Improving the Draft Assembly of the Horse Genome:

Final Report

Megan Smedinghoff, smeds@umd.edu

Advisor: James A. Yorke, yorke@umd.edu

May 19, 2008

Abstract

In February 2007, Broad Institute of MIT released a draft assembly of the horse genome. To fulfill the requirements of AMSC 663-664, I reassembled the genome using the Celera Assembler and then used my assembly to fix gaps and compressions in the Broad assembly. I was able to increase the N50 contig size of the Broad assembly by 15% and increase the total assembly size by .18%. As a part of this process, I also produced a parallelized version of the University of Maryland Overlapper. Initial tests show that the parallelized overlapper runs 2-4 times faster than the serial version.

Background and Motivation

In February 2007, Broad Institute of MIT announced that it had completed a draft assembly of the horse genome (*equus caballus*). The announcement was the culmination of a $15 million project funded by the National Institute of Human Genome Research and the National Institute of Health (1). The draft genome will allow the equine research community to better understand diseases that affect horses. Additionally, the release of the horse genome has caused some excitement in the human genome research community. There are over 80 known conditions in horses that are analogous to disorders in humans, including arthritis and allergies (2). Many believe that comparative genomics methods will lead to a better understanding of these disorders and therefore better treatments for both animals.

The draft genome produced by Broad Institute was done using the Arachne assembler (3-4). I reassembled the genome using the Celera Assembler (5) and the University of Maryland overlapper (6-7). The Celera Assembler and the Arachne Assembler have slightly different underlying algorithms and therefore produce assemblies that are almost identical but do contain some differences. I also used the University of Maryland overlapper as a preprocessor to Celera. UMD Overlapper has been shown to improve genome quality when used in the Celera pipeline (7). After running UMD overlapper and then Celera, I used the resulting assembly to fix gaps and compressions in the Arachne draft.

In the process of producing a reconciled assembly, I was also able to make improvements to the UMD overlapper. Since the horse genome is large, I was initially forced to split the data into groups and run many of the overlapper commands by hand.

This process was complicated and time-consuming. Fortunately, the computationally intensive part of the overlapper algorithm was highly parallelizable, and I was able to produce a new version that runs well on mammalian sized genomes.

Producing a Celera Assembly of the Horse

Before we can begin to assemble a genome, we must first sequence the organism. Sequencing begins by extracting DNA from the nucleus of a cell (see Figure 1). The DNA is then cut up into smaller pieces. After shearing the DNA, we sort the fragments by size and pick certain sized pieces to make up a "library". A library is a set of DNA strings that have a certain mean length and standard deviation. These strings of DNA are known as "inserts". Once we have a library of inserts, we clone them and put them into a vector. Current sequencing technology does not allow us to read the nucleotides from one side of an insert to the other. We can only usually read 800-1000 nucleotides before the sequence quality becomes unreliable. While we cannot determine the sequence of the entire insert, we can determine the first 800-1000 nucleotides on each end of the insert. These two pieces of DNA are known as "reads". Since we know the length of the insert, we also know the distance between the two reads as well as their orientation relative to each other. The two reads from each insert are known as "mate pairs".

