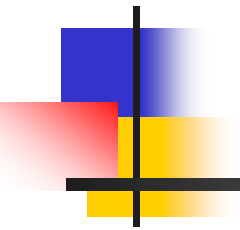
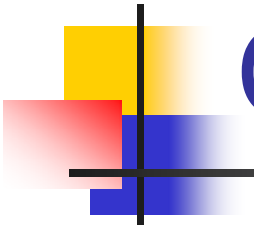


Improving Genome Assemblies without Sequencing

Michael Schatz

Sept 28, 2005
CBCB Seminar



Outline

- Theoretical Foundations of Assembly
- Modern Assembly in Practice
  - Case Study: Celera Assembler
- Assembly Improvement
  - AutoEditor
  - AutoJoiner
- Future Directions
- Acknowlegements

Shortest Common Superstring

Given: $S = \{s_1, \ldots, s_n\}$

Problem: Find minimal superstring of S

s_1 CACCC

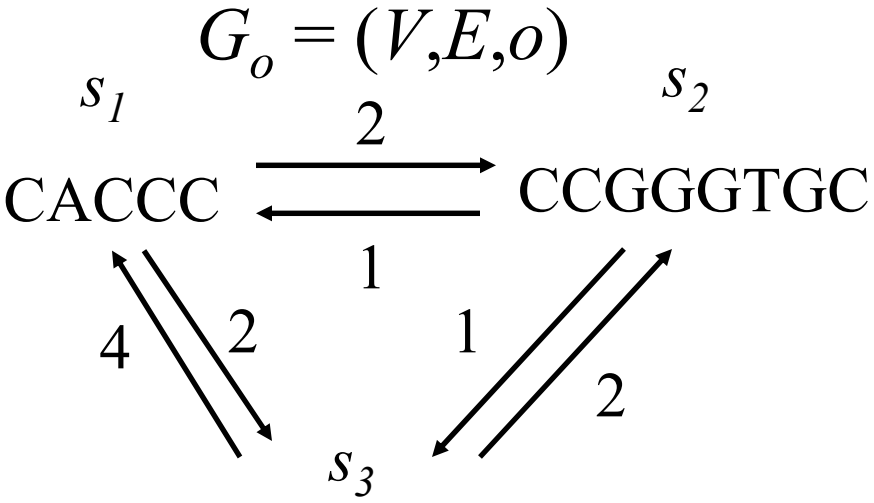
s_2 CCGGGTGC

s_3 CCACC

$s_1, \mathbf{s_2}, s_3$ = CAC**CCGGGTGC**CACC	15
$s_1, \mathbf{s_3}, s_2$ = CAC**CCACC**GGGTGC	14
$s_2, \mathbf{s_1}, s_3$ = CCGGGTG**CACCC**ACC	15
$s_2, \mathbf{s_3}, s_1$ = CCGGGTG**CCACC**C	13
$s_3, \mathbf{s_1}, s_2$ = C**CACCC**GGGTGC	12
$s_3, \mathbf{s_2}, s_1$ = CCA**CCGGGTGC**ACCC	15

NP-Complete by reduction from VERTEX-COVER and later DIRECTED-HAMILTONIAN-PATH

Overlap Graph



$$V = \{s_1, s_2, s_3\} \qquad E = \{s_i, s_j\}$$

$$o(s_i,s_j) = |v| \mid s_i = uv,\ s_j = vw$$

The overlap graph, G_o, encodes the amount of overlap between all pair of strings.

Greedy Approximation

$$G_o = (V,E,o)$$

$$\text{GREEDY}(S) \leq 2.5\ \text{OPT}(S)$$

$$\text{Runtime } O(\binom{n}{2} l^2)$$

SUPERSTRING is MAX SNP-hard, so one of the best approximation algorithms possible.

RECONSTRUCT

Given: $F = \{f_1, \ldots, f_n\}$, error rate ε

Problem: Find minimal sequence S over F such that for all f_i in F, there is a substring B of S such that:

$$\min(\text{ed}(f_i, B), \text{ed}(f_i^c, \text{B})) \leq \varepsilon\, |f_i|$$

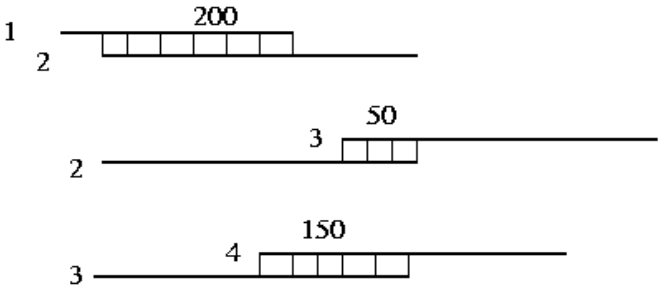
f_1^c GGGTG	ed(ACGTA, ACGGTA) =1
f_2^c GCACCCGG	ed(ACGGGTA, ACGGTA) =1
f_3^c GGTGG	ed(ACGCTA, ACGGTA) = 1

Also NP-complete: Take instance of SUPERSTRING, expand strings to force the original orientation, set $\varepsilon = 0$, and attempt to solve with RECONSTRUCT.

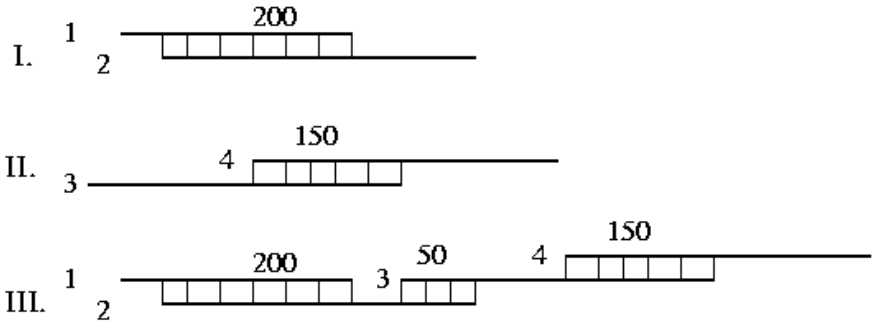
Early Assemblers

Greedy Algorithm

1. Build a rough map of fragment overlaps
2. Pick the largest scoring overlap
3. Merge the two fragments
4. Repeat until no more merges can be done

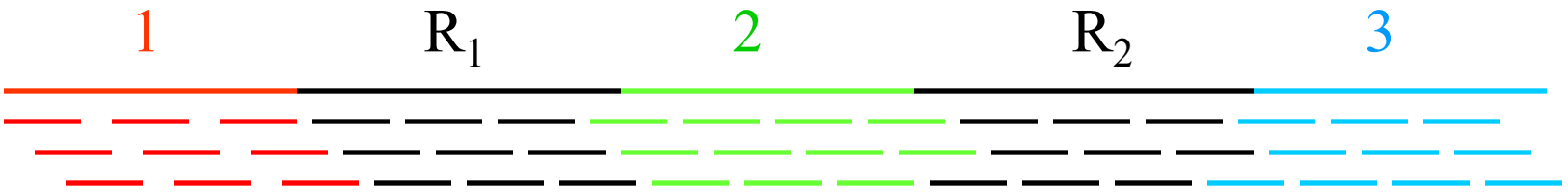


- TIGR Assembler
- phrap
- gap

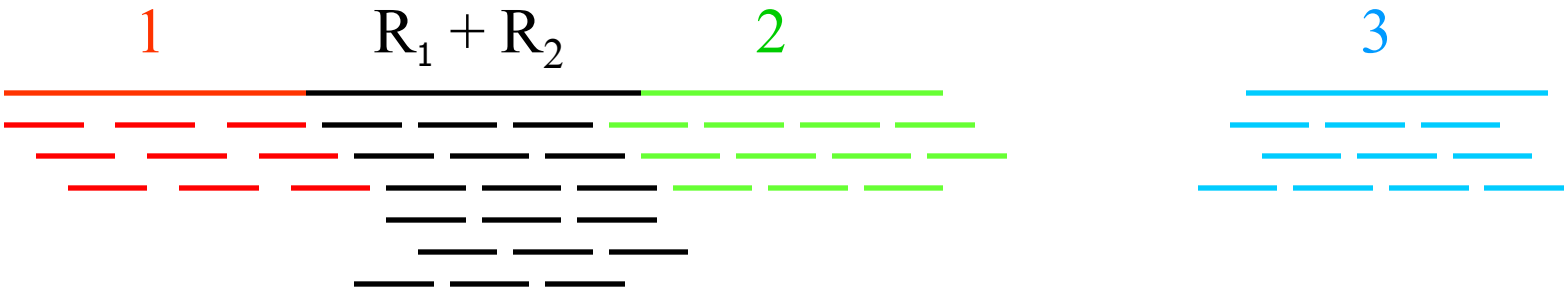


Repeats!

True Layout of Reads



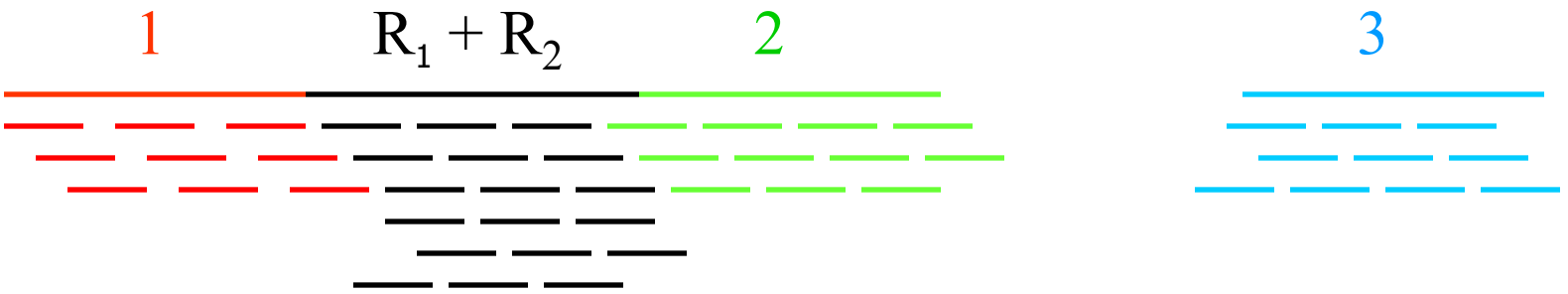
Greedy Reconstruction



Modern Assembly

Try to detect presence of repeats by

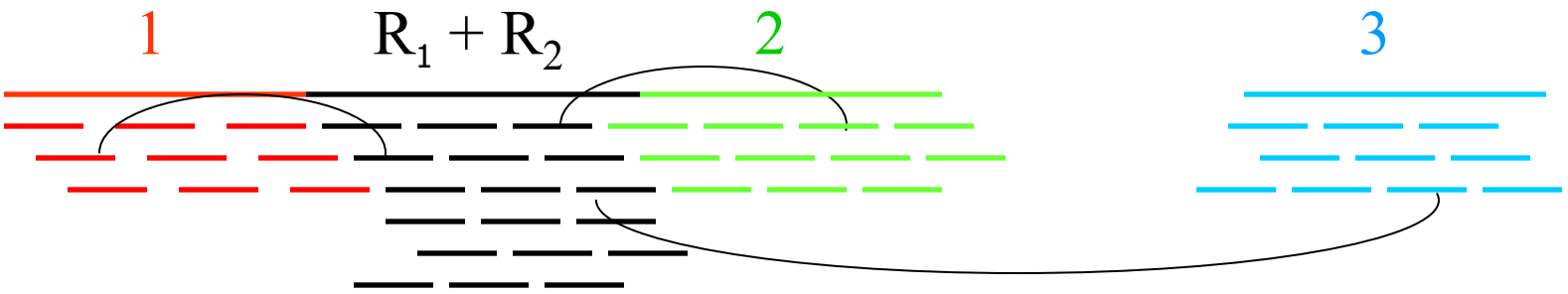
1. Unusual depth of coverage (arrival rate)
2. Mate Pair information
3. Forks in overlap graph



