修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報理工学研究科　情報・ネットワーク工学専攻　博士前期課程		
氏　　名	加藤 弘也	学籍番号	2331039
論 文 題 目	非制約型条件付き独立性の推移性を用いた 大規模ベイジアンネットワーク構造学習		

要　旨

ベイジアンネットワーク構造学習は変数数の増加に伴い膨大な計算時間を要するため，大規模変数に対する構造学習が困難である．近年，ベイジアンネットワークの推移性を RAI アルゴリズムに組み込むことで条件付き独立性検定（CI テスト）の数を大幅に削減し，大規模構造学習を可能にする手法が提案されてきた．この推移性は真の親変数集合を所与とした二変数の条件付き独立性から，その各変数と他変数との条件付き独立性を保証する．しかし，RAI アルゴリズムでは事前に真の親変数集合を推定できる保証がないため，この推移性は厳密には成り立たない．そこで，本論文では，条件付き独立の所与とする変数集合を真の親変数集合から任意の変数集合に一般化した推移性を証明する．さらに，より大規模な構造学習のために推移性が連鎖的に成り立つことを証明し，推移性の連鎖を RAI アルゴリズムに組み込んだ新しいアルゴリズムを提案する．推移性の連鎖を用いることにより，少なくとも一つの条件付き独立が保証される二組のエッジの CI テストを優先して実施し続けることができる．提案手法は，推移性を連鎖的に用いることで，エッジ削除および RAI アルゴリズムにおけるグラフ分割を加速させる．また，信頼性の高い低次の CI テストにおいて，推移性を連鎖的に用いたエッジ削除を学習の早期に行うことで，信頼性の低い高次の CI テストを減らすことができる．提案手法は従来手法よりも (1) CI テスト数および計算時間を削減し，(2) 信頼性の低い CI テストを削減することで学習の精度を向上させる．大規模ネットワークを用いた実験により，提案手法は従来手法では学習できない規模の構造学習を実現することを示す．

2024 年度 情報数理工学 (MI) プログラム
修士論文

非制約型条件付き独立性の推移性を用いた大規模ベイジアンネットワーク構造学習

2025 年 3 月 3 日

電気通信大学 情報数理工学プログラム
学籍番号 2331039

加藤 弘也

主任指導教員 植野 真臣

副指導教員 岡本 吉央

目次

1 まえがき 2

2 ベイジアンネットワーク構造学習 4
- 2.1 ベイジアンネットワーク 4
- 2.2 厳密解探索アプローチ 4
- 2.3 制約ベースアプローチ 5

3 推移性を用いた RAI アルゴリズム 7
- 3.1 RAI アルゴリズム 7
- 3.2 推移性を用いた RAI アルゴリズム 8

4 条件付き集合に制約のない推移性と推移性連鎖による BN 学習法の提案 11
- 4.1 条件付き集合に制約のない推移性 11
- 4.2 推移性の連鎖を用いた提案アルゴリズム 12

5 大規模ネットワークによる評価実験 16

6 むすび 23

付録 A 定理 4.1 の証明 28

付録 B 定理 4.2 の証明 29

付録 C 定理 4.3 の証明 30

1　まえがき

ベイジアンネットワーク (Bayesian Network：BN) は離散確率変数をノードで表し, ノード間の依存関係を非循環有向グラフ (Directed Acyclic Graph：DAG) で表現する確率的グラフィカルモデルである. BN は, 確率構造に DAG を仮定することで, 同時確率分布を条件付き確率の積に分解する.

BN の DAG 構造はデータから推定する必要があり, この問題を BN の構造学習と呼ぶ. 近年では, 漸近一致性を有し, 計算効率の高い構造学習法である, Bayes factor を用いた制約ベースアプローチが提案されている [1, 2, 3]. このアプローチでは, 完全無向グラフに, Bayes factor を用いた二ノード間の条件付き独立性検定 (Conditional Independence test：CI テスト) を適用して学習される無向グラフに対し, オリエンテーションルール [4] によるエッジの方向付けを行うことで, 構造を学習する.

Natori ら [2] は, 制約ベースアプローチで最も効率的と知られている RAI アルゴリズム [5] に Bayes factor を用いた CI テストを適用する手法を提案した. RAI アルゴリズムは構造全体を部分構造に分割することで, CI テスト数を削減する. 特に, 分割により信頼性の低い高次の CI テスト数を削減できるため, このアルゴリズムは構造を高速かつ高精度に学習できる. Natori らの手法は, 1000 ノード程度の構造学習を高精度に実現した. また, 本田ら [6] は BN の条件付き独立性において推移性が成り立つことを証明し, Bayes factor を用いた RAI アルゴリズム [2] に組み込んだ. 彼らの証明した推移性は二変数 X と Y が構造 G における X の親変数集合 $\mathbf{Pa}(X, G)$ を所与として条件付き独立ならば, ある二つの変数対のうち少なくとも一つは $\mathbf{Pa}(X, G)$ を所与として条件付き独立となる性質である. この推移性を組み込んだ彼らの手法は, 少なくとも一つの条件付き独立が保証される二組のエッジの CI テストを優先して実施できるため, CI テスト数を大幅に削減した. しかし, RAI アルゴリズムでは, 二変数 X, Y 間の CI テストを行う場合, 事前に X の真の親変数集合 $\mathbf{Pa}(X, G)$ を正確に知ることはできず, 二変数の条件付き独立の所与とする変数集合が $\mathbf{Pa}(X, G)$ である保証はない. しかし, 本田ら [6] は RAI における学習途中の条件付き独立の所与とする変数集合が常に $\mathbf{Pa}(X, G)$ であると仮定しており, 彼らの証明した BN の推移性は厳密には成り立たない.

本論文では, 以上の問題を解決するために, 二変数の条件付き独立の所与とする変数集合に制約を加えないで BN の推移性を証明する. 具体的には, 二変数 X と Y が任意の変数集合 $\mathbf{Z}$ $(X, Y \notin \mathbf{Z})$ を所与として条件付き独立ならば, ある二つの変数対のうち少なくとも一つは $\mathbf{Z}$ を所与として条件付き独立となる性質を証明する. 本田らの推移性は二変数

の条件付き独立の所与とする変数集合が真の親変数集合 $\mathbf{Pa}(X, G)$ の場合のみで成り立つ性質だったのに対し, 本論で証明する推移性は, 条件付き独立の所与とする変数集合が任意の変数集合 $\mathbf{Z}$ に対して成り立つ. さらに, 新たに証明された推移性を用いた RAI アルゴリズムを提案する.

本田ら [6] の証明した推移性では, 二変数の条件付き独立の所与とする変数集合が真の親変数集合 $\mathbf{Pa}(X, G)$ の場合のみに成り立つという制約があり, 推移性を連鎖的に用いることができなかった. 本論文では, その制約のない条件で新たに導出された推移性が連鎖的に成り立つことを証明する. この定理に基づき, Bayes factor を用いた RAI アルゴリズムにおいて推移性の連鎖を用いる新しいアルゴリズムを提案する. 推移性の連鎖を用いることにより, 少なくとも一つの条件付き独立が保証される二組のエッジの CI テストを優先して実施し続けることができる. 提案手法は, 推移性を連鎖的に用いることで, エッジ削除および RAI アルゴリズムにおけるグラフ分割を加速させる. また, 信頼性の高い低次の CI テストにおいて, 推移性を連鎖的に用いたエッジ削除を学習の早期に行うことで, 信頼性の低い高次の CI テストを減らすことができる. このアルゴリズムの利点として, 以下が挙げられる. (1) エッジ削除が加速することで CI テスト数が従来手法に比べて削減されるだけでなく, RAI アルゴリズムによるグラフ分割も加速し, 学習に要する計算時間が削減される. (2) 従来手法に比べて信頼性の低い高次の CI テスト数が減ることで, 学習の精度が向上する.

ランダムに生成した大規模ネットワークを用いたシミュレーション実験により, 提案手法は構造学習の精度を向上させつつ, CI テスト数および計算時間を従来手法より削減することを示す. さらに, 提案手法は従来手法では学習できない 14000 変数の構造学習を実現することを示す.

2　ベイジアンネットワーク構造学習

2.1　ベイジアンネットワーク

ベイジアンネットワーク (以降, BN) は離散確率変数をノードとし, ノード間の依存関係をエッジで表す非循環有向グラフ (以降, DAG) G と, 各ノードの親ノード集合を所与とした条件付き確率パラメータ集合で表現される.

今, G は離散確率変数集合 $\mathbf{V} = \{X_1, \ldots, X_n\}$ の各変数をノードとしてもつとする. また, 各変数 X_i は r_i 個の状態集合 $\{1, \ldots, r_i\}$ から一つの値を取るとし, X_i が状態 k を取るとき, $X_i = k$ と書く. このとき, BN では, 同時確率分布を条件付き確率の積に分解して以下のように表せる.

$$P(X_1, \ldots, X_n \mid G) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}(X_i, G), G). \tag{1}$$

ここで, $\mathbf{Pa}(X_i, G)$ は X_i の G における親ノード集合を表す.

BN は二ノードを結ぶ道をブロックすることで条件付き独立性を表現する. 以下では, ブロックの説明のために道と三つの結合を定義する. 二ノード X, Y を結ぶ道とは, X から Y までのエッジの方向を考慮しない隣接ノードの連なりである. また, 道上で三ノード X, Y, Z が $X \to Z \to Y, X \leftarrow Z \to Y, X \to Z \leftarrow Y$ と結合することをそれぞれ, ノード Z で逐次結合, 分岐結合, 合流結合すると呼ぶ. これらを用いてブロックは以下のように定義される [7].

定義 2.1. 二ノード X, Y を結ぶ道 p が以下のいずれかの条件を満たすとき, 道 p はノード集合 $\mathbf{Z}$ でブロックされる.

