4E－8

ホモロジー検索プログラムＦＬＡＳＨの改良について

小出昭夫
日本アイ・ビー・エム株式会社東京基礎研究所

１．はじめに

遺伝子（ＤＮＡ）も蛋白も３次元構造を無視すれば共に文字列としてモデル化できる[1,2]。1993年の時点で主要な遺伝子データベース（DDBJ、GenBank、EMBL）はすでに１億塩基対を越え、２年ごとに倍に増えている。医療、薬品、食品、農業など広範囲の産業への応用のためにも、この大量データへの高速で高感度の検索技術が必要となっている。現在、最も基本となる検索の仕方は、ユーザの指定した参照配列に類似した配列を含む遺伝子や蛋白を取り出す類似検索（ホモロジー検索）である。類似検索の実現手法には、(1)データベース全体を毎回サーチするダイナミック・プログラミングの手法（BLAZEやSSEARCHなど）と(2)検索インデックスを事前に作成しそれを用いて検索する手法（BLASTやFASTAなど）とがある。後者は検出速度が速いが、検出感度や柔軟性の点で劣ると考えられてきた。これに対し、Califanoら[3,4]は検出速度と検出感度が共に高い手法として多重インデックスに基づく手法（FLASH）を最近提唱した。ここでは、このアルゴリズムについて簡単に述べた後、その改良として、挿入/削除における感度の向上と局所的類似検索への拡張を述べる。

２．ＦＬＡＳＨのアルゴリズム

Califanoらの手法[3,4]では、事前にデータベース上の配列の各位置に複数のインデックスを対応させ、検索実行時に、それと参照配列から生成したインデックスとの合致のヒストグラムを作成し、ヒストグラムの高さの順に遺伝子や蛋白を取り出す。

従来手法と異なり、不連続に部分列を抜き出すことにより多重インデックスを作成する。指定した長さＬのウィンドウ枠（部分列）の中からＫ個の文字の抜き出し方はC_{LK}通りだが、ｄ個の抜き出し方を決めておき、抜き出された長さＫのパックした文字列をインデックスとする。これにより、文字種がｔとすればt^K種のインデックスが生成され、配列の各位置はｄ個のインデックスを持つ。データベースに対し、事前に、参照入口t^Kの逆インデクッス・テーブルを作成しておく。

検索実行時に、同一の手法でユーザの参照配列上の各位置ｘでインデックス$\{D_i(x)\}_{i=1,d}$を作成し、テーブル参照でインデックスの合致するデータベース上の位置y_{ij}を求め、$y_{ij}-x$の位置のヒストグラムの値を１増加させる。ヒストグラムはハッシュイング技法で実現される。

検索実行時間は、長さＮの参照配列とサイズＭのデータベースに対し、

$$T=Nd\{T_1+[(Md)/(t^K)]T_2\} \qquad (1)$$

で近似的に与えることができる。すなわち、Ｋが大きい程、検索は高速になる。

偶然の一致によるヒストグラムの平均の高さは、文字の入れ替えが均等に起きるとすれば、

$$H_{noize}=Nd^2/t^K \qquad (2)$$

で与えられる。一方、長さＮの参照配列にｎ個の置換が起きている配列のヒストグラムの平均の高さは、

Some Improvements on Similarity Search Algorithm FLASH

Akio Koide
Tokyo Research Labratory, IBM Japan
1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan

$$H_n = N d\,[(N-n)/N]^K \qquad (3)$$

で近似できる。通常の類似検索では

$$(N-n)/N > 1/t \qquad (4)$$

の範囲までの置換に対してしか検索しないので、形式上、$H_{noise} < H_n$が満たされる。しかしながら、もしここでd＝1でNが小さくKが大きいと、簡単にH_nは1のオーダの値になってしまい、偶然の一致の作るゆらぎの中に埋没してしまう。これが、Califanoらの多重インデックスのアルゴリスムが高速で高感度であるとする根拠で、実際の遺伝子検索ではd＝40、K＝12あたりのパラメータが使用されている。

3．挿入／削除における感度の向上

配列と配列とが類似である定義において、ダイナミック・プログラミングのSmith-Waterman法では、置換に加え、挿入と削除を考慮している。FLASHのインデックスは不連続に抜き出された部分列をパックしたものである。従って、抜き出しの位置に削除が合致するか、抜き出しの位置が挿入によりウィンドウ枠からはみだすかしない限り、インデックスそのものは破壊されない。しかしながら、残ったインデックスの対応位置は挿入や削除によってずれる。従って、挿入や削除によるヒストグラムの劣化は、全体のカウントの減少とともにヒストグラムの山がぼやける形で起きる。

ぼやけたヒストグラムの山を見落とさないためには、前後領域のヒストグラムの比重和の順序で類似配列を取り出せばよい。FLASHでは記憶域の節約の観点からヒストグラムをハッシュイング技法で実現しているので、作成ヒストグラムに比重和をほどこすのは効率的でない。更新時に前後のヒストグラム値を比重つきで増加する。この比重は、Smith-Waterman法でそうであるように、ユーザの検索パラメータである。

4．局所的類似検索への拡張

通常の検索では式(4)が満たされていると述べたが、そうでない場合もある。全体を見れば類似度が低くとも、ある連続部分列に着目すれば非常に類似度が高いのだが、その着目すべき部分列が事前に判っていないことがある。Smith-Waterman法では、漸化式の計算の中で得点が負になるとゼロに再初期化し、途中の最高得点のみに着目するオプションにより、局所的類似検索を実現している。

FLASHのインデックスを用いて局所的類似検索を行うには、常に最後にヒストグラムが更新されたときの参照配列の位置を残し、次の更新時に、ユーザの指定する間隔以上あいたとき、ヒストグラムの値を別の領域に移し、ヒストグラムを再初期化することによって実現する。

5．おわりに

FLASHのために作成された多重インデックスをそのまま用い、検索実行時のヒストグラム作成の工夫で、挿入/削除における感度の向上と局所的類似検索への拡張が可能であることを述べた。

参考文献

1) 高木利久：ｹﾉﾑﾃﾞｰﾀﾍﾞｰｽ，情報処理，Vol. 33, No. 10, pp. 1126-1133 (1992).

2) 石川幹人，金久實：文字列を比較し並べる，日本物理学会誌，Vol. 45, No. 48, pp. 341-343 (1993).

3) A. Califano and I. Rigoutsos: "FLASH: Fast Look-Up Algorithm for String Homology,", Computer Vision and Pattern Recognition, (1993).

4) I. Rigoutsos and A. Califano: "dFLASH: Distributed Fast Look-Up Algorithm for String Homology," accepted, IEEE Computational Science and Engineering.