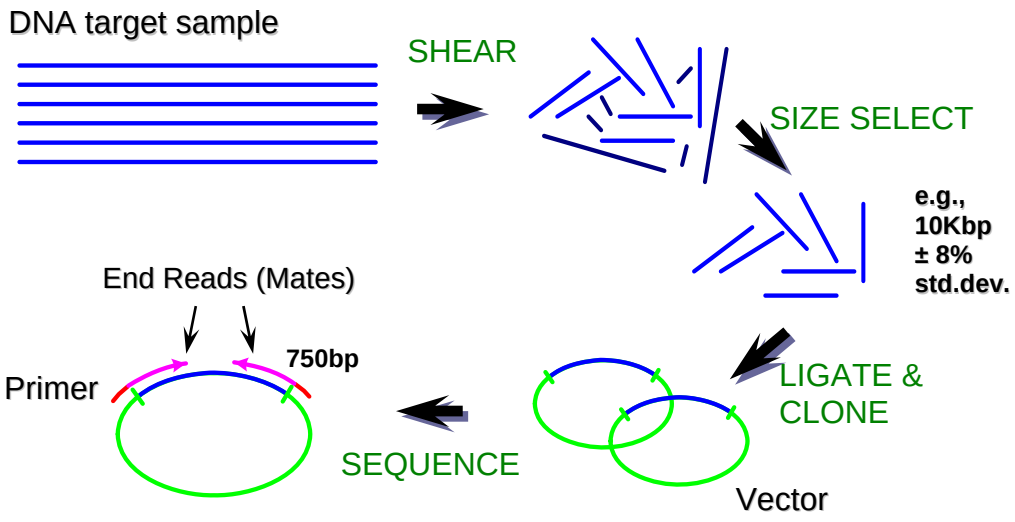

Figure 1: DNA sequencing process (figure courtesy of Art Delcher)

At the end of sequencing we have a set of overlapping reads. The assembler takes these reads, calculates which reads overlap, and then fits them together to produce a genome. The Celera Assembler's algorithm for fitting reads together begins by building "unitigs" (see Figure 2). A unitig is a piece of sequence built from overlapping reads where there is no possibility of an ambiguous overlap. For example, if there is a repeat region in the genome, a read in this region might have two overlaps that are clearly from different sections of the genome. This case is considered an ambiguous overlap and would not be included in a unitig. Once Celera builds all the unitigs, it then uses mate pair information to build "contigs". A contig is a set of unitigs that are placed together. The assembler figures out which unitigs fit together by looking at the overlaps but also by calculating if placing two unitigs together causes paired reads to be the correct distance apart. Finally, once the assembler has built unitigs, it again uses the mate pair information to build scaffolds. Building scaffolds is similar to building contigs, but in this case there is usually little or no overlap between the pieces of sequence. Instead of

seeing that two pieces fit together, the assembler relies on knowing how far apart pairs of reads must be from each other.

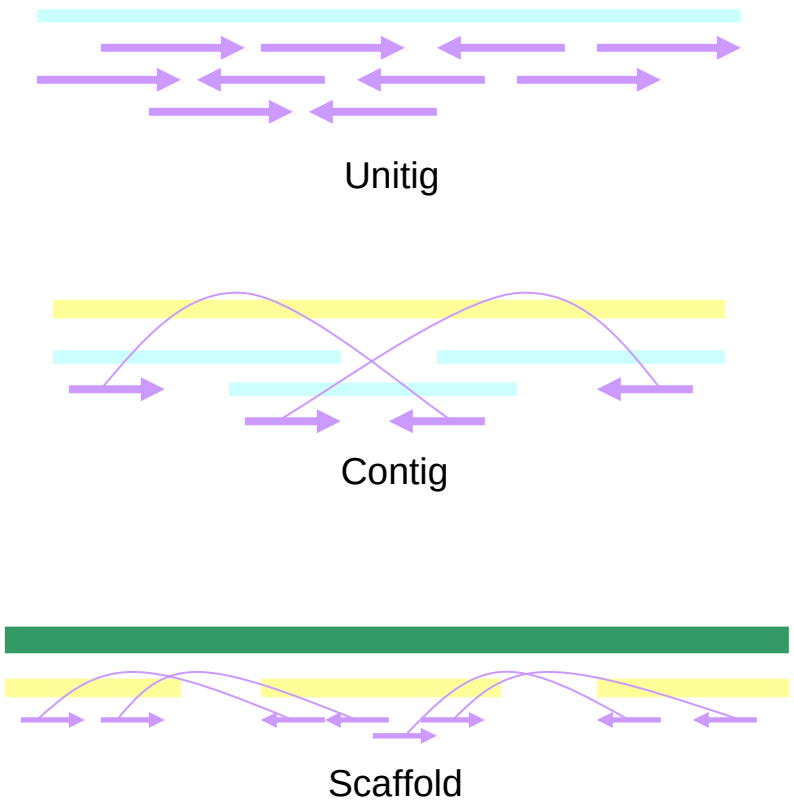

Figure 2: Celera assembler builds successively bigger pieces of sequence known as unitigs, contigs, and scaffolds to create an assembly

To produce an assembly of the horse, I ran the Celera assembler with default parameters. The only modification I made to the Celera pipeline was that I used UMD overlapper to produce the list of read overlaps instead of using Celera's internal routines. It took nine days to run the assembler and produce a draft horse genome (see Table 1).

