アルゴリズム論
Theory of Algorithms

第11回講義
グラフアルゴリズム
～特別編～

アルゴリズム論
Theory of Algorithms

Lecture #11
Graph Algorithm
～Special Edition～

アルゴリズム的に有用なグラフクラス

- あるパラメータで測ることのできるグラフ
  - Treewidth
    - グラフがどのくらい「木」に近い構造を持っているか
  - Pathwidth
    - グラフがどのくらい「パス」に近い構造を持っているか
- {tree/path}width の特徴
  - どんなグラフも高々 *xxx*width ≦ n
  - *xxx*width≦k のグラフについていろいろな問題がDPを使うとO(2^k poly(n))時間で解ける(Ex. 最長路、最大独立点集合)
  - 与えられたグラフの *xxx*width が k 以下であるかどうかを判定する問題は
    - 一般には NP 完全
    - k が定数なら（n の）多項式時間（線形時間）で解ける

アルゴリズム的に有用なグラフクラス

- あるパラメータで測ることのできるグラフ
  - Treewidth：グラフがどのくらい「木」に近い構造を持っているか
  - Pathwidth：グラフがどのくらい「パス」に近い構造を持っているか

1. Chordal graph G の treewidth は G の最大クリークのサイズ
2. Interval graph G の pathwidth は G の最大クリークのサイズ

[演習問題] 上記の1,2はなぜか？

アルゴリズム的に有用なグラフクラス

- 関連する（小さな）グラフクラス群に関する文献
  - Ryuhei Uehara and Yushi Uno: Laminar Structure of Ptolemaic Graphs with Applications, *Discrete Applied Mathematics*, accepted, 2008.
  - Yoshihiro Takahara, Sachio Teramoto, and Ryuhei Uehara: Longest Path Problems on Ptolemaic Graphs, *IEICE Transactions*, E91-D, No. 2, pp. 170-177, 2008.
  - Ryuhei Uehara and Yushi Uno: On Computing Longest Paths in Small Graph Classes, *International Journal of Foundations of Computer Science*, 18(5), pp.911-930, 2007.

アルゴリズム的に有用なグラフクラス

- あるパラメータで測ることのできるグラフ
  - グラフがどのくらい「パス」に近い構造を持っているか
  - グラフの頂点を上手に並べたときのパラメータ
    - Bandwidth
      - グラフの頂点がどのくらい離れているか

        Ryuhei Uehara: Bandwidth of Bipartite Permutation Graphs, *19th Annual International Symposium on Algorithms and Computation (ISAAC 2008)*, Lecture Notes in Computer Science Vol. 5369, pp. 825-836, 2008/12.
    - Cutwidth
      - グラフを切断したときにどれくらいの辺が切られるか

        Pinar Heggernes, Daniel Lokshtanov, Rodica Mihai, and Charis Papadopoulos: Cutwidth of split graphs, threshold graphs, and proper interval graphs, WG 2008, LNCS, Springer. 2009.

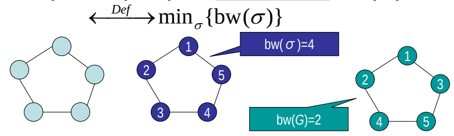
Bandwidth とは…

- グラフ *G*=(*V*,*E*) の Layout とは *V* の順序付け σ : (v_1, v_2, …, v_n) のこと。
- Layout σ の bandwidth *bw*(σ)

$$\xleftrightarrow{Def} \max_{i,j}\{|i-j| \mid \{v_i,v_j\}\in E\}$$

- Graph *G*=(*V*,*E*) の bandwidth *bw(G)*

$$\xleftrightarrow{Def} \min_{\sigma}\{\mathrm{bw}(\sigma)\}$$



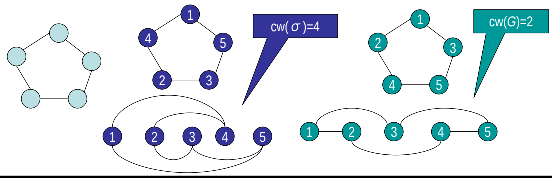
Cutwidth とは…

- Layout σ の cutwidth *cw*(σ)

$$\xleftrightarrow{Def} \max_{k}\left|\{\{v_i,v_j\}\in E \mid i<k\le j\}\right|$$

