情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-MPS-81 No.24
Vol.2010-BIO-23 No.24
2010/12/17

最長共通部分列に基づくDNA配列の高速クラスタリング

並木洋平[†1]　石田貴士[†1]　秋山泰[†1]

近年，次世代シーケンサーの登場により一度のシーケンシングで大量のDNA配列データが得られるようになった．これに伴い読み取られた配列データの解析に要する時間が急激に増大したため，大量の配列データを高速処理できる新しい解析アルゴリズムが必要になりつつある．

本研究では次世代シーケンサーから得られたDNA配列のクラスタリングを高速化することを目的とし，配列クラスタリングにおける類似配列ペアの高速フィルタリング手法として「LCS filtering」を開発した．これを既存手法と組み合わせることで，精度を維持しつつ大規模DNA配列データのクラスタリング処理を大幅に高速化することに成功した．

Fast DNA clustering based on Longest Common Subsequence

Youhei Namiki,[†1] Takashi Ishida[†1]
and Yutaka Akiyama[†1]

Next generation sequencers enabled us to read huge number of DNA sequences at one time. With this innovation, it became to take much longer time to analyze such vast amounts of sequence data, so now it is highly demanded to develop new faster analysis algorithms to deal with those sequence data.

The objective of this study is to speed up clustering of DNA sequences produced by next generation sequencers, and we developed a new fast filtering method called “LCS filtering” to filter similar sequence pairs. By combining this with previous clustering methods, we could accomplish considerable speed up of clustering process of huge amount of DNA sequence data without losing sensitivity.

†1 東京工業大学 大学院情報理工学研究科 計算工学専攻
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

1. はじめに

近年，次世代シーケンサーの登場に伴い，DNA配列のシーケンシング技術が飛躍的に向上し，一度のシーケンシングで大量のDNA配列データが読み取られるようになった．次世代シーケンサーから得られるDNA配列は，長さは固定で比較的短く (〜150塩基)，かつ本数が非常に多い (1千万本〜) という特徴がある．DNA配列の解読量は年々増加している一方，これらのシーケンサーから得られた大量データを解析するための技術が追い付いておらず，データ解析技術のさらなる高速化が必要になりつつある．

本研究では次世代シーケンサーから得られたDNA配列のクラスタリングの高速化を目的とする．配列クラスタリングとは，大量のDNA配列やタンパク質配列のデータセットから類似配列のグループを見つけ，同じクラスタに割り当てていくデータ解析手法のことである．配列クラスタリングの目的は主に2つある．ひとつは，大量の配列データをクラスタリングして類似配列のクラスタを見つけ，データセットからクラスタの代表配列以外の類似配列を取り除くことで，これはデータセットサイズの削減に繋がる．もうひとつは，得られた各クラスタのサイズ (メンバー数) からデータセット中で特異的に出現する配列パターンを発見することである．

広く利用されている既存の配列クラスタリングツールとして，Liらによる CD-HIT[1)–3)] がある．CD-HITは近似的なクラスタリング手法を用いることによって比較的大量の配列を高速に処理することができ，メタゲノムデータのアノテーションパイプライン中で用いられたり[4)]，UniprotやPDBといった公共データベースにおいて冗長な類似配列を列挙して取り除くためにも使われている．しかし，1000万本の配列のクラスタリングは数日かかるという具合に，次世代シーケンサーから得られた配列ほどの規模のデータをクラスタリングする場合，現実的な時間で処理するのは困難になってくるという問題がある．

本研究では次世代シーケンサーから得られたDNA配列をターゲットとし，最長共通部分列 (Longest Common Subsequence, LCS) の性質を用いて類似配列ペアの高速フィルタリング手法「**LCS filtering**」を開発した．CD-HITのクラスタリングアルゴリズムの一部を改変しつつLCS filteringを導入することで，精度を維持しながらDNA配列クラスタリング処理の大幅な高速化を実現した．

