アルゴリズム 74－9
（2000．9．21）

データに誤りのある場合のDNAチップを使った配列決定アルゴリズム

土井 晃一郎　今井 浩

東京大学大学院理学系研究科情報科学専攻
〒113-0033 東京都文京区本郷 7-3-1
doi@is.s.u-tokyo.ac.jp, imai@is.s.u-tokyo.ac.jp

DNA配列を決定することはゲノムサイエンスにおいて大変重要な問題である。配列を決定する方法は様々な方法が考えられている。ここではシーケンシング バイ ハイブリダイゼーション (SBH) と呼ばれるDNAチップによって得られる部分文字列の有無のデータによる配列決定の方法について扱う。最近提唱された効率の良いDNAチップに対しては、いままでエラーのある場合については考えられていない。本研究では、DNAチップにおけるハイブリダイゼーションに誤りがある場合にも配列決定アルゴリズムを提唱して、ランダムデータに対して適用し、その有用性を示した。

A sequencing algorithm using DNA chips in the presence of errors

Koichiro Doi　Hiroshi Imai

Department of Information Science, Graduate School of Science, The University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
doi@is.s.u-tokyo.ac.jp, imai@is.s.u-tokyo.ac.jp

Sequencing DNA sequence is a very important problem in genome science. Various sequencing methods was considered. We can get information about the existence of subsequences in DNA sequence by the DNA chip. This paper deal with sequencing by the hybridization data from DNA chip, called Sequencing by Hybridization (SBH). Because the case of existence of errors was not considerd for the novel efficiency DNA chip, this paper proposes a sequencing algorithm in the presence of hybridization errors in the DNA chip, and apply this algorithm to random data, and show this algorithm is useful for error data.

1　はじめに

DNA配列のシーケンシング、つまり、DNA配列がなんであるかを決定していくことは大変重要な問題である。これに対しては様々な方法が考えられ、生物実験も数多く行われてヒトゲノムをはじめとして多くのDNA配列が決定されてきている。

ここで扱う方法はシーケンシング バイ ハイブリダイゼーション (以下、SBH) と呼ばれている手法である [2, 3, 6]。この方法はDNAチップによって得られるデータを使用してDNA配列を決定する手法である。DNAチップとはプローブと呼ばれる短い配列をたくさんくっつけたものであり、ハイブリダイゼーションを行うことにより、DNA配列上にプローブとマッチしている配列が

存在するかどうかが分かる。配列決定をしたいDNA配列とハイブリダイゼーションを行うことによりDNA配列中の部分文字列が分かるので、それによってDNA配列を決定していく手法である。この手法は、任意のある長さのDNA配列を高い確率でいかに少ない数のプローブで配列決定ができるかという理論的な解析がなされてきたが、実際には生物実験上のエラーなどの問題があり実用化はあまりなされていなかった。この手法には、どのようなプローブをDNAチップ上におくかというプローブの設計の問題と、チップから得られたデータに対してどのようなアルゴリズムを適用して配列を決定するかという配列決定問題の2つの問題がある。この論文では後者について考えている。

既存研究としては、長さ k のプローブをすべてDNAチップ上におく方法が考えられたのが始まりである [2, 3, 6]。つまり、DNA配列は $\{A, T, G, C\}$ の4種類の文字からなる文字列なので 4^k のプローブを使用するものである。これについては詳しい説明ははぶくが、ハミルトンパスが1つだけか判定する問題となり、それをオイラーパスが1つしかだけしかないか判定する問題に変換できて計算量を減少させることができるという理論的にはきれいなものであるが、冗長性が高く、効率の良いものではなかった。

この後、$\{A, T, G, C\}$ だけではなく、A, T, G, C どれにもマッチするuniversal base、A, T だけにマッチする文字などを使用することによってチップの効率を理論的にあげようとする試みがなされた [3, 6]。この場合にははじめに考えられたチップのようなハミルトンパスといった性質が考えられないので、配列決定をしようとするDNA配列のはじめの部分はすでに分かっているものと仮定して、そこから1つ1つ配列を決定していこうとするアルゴリズムで性質が調べられた。これにより、単なる部分文字列を使ったものよりは効率の良いものとなったが、それほど効率のよいものではなかった。

