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

2007－MPS－67 (46)
2007－BIO－11 (46)
2007／12／21

ACOを用いたモチーフ抽出アルゴリズムに関する研究

三上 あい , 施 建明
室蘭工業大学 情報工学科
Email: shi@mmm.muroran-it.ac.jp

概要

タンパク質のアミノ酸配列に存在する局所的な共通配列をモチーフ と呼ぶ. 配列からモチーフを発見することはタンパク質の機能解析に欠かせない重要な作業である. 本研究では Ant Colony System を用いた多配列間から未知のモチーフを発見するアルゴリズムを提案し, その動作を計算機実験で確認する. また, 提案したアルゴリズムと他の幾つかのアルゴリズムとの比較を行った. 計算機実験はタンパク質断片に対して行い, モチーフの評価は位置特異的スコアを用いて算出した. 結果, ACS-Motif は収束性で劣るが, 解の精度と探索時間において優れていることがわかった. ACO はモチーフ抽出問題の解法として発展の余地のある有効なアルゴリズムといえる.

An efficient algorithm for finding motif in multiple sequences using the ant colony optimization

Ai Mikami and Jianming Shi*
Muroran Institute of Technology
Computer Science and Systems Engineering

Abstract

The motif finding problem is important to predict the protein functions. In this study, we suggest an algorithm to find the motif with help of the ant colony optimization. To investigate the efficiency of the algorithm, we compare its behavior with the other existing algorithms used in this problem. The results show that the proposed algorithm achieves better in accuracy of the solutions and is much faster in computational time. We believe that the suggested algorithm is promising in motif finding research.

* Email: shi@mmm.muroran-it.ac.jp (J. Shi)

1 序論

モチーフとは複数の塩基またはアミノ酸配列中に共通して存在する類似した部分配列を指す. 同じモチーフを持った異なる複数のタンパク質間に共通した機能が見つかれば, それはモチーフによって発現する機能であると推定される. これより未知のタンパク質の配列中に同じモチーフが存在すれば, そのタンパク質も同じ機能を持っていると予測することができる. モチーフ抽出問

題はその目的の違いによっていくつかの種類が考えられる．本研究で問題とするのは，ギャップ無しの異なる複数配列間から未知のモチーフを発見することである．問題の解法としては，異なる配列同士を類似した部分配列を基準として整列させるアラインメントという方法を採る．この問題は$\cap NP$-困難な問題として知られており，厳密解法をもって解を求めるのは実用的ではない．そのため，ある程度短い時間で探索を行える近似解法が用いられている．本研究ではメタヒューリスティクスアルゴリズムの1つである Any Colony Optimization(ACO) を用いてモチーフ抽出を行うアルゴリズムを提案し，その有効性を試す実験を行った．

ACO は自然界でアリが餌を探索するとき，フェロモン情報を基にして餌までの最短経路を発見するというメカニズムに発想を得たアルゴリズムである [1]．アリはまず巣の周りをランダムに探索し，餌を発見すると通った道にフェロモンを分泌しながら巣に戻る．他のアリは仲間のフェロモンを見つけるとランダムな徘徊を止め，その後をたどる．そして餌を発見すると同じようにフェロモンを分泌して巣に戻り，経路を補強する．アリのフェロモン探知能力は濃度が濃いほどよく発揮されるため，多くのアリが通った経路ほど他のアリがその後をたどる可能性が高くなる．またフェロモンは揮発性であるため，距離の長い経路は行進に時間がかかる分，短い経路に比べて多くのフェロモンが蒸発することになる．以上をまとめると，アリは巣と餌の間を往復するうちにより短い経路をたどるものが多くなり，その経路のフェロモンが補強される．フェロモン濃度が濃いためにさらに多くのアリが同じ経路を選択するようになり，この繰り返しによって最終的に全てのアリが最短経路を通るようになる．

ACO に適用されるルールは以下の4つである．

- 初期状態でアリはランダムに探索する
- 餌を発見したアリはフェロモンを分泌しながら巣に戻る
- アリはフェロモンをたどる
- フェロモンは揮発性で，濃度が濃いほど発見しやすい

