Computer Science and Reconstructing Evolutionary Trees

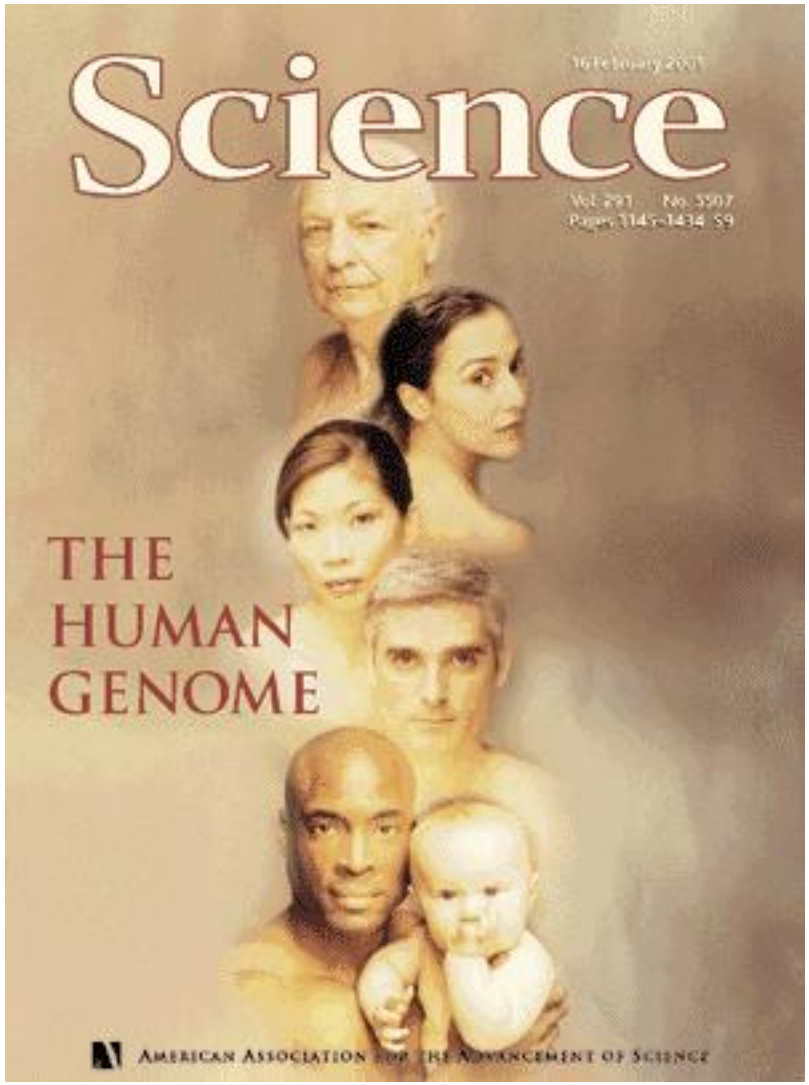
Tandy Warnow
Department of Computer Science
University of Illinois at Urbana-Champaign


BIG DATA
A REVOLUTION
THAT WILL TRANSFORM HOW WE LIVE, WORK, AND THINK
VIKTOR MAYER-SCHÖNBERGER
KENNETH CUKIER


Data
terabytes
mobile
time
databases
tools
store
now
query
big
compression
Twitter
Hadoop
NoSQL
SQL
big-data
new
support
database
storage
processing
information
data
analysis
analyses
example
column-store

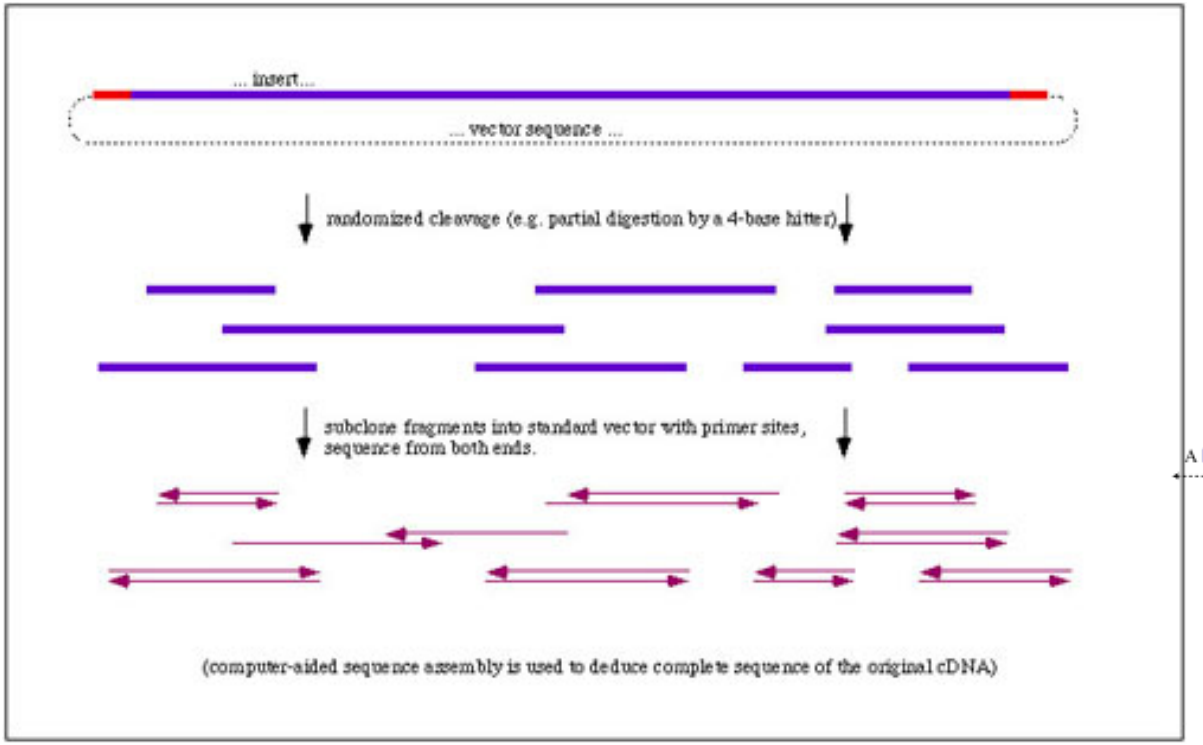

IT COMES FROM EVERYWHERE. IT KNOWS ALL.
ACCORDING TO THE BOOK OF WIKIPEDIA, ITS NAME IS "BIG DATA."
BIG DATA
BUT IF WE ACCEPT BIG DATA IN OUR SERVERS, WE WILL BE SAVED FROM BANKRUPTCY.
LET US PAY.

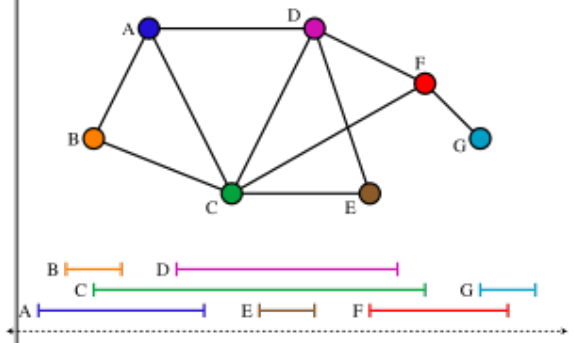
BigData for Biology: Genomics



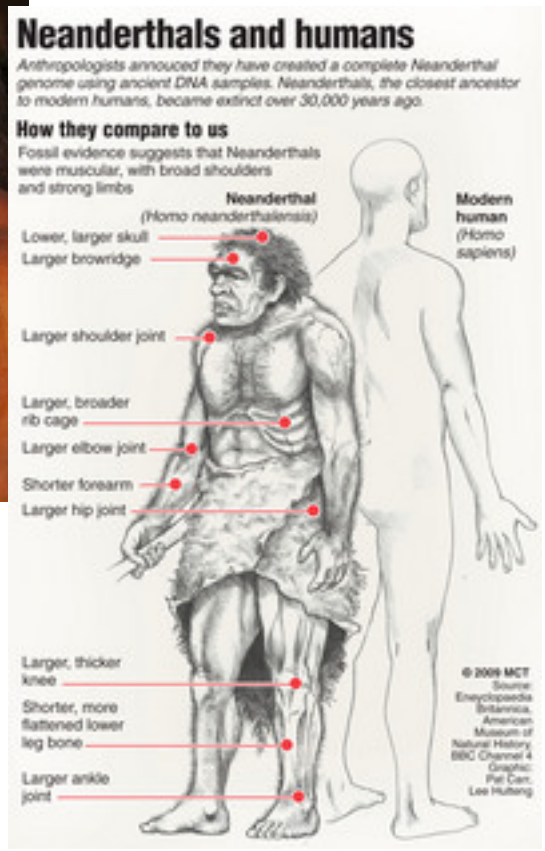
Whole Genome Sequencing:

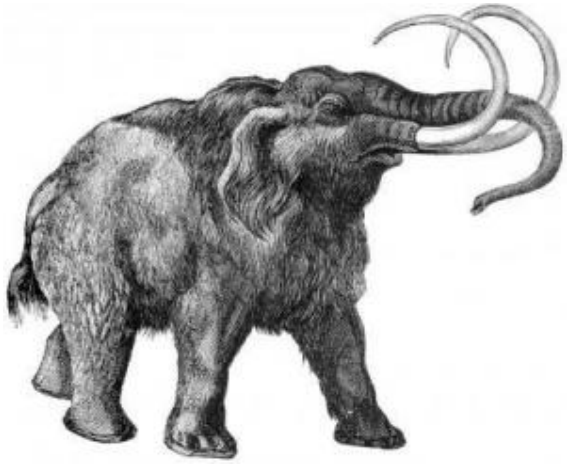
Graph Algorithms and Combinatorial Optimization!





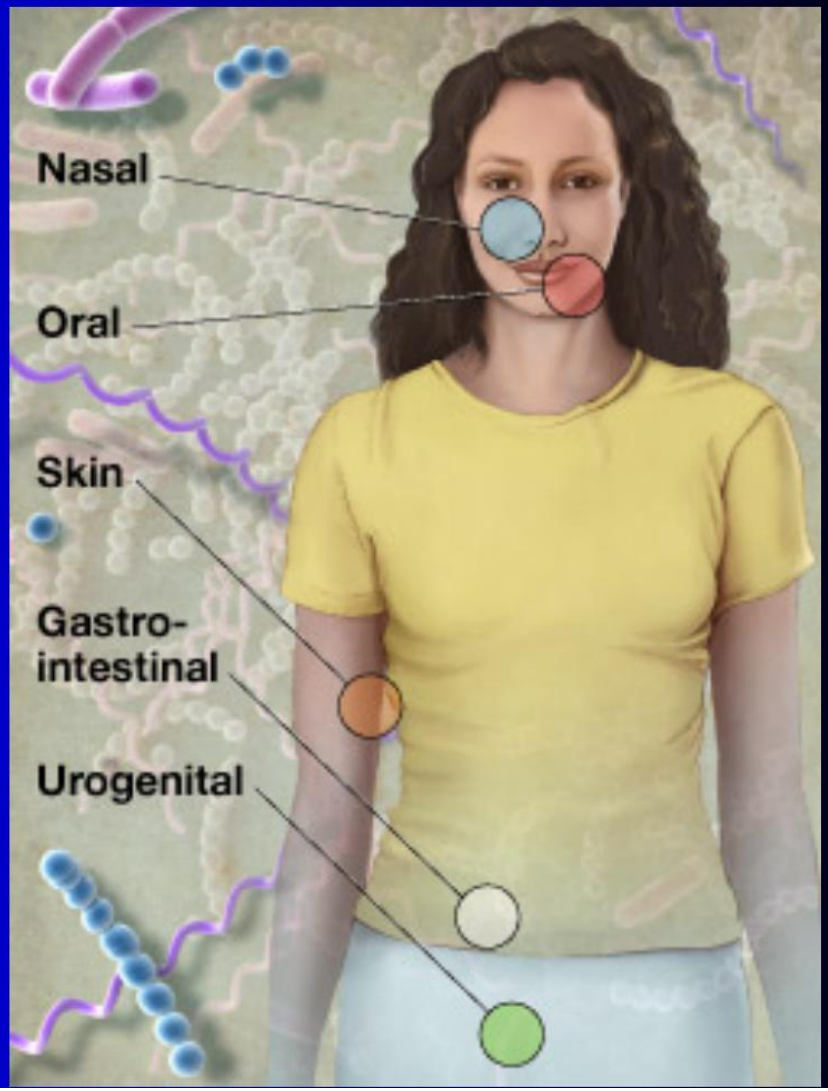
Other Genome Projects! (Neandertals, Wooly Mammoths, and more ordinary creatures…)



