BIOINFORMATICS ORIGINAL PAPER

Vol. 24 no. 1 2008, pages 42–45
doi:10.1093/bioinformatics/btm542

Sequence analysis

Assembly reconciliation

Aleksey V. Zimin[1,*], Douglas R. Smith[2], Granger Sutton[3] and James A. Yorke[1]

[1]IPST, University of Maryland, College Park, [2]Agencourt Bioscience Inc., Beverly, MA and [3]The J. Craig Venter Instutute, Rockville, MD, USA

Received on August 7, 2007; revised on October 15, 2007; accepted on October 22, 2007

Advance Access publication December 5, 2007

Associate Editor: Alex Bateman

ABSTRACT

Motivation: Many genomes are sequenced by a collaboration of several centers, and then each center produces an assembly using their own assembly software. The collaborators then pick the draft assembly that they judge to be the best and the information contained in the other assemblies is usually not used.

Methods: We have developed a technique that we call *assembly reconciliation* that can merge draft genome assemblies. It takes one draft assembly, detects apparent errors, and, when possible, patches the problem areas using pieces from alternative draft assemblies. It also closes gaps in places where one of the alternative assemblies has spanned the gap correctly.

Results: Using the Assembly Reconciliation technique, we produced reconciled assemblies of six Drosophila species in collaboration with Agencourt Bioscience and The J. Craig Venter Institute. These assemblies are now the official (CAF1) assemblies used for analysis. We also produced a reconciled assembly of Rhesus Macaque genome, and this assembly is available from our website http://www.genome.umd.edu.

Availability: The reconciliation software is available for download from http://www.genome.umd.edu/software.htm

Contact: alekseyz@ipst.umd.edu

1 INTRODUCTION

Draft genome assemblies have misassemblies and gaps. Many genomes (e.g. mouse, several species of Drosophila and Rhesus Macaque) are sequenced by several centers, and then assembled using two or more assembly programs. In the end, the collaborators pick the draft assembly that they judge to be the best. Most major assembly programs such as Arachne (Batzoglou *et al.*, 2002, Jaffe *et al.*, 2003, Vinson *et al.*, 2005), PCAP (Huang *et al.*, 2003), Phusion (Mullikin and Ning, 2003), JAZZ and Celera Assembler (Myers *et al.*, 2000) are similar in that they use the variations on the traditional overlap, layout, consensus approach. The details of the techniques used by different assembly programs differ, and frequently one assembly program is able to properly assemble a difficult region of the genome, while the other ones cannot.

The major kind of misassembly in the contigs found in draft genomes is the omission of one or more copies of repetitive sequence, and, more generally, the loss of the unique chunks of sequence that are surrounded by copies of a repeat along with one of the repeat copies. Occasionally assemblers err by including extra sequence in an assembly, but such 'expansion' errors are less common.

We used Nucmer (Delcher *et al.*, 1999, 2002; Kurtz *et al.*, 2004) to align the two assemblies of *Drosophila willistoni* produced by using two different assembly programs from the same data. We used the draft assemblies produced by two major assembly programs: Celera Assembler and Arachne. We aligned the contigs of the two assemblies and looked for cases, where it is evident that one assembly was missing a chunk of sequence that was present in the other one. We call these discrepancies 'compression misassemblies' or simply 'compressions'. In each case of compression there are two possibilities: (i) one of the assemblies is correct, or (ii) both are wrong, so each compression counts as a misassembly. Figure 1 illustrates how we identified compressions by analyzing the alignments of the contigs from the two assemblies. We only counted compressions within contigs that were at least 1000 bases away from the ends of the contigs. We did not use any scaffold information for this analysis. These compressions are not due to polymorphisms, which can be verified by looking at the insert size statistics for inserts spanning the missing regions: usually all inserts are uniformly short, which would not be true for polymorphisms, where one would expect a bimodal distribution of lengths. Table 1 summarizes the results, showing that there are about 1.15 million bases in compressions between the two assemblies. Furthermore, these errors are distributed quite uniformly along the contig sequences and they are not concentrated in the regions of the centromeres and telomeres (which are generally not in either of the assemblies). These errors change distances between genes and regulatory regions; therefore they are biologically significant. The omissions may also contain regulatory sequences or even coding sequences. This kind of a comparison was a major motivation to develop techniques for merging assemblies.

