G-10

GPU を用いた高速なベイジアンネットワーク学習手法

A Fast Bayesian Network Learning Algorithm Using GPU Computation

森 隆史† 山中 優馬† 藤木 生聖† 吉廣 卓哉‡
Takashi Mori Yuma Yamanaka Takatoshi Fujiki Takuya Yoshihiro

1. はじめに

ベイジアンネットワークモデルとは，事象を「ノード」その因果関係を「リンク」で表した確率モデルであり，予測や原因推定に用いられる．ベイジアンネットワークで膨大なデータの解析を行うことで，バイオインフォマティクス，医学，情報検索，意思決定支援など様々な分野での応用が期待されている．

ベイジアンネットワークモデルの構造学習は，モデルのスコアを情報量基準とし，モデルのスコアが最も良いモデルを探索する最適化問題と定式化できる．しかし，ノードの数が増加するにしたがって，探索すべきモデルの候補が指数的に増加するため，その最適化問題は NP 困難である[1]．よって，現実的な実行時間で近似的に優れたモデルを求めるアルゴリズムが多数提案されている．

モデルの探索空間を削減するために順序制約を用いた K2[2]と呼ばれるアルゴリズムがある．ここで順序制約とは，事象を $X_1, X_2, \cdots, X_n$ とすると，X_i の親は $X_i \prec X_j$ となる X_j のみとなる制約である．しかし，実用の場面では，順序制約が仮定できない場合が多く，利用できる場面が限られる．

順序制約が仮定できない場合を対象として，優れた順序関係を探索し，K2 などを用いてモデルを探索する方法がある．優れた順序関係を探索するために遺伝的アルゴリズム（GA）適用させ，より良い順序関係に進化させる方法[3]やマルコフ連鎖モンテカルロ（MCMC）法[4]を用いてもサンプリングを行う方法が提案されている．しかし，順序関係の探索は，モデルの探索に対して間接的な方法であるため，優れたモデルを見つけ難く，まだ実用的な水準に達しているとは言い難い．

そこで，モデルの探索空間に対して直接探索を行う方法が提案されている．最も基本的な戦略の 1 つとして，ヒルクライム法[5]や焼きなまし法[6]を用いたモデルの探索が提案されている．また，近年，遺伝的アルゴリズム（GA）を用いた方法が数多く提案されている．その中でも EDA（Estimation of Distribution Algorithm）と呼ばれる種類のアルゴリズムが注目されている．EDA とは，確率ベクトルを用いた遺伝的アルゴリズムの一手法であり，各世代で個体を生成し，優れた個体を用いて確率ベクトルを学習することで，確率的な探索効率を向上する探索手法である．EDA にもいくつかの方法があるが，中でも PBIL（Population-Based Incremental Learning）[7]と呼ばれる方法がベイジアンネットワークの学習に最も適しており，優れた解を効率的に計算することが可能であるという報告がある[8]．しかし，PBIL に基づいたアルゴリズムは局所解に陥る問題があるため，我々はその問題克服した PBIL-RS[9]を提案した．様々なアルゴリズムが提案されているが，大規模なベイジアンネットワークの構造学習は，まだ膨大な時間を要する．莫大なデータの分析が必要とされる今日において，ベイジアンネットワーク学習アルゴリズムの高速化は重要である．

本論文では，ベイジアンネットワーク学習アルゴリズム PBIL-RS に着目し，GPU を用いて高速に構造を学習する手法を提案する．37 ノードのベイジアンネットワーク Alarm network[10]を学習することで評価を行い，一般向け GPU を用いることで，CPU のみの実行に比べ約 14 倍の高速化に成功した．

本論文では，まず，第 2 章でベイジアンネットワークの定義を行ったうえで，PBIL に基づくベイジアンネットワーク学習手法を説明する．そして第 3 章提案手法で想定する GPU のアーキテクチャについても述べる．第 4 章では，本研究の提案手法を説明し，第 5 章で提案手法を評価する．最後に，第 6 章で本研究についてまとめる．

