論文 / 著書情報
Article / Book Information

題目(和文)	環境中の微生物叢解析のための超高速な配列解析ツールの開発
Title(English)	
著者(和文)	矢野雅大
Author(English)	Masahiro Yano
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第10498号, 授与年月日:2017年3月26日, 学位の種別:課程博士, 審査員:山田 拓司,黒川 顕,伊藤 武彦,山口 雄輝,中島 信孝
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第10498号, Conferred date:2017/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

学位論文

平成28年度

環境中の微生物叢解析のための
超高速な配列解析ツールの開発

東京工業大学
大学院生命理工学研究科
生命情報専攻

14D23038
矢野　雅大

指導教員　山田　拓司
黒川　顕

第1章　背景と目的

1.1　生物学的研究において計算可能性が持つ意義

近年の生物学的研究においては、科学技術計算の結果に基づいて数多くの新たな事業が開かれてきただけでなく、数多くの新たな研究分野も開かれてきた。例えば、「ゲノムを断片的かつランダムかつ大量に読み取り、その読み取られた断片配列を計算によって繋ぎ合わせる (アセンブルする)」という全ゲノムショットガン法 (Weber and Myers 1997)を手法の一部に用いることによって、1998年まで全体の5%未満しか解読されていなかったヒトゲノム配列は、2001年にその大部分の解読が宣言され (Venter *et al*. 2001)、そのヒトゲノム配列の情報をもとにがんの早期発見を目指す研究が行われるようになった (Hanashi *et al*. 2002)。

科学技術計算では、計算機環境に応じて計算可能な問題と計算不可能な問題の境界が変化する。Venter *et al*. 2001の研究では断片配列をアセンブルする過程で、64 GBの共有メモリと16個のプロセッサを持つ計算機と10個のスイッチモード電源を用いて、合計20,000 CPU時間もの計算を行った。ここで、計算機の計算速度の向上とメモリの大容量化が、トランジスタの集積度の18カ月で2倍という速度での指数的な向上に伴い、1960年代より増加し続けてきていたこと (Mack 2011)から逆算すると、この計算は1990年以前には極めて困難であったと考えられる。

また、一般に同じ問題を解くための計算であっても、使用するアルゴリズムによって必要な計算量は大幅に変わってくる。例えばフィボナッチ数列を計算していく問題では、動的計画法を用いた場合では計算量は$O(n)$であるが、再帰的アルゴリズムを用いた場合では計算量は$O(2^n)$になる。つまり、この問題においてnが十分大きいならば、動的計画法では容易に計算が終わるが、再帰的アルゴリズムでは計算が現実的な時間内に終わらないという場合も生じうる。

計算に基づいて知見が得られるか否かが計算機環境と使用するアルゴリズムに影響される以上、可能な限り多くの知見を可能な限り多くの利用者が得られるようにするためには、計算機環境の状況を鑑みた上で、適切なアルゴリズムを用いることで計算可能な問題を見

極め、その問題を計算可能なアルゴリズムを実装し、多くの利用者にその実装を使用させることが有効な手段の一つとなる。

より多くの利用者がより多くの知見を得ることで、その知見を踏まえた様々な新しい研究や事業が促進され、さらに多くの知見が得られる結果にも繋がると考えられる。本研究は、環境中に生息する微生物叢の解析において、より多くの知見をより多くの利用者に提供するために新たなアルゴリズムを開発することを目的として研究を行った。以下で、この目的を詳しく説明する。

1.2 環境中の微生物叢を解析する意義と方法

1.2.1 環境中の微生物叢の生態を探求する意義

土壌や海洋や大気中等といった地球上の様々な環境で、多種多様な微生物が微生物叢として生息している (Pace 1997)。微生物叢が存在している環境には、ヒトの腸内・膣内・皮膚等の、生物の身体の内外も含まれる (Weinstock 2012, Kong 2011)。

微生物叢は、環境中から物質を取り込み、体内で化学反応を起こしてエネルギーを得て、環境中に物質を放出するという代謝活動を行っている (Falkowski *et al*. 2008)。この微生物叢の代謝によって、環境は影響を受ける。例えば、微生物叢の代謝プロセスによってミネラルが溶解する場合があり、あるいは地質学的な時間にわたる微生物の代謝プロセスによって鉱床が形成される場合もある (Haferburg and Kothe 2007)。こうした微生物叢の代謝活動は、生物の身体にも影響を及ぼす。例えば、微生物から植物が栄養分を獲得する場合やホルモン刺激を受ける場合、あるいは微生物が植物に対する病原体を抑制する場合も報告されている (Berg 2009)。

よって、生物の身体の内外を含む環境全般の動態を正しく認識するためには、微生物叢の生態を理解することが不可欠であると考えられる。むしろ、微生物叢の生態を理解することによって、環境の動態がより管理しやすいものとなる可能性も考えられる。環境の動態が管理しやすくなれば、農地管理や健康維持といった様々な社会的課題の解決を前進させることができるとも期待できる。実際に、バイオパイル法によって、炭化水素で汚染された土

壌中に生息する炭化水素を分解する好気性微生物を増加させ、汚染された土壌の浄化が促進できると報告されている (Yergeau *et al*. 2012)。

1.2.2 環境中の微生物叢の解析を困難にする難培養性の問題

微生物の生態を明らかにするために、炭疽菌の純粋培養が達成されて (Koch 1876)以降、微生物学は単一種の微生物を培養することで、多くの微生物種の特徴を明らかにしてきた。だが、微生物叢中の微生物種の大部分は培養が困難である。微生物叢中の微生物種の多くの培養が困難であることは、微生物叢に含まれる微生物のうち培養可能なものの割合を顕微鏡を用いてカウントした微生物数と培養後のコロニー数との比較から概算した結果から明らかになった (Xu *et al.* 1982)。

単一種の培養に依存していては、培養が困難な種も含めた種ごとの特徴については知見が得にくい。そのため、環境中の微生物叢がどのような系統の微生物をどれほどの割合で含んでいるか (微生物叢の系統組成)や、あるいはそれぞれの微生物が環境中で行っている代謝活動の詳細等については、解明されていない部分が多い (Raes and Bork 2008)。

1.2.3 メタ16S解析の方法と成果

しかし、微生物を培養することなしに微生物叢の系統組成を解析する手法に、系統マーカー遺伝子を特異的にPCR増幅したものを利用する方法がある。この系統マーカー遺伝子を用いた微生物叢の系統組成解析において、系統マーカー遺伝子として最も広く使われているのは16S rRNA遺伝子である (Vanwonterghem *et al*. 2014)。16S rRNA遺伝子が使われる理由の一つは、16S rRNA遺伝子配列が、PCR増幅をする際のプライマーをデザインするのに適した進化的に変化が入りにくい領域と、微生物の系統情報を解析するのに適した進化的に変化が入りやすい領域 (可変領域)とから、構成されているためである (Tringe and Hugenholtz 2008)。16S rRNA遺伝子配列の全長は約1,500塩基である (Wintzingerode *et al*. 2001)。16S rRNA遺伝子配列の可変領域は9個所あり、それぞれV1~V9と名付けられている (Yarza *et al.* 2014)。

16S rRNA遺伝子配列を用いた微生物叢の系統組成の解析(メタ16S解析)によって、微生物叢の系統組成について様々な知見が得られてきた。例えば、ヒト乳児腸内微生物叢の系統組成での個体差は、ヒト成人腸内微生物叢の系統組成での個体差に比べて大きいことが分かった (Kurokawa *et al*. 2007)。こうしたメタ16S解析を行うためにQIIME (Kuczynski *et al*. 2012)やVITCOMIC (Mori *et al.* 2010)等の様々なバイオインフォマティクスのツールが開発されている。

メタ16S解析では通常、環境中の微生物叢からDNAを抽出し、そのDNAのうち16S rRNA遺伝子をPCR増幅した後でクローニングし、その後でクローニングされた16S rRNA遺伝子をシークエンシングすることで得られた16S rRNA遺伝子配列について系統推定等の様々な解析を行う (Hamady and Knight 2009)。このうち、16S rRNA遺伝子配列の系統推定では、微生物叢から得られた16S rRNA配列をクエリ配列群として、系統情報が既知の16S rRNA遺伝子の配列からつくられた配列データベースに対しての配列相同性検索が行われることが多い。メタ16S解析における配列相同性検索では通常、16S rRNA遺伝子はアミノ酸配列へと翻訳されないノンコーディングRNA遺伝子である (Griffiths-Jones *et al.* 2005)ため、クエリ配列群と配列データベースは共に塩基配列である。

1.2.4 メタゲノム解析の方法と成果

だが、メタ16S解析のみでは、各微生物がどのような代謝を行っているのかや、各微生物がそうした代謝によって環境や他の微生物に対してどのような影響を及ぼしているのかという側面については、解析が困難なままとなる。メタ16S解析では環境中の微生物叢に含まれる16S rRNA遺伝子配列以外の配列は解析されないからである。

ここで、環境中の微生物叢から抽出したDNAをランダムにシークエンスすることで、微生物叢に含まれる配列の総体についての解析を行うメタゲノム解析を行えば、環境中の微生物叢について系統組成に関する以上の知見が得られる。例えば、メタゲノム解析によって、ヒト腸内メタゲノムを代表する330万の遺伝子が、124人のヒト糞便試料から明らかにされた (Qin *et al.* 2010)。

メタゲノム解析では通常、環境中の微生物叢からDNAを抽出し、そのDNAからシークエンシングライブラリを作成してそれをシークエンシングし、そのシークエンシングされた配列について系統組成および遺伝子機能の推定等の様々な解析を行う (Kunin *et al*. 2008)。このうち、系統組成および遺伝子機能の推定では、微生物叢から得られた塩基配列をクエリ配列群として、系統情報あるいは遺伝子の機能情報が既知のゲノム配列からつくられた配列データベースに対しての配列相同性検索が行われることが多い。このような既知のゲノム配列からなる配列データベースと微生物叢から得られた塩基配列との配列相同性検索をメタゲノム解析において行うことで、アミノ酸配列に翻訳されない部分をも含めたゲノム配列全体の解析が可能になる。

1.3 メタゲノム解析およびメタ16S解析における配列相同性検索の計算量の問題

1.3.1 配列相同性検索の計算量

上述の通り、メタ16S配列でもメタゲノム配列でも、共に配列相同性検索が行われている。この配列相同性検索は、メタゲノム解析やメタ16S解析に限らず、DNA配列の相同性を検出して新たな知見を得ようとする様々な試みに使われてきた。複数のデータベースを構成する配列 (データベース配列)を配列データベースとし、複数のクエリとして扱われる配列(クエリ配列)をクエリ配列群として、配列データベースとクエリ配列群との間で配列相同性検索を行うことで、それぞれのクエリ配列に関する情報をデータベース配列の情報から推定できるためである。

配列相同性の計算には、Needleman-Wunschアルゴリズム (Needleman and Wunsch 1970)やSmith-Watermanアルゴリズム (Smith and Waterman 1981)が用いられてきた。その中でも特に、配列データベースとクエリ配列群との間にある局所的な相同性を検出できる Smith-Watermanアルゴリズムがよく使われてきた。Smith-Watermanアルゴリズムにおける探索空間は、二本の配列の間にある相同性を検出する場合では、その二本の配列を縦の辺と横の辺とした長方形で表現できる。この探索空間の全体に対して計算を行う配列相同性検索

ツールにSSEARCHがある (Pearson 1991)。しかし、Smith-Watermanアルゴリズムの計算量は、探索空間のサイズとともに、配列データベースの分量とクエリ配列群の分量の両方に比例する。近年、配列データベースの分量とクエリ配列群の分量は共に増大傾向にあるため、メタゲノム解析とメタ16S解析に必要な計算量も、急激に増大する傾向にある。

この増大する配列相同性検索における計算量は、メタ16S解析とメタゲノム解析のリアルタイム化と一般化を妨げる要因の一つになっていると言える。なぜならば、解析手法の複雑さは前述のQIIMEやVITCOMICによるパッケージ化により縮減できるが、その解析にかかる計算時間は解析手法のパッケージ化だけでは解決できないからである。

1.3.2 クエリ配列群を増大させているシークエンシングの安価化

クエリ配列群の分量が増大したのは、DNAシークエンサーの性能が上がったことによって、シークエンシングの安価化が飛躍的に進行したからである。特に2007年から2011年にかけて、計算機のおおよその性能向上速度を予測するのに有効なムーアの法則をはるかに上回る爆発的な勢いで、DNA配列1塩基あたりのシークエンスコストが減少した (Wetterstrand 2016)。この時期に新たに現れたシークエンサーとして代表的な機種にはIllumina社のHiSeq等がある。2016年12月の時点ではIllumina社のHiSeq 2500は、1回のシークエンシング (6日超かかる)で250 bpのDNA配列を最大で約1 Tb以上読み取ることができる。Illumina社の新型シークエンサーは、メタゲノム解析を含む多くの大規模なシークエンシングプロジェクトに使われてきた。 例えば、Human Microbiome Project (HMP)は、Illumina Genome Analyzer IIxシステムを用いて、健康なヒト成人由来の微生物叢から合計8.8 Tbの配列を生成した (The Human Microbiome Project Consortium 2012)。こうしたシークエンシングの安価化によって、クエリ配列群の量が増大したのである。

1.3.3 シークエンシングの安価化による配列データベースの巨大化

配列データベースの量が増大している主たる原因も、シークエンシングの安価化である。シークエンシングの安価化によって、より多くの微生物からDNA配列を大量に得られるよう

になった。このDNA配列をアセンブルすることで、より多くの微生物のゲノム配列が解読可能になったのである。具体的には、NCBIのデータベース上で公開されている細菌のドラフト/完全ゲノム配列の数が、2011年時点では2,000弱であったのが、2014年時点では30,000以上になっている (Land *et al*. 2015)。昨今では、メタゲノム配列から直接、その微生物叢に生息する微生物のゲノムを解読するというツールも開発・発展してきている (Li *et al*. 2015)。

1.4 メタゲノム解析およびメタ16S解析における計算時間を削減する試み

1.4.1 配列相同性検索ツールの発展

配列相同性検索を高速に行う目的で、相同性がある程度以上に高いと期待できる箇所にのみSmith-Watermanアルゴリズムを用いることで計算速度を向上させた配列相同性検索ツールBasic Local Alignment Search Tool (BLAST) (Altschul et al. 1990)が開発された。だがシークエンサーの発達によるクエリ配列の増加や、データの蓄積による配列データベースの巨大化に応じて、BLASTよりもさらに高速な配列相同性検索アルゴリズムが開発されてきた。それらは BLASTよりもさらに高い相同性を期待できる箇所にのみ Smith-Waterman アルゴリズムを用いて配列相同性を検出したり、あるいは極端に高い相同性のみに注目することで、Smith-Waterman アルゴリズムの使用を回避してきた。前者の代表的なツールにはBLAST-like Alignment Tool (BLAT) (Kent 2002)が挙げられ、後者の代表的ツールにはBowtie (Langmead *et al.* 2009)が挙げられる。これらのツールでは検出できる相同性が限定されている。例えばBLATは90%以下の相同性を検出しようとすると感度が低下し始め(Kent 2002)、Bowtieでは挿入や欠失を含む相同性の検出が困難である (Langmead *et al.* 2009)。なお、配列相同性検索における感度を維持しつつさらなる高速化を実現する方法として、アミノ酸の置換頻度に関する既知の知見を利用して高速に高い相同性を期待できる箇所を絞り込むというRAPSearch (Ye *et al*. 2011)で実装されたアルゴリズムがあるが、このアルゴリズムは配列データベースがアミノ酸配列から構成されている必要があるためアミノ酸に翻訳されない箇所を含むゲノム全体の解析はできない。このアミノ酸の置換頻度を利用し

た超高速かつ高感度な配列相同性検索ツールには、GHOSTX (Suzuki *et al*. 2014)やDIAMOND (Buchfink *et al.* 2015)がある。

使うべき配列相同性検索アルゴリズムは、どの程度低い相同性まで検出することが必要かによって変わってくる。例えば、既にゲノムが解読されて完全ゲノム配列が得られている生物種において、個々の個体が持つ一塩基多型 (SNPs)を明らかにする目的では、Bowtieを改良して挿入や欠失に対応し長いクエリ配列群にも対応したBowtie 2 (Langmead and Salzberg 2012)や、BWA (Li and Durbin 2009)とBWA-SW (Li and Durbin 2010)や、SOAP2 (Li 2009)等の配列リシークエンスデータ用マッピングツールが適している。

しかしメタゲノム解析においては、これらの配列リシークエンスデータ用マッピングツールを用いて得られる知見は極めて限定的なものとなる。メタゲノム解析では、クエリ配列が由来した微生物と系統的に離れた微生物の配列しか配列データベース中に存在しないことが多いため、配列リシークエンスデータ用マッピングツールによってではクエリ配列と相同な部分を持つデータベース配列を検出できない場合も多くなってしまうためである。

そのため、メタゲノム解析にはBLASTを代表とした、低い相同性までも検出できるツールが用いられてきた。その中でも特にBLASTとBLATは、現在行われてきたメタゲノム解析の結果の多くに影響を及ぼしている。例えば、メタゲノミクスによく使用されている系統推定および機能推定ツールであるMEGANはBLASTの結果を使用し (Huson 2013)、一般的に使用されているメタゲノム解析Webサービスであるmg-rastはBLATを配列相同性検索に使用している (Wilke 2013)。

だがBLASTもBLATも、元々はメタゲノム解析のために設計されたツールではなかった。そこで、メタゲノム解析のために、検出可能な相同性をなるべく限定することなく、配列相同性検索にかかる時間を短縮するツールが開発されてきた。その一例が Fragment Recruitment at High Identity with Tolerance (FR- HIT)である。FR-HITはBLASTと同程度の感度を維持しつつBLASTよりも高速な配列相同性検索を可能にしたツールであった (Niu *et al.* 2011)。

1.4.2 計算機の性能向上による超並列解析の発展

しかし2000年代後半以降、画像処理に使用されていた計算ユニットであるGraphics Processing Unit (GPU)が持つ数百以上の計算コアで並列に計算を行うことで、CPUで計算を並列に行わなかった場合の百倍以上の計算速度を発揮できるツールが数多く開発されてきている。GPUを画像処理以外の汎用目的に転用した計算をGeneral-Purpose GPU (GPGPU)と呼び、非常に多くの計算スレッドを起動しての計算を超並列での計算と呼ぶ。GPUはそもそも画像処理という条件分岐が少ない画一的な計算を超並列で行うために発達してきた計算デバイスであり、科学技術計算であってもその計算が条件分岐が少なくかつ超並列で実行可能であるならば、高い計算性能を発揮できる (Nickolls and Dally 2010)。しかも、GPUは計算機の消費電力あたりでの計算速度を高くする目的でも優れている。具体的には2016年11月現在、LINPACKベンチマーク時の消費電力当たり計算速度が高い計算システムを記録したGreen500の一位と二位はNVIDIA社製のGPUを使用している (https://www.top500.org/green500/lists/2016/11/)。具体的なGPUの用途としては画像認識における深層学習などがあり、FacebookやAdobeをはじめとした様々な組織がGPGPUに対応した深層学習ライブラリであるConvolutional Architecture for Fast Feature Embedding (Caffe)を用いている (Jia *et al*. 2014)。

かつてGPGPUの欠点として、プログラミングの難しさがあった。GPU 上で動作するプログラムの作成では、画像処理以外の計算が困難だったためである。しかし、様々なアーキテクチャの計算機での連携のために策定されたフレームワークであるOpenCL (Stone *et al.* 2010)や、NVIDIA社が設計した統合開発環境であるCUDA (Compute Unified Device Architecture) (Nvidia 2007)等の登場と発展によって、GPU上で動作する画像処理以外の用途のプログラムのプログラミングは以前よりも容易になった。

また、GPGPUの他の欠点として、GPGPU環境は個人で整えることも十分に可能ではあるものの、それでもGPGPU環境を整えるのには金銭的および技術的なコストがかかるというものがある。しかしGPGPU環境は近年多くのスーパーコンピュータに標準で搭載されるようになってきており、このコストに関する問題は緩和されつつある。例えば、DNA Data Bank of

Japan (DDBJ)のスーパーコンピュータや東京工業大学のスーパーコンピュータTSUBAME 2.0 の計算ノードでは、GPUは標準で搭載されており (Turkel 2012)、それぞれCUDAが標準で使用可能となっている (https://sc.ddbj.nig.ac.jp/index.php/en/en-programming, http://tsubame.gsic.titech.ac.jp/docs/guides/tsubame2/html_en/programming.html)。こうした計算機環境では、ユーザーはGPGPU環境を整えるコストなしでGPUを用いるツールを開発・使用することができる。

そのため、GPUを用いて計算を並列化することで、配列相同性検索も十分な感度を維持しつつ高速化できる可能性を期待できる。だが、GPUを用いて配列相同性検索を行うツールとして CUDASW++ 2.0 (Liu *et al.* 2010)、GPU-BLAST (Vouzis and Sahinidis 2011)、GHOSTM (Suzuki *et al.* 2012)、G-BLASTN (Zhao and Chu 2014)、MUMmerGPU (Schatz *et al.* 2007)、SARUMAN (Blom *et al.* 2011)等の多くのツールが開発されてきたものの、これらのツールはどれも、メタゲノム解析における塩基配列の配列相同性検索のために作られたツールではない(表1.1)。つまり、これらのツールは、数千本以上の既知のゲノム配列からなる配列データベースと数百万本以上のDNAの断片配列からなるクエリ配列群との間にある弱い相同性を超高速に検出する目的には作られてはいない。よって、これらのツールよりもメタゲノム解析に適したツールの開発を試みることで、メタゲノム解析における配列相同性検索を高速化できる可能性があった。配列相同性検索の高速化により、メタゲノム解析のリアルタイム化と一般化が促進できるだろうと考えられる。

1.4.3 配列クラスタリングによる配列数の削減

配列相同性検索を高速化する手段の一つに、高い相同性が認められる複数のクエリ配列を配列クラスタリングによってまとめてしまうというものがある (Li *et al*. 2012)。特にメタ16S解析では、配列相同性検索を行う前に、同じ種に属すると考えられる16S rRNA遺伝子配列を一つにまとめることが多い。同じ 種に属するメタ16S rRNA遺伝子配列は、一般に可変領域では相互に97%の相同性が認められる (Drancourt *et al*. 2000)ので、メタ16S解析における配列クラスタリングでは、相互に97%以上の相同性があるクエリ配列がクラスタリングによってまとめられることが多い。この配列クラスタリングは、(例えば数百以上などの)大量のサン

プル由来のメタ16S配列を現実的な計算資源および計算時間で解析するためには必要不可欠である。

メタ16S解析に使うことができる配列クラスタリングツールは多く開発されてきた。代表的なものにCD-HIT (Li and Godzik 2006)、UCLUST (Edgar 2010)、DNACLUST (Ghodsi *et al.* 2012)、Swarm2 (Mahé *et al*. 2015)、ESPRIT-TREE (Cai and Sun 2011)等がある。しかし、これまでの配列クラスタリングツールでは速度と精度は相反する傾向があった。速いツールにはUCLUSTとDNACLUSTとSwarm2等を挙げることができ、精度が高いツールにはESPRIT-TREE等を挙げることができる。そのため、速度と精度の両方を高いレベルで両立させた配列クラスタリングツールの開発を試みることで、メタ16S解析を高速化できると期待できた。配列クラスタリングの高速化によって、メタ16S解析のリアルタイム化と一般化も促進できるだろうと考えられる。

1.5 本研究の目的

急激に発達している新型シークエンサーによって、微生物叢に由来して出力されるDNAの断片配列の分量と微生物に関する配列データベースの分量が急激に巨大化してきている。その結果、配列を解析する際に必要な計算量も急激に増大してきている。特に、配列相同性検索の計算量は配列データベースの分量とクエリ配列群の分量の両方に比例しているため、その計算量の増大は爆発的である。この配列相同性検索の計算量の爆発的増大は、環境中に生息する微生物叢を理解するという目的の計算工学上の妨げとなっている。そこで、微生物叢の解析における配列相同性検索にまつわるこの計算工学上の問題を解決するために、以下の2つの目的を掲げた。

1) メタゲノム解析で微生物叢の系統情報および遺伝子機能の組成を解析する目的で行われる配列相同性検索を、GPUの高い計算性能を引き出すことで、超高速かつ高感度に行う新しい配列相同性検索アルゴリズムの開発

2) メタ16S解析における配列相同性検索の前にその計算量を削減する目的で行われる配列クラスタリングを、超高速かつ高精度で行う新しい配列クラスタリングアルゴリズムの開発

この目的が達成されたかを検討するため、各アルゴリズムに基づいた実装ついて性能評価を行い、その実装の有効性を考察した。

図表

表1.1 GPUを用いた既存の配列相同性検索ツールの用途とメタゲノム解析における問題点

ツール名	用途	メタゲノム解析に使用する際の問題点
CUDASW++ 2.0	高感度な 配列相同性検索	速度が十分でない (*)
GPU-BLAST	アミノ酸配列での 配列相同性検索	速度が十分でない (**) 塩基配列向けに設計されていない
GHOSTM	アミノ酸配列での 配列相同性検索	塩基配列向けに設計されていない
G-BLASTN	塩基配列での 配列相同性検索	速度が十分でないか、 感度が十分でない (***)
MUMMER-GPU	ゲノム配列対ゲノム配列での配列相同性検索	多数の配列を処理できるように 設計されていない
SARUMAN	短い塩基配列での 配列相同性検索	長い塩基配列向けに設計されていない

(*) CUDA SW ++ 2.0は、GeForce GTX 280というGPU 1枚で、1秒あたり170億セル分だけ、Smith-Watermanアルゴリズムによる計算を行う (Liu *et al.* 2010)。ここから、CUDA SW++ 2.0が長さ100塩基クエリ配列100万本と合計40億塩基の配列データベースとの間で計算を行うと、同条件下では4百万秒 (約46日)かかると試算できる。これでは、CUDA SW++ 2.0は、次世代シークエンシング技術を用いて得られたメタゲノム解析には適しているとは言いがたい。

(**) GPU-BLASTはNCBI BLASTより3倍から4倍ほど速い (Vouzis and Sahinidis 2011)。しかし、この速度はBLAT以下である (Kent 2002)。

(***) セクション2.3で論じられた結果による。

第2章　GPUを用いた超高速配列相同性検索ツールCLAST

2.1　緒言

一般にメタゲノム解析では、BLAST (Altschul *et al*. 1990)、BLAT (Kent 2002)、FR-HIT (Niu *et al*. 2011)等の配列相同性検索ツールを用いて、遺伝情報あるいは系統情報が既知の配列データベースの中から、環境中から得られたDNAの断片配列と相同であると考えられる部分を検出する。このメタゲノム解析での配列相同性検索において、特に配列データベースとクエリ配列群の両方が塩基配列である場合は、16S rRNA遺伝子を始めとしたゲノム配列上のアミノ酸配列に翻訳されない部分についても解析が可能であるため、遺伝情報や系統情報以外にもGC含量などについて様々な解析を行うことができる。現在、高精度でありながら超高速な配列相同性検索としてGHOSTX (Suzuki *et al*. 2014)やDIAMOND (Buchfink *et al.* 2015)等が開発されているが、少なくともGHOSTXとDIAMONDはアミノ酸配列をデータベース配列とした配列データベースが必要であるため、アミノ酸配列に翻訳されない部分の解析が困難である。

しかも、メタゲノム解析における配列相同性検索では、微生物叢から得られた断片配列の多くは既知の微生物から系統的に離れているため、低い相同性でも検出できる高い感度が必要である。それに加えてメタゲノム解析における配列相同性検索では、シークエンシング技術の進歩によって生み出されるデータの量が増加してきたため、その速度も重要である。しかし、高い感度を達成するためには広い探索空間に対してSmith-Watermanアルゴリズム (Smith and Waterman 1981)を用いて相同性の計算を行う必要があるので、感度と検索速度はしばしば相反する。そのため、メタゲノム解析に使用されるアライメントツールには、感度と検索速度のうちいずれか一方を犠牲にしてきたという歴史があった。

十分な感度を維持しながら配列相同性検索を高速化するうえで有効な方法の一つは、計算を並列で行うことである。計算を並列で行う手段の一つに、GPUを用いるという方法がある。しかし、GPUを用いてメタゲノム解析のために設計された塩基配列用の配列相同性検索ツールは、私たちがCLASTを開発するまで存在しなかった。

