Improved Consensus Calling for Pacbio data

Rathish Das and Zafar Ahmad

Department of Computer Science, Stony Brook University, NY, USA

Abstract

Motivation: Generating consensus sequence from the subread is one of the most prominent research field in computational biology. The most intuitive way to approach this problem is to perform MSA (Multiple sequence Alignment). There are several techniques of doing MSA. In this paper, we will represent two ways of generating consensus from the pacbio data.

Results: Circular Concensus Sequencing (CCS) is one of the best tool for generating consensus. However, we have implemented Phylogeny based consensus generation and modified pbCCS codebase for analyzing the performance variation. We have introduced some new flags in pbCCS like minCoverage, spanMult, containMult. Our Phylogeny based implementation performance is 83.5% similarity with reference gnome. With the change of new flags pbCCS generated maximum of 87% similarity with the reference gnome.

Availability: Project repository is available in https://github.com/zafarsustbd/Improved-Consensus-for-Pacbio-Data.git

Contact: radas@cs.stonybrook.edu, zafahmad@cs.stonybrook.edu

1 Introduction

Sequence alignment is considered as one of the most important and active research field in computational biology. In traditional pairwise alignment, there could be multiple way to align them. Using scoring can mitigate this problem partially. But, in MSA this problem dominates even more. The partial order alignment graph differs from the alignment strings in that a given base can have multiple predecessors or successors. Aligning a sequence to a DAG introduces surprisingly little complexity to the dynamic programming problem; the clever diagram in the POA paper with a dynamic programming matrix with 3D "bumps" may have had the unintended consequence of making it look more complicated than it is. The primary difference for the purposes of dynamic programming is that while a base in a sequence has exactly one predecessor, a base in a graph can have two or more. Thus, the cursor may have come from one of several previous locations for the same Insert or Align moves being considered; and thus, those scores must be considered too in determining the best previous position.

For generating consensus from the POA graph data we have used two approaches-

1. **Consensus Using CCS**,
2. **Consensus Using Phylogeny**.

In the next few section, we will define the problem and show the methodology our workflow.

2 Methods

2.1 Consensus generation using CCS

We used Unanimity project to make consensus sequence from reads. Following steps are required for consensus generation with our modification.

2.1.1 Downloading and compiling modified Unanimity

We have provided the executable Unanimity in our github repository. But if anyone wants to compile the CCS manually, then please execute the following terminal commands.

```
git clone https://github.com/PacificBiosciences/unanimity
cd unanimity
git submodule update --init –remote
```

Delete previous files,

```
> rm -f ~/unanimity/src/ConsensusSettings.cpp
> rm -f ~/unanimity/src/poa/PoaGraphTraversals.cpp
> rm -f  ~/unanimity/include/pacbio/ccs/Consensus.h
> rm -f ~/unanimity/src/main/ccs.cpp
```

Copy the modified files,

```
> cp ~/Improved-Consensus-for-Pacbio-Data/ConsensusUsingCCS/UnanimityChanges/ConsensusSettings.cpp  ~/unanimity/src/
> cp ~/PoaGraphTraversals.cpp  ~/unanimity/src/poa/
> cp ~/Consensus.h  ~/unanimity/include/pacbio/ccs/
> cp ~/ccs.cpp  ~/unanimity/src/main/
```

2.1.2 Filtering the Input BAM file

The input BAM file has size more than 9GB. It also contains some ID with only 1 or 2 reads. Hence we first filtered out those IDs which has at least 5 reads.

First, we cut the first 6 lines (BAM header) from the original BAM file. Then we chose a ID which has more than 5 reads and copied its read string. We have attached such a BAM file with name 7864612.bam in the uploaded folder.

