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

河野 修久† 加藤 智之‡ 田村 慶一‡ 北上 始‡

†広島市立大学情報科学部 ‡広島市立大学大学院情報科学研究科
〒731-3194 広島市安佐南区大塚東 3-4-1
E-mail: †kono@db.its.hiroshima-cu.ac.jp, ‡kato@db.its.hiroshima-cu.ac.jp,
{ktamura,kitakami}@its.hiroshima-cu.ac.jp

あらまし 配列データマイニング処理で抽出される配列パターン数を低減する方法として，著者らは，ギブスサンプリング(GS)を用いて，配列データベースから類似部分配列を予め取り出し，配列データマイニング処理の入力データのサイズを小さくする方法を既に提案してきた．しかしながら，GS は収束性が必ずしもよいとは限らない．そこで本論文では，GS に最適化処理を応用した 2 つの GS 最適化手法を提案する．1 つめの方法は，Minimal Generation Gap を応用して，初期収束や局所解を回避させる．2 つめの方法は，Extremal Optimization を応用して，類似部分配列集合の中から最も適していないと考えられる部分配列の検出位置を繰り返し更新していく．評価のために，Leucine Zipper モチーフを含むデータセットを用いて類似部分配列を抽出して収束性を調べ，比較を行ったので，その結果について報告する．

キーワード テキストマイニング，科学 DB，バイオインフォマティクス，データマイニング，情報抽出

GS Optimization Technique for Extracting Similar Partial Sequence from Sequence Database

Nobuhisa KONO† Tomoyuki KATO‡ Keiichi TAMURA‡ and Hajime KITAKAMI‡

†Faculty of Information Science, Hiroshima City University
‡Graduate school of Information Science, Hiroshima City University
3-4-1 Ozuka-Higashi, Asa-Minami-Ku, Hiroshima, 731-3194 Japan
E-mail: †kono@db.its.hiroshima-cu.ac.jp, ‡kato@db.its.hiroshima-cu.ac.jp,
{kitakami,ktamura}@its.hiroshima-cu.ac.jp

Abstract In order to reduce the number of the sequential patterns extracted by sequence data mining, we have already proposed a method to decrease the size of input data for the sequence data mining in which similar subsequences are found from a sequence database using Gibbs Sampling (GS) beforehand. However, the convergence of this method is not necessarily better. In this paper, we propose two GS optimization methods that are applied to GS respectively. One of them avoids premature converge and the local optimum by applying Minimal Generation Gap to GS. Another replaces iteratively the position of the most unsuitable subsequence in the set of similar subsequences by applying Extremal Optimization to GS. We report experimental results that our proposed method was evaluated by using a dataset including the Leucine Zipper motif.

Keyword Text Mining, Scientific DB, Bioinformatics, Data Mining, Extracting of Information

1. はじめに

配列データベースから頻出なパターンを抽出する手法は，DNA やアミノ酸などの生物配列データのモチーフ抽出などの多くの問題解決に有用であるといわれている．モチーフとは，PROSITE[1]や Pfam[2]などで見られる生物学的に重要な機能を持つ特徴的なパターンである．アミノ酸配列や，テキスト情報などを含むデータベースに対する配列データマイニングでは，固定長や可変長のワイルドカード領域を持つ頻出配列パターンを抽出する手法[3][4][5]の研究が進められてきた．パターン成長アプローチにワイルドカード領域を抽出する機能を組み込んだ固定長パターン抽出法[3]や可変長パターン抽出法[4]，ワイルドカード領域の極小化や冗長性除去を可能とし，正確な表現の頻出パターンの

抽出ができるスコープデータベース[5]などがそれに該当する．しかしながら，どの手法においても，配列データベース全体から頻出パターンを抽出しようとすると，明らかに不要と思われるパターンが大量に抽出されてしまう．そこで，抽出される配列パターン数を低減する方法の研究が重要となっている．

著者らは，配列データマイニング処理で抽出される配列パターン数を低減する方法として，ギブスサンプリング(GS)[6][7]のアルゴリズムを用いて，配列データベースに含まれる各配列データから類似部分配列データを予め取り出し，配列データマイニング処理の入力データのサイズを小さくする方法[8]を既に提案してきた.しかしながら,通常のギブスサンプリング処理は，収束性が必ずしもよいとは限らないため，これに対する最適化処理が必要となっている．

本論文では，Genetic Algorithm(GA)の世代交代モデルの中で性能がよいとされている Minimal Generation Gap(MGG) [9]と，自己組織化臨界のモデルを用いた発見的探索手法である Extremal Optimization(EO)[10]という 2 つの最適化手法を導入し，それぞれの最適化手法をギブスサンプリングに応用した手法の性能を評価・比較する．

2. 関連研究

現在までに,GS 最適化手法に関する研究はほとんど行われていない．一般によく知られている最適化手法には，生物の進化の過程を真似て作られたアルゴリズムである GA[9]や，金属工学における焼きなましを元にした Simulated Annealing(SA)[11]，統計物理学での自己組織化臨界のモデルを用いた EO[10]などが存在する．

