遺伝的アルゴリズムによるマルチプルストリングアライメント

中川裕之[†] 伊藤 実[†] 橋本昭洋[††]

マルチプルストリングアライメントとは，複数個の有限長文字列を入力とし，各文字列の適当な位置に，ギャップ文字と呼ばれる文字を挿入して，類似したパターンを揃える操作であり，生物学において，蛋白質配列を解析する際に利用される重要な技術のひとつである．遺伝的アルゴリズム（Genetic Algorithm）は，探索空間内の多数点の情報を利用した確率的探索法の一種であり，組合せ最適化問題のような，膨大な計算量を必要とする問題の近似解を，比較的短時間に見つけることができるという特長を持つ．本論文では，入力文字列集合に対して，できるだけ長い共通部分文字列を抽出するような遺伝的アルゴリズムを構成し，これを用いてマルチプルストリングアライメントを行う手法を示す．まず，遺伝的アルゴリズムによって共通部分文字列を取り出し，各文字列をその共通部分文字列の左右の部分文字列に分割する．次に，その各断片に対して，遺伝的アルゴリズムによる共通部分抽出を行う．同様の処理を繰り返すことによって，準最適なマルチプルストリングアライメントを実現する．また，本手法を用いた場合，動的計画法によるマルチプルストリングアライメントよりも処理時間は要するが，得られる解の精度が良いことを実験によって確認した．さらに，文字列の個数が大きいインスタンスを効率的に処理するために，その部分集合を取り出して，多段階に処理する手法を示す．

Multiple String Alignment using Genetic Algorithms

HIROYUKI NAKAGAWA,[†] MINORU ITO[†] and AKIHIRO HASHIMOTO[††]

Multiple string alignment is the operation to find similar patterns from a set of strings by inserting gap symbols in appropriate positions of each string. It is an important technique in molecular biology. For example, it is used for examining the relationship between the structures and functions of proteins. A genetic algorithm (GA) is a kind of search method with probability. One of the characteristics of GA is that it can find in relatively short time an approximate solution to a problem, such as combinatorial optimization, which requires high order time by a greedy method. In this paper, a new approach is presented for multiple string alignment using GA's. First, a long common substring is extracted by a GA. Second, each string is devided into three substrings: common substring, left substring and right substring. And then, a long common substring of left (right) substrings is extracted by a GA. To iterate the similar process, quasi-optimum multiple string alignment can be achieved. As a result, this method can find more accurate solution than one by a dynamic programming. And, for large instance, we divide the input strings into smaller sets of strings, and then obtain the solution by combining the sets gradually.

1. はじめに

蛋白質配列は，アミノ酸を表す20種類の英文字から成る文字配列として表現できる．蛋白質配列の解析技術は生物学などにおいて重要な位置を占めるが，アミノ酸配列の決定手法の進歩にともない，配列データベースに膨大なデータが蓄えられ，解析の自動化手法を確立することが重要な課題となっている．

配列解析技術のひとつにマルチプル（ストリング）アライメントがある．これは，各文字配列の適当な位置に有限個のギャップ文字を挿入することによって，文字配列間の相同性を最大にし，かつ，すべての文字配列の長さを等しくする操作である．たとえば，**ABCDEFG**, **ABDEF**, **BEFAG** という3本の文字配列を入力としたアライメントの出力は，**図 1** の右側のようになる．ただし，記号'-'はギャップ文字を表している．また，出

† 奈良先端科学技術大学院大学情報科学研究科
Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology
†† 大阪大学基礎工学部情報工学科
Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University

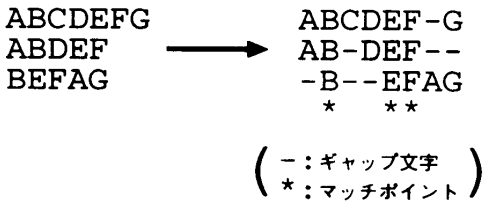

図1 アライメントの例
Fig. 1 A multiple string alignment.

力の最下行にある星印は，すべての文字配列が同じ文字で揃っている列（マッチポイント）を示している．

従来，この操作は主に生物学者の経験に頼って行われていたが，最近では，計算機による自動化も試みられている．しかし，その計算量は非常に大きく，たとえば，動的計画法によって厳密なアライメントを行う場合，平均長が l である n 本の文字配列に対して $O(n^l)$ 時間を要する．また，2つの文字配列を動的計画法によってアライメントし，その結果を順次組み上げることによって，$O(nl^2)$ 時間で近似解を得る手法もあるが[1]，組み上げる際にずれが生ずるなどの問題点がある．また，シミュレーテッドアニーリングを取り入れたアライメントの研究も行われている[2]．

遺伝的アルゴリズム（Genetic Algorithm: GA）は，自然淘汰理論に基づいて生物の進化過程を模倣した工学的モデルであり，多点情報を利用した確率的探索法の一種である．適切にGAを構成すれば，組合せ最適化問題のような，多くの計算量を要する問題の近似解を比較的短時間に見つけることが可能である．GAの性能は，染色体と呼ばれるデータ構造の定義や，それにともなう適応度関数の定義の仕方によって，大きく左右される．しかし，その方法論は確立されておらず，問題ごとに適切な符号化などを考える必要がある．アライメントにGAを応用する手法としては，動的計画法を適用するべき文字列集合を決定する前処理にGAを用いる手法[3]や，アライメントパスを求めるための補助行列に対してGAを用いる手法[4]がこれまでに提案されている．また，配列の相同性の評価には，物理的・化学的な観点から提案されたアミノ酸スコア[5]が利用されることが多い（文献2), 4), 6) など）．一方，生物学的には，できるだけ長い共通部分文字列を含むことが望ましいが，アミノ酸スコアに基づいて評価する場合，マッチポイントの連続性は考慮されないので，長い共通部分文字列があっても分断されてしまうことが少なくない．そこで，本論文では，長い共通部分文字列の抽出に主眼を置いたGAを構成し，準最適なアライメントを実現する．問題を共通部分文字列の抽出に限定すると，任意にギャップ文字を挿入する場合よりも扱うべき探索空間が小さくなるという利点もある．まず，そのような探索空間に適した染色体の符号化方法を示し，それに基づいて構成したGAによって得られた結果を評価する．さらに，入力文字配列の本数が大きい場合には，部分集合を処理対象とした多段階処理によって，探索時間を短縮する手法を用いる．

