Vol.2020-BIO-61 No.10
2020/3/13

圧縮アミノ酸を利用した二段階のシード探索による メタゲノム配列相同性検索の改良

高畠 和輝[1]　伊澤 和輝[1]　秋川 元宏[1]　大上 雅史[1]　秋山 泰[1,a)]

概要：土壌や海洋，生体内などの環境に生息する微生物を網羅的に解析するメタゲノム解析の手法の一つとして，大量のシークエンスデータに高精度な配列相同性検索を行うものがある．従来法である BLAST などの配列相同性検索ツールでは，最新の次世代 DNA シーケンサーのスループットに対して計算速度が不十分であり，前述のような解析のボトルネックとなっている．本研究では，データベース配列とクエリ配列の間で類似度の高い部分配列を二段階で探索し，各段階において文字数の異なる圧縮アミノ酸集合を適用することで，高精度かつ高速な相同性検索を行う新たなアルゴリズムの提案・実装を行った．また評価実験により，従来手法と比較した際の提案手法の有効性を確認した．

キーワード：メタゲノム解析，相同性検索，圧縮アミノ酸

Improvement of homology search for metagenomic analysis by two-step seed search with reduced amino acid alphabet

Kazuki Takabatake[1]　Kazuki Izawa[1]　Motohiro Akikawa[1]　Masahito Ohue[1]
Yutaka Akiyama[1,a)]

Abstract: Metagenomic analysis is a technique for comprehensively analyzing microorganisms existing in environments such as soil, the ocean, and living organisms. In metagenomic analysis, it is necessary to perform high-precision homology searches on large amounts of sequence data. In recent years, the amount of analysis data has increased rapidly with the development of next-generation sequencing (NGS). NCBI BLAST has been the most widely-used for this problem, but it's calculation speed is insufficient for the throughput of current DNA sequencers. In this study, we propose a new, high-performance homology search tool by using reduced amino acid alphabet to search for high similar subsequences in two-steps. Additionally compared with several previous tools, we evaluated the validity of the proposed method.

Keywords: metagenomic analysis, homology search, reduced amino acid alphabet

1. 背景

土壌や海洋，生体内などに存在する微生物から DNA を抽出・シークエンスすることで，環境に存在する微生物のゲノムを網羅的に解析することをメタゲノム解析と呼ぶ．メタゲノム解析では環境中に含まれる微生物の種類やその存在比率を推定可能であるほか，培養困難な未知の微生物のゲノム解析が可能になる．

環境に含まれる微生物のすべてのゲノム配列が既知であることは稀であるため，メタゲノム解析では同一種のみではなく，近縁の生物のゲノム情報を参照する必要がある．このためメタゲノム解析では，既知の配列データベースに対して，DNA シーケンスで得られた配列と類似した配列を検索する配列相同性検索を行う．またこの際，遺伝子領域のみに着目する場合は，DNA 配列を 6 フレームのタンパク質配列に翻訳してから検索を行う．DNA 配列が通常

[1] 東京工業大学 情報理工学院 情報工学系
Department of Computer Science, School of Computing, Tokyo Institute of Technology
[a)] akiyama@c.titech.ac.jp

A，T，G，C の 4 文字で表現されているのに対して，タンパク質配列は標準的には 20 文字で表現されている．さらに，DNA 配列間の比較では各塩基について一致か不一致の 2 状態でしか区別しないことが多いが，タンパク質配列間の比較では，各アミノ酸ごとに他のアミノ酸への置換の起こりやすさを表した行列に基づいて残基同士の比較が行われる [1]．このため，タンパク質配列で配列相同性検索を行うことは，DNA 配列の場合と比べてより複雑で困難なものとなっている．

配列相同性検索においては，動的計画法を用いて厳密に最適なアラインメントを計算可能である Smith-Waterman アルゴリズム [2] を実装した SSEARCH[3] 等を利用することが望ましい．しかし SSEARCH は低速であるという問題があり，データ量が多い場合現実的な時間で計算が終わらない．このため，現在では高速な配列相同性検索を行う BLAST[4], [5] などのプログラムが一般的に広く利用されている．

