Celera Assembler

Theory and Practice

Michael Schatz

August 13, 2006

University of Hawaii

Celera Assembler Overview

- Primarily developed in 25 man years by 13 computer scientists at Celera for the private human genome effort.
- Attacks repeats by screening high copy repeats, finding repeat boundaries, and utilizing mate-pair information.
- Currently available as an open source project: http://wgs-assembler.sourceforge.net

Celera Sequencing Factory

The DNA is loaded into automated sequencers. Celera's automated sequencers run 24-7 and have the ability to decipher more than 100 million letters of genetic code per day - the equivalent of 3 percent of the entire human genetic code every day.

The sequencers create an image of the DNA samples being decoded. The four letters of the genetic code -- A, C, T, G -- each are assigned a color.

Celera Sequencing Factory

- 300 ABI 3700 DNA Sequencers
- 50 Production Staff
- 20,000 sq. ft. of wet lab
- 20,000 sq. ft. of sequencing space
- 800 tons of A/C (160,000 cfm)
- $1 million / year for electrical service
- $10 million / month for reagents

Human Data (April 2000)

- Collected 27.27 Million reads = 5.11X coverage
- 21.04 Million are paired (77%) = 10.52 Million pairs

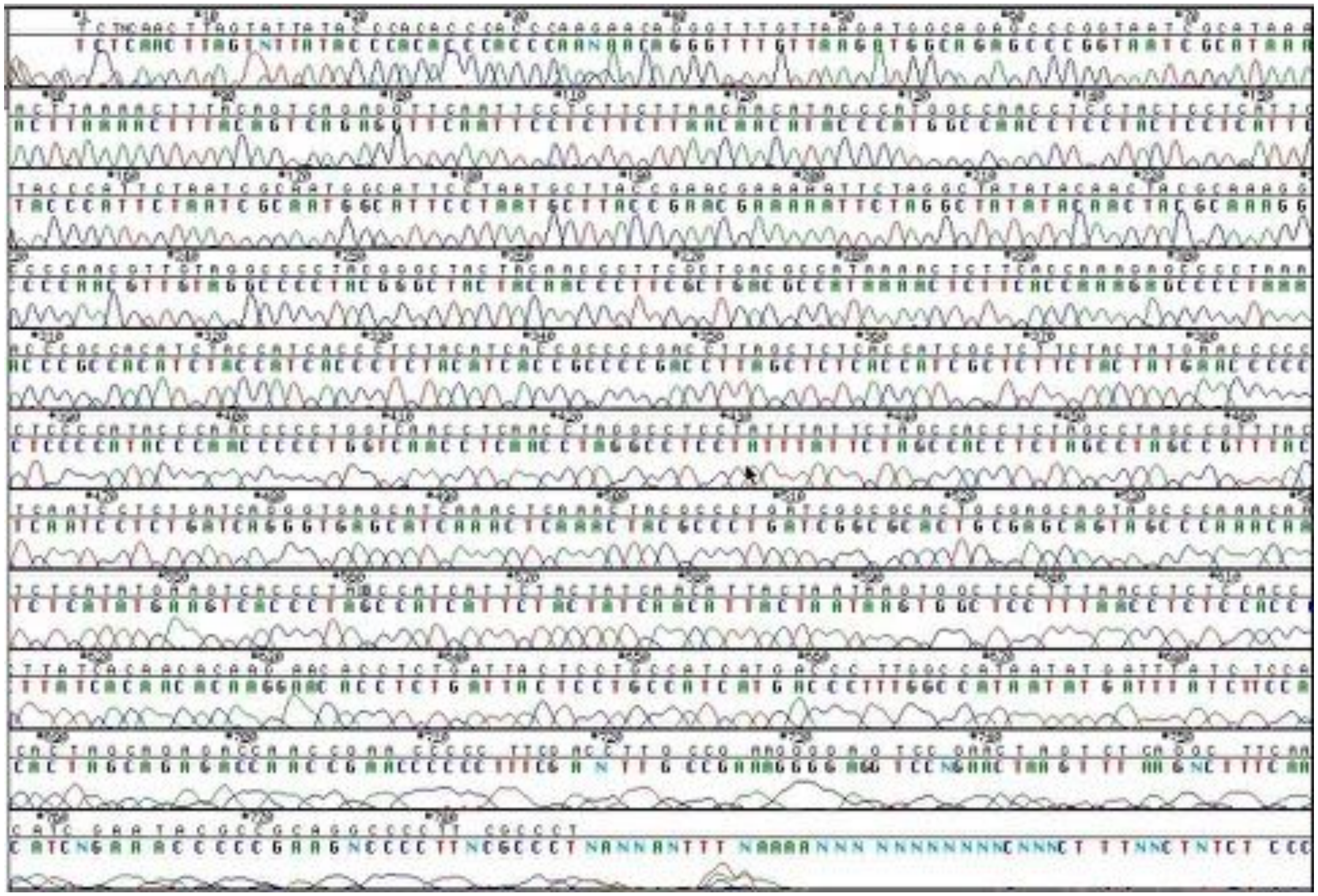
2Kbp	5.045M	98.6% true *	<6% std.dev.
10Kbp	4.401M	98.6% true *	<8% std.dev.
50Kbp	1.071M	90.0% true *	<15% std.dev.

* validated against finished Chrom. 21 sequence

- The clones cover the genome 38.7X times
- Data is from 5 individuals (roughly 3X, 4 x .5X)

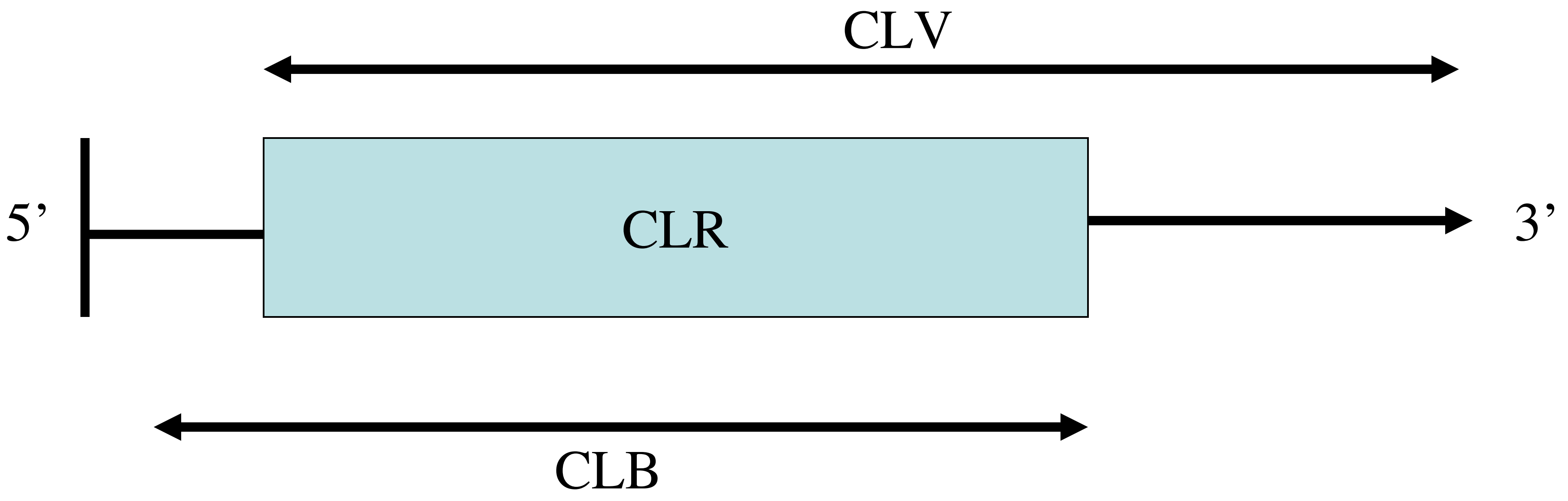
Center for Bioinformatics & Computational Biology

Chromatogram Base Calling



A sequence of basecalls is generated by mapping the recorded peaks to an idealized trace by omitting some peaks, and splitting others.

