Towards More Effective Formulations of the Genome Assembly Problem

Alexandru Tomescu
Department of Computer Science
University of Helsinki, Finland

DACS
June 26, 2015


VIKING LINE

Central dogma of biology

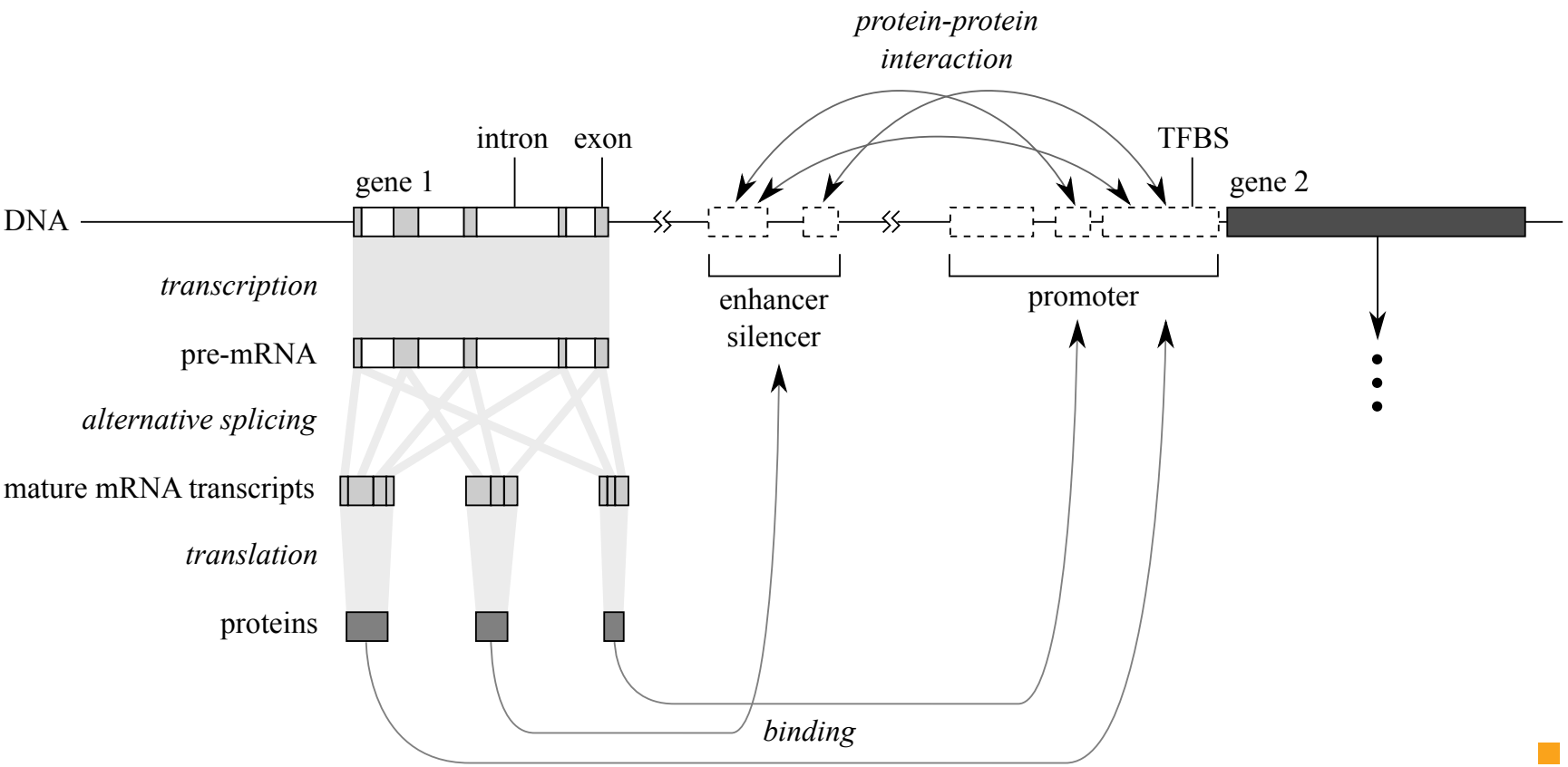