Modern Assembly

Try to detect presence of repeats by

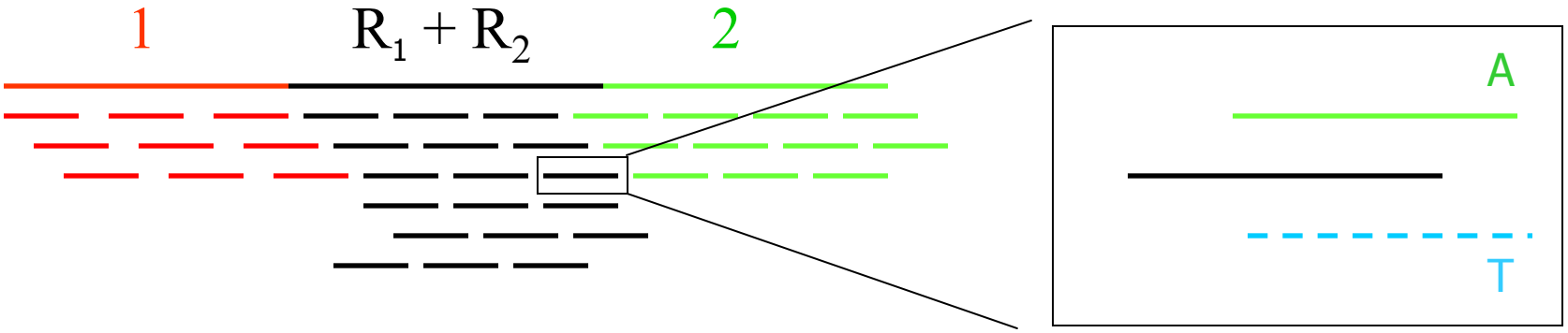
1. Unusual depth of coverage (arrival rate)
2. Mate Pair information
3. Forks in overlap graph



Modern Assembly

Try to detect presence of repeats by

1. Unusual depth of coverage (arrival rate)
2. Mate Pair information
3. Forks in overlap graph



Celera Assembler Overview

- Primarily developed in 25 man years by 13 computer scientists at Celera for the private human genome effort.

- Currently available as an open source project:
  http://wgs-assembler.sourceforge.net

Preprocessing

1. Sequence Reads
2. Base Calling and Trimming

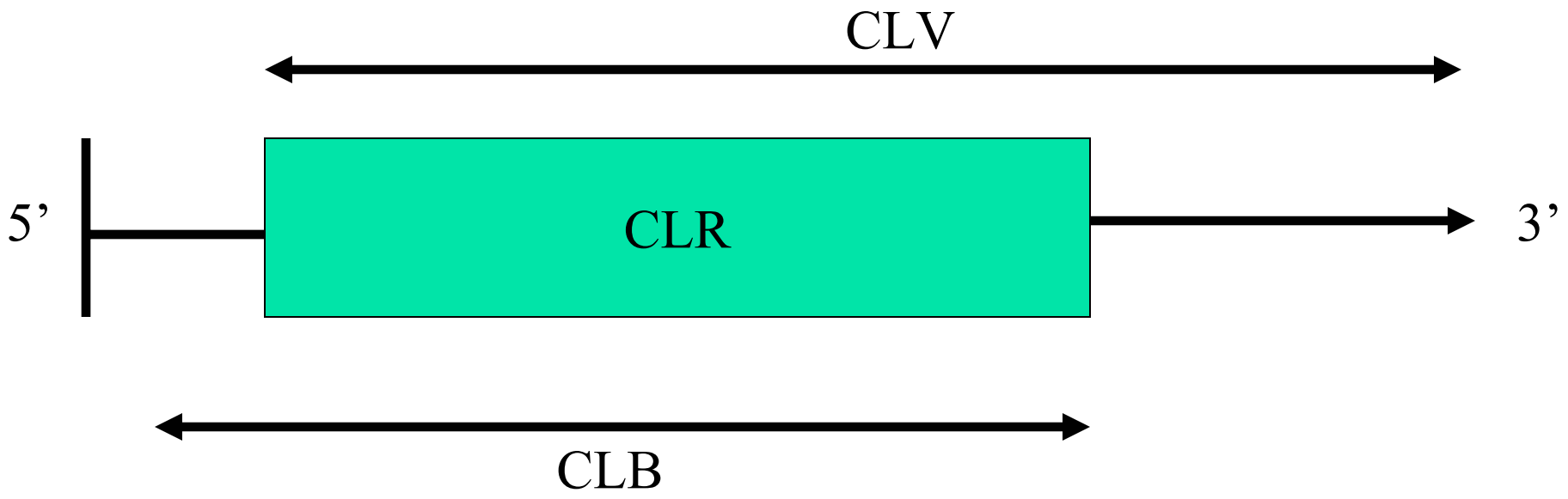
Pipeline Summary

1. Overlap Reads
2. Unitig Creation
3. Scaffolding & Repeat Resolution
4. Final Consensus

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.

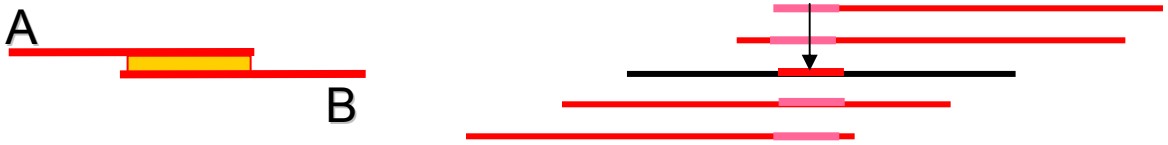
Trimming



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).

Overlapper & Error Correction

- Find all overlaps ≥ 40bp allowing 6% mismatch using k-mer (k=22) seed matches with O(nd) extension
- Avoid seeding overlaps with k-mers whose occurrence ≥ 100 in the trimmed read set.
- 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.