一方，次世代 DNA シークエンサと呼ばれるハイスループットな DNA シーケンサが登場したことにより，解析で扱うデータ量が急激に増加している．illumina 社の NovaSeq 6000 DNA シーケンサを例にとると，数十時間程度の 1 ランで出力は最大 6T 塩基にも達し，BLAST による解析には数十万 CPU 日もの時間が必要になると推定される．

BLAST 以降も RAPSearch2[6] や GHOSTZ[7], DIAMOND[8] などより高速な配列相同性検索プログラムが提案されているが，検索の速度と精度はトレードオフの関係にあり，一般的に相同性があると判断される E-value 10^{-5} 以下の範囲で BLAST と同等の精度を持ちつつ十分に高速なプログラムは存在しない．そのため，高精度かつ高速な配列相同性検索手法の必要性が高まっている．

RAPSearch2，GHOSTZ，DIAMOND では標準的なアミノ酸 20 種類をより少ないアミノ酸で表現する圧縮アミノ酸を利用することで，高効率かつ高精度な検索を実現している．本研究では，これに加えて，文字数の異なる圧縮アミノ酸集合を用いて段階的に類似度の高い部分文字列を探索する新たなシード探索アルゴリズムを提案し，より高速な配列相同性検索プログラムを実装することを目的とする．

2. 提案手法

一般に配列相同性検索では，データベースとクエリ配列間で類似度の高い部分文字列（シード）を探索することで，アラインメント計算を行う対象となるデータベース位置の候補を大幅に削減する．本研究では，圧縮アミノ酸を用いた二段階のシード探索による高精度かつ高速な配列相同性検索手法を提案する．

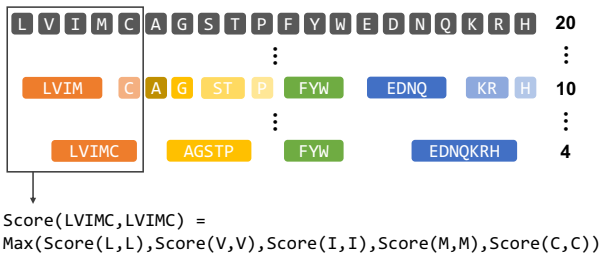

図 1　圧縮アミノ酸
Fig. 1　Reduced amino acid alphabet

2.1 圧縮アミノ酸

圧縮アミノ酸は**図 1** のように標準的なアミノ酸 20 種類をより少ないアミノ酸で表現する技術である．同じ圧縮アミノ酸グループに属するアミノ酸の一致を完全一致として扱うことにより，曖昧な一致を許容した高速なタンパク質配列の比較が可能になる．本研究においては，Murphy LR, *et al.* (2000) [9] によって考案されたものを用いた．この圧縮アミノ酸は 20 種類の各アミノ酸間の類似度が記述された置換行列 BLOSUM62 により導出されたものである．圧縮アミノ酸グループ内の一致スコアには，そのグループ内の各アミノ酸が完全一致した場合のスコアの最大値を用いる（図 1）．

2.2 二段階のシード探索

提案手法では異なる圧縮アミノ酸を用いて段階的にシード探索を行う．提案手法におけるシード探索の全体像を**図 2** に示す．まず探索の一段階目では，データベースとクエリ配列をそれぞれ圧縮アミノ酸 A_1 種に変換する．次に，データベースとクエリ配列の間で圧縮アミノ酸がハミング距離 H_1 以下で，残基長 L_1 に渡り一致する位置を探索する．この一段階目で探索された部分文字列を seed1 とする．

探索の二段階目では，一段階目で得られた seed1 の両端について，それぞれ残基長 L_2 を伸ばした範囲を seed2 とする．次に，seed2 の範囲を圧縮アミノ酸 A_2 種に変換し，ハミング距離 H_2 以下で一致するかどうか計算する．seed2 のいずれも一致する場合は，該当部分がアラインメント計算の候補となる．

