

遺伝情報学におけるセグメント法を用いたマルチプルアライメント

東京理科大学大学院大矢研究室 村松重人 (Shigeto Muramatsu)

東京理科大学理工学部情報科学科 大矢雅則 (Masanori Ohya)

1.はじめに

遺伝子のDNA (デオキシリボ核酸) は4種類の塩基, A (アデニン), T (チミン), G (グアニン), C (シトシン) で構成され, アミノ酸はこの4種類の塩基からなり, さらにタンパク質は20種類のアミノ酸で構成されている. この1次元のデータを解析し, そのデータを用いて生物の進化系統樹を作成する. また, 直接このデータを比較することにより, 生物間における共通の部分を見つけだし, 構造や機能の推定をすることが可能となる. この1次元のデータの解析を行う際 (特にアミノ酸において), まず最初に行わなければならない操作がアライメントであり, これは, 配列を観察しやすくするために, 各アミノ酸間にギャップ (配列の一部が失われたり, 新たな断片が入り込んだことに対応するもの) を挿入し, もとのアミノ酸配列の順序を保存したまま配列を並び換える操作である. 例えば, 次のような生物 **A**, **B** の同じ機能を持つタンパク質におけるアミノ酸配列について試みることにする.

A: MNPQYSTWQYT

B: MNPQYTVWPY

この **A** と **B** の間に進化の過程に於いて, アミノ酸の変化 (置換, 欠落, 挿入) が起こったとする考え方が妥当である. そのため, その変化の場所に * (ギャップ) を挿入してみると,

A: MNPQYST*WQYT

B: MNPQY*TVWPY*

となり, **A** と **B** の類似性が視覚的に捉えることができる. このように, 特に2つのアミノ酸配列を対象としたアライメントをペアワイズアライメント, 3本以上のアミノ酸配列を

対象としたアライメントをマルチプルアライメントという.

一般にペアワイズアライメントのアルゴリズムでは, 全てのギャップの入り方を考慮して計算するが, マルチプルアライメントでギャップの入り方を全て考慮すると, その計算コストは配列の長さを L , 配列の本数を N としたときに $O(L^N)$ となり, アミノ酸配列の本数の指数関数的に増えてしまい事実上計算不可能となる. そのため, マルチプルアライメントにおいては, ギャップの入り方の全てについては考慮しなくて済むアルゴリズムが必要となる. そのため, シミュレーテッドアニーリングを用いた方法 [3] などが知られている.

2.シミュレーテッドアニーリングを用いたマルチプルアライメント

この節では, 組み合わせ最適化問題で用いられるシミュレーテッドアニーリング [4] について簡単に述べ, それを用いたマルチプルアライメントについて説明する.

いま, X を空でない有限集合とする. 有限集合 X の要素を基本状態と呼ぶ. そして, 有限集合 X 上の実数値関数

$$f: X \rightarrow \mathbb{R}$$

を考える. この関数 f を目的関数と呼ぶ. このとき f を最小にする X の要素 (状態) をシミュレーテッドアニーリングを用いて求める. まず, X 上での f の最小値を $f_{\min}$ とし, $f_{\min}$ を与える X の要素の集合を $X_{\min}$ と定義する.

[定義 2-1]

$$f_{\min} \equiv \min\{f(x) | x \in X\}$$

$$X_{\min} \equiv \{x | x \in X; f(x) = f_{\min}\}$$

次に、確率分布 $q_0(x)$ を次のように定義する。この q_0 は最適分布と呼ばれる。この最適分布を発生させることができれば、 f を最小にする X の要素を求めることができる。

[定義 2-2]

$$p_0(x) \equiv \begin{cases} \frac{1}{|X_{\min}|} & (x \in X_{\min}) \\ 0 & (\text{その他}) \end{cases}$$

ただし、 $|X_{\min}|$ は $x \in X_{\min}$ となる x の個数で

ある。そして、正のパラメータ β と X 上の関数 f を用いて関数 $p_\beta(x)$ を次のように定義する。

[定義 2-3]

