社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2009-SLDM-138 (34)
2009 / 1 / 30

高速モチーフ探索を目指したGibbs Sampling法のFPGAによる実装

佐藤 由香† 田沢 純子† 宮崎 敏明†

†会津大学 〒965-8580 福島県会津若松市一箕町鶴賀
E-mail: †{m5111110, s1140139, miyazaki}@u-aizu.ac.jp

あらまし 遺伝子を構成する複数の塩基配列から，共通のモチーフ（小さな部分配列）を探索することは，バイオインフォマティクス分野で強く求められ，幾つかの手法が提案されている．筆者らは，その手法の一つとしてよく知られたGibbs sampling法のハードウェアによる高速化を目指し，最適なモチーフを検出するための評価値の計算方法を再考した．また，その結果を元に，FPGAに実装する事を試みた．本稿では，そのアーキテクチャの概要を示し，現状を報告する．

キーワード モチーフ検出，バイオインフォマティクス，Gibbs Sampling，FPGA 実装

An FPGA implementation of Gibbs sampling method towards high-speed motif search

Yuka SATO† Junko TAZAWA† and Toshiaki MIYAZAKI†

†University of Aizu Tsuruga, Ikki-machi, Aizu-wakamatsu City, Fukushima, 965-8580 Japan
E-mail: †{m5111110, s1140139, miyazaki}@u-aizu.ac.jp

Abstract It is very important to detect a common motif, i.e., partial base sequence, from DNA sequences in bioinformatics research field, and some methods have been proposed. Gibbs sampling is one of the popular methods to perform it. To accelerate Gibbs sampling with a dedicated hardware, we reconsider the way of the evaluation-value calculation part that is the performance bottleneck of Gibbs sampling. In this paper, after discussing the evaluation-value modification, an FPGA implementation and its current status are introduced.

Keyword Motif detection, Bioinformatics, Gibbs Sampling, FPGA implementation

1. はじめに

1.1. 背景

遺伝子配列モチーフとは，PROSITE[1]やPfam[2]などで見られる"A", "T", "G", "C"という塩基で構成された複数の遺伝子配列に現れる局所的な共通もしくは類似部分配列のことである．そのような部分配列は，立体構造の形成や機能に影響を与えるために，変化を許容できなかった重要な場所と考えることができる．したがって新規配列に特徴的なモチーフが見つかると，そこから機能予測やファミリーの同定をすることができるため，モチーフを検出する事は非常に重要である．これまで，データベース，ヒューリスティック，動的計画法などを用いた検索法が，幾つか提案されてきた．データベースを用いた検索ツールには，FASTA[3]，BLAST[4]，SSEARCH[5]がある．ヒューリスティックなアルゴリズムには，与えられた配列から固定長のモチーフをそれぞれ1つずつ見つけ出すMetropolis-HastingやGibbs Samplingがよく知られており，Gibbs Samplingを使ったツールにはAlignACE[6]がある．

また，動的計画法に基づく手法ではSmith-Waterman法[7]があり，Smith-Waterman法を用いてデータベース検索を行うSSEARCH[5]などが有名である．上述した方法は，これまで汎用計算機を用いたソフトウェアとして実現されているが，計算量・計算時間が莫大なため，より高速なモチーフ探索法が求められている．

本稿では，モチーフ探索によく使われるGibbs Samplingの高速化を目指し，FPGAに実装するハードウェアアーキテクチャを提案する．また，Gibbs Samplingを用いたソフトウエアツールであるAlignACEとの性能比較も試みる．

1.2. 関連研究

文献[8]では，Gibbs Samplingの収束性を向上させる手法を提案している．具体的には，Genetic Algorithmの世代交代モデルであるMinimam Generation Gapと，

自己組織化臨界モデルを用いた発見的検索手法であるExtremal Optimizationという2つの最適化手法をGibbs Samplingに導入し，性能評価を行っている．ただし，Gibbs Sampling自体の高速化を目指したものではない．また，Gibbs Sampling以外の方法でモチーフを検出する研究としては，Expectation Maximizationを用いたMEME[9]などがあるが，アルゴリズムの収束が非常に遅いという欠点がある．

