Basics of Multiple Sequence Alignment

Tandy Warnow

February 18, 2020

Basics of Multiple Sequence Alignment

Tandy Warnow

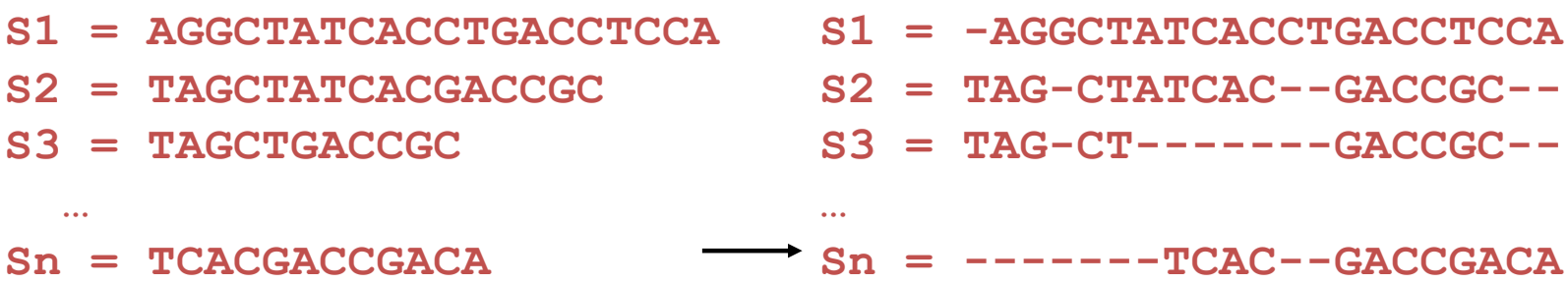
Basic issues

- Evolutionary processes operating on sequences
- What is homology?
- What is a correct pairwise alignment?
- What is a correct multiple sequence alignment?
- How to evaluate alignments
- Fundamental limitations of nearly all multiple sequence alignment methods
- Optimization problems
- Basic techniques of standard methods
- Performance studies of multiple sequence alignment methods

A multiple sequence alignment

s_1	-	-	-	T	A	C
s_2	-	-	A	T	A	C
s_3	C	-	A	-	-	G
s_4	C	-	A	A	T	G
s_5	C	-	-	T	-	G
s_6	C	T	-	-	A	C
s_7	C	-	A	T	A	C
s_8	G	-	A	-	A	T

Multiple Sequence Alignment (MSA): *a scientific grand challenge*[1]



Novel techniques needed for scalability and accuracy

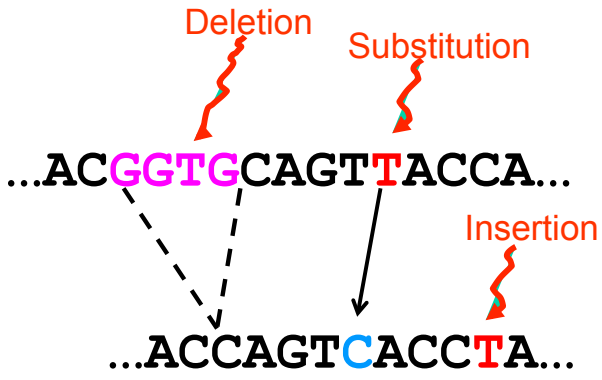
NP-hard problems and large datasets
Current methods do not provide good accuracy
Few methods can analyze even moderately large datasets

Many important applications besides phylogenetic estimation

[1] Frontiers in Massive Data Analysis, National Academies Press, 2013

Homology

Two letters in two sequences are *homologous* if they descend from a letter in a common ancestor.



...ACGGTGCAGTTACC-A...

...AC----CAGTCACCTA...

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**

Evolutionary processes operating on sequences

- Substitutions
- Insertions and deletions of strings (indels)
- Rearrangements (inversions and transpositions)
- Duplications of regions

Note: most alignment methods stretch out sequences so that the line up well, and so only address substitutions and indels.

True Pairwise Alignment

- Suppose X and Y are two sequences, and X evolves into sequence Y via insertions, deletions, and substitutions.
- The true pairwise alignment of X and Y represents this true history.
- Examples:
  - AAT evolves into ACCAT by the insertion of CC
  - ATGA evolves into ATTAG by changing G to T, and then adding G
  - CTAA evolves into CTTAA by inserting a T.

