論理関数処理技術による大規模ベイジアンネットワークに対する確率推論手法の研究

研究代表者　浜　口　清　治　島根大学　大学院総合理工学研究科　教授
(研究遂行時 大阪大学 大学院情報科学研究科 准教授)

1 背景と研究目的

確率推論・統計学習の技術は，データを解析し有益な情報を抽出して，より適切な選択を行うための基幹技術として，必要不可欠となっている．確率モデルのひとつであるベイジアンネットワーク[1]は，複数の確率変数の間の関係を有向グラフで表現したグラフィカルモデルの一種であり，医療診断，故障解析，画像処理，バイオ情報処理など多くの分野の問題に適用されてきている．

本研究では，ベイジアンネットワークに関連する問題の内，確率推論の問題をターゲットとしている．確率推論はベイジアンネットワークのモデルの上で，特定の事象が発生する確率を求める問題である．具体的には，故障が起こったときに，原因事象のうち最も確率が大きなものを求める，といった形で利用される．探索空間が巨大であり，確率変数の数が増大するについて，指数的に計算時間が増加する計算困難問題である．一方，高次元の空間の探索に関しては，BDD (Binary Decision Diagram, 二分決定グラフ) によるデータ表現を用いる研究が行われて来ており，実用に供されている．本研究では，特に，BDD の変種である ZBDD (Zero-suppress BDD, 以下 ZBDD)[2] と呼ばれるグラフ構造を利用して，ベイジアンネットワークの確率推論に対する効率的な手法の確立を目指す．

既存の確率推論の高速化の手法として，ベイジアンネットワークを論理式の積和標準形により表現する手法[7] や算術演算回路(Arithmetic Circuit: 以下 AC) を用いる手法[8]，さらに，ZBDD[2]を用いる手法がある．以上は，確率推論を行う前にベイジアンネットワークを別の構造に変換する手法である．別の構造への変換は計算時間の点でオーバーヘッドとなるが，繰り返し確率推論を行う場合には大きく計算量を削減することが可能である．

ZBDD は組み合わせ集合の表現が可能であり，[2] ではベイジアンネットワークをマルチリニアファンクションとよばれる関数と見なして，その各項を組合せ集合で表現している．ZBDD の手法では確率推論時に計算に必要のない部分ネットワークを含まない形でのコンパイルを行うことによって計算量を削減することが可能である．ZBDD の確率推論の計算時間は ZBDD のサイズに比例して増大することがわかっており，ベイジアンネットワークから ZBDD に変換する場合により小さな ZBDD に変換すれば，より効率的に確率推論を行うことが可能となる．

ZBDD への変換には d ツリー[6]と呼ばれる，グラフの分解構造木を用いる手法がある．先行研究では，ZBDD による確率推論の手法が他の手法よりも効果的となる場合があることが示されているが，ZBDD を用いる手法における d ツリーの利用については十分な研究はされていない．本研究では特に d ツリーの生成を工夫することで，ZBDD のサイズを削減して，最終的に確率推論の高速化を目指す．

本研究の手法では，ベイジアンネットワークの確率変数に対して，変数消去順序と呼ばれる順序付けを行って，これを用いて d ツリーを生成する．ここでは，d ツリーから構築される ZBDD の大きさを推測するために，最大クラスタサイズという指標を用いる．より小さな ZBDD への変換するために，この最大クラスタサイズを小さくするような変数消去順序を求める．また，変数消去順序を求める手法として３つの手法を比較する．この３つの手法に対して，ZBDD の２種類の確率推論の手法であるモノリシック法およびパーシャル法で確率推論を行い，他の確率推論の手法である Ace[10]と実験・比較した結果を示す．実験の結果，ベイジアンネットワークのノード数が少ない場合やノード数が多くても観測値を与える確率変数が少ない場合では，本研究における手法がより高速に確率推論を実行できることが示せた．

2 準備

2-1 記法

以下では，確率変数は英字の大文字で表し，その確率変数がとる値を小文字英字と数字で表す．確率変数

の集合 $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ に対して，X_1 の値が x と定まったとき，X_1 はインスタンス化されたという．表記 Σ_A は A が取り得る値について総和を計算することを表す．また $\Sigma_{\mathbf{X}}$ は集合 $\mathbf{X}$ の各確率変数が取り得る値の組合せについて総和を計算することを示す．A が a_1，a_2，a_3 の値を取り得るとき，$A = \{a_1, a_2, a_3\}$ と表記する．確率変数が取り得る値の個数を A_{num} と表記する．例えば，$A = \{a_1, a_2, a_3\}$ のとき $A_{num}=3$ である．