Another way to compare two assemblies of the same species is to count how many bases in contigs of a draft assembly align to the contigs of an alternative draft assembly. We performed the comparison using the two assemblies of *D.willistoni* mentioned above. We used Nucmer with default settings and considered all alignments with 98% or higher identity. We found that out of 223 M total bases in the contigs of the assembly A, 12 M bases did not align to assembly B. Vice versa,

*To whom correspondence should be addressed.

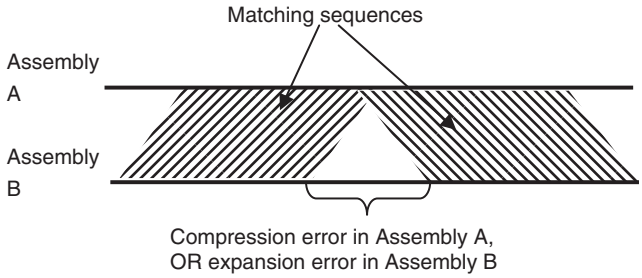

Fig. 1. Identifying a compression by aligning draft assemblies A and B.

Table 1. Compression misassemblies detected in the two alternative draft assemblies of *D.willistoni*

Contig compressions in assembly A compared to assembly B:		
Compression size	100 or greater	1,000 or greater
Number of compressions	157	71
Number of bases in compressions	540 539	512 722
Contig compressions in assembly B compared to assembly A:		
Compression size	100 or greater	1,000 or greater
Number of compressions	348	136
Number of bases in compressions	615 420	538 659

out of 229 M total bases in assembly B, 7 M bases did not align to the assembly A. Adding up these numbers give 19 M bases of differences or ~8.5% of the total bases.

In this article, we describe an *Assembly Reconciliation* technique that can merge draft genome assemblies. In a nutshell, Assembly Reconciliation takes an original draft assembly, detects apparent errors, and, when possible, patches the problem areas using pieces from one or more alternative draft assemblies. It also closes gaps in scaffolds where the alternative assembly has spanned the gap correctly. Several alternative assemblies can be used to incrementally improve the original assembly. Based on this technique, we developed software that improves a 'reference' assembly using alternative assemblies of the same read data. The improved reference, or *reconciled* assembly is produced, with fewer gaps and misassemblies. Our software works for genomes of up to 4 GB in size. In what follows we list the concepts on which the software is based and the results of our recent work on seven fruit fly genomes.

2 METHODS

Reconciliation is based on two methods: detecting misassemblies and closing gaps. We currently use the 'CE statistic' to detect compression/expansion misassemblies and to verify the validity of gap spanning. We briefly review the concept of the CE statistic.

For a given location in a draft assembly, we examine the sample of inserts from a given library that span this location in the genome. Using the read placement coordinates from the assembly, we compute the mean M of the implied insert lengths l_i for the sample. More precisely, if N is the local coverage, or the number of inserts in a sample, and the lengths are denoted $l_1, \ldots, l_N$, then

$$M = \frac{1}{N}\sum_{i=1}^{N} l_i.$$

The CE statistic is based on the fact that the variance of the sum of independent random variables is equal to the sum of their variances. The statistic measures the distance between the mean M of the local sample and the library mean μ in the units of expected sample standard deviation $\sigma/\sqrt{N}$. We define the value of the CE statistic Z as

$$Z = \frac{M-\mu}{\sigma/\sqrt{N}}.$$

Large negative Z implies that the sample of inserts is compressed, thus it is possible that there was an omission of a chunk of sequence in the region spanned by the inserts in the sample. Likewise, large positive Z indicates that a chunk of sequence may have been erroneously inserted. The distribution of the insert lengths in the library is in general not normal, but for sufficiently large sample size (coverage) N we can approximate the distribution of M (and therefore Z) by the normal distribution due to Central Limit Theorem. For any $Z_0 > 0$ and sample size N, we can compute the probability that a value of $|Z| \geq Z_0$ occurs at random. Setting the threshold for problem detection at $Z_0 = 3.3$, when N is large, detects events that have approximately 0.001 probability to occur at random. For smaller values of N, we determine the cutoff value for each N.

