岡山理科大学紀要 第40号 A pp. 121-128(2004)

# マルチプルアライメントに対する改善法

畝　哲広・片山　謙吾*・成久　洋之*
岡山理科大学大学院工学研究科情報工学専攻
*岡山理科大学工学部情報工学科

(2004年9月30日受付、2004年11月5日受理)

## 1. まえがき

現在，分子生物学分野の発展によりタンパク質のアミノ酸配列を読み取ることが可能となり解析が進められている．これらの解析により，治療法の存在しない病気の治療法の解明や未知のウイルスに対するワクチンの開発などの医学的利用，生物の進化の過程でどのような遺伝的変化が起こったかなどの生物学的利用が可能となる[8, 10, 2]．実際の解析は，タンパク質の構造を調べ特徴を導き出すことである．タンパク質の構造は，アミノ酸が連なった状態の一次構造，タンパク質の部分的な規則構造である二次構造そして，タンパク質の空間的な折り畳みによる立体構造である三次構造と分類できる．本研究では，その中でも基本的な構造である一次構造を対象とするアミノ酸配列のアライメントを行う．アライメントとは，複数本のアミノ酸配列に対しギャップを配列中に挿入し，一致もしくは類似するアミノ酸が縦に揃うように並べる方法である．アライメントにより，類似する部分が縦に揃い比較した配列の間にある関連性が導き出せる．その関連性から，配列の構造や機能などが推測される．もともとアライメントは，生物学者がタンパク質の進化についての知識を用いて手作業で行っていたため，大規模な問題を扱うことはほぼ不可能であった．しかし，コンピュータの発展により手作業で行っていたアライメントをコンピュータで行うことが可能となった．それに伴い，情報科学分野におけるアイディアを用いた様々なアライメント手法が提案されるようになった．

本研究では，現在提案されているアライメント手法がギャップを挿入することだけで最適なアライメントを求めていることに注目した．実際のアライメントにおいて，確かにギャップを挿入することにより最適なアライメントを求めるのであるが，アルゴリズムの仕様によっては挿入されるべきではない場所にギャップが挿入されてしまう場合が多く見られる．このような状態になってしまった場合，不必要なギャップを取り除くことで最適なアライメントに近づくことができると考えられるが，実際このような手段を持ち合わせている手法はほとんどない．そこで，ギャップを段階的に削除することによりアライメントを改善する手法を提案する．その方法は，解の近傍探索に基づいており，すでにある程度過剰にギャップが挿入されている状態を対象とする．なぜなら，既存の手法と違いギャップを挿入することを行わず削除のみでアライメントの改善を行うため，削除を行う分のギャップが必要であるからである．また，提案法の有効性を検証するため動的計画法をベースとした遺伝的アルゴリズムによるアライメント手法に提案法を導入し実験を行った．実験結果から提案法の有効性と他のアライメント手法にも導入可能であることを示す．

## 2. アミノ酸配列とアライメント

タンパク質は20種類のアミノ酸からなり，それぞれのアミノ酸は以下のアルファベット1文字で表される．

A, C, D, E, F, G, H, I, K, L
M, N, P, Q, R, S, T, V, W, Y

また，アミノ酸配列はこのアルファベットを一列に並べた形で表される．このアミノ酸配列を解析に利用する場合，一本のみを扱うのではなく複数のアミノ酸配列を対象とする．アミノ酸配列二本を対象とすることをペアワイズアライメントと呼び，三本以上を対象とすることをマルチプルアライメントと呼ぶ．一般に，アライメントを行う配列の本数が多いほどアライメントが困難であるといわれている．

アミノ酸配列の特徴を調べるために行われるのがアライメントである．アライメントとは，アミノ酸配列の一致もしくは類似する部分を縦に揃える操作である．アライメントによってアミノ酸が縦に揃えられた部分が配列の特徴といえる．アライメントによって特徴付けられる理由として，アミノ酸配列が進化（変化）する際その配列の一部にのみ変化が生じるということが挙げられる．つまり，進化前と進化後では一部分の変化しかないということになり，一部分の変化しかない配列同士は類似した構造や特徴を持っているのではないかと考えられる．主な変化としては，新たにアミノ酸が追加される挿入，アミノ酸が抜け落ちる欠失，アミノ酸が類似したアミノ酸に変化する置換がある．アライメントを行う際に欠失が起きたと思われる部分に挿入するのがギャップである．

