5. ジョインツリーアルゴリズム

植野真臣
電気通信大学
大学院情報システム学研究科

1. ジョンツリーアルゴリズム

前章で紹介した変数消去法は，計算量は$O(N^2\exp(w))$であった．
ここでwはファクター中の最大ウィズである．今回，紹介するジョインツリーアルゴリズムは，$O(\mathrm{N}\exp(w))$ まで減じることができる．

計算量

アルゴリズムで重要なことは、少ない計算量（短い時間）で、解に達することである。
例えば、$2x^3+4x^2+2x+5$ を計算するのに、
$2\times x\times x\times x+4\times x\times x+2\times x+5$ ・・・A
とすると、乗算の回数は6回になる。
この式を変形して、
$(((2x+4)x+2)x)+5$ ・・・B
とすれば、乗算の回数は3回になる。
一般化して、n次式、
$a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$
をAのように計算するならば、乗算回数は
$(n+1)+n+\cdots+1+0=n(n+1)=(1/2)n^2+(1/2)n$ ・・・C
になる。また、Bの形式に変形して
$((\ldots(a_n\times x+a_{n-1})\times x+a_{n-2})\times\ldots)\times x+a_0$
とすれば、乗算回数はn回になる。
ここで、乗算の回数を計算量の尺度にしたのは、加減算に比べて乗除算に要する時間は非常に大きいからである。
このように、アルゴリズムでの主要な操作に着目して、解に達するまでに必要な操作回数のことを**計算量**（注）（オーダー）といい、O(nの式) で表現する。

計算量O

- O(nの式) の表現では、次のように簡略化する。
  - ・多項式になる場合、n→大としたとき最大になる項だけにする。
    （Cの場合：$(1/2)n^2+(1/2)n$ を $(1/2)n^2$ とする）
  - ・係数を無視する。（Cの場合：$(1/2)n^2$ を n^2 とする）

  すなわち、Cの計算量は$O(n^2)$ となる。
- このようにする理由は、
  - ・計算量が問題になるのは、nが大きいときに限られる。
  - ・nの式による違いに対して、係数の違いは相対的に影響が小さい。

P問題

- 計算量が多項式$O(n_m)$ になるアルゴリズムが存在する問題を**P問題**と呼ぶ。
- P問題は、数学者にとっては、効率的に解ける問題だとして取り扱われる。
  ところが、一般的な解法が見つからない問題や、解法はあるのだが、$O(n_m)$ にはならない問題（$O(2_n)$ や $O(n!)$ が多くある。それを**NP問題**と呼ぶ。
- 数学的な関心事としては、
  P＝NP（本当はP問題として解けるのに、未だ発見されていないだけだ）
  P≠NP（本当にP問題にはならない問題があるのだ）
  のどちらなのだろうかが、大きな課題となる。

NP困難

- NPに属すとは限らないが、NPに属する任意の問題と比べて、少なくとも同等以上に難しいこと
- NP完全問題とは、NP困難であり、かつNPに属する問題である。NPの中でもっとも難しい問題。

2. ファクター消去

ベイジアンネットワークを所与としたとき，クエリ変数集合Qの周辺事前確率を求めることを考える．変数消去アルゴリズムは，ネットワークからQ以外の変数を一つずつ周辺消去して周辺事前確率が得られた．一方，ファクター消去アルゴリズムでは，Qを含むファクター以外のすべてのファクターを消去することによって周辺事前確率を得る．

ファクター消去アルゴリズム

定義73（ファクター消去アルゴリズム）
以下のように，ファクター集合Sからファクターφ_iを消去する．まず，ファクターφ_iにのみ出現するすべての変数集合Vを消去し，その結果の$\sum_V \varphi_i$に集合Sの他のファクターφ_iを掛け合わせる．