1. 逐次結合または分岐結合をするノード $Z \in \mathbf{Z}$ が p 上に存在する.
2. 合流結合をするノード $W \notin \mathbf{Z}$ が p 上に存在し, W の子孫は $\mathbf{Z}$ に属さない.

二ノード X, Y を結ぶ全ての道がノード集合 $\mathbf{Z}$ でブロックされるとき, BN の構造 G における X, Y は $\mathbf{Z}$ を所与として条件付き独立であり, $X \perp Y \mid \mathbf{Z}$ と表される.

2.2　厳密解探索アプローチ

BN の構造 G はデータから推定する必要があり, この問題を BN の構造学習という. BN の構造学習では, 全ての候補構造の中から周辺尤度を最大化する構造を探索する. 今,

$\mathbf{V}$ に対する N 個のデータを $\mathbf{D} = \{D_1, \ldots, D_N\}$ とすると, 周辺尤度は次式で表される [8].

$$P(\mathbf{D} \mid G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}. \tag{2}$$

ここで, q_i は $\mathbf{Pa}(X_i, G)$ の取りうるパターン数であり, $q_i = \prod_{l:\, X_l \in \mathbf{Pa}(X_i,G)} r_l$ で定義される. N_{ijk} は $X_i = k$ かつ $\mathbf{Pa}(X_i, G)$ が j 番目のパターンを取る頻度を表す. また, α_{ijk} はディリクレ事前分布のハイパーパラメータを表し, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}, \alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$ である. 近年では, $\alpha_{ijk} = \alpha/(r_i q_i)$ とした, Bayesian Dirichlet equivalent uniform (BDeu) が一般的に用いられる [8, 9]. ここで, α は Equivalent Sample Size (ESS) と呼ばれる事前知識の重みを示す擬似サンプルのサイズである.

一般にこの BN 構造学習法は, 厳密解探索アプローチと呼ばれる. このアプローチによる構造学習法は NP 困難であり [10], ノード数の増加に伴い, 膨大な計算コストが必要になる. 効率的に構造を探索するために, 動的計画法 [11], A^* 探索 [12], 整数計画法 [13], 制約プログラミング [14] などの構造学習アルゴリズムが提案されている. しかし, 最先端手法でも 60 ノード程度の構造学習が限界である.

近年では, 厳密解探索アプローチを用いてより大規模な構造を学習するために, 強い制約を用いた様々な手法が提案されている. Trösser ら [14] は, 効率的に構造の非循環性を確認する手法を組み込んだ制約プログラミングによる構造学習法 (ELSA) を提案した. 彼女らは, その手法において親変数数を制限することで大規模な構造学習の実現を報告している [14]. Scanagatta らは, ordering-based search (OBS) [15] を元にした, 変数順序によって各変数の親変数集合を制限する ASOBS と呼ばれる構造学習法を提案し, 1500 変数程度の構造学習の実現を報告している [16]. これらの手法は計算コストを削減し, 大規模な構造を学習できるが, 強い制約のために学習の精度が落ち, 漸近一致性を持たないという問題がある.

2.3 制約ベースアプローチ

一方, 厳密解探索アプローチに比べて計算コストを大幅に削減する, 制約ベースアプローチと呼ばれる構造学習法が提案されている [2, 3, 5, 17, 18]. このアプローチでは, (1) 完全無向グラフを生成し, (2) 生成された完全無向グラフに対し条件付き独立性検定 (以降, CI テスト) によるエッジの削除を行い, (3) (2) で得られた無向グラフに対してオリエンテーションルール [7] を用いて方向付けを行う. 制約ベースアプローチによる構造学習法の精度は CI テストの精度に依存し, その計算時間は CI テストの数に依存する. また,

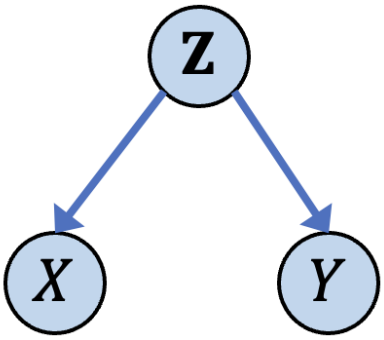

図 1: 独立モデル G_1

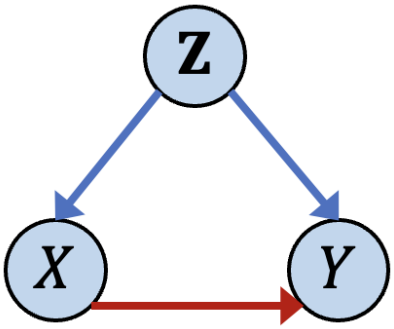

図 2: 従属モデル G_2

条件付き独立の所与とするノード集合の要素数の大きい高次の CI テストでは, 多くの計算コストがかかるだけでなく, 精度が著しく低下することが知られている.

Steck らは, 漸近一致性を有する Bayes factor を用いた CI テストを提案した [19]. 例として, X と Y 間について条件付き独立の所与とするノード集合を $\mathbf{Z}$ としたときの独立なモデルを G_1, 従属なモデルを G_2 とし, それぞれ図 1, 2 に示す. このときの Bayes factor を $\mathrm{BF}(X, Y \mid \mathbf{Z})$ とすると, 対数 Bayes factor は,

$$\log \mathrm{BF}(X, Y \mid \mathbf{Z}) = \log \frac{P(\mathbf{D} \mid G_1)}{P(\mathbf{D} \mid G_2)} \tag{3}$$

と表される. Bayes factor を用いた CI テストでは, 対数 Bayes factor が 0 より大きい場合, 二ノード間のエッジを削除する.

Natori ら [2] は, 制約ベースアプローチの一つである RAI アルゴリズム [5] に Bayes factor を用いた CI テストを適用する手法を提案した. 彼らは, これまでに達成できなかった漸近一致性を有して 1000 変数を超える大規模な構造学習を実現した. また, Cui ら [3] は, サンプルサイズが小さい場合にも頑健な学習を可能にする Bayes factor を用いた CI テストを提案し, RAI アルゴリズムに適用している. 彼女らの手法は, サンプルサイズが小さい場合でも高精度な学習を実現した.

3　推移性を用いた RAI アルゴリズム

3.1　RAI アルゴリズム

RAI アルゴリズム [5] は, 制約ベースアプローチの一つである PC アルゴリズム [17] を改良したもので, 効率的かつ高精度な構造学習アルゴリズムとして知られている. PC アルゴリズムでは, $n-2$ 個のノードを条件付き独立の所与とする高次の CI テストまで繰り返す. しかし, 高次の CI テストは, 低次の CI テストに比べて精度が非常に低く, 構造学習の精度を著しく悪化させる可能性がある. さらに, 高次の CI テストは多くの計算時間を必要とする. 高次の CI テスト数を削減するために, RAI アルゴリズムは, 各次数の CI テスト後に方向付けを行い, その結果を用いて全体構造を部分構造に分割する処理を繰り返す. この分割を説明するために, まず用語の定義を行う. 今, 有向エッジと無向エッジを併せ持つグラフ構造を $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ と表し, $\mathbf{V}, \mathbf{E}$ はそれぞれ $\mathcal{G}$ のノード集合, エッジ集合を表すとする. また, $\mathbf{Adj}(X, \mathcal{G})$ はグラフ $\mathcal{G}$ におけるノード X の隣接ノード集合を表し, $\mathbf{Ch}(X, \mathcal{G})$ はグラフ $\mathcal{G}$ におけるノード X の子ノード集合を表すとする. このとき, $\mathbf{Pa}_p(X, \mathcal{G})$ を $\mathbf{Adj}(X, \mathcal{G}) \setminus \mathbf{Ch}(X, \mathcal{G})$ と定義する. $\mathcal{G}$ の部分グラフ $\mathcal{G}' = (\mathbf{V}', \mathbf{E}')$ が存在するとき, RAI アルゴリズムは構造全体を以下で定義される外生因果からなる構造および自律的部分構造に分割する処理を繰り返す.

定義 3.1 (外生因果)**.** $Y \in \mathbf{V} \setminus \mathbf{V}'$ で, $\forall X \in \mathbf{V}'$ に対し, $Y \in \mathbf{Pa}_p(X, \mathcal{G})$ もしくは $Y \notin \mathbf{Adj}(X, \mathcal{G})$ が成り立つとき, Y は $\mathcal{G}'$ の外生因果である.

定義 3.2 (自律的部分構造)**.** $\mathbf{V}_{ex}$ を $\mathcal{G}'$ の外生因果からなるノード集合とする. $\forall X \in \mathbf{V}'$ に対し, $\mathbf{Pa}_p(X, \mathcal{G}) \subset \mathbf{V}' \cup \mathbf{V}_{ex}$ が成り立つとき, $\mathcal{G}'$ は $\mathbf{V}_{ex}$ を所与として自律的部分構造である.