ACGTACCGATATGACAC

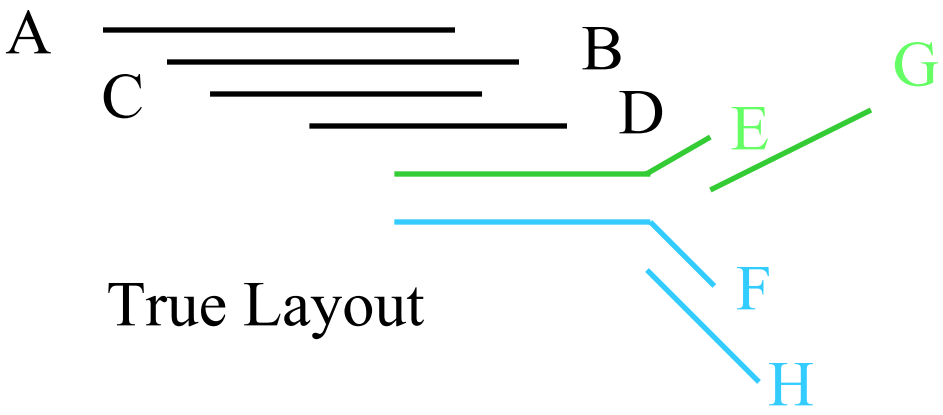
ACGTACCGATATGACAC

ACGTACCGTTATTACAC

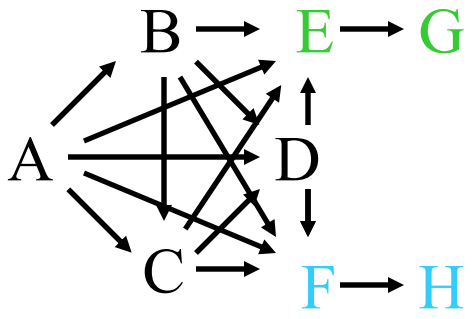
ACGTACCGATATTACAC

ACGTACCGATATGACAC

Unitigging



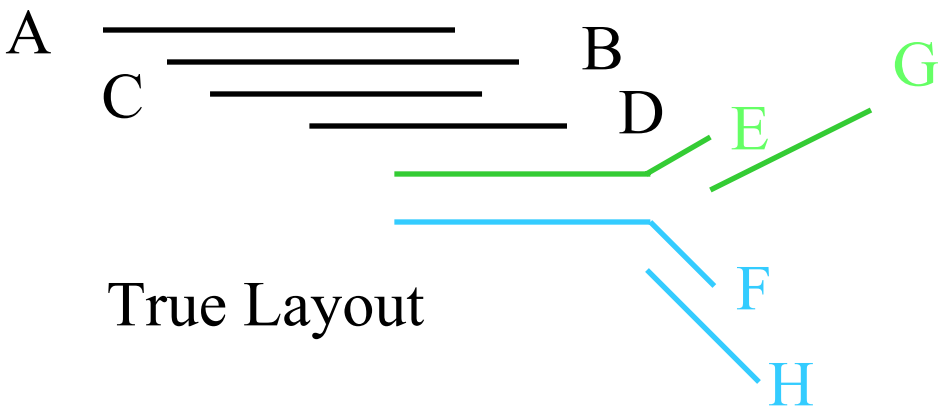
True Layout



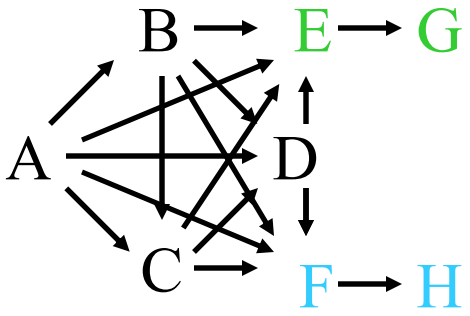
Original Overlap Graph

$|V| = 8, |E| = 16$

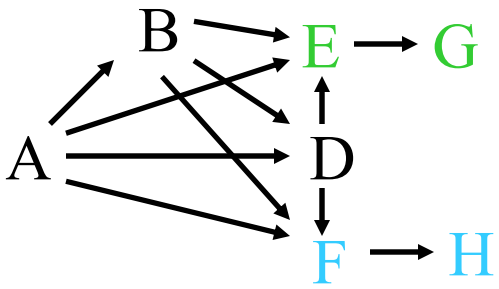
Unitigging



True Layout

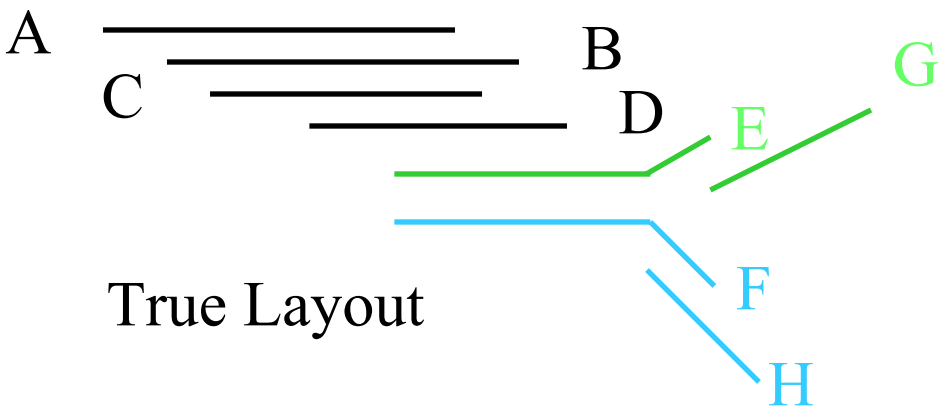


Original Overlap Graph
|V| = 8, |E| = 16

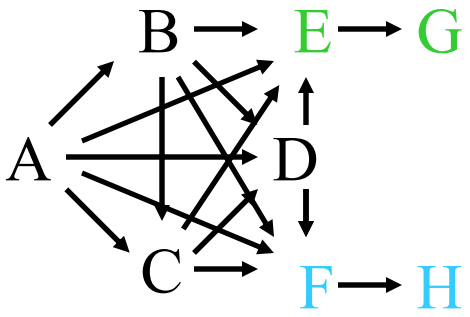


Contained
Read
Removal
|V|=7, |E|=11

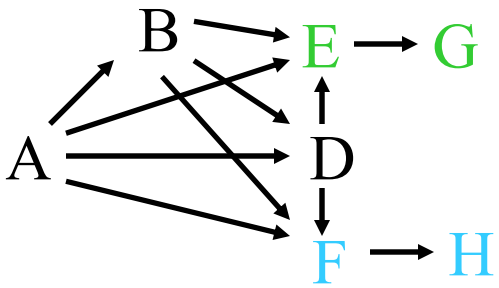
Unitigging



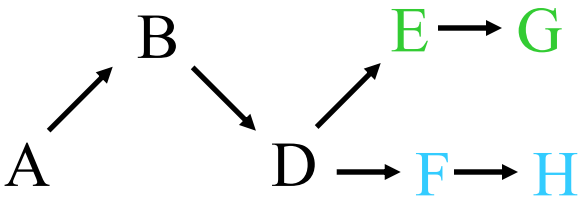
True Layout



Original Overlap Graph
|V| = 8, |E| = 16

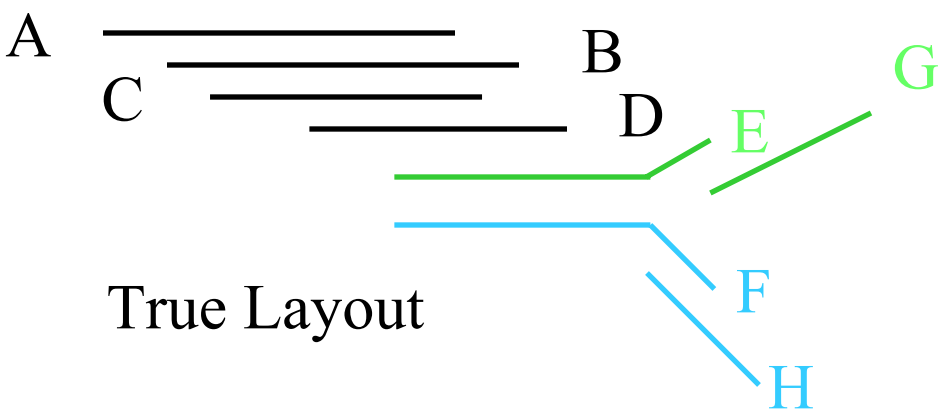


Contained
Read
Removal
|V|=7, |E|=11

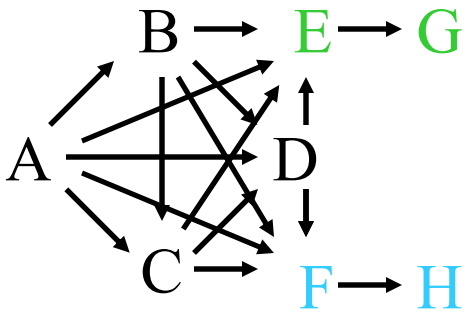


Transitive
Edge
Removal
|V|=7, |E|=6

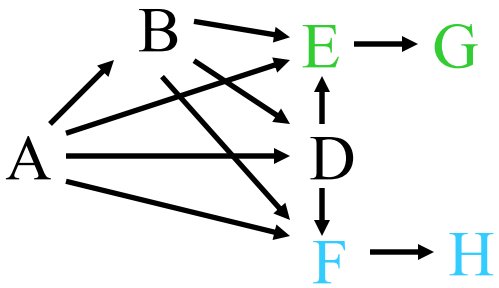
Unitigging



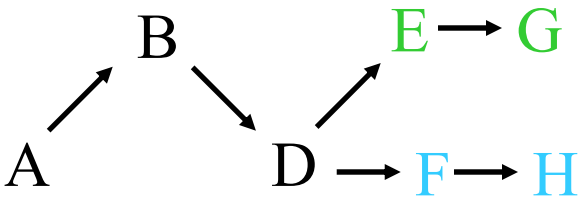
True Layout



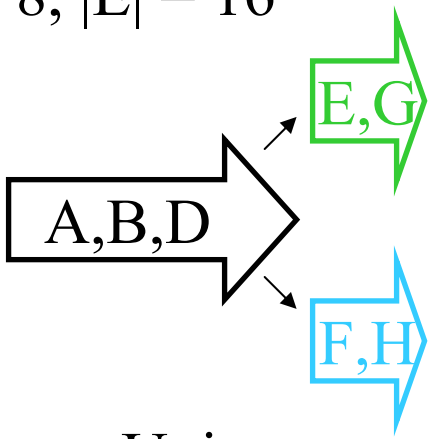
Original Overlap Graph
|V| = 8, |E| = 16