2. ベイジアンネットワークとその学習手法

2.1 ベイジアンネットワークモデル

ベイジアンネットワークモデルとは，事象を「ノード」その因果関係を「リンク」で表した確率モデルであり，予測や原因推定に用いられる．単純なベイジアンネットワークの例を図 1 に示し，これを用いて説明する．ノード X_1，X_2，X_3，はそれぞれの事象を表す確率変数であり，事象が生起するとき 1 をとり，生起しなければ 0 をとる．リンク $X_1 \rightarrow X_3$，$X_2 \rightarrow X_3$ は因果関係を示し，X_1，X_2 を親ノード，X_3 を子ノードと呼ぶ．X_1 と X_2 の生起確率は，それぞれ $P(X_1)$ と $P(X_2)$ で表され，この生起確率 $P(X_1)$，$P(X_2)$ に従って X_1，X_2 の値が決まる．ノード X_3 は X_1 と X_2 を親ノードに持つので，条件付確率 $P(X_3 | X_1, X_2)$ に従って X_3 の値が決まる．例えば，X_1 と X_2 の値がともに 0 をとるとき，X_3 の生起確率は 0.01 である．ここで子ノード X_3 のみが観測された場合，ベイズ推定を用いることにより親ノードの X_1 と X_2 の事後確率を求めることができる．これにより観測した値から知りたい事象の確率を知ることが可能となり，観測結果から原因となった事象を推定することができる．このようにベイジアンネットワークモデルを学習することによって，事象の因果関係を推定することができる．

事象を観測することにより得られたデータ，観測データを $O=\{o_j\}$，$1 \leqq j \leqq |O|$ で表す．事象の数を n とし，事象 X_i に対する j 番目の観測(サンプル)を x_{ij} で表すと，j 番目のサンプルは $o_j=(x_{1j}, x_{2j}, \cdots, x_{nj})$ で表せる．ベイジアンネットワークでは，この観測データ O から優れたモデルを学習する．ここで優れたモデルとは，その優れたモデルに従って事象を生起させた場合に，観測データ O に近いデータが生成されるようなモデルである．モデルの評価基準となる情

†和歌山大学大学院 システム工学研究科，Graduate School of Systems Engineering, Wakayama University, Japan

‡和歌山大学 システム工学部，Faculty of Systems Engineering, Wakayama University, Japan

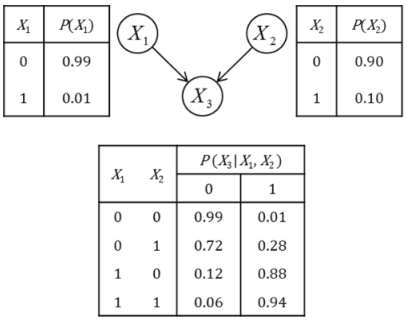

図 1 ：ベイジアンネットワークモデルの例

報量基準（AIC（Akaike’s Information Criterio）[11]など）を用いてモデルのスコアを算出し，スコアが最も低い値をとるベイジアンネットワークモデルが優れたモデルとなる．

2.2 PBIL に基づくベイジアンネットワーク学習

PBIL を用いたベイジアンネットワーク学習手法は，ある 1 つの個体を 1 つのベイジアンネットワークモデルとする．スコアの良いベイジアンネットワークモデルの構造を確率ベクトルに学習させることにより，確率ベクトルが生成するモデルが徐々に最良のベイジアンネットワークモデルに近づいていくことを狙っている．ノードの数を n とし，1 つのベイジアンネットワークモデルを m=(v_{11}，v_{12}，…，v_{1n}，v_{21}，v_{22}，…，v_{n1}，v_{n2}，…，v_{nn})で定義する．ここで v_{ij}（1≦i,j≦n）は，0 か 1 の値を取り，X_i から X_j へのリンクを示す．例えば，v_{ij} = 1 であれば，リンクが存在し，v_{ij} = 0 であれば，リンクが存在しないことを表す．また，個体となるベイジアンネットワークモデルを生成する元となる確率ベクトル P を P=(p_{11}，p_{12}，…，p_{1n}，p_{21}，p_{22}，…，p_{n1}，p_{n2}，…，p_{nn})と定義する．p_{ij} は X_i から X_j へのリンク，つまり v_{ij} = 1 となる確率を表すので，確率ベクトル P は確率表とみなすことができる．その確率表を図 2 に示す．

PBIL によるベイジアンネットワーク学習アルゴリズムの動作は，基本的に PBIL と同じであり，手順は以下のようになる（図 3 は(2)~(4)の手順を示す）．

