SAFARI

Apollo: A Sequencing-Technology-Independent, Scalable, and Accurate Assembly Polishing Algorithm

arXiv.org

Scan me

GitHub

Scan me

Can Firtina[1], Jeremie S. Kim[1,2], Mohammed Alser[1], Damla Senol Cali[2], A. Ercument Cicek[3], Can Alkan[3], and Onur Mutlu[1,2,3]

[1] ETH zürich [2] Carnegie Mellon [3] Bilkent University

1: High Throughout Sequencing (HTS)

HTS: Produces large amount of sequencing data at relatively low cost compared to first-generation sequencing methods.

Two types of HTS technologies:

1. Second-generation sequencing technologies (e.g., Illumina) generate the most accurate reads (e.g., 99.9% accuracy), but the length of these reads are short (e.g., 100-300 basepairs).
2. Third-generation sequencing technologies (e.g., PacBio's SMRT) produce long reads (e.g., up to 2M basepairs) at the cost of high error rate (e.g., an error rate of 10%).

Motivation: Long reads make it more likely to generate chromosome-size contigs but also more challenging as the error-prone reads often result in an **erroneous assembly**.

2: Error Correction

Error-prone assemblies can be corrected in two ways:

1. Correcting the errors of long reads before generating the assembly (i.e., **error correction**), which requires either:
   - ✖ Reads from multiple sequencing technologies (costly) or
   - ✖ High coverage long reads (costly)
2. Correcting the errors of the assembly using long or short reads (i.e., **assembly polishing**) that
   - ✖ Mostly works with only reads from a limited set of sequencing technologies
   - ✖ Cannot use multiple read sets within a single run
   - ✖ Cannot scale well to polish large genome

✓ Both approaches can improve accuracy of an assembly

3: Problem

The technology and genome-size dependency prevents state-of-the-art assembly polishing algorithms from either

1. Using all available read sets from multiple HTS technologies
2. Polishing large genomes (e.g., a human genome)

4: Our Goal

Provide a universal algorithm to improve accuracy of genome assembly that

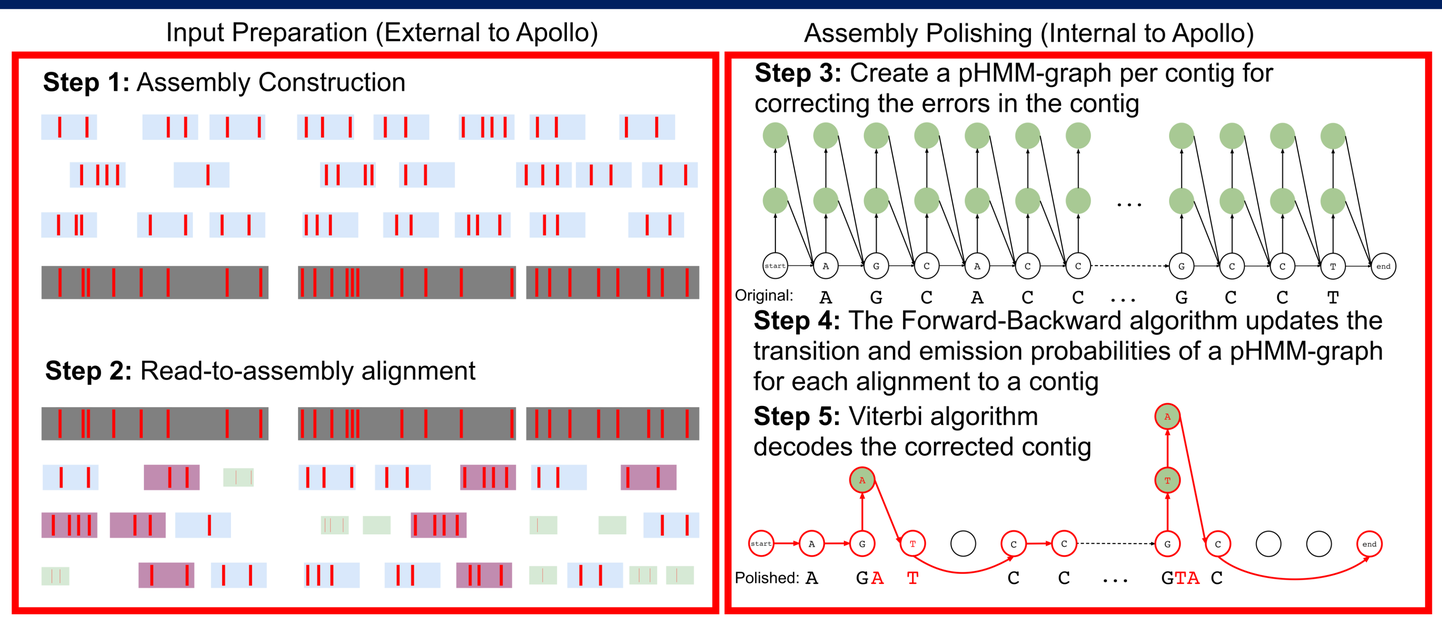
1. Uses read sets from **all available HTS technologies within a single run**
2. Scales well to **polish large genomes**