ACO においてアリの探索する空間はグラフ上の経路と端点，餌までの最短経路は最適解と考えられる．フェロモンは経路を選択する指針であり，複数の経路が存在した場合，フェロモンが多い道ほど高い確率で選択される．強化や蒸発によりフェロモン量は常に変動するため，ヒューリスティクス情報と併せて経路選択の参考にすることにより，良質な解付近での探索の集中化と，より良い解を求めるための探索の分散化という相反する欲求を同時に満たすことが可能となる．

実験はプログラムをコンピュータ上で実装し，その動作について比較を行った．1つめの実験は ACO をモチーフ抽出問題に適応させたアルゴリズムである，ACS-Motif に収束性向上のための改良を施してその動作を確認することであり，2つめは改良 ACS-Motif と Genetic Algorithm, Gibbs Sampling との動作比較である．結果，改良した ACS-Motif は狙い通り収束性が改善され，他2つのアルゴリズムと比較して解の精度や探索時間に優れていることが実証された．これより ACO アルゴリズムは，改善の余地はあるがモチーフ抽出問題を解くアルゴリズムとして優れていると確認された．

2 ACO のモチーフ抽出問題への適用

2.1 Ant Colony System; ACS

ACS は Dorigo らによって，巡回セールスマン問題 (Traveling Salesperson Problem; TSP) を対象として提案されたアルゴリズムである [2, 3]．

TSP とは，都市間の距離が d であるような n 個の都市を一度ずつ訪問して戻ってくる巡回経路のうちで最も移動距離が短い経路を探す問題である．ACS はフェロモンによる経路選択，解の生成と評価，フェロモンの更新という手順を用いる．一連の作業を1反復としてこれを $t_{\max}$ 回行う．この問題においてはアリをセールスマンに見

立て都市を巡回させて巡回路, つまり解を形成する. フェロモンは各2都市間の経路に敷かれるもので都市 i, j 間のフェロモンは $\tau_{ij}(t)$ と与えられる. またヒューリスティクス情報は都市間距離 d の逆数として定義される:

$$\eta_{ij} = \frac{1}{d_{ij}}. \tag{1}$$

都市間移動の選択方式は2つ存在し, パラメータ q_0 と, 乱数 $q \in (0,1)$ との比較で決定される. 1つはフェロモンとヒューリスティクス情報に基づいて確率的に経路を選択する方法であり, 次式で定義される:

$$P_{ij}^k(t) = \frac{[\tau_{ij}(t)][\eta_{ij}]^\beta}{\sum_{l \in N_k}[\tau_{ij}(t)][\eta_{ij}]^\beta} \qquad j \in N_k. \tag{2}$$

ただし N_k は現在いる都市から移動可能な都市の集合, β はヒューリスティクスの信頼度を決めるパラメータである. もう1つの方式は, 次式のように P_{ij}^k が最大となるような都市 j へ移動するものである:

$$\arg\max_{j \in N_k}[\tau_{ij}(t)][\eta_{ij}]^\beta. \tag{3}$$

移動と同時にフェロモンを減少させる. この更新は後のアリが同じ経路に集中しすぎるのを防ぎ, 最適解の精度を上げる目的で行われ, 次式のようにして定義される (ローカルアップデート):

$$\tau_{ij}(t) = (1-\rho_L)\tau_{ij}(t) + \rho_L \Delta\tau_0. \tag{4}$$

ρ_L はフェロモンの蒸発率で $\rho_L \in (0,1), \Delta\tau_0$ はフェロモンの減少度合いを決めるパラメータで, 普通最短経路を通った場合の移動距離の逆数をとる. これらを $\{N_k\} = \phi$ となるまで繰り返すことで巡回路を形成する. すべてのアリが巡回を終えたら, 探索の集中化を図る目的でそれまでの最適経路のみフェロモンを増加させる (グローバルアップデート):

$$\tau_{ij}(t+1) = (1-\rho_G)\tau_{ij}(t) + \rho_G \Delta\tau_{ij}(t). \tag{5}$$

$\rho_G \in (0,1), \Delta\tau_{ij}(t)$ はフェロモンの増加に関わるパラメータである. 以上

アルゴリズム 1: ACS

Step1 必要なパラメータの設定. $t = 1$ とする.

Step2 $k = 1$ とする.

