Algorithms for Genome Research

Pedro Feijão

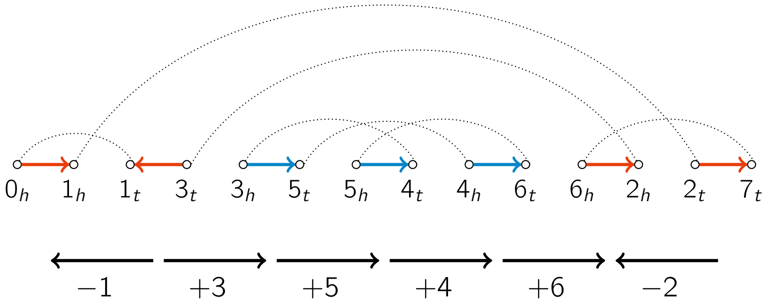
Lecture 2 – Sorting by Signed Reversals II

Summer 2014

pfeijao@cebitec.uni-bielefeld.de

Quick Recap

BP Graph, **oriented** and **unoriented** components:



Quick Recap

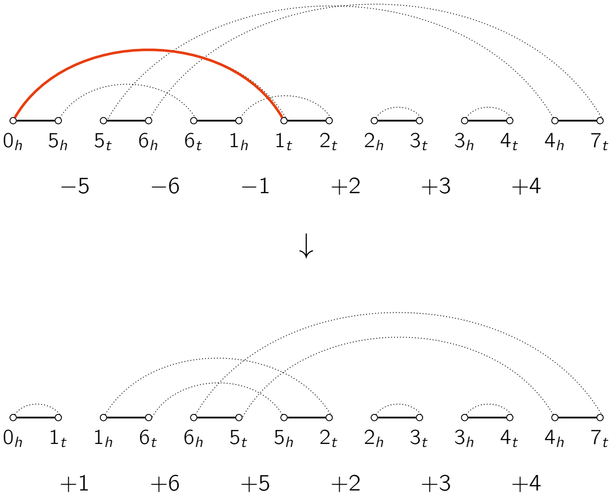
- Sorting is equivalent of increasing # of cycles in BP graph
  - In *oriented (good) components* – *at least 1 oriented edge* – this is always possible.
  - In *unoriented (bad) components*, not, so we need extra operations.

So we have the following **lower bound**:

$$d_R(\pi) \geq N - C$$

- There are also reversals that increase the number of cycles, but create unoriented components.

Bad reversal - Example



- Increased number of cycles but created a bad component!

Finding "good" reversals

- Is it possible to find a reversal that increases the number of cycles **AND** also does not create an unoriented component? **YES!**

Sorting oriented components

Theorem (Hannenhalli-Pevzer, 95)

If the graph $BP(\pi)$ has only **oriented components***, then*

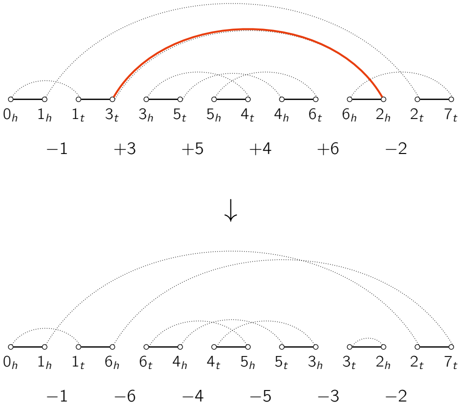
$$d_R(\pi) = N - C$$

where N is the number of elements of π and C is the number of cycles of $BP(\pi)$.

- This means that there is always at least one “good” reversal, that increases the number of cycles of $BP(\pi)$ and *does not create any unoriented component*.
- These are called **safe reversals**. How can we find them?

Safe reversals - Definitions

- The **score** of a reversal is the number of *oriented edges* in the BP graph, *after* the application of the reversal.



The score of this reversal is **two**.

Safe reversals

- **Safe reversals** are reversals that increase the number of cycles of the BP graph by one and do not create new unoriented components.
- Can we always find safe reversals? Yes:

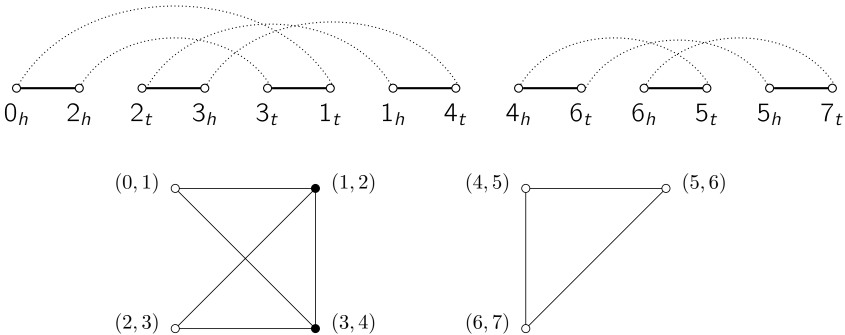
Theorem (Bergeron, 2001)

Among all possible oriented reversals, a reversal of maximal score is always safe.