2. Gibbs Sampling

Gibbs Samplingは，元々ノイズのある画像から元画像を推定する方法として提案されたものであり，マルコフ連鎖モンテカルロ法の解法であるMetropolis-Hastin法の異種にあたるものである．Gibbs Samplingはサンプルの推移が必ず生じるMetropolis-Hasting法であるともいえる．文献[10]は，Gibbs Samplingを最初にモチーフ探索に導入した研究であり，その概要は以下の通りである（図1も参照）．Gibbs Samplingの入力は，N本の長さLの塩基配列で，それぞれの塩基配列から長さLのモチーフを検索する．

1. N本の配列それぞれのモチーフ候補の開始位置の初期値Site = {$Site_1$, $Site_2$, $Site_3$, ... , $Site_N$}を決定．
2. N本の配列からひとつの配列Mをランダム（または順次）に選択．
3. 配列M以外の残りのN-1本の配列について，手順1で決定したモチーフ候補の各位置i={1, ..., L}での文字j={A, T, G, C}の出現頻度$A_{i,j}$を計算．
4. 各配列のモチーフ候補に含まれていない部分から，各文字jの背景出現頻度$P_{i,j}$を計算．
5. 配列Mで，各位置iをモチーフ候補の更新候補の開始位置とし，それぞれのモチーフ候補の更新候補について評価値U_iを$A_{i,j}$とP_jを用いて計算．
6. 手順5で計算した{U_1, U_2, ... , U_N}の各評価値から，ランダムにU_kを選択し，配列Mのモチーフ候補の開始位置をkに更新する．ただし，評価値はできるだけ大きな値が選ばれるものという条件がある．

収束するまで，上記した手順2～6を繰り返す．

ここで，評価値U_kの計算は下記に従い行う．配列長8で，モチーフ長4，配列MをATTAGCCCとすると，配列Mの位置1からの出現頻度A_1を，$A_{1,A}\times A_{2,T}\times A_{3,T}\times A_{4,A}$と，背景出現頻度$P_1$を$P_A\times P_T\times P_T\times P_A$と計算することができ，$U_1$は$A_1$と$P_1$を用いて$A_1/P_1$となる．この方法で，取り得るすべての位置から$U_k$を計算することが出来る．

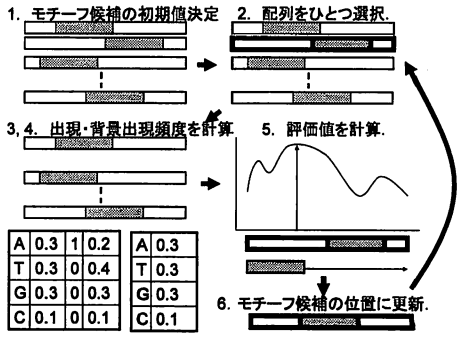

図1． Gibbs Samplingによるモチーフ検索．

3. 評価値の吟味

Gibbs Samplingでは，評価値を求める際に出現頻度A_1と背景出現頻度P_1を用いて$U_1=A_1/P_1$とする．これは，対象となる塩基配列の構成要素である“A”, “T”, “G”, “C”のうち全体の出現頻度が高いだけのものが検出されないように，モチーフ候補以外の背景出現頻度で除算することで正規化している．しかし，除算は，FPGA実装において非常に実装コストが高く，面積も実行ステップも多くなる．そこで背景出現頻度による正規化が，結果にどの程度影響を与えているか，入力の配列の長さ，構成要素，モチーフ長の長さをさまざまな値に変更して実験を行った．

表1．実験条件

	Length of sequence	Length of Motif
State 1	1,000	4/8/16/32
State 2	100	
State 3	10	

実験では，同一入力データを用い表1に示した条件下で，1）前述したGibbs Samplingのアルゴリズムで出現頻度と背景出現頻度から評価値を算出し，モチーフ候補値位置を更新する方法と，2）背景出現頻度を用いず出現頻度を評価値として利用し，モチーフ候補値位置を更新する方法を個別に実行し，各条件下で得られたそれぞれのモチーフ候補値を比較した．ここで2つの値に有意な違いがあれば正規化の意味はあると認められる．反対に，違いが認められなければ，正規化の影響は少ないため正規化の工程を省くことができる．図2および図3は，条件（State3, even, 16）での実験