Trimming



Trimming identifies the regions of good quality for the assembler to use (CLR), as the intersection of the region free of vector (CLV) and the region free of bad quality (CLB).

runCA Pipeline

1. Create Stores
   - gatekeeper
   - PopulateFrgStore
2. Find Repeats
   - meryl
3. Overlap
   - overlap
   - grow-overlap-store
4. Error Correction
   - correct-frags
   - correct-olaps
   - update-erates
5. Unitigging
   - unitigger
   - consensus -U
6. Scaffolding
   - cgw
   - consensus
7. Finalize Data
   - Terminator
   - qc file

runCA Pipeline

1. Create Stores
   - gatekeeper
   - PopulateFrgStore
2. Find Repeats
   - meryl
3. Overlap
   - overlap
   - grow-overlap-store
4. Error Correction
   - correct-frags
   - correct-olaps
   - update-erates
5. Unitigging
   - unitigger
   - consensus -U
6. Scaffolding
   - cgw
   - consensus
7. Finalize Data
   - Terminator
   - qc file

Assembly Stores

- asm.gkpStore - name-id mapping, mate pairs
  - populated by gatekeeper
  - dump with dumpGatekeeper (output in STDERR)
- asm.frgStore - bases, qualities, clear range
  - populated by PopulateFragStore
  - dump with dumpFragStore
- asm.ovlStore - overlaps between reads
  - populated by grow-olap-store
  - dump with dump-olap-store

runCA Pipeline

1. Create Stores
   - gatekeeper
   - PopulateFrgStore
2. Find Repeats
   - meryl
3. Overlap
   - overlap
   - grow-overlap-store
4. Error Correction
   - correct-frags
   - correct-olaps
   - update-erates
5. Unitigging
   - unitigger
   - consensus -U
6. Scaffolding
   - cgw
   - consensus
7. Finalize Data
   - Terminator
   - qc file

Meryl: k-mer statistics

Frequent k-mer statistics: asm.mers

count	>325
22-mer sequence	AAAGCCCAAAGCCCAAAGCCCA
	>228
	AACAGCTCGATCACGTCGCTGT

How much of the DNA is in 300 copies or more?

```
% grep ’>’ asmbl.mers | sed ’s/>//’ | awk ’{if ($1>300) sum+= $1} END {print sum;}’
```

Not every repeat is mis-assembled,
but repeats cause (almost) every mis-assembly.

runCA Pipeline

1. Create Stores
   - gatekeeper
   - PopulateFrgStore
2. Find Repeats
   - meryl
3. Overlap
   - overlap
   - grow-overlap-store
4. Error Correction
   - correct-frags
   - correct-olaps
   - update-erates
5. Unitigging
   - unitigger
   - consensus -U
6. Scaffolding
   - cgw
   - consensus
7. Finalize Data
   - Terminator
   - qc file

Overlap between two sequences

overlap (19 bases) overhang (6 bases)

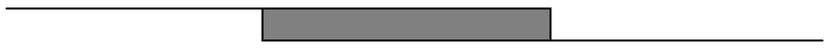
```
...AGCCTAGACCTACAGGATGCGCGGACACGTAGCCAGGAC
             CAGTACTTGGATGCGCTGACACGTAGCTTATCCGGT...
```

overhang % identity = 18/19 % = 94.7%

overlap - region of similarity between regions
overhang - un-aligned ends of the sequences

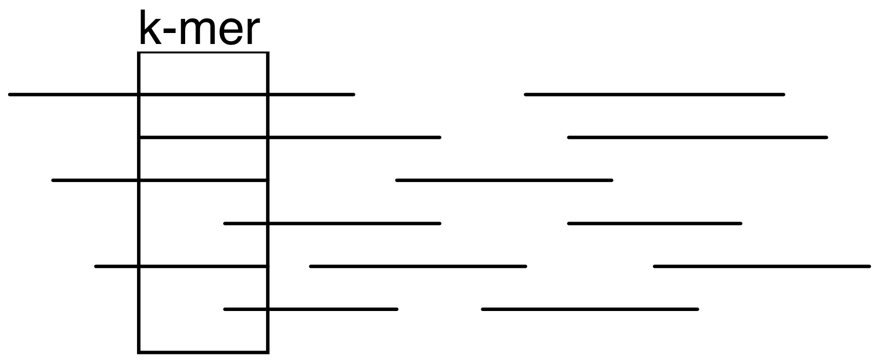
The assembler screens merges based on:
• length of overlap
• % identity in overlap region
• maximum overhang size.

Defines **dove-tail** overlap



All pairs alignment

- Needed by the assembler
- Try all pairs – must consider ~ n^2 pairs
- Smarter solution: only n x coverage (e.g. 8) pairs are possible
  - Build a table of k-mers contained in sequences (single pass through the genome)
  - Generate the pairs from k-mer table (single pass through k-mer table)



Overlapper

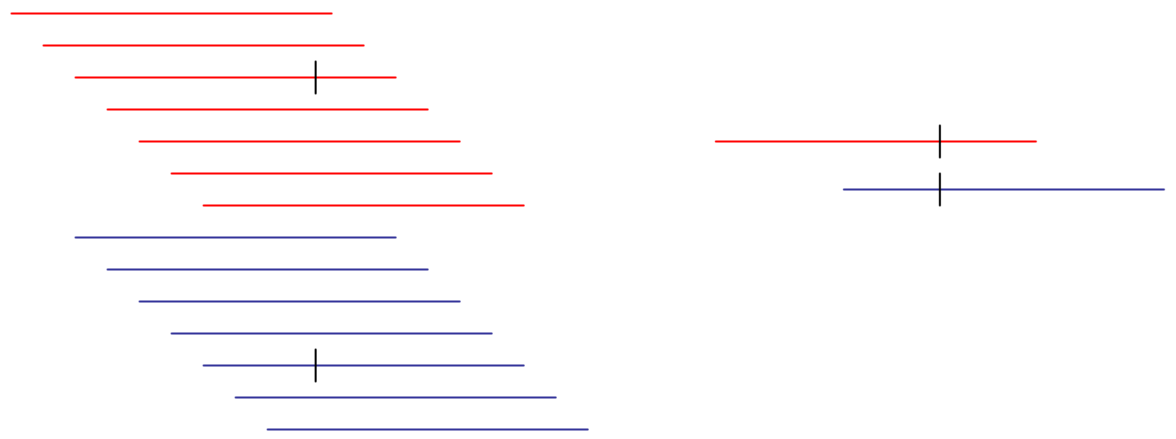
- Find all overlaps ≥ 40bp allowing 6% mismatch.
- Use k-mer (k=22) seed matches with O(nd) extension where extension quits when probability of seeing given # of errors for amount of sequence aligned is less than 1 in a million.
- Avoid seeding overlaps with k-mers whose occurrence >= 100 in the trimmed read set.
- Multiple threads & multiple instances allowed depending on the input size.

Overlapper & Screening

- High copy repeats are filtered by excluding high copy (>= 100) 22-mers as seeds.

- Warning:

  Sequencing error can accidentally cause low copy number seeds in high copy repeat regions creating low coverage unitigs of collapsed repeats.



runCA Pipeline

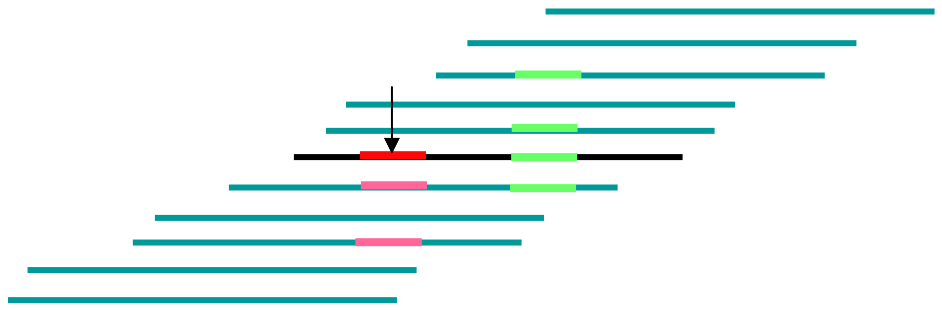
1. Create Stores
   - gatekeeper
   - PopulateFrgStore
2. Find Repeats
   - meryl
3. Overlap
   - overlap
   - grow-overlap-store
4. Error Correction
   - correct-frags
   - correct-olaps
   - update-erates
5. Unitigging
   - unitigger
   - consensus -U
6. Scaffolding
   - cgw
   - consensus
7. Finalize Data
   - Terminator
   - qc file

Error Correction

If a k-mer (k=10) matches a k-mer from an overlapping read then the bases in the k-mer of the read are confirmed.