- **Algorithm**: Apply maximal score reversals until all components are sorted.

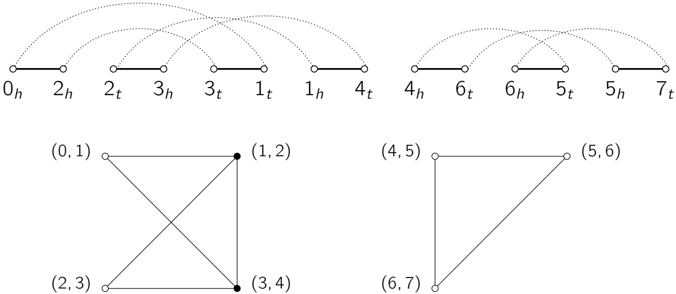
Finding safe reversals with the Overlap Graph

- The **overlap graph** $O(\pi)$ is a graph where:
  - Vertices are the grey edges of $BP(\pi)$. If the edge is oriented, the vertex is black, otherwise is white.
  - When two grey edges overlap, there is an edge between the corresponding vertices.



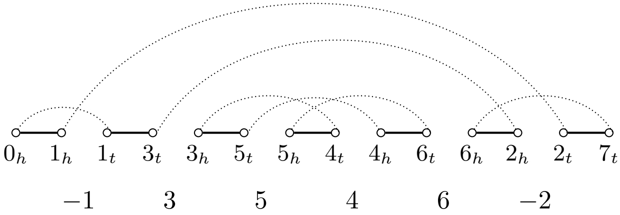
BP Graph vs Overlap Graph

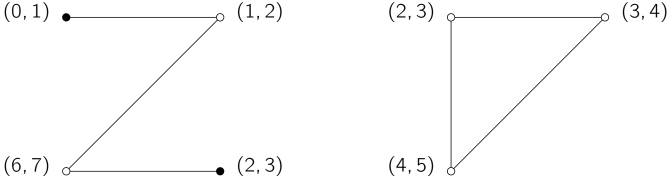
BP Graph	Overlap Graph
Component	Connected component
Oriented edge	Black vertex, *odd degree*
Unoriented edge	White vertex, *even degree*
Oriented component	Component with at least 1 black vertex
Unoriented component	Component with only white vertices



Another Example

$$\pi = [\,-1\;\;3\;\;5\;\;4\;\;6\;\;-2\,]$$

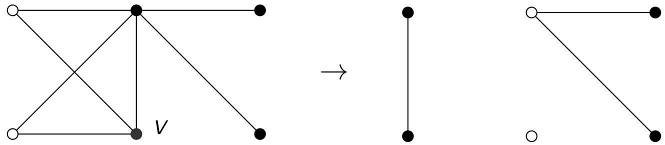




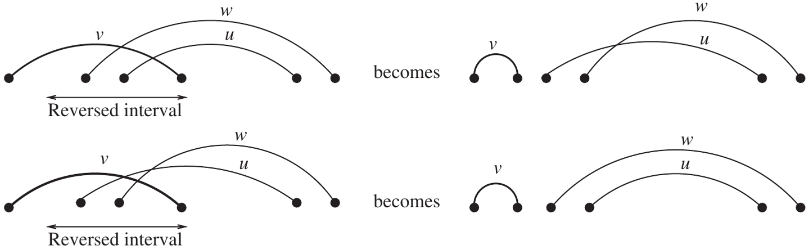
Effect of Reversal in the Overlap Graph

- A reversal *induced by a vertex v* is the reversal that is induced by the corresponding grey edge in the breakpoint graph.
- What happens in $O(\pi)$ after applying an oriented reversal in a vertex v?

1. The subgraph induced by v and its neighbours is **complemented**.



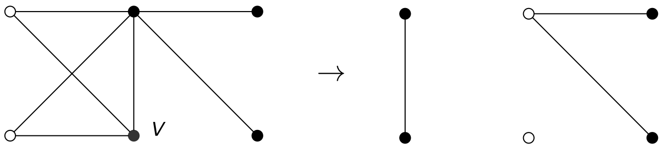
Why?



A. Bergeron / Discrete Applied Mathematics 146 (2005) 134–145

Effect of Reversal in the Overlap Graph

2 All neighbours of v have their orientation inverted.



Why?