$\forall x \in X:$

$$p_\beta(x) \equiv \frac{1}{Z} \exp\{-\beta f(x)\} \quad (\beta \geq 0)$$

$$Z \equiv \sum_{x \in X} \exp\{-\beta f(x)\}$$

この p_β を、パラメータ β におけるギブス分布と呼び、このギブス分布と最適分布の間には、次の定理が成り立つことが知られている。

<定理 2-1>

$$\lim_{\beta \rightarrow \infty} p_\beta(x) = p_0(x)$$

このことより、 β を十分大きくすれば、ギブス分布は最適分布に収束することが分かる。

次に、シミュレーテッドアニーリングを用いたマルチプルアライメントについて説明する。まず、一般的に n 本のアミノ酸配列 $\mathcal{A}_1$ から $\mathcal{A}_n$ を以下のように定める。

$$\begin{aligned} \mathcal{A}_1: & a_1^1 a_2^1 \cdots a_{k_1}^1 \\ \mathcal{A}_2: & a_1^2 a_2^2 \cdots a_{k_2}^2 \\ & \dots \\ \mathcal{A}_n: & a_1^n a_2^n \cdots a_{k_n}^n \end{aligned}$$

ここで、 a_j^i はアミノ酸配列 $\mathcal{A}_i$ の j 番目のアミ

ノ酸を表す。最長のアミノ酸配列の長さと同じになるように各アミノ酸配列の後ろにギャップを挿入して配列の長さを等しくする。次に、アライメントを行うために十分な数のギャップを追加する。

$$\begin{aligned} \mathcal{A}_1: & a_1^1 a_2^1 \cdots a_L^1 \\ \mathcal{A}_2: & a_1^2 a_2^2 \cdots a_L^2 \\ & \dots \\ \mathcal{A}_n: & a_1^n a_2^n \cdots a_L^n \end{aligned}$$

アミノ酸配列を計算機で摂動させるためには、そのままの状態では困難なため、アミノ酸配列のアミノ酸を“1”，ギャップを“0”として数列に置き換える。例えば以下になる。

$$\begin{aligned} \mathcal{A}_1: & \text{MNPQ**} \rightarrow \mathcal{A}_1': 111100 \\ \mathcal{A}_2: & \text{MPQ***} \rightarrow \mathcal{A}_2': 111000 \end{aligned}$$

この数列に対して摂動を行い摂動済みの配列を得る。具体的には、無作為に選択した数列（アミノ酸配列）の、無作為に選択した要素“1”（アミノ酸）と、“0”（ギャップ）を交換する。例えば、 $\mathcal{A}_2'$ の2番目の“1”と1番後ろの“0”が選択されると、

$$\begin{aligned} \mathcal{A}_1': & 111100 \rightarrow \mathcal{A}_1: \text{MNPQ**} \\ \mathcal{A}_2': & 101001 \rightarrow \mathcal{A}_2: \text{M*P***Q} \end{aligned}$$

となり、摂動を行ったアミノ酸配列が得られる。この配列に対して目的関数を求める。目的関数としては、さまざまなものが考えられるが、ハミング距離をベースとした次の式がある。

$$f(x) = \sum_{k=1}^L \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{d(a_k^i, a_k^j)}{n C_2}$$

ただし、

$$d(a_k^i, a_k^j) = \begin{cases} 0 & (a_k^i = a_k^j) \\ 1 & (a_k^i \neq a_k^j \text{かつ} (a_k^i \neq * \text{かつ} a_k^j \neq *)) \\ w & (a_k^i \neq a_k^j \text{かつ} (a_k^i = * \text{または} a_k^j = *)) \end{cases}$$

とする。ここで、 w はギャップの重みを表し、一般的に2.0と取られることが多い。

この目的関数からその摂動を受理するかどうか判断する。受理する場合には摂動済みの状態を、受理しない場合は摂動前の状態を採用する。この操作を受理行列の β の値を大きくしながら繰り返すことによって、アライメント結果を得る。

3.目的関数の補正

2節に述べた目的関数においては、例えばA(アラニン)がG(グリシン)に置換されても、P(プロリン)に置換されても、その置換におけるアミノ酸同士の距離は1であり、違いが現れない。しかし、実際にはそれは不自然なことであり、タンパク質を形成する20種類のアミノ酸は任意の形をとれないことから、この節ではラムチャンドランプロットを用いてアミノ酸同士の距離を求めアライメントの目的関数に利用する方法を述べる。