そこで私たちは、数百万本以上の短い断片配列 (100塩基前後を想定)あるいは長い断片配列 (800塩基前後を想定)をクエリ配列群として扱うことができ、数千本以上の既知のゲノム配列を配列データベースとして扱うことができ、しかもそのクエリ配列群と配列データベースとの間にある弱い相同性 (BLATが検出できない程度を想定)を高速に検出できる配列相同性検索ツールCLAST (CUDA implemented large-scale alignment search tool)を開発した。CLASTはGPUを使用し、グローバルアライメントあるいはローカルアライメントを配列相同性検索に用いることができる。グローバルアライメントは、断片配列から系統的および機能的な推定を容易にするために、ローカルアライメントはモチーフ探索を行うために実装された。さらに私たちは、Q-gramインデックス (Jokinen and Ukkonen 1991)を構築するために新しくアルゴリズムを設計した。このアルゴリズムによって、事前にインデックス化されていない配列データベースでの高速な配列相同性検索が可能になる。この機能は、CLASTのメモリ必要量を最小限に抑え、しかも大規模な配列データベースが頻繁に更新されても容易に対応できるようにするために実装された。CLASTは、NVIDIA社のFermiアーキテクチャおよびKeplerアーキテクチャのGPUに最適化された。

GPUを用いた超高速かつ高精度な配列相同性検索ツールCLASTを、個人で構築したGPGPUに対応した計算機や、あるいはDDBJのスーパーコンピュータやTSUBAME 2.0といったGPUを搭載した計算機サーバーで使用できるようにすることで、多くの利用者がメタゲノム解析における塩基配列同士の配列相同性検索を利用可能にすることがCLAST開発の目的であった。

2.2 材料と方法

2.2.1 類似箇所の検出方法

CLASTは、2段階の処理 (図2.1)によって、クエリ配列群と配列データベースとの間の相同な領域を検出する。第1段階では、CLASTはクエリ配列群と配列データベースとの間で完全に一致する個所を同定する (シードの作成)。第2段階では、CLASTはこれらのシードの

周りでバンド状のグローバルアライメントまたはローカルアライメントを実行する (Chao *et al*. 1992)。

第1段階では、CLASTは、開始端の間隔p塩基でk塩基ずつ読み取って配列データベースからk-merを読み取る (pおよびkはユーザが設定できるパラメータである)。次に、CLASTは、配列相同性検索を劇的に加速するために、配列データベースのk-merから読み取り専用Q-gramインデックスを構築する (読み取り専用Q-gramインデックスを作成するアルゴリズムは以下で説明する)。この実行時にGPU上で読み取り専用Q-gramインデックスを構築するという設計が、CLASTの特徴的な設計である。この際、Q-gramインデックスは使用するメモリ容量の設定に合わせて配列データベースを分割して構築される。この設計があることによって、CLASTは配列データベースが更新されても即座にその配列データベースを配列相同性検索に用いることができる。最後にCLASTは、開始端の間隔1塩基でk塩基ずつ読み取ってクエリ配列群からk-merを読み取って、これをQ-gramインデックスに照合することでシードを作成する。

第2段階では、CLASTは、アイデンティティと相同性スコアとE-value (表2.1) (http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html)を持つ「ヒット」を作成するために、各シードについてバンド状にアライメントを計算する (図2.2 A)。アライメントの計算時間を短縮するために、CLASTは他のシードに隣接するシードのみを選択する (シードが集まっている「シードクラスタ」を見つけ、そこから1つだけシードを選択する。このアルゴリズムの詳細は後述する)。このプロセスは、後続のグローバル (デフォルトで実行：図2.2 B)およびローカル (オプションで実行可能：図2.2 C)のアライメントを行う領域をも大幅に削減するため、計算時間の大幅な削減につながる。続いてCLASTは各シードごとのアライメントで相同な領域を検出し、アイデンティティと相同性スコアとE-valueを計算する。表2.2は、ユーザーがCLASTで結果出力時のE-valueによる閾値、そしてkの値とpの値のようなパラメータを制御するためのコマンドラインオプションについての記述である。

2.2.2 読み込み専用Q-gramインデックスを作成する並列アルゴリズム

私たちは新たに、GPU等の並列アーキテクチャ向けに最適化された、読み取り専用Q-gramインデックスを作成/参照するアルゴリズムを作成した (図2.3)。CLASTが用いるこの読み取り専用Q-gramインデックスはハッシュテーブルでは実装されておらずsuffix arrayで実装されているため、メモリ使用量はハッシュキーの値域に依存するのではなく、読み取り専用Q-gramインデックスに含まれるデータ数に依存する。したがって、この読み取り専用Q-gramインデックスでは、ハッシュキーの値の制限は変数の値の制限と同じである (例えば、キーが64ビットの整数の場合、ハッシュキーの値域は-2^{63}～$2^{63}-1$である)。この設計により、CLASTにおいてkの最大値は31となった (4種類の塩基を2 bitで表現し、ハッシュキーのうち1 bitをk-merにATGCの4種類以外の「異常な」文字が入っているか否かのフラグに使用したため)。

まず、複数の要素からなるデータ (図2.3 A)の各要素のキーと値から、読み取り専用Q-gramインデックスを作成する手順を説明する。読み取り専用Q-gramインデックスは、「データ配列」と「インデックス配列」から構成される。データ配列はソートされた「キー配列」とキー値順にソートされた「値配列」で構成される。インデックス配列は、キー配列から冗長な配列要素を削除した「非冗長キー配列」、各キー値の冗長度をキー値順に記録した「cellSize配列」、およびcellSize配列のexclusive prefix summationを計算した「ゲートウェイ配列」の三つの配列から構成される。Exclusive prefix summationとは、生成される配列のn番目の配列要素を、元となる配列の1番目からn-1番目までの総和とする計算である。

次に、ハッシュキーに相当する「queryKey」を用いて、読み取り専用Q-gramインデックスの中にある対応する値を取得する方法を説明する (図2.3 B)。まずは非冗長キー配列のバイナリ検索で、queryKeyに対応するcellSize配列の要素 (gottenCellSize)と、queryKeyに対応するゲートウェイ配列の要素 (gottenGateway)が得られる。そこから、そのqueryKeyは、値配列の”gottenGateway+1”番目から”gottenGateway+gottenCellSize”番目までの要素 (writeValues)に対応するということが分かる。

最後に、読み取り専用Q-gramインデックスからqueryValue配列に対応する値を書き込む方法を説明する (図2.3 C)。まずQueryKey (通常はqueryValueから生成される)配列によって読み取り専用Q-gramインデックスを参照すると、gottenCellSizeの配列(gottenCellSizeArray)が作成される。次に、gottenCellSizeArrayのexclusive prefix sumationを計算することによって「writeIndexArray」が作成される。このwriteIndexArrayのうち、個々のqueryValueに対応する要素を「writeIndex」という。最後に、queryValueと対応するwriteValuesが、結果配列の”writeIndex+1”番目から”writeIndex+gottenCellSize”番目までの位置に書き込まれる。

CLASTでは、元のデータのキーと値の配列は、それぞれ配列データベース中のk-merのハッシュキーとその位置情報である。CLASTは、既に説明した読み取り専用Q-gramインデックスを作成するアルゴリズムに加えて、反復閾値 (ユーザが調整可能なパラメータ)よりも大きいcellSize配列の要素をゼロで上書きする。これは、あまりにも頻出するk-merは情報量の乏しいk-merであるとみなして、そのk-merからシードが作成されるのを抑えるためである。そしてCLASTでは、queryKey配列とqueryValue配列は、それぞれクエリ配列群から読まれたk-merのハッシュキーとその位置情報である。よってCLASTにおいて、この読み取り専用Q-gramインデックスを作成/参照して得られる結果が、配列データベースとクエリ配列群の間におけるk-merの位置の対応、すなわち完全に一致する文字列の位置を示したシードの配列となる。

また、CLASTでは配列データベースをユーザーが規定したサイズMごとに分割して読み取り専用Q-gramインデックスを構築する。GPU内部のメモリ (VRAM)内で構築できる読み取り専用Q-gramインデックスのサイズには限りがあるからである。そのため、この読み取り専用Q-gramインデックスの構築時間の計算量のオーダーは配列データベースのサイズをNとしたときにO(NlogM)となる。したがって、CLASTで読み取り専用Q-gramインデックスを構築するのにかかる時間は配列データベースのサイズNに比例することとなる。そのため、CLASTの計算時間全体のオーダーも配列データベースのサイズNに関して比例することとなる。

2.2.3 シード数の削減アルゴリズム

まず、シード数の削減アルゴリズムを説明するために、各シードの「周辺領域」を定義する。そのために、クエリ配列とデータベース配列をそれぞれ一辺とする長方形を用意する。この長方形の内部で、辺に対し傾斜45度の線を「斜線」と定義する。シードの周囲で、クエリ配列方向にz文字 (ユーザが調整可能なパラメータ)以内かつ、この斜線の方向にw文字 (ユーザが調整可能なパラメータ)以内の領域を、「周辺領域」として定義した (図2.4 A)。

次に、シード数の削減アルゴリズムの説明をする。最初に、CLASTは配列中の位置によってシードをソートする (図2.4 B)。次に、CLASTは次のシードが周辺領域にないシードを除去する (図2.4 C)。これはシードクラスタを形成していないシードの削除に相当する。次に、CLASTは次のシードが周辺領域にあるシードを除去する (図2.4 D)。これはシードクラスタの中から一つを除いてすべてシードを削除することに相当する。これによって、CLASTは孤立したシードを除去し、次のステップのためにシードクラスタをそれを代表するシードを除いて除去したことになる (図2.4 E)。

2.2.4 データベース配列の分割

ユーザー定義の制限値L (デフォルト値64 Mb)よりも長いデータベース配列は、CLASTのアクセサリツールを使用して、重複区間 (のりしろ)を持ったより短い (1本32Mb程度の)断片配列に分割する方が、VRAM不足が起こりにくくなるため安全である。それは通常、VRAMはCPUから直接アクセスできるメインメモリ (RAM)と比べて小さいため、CLASTが一度にGPU上で扱うことができる配列長の総和についての限界は数十MBから数百MB程度となることが多いからである。今日までに得られた全ての微生物のゲノム長はLのデフォルト値よりも短いので、微生物ゲノムをデータベース配列に用いる場合、現状ではデータベース配列を分割する必要は通常ない。

2.2.5 プロセスごとの実行GPUの指定

各CLASTプロセスは1つのGPUを使用し、ユーザーはCLASTが実行されるGPUを指定できる。この設計があるため、CLASTのプロセスが用いるGPUを制御するのも容易となっている。これは、スーパーコンピュータなどで複数のプロセスのCLASTを起動する際に、CLASTのプロセスごとに使用するGPUを分けることも容易となっている。

2.2.6 検索精度を検討するための精度試験の方法

CLASTの検索精度を測定するために、BLAST バージョン2.2.25、BLAT バージョン34、およびCLASTバージョン 0.1.0の出力結果をSSEARCH (Pearson 1991)バージョン36.3.6の出力結果と比較した (以下、精度試験と呼ぶ)。この試験は6つのフェーズで構成されている。まず、完全に配列が決定され、確立された系統分類情報を持っているNational Center for Biotechnology Information (NCBI) RefSeq Genomeデータベース (2011年10月、4.3 GB、2,314 配列) (ftp://ftp.ncbi.nih.gov/genomes/Bacteria/)の全ての細菌と古細菌の完全ゲノム配列を取得し、これを配列データベースとした。次に、100塩基と800塩基を配列データベースから無作為にコピーすることで2つのクエリ配列群を作成した (100塩基試験では疑似Illuminaシークエンサー配列として100塩基の断片配列を10,000本、800塩基試験では疑似Roche 454シークエンサー配列として800塩基の断片配列を10,000本)。第3に、これらのクエリ配列群を配列データベースに対し、SSEARCH、BLAST、BLAT、およびCLASTを用いて配列相同性検索を行い、相同な領域を検出した (ヒットの作成)。第4に、それぞれのクエリ配列が由来するデータベース配列上にあるヒットを、各アライメントツールの結果から除去した (非自己ヒットの作成)。このステップによって、元のクエリ配列が由来したゲノム配列がない配列データベースを用いてクエリ配列群の配列相同性検索を行った場合と結果が等価になる。第5に、各ツールの結果から各アライメントツールが計算したビットスコアを使用して「最良の非自己ヒット」を選択した。最後に、BLAST、BLAT、およびCLASTは、SSEARCHと同じヒットとアライメント位置を報告したときに、正確にヒットを発見できたとみなした。この精

度テストは、Intel Xeon X5670 6コア 2.93 GHz CPU、48 GB RAM、2つのNVIDIA Tesla C2050 GPUを搭載したデスクトップコンピュータで実行した。

この精度試験によって、SSEARCH以外のツールが配列が相同性な部分をどれほど正確、そしてどれほどの感度で検出できるかを検討することができる。というのも、SSEARCHは配列データベースとクエリ配列群との間で形成される探索空間の全体についてSmith-Waterマンアルゴリズムを用いた配列相同性検索を行うツールだからである。

2.2.7 速度・感度・系統推定への正確性を検討する疑似メタゲノム解析の方法

一般的に大規模なメタゲノム解析では、系統推定を行うために、ゲノム配列を配列データベースとしてクエリ配列群の配列相同性検索を行う。そこで、私たちは、CLASTの感度および系統推定の精度ならびに計算時間を評価するために疑似メタゲノム解析試験を設計した。

疑似メタゲノム解析試験は、6つの段階から構成されている。まず、精度試験で用いた2,314個の完全ゲノム配列から2つのクエリ配列群 (100塩基または800塩基の100,000本の配列)を精度試験と同様に作成した。次に、精度試験と同様に、各クエリー配列群と配列データベースとの間で配列相同性検索を行い、ヒットを作成した。第3に、精度試験と同様に、非自己ヒットの作成を行った。第4に、精度試験と同様に、各ツールの結果から最良の非自己ヒットを選択した。第5に、最良の非自己ヒットのgenus情報を用いて、クエリ配列の系統推定を行った (図2.5 A)。最後に、任意のデータベース配列から相同な領域を検出できたクエリ配列の数 (ヒット数)と、正しいgenusに割り当てられたクエリ配列の数 (correct genus assignment)の数を数えた (図2.5 B)。私たちは、試験に使われたツール間でのヒット数とcorrect genus assignmentの数、および正しいgenusの割り当てられた比率 (CGA-ratio; ”correct genus assignment”の数/ヒット数)を比較した。ヒット数はアライメントツールの感度の尺度となり、CGA-ratioは系統推定の精度の尺度となる。より感度の高い配列相同性検索ツールは弱い相同性でも検出でき、結果的にヒット数が多くなるため、そうしたツールはモチーフ検索 (図2.5)により有用である。

疑似メタゲノム解析試験ではCLASTバージョン 0.1.0を、BLASTバージョン2.2.25、BLATバージョン34、FR-HITバージョン0.6、Burrows-Wheeler Aligner (BWA) (Li and Durbin 2009)バージョン0.5.9、BWA バージョン0.5.9に含まれるBWA-SW (Li and Durbin 2010)、BWA バージョン0.7.15に含まれるBWA-MEM (Li 2013)、Bowtie 2 (Langmead 2012)バージョン2.0.4、そしてG-BLASTN (Zhao and Chu 2014)バージョン1.1 (BLASTバージョン2.2.28+を使用)と比較した。G-BLASTNはNVIDIA Keplerアーキテクチャの GPU向けに設計されているため、G-BLASTNとCLASTとの比較は他のツールとの比較とは別途に行った。デフォルトのコマンドラインオプションを、試験されたアライメントツール(表2.3)ごとに使用した。BWA / BWA-SWバージョン0.5.9、Bowtie 2バージョン2.0.4、およびBLATバージョン34では、4 GBを超える配列データベースは一度には処理できない (http://genome.ucsc.edu/goldenPath/help/blatSpec.html, http://bio-bwa.sourceforge.net, http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml)。そこで、これらのプログラムを試験するときは、配列データベースを3分割してこれを使用し、そのそれぞれから得られた配列相同性検索の結果を結合した。CIGARコードとMDタグ (BWA)、E-value (FR-HIT)、およびアライメントスコア (Bowtie 2、BWA-SW、BWA-MEM、BLAST、BLAT、およびCLAST)を使用して、最良の非自己ヒットを選択した。疑似メタゲノム解析試験 (G-BLASTNを除く)は、精度試験と同じデスクトップコンピュータ上で実施された。

2.2.8 複数のGPUを使用した場合の計算時間の変化を検討する方法

計算時間に及ぼすGPU数の影響を調べるために、1つ、2つ、8つのGPUでCLASTバージョン 0.1.0を実行した。配列データベースは、疑似メタゲノム解析試験で用いられたものと同じであった。クエリ配列群には実際のメタゲノム配列として、ヒト腸内微生物群からのIllumina Genome Analyzer IIxが出力した断片配列 (NCBI SRA accession number: ERR011343; 75 bp; 21,739,219本)を使用した (Qin *et al.* 2010)。この試験では、4ノードのGPUサーバーを使用した。各ノードには、Intel Xeon X5690 6コア3.47 GHz CPU、64 GB RAM、2つのNVIDIA Tesla C2075 GPUが搭載されていた。

2.2.9 G-BLASTNとの比較方法

CLASTとCPUベースのツールとの比較として、CLASTバージョン 0.1.0の系統推定の速度、感度、精度をG-BLASTN (Keplerアーキテクチャ GPUコンピューティングに最適化されたBLASTアルゴリズム)のそれと比較した。G-BLASTNとの比較のためのデータセットおよび分析パイプラインは、疑似メタゲノム解析試験のものと同じであった。試験には2つのIntel Xeon E5-2687 W 8コア3.10 GHz CPU、62.9 GB RAM、2つのNVIDIA Tesla K20m GPUを搭載したワークステーション(以下、K20x2マシン)を使用した。CLASTアルゴリズムがG-BLASTNと同じ速度を達成する場合、G-BLASTN (デフォルト設定)は自動的に全ての使用可能なGPUを使用し、CLAST (デフォルト設定)は1つのプロセスごとに指定された1つのGPUのみを使用するため、K20x2マシンではG-BLASTNはCLASTの半分の時間で計算を終えると考えられる。私たちはG-BLASTNを高い相同性だけを検出するために設計されたmegablastモードと低い相同性まで検出できるblastnモードで実行して、CLASTと結果を比較した。G-BLASTNに与えたコマンドラインパラメータは、megablastモードで”-use_gpu true -outfmt 6 -task megablast”であり、blastnモードで”-use_gpu true -outfmt 6 -task blastn”であった。

2.3 結果

2.3.1 精度試験の結果

100塩基と800塩基の両方の精度試験では、CLASTの検索精度は、ビットスコアが90以上(100塩基試験)または200以上 (800塩基試験)の場合で、BLASTの検索精度に匹敵していた。ほとんどの場合で、CLASTの検索精度はBLATの検索精度よりも高かった (図2.6)。

2.3.2 疑似メタゲノム解析による速度の結果

100塩基のクエリ配列群を用いた疑似メタゲノム解析試験(100塩基試験)では、BWA-MEMが計算時間17.1秒で最速のツールで、Bowtie 2 (グローバルモード)、Bowtie 2 (ローカルモード)、BWA、CLAST (グローバルモード)、CLAST (ローカルモード)、BLAT、FR-HIT (グローバルとグローバルの両方)、そしてBLASTがこの順に続いた。CLAST (グローバルモード)はBLASTより72.6倍速かった。CLAST (ローカルモード)の速度はCLAST (グローバルモード)とほぼ同等で、BLATよりも2.35倍速かった。

800塩基のクエリ配列群を用いた疑似メタゲノム解析試験 (800塩基試験)でも、BWA-MEMが計算時間261.8秒で最速のツールであった。Bowtie 2 (グローバルモード)とCLAST (グローバルモードとローカルモードの両方)の計算時間は1,300秒程度で同等であった。CLASTの計算時間は、Bowtie 2 (グローバルモード)に匹敵し、BWA-SW (図2.7)よりも速かった。CLAST (グローバル・モード)は、BLATおよびBLASTよりそれぞれ9.64倍および80.8倍高速であった。

2.3.3 疑似メタゲノム解析による感度と系統推定への正確性の結果

100塩基試験では、BLAST (図2.8)が最も多くのヒット数を出力でき、これにFR-HIT (ローカルモード)、CLAST (ローカルモード)、そして残りのツールがこの順に続いた。100塩基試験でのFR-HIT (ローカルモード)とCLAST (ローカルモード)のヒット数は同等であった。800塩基試験でも、最も多くのヒット数を出力したのはBLASTであり、CLAST (ローカルモード)がこれと同等のヒット数を出力した一方で、他のツールはBLASTより10,000以上少ないヒット数しか出力しなかった。この100塩基試験と800塩基試験の結果が示すことは、BLASTとFR-HIT (ローカルモード)とCLAST (ローカルモード)は他のツールよりも高感度を達成したということと、CLAST (ローカルモード)は環境から得られたDNA配列を完全ゲノム配列を用いて作られた配列データベースにマッピングするのに十分に高い感度を持つということである。

両方のクエリ配列長について、BWA-MEM、Bowtie 2 (グローバルモード)、CLAST (グローバルモード)、FR-HIT (グローバルモード)は、他のツールより高いCGA-ratioを達成した。100塩基試験では、BWAとBWA-MEMとBowtie 2 (グローバルモード)が非常に高いCGA-ratio (それぞれ98%と95%と96%)を達成した。だが800塩基試験では、BWA-SWの合計ヒット数はCLAST (グローバルモード)およびFR-HIT (グローバルモード)の0.96倍と1.16倍であったにも関わらず、BWA-SWのcorrectでなかったgenus assignmentの数はCLAST (グローバルモード)とFR-HIT (グローバルモード)の1.58と3.79倍もあった。同様に800塩基試験では、Bowtie 2 (ローカルモード)のヒット数はCLAST (グローバルモード)およびFR-HIT (グローバルモード)の1.10倍および1.33倍あったが、Bowtie 2 (ローカルモード)のcorrectでなかったgenus assignmentの数はCLAST (グローバルモード)とFR-HIT (グローバルモード)の1.80倍、4.31倍となった。これらの結果は、グローバルアライメントが系統推定に有用であることを示している。なお、BWA-MEMはローカルアライメントを用いている(http://bio-bwa.sourceforge.net/bwa.shtml)が、BWA-MEMのヒット数とcorrect genus assignmentの数は両試験でBowtie 2 (グローバルモード)のそれと同等であった (ただし、BWA-MEMのCGA-ratioはBowtie 2 (グローバルモード)のそれよりもわずかに低かった)。BWA、BWA-MEM、Bowtie 2 (グローバルモード)、FR-HIT (グローバルモード)、およびCLAST (グローバルモード)は、高い精度で系統を推定することができた。特にCLAST (グローバルモード)とFR-HIT (グローバルモード)は、系統推定の精度が高かっただけでなく、適度な検索感度も同時に達成できた (図2.9)。つまり、CLAST (グローバルモード)とFR-HIT (グローバルモード)の系統推定の精度はBowtie 2 (ローカルモード)とBWA-SWの系統推定の精度よりも高く、CLAST (グローバルモード)とFR-HIT (グローバルモード)の検索感度はBowtie 2 (グローバルモード)とBWAとBWA-MEMの検索感度よりも高かったのである。

さらに、アイデンティティ閾値およびカバレッジ閾値を変更することで、BLAST、BLAT、およびCLAST (グローバルモードおよびローカルモードの両方)のヒット数とcorrect genus assignmentの数との関係が折れ線として示された (図2.10)。アイデンティティ閾値とは相同だと報告された領域のうち、クエリ配列とデータベース配列とで塩基が一致した部分の割合

に関する閾値である。カバレッジ閾値とは、相同だと報告された領域の長さがクエリ配列の全量のどれほどであったかの割合に関する閾値である。全てのツールの折れ線はどれも上に凸であったが、100塩基試験においてCLAST (グローバルモード)の折れ線は他のツールの折れ線よりわずかに高かった。Bowtie 2 (グローバルモード)の点は、両方の試験で90%のアイデンティティ閾値を持つCLAST (グローバルモード)の点に近かった。Bowtie 2 (ローカルモード)の点が達成したCGA-ratioは、100塩基試験でのBLAST、BLAT、およびCLAST (グローバルモード)の折れ線よりも低かった。また、Bowtie 2 (ローカルモード)の点は800塩基試験ではBLATの折れ線に近かった。

2.3.4 複数のGPUを使用した場合の計算時間の変化を検討した結果

実際のメタゲノム配列を用いたCLASTの配列相同性検索にかかった計算時間は、CLASTが使用したGPU数と概ね反比例の関係を示した (図2.11)。1つのGPUだけでは、CLASTの計算時間は355分 (グローバルモード)と373分 (ローカルモード)であった (図2.11)。2つのGPUを使用した場合ではCLASTの計算時間は188分 (グローバルモード)と192分 (ローカルモード)、8つのGPUを使用した場合ではCLASTの計算時間は49分 (グローバルモード)と50分 (ローカルモード)であった。この結果は、複数のGPUを使用することでCLASTを大幅に高速化できることを示している。

2.3.5 G-BLASTNとの比較結果

疑似メタゲノム解析試験では、K20x2マシンでG-BLASTN (blastnモード)は、クエリ長が100塩基の場合は15,970秒、クエリ長が800塩基の場合は136,560秒かかった。一方、CLASTは同じマシンで、100塩基のクエリ長で210秒 (グローバルモード)、215ベース (ローカルモード)、800塩基のクエリ長で1,248秒 (グローバルモード)、1,352秒 (ローカルモード)であった。つまり、CLASTはG-BLASTN (blastnモード)よりも150~200倍速かった。さらに、G-BLASTN (megablastモード)は、クエリ長が100塩基の場合は199秒、クエリ長が800塩基の場合は724秒であった。したがって、CLASTはGPU 1基あたりの速度ではG-

BLASTN (megablastモード)より1.07~1.85倍速かった。これらの結果は、CLASTがG-BLASTN (blastnモード)よりもはるかに速く、G-BLASTN (megablastモード)よりもわずかに速いことを示している。

クエリの長さが100塩基の場合、G-BLASTN (blastnモード)のヒット数とcorrect genus assignmentの数はそれぞれ99,841と56,151であった (CGA比率:58%)。クエリの長さが800塩基である場合、G-BLASTN (blastnモード)のヒット数とcorrect genus assignmentの数はそれぞれ100,000および62,728であった (CGA比率:63%)。したがって、G-BLASTN (blastnモード)は、疑似メタゲノム解析試験においてBLASTと同様の結果を出力した。この結果は、クエリの長さが800塩基の場合、CLAST (ローカルモード)がG-BLASTN (blastnモード)と同程度の情報を検出できることを示している。

クエリの長さが100塩基の場合、G-BLASTN (megablastモード)でヒット数とcorrect genus assignmentの数はそれぞれ46,720および42,664であった (CGA率:91%)。クエリの長さが800塩基の場合、G-BLASTN (megablastモード)のヒット数とcorrect genus assignmentの数はそれぞれ65,108と52,754であった (CGA比率:81%)。したがって、G-BLASTN (megablastモード)は、疑似メタゲノム解析試験においてBowtie 2 (localモード)と類似した結果を出した。この結果は、CLAST (グローバルモード)の系統推定の精度がG-BLASTN (megablastモード)のそれよりも大きく、CLAST (ローカルモード)の感度がG-BLASTN (megablastモード)の感度よりも大きいことを示している。

2.4 考察

メタゲノムシークエンシングプロジェクトで出力された、莫大な量でありかつその多くは未知の微生物由来の断片配列の解析には、高速かつ高感度な配列相同性検索が必要である。CLASTは、新型シークエンシング技術による大規模なメタゲノム解析向けに最適化された、超高速かつ高感度な配列相同性検索ツールである (図2.12)。ここでは、計算速度と感度の両面でCLASTツールの優れたパフォーマンスが実証できた。CLASTの高速性は、他の専用アクセラレータと比較して安価な傾向があり、しかも強力なためHPC向けに広く使用

されているGPUの使用に大きく起因していると考えられる。CLASTの感度は、大部分がシードの伸長のためにバンド状のSmith-Watermanアライメントの使用に由来すると考えられる。CLASTのk-merの長さ (k-mer長)がデフォルトでは15塩基でありBLATのk-mer長がデフォルトでは11塩基であるにも関わらず、CLASTの感度がBLATの感度よりも高かったのは、CLASTでのk-merの読み取り開始位置が5塩基ずらしである一方でBLATでのk-merの読み取り開始が11塩基ずらしであったためだと考えられる。CLASTの速度および感度は、より長いまたはより短いk値をそれぞれ指定することでさらに改善できる可能性がある。しかし、CLASTはGPUコンピューティングに精通していないユーザーにとっては少し導入が難しいことが危惧される。また、CLASTはメタゲノム解析用に設計されているため、他の用途では他のツールが適していることも多いと考えられる。例えば、BWAやBowtie 2は、完全ゲノムが決まっている生物を対象にしたゲノムリシークエンシングプロジェクトに適しており、これらの解析にはより高感度のツールはそもそも必要ない (図2.12)。こうしたゲノムリシークエンシング用途では、CLASTよりもBWAやBowtie 2等の方が適切だろう。特にBWA-MEMの速度はBWAに含まれる他のプログラムやBowtie 2と比較しても際立っていることから、膨大な量のゲノムリシークエンシング用途には特にBWA-MEMが適していると考えられる。このBWA-MEMの速度は、BWA-MEMが採用している新しいシード作成/利用アルゴリズムによるところが多いと考えられる (Li 2013)。