Image taken from
Genome-Scale Algorithm Design, Cambridge University Press, 2015

Sequencing atlas

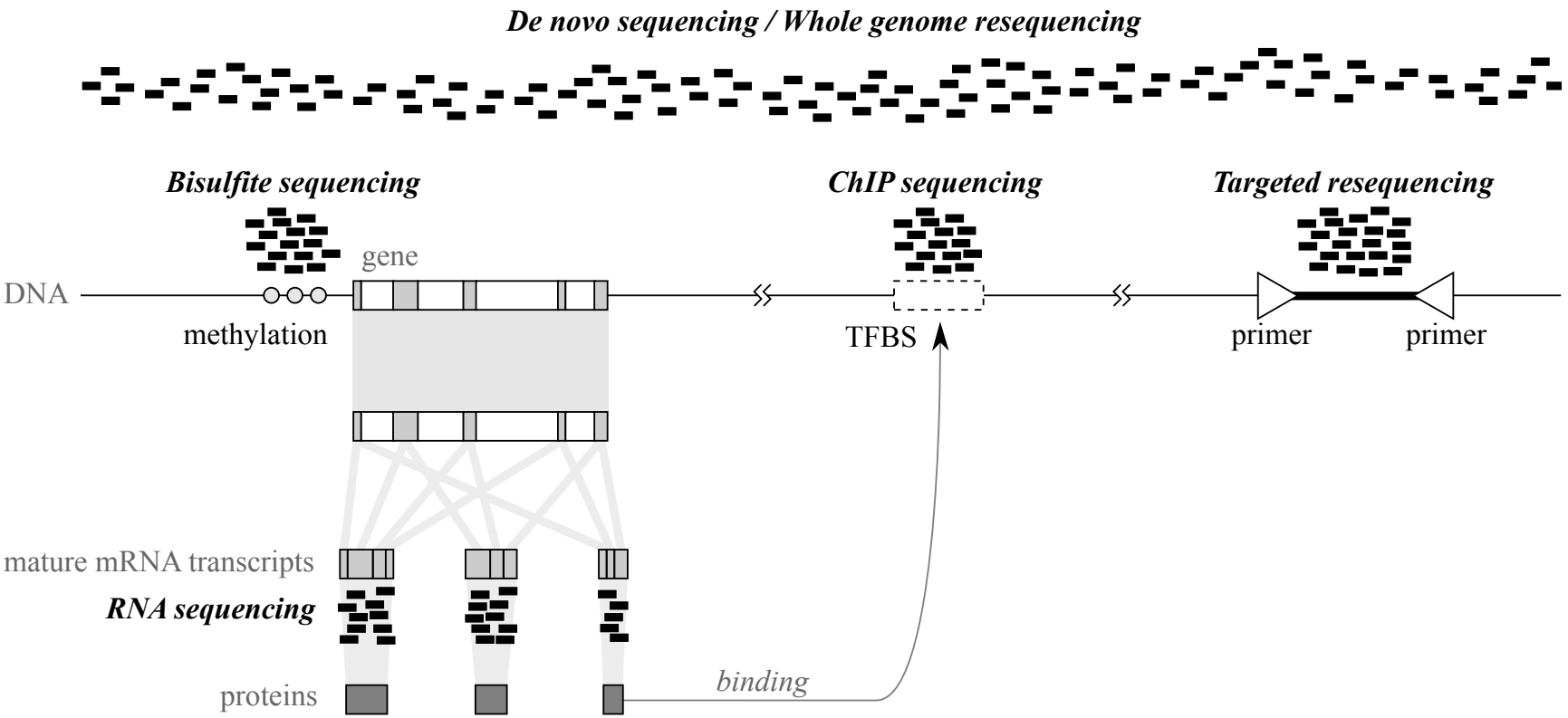
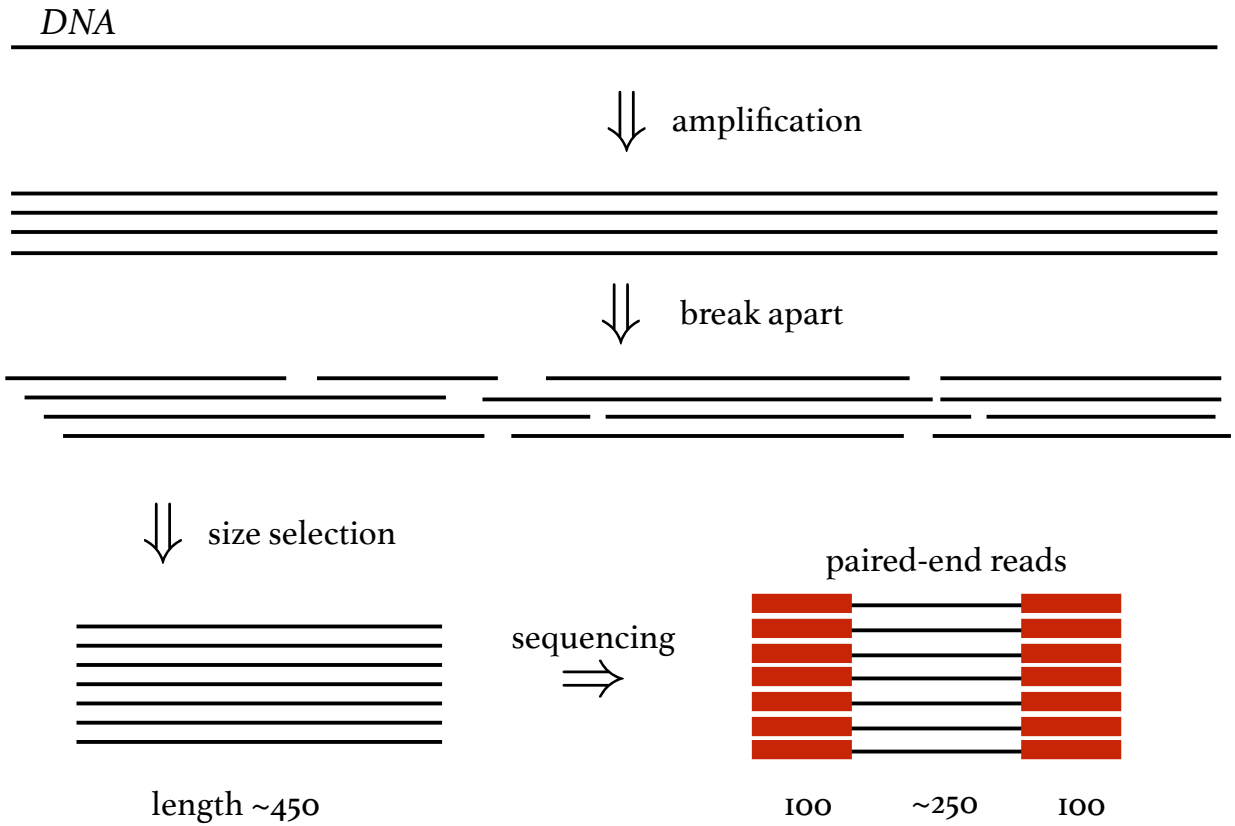
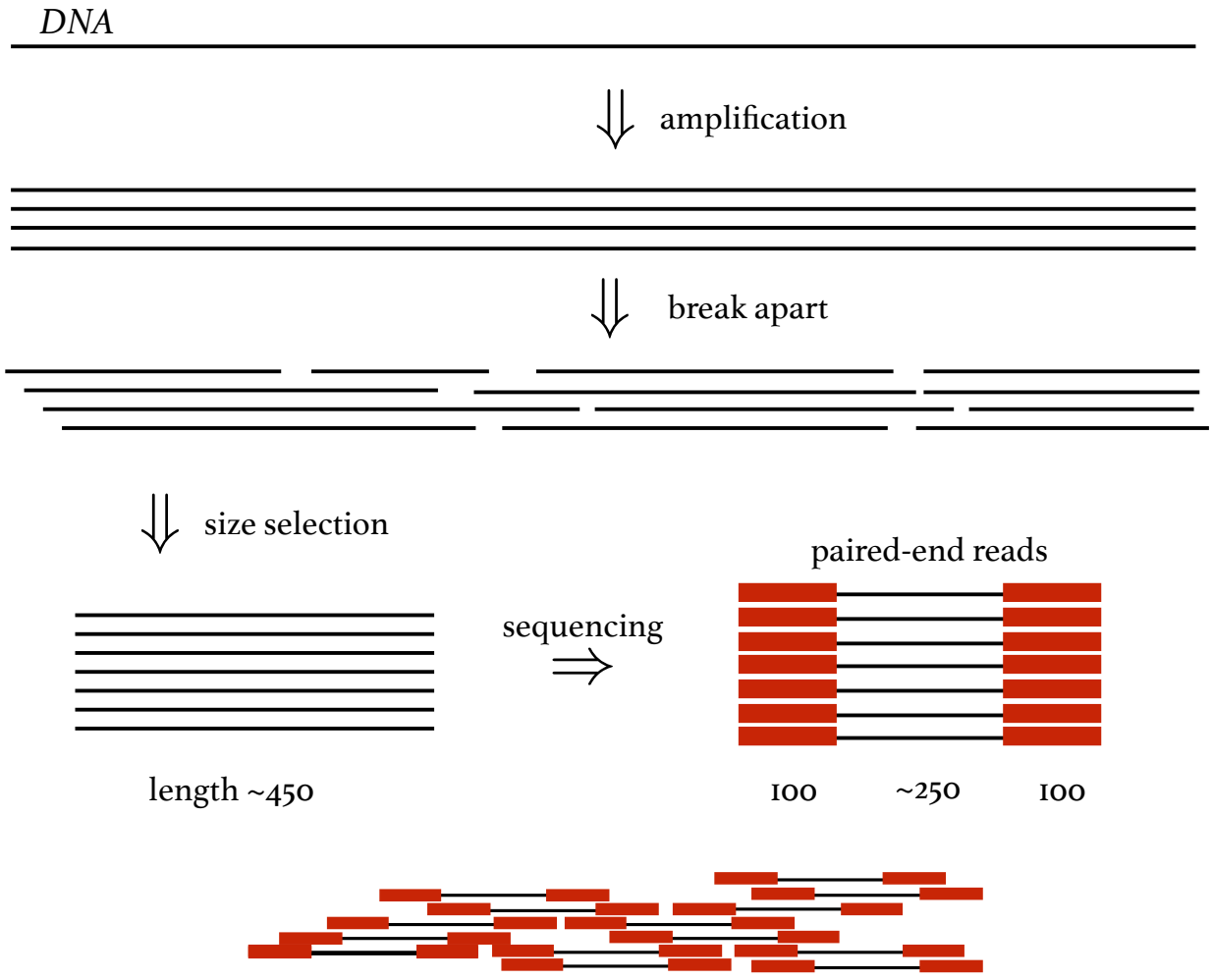