(1)確率ベクトル P を初期化する．この時 $i = j$ ならば p_{ij} = 0，それ以外は p_{ij} = 0.5 として初期化する．

(2)確率ベクトル P を用いて，$|M|$個のモデルの集合 M を生成する．

(3)全てのモデル $m \in M$ に対して情報量基準から計算されるモデルのスコアを算出する

(4)スコアが良い上位 k 個のモデルをモデル集合 M'とする．M'を用いて以下の式により確率ベクトルを更新する．

$$p'_i = p_i(1-\alpha) + ratio(i) \times \alpha \quad (1)$$

ここで p_i は更新前の確率，p'_i は更新後の確率を表し，$ratio(i)$は，スコアが上位のモデル集合 M'に含まれるモデルのリンク i の割合を表す．α(0<α<1)は学習率と呼ばれるパラメータであり，確率ベクトル P が選択したモデルの構造を学習する程度を表す．

(5)(2)~(4)までの手順を 1 世代とし，確率ベクトル P が収束するまで何世代も繰り返す．

2.3 PBIL-RS

PBIL に基づいたベイジアンネットワーク学習アルゴリズムは優れたベイジアンネットワークモデルを学習するが，

P		親ノード					
		X_1	X_2	…	X_i	…	X_n
子ノード	X_1	0.0	0.5	…	p_{i1}	…	0.5
	X_2	0.5	0.0	…	p_{i2}	…	0.5
	⋮	⋮	⋮	⋱	⋮	…	⋮
	X_j	p_{1j}	p_{2j}	…	p_{ij}	…	p_{nj}
	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	X_n	0.5	0.5	…	p_{in}	…	0.0

図 2：確率ベクトル P

学習によって局所解に陥る問題があった．この問題に対して PBIL-RS[9]は，突然変異を発生させることで，探索領域を広げる．そして，探索領域の拡大幅を段階的に広げていくことで同じ局所解に陥らないように設計されている．PBIL-RS は，探索領域の拡大により，突然変異に基づく他の方法よりも優れた解を見つけることができると報告されている[9].

3. GPU アーキテクチャ

3.1 GPU の基本構造

GPU (Graphics Processing Unit)は元々，画像処理を高速化するために開発された演算装置で，現在は科学技術計算の高速化のためにも利用されている．本研究では GPU を用いるため，想定する GPU の基本構造を図 4 に示し，これを用いて説明する．今回は NVIDIA 社の GPU を対象としているが，他社の GPU も構造は大きく変わらない．GPU 上にはストリーミングマルチプロセッサ（SM）という演算部が複数ある．その中に数百個の最小単位の演算ユニットである CUDA コア（コア）が搭載されている．SM 内のコアが同時に稼働することで並列処理を実現している．

CPU と GPU 間のデータを交換するために，GPU にはグローバルメモリと呼ばれる大容量なメモリが用意されている．GPU は，グローバルメモリに CPU から転送されたデータを保持し，結果をグローバルメモリを通して転送する．また，高速な並列処理を実現するために SM の内部には，シェアードメモリと呼ばれる高速で少容量のメモリが搭載されている．シェアードメモリは同じ SM に属するすべてのコアがアクセスできる．各コアでスレッドが実行され，同じ SM 内の複数のスレッドがそれぞれ異なるメモリ領域に同時にアクセスし，同じ処理を同時に行う．ここでスレッドとは，1 つのコアが行う 1 つの処理である．効率的な並列のメモリアクセスに最適化することが GPU を用いたアルゴリズムを設計する上で重要となる．スレッド数が SM 内部のコア数よりも多い場合は，1 つのコアが複数のスレッドを逐次実行する．

3.2 コアレスアクセス

GPU の性能を可能な限り発揮するためには，最も基本的な技術としてコアレスアクセスと呼ばれるグローバルメモリへのアクセス方法が重要になる．コアレスアクセスを図 5 に示しこれを用いて説明する．コアレスアクセスは，メモリへの並列アクセス方法である．メモリの帯域幅を最大限に活用するために，SM 内の複数のスレッドが同期してグローバルメモリの連続するメモリ領域にアクセスする．コアレスアクセスを行うと，SM 内の複数のスレッドは同時にメモリの読み込みや書き込みを行うことができ，これ