2. 準　　備

2.1 諸 定 義

文字の有限集合をアルファベットと呼び，Σ で表す．Σ に属する任意の要素を有限個並べたものを Σ 上の文字列という．文字列 $s = a_1a_2\cdots a_l$ に対し，s の連続した一部分を取り出した文字列 $a_ia_{i+1}\cdots a_{i+k}$ を s の部分文字列という．また，l を s の長さといい，記号 $|s|$ で表す．

アライメントの対象を入力文字列集合と呼び，S で表す．ある文字列 s が，S に属するすべての文字列の部分文字列であるとき，s を S の共通部分文字列と呼ぶ．S の共通部分文字列は，一般に複数個存在するが，その中で長さが最大のものを最長共通部分文字列という．

ギャップ文字が挿入された各文字列を図1の右側のように並べたものを出力配列と呼ぶ．出力配列において，すべての文字列が等しい文字で揃っている列（星印の付けられた列）をマッチポイントと呼び，その総数を M で表す．また，出力配列の横の長さをコンセンサスサイズと呼び，C で表す．図1の出力例では，$M = 3$, $C = 8$ であり，最長共通部分文字列 **EF** を含む2つの共通部分文字列 **B**, **EF** が抽出されているが，後述するように，アライメントの仕方によっては，必ずしも最長共通部分文字列が取り出されるとは限らない．

2.2 GAの概念

最適化問題をGA上にモデル化する場合，各解候補は，染色体と呼ばれる有限長文字列に符号化される．染色体を構成する各文字を遺伝子という．染色体の集合を集団と呼び，集団に属する染色体数を集団サイズという．

GAでは基本的に，**図2** に示すように，初期設定の後，適応度関数と呼ばれる実数値関数の値に基づいて，染色体の評価・更新のサイクル（世代）を繰り返すことによって解を探索する．更新処理として以下の3つの遺伝的操作を考える．

選択： 適応度に依存して，次世代に生き残る染色体を選択し，集団を更新する操作．

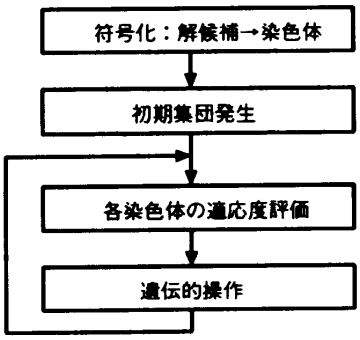

図2 GAの流れ
Fig. 2 Flow chart of genetic algorithm.

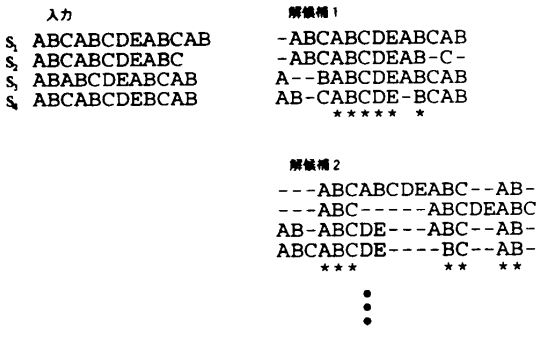

図3 集団に含まれる解候補
Fig. 3 Candidtates for optimum solution in population.

交叉： 2つの染色体（親）の一部分を交換して新しい染色体（子）を作る操作.

突然変異： ランダムに選んだ遺伝子を別の文字に書き換える操作.

上記の操作により，更新された染色体が解空間から逸脱することがあるが，これは実行不能解に相当する．このような染色体は致死遺伝子を持つという．

3. 最適化の手法

本GAでは，1つの出力配列を1つの染色体に対応させる．すなわち，各世代における各染色体は，アライメントの過程に現れる中間出力（解候補）である．したがって，ある世代における集団には，異なる配列に対応する染色体が属することになる（図 **3** 参照）．

適応度を計算する際には，各染色体に対応する配列の質の良否を反映する必要がある．配列の質の評価方法には様々なものが考えられるが，基本的に，次の2点を満たしていることが望まれる．

[**評価基準**]

(1) 長い共通部分文字列が抽出されている．

(2) マッチポイント総数 M が大きい．

図3の例では，解候補1の配列は最長共通部分文字列 **ABCDE** を抽出しており，評価基準1を満たしている．しかし，評価基準2に照合すれば，解候補2の方がよい．このように，2つの評価基準が両立しない場合があるが，生物学的には，マッチポイントが何カ所にも分散している配列よりも，長い共通部分文字列を持つものの方が意味がある．したがって，評価基準1に主眼を置いたGAをサブルーチンとして用い，二分木を構成していくことで全体のアライメントを行う．まず，できるだけ長い共通部分文字列をGAによって抽出し，二分木の根とする．次に，抽出した共通部分の左側の部分文字列から成る集合と，右側の部分文字列から成る集合をそれぞれ，新たな入力文字列集合としてGAを実行し，取り出された共通部分配列をそれぞれ，左右の子節点とする．同様の処理を，次の終了条件のいずれかを満たすまで繰り返す（図 **4** 参照）．

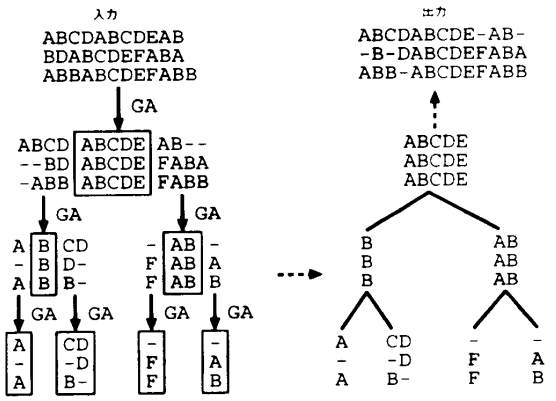

図4 アライメント木
Fig. 4 An alignment tree.