Assembly	Celera	Arachne
Number of Scaffolds	59,044	9,687
Number of Contigs in Scaffolds	126,810	55,316
Genome Size	2.5 billion bases	2.4 billion bases
N50 Contig Size	77,479 bases	112,381 bases

Table 1: Celera and Arachne assembly statistics

The Celera assembly contained many more scaffolds than the Arachne assembly, but the two genomes were approximately the same size. The Arachne assembly had a larger N50 contig size than the Celera assembly. N50 contig size is the size of a contig such that 50% of all nucleotides are in larger contigs and 50% of all nucleotides are in smaller contigs. N50 contig size is frequently used to measure the quality of an assembly. A larger N50 contig size is usually considered better than a smaller one.

Comparing the Two Assemblies

After producing a Celera assembly of the horse, I started comparing the two assemblies at the sequence level to see how similar they were in terms of scaffold structure. The first step in this process was to run NUCmer, an open source program designed to quickly align large genomes (8-9). Once I had an alignment, I was able to map Celera contigs to Arachne contigs. Not all Celera contigs mapped to Arachne contigs, but I compiled a list of those that did. I then used this list to compare the scaffold structures of the two assemblies. In performing this comparison, I encountered three different types of disagreement between the assemblies: orientation problems, ordering problems, and gap size problems.

Orientation Problems – An orientation problem occurs when contigs from a Celera scaffold map to contigs of an Arachne scaffold but one of the contigs is oriented in the opposite direction from its match (see Figure 3).

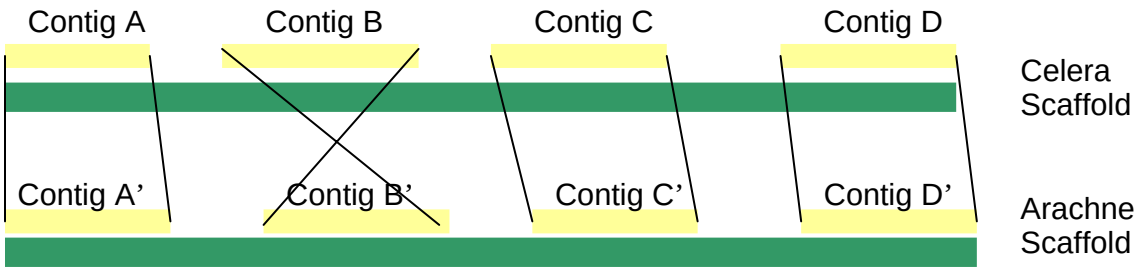

Figure 3: Illustration of an orientation problem

Ordering Problems – An ordering problem occurs when a set of contigs from a Celera scaffold map to a set of contigs from an Arachne scaffold but they map in a different order (see Figure 4).

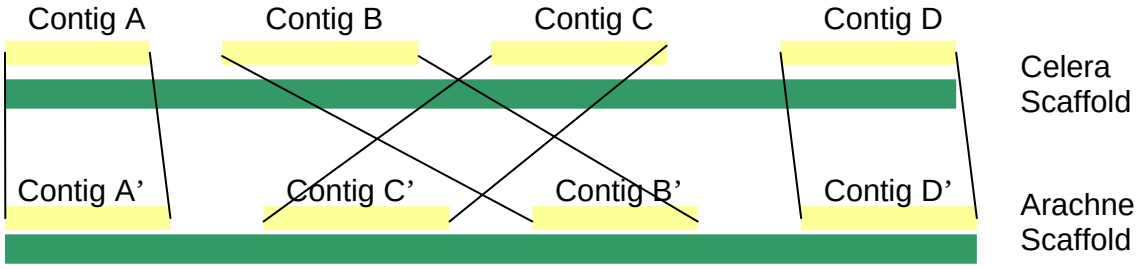

Figure 4: Illustration of an ordering problem

Gap Size Problems – Scaffolds are made up of a set of contigs with gaps between the contigs. We do not know the exact size of each gap, but we do know the mean and standard deviation of the gap from mate pair information. We can use this information to

compare the sizes of the gaps between the Celera contigs and the sizes of the gaps between the Arachne contigs. If the gap between two Celera contigs is considerably bigger or smaller than the gap between two corresponding Arachne contigs then there is a gap size problem. The criteria I used for determining if a gap was too big or too small was that the mean Arachne gap size had to be within four standard deviations of the mean Celera gap size (see Figure 5).