Reversal Score with $O(\pi)$

We know how the overlap graph changes with a reversal, then it is possible to find an equation for the reversal score of any vertex v:

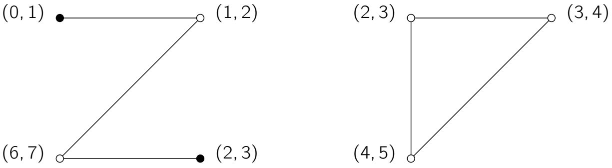
Definition (Reversal score)

The score of a reversal induced by a vertex v in the overlap graph is given by

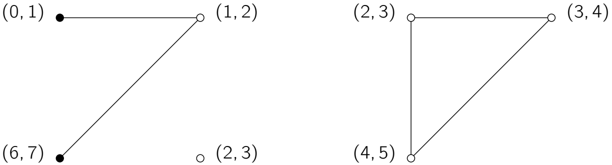
$$s(v) = T + U - O - 1$$

where T is the number of oriented vertices in the graph, U and O are the number of unoriented and oriented vertices adjacent to v, respectively.

Reversal Score - example



For $v = (2, 3)$, we have $T = 2$, $U = 1$, $O = 0$. Therefore $s(v) = T + U - O - 1 = 2$.
After applying the reversal, we have the following graph:



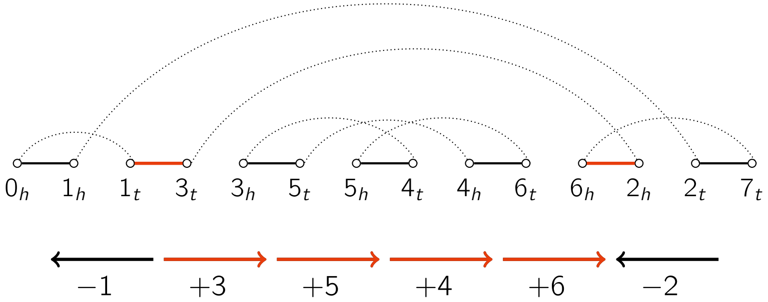
and we see that the score (number of oriented vertices) is indeed 2.

Sorting Example

$$\pi = (0 \quad 3 \quad 1 \quad 6 \quad 5 \quad -2 \quad 4 \quad 7)$$

Sorting Unoriented Components

- Let' s analyse the effect that reversals have on cycles of $BP(\pi)$.
- Reversals change # of cycles by -1, 0, or $+1$.
- What happens exactly when we apply a reversal **defined** by two black edges?



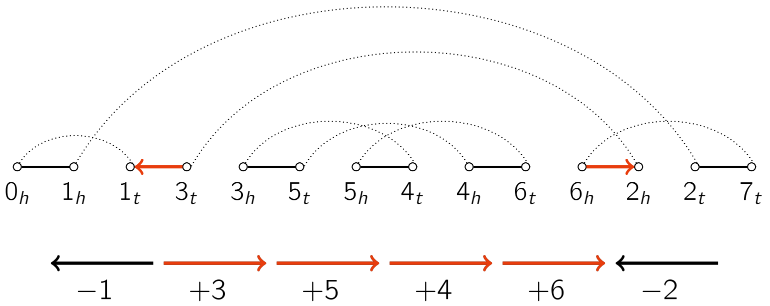

```
java InversionVisualisation L2/recap1.txt
```

Reversals and effect on cycles

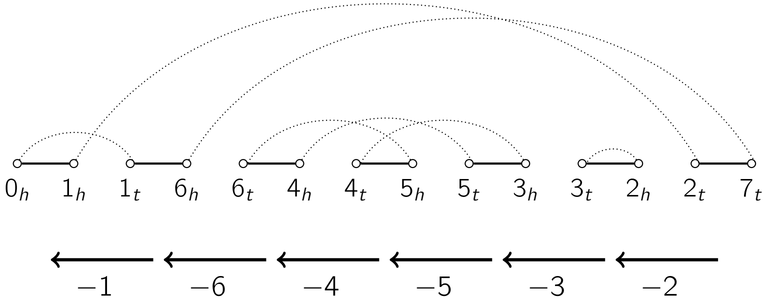
1. Edges are on the **same cycle**:
   - **Type I**: Divergent edges: breaks the cycle. $\Delta C = +1$.
   - **Type II**: Convergent edges: $\Delta C = 0$, may change cycle orientation.
2. Edges on **different cycles**:
   - **Type III**: Merges the two cycles. $\Delta C = -1$.

So far, we only used **Type I** operations, to sort oriented components.