の結果を示している．図２は，本来の評価値による結果である．図３は，出現頻度を背景出現頻度で正規化せず，そのまま評価値として用いた場合である．各図の横軸は塩基配列のすべての部分配列を示しており，縦軸はその出現頻度と評価値である．

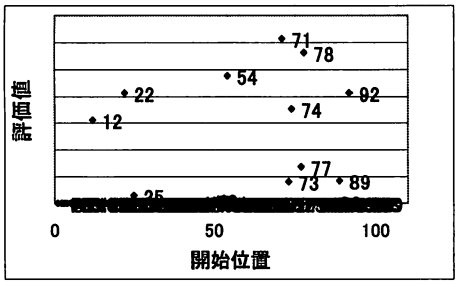

図２．条件（State3, even, 16）おけるモチーフ開始位置．本来の評価値を用いた場合．

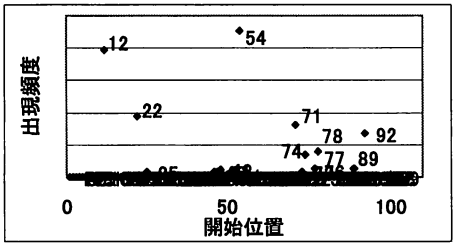

図３．条件(State3, even, 16)におけるモチーフ開始位置．出現頻度を用いた場合．

また，グラフ中の各点のラベルは開始位置を示している．なお，実験の結果，すべてのState，他条件で，図２および図３で示したものと同様の結果が得られた．これらの図から分かるように，本来の評価値と出現頻度の値自体は異なるものの，その値が大きくなるモチーフの開始位置に差異はない．降順に並べた際の順番は異なるが，アルゴリズムのランダム性からその点は許容できる．また，前述したアルゴリズムから，ある程度評価値が大きなものから次のモチーフ候補値を決定するので，上位20個ずつのモチーフ開始位置を抽出して，t検定を行ったところ全ての実験で差異はなかった．

この結果から，今回FPGAに実装するにあたり，背景出現頻度の計算およびそれによる正規化（除算）の工程を省略し，乗算のみで求まる出現頻度を評価値として用いることとした．

4. ハードウェアアーキテクチャ

図４に今回提案するハードウェアアーキテクチャの概要を示す．

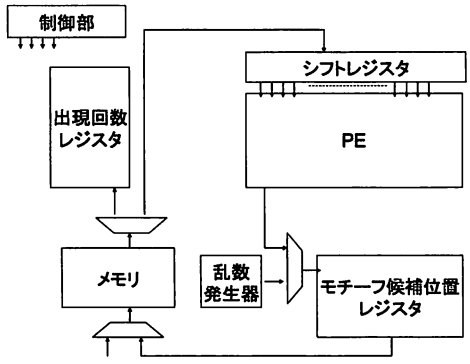

図４．ハードウェアアーキテクチャ概要

アーキテクチャは，評価値を計算し次のモチーフの候補位置を出力するPE (Processing Element)，PEの入力に接続されるシフトレジスタ，各配列のモチーフ候補位置を保持するモチーフ候補位置レジスタ，全ての配列のモチーフと各行の出現回数を保持する出現回数レジスタ，乱数生成器，メモリ，制御部からなる．

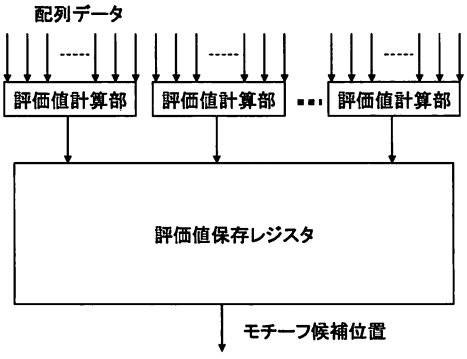

図５．PEの概要

PEは，図5に示すように，並列で評価値の計算を行う複数個の計算部と配列のすべての評価値から上位20個の値を保存するレジスタからなる．今回，プロトタイプをFPGAに実装するにあたり，評価値計算部は4個実装することとした．PEの入力に接続されるレジスタは，メモリから値を入力するモードと，レジスタからレジスタへ同じ方向へデータを転送することでシフトレジスタを構成するモードがある．メモリからのデータをシフトレジスタに保持し，その値を用いてPEで評価値を計算する．例として，最初に評価値が並列計算される部分配列は，位置0から7までの部分配列，位置8