2.1.3 **Issues Resolved in Unanimity Build in Development mode**

There was some compiling issue while making unanimity. It was not building libhts.a as dependency Below, the screenshot is given.

```
rathishdas@rathishdas-HP-ZBook-14:~/CompBio/newccs/unanimity/build$ make ccs
[  9%] Built target unanimity
[ 19%] Built target pbcopper
[ 70%] Built target pbbam
[ 98%] Built target cc2
make[3]: *** No rule to make target 'external/pbbam/build/external/htslib/libhts.a', needed by 'ccs'.  Stop.
CMakeFiles/Makefile2:453: recipe for target 'src/CMakeFiles/ccs.dir/all' failed
make[2]: *** [src/CMakeFiles/ccs.dir/all] Error 2
CMakeFiles/Makefile2:465: recipe for target 'src/CMakeFiles/ccs.dir/rule' failed
make[1]: *** [src/CMakeFiles/ccs.dir/rule] Error 2
Makefile:227: recipe for target 'ccs' failed
make: *** [ccs] Error 2
rathishdas@rathishdas-HP-ZBook-14:~/CompBio/newccs/unanimity/build$ ls external/pbbam/build/external/htslib/
CMakeFiles/         cmake_install.cmake  Makefile
rathishdas@rathishdas-HP-ZBook-14:
```

Resolution:

> cd ~/unanimity/build/external/pbbam/build/external/htslib/
> make

(This will create libhts.a in ~/unanimity/build/external/pbbam/build/external/htslib/)

> cd ~/unanimity
> make ccs

2.1.4 Change Made in CCS Sourcecode:

We have tried to come up a better scoring function for the node in the POA graph.

2.1.5 Change 1

Currently, in CCS, a score is assigned to a node based on how many read it represent.

score(v) = (2 * numReads[v] - coverage[v] - epsilon)

Where, numReads[v] denotes how many reads node *v* represents. The authors have implemented coverage using "tagSpan" approach where, when each read is added, they increment the spanning coverage for all vertices "covered" by the read.

The idea here is that when a vertex represents less than half of reads covered at its position, its score becomes negative and hence reducing its chance of inclusion in the POA consensus.

```
rathishdas@rathishdas-HP-ZBook-14:~/CompBio/newccs/unanimity/build$ ./ccs --he
Usage: ccs [options] INPUT OUTPUT
Generate circular consensus sequences (ccs) from subreads.

Options:
  -h,--help                   Output this help.
  --log-level,--logLevel      Set log level. ["INFO"]
  --version                   Output version info.
  --force                     Overwrite OUTPUT file if present.
  --zmws                      Generate CCS for the provided comma-separated hole
  --maxLength                 Maximum length of subreads to use for generating C
  --minLength                 Minimum length of subreads to use for generating C
  --minPasses                 Minimum number of subreads required to generate CC
  --minPredictedAccuracy      Minimum predicted accuracy in [0, 1]. [0.9]
  --minZScore                 Minimum z-score to use a subread. NaN disables thi
  --maxDropFraction           Maximum fraction of subreads that can be dropped b
  --minSnr                    Minimum SNR of input subreads. [3.75]
  --minReadScore              Minimum read score of input subreads. [0.75]
  --byStrand                  Generate a consensus for each strand.
  --noPolish                  Only output the initial template derived from the
  --polish                    Emit high-accuracy CCS sequences polished using th
  --richQVs                   Emit dq, iq, and sq "rich" quality tracks.
  --reportFile                Where to write the results report. ["ccs_report.tx
  --modelPath                 Path to a model file or directory containing model
  --modelSpec                 Name of chemistry or model to use, overriding defa
  --numThreads                Number of threads to use, 0 means autodetection. [
  --logFile                   Log to a file, instead of STDERR.
  --containMult               Containing reads counter multiplier used in scorin
  --spanMult                  Containing reads counter multiplier used in scorin
  --minCoverage               Minimum coverage for a node [1]
  --emit-tool-contract        Emit tool contract.
  --resolved-tool-contract    Use args from resolved tool contract.

Arguments:
  input                       Input file.
  output                      Output file.

rathishdas@rathishdas-HP-ZBook-14:~/CompBio/newccs/unanimity/build$
```

However, instead of making this as constant (here ½ of the spanning coverage), we can tune this parameter and see how it makes the consensus.

