DSB 2022

The reverse complement symmetry advantage of DNA fragments relationships for their storage in a directed graph

Victor EPAIN

2nd year PhD, Inria

under the responsibilities of :

Rumen ANDONOV
PR Univ. Rennes 1

Jean-François GIBRAT
DR INRAe

Dominique LAVENIER
DR CNRS

dsb

June, the 13th 2022

Outline

I. **Introduction**
   1. DNA fragments
   2. DNA fragments relationships

II. **Overlaps Graph**
   1. Bidirected *versus* directed
   2. Efficient directed graph

III. **RevSymG**
   1. Structure
   2. Weakly connected component
   3. Python3 package

IV. **Conclusion & Discussion**

Introduction – DNA fragments

Raw data

- DNA fragments
  - raw reads
  - contigs

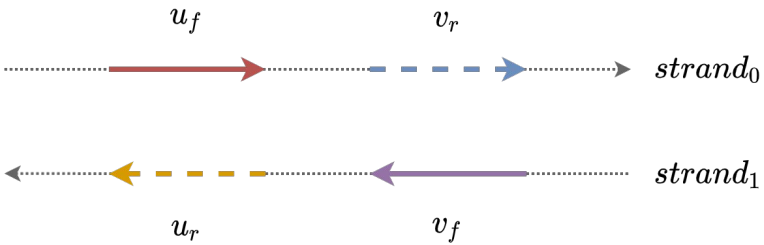
Axiom

- 2 DNA strands are both sequenced in reverse reading
- Reads are randomly sampled from either a strand or its complementary

$\mathcal{R}_{aw}$

u_f

v_f

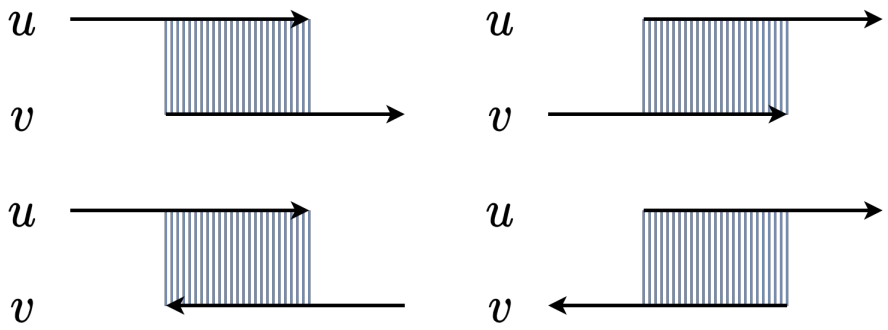


Introduction – DNA fragments relationships

- Oriented fragments
  - *forward / reverse*
- Oriented alignments
  - *oriented u before / after oriented v*

- In an alignment file, each of these 4 overlaps corresponds to one line
  - u is considered in forward orientation