アライメントの結果，類似する部分が縦に揃うことにより，配列間にある関連性が示される．しかし，得られたアライメントの結果が正しくなければ間違った結果を示してしまうこととなる．そこで，正しいアライメント結果であるかどうかを示す方法としてアライメントスコアを用いることでアライメントの正しさを調べることができる．アライメントスコアとは，アミノ酸置換行列と呼ばれるアミノ酸同士の類似性を示した行列で規定されるアミノ酸類似度を足し合わせたものである．また，スコアのつけ方によってはアミノ酸類似度の足し合わせに加えて，ギャップに対してギャップペナルティを科す方法もある．アミノ酸置換行列にはいくつかの種類があり，Dayhoff による PAM 行列[1]，S.Henikoff と J.G.Henikoff による BLOSUM 行列[5] などがある．本研究では，BLOSUM 行列の中でもアライメントでよい結果が得られるといわれている BLOSUM50 を用いてアライメントスコアの計算を行っている．ペアワイズアライメントにおけるアライメントスコアは以下の式で求めることができる．

$$S(m) = G + \sum_i S(m_i)$$

ここで，$m$ はアライメントを，$m_i$ はアミノ酸配列の $i$ 番目を，$S(m_i)$ は $i$ 番目のアミノ酸類似度を表している．$G$ はギャップペナルティに対応する関数値である．本研究では，ギャップペナルティとしてアフィンギャップペナルティ[4] を用いる．アフィンギャップペナルティは，ギャップを開始する際に与えられるペナルティ $d$ と，すでにあるギャップを伸ばす際に与えられるペナルティ $e$ の二種類のペナルティを用いる．長さ $g$ のギャップに対するペナルティは以下の式で求められる．

$$G = -d - (g-1)e$$

同様にマルチプルアライメントのスコアは以下の式で求めることができる．

$$S(m_i) = G + \sum_{k<l} s(m_i^k, m_i^l)$$

$m_i^k$ はマルチプルアライメントの $k$ 本目のアミノ酸配列の $i$ 番目を表している．このようなスコア方法を，SP スコア（sum of pairs スコア）という．得られたアライメントスコアが高いほど最適なアライメントに近いといえる．

最適なアライメントとは，アライメントスコアが最も高くかつ類似するアミノ酸が縦によく揃っているものである．図 1 のマルチプルアライメントをアライメントした結果が図 2 と図 3 である．図中の “*”は同一のアミノ酸が縦に揃っていることを表している．両方のアライメント結果を比較すると，図 2 の方が揃っている部分が多い．それに対し図 3 は，図 2 よりも揃っている部分が少なくさらにギャップが多く見られる．最適なアライメントは，少ないギャップで類似する部分を縦に揃えているものがほとんどである．それゆえ，いかに類似した部分を少ないギャップで縦に揃えるかが重要となってくる．

```
NLFIYALKDVAQD
NIYALKVAQD
NYYALKDVAD
NARYALKVDSD
NFKYALKVASD
```

図1　アライメント前

```
NLFIYALKDVAQD
N--IYALK-VAQD
NY--YALKDVA-D
NAR-YALK-VDSD
NFK-YALK-VASD
*    ****  *   *
```

図2　優れたアライメント結果

```
NLFIYALKDVA----QD
N--IYALK-VA----QD
NY--YALKDVA-----D
N----AR-Y-ALKVDSD
NFK-YALK-VA----SD
*     *     *     *
```

図3　あまりよくないアライメント結果

## 3. 動的計画法を用いた遺伝的アルゴリズム

### 3.1　動的計画法（Dynamic Programming：DP）

DPとは，組合せ最適化問題における厳密解法の一つであり，アライメントにおいて用いるDPは最短路問題におけるDPを応用したものである[11]．またこのDPは，提案者の名に因んでNeedleman-Wunsch法とも呼ばれる．このDPをペアワイズアライメントに適用した場合を説明する．