If a base is not confirmed and the 1-neighborhood of an overlapping k-mer matches it then there is a vote for correction. The majority correction vote is applied to the sequence.

Note: Sequences are not actually changed, only overlaps are re-evaluated as single base pair errors are "corrected".



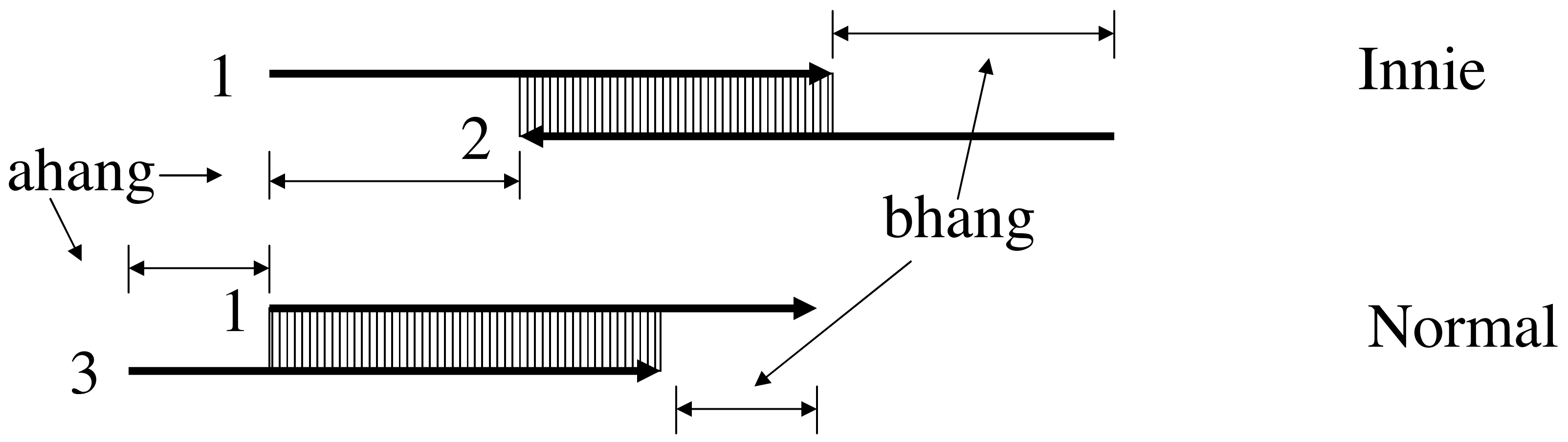
ACGTACCGATATGACAC

ACGTACCGTTATGACAC

ACGTACCGATATGACAC

ACGTACCGATATGACAC

dump-olap-store



```
1 2 I  ahang  bhang 1.3 0.1
1 3 N -ahang -bhang 2.3 0.4
```

Original Error Rate

After Error Correction

Overlap degrees

8x coverage: each read overlaps approx. 8 reads off of each end

ahang < 0 - overlap off of 5’ end
bhang > 0 - overlap off of 3’ end

```
% awk ’{if ($4 < 0) end5++; if ($5 > 0) end3++;} END {print end5, end3}}’ asm.overlaps
```

end5 overlaps > end3 overlaps - normal (3’ end is “dirtier”)
end5 overlaps < end3 overlaps - possible vector trimming problem

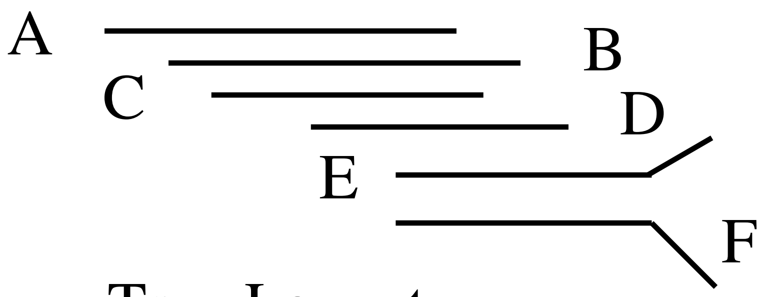
`% awk ’{print $1}’ asm.overlaps | sort -u | wc -l` - # reads with overlaps

many reads w/o overlaps - trimming problem or ubiquitous repeat

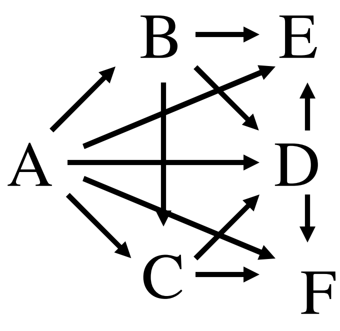
runCA Pipeline

1. Create Stores
   - gatekeeper
   - PopulateFrgStore
2. Find Repeats
   - meryl
3. Overlap
   - overlap
   - grow-overlap-store
4. Error Correction
   - correct-frags
   - correct-olaps
   - update-erates
5. Unitigging
   - unitigger
   - consensus -U
6. Scaffolding
   - cgw
   - consensus
7. Finalize Data
   - Terminator
   - qc file

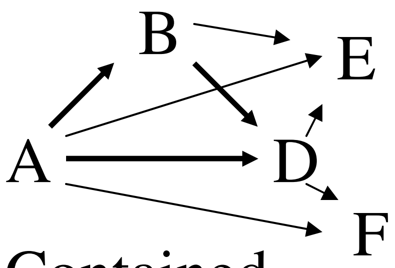
Unitigging



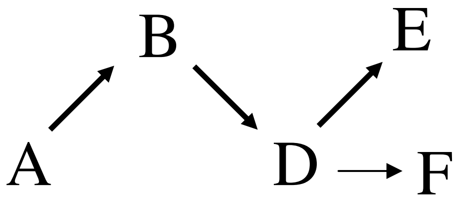
True Layout



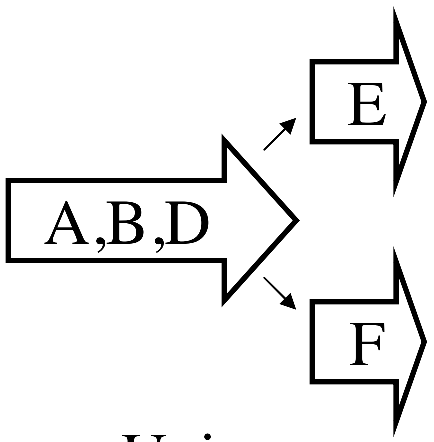
Original Overlap Graph



Contained Read Removal



Transitive Edge Removal



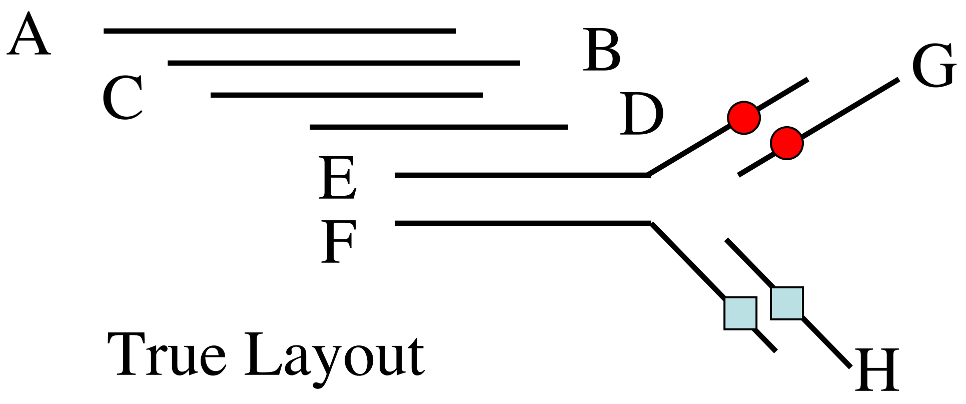
Unique Join Collapsing

Theorem: SCS of unitigs = SCS of reads

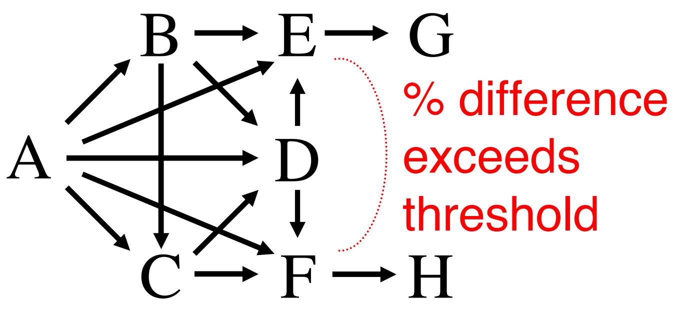
Revised Unitigging

- Exact Unitigging is computationally expensive
- Instead CA unitigger finds the “best” overlap on each end of each read—its “best buddy”.
- Unitigs are chains of mutually unique best buddies—adjacent reads are best buddies of each other and of no other read.
- This takes time and space linear in the number of reads.
- In rare cases results are different from graph reduction.
  - Low coverage regions
  - High fidelity repeat copies

