数理解析研究所講究録
第1950巻 2015年 1-11

決定木複雑性における複数アドバーサリーの方法：有向アサイクリックグラフの場合

金山 寛奈
首都大学東京 理工学研究科 数理情報科学専攻
Hirona Kanayama
Department of Mathematics and Information Sciences,
Tokyo Metropolitan University

1 序

1.1 背景

有向グラフがcycleをもたないとき，Directed Acyclic Graph(DAG) という．

DAGは，ベイジアンネットワークに応用がある．ベイジアンネットワークとは，有限個の確率変数の集合に対して，それらの条件付き独立性を有向グラフのd-分離性で表現したものである．このときに使われるグラフがDAGである．ベイジアンネットワークを応用し，スパムメールフィルターが実用化されている（鈴木譲[8]）．

与えられた有向グラフGがDAGであるかを判定するための計算コスト，具体的に言い換えると，次の関数fの計算コストはどれだけ必要なのだろうか？

f:頂点数nのGを入力として受けとり，GがDAGのとき1，
そうでないとき0を返す（Gは連結なものに限る）．

ここで，グラフは隣接行列を用いて表現される．隣接行列の各成分をブール変数とみなすと，fはブール関数となる．ブール関数の計算コストとして活発に研究されているものに決定木複雑性がある．

本研究では，入力したグラフがDAGであるか判定する計算を次のように考える．関数fは，隣接行列で表現したグラフを入力とする．決定性アルゴリズムは，各辺へ問い合わせを行い，1あるいは0の値を応答として受け取る．1はそこに辺があることを表し，0は

辺がないことを表す．このような問い合わせと応答のやりとりを何回か行い，そのグラフがDAGであると判定できた時点で1，そうでないとき0を出力する．そのとき，決定性アルゴリズムAに対するグラフの計算コストを，「Aに対して，DAGかどうかが判明するまでにかかった辺への問い合わせ回数」と定義する．決定性アルゴリズムは，2種類のものについて考える．探索の履歴によって次に探索する辺を変えないものをnon-adaptive algorithmといい，そうでないものをadaptive algorithmという．

ブール関数の決定木複雑性に関しては，乱数なしの指標の一種であるdeterministic complexity $D(f)$とrandomized complexity$R(f)$が度々研究の対象となる．deterministic complexityとは，最も効率のよい決定性アルゴリズムに最悪な入力を入れて計算した時にかかるコストを表す．randomized complexityとは，決定性アルゴリズムの集合上の確率分布について，その確率分布が真理値割り当て全体に対して，どれだけコスト期待値を抑えることができるかという指標である．

fのdeterministic complexityについては，先行研究によって明らかになっている．Holt, Reingold [5]は$D(f) \geq n(n-1)/2$を，Best,Boas,Lenstra[2]はAanderaa-Karp-Rosenberg conjectureに関連した研究で，$D(f) = n^2 - n$を示した．後者は，多項式と数え上げを用いて示されている．なお，Aanderaa-Karp-Rosenberg conjectureは決定木複雑性についての有名な予想である．後の1.2項でその概要を述べる．

non-adaptive algorithmの場合に，$D(f) = n^2 - n$であることを示すために用いた手法が本研究における主結果である．$D(f) = n^2 - n$であることを，我々はadversary argumentを用いて示す．ここで言うadversaryとは，与えれたアルゴリズムに対して，$n^2 - n$手かかる入力を与える関数である．一般的には，用いるadversaryは一つであるが，本研究では二つのadversaryで挟み撃ちにすることに特色がある．一つ目のadversaryでコストを最大化できないアルゴリズムは，二つ目のadverary を適用することでコストを最大化できる．この手法はnon-adaptiveであるという制約をつけているため，Best et al.[2]と完全に同じ結果を出すには至っていない．しかし，二つのadversaryを併用する手法は以前からはあまり知られていない手法であり，今後の発展が期待できる．

また，サブの結果として，fのrandomized complexityや，adaptive algorithmの場合の頂点数が3のときのfのdeterministic complexityを紹介する．