から15までの部分配列,位置16から23までの部分配列,位置25から31までの部分配列の4つである.次にデータをシフトして,同様に評価値の計算を行う.例で,次に評価値が計算される部分配列は,位置1から8までの部分配列,位置9から16までの部分配列,位置17から24までの部分配列,位置25から31までの部分配列の4つとなる.同様に,データをシフトさせて全ての部分配列の評価値を計算し,評価値の上位20個を保持し最終的に20個の中からランダムに次のモチーフの位置を決定する.

評価値の計算には2章で説明したとおり,各配列のモチーフの各行の"A", "T", "G", "C"の出現頻度を参照する.提案アーキテクチャでは,モチーフ自体とその出現頻度を保持する出現回数レジスタを設置することで,各行のモチーフ候補位置が更新されるごとに対応するレジスタの内容だけを書き換え,他の配列のモチーフを再度計算し直す事なく出現頻度を計算することができる.

乱数発生器には,線形帰還シフトレジスタ(LFSR, Liner Feedback Shift Register)を用いた.LFSRは,レジスタで構成することが可能であるが,レジスタの動作は決定的であるため,レジスタが生成する値はその状態によって完全に決定される.同様にレジスタの取り得る状態は有限個であるため,最終的に周期的動作になる.しかし,帰還関数を適切に設定したLFSRは乱数のようなビット列を生成し,その周期も非常に長い.ここでは,文献[11]で示された最適な帰還関数を用いた.

N本の塩基配列から長さ8のモチーフを検出する際の実行ステップと実行クロック数を以下に示す.

1. モチーフ候補位置の初期値を決定.LFSRは1クロックで1個の乱数を生成するため,実行クロック数はNクロック.
2. モチーフ候補を更新する配列を選択.実行クロック数は1クロック.
3. モチーフ候補をメモリから出現回数レジスタに読み出し,同時に出現頻度を計算.実行クロック数はNクロック.
4. 評価値を計算.シフトレジスタの深さがm,並列に計算する乗算器を1個設置するとし,塩基配列が長さLとすると,計算とシフトレジスタに値を保存するクロック数を足して,実行クロック数は(L/l+L/m)クロック.
5. モチーフ候補値を決定し,モチーフ候補位置レジスタに保存.実行クロック数は1クロック.
6. 手順2に戻り繰り返す.

5. FPGA実装

4章で述べたアーキテクチャの実現可能性を検証するために,プロトタイプをFPGA (StratixIII EP3SE110F780C4) 上に実装した.実装したプロトタイプは,4章でも述べたとおり長さ8のモチーフを探索するものである.

表2はプロトタイプをFPGA上に実現したときのFPGA利用率を示している.また,本プロトタイプは48.17MHzで動作する.

表2. FPGA実装結果

ブロック名	LC数	DSP Elements*	LC %
PE	1,845	112	2.16
シフトレジスタ	4,058	0	4.76
メモリ	13,368	0	15.69
モチーフ位置レジスタ	197	0	0.23
出現回数レジスタ	4,254	0	4.99
乱数発生器	3	0	<0.01
データパス全体	22,699	112	26.64

