CS Research for The Tree of Life

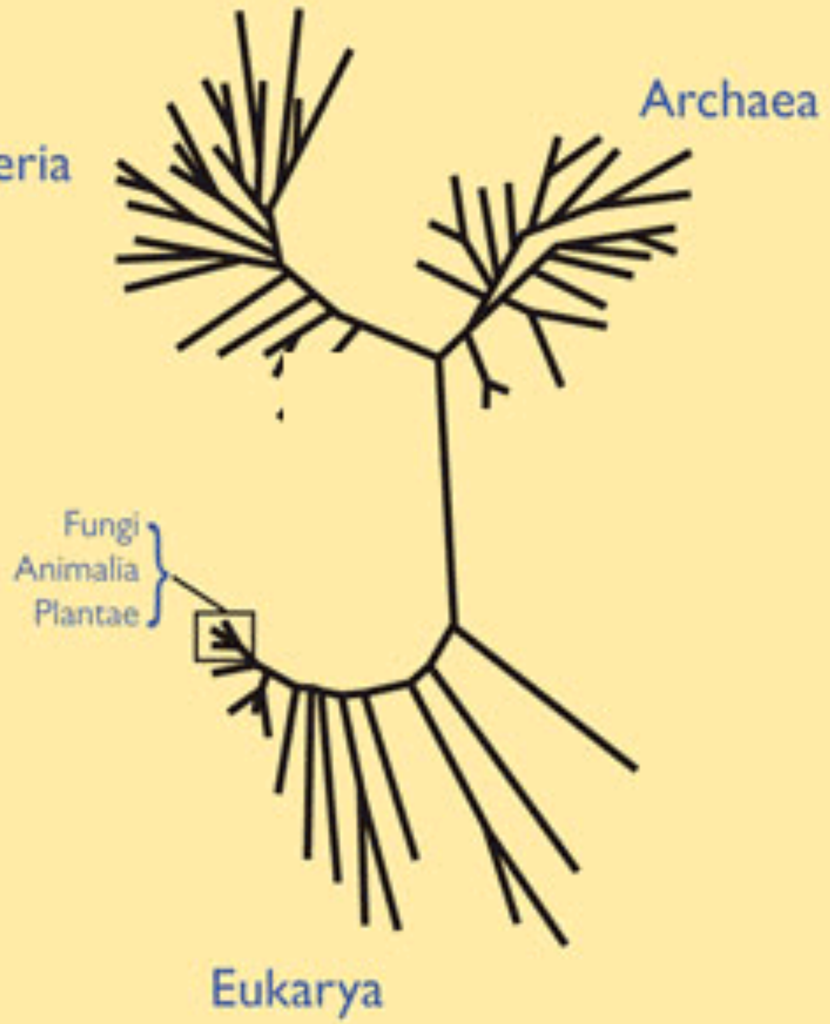
Tandy Warnow

“The Tree of Life”

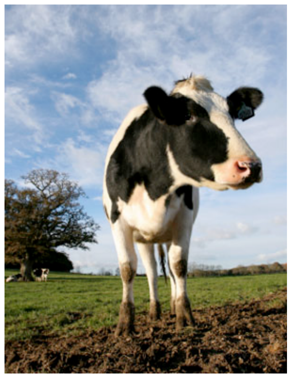
Fundamental science: Molecular biology, Genetics, Ecology, Behavior, etc.

Applications: Drug design, Forensics, Human migrations, etc.