Image taken from
Genome-Scale Algorithm Design, Cambridge University Press, 2015

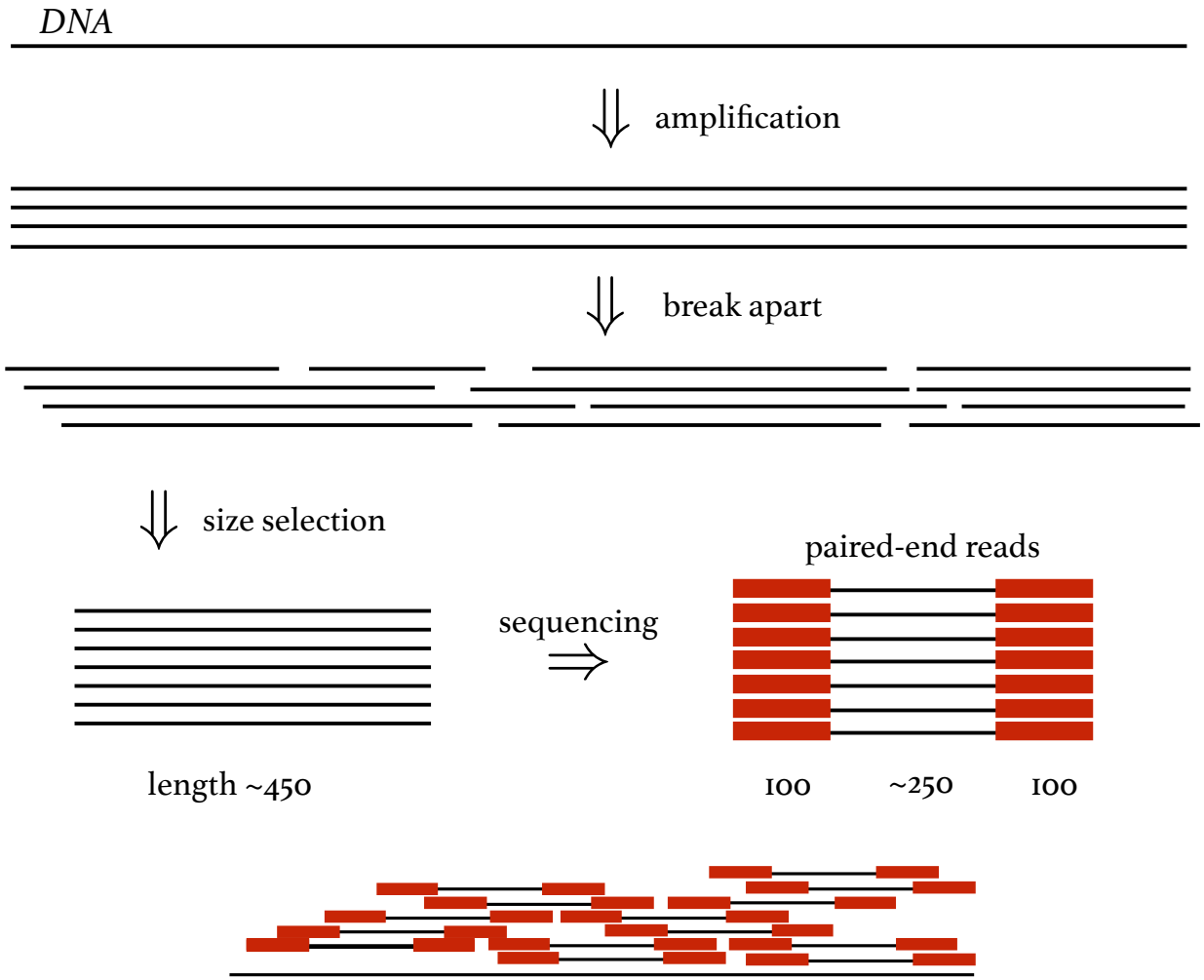
High-throughput sequencing



High-throughput sequencing



High-throughput sequencing



We assume here that DNA is a single stranded, single chromosome

Illumina HiSeqX

Genome assembly problem

E.coli	$4.6 \cdot 10^6$
Human	$3.2 \cdot 10^9$
Spurce	$25 \cdot 10^9$

INPUT: A collection of paired-end reads
OUTPUT: The genome

Genome assembly problem

E.coli	$4.6 \cdot 10^6$
Human	$3.2 \cdot 10^9$
Spurce	$25 \cdot 10^9$

INPUT: A collection of paired-end reads
OUTPUT: The genome

Initial formulations:

- Shortest superstring problem (NP-hard)
- Build a graph with reads as nodes, and significant overlaps between reads as directed edges:
  - Find a walk that passes through every node exactly once (NP-complete)
  - Find a walk that passes through every node at least once

Genome assembly problem

E.coli	$4.6 \cdot 10^6$
Human	$3.2 \cdot 10^9$
Spurce	$25 \cdot 10^9$

INPUT: A collection of paired-end reads
OUTPUT: The genome

Initial formulations:

- Shortest superstring problem (NP-hard)
- Build a graph with reads as nodes, and significant overlaps between reads as directed edges:
  - Find a walk that passes through every node exactly once (NP-complete)
  - Find a walk that passes through every node at least once

Unrealistic:

- Longer repeated regions are collapsed
- Genome coverage is not uniform
- We cannot choose between multiple solutions

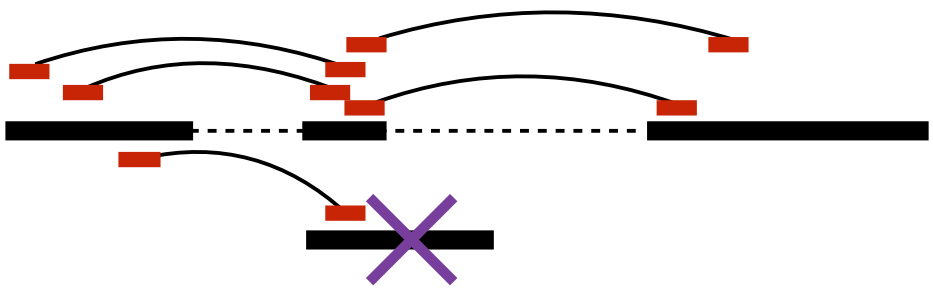
Practical formulations / pipeline

1. **Contig assembly**: assemble the reads into strings (*contigs*) that are *guaranteed* to occur in the genome

Practical formulations / pipeline

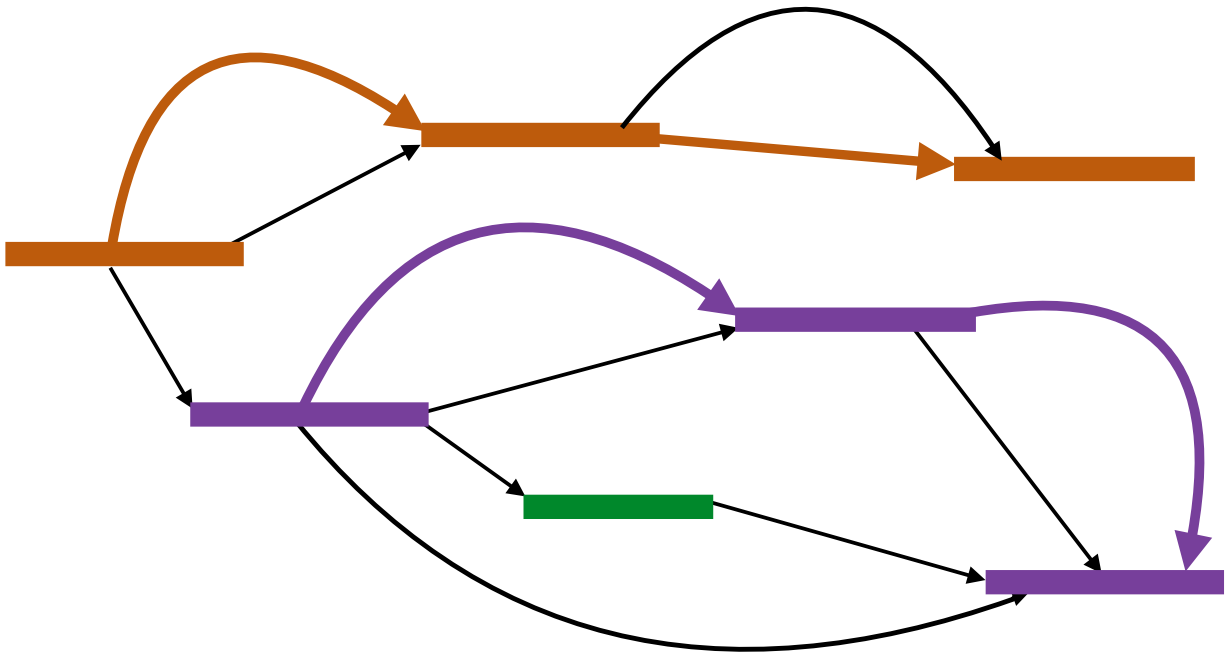
1. **Contig assembly**: assemble the reads into strings (*contigs*) that are *guaranteed* to occur in the genome

2. **Scaffolding**: using paired-end reads, chain the contigs into scaffolds that are *guaranteed* to occur in the genome



Practical formulations / pipeline (2)

3. **Gap filling**: fill the gaps in the scaffolds



Tens of genome assembly programs available: ABySS, Velvet, Allpaths-LG, Bambus2, MSR-CA, SGA, Cortex, SOAPdenovo, Opera-LG, SPADES, ...

De Bruijn graphs

Definition

Given a set R of strings, the de Bruijn graph of order k of R is the directed graph $DB^k(R)$ with

- node set: the set of k-mers of R
- edge set: the set of $k+1$-mers of the strings of R

Also edges occur in the strings of R!