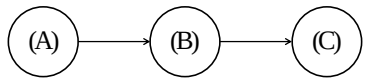
アルゴリズム11（周辺事前確率のためのファクター消去アルゴリズム）

- **Input:** ベイジアンネットワーク$\{G, \Theta\}$，ベイジアンネットワークのクエリ変数集合Q
- **Output:** ベイジアンネットワークp$(Q|G)$

1. **main**
2. S←ベイジアンネットワークのすべてのファクター
3. φ_r← Qを含むS 中のファクター
4. **while** Sに一つ以上のφ_r 以外のファクターが残っている **do**
5. Sからファクター$\varphi_i \neq \varphi_r$ を一つ消去
6. **V**←ファクターφ_i に出現してSに含まれない変数
7. S 中のあるφ_j について$\varphi_j \leftarrow \varphi_j \sum_V \varphi_i$
8. **end while**
9. **return** p$(Q|G) = project(\varphi, Q)$
10. **end procedure**

演習

例41 **図4.1**のような簡単なベイジアンネットワークについて，アルゴリズム11を用いて変数Cについての周辺事前確率を求めてみよう．



A	φ(A)
真	0.6
偽	0.4

A	B	φ(B|A)
真	真	0.9
偽	偽	0.1
偽	真	0.2
偽	偽	0.8

B	C	Φ(C|B)
真	真	0.3
偽	偽	0.7
偽	真	0.5
偽	偽	0.5

図4.1 簡単なベイジアンネットワーク例[1)]

3. エリミネーションツリー

前章で示したように，変数消去アルゴリズムでは変数消去順序により計算効率が大きく変化した．同様にファクター消去アルゴリズムでも，変数消去順序が重要な役割を果たし，**エリミネーションツリー**（elimination trees）によって示される．

定義74 ファクター集合Sのエリミネーションツリーは，木Tとファクター集合φのペア(T, φ)によって示される．ファクター集合S 中の各ファクターはT の一つのノードに割り当てられ，ノードi に対応したファクターをφ_i と書き，φ_i に含まれるすべての変数集合を$vars(\varphi_i)$ と書く．ノード間のエッジは，木が構成されるのであればどのように引いてもよい．

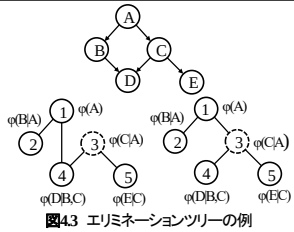


図4.3 エリミネーションツリーの例

図4.3の，上図の5ノードのベイジアンネットワークは，五つのファクター$\varphi(A), \varphi(B|A), \varphi(C|A), \varphi(D|B,C), \varphi(E|C)$ をもつ．これらの各ファクターごとにそれぞれ1, 2, 3, 4, 5と番号を割り当て，それぞれをノードとしたツリーを構成したものが図4.3の下図の二つのエリミネーションツリーである．

エリミネーションツリーを用いるとき，「ルート」と呼ばれるrをQ ⊆ $vars(r)$となるように選択しなければならない．図4.3でノードC の周辺確率を求めたい場合，Cを含むノード3, 4, 5 のーつをルートとして選べばよい．

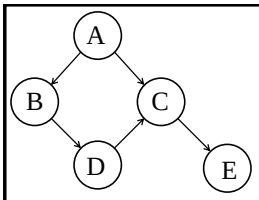
そして，ファクターφ_i が$i \neq r$，かつ，たったーつの隣接ノードj をもつ場合にのみ，ファクターφ_i を消去する．ファクターφ_i を消去するために，まず，φ_i に出現するが残ったツリーには存在しない変数**V**についてファクターφ_i を足し合わせ，その結果$\sum_V \varphi_i$ を，そのーつだけの隣接ノードj に対応するファクターφ_j に掛け合わせればよい．

アルゴリズム12(エリミネーションツリーによるファクター消去アルゴリズム)