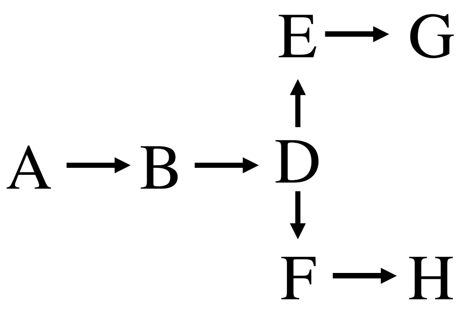
Best Buddy Unitigging



True Layout

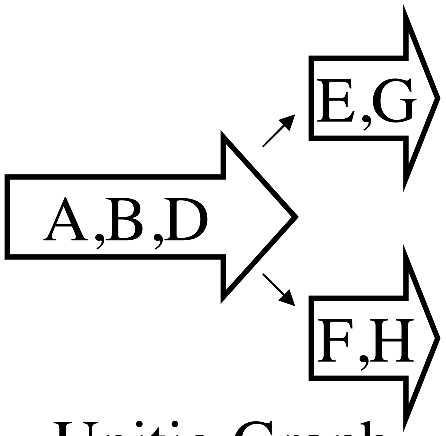


Original Overlap Graph



$$\begin{tikzcd}
 & & E \arrow[r] & G \\
A \arrow[r] & B \arrow[r] & D \arrow[u] \arrow[d] & \\
 & & F \arrow[r] & H
\end{tikzcd}$$

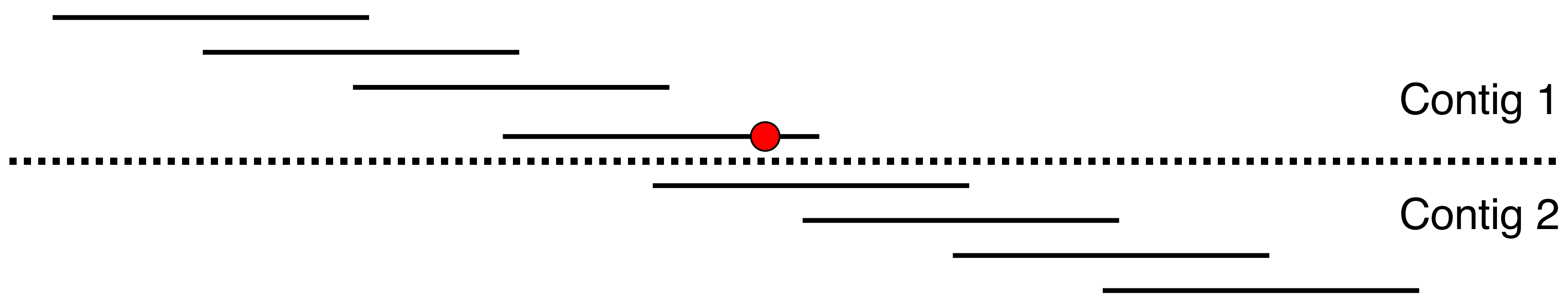
Best Buddy Graph



Unitig Graph

Threshold set with unitigger –e (ERATE, utgErrorRate)

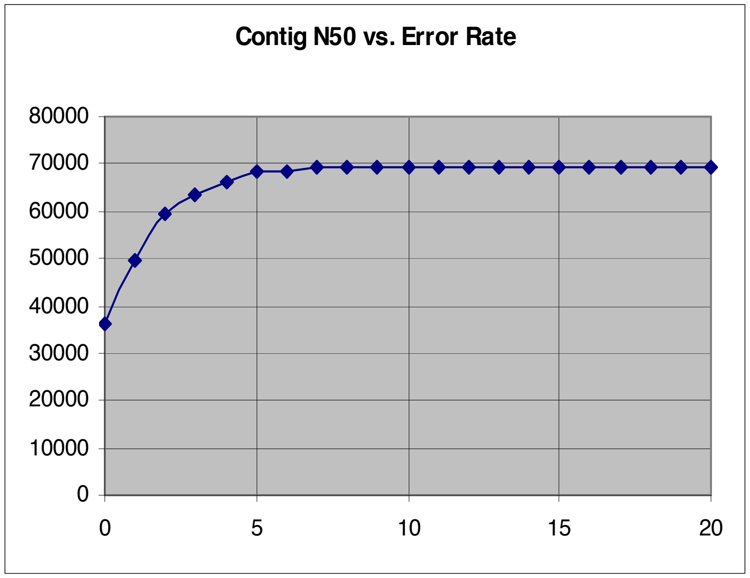
False Negatives: Sequencing Error, Trimming

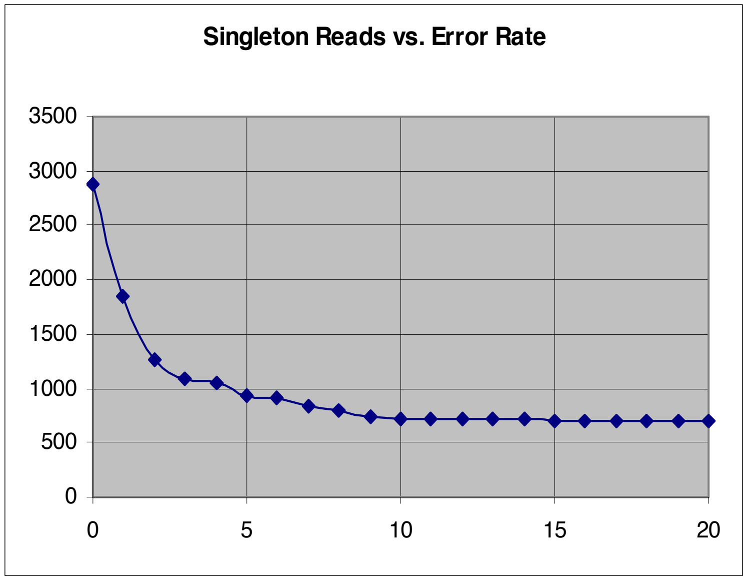


- Overlaps are “missed” if the overlapping basecalls have sequencing error beyond the threshold.

- Assembly is fragmented into smaller chunks, or reads left as singletons.

