Models, Algorithms, and Parallel Computing for Large-Scale Phylogenetic Inference

Alexandros Stamatakis

The Exelixis Lab
Teaching and Research Unit Bioinformatics
Department of Computer Science
Ludwig-Maximilians-University Munich

stamatakis@bio.ifi.lmu.de
http://icwww.epfl.ch/~stamatak

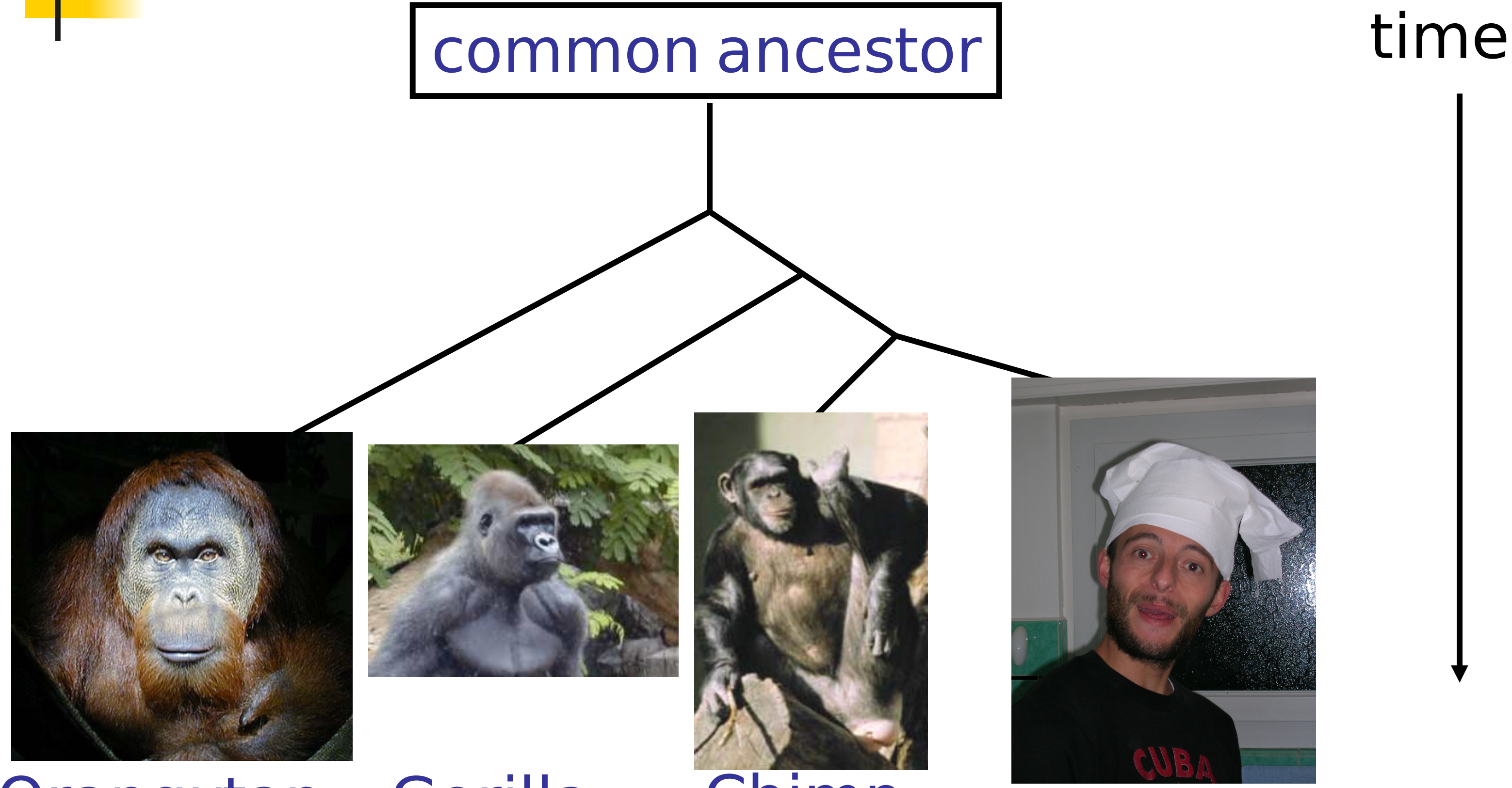
DNA Sequences

Orangutan	AACGTTTT
Gorilla	AAGGTTT
Chimp	AGGTTTT
Homo Sapiens	AGGATTTTT

DNA Alignment

Orangutan	A	A	C	G	T	T	T	T	-
Gorilla	A	A	G	G	T	T	T	-	-
Chimp	A	-	G	G	T	T	T	T	-
Homo Sapiens	A	G	G	A	T	T	T	T	T

Phylogeny of great Apes



common ancestor

time

Orangutan

Gorilla

Chimp

Homo Sapiens

Remember !

- Input need not be DNA or protein sequence data → gene order data
  - Moret et al (2001) GRAPPA: a high performance computational tool for phylogeny reconstruction from gene-order data
- Model need not be a tree → networks
  - Gusfield et al (2003) Efficient reconstruction of phylogenetic networks with constrained recombination
- Output need not be a strictly bifurcating tree → multifurcating tree

Remember !

- Input need n t be DNA or protein sequence d → gene order data
  - M erformance c r .
- Mode works
  - C truction of p ned r
- Output need not be a strictly bifurcating tree → multifurcating tree

We focus on computation of strictly bifurcating phylogenetic trees with maximum likelihood for DNA and Protein sequence data!

Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Web Servers
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

Phylogenetics

- Input: “good” multiple Alignment
- Output: unrooted binary tree
- Various methods for phylogenetic inference
  - Neighbor Joining (fast & simple)
  - Maximum Parsimony (relatively fast & simple)
  - Maximum Likelihood (complex & slow)
  - Bayesian Methods (complex & slower)

Phylogenetics

- Input: “good” multiple Alignment
- Output: un
- Various me c inference
  - Neighbor Joini (fast & simple)
  - Maximum Par imony (relatively fast & simple)
  - Maximum Likelihood (complex & slow)
  - Bayesian Methods (complex & slower)

ML & Bayesian: explicit model choice

Phylogenetics

- Input: “goo
- Output: unr
- Various me inference
  - Neighbor J
  - Maximum Pars y (relatively fast & simple)
  - Maximum Likelihood (complex & slow)
  - Bayesian Methods (complex & slower)

Complex Methods & Models required to reconstruct large & complicated trees !

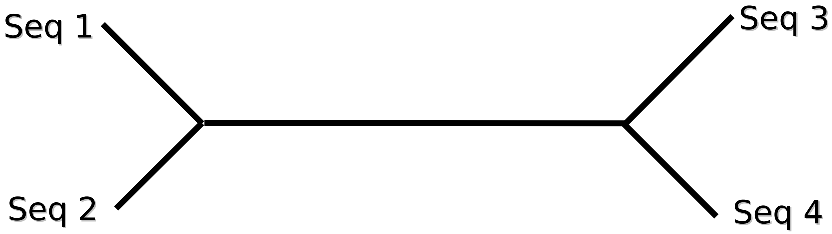
Focus of this lecture is on Maximum Likelihood!

Motivation

- Phylogenies to obtain insights in medical and biological research:
  - Epidemiology
  - Virology
  - Conservation Biology
  - Cancer, e.g., Papillomavirus phylogenies
  - Classification of unidentified sequences

Use Case: Rapid Phylogenetic Classification of unidentified Sequences

query sequence(s)

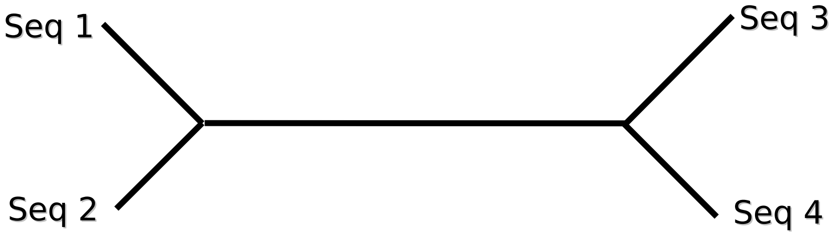


reference tree

Use Case: Rapid Phylogenetic Classification of uni

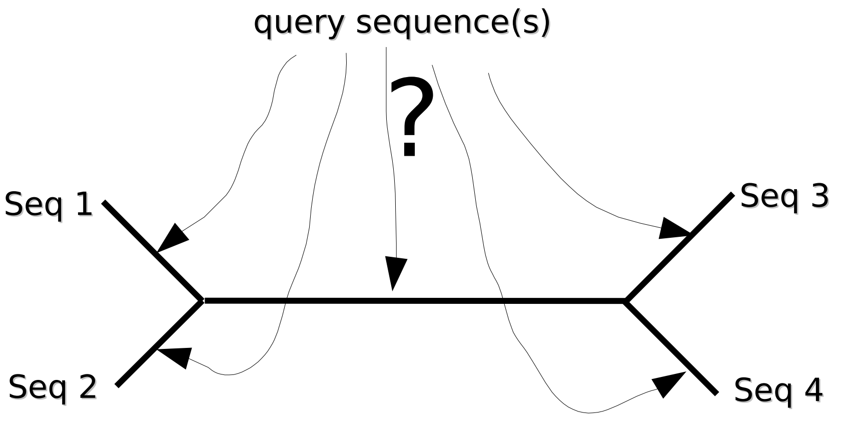
- Sequences in AC filters
- Microbial communities
- Birdstrike victims