A_1，A_2，L_1，L_2 の値が増加するにつれて探索されるシードの数が減り，以降のアラインメント計算回数が削減されるため処理は高速になる．しかし，部分文字列全長で完全一致する位置だけを探索すると，検索の精度は低下してしまう．そこで，許容されるハミング距離 H_1，H_2 を定義し，一部にミスマッチが含まれる部分文字列をシードに加えることで，相同性検索の精度向上を図っている．

2.3 提案手法の概要

本節では本研究で提案する手法の概要を説明する．**図 3**

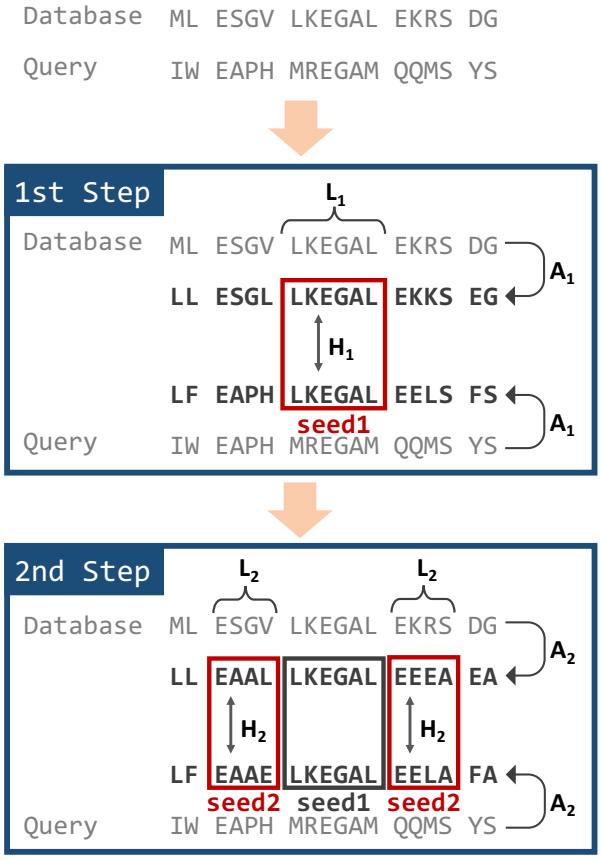

図 2　提案手法におけるシード探索
Fig. 2　Seed search of proposed method

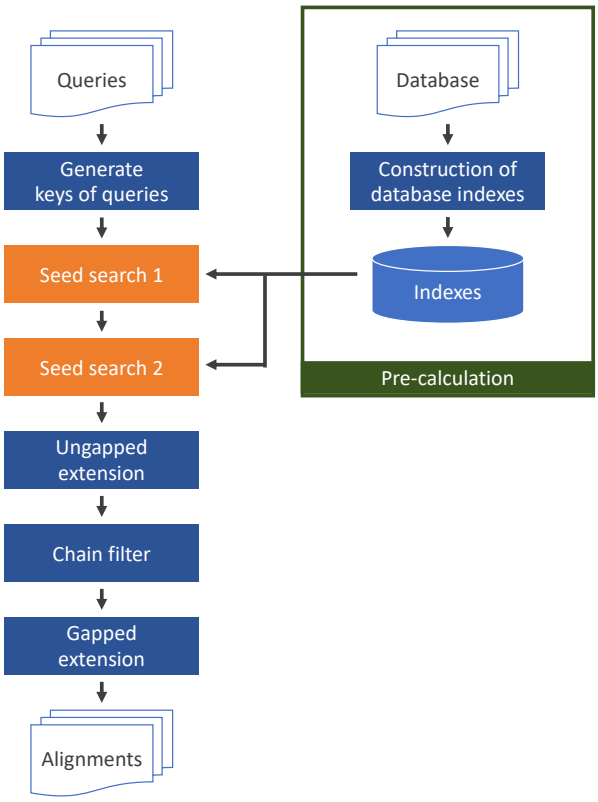

図 3　提案手法の実行の流れ
Fig. 3　Processing flow of proposed method

に提案手法の流れを示した．
まず，予め対象となるデータベースから，圧縮アミノ酸に変換した部分文字列のインデックスを構築する．検索時には DNA シーケンサによって読み取られた DNA クエリ配列をタンパク質配列に翻訳し，データベースと同様の圧縮アミノ酸に変換したのちに部分文字列キーを生成する．次に，データベースが持つ部分文字列インデックスに対してクエリ配列が持つ部分文字列キーを検索し，データベースとクエリ配列の間で類似度が高い位置（シード）を探索する．探索で見つかったシードを中心にアラインメントの伸長を行い，スコアを計算することで相同性検索を行う．
各処理の詳細を以降に述べる．

2.3.1　データベースの部分文字列インデックスの構築

まず，データベース中のすべてのタンパク質配列を ‘#’ を配列の区切り文字として連結する．この連結配列中を，2.2 節において述べたシードの全長を window として 1 文字ずつずらしながら，seed1 及び seed2 に対応する部分文字列をパラメータ A_1，A_2，L_1，L_2 を用いて生成する．この際，生成した部分文字列中に区切り文字が含まれる場合は，その部分文字列を破棄する．この操作を連結配列の終端まで繰り返すことで，データベース中に含まれる部分文字列とその位置を対応付けるインデックスを構築する．

2.3.2　クエリ配列の部分文字列キーの生成とシード探索