そして最近、[4, 5] で新しいチップとそのデータに対する新しい配列決定アルゴリズムが提唱され、配列決定の能力がかなり伸びて、その情報理論的な最適性も示されている。この結果に基づいて本研究のアルゴリズムを構築している。

いままで述べた研究は、ハイブリダイゼーションがミスなしに起きた場合の話である。実用的な手法とするにはエラーが存在する場合について考えなくてはならない。最初に考えられたチップ、つまり長さ k のプローブをすべてプローブとして使った場合にはいくつかアルゴリズムが提唱されている。[2] にはfalse negativeのみが存在した場合に最小費用流問題に持っていくアルゴリズム、[1] にはfalse positive, false negativeが存在する場合に確率的なエラーのモデルを作って最尤法で解くアルゴリズムが提唱されている。しかし、universal baseを使ったようなDNAチップにはエラーの存在する場合の考慮がなされていない。

この論文では [4, 5] で提案されているDNAチップに対してエラーが存在しても動くように配列決定アルゴリズムを作成した。このアルゴリズムは [4] のエラーのないときの配列決定アルゴリズムをエラーのあるときにも適用出来るように拡張したものであり、エラーが存在していても、存在していなくても統一的に適用できるアルゴリズムである。このアルゴリズムをランダムデータに対して適用して、少量 (0.1% 程度) のエラーに関してはこのアルゴリズムは有用であることが分かった。

2 Preparata et al. [4, 5] の手法

この章ではPreparata et al. [4, 5] におけるDNAチップと配列決定アルゴリズムについて述べる。この論文ではこのチップを用いて配列決定アルゴリズムを拡張することによってエラーに対する対処をしている。

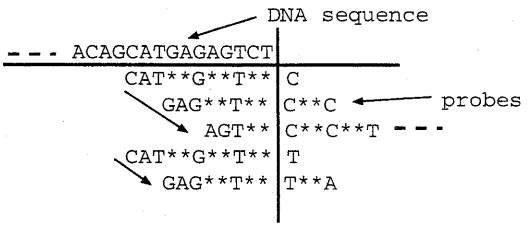

図 1: プローブの候補が複数ある場合のプローブの探索の仕方。幅優先探索のやり方で探索する。

2.1 DNA チップ

まず、この論文で扱う DNA チップは以下のようなものである。DNA 配列は A,T,G,C の 4 種類からなっているが、ここではそれに加えて* で表す don't care のように A,T,G,C の 4 種類のどれが来てもマッチする文字を使用して DNA チップを作成する。この*にあたるものは実際の生物実験でも作成することができる。

定義 1 ([4, 5]) 2 つの正の整数のパラメータ s, r を用いて (s, r) で表される *DNA* チップは以下のようなものである。
(s, r)-*probe* $= X^s(*^{s-1}X)^r$
X は A, T, G, C の 4 種類それぞれに対して全種類のプローブを作成することを表している。

よってチップのサイズは 4^{s+r} である。

例 1 $(3, 4)$ の場合には以下のようになる。
$(3, 4)$-*probe* $= XXX**X**X**X**X**X$

2.2 配列決定アルゴリズム

この配列決定アルゴリズムの入出力は以下のようになる。

入力 (s, r)-probe のうちハイブリダイゼーションをしたプローブの集合 S と決定すべき DNA 配列の長さ n と決定すべき DNA 配列の $s(r+1)$ の長さの正しい接頭辞

出力 DNA 配列が決定できたときにはその DNA 配列、決定できなかったときには決定できたところまでの配列

エラーのないときは入力はすなわち、決定すべき DNA 配列のなかに存在する (s, r) の形をした部分文字列すべての集合ということになる。また、ここで考えられているアルゴリズムは DNA 配列すべてを特定するものではなく決定すべき DNA 配列の $s(r+1)$ の長さの正しい接頭辞が与えられているときに残りの DNA 配列を決定するアルゴリズムとなっている。
これに関してエラーのない場合は以下のようなアルゴリズムが考えられている。

基本アルゴリズム

1. 1 番目から ℓ 番目までの配列が決定されているときに $\ell+1$ 番目の文字を特定することを考える。$\ell+1$ 番目に右端が来るようにおいて、$\ell - s(r+1) + 2$ 番目から ℓ 番目のすでに決定されている配列にちゃんとマッチするプローブの集合 C を S の中からすべてとってくる。