query sequence(s)



reference tree

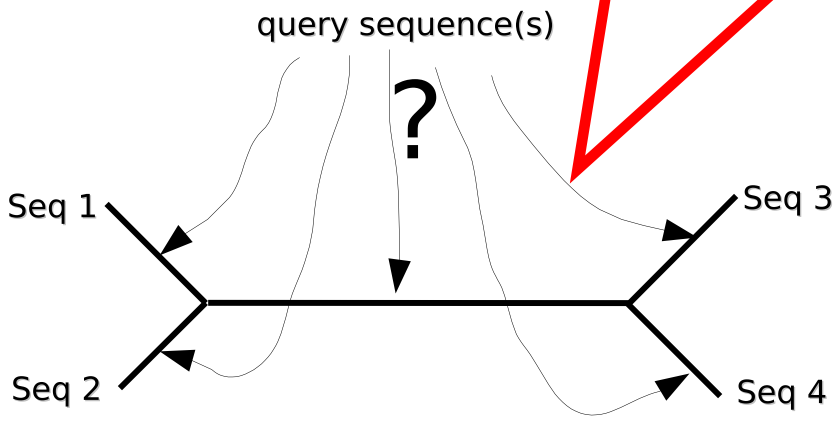
Use Case: Rapid Phylogenetic Classification of unidentified Sequences



reference tree

Use Case: Rapid Phylogenetic Classification of unidentified Sequences

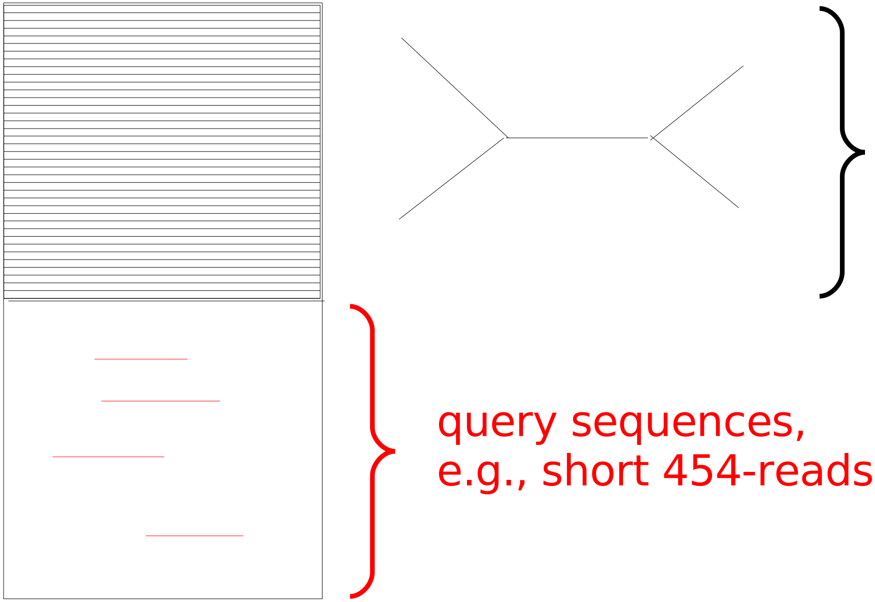
Can we compute an assignment to certain regions of the reference tree with some measure of support?



reference tree

Phylogenetic Classification: Input Data

sequence alignment

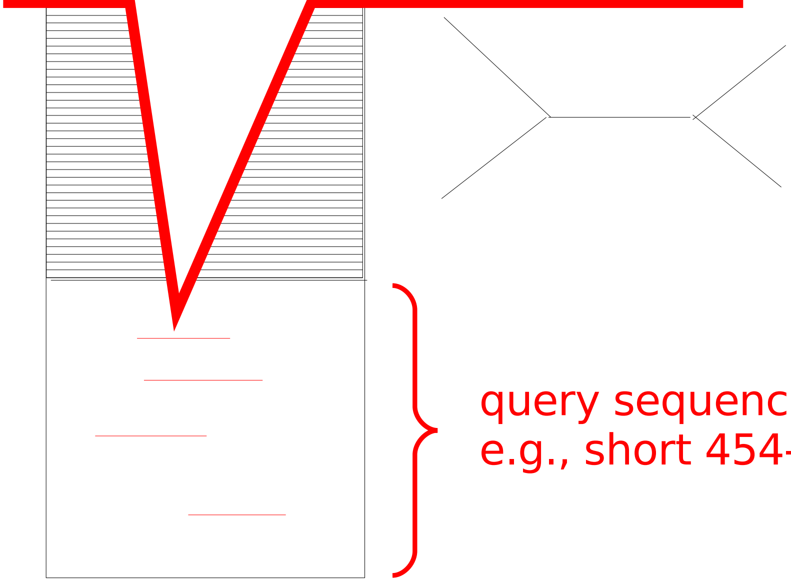


reference sequences & reference tree

query sequences,
e.g., short 454-reads (approx 250bp)

Phylogenetic Classification: Input Data

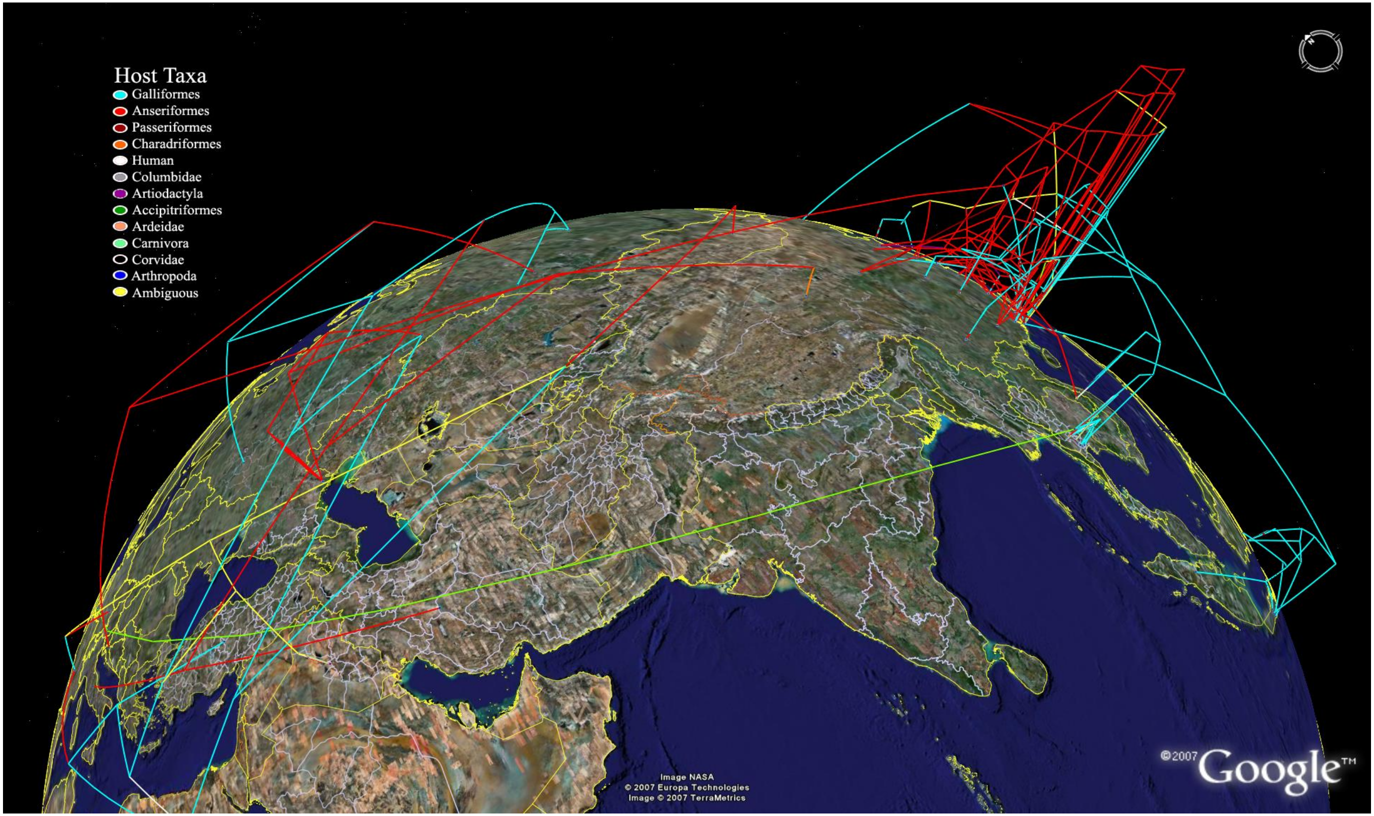
How do we align query sequences to the reference alignment?