DNA シーケンサーによって出力された DNA 断片配列を 6 フレームに翻訳し，得られた 6 つのタンパク質配列を区切り文字#を用いて連結する．この連結配列中を 2.3.1 項と同様の window 幅で 1 文字づつずらしながら，区切り文字を含まない部分文字列のキーを生成する．
シード探索では，2.2 節で述べたパラメータ H_1，H_2 を用いて，クエリ配列の部分文字列キーと一致するデータベース位置を探索する．一致する位置が見つかった場合はアラインメントの候補として記録し，次の伸長に移る．

2.3.3　Ungapped Extension と Chain Filtering

伸長では最初に，シード探索でアラインメントの候補となった位置を中心として ungapped extension を行う．ungapped extension では BLAST と同様に，スコアが低下し始めたら伸長を停止する X-dropoff[4] を利用する．このときのスコアが閾値を超えた候補のみ，次の gapped extension の候補として記録しておく．

ungapped extension によって出力されたシード同士は近傍に位置する場合がある．特にクエリ配列と一致率の高い配列がデータベースに存在する場合，多くのシードがその領域に存在し，それらは gapped extension でほぼ同じアラインメント結果となる．このため，提案手法では近傍に位置するシード同士を 1 つにまとめることで gapped extension の実行回数を削減する chain filtering を用いた [5]．本研究

表 1　比較対象プログラムと検索実行オプション
Table 1　compared program and runtime options

プログラム	オプション
BLAST	-outfmt 6 -comp_based_stats 0
	-num_alignments 10 -seg no
RAPSearch2 fast	-v 10 -b 0 -t n -a t
RAPSearch2	-v 10 -b 0 -t n
GHOSTX	-q d -F F
GHOSTZ	-q d -F F
DIAMOND	-f 6 -k 10 -e 10 -p 1
	–masking 0 –comp-based-stats 0
DIAMOND	-f 6 -k 10 -e 10 -p 1 –sensitive
sensitive	–masking 0 –comp-based-stats 0
DIAMOND	-f 6 -k 10 -e 10 -p 1 –more-sensitive
more sensitive	–masking 0 –comp-based-stats 0