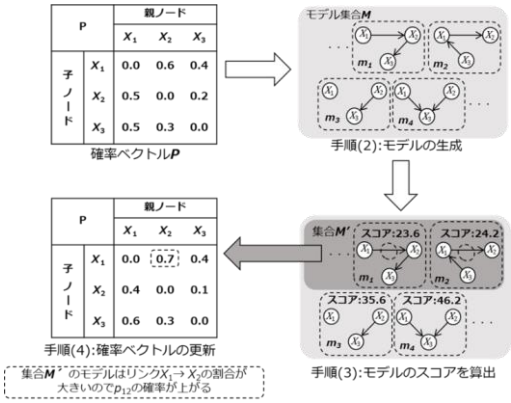

図 3：PBIL の概要

を利用することが GPU を活用するアルゴリズムを設計する上で重要となる．

4. GPU による高速なベイジアンネットワーク学習

4.1 概要

本論文では，PBIL に基づくベイジアンネットワーク学習アルゴリズム(PBIL-RS)を GPU による並列計算で高速化する方法を提案する．2.2 節で示した PBIL に基づくベイジアンネットワーク学習手順(3)，つまり全てのモデル $m \in M$ に対するスコアの算出に着目し，GPU で並列処理できるように再度設計する．PBIL に基づく学習アルゴリズムは，1 世代で数百から数千のモデルを生成し，全てに対してスコアを計算する．そのため，手順(3)を並列計算することが高速化において非常に効果的である．

再度設計した手順(3)を具体的に以下に示す．

(3a) CPU からグローバルメモリへデータを転送する．

(3b) それぞれのモデル $m \in M$ に対して手順（3b-1）と（3b-2）を行うことでモデルのスコアを算出する．

(3b-1) 観測データ中にある，それぞれの値のパターンの出現数をカウントする．

(3b-2) 求めたカウント数よりモデルのスコアを算出する．

(3c) (3b)より求めたそれぞれのモデル $m \in M$ のスコアを CPU に転送する．

手順（3b）では，まず始めに観測データ O 中にある，それぞれの値のパターンの出現数をカウントする．ここで値のパターンとは，あるノードの値とその親ノードの値からなる値の集合である．例えば，ノード X_1 と X_2 を親ノードとするノード X_3 の値のパターンは，自身のノード X_3 の値とその親ノード X_1 と X_2 の値の組合せとなる．ノード X_1，X_2，X_3 の値が 0 か 1 をとるとすると，値のパターンは，（X_1，X_2，X_3）=（0,0,0），（0,0,1），…，（1,1,1）となり，全部で 8 通りである．一般に，ノード X_i の親ノード数を r，親ノード集合を $pa(X_i) = \{p_1, p_2, \cdots p_r\}$ で表すと，ノード X_i の値のパターンの集合は $V_i = p_1 p_2 \cdots p_r X_i = \{0..00, 0..01, \cdots, 1..11\}$ と表せる．

その後（3b）では，全てのノード $i \in N$ に対する全ての値のパターン $v \in V_i$ の出現数 C_{iv} よりモデルのスコアとなる情報量基準を計算する．ベイジアンネットワークモデルの構造を学習する際，モデルのスコアとして AIC や BIC，MDL などの情報量基準がよく用いられる．そして，それら情報量基準は 4.3 節で示すように観測データ中の値のパターンの出現数 C_{iv} より計算される．

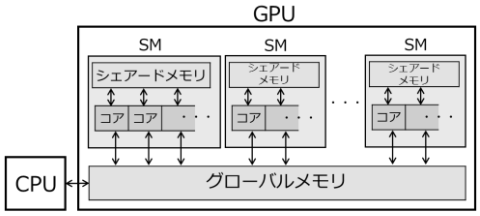

図 4：GPU の基本構造

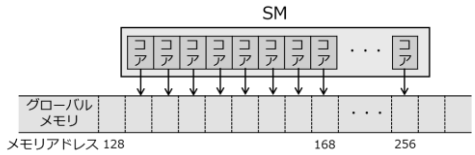

図 5：コアレスアクセス

モデル $m \in M$ に対し 1 つの SM を割り当て，モデル m の全てのノードに対する全ての値のパターンのカウントを SM 内のコアを用いて並列に行う．そして，その SM でモデル m のスコアを計算する．つまり，GPU 内の各 SM がすべてのモデル $m \in M$ に対して 1 つずつ並列にモデルのスコアを計算することで高速化する．以下の節では，モデル m に対する，コアレスアクセスを用いた効率的な SM の処理アルゴリズムを記述する