Contained
Read
Removal
|V|=7, |E|=11

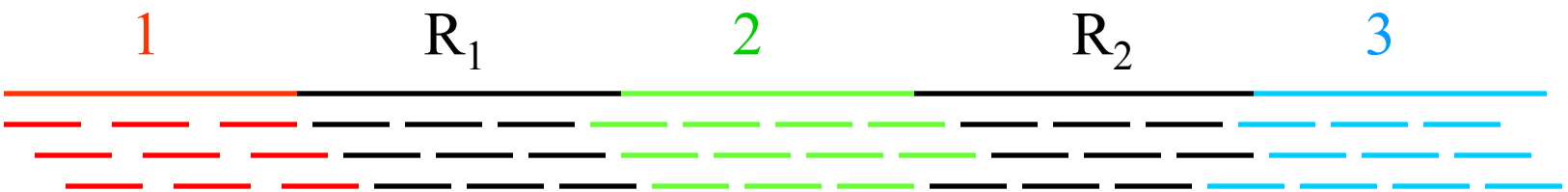


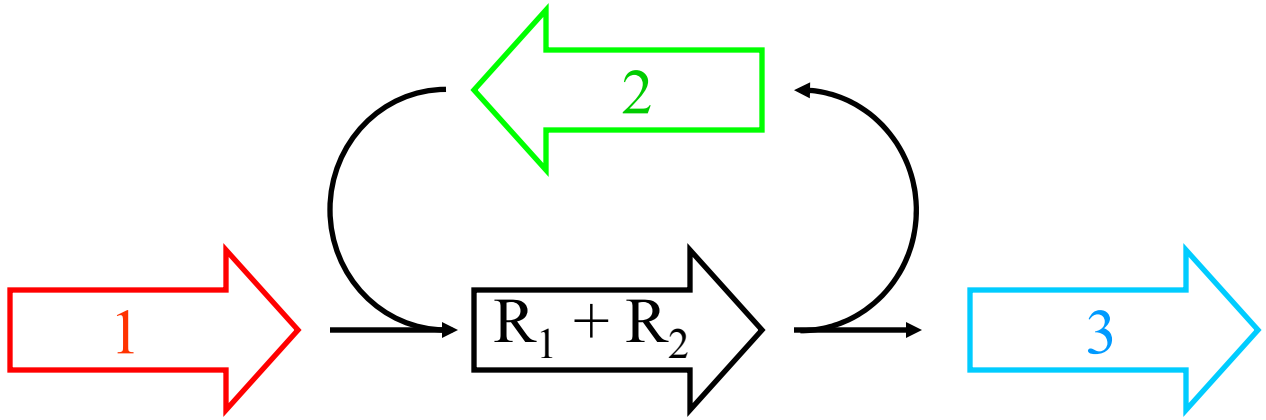
Transitive
Edge
Removal
|V|=7, |E|=6

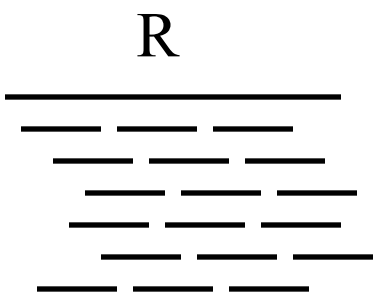


Unique
Join
Collapsing
|V|=3, |E|=2

Unitig Graph

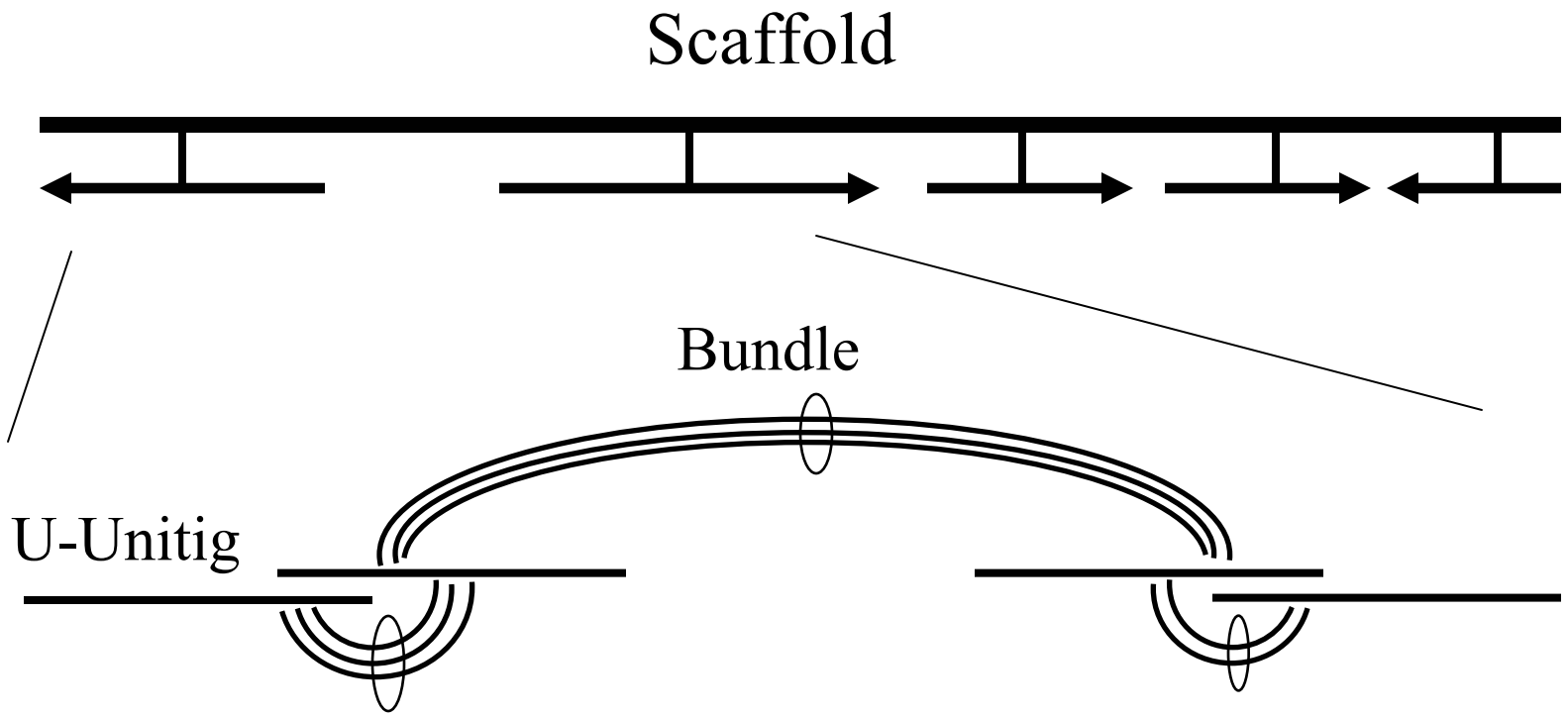






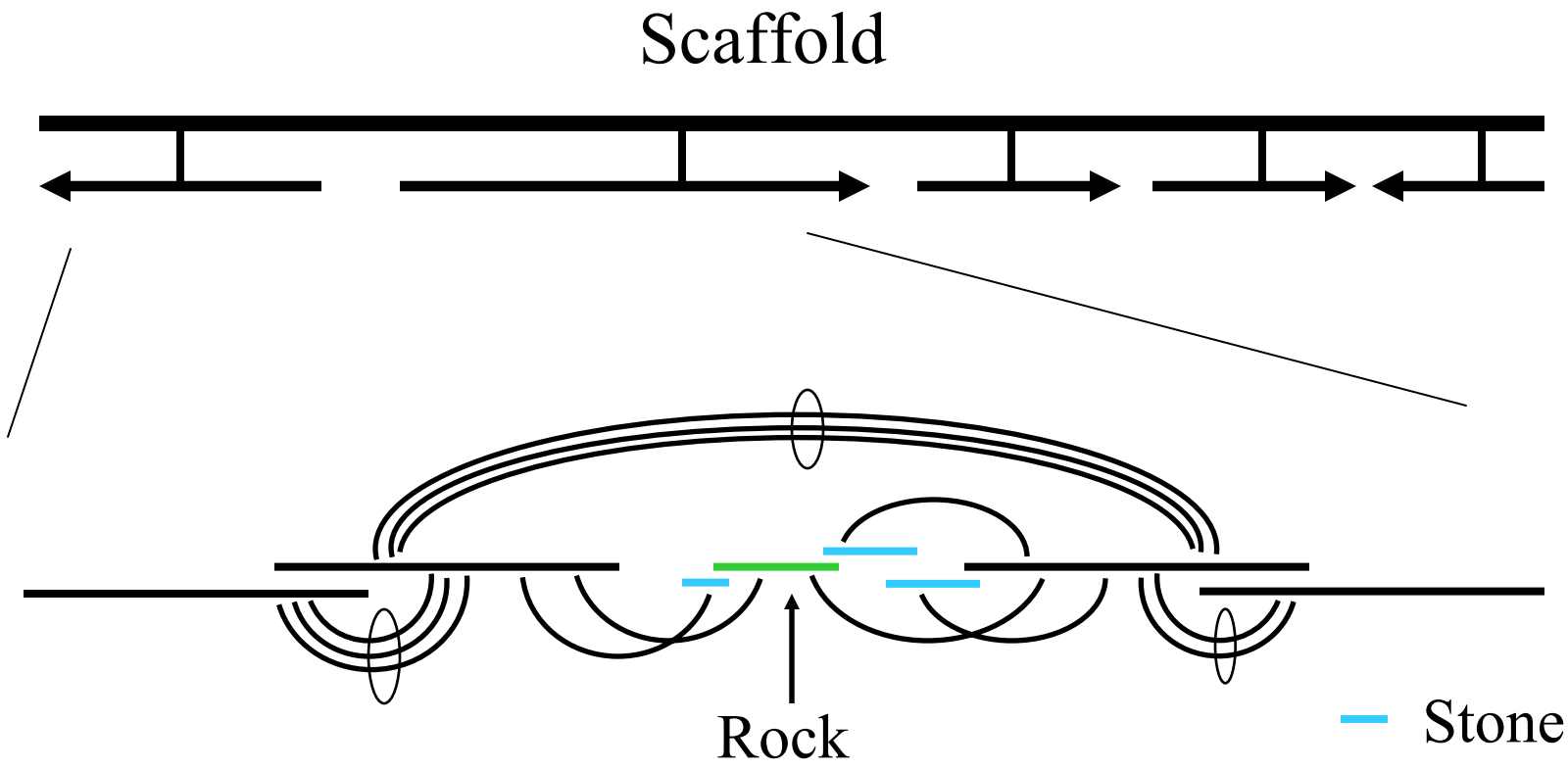
The arrival rate of reads within unitig R will be much higher than for unique unitigs A, B or C. Its A-stat will be lower and flag the unitig as unreliable.

Initial Scaffolding



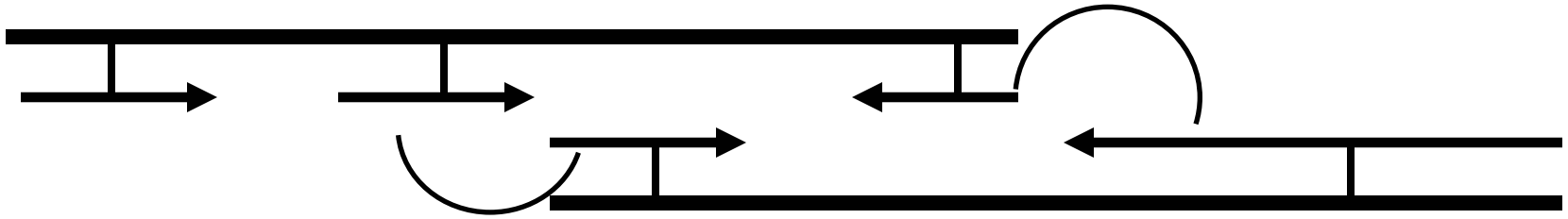
Create a initial scaffold of unique unitigs (U-Unitigs) whose A-stat > 5. Also recruit borderline unitigs whose A-stat is > 2 and have consistent mates with the U-Unitigs.

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

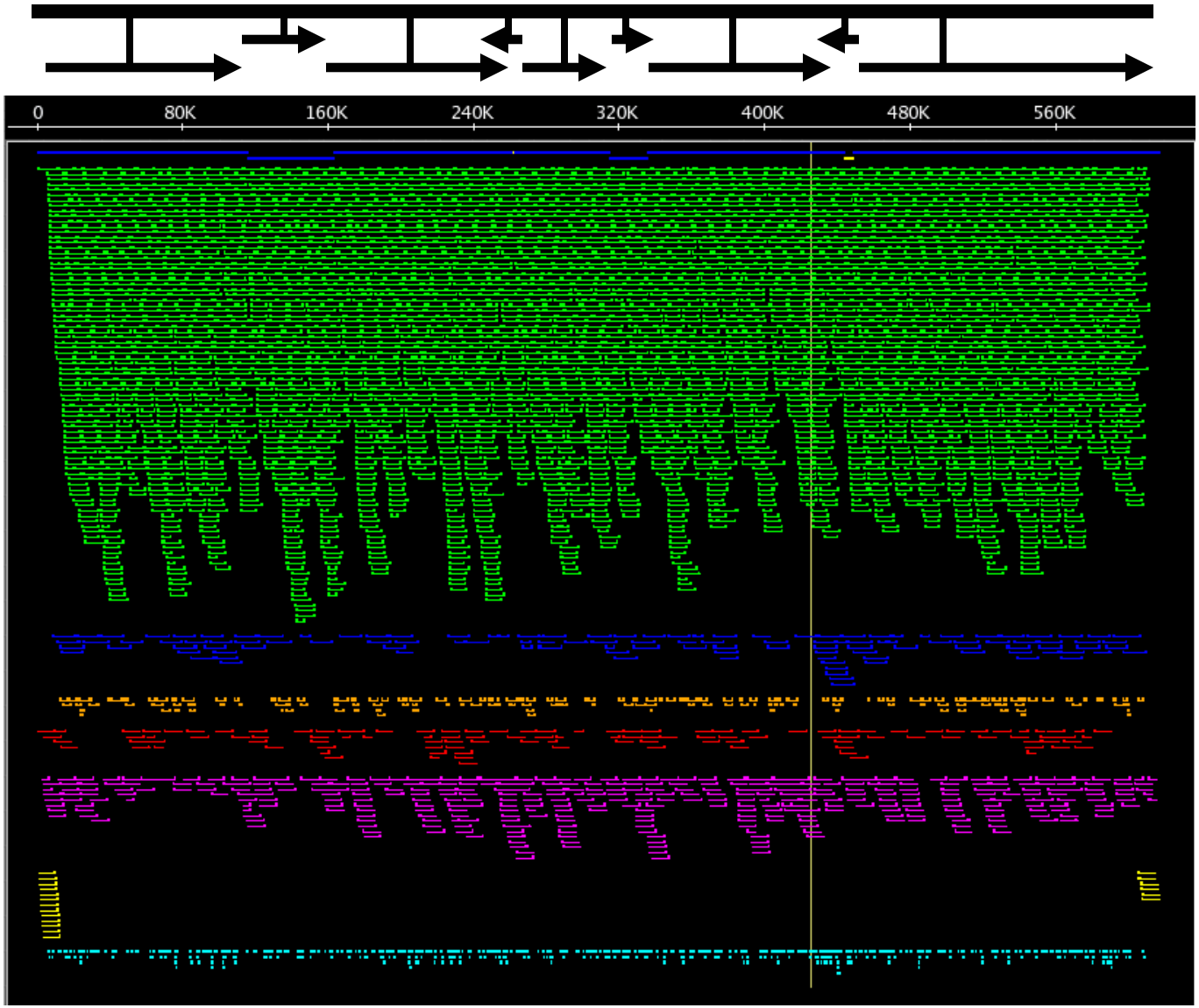


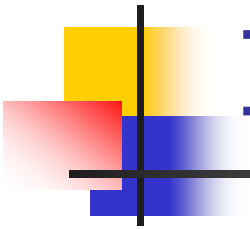
After placing borderline unitigs and rocks, there may be sufficient mates to merge scaffolds (mates from stones are not considered). If multiple orientations are possible, choose the scaffold merge with the happiest mates.