本論文の構成は以下の通りである．

第2節では，有向グラフや決定木計算量に関する定義と解説を述べる．第3節において，本研究の結果を述べる．第4節において，本研究から考えられる今後の課題について述べる．第3節にあるサブの結果については，証明は省く．なお，本論文は修士論文(金山寛奈[7])をもとに書き改めたものである．

1.2 Aanderaa-Karp-Rosenberg conjecture

グラフの計算量について，Aanderaa-Karp-Rosenberg(A.K.R) conjecture がある．「すべての nontrivial(恒真でも恒偽でもない) で monotone(辺を加えても成り立つ) な graph property(頂点のラベル付けに依存せず成り立つ性質) は evasive である」という予想である (Khan,Saks,Sturtevant[6])．evasive とは，隣接行列の全成分を調べないと，property を満たすか判断できないということである．グラフに loop は含まれないとしているので，$n \times n$ 行列の場合，deterministic complexity が $n^2 - n$ であるだろうということを示している．本研究で扱う DAG 判定も，この conjecture の条件を満たす property である．

2 定義

2.1 有向グラフと経路

定義 2.1 (鈴木譲 [8]) グラフとは，いくつかの頂点を辺で結んだものである．グラフには大きく分けて 2 種類あり，辺に向きがついていないものを**無向グラフ**，向きのついたものを**有向グラフ**という．本稿では有向グラフのみを扱うため，ここでは無向グラフの定義は省く．

$\mathbf{U}$ を有限集合，$\vec{\mathbf{E}} \subseteq \vec{\epsilon} = \{(U,V) \mid U,V \in \mathbf{U}, U \neq V\}$ とする．$\vec{G} = (\mathbf{U}, \vec{\mathbf{E}})$ を**有向グラフ**とよぶ．$\mathbf{U}$ を $\vec{G}$ の**頂点集合**，その要素を $\vec{G}$ の**頂点**とよび，$\vec{\mathbf{E}}$ を $\vec{G}$ の**辺集合**，その要素を $\vec{G}$ の（**有向**）**辺**とよぶ．$U,V \in \mathbf{U}$ について、$(U,V) \in \vec{\mathbf{E}}$ を U から V に向かう矢印で表す．$(U,V) \in \vec{\mathbf{E}}$ または $(V,U) \in \vec{\mathbf{E}}$ のとき，U と V は**隣接**しているという．

$U_0, \cdots, U_k \in \mathbf{U}$ とする．$i = 1, \cdots, k$ のそれぞれで $(U_{i-1}, U_i) \in \vec{\mathbf{E}}$ または $(U_i, U_{i-1}) \in \vec{\mathbf{E}}$ を満たすとき，$\{U_i\}_{i=0}^k$ を U_0, U_K を結ぶ，長さ k の**無向経路**とよぶ（簡単に経路とよぶこともある）．

$$U_0 = U_k, U_j \neq U_i, j = 0, \cdots, i-1; i = 1, \cdots, k-1$$

となる無向経路 $\{U_i\}_{i=0}^k$ を長さ $k(\geq 3)$ の**無向巡回経路**とよぶ．

他方，$i = 1, \cdots, k$ のそれぞれで $(U_{i-1}, U_i) \in \vec{\mathbf{E}}$ を満たすとき，$\{U_i\}_{i=0}^k$ を U_0 から U_k への長さ k の**有向経路**とよぶ．

$$U_0 = U_k, U_j \neq U_i, j = 0, \cdots, i-1; i = 1, \cdots, k-1$$

となる有向経路 $\{U_i\}_{i=0}^{k}$ を長さ $k(\geq 2)$ の**有向巡回経路**とよぶ．$\vec{G} = (\mathbf{U}, \vec{\mathbf{E}})$ が有向巡回経路を含まないとき，**有向非巡回グラフ** (Directed Acyclic Graph) とよぶ．簡単のため有向巡回経路を **cycle**，有向非巡回グラフを **DAG** とよぶ．