The algorithm that we currently use to reconcile two assemblies (called the *reference* assembly and the *supplementary* assembly) is as follows (see also Figure 2 for illustration).

(1) **Create gaps.** We first compute the CE statistic on the reference assembly, and find all locations in the assembly where the absolute value of the CE statistic is larger than the threshold. We then break the assembly at these locations. We introduce positive gaps in the sequence for the compressions and negative gaps for expansions, creating a gapped reference assembly. We also separate the read multi-alignment according to the gap in the sequence.

(2) **Align sequences.** We next align the gapped reference assembly to the supplementary assembly using Nucmer. The Nucmer settings are modified to only use seeds that are unique both in reference and query sequences and to require the minimum length of the cluster of matches of 400 to avoid short repeat-induced matches (see Nucmer documentation at http://mummer.sourceforge.net/).

(3) **Identify possible gap closures.** After that we use the alignment to find out which contigs in the supplementary assembly span the intra-scaffold gaps of the reference assembly (both pre-existing gaps and gaps introduced in step 1) such that (i) the orientation of the alignments is correct; (ii) the gap size with respect to the alignment is within 3 reported SDs of the reported scaffold gap size and (iii) the absolute value of the CE statistic in the supplementary assembly over the closure region is less than 3.3. This generates a list of candidate gap closures.

(4) **Find read placements for closed gaps.** We use the candidate gap closures based on the sequence alignments from the previous step and examine the reads in the gapped reference assembly placed on both sides of the gaps, which cover the portions of the contigs that aligned the supplementary assembly. We look for two *anchor* reads on both sides of the gap that are placed at the same relative location in the supplementary assembly. We then take the set of reads from the supplementary assembly that are located between the two anchor reads and insert it into the reference, closing the gap. We extract from the supplementary assembly the sequence

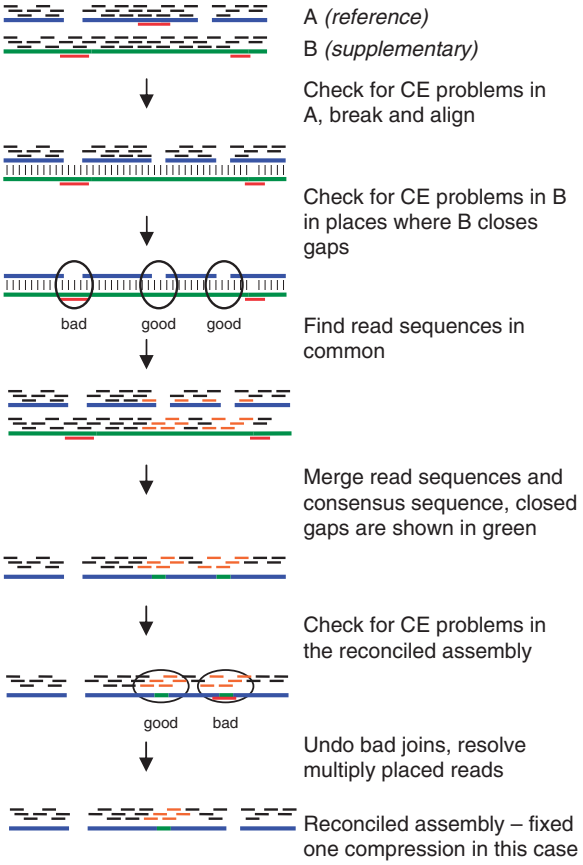

Fig. 2. Illustration of the assembly reconciliation process. Underlined red regions are the CE problems. Reads are shown above the blue and green lines representing the consensus sequence. Assembly A is the reference assembly. Assembly B remains unmodified.

that spans the gap. We then insert that sequence into the gap of the reference assembly.

