Localized genome assembly from reads to scaffolds: practical traversal of the paired string graph

Rayan Chikhi, Dominique Lavenier

ENS Cachan/Irisa, France

WABI 2011

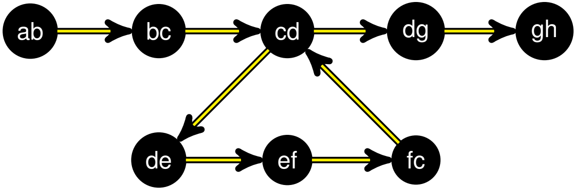
Outline

INTRODUCTION

Assembly input : reads R are substrings of an unknown genome g.
Two reads (r, r') k-overlap if a suffix of r matches a suffix of r' over exactly k characters.

String graph $SG^k(R)$

- Directed graph
- $V = R$
- $E = \{(r, r') \in R^2 \text{ s.t. } r\ k\text{-overlaps } r'\}$
- Transitively reduced, contained reads removed



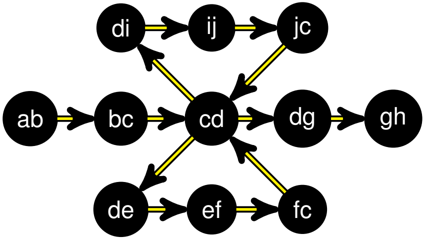
STRING GRAPH ASSEMBLY

Assembly problem

- Find a path which visits each node of $SG^k(R)$ at least once (generalized Hamiltonian Path), minimizing path-string length. [Nagarajan 09]
- NP-hard, reduces from shortest common superstring. [Medvedev 07]

Practically :

repeats $\rightarrow$ many min-cost solutions : (e.g. *abcdefcdijcdgh*, *abcdijcdefcdgh*)



STRING GRAPH ASSEMBLY

Assembly problem

- Find a path which visits each node of $SG^k(R)$ at least once (generalized Hamiltonian Path), minimizing path-string length. [Nagarajan 09]
- NP-hard, reduces from shortest common superstring. [Medvedev 07]

Practically :

repeats $\rightarrow$ many min-cost solutions : (e.g. *abcdefcdijcdgh*, *abcdijcdefcdgh*)

imperfect coverage $\rightarrow$ several unconnected components

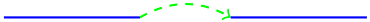
Heuristics

- Output set of linear paths (contigs) from the string graph

PAIRED READS IN ASSEMBLY

Paired reads

- Set of reads pairs $(r_1, r_2) \in (R_1, R_2)$ such that in genome, $G = ..r_1 s r_2..$, $|s|$ is known

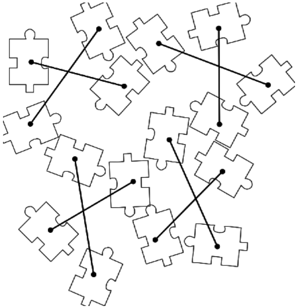


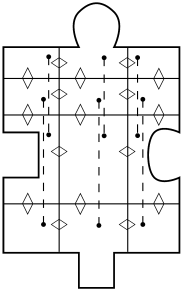
- Reads assembly is often related to jigsaw puzzles.
- With paired reads data, a paired jigsaw puzzle can be defined.

PAIRED READS IN ASSEMBLY

Paired jigsaw problem

- Solve the puzzle with pairs of jigsaw pieces linked by a string, string has to be tightened in the solution.
- NP-complete, graphical reduction (right) to classical jigsaw.





PAIRED READS IN ASSEMBLY

Scaffolding problem

- Construct a bi-directed contig graph
- $V = \{contigs\}$
- $E = \{(c_1, c_2) \text{ s.t. } |\{(r_1, r_2), r_1 \in c_1, r_2 \in c_2\}| \geq t\}$
- Find an ordering of contigs (scaffold) that contains a maximal number of valid links. [Gao 11]

Scaffolding requires a complete set of contigs.
Can paired information be used at read-level assembly ?

PAIRED READS IN ASSEMBLY

Three recent approaches to incorporate paired reads in the reads graph.

