LA-3

DNA 配列におけるプローブの順序付けに必要な最小フラグメント集合
Minimum Fragments for Deciding Probe Sequences for DNA Strands

田村武幸* 土田大輔 * 伊藤大雄 * 岩間一雄 *
Takeyuki Tamura Daisuke Tsuchida Hiro Ito Kazuo Iwama

1 はじめに

ゲノム解析とは、1 個体の持つ DNA 全体の塩基配列を決定することである。ゲノムの塩基数が膨大 (ヒトゲノム約 30 億の塩基対からなる) である一方、シーケンサで DNA の塩基配列を高精度に解読できるのは数百塩基程度である。そこで、ゲノム全体を数百塩基程度のフラグメントに分断し、それぞれのフラグメントをシーケンサにかけ塩基配列を決定したあと、それらを繋ぎ合わせるという方法が取られる。フラグメントとは DNA の部分配列のことである。

実験により、塩基の文字列がそのフラグメントに含まれるかどうか調べることができる。この塩基の文字列のことをプローブという。例えば図 1 では、フラグメント 2 にプローブ B とプローブ G が、フラグメント 3 にプローブ G とプローブ E が存在する。よってプローブの順序は BGE(あるいは EGB) であることがわかる。

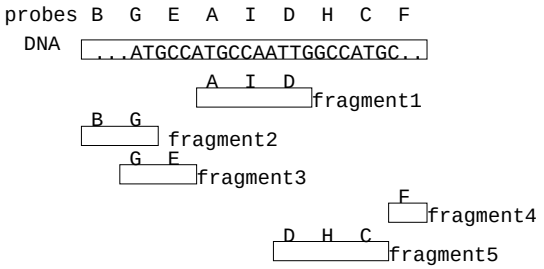

図 1: プローブとフラグメントによるゲノム解析

どのプローブがどのフラグメントに存在するかによって、プローブやフラグメントの順序が決定されていく。例えば図 1 の例では、BGE や AIDHC がそれぞれ近くにあることが決定される。しかし正確なフラグメントやプローブの順序までが必ずしも一意に決定されるわけではない。例えばプローブ A,I はプローブ D を狭んで、プローブ H,C とは反対側に存在するということは決定されるが、AI は IA であってもかまわないし、HC は CH であってもかまわない。また BGE や AIDHC や F の位置関係も決定されないし、BGE は EGB かもしれない。

フラグメントの順序を決定する問題は古くから論じられている [1]。しかしフラグメントの順序だけでなくプローブの順序までも一意に決定することができれば、その後の解析における効率化につながると考えられる。我々はプローブの順序を一意に決定するために、あとどのくらいのフラグメント数が必要かについて議論する。プローブの順序を一意に決定するために必要なフラグメント集合の性質を解析し、必要な最小フラグメント数を示した。

*京都大学情報学研究科
{tamura, tsuchida, itohiro, iwama}@kuis.kyoto-u.ac.jp

2 定式化

図 1 においてプローブを列、各フラグメントを行、フラグメントの存在するところを 1、存在しないところを 0 とすれば、(1) のような 0/1-行列で表現することができる。フラグメントの順番を決定するためには (2) のような、各行の 1 が連続している行列を導かなければならない。

$$\begin{pmatrix} A & B & C & D & E & F & G & H & I \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad (1)$$

$$\begin{pmatrix} B & G & E & A & I & D & H & C & F \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \quad (2)$$

行列の各行の 1 を全て連続させる列の並べ替えが存在するかどうかを判定し、存在する場合は、その列の並び順を示す線形時間のアルゴリズムが存在する (Kellog ら [2])。

Kellog ら [2] のアルゴリズムは、解さえあれば各行の 1 を連続させられる。ところが 1 が連続したからといって、プローブの順序まで一意に決定されるわけではない。(2) の行列は明らかに AI や HC を IA や CH にできるし、BGE-AIDHC-F を BGE-F-AIDHC にしてもよい。ある 0/1-行列が与えられた時にプローブの順序を一意に決定するためには、最低いくつのフラグメントを追加しなければならないだろうか。

3 解法

Kellog ら [2] のアルゴリズムでは PQ 木と呼ばれるデータ構造が用いられる。PQ 木とは、並べ替えを表すデータ構造で、P 節点と Q 節点と葉節点から構成される。P 節点は子節点の順番を任意に変更してよいことを表す節点であり、図で表すときは丸で表し、Q 節点は子節点の全体の順番を前後ひっくり返してもよいことを表す節点であり、図で表すときは矩形で表す。例えば行列 (2) を PQ 木で表現すると図 2 のようになる。

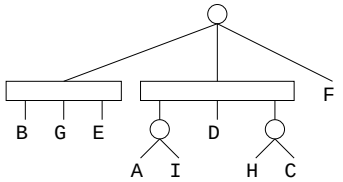

図 2: 行列 (2) に対応する PQ 木

図 2 の PQ 木は図 3 のように、(A,B,C,D,E,F,G,H) に (1,0,0,1,1,0,0,1,1) と (0,0,1,1,0,1,0,1,1) という 2 つのフラグメントを追加すれば、木が変形しプローブの順序を一意に決定できる。1 つの Q 節点のみからなる PQ 木を 1Q 木と呼ぶ。ある PQ 木にフラグメント集合を追加して、プローブの順序が一意に決定できるようになれば、図 3 のように 1Q 木へと変形しているはずである。図 3 の変形において、1 つ