(5) **Validate gap closures.** Using the newly placed reads, we then compute the CE statistic on the updated reference assembly and find out which gap closures resulted in compressions or expansions. If a gap was an intra-scaffold gap and closing it resulted in a compression or expansion, we undo the closure and return the reference assembly to its initial state. If a gap is introduced as a result of the compression or expansion in the reference and closing it results in a compression or expansion, we return the reference to its initial state. Thus, we only keep the gap closures that have proper CE statistic values over the closure region.

(6) **Resolve multiply placed reads.** Finally, we resolve the problem of the multiply placed reads using mate pairs. If a read is placed twice, we look for the placement of its mate and choose the placement that is most consistent with the mate. If the mate is not placed, or if neither placement is better, we choose the placement at random, making sure we do not create gaps in coverage. The final read placements do not affect the sequence of the reconciled assembly.

The algorithm is not symmetric with respect to the assemblies, because it only closes the gaps within scaffolds of the reference assembly. The reconciled assembly's scaffolds are almost identical in sizes to the scaffolds of the reference assembly; the only difference is that there are fewer gaps in them.

The only possible scenario for improperly closed gap would be if the supplementary assembly closed a gap incorrectly, and the misassembly in the supplementary assembly is so small that it is indetectable by the mate pair placement statistic. We should also mention that if both initial assemblies misassembled a region, then the reconciled assembly will also contain a misassembly in the same region.

Since reconciliation uses insert size statistics for detecting errors, the algorithm's ability to detect (and correct) misassemblies depends on the insert coverage in the assemblies. Even if the read coverage is relatively small but the inserts are large, the software will perform well.

The algorithm is currently coded in PERL, and it takes 2 h to run on a single 2.4 GHz Opteron processor to reconcile two fly genome assemblies of ~200 MB each. The majority of the run time is spent on running the assembly alignment.

The reconciliation software also creates a list of locations in the assembly that are likely to be misassembled, even when it was unable to fix them.

We note that often, in creating its best possible assembly, a team will often make numerous runs using different settings in the assembly software. Reconciliation can be used to combine the results of the different runs. For example, it may be useful to create a reference assembly with conservative settings and use more aggressive assembly as supplementary to close gaps and thus increase the contig sizes.

Finally, we note that intra-scaffold gaps and expansion/compression errors are not the only kinds of deficiencies in the draft assemblies. Rearrangements are also common, but they are not addressed by our algorithm. Also future versions of the software will address the question of filling inter-scaffold gaps and possibly use an additional set of techniques for finding errors.

3 RESULTS

To test the Assembly Reconciliation software, we applied it to the two draft assemblies of *Wolbachia pipientis wMel*. We chose as reference the assembly produced by TIGR. The supplementary assembly was produced by our group (UMD). Both assemblies were created using Celera Assembler with the only difference that UMD assembly used the read overlaps generated by the UMD overlapper (Roberts, 2004) and TIGR assembly used the overlaps generated by Celera overlapper. Bacterial genomes generally lack the complex repeat structure present in the genomes of larger multicellular organisms, and the reconciliation software did not locate any compression misassemblies in the TIGR assembly. Reconciliation closed four gaps. We checked the closures by aligning the modified (merged) contigs to the finished sequence. These contigs aligned perfectly. Other contigs, of course were unmodified.

We have applied the reconciliation software to assemblies of eight *Drosophila* species: *yakuba*, *virilis*, *grimshawi*, *erecta*, *willistoni*, *ananassae*, *mojavensis* and *pseudoobscura*. The reconciliation results are listed in Table 2. The first column in Table 2 lists the fly species and the assemblies used for reconciliation. All Agencourt assemblies were produced using Arachne assembler. All J. Craig Venter Institute (VI) and TIGR assemblies were produced using Celera Assembler. Washington University (WashU) assembly of *D.yakuba* was produced using PCAP, and University of Maryland (UMD) assembly of *D.virilis* was produced using Celera Assembler

