Vol.2012-MPS-91 No.24
Vol.2012-BIO-32 No.24
2012/12/6

De Bruijn GraphのPreGraphの分割によるVelvetの改良

杉浦 典和[1]　石田 貴士[1]　秋山 泰[1]　関嶋 政和[1]

概要：de Bruijn graph を用いる代表的 *de novo* アセンブラである Velvet は, その消費メモリの多さが課題とされている.Velvet は大きく２つのステップから構成されており, １つ目のステップについてはハッシュテーブルの分割による消費メモリの削減手法が既に提案されている. 本稿では後半のステップで Velvet が作成する１つ目の de Bruijn Graph である PreGraph を分割することで,Velvet の特定の処理の消費メモリを削減した.

キーワード：*de novo* アセンブリ, Velvet, 分散処理, de Bruijn Graph, 次世代シークエンサ

Inprovement of Velvet by dividing PreGraph

Sugiura Norikazu[1]　Ishida Takashi[1]　AkiyamaYutaka[1]　Sekijima Masakazu[1]

Abstract: It is a well-known fact that the memory consumption of Velvet, which is one of the representative *de novo* assembler based on de Bruijn Graph, is too large. Velvet is composed of two steps, and several methods have been already proposed for decreasing the memory consumption of the first step by dividing the hash table. Here we proposed a graph dividing method. By using this method, we have succeeded to decrease the memory consumption of a part of the latter step of Velvet, which makes the PreGraph.

Keywords: *de novo* asesmbly, Velvet, Distrubuted precessing, de Bruijn Graph, Next Generation Sequener

1. はじめに

次世代シークエンサの登場により, 大量のゲノムを高速かつ低価格に読み取ることが可能になった. しかしこのシークエンサは読み取るゲノム断片長配列の長さが数十から数百塩基しかないという欠点を持っているため, 出力された短いゲノム断片配列をアセンブリすることによって元の長いゲノム配列に再構築しなければならない.

シークエンサから得られるゲノム断片配列をリードと呼び, このリードを他のゲノム配列を参照すること無しに元の長いゲノム配列に再構築することを *de novo* アセンブリと言う. またアセンブリを行うツールのことをアセンブラと呼ぶ. 次世代シークエンサの登場以来, 短いリードをアセンブリするため,Velvet[1],ABySS[2] など様々なショートリード向けのアセンブラが開発されてきた. これによって次世代シークエンサが出力するような短いリードでも, おおよそ元の長いゲノム配列にアセンブリすることが可能になってきた.

しかし *de novo* アセンブリでは消費メモリ量と実行時間の長さが課題とされており, 特にヒトゲノムのように大規模なゲノムの *de novo* アセンブリの場合は数テラバイトに及ぶは膨大な量のメモリが必要となることが知られている [3]. そこで本研究では, 代表的なショートリード向けのアセンブラの１つである Velvet に着目し, この消費メモリを削減することを目指す.

Velvet は前半と後半の２ステップから構成されており, 一つ目は velveth, 二つ目は velvetg と呼ばれている.velveth はハッシュテーブルを利用することで, リードのオーバー

[1] 東京工業大学　大学院情報理工学研究科
Tokyo Institute of Technology, Meguro, Tokyo 152-8550. Japan

ラップの情報を記録した ROADMAP というファイルを作成しており, この過程はハッシュテーブルを分割することで既に分散化が実装されている [4][5]. これに対し velvetg は ROADMAP に書かれた情報を元にして de Bruijn Graph[6] を作成しアセンブリを行う.velvetg では, はじめに PreGraph と呼ばれる de Bruijn Graph を作成し, この Graph を元にして再度 de Bruijn Graph を作成している. 本研究では PreGraph の作成過程を分散化することで, 少量のメモリしか搭載していない 1 台の計算機でも大きなゲノムを高速にアセンブリできるように改良することを目指した.

2. PreGraph の分割手法

2.1 PreGraph の作成過程

velvetg において PreGraph は以下のステップで作成される. まず ROADMAP を全て主記憶に読み込み, 作成する PreGraph のバーテックス数を数える. このとき各バーテックスがどのリードから生成されるかという情報を, 主記憶上の ROADMAP に新たに書き込む. 次にこれらの情報に基づきバーテックスとエッジを作成し, 最後にシーケンシングエラーを削除するために短い dead-end パスの削除や一続きのパスの単純化を行う. このようにして PreGraph を作成する.

2.2 PreGraph の分割手法

PreGraph の作成過程を分散化するため分割 PreGraph を作成する. 分割 PreGraph の作成・枝狩りは以下の 4 つのステップで構成した.

まず 1 つ目のステップで, その PreGraph が所持する全てのバーテックスの個数を数える. このバーテックス数を計算ノードで割った商を各分割 PreGraph が持つバーテックスの数とし, 各分割 PreGraph のバーテックスによる消費メモリが均一になるようにする. 1 つ目のステップでは実際にメモリは確保せず, 各分割 PreGraph のバーテックス数を数えてファイルに記録する.

