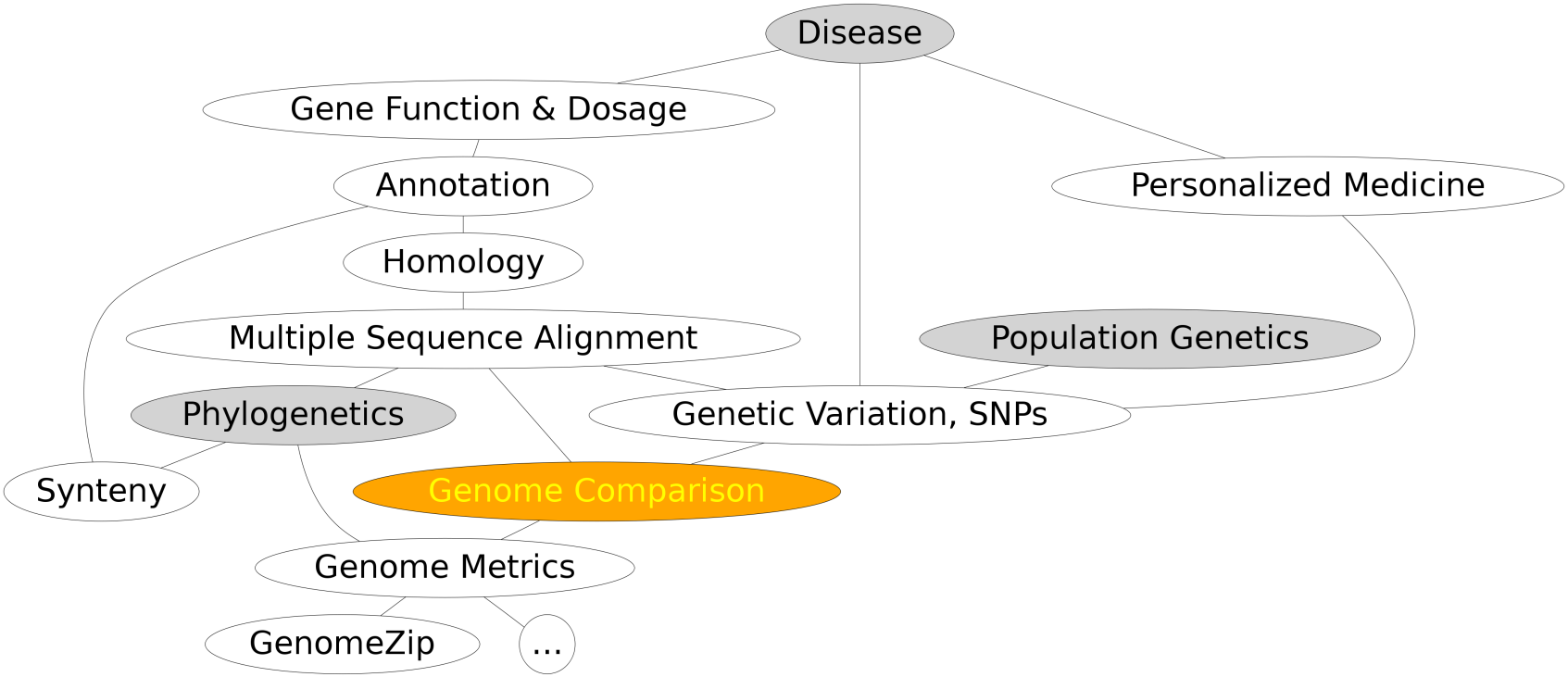
Whole Genome Comparison: Colinear Alignment

Felix Heeger, Max Homilius, Ivan Kel, Sabrina Krakau,
Svenja Specovius, John Wiedenhoeft

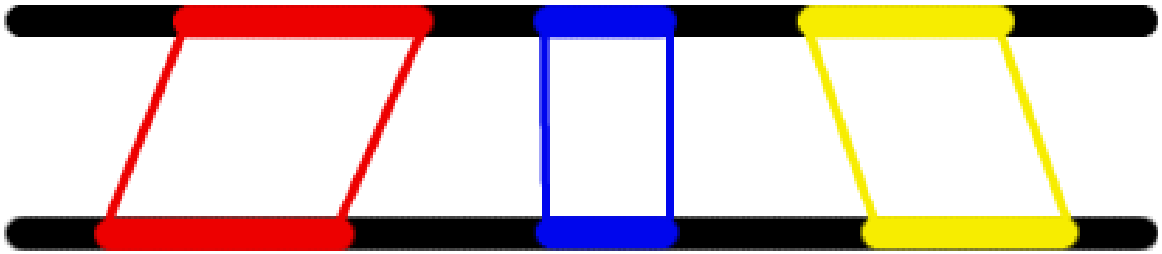
May 10, 2010

The Big Picture



Colinear Alignment

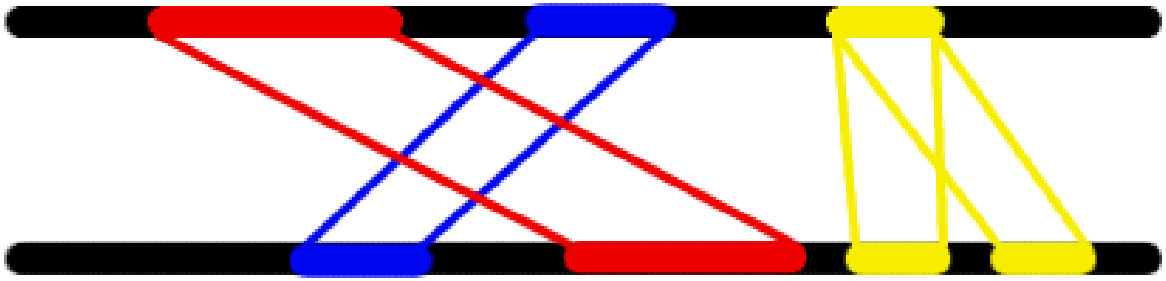
Colinear Alignment: Containing elements that are arranged in the same linear order in all sequences.



colinear alignment

Non-Colinear Alignment

Non-Colinear Alignment: Containing elements that are arranged in some non-linear order.



non-colinear alignment

Colinear vs Non-Colinear

Why use colinear alignment at all?

Colinear vs Non-Colinear

Why use colinear alignment at all? $\Rightarrow$ It's faster!

BUT Careful!

Colinear vs Non-Colinear

Why use colinear alignment at all? $\Rightarrow$ It's faster!

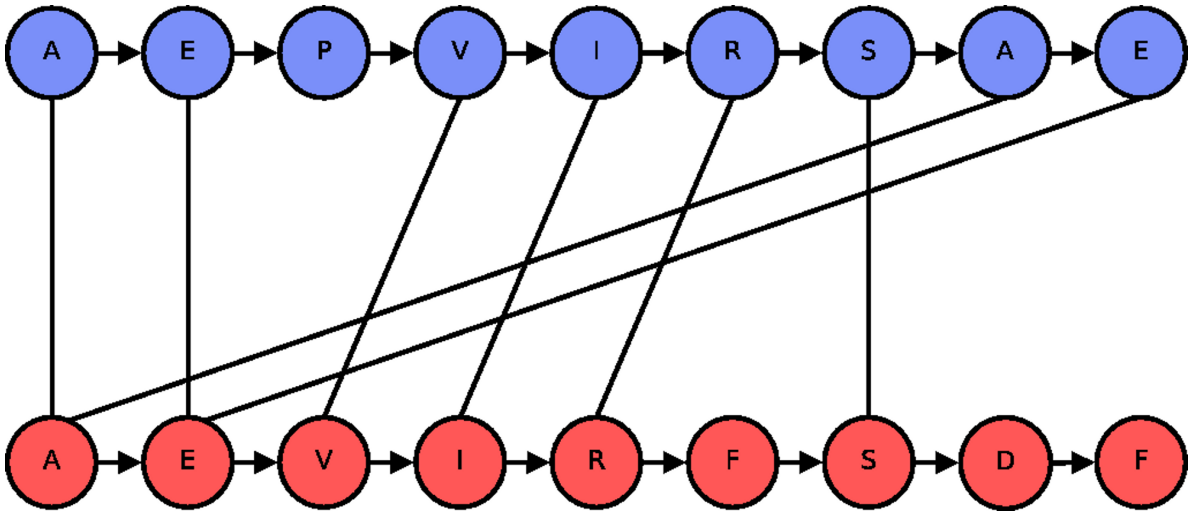
BUT Careful!

