3. グラフ理論

植野真臣
電気通信大学
大学院情報システム学研究科

1. 定義

今，$\mathbf{V} = \{V_1, V_2, \ldots, V_N\}$ の各要素をグラフのノード（node）（または，節点・頂点（Vertex））とする．これらのノードはエッジ（edge）（または，枝・エッジ）と呼ばれるアークにより結ばれる．もし，二つのノード V_i と V_j の間にエッジが存在すれば E_{ij} と書くことにする．グラフのエッジ集合は，$\mathbf{E} = \{E_{ij} | V_i$ と V_j の間にエッジが存在$\}$ として示される．すなわち，グラフ（Graph）は$(\mathbf{V}, \mathbf{E})$によって定義される．

例11 図2.1は七つのノード$\mathbf{V} = \{A, B, \ldots, G\}$，六つのエッジ
$\mathbf{E} = \{E_{AB}, E_{AC}, E_{BD}, E_{CE}, E_{DF}, E_{DG}\}$
によって構成されたグラフの例である．

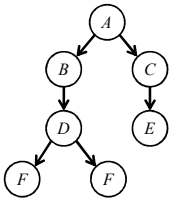


図2.1 グラフの例

2. グラフの定義

定義 20
グラフ $G = (\mathbf{V}, \mathbf{E})$ は二つの集合 **V** と **E** によって定義され，**V** は ノードの有限集合 $\mathbf{V} = \{V_1, V_2, \ldots, V_N\}$で，**E** はエッジ集合である．さらに，グラフは個々のノードにおける二つの組をエッジで結合したすべての可能性のある集合の部分集合である．

定義 21
$G = (\mathbf{V}, \mathbf{E})$をグラフとする．$E_{ij} \in \mathbf{E}$ かつ$E_{ji} \notin \mathbf{E}$のとき，エッジE_{ij} を**有向エッジ**（directed edge）と呼ぶ．V_i と V_jの有向エッジは$V_i \rightarrow V_j$と書く．

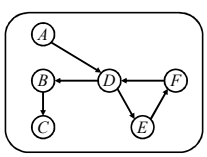
3. 定義

定義22
$G = (\mathbf{V}, \mathbf{E})$ をグラフとする．$E_{ij} \in \mathbf{E}$ かつ$E_{ji} \in \mathbf{E}$のとき，エッジ E_{ij}を**無向エッジ**（undirected edge）と呼ぶ．V_i と V_jの無向エッジは $V_i - V_j$ または$V_j - V_i$と書く．

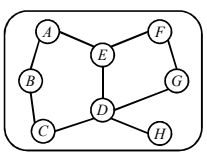
定義23
すべてのエッジが有向エッジのグラフを**有向グラフ**（directed graph）
と呼び，すべてのエッジが無向エッジのグラフを**無向グラフ**（undirected graph）
と呼ぶ．

4. 例

例12
図2.2 は有向グラフと無向グラフの例を図(a), (b) にそれぞれ示している．有向グラフ(a) では，グラフは以下で与えられ，
$\mathbf{V} = \{A, B, C, D, E, F\}$,
$\mathbf{E} = \{A \rightarrow D, B \rightarrow C, D \rightarrow B, F \rightarrow D, D \rightarrow E, E \rightarrow F\}$,
無向グラフ(b) では，グラフは以下で与えられる．
$\mathbf{V} = \{A, B, C, D, E, F, G, H\}$,
$\mathbf{E} = \{A - B, B - C, C - D, D - E, E - A, E - F, F - G, G - D, D - H\}$.



(a)



(b)

5. 定義

定義24
$G = (\mathbf{V}, \mathbf{E})$ を所与として，V_iの隣接ノード集合（adjacency nodes set）は，V_iから直接エッジが引かれたノード集合$Adj(Vi) = \{V_j \in \mathbf{V} | E_{ij} \in \mathbf{E}\}$ を示す．

定義25
V_i からV_j への**路**（path）は，$V_{i_1} = Vi$ で始まり，$V_{i_r} = V_j$ で終わるような以下を満たす順序化されたノード集合$(V_{i_1}, \ldots, V_{i_r})$ を示す．
$V_{i_{k+1}} \in Adj(V_{i_k})$. $(k = 1, \ldots, r - 1)$
このとき路の長さ（length）は，エッジの数を示し，$r - 1$である．

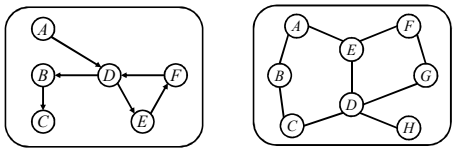
定義26
始点と終点が同じノードとなる場合（すなわち，$V_{i_1} = V_{i_r}$），（通）路（path）$(V_{i_1}, \ldots, V_{i_r})$ は**閉路**（closed path）と呼ばれる．

6. 例

例13

図2.2の有向グラフ(a)の路$D \to E \to F \to D$ は**閉路**である.

図2.2の無向グラフ(b)の路$A - B - C - D - E - A$ は**閉路**である.



(a) (b)

7. 無向グラフ

定義27

すべてのノード間にエッジが張られた無向グラフを**完全グラフ**(complete graph)と呼ぶ. N ノードの完全グラフをK_N と示す.

定義28

グラフGの部分ノード集合Sが, すべてのノード間にエッジが張られている場合, Sを**完全集合**(complete set)と呼ぶ.