De Bruijn graphs

Definition

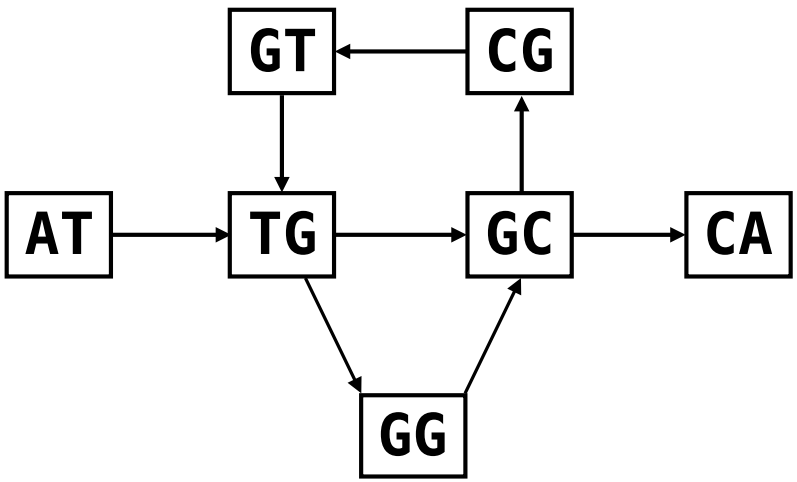
Given a set R of strings, the de Bruijn graph of order k of R is the directed graph $DB^k(R)$ with

- node set: the set of k-mers of R
- edge set: the set of $k+1$-mers of the strings of R

Also edges occur in the strings of R!

ATGCGTGGCA
ATGCG
CGTG
TGGCA

$k = 2$

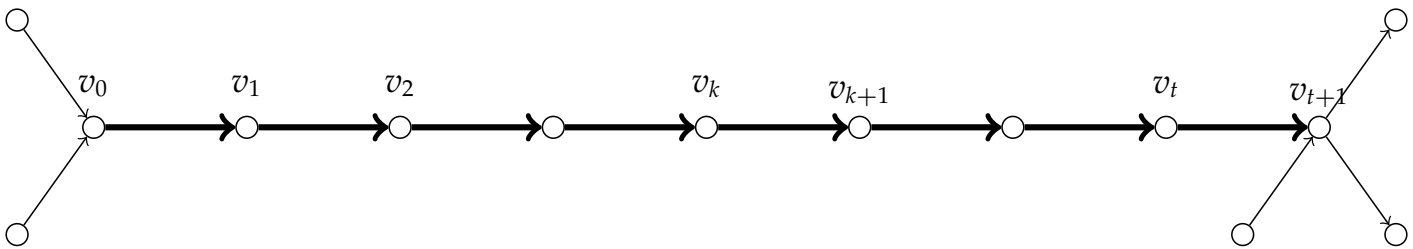


Contig assembly

joint work with Paul Medvedev

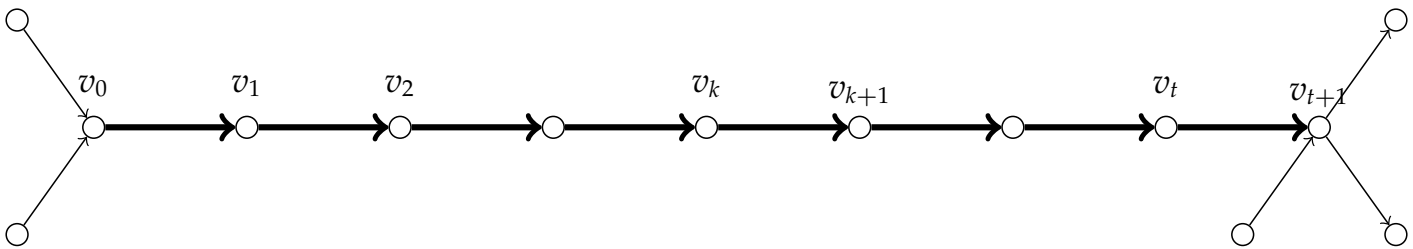
Contig assembly

- No previous formal definition of *contig*
- Usually, contigs are maximal, unary paths (i.e., whose internal nodes have in-degree and out-degree 1, aka unitigs)



Contig assembly

- No previous formal definition of *contig*
- Usually, contigs are maximal, unary paths (i.e., whose internal nodes have in-degree and out-degree 1, aka unitigs)

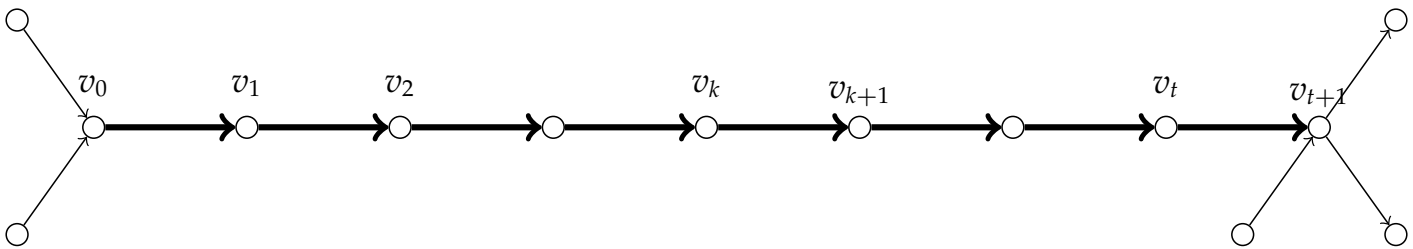


Given a dBG G:

- a genomic walk of G is a circular edge-covering walk of G
- a walk is safe if it is a sub-walk of all genomic walks of G

Contig assembly

- No previous formal definition of *contig*
- Usually, contigs are maximal, unary paths (i.e., whose internal nodes have in-degree and out-degree 1, aka unitigs)



Given a dBG G:

- a genomic walk of G is a circular edge-covering walk of G
- a walk is safe if it is a sub-walk of all genomic walks of G

We now assume that the dBG admits a genomic walk (i.e., is strongly connected) and is not a single cycle.