[**終了条件**]

(1) 新たな入力文字列集合が空集合．

(2) 新たな入力文字列集合の要素の最大長が1．

(3) 得られた出力配列のマッチポイント数が0．

4. GAの構成

4.1 染色体の符号化方法

文字列に対するギャップ文字の挿入方法は様々なものが考えられる．単純な例として，各文字列 s_i の任意の位置に合計 G_i 個のギャップ文字を挿入する方法を考えると，解候補の個数 V' は，長さ $|s_i|+G_i$ の文字列に G_i 個のギャップ文字を割り当てる組合せの数であるから，式(1)で表せる．この場合，非常に大きな解空間を探索しなければならず，必要な世代数が大きすぎて，処理時間の点からは実用的でない．

$$V' = \prod_i \binom{|s_i|+G_i}{G_i} = \prod_i \binom{|s_i|+G_i}{|s_i|}. \quad (1)$$

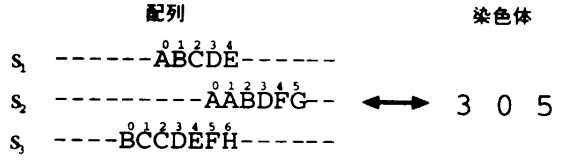

図5 染色体の符号化方法
Fig. 5 A coding scheme of chromosome.

本手法におけるGAの役割は，できるだけ長い共通部分文字列を抽出することなので，ギャップ文字を各文字列の先頭にだけ挿入すれば十分である．つまり，挿入する個数を変化させることによって，文字列を相対的にずらす操作が実現できる．ただし，他の文字列とまったく重ならない文字列が存在するようなずらし方は無意味なので，すべての文字列が少なくとも1カ所で重なるような配列だけを扱う．この制限によって，さらに探索空間が小さくなる．このような探索空間を実現するために，次のような染色体を考える．

まず，S の要素数を n とする．S に属する n 本の文字列を長さの小さい順に並べたものを $s_1, \ldots, s_n$ で表し，各文字列の文字に先頭から，番号 $0, 1, \ldots, |s_n|-1$ を付ける．このとき，$0 \leq g_i < |s_i|$ を満たすような任意の整数 g_i を n 個並べた系列 $g_1 g_2 \cdots g_n$ を染色体と定義する．つまり，g_i は s_i に属するある1文字の位置を指し示すポインタである．s_1 の g_1 番目の文字に s_i $(2 \leq i \leq n)$ の g_i 番目の文字を合わせたものを，対応する配列であると解釈すれば（図5参照），少なくとも1カ所で重なる配列をすべて含む探索空間を考えることになる．

次に，各染色体に対応する配列のコンセンサスサイズが等しくないと，後述する適応度関数による妥当な比較が行えない．したがって，最短文字列 s_1 の前後に挿入するギャップ文字の個数を固定することによって，コンセンサスサイズを等しくする．すべての文字列が少なくとも1カ所で重なるような配列が最長になるのは，(a) 最短の文字列 s_1 の左端の文字と最長の文字列 s_n の右端の文字を合わせた場合，(b) s_1 の右端の文字と s_n の左端の文字を合わせた場合，という2通りがある．そこで，s_1 の両端に $|s_n|-1$ 個のギャップ文字を挿入すれば，(a)，(b) のいずれの場合も表すことができる．このとき，コンセンサスサイズは $C = |s_1| + 2 \cdot (|s_n| - 1)$ と定義される．さらに，s_1 以外の文字列についても，長さが C に等しくなるように前後に必要なだけギャップ文字を挿入する．

以上のような符号化によって，必要なすべての解候補を表現することが可能であり，しかも，交叉処理によって探索空間から逸脱する染色体が生じることがないので，更新された染色体が解候補として適切かどうかの検査をする必要がなく，効率の面でも望ましい．また，この符号化による探索空間の大きさ V は式(2)で表され，式(1)の V' よりもかなり小さくなっている．

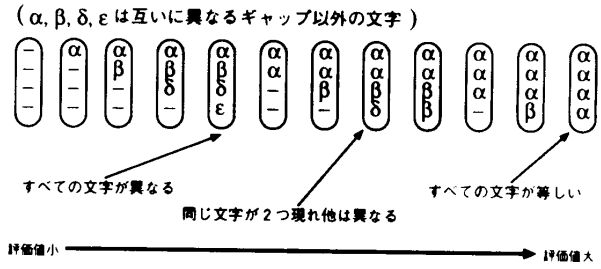

図6 列のパターン
Fig. 6 A column pattern.

$$V = \prod_i |s_i|. \tag{2}$$

4.2 適応度関数

評価基準1に基づいて探索を行うために，より長い共通部分文字列を抽出している染色体ほど高く評価されるように適応度関数 f_1 を定義する．f_1 は単純に，連続するマッチポイントの長さの最大値であると定義すればよい．しかし，たとえば，図4の第2段左側では，3番目の文字列中2カ所に現れるBのいずれを合わせればよいかを判断する必要がある．このように，入力文字列集合が同じパターンの共通部分文字列を複数個含んでいる場合や，同じ長さの共通部分文字列を抽出している染色体が複数個存在する場合，あるいは，マッチポイントが見つからない場合にも，適応度によって比較評価しなければならない．そこで，配列の相同性を評価するために，以下のような第2の適応度関数 f_2 を併用する．

まず，列ごとに相同性を評価し，その総和を配列全体の評価値と定義する．各列の評価については，同一文字の出現回数が多い列ほど高い値を与える方法が自然である．また，生物学的な意味から，ギャップ文字よりもギャップ文字以外の文字の評価を高くする必要がある．たとえば，4種類の文字を含む，4本の文字列をアライメントする場合，1つの列に出現しうる文字の個数のパターンは，**図6**のように12通りあり，図の左から右に向かって評価が高くなればよい．第 i 列に含まれる文字（ギャップ文字も考える）全体の集合を Γ_i とし，Γ_i の要素数を γ_i で表す．また，第 i 列に現れる文字 a の個数を $\mathrm{num}_i(a)$ で表すことにする．このとき，$\mathrm{num}_i(a)$ の関数 $\varphi(\mathrm{num}_i(a))$ を用い