Questions:

1. What are the pairwise alignments?
2. How can we guess at these evolutionary histories (and so pairwise alignment)?

How to evaluate methods

Given an estimated and true (or reference alignment), we can compute various statistics, many of which are based on "homology pairs":

- SPFN: sum of the false negative homology pairs
- SPFP: sum of the false positive homology pairs
- TC: total column score
- Compression: ratio of the estimated alignment length to true alignment length
- Distance between gap length distributions

Issues to consider

- Most methods can only handle indels and substitutions (i.e., no rearrangements or duplications).
- Most methods assume full-length sequences.
- Statistical methods are all based on models of sequence evolution, and the models are limited.
- Most methods cannot analyze very large datasets.
- Evaluation of methods is tricky.

Edit distance and pairwise alignment

Suppose each event (insertion, deletion, and substitution) costs 1. Can we compute the minimum cost edit transformation between two sequences?

DP algorithm for the edit distance

Input: sequences a and b of lengths m and n, respectively.
Output: minimum number of indels and substitutions needed to transform a into b.
A two-dimensional matrix, F[0..m,0..n] is used to hold the edit distance values:

$F(i,j) = d(a[1..i], b[1..j])$ (Definition of what we want)

$F(0,0) = 0$

$F(i,0) = i, i = 1..m$

$F(0,j) = j, j = 1..n$

For $i,j \geq 1$, $F[i,j] = min\{$

$F[i-1,j-1]+$ if a[i]=b[j] then 0 else 1,
$F[i-1,j]+1$,
$F[i,j-1]+1$
$\}$

Needleman-Wunsch minimum edit distance

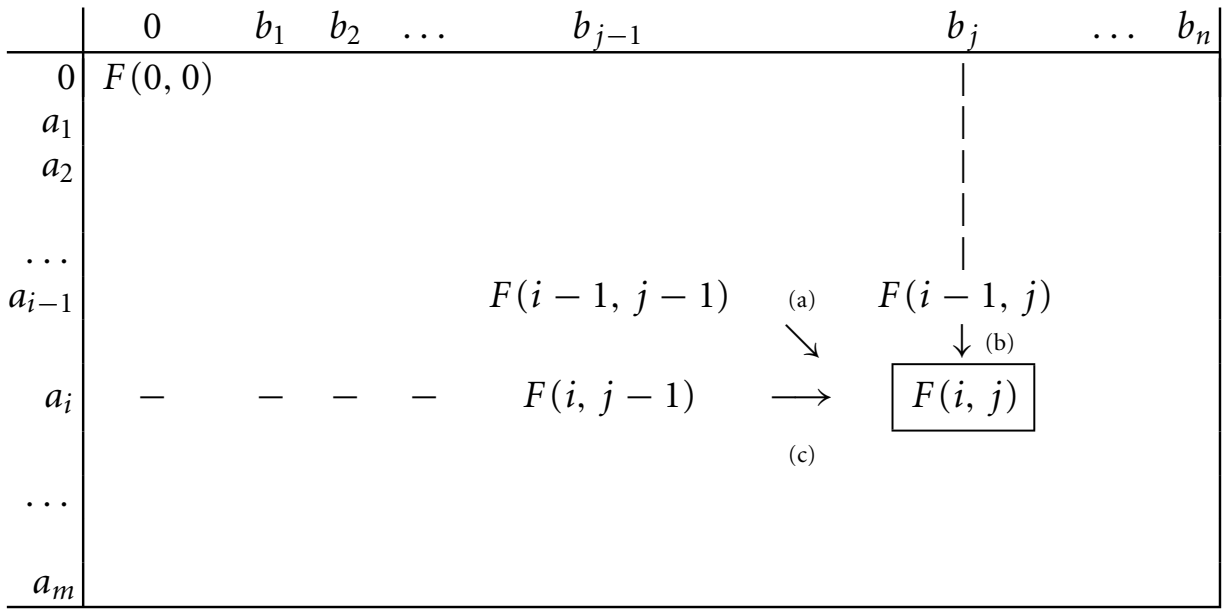


Figure 9.3 (Figure 2.4 in Huson et al. (2010)) The Needleman-Wunch dynamic programming ap-

DP algorithm for minimum edit distance

Compute the matrix from the bottom up!
Running time is $O(mn)$ where the first sequence has length m and the second sequence has length n.
Example: $ACAT$ and $CCGT$

Finding the actual transformation - backtracing

To find the actual transformation, use backtracing.
Example: *ACAT* and *CCGT*

Extensions

Minimum cost approaches:

- How would you modify this algorithm if indels cost C and substitutions cost C'?
- How would you modify the edit distance algorithm if a indel of length p has cost $C + C'p$?