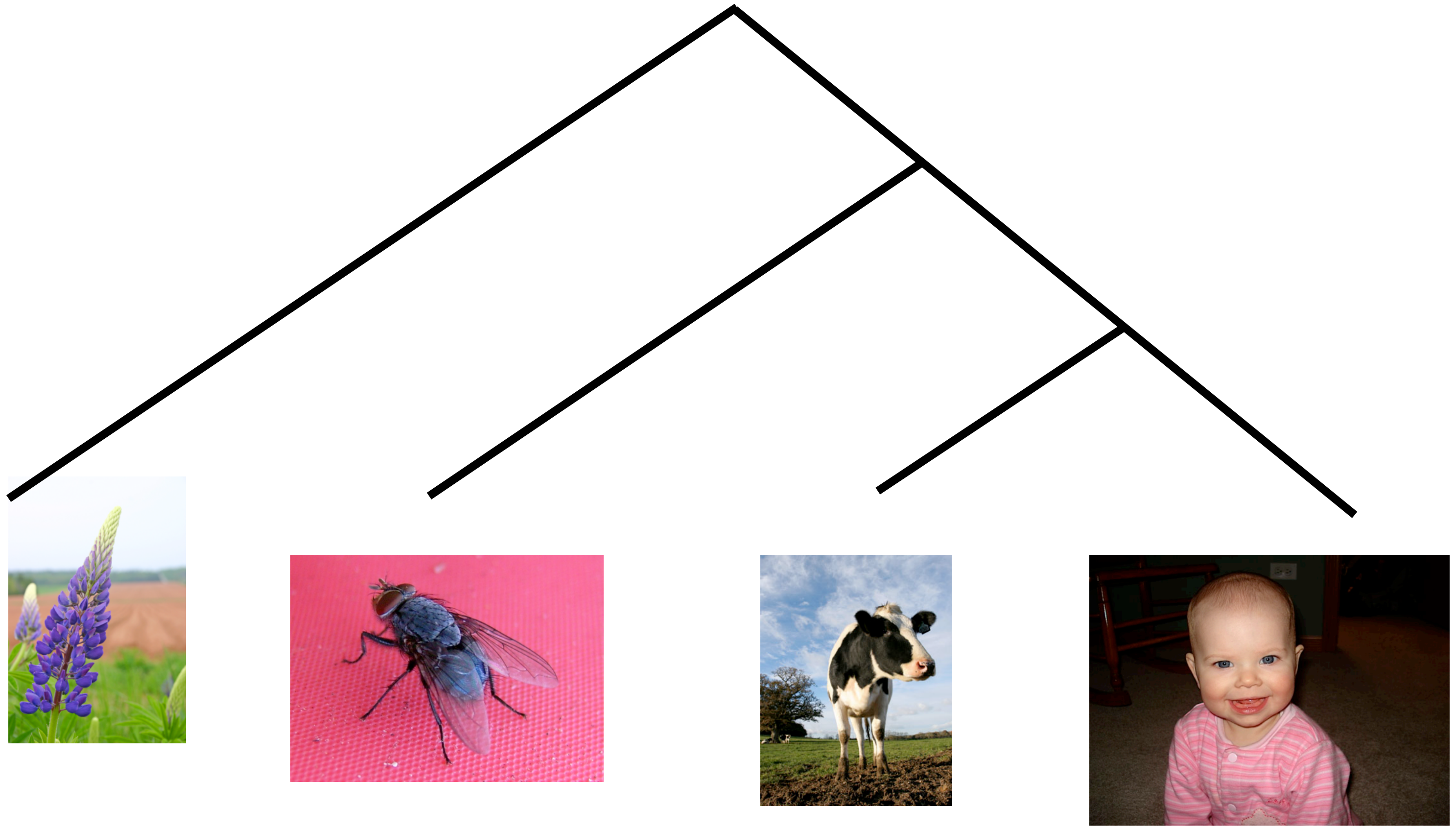
The Three-Domain Tree of Life



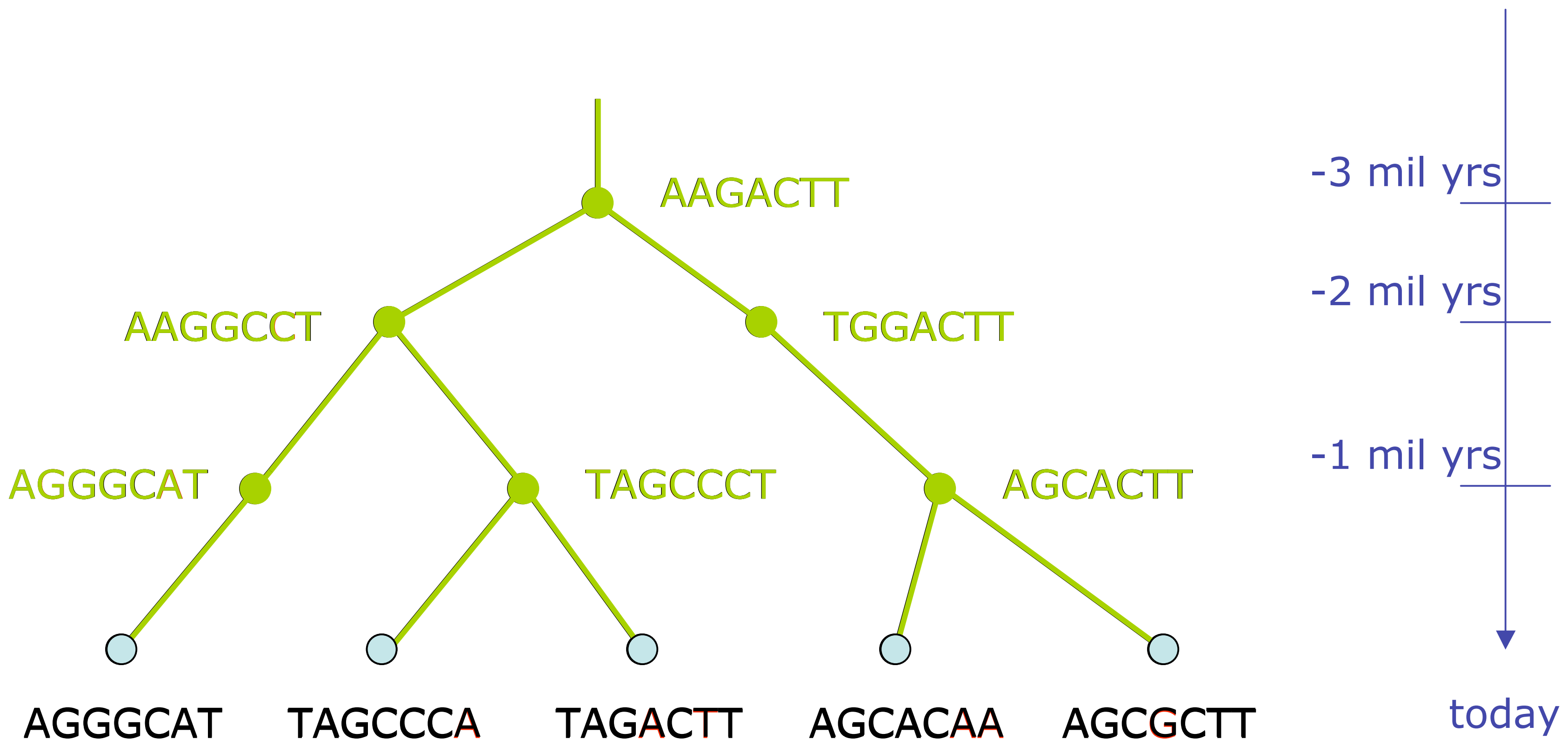
Estimating evolutionary trees

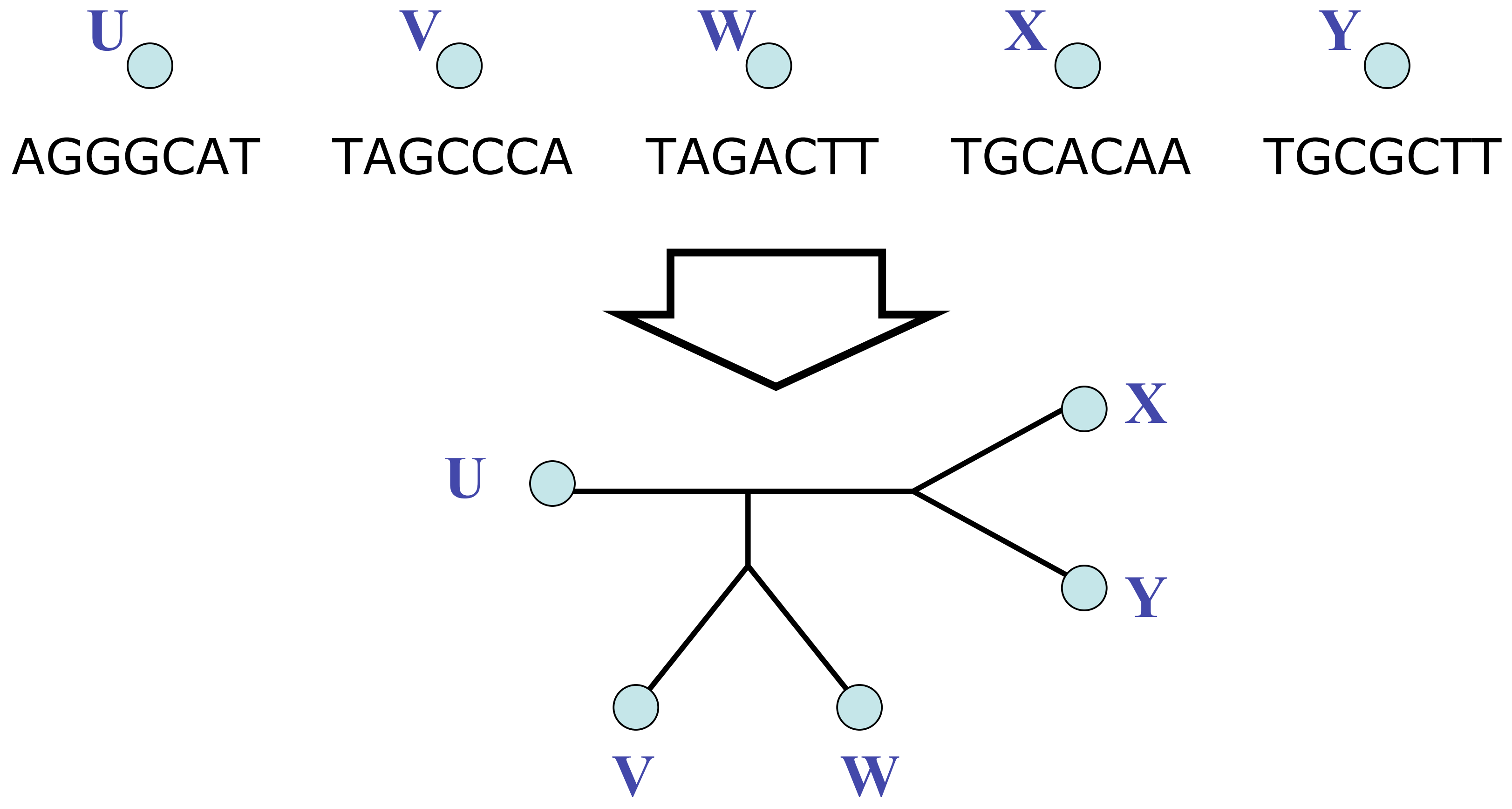


Easy cases: use morphology

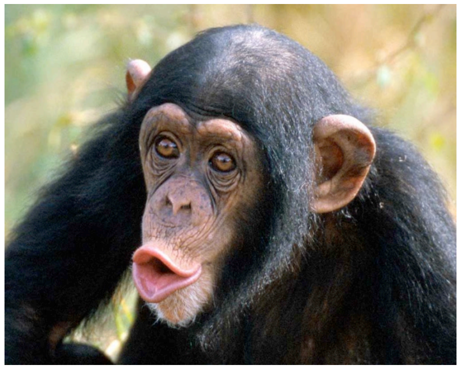