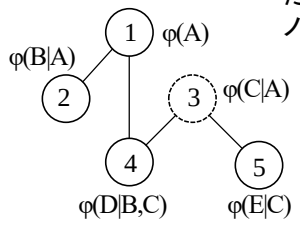
- **Input:** ベイジアンネットワーク$\{G, \Theta\}$, ベイジアンネットワークのクエリ変数集合Q
- (T, φ): エリミネーションツリー
- $r: Q \subseteq vars(r)$ であるツリーT のノード
- **Output:** $\mathrm{p}(Q|G)$

1. **main**
2. **while** ツリーT に二つ以上のノードをもつ **do**
3. ツリーT から隣接ノードに一つしかないノード$i \neq r$を消去
4. **V** $\leftarrow \varphi_i$ にあるが, 残ったツリーT にはない変数
5. $\varphi_j \leftarrow \varphi_j \sum_{\mathbf{V}} \varphi_i$
6. **end while**
7. **return** $project(\varphi_r, Q)$
8. **end procedure**

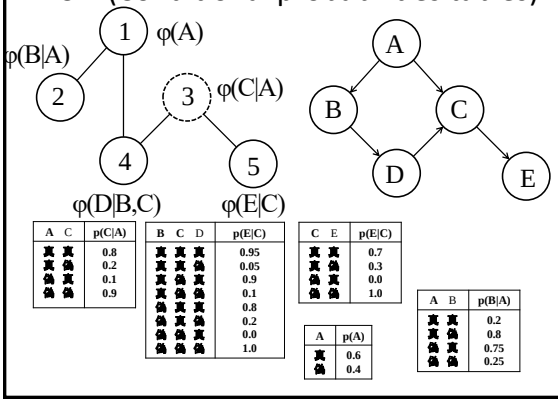
演習



左上のベイジアンネットワークのCの周辺確率を左下のエリミネーションツリーによりもとめてみよう.
ただし、CPTは次ページ。
ノード3をルートとせよ.



CPT(Conditional probabilities tables)



A	C	p(C|A)
真	真	0.8
真	偽	0.2
偽	真	0.1
偽	偽	0.9

B	C	D	p(E|C)
真	真	真	0.95
真	真	偽	0.05
真	偽	真	0.9
真	偽	偽	0.1
偽	真	真	0.8
偽	真	偽	0.2
偽	偽	真	0.0
偽	偽	偽	1.0

C	E	p(E|C)
真	真	0.7
真	偽	0.3
偽	真	0.0
偽	偽	1.0

A	p(A)
真	0.6
偽	0.4

A	B	p(B|A)
真	真	0.2
真	偽	0.8
偽	真	0.75
偽	偽	0.25

4. セパレータとクラスター

どのようなエリミネーションツリーでも, 正しい結果を出すことが保証されている. しかし, 変数消去法同様にエリミネーションツリーすなわち消去順序により最終的な計算量は大きく変化する. この計算量を軽減させようとするアルゴリズムが2章で紹介した古典的なジョインツリーアルゴリズムである. ジョインツリーアルゴリズムを導入する前に, アルゴリズム12は, セパレータとクラスターという概念を導入することにより, より簡潔なアルゴリズムに書き直すことができる. 次節では, セパレータとクラスターを紹介する.

セパレータ

定義75 エリミネーションツリーT のエッジ (i,j) の**セパレータ**(separator)は以下のように定義される.

$$\mathbf{S}_{ij} \equiv vars(i,j) \cap vars(j,i),$$

ここで, $vars(i,j)$ はエッジ (i,j) のi 側にあるファクターに出現するすべての変数集合を示し, $vars(j,i)$ はエッジ (i,j) のj側にあるファクターに出現するすべての変数集合を示している.