2. DNA配列クラスタリング

本章ではクラスタリングの際に用いる配列ペアの類似度「sequence identity」，および本

Vol.2010-MPS-81 No.24
Vol.2010-BIO-23 No.24
2010/12/17

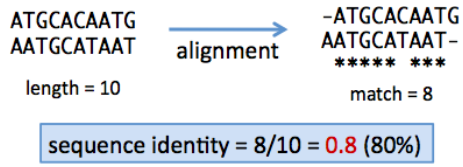

図 1 Sequence identity
Fig. 1 Sequence identity

研究で新たに提案する DNA 配列クラスタリング処理の流れについて説明する．

2.1 Sequence identity

sequence identity とは配列ペアの類似度を表す指標のひとつで，配列ペアの最適アラインメントにおいてマッチした塩基の数を配列長で割った値として定義される．アラインメントスコアなどの類似度の指標と比べて直感的に分かりやすく，またマッチ数を配列長で割って正規化するため配列長に依存しないという特徴がある．1 塩基もマッチしない配列同士の sequence identity は 0 となり，完全に同一の配列同士の sequence identity は 1 となる．

配列ペア間の sequence identity の計算例を図 1 に示す．この例では長さ 10 塩基の 2 つの DNA 配列 ATGCACAATG と AATGCATAAT の間で sequence identity を求めている．この配列ペアの最適アラインメントのマッチ数は 8 であるため，sequence identity は $8/10 = 0.8(80\%)$ となる．

本研究のクラスタリング手法では，あらかじめ同クラスタにまとめる sequence identity の閾値を決めておき，sequence identity が閾値を超える類似配列ペアを同クラスタにまとめていくという手法をとる．これは CD-HIT のクラスタリング基準と同じである．

2.2 本研究の DNA 配列クラスタリング処理の流れ

本研究で提案する DNA 配列のクラスタリング処理の流れを以下に示す．このクラスタリング手法は主に CD-HIT のアルゴリズムを基にしている．

(1) クラスタリングする配列の集合を Q とし，既存クラスタの代表配列の集合を $R = \emptyset$ とする．

(2) 各 $q \in Q$ について，

(a) short word table を用いて q と R を比較し，q と共通の k-mer (k 塩基の部分配列) を t 個以上持つ既存クラスタの代表配列の集合 $R_{q,k}(\subseteq R)$ を求める **(short word filtering)**

(b) q と各 $r \in R_{q,k}$ について，

(i) q，r 間の最長共通部分列の長さ $LLCS(q,r)$ を求める．$LLCS(q,r)$ が一定値以上の場合は，q と r を同じクラスタに割り当てられる類似配列ペアの候補として残し，それ以外の場合は別クラスタとみなす．**(LCS filtering)**

(ii) 残った配列ペア q，r について，アフィンギャップスコアを用いて最適アラインメントを求め，その結果から q，r 間の sequence identity を求める．sequence identity が閾値以上の場合は q と r を同クラスタとみなし，q を r のクラスタに追加する (代表配列は r のままで更新しない)．それ以外の場合は別クラスタとみなす．

(c) q が既存のどのクラスタにも属さない場合は，新たに q を代表配列とするクラスタを作る．また，q を short word table に登録し，$R \leftarrow R \cup q$ とする．

ある配列ペアが同クラスタに入るかどうかを判定するには最適アラインメントと sequence identity を求める必要がある．しかし，最適アラインメントの時間計算量は，動的計画法を用いた場合 $O(mn)$ (m，n は 2 配列の配列長) と非常に大きいため，全配列ペアに対してアラインメント処理を行うのは非現実的である．また，配列クラスタリングでは sequence identity が閾値を超えるような類似配列ペアの数に対し，sequence identity が閾値を超えない非類似配列ペアの数の方が圧倒的に多い．そこで，本手法ではより計算量の小さい手法であらかじめ類似配列ペアをフィルタリングして残し，非類似配列ペアは捨てて探索を途中で打ち切ることで，アラインメント処理の回数を劇的に減らし，クラスタリング処理全体の計算時間を抑えている．フィルタリング処理は short word filtering と LCS filtering の 2 段階があり，それぞれ次章以降で説明する．