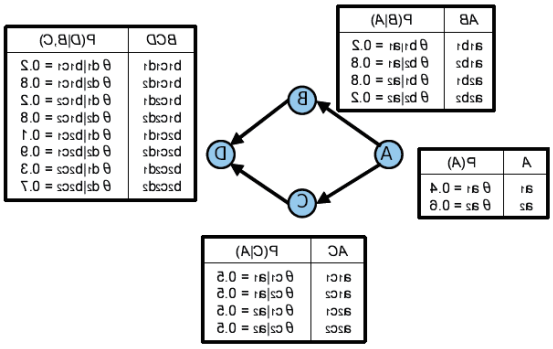

図 1　ベイジアンネットワークの例

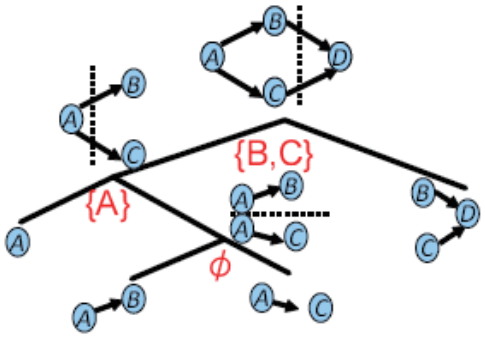

図 2 グラフの分解と d ツリー

2-2 ベイジアンネットワーク

ベイジアンネットワーク(Bayesian Network) は非循環有向グラフと非循環有向グラフの各ノードに付随する条件付き確率分布表(conditional probability table: CPT) で表される．ベイジアンネットワークでは，ノードは確率変数を表しており，エッジによって確率変数間の依存関係を表している．本研究では確率変数が離散であるベイジアンネットワークを取り扱う.それぞれの確率変数は 1 つの CPT と関連づけられている．例えば親ノードにノード A,B を持つノード C の CPT は分布 P(C | A,B) を表す．また，親ノードを持たないノード X の CPT は分布 P(X) を表す．ここではベイジアンネットワークのノード X に対する CPT を CPT(X) と表記する．ベイジアンネットワークに対する同時確率分布は，ベイジアンネットワークに含まれるノードを $X_1, X_2, \ldots, X_n$ とした時，$P(X_1, X_2, \ldots, X_n) = \Pi^n_{i=1}\mathrm{CPT}(X_i)$ で表される．

確率推論は，ある事象が発生する確率を CPT を元に計算することである．ベイジアンネットワークでの確率推論はグラフで示される依存関係を元に計算する．ベイジアンネットワークでの事象 $\mathbf{e}$ に対する確率推論の計算 $\Pr(\mathbf{e})$ は次のようになる．ベイジアンネットワークの全ての確率変数の集合を $\mathbf{X}$，確率変数の集合 $\mathbf{Y} = \{ Y_1, Y_2, \ldots, Y_n \} \subseteq \mathbf{X}$ に対して，$\mathbf{Y}$ は事象 $\mathbf{e}$ によってインスタンス化される確率変数の集合とする．インスタンス化とは確率変数の部分集合の各変数に値を割り当てることである．このとき，次が成立する．

$\Pr(\mathbf{e}) = \Sigma_{\mathbf{X}\setminus\mathbf{Y}}\Pr(\mathbf{X}\setminus\mathbf{Y}, \mathbf{e})$

次に，ベイジアンネットワークの分解[6]について説明する．全ての確率変数の集合を $\mathbf{A}$ とする．確率変数の集合 $\mathbf{X}$，$\mathbf{B}(\mathbf{A} \cap \mathbf{B} = \phi, \mathbf{X} \cup \mathbf{B} = \mathbf{A})$ について $\Pr(\mathbf{X})$ を計算するには $\mathbf{B}$ の各場合について確率を計算し，その和を求めれば良い．つまり，次のようになる．

$\Pr(\mathbf{X}) = \Sigma_{\mathbf{B}}\Pr(\mathbf{X}, \mathbf{B})$

分割統治法の考えに基づいて，$\mathbf{B}$ の各場合について問題を分割する．つまり，$\mathbf{B}$ の確率変数がインスタンス化された状態のベイジアンネットワークを考える．ベイジアンネットワーク$\langle N \rangle$が $\mathbf{e}$ でインスタンス化された時，$\langle N, \mathbf{e} \rangle$ と表記する．この場合，インスタンス化されている確率変数とその確率変数に依存していた確率変数間の依存関係がなくなり，CPT のサイズが小さくなる．

$\Pr^{\langle N;\mathbf{e}\rangle}(\mathbf{X}) = \Pr^{\langle N\rangle}(\mathbf{X}, \mathbf{e})$

以上より次式が成り立つ．

$\Pr^{\langle N\rangle}(\mathbf{e}) = \Sigma_{\mathbf{B}}\Pr^{\langle N\rangle}(\mathbf{e}, \mathbf{B}) = \Sigma_{\mathbf{B}}\Pr^{\langle N;\mathbf{B}\rangle}(\mathbf{e})$

この式は，ある確率変数をその値によって場合分けすることで，元の確率推論の問題を小さな確率推論の問題に分割して独立に計算可能である事を示している．問題の分割に用いられた確率変数の集合 $\mathbf{X}$ をカットセットと呼ぶ．分解されたベイジアンネットワークをそれぞれ$\langle N, \mathbf{X}\rangle^l$，$\langle N, \mathbf{X}\rangle^r$ と表記する．確率変数の集合 $\mathbf{E}$ に対して，$\langle N, \mathbf{X}\rangle^l$ に含まれる確率変数の集合を $\mathbf{E}^l$，$\langle N, \mathbf{X}\rangle^r$ に含まれる確率変数の集合を $\mathbf{E}^r$ とする．このとき次の式が成り立つ．ここで，$\mathbf{C}$ はカットセットである．