タンパク質の主鎖を構成する結合部分は様々な角度に折れ曲がっており、多くのコンホメーションをとる。 $C_\alpha-C$ 結合、 $C_\alpha-N$ 結合及び $C-N$ 結合のまわりの3箇所では回転することができるということになる。しかし、 ω はペプチド結合の二重結合性の問題や側鎖のぶつかりを避けるために ± 180 度程度の値とすることが多い。このことよりタンパク質の主鎖の構造は、 ϕ と ψ の2つで記述することができる。そのために (ϕ, ψ, ω) のことを二面角という。 ω が固定されているので、 C_α の両側に2枚の平面がありその回転角が ϕ と ψ であると考えることができる。この二つによってタンパク質の主鎖の立体構造が決定されている。実際には主鎖と C_α に付随する側鎖の間には立体的な障害があり、許される $C_\alpha-N$ 結合まわりの回転角 ϕ 、 $C_\alpha-C$ 結合まわりの回転角 ψ の値は制約されるため、そ

の分布はアミノ酸ごとにある領域に限定されている。この回転角 (ϕ, ψ) をそれぞれ縦軸横軸方向にプロットした図をラムチャンドランプロットと呼ぶ。

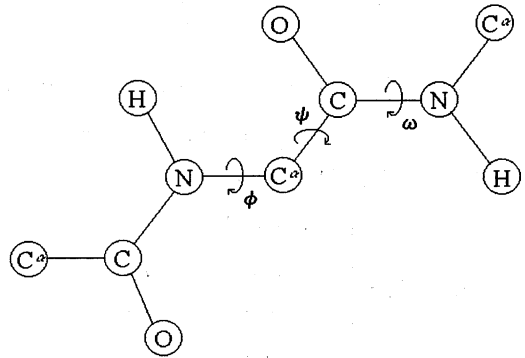


図 3.1 タンパク質の主鎖の二面角

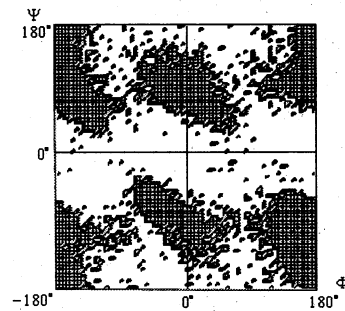


図 3.2 グリシンのラムチャンドランプロット

2つのアミノ酸の角度分布を用いてエントロピー進化率を計算し、正規化したものを距離とする。

エントロピー：

$$S(a) = -\sum_{i=1}^N \sum_{j=1}^N \frac{n_{ij}^a}{N_a} \log \frac{n_{ij}^a}{N_a}$$

$$S(b) = -\sum_{i=1}^N \sum_{j=1}^N \frac{n_{ij}^b}{N_b} \log \frac{n_{ij}^b}{N_b}$$

相互エントロピー：

$$I(a, b) = \sum_{i=1}^{N \times N} \sum_{j=1}^{N \times N} r_{ij} \log \frac{r_{ij}}{p_{ij} q_{ij}}$$

対称エントロピー比：

$$r(a|b) = \frac{I(a,b)}{S(a)}$$

$$r(a,b) = \frac{\{r(a|b) + r(b|a)\}}{2}$$

エントロピー進化率： $d'(a,b) = 1 - r(a,b)$

そしてこのアミノ酸間の距離行列を用いて次のように目的関数の補正をする。

$$d(a_k^i, a_k^j) = \begin{cases} 0 & (a_k^i = a_k^j) \\ 1 + \alpha \{d'(a_k^i, a_k^j) - \beta\} & (a_k^i \neq a_k^j \text{かつ} a_k^i \neq * \text{かつ} a_k^j \neq *) \\ w & (a_k^i \neq a_k^j \text{かつ} (a_k^i = * \text{または} a_k^j = *)) \end{cases}$$