(a) 乱数 q を生成し, $q_0 < q$ なら式 (2), $q_0 \leq q$ なら式 (3) に従って $\{N_k\} = \phi$ となるまで経路選択を続ける.

(b) 式 (4) に従ってローカルアップデートを行う.

(c) 解の評価を行う.

(d) $k = k+1$. $k \leq m$ なら (a) に戻る.

Step3 最良解に対して (5) に従いグローバルアップデートを行う.

Step4 $t = t+1$. $t \leq t_{\max}$ なら Step2 に戻る.

2.2 モチーフ抽出問題への適用

解の定義

モチーフ抽出問題を ACO アルゴリズムで解くにあたり, 解を以下のように定義する.

N 本の配列 S_l $(l = 1, \ldots, N)$ が存在するとき, 各配列中から長さ M のモチーフを抽出したものをモチーフ行列とする. 配列 S_l のモチーフの1つめの位置を i_l, 要素を S_{l,i_l} とする. 配列の長さを $|S_l|$ とすれば選択可能な i_l は $i_l \leq |S_l| - M + 1$ である. 1から N までの i_l を要素とする数列がアリのたどった経路, つまり解となる.

$$\mathrm{Motif}(i_1, \ldots, i_N) = \begin{pmatrix} S_{1,i_1} & \cdots & S_{1,i_1+M-1} \\ S_{2,i_2} & \cdots & S_{2,i_2+M-1} \\ \vdots & \vdots & \vdots \\ S_{N,i_N} & \cdots & S_{N,i_N+M-1} \end{pmatrix}$$

ただし, アリは S_{l,i_l} から $S_{l+1,i_{l+1}}$ に直接移動するのではなく, S_l のどの要素からでも共通の中継地点を通って S_{l+1} の要素に移動するものとする. さらに全ての i_l に対して S_{l,i_l} から中継地点までの経路を便宜上同じものと考えることとする. これによって経路を減少させ, 問題の簡略化を図る.

解の評価

生成された解について評価を行う. モチーフの評価基準は類似度だが, 単なる記号として類似の程度を測るのではなく, アミノ酸としての

特性を考慮しなければならない．本研究では位置特異的スコア行列 (Position Specific Scoring Matrix(PSSM)) を用いる．

抽出したモチーフ行列について，各列でのアミノ酸の種類ごとの出現数を数える．位置 i $(i = 1, \ldots, M)$ でのアミノ酸 A の出現数を $n(A,i)$ と表す．位置 i でのアミノ酸 A の出現頻度を示す擬似行列 $Freq(A,i)$ は，サンプル数の少なさを補うための擬似度数 x, y を用いて次の式で計算される:

$$Freq(A,i) = \frac{n(A,i) + x}{N + y}. \tag{6}$$

位置 i におけるアミノ酸 A の位置特異的スコアは次の式によって求められる:

$$PSSM(A,i) = \ln \frac{Freq(A,i)}{P(A)}. \tag{7}$$

ここで，$P(A)$ は SWISS-PROT のデータを基にした，アミノ酸 A の統計的な出現確率である．

Motif$(i_1, \ldots, i_N)$ のスコア $E(i_1, \ldots, i_N)$ は次の式で与えられる:

$$E(i_1, \ldots, i_N) = \sum_{i=1}^{M} \sum_{allA} n(A,i) \times PSSM(i,j). \tag{8}$$

ヒューリスティクス情報

ヒューリスティクス情報についてはそれまでの暫定解の $Freq$ を用いて，抽出されるモチーフが統計的確率からみてどれだけ稀であるかを示すオッズ比によって計算される．S_{l,i_l} から $S_{l+1,i_{l+1}}$ への経路のヒューリスティクスは，前述の $l+1$ 番目の中継地点から S_{l+1} の i_{l+1} 番目の要素へのヒューリスティクスとみなすことができる:

$$\eta(S_{l,i_l}, S_{l+1,i_{l+1}}) = \eta(l+1, i_{l+1}), \tag{9}$$

$$\eta(l+1, i_{l+1}) = \frac{\prod_{i=1}^{M} Freq(A_{l+1,i_{l+1}}, i)}{\prod_{i=1}^{M} P(A_{l+1,i_{l+1}})}. \tag{10}$$