1 DSP element = 9bitx9bit組み込み乗算器

6. 評価

今回実装したプロトタイプシステムは,N個の塩基配列からそれぞれ1つずつ,長さ8のモチーフを検出する.また,搭載している乗算器は,整数演算のみをサポートしているため,本来小数となる出現頻度を,スケーリングして,整数値となるように計算を行っている.使用した入力データは,"A", "T", "G", "C"をランダムに発生させて人工的に作成した長さ108,配列数20のデータ(1つずつ共通部分配列を含む)と,実応用データの塩基配列(YBR018C gal7とYBR019C gal10,および,それらの逆方向の配列,計4本)を用いた.YBR018C gal7の塩基配列の長さは500, YBR019C gal10のそれは655である.結果は,AlignAceと比較した.AlignAceの実行には,Dell Precision PWS670 Intel® Xeon™ CPU 3.00 GHz 2.99GHz, 1.00GB RAMを用いた.実装したプロトタイプシステムの検証は,Cadence社NC-Verilogシミュレータを用いて行なった.データが収束しない場合,AlignAceと同様に,初期値を再設定して実行している.実行に要した全てのクロック数に,5章で示した48.17MHzを乗じて実行時間を算出した.Gibbs Samplingは,ランダム値を用いるアルゴリズムであるため,100回の実行結果の平均を出した.結果を表3に示す.

表3. 100回の実行結果の平均

データ種類	AlignAce	本アーキテクチャ
人工データ	4.15 s	0.58 ms
実応用データ	10.93 s	16.09 ms

表から分かるように，本プロトタイプシステムは，AlignAceで実行するよりも長さ108の人工データの場合，約7000倍，長さ約500の実応用データの場合，約680倍高速である．本アルゴリズムでは，塩基配列のすべての部分配列位置の評価値の算出時間が実行時間のボトルネックとなる．したがって，塩基配列長が短く，しかも共通部分配列をもつ人工データの方が，収束し易くより高速化されている．

7. まとめ

塩基配列中の共通部分配列であるモチーフを高速に検出することを目指し，PEが評価値を並列計算するアーキテクチャを提案した．本アーキテクチャの考え方は，他のマルコフ連鎖モデルを用いる問題などに広く適用可能である．今回，N個の塩基配列から，長さ8のモチーフを1つずつ検出するプロトタイプをFPGAに実装した．本アーキテクチャでも，ソフトウェア実装に比べて約680倍高速化できることがわかった．

今後，ハードウェアによるGibbs Samplingのさらなる高速化を検討する．近年，交換モンテカルロ法というモンテカルロ法にシミュレーテッドアニーリング法のように温度の概念を導入し，温度の異なるGibbs Sampling自体を並列に実行するという新たな手法が提案されている[12]．交換モンテカルロ法が内包する並列性に着目したFPGAによる高速化手法を検討していく予定である．

謝　辞

本研究を進めるにあたり，協力いただいた 会津大学 五十嵐裕之氏に心より感謝いたします．

文　献

1 PROSITE, http://au.expasy.org/prosite/

2 Pfam, http://pfam.sanger.ac.uk/

3 FASTA, http://fasta.genome.jp/

4 BLAST, http://blast.ncbi.nlm.nih.gov/

5 SSEARCH, http://ssearch.ddbj.nig.ac.jp

6 Roth, F. P., Hughes, J. D., Estep, P. W. & Church, G. M, "Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation," Nat Biotechnol, Vol.16, No. 10, pp.939-45 , 1998.

7 Smith TF, Waterman MS, "Identification of Common Molecular Subsequences," Journal of Molecular Biology, Vol.147, No.1, pp. 195-197, 1981.

8 河野修久,加藤智之,田村慶一,北上始, "配列データベースから類似部分配列を抽出するためのGS最適化手法に関する考察," DEWS2008, 2008.

9 Bailey, T. L. & Elkan, C. "Fitting a mixture model by expectation maximization to discover motifs in biopolymers," Proc Int Conf Intell Syst Mol Biol, Vol.2, pp.28-36 , 1994.

10 Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F., Wootton, J.C. "Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment," Science, Vol.262, No.5131, pp. 208-214, 1993.

11 Peter Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators," http://www.xilinx.com/bvdocs/appnotes/xapp052.pdf, 1998

12 K. Hukushima and K. Nemoto, "Exchange Monte Carlo Method and Application to Spin Glass Simulations," J. Phys. Soc. Jpn. Vol.65, No.6, pp. 1604-1608, 1996.