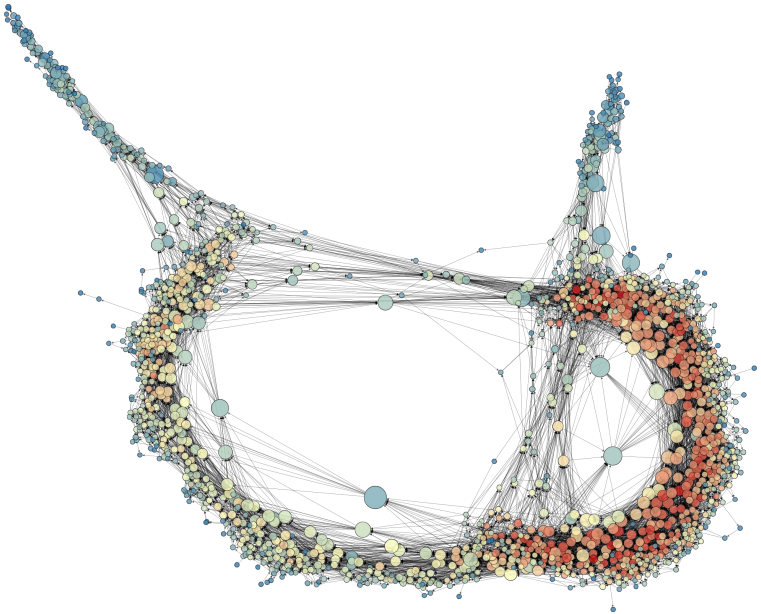
`u_id  v_id  v_or  u_before_v?`



Introduction – DNA fragments relationships

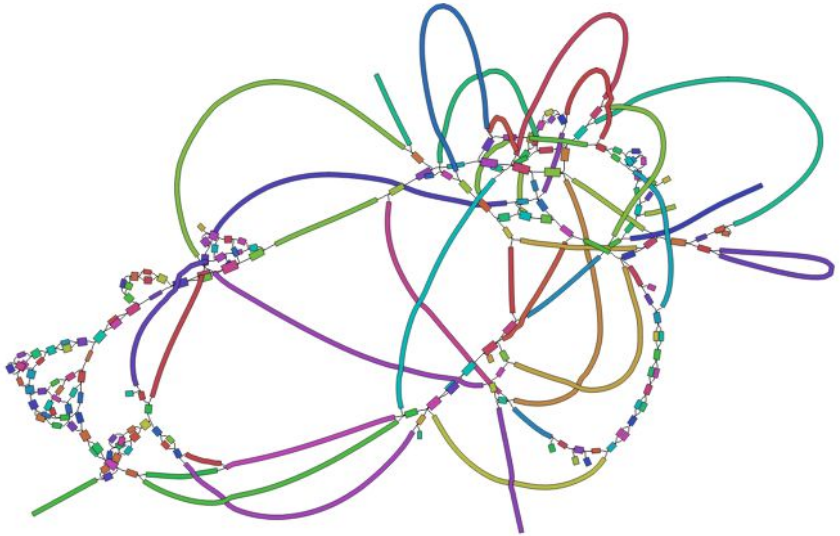
Assembly – Compute contigs

- From reads to contigs
  - *e.g. **O**verlaps **L**ayout **C**onsensus* paradigm



Assembly – Scaffolding

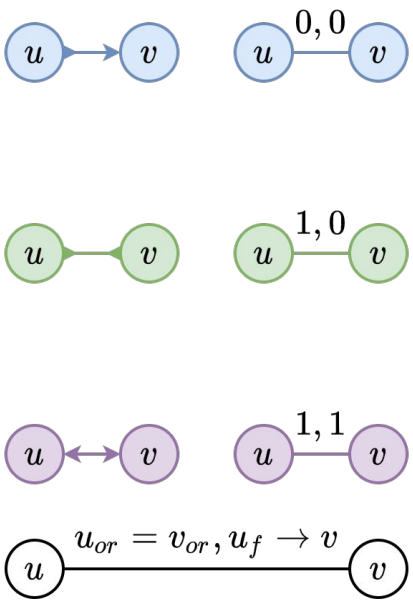
- Finishing assembly:
  ordering & orienting merged reads
  - *Input: assembly graph*



Overlaps Graph – Bidirected *Versus* Directed Graphs

Bidirected graph
[Myers, 1995]

- Fragments are not doubled according their two orientations
- Must verify if oriented neighbors are predecessors or successors



(In alignments file)

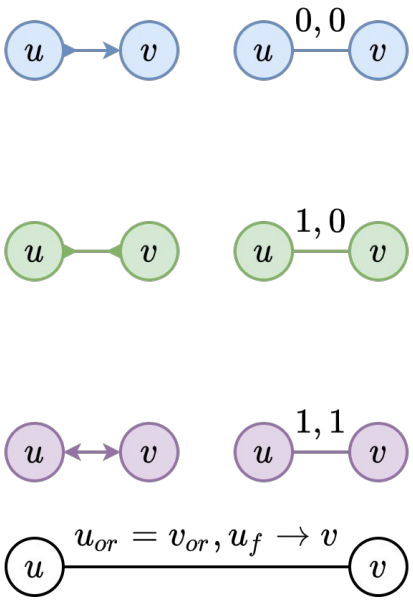
```
u_id  v_id  v_or  u_before_v?
```

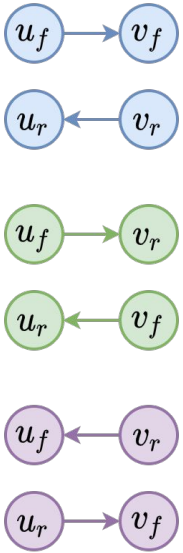
Myers EW. Toward simplifying and accurately formulating fragment assembly. J Comput Biol. 1995;2(2):275–90.

Overlaps Graph – Bidirected *Versus* Directed Graphs

Bidirected graph
[Myers, 1995]

- Fragments are not doubled according their two orientations
- Must verify if oriented neighbors are predecessors or successors





Directed graph
[our approach]

- Fragments are doubled according their two orientations
- Getting oriented predecessors / successors is immediate

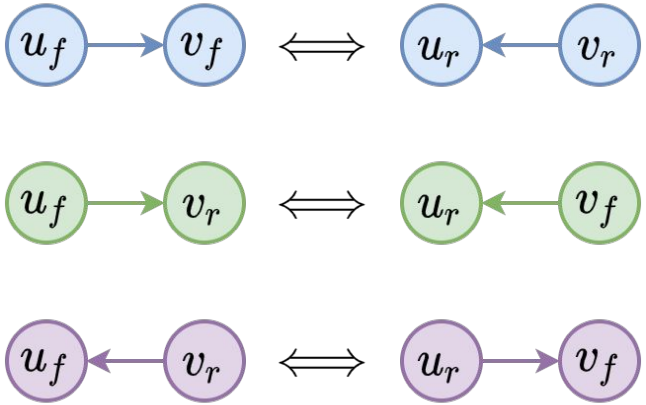
(In alignments file)

```
u_id  v_id  v_or  u_before_v?
```

Myers EW. Toward simplifying and accurately formulating fragment assembly. J Comput Biol. 1995;2(2):275–90.