Maximize similarity approaches:

- How would you modify the algorithm to maximize score, so matches have value 1 and mismatches and indels each have negative value -1 (i.e., they cost)?
- How would you modify the algorithm to maximize score, where matches and mismatches have scores (possibly negative) that depend on the pair of letters, and indels all have negative scores?
- How would you modify the algorithm if you want a *local alignment* (maximize cut off some prefix and suffix)

More generally,

- How would you align two alignments?

Extending to multiple alignment

Now that we have a way of defining the "cost" of a pairwise alignment, how can we extend to a set of three or more sequences?

Optimization criteria

Three criteria, two of which are extensions of edit distances:

- Sum-of-pairs (sum of edit distances on induced pairwise alignments)
- Tree alignment (sum of costs of edges)
- Maximum likelihood under a statistical model of sequence evolution

All three are NP-hard, even if the tree is given.

Treelength

Just like parsimony, but allows indels!

- Fixed tree version: Given unaligned sequences at the leaves of a fixed tree, finding the sequences at internal nodes to minimize the cost.
- Another fixed tree version: given aligned sequences at the leaves of the tree, find the sequences at internal nodes to minimize the cost.
- Generalized Tree Alignment problem (Sankoff): given unaligned sequences, find the tree and sequences at inernal nodes to minimize the cost.

All are NP-hard.
Methods for Treelength include POY and BeeTLe.
Very computationally intensive (worse than MP) ...and also controversial!

Example 9.5 from textbook

The input is $s_1 = AC, s_2 = ATAC, s_3 = CAG$. We seek the sequence X at the internal node of the tree with s_1, s_2, and s_3 at the leaves.

1. Compute the pairwise alignments obtained on each edge for $X = AC$.
2. Compute the multiple sequence alignment (MSA) defined by the pairwise alignments computed in (1).
3. Compute the SOP-score of the MSA computed in (2).
4. Compute the treelength of the MSA computed in (2).

Tree Alignment

The (Fixed) Tree Alignment problem (finding the sequences at internal nodes of a fixed tree to find the minimum cost) is NP-hard.

Generalized Tree Alignment

- **Input:** Set S of sequences and positive constants C and C', where C is the cost of a single letter indel and C' is the cost for a substitution.
- **Output:** Tree T with S at the leaves and internal nodes labelled by sequences so that the treelength is minimized.

The Generalized Tree Alignment problem is also NP-hard.

Methods for Treelength include POY and BeeTLe, but these are heuristics without provable guarantees. Furthermore, they are computationally intensive.

Constraining the input sequences in GTA

- **Input:** Set S of sequences and positive constants C and C', where C is the cost of a single letter indel and C' is the cost for a substitution.
- **Output:** Tree T with S at the leaves and internal nodes labelled by sequences so that the treelength is minimized.

What happens if we constrain the sequences at the internal nodes to be drawn from S?

Can we solve this problem in polynomial time?

Minimum Spanning Tree

Given a graph $G = (V, E)$ with positive weights on the edges of G, a **Minimum Spanning Tree** (MST) is a subgraph of G that contains all the vertices and that has minimum total cost.

Finding the MST is solvable in polynomial time (Kruskal's Algorithm and Prim's Algorithm).

Let the vertices of the graph be all the sequences in S, and the weight of the edge between two vertices be the edit distance (computed by Needleman-Wunsch).

Hence, we can find an optimal solution to the constrained GTA problem in polynomial time.

Tree Alignments

Let T be a tree with leaves labelled by S and let S' be the assignment of sequences to the internal nodes in T. Then (T, S') defines a multiple sequence alignment $\mathcal{A}$. *Tree alignments* are those alignments that can be obtained in this way.

We are given a set S of sequences, and we wish to find an approximate solution to the Generalized Tree Alignment (GTA) problem.

Specifically, we say an algorithm Φ for GTA is a c-approximation algorithm if the GTA score of $\Phi(S)$ is no more than c times the best possible score.

Theorem 9.6 from the textbook: A minimum spanning tree T is a 2-approximation to the GTA problem.

Likelihood-based approaches

Just like GTR maximum likelihood or GTR Bayesian methods, but allows indels!
A model tree will include parameters for the probability of an indel and its length.
Different problems:

- estimate the alignment given the tree,
- estimate the tree given the alignment, or
- co-estimate the alignment and tree.

Statistical alignment estimation

Some alignment methods are based on explicit parametric models of sequence evolution that include insertions and deletions (indels) as well as substitutions. Examples:

- BAli-Phy
- StatAlign
- Prank
- PAGAN

The good: appealing statistical proeprties
The bad: more computationally intensive than standard MSA methods, and they don't work as well as expected.