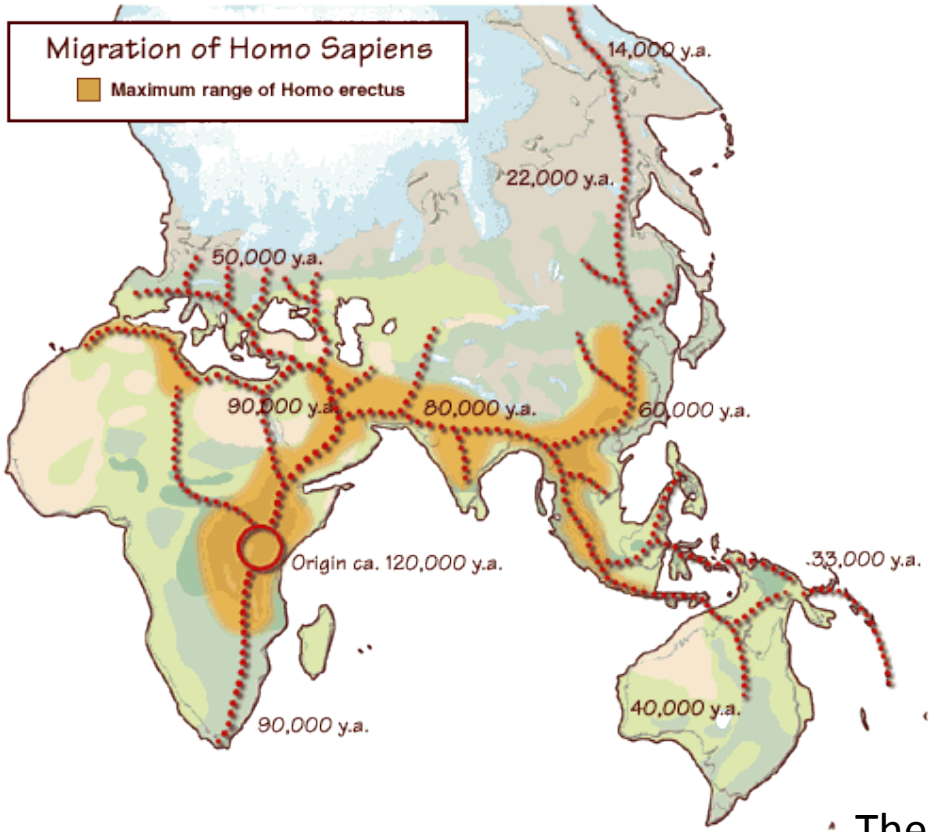


The NIH Human Microbiome Project

25,000 human genes,
1,000,000 bacterial genes

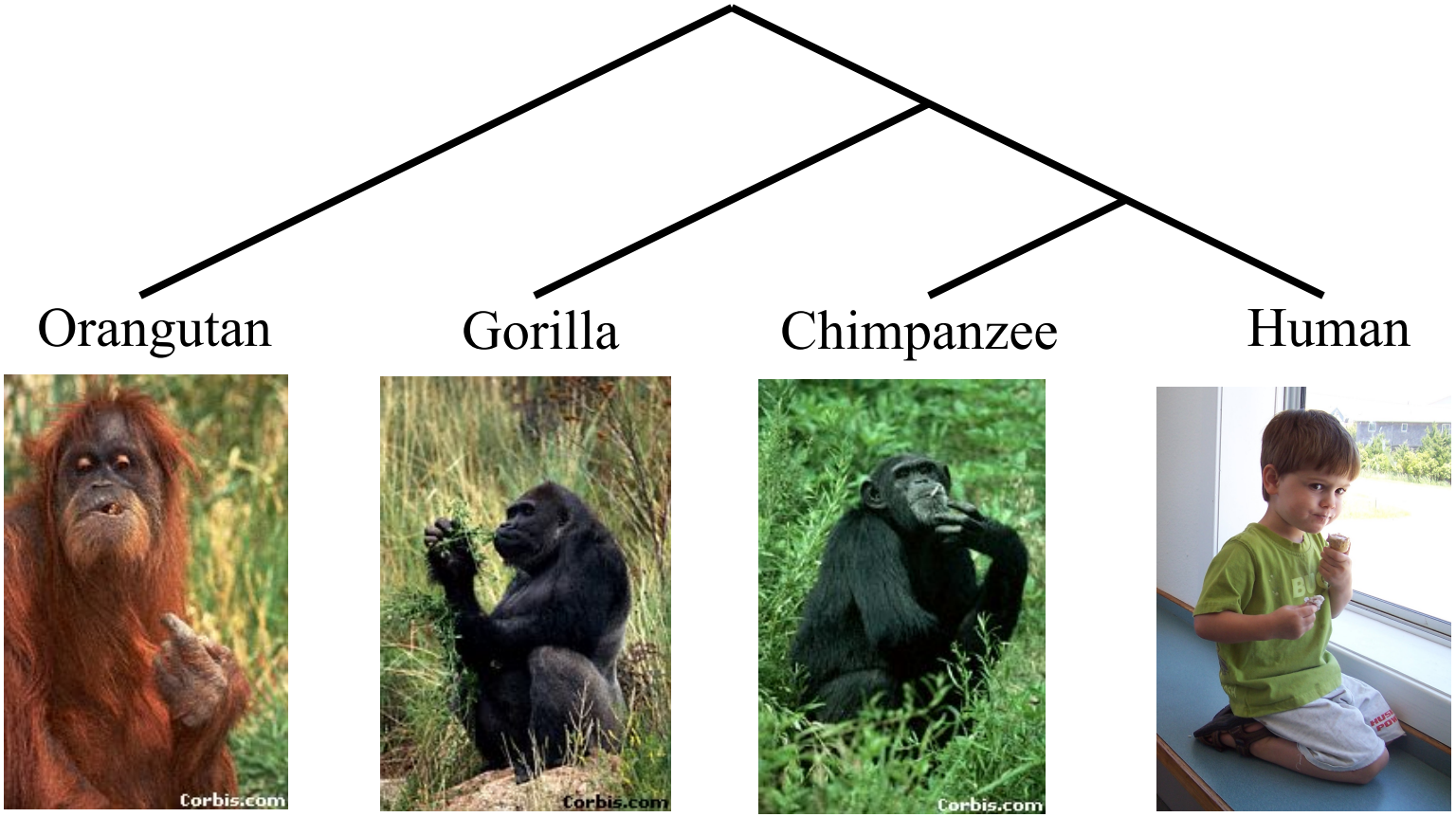


Where did humans come from, and how did they move throughout the globe?



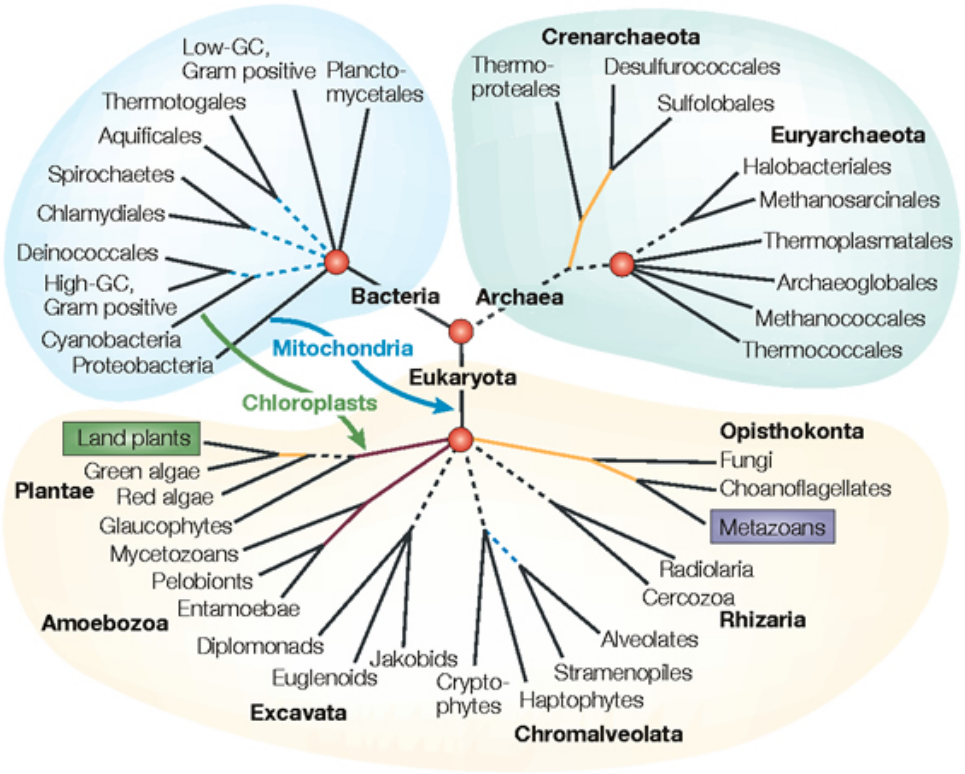
- The 1000 Genome Project: using human genetic variation to better treat diseases

Phylogeny (evolutionary tree)



From the Tree of the Life Website,
University of Arizona

Assembling the Tree of Life



Nature Reviews | Genetics

Goal: Constructing a Tree of Life

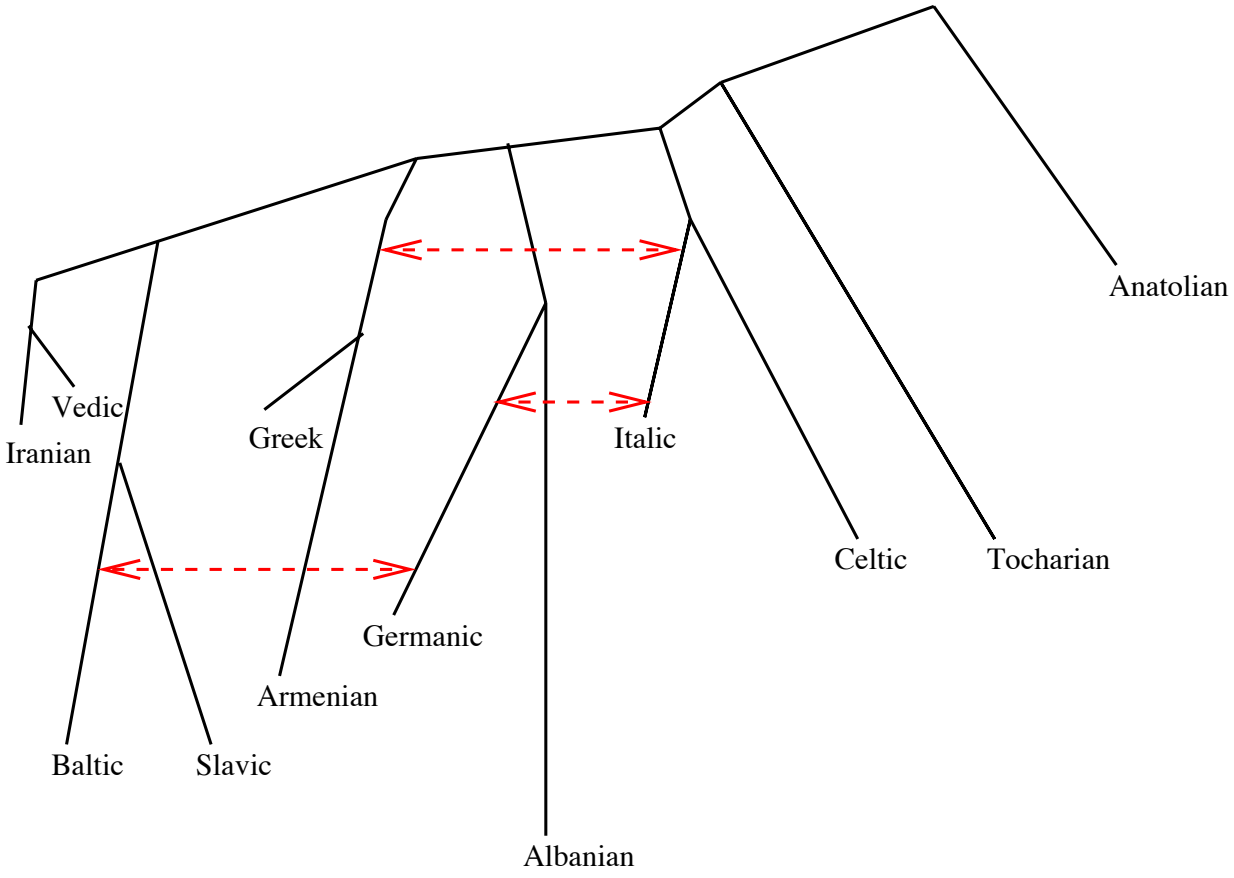
- Why do we want to do this?
- What are the computer science challenges?
- What kinds of techniques help us do this well?

Why? Because!

- Evolutionary history relates all organisms and genes, and helps us understand and predict
  - interactions between genes (genetic networks)
  - drug design
  - predicting functions of genes
  - influenza vaccine development
  - origins and spread of disease
  - origins and migrations of humans

"Nothing in biology makes sense except in the light of evolution"
Dobzhansky

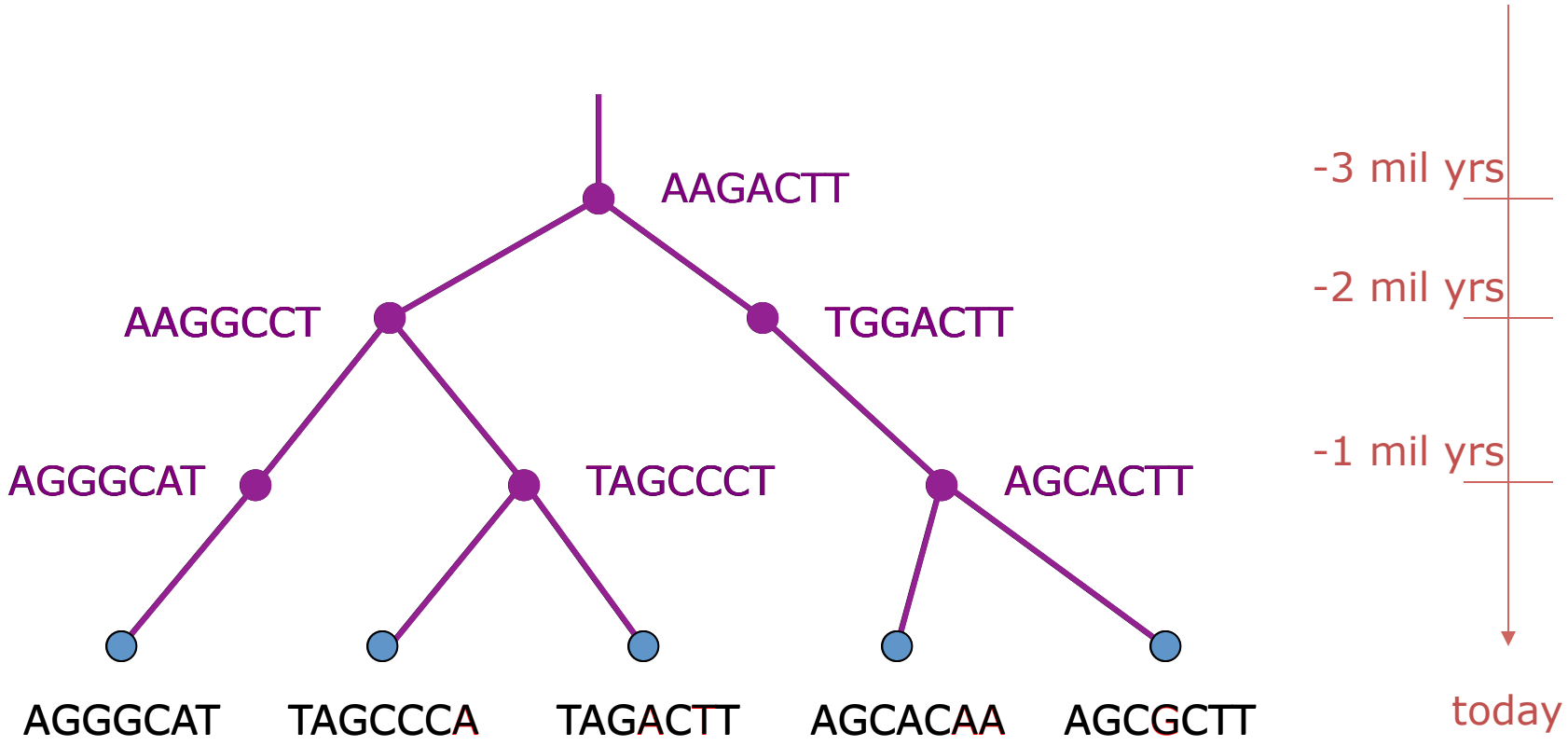
Indo-European Language Evolution (Nakhleh et al.)



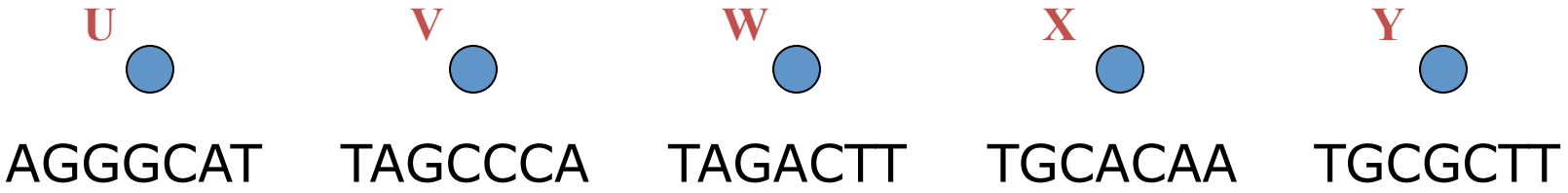
Main Points

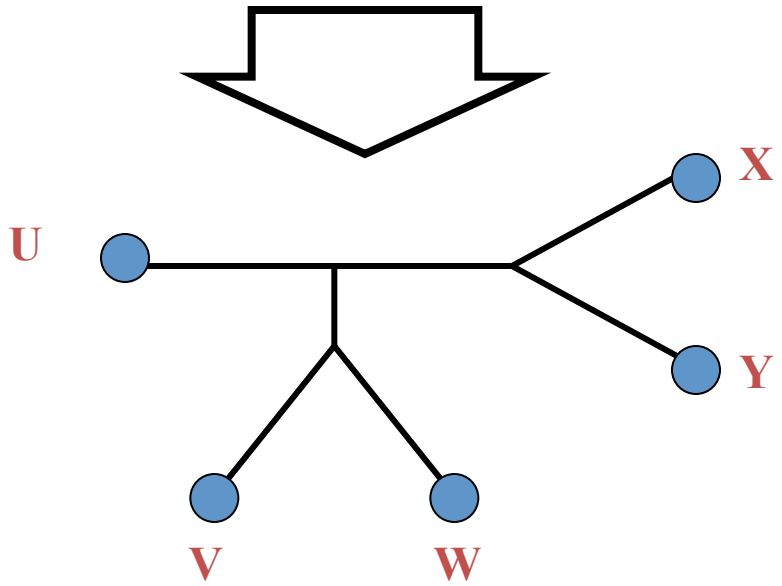
- Computer scientists develop algorithms and software to make it possible for scientists to get improved accuracy in their analyses.
- These algorithms involve creative strategies, including divide-and-conquer, iteration, and randomization.
- Extensive simulations and data analyses are part of the evaluation process!

DNA Sequence Evolution



Phylogeny Problem





Tree Construction

- **Input**: Four sequences
  - ACT
  - ACA
  - GTT
  - GTA
- **Question**: what is the best tree for this dataset?