reference sequences & reference tree

query sequences,
e.g., short 454-reads (approx 250bp)

Spread of Avian Influenza 1996-2004

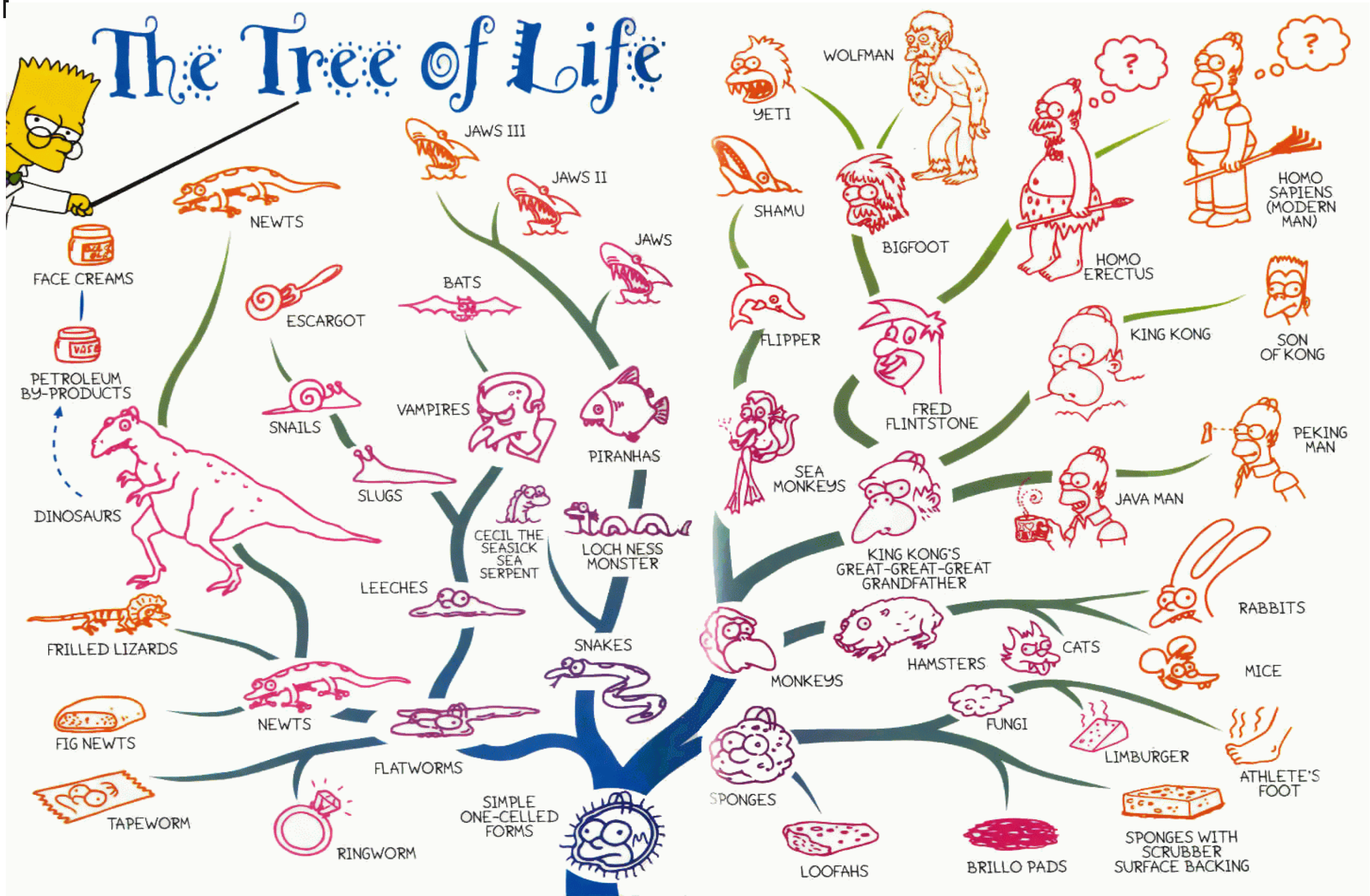


Courtesy of Dan Janies, Ohio State

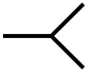
Challenges for Phyloinformatics

- Holy grail: “Tree of Life”
- Amount of available data grows at a higher rate than algorithms are getting faster
- Large multi-gene phylogenies
- Port codes to multi-core architectures
- What is a “good alignment” in a phylogenetic context?
- How do we assess confidence for our results?