2 つ目のステップでは各バーテックスがどのリードから生成されるかという情報を得る. オリジナルの velvetg ではこの情報を主記憶上の ROADMAP に書き込むが,ROADMAP を全て主記憶に読み込むと消費メモリが削減できないので, 本実装では分割 PreGraph の作成に必要な ROADMAP の行だけを選択的に読み込む. これを用いて分割 PreGraph の各バーテックスがどのリードから生成されるかという情報を獲得し, 中間ファイルに書き込む. この処理は各分割 PreGraph について行う.

3 つ目のステップではステップ 2 で生成したファイルを参考にし, 各分割 PreGraph にを実際にメモリに展開し, また各バーテックスを接続するエッジも作成する. 分割 PreGraph は一つずつメモリに展開され, 解放してから次の分割 PreGraph を展開する. また各分割 PreGraph について, 一本道のパスとなっている複数のバーテックスが存在すれば, これを一つの長いバーテックスに結合しバーテックスの本数を減らす. 各分割 PreGraph は, 解放されるときにそのバーテックスとエッジをファイルに書き出す.

4 つ目のステップでは, ステップ 2 で書き出された分割 PreGraph のファイルを 2 つずつメモリに再展開し, 再びステップ 2 で行ったパスの結合を行う. こうすることで 2 つの分割 PreGraph にまたがるバーテックスも 1 つのバーテックスにまとめることができる. またステップ 3 ではパスの結合のみでなく, 長さが 2k 未満の枝（tip）の削除も行う. この処理を, 変更がなくなるまで何週も, 全ての分割 PreGraph のペアに対して行う.

3. 実験と考察

ここでは分割 PreGraph を利用する手法とオリジナルの velvetg を比較し, 消費メモリと実行時間について比較する. 本実験では 1 代の計算機で順次分割 PreGraph を作成するが,2 章で述べた 2 つ目と 3 つ目のステップのみであれば複数の計算機に分割して実行することも可能である. 実験に使用した計算機は, 東京工業大学が所有する計算機,TSUBAME 2.0 であり, 使用した計算ノードは CPU が Intel Xeon 2.93 GHz (6 cores) × 2, メモリが 54GB である. また実験には NCBI の SRR173084（長さ 51bp のリード約 3337 万本）を用い,k=31 で実行した. また消費メモリと実行時間の測定には, 1 秒おきに ps l コマンドを実行することで消費メモリを求め,velvetg プロセスが存在する時間から実行時間を求めた.（図 1〜図 3）

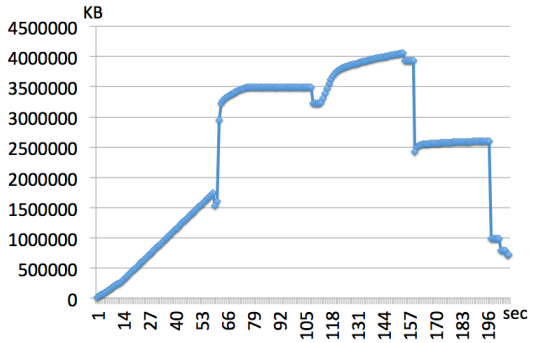

図 1 velvetg の PreGraph 作成までの消費メモリ推移

PreGraph の分割により, 各分割 PreGraph の消費メモリが減少し,PreGraph の作成過程の消費メモリを削減することに成功した. また Velvet の中間ファイルである ROADMAP を選択的に読み込むことにより,RAODMAP のサイズが大きい場合でもピークメモリの削減が可能となった. しかし,2 章の 1 つ目のステップで紹介した全バーテックスを数えるステップで一度 ROADMAP を全て読み込んでおり, この過程が分散かできていないので, 図 3 ではこの過程が消費

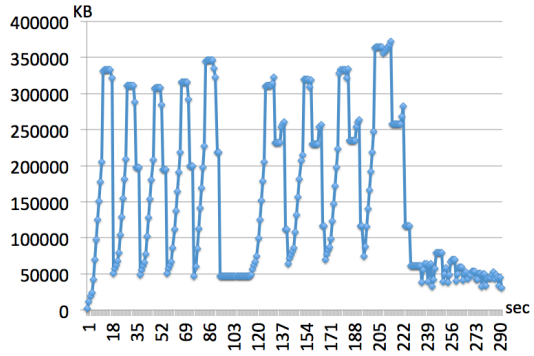

図 2 PreGarph 作成までの消費メモリ推移（４分割）

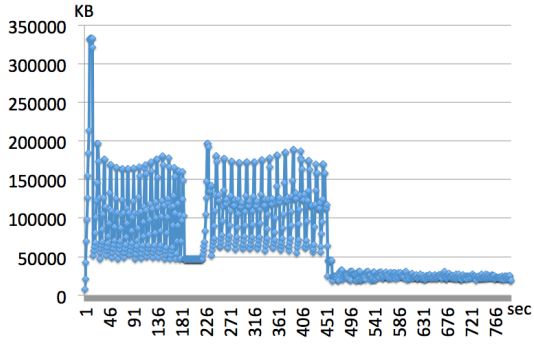

図 3 PreGarph 作成までの消費メモリ推移（１６分割）

メモリのネックになっている. 今後はこの過程の消費メモリの削減が課題である.