ではシードが重なっている場合，または 2 つのシード間で ungapped extension を行い，スコアが閾値以下である場合にシードを 1 つにまとめる．

2.3.4　Gapped Extension

提案手法では gapped extension は BLAST と同じ方法を用いた．ungapped extension と同様に，スコアが低下し始めると伸長を停止する X-dropoff を利用している．また，アラインメントにおいて連続したギャップが挿入される際に，そのペナルティーを軽減する affine gap を考慮した gotoh アルゴリズム [10] を利用している．

3.　評価実験

3.1　計算環境およびプログラム

本研究で提案した配列相同性検索手法を実装したプログラムの計算速度と精度を評価する実験を行った．実験には東京工業大学の TSUBAME 3.0[11] q_core 環境を用いた．この環境の CPU は Intel Xeon E5-2680 v4 (14core, 2.4GHz) のうち 4core，メモリは 30GB である．また OS は SUSE Linux Enterprise Server 12 SP2，コンパイラは gcc の version 4.8.5 を利用し，コンパイル時の最適化オプションとして -O3 を指定した．

比較対象プログラムとしては BLAST[4], [5] (version 2.7.1)，RAPSearch2[6] (version 2.22)，GHOSTX[12] (version 1.3.7)，GHOSTZ[7] (version 1.0.2)，DIAMOND[8] (version 0.9.14.115) を利用した．各プログラムのオプションでは，スコア行列に BLOSUM62，配列の低複雑度領域を無視する seg フィルタを使用しない，クエリに対するアラインメント出力件数を 10 件，また 1 スレッドで計算を実行する，とした．実際に指定したオプションを**表 1** に示す．

3.2　使用データ

タンパク質配列データベースには 2019 年 2 月に取得した KEGG GENES prokaryotes[13], [14] を使用した．このデータベースはタンパク質配列約 1,771 万本から構成されており，合計残基長は約 56 億残基である．クエリデータとしては NCBI Sequence Read Archive から取得した SRR5788325[15] を利用した．SRR5788325 は illumina 社の NextSeq 550 で読み取られた南太平洋の海洋メタゲノムであり，平均残基長 150 の塩基配列約 3,679 万本から構成されている．本研究では SRR5788325 に対して FastQC (version 0.11.5) を用いてクオリティコントロールを行った上で，1 万配列をランダムに選択しクエリとした．

表 2　提案手法におけるパラメータ探索範囲
Table 2　parameter range of proposed method

パラメータ	探索範囲
(H_1, H_2)	{(0, 0),(0, 1),(0, 2),(1, 1)}
L_1	{2, 4, 6, 8}
A_1	{4, 6, 10, 14}
L_2	{2, 3, 4, 5, 6}
A_2	{4, 6, 10, 14}

3.3　提案手法のパラメータを変化させたときの性能評価

提案手法の検索精度を評価するために，各クエリ配列に対して最適アラインメントを計算する SSEARCH の出力を正解とし，提案手法の出力が正解を網羅する割合を網羅率として計測する．

網羅率の計算手順

(1) SSEARCH を実行し，各クエリの出力上位 10 件の中から E-value 10^{-5} 以下のアラインメントをそのクエリに対する正解アラインメントとする．
(2) 提案手法における各クエリの出力上位 10 件に含まれる正解アラインメント数を網羅数とする．このとき提案手法における E-value，およびアラインメントの詳細は考慮しない．
(3) 提案手法の網羅率 = 提案手法の総網羅数 / SSEARCH の総正解アラインメント数 として計測する．

提案手法においては**表 2** に示すパラメータ探索範囲について，網羅的に実行した．

結果を**図 4** に示す．横軸に網羅率，縦軸に CPU 時間の対数を取った値をプロットしている．図 4 より，色分けされた H_1, H_2 の各組み合わせに対して，固有のパレート面（自身より網羅率が高く，かつ CPU 時間が短いパラメータが存在しないパラメータの集合が形成する面）が見られる．