GA は，生物の進化の過程を真似て作られたアルゴリズムで，解空間構造が不明であり，決定的な優れた解法が発見されておらず，また，全探索が不可能と考えられるほど広大な解空間を持つ問題に有効である．GA は，解の候補を遺伝子で表現した個体を複数用意し，適応度の高い個体を優先的に選択して交叉・突然変異などの操作を繰り返し行うことで最適解を探す方法である．GA の問題点である初期収束を回避し，探索の後半で進化的停滞を抑制する手法として，MGG[9]と呼ばれる世代交代モデルが存在する．

SA[11]は,高温で加熱した金属の温度を徐々に下げて冷やすことにより，元の金属より欠陥の少ない優れた結晶構造を作る焼きなましを模倣した最適化手法である．SA は，局所解に陥りにくく様々な問題に適用できる反面，パラメータチューニングが容易でないという欠点をもつ．

EO[10]は統計物理学での自己組織化臨界のモデルを用いた発見的探索手法であり，GA と違い，1 つの解を用いて，解の中の最悪の構成要素を修正(更新)していくことで最適解を探す手法である．

本論文では，MGG と EO の 2 種類の方法を採用し，それぞれを GS の最適化に応用した結果について考察している．MGG を採用する理由は，近年提案された世代交代モデルの中で優れた世代交代モデルであることが知られていることによる．また，EO を採用する理由は，EO は GA とよく比較される SA に似た手法であり，かつ SA に比べて入力パラメータが少ないことによる．

3. 用語の定義

配列データベース $DB=\{t_1,t_2,\cdots,t_n\}$において，各配列は s_{sid} と表現される(n は配列数, sid は配列識別番号)．配列データベースの配列識別子の集合は $\Omega=\{1,2,\cdots,n\}$ と表現する．各 s_{sid} は，配列識別番号 sid をもつ配列データであり，あるアルファベット Σ 上で定義される文字列である.配列データ s_{sid} の先頭から j 番目の文字は，$s_{\mathrm{sid}}[j]$と表現する．

3.1. パターン

k-パターンとは，複数の配列データに共通に含まれている k-ストリング(k 個のアルファベットをもつ)からなる特定の集合に対する表現形式である．例えば，2-ストリングの集合{<*FK*>,<*F***K*>,<*F*K*>}を表現する 2-パターン<pat^2>は，<*F*-*x*(0,2)-*K*>と表現される．σ_i をアルファベット Σ の要素とすると，k-パターン<pat^k>は以下のように表現される．

$$<pat^k> = <\sigma_1\text{-}x(i_1,j_1)\text{-}\sigma_2\text{-}x(i_2,j_2)\text{-}\cdots\text{-}x(i_{k-1},j_{k-1})\text{-}\sigma_k>:cnt \quad (1)$$

cnt は支持数を表し，省略されることもある．ユーザが与えた最小支持数以上の支持数をもつパターンを頻出パターンと呼ぶ．

$x(i,j)$はワイルドカード領域であり，文字 σ_i と σ_{i+1} の間にワイルドカードが i 個から j 個含まれていることを表している．$i<j$ のとき，$x(i,j)$は可変長ワイルドカード領域と呼ばれ，$i=j$ のとき，$x(i,j)$は $x(i)$と簡略表現され，固定長ワイルドカード領域と呼ばれる．

3.2. k-部分文字列

配列データベースに対してギブスサンプリングを適用すると，各配列から指定した長さ k の部分文字列が抽出される．これを k-部分文字列と呼び，各配列の k-部分文字列を集合化したものを k-部分文字列集合と呼ぶ．また，各配列から抽出するために設定される，長さ k の部分文字列の候補領域をウィンドウと呼ぶ．また，ウィンドウの長さはウィンドウサイズと呼ぶ．

配列データベースと k-部分文字列集合の関係を図 1 に表す.

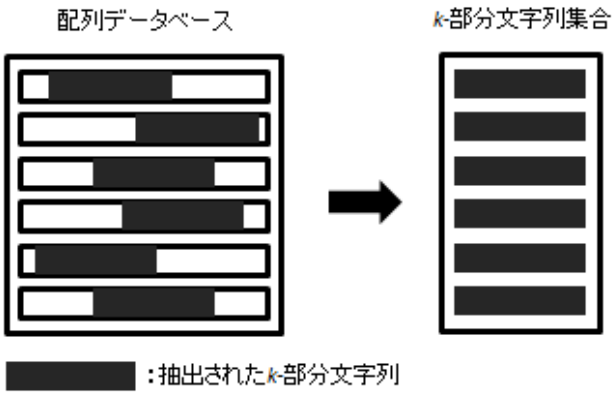

図 1 ：配列データベースと k-部分文字列集合の関係

3.3. 世代交代モデル

MGG などの世代交代モデルでは,対象となる問題中から，初期状態として予め複数個のデータの集合を作成する．このデータは個体と呼ばれ，データ集合を母集団と呼ぶ.