Watch out for:

- Translocations
- Duplications
- Inversions

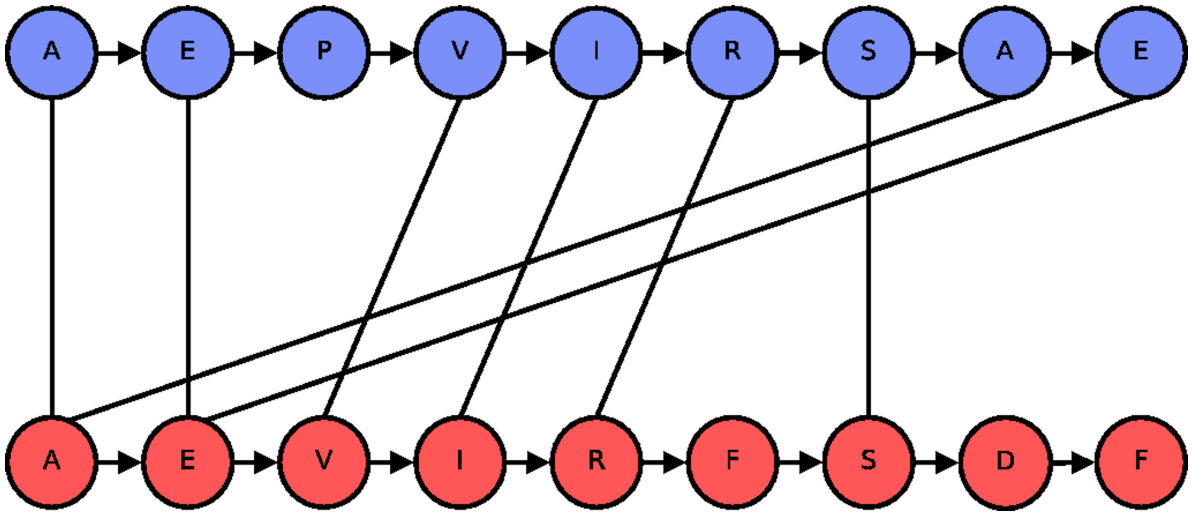
Graph Representation of Alignments

Alignment Graph of Single Characters



Graph Representation of Alignments

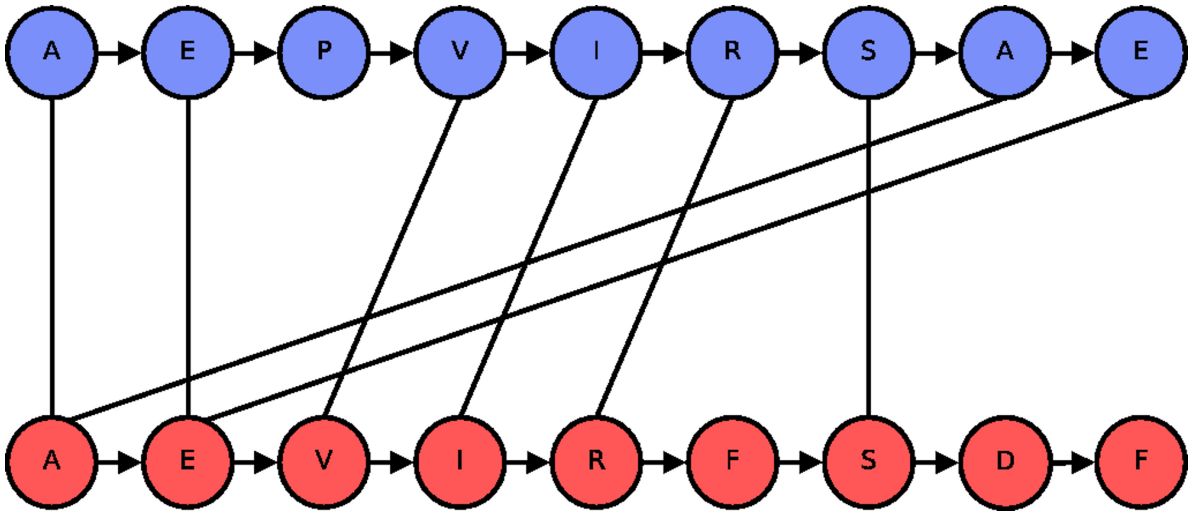
Alignment Graph of Single Characters



- sequences are represented by (directed) chains of nodes

Graph Representation of Alignments

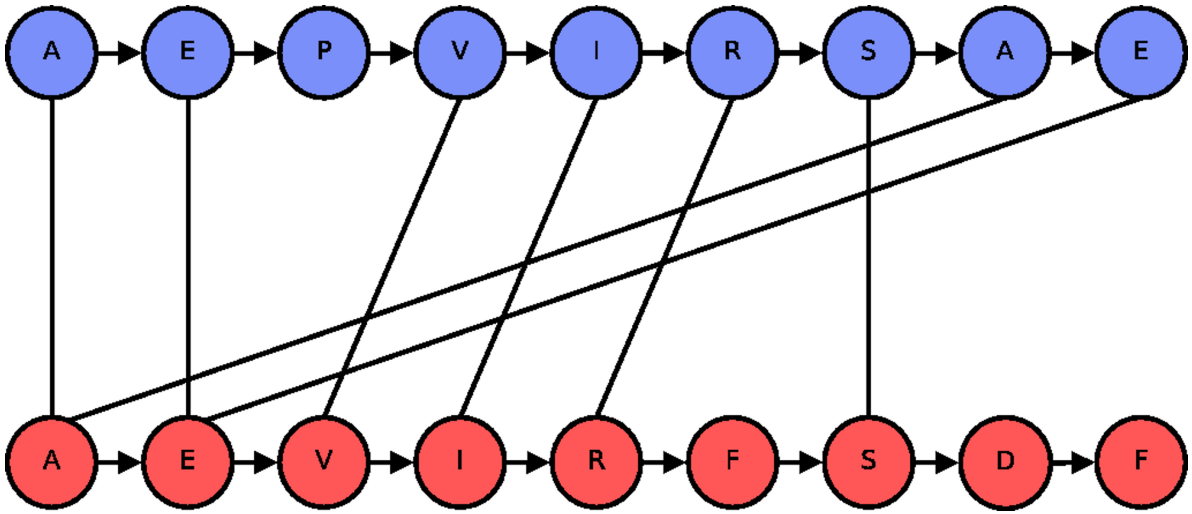
Alignment Graph of Single Characters



- sequences are represented by (directed) chains of nodes
- matches are represented by (undirected) edges

Graph Representation of Alignments

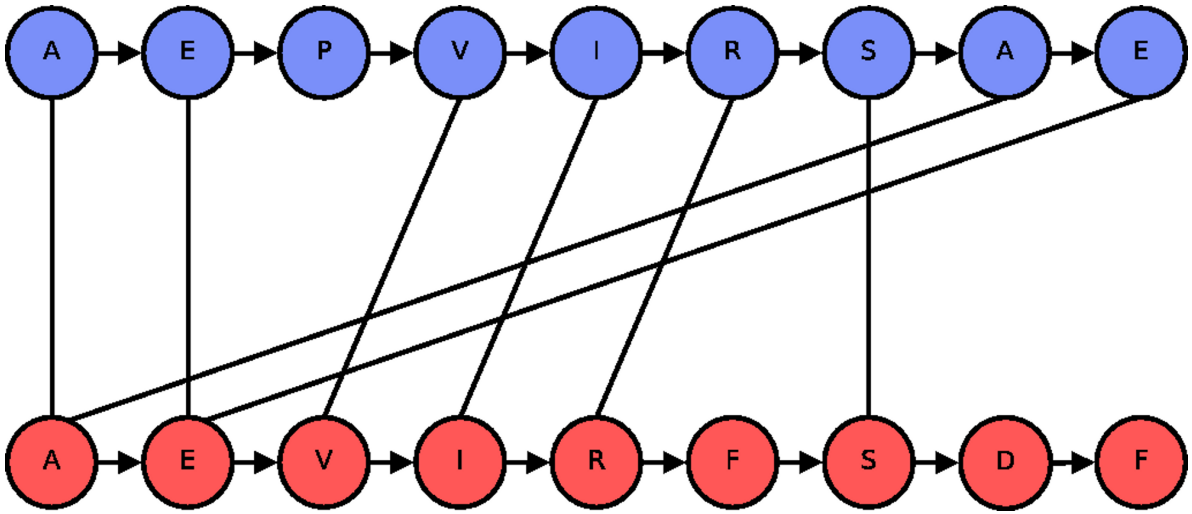
Alignment Graph of Single Characters



- sequences are represented by (directed) chains of nodes
- matches are represented by (undirected) edges
- some edges may be inconsistent

