Genome sequence assembly

Assembly concepts and methods

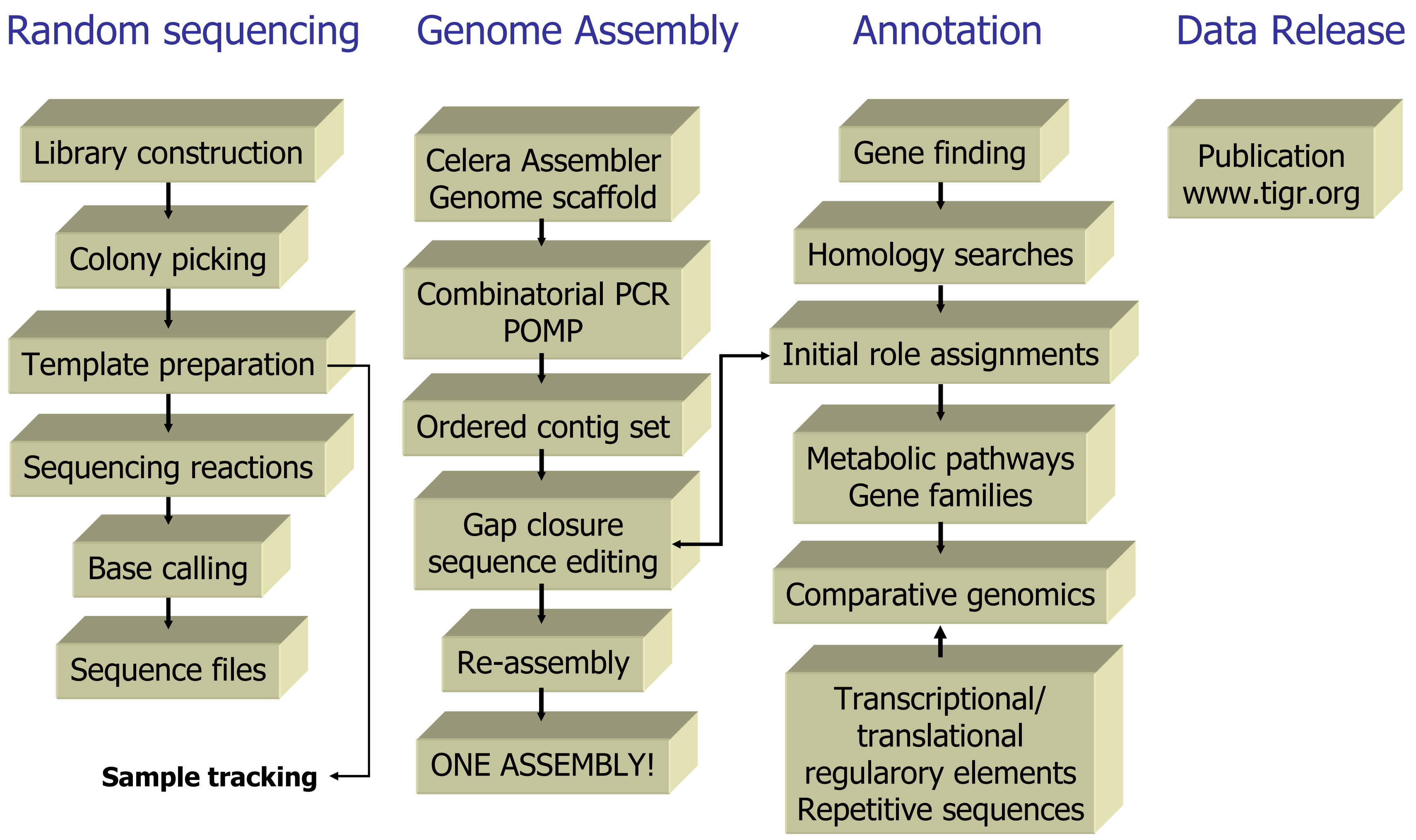
Mihai Pop & Michael Schatz
Center for Bioinformatics and Computational Biology
University of Maryland

August 13, 2006

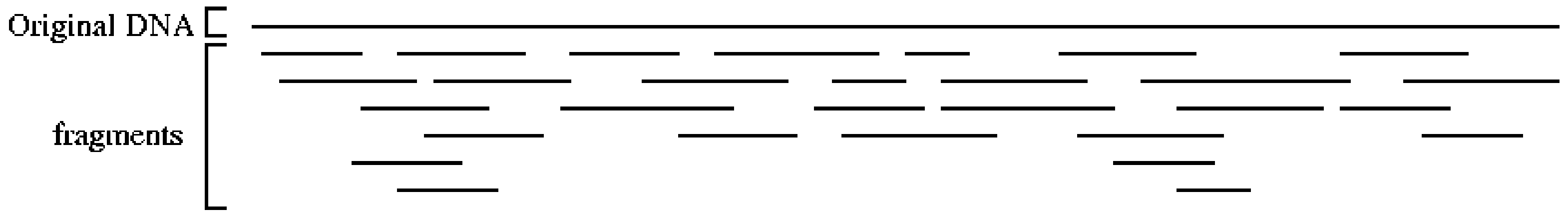
Outline

- Shotgun sequencing overview
- Shotgun sequencing statistics
- Theoretical Foundations
- Assembly algorithms
- Scaffolding

A Genome Sequencing Project

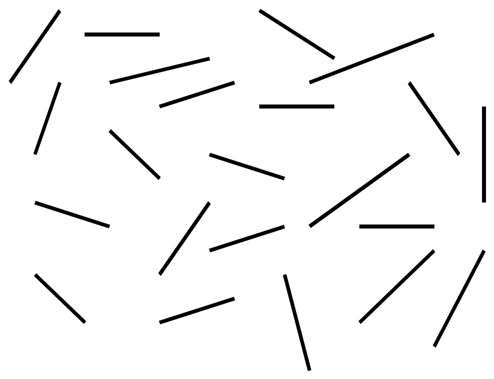


Building a library

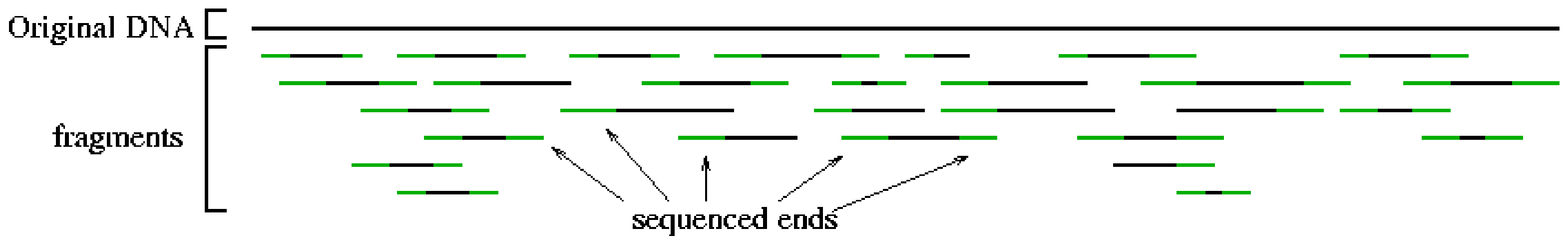


- Break DNA into random fragments (8-10x coverage)