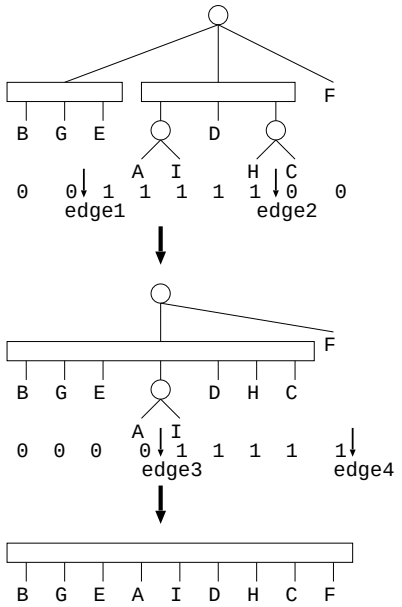

図 3: 図 2 の PQ 木の 1Q 木への変形

目のフラグメントでは GE 間と HC 間、2 つ目のフラグメントでは AI 間と F の右側に 0 と 1 の境界があると考えられる。この境界をエッジと呼ぶ。任意の隣接するプローブ間はエッジになる可能性がある。また最左のプローブの左側、最右のプローブの右側もエッジになる可能性がある。よって 2 つのエッジを指定すれば、1 つのフラグメントを特定することができる。

与えられた PQ 木を与えられた葉の順序を変えずに 1Q 木に変形する操作を**固定操作**と呼ぶことにする。

補題 1 固定操作には以下のエッジを端にもつフラグメントの追加が必要であり、両端ともそれ以外のエッジであるフラグメントの追加は不要である。

- タイプ *1:P* 節点の子で隣接する葉の間のエッジ。
- タイプ *2:*子節点をもたない根でない *Q* 節点の下のどこかひとつのエッジ。
- タイプ *3:*与えられた *PQ* 木の左端 *(*右端*)* が根 *P* 節点を親にもつ葉である時、その左 *(*右*)* 側のエッジ。

例えば図 3 のエッジ 1 はタイプ 2、エッジ 2 とエッジ 3 はタイプ 1、エッジ 4 はタイプ 3 である。補題 1 で選ばれたエッジのことを**必須エッジ**と呼び、それ以外のエッジのことを**不要エッジ**と呼ぶことにする。次の補題はフラグメントの追加によって、必須エッジの数を確実に減少させることができることを示している。

補題 2 間に必須エッジを含むような、*2* つの必須エッジに挟まれたフラグメントを追加すれば、変形後の *PQ* 木において、それらのエッジは不要エッジになる。また変形前に不要エッジであれば、変形後も不要エッジのままである。

例えば図 2 において必須エッジは 4 つあるが、図 3 でまず、GE 間と HC 間のエッジから構成されるフラグメントが追加され、ついで AI 間と F の右側のエッジから構成されるフラグメントが追加されることにより、必須エッジはすべて不要エッジとなり、1Q 木が導かれている。また、それら以外のエッジが木の変形によって、必須エッジに変化することはない。

固定操作の際には、まず PQ 木の葉の順番を目的とする 1Q 木の葉の順番と同じにしてから固定操作をほどこすものと考えてよい。葉の順番を入れ替えた後の必須エッジの数が少ないほど追加すべきフラグメント数も少ないと思われる。このことを示したのが補題 3 である。なお 1 つの P 節点に着目した時、その子節点のうち PQ 節点の数を v、葉の数を l で表すことにする。

補題 3 固定操作における必須エッジの最小値は以下の値の合計で求められる。

1. 子節点を持たない Q 節点の数。
2. 根以外の P 節点に対する、$max\{|l|-|v|-1,0\}$ の合計。
3. 根節点が P 節点であれば、$max\{|l|-|v|+1,0\}$。

例えば図 2 の PQ 木において、プローブ F を BGE と AIDHC の間に移動させれば、必須エッジは 1 つ減って 3 個にできる。

定理 1 与えられた PQ 木において補題 *3* で求められる必須エッジの数を e とする。固定操作に必要な最小フラグメント数は以下のようである。

1. $e \geq 3$ の時、$\lceil \frac{e}{2} \rceil$ 個。
2. $e = 2$ の時
   - *2* 個の必須エッジの間の葉が *1* つの時、*1* 個
   - *2* 個の必須エッジの間に葉が *2* つ以上ある時、*2* 個

4 今後の課題

以上で最小フラグメント数が得られたが、組合せるエッジによっては、とても長いフラグメントが生成されることもある。しかし、あまりにも長いフラグメントは非現実的である。長いフラグメントを短くするためには、以下の補題を用いることができる。

補題 4 長さ k のフラグメントは長さ $\lceil \frac{k+1}{2} \rceil$ の *2* つのフラグメントに置き換えることができる。

しかしこの補題を用いると、フラグメント数は増加してしまう。フラグメントの長さに上限がある時の最小フラグメント数は今後の課題である。

参考文献

[1] 21 世紀の医療・福祉を支える科学技術特集 電子情報通信学会誌 Vol.84 No.5 pp.341-367 2001 年 5 月

[2] Kellogg S. Booth and George S. Lueker. Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms: Journal of Computer and System Sciences 13, pp.335-379, 1976.