Table 2. Assembly reconciliation results for seven Drosophila species

Species	Before reconciliation		After reconciliation	
	Contig N50	CE problems	Contig N50	CE problems
D.virilis (Agencourt+VI + UMD)	101 kb	1566	118 kb (+17%)	**1094 (−30%)**
D.yakuba (WashU + Agencourt + TIGR)	116 kb	954	**164 kb (+41%)**	685 (−28%)
D. willistoni (VI + Agencourt)	145 kb	1058	165 kb (+14%)	893 (−16%)
D.grimshawi (Agencourt + VI)	78 kb	1010	91 kb (+17%)	936 (−7%)
D.erecta (Agencourt + VI)	366 kb	798	448 kb (+22%)	645 (−19%)
D.ananassae (Agencourt + VI)	83 kb	2206	93 kb (+12%)	1903 (−14%)
D.mojavensis (Agencourt + VI)	100 kb	1045	121 k (+21%)	841 (−20%)
D.pseudoobscura (Agencourt + TIGR)	97 kb	369	103 kb (+6%)	287 (−22%)

Contig N50 denotes the size of the largest contig for which at least half of the bases in the assembly are in contigs of that size or larger. Bold font highlights the largest improvement. For each fly the assemblies used are listed in parentheses, and the first in each is the reference assembly.

with read overlaps produced by the UMD overlapper. Reconciliation used the assemblies in the order given in the table. For example for *D.ananassae*, the Agencourt assembly was the reference and VI assembly was supplementary. For each fly Agencourt Bioscience and J. Craig Venter Institute chose the reference assembly based on the objective assembly statistics—generally contig and scaffold sizes. 'Before reconciliation' column in the Table 2 gives the statistics of the reference assemblies.

These results show that assemblies can be improved significantly using assembly reconciliation; at a minimum, many of the compression problems—which represent erroneous deletions—can be fixed. In every reconciled assembly the contigs get larger, i.e. the N50 contig size increased compared to the reference draft. That constitutes significant improvement of the reference draft genome. We observed that the greatest improvements in the assembly contig statistics and the number of CE problems were achieved when we reconciled three assemblies produced by three different centers (*D.virilis* and *D.yakuba* assemblies). Thus using more assemblies seems to be better, at least in this small sample. All except two (*D.yakuba* and *D.pseudoobscura*) of the assemblies shown in the table are now the official versions and are posted on the Flybase website at http://www.flybase.org/docs/news/DrosTimelinesStatusMar06.htm. The assembly reconciliation software is available at http://www.genome.umd.edu/software.htm.

ACKNOWLEDGEMENTS

This work was supported under NSF grant DMS0616585, and under NIH Grant 1R01HG0294501.

Conflict of Interest: none declared.

REFERENCES

Batzoglou,S. *et al.* (2002) ARACHNE: a whole-genome shotgun assembler. *Genome Res.*, **12**, 177–189.

Delcher,A.L. *et al.* (1999) Alignment of whole genomes. *Nucleic Acids Res.*, **27**, 2369–2376.

Delcher,A.L. *et al.* (2002) Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.*, **30**, 2478–2483.

Dew,I.M. *et al.* (2005) A tool for analyzing mate pairs in assemblies (TAMPA). *J. Comput. Biol.*, **12**, 497–513.

Jaffe,D.B. *et al.* (2003) Whole-genome sequence assembly for mammalian genomes: Arachne 2. *Genome Res.*, **13**, 91–96.

Huang,X. *et al.* (2003) PCAP: a whole-genome assembly program. *Genome Res.*, **13**, 2164–2170.

Kurtz,S. *et al.* (2003) Versatile and open software for comparing large genomes. *Genome Biol.*, **5**, R12.

Mullikin,J.C. and Ning,Z. (2002) The phusion assembler. *Genome Res.*, **13**, 81–90.

Myers,E.W. *et al.* (2000) A whole-genome assembly of Drosophila. *Science*, **287**, 2196–2204.

Roberts,M. *et al.* (2004) A preprocessor for shotgun assembly of large genomes. *J. Comput. Biol.*, **11**, 734–752.

Sanger,F. *et al.* (1982) Nucleotide sequence of bacteriophage lambda DNA. *J. Mol. Biol.*, **162**, 729–773.