Comparative Genome Assembly

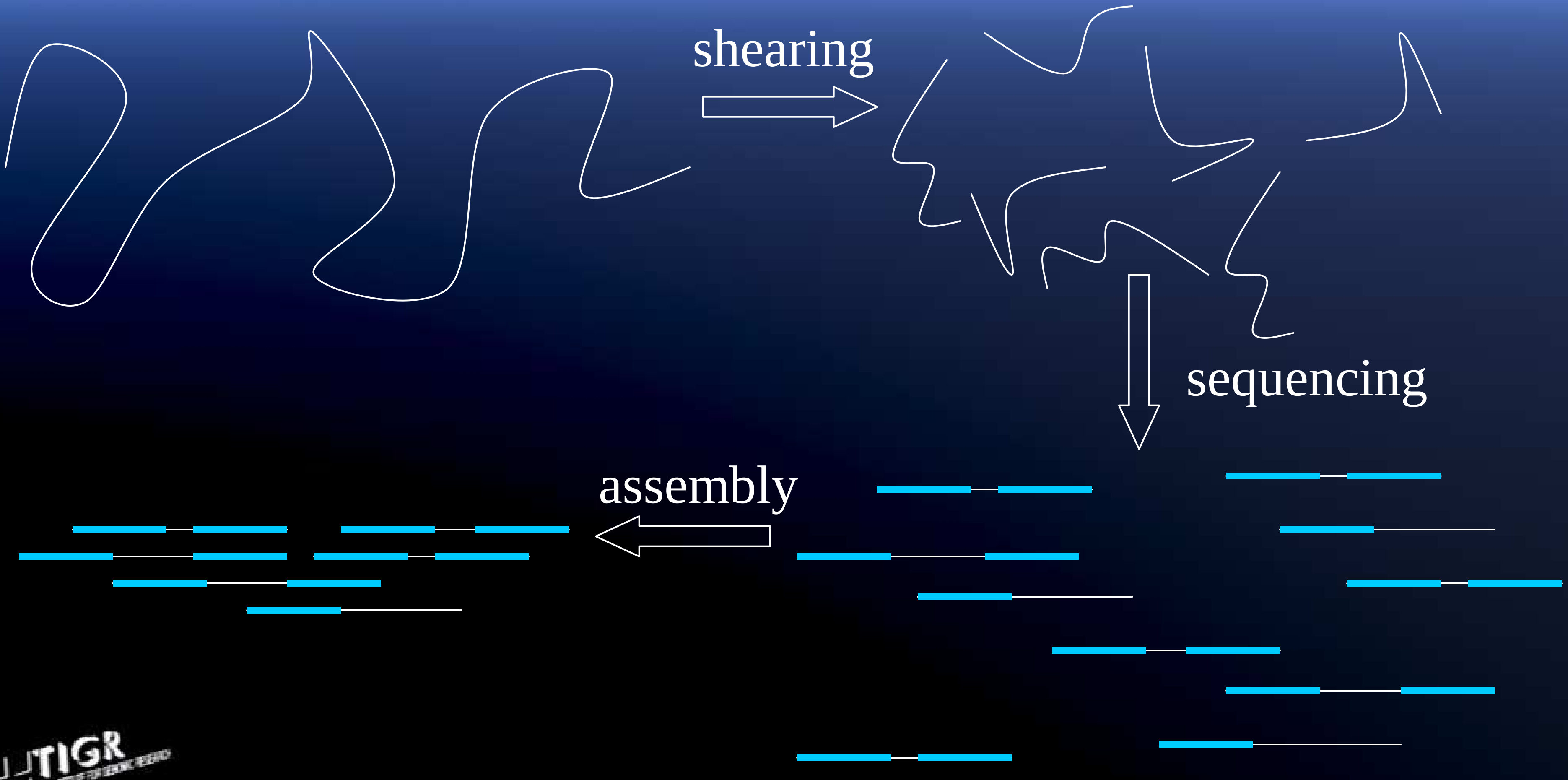
-- and --

Lessons learned while building the first comparative genome assembler, AMOScmp

Adam M Phillippy

Center for Bioinformatics and Computational Biology

WGS sequencing



TIGR THE INSTITUTE FOR GENOMIC RESEARCH

WGS assembly

- Overlap reads
  - identify reads with shared k-mers
  - calculate edit distance
- Layout reads
  - walk the overlap graph
  - hierarchically build contigs
- Generate consensus
  - multi-align read layouts

Limitations of WGS

- Algorithmically hard
  - Overlap reads
    - 70,000 choose 2 = 2.5 billion combinations
      - hard for large eukaryotic genomes
  - Layout reads
    - interpret the overlap graph
      - hard for low coverage projects (too few edges)
      - hard for repetitive projects (too many edges)

AMOScmp overview

- Pick a reference sequence
  - assembly template
- Align target reads to the reference
  - 2.5 billion → 70,000 combinations
- Infer read relationships from alignments
  - if their mappings overlap, they must overlap
- Create read layout
  - fine tune the mappings
- Build a consensus