1. Paired de Bruijn graph [Medvedev 11]
   - Vertices are paired k-length substrings of $(r_1, r_2) : (k_1|k_2)$.
   - Edges : $(AG|TG) \overset{(AGC|TGT)}{\longrightarrow} (GC|GT)$
2. Mate pair graph (based on the string graph) [Donmez 11]
   - Find paths between pairs in the string graph.
   - Vertices are pairs, edges are overlapping paths between two mate-pairs.
3. Greedy with paired consistency [Nuwantha 10]

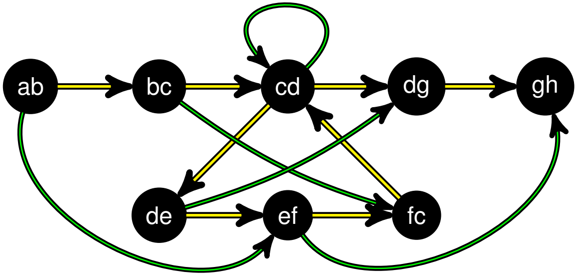
- All these approaches aim to produce contigs.
- Is it possible to use paired information to generate scaffolds locally ?

OUTLINE

PAIRED STRING GRAPHS

Paired string graph $PG^k(R_1 \times R_2)$

- Directed graph
- $V = R_1 \cup R_2$
- $E_o = \{(r, r') \in (R_1 \cup R_2)^2$ s.t. r k-overlaps $r'\}$ (overlap edges)
- $E_p = \{(r_1, r_2) \in R_1 \times R_2\}$ (paired edges)
- Transitive reduction on overlap edges only



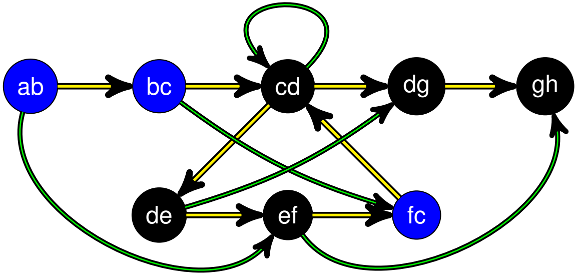
PAIRED ASSEMBLY PROBLEM

Mixed path p Succession of vertices linked by overlap or paired edges
e.g. $p = ab \rightarrow bc \dashrightarrow fc$

Path-string of p Classical path-string with gaps over paired edges
e.g. path-string(p) $= abc \diamond^2 fc$

Paired assembly problem

- Find a generalized Hamiltonian Path of $PG^k(R_1 \times R_2)$ satisfying paired edges constraints, minimizing path-string length.
- Heuristics : non-branching paths.

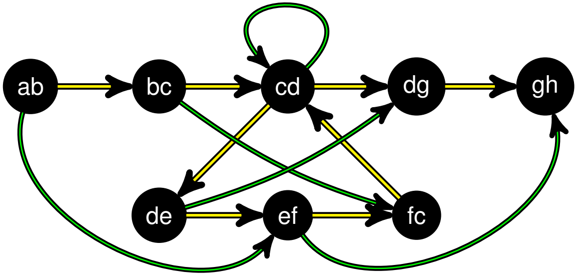


NON-BRANCHING PATHS

Assuming error-free sequencing with perfect coverage, exact insert size.

Non-branching paths *p*

- For each internal node, in-degree of 1 w.r.t path in-edge type, similarly for out-degrees.

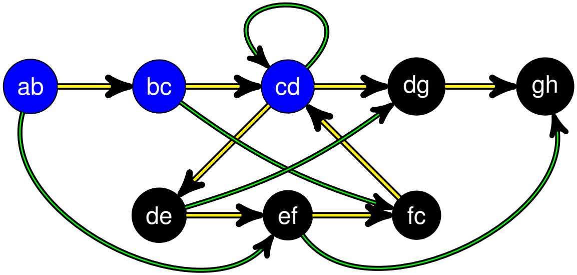


NON-BRANCHING PATHS

Assuming error-free sequencing with perfect coverage, exact insert size.

Non-branching paths p

- For each internal node, in-degree of 1 w.r.t path in-edge type, similarly for out-degrees.
- e.g. contigs : $p = ab \rightarrow bc \rightarrow cd$

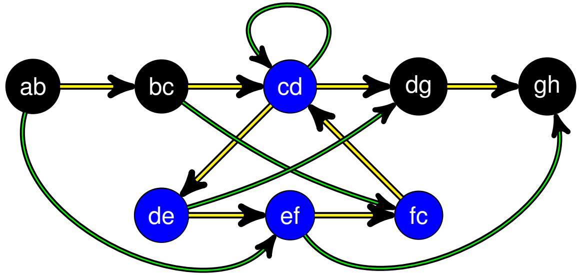


NON-BRANCHING PATHS

Assuming error-free sequencing with perfect coverage, exact insert size.