Overlaps Graph – Efficient Directed Graph

How to take advantage of DNA reads overlaps' reverse complement symmetry to store them in an oriented graph?



$$o \in \mathcal{O} \,|\, o \in \{(u_f, v), (v, u_f)\} \iff \bar{o}$$

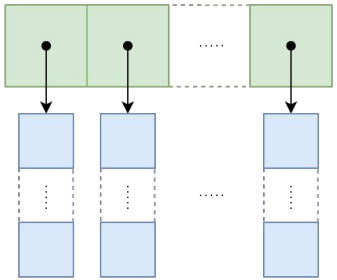
Overlaps Graph – Efficient Directed Graph

Construction

- A fragment = two nodes *(forward & reverse)*
- Tables of predecessors & successors only for the forwards
- Keep overlaps' reverse complement symmetry
  - edges are not really duplicated in memory

$0 \quad 1 \quad |\mathcal{I}nd| - 1 \} \quad \mathcal{I}nd$

Fragments' indices dimension



$\mathcal{I}ndor = \mathcal{I}nd \times \mathbb{B}$

Oriented fragments' index-orientation dimension

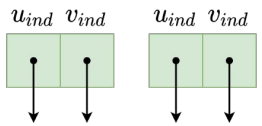
Advantages

- Classical directed graph structure
  - iterating over predecessors / successors oriented fragments is immediate
- Better view of two strands sequencing

Storage of all overlap cases between two fragments

$$o \in \mathcal{O} \,|\, o \in \{(u_f, v), (v, u_f)\} \iff (\bar{o})$$

$N^-_{V_f}$ $N^+_{V_f}$

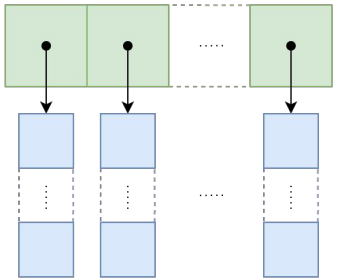


$f_{\mathbb{B}} = 0$

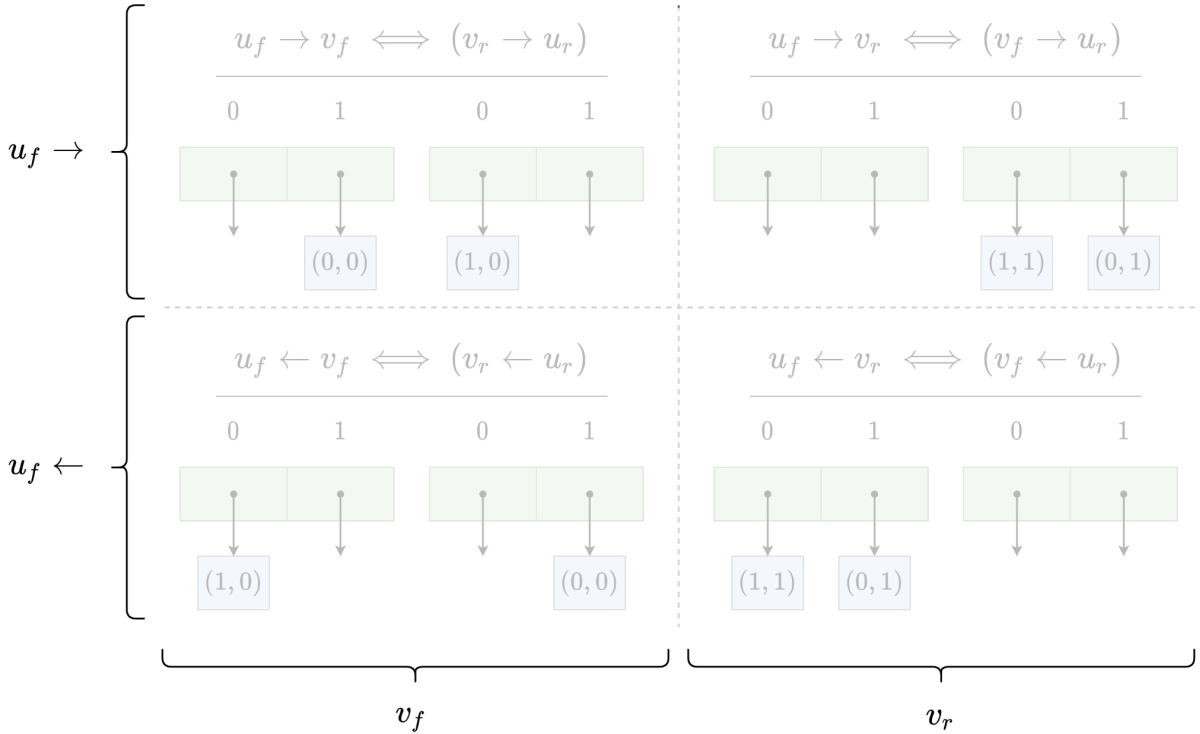
$r_{\mathbb{B}} = 1$

Fragments' indices
&
Oriented fragments' index-orientation dimensions

$0 \quad 1 \quad |\mathcal{I}nd| - 1 \} \quad \mathcal{I}nd$



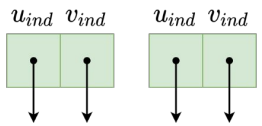
$\mathcal{I}ndor = \mathcal{I}nd \times \mathbb{B}$