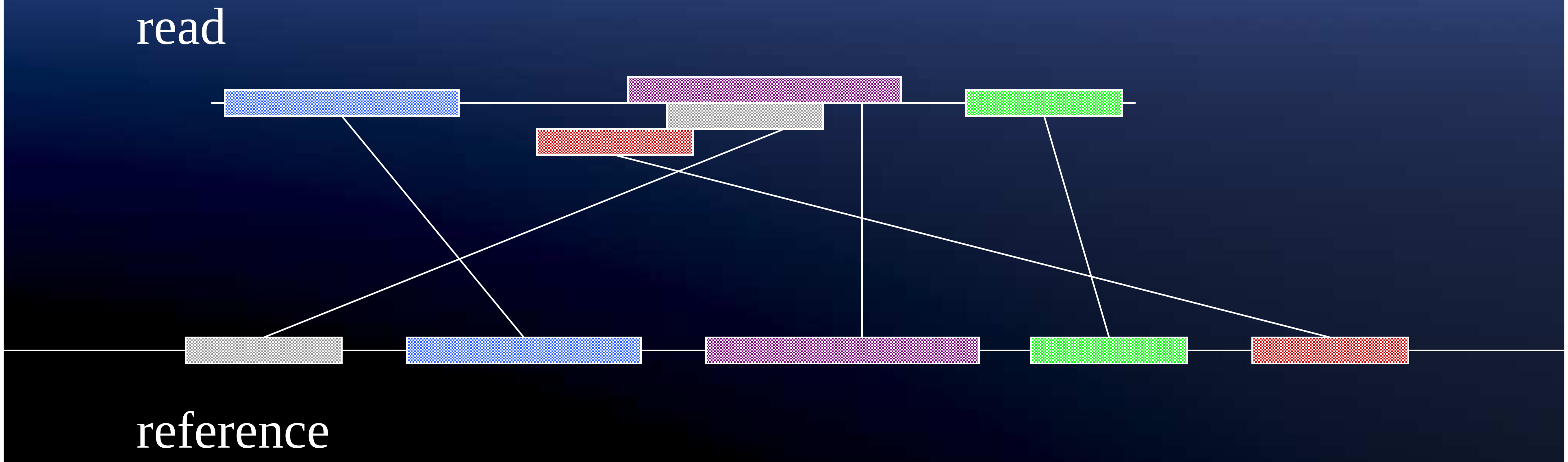
Picking a reference

- The closer the better
  - sequence similarity
    - high identity
  - structural similarity
    - similar repeat distributions
    - few rearrangements
- Preferably complete
  - non-contiguous reference
    - fragmented results
    - forced alignments
    - singletons

Mapping the reads

- Generate read to reference alignments
  - using MUMmer (nucmer)
- Pick the correct alignments
  - using modified LIS algorithm
  - allow fragmented mappings
  - allow multiple, equivalent mappings
- Select repeat copies
  - use mate information
  - “randomly” place leftovers

Read alignments

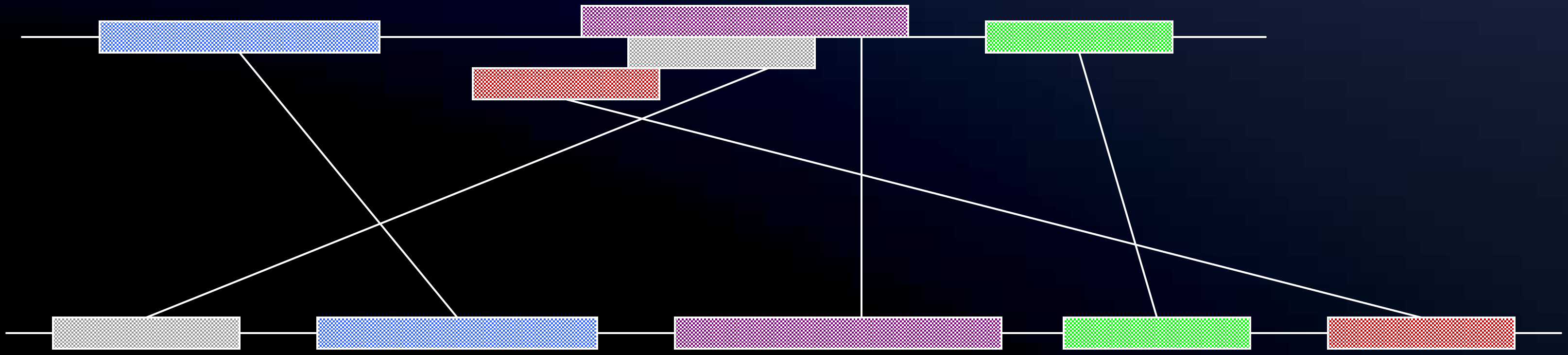


Longest Increasing Subsequence

- Problem
  - For a list of *n* integers, find the longest strictly increasing subsequence from left to right
  - 5 **0** 3 5 **1** **2** **4** **8** 4 **9**
- Complexity
  - *O(n log n)* via greedy set cover
  - $O(n^2)$ via dynamic programming
    - *O(l)* for *n < l / log l*

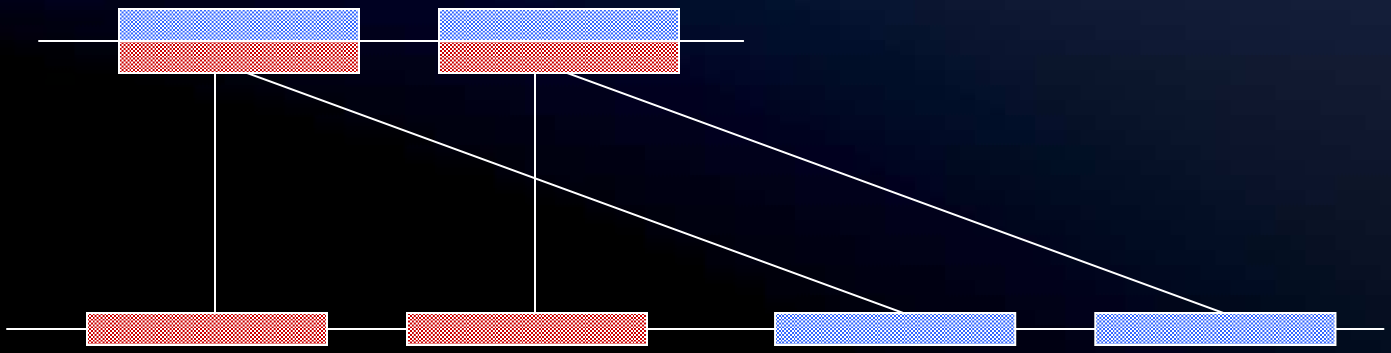
LIS for alignments

- Alignments are not integers
  - $S_i = S_j + (len_i * idy_i) – max(olapR_{ij}, olapQ_{ij})$
    - reward greater length and identity
    - force mutually consistent ordering
    - penalize overlap