Non-branching paths *p*

- For each internal node, in-degree of 1 w.r.t path in-edge type, similarly for out-degrees.
- e.g. contigs : $p = ab \rightarrow bc \rightarrow cd$
- $p = cd \rightarrow de \rightarrow ef \rightarrow fc \rightarrow cd$

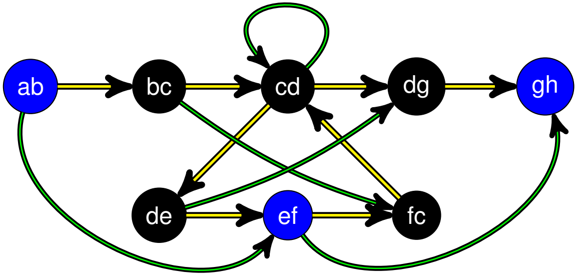


Non-branching paths

Assuming error-free sequencing with perfect coverage, exact insert size.

Non-branching paths p

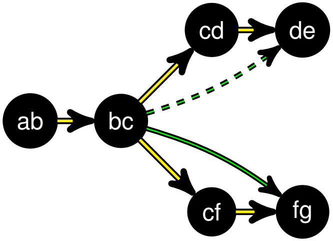
- For each internal node, in-degree of 1 w.r.t path in-edge type, similarly for out-degrees.
- e.g. contigs : $p = ab \rightarrow bc \rightarrow cd$
- $p = cd \rightarrow de \rightarrow ef \rightarrow fc \rightarrow cd$
- as well as scaffolds : $p = ab \dashrightarrow ef \dashrightarrow gh$
- in-branching and out-branching paths are also valid sequences



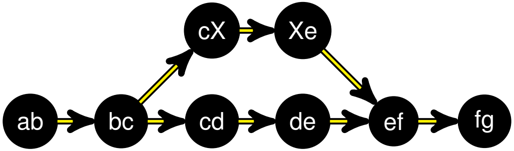
PRACTICAL CASES

Two problems with non-branching paths in actual sequencing :

1) Imperfect pairing coverage : cannot extend on the basis of a single paired edge



2) Additional overlap branching : branching due to errors, biological variations



Practical cases : 1) Undetected paired branching

We cannot trust single paired edges anymore.

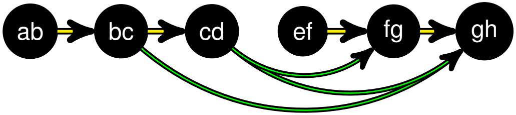
- Assume bounded insert deviation i
- Consider a simple (overlap) path p of length $2i + 1$ with central node n

Heuristic : $n \dashrightarrow n'$ can be included in a non-branching path if :

Property 1 :

The sub-graph of opposite mates of p is a simple path of central node n'.

In other words, p' is the only region that appears after p.



Practical cases : 2) Additional overlap branching

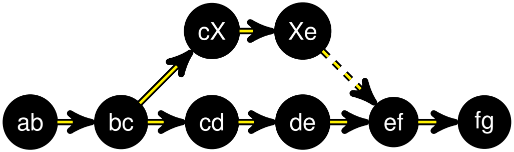
Bubble and dead-end traversal.

Variant sub-graph from node *n*

A sub-graph is a *variant sub-graph* from node n if its BFS tree has a single node of depth d.

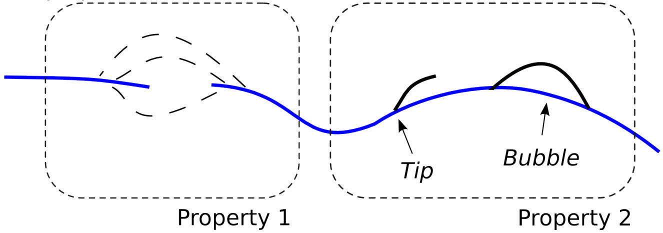
Property 2 :

$n \rightarrow n'$ can be included in a non-branching path if n, n' are part of a variant sub-graph.