CI テストに Bayes factor を用いた RAI アルゴリズムの疑似コードを Algorithm1 に示す. Algorithm1 では, 完全無向グラフ $\mathcal{G}_{uc}$ とデータ $\mathbf{D}$ を入力として, 関数 RAI を CI テストの次数を増やしながら再帰的に実行することで構造を学習する. $\boldsymbol{\mathcal{G}}$ をグラフ集合とすると, Algorithm1 中の $\mathbf{Pa}(X, \boldsymbol{\mathcal{G}})$ は $\bigcup_{\mathcal{G} \in \boldsymbol{\mathcal{G}}} \mathbf{Pa}(X, \mathcal{G})$ と定義される. また, $\mathbf{V}[i]$ はノード集合 $\mathbf{V}$ の i 番目の要素を表し, $\boldsymbol{\mathcal{G}}[i]$ はグラフ集合 $\boldsymbol{\mathcal{G}}$ の i 番目の要素を表すとする. 関数 RAI では, (1) 入力されたグラフを $\mathcal{G}_s$ として, 各次数の Bayes factor を用いた CI テストを実施し, ノード X, Y が条件付き独立だと判定された場合, X, Y 間のエッジを削除する. (8 行目から 24 行目) (2) (1) で得られた構造に対し, オリエンテーションルールによる方

向付けを行う (17, 25 行目). (3) 方向付けの結果を用いて $\mathcal{G}_s$ から自律的部分構造を分離する (26 行目から 36 行目). (4) $\mathcal{G}_s$ から外生因果を構成するノード集合を分離する (37 行目から 44 行目). (5) (4) で得られた外生因果からなる構造を引数にして, 再帰的に RAI アルゴリズムを呼び出す (45 行目から 47 行目). (6) (3) で得られた自律的部分構造を引数にして, 再帰的に RAI アルゴリズムを呼び出す (49 行目).

Natori ら [2] はこの Bayes factor を用いた RAI アルゴリズムを用いて, それまでの制約ベースアプローチの中で最も高精度な大規模構造学習を実現した.

3.2 推移性を用いた RAI アルゴリズム

本田ら [6] は, BN において, ある二変数が条件付き独立ならばその各変数と他変数との条件付き独立性の少なくとも一つが成り立つ推移性を証明し, Bayes factor を用いた RAI アルゴリズム [2] に適用した. 彼らの証明した推移性は以下の通りである [6].

定理 3.1. $G = (\mathbf{V}, \mathbf{E})$ を DAG とし, X, $Y \in \mathbf{V}$ で, Y は X の非子孫とする. このとき, $A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Pa}(X, G) \cup \mathbf{W})$ とすると, 以下が成り立つ.

$$X \perp Y \mid \mathbf{Pa}(X, G) \Rightarrow X \perp A \mid \mathbf{Pa}(X, G) \ or \ A \perp Y \mid \mathbf{Pa}(X, G). \tag{4}$$

ここで, $\mathbf{W}$ は X の子孫であり, X と Y が合流結合するノードとその子孫からなるノード集合を表す.

証明は本田ら [6] を参照してほしい. 定理 3.1 から, X と Y が $\mathbf{Pa}(X, G)$ を所与として条件付き独立であるとき, 条件を満たす A について X と A または A と Y が $\mathbf{Pa}(X, G)$ を所与として条件付き独立になることが保証される. したがって, Algorithm1 でエッジ削除を行う際に, 定理 3.1 の推移性によるエッジ削除法を組み入れることで, 少なくとも一つの条件付き独立が保証される二組のエッジの CI テストを優先して実施することができる. 定理 3.1 の推移性によるエッジ削除法の概要は以下の通りである. (1) Algorithm1 でノード X, Y が条件付き独立だと判定された場合, 定理 3.1 に基づき, 式 (4) を満たすノード A を探索する. (2) (1) でノード A が存在した場合, 定理 3.1 に基づき, 少なくとも一つの条件付き独立が保証される $\mathbf{Pa}(X, \mathcal{G})$ を所与とした, X と A, A と Y の CI テストを行う.

彼らの手法は, 少なくとも一つの条件付き独立が保証される二組のエッジの CI テストを優先して実施することができるため, CI テスト数を削減し, 3500 変数の構造学習を達成した.

しかし, RAI アルゴリズムでは, 二ノード X, Y 間の CI テストを行うとき, 事前に X

Algorithm 1 The RAI algorithm using Bayes factor

function MAIN($\mathcal{G}_{uc}, \mathbf{D}$)
  $\mathcal{G}_{uc} = (\mathbf{V}_{uc}, \mathbf{E}_{uc})$: 完全無向グラフ
  $\mathbf{D}$: データ
  return RAI (0, $\mathcal{G}_{uc}$, $\emptyset$, $\mathcal{G}_{uc}$, $\mathbf{D}$)
end function

function RAI(n_z, $\mathcal{G}_s$, $\boldsymbol{\mathcal{G}}_{ex}$, $\mathcal{G}_{all}$, $\mathbf{D}$)
  n_z: CI テストの次数
  $\mathcal{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$: 入力グラフ
  $\boldsymbol{\mathcal{G}}_{ex}$: 外生因果からなるグラフの集合
  $\mathcal{G}_{all} = (\mathbf{V}_{all}, \mathbf{E}_{all})$: CI テストと方向付けによって得られる出力グラフ
  if 全ての $V \in \mathbf{V}_s$ について $|\mathbf{Pa}_p(V, \mathcal{G}_{all})| < n_z + 1$ **then**
    return $\mathcal{G}_{all}$
  end if
  // CI テストによるエッジの削除
  for $\mathcal{G}_{ex} = (\mathbf{V}_{ex}, \mathbf{E}_{ex}) \in \boldsymbol{\mathcal{G}}_{ex}$ **do**
    for $X \in \mathbf{V}_s$, $Y \in \mathbf{V}_{ex}$ **do**
      for $\mathbf{Z} \subseteq \mathbf{Pa}_p(X, \mathcal{G}_s) \cup \mathbf{Pa}(X, \boldsymbol{\mathcal{G}}_{ex}) \setminus \{Y\}$ **do**
        if $|\mathbf{Z}| = n_z$ かつ $\log \mathrm{BF}(X, Y \mid \mathbf{Z}) > 0$ **then**
          $\mathbf{E}_{all} \leftarrow \mathbf{E}_{all} \setminus \{E_{XY}\}$ ▷ E_{XY}: XY 間のエッジ
        end if
      end for
    end for
  end for
  オリエンテーションルールを用いて $\mathbf{E}_{all}$ を方向付け
  for $X \in \mathbf{V}_s$, $Y \in \mathbf{V}_s$ **do**
    for $\mathbf{Z} \subseteq \mathbf{Pa}_p(X, \mathcal{G}_s) \cup \mathbf{Pa}(X, \boldsymbol{\mathcal{G}}_{ex}) \setminus \{Y\}$ **do**
      if $|\mathbf{Z}| = n_z$ かつ $\log \mathrm{BF}(X, Y \mid \mathbf{Z}) > 0$ **then**
        $\mathbf{E}_{all} \leftarrow \mathbf{E}_{all} \setminus \{E_{XY}\}, \mathbf{E}_s \leftarrow \mathbf{E}_s \setminus \{E_{XY}\}$
      end if
    end for
  end for
  オリエンテーションルールを用いて $\mathbf{E}_{all}$, $\mathbf{E}_s$ を方向付け
  // $\mathcal{G}_s$ から自律的部分構造を分離
  $\mathbf{E}_U \leftarrow \mathbf{E}_s$ の無向エッジ集合
  $\mathbf{V}_c \leftarrow \mathbf{V}_s$ の子ノードを持たないノード集合
  $\mathbf{V}_p \leftarrow \mathbf{V}_s \setminus \mathbf{V}_c$
  for $k = 1$ *to* $|\mathbf{V}_c|$ **do**
    if $\mathbf{V}_c[k]$ が $\mathbf{E}_U$ の要素を用いて $\mathbf{V}_p$ のいずれかの要素に到達可能 **then**
      $\mathbf{V}_c \leftarrow \mathbf{V}_c \setminus \mathbf{V}_c[k]$
    end if
  end for
  $\mathbf{E}_c \leftarrow \mathbf{E}_U$ において $\mathbf{V}_c$ の要素を頂点として持つエッジ集合
  $\mathbf{E}_s \leftarrow \mathbf{E}_s \setminus \mathbf{E}_c$
  $\mathbf{V}_s \leftarrow \mathbf{V}_s \setminus \mathbf{V}_c$
  //$\mathcal{G}_s$ から外生因果を分離
  $\boldsymbol{\mathcal{G}}_e \leftarrow \phi$
  for $V \in \mathbf{V}_s$ **do**
    $\mathbf{V}_e \leftarrow \{V\} \cup (V$ から到達可能な $\mathcal{G}_s$ のノード集合$)$
    $\mathbf{E}_e \leftarrow \mathbf{E}_s$ において $\mathbf{V}_e$ の要素を頂点として持つエッジ集合
    $\boldsymbol{\mathcal{G}}_e \leftarrow \boldsymbol{\mathcal{G}}_e \cup \{(\boldsymbol{\mathcal{G}}_e, \mathbf{E}_e)\}$
    $\mathbf{V}_s \leftarrow \mathbf{V}_s \setminus \mathbf{V}_e$
    $\mathbf{E}_s \leftarrow \mathbf{E}_s \setminus \mathbf{E}_e$
  end for
  // 再帰的に関数 RAI を呼び出す
  for $i = 1$ *to* $|\boldsymbol{\mathcal{G}}_e|$ **do**
    $G_{all} \leftarrow \mathrm{RAI}(n_z + 1, \boldsymbol{\mathcal{G}}_e[i], \boldsymbol{\mathcal{G}}_{ex}, \mathcal{G}_{all}, \mathbf{D})$
  end for
  $\boldsymbol{\mathcal{G}}_{ex} \leftarrow \boldsymbol{\mathcal{G}}_{ex} \cup \boldsymbol{\mathcal{G}}_e$
  return $\mathrm{RAI}(n_z + 1, (\mathbf{V}_c, \mathbf{E}_c), \boldsymbol{\mathcal{G}}_{ex}, \mathcal{G}_{all}, \mathbf{D})$
end function

の真の親ノード集合 $\mathbf{Pa}(X, G)$ を推定できている保証はない. しかし, 本田ら [6] は, RAI アルゴリズムにおける学習途中の条件付き独立の所与となる変数集合が常に $\mathbf{Pa}(X, G)$ であると仮定しており, 彼らの証明した定理 3.1 は厳密には適用できない. すなわち, 定理 3.1 の推移性によるエッジ削除法を用いた RAI アルゴリズムは, 条件付き独立の所与とする変数集合が $\mathbf{Pa}(X, G)$ と異なる場合, 優先して行う X と A, A と Y の少なくとも一つが条件付き独立になる保証がない.

