Theo Zimmermann

Description of a genome assembler: CABOG

CABOG (Celera Assembler with the Best Overlap Graph) is an assembler built upon the Celera Assembler, which, at first, was designed for Sanger sequencing, but it was revised to handle medium-length sequencing produced with the 454 sequencing machines (Miller et al., 2008). It is scientifically more interesting than its competitor Newbler because it is distributed as a free software and more information is available on its algorithm.

Sanger chemistry produces long reads, with high accuracy. Next-generation sequencing produces short reads, with higher error rate but high coverage. New software have been devised for this type of sequencing. The software CABOG uses a hybrid approach and is more robust to homopolymer (single-letter) read length uncertainty, varying read length and higher error rate. It has been designed in particular to take advantage of paired-end mate information. It accepts "pyrodata" alone or in a combination with Sanger data.

CABOG belongs to the class of Overlap/Layout/Consensus assemblers, which are basically looking for Hamiltonian paths. It extends these three steps by computing first unitigs from pairwise overlaps, then contigs, then scaffolds and by combining them with multiple sequences alignment techniques, during the consensus phase (Miller, Koren, & Sutton, 2010). Unitigs are uncontested contigs (largest brick the assembler constructs with certainty). Contigs are MSA without gaps and scaffolds contain gaps. They are combined using, in particular, the paired-end information.

Compared to the initial technique for Sanger data, some modules of the Celera Assembler have been reused, or improved by considerations orthogonal to the problem of next-generation sequencing. It is the case of modules to handle scaffolds because they are large enough to be no different with next-generation sequencing from what they used to be with Sanger data. The orthogonal improvements will not be described here.

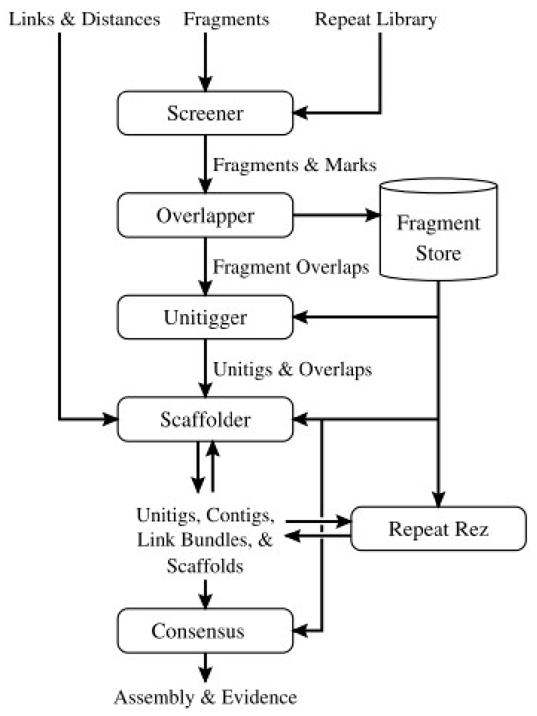

Fig. 1. Assembly pipeline. From an engineering perspective, sequences of messages flow from one stage to the next. Each stage performs work on its input stream, producing a stream of output messages reflecting its transformational function. The text gives the function of each stage.

The Celera assembler

Let's describe more precisely the general algorithm behind the Celera Assembler, at the basis of CABOG.

According to (Miller et al., 2010), all OLC assemblers are built on the following schema:

- The first phase is the discovery of overlaps. It relies upon pairwise comparison of reads and to speed the process, the common heuristic is to use seeds which are k-mers. Details and specificities of CABOG in the domain are explained later.
- The second phase is to build an approximate layout of the whole genome by using the resulting graph from the first phase.
- The third and last phase uses multiple sequence alignment to place all reads on the approximate layout. During this phase, a consensus sequence for the whole genome is determined.

The Celera assembler reproduce these steps several time during the process (Miller et al., 2008):

- Overlap detection;
- Combination of reads into unitigs;
- Consensus sequence for each unitig;
- Combination of unitigs into contigs and scaffolds;
- Consensus sequence for the scaffolds (i.e., the whole genome if the number of scaffolds corresponds to the number of chromosomes).