$\Pr^{\langle N\rangle}(\mathbf{E}) = \Sigma_{\mathbf{C}}\Pr^{\langle N;\mathbf{C}\rangle L}(\mathbf{E}^l) \times \Pr^{\langle N;\mathbf{C}\rangle R}(\mathbf{E}^r)$

この式は，カットセットによってネットワークが 2 つに分解される時に，カットセットによる場合分けで生じるネットワークを，さらに 2 つに分解して独立に問題を解くことが可能であることを示唆している．ノ

ードが１つになるまで再帰的に分解し，その時得られるグラフの分解木は d ツリーと呼ばれる．d ツリーは２分木であり，その葉には１つのノード，すなわち CPT がラベル付けされる．図 2 は d ツリーの例である．d ツリーについて,子ノード T^l,T^r を持つノード T に対するカットセット cutset(T) とクラスター cluster(T) は次のように定義される．

$vars(T) = vars(T^l) \cup vars(T^r)$

$cutset(T) = (vars(T^l) \cap vars(T^r)) \setminus acutset(T)$

$acutset(T) = \cup_{T \text{ の祖先ノード } T^*} cutset(T^*)$

$context(T) = vars(T) \cap acutset(T)$

$cluster(T) = cutset(T) \cup context(T)$

ここで，T が葉ノードの時，vars(T) と cluster(T) は vars(T)=cluster(T) であり，d ツリーの葉ノードに対応するベイジアンネットワークの変数を表す．例えば，図 2 の根ノード T に対して cutset(T)={B, C} である．

2.3 マルチリニアファンクションに基づく確率推論

本節では[2][7][8][9] で利用されているマルチリニアファンクション(Multi-Linear Function:MLF) とゼロサプレス二分決定図(Zero-suppressed BDD: ZBDD) について説明する．ベイジアンネットワークを表現する MLF とは，項が確率変数のインスタンス化に対応するブール変数 λ と，条件付き確率を表す定数 θ の積によって構成される多項式である．例えば，ブール変数 λ_a は事象 A = a が発生するかどうかを，定数 $\theta_{a1|b1c1}$ は B = b1 かつ C = c1 である時に A = a1 が発生する確率をそれぞれ表わしており，これらの積により項を構成する．

例えば，図 1 のベイジアンネットワーク全体を表す MLF は次のようになる．

$$\begin{aligned} &\lambda_{a1}\lambda_{b1}\lambda_{c1}\lambda_{d1}\theta_{a1}\theta_{b1|a1}\theta_{c1|a1}\theta_{d1|b1c1} \\ +\ &\lambda_{a1}\lambda_{b1}\lambda_{c1}\lambda_{d2}\theta_{a1}\theta_{b1|a1}\theta_{c1|a1}\theta_{d2|b1c1} \\ +\ &\lambda_{a1}\lambda_{b1}\lambda_{c2}\lambda_{d1}\theta_{a1}\theta_{b1|a1}\theta_{c2|a1}\theta_{d1|b1c2} \\ &\qquad + \\ +\ &\lambda_{a2}\lambda_{b2}\lambda_{c2}\lambda_{d2}\theta_{a2}\theta_{b2|a2}\theta_{c2|a2}\theta_{d2|b2c2} \end{aligned}$$

MLF を使って事象 e について確率推論するには，各 $u \in e$ について λ_u を１に，それ以外の λ を０にする.ベイジアンネットワーク全体に対する MLF を構成することで確率推論を行う方法は列挙法と同じであり，確率変数の個数が n 個で，各確率変数の取り得る値の数が m であるとすると，必要な計算量は O(mn) となる．そのため，MLF を適切な構造で表現する必要がある．

2.4 ゼロサプレス二分決定図とマルチリニアファンクション

順序付け規約二分決定図(Reduced Ordered Binary Decision Diagram:ROBDD) は[7] などで述べられているように，ブール関数を表現したものである．以下では，BDD と記す．BDD はシャノン展開を繰り返し適用することで得られる二分木を規約化することにより得られる．また，木を根から葉に向けてたどった時に登場する変数の順序が固定されている場合，グラフとして一意となる．BDD の内部ノードは１つの変数と対応しており，各内部ノードは 0 枝，1 枝を持ち，終端には 0 終端ノードと 1 終端ノードが存在する．BDD を根から 1 終端ノードへたどった時，そのパスは関数が真となる割り当てに対応している．例えば，a から出ている 1 枝と，b でラベル付けされているノードの 0 枝を通り，1 終端に向かっているパスは，$a \wedge \neg b$ に対応しており，表現されている関数が真であることをあらわす(0 終端に向かっている場合は偽)．パス中に登場しない変数が存在する場合は，その変数については don't care であることを表す.通常 BDD は論理関数を表すのに用いられるが，組み合わせ集合を表すために用いることも可能である．組み合わせ集合とはアイテムの組み合わせからなる要素の集合のことである．例えば，3 つのアイテム a，b，c を考えると組み合わせは a，ab，bc などであり，組み合わせ集合は{ab，c},{bc，1} などとなる．ここで，要素 1 はどのアイテムも選択しない場合を表している．組み合わせ集合 S を論理関数 F で表すためには，F が真となる変数の割り当てと S の要素を対応させる．