Storage of all overlap cases between two fragments

$o \in \mathcal{O} \,|\, o \in \{(u_f, v), (v, u_f)\} \iff (\bar{o})$

$N^-_{V_f}$ $N^+_{V_f}$

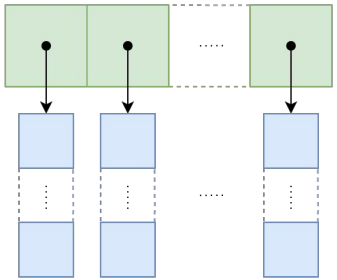


$f_{\mathbb{B}} = 0$

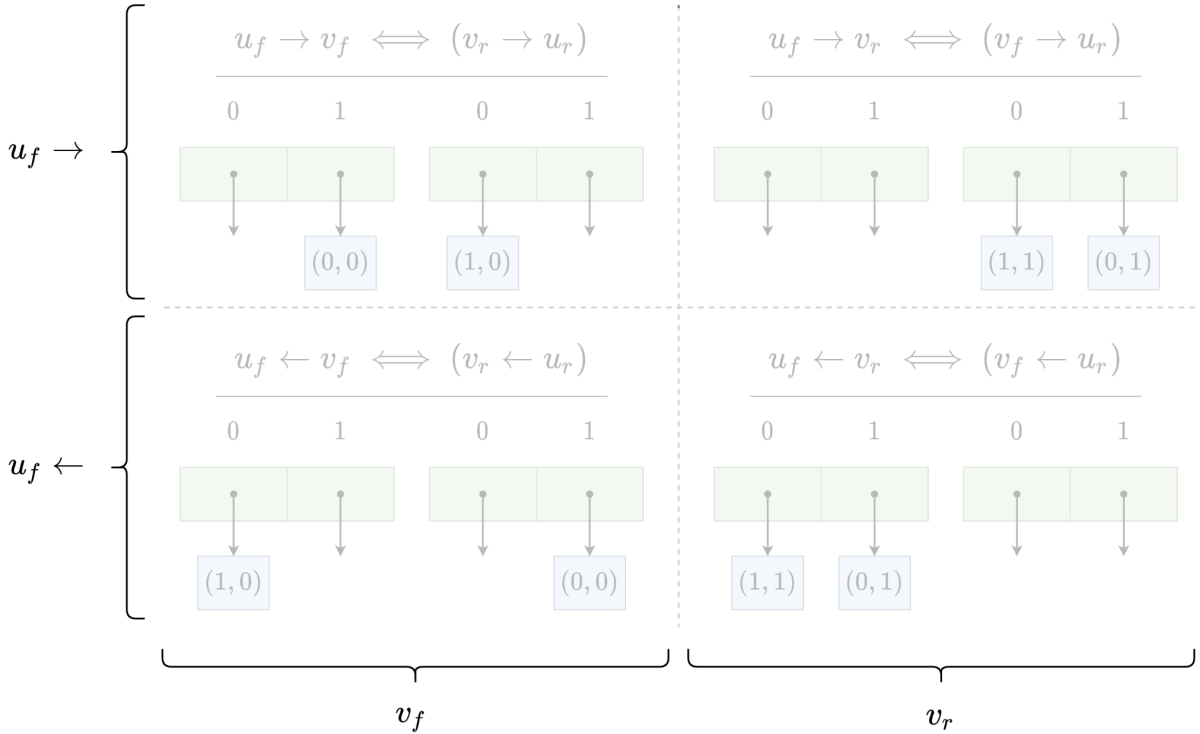
$r_{\mathbb{B}} = 1$

Fragments' indices
&
Oriented fragments' index-orientation dimensions

$0 \quad 1 \quad |\mathcal{I}nd| - 1 \} \quad \mathcal{I}nd$



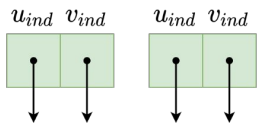
$\mathcal{I}ndor = \mathcal{I}nd \times \mathbb{B}$