DNA Sequence Evolution

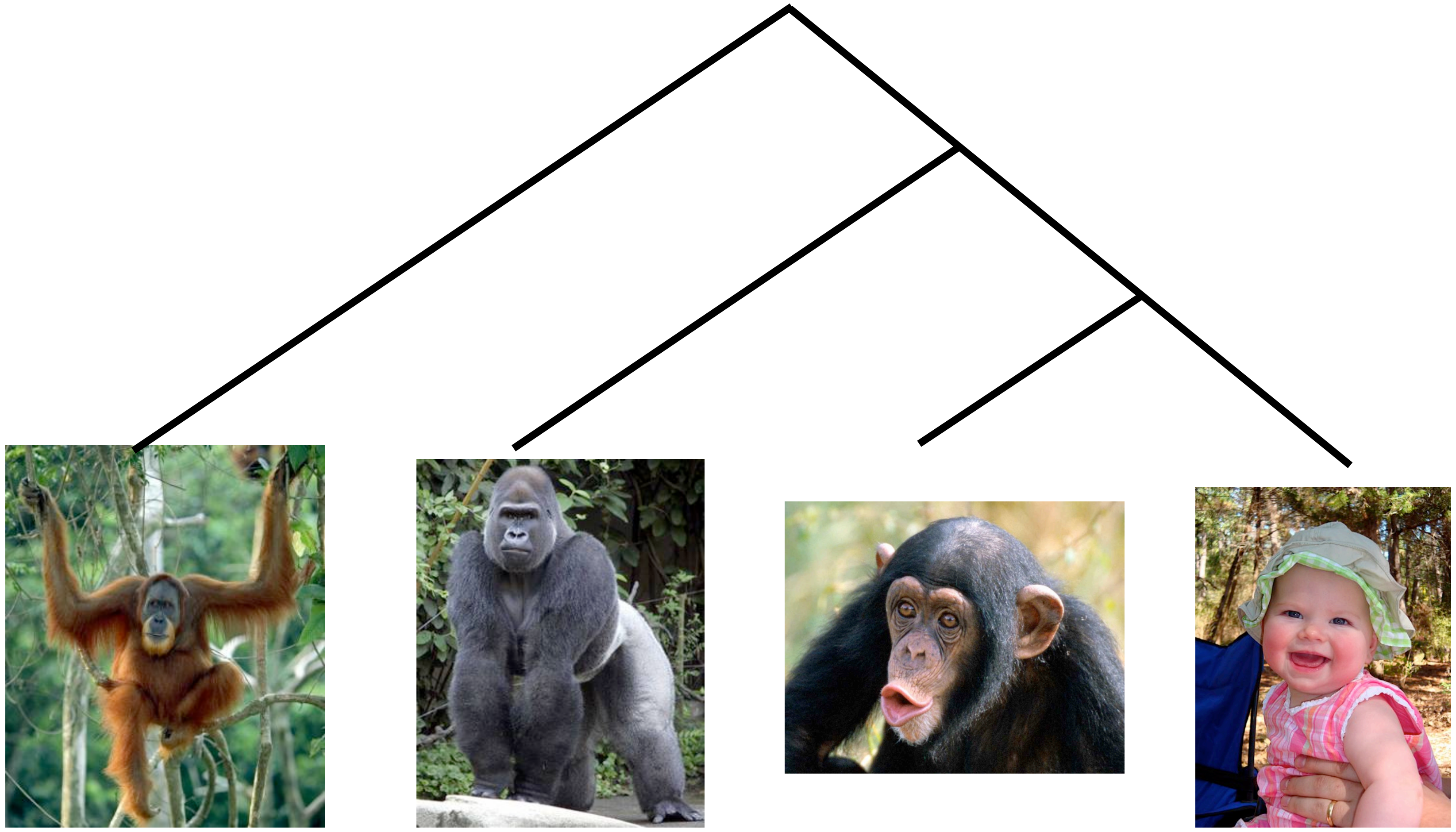



U
V
W
X
Y
AGGGCAT
TAGCCCA
TAGACTT
TGCACAA
TGCGCTT
X
U
Y
V
W

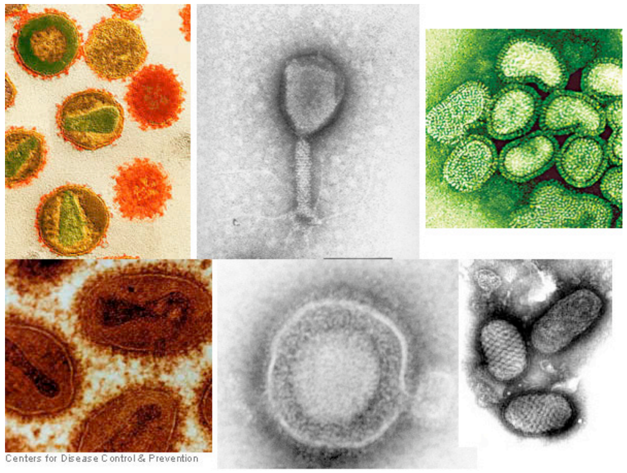
Harder problems!



Harder problems need DNA!