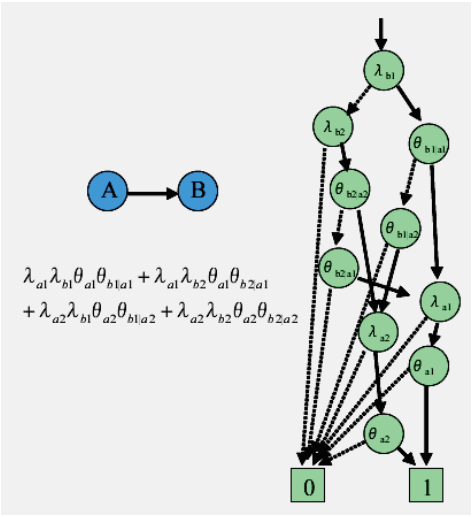

図 3 MLF を表した ZBDD の例

ZBDD は BDD の一種であり，組み合わせ集合を操作するのに適している[3]．ZBDD は BDD と異なり次の簡約化規則を持って

いるが，変数順序が同じであればグラフとして一意となる．

［ZBDD の簡約化規則］1 枝が 0 終端を指しているノードを削除し，削除するノードを指しているエッジを，削除するノードの 0 枝が指している先につなぐ．

ZBDD の特徴は次の通りである．

組み合わせ要素に含まれないアイテムに対応するノードは，ZBDD の簡約化規則から自動的に削除される．

根から 1 終端ノードへの各パスは組み合わせ集合中の各要素に一対一に対応する．

同型の ZBDD が生成されると，共有される．これにより，ノード数がより少ないグラフで表現することが可能になる．

本研究の手法では，MLF を組み合わせ集合と見なし，ZBDD で表現する．具体的には MLF の各項を組み合わせ集合の要素と見なし，MLF の変数 λ と定数 θ を ZBDD の変数に対応させる．図 3 は簡単なベイジアンネットワークの例とその MLF，さらに対応する ZBDD を示している．ZBDD は MLF を少ないノード数で表すことが可能である．また，確率推論の計算は ZBDD を再帰的に探索して，終端ノードから根に向かって確率を伝搬して行うことができ，ZBDD のグラフサイズに比例した時間を要する．また，確率推論を行う際に，インスタンス化される変数によって確率の伝搬が不必要な枝が発生することがある．また，インスタンス化される変数がある場合，確率の伝搬が不必要な枝が発生するため，計算時間が減少する．

3 ゼロサプレス二分決定図による確率推論

3.1 分解ノード付きゼロサプレス二分決定図

前節でベイジアンネットワークのサイズが大きい場合は構成される ZBDD が著しく大きくなる．これは，全体のベイジアンネットワークを表す時にノードの共有を十分に利用できないためである．例えば図 4(a) のベイジアンネットワークに対して，ZBDD を構築すると，(c) のようになる．一方，Pr(B|A),Pr(C|A) に対する MLF を ZBDD で表現すると，(b) のようになる．AB 間つまり，P(B|A)を ZBDD で表すと，図のように B に対するノードが共有されているが，全体の ZBDD を構築すると B に対するノードは共有されていない．この図の A と B，A と C 間のように依存関係がある場合では，B と C の部分を独立して取り扱うことが不可能であるためにノードの共有が発生せず，ZBDD のグラフサイズが大きくなる．

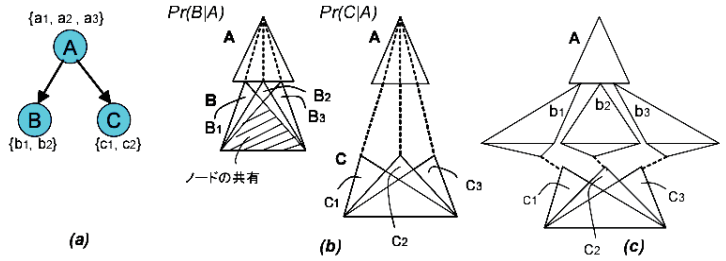

図 4　ZBDD の非共有化

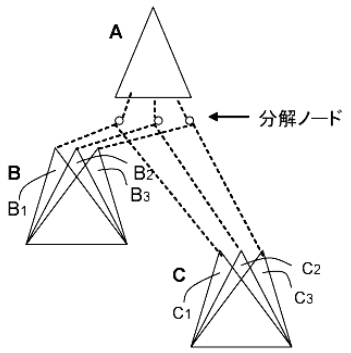

図 5　ZBDD の分解

この問題を解決するために，分解ノードというノードを導入して ZBDD を構成することで ZBDD のグラフサイズを低減する手法を説明する．ZBDD の分解ノードは例えば図 4 のようなベイジアンネットワークに対しては，図 5 のように用いられる．このベイジアンネットワークでは A から B と C の 2 つに依存関係があるため，全体の ZBDD では A の後に分解ノードが置かれている．分解ノードによって B と C を独立して取り扱うことが可能となるため，全体の ZBDD を構築した場合でも ZBDD のサイズが増大しない．

分解ノード付き ZBDD は d ツリーのノードをたどって，λ 変数，θ 変数，分解ノードの順序を決定することで構築される．具体的には d ツリーのノードを次のようにたどっていく．(1) 根ノードから深さ優先探索を行う．(2) 内部ノードを訪問した場合は，カットセットに含まれる変数に対する λ 変数を順序の後ろに加える．さらに，内部ノードに対する分解ノードを順序の後ろに付け加える．