Actual situation

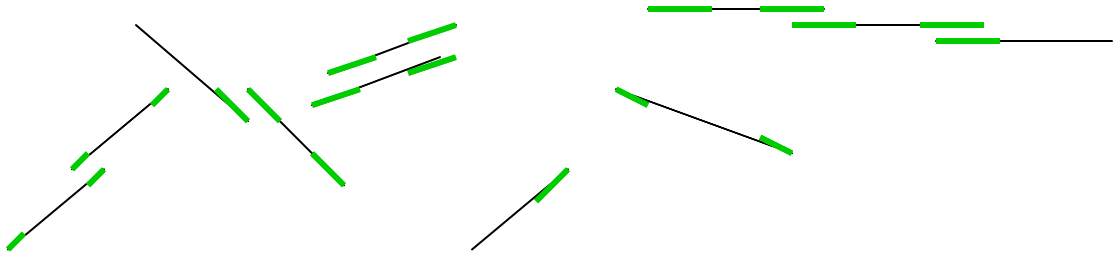


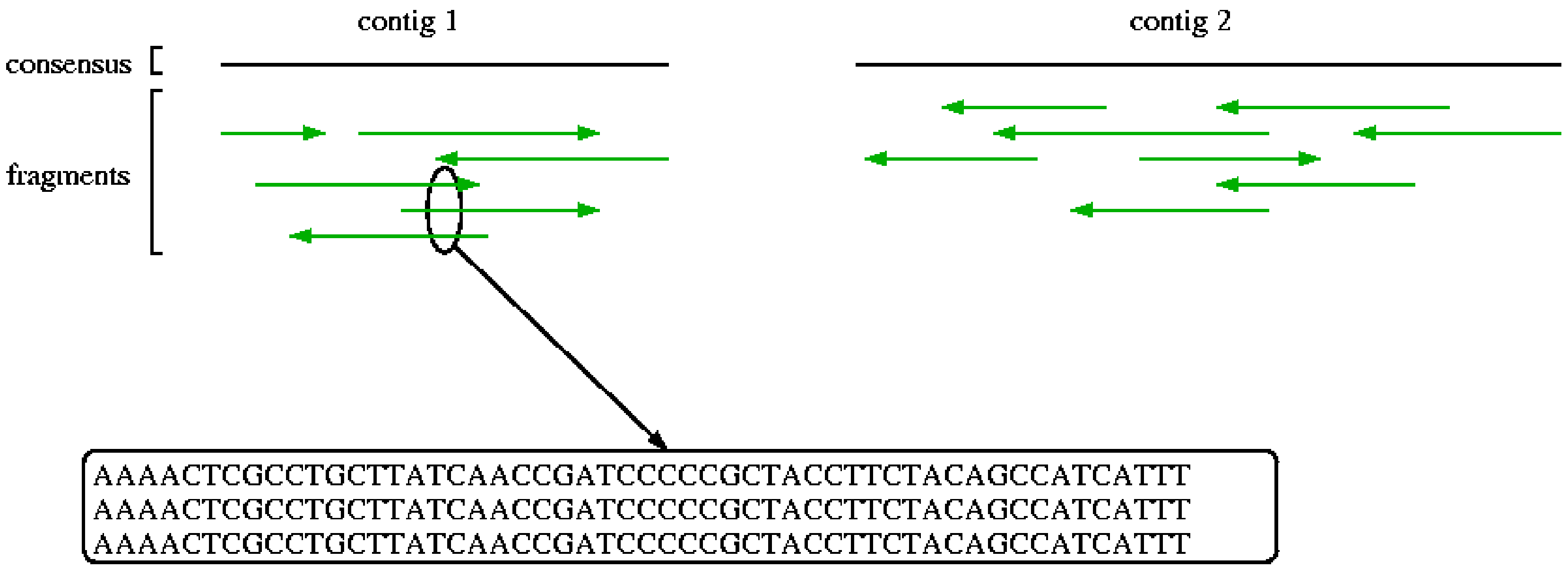
Building a library



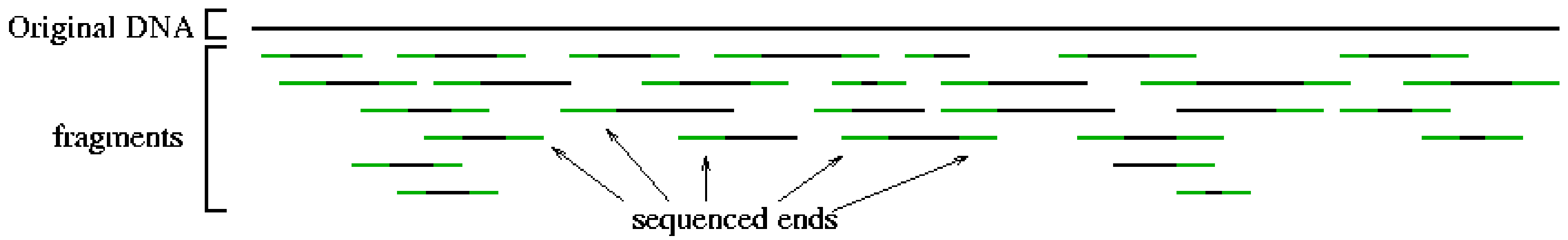
- Break DNA into random fragments (8-10x coverage)
- Sequence the ends of the fragments
  - Amplify the fragments in a vector
  - Sequence 800-1000 (500-700) bases at each end of the fragment

Assembling the fragments

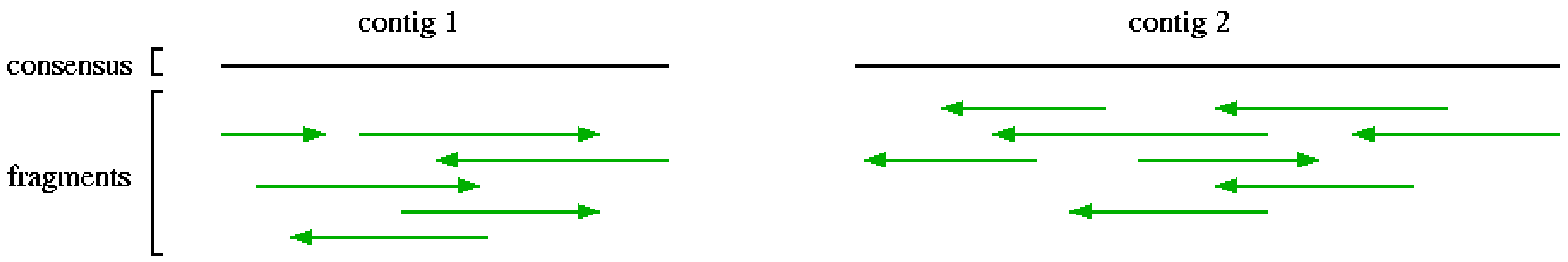



contig 1
contig 2
consensus
fragments
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT

Assembling the fragments

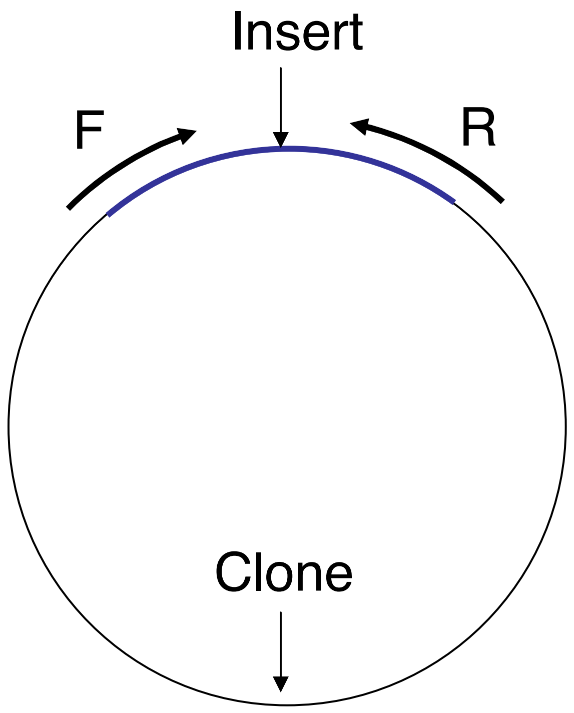


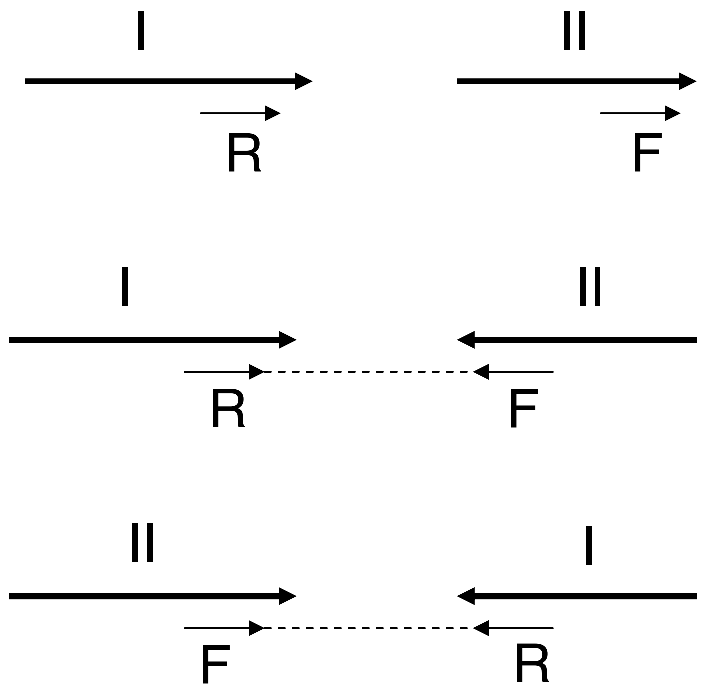
- Break DNA into random fragments (8-10x coverage)
- Sequence the ends of the fragments
- Assemble the sequenced ends



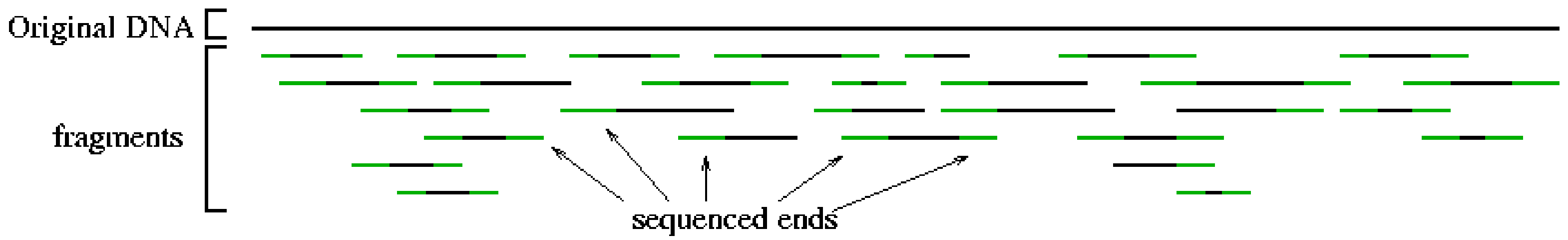
Forward-reverse constraints

- The sequenced ends are facing towards each other
- The distance between the two fragments is known (within certain experimental error)