USA 33
Black widow
Elderberry longhorn
Lady beetle
Yellow garden spider
Dogbane beetle
Flower fly
Assassin bug
Ebony jewelwing
Velvet ant
Monarch caterpillar
Monarch butterfly
Eastern Hercules beetle
Bombardier beetle
Dung beetle
Spotted water beetle
True katydid
Spinybacked spider
Periodical cicada
Scorpionfly
Jumping spider

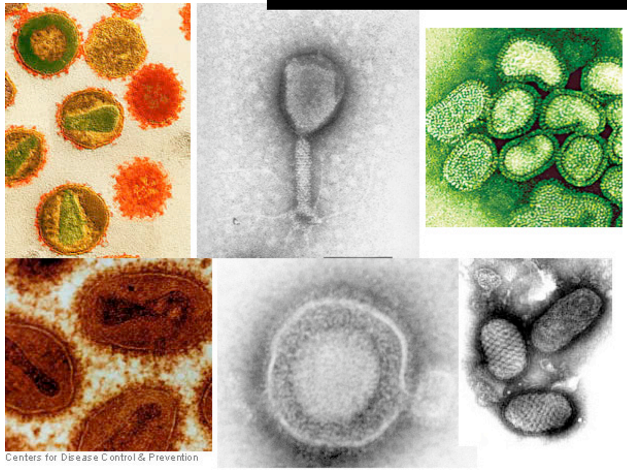

Centers for Disease Control & Prevention

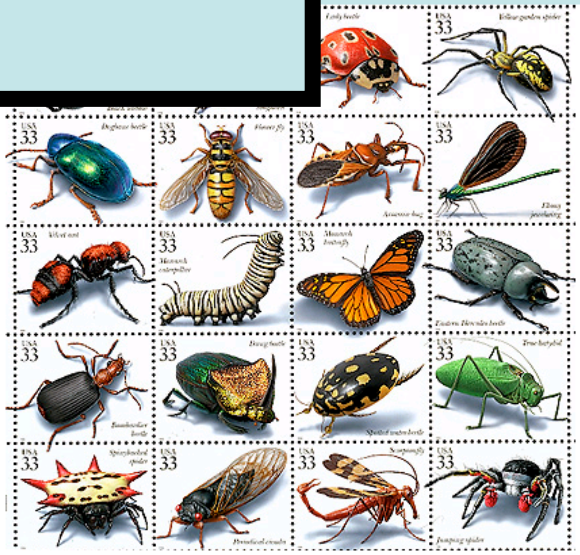
Many, Many Trees

# of Species	# of Unrooted Trees
4	3
5	15
6	105
7	945
8	10,395
9	135,135
10	2,027,025
20	2.2×10^{20}
100	4.5×10^{190}
1000	2.7×10^{2900}

8+ million species

NP-hard problems





Today (this lecture)

- What is a computational problem?
- What is an algorithm?
- How to design and analyze algorithms
- What NP-hardness means (and what to do about it)
- My research (phylogeny estimation)

Some computational problems

1. Given a list of numbers, put it into sorted order
2. Given a map and a collection of cities, find the shortest tour that visits every city
3. Given a collection of people, find the largest subset of them that all know each other
4. Given a collection of people, find the smallest number of groups so that no two people in the same group know each other.

Some computational problems

1. Given a list of numbers, put it into sorted order
2. Given a map and a collection of cities, find the shortest tour that visits every city
3. Given a collection of people, find the largest subset of them that all know each other
4. Given a collection of people, find the smallest number of groups so that no two people in the same group know each other.

Which ones can be solved in polynomial time?

Sorting

- Given a list of n numbers, put it into sorted order
- Algorithm: find smallest number, and put it in the front of the list. Repeat the process on the last n-1 numbers.
- Running time: **$O(n^2)$** (polynomial time)

Some computational problems

1. Given a list of numbers, put it into sorted order
2. Given a map and a collection of cities, find the shortest tour that visits every city
3. Given a collection of people, find the largest subset of them that all know each other
4. Given a collection of people, find the smallest number of groups so that no two people in the same group know each other.

Which ones can be solved in polynomial time?

Some computational problems

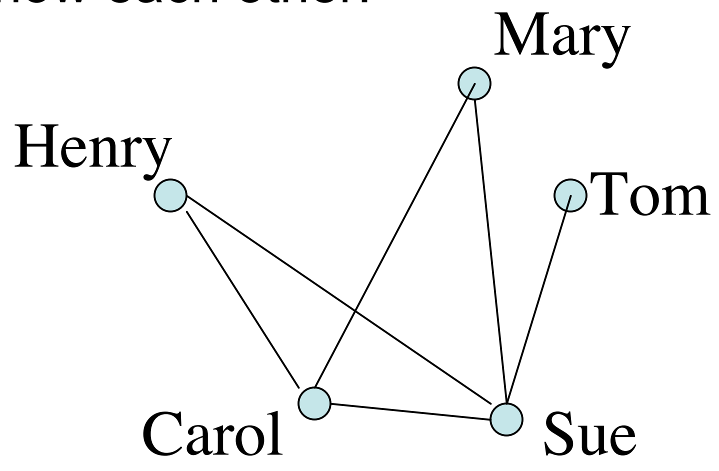
1. Given a list of numbers, put it into sorted order
2. Given a map and a collection of cities, find the shortest tour that visits every city
3. Given a collection of people, find the largest subset of them that all know each other
4. Given a collection of people, find the smallest number of groups so that no two people in the same group know each other.

Which ones can be solved in polynomial time?

Is this problem polynomial?

Problem: Given a collection of people, determine if they can be put into 2 groups so that no two people in the same group know each other

Graph-theoretic representation: Create a graph with vertices for the people, and edges between vertices if the two people know each other!



2-coloring

- **2-colorability**: Given graph G = (V,E), determine if we can assign colors **red** and **blue** to the vertices of G so that no edge connects vertices of the same color.

2-coloring

- **2-colorability**: Given graph G = (V,E), determine if we can assign colors **red** and **blue** to the vertices of G so that no edge connects vertices of the same color.

- Greedy Algorithm. Start with one vertex and make it **red**, and then make all its neighbors **blue**, and keep going. If you succeed in coloring the graph without making two nodes of the same color adjacent, the graph can be 2-colored.

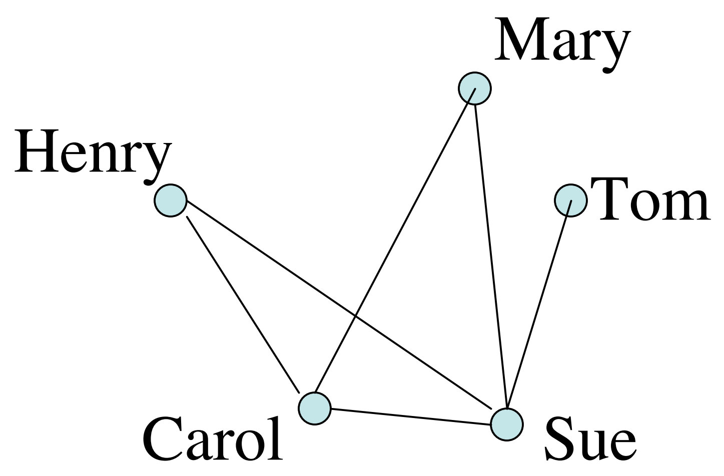
2-coloring

- **2-colorability**: Given graph G = (V,E), determine if we can assign colors **red** and **blue** to the vertices of G so that no edge connects vertices of the same color.