母集団の中から交叉のための親個体ペアを選択することを複製選択と呼ぶ．親個体ペアに対して，交叉や突然変異を行い作成された新たな個体を子と呼び，作成された子から次世代に生き残る (母集団に戻す) 個体を選択する操作を生存選択と呼ぶ．交叉とは複製選択で選ばれた 2 個体のデータの一部を入れ替えること意味し，突然変異とは，何らかの方法で個体の一部を変化させることである.

4. 類似部分配列の抽出

本章では，従来の手法である，ギブスサンプリングを用いて配列データベースから類似部分配列(k-部分文字列集合)を抽出する方法と,抽出した k-部分文字列集合の評価法について説明する.

4.1. ギブスサンプリング

ギブスサンプリングは，配列データベース DB の各配列から，指定した長さ k をもつ互いにできるだけ類似した部分文字列の集合(k-部分文字列集合)を求めることができる．ギブスサンプリングのアルゴリズムを図 2 に示す．また，以下に補足としてアルゴリズム中の評価値 U_m の計算方法について例を用いて示す.

ウィンドウサイズ 4 で，配列 Z を $CATTAGT$ とすると，配列 Z の位置 1 からの出現頻度 A_1 を，$A_{1,C}\times A_{2,A}\times A_{3,T}\times A_{4,T}$ と，背景的出現頻度 P_1 を $P_C\times P_A\times P_T\times P_T$ 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．この方法でとりえるすべての位置から U_m を計算する.

1. t 本の配列をもつ DB の各配列に対して，k-部分文字列の開始点 st_i をランダムに選び，それらを配列順に並べた集合を $S=\{s_1,s_2,\ldots,s_t\}$とする.
2. DB からランダムに 1 つの配列 Z を選択する.
3. 配列 Z 以外の残りの $t-1$ 本の配列から取り出される k-部分文字列集合から各位置 i での文字 j の出現頻度 $A_{i,j}$ を計算する.
4. DB の k-部分文字列集合に含まれない部分から各文字 j の背景的出現頻度 P_j を計算する.
5. Z 上の各位置 i を k-部分文字列の開始点とし,それぞれの k-部分文字列について評価値 U_i を $A_{i,j}$ と P_j を用いて計算する.
6. $\{U_1,U_2,\ldots,U_n\}$の各評価値の中から，ランダムに U_m を選択し，U_m に対応する k-部分文字列の配列上の新しい開始点 st'_m を選び，S を更新する．ただし，U_m はできるだけ値が大きいものが選ばれるものとする.
7. 収束するまで 2～6 を繰り返す.

図 2：ギブスサンプリングのアルゴリズム

4.2. k-部分文字列集合の評価

配列データベースから抽出した k-部分文字列集合を評価するために，以下の式(2)を用いる．式(2)は相対エントロピーを用いており，配列データベース内の，抽出した k-部分文字列部分の分布と，それ以外の部分の分布の差を計算したものとなっている．これ以降，式(2)を用いて算出される値を F 値と呼ぶ.

$$F=\sum_{i=1}^{k}\sum_{j=1}^{20}C_{i,j}\log\frac{Q_{i,j}}{P_j} \qquad (2)$$

P_j は，ギブスサンプリングでの背景的出現頻度 P_j と同様のものである $Q_{i,j}$ について式(3)に示す.

$$Q_{i,j}=\frac{C_{i,j}+b_j}{N-1+B} \qquad (3)$$

$C_{i,j}$ は，位置 i に存在する文字 j の個数を意味する．b_j は擬似度数であり，$C_{i,j}$ が 0 となるときに，$Q_{i,j}$ が 0 となることを防ぐために用いられ，各 b_j は文字 j の相対出現頻度×B により決定される．N は配列数を表しており，B は経験的に $\sqrt{N}$ としてよいことがわかっている.

5. 最適化手法

本章では，GS 最適化に用いる最適化手法である MGG と EO について説明する.

5.1. MGG

MGG(Minimal Generation Gap)は，世代間での個体分布の差異を最小化することが望ましいとの考えに基づき，探索序盤における選択圧をできるだけ下げて初期収束を回避するとともに，探索の後半においても集団内に多種多様な個体を生存させやすくして進化的停滞を抑制することを意図した世代交代モデルである．

MGG では，世代交代の対象としてランダムに選ばれた 2 個体を用いるため，初期収束が起こりにくくなっている．また，生存選択時には最良選択とルーレット選択を組み合わせることにより適合度の分散をできるだけ維持できるようにして進化的停滞を抑制できるようにしている．MGG の流れを図 3 に示す．