$\vec{G}$ の任意の異なる 2 つの頂点 X，Y に無向経路が存在するとき，$\vec{G}$ は**連結されている**という．

本研究では，グラフ上の任意の頂点 $U_i, U_j \in \mathbf{U}, (i \neq j)$ について，U_i と U_j を結ぶ辺は $U_i \rightarrow U_j$, $U_i \leftarrow U_j$ の 2 本のみとし，3 本以上は存在しないこととする．

定義 2.2 2 つの頂点 U_i, U_j を結び得る $U_i \rightarrow U_j$, $U_i \leftarrow U_j$ の 2 本の辺を合わせて，**組辺**とよぶ．

2.2 DAG 判定の計算

グラフを表現するために用いる行列を，**隣接行列**という．頂点数 n であるグラフの隣接行列は $n \times n$ 行列 G である．ただし，

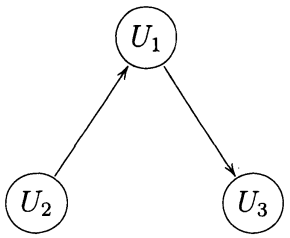
$$G_{ij} = \begin{cases} 1 & ((U_i, U_j) \in \vec{\mathbf{E}}) \\ 0 & (\text{それ以外}) \end{cases}$$

である．(Cormen,Leiserson,Rivest,Stein[4]) $\vec{\mathbf{E}}$ の定義から各対角成分 G_{ii} は 0 である．本論文では，G_{ij} を $G(U_i, U_j)$ と表すこともある．

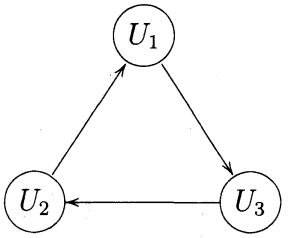
ここで，$\vec{G}$ が DAG か判定する関数 f を定義する．

f:頂点数 n の G を入力として受けとり，G が DAG のとき 1，
そうでないとき 0 を返す（G は連結なものに限る）．

例 2.3 • $G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ のとき，$f(G) = 1$



- $G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ のとき，$f(G) = 0$



2.3 決定木複雑性

本研究で言う「計算量」とは時間計算量ではなく，何個のブール変数に問い合わせたかで計る計算コストのことである．本稿では，グラフを隣接行列で表し，各成分に何回問い合わせたかで計算量を計る．

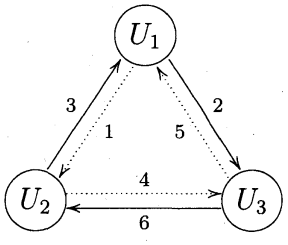
定義 2.4 頂点数 n のグラフの辺の真理値割り当て全体を $\mathcal{G}$ と表記する．ただし，グラフは連結されているものに限る．また，定義から隣接行列の対角成分は 0 なので，$\mathcal{G}$ の各要素は，対角成分以外の成分を表す $n^2 - n$ ビット列とみなせる．

f の計算コストを以下のように定義する．グラフの各辺に真理値を割り当て，その値は隠されているものとする．次に，決定性アルゴリズムにより辺を探索させ，隠された真理値を明らかにする．ここで，決定性アルゴリズムとは，「隠された辺の真理値を知るために，どの辺をどの順番で探索するかの手順」のことである．探索の履歴（クエリの応答の履歴）に依存せず，探索に先立って固定した順序にしたがって次に探索する辺を選ぶアルゴリズムを non-adaptive algorithm といい，そうでないものを adaptive algorithm という．(Buhrman,Spaan[3])

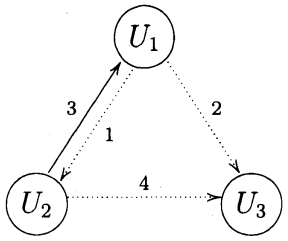
このとき，入力されたグラフが DAG かどうかがわかるまでに探索する辺の数は，入力するグラフや決定性アルゴリズムによって変わる．そこで，G に対しアルゴリズム A によって探索していくときの**コスト** $\mathrm{cost}(A, G)$ を，「G が DAG かどうかがわかるまでに探索した辺の数」と定義する．