Storage of all overlap cases between two fragments

$$o \in \mathcal{O} \,|\, o \in \{(u_f, v), (v, u_f)\} \iff (\bar{o})$$

$N^-_{V_f}$ $N^+_{V_f}$

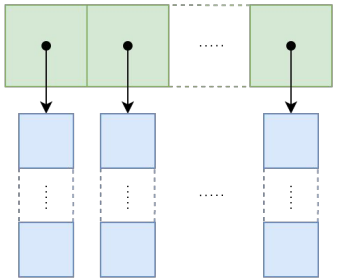


$f_{\mathbb{B}} = 0$

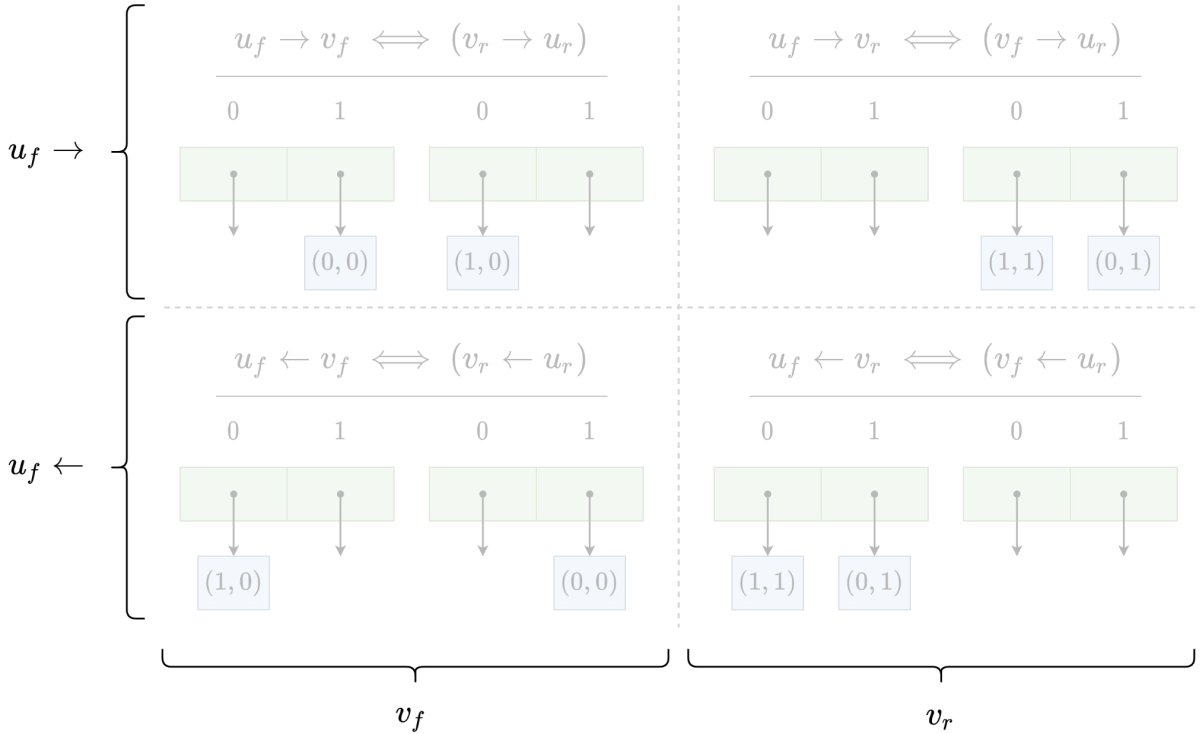
$r_{\mathbb{B}} = 1$

Fragments' indices
&
Oriented fragments' index-orientation dimensions

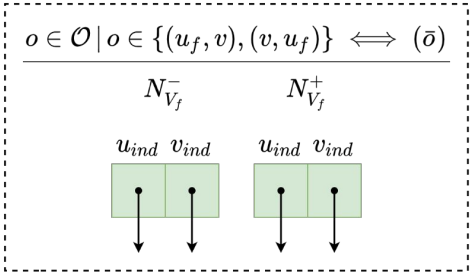
$0 \quad 1 \quad |\mathcal{I}nd| - 1 \} \quad \mathcal{I}nd$



