Phylogenetic Reconstruction (2)

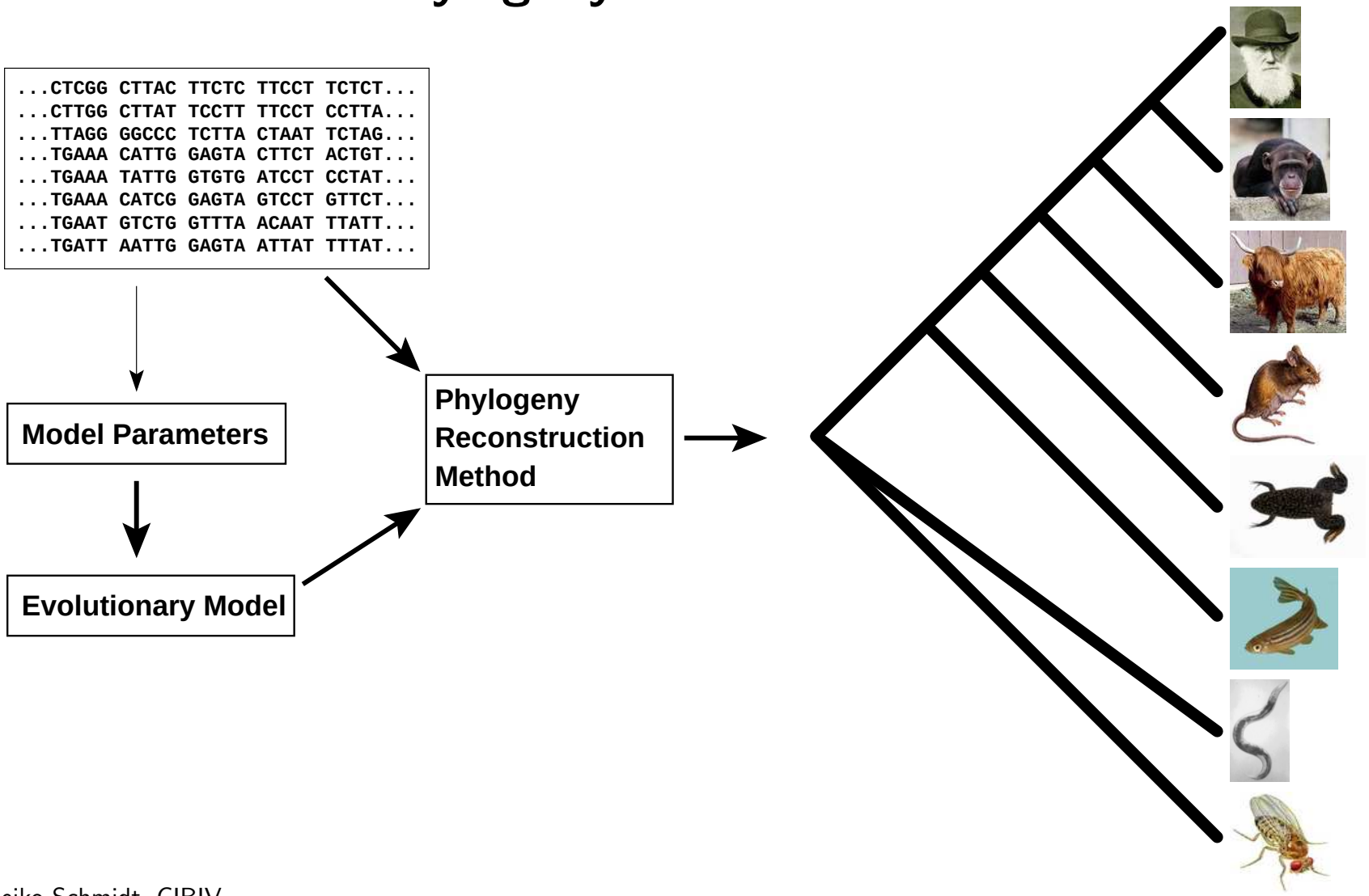
TREE-PUZZLE, parallel computing, obstacles, problems, wishes. . .

Heiko A. SCHMIDT

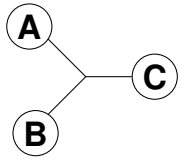
BUI Quang Minh, Arndt von HAESELER

Center for Integrative Bioinformatics Vienna (CIBIV)
Max F. Perutz Laboratories (MFPL)
Vienna
heiko.schmidt@univie.ac.at