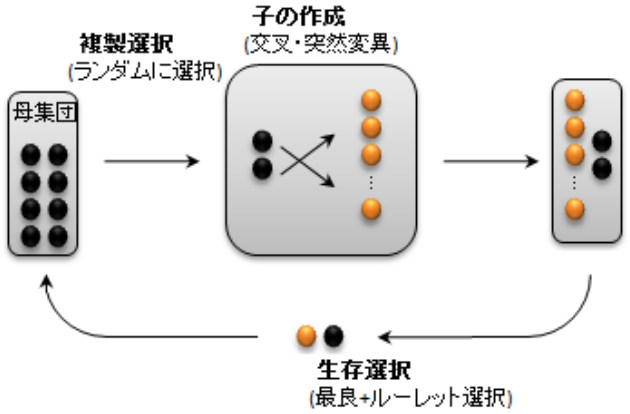

図 3：Minimal Generation Gap(MGG)

5.2. EO

EO は，最適化問題の高品質な解決を得るために近年提案されたものである．EO は，自己組織化臨界のモデルを用いた発見的探索手法であり，解の極めて望ましくない部分をうまく繰り返し除去していき，最適解を導く．以下では，グラフ分割問題の例を用いて EO による処理手順を説明する．グラフ分割問題とは，無向グラフが与えられたとき，その頂点数が等しくなるようなグラフの 2 分割で，境界線と交差する辺の本数が最小となるような分割を求める問題である．以後，この辺の本数をカットサイズと呼ぶ．

図 4 のようなグラフ $G:=(V, E)$が与えられたとする．節点内の数値は各節点に対応する番号であり，これ以降各節点をその番号を用いて $v_1,v_2,\cdots,v_6$ と表記する．また，図中の点線は初期分割の線を表しているが，これはランダムに選択される．

分割を更新していくために各節 v_i の適合値 λ_i を求める$(1\leqq i\leqq 6)$．グラフ分割では，異なる分割へとつながる辺の数は少ない方がいいことから，適合値 λ_i は，点 v_i からのびる枝のうち同じ分割内への辺の数を g_i，異なる分割内への辺の数を b_i として，式：$\lambda_i=g_i/(g_i+b_i)$を用いて求める．

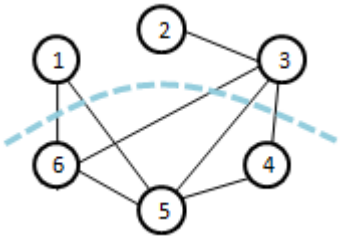

図 4：グラフ G

次に求めた λ_i をソートし，最小の λ_i をもつ節点 v_i を選択し，分割の更新作業を行う．グラフ分割問題での更新作業は，節点 v_i を異なる分割内の任意の点と交換することである．グラフ G では，λ_1 が最小となっていることから，節点 v_1 を異なる分割内の任意の節点(ここでは v_4 を選択)と交換すると図 5 のようになり，カットサイズは 4 から 3 へと減少している．これを繰り返すことにより最適解を求める．

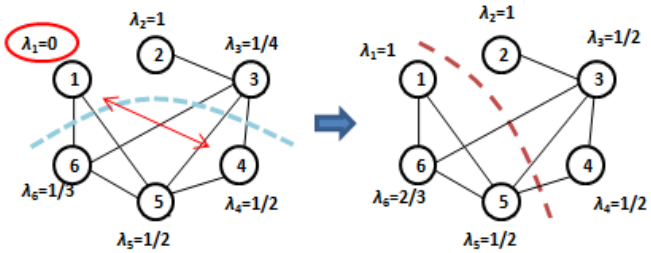

図 5：分割の更新

図 6 に一般的な EO アルゴリズムを示す．アルゴリズム内の変数 i は問題によって変化する．例えば，グラフ分割問題では節点だが，巡回セールスマン問題に適用する場合は都市となる．

1. 初期状態をランダムに決定する．
2. そのシステムの変数 i を決定し，各 i の適合値 λ_i を求める．
3. λ_i をソートし，最小の λ_i をもつ i の更新作業を行う．
4. 2,3 を繰り返す

図 6：EO アルゴリズム

5.3. τ-EO

EO は，最小の λ_i をもつ節点を必ず分割の更新に用いるという特徴から局所解に陥りやすいという傾向が

ある．そこで，更新する節点の決定を確率的に行うために，τ という値を用いて確率 $P(n) \propto n^{-\tau}$ を備えたランク n をもつ節点を選ぶようにする．よって，変数 i の λ_i 値がランク R_i のとき，i が選択される確率 $P_i=R_i^{-\tau}$ となる．例として，図 4 のグラフ G でこの確率を計算した結果を表 1 に示す．

表 1：適合値と確率の関係

i	適合値 λ_i	ランク R_i	確率 P_i
1	0	1	$1^{-\tau}$
2	1	6	$6^{-\tau}$
3	1/4	2	$2^{-\tau}$
4	1/2	4	$4^{-\tau}$
5	1/2	5	$5^{-\tau}$
6	1/3	3	$3^{-\tau}$

