Computational Biology

Lecture 15

Saad Mneimneh

DNA sequencing

- Shortest common superstring SCS
  - "An elegant theoretical abstraction, but fundamentally flawed" – R. Karp

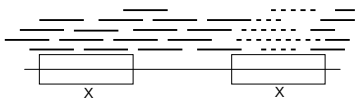
Given a set of fragments F,

Find the shortest string s that contains every $f \in F$ as a substring

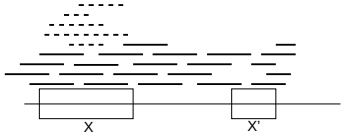
- This is NP-hard
- The SCS might not be what we really want

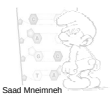
Saad Mneimneh

Bad example (repeats)



Shortest common superstring will give:





Saad Mneimneh

Solving SCS

We are going to consider a Hamiltonian path approach to solving the SCS problem

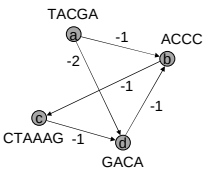
Saad Mneimneh

Overlap graph

- Consider the complete directed weighted graph $G = (V, E)$, called the overlap graph
  - $V = F$ (each fragment is a vertex)
  - $(u,v) \in$ E with weight $-t$ iff t is the length of the maximal suffix of u that is a prefix of v
- We allow self loops and zero weight edges

Saad Mneimneh

Example



a = TACGA
b = ACCC
c = CTAAAG
d = GAGC

0 weight edges
not shown

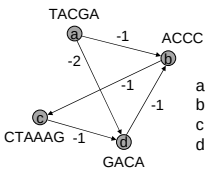
Saad Mneimneh

A path defines a superstring

- Every simple path *P* in the overlap graph involving a set of vertices (fragments) *A* defines a superstring *s*(*P*) for the set *A*.
- Therefore, a Hamiltonian path in the overlap graph defines a superstring for the set of fragments *F*.
- A Hamiltonian path must exist because the graph is complete (how many do we have?).

Saad Mneimneh

Example



a = TACGA
b = ACCC
c = CTAAAG
d = GAGC

0 weight edges not shown

P = adbc

```
TACGA
    GACA
        ACCC
            CTAAAG
----------------
```

s(*P*): TACGACACCCTAAAG

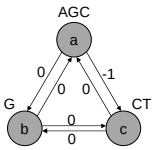
Saad Mneimneh

Does a superstring define a path?

- We have seen that every Hamiltonian path corresponds to a superstring.
- Is the converse true?
  - No: A superstring can contain arbitrary characters that are not present in any fragments
- Does a **shortest** superstring correspond to a Hamiltonian path?
  - Yes: if *F* is substring-free, i.e. no fragment in *F* is contained in another

Saad Mneimneh

Example



The shortest superstring is

AGCT

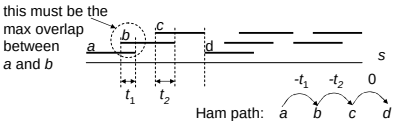
There is no Hamiltonian path *P*, such that *s*(*P*) = AGCT

Saad Mneimneh

Subtring-free collection *F*

Let *F* be a substring free set, then for every shortest superstring *s*, there is a Hamiltonian path *P*, such that *s*(*P*) = *s*.

Proof: assume the fragments appear in *s* as follows
(no gaps and no one can be contained in another)



Saad Mneimneh

Non substring-free *F*

- If *F* is not substring-free, then we can remove all fragments from *F* that are substrings of other fragments
- We end up with a set *F*'
- But any superstring of *F*' is a superstring of *F*
- Therefore, we can use *F*'

Saad Mneimneh

Length of string v.s. weight of path

- Let P be a Hamiltonian path.
- Let $w(P)$ be the weight of P.
- Let $||F|| = \Sigma_{a \in F}|a|$
- Then $|s(P)| = ||F|| + w(P)$ [proof is simple]
- Therefore, the shortest common superstring corresponds to the Hamiltonian path with minimum weight

Saad Mneimneh

Proof

Let P be a Hamiltonian path with minimum weight

we need to show that $s(P)$ is a shortest superstring

- Let s be a shortest superstring with $|s| < |s(P)|$
- Then there is a Hamiltonian path P' such that $s = s(P')$
- $|s(P')| = ||F|| + w(P') < |s(P)| = ||F|| + w(P)$
- Therefore, $w(P') < w(P)$, contradiction

Saad Mneimneh

Hamiltonian path approach

- Finding a minimum weight Hamiltonian path is NP-hard (you can reduce HAMPATH to it)
- Unfortunately, there is no “better” approach to solve SCS, because SCS itself is NP-hard
- Let’s consider a greedy algorithm for finding a Hamiltonian path