灰色で示した点は，全長で完全一致するシードのみを探索する．網羅率が 0.7 以下の範囲では CPU 時間が他の点に比べて短いパラメータが多く，精度より速度を重視する

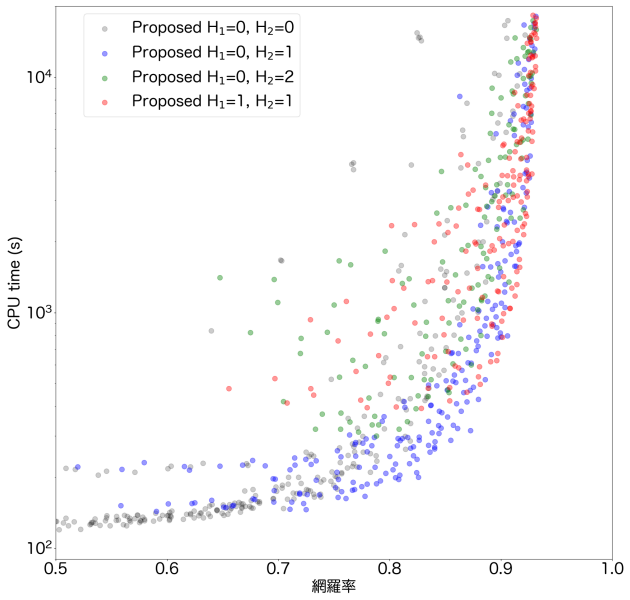

図 4　提案手法のパラメータを変化させた場合の網羅率と CPU 時間の変化

Fig. 4　Correct alignment rate and CPU time on various parameters of proposed method

検索に対して優れたパラメータであることが分かる．

一方赤色で示した点は，網羅率 0.9 以上の範囲で CPU 時間が短いパラメータが多く，速度より精度を重視する検索に優れている．これはシード探索一段階目の seed1 でハミング距離 1 を許容する設定であり，類似度は高いがギャップを多く含むシードを探索することができていると考える．

青色で示した点は，灰色の点と赤色の点の中間にあたる特徴を持っている．特に網羅率 0.7 以上 0.9 以下の範囲では，速度・精度のバランスが良い検索を行うパラメータが多く存在することがわかる．

緑色で示した点では，検索のバランスに優れるパラメータが殆ど存在しなかった．提案手法では，クエリ配列の部分文字列キーに対して指定されたハミング距離以下で一致する部分文字列集合を生成し，それらをインデックスに対して検索する．$(H_1 = 0, H_2 = 2)$ の検索性能が悪化しているのは，H_2 が大きくなるにつれて生成される部分文字列集合のサイズが急激に増加し，シード探索のオーバーヘッドが増大しているためであると考える．

全体を通して，シード長が長く圧縮アミノ酸種類数が少ないパラメータは，検索精度・速度共に悪くなる傾向にあった．これは少ない圧縮アミノ酸に変換した上で長い範囲が一致するデータベース位置を探索するシード探索が，類似度が高いシードを上手く絞り込むことができないためであると考える．

3.4　既存手法との比較

提案手法と既存手法の比較を行った．精度の評価には，3.2 節で述べた網羅率を用いる．また計算時間を評価するために，BLAST の CPU 時間を 1 としたときの計算速度比を用いた．結果を**表 3** および**図 5** に示す．

表 3　網羅率と CPU 時間，および BLAST に対する速度比

Table 3　Correct alignment rate, CPU time and speedup ratio to the BLAST

プログラム	網羅率	CPU 時間 (s)	BLAST からの速度比
BLAST	0.950	139,199.63	1.0
RAPSearch2 fast	0.755	220.99	629.9
RAPSearch2	0.896	2,392.03	58.2
GHOSTX	0.881	3,123.73	44.6
GHOSTZ	0.894	1,241.25	112.1
DIAMOND	0.851	326.06	426.9
DIAMOND sensitive	0.903	12,402.30	11.2
DIAMOND more sensitive	0.904	12,747.51	10.9
Proposed α	0.906	797.32	174.6
Proposed β	0.914	1,318.55	105.6