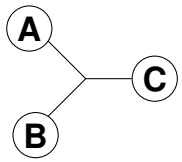
Phylogeny-Reconstruction

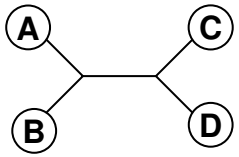


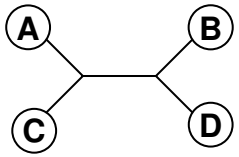
Binary Unrooted Trees

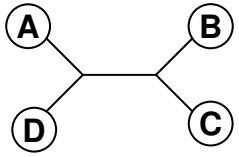


Binary Unrooted Trees

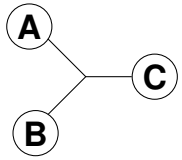


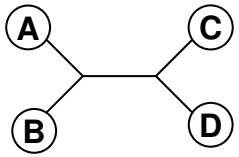


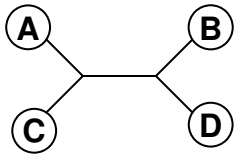


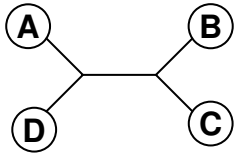


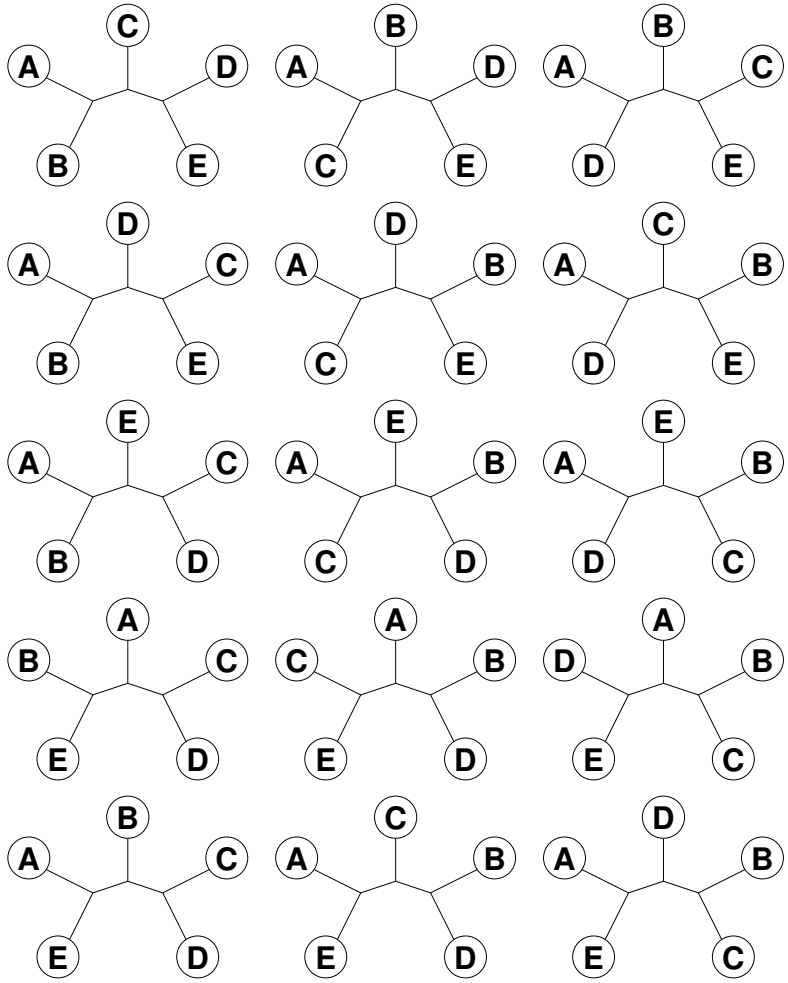
Binary Unrooted Trees



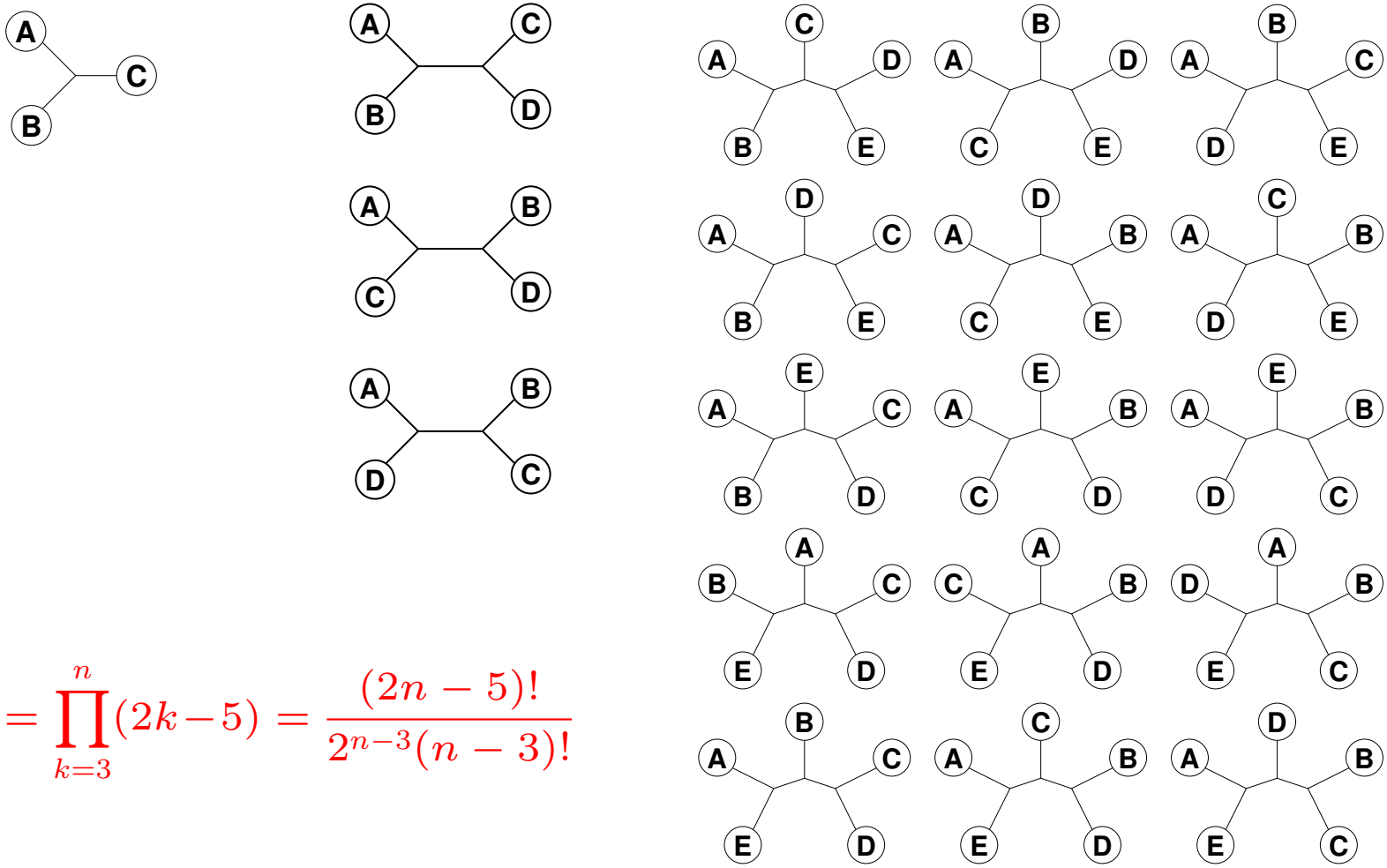








Binary Unrooted Trees



$$B(n) = \prod_{k=3}^{n}(2k-5) = \frac{(2n-5)!}{2^{n-3}(n-3)!}$$

Quartet Methods

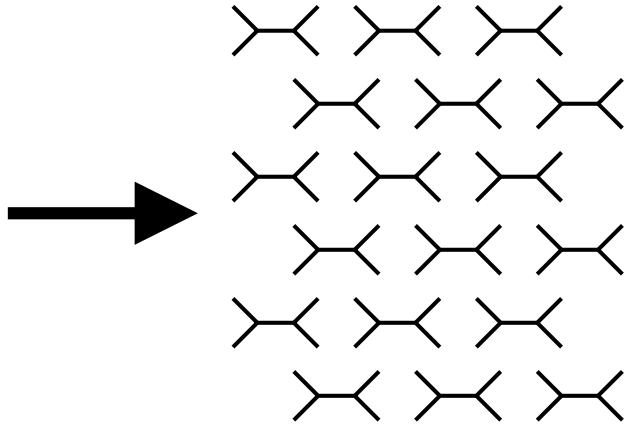
```
...CTCGG CTTAC TTCTC TTCCT...
...CTTGG CTTAT TCCTT TTCCT...
...TTAGG GGCCC TCTTA CTAAT...
...TGAAA CATTG GAGTA CTTCT...
...TGAAA TATTG GTGTG ATCCT...
...TGAAA CATCG GAGTA GTCCT...
...TGAAT GTCTG GTTTA ACAAT...
...TGATT AATTG GAGTA ATTAT...
```

Data

Heiko Schmidt, CIBIV

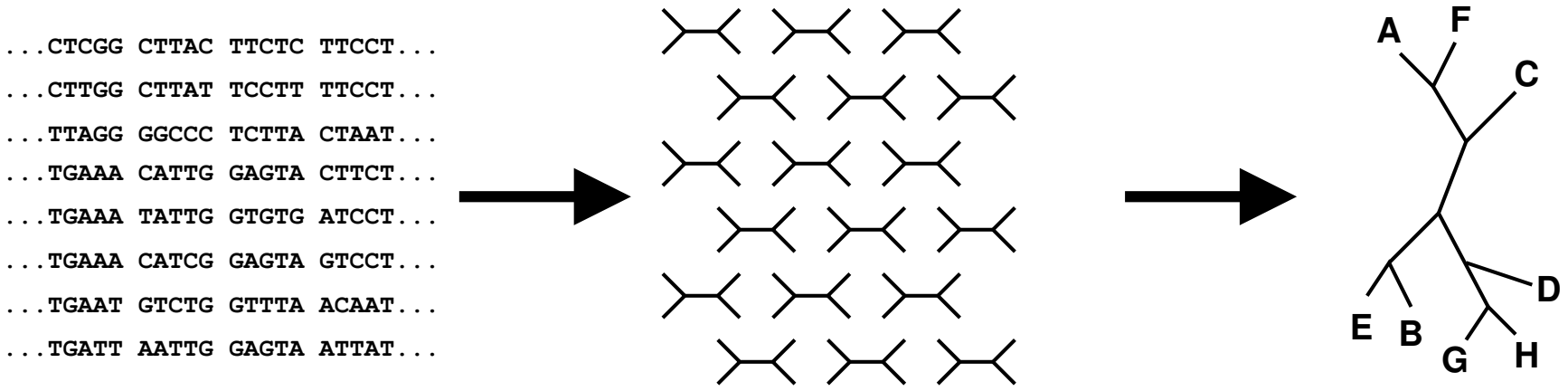
Quartet Methods

```
...CTCGG CTTAC TTCTC TTCCT...
...CTTGG CTTAT TCCTT TTCCT...
...TTAGG GGCCC TCTTA CTAAT...
...TGAAA CATTG GAGTA CTTCT...
...TGAAA TATTG GTGTG ATCCT...
...TGAAA CATCG GAGTA GTCCT...
...TGAAT GTCTG GTTTA ACAAT...
...TGATT AATTG GAGTA ATTAT...
```



Data **Quartet trees**

Quartet Methods

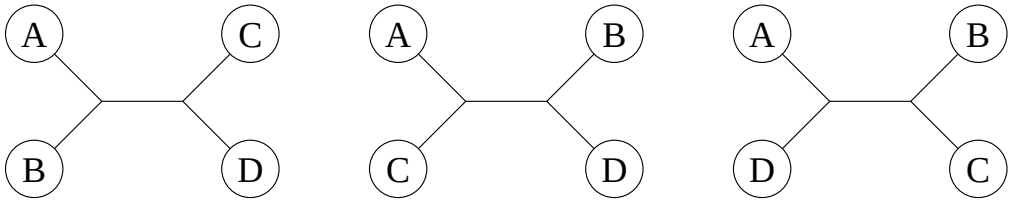


Data **Quartet trees** **Tree**

Why Quartets?

Why Quartets?

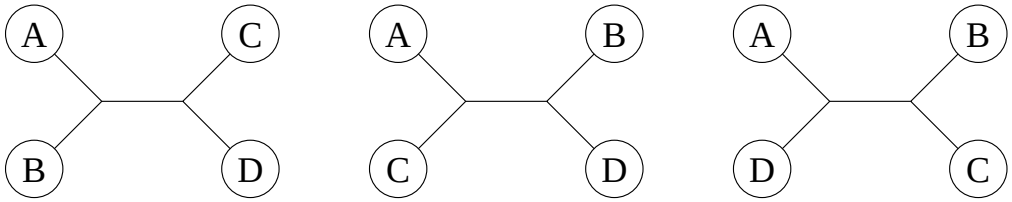
- There are only 3 topologies for 4 sequences.



$AB|CD$ $AC|BD$ $AD|BC$

Why Quartets?

- There are only 3 topologies for 4 sequences.

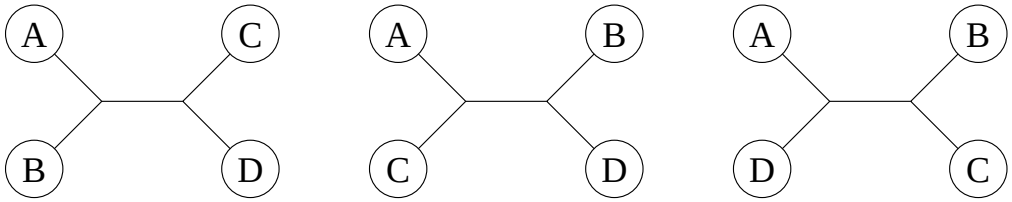


$AB|CD$ $AC|BD$ $AD|BC$

- Quartet topologies can easily be evaluated by ML (and other methods).

Why Quartets?

- There are only 3 topologies for 4 sequences.



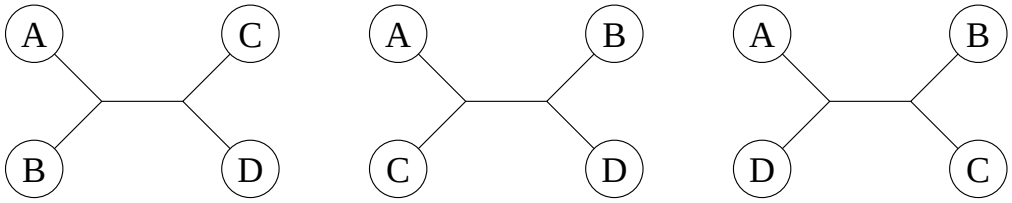
$AB|CD$ $AC|BD$ $AD|BC$

- Quartet topologies can easily be evaluated by ML (and other methods).

- Although the number of quartets $\binom{n}{4}$ can be large,
  this is out-numbered by the number of full trees.

Why Quartets?

- There are only 3 topologies for 4 sequences.



$AB|CD$ $AC|BD$ $AD|BC$

- Quartet topologies can easily be evaluated by ML (and other methods).

- Although the number of quartets $\binom{n}{4}$ can be large, this is out-numbered by the number of full trees.

- Hence, the computational burden is reduced from huge equations on huge numbers of large trees to 'small' equations on a large number of 4-trees

Quartet Puzzling

The Quartet Puzzling algorithm implemented in the TREE-PUZZLE program is a four-step procedure:

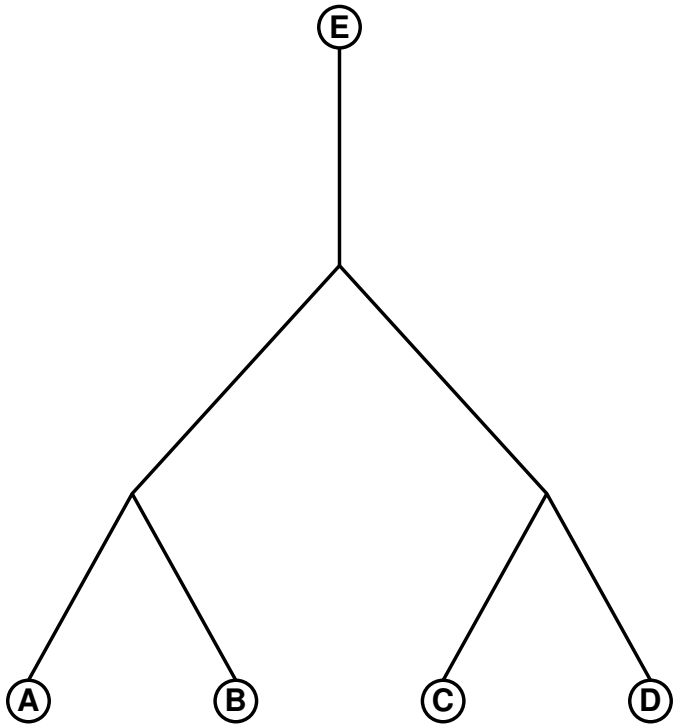
1) model parameter estimation

2) maximum-likelihood step: compute ML trees for all quartets of an alignment.