Contig assembly (2)

We say that a contig assembly algorithm is

- sound: if every output walk is safe
- complete: if every safe walk is in the output

Contig assembly (2)

We say that a contig assembly algorithm is

- sound: if every output walk is safe
- complete: if every safe walk is in the output

The unitig algorithm is:

- outputting all maximal unitigs
- sound
- not complete

Contig assembly (2)

We say that a contig assembly algorithm is

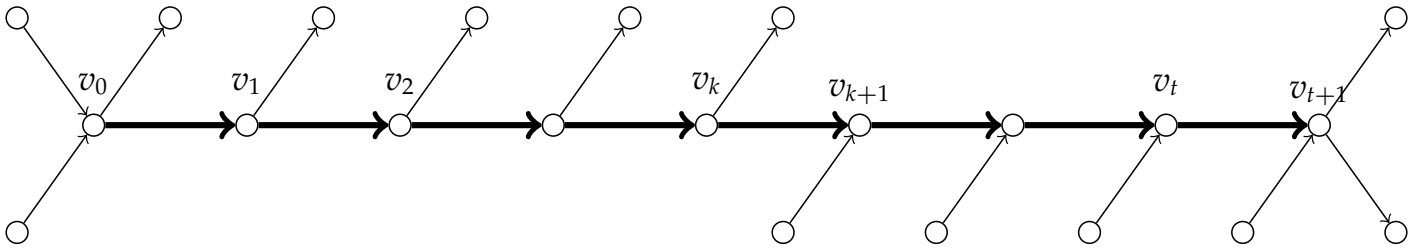
- sound: if every output walk is safe
- complete: if every safe walk is in the output

The unitig algorithm is:

- outputting all maximal unitigs
- sound
- not complete

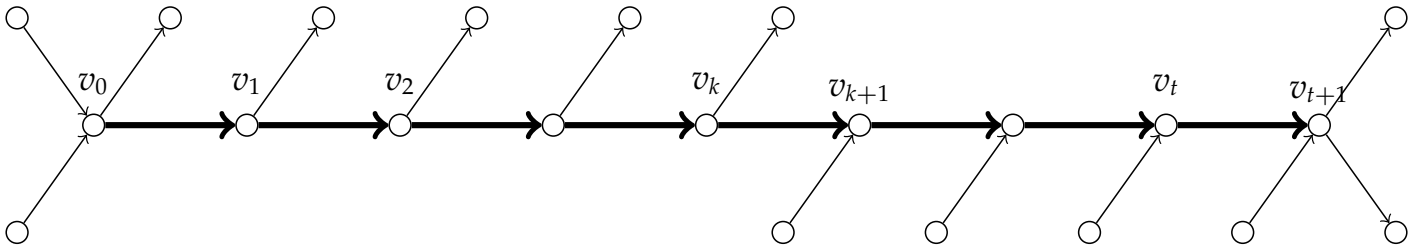
Is there a sound and complete contig assembly algorithm?

Non-switching contigs



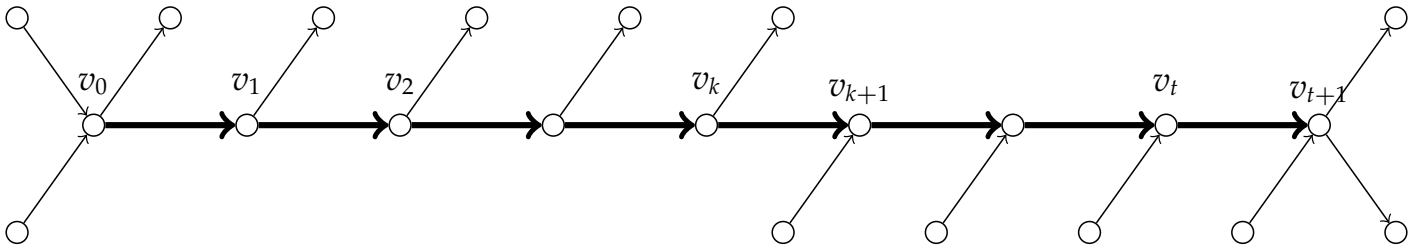
- A path with all out-branching nodes before all in-branching nodes

Non-switching contigs



- A path with all out-branching nodes before all in-branching nodes
- Related to transformation-based algorithms of
  - Kingsford, Schatz, Pop 2010
  - Jackson 2009
  - Medvedev, Georgiou, Myers, Brudno 2007

Non-switching contigs



- A path with all out-branching nodes before all in-branching nodes
- Related to transformation-based algorithms of
  - Kingsford, Schatz, Pop 2010
  - Jackson 2009
  - Medvedev, Georgiou, Myers, Brudno 2007

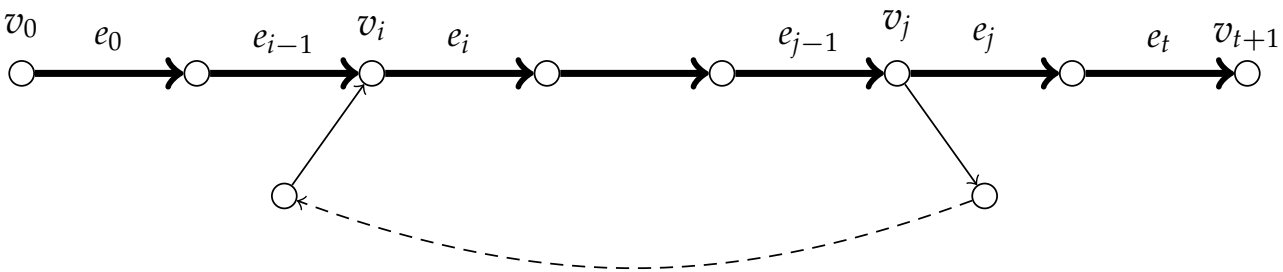
Theorem

There is an $O(|G| + |output|)$-time algorithm to output all maximal non-switching contigs of G.

Theorem

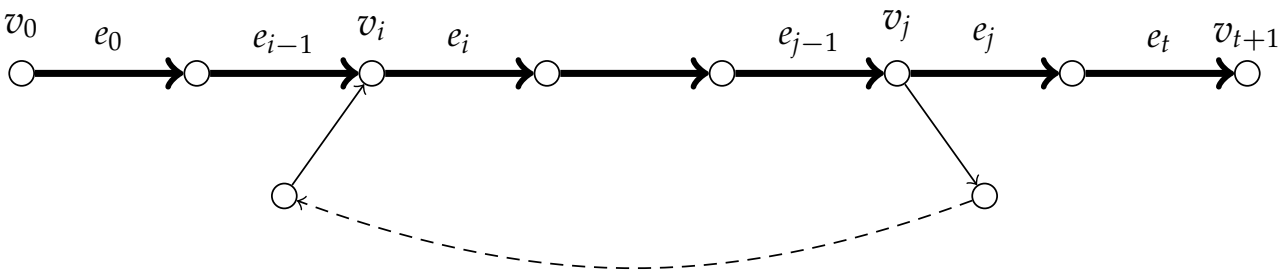
The non-switching contig assembly algorithm is sound but not complete.