CLASTの特徴として、その速度と感度に加えて、グローバルとローカルの両方のアライメントが可能であるというものがある。グローバルアライメントは、クエリ配列とデータベース配列のうちクエリ配列に対応する部分の全域での相同性を評価することができる。そのため、グローバルアライメントは断片配列の系統推定に有用である (図2.8)。疑似メタゲノム解析の試験結果が、CLAST (グローバルモード)を含むグローバルにアライメントを行うツールはローカルアライメントを用いたものよりも正確な系統推定が可能である、ということを示している。しかもCLAST (グローバルモード)の感度はBWAやBowtie 2 (グローバルモード)の感度よりも高かった (図2.8)。一方、ローカルアライメントは、メタゲノム解析における機能推定で頻繁に使用されるモチーフ探索 (Dinsdale et al. 2008)に有用である。なぜなら、ローカルアライメントによってクエリ配列とデータベース配列との間にある部分的な配列相同性を検出

できるからである。ローカルとグローバルの両方のアライメントを実行できるというCLASTの特徴は、CLASTのメタゲノム解析における有用性を大幅に向上させている (表2.4)。

大規模なメタゲノム解析では、頻繁に更新される大規模な配列データベースの使用が必要になることがある。CLASTは、大規模な配列データベースでも、一部のデータベース配列を分割するという最小限の前処理のみで使用でき、しかも扱えるデータベースサイズに制限がないため、大規模なメタゲノム解析に非常に適している。ただし、配列データベースが頻繁には更新されない場合、配列相同性検索の度に読み取り専用Q-gramインデックスが構築されるため、その分だけ計算時間が増えることは考えられる。さらに、CLASTの最大メモリ使用量は、配列データベースの総サイズとは無関係である。それは、CLASTが配列データベースとクエリ配列群をともに分割して配列相同性検索を行うからである。

他のいくつかのツールでも大規模な配列相同性検索は実行できるが、CLASTよりも多くのデータベースの前処理と多くのメモリ使用量が必要になる傾向がある。例えば、BLASTにはデータベースの前処理が必要である。この前処理の計算時間は、データベースのサイズと関連している。 BLATは前処理を必要としないが、4 GBを超えるデータベースは使用できない。 FR-HITは前処理を必要としないが、そのメモリ使用量は通常は配列データベースのサイズの2~3倍である。BWA/BWA-SW/BWA-MEMやBowtie 2といったBurrows-Wheeler変換ベースのマッピングツールでは通常、データベースの前処理が必要だが、これらのマッピングツールではブロックソートが使用されるため、一般的に前処理には無視できない時間がかかる。例えば、Burrows-Wheeler変換ベースのツールによるヒトゲノムデータベース(3GB程度)の前処理には、通常は数時間かかる。しかし、微生物ゲノム配列データベースは2014年の時点で5 GBを超えており、NCBIの非重複ヌクレオチド配列データベースは40 GBを超えている。しかも、これらのデータベースは知見が蓄積される限り今後も成長を続けるだろうと予測できる。CLASTは、これらの大規模でしかもこれからも大きくなり続ける配列データベースを、最低限の前処理のみで使用できるという点で、ゲノム解析とメタゲノム解析のための配列相同性検索ツールとして非常に有望であると言える。

CLASTを通常のメタゲノム解析でデフォルト設定で使うと、必要になるメモリ消費量は 通常、RAMで2 GB以下、VRAMでも2 GB以下である。16S rRNA遺伝子のアンプリコン配列解析等のように、少量のクエリ配列群と配列データベースから相同な部分が大量に検出さ

れる場合では、VRAMがさらに消費される可能性があるが、ユーザーは特定のパラメータを指定することによってCLASTが使用するVRAMの量を操作できる。このようにCLASTのメモリ使用量は少なく抑えられるため、特別な大容量メモリのコンピュータを使用できない利用者にとっても、CLASTの使用は大規模なメタゲノム解析を可能にする合理的なアプローチになる。具体的には、4 GB程度のRAMしか持たない通常のデスクトップコンピュータでも、Fermi世代以降のNVIDIA社製GPUを搭載すれば、CLASTによる超高速かつ高精度な配列相同性検索が可能になる。このCLASTの少ないメモリ使用量は、配列データベースおよびクエリ配列群の両方を、RAMに段階的に分割してロードすることで達成された。CLASTは配列データベースのQ-gramインデックスを実行時に作成するが、読み取り専用Q-gramインデックスの作成もGPUによって実行されるため、Q-gramインデックスの作成は実行時間にはそれほど影響を及ぼさない。この機能は、CLASTの最も重要かつ革新的な進歩の1つである。対照的に、BLATおよびFR-HITは、全ての参照ゲノム配列データを同時にRAMにロードするので、より大きな配列データベースを扱うためにはより大きなRAMが必要となる。実際、FR-HITでは、私たちの疑似メタゲノム解析試験において13 GB以上ものメモリを使用した。

CLASTを複数のプロセスを立てて複数のGPUで実行することで、GPUの並列計算能力をさらに活用し、相同性検索を劇的に加速できる。CLASTのプロセスあたりのメモリ使用量は少ないので、複数のGPUと100 GB未満のメモリを搭載したノードを装備しているようなGPUクラスタやスーパーコンピュータでも、CLASTをそのように複数プロセス立てることができる。自前で使用可能な計算資源が限られた利用者は、GPUクラスタやGPGPU環境が整ったスーパーコンピュータでCLASTを複数プロセス立てることで、高感度な配列相同性検索を超高速で行う手段が得られるようになり、メタゲノム解析における配列相同性検索を行う手段をも得られるようになる。

GPGPU環境が整ったデスクトップコンピュータでのCLASTの使用によってメタゲノム解析における配列相同性検索がより多くの利用者にとっても容易なものとなり、GPUクラスタあるいはGPGPU環境が整ったスーパーコンピュータでのCLASTの使用によって多地点の微生物叢から得られたメタゲノム配列をリアルタイムで解析することもできるようになると考えられ

る。より狭い時間間隔かつより狭い空間感覚で微生物叢をモニタリングし、そのそれぞれに関して過去の知見に基づいた意思決定が可能になれば、環境と微生物叢との相互作用に関する知見がよりよく得られるようになるだけでなく、微生物叢の管理を通じた環境の管理もより現実的なものとなると考えられる。

CLASTのソースコードはWWW上に公開してある (https://github.com/masayano)。CLASTの最新バージョンは2017年1月27日現在0.1.5である。CLASTはバージョン0.1.0からバージョン0.1.5にかけて、より細かくVRAMとRAMの使用量が調整できるようになった他、ファイルの読み込みにおける処理がより簡潔になっている。細かくVRAMとRAMの使用量が調整できるようになったことで、メタ16S配列での配列相同性検索などの少量のデータベース配列と少量のクエリ配列のみからでも多量のVRAMが消費されるような計算にもCLASTが使用しやすくなり、利便性が向上した。このアップデートは出力される結果に変化をもたらさない。また、計算時間全体におけるファイル読み込みの時間の割合がわずかである (data not shown)ため、このアップデートによる計算時間の変化もわずかになると考えられる。そのため、CLASTのバージョンを0.1.5に切り替えての試験は行わなかった。

図表

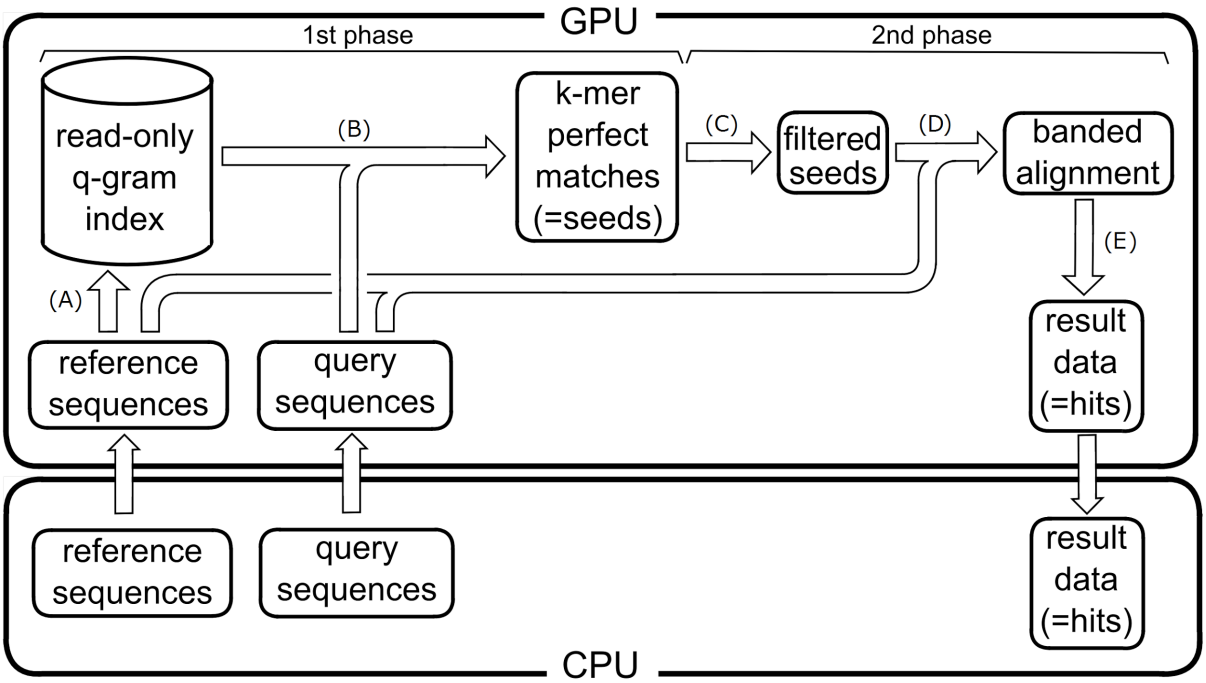


図 2.1 CLAST検索処理フェーズの概要

(A)並列アーキテクチャのために新しくアルゴリズムを設計し、配列データベースから読み取り専用Q-gramインデックスを作成 (図2.3)。(B)読み取り専用Q-gramインデックスを用いて、クエリ配列群と配列データベースとの間でシードを作成。(C)シードをフィルタリングして計算時間を短縮 (図2.4)。(D)シードをもとにアライメントを実行 (図2.2)。(E)アライメントの結果を、E-valueおよびアライメント長に従ってさらにフィルタリング。

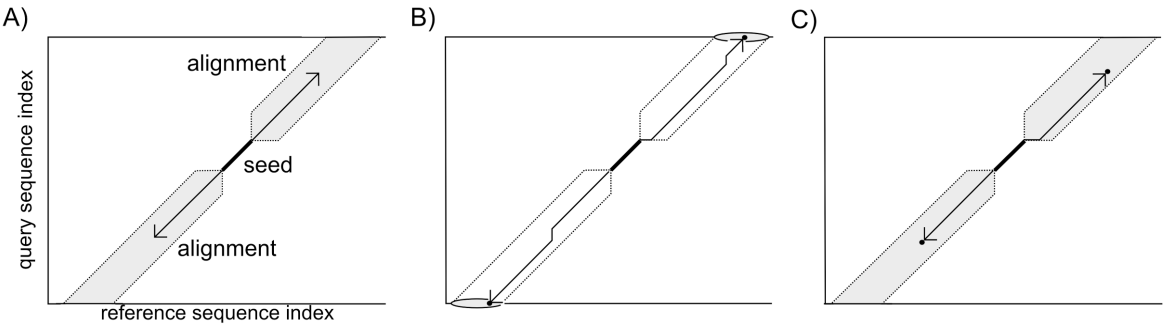


図 2.2 バンド状のグローバルアライメントとローカルアライメント

(A)灰色の領域は、この帯状のアライメントにおいて計算される領域を示す。配列は、グローバルモードとローカルモードの両方において、シードの端からアライメントが行われた。アライメントのパスの終点は、グローバルアライメントでは (B)、ローカルアライメントでは (C)の灰色領域のなかでアライメントスコアが最大となるところとした。

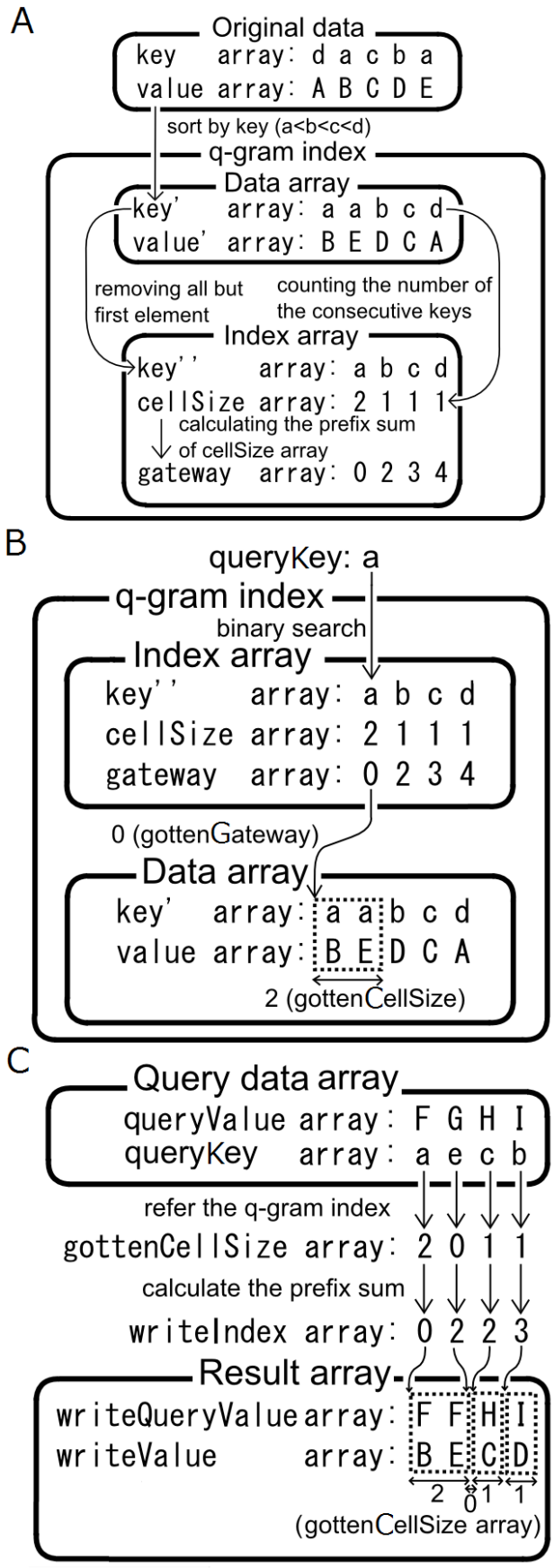

A
Original data
key array: d a c b a
value array: A B C D E
sort by key (a<b<c<d)
q-gram index
Data array
key' array: a a b c d
value' array: B E D C A
removing all but first element
counting the number of the consecutive keys
Index array
key'' array: a b c d
cellSize array: 2 1 1 1
calculating the prefix sum of cellSize array
gateway array: 0 2 3 4
B
queryKey: a
q-gram index
binary search
Index array
key'' array: a b c d
cellSize array: 2 1 1 1
gateway array: 0 2 3 4
0 (gottenGateway)
Data array
key' array: a a b c d
value array: B E D C A
2 (gottenCellSize)
C
Query data array
queryValue array: F G H I
queryKey array: a e c b
refer the q-gram index
gottenCellSize array: 2 0 1 1
calculate the prefix sum
writeIndex array: 0 2 2 3
Result array
writeQueryValue array: F F H I
writeValue array: B E C D
2 0 1 1
(gottenCellSize array)

図 2.3 パラレル・アーキテクチャで読取り専用Q-gramインデックスを作成するためのアルゴリズムの概略

(A)読取り専用Q-gramインデックスを作成する並列アルゴリズム。(B)読み取り専用Q-gramインデックスから、queryKeyに対応する値を取得するアルゴリズム。(C)読み取り専用Q-gramインデックスから、多くのqueryKeyに対応する値を取得する並列アルゴリズム。

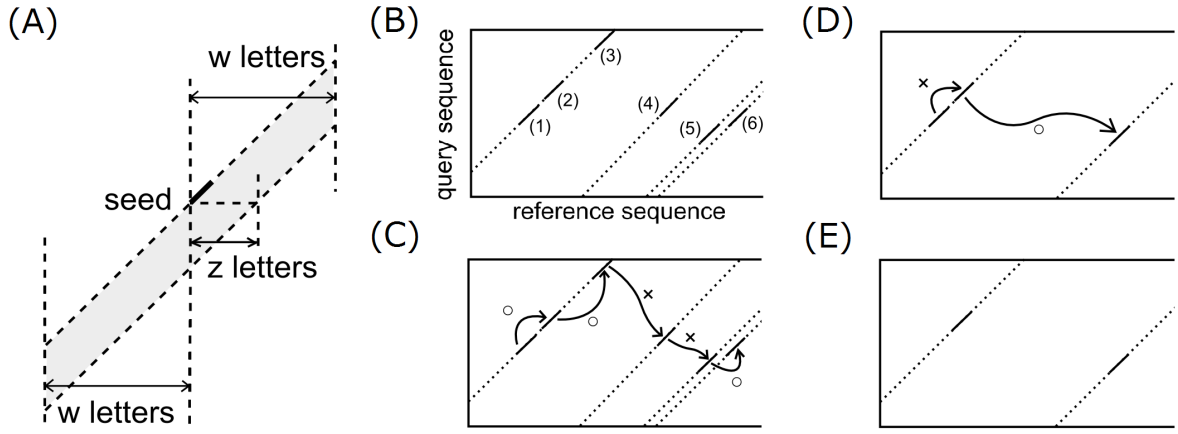
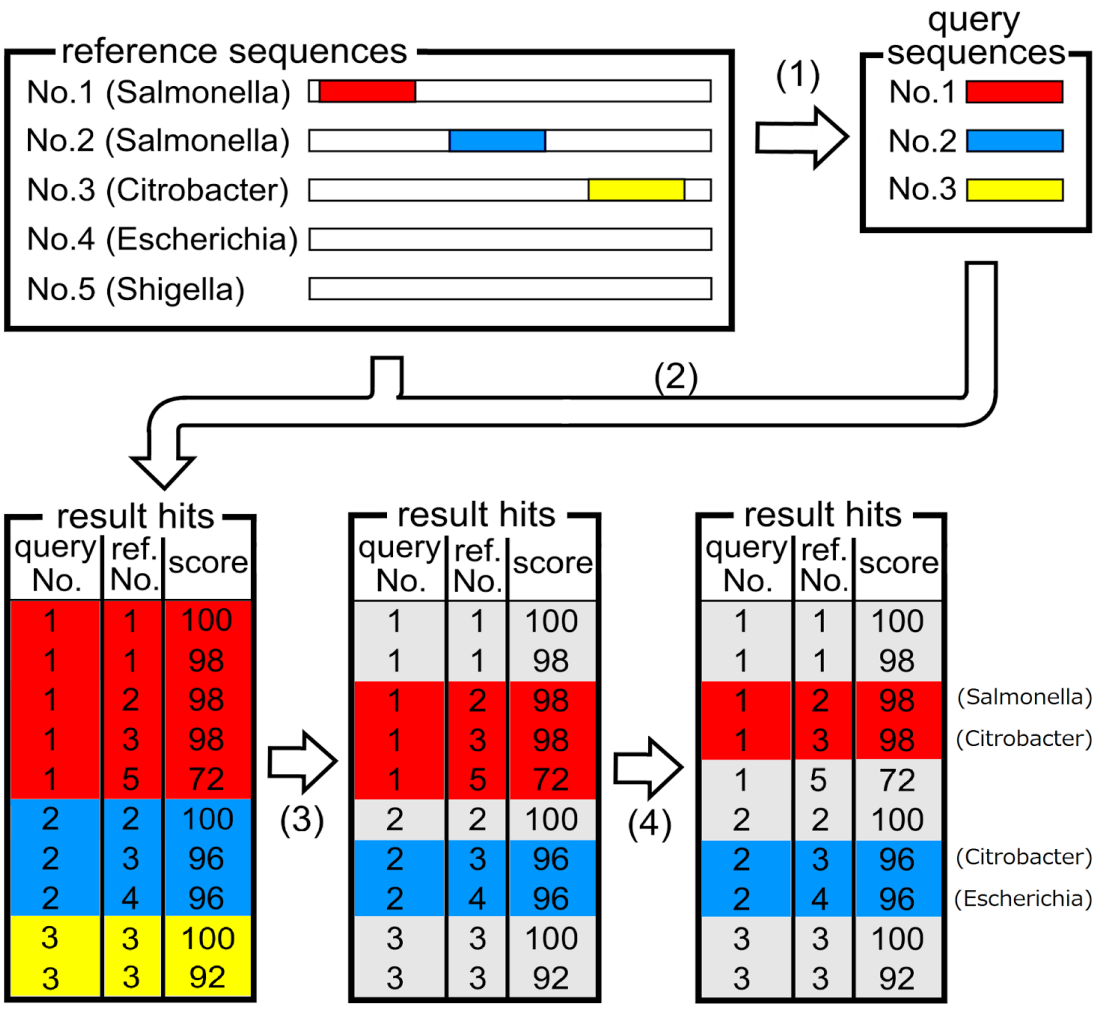


図 2.4 シードの数を減らすアルゴリズム

(A)灰色の領域は、各シードの「周辺領域」を表す。(B)シードの削減例。各シードに付けられた番号は、位置によってソートされたシードの順序を表す。(C)シードをチェックする最初の手順。マル印は次のシードが周辺領域にあることを意味し、バツ印は次のシードが周辺領域にないことを意味する。CLASTはバツ印が付いたシードを除去する。(D)シードをチェックする第2の手順。バツ印は、次のシードが周辺領域にあることを意味する。CLASTはバツ印が付いたシードを除去する。(E)この例で残っているシード。残ったシードは孤立しており、その周辺領域にシードはない。

A



B

taxonomic assignment

query No.	query genus	predicted genus	predicted	correct genus assignment
1	Salmonella	Salmonella Citrobacter	○	○
2	Salmonella	Citrobacter Escherichia	○	×
3	Citrobacter	-	×	×
		Sum	2	1

→ Total reported hits = 2
Correct genus assignments = 1
Correct genus assignment ratio (CGA-ratio) = 50% (1/2)

図 2.5 異なるアライメントツールの検索精度の比較手法

(A)疑似メタゲノム解析試験におけるクエリ配列の系統推定は、以下のステップで実施した。1:配列データベースから短い塩基配列をランダムにコピーすることによって、クエリ配列群を生成。2:配列相同性検索を、クエリ配列群と配列データベースとの間で計算。3:クエリ配列が元のデータベース配列と一致した場合、結果から削除。4:最良の非自己ヒットを系統推定のために選択。(B)系統推定の結果がgenusレベルで正しいか否かを系統データベースに基づいて評価した。

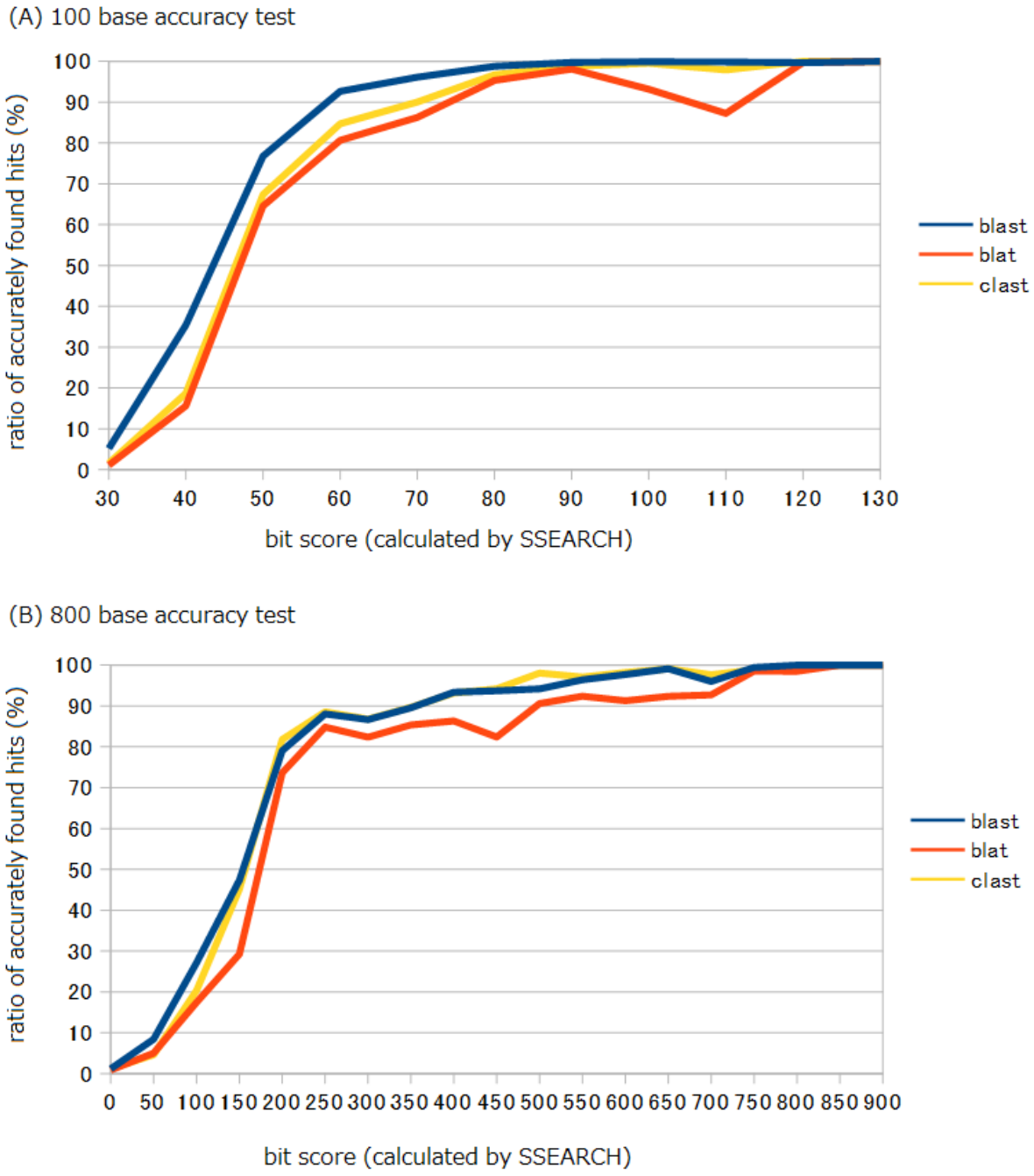


図 2.6 精度試験における各ツールの結果

両方のグラフは、疑似メタゲノム解析試験の結果を表す。 横軸はSSEARCHにより算出されたビットスコアを表し、縦軸は各ツールで正確に検出されたヒットの割合を表す。(A)精度試験のうちクエリ配列の長さが100塩基であった場合の結果。(B)精度試験のうちクエリ配列の長さが800塩基であった場合の結果。