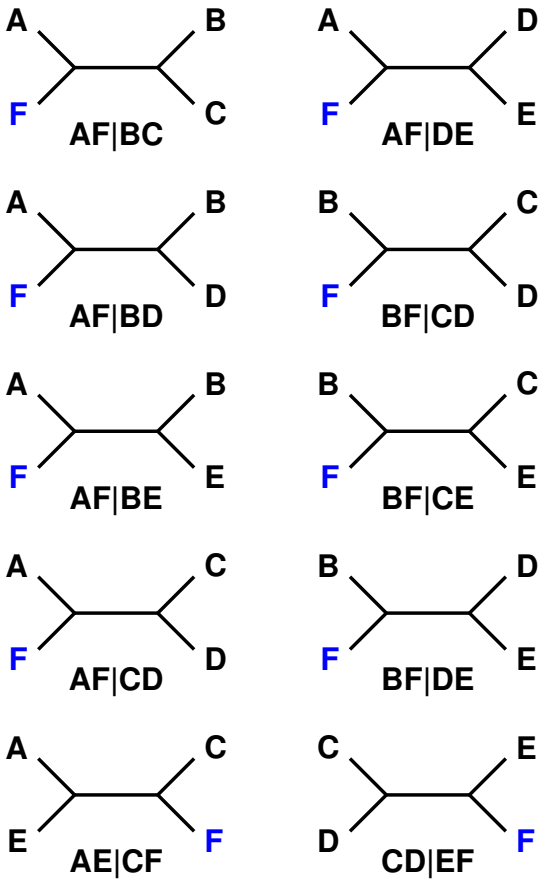
3) puzzling step: compose an intermediate tree from quartet trees.
Starts with a quartet topology and adds sequences by stepwise insertion. (this is done multiple times).

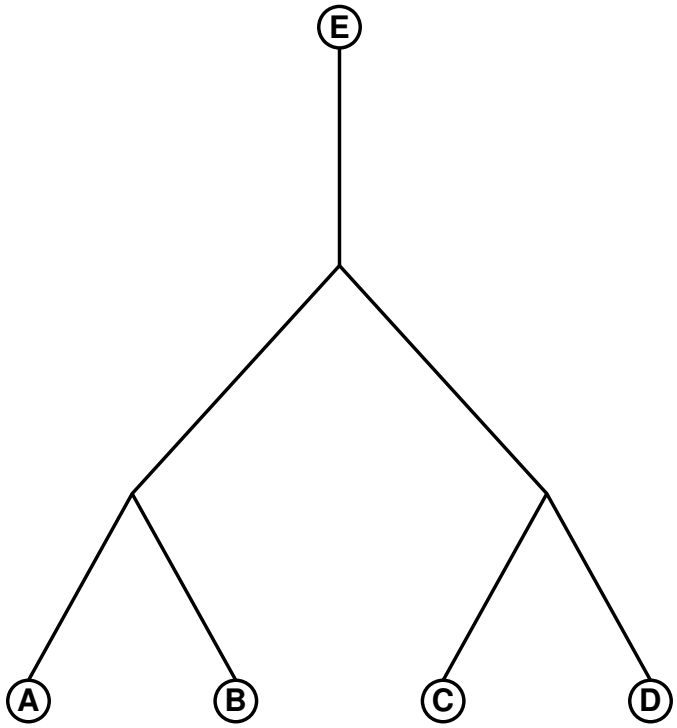
4) consensus step: construct a majority rule consensus tree from the intermediate trees and evaluate the branch lengths.