新たにクラスタを作るとき，最初にそのクラスタに割り当てられた配列がクラスタの代表点となり，その後クラスタに配列が追加されても代表配列は更新しない．そのため，クラスタリング結果は入力配列の順番に依存するが，結果としてクラスタリング処理にかかる時間を短くすることができる．

本研究で提案するクラスタリング手法は CD-HIT のアルゴリズムを基にしているため，両者のクラスタリングアルゴリズムの間には共通点が多い．両者の相違点は主に以下の 2 つである．

(1) short word filtering のフィルタリング基準が異なる

(2) 本手法は short word filtering の直後に新たなフィルタリング処理 (LCS filtering) を導入している

Vol.2010-MPS-81 No.24
Vol.2010-BIO-23 No.24
2010/12/17

3. Short word filtering

short word filtering は，既存のクラスタ代表配列群と新たにクラスタリングするクエリ配列の間で共通の k-mer (k 塩基の部分配列) を持つペアを列挙することで，共通の k-mer を持つ類似配列ペアを高速にフィルタリングする手法である．この手法は，sequence identity が閾値を超えるほど類似している配列ペアは同時にある程度の長さの部分配列を共有しているだろうという考えに基づいている．k の値はクエリ配列長や sequence identity の閾値に合わせて調節する必要がある．short word filtering は CD-HIT でも用いられているが，本手法では後段にさらに LCS filtering を導入することに伴い，CD-HIT の short word filtering とは異なるフィルタリング基準を用いている．

short word filtering では，k-mer とクラスタ代表配列を関連付けた short word table を用いることで，特定の k-mer を持つクラスタ代表配列を高速に列挙する．

3.1 Short word table への登録

新たにクラスタができたとき，その代表配列を short word table に登録する必要がある．まず，新しいクラスタの代表配列がもつ k-mer を k 文字ずらしで列挙する．次に，これらを short word table に登録し，k-mer の塩基配列とそれを部分配列に持つクラスタの代表配列を関連付ける．

short word table には hash table のデータ構造を用いる．このようにすることで，指定した k-mer を持つクラスタ代表配列を高速に列挙することが可能になる．塩基配列の場合は 4^k 通りの k-mer が存在するため，4^k 個のエントリリストを持つ hash table を用意する必要がある．

3.2 Short word table の検索

既存クラスタの代表配列と比較するクエリ配列側を 1 文字ずらしで k-mer を列挙し，short word table を使って共通 k-mer を持つクラスタの代表配列を列挙する．そして，共通 k-mer が t 個以上のクラスタ代表配列があった場合，同じクラスタに割り当てられる候補の類似配列ペアとしてフィルタを通過させ，共通 k-mer が一定個数に満たない非類似配列ペアは捨てる．共通 k-mer の個数の閾値 t は配列長や k の値，sequence identity の閾値に応じて設定するが，本手法ではクラスタ代表配列の k-mer を粗く k 文字ずらしで登録しているため，類似配列ペアの列挙漏れを防ぐために小さめに設定しておく．

3.3 本手法の Short word filtering の特徴

本手法の short word filtering では，上述の通りクラスタの代表配列は k 文字ずらしで，クエリ配列側は 1 文字ずらしで k-mer を列挙する．一方，CD-HIT の short word filtering では，クラスタの代表配列とクエリ配列のどちらも 1 文字ずらしで k-mer を列挙する．

本手法のようにクラスタの代表配列の k-mer を k 文字ずらしで short word table に登録すると，CD-HIT の 1 文字ずらしの場合と比較して以下の 2 点のメリットが挙げられる．

(1) short word table のメモリ使用量を約 $1/k$ に抑えることができる．
(2) short word table のエントリ数が減るため，探索にかかる時間を抑えることができる．