We here introduced 2 new flags namely "containMult" and "spanMult" by changing the source code of CCS. Users can give as input these multiplier. **[Options in red box are introduced new.]**

Now the score function becomes as follows:

score(v) = (containMult * numReads[v] - spanMult * coverage[v] - epsilon)

Files have been changed:

a) ~/unanimity/include/pacbio/ccs/Consensus.h
b) ~/unanimity/src/ConsensusSettings.cpp
c) ~/unanimity/src/main/ccs.cpp
d) ~/unanimity/src/poa/PoaGraphTraversals.cpp

```
rathishdas@rathishdas-HP-ZBook-14:~/CompBio/pitchfork/deployment
build/result/test_2_1_0.bam ../../../../data/reference\ data/Ara
[INFO] 2016-12-15T02:47:08 [blasr] started.
    nMatch: 1564
 nMisMatch: 63
      nIns: 114
      nDel: 56
      %sim: 87.0339
     Score: -6592
         Query: m54113_160913_184949/7864612/ccs/0_3016
        Target: 2
         Model: a hybrid of global/local non-affine alignment
     Raw score: -6592
        Map QV: 254
  Query strand: 0
 Target strand: 1
   QueryRange: 11 -> 1752 of 3016
  TargetRange: 10752497 -> 10754180 of 19698289
       11  AGGAAGCAGTAG--G---CACAAGCTTTAAAACGAAATTTCC-AGCATA-
           |||||||||*||  |   ||||||||||  *|||  |||||| ||||*|
10752497  AGGAAGCAGAAGCAGTATCACAAGCTTT--GACG--ATTTCCAAGCAGAG

       54  GGAAAGATCAATT-GAGTTCCCAGTAACACGGATAAGCTGCTCCAAGAAG
           ||||||||| *|| |*|||*|||||||| |*|||||||| || *||| ||
10752543  GGAAAGATC-GTTGGCGTTGCCAGTAAC-CTGATAAGCT-CT-AAAG-AG
```

2.1.6 Analysis for Change 1 (Scoring Function Update)

When we made containMult = 2 and spanMult =1 then, using blasr we get 87% similarity.

> ./ccs --noPolish --containMult=2 --spanMult=1 –minCover

age=0 ../../../../data/7864612.bam result/test_1_1_0.bam

However, when we make both containMult and spanMult =1 , then we got 82.5% similarity using blasr.

./ccs --noPolish --containMult=1 --spanMult=1 --minCoverage=0 ../../../../data/7864612.bam result/test_1_1_0.bam

The reason behind this : when we are making both containMult and spanMult same, we are penalizing a vertex more to be included in POA consensus if it's containing count is not same as spanning count. Hence the consensus string becomes relatively short when both containMult and spanMult are same.

For other values (containMult, spanMult) like (3,2), (5,3) we don't get long consensus hence blaser doesn't give any output while comparing with reference genome.

2.1.7 Change 2 (MinCoverage update)

In the unanimity README, it is mentioned as follows,

"The value of minCoverage is unclear---it was unclear why it was added, why it was deemed necessary. It might be helpful if we could use it to completely replace the local coverage calculation. We need to do some experiments on extensive datasets to see how, in practice, this affects observed truncations in HLA datasets."

Hence, we thought to do experiment with different minCoverage value. For this purpose, we make the minCoverage as a flag and user can set minCoverage as part of input.

Files have been changed:

a) ~/unanimity/include/pacbio/ccs/Consensus.h
b) ~/unanimity/src/ConsensusSettings.cpp
c) ~/unanimity/src/main/ccs.cpp
d) ~/unanimity/src/poa/PoaGraphTraversals.cpp

2.1.8 Analysis of Change 2

We have seen that minCoverage value works for two non-negative integer values, namely 0, 1. Also any values \in [0,2) will work.

When we tried with minCoverage 2, we could not able to generate any consensus string.

