CSE 566 Spring 2023

Genome Assembly

Instructor: Mingfu Shao

Error Correction

- An important task with lots of existing tools: Lighter, LoRDEC, Nanocorr, proovread, etc.
- Error correction using frequency of kmers
- Observation: each error creates k new kmers; they are often less frequent due to the low frequency of errors.
- Step 1: count the frequency of kmers in the given reads (tools: Jellyfish, KMC, Squeakr, etc)
- Step 2: identify “k-window of low frequency” in reads

k=6

Error Correction using kmers

```
GTATTACTCGTCTGG     (2)
 TATTACTCGTCTGGC    (3)
GTATTACACGTCTGGC    (1)
 TATTACTCGTCTGG     (1)

GTATTA              (3)
 TATTAC             (7)
  ATTACT            (6)
ATTACA              (1)
 TTACTC             (6)
 TTACAC             (1)
  TACTCG            (6)
  TACACG            (1)
   ACTCGT           (6)
   ACACGT           (1)
    CTCGTC          (6)
    CACGTC          (1)
     TCGTCT         (6)
     ACGTCT         (1)
      CGTCTG        (7)
       GTCTGG       (7)
        TCTGGC      (4)
```

```
GTATTACACGTCTGGC
3711111117 74
```

```
GTATTACTCGTCTGG
3766666677
```

Formulating Genome Assembly

- All models/formulations are wrong. No formulation can fully model all constraints, complication of the genome, and the noisy, imperfect data.
- Balance the complexity of the formulation/model, and the hardness of the resulting formulation.
- All 3 formulations (and more) were deeply studied, but none of them was actually widely used in practice. Why? We will focus on discussing their advantages of disadvantages, rather than algorithms.

Formulation 1

- Intuition: to cover all reads; principle of parsimony
- Input: all reads
- Output: the shortest string that include each read as a substring (shortest common superstring; SCS)
- Example:
  - G* = AATTCCAGCTGATTCCAGT (length = 19)
  - Reads (every 3 mers): AAT, ATT, TTC, TCC, CCA, CAG, AGC, GCT, CTG, TGA, GAT, AGT
  - SCS solution: AATTCCAGCTGATAGT (length = 16)