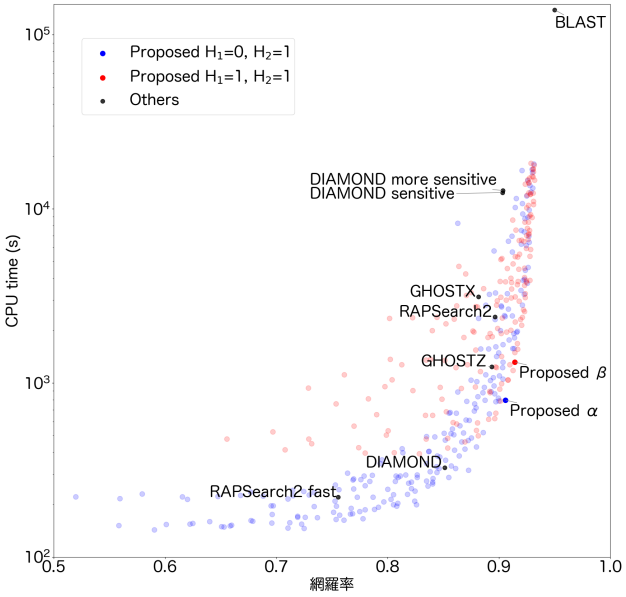

図 5　提案手法および既存手法の網羅率と CPU 時間

Fig. 5　Correct alignment rate and CPU time

図 5 では，3.2 節で性能が高かったパラメータ $(H_1 = 0, H_2 = 1), (H_1 = 1, H_2 = 1)$ を提案手法として記載している．また，提案手法の中で GHOSTZ と比べて CPU 時間が短く，網羅率が高いパラメータを Proposed α，GHOSTZ と同程度の時間で最も網羅率が高いパラメータを Proposed β として示した．各パラメータの詳細は以下の通りである．

Proposed α：

$$H_1 = 0, H_2 = 1, L_1 = 4, A_1 = 14, L_2 = 4, A_2 = 6$$

Proposed β：

$$H_1 = 1, H_2 = 1, L_1 = 8, A_1 = 10, L_2 = 3, A_2 = 6$$

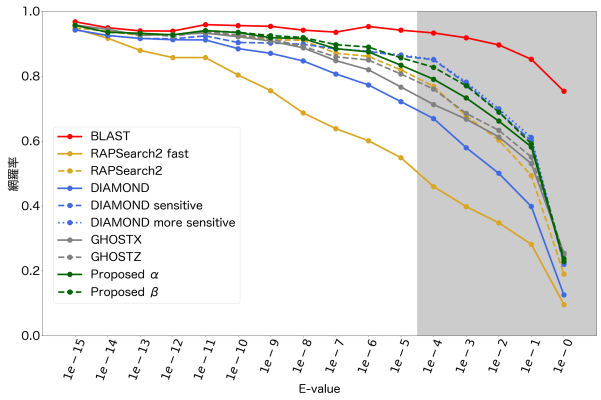

図 6 E-value を変化させた場合の網羅率の変化
Fig. 6 Correct alignment rate for each E-value

実験では BLAST の精度が最も高く，網羅率は 0.95 であった．しかし BLAST の CPU 時間は 14 万秒近くに達しており，他のツールと比較すると 10 倍から最大 600 倍程度低速である．

次に提案手法である Proposed β の精度が高かった．BLAST と比較して網羅率は 0.036 程度低下しているものの，100 倍以上の速度比を達成している．Proposed α や DIAMOND sensitive，DIAMOND more sensitive の精度は同程度である．しかし実行速度では Proposed α は BLAST に対して 174.6 倍高速であり，DIAMOND sensitive や DIAMOND more sensitive に対しても 15 倍以上高速であった．さらに，Proposed α は GHOSTX，RAPSearch2，GHOSTZ などに比べると網羅率が高く，かつ CPU 時間も短いため，より効率的な検索を行うことができているといえる．また DIAMOND や RAPSearch2 fast など検索の速度を重視したツールと比較しても，提案手法の中でより精度が高くかつ高速なパラメータが存在していることが分かる．