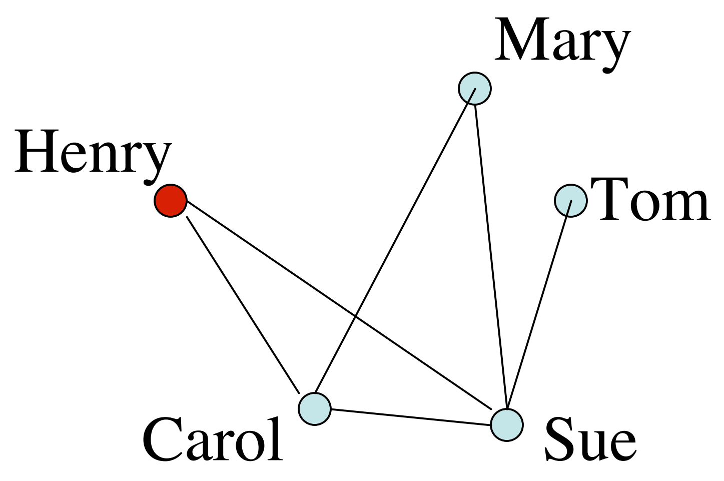
- Greedy Algorithm. Start with one vertex and make it **red**, and then make all its neighbors **blue**, and keep going. If you succeed in coloring the graph without making two nodes of the same color adjacent, the graph can be 2-colored.

- Running time: **$O(n^2)$ time, where n is the number of vertices.**

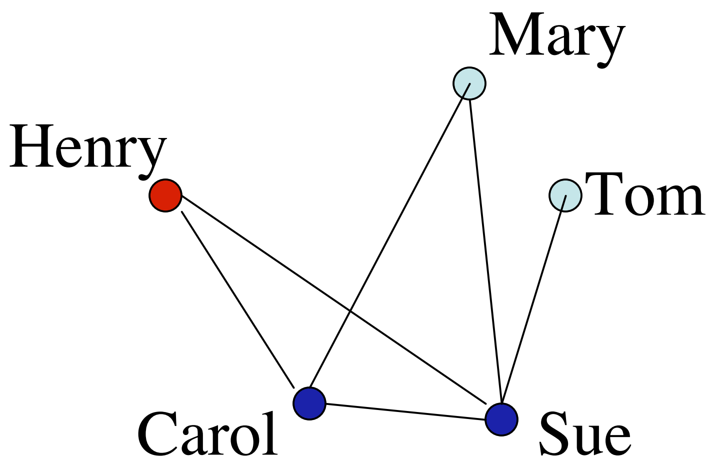
Can we group this set into two groups so that no two people know each other?
Or Can we 2-color the graph?



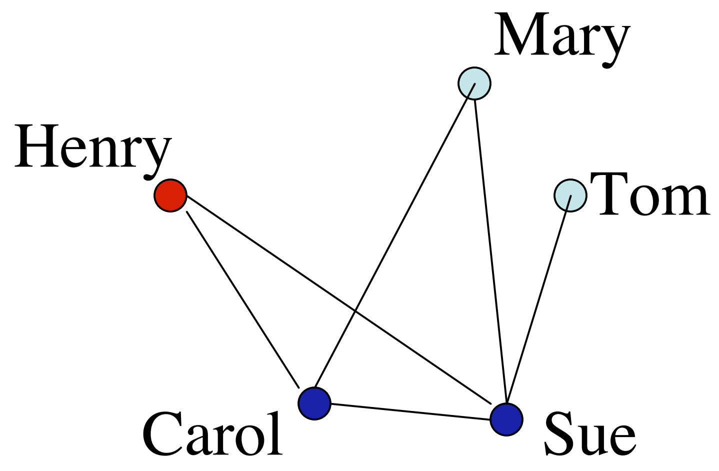
Can we group this set into two groups so that no two people know each other?
Or Can we 2-color the graph?



Can we group this set into two groups so that no two people know each other?
Or Can we 2-color the graph?



Can we group this set into two groups so that no two people know each other? *Or* Can we 2-color the graph?



No! We cannot!

What about this?

- **3-colorability:** Given graph G, determine if we can assign **red, blue,** and **green** to the vertices in G so that no edge connects vertices of the same color.

What about this?

- **3-colorability:** Given graph G, determine if we can assign **red,** **blue,** and **green** to the vertices in G so that no edge connects vertices of the same color.

A brute-force solution seems to require **$O(3^n)$** time, where n is the number of vertices.

- Some decision problems can be solved in polynomial time:
  - Can graph G be 2-colored?
- Some decision problems *seem* to not be solvable in polynomial time:
  - Can graph G be 3-colored?
  - Does graph G have a Hamiltonian cycle (a cycle that visits every vertex exactly once)?

In fact, some problems are "NP-hard"

- **3-colorability:** Given graph G, determine if we can assign **red, blue,** and **green** to the vertices in G so that no edge connects vertices of the same color.

- 3-colorability is provably NP-hard. What does this mean?

Most computer scientists are willing to bet that no NP-hard problem can be solved in polynomial time.

Therefore, the options are:

- Solve the problem *exactly* (but use lots of time on some inputs)
- Use *heuristics* which may not solve the problem correctly (and which might be computationally expensive, anyway)

Computational problems in Biology are almost always NP-hard!

In particular, inferring evolutionary trees generally involves trying to solve NP-hard problems.

My research

Methods that produce accurate
phylogenetic trees
on hard-to-analyze datasets
(thousands of sequences)
within reasonable times

Problem: all the “good” methods require finding “good” solutions to NP-hard optimization problems!