Puzzling Step: Quartets $\longrightarrow$ Full Tree

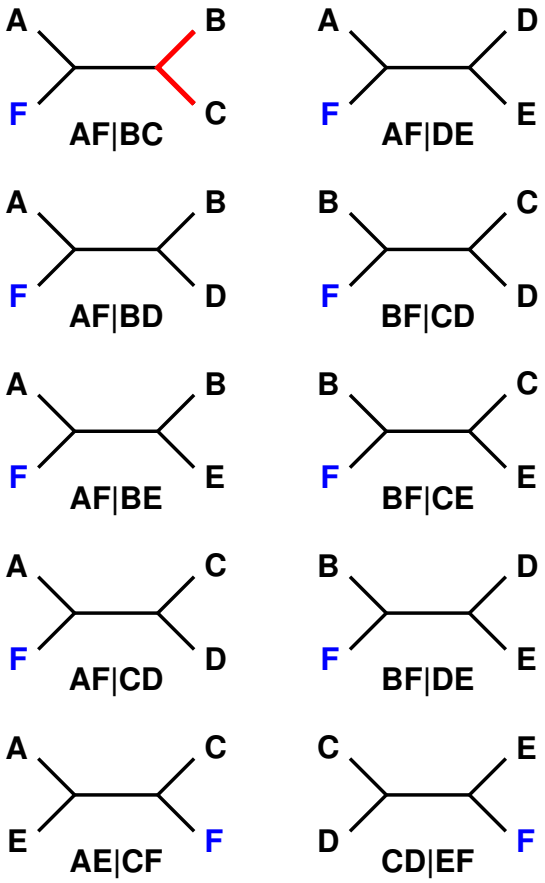


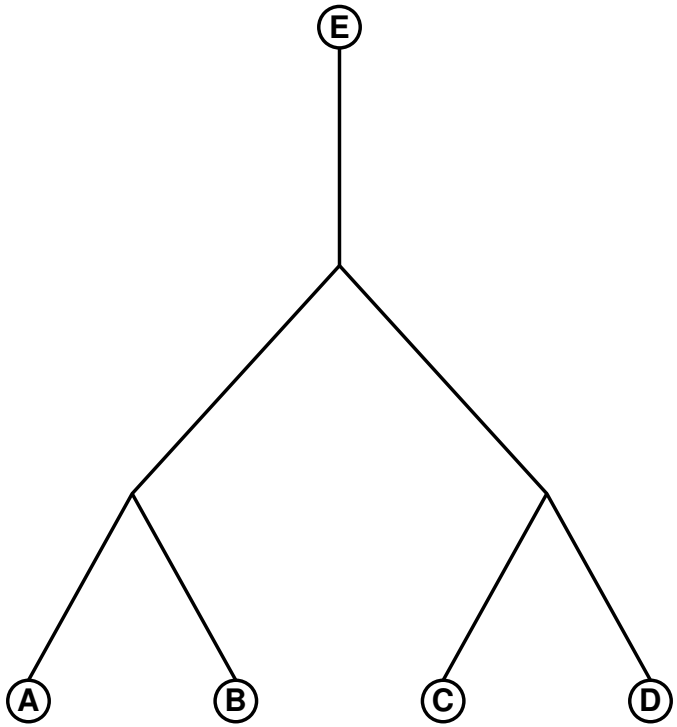
Puzzling Step: Quartets → Full Tree



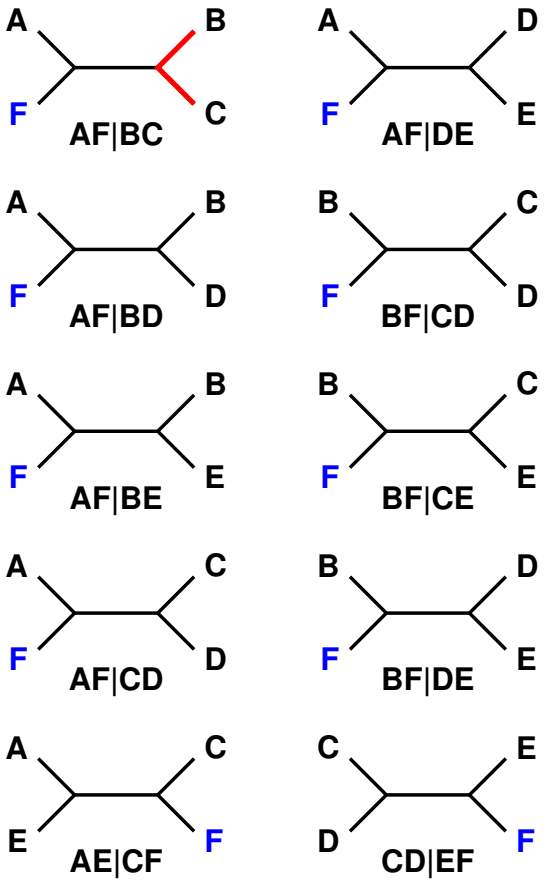


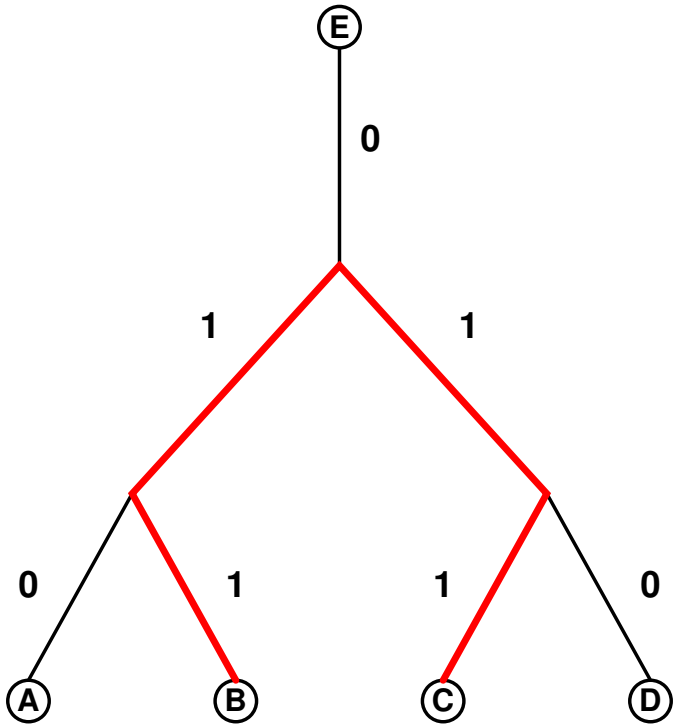
Puzzling Step: Quartets → Full Tree



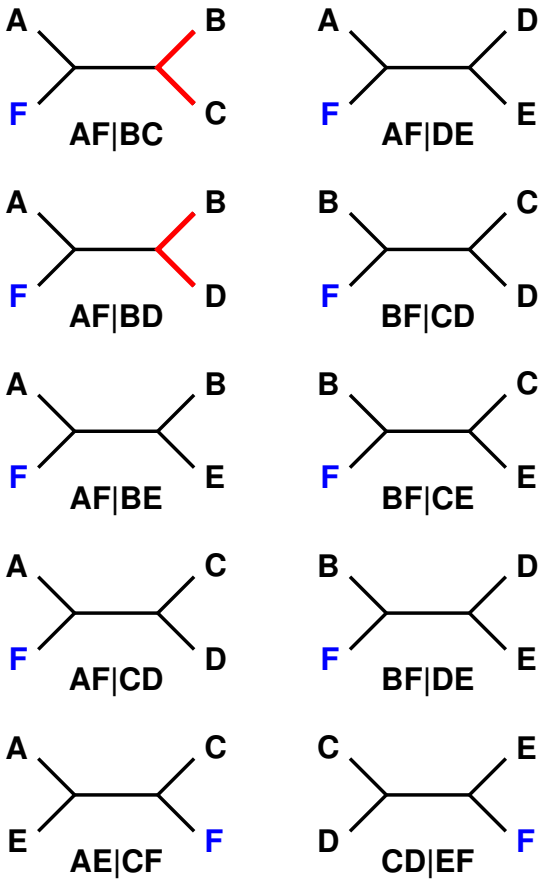


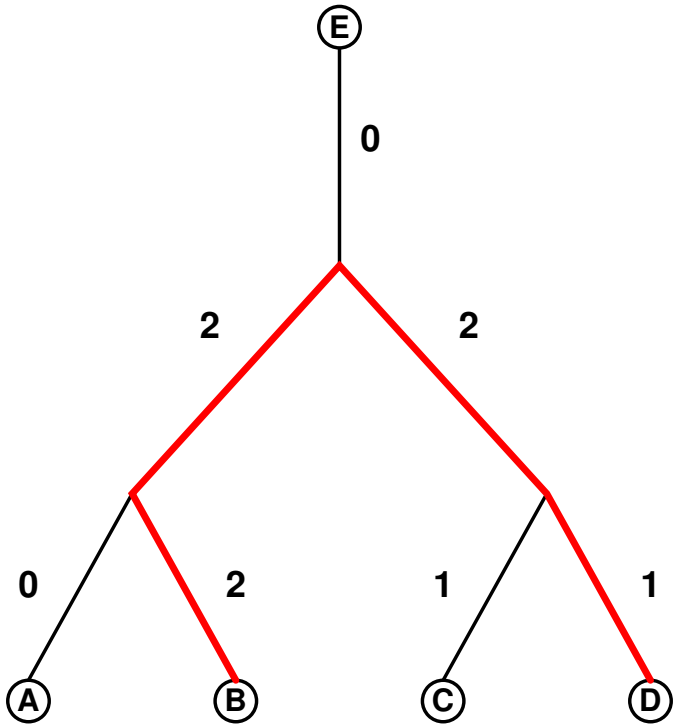
Puzzling Step: Quartets → Full Tree



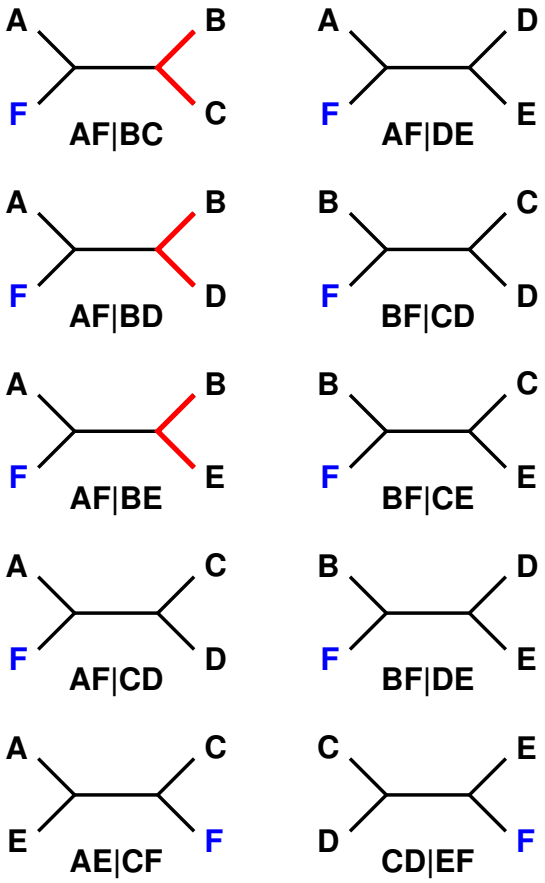


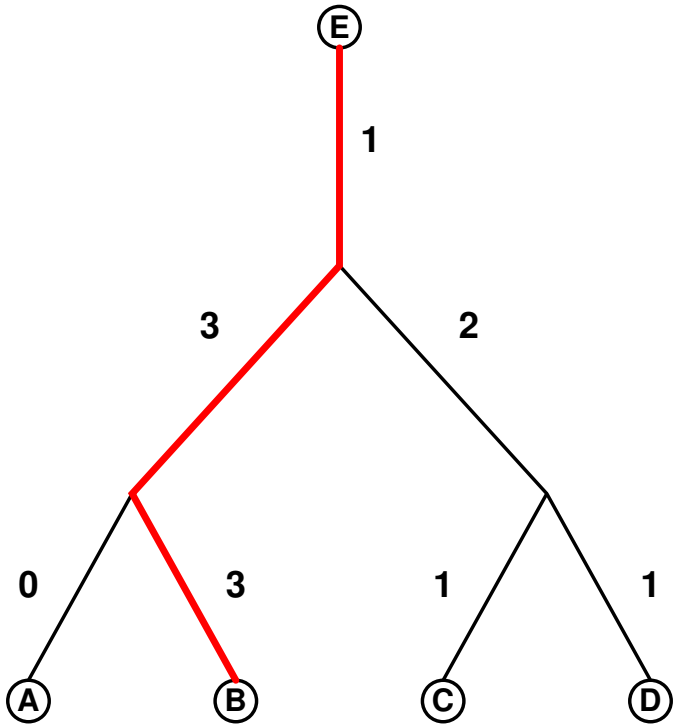
Puzzling Step: Quartets → Full Tree



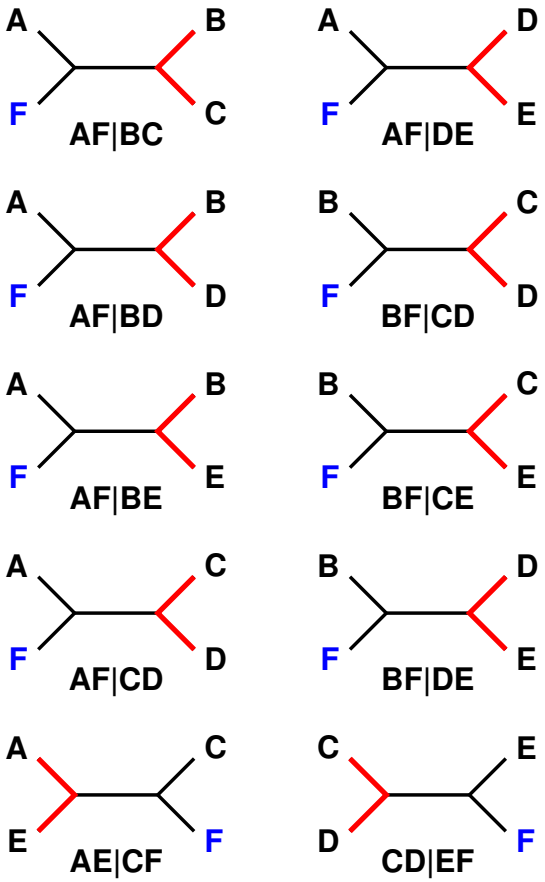


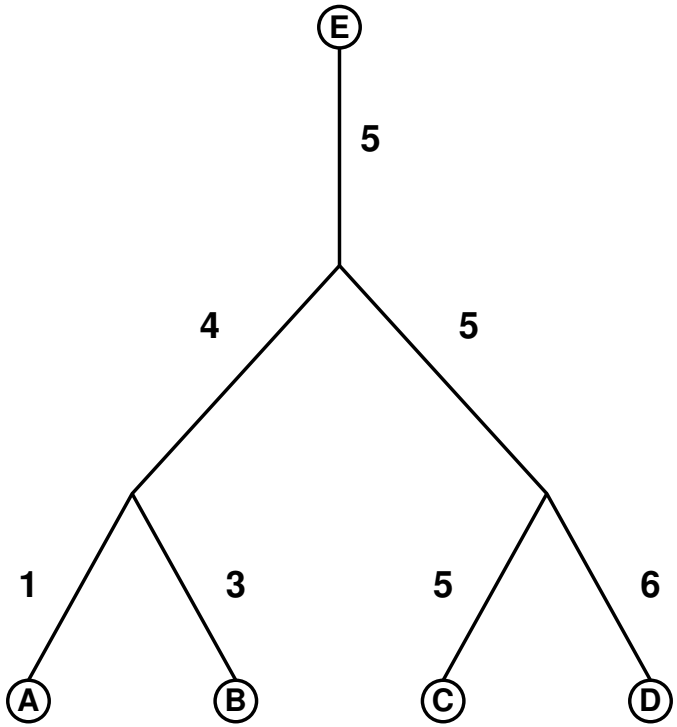
Puzzling Step: Quartets → Full Tree



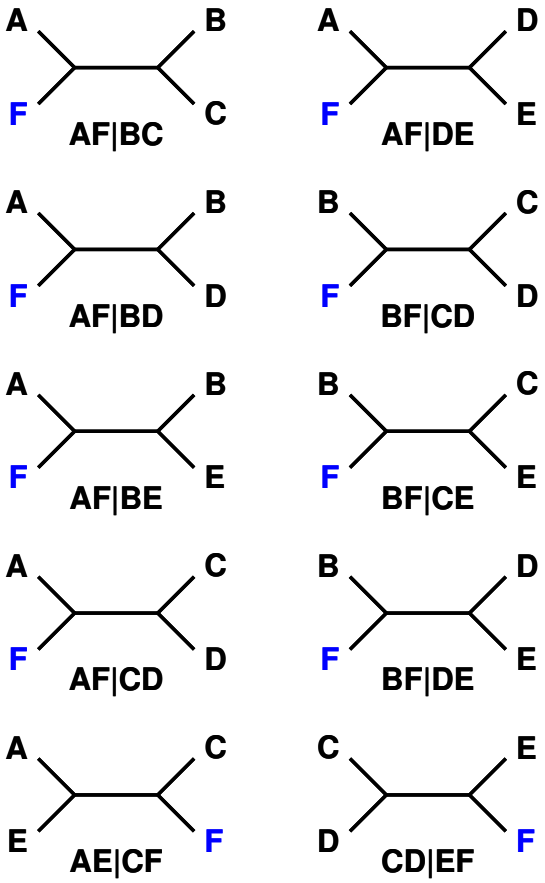


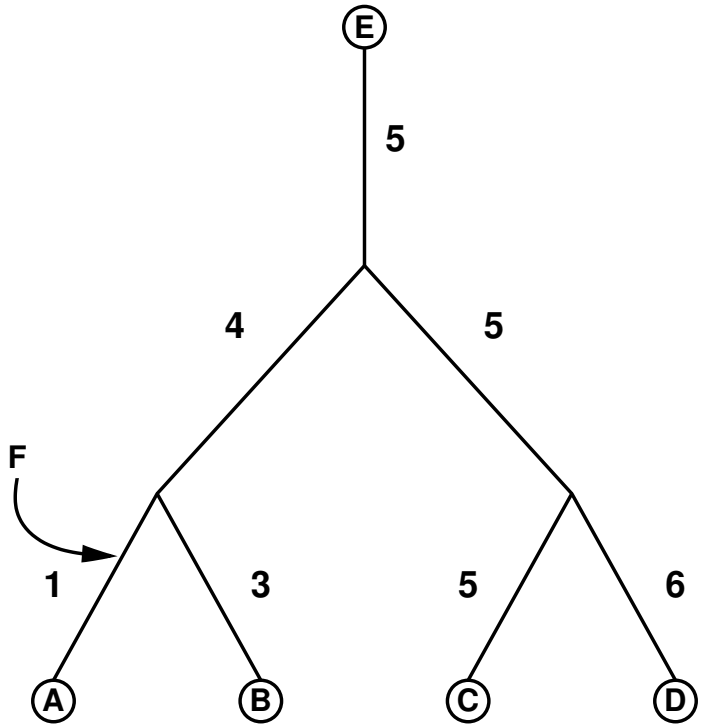
Puzzling Step: Quartets → Full Tree





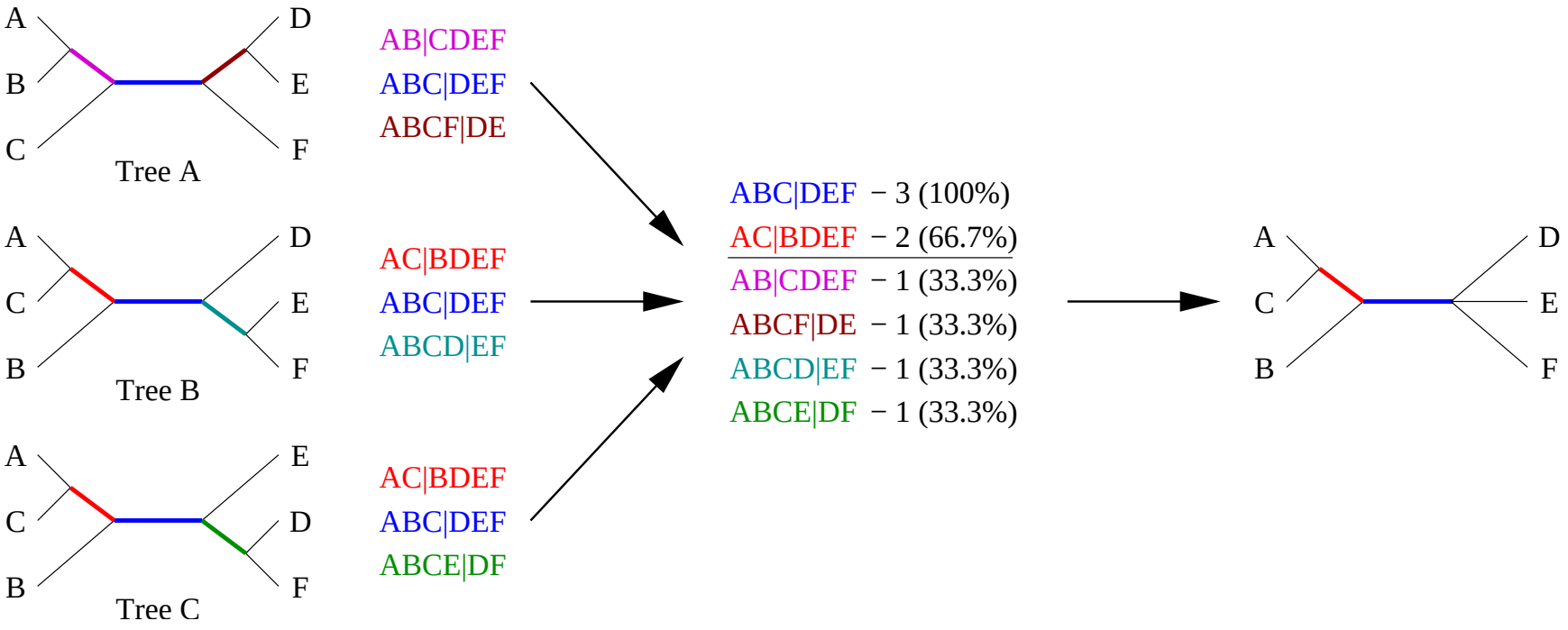
Puzzling Step: Quartets → Full Tree





Consensus Step

From the collection of trees from the puzzling step we compute a majority consensus tree:



Speeding up TREE-PUZZLE

Although the Quartet Puzzling reduces the runtime compared to exhaustive searches, it still needs long time for large datasets.