すべてのファクターに出現する変数集合
→
それぞれのファクターに出現する変数の和集合

例43 **図4.5**は, 図4.3の左下のエリミネーションツリーにセパレータを加えたものである. 例えば, $\mathbf{S}_{14}$について考えよう. エッジ$(1,4)$のノード1側にあるファクター(φ_1, φ_2) に出現するすべての変数集合は, A とB であるので

$$vars(1,4) = \{A, B\}.$$

一方, エッジ$(1,4)$のノード4側にあるファクター(φ_3とφ_4とφ_5)に出現するすべての変数集合は, A, B, C, D, Eであるので

$$vars(4,1) = \{A, B, C, D, E\}.$$

結果として,

$$\mathbf{S}_{14} = \mathbf{S}_{41} = vars(1,4) \cap vars(4,1) = \{A, B\} \cap \{A, B, C, D, E\} = \{A, B\}.$$

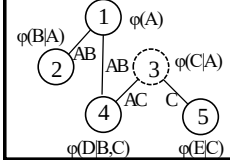


図4.5 エリミネーションツリーのセパレータ例

セパレータの利点

セパレータを導入することの最も重要な利点は，アルゴリズム12の5行目が非常に簡潔に書けるようになることである．すなわち，4行目，5行目が

$$\sum_{\mathbf{V}} \varphi_i = project(\varphi_i, \mathbf{S}_{ij}) \qquad (4.1)$$

と書ける．これより，アルゴリズム12の4行目は必要なくなり，よりアルゴリズムを簡潔に書き直すことができる．

アルゴリズム12（エリミネーションツリーによるファクター消去アルゴリズム）

- **Input:** ベイジアンネットワーク$\{G, \Theta\}$，ベイジアンネットワークのクエリ変数集合Q
- (T, φ): エリミネーションツリー
- $r: Q \subseteq vars(r)$ であるツリーT のノード
- **Output:** $\mathrm{p}(Q|G)$

1. **main**
2. **while** ツリーT に二つ以上のノードをもつ **do**
3. ツリーT から隣接ノードに一つしかないノード$i \neq r$を消去
4. $\varphi_j \leftarrow \varphi_j project(\varphi_i, \mathbf{S}_{ij})$
5. **end while**
6. **return** $project(\varphi_r, Q)$
7. **end procedure**

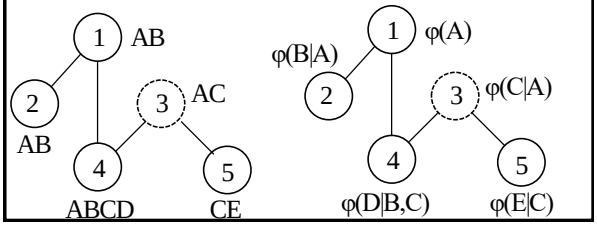
クラスター

定義76 エリミネーションツリーT のノードi のクラスター$\mathbf{C}_i$ は以下のように定義される．

$$\mathbf{C}_i \equiv vars(i) \cup \bigcup_j \mathbf{S}_{ij}\,.$$

ただし，エリミネーションツリーの $width$ は最大のクラスターサイズ -1 で求められる．

例44 **左図**は，右図に対応したクラスターをもつエリミネーションツリーのである．最大のクラスターサイズはノード4の4であり，エリミネーションツリーの$width$ は3となる．また，アルゴリズム12におけるφ_i に出現する変数集合は$\mathbf{C}_i$に一致する．したがって，アルゴリズム12はクラスターのすべての部分集合についての周辺確率の計算に利用できる．



5．メッセージパッシング

より一般化されたファクター消去アルゴリズムの定式化として，複数のエビデンスの情報伝搬にも対応できる**メッセージパッシング**（message-passing）アルゴリズム（Pearl 1988）が有用である．メッセージパッシングの利点は，一度計算したメッセージ伝搬の情報が再利用でき，複数のエビデンスの情報伝搬時の計算が容易に行える点である．エリミネーションツリーの枠組みでメッセージパッシングを定式化するためには，以下のようにエリミネーションツリーを有向木に変換しなければならない．