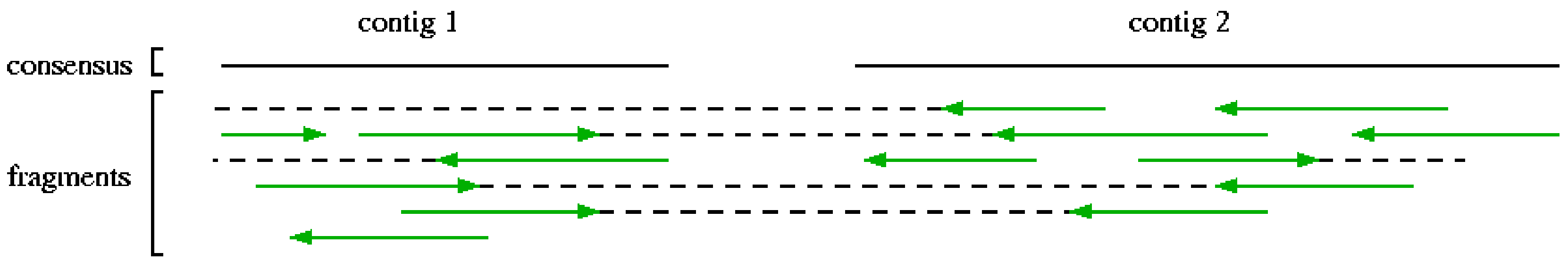




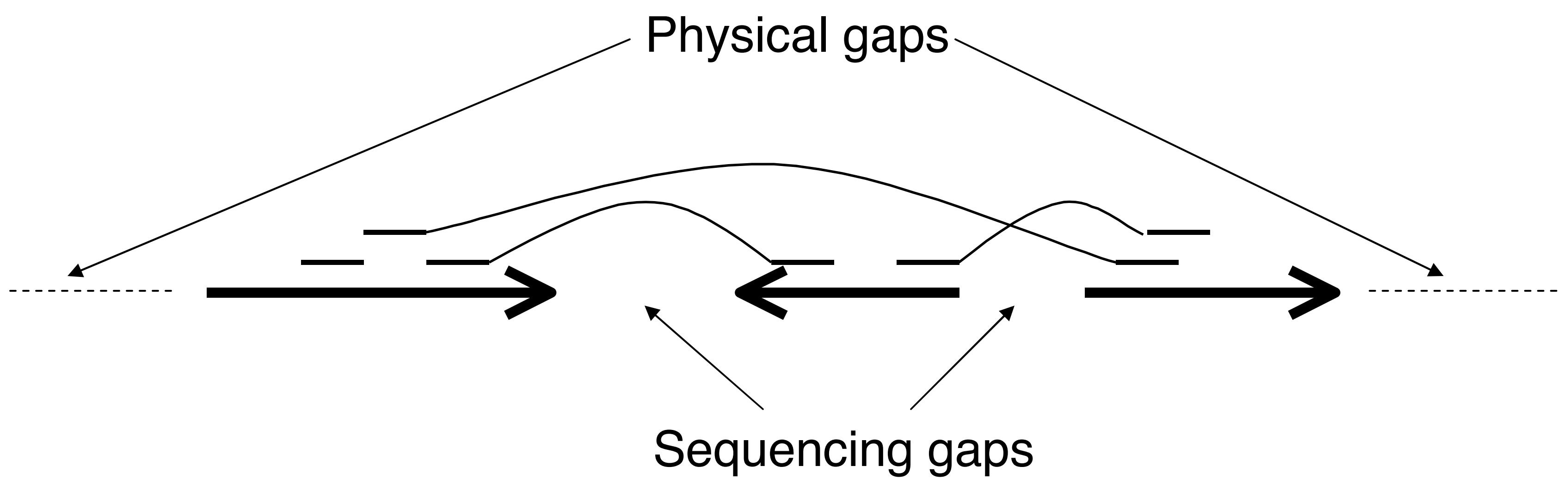
Building Scaffolds



- Break DNA into random fragments (8-10x coverage)
- Sequence the ends of the fragments
- Assemble the sequenced ends
- Build scaffolds



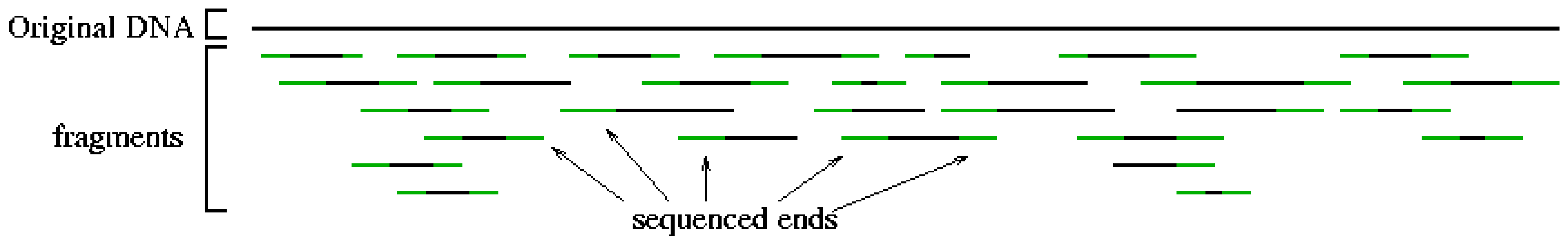
Assembly gaps



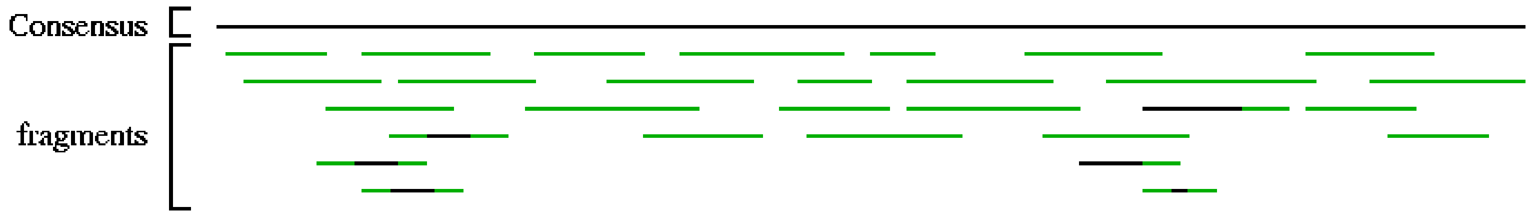
sequencing gap - we know the order and orientation of the contigs and have at least one clone spanning the gap

physical gap - no information known about the adjacent contigs, nor about the DNA spanning the gap

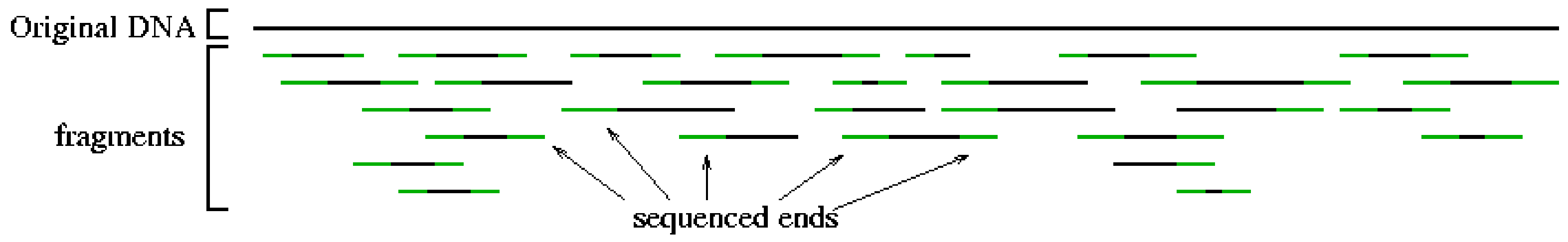
Finishing the project

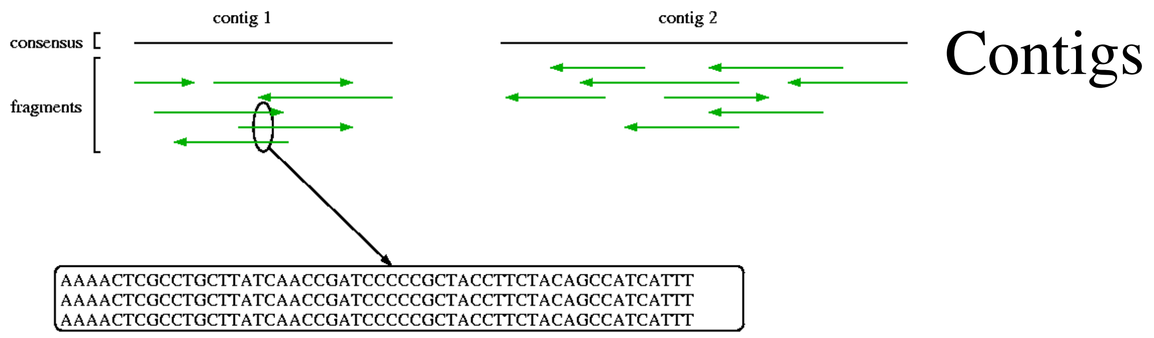


- Break DNA into random fragments (8-10x coverage)
- Sequence the ends of the fragments
- Assemble the sequenced ends
- Build scaffolds
- **Close gaps**

