Comparison of multiple biological sequences

Project in Bioinformatics
Astrid Christiansen, 201404423
Supervised by Christian N. S. Pedersen
Aarhus University

June 2022

Table of Contents

Table of Contents

MSA and SP score

$$M = \begin{bmatrix} A & C & C & A & C & C & A & G & - & G & T \\ G & C & - & A & - & C & T & - & A & - & T \\ A & C & C & - & - & A & T & G & A & G & G \end{bmatrix} \begin{array}{l} \rangle \textcircled{1} \\ \rangle \textcircled{2} \end{array} \rangle \textcircled{3}$$

$$A_1 = \begin{bmatrix} A & C & C & A & C & C & A & G & - & G & T \\ G & C & - & A & - & C & T & - & A & - & T \end{bmatrix} \quad \textcircled{1}$$

$$A_2 = \begin{bmatrix} G & C & - & A & \cancel{-} & C & T & - & A & - & T \\ A & C & C & - & \cancel{-} & A & T & G & A & G & G \end{bmatrix} \quad \textcircled{2}$$

$$A_3 = \begin{bmatrix} A & C & C & A & C & C & A & G & - & G & T \\ A & C & C & - & - & A & T & G & A & G & G \end{bmatrix} \quad \textcircled{3}$$

$$\begin{aligned} SP(M) &= \sum_{0 \le c < \ell} \sum_{0 \le i < j < k} d(M_{i,c}, M_{j,c}) \\ &= \sum_{0 \le i < j < k} \sum_{0 \le c < \ell} d(M_{i,c}, M_{j,c}). \end{aligned}$$

MSA and SP score

$$M = \left[\begin{array}{ccccccccccc} A & C & C & A & C & C & A & G & - & G & T \\ G & C & - & A & - & C & T & - & A & - & T \\ A & C & C & - & - & A & T & G & A & G & G \end{array}\right] \begin{array}{l} \big\rangle \text{①} \\ \big\rangle \text{②} \end{array} \Big\rangle \text{③}$$

$$A_1 = \left[\begin{array}{ccccccccccc} A & C & C & A & C & C & A & G & - & G & T \\ G & C & - & A & - & C & T & - & A & - & T \end{array}\right] \quad \text{①}$$

$$A_2 = \left[\begin{array}{ccccccccccc} G & C & - & A & \cancel{-} & C & T & - & A & - & T \\ A & C & C & - & \cancel{-} & A & T & G & A & G & G \end{array}\right] \quad \text{②}$$

$$A_3 = \left[\begin{array}{ccccccccccc} A & C & C & A & C & C & A & G & - & G & T \\ A & C & C & - & - & A & T & G & A & G & G \end{array}\right] \quad \text{③}$$

$$\begin{aligned} SP(M) &= \sum_{0 \le c < \ell} \; \sum_{0 \le i < j < k} d(M_{i,c}, M_{j,c}) \\ &= \sum_{0 \le i < j < k} \; \sum_{0 \le c < \ell} d(M_{i,c}, M_{j,c}). \end{aligned}$$

- Dyn. prog.: $O(n^k)$.

MSA and SP score

$$M = \begin{bmatrix} A & C & C & A & C & C & A & G & - & G & T \\ G & C & - & A & - & C & T & - & A & - & T \\ A & C & C & - & - & A & T & G & A & G & G \end{bmatrix} \begin{array}{l} \rangle \textcircled{1} \\ \rangle \textcircled{2} \end{array} \Big\rangle \textcircled{3}$$

$$A_1 = \begin{bmatrix} A & C & C & A & C & C & A & G & - & G & T \\ G & C & - & A & - & C & T & - & A & - & T \end{bmatrix} \quad \textcircled{1}$$

$$A_2 = \begin{bmatrix} G & C & - & A & \cancel{-} & C & T & - & A & - & T \\ A & C & C & - & \cancel{A} & A & T & G & A & G & G \end{bmatrix} \quad \textcircled{2}$$

$$A_3 = \begin{bmatrix} A & C & C & A & C & C & A & G & - & G & T \\ A & C & C & - & - & A & T & G & A & G & G \end{bmatrix} \quad \textcircled{3}$$

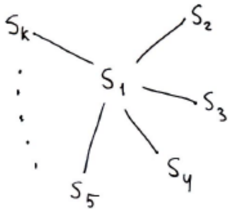
$$SP(M) = \sum_{0 \le c < \ell} \sum_{0 \le i < j < k} d(M_{i,c}, M_{j,c}) = \sum_{0 \le i < j < k} \sum_{0 \le c < \ell} d(M_{i,c}, M_{j,c}).$$

- Dyn. prog.: $O(n^k)$.
- Gusfield and MST algo.: $O(k^2 n^2)$.