2つの配列を図4のような2次元のネットワークの辺に対応させる．斜め方向のアーク（矢）は，そのアークの位置に対応する2つのアミノ酸の類似度を割り振る．また，縦および横方向のアークは，ギャップに対応し，ギャップスコアを割り振る．最適なアライメントを求めることは，このネットワーク上の最良の経路を求めることに対応している．最良の経路は，左上の端から右下の端に向かって，各ノードに至る最良経路を段階的に決定していくことにより求めることができる．太いアークで表された経路が最良とする場合，この太いアークで表された経路を順に見ていくと，AとAが対応し，Iギャップが対応，MにはMが対応という具合になる．結果として図4の右下にあるアライメントが得られる．

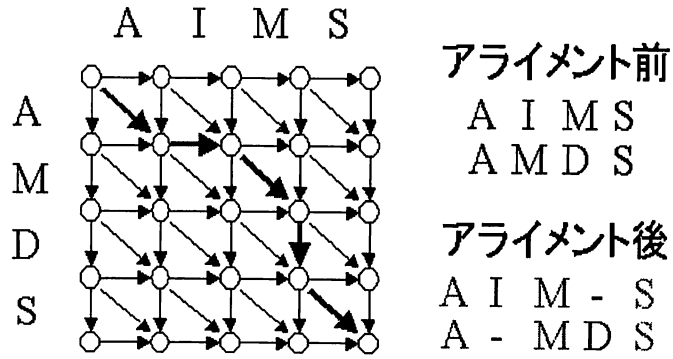


図4　動的計画法によるアライメント

このDPはマルチプルアライメントに対しても利用可能であるが，配列の本数に比例してネットワークの次元数が増加し計算量が増大する．長さ$L$の配列$n$本をアライメントすると，最短経路を求めるための計算量は$O(2^n L^n)$となるため，3本までのアライメントが実用的である．しかし，計算量を削減することで3本以上のアライメントを可能にする改良型DPとしてMSA[9]がある．詳細についてはここでは省くが，7本程度のアライメントが可能である．それでも，さらに多くの本数のアライメントには対応できないため，別のアプローチが模索された．その方法の一つとして考案されたのが，アライメントを分割しペアワイズアライメントのDPを適用する方法である[3, 13]．DPを適用する配列の本数を2本とすることで，計算量の削減を達成している．本研究で用いるアライメント手法は，この考えに基づいている．

### 3.2 遺伝的アルゴリズム（Genetic Algorithm：GA）

GAとは，生物個体群が世代交代を繰り返して，環境に適応していく過程を模擬したもので，繁殖・交配・突然変異・自然淘汰の過程を解空間内の探索に当てはめたものである[15]．以下に本研究で設計を行うGAの解のコーディングについて示す．

解：マルチプルアライメントを表す2次元配列
解の長さ：マルチプルアライメント全体の長さ
配列数：アミノ酸配列の総本数
配列長：アミノ酸配列の長さ
この解の情報を元に以下の一連の操作でGAを行い解を改善させる．

**初期解生成**

マルチプルアライメントを親の数だけ$n$個複製し，その複製したアミノ酸配列群の中で，一番配列長の長いものと同じ長さになるようにその他の配列のランダムな位置にギャップを挿入する．

**交叉**

$n$個の親の中からランダムに2つ解を選択し，それぞれの解をランダムに2つの配列群に分割する．そして，その分割した配列群の片方を交換し配列間DP[7]を用いて融合を行い，子を2個生成する．最終的には，子は$n$個生成される．

**突然変異**

生成した$n$個の子に対し突然変異率に従い突然変異を行う．突然変異はDPを利用して，配列を類似性の高いものから順にツリー状に組み合わせDPを行う．

**大突然変異**

20世代解集団に変化が見られない場合，$n$個の親の中で最良の解以外の$n-1$個の解それぞれに対して，次の操作を行う．$n-1$個の解の内ランダムに選択した$n/2$個に対し，解に存在するギャップの位置をランダムに別の位置に移動させる．残りの解に対して，縦に同一のアミノ酸が揃っている部分を仕切りとして区分けされた領域内に存在するギャップの位置をランダムに別の位置に移動させ，これを全領域に対して行う．