5: Key Observations

1. Sequencing errors are ***not* entirely random**
2. **A profile hidden Markov model** (pHMM) graph is a good fit to represent a sequence and its error profile
3. **Read-to-assembly alignment:** Aligning reads to a contig provides a clue about the differences between a contig and an aligned read
4. Read-to-assembly alignment can be used to train a pHMM-graph to correct the errors in the assembly

Based on these observations, we propose a machine learning-based universal technology-independent assembly polishing algorithm, called **Apollo**

6: Apollo Walkthrough



7: Experimental Setup and Data Sets

- We evaluate the polished assemblies based on:
  1. ***Aligned Bases:*** The percentage of bases of an assembly that align to its reference
  2. ***Accuracy:*** The fraction of identical portions between the aligned bases of an assembly and its reference
  3. ***Polishing Score:*** *Accuracy* x *Aligned Bases*
  4. **Runtime** and the peak **memory** usage
- We ran all the tools on a server with 192 GB of memory by assigning 45 threads for each run
- *Apollo* is compared with *Nanopolish*, *Racon*, *Quiver*, and *Pilon*
- We used E.coli K-12, E.coli O157, E.coli O157:H7, Yeast S288C, Human CHM1, and Human HG002 **data sets** in our experiments
- **Ground truth:** Highly accurate assemblies either from the same sample or a well-known reference of the species

8: Applicability of the Polishing Algorithms to Large Genomes

✖ **Racon, Pilon, and Quiver cannot polish the large genome assembly** using high coverage read sets due to **high** computational resources they require

✖ **Racon is only able to polish a large genome when using low coverage read sets**

✓ **Apollo is the *only* assembly polishing algorithm that can scale well to polish large genome assemblies**

Aligner	Sequencing Tech. of the Reads	Polishing Algorithm	Runtime	Memory (GB)
Minimap2	PacBio (35X)	Apollo	227h 12m 15s	62.91
BWA-MEM	PacBio (35X)	Apollo	198h 41m 15s	58.60
Minimap2	PacBio (35X)	Racon	N/A	N/A
BWA-MEM	PacBio (35X)	Racon	N/A	N/A
pbalign	PacBio (35X)	Quiver	N/A	N/A
Minimap2	PacBio (8.9X)	Apollo	55h 38m 44s	44.99
BWA-MEM	PacBio (8.9X)	Apollo	41h 38m 27s	45.00
Minimap2	PacBio (8.9X)	Racon	2h 48m 25s	54.13
BWA-MEM	PacBio (8.9X)	Racon	1h 36m 39s	51.55
pbalign	PacBio (8.9X)	Quiver	N/A	N/A
Minimap2	Illumina (22X)	Apollo	96h 22m 16s	101.12
BWA-MEM	Illumina (22X)	Apollo	102h 01m 57s	107.06
Minimap2	Illumina (22X)	Racon	N/A	N/A
BWA-MEM	Illumina (22X)	Racon	N/A	N/A
Minimap2	Illumina (22X)	Pilon	N/A	N/A
BWA-MEM	Illumina (22X)	Pilon	N/A	N/A