Speeding up TREE-PUZZLE

Although the Quartet Puzzling reduces the runtime compared to exhaustive searches, it still needs long time for large datasets.

- More efficient puzzling step's algorithm: $O(n^5) \rightarrow O(n^4)$

Speeding up TREE-PUZZLE

Although the Quartet Puzzling reduces the runtime compared to exhaustive searches, it still needs long time for large datasets.

- More efficient puzzling step's algorithm: $O(n^5) \rightarrow O(n^4)$
  Saves more than 50% runtime of puzzling step

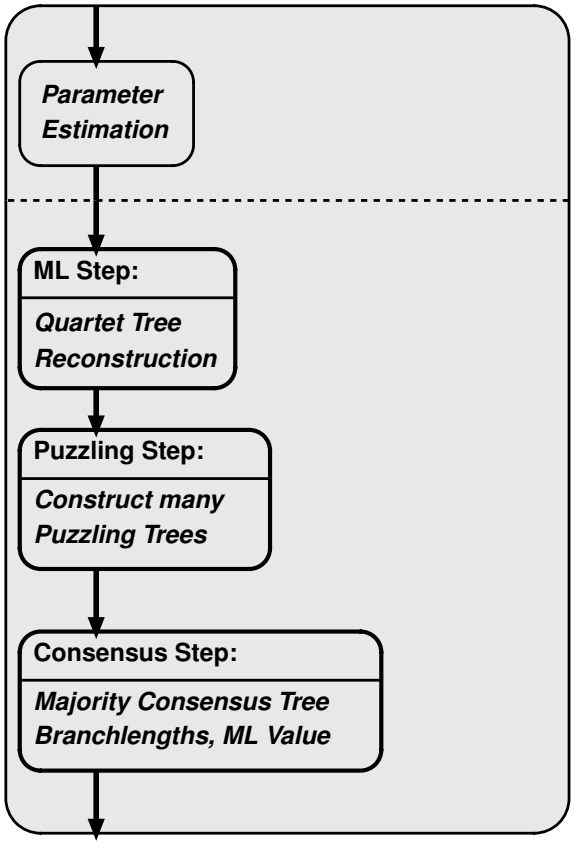
Speeding up TREE-PUZZLE

Although the Quartet Puzzling reduces the runtime compared to exhaustive searches, it still needs long time for large datasets.

- More efficient puzzling step's algorithm: $O(n^5) \rightarrow O(n^4)$
  Saves more than 50% runtime of puzzling step

- Parallel Computing

The *Quartet Puzzling* Algorithm

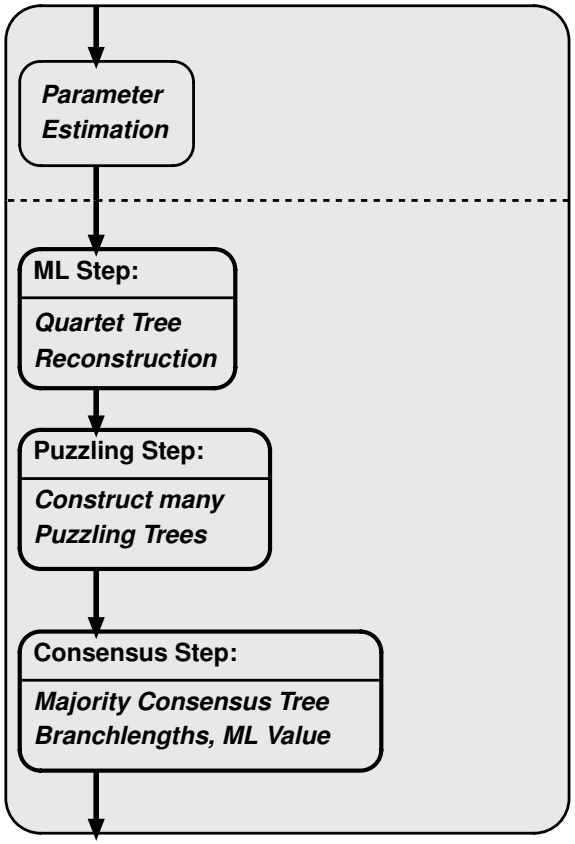


The *Quartet Puzzling* Algorithm

Runtime-Profile

64 sequences, 1000 bp, 11 h

- Parameters: 1.27 %
- ML Step: 43.15 %
- Puzzling Step: 55.53 %
- Consensus Step: 0.05 %