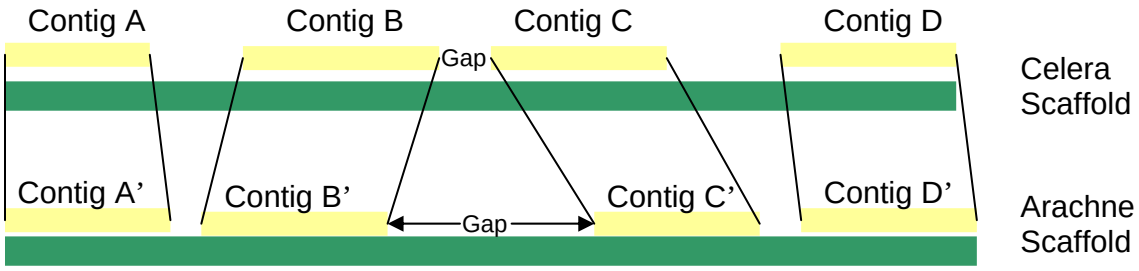

Figure 5: Illustration of a gap size problem

2,252 of the 59,044 Celera scaffolds contained enough matching contigs to compare to Arachne scaffolds. While these scaffolds only represent 4% of all Celera scaffolds, they did contain 96.6% of the nucleotides. I compared each of these scaffolds to Arachne and counted how many times each type of problem occurred (see Table 2).

Total Celera Scaffolds	59,044
Celera Scaffolds Compared to Arachne	2,252 (96.6% of bases)
Celera Scaffolds with Orientation Problems	42
Celera Scaffolds with Ordering Problems	109
Celera Scaffolds with Gap Size Problems	162
Celera Scaffolds with at least one problem	174 (93.1% of bases)

Table 2: Results of the Celera-Arachne comparison

The results show that 92% of the Celera scaffolds that I was able to compare to Arachne had matching scaffold structures. The 174 scaffolds that had some type of problem represented 93.1% of the total genome. In other words, most of the largest scaffolds had some type of problem. This result is to be expected since larger scaffolds have more contigs and therefore more opportunity to have problems.

Producing a Reconciled Assembly

The comparison of the two assemblies provided evidence that the two drafts were similar but contained some minor differences. This observation indicated that we could use the Celera assembly to fix gaps and compressions in the Arachne assembly. A gap is simply two pieces of the official assembly that have a gap between them and match to a portion of the draft assembly that has sequence between those two pieces instead of a gap (see Figure 6). A compression point is a portion of the official assembly that matches a portion of the draft assembly where the draft has additional sequence inserted within the matching region (see Figure 7). Reconciliation software detects gaps and compressions in the official assembly and uses mate placement statistics to decide if the draft assembly is more likely to be correct. If the draft assembly is deemed to be better in a certain region, the change is incorporated into the reconciled assembly.