SSEARCH における正解セットを作成する際に対象となる E-value の範囲を変化させた場合の各手法の網羅率の変化を図 **6** に示す．基本的には，E-value の範囲によって，各ツールの精度の良い・悪いという上下関係は入れ替わらない．例外として，DIAMOND sensitive，DIAMOND more sensitive は図 5 で示した E-value 10^{-5} 以下の範囲では Proposed β よりも網羅率が低かったが，10^{-5} 以上の範囲では Proposed β よりも網羅率が高くなっている．

4. まとめ

4.1 結論

本研究では，異なる圧縮アミノ酸を用いて二段階のシード探索を行う新たな配列相同性検索手法を提案し，その実装と評価実験を行った．BLAST に対しては多少精度が劣るものの，比較した全ての既存ツールに対して同等の精度を保ちつつ，より高速に検索が可能であることを示した．特に，既存の GHOSTZ や RAPSearch2 と同等以上の精度を持つパラメータを用いて，BLAST と比較して 174.6 倍の高速化を達成した．また GHOSTZ に対しては 1.5 倍程度，RAPSearch2 に対しては 3 倍程度の高速化を達成した．

4.2 今後の課題

現状の実行速度は未だ DNA シーケンサーのスループットに対して十分であるとは言えず，今後より一層の高速化を図る必要がある．現状の課題としてシード探索で探索されるアラインメントの候補が多いことや，シード探索にかかるオーバーヘッドが大きいことが挙げられるため，更なる改良が必要である．またデータベース配列に対して構築されるインデックスのサイズが比較的大きく I/O 部分に時間がかかるため，より省メモリな検索を行う必要がある．

またアルゴリズムの改良以外にも，GPU などを利用した計算にも対応する必要があると考える．

参考文献

[1] S. Henikoff and JG. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89(22):10915-10919, 1992.
[2] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195-197, 1981.
[3] W. R. Pearson. Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, 11(3):635-650, 1991.
[4] S. F. Altschul *et al.* Basic local alignment search tool. *J Mol Biol*, 215:403-410(1990).
[5] S. F. Altschul *et al.* Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl Acids Res*, 25(17):3389-3402, 1997.
[6] Y. Zhao *et al.* RAPSearch2: a fast and memory-efficient protein similarity search tool for next-generation sequencing data. *Bioinformatics*, 28(1):125-126, 2012.
[7] S. Suzuki *et al.* Faster sequence homology searches by clustering subsequences. *Bioinformatics*, 31(8):1183-1190, 2015.
[8] B. Buchfink *et al.* Fast and sensitive protein alignment using DIAMOND *Nat Methods*, 12: 59-60, 2015.
[9] L. R. Murphy *et al.* Simplified amino acid alphabets for protein fold recognition and implications for folding. *Protein Eng*, 13(3):149-152, 2000.
[10] O. Gotoh. An improved algorithm for matching biological sequences. *J Mol Biol*, 162(3):705-708, 1982.
[11] TSUBAME とは — [GSIC] 東京工業大学学術国際情報センター. https://www.gsic.titech.ac.jp/tsubame
[12] S. Suzuki *et al.* GHOSTX: An Improved Sequence Homology Search Algorithm Using a Query Suffix Array and a Database Suffix Array. *PLoS One* 9(8):e103833, 2014
[13] M. Kanehisa *et al.* Data, information, knowledge and principle: back to metabolism in KEGG. *Nucl Acids Res*, 42(D1):D199-D205, 2014.
[14] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucl Acids Res*, 28(1):27-30, 2000.
[15] S. Biller *et al.* Marine microbial metagenomes sampled across space and time. *Sci Data*, 5: 180176, 2018