- Graph *G*=(*V*,*E*) の cutwidth *cw(G)*

$$\xleftrightarrow{Def} \min_{\sigma}\{\mathrm{cw}(\sigma)\}$$



アルゴリズム的に有用なグラフクラス

- グラフの頂点を上手に並べたときのパラメータ
  - Bandwidth
    - グラフの頂点がどのくらい離れているか
  - Cutwidth
    - グラフを切断したときにどれくらいの辺が切られるか
- こうした「頂点の最適な Layout を求める」タイプの問題はどれも$\mathcal{NP}$完全
- 限定されたグラフクラスの Cutwidth についてはそれほどわかっていない…と思う

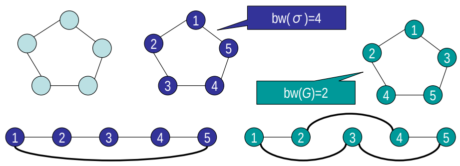
Bandwidth of Bipartite Permutation Graphs

Ryuhei UEHARA
uehara@jaist.ac.jp
http://www.jaist.ac.jp/~uehara

Bandwidth とは…

- Graph *G*=(*V*,*E*) の *bw*(*G*)

$$\xleftrightarrow{Def} \min_{\sigma}\max_{i,j}\{|i-j| \mid \{v_i,v_j\}\in E\}$$



高速な行列演算への応用などがある。

Bandwidth とは…

- Graph *G*=(*V*,*E*) の *bw*(*G*)

$$\xleftrightarrow{Def} \min_{\sigma}\max_{i,j}\{|i-j| \mid \{v_i,v_j\}\in E\}$$

- *bw*(*G*)を求める問題は…
  - $\mathcal{NP}$完全問題[Papadimitriou 1976] even for…
    - trees of max degree=3[Garey, Graham, Johnson, Knuth 1978]
    - caterpillars with hair length at most 3
    - caterpillars with at most one hair attached to every vertex in the body [M…

- 近似アルゴリズム…many papers.
- Exact アルゴリズム…O(10^n)→O(5^n) (2008)
- グラフを制限すると…

Bandwi…

bw(G)を求めるには
$O(n \times \min\{\log n, bw(G)\} \times \log bw(G))$時間

- *bw*(G)を求める問題は…
  - 以下のクラ… …のを…
    - 区間グラフ

Inf & Comp.
間違ってる!!

Open: *bw*(G)を直接求めるアルゴリズム

[Kratsch 1987][Mahesh, Rangan, Srinivasan 1991]
[Kleitman, Vohra 1990][Sprague 1994]

SIAM J. of Comput.
$O(bw(G) \times n)$ for given *bw*(G)

SIAM J. of Comput.
$O(n \log n)$ for given *bw*(G)

- chain graph [Kloks, Kratsch, Müller 1998]

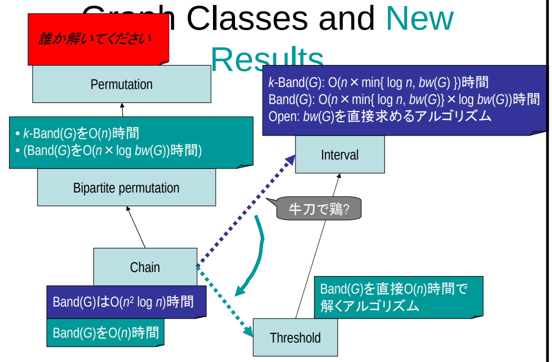
Band(G): *bw*(G)を計算する問題
k-Band(G): *bw*(G)≦*k* を判定する問題

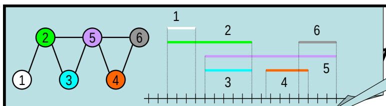
…s, Tan…

Inf. Proc. Let.
$O(n^2 \log n)$, but it uses [Spr94] as subroutine.

なんとなく牛刀をもって鶏を裂いている…?