ここで，内部ノード T を訪問した時に加えられる分解ノードの数に着目する．分解ノードは，内部ノード T に対応するノードを通るパスの数が加えられることになる．内部ノード T のクラスタを cluster(T)={A,B,C}とすると，内部ノード T を訪問した際に加えられる分解ノードの数は最大で $A_{num} \times B_{num} \times C_{num}$ となる．

3.2 モノリシック法とパーシャル法による確率推論

この節では ZBDD で確率推論を行うために用いられる，ZBDD を構築する手法である，モノリシック法とパーシャル法[4]について説明する．どちらの手法も MLF を ZBDD で表すことで確率推論を行う．モノリシック法は図 6 のように，ベイジアンネットワーク全体を 1 つの ZBDD で表して確率推論時にその ZBDD を使う手法である．一方，パーシャル法は，まず図 7 のように各ノードに対してそのノードと祖先ノードを含むベイジアンネットワークを表す ZBDD を 1 つずつ構築しておき，確率推論時には必要な ZBDD を組み合わせることで構築して計算を行う手法である．つまり，パーシャル法で確率推論時に用いられる ZBDD は，必ずしも全体のベイジアンネットワークを構築する必要がない．

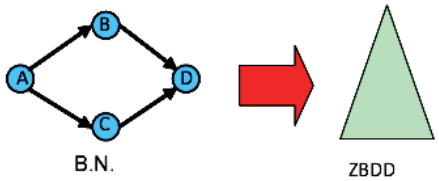

図 6 モノリシック法による ZBDD の構成

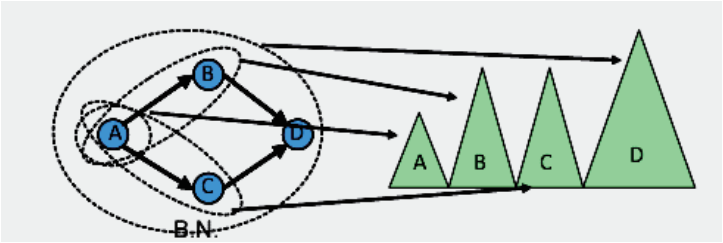

図 7 パーシャル法による ZBDD の構成

3.3 最大クラスタサイズ，変数消去順序，d ツリー

d ツリーの全ての内部ノード T のクラスタ cluster(T) の中でもっとも要素数の大きいものを最大クラスタと呼び，最大クラスタの要素数を最大クラスタサイズと呼ぶ．例えば，内部ノード T のクラスタ cluster(T)={A, B, C} に対するクラスタサイズは 3 となる．d ツリーの各ノードに対して ZBDD に追加される分解ノードの数は，d ツリーのクラスタに依存する．したがって，サイズの大きなクラスタに含まれる変数に対する分解ノード数は，他の内部ノードに比べると多くなると予測でき，最大クラスタサイズを，その d ツリーから構築される ZBDD の最大幅を予測するための指標とすることができると考えられる．したがって，最大クラスタサイズの小さな d ツリーを構築することで効率的な確率推論が可能になる．

以下では，クラスタサイズに重みをつけた重み付きクラスタサイズを定義する．最も重みが大きいクラスタを重み付き最大クラスタと呼び，重み付き最大クラスタにつけられた重みを，重み付き最大クラスタサイズと呼ぶ．クラスタサイズの重みは，クラスタ内の変数 X のとり得る値の数 X_{num} の全ての積の対数(底は２)であり，cluster(T)={A,B,C}となるノード T に対して $\log(A_{num}\times B_{num}\times C_{num})$ である． d ツリー上で同じクラスタサイズを持っていても，クラスタに含まれる変数がとり得る値の数が違う場合には，ZBDD に追加される分解ノードの数は異なる．そこで，重み付き最大クラスタサイズによってその違いを反映させる．

モラルグラフは各確率変数間の関係の有無を表した無向辺グラフであり，各 CPT ごとに，その中の確率変数の間には関係があるとみなす．モラルグラフは次の手順によりベイジアンネットワークから構築される．(1)ベイジアンネットワークの有向辺を無向辺に変える．(2)ベイジアンネットワークで２つ以上のノードからエッジを向けられているノードがあって，エッジの始点となるノードの間にエッジがなければ，無向辺を追加する．

変数消去順序は，ベイジアンネットワークの変数間の全順序である．d ツリーは変数消去順序から構築できる．変数消去順序とモラルグラフから次の width と呼ばれるパラメータが定義される． π(i) を変数消去順序の i 番目の変数とする．ここで，$|C_i|$ は C_i に含まれる変数の数を表す．

C_1 = {π(1) に隣接するノード}

C_i = {π(i) に隣接するノード}＼$\Sigma_{j<i}C_j$　　(i ＞ 2 のとき)

$width(\pi) = \max^{n}_{i=1} |C_i|-1$

width の値が w の時，ベイジアンネットワークと変数消去順序からクラスタサイズが w 以下の d ツリーを構築する手法がある．ZBDD のサイズは d ツリーの最大クラスタサイズの大きさに依存することから，より最大クラスタサイズの小さい d ツリーを構築するような変数消去順序を求めることで，効率的な確率推論が可能となる．