例 2.5 A を $G(U_1, U_2), G(U_1, U_3), G(U_2, U_1), G(U_2, U_3), G(U_3, U_1), G(U_3, U_2)$ の順番に探索していくアルゴリズムとする．

- $G_1 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ のとき，$\mathrm{cost}(A_1, G_1) = 6$



- $G_2 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ のとき，$\mathrm{cost}(A_2, G_2) = 4$



定義 2.6 (Arora,Barak [1]) 頂点数 n のグラフの辺を探索する決定性アルゴリズム全体を $\mathcal{A}_D$ と表記する．

$$D(f) = \min_{A \in \mathcal{A}_D} \max_{G \in \mathcal{G}} \mathrm{cost}(A, G)$$

を f の **deterministic complexity** という．

定義 2.7 (Arora,Barak[1]) 決定性アルゴリズム全体の集合上の確率分布を考える．このような分布全体の集合を $\mathcal{A}_R$ とする．このとき，

$$R(f) = \min_{A_R \in \mathcal{A}_R} \max_{G \in \mathcal{G}} \mathrm{cost}(A_R, G) \tag{1}$$

を f の **randomized complexity** という．
ここで，$\mathrm{cost}(A_R, G)$ はコスト期待値を表し，

$$\mathrm{cost}(A_R, G) = \sum_{A \in \mathcal{A}_D} \mathrm{prob}[A_R \text{ is } A] \times \mathrm{cost}(A, G) \tag{2}$$

である．

2.4 Adversary

Deterministic complexity の下界を調べる手法の一つに adversary がある．Adversary とは，(1)k 手目より前のクエリとアンサーの履歴と (2)k 手目のクエリが指定したブール変数 x の二つを受け取って，0 または 1 を返す関数である..

特に，与えられたアルゴリズムに対して，出力が判明するまでの手数をなるべく多くするようなものを考える．直観的な解釈として，次のように説明できる．アルゴリズムがプレイヤーI，隣接行列の各成分をセットする側がプレイヤーIIとなり，成分を1つ知るたびプレイヤーIがプレイヤーIIに1ドル払う状況を考える．このとき，プレイヤーIIにとっての戦略が adversary であり，プレイヤーIから見て敵 (adversary) なのである．

例 2.8 (Arora,Barak [1])n 変数の OR 関数について，deterministic complexity を考える．この場合の adversary を，「$n-1$ 手目までの各クエリに対して 0 と答える」とする．すると，どのようなアルゴリズムであろうと n 手目をクエリするまで OR 関数の値はわからない．このことから，$D(f) = n$ が導ける．

3 結果

3.1 DAG判定のdeterministic complexity

本節では，アルゴリズムとしてnon-adaptive algorithmを考える．まず，DAGの判定をする関数fについて，$D(f) = n^2 - n$になることを示す．そのためにadversary argumentを用いる．adversaryが一つではうまくいかないが（例3.2），二つ併用することで証明できる．

定義 3.1 決定性アルゴリズムによって辺を一つずつ探索するとき，二つのadversary，アド1とアド2を次のように定める．

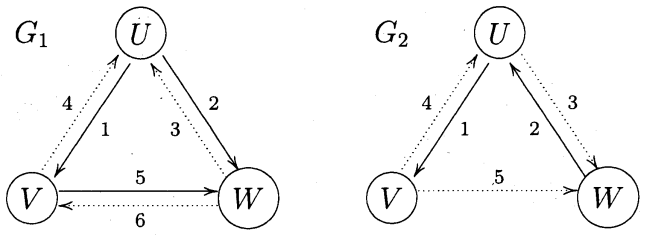
- アド1：各クエリに対して，基本的に1を返す．ただし，1を返すことでcycleができてしまう場合は，1を返さずに0を返す．
- アド2：最初のクエリから1ステップずつアド1のシミュレーションをしていく．シミュレーションが問題なく続いている間は常にアド1の0,1を反転させた結果を返す．ただし，アド1のシミュレーションが終了してしまった後は，ひたすら0と答える．