τ 値は，大きすぎる値をとると，ランク R_i =1 での確率 P_i と，ランク R_j =2 での確率 P_j との差が大きくなってしまい，EO のように λ_i が最小の i を毎回選択するようになり局所解に陥りやすくなる $(i \neq j)$. 逆に小さすぎる値をとった場合，異なる 2 つの確率 P_i と P_j の間の差が小さくなり，単なるランダム選択に近づき頻繁に良い結果が破壊されてしまう．τ の最適値はだいたい 1.4 前後といわれている．

6. MGG の応用

MGG を応用して，GS の収束率を向上させる手法について説明する．これ以降，この章で説明する MGG を応用したギブスサンプリングの事を MGG 版 GS と呼ぶことにする．

従来の GS は，ある程度繰り返し処理を行うと k-部分文字列の開始位置がほとんど変化しなくなる．そこで，MGG を応用することで初期収束を避け，繰り返し処理の後半でも複数の選択肢を持たせ，局所解の回避を図る．MGG を用いるために，GS で抽出される k-部分文字列集合を個体と考え，個体の適応度には F 値を用いる．子の作成時に行う交叉は，ユーザが指定した作成する子個体の数 M にあわせて，M 個の交叉点をランダムに決め，複製選択で選ばれた 2 つの個体の一部を入れ替える．2 つの個体から作成される子個体の例を図 7 に示す．

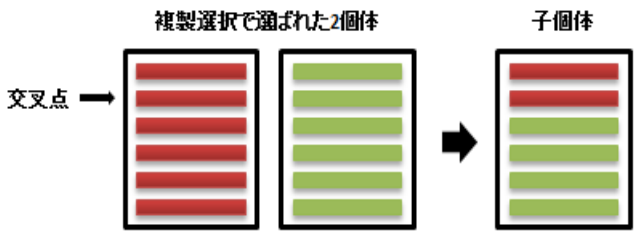

図 7：子個体の作成方法

生存選択には，複製選択で選ばれた 2 個体と子個体から，最も適応度が高い 1 個体と，ルーレット選択で選ばれた 1 個体を母集団に戻す．生存選択終了後に突然変異処理を行う．突然変異にはギブスサンプリングを用いる．ユーザが選択した L 個の個体を母集団からランダムに選択し，ギブスサンプリング処理を繰り返し回数 1 として行う．MGG 版 GS のアルゴリズムを図 8 に示す．

1. 初期状態の母集団として，N 個の個体(k-部分文字列集合)をランダムに生成する．
2. 母集団の中から 2 つの個体をランダムに選択する．
3. 選ばれた 2 個体で交叉処理を行い，M 個の子個体を作成する．
4. 作成されたすべての子個体と，母集団から選ばれた 2 個体の F 値を計算し，最良個体 1 個体と確率的に選んだ 1 個体を母集団に戻す．
5. 母集団の中から L 個の個体をランダムに選択し，突然変異を行う．
6. 指定した回数 2～5 を繰り返す．

図 8：MGG 版 GS のアルゴリズム

7. EO の応用

EO を応用して，GS の収束率を向上させる手法について説明する．これ以降，この章で説明する EO を応用したギブスサンプリングの事を EO 版 GS と呼ぶことにする．

GS では，部分文字列の位置を更新する配列 Z をランダムに選んでいることから，収束にばらつきが生じやすい．この点に着目して，更新配列 Z の決定に EO アルゴリズムの適合値 λ を用いた方法を適用することで収束率向上を図る．このとき，EO の方法を用いるか τ-EO の方法を用いるかはユーザ側に指定させる．また，通常 GS では配列 Z を 1 本選択して部分文字列の位置の更新しているのに対し，EO 版 GS では，配列を n 本選択して $(Z_1,Z_2,\cdots,Z_n)$，選択された n 本すべてに対して部分文字列の位置を仮更新し，その中から最良のもの 1 本を更新として決定するようにした．まず，以下に λ 値の算出方法など更新配列 Z の決定手順について示す．

1. t 本の配列をもつ配列データベース DB と DB から抽出した k-部分文字列集合 S に対して，DB から $sid=i$ の配列 s_i を削除した $DB'_i=\{s_1,s_2,\ldots,s_{i-1},s_{i+1},\ldots,s_t\}$ と，DB'_i の k-部分文字列集合 S_i を作成し，S と S_i の F 値を求める $(1 \leqq i \leqq t)$. 以下では，S の F 値を f，S_i の F 値を f_i と表すことにする．

2. 求めた f, f_i から，s_i の適合値 λ_i を求める．λ_i の計算式を式(4)に示す．

$$\lambda_i = \frac{f}{f + f_i} \qquad (4)$$

3. λ_i をソートし，EO ならば λ_i が小さいものから順に n 個，τ-EO ならば λ_i のランク R_i から確率 P_i を求めて確率的に n 個の更新候補配列を決定する．