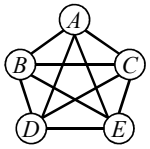


図2.3 完全グラフK_5

8. クリーク

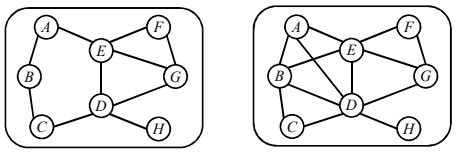
定義29

完全集合C が他のどの完全集合の部分集合にもなっていない場合, すなわち, 最大の完全集合である場合, C を**クリーク**(clique)と呼ぶ.

例15

図**2.4** は, 二つの異なるグラフのクリークを示している. グラフ(a)は, クリーク$C_1 = \{A, B\}$, $C_2 = \{B, C\}$, $C_3 = \{C, D\}$, $C_4 = \{D, H\}$, $C_5 = \{A, E\}$, $C_6 = \{D, E, G\}$, $C_7 = \{F, E, G\}$を含む.

グラフ(b)は, クリーク$C_1 = \{A, B, D, E\}$, $C_2 = \{B, C, D\}$, $C_3 = \{D, H\}$, $C_4 = \{D, E, G\}$, $C_5 = \{E, F, G\}$を含む.



(a) (b)

9. 近傍

定義30

無向グラフにおける閉路(closed path)を**ループ**(loop)と呼ぶ.

定義31

無向グラフにおいてノードV_i に隣接するノード集合を**近傍**(neighbours)と呼ぶ.

$$Nbr(V_i) = \{V_j \mid V_j \in Adj(V_i)\}$$

例16

図**2.5** におけるノードE の近傍は$\{A, D, F\}$ である.

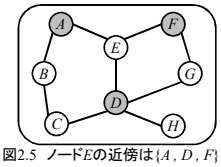


図2.5 ノードEの近傍は$\{A, D, F\}$

10. 境界

定義32

無向グラフにおいてある部分集合S と隣接しているノード集合をSの境界(boundary)と呼び, $Bnd(S)$ と書く.

$$Bnd(S) = \left(\bigcup_{V_i \in S} Nbr(V_i)\right) \setminus S$$

ここで$X \setminus S$はX のS を除いたすべてのノード集合を示す.

例17

図**2.6** における$\{D,E\}$ の境界は$\{A, C, F, G, H\}$ である.

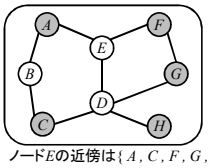


図2.6 ノードEの近傍は$\{A, C, F, G, H\}$

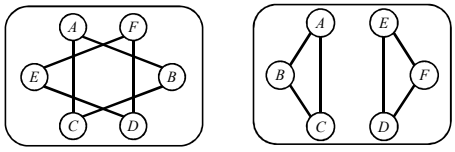
11. 無向グラフのタイプ

定義33

無向グラフのすべての二つのノード間で少なくとも一つの路が存在するとき, **連結グラフ**(connected graph)と呼ぶ. それ以外を**非連結グラフ**(disconnected graph)と呼ぶ.

例18

図**2.7** は, 同じ構造をもつ非連結グラフの異なる二つの表現である. 図(a)はエッジが交差しており, 非連結には見えないが, 図(b)のように交差を外し, 分離すればより非連結性が強調される.



(a) (b)

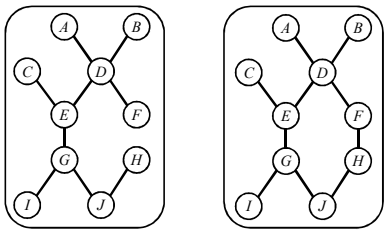
12. 木と複連結グラフ

定義34
無向グラフのすべての二つのノード間にたった一つの路しか存在しない連結グラフを，**木**(tree)と呼ぶ.

定義35
無向グラフで二つ以上の路がある頂点ペアが少なくとも一つ存在する，もしくは少なくとも一つのループを含む連結グラフを，**複連結グラフ**(multiplyconnected graph)と呼ぶ.

12. 木と複連結グラフ

例19
図**2.8** (a) は木であり，図(b) はノードDとJをつなぐ路が二つあるので複連結グラフを示している.



13. 無向グラフのまとめ

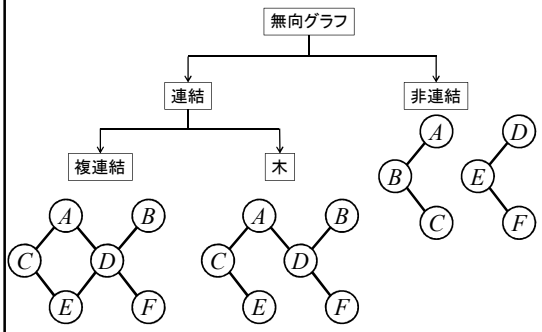


図2.9 無向グラフの種類

14. 有向グラフ

定義36
V_j からV_i への直接のエッジ$V_j \rightarrow V_i$ が存在するとき，V_i をV_j の**親ノード**(parent)と呼び，V_j をV_i の**子ノード**(child)と呼ぶ. ノードV_i の親ノード集合を$\mathbf{\Pi}_i$と書く.
図**2.10** は，ノードE の親ノードと子ノードの例である.

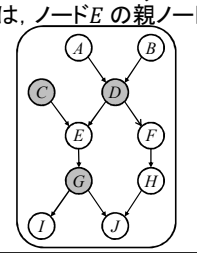


図**2.10** ノードE の親ノードと子ノード

14. 有向グラフ

定義37
あるノードとその親ノードからなる集合をそのノードのファミリー(family)と呼ぶ. 図**2.11** に，ノードD, E, J のファミリーの例を示した.
定義38
V_j からV_i への路が存在すれば，V_j をV_i の先祖(ancestor)と呼ぶ.
定義39
ノード集合**S** がノードV_i の先祖をすべて含んでいる場合，**S** をアンセスタル集合(ancestal set)と呼ぶ.
定義40
ノードV_i から路が存在するV_j を含まないすべてのノード集合をV_iの子孫(descendant)と呼ぶ. 図**2.12** に，ノードE の先祖と子孫の例を示した.