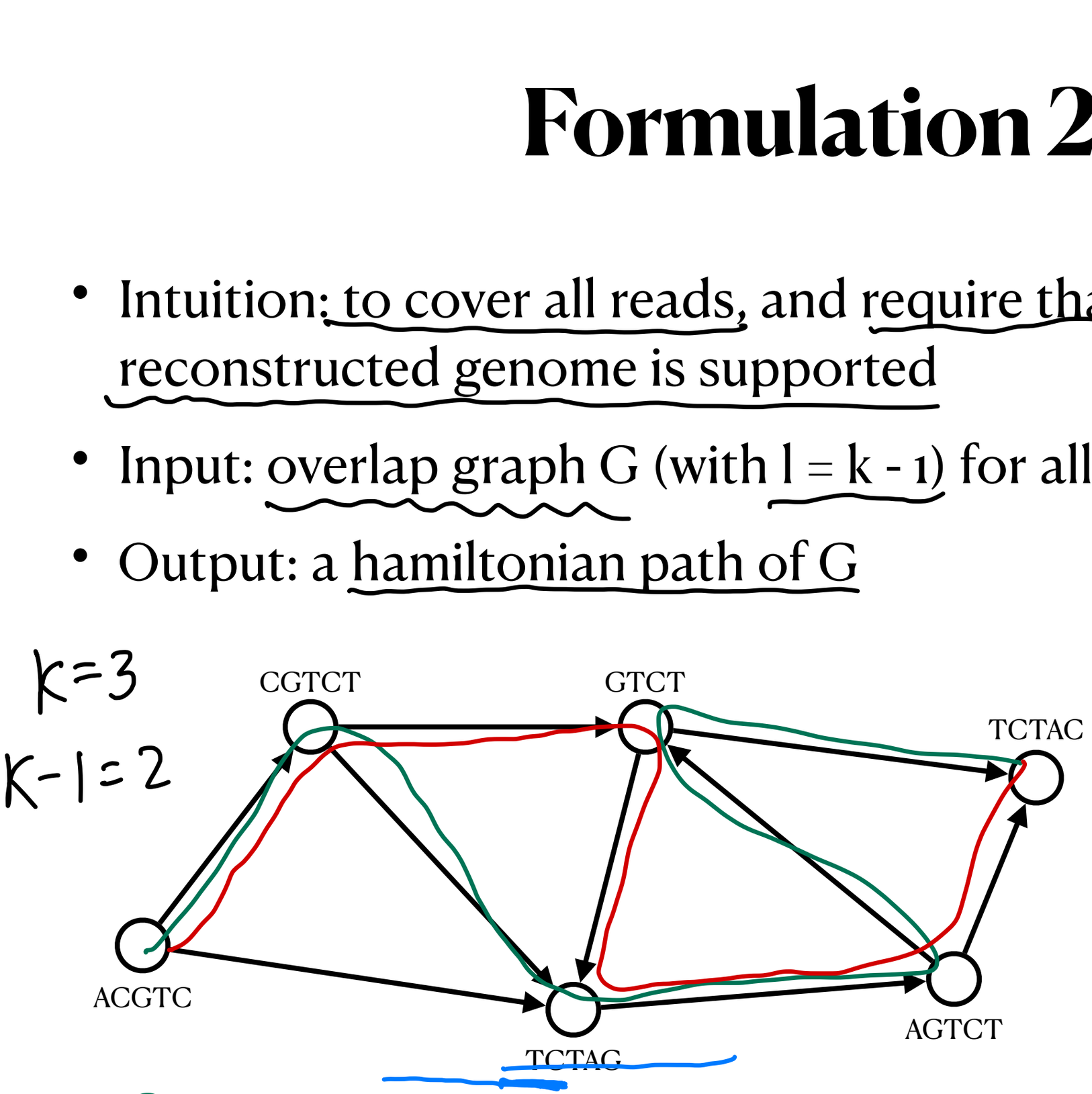
L = 3

Properties of Formulation 1

- This formulation is NP-hard.
- Over-collapsing on repeats: if every read is of length <= L, then the SCS does not contain a repeat of length >= 2L - 1.
- Can create parts not supported by reads. If we assume “full” coverage, then this should not be allowed.

Formulation 2

- Intuition: to cover all reads, and require that every kmer of the reconstructed genome is supported
- Input: overlap graph G (with l = k - 1) for all reads
- Output: a hamiltonian path of G



Properties of Formulation 2

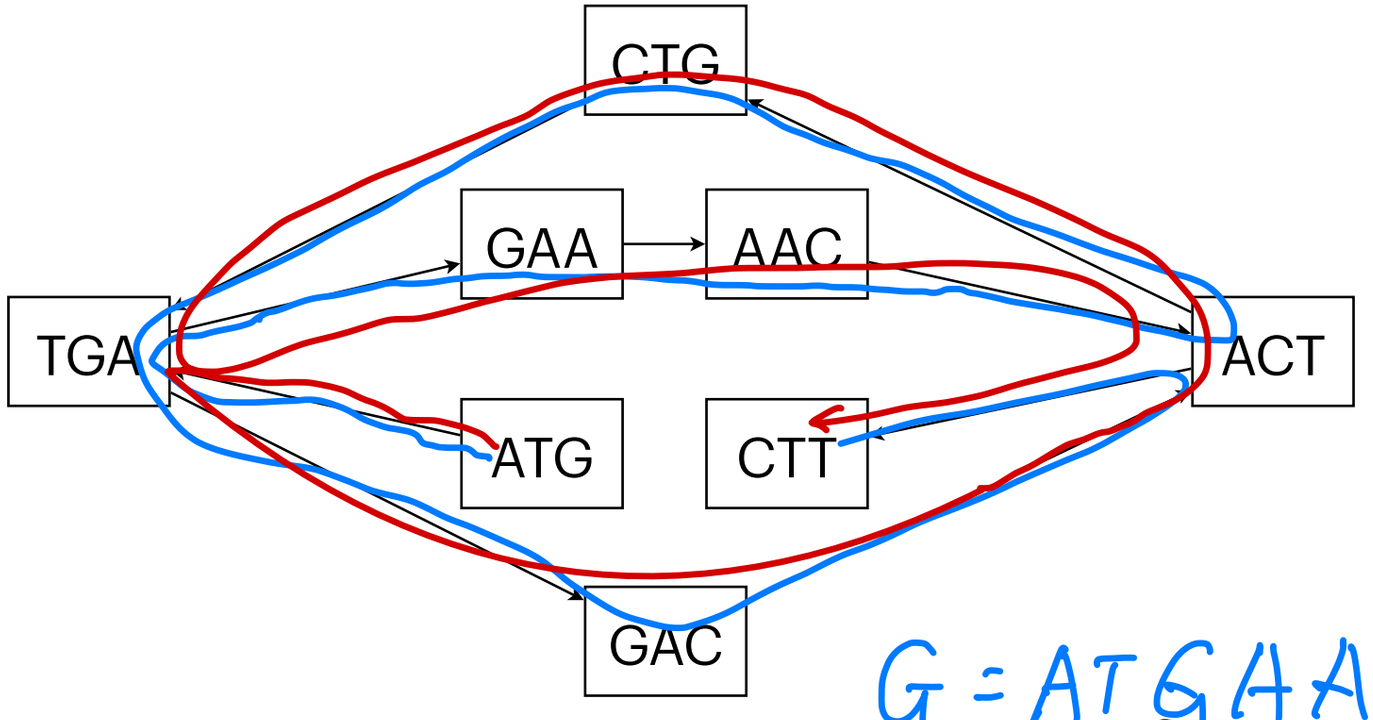
- This formulation is also NP-hard.
- There may be multiple Hamiltonian paths.
- Hamiltonian path may not exist, due to errors and inadequate sequencing depths.