次に，n=3 の場合を用いて，n 本の更新候補配列 $Z_1,Z_2,\cdots,Z_n$ から部分文字列の位置を更新する方法を図 9 に示す．Z_1,Z_2,Z_3 それぞれで部分文字列の位置を更新する処理を行い $DB=\{s_1,s_2,\cdots,s_t\}$を仮更新し，仮更新後の配列データベース DB_1, DB_2, DB_3 からそれぞれの F 値を求め，F 値が最大となる更新を適用する．

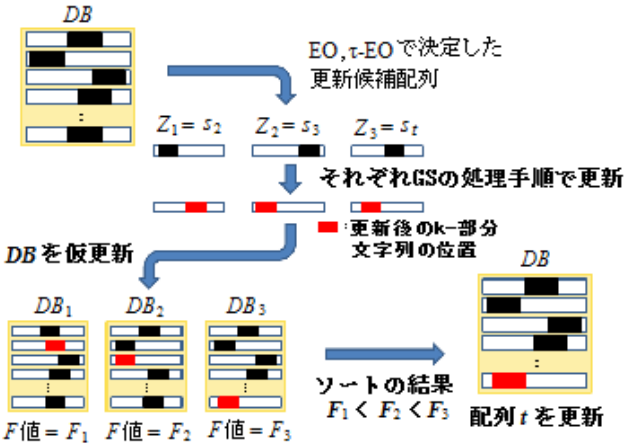

図 9：候補配列からの更新配列決定手順

8. 性能評価

本章では，GS，EO 版 GS，MGG 版 GS の 3 手法について計算結果を比較する．このために利用した計算機環境は，Intel(R) Pentium(R) 4 CPU 2.53GHz，メモリ：1.5GB，SWAP メモリ：2GB，HDD：66.5GB，OS：Fedora 7(moonshine)である．性能評価のために使用した配列データベースは，PROSITE に含まれる Leucine Zipper データセット(登録番号 PS00036)用いた．データセット内には 188 件の配列が含まれている．Leucine Zipper モチーフの形式は，**<[*KR*]-*x*(1,3)-[*RKSAQ*]-*N*-*x*(2)-[*SAQ*](2)-*x*-[*RKTAENQ*]-*x*-*R*-*x*-[*RK*]>**であり，最大 16 文字で形成される．

8.1. 評価の指標

ここでは，評価のための指標となる再現率(recall)と適合率(precision)について説明する．

再現率と適合率は情報検索システムの評価尺度であり，一般に，適合文書=A, 検索された文書=B, $A\cap B=C$ とするとき，再現率=$|C|/|A|$，適合率=$|C|/|B|$となる．これを，配列データベースから抽出した k-部分文字列集合の評価に用いるために，A を配列データベース内の配列データに含まれるモチーフの文字列，B を k-部分文字列集合の全文字列とする．A の計算対象となるモチーフは，1 本の配列に対し 1 つとする．1 本の配列に複数のモチーフが存在する場合は，k-部分文字列にもっとも近いモチーフを計算対象とした．

以下では，例として表 2 を用いて再現率と適合率を求める．ただし，ウィンドウサイズ 4，モチーフを<*ATF*>とし，数値は少数第 2 位を四捨五入したものである．

表 2：配列データベースと k-部分文字列

sid	配列データ	k-部分文字列
1	TATKFATFKT	ATFK
2	KATFAFTFAF	FAFT
3	AAKAKATFTK	AKAK
4	FAKATATFAA	ATFA
5	AATFTKFTTF	AATF

$sid=i$ の配列データに含まれるモチーフの文字列を A_i，k-部分文字列を B_i，$A_i\cap B_i=C_i$ と表すと，$|A|$ は $|A_1|+|A_2|+\ldots+|A_5|$ と計算することができ，$|A_1|=|A_2|=|A_3|=|A_4|=|A_5|=3$ なので，$|A|=15$ となる．B も同様に計算できるので，ウィンドウサイズ 4 より $|B|=20$ となる．次に C_i だが，これは k-部分文字列内に含まれるモチーフの文字列となる．例えば，sid=1 の配列の k-部分文字列にはモチーフがすべて(3 文字)含まれているので，$|C_1|=3$ となる．sid=2 の配列の k-部分文字列には，モチーフ<*ATF*>のうち“*F*”しか含まれていない．この場合，モチーフは 1 文字しか含まれていないので $|C_2|$ =1 となる．同様に，$|C_3|=0$，$|C_4|=3$，$|C_5|=3$ となるので，$|C|=3+1+0+3+3=10$ と計算できる．よって，この k-部分文字列集合の再現率と適合率は，再現率=10/15=66.7%，適合率=10/20=50.0%と求められる．