Type I - Same Cycle, divergent

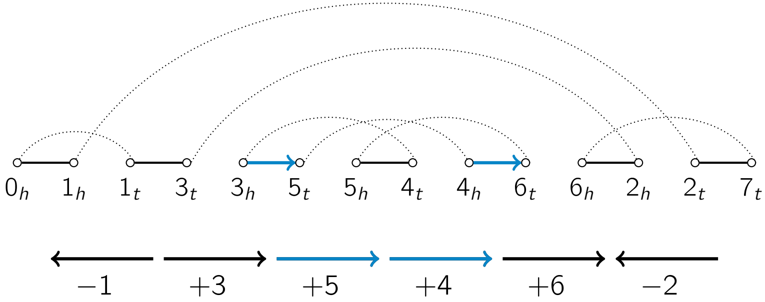


Type I - Same Cycle, divergent

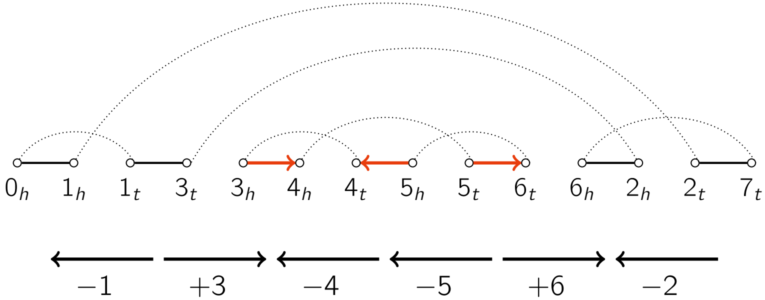


This reversal increases the number of cycles by one, $\Delta C = +1$.

Type II - Same Cycle, convergent

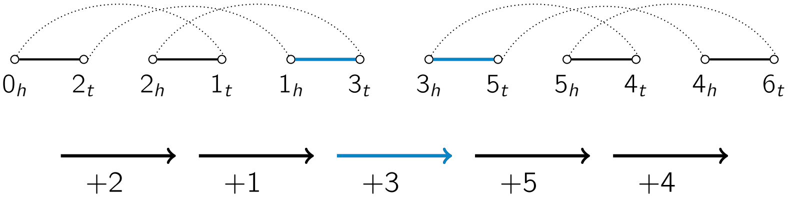


Type II - Same Cycle, convergent

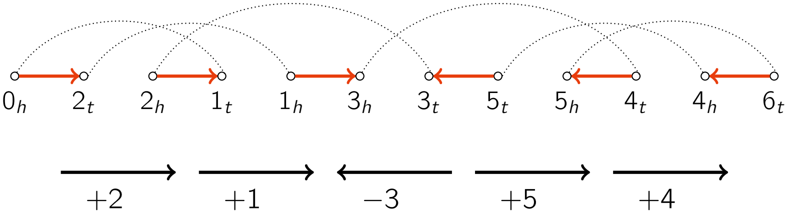


Does not change number of cycles ($\Delta C = 0$), but the cycle is **oriented**.

Type III - Different Cycles



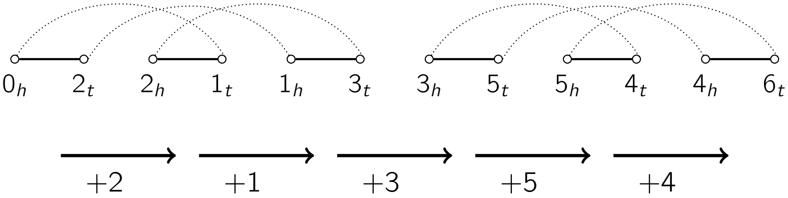
Type III - Different Cycles



Merges the two cycles, decreasing the number of cycles by one ($\Delta C = -1$), but the new cycle is **oriented**.

Extra Operations

- How many extra operations do we need to sort unoriented components?




```
java InversionVisualisation L2/2unoriented.txt
```