Definitions: (Myers, 2000) gives precise definitions of the different terms:

- A unitig is "a maximal interval subgraph of the graph of all fragment overlaps for which there is no conflicting overlaps to an interior vertex". Almost every unitig is correct. The only exception is when very similar repeats are collapsed into a single unitig but they came from different positions.
- A contig is "a series of overlapping unitigs".
- A scaffold is a set of contigs which are "ordered, oriented, positioned with respect to each other by mate pairs whose reads are in adjacent contigs" so that the distance between two successive contigs is approximately known.

Only the first two steps have been modified in CABOG compared to what was done in the Celera assembler. Indeed, the authors of the assembler determined that these phases were at the origin of most of the problems induced by hybrid data. Most of the changes will be done to make the two phases less specific to Sanger data, by replacing the costly process they used. Also some correction heuristics were too often triggered when using short and repetitive reads so they were removed.

Overlap detection

Because it has been observed that errors are more frequent toward the end of a read and it may prevent true overlaps from being detected, a trimming step removes the end of the longest reads if an overlap has been detected that does not span the end.

A question to discuss: would this step be useless in a method using only k-mers because k-mers would already induce trimmed reads? Also, we can remark that this trimming phase creates a bias toward finding more overlaps, because this is the only case when it applies, whereas k-mer splitting is unbiased because applied uniformally. Nevertheless, some Eulerian methods also trim the reads.

The algorithm for overlap detection first looks for seeds for overlap in terms of short exact matches on compressed sequences. Seeds are k-mers of a predefined length (k = 22 by default). k-mers which are too frequent are excluded. The homopolymers are reduced to only one instance of the letter in the compressed format because it is one of the most common error of sequencers to miscalculate the length of a homopolymer. All reads that share sufficiently many k-mers are considered. Then, the uncompressed version is considered to determine if there actually is overlap but this time, it does not need to match exactly (to be robust to substitution errors or other small read errors). The overlaps require to be 94% accurate on a length of 40bp at least. Note how these defaults values are suited for 454 data but might be too big for Illumina data.

Unitigs

The assembler is said to use "best overlaps" because it builds unitigs in a greedy fashion (choosing at each time, the next best overlapping read). This method may create errors which are hopefully corrected later but it has not the problems of the former heuristic s the Celera assembler used which were very sensitive to repeats of length longer than a read. Moreover, it is a linear-time method. In more details, a multigraph is constructed where nodes represent each end of a read, undirected edges link two nodes and represent reads and directed edges represent best overlaps ("best is defined as aligning the most bases" (Miller et al., 2010)). If there is no error, best overlaps must be "mutual" it means that if the end of a read is the best overlap for one end of another read then the relation is reciprocal (the directed edges exist in both ways).

After breaking arbitrarily potential cycles, for each path in the graph, a unitig is built by selecting one (best-scoring) read and following its two ends and the best overlaps until it reaches the path end or a read which is already in another unitig. Unitigs are then broken again at most intersections where there would have been several possibilities for constructing the unitig. These intersections correspond to repeats or noise (for instance, spurs which are error-prone ends of reads) that make their analysis more complicated. CABOG tries to break unitigs only at intersections which correspond to noise. Unitigs are also broken when a mate-pair constraint is violated.

The rest of the assembler is similar to what the Celera assembler did for Sanger data. In particular, scaffolding use again mate-pair constraints.

Scaffolds

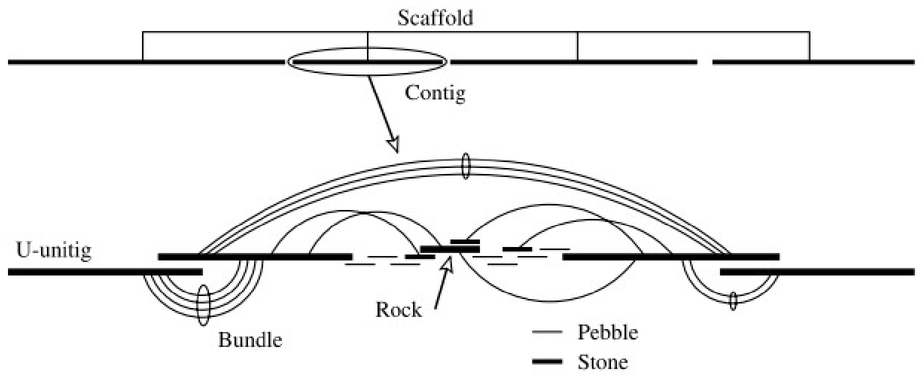

Fig. 4. Anatomy of a scaffold. A scaffold is a collection of ordered contigs with approximately known distances between them. Our contigs are built from U-unitigs that form a scaffold via bundles and then have a series of rocks, stones, and pebbles filled into the gaps between them (where possible).