- エリミネーションツリーにおける各ノード$i \neq r$には固有のルートrに最も近い隣接ノードが存在する．エリミネーションツリーの各エッジについて，ノードiからそのルートrに最も近い隣接ノードに有向エッジを張り，エリミネーションツリーを有向木に変換する．
- ノードiは，ノードiからそのルートrに最も近いものを除くすべての隣接ノードが消去された後，消去される．
- ノードiが消去されるときには，単一の隣接ノードjをもつ．現在のファクターはiとjの間のセパレータに射影され，ノードjのファクターに掛け合わされる．

さらに，いま，ノードi が単一の隣接ノードj をもつ状態で，ノードi からノードj へのメッセージM_{ij} を伝搬させるプロセスは以下のように定式化できる．

1. j がメッセージM_{ij} を受け取ると，それをファクターφ_j に掛け合わせる．
2. ノードi は隣接ノードk $\neq j$ からのすべてのメッセージを受け取るまでi はj にメッセージを送れない．
3. ノードi がこれらのメッセージを受け取ると，現在のファクターは

   $\varphi_i \prod_{k \neq j} M_{ki}$

   となり，j へ送るメッセージは

   $M_{ij} \equiv project\left(\varphi_i \prod_{k \neq j} M_{ki}, \mathbf{S}_{ij}\right).$ (4.2)

例45 **図4.7**は，図4.3上図についてのノード5をルートとしたときの有向エリミネーションツリーの例である．いま，ノード4にノード1, 2, 3からメッセージが届いたとすると，この時のファクターは

$$\varphi_4 M_{14} M_{24} M_{34}$$

となり，ノード5へ送られるメッセージは

$$project(\varphi_4 M_{14} M_{24} M_{34}, \mathbf{S}_{45})$$

となる．

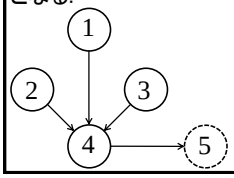


図4.7 有向エリミネーションツリーの例

さらに，他のクラスター$\mathbf{C}_i, (i \neq r)$について周辺化する場合には，新たにiを新しいルートとして選択し，先のメッセージパッシングのプロセスを繰り返せばよい．ただし，いくつかのメッセージは先のプロセスで計算しているので，それをメモリ上に格納しておけば，すべてのメッセージを再計算する必要はなくなる．これをシステマティックに行うために，一般には以下の二つのフェーズが実施される．

1.集積フェーズ（collect phase）：各ノードからルートに向けてメッセージが送信されるフェーズ．エリミネーションツリーにおける各ノード$i \neq r$には固有のルートrに最も近い隣接ノードが存在する．エリミネーションツリーの各エッジについて，ノードiからそのルートrに最も近い隣接ノードに向けて有向エッジを張り，エリミネーションツリーを有向木に変換し，各エッジのメッセージを計算し送信する．

2.分配フェーズ（distribute phase）：ルートから各ノードに向けてメッセージが配信されるフェーズ．エリミネーションツリーにおける各ノード$i \neq r$には固有のルートrに最も近い隣接ノードが存在する．エリミネーションツリーにおける各エッジについて，ノードiのルートrに最も近い隣接ノードからノードiに向かって有向エッジを張り，エリミネーションツリーを有向木に変換し，各エッジのメッセージを計算し送信する．エリミネーションツリーのノード数mについて$m-1$のエッジが存在し，このプロセスで$2(m-1)$のメッセージを計算すればよい．

エビデンスを得た後のファクター消去法も，変数消去法と同様に行うことができる．定義70により，変数消去法でのエビデンス**e** を所与としたときのファクター $\varphi^e(\mathrm{x})$は以下のように定義された．

$$\varphi^e(\mathrm{x}) = \begin{cases} \varphi(\mathrm{x}) & (\mathrm{x}\text{が}\mathbf{e}\text{に一致しているとき}) \\ 0 & (\text{上記以外}) \end{cases}$$