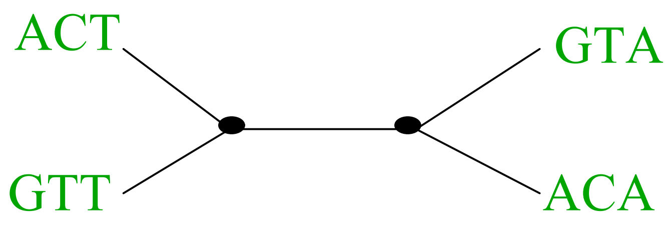
Maximum Parsimony

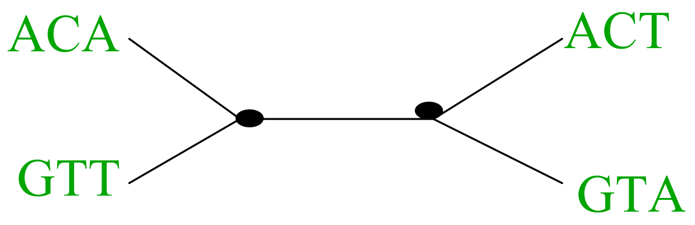
- Given a set of DNA sequences
- Find a tree for the sequences with the minimum total number of changes

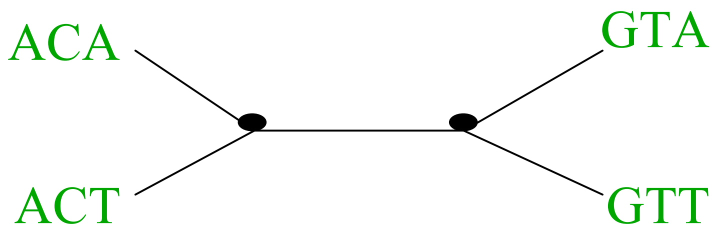
Maximum parsimony (example)

- **Input**: Four sequences
  - ACT
  - ACA
  - GTT
  - GTA
- **Question**: which of the three trees has the best MP scores?

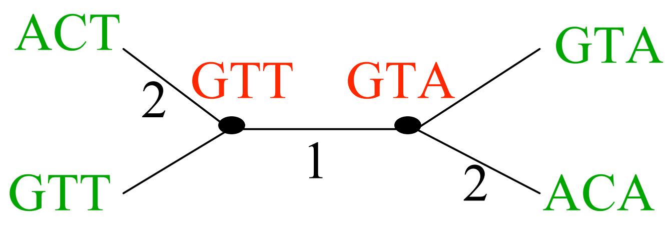
Maximum Parsimony



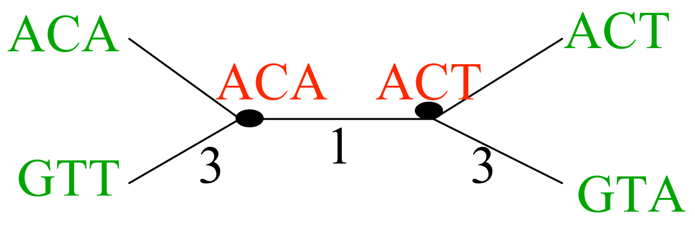




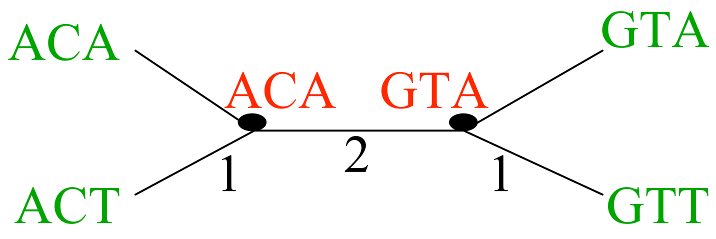
Maximum Parsimony



MP score = 5



MP score = 7

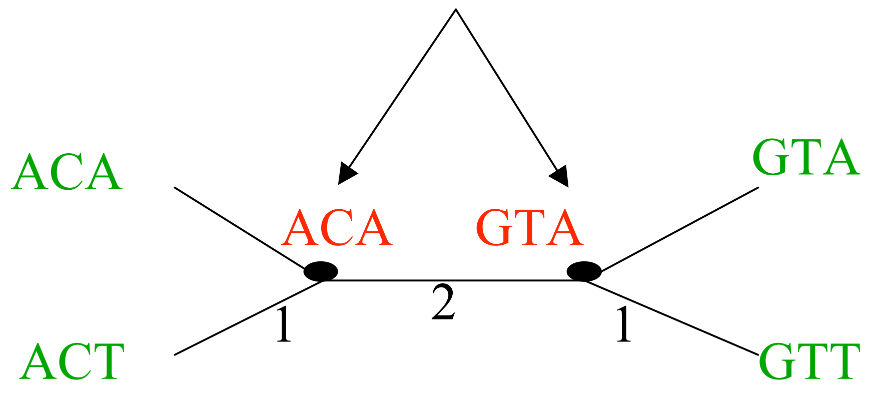


MP score = 4

Optimal MP tree

Maximum Parsimony

Optimal labeling can be computed in polynomial time using Dynamic Programming



MP score = 4

Finding the optimal MP tree is **NP-hard**

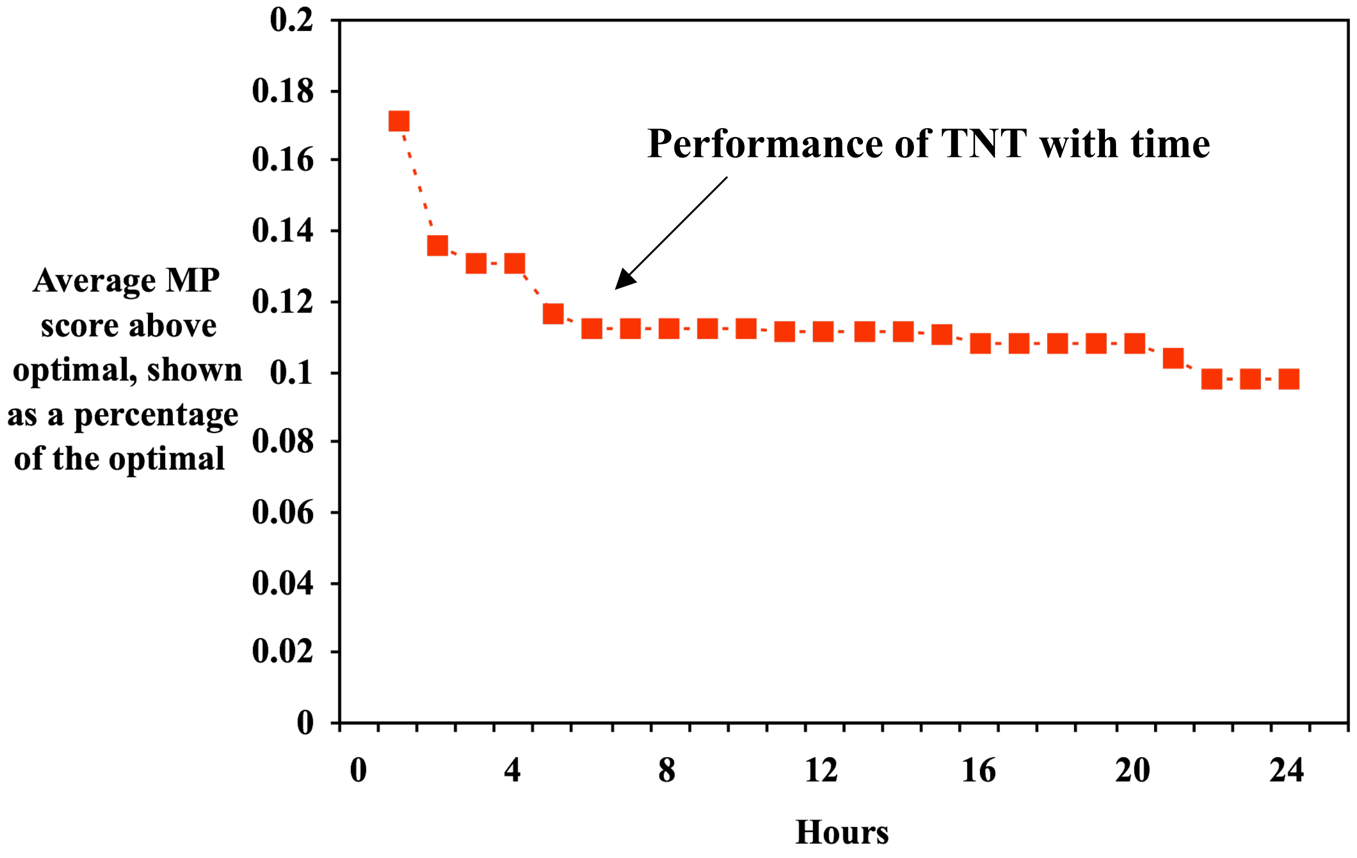
Solving NP-hard problems exactly is … unlikely