Maximum Parsimony

Maximum parsimony:

- Input: set S of sequences (all of the same length),
- Output: a tree T with the sequences at the leaves that minimizes the total number of changes

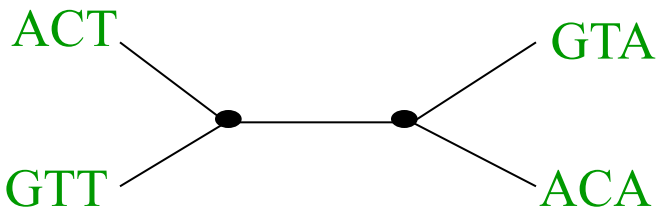
This is an optimization problem.

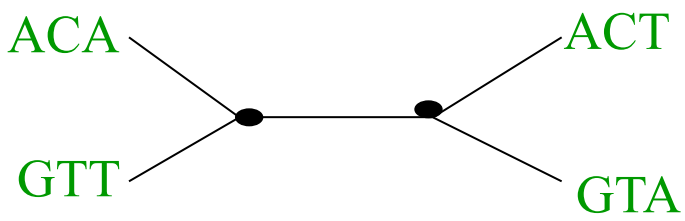
How hard is it to solve this problem exactly?

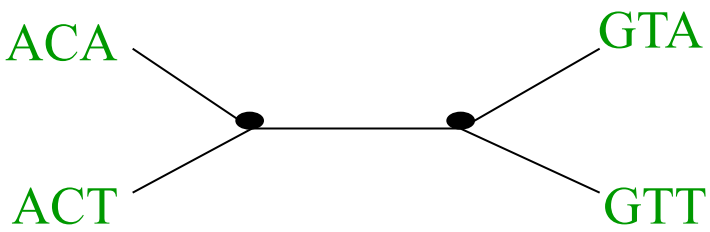
Maximum parsimony (example)

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

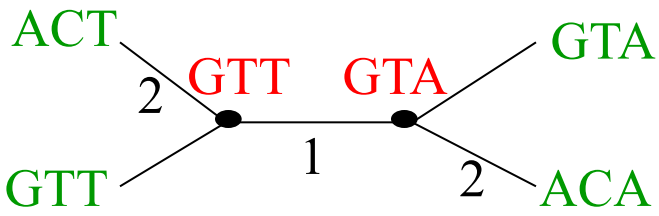
Maximum Parsimony



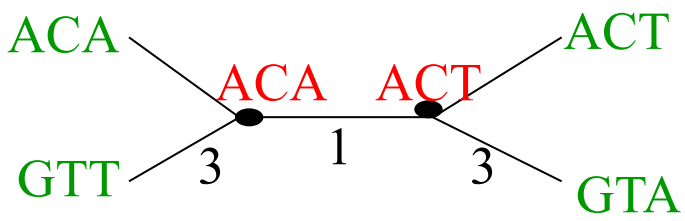




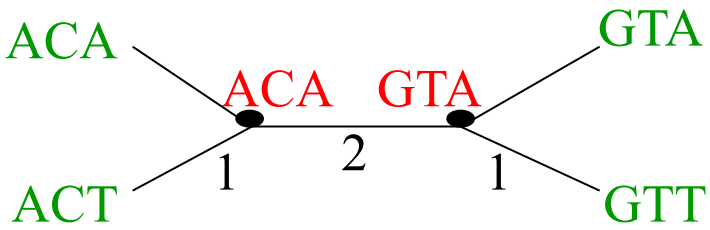
Maximum Parsimony



MP score = 5



MP score = 7

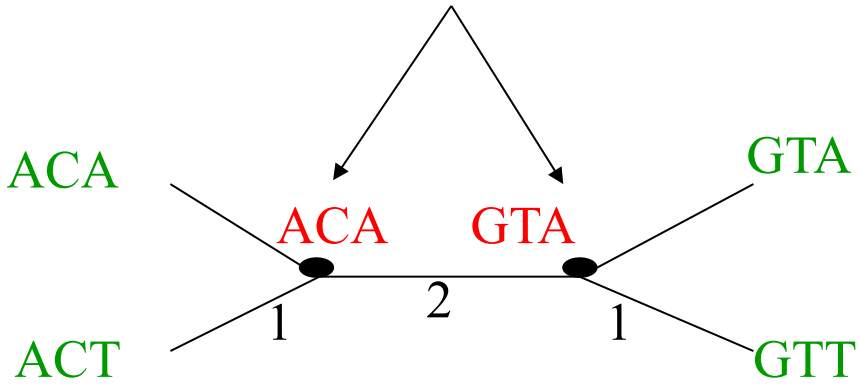


MP score = 4

Optimal MP tree

Maximum Parsimony: computational complexity

Optimal labeling can be computed in linear time O(nk)



MP score = 4

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

Exhaustive search is… exhausting

- Number of (unrooted) binary trees on *n* leaves is (2n-5)!!
- If each tree on **1000** taxa could be analyzed in **0.001** seconds, we would find the best tree in

  2890 millennia

#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}

NP-hard problems in Biology

- Optimization problems in biology are almost all NP-hard, and heuristics may run for months (or years!!) before finding *local optima*.

- The challenge here is to find better heuristics, since exact solutions are very unlikely to ever be achievable on large datasets.

Polynomial time problems

- A problem is polynomial time if it can be solved in time that runs (at worst) using a polynomial number of operations, in terms of the input size.
- Examples:
  - Are there ten people in this room with the same birthday?
  - Are there five people in this room who all like each other?
  - Find someone in the room who is friends with the most number of people in the room.
  - Can we divide the people in the room into two sets, so that no two people in the same set dislike each other?

A polynomial-time graph problem

- **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.

A polynomial-time graph problem

- **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.

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.

- Some decision problems can be solved in polynomial time:
  - Can graph G be 2-colored?
  - Does graph G have a Eulerian tour?
- Some decision problems *seem* to not be solvable in polynomial time:
  - Can graph G be 3-colored?
  - Does graph G have a Hamiltonian cycle?

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.

- This problem is provably NP-hard. What does this mean?

P vs. NP, continued

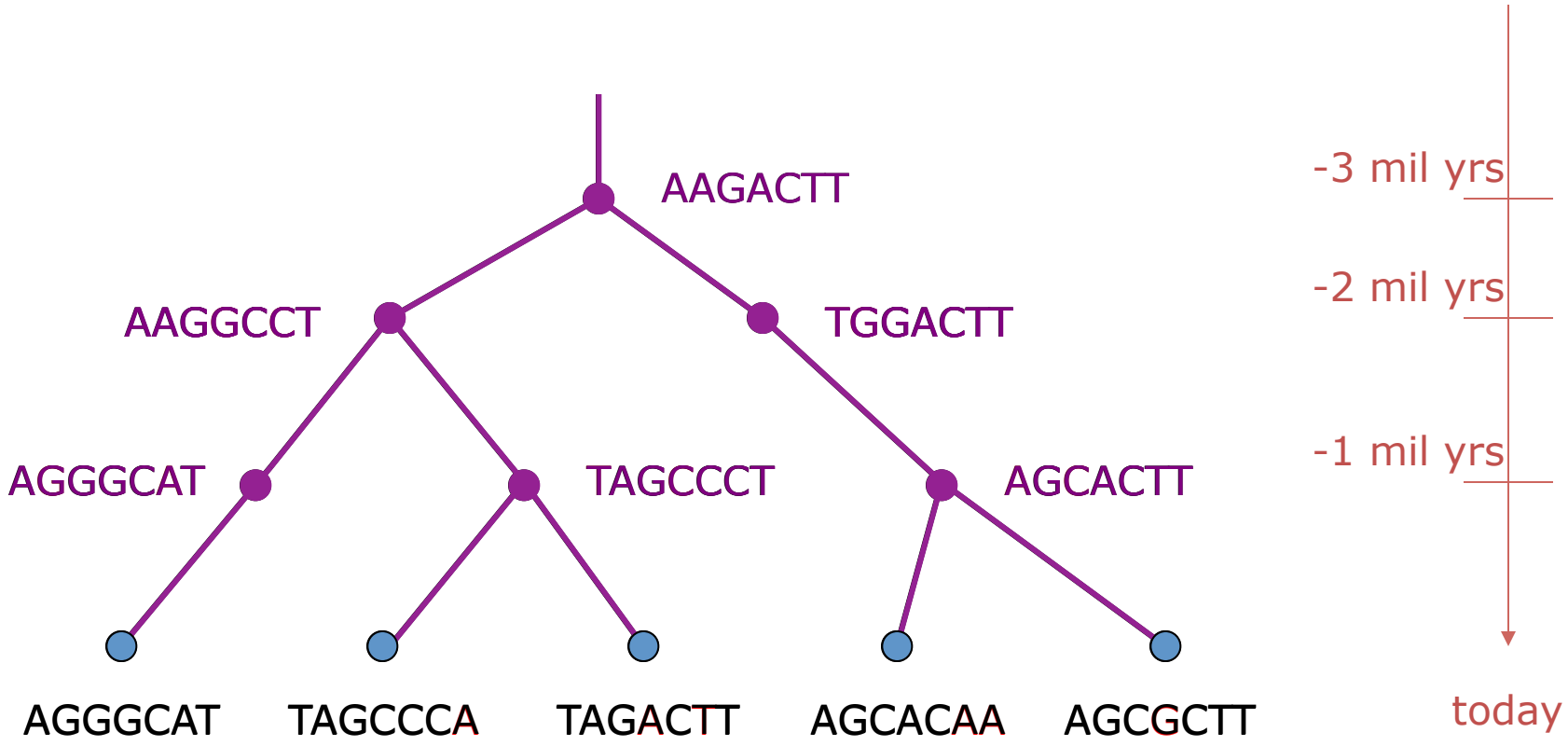
- The "big" question in theoretical computer science is:
  - Is it possible to solve an NP-hard problem in polynomial time?
- If the answer is "yes", then **all** NP-hard problems can be solved in polynomial time, so **P=NP**. This is generally not believed.

Coping with NP-hard problems

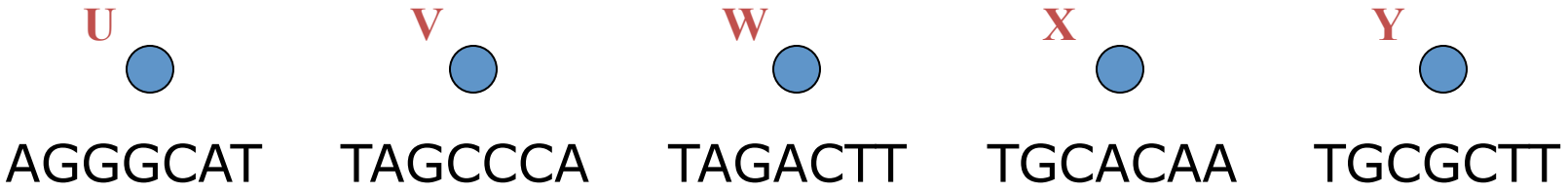
Since NP-hard problems may not be solvable in polynomial time, the options are:

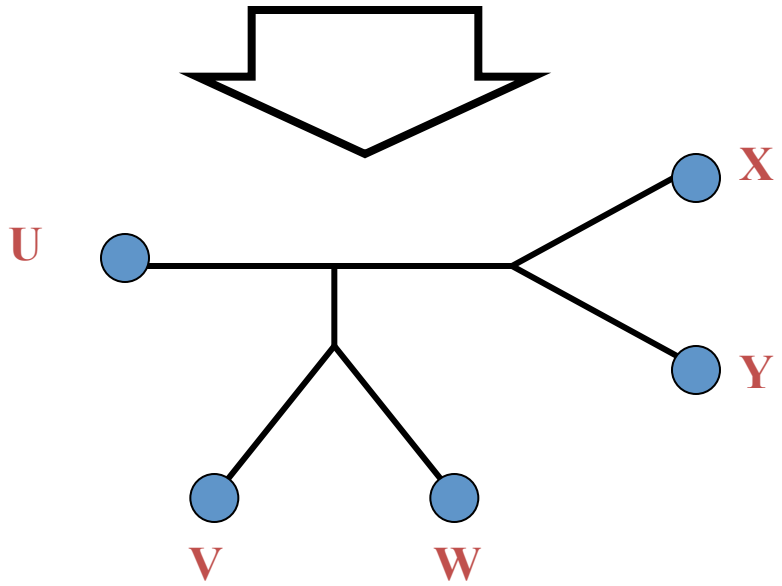
- Solve the problem *exactly* (but use lots of time on some inputs)
- Use *heuristics* which may not solve the problem exactly (and which might be computationally expensive, anyway)

DNA Sequence Evolution



Phylogeny Problem





Maximum Parsimony is NP-hard

Maximum parsimony:

- Given sequences (all of the same length),
- Find a tree with the sequences at the leaves that minimizes the total number of changes

NP-hard problems in Biology

- Optimization problems in biology are almost all NP-hard, and heuristics may run for months (or years!!) before finding *local optima*.

- The challenge here is to find better heuristics, since exact solutions are very unlikely to ever be achievable on large datasets.

Divide-and-Conquer

- Divide-and-conquer is a basic algorithmic trick for solving problems!

- Basic idea: divide a dataset into two or more sets, solve the problem on each set, and then combine solutions.

- Example: MergeSort to sort a list of k integers!

Divide-and-Conquer

- MergeSort solves the “Sorting Problem”: given a list of integers, put them into increasing order (smallest to largest).

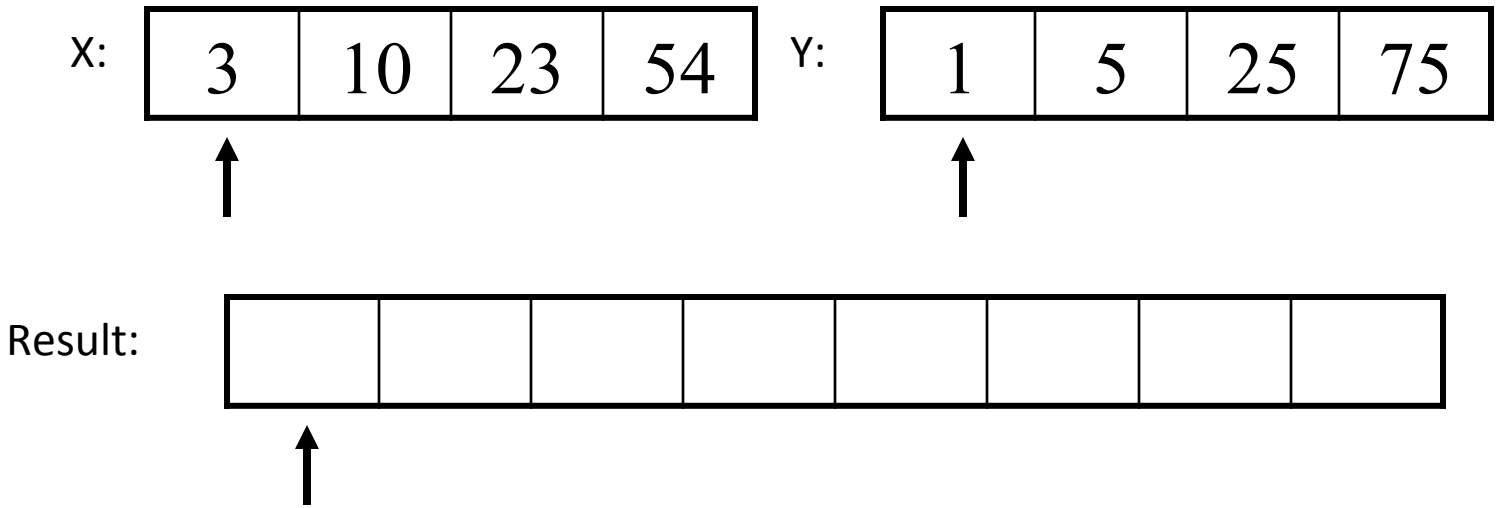
- This can be done by recursively dividing the unsorted list in half, applying MergeSort to each side, and then merging the right and left back together.