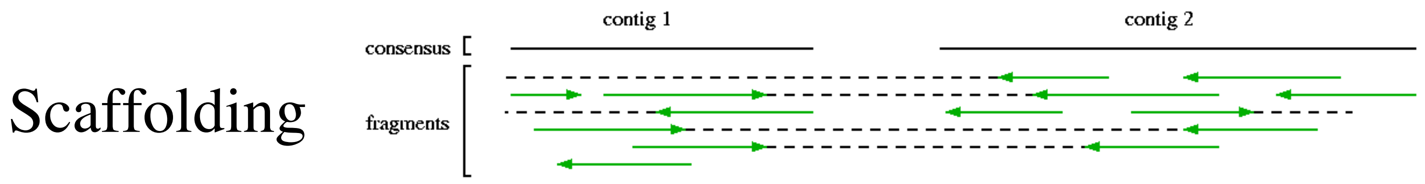


Unifying view of assembly



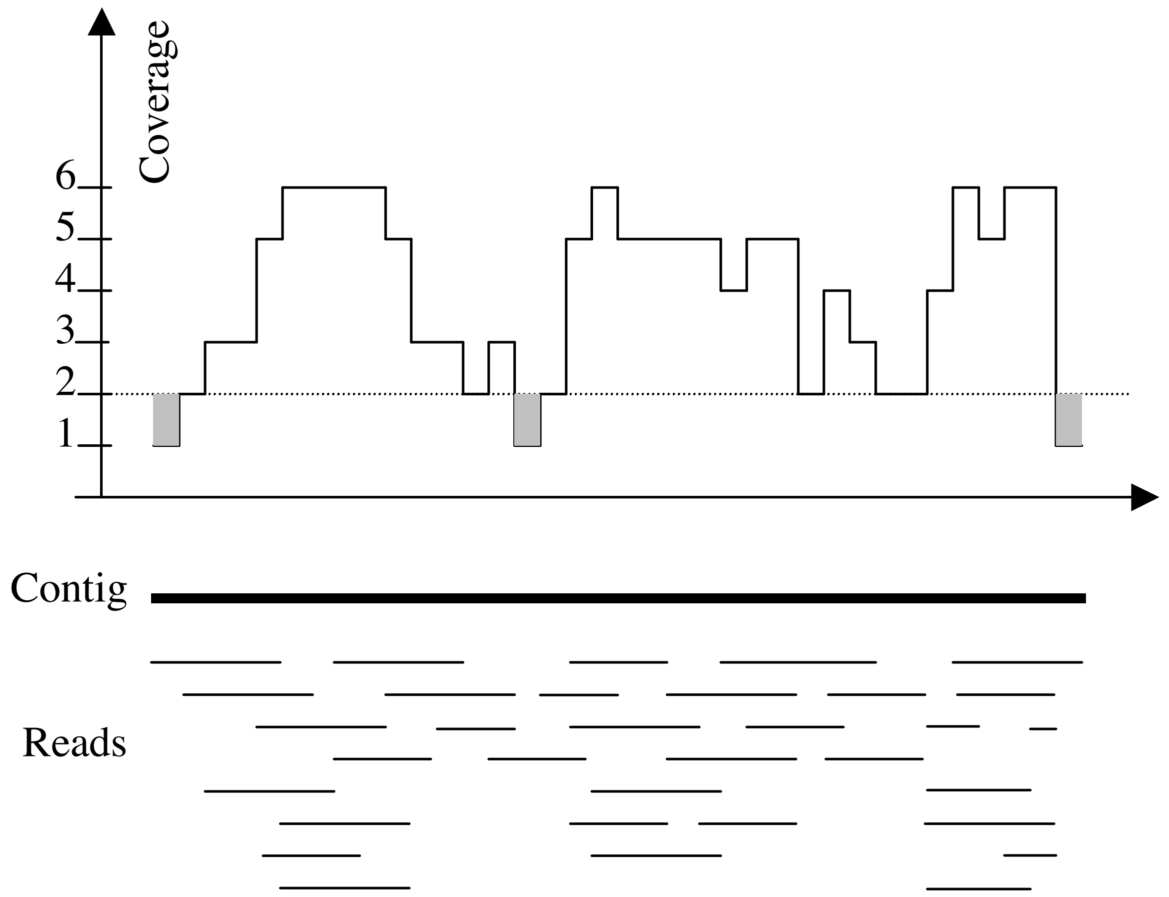


AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT



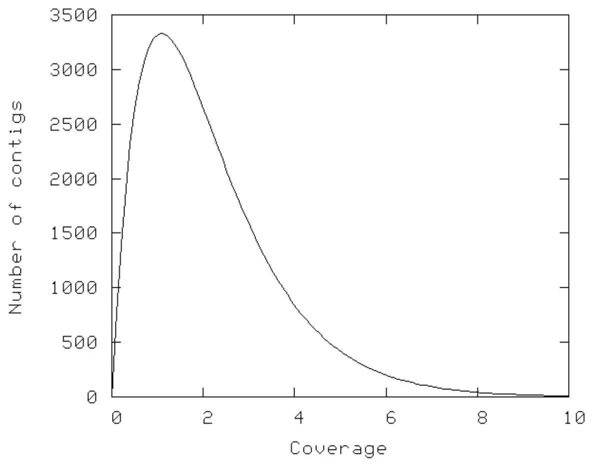
Shotgun sequencing statistics

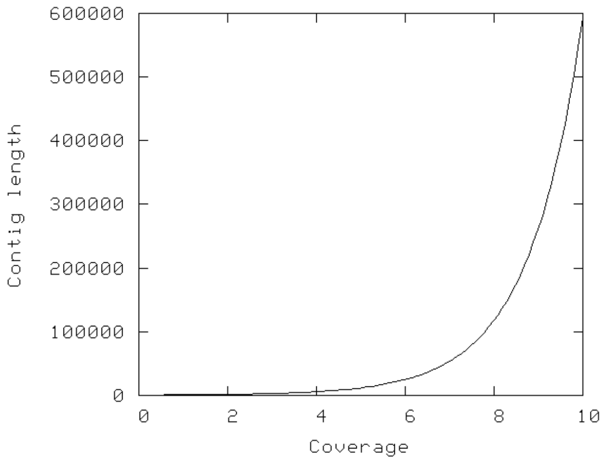
Typical contig coverage



Imagine raindrops on a sidewalk

Lander-Waterman statistics





L = read length

T = minimum overlap

G = genome size

N = number of reads

c = coverage (NL / G)

$\sigma = 1 - T/L$

E(#islands) = $Ne^{-c\sigma}$

E(island size) = $L(e^{c\sigma} - 1) / c + 1 - \sigma$

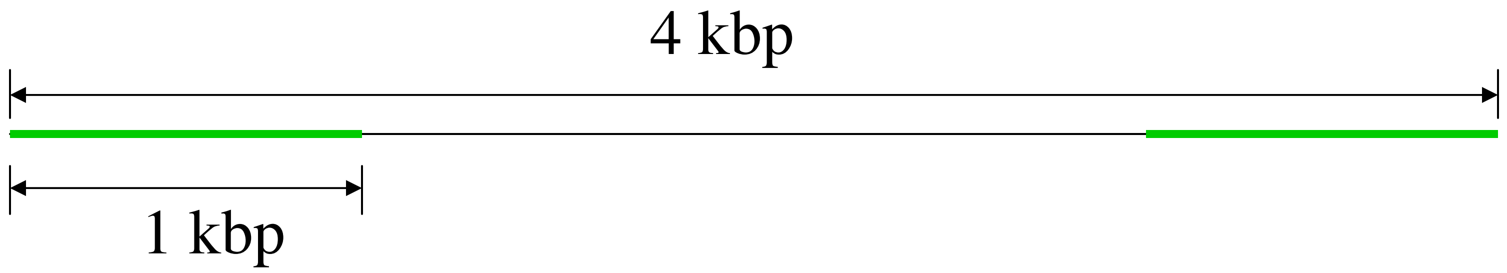
contig = island with 2 or more reads

Example

Genome size: 1 Mbp   Read Length: 600   Detectable overlap: 40

c	N	#islands	#contigs	bases not in any read	bases not in contigs
1	1,667	655	614	698	367,806
3	5,000	304	250	121	49,787
5	8,334	78	57	20	6,735
8	13,334	7	5	1	335

Read coverage vs. Clone coverage



Read coverage = 8 x

Clone (insert) coverage = ?   16

BAC-end 2x coverage implies 100x coverage by BACs

(1 BAC clone = approx. 100kbp)

Theoretical Foundations

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

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:

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

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.

Overlap Graph

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

$$\begin{tikzcd}
\overset{s_1}{\text{CACCC}} \arrow[rr, shift left, "2"] \arrow[dr, shift left, "2"] & & \overset{s_2}{\text{CCGGGTGC}} \arrow[ll, shift left, "1"] \arrow[dl, shift left, "1"] \\
& \overset{s_3}{\text{CCACC}} \arrow[ul, shift left, "4"] \arrow[ur, shift left, "2"] &
\end{tikzcd}$$

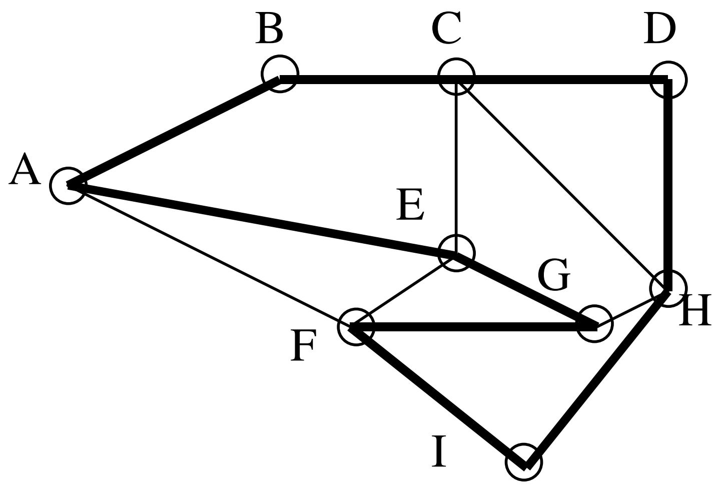
$$V = \{s_1, s_2, s_3\} \qquad E = \{s_i, s_j\}$$

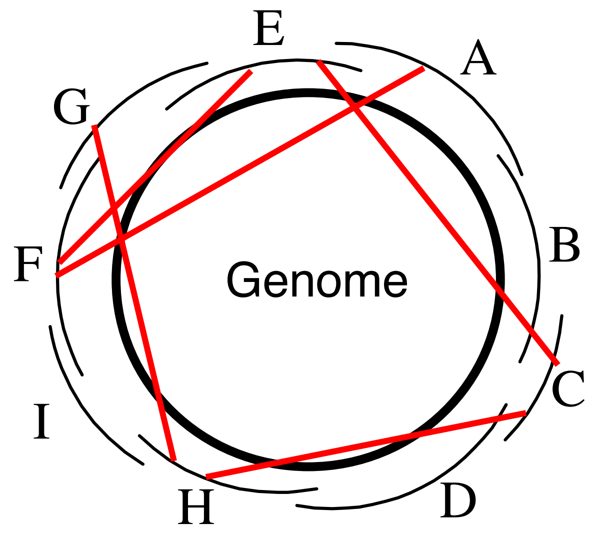
$$o(s_i,s_j) = |v| \mid s_i = uv,\ s_j = vw$$

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

Paths through graphs and assembly

- Hamiltonian circuit: visit each node (city) exactly once, returning to the start





Greedy Approximation

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

$$\begin{tikzcd}
\overset{s3}{\text{CCACC}} \arrow[r, "4"] & \overset{s1}{\text{CACCC}} \arrow[r, "2"] & \overset{s2}{\text{CCGGGTGC}}
\end{tikzcd}$$

GREEDY(S) $\leq$ 2.5 OPT(S)