Formulation 3

- Intuition: to cover all kmers in reads, and require every kmer in the genome is supported.
- Input: dBG (with parameter k) of all reads
- Output: an Eulerian path of dBG

kmer

k=4



G = ATGAACTG ...

Properties of Formulation 3

- Determining if a graph is Eulerian is polynomial-time solvable.
  - Nodes are “balanced” (in-degree = out-degree) except two “semi-balanced” (|in-degree - out-degree| = 1) ones.
- Finding one Eulerian path is also polynomial-time solvable.
  - Start from one semi-balanced node to find a path to another semi-balanced node.
  - Find cycles for unused edges, and merge.
- There are exponential number of optimal solutions.

Why none of them is actually used?

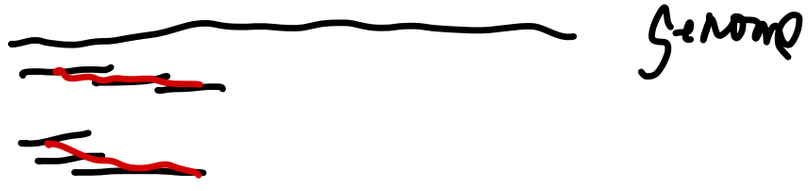
- Too ambitious to reconstruct the full-length genome: it is often that the data does not contain enough information to reconstruct the genome confidently (i.e., to resolve the repeats and to correct all errors).
- Genome assembly is so important that we do not want to make arbitrary decisions — better to keep an incomplete but correct genome, rather than a full-length but wrong one.
- Read: “What do Eulerian and Hamiltonian cycles have to do with genome assembly?” By P. Medvedev & M. Pop, 2021

Practical Assembly Paradigms

- Step 1: producing accurate, refined ~~**contigs**~~ contigs
  - Based on overlap graph: overlap->layout->consensus (OLC)
  - Based on dBG: dBG->refinement->build contigs
- Step 2: **scaffolding** that orders and orients contigs
  - often needs additional information

Consensus in the OLC Paradigm

- Multiple contigs correspond to the same genomic region.



- Use consensus (majority vote) to further correct errors.