4 条件付き集合に制約のない推移性と推移性連鎖による BN 学習法の提案

本論文では, 条件付き独立の所与とする変数集合に制約のない新しい推移性を証明し, それを用いた RAI アルゴリズムを提案する. また, 更なる大規模化および高精度化のために, その推移性の連鎖を用いる新しいアルゴリズムも提案する. 推移性の連鎖を用いることで, 従来手法に比べて計算時間の削減および精度向上が期待される.

4.1 条件付き集合に制約のない推移性

前章で述べたように, 本田ら [6] の提案したエッジ削除法では条件付き独立の所与とする変数集合が $\mathbf{Pa}(X, G)$ と異なる場合, 優先して行う CI テストの少なくとも一つが条件付き独立になる保証がない. そこで, 本論文では, 二変数の条件付き独立の所与とする変数集合を一般化した以下の定理を証明する.

定理 4.1. *$G = (\mathbf{V}, \mathbf{E})$ を DAG とし, X, $Y \in \mathbf{V}$ とする. このとき, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}, \forall A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Z} \cup \mathbf{W})$ について, 以下が成り立つ.*

$$X \perp Y \mid \mathbf{Z} \Rightarrow X \perp A \mid \mathbf{Z} \ or \ A \perp Y \mid \mathbf{Z}. \tag{5}$$

ここで, $\mathbf{W}$ は X と Y が合流結合するノードとその子孫からなるノード集合を表す.

定理 4.1 の証明は付録に記した. この定理 4.1 は, 二変数の条件付き独立の所与とする変数集合が真の親ノード集合 $\mathbf{Pa}(X, G)$ の場合のみで成り立つ定理 3.1 とは異なり, 任意のノード集合 $\mathbf{Z}$ に対して成り立つ. したがって, 条件付き独立の所与とする変数集合が真の親ノード集合である保証のない RAI アルゴリズムに組み込むことが可能である.

この定理 4.1 の推移性を利用したエッジ削除法を Algorithm2 に示す. Algorithm2 中の関数 TRANSITIVE_CUT は Algorithm1 における学習途中のグラフ $\mathcal{G}_{all} = (\mathbf{V}_{all}, \mathbf{E}_{all})$ と $\mathcal{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$, $X \perp Y \mid \mathbf{Z}$ となる二ノード X, Y とノード集合 $\mathbf{Z}$, データ $\mathbf{D}$ を入力とし, 推移性に基づくエッジの削除後の $\mathcal{G}_{all}, \mathcal{G}_s$ を出力する. 関数 TRANSITIVE_CUT の概略は次の通りである. (1) 定理 4.1 の A にあたるノードを探索する (2 行目). (2) 定理 4.1 に基づき, ノード集合 $\mathbf{Z}$ を所与として, X と A, A と Y の CI テストを行う (3 行目から 16 行目). このエッジ削除法を Algorithm1 の 12 行目と 21 行目の後に呼び出す. ここで, CI テストの次数が 0 のとき, 推移性が成り立つかによらず, 全てのノードの組に対して CI テストを行う必要があるため, このエッジ削除法を CI テストの次数が 1 以上のとき

Algorithm 2 Edge cutting with transitivity

function TRANSITIVE_CUT($\mathcal{G}_{all}$, $\mathcal{G}_s$, X, Y, $\mathbf{Z}$, $\mathbf{D}$)
  $\mathcal{G}_{all} = (\mathbf{V}_{all}, \mathbf{E}_{all})$: 全体グラフ
  $\mathcal{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$: $\mathcal{G}_{all}$ の部分構造
  $X, Y, \mathbf{Z}$: $\log \mathrm{BF}(X, Y \mid \mathbf{Z}) > 0$ となる二ノード X, Y とノード集合 $\mathbf{Z}$
    $\mathbf{A} \leftarrow \mathbf{Adj}(X,\mathcal{G}_{all}) \cap \mathbf{Adj}(Y,\mathcal{G}_{all}) \setminus (\mathbf{Ch}(X,\mathcal{G}_{all}) \cap \mathbf{Ch}(Y,\mathcal{G}_{all}) \cup \mathbf{Z})$
    for $A \in \mathbf{A}$ **do**
      if $\log \mathrm{BF}(X, A \mid \mathbf{Z}) > 0$ **then**
        $\mathbf{E}_{all}$ から $\{E_{XA}\}$ を除く ▷ E_{XA}: 二ノード XA 間のエッジ
        if $X \in \mathbf{V}_s$ and $A \in \mathbf{V}_s$ **then**
          $\mathbf{E}_s$ から $\{E_{XA}\}$ を除く
        end if
      end if
      if $\log \mathrm{BF}(A, Y \mid \mathbf{Z}) > 0$ **then**
        $\mathbf{E}_{all}$ から $\{E_{AY}\}$ を除く
        if $A \in \mathbf{V}_s$ and $Y \in \mathbf{V}_s$ **then**
          $\mathbf{E}_s$ から $\{E_{AY}\}$ を除く
        end if
      end if
    end for
    return $(\mathcal{G}_{all}, \mathcal{G}_s)$
end function

にのみ呼び出す. Algorithm2 は, 事前に真の親ノード集合 $\mathbf{Pa}(X, G)$ を知ることができなくても, 定理 4.1 によって, X と A, A と Y の少なくとも一つが条件付き独立になることが保証されるため, CI テスト数を削減できる. 制約ベースアプローチの計算時間は CI テスト数に依存するため, Algorithm2 は計算時間を削減する.

4.2 推移性の連鎖を用いた提案アルゴリズム

本節では, 前節で証明した推移性を連鎖的に用いる, より効率的なアルゴリズムを提案する.

まず, 推移性によって検出した条件付き独立性から新たな推移性を導くことができる以

下の定理を証明する.

定理 4.2. $G = (\mathbf{V}, \mathbf{E})$ を DAG とし, $X, Y \in \mathbf{V}$, $\mathbf{Z} \subseteq \mathbf{V} \backslash \{X, Y\}$ について, $X \perp Y \mid \mathbf{Z}$ が成り立つとする. このとき, $A \in \mathbf{V} \backslash (\{X, Y\} \cup \mathbf{Z} \cup \mathbf{W})$ について, $X \perp A \mid \mathbf{Z}$ *or* $A \perp Y \mid \mathbf{Z}$ が成り立つ. このとき, $\forall B \in \mathbf{V} \setminus (\{X, A\} \cup \mathbf{Z} \cup \mathbf{W}')$ について, 以下が成り立つ.

$$X \perp A \mid \mathbf{Z} \Rightarrow X \perp B \mid \mathbf{Z} \ or \ B \perp A \mid \mathbf{Z}. \tag{6}$$

また, $\forall C \in \mathbf{V} \backslash (\{A, Y\} \cup \mathbf{Z} \cup \mathbf{W}'')$ について, 以下が成り立つ.

$$A \perp Y \mid \mathbf{Z} \Rightarrow A \perp C \mid \mathbf{Z} \ or \ C \perp Y \mid \mathbf{Z}. \tag{7}$$

ここで, $\mathbf{W}$ は X と Y が合流結合するノードとその子孫からなるノード集合を表し, $\mathbf{W}'$ は X と A が合流結合するノードとその子孫からなるノード集合を表す. また, $\mathbf{W}''$ は A と Y が合流結合するノードとその子孫からなるノード集合を表す.

定理 4.2 の証明は付録に記した. この定理 4.2 により, 定理 4.1 で保証できる条件付き独立性である, $X \perp Y \mid \mathbf{Z} \Rightarrow X \perp A \mid \mathbf{Z}$ *or* $A \perp Y \mid \mathbf{Z}$ から, 新たにその各変数と他変数との条件付き独立性の少なくとも一つを保証できる推移性を導ける (式 (6), (7)). さらに, 式 (6), (7) で保証できる条件付き独立性からも同様に推移性を導けるため, 推移性を連続的に用いることができる. したがって, この推移性の連鎖を RAI アルゴリズムに組み込むことで, 従来の制約ベースアプローチより CI テスト数を増加させずに, 新たな条件付き独立性の候補を同定し続けながら学習ができる.

定理 4.2 を利用したエッジ削除法を Algorithm3 に示す. Algorithm3 中の関数 TRANSITIVE_CUT_CHAIN では, より早期に発見した条件付き独立性から導かれる推移性を優先して利用するために幅優先で推移性の連鎖を探索する. ここで, 推移性の連鎖を幅優先ではなく, 深さ優先で探索する手法も考えられる. 深さ優先探索では, より直近に発見した条件付き独立性から導かれる推移性を優先して利用するため, 探索の早期における CI テストが誤って独立と推定してしまうとそれから導かれる CI テスト結果への影響が大きく, 結果として学習される構造の誤差が大きくなってしまう危険性がある.