Saad Mneimneh

Greedy algorithm

- Greedy:
  - start with an empty path
  - repeatedly add the least weighted available edge until you get a Hamiltonian path
- Every time we add an edge (u,v), we need to check:
  - (u,v) does not create a cycle with the previously added edges
  - u has no previously added outgoing edge
  - v has no previously added incoming edge

Saad Mneimneh

Greedy algorithm

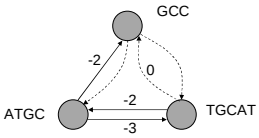
sort edges by their weight: $e_1, e_2, \ldots e_{|E|}$
for all $v \in V$
  in(v) ← 0
  out(v) ← 0
$H \leftarrow \phi$
$i \leftarrow 1$
while $|H| < |F| - 1$
  $(u,v) \leftarrow e_i$
  if out(u) = 0 and in(v) = 0
  then
    if H $\cup$ e_i does not contain a cycle [disjoint set data structure]
      $H \leftarrow H \cup e_i$
      out(u) ← 1
      in(v) ← 1
      $i \leftarrow i + 1$

To build the graph: trivially $O(|F|^2)$
(could be done optimally in $O(n^2 + |F|)$ using suffix trees)

To run the algorithm: $O(n^2 \log n)$

Saad Mneimneh

Example



Greedy algorithm will choose:

```
ATGC
  TGCAT
      GCC
--------
ATGCATGCC
```

Optimal is:

```
TGCAT
    ATGC
      GCC
--------
TGCATGCC
```

Saad Mneimneh

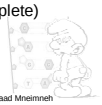
Sequncing By Hybridization SBH

- Use all possible probes of length l and obtain hybridization data with the DNA.
- If no errors, we have all substrings of length l.
- We would like to reconstruct the DNA from those substrings.
- We can formalize this as SCS and solve it as before.
- But we can simplify a little bit…

Saad Mneimneh

SBH and SCS

- SBH is a special case of the SCS problem where all fragments of F have the same length l.
- In the overlap graph, we will keep only the edges with weights equal to $-(l-1)$.
- By construction of these fragments, we know that there must be a Hamiltonian path in this modified overlap graph.
- All Hamiltonian paths now have the same weight $= -(n-1)(l-1)$
- Thus we only need to find a Hamiltonian path (still NP-complete)

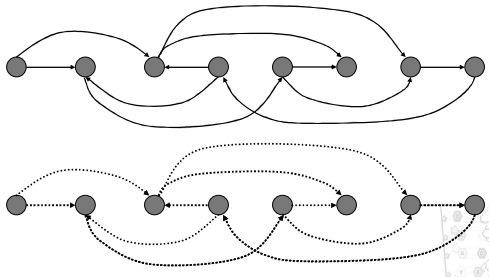


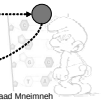
Saad Mneimneh

Example

$l = 3$

ATG TGG TGC GTG GGC GCA GCG CGT





Saad Mneimneh

Idea…

- Instead of representing fragments as vertices, represent them as edges.
- Then, instead of looking for a Hamiltonian path (a path that goes through each vertex once), look for an Euler path (a path that goes through each edge once).
- Euler path can be found in linear time.

Saad Mneimneh

Fragments as edges

- Construct a directed graph $G = (V, E)$
  - V: $(l - 1)$ length fragments

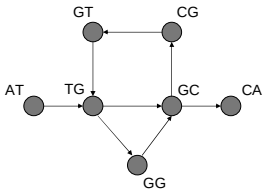
    (these can be obtained from our set F by considering the first and last $l - 1$ characters of each fragment)
  - E: A directed edge (u,v) for each fragment in F that starts with u and ends with v

Saad Mneimneh

Example

$l = 3$

ATG TGG TGC GTG GGC GCA GCG CGT



Saad Mneimneh

Euler Cycle

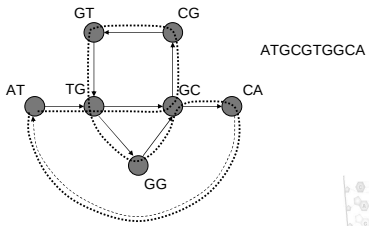
- By construction of the fragments, we know that the graph will have all vertices balanced except possibly for two unbalanced vertices
  (each occurrence of an *l*-1 fragment is shared by two *l* length fragments, except possibly for the first and last one)
- By adding an edge between two unbalanced vertices we can make the graph balanced
- Then we can find an Euler cycle in the graph (since it is balanced, there is one)

Saad Mneimneh

Example

l = 3

ATG TGG TGC GTG GGC GCA GCG CGT



Saad Mneimneh