エビデンス**e** を所与としたときのファクター $\varphi^e(x)$をアルゴリズム12に単純に挿入することにより，周辺事後確率のためのファクター消去アルゴリズム13を得ることができる．

アルゴリズム13（周辺事後確率のためのファクター消去アルゴリズム）

- **Input:** ベイジアンネットワーク$\{G, \Theta\}$，ベイジアンネットワークのクエリ変数集合Q
- (T, φ): エリミネーションツリー
- **e**: エビデンス
- **Output:** $p(\mathbf{C}_i, \mathbf{e}|G)$

1. **for** エビデンス**e** 中の各変数E **do**
2. $i \leftarrow E \in \mathbf{C}_i$ を満たすツリーT 中のノード
3. $\varphi_j \leftarrow \varphi_i^e$: エビデンス情報をノード$i$ に入力
4. **end for**
5. ツリーT からルートとなるノードr を選択
6. 式(4.2)を用いてr 以外のすべてのノードからルートr へのメッセージ伝搬（集積フェーズ）
7. 式(4.2)を用いてrから他のすべてのノードへメッセージ伝搬（伝搬フェーズ）
8. **return** $\varphi_i \prod_k M_{ki}$ をツリーT 中の各ノードi について出力（同時周辺確率$p(\mathbf{C}_i, \mathbf{e}|G)$）

6．ジョインツリーアルゴリズム

アルゴリズム13に示したファクター消去アルゴリズムにより，クリークiの周辺事後確率が得られる．一般には計算量削減のため最小の*width* をもつツリーを構成したいが，構成されるツリーの*width* は入力されるエリミネーションツリー(T, φ) によってさまざまに変容してしまう．そこで最小の*width* をもつエリミネーションツリーを目指して提案されたのが**ジョインツリーアルゴリズム**（jointree algorithm）である（Lauritzen and Spiegelhalter 1988）．「ジョインツリーアルゴリズム」はジャンクションツリーアルゴリズム（junctiontree algorithm）とも呼ばれる．ジョインツリーアルゴリズムの定義は2章ですでに与えたが，ここで以下のように再定義しよう．

定義77 DAG Gのジョインツリー（jointree）は，木T とその中の各ノードi をクラスター$\mathbf{C}_i$ に写像する関数$\mathbf{C}$のペア$(T, \mathbf{C})$によって表現され，以下の性質を満たす．

1. クラスター$\mathbf{C}_i$ はDAG Gからのノード集合である．
2. DAG Gの各ファミリーにはどこかのクラスター$\mathbf{C}_i$ に含まれないといけない．
3. ノードが二つのクラスター$\mathbf{C}_i$と $\mathbf{C}_j$に含まれていれば，ジョインツリーのノードi とj を結ぶ路（path）上のすべてのクラスター$\mathbf{C}_k$に含まれなければならない．

ジョインツリーのエッジ$i-j$のセパレータは$\mathbf{C}_i \cap \mathbf{C}_j$として定義され，$\mathbf{S}_{ij}$と書く．ジョインツリーの*width* は，最大クラスターのサイズ−1で定義される．

定理25 エリミネーションツリーのクラスターは，定義77の三つの性質を満たす．

古典的なジョインツリーアルゴリズムで最も一般的に知られているのは以下のShenoy-Shaferアルゴリズムである（Shenoy and Shafer 1990）

1. 2章のアルゴリズム5により，ジョインツリー$(T, \mathbf{C})$を構成する．
2. 各クラスター$\mathbf{C}_i$に対応するCPTファクターφ_iを割り当て，エリミネーションツリー(T, φ)を構成する．
3. アルゴリズム13に適用する．

具体的には，Shenoy-Shaferアルゴリズムはアルゴリズム14として書ける．

アルゴリズム14（Shenoy-Shaferアルゴリズム）

- **Input:** ベイジアンネットワーク$\{G, \Theta\}$
- **e**: エビデンス
- **Output:** $p(\mathbf{C}_i, \mathbf{e}|G)$