ここで、 $0 \leq \alpha, \beta \leq 1$ であり、 α は角度による距離の影響の大きさを決定するパラメータで、その値が大きいほど影響は大きくなる。 β は角度による距離がアミノ酸間の距離をどのように見せるかを決定するパラメータで、その値が0に近いほど遠く見せ、1に近いほど近く見せる。

4.対応する部分を固定するアライメント

タンパク質には、ある原子団と相互作用したり、タンパク質の構造を安定にする役割を担ったりする重要な役割を果たす部分が存在する。この部分に当たるアミノ酸は、ある固有の機能を実現するための特定の性質を持ったものでなければならない場合がある。もしこの特別な役割を果たしている部分のアミノ酸が、欠落したり、他のアミノ酸に置換されてしまうと、その生物は正常な生体反応を維持できなくなり、長期間生存することができなくなる可能性がある。現在生存している生物、あるいは化石として発見される生物は、ある程度の期間にわたって生息した生物であるから、そのような重要な部分にあたるアミノ酸は、進化しても保存されるか、似たような性質のアミノ酸に置換されているかのいずれかと考えられる。このような、重要なアミノ酸の部分には構造保存部位や α -ヘリックス、 β -シートなどの2次構造があげられる。

そのため、このようなアミノ酸については、優先して合わせてからアライメントを行えば、同じ位置に揃いアライメント結果を補正できると考えられる。この節ではその方法について説明する。

固定するアミノ酸の位置を縦に揃えて摂動させるために、固定するアミノ酸の前後に存在するアミノ酸とギャップの数が増えないようにする。

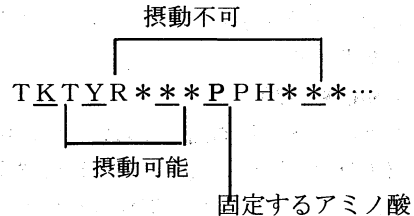


図 4.1 摂動の方法

よって、固定するアミノ酸の前後のアミノ酸配列には別々のギャップを挿入し、それらを別々のアミノ酸配列と見なしアライメントを行えば良いということになる。例えば、次のようなアミノ酸配列があったとする。

A : TKTYPPHFDLSHGSA
B : TKTYPPHDLSHGSA
C : QRFPESFGDLSPDA
D : QRFPESFGDLSTPDA
E : RFPPESFGDLSTADAV

ただし、このアミノ酸配列の下線の引かれているアミノ酸が構造保存部位で、あらかじめ固定を行う部分だとする。すると、最初に固定する部分で区切られた各アミノ酸配列に必要なだけのギャップを挿入する。また、対応が分かっていないアミノ酸配列（ここでは、**C** の後ろの部分）についても、初期状態でアミノ酸の対応が大きいくずれないように、同様にギャップを挿入する。**C** のみが2本の配列として扱われ、その他の**A**, **B**, **D**, **E** については、3本の配列として扱われ全部

で14本の配列と見なせる。この状態を初期状態としてアライメントを行うと、14本の配列からランダムに1本の配列を選び摂動を加え、目的関数を計算することができる。

長さが異なる小さな配列への摂動の振り分けかたとして、今回は、初期状態においてそれぞれの配列のギャップの入り方の総数に比例した確率で摂動を振り分ける。 N 本のアミノ酸配列のうち、 i 番目のアミノ酸配列に摂動が割り振られる確率 p_i は、次のようになる。

$$p_i = \frac{(l_i + m_i) C_{m_i}}{\sum_{k=1}^N (l_k + m_k) C_{m_k}}$$

ただし、 l_i, m_i は、それぞれ i 番目のアミノ酸配列に含まれるアミノ酸の数とギャップの数である。

アニーリングアライメントにおいては、ギャップの入り方の総数が少ないと解が求めやすい。そのため、この方法により、構造保存部位を考慮しない場合に比べギャップの入り方の総数を減らすことができ、解が求めやすくなる。実際に、アミノ酸配列にアミノ酸が l 個、ギャップが m 個、対応する部分が n 個あるとする。このとき、対応する部分を考慮に入れない場合、ギャップの入り方は $(l+m)C_m$ 通りとなる。これに対して、対応する部分を考慮に入れるとギャップの入り方は最大でも $(l+m-n)C_m$ 通りである。また、この配列は $n+1$