A_{l,i_l} は S_{l,i_l} のアミノ酸の種類を表している．

フェロモンについても同様に考える:

$$\tau(S_{l,i_l}, S_{l+1,i_{l+1}}) = \tau(l+1, i_{l+1}). \tag{11}$$

2.3 ACS-Motif

ACS をモチーフ抽出問題に適応させたアルゴリズムが ACS-Motif である [4, 5]．経路選択に一部変更があり，式 (2)-(3) でフェロモンの重要度を決めるパラメータとして α を導入している:

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}^{\beta}]}{\sum_{l \in N_k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} \qquad j \in N_k, \tag{12}$$

$$\arg\max_{j \in N_k}[\tau_{ij}(t)]^{\alpha}[\eta_{ih}]^{\beta}. \tag{13}$$

フェロモン更新についてはグローバルアップデートとローカルアップデートのフェロモン蒸発率を常に等しいものとして共通の変数 ρ であらわす．$\Delta\tau_{ij}(t)$ については良い経路ほど多くのフェロモンが残るようにするため，暫定解のスコアをあてる．

アルゴリズム 2: ACS-Motif

Step1 必要なパラメータの設定．$t = 1$ とする．

Step2 $k = 1$ とする．

- **(a)** $l = 0$ とする．
  - **(I)** 乱数 q を生成し，$q_0 < q$ なら式 (12)，$q_0 \leq q$ なら式 (13) に従って次の移動地点 i_{l+1} を決定．
  - **(II)** 解行列に i_{l+1} を追加．
  - **(III)** $l = l + 1$. $l \leq N - 1$ なら (I) に戻る．
- **(b)** 生成した解行列の各経路について，(4) に従ってローカルアップデートを行う．
- **(c)** (8) によって解のスコアを計算．
- **(d)** $k = k + 1$. $k \leq m$ なら (a) に戻る．

Step3 最良解に対して (5) に従いグローバルアップデートを行う．

Step4 $t = t + 1$. $t \leq t_{\max}$ なら Step2 に戻る．

2.4 ACS-Motif の改定

収束性の向上

式 (4)-(5) より最適経路のフェロモン $\tau_{best(ij)}(t)$ とそれ以外の経路のフェロモンについて次のように言える:

$$\lim_{t \to \infty} \tau_{best(ij)}(t) = \Delta\tau_{ij}(t), \tag{14}$$

$$\lim_{t \to \infty} \tau_{ij}(t) = \Delta\tau_0. \tag{15}$$

これより ACS-Motif が収束する条件の 1 つは $\Delta\tau_0 \ll \Delta\tau_{ij}(t)$ といえる. ただし (14) は $k = 1, \ldots, m$ の反復を行う前の残留フェロモンである. $\Delta\tau_0 \ll \Delta\tau_{ij}(t)$ を満たし, $t \to \infty$ の時点で最適経路を $K(\leq m)$ 回通ったとすると, この時点での残留フェロモンは次のように近似できる:

$$\tau_{best(ij)}(t) \approx (1-\rho)^K \Delta\tau_{ij}(t). \qquad (16)$$

この式より m と ρ が大きいほど収束しずらいことがわかる. この 2 つのパラメータは広範囲を探索する性能にも影響する値であり, この目的を達成するためにはある程度大きい m と ρ が必要となる. 以上より解の精度を維持しながら収束性も持たせるような m と ρ を探すのは困難であることがわかる.

そこでローカルアップデートの更新式 (4) を以下のように変更する:

$$\tau_{ij}(t) = (1-\rho\gamma^w)\tau_{ij}(t) + \rho\gamma^w \Delta\tau_0. \qquad (17)$$

ただし $\gamma \in (0,1)$, w は時点 t で最適経路が更新されなかった回数である. γ を適当に設定することで m と ρ の値によらない一定回数の反復での収束が期待され, また探索の初期段階では頻繁に最適経路の更新がなされるため解の精度に与える影響は少ないと予測される.

アルゴリズム 3: 改良 ACS-Motif

Step1 必要なパラメータの設定. $t = 1, w = 1$ とする.

Step2 $k = 1$ とする.