例 3.2 A_1, A_2をそれぞれ

$$A_1\colon G(U,V), G(U,W), G(W,U), G(V,U), G(V,W), G(W,V)$$

$$A_2\colon G(U,V), G(W,U), G(U,W), G(V,U), G(V,W), G(W,V)$$

の順番にクエリするアルゴリズムとする．A_1，A_2にアド1を適用してできるグラフをそれぞれG_1, G_2とすると，$\mathrm{cost}(A_1, G_1) = 6 = 3^2 - 3$となるが，$A_2$にアド1を適用すると，$\mathrm{cost}(A_2, G_2) = 5 < 3^2 - 3$となる．つまり，OR関数の場合（例2.8）と異なり，アド1だけでコストを最大化できないことがある．同様に，アド2だけでもコストを最大化できないことがある．



補題 3.3 アド1を適用したとき，ある組辺（p.4定義2.2参照）の1本目が0ならば2本目は1である．ただし，この2本目をクエリする時点でDAG判定がまだできていないものとする．

証明 ある組辺の1本目が$G(U_i, U_j) = 0$とする．アド1の定義により，U_jからU_iへの有向経路が存在する．この組辺の2本目が$G(U_j, U_i) = 0$とすると，U_iからU_jへの有向経路が存在し，cycleができる．2本目をクエリする時点でDAG判定がまだできていないという仮定に矛盾しているので，組辺の1本目が0ならば2本目$G(U_j, U_i) = 1$である． □

補題 3.4 アド1を適用して，判定の結果がDAGであるとき，すべての組辺は，

(1) 2本ともクエリされていて，0,1が1本ずつ

(2) 1本目は0，2本目がクエリされていない

のどちらかの状態になっている．

証明 2本ともクエリされていない組辺があるとすると，そこで長さ2のcycleを作ることが考えられるので，判定結果がDAGであることに矛盾する．よって，すべての組辺は少なくとも1回はクエリされる．

1本目が1である組辺は，2本目もクエリしないと判定できないので，このような組辺はすべて2本目もクエリされる．アド1を適用しているので，2本目には必ず0が返される．よって(1)の状態になる．

1本目が0である組辺は，2本目がクエリされなければ(2)の状態になる．そうでなければ補題3.3により2本目は1と答える．このとき状態(1)になる．

以上により，各組辺は(1)，(2)のどちらかの状態に決まる． □

アド1を適用したとき，決定性アルゴリズムによってクエリの回数が異なる．$n^2 - n$手かかるとき，すべての組辺は(1)の状態になるが，$n^2 - n$手未満のとき(1)の状態の組辺と(2)の状態の組辺が混在する．

定理 3.5 (主定理) アド1を適用したとき$n^2 - n$手未満でDAG判定できるならば，アド2を適用したとき$n^2 - n$手かかる．

証明 アド1を適用して，$k(< n^2 - n)$手でできたグラフをGとする．Gの各組辺は

(1) 2本ともクエリされていて，0,1が1本ずつ

(2) 1本だけクエリされていて，0が1本のみ

のどちらかになっている．同じアルゴリズムでアド2をk手まで適用すると，Gの各辺の0,1が反転したグラフG'ができる．G'の各組辺は，

(3) 2本ともクエリされていて，0,1が1本ずつ

(4) 1本だけクエリされていて，1が1本のみ

のどちらかの状態になっている．G' にcycleが存在しないことを示そう．G' がcycle Γ をもつと仮定する．Γ 上の各々の有向辺 $U_i \to U_j$ に対し，G は U_j から U_i への有向経路を持つことを示す．G' において $U_i \to U_j$ が (3) の場合，G において有向辺 $U_j \to U_i$ が存在する．それ以外，つまり (4) の場合，G において U_iから U_j への有向辺はない．アド1の定義により，U_j から U_i への有向経路が存在する．

したがって，G は Γ の逆向き，もしくはそれに頂点をいくつか補完したcycleをもつことになり，G がDAGであることに矛盾する．よって，k 手の時点で G' にcycleは存在しない．

G' には (4) の辺があるので，これらの2本目をクエリしないとDAG判定ができない．アド2を適用すると，これらの2本目にはすべて0を返すので，すべての辺をクエリする前に判定できてしまうことはない．ゆえに，アド2を適用すると $n^2 - n$ 手かかる． □