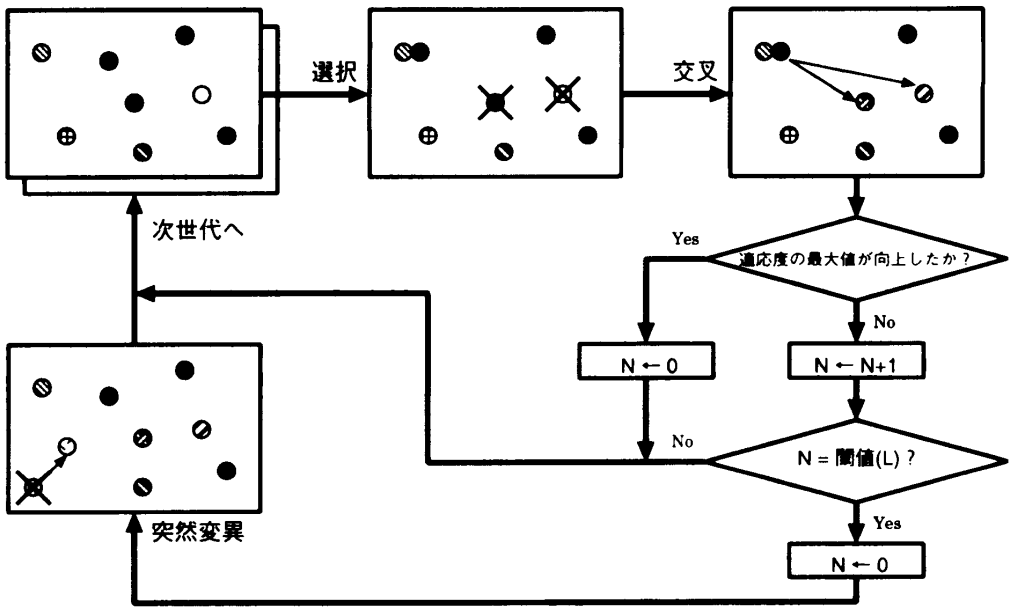

図7 本GAの流れ
Fig. 7 Outline of the GA.

て第 i 列の評価値を次式で定義する．

$$\sum_{a\in\Gamma_i}\varphi(\mathrm{num}_i(a)).$$

ここで，φ は単調増加でなければならない．また，図6の評価順序を実現するためには，一般に n 本の文字列を考える場合，γ_i 種類の文字がそれぞれ n/γ_i 回出現する場合よりも，ある1種類の文字が n/γ_i+1 回出現する場合の評価を高くする必要がある．つまり，φ は次式を満たさなければならない．

$$\gamma_i\cdot\varphi\left(\frac{n}{\gamma_i}\right)<\varphi\left(\frac{n}{\gamma_i}+1\right).$$

すなわち，$\varphi(x)$ は ${\gamma_i}^x$ よりも増加率が大きいことが必要である．さらに，ギャップ文字だけから成る列の評価値が，それ以外の1文字の評価値よりも小さくなるような重み w_i を導入すると，φ は次式で定義できる．

$$\varphi(\mathrm{num}_i(a))=w_i\cdot(\gamma_i+1)^{\mathrm{num}_i(a)}.$$

$$w_i=\begin{cases}\dfrac{\gamma_i}{(\gamma_i+1)^n} & (a=\text{'-'})\\ 1 & (a\neq\text{'-'}).\end{cases}$$

この φ を用いて f_2 は次式で定義される．ただし，C はコンセンサスサイズである．

$$f_2=\sum_{i=1}^{C}\sum_{a\in\Gamma_i}\varphi(\mathrm{num}_i(a)).$$

4.3 遺伝的操作

本GAの初期設定後の全体的な流れを**図7**に示す．各遺伝的操作は以下のように行う．

選択： 前節で定義した適応度関数 f_1 および f_2 を用いて各個体をランク付けすることで選択を行う．まず，f_1 によって集団内の各染色体を順位付けし，f_1 の値が等しい染色体が複数個あれば，それらを f_2 によってさらに順位付けする．この方法によって，抽出した共通部分文字列が長く，かつ，高い相同性を持つ染色体が高いランクになる．交叉の親として，最上位のランクから2つの染色体を選び，交叉の結果作られる2つの子と，最下位のランクの2つの染色体を置き換える（steady-state model）．

交叉： 交叉方法には様々なものが提案されているが[7]，解空間をある程度均等に探索でき，しかも，処理がそれほど複雑でないものとして，2点交叉を採用する．これは，染色体中の2点を選び，その2点に挟まれた部分を交換する方法である（**図8**参照）．ここで，もし交叉によって，致死遺伝子を生じる可能性があるならば，その検査を行う必要があり，効率の面で望ましくないが，前述の染色体の符号化方法では，2点交叉を行っても致死遺伝子を生じることはない．

突然変異： 標準的なGAでは，一定の突然変異率にしたがって頻度を制御するが，この確率が大きすぎると，それまでに得られた高い適応度を持つ染色体が壊れる可能性が高くなる．一方，突然変異は交叉だけからでは得られないパターンを生成する役割を持っており，局所解への収束を避けるためには，ある程度の頻

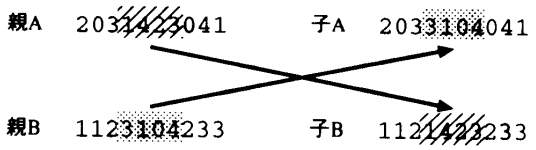

図8 2点交叉
Fig. 8 Two-point crossover.