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

Assembly Results

- Chromosomes of Scaffolds (616691bp)
- Scaffolds of Contigs (9 Contigs)
- Contigs of Reads (9658 Reads)





Improving the Assembly

Key Idea:

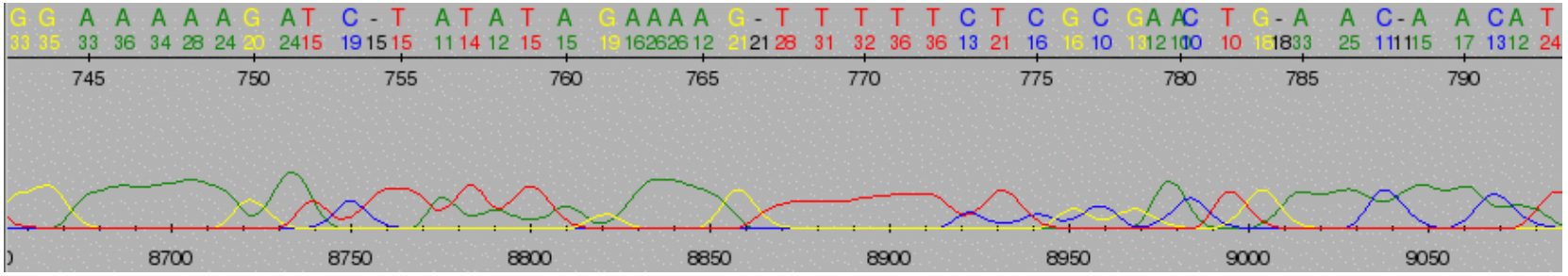
- Have to be relatively conservative at first
- But, there is a lot of additional contextual information available after the initial assembly.

Use this contextual information to revise original

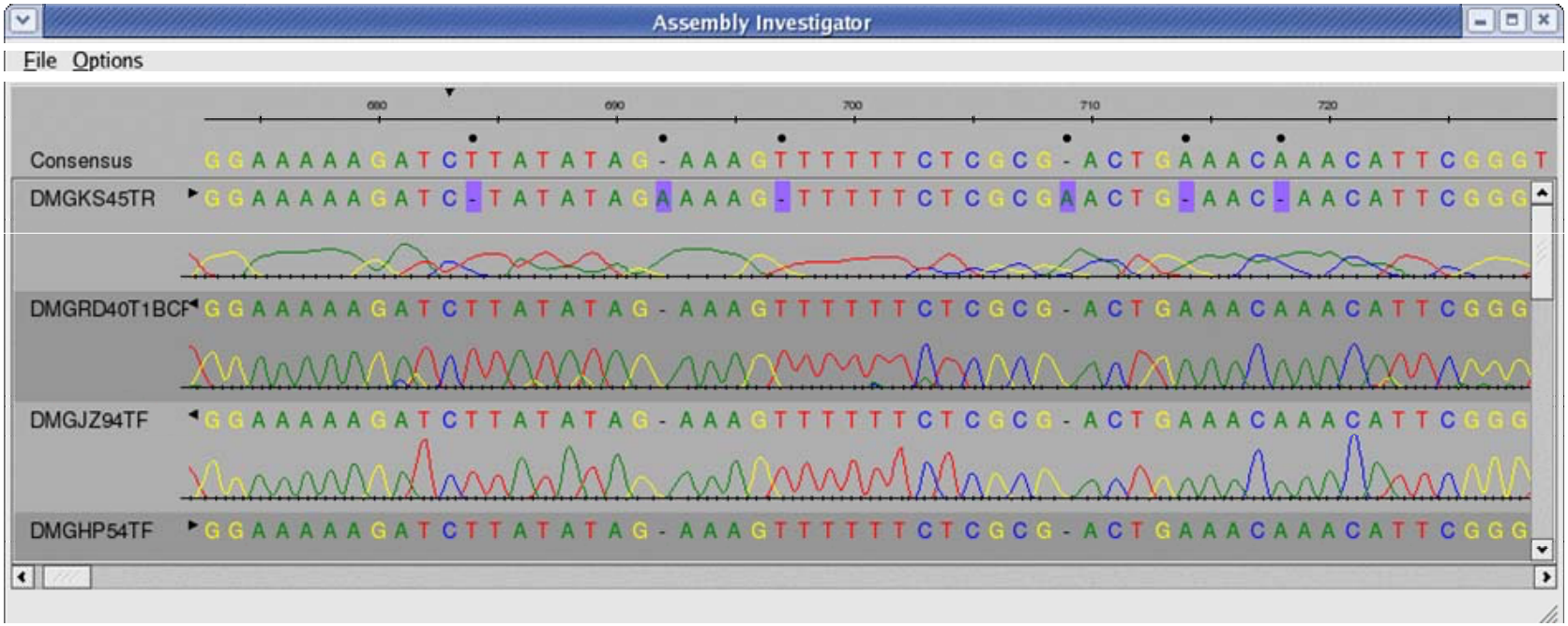
- Base-calling: AutoEditor
- Clear ranges: AutoJoiner

AutoEditor

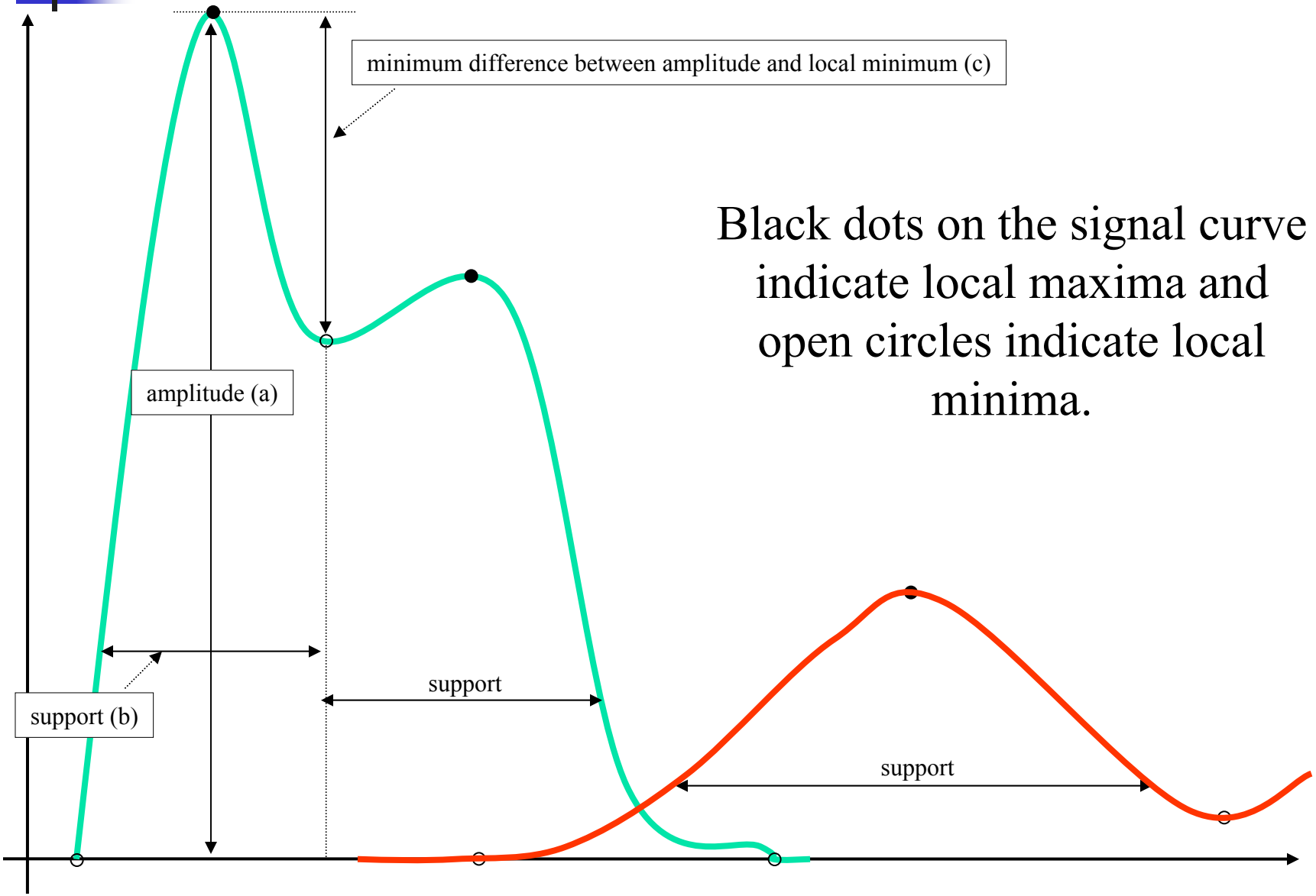
Base-calling in the context of single chromatogram is hard…



but finding base-calling "mistakes" in a multiple alignment is easy.



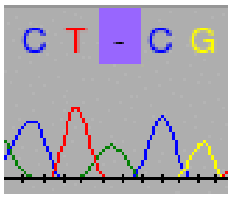
Signal Parameters

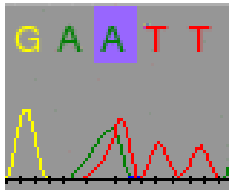


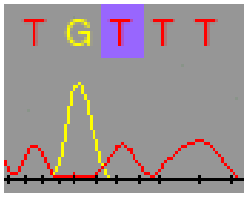
Black dots on the signal curve indicate local maxima and open circles indicate local minima.