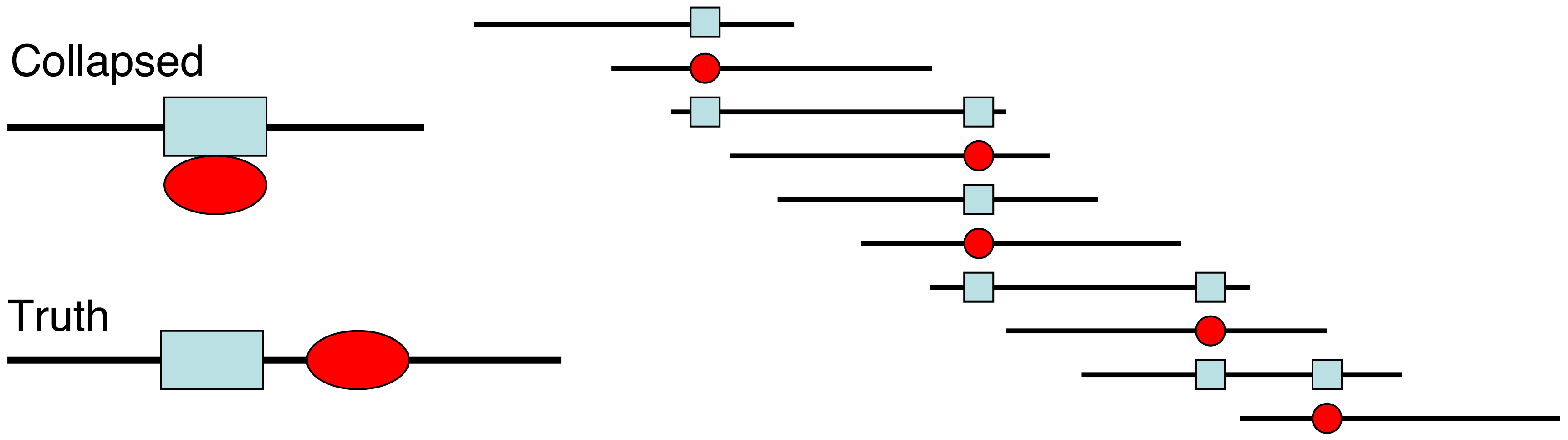
Sequencing Error Effect





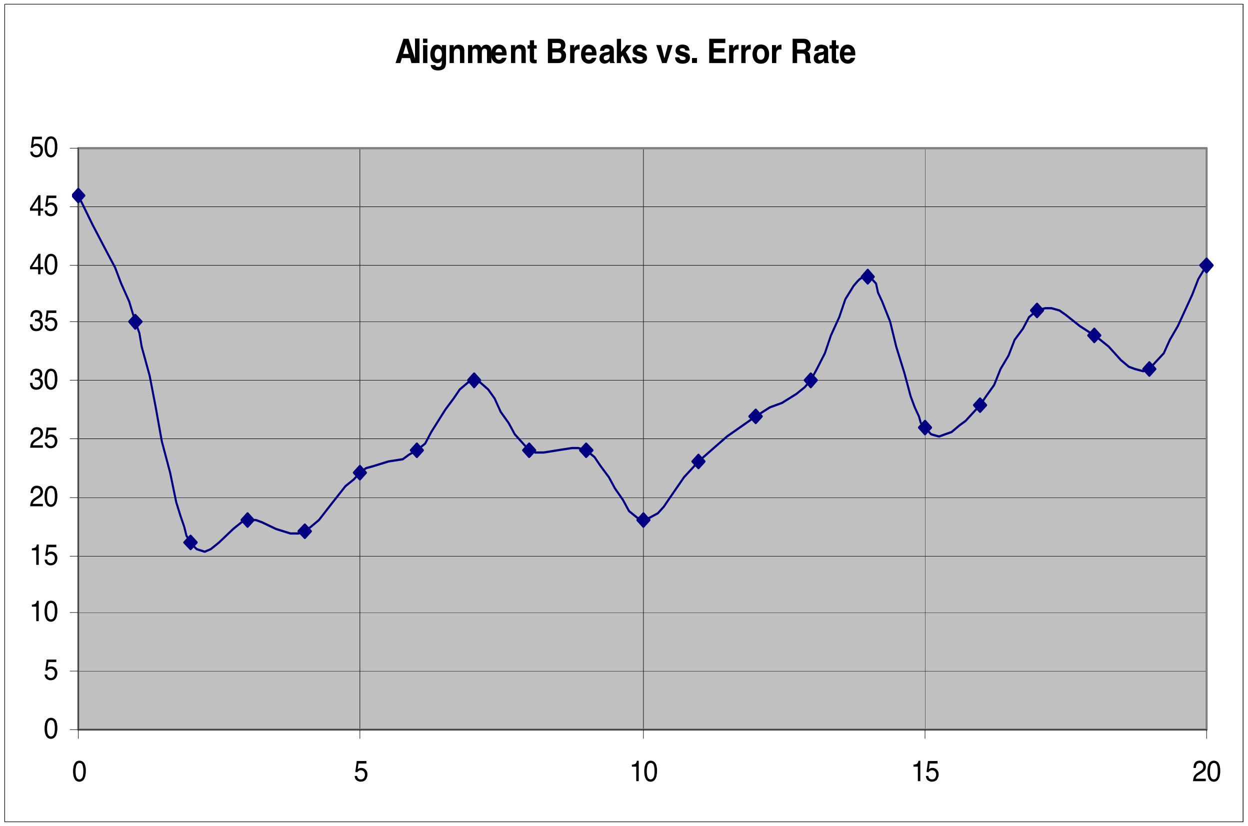
In general, contigs get larger and more reads are placed as the error rate threshold is increased.

False Positives: Repeats



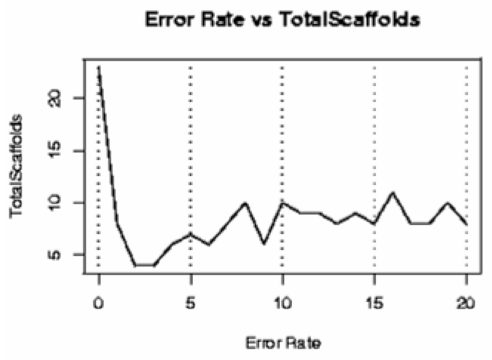
- Reads originating in different copies will “falsely” overlap if % difference between repeats is less than threshold.
- Genome is mis-assembled as reads from different repeat copies are collapsed together as the unitigger becomes less sensitive to slight differences between repeats.

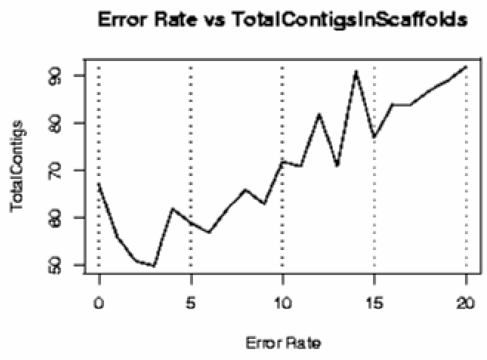
Repeat Effect

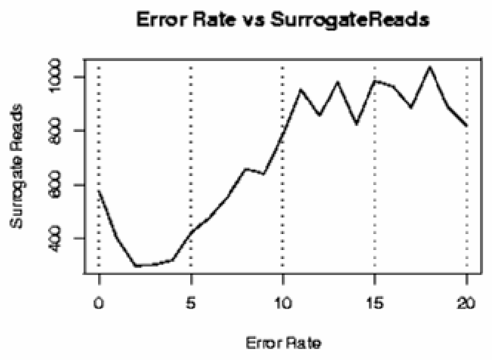


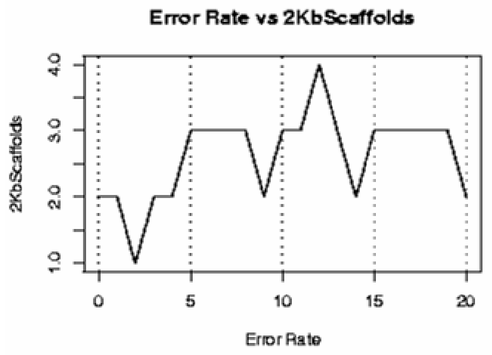
In general, more repeats are mis-assembled as the error rate threshold is increased.