Table of Contents

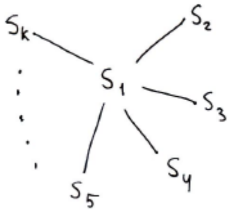
Gusfield's approximation algorithm

$$\arg\min_{S_1} \sum_{s \in S} OPT(S_1, s).$$



Gusfield's approximation algorithm

$$\arg\min_{S_1} \sum_{s \in S} OPT(S_1, s).$$

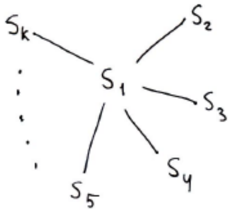


Algorithm GUSFIELD-MSA(S)

```
1: $S_1 \leftarrow center(S)$
2: $M \leftarrow [\,]$
3: for $i = 2$ to $k$ do
4:     $A \leftarrow align(S_1, S_i)$
5:     $M \leftarrow extend(M, A)$
6: end for
7: return $M$
```

Gusfield's approximation algorithm

$$\arg\min_{S_1} \sum_{s \in S} OPT(S_1, s).$$



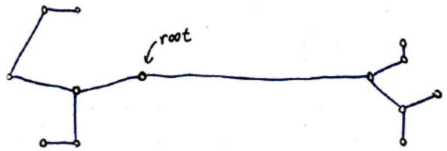
Algorithm GUSFIELD-MSA(S)

```
1: $S_1 \leftarrow center(S)$
2: $M \leftarrow [\,]$
3: for $i = 2$ to $k$ do
4:     $A \leftarrow align(S_1, S_i)$
5:     $M \leftarrow extend(M, A)$
6: end for
7: return $M$
```

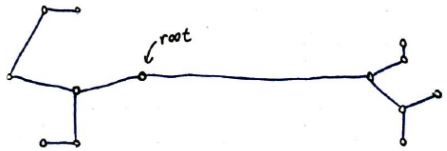
- Time $O(k^2 n^2)$.

Table of Contents

MST algorithm



MST algorithm



Algorithm MST-MSA(MST)

```
1: $M \leftarrow [\,]$
2: for $(S_i, S_j) \in MST$ do
3:     $A \leftarrow align(S_i, S_j)$
4:     $M \leftarrow extend(M, A)$
5: end for
6: return $M$
```

Prim's algorithm

Algorithm MST-PRIM-SIMPLE($G, w, root$)

```
1: for each $u \in V[G]$ do
2:     $key[u] \leftarrow \infty$
3:     $\pi[u] \leftarrow NIL$
4: end for
5: $key[root] \leftarrow 0$
6: for $i = 1$ to $|V[G]|$ do
7:     $u \leftarrow$ FIND-MIN($V[G], key$)
8:     report $(\pi[u], u)$
9:     for each $v \in Adj[u]$ do
10:        if $v \notin MST$ and $w(u, v) < key[v]$ then
11:            $\pi[v] \leftarrow u$
12:            $key[v] \leftarrow w(u, v)$
13:        end if
14:    end for
15: end for
```

Table of Contents

Implementation

```python
def MST_MSA_approx(seq_indices, seqs, sub_matrix, gap_cost):
    M = []
    # Keep track of seq index vs row index in M
    seq_idx_to_row = [None] * len(seq_indices)
    # Iterate MST edges
    MST = MST_prim(seq_indices, seqs, sub_matrix, gap_cost)
    for i in range(len(MST)):
        # Edge (parent, node)
        parent = MST[i][0]
        node = MST[i][1]
        # Fill out dyn. prog. table for pairwise alignment
        table = fill_table(seqs[parent], seqs[node],
                    sub_matrix, gap_cost)
        # Construct opt. pairwise align. from table
        A = construct_alignment(table, seqs[parent], seqs[node],
                sub_matrix, gap_cost)
        if i == 0:
            seq_idx_to_row[parent] = 0
            seq_idx_to_row[node] = 1
            M = A
        else:
            seq_idx_to_row[node] = len(M)
            M = extend_M(M, A, seq_idx_to_row[parent])
    return M
```