AutoEditor Results

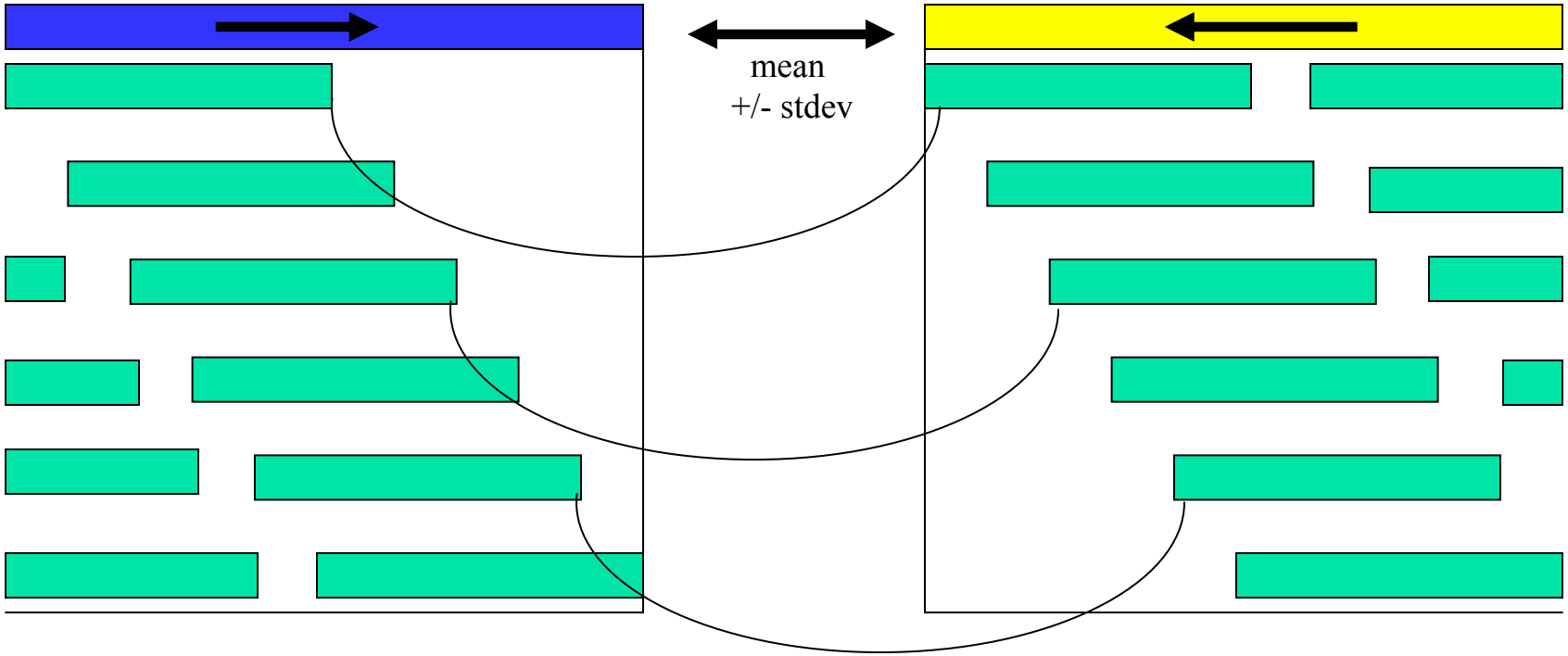
- Corrects 80% of all discrepant base-calls with an error rate better than 1/8800.
- Increase consensus quality, decrease finishing costs
- Remaining discrepancies highlight assembly problem regions or interesting biological events.







Quick Assembly Review

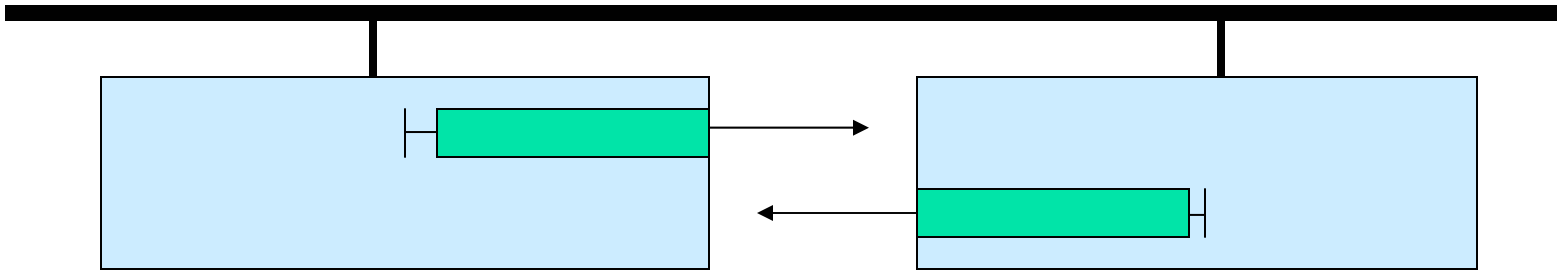


The individual reads (green) have been assembled into 2 contigs (blue & yellow). The mate relationship between the reads allows for the contigs to be oriented and the gap size to be estimated.

AutoJoiner Architecture

Automatic Gap Closure

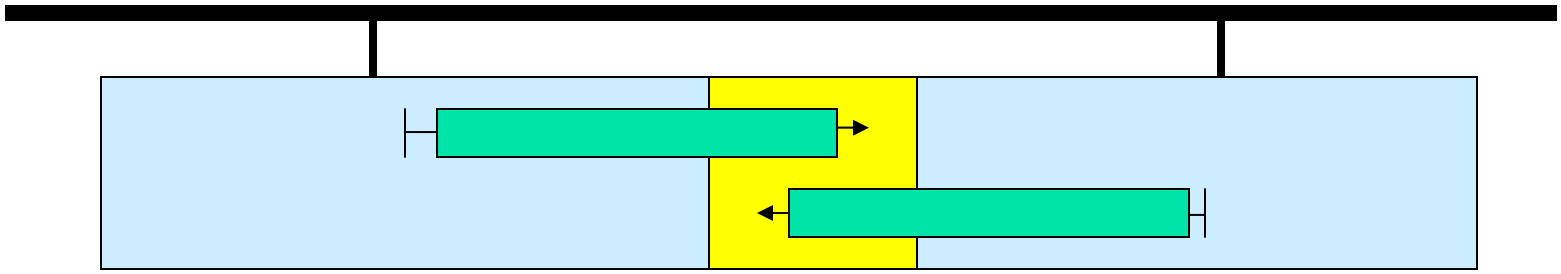
- All-vs-All Alignment
- Analyze Alignments
- Extend Contigs
- Join Contigs
- Contig Fattening



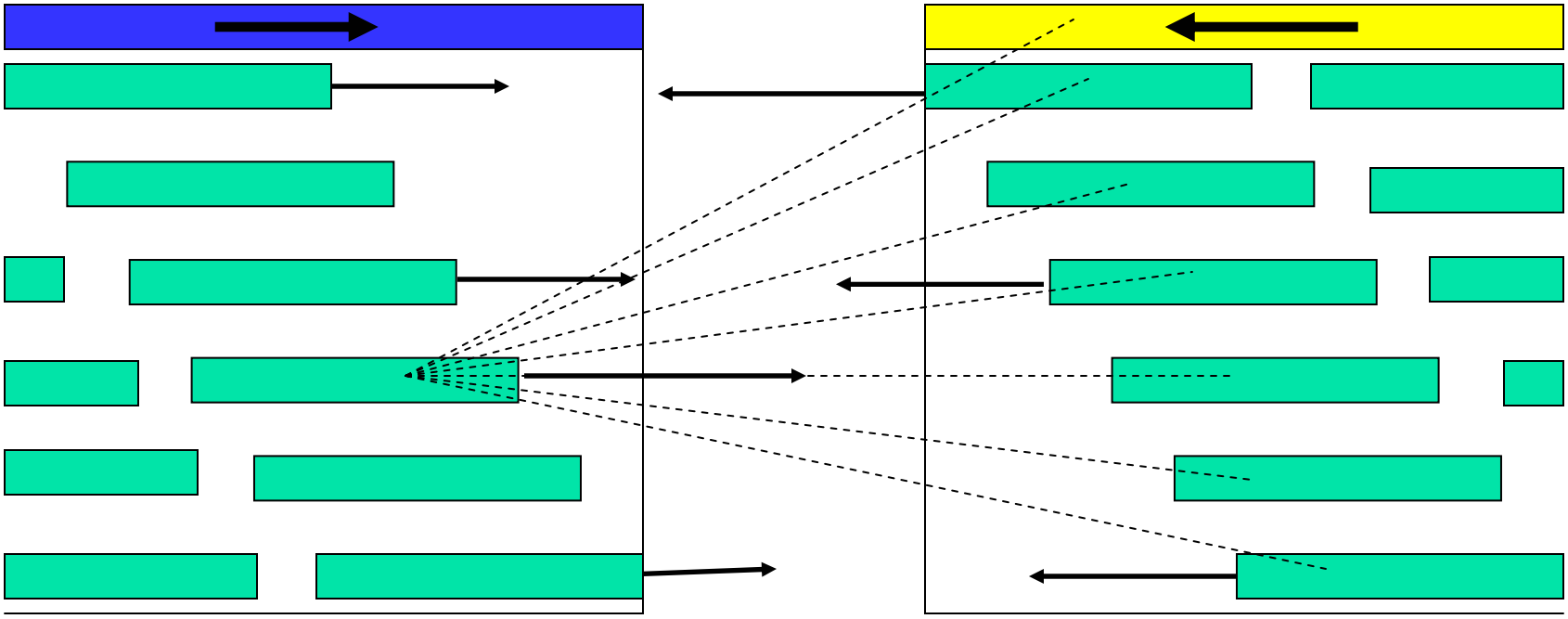
AutoJoiner Architecture

Automatic Gap Closure

- All-vs-All Alignment
- Analyze Alignments
- Extend Contigs
- Join Contigs
- Contig Fattening