$$\mathcal{I}ndor = \mathcal{I}nd \times \mathbb{B}$$



Storage of all overlap cases between two fragments

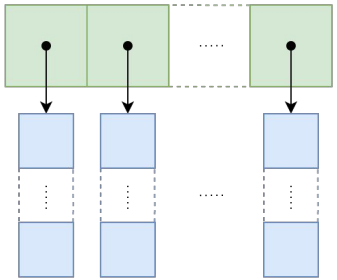


$f_{\mathbb{B}} = 0$

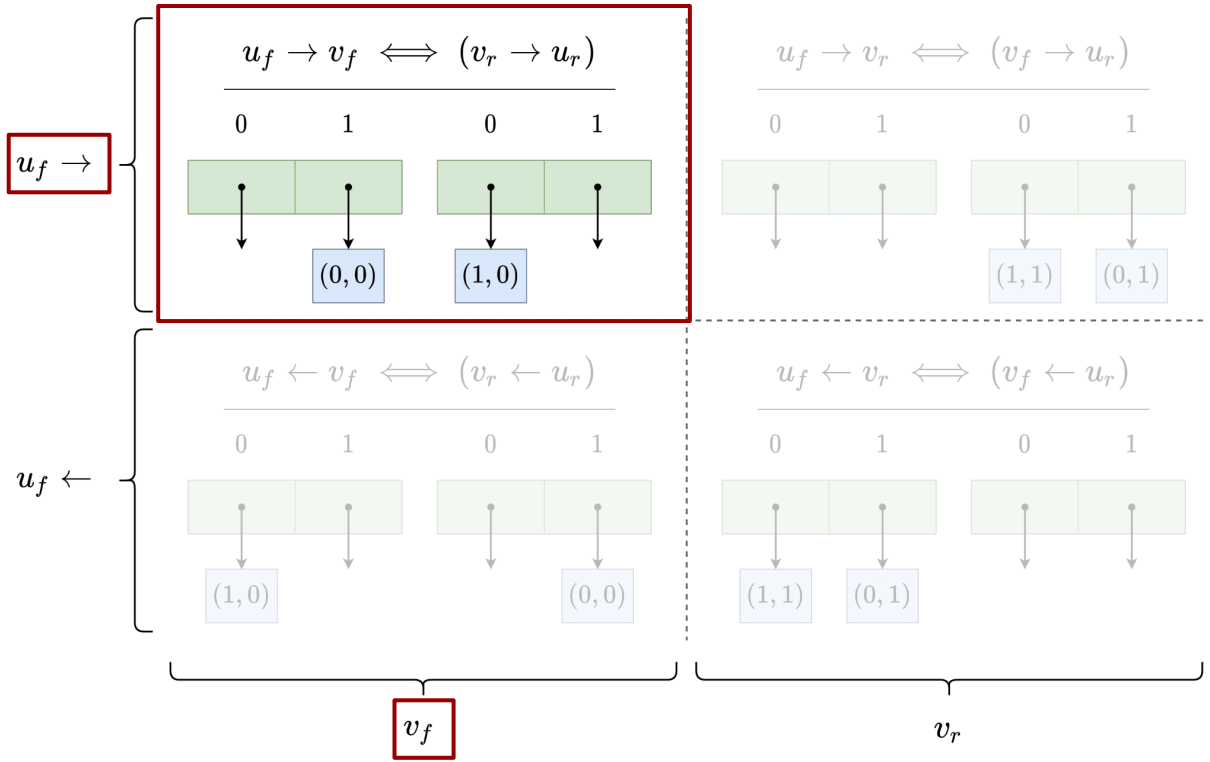
$r_{\mathbb{B}} = 1$

Fragments' indices
&
Oriented fragments' index-orientation dimensions

$0 \quad 1 \quad |\mathcal{I}nd| - 1 \}\ \mathcal{I}nd$



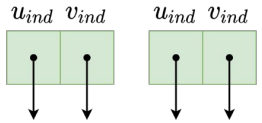
$\mathcal{I}ndor = \mathcal{I}nd \times \mathbb{B}$



