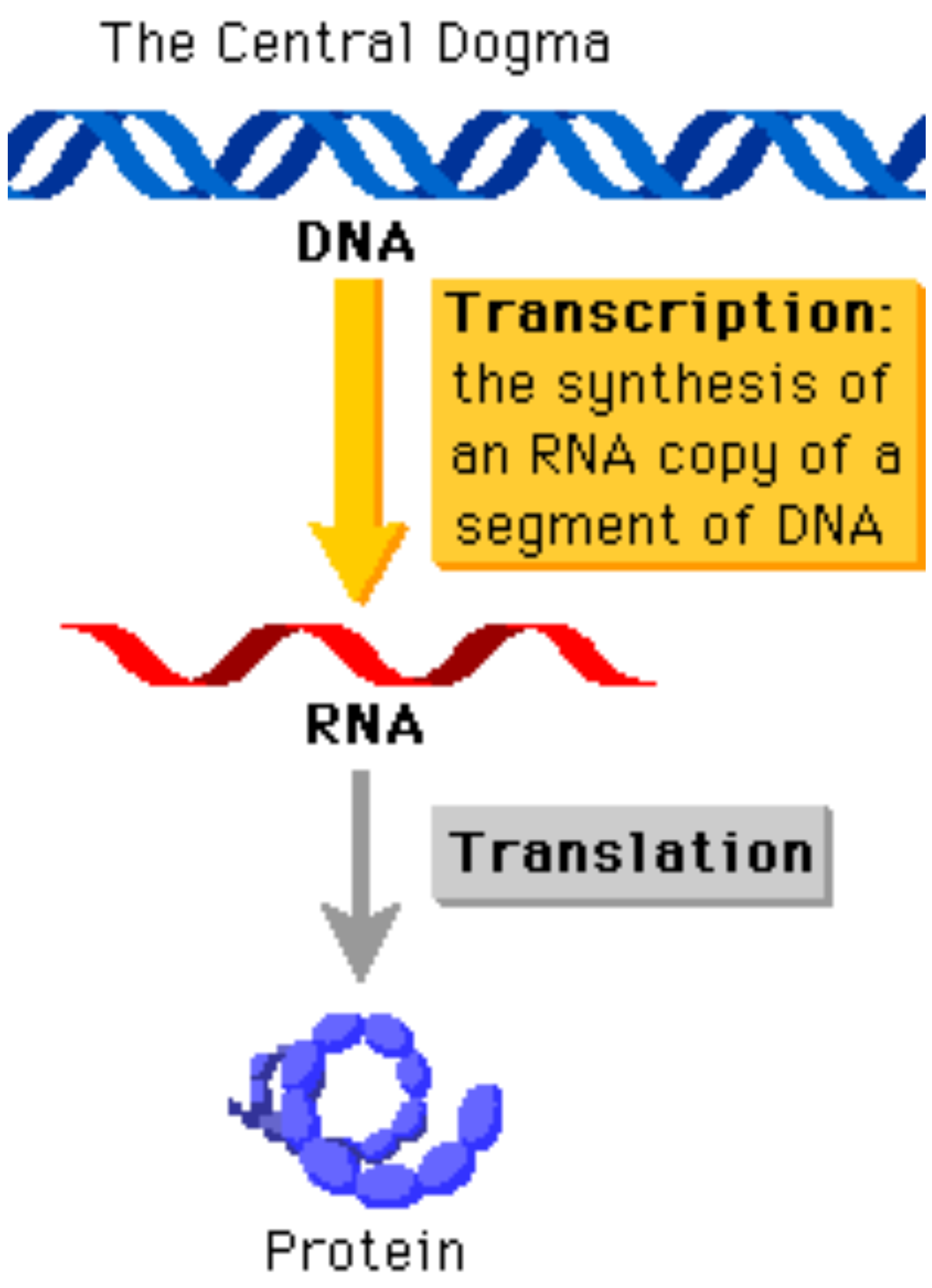
Enumeration problems in RNA-seq data

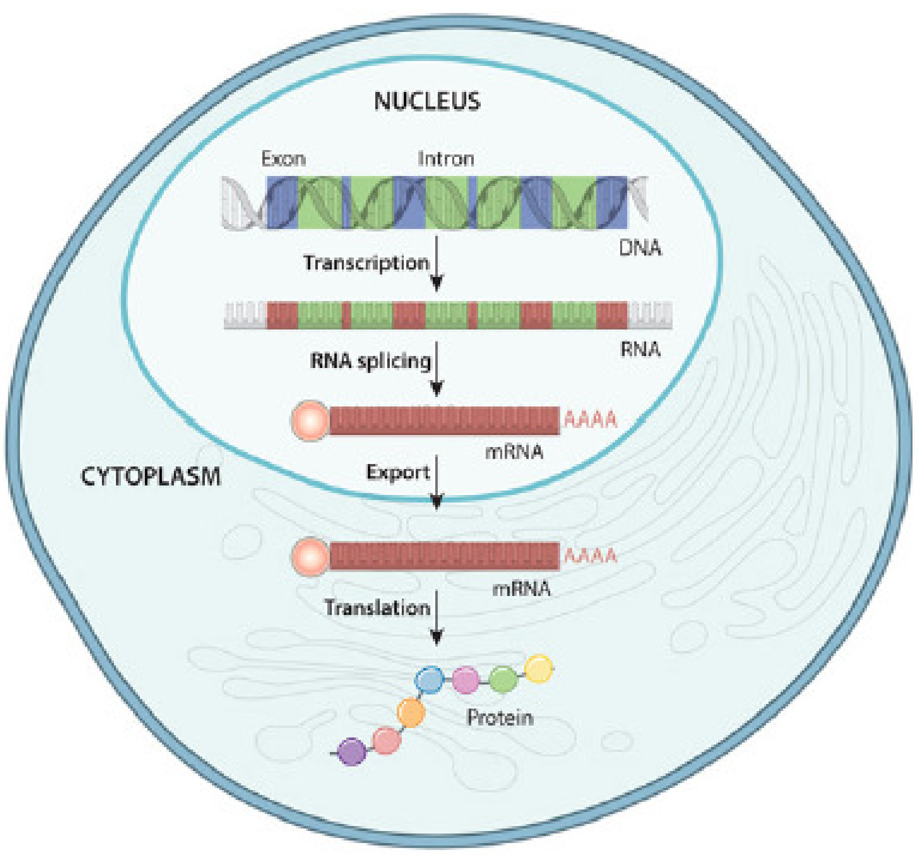
blerina sinaimeri

The central dogma of molecular biology

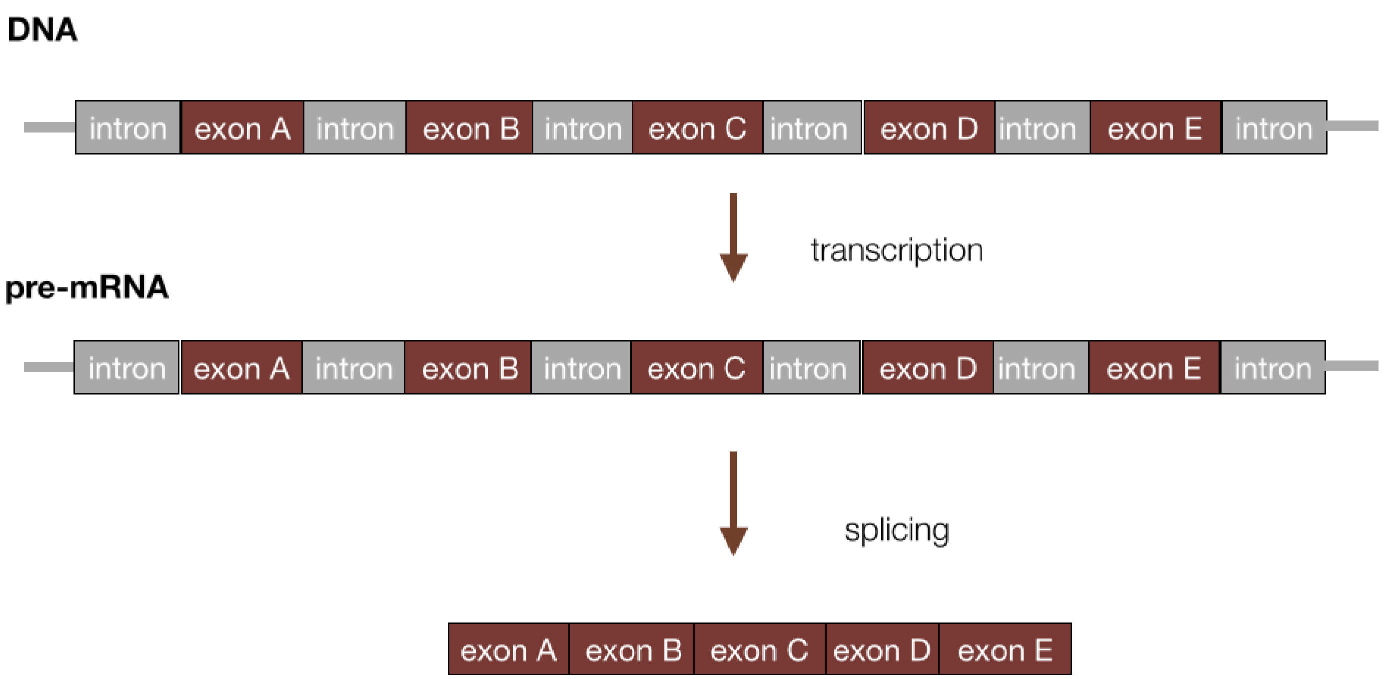


From DNA to RNA to proteins in eucaryotes

From DNA to proteins in eucaryotes



RNA-splicing in eucaryotes



Modelling and assembling NGS data (III)
de Bruijn graph

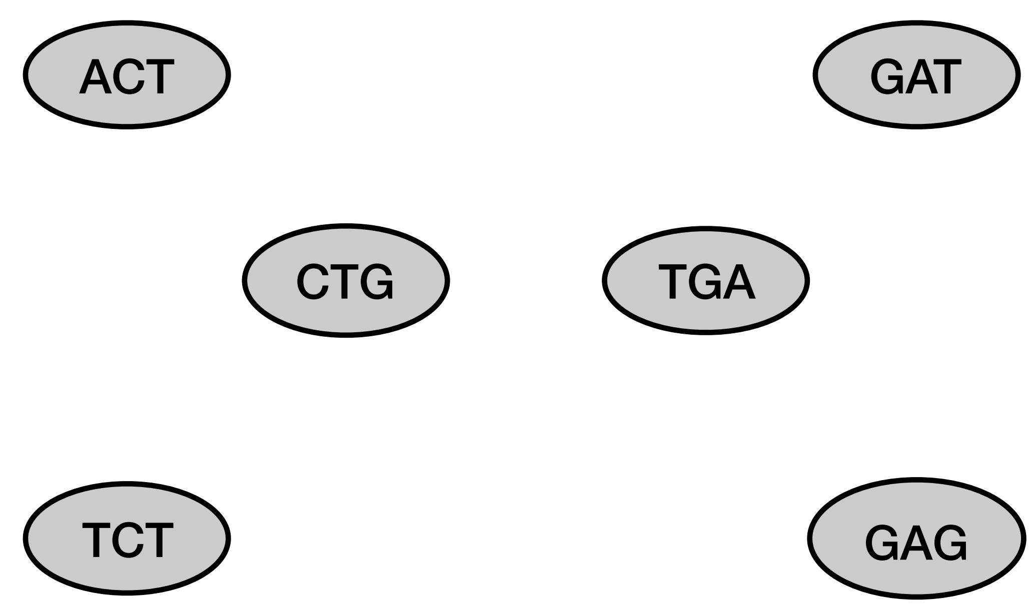
De Bruijn graph

Given a set of reads *R* and an integer *k* we define the de Bruijn graph *B(R,k)*

- Vertices are substrings of length **k** (**k**-mers)
- Arcs are *k-1* suffix-prefix overlaps that appear as a substring in ***R*.**

Example

R={ACTGAT,TCTGAG}, k=3



de Bruijn graph

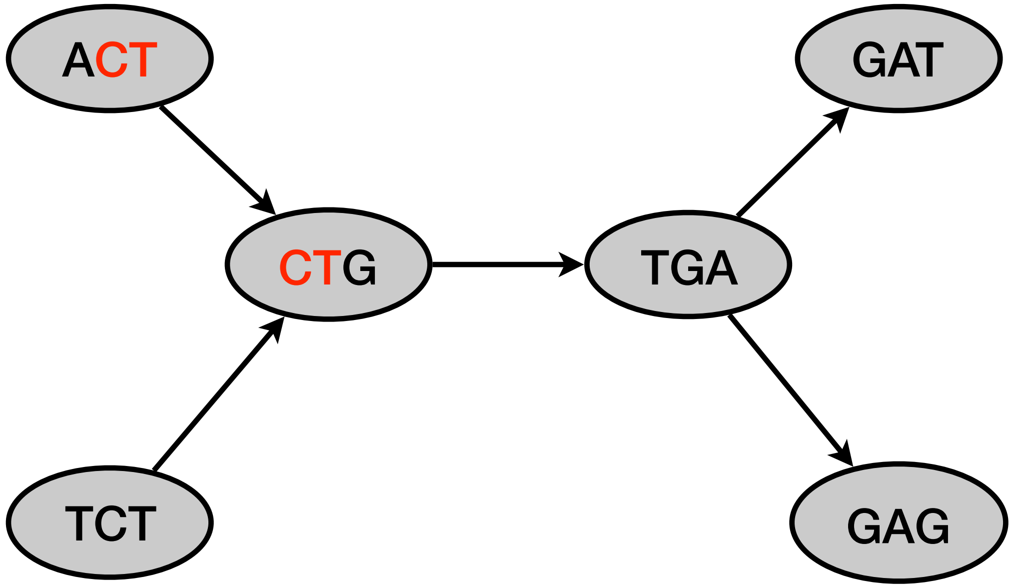
De Bruijn graph

Given a set of reads *R* and an integer *k* we define the de Bruijn graph *B(R,k)*

- Vertices are substrings of length **k** (**k**-mers)
- Arcs are *k-1* suffix-prefix overlaps that appear as a substring in *R*.