Profile Hidden Markov Models

Profile Hidden Markov Models (HMMs) are another kind of statistical model in frequent use in multiple sequence alignment. We will discuss them in detail later.

Aligning a set S of sequences

Suppose S is a set of unaligned sequences and we are told they are all homologous (i.e., share a common evolutionary history) with the sequences in a family $\mathcal{F}$.

How shall we compute a multiple sequence alignment for S?

Basic techniques

Multiple sequence alignment methods generally use one or more of the following techniques to align a set S of sequences:

- Align all sequences in S to a single sequence s^* or to a model (e.g., profile HMM)
- **Progressive alignment**: compute a guide tree, and then align sequences from the bottom up
- **Consistency**: infer support for homology between two letters using third sequences
- **Divide-and-conquer** (especially based on a tree)
- **Iteration** between tree estimation and alignment estimation

Aligning a set S of homologous sequences

- Compute an MSA A for the sequences in $\mathcal{F}$.
- Build the profile HMM H for the alignment A.
- Add all the sequences in S to A, independently.
- The alignment produced will contain all the sequences of $\mathcal{F} \cup S$; you can then restrict to just the sequences in S.

Progressive Alignment

- Given sequences S, find rooted tree (somehow)
- Align sequences from the bottom up (note this requires the ability to align two alignments)
- Return the alignment produced at the root

Progressive Alignment

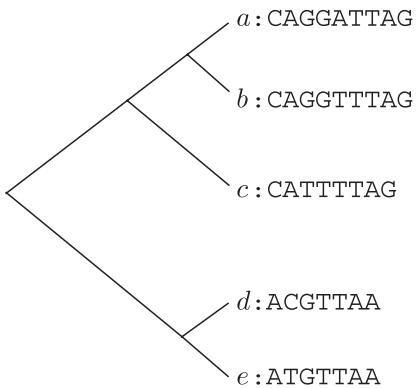
- Build a **guide tree** from the sequences
- Align the sequences from the bottom-up (aligning alignments as you go up)

a: CAGGATTAG
b: CAGGTTTAG
c: CATTTTAG
d: ACGTTAA
e: ATGTTAA

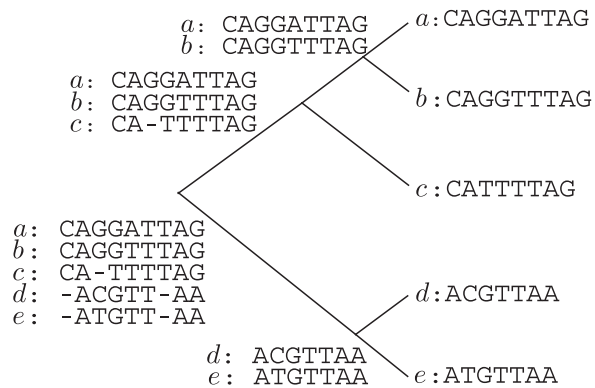
(a) input

	a	b	c	d	e
a	0	1	3	4	4
b	1	0	2	4	4
c	3	2	0	5	5
d	4	4	5	0	1
e	4	4	5	1	0

(b) pairwise distances



(c) Guide tree



(d) Progressive alignment

Aligning alignments

In a progressive alignment, alignments on disjoint sets are aligned together, to make an alignment on the combined set of sequences.

To do this, the two alignments are first represented by profiles, and then these profiles are aligned to each other.

This is performed using dynamic programming, similar to Needleman-Wunsch.

Examples of methods that can align two alignments include Opal and Muscle.

However, another approach is to represent each of the alignments as profile HMMs, and then align the two profile HMMs.

Using libraries of pairwise alignments, part 1

Suppose we have a set S of sequences, and a library L of pairwise alignments for the sequences in S.

For each pair x, y of letters (one from each of two sequences), you have the frequency with which the two letters are aligned in L (i.e., the *support* for the homology pair x, y).

Given a *library* of pairwise alignments, we can define the support for all homology pairs, and then seek the best MSA for these support values.