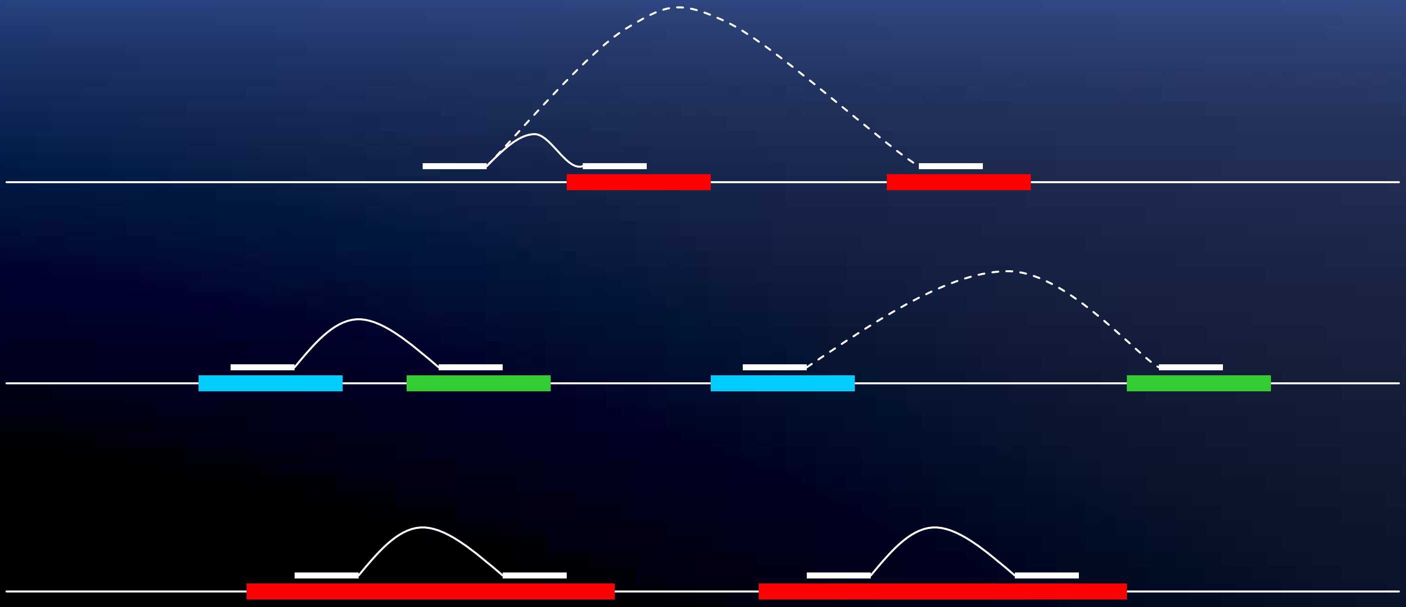


LIS with repeats

- Problem
  - For a list of *n* integers, find a set of disjoint subsequences within a given length of the LIS
  - 1 5 2 6 3 7 4 8 5 9



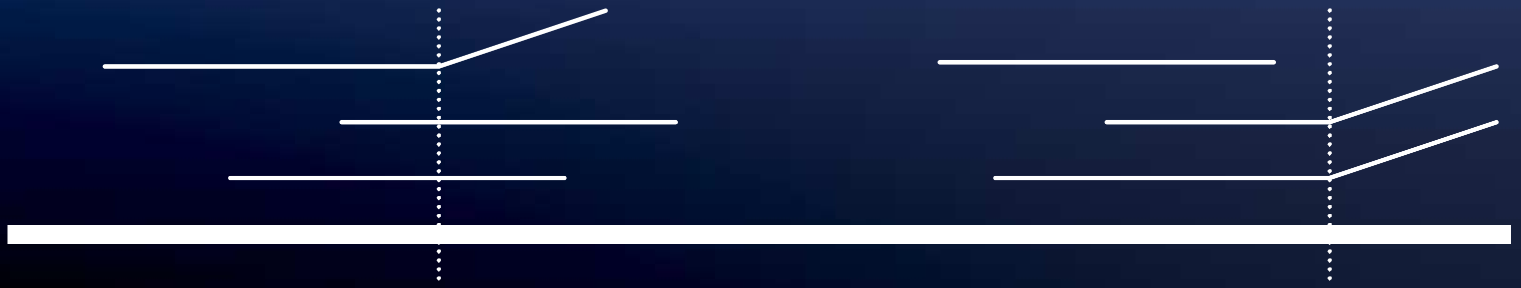
Repeat selection



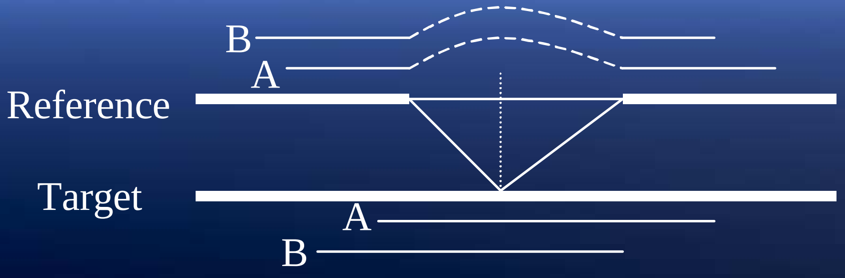
Making the layout

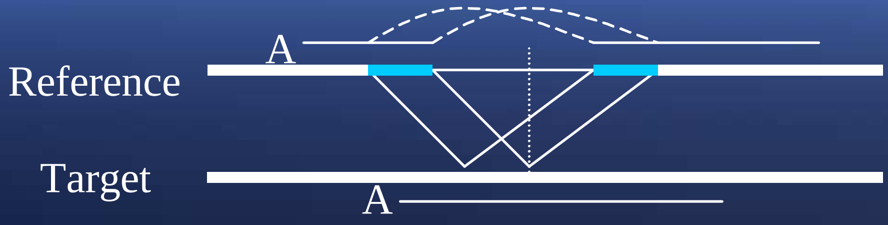
- Locate all alignment breaks
- For each break, count yay and nay reads
  - scan across the reference from left to right
  - read heap contains all the spanning reads
  - count supporting, discounting, fuzzy
    - keep the majority and toss the minority OR toss everything
- Adjust for polymorphism
  - reads inside an insertion need to be handled separately
  - reads after an insertion need to be offset accordingly
- Worst case *O(cr log r)*