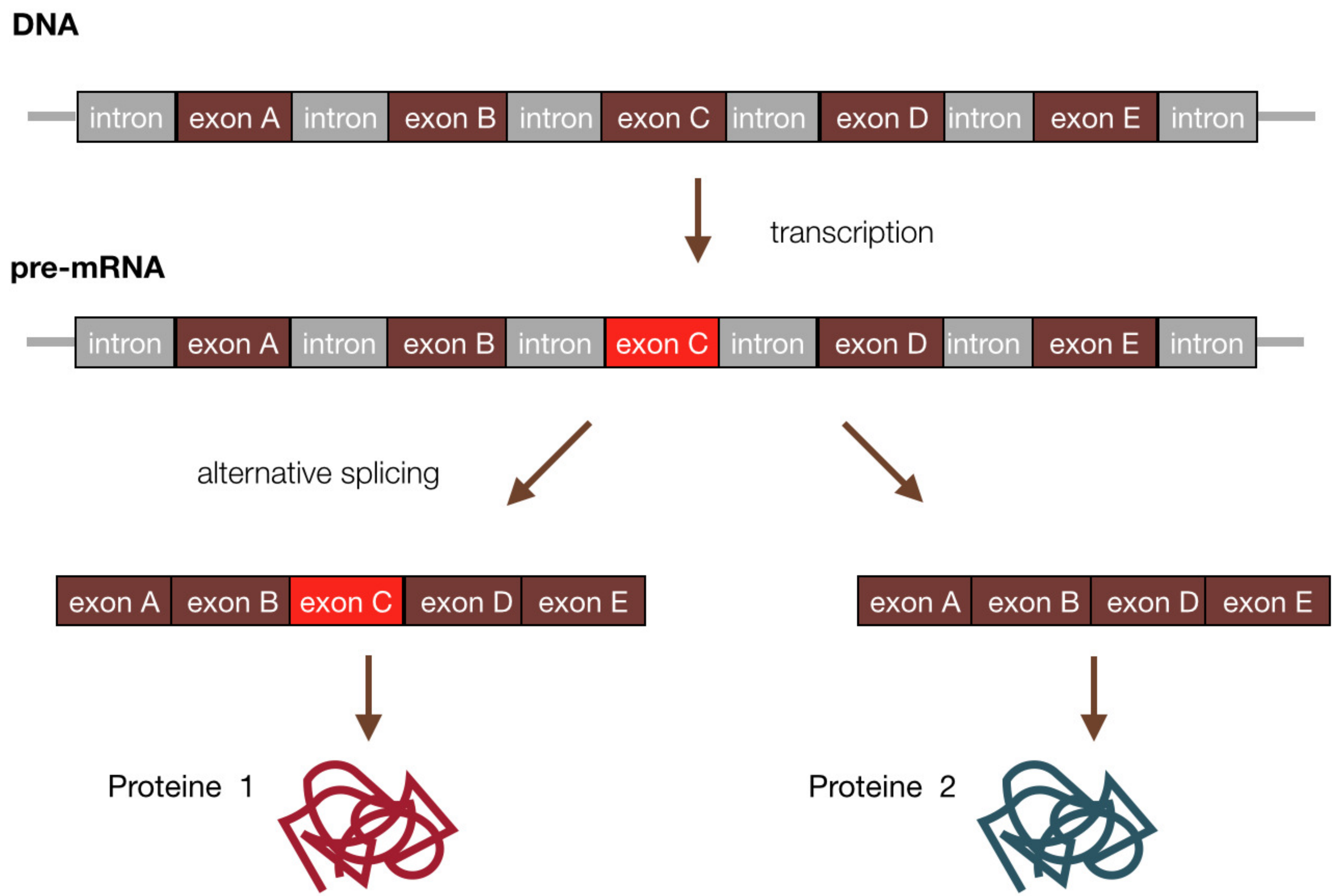
Example

R={ACTGAT,TCTGAG}, k=3



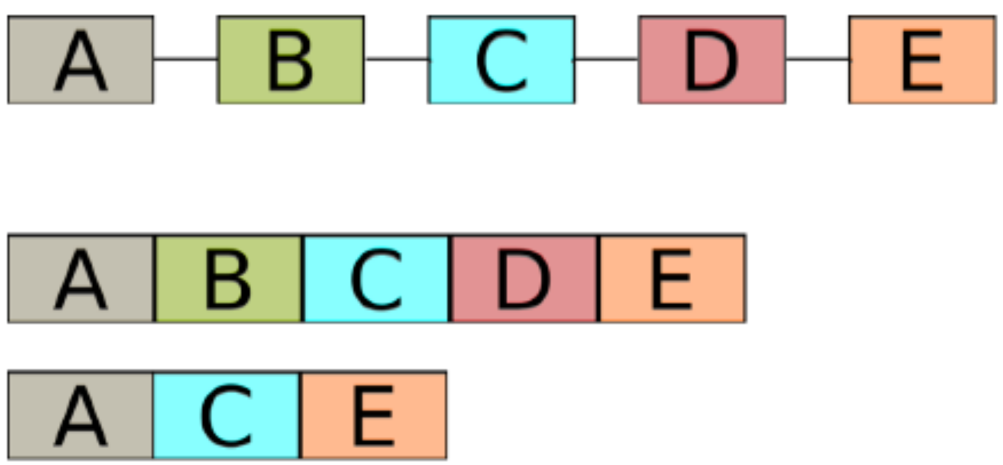
Local assembly

Alternative splicing (AS) in RNA

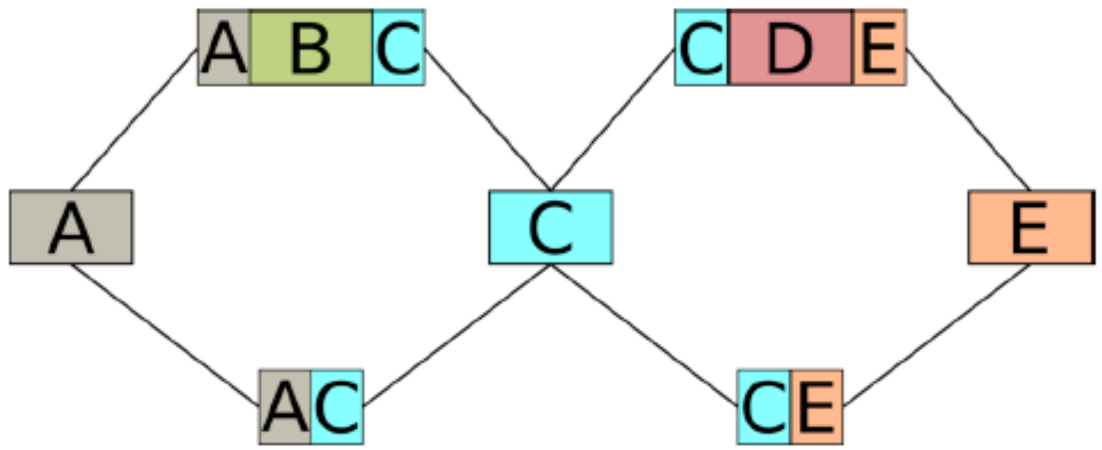


Assembly: Global vs Local

A gene with 2 alternative transcripts



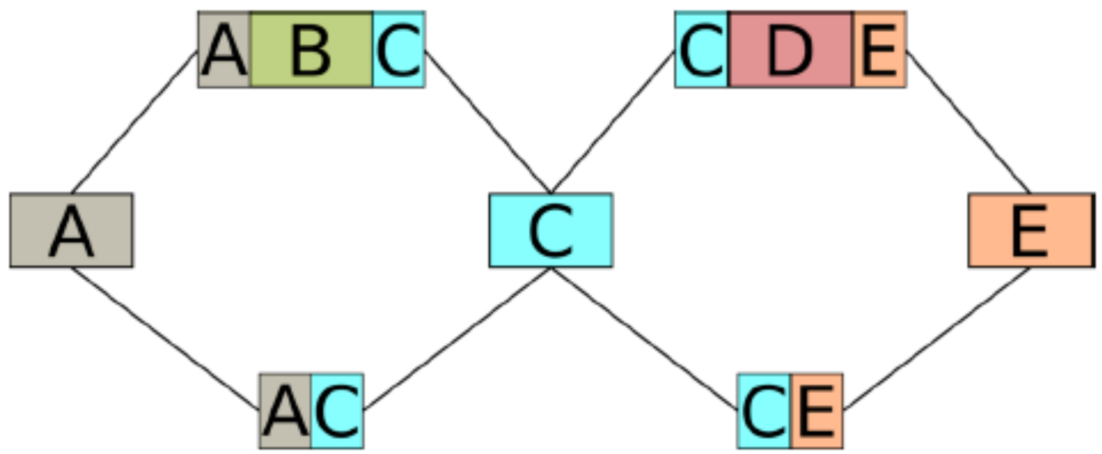
The corresponding de Bruijn graph



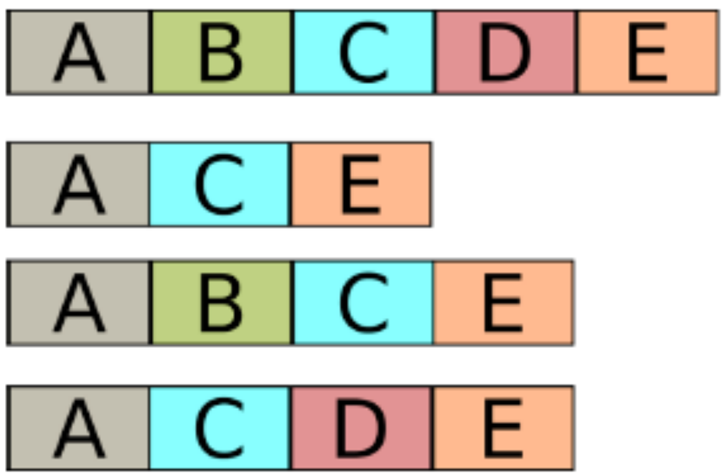
A global assembler will search for maximal walks in the graph.