Graph Representation of Alignments

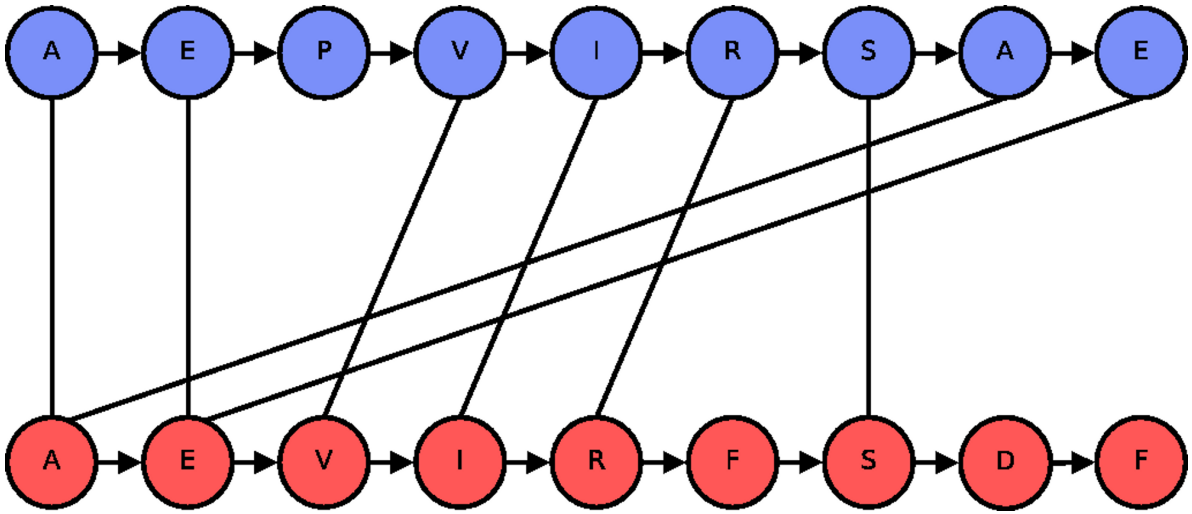
Alignment Graph of Single Characters



- sequences are represented by (directed) chains of nodes
- matches are represented by (undirected) edges
- some edges may be inconsistent
- *trace*: set of (consistent) realized edges in the alignment graph

Graph Representation of Alignments

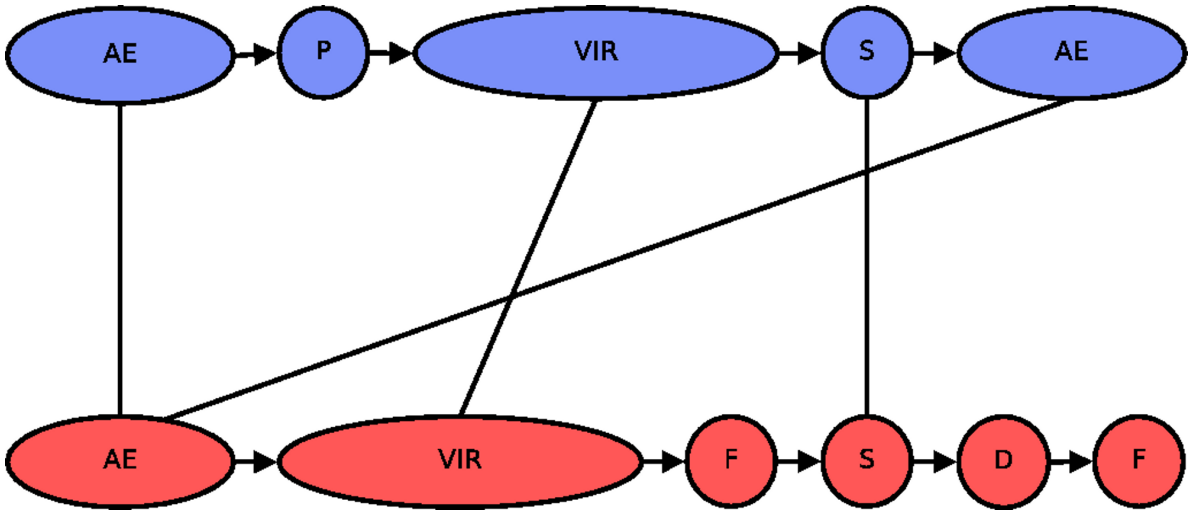
Alignment Graph of Single Characters



- sequences are represented by (directed) chains of nodes
- matches are represented by (undirected) edges
- some edges may be inconsistent
- *trace*: set of (consistent) realized edges in the alignment graph
- finding the optimal set of consistent edges: Maximum-weight trace problem (MWT)

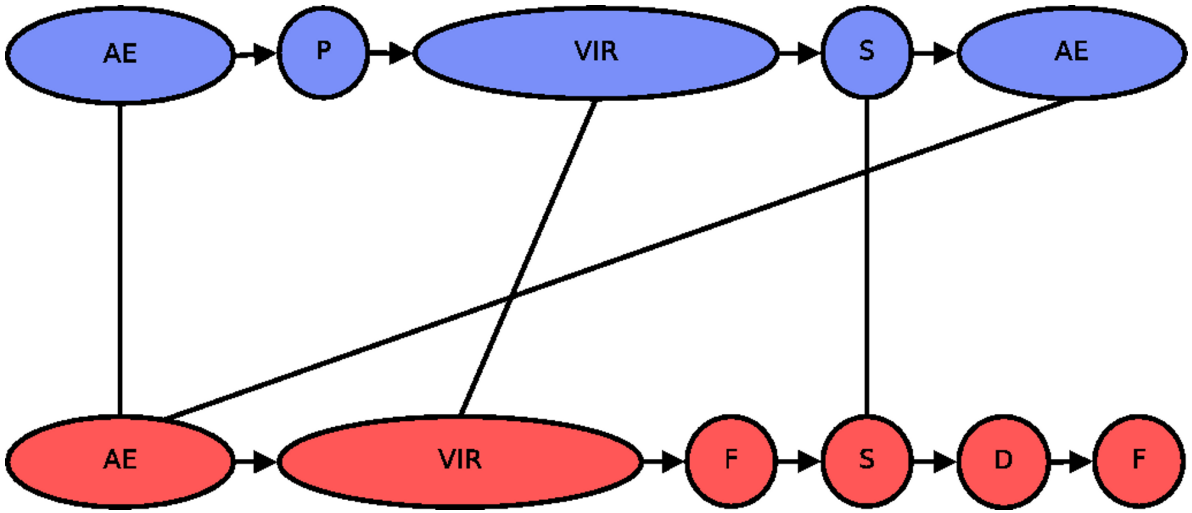
Graph Representation of Alignments

Alignment Graph of "Segments"