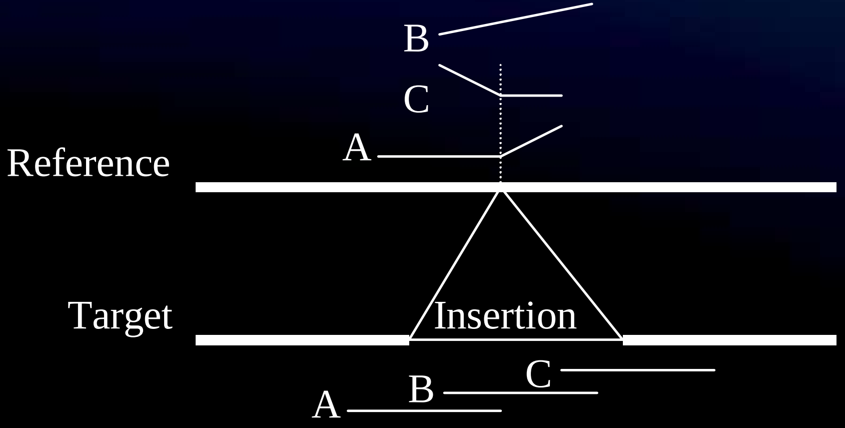
Alignment breaks

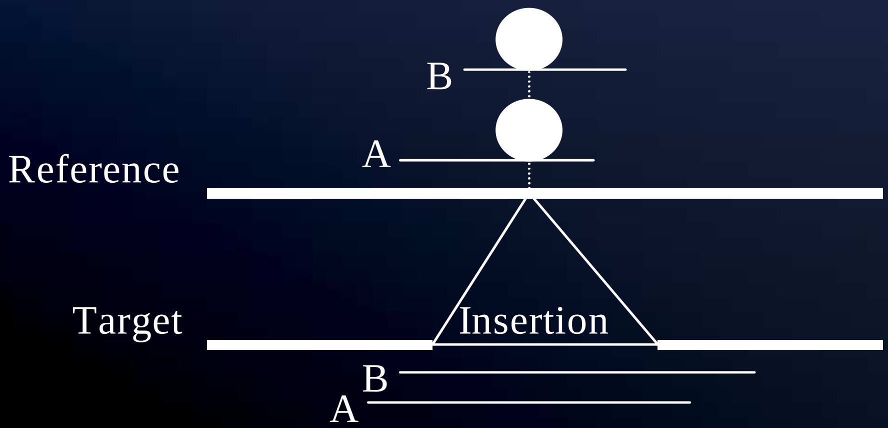


Insertions

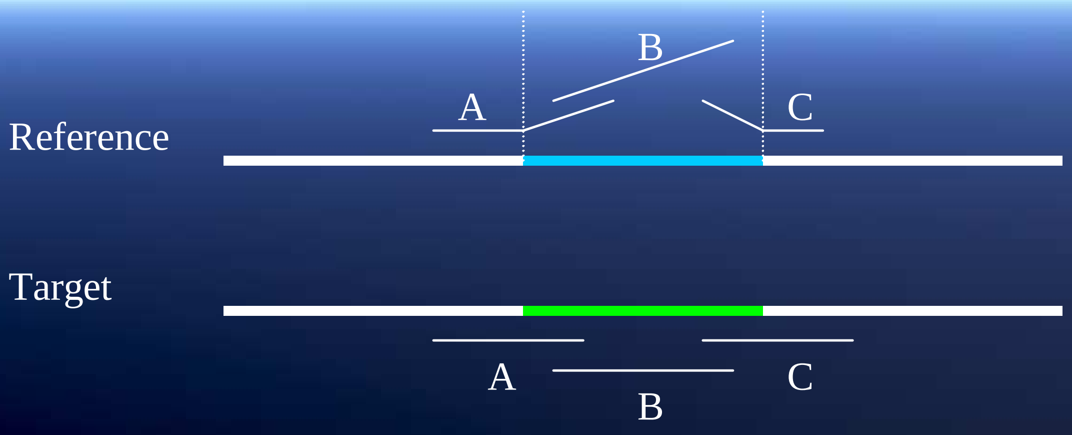

B
A
Reference
Target
A
B

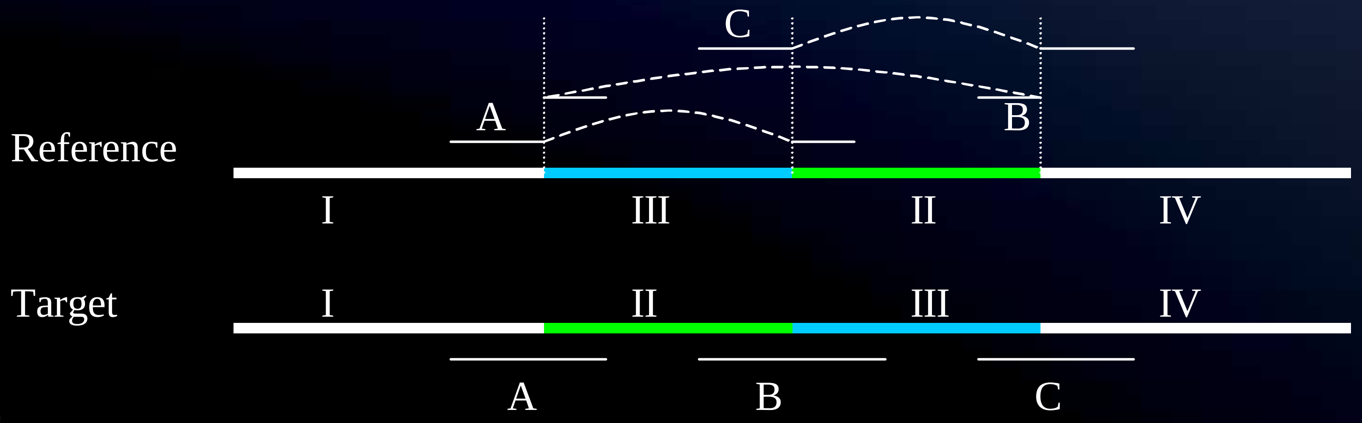