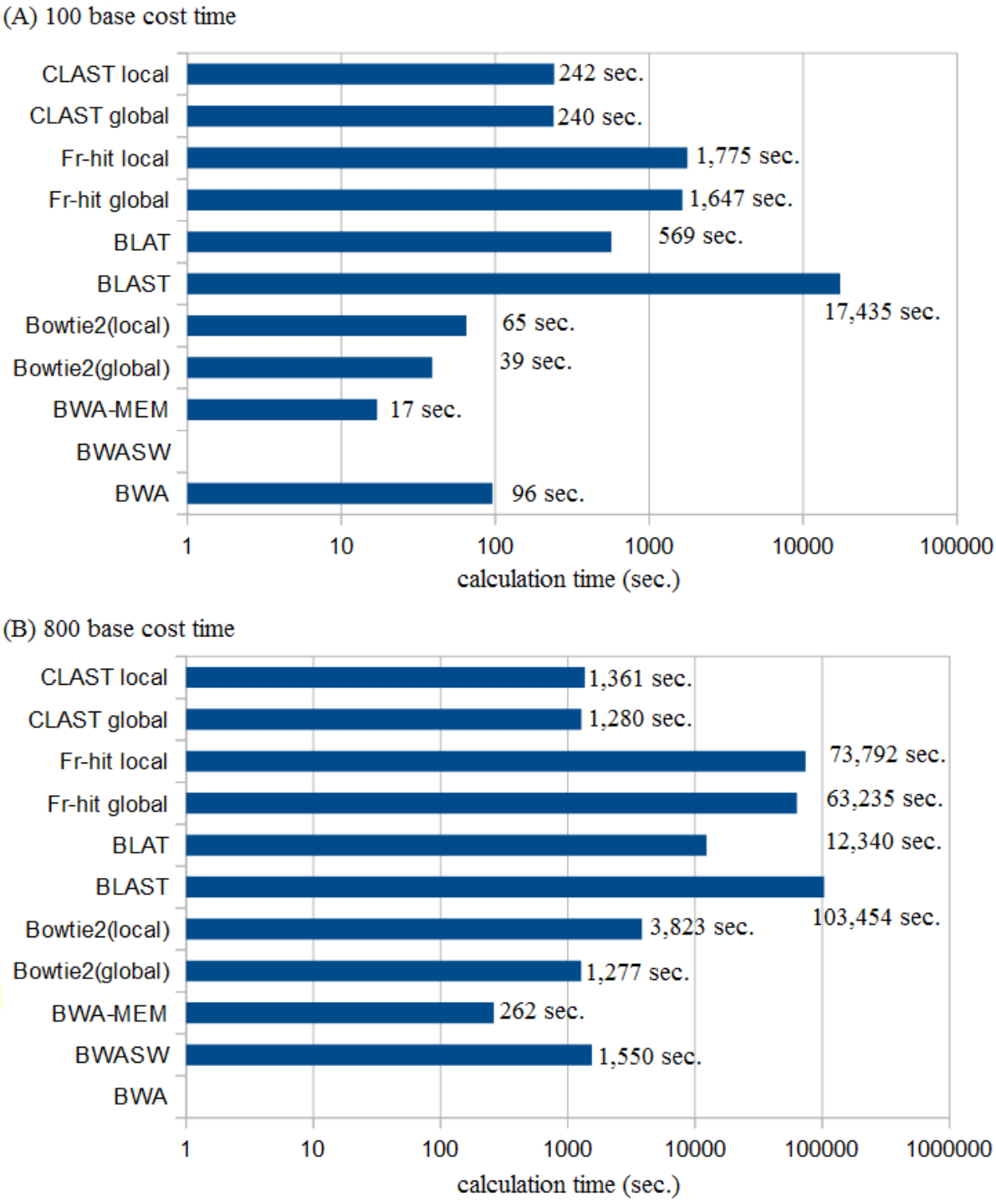
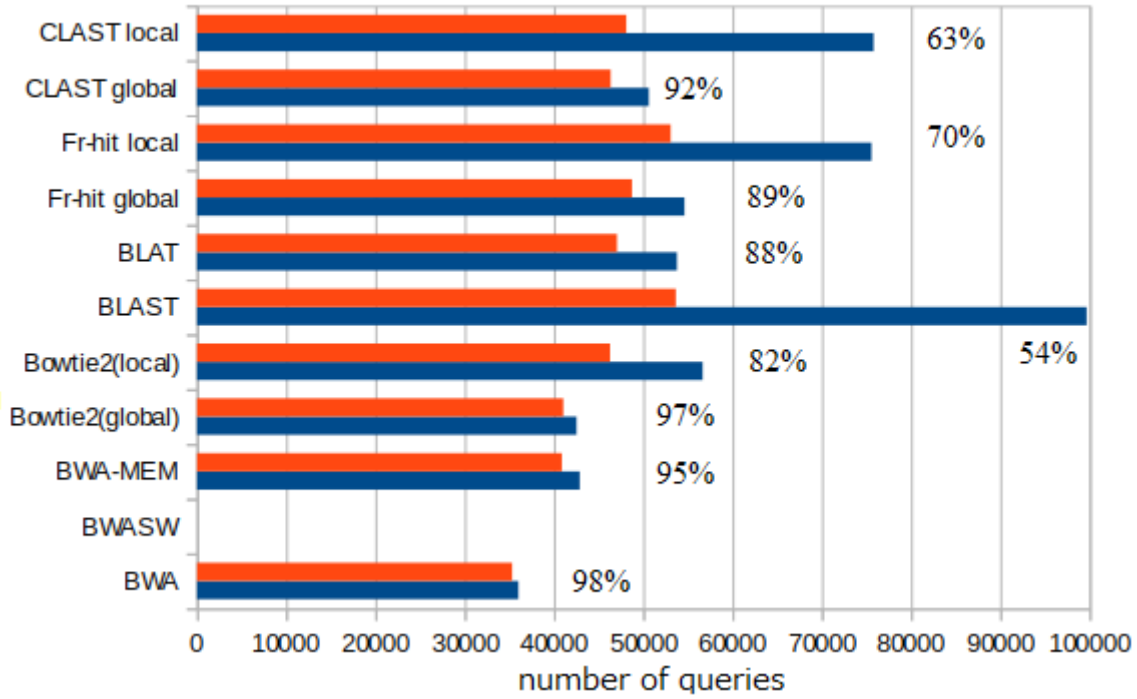


図 2.7 疑似メタゲノム解析試験で各ツールが2,314本のデータベース配列と10万本のクエリ配列とのあいだで配列相同性検索を行うのに要した時間

横軸が計算時間 (秒、対数スケール)。(A)100塩基試験の結果。(B)800塩基試験の結果。

(A) 100 base best non-self hits



(B) 800 base best non-self hits

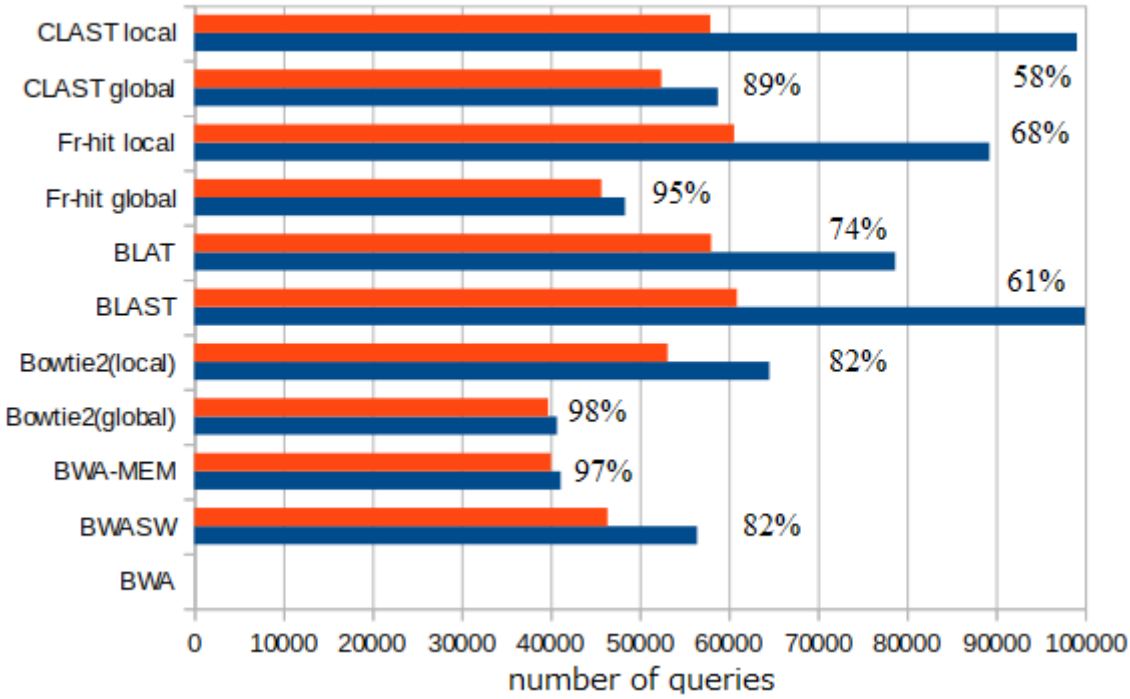
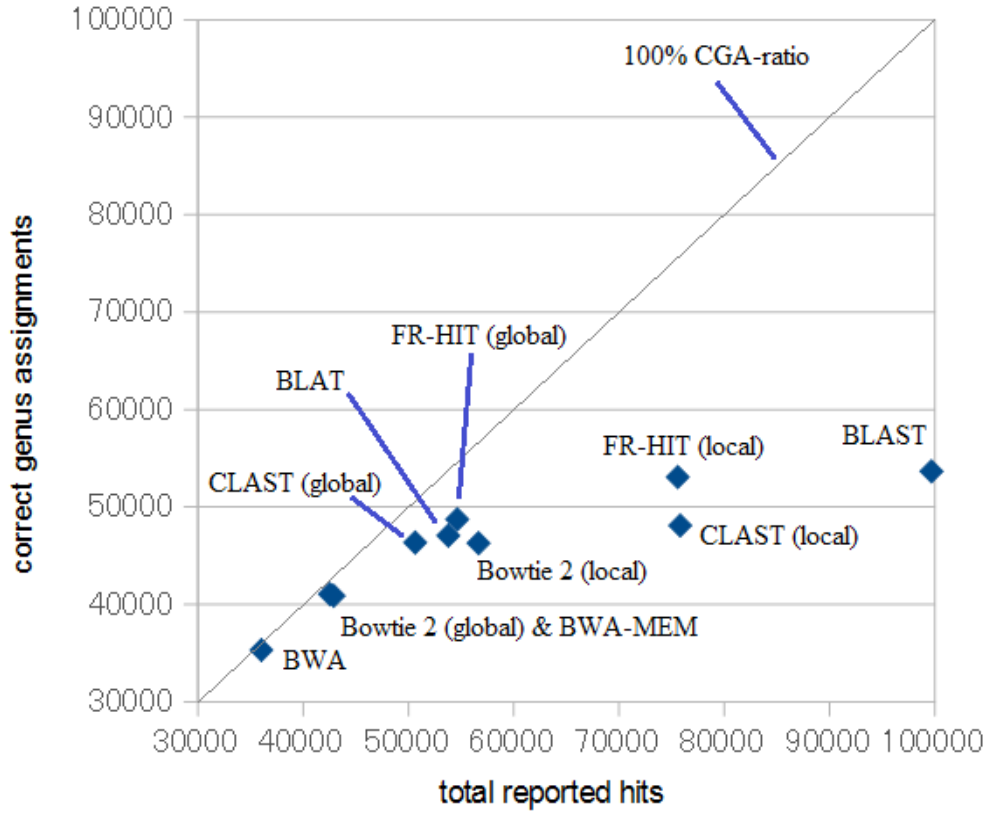


図 2.8 疑似メタゲノム解析試験の結果

青:データベース内に少なくとも1つの類似したシークエンスを持つクエリ配列の数(ヒット数)。赤:正確な系統推定 (correct genus assignment)を伴うクエリ配列の数。百分率はCGA-ratio (”correct genus assignmentの数”/”ヒット数”×100)である。横軸はクエリの数を表す。(A)100塩基試験の結果。(B)800塩基試験の結果。

(A) 100 base



(B) 800 base

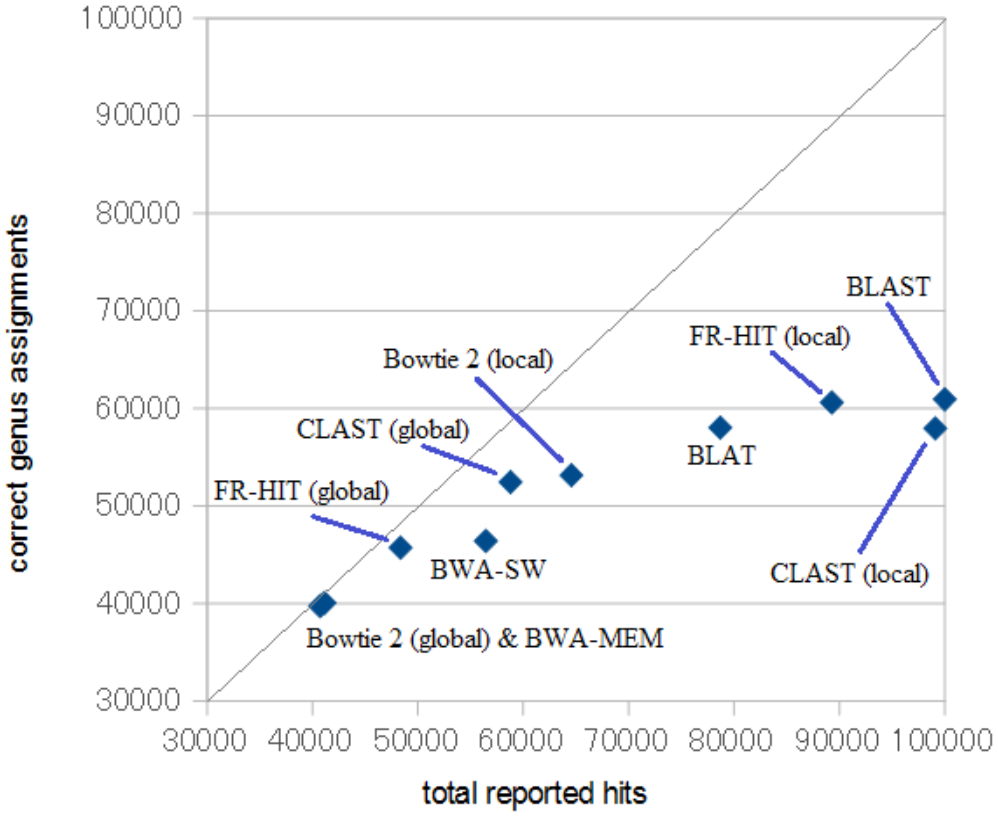
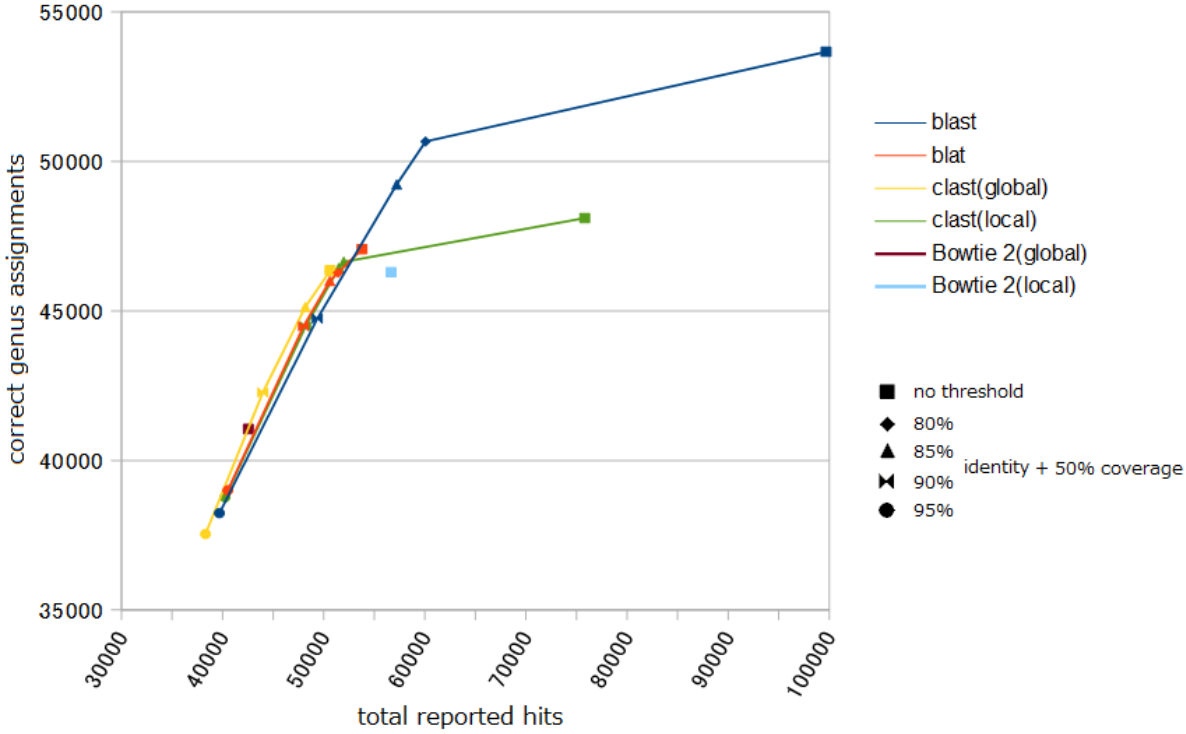


図 2.9 疑似メタゲノム解析の100塩基試験と800塩基試験の両方のツールごとの感度 (ヒット数)と特異性 (correct genus assignmentの数)の関係

各点は、BLAST、BLAT、CLAST (グローバルモードとローカルモードの両方)、FR-HIT (グローバルモードとローカルモードの両方)、BWA、BWA-SW、Bowtie 2 (グローバルモードとローカルモードの両方)の疑似メタゲノム解析での結果を表す。 各グラフの灰色の斜線は100％のCGA-ratioを表す。いかなる点も灰色の線より上にあることはできない。横軸はヒット数を表し、縦軸はcorrect genus assignmentの数を表す。(A)100塩基試験の結果。(B)800塩基試験の結果。

(A) 100 base



(B) 800 base

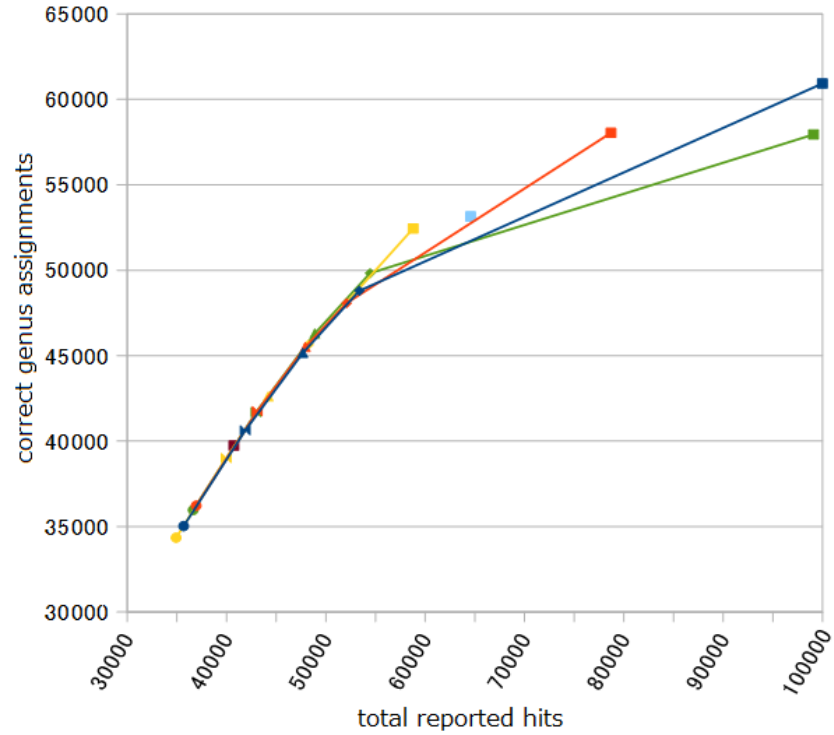


図 2.10 疑似メタゲノム解析の100塩基試験と800塩基試験の両方でアイデンティティ閾値とカバレッジ閾値を変更して、BLAST、BLAT、およびCLASTそれぞれの感度と特異性との間の関係を調べた結果

各折れ線は、BLAST、BLAT、およびCLAST (グローバルモードとローカルモードの両方)の疑似メタゲノム解析の結果をいくつかの閾値ごとに表したものである。各折れ線は5つの点で構成され、5つの異なる閾値を用いた疑似メタゲノム解析の結果を示している。1つの点は、アイデンティティ閾値とカバレッジ閾値でフィルタリングされなかった結果 (図2.9の点と同じ)であり、その他はアイデンティティ閾値とカバレッジ閾値でフィルタリングされた結果に基づく。アイデンティティ閾値は95%、90%、85%、80%であった。カバレッジ閾値は50%に統一された。全ての折れ線において、高いアイデンティティ閾値は、少ないヒット数と高いCGA-ratioを表す。Bowtie 2の結果(グローバルモードとローカルモードの両方)のアイデンティティ閾値とカバレッジ閾値でフィルタリングされなかった点 (図2.9の点と同じ)も、他のツールの折れ線と比較できるようにプロットされている。横軸はヒット数を表し、縦軸はcorrect genus assignmentの数を表す。(A)100塩基試験の結果。(B)800塩基試験の結果。

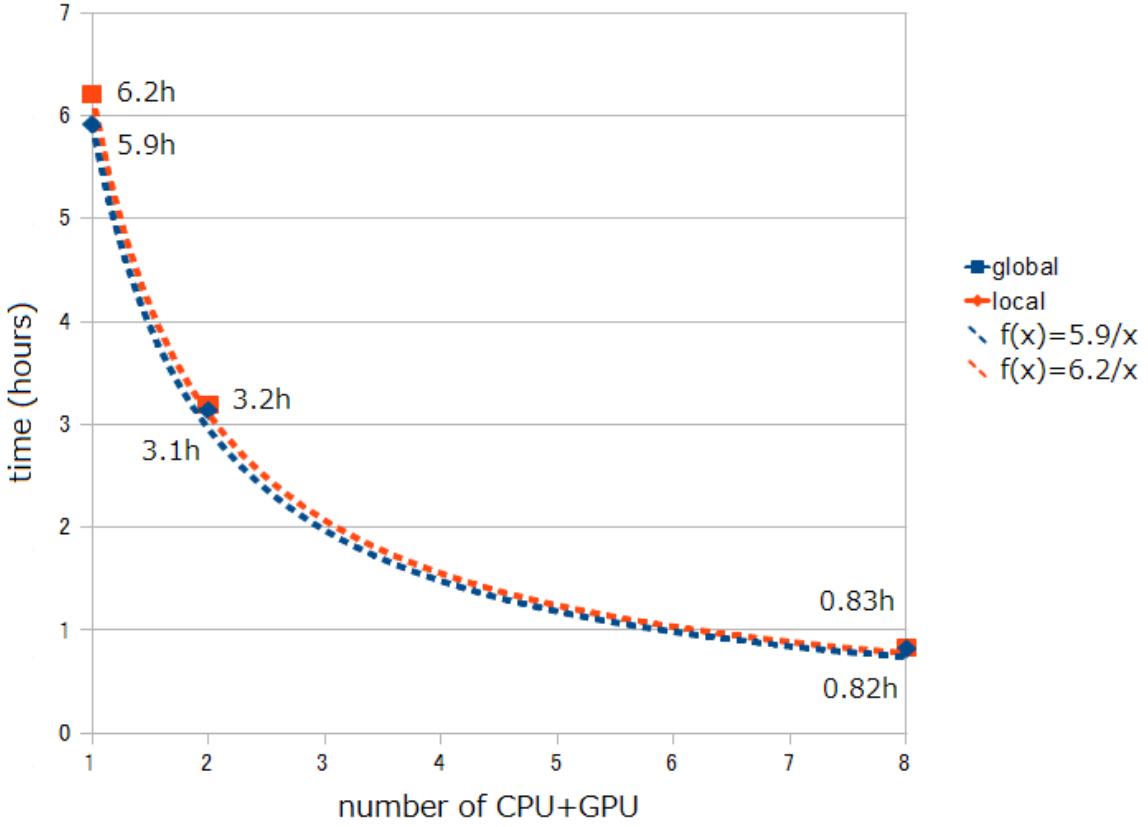


図 2.11 実際のメタゲノム配列をCLASTで計算するのに要した時間を使用するGPUの数ごとにプロットした結果

点線はその点に対して反比例グラフをフィッティングしたもの。

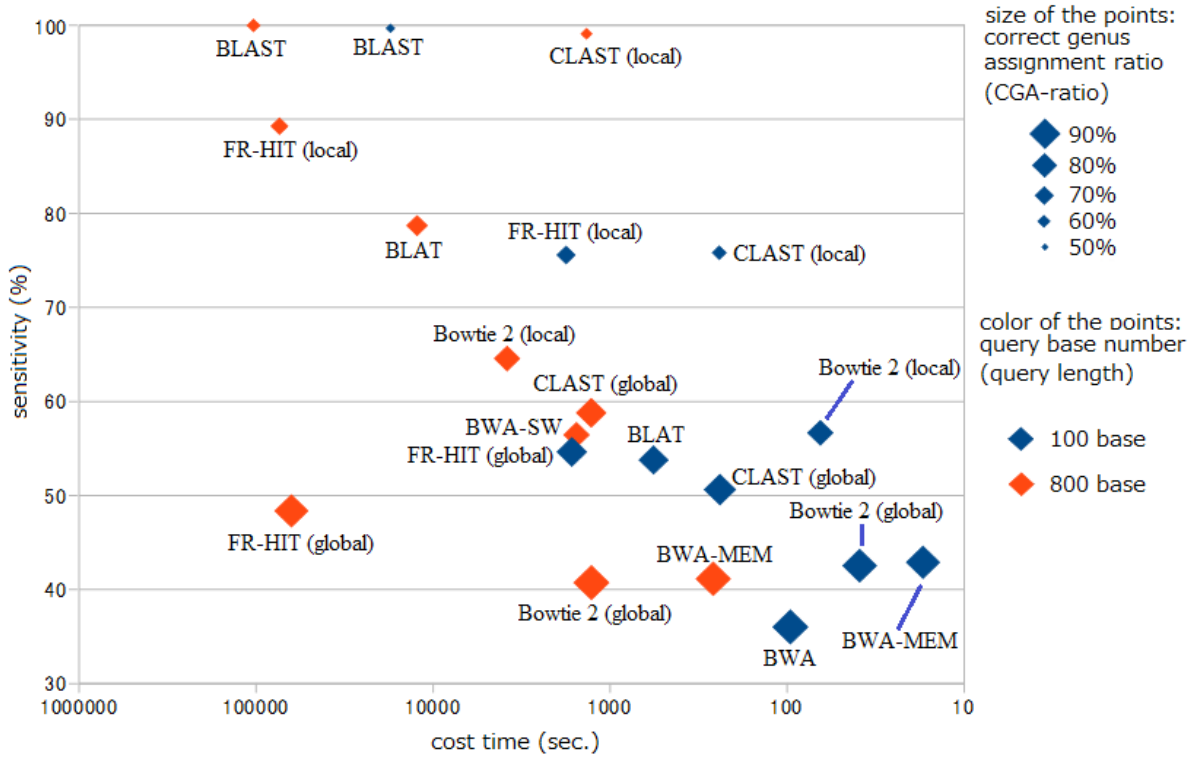


図 2.12 各ツールの速度と感度から作成した散布図

表 2.1 CLASTの出力形式

項目番号	value
1	クエリ配列のFASTA形式におけるラベル
2	クエリ配列のどこからデータベース配列と相同な部分が始まるか
3	クエリ配列上でのデータベース配列との相同な部分の長さ
4	クエリ配列のストランド(+/-)
5	データベース配列のFASTA形式におけるラベル
6	データベース配列のどこからクエリ配列と相同な部分が始まるか
7	データベース配列上でのクエリ配列との相同な部分の長さ
8	アイデンティティ (一致した塩基の数 / 項目番号3の値 * 100%)
9	ビットスコア
10	E-value

表 2.2 CLASTのコマンドラインパラメータ

入力形式	説明	デフォルト値	値域
-t [文字列] [文字列] ...	必須。データベース配列として使用するFASTA形式ファイル(1つ以上)を指定。	-	-
-q [文字列] [文字列] ...	必須。クエリ配列として使用するFASTA形式ファイル (1つ以上)を指定。	-	-
-o [文字列]	必須。出力先ファイルを指定。	-	-
-tRAM [double型]	CPU側のRAMに送るデータベース配列の量 (M Base単位)。	64	-
-qRAM [double型]	CPU側のRAMに送るクエリ配列の量 (M Base単位)。	64	-
-tVRAM [double型]	GPU側のVRAMに送るデータベース配列の量 (M Base単位)。	64	-
-qVRAM [double型]	GPU側のVRAMに送るクエリ配列の量 (M Base単位)。	2	-
-lMer [int型]	K-mer長。	15	31以下
-stride [int型]	K-merストライド長。	5	-
-repeat [int型]	K-merの重複数の閾値。	20	-
-width [int型]	シード間の許容距離。	100	-
-gap [int型]	アライメント幅。	8	16以下
-numOut [int型]	クエリごとの出力結果上限数(-1:無制限)。	-1	-
-local [int型]	アライメントのタイプ。 0以外の値でローカル、0でグローバル。	0	-
-device [int型]	使用するGPUのデバイスID。	0	-
-sleep [int型]	計算の合間のスリープ時間(秒単位)。	0	-
-cutOff [double型]	結果からカットするE-value の閾値(-1:カットなし)。	10	-

表 2.3 性能評価のために行った試験において (G-BLASTNを除く)各ツールに渡したオプション

ツール名(+モード)	オプション
BWA	samse -n 100
BWASW	
BWA-MEM	-a
bowtie2 global	-f -k 10
bowtie2 local	-f -k 10 --local
BLAST	-m 8
BLAT	-out=blast8
FR-HIT(local)	
FR-HIT(global)	-g 1
CLAST(global)	
CLAST(local)	-local 1