Graph Representation of Alignments

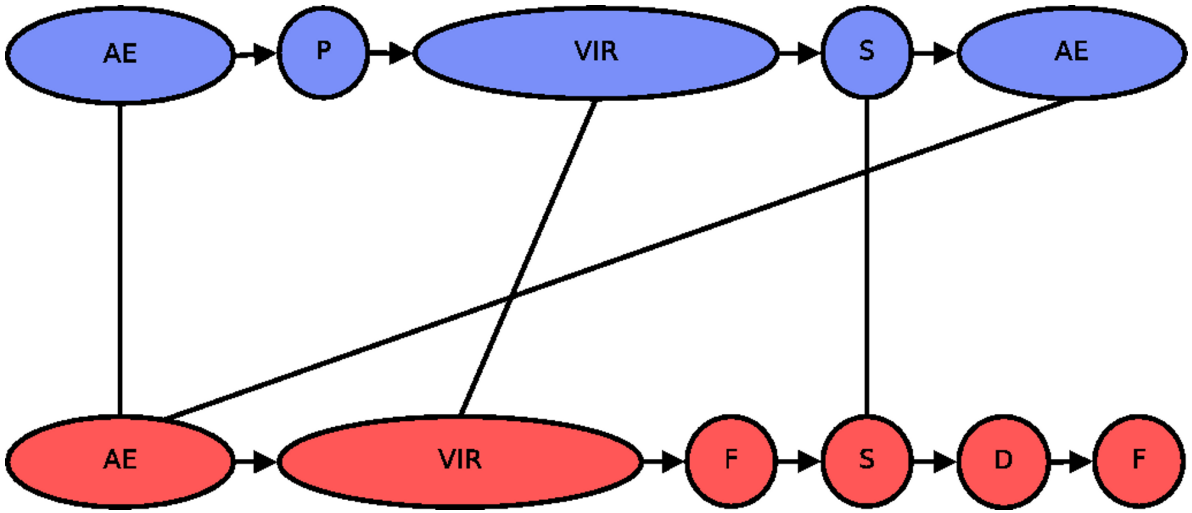
Alignment Graph of "Segments"



- combine consecutive matching characters to a segment

Graph Representation of Alignments

Alignment Graph of "Segments"



- combine consecutive matching characters to a segment
- reduction of edges $\longrightarrow$ smaller input

Solving the Maximum-Weight Trace Problem

exact:

- integer linear programming
- n-dimensional dynamic programming

  $\longrightarrow$ NP-hard

Solving the Maximum-Weight Trace Problem

exact:

- integer linear programming
- n-dimensional dynamic programming

  $\longrightarrow$ NP-hard

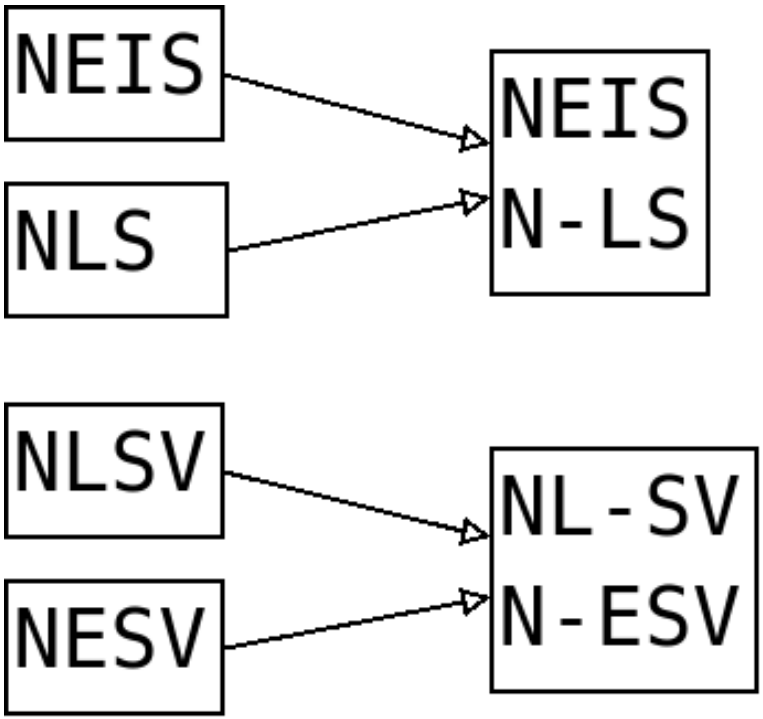
heuristical:

- progressive alignment

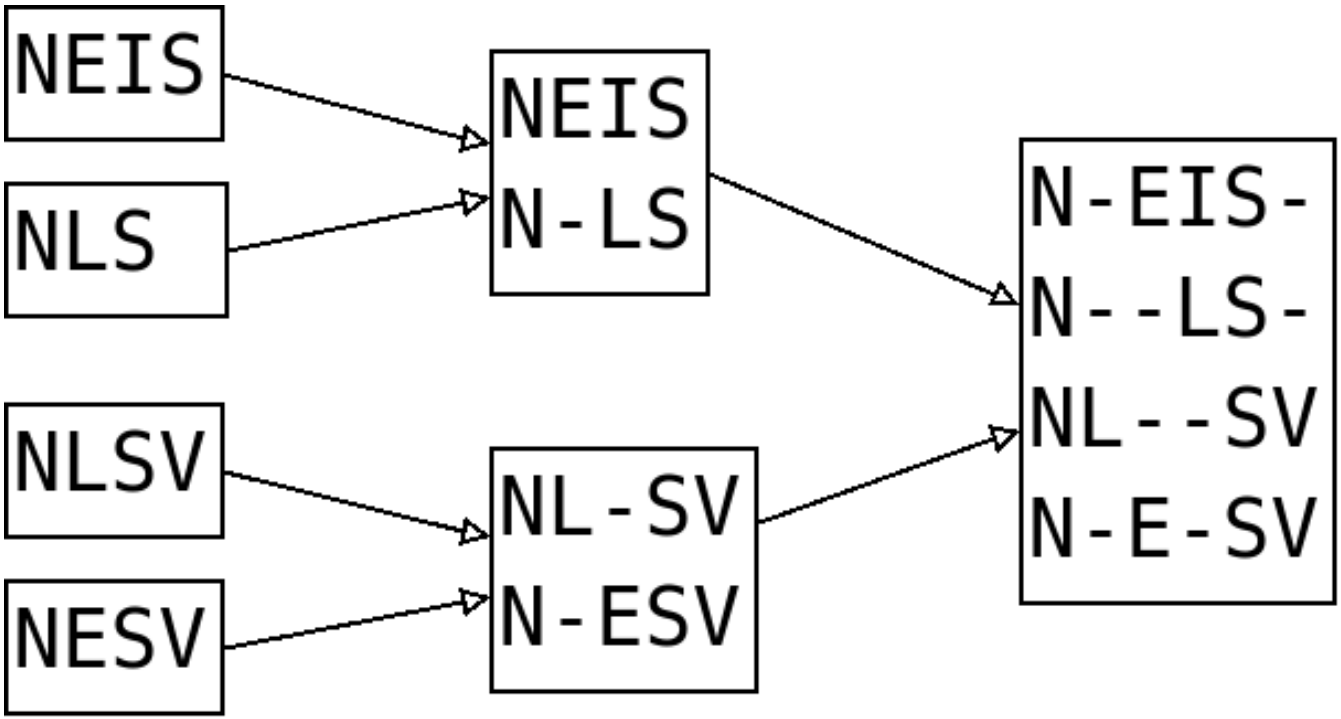
  $\longrightarrow$ problem of local optima