本の独立な配列と見なせるので、 i 番目($i=1,2,\dots,n+1$)の配列におけるアミノ酸の数を l_i 個、ギャップの数を m_i 個と置くと、ギャップの入り方は、

$$\prod_{i=1}^{n+1} (l_i + m_i) C_{m_i} \quad \left(\sum_{i=1}^{n+1} l_i = l - n, \sum_{i=1}^{n+1} m_i = m \right)$$

である.. これらより、あらかじめ対応する部分を考慮に入れたアライメント方法の方が最適値により早く収束するということが言える。

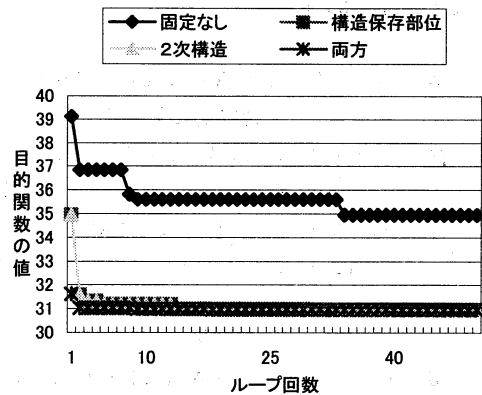


図 4.2 固定別による収束状況の相違

5. 固定するセグメントの検出

3節の特殊なアミノ酸の部分 considering アライメントを行う方法では、アライメントを行う前に固定するアミノ酸が複数のアミノ酸配列に対して判っている必要がある。しかし、アライメントを行う段階において必ずしも、そのような位置がわかっているとは限らない。特にシーケンスアライメントの場合はタンパク質の立体構造がわかっていない場合に行われるので、そのような場合には上記のアルゴリズムを適用することができないことがある。そこで、固定するセグメントを検出する方法について述べる。

次のような配列を例にしてマルチプルアライメントアルゴリズムを説明する。

A : G F I L A T P A A A A F C Q K L Q I

B : Y S H F A A A B I Q E F L A R W

C : G V M F A T T L A A B A M D V T

アミノ酸配列の中からどれか1つを選択し、選んだ配列中の任意の位置から連続するいくつかの配列を選択し、部分配列 (セグメント) とする。

この際、どの位置の連続数個を選択するかは、色々と試行してみる必要がある。対象となっている部分配列に対して、対象となっている配列以外の配列と総当たりに距離を計り、

最小となる部分を検索する。

A : GFILATP **AAAA** FCQKLQI

選択したセグメント

B : YSHF **AAAB** IQEFLARW

AAAA-----→

C : GVMFATTLAABAMDVT

AAAA-----→



A : GFILATP **AAAA** FCQKLQI

B : YSHF **AAAB** IQEFLARW

C : GVMFATTL **AAC** AMDVT

検索でヒットしたセグメントを固定しギャップを補充する。これによって1つのアミノ酸配列が2つのパートに分けられる。

A : GFILATP* **AAAA** FCQKLQI*

B : YSHF***** **AAAC** IQEFLARW

C : GVMFATTL **AAC** AMDVT*****

このアミノ酸配列に対して、シミュレーテッドアニーリングを用いたアライメントを行う。

6. 検証実験と考察

まず、以下のデータを用いて5節のアライメントを行ってみる。

A : VLSPADKSNVKA **AWGK** VGG

B : VLSAADKGNVKA **AWGK** VGG

C : VHLTPEEKNAVTTTLWGKVN

A HAGEYGAEALER

B HAAEYGA

C VDEVGGEALGR

すると、最適解は次のようになる。

A : V*LSPAD**K**SNVKA**AWGK**VG

B : V*LSAAD**K**GNVKA**AWGK**VG

C : VHLTPEE**K**NAVTTTL**W**GKV*

A GHAGEYGAEALER

B GHAAEYG***A***

C NVD*EVGGEALGR

しかし、総当たりに距離を計るためにセグメント検索するとき、最初に対象としたセグメントの位置とあまりにかけ離れた位置にヒットしてしまうことがある。このとき、アライメントを行い得られた結果が以下の通りである。