15. 例

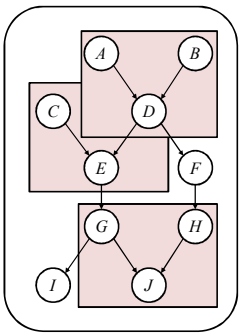


図2.11 ノード D, E, Jのファミリー

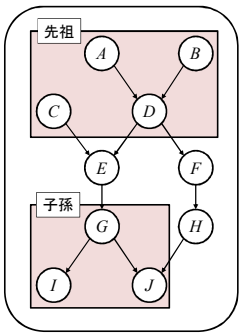


図2.12 ノードE の先祖と子孫

16. ナンバリング

定義41
$\mathbf{V} = \{V_1, \cdots, V_N\}$ を所与として，**ナンバリング**（numbering）α とは $\{1, \cdots, N\}$ の各数字をノード集合の要素に全単射することである．すなわち，

$$\alpha : \{1, \cdots, N\} \rightarrow \{V_1, \cdots, V_N\}.$$

$\alpha(i)$ はノードV_i に対応し，ナンバリングとはノードの順序化$(\alpha(1), \cdots, \alpha(N))$であると解釈できる．

定義42
有向グラフでノードの番号がその子ノードよりも小さい番号が振り分けられるとき（先祖から順に番号を振り分けていくナンバリングのとき），**アンセストラルナンバリング**（ancestral numbering）と呼ぶ．

例20
同一の有向グラフでも異なるアンセストラルナンバリングが可能である，図**2.13** は同一の有向グラフにおける異なる二つのアンセストラルナンバリングの例を示している．

16. ナンバリング

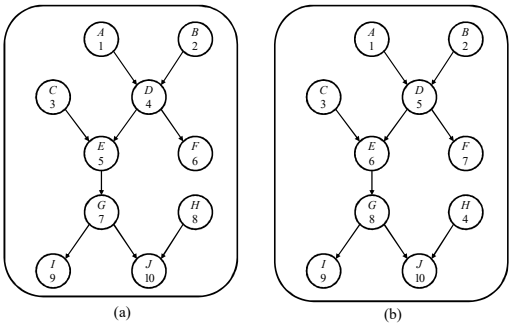


図2.13 同一の有向グラフにおける異なる二つのアセンストラルナンバリング

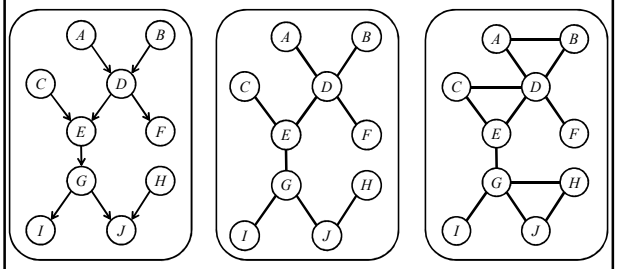
17. モラルグラフ

定義43
有向グラフにおける有向エッジを無向エッジに置き換えたグラフを**有向グラフに対応した無向グラフ**（undirected graph associated with a directed graph）と呼ぶ．

定義44
有向グラフにおいて共通の子ノードをもつすべての親ノードの対にエッジを張り，方向性を取り除いて構築された「有向グラフに対応した無向グラフ」を**モラルグラフ**（moral graph）と呼ぶ．

例21
有向グラフ，有向グラフに対応した無向グラフ，対応したモラルグラフの例を，それぞれ図**2.14**(a), (b), (c) に示した．

17. モラルグラフ



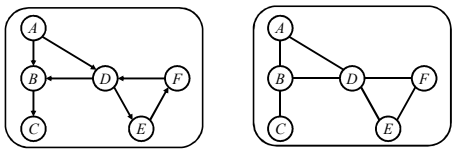
(a) 有向グラフ　(b) 有向グラフに対応した無向グラフ　(c) 対応したモラルグラフ

図2.14 モラルグラフの例

18. 循環

定義45
循環（cycle）とは，有向グラフにおける閉路のことをいう．

例22
図**2.15** に，循環をもつ有向グラフとそれに対応したループをもつ無向グラフを示した．



（a）　（b）

図2.15 ループと循環

19. 有向グラフのタイプ

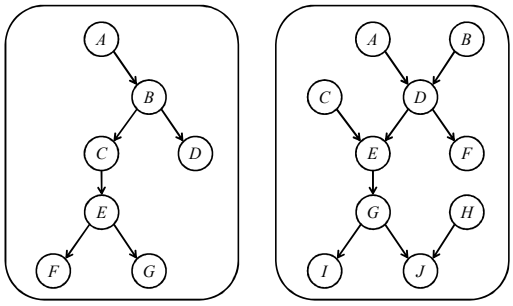
定義46
有向グラフの対応する無向グラフが連結グラフであるとき，そのグラフを**連結有向グラフ**（connected directed graph）と呼ぶ．それ以外の場合，**非連結有向グラフ**（disconnected directed graph）と呼ぶ．

定義47
有向グラフの対応する無向グラフが木であるとき，そのグラフも**有向木**（directed tree）と呼ばれる．それ以外の場合，**複連結有向グラフ**（multiplyconnected directed graph）と呼ぶ．