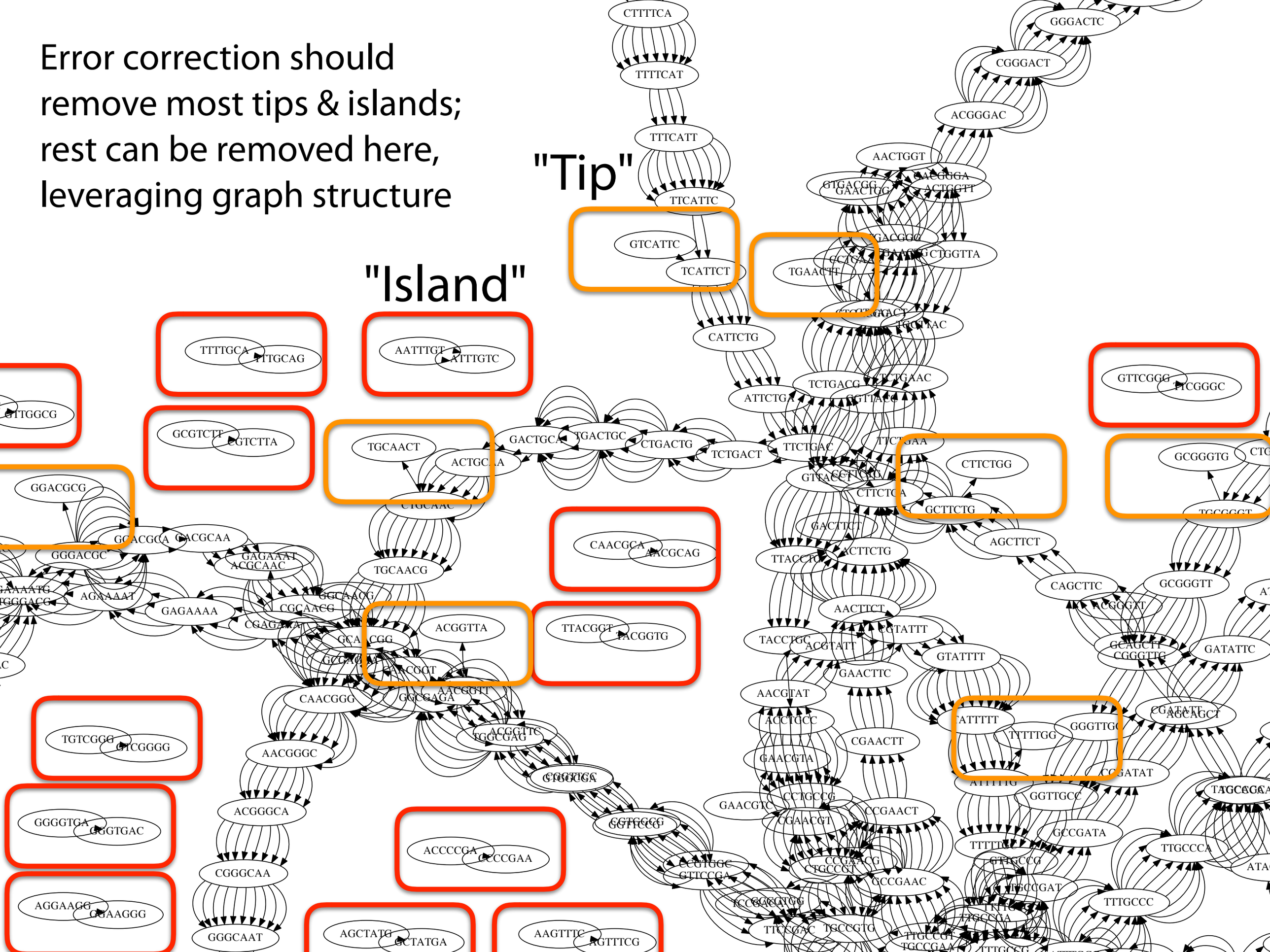
```
GCATTACTC-TCTGGC
GTATTACTCGTCCGGC
GTATTACACGTCTGGC
GTA-TACTCGTCTGGC
```

```
GTATTACTCGTCTGGC
```

Error correction should remove most tips & islands; rest can be removed here, leveraging graph structure