Using libraries of pairwise alignments, part 2

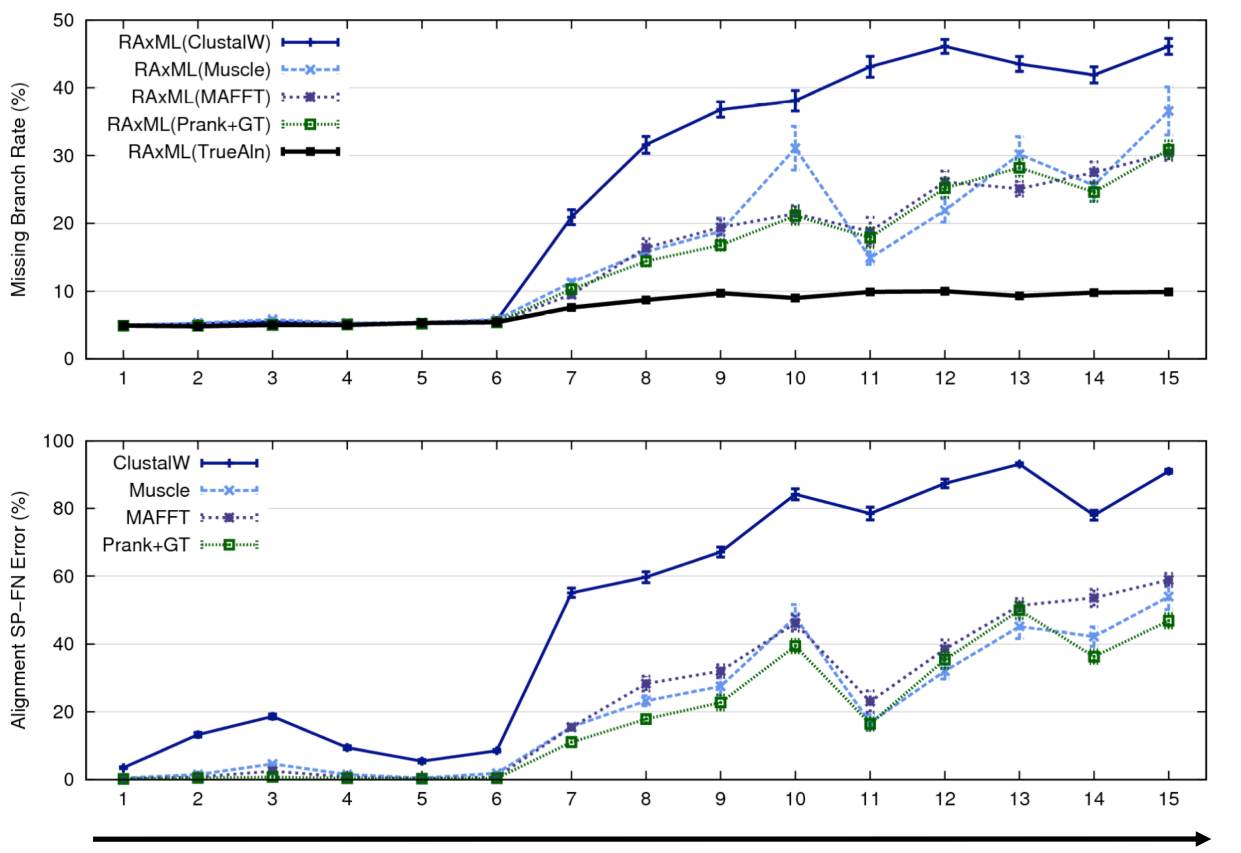
The *consistency technique* is another way of using a library L of pairwise alignments.

Another way of defining the support for the homology pair is the number of letters z (in a third sequence) such that x and z and y and z are aligned in some pairwise alignments in P.

This is how "consistency" is defined – support via a third sequence.

Many of the best MSA methods (e.g., T-Coffee and ProbCons) use the consistency technique in some way, and differ mainly in how they construct the library.

Comparing alignment methods

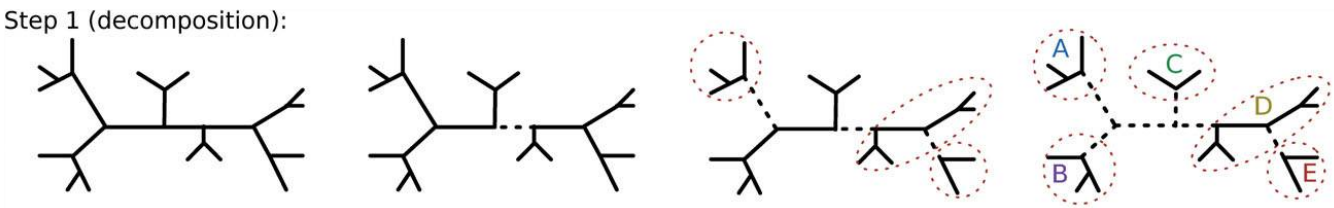


Evolutionary rates increase from left-to-right!