定義48
少なくとも一つ以上の循環をもつ有向グラフを**循環グラフ**（cyclic graph）と呼ぶ，それ以外のグラフを**非循環グラフ**（directed acyclic graph）と呼び，略して**DAG** と呼ばれる．

定義49
多くとも一つだけの親ノードしかもたない木を**単結合木**（simple tree）と呼び，それ以外の木を**複結合木**（polytree）と呼ぶ．

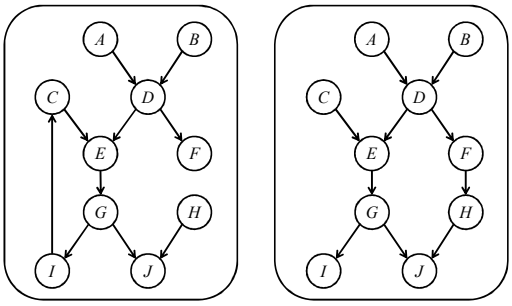
19. 有向グラフのタイプ



(a) 単結合木　(b) 複結合木

図2.16 単結合木と複結合木の例

19. 有向グラフのタイプ



(a) 循環グラフ　(b) 複連結グラフ

図2.17 循環グラフと複連結グラフの例

20. 弦

定義50
ループの**弦**(chord)とは，ループ中の2 ノードに張られたエッジを示し，そのループを分断し二つのループに分解するようなものをいう．
例24
図2.19 では，エッジ$E-G$ はループ$E-F-G-D-E$ の弦である．

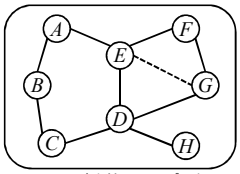


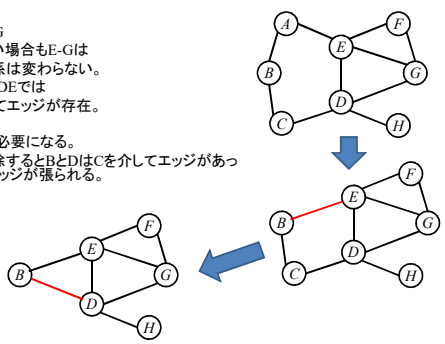
図2.19 弦を持つループの例

21. 三角グラフ

定義51
4 以上の長さをもつすべてのループが少なくとも一つの弦をもつとき，**三角グラフ**(triangulated graph)もしくは**コーダルグラフ**(chordal graph)と呼ぶ．

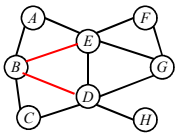
21. 三角グラフ

イメージ
部分グラフ EFG
についてFがない場合もE-Gは
エッジがあり関係は変わらない。
部分グラフABCDEでは
BとEはAを介してエッジが存在。
Aを削除すると
BとEにエッジが必要になる。
BCEDのCを削除するとBとDはCを介してエッジがあったので、BDにエッジが張られる。



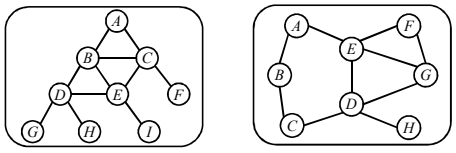
21. 三角グラフ

イメージ
部分グラフ EFG
についてFがない場合もE-Gは
エッジがあり関係は変わらない。
部分グラフABCDEでは
BとEはAを介してエッジが存在。
Aを削除すると
BとEにエッジが必要になる。
BCEDのCを削除するとBとDはCを介してエッジがあったので、BDにエッジが張られる。



22. 三角化

例25
図2.20 (a), (b) に，それぞれ三角グラフと非三角グラフの例を示す．三角グラフでない場合，グラフは**フィルイン**（fill–in）もしくは**三角化**（triangulation）と呼ばれる手法によって三角グラフを構成できる．

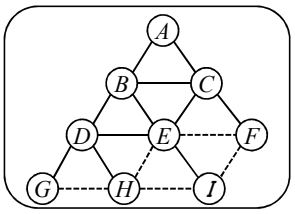


(a) 三角グラフ　(b) 非三角グラフ

図2.20 三角グラフと非三角グラフ

22. 三角化

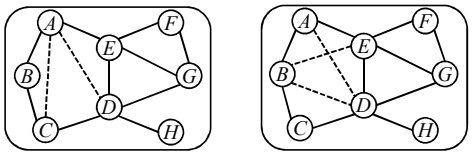
ただし，三角グラフとはグラフを三角形に分配することではない．つまり，図2.21 のように三角形をつくるためにエッジを追加する必要はない．



(a) 三角グラフの誤解

22. 三角化

ループに対する弦は複数考えられるため，当然，一つのグラフを三角化する方法は複数考えられる．図2.22 (a), (b) は同一グラフを二つの方法で三角化した例である．



(a) (b)

図2.22 同一グラフからの二つの異なる三角化

- 先のグラフからノードを消去する順序によって得られる三角グラフが異なる！！

ノードの消去順の　順序を考える！！

23. 完全ナンバリング

定義52
ナンバリングαが，対象グラフの部分集合ノードについて

$$Bnd(\alpha(i)) \cap \{\alpha(1), \cdots, \alpha(i-1)\}$$

が$i = 2, \cdots, N$について完全であるとき，**完全ナンバリング**（complete numbering）と呼ぶ．

定理13
無向グラフが三角グラフであるときのみに，完全ナンバリングが可能となる．

24. MCSナンバリング

アルゴリズム1（MCSナンバリング）
- Input:無向グラフ$G=(\mathbf{V}, \mathbf{E})$と最初のノード$V_1$
- Output:Vにおけるナンバリングα