(2a) $l = 0$ とする.

(I) 乱数 q を生成し, $q_0 < q$ なら式 (12), $q_0 \leq q$ なら式 (13) に従って次の移動地点 i_{l+1} を決定.

(II) 解行列に i_{l+1} を追加.

(III) $l = l + 1$. $l \leq N - 1$ なら (I) に戻る.

(2b) 生成した解行列の各経路について, 式 (17) に従ってローカルアップデートを行う.

(2c) 式 (8) によって解のスコアを計算.

(2d) $k = k + 1$. $k \leq m$ なら (2a) に戻る.

Step3 **(3a)** 最良解に対して式 (5) に従いグローバルアップデートを行う.

(3b) 最良解が更新されなければ $w = w + 1$

Step4 $t = t + 1$. $t \leq t_{\max}$ なら Step2 に戻る.

3 ACS-Motifの計算機上での実装と動作

提案したアルゴリズムを実装してその動作を確認し, 既存の幾つかのアルゴリズムと比較実験を行った. 比較対象は Genetic Algorithm(GA) と Gibbs Sampling 法である.

3.1 パラメータの設定

ACS-Motif のパラメータの設定値は以下のとおりである: $\alpha = 1.8, \beta = 0.5, \gamma = 0.95, \rho = 0.5, q_0 = 0.9$. $\Delta\tau_0$ はランダムに 10 個作ったモチーフの平均スコアとして, 同時にこれを初期フェロモンとして各経路に与えた. $\Delta\tau_{ij}(t)$ は $\Delta\tau_0$ との兼ね合いを考えて暫定解の 3 倍とした.

3.2 比較対象のアルゴリズムについて

比較対象としたアルゴリズムは 2 つ, ヒューリスティクス情報を用いない基礎的な GA と, Metropolis によって提案された Monte Carlo sampling とも言われる Gibbs Sampling の一種である [6, 7].

以下のアルゴリズムで $q_1, q_2, q_3 \in (0,1)$ はパラメータであり, 特断わらない場合, 記号は前出と同じものを意味する.

アルゴリズム 4: GA

Step1 必要なパラメータの設定. $t = 1$ とする. 集団内の個体をスコアを降順にソート.

Step2 $k = \frac{m}{2} + 1$ とする.

(2a) $l = 1$ とする.

(2b) q_1 の確率で整数 $i_{l\text{new}} \in \{1, 2, \ldots, |S_l| - M + 1\}$ を選ぶ.

(2c) $l = l + 1$. $l \leq N$ なら (b) に戻る.

(2d) k 個めの解について, 全ての l について i_l と $i_{l\text{new}}$ を入れ替えた場合のスコアを計算. もとのスコアより良ければ $i_l = i_{l\text{new}}$ とする.

(2e) $k = k + 1$. $k \leq m$ なら (2a) に戻る.

Step3 $i = 1$ とする.

(3a) $k = 1, \ldots, \frac{m}{2}$ の中から, $P_k = \frac{E_k}{\sum_{j=1}^{\frac{m}{2}} E_j}$ に従って k を選択. $k_{p1} = k$ とする.

$k = \frac{m}{2}+1, \ldots, m$ の中から,
$P_k = \frac{E_k}{\sum_{j=\frac{m}{2}+1}^{m} E_j}$ に従って k を選択.
$k_{p2} = k$ とする.

(3b) $l = 1$ とする.

(3c) q_2 の確率で行列 L に l を加える.

(3d) $l = l+1$. $l \leq N$ なら (3c) に戻る. $l = N$ なら L に N を加える.

(3e) $j = 1$.

(I) 親 k_{p1} と k_{p2} のモチーフ行列の L_j から L_{j+1} までの要素を交換する.

(II) L から L_j を消去. $j = j+1$. $\{L\} \neq \phi$ なら (I) に戻る.

(3f) $i = 1+1$. $i \leq m$ なら (3a) に戻る.

Step4 作成した子世代をスコアの降順にソート. 1 から $\frac{m}{2}$ までを次世代の親として残す.

Step5 $t = t+1$. $t \leq t_{\max}$ なら Step2 に戻る.

アルゴリズム 5: Gibbs Sampling

Step1 必要なパラメータの設定. $t = 1$ とする.