Assembly: Global vs Local

The corresponding De Bruijn graph

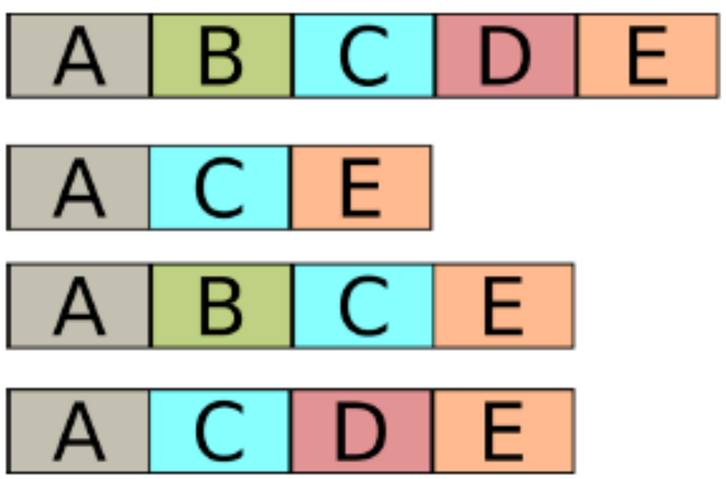


4 possible walks corresponding to:

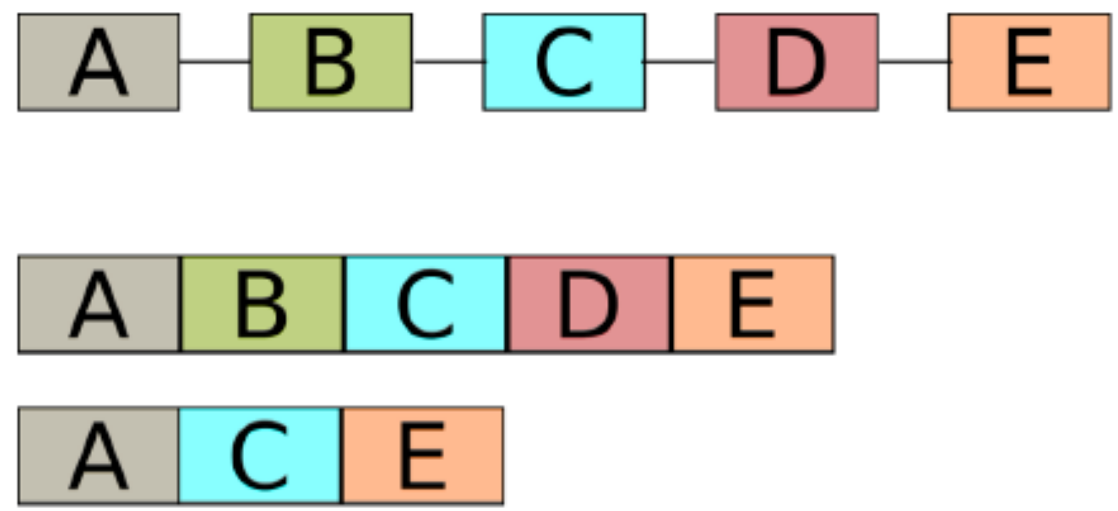


Assembly: Global vs Local

4 possible walks corresponding to:



But only 2 alternative transcripts

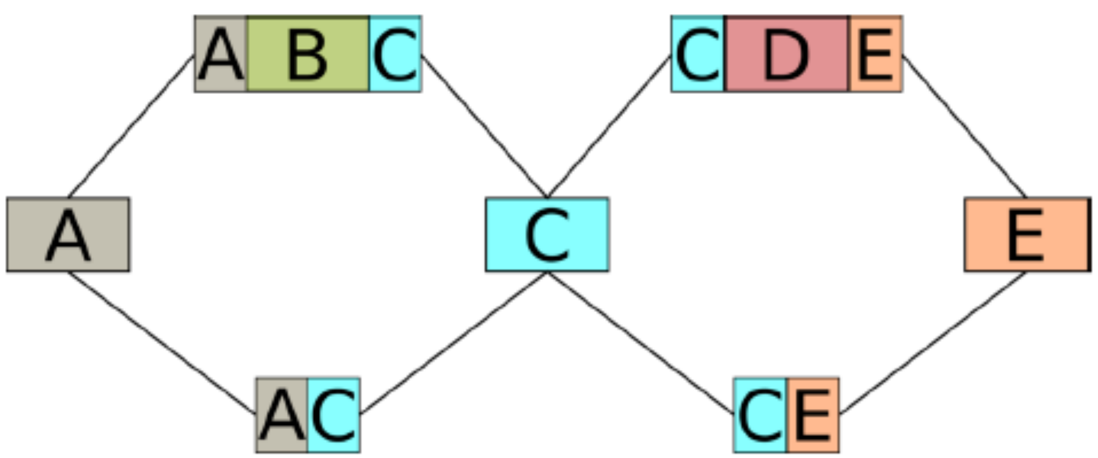


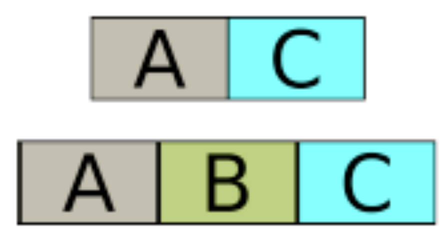
Every transcript corresponds to a walk but not every walk to a transcript. Global assemblers have to choose the “right” walk.

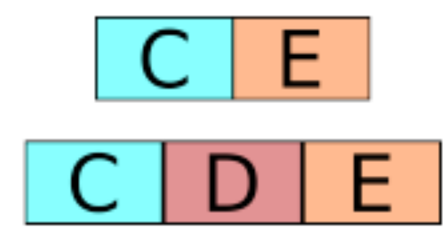
Local assembly

Main idea: To find an AS event consider only the region of

the graph “near” the skipped part (cycle-like pattern)







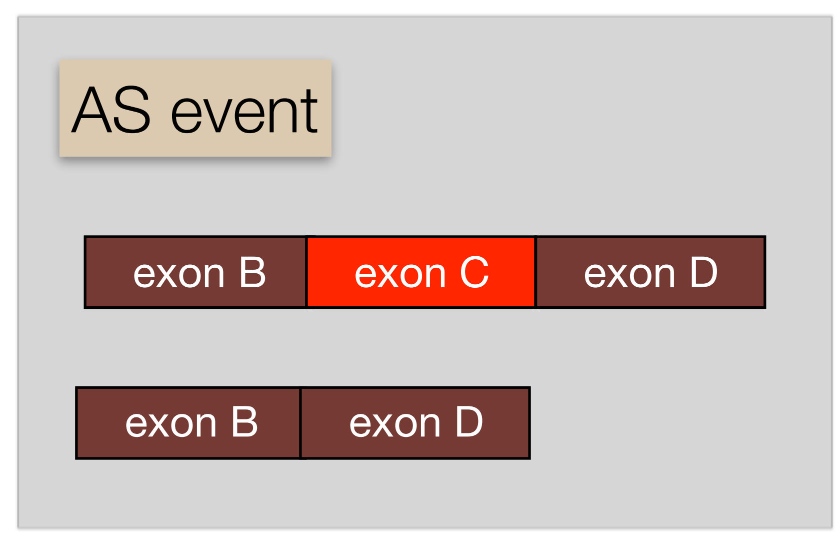
The problem

Our goal

Identify in RNA-seq data alternative splicing events, **without a reference genome.** We will only **locally** assemble them.

Input: A set of reads **R**

Output: The set of AS events

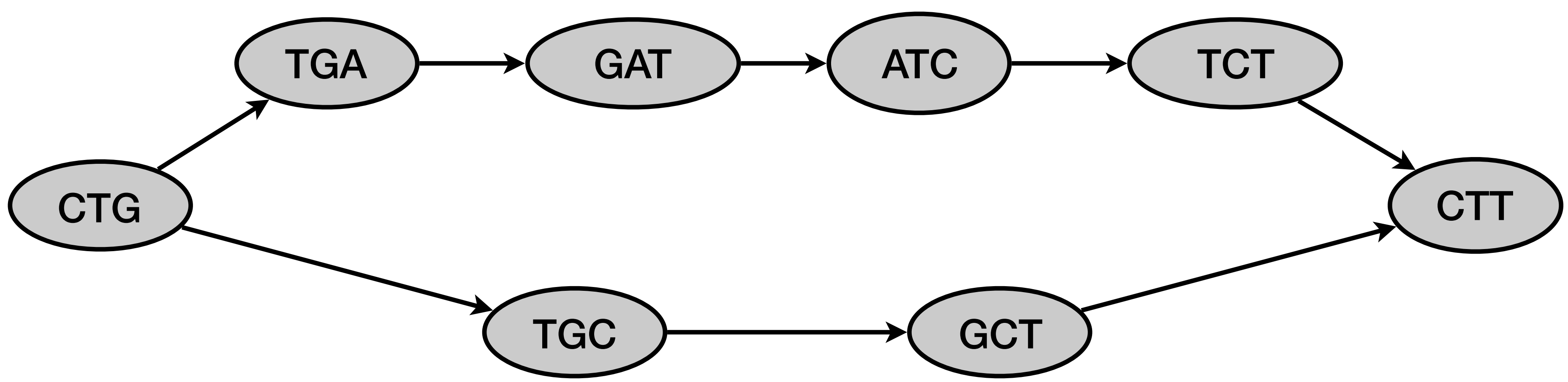


AS Events in de Bruijn graph

AS events will correspond to sequences, **awb, ab**.
What will these correspond in the de Bruijn graph?

Example

ab=CTGCTT awb=CTGATCTT

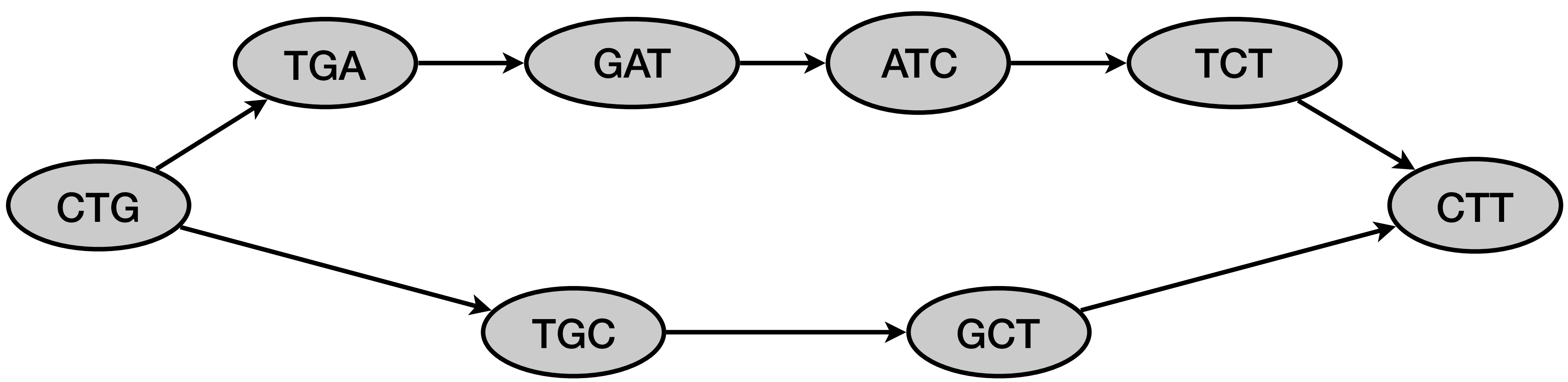


The strings **awb** and **ab** will correspond to a **bubble,** i.e. a pair of internally vertex-disjoint paths, in the de Bruijn graph.

AS Events in de Bruijn graph

Example

ab=CTGCTT   awb=CTGATCTT

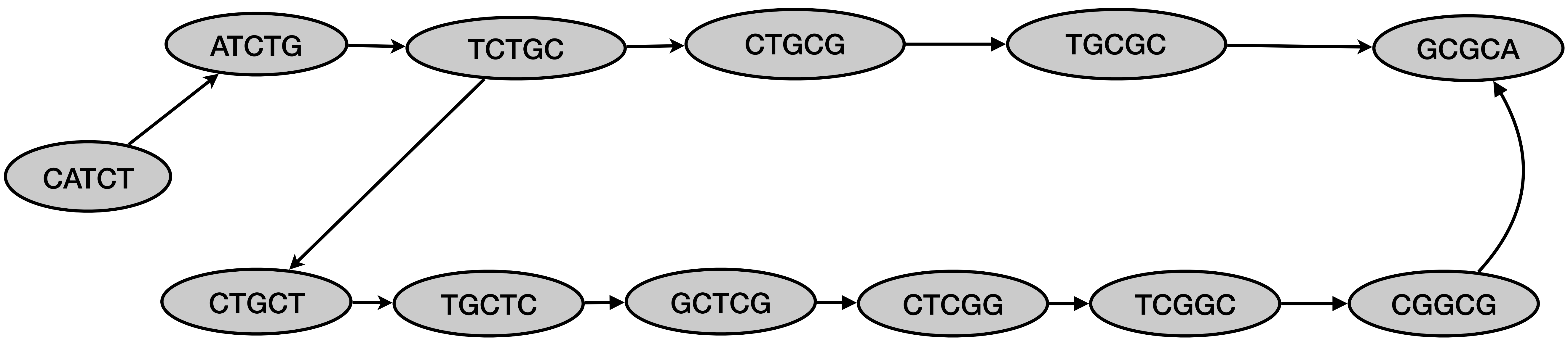


$|a| \geqq k$, $|b| \geqq k$. What characteristics has a bubble generated by an AS event?