8.2. 評価実験

各手法に対して，ウィンドウサイズを 16，32，48，64，80 と変化させてそれぞれ 10 回ずつ実験を行い，一定時間毎に k-部分文字列集合を抽出して再現率と適合率を求めた．ウィンドウサイズが 32 のときの処理時間に対する再現率，適合率の平均遷移を図 10 に，最大値と最小値，局所解数を表 3 に示す．ここでの局所解とは，k-部分文字列集合がモチーフとは少しずれた位置，もしくはモチーフとは全く異なる位置に収束した場合を指すものとし，局所解数とは 10 回の実験の中で

局所解となった回数のことを表す．EO 版 GS は，EO と τ-EO のどちらを用いるかのフラグとなる E(EO=0, τ-EO=1)，ウィンドウサイズ k，配列数 num，処理の繰り返し回数 I を，MGG 版 GS は，ウィンドウサイズ k，母集団数 N，作成する子個体の数 M，突然変異の数 L，処理の繰り返し回数 I をそれぞれ入力パラメータとして与える必要があるが，ここでは，EO 版 GS のパラメータを E=1，num=3，τ-EO の τ を 1.2 に，MGG 版 GS のパラメータを N=5, M=5, L=1 として実験を行った．処理の繰り返し回数は各手法とも収束に十分な回数繰り返している．

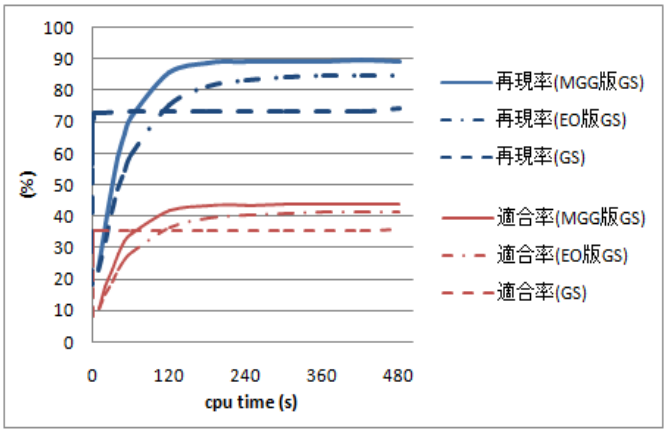

図 10：ウィンドウサイズ 32 での各手法の再現率と適合率の遷移

表 3：各手法の再現率の最大値，最小値，局所解数

	GS	EO 版 GS	MGG 版 GS
最大値	89.8	97.4	100.0
最小値	43.6	48. 6	44. 9
局所解数	4	3	2

従来の GS は，他の手法と比べて収束するのにかかる時間が格段に短いものの，4 割局所解に陥ってしまう上に再現率も 90%以上になることがなく，k-部分文字列集合の収束率はあまり良くない．しかし，EO 版 GS は，局所解へ陥ることを 3 割前後に抑えることができ，局所解となったものを除けば，再現率も平均して 95%程度の値を得ることができた．さらに MGG 版 GS は，局所解へ陥ることを 2 割まで抑えることができ，局所解を除けば，再現率もほぼ 100%近い値を得ることができた．

次に，繰り返し回数一定でウィンドウサイズを変化させたときの再現率と適合率の平均遷移について，EO 版 GS での結果を図 11 に，MGG 版 GS での結果を図 12 に示す．図中の縦線は再現率の最大値と最小値をつないだものである．

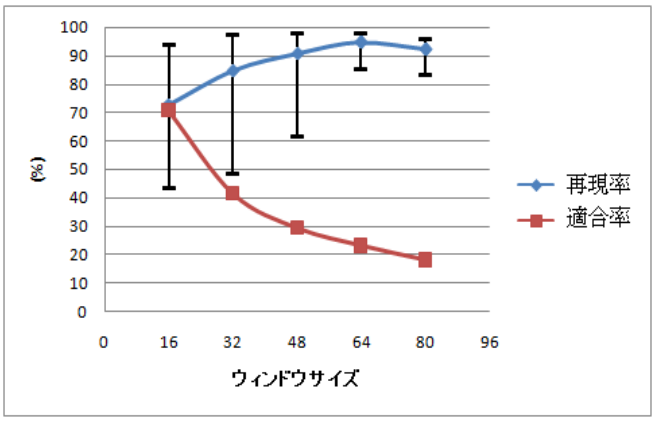

図 11：ウィンドウサイズの変化に対する再現率と適合率の遷移(EO 版 GS)

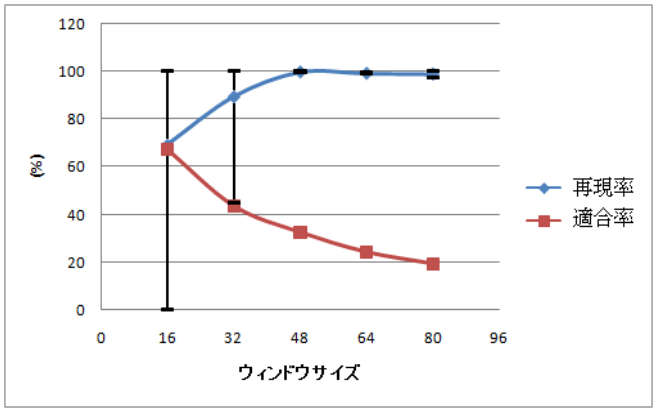

図 12：ウィンドウサイズの変化に対する再現率と適合率の遷移(MGG 版 GS)