表 2.4 疑似メタゲノム解析で用いられたツールの特徴

ツール名	疑似メタゲノム解析におけるメモリ使用量 (*)	ツールが扱えるデータベース配列の総量	アライメントのアルゴリズム (グローバルかローカルか)	疑似メタゲノム解析のために、事前のデータベース配列のインデックス化で要した時間 (秒) (*)
CLAST	1.8 GB	制限なし	選択可能	-
BLAST	1.1 GB	制限なし	ローカルのみ	89.3
BLAT	2.5 GB+2.5 GB +0.86 GB	4 GB	ローカルのみ	-
FR-HIT	13GB	制限なし	選択可能	-
Bowtie 2	2.2 GB+2.2 GB +0.74 GB	4 GB	選択可能	5,731+4,801+970
BWA バージョン 0.5.9	BWA: 1.4 GB+1.4 GB +0.50 GB, BWA-SW: 2.3 GB+2.3 GB +0.77 GB	4 GB	BWAではグローバルのみ BWA-SWではローカルのみ	1,613+1,810.7 +606.5
BWA バージョン 0.7.15	BWA-MEM: 7.4 GB	制限なし	BWAではグローバルのみ BWA-SWおよびBWA-MEMではローカルのみ	7,020

(*) 配列データベースを分割して計算を行った場合は、数値を加算形式(”A+B+C+...”)で示してある。

第3章　超高速かつ高精度な配列クラスタリングツールSaturn

3.1　緒言

環境中の微生物群集の系統組成を推定する際には、配列相同性検索が主に行われる。通常、メタ16S解析で行われる配列相同性検索は、系統情報が既知の16S rRNA遺伝子配列からなる配列データベースに対し、環境中の微生物叢に由来する16S rRNA遺伝子配列をクエリ配列群としたものである。

配列相同性検索が抱える問題点の一つが、配列相同性検索の計算量が爆発的に増加しつつあるというものである。配列相同性検索の計算量は、配列データベースとクエリ配列群の両方の分量に比例する。だが、配列データベースの分量もクエリ配列の分量も増えてきている。配列データベースの分量が増えてきているのはこれまでの研究結果が蓄積されてきたからであり、クエリ配列群の分量が増えているのはシークエンサーの性能が向上して配列を読み取るコストが劇的に減少したからである。

配列相同性検索の計算量は多くの解析で問題となってきたため、高速に配列相同性検索を行うために多くのツールが開発されてきた。配列相同性検索ツールとして広くバイオインフォマティクスで使われているのがBLAST (Altshul *et al*. 1990)であり、それをさらに高速化するべく開発されたのがBLAT (Kent 2002)である。また、検出できる配列の相同性を限定することで、さらなる高速化を実現したのがBowtie 2 (Langmead 2012)をはじめとした配列リシークエンスデータ用マッピングツールである。第2章で扱った、私たちが開発したツールCLAST (Yano *et al*. 2014)は、メタゲノム解析のために開発されたGPUを用いる超高速かつ高感度な配列相同性検索ツールであった。

配列相同性検索を高速化する手段の一つに、クエリ配列群のうち相互に高い相同性が認められる配列を配列クラスタリングによってまとめて、各クラスタを代表する配列(クラスタ代表配列)のみをクエリ配列群として配列相同性検索を行うというものがある (Li *et al*. 2012)。特にメタ16S解析の配列クラスタリングでは、相互に97%以上の相同性があるか否かを基準にして同じ種に属すると考えられる16S rRNA遺伝子配列を一つのクラスタにまとめることが多い。同じ種に属する16S rRNA遺伝子配列は、一般に相互に97%以上の相同性が認められ

るからである。このような配列クラスタリングは、特に数百サンプル以上にのぼるような多量のメタ16S配列を解析する場合では莫大な計算時間の削減に繋がる。

メタ16S解析に使うことができる配列クラスタリングツールは多く開発されてきたが、これまでの配列クラスタリングツールでは速度と精度は相反する傾向があった。しかし私たちは、高い精度を維持したまま超高速な配列クラスタリングを行えるツールSaturnを開発したので、本章でその設計を説明し、性能を検討する。

3.2 材料と方法

ここではSaturnのアルゴリズムとSaturnの性能を評価するために行った試験の方法について記述する。Saturnのアルゴリズムについての記述では、第一にSaturnのアルゴリズムを構成する要素の定義を行い、第二に実装の要素の説明を行い、第三に出力するデータについて説明を行う。

3.2.1 アルゴリズムを構成する要素の定義

アルゴリズムを構成する要素の定義のため、第一に「用語の定義」を行い、第二に「データ構造の定義」を行い、第三に「アルゴリズムの各段階の定義」を行う。

定義する用語は「配列同士の距離」と「配列同士のエラー率」である。配列同士の距離は、配列同士の編集距離とする。配列同士の編集距離は、グローバルアライメントで計算する(ただし、アラインメントの開始位置は後述のk-mer一致の両端である)。グローバルアライメントは系統推定に有用である (Yano *et al*. 2014)。16S rRNA遺伝子配列のクラスタリングは系統情報の集約が目的の場合が多いため、グローバルアライメントを用いた。配列同士のエラー率は、配列同士の距離を配列同士がアライメントを形成した長さで除算したものとする。二本の配列S_1とS_2のエラー率を$Fdiff(S_1,S_2)$と表記する。

定義するデータ構造は「クラスタ・マネージャ」「クラスタ」「配列」である。データ構造は3段階構造になっている図3.1)。クラスタ・マネージャは一つだけ存在する。クラスタ・マネージャ

をMと表記する。MはX個 (Xは自然数)のクラスタを持つ。クラスタリングの過程でXは減少していく。Mが持つクラスタを

$$\boldsymbol{C}=\{C(1)...C(x)...C(X)\}$$

と表記する。C(x)はY(x)個(Y(x)は自然数)の配列を持つ。C(x)が持つ配列を

$$\boldsymbol{S(x)}=\{S(x,1)...S(x,y)...S(x,Y(x))\}$$

と表記する。C(x)は重心配列を持つ。C(x)の重心配列をSc(x)と表記する。Sc(x)は***S(x)***から計算された仮想的な配列である。全てのSc(x)を

$$\boldsymbol{Sc}=\{Sc(1)...Sc(x)...Sc(X)\}$$

と表記する。C(x)はクラスタ代表配列を持つ。C(x)のクラスタ代表配列をSo(x)と表記する。So(x)は***S(x)***のうち最もSc(x)とのエラー率が小さい配列である。代表配列と仮想的な重心配列を別個に持つのはSaturnの設計上の特色の一つである。配列はFASTA形式の塩基配列とする。配列の本数はN本 (Nは自然数定数)。M内では常に

$$Y(1)+Y(2)+...+Y(X)=N$$

が成り立つ。

アルゴリズムの各段階の定義を行う。アルゴリズムは、「準備」「クラスタリング」「出力準備」の三つの段階からなり、この順で実行される (図3.2)。以下で概要をまず説明する。

準備では、第一にFASTAファイルから配列を読み込み、第二に読み込んだ配列をソートし、第三に同じ長さかつ完全に一致する配列 (重複配列)を一まとめにして、第四にまとまった配列ごとにクラスタを形成し、第五にクラスタマネージャを作り各クラスタをその中に入れる。ソートでは長さ順の優先度と配列の辞書順の優先度を用い、前者の方が後者よりも優先度が高い。クラスタリングでは、***C***のうち「結合」できるもの同士をまとめていく。結合とは二つのクラスタ$C(x_1)$と$C(x_2)$をまとめて新しいC(new)にすることである(図3.3)。C(new)が持つ配列を***S(new)***と表記する。つまり、この***S(new)***について、

$$\boldsymbol{S(new)}=\{S(x_1,1)...S(x_1,Y(x_1)),S(x_2,1)...S(x2,Y(x_2))\}$$

が成り立つ。C(new)が持つ重心配列をSc(new)と表記する。出力準備では、Saturnは第一に***C***からファイルに書き出せる形式の出力を作り、第二に出力を見やすいように整えてファイルに書き出す。

3.2.2 配列クラスタリングのアルゴリズム

実装の要素の説明として第一に、クラスタリングの実装の説明を行う (図3.4)。クラスタリングでは、第一に「k-merデータベースの作成」を行い (図3.4 A1~3.4 A2)、第二に「クエリ・クラスタへのクラスタの割り当て」を行う (図3.4 B1~3.4 B4)。

K-merデータベースの作成では、***Sc***からk-merを読みとる。K-merデータベース作成時のk-merの長さ (k-mer長)はデフォルトで32であり (コマンドラインパラメータで変更可能)、k-merを読み取る始点の間隔はk-mer長に等しい。Saturnのデフォルトでのk-mer長を32塩基にしたのは、配列長が長くなるかあるいは配列間の相同性が高くなると、最低でも一箇所のk-mer一致が見つかる可能性も高くなるため、Saturnの用途では問題がないと判断したためである。読み取ったk-merがどのクラスタの重心配列のどの位置由来のものなのかをデータベースとしてまとめる。このデータベースをDと表記する。

クエリ・クラスタへのクラスタの割り当てでは、***C***は1個の「クエリ・クラスタ」とX-1個の「ターゲット・クラスタ」に分けられる。クエリ・クラスタをCqと表記する。Cqの重心配列をSc(x=Cq)と表記する。ターゲット・クラスタを

$$\boldsymbol{Ct}=\{Ct(1)...Ct(z)...Ct(X-1)\}$$

と表記する。Ct(z)の重心配列をSc(x=Ct(z))と表記する。クエリ・クラスタへのクラスタの割り当てでは、第一に「クエリ・クラスタの選定」を行い(図3.4 B1)、第二に「Sc(x=Cq)からのk-mer読み取り」を行い(図3.4 B2)、第三に「Dとのk-mer一致数のクラスタ別カウント」を行い(図3.4 B3)、第四に「k-mer一致数が多い順の結合試行」を行う(図3.4 B4)。クエリ・クラスタの選定では、***C***をソートして、ソートされた順番にCqにしていく。ソートはY(x)が大きい順に行うが、もしY(x)が同じ場合はSc(x)の長さ順に行う。すでにソートされて***C***から削除されたクラスタはCqにならない。Y(x)が大きい順にソートを行ってCqを選んでいくというのは、重複配列が多い順にCqとしていくということを意味している。重複配列が多い順にCqに選ぶのは、配列にシークエンシングエラーがランダムに入っているならば、重複配列が多い配列がクラスタリング後のクラスタ中心になると推定できるためである。デフォルトでこのように重複配列が多い順にCqを選んでクラスタリングを行うというのは、Saturnのアルゴリズム上の特色の一つであ

る。Sc(x=Cq)からのk-mer読み取りでは、Sc(x=Cq)からk-merを読み取る。この時の間隔は1である。Dとのk-mer一致数のクラスタ別カウントでは、Sc(x=Cq)のk-merからDを参照し、Sc(x=Ct(z))それぞれとのk-mer一致数をカウントする。K-mer一致数が多い順のjoin試行では、k-mer一致数が多い順にCt(z)をCqに結合できないか試みていく。Ct(z)とCqが結合可能だった場合、Ct(z)をCqに結合し、結合されたCt(z)の情報はDから削除する (ここでXが1減少する)。結合が一定の回数以上連続して不可能だった場合、そのCqに結合するCt(x)を探すのを終了する。そして、全てのC(x)がCt(x)として他のCqに結合されるか、Cqとして使われるかしたら、クラスタリング終了とする。

実装の要素の説明として第二に、結合の実装について説明を行う(図3.5)。結合では、第一に「重心配列の距離の比較」を行い、第二に「新しいクラスタC(new)の作成」を行い、第三に「C(new)のチェック」を行う。結合を試みるクラスタを以下では$C(x_1)$と$C(x_2)$とする($Y(x_1)>=Y(x_2)$とする)。

重心配列の距離の比較では、$Fdiff(Sc(x_1),S(x_2))$が指定されたエラー率の二倍より大きい場合は結合せず、$Fdiff(Sc(x_1),S(x_2))$が指定されたエラー率の二倍より小さい場合に結合を試みる。

新しいクラスタC(new)の作成では、C(new)を$C(x_1)$と$C(x_2)$から構築する。このとき、Sc(new)は$Sc(x_1)$をベースに作る。このとき、$Sc(x_1)$の始端と末端までは変更せず、$Sc(x_2)$の$Sc(x_1)$の始端と末端から先にはみ出た部分をSc(x1)に継ぎ足す。

C(new)のチェックでは、***S(new)***の全ての配列がSc(new)から指定されたエラー率以下の場合、$C(x_1)$と$C(x_2)$を***C***から削除して、C(new)を***C***に登録する。C(new)が***C***に登録された場合、結合が成功したと見なす。結合が成功したとき、$C(x_1)$がCqである場合は$C(x_2)$が***C***から削除されたと見なし、$C(x_2)$がCqである場合は$C(x_1)$が***C***から削除されたと見なす。

3.2.3 出力データの形式

Saturnの出力形式は、UCLUSTの出力形式をベースにしている (表3.1)。出力は1行1データである。各行のデータはタブ区切りで列に分かれている。各行の1番目の列はラベルであ

る。Saturnは「C」「S」「H」以外のラベルを使用しない。クラスタ代表配列に関するデータではSを、クラスタ代表配列にマップされた配列 (クエリ配列)に関するデータではHを、クラスタの情報に関するデータではCが使われる。

Saturnは、クラスタ代表配列と重心配列との間のアライメントと、クエリ配列と重心配列との間のアライメントから、クエリ配列とクラスタ代表配列のアライメントを計算する (図3.6)、(図3.7)。SaturnはクラスタC(x)の半径として***S(x)***とSc(x)とのエラー率を使っているため、So(x)と***S(x)***とのエラー率は、指定した許容エラー率より若干高くなることがある。

3.2.4 性能評価のために行った3つの試験

Saturnの性能を評価するために第一に行った試験が「速度試験」である。速度試験の目的は、Saturnの速度を調べることである。速度試験の方法としては、用意した複数本の配列を各ツールが処理するのに要する時間を比較した。この他、各ツールが出した結果に含まれるクラスタの数も記録した。

Saturnの性能を評価するために第二に行った試験が「精度試験」である。精度試験の目的は、Saturnの結果の精度を検討することである。速度試験の方法として、微生物叢に由来する複数本の配列を各ツールが処理した結果とESPRIT-TREE (Cai and Sun 2011)が処理した結果を、Normalized Mutual Information (NMI)を尺度として比較した。ESPRIT-TREEが処理した結果との比較を行ったのは、ESPRIT-TREEは群平均法による階層的クラスタリングを行うことで、安定して種に応じた正確なクラスタリングを行うことができると報告されているからである (Sun *et al*. 2012)。人工的に生成した配列を用いることによる真のクラスタリング結果が分かっている試験を行わなかったのは、微生物叢に由来する配列を正しく人工的に生成できるモデルの構築が困難であったためである。NMIとは2つの確率変数XとYとの間で共有される相互情報量を正規化したものである。例えば、同一のデータの集合を二つの手法でクラスタリングした結果AとBに関して、AからBが完全に予想できる (即ち、AとBが等しい)場合はAとBのNMIは1となり、Bを予想するのにAが全く役に立たない (即ち、AとBが確率論的に独立とみなせる)場合はAとBのNMIは0となる。したがって、NMIが高いか低いかによって、クラスタリングの結果の類似度が検討できる (Sun *et al*. 2012)。ここから、精度試

験においてあるツールのNMIが高いならば、そのツールの結果はESPRIT-TREEの結果に類似した種に応じた正確なものであるということが言える。

Saturnの性能を評価するために第三に行った試験が「系統樹試験」である。系統樹試験の目的はSaturnの結果の信頼度を検討することである。系統樹試験の方法として、用意した配列を各ツールでクラスタリングし、できたクラスタに含まれる配列が単系統になる配列のみから構成されているかを比較した。クラスタが単系統になる配列のみから構成されるとは、クラスタに含まれる全ての配列が系統樹の分岐をまたいでいないということである (図3.8)。以下、そのような単系統になる配列のみから構成されるクラスタを、単系統クラスタと呼び、全てのクラスタにおける単系統クラスタの割合を単系統クラスタ率と呼ぶ。単系統クラスタ率が高いクラスタリング結果を出力するということは、そのクラスタリングツールの結果は系統樹との整合性があるということを示す。

各試験で使ったツールとそのツールに与えたコマンドラインパラメータをまとめたのが表3.2である。どの試験でも与えたコマンドラインパラメータは同じであった。速度試験ではSaturnの他にUCLUST (Edgar 2010)、CD-HIT (Li and Godzik 2006)、DNACLUST (Gihosi *et al*. 2011)、Swarm2 (Mahé *et al*. 2015)、ESPRIT-TREEを使用した。Swarm2は前処理にVSEARCHを使用した。精度試験では、Saturnの他に、UCLUST、CD-HIT、DNACLUST、Swarm2、ESPRIT-TREE、mothur (Schloss *et al*. 2009)を使用した。mothurは前処理にMAFFT (Katoh *et al*. 2002)とPHYLIP (Felsenstein 2005)を使用した。系統樹試験でも、Saturnの他にUCLUST、CD-HIT、DNACLUST、Swarm2、ESPRIT-TREE、mothurを使用した。系統樹の作成にはMAFFTとRAxML version 8.0.3 (Stamatakis 2014)を使用した。系統樹作成に使用したMAFFTのバージョンはv7.221であり、mothurの前処理に使用したMAFFTのバージョンと異なった。なお、速度に関わる試験は全てIntel Xeon CPU E7-4870 10コア 2.40 GHz CPU 4個、1.8TB RAMを搭載したサーバで行った。

3.2.5 速度試験と精度試験で用いたデータ

速度試験と精度試験には3種類のメタ16S配列データを用意した。3種類のデータは、微生物叢の系統組成が多様性の高いもの、多様性の中程度であるもの、多様性の低いもの

である。本論文では、このヒト腸内微生物叢の多様性を中程度の多様性とした。様々な多様性のデータで試験を行ったのは、扱うデータが持つ多様性によるツールの挙動の差を調べるためであった。なお、3種類の配列データは各論文と同じ手法でクオリティフィルタリング済みであった。

一つ目のメタ16S配列データは、マウス腸内微生物叢に由来する16S rRNA遺伝子のV3-V4領域をPCR増幅して、Illumina MiSeqで読み取ったpaired-end配列のpairをoverlapを元にマージして一本の配列として使用したもののうちの25サンプル (DDBJ DRX ID: DRX055749-DRX055753, DRX055755-DRX055774; 1,838,038本)であった (Okai *et al.* 2016)。Paired-end配列のpairをoverlapを元にマージしたため、配列はV4領域の全体を読めていた。このデータは複数サンプル由来のマウス腸内の配列を一つにしたものである。一般的に、マウス腸内微生物叢の系統組成の多様性はヒト成人腸内微生物叢の多様性と同程度である (Ley *et al*. 2006)ことから、本論文の基準では中程度であると言える。このデータをESPRIT-TREEでクラスタリングした結果についてShannonインデックス (Spellerberg and Fedor 2003)を計算することで得られた多様性の指数は4.586 (有効数字4桁)であった。

二つ目のメタ16S配列データは、土壌微生物叢に由来する16S rRNA遺伝子のV3-V4領域をPCR増幅して、Roche 454 GS FLX Titaniumで読み取ったもののうちの1サンプル (DDBJ DRX ID: DRX001020; 27,980本)であった (Kato et al. 2015)。土壌微生物叢の系統組成の多様性は、先行研究において土壌微生物叢のメタゲノム配列はヒト成人腸内微生物叢のメタゲノム配列よりもの1/50未満の割合でしかアセンブルが可能でなかった (Kurokawa *et al.* 2007)ことから、本論文の基準では高いと言える。このデータをESPRIT-TREEでクラスタリングした結果についてShannonインデックスを計算することで得られた多様性の指数は6.387 (有効数字4桁)であった。

三つ目のメタ16S配列データは、ヒト乳児腸内微生物叢に由来する16S rRNA遺伝子のV1-V2領域をPCR増幅して、Roche 454 GS Juniorを用いて読み取ったもののうちの1サンプル (DDBJ DRX ID: DRX046532; 3,119本)であった (Matsuki *et al.* 2016)。配列の本数が土壌やマウス腸内と比べて少なく、Matsuki *et al*. 2016の研究では他のヒト乳児の腸内微生物叢のデータも存在したため複数サンプルを1つにまとめて解析することも可能であったが、そ

れをしなかった理由としては、ヒト乳児腸内微生物叢はヒト乳児間で個体差が大きいことが挙げられる (Kurokawa *et al.* 2007)。個体差が大きい微生物叢においては、複数サンプルの配列をまとめることによって系統情報の多様性が顕著に上がってしまうことが懸念されたからである。ヒト乳児腸内微生物叢の系統組成の多様性は、先行研究においてヒト乳児腸内微生物叢のメタゲノム配列はヒト成人腸内微生物叢メタゲノム配列よりも多くの割合でアセンブルが可能であった (Kurokawa *et al.* 2007)ことから、本論文の基準では低いと言える。このデータをESPRIT-TREEでクラスタリングした結果についてShannonインデックス を計算することで得られた多様性の指数は1.017 (有効数字4桁)であった。

3.2.6 精度試験の方法

精度試験では、ツールで解析する配列の本数を調整した。用意した配列が5,000本以上の場合、配列を5,000本ずつ10回サンプリングしてESPRIT-TREEの結果と比較したNMIの平均と不偏標準偏差を計算した。一度にクラスタリングする配列の本数を5,000本に限定したのは、クラスタリングする配列の本数が多い場合ではmothurでの計算にかかる時間が長くなるからである。ヒト乳児腸内のデータについては3,119本すべてを1回使用した。本数調整の方法としては、データから配列をランダムで1本選ぶという行為を用意したい本数の回数だけ繰り返した。本数調整の際の注意点としては第一に、同じ配列が複数回現れる場合、ヘッダは別のものを用意するようにした。これは同じヘッダの配列をクラスタリング前にまとめてしまうツールがあるためである。本数調整の際の注意点としては第二に、ヘッダ内の空白文字は"_"で置換した。UCLUSTは空白文字以降を解析時に削除してしまうためである。

3.2.7 系統樹試験の方法

私たちは系統樹試験に使った系統樹を、1 species 1配列の16SrRNA 遺伝子配列から作成した。その配列は、NCBI RefSeqデータベースから完全または精度の高いドラフトゲノム解読済みの株の16S rRNA遺伝子配列を1 species 1配列ランダムに取得し、1200 bp以下または1800 bp以上の配列を除去し、曖昧塩基を含んだ配列を除去することで選ばれた。配

列の総数は1,472本であった。私たちは、この配列MAFFTをdefault parameterで用いてマルチプルアライメントし、その結果に対してRAxMLのGTRGAMMAモデルを使って推定された最尤系統樹を、系統樹試験でReferenceとなる系統樹として使った。

3.3 結果

3.3.1 速度試験の結果

試験の結果総評として第一に、速度試験の結果について総評を行う (表3.3)。マウス腸内微生物叢由来のデータでは、Swarm2が一番速く、Saturn、UCLUST、DNACLUST、CD-HIT、ESPRIT-TREEがこの順に続いた。この中でもSaturnとSwarm2とUCLUSTが一番速いグループを作っていた。ヒト乳児腸内微生物叢由来のデータでは、Saturnが一番速く、DNACLUST、UCLUST、CD-HIT、Swarm2、ESPRIT-TREEがこの順に続いた。ここでは、CD-HITとUCLUSTとSaturnとDNACLUSTの速度は同等であった。土壌微生物叢由来のデータでは、UCLUSTが一番速く、Swarm2、Saturn、CD-HIT、DNACLUST、ESPRIT-TREEがこの順に続いた。ここでもマウス腸内微生物叢由来のデータの場合と同じく、SaturnとSwarm2とUCLUSTが一番速いグループを作っていた。UCLUSTの計算時間をSaturnの計算時間と比較すると、マウス腸内微生物叢由来のデータではUCLUSTはSaturnの1.39倍の計算時間がかかっており、土壌微生物叢由来のデータではUCLUSTはSaturnの0.34倍の計算時間がかかっており、ヒト乳児微生物叢由来のデータでは1.75倍の計算時間がかかっていた。

また、各ツールの結果のクラスタ数をESPRIT-TREEの結果のクラスタ数と比較した。マウス腸内微生物叢由来のデータでは、ESPRIT-TREEの結果のクラスタ数に近い順にSaturn、CD-HIT、DNACLUST、UCLUST、Swarm2となった。このケースでは、全てのツールの結果でクラスタ数はESPRIT-TREEの結果のクラスタ数よりも多かった。土壌微生物叢由来のデータでは、ESPRIT-TREEの結果のクラスタ数に近い順にSaturn、DNACLUST、CD-HIT、UCLUST、Swarm2となった。このケースでは、CD-HITの結

果のクラスタ数がESPRIT-TREEの結果のクラスタ数よりも少ない以外は、全てのツールの結果でクラスタ数はESPRIT-TREEの結果のクラスタ数よりも多かった。ヒト乳児微生物叢由来のデータでは、ESPRIT-TREEの結果のクラスタ数に近い順にCD-HIT、Saturn、DNACLUST、UCLUST、Swarm2となった。このケースでは、全てのツールの結果でクラスタ数はESPRIT-TREEの結果のクラスタ数よりも多かった。全てのケースで、UCLUSTとSwarm2の結果のクラスタ数は、他のツールの結果のクラスタ数よりもESPRIT-TREEの結果のクラスタ数よりも多くなっていた。また全てのケースで、Saturnの結果のクラスタ数はDNACLUST、UCLUST、Swarm2の結果のクラスタ数よりもESPRIT-TREEの結果のクラスタ数に近かった。

3.3.2 精度試験の結果

試験の結果総評として第二に、精度試験の結果について総評を行う (表3.3)。マウス腸内微生物叢由来のデータでは、mothurが一番NMIが高く、Saturn、CD-HIT、DNACLUST、UCLUST、Swarm2がこの順に続いた。このケースでは、SaturnとCD-HITとDNACLUSTの結果が 二番目にESPRIT-TREEの結果に近いグループを作っていた。土壌微生物叢由来のデータでは、mothurが一番NMIが高く、Saturn、DNACLUST、CD-HIT、UCLUST、Swarm2がこの順に続いた。ヒト乳児腸内微生物叢由来のデータでは、Saturnが一番NMIが高く、mothur、CD-HIT、UCLUST、Swarm2、DNACLUSTがこの順に続いた。このケースでは、Saturnとmothurの結果が一番目にESPRIT-TREEの結果に近いグループを作っていた。

3.3.3 系統樹試験の結果

試験の結果総評として第三に、系統樹試験の結果について総評を行う(表3.4)。各ツールを、単系統クラスタ率が高い順に並べると、Swarm2、mothur、ESPRIT-TREE、Saturn、UCLUST、CD-HIT、DNACLUSTとなった。各ツールを、クラスタリングされずに残ったシングルトンの本数が少ない順に並べると、ESPRIT-TREE、CD-HIT、DNACLUST、mothur、

Saturn、UCLUST、Swarm2となった。Swarm2とESPRIT-TREE以外のツールでは、結果に含まれるシングルトンの本数は802本 (全体の54.48%)以上830本 (全体の56.39%)以下であった。Swarm2の結果に含まれるシングルトンの本数は1424本(全体の96.74%)であり、ESPRIT-TREEの結果に含まれるシングルトンの本数は769本(全体の52.24%)であった。

3.4 考察

第一に、Saturnの性能について論じる。論じられる性能は、速度と精度と系統樹との整合性である。

まずSaturnの速度について論じる。速度試験の全てのケースで、SaturnはESPRIT-TREE、CD-HIT、DNACLUSTよりも速かった。Saturnの速度はUCLUSTやSwarm2のそれと競い合うレベルであった。土壌微生物叢のケースで最速だったのはUCLUSTであり、ヒト乳児腸内微生物叢のケースで最速だったのはSaturnであり、マウス腸内微生物叢のケースで最速だったのはSwarm2であった。Swarm2とUCLUSTの高速性は、Srawm2とUCLUSTの結果のクラスタ数が全てのケースで他のツールの結果のクラスタ数よりも多いことから、他のツールよりも実行するクラスタリングの回数が少ないためであると考えられる。Swarm2がヒト乳児微生物叢のケースでSaturnとUCLUSTよりも低速になったのは、配列の本数が少ないためクラスタリング以外でのオーバーヘッドが大きくなったからである可能性が考えられる。また、Saturnの計算時間と比較した結果から、配列の多様性が低いケースではUCLUSTの計算時間は長くなる傾向があり、配列の多様性が高いケースではUCLUSTの計算時間は短くなる傾向があると考えられる。データの多様性が低くなるとデータの多様性が高い場合に比べてUCLUSTが低速になるとすると、その原因はUCLUSTが行っているk-merフィルタリングが多様性が低い場合ではクラスタリングすべき配列を有効に絞り込めていないためであると考えられる。これらの考察から、マウス腸内微生物叢のケースなどのデータサイズが大きくかつデータの多様性が中程度である場合では、Swarm2が最も速く、SaturnとUCLUSTがこの順で続くと考えられる。このマウス腸内微生物叢のケースではSaturnはUCLUSTよりも1.39倍高速であった。