3.4 変数消去順序からの d ツリーの構築

変数消去順序から d ツリーを構築する手順を説明する．ベイジアンネットワークのノード数を n とする．まず，ベイジアンネットワークの確率変数に対応する葉ノードを 1 つだけ持つ d ツリーを n 個用意する．変数消去順序 π の i 番目の変数を π(i) とし，d ツリーに対して，i = 1 から i = n まで順に，変数 π(i) に対応する葉ノードを含む d ツリーが複数あるとき，d ツリー同士をつなげた d ツリーを構築する．

変数消去順序はベイジアンネットワークからモラルグラフを構築し，そのモラルグラフに対して図 8 のア

ルゴリズムを実行することで求める．本研究で比較する３つの手法では次のアルゴリズムの中でコストを求める関数と変数を取り除く際の処理で差異がある．モラルグラフの各頂点のコストを求め，最もコストが小さい頂点を取り除く．この処理を全部の頂点がなくなるまで繰り返し実行し，頂点を取り除いた順番を変数消去順序とする．コスト関数はベイジアンネットワークに対してそのノード分解した時，構成される d ツリーのクラスタサイズがどれほど大きくなるかを見積もるための関数である．本研究で用いる変数消去順序の３つの手法はコスト関数の部分で次のような違いがある．

```
Algorithm: EliminateOrder
入力: モラルグラフG　出力: 頂点のリストH
1: V ← G の頂点の集合;
2: H;　/*頂点のリスト*/
3: while ( V ≠ φ ) {
4:　　min = INFINITY;
5:　　for i = 1 to n {
6:　　　　c = cost(V[i]); /*cost は頂点のコストを求める関数*/
7:　　　　if c < min then {
8:　　　　　　min = c;
9:　　　　　　min_i = i ;
10:　　　　 }
11:　　}
12:　　add V[min_i] in H; /*リストH の最後に頂点V[min_i] を追加*/
13:　　remove V[min_i] from V;　/*必要ならモデルグラフの変形も行う*/
14: }
15: return H;
```

図８　変数消去順序を求めるアルゴリズム

min degree は各頂点の隣接ノード数をコストとする．min fill では最小コストとなるノードを取り除くとき，そのノードに対する隣接ノード同士間でエッジがなければエッジを追加する．この時追加されるエッジを fill in edge と呼び，min fill では各ノードに対する fill in edge の数をコストとして計算する．

重み付き min fill では，両端のノードの確率変数 X,Y である fill in edge に対して，$\log(X_{num} \times Y_{num})$の重みをつける．例えば，図 22 で AE 間の fill in edge の重みは $A_{num} = 2, E_{num} = 3$ より log 6 になる．重み付き min fill をコストとして用いた場合，各ノードに対する fill in edge の重みの合計をコストとして計算し，最小のコストとなるノードを取り除き，かつ fill in edge をエッジとして追加する手法である．

min degree や min fill では最大クラスタサイズを小さくするために，隣接ノードや fill in edge の数をコストとみなして計算していたが，重み付き min fill では重み付き最大クラスタサイズが小さい d ツリーを構築することを目的としてコスト計算を行なっている．

4 実験と考察

本節では ZBDD を利用してベイジアンネットワークで確率推論を行う手法について前節で提案した手法を実装して，それぞれのアルゴリズムをモノリシック法とパーシャル法で確率推論を行った結果について述べる．さらに，Ace コンパイラ[10]との比較結果を示す．Ace コンパイラはベイジアンネットワーク全体を一つの AC(Arithmetic Circuit)と呼ばれるグラフ構造にコンパイルし，確率推論を行う．この手法は前節のモノリシック手法に類似した手法である．

実験に使用したベンチマークは[11] で提供されているベイジアンネットワークである．以下の実験は，CPU:Intel Pentium Dual-Core 2.5GHz，メモリ：3.2GB.OS: Vine Linux という環境で行った．

表１は３節で示した３つのコスト関数のもとで変数消去順序を計算するのに要した時間と，その変数消去

表１ 変数消去順序の計算結果

	ノード数	手法	計算時間 (秒)	Max clu	W Max Clu
alarm	37	min degree	**0.01**	5	8.17
		min fill	**0.01**	4	**7.17**
		重み付き min fill	**0.01**	4	**7.17**
hailfinder	56	min degree	**0.01**	10	18.1
		min fill	**0.01**	**5**	**11.7**
		重み付き min fill	**0.01**	**5**	**11.7**
Water	32	min degree	**0.01**	12	22.3
		min fill	**0.01**	11	**20.8**
		重み付き min fill	**0.01**	11	**20.8**
Pigs	441	min degree	**0.01**	22	39.6
		min fill	1.2	11	**17.4**
		重み付き min fill	1.23	11	**17.4**
Mildew	35	min degree	**0.01**	8	26.7
		min fill	**0.01**	**5**	**20.7**
		重み付き min fill	**0.01**	**5**	**20.7**
Munin2	1003	min degree	**0.04**	24	63.0
		min fill	4.16	9	18.9
		重み付き min fill	4.23	**8**	**17.4**
Munin3	1044	min degree	**0.04**	20	62.5
		min fill	8.47	**8**	17.8
		重み付き min fill	8.75	**8**	**17.3**
Munin4	1041	min degree	**0.05**	17	35.3
		min fill	9.43	**9**	**21.4**
		重み付き min fill	9.59	**9**	**21.4**
Diabetes	413	min degree	**0.02**	65	63.0
		min fill	0.64	**5**	**17.2**
		重み付き min fill	0.68	**5**	**17.2**
Barley	48	min degree	**0.01**	13	37.4
		min fill	**0.01**	**8**	23.6
		重み付き min fill	**0.01**	**8**	**22.8**
Link	724	min degree	**0.05**	37	56.0
		min fill	6.53	**16**	24.0
		重み付き min fill	6.77	14	**21.0**