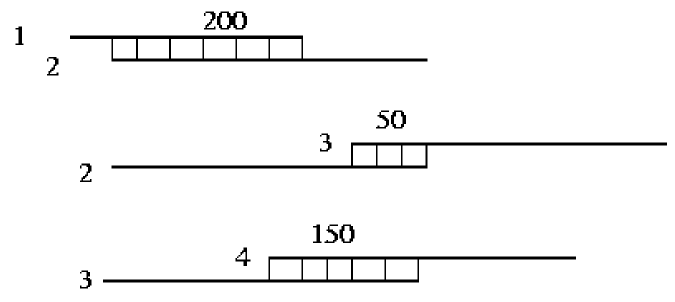
Runtime $O(\binom{n}{2}\, l^2)$

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

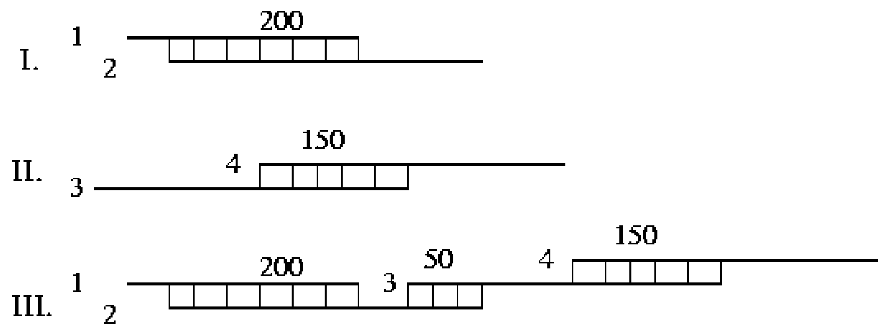
Greedy Assembly

Build a rough map of fragment overlaps

1. Pick the largest scoring overlap
2. Merge the two fragments
3. Repeat until no more merges can be done



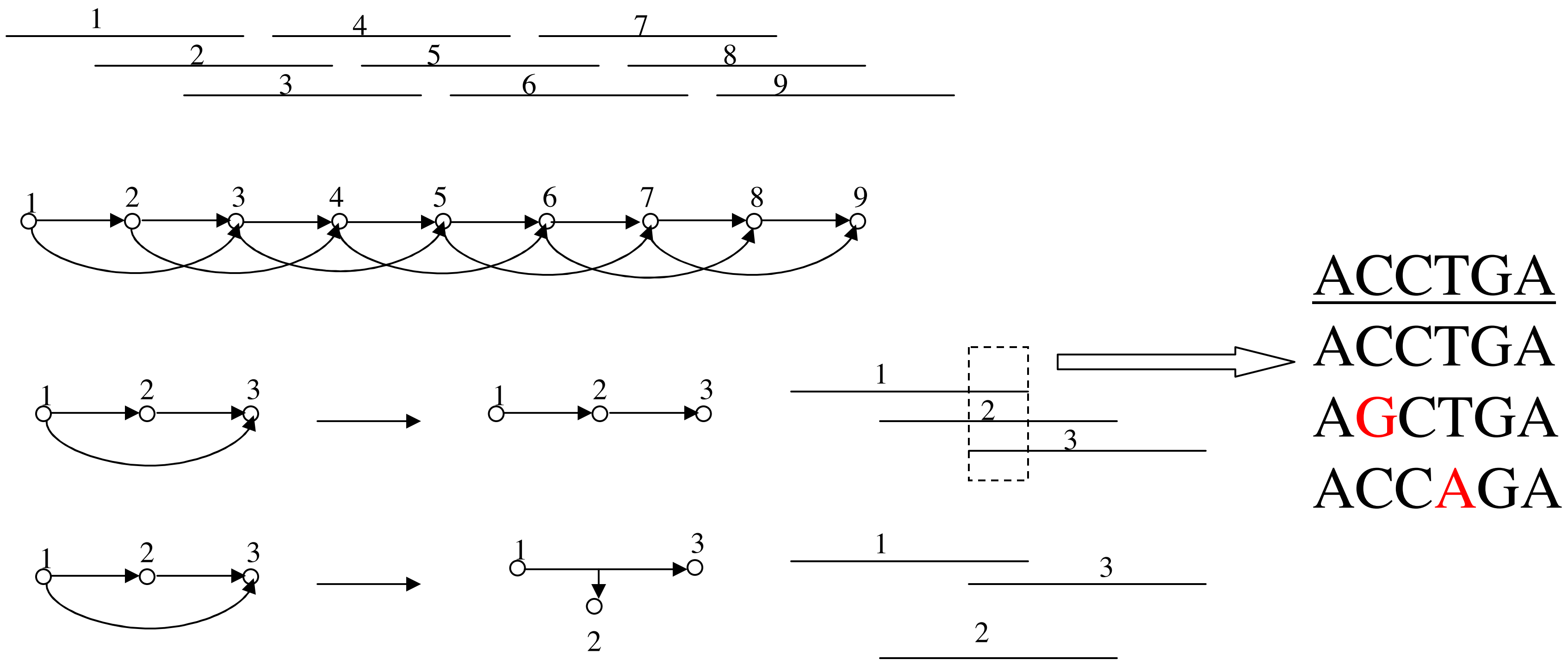
- TIGR Assembler
- phrap
- gap



Overlap-layout-consensus

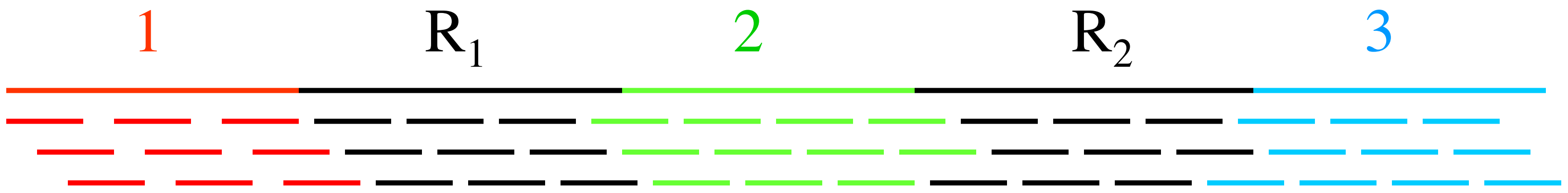
Main entity: read

Relationship between reads: overlap

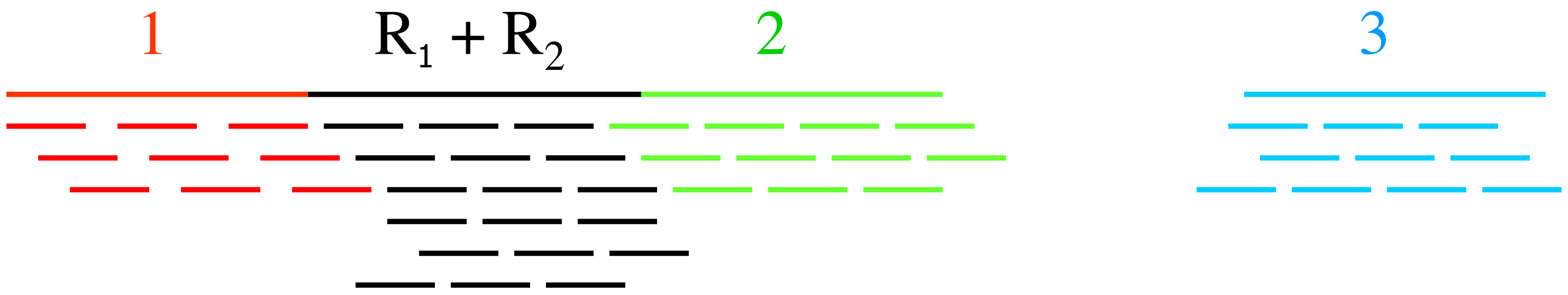


Repeats!