Table of Contents

Running time

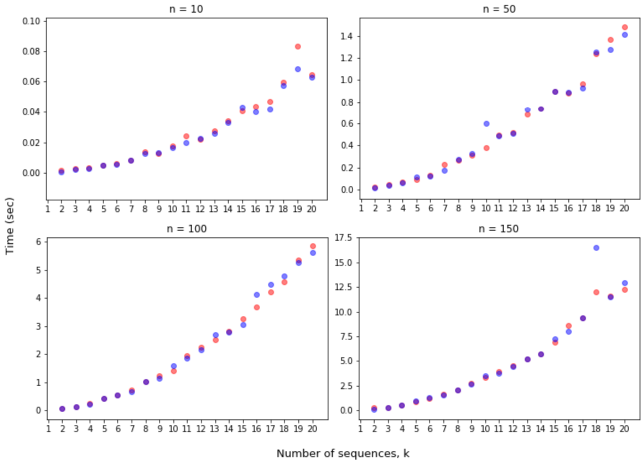


Figure: Running time of the two algorithms on random data as a function of k for four different values of n. Gusfield is red, MST is blue.

Running time

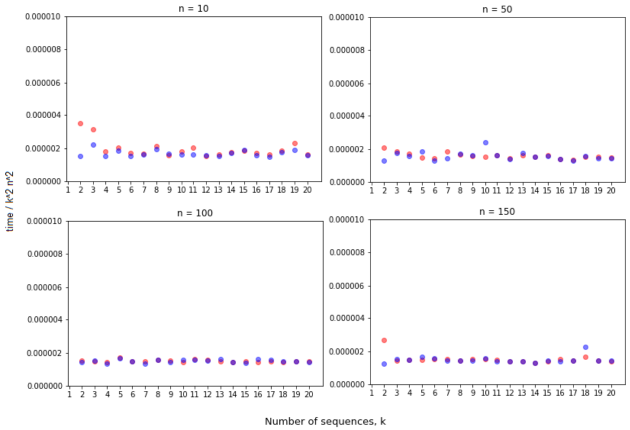


Figure: Running time of the two algorithms divided by expected worst case running time, k^2n^2, on random data. Gusfield is red, MST is blue.

SP score - random data

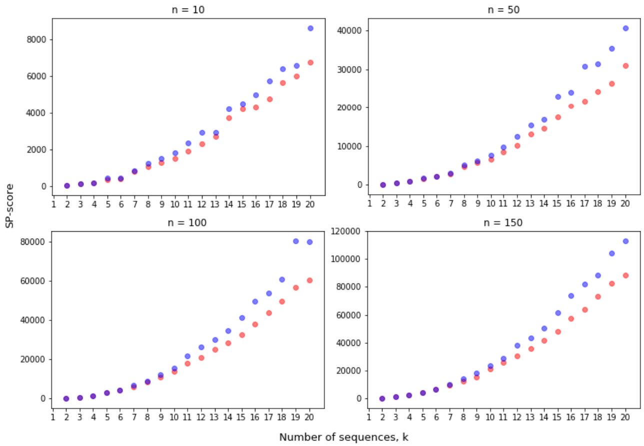


Figure: SP scores for the two algorithms as a function of *k* for four different values of *n* on random data. Gusfield is red, MST is blue.

SP score - low mutation rate data

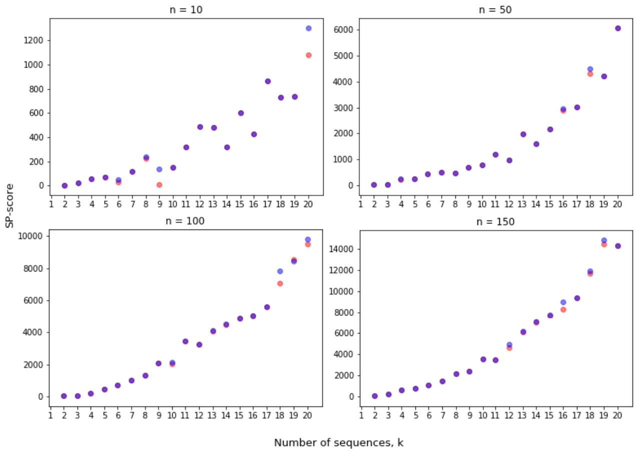
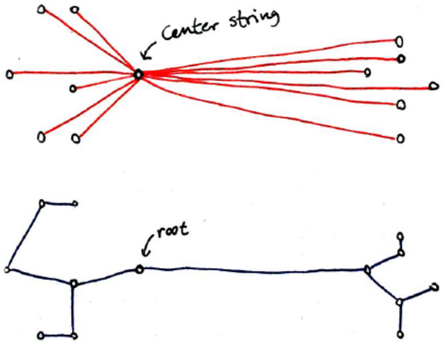


Figure: SP scores for the two algorithms as a function of *k* for four different values of *n* for simulated data with a low substitution rate of 0.05. Gusfield is red, MST is blue.