2. $|C|$ の数にしたがって以下のことを行う。

- $|C| = 1$ のときには $\ell+1$ 番目の文字は $|S|$ に含まれるプローブの右端の文字となる。この文字を $\ell+1$ 番目の文字と決定して、$\ell = \ell+1$ とする。してステップ 1 に戻る。
- $|C| > 1$ のときには C の中のこの位置にマッチするプローブの集合それぞれに対して、幅優先探索のやり方で $\ell - sr + 2$ 番目から $\ell + s + 1$ 番目までの位置にマッチするプローブの候補、$\ell - s(r-1) + 2$ 番目から $\ell + 2s + 1$ 番目までの位置にマッチするプローブの候補というように探索していく。(図 1 参照) この幅優先探索は最大深さ $rs+1$ まで行う。
  - 途中で枝が 1 本になれば配列はその枝に DNA 配列が決定され、引き続いてまだ決定されていない部分をステップ 1 に戻って決定していく。
  - 深さ $rs+1$ まで行ってまだ枝が複数存在するときには、その最大一致接頭辞が存在すればその部分の DNA 配列が決定され、引き続いてまだ決定されていない部分をステップ 1 に戻って決定していく。
  - 最大一致接頭辞が存在しなければ配列は決定出来なかったとして終了する。

3. 長さ n まですべて決定できたところで終了する。

このアルゴリズムは [3] でも考えられているような基本的に 1 つ 1 つ順番に配列の最初の方から決定していくアルゴリズムである。しかし、それだけでは次にどのプローブが来るのか分からない状況がすぐおこってしまうので、ここではプローブの候補が複数考えられる場合にその先まで読み進めることによって、次に何が来るのか分からないような状況を回避しようとしている。また、このようなアルゴリズムを行うのに適しているようにプローブも設計されている。計算量は DNA 配列の長さ n に対して DNA チップのサイズが最適なときには $O(n)$ と [4] において解析されているが、現実的にはもっと大きいものとなる。

エラーがないときにはステップ 2 で $|C| = 0$ となることはありえない。しかしエラーがあるときには対処すべきなので次章で考慮をする。

3 エラー対処アルゴリズム

3.1 エラー

エラーには簡単にいうと以下のものが考えられる。

false positive 存在しない部分文字列に相当するプローブがハイブリダイゼーションしてしまい、存在すると認識してしまうエラー

false negative 存在する部分文字列に相当するプローブがハイブリダイゼーションせずに、存在しないと認識してしまうエラー

これらのエラーが存在するときにも正しくシーケンシングが行えるようなアルゴリズムをこの論文では提唱する。また、チップに関しては前章で紹介した [4, 5] でエラーのない場合のよさが示されているものに関して考えることにする。

3.2 アルゴリズム

基本的にはエラーがあっても基本アルゴリズムで動かないことはない。しかし、エラーがない場合は次にくるべきプローブの候補が存在しないということはありえないが、false negative が存在するときにはその可能性が存在する。よってその場合にも対処できるように以下のようにアルゴリズムを拡張する。

このアルゴリズムは、エラーがないときのアルゴリズムを拡張したものであり、エラーが存在していても、存在していなくても統一的に適用できるアルゴリズムである。

エラー対処アルゴリズム

1. 1番目からℓ番目までの配列が決定されているときに$\ell+1$番目の文字を特定することを考える。$\ell+1$番目に右端が来るようにおいて、$\ell-s(r+1)+2$番目からℓ番目のすでに決定されている配列にちゃんとマッチするプローブの集合CをSの中からすべてとってくる。