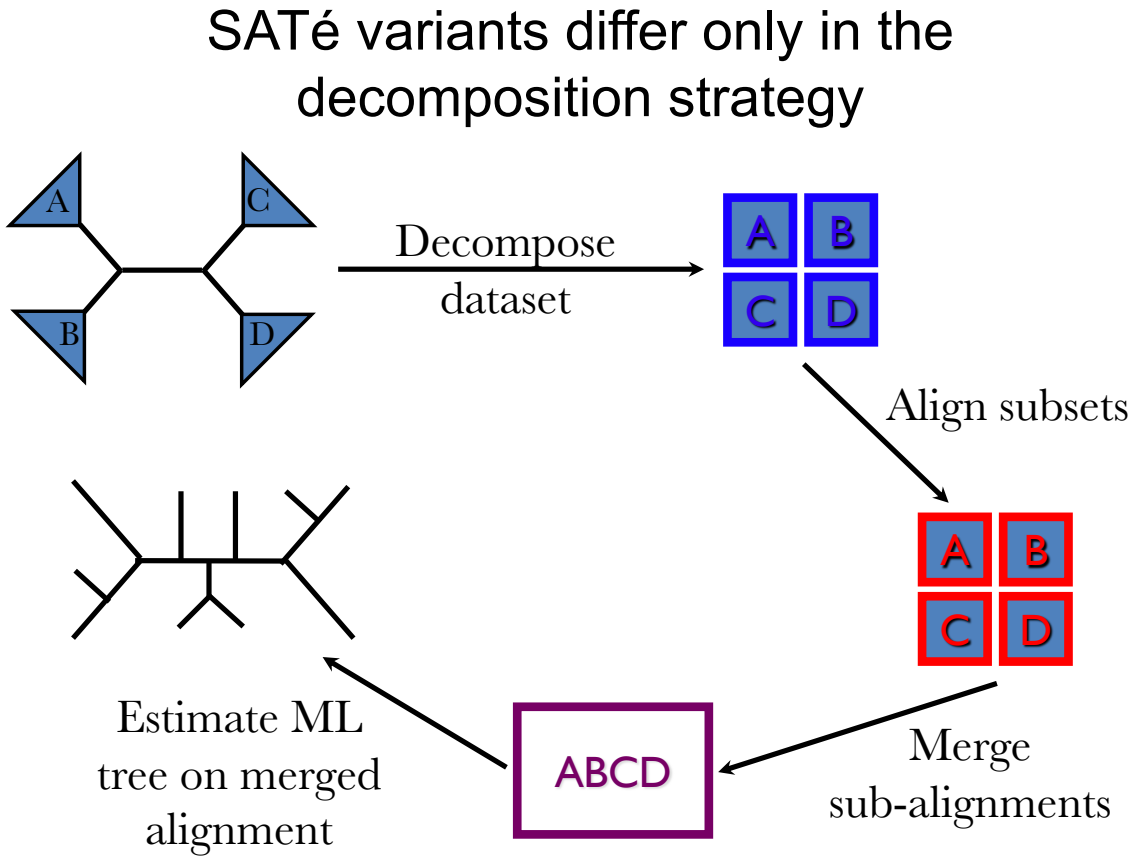
Divide-and-conquer using trees, cont.

The main objective of divide-and-conquer is to scale good MSA methods to larger datasets, so that they are more accurate or can analyze larger datasets.

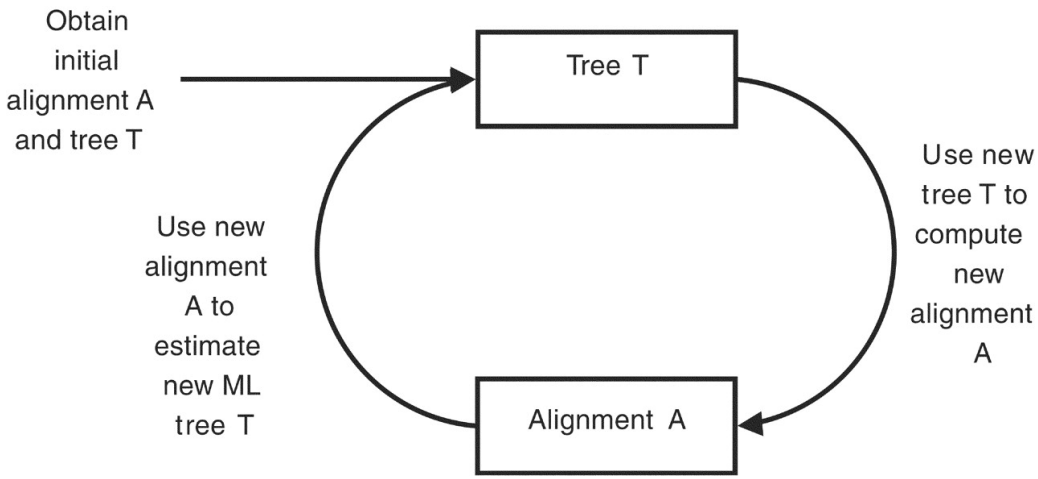
- Build a guide tree (perhaps by computing pairwise edit distances and then a tree based on the distances)
- Divide sequence dataset into disjoint subsets using the guide tree
- Align subsets
- Align alignments together (e.g., profile-profile alignment)