Step2 ランダムに整数 $l_{\mathrm{pos}} \in [1, N]$ を選ぶ.

(2a) ランダムに整数 $i_{\mathrm{new}} \in \{1, 2, \ldots, |S_{lpos}| - M + 1\}$ を選ぶ.

(2b) 現在のモチーフ行列の l_{pos} 番目の要素を i_{new} に入れ替えた場合のスコア。を E_{new} として現在のモチーフのスコアとの差 $dE = E_{\mathrm{new}} - E$ を計算.

(2c) $\min[1, e^{\frac{dE}{T}}]$ の確率で $i_{l\mathrm{pos}} = i_{\mathrm{new}}$ とする.

Step3 $l = 1$ とする.

(3a) q_3 の確率で行列 L に l を加える.

(3b) $l = l+1$. $l \leq N$ なら (3a) に戻る. $l = N$ なら L に N を加える.

Step4 $j = 1$ とする.

(4a) ランダムに整数 $i_{\mathrm{shift}} \in \{1, 2, \ldots, \min_{l \in L_j, L_{j+1}} |S_l| - M + 1\}$ を選ぶ. ただし L_j は L の要素である.

(4b) $l_{\mathrm{pos}} = \arg\min_{l \in L_j, L_{j+1}} |S_l| - M + 1$ とする. 全ての $l = L_j, \ldots, L_{j+1}$ に対してモチーフの開始位置を $(i_{\mathrm{pos}} - i_{\mathrm{shift}})$ だけシフトしたモチーフのスコアを E_{new} として元のモチーフのスコア E との差 $dE = E_{\mathrm{new}} - E$ を計算.

(4c) $\min[1, e^{\frac{dE}{T}}]$ の確率で全ての $l = L_j, \ldots, L_{j+1}$ に対して $i_l = i_l - (i_{\mathrm{pos}} - i_{\mathrm{shift}})$ とする.

(4d) L から L_j を消去. $j = j+1$. $\{L\} \neq \phi$ なら (4a) に戻る.

Step5 $T = q_4 T$, $t = t+1$ として $t \leq t_{\max}$ なら Step2 に戻る.

設定		avg	max	time
$m = 10$	改良前	246	266	8.8
$\rho = 0.5$	改良後	248	260	6.0
$m = 10$	改良前	247	268	9.5
$\rho = 0.8$	改良後	253	259	7.1
$m = 20$	改良前	247	266	21.4
$\rho = 0.5$	改良後	255	266	11.2
$m = 20$	改良前	255	269	18.8
$\rho = 0.8$	改良後	255	269	13.5

表 1: 各設定値における改良前後の ACS-Motif の data1 に対する動作結果.

3.3 実験

実験に用いた配列は 47 配列 B 型肝炎ウイルス, 127 配列 C 型肝炎ウイルス, 64 配列ヒト免疫不全ウイルス, 188 配列ホモ・サピエンス, 160 配列熱帯熱マラリア原虫の 20 種類アミノ酸配列である [8]. 以降はこれらを順に data1 から data5 と呼ぶ. この配列中から 9 残基のモチーフを抽出した. スコアの算出法は式 (6)-(8) に従う. 式 (6) の $Freq$ はその時点での暫定解のモチーフ行列で計算し, パラメータの設定は $(x, y) = (1, 20)$ とした. 実験はそれぞれ 10 回ずつ行った. また, 実験結果の時間単位は秒である.

実験 1 改良前と後の ACS-Motif の動作比較

改定した ACS-Motif の収束性が向上していることを確かめるため, 改良前後のアルゴリズムについて data1 を対象に動作比較を行った. この実験は ρ と m の値を変更して 4 つの場合に分けており, その値は $(m, \rho) = (10, 0.5), (10, 0.8), (20, 0.5), (20, 0.8)$ とした. 反復数は 200 回である. 結果を表 1 に示す. avg は 10 回の試行で出力された最適解の平均であり, max は試行中の最高の値, time は最適解に到達するまでの時間である. スコアは小数点以下を丸めてある. 図 1 は各設定値の A_{roc} を表している. 改良前の A_{roc} の平均値は 0.67, 改定後は 0.83 である. また図 2 はそれぞれ改良前と後のグラフである. 他の設定値についても類似した形状のグラフが出力された.