1. main

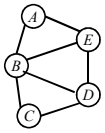
Initial Step:
2. $\alpha(1) \leftarrow V_1$
3. $Numbered \leftarrow \{V_1\}$

Iteration Step:
4. for $i = 2$ to N
5. $V_k \leftarrow$ choose a node V_k in $\mathbf{V} \setminus Numbered$ with maximum $|Nbr(V_k) \cap Numbered|$
6. $\alpha(i) \leftarrow V_k$
7. add V_k to $Numbered$
8. end for
9. return $Numbered$
10. end procedure

例

α(1)=C
$|Nbr(V_k) \cap Numbered|$
→BかD　α(2)=B
α(3)=D
α(4)=E
α(5)=A

完全ナンバリング　{C, B, D, E, A}



24. MCSナンバリング

定理14
三角グラフからのMCSアルゴリズムを用いて生成されるすべてのナンバリングは完全ナンバリングである.

すなわち，MCS アルゴリズムによって完全ナンバリングが得られないとき，そのグラフは三角グラフでない．MCS アルゴリズムは，与えられたグラフが三角グラフかどうかをチェックすることができるように簡単に改良することができる．すなわち，グラフが三角グラフでないときには，アルゴリズムはフィルインと呼ばれる三角グラフを構成するために足らないエッジを継ぎ足すという手続きを行う．それが以下のMCS Fill–in アルゴリズムである.

25. MCS Fill–in アルゴリズム

アルゴリズム2（MCS Fill-inアルゴリズム）
- Input:無向グラフG=(**V**, **E**)と最初のノードVs
- Output:$G' = (\mathbf{V}, \mathbf{E} \cup \mathbf{E}')$が三角グラフになるような$\mathbf{E}'$

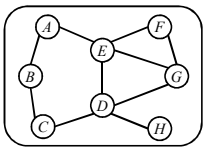
1. main
2. $\mathbf{E}' \leftarrow \boldsymbol{\phi}$
3. $\alpha(1) \leftarrow V_s$, $Numbered \leftarrow \{V_s\}$

Iteration Step:

4. $V_k \leftarrow$ choose a node V_k in $\mathbf{V} \setminus Numbered$ with maximum $|Nbr(V_k) \cap Numbered|$
   $\alpha(i) \leftarrow V_k$, add V_k to $Numbered$
5. if $Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合でない then
   $Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合になるよう
   $\mathbf{E}'$に最小数のエッジを加え3. に戻る
6. if $i < N$ then $i = i + 1$, 4. に戻る
7. return $\mathbf{E}'$
8. end procedure

演習

左図の非三角グラフをMCS fill-inによって三角化してみよう



アルゴリズム2（MCS Fill-inアルゴリズム）
- Input:無向グラフG=(**V**, **E**)と最初のノードVs
- Output:$G' = (\mathbf{V}, \mathbf{E} \cup \mathbf{E}')$が三角グラフになるような$\mathbf{E}'$

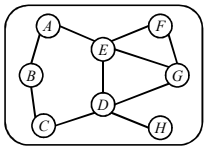
1. main
2. $\mathbf{E}' \leftarrow \boldsymbol{\phi}$
3. $\alpha(1) \leftarrow V_s$, $Numbered \leftarrow \{V_s\}$

Iteration Step:

4. $V_k \leftarrow$ choose a node V_k in $\mathbf{V} \setminus Numbered$ with maximum $|Nbr(V_k) \cap Numbered|$
   $\alpha(i) \leftarrow V_k$, add V_k to $Numbered$
5. if $Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合でない then
   $Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合になるよう
   $\mathbf{E}'$に最小数のエッジを加え3. に戻る
6. if $i < N$ then $i = i + 1$, 4. に戻る
7. return $\mathbf{E}'$
8. end procedure

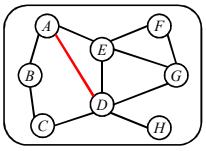
回答

α(1)=C
→BかD　α(2)=B
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→AかD　α(3)=A
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→EかD　α(4)=D
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→EかH　α(5)=E
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合でない



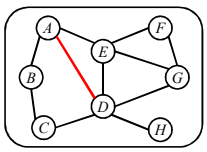
回答

α(1)=C
→BかD　α(2)=B
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→AかD　α(3)=A
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→EかD　α(4)=D
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→EかH　α(5)=E
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合でない
ADにフィルインエッジを引く。
アルゴリズム　Line3に戻る



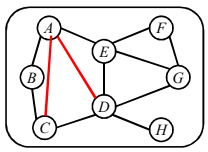
回答

α(1)=C
→BかD　α(2)=B
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→AかD　α(3)=A
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→α(4)=D
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合でない



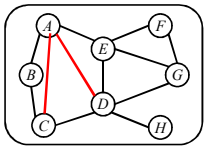
回答

α(1)=C
→BかD　α(2)=B
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→AかD　α(3)=A
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→α(4)=D
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合でない
ACにフィルインエッジを引く。
アルゴリズム　Line3に戻る



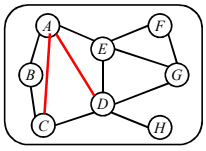
回答

α(1)=C
→BかD　α(2)=B
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→AかD　α(3)=A
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→α(4)=D
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合でない
ACにフィルインエッジを引く。
アルゴリズム　Line3に戻る



回答

α(1)=C
→AかBかD　α(2)=B
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→α(3)=A
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→α(4)=D
$Nbr(V_k) \cap \{\alpha(1), \ldots, \alpha(i-1)\}$が完全集合
→α(5)=E　→α(6)=G →α(7)=F →α(8)=H