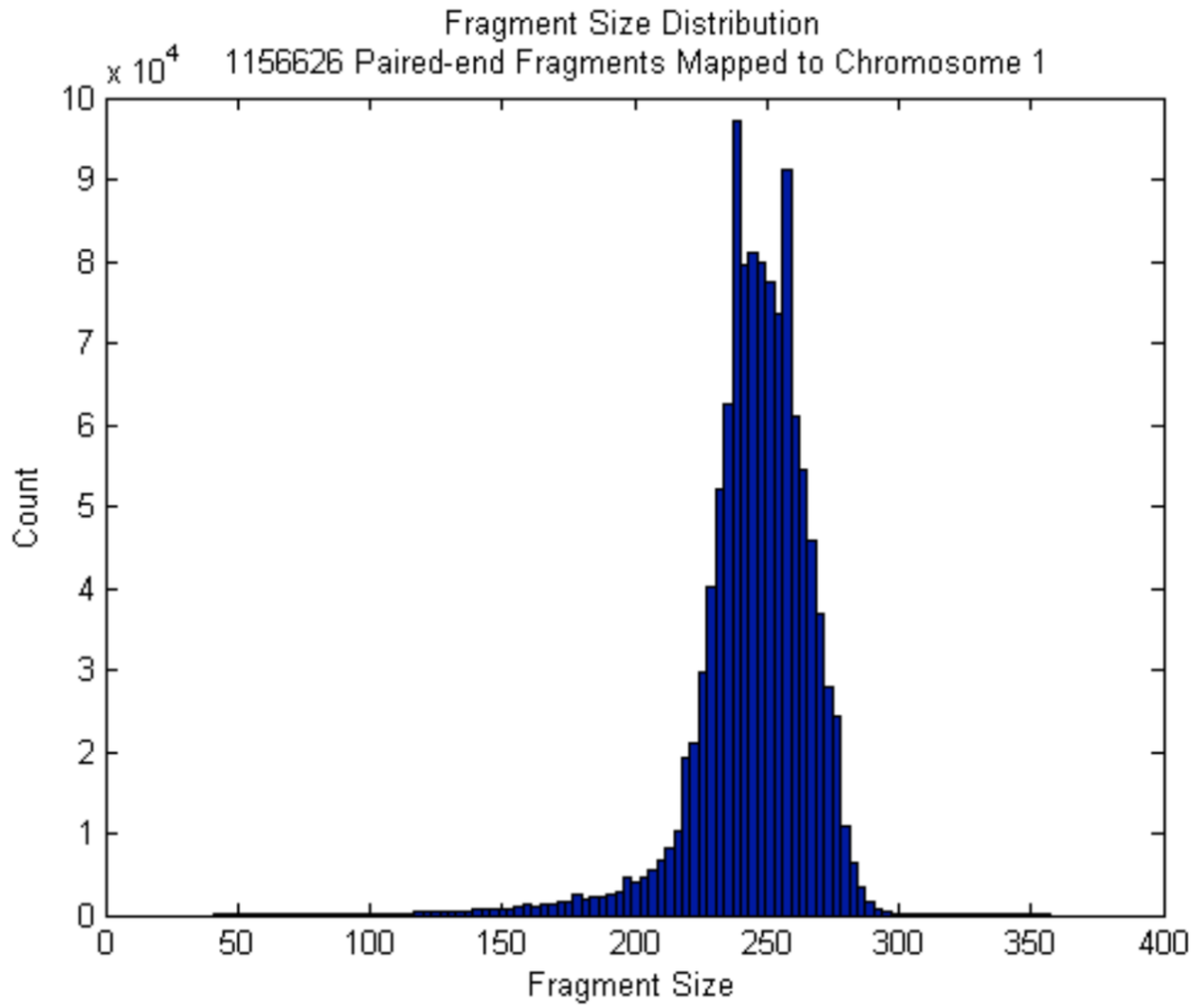
"Tip"

"Island"