True Layout of Reads

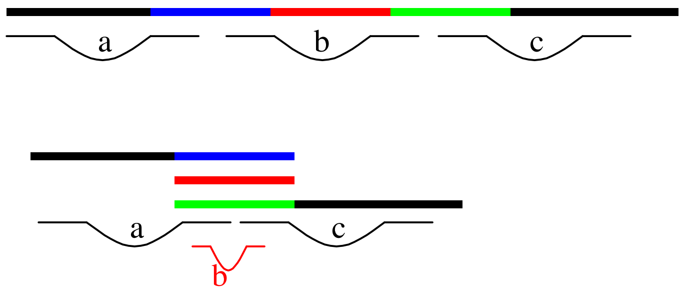


Greedy Reconstruction

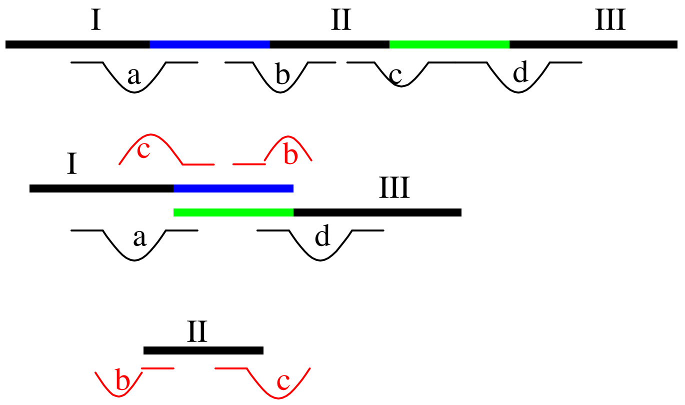


Mis-assembled repeats

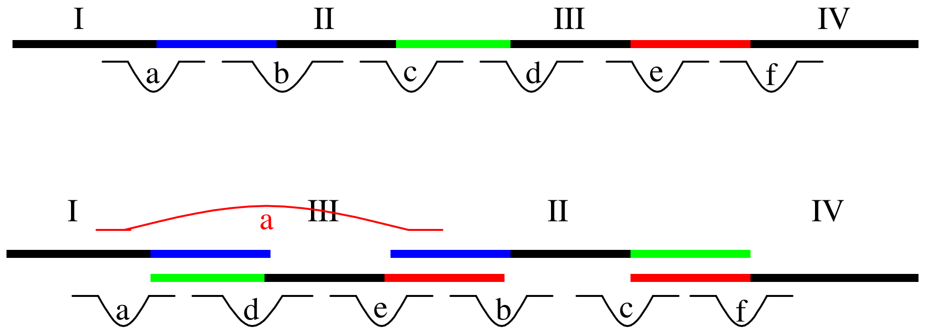
collapsed tandem



excision



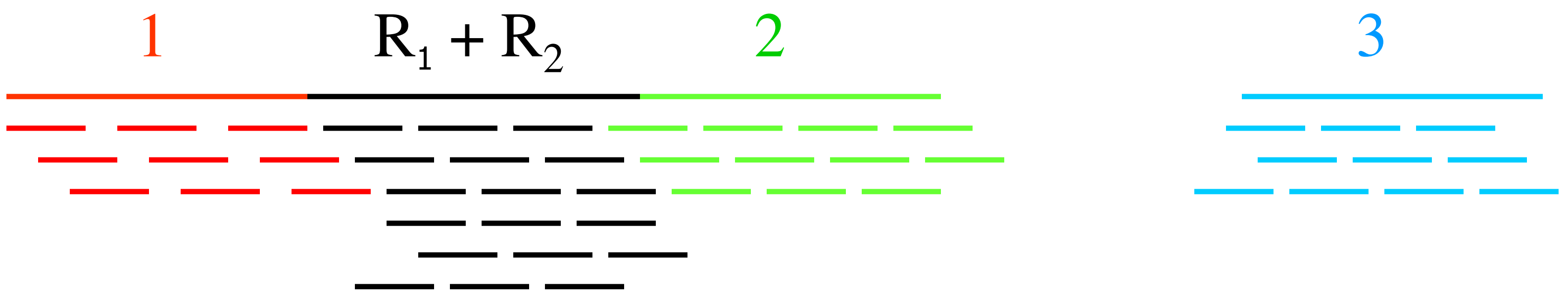
rearrangement



Modern Assembly

Try to detect presence of repeats by

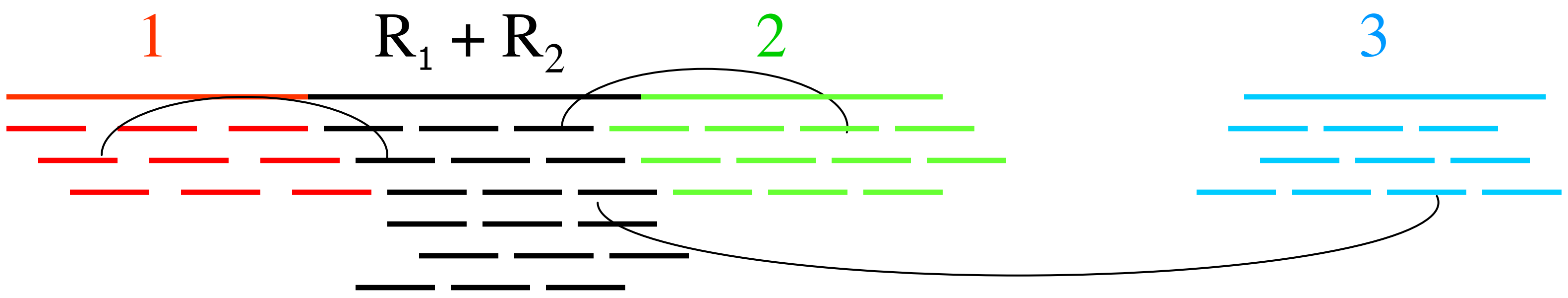
1. Unusual depth of coverage (arrival rate)
2. Mate Pair information
3. Forks in overlap graph



Modern Assembly

Try to detect presence of repeats by

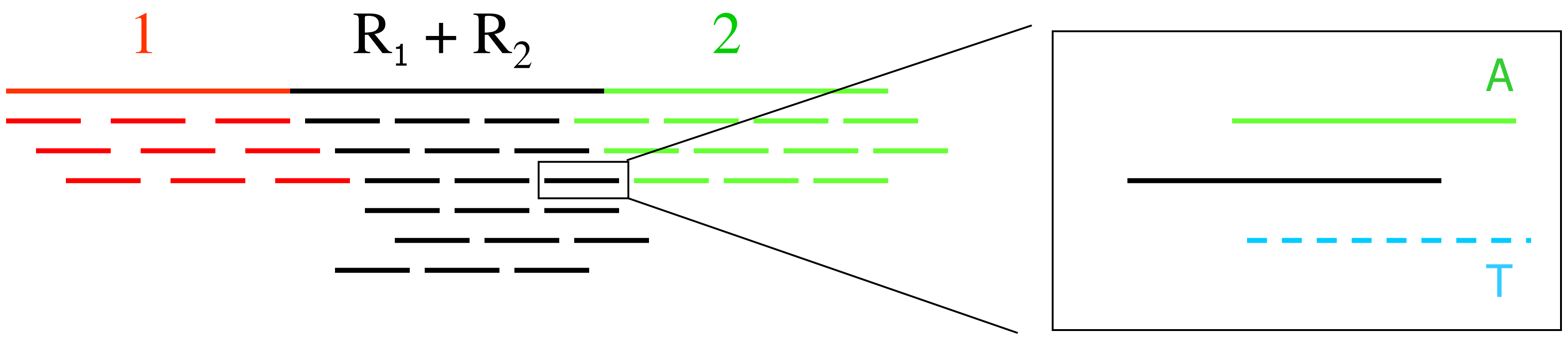
1. Unusual depth of coverage (arrival rate)
2. Mate Pair information
3. Forks in overlap graph



Modern Assembly

Try to detect presence of repeats by

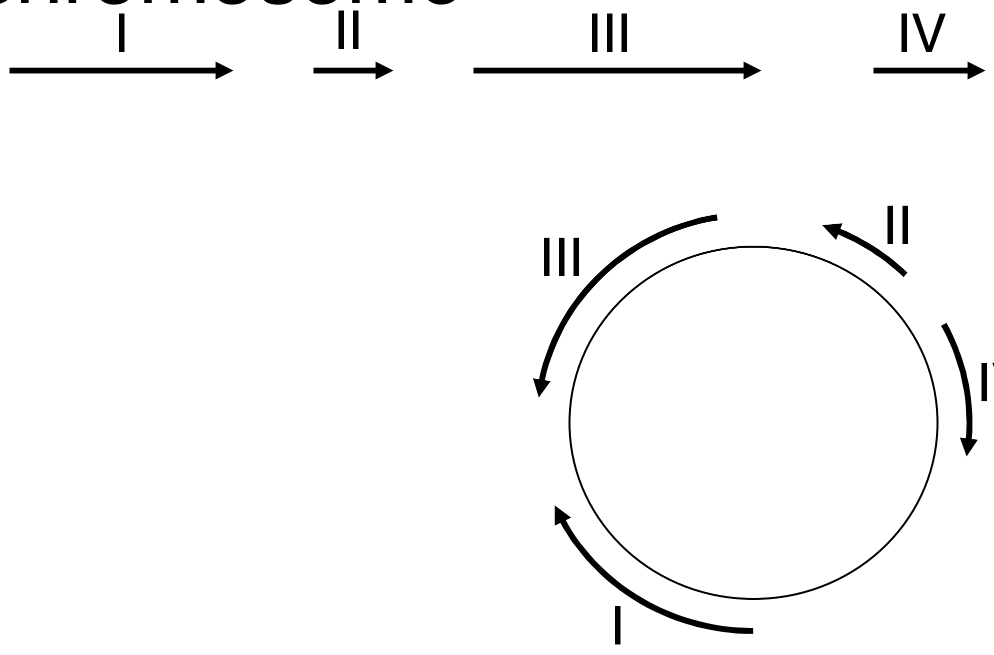
1. Unusual depth of coverage (arrival rate)
2. Mate Pair information
3. Forks in overlap graph



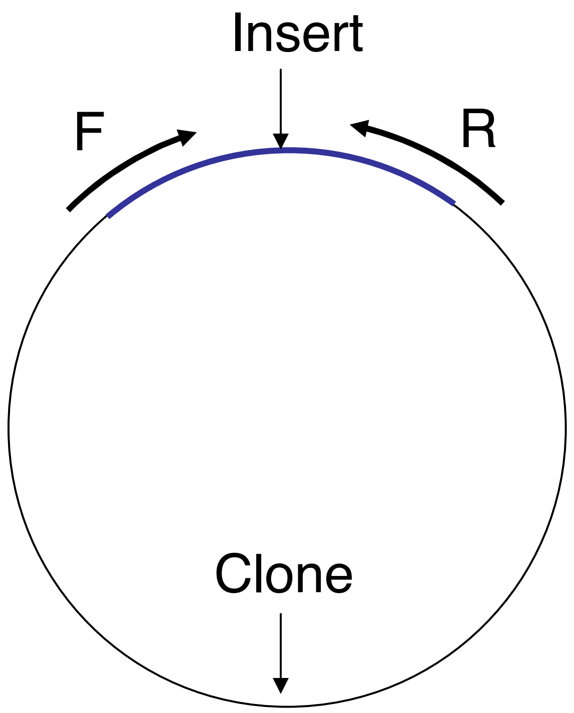
SCAFFOLDING

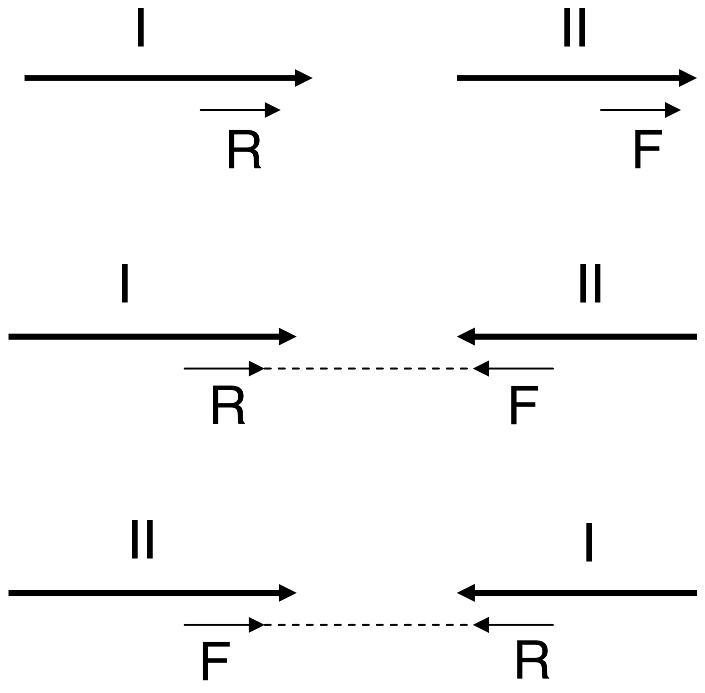
Scaffolding

- Given a set of non-overlapping contigs order and orient them along a chromosome