MCS Fill-inアルゴリズムの問題

残念なことに，MCS fill–in アルゴリズムはベイジアンネットワークの計算量にとって最適な三角化を保証しない．三角化の最適化については4 章で述べる．

26. 交差法則

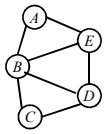
定義53
交差法則$C_i \cap (C_1 \cup C_2 \cdots C_{i-2} \cup C_{i-1})$が$\{C_1, \cdots, C_{i-1}\}$の少なくとも一つのクリークに含まれているとき，無向グラフのクリークの順序$(C_1, \cdots, C_m)$は**交差法則**(running intersection property)を満たすという．

この性質は，与えられたクリークとそれまでのクリーク集合の重複ノードが，少なくともそれまでのクリーク集合の一つに含まれるようにグラフのクリークを順序化できる，ということを示している．このとき，交差法則を満たすクリークの順序を**クリークチェーン**(chain of clieques)と呼び，以下の性質をもつ．

定理15
無向グラフは，それが三角グラフである場合にのみ，対応するクリークチェーンをもつ．

例

- C_1 =ABE, C_2 =BDE, C_3=BCD
- $C_3 \cap (C_1 \cup C_2)$=BD
- BDはC_2 に含まれている。



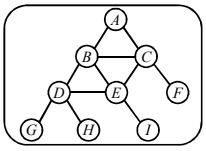
27. クリークチェーンの生成

アルゴリズム3（クリークチェーンの生成）
- Input:三角グラフG=(**V** , **E**)
- Output:グラフGのクリークチェーン(C_1, ・・・, C_m)

1. main
2. 初期化：ある初期ノードを選択し，アルゴリズム1 を用いてノードV_1, ・・・,V_N の完全ナンバリングを得る.
3. グラフのクリークを計算する.
4. そのノードの完全ナンバリングにおける最大値を各クリークに付与する.
5. 振り分けられた番号に従って，クリーク(C_1, ・・・, C_m) に並べ替える.
6. return (C_1, ・・・, C_m)

演習

左グラフよりクリークチェーンを生成せよ

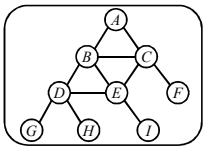


アルゴリズム3（クリークチェーンの生成）
- Input:三角グラフG=(**V** , **E**)
- Output:グラフGのクリークチェーン(C_1, ・・・, C_m)

1. main
2. 初期化：ある初期ノードを選択し，アルゴリズム1 を用いてノードV_1, ・・・,V_N の完全ナンバリングを得る.
3. グラフのクリークを計算する.
4. そのノードの完全ナンバリングにおける最大値を各クリークに付与する.
5. 振り分けられた番号に従って，クリーク(C_1, ・・・, C_m) に並べ替える.
6. return (C_1, ・・・, C_m)

演習

左グラフよりクリークチェーンを生成せよ



アルゴリズム1（MCSナンバリング）
- Input:無向グラフG=(**V** , **E**)と最初のノードV_i
- Output:Vにおけるナンバリングα

1. main

Initial Step:

2. $\alpha(1) \leftarrow V_i$
3. $Numbered \leftarrow \{V_i\}$

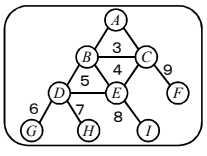
Iteration Step:

4. for $i = 2$ to N
5. $V_k \leftarrow$ choose a node V_k in **V** $\setminus Numbered$ with maximum $|Nbr(V_k) \cap Numbered|$
6. $\alpha(i) \leftarrow V_k$
7. add V_k to $Numbered$
8. end for
9. return $Numbered$
10. end procedure

完全順序
$\alpha(1)$=A
B or C $\alpha(2)$=B
$\alpha(3)$=C
$\alpha(4)$=E
$\alpha(5)$=D
F or G or H or I
$\alpha(6)$=G
$\alpha(7)$=H
$\alpha(8)$=I
$\alpha(9)$=F

演習

左グラフよりクリークチェーンを生成せよ



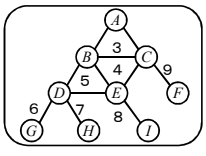
アルゴリズム3（クリークチェーンの生成）
- Input:三角グラフG=(**V** , **E**)
- Output:グラフGのクリークチェーン(C_1, ・・・, C_m)

1. main
2. 初期化：ある初期ノードを選択し，アルゴリズム1 を用いてノードV_1, ・・・,V_N の完全ナンバリングを得る.
3. グラフのクリークを計算する.
4. そのノードの完全ナンバリングにおける最大値を各クリークに付与する.
5. 振り分けられた番号に従って，クリーク(C_1, ・・・, C_m) に並べ替える.
6. return (C_1, ・・・, C_m)

完全順序
$\alpha(1)$=A
B or C $\alpha(2)$=B
$\alpha(3)$=C
$\alpha(4)$=E
$\alpha(5)$=D
F or G or H or I
$\alpha(6)$=G
$\alpha(7)$=H
$\alpha(8)$=I
$\alpha(9)$=F

演習

左グラフよりクリークチェーンを生成せよ



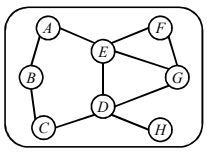
アルゴリズム3（クリークチェーンの生成）
- Input:三角グラフG=(**V** , **E**)
- Output:グラフGのクリークチェーン(C_1, ・・・, C_m)

1. main
2. 初期化：ある初期ノードを選択し，アルゴリズム1 を用いてノードV_1, ・・・,V_N の完全ナンバリングを得る.
3. グラフのクリークを計算する.
4. そのノードの完全ナンバリングにおける最大値を各クリークに付与する.
5. 振り分けられた番号に従って，クリーク(C_1, ・・・, C_m) に並べ替える.
6. return (C_1, ・・・, C_m)

C_1=ABC
C_2=BCE
C_3=BDE
C_4=DG
C_5=DH
C_6=EI
C_7=CF

演習

- グラフを三角化し、
- クリーク順序を得よ。



28. ジョインツリー

定義54
グラフにおけるあるノード集合を**クラスター**(cluster)と呼ぶ.