AS events in de Bruijn graph

Example

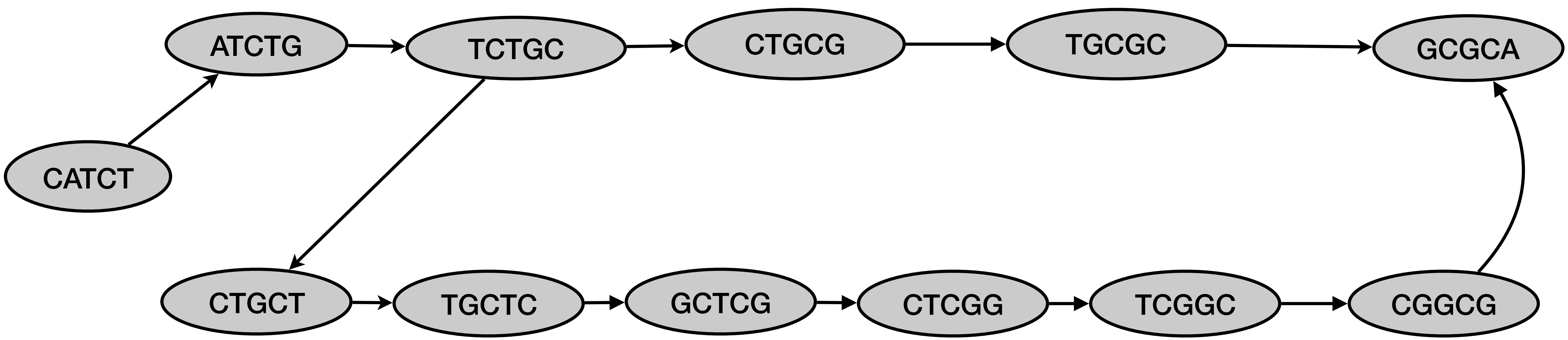
ab=CATCTGCGCA   awb=CATCTGCTCGGCGCA



AS events in de Bruijn graph

Example

ab=CATCTGCGCA awb=CATCTGCTCGGCGCA



The shortest path has length <k-1 (vertices) as **w** and **b** share a prefix.

AS Events in de Bruijn graph

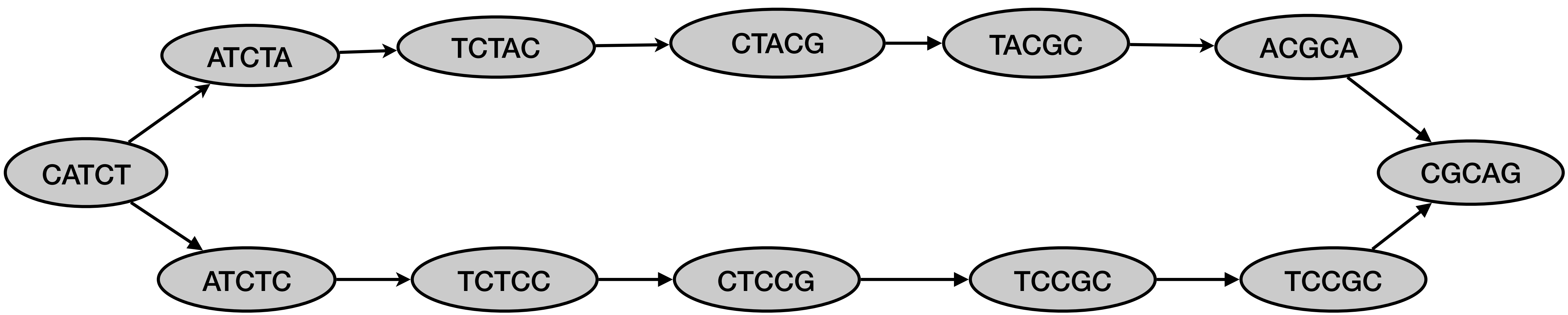
Question:

- What is the length of the shorter path for a bubble generated by the pattern **awb** and **ab**?

SNPs events in de Bruijn graph

Example

x=CATCTACGCAG   y=CATCTCCGCAG



Two paths of the same length k (vertices).

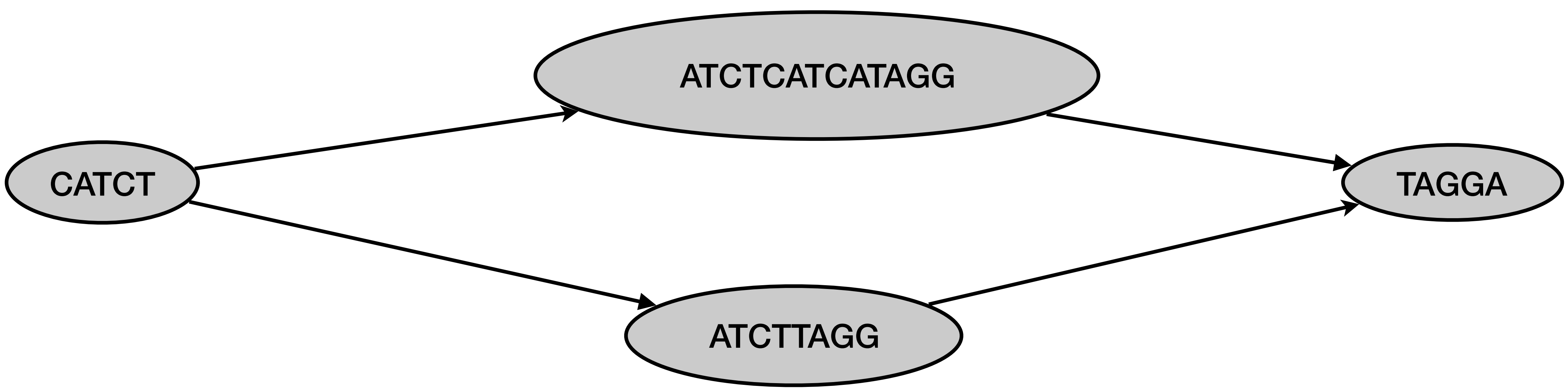
Approximate repeats in de Bruijn graph

Example

Inexact repeats may generate bubbles with a similar path length as bubbles generated by AS events.

x=CATCTTAGGA y=CATCTCATCATAGGA

CATCTCATCA is an inexact repeat.



This can be easily identified: the longer path contains an inexact repeat. it is sufficient to compare the shorter path with one of the ends of the longer path.

AS Events in de Bruijn graph

Example

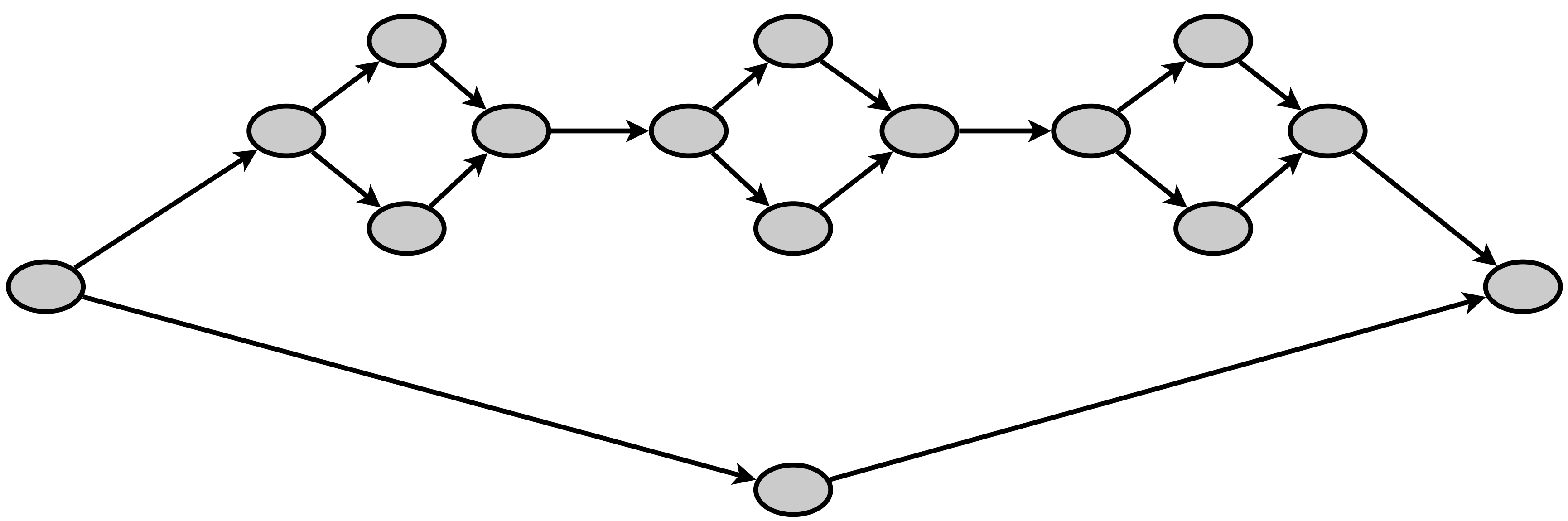
- Every AS event generates a bubble.
- Not every bubble with a shorter path with at most k-1 vertices correspond to an AS event.
    - Repeat-associated bubbles: “similar” paths (small edit distance)

Listing all the bubbles

The problem

Given R, k list all the bubbles in the de Bruijn graph **B(R,k)**

- The number of bubbles can be exponential in the size of the graph.



- A good algorithm: polynomial delay (polynomial time between two outputs).

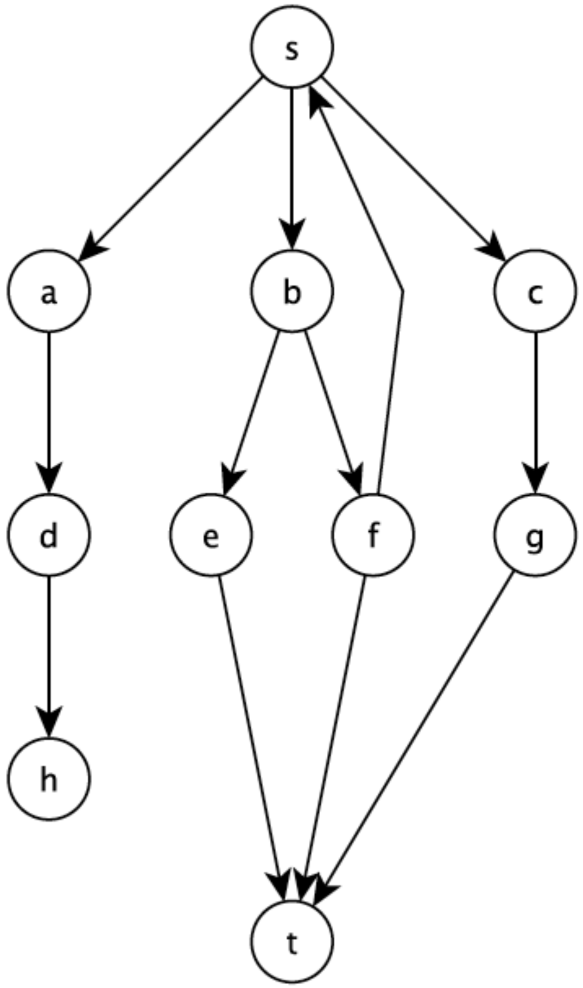
Listing (s,t)-paths

Listing all (s,t)-paths

The problem

Given a directed graph **G** list all the (s,t)-paths in **G**.

Idea: Partition the set of solutions



The set of paths $s \rightsquigarrow t$ in G can be partitioned in:

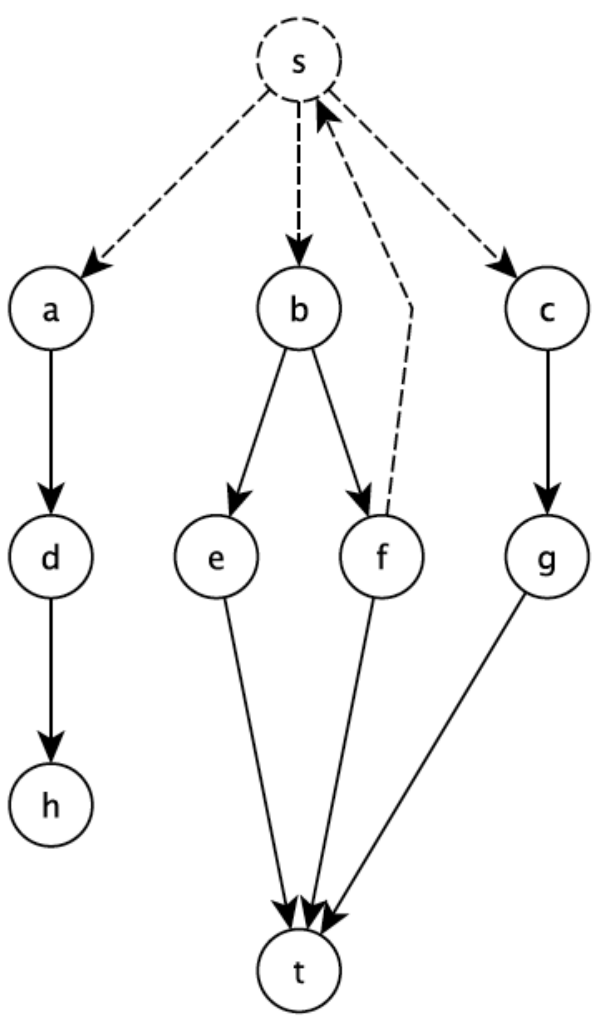
- paths that use (s, a);
- paths that use (s, b);
- paths that use (s, c).