次にSaturnの結果の精度について論じる。ヒト乳児微生物叢のケースでSaturnは一番ESPRIT-TREEに近い結果を出し、マウス腸内微生物叢のケースと土壌微生物叢のケースでSaturnはmothurについで二番目にESPRIT-TREEに近い結果を出した。ここから、他のツールよりも正確であると考えられるESPRIT-TREEに、Saturnは配列の多様性に関わらずCD-HIT、UCLUST、DNACLUST、Swarm2の結果よりも類似したクラスタリング結果を出力することができると考えられる。各ツールの速度試験と精度試験の結果を2次元プロットにしたものが図3.9である。このプロットには速度が計算されなかったmothurの結果はプロットされていない。このプロットにおける各ツールのNMIの値には各試験でのNMIの平均を用いた。このプロットから、どの試験データにおいてもSaturnが速度と精度の両方を高い水準で両立していることが分かる。

そしてSaturnの結果の系統樹との整合性について論じる。系統樹試験での単系統クラスタ率ではSaturnはmothurとESPRIT-TREEに劣った。だがこの結果は、ESPRIT-TREEの結果は精度が高いと考えられるという先行研究と、精度試験でのNMIの値の比較結果が示唆するmothurの方がSaturnよりも精度が高いであるという可能性に合致する。また、mothurとESPRIT-TREEは計算速度が低いことから、精度と速度とのトレードオフが起こっているとも考えられる。ESPRIT-TREEの結果の方がmothurの結果よりも系統樹試験での単系統クラスタ率は0.57%低いが、ESPRIT-TREEの結果に含まれるシングルトンの数がmothurの結果に含まれるシングルトンの数よりも43本 (全体の2.92%)低い。したがって、この系統樹試験の結果は、先行研究のESPRIT-TREEの方がmothurの結果よりも精度が高いという結論 (Sun *et al*. 2012)に矛盾せず、精度試験においてESPRIT-TREEの結果との近さを各ツールの精度の尺度としたことの妥当性を支持していると考えられる。系統樹試験におけるSwarm2の結果の単系統クラスタ率は他のどのツールの結果の単系統クラスタ率よりも高いが、Swarm2の結果ではシングルトンの数が他のツールの結果のシングルトンの数よりも最低でも594本(全体の40.35%)多い。これは、そもそも他のツールとは違い、Swarm2は配列相同性を割合で判定してクラスタリングするツールではなく (Mahé *et al*. 2015)、相互に極めて類似した配列しかクラスタリングしていないからだと考えられる。系統樹試験での単系統クラスタ率は、Saturnの結果の方がDNACLUST、CD-HIT、UCLUSTの結果よりも最大で0.72%高い

が、Saturnの結果のシングルトンの数はCD-HIT、DNACLUSTのそれよりも最大で23本 (全体の1.56%)多かった。したがって以上をまとめると、系統樹試験から測られるSaturnのクラスタリング結果の系統樹との整合性は、UCLUSTのそれよりは良好であり、ESPRIT-TREEとmothurのそれよりは悪く、そしてCD-HITとDNACLUSTのそれと同等であると考えられる。

第二に、精度試験から測られたSaturnの精度が良好である理由について考察を行う。グリーディに配列をクラスタリングしていく場合では、その順序が精度に影響を及ぼすと言われている (http://drive5.com/usearch/manual/uclust_algo.html)。Saturnもグリーディに配列をまとめていることから、Saturnが配列をクラスタリングする順序は、他のグリーディなアルゴリズムを用いているツールが配列をクラスタリングする順序よりも、ESPRIT-TREEの結果によく類似した結果を出すのに適しているのではないかと考えられる。Saturnは重複配列の数が多い順にクラスタの中心と推定してクラスタリングを行っている。ここから、重複配列が多いか否かで真のクラスタの中心を推定することで、グリーディでも高精度なクラスタリングができるのではないかと推測できる。実際、配列の長さに違いがない場合では、重複配列の本数でクラスタリングを行う方がクラスタリングの精度が向上するという主張はこれまで既にされてきている (Quince *et al.* 2009, Huse *et al.* 2010)。また現在よく使われているIllumina や454やIon Protonといった多くのシークエンサーはほぼ均一な長さの配列を出力するので、そうしたシークエンサーが出力したメタ16S配列をクラスタリングする場合は、クラスタの中心の推定には重複配列の本数を使うのが良いと考えられる。だが、CD-HITとDNACLUSTは配列長でソートした順に配列をクラスタリングしている。これがCD-HITとDNACLUSTの精度試験での結果がSaturnの結果よりも優れない理由であると考えられる。UCLUSTはデフォルトでは配列長でソートした順に配列をクラスタリングするが、より新しいバージョンのUCLUSTでは同じ配列がsizeアノテーション付きでまとめてある場合に配列の数が多い順でクラスタリングすることもできる (http://drive5.com/usearch/manual/abundance_sort.html)。しかしUCLUSTでは、sizeアノテーションをつけるために別途 dereplicationを行う必要性がある (http://www.drive5.com/usearch/manual/cluster_sizes.html, http://www.drive5.com/usearch/manual/cmd_fastx_ uniques.html)。一方でSaturnならば1コマンドで重複配列の数が多い順のクラスタリングができる。

精度試験のヒト乳児腸内微生物叢由来のデータでSaturnの結果がmothurの結果よりも良いにも関わらず、系統樹試験でSaturnの結果がCD-HITとDNACLUSTの結果と大差ないのも、Saturnが配列をまとめる順序を決めるアルゴリズムから説明できる。真のクラスタの中心と周縁で重複配列の本数に差がある場合ではSaturnはクラスタの中心の推定が良好にできる一方で、1 species 1配列といったように重複配列がほとんど存在しないと考えられる場合ではSaturnのクラスタの中心の推定がうまく働かないと考えられるからである。しかし、Saturnは精度試験の土壌微生物叢由来のデータでもmothurに次いで2番目に良好な精度を出していることから、Saturnがクラスタの中心の推定をうまくできないケースは実際のメタ16S解析では稀であると考えられる。

第三に、Saturnの試験していない性能として、メモリ使用量と重心配列を計算するアルゴリズムについて論じる。

まず、Saturnのメモリ使用量は、今日の計算機環境からいえば十分少ないと言える。というのも、Saturnのメモリ使用量は、速度試験ではマウス腸内微生物叢由来のデータをクラスタリングする際に約10GB消費したのが最高値であったからである (data not shown)。メモリの大容量化と安価化が進んできている2016年現在では、個人で20GB以上のメモリを搭載したPCも容易に構築できるので、10GBというメモリ使用量はツールの利便性には影響が少ないと考えられる。

また、Saturnが重心配列を計算する際のアルゴリズムの特徴として、Saturnは仮想的な重心配列を推定し、その重心配列がクラスタリングの過程で移動するようにしたという点を挙げられる。この設計はクラスタリングに伴い計算中のクラスタの中心が移動するのを追跡することで、クラスタリング結果の精度がより向上すると考えたため導入した。この設計により、結合の過程で重心配列の末端を延長できるようになったことも、Saturnのクラスタリング結果の精度が高い原因であるかもしれない。しかし、この設計がSaturnの性能にどれほど寄与したかを判断するのに十分な材料となる試験結果は、速度試験と精度試験と系統樹試験のどの結果には見ることができなかった。

なお開発時には、重心配列を計算するアルゴリズムについて、Saturnはプロファイルを使ったアルゴリズムも試験した。コンセンサス配列を計算するためにはプロファイルが多く使

われてきたからである (Gribskov *et al*. 1987)。しかし、プロファイルを用いた場合のSaturnは精度試験でESPRIT-TREEとのNMIがプロファイルを使わなかった場合よりも低くなり、系統樹試験における単系統クラスタ率もプロファイルを使わなかった場合よりも低くなったが、いずれの場合も結果に含まれるクラスタの数はプロファイルを使わなかった場合よりも少なくなった (data not shown)。プロファイルを用いた場合のSaturnの結果が芳しくなかったのは、プロファイルを用いることでクラスタの中心が移動しやすくなり、最初に推定したクラスタの中心から重心配列がクラスタリングの過程で離れて、複数のクラスタが一つにまとめられてしまった結果だと考えられる。

Saturnは16S rRNA遺伝子配列をクラスタリングするツールであるが、データベースの作成などの16S rRNA遺伝子配列に特有の知見は使っていない。したがって、16S rRNA遺伝子配列の可変領域と同程度の長さ (数百塩基)の配列に関して97%程度の相同性で配列クラスタリングを行う用途であれば、Saturnは16S rRNA遺伝子配列と同様にクラスタリングが可能であると考えられる。また、さらに別の用途として、配列相同性検索における配列データベースをSaturnを用いて圧縮することで、配列相同性検索の計算量をさらに削減することも可能だと考えられる。しかし、配列データベースを圧縮した場合、配列データベースに登録されるデータベース配列の本数が減る以上、配列相同性検索における検出感度が下がる可能性はある。なお、Saturnは配列をマッチしたk-merの両端を始点としたグローバルアラインメントにおけるエラー率を基準にクラスタリングするか否かを決めるため、Saturnによりデータベース配列の本数の削減を期待できるのは、配列データベースが遺伝子配列など数百塩基程度の配列から構成されている場合が主であると考えられる。配列データベースがゲノム配列から構成されている場合には、マッチしたk-merの両端を始点としたグローバルアラインメントにおいてゲノム配列同士のエラー率が低くなる可能性が低くなり、データベース配列の本数の削減は期待しがたいと考えられる。

以上の結果および考察より、Saturnはメタ16S解析において、長さが揃った16S rRNA遺伝子配列の種に応じた高精度なクラスタリングを、超高速に行うことができるツールだと言える。Saturnが超高速なのはグリーディなアルゴリズムによるクラスタリングを行ったためであり、Saturnのクラスタリング結果が高精度であるのは重複配列の数が多い順にクラスタの中心と推定した手法が有効であったためである。また、Saturnのメモリ使用量は現在の計算機

環境ではツールの利便性を損なわない程度であり、しかもSaturnはデフォルトで重複配列の本数によるクラスタの中心の推定を行うため、現在のシークエンサーを用いたメタ16S解析の配列クラスタリングによる高速化を高精度なままより簡便にできる。これは、使用できる計算資源が限られた利用者でも、Saturnを使用することでメタ16S解析がより容易に実行できるようになることを示している。またSaturnを使用することで、数百以上といった多数のサンプル由来のメタ16S配列を超高速かつ高精度にクラスタリングすることも可能になったため、膨大な環境の系統組成をモニタリングすることが現実的なものとなったと言える。

図表

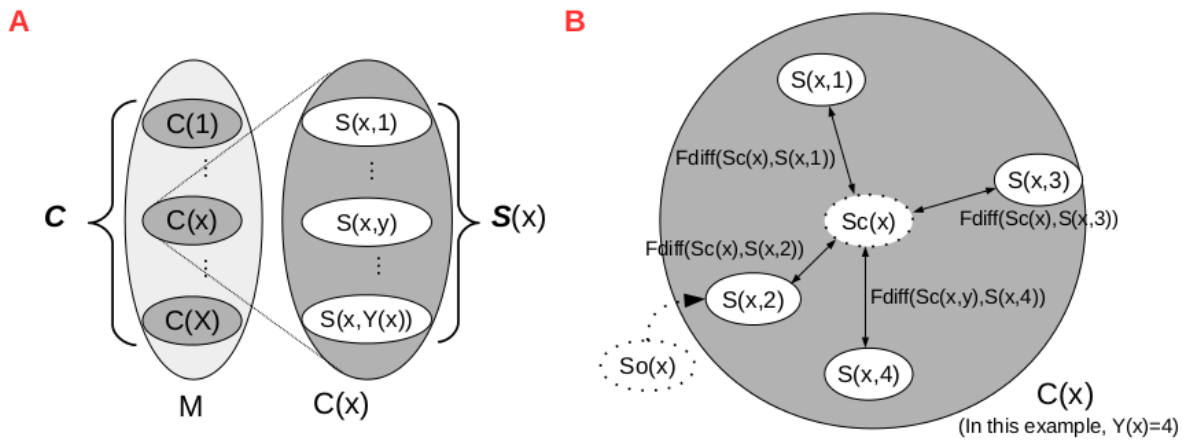


図3.1 Saturnのデータ構造

(A)クラスタ・マネージャMとクラスタC(x)と配列S(x,y)の入れ子構造。(B)クラスタ内C(x)の配列S(x,y)。Sc(x)が仮想的な重心配列。So(x)がクラスタ代表配列。So(x)はS(x,y)の中で最もSc(x)に近いもの (すなわち、Fdiff(Sc(x),S(x,y))が小さいもの)。Fdiff(Sc(x),S(x,y))はユーザーが指定した値 (クラスタの半径)に収まる。

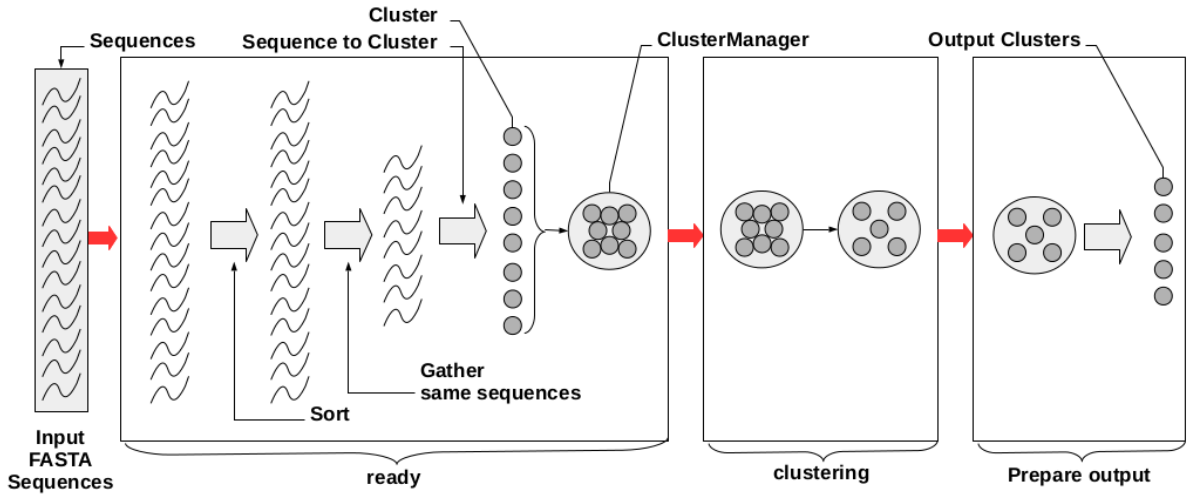


図3.2 アルゴリズムの各段階の定義

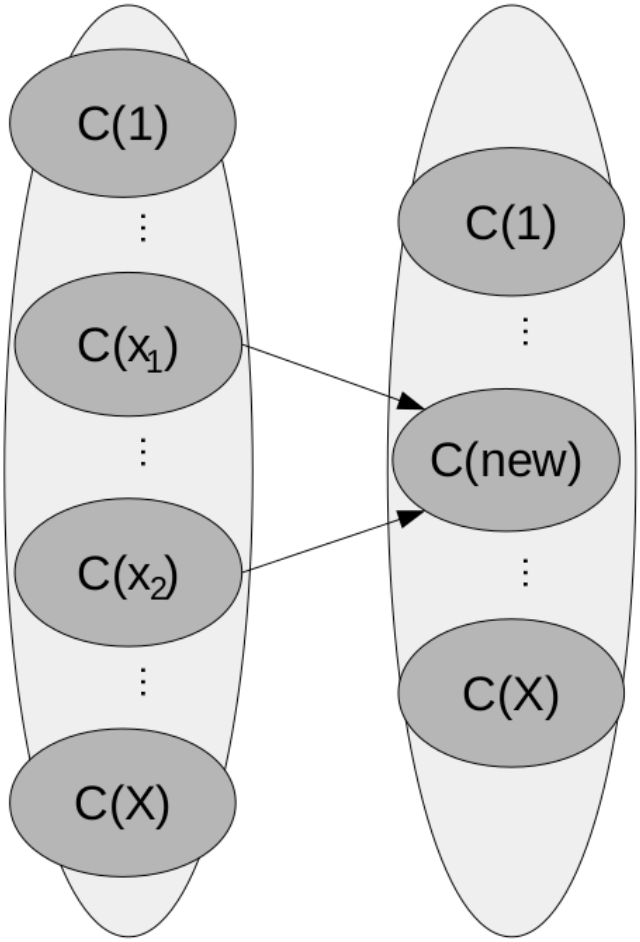
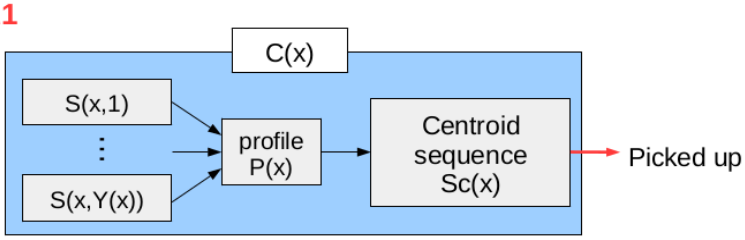
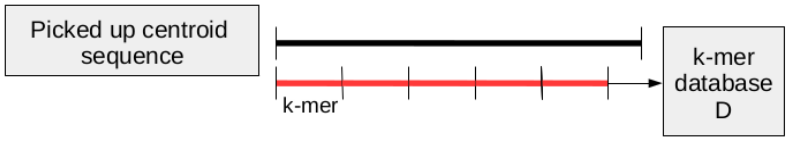


図3.3 クラスタの結合の模式図

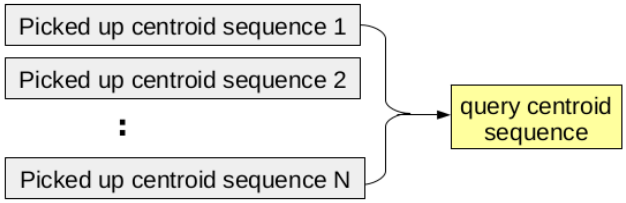
A1



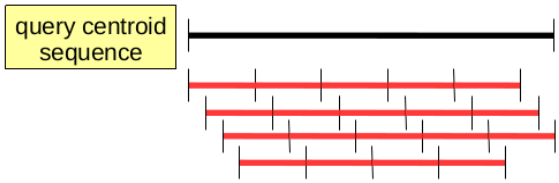
A2



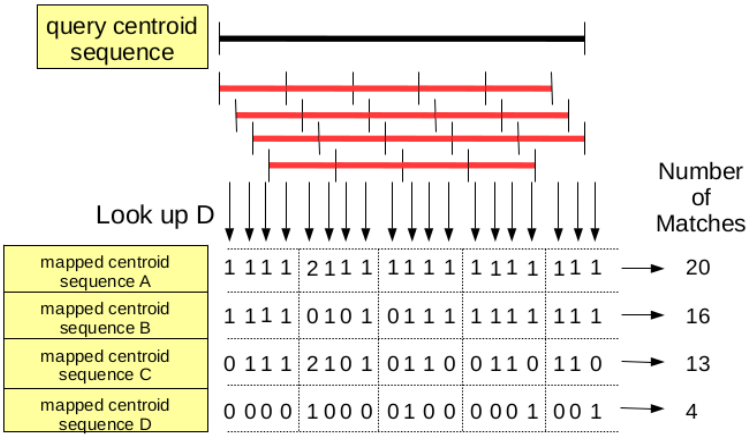
B1



B2



B3



B4

	Number of match	Try priority	Result of the trial
mapped centroid sequence A	20	1	succeeded
mapped centroid sequence B	20	2	succeeded
mapped centroid sequence C	15	3	failed ← 1 time
mapped centroid sequence D	14	4	succeeded
mapped centroid sequence E	10	5	failed ← 3 times
mapped centroid sequence F	9	6	failed ← 3 times
mapped centroid sequence G	7	7	failed ← 3 times
mapped centroid sequence H	7	8	succeeded
:	:	:	:

図3.4 クラスタリングの手順

(A1)クラスタの重心配列の読み取り。(A2)重心配列からのk-merデータベースDの作成。(B1)クエリ・クラスタの選定。(B2)クエリ・クラスタの重心配列からのk-mer読み取り。(B3)Dとのk-mer一致数のクラスタ別カウント。(B4)k-mer一致数が多い順の結合試行。

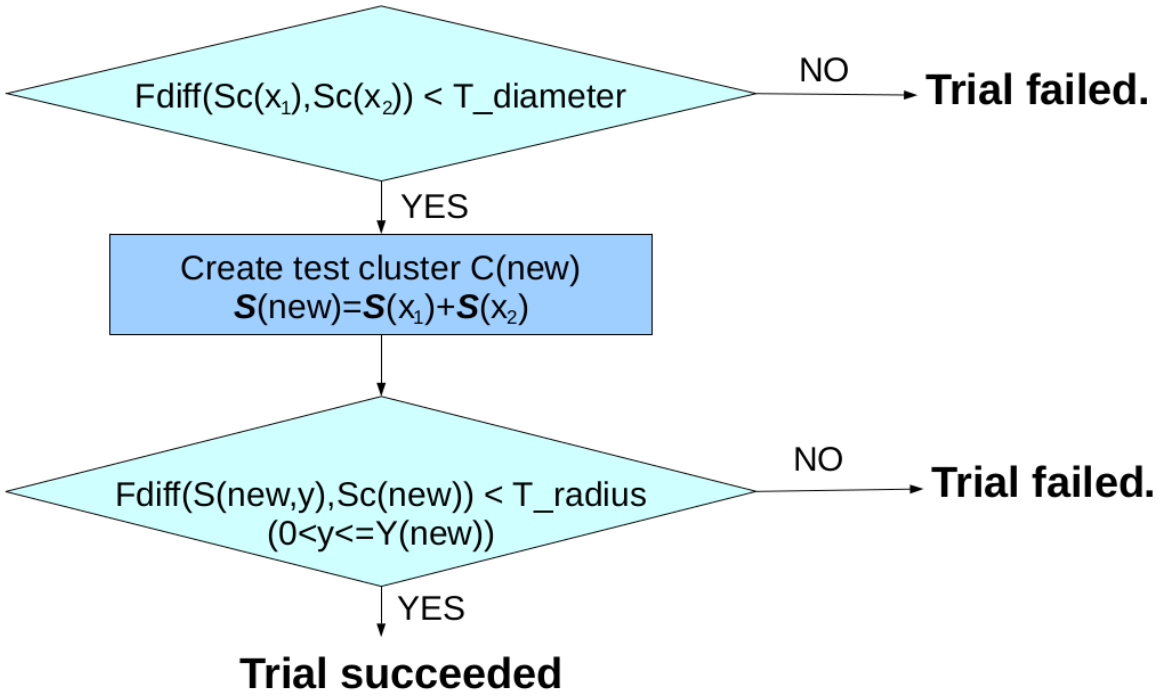
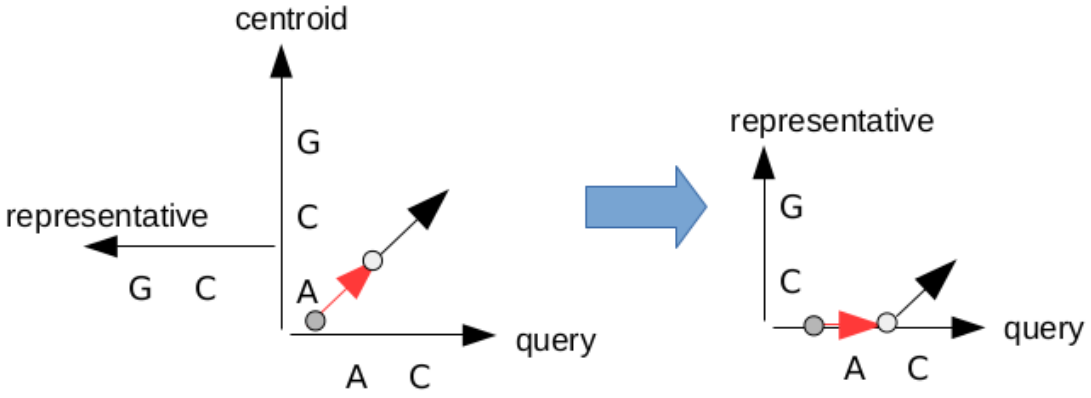
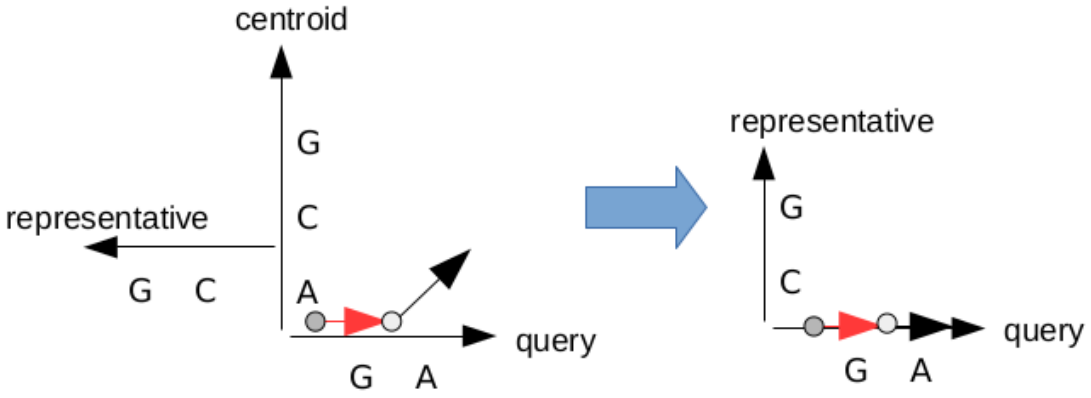


図3.5 クラスタとクラスタを結合するか否かの判定フローチャート

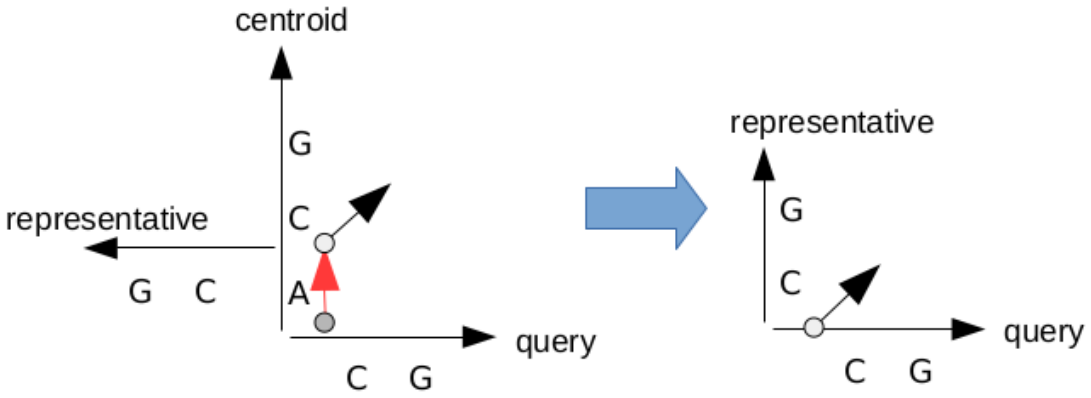
[A1] M (query→centroid) & ([query start position] < [representative start position])
= I (query→representative)



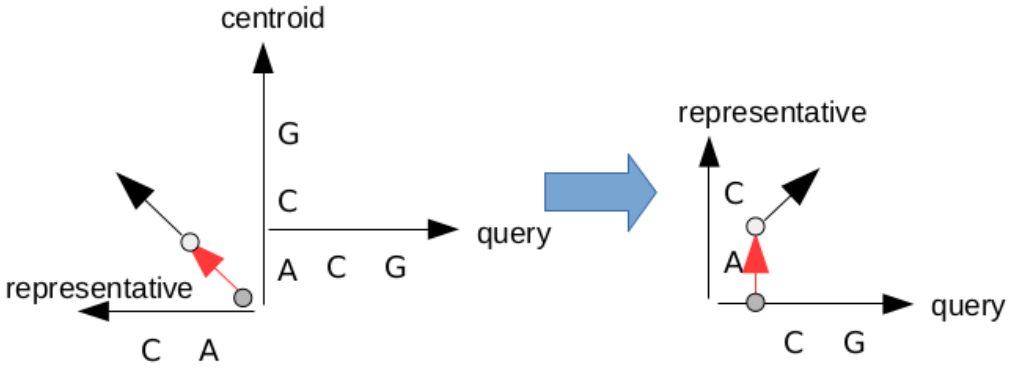
[A2] I (query→centroid) & ([query start position] < [representative start position])
= I (query→OTU)