Algorithm3 中の関数 TRANSITIVE_CUT_CHAIN は Algorithm2 と同様に Algorithm1 における学習途中のグラフ $\mathcal{G}_{all} = (\mathbf{V}_{all}, \mathbf{E}_{all})$ と $\mathcal{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$, $X \perp A \mid \mathbf{Z}$ となる二ノード X, A とノード集合 $\mathbf{Z}$, データ $\mathbf{D}$ を入力とし, 推移性に基づくエッジの削除後の $\mathcal{G}_{all}, \mathcal{G}_s$ を出力する. 関数 TRANSITIVE_CUT_CHAIN の概略は以下の通りである. (1) キュー q を初期化し, 変数対 (X, A) を q に挿入する. (2,3 行目) (2) q が空になるまで以下の (3)〜(7) を繰り返す. (4 行目から 23 行目) (3) q の先頭要素を取り出し, 定理 4.2 の B にあたるノードを探索する (5,6 行目). (4) X と B に対して $\mathbf{Z}$ を所与とした

Algorithm 3 Edge cutting with transitivity recursively

function TRANSITIVE_CUT_CHAIN($\mathcal{G}_{all}$, $\mathcal{G}_s$, X, A, $\mathbf{Z}$, $\mathbf{D}$)
  $\mathcal{G}_{all} = (\mathbf{V}_{all}, \mathbf{E}_{all})$: 全体グラフ
  $\mathcal{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$: $\mathcal{G}_{all}$ の部分構造
  $X, A, \mathbf{Z}$: $\log \mathrm{BF}(X, A \mid \mathbf{Z}) > 0$ となる二ノード X, A とノード集合 $\mathbf{Z}$
    キュー q の初期化
    PUSH(q, (X, A))
    while q が空でない **do**
      $(X, A) = \mathrm{POP}(q)$
      $\mathbf{B} \leftarrow \mathbf{Adj}(X,\mathcal{G}_{all}) \cap \mathbf{Adj}(A,\mathcal{G}_{all}) \setminus (\mathbf{Ch}(X,\mathcal{G}_{all}) \cap \mathbf{Ch}(A,\mathcal{G}_{all}) \cup \mathbf{Z})$
      for $B \in \mathbf{B}$ **do**
        if $\log \mathrm{BF}(X, B \mid \mathbf{Z}) > 0$ **then**
          $\mathbf{E}_{all}$から $\{E_{XB}\}$ を除く ▷ E_{XB}: 二ノード XB 間のエッジ
          if $X \in \mathbf{V}_s$ かつ $B \in \mathbf{V}_s$ **then**
            $\mathbf{E}_s$から $\{E_{XB}\}$ を除く
          end if
          PUSH(q, (X, B))
        end if
        if $\log \mathrm{BF}(B, A \mid \mathbf{Z}) > 0$ **then**
          $\mathbf{E}_{all}$から $\{E_{BA}\}$ を除く
          if $B \in \mathbf{V}_s$ かつ $A \in \mathbf{V}_s$ **then**
            $\mathbf{E}_s$から $\{E_{BA}\}$ を除く
          end if
          PUSH(q, (B, A))
        end if
      end for
    end while
    return $(\mathcal{G}_{all}, \mathcal{G}_s)$
end function

Bayes factor を用いた CI テストを行う (8 行目). (5) (4) において, $X \perp B \mid \mathbf{Z}$ が成り立つならば, X, B 間のエッジを削除し, 変数対 (X, B) を q に挿入する (8 行目から 14 行目). (6) B と A に対して $\mathbf{Z}$ を所与とした Bayes factor を用いた CI テストを行う (15 行目). (7) (6) において, $B \perp A \mid \mathbf{Z}$ が成り立つならば, B, A 間のエッジを削除し, 変数対 (B, A) を q に挿入する (15 行目から 21 行目). このエッジ削除法を Algorithm2 の代わりに Algorithm1 の 12 行目と 21 行目の後に同様に呼び出す.

提案手法は推移性の連鎖を用いることで, 少なくとも一つの条件付き独立性が保証される二組のエッジの CI テストを優先して実施し続けることができるため, 推移性の連鎖を用いない場合に比べて, 学習のより早期に多くの条件付き独立性を検出できる. 学習の早期にエッジを削除するほど CI テスト数は削減され, RAI アルゴリズムにおけるグラフ分割が加速される. したがって, 提案手法は (1)CI テスト数の削減, (2) エッジ削除の加速, (3)RAI アルゴリズムにおけるグラフ分割の加速により, 従来手法では実現できなかった大規模な構造学習の実現が期待される. また, 提案手法は, 学習の早期に信頼性の高い低次の CI テストを用いたエッジ削除を多く実施できるため, 推移性の連鎖を用いない場合に比べて, 学習精度の向上も期待される.

ただし, Algorithm3 で, 定理 4.2 の推移性の連鎖を利用するためには, 二ノードが合流結合するノードとその子孫からなるノード集合を毎回探索する必要がある. この合流結合を厳密に同定することは, そのノード間の全ての道を探索する必要があり, 計算コストが大きい. そこで, 本論文では, その探索空間を削減できる定理を証明する. E_{XY} を二ノード X, Y 間のエッジとすると, 以下の定理 4.3 が成り立つ.

定理 4.3. *Algorithm3* における入力時の全体グラフ $\mathcal{G}_{all} = (\mathbf{V}_{all}, \mathbf{E}_{all})$, 二ノード X, A, ノード集合 $\mathbf{Z}$ に関して, $\mathbf{W}'$ を真の構造において X と A が合流結合するノードとその子孫からなるノード集合とし, $(\mathbf{Adj}(X, \mathcal{G}_{all}) \cap \mathbf{Adj}(A, \mathcal{G}_{all})) \cap \mathbf{W}' = \mathbf{Ch}(X, \mathcal{G}_{all}) \cap \mathbf{Ch}(A, \mathcal{G}_{all})$ と仮定する. このとき, $\forall B \in \mathbf{Adj}(X, \mathcal{G}_{all}) \cap \mathbf{Adj}(A, \mathcal{G}_{all}) \backslash (\mathbf{Ch}(X, \mathcal{G}_{all}) \cap \mathbf{Ch}(A, \mathcal{G}_{all}) \cup \mathbf{Z})$ について以下が成り立つ.
$\mathcal{G}_{all}$ において E_{XB} かつ E_{BA} が存在し, $X \perp B \mid \mathbf{Z}$ *or* $B \perp A \mid \mathbf{Z}$.

定理 4.3 の証明は付録に記した. 定理 4.3 より, ノード B の探索には二ノード X と A の共通隣接ノードのみを探索すればよく, ノード数に対して線形時間で計算できる. ノード C についての探索も同様に線形時間で計算できる. したがって, 推移性の連鎖を利用したとしても, 条件を満たすノード B や C の探索時間は小さく, 推移性の利用による計算時間の削減量の方が多いことが期待される.

5　大規模ネットワークによる評価実験

この章では提案手法の利点を示すために以下の 7 種類の手法で比較実験を行う.

- ELSA [14]
- ASOBS [16]
- PC アルゴリズム (以降, PC と表記) [17]
- RAI アルゴリズム (以降, RAI と表記) [5]
- 定理 4.1 の推移性を用いた本田らの RAI アルゴリズム (以降, RAI-transitivity と表記) [6]
- 推移性の連鎖を深さ優先で探索する RAI アルゴリズム
- 推移性の連鎖を幅優先で探索する RAI アルゴリズム

ELSA[14] は厳密解探索アプローチの一つであり, 制約プログラミングにより周辺尤度を最大にする構造を探索する. ASOBS[16] は変数順序によって 各変数の親変数集合を制限する近似的な手法である. PC[17] は制約ベースアプローチの一つであり, $n-2$ 個のノードを条件付き独立の所与とする高次の CI テストまで繰り返す. RAI[5] は RAI-transitivity と異なり, CI テストを行う変数対の順序を推移性を用いずに, ランダムに選択する. 本論では, 推移性の連鎖を深さ優先で探索する RAI アルゴリズムと推移性の連鎖を幅優先で探索する RAI アルゴリズムを提案手法とする. PC, RAI, RAI-transitivity, 提案手法については, BDeu に基づく Bayes factor (ESS = 1.0 [20, 21, 22]) による CI テストを適用した. ELSA は Trösser らが公開している実装[*1]を, ASOBS については, Scanagatta らが公開している実装[*2] を用いた. また, RAI は, Lerner が公開している実装[*3]を使用した. PC, RAI-transitivity と提案手法は独自に実装した. 以下の実験については, 3.2GHz の 16 コアプロセッサと 128GB のメモリを搭載した計算機で実験を行った.

本章では, 提案手法が従来手法では学習できない規模の構造を学習できることを確認するために, ランダムに生成した大規模ネットワークを用いた実験を行う. ランダムネットワークの生成には, BNGenerator[*4][23, 24] を用いた. 各ランダムネットワークの概要を表 1 に示す.

[*1] https://gkatsi.github.io/#tdk-ijcai21
[*2] https://github.com/mauro-idsia/blip?tab=readme-ov-file
[*3] http://www.ee.bgu.ac.il/~boaz/software.html
[*4] http://sites.poli.usp.br/pmr/ltd/software/bngenerator/

表 1: ランダムネットワークの概要

network	ノード数	エッジ数	最大 親変数数	パラメータ数
random1000	1000	2472	5	16712
random2000	2000	4959	5	33528
random3000	3000	7454	5	50710
random4000	4000	9946	5	67718
random10000	10000	24907	5	168786
random11000	11000	27407	5	186576
random12000	12000	29896	5	203002
random13000	13000	32393	5	219160
random14000	14000	34896	5	237736

実験手順は以下の通りである.

1. 各ランダムネットワークからデータセットを $N = 10,000,\ 1,000,000$ だけランダムに発生させる.
2. 手順 (1) で生成したデータを用いて, 各比較手法により構造学習を行う.

また, 構造学習は, 24 時間の制限時間を設け, 超過する場合は学習を打ち切った.

最初に, 提案手法の推移性の連鎖が学習精度を向上させることを確認するために, Structural Hamming Distance (SHD) の比較を行った. SHD は真の構造には存在するが, 学習過程で削除されたエッジ数と真の構造には存在しないが, 学習で残ったエッジ数, エッジの方向付けの誤りの和によって, 真の構造と学習した構造の距離を表す. SHD が 0 のとき, 真の構造と学習した構造が一致したことを表す. その結果を表 2 に示す. 表 2 における"TO"は 24 時間の制限時間以内に構造学習が完了しなかったことを表し, "MO"はメモリ不足により構造学習が完了しなかったことを表す.