Merge Sort Algorithm

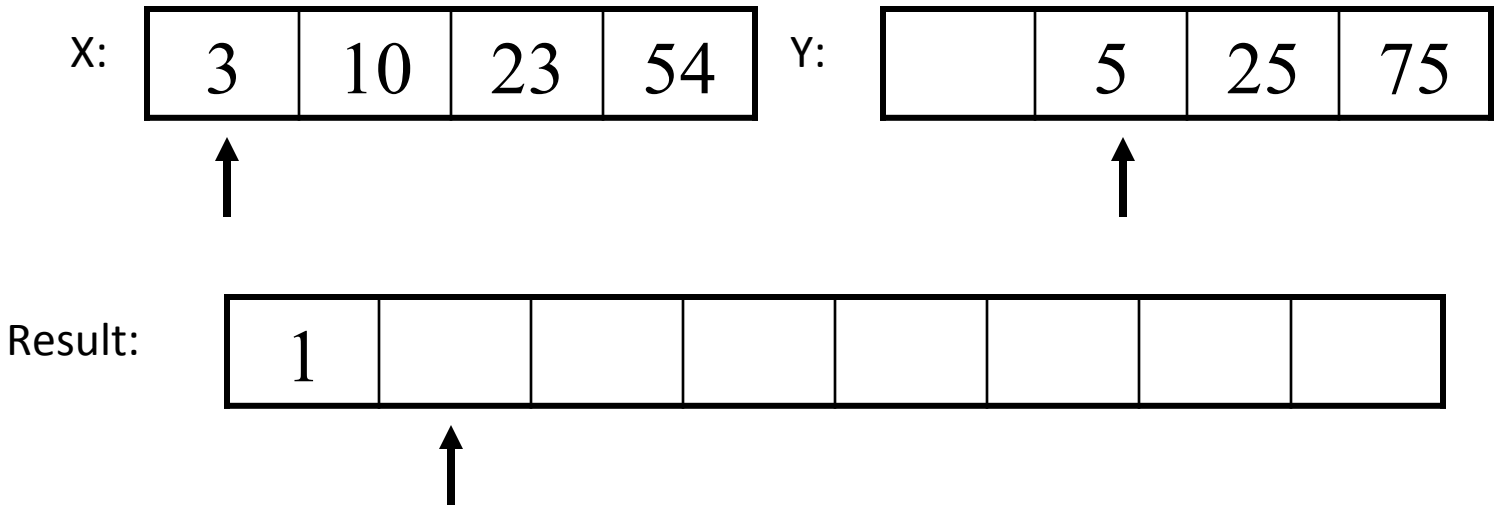
Given a list L with a length k:

- If k == 1 → the list is sorted
- Else:
  - Merge Sort the left side (0 through k/2)
  - Merge Sort the right side (k/2+1 thru k)
  - Merge the right side with the left side

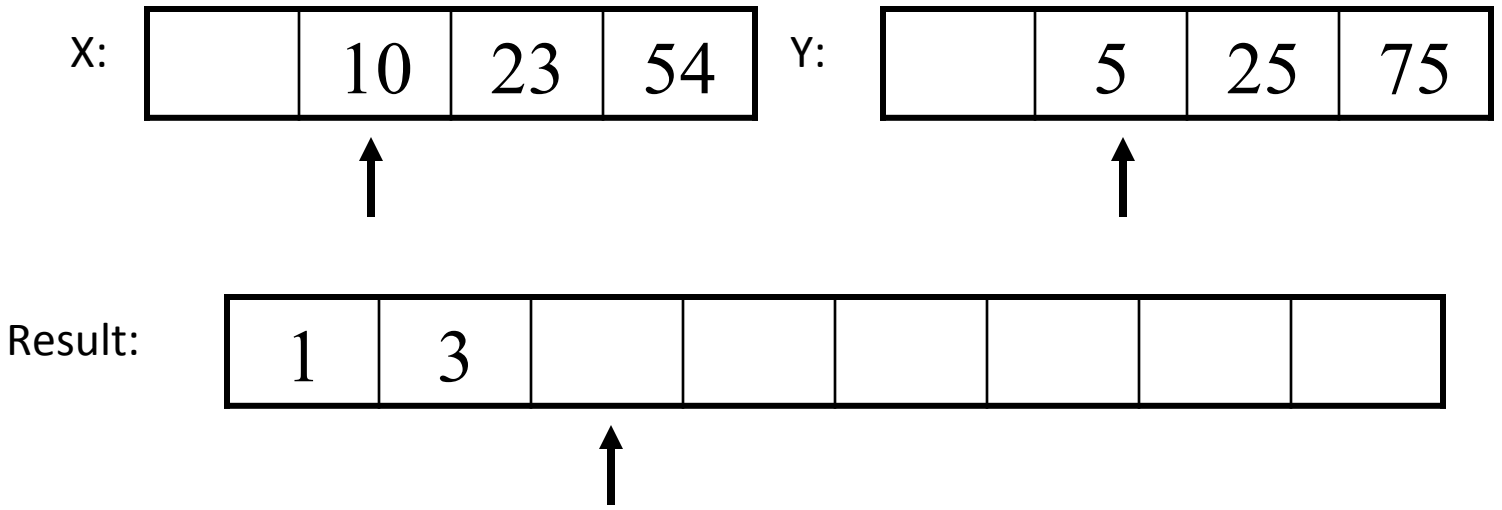
Merging two sorted lists



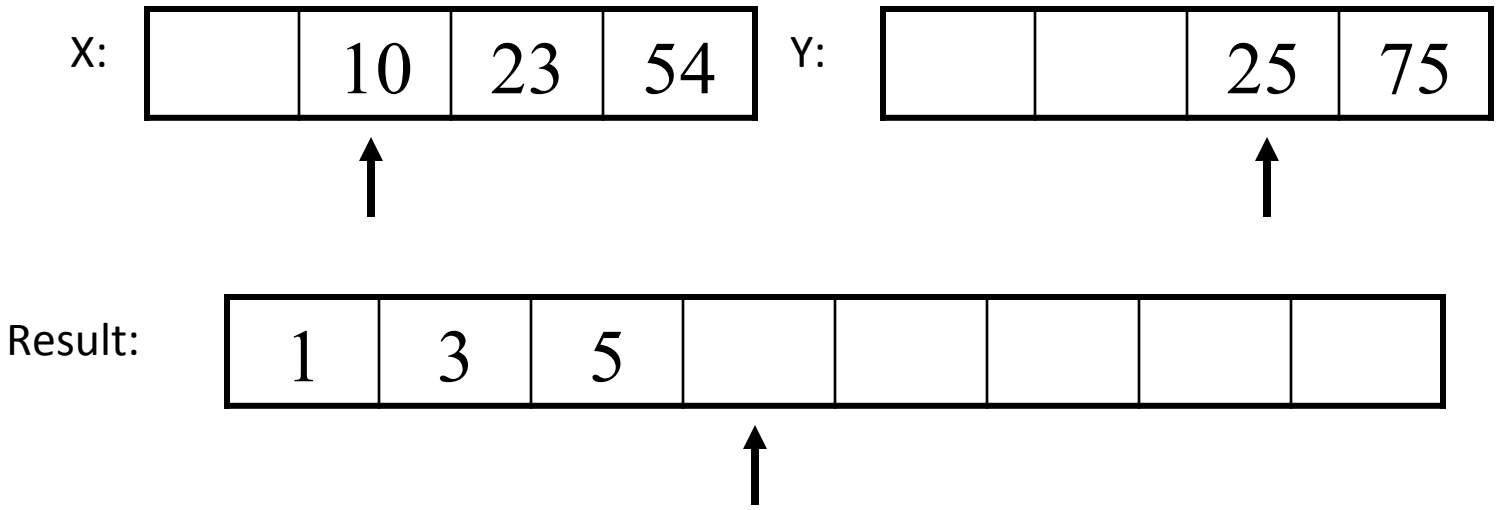
Merging (cont.)



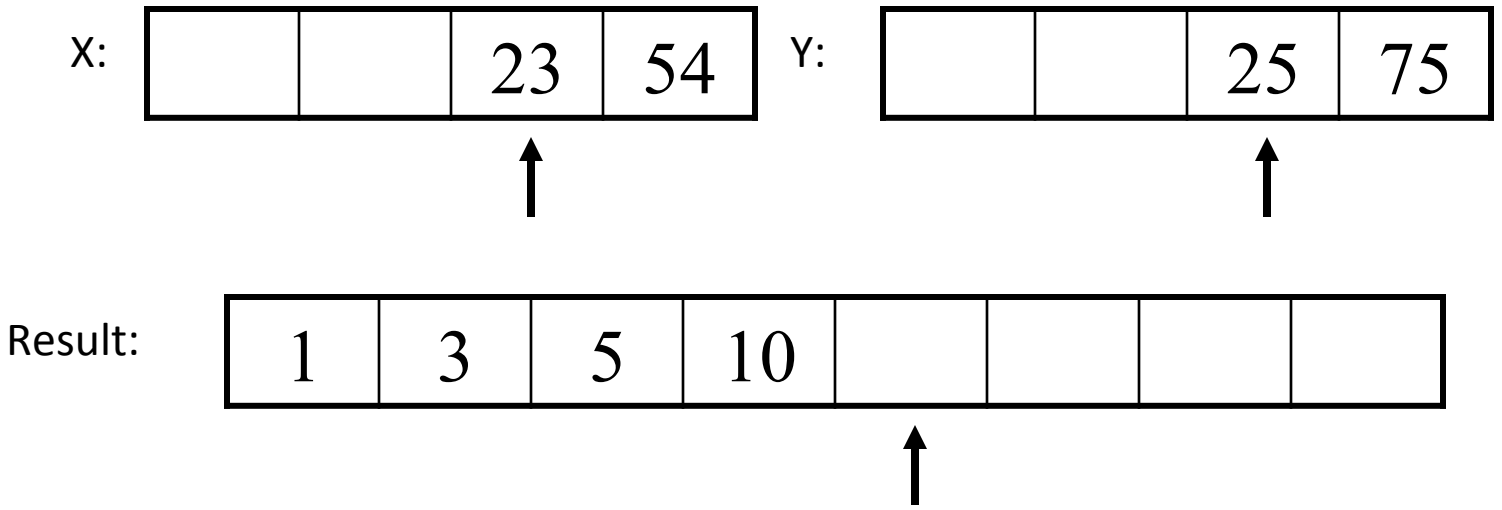
Merging (cont.)



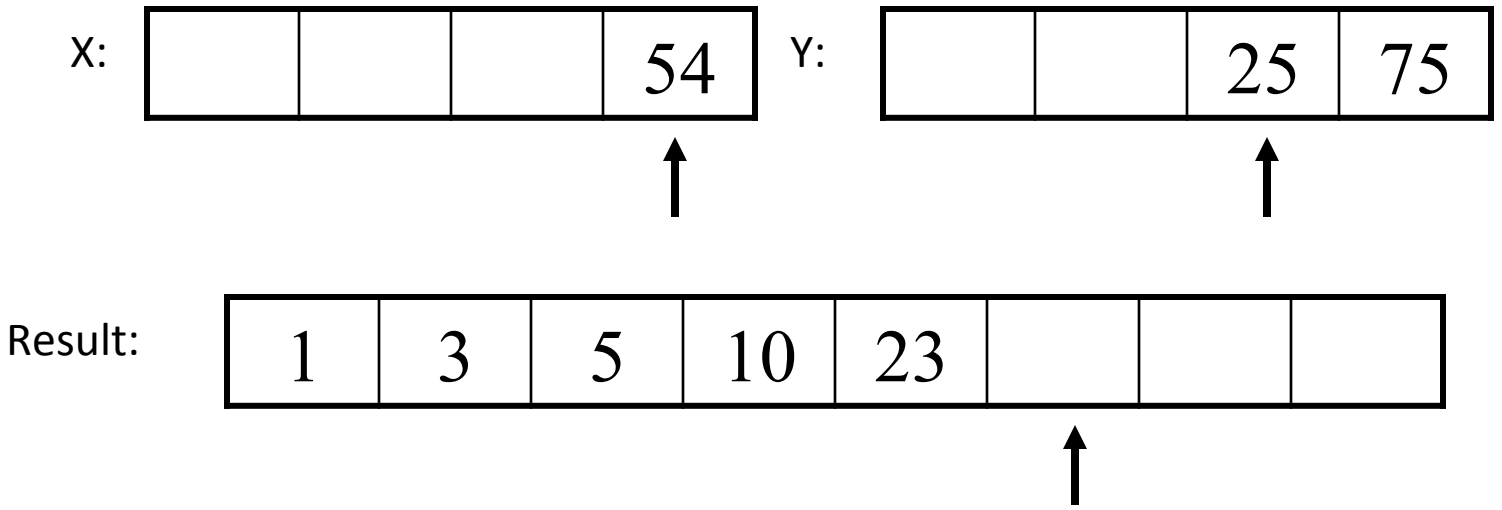
Merging (cont.)



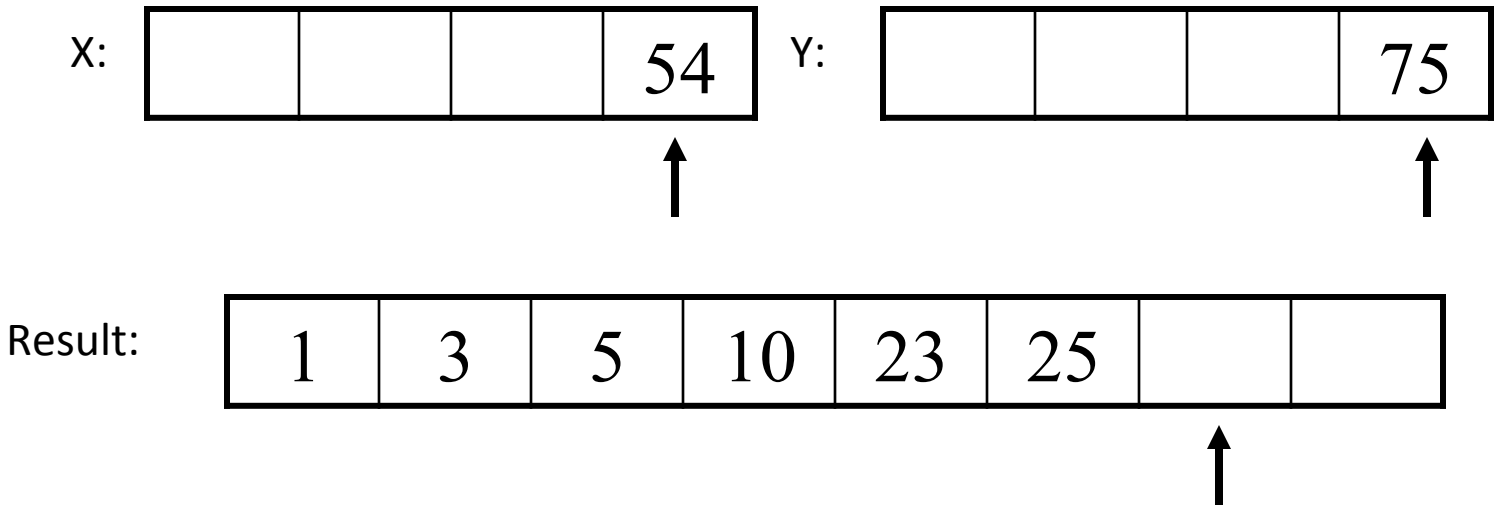
Merging (cont.)



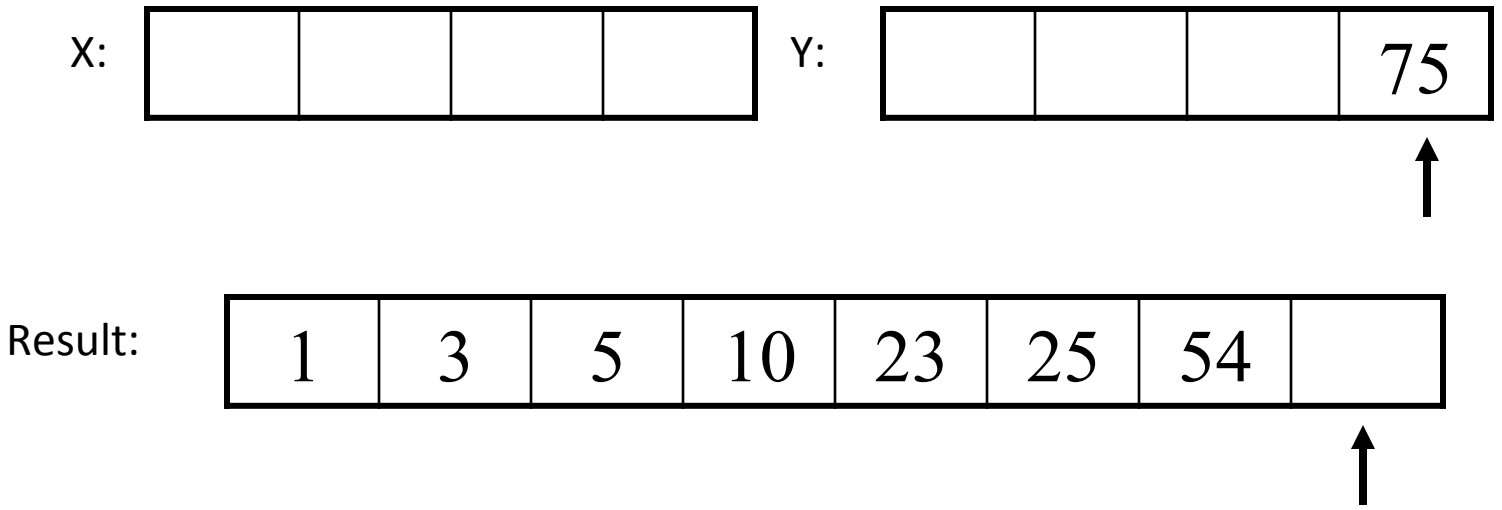
Merging (cont.)