A
Reference
Target
A


B
C
A
Reference
Target
Insertion
A
B
C

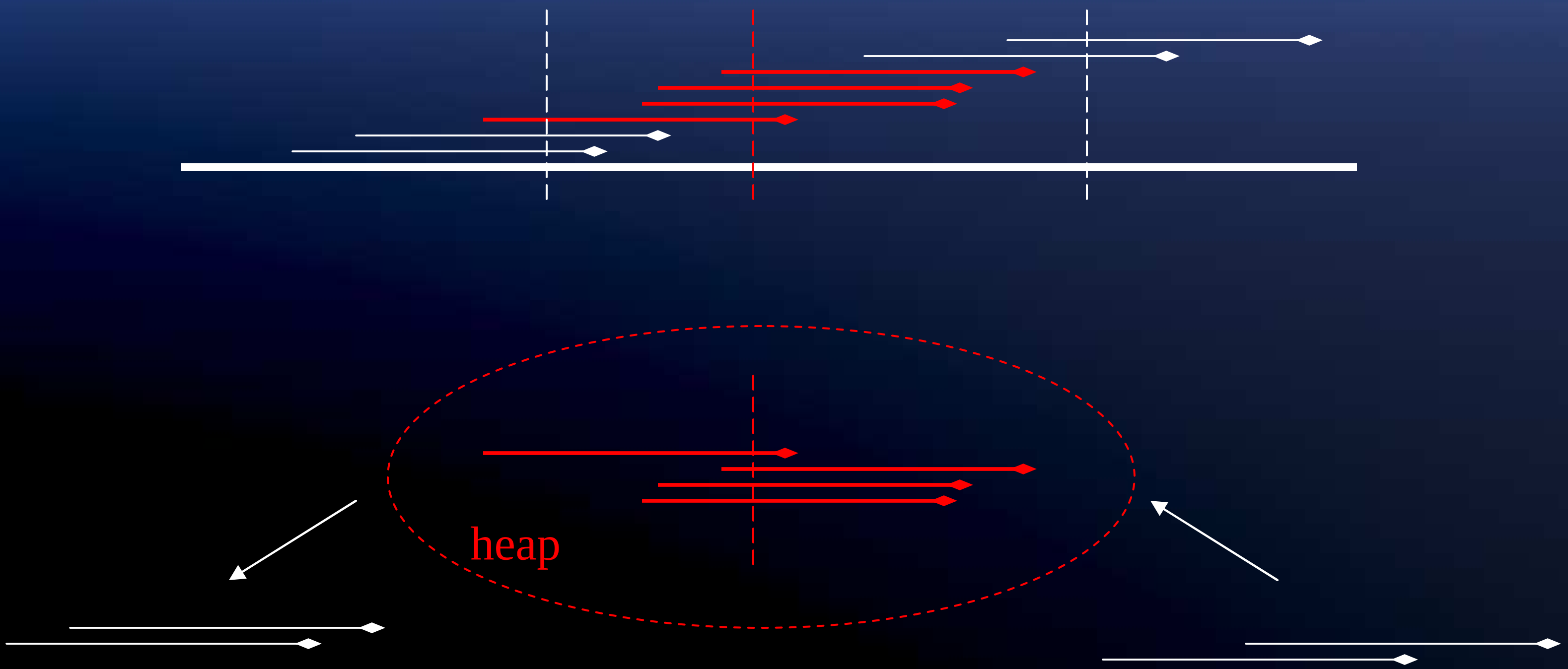

B
A
Reference
Target
Insertion
B
A

Rearrangement

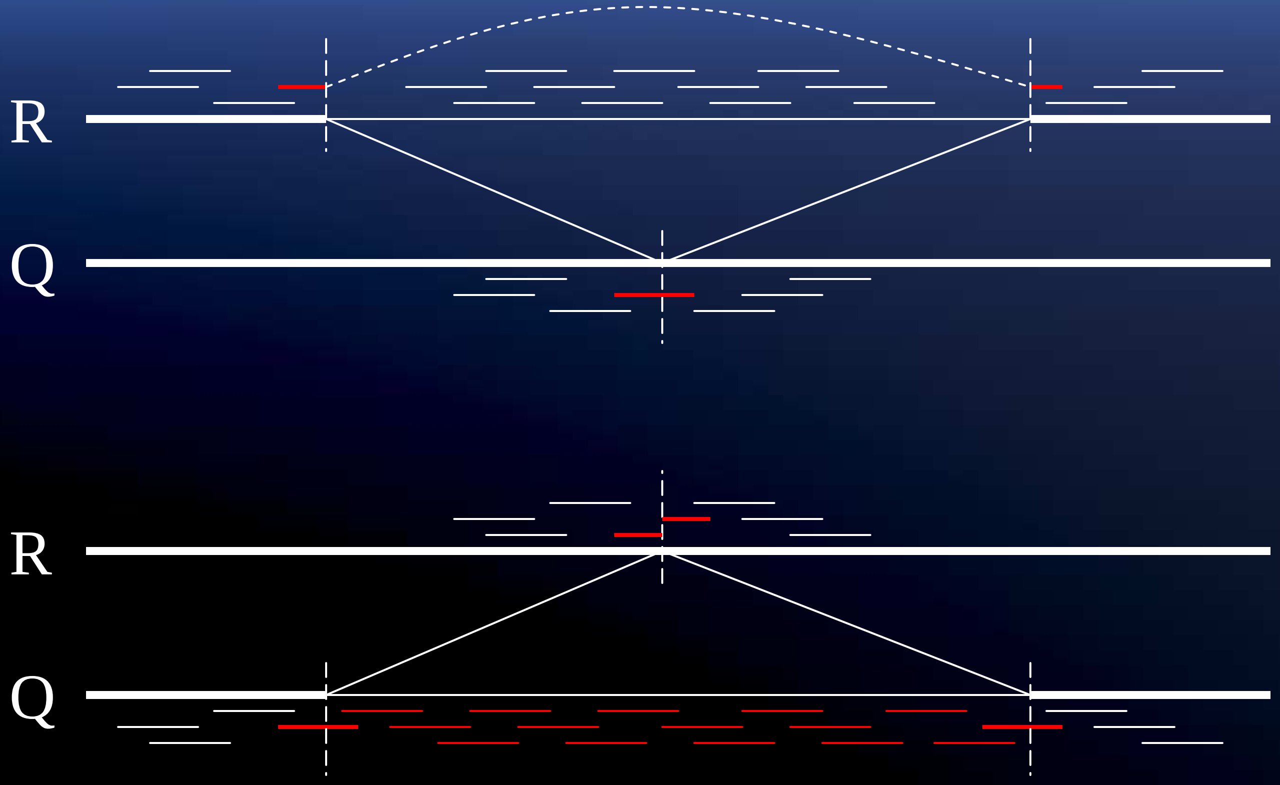




Validating conflicts