度で実行する必要がある．突然変異をまったく行わず，交叉だけで染色体を更新する場合，適応度の最大値は階段状に増加していくが，その様子は一般に，世代の浅い間は比較的急激に増加し，世代が進むと非常に緩やかにしか増加しなくなる．すなわち，処理開始後しばらくの世代は，交叉だけでも十分な速度で適応度が向上するので，突然変異を抑制しておき，交叉による効果が小さくなるにしたがってその頻度を高めることによって，効果的に突然変異が機能すると考えられる．また，最適解やその近傍の近似解を表している染色体が突然変異によって破壊されることを避けるために，適応度最大の染色体は突然変異の対象から除外する．

交叉の効果が小さくなったことを，適応度の最大値が向上しない世代数で判断することにし，その閾値 L，および，一世代に突然変異させるべき染色体の割合 R を定めるために実験を行った．一例として，文字列の平均長が 142 であるような 3 本の文字列を入力とした場合の，200 世代までに得られたマッチポイントの総数 M と実行時間 T の平均値を表 1 に示す．この表によれば，R が大きく L が小さい方が，得られる解の精度が高くなる傾向があるが，実行時間も大きくなることが分かる．この例以外の入力に対して実験した結果でも同様の傾向が現れており，それらを総合して，ここでは $L=5, R=1/4$ と決定した．この値にしたがって，$L=5$ 世代連続して適応度の最大値が増加しなかった場合に限り，次のように突然変異を行う．まず，集団サイズの $R=1/4$ にあたる染色体をランダムに非復元抽出し，次に，選ばれた染色体のそれぞれについて 1 つの遺伝子をランダムに選ぶ．i 番目の遺伝子は文字列 s_i に対応しているので，取りうる値は 0 から $|s_i|-1$ までの整数である．この範囲から整数をランダムに選びその遺伝子の値とする．このように突然変異を行えば，交叉と同様，致死遺伝子は生じない．

4.4 時間計算量の評価

GA の時間計算量を評価するためには，集団サイズ P および世代数 G を定めなければならない．一般に，P が大きいほど 1 世代あたりの探索点が多くなるので，等しい G に対して，より広い解空間を探索できる．一方，各染色体に対して適応度を計算する必要があるので，適応度関数の計算にかかる時間は P に比例する．G についても同様の議論が成り立つ．

表 1 L, R および M, T の関係
Table 1 The relation between L, R and M, T.

	R					
	1/8		1/4		1/2	
L	M	T	M	T	M	T
5	111	12.39	111	15.79	111	25.44
10	107	10.52	110	12.56	111	15.86
15	51	5.57	111	10.15	109	13.19
20	99	11.19	93	9.98	111	11.37
25	87	10.62	108	11.18	111	10.49
30	56	10.83	95	9.94	107	10.92
35	77	9.63	77	11.10	103	10.47
40	63	9.08	100	9.93	81	11.14

L ： 突然変異を行う世代間隔
R ： 突然変異させる染色体の割合
M ： 200 世代までに得られたマッチポイント数
T ： 実行時間（秒）
（M と T は試行 50 回の平均値）

GA がランダムサーチと同程度の性能しか持っていないと仮定した場合，集団サイズ P の GA を世代数 G だけ実行すれば，$P \cdot G$ 個の解を探索することになる．したがって，式 (2) から，$P \cdot G = O(|s_{\max}|^n)$ とすれば十分である．ここで，n は S の要素数であり，$s_{\max}$ は S に属する最長の文字列を表す．しかし，このように定めると，n が大きい場合に実行時間が非常に大きくなることが問題になる．実際には，ランダムサーチに比べて，GA の方がかなり効率良く探索できるため，$P \cdot G$ を探索空間の広さほど大きくしなくても，準最適解を得ることができると考えられる．一般に，P はそれほど大きくない定数にしておき，G をインスタンスごとに設定することが多い．P の値と収束するまでの世代数，および出力の精度の関係を見るために予備実験を行い，その結果から，精度の高い解を得るために必要な世代数が比較的小さな値で安定していた P の値のうち，最も小さな値として $P=64$ と決定した．また，実行世代数の上限 $G_{\max}$ は外部から与えることにしたが，必要以上に大きな $G_{\max}$ を与えた場合に冗長な探索を避ける意味で，$G = \min(|s_{\max}|^n, G_{\max})$ と定義する．

初期設定では，乱数系列を用いて長さ C の染色体を P 個生成するので，必要な時間計算量は $O(P \cdot C)$ である．入力文字列の長さに関するソートは $O(n \log n)$ で行える．選択は，f_1 と f_2 の値を用いてランクの上位および下位それぞれ 2 つの染色体を決定すればよいので，$O(P)$ で行える．交叉は，2 つの親染色体を複写し，その一部分を交換するので，1 ペアあたり

AKCAKCAKCAKCAKCDEFAKCDAKCAKCAKCAKCDA
KCAKCAKACDAKAKCAKAAKCAKCAC

AKAKCAKCDAKCDEAKCDEFAKCAKAAKCDEFAKCA
KAKCDAKCAKCAKCDAKAKAKCDAKC

AKCDAKCAKAKAKAKAKCDEFGAKCDAKAKAKCAKK
AKCDAKCDAAKCAKAKAKCD

AKCDEAKCAKAAKCDEKAKCDEFGAKAKAKCAKCDA
KAKCDDDAKCDAKCDEAKCDE

図9 同一パターンを含む入力例
Fig. 9 An instance with many same patterns.

$O(C)$ で処理できる．突然変異は1染色体について定数時間で済むので $O(P)$ で行える．f_2 は定義式から $O(C\cdot|\Sigma|\cdot n)$ で計算できる．ただし，$|\Sigma|$ は Σ の要素数である．f_1 の値は f_2 を計算する過程で計算できるので，適応度の時間計算量は全体で $O(P\cdot C\cdot|\Sigma|\cdot n)$ になる．$C=O(|s_{\max}|)$ であること，および $G\leq G_{\max}$ が成り立つことを用いれば，GA の時間計算量 τ_{GA} は次式で与えられる．

$$\tau_{GA}=O(G_{\max}\cdot P\cdot|s_{\max}|\cdot|\Sigma|\cdot n).$$

S に属する最短の文字列を $s_{\min}$ で表すと，アライメント過程で作られる木の節点数はたかだか $|s_{\min}|$ なので，アライメント全体の時間計算量 τ は次式で与えられる．

$$\begin{aligned}\tau&=O(|s_{\min}|)\cdot\tau_{GA}\\&=O(G_{\max}\cdot P\cdot|s_{\min}|\cdot|s_{\max}|\cdot|\Sigma|\cdot n).\quad(3)\end{aligned}$$