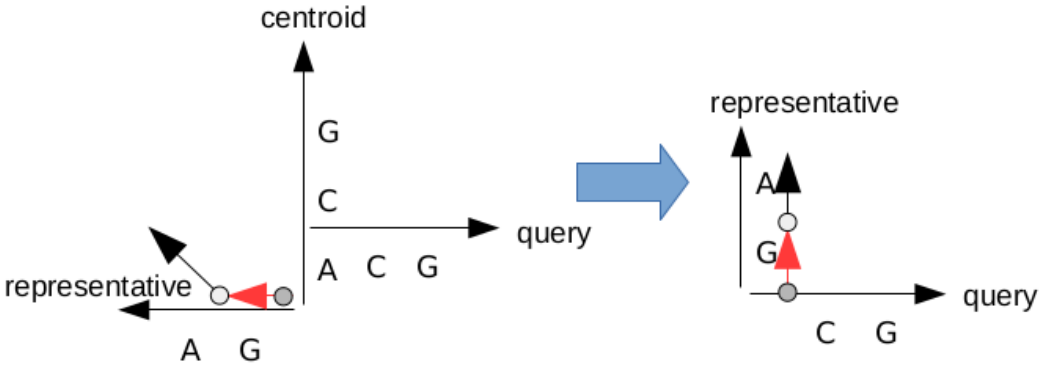
[A3] D (query→centroid) & ([query start position] < [representative start position])
= [No Alignment] (query→representative)



[B1] M (representative→centroid) & ([query start position] > [representative start position])
= D (query→representative)



[B2] I (representative→centroid) & ([query start position] > [representative start position])
= D (query→representative)



[B3] D (representative→centroid) & ([query start position] > [representative start position])
= [No Alignment] (query→representative)

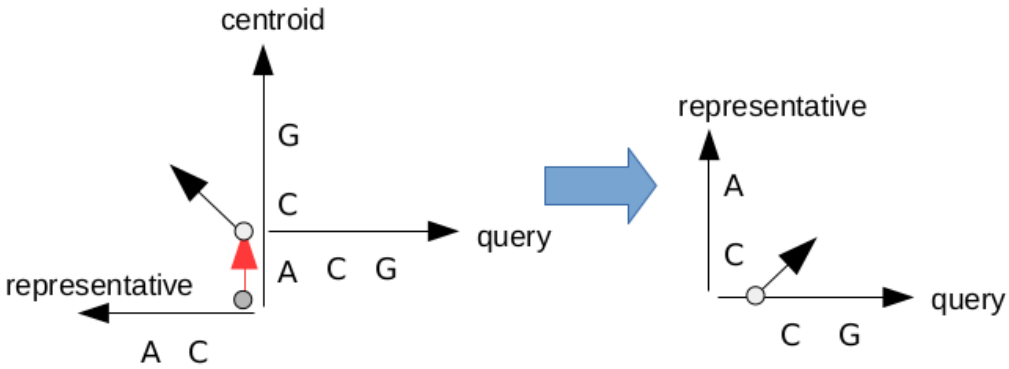
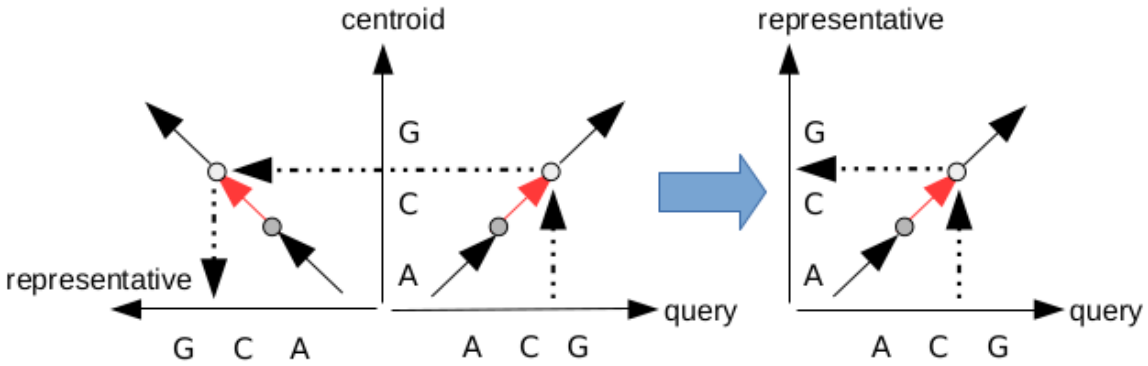


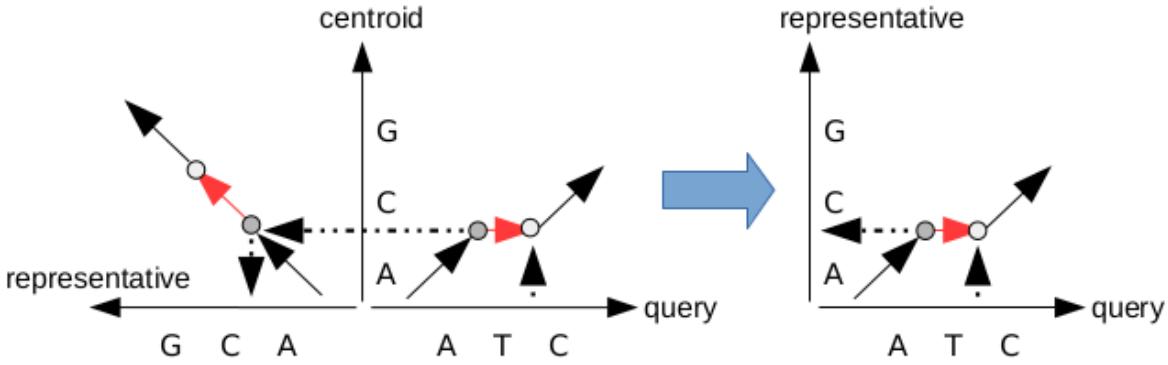
図3.6 クエリ配列と重心配列とのアライメントと、クラスタ代表配列と重心配列とのアライメントから、クエリ配列とクラスタ代表配列とのアライメントを生成する方法 (アライメントの始端部分の生成法)

“X(S→S’)”という表記で、配列Sから配列S’へのアライメントがXであったことを示す。XはMかIかDである。MはMatchあるいはMismatchであり、IはInsertion (挿入)であり、DはDeletion (欠失)である。”No alignment”とは、新しく生成されるアライメントにMもIもDも追加されないことを示す。例えば[A1]は、クラスタ代表配列から重心配列へのアライメントがMであり、かつ重心配列におけるクラスタ代表配列の先端の位置よりもクエリ配列の先端の位置が後ろにある場合、クエリ配列からクラスタ代表配列へのアライメントにDが追加されるということを示している。

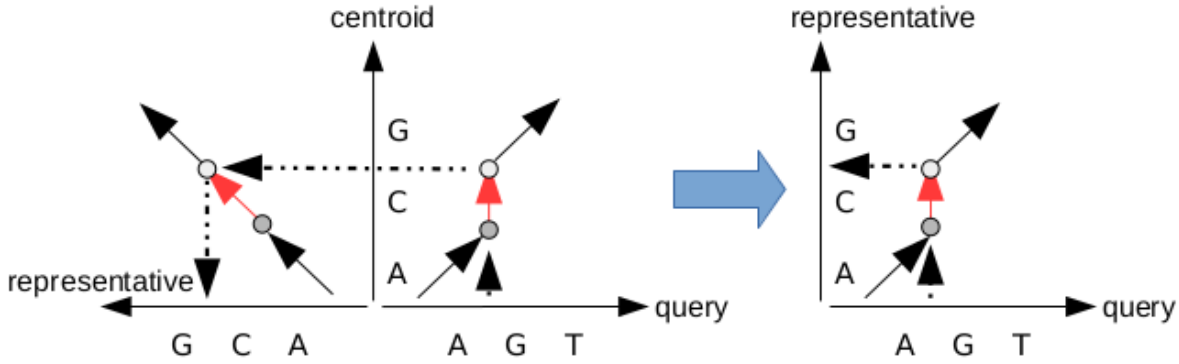
[1] M (query→centroid) - M (representative→centroid)
= M (query→representative)



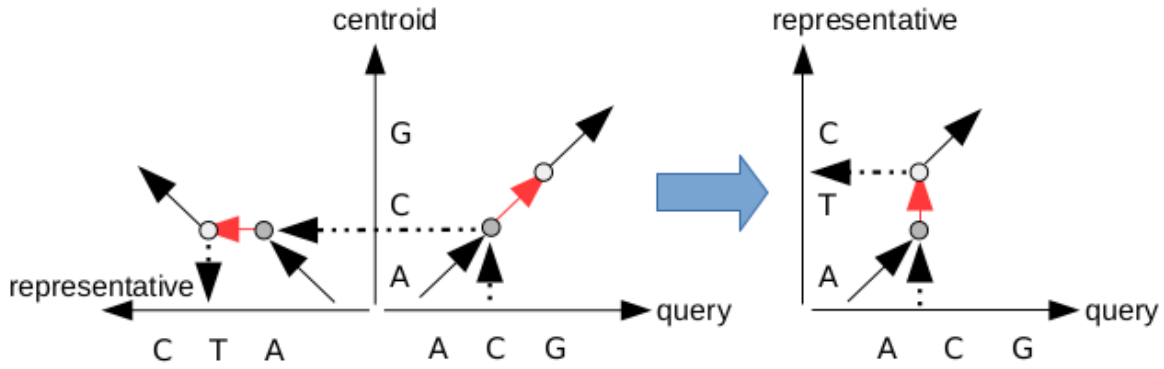
[2] I (query→centroid) - M (representative→centroid)
= I (query→representative)



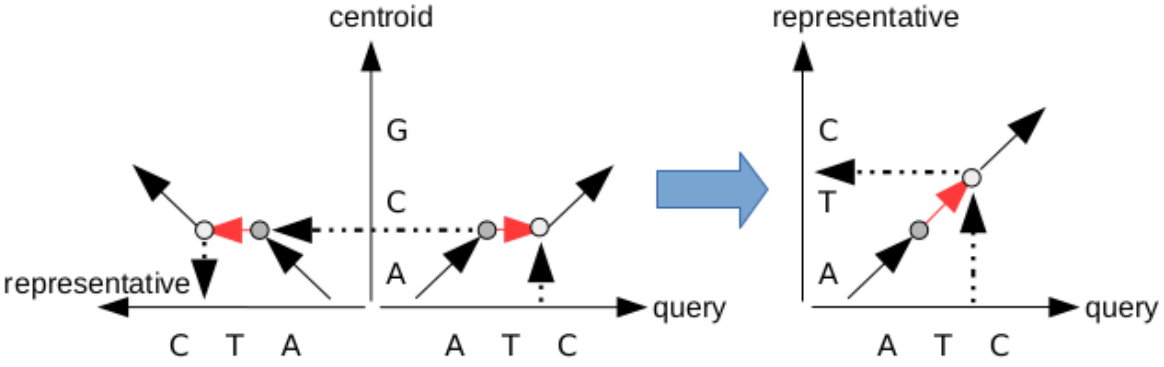
[3] D (query→centroid) - M (representative→centroid)
= D (query→representative)



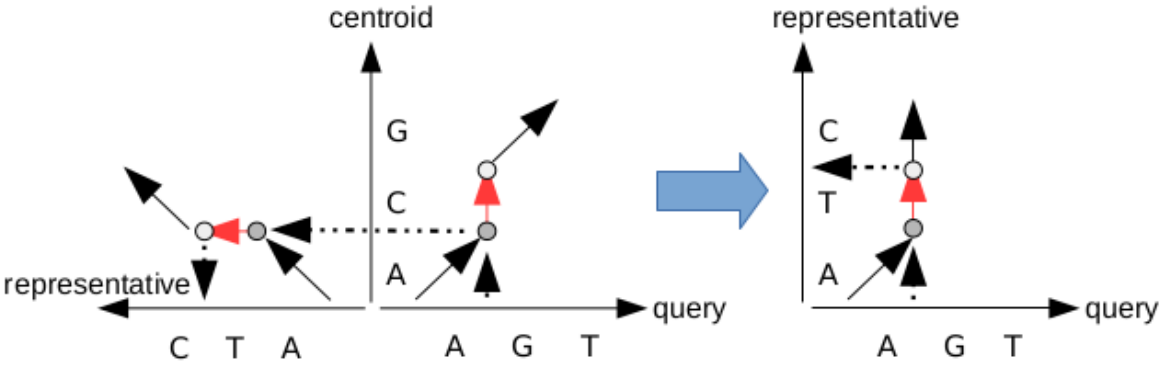
[4] M (query→centroid) - I (representative→centroid)
= D (query→representative)



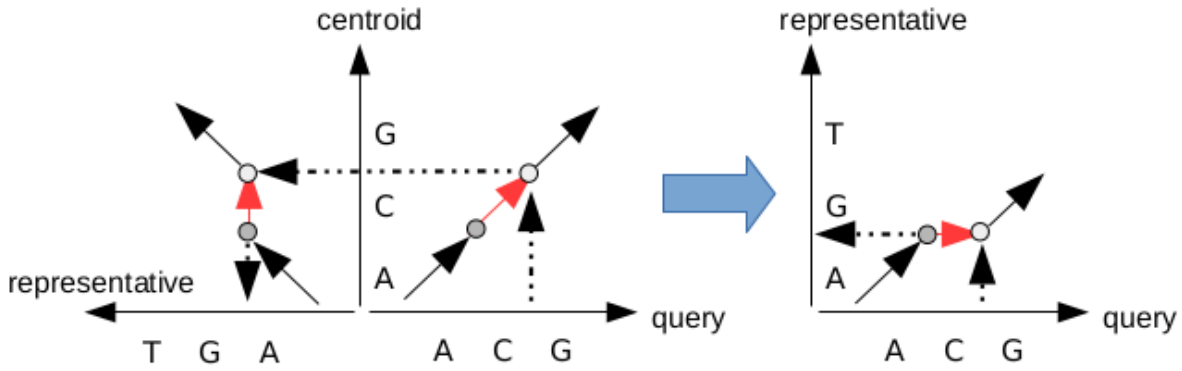
[5] I (query→centroid) - I (representative→centroid)
= M (query→representative)



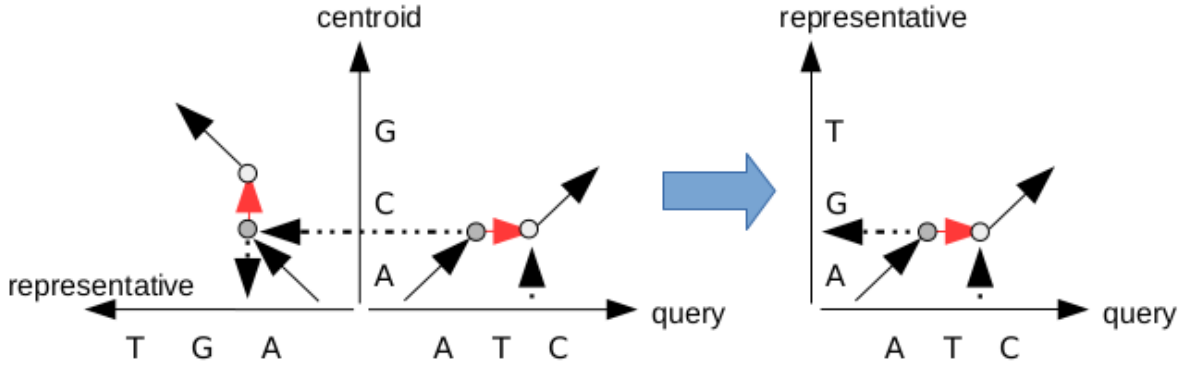
[6] D (query→centroid) - I (representative→centroid)
= D (query→representative)



[7] M (query→centroid) - D (representative→centroid)
= I (query→representative)



[8] I (query→centroid) - M (representative→centroid)
= I (query→representative)



[9] D (query→centroid) - D (representative→centroid)
= [No Alignment] (query→representative)

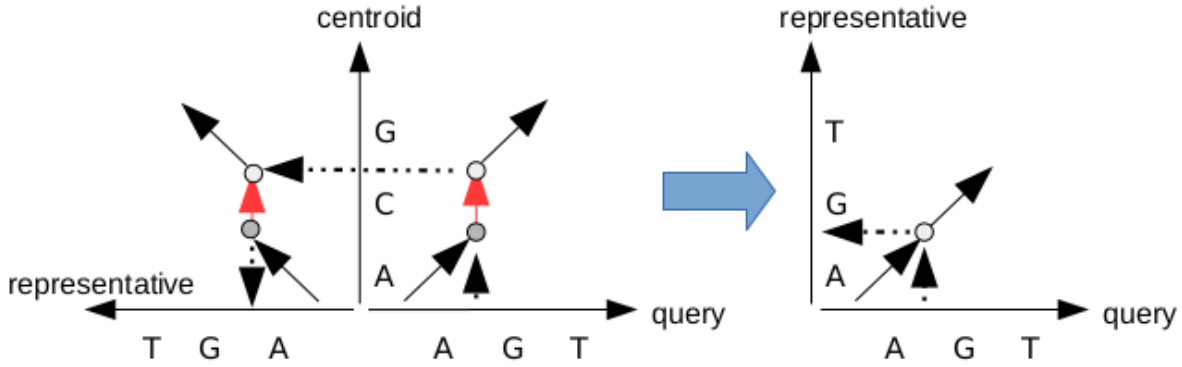


図3.7 クエリ配列と重心配列とのアライメントと、クラスタ代表配列と重心配列とのアライメントから、クエリ配列とクラスタ代表配列とのアライメントを生成する方法 (アライメントの途中部分の生成法)

図中の”X(S→S’)”の読み方は図3.6と同じ。たとえば[1]は、クエリ配列から重心配列へのアライメントがMであり、クラスタ代表配列から重心配列へのアライメントもMである場合、クエリ配列からクラスタ代表配列へのアラインメントもMになるということを示している。

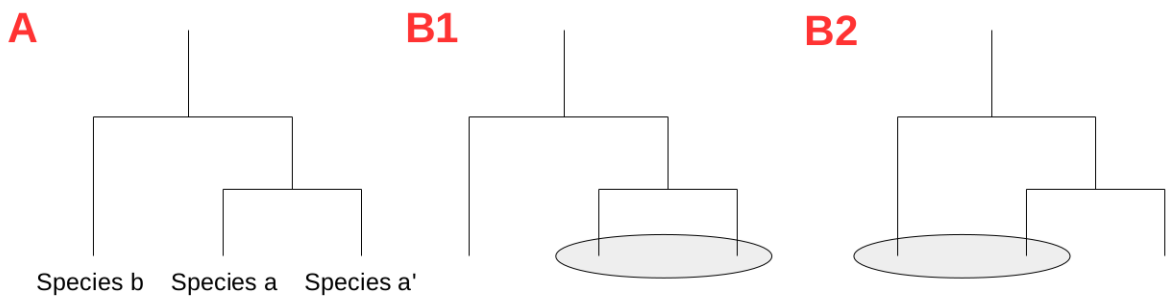


図3.8 単系統になる配列のみから構成されるクラスタ (単系統クラスタ)

(A) 3つの種a、a'、bからなる系統樹。(B1) 種aとa'からなるクラスタ。これは系統樹上の枝をまたいでいないので、単系統になる配列のみから構成されるクラスタである。(B2) 種aとbからなるクラスタ。これは系統樹上の枝をまたいでいるので、単系統になる配列のみから構成されるクラスタではない。

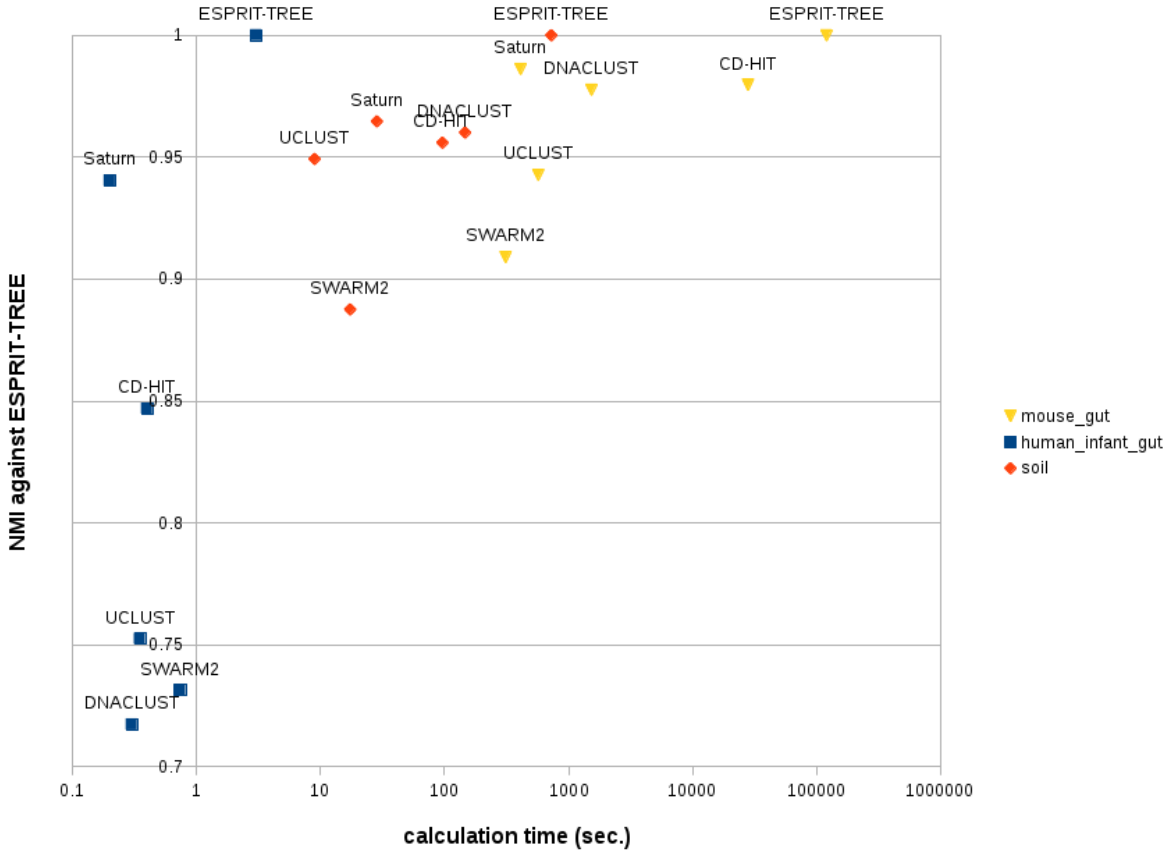


図3.9 サンプル別の各ツールでの計算時間とESPRIT-TREEと比較した際のNMIを散布図にした結果

表3.1 出力ファイルの形式

label	**S**	**F**	**C**
Column 2	クラスタID	クラスタID	クラスタID
Column 3	クラスタ代表 配列の長さ	クエリ配列の長さ	配列の数
Column 4	*	クラスタ代表配列とクエリ配列 のアイデンティティ	*
Column 5	*	ストランド(+のみ)	*
Column 6	*	クエリ配列の先端を基点とした クラスタ代表配列の先端の位置	*
Column 7	*	クラスタ代表配列の先端を基点 としたクエリ配列の先端の位置	*
Column 8	*	アライメント	*
Column 9	クラスタ代表 配列のヘッダ	クエリ配列のヘッダ	クラスタ代表 配列のヘッダ
Column 10	*	クラスタ代表配列のヘッダ	*

表3.2 試験したいツール別の、使用したツールとそれに与えたコマンドラインオプション

試験したいツール名	実行するツール名	バージョン	モード	コマンドラインオプション
SaturnCluster	SaturnCluster	-	-	-
UCLUST	UCLUST	7.0.1090	sort	-sortbylength -minseqlength 64
			clustering	-cluster_smallmem -id 0.97 -query_cov 0.9 -target_cov 0.9
CD-HIT	CD-HIT	4.6.1	-	-uL 0.1 -uS 0.1 -n 5 -M 200000 -d 100
DNACLUST	DNACLUST	release3	-	-l -s
Swarm2	Swarm2	2.1.9	-	-
	VSEARCH	2.0.2	-	--derep_fulllength --sizeout --relabel_sha1 –fasta_width 0
ESPRIT-TREE	ESPRIT-TREE	2011 Nov.15	-	(*)
Mothur	MAFFT	7.058b	-	--maxiterate 1000
	PHYLIP	3.696	-	-
	mother	1.37.6	-	(**)

(*) esprit-tree.shのpbpclusterを呼ぶ行に"-u 0.03"オプションを追加。

(**) cluster()関数にphylipファイルを渡して”hard=t"オプション付きで実行。

表3.3 サンプル別の各ツールでの計算時間とESPRIT-TREEと比較した際のNMI

ツール名	計算時間(秒)	クラスタ数	NMI	NMIの不偏標準偏差
Saturn	410.779	46,807	0.9862	0.0012
UCLUST	569.79	68,274	0.9428	0.0121
CD-HIT	28026	49,378	0.9796	0.0052
Swarm2	310.78	196,746	0.9090	0.0018
DNACLUST	1533.312	54,886	0.9777	0.0000
mothur	-	-	0.9939	0.0011
ESPRIT-TREE	120436.503	46,380	1.0000	0.0000

上の表がマウス腸内の場合。mothurは速度試験では使用していない。

ツール名	計算時間(秒)	クラスタ数	NMI	NMIの不偏標準偏差
Saturn	0.2	51	0.9404	-
UCLUST	0.35	74	0.7524	-
CD-HIT	0.4	28	0.8469	-
Swarm2	0.74	216	0.7313	-
DNACLUST	0.301	59	0.7174	-
mothur	-	-	0.9310	-
ESPRIT-TREE	3.039	47	1.0000	-

上の表がヒト乳児腸内の場合。mothurに関して同上。1回の試行でデータの全体を試験できたため、不偏標準偏差は計算していない。

ツール名	計算時間(秒)	クラスタ数	NMI	NMIの不偏標準偏差
Saturn	28.467	5,243	0.9647	0.0023
UCLUST	8.96	6,465	0.9493	0.0036
CD-HIT	96.35	4,903	0.9560	0.0023
Swarm2	17.35	20,498	0.8876	0.0017
DNACLUST	146.549	5,319	0.9601	0.0021
mothur	-	-	0.9827	0.0020
ESPRIT-TREE	726.293	4,855	1.0000	0.0000

上の表が土壌の場合。mothurに関して同上。

表3.4 系統樹試験の結果

ツール名	単系統クラスタ率 (有効数字4桁)	シングルトンの数
mothur	0.9761	812
Swarm2	0.9979	1424
ESPRIT-TREE	0.9704	769
Saturn	0.9634	825
UCLUST	0.9617	830
CD-HIT	0.9569	802
DNACLUST	0.9562	808

第4章　本研究のまとめ

新しい高速かつ高感度な配列相同性検索ツールCLASTは、メタゲノム解析へと応用されるために設計され、検証された。CLASTでは、ノンコーディングRNA遺伝子配列のようなアミノ酸配列に翻訳されない配列をも解析できるようにするために、配列データベースもクエリ配列群も共に塩基配列での配列相同性検索を行う。CLASTは、CUDAを用いたプログラムを使用可能なNVIDIA社製GPUを用いて配列相同性検索を並列化したことで、BLASTより80.8倍高速でBLATよりも9.6倍速く計算できた。CLASTは、グローバルおよびローカルの両方のアライメントをサポートすることで、系統推定の精度とモチーフ検索の感度を向上させることもできた。さらに、CLASTは大規模なデータベースの前処理を必要としないため、配列データベースの更新に即応することが可能である。しかも、CLASTのメモリ使用量はRAMもVRAMも共に配列データベースのサイズにもクエリ配列群のサイズにも比例しないため、CLASTはNVIDIA社製GPUを搭載した標準的なデスクトップコンピュータでも容易に実行することができる。またGPUを使った計算クラスタまたはスーパーコンピュータ上でCLASTを実行すれば、それは新型シークエンサーから得られた大量の配列データを分析する最も強力かつ現実的なアプローチの一つになると考えられる。

新しい超高速かつ種に応じた高精度な配列クラスタリングツールSaturnは、メタ16S解析へと応用されるために設計され、検証された。Saturnは、同じ長さかつ完全に一致する配列(重複配列)の本数が多い順に、クラスタの中心を推定してグリーディー・アルゴリズムによる配列クラスタリングを行う。またSaturnは、クラスタに仮想的な重心配列と代表配列を別個に持つ。Saturnは、UCLUSTよりも1.39倍高速にクラスタリングを行うことができ、さらにCD-HITとUCLUSTとDNACLUSTのいずれよりも試験した全てのケースでESPRIT-TREEの結果に近いクラスタリング結果を出力できた。また、speciesあたり1本の配列1,472本から作成した進化系統樹を用いた試験では、SaturnはUCLUST よりも5本 (全体の0.34%)多くの配列をクラスタリングできており、かつSaturnの方がUCLUSTよりも単系統クラスタ率も0.16%高かった。Saturnの高い精度は重複配列の本数を基準に用いてクラスタリングを行うという特徴に由来すると考えられた。まとめると、Saturnは、既存のグリーディなクラスタリングツールの中