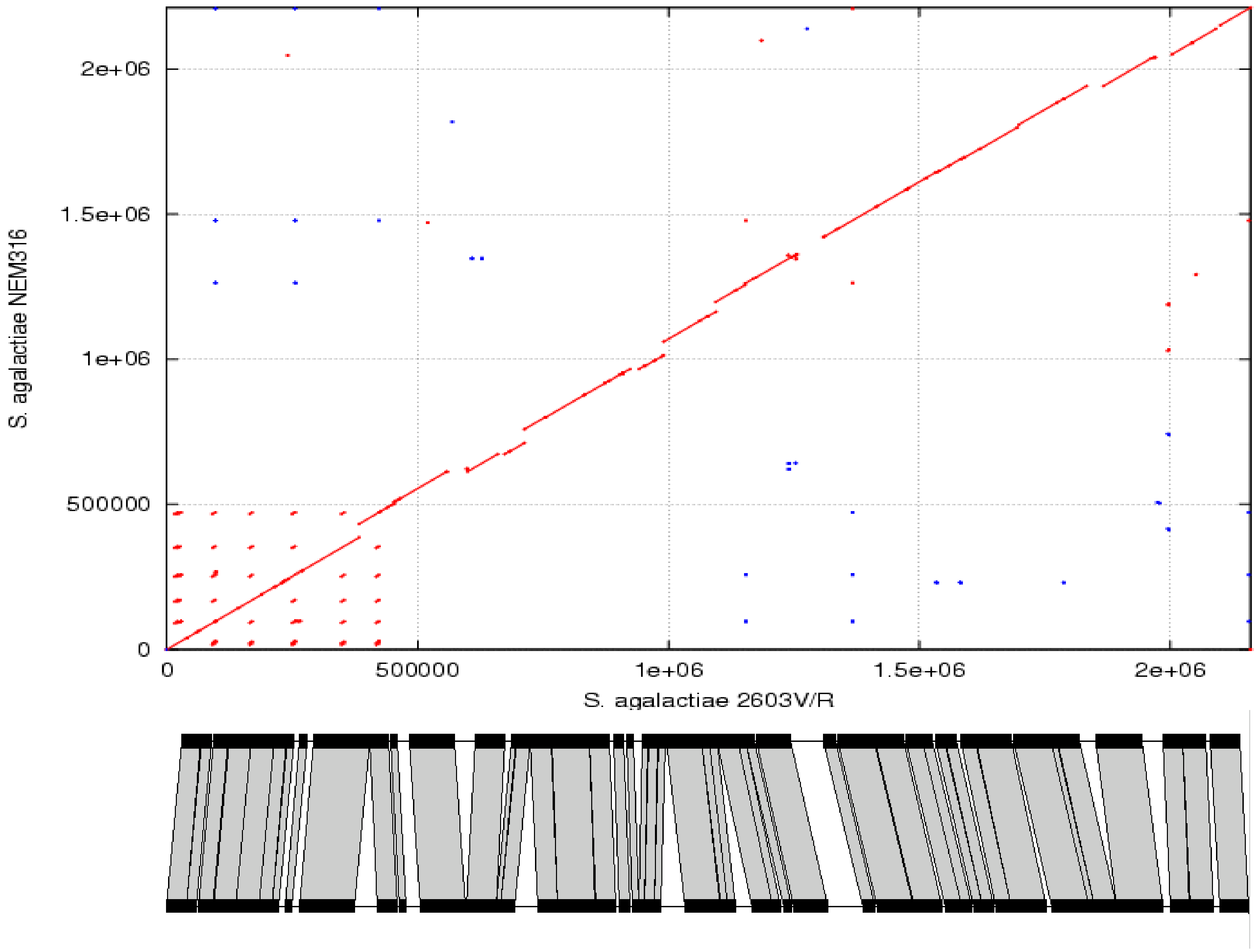


Handling inserts



Example results

- Target
  - *Streptococcus agalactiae* 2603 V/R
- Reference
  - *Streptococcus agalactiae* NEM316
  - *Streptococcus agalactiae* 2603 V/R