5. 実験結果と改良

前章で構成した GA を，C 言語を用いて DEC station 5000/25 上に実装し，アライメントを行った．本章ではその結果を考察する．

5.1 入力文字列の本数が少ない場合

特に最長共通部分文字列を抽出しにくい文字列集合に対する性能を見るために，同じパターンが繰り返し現れる文字列を入力としてアライメントを行った．本 GA では，確率的処理に乱数系列を利用しているので，一般に，用いる乱数系列ごとに処理過程が異なる．したがって，同じ入力に対して異なる乱数初期値を用いて実験した結果を評価する必要がある．**図 9** は，実験用に作成した入力で，**AKC** というパターンが多数回現れる例である．この文字列集合は，長さ 6 の最長共通部分文字列（**AKCDEF** および **AKCDAK**）と，これらの部分文字列になっていないものとして，長さ 5 の共通部分文字列（**AKCAK**）を持つ．この入力に対し，異なる乱数初期値 100 個を用いてそれぞれ世代数を $G_{\max}=180$ と一定にして実行した場合の，長さ 5 以上の共通部分文字列の抽出の成否，得られた総マッチポイント数，実行時間について，結果の一部を**表 2** に示す．一方，アミノ酸スコア[5]を用いた動的計画法によるプログラム CLUSTAL V[6]では，最長共通部分文字列 **AKCDAK** が抽出されなかった．他の入力についても数本の文字列に対して，本 GA では 40～60%程度の割合で最長部分文字列の抽出に成功している．この割合は同一パターンを多数含むような極端な入力を与えた場合の数値であるが，現実のアミノ酸配列の場合は図 9 ほど偏りがないので，成功率はこれよりも高く，実験を行った限りでは 80%を下回ることはなかった．一例として，**図 10** に実際のアミノ酸配列および本手法によるアライメント結果を示す．50 種類の乱数初期値に対して，この解が得られるまで実行を続けた結果，必要な世代数，および実行時間の平均値は，それぞれ，174 世代，27.56 秒であった．平均世代数が 174 であることから，解空間の大きさと比較してかなり効率良く探索が行われていると結論できる．分子生物学での応用を考えた場合，必ずしもリアルタイムに処理する必要はなく，場合によっては 1 入力の処理に数時間以上を許容されることもあるため，この程度の実行時間であれば十分実用的であると考えられる．さらに，**図 11** は文献 4) との比較である．文献 4) では，富士通の並列計算機 AP1000（プロセッサ数 64）を用いて，アミノ酸スコア[5]に基づく並列 GA を実行している．文献 4) の手法では抽出されない最長共通部分文字列（**VQS**）を，本方法では抽出していることから，基準 1 が有効に作用していることが分かる．

表 2 アライメント結果の一例
Table 2 An example of results.

seed	Ext1	Ext2	Ext3	M	T
0	×	○	○	22	8.70
1	×	×	○	24	9.11
2	○	○	○	25	7.97
3	○	×	○	13	10.10
4	○	×	○	28	8.90
.					
.					
.					
99	×	○	○	28	9.06
合計	54	58	92	–	–
平均	–	–	–	24	8.87

seed ： 乱数初期値　　Ext1 ： AKCDEF 抽出成否
M ： マッチポイント数　Ext2 ： AKCDAK 抽出成否
T ： 実行時間（秒）　　Ext3 ： AKCAK 抽出成否

5.2 入力文字列の本数が多い場合

実際には，数十以上のアミノ酸配列をアライメントする場合が多いが，本 GA では，入力文字列の本数を増やすと，共通部分文字列の抽出能力が落ち，得られ

```
[入力]
HYAMKILDKQKVVKLKQIEHTLNEKGGEMFSHLPHDLIYRDLKPENLLIDQQGYIQVTDFGFAKRVKCAVDWWA
LGVLIYEMAAGYPPFFADQPIQIYEKIVSGKVSHFSSDLKDLLRNLLQVDLTKRFGNLKNGVNDIKNHKWF

HYAMKILNKQKVVKMKQVEHILNEKGEMFSFSEPHDLIHRDLKPENLLIDQQGYLQVTDFGFAKRVKCAVDWWA
LGVLIYEMAVGFPPFYADQPIQIYEKIVSGRVSKLRSLLQVDLTKRFGNLRNGVGDIKNHKWF

DYAMKILDKQKVVKLKQVEHTLNEKRMFSHLPHSDLIYRDLKPENLLIDSQGYLKVTDFGFAKRVKGCVDWWAL
GVLVYEMAAGYPPFFADQPIQIYEKIVSGKVSHFGSDLKDLLRNLLQVDLTKRYGNLKAGVNDIKNQKWF
```

⇓

```
[出力] (174 世代 (27.56 秒)：試行 50 回の平均値)
HYAMKILDKQKVVKLKQIEHTLNEKGGEMFSHL-PH-DLIYRDLKPENLLIDQQGYIQVTDFGFAKRVK-CAVD
HYAMKILNKQKVVKMKQVEHILNEK-GEMFSFSEPH-DLIHRDLKPENLLIDQQGYLQVTDFGFAKRVK-CAVD
DYAMKILDKQKVVKLKQVEHTLNEK--RMFSHL-PHSDLIYRDLKPENLLIDSQGYLKVTDFGFAKRVKGC-VD
 ****** ****** ** ** ****   ***   ** ***************************** *   * **

WWALGVLIYEMAAGYPPFFADQPIQIYEKIVSGKVSHFSSDLKDLLRNLLQVDLTKRFGNLKNGVNDIKNHKWF
WWALGVLIYEMAVGFPPFYADQPIQIYEKIVSGRVS------K--LRSLLQVDLTKRFGNLRNGVGDIKNHKWF
WWALGVLVYEMAAGYPPFFADQPIQIYEKIVSGKVSHFGSDLKDLLRNLLQVDLTKRYGNLKAGVNDIKNQKWF
******* **** * *** ************** **      *  ** ********* ***  ** **** ***
```

図 10 アミノ酸配列のアライメント
Fig. 10 An alignment of amino acid strings.

```
[入力]
EVQLVQS
QLVQSGG
VQSGGGV
```

⇓

[本 GA の出力] (8 世代 (0.04 秒): 試行 50 回の平均値)	[文献 4) の出力] (5 世代 (1.7 秒))
EVQLVQS----	E-VQLVQS-
--QLVQSGG--	QLVQ-SGG-
----VQSGGGV	--VQSGGGV
***	**

図 11 アミノ酸スコアに基づく GA との比較
Fig. 11 Comparison between GA based on the Dayhoff matrix and ours.

