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

Can Firtina1, Jeremie S. Kim1,2, Mohammed Alser1, Damla Senol Cali3, A. Erçument Cicek3, Can Alkan1, and Onur Mutlu1,2,3

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 2 M 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 erroneous reads often result in an erroneous 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 assemblies 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 (hPMM) 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

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

- We use the following tools for error correction:
  - Racon
  - Pilon
  - Quiver

- We use 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**

- Apollo 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

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

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.