Storage of all overlap cases between two fragments

$o \in \mathcal{O} \,|\, o \in \{(u_f, v), (v, u_f)\} \iff (\bar{o})$

$N^-_{V_f}$ $N^+_{V_f}$

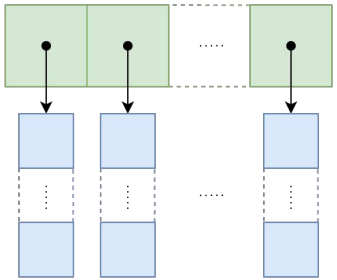


$f_{\mathbb{B}} = 0$

$r_{\mathbb{B}} = 1$

Fragments' indices
&
Oriented fragments' index-orientation dimensions

$0 \quad 1 \quad |\mathcal{I}nd| - 1 \} \quad \mathcal{I}nd$



$\mathcal{I}ndor = \mathcal{I}nd \times \mathbb{B}$

$u_f \rightarrow$

$u_f \leftarrow$

$u_f \rightarrow v_f \iff (v_r \rightarrow u_r)$

0 1 0 1

$(0,0)$ $(1,0)$

$u_f \rightarrow v_r \iff (v_f \rightarrow u_r)$

0 1 0 1

$(1,1)$ $(0,1)$

$u_f \leftarrow v_f \iff (v_r \leftarrow u_r)$

0 1 0 1

$(1,0)$ $(0,0)$

$u_f \leftarrow v_r \iff (v_f \leftarrow u_r)$

0 1 0 1

$(1,1)$ $(0,1)$

v_f

v_r

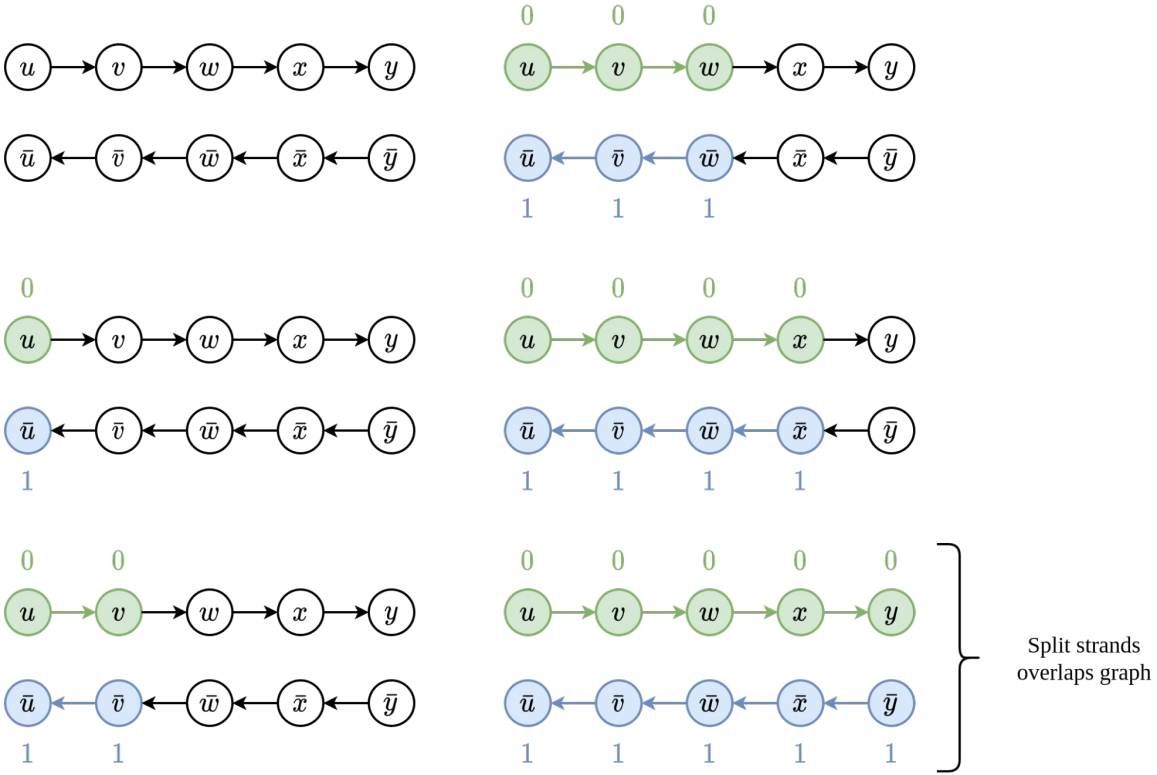
Overlaps Graph – RevSymG

Algorithmic advantage

- Transitive reduction is more efficient
  - iterate over successors of successors is less time consuming
- Detect and identify weakly connected component
  - is there a connexion between the two strands?
  - time consumption divided by two when giving component's identifier to oriented fragments

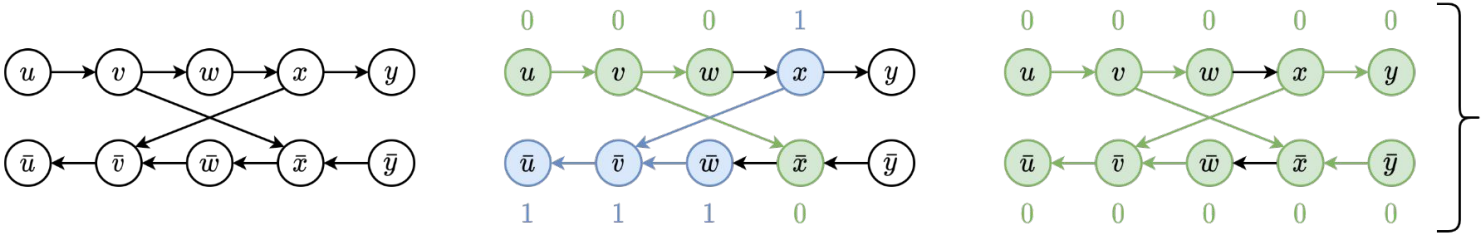
RevSymG – Weakly Connected Components

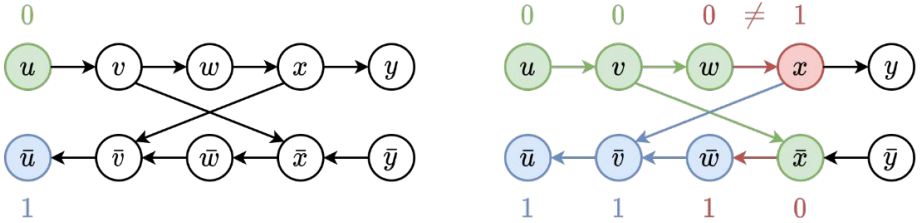
Is there a connexion between the two strands?