9

A : V*LSP*****A***D**KSN

B : V*LSA*****A***D**KGN

C : VHLTPEEKNAVTTTLWGKVN

19

12

A VK**A**AWGKVGGHAGEYGAEA

B VK**A**AWGKVGGH***A***AE*

C VDE**E**V*G**G*****E**A**

22

A LER

B YGA

C LGR

結果は好ましくないものとなってしまう、その原因は固定するセグメントが最初に指定したものとは比べて上記のように大きく離れた位置が固定するセグメントになってしまうことにありと考える。そこで、固定するセグメントを検索する際に、次の方法で距離を計算し、離れた位置にあるアミノ酸に対してペナルティーをもうけることとする。

比較するアミノ酸をそれぞれ a, b とする。

$d(a_i, b_j)$ は今までと同様にアミノ酸間の距離、 l は区間の長さとする。また、 p_a, p_b はそれぞれのアミノ酸が前からの何番目かを表すものとする。

$$s(a, b) = \begin{cases} 0 & (a = b) \\ 1 + d(a_i, b_j) + \frac{(p_a - p_b)(a_i \neq b_j)}{l} & (a_i \neq b_j) \end{cases}$$

離れた位置に出現するアミノ酸に対して重みをつけることによって、妥当なアミノ酸列をヒットさせることができる。

実験は次のデータを用いて行った。

A : VLSPADKTNVKA AWGKVGA
B : VLSPADKSNVKA AWGKVGG
C : VL SAADKGNVKA AWGKVGG
D : VHLTPEEKNAVTTLWGKVN
E : VHLTPEEKSAVTALWGKVN
F : MLTAE EKA AVTAFWSKVHV

A HAGEYGAEALERMFLSFPT
B HAGEYGAEALERMFLSFPT
C HAAEYGAEALERMFLSFPT
D VDEVGGEALGRLLLVYPW
E VDEVGGEALGRLLLVYPW
F DEVGGEALGRLLLVYPW

アミノ酸配列1

固定されるセグメントは以下のようになる。

A : VLSPADKTNVKA AWGKVGA
B : VLSPADKSNVKA AWGKVGG
C : VL SAADKGNVKA AWGKVGG
D : VHLTPEEKNAVTTLWGKVN
E : VHLTPEEKSAVTALWGKVN
F : MLTAE EKA AVTAFWSKVHV

A HAGEYGAEALERMFLSFPT
B HAGEYGAEALERMFLSFPT
C HAAEYGAEALERMFLSFPT
D VDEVGGEALGRLLLVYPW
E VDEVGGEALGRLLLVYPW
F DEVGGEALGRLLLVYPW

アミノ酸配列1の固定するセグメント

A : IVGGYKCEKNSQPWQVAVINEYLCGGVLIDPS
B : IVGGQECKDGECPWQALLINEENEGFCGGTIL
C : VVGGTEAQRNSWPSQISLQYRSGSSWAHTCG
D : IVNGEEAVPGSWPWQVSLQDKTGFHFCGGS
E : IVGGYTCGANTVPYQVSLNSGYHFCGGSLS

A WVITAAHCYSNNYQVLLGRNNLFKDEPFAQ
B SEFYILTAACHCLYQAKRFKVRVGDNRTE QEE
C GTLIRQNWVMTAAHCVDRELTFRVVVEGHN
D INENWVVTAAHCGVTTSDV VVAGEFDQGSS
E QWVVSAAHCYKSGIQVRLGEDNIN VVEGNE

B GGEA
C LNQNGTEQ
D SEKIQ
E Q

アミノ酸配列2

固定されるセグメントは以下のようになる。

A : IVGGYKCEKNSQPWQVAVINEYLCGGVLIDPS
B : IVGGQECKDGECPWQALLINEENEGFCGGTIL
C : VVGGTEAQRNSWPSQISLQYRSGSSWAHTCG
D : IVNGEEAVPGSWPWQVSLQDKTGFHFCGGS
E : IVGGYTCGANTVPYQVSLNSGYHFCGGSLS