Progressive Alignment

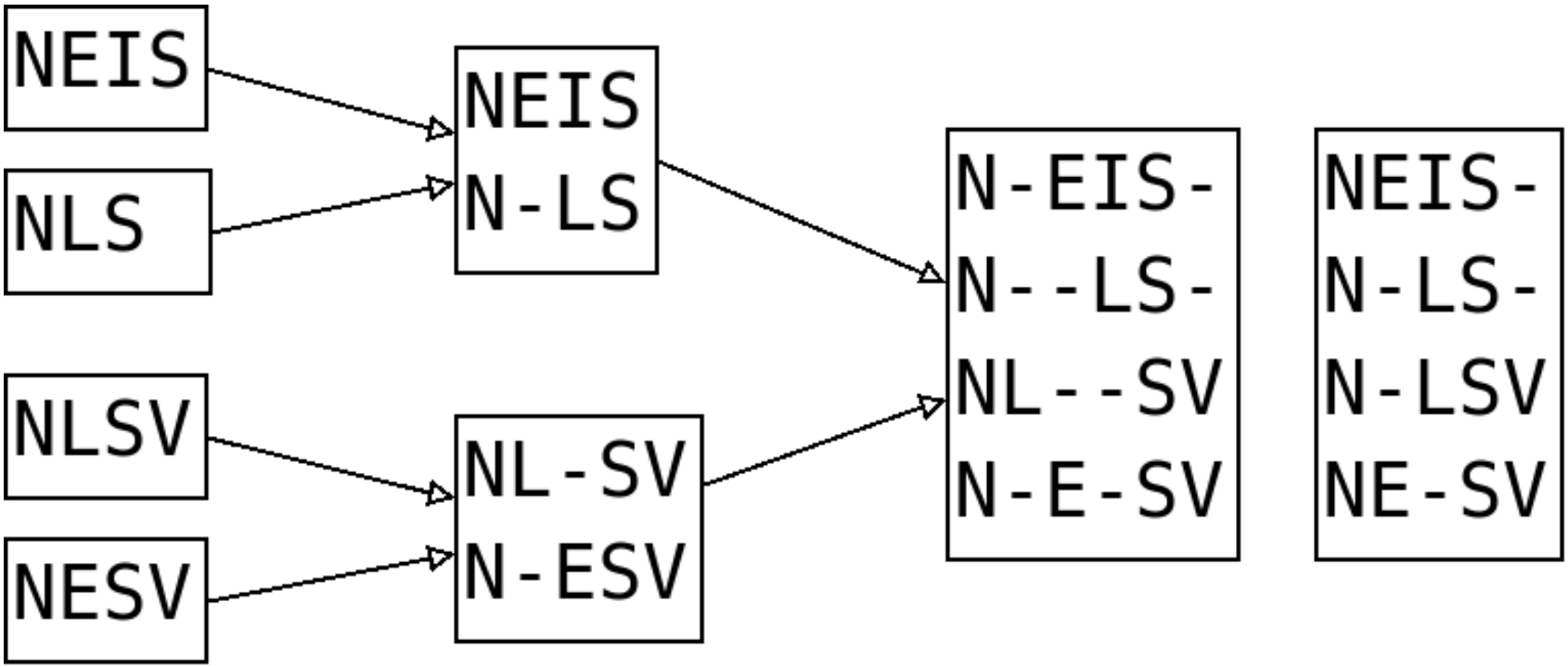
Progressive Alignment



Progressive Alignment



Progressive Alignment



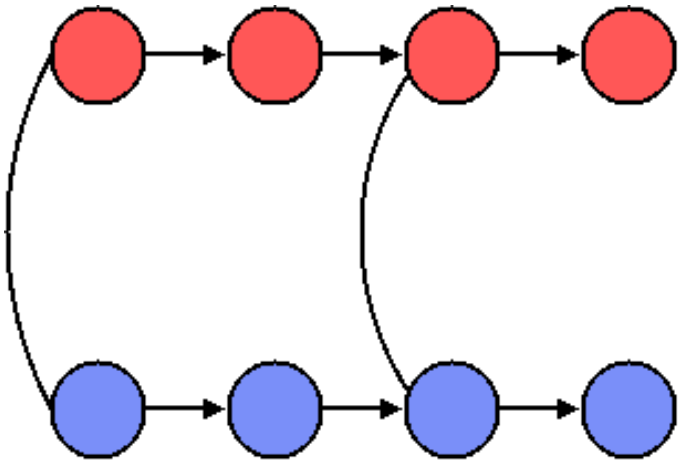
Progressive Alignment

Consistency

1. re-scoring of pairwise alignment to integrate information from other alignments
2. use of individual match scores for progressive alignment

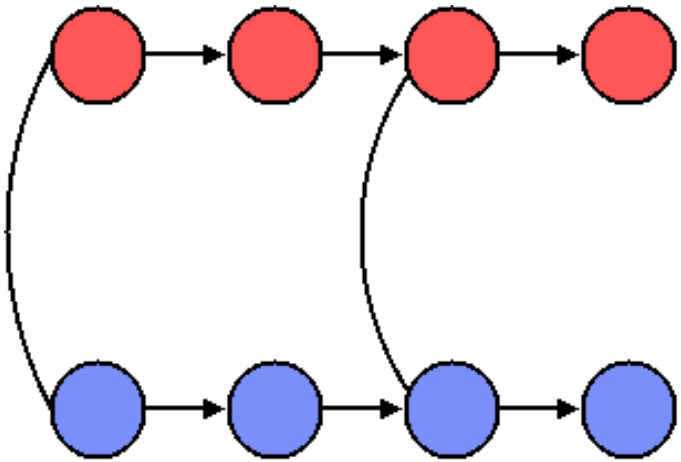
Progressive Alignment

Consistency



Progressive Alignment

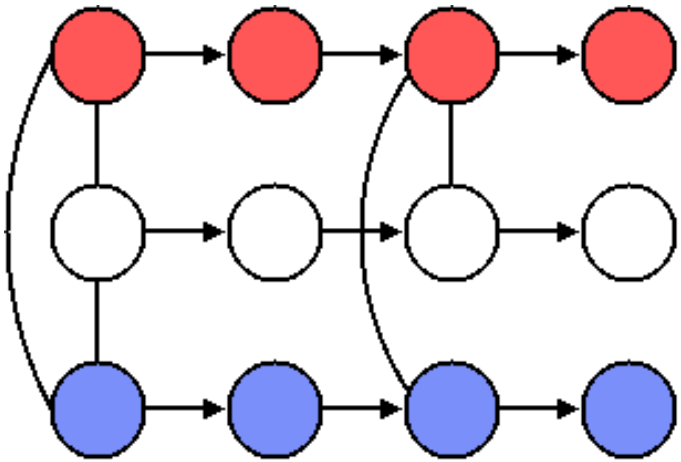
Consistency



- find a path "through" one other alignment to the matching node

Progressive Alignment

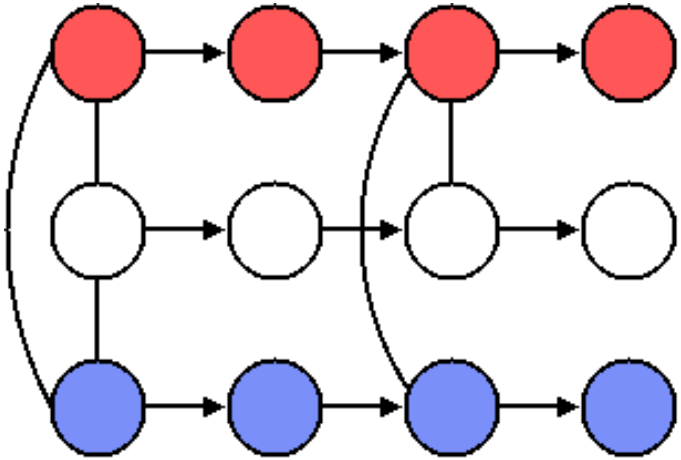
Consistency



- find a path "through" one other alignment to the matching node

Progressive Alignment

Consistency



- find a path "through" one other alignment to the matching node
- the triplet of alignments is then consistent for this match