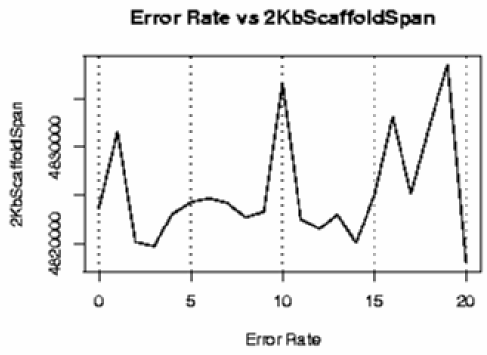
Unitig Error Rate Impact

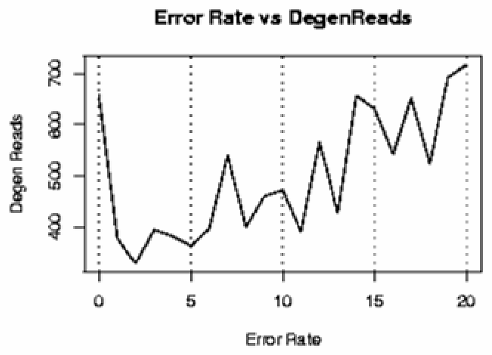


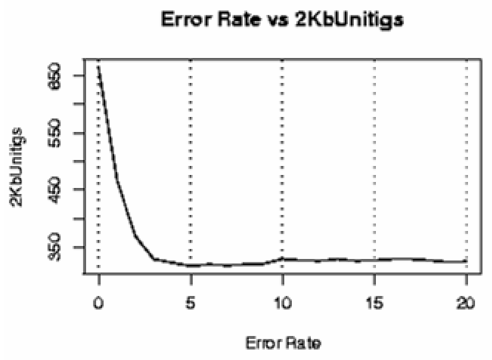


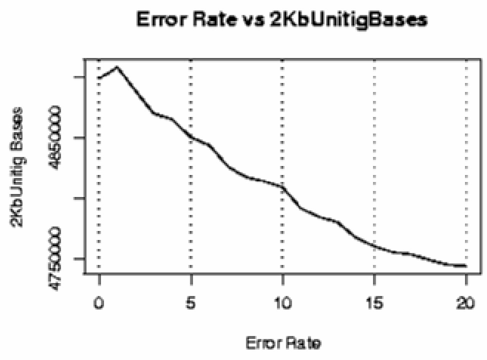


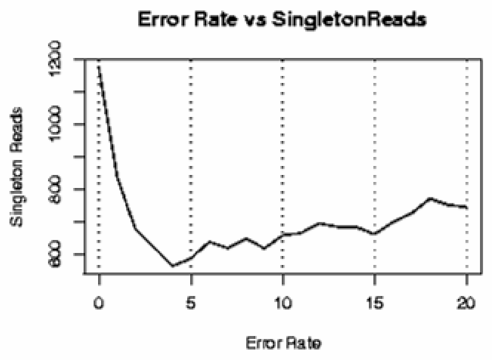




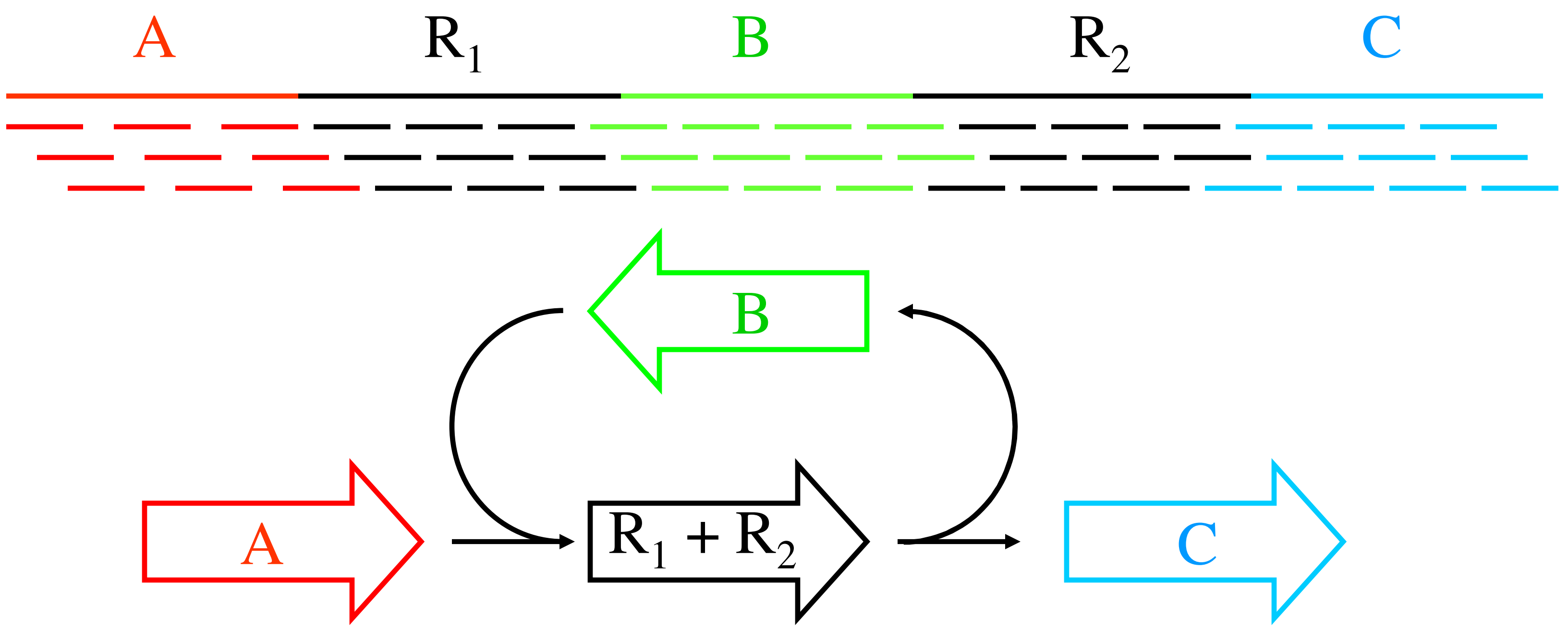




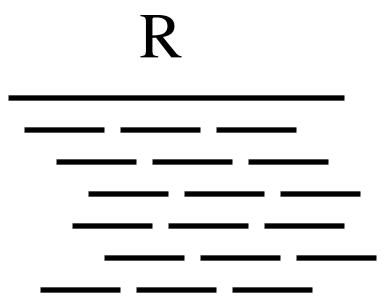




Unitig Scoring

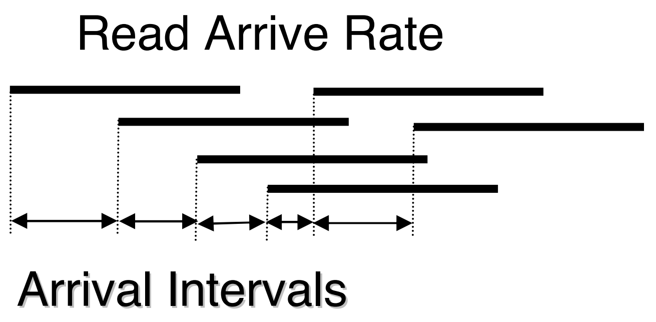


The arrival rate of reads within repeat unitig R is statistically higher than for unique unitigs A, B or C. The corresponding A-stat will mark the unitig as unreliable.



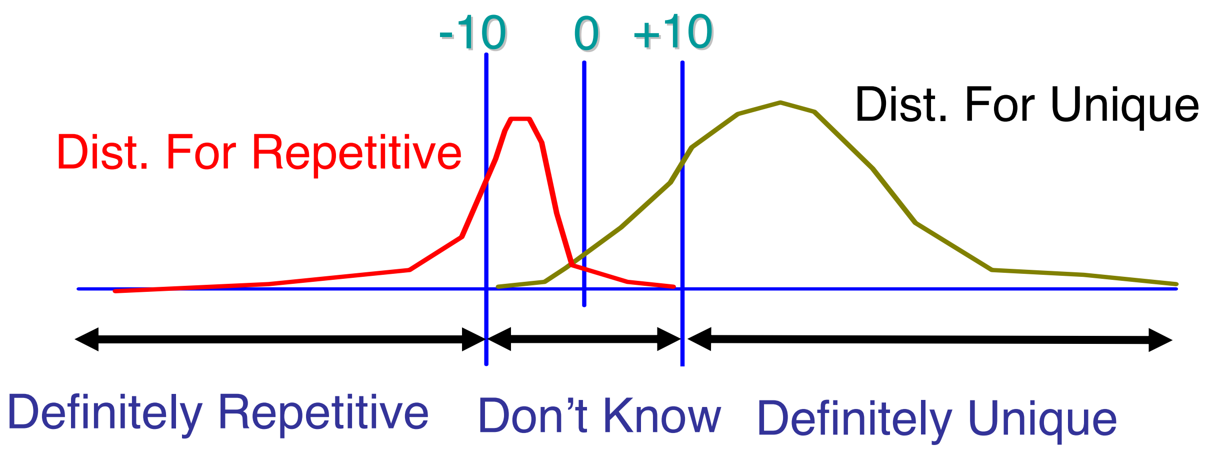
Note: Requires uniform distribution of reads.

Identifying Unique DNA



Expected Coverage is:
(Sum of read lengths) /
Genome Size

Discriminator A-Statistic is log odds ratio of probability unitig is for unique DNA versus 2-copy DNA.