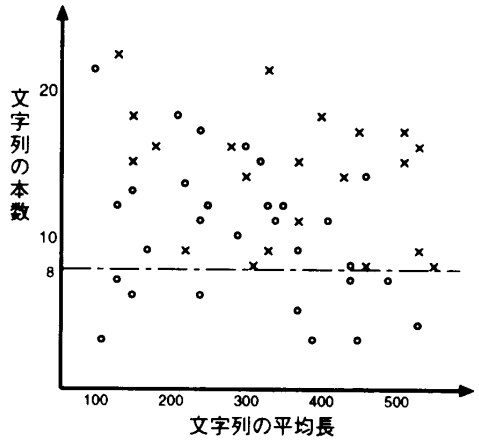

図 12 本 GA による成功と失敗の分布
Fig. 12 Distribution of success and failure.

る配列の質が低下する傾向が観察された．これは，解空間の大きさ V が本数に関して指数的に増加することによる．世代数や集団サイズを指数的に大きくすればこの問題は回避できるが，実行時間が非常に大きくなり実用的でない．そこで，次の方法によって多数本の文字列を処理する．

まず，入力文字列集合から，長さの短い順に選んだ k 本の文字列を入力として GA を実行する．このとき抽出された最長の共通部分文字列を s_0 で表す．次に，最初に選んだ k 本を除外して，別の文字列を 1 つ選び，これと s_0 の 2 本を入力として GA を実行する．このとき共通部分文字列が抽出されれば，それは s_0 自身，または s_0 の部分文字列である．これを新たに s_0 とする．以下，選ばれていない文字列がなくなるまで同様の処理を繰り返す．すべての処理が終了した後，s_0 が空文字列でなければ，s_0 は S 全体の共通部分文字列であることが保証される．ここで，最初に選ぶ文字列の本数 k が小さすぎると，選んだ k 本だけに特有な共通部分文字列を抽出してしまい，S 全体の共通部分文字列でないものが s_0 になる可能性が高くなる．したがって，k はできるだけ大きい方が望ましい．しかし，入力文字列の文字の並び方にはばらつきがあり，k を理論的に定めるのは困難なので，実験結果をもとにして定めることにした．

平均長および本数の異なる 50 の入力に対して，本 GA を実行したときに，最長共通部分文字列の抽出の成否をプロットしたものが**図 12** である．図中の○と×はそれぞれ，抽出に成功した場合と失敗した場合を表している．最悪の場合，8 本の文字列に対して失敗していることから，最長共通部分文字列を発見できると見込まれる本数として，やや小さめに $k=5$ とする．このように k を定めると，まず，最初に 5 本の文字列について GA を実行し，次に 2 本ずつ $n-5$ 回の

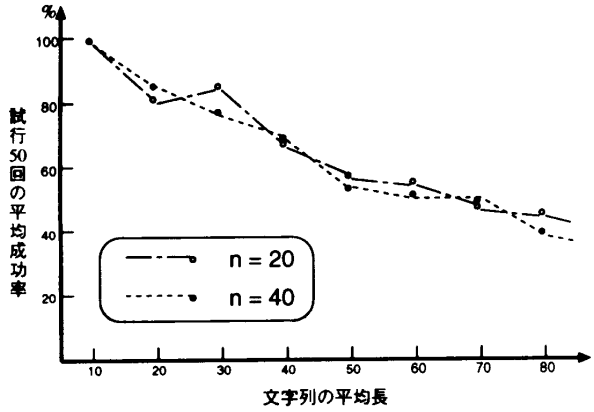

図 13 最長共通部分文字列の抽出成功率
Fig. 13 Percentage of extracting a longest common substring.

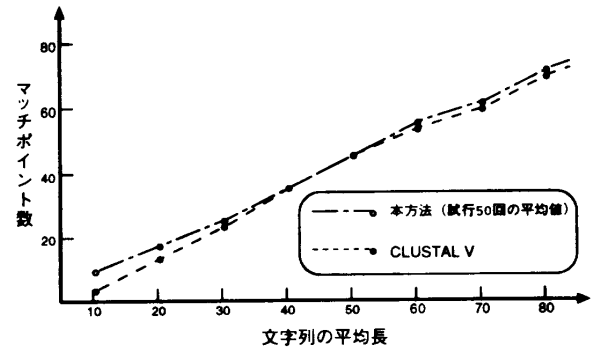

図 14 CLUSTAL V との比較
Fig. 14 Comparison between CLUSTAL V and our GA.

GA を実行することになるので，時間計算量 τ_s は，

$$\tau_s = O(G_{\max} \cdot P \cdot |s_{\min}| \cdot |s_{\max}| \cdot |\Sigma|) + O(G_{\max} \cdot P \cdot |s_{\min}| \cdot |s_{\max}| \cdot |\Sigma| \cdot n). \quad (4)$$

となり，最初の方法の時間計算量 τ（式 (3)）と一致する．

図 9 と同様の 20 本および 40 本の文字列に対して，この手法を用いてアライメントを行った場合に，最長共通部分文字列を抽出できた割合の平均値（試行 50 回）を**図 13** に示す．また，得られた解の精度の指標として，20 本の入力に対して得られたマッチポイント総数を CLUSTAL V[6] と比較したグラフを**図 14** に示す．

6. むすび

本論文では，蛋白質のアミノ酸配列の集合に対して，GA を利用して共通部分文字列を取り出すことによってマルチプルアライメントを実現する手法を示した．

問題点として，共通部分文字列を持たない入力文字列集合に対しては，まったくアライメントできないことがあげられる．たとえば，ABCD,AC, BD をアライメントする場合，最適なものは**図 15** (b) の配列であるが，本方法では (a) の配列しか得られないということである．

```
ABCD        ABCD
AC--        A-C-
--BD        -B-D
```

(a) (b)

図 15 共通部分文字列を持たない例
Fig. 15 An instance without common substrings.

