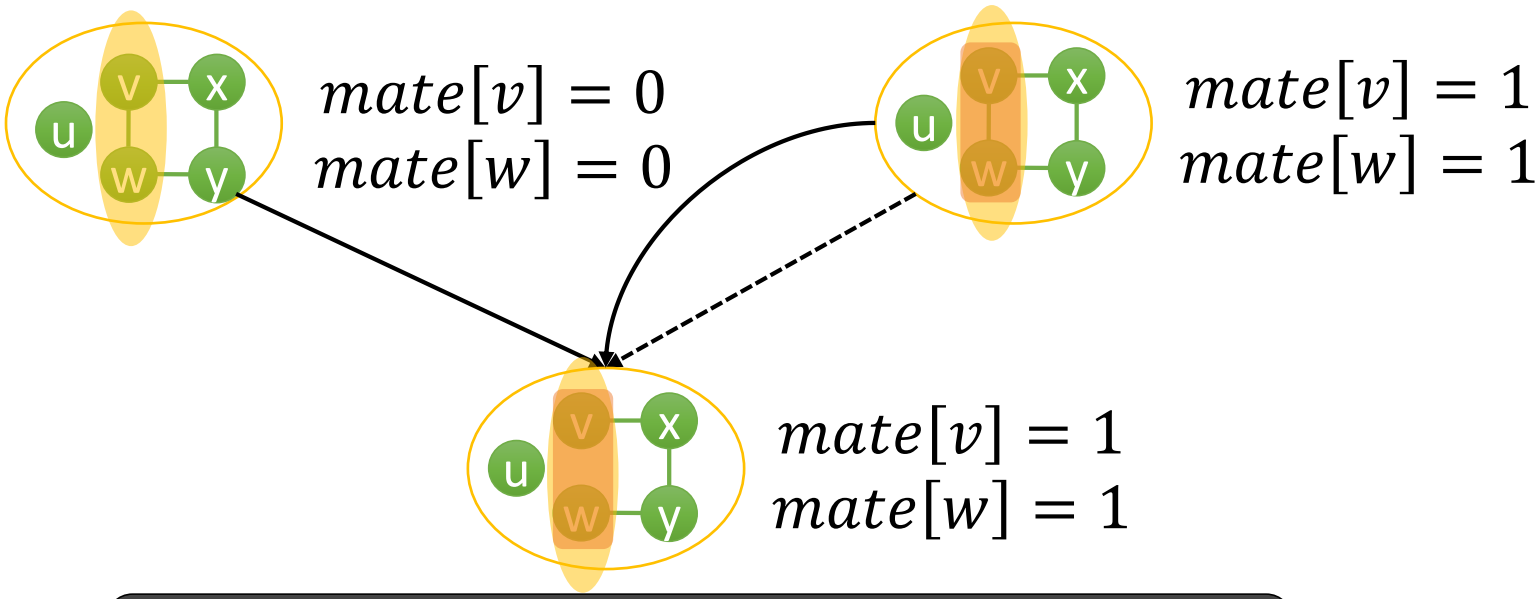


$mate[v] = 0$
$mate[x] = 1$
$mate[y] = 2$

$\forall v \notin F_i$の情報は不要
- フロンティアを出た
  ⇒グループの仲間がわかっているので,後は仲間にまかせる
- フロンティアに入ったことがない
  ⇒どのグループにも属さない