1. **main**
2. アルゴリズム5により，ジョインツリー$(T, \mathbf{C})$を構成
3. 各クラスター$\mathbf{C}_i$に対応するCPTファクターφ_iを割り当て，エリミネーションツリー(T, φ)を構成
4. **for** エビデンス**e** 中の各変数E **do**
5. $i \leftarrow E \in \mathbf{C}_i$ を満たすツリーT 中のノード
6. $\varphi_j \leftarrow \varphi_i^e$: エビデンス情報をノード$i$ に入力
7. **end for**
8. ツリーT からルートr となるノードを選択
9. 式(4.2)を用いてr 以外のすべてのノードからルートr へのメッセージ伝搬（集積フェーズ）
10. 式(4.2)を用いてルートr から他のすべてのノードへメッセージ伝搬（分配フェーズ）
11. **return** $\varphi_i \prod_k M_{ki}$ をツリーT 中の各ノードi について出力（同時周辺確率$p(\mathbf{C}_i, \mathbf{e}|G)$）
12. **end procedure**

Shenoy-Shaferアルゴリズムの計算量は，

$$\sum \mathrm{O}\left(\left({N_i}^2 + N_i + 1\right)\exp(w)\right)$$

N_iは各クラスター中の変数の数

もう一つの有名なアルゴリズムは，HUGINアルゴリズム（Jensen, Lauritzen and Oleson 1990）である．Shenoy-ShaferアルゴリズムもHUGINアルゴリズムもLauritzen and Oleson (1988)を基礎として構築されたので基本的に似通っているが，隣接したクリークへのファクターの伝搬法が異なる．いま，ジョインツリーにおけるノードx_iの隣接ノードが$x_j, (j = 1, \cdots, m)$であるとする．集積フェーズで

$$M_{ij} = \varphi_{x_i} \prod_{j=1}^{m} \varphi_{x_j}$$

が計算され，それを分解フェーズでどのように隣接ノードに伝搬するかが問題となる．一つの解決法は，Shenoy-Shaferアルゴリズムのように各隣接ノードx_iでも同様に$M_{ij} \equiv project\left(\varphi_i \prod_{k \neq j} M_{ki}, \mathbf{S}_{ij}\right)$を繰り返すことである．

もう一つは，隣接ノードx_jへの伝搬メッセージとしてφ_jで除して

$$M_{ij} = \varphi_{x_i} \prod_k \varphi_{k_i} / \varphi_j$$

を送信し，

$$\varphi_j \leftarrow \varphi_j M_{ij}$$

としてノードjが受信する手法が考えられる．これがHUGINアルゴリズムである．すべての隣接ノードの積を一度しか計算しなくてよいので，HUGINアルゴリズムの計算量は

$$O(\mathrm{N}\exp(w))$$

となる．ただし，計算中にメモリ内で記憶しておかねばならない容量はHUGINアルゴリズムのほうがShenoy-Shaferアルゴリズムよりも大きい．計算量ならばHUGINのほうが効率がよく，メモリスペースではShenoy-Shaferアルゴリズムのほうが効率がよい．

7．変数消去順序の最適化

ファクター消去アルゴリズムの計算量はexp(*width*)に比例し，*width*はエリミネーションツリーの構成に依存していることについてすでに述べた．さらに，変数消去プロセスにおける*width*の計算法については，3章のアルゴリズム8でインターラクショングラフとして紹介した．

実は，このインターラクショングラフとは，2章の定義43で紹介したベイジアンネットワークをモラル化し，モラルグラフを得ることと同値である．定義43より，ベイジアンネットワークGからモラルグラフは以下のように得られる．

