Tree Alignment

Tandy Warnow

February 13, 2017

Tree alignment
Tandy Warnow

Today's material:

- Review of last class
- Optimization problems for multiple sequence alignment
- Tree Alignment and Generalized Tree Alignment

Section 9.5 in the textbook covers this material.

Pairwise alignments and evolutionary histories

Recall:

- If we know how a sequence s evolves into another sequence s', we know the *true alignment* between the two sequences.
- We can use dynamic programming to find a minimum cost transformation of s into s', where each single letter indel costs C and each substitution costs C'. This algorithm is called *Needleman-Wunsch*.
- We can extend the Needleman-Wunsch algorithm to handle more general cost functions: where the cost for a gap of length L isn't just CL, and where the cost of a substitution depends on the pair of letters.

Multiple sequence alignment (MSA)

Now that we have a way of defining the “cost” of a pairwise alignment, we can extend the definition to a set of three or more sequences in at least two ways:

- Sum-of-pairs (SOP) score (find MSA to minimize the sum of costs of all induced pairwise scores)
- Treelength (find tree and sequences at all nodes of the tree to minimize the sum of costs on the edges of the tree)

Both problems are NP-hard.

Treelength and Tree Alignments

- **Input:** We are given a tree T with sequences at the leaves, and a cost C for indels and cost C' for substitutions.
- **Output:** Sequences at each node to minimize the total treelength (summing over all the edges) of the tree.

Example 9.5 from textbook

The input is $s_1 = AC, s_2 = ATAC, s_3 = CAG$. We seek the sequence X at the internal node of the tree with s_1, s_2, and s_3 at the leaves.

1. Compute the pairwise alignments obtained on each edge for $X = AC$.
2. Compute the multiple sequence alignment (MSA) defined by the pairwise alignments computed in (1).
3. Compute the SOP-score of the MSA computed in (2).
4. Compute the treelength of the MSA computed in (2).

Tree Alignment

The Tree Alignment problem (finding the sequences at internal nodes of a fixed tree to find the minimum cost) is NP-hard.

Generalized Tree Alignment

- **Input:** Set S of sequences and positive constants C and C', where C is the cost of a single letter indel and C' is the cost for a substitution.
- **Output:** Tree T with S at the leaves and internal nodes labelled by sequences so that the treelength is minimized.

The Generalized Tree Alignment problem is also NP-hard.

Methods for Treelength include POY and BeeTLe, but these are heuristics without provable guarantees. Furthermore, they are computationally intensive.

Constraining the input sequences in GTA

- **Input:** Set S of sequences and positive constants C and C', where C is the cost of a single letter indel and C' is the cost for a substitution.
- **Output:** Tree T with S at the leaves and internal nodes labelled by sequences so that the treelength is minimized.

What happens if we constrain the sequences at the internal nodes to be drawn from S?

Can we solve this problem in polynomial time?

Minimum Spanning Tree

Given a graph $G = (V, E)$ with positive weights on the edges of G, a **Minimum Spanning Tree** (MST) is a subgraph of G that contains all the vertices and that has minimum total cost.

Finding the MST is solvable in polynomial time (Kruskal's Algorithm and Prim's Algorithm).

Let the vertices of the graph be all the sequences in S, and the weight of the edge between two vertices be the edit distance (computed by Needleman-Wunsch).

Hence, we can find an optimal solution to the constrained GTA problem in polynomial time.

Tree Alignments

Let T be a tree with leaves labelled by S and let S' be the assignment of sequences to the internal nodes in T. Then (T, S') defines a multiple sequence alignment $\mathcal{A}$. *Tree alignments* are those alignments that can be obtained in this way.

We are given a set S of sequences, and we wish to find an approximate solution to the Generalized Tree Alignment (GTA) problem.

Specifically, we say an algorithm Φ for GTA is a c-approximation algorithm if the GTA score of $\Phi(S)$ is no more than $c\times$ times the best possible score.

Theorem 9.6 from the textbook: A minimum spanning tree T is a 2-approximation to the GTA problem.

Summary of today's lecture

- Tree Alignment (where the tree is given)
- Generalized Tree Alignment (where the tree is not known)
- Minimum Spanning Trees
- Approximation algorithms for GTA