S. agalactiae NEM316
S. agalactiae 2603V/R
0
500000
1e+06
1.5e+06
2e+06

2603 read placement

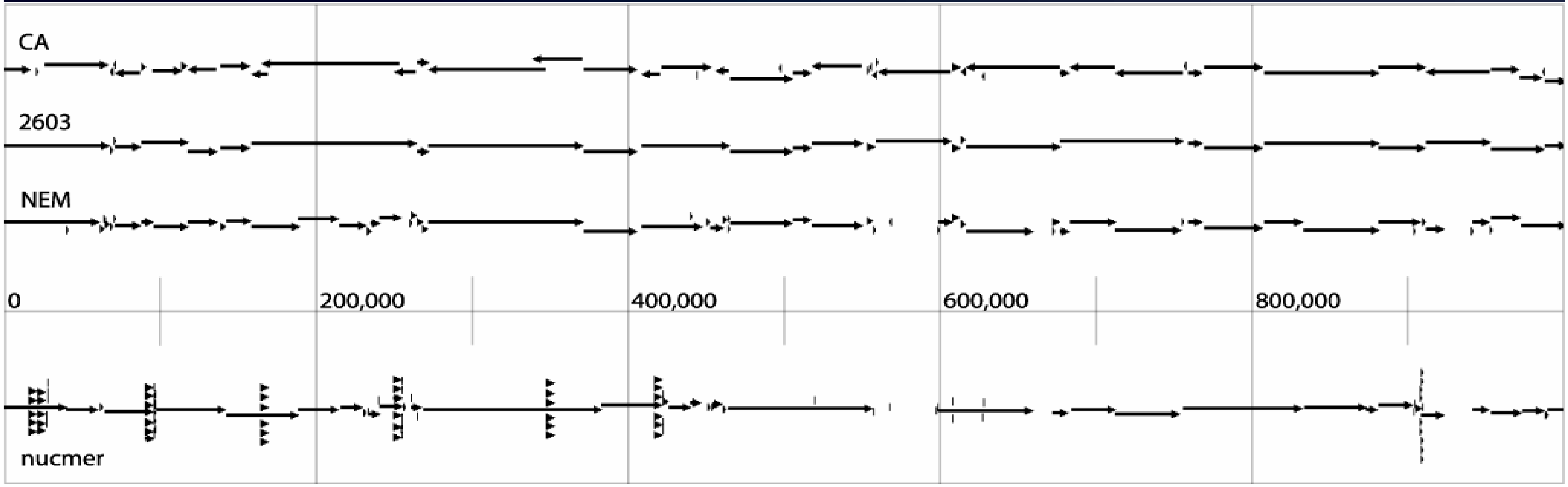
- NEM 316 reference
  - 29,456 alignments
    - ~23,000 after LIS
  - 26,099 total reads
    - 21,816 unique
    - 148 unique mate
    - 22 mate constraints
    - 443 random
    - 3670 unplaced

- Self reference
  - 34,846 alignments
    - ~26,000 after LIS
  - 26,099 total reads
    - 25,301 unique
    - 314 unique mate
    - 22 mate constraints
    - 442 random
    - 20 unplaced

2603 read layout

- NEM 316 reference
  - 312 conflicts
    - 34 accepted
    - 185 rejected
    - 93 unknown
  - 155 contigs

- Self reference
  - 138 conflicts
    - 0 accepted
    - 133 rejected
    - 5 unknown
  - 86 contigs