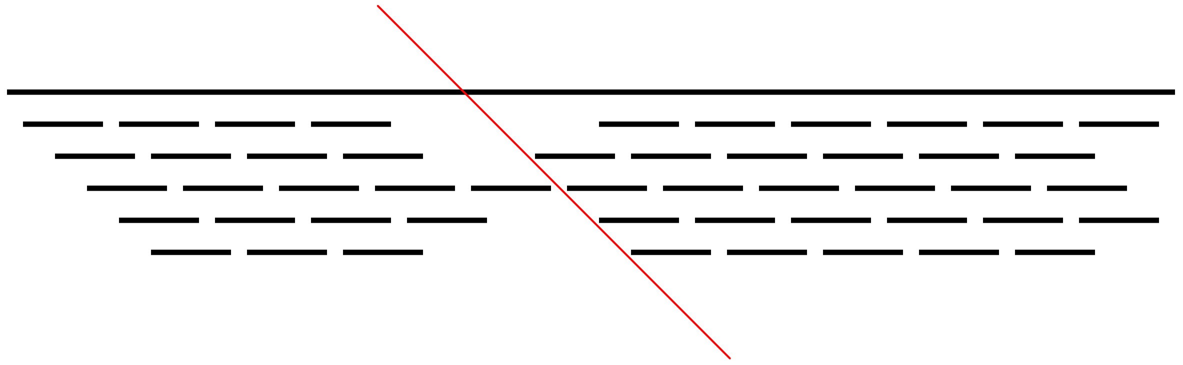
Correct for biases:

- cgw –j (ASTAT) : set threshold for definitely unique
- unitigger –l (utgGenomeLen) : adjust genome size estimate, boost borderline unitigs

runCA Pipeline

1. Create Stores
   - gatekeeper
   - PopulateFrgStore
2. Find Repeats
   - meryl
3. Overlap
   - overlap
   - grow-overlap-store
4. Error Correction
   - correct-frags
   - correct-olaps
   - update-erates
5. Unitigging
   - unitigger
   - consensus -U
6. Scaffolding
   - cgw
   - consensus
7. Finalize Data
   - Terminator
   - qc file

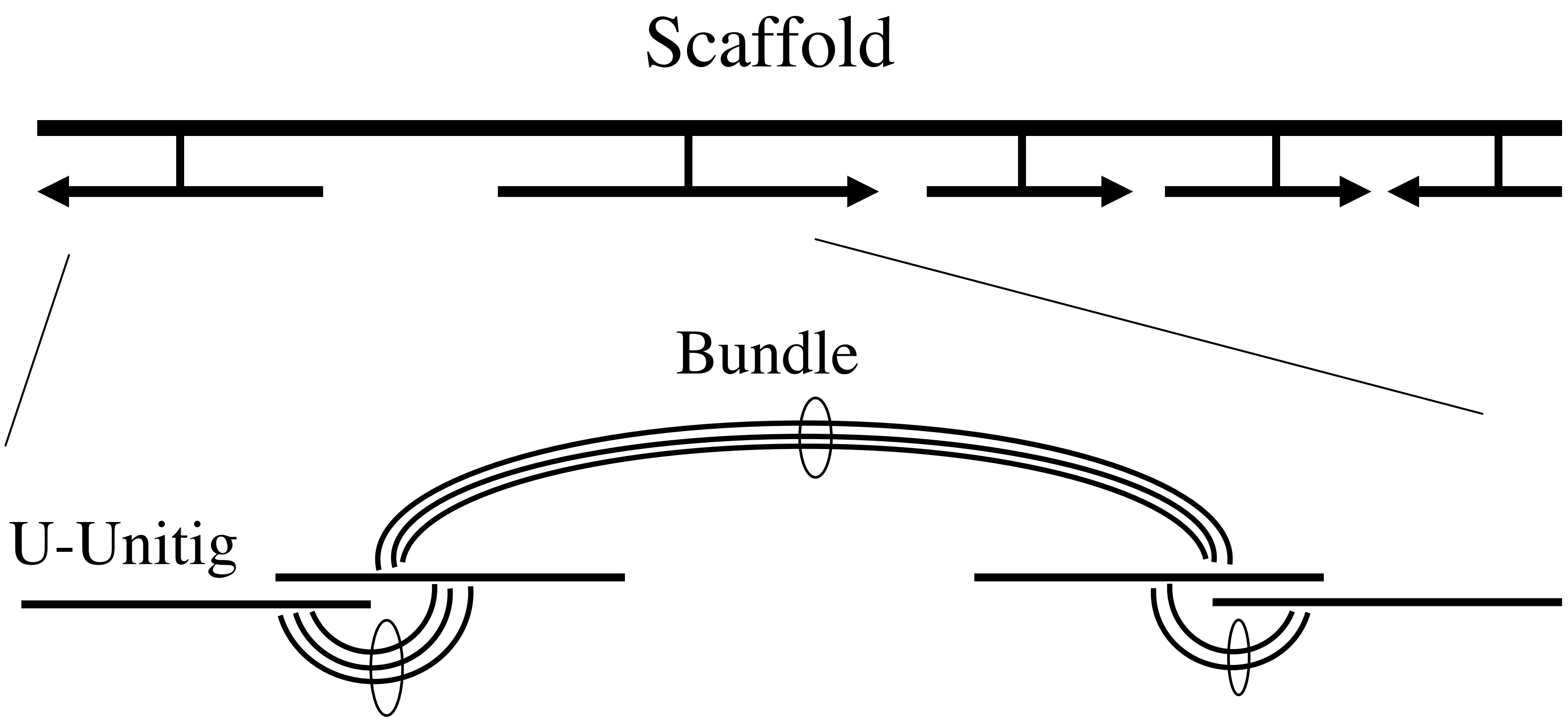
Unitig Splitting



Unitigs are split when the coverage level drops below a threshold, and there are no mates connecting the unitig.

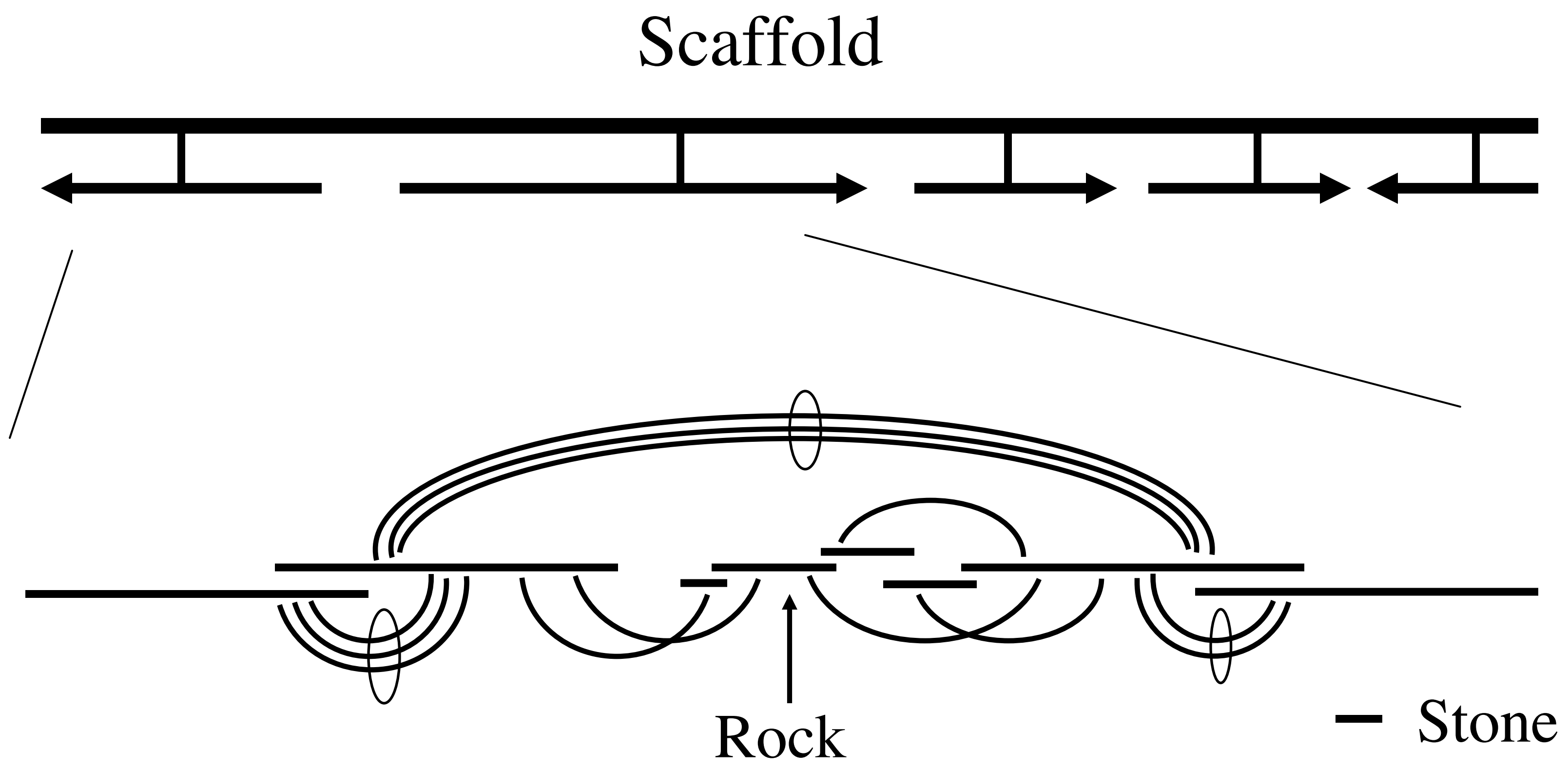
After this step, unitigs are opaque, and every read will be placed in exactly one unitig.