- The number of (unrooted) binary trees on *n* leaves is (2n-5)!!

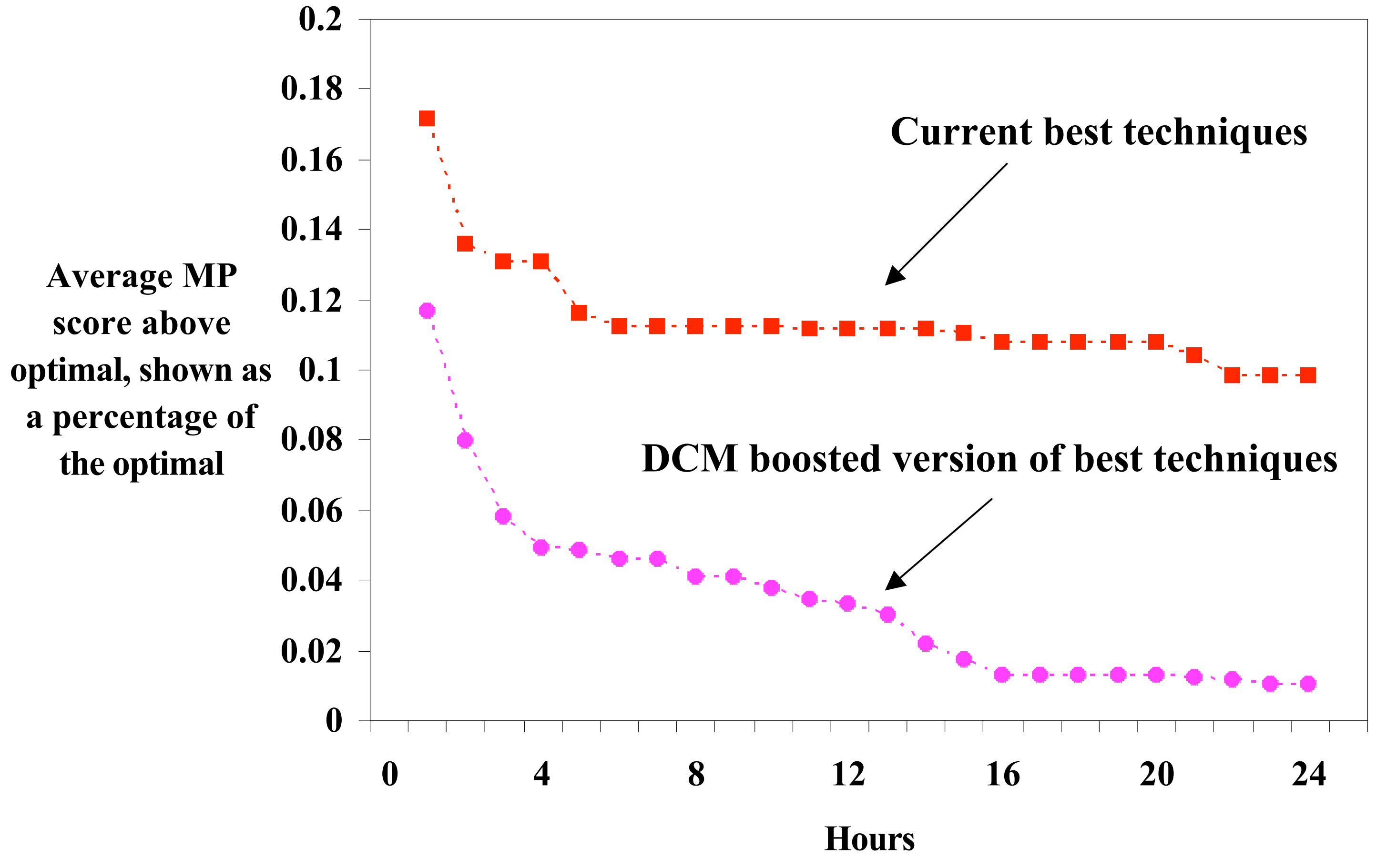
#leaves	#trees
4	3
5	15
6	105
7	945
8	10395
9	135135
10	2027025
20	2.2×10^{20}
100	4.5×10^{190}
1000	2.7×10^{2900}

Problems with techniques for MP and ML

Shown here is the performance of a TNT heuristic maximum parsimony analysis on a real dataset of almost 14,000 sequences. ("Optimal" here means ***best score to date***, using any method for any amount of time.) Acceptable error is below 0.01%.