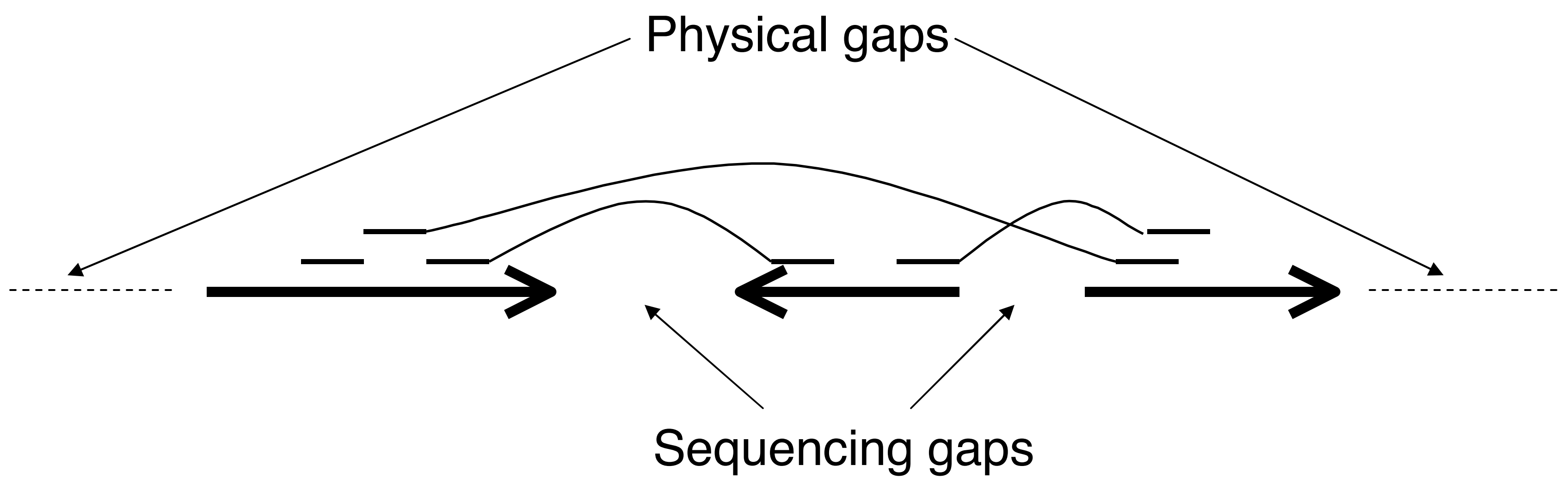


Clone-mates





Scaffolder output



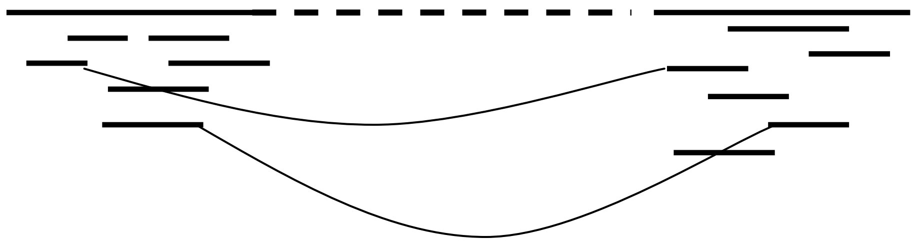
- order and orientation of contigs
- size of gaps between contigs
- linking evidence: mate-pairs spanning gaps

Problems with the data

- Incorrect sizing of inserts
  - cut from gel – sizing is subjective
  - error increases with size
- Chimeras (ends belong to different inserts)
  - biological reasons (esp. for large sized inserts)
  - sample tracking (human error)
- Software must handle a certain error rate.

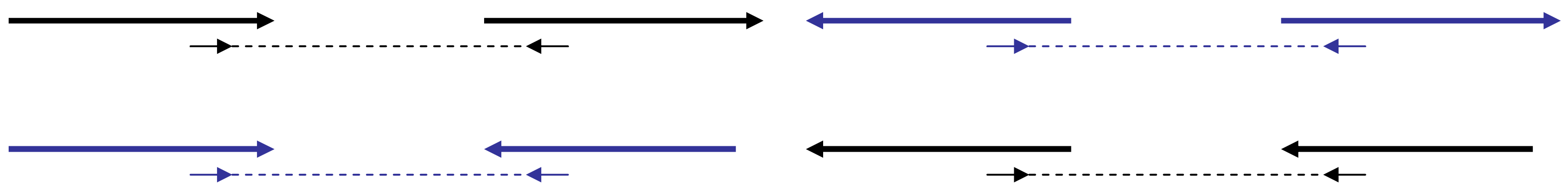
Theoretical abstraction

- Given a set of entities (reads/contigs) and constraints between them (overlaps/mate pairs) provide a linear/circular embedding that preserves most constraints.



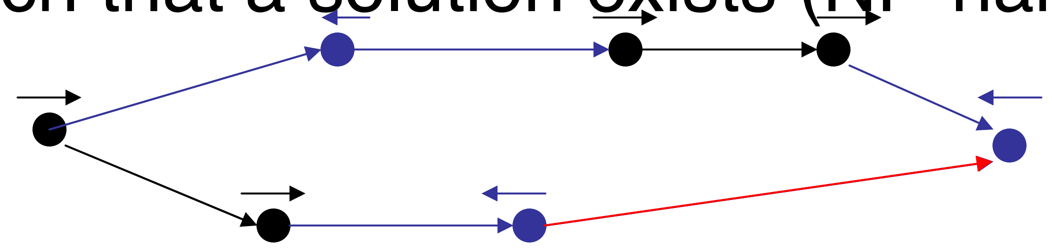
Graph representation

- Nodes: contigs
- Directed edges: constraints on relative placement of contigs – relative order and relative orientation
- Embedding: order (coordinate along chromosome) and orientation (strand sampled)



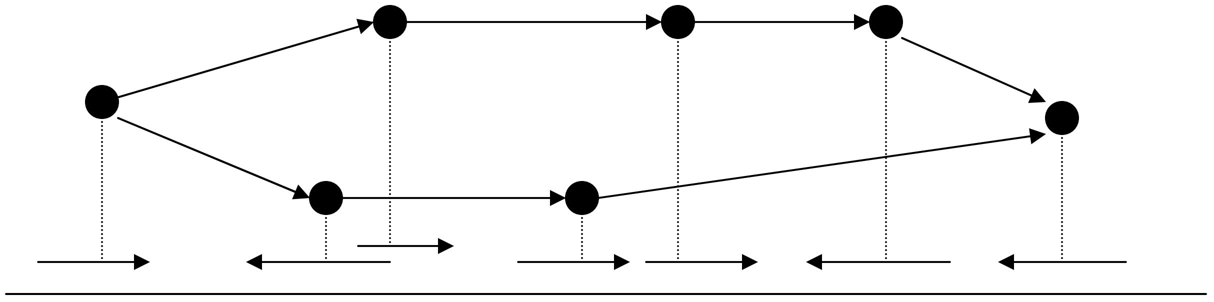
Challenges

- Orientation – node coloring problem (forward/reverse)
  - feasibility – no cycles with odd number of "reversal" edges
  - optimality – remove minimum number of edges
    such that a solution exists (NP-hard)



Challenges

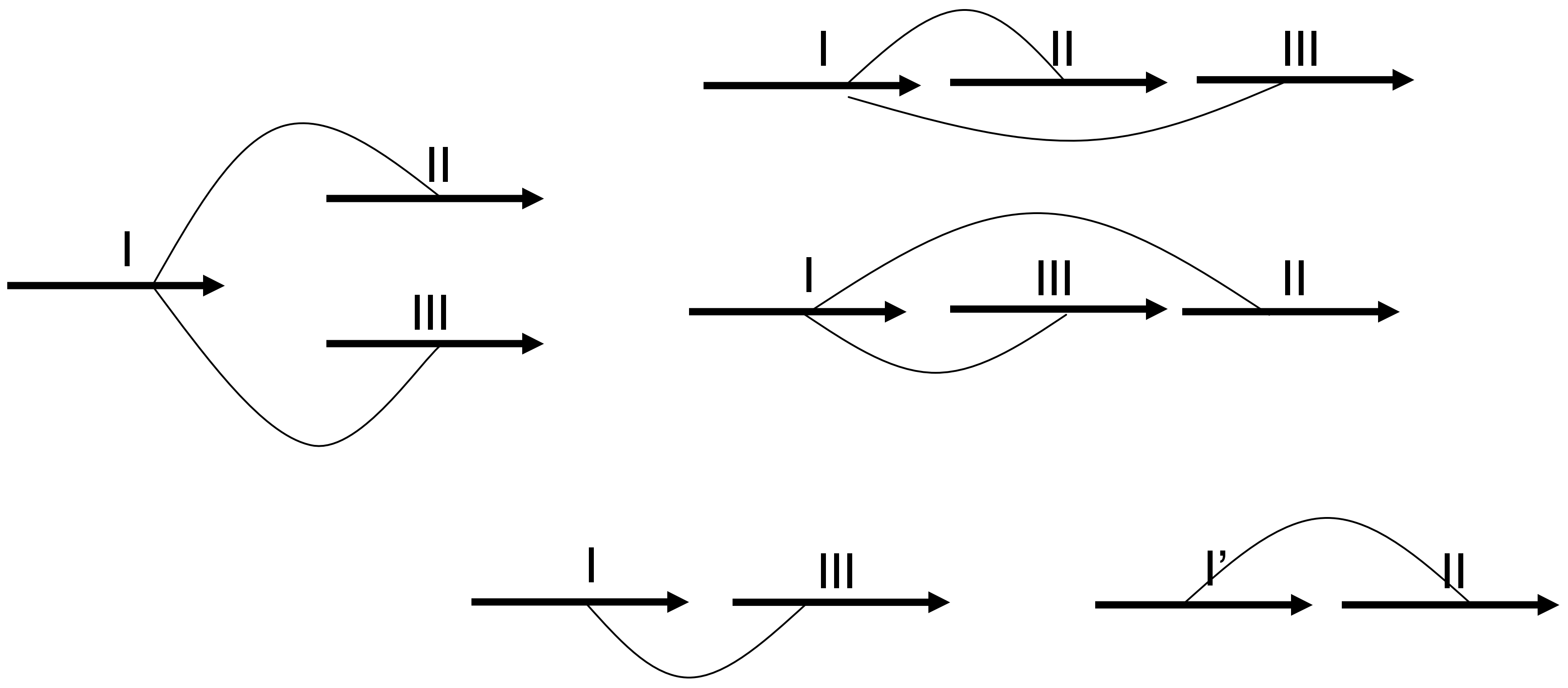
- Ordering – generate a linear embedding
  - feasibility – lengths of parallel DAG paths are consistent
  - optimality – remove minimum number of edges
    such that DAG is feasible (NP-hard)