Extra Operations

- Applying one reversal in each cycle, orients both cycles, with 2 extra operations:

$$d = N - C + 2$$

- Applying one reversal merging both cycles, creates one new **oriented** cycle. Only one operation, but also one less cycle:

$$d = N - (C - 1) + 1 = N - C + 2$$

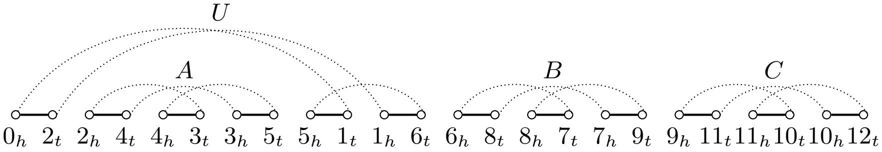
- In both cases, 2 extra operations. Does this mean that

$$d = N - C + K$$

where K is the number of unoriented components? **Almost...**

Definitions

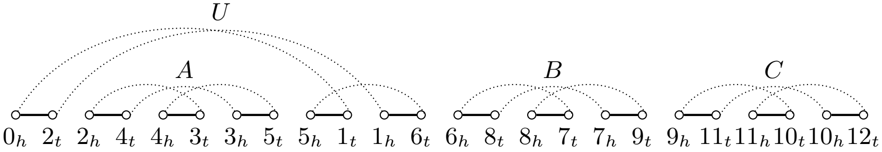
- A Component U **separates** two other components A and B if any edge from a vertex from A to B would cross an edge of U.



- U separates A and B. (Also A and C).

Definitions

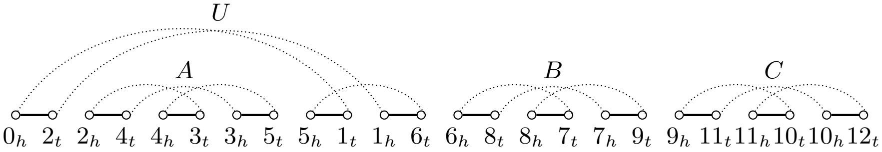
- A **hurdle** is an unoriented component that does **not** separate other two unoriented components.



- A,B and C are hurdles.

Definitions

- A **super-hurdle** is a hurdle that, when removed, causes the creation of a new hurdle.



- *A* is a super-hurdle. *B* and *C* are called *simple* hurdles.

- Why are these definitions important? Because except for one very rare special case, we have

$$d = N - C + H$$

where H is the number of hurdles.

BP Graph – Component Types

- Component
  - Oriented
  - Unoriented
    - Non-Hurdle
    - Hurdle
      - Simple
      - Super-Hurdle

Reversal Types

- **Type I**: **Oriented Reversal**: $\Delta C = +1$.
  - Edges on same cycle, divergent.
- **Type II**: **Hurdle Cutting**: $\Delta C = 0$, $\Delta H = -1$.
  - Edges on same cycle (hurdle), convergent.
- **Type III**: **Hurdle Merging**: $\Delta C = -1$, $\Delta H = -2$.
  - Edges on different cycles (hurdles).

Separating component

- Why a separating component is not a Hurdle?
- Because it can be oriented by a **Hurdle Merging** of two hurdles that is separates.

```
java InversionVisualisation L2/sep-hur-example.txt
```

Super-hurdles: Problems might occur

- Cutting a super-hurdle is bad.
- Merging hurdles that are separated from a super-hurdle can cause the separating component to become a hurdle.

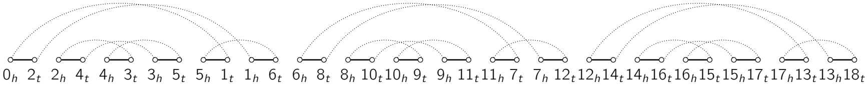
```
java InversionVisualisation L2/sep-hur-example.txt
```

Super-hurdles: Problems might occur

- How to avoid those problems?
- When there is an odd # of hurdles, cut a simple hurdle.
- When there is an even # of hurdles, merge opposite hurdles.
- Can we always do that? No... meet the **fortress**!

Fortresses

- A **fortress** is a permutation that has an odd number of hurdles, and all are super-hurdles.



In this kind of permutation, there is no way to avoid an **extra operation**, a hurdle cut that creates a new hurdle.

```
java InversionVisualisation L2/fortress.txt
```

Reversal Distance - Complete equation

Theorem (Reversal Distance, HP 95)

The reversal distance of a permutation π is given by

$$d(\pi) = N - C + H + F$$

where:

- *N is the number of genes*
- *C is the number of cycles in $BP(\pi)$*
- *H the number of hurdles in $BP(\pi)$*
- $F = \begin{cases} 1, & \pi \text{ is a fortress} \\ 0, & \text{otherwise} \end{cases}$

Reversal Distance - Complete Algorithm

```
procedure ReversalSort(π)
    while π ≠ identity do
        if ∃ oriented component in BP(π) then
            → Apply a max score oriented reversal – Type I
        else if even # of hurdles then
            → Apply a Hurdle Merging on opposite hurdles – Type III
        else if ∃ simple hurdle then
            → Apply a Hurdle Cutting on a simple hurdle – Type II
        else
            → Merge any two super hurdles (Fortress)
        end if
    end while
end procedure
```