Mate-pairs are very important for the Celera assembler. They were already used to split suspicious unitigs in the previous phase, but they really become necessary for the contigs construction and scaffolding phases. A mate-pair which has one end in one unitig and one end in another unitig is useful to orient and position the two. To reach a level of near certainty, at least two such mate-pairs are necessary. Consequently, the Celera assembler will first use bundles of mate-pairs to combine unitigs into contigs. At this point, contigs contain a very low number of errors. Then, to finish the construction of contigs, more aggressive and less reliable steps are followed: unitigs (rocks) are positioned if there are mate-pairs which consistently place them with respect to other unitigs; a unitig (stone) is positioned if there is one mate-pair and a consistent tiling of overlapping unitigs which place it with respect to the same given unitig; the last step attempts to fill the gaps with the best possible tiling of overlapping unitigs (pebbles). (Myers, 2000) provides the level of certainty of each of the steps.

The ordering and positioning of contigs to form scaffolds is not a separate step from the contigs construction. In fact, when unitigs are linked by a bundle of mate-pairs but there remains a gap at the end of the scaffolding process, then we have got a scaffold and not a simple contig. The gaps will only be filled during the consensus phase, when reads are aligned to the layout.

Consensus

During the last step, every read is aligned with respect to the approximate layout and some pebbles can still be moved during this stage. A consensus genome is computed.

Results

According to the GAGE study which was conducted on Illumina data (Salzberg et al., 2012), CABOG is among the best performing assemblers. It gives reasonable-sized contigs and scaffolds with a limited amount of errors. This is surprisingly enough because CABOG was not designed for Illumina data. But changing the parameters might be enough to tune it for a different kind of data. And I am not aware of the possible improvements the authors of CABOG have done between 2008 and 2012.

The evaluation on a human chromosome which was conducted in the study is of particular interest. CABOG and AllPaths-LG were the two best performing assemblers. But the study conclude that all assemblers are still very rudimentary with human data. So Hamiltonian and Eulerian assemblers perform equally well. But next-generation data is not good enough yet for human-scale genomic assembly. One idea to improve it would be to provide more mate-pairs.

Conclusion

This assembler is interesting because it can target hybrid data, produced with Sanger sequencers and next-generation sequencers. It was designed by modifying the parts of the algorithm that were not working well with pyrodata. Essentially, it seems that the algorithm has been simplified, heuristics have been removed... The greedy construction of the overlap graph makes it computationally tractable, but at the cost of more errors. Also, even if it is not clear what exactly makes the assembler better for handling hybrid data, my guess is that it just comes from the fact CABOG was designed in a non-specific way to one particular kind of data.

Credits

All the pictures have been reproduced from (Myers, 2000).

Bibliography

Miller, J. R., Delcher, A. L., Koren, S., Venter, E., Walenz, B. P., Brownley, A., … Sutton, G. (2008). Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics (Oxford, England)*, *24*(24), 2818–24. doi:10.1093/bioinformatics/btn548

Miller, J. R., Koren, S., & Sutton, G. (2010). Assembly algorithms for next-generation sequencing data. *Genomics*. Retrieved from http://www.sciencedirect.com/science/article/pii/S0888754310000492

Myers, E. W. (2000). A Whole-Genome Assembly of Drosophila. *Science*, *287*(5461), 2196–2204. doi:10.1126/science.287.5461.2196

Salzberg, S. L., Phillippy, A. M., Zimin, A., Puiu, D., Magoc, T., Koren, S., … Yorke, J. a. (2012). GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome research*, *22*(3), 557–67. doi:10.1101/gr.131383.111