定義55
グラフG = (**V**,**E**) と**V** = $C_1 \cup C_2 \cdot\cdot\cdot C_{m-1} \cup C_m$ となるような**V** のクラスター集合**C** = (C_1, ・・・, C_m)を所与として, クラスター間の共通ノードが存在するときのみエッジが存在する, すなわち, (C_i, C_j) ∈ **E'** ならば$C_i \cap C_j \neq \varphi$ のとき, グラフG = (**C**,**E'**) を**クラスターグラフ**(cluster graph)と呼ぶ.

28. ジョインツリー

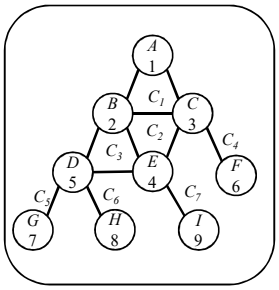


図2.24　三角グラフからの完全ナンバリング

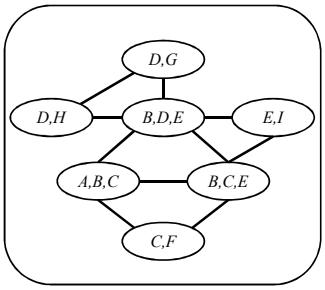


図2.25　図2.24に対応したクラスターグラフ

29. ジョイングラフ

定義56
クラスターがクリークによって構成されるクラスターグラフを**クリークグラフ**(clique graph)と呼ぶ.

定義57
共通のノードをもつ二つのクリーク間すべてにエッジが引かれているクリークグラフを**ジョイングラフ**(join graph), または**ジャンクショングラフ**(junction graph)と呼ぶ.

与えられたグラフに対し, ジョイングラフは一意に決まる. また, ジョイングラフの中で共通のノードをもつクリーク同士は完全グラフを構成する. したがって, ジョイングラフはそれがたとえ小さなグラフであっても密にエッジが張られたグラフとなることが多い.

30. ジョインツリー

定義58
クリークグラフが木で, 二つのクラスターに属するすべての共通ノードがそれらの間に存在する路のすべてのクラスターにも属しているとき, **ジョインツリー**(join tree)または**ジャンクションツリー**(junction tree)と呼ぶ.

例29
図2.26 では, ジョイングラフから四つのエッジを削除して(図(a))ジョインツリーを得ている(図(b)).

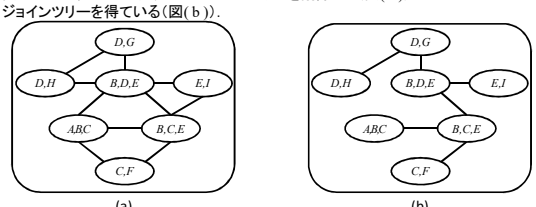


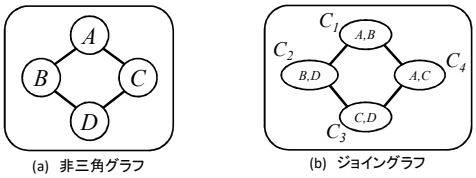
図2.26　ジョイングラフとジョインツリー

定理

- ジョイングラフは　三角グラフであるとき、対応するジョインツリーを持つ。

30. ジョインツリー

例30
図2.27 は，非三角グラフであるために(図(a))，対応したジョイングラフがジョインツリーをもたない(図(b))ことを示している．例えば，C_1 $-C_4$を図(b) のクラスターグラフから削除すればグラフはツリーとなるが，これはノードA がC_1 とC_4 に含まれるのにC_2, C_3 は含まれないので，ジョインツリーではなくなる．



(a) 非三角グラフ　(b) ジョイングラフ

図2.27 ジョイングラフとジョインツリー

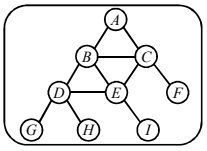
31. ジョインツリーの生成

アルゴリズム4(ジョインツリーの生成)
- Input: 三角グラフ$G = (V, E)$
- Output: Gに対応したジョインツリー$G' = (C, E')$

1. main
2. 初期化：アルゴリズム3を用いてグラフGのクリークチェーン $C_1, \dots, C_m$を生成する．
3. 各クリーク$C_i \in C(i = m, \dots, 1)$について，$\{C_1, \dots, C_{i-1}\}$から最大数の共通ノードを持つクリーク$C_k$を選択し，エッジ$C_i - C_k$を$E'$(初期状態では空)に追加する．最大数の共通ノードを持つクリークが複数ある場合，そのうちの一つを任意に選択してよい．
4. return $G' = (C, E')$
5. end procedure

演習

左グラフよりジョインツリーを生成せよ

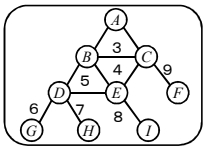


アルゴリズム4(ジョインツリーの生成)
- Input: 三角グラフ$G = (V, E)$
- Output: Gに対応したジョインツリー$G' = (C, E')$