また今回の実装では作られる PreGraph の構造がオリジナルの Velvet が出力する PreGraph と比べ若干の差異が見られ,PreGraph の分割数が増えるに従い一致しないバーテックスの本数が増える傾向が見られた（449041 本中, 数本〜数十本）.PreGraph のノードがに差異があれば, この次のステップで生成される２つ目の de Bruijn Graph の構造もやや異なるものになり, 最終的に出力されるコンティグも一致しない可能性がある.Velvet は長く使われてきた信頼性の高いアセンブラなので,Velvet と全く同じ結果を出力できるよう,PreGraph も全く同じ構造のものが作れることが望ましい.

4. 今後の課題

本研究では PreGraph を分割することで消費メモリの削減を試みたが, 最初の全 PreNode 数を数えるステップの消費メモリが削減できていないので, このピークメモリを削減することが重要と言える. また今回の実装では作られる PreGraph の構造がオリジナルの Velvet が出力する PreGraph と比べ若干の差異が見られたため, 各分割 PreGraph から元の PreGraph と完全一致するものを作るための新たなアルゴリズムの提案が必要である.

また今回は PreGraph についてのみ分割を行ったが, 今後は velvetg が扱う２つ目の de Bruijn Graph についても分散化を行い,Velvet 全体の消費メモリを削減する必要がある.

参考文献

[1] Daniel R. Zerbino and Ewan Birney: Velvet: Algorithms for *de novo* short read assembly using de Bruijn graphs, *Genome Res.* 18: 821-829 ,(2008)

[2] Jared T. Simpson, Kim Wong, Shaun D. Jackman, Jacqueline E. Schein, Steven J.M. Jones, and Inanxc Birol, et al.: ABySS: A parallel assembler for short read sequence data, *Genome Res.* 19: 1117-1123,(2009)

[3] Schatz, Michael: Assembly of Large Genomes using Cloud Computing, http://schatzlab.cshl.edu/presentations/2010-07-23.Illumina.pdf, (2010)

[4] 宇治橋善史，成瀬彰，宮本青，重元康，北館智：de novo assembler Velvet の メモリ使用量を削減するプロセス並列手法, *IPSJ SIG technical reports* 2012-BIO-28(9), 1-6, 2012-03-21,(2012)

[5] 杉浦 典和, 石田 貴士, 関嶋 政和, 秋山 泰: ハッシュテーブルの分割による de novo アセンブリの改良, *IPSJ SIG Technical Report.* (2012)

[6] Pevzner, P.A., Tang, H. and Waternam, M.S.: An Eulerian path approach to DNA fragment assembly, *Proc. Natl Acad. Sci. USA*, Vol.14, pp.9748-9753.

【正誤表】
原稿３ページ目

誤

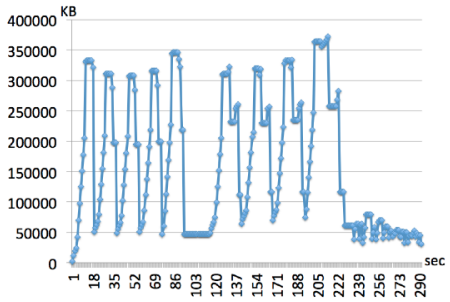

図 **1**　図１　PreGarph 作成までの消費メモリ推移（４分割）

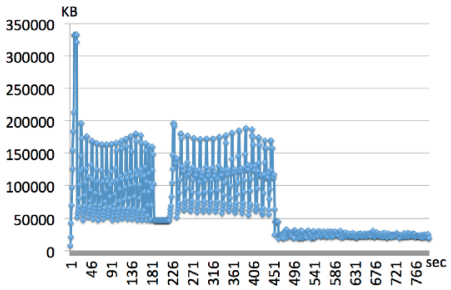

図 **2**　図２　PreGarph 作成までの消費メモリ推移（１６分割）

正

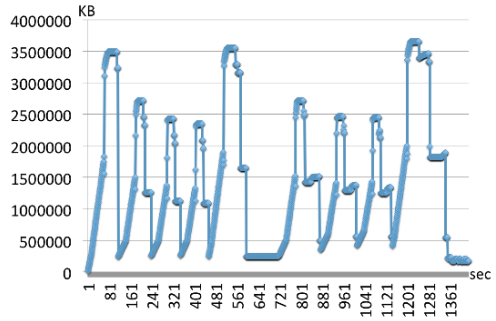

図 **1**　PreGarph 作成までの消費メモリ推移（４分割）

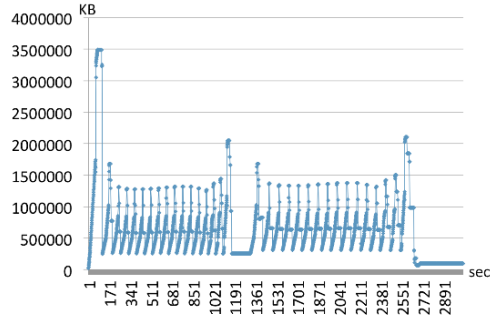

図 **2**　PreGarph 作成までの消費メモリ推移（１６分割）