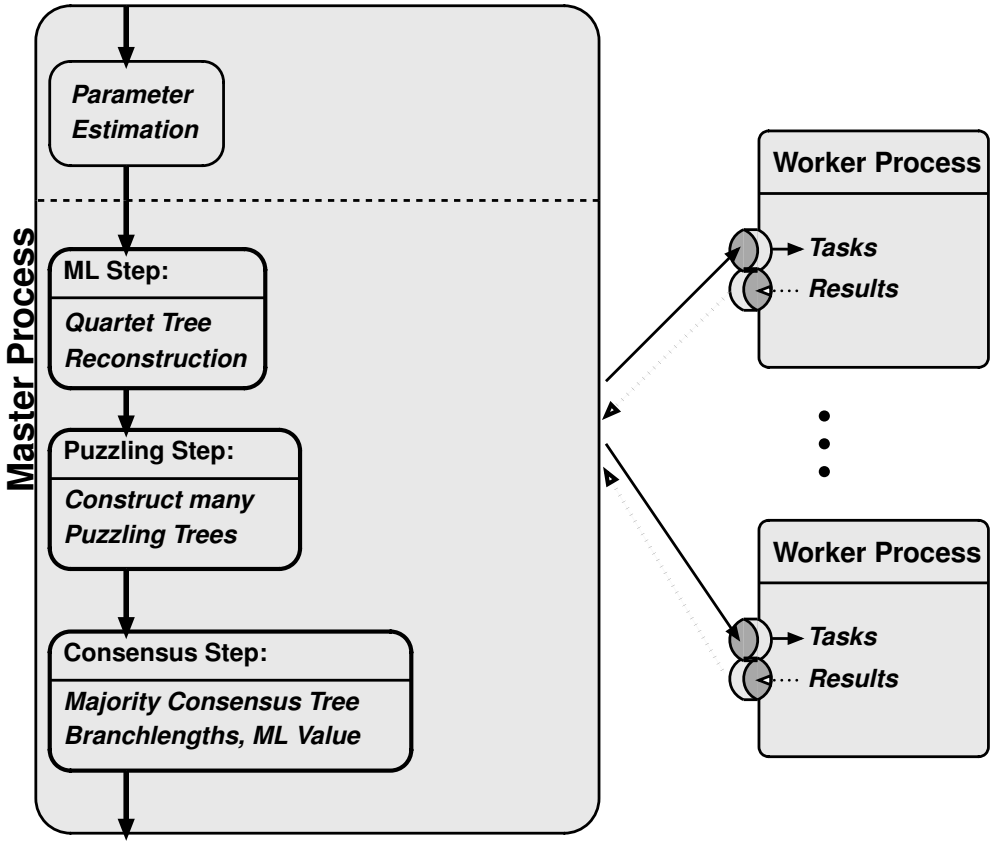


The *Quartet Puzzling* Algorithm

Runtime-Profile

64 sequences, 1000 bp, 11 h

- Parameters: 1.27 %
- ML Step: 43.15 %
- Puzzling Step: 55.53 %
- Consensus Step: 0.05 %

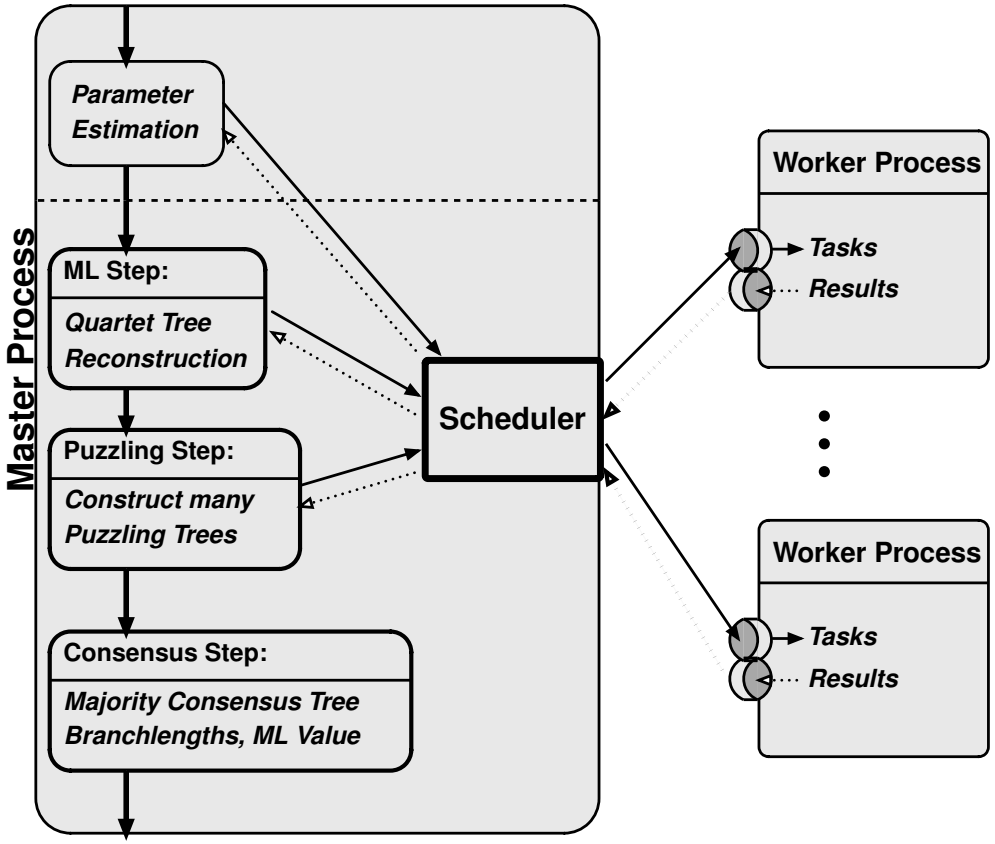


The *Quartet Puzzling* Algorithm

Runtime-Profile

64 sequences, 1000 bp, 11 h

- Parameters: 1.27 %
- ML Step: 43.15 %
- Puzzling Step: 55.53 %
- Consensus Step: 0.05 %

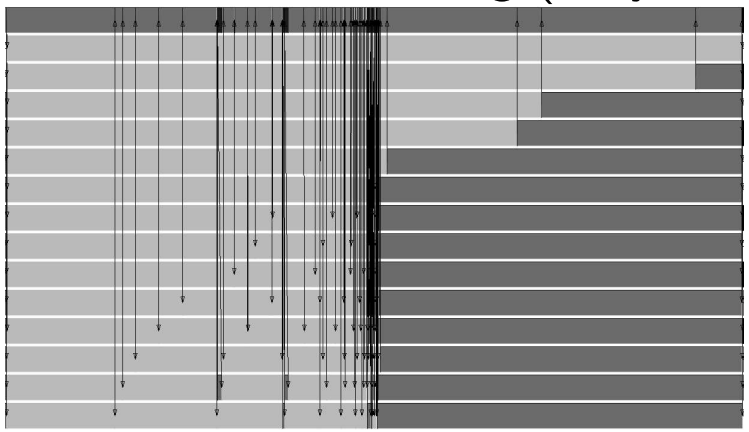


Scheduling Algorithm (heterogeneous SUN-Cluster)

- Guided Self Scheduling (Polychronopoulos/Kuck, 1987; Hagerup, 1997)

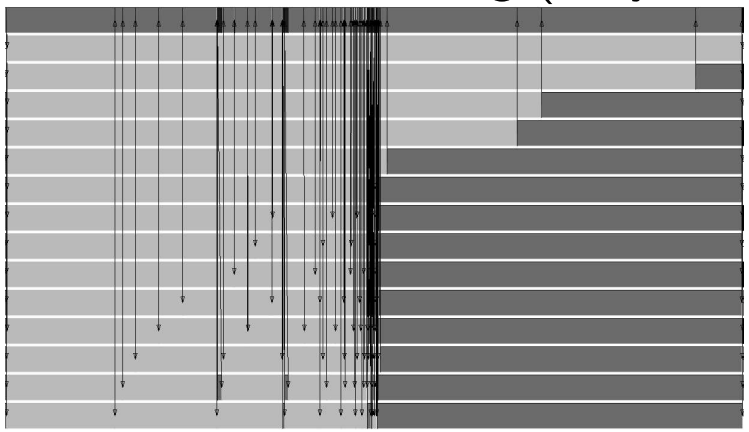
Scheduling Algorithm (heterogeneous SUN-Cluster)

- Guided Self Scheduling (Polychronopoulos/Kuck, 1987; Hagerup, 1997)

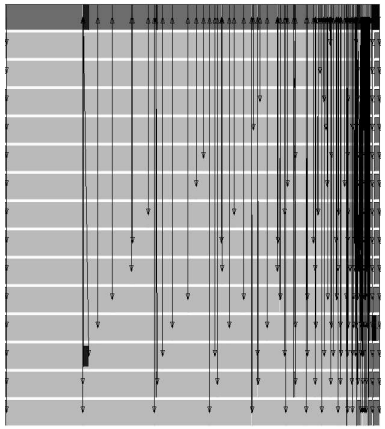


Scheduling Algorithm (heterogeneous SUN-Cluster)

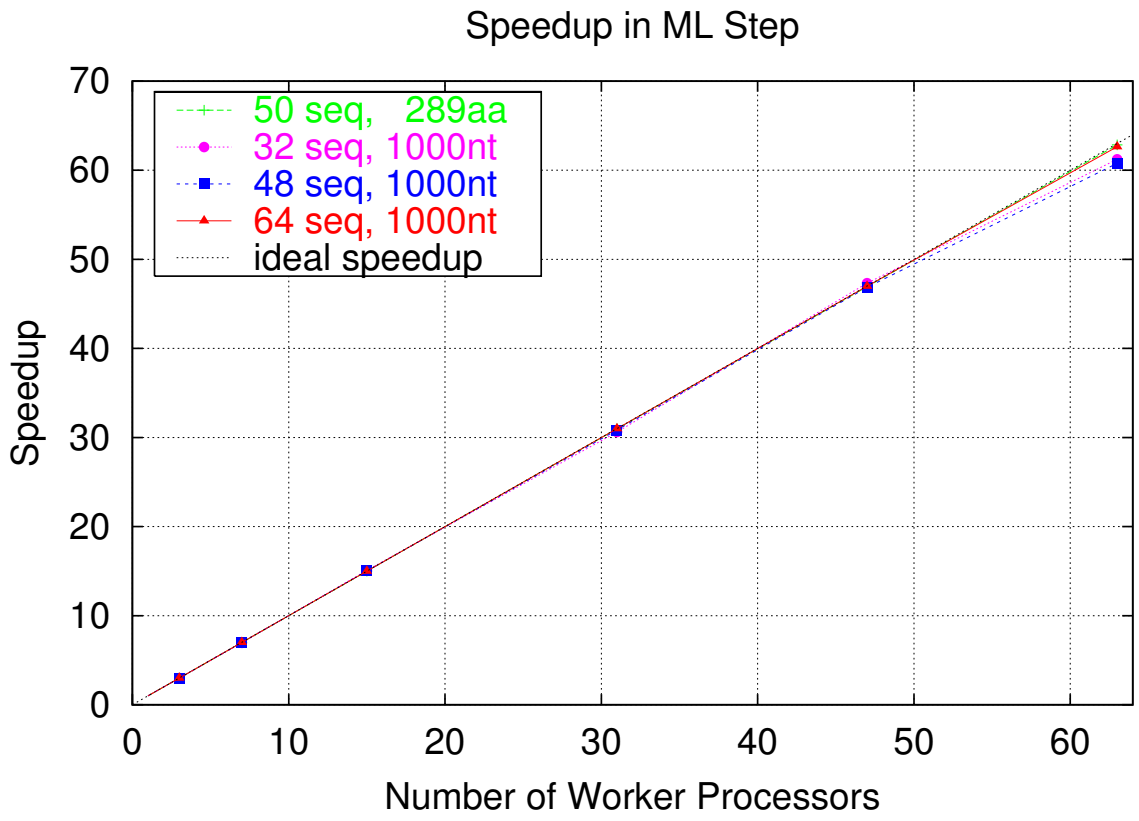
- Guided Self Scheduling (Polychronopoulos/Kuck, 1987; Hagerup, 1997)