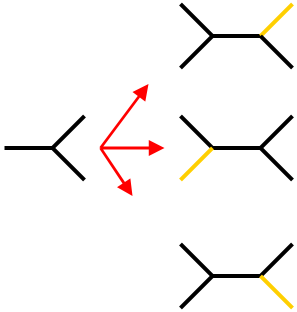
The Tree of Life



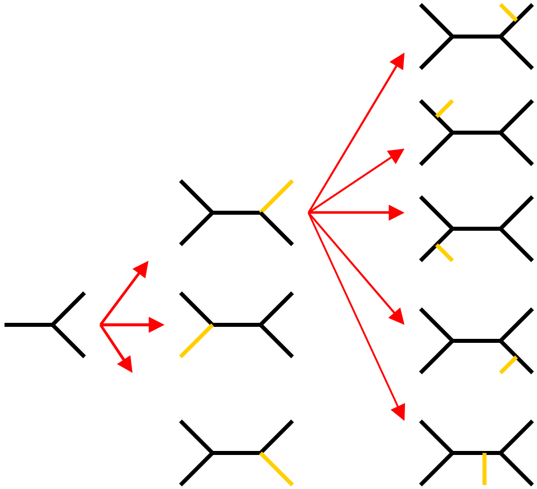
The number of trees



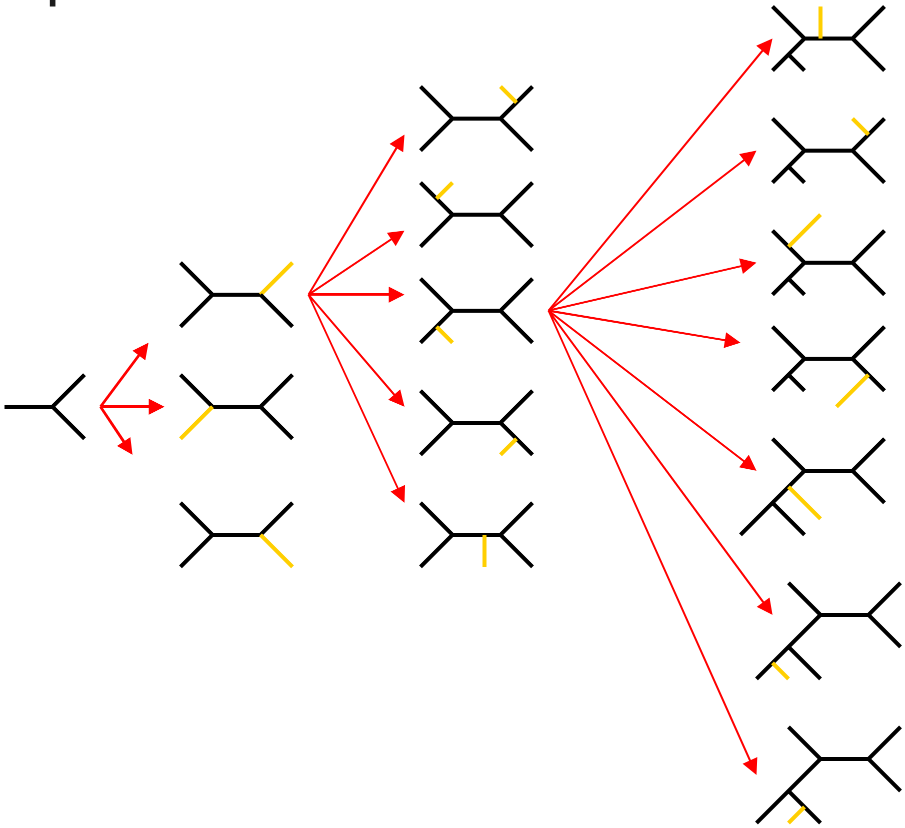
The number of trees



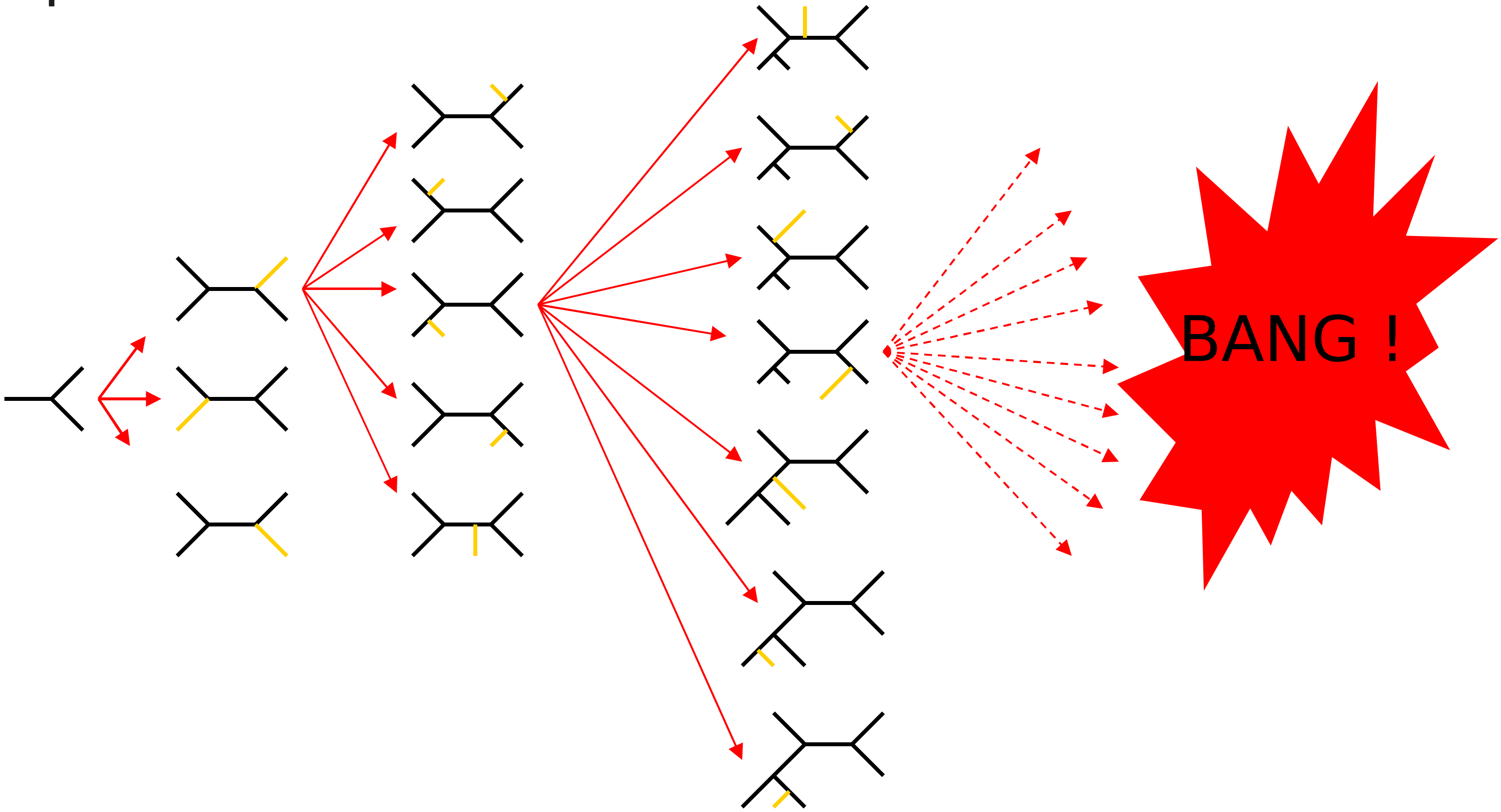
The number of trees



The number of trees



The number of trees explodes!



The Algorithmic Problem

- Number of potential trees grows exponentially

# Taxa	# Trees
5	15
10	2.027.025
15	7.905.853.580.625
50	2.84 * 10^76

The Algorithmic Problem

- Number of potential trees grows exponentially

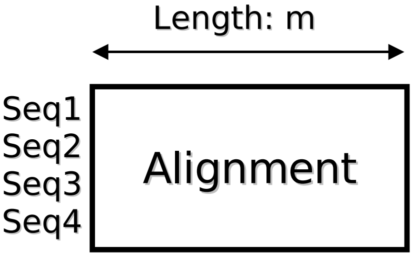
# Taxa	# Trees
5	15
10	2.027.025
15	7.905.853.580.6
50	2.84 * 10^76

This is ≈ the number of atoms in the universe 10^80

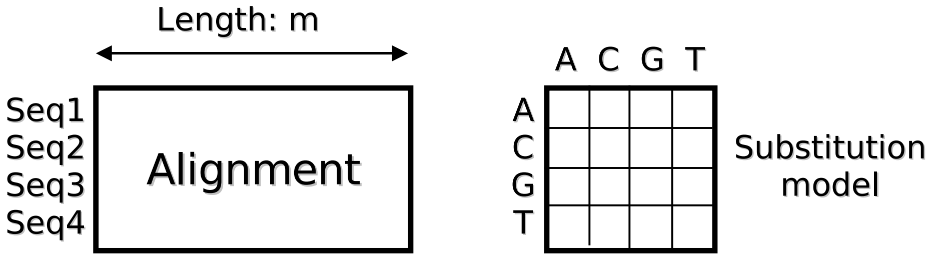
Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Web Servers
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