でもトップクラスの計算速度で、しかも階層的クラスタリングを用いたESPRIT-TREEの結果に近い種に応じた高精度なクラスタリングを実現できることが分かった。なおSaturnは、16S rRNA遺伝子配列について特別にデータベースを構築することはないため16S rRNA遺伝子配列以外の配列クラスタリングにも使用可能であり、配列相同性検索における配列データベースの配列クラスタリングによるサイズ削減にも使用可能である。Saturnを用いることで、使用できる計算資源が限られた多くの利用者にもメタ16S解析がより容易に行えるようになり、また数百以上という多数のサンプルに由来するメタ16S配列の解析をより現実的なものにできると考えられる。

本研究で実装したCLASTとSaturnにより、メタゲノム解析とメタ16S解析に際し行われてきた計算それぞれの高速化と高精度化とが同時に達成された。CLASTは新しく台頭してきた計算機環境を活用するために新しくアルゴリズムを開発することによって実装され、Saturnは処理するデータの特徴に注目した新しいアルゴリズムを開発することによって実装された。CLASTとSaturnを使用することにより、環境中の微生物叢に関する知見がより容易に得られるようになると考えられる。ここから、本研究は環境と環境中の微生物叢との相互作用についての探求を一般化・大規模化・リアルタイム化していく有力な手段を開発できたと考えられる。様々な場所ごとの環境と環境中の微生物叢との相互作用をより多くの利用者がリアルタイムで解析可能になれば、様々な場所ごとでの微生物叢のモニタリングに基づいた環境の制御も社会でより広範に可能になると考えられる。

本研究の今後の進展としては、新しい計算機環境へのさらなる対応が考えられる。まずCLASTは、NVIDIA社が開発したFermiアーキテクチャおよびKeplerアーキテクチャより新しいアーキテクチャのGPUには最適化されていないことから、そうした新しいアーキテクチャのGPUへとCLASTを最適化することでさらなる高速化が実現できると考えられる。そしてSaturnは、並列計算にはまだ対応していないことから、互いに距離が離れたクラスタのクラスタリングを並列で行うことでさらなる高速化が可能であると考えられる。配列相同性検索でも配列クラスタリングでも、常に新しいアーキテクチャの計算機に最適化された新しいアルゴリズムを開発し続けることで、計算機環境の発展の成果を活用し続けることができると期待できる。

CLASTは2000年代後半以降広く活用されるようになってきたGPUを新しく開発したアルゴリズムを用いることでメタゲノム解析における配列相同性検索に応用したツールであり、Saturnは「メタ16S解析における配列クラスタリングにおいて、クラスタの中心を重複配列の本数により推定することでグリーディでも種に応じたクラスタリングができるようになる」という発想のアルゴリズムを用いたツールであった。ここから私たちは、第一章で述べた通り、計算機環境の状況と問題の特性を鑑みた適切なアルゴリズムを用いたツールを作成することで、より多くの研究者により多くの知見を得られるようにすることを目的に研究を行い、この目的を達成したと総括できる。

計算機の計算速度とメモリ容量と選択するアルゴリズムにより変わる計算可能な計算量の差異は量的である一方で、個々の研究者にとって個々の問題が計算可能か否かの差異は質的である。この計算機の計算速度とメモリ容量と選択するアルゴリズムによって個々の研究者にとって計算可能な問題が変わるという事態は、量の変化が質の変化に転化する(Engels and Haldane 1940)というDNAの存在が発見される以前に既に提唱されていた「量質転化の法則」の一部に従っている。

閾値が介在する現象においては、量質転化の法則が成り立つ。世界には数多くの閾値が存在しており、それぞれの閾値がどれほど超えられやすいかは状況により変わる。解決可能な問題も、必要な計算量を閾値として状況により変化する。状況を調査・判断したうえで必要な計算量を実行できるか否かを検討することによって、解決可能な問題は予想できる。解決可能だと予想された問題に注力することができれば、それができなかった場合と比べて、より多くの問題に対して解決手段を与えることができ、それまで解決不可能であった問題が解決可能になっていくだろう。そうした中から新たな研究分野が開かれることもあるのは、私たちが第1章の冒頭で例示したとおりである。

まとめると、本研究で開発されたCLASTとSaturnによって、それぞれメタゲノム解析における配列相同性検索とメタ16S解析における配列クラスタリングが、結果の精度を維持したまま高速化された。この成果は、計算機環境の発展を有効に活用する新たなアルゴリズムと、分析するデータの特徴に注目した新たなアルゴリズムによって達成された。このCLASTと

Saturnによって、環境と環境中の微生物叢との相互作用についての探求を推進することができると考えられる。しかも、これからも計算機環境の変化や、分析するデータの特徴に関する新たな知見を踏まえることで、さらなる高速化や精度の向上が期待できる。そうしたさらなる高速化や精度の向上によって、さらに多くの問題をさらに多くの利用者が計算可能になり、環境と環境中の微生物叢との相互作用についての理解が進み、その過程で新たな研究分野が開かれる可能性も期待できる。

参考文献

6. Programming Environment: Available at: http://tsubame.gsic.titech.ac.jp/docs/guides/tsubame2/html_en/programming.html Accessed Jan. 17 2017.

Altschul S.F., Gish W., Miller W., Myers E.W., Lipman D.J.: Basic local alignment search tool. *Journal of Molecular Biology*. 1990, 215: 403-410. 10.1016/S0022-2836(05)80360-2.

abundance sort: Available at: http://drive5.com/usearch/manual/abundance_sort.html Accessed Dec.12 2016.

Berg G.: Plant-microbe interactions promoting plant growth and health: perspectives for controlled use of microorganisms in agriculture. *Applied Microbiology and Biotechnology*. 2009, 84(1), 11-18.

BLAT Suite Program Specifications and User Guide: Available at: http://genome.ucsc.edu/goldenPath/help/blatSpec.html Accessed Dec.12 2016.

Blom J., Jakobi T., Doppmeier D., Jaenicke S., Kalinowski J., Stoye J., Goesmann A.: Exact and complete short read alignment to microbial genomes using GPU programming. *Bioinformatics*. 2011, 27: 1-8. 10.1093/bioinformatics/btr151.

Bowtie2: Manual: Available at: http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml Accessed Dec.12 2016.

Buchfink B., Xie C., Huson D.H.: Fast and sensitive protein alignment using DIAMOND. *Nature Methods*. 2015, 12(1), 59-60.

Burrows-Wheeler Aligner: Available at: http://bio-bwa.sourceforge.net Accessed Dec.12 2016.

bwa.1: Available at: http://bio-bwa.sourceforge.net/bwa.shtml Accessed Jan.28 2017.

Cai Y., Sun Y.: ESPRIT-Tree: hierarchical clustering analysis of millions of 16S rRNA pyrosequences in quasilinear computational time. *Nucleic Acids Research*. 2011, gkr349.

Chao KM, Pearson WR, Miller W: Aligning two sequences within a specified diagonal band. *Computer Applications in the Biosciences*. 1992, 8: 481-487.

cluster sizes: Available at: http://www.drive5.com/usearch/manual/cluster_sizes.html Accessed Dec.12 2016.

Dinsdale E.A., Edwards R.A., Hall D., Angly F., Breitbart M., Brulc J.M., Furlan M., Desnues C., Haynes M., Li L., McDaniel L., Moran M.A., Nelson K.E., Nilsson C., Olson R., Paul J., Brito B.R., Ruan Y., Swan B.K., Valentine R.S.D.L., Thurber R.V., Wegley L., White B.A., Rohwer F.: Functional metagenomic profiling of nine biomes. *Nature*. 2008, 452: 629-632. 10.1038/nature06810.

Drancourt M., Bollet C., Carlioz A., Martelin R., Gayral J.P., Raoult D.: 16S ribosomal DNA sequence analysis of a large collection of environmental and clinical unidentifiable bacterial isolates. *Journal of Clinical Microbiology*. 2000, 38(10), 3623-3630.

Edgar R.C.: Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*. 2010, 26(19), 2460-2461.

Engels F., Haldane J.B.S.: *Dialectics of nature* (pp. 291-92). C. P. Dutt (Ed.). New York: International publishers. ISO 690. 1940.

Falkowski P.G., Fenchel T., Delong E.F.: The microbial engines that drive Earth's biogeochemical cycles. *Science*. 2008, 320(5879), 1034-1039.

Felsenstein J.: PHYLIP (Phylogeny Inference Package) version 3.6. *Distributed by the author. Department of Genome Sciences, University of Washington, Seattle*. 2005.

Ghodsi M., Liu B., Pop M.: DNACLUST: accurate and efficient clustering of phylogenetic marker genes. *BMC Bioinformatics*. 2011, 12(1), 1.

Gribskov M., McLachlan A.D., Eisenberg, D.: Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences*. 1987, 84(13), 4355-4358.

Griffiths-Jones S., Moxon S., Marshall M., Khanna A., Eddy S.R., Bateman A.: Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Research*. 2005, 33(suppl 1), D121-D124.

Haferburg G., Kothe E.: Microbes and metals: interactions in the environment. *Journal of Basic Microbiology*. 2007, 47(6), 453-467.

Hamady M., Knight R.: Microbial community profiling for human microbiome projects: Tools, techniques, and challenges. *Genome Research*. 2009, 19(7), 1141-1152.

Hanash S.M., Bobek M.P., Rickman D.S., Williams T., Rouillard J.M., Kuick R., Puravs E.: Integrating cancer genomics and proteomics in the post-genome era. *Proteomics*. 2002, 2(1), 69-75.

Huse S.M., Welch D.M., Morrison H.G., Sogin M.L.: Ironing out the wrinkles in the rare biosphere through improved OTU clustering. *Environmental Microbiology*. 2010, 12(7), 1889-1898.

Huson D.H., Weber N.: Microbial community analysis using MEGAN. *Methods in Enzymology*. 2013, 531: 465-485. 10.1016/B978-0-12-407863-5.00021-6.

Jia Y., Shelhamer E., Donahue J., Karayev S., Long J., Girshick R., Guadarrama S., Darrell T.: Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. (pp. 675-678). ACM. 2014.

Jokinen P, Ukkonen E: Two algorithms for approximate string matching in static texts. *Lecture Notes in Computer Science*. 1991, 520: 240-248. 10.1007/3-540-54345-7_67.

Kato H., Mori H., Maruyama F., Toyoda A., Oshima K., Endo R., Fuchu G., Miyakoshi M., Dozono A., Ohtsubo Y., Nagata Y., Hattori M., Fujiyama A., Kurokawa K., Tsuda M.: Time-series metagenomic analysis reveals robustness of soil microbiome against chemical disturbance. *DNA Research*. 2015, 22(6), 413-424.

Katoh K., Misawa K., Kuma K., Miyata T.: MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*. 2002, 30(14) 3059-3066.

Kent W.J.: BLAT—the BLAST-like alignment tool. *Genome Research*. 2002, 12: 656-664. 10.1101/gr.229202. Article published online before March 2002.

Koch R.: The etiology of anthrax, based on the life history of Bacillus anthracis. *Beitrage zur Biologie der Pflanzen*. 1876, 2, 277-310.

Kong H.H.: Skin microbiome: genomics-based insights into the diversity and role of skin microbes. *Trends in Molecular Medicine*. 2011, 17(6), 320-328.

Kuczynski J., Stombaugh J., Walters W.A., Gonzalez A., Caporaso J.G., Knight R.: Using QIIME to analyze 16S rRNA gene sequences from microbial communities. *Current Protocols in Microbiology*. 2012, 1E-5.

Kunin V., Copeland A., Lapidus A., Mavromatis K., Hugenholtz P.: A bioinformatician's guide to metagenomics. *Microbiology and Molecular Biology Reviews*. 2008, 72(4), 5.

Kurokawa K., Itoh T., Kuwahara T., Oshima K., Toh H., Toyoda A., Takami H., Morita H., Sharma V.K., Srivastava T.P., Taylor T.D., Noguchi H., Mori J., Ogura Y., Ehrlich D.S., Itoh K., Takagi T., Sakaki Y., Hayashi T., Hattori M.: Comparative metagenomics revealed

commonly enriched gene sets in human gut microbiomes. *DNA Research*. 2007, 14(4), 169-181.

Land M., Hauser L., Jun S.R., Nookaew I., Leuze M.R., Ahn T.H., Karpinets T., Lund O., Kora G., Wassenaar T., Poudel S., Ussery D.W.: Insights from 20 years of bacterial genome sequencing. *Functional & Integrative Genomics*. 2015, 15(2), 141-161.

Langmead B., Salzberg S.L.: Fast gapped-read alignment with Bowtie 2. *Nature Methods*. 2012, 9(4), 357-359.

Langmead B., Trapnell C., Pop M., Salzberg S.L.: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*. 2009, 10:R25.

Ley R.E., Peterson D.A., Gordon J.I.: Ecological and evolutionary forces shaping microbial diversity in the human intestine. *Cell*. 2006, 124(4), 837-848.

Li D., Liu C.M., Luo R., Sadakane K., Lam T.W.: MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*. 2015, btv033.

Li H.: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997*. 2013.

Li H., Durbin R.: Fast and accurate long read alignment with Burrows-Wheeler transform. *Bioinformatics*. 2010, 26: 589-595. 10.1093/bioinformatics/btp698.

Li H., Durbin R.: Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*. 2009, 25: 1754-1760. 10.1093/bioinformatics/btp324.

Li W., Fu L., Niu B., Wu S., Wooley J.: Ultrafast clustering algorithms for metagenomic sequence analysis. *Briefings in Bioinformatics*. 2012, bbs035.

Li W., Godzik A.: Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*. 2006, 22(13), 1658-1659.

Li, R., Yu, C., Li, Y., Lam, T. W., Yiu, S. M., Kristiansen, K., Wang, J.: SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*. 2009, 25(15), 1966-1967.

Liu Y., Schmidt B., Maskell D.L.: CUDASW++2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions. *BMC Research Notes*. 2010, 3: 93-10.1186/1756-0500-3-93.

Mack C.A.: Fifty years of Moore's law. *IEEE Transactions on semiconductor manufacturing*. 2011, 24(2), 202-207.

Mahe F., Rognes T., Quince C., De Vargas C., Dunthorn M.: Swarm v2: highly-scalable and high-resolution amplicon clustering. *PeerJ*. 2015, 3, e1420.

Matsuki T., Yahagi K., Mori H., Matsumoto H., Hara T., Tajima S., Ogawa E., Kodama H., Yamamoto K., Yamada T., Matsumoto S., Kurokawa K.: A key genetic factor for fucosyllactose utilization affects infant gut microbiota development. *Nature Communications*. 2016, 7.

Mori H., Maruyama F., Kurokawa K.: VITCOMIC: visualization tool for taxonomic compositions of microbial communities based on 16S rRNA gene sequences. *BMC Bioinformatics*. 2010, 11(1), 1.

NCBI RefSeq Genome Database: Available at: ftp://ftp.ncbi.nih.gov/genomes/Bacteria/ Accessed Oct.15 2011.

Needleman S.B., Wunsch C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*. 1970, 48:443-453.

Nickolls J., Dally W.J. The GPU computing era. Micro, *IEEE*. 2010, 30(2), 56-69.

Niu B., Zhu Z., Fu L., Wu S., Li W.: FR-HIT, a very fast program to recruit metagenomic reads to homologous reference genomes. *Bioinformatics*. 2011, 27: 1704-1705. 10.1093/bioinformatics/btr252.

November 2016 | TOP500 Supercomputer Sites Available at: https://www.top500.org/green500/lists/2016/11/ Accessed Dec.14 2016.

Nvidia C.U.D.A.: Compute unified device architecture programming guide. 2007.

Okai S., Usui F., Yokota S., Hori-I Y., Hasegawa M., Nakamura T., Kurosawa M., Okada S., Yamamoto K., Nishiyama E., Mori H., Yamada T., Kurokawa K., Matsumoto S., Nanno M., Naito T., Watanabe Y., Kato T., Miyauchi E., Ohno H., Shinkura R.: High-affinity monoclonal IgA regulates gut microbiota and prevents colitis in mice. *Nature Microbiology*. 2016, 1, 16103.

Pace, N.R.: A molecular view of microbial diversity and the biosphere. *Science*. 1997, 276(5313), 734-740.

Pearson W.R.: Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*. 1991, 11: 635-650. 10.1016/0888-7543(91)90071-L.

Programming Environment: Available at: https://sc.ddbj.nig.ac.jp/index.php/en/en-programming Accessed Jan. 19 2017.

Qin J., Li R., Raes J., Arumugam M., Burgdorf K.S., Manichanh C., Nielsen T., Pons N., Levenez F., Yamada T., Mende D.R., Li J., Xu J., Li S., Li D., Cao J., Wang B., Liang H., Zheng H., Xie Y., Tap J., Lepage P., Bertalan M., Batto J.M., Hansen T., Le Paslier D., Linneberg A., Nielsen H.B., Pelletier E., Renault P., Sicheritz-Ponten T., Turner K., Zhu H., Yu C., Li S., Jian M., Zhou Y., Li Y., Zhang X., Li S., Qin N., Yang H., Wang J., Brunak S., Dore J., Guarner F., Kristiansen K., Pedersen O., Parkhill J., Weissenbach J. MetaHIT

Consortium, Bork P., Ehrlich S.D., Wang J.: A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*. 2010, 464: 59-65. 10.1038/nature08821.

Quince C., Lanzen A., Curtis T.P., Davenport R.J., Hall N., Head I.M., Read F.L., Sloan W.T.: Accurate determination of microbial diversity from 454 pyrosequencing data. *Nature Methods*. 2009, 6(9), 639.

Raes J., Bork P.: Molecular eco-systems biology: towards an understanding of community function. *Nature Reviews Microbiology*. 2008, 6(9), 693-699.

Schatz M.C., Trapnell C., Delcher A.L., Varshney A.: High-throughput sequence alignment using Graphics Processing Units. *BMC Bioinformatics*. 2007, 8: 474-10.1186/1471-2105-8-474.

Schloss P.D., Westcott S.L., Ryabin T., Hall J.R., Hartmann M., Hollister E.B., Lesniewski R.A., Oakley B.B., Parks D.H., Robinson C.J., Sahl J.W., Stres B., Thallinger G.G., Horn D.J.V., Weber C.F.: Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and Environmental Microbiology*. 2009, 75.23, 7537-7541.

Smith T.F., Waterman M.S.: Identification of common molecular subsequences. *Journal of Molecular Biology*. 1981, 147:195-197.

Spellerberg I.F., Fedor P.J.: A tribute to Claude Shannon (1916–2001) and a plea for more rigorous use of species richness, species diversity and the 'Shannon–Wiener' Index. *Global Ecology and Biogeography*. 2003, 12(3), 177-179.

Stamatakis A.: RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*. 2014, 30(9), 1312-1313.

Stone J.E., Gohara D., Shi G.: OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in Science & Engineering*. 2010, 12(1-3), 66-73.

Sun Y., Cai Y., Huse S.M., Knight R. Farmerie W.G., Wang X., Mai V.: A large-scale benchmark study of existing algorithms for taxonomy-independent microbial community analysis. *Briefings in Bioinformatics*. 2012, 13(1) 107-121.

Suzuki S., Ishida T., Kurokawa K., Akiyama Y.: GHOSTM: A GPU-accelerated homology search tool for metagenomics. *PLoS One*. 2012, 7(5): e36060-10.1371/journal.pone.0036060.

Suzuki S., Kakuta M., Ishida T., Akiyama Y.: GHOSTX: an improved sequence homology search algorithm using a query suffix array and a database suffix array. *PLoS One*. 2014, 9(8), e103833.

The Human Microbiome Project Consortium: A framework for human microbiome research. *Nature*. 2012, 486: 215-221. 10.1038/nature11209.

The Statistics of Sequence Similarity Scores: Available at: http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html Accessed Dec.12 2016.

Tringe S.G., Hugenholtz P.: A renaissance for the pioneering 16S rRNA gene. *Current opinion in microbiology*. 2008, 11(5), 442-446.

Turkel E.: Accelerating innovation in HPC. *Proceedings of the ATIP/A* CRC Workshop on Accelerator Technologies for High-Performance Computing: Does Asia Lead the Way?* A* STAR Computational Resource Centre. 2012, p26.

UCLUST algorithm: Available at: http://drive5.com/usearch/manual/uclust_algo.html Accessed Dec.12 2016.

USEARCH manual: Available at: http://www.drive5.com/usearch/manual/cmd_fastx_uniques.html Accessed Dec.12 2016.

Vanwonterghem I., Jensen P.D., Ho D.P., Batstone D.J., Tyson G.W.: Linking microbial community structure, interactions and function in anaerobic digesters using new molecular techniques. *Current Opinion in Biotechnology*. 2014, 27, 55-64.

Venter J.C., Adams M.D., Myers E.W., Li P.L., Mural R.J., Sutton G.G., Smith H.O., Yandell M., Evans C.A., Holt R.A., Gocayne J.D., Amanatides P., Ballew R.M., Huson D.H., Wortman J.R., Zhang Q., Kodira C.D., Zheng X.H., Chen L., Skupski M., Subramanian G., Thomas P.D., Zhang J., Miklos G.L.G., Nelson C., Broder S., Clark A.G., Nadeau J., McKusick V.A., Zinder N., Levine A.J., Roberts R.J., Simon M., Slayman C., Hunkapiller M., Bolanos R., Delcher A., Dew I., Fasulo D., Flanigan M., Florea L., Halpern A., Hannenhalli S., Kravitz S., Levy S., Mobarry C., Reinert K., Remington K., Abu-Threideh J., Beasley E., Biddick K., Bonazzi V., Brandon R., Cargill M., Chandramouliswaran I., Charlab R., Chaturvedi K., Deng Z., Francesco V.D., Dunn P., Eilbeck K., Evangelista C., Gabrielian A.E., Gan W., Ge W., Gong F., Gu Z., Guan P., Heiman T.J., Higgins M.E., Ji R.R., Ke Z., Ketchum K.A., Lai Z., Lei Y., Li Z., Li J., Liang Y., Lin X., Lu F., Merkulov G.V., Milshina N., Moore H.M., Naik A.K., Narayan V.A., Neelam B., Nusskern D., Rusch D.B., Salzberg S., Shao W., Shue B., Sun J., Wang Z.Y., Wang A., Wang X., Wang J., Wei M.H., Wides R., Xiao C., Yan C., Yao A., Ye J., Zhan M., Zhang W., Zhang H., Zhao Q., Zheng L., Zhong F., Zhong W., Zhu S.C., Zhao S., Gilbert D., Baumhueter S., Spier G., Carter C., Cravchik A., Woodage T., Ali F., An H., Awe A., Baldwin D., Baden H., Barnstead M., Barrow I., Beeson K., Busam D., Carver A., Center A., Cheng M.L., Curry L., Danaher S., Davenport L., Desilets R., Dietz S., Dodson K., Doup L., Ferriera S., Garg N., Gluecksmann A., Hart B., Haynes J., Haynes C., Heiner C., Hladun S., Hostin D., Houck J., Howland T., Ibegwam C., Johnson J., Kalush F., Kline L., Koduru S., Love A., Mann F., May D., McCawley S., McIntosh T., McMullen I., Moy M., Moy L., Murphy B., Nelson K., Pfannkoch C., Pratts E., Puri V., Qureshi H., Reardon M., Rodriguez R., Rogers Y.H., Romblad D., Ruhfel B., Scott R., Sitter C., Smallwood M., Stewart E., Strong R., Suh E., Thomas R., Tint N.N., Tse S., Vech C., Wang G., Wetter J., Williams S., Williams M., Windsor S., Winn-Deen E., Wolfe K., Zaveri J., Zaveri K., Abril J.F., Guigo R., Campbell M.J., Sjolander K.V., Karlak B., Kejariwal A., Mi H., Lazareva B., Hatton T., Narechania A., Diemer K., Muruganujan A., Guo N., Sato S., Bafna V., Istrail S., Lippert R., Schwartz R., Walenz B., Yooseph S., Allen D., Basu A., Baxendale J., Blick L., Caminha M., Carnes-Stine J., Caulk P., Chiang Y.H.,

Coyne M., Dahlke C., Mays A.D., Dombroski M., Donnelly M., Ely D., Esparham S., Fosler C., Gire H., Glanowski S., Glasser K., Glodek A., Gorokhov M., Graham K., Gropman B., Harris M., Heil J., Henderson S., Hoover J., Jennings D., Jordan C., Jordan J., Kasha J., Kagan L., Kraft C., Levitsky A., Lewis M., Liu X., Lopez J., Ma D., Majoros W., McDaniel J., Murphy S., Newman M., Nguyen T., Nguyen N., Nodell M., Pan S., Peck J., Peterson M., Rowe W., Sanders R., Scott J., Simpson M., Smith T., Sprague A., Stockwell T., Turner R., Venter E., Wang M., Wen M., Wu D., Wu M., Xia A., Zandieh A., Zhu X.: The sequence of the human genome. *Science*. 2001, 291(5507), 1304-1351.

Vouzis P.D., Sahinidis N.V.: GPU-BLAST: using graphics processors to accelerate protein sequence alignment. *Bioinformatics*. 2011, 27: 182-188. 10.1093/bioinformatics/btq644.

Weber J.L., & Myers E.W.: Human whole-genome shotgun sequencing. *Genome Research*. 1997, 7(5), 401-409.

Weinstock G.M.: Genomic approaches to studying the human microbiota. *Nature*. 2012, 489(7415), 250-256.

Wetterstrand K.A.: DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program(GSP) Available at: https:www.genome.gov/sequencingcostsdata Accessed Dec.8 2016.

Wilke A., Glass E.M., Bartels D., Bischof J., Braithwaite D., D'Souza M., Gerlach W., Harrison T., Keegan K., Matthews H., Kottmann R., Paczian T., Tang W., Trimble W.L., Yilmaz P., Wilkening J., Desai N., Meyer F.: A metagenomics portal for a democratized sequencing world. *Methods in Enzymology*. 2013, 531: 487-523. 10.1016/B978-0-12-407863-5.00022-8.

Wintzingerode F.V., Schattke A., Siddiqui R.A., Rösick U., Göbel U.B., Gross R.: Bordetella petrii sp. nov., isolated from an anaerobic bioreactor, and emended description of the genus Bordetella. *International Journal of Systematic and Evolutionary Microbiology.* 2001, 51(4), 1257-1265.

Xu H.S., Roberts N., Singleton F.L., Attwell R.W., Grimes D.J., Colwell R.R.: Survival and viability of nonculturableEscherichia coli andVibrio cholerae in the estuarine and marine environment. *Microbial Ecology*. 1982, 8(4), 313-323.

Yano M., Mori H., Akiyama Y., Yamada T., Kurokawa K.: CLAST: CUDA implemented large-scale alignment search tool. *BMC Bioinformatics*. 2014, 15(1), 406.

Yarza P., Yilmaz P., Pruesse E., Glockner F.O., Ludwig W., Schleifer K.H., Whitman W.B., Euzeby J., Amann R., Rossello-Mora R.: Uniting the classification of cultured and uncultured bacteria and archaea using 16S rRNA gene sequences. *Nature Reviews Microbiology*. 2014, 12(9), 635-645.

Ye Y., Choi J.H., Tang H.: RAPSearch: a fast protein similarity search tool for short reads. *BMC Bioinformatics*. 2011, 12(1), 1.

Yergeau E., Sanschagrin S., Beaumier D., Greer C.W.: Metagenomic analysis of the bioremediation of diesel-contaminated Canadian high arctic soils. *PLoS ONE*. 2012, 7(1), e30058.

Zhao K., Chu X.: G-BLASTN: accelerating nucleotide alignment by graphics processors. *Bioinformatics*. 2014, 30: 1384-1391. 10.1093/bioinformatics/btu047.

謝辞

本研究を進めるにあたり、指導教員として終始温かいご指導を賜り、援助し続けてくださった東京工業大学大学院の　山田　拓司　准教授、国立遺伝学研究所の　黒川　顕　教授(東京工業大学大学院　特任教授)に心から感謝いたします。また、研究生活を通して懇切丁寧なご指導をしてくださった国立遺伝学研究所の　森　宙史　助教に心から感謝いたします。Saturnの性能試験に関して協力をしてくださった国立遺伝学研究所の　東　光一　博士に心から感謝いたします。特に黒川教授には、私が学部生だった頃からご多忙にも関わらず多くの時間を割いていただきました。重ねて御礼申し上げます。

大学院に入学してから経済的なご支援だけでなく数多くの機会や温かい応援の言葉をくださった東京工業大学情報生命博士教育院の　秋山　泰　教育院長、岡田　知子　事務員をはじめとする皆様に、心から感謝申し上げます。

また、山田・中島研究室の皆様には、毎日様々な面でお世話になりました。心から感謝いたします。

課外時間に精神面で支え続けてくださり様々な知的刺激を与え続けてくださった　田口　了麻　さん、堀江　文晴　さん、吉野　裕太　さん、小泉　亮　さん、高田　正或　さん、石川　秀美　さん、Hildebrand　Janel　さん、Нахумова　Ирина　さん、Berta　Zsofia　さんに、心から感謝申し上げます (名前はすべて姓が先)。

最後になりましたが、私の長い学生生活を、経済的な支援にとどまらず、様々な局面で支えてくださった両親、妹に深く感謝いたします。ありがとうございました。