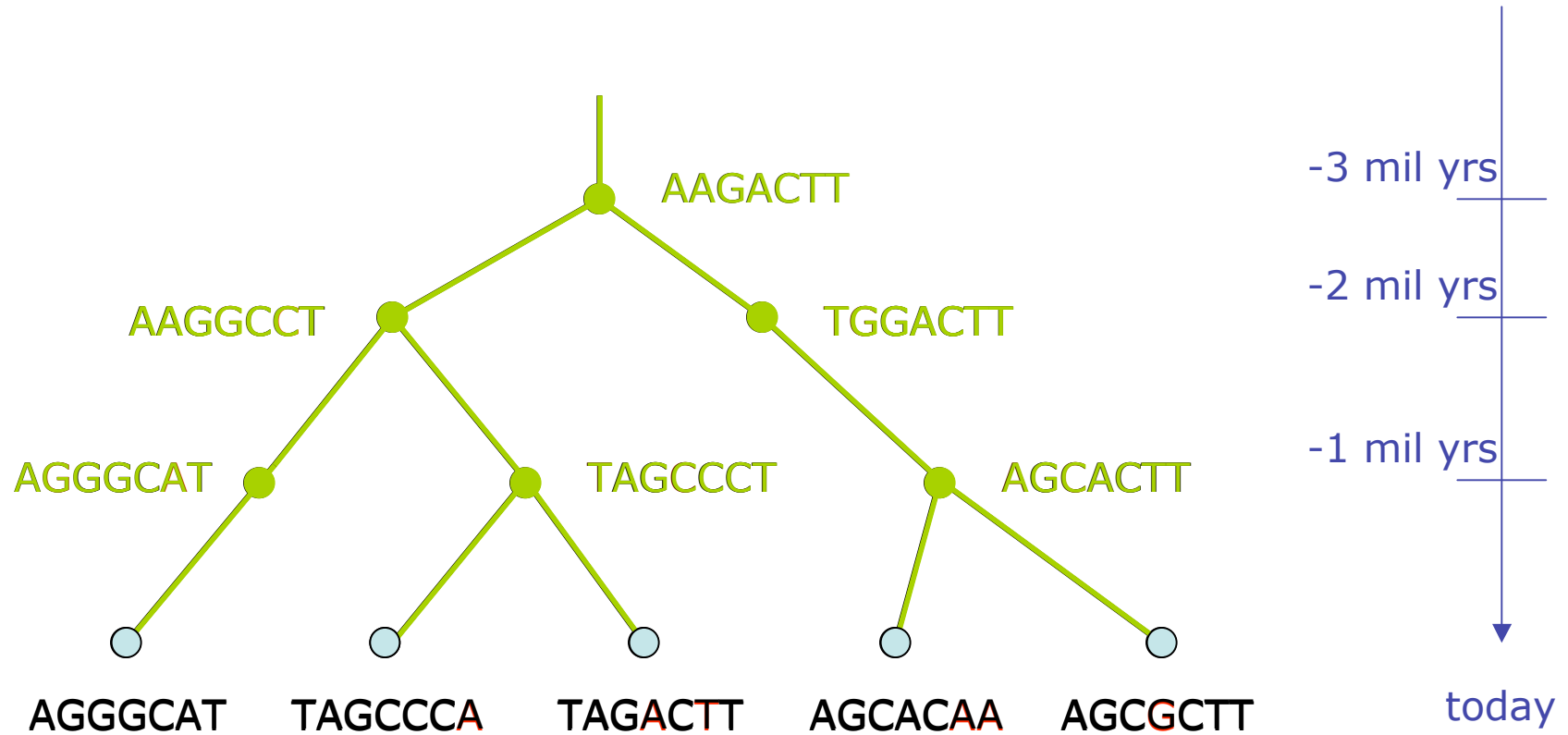


January 17, 2008

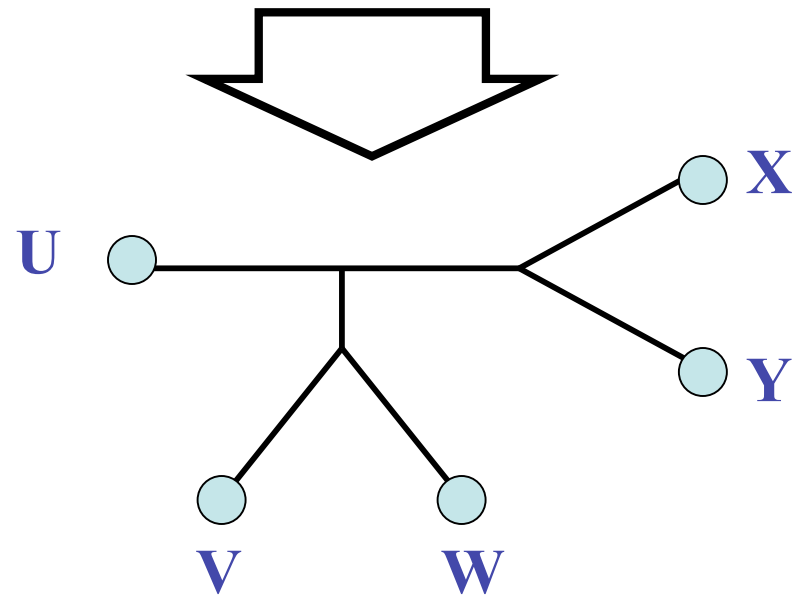
CS 395T: Computational Phylogenetics

# DNA Sequence Evolution



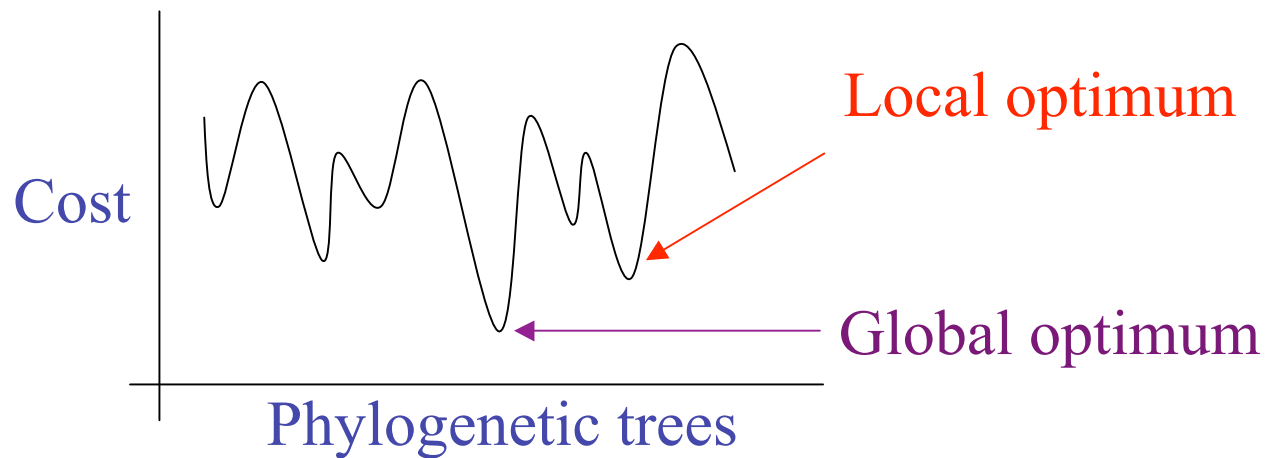
# Phylogeny Problem

U                    V                    W                    X                    Y  
AGGGCAT      TAGCCCA      TAGACTT      TGCACAA      TGCGCTT



# Phylogenetic reconstruction methods

1. Heuristics for NP-hard optimization criteria (Maximum Parsimony and Maximum Likelihood)



2. Polynomial time distance-based methods: Neighbor Joining, FastME, etc.
3. Bayesian MCMC methods.

# Empirical problems with existing methods

- Polynomial time methods have poor topological accuracy on large datasets – *we need better polynomial time methods.*
- Heuristics for Maximum Parsimony (MP) and Maximum Likelihood (ML) and Bayesian MCMC methods cannot handle large datasets (take too long!) – *we need new heuristics that can analyze large datasets.*

# Course outline

- Basics: phylogenies, data, stochastic models of evolution, and representations of trees
- Phylogeny reconstruction methods: distance-based and character-based (MP, ML, and Bayesian), and their performance issues
- Multiple sequence alignment, and the connections (both ways) between MSA and phylogenetics
- The development of new methods with improved performance (including DCM-boosting)
- Special topics: reticulate evolution, whole genome evolution, consensus methods, data mining, and applications

# Today

- Representations of trees: distance-matrices, clades, splits (bipartitions), and quartets
- Computing trees from dissimilarity matrices: the “naïve” quartet method
- (Hints) Connections to estimation of phylogenies from empirical data

# Bipartitions

- Given an edge  $e$  in a leaf-labelled tree  $T$ , the removal of the edge  $e$  (but not its endpoints) defines a bipartition on the leaves of the tree  $T$ . We denote by  $\mathbf{c}_e$  the bipartition defined by the edge  $e$ . We let  $\mathbf{C}(T)=\{\mathbf{c}_e: e \text{ in } E(T)\}$ .
- Questions: Given  $\mathbf{C}(T)$ , can we compute  $T$ ?