順序を用いて d ツリーを構成したときの重み付き最大クラスタサイズを比較したものである．Max Clu は最大クラスタサイズを,W Max Clu は重み付き最大クラスタサイズを表している.重み付き min fill は min fill と同じかより小さいクラスタサイズを持つ d ツリーを構成できていることがわかる．ベイジアンネットワークのノード数が 1000 を越える場合でも，10 秒以下で計算が終了している．

表 2 は 3 つのコスト関数を使って，モノリシック法で ZBDD を構築した場合の ZBDD サイズを比較したものである.表の M.O. はメモリが不足したことにより ZBDD が構築出来なかったことを示す．1 つの例題(mildew)を除いて,重み付き min fill による手法が小さな ZBDD の構築に有効であることがわかる．

３つのアルゴリズムに対してそれぞれモノリシック法とパーシャル法で確率推論を行った結果と，Ace コンパイラで確率推論を行った結果を典型的な場合について抜粋して示す．なお，min degree についても実験を行っているが，他の手法よりも常に大きな計算時間を要するため，紙面の都合で割愛し，以下では， min fill と 重み付き min fill，そして ACE についての結果を示す．

表 2 64 ビット CPU による結果(パーシャル法)

	コンパイル時間	1ins	2ins	3ins	4ins	5ins
alarm	0.49	0.0001	0.0001	0.0003	0.0005	0.0008
hailfinder	0.57	0.0001	0.0002	0.0009	0.0013	0.0012
Water	0.69	0.0006	0.001	0.0019	0.0005	0.0008
Pigs	0.63	0.0001	0.0004	0.0007	0.0007	0.0015
Mildew	6.4	0.0331	0.2231	0.4143	0.6054	0.9782
Munin1	91.49	0.1724	1.6401	6.6934	44.1036	20.1629
Munin2	10.35	0.0069	0.0161	0.0233	0.0346	0.0425
Munin3	33.25	0.0266	0.0656	0.1171	0.2303	0.3676
Munin4	11.05	0.0066	0.0139	0.0273	0.0543	0.0624
Diabetes	7423.27	10.398	66.4152	111.133	159.276	230.997
Barley	444.69	3.3135	32.5016	75.2513	74.5654	194.889
Link	1.75	0.001	0.0117	0.0442	0.1748	0.5308

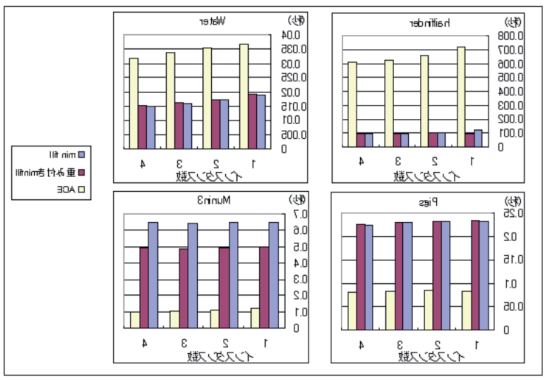

図 9 確率推論の実験結果(モノリシック法)

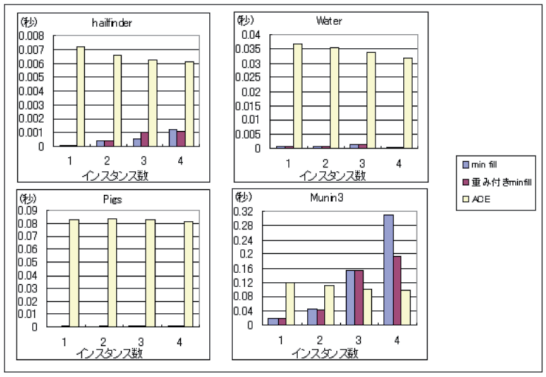

図 10 確率推論の実験結果(パーシャル法)

図 9 に，モノリシック法で min fill，重み付き min fill と Ace コンパイラで確率推論を行った結果を示している．インスタンス化された確率変数の数が 1，2，3，4 個の場合で確率推論を行い，それぞれ 100 回実行した平均時間である.コンパイルに要した時間は含まれていない.ノード数が小さい hailfinder と Water では ZBDD を用いた手法のほうが高速であるが，ノード数が大きい Pigs，Munin3 では Ace コンパイラの方が高速という結果になっている．次に 図 10 にパーシャル法で min fill，重み付き min fill と Ace コンパイラで確率推論を行った結果を示す．パーシャル法ではインスタンス化された確率変数の数で時間を比較すると，数が小さいほど計算時間も小さくなっていることがわかる．特に，インスタンス化された変数の数が１個，２個の場合の時間は，どの場合でも Ace コンパイラよりも短い．これは，パーシャル手法では確率推論時に，必要な ZBDD を構築するため，インスタンス化された変数の数が小さい場合は，ZBDD の構築にあまり時間を必要とせず，また小さな ZBDD に対して確率推論を行えばよいためであると考えられる．

