An additive approximation to near-perfect phylogeny problem

Pranjal Awasthi
Jamie Morgenstern
Or Sheffet

July 30, 2011

1 Introduction and notation

Consider the problem of constructing a model of the evolution of a set of taxa. The input to the problem is some set of species, which may or may not include all ancestors of the members, and one would like to output an ancestral tree showing the relationship between the taxa, including ancestors not given as input to the algorithm. One would like the output to be biologically significant: that is to say, the tree should make species which are closely related to one another through a common ancestor appear as siblings, or cousins, in the tree.

The input for such a problem is a set of binary strings $M \subseteq \{0,1\}^m$ (referred to as taxa). The goal is to output a tree T with leaves and nodes labeled with taxa T_l (where $M \subseteq T_l$), which has low cost with respect to M for some notion of cost. For this paper, we will consider the parsimony cost of $T = (E, V)$, which is defined as $\sum_{(u,v)\in E} d(u,v)$ where d is the Hamming distance between u and v.

Certain instances of this problem are computationally easy. For instance, if the set M has the property that any coordinate $i \in [0, m-1]$ changes across exactly one edge, M is referred to as a perfect phylogeny and constructing an maximum parsimony tree can be done in linear time. Here, we assume no coordinate is constant across all taxa, or it could be removed with no cost from any optimal tree. The cost of such a phylogenetic tree is m, the dimensionality of the taxa.

An interesting question arises as to whether one can solve the problem easily in the case that the optimal tree has cost $m + q$, where q is relatively small compared to m.

Previous work...

2 Motivation for additive error

3 Algorithm

```
TerminalSet = M
LeafSet = $\emptyset$
Tree = $(V = M, E = \emptyset)$
Indices = $\emptyset$


PluckLeaf(Tset) :
if $\exists t \in Tset,\ i \in [m] \setminus Indices \mid t_i \neq t'_i \forall t' \in Tset$ then
    Tree = $(V \cup \{t_{\bar{i}}\}, E \cup \{(t, t_{\bar{i}}\})$
    PluckLeaf($Tset \setminus \{t\} \cup \{t_{\bar{i}}\}$)
    Indices = Indices $\cup \{i\}$
else if $m - q^2 >\mid Indices \mid$ then
    SplitProblem(Indices, Tset)
else
    Connect each $t \in Tset$ to $t_{[n] \setminus indices = 0}$
    PerfectPhylogeny the above
end if


SplitProblem(Indices, Tset):
Randomly select some $i \in [n] \setminus Indices$
Indices = Indices $\cup \{i\}$
$T_0$ = PluckLeaf(Tset$|_{i=0}$)
$T_1$ = PluckLeaf(Tset$|_{i=1}$)
ConnectTrees($i, T_0, T_1$, Tset)


ConnectTrees($i, T_0, T_1$, Tset):
mincost = $\infty$
FinTree= ()
for each leaf in $l \in T_0$ do
    $T_{1?} = PluckLeaf(Tset\mid_{i=1} \cup l_{\bar{i}})$
    $T_? = T_0 \cup T_{1?} \cup \{(l, l_{\bar{i}})\}$
    if $cost(T_?) < mincost$ then
        mincost = cost($T_?$)
        FinTree = $T_?$
    end if
end for
for each leaf in $l \in T_1$ do
    $T_{0?} = PluckLeaf(Tset\mid_{i=0} \cup l_{\bar{i}})$
    $T_? = T_1 \cup T_{0?} \cup \{(l, l_{\bar{i}})\}$
    if $cost(T_?) < mincost$ then
        mincost = cost($T_?$)
        FinTree = $T_?$
      end if
    end forFinTree
```

4 Analysis

Assuming $n > m$.

Theorem: this algorithm, with probability at least 1/2, outputs a tree with cost $m + O(q^4)$.

FIXME we assume we are building a binary tree (all trees can be converted to a binary tree in this case). Moreover, in our pruned tree, we assume we're thinking that adjacent goodedges in chains are compressed (we can see if two coordinates make the same partition).

Notice that, until splitproblem is called, we are buiding the tree optimally, e.g. that all leaves we identify are predetermined to be such leaves by the dataset for any optimal tree.

We call the piece of the tree we've left to build at this point the *pruned tree.* We will refer to the connections from the pruned tree to the already built part the *pruned leaves.*

Once SplitProblem is called, all of the pruned leaves are badedges in an optimal tree. If this were not the case, and some subtree were connected by a goodedge to the pruned tree, the terminal set would be partitionable into the terminals on the subtree and the rest of the terminals by a (set of) unused goodcoordinates, and PluckLeaf would be called again.

There are at most q badcoordinates and at most $2q$ badedges, there are at most $2q$ pruned leaves when SplitProblem is called. The pruned tree, as a binary tree, can have at most $2q$ splits (a split results in another pruned leaf, which requires a badedge). Thus, there are at most $2q$ vertices of degree 3 in the pruned tree and at most $2q$ vertices of degree 1 in it.

Claim: if, when we call SplitProblem, we pick a coordinate i such that in an optimal tree, the following all hold:

- i is good
- e_i's endpoints have degree 2
- e_i's adjacent edges are good
- One endpoint of e_i is a terminal, or one enpoint of e_i is part of a chain of degree 2 steiner nodes all connected by goodedges which ends in a terminal.

then, we can uniquely label the other endpoint of e_i, add it as a terminal to the other subproblem, and both subtrees will then have leaves which are pluckable.

What is the probability of any one of those conditions failing? The probability that i is badis $q/m_c < 1/50q$. Either of the endpoints having degree 1 or 3 has probability at most $4/50q$. The probability of either of the adjacent edges being badis at most $1/50q$. Conditioning on being on a chain of length at least 2 of goodedges, the probability that the end of the chain ends in either a badedge or a split is at most $4/50q$. So, the probability of failure is at most $1/5q$.

Each call to either SplitProblem or PluckLeaf, the current dimensionality of the problem we're working in decreases by at least 1. Moreover, we'll only need to call SplitProblem $O(q)$ times, since once we call SplitProblem we can work on that chain until we hit a badedge, which can happen at most $2q$ times. Thus, there is a constant probability of success of the algorithm's making the correct guesses each time SplitProblem is called.

Or, what here?