これに対して，本手法の k 文字ずらしの場合におけるデメリットは以下の 2 点が挙げられる．

(1) フィルタを通過するのに必要な共通 k-mer の個数の閾値を下げるため，偶然的な配列の部分一致により非類似配列ペアがフィルタを通過する可能性が増え，後のアラインメント処理回数が増加する可能性がある．
(2) クラスタ代表配列とクエリ配列の間でほぼ等間隔にミスマッチやギャップが入った場合は類似配列ペアでも共通 k-mer が見つからない場合がある．

このうち，1 つ目のデメリットについては，short word filtering の直後かつアラインメント処理の前にもうひとつ高速で高精度な類似配列ペアのフィルタリング処理を入れることで解決できる．本研究で提案する最長共通部分列を用いた手法 (**LCS filtering**) がそれに当たる．LCS filtering の詳細は 4 章で説明する．

また，2 つ目のデメリットについては，そもそも DNA 配列は単純なランダム配列ではなく，等間隔でミスマッチやギャップが入り，かつ sequence identity が閾値を超える類似配列ペアが存在することは非常に稀であると考えられる．そのため，k 文字ずらしで short word table に登録する手法がクラスタリングの精度に大きな悪影響を与えることはないと考えられる．

4. 最長共通部分列の長さに基づく類似配列ペアのフィルタリング

本章では本研究の提案手法 **LCS filtering** について説明する．この手法は最長共通部分列の長さと sequence identity の関係を利用する．また，最長共通部分列の長さはビット演算を用いることで高速計算する．

short word filtering による類似配列ペアのフィルタリングは，少ない計算量で非常に大量の配列ペアに対して処理を行うことができ，次世代シーケンサーのように大規模な配列数のデータから類似配列ペアを見つける上で有用である．しかし，short word filtering によるフィルタリングは粗く，フィルタを通過して次の処理に進む配列ペアの中には sequence

Vol.2010-MPS-81 No.24
Vol.2010-BIO-23 No.24
2010/12/17

identity が閾値を超えない非類似配列ペアも多く含まれている．これらをそのままアラインメントによる sequence identity の計算処理に流した場合，アラインメント処理回数が非常に多くなり，クラスタリング処理に長時間を要する恐れがある．そのため，short word filtering によるフィルタリングの後に高速かつ高精度なフィルタリング処理を導入することはクラスタリング処理を高速化する上で非常に有効である．

4.1 最長共通部分列

ある配列から順序を保存しながら要素を取り出して作った配列を部分列といい，2 つの配列から得られる共通の部分列を共通部分列という．最長共通部分列 (Longest Common Subsequence, LCS) とは，この共通部分列のうち長さが最長のもののことである[5]．例えば，2 つの配列 ATCAGTC と CTAGAC の間の最長共通部分列は TAGC となる．

配列ペア間の最長共通部分列を求めることは，開始ギャップに大きなペナルティを与えずに 2 配列のマッチ数が最大になるようアラインメントした場合のマッチした塩基だけで構成される配列を求めることと同義である．

4.2 最長共通部分列の長さ

2 つの配列 $X = (x_1, x_2, ..., x_m)$，$Y = (y_1, y_2, ..., y_n)$ の最長共通部分列の長さは以下のようにして求められる．

X，Y の先頭からそれぞれ i 個，j 個までの要素の部分配列を $X_i = (x_1, x_2, ..., x_i)$ $(0 \leq i \leq m)$，$Y_j = (y_1, y_2, ..., y_j)$ $(0 \leq j \leq n)$ とし，配列 X と Y の最長共通部分列の長さを $LLCS(X, Y)$ (the Length of LCS) とする．このとき，以下の漸化式が成り立つ．

$$LLCS(X_i, Y_j) = \begin{cases} 0 & (\text{if } i = 0 \text{ or } j = 0) \\ LLCS(X_{i-1}, Y_{j-1}) + 1 & (\text{if } x_i = y_j) \\ \max(LLCS(X_i, Y_{j-1}), LLCS(X_{i-1}, Y_j)) & (\text{if } x_i \neq y_j) \end{cases}$$