Practical non-branching paths

In summary, practical non-branching paths extend non-branching paths with two properties.



for paired edges $n \dashrightarrow n'$:

Property 1 :

The sub-graph of opposite mates of p is a simple path of central node n'.

for overlap edges $n \rightarrow n'$:

Property 2 :

Both nodes are part of a variant sub-graph.

LOCALIZED ASSEMBLY

Constructing explicitly the whole paired string graph is too memory-intensive.

- Proposed "Enhanced greedy" assembly approach :
  - Greedily construct a sub-graph starting from a specific read.
  - Stop when the sub-graph is not a PNBP anymore, don't reuse nodes.
  - Repeat as long as there are unused reads.

This allows to perform targeted scaffolds assembly (around a region of interest).

- Extends recent unpaired targeted assembly approaches (TASR and Mapsembler).

OUTLINE

EXPERIMENTAL VALIDATION

Focus : Illumina assemblies. Assembly of two *E. coli*-based datasets.

- Experimental haploid is *E. coli* SRX000429 run, 2x36 bp (200 bp insert), 80x coverage.
- Simulated diploid is *E. coli* with artificial SNPs, 2x75 bp (500 bp insert), 100x coverage.

Scaffold N50 results (no mis-joins) :

Dataset	Our method	Velvet (dBG)	Ray (greedy)
Exp. haploid	101.8	95.3	87.3
Sim. diploid	134.1	132.6	10.2

- At least on bacterial genomes, localized scaffold construction yields similar results to classical scaffolding methods
- Practical non-branching paths overcome greedy assembly limitations

EXPERIMENTAL VALIDATION

Focus : Illumina assemblies. Assembly of two *E. coli*-based datasets.

- Experimental haploid is *E. coli* SRX000429 run, 2x36 bp (200 bp insert), 80x coverage.
- Simulated diploid is *E. coli* with artificial SNPs, 2x75 bp (500 bp insert), 100x coverage.

Scaffold N50 results (no mis-joins) :

Dataset	Our method	Velvet (dBG)	Ray (greedy)
Exp. haploid	**101.8**	**95.3**	87.3
Sim. diploid	**134.1**	**132.6**	10.2

- At least on bacterial genomes, localized scaffold construction yields similar results to classical scaffolding methods
- Practical non-branching paths overcome greedy assembly limitations

EXPERIMENTAL VALIDATION

Focus : Illumina assemblies. Assembly of two *E. coli*-based datasets.

- Experimental haploid is *E. coli* SRX000429 run, 2x36 bp (200 bp insert), 80x coverage.
- Simulated diploid is *E. coli* with artificial SNPs, 2x75 bp (500 bp insert), 100x coverage.

Scaffold N50 results (no mis-joins) :

Dataset	Our method	Velvet (dBG)	Ray (greedy)
Exp. haploid	101.8	95.3	87.3
Sim. diploid	**134.1**	132.6	**10.2**

- At least on bacterial genomes, localized scaffold construction yields similar results to classical scaffolding methods
- Practical non-branching paths overcome greedy assembly limitations

EXPERIMENTAL VALIDATION

Running time (6 threads) and memory consumption for the experimental dataset :

Ressources	Our method	Velvet (dBG)	Ray (greedy)
Running time (min)	7	8	16
Memory usage (Gb)	0.6	2.4	3.2

- Paired reads are indexed using a *k*-mer based hashing scheme.
- Erroneous *k*-mers are discarded early. Each *k*-mer references only a few reads [Chapuis 11].

OUTLINE

CONCLUSION

Our contribution

- Paired string graphs, direct construction of scaffolds from reads.
- Localized assembly using practical non-branching paths, overcomes greedy assemblers limitations.

Future directions/questions

- Mate-pairs probably cannot be included in this formalism. Looking forward build to a Chinese Postman common sub-paths scaffolder. [Nagarajan 09]
- Is these a non-heuristic gap-closing formalism for de Bruijn/string graphs ?

ACKNOWLEDGEMENTS

- BioGenouest Platform and the Symbiose team
- Assemblathon1 team :
  - Guillaume Chapuis
  - Delphine Naquin
  - Nicolas Maillet
  - Dominique Lavenier