Graph Classes and New Results





区間グラフで、区間の長さがすべて同じにできるもの

- グラフ *G*=(*V*,*E*) の proper interval completion *F* とは、*E*⊆*F* で *G*'=(*V*,*F*) が proper interval graph になるもの
- *G*=(*V*,*E*) の minimum prop. int. comp. *F* とは、*G* の p.i.c. の中で *G*' の maximum clique size が最小になるもの
- すると… *bw*(G) = (max. clique size of min. p.i.c of G) -1
[Kaplan, Shamir 1996]

… つまり max clique size が小さくなる prop. int. comp. を見つければよい。
…その prop. int. comp. の区間表現を順番に並べれば、最適な bandwidth を与えるレイアウトが得られる。

Catalogue of Algorithms…

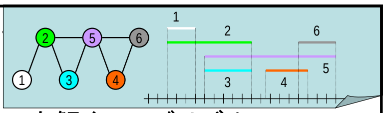
1. Algorithm for an interval graph (Outline)

2. **Algorithm for a threshold graph**
3. **Algorithm for a chain graph**

4. Algorithm for a bipartite permutation graph (Outline)

---

1. Alg. …



- *k*-Band(*G*) を解くアルゴリズム:
  1. 区間グラフの区間表現を構成する
  2. 左から右に sweep し、
     1. すでにあるレイアウトから距離 *i* に抑えなければならない頂点集合 S_i を構成する
     2. どこかの*i*で S_i が *k* に対して破綻したら失敗
     3. index 最小の S_i を選び、その中でもっとも余裕のない頂点をレイアウトの最後に加える

1. 前に余裕がない
2. 後ろに余裕がない

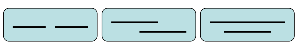
---

1. Alg. for an interval graph (Outline)

- 区間グラフのAlgからの知見:
  - まず(任意の)区間表現を1つ固定して、その順序に関してgreedyに加えればよい。
  - 固定した区間表現において、区間 I_v が I_u よりも真に左にあるなら、v < u を満たすレイアウトだけを考えればよい。

$L(I_v) \le L(I_u)$
$R(I_v) \le R(I_u)$

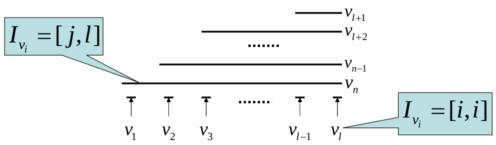
OK!! OK!! How can we...?



2. Algorithm for a threshold graph

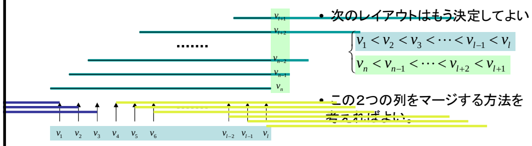
- Threshold graph とは…?
  - G=(V,E) が Threshold graph ⇔
    ∃ 重み $w(\cdot)$, 閾値t, s.t. {u,v}∈E iff $w(u)+w(v) \geqq t$
  - 頂点を軽い方から番号付けをすると、ある l に対して次の区間表現を持つ:

$w(v_{l-1}) + w(v_l) < t$
$w(v_l) + w(v_{l+1}) \geq t$



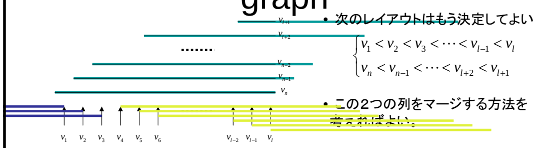
2. Algorithm for a threshold graph

- Threshold graph: 頂点を軽い方から番号付けをすると、ある l に対して次の区間表現を持つ:



- max. clique が最小になるような proper interval completion を求めればよい。
  - ✓ $v_n, v_{n-1}, \ldots, v_{l+2}, v_{l+1}$ は自然に右に伸ばせばよい。
  - ✓ $v_1, v_2, \ldots, v_{l-1}, v_l$ は…
    - ➢ ある m を見つけてやって、$v_1,\ldots,v_m$ は右端が i になるように、$v_{m+1},\ldots,v_l$ は左端がi になるようにする。