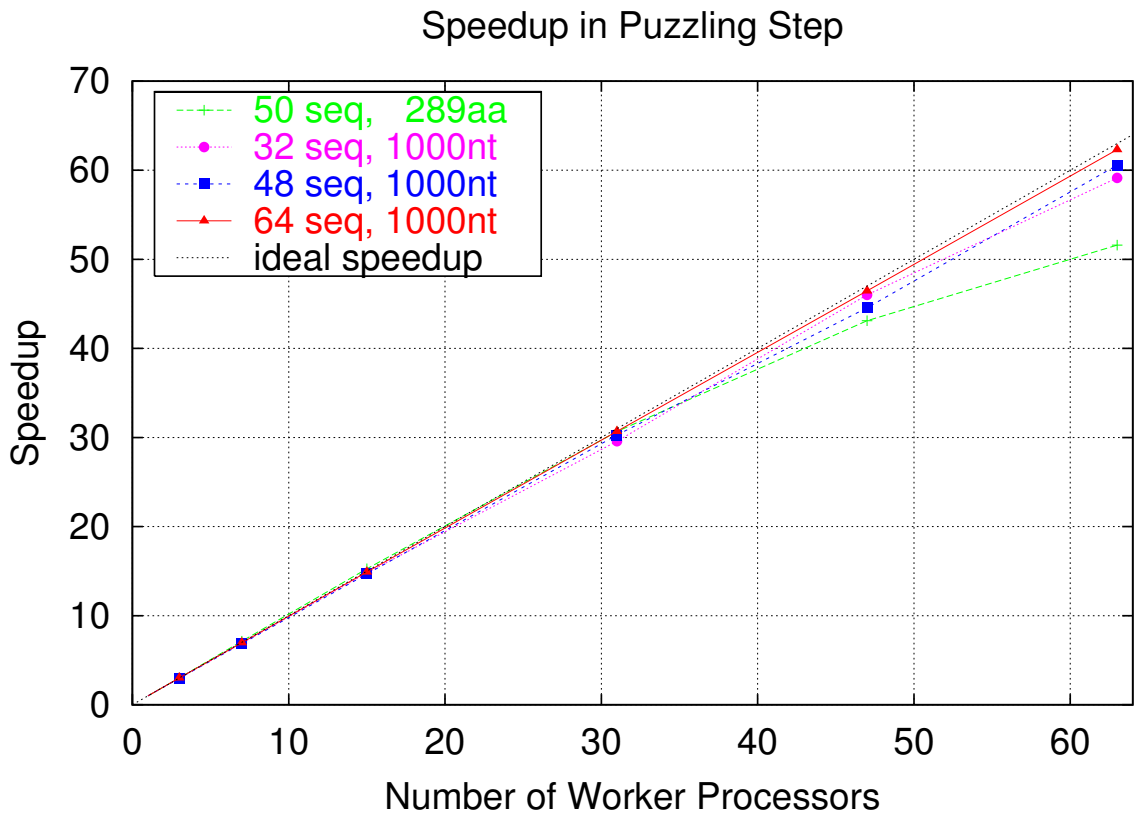
- Smooth Guided Self Scheduling (Schmidt et al., 2002)



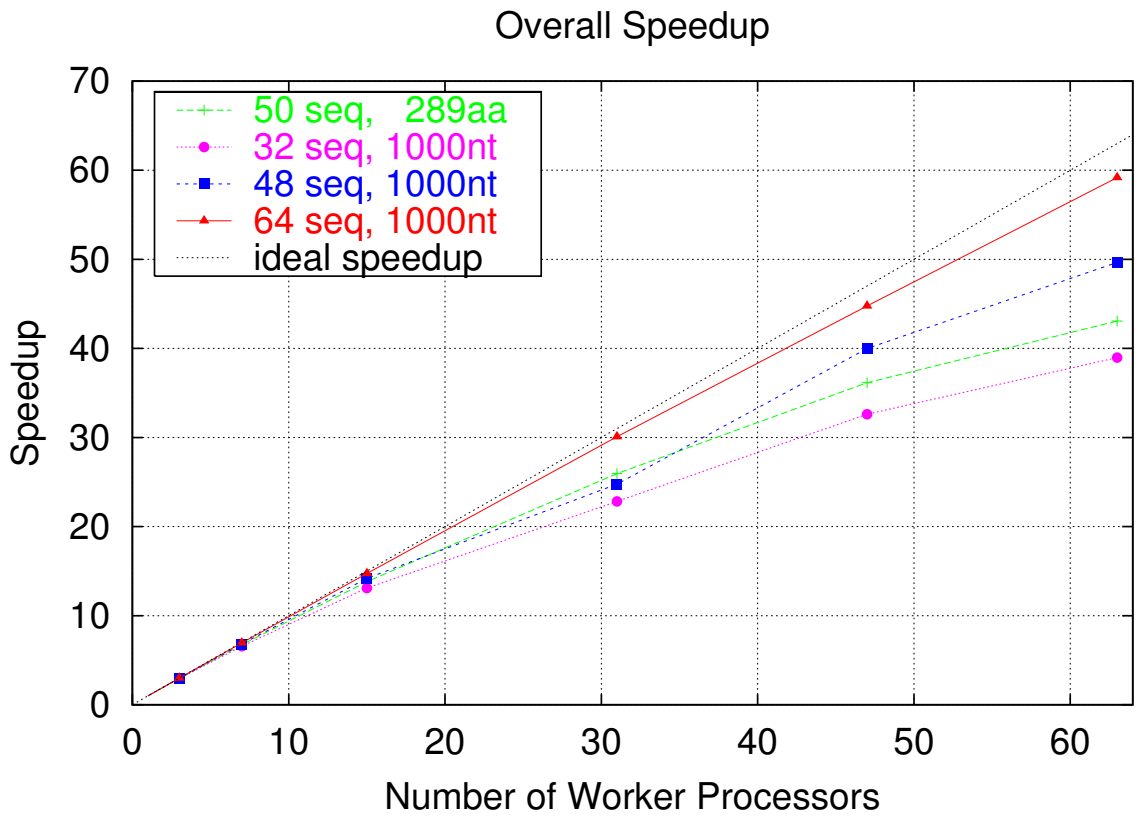
Parallel ML Step



Parallel Puzzling Step



Parallel TREE-PUZZLE



Speedup by Optimization and Parallel Computing

Dataset: 64 sequences, 1000 bp, rate heterogeneity

11 h (original algorithm)

Speedup by Optimization and Parallel Computing

Dataset: 64 sequences, 1000 bp, rate heterogeneity

11 h (original algorithm)

$\rightarrow$ 7.8 h (recursive algorithm)

Speedup by Optimization and Parallel Computing

Dataset: 64 sequences, 1000 bp, rate heterogeneity

11 h (original algorithm)

$\rightarrow$ 7.8 h (recursive algorithm)

$\rightarrow$ 2.6 h (4 processors)

Speedup by Optimization and Parallel Computing

Dataset: 64 sequences, 1000 bp, rate heterogeneity

11 h (original algorithm)

$\rightarrow$ 7.8 h (recursive algorithm)

$\rightarrow$ 2.6 h (4 processors)

$\rightarrow$ 66 min (8 processors)

Speedup by Optimization and Parallel Computing

Dataset: 64 sequences, 1000 bp, rate heterogeneity

11 h (original algorithm)

$\longrightarrow$ 7.8 h (recursive algorithm)

$\longrightarrow$ 2.6 h (4 processors)

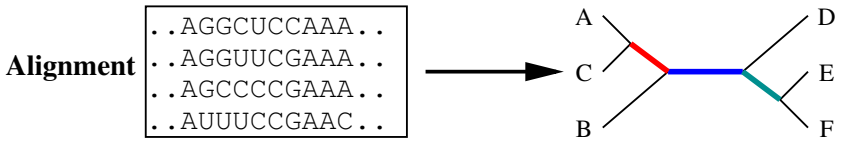
$\longrightarrow$ 66 min (8 processors)

$\longrightarrow$ 8 min (64 processors)

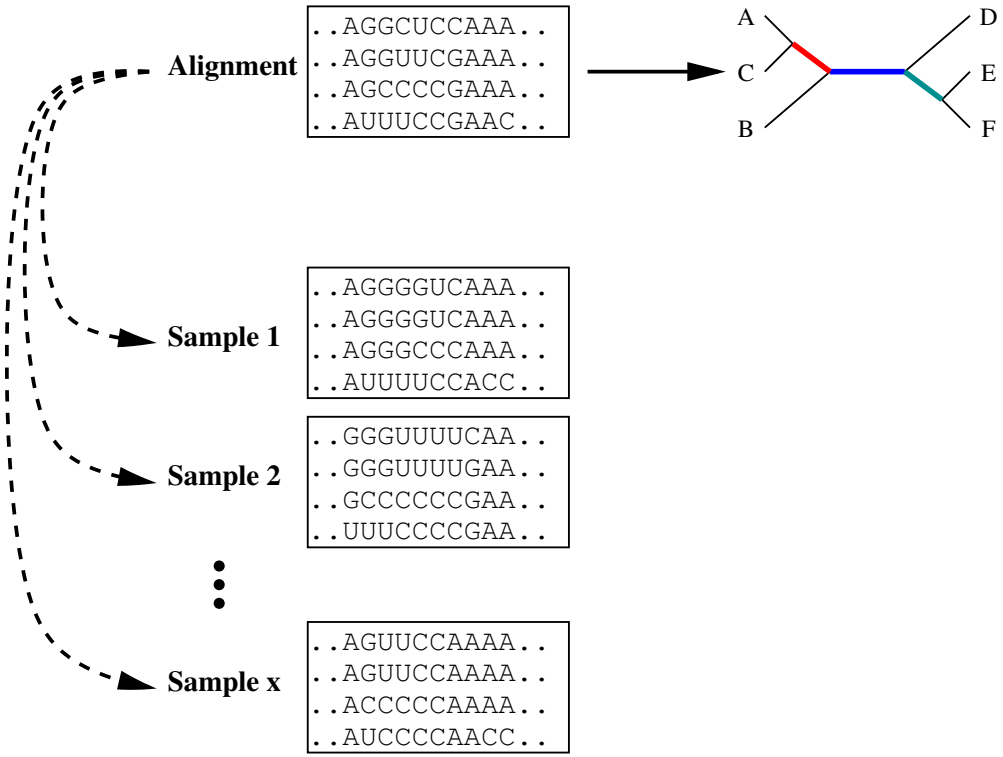
Confidence?

- TREE-PUZZLE produces support values in the consensus step.
- Bootstrap analysis.