表 2 より, 提案手法は PC を除く全ての制約ベースアプローチ手法, 全てのネットワークで SHD が最も小さいことが確認できる. この結果は推移性の連鎖が学習精度を向上させたことを示している. その理由は, 推移性の連鎖により信頼性の低い高次の CI テスト数が RAI, RAI-transitivity に比べて削減されたためだと考えられる.

また, 表 2 より, PC の SHD は学習を完了した全てのネットワークで提案手法の SHD

表 2: ランダムネットワークにおける SHD

network	ELSA	ASOBS	PC	RAI	RAI -transitivity	提案手法 深さ優先	提案手法 幅優先
$N = 10,000$							
random10000	TO	25588	15472	20777	19745	19421	19416
random11000	TO	27939	TO	22777	21599	21135	21152
random12000	TO	30467	TO	24809	23509	23041	23047
random13000	TO	32468	TO	TO	25596	25134	25143
random14000	TO	MO	TO	TO	TO	26982	26989
$N = 1,000,000$							
random1000	TO	MO	635	3051	2487	2125	2129
random2000	TO	MO	9556	13314	12122	11429	11476
random3000	TO	MO	2020	TO	7739	6739	6763
random4000	TO	MO	TO	TO	TO	8358	8400

よりも小さいことが確認できる. 提案手法は構造の分割により探索空間を削減しているため, 提案手法の探索空間が PC の探索空間の部分集合になっていることが要因だと考えられる. ただ, 表 2 から分かるように, PC はその大きな探索空間のために, $N = 10,000$ では random11000, $N = 1,000,000$ では, random4000 で学習が打ち切られている.

さらに, 表 2 より, 提案手法における深さ優先の SHD と幅優先の SHD を比較すると, ほぼ同等であることが確認できる. 推移性の連鎖を深さ優先的に探索すると, 探索の早期における CI テストで誤って独立と推定した結果から導かれる推移性を優先してしまい, 条件付き独立になる保証のない CI テストを早期に行なってしまう場合がある. その場合, 深さ優先の方が幅優先よりも学習精度が低下する可能性があるが, 今回のネットワークではそのような問題は確認できなかった.

また, 表 2 より, $N = 10,000$ において, ASOBS は全てのネットワークで最も学習精度が低い. この理由は, ASOBS が各変数の親候補変数集合に強い制約を加えているためである.

次に, 推移性の連鎖を用いることで計算コストを削減できるか確認するために, 計算時間の比較を行った. その結果を表 3 に示す. 表 3 より, 提案手法は従来の制約ベースアプローチである, PC, RAI, RAI-transitivity が 24 時間以内に学習できなかった 14000 変

表 3: ランダムネットワークにおける計算時間 (s)

network	ELSA	ASOBS	PC	RAI	RAI -transitivity	提案手法 深さ優先	提案手法 幅優先
$N = 10,000$							
random10000	TO	1132	64361	46612	39529	33263	32778
random11000	TO	1135	TO	52926	46756	38361	34618
random12000	TO	1575	TO	65024	61342	47322	44301
random13000	TO	1823	TO	TO	76716	57010	53434
random14000	TO	MO	TO	TO	TO	70726	64738
$N = 1,000,000$							
random1000	TO	MO	9401	11930	8410	7773	7447
random2000	TO	MO	28685	66245	27437	21766	19787
random3000	TO	MO	68953	TO	49786	39598	39065
random4000	TO	MO	TO	TO	TO	70085	67047

数の構造学習を達成していることが確認できる. また, $N = 10,000,\ 1,000,000$ の全てのランダムネットワークにおいて提案手法は他の制約ベースアプローチに比べて計算時間を削減した. この結果は, 推移性の連鎖を用いることにより, CI テスト数を削減し, エッジ削除と RAI アルゴリズムにおける構造分割が加速したためであると考えられる. また, 表 3 より, 提案手法における深さ優先と幅優先を比較すると, 幅優先の計算時間の方がわずかに短いことが確認できる.

さらに, 表 3 より, $N = 10,000$ における random10000 $\sim$ random13000 では, ASOBS の計算時間が最も短いことが確認できる. この結果は, ASOBS が各変数の親候補変数集合に強い制約を加えているためである. しかし, 親変数集合を制限してしまうと, 表 2 で示されているように構造学習の精度が著しく悪化してしまう. また, $N = 10,000$ の random14000, $N = 1,000,000$ では, ASOBS はメモリ不足により計算が打ち切られている. これは, ASOBS が周辺尤度スコアを効率的に計算するために, 全てのデータをメモリ上に保持しているためだと考えられる. また, ELSA は全てのランダムネットワークで構造学習を完了することができなかった. ELSA は全ての構造の候補から周辺尤度が最大になる構造を探索する厳密解探索アプローチであり, 膨大な計算コストが必要になるためである.

表 4: ランダムネットワークにおける CI テスト数

network	PC	RAI	RAI -transitivity	提案手法 深さ優先	提案手法 幅優先
$N = 10,000$					
random10000	50830458	50548149	50495256	50481813	50471801
random11000	TO	61727823	61447707	61352080	61335195
random12000	TO	72812503	72704479	72683177	72683354
random13000	TO	TO	85228576	85193124	85193285
random14000	TO	TO	TO	98774543	98774549
$N = 1,000,000$					
random1000	719713	861871	680245	643463	643166
random2000	2465852	2979304	2419915	2316886	2318677
random3000	5109797	TO	5094864	4931072	4933928
random4000	TO	TO	TO	8520211	8521028

以下では, 提案手法の計算時間が短くなった要因として考えられる, (1) CI テスト数の削減, (2) エッジ削除の加速, (3) RAI における構造分割の加速を確認する. まず, 提案手法が CI テスト数を削減していることを確認するために, 制約ベースアプローチの CI テスト数を推定した. その結果を表 4 に示す.

表 4 より, 提案手法は全てのランダムネットワークにおいて CI テスト数が最も少ないことが確認できる. この結果は, 提案手法が少なくとも一つの条件付き独立が保証される二組のエッジの CI テストを優先して実施し続けることで, 学習の早期に多くのエッジ削除を行ったためだと考えられる. また, 提案手法における深さ優先と幅優先を比較すると, ほぼ同等の CI テスト数であることが確認できる.

次に, 提案手法のエッジ削除が加速していることを確認するために, RAI, RAI-transitivity, 推移性の連鎖を幅優先で探索する提案手法の時間経過によるエッジ削除数をプロットした. このプロットでは推移性を用いる 1 次の CI テストが始まった時点を時間計測の始点とし, $N = 1,000,000$ における random1000 で計測を行った. そのプロットを図 3 に示す.

図 3 より, 提案手法のエッジ削除が RAI, RAI-transitivity のエッジ削除に比べて速いことが分かる. この結果は, 推移性の連鎖が RAI におけるエッジ削除を加速させたことを

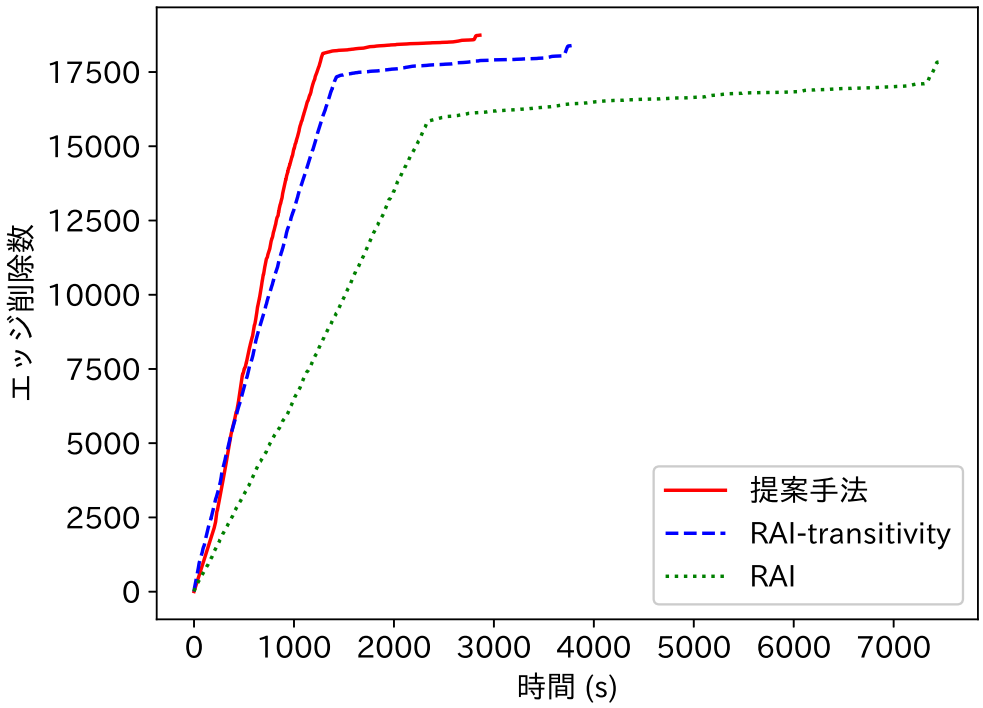

図 3: random1000 におけるエッジ削除数の変遷

示している．特に, 図 3 より, 提案手法は信頼性の高い低次の CI テストを用いる学習の早期にエッジ削除を加速させていることが分かる．

次に, 提案手法が RAI における構造分割を加速させることを確認するために, RAI, RAI-transitivity, 推移性の連鎖を幅優先で探索する提案手法の時間経過による構造の分割数をプロットした． このプロットでは最初の分割時点を時間計測の始点とし, $N = 1,000,000$ における random1000 で計測を行った．その結果を図 4 に示す．