上式から $LLCS(X, Y)$ を求めれば X，Y 間の最長共通部分列の長さを求めることができる．一般に $LLCS(X, Y)$ は動的計画法を用いて求めることができ，この場合の時間計算量は $O(mn)$ となる．

4.3 LCS filtering による類似配列ペアのフィルタリング

本研究で提案する **LCS filtering** は最長共通部分列の長さを以下のように用いて類似配列ペアを検索する．

2 配列間の最長共通部分列の長さ $LLCS(X, Y)$ は，言い換えれば 2 つの配列をマッチ数が最大になるようにアラインメントしたときのマッチ数である．また，2 配列間の sequence identity はアラインメントスコアが最大になるように 2 つの配列をアラインメントしたときのマッチ数を配列長 n で割った値である．このことから，最長共通部分列の長さと sequence identity の間には以下の式が成立する．

$$\frac{LLCS(X, Y)}{n} \geq \text{sequence identity}$$

つまり，2 配列間の最長共通部分列の長さ $LLCS(X, Y)$ を配列長 n で割った値は，2 配列間の sequence identity が取り得る値の上界となる．この性質を利用すれば，アラインメント処理を行い sequence identity を求める前に，$LLCS(X, Y)/n$ の値を計算し，sequence identity の閾値以上ならば同じクラスタに割り当てられる可能性のある類似配列ペアとしてフィルタを通過させ，閾値未満ならば捨てて処理を打ち切ることで，非類似配列ペアに対するアラインメント処理を省くことができる．以上が **LCS filtering** の概要である．

LCS filtering には以下の利点がある．

(1) フィルタリングの基準に sequence identity の閾値をそのまま利用することができ，新たにヒューリスティックな閾値を設ける必要がない．

(2) 定義上 sequence identity が閾値を超える類似配列ペアは $LLCS(X, Y)/n$ の値も必ず sequence identity の閾値以上の値を取るため，LCS filtering により誤って類似配列ペアが捨てられることはない．

(3) 最長共通部分列の長さを用いるため，共通 k-mer の有無による short word filtering と比べて 2 配列間の共通塩基の位置関係やギャップを厳密に加味することができる．sequence identity の閾値が高いときは，2 配列のほぼ全体が類似していなければ $LLCS(X, Y)/n$ の値は閾値を超えない．

以上より，LCS filtering は short word filtering の後かつアラインメント処理の前に行うフィルタリング処理として特に適していると考えられる．

4.4 ビット演算による最長共通部分列の長さの高速計算

以上のように，最長共通部分列の長さ $LLCS(X, Y)$ を用いることにより，short word filtering よりも高い精度で類似配列ペアの検索を行うことができる．しかし，前述の通り 2 配列 X，Y 間の $LLCS(X, Y)$ を単純に動的計画法で計算すると $O(mn)$ の時間計算量がかかり，フィルタリング処理として用いるには計算量が大きすぎるため時間がかかってしま

Vol.2010-MPS-81 No.24
Vol.2010-BIO-23 No.24
2010/12/17

う．しかし，$LLCS(X,Y)$ の計算についてはビット演算で高速計算する手法が提案されており，これらを用いることで $O(n)$ に近い時間計算量で $LLCS(X,Y)$ を求めることが可能になる[6)–8)]．そこで，本手法ではそれらの中で最も高速な Hyyrö らによるビット演算の手法[8)] を用いることで，$LLCS(X,Y)$ を高速に計算する．

以下において，ビット b が i 桁並んでいる二進数を b^i と表記する．この表記法を用いると，例えば 1111 は 1^4，0011100 は $0^2 1^3 0^2$ と表される．また，配列 X，Y 中に出現する文字の集合を Σ とし，その種類の数を $\sigma = |\Sigma|$ とする．