↑ ① ↑ ①'
A WVITAAHCYSNNYQVLLGRNNLFKDEPFAQ
B SEFYILTAACHCLYQAKRFKVRVGDNRTEQEE
C GTLIRQNWVMTAAHCVDRELTFRVVVEGHN
D INENWVVTAAHCGVTTSDV VVAGEFDQGSS
E QWVVSAAHCYKSGIQVRLGEDNIN VVEGNE

↑ ②
B GGEA
C LNQNGTEQ
D SEKIQ
E Q

①, ② : 固定するセグメント

①' : 補正前の目的関数による *D* セグメント①

アライメントを行うとそれぞれの目的関数の収束状況は図 6.1, 図 6.2 の用になり、セグメントを固定してアライメントするとその収束

が早くなることが見て取れる。しかし、指定するセグメントの位置と長さが問題で、アミノ酸配列の端寄りのセグメントを指定した場合と、中央寄りのセグメントを指定した場合とでは、後者の方がギャップの入り方の組み合わせが少なくアライメントの効率も上がるため、最初に指定したセグメントがどの位置にあるかが問題である。また、最初に指定するセグメントの長さが短いとそのセグメントに対して検索でヒットするセグメントの候補が多く存在することになり、妥当性の低いものもヒットしやすいと考えられる。逆に最初に指定するセグメントの長さが長いとアライメントにおける摂動の自由度が小さくなり固定するセグメントが邪魔をして目的関数の最小値が大きくなってしまふことが考えられる。

そこで、4節でのアライメントの方法では複数のアミノ酸配列に対して何らかのタンパク質の立体構造がわかっている必要があったが、例えばアライメントを行いたいアミノ酸配列に対して、1つでもその2次構造がわかっている場合、(Brookhaven Protein Data Bankなどで検索可能)その特徴的な構造の部分はアミノ酸の連続であり、ほかのアミノ酸配列にもその部分と相同性の高い部分があると考えられるため、その部分をセグメントに指定してアライメントを行うとよいと考えられる。

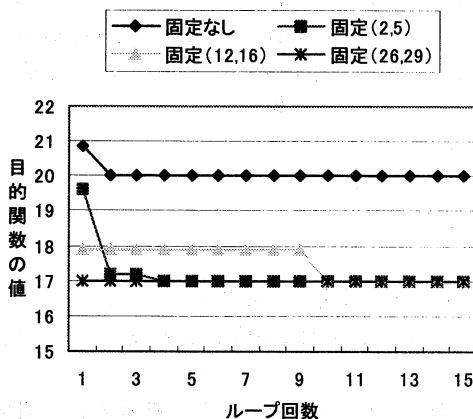


図 6.1 アライメント 1 の収束

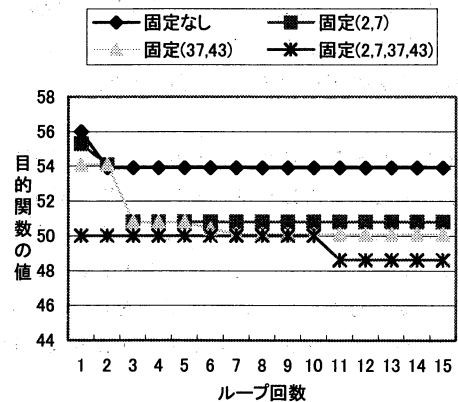


図 6.2 アライメント 2 の収束

文献

- [1] Ohya M. and Uesaka Y.: "Amino Acid Sequences and DP Matching", Information Science 63, pp.139-151(1992)
- [2] Ohya M. and Miyazaki S. and Ohshima Y.: "A new method of alignment of Amino Acid Sequences", Viva Origino 17, pp.139-151(1989)
- [3] Ohya M. and Miyazaki S. and Ogata K.: "On Multiple Alignment of Genome Sequences", IEICE 6, pp.453-457(1992)
- [4] Aarts E. and Korst J.: "Simulated Annealing and Boltzmann Machines", John Wiley and Sons(1989)
- [5] M.Ohya; Information theoretical treatment of genes, The Trans. of the IEICE, Vol.E 72, No.5, pp.556-560(1989)
- [6] 三宅成樹・金久實;"ヒトゲノム計画と知識情報処理", 培風館(1995)