1. ベイジアンネットワークGにおいて，共通の子ノードをもつすべての親ノードペア間に無向エッジを引く．
2. Gのすべての有向エッジを無向エッジに変換する．

例46 変数消去順序$Order$に従った変数消去により，インターラクショングラフの系列$G_1, G_2, \cdots, G_N$が得られる．さらに，i番目に消去されるノードの隣接ノード集合のクリーク$\mathbf{C}_i$の系列，「クリークチェーン」も$\mathbf{C}_1, \cdots, \mathbf{C}_N$として得ることができる．

図4.3上のベイジアンネットワークについて，変数消去順序$Order = \{E, D, C, B, A\}$として，アルゴリズム8に従うと**図4.8**を得る．すなわち，インターラクショングラフの系列$G_1, G_2, \cdots, G_5$として得られたことになる．

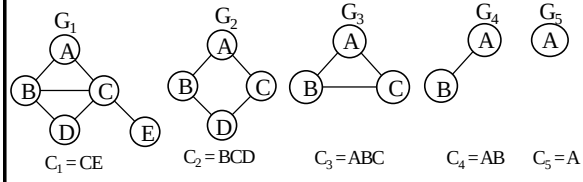


図4.8 インターラクショングラフとクリークチェーンの例

クリークチェーンとして$\mathbf{C}_1 = CE, \mathbf{C}_2 = BCD, \mathbf{C}_3 = ABC, \mathbf{C}_4 = AB, \mathbf{C}_5 = A$が得られる. モラル化により, $B - C$ 間にエッジが引かれているが, これはノードDが変数消去されたときに元のグラフにおける$B - C$間にエッジが引かれることに対応している. モラル化とは, 変数消去したときに追加されるエッジをあらかじめグラフに加えているにすぎない.

また, このとき, 計算量 $width$ は $Order$を所与として,

$$width(Order) = \max_{i=1}^{N} |\mathbf{C}_i| - 1$$

である.

定義78 ベイジアンネットワークG の $treewidth$ は

$$treewidth(G) = \min_{Oreder} width(Order)$$

と定義される.

また, 消去順序$Order$ の $width$ がG の $treewidth$ に一致するとき, **最適な消去順序**(the optimal order)という.

ただし, 最適な消去順序を求めるのは, NP困難であり(Wen 1990), $treewidth$は求めるのが困難である. そこで, 最適な消去順序を近似的に求めるために, 3章で紹介した以下の二つのヒューリスティックな手法が知られている.

- 最小次数法(min-degree):隣接ノード数が最小のノード順に消去する(アルゴリズム9)
- 最小フィルイン法(min-fill):最小のフィルインエッジ数の追加しか導かないノード順に消去する(アルゴリズム10)

一般に二つのアルゴリズムは, 最小隣接ノード数をもつ消去ノード候補や最小のフィルインエッジ数の追加しか導かない消去ノード候補が複数ある場合がある. 一般には, この二つの手法を組み合わせて消去ノードを決定し, それでも決定できない場合は乱数でタイブレークする.

定義79 ベイジアンネットワークG の消去プロセスにおいて, フィルインエッジの追加を必要としない消去順序$Order$ を**「完全」**(perfect)と呼ぶ.

この定義は2章「グラフ理論」での定義52に対応し, 以下の定理が成り立つ.

定理26 $Order$ が完全ナンバリング(定義52)であれば, 消去順序$Order$ は「完全」である.

すなわち, 変数を消去した後のフィルインエッジは三角化に対応していることがわかり, 最適な消去順序は最適な三角化の順序に対応する. ただし, 「完全」が計算量の意味で最適な三角化を保証しているわけではない.

また, 定理13より, 以下が成り立つ.

定理27 ベイジアンネットワークG が消去順序$Order$ が「完全」なときのみ, 対応する無向グラフは三角化されている.

定理28 クリークチェーン$\mathbf{C}_1, \cdots, \mathbf{C}_N$の最大クリークは, 三角化されたグラフの最大クリークに一致する.