Maximum Likelihood



Maximum Likelihood



Length: m

Seq1
Seq2
Seq3
Seq4

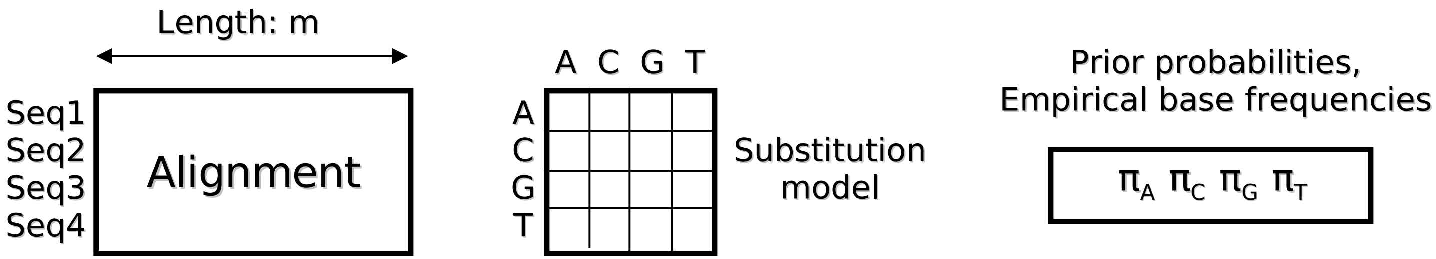
Alignment

A C G T

A
C
G
T

Substitution model

Maximum Likelihood



Length: m

Seq1
Seq2
Seq3
Seq4

Alignment

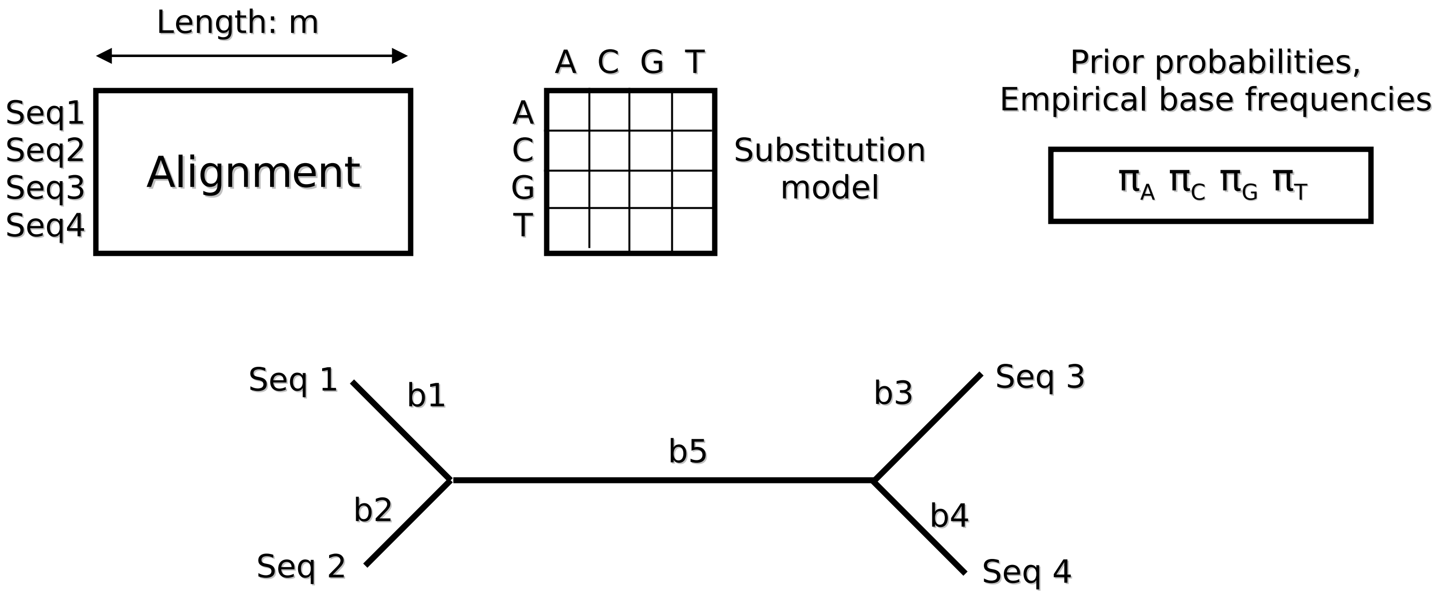
	A	C	G	T
A				
C				
G				
T				

Substitution model

Prior probabilities,
Empirical base frequencies

π_A π_C π_G π_T

Maximum Likelihood



Length: m

Seq1
Seq2
Seq3
Seq4

Alignment

A C G T
A
C
G
T

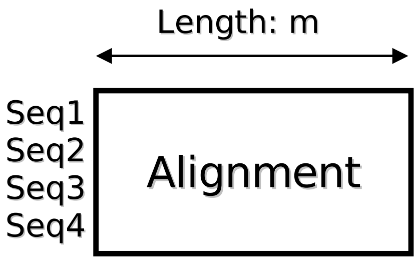
Substitution model

Prior probabilities,
Empirical base frequencies

π_A π_C π_G π_T

Seq 1
b1
Seq 2
b2
b5
b3
Seq 3
b4
Seq 4

Maximum Likelihood

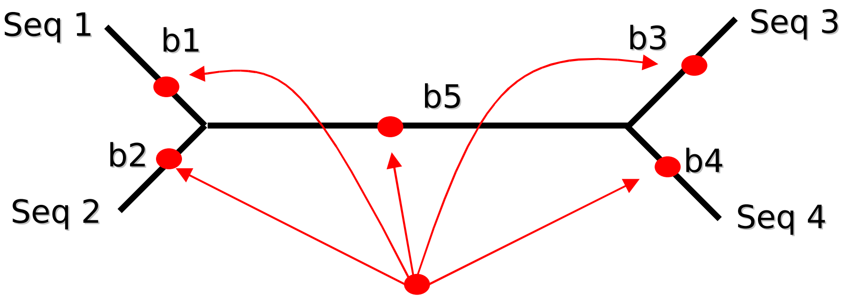


	A	C	G	T
A				
C				
G				
T				

Substitution model

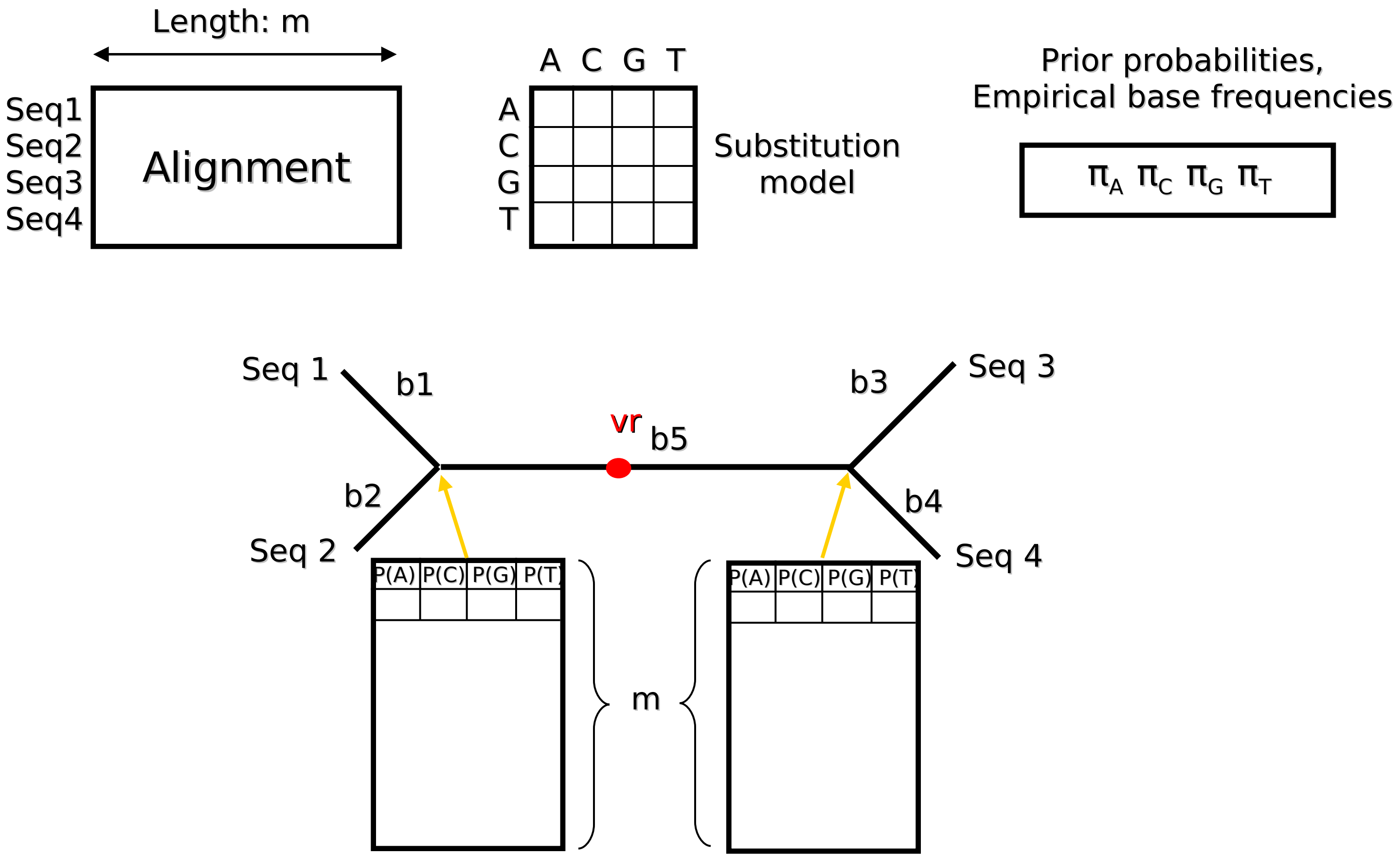
Prior probabilities, Empirical base frequencies