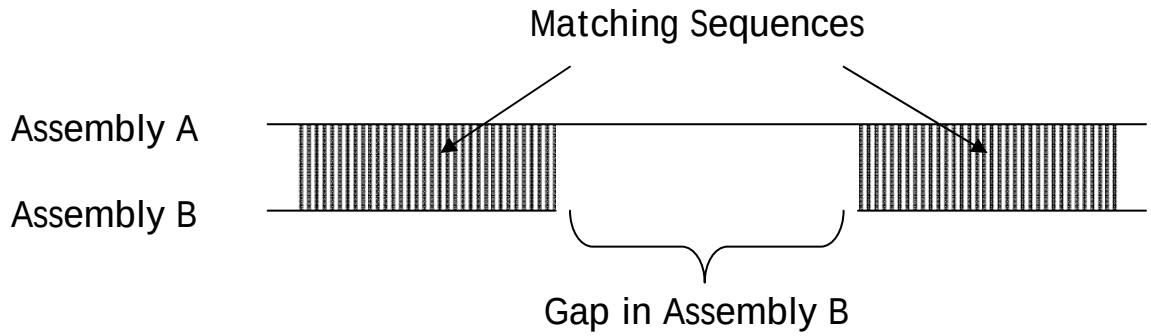

Figure 6: Illustration of a gap

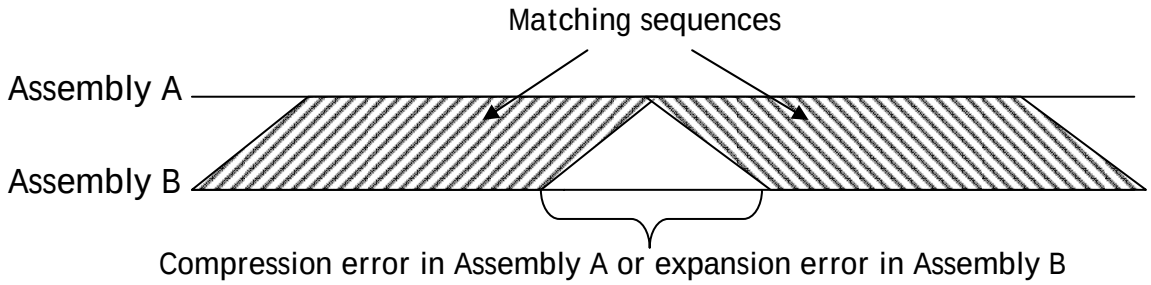

Figure 7: Illustration of a compression

I treated the Arachne assembly as the official assembly and used the Celera assembly to close gaps and fix compressions (see Table 3). The results show that reconciliation was able to fix 80% of the detected compressions. The genome size increased by .18% (4.3 million bases), and the N50 contig size increased by 15%. These increases in size represent increases in quality since the reconciliation software only fixes the official assembly if there is overwhelming evidence that the draft assembly is better.

Assembly	Celera	Arachne	**Reconciled**
Number of Compressions	N/A	687	**136**
Genome Size	2.512 billion bases	2.428 billion bases	**2.433 billion bases**
N50 Contig Size	77,479 bases	112,729 bases	**130,123 bases**

Table 3: Reconciliation statistics

Parallelizing the Overlapper

In addition to producing a reconciled assembly of the horse genome, I also made improvements to the University of Maryland Overlapper. UMD Overlapper was developed by Mike Roberts and relies on the concept of "minimizers" to efficiently detect overlaps (6-7). The algorithm begins by examining each 20-mer in each read and

recording an associated hex value for that 20-mer (see Figure 8). The program then uses a sliding 20 base window and finds the minimum hex value in that window. The 20-mer associated with this value (known as a "minimizer") is then recorded along with the read name and the offset from the end of the read. This list is then sorted and filtered by minimizers that have fewer than 70 matches in the genome (minimizers with more than 70 matches frequently represent repeat regions). This sorted list of minimizers represents sets of candidate pairs for overlap. Reads in these groups are then compared in a process known as "checker", which outputs a list of overlaps. There are some additional post-processing steps including a Poisson test that eliminates overlaps that have a probability of less than 10^{-8} of being true overlaps. There is also an additional step that splits groups of overlaps if there are a certain number of differences within the group. After the post-processing, the final list of overlaps is output.