4.2 データ構造

手順（3a）では，モデルのスコアを計算するために必要なデータをグローバルメモリに転送する．グローバルメモリに転送するデータを以下に示す．

i. 観測データ O
ii. モデル集合 M
iii. 計算されたモデルのスコアを格納する配列

これらのデータを定義するための疑似コードを以下に示す．

```
u_int8_t observation[N][O];
boolean model[M][N][N];
float modelScore[M];
```

ここで変数 `M`，`N`，`O` は$|M|$，$|N|$，$|O|$を表す．観測データ O を 2 次元の配列 `observation` で表し，1 次元目をノード（事象），2 次元目を各ノードのサンプル（観測）とする．モデル集合 M を 3 次元の配列 `model` で表し，1 次元目をモデル $m \in M$ の番号，2 次元，3 次元目をモデル m の構造とする．モデルの構造は隣接行列で表現でき，モデル m でノード i からノード j にリンクがあるとき，`model[m][i][j]` が真値をとる．各モデルのスコア配列 `modelScore` は，各モデルの計算されたスコアを保持する．

あるモデルの全てのノードに対する全ての値のパターンのカウント数を保持するために，各 SM 内部に搭載されているシェアードメモリには，以下のカウント配列を用意する．

```
u_int16_t counter[Vi];
```

ここで，V_i はノード $i \in N$ における値のパターンの数$|V_i|$を示す．V_i は，ノード i の親ノードの数とそれらのノードが取り得る値の数に依存して決定され，ノード i の取り得る

値の数を $w(i)$とすると，$V_i = w(i)w(p_1)w(p_2)\cdots w(p_r)$と求められる．$V_i$によりカウント配列 `counter` がシェアードメモリの容量を超える可能性がある．`counter` がシェアードメモリに確保できるなら，4.4 節に示す高速なカウントアルゴリズム Case-1 を用い，確保できないなら，4.5 節に示す代替アルゴリズム Case-2 を用いる．

4.3 モデルのスコア

モデルのスコアを計算するために，それぞれの値のパターンの出現数をカウントする．ベイジアンネットワークモデルの構造を学習する際，モデルのスコアとして AIC や BIC，MDL などの情報量基準がよく用いられる．例えば AIC は以下の式で計算される．

$$AIC = -2l(\theta|\boldsymbol{O}) + 2k \quad (2)$$

ここで$\boldsymbol{\theta}$は，パラメータ集合を示し，$l(\theta|\boldsymbol{O})$は観測データ O のパラメータに対する適合度を表している．k はペナルティ項であり，パラメータ数$|\boldsymbol{\theta}|$となる．また，$l(\theta|\boldsymbol{O})$は以下の式で近似できる．

$$l(\theta|\boldsymbol{O}) \propto \sum_{i\in N}\sum_{v\in V_i}(C_{iv}) \log \theta_{iv} \quad (3)$$

i はノード，v は 1 つの値のパターンで，C_{iv}は i と v に合致する観測データ O 中の出現数を示す．θ_{iv}は C_{iv}から計算されるパラメータである．つまり，モデル m のスコアとなる AIC は，それぞれの値のパターンの出現数をカウントすることで計算できる．モデルのスコアとして用いられる BIC や MDL なども，同様に観測データ O 中の値のパターンの出現数をカウントすることで計算できる．

4.4 モデルのスコア計算(Case-1)

カウント配列 `counter[Vi]`がシェアードメモリの容量で収まる場合，この節で説明するアルゴリズム（Case-1）を適用する．モデル $m\in M$ のスコアを計算するために，SM は全てのノード $i\in N$ に対する全ての値のパターン $v\in V_i$ の C_{iv} を計算する必要がある．提案アルゴリズム（Case-1）では，ノード $i\in N$ の全ての値のパターン $v\in V_i$ の C_{iv} を計算することに着目する．

C_{iv} を計算するための戦略として，SM 内部の複数コアを用いて観測データを並列に処理する．1 つのスレッドが観測データ O の 1 つのサンプルを読み込み，`counter` の対応する領域の値を増分する．各ノード（X_i，p_1，p_2，…，p_r）のサンプルのアドレスは配列 `observation` 内で連続している．よって，1 サンプルに 1 スレッドを割り当てることで，観測データ O にコアレスアクセスで高速にアクセスできる（図 6 に示す）．