1. main
2. 初期化：アルゴリズム3を用いてグラフGのクリークチェーン$C_1, \dots, C_m$を生成する．
3. 各クリーク$C_i \in C(i = m, \dots, 1)$について，$\{C_1, \dots, C_{i-1}\}$から最大数の共通ノードを持つクリーク$C_k$を選択し，エッジ$C_i - C_k$を$E'$(初期状態では空)に追加する．最大数の共通ノードを持つクリークが複数ある場合，そのうちの一つを任意に選択してよい．
4. return $G' = (C, E')$
5. end procedure

演習

左グラフよりジョインツリーを生成せよ



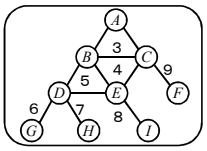
C_1=ABC
C_2=BCE
C_3=BDE
C_4=DG
C_5=DH
C_6=EI
C_7=CF

アルゴリズム4(ジョインツリーの生成)
- Input: 三角グラフ$G = (V, E)$
- Output: Gに対応したジョインツリー$G' = (C, E')$

1. main
2. 初期化：アルゴリズム3を用いてグラフGのクリークチェーン$C_1, \dots, C_m$を生成する．
3. 各クリーク$C_i \in C(i = m, \dots, 1)$について，$\{C_1, \dots, C_{i-1}\}$から最大数の共通ノードを持つクリーク$C_k$を選択し，エッジ$C_i - C_k$を$E'$(初期状態では空)に追加する．最大数の共通ノードを持つクリークが複数ある場合，そのうちの一つを任意に選択してよい．
4. return $G' = (C, E')$
5. end procedure

演習

左グラフよりジョインツリーを生成せよ

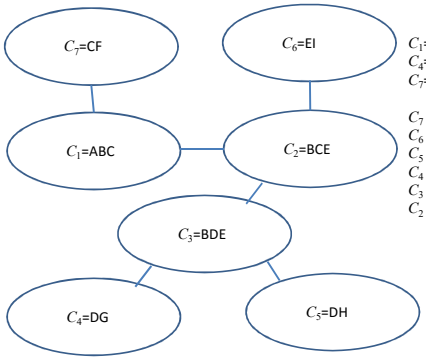


C_1=ABC C_2=BCE C_3=BDE
C_4=DG C_5=DH C_6=EI
C_7=CF

$C_7 - C_1$: C_1 *or* C_2
$C_6 - C_2$: C_2 *or* C_3
$C_5 - C_3$
$C_4 - C_3$
$C_3 - C_2$: C_1 *or* C_2
$C_2 - C_1$

アルゴリズム4(ジョインツリーの生成)
- Input: 三角グラフ$G = (V, E)$
- Output: Gに対応したジョインツリー$G' = (C, E')$

1. main
2. 初期化：アルゴリズム3を用いてグラフGのクリークチェーン$C_1, \dots, C_m$を生成する．
3. 各クリーク$C_i \in C(i = m, \dots, 1)$について，$\{C_1, \dots, C_{i-1}\}$から最大数の共通ノードを持つクリーク$C_k$を選択し，エッジ$C_i - C_k$を$E'$(初期状態では空)に追加する．最大数の共通ノードを持つクリークが複数ある場合，そのうちの一つを任意に選択してよい．
4. return $G' = (C, E')$
5. end procedure

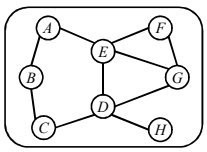


C_1=ABC C_2=BCE C_3=BDE
C_4=DG C_5=DH C_6=EI
C_7=CF

$C_7 - C_1$: C_1 *or* C_2
$C_6 - C_2$: C_2 *or* C_3
$C_5 - C_3$
$C_4 - C_3$
$C_3 - C_2$: C_1 *or* C_2
$C_2 - C_1$

演習

- グラフを三角化し、ジョインツリーを得よ。



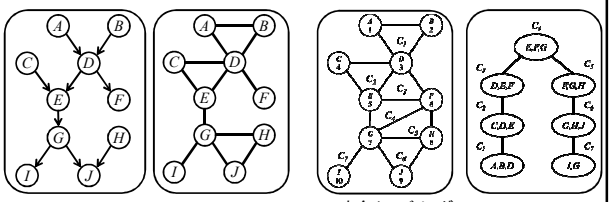
これまでのところ，無向グラフのクラスターツリー構築問題を取り扱ってきた．しかし，ベイジアンネットワークに適用するために，クラスターツリーの概念は，有向グラフに対応した無向グラフを用いることにより適用することができる．このとき，有向グラフにおけるノードのファミリーがエビデンス更新で重要な役割を担う．いくつかの確率モデルは，ノードのファミリーのローカルな機能を通して規定される．以上のように，グラフのすべてのファミリーが少なくとも一つのクラスターに含まれるようなクラスターに注目することができる．このため，以下の定義を行う．

32．ファミリーツリー

定義**59** (ファミリーツリー)
有向グラフD の**ファミリーツリー**（family tree）は，すべてのノードのファミリーが少なくとも一つのクラスターに含まれるDに対応した無向グラフG のジョインツリーのことである．

33．ファミリーツリーの生成

例32 （ファミリーツリーの生成）
(a)の有向グラフを考える．アルゴリズム5に適用してみよう．
1. (a) に対応したモラルグラフ(b)を生成する．
2. アルゴリズム2 に適用して得られた三角グラフ(c)を得る．
3. 最後に，アルゴリズム4 を適用して対応するファミリーツリー(d)を生成する．



(a) 有向グラフ (b) モラル化されたグラフ (c) 完全ナンバリングとクリーク順序 (d) 対応するジョインツリー