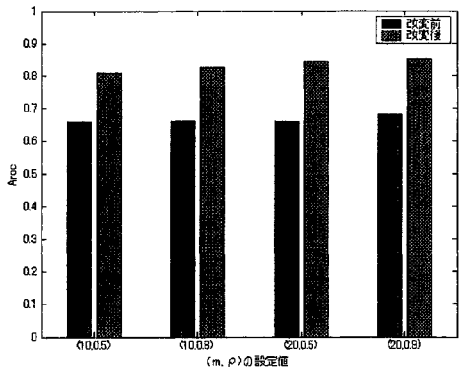

図 1: 各設定値における改良前後の ACS-Motif の data1 に対する A_{roc} の値.

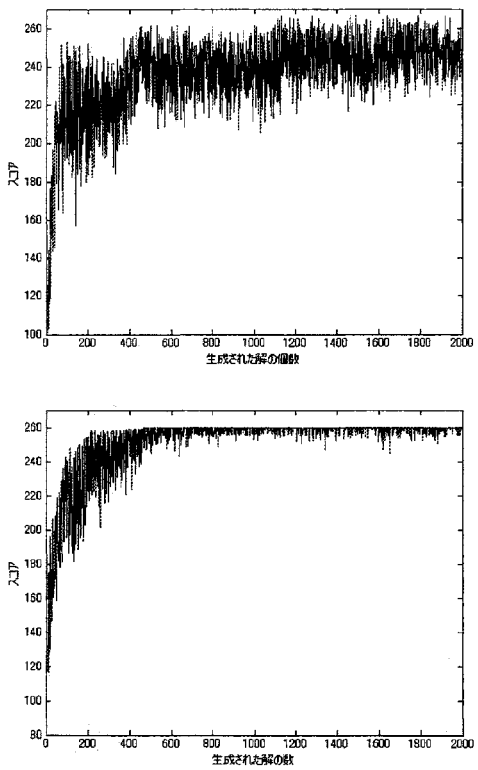

図 2: $m = 10, \rho = 0.5$ のときの改良前後の ACS-Motif の data1 に対するスコアの推移. 上が改良前, 下が改良後.

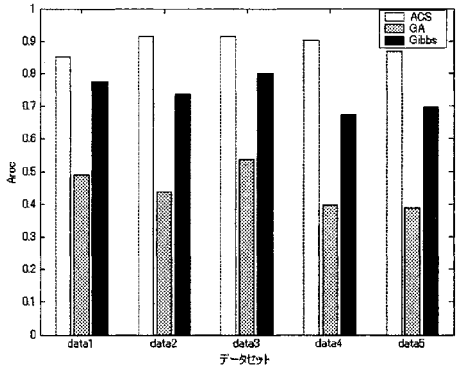

図 3: data1 から data5 に対する各アルゴリズムの A_{roc}.

実験 2 5 種類の配列セットに対するアルゴリズムの動作比較

5 つの配列セットに対して改定した ACS-Motif, GA, Gibbs-Sampling を動作させた. この実験での ACS-Motif の設定は $m = 10, \rho = 0.5$ としている. GA の個体数は 100, 他 $q_1 = 0.7, q_2 = 0.3$ である. Gibbs Sampling の T は $T = 1$ からはじまり, 各反復時に 0.999 をかけて減少させ, また q_3 は $q_3 = 0.1$ とした. 実験では各アルゴリズムで解を 5000 個生成するまで反復を続けた.

結果は表 2 のようになった. また各アルゴリズムが解を 1 つ作成するのにかかる時間の平均は ACS-Motif が 0.017, GA が 0.015, Gibbs Sampling で 0.008 だった. 図 3 は A_{roc} を表しており, 図 4 は data5 に対する各アルゴリズムのスコアの推移を示したグラフである. なお, 図 4 のグラフ中一番上が ACS-Motif, 中央が Gibbs Sampling, 下が GA である. A_{roc} の平均値は ACS が 0.89, GA が 0.45, Gibbs が 0.74 となった.