*Omni*TIGS



We say that a walk $w = (v_0, e_0, v_1, e_1, \ldots, v_t, e_t, v_{t+1})$ is an omnitig if for all $1 \leq i \leq j \leq t$, there is no proper v_j-v_i path with first edge different from e_j, and last edge different from e_{i-1}.

*Omni*TIGS



We say that a walk $w = (v_0, e_0, v_1, e_1, \ldots, v_t, e_t, v_{t+1})$ is an omnitig if for all $1 \leq i \leq j \leq t$, there is no proper v_j-v_i path with first edge different from e_j, and last edge different from e_{i-1}.

THEOREM

A walk w is safe $\Leftrightarrow$ w is an omnitig.

THEOREM

There is a polynomial time algorithm for outputting all maximal omnitigs.

COROLLARY

The omnitig algorithm is sound and complete.

Experimental results

Algorithm	#strings	E-size	AVG length
unitig	41,524	7,806	893
non-switching contigs	32,589	7,822	1,136
omnitigs	24,949	7,850	1,479

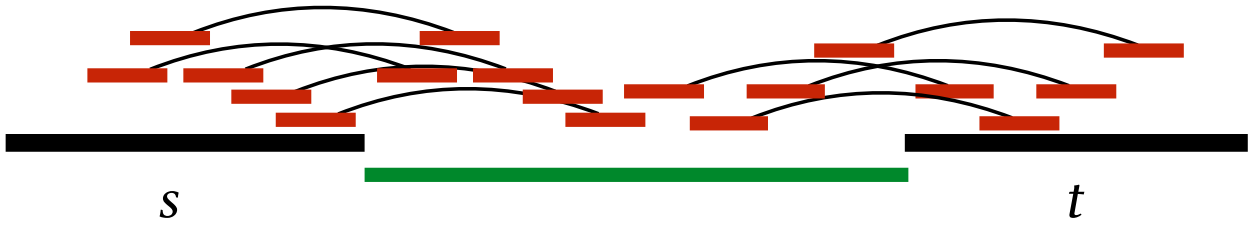
- genome: circularized human chr21 (length $48 \cdot 10^6$)
- graph: dBG^k(chr21) for $k = 55$
- e-size: given a set of substrings of genome, their e-size is the average, over all genomic positions i, of the mean length of the strings spanning position i

GAP FILLING

joint work with Leena Salmela, Kristoffer Sahlin and Veli Mäkinen

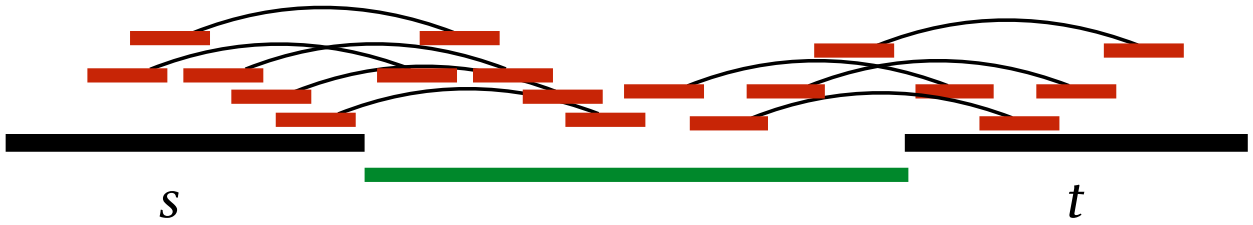
Problem formulation

Previous formulations (GapCloser 2012, GapFiller 2012):



Problem formulation

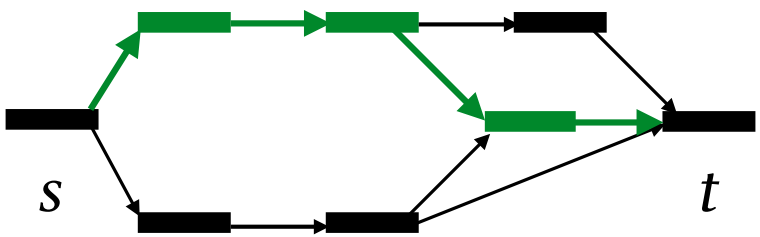
Previous formulations (GapCloser 2012, GapFiller 2012):



We formulate it as Exact Path Length problem. Given:

- $G = dBG^k(R)$, for some k
- $s, t \in V(G)$, the two k-mers flanking the gap
- $[d'..d]$ an estimate on the gap length

For every $x \in [d'..d]$ find an s-t path spelling a string of length x (i.e. a path of length $x - k$).



Dynamic programming (DP)

Usually, $d < |V(G)|$.

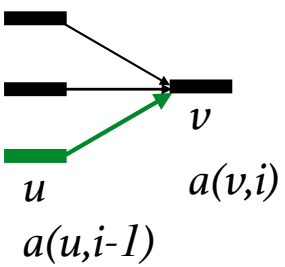
Can be solved by DP in time $O(d \cdot |E(G)|)$:

- for every node v store:

$$a(v,i) := \begin{cases} 1 & \text{if there exists an } s\text{-}v \text{ path of length } i, \\ 0 & \textit{otherwise}. \end{cases}$$