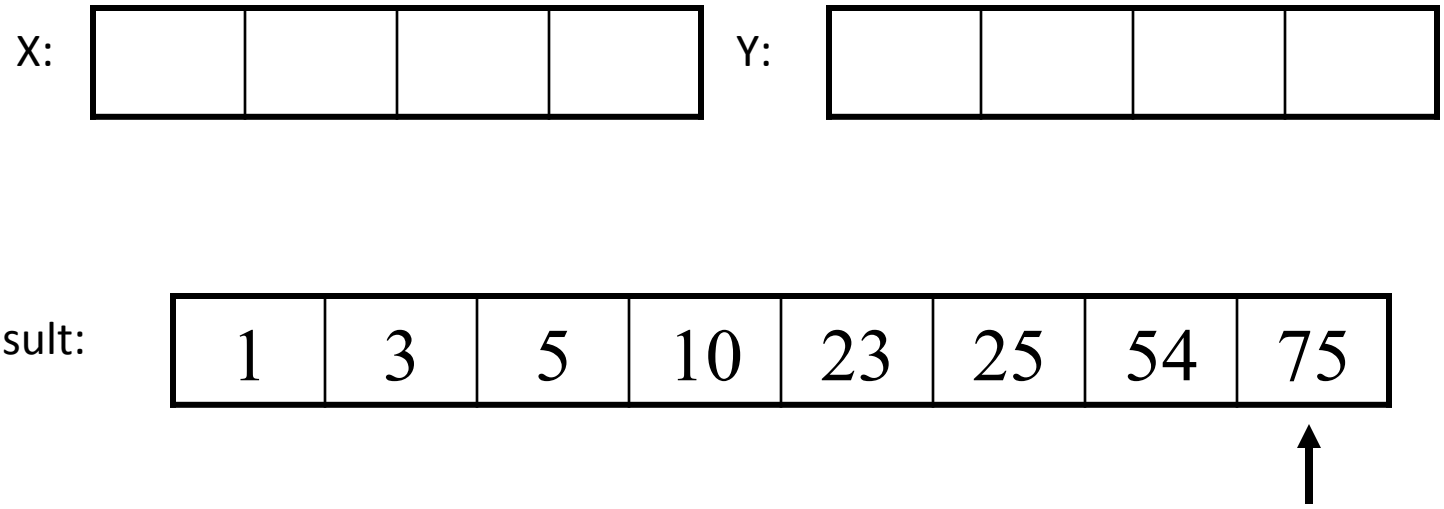
Merging (cont.)



Merging (cont.)



Merging (cont.)

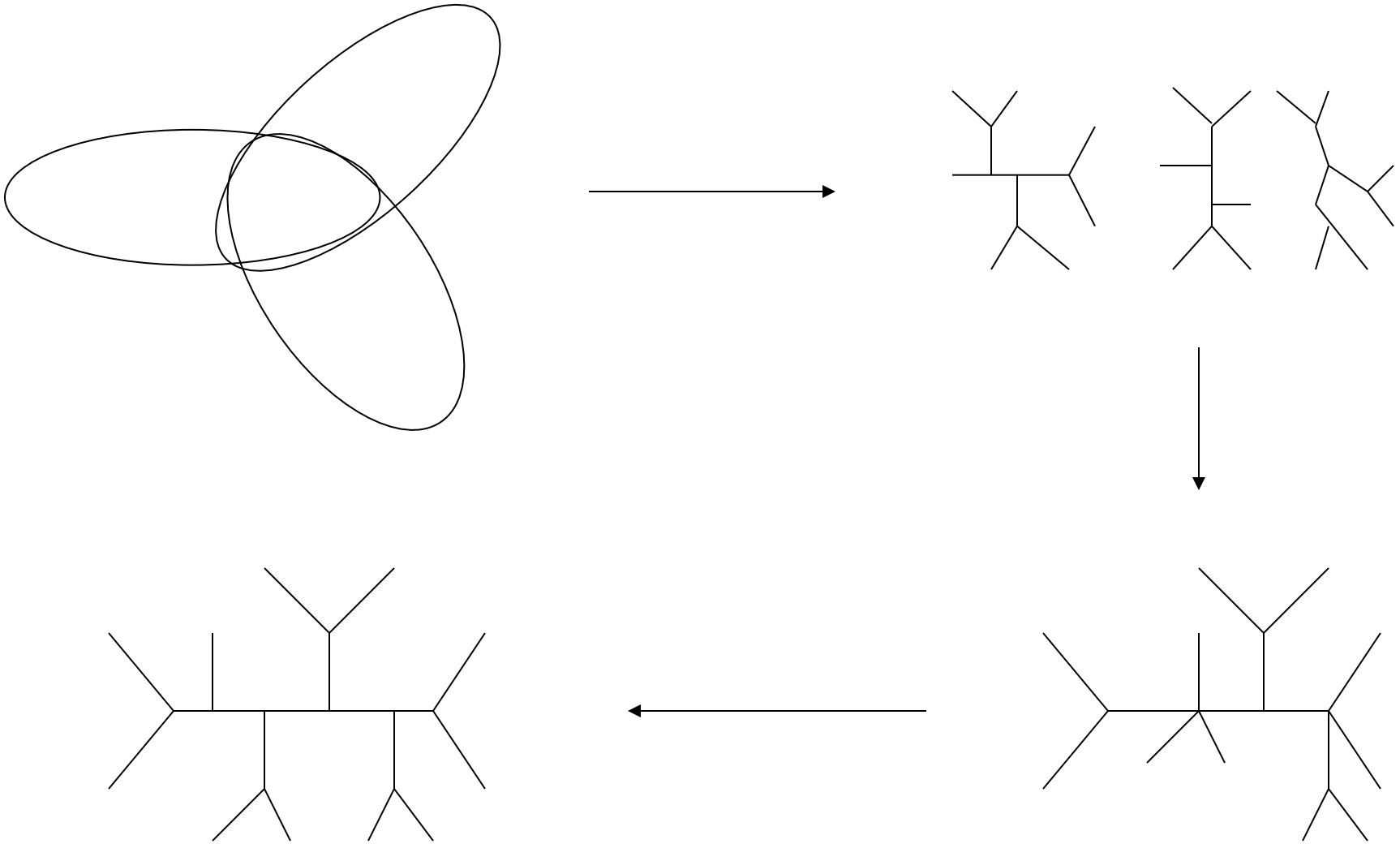


Divide-and-Conquer for Tree Estimation

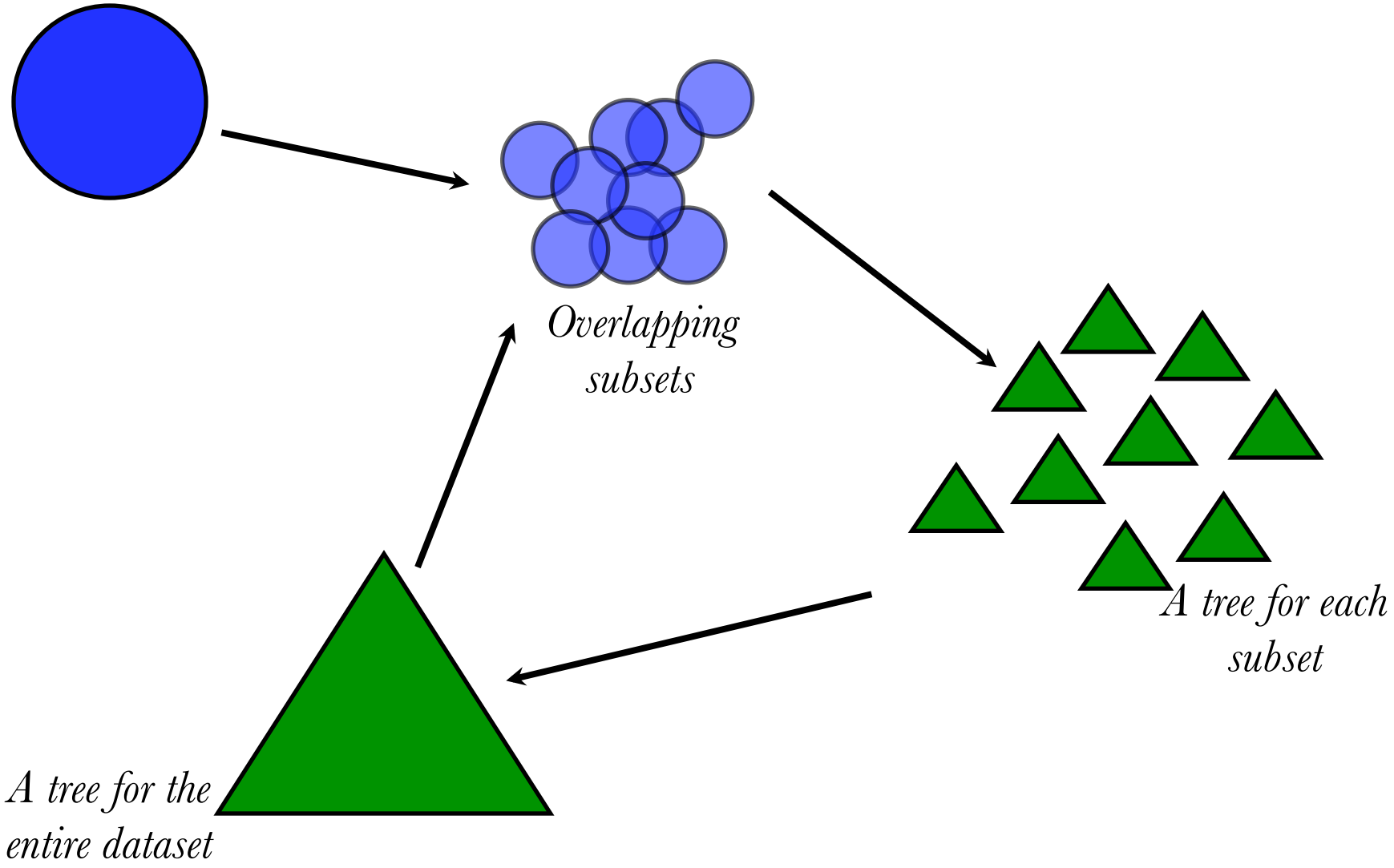
- Basic idea: divide a dataset into two or more overlapping subsets, construct trees on each set, and then combine trees!

- Somewhat easy to figure out how to do this on rooted trees, but a bit harder on unrooted trees.

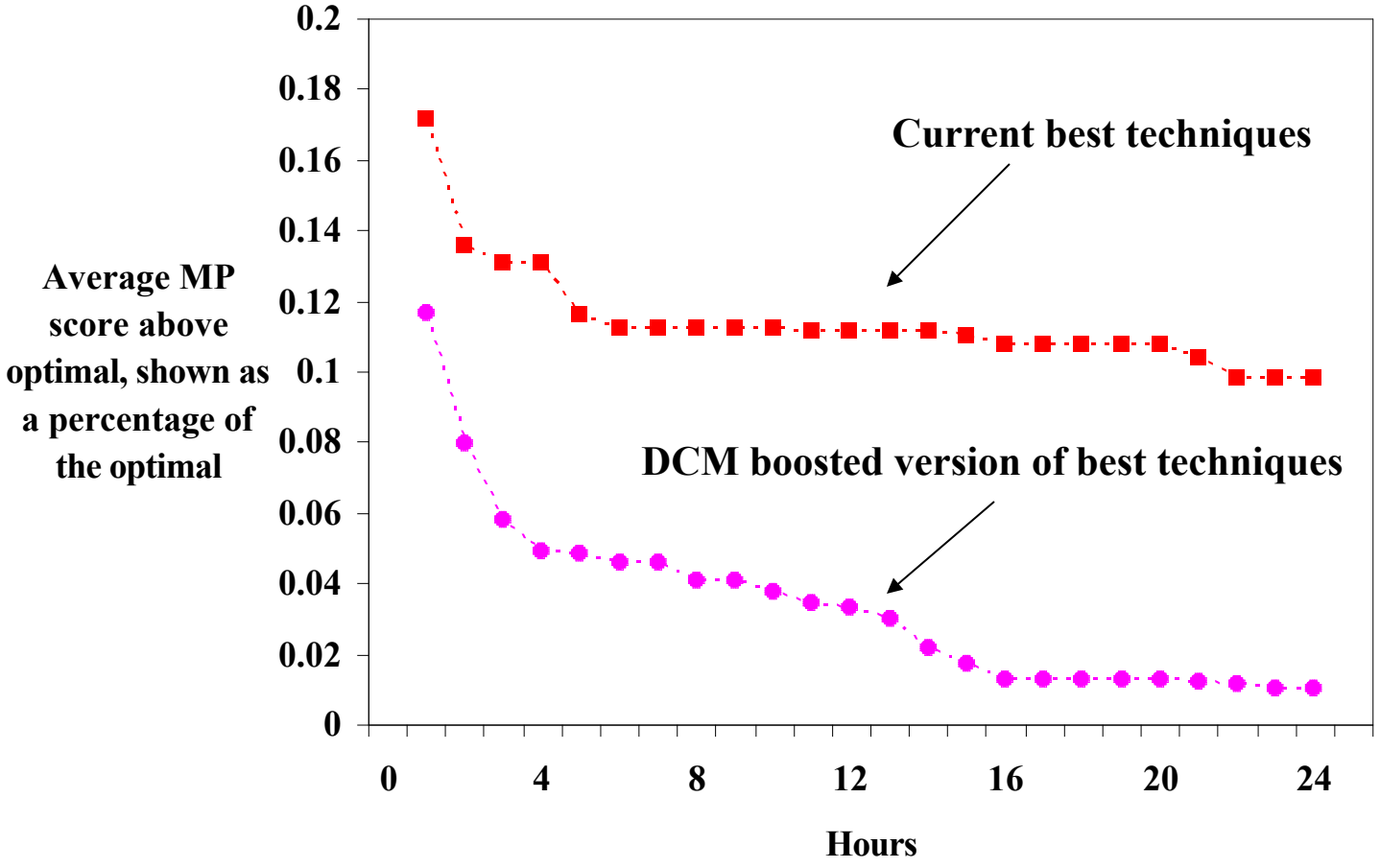
DCMs: Divide-and-conquer for improving phylogeny reconstruction



Iteration plus divide-and-conquer



NP-hard optimization problems: better accuracy, but slow to find good solutions



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

Computer Science Solving Problems in Biology and Linguistics

- Algorithm design using
  - Divide-and-conquer
  - Iteration
  - Heuristic search
  - Graph theory
- Algorithm analysis using
  - Probability Theory
  - Graph Theory
- Simulations and modelling
- Collaborations with biologists and linguists and data analysis
- Discoveries about how life evolved on earth (and how languages evolved, too)

Main Points

- Computer scientists develop algorithms and software to make it possible for scientists to get improved accuracy in their analyses.
- These algorithms involve creative strategies, including divide-and-conquer, iteration, and randomization.
- Extensive simulations and data analyses are part of the evaluation process!