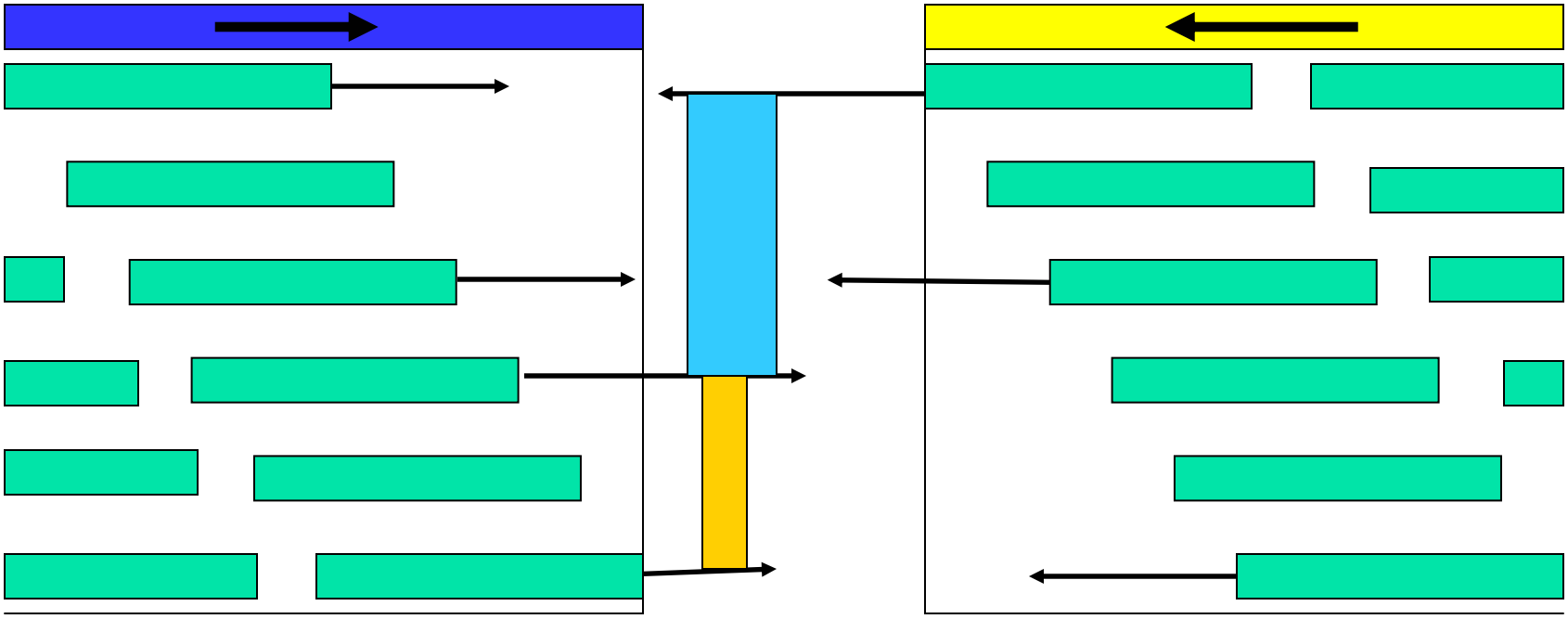


All-vs-all Alignment



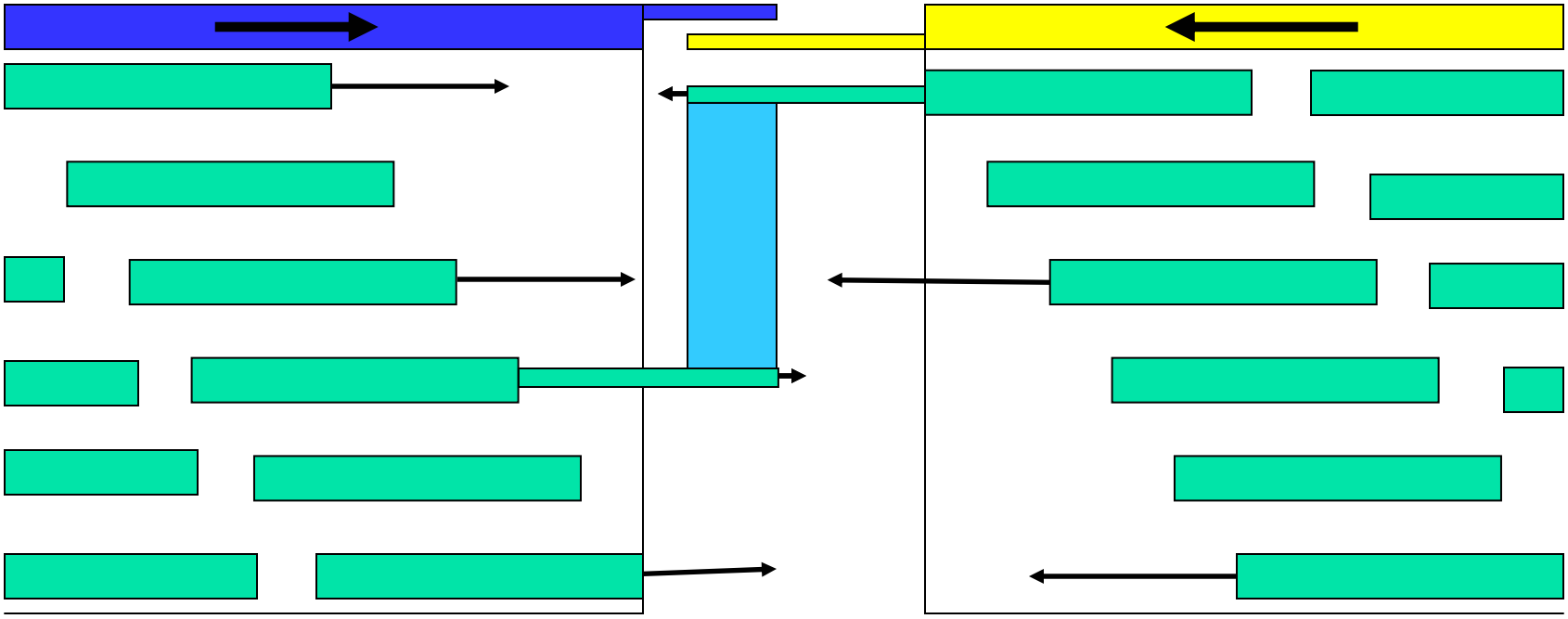
1. An all-vs-all pairwise alignment between the full range sequences from the flanking contigs is computed.

Alignment Analysis



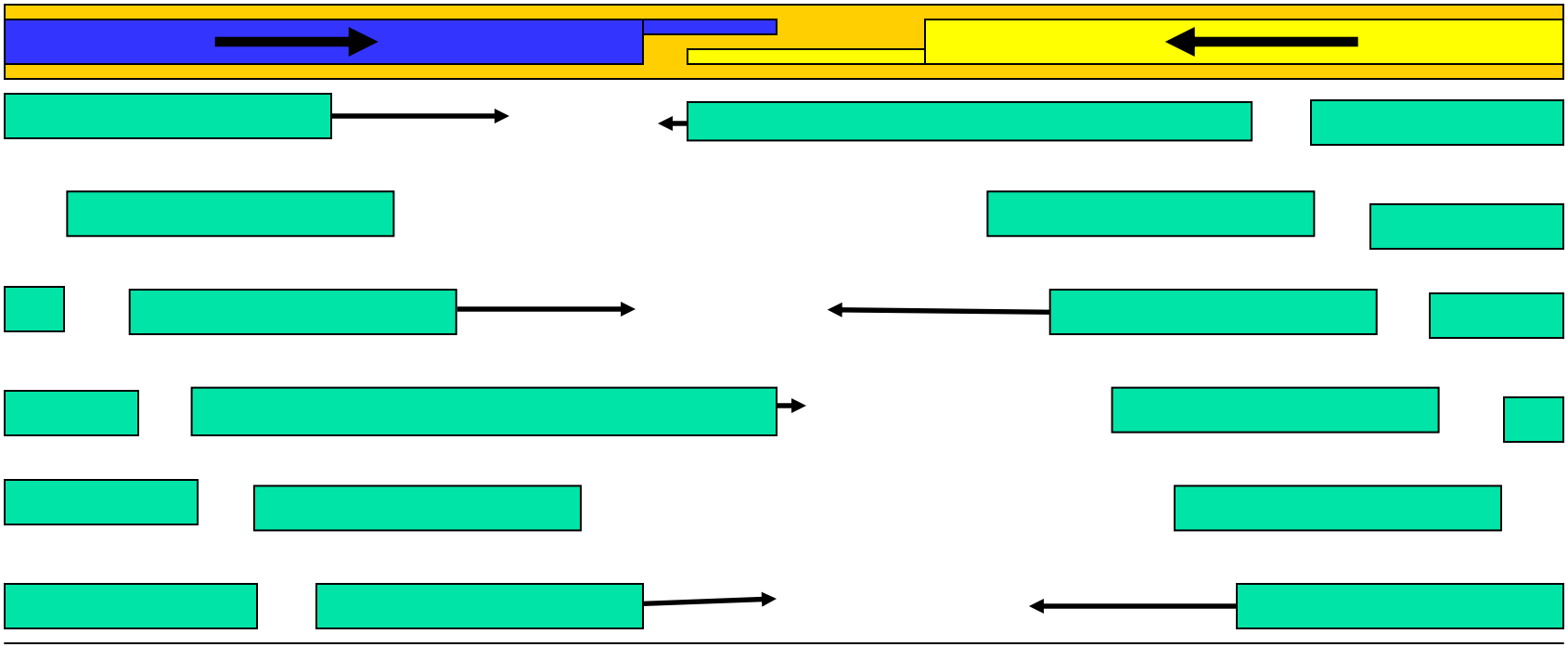
2. The alignments are tested for consistency with the scaffold and for being of sufficient quality. If any alignments satisfy the requirements, the best alignment (blue) is selected for joining the contigs.

Contig Extension



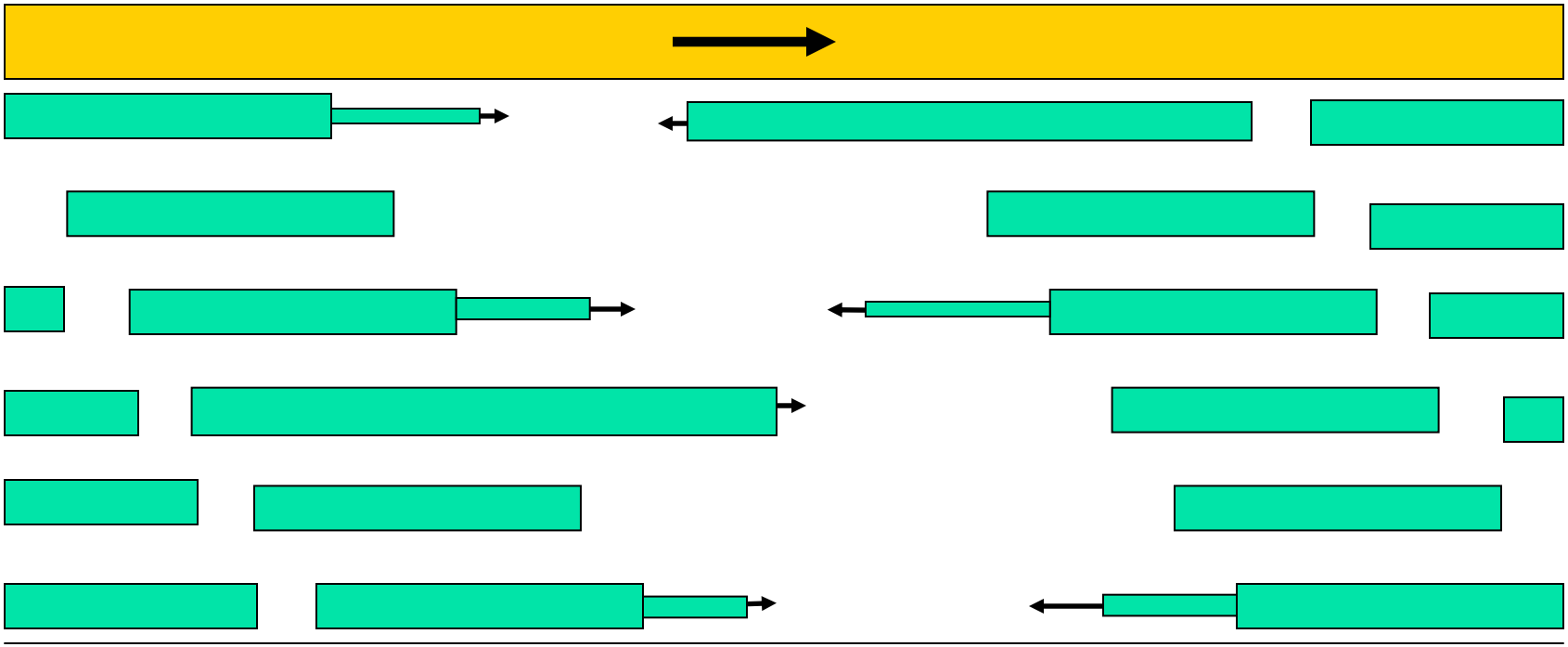
3. The contigs are extended by extending the selected reads beyond their original clear range to the desired position. If necessary, the reads are first aligned to the existing consensus.