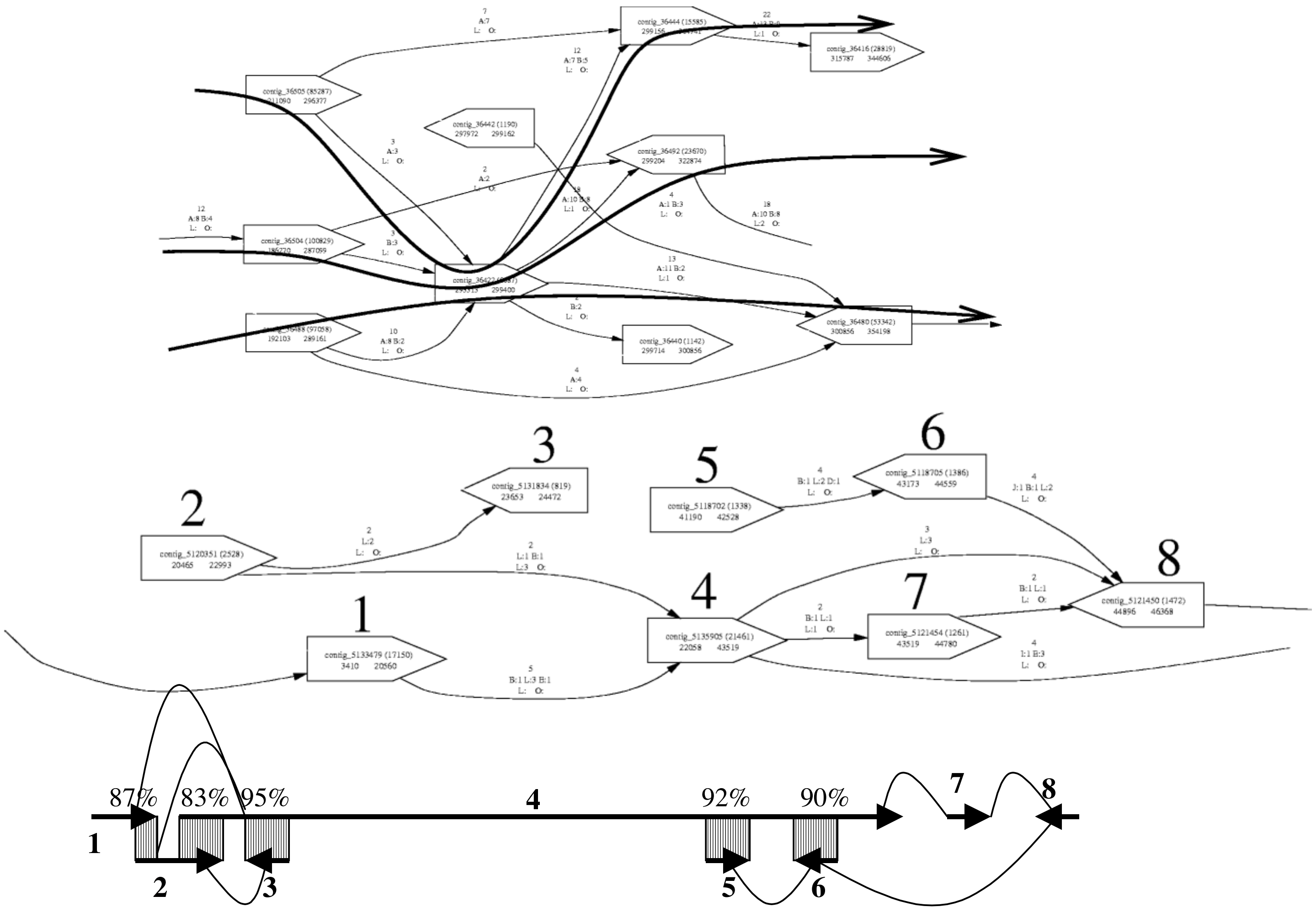
The real world

- Use of scaffolds
  - Analysis – longest unambiguous sub-graphs
  - Finishing – present all “reliable” relationships between contigs
- Sources of error
  - mis-assemblies
  - sizing errors (increases with library size)
  - chimeras

Ambiguous scaffold

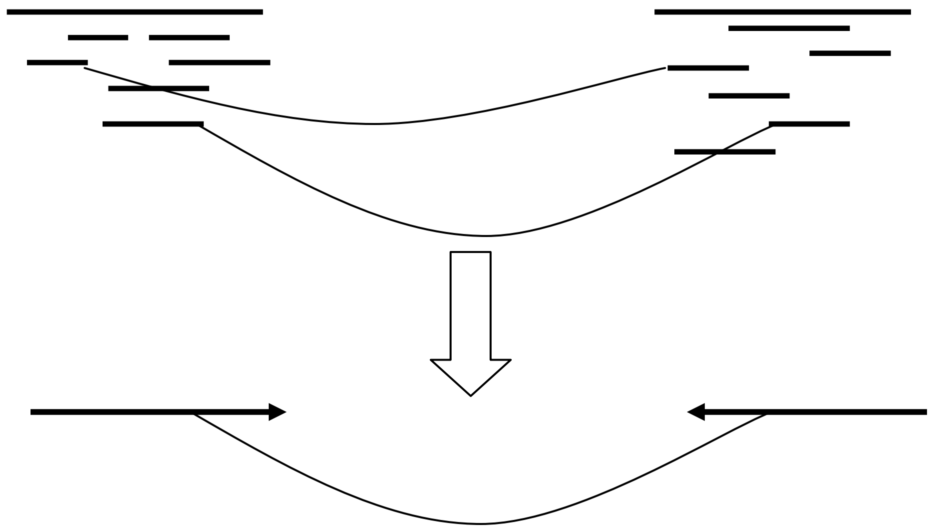


Repeats vs. Haplotypes



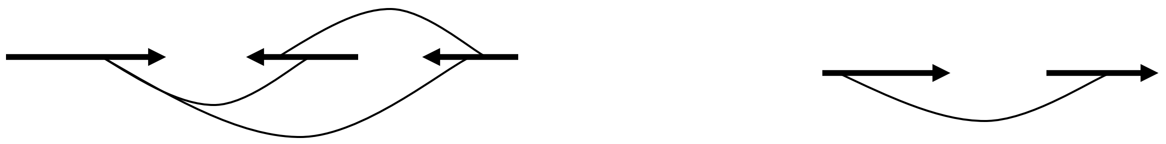
Hierarchical scaffolding

1. For each contig pair, consolidate all linking data into a single relationship – 2 correct links required

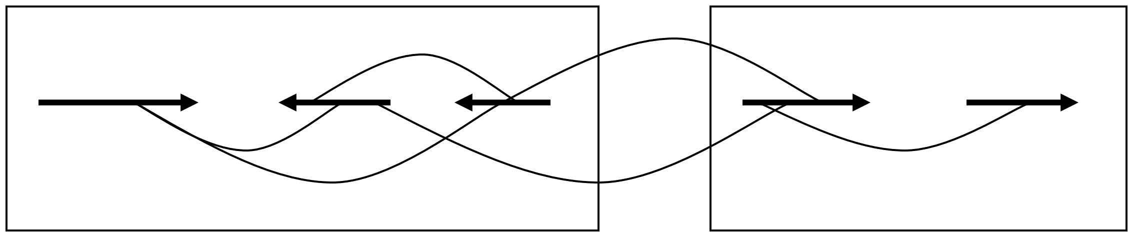


Hierarchical scaffolding

2. Use most reliable links to build scaffolds

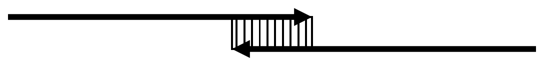


3. Repeatedly build super-scaffolds based on less reliable linking data

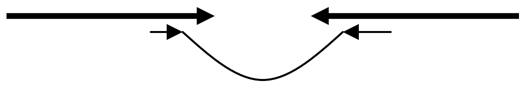


Linking information

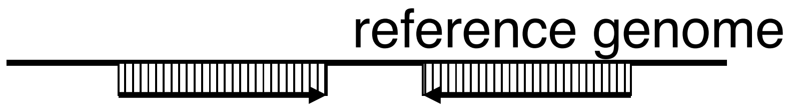
- Overlaps



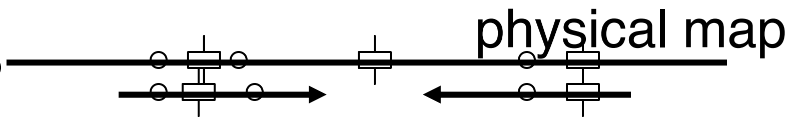
- Mate-pair links



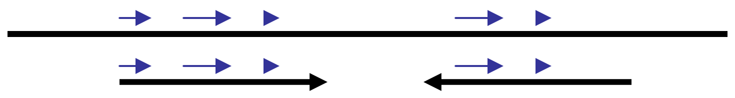
- Similarity links



- Physical markers



- Gene synteny



BAMBUS (bamboo)

Best effort **A**ttempt

Multiple **B**ranches allowed

Order, **O**rient

References

Review of assembly Pop, M. *Shotgun sequence assembly*; in Advances in Computers vol. 60. Elsevier, 2004, pp. 193-247

TIGR Assembler Sutton, G.G., et al., *TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects.* Genome Science and Technology, 1995. 1:9-19.

Celera Assembler Myers, E.W. et al. 2000. *A whole-genome assembly of Drosophila.* Science 287: 2196-2204.

Arachne Batzoglou, S., et al. 2002. *ARACHNE: a whole-genome shotgun assembler.* Genome Res 12: 177-189.

Jaffe, D.B., et al. 2003. *Whole-genome sequence assembly for Mammalian genomes: arachne 2*. Genome Res 13: 91-96.

phrap Green, P., *PHRAP documentation: ALGORITHMS*. 1994 http://www.phrap.org.

Euler Pevzner, P. et al. 2001. *Fragment assembly with double-barreled data*. Bioinformatics. 17: S225-S233.

CAP3 Huang, X. and A. Madan, *CAP3: A DNA Sequence Assembly Program.* Genome Research, 1999. 9:868-877.

BAMBUS Pop, M. et al. *Hierarchical scaffolding with Bambus*, Genome Research, 2004, 14(1):149-159