図 4 より, 提案手法は RAI, RAI-transitivity に比べて RAI の構造分割をより早期の段階で行えていることがわかる．この結果は, 推移性の連鎖が RAI の構造分割を加速させ, 結果として計算時間を削減させたことを示している．

最後に, 提案手法が推移性の連鎖により信頼性の低い高次の CI テスト数を RAI, RAI-transitivity に比べて削減したことを確認するために, $N = 1,000,000$ における random1000 の各次数の CI テスト数とエッジ削除数を推定した．その結果を表 5 に示す．

表 5 より, 提案手法は 2 次以上の次数の CI テスト数を大幅に削減できていることが確認できる． さらに, 信頼性の高い低次の CI テストにおいてエッジ削除数が RAI, RAI-transitivity に比べて大きいことが分かる．この理由は, 提案手法が学習の早期に推

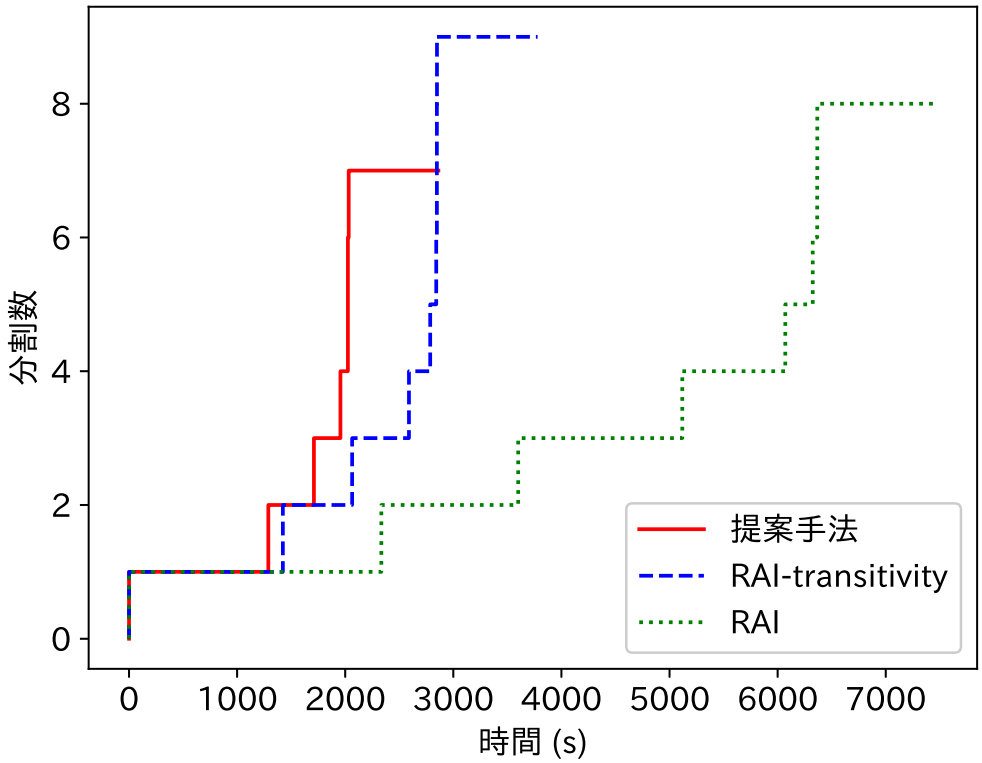

図 4: random1000 における RAI の分割数の変遷

表 5: random1000 における各次数の CI テスト数およびエッジ削除数

					提案手法			
random1000	RAI		RAI-transitivity		深さ優先		幅優先	
order	CI テスト数	エッジ削除数	CI テスト数	エッジ削除数	CI テスト数	エッジ削除数	CI テスト数	エッジ削除数
ord0	499500	477543	499500	477543	499500	477543	499500	477543
ord1	173585	16536	96325	17655	87482	18261	86712	18256
ord2	68941	601	35841	363	25319	259	25538	258
ord3	67380	363	28609	193	18732	129	18918	130
ord4	38626	224	14089	123	8999	63	9060	63
≥ord5	13839	122	5881	59	3431	36	3438	37
total	861871	495389	680245	495936	643463	496291	643166	496287

移性の連鎖を有効的に用いることで, 条件付き独立となる二ノードを効率的に検出できているからだと考えられる.

6 むすび

本論文では, 先行研究で証明されていたベイジアンネットワークの推移性が条件付き独立の所与とする変数集合に真の親変数集合を仮定していることの問題を指摘し, その変数集合を一般化した新たな推移性を証明した. また, 新たに証明された推移性から, 推移性の連鎖を用いる新しいアルゴリズムを提案した.

提案手法は以下の利点をもつ. (1) 推移性の連鎖を用いることで, エッジ削除および RAI アルゴリズムにおけるグラフ分割が加速し, 計算時間を削減する. (2) 信頼性の低い高次の CI テスト数が減ることで, 学習の精度が向上する.

大規模ランダムネットワークによる実験により, 提案手法は従来のアルゴリズムに比べ学習の精度を向上させ, 計算時間を削減できることを示した. また, 従来手法では, 13000 変数程度の構造学習が限界であったが, 提案手法は 14000 変数の構造学習を実現した.

今後の課題として, ベイジアンネットワーク分類器 [25, 26, 27, 28, 29] への応用が挙げられる.

謝辞

本論文を作成するにあたり，指導教員の植野真臣教授から，丁寧かつ熱心なご指導を賜りました. ここに感謝の意を表します．そして，ゼミや日常の議論を通じて多くの示唆や知識を頂いた菅原聖太助教，研究室の先輩・同期・後輩に感謝いたします.

参考文献

[1] Joaquín Abellán, Manuel Gómez-Olmedo, Serafín Moral, et al. Some variations on the PC algorithm. In *International Conference on Probabilistic Graphical Models*, pages 1–8, 2006.

[2] Kazuki Natori, Masaki Uto, and Maomi Ueno. Consistent learning Bayesian networks with thousands of variables. In *Advanced Methodologies for Bayesian Networks (Proceedings of Machine Learning Research)*, volume 73, pages 57–68, 2017.

[3] Cui Zijun, Yin Naiyu, Wang Yuru, and Ji Qiang. Empirical Bayesian Approaches for Robust Constraint-based Causal Discovery under Insufficient Data. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI 2022)*, pages 4850–4856, Vienna, 2022.

[4] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[5] R. Yehezkel and B. Lerner. Bayesian network structure learning by recursive autonomy identification. *Journal of Machine Learning Research*, 10:1527–1570, 2009.

[6] 本田和雅, 名取和樹, 菅原聖太, 磯崎隆司, and 植野真臣. 推移性を利用した大規模ベイジアンネットワーク構造学習. **信学論** *(D)*, 102(12):796–811, Dec 2019.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann, 1988.

[8] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[9] W. Buntine. Theory refinement on Bayesian networks. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, pages 52–60, 1991.

[10] D. M. Chickering. Learning Bayesian networks is NP-Complete. In *Learning from Data: Artificial Intelligence and Statistics*, volume V, pages 121–130. Springer, 1996.

[11] T. Silander and P. Myllymaki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of Uncertainty in Artificial*

Intelligence (UAI), pages 445–452. AUAI Press, 2006.

[12] C. Yuan, B. Malone, and W. Xiaojian. Learning optimal Bayesian networks using A* search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2186–2191, 2011.

[13] J. Cussens. Bayesian network learning with cutting planes. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, pages 153–160. AUAI Press, 2011.

[14] Fulya Trösser, Simon de Givry, and George Katsirelos. Improved acyclicity reasoning for bayesian network structure learning with constraint programming. In *International Joint Conference on Artificial Intelligence (IJCAI 2021)*, pages 4250–4257, 2021.

[15] M. Teyssier and D. Koller. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainly in Artificial Intelligence (UAI 2005)*, pages 584–590, 2005.

[16] M. Scanagatta, C. P de Campos, G. Corani, and M. Zaffalon. Learning Bayesian Networks with Thousands of Variables. In *Neural Information Processing Systems (NIPS 2015)*, pages 1864–1872, 2015.

[17] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT press, 2000.

[18] Ehsan Mokhtarian, Sina Akbari, Fateme Jamshidi, Jalal Etesami, and Negar Kiyavash. Learning bayesian networks in the presence of structural side information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7814–7822, 2022.

[19] H. Steck and T.S. Jaakkola. On the dirichlet prior and Bayesian regularization. In *Neural Information Processing Systems (NIPS 2002)*, pages 697–704. MIT Press, 2002.

[20] Maomi Ueno. Learning likelihood-equivalence Bayesian networks using an empirical Bayesian approach. *Behaviormetrika*, 35(2):115–135, 2007.

[21] Maomi Ueno. Learning networks determined by the ratio of prior and data. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, pages 598–605. AUAI Press, 2010.

[22] Maomi Ueno. Robust learning Bayesian networks for prior belief. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, pages 689–707. AUAI Press, 2011.

[23] Jaime S Ide and Fabio G Cozman. Random generation of Bayesian networks. In

Brazilian symposium on artificial intelligence, pages 366–376. Springer, 2002.

[24] Jaime Shinsuke Ide, Fabio Gagliardi Cozman, and Fabio Tozeto Ramos. Generating random Bayesian networks with constraints on induced width. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, volume 16, pages 323–327, 2004.

[25] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[26] Shouta Sugahara, Masaki Uto, and Maomi Ueno. Exact learning augmented naive Bayes classifier. In *International Conference on Probabilistic Graphical Models*, pages 439–450, 2018.

[27] Shouta Sugahara and Maomi Ueno. Exact learning augmented naive bayes classifier. *Entropy*, 23(12):1703, 2021.