Contig Joining



4. The contigs are joined by aligning the newly extended consensus sequences. The joined contig (orange) replaces the original two in the scaffold.

Contig Fattening



5. The join region is fattened to increase the depth of coverage and enhance the consensus quality.

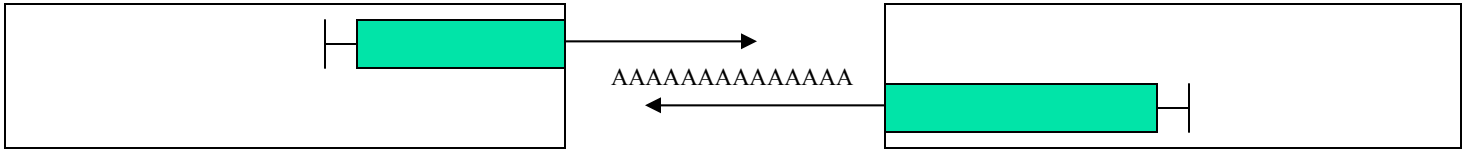
AutoJoiner Validation

- Tested against assemblies of 30 finished genomes and chromosomes.
- Over 25% of gaps closed
- Only 3 invalid joins.

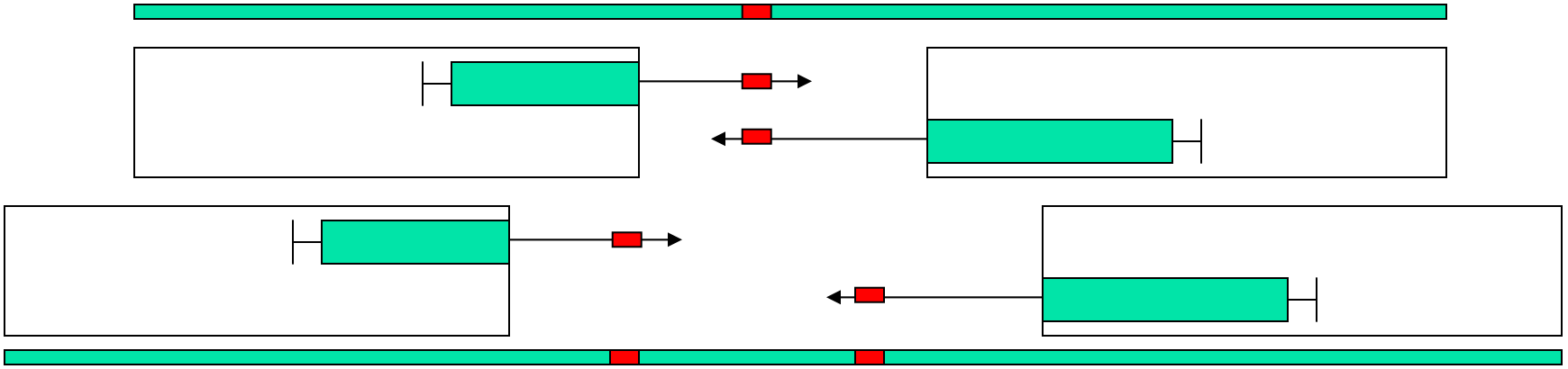
Organism	Genome Size (Mbp)	Gaps	Joined	%	False Joins	Gap Size	Mean
Bacillus anthracis Ames	5.22	110	38	34.5%	0	-452:229	29.18
Bacillus anthracis Ames Ancestor	5.22	32	10	31.2%	0	-13:146	42.3
Brucella suis	3.31	32	11	34.4%	0	-62.5:103.5	15.36
Burkholderia mallei	5.83	43	6	14.0%	0	-17:22	-4.67
Campylobacter jejuni	1.78	22	11	50.0%	0	-53:139	27.45
Chlamydophila caviae	1.17	25	8	32.0%	0	-555:184.5	-75.31
Coxiella burnetii	1.99	10	2	20.0%	0	-37.5:-4.5	-21
Cryptococcus neoformans 1	2.3	20	8	40.0%	0	-36:186.5	63.62
*Cryptococcus neoformans 2**	1.63	7	5	71.4%	1	-39:148	27.8
Cryptococcus neoformans 3	2.11	21	8	38.1%	0	-93:67.5	-3.06
Cryptococcus neoformans 4	2.04	25	7	28.0%	0	-90:159	45.21
Cryptococcus neoformans 5	1.78	23	8	34.8%	0	-111.5:249	35.12
Cryptococcus neoformans 6	1.51	14	7	50.0%	0	-14:192	37.21
Cryptococcus neoformans 7	1.44	17	6	35.3%	0	-3.5:230.5	66.67
Cryptococcus neoformans 8	1.35	15	6	40.0%	0	-19:57.5	15
Cryptococcus neoformans 9	1.18	12	6	50.0%	0	-423:34	-120.5
Cryptococcus neoformans 10	1.09	14	7	50.0%	1	-777:124	-91.21
Cryptococcus neoformans 11	1.02	12	2	16.7%	0	-6:69.5	31.75
Cryptococcus neoformans 12	0.79	10	4	40.0%	0	-340:77.5	-69.88
Cryptococcus neoformans 13	0.76	13	7	53.8%	1	-213.5:144	19.07
Dehalococcoides ethenogenes	1.47	82	17	20.7%	0	-113:203.5	22.29
Fibrobacter succinogenes	3.84	131	33	25.2%	0	-182.5:212	21.79
Listeria monocytogenes	2.9	106	14	13.2%	0	-200.5:156	21.18
Mycoplasma capricolum	1.15	10	2	20.0%	0	-11.5:171	79.75
Neorickettsia sennetsu Miyayama	0.86	27	17	63.0%	0	-779:-302	-586.71
Prevotella intermedia	2.68	150	52	34.7%	0	-231.5:181	20.3
Pseudomonas syringae	6.53	162	43	26.5%	0	-1069.5:213.5	-36.13
Staphylococcus aureus	2.8	262	32	12.2%	0	-618:136	-43.44
Streptococcus agalactiae	2.16	31	5	16.1%	0	-5:32.5	10.9
Wolbachia sp.	1.27	52	13	25.0%	0	-666.5:36.5	-140.73
Composite	**69.18**	**1490**	**395**	**26.5%**	**3**	**-1069.5:249**	**-25.89**

Complicating Issues

- Poly-monomer tails
  - Use dust to filter low complexity sequence



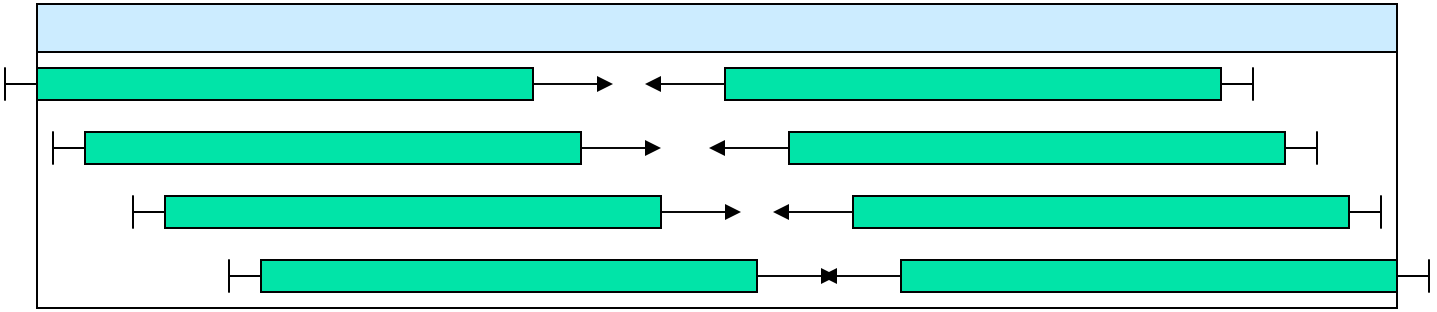
- Undetected repeats
  - Require strict agreement with scaffold



- Chimeric reads / Hard Stops
  - Good: Require high alignment similarity.
  - Better: Recognize hard stops by coverage gradients, other clues.
  - Best: Recognize unreliable sequence at chromatogram level.

Pre-Production Techniques

- Contig Fattening
  - TVG coverage increased from 5.83X to 6.10X (mean extension: 80.5bp)



- Contig Growing
  - Extended 6144 edges in TVG (mean extension: 59.0bp)

Research Directions

- AMOS Framework
- AutoEditor 2.0: Better results, better engineering
- Context Based trimming
  - Partial Overlaps
  - Reference sequence
- Context Based Unitigging
  - Unitig Splitting & Error Correction
  - Assembling in the gap
- Assembly Forensics
- Assembly Visualization / Navigation

- More Complicated Genomes
- New Sequencing Technologies

Conclusions

- Assembly is complicated by genome structure, repeat characteristics, data quality, data management- one size does not fit all.
- Overriding strategy: Start conservatively, and iteratively build as more information becomes available.
- 95.5% - 99.2% of a chromosome in a single scaffold not typical yet, but it could be.
  - Be aware of potential size/quality tradeoffs, though.
- State-of-the-art assembly is still a craft- lots of room for innovation and better algorithms.

Acknowledgements

CBCB

- Steven Salzberg
- Art Delcher
- Adam Phillippy
- Mihai Pop
- Dan Sommer

TIGR

- Pawel Gajer
- Martin Shumway
- Jason Miller