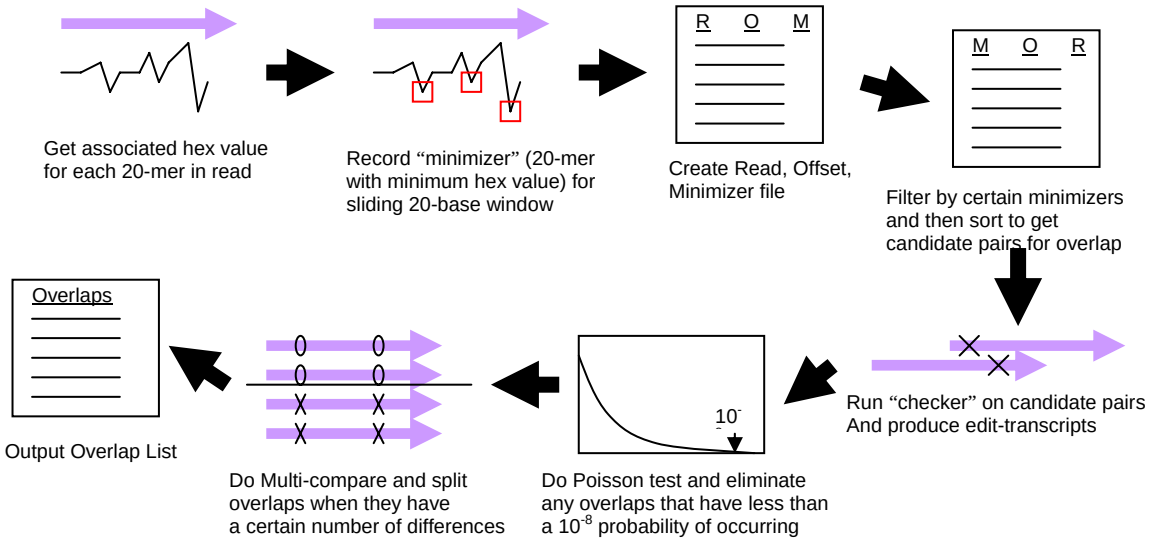

Figure 8: UMD Overlapper algorithm

The most computationally intensive part of the algorithm is the Checker routine that compares the reads. The routine is computationally expensive since every read in a minimizer group needs to be compared to every other read in the group. While this process is time-consuming, it is advantageous in that each minimizer group can be run on a different processor and the routine is therefore easily parallelizable. An additional merit is that there is no overhead for the parallelization since the reads are already in groups. I used Sun Grid Engine functions to submit checker commands to available processors on the cluster. To make sure that my parallelization was working properly, I tested it on a bacterium, a fly, and the horse. In each case I was able to produce correct output and dramatically reduce the running time of algorithm (see Table 4). For a bacteria sized genome, the speedup was 2x. For a fly-sized genome, the speedup was 3.5x. In the case of the horse, I was unable to run the serial version of the overlapper. Estimates indicate that it would have taken about nine days. The corresponding speedup would be 3x.

	Current Overlapper	Parallelized Overlapper	Speedup
Bacteria	30 Minutes	14 Minutes	2.1x
Fly	21.5 Hours	6.25 Hours	3.4x
Horse	9 Days (estimate)	3.07 Days	2.9x

Table 4: Timing results for the parallelization of UMD Overlapper

Conclusion

I produced a 2.5 billion base genome of the horse using the Celera assembler that closely resembled the draft genome produced by Broad Institute. I used the Celera assembly to improve the Arachne assembly and fixed 80% of the 687 detected compressions. I was

also able to increase the Arachne genome size by .18% and increase the N50 contig size by 15%. The reconciliation process inspired a secondary project of improving UMD Overlapper. I parallelized the most computationally intensive routine in the overlapper algorithm and was able to produce a 2-4 times speedup on several different sized genomes.

Acknowledgements

I would like to thank my advisor, Jim Yorke. I would also like to thank the members of the University of Maryland genome group for advice and support during the duration of this project.

References

1. NIH News (February 7, 2007) Horse Genome Assembled.
http://www.nih.gov/news/pr/feb2007/nhgri-07.htm

2. Broad Institute Horse Genome Project Webpage
http://www.broad.mit.edu/mammals/horse/

3. Batzoglou, S., Jaffe, D.B., Stanley, K. et al. (2002), 'ARACHNE: A whole-genome shotgun assembler', Genome Res., Vol. 12(1), pp. 177-189.

4. Jaffe, D. B., Butler, J., Gnerre, S. et al. (2003), 'Whole-genome sequence assembly for mammalian genomes: Arachne 2', Genome Res., Vol. 13(1) pp. 91-96.

5. Myers, E. W., Sutton, G. G., Delcher, A. L. (2000), 'A whole-genome assembly of Drosophila', Science, Vol. 287(5461), pp. 2196-2204.

2478-2483.

6. Roberts, M., Hayes, W., Hunt, B. R., Mount, S. M., and Yorke, J. A. (2004), 'Reducing storage requirements for biological sequence comparison', Bioinformatics, Vol. 20(18), pp. 3363-3369.

7. Roberts, M., Hunt, B. R., Yorke, J. A., Bolanos, R. A., and Delcher, A. L. (2004), 'A preprocessor for shotgun assembly of large genomes', J Comput Biol., Vol. 11(4), pp. 734-752.

8. Delcher, A.L., Phillippy, A., Carlton, J. and Salzberg, S.L. (2002), 'Fast algorithms for large-scale genome alignment and comparison', Nucleic Acids Res., Vol. 30(11) pp.

9. Pop, M., Phillippy, A., Delcher, A. L., and Salzberg, S. L. (2004), 'Comparative genome assembly', Brief Bioinform., Vol. 5(3), pp. 237-248.