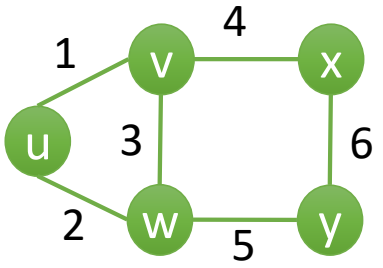
mate配列を用いた共有
事例：P =連結





mate配列の中身が一致すれば共有可能

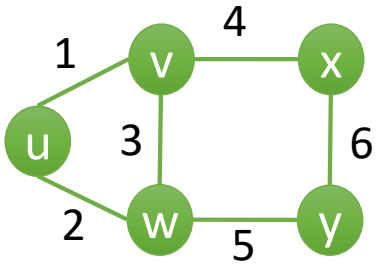
mate配列の更新
事例：P =連結



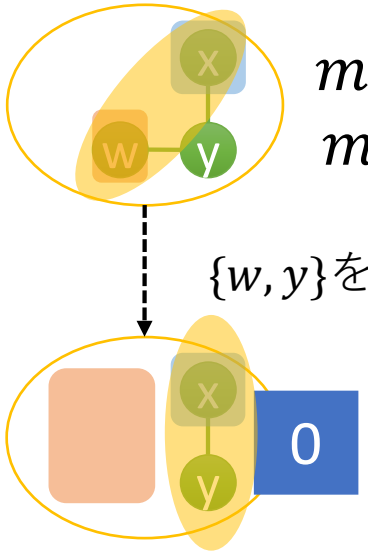
- 辺を使わないとグループは変化しないので，使う場合だけ更新が起こる
- $e_i = \{u, v\}$を使うとき
  - $\forall p \in e_i$について
    - $mate[p] = 0$ならば$mate[p] \leftarrow mate$にないグループ番号
  - $a \leftarrow mate[u], b \leftarrow mate[v]$
  - $\forall p \in F_i$について
    - $mate[p] = a$ならば$mate[p] \leftarrow b$
      （グループaをグループbにマージ）

※グループ分けが同じならば，使われているグループ番号も同じになるように，カノニカルな割り当てをし直すと共有がききやすい
「若い頂点のいるグループに若い番号を付ける」など

枝刈り
事例：P ＝連結



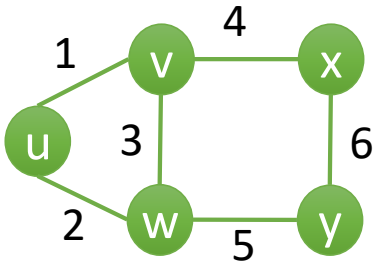
- 複数のグループが存在している中で，孤立することが確定したグループが現れたら枝刈り



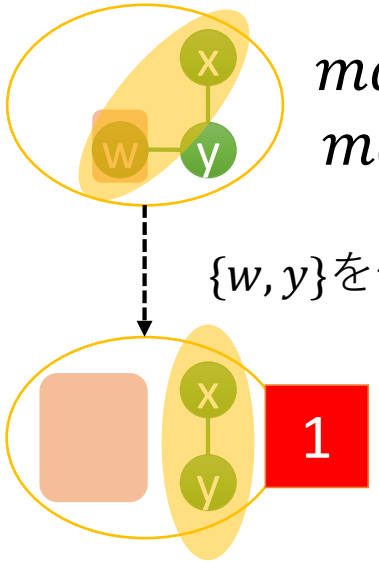
$mate[w] = 1$
$mate[x] = 2$

F_iを出た頂点がF_{i+1}に仲間を持たず，
F_iに他のグループが存在する
⇒非連結であることが確定

完成条件
事例：P =連結



- 1つのグループが確定して，他のグループが存在しない



F_iを出た頂点がF_{i+1}に仲間を持たず，
F_iに他のグループが存在しない
⇒連結な部分グラフが確定

コラム：$P =$ 連結，は便利

- 追加制約で他の構造を容易に表現
  - $\wedge$(次数1の頂点が2つ，他は次数2か0)
    - 任意のパスを表現
  - $\wedge$(すべての頂点の次数が2か0)
    - 任意のサイクルを表現
  - $\wedge$(サイクルを禁止)
    - 任意の木を表現

- 追加制約の反映についての詳細は省きます

フロンティア法のライブラリ

- Graphillion(https://github.com/takemaru/graphillion/wiki)
  - Pythonライブラリ
  - グラフを与えて制約を指定するだけ
  - とにかく記述が楽
  - 演算，最適化，サンプリングも簡単

- TdZdd(https://github.com/kunisura/TdZdd)
  - ZDDの「トップダウン構築」を提供，Graphillionの裏方
  - 探索の状態管理を自由に設計できるので上級者向け
  - グラフに限らず組合せ列挙の汎用ライブラリ
  - ZDD演算は実装されていない

まとめ

- ZDD
  - 集合族を扱うデータ構造
  - 集合族への様々な解析・処理
  - グラフ列挙問題とその応用との相性が良い

- フロンティア法
  - グラフ列挙問題のすべての解を保持するZDDを構築
  - 効率的な場合分け探索
    - フロンティア，mate配列
    - 枝刈り，探索空間の共有
  - ライブラリもある程度充実

おわり

ポスター発表では有向グラフを扱います。

ぜひ、議論をしにきてください！