2. Algorithm for a threshold graph



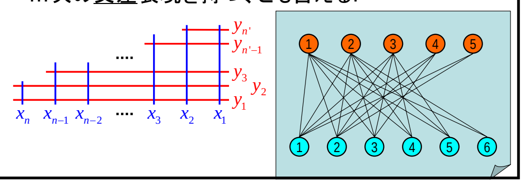
- max. clique の候補は
  - ✓ [v_m…v_n] は m の右側の clique の中で最大
  - ✓ m の左側はまじめに探す
- 線形時間アルゴリズム
  1. for m=1,2,…l do
     - 右側の候補のサイズ = m-1 のときの右側の候補のサイズ-1
     - 左側の候補のサイズ = m-1 のときの左側の候補のサイズ+1 と点 v_m において導出されるcliqueの大きい方
  2. output min $_m$ max{右側の候補 v.s. 左側の候補}

3. Algorithm for a chain graph

- Chain graph: 2部グラフG=(X,Y,E)で、X, Y が次の条件を満たすように順序付けることができる:

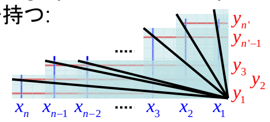
$N(x_n) \subseteq N(x_{n-1}) \subseteq \cdots \subseteq N(x_2) \subseteq N(x_1) (= Y)$
$N(y_{n'}) \subseteq N(y_{n'-1}) \subseteq \cdots \subseteq N(y_2) \subseteq N(y_1) (= X)$

- …次の交差表現を持つ、とも言える:



3. Algorithm for a chain graph

- Chain graph: 2部グラフG=(X,Y,E)で、次の交差表現を持つ:



G_iはどんな
区間表現を持つ
区間グラフなのか??

- Def: G_i ⇔ G で以下の集合を clique にしたもの

$$\{x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_m\} (m := |N(x_i)|)$$

Naïve!!

[定理] (Kloks, Kratsch, Müller 1998)
(1) 各 G_i は区間グラフである。
(2) $bw(G)$ = min $_i$ $bw(G_i)$ となる。

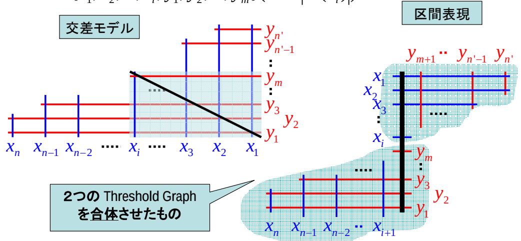
[KKM98] のO(n^2log n)アルゴリズム
For each i = 1,2,…,n
G_i を構成して $bw(G_i)$ を求める
min $bw(G_i)$ を出力.

3. Algorit

[Helly property] より、G_i の区間表現において、
clique に属するすべての頂点は、ある1点を通る。

- Def: G_i ⇔ G で以下の集合を clique にしたもの

$$\{x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_m\} (m := |N(x_i)|)$$



[Note] 左のclique, 中央の clique を特定の k 以下に抑えた prop. int. cmp. も計算できる…後で使います…

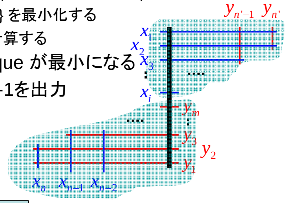
… for a chain graph

2つの Threshold Graph を合体させたもの

- 線形時間アルゴリズム
  1. For each …
     1. G_i を作る
     2. max{左側のmax.clique, 中央の max. clique, 右側の max. clique} を最小化する 左の点と右の点を計算する
  2. G_i の min. max. clique が最小になる i をみつけて$bw(G_i)$-1を出力

G_{i-1}との差分はx_i と隣接点

G_{i-1}との差分は少しだけ

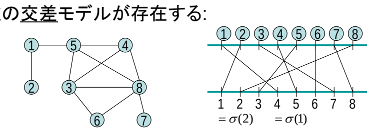


- G_1の計算は真面目にやる
- 残りは max. clique of Left, Center, Right の差分だけを上手に管理する

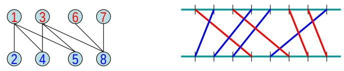
O(n) time & space

4. Alg. for a bip. perm. graph (Outline)

- Permutation graph:
  - $G=(V,E)$ with $V=\{1,2,\ldots,n\}$ が permutation graph ⇔ ある permutation σ s.t. $\{i,j\}\in E$ iff $(i-j)(\sigma(i)-\sigma(j))<0$
  - 次の交差モデルが存在する:

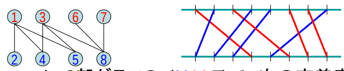


- Bipartite permutation graph: Bipartite ∩ Permutation

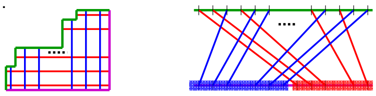


4. Alg. for a bip. perm. graph (Outline)

- Bipartite permutation graph: Bipartite ∩ Permutation



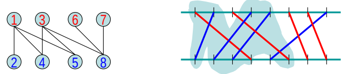
- Chain graph: 2部グラフ$G=(X,Y,E)$で、次の交差表現を持つ:



Bipartite permutation graph を左から上手に分離していくと、連続した chain graphs にできる。[Brandstädt, Lozin 2003][Uehara, Valiente 2007]

4. Alg. for a bip. perm. graph (Outline)

- Bipartite permutation graph: Bipartite ∩ Permutation



  - Bipartite permutation graph $G=(X,Y,E)$ について
    - $V_0 := \{x_0\}$; leftmost な頂点
    - $V_i := \{v \mid d(x_0,v) = i\}$
    - $G_i := G[V_i \cup V_{i+1}] = G(V_i \cup V_{i+1}, E_i)$

Thm:
$G_i=(V_i,V_{i+1},E_i)$ は chain graph.

4. Alg. for a bip. perm. graph (Outline)

- k-Band(G)に対する線形時間アルゴリズム

Input: $G=(X,Y,E)$, $\underline{k}$
Output: (~~Minimum~~) proper interval completion of G supporting $bw(G)\leqq k$ if it exists

1. $X\cup Y$ を $V_0, V_1, \ldots, V_m$ に分割する
2. For each $i = 0, 1, 2, \ldots, m$ do
   - chain graph G_i に対して “左の clique” と “中央の clique” を k に関して可能な限り pack した completion を作る
     - 途中で作れなくなったら $bw(G)>k$
     - “左のclique”は $i>0$ ならすでに少し決まっている…包含性から正当性が言える

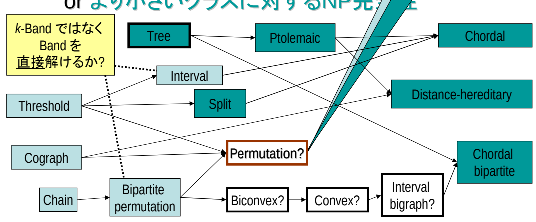
– 最後まで作れたらそれが $bw(G)\leqq k$ の証拠を与える。

これまでと同様のテクニックで線形時間アルゴリズムとして実装可能。

Bandwidth Problem

k-partite として f(k) poly(n) ?
NP-hard か?

- より広いクラスに対する多項式時間アルゴリズム or より小さいクラスに対するNP完全性



- Exact Algorithm for {these|general} graphs
  - O(10^n) (未公開) → O(5^n) (WG 2008)

Cutwidth Problem

- The cutwidth problem:
  - Bandwidth に関する以下の結果のテクニックを
    Ryuhei Uehara: Bandwidth of Bipartite Permutation Graphs, *19th Annual International Symposium on Algorithms and Computation (ISAAC 2008)*, Lecture Notes in Computer Science Vol. 5369, pp. 825-836, 2008/12.
  - 以下の論文の結果(Cutwidth)に適用することができそう…?
    Pinar Heggernes, Daniel Lokshtanov, Rodica Mihai, and Charis Papadopoulos: Cutwidth of split graphs, threshold graphs, and proper interval graphs, WG 2008, LNCS, Springer. 2009.

- Exact Algorithm for some graph classes
  – Bandwidth の $O(5^n)$ 時間アルゴリズムを応用?