4 ACS-Motifの性能についての考察

図 1, 2 より改定後の ACS-Motif の収束性が向上していることがわかる. また平均スコアや最

	data1			data2			data3			data4			data5		
	avg	max	time	avg	max	time	avg	max	time	avg	max	time	avg	max	time
ACS	249	265	10.1	561	573	65.3	334	344	15.0	557	578	93.7	757	774	102.2
GA	232	248	26.3	447	478	78.6	314	332	37.9	451	477	107.0	595	621	124
Gibbs	242	248	10.3	537	574	38.4	324	331	40.0	520	555	49.8	720	752	56.1

表 2: data1 から data5 に対する各アルゴリズムの動作結果.

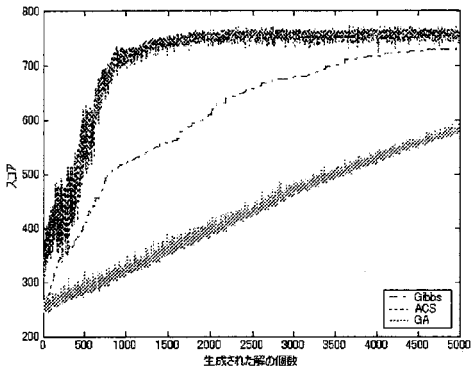

図 4: data5 に対する各アルゴリズムのスコア推移.

高スコアから, どの設定においても改定前後で解の精度にほとんど違いがないことも見て取れる. 暫定解付近で集中的に探索を行ったため, 最終的に出力される解に到達するまでの時間は短くなっている. 以上より, 改定に期待した効果はある程度確認することができた. 完全な収束には至らなかった原因はパラメータ α の影響と考えられる. 振動幅は規模の大きな配列セットでより大きくなる.

次に他のアルゴリズムとの比較については, data2 を除く全ての対象配列について平均スコア, 最高スコアともに最も良い結果を出している. また data2 に関しては Gibbs の最高スコアがわずかに上回っているが, 誤差の範囲といえる程度の差である. 問題となるのは最適解への到達時間で, これは配列規模の大きい data2, 4, 5 では Gibbs に大きく水をあけられている. しかし図 4 から, 最適解近傍へ到達するまでに必要な解の作成個数は ACS が 1000 個程度であるのに対して Gibbs では 4000 個にもなっている. これに一つの解を作成するのにかかる時間をかければ ACS は 17, Gibbs で 32 となり, 結果 ACS のほうが動作時間に対する解の期待値は高いことがわかる.

これらの結果より ACO は多配列アラインメントを行うアルゴリズムとして特に最適性と探索の速さの点で優れていることが実証された. 反面収束性においては, ほとんど確実な収束が期待できる Gibbs Sampling と比べると不安定であると言わざるを得ない. しかしこの問題点を考慮した上でも, ACO はモチーフ抽出問題の解法として発展性のある優秀なアルゴリズムであると考えられる.

参考文献

[1] Dorigo, M. Maniezzo, V. and Cambini, A. C. (1996): The Ant System: Optimization by a colony of cooperating agents, *IEEE Transaction on Systems, Man, and Cybernetics-Part B* **26** 29-41.

[2] Dorigo, M. and Gambardella, L. M. (1997): Ant colonies for the traveling salesman problem, *BioSystems* **43**, 73-81.

[3] Dorigo, M. and Gambardella, L. M. (1997): Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation* **1**, 53-66.

[4] 竹中康弘 (2006) : ACO を用いた大域最適化アルゴリズムとその応用に関する研究, 平成 17 年度室蘭工業大学修士学位論文.

[5] Karpenko, O., Shi, J. and Dai, Y. (2005): Prediction of MHC class II binders using the ant colony search strategy, *Artificial Intelligence in Medicine* **35**, 147-156.

[6] Mount, D. W. (監訳: 岡崎康司, 坊農秀雅) (2002): バイオインフォマティクス-ゲノム配列から機能解析へ, メディカル・サイエンス・インターナショナル.

[7] Lund, O., Nielsen, M., Lundegaard, C., Kesmir, C. and Brunak, S. (2005): Immunological Bioinformatics, The MIT Press, Massachusetts.

[8] The Immune Epitope Database and Analysis Resource (IEDB) : http://www.immuneepitope.org/