Any negative value doesn't make difference, because in the score function we take max(spanningReads, minCoverageArg) to calculate the function and spanningReads is always greater than zero.

```
float score =
        (mode != AlignMode::GLOBAL)
            ? (containMult * containingReads - spanningMult *
std::max(spanningReads, minCoverageArg) - 0.0001f)
            : (containMult * containingReads - spanningMult *
totalReads - 0.0001f);
```

```
rathishdas@rathishdas-HP-ZBook-14:~/CompBio/pitchfork/deployme
uild/result/test_1_1_0.bam ../../../../data/reference\ data/Ar
[INFO] 2016-12-15T02:58:04 [blasr] started.
    nMatch: 1099
 nMisMatch: 72
      nIns: 117
      nDel: 43
     %sim: 82.5695
    Score: -4263
        Query: m54113_160913_184949/7864612/ccs/0_1288
       Target: 2
        Model: a hybrid of global/local non-affine alignment
    Raw score: -4263
       Map QV: 254
  Query strand: 0
 Target strand: 1
   QueryRange: 0 -> 1288 of 1288
  TargetRange: 10751230 -> 10752444 of 19698289
       0  TGTTATCGATCAAGGATGTTTGAAAGCTCTTAA-CTGGGGAACCACTGGC
          ||||||||||||| ||*|||| |||||||*||| || ||||| | ||||
10751230  TGTTATCGATCAA-GAAGTTT-AAAGCTCGTAACCT-GGGAA-C-CTGG-
```

Other attempts:

We also tried to change the scoring function as folllows:

If (containing[v] == Spanning[v])
  Then, score(v) = score(v) *2;

But, this did not improve the accuracy of consensus sequence

2.1.9 All commands to run CCS and aligning to reference genome

Running CCS:

./ccs --noPolish --containMult=2 --spanMult=1 --minCoverage=0 <input bam file> <output bam file>

samtools view -h <output bam file>

Running Blasr:

blasr <new consensus.bam> <reference consensus.fa> -m 0

In our experiments, we have used **Arabidopsis_thaliana.TAIR10.dna.chromosome.1.fa** as reference genome.

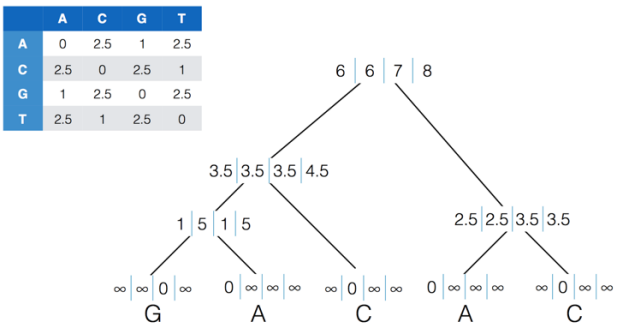
2.2 Consensus generation using Phylogeny

We will use **Clustal** formatted aligned file for generating the consensus sequence. The details of generating the Clustal file will be given below. So now we can represent our clustal columns as small phylogeny problem.

Given: a set of characters at the leaves, a set of states for each character, the cost of transition from each state to every other, and the topology of the phylogenetic tree.

Find: a labeling for each internal node that minimizes the overall cost of transitions.

One way to define the lowest cost set of transitions is to maximize parsimony. Using Sankoff algorithm we can generate consensus character for each column of the aligned reads.



2.2.1 Data Preparation:

a) Using samtools read the sequences of desired id number and save them.
   samtools view -h m54113_160913_184949.subreads_sorted.bam | head -20000 | grep 7864612 | awk '{print $10}' > 7864612_tmp.fa
   Here,
   "M54113_160913_184949.subreads_sorted.bam" is ID sorted input '.bam' file,
   "7864612" is desired id number,
   "7864612_tmp.fa" is the output file

b) Convert that temporary file (i.e: "7864612_tmp.fa") to a '.fa' file using our "read_c_fa.cpp" file
   ./read_c_fa 7864612_tmp.fa 7864612.fa
   Here,
   "7864612_tmp.fa" is input file

"7864612.fa" is output file

2.2.2 Generating Clustal alignment

a) Please extract "poaV2.tar.gz" library or download the c implementation of POA library from here, https://sourceforge.net/projects/poamsa/
b) Compile the POA library via 'make poa' inside the directory
c) Generate the Clustal file using 'poa' executable.
./poa -read_fasta ../7864612.fa -clustal \\
../7864612_clustal.aln blosum80.mat -tolower
Here,
"../7864612.fa" is input file
"../7864612_clustal.aln" is output file

2.2.3 Generating the Consensus

Use our '.cpp' implementation (i.e: "clustal_to_consensus_parsimony.cpp") for generating the consensus.
./clustal_to_consensus_parsimony 7864612_clustal.aln
7864612_clustal_consensus.fa
Here,
"7864612_clustal.aln" is clustal input file
"7864612_clustal_consensus.fa" is final consensus fasta file

Note: Please make sure that penalty matrix file (i.e: "score_matrix.txt") is in the same directory of "./clustal_to_consensus_parsimony" executable.