Listing all (s,t)-paths

The problem

Given a directed graph **G** list all the (s,t)-paths in **G**.

Idea: **Recursively** partition the set of solutions



The set of paths $s \rightsquigarrow t$ in G can be partitioned in:

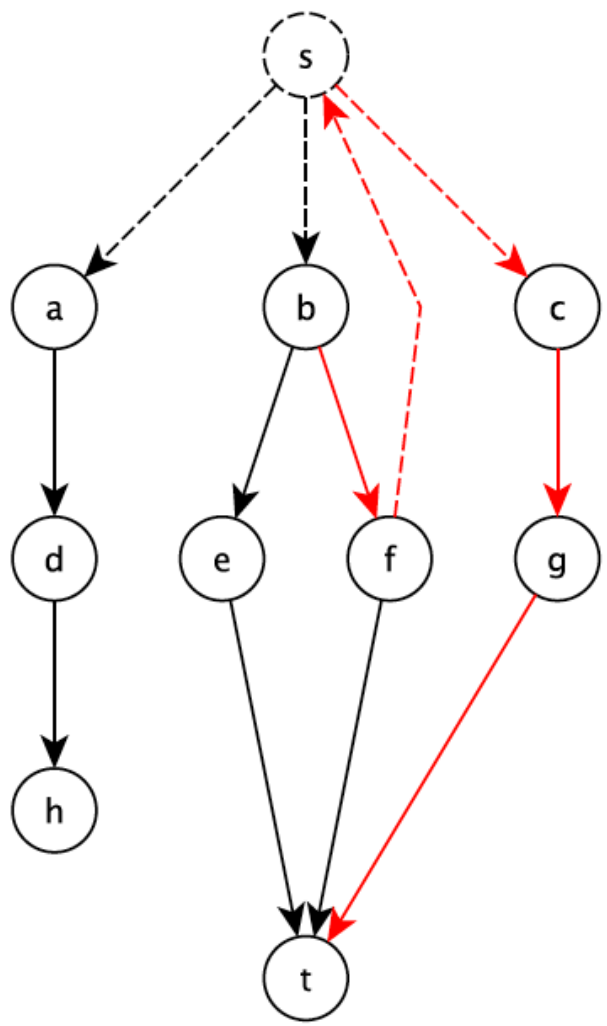
- (s, a) plus $a \rightsquigarrow t$ in $G - s$;
- (s, b) plus $b \rightsquigarrow t$ in $G - s$;
- (s, c) plus $c \rightsquigarrow t$ in $G - s$.

Listing all (s,t)-paths

The problem

Given a directed graph **G** list all the (s,t)-paths in **G**.

Idea: **Recursively** partition the set of solutions



The set of paths $s \rightsquigarrow t$ in G can be partitioned in:

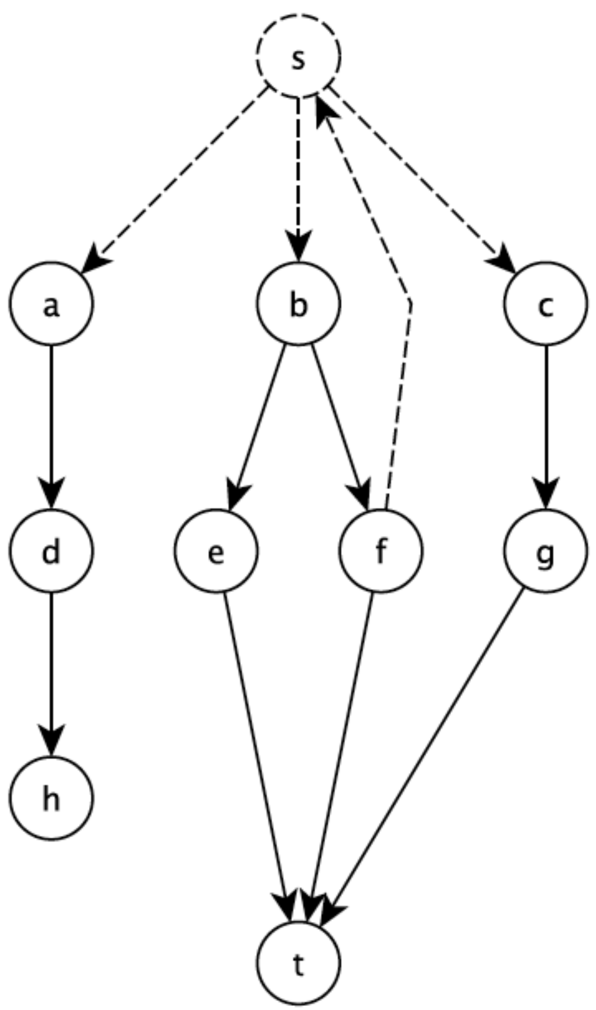
- (s, a) plus $a \rightsquigarrow t$ in $G - s$;
- (s, b) plus $b \rightsquigarrow t$ in $G - s$;
- (s, c) plus $c \rightsquigarrow t$ in $G - s$.

Listing all (s,t)-paths

The problem

Given a directed graph **G** list all the (s,t)-paths in **G**.

Idea: Explore only non-empty partitions



- There is no $s \rightsquigarrow t$ path using (s, a).
- Before exploring a partition, test if it contains at least one solution.

Listing all (s,t)-paths

The algorithm

Algorithm 1.2: $st\textsc{Paths}(G, s, t, \pi)$

Input: An undirected graph G, vertices s and t, and a path π (initially empty).
Output: The paths from s to t in G.

if $s = t$ **then**
  output S
  return

choose an edge $e = (s, v)$
if *there is a vt-path in* $G - s$ **then**
  $st\textsc{Paths}(G - s, v, t, \pi(s, v))$

if *there is a st-path in* $G - e$ **then**
  $st\textsc{Paths}(G - e, s, t, \pi)$

Listing all (s,t)-paths

The algorithm

Algorithm 1.2: $st\textsc{Paths}(G, s, t, \pi)$

Input: An undirected graph G, vertices s and t, and a path π (initially empty).
Output: The paths from s to t in G.

if $s = t$ **then**
    output S
    return
choose an edge $e = (s, v)$
if *there is a vt-path in* $G - s$ **then**
    $st\textsc{Paths}(G - s, v, t, \pi(s, v))$
if *there is a st-path in* $G - e$ **then**
    $st\textsc{Paths}(G - e, s, t, \pi)$

$O(|V| + |E|)$ using DFS

Listing all (s,t)-paths

The algorithm

Algorithm 1.2: stPATHS(G, s, t, π)

Input: An undirected graph G, vertices s and t, and a path π (initially empty).
Output: The paths from s to t in G.

```
if s = t then
    output S
    return
choose an edge e = (s, v)
if there is a vt-path in G − s then
    stPaths(G − s, v, t, π(s, v))
if there is a st-path in G − e then
    stPaths(G − e, s, t, π)
```

$O(|V| + |E|)$ using DFS

Delay: $O((|V| + |E|)^2)$.

Listing bubbles

Listing bubbles

Definition

(s,t,a_1,a_2)-bubble is a pair of vertex disjoint st-paths with lengths bounded by a_1, a_2.

What if we require a lower bound on the length of the paths?

Listing bubbles

- Two paths $p_1 = s_1 \rightsquigarrow t_1$ and $p_2 = s_2 \rightsquigarrow t_2$ are called compatible if $t_1 = t_2$ and they respect the upper bounds on the lengths.

- Let $\mathcal{P}_{\alpha_1,\alpha_2}(s_1, s_2, G)$ be the set of be the set of all pairs of compatible paths for s_1 and s_2

- $$\mathcal{P}_{\alpha_1,\alpha_2}(s_1, s_2, G) = \mathcal{P}_{\alpha_1,\alpha_2}(s_1, s_2, G') \bigcup_{v \in \delta^+(s_2)} (s_2, v)\mathcal{P}_{\alpha_1,\alpha_2'}(s_1, v, G - s_2)$$

$\alpha_2' = \alpha_2 - w(s_2, v)$ $\quad G' = G - \{(s_2, v) | v \in \delta^+(s_2)\}$

Listing all (s,*)-bubbles

The algorithm

Algorithm 1: $\texttt{enumerate\_bubbles}(s_1, \alpha_1, s_2, \alpha_2, B, G)$

if $s_1 = s_2$ **then**
    if $B \neq \emptyset$ **then**
        output(B)
        return
    else if *there is no* $(s, t, \alpha_1, \alpha_2)$*-bubble, where* $s = s_1 = s_2$ **then**
        return
    end
end
choose $u \in \{s_1, s_2\}$, such that $\delta^+(u) \neq \emptyset$
for $v \in \delta^+(u)$ **do**
    if *there is a pair of compatible paths using* (u, v) *in* G **then**
        if $u = s_1$ **then**
            $\texttt{enumerate\_bubbles}(v, \alpha_1 - w(s_1, v), s_2, \alpha_2, B \cup (s_1, v), G - s_1)$
        else
            $\texttt{enumerate\_bubbles}(s_1, \alpha_1, v, \alpha_2 - w(s_2, v), B \cup (s_2, v), G - s_2)$
        end
    end
end
if *there is a pair of compatible paths in* $G - \{(u, v) | v \in \delta^+(u)\}$ **then**
    $\texttt{enumerate\_bubbles}(v, \alpha_1, s_2, \alpha_2, B, G - \{(u, v) | v \in \delta^+(u)\})$
end

Listing all (s,*)-bubbles

Lemma 1. *There exists a pair of compatible paths for* $s_1 \neq s_2$ *in* G *if and only if there exists* t *such that* $d(s_1, t) \leq \alpha_1$ *and* $d(s_2, t) \leq \alpha_2$.

Lemma 2. *The test of line 5 can be performed in* $O(n(m + n \log n))$.