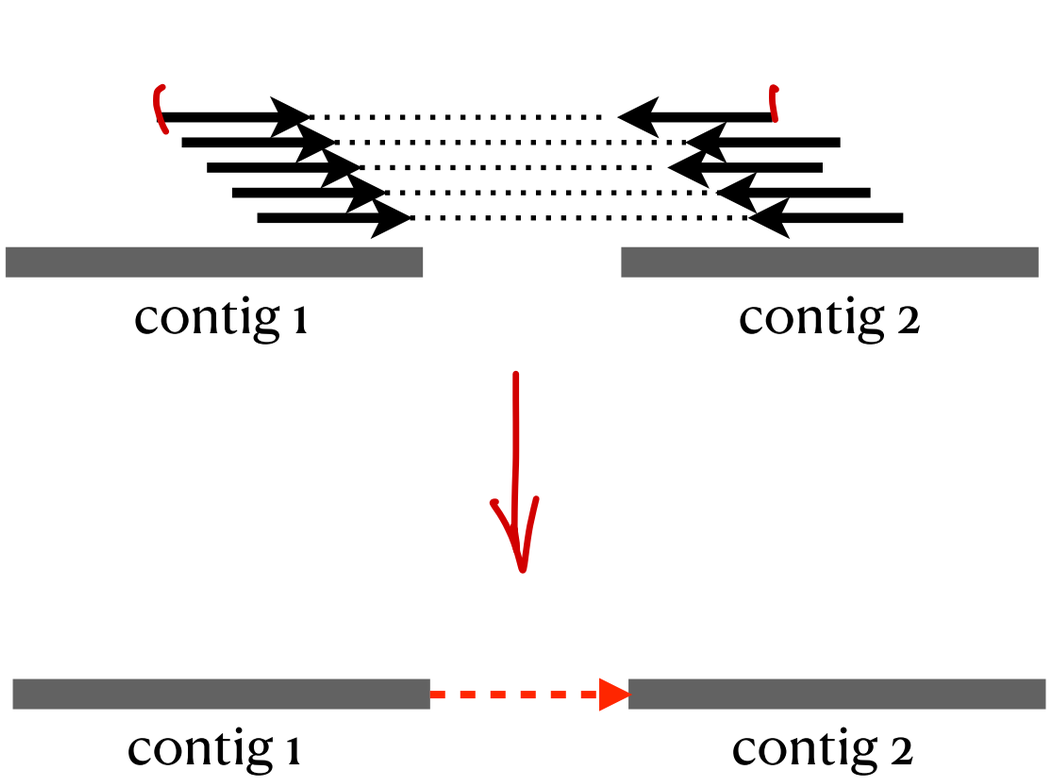


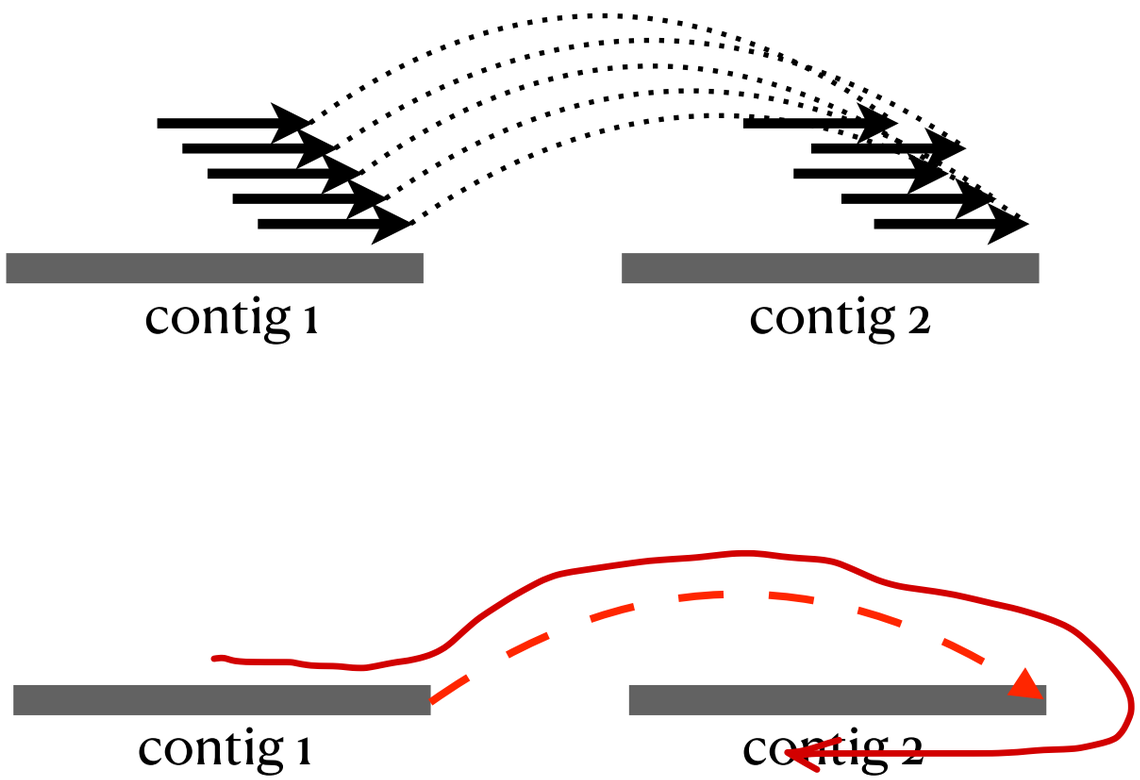
Scaffolding with Paired-end Data

- The length of (unknown) fragments follows a certain distribution.
- Can infer nearby contigs, their orientations, and estimate the distances.



Scaffolding with Paired-end Data





The Complete Human Genome

- High-quality, hybrid dataset
  - PacBio HiFi data: long (~20kbp) and accurate (<1%), which can differentiate slightly diverged repeat regions.
  - ONT data: ultra long (1Mbp), but error rate = 15%, which can span long, identical repeat regions.
- Assembly approach
  - Build an overlap graph from HiFi reads (l = 8kbp).
  - Align ONT reads to the overlap graph, to guide scaffolding.
  - Manual curation.