9: Using Read Sets from Multiple Sequencing Technologies

✓ **Apollo generates the most accurate Canu assemblies for a species** than running other polishing tools multiple times

✓ **Apollo never generates an assembly with a polishing score lower than the original assembly** whereas other polishing tools **may** produce such assemblies

✖ **Running Apollo once is significantly slower** than running other polishing tools *multiple* times

Data Set	First Run	Second Run	Aligned Bases (%)	Accuracy	Polishing Score	Runtime	Memory (GB)
E.Coli O157			99.94	0.9998	0.9992	43m 53s	3.79
E.Coli O157	Apollo (Hybrid)		99.94	0.9999	**0.9993**	8h 16m 08s	13.85
E.Coli O157	Racon (PacBio)	Racon (Illumina)	99.94	0.9994	0.9988	21m 44s	22.65
E.Coli O157	Racon (PacBio)	Racon (PacBio)	99.94	0.9984	0.9978	**4m 58s**	**2.43**
E.Coli O157	Racon (PacBio)	Pilon (Illumina)	99.40	0.9989	0.9829	12m 14s	8.51
E.Coli O157	Pilon (Illumina)	Pilon (Illumina)	99.94	0.9999	**0.9993**	4m 10s	11.40
E.Coli O157	Pilon (Illumina)	Racon (PacBio)	99.94	0.9986	0.9980	4m 58s	11.40
E.Coli O157	Quiver (PacBio)	Pilon (Illumina)	99.94	0.9998	0.9992	5m 01s	7.50
E.Coli O157	Quiver (PacBio)	Racon (PacBio)	99.94	0.9986	0.9980	5m 13s	2.48
E.Coli O157:H7			100.00	0.9998	0.9998	43m 19s	3.39
E.Coli O157:H7	Apollo (Hybrid)		100.00	0.9999	**0.9999**	5h 58m 05s	8.86
E.Coli O157:H7	Racon (PacBio)	Racon (Illumina)	100.00	0.9995	0.9995	9m 43s	6.56
E.Coli O157:H7	Racon (PacBio)	Racon (PacBio)	100.00	0.9970	0.9970	**5m 36s**	**2.24**
E.Coli O157:H7	Racon (PacBio)	Pilon (Illumina)	100.00	0.9996	0.9996	10m 23s	6.41
E.Coli O157:H7	Pilon (Illumina)	Pilon (Illumina)	100.00	0.9998	0.9998	35m 12s	10.79
E.Coli O157:H7	Pilon (Illumina)	Racon (PacBio)	100.00	0.9996	0.9996	6m 04s	10.75
Yeast S288C			99.89	0.9998	**0.9987**	1h 20m 39s	6.24
Yeast S288C	Apollo (Hybrid)		99.89	0.9998	**0.9987**	11h 08m 41s	**6.38**
Yeast S288C	Racon (PacBio)	Racon (Illumina)	99.89	0.9994	0.9983	38m 21s	6.93
Yeast S288C	Racon (PacBio)	Racon (PacBio)	99.89	0.9949	0.9938	49m 52s	6.93
Yeast S288C	Racon (PacBio)	Pilon (Illumina)	99.89	0.9992	0.9981	26m 25s	14.25
Yeast S288C	Pilon (Illumina)	Pilon (Illumina)	99.89	0.9998	**0.9987**	**1m 10s**	11.85
Yeast S288C	Pilon (Illumina)	Racon (PacBio)	99.89	0.9960	0.9949	21m 42s	11.85

10: Conclusion

- **Two major functionalities that are *not* possible with prior tools:**
  1. Apollo scales well with **polishing large genome assemblies**
  2. Apollo is the best tool that can **consistently construct the most reliable Canu-generated assemblies** when reads from multiple sequencing technologies are used
- We show there is a dramatic difference between non-machine learning based algorithms and the machine learning based ones in terms of **runtime**
- As **future work**, it is possible to accelerate the calculation of the Forward-Backward algorithm and the Viterbi algorithm using Tensor cores, SIMD, and GPUs.