上記の手順を全てのノード $i\in N$ に行うことで，全てのノード $i\in N$ に対する全ての値のパターン $v\in V_i$ の C_{iv} を計算する．C_{iv} の計算後は，モデル m のスコアを計算する．このモデルのスコア計算は，式（2）に従って並列に行う．その際，コアに各ノード $i\in N$ を割り当て，並列リダクションと呼ばれる総和手法を用いて，シェアードメモリ上でモデルのスコアを合算する．最後に算出したモデルのスコアをグローバルメモリ上の配列 `modelScore` に格納する．

4.5 モデルのスコア計算(Case-2)

カウント配列 `counter[Vi]`がシェアードメモリの容量を超える場合，この節で説明するアルゴリズム（Case-2）を適用する．本論文で評価に用いた各ノードが最大 4 つの値を取る Alarm network[10]では，48KB のシェアードメモリを用いた際，親ノード数 r が 6 つ以下の時のみ Case-1 アルゴリズムを使用できる．

Case-2 アルゴリズムでは，カウント配列 `counter[Vi]`の代わりに以下に示す配列をシェアードメモリに用意する．

```
u_int16_t patternValue[O];
```

単純に，観測データ O に出現する値のパターンをこの配列 `patternValue[O]`に格納する．Case-1 と同様の方法でコアレスアクセスを使用して，グローバルメモリから値のパターンを取得する．そして，シェアードメモリの配列 `patternValue[O]`に値のパターンを格納した後，図 7 に示すように `patternValue` をソートする．ソートはバイトニックソートと呼ばれる GPU 特有のソートアルゴリズムを用いて高速に行う．その後，配列 `patternValue` を順番にたどりながら，それぞれの値のパターンをカウントし，モデルのスコアを合算する．Case-2 は Case-1 よりも少し時間がかかるが，比較的早くモデルのスコアを計算できる．

5. 評価

本論文では，提案手法の実行時間を評価する．計算速度を高速化する提案手法の性能を明確にするため，GPU 上で実行する提案アルゴリズムを用いた PBIL-RS の実行時間と，CPU で実行する PBIL-RS 実行時間を比較する．両者はベイジアンネットワークモデルの学習の結果は同じで，実行時間のみが異なる．アルゴリズムは，C++で実装し，GPU 上の提案アルゴリズムは CUDA ライブラリを利用した．詳細な実行環境は表 1 に示す通りである．評価に用いたベイジアンネットワークモデルは，Alarm network[10]と呼ばれる評価によく用いられる 37 ノードのモデルである．この Alarm network の各ノードが持つ条件付確率より 1024 サンプルの観測データを作成する．そして，作成した観測データより，比較対象のアルゴリズムを用いて再度ベイジアンネットワークを学習する．

5.1 実行時間の結果

図 8 は，それぞれの世代数までにかかるベイジアンネットワークモデルの学習の実行時間を示す．GPU を用いる提案手法は，CPU のみで実行する PBIL-RS よりも約 14 倍高速である．また，親ノードが少ない場合のみ使用できる Case-1 の代わりに，常に Case-2 を実行した結果を“Case-2 only”として図 8 に示す．Case-1 は Case-2 よりも高速なため，“Case-2 only”は，Case-1 と Case-2 を用いる提案手法（GPU）よりも 1.6 倍遅い．そして，より正確な速度差を確認するため，各世代における Case-1 と Case-2 の実行率を図 9 に示す．Case-1 は，ほぼ 100%の実行率であるため，Case-1 は Case-2 よりも 1.6 倍高速であると推測できる．

6. おわりに

本論文では GPU を用いて，PBIL に基づくベイジアンネットワーク学習アルゴリズムを高速化する方法を提案した．提案手法は，GPU 特有のアクセス方法コアレスアクセスを活用することで，計算時間を削減した．そして評価の結果，我々は CPU のみで実行する PBIL-RS に比べ約 14 倍の高速化に成功した．

参考文献

[1] D.M. Chickering, D. Heckerman, C. Meek, “Large-Sample Learning of Bayesian Networks is NP-Hard,” Journal of Machine Learning Research, Vol . 5, pp.1287-1330 , (2004).