Computational Phylogenetics

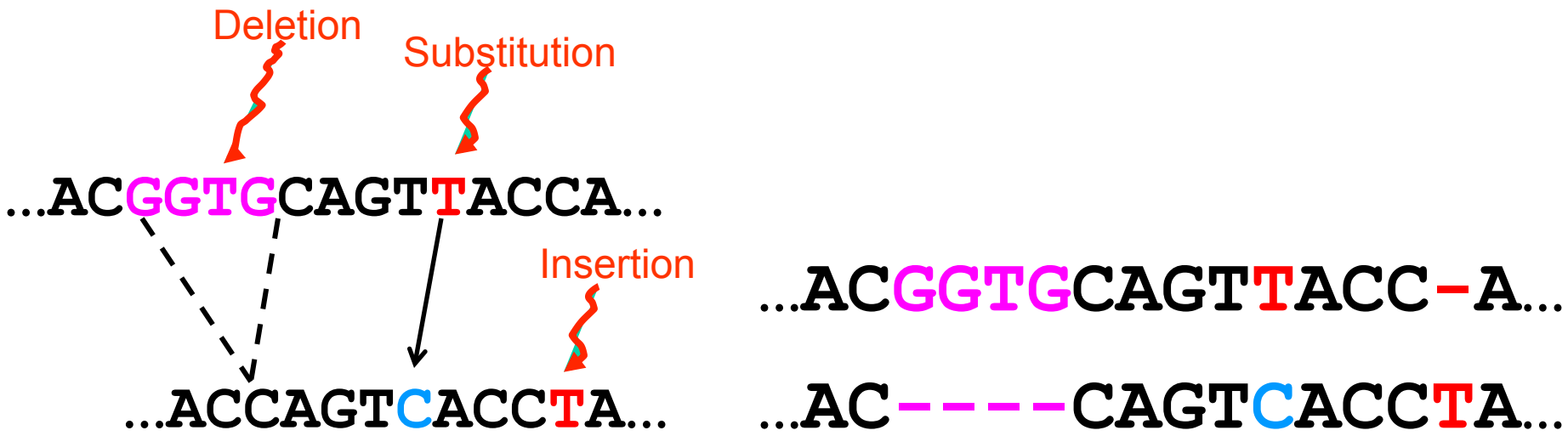
Courtesy of the Tree of Life project

Current methods can use months to estimate trees on 1000 DNA sequences

Our objective:

More accurate trees and alignments on 500,000 sequences in under a week

We prove theorems using graph theory and probability theory, and our algorithms are studied on real and simulated data.



The true multiple alignment

- **Reflects historical substitution, insertion, and deletion events**
- **Defined using transitive closure of pairwise alignments computed on edges of the true tree**

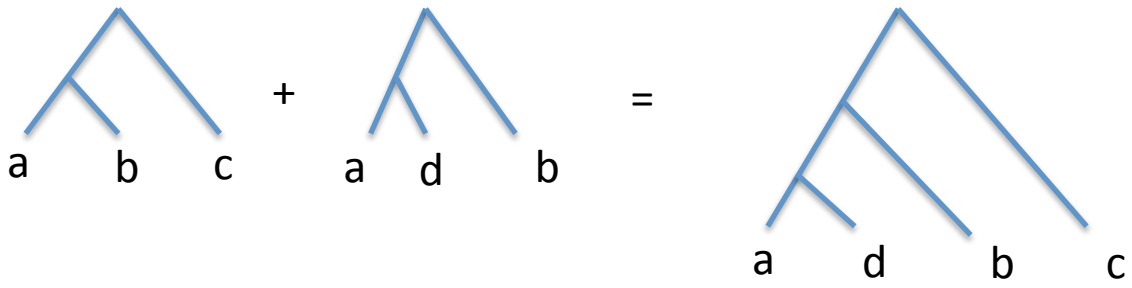
(Some of) our Methods

- SATé, PASTA, and UPP: large, very large, and ultra-large multiple sequence alignment
- DACTAL and DCM: ultra-large tree estimation

- Techniques:
  - Divide-and-conquer
  - Iteration
  - Hidden Markov Models
  - Graph Theory

Constructing trees from smaller trees

- Rooted trees: If they agree, it's easy! (Not so easy if they disagree)
- Unrooted trees: NP-hard, even if they agree!



Four Boosters

- DCM1: improves accuracy of distance-based methods using divide-and-conquer and chordal graph theory
- DACTAL: uses divide-and-conquer plus iteration to get a very large tree without needing a multiple sequence alignment

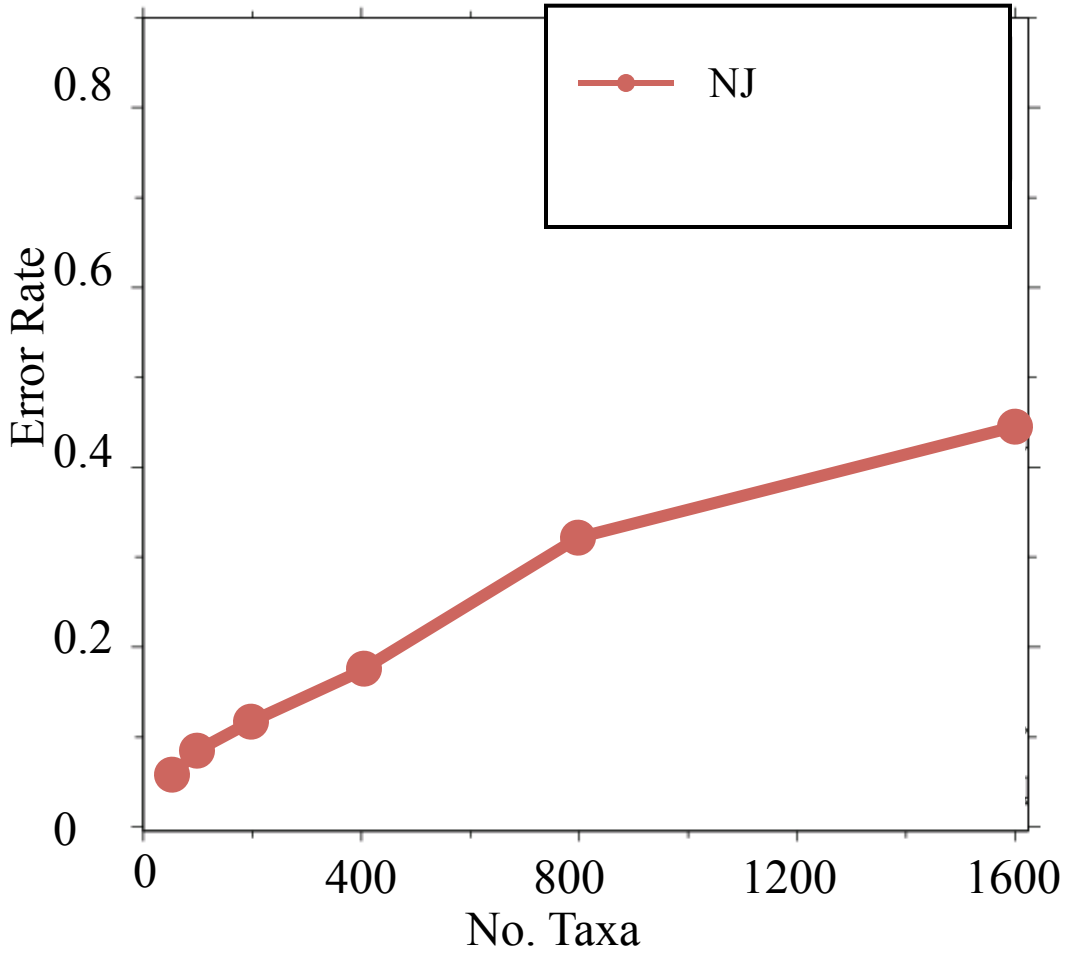
- SATé: uses divide-and-conquer plus iteration to co-estimate the multiple sequence alignment and a tree
- UPP: uses divide-and-conquer, randomization, and Hidden Markov Models to obtain ultra-large alignments

First example: DCM1

DCM = “Disk Covering Method”

- Theory: Warnow, St. John, and Moret, SODA 2001,
- Practice:Nakhleh et al., ISMB 2001

Performance on large diameter trees



Simulation study based upon fixed edge lengths, K2P model of evolution, sequence lengths fixed to 1000 nucleotides.

Error rates reflect proportion of incorrect edges in inferred trees.

[Nakhleh et al. ISMB 2001]

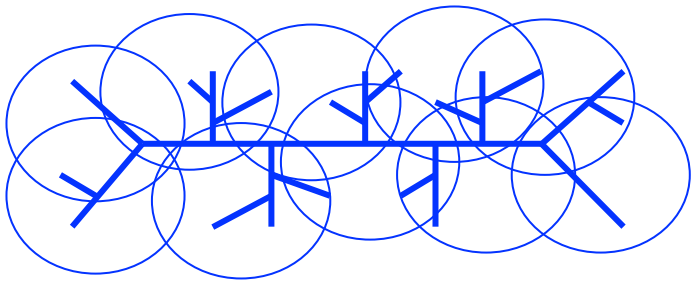
DCM1 Decompositions

Input: Distance matrix (distances between species) and threshold q.

Output: Division of the set of species into *overlapping subsets*.

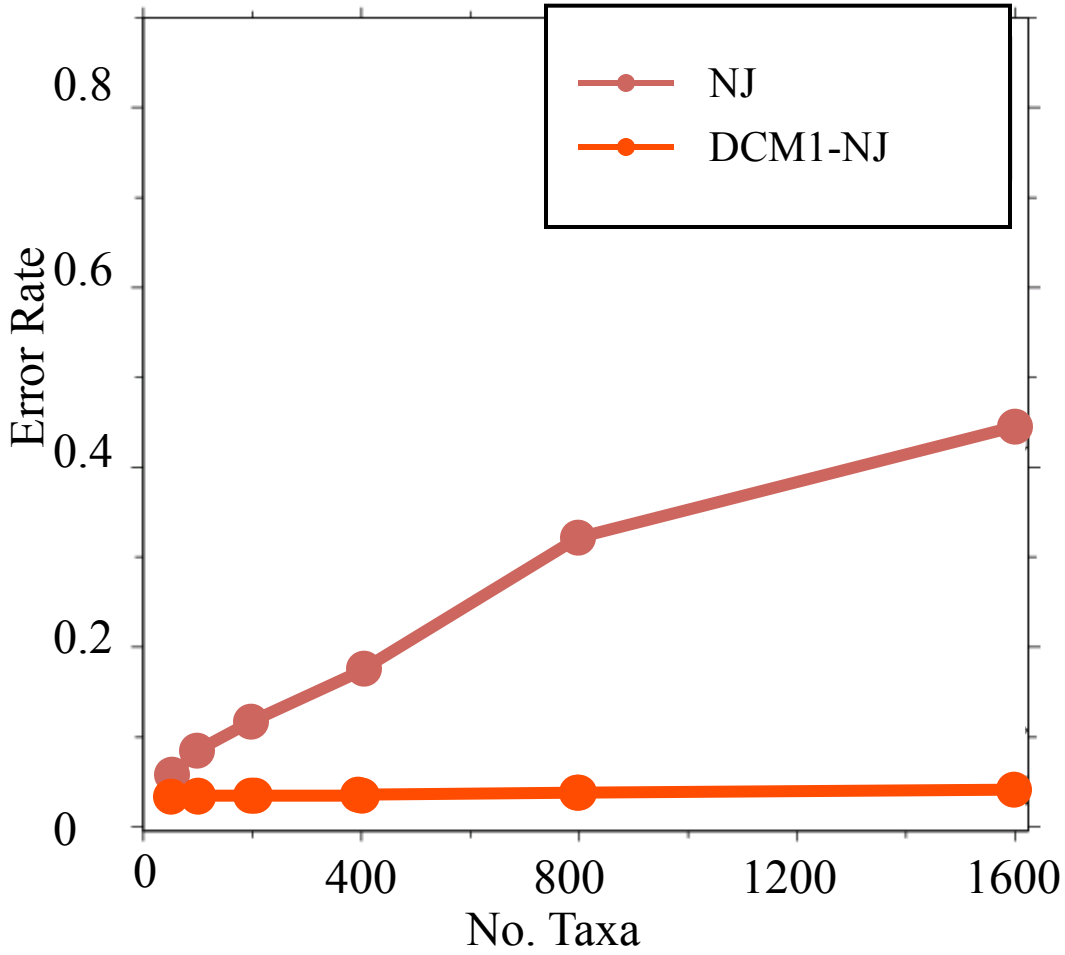
Technique: Compute maximal cliques in a “Triangulated Threshold Graph”.

Looks like moving a disk across a tree!



DCM1-boosting distance-based methods

[Nakhleh et al. ISMB 2001]



Theorem (Warnow et al., SODA 2001): DCM1-NJ converges to the true tree from polynomial length sequences

Second Example: DACTAL

(Divide-And-Conquer Trees (Almost) without alignments)

- Input: set S of unaligned sequences
- Output: tree on S (but no alignment)

Nelesen, Liu, Wang, Linder, and Warnow, ISMB and Bioinformatics 2012

Divide-and-Conquer for Alignment

- SATé: uses divide-and-conquer plus iteration to co-estimate the multiple sequence alignment and a tree
- UPP: uses divide-and-conquer, randomization, and Hidden Markov Models to obtain ultra-large alignments