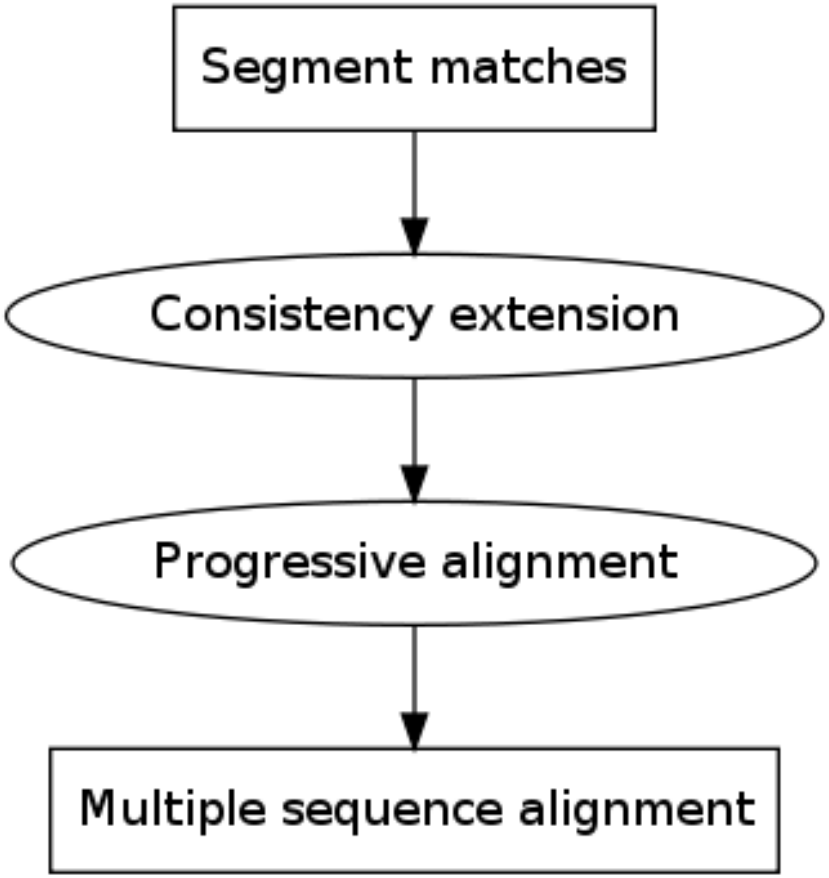
Progressive Alignment

Consistency

for all nodes n **do**
  for all pairs of nodes i,j adjacent to n **do**
    if i and j adjacent **then**
      w(e_{ij})+=min(w(e_{ni}),w(e_{nj}))
    else
      add e_{ij} to E
      w(e_{ij})=min(w(e_{ni}),w(e_{nj}))
    end if
  end for
end for

Segment Based Multiple Sequence Alignment

using consistency



Segments

How to Obtain Segments?

- retrieving the segments from the alignment graph takes too long

Segments

How to Obtain Segments?

- retrieving the segments from the alignment graph takes too long
- we can get approximate segments by local alignment methods

Segments

How to Obtain Segments?

- retrieving the segments from the alignment graph takes too long
- we can get approximate segments by local alignment methods
  - dynamic programming (for short sequences)

Segments

How to Obtain Segments?

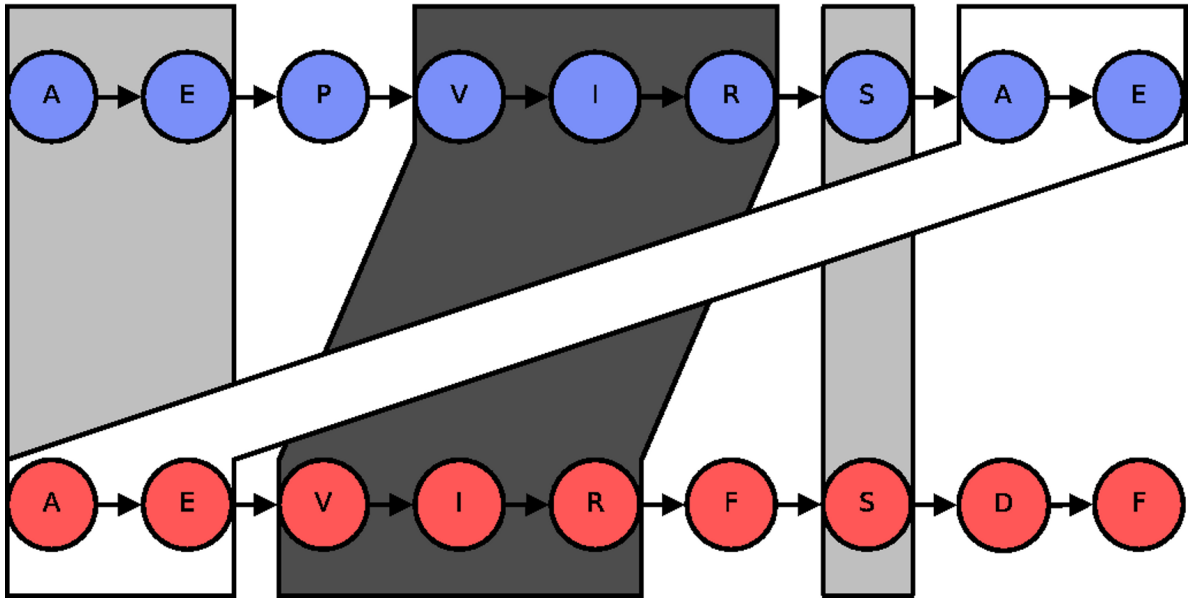
- retrieving the segments from the alignment graph takes too long
- we can get approximate segments by local alignment methods
  - dynamic programming (for short sequences)
  - Suffixarrays (for exact matches)

Segments

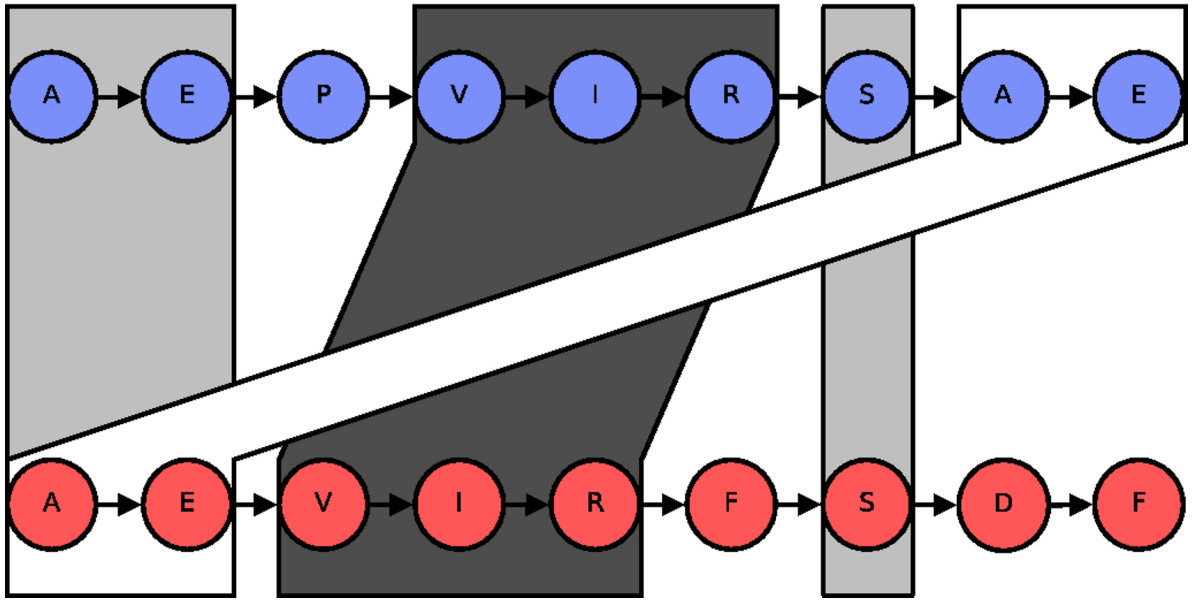
How to Obtain Segments?