**淘汰**

親と子の両方の解の中からスコアの高いもの$n$個を次世代に残す（エリート戦略）．

## 4. ギャップ削除による改善法

### 4.1 局所探索法

提案法は，組合せ最適化問題の解法の一つである局所探索法[15]の考えに基づくため，まずこの局所探索法について述べる．局所探索法において最も重要な概念は近傍である．近傍とは，ある解を少しだけ変化させたものの集合であり，解に変化を加え近傍解を得る操作のことを近傍操作と呼ぶ．局所探索法は近傍操作により得られた解の中から良い解をひとつ選び出し，それを新たな解としてさらにその解に対して近傍操作を行い，さらに優れた解がないかを探索するということを繰り返し行うことで最適な解を求めてゆく方法である．これをマルチプルアライメントに当てはめると，近傍はギャップの位置を変化させることにより得られる新たなアライメントであり，近傍操作はギャップの位置を変える操作と置き換えることができる．

### 4.2 1Gap-Delete

多くの手法で用いられている最適なアライメントを求める方法は，ギャップを挿入することによりアライメントスコアの高いものつまり最適なアライメントを求めようとする方法である．実際にこの方法論は優れているといえるが，この方法とは逆にすでにいくつかギャップを挿入しておき，そのギャップを段階的に削除することも有効ではないかと考えた．つまり，次々とギャップを詰め込んでいくのではなく，すでにギャップが挿入されている状態から余分なギャップを削っていくということである．また提案法は，一度に複数のギャップを削除の候補にするのではなく，一つ一つのギャップを対象とする．対象を一つとすることで，複数を対象とした際に生じる計算量の増大を抑えることが可能となる．

1Gap-Deleteについて説明する前に，マルチプルアライメントについていくつかの定義を行う．マルチプルアライメント全体を$M$，アライメント$M$の$k$本目のアミノ酸配列を$M^k$，アライメント$M$の$i$番目の縦一列を$M_i$，配列$M^k$の$i$番目のアミノ酸を$M_i^k$，アライメント$M$の長さを$L$とする．

以下に1Gap-Deleteの詳細について$k$本の配列のアライメントを例に述べる．

*Step*.1

長さ$L$のアライメント$M$において，$M_1$～$M_L$までについてすべて同一のアミノ酸が揃っている部分を$P$とする．また，縦に同一のものが揃っていなくとも$M_1$と$M_L$は$P$とする．$P$が$n$個あるとすると，$P=\{P_1,\ldots,P_n\}$となる．なお，$P_i$と$P_{i+1}$が隣り合う場合$P_{i+1}$を$P_i$に融合させる．この$P$により縦に分割される領域を$T$とし，$P_i$と$P_{i+1}$の間の領域を$T_i$とする．$P$が$n$個あるとすると，$T=\{T_1,\ldots,T_{n-1}\}$となる．また$T_i$の長さを$l_i$とし，$T_i$をアライメント$M$の部分アライメントとして捉えると．部分アライメント$T_i$の$k$番目の配列を$T_i^k$とする．このようにして分割した領域$T$に対して1Gap-Deleteの操作を行う．なお，以下$T=\{T_1,\ldots,T_K\}$とする．

*Step*.2

$T_1$の$T_1^1$に対し，$T_1^1$内に存在する$n$個のギャップを$g=\{g_1,\ldots,g_n\}$とする．まず$g_1$を$M_L^1$に移動させ，新たなアライメント$M'$としこのときのスコアを求める．同様の操作を$g_2$～$g_n$に対して行い，得られた結果の中で最も良いスコアを得ることができたギャップの移動を採用する（図6）．

*Step*.3

残りの$T_1^2$～$T_1^k$に対しても同様の操作を行うと$M_L$にギャップが揃う．この縦一列に揃ったギャップは，アライメント中では不要であるので削除することが可能となる．

*Step*.4

*Step*.3で得られたアライメントのスコアを削除前のスコアを比較して，スコアに改善が見られればこれを新たなアライメント$M$としてさらにギャップを削除する操作を行う．改善が見られない場合は次の領域$T_2$に対しギャップの削除を行い，これを$T_K$まで行うと縦に揃えたギャップをいくつか削除することが可能となる（図8）．

*Step*.5

$T_K$まで操作を終わった後に，再び*Step*.1～*Step*.4までの操作を行いこれ以上ギャップを削除できないようであればそこで全操作が終了となる．削除が可能であれば再び同様の操作を繰り返しギャップが削除できなくなるまで行う．