Clustered data



SP score - two clusters

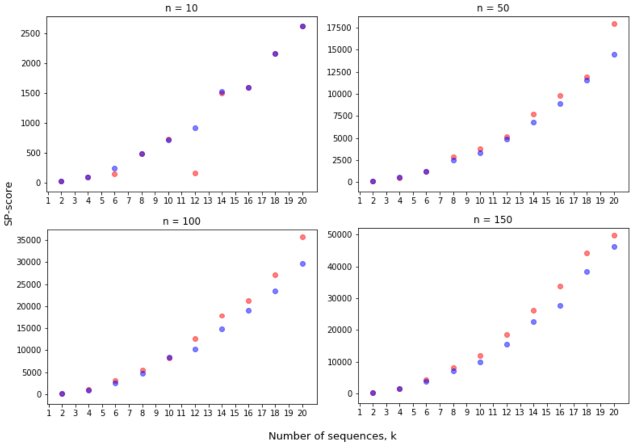
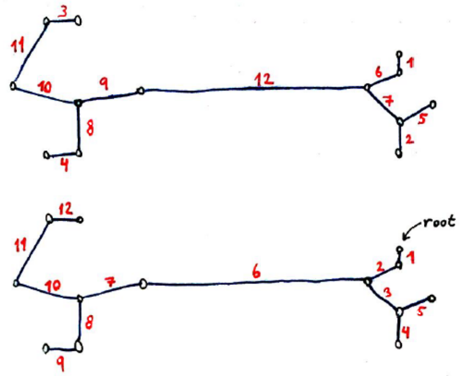


Figure: SP scores for the two algorithms as a function of *k* for four different values of *n* for simulated data consisting of two clusters (each made with a low substitution rate of 0.05). Gusfield is red, MST is blue.

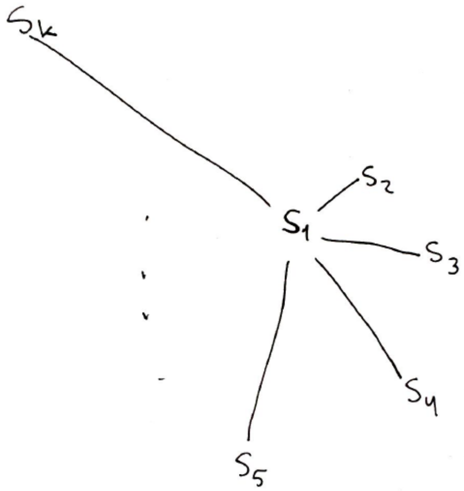
Conclusion

- Random data, high mutation rate: Gusfield is best.
- Low mutation rate: Very similar results.
- Two clusters: MST is best.

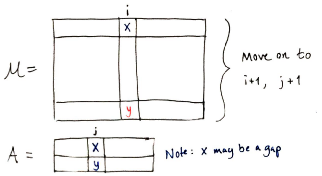
Alternative MST approach: Kruskal



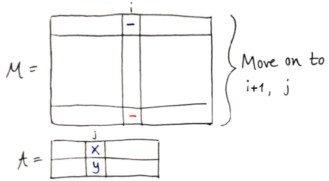
Alternative star tree approach



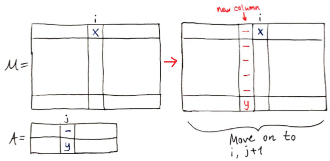
Extending M



M =
A =
i
j
x
y
Move on to i+1, j+1
Note: x may be a gap



M =
A =
i
j
–
x
y
Move on to i+1, j



M =
A =
i
j
x
y
–
new column
Move on to i, j+1

Prim's algorithm using queue

Algorithm MST-PRIM($G, w, root$)

```
for each $u \in V[G]$ do
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow NIL$
end for
$key[root] \leftarrow 0$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$ do
    $u \leftarrow \mathrm{EXTRACT\text{-}MIN}(Q)$
    report $(\pi[u], u)$
    for each $v \in Adj[u]$ do
        if $v \in Q$ and $w(u, v) < key[v]$ then
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$
        end if
    end for
end while
```

FIND-MIN

Algorithm FIND-MIN(V, key)

```
1: $min\_key \leftarrow \infty$
2: $min\_node \leftarrow NIL$
3: **for** $v \in V$ **do**
4:     **if** $v \notin MST$ **and** $key[v] < min\_key$ **then**
5:         $min\_node \leftarrow v$
6:         $min\_key \leftarrow key[v]$
7:     **end if**
8: **end for**
9: **return** $min\_node$
```