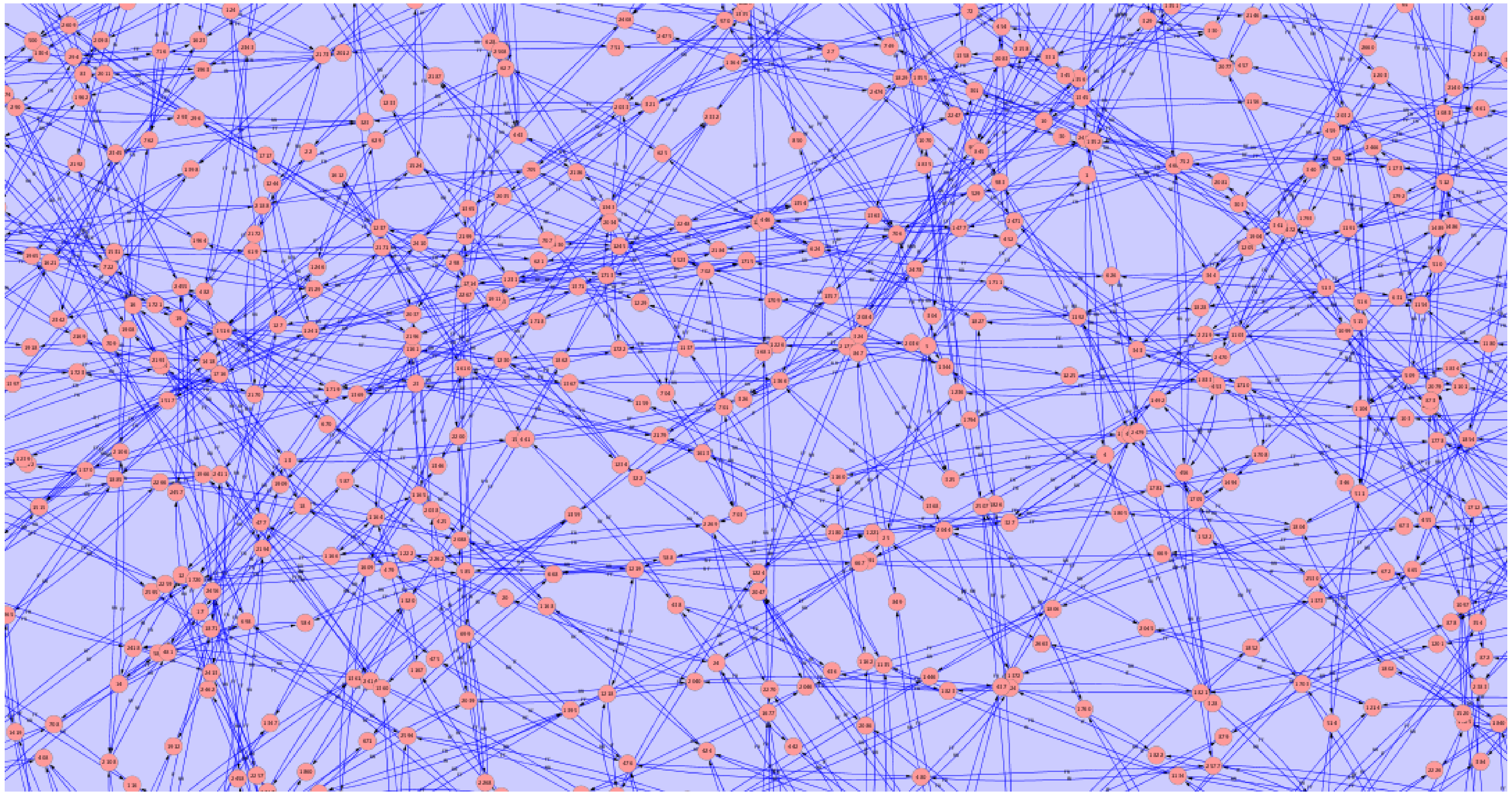
Lemma 3. *The test of line 11, for all* $v \in \delta^+(u)$, *can be performed in* $O(m + n \log n)$ *total time.*

Theorem 1. *Algorithm 1 has* $O(n(m + n \log n))$ *delay.*

everything solved?

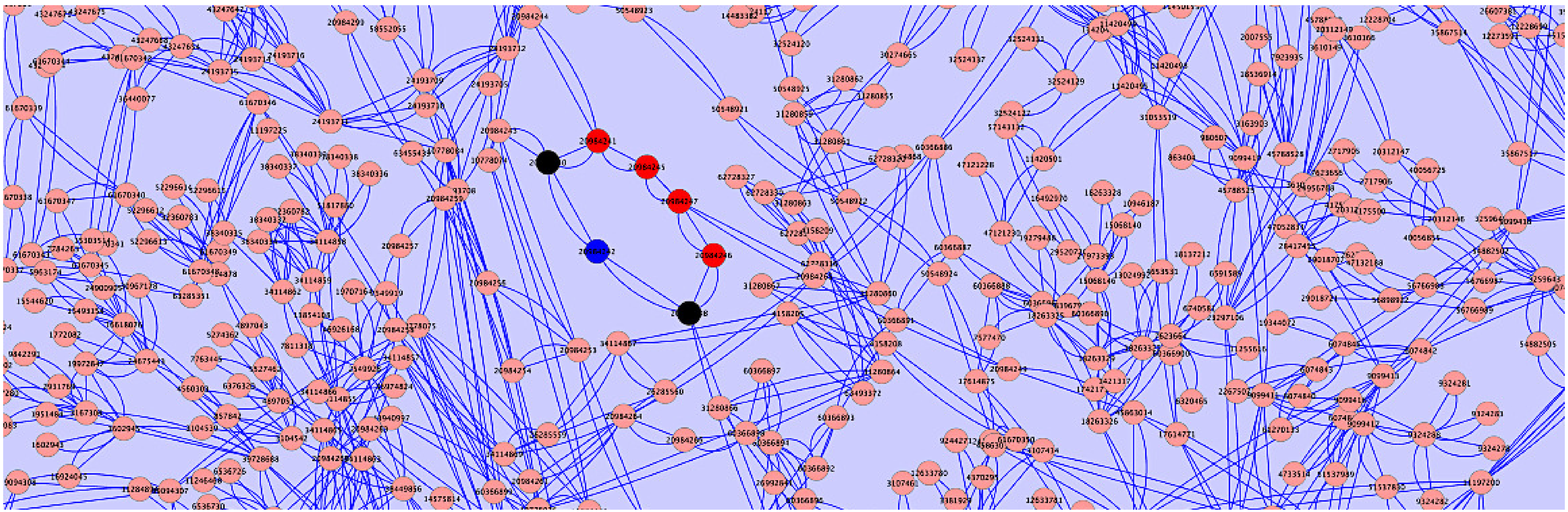
Listing all the bubbles: Problems

De Bruijn graph: snapshot



Listing all the bubbles: Problems

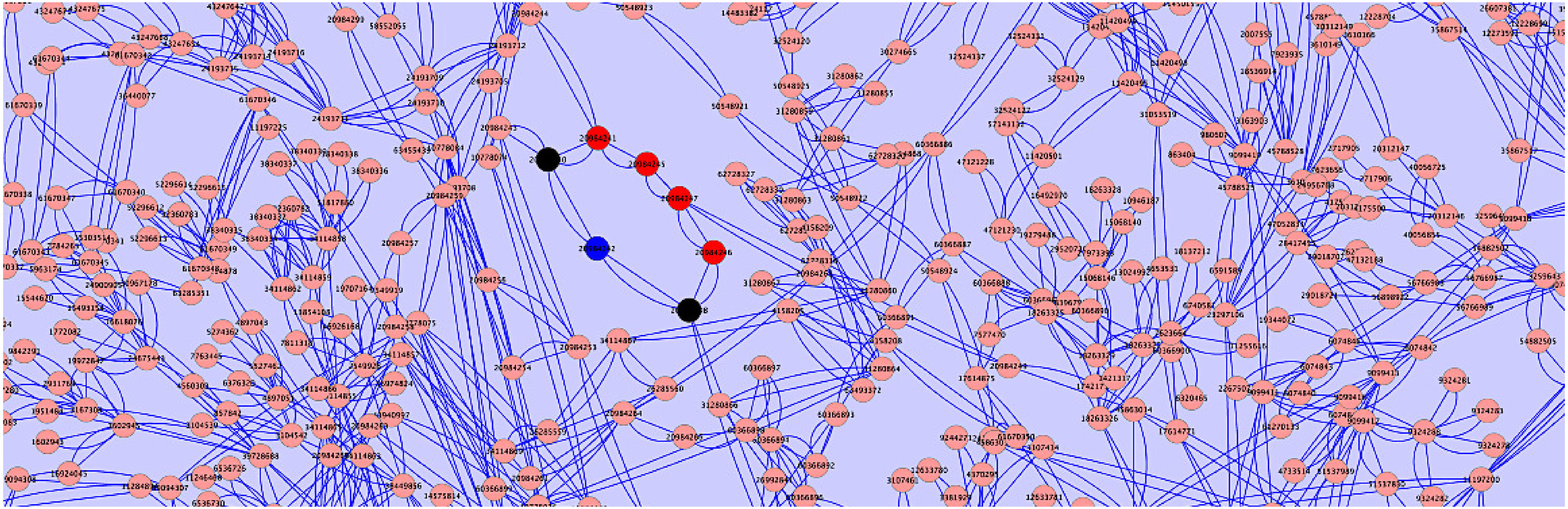
De Bruijn graph: snapshot



An alternative splicing event in the SCN5A gene (human) trapped inside a complex region.

Listing all the bubbles: KisSplice

De Bruijn graph: snapshot



An alternative splicing event in the SCN5A gene (human) trapped inside a complex region.

- The complexity comes from highly repeated sequences e.g. TEs in introns of pre-mRNA not yet spliced in RNA-seq data.

Repeat identification

The problem

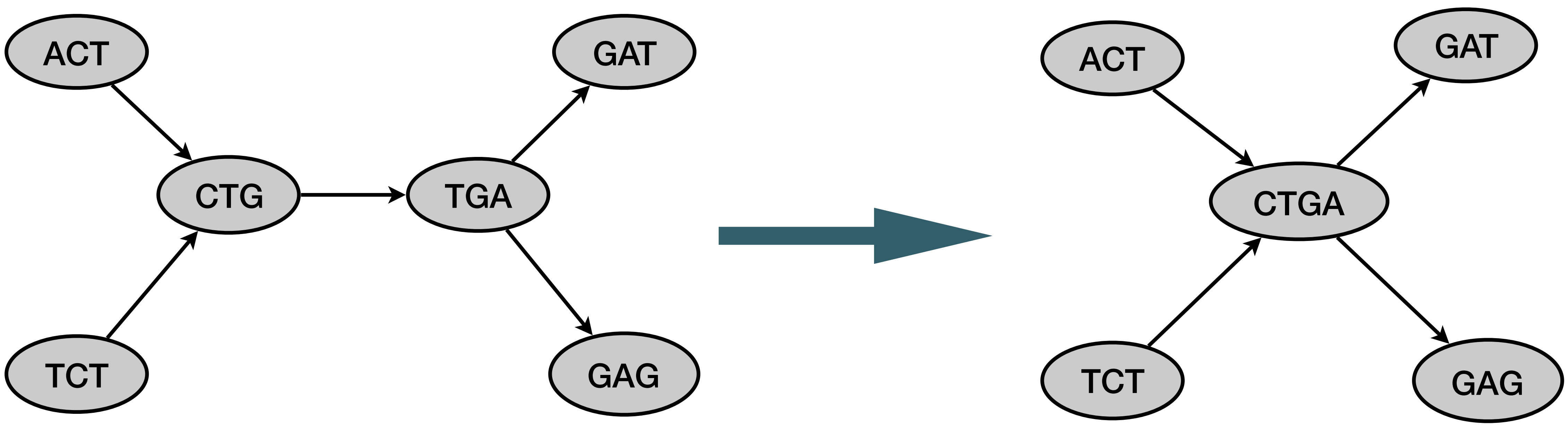
- Can we identify in a de Bruijn graph a subgraph corresponding to repeats?
- What characteristics has the subgraph induced by the repeats?

Our case

- no reference genome or repeat database
- no information on the coverage
  (on RNA-seq this depends also on the expression level of a gene, thus it is not informative)
- high-copy number approximate repeats

Repeats in the de Bruijn graph

Compressed de Bruijn graph

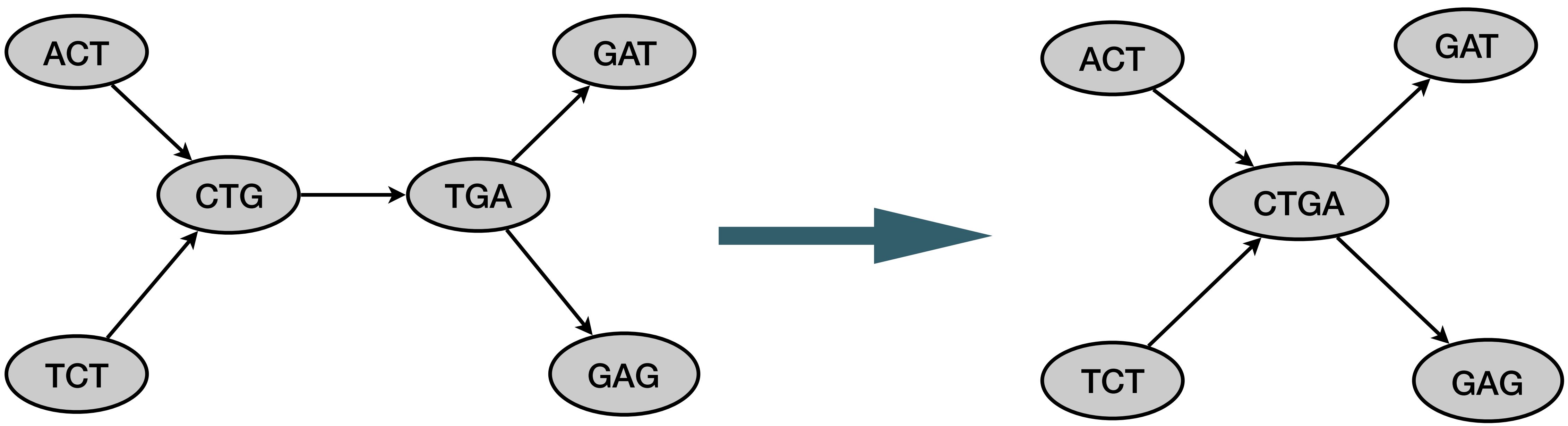


- The arc **(CTG,TGA)** can be compressed.