- initialize $a(s,0) = 1$, and compute

$$a(v,i) := \bigvee_{u \in N^-(v)} a(u, i-1).$$

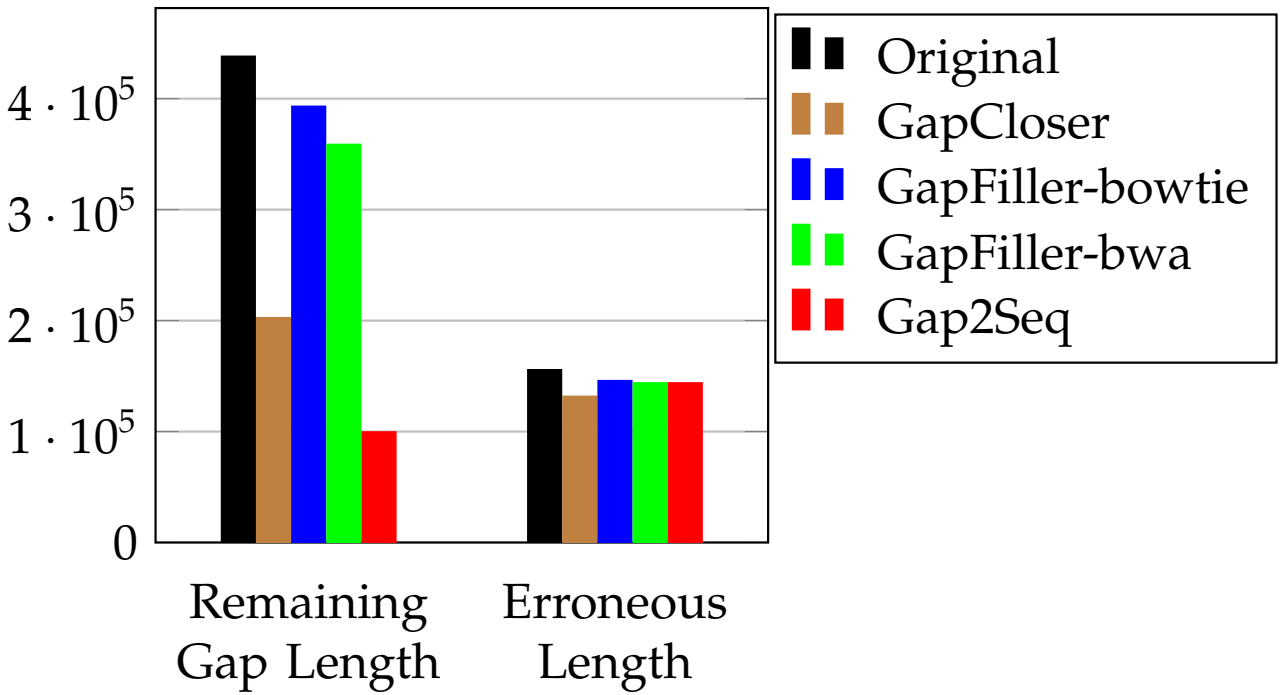


- back-tracking in the DP matrix from $a(t, x-k)$ gives a path (if exists)

Engineered implementation

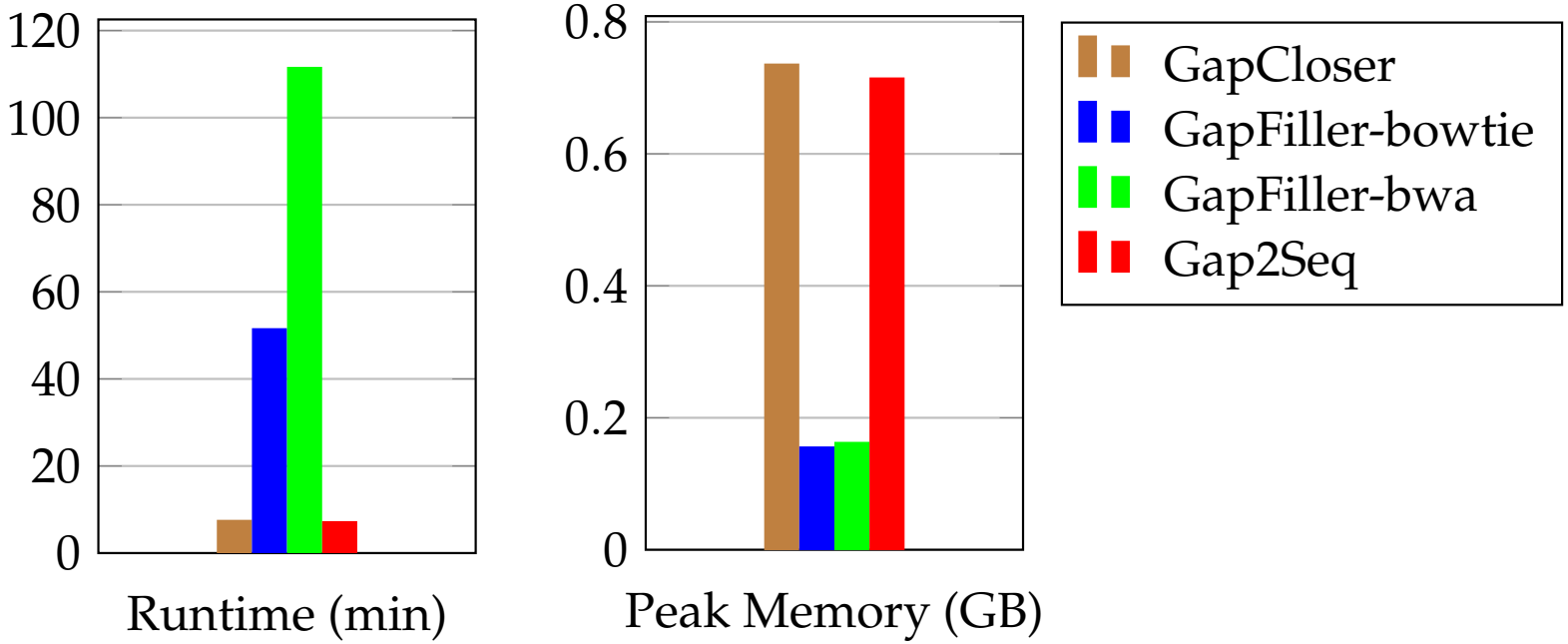
- *k*-mers flanking the gaps can have errors
  - allow paths to start/end at up to *t* flanking *k*-mers
- we should not explore the entire graph!
  - *meet in the middle* optimization
- DP matrix rows are sparse
  - store only the non-zero entries in each row
- parallelization on scaffold level
- limit the memory usage of the DP matrix
  - abandon the search on a gap if limit exceeded

Experimental results (*Gap2Seq*)



- genome: Staphylococcus aureus (length $2.8 \cdot 10^6$)
- graph: dBG with $k = 31$, for $R =$ a collection of real reads
- results are totals over all scaffolds from a benchmark dataset (ABySS, Allpaths-LG, Bambus2, CABOG, MSR-CA, SGA, SOAPdenovo, Velvet)

EXPERIMENTAL RESULTS (*Gap2Seq*)



Conclusions

Potential uses for omnitigs:

- longer contigs = better starting point for scaffolding and gap filling
- more flanking information around loci of interest

Directions for omnitigs:

- robustness to errors, coverage gaps, reverse complements?
- faster omnitig algorithm
- a sound and complete algorithm when the genomic walk is linear?

Gap Filling:

- performance on human data is 'so and so' (but all tools have problems)
- we need to improve the runtime and memory usage
- we need a way to choose between multiple solutions (contig assembly?)

MULŢUMESC