2. $|C|$の数にしたがって以下のことを行う。

   - $|C|=1$のときには$\ell+1$番目の文字は$|S|$に含まれるプローブの右端の文字となる。この文字を$\ell+1$番目の文字と決定して、$\ell=\ell+1$とする。してステップ1に戻る。
   - $|C|>1$のときにはCの中のこの位置にマッチするプローブの集合それぞれに対して、幅優先探索のやり方で$\ell-sr+2$番目から$\ell+s+1$番目までの位置にマッチするプローブの候補、$\ell-s(r-1)+2$番目から$\ell+2s+1$番目までの位置にマッチするプローブの候補というように探索していく。(図1参照) この幅優先探索は最大深さ$rs+1$まで行う。
     - 途中で枝が1本になれば配列はその枝にDNA配列が決定され、引き続いてまだ決定されていない部分をステップ1に戻って決定していく。
     - 深さ$rs+1$まで行ってまだ枝が複数存在するときには、その最大一致接頭辞が存在すればその部分のDNA配列が決定され、引き続いてまだ決定されていない部分をステップ1に戻って決定していく。最大一致接頭辞が存在しなければ配列は決定出来なかったとして終了する。
     - 途中で、枝がすべてたどれなくなった場合には、1つ前までたどれていた枝のそれぞれに対して、次に右端がA,T,G,Cである4種類のプローブがマッチすると思って深さ制限つきの幅優先探索を引き続いて行う。
   - $|C|=0$のときにはその位置にマッチする右端がA,T,G,Cである4種類のプローブが存在すると思って$|C|>1$のときと同じように深さ制限つきの幅優先探索を行う。

3. 長さnまですべて決定できたところで終了する。

このアルゴリズムは次に来るべきプローブの候補が見当たらないときにそこで終了するのではなく、候補がないのだから全ての可能性 (接尾辞がA,T,G,Cの4通り) があると考えて、深さ制限付きの幅優先探索を行った。これにより、[4]のDNAチップ、配列決定アルゴリズムのよい性質をそのまま利用してエラー対処アルゴリズムを構成した。この場合の計算時間は、エラーが増えると幅優先探索を行う回数が増えるが、実際にアルゴリズムが適用できるような少量のエラーの場合はほとんど変わらないと思われる。

しかし、これでエラーがある場合にどの程度対処できているかは分からない。そこで次章でランダムな配列に対してハイブリダイゼーションのエラーを入れて計算機実験を行った。

4 計算機実験

ランダムな配列に関して前章のエラー対処アルゴリズムを適用した。DNAチップは$s=4,r=4$の場合、つまり$(4,4)$を使用した。このときのチップのサイズは4^8である。false positvieの確率P_pとfalse negativeの確率P_nをパラメータとしてあたえることにより以下のようにエラーをランダムにいれてDNAチップより得られるデータをシミュレートした。

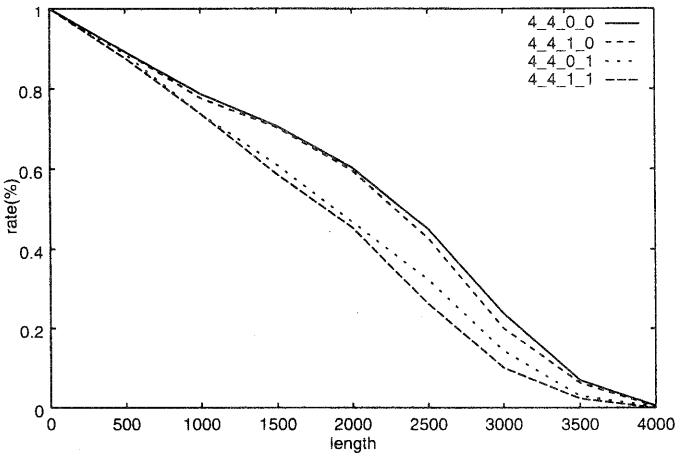

図 2: すべての DNA 配列が決定された比率 ($P_p = 0.1\%, P_f = 0.1\%$ の場合)

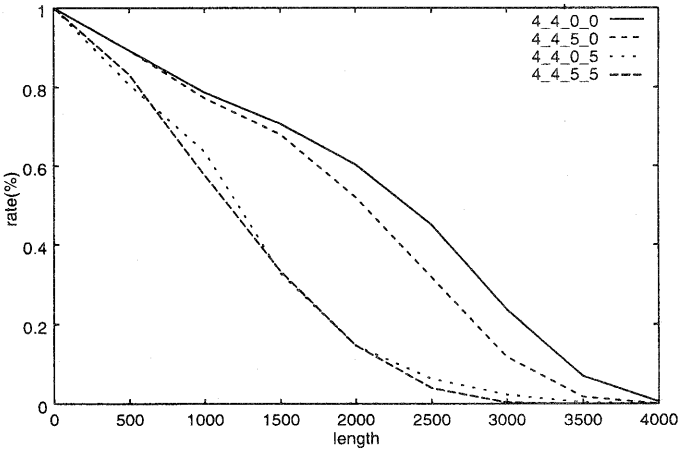

図 3: すべての DNA 配列が決定された比率 ($P_p = 0.5\%, P_f = 0.5\%$ の場合)