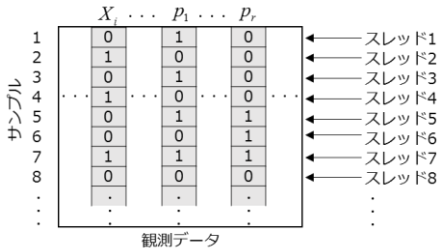

図 6：値のパターンのカウント(Case-1)

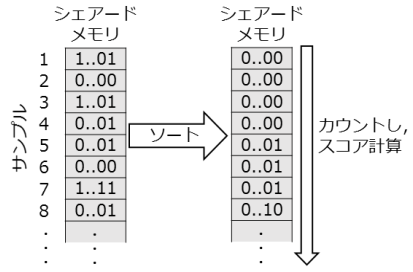

図 7：値のパターンのカウント(Case-2)

表 1：実行環境

OS		CentOS 5.0
CPU		Intel Core i7 4770k (3.50GHz)
メモリ		32Gbytes
GPU	Model	GeForce GTX TAITAN Black (0.98 GHz)
	SM	15
	Core / SM	192 (2880 Cores in total)
	global memory	6143 Mbytes
	shared memory per block	49152 bytes
プログラミング言語		CUDA, C++
コンパイラ		CUDA6.0, gcc 4.1.2

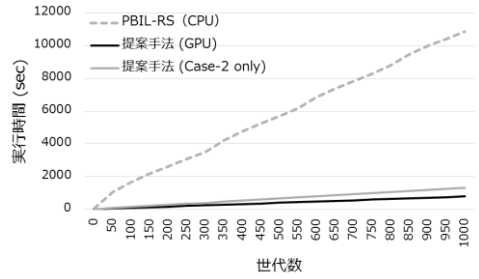

図 8 実行時間(CPU vs GPU)

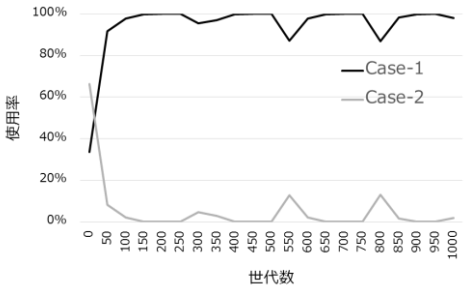

図 9：Case-1 と Case-2 の割合

[2] G. F. Cooper, and E.. Herskovits, ``A Bayesian Method for the Induction of Probabilistic Networks from Data," Machine Learning, Vol.9, pp.309--347 (1992).

[3] P. Larranaga, C.M.H. Kuijpers, R.H.Murga, and Y.Yurramendi, ``Learning Bayesian Network Structures by Searching for the Best ORdering with Genetic Algorithms," IEEE Transactions on Systems, Man, and Cybernetics, Vol.26, No.4 (1996).

[4] N.B.Asadi, T.H.Meng, and W.H.Wong, ``Reconfigurable Computing for Learning Bayesian Networks" In Proc. of FPGA'08, pp.203--211 (2008).

[5] I.Tsamardinos, L.E.Brown, C.F.Aliferis, ``The max-min hill-climbing Bayesian Network structure learning algorithm,"Machine Learning, Vol.65, Issue.1, pp.31--78 (2006).

[6] A.S.Hesar, ``Structure Learning of Bayesian Belief Networks Using Simulated Annealing Algorithm, "Middle-east journal of Scientific Research, Vol.18, No.9, pp.1343--1348 (2013).

[7] S. Baluja, ``Population-Based Incremental Learning: A method for integrating genetic search based function optimization and competitive learning,'' Technical Report CMU-CS-94-163, Carnegie Mellon University (1994).

[8] D.W.Kim, S.Ko, and B.Y.Kang, "Structure Learning of Bayesian Networks by Estimation of Distribution Algorithms with Transpose Mutation, " Journal of Applied Research and Technology, Vol.11, pp.586– 596 (2013).

[9] Yuma Yamanaka, Takatoshi Fujiki, Sho Fukuda, and Takuya Yoshihiro, ``PBIL-RS: An Algorithm to Learn Bayesian Networks Based on Probability Vectors," International Workshop on Informatics (IWIN2015), 2015.

[10] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, G.F. Cooper : The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In: Second European Conference on Artificial Intelligence in Medicine, Vol. 38, pp.247–256 (1989).

[11] H. Akaike, "Information theory and an extension of the maximum likelihood principle," Proceedings of the 2nd International Symposium on Information Theory, pp.267-281 (1973).