定理3.5より，以下を得る．

系 3.6 (Best et al.(1974) の別証明) non-adaptive algorithmだけを考えるとき，任意の頂点数 n に対し，$D(f) = n^2 - n$

証明 主張3.5より，アド1で $n^2 - n$ 手かからないアルゴリズムが存在しても，アド2を適用すると $n^2 - n$ 手かかることが言えた．すなわち，任意のアルゴリズムに対して $n^2 - n$ 手かかる入力が存在する．よって，題意が示せた． □

3.2 DAG判定のrandomized complexity

乱数を用いることによって計算量を落とせるかどうかは，決定木計算量にとって基本的な問題である．$R(f) < D(f)$ が成り立てば，乱数の利用によって計算コストの節約ができると言える．本節の結果は，指導教員である鈴木登志雄氏によるものである．なお，本節の結果は，主結果である二つのadversaryを用いた解法によるものではない．

命題 3.7 non-adaptive algorithm，adaptive algorithmのうちどちらの場合でも，以下が成り立つ．

(1) $n = 2$ のとき，$R(f) = 2$

(2) $n \geq 3$ のとき，$R(f) < n^2 - n$

系3.6と命題3.7を合わせると，以下を得る．

命題 3.8 non-adaptive algorithm のみを考えるとき,

- $n = 2$ の場合　$R(f) = D(f) = 2^2 - 2 = 2$
- $n \geq 3$ の場合　$R(f) < D(f) = n^2 - n$

3.3 adaptive algorithm の場合の deterministic complexity

本節では，アルゴリズムとして adaptive algorithm を含めた場合を考える．頂点数が 3 の場合 $D(f) = 3^2 - 3 = 6$ になり，non-adaptive algorithm だけ考えたときと同様のことが成り立つ.

補題 3.9 頂点数が 3 のとき，任意のアルゴリズムに対して，次のような入力がある：1 手目が 0 かつ $3^2 - 3 = 6$ 手かかる.

補題 3.9 より，以下が言える.

命題 3.10 頂点数が 3 のとき,

$$D(f) = 3^2 - 3 = 6$$

4 今後の課題について

$D(f)$ について，本研究では non-adaptive algorithm に限定しているため，Best et al.[2] と完全に同じ結果を出すには至っていない．本研究の手法の応用範囲を広げていくのが今後の課題である．特に，Best et al.[2] のようなアルゴリズムに制約のない結果に対して，本研究の手法による別証明を与えることが今後の課題として興味深い.

参考文献

[1] S.Arora, B.Barak, *Computational complexity : A modern approach*, Cambridge university press, Cambridge, 2009.

[2] M.R.Best, P.van Emde Boas, H.W.Lenstra Jr., A sharpened version of the Aanderaa-Rosenberg conjecture, *Report* ZW 30/74, Mathematisch Centrum Amsterdam, 1974.

[3] H.Buhrman, E.Spaan, L.Torenvliet, The relative power of logspace and polynomial time reductions, *Computational complexity*, 3(1993), pp. 231-244.

[4] T.H.Cormen, C.E.Leiserson, R.L.Rivest, C.Stein, *Introduction to algorithms, Third edition*, MIT Press, 2009. 邦訳：アルゴリズムイントロダクション第3版 第2巻＝高度な設計と解析手法・高度なデータ構造・グラフアルゴリズム，浅野哲夫，岩野和生，梅尾博司，山下雅史，和田幸一 共訳，近代科学社，2012.

[5] R.Holt, E.Reingold, On the time required to detect cycles and connectivity in graphs, *Mathematical systems theory*, 6(1972), pp. 103-106.

[6] J.Kahn, M.Saks, D.Sturtevant, A topological approach to evasiveness, *Combinatorica*, 4(1984), pp. 297-306.

[7] 金山寛奈，決定木複雑性における複数アドバーサリーの方法：有向アサイクリックグラフの場合，首都大学東京修士論文，2015.

[8] 鈴木譲，ベイジアンネットワーク入門 確率的知識情報処理の基礎，培風館，2009.