Readjoiner: a fast and memory efficient string graph-based sequence assembler

Giorgio Gonnella and Stefan Kurtz
Center for Bioinformatics (ZBH), University of Hamburg

gonnella@zbh.uni-hamburg.de

In a recently published paper [GK12] we describe a new fast and memory efficient string graph-based sequence assembler: *Readjoiner*. In this extended abstract, we summarize the background, methods and results.

Background

The amount of data delivered by next-generation DNA sequencing technologies challenges the current generation of *de novo* sequence assemblers based on De Bruijn graphs.

An alternative framework of growing interest is the assembly string graph [Mye05]. As the classical overlap graph, the string graph represents sequencing reads by vertices and overlaps between reads by edges: however, in the string graph only *irreducible* suffix-prefix matches are considered.

The string graph combines the strengths of the classical overlap-layout-consensus paradigma with a compact representation suitable for the next-generation sequencing datasets. The main advantage over the De Bruijn graph is that it does not require to artificially split the reads into k-mers, thus improving the assembly of sequences containing short repeats. Furthermore, the string graph is more compact than the De Bruijn graph, thus allowing to efficiently handle larger datasets.

To construct the string graph, fast and space efficient algorithms for the computation of all suffix-prefix matches are required. Previous approaches use a suffix array (Edena, [HFF$^+$08]) or an FM-index (SGA, [SD12]) or a compact representation of the overlap graph (Leap, [DR11]).

Methods

We developed efficient methods for the construction of a string graph from a set of sequencing reads. We use suffix sorting and scanning methods to compute suffix-prefix matches; furthermore, transitive edges are recognized early in the process and excluded from the graph.

The first step of our assembly approach is to eliminate reads that are prefixes or suffixes of other reads: these are recognized by lexicographically sorting all reads and their reverse complements, using a modified radixsort for strings [KR09].

In the following step, suffix-prefix matches longer than ℓ_{min} are computed, where ℓ_{min} is an user-defined parameter. The method consists of two main algorithms. The first algorithm identifies and lexicographically sorts all *SPM-relevant suffixes*: these are suffixes of reads, sharing a prefix of length $k \leq \ell_{min}$ with some read in the readset. Here k is a parameter allowing for time / space tradeoffs in the computation. The second algorithm enumerates the suffix-prefix matches given a sorted list of SPM-relevant suffixes.

SPM-relevant suffixes are sorted using a strategy borrowed from the counting sort algorithm [CLR90]. An efficient solution is achieved by combining the use of sorted buffers for the elements to be counted/inserted, a filter based on substrings of the initial k-mers of the reads and a partitioning strategy considerably reducing the space peak of the implementation.

The suffix-prefix matches are computed using an algorithm based on a bottom-up traversal of the lcp-inverval tree. This is obtained by processing the buckets of SPM-relevant suffixes with a variant of the algorithm presented in [AKO04], additionally delivering the leaf edges of the virtual lcp-interval tree.

In order to only output irreducible suffix-prefix matches, we maintain an additional trie-data structure and exploit a novel characterization of transitive suffix-prefix matches.

The assembly string graph is constructed from the list of all irreducible suffix-prefix matches, as described in [Mye05]. Heuristically, bubbles and short dead-end paths likely arising from sequencing errors, are optionally removed from the graph. Finally, the sequence corresponding to all unbranched paths in the graph is output as a collection of contigs.

Results and Conclusion

We implemented our methods in a new open source sequence assembler, called *Readjoiner*, as part of the *GenomeTools* [GEN] genome analysis suite. Readjoiner is freely available at `http://www.zbh.uni-hamburg.de/readjoiner`.

We extensively evaluated our assembler on simulated error-free sequencing read sets based on human genomic sequences. We compared the performance of *Readjoiner* with that of the previous string graph-based tools: Edena [HFF+08], SGA [SD12] and Leap [DR11]. The results were evaluated using metrics developed by the Assemblathon project [EBSJ+11] and using the Plantagora assessment tool [BMRY11].

Our tests show that *Readjoiner* is faster and more space efficient than previous string graph-based tools. *Readjoiner* was $13-14\times$ faster than Edena, $19-20\times$ faster than SGA and $1.6-1.8\times$ faster than LEAP. Furthermore it uses about $9.1-9.3\times$ less memory than Edena, $1.1-1.2\times$ less memory than SGA and $1.6-3.0\times$ less memory than LEAP. Furthermore, it scales well for large datasets. For example, a 40 $\times$ coverage human genome dataset (100 nt reads for a total of 115 Gb) can be assembled on a single core in 51 hours using 52 Gb RAM.

Readjoiner is actively developed and improvements over the version described in [GK12] have been achieved. For example, suffix-prefix matches derived from independent parts of our data structures are computed in threads. We plan to integrate an error correction algorithm and incorporate mate pairs information during the assembly phase.

We would like to remark that our paper, published less than a month ago, attracts considerable interest: besides acquiring the "Highly accessed" designation by the publisher (BioMed Central), the paper was, as of May 31th, the most viewed paper for BMC Bioinformatics during May 2012 [BMC12].

References

[AKO04] M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch. Replacing Suffix Trees with Enhanced Suffix Arrays. *Journal of Discrete Algorithms*, 2:53–86, 2004.

[BMC12] BMC Bioinformatics, Most Viewed, Last 30 days. http//www.biomedcentral.com/bmcbioinformatics/mostviewed, accessed on May 31st, 2012.

[BMRY11] Roger Barthelson, Adam J. McFarlin, Steven D. Rounsley, and Sarah Young. Plantagora: Modeling Whole Genome Sequencing and Assembly of Plant Genomes. *PLoS ONE*, 6(12):e28436, 12 2011.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[DR11] Hieu Dinh and Sanguthevar Rajasekaran. A memory-efficient data structure representing exact-match overlap graphs with application for next-generation DNA assembly. *Bioinformatics*, 27(14):1901–1907, Jul 2011.

[EBSJ+11] Dent A. Earl, Keith Bradnam, John St. John, Aaron Darling, Dawei Lin, Joseph Faas, Hung On Ken Yu, Buffalo Vince, et al. Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Research*, 2011.

[GEN] GenomeTools - The versatile open source genome analysis software. `http://genometools.org`.

[GK12] Giorgio Gonnella and Stefan Kurtz. Readjoiner: a fast and memory efficient string graph-based sequence assembler. *BMC Bioinformatics*, 13(1):82, 2012.

[HFF+08] David Hernandez, Patrice Franois, Laurent Farinelli, Magne Osters, and Jacques Schrenzel. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res*, 18(5):802–809, May 2008.

[KR09] Juha Kärkkäinen and Tommi Rantala. Engineering Radix Sort for Strings. In Amihood Amir, Andrew Turpin, and Alistair Moffat, editors, *String Processing and Information Retrieval*, volume 5280 of *Lecture Notes in Computer Science*, pages 3–14. Springer Berlin / Heidelberg, 2009.

[Mye05] E. W. Myers. The fragment assembly string graph. *Bioinformatics*, 21 Suppl 2:79–85, Sep 2005.

[SD12] Jared T Simpson and Richard Durbin. Efficient de novo assembly of large genomes using compressed data structures. *Genome Research*, 22(3):549–556, 2012.