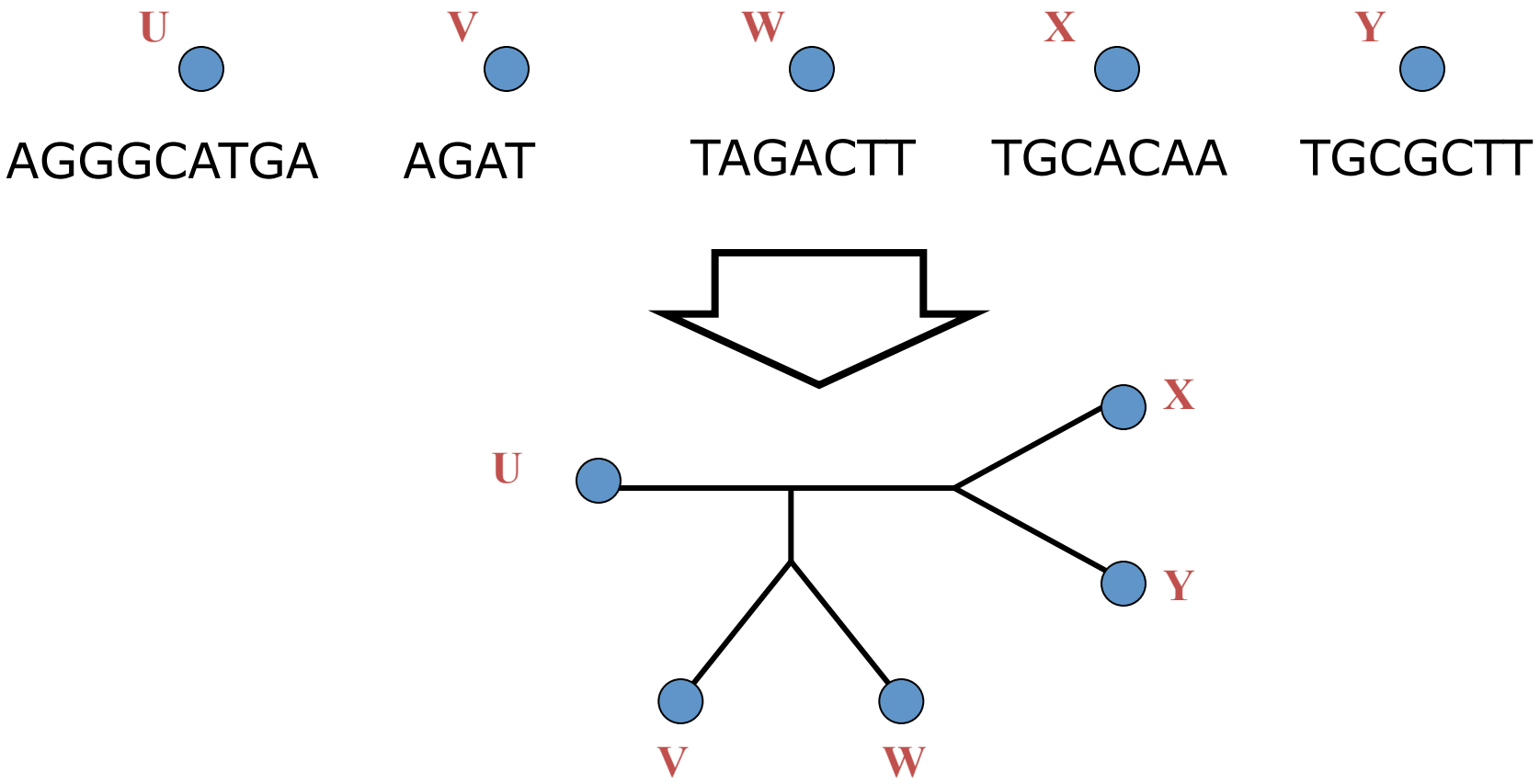
Example 3: SATé

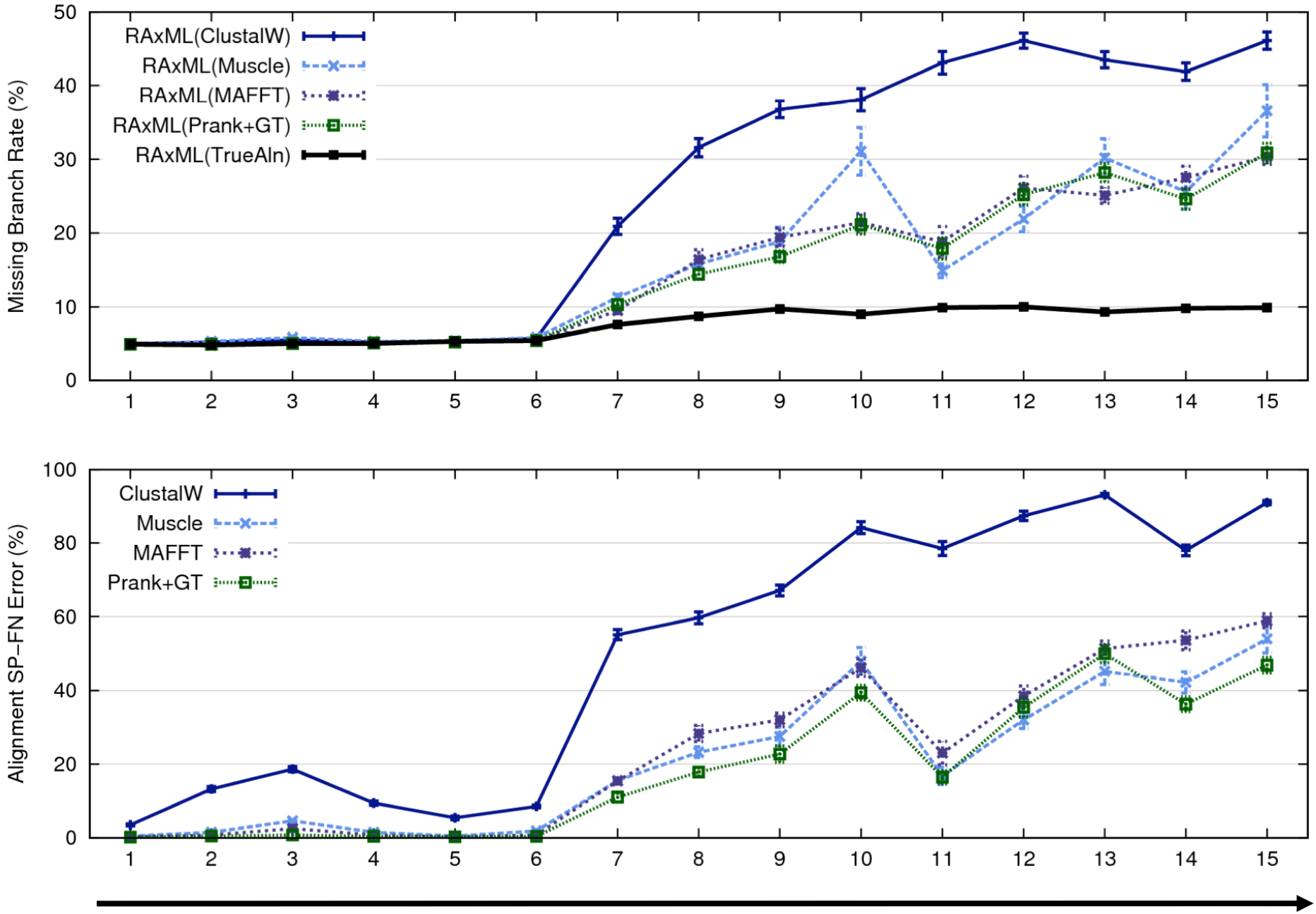
SATé: Simultaneous Alignment and Tree Estimation

Liu, Nelesen, Raghavan, Linder, and Warnow, *Science*, 19 June 2009, pp. 1561-1564.

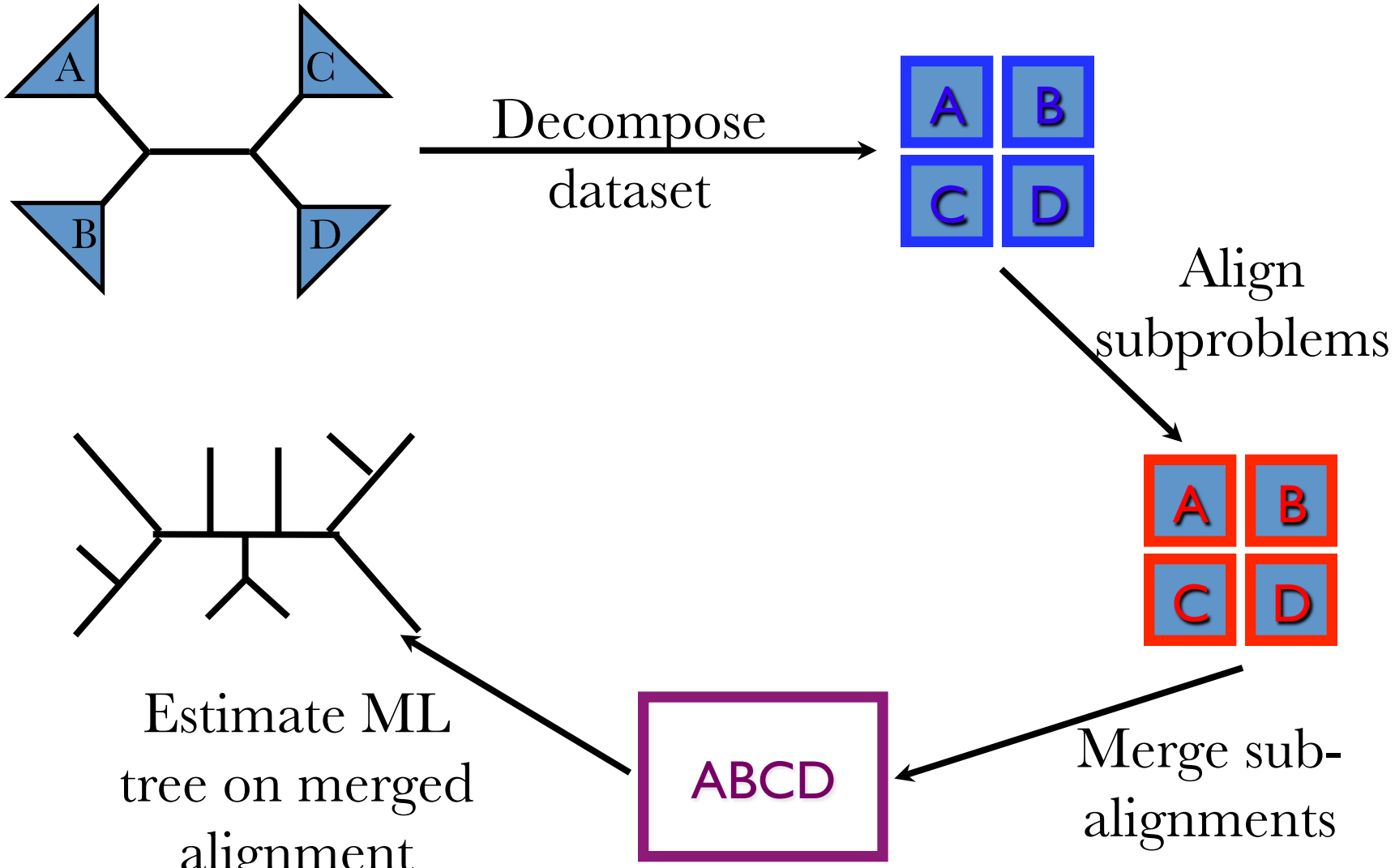
Liu et al., Systematic Biology 2012

Public software distribution (open source) through Mark Holder's group at the University of Kansas


U
V
W
X
Y
AGGGCATGA
AGAT
TAGACTT
TGCACAA
TGCGCTT
X
U
Y
V
W

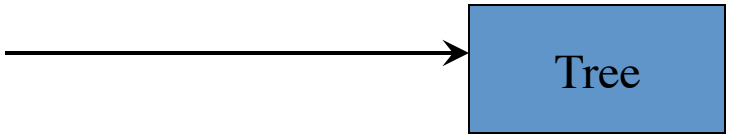

Missing Branch Rate (%)
RAxML(ClustalW)
RAxML(Muscle)
RAxML(MAFFT)
RAxML(Prank+GT)
RAxML(TrueAln)
Alignment SP–FN Error (%)
ClustalW
Muscle
MAFFT
Prank+GT
1000 taxon models, ordered by difficulty (Liu et al., 2009)

Re-aligning on a tree



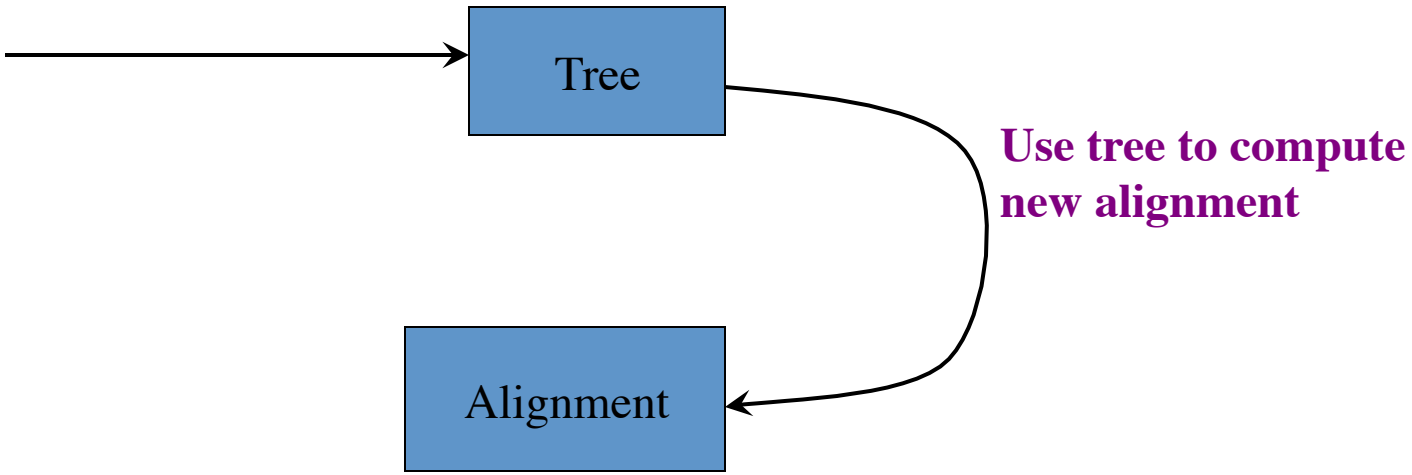
SATé Algorithm

Obtain initial alignment and
estimated ML tree



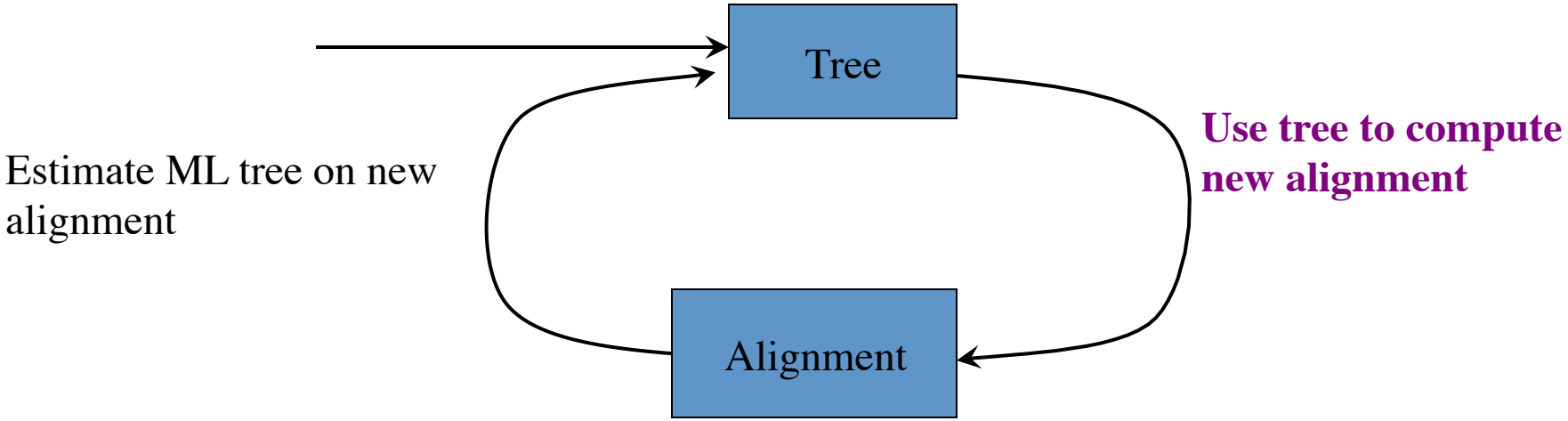
SATé Algorithm

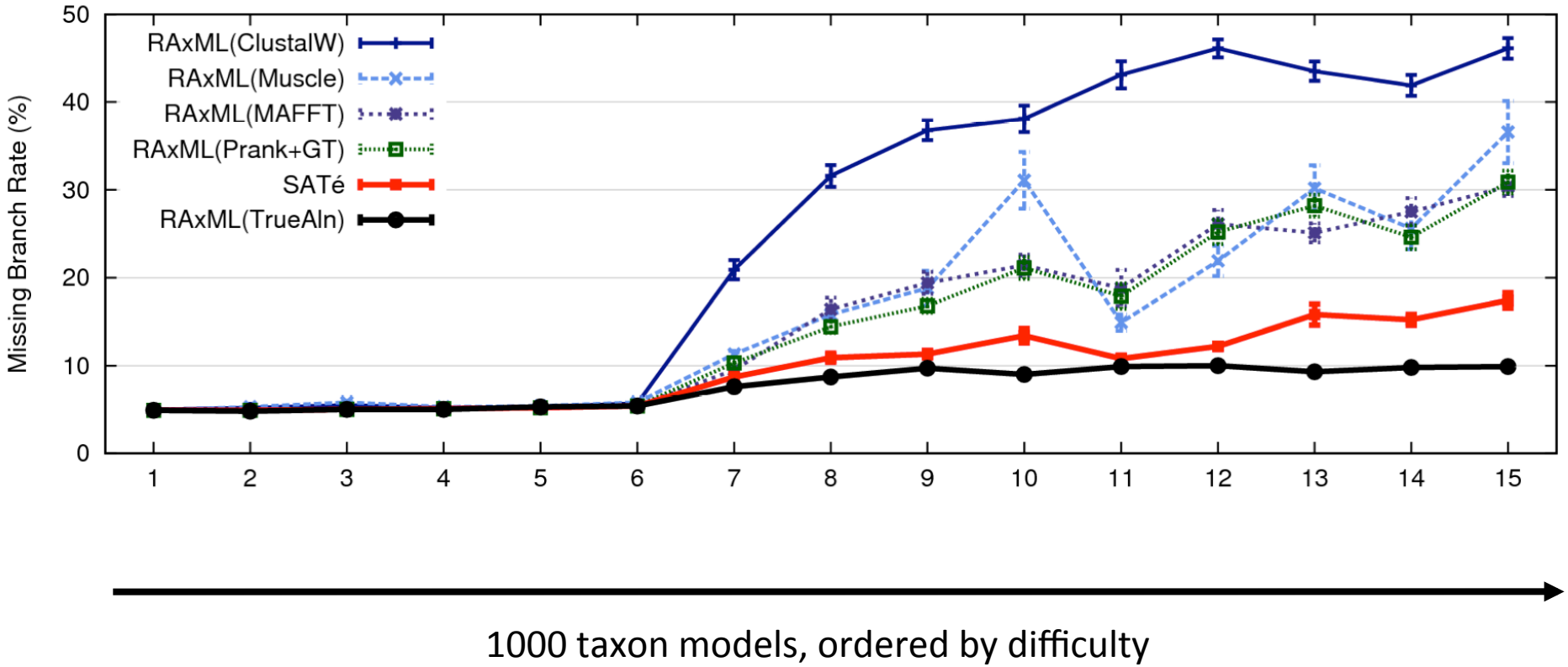
Obtain initial alignment and
estimated ML tree