2603 assembly

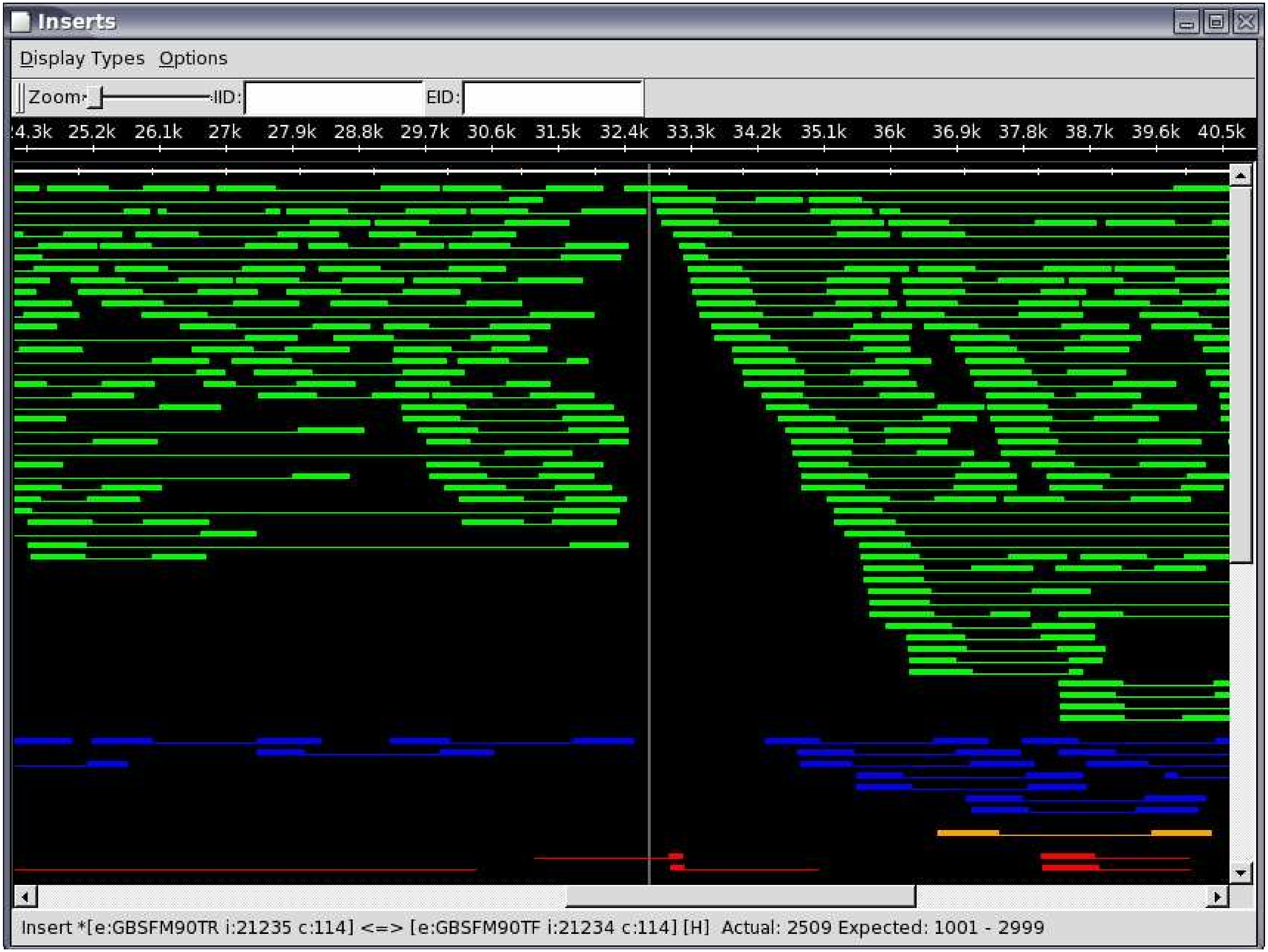
	vs 2603			vs. NEM 316			CelAsm		
X	N	total contig size	N50	N	total contig size	N50	N	total contig size	N50
1	604	1,001,743	0	527	839,315	0	585	903,184	0
2	619	1,593,364	2,294	586	1,393,287	1,479	657	1,488,287	1,595
3	443	1,856,394	5,707	450	1,640,231	4,179	506	1,812,266	4,981
5	243	2,043,842	14,915	277	1,829,976	10,395	293	2,046,730	12,458
7	144	2,100,541	27,364	198	1,891,527	18,142	189	2,110,396	21,926
9	86	2,119,579	42,679	155	1,919,237	24,239	130	2,132,490	33,953

	vs 2603			vs NEM 316			CelAsm			LW
X	gaps	gap size	coverage	gaps	gap size	coverage	gaps	gap size	coverage	coverage
1	588	1,168,208	45.92	511	1,329,996	38.43	562	1,261,419	41.61	39.31
2	596	577,987	73.24	552	778,491	63.96	601	679,386	68.55	74.10
3	430	301,899	86.02	415	530,417	75.45	455	365,736	83.07	89.88
5	232	119,917	94.45	240	347,697	83.90	257	153,824	92.88	98.56
7	132	62,410	97.11	155	292,068	86.48	146	81,406	96.23	99.79
9	80	43,408	97.99	110	270,210	87.49	97	61,544	97.15	99.97

TIGR THE INSTITUTE FOR GENOMIC RESEARCH

Benefits

- Low coverage projects
  - very thin overlaps permissible
    - larger contigs
    - higher assembly confidence
- High coverage projects
  - algorithmically simplified
    - fewer misassemblies
      - given a good reference and implementation
    - greatly reduced time and memory requirements
      - under 5 min / 100 MB for a 5 Mbp genome
  - more reads included in the assembly



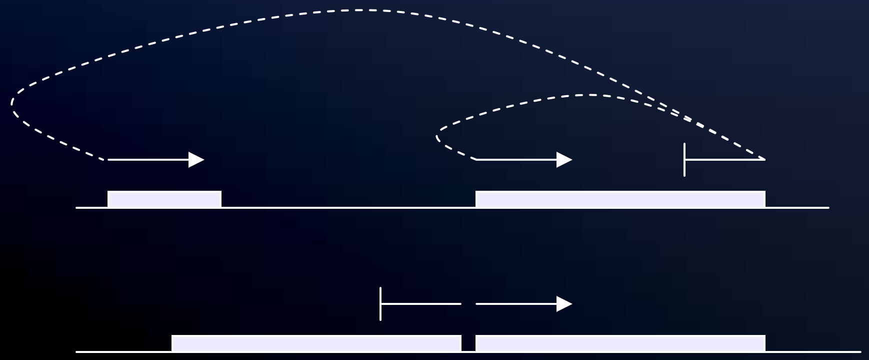
Insert *[e:GBSFM90TR i:21235 c:114] <=> [e:GBSFM90TF i:21234 c:114] [H] Actual: 2509 Expected: 1001 - 2999

Applications

- Low coverage projects
  - thin overlaps make for bigger contigs
  - allow for earlier SNP detection
- Environmental sequencing
  - hybrid assembly of multiple strains
- Short read sequencing
  - traditional algorithms fail for short reads
    - overlaps too short, coverage too deep, non-uniform coverage
- Assembly validation
  - self reference alignment breaks
    - tandem collapse
    - polymorphism

Open questions

- Hybrid assembly
  - conventional / comparative
    - who comes first?
- Read mapping
  - repeats increase runtime
    - sensitivity / specificity
      - exact matches only
- Layout
  - missing sequence
    - inexact repeat copies
      - identity cutoff
      - surrogates
  - polymorphisms
    - query insertions
      - assembly separately
      - bambus
    - rearrangements / tandems
      - examine location



Mihai Pop, Adam Phillippy, Arthur L. Delcher, Steven L. Salzberg. “Comparative genome assembly.” Briefings in Bioinformatics. 2004 Sep; 5(3):237-48.

Mihai Pop

Arthur Delcher

Steven Salzberg

Stefan Kurtz

NIH

Michael Schatz

Pawel Gajer