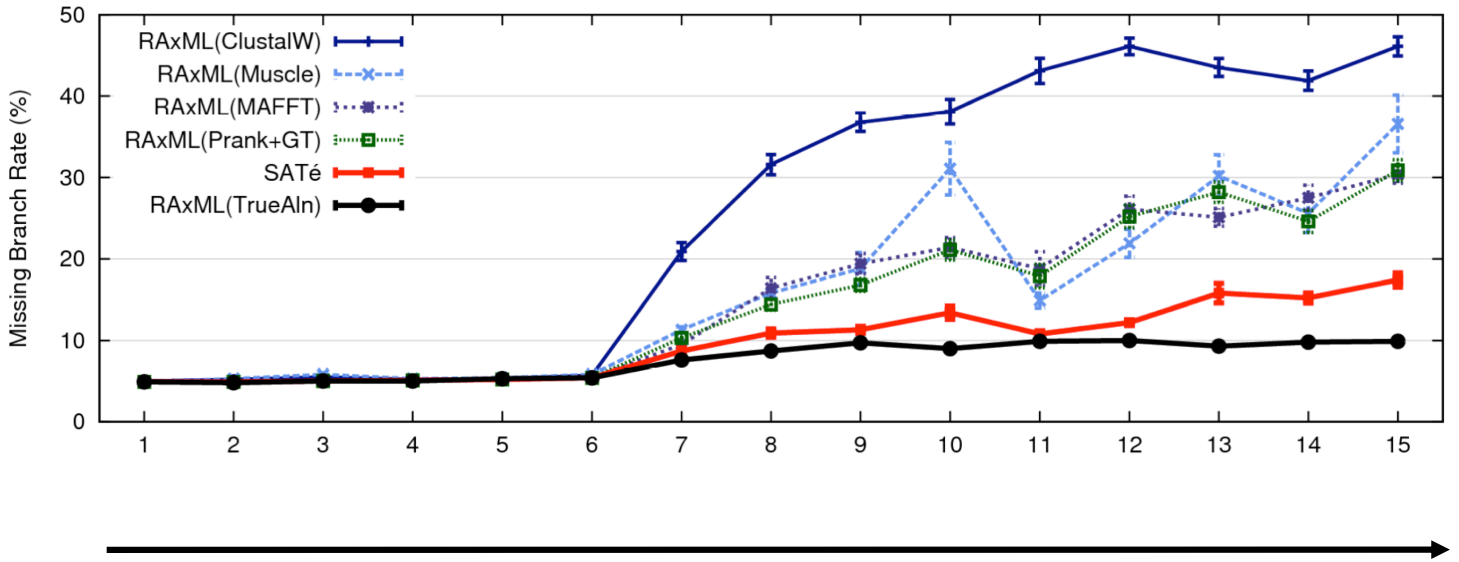
One SATé/PASTA iteration



Iteration between MSA and tree estimation



SATé-1 (Science 2009) performance



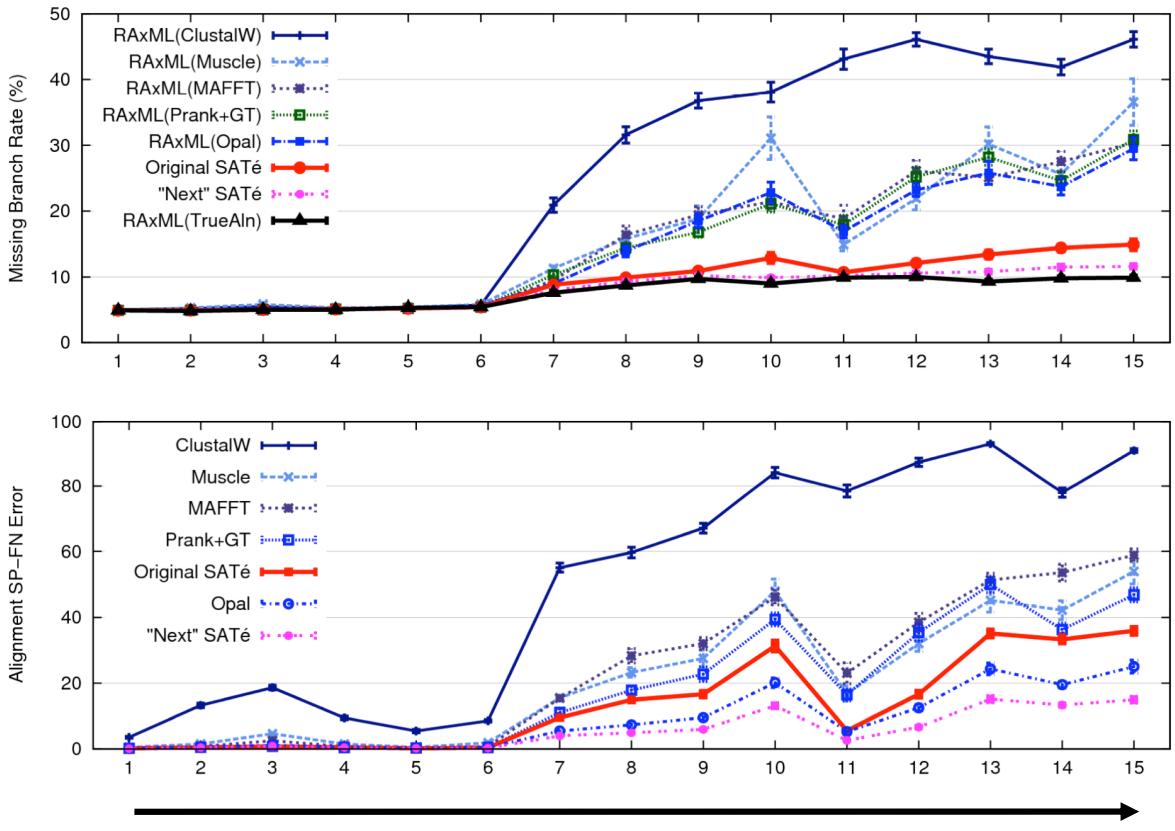
1000-taxon models, ordered by difficulty – rate of evolution generally increases from left to right

SATé-1 24 hour analysis, on desktop machines

(Similar improvements for biological datasets)

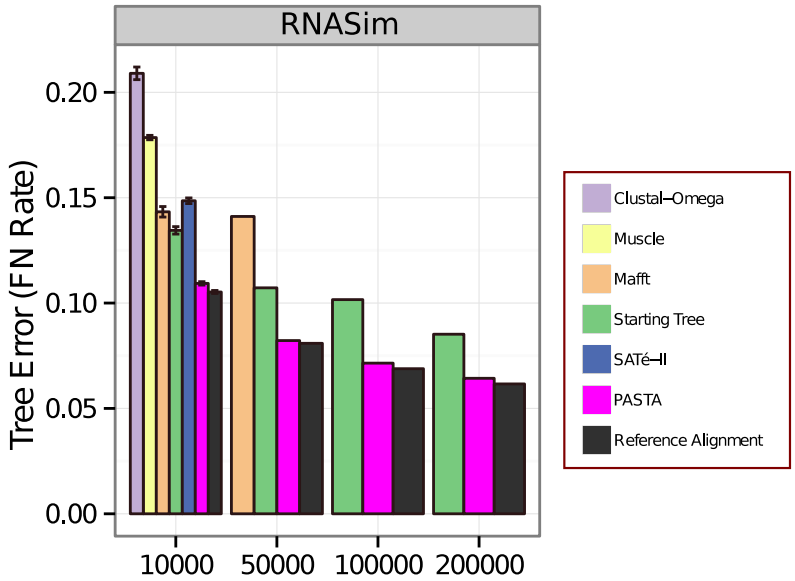
SATé-1 can analyze up to about 8,000 sequences.

SATé-1 and SATé-2 (Systematic Biology, 2012)



SATé-1: up to 8K
SATé-2: up to ~50K

Tree accuracy



1 million sequences:

- PASTA finished one iteration in 15 days
- PASTA tree had 6% error, compared to 5.6% when using true alignment
- Starting tree had 8.4% error

Summary

Accurate large-scale MSA estimation is challenging!

But various techniques can help improve accuracy and scalability, including:

- divide-and-conquer
- iteration
- consistency
- statistical methods