- retrieving the segments from the alignment graph takes too long
- we can get approximate segments by local alignment methods
  - dynamic programming (for short sequences)
  - Suffixarrays (for exact matches)
  - Blast

Segments

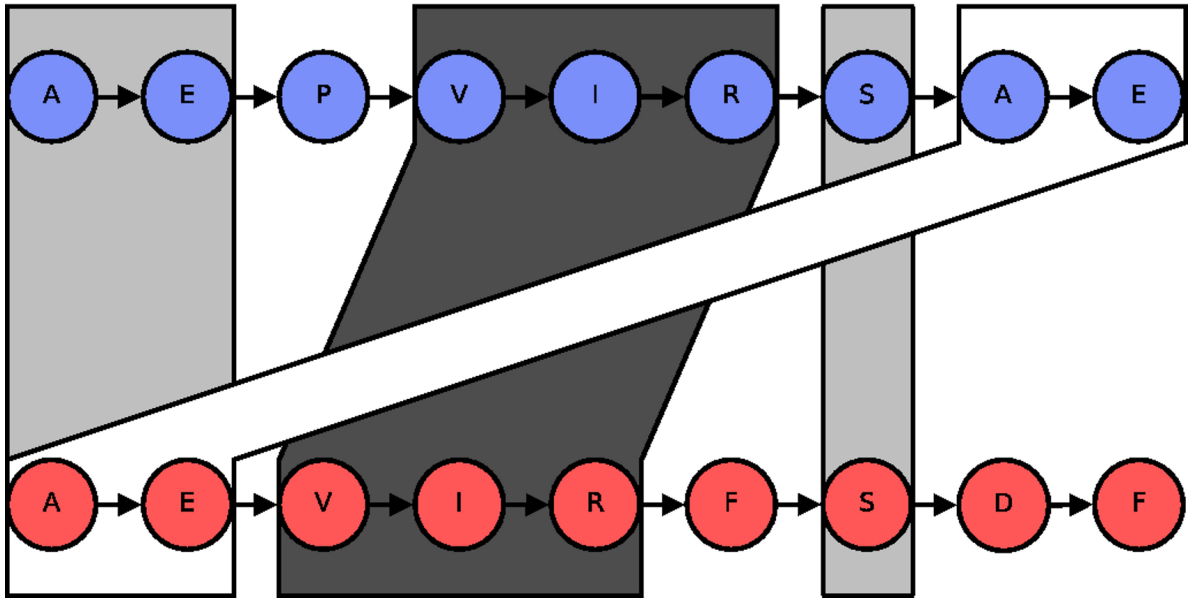


Segments



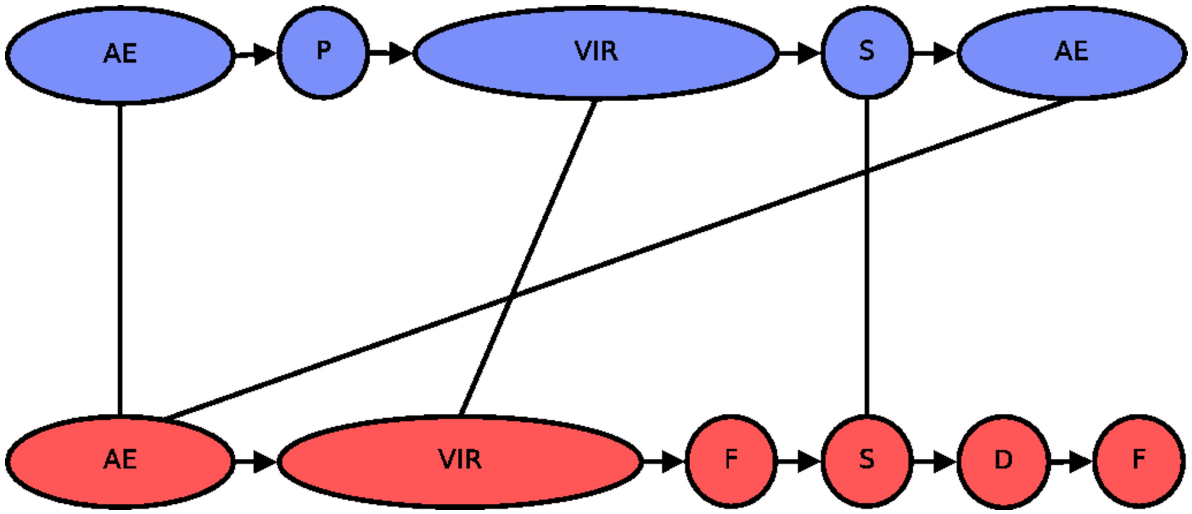
- combine each matched region on one sequence to one node

Segments



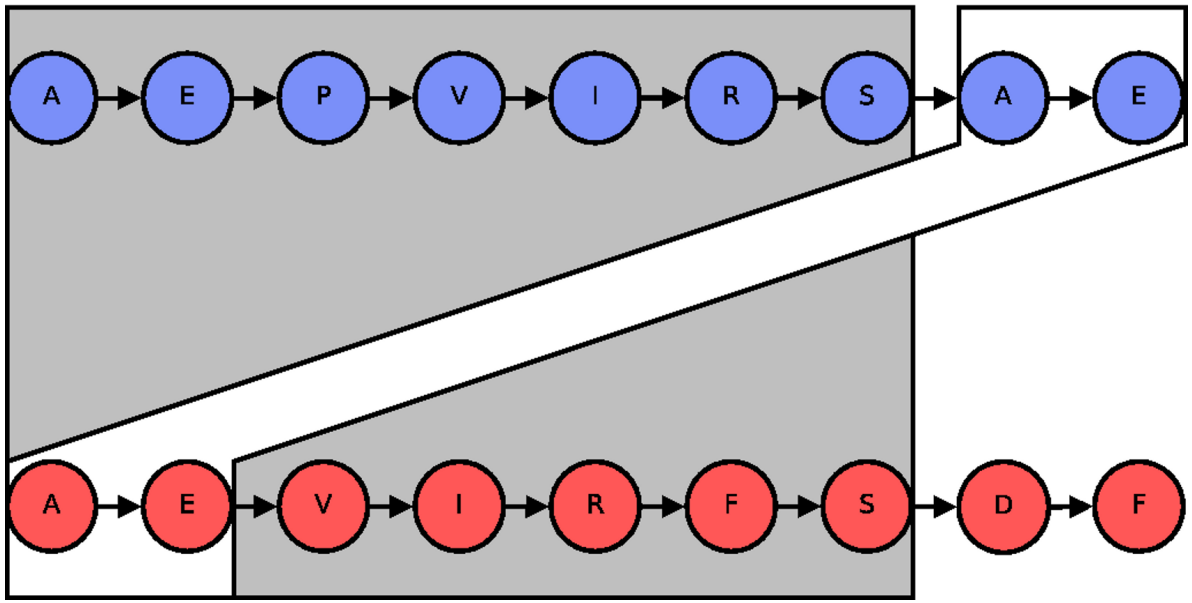
- combine each matched region on one sequence to one node
- connect the matched regions on different sequences with an edge

Segments

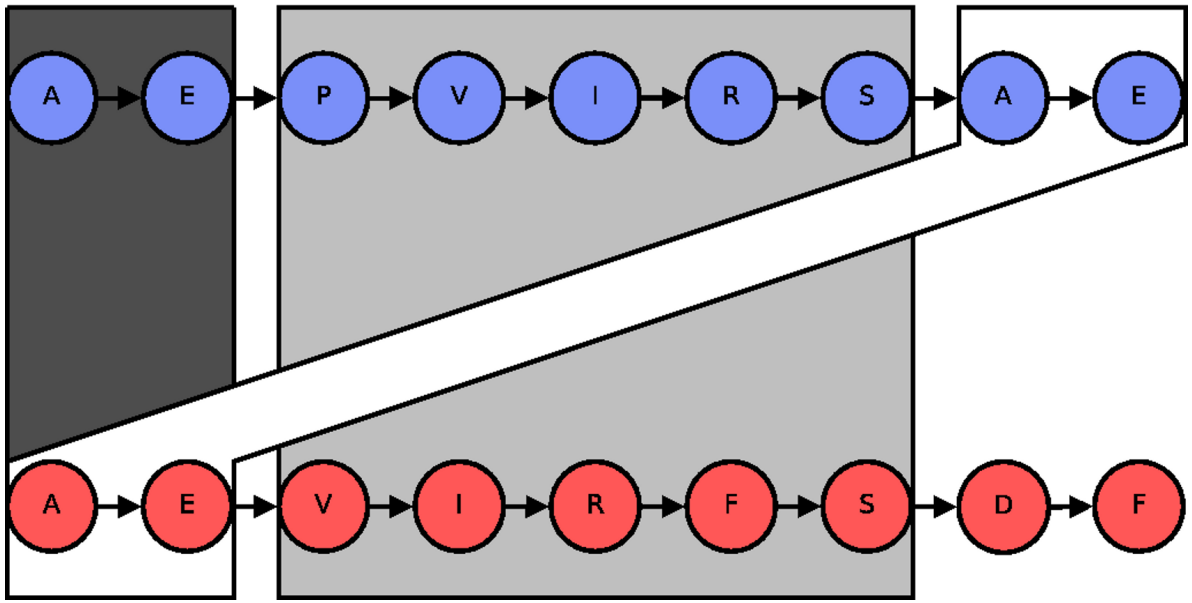


- combine each matched region on one sequence to one node
- connect the matched regions on different sequences with an edge