[28] Shouta Sugahara, Wakaba Kishida, Koya Kato, and Maomi Ueno. Recursive autonomy identification-based learning of augmented naive bayes classifiers. In *International Conference on Probabilistic Graphical Models*, pages 265–276. PMLR, 2022.

[29] Shouta Sugahara, Koya Kato, and Maomi Ueno. Learning bayesian network classifiers to minimize class variable parameters. In *38th AAAI Conf. on Artificial Intelligence (AAAI)*, pages 20540–20549, 2024.

付録 A　定理 4.1 の証明

以下の補題付録 A.1 を証明し, ベイジアンネットワークの弱推移性 [7] を示した後で, ベイジアンネットワークの推移性である定理 4.1 を示す.

補題 付録 A.1. $G = (\mathbf{V}, \mathbf{E})$ を *DAG* とし, $X, Y \in \mathbf{V}$ とする. また, $\mathbf{W}$ を X と Y を結ぶ道において合流結合するノードとその子孫からなるノード集合とする. このとき, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$, $\forall A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Z} \cup \mathbf{W})$ について, 以下が成り立つ.

$$X \perp Y \mid \mathbf{Z} \Rightarrow X \perp Y \mid \mathbf{Z} \cup \{A\}. \tag{8}$$

証明

(補題付録 A.1)**.** $X, Y \in \mathbf{V}$, $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ が $X \perp Y \mid \mathbf{Z}$ を満たすと仮定し, $A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Z} \cup \mathbf{W})$ とする. X と Y を結ぶ全ての道は, 内点に A が存在する道 p と存在しない道 q に分けられる. それぞれの道が, $\mathbf{Z} \cup \{A\}$ でブロックされるならば, X と Y を結ぶ全ての道が $\mathbf{Z} \cup \{A\}$ でブロックされる.

(1) **道 p が存在する場合**　$A \notin \mathbf{W}$ であることから, 道 p の内点には A が合流結合せずに存在する. したがって, 道 p は A でブロックされる. また, A が逐次結合または分岐結合で道 p をブロックするため, 道 p は $\mathbf{Z} \cup \{A\}$ でもブロックされる.

(2) **道 q が存在する場合**　$X \perp Y \mid \mathbf{Z}$ が成り立つため, 定義 *2.1* より, 道 q は, *(i)* 内点に $\mathbf{Z}$ に属するノードが合流結合せずに存在する場合と, *(ii)* 内点に合流結合するノード $W \in \mathbf{W} \setminus \mathbf{Z}$ が存在し, $\mathbf{Z}$ は W の子孫を含まない場合, *(iii) (i), (ii)* を同時に満たす場合に分けられる. 以下では, *(i), (ii), (iii)* のそれぞれの場合について道 q が $\mathbf{Z} \cup \{A\}$ でブロックされることを示す.

(i) **の場合**　道 q の内点に, $\mathbf{Z}$ に属するノードが合流結合せずに存在するため, 道 q は $\mathbf{Z}$ でブロックされる. また, $\mathbf{Z}$ に属するノードが逐次結合または分岐結合で道 q をブロックしているため, 道 q は $\mathbf{Z} \cup \{A\}$ でもブロックされる.

(ii) **の場合**　道 q の内点に合流結合するノード $W \in \mathbf{W} \setminus \mathbf{Z}$ が存在し, $W \notin \mathbf{Z}$ である. また, W の子孫は $\mathbf{Z}$ に含まれないため, 道 q は $\mathbf{Z}$ でブロックされる. また, $A \notin \mathbf{W}$ であるため, 道 q は $\mathbf{Z} \cup \{A\}$ でもブロックされる.

(iii) **の場合**　*(i)* が成り立つ場合, $\mathbf{Z}$ に属するノードが逐次結合または分岐結合で道 q をブロックしているため, *(ii)* が同時に成り立つか否かによらず, 道 q は $\mathbf{Z} \cup \{A\}$ でブロックされる. 以上から, 道 q は *(i), (ii)* を同時に満たす場合,

$\mathbf{Z} \cup \{A\}$ でブロックされる.

(3) **道 p, q が存在しない場合**　X と Y を結ぶ道が存在しないため, 任意のノード集合を所与として X と Y は条件付き独立である. したがって, X と Y は $\mathbf{Z} \cup \{A\}$ を所与として条件付き独立である.

以上より, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$, $\forall A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Z} \cup \mathbf{W})$ について, 式 *(8)* が成り立つ. □

Pearl は以下の弱推移性がベイジアンネットワークの条件付き独立性で成り立つことを示した [7].

定理 付録 A.1 (Pearl [7])**.** $G = (\mathbf{V}, \mathbf{E})$ を *DAG* とし, $X, Y \in \mathbf{V}$ とする. このとき, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$, $\forall A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Z})$ について, 以下が成り立つ.

$$\begin{aligned} &X \perp Y \mid \mathbf{Z} \ \textit{and}\ X \perp Y \mid \mathbf{Z} \cup \{A\} \\ &\Rightarrow X \perp A \mid \mathbf{Z} \ \textit{or}\ A \perp Y \mid \mathbf{Z}. \end{aligned} \tag{9}$$

補題付録 A.1 と定理付録 A.1 を用いて, 定理 4.1 は以下のように証明できる.

証明

(定理 4.1)**.** 補題付録 *A.1* より, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$, $\forall A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Z} \cup \mathbf{W})$ について,

$$X \perp Y \mid \mathbf{Z} \Rightarrow X \perp Y \mid \mathbf{Z} \cup \{A\} \tag{10}$$

が成り立つため, 定理付録 *A.1* より,

$$\begin{aligned} X \perp Y \mid \mathbf{Z} &\Rightarrow X \perp Y \mid \mathbf{Z} \ \textit{and}\ X \perp Y \mid \mathbf{Z} \cup \{A\} \\ &\Rightarrow X \perp A \mid \mathbf{Z} \ \textit{or}\ A \perp Y \mid \mathbf{Z}. \end{aligned} \tag{11}$$

よって, 式 *(11)* より, $\forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$, $\forall A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Z} \cup \mathbf{W})$ について, 式 *(5)* が成り立つ. □

付録 B　定理 4.2 の証明

証明

(定理 4.2)**.** $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ について, $A \in \mathbf{V} \setminus (\{X, Y\} \cup \mathbf{Z} \cup \mathbf{W})$ であるから, $A \notin \mathbf{Z}$ が成り立つ. したがって, $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, A\}$ が成り立つため, 定理 *4.1* より,

$\forall B \in \mathbf{V} \setminus (\{X, A\} \cup \mathbf{Z} \cup \mathbf{W}')$ について, 以下が成り立つ.

$$X \perp A \mid \mathbf{Z} \Rightarrow X \perp B \mid \mathbf{Z} \ or \ B \perp A \mid \mathbf{Z}. \tag{12}$$

また, $\mathbf{Z} \subseteq \mathbf{V} \setminus \{A, Y\}$ が成り立つため, 定理 *4.1* より, $\forall C \in \mathbf{V} \setminus (\{A, Y\} \cup \mathbf{Z} \cup \mathbf{W}'')$ について, 以下が成り立つ.

$$A \perp Y \mid \mathbf{Z} \Rightarrow A \perp C \mid \mathbf{Z} \ or \ C \perp Y \mid \mathbf{Z}. \tag{13}$$

□

付録 C　定理 4.3 の証明

証明

(定理 4.3)**.** *Algorithm3* における入力時の全体グラフ $\mathcal{G}_{all} = (\mathbf{V}_{all}, \mathbf{E}_{all})$, 二ノード X, A, ノード集合 $\mathbf{Z}$ について, $X \perp A \mid \mathbf{Z}$ が成り立つ. このとき, $B \in \mathbf{Adj}(X, \mathcal{G}_{all}) \cap \mathbf{Adj}(A, \mathcal{G}_{all}) \backslash (\mathbf{Ch}(X, \mathcal{G}_{all}) \cap \mathbf{Ch}(A, \mathcal{G}_{all}) \cup \mathbf{Z})$ とすると, $B \in \mathbf{Adj}(X, \mathcal{G}_{all}) \cap \mathbf{Adj}(A, \mathcal{G}_{all})$ であるため, $\mathcal{G}_{all}$ に E_{XB} かつ E_{BA} が存在する. また, $\mathbf{Adj}(X, \mathcal{G}_{all}) \cap \mathbf{Adj}(A, \mathcal{G}_{all}) \subseteq \mathbf{V}_{all} \setminus \{X, A\}$ であり, $\mathbf{W}'$ を真の構造において X と A が合流結合するノードとその子孫からなるノード集合とすると, 仮定から, $(\mathbf{Adj}(X, \mathcal{G}_{all}) \cap \mathbf{Adj}(A, \mathcal{G}_{all})) \cap \mathbf{W}' = \mathbf{Ch}(X, \mathcal{G}_{all}) \cap \mathbf{Ch}(A, \mathcal{G}_{all})$ であるから,

$$\mathbf{Adj}(X, \mathcal{G}_{all}) \cap \mathbf{Adj}(A, \mathcal{G}_{all}) \setminus (\mathbf{Ch}(X, \mathcal{G}_{all}) \cap \mathbf{Ch}(A, \mathcal{G}_{all}) \cup \mathbf{Z}) \subseteq \mathbf{V}_{all} \setminus (\{X, A\} \cup \mathbf{Z} \cup \mathbf{W}') \tag{14}$$

である. したがって, $B \in \mathbf{V}_{all} \setminus (\{X, A\} \cup \mathbf{Z} \cup \mathbf{W}')$ である. また, $X \perp A \mid \mathbf{Z}$ であるから, 定理 *4.2* より,

$$X \perp B \mid \mathbf{Z} \ or \ B \perp A \mid \mathbf{Z} \tag{15}$$

が成り立つ. □