An arc **(u, v)** is compressible if $\mathbf{d^+(u) = d^-(v) = 1}$.

Repeats in the de Bruijn graph

Compressed de Bruijn graph



- The arc **(CTG,TGA)** can be compressed.

Idea: Repeats must induce a subgraph of "few" compressible arcs

Is it a good characteristics?

Random sequences

- Choose a set of m sequences of length **n** randomly from $\{A,C,T,G\}^n$

Repeats

- Let **α** be the mutation factor, $s_0 \in \{A,C,T,G\}^n$

$$
S(m,n,\alpha) = \begin{array}{cccccccccc|c}
A & A & C & T & G & T & A & T & C & C & s_0 \\
\hline
A & C & C & T & G & T & A & G & C & C & s_1 \\
G & A & C & T & C & A & A & T & C & C & s_2 \\
A & A & C & T & C & T & A & T & C & C & s_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\
A & A & A & T & G & T & A & T & C & T & s_m
\end{array}
$$

Is it a good characteristics?

Random sequences

- Choose a set of m sequences of length **n** randomly from $\mathbf{\{A,C,T,G\}^n}$

The expected number of compressible edges is $\mathbf{\Theta(mn)}$.

Repeats

- Let $\boldsymbol{\alpha}$ be the mutation factor, $\mathbf{s_0} \in \mathbf{\{A,C,T,G\}^n}$

The expected number of compressible edges is $\boldsymbol{o(mn)}$.

Identifying the repeat associated subgraph

Problem (Repeat Subgraph)

Instance: A directed graph **G** and two positive integers **n, t**

Decide: If there exists a connected subgraph **G'=(V', E')** with **|V'|≥n** and having at most **t** compressible edges.

Theorem

The Repeat Subgraph Problem is NP-complete even for subgraphs of de Bruijn graphs on an alphabet on 4 symbols.

Identifying the repeat associated subgraph

Sketch of the proof

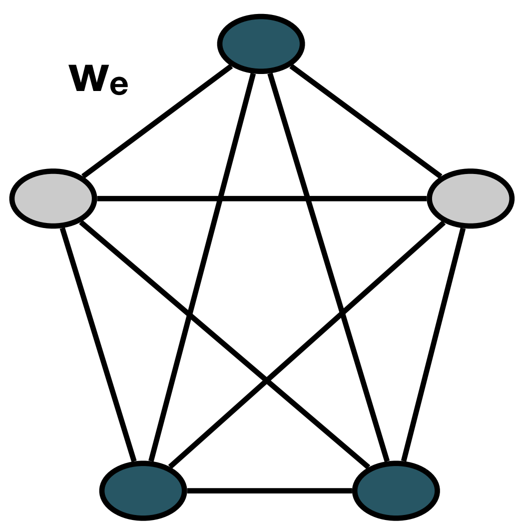
Problem (STEINER(1,2))

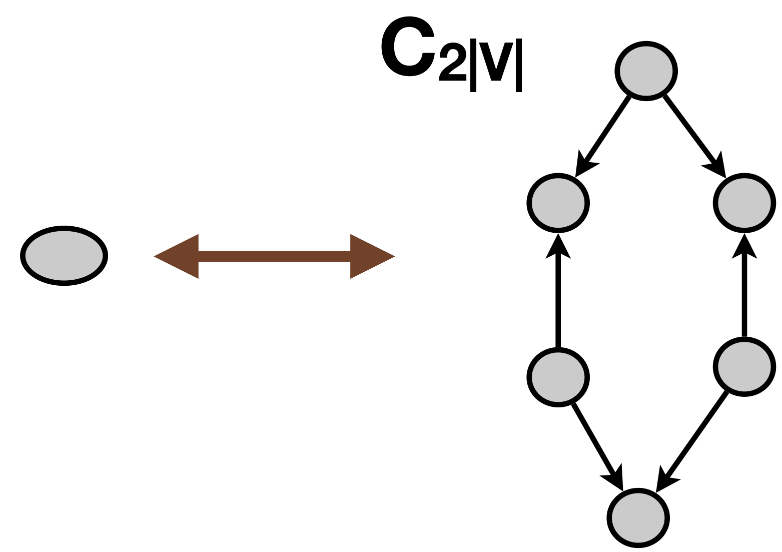
Instance: A complete undirected graph **G**, with edge weights in **{1,2}**, a set of terminal vertices **N** and an integer **B**

Decide: If there exists a connected subgraph **G'=(V', E')** with weight at most **B** containing all terminal vertices in **N**.