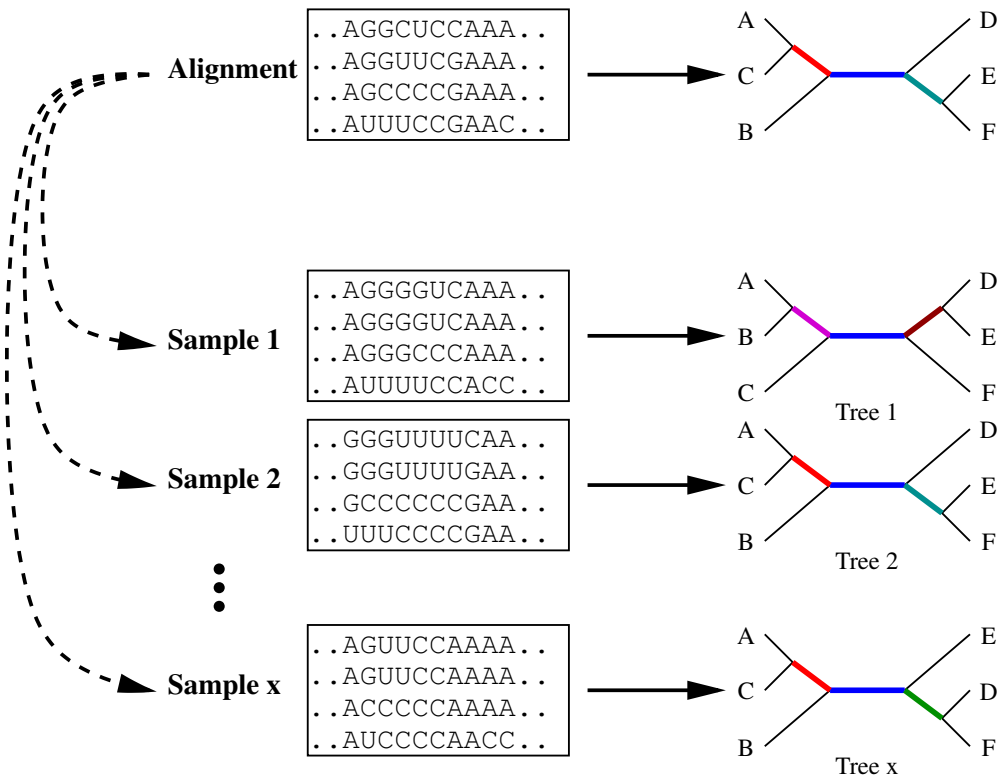
Estimating Confidence: The Bootstrap



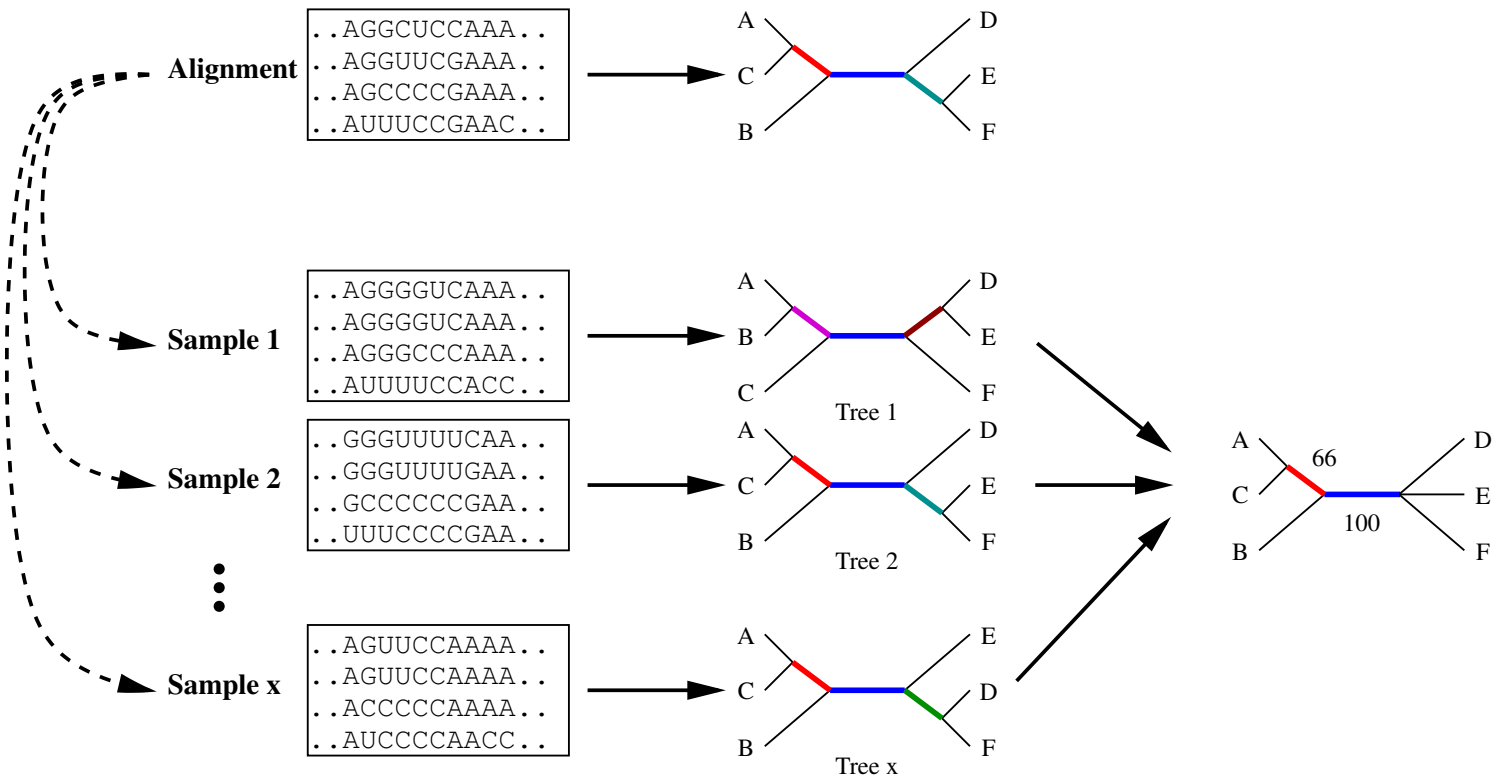
Estimating Confidence: The Bootstrap



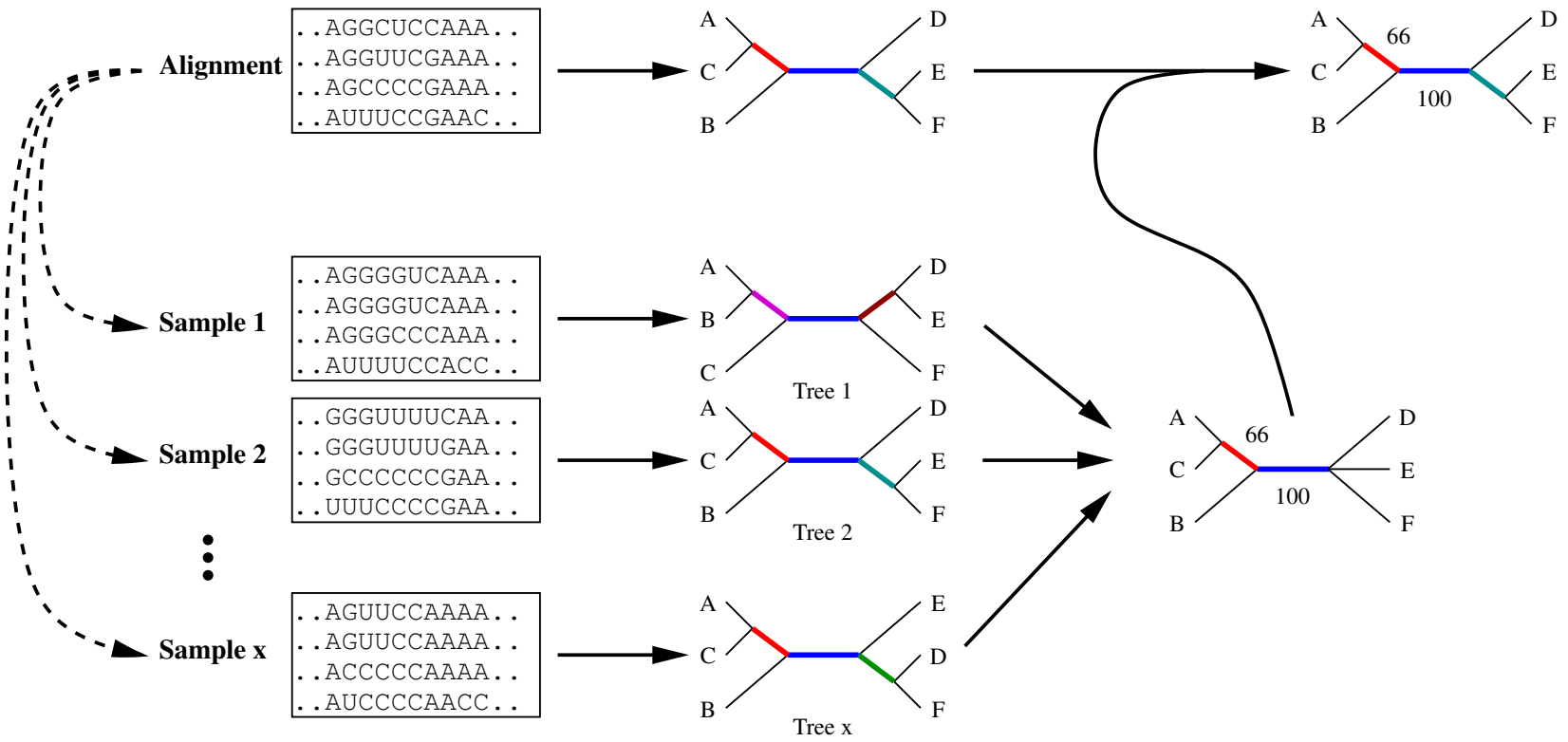
Estimating Confidence: The Bootstrap



Estimating Confidence: The Bootstrap



Estimating Confidence: The Bootstrap



Obstacles in the analysis: numbers

- large numbers of possible trees:

$$B(n) = \prod_{k=3}^{n}(2k-5) = \frac{(2n-5)!}{2^{n-3}(n-3)!}$$

- Tree optimizations do examine many trees [NNI: $O(n)$, SPR: $O(n^2)$, TBR: $O(n^3)$]

- large numbers of iterations (IQPNNI), generations (MCMC) or samples (Bootstrap analysis, > 100).

- large numbers of $\binom{n}{4}$ quartets (quartet methods, concerning time and memory)

Hopes/wishes for discussion. . .

- efficient and applicable (distributed) algorithms
- (distributed) data structures (e.g. sparse arrays of quartets with a lot of missing data)
- ideas to improve the current algorithms, parallelizations, communication. . .
- new ideas for different new heuristics
- (parallel) optimization techniques
- ideas for (parallel) simultaneous branch length optimization (dependencies!)
- . . .
- maybe there is a good way for a more general distributed system usable for other problems in Bioinformatics.