SATé Algorithm

Obtain initial alignment and estimated ML tree

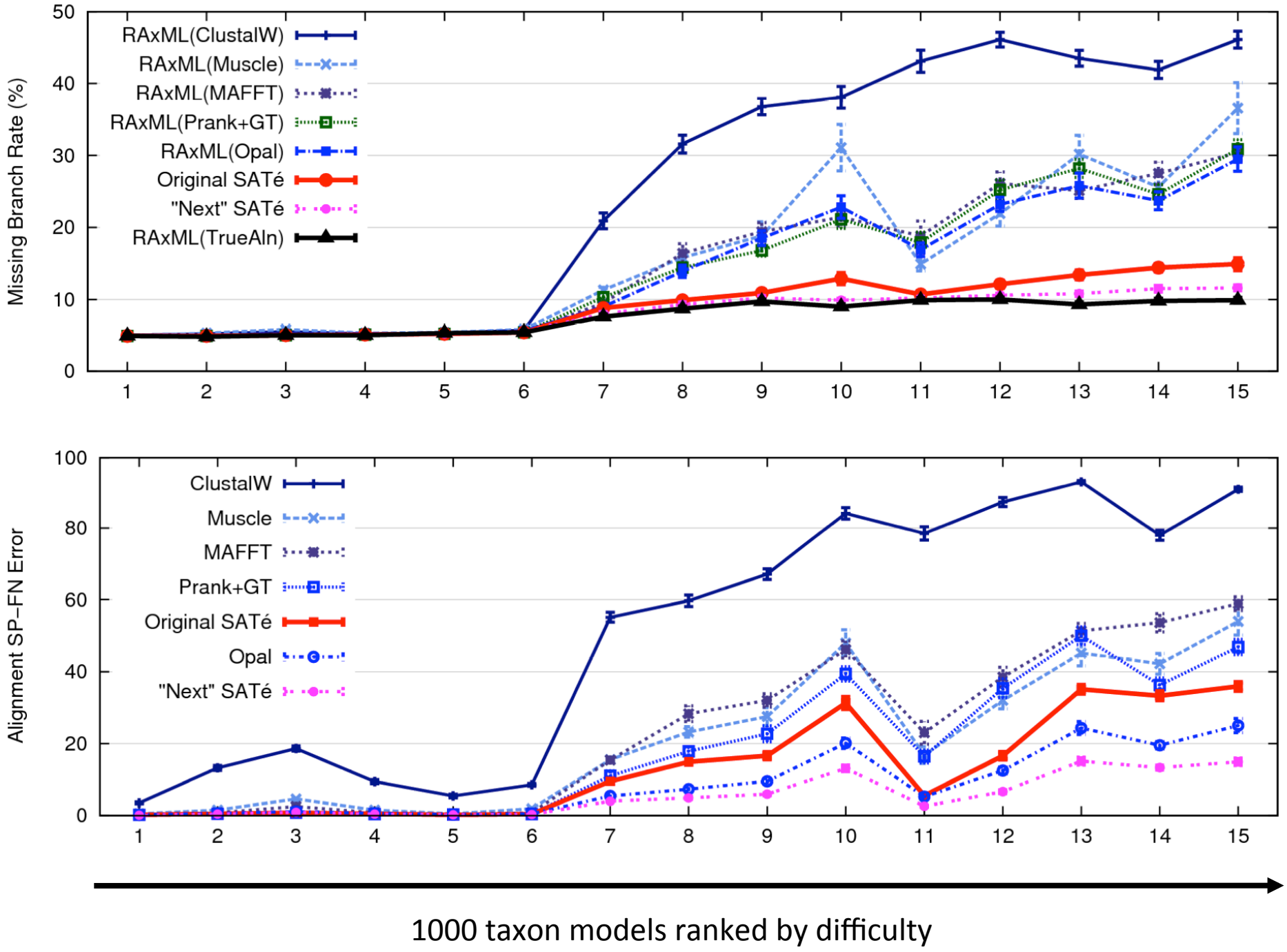




1000 taxon models, ordered by difficulty

24 hour SATé analysis, on desktop machines

(Similar improvements for biological datasets)


Missing Branch Rate (%)
RAxML(ClustalW)
RAxML(Muscle)
RAxML(MAFFT)
RAxML(Prank+GT)
RAxML(Opal)
Original SATé
"Next" SATé
RAxML(TrueAln)
Alignment SP–FN Error
ClustalW
Muscle
MAFFT
Prank+GT
Original SATé
Opal
"Next" SATé
1000 taxon models ranked by difficulty

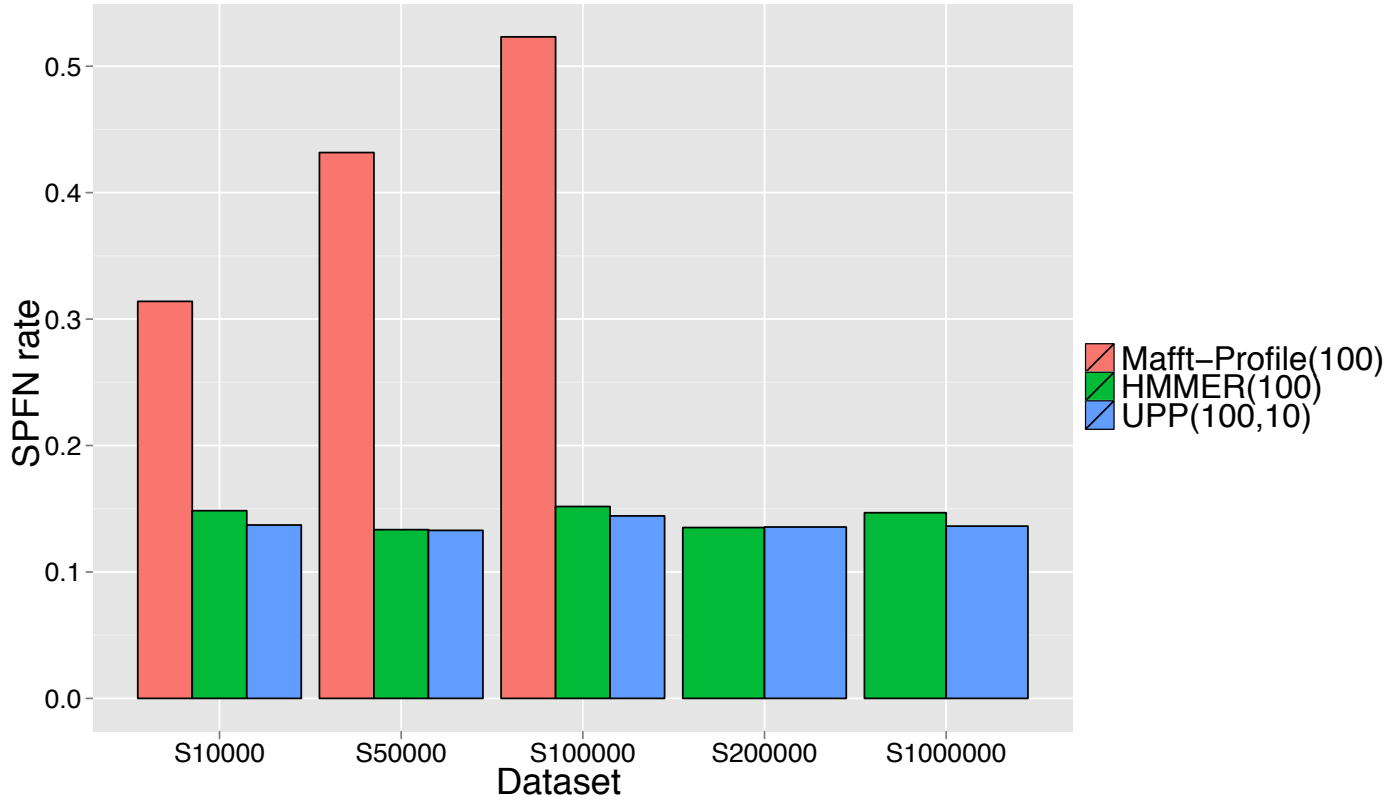
Example 4: UPP

- UPP: Ultra-large alignments using SEPP
- Authors: Nam Nguyen, Siavash Mirarab, and Tandy Warnow

Basic idea:

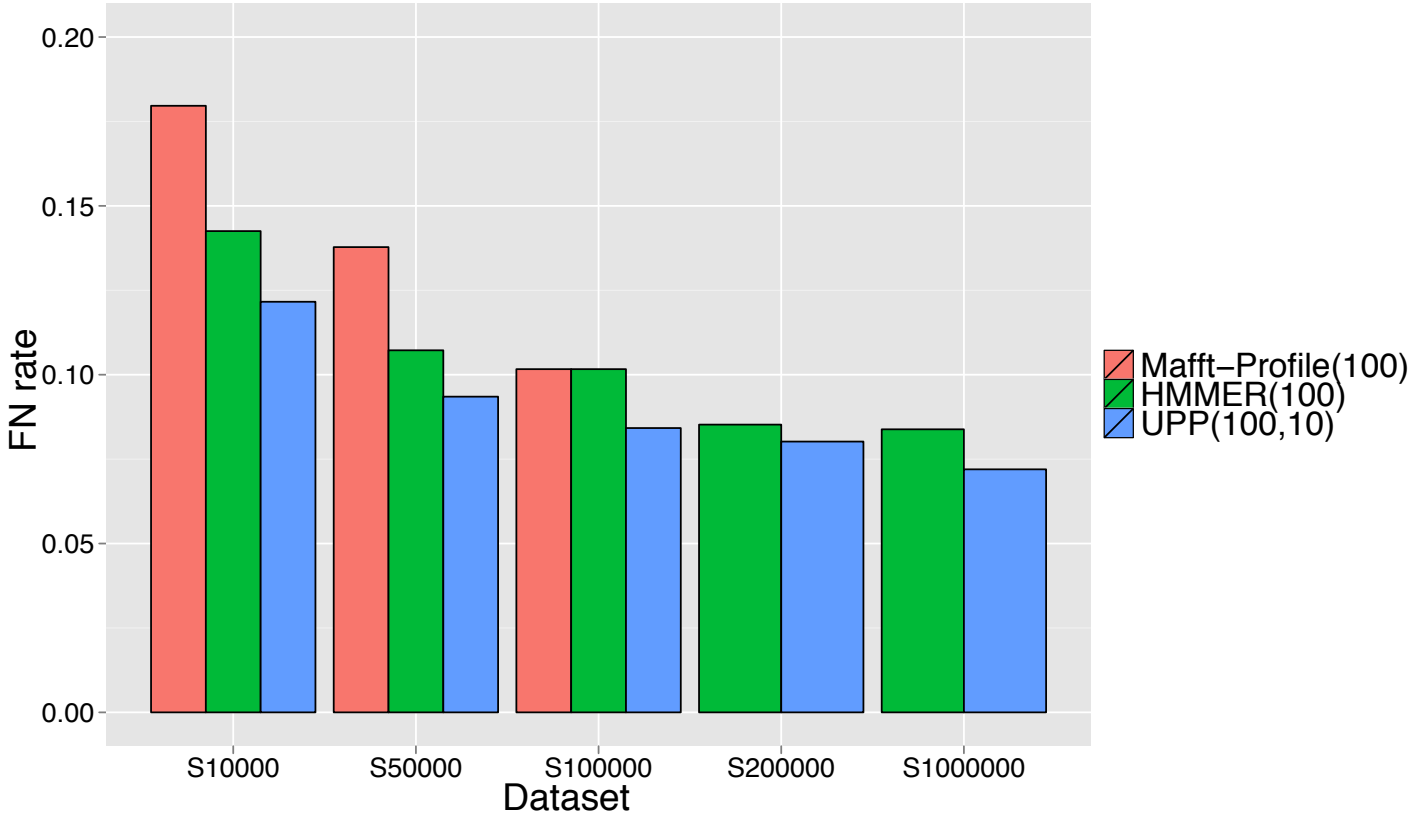
- estimate an alignment on a small random subset of the sequence dataset
- add all the remaining sequences into the small alignment (one by one, independently), using multiple Hidden Markov Models

UPP vs. HMMER vs. MAFFT (alignment error)



MAFFT-profile alignment strategy not as accurate as UPP(100,10) or UPP(100,100).

UPP vs. HMMER vs. MAFFT (tree error)



ML on UPP(100,10) and UPP(100,100) alignments both produce produce better trees than MAFFT.

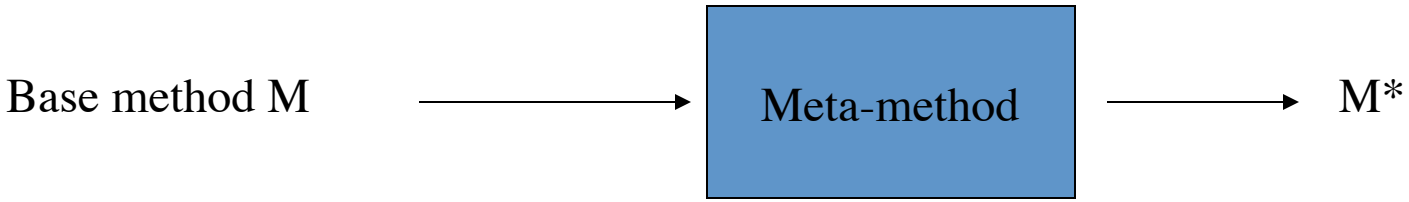
Decomposition into a family of HMMs improves resultant trees.

Four Boosters

- DCM1: improves accuracy of distance-based methods using divide-and-conquer and chordal graph theory
- DACTAL: uses divide-and-conquer plus iteration to get a very large tree without needing a multiple sequence alignment
- SATé: uses divide-and-conquer plus iteration to co-estimate the multiple sequence alignment and a tree
- UPP: uses divide-and-conquer, randomization, and Hidden Markov Models to obtain ultra-large alignments

“Boosters”, or “Meta-Methods”

- Meta-methods use divide-and-conquer and iteration (or other techniques) to “boost” the performance of base methods (phylogeny reconstruction, alignment estimation, etc)



Summary

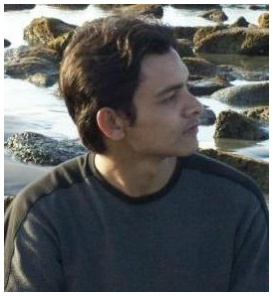
- Standard alignment and phylogeny estimation methods do not provide adequate accuracy on large datasets.
- When markers tend to yield poor alignments and trees, don't throw out the data – *get a better method!*

Summary

- Computer scientists develop algorithms and software to make it possible for scientists to get improved accuracy in their analyses.
- These algorithms involve creative strategies, including divide-and-conquer, iteration, and randomization.
- Extensive simulations and data analyses are part of the evaluation process!

Warnow Laboratory





PhD students: Siavash Mirarab[1], Nam Nguyen, and Md. S. Bayzid[2]

Undergrad: Keerthana Kumar

Lab Website: http://www.cs.utexas.edu/users/phylo

Funding: Guggenheim Foundation, Packard Foundation, NSF, Microsoft Research New England, David Bruton Jr. Centennial Professorship, and TACC (Texas Advanced Computing Center)

[1]HHMI International Predoctoral Fellow, [2]Fulbright Predoctoral Fellow

DCM1 Decompositions

Input: Set S of sequences, distance matrix d, threshold value $q \in \{d_{ij}\}$

1. Compute threshold graph

$$G_q = (V, E), V = S, E = \{(i, j) : d(i, j) \leq q\}$$

2. Perform minimum weight triangulation (note: if d is an additive matrix, then the threshold graph is provably triangulated).

DCM1 decomposition : Compute maximal cliques