まず，$ComputePM(X)$ を図 2 のように定義する．$ComputePM(X)$ は，配列 X の各塩基についてその塩基の出現位置に応じた PM(Position Matrix) のビットを立てる処理を行う．ここで，| はビット単位の OR 演算を表す．

この $ComputePM(X)$ を用いることで，$LLCS(X,Y)$ は図 3 のようにして計算することができる．ここで，&はビット単位の AND 演算を表す．また，+，− は一般的な整数の加算，減算であり，演算の際には隣接ビット間で繰り上がりや繰り下がりが生じる．

$ComputePM(X)$ の時間計算量は $O(\sigma\lceil m/w\rceil + m)$，$LLCS(X,Y)$ の時間計算量は $O(\lceil m/w\rceil n)$ である．これより，$ComputePM(X)$ と $LLCS(X,Y)$ の計算にかかる時間は動的計画法による $LLCS(X,Y)$ の計算にかかる時間 $O(mn)$ よりも短く，同様にアラインメント処理にかかる時間 $O(mn)$ と比べても高速であることが分かる．

5. 評価実験

本研究で提案する手法を用いたクラスタリングツールを C++で実装し，CD-HIT との比較実験を行った．

5.1 実験で使用したデータなど

評価実験の入力データセットは，MetaSim[9)] を用いて生成したメタゲノムの配列データを用いた．入力データセットの DNA 配列の配列長は，現在主に使われている次世代シーケンサーの出力配列長に合わせて 100b，150b，400b の 3 通り (固定長)，配列数は 100 万本，200 万本，500 万本の 3 通りで，計 9 通りの入力データセットを用いた．また，クラスタリングの際の閾値やパラメータについては，sequence identity の閾値は 0.9 (90%)，short word filtering の k の値は 9 とし，共通 k-mer の個数の閾値 t については，100b と 150b の場合は 1，400b の場合は 4 とした．計算機環境は SUSE Linux 10 のワークステーションで，AMD Opteron 2.8GHz のシングルコアを用いた．

$ComputePM(X)$

1: **for** $\lambda \in \Sigma$ **do**
2: $\quad PM_\lambda \leftarrow 0^m$
3: **end for**
4: **for** $i \in 1...m$ **do**
5: $\quad PM_{X_i} \leftarrow PM_{X_i} \mid 0^{m-i}10^{i-1}$
6: **end for**

図 2 $ComputePM(X)$ の計算
Fig. 2 Computation of $ComputePM(X)$

$LLCS(X,Y)$

1: $ComputePM(X)$
2: $V' \leftarrow 1^m$
3: **for** $j \in 1...n$ **do**
4: $\quad U \leftarrow V'$ & PM_{Y_j}
5: $\quad V' \leftarrow (V'+U) \mid (V'-U)$
6: **end for**
7: Return the number of unset bits in V'

図 3 ビット演算による $LLCS(X,Y)$ の計算
Fig. 3 Computation of $LLCS(X,Y)$ by bit operations

5.2 結　　果

CD-HIT と提案手法のクラスタリングの所要時間をそれぞれ表 1〜3，および図 4〜6 に示す．また，表 1〜3 の提案手法の計算時間の横には CD-HIT に対する提案手法の速度向上率を括弧書きで示している．

これらの表と図から分かる通り，100b〜400b，100 万本〜500 万本の全ての場合において CD-HIT よりも提案手法の方が速いという結果が得られた．固定長の DNA 配列 500 万本について，配列長 100 塩基の場合は CD-HIT に対して約 5.2 倍，150 塩基の場合は約 3.7 倍，400 塩基の場合は約 3.2 倍の高速化を実現した．

Vol.2010-MPS-81 No.24
Vol.2010-BIO-23 No.24
2010/12/17

表 1 各手法の計算時間 (100 万本)
Table 1 Computation time (1 million sequences)

	100b		150b		400b	
CD-HIT	41m40s		45m15s		1h17m01s	
提案手法	7m10s	(5.81)	13m45s	(3.29)	30m18s	(2.54)