Overlaping Segments



Refined Segments



Segment Match Refinement Algorithm

Notation

- $\mathcal{S} = \{S^0, S^1, \ldots, S^{n-1}\}$ a set of n sequences

Segment Match Refinement Algorithm

Notation

- $\mathcal{S} = \{S^0, S^1, \ldots, S^{n-1}\}$ a set of n sequences
- $S^i_{u,v} = s^i_u s^i_{u+1} \ldots s^i_{v-1}$ a *segment* in sequence i

Segment Match Refinement Algorithm

Notation

- $\mathcal{S} = \left\{S^0, S^1, \ldots, S^{n-1}\right\}$ a set of n sequences
- $S^i_{u,v} = s^i_u s^i_{u+1} \ldots s^i_{v-1}$ a *segment* in sequence i
- $M = \left(S^i_{u,v}, S^j_{x,y}\right)$ a *segment match*

Segment Match Refinement Algorithm

Notation

- $\mathcal{S} = \left\{S^0, S^1, \ldots, S^{n-1}\right\}$ a set of n sequences
- $S^i_{u,v} = s^i_u s^i_{u+1} \ldots s^i_{v-1}$ a *segment* in sequence i
- $M = \left(S^i_{u,v}, S^j_{x,y}\right)$ a *segment match*
- $\mathcal{M} = \left\{M^0, M^1, \ldots, M^{m-1}\right\}$ a set of m segment matches

Segment Match Refinement Algorithm

Definitions

Refinement

$\mathcal{M}_* = \left\{ M_*^0, M_*^1, \ldots, M_*^{m'-1} \right\}$ is a *refinement* of $\mathcal{M}$, if every segment $S \in \mathcal{M}$ is tiled by a subset of $\mathcal{M}_*$

Segment Match Refinement Algorithm

Definitions

Refinement

$\mathcal{M}_* = \left\{ M_*^0, M_*^1, \ldots, M_*^{m'-1} \right\}$ is a *refinement* of $\mathcal{M}$, if every segment $S \in \mathcal{M}$ is tiled by a subset of $\mathcal{M}_*$

Resolved Refinement

a refinement $\mathcal{M}_*$ is *resolved*, if every two segments S^i and S^j in $\mathcal{M}_*$ are either identical or disjoint, i.e. they do not partially overlap

Segment Match Refinement Algorithm

Definitions

Refinement

$\mathcal{M}_* = \left\{ M_*^0, M_*^1, \ldots, M_*^{m'-1} \right\}$ is a *refinement* of $\mathcal{M}$, if every segment $S \in \mathcal{M}$ is tiled by a subset of $\mathcal{M}_*$

Resolved Refinement

a refinement $\mathcal{M}_*$ is *resolved*, if every two segments S^i and S^j in $\mathcal{M}_*$ are either identical or disjoint, i.e. they do not partially overlap

Minimal Resolved Refinement

a resolved refinement of minimal cardinality is a *minimal resolved refinement*

Segment Match Refinement Algorithm

Input/Output

Input

a set $\mathcal{M}$ of segment matches

Output

a set $\mathcal{M}^*$, which is the minimal resolved refinement of $\mathcal{M}$

Segment Match Refinement Algorithm

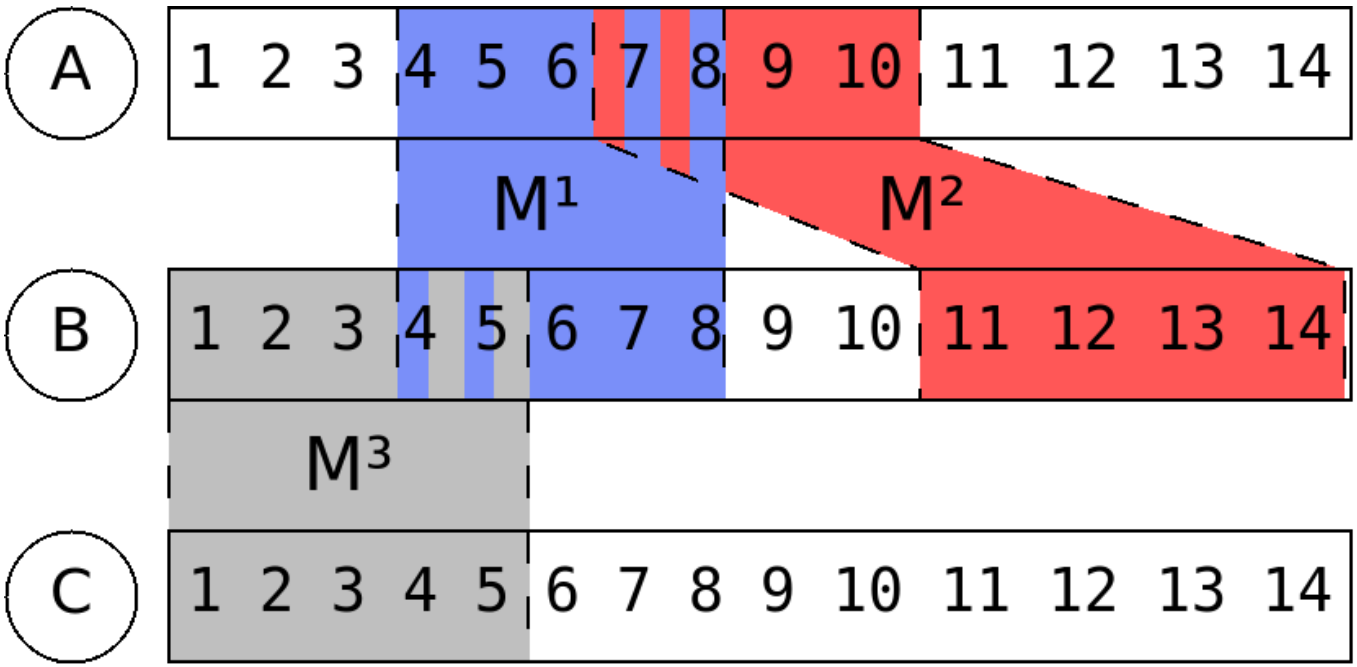
Pseudocode

def cut(a):
  $\mathcal{L} = \{M^i = (S^i_{u,v}, S^j_{x,y}) \,|\, u < a < v\}$
  for $L \in \mathcal{L}$:
    b = match postion of a given by L
    if $b \notin V^j$:
      insert b into V^j
      insert edge (i, a, b) into E
      cut(b)
def refine($\mathcal{M}$):
  V^i = boundary positions of segments in sequence i
  for $M^i = (S^i_{u,v}, S^j_{x,y}) \in \mathcal{M}$:
    insert edge (i,u,x) and (i,v,y) into E
    for boundary position w of M^i:
      cut(w)
  lexicographically order E

Segment Match Refinement Algorithm

- build a n-partite graph with node sets $V^0, \ldots, V^{n-1}$ and edges E (with code above)
- the Graph (V,E) defines "cuts" in $\mathcal{M}$
- consecutive pairs of edges (i,a,b) and (j,x,y) in E define a segment match in M_*, if $i = j$

Example



$M^1 = (A_{4,9}, B_{4,9})$
$M^2 = (A_{7,11}, B_{11,15})$
$M^3 = (B_{1,6}, C_{1,6})$

References

Halpern, A. L., Huson, D. H., and Reinert, K. (2002).

Algorithms in Bioinformatics, chapter Segment Match Refinement and Applications, pages 126–139. Springer Berlin / Heidelberg.

Notredame, C., Higgins, D. G., and Heringa, J. (2000).

T-coffee: A novel method for fast and accurate multiple sequence alignment.
J Mol Biol, 302(1):205–217.

Rausch, T., Emde, A.-K., Weese, D., Döring, A., Notredame, C., and Reinert, K. (2008).

Segment-based multiple sequence alignment.
Bioinformatics, 24(16):i187–i192.

Reinert, K. (2002).

Segment match refinement and applications.

Reinert, K. (2009).

Multiple match refinement and t-coffee.