2.2.4 Aligning with blasr

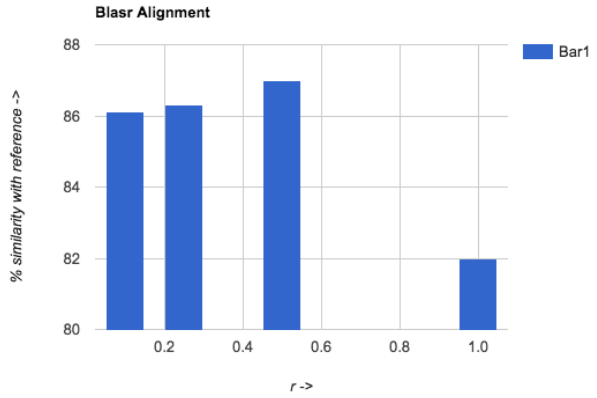
You can align our consensus with reference gnome with blasr.
./blasr 7864612_clustal_consensus_1.fa Arabidopsis_thaliana.TAIR10.dna.chromosome.1.fa -m 0
Here,
"7864612_clustal_consensus_1.fa" is final consensus output of our algorithm.

"**Arabidopsis_thaliana.TAIR10.dna.chromosome.1.fa**" is the reference gnome that we aligned with our output consensus.

3 Results

3.1 Consensus using CCS

We defined our scoring function as
score(v) = (containMult * numReads[v] - spanMult * coverage[v] - epsilon)
Spanning coverage r = (SpanMult / ContainMult)
Finally we evaluated our output by aligning with the reference genome using Blasr.



3.2 Consensus using Phylogeny

We have tried different penalty matrix in our "**score_matrix.txt"** and finally concluded that the best alignment is generated using unit cost matrix. We achieved 83% of similarity with the reference genome.
Blasr aligned output sample:

```
[INFO] 2016-12-15T00:51:25 [blasr] started.
    nMatch: 2826
 nMisMatch: 159
      nIns: 313
      nDel: 90
      %sim: 83.412
     Score: -11161
        Query: 7864612_clustal_consensus_1.fa/0_15787
       Target: 2
        Model: a hybrid of global/local non-affine alignment
    Raw score: -11161
       Map QV: 254
  Query strand: 0
 Target strand: 1
   QueryRange: 12293 -> 15591 of 15787
  TargetRange: 10748084 -> 10751159 of 19698289
   12293  TGTGGAACACA--GAGGG-AAGCTCCATCAGCAGCCACCGTGTGATTCCC
          |||||||||||  | ||| ||||||||||||||||**|||||||| *|
10748084  TGTGGAACACATTG-GGGAAAGCTCCATCAGCAGCCGTCGTGTGATT-TC

   12340  CCCGAGTCCCCCCCACCCCACCCCCCGGGACCTCACGGGTA-CGTATTTT
          |||*|||         |*||        | ||||||*|||| *||| |||
10748132  CCCAAGT---------CGCA--------G-CCTCACTGGTATTGTA-TTT

   12389  TCCCAGCCCAGAGGGAAGTGATTTCTTTTGTTTGAGGTTTGACCGCATTA
          |||||*|||| | |||||||||||| *|||||||| |||||||**||||||
```

4 Acknowledgements

We wish to thank **Avi Srivastava** for his helpful discussions and comments on this work.

5 References

Lee, Grasso, and Sharlow (2002) Multiple sequence alignment using partial order graphs, Bioinformatics.
Lee (2003) Generating consensus sequences from partial order multiple sequence alignment graphs, Bioinformatics.
Grasso and Lee (2004) Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems, Bioinformatics.
Simpson Lab of the Ontario Institute for Cancer Research.
Small phylogeny image is collected from Lecture Slide.