表 2 各手法の計算時間 (200 万本)
Table 2 Computation time (2 million sequences)

	100b		150b		400b	
CD-HIT	2h11m47s		2h17m56s		3h11m17s	
提案手法	18m42s	(7.05)	31m41s	(4.35)	1h06m55s	(2.86)

表 3 各手法の計算時間 (500 万本)
Table 3 Computation time (5 million sequences)

	100b		150b		400b	
CD-HIT	11h17m22s		11h28m17s		15h28m50s	
提案手法	2h11m09s	(5.16)	3h04m43s	(3.73)	4h53m22s	(3.17)

配列長が短い時の CD-HIT に対する提案手法の速度向上率が大きい一方で，配列長が長くなるにつれて速度向上率が低下していく傾向が見られる．この問題は，配列長が長くなることによりアラインメント処理の計算量が大きくなっていくことが原因のひとつと考えられる．提案手法を実装した際に用いたアラインメント処理は，配列長と sequence identity の閾値から考えられる最大許容ギャップ数を含む範囲を計算するようになっている．一方，CD-HIT のアラインメント処理は実装上さらに計算量を抑える工夫がなされている．そのため，CD-HIT と同様に提案手法でも LCS filtering 後のアラインメント処理に工夫を加えればこの問題は解決できる可能性がある．

クラスタリングの際のメモリ使用量については，提案手法は CD-HIT の約 1.6 倍以内のメモリ使用量に収まっており，400b の配列 500 万本で 5GB 程度であることが分かっている．3.3 で述べた通り，提案手法ではクラスタ代表配列の k-mer を k 文字ずらしで short word table に登録しているため，提案手法の short word table のメモリ使用量は CD-HIT と比べて非常に小さくなっている．一方，CD-HIT は short word table で使うメモリ使用量の上限を指定することができ (デフォルトで 800MB)，その使用量の範囲でクラスタリング処理を行うように実装の工夫がなされているため，配列数やクラスタ数が増えてもメモリ使用量の増加をある程度抑えられるようになっている．そのため，提案手法で short word

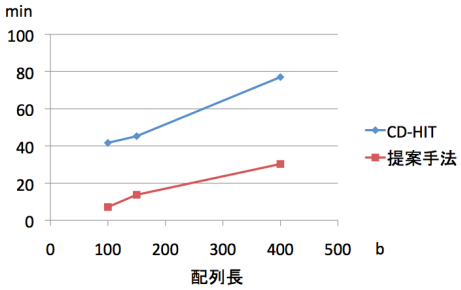

図 4 各手法の計算時間 (100 万本)
Fig. 4 Computation time (1 million sequences)

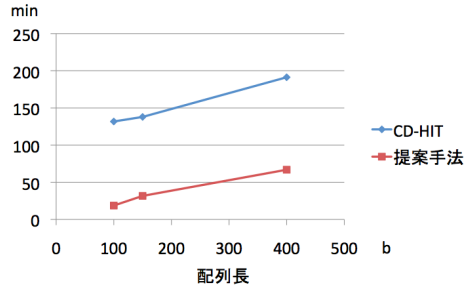

図 5 各手法の計算時間 (200 万本)
Fig. 5 Computation time (2 million sequences)

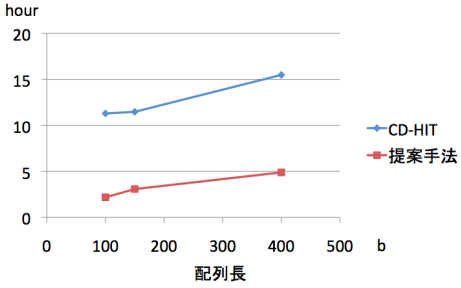

図 6 各手法の計算時間 (500 万本)
Fig. 6 Computation time (5 million sequences)

table のメモリ使用量が削減されたにもかかわらず，CD-HIT の方がメモリ使用量が少ないという結果になったものと考えられる．この問題については，提案手法の実装ではメモリの使用方法についてはまだ改善の余地があるため，メモリ使用量の点で提案手法が CD-HIT に大きく劣っている可能性は低いと考えられる．