$\pi_A \; \pi_C \; \pi_G \; \pi_T$



virtual root: vr

Maximum Likelihood



Length: m

Seq1
Seq2
Seq3
Seq4

Alignment

	A	C	G	T
A				
C				
G				
T				

Substitution model

Prior probabilities,
Empirical base frequencies

π_A π_C π_G π_T

Seq 1

b1

Seq 2

b2

vr

b5

b3

Seq 3

b4

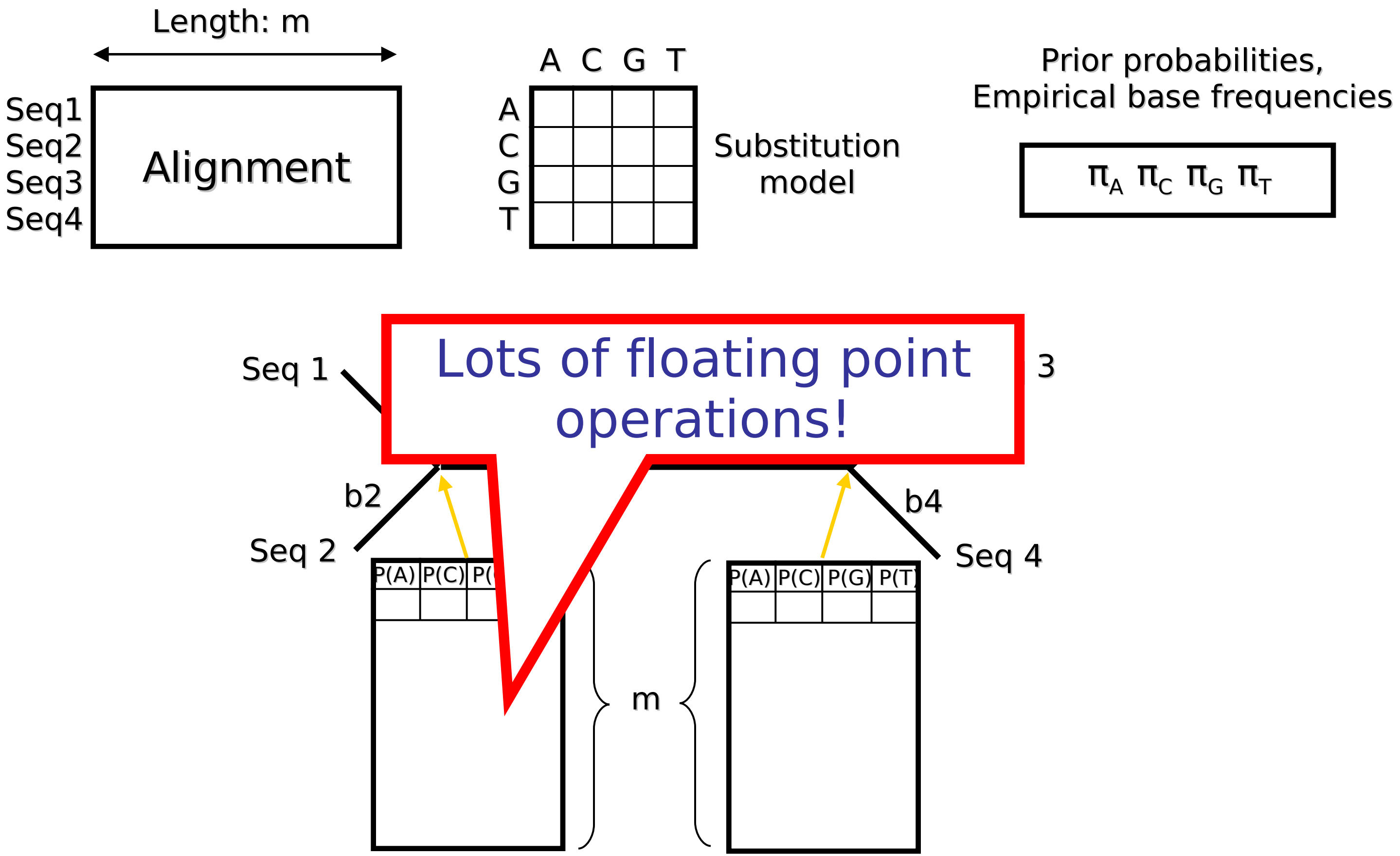
Seq 4

P(A)	P(C)	P(G)	P(T)

m

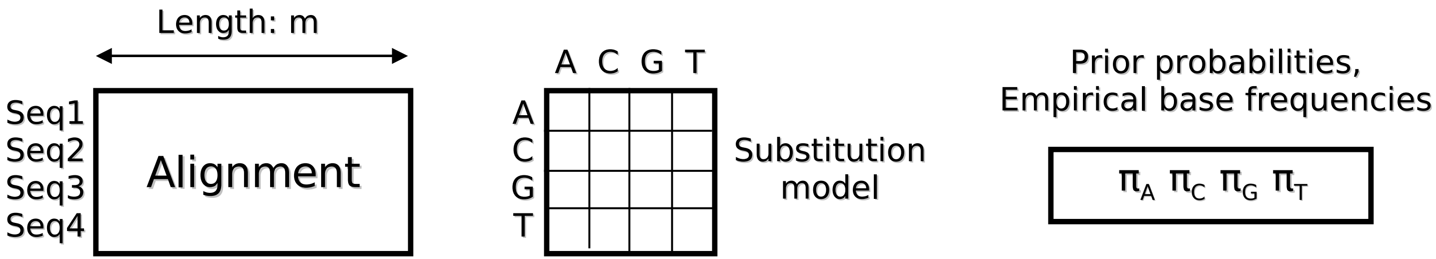
P(A)	P(C)	P(G)	P(T)

Maximum Likelihood



Lots of floating point operations!

Maximum Likelihood



Length: m

Seq1
Seq2
Seq3
Seq4

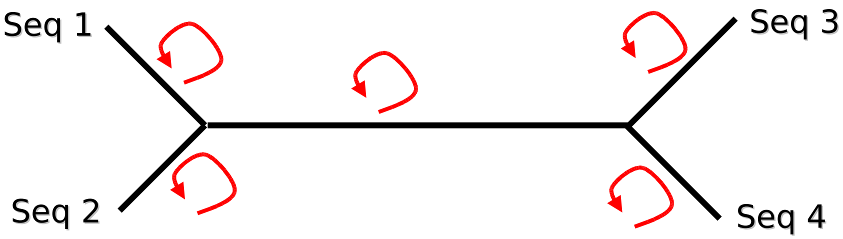
Alignment

	A	C	G	T
A				
C				
G				
T				

Substitution model

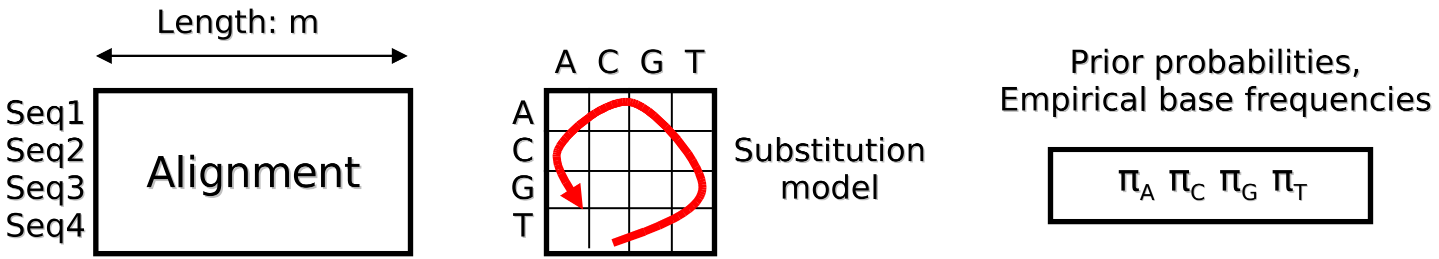
Prior probabilities,
Empirical base frequencies

$\pi_A \; \pi_C \; \pi_G \; \pi_T$



optimize branch lengths

Maximum Likelihood



Length: m

Seq1
Seq2
Seq3
Seq4

Alignment

A C G T

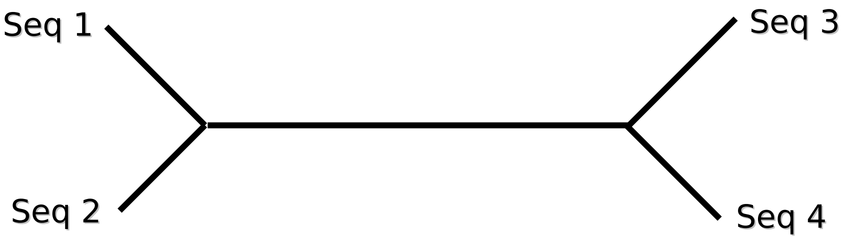
A
C
G
T

Substitution model

Prior probabilities, Empirical base frequencies

π_A π_C π_G π_T

optimize model parameters



Maximum Likelihood

Goal: Obtain topology with maximum likelihood value

Problem I: Number of possible topologies is exponential in n

Problem II: Computation of likelihood function is expensive

Problem III: Probably high score accuracy required

Problem IV: High memory consumption

Solution:

- New Algorithms
- New Models
- High Performance Computing

Maximum Likelihood

Goal: Obtain topology with maximum likelihood value

Problem I: Number of possible topologies is exponential in n

Problem II: Computation of likelihood fu

Problem III: Probably high score accura

Problem IV: High memory consumption

Solution:

- New Algorithms
- New Models
- High Performance Computing

Exemplary solutions:
RAxML
Randomized
Axelerated
Maximum Likelihood
Open-Source Code

Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Web Servers
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

RAxML Usage & Web Servers

- Since August 2006 approx. 3,000 downloads from distinct IPs
  - USA: 44%
  - Germany: 11%
  - 58 other countries < 5%
- RAxML Web-Servers using Rapid Bootstrap Algorithm
  - San Diego Supercomputing Center
    - Since December 2007 over 3,000 jobs
    - **http://phylobench.vital-it.ch/raxml-bb/**
  - Vital-IT unit of Swiss Institute of Bioinformatics
    - Since September 2007 over 8,000 jobs
    - **http://8ball.sdsc.edu:8889/cipres-web/Bootstrap.do**