両手法とも，ウィンドウサイズを大きくしていくと局所解となることがなくなり，再現率の最大と最小の幅が少なくなる．ウィンドウサイズが小さい時は，EO 版 GS の方が再現率の最大と最小の差が比較的小さいが，ウィンドウサイズ 48 以上になると，MGG 版 GS は再現率の最大と最小にほとんど差がなく，安定してモチーフの部分を抽出できているといえる．また，ウィンドウサイズを大きくしていくと，EO 版 GS では 64 以降，MGG 版 GS では 48 以降から再現率の値が下がってきていることがわかる．これは，ウィンドウサイズを長くすればするほど背景的出現頻度の部分が減り，効果的に収束ができなくなっているものと考えられる．

以上より，ウィンドウサイズ等を含め，適切に各パラメータを与えてやれば，MGG 版 GS と EO 版 GS，特に MG 版 GS は従来の GS よりも優れた抽出能力を持っているといえる．

9. まとめ

本論文では，MGG と EO という 2 種類の最適化手法をそれぞれ応用したギブスサンプリング処理について，評価・比較を行った．PROSITE から取り出した Leucine Zipper モチーフを含むデータセットを用いて，それぞれの手法を実行し，長さ k の類似部分配列を抽出した．その結果，従来のギブスサンプリングに比べ，うまくモチーフ部分を抽出できない回数を，EO 版 GS では 3/4 に，MGG 版 GS では 1/2 に減らすことに成功し，両手法共に再現率も上昇した．特に MGG 版 GS は，全モチーフを網羅していることも多々あり，優れた類似部分配列抽出方法だといえる．

以下に今後の課題について述べる．

(1). 抽出された類似部分配列の集合に対して頻出配列パターンの抽出を行い，本提案手法の再現率や適合率と抽出される頻出配列パターン数との関係を調べることにより，抽出される頻出配列パターン数を減少させる方法の検討．

(2). 1 本の配列内に複数のモチーフが存在する場合に，モチーフの繰り返しも抽出できるようにウィンドウ数を増やす方法の検討．

(3). ウィンドウサイズを自動検出する方法の検討．

謝　辞

本研究の一部は，日本学術振興会，科学研究費補助金(基盤研究(C)，課題番号：17500097) の支援により行われた．

文　献

[1] PROSITE：http://kr.expasy.org/prosite.

[2] Pfam：http://www.sanger.ac.uk/Software/Pfam.

[3] Hajime Kitakami, Tomoki Kanbara, Yasuma Mori, Susumu Kuroki, and Yukiko Yamazaki: Modified PrefixSpan Method for Motif Discovery in Sequence Databases, Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence (PRICAI2002), Springer-Verlag ,pp.482-491 , August 2002.

[4] 塔野 薫隆，北上 始，田村 慶一，森 康真，黒木 進：Modified PrefixSpan 法を用いた頻出正規パターンの抽出をめざして，日本データベース学会 Letters，Vol.3，No.1，pp.61-64，2004 年 6 月．

[5] 加藤 智之，北上 始，森 康真，田村 慶一，黒木 進：極小かつ非冗長な可変長ワイルドカード領域を持つ頻出パターンの抽出，電子情報通信学会論文誌 D「データ工学特集号」，Vol.J90-D，No.2，pp.281-291，2007 年 2 月．

[6] Lawrence C.E., Altschul,S.F., Boguski,M.S., Liu,J.S., Neuwald,A.N. and Wotton,J.: Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment, Science,263,208-214, 1993.

[7] Liu, J.S., Neuwald, A.N. and Lawrence,C.E.:Bayesian models for multiple local sequence alignment and Gibbs sampling strategies, JASA, 90, 1156-1170, 1995.

[8] 加藤 智之，森 康真，荒木 康太郎，黒木 進，北上 始：可変長配列パターン抽出法におけるギブスサンプリングを用いた不要パターン の除去方式，日本データベース学会論文誌（DBSJ Letters），Vol.6, No.1, pp.65-68, 2007 年 6 月．

[9] 佐藤 博，小野 功，小林 重信：遺伝的アルゴリズムにおける世代交代モデルの提案と評価，人工知能学会誌,Vol.1,No.51,pp.734-743,1997.

[10] Stefan Boettcher, Allon Percus: Nature’s way of optimizing, Artificial Intelligence 119, 275–286, 2000.

[11] S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol 220, Number 4598, pages 671-680, 1983.