Algorithms for Computing Maximum Agreement Subtrees

Nikolaj Skipper Rasmussen 20114373

Thomas Hedegaard Lange 20113788

Master's Thesis, Computer Science

February 2016

Advisor: Christian Nørgaard Storm Pedersen

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Abstract

►in English. . . ◄

Resumé

►in Danish. . . ◄

Acknowledgements

►. . . ◄

Karsken Bælg,
Aarhus, February 12, 2016.

Contents

Chapter 1

Introduction

The Maximum Agreement Subtree problem (MAST) provides a measure of similarity, and is defined as such: Given two rooted trees, T1 and T2, created over the same leaf-set {1,2,3...,n}, determine the largest possible subset of leaves inducing an agreeing subtree of T1 and T2. For a set of leaves to induce an agreeing subtree for T1 and T2, the subtrees restricted to the set of leaves must be isomorphic, which means that they are structurally equivalent.

Let us start by motivating the interest in MAST by giving an example of its application. Suppose that we are interested in inspecting the relationship between DNA obtained from different species. This is typically done by the use of Hierachical Clustering (REF) or Neighbor Joining (REF) to construct evolutionary trees. However, finding the true evolutionary tree is often hard (find another way of expressing hard), and evidence is required to support any suggested tree topology.

The MAST problem is one of several ways of defining tree distances. -which one is supeiror?

The MAST problem applies to all trees, but we will choose to focus on the rooted, binary trees given that the motivation for the problem is primarily rooted in biology and linguistics, where these trees are most common.

$$\begin{aligned} \frac{3x+y}{7} &= 9 && \text{given} \\ 3x+y &= 63 && \text{multiply by 7} \\ 3x &= 63-y && \text{subtract y} \\ x &= 21-\frac{y}{3} && \text{divide by 3} \end{aligned}$$

$$f(T_a, T_x) = Max \begin{cases} f(T_b, T_y) + f(T_c, T_z) & \text{if TypeOf}(T_a)\text{=Type Of}(T_x)\text{=Internal Node} \\ f(T_b, T_z) + f(T_c, T_y) & \text{if TypeOf}(T_a)\text{=Type Of}(T_x)\text{=Internal Node} \\ f(T_a, T_y) & \text{if TypeOf}(T_x)\text{=Internal Node} \\ f(T_a, T_z) & \text{if TypeOf}(T_x)\text{=Internal Node} \\ f(T_b, T_x) & \text{if TypeOf}(T_a)\text{=Internal Node} \\ f(T_c, T_x) & \text{if TypeOf}(T_a)\text{=Internal Node} \\ 1 & \text{if TypeOf}(T_a)\text{=TypeOf}(T_x)\text{=Leaf} \wedge T_a{=}T_x \\ 0 & \end{cases}$$

►. . . ◄

Chapter 2

►. . . ◄

►example of a citation to primary literature: [1], and one to secondary literature: [?]◄

Chapter 3

►The NSquared algorithm◄

Goddard et. al.[1] describes a **►O(n2)◄** algorithm for finding the largest agreement subtree for two rooted binary trees. Given two trees T and U of size m and n, the idea is to iteratively find the largest agreement subtree and its size for every pair of subtrees from T and U. This can be done in quadratic time by using Lemma 1: **►...◄**

3.1 Implementation

We implemented the algorithm in java (using the forrester [**?**] library to represent the trees?). We computed the agreement subtrees for each pair of subtrees in the two trees by doing a postorder traversal of the first tree and for each node did a postorder traversal of the second tree. For each pair of nodes `a` and `w` we computed the agreement subtree $A_{a,w}$ for the two subtrees T_a and T_w having respectively `a` and `w` as roots. For such a pair of nodes there are three cases.
The first case is that `a` and `w` are both leaves. If the leaves are equal, i.e. they have the same name, then $A_{a,w}$ will be the tree consisting of exactly one leaf with that name. Otherwise $A_{a,w}$ is empty.
The second case is that we have a leaf and an internal node. Let's say `w` is the internal node having `x` and `y` as children. If the leaf corresponding to `a` is contained in T_w, it will also be contained in either T_x or T_y and $A_{a,w}$ will be the same as either $A_{a,x}$ or $A_{a,y}$. Since we do a postorder traversal of the trees, $A_{a,x}$ and $A_{a,y}$ have already been computed and $A_{a,w}$ can just be set to the largest of the two.
The third case is that both `a` and `w` are internal nodes, where `a` have children `b` and `c` and `w` have children `x` and `y`. In this case we can use Lemma 1. Again, all the agreement subtrees to consider have already been computed. If the largest subtree is either $A_{b,x} + A_{c,y}$ or $A_{b,y} + A_{c,x}$, $A_{a,w}$ will be the tree having the two subtrees as children.

All these agreement subtrees were stored in a matrix together with their sizes. The last cell in the matrix would then at the end contain the agreement subtree for the two input trees.

How did we verify the algorithm? E.g. bruteforce MAST and compare results.
Show trees and MAST outputted by the program.

►. . . ◄

Chapter 4

Conclusion

►. . . ◄

Bibliography

[1] Wayne Goddard, Ewa Kubicka, Grezegorz Kubicki, and F. R. McMorris. The agreement metric for labeled binary trees. *Mathematical Biosciences*, 123(2):215–226, October 1994.