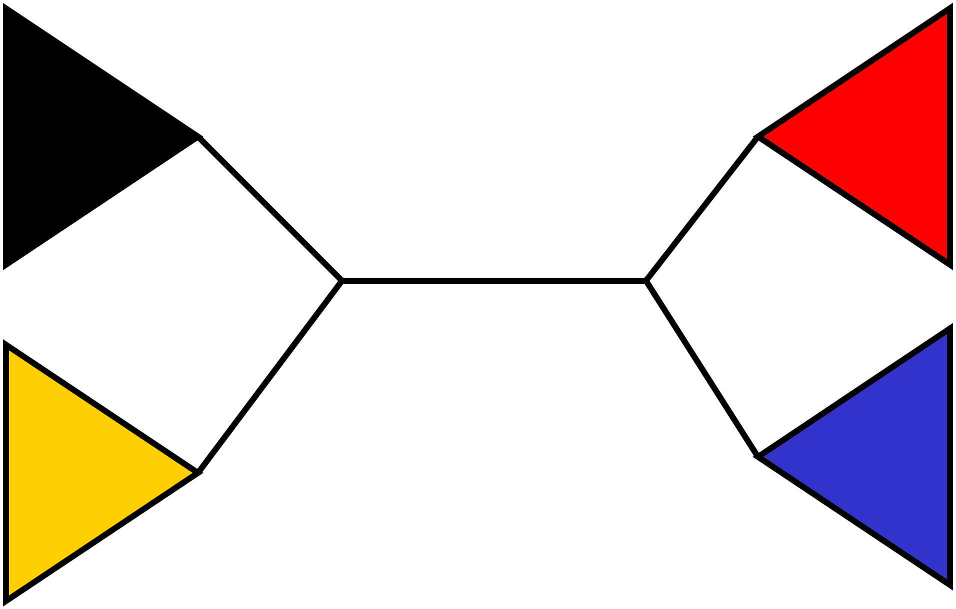
Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Web Servers
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

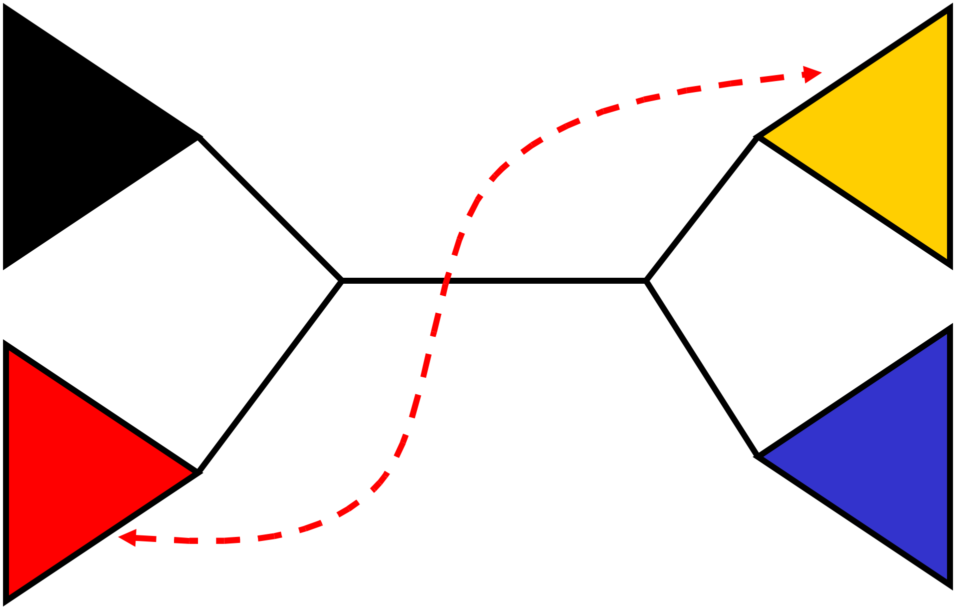
Basic Algorithm

- Compute comprehensive starting tree
  - Complete Random Starting Tree (MrBayes, Garli)
  - Neighbor Joining (IQPNNI, PHYML)
  - Maximum Parsimony (RAxML)
- Optimize tree by application of standard topological alterations
  - NNI: Nearest Neighbor Interchange
  - TBR: Tree Bisection Reconnection
  - SPR: Subtree Pruning Re-Grafting (Subtree Rearrangements)
- Search Algorithms
  - Hill-Climbing
  - Simulated Annealing
  - Genetic Algorithms
  - Metropolis-Coupled Markov-Chain Monte-Carlo (MC^3)