false positive DNA 配列中に存在する部分配列と 1 文字だけ異なる配列に相当するプローブそれぞれが確率 P_p でハイブリダイゼーションを行われたことにする。

false negative DNA 配列中に存在する部分配列に相当するプローブが確率 P_n でハイブリダイゼーションを行なわれていないことにする。

この実験結果をまとめたグラフが図 2-6 である。このグラフの線の名前は $s\_r\_1000P_p\_1000P_f$ を表している。図 2, 図 3 は DNA 配列全体が決定された比率を DNA 配列の長さを x 軸として描いたものである。false positive は 0.5%ぐらいあっても長さ 1500 で 2% ぐらい決定される比率が少なくなる程度であるが、false negative は 0.5%ぐらい入るとかなり比率に違いが出て、長さ 1500 で半分以下になってしまっている。正しいプローブがなくなってしまう false negative の方が多くなると配列を決定しにくいものとなってしまっている。また、エラーがないときに比べてエラーが存在しているときの方がグラフの傾きが大きいことが分かる。DNA 配列の長さが大きくなってエラーがなくても決定しにくくなってくるにしたがい、エラーが入ってくることに対して対処し

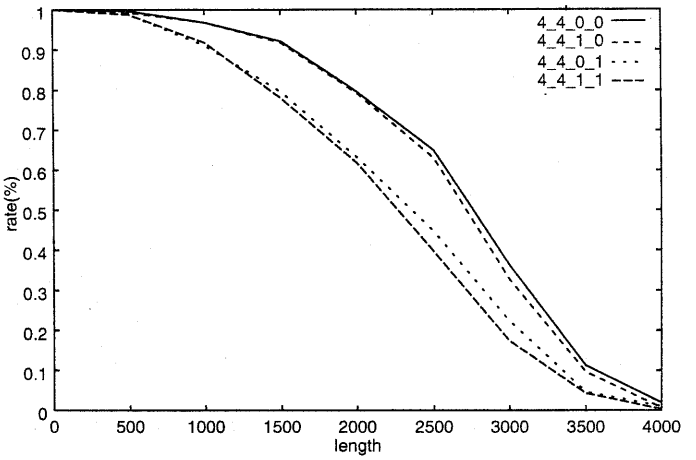

図 4: $n-s$ の DNA 配列が決定された比率 ($P_p = 0.1\%, P_f = 0.1\%$ の場合)

ていくことが出来にくくなっていることが分かる。

アルゴリズムの性質上、DNA 配列の最後の部分は決定しにくくなっている。なぜなら、最後の方では $rs+1$ の深さまで幅優先探索が行うことができないからである。そこで特に幅優先探索をまったく行うことができない部分を除いた $n-s$ の長さまで DNA 配列が決定できる比率を表したグラフが図 4 である。s はこの実験では 4 に設定している。図 2 と比較すると最後の 4 だけを考えにいれないだけで顕著に違いが表れ、エラーがない場合、長さが 1500 から 2000 ぐらいで 20% 程度、決定される割合が異なってしまっている。図 5, 図 6 は DNA 配列が何%ぐらい決定されるかを表したグラフである。これをみても、DNA 配列全体が決定出来ない場合でも全く配列決定が出来ていない訳ではなく、エラーがあっても長さ 2000 で 8 割以上配列決定出来ている場合が多いことが分かる。

5 まとめ

本論文ではエラーが存在するときに対する配列決定アルゴリズムを提唱して、計算機実験を行った。エラーがある場合でも false negative が 0.1%、false positive が 0.5% ぐらい存在していても有用に配列決定アルゴリズムが動いていることが分かった。また、この手法の場合は DNA 配列の最後の部分の配列決定の能力が他の部分に比べて落ちてしまうので、そのことを考慮にいれて配列決定を行うべきであることも分かった。エラーがある場合の理論的な解析、エラーの入り方のモデルに関する再検討を今後行う予定である。また、これらの理論的な解析は DNA 配列がランダムな文字列と仮定しているが、DNA 配列は何らかの生物学的な規則性があると考えられる。この SBH を利用することにより DNA 配列の特徴づけをすることも計画をしている。