# Clades

- Definition: Let  $T$  be a rooted tree leaf-labelled by  $S$ , let  $v$  an internal node in  $T$ , and let  $X_v$  the leaves in  $T$  below  $v$ . Let  $\mathbf{Clades(T) = \{X_v: v \text{ in } V(T)\}}$ .
- Question: Given  $\mathbf{Clades(T)}$ , can we compute  $T$ ?

# Quartet subtrees

- Given tree  $T$  leaf-labelled by  $S$ , and quartet  $a,b,c,d$  of leaves, we let  $T|_{\{a,b,c,d\}}$  denote the minimal homeomorphic subtree of  $T$  restricted to  $\{a,b,c,d\}$ . We let  $Q(T)$  denote  $\{T|_X: X \text{ is a four taxon subset of } S\}$ .
- Question: Given  $Q(T)$ , can we compute  $T$ ?

# Computing trees

- Given  $Q(T)$  (the quartet subtrees of  $T$ ), can we determine  $T$ ?
- Given  $C(T)$  (the bipartitions of  $S$  defined by the edges of  $T$ ), can we determine  $T$ ?
- Given  $\text{Clades}(T)$  (the sets of leaves defined by internal nodes in the rooted tree  $T$ ), can we determine  $T$ ?

# Quartet-based reconstruction

- Definition: Let  $T$  be a tree leaf-labelled by a set  $S$ , and let  $Q(T)$  be the set of quartet subtrees of  $T$  (derived from each of the four-taxon subsets of  $S$ ).

**Question: can we reconstruct  $T$  from  $Q(T)$ ?**

# Computing $T$ from $Q(T)$

- Given  $Q(T)$ :
  - Find a sibling pair  $A, B$  (a pair of leaves which are always together in every quartet in which they both appear)
  - Compute the tree  $T'$  for  $S - \{A\}$  by recursing on the subset of  $Q(T)$  that doesn't include taxon  $A$
  - Insert  $A$  into  $T'$  by making  $A$  a sibling to  $B$ , and return the tree obtained

# Analysis of the algorithm

Questions:

- Accuracy?
- Running time?
- But: how are we to compute quartet subtrees?

# Clade compatibility

- Definition: Let  $T$  be a rooted tree leaf-labelled by  $S$ ,  $v$  an internal node in  $T$ , and  $X_v$  the leaves in  $T$  below  $v$ . Let  $\text{Clades}(T) = \{X_v : v \text{ in } V(T)\}$ .
- Theorem: Let  $X$  be a set of subsets of  $S$ . Then there exists a tree  $T$  leaf-labelled by  $S$  such that  $X = \text{Clades}(T)$  if and only if for all  $A, B$  in  $X$ , either  $A$  and  $B$  are disjoint, or one contains the other.

# Proof of the theorem

- One direction is easy
- The other direction is a proof by construction!

# Computing rooted trees from clades

- Partially order the set of clades by containment, add in the full set  $S$ , and compute the Hasse Diagram of the resultant poset

# Tree construction from clades

Questions:

- Accuracy?
- Running time?
- But, how are we to compute clades?

# Bipartition compatibility

- Definition: Let  $C$  be a set of bipartitions on a set  $S$ . Then  $C$  is said to be **compatible** if there exists a tree  $T$  leaf-labelled by  $S$  such that  $C=C(T)$ , where  $C(T) = \{c_e: e \text{ in } E(T)\}$ .

**Question: Can we construct the tree  $T$  from  $C(T)$ ?**

# Computing trees from bipartitions

- Given the set of bipartitions on the leaf-set induced by the edges of a tree  $T$ , how can we compute the tree  $T$ ?

Hint: “root” the tree  $T$  by picking it up at a leaf, and then consider the set of bipartitions as a set of “clades”, and apply the previous algorithm. (Note: the choice of leaf does not matter!)

# So?

- We can compute a tree from its set of clades, bipartitions, or quartets. But how do we get these sets?
- Primary data are generally characters (columns within alignments of biomolecular sequences, morphological features, or other such features). These don't directly produce these sets.
- But often evolutionary biologists have techniques for estimating “evolutionary distances” between taxa, and these are then useful for computing these sets!

To be continued