Initial Scaffolding



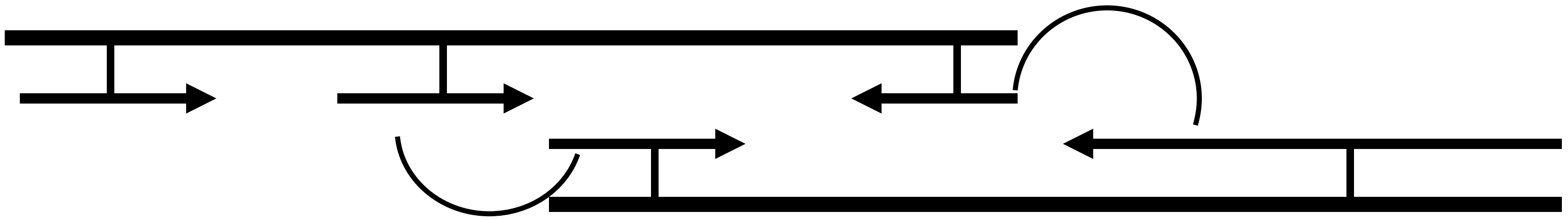
Create a initial scaffold of unique unitigs (U-Unitigs) whose A-stat > 5. Also recruit borderline unitigs whose A-stat is > 2 and have consistent mates with the U-Unitigs.

Repeat Resolution



Place rocks (A-stat > 0 with multiple consistent mates), and stones (single mate and overlap path with placed objects) into the gaps. Pebbles, unitigs lackings mates, are no longer incorporated regardless of overlap qualities.

Scaffold merging



After placing borderline unitigs and rocks, there may be sufficient mates to merge scaffolds (mates from stones are not considered). If multiple orientations are possible, choose the scaffold merge with the happiest mates.

This in turn may allow for new rocks and stones to be placed, so iterate these steps until the scaffold stabilizes.

Mate Bundling

- The CA scaffolder requires accurate library size estimates.
- Generally necessary to run scaffolder at least twice.
- CGW outputs revised library sizes, repeat until convergence.
- May need to manually split libraries if distributions are multi-modal.

Assembly Dregs

- Degenerate unitigs are unitigs with poor A-stat values and not in any scaffold as a rock or stone. (Single contig/unitig scaffolds with a good A-stat are acceptable).
- Non-unique surrogate unitigs are unitigs incorporated as stones in multiple places in the scaffold. Consequently, their reads will be multiply placed.
- Scaffolding Merging is not done with stones or degenerates so scaffolds may end even though there are unambiguous mates links to follow.

runCA Pipeline

1. Create Stores
   - gatekeeper
   - PopulateFrgStore
2. Find Repeats
   - meryl
3. Overlap
   - overlap
   - grow-overlap-store
4. Error Correction
   - correct-frags
   - correct-olaps
   - update-erates
5. Unitigging
   - unitigger
   - consensus -U
6. Scaffolding
   - cgw
   - consensus
7. Finalize Data
   - Terminator
   - qc file

Assembler outputs

asmbl.asm - all the information in Celera message format
asmbl.qc - summary statistics

asmbl.fasta, asmbl.contig - all the contigs, surrogates and degenerates
asmbl.placed.fasta, .contig - all the contigs
asmbl.surrogates.fasta, .contig - all the surrogates
asmbl.degenerates.fasta, .contig - all the degenerates

asmbl.singletons - all the singletons

asmbl.scaffolds.fasta - all the scaffolds, 60 Ns replace the gaps
asmbl.scaffolds.info - contig order/orientation for scaffolds

The .qc file

```
[Scaffolds]
TotalScaffolds=2
MeanContigsPerScaffold=23.50
MaxContigsPerScaffold=30

TotalBasesInScaffolds=3298141
MeanBasesInScaffolds=1649070.50
MaxBasesInScaffolds=2100614
N50ScaffoldBases=2100614

TotalSpanOfScaffolds=3310522
MeanSpanOfScaffolds=1655261.00
MaxScaffoldSpan=2104833
IntraScaffoldGaps=45
MeanSequenceGapSize=275.13

[Top_5_Scaffolds_contigs_size_span_avgContig_avgGap]
0=30 2100614 2104833 70020.47 145.48
```

http://www.cbcb.umd.edu/research/castats.shtml

N50 size

50% of genome is in contigs larger than N50

Example:

1 Mbp genome
Contigs: 300, 100, 50, 45, 30, 20, 15, 15, 10, ...
N50 size = 30 kbp
(300+100+50+45+30 = 525 >= 500kbp)

Note:

N50 is meaningful for comparison only when genome size is the same

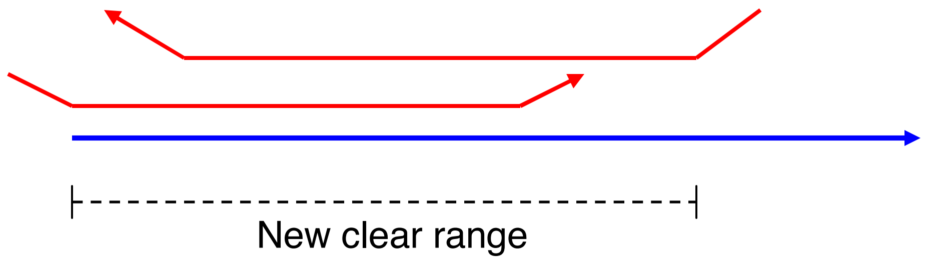
Assembly Quality

- AMOS Validation Tools
  - Library Construction
  - Contaminate Sequences
  - Read Trimming
  - Coverage Levels
  - A-stat problems / Degenerate Contigs
  - Local Mis-assembly

- Be aware of potential size/quality tradeoffs.

runCA-OBT Overlap-Based-Trimming

- Find local alignments (“partial overlaps”) between untrimmed reads.
- Use overlapping alignment regions to set new clear range.
- Patterns of overlap forks can automatically find and trim unknown vector sequences.
- runCA-OBT is a work in progress at Venter Institute
  - Does several advanced operations as well: extendClearRanges, resolveSurrogates, resizes Libraries
  - wgs-assembler/src/AS_RUN/runCA-OBT/doc.tex



Celera Assembler Summary

- Strategy
  1. Compute Overlaps between reads
  2. Simplify Overlap Graph into Unitigs
  3. Score Unitigs based on Coverage
  4. Create Contigs & Scaffold of Unique Unitigs
  5. Fill in gaps with repetitive unitigs

- Complications
  1. Vector & Quality trimming to find all overlaps
  2. Unitig Error Rate to separate repeat copies
  3. Unitig Scoring (A-stat) to build contigs from unique pieces

Current Development

• UMd / CBCB – Overlapping, Repeat Resolution	Steven Salzberg Art Delcher
• UMd / IPST – Error Correction, Unitigging	Jim Yorke
• Venter Institute – OBT, Scaffolding, Consensus	Granger Sutton
• TIGR – Code Engineering, Bug Fixes	Martin Shumway Jason Miller