謝辞 今井の研究の一部は文部省科学研究費特定領域研究「ゲノムサイエンス」の援助を受けた。土井は日本学術振興会特別研究員として科学研究費補助金の援助を受けた。

参考文献

[1] R. J. Lipshutz, Maximum Likelihood DNA Seqencing by Hybridization, Journal of Biomoleculer Structure & Dynamics, 11, pp.637–653, 1993.

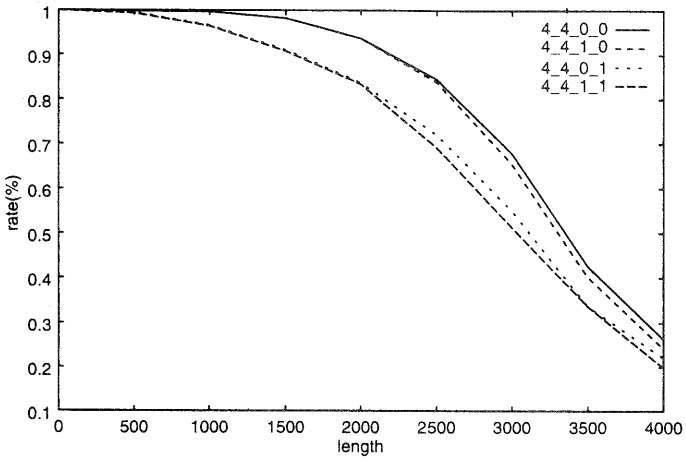

図 5: DNA 配列が決定された長さをその DNA 配列の長さで割ったグラフ ($P_p = 0.1\%, P_f = 0.1\%$ の場合)

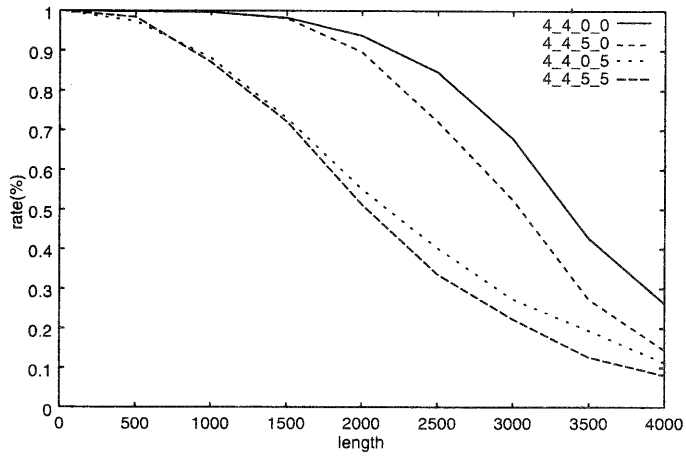

図 6: DNA 配列が決定される割合 ($P_p = 0.5\%, P_f = 0.5\%$ の場合)

[2] P. A. Pevzner, l-tuple DNA Sequencing: Computer Analysis, Journal of Biomoleculer Structure & Dynamics, 7,1,pp.63–73,1989.

[3] P. A. Pevzner and R. J. Lipshutz, Towards DNA Sequencing Chips, 19th Symposium on Mathematical Foundation of Computational Science, LNCS-841, pp.143–158, 1994.

[4] F. P. Preparata, A. M. Frieze, and E. Upfal, Sequencing-by-Hybridization at the Information-Theory Bound: An Optimal Algorithm, Proceeding of the Forth Annual International Conference on Computational Molecular Biology (RECOMB 2000), pp.245–253, 2000

[5] F. P. Preparata, A. M. Frieze, and E. Upfal, On the Power of Universal Bases in Sequencing by Hybridization, Proceeding of the Third Annual International Conference on Computational Molecular Biology (RECOMB 1999), pp.295–301, 1999

[6] M. S. Waterman, Introduction to Computational Biology, Chapman and Hall, 1995.