実験結果として得られたクラスタデータについては, CD-HIT と提案手法の間に大きな差異は見られなかったが，同じクラスタに入るべき類似配列ペアが CD-HIT ではクラスタリングされず提案手法では正しくクラスタリングされるという例が，その逆の例と比べて多く見つかっている．

6. 結　　論

本研究では，次世代シーケンサーから得られた DNA 配列のクラスタリングを高速化す

Vol.2010-MPS-81 No.24
Vol.2010-BIO-23 No.24
2010/12/17

るための手法として，最長共通部分列に基づく類似配列ペアのフィルタリング手法 **LCS filtering** を開発した．LCS filtering を short word filtering と組み合わせることで，short word filtering では捨てることができなかった非類似配列ペアを LCS filtering が高精度かつ高速に判別して排除するため，クラスタリング処理全体の高速化を図ることができる．また，short word filtering の際のメモリ使用量や処理時間を短くするために short word filtering の基準を若干甘くした場合でも，LCS filtering は第二のフィルタリング処理として非常に有効である．評価実験では，固定長の DNA 配列 500 万本について，配列長 100 塩基の場合は CD-HIT に対して約 5.2 倍，150 塩基の場合は約 3.7 倍，400 塩基の場合は約 3.2 倍の高速化を実現した．本手法を用いることで，今までのクラスタリングツールでは難しかった大規模な DNA 配列データのクラスタリングが現実的な時間内で可能になるだろう．

今後の課題として考えられる点のひとつに，本クラスタリング手法を分散並列処理に対応させることが挙げられる．本研究の評価実験ではクラスタリング処理にシングルコアを用いたが，近年では計算機環境のマルチコア化やクラスタ化が急速に進んでいる．そのため，本クラスタリング手法も分散並列処理できるように改変し，それらの最新の計算機資源を有効活用できるようにすることが重要であると考えられる．

参 考 文 献

1) Li, W., Jaroszewski, L. and Godzik, A.: Clustering of highly homologous sequences to reduce the size of large protein databases, *Bioinformatics*, Vol.17, pp.282–283 (2001).
2) Li, W., Jaroszewski, L. and Godzik, A.: Tolerating some redundancy significantly speeds up clustering of large protein databases, *Bioinformatics*, Vol.18, pp.77–82 (2002).
3) Li, W. and Godzik, A.: Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences, *Bioinformatics*, Vol.22, No.13, pp.1658–1659 (2006).
4) Li, W.: Analysis and comparison of very large metagenomes with fast clustering and functional annotation, *BMC Bioinformatics*, Vol.10, p.359 (2009).
5) Leiserson, C.E., Rivest, R.L., Stein, C. and Cormen, T.H.: アルゴリズムイントロダクション　第 2 巻　アルゴリズムの設計と解析手法　改訂第 2 版，近代科学社 (2007).
6) Allison, L. and Dix, T.: A bit-string longest-common-subsequence algorithm, *Information Processing Letters*, Vol.23, pp.305–310 (1986).
7) Crochemore, M., Iliopoulos, C., Pinzon, Y. and Reid, J.: A fast and practical bit-vector algorithm for the longest common subsequence problem, *Information Processing Letters*, Vol.80, pp.279–285 (2001).
8) Hyyrö, H.: Bit-parallel LCS-length computation revisited, *Proc. 15th Australasian Workshop on Combinatorial Algorithms (AWOCA)*, pp.16–27 (2004).
9) Richter, D.C., Ott, F., Auch, A.F., Schmid, R. and Huson, D.H.: MetaSim: a sequencing simulator for genomics and metagenomics, *PLoS ONE*, Vol.3, No.10, p. e3373 (2008).