Research: we try to develop better heuristics



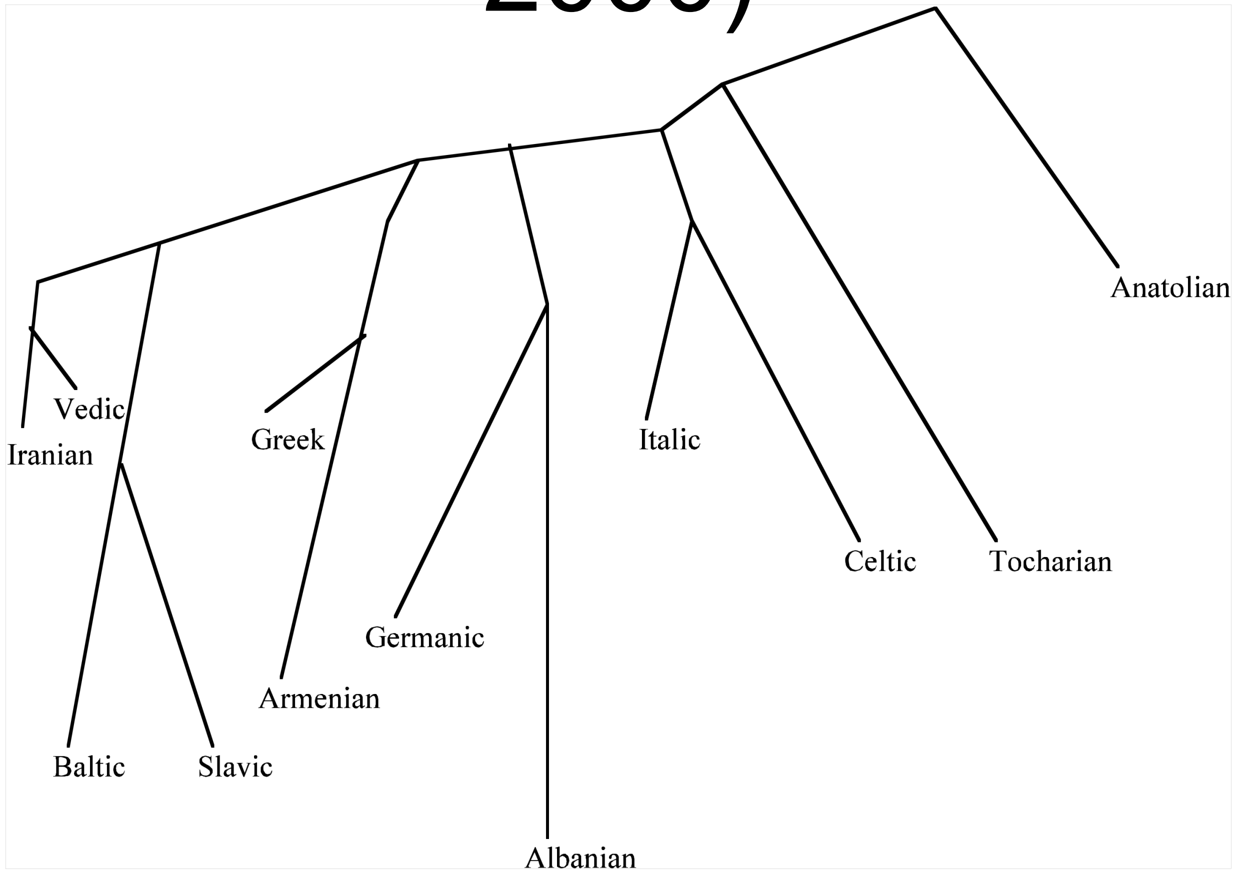
Comparison of TNT to Rec-I-DCM3(TNT) on one large dataset

Other problems I study

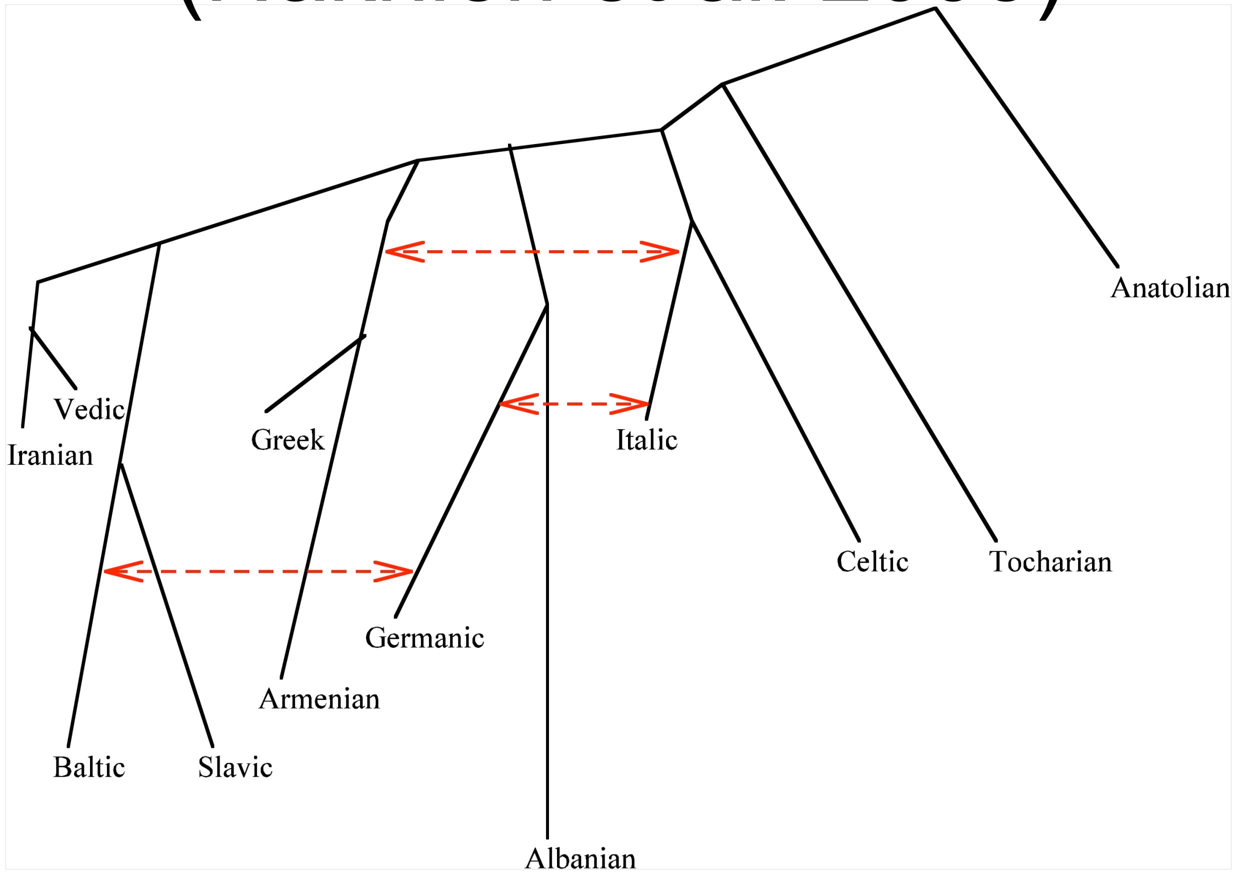
- Multiple sequence alignment
- Detecting Horizontal Gene Transfers (and hybrid species)
- Whole genome evolution
- Evolution of languages and human origins

And more!

Possible Indo-European tree (Ringe, Warnow and Taylor 2000)



Possible IE Phylogenetic Network (Nakhleh et al. 2005)



Computational biology research is fun, multi-disciplinary, and collaborative!

- Software development
- Mathematics
- Probability and Statistics
- Biology
- Chemistry
- Linguistics

Plus, you will get to travel to far away lands

My research group

- Tandy Warnow (UT-Austin)
- Randy Linder (UT-Austin)
- UT PhD Students: Serita Nelesen, Kevin Liu, Sindhu Raghavan, Shel Swenson
- Collaborators at many other universities around the world