Identifying the repeat associated subgraph

Sketch of the proof

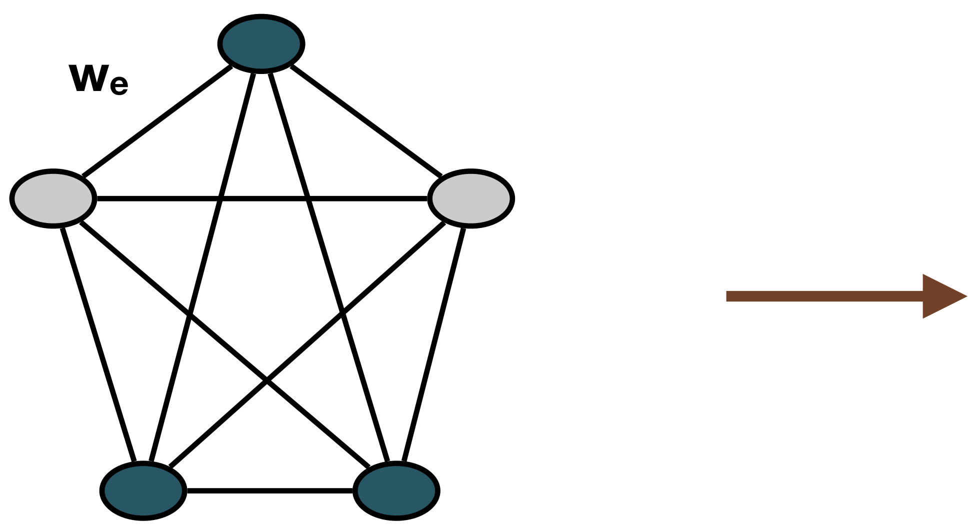


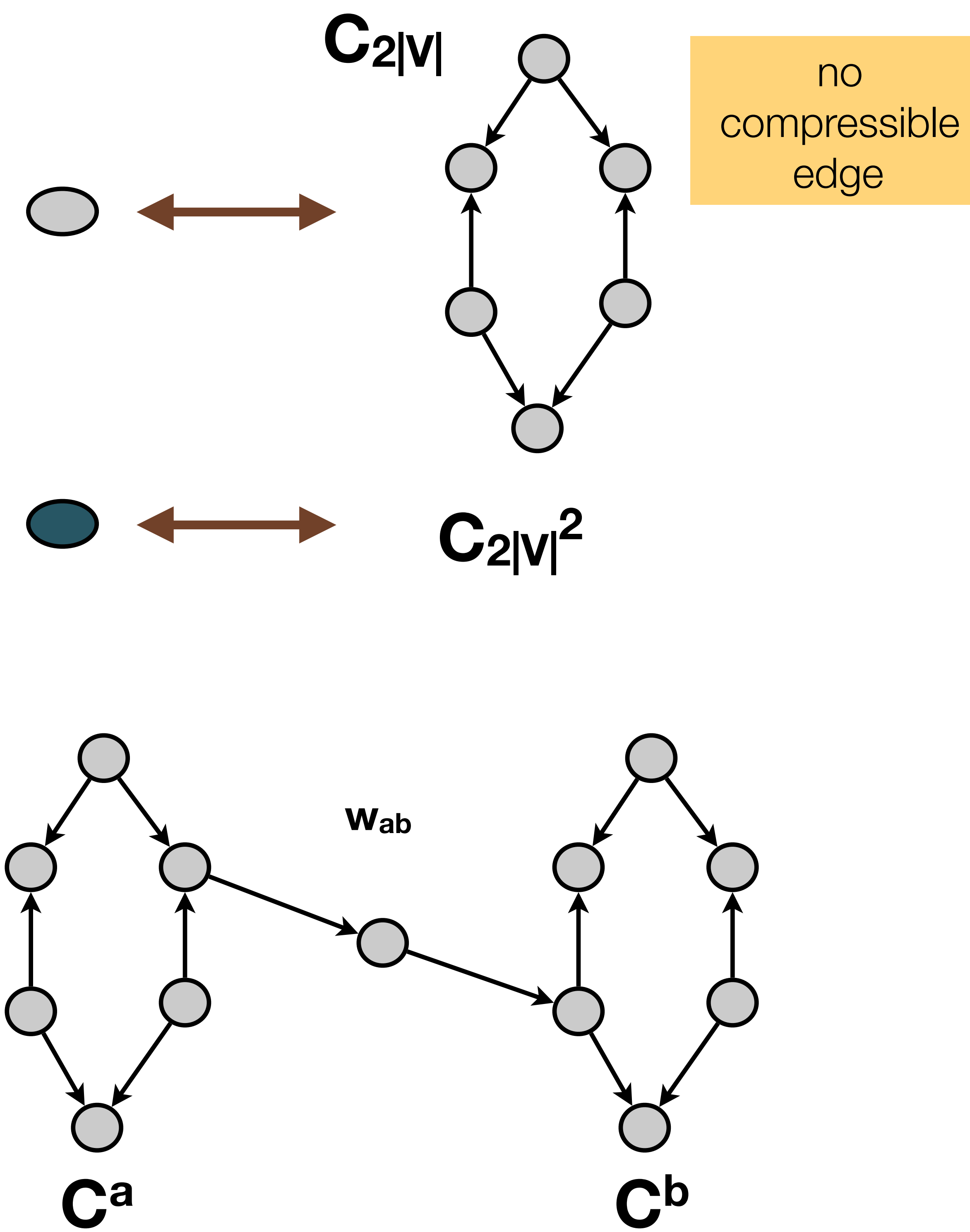


no compressible edge

Identifying the repeat associated subgraph

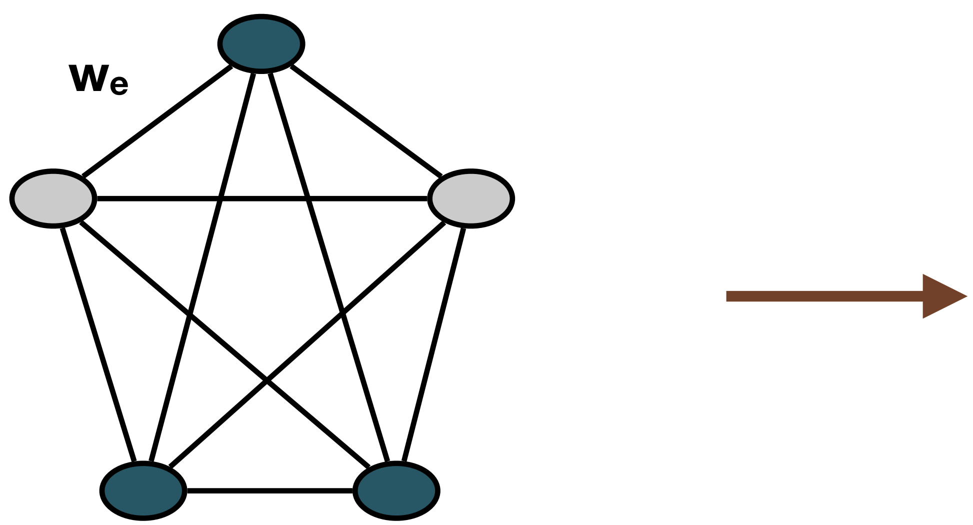
Sketch of the proof

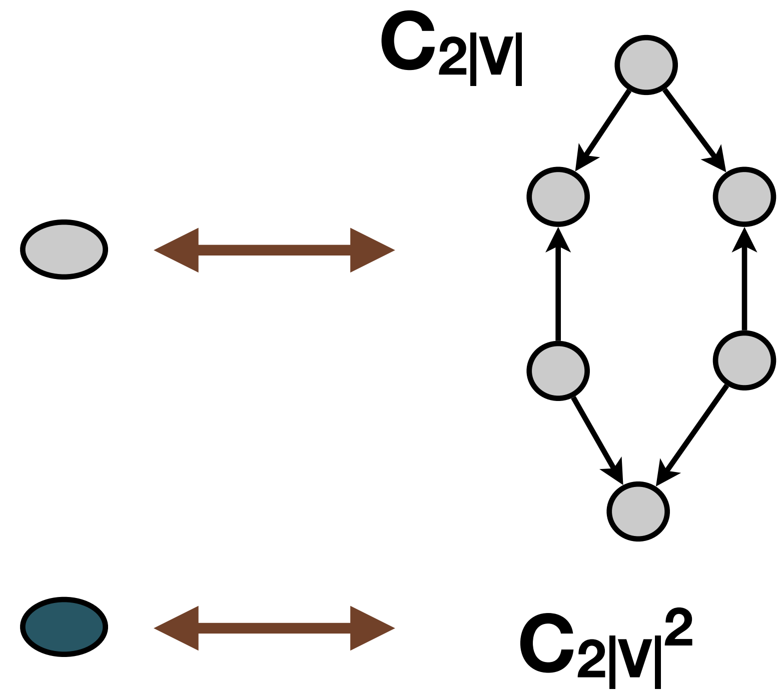




Identifying the repeat associated subgraph

Sketch of the proof

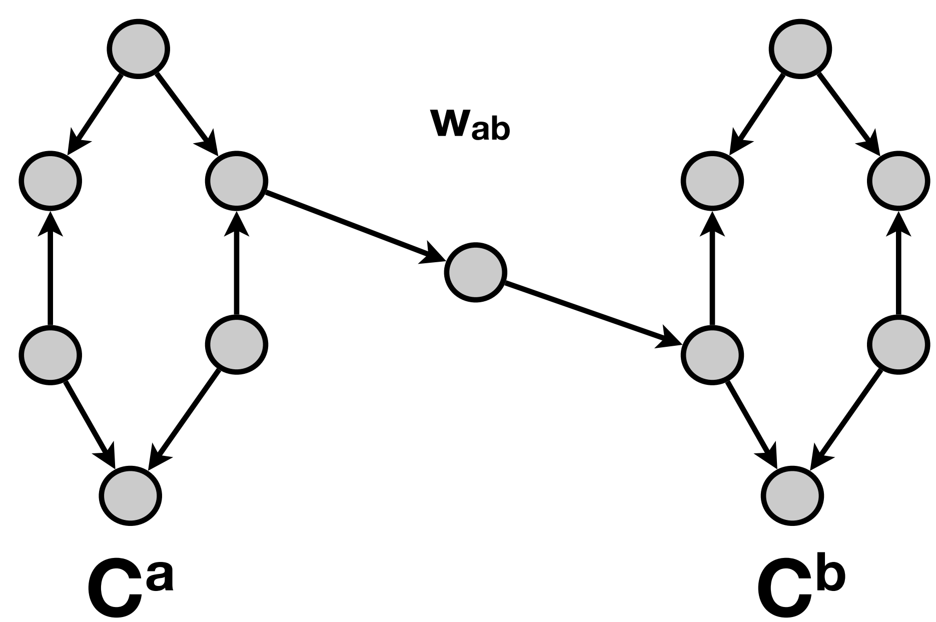




no compressible edge

N vertices in **G** -> **Nx2|V|^2** vertices in **H**

subgraph **G'** of weight at most **B** -> subgraph **H'** with at most **B** compressible edges.

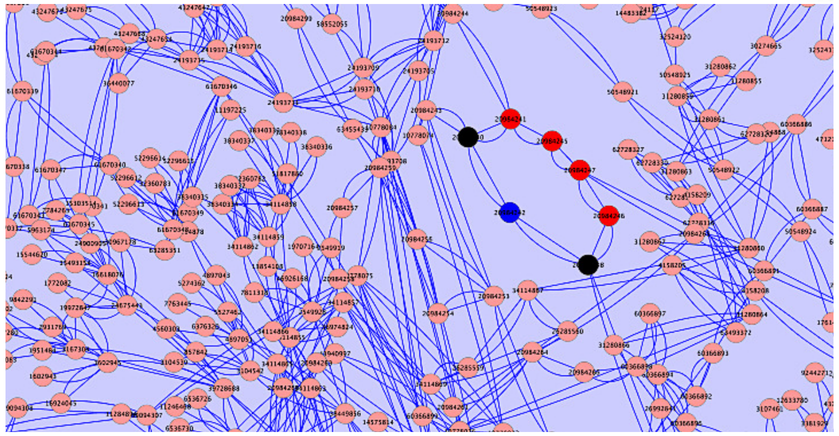


Enumerating bubbles avoiding repeats

For local assembly of AS events we can implicitly avoid repeat-associated subgraphs.

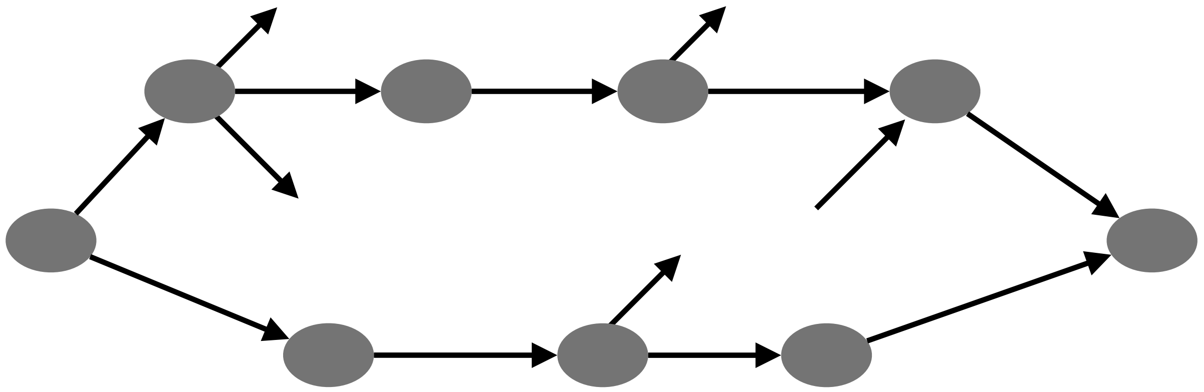
Main idea

Avoid paths with “many” branching vertices.



(s,t,a_1,a_2,b)-bubble is a pair of vertex disjoint st-paths with lengths bounded by a_1, a_2 and each one of them containing at most b branching vertices.

- $a_1 = 5$, $a_2 = 6$
- $b = 3$



Enumerating bubbles avoiding repeats

Algorithm (Main idea)

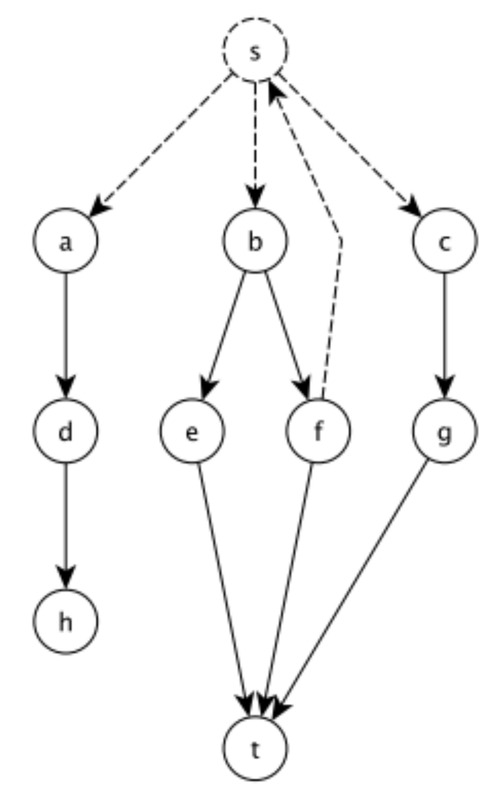
(s,t,a_1,a_2,b)-bubble is a pair of vertex disjoint st-paths with lengths bounded by a_1, a_2 and each one of them containing at most b branching vertices.

For every vertex ***s*** **do**
// Generate $B_s(s, * , a_1, a_2, b)$

For every edge ***e*** outgoing ***s*** **do**
// bubbles from B_s that contain edge e.
$B(p_1\, e, p_2, G' - u_1)$
// bubbles from B_s that do not contain edge e.
$B(p_1 , p_2, G' - u_1)$

Initially

- $u_1 = u_2 = s$
- $p_1 = s \longrightarrow u_1$
- $p_2 = s \longrightarrow u_2$
- $G' = G$



Enumerating bubbles avoiding repeats

Algorithm (Main idea)

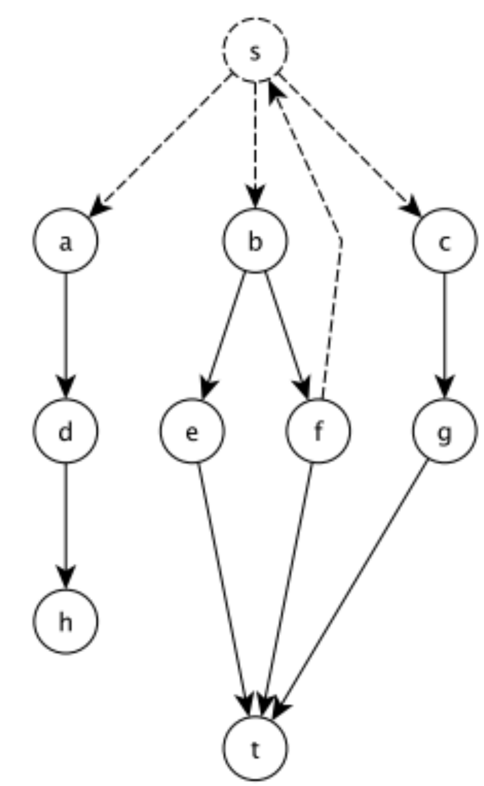
(s,t,a_1,a_2,b)-bubble is a pair of vertex disjoint st-paths with lengths bounded by a_1, a_2 and each one of them containing at most b branching vertices.

For every vertex ***s* do**
// Generate $B_s(s, *, a_1, a_2, b)$

For every edge ***e*** outgoing ***s* do**
// bubbles from B_s that contain edge e.
$B(p_1\, e, p_2, G' - u_1)$
// bubbles from B_s that do not contain edge e.
$B(p_1, p_2, G' - u_1)$

Initially

- $u_1 = u_2 = s$
- $p_1 = s \longrightarrow u_1$
- $p_2 = s \longrightarrow u_2$
- $G' = G$



decide whether these calls are not empty

$p'_1 = u_1 \longrightarrow t$ and $p'_2 = u_2 \longrightarrow t$ with $|p'_1| \le a_1$; $|p'_2| \le a_2$ and at most b branching vertices.

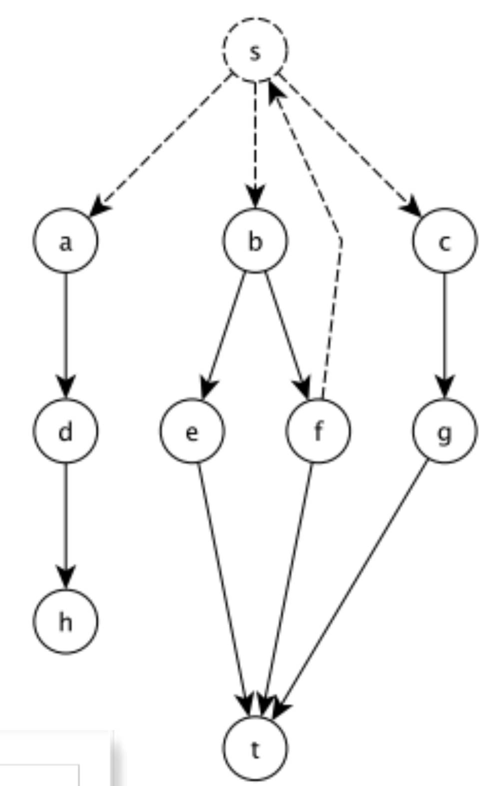
Enumerating bubbles avoiding repeats

Algorithm (Main idea)

(s,t,a_1,a_2,b)-bubble is a pair of vertex disjoint st-paths with lengths bounded by a_1, a_2 and each one of them containing at most b branching vertices.

For every vertex ***s* do**
// Generate $B_s(s, *, a_1, a_2, b)$

For every edge ***e*** outgoing ***s* do**
// bubbles from B_s that contain edge e.
$B(p_1\, e, p_2, G' - u_1)$
// bubbles from B_s that do not contain edge e.
$B(p_1, p_2, G' - u_1)$



Enumerate bubbles with at most **b** branching vertices with polynomial delay $\boldsymbol{O(b\,|V|^3|E|)}$.

What’s next?

Third Generation Sequencing