ここで，領域に分割しその領域内でのみの近傍操作を行う理由として，例えば図5のようにすでに類似する部分が縦に揃っているアライメントに対して区分けなしで1Gap-Deleteを行うとすると，ギャップを移動した際に縦に揃った部分が崩れてしまう場合があるからである．しかし，領域内でのみ近傍操作を行えばそのようなことは起きず，その部分は保持されることになる．このように制限を加えることで，すでに縦に揃った部分の破壊を食い止めることが可能となる．さらに，領域に分割した際に領域内のある配列においてギャップが存在しない場合はその領域に対しては操作を行わない（行えない）ために，分割せずに全体を対象とするよりも若干の計算時間の短縮が図られる．

しかし領域に分割することは決してよいことばかりではなく，分割する際に利用する仕切りつまり縦に同一のアミノ酸が揃っている部分が適切でない場合に問題が生じる．本来その部分は揃うべきではないのに揃っていた場合，そのままの悪い状態を保持し続けてしまうため結果としてよくないアライメント結果になってしまうことも考えられる．それを回避する方法としては，一度操作を行った後に領域に分割することに用いた仕切りを一部もしくはすべて取り払い，領域を拡張し再び操作を行うという方法が考えられる

が今回提案する方法ではこれは用いない．

また，移動を行うギャップの候補をその領域内に存在するすべてのギャップではなく，一本の配列内のギャップに限定したのは無駄な計算を省くためである．例えば，その領域内に存在するすべてのギャップに対して移動を行いそのときのスコアを得て，最良の移動を採用するとその移動したギャップのある配列はすでに移動済みとなり残りの配列の移動を調べることになる．ここで，先ほど調べたギャップを移動した際のスコアの結果が利用可能であればよいが，すでに一つの配列においてギャップの移動が行われアライメントの状態が変化しているために，再び一から調べなおす必要が出てくる．しかし，一本の配列のみを対象とするとそのようなことをする必要はなくなる．

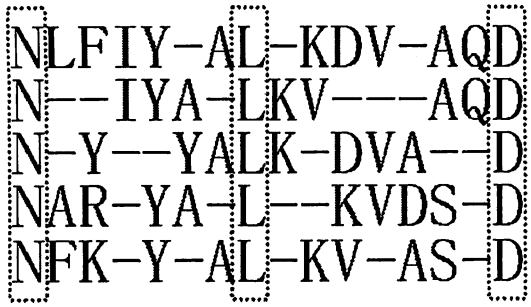


図5　複数の領域に区分け

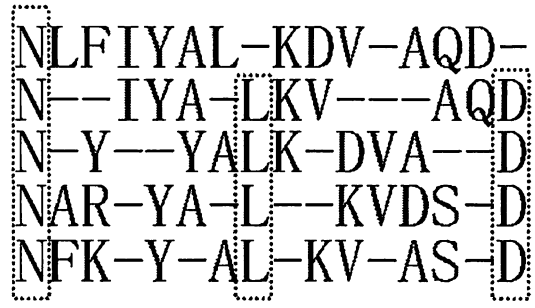


図6　ギャップを最後尾に移動

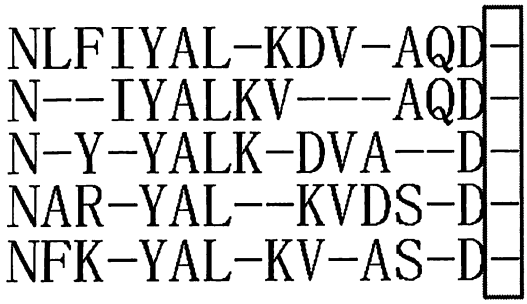


図7　右端にギャップが揃う

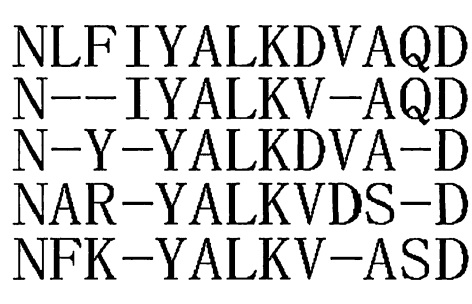


図8　操作後

## 5. 実験結果

提案する1Gap-Deleteを評価検討するため，設計したGAに1Gap-Deleteを加えたもの（1Gap-Del）と加えないもの（GA）とで，マルチプルアライメントのベンチマークであるBAliBASE[14]の問題を20題用い比較実験を行った．また，我々と同様にGAを用いたアライメント手法であり，優れたアライメントを導出することが可能であるSAGA[12]との比較も行った．GA・1Gap-Delともに突然変異率10%とし，3手法とも個体数100，100世代最良のスコアに変化が見られない場合終了し，それぞれ3回の試行を行った．表1の値はBAliBASEによるアライメントスコアと計算時間で，それぞれ3回試行の平均値である．なお，アライメントスコアは1に近いほど最適なアライメントに近いことを表している．実験環境は，CPU:Pentium4 2.0GHz，OS:Turbo Linux 7，開発言語はC言語を使用した．

実験の結果，ギャップを削除する操作を加えることでスコアに改善が見られ，アライメントの結果もギャップの入りが少なくなり，類似する部分も比較的よく揃っているという結果が得られた．このようになった要因として，アライメント中の不必要なギャップが削除できたことによりスコアに改善がみられ，この改善された解に対してGAの各操作を行っていくため導入前と比較するとよい結果が得られるようになったのではないかと考えられる．また，単純なアライメント手法であるDPをベースとしたGAであるにもかかわらず，ギャップを削除するという操作を加えることで解に大幅な改善が見られ，優れたアライメント手法であるSAGAに近い結果が得られた．このことから提案法は，有効なギャップ削除法であり他のアライメン

表1 BAliBASEの問題例に対する実験結果

| | GA | | 1Gap-Del | | SAGA | |
|---|---|---|---|---|---|---|
| Instance | Score | Time[s] | Score | Time[s] | Score | Time[s] |
| 1aab_ref1 | 0.489 | 126 | 0.545 | 139 | 0.823 | 19 |
| 1aboA_ref1 | 0.429 | 151 | 0.501 | 341 | 0.524 | 107 |
| 1csp_ref1 | 0.905 | 189 | 0.921 | 115 | 0.981 | 10 |
| 1csy_ref1 | 0.746 | 241 | 0.868 | 132 | 0.954 | 54 |
| 1dox_ref1 | 0.819 | 194 | 0.841 | 140 | 0.872 | 17 |
| 1fjlA_ref1 | 0.828 | 190 | 0.912 | 141 | 0.973 | 38 |
| 1fkj_ref1 | 0.920 | 372 | 0.944 | 91 | 0.980 | 69 |
| 1hpi_ref1 | 0.786 | 75 | 0.846 | 51 | 0.914 | 31 |
| 1idy_ref1 | 0.289 | 111 | 0.311 | 113 | 0.341 | 46 |
| 451c_ref1 | 0.608 | 152 | 0.623 | 205 | 0.652 | 85 |
| 1ad2_ref1 | 0.820 | 375 | 0.867 | 709 | 0.907 | 108 |
| 1amk_ref1 | 0.846 | 923 | 0.912 | 2061 | 0.987 | 193 |
| 1bbt3_ref1 | 0.489 | 2184 | 0.501 | 3567 | 0.642 | 591 |
| 1gdoA_ref1 | 0.808 | 1453 | 0.852 | 978 | 0.901 | 277 |
| 1havA_ref1 | 0.311 | 1088 | 0.357 | 5536 | 0.401 | 199 |
| 1mrj_ref1 | 0.816 | 565 | 0.895 | 1151 | 0.939 | 80 |
| 1pgtA_ref1 | 0.745 | 485 | 0.843 | 582 | 0.980 | 43 |
| 1zin_ref1 | 0.808 | 664 | 0.867 | 790 | 0.971 | 42 |
| 3grs_ref1 | 0.588 | 1629 | 0.642 | 1438 | 0.684 | 245 |
| kinase_ref1 | 0.586 | 6343 | 0.611 | 6475 | 0.672 | 432 |

ト手法にも導入可能であると考えられる.

## 6. むすび

アライメントの近傍探索によりギャップを削除する操作を設計し，DPをベースとするGAに操作を加え実験を行った．実験の結果，提案法を導入することでスコアに改善が見られた．このことから，提案法はアライメントの改善法として有効であるといえる．提案法はギャップを削除することのみを注目しているために，更なる改善法としてギャップの削除以外の近傍操作が必要であると考えられる．ギャップの削除以外にギャップに対するアプローチとしてこのほかに考えられるものとして，ギャップの移動と新たなギャップの挿入が考えられる．提案法ではギャップの移動を行ってはいるが，これは削除を行うために意図的に最後尾に移動を行っているので移動とは考えない．ここでいう移動とはどこかいずれかの場所に移動させるということである．また，挿入は何れかの場所にギャップを挿入することである．この移動や挿入により，さらに広い範囲での探索が可能となるのではないかと考えられるが，その分探索領域が増え計算量が増加してしまうという欠点もある．そのため，探索範囲を広げつつ有効な対象に絞って探索を行う方法を考案する必要があるといえる.

## 参考文献


1) M. O. Dayhoff, Ed, "Atlas of protein sequence and structure," National Biomedical Reserch foundation, Georgetown University, Washington, D.C., Vol.5, pp.89–99, 1972.

2) R. Durbin, S. R. Eddy, A. Krogh, G. Mitchison, "バイオインフォマティクス–確率モデルによる遺伝子配列解析–," 医学出版, 2001.

3) D. F. Feng, R. F. Doolittle, "Progressive sequence alignment as a prerequisite to correct phylogenetic trees," Journal of Molecular Evolution, Vol.25, pp.351–360, 1987.

4) O. Gotoh, "An improved algorithm for matching biological sequences," Journal of Molecular Evolution, Vol.162, pp.705–708, 1982.

5) S. Henikoff, J. G. Henikoff, "Amino acid substitution matrices from protein blocks," Proceedings of the National Academiy of Sciences of the USA, Vol.89, pp.10915–10919, 1992.

6) J. H. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

7) 石川幹人, 十時泰, 戸谷智之, 星田昌紀, 広沢誠, "並列反復改善法によるタンパク質の配列解析," 情報処理学会論文誌, Vol.35, No.12, pp.2816–2828, 1994.

8) 金久寛, "ポストゲノム情報への招待," 共立出版, 2001.

9) D. J. Lipman, S. F. Altschul, and J. D. Kececioglu, "A tool for multiple sequence alignment," Proceedings of the National Academiy of Sciences of the USA, Vol.86, pp.4412–4415, 1989.

10) D. W. Mount, "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, 2002.

11) S. B. Needleman, and C. D. Wunsch, "A general method applicable to the search for similatries in the amino acid sequence of two proteins," J. Mol. Biol., Vol.48, pp.443–453, 1970.

12) C. Notredame, D.G. Higgins, "SAGA: Sequence Alignment by Genetic Algorithm," Nucleic Acid Research, Vol. 24, 1515-1524, 1996.

13) J. D. Thompson, D. G. Higgins, T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice," Nucleic Acid Research, Vol. 22, 4673-4680, 1994.

14) J. D. Thompson, F. Plewniak, and O. Poch, "BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs," Bioinformatics, Vol.15, pp.87–88, 1999.

15) 柳浦睦憲, 茨木俊秀, "組合せ最適化–メタ戦略を中心として–," 朝倉書店, 2001.

# Improve Method for Multiple Sequence Alignment

Akihiro UNE, Kengo KATAYAMA* and Hiroyuki NARIHISA*

*Graduate School of Engineering, Okayama University of Science.*
**Department of Information and Computer Engineering, Faculty of Engineering, Okayama University of Science.*
*1 - 1 Ridai-cho, Okayama, 700-0005, Japan.*

(Received September 30, 2004; accepted November 5, 2004)

Many multiple alignment techniques are based on inserting gaps. The alignment technique based on inserting gaps does not have a measure in the case of having inserted gaps superfluously in many cases. Therefore, we consider that it is possible to obtain the high quality alignment by inserting gaps to some extent as a reverse view, and deleting gaps gradually. In this paper, we propose a gap deleting method based on this idea. We incorporate the method into the alignment technique of genetic algorithm based on dynamic programing, and we experiment with the alignment technique for benchmark problems. Experiment results show the effectiveness of the method in alignment techniques.