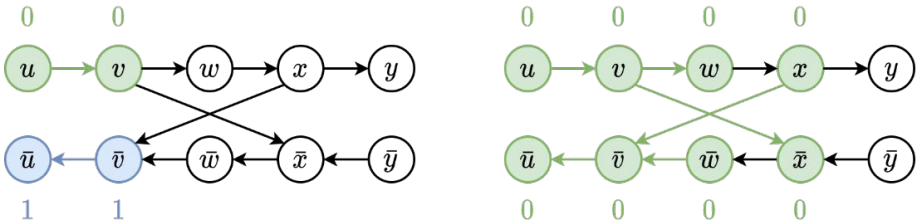


RevSymG – Weakly Connected Components

Is there a connexion between the two strands?







RevSymG – Weakly Connected Components

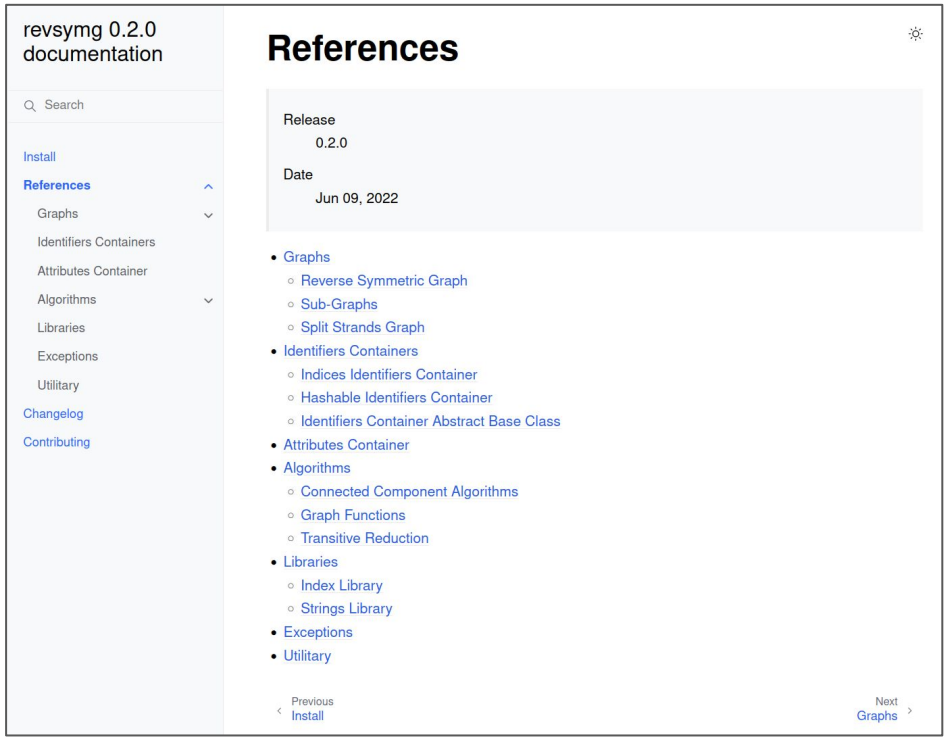
Is there a connexion between the two strands?

- Assumption: *the two orientations of a fragment belong to two different connected components (c.c.)*
  - c.c. identifiers are in a list of couples of integers `(i, i + 1)`
  - each forward vertex is associated with a c.c. identifier `(couple_index, cc_index_in_couple)`
- Explore neighbours
  - if neighbour does not have yet a c.c. identifier
    - give the c.c. identifier of its source
    - put the neighbour in a FIFO container
      - better to use reads coverage constant than graph depth with a LIFO
  - else if its c.c. identifier = to the one of its source: continue
  - else: set the second identifier to `i` in the couple `(i, i)`
- Explore from a new exploration root
  - Add a new couple of c.c. identifiers
    - distant by 1 from the max of the last couple
  - repeat exploring neighbours step

RevSymG – Python3 Package

Implementation

- Python3 package
  - tests coverage
    - > 90%
  - strict coding conventions
    - modular
    - linters
    - tox environment
  - (quasi-)ready to be deployed



Conclusion

- Overlaps between fragments are first modelled by a bidirected graph
  - Myers' structure
  - no node duplication implies a cost on iteration over oriented predecessors / successors
- *Symbolically* double the nodes permits to structure overlaps with a directed graph
  - fragments' reverse complement symmetry is benefic to not double edges (and nodes)
- RAM implementation
- More time efficient transitive reduction algorithm & easy inverted repeats identification
- Overlaps graph is a common structure
  - reads' overlaps graph
  - assembly graph
  - *other usages? (blind-double-strands sequencing: RNA? ChIP-seq?)*

This talk is funded by the French Bioinformatics Society (SFBI)

Discussion – What I did not speak about

- Who use explicitly overlaps graph?
  - how they implement it?
- Fragments' indices dimension
  - how to hash fragments' identifiers
- Union disjoint sets for weakly connected component
- Transitive reduction details
- RAM based structure
  - *QUID* Disk based structure?
- Why Python?
- Package and code architecture