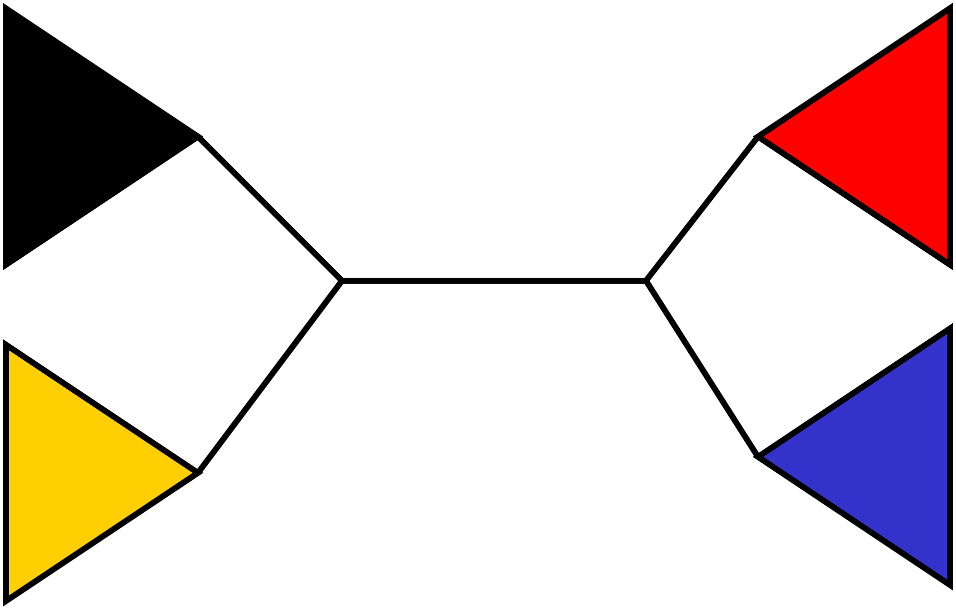
NNI



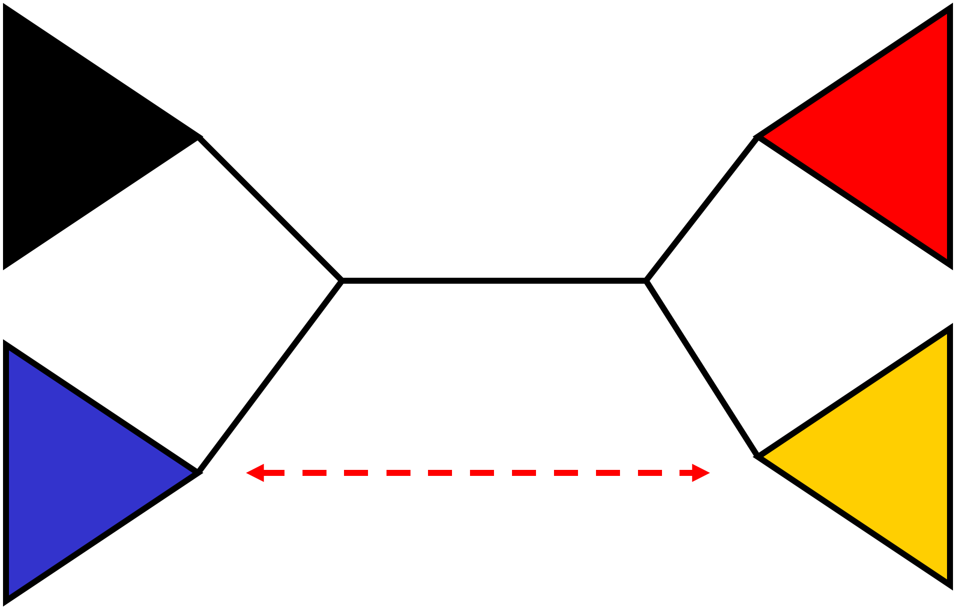
NNI



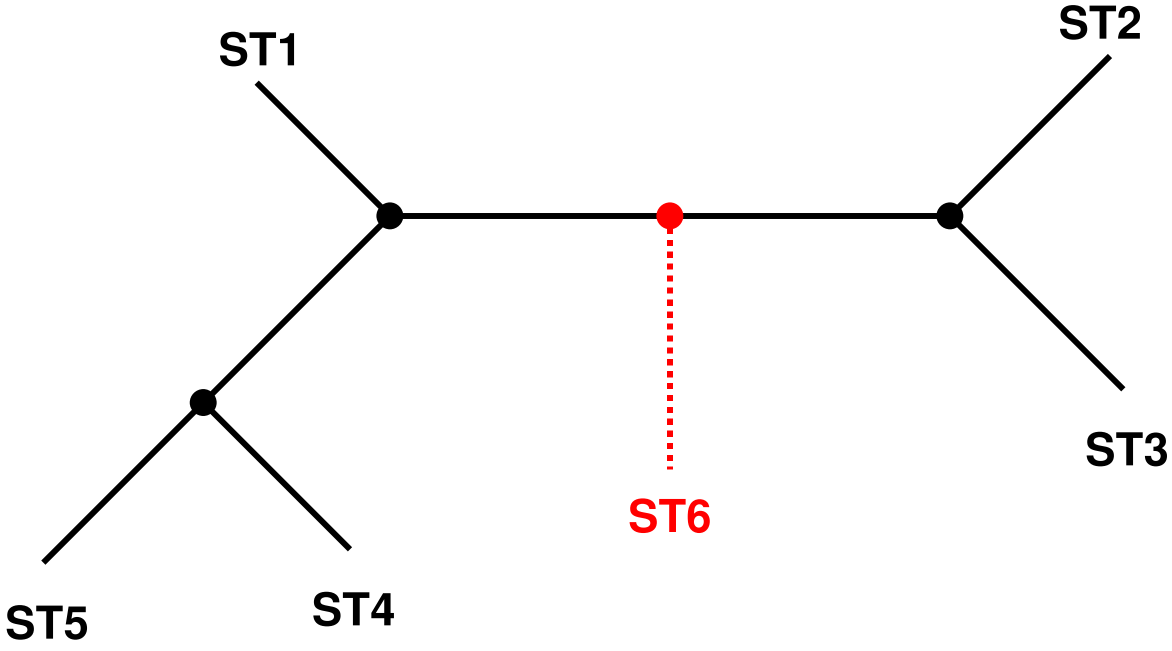
NNI



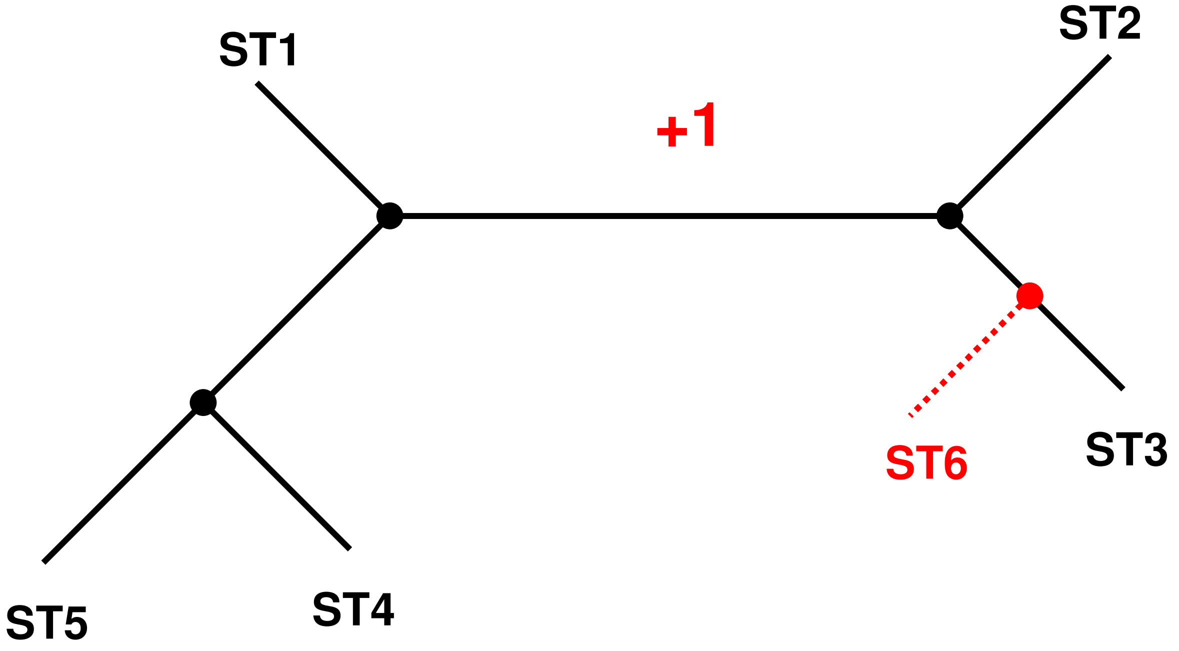
NNI



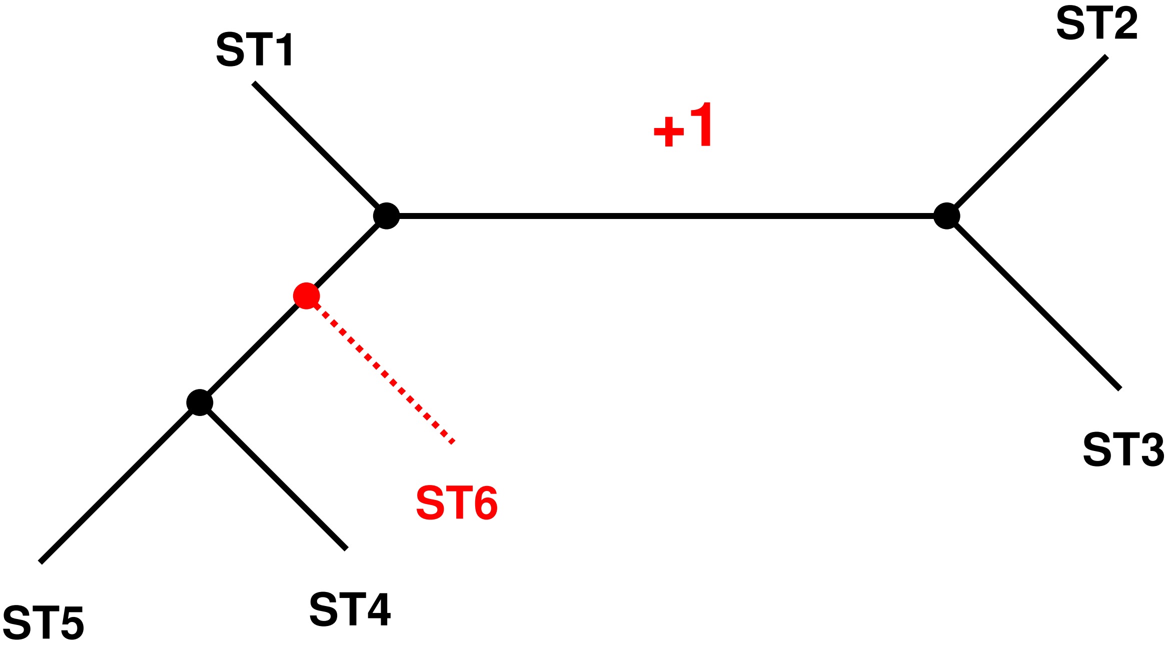
SPR



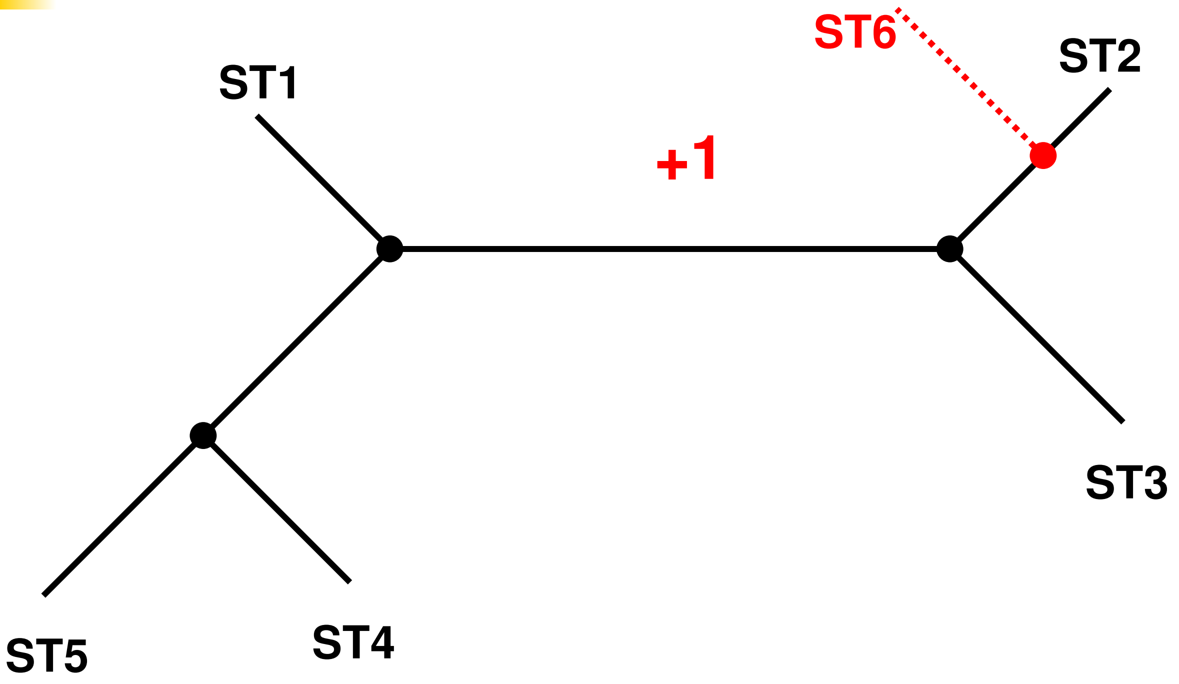
SPR