以上の結果をまとめると，モノリシック法ではベイジアンネットワークでノード数が小さい場合，パーシャル法ではノード数が小さい場合か，ノード数が大きくてもインスタンス化された変数の数が少ない場合で ZBDD を用いた手法のほうが Ace コンパイラよりも，短い時間で確率推論が可能となっている．また，３つのコスト関数は min degree が長く時間を要し，ベイジアンネットワークのノード数が大きい場合や，インスタンス化された確率変数の数が多い場合は重み付き min fill が短い時間で確率推論が可能となっている．

ここまでは，ACE との比較のため，32 ビット CPU を用いた実験としていた．次に 64 ビット CPU を用いた実験環境で,重み付き min fill をパーシャル法で確率推論した結果を示す(表 4)．実験は CPU: 2 Quad-Core AMD 2GHz,メモリ: 64GB，OS: Red Hat Linux という環境で行った．Munin1 はノード数 189 のベイジアンネットワークで，前述の実験ではどの手法でも ZBDD に変換出来なかったベイジアンネットワークである．また，Barley および Link など 32 ビット CPU の環境では ZBDD の構築ができない例でも処理することができ

るようになっていることがわかる．

5. まとめ

本研究では，ZBDD を用いたベイジアンネットワーク上での確率推論の速度向上の手法を提案した．本研究の手法では d ツリーを構成する際，変数間で順序を決定する必要があるが，その際のノードの選び方について３つの手法を実装し，比較した．実験の結果，ZBDD を用いた確率推論を行う場合３つの手法では重み付き最大クラスタサイズの小さな d ツリーを構築することを目標としている重み付き min fill が最も少ない時間で確率推論を行うことができた．ACE コンパイラとの比較では，B.N. のノード数が小さい場合や，ベイジアンネットワークのノード数が大きい場合であっても値が定まった確率変数の個数が少ない場合に重み付き min fill による手法が有効であることが示せた．本研究で得られた結果を実用的に用いるためには，特にベイジアンネットワークのノード数が大きく，インスタンス化された確率変数の個数が多い場合には，他の手法と組み合わせる必要がある．実用的には，ZBDD の大きさを d ツリーを構築した段階で，あらかじめ予測できることが必要となるが，重み付き最大クラスタサイズを用いても，ZBDD のサイズを予測することはできておらず，これは今後の課題となっている．

【参考文献】

[1] 繁桝算男(著), 本村陽一(著), 植野真臣(著),“ ベイジアンネットワーク概説”, 培風館, 2006.

[2] S. Minato, K. Satoh, T. Sato, ”Compiling Bayesian Networks by Symbolic Probability Calculation Based on Zero-suppressed BDDs,”In Proc. of 20th InternationalJoint Conference of Artificial Intelligence(IJCAI-2007), pp. 2550-2555, 2007.

[3] S. Minato, ”Zero-suppressed BDDs for set manipulation in combinatorial problems,” In Proc. of 30th Design Automation Conf.(DAC-93), pp.272-277, 1993.

[4] D. Tokoro, K. Hamaguchi, T. Kashiwabara, S. Minato, ”Monolithic and Partial CompilationMethods for Probabilistic Inference of Bayesian Networks using ZBDDs”, 4thInternational Workshop on Data-Mining and Stastical Science, pp.98-103, July 2009.

[5] A. Darwiche,“ Modeling and Reasoning with Bayesian Networks ”, Cambridge University Press, 2009.

[6] A. Darwiche, ”Recursive conditioning,” Artificial Intelligence, vol. 126, No.1-2, pp.5-41, 2001.

[7] M. Chavira and A. Darwiche, ”Compiling Bayesian Netowrks with Local Structure,” In Proc. 19th International Joint Conference on Artificial Intelligence(IJCAI-2005),pp. 1306-1312, Aug. 2005.

[8] A. Darwiche, ”A Differential Approach to Inference in Bayesian Networks,” Journal of the ACM, Vol.50, No.3, pp.280-305, May. 2003.

[9] M. Chavira and A. Darwiche, ”Compiling Bayesian Networks Using Variable Elimination,” In Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 2443-2449.

[10] http://reasoning.cs.ucla.edu/ace/

[11] http://www.cs.huji.ac.il/site/labs/compbio/Repository/

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月
MAP Inference on ZDD-based Representation of Bayesian Networks	Workshop on Advanced Methodology for Bayesian Networks	2010 年 11 月
Monolithic and Partial Compilation Methods for Probabilistic Inference of Bayesian Networks using ZBDD	Workshop on Data-Mining and Statistical Science	2009 年 7 月
Exploiting Global Structures in Bayesian Network Compilation by Zero-suppressed BDDs	International Conference on Inductive Logic Programming	2009 年 7 月