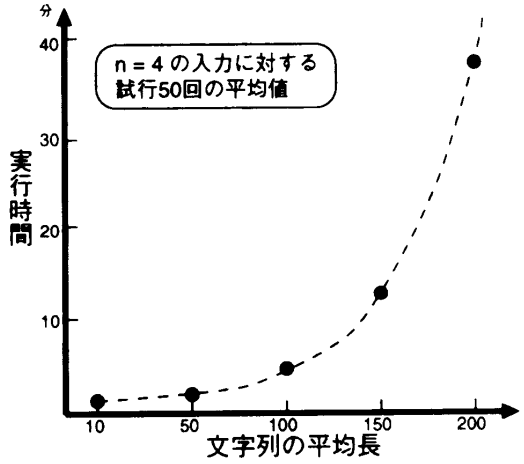

図 16 f_2 による GA の処理時間
Fig. 16 The time required by GA based on f_2.

同様の問題は，アライメントの中間段階において共通部分文字列を取り出した残りの部分文字列から成る集合が共通部分文字列を持たない場合にも生じる．これを解決する方法として，このような場合には GA 以外の方法を適用するか，あるいは，ギャップ文字を任意の位置に挿入した配列を染色体として符号化し，適応度関数 f_2 を用いて染色体を評価するような，別の GA を用いることが考えられる．実際そのような GA を構成し実験を行った結果，文字列の平均長が十分に小さい入力に対しては精度の高い解を得ることができた．しかし，式 (1) の大きさを持つ解空間を扱うため，**図 16** に示したように膨大な実行時間がかかることや，処理できる文字列長や本数に上限があることなどから，そのままでは実用上問題がある．

次に，入力文字列の本数が大きい場合に，最初の入力として短い順に 5 本の文字列を選んだが，改良の方向として，ホモロジーマトリクス法[8]などを取り入れることで，より相同性の高い文字列を選び，効率を向上させることが考えられる．

実用性の観点から，与えられた入力文字列集合に対して，必要な世代数の最小値があらかじめ予想できないので，ある程度試行錯誤的に実行しなければならないという問題がある．しかし，G_{max} を予測するためには，入力文字列の長さや本数だけでなく，文字列を定性的に分析する必要があり，解決が困難な問題である．

本手法の効率改善の可能性としては，二分木を構成していく過程で，入力がある長さ以下になった段階以降は，全点探索などのGAを用いない手法に切り替えることが考えられる．この場合，本GAよりも切り替えた方法の方が効率的でなければ無意味であるから，本GAの実行時間以下で実行可能で，かつ解の精度が同程度以上になるような入力サイズを決定する必要がある．

最後に，今後の課題として，文献5)のアミノ酸スコアのような，生物学的な意味を加味した配列の評価を行い，より有意なアライメントを行うことがあげられる．本論文では，文字の評価を一致不一致だけで行っていたため共通部分文字列の長さを評価できたが，生物学的な意味を加味した場合にも共通部分文字列の評価が妥当にできるような手法の確立が必要である．

謝辞 多くの貴重なご意見をいただいた査読者の方々に感謝いたします．また，分子生物学に関して多くのご助言をいただいた大阪大学情報処理教育センターの中西通雄助教授にお礼申し上げます．

参考文献

1) Fang, D.-F. and Doolittle, R.F.: *J. Mol. Evol.*, Vol.25, pp.351-360 (1987).
2) 広沢 誠，星田昌紀，石川幹人，戸谷智之：MASCOT：3次元ダイナミックプログラミングに基づいた蛋白質のアライメントシステム，「情報学」シンポジウム (1992).
3) 戸谷智之，石川幹人，星田昌紀，小長谷明彦：遺伝的アルゴリズムを取り入れたタンパク質配列解析，第47回情報処理学会全国大会論文集，pp.373-374 (1993).
4) Tajima, K.: Multiple Sequence Alignment Using Parallel Genetic Algorithms, *Proc. Genome Informatics Workshop IV*, pp.183-187 (1993).
5) Schwartz, R.M. and Dayhoff, M.O.: *Atlas of Protein Sequence and Structure*, Vol.5, National Biomedical Research Foundation, p.353 (1978).
6) Higgins, D. Bleasby, A. and Fuchs, R.: CLUS-TAL V: Improved Software for Multiple Sequencealignment, *CABIOS*, Vol.8, No.2, pp.189-191 (1992).
7) 北野宏明（編）：遺伝的アルゴリズム，産業図書，pp.13-15 (1993).
8) 日本生化学会（編）：遺伝子研究法I，東京化学同人，pp.381-421 (1986).

(平成7年7月21日受付)
(平成8年5月10日採録)

中川 裕之

平成6年大阪大学基礎工学部情報工学科卒業．平成8年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．現在，同大学院博士後期課程に在学中．主に，計算機科学の分子生物学への応用に関する研究に従事．

伊藤 実（正会員）

昭和52年大阪大学基礎工学部情報工学科卒業．昭和54年同大学院修士課程修了．同年同大学基礎工学部助手．平成5年奈良先端科学技術大学院大学情報科学研究科教授．工学博士．主に，関係データベース，オブジェクト指向データベースの設計理論に関する研究に従事．電子情報通信学会，ACM，IEEE Computer Society 各会員．

橋本 昭洋（正会員）

1966年大阪大学大学院工学研究科通信工学専攻博士課程修了．工学博士．同年NTT電気通信研究所に勤務．1969～1971年イリノイ大学計算機科学科客員助教授．1985年NTTデータ処理研究部長，1987年情報科学研究部長．この間計算機の故障診断，自動設計，大型計算機DIPSの開発等に従事．1989年大阪大学基礎工学部情報工学科教授，1994年同大学情報処理教育センター長を併任，1996年改組により同大学院基礎工学研究科情報数理系教授，現在に至る．最近は分子生物学関連の情報処理技術の研究に従事．著書：計算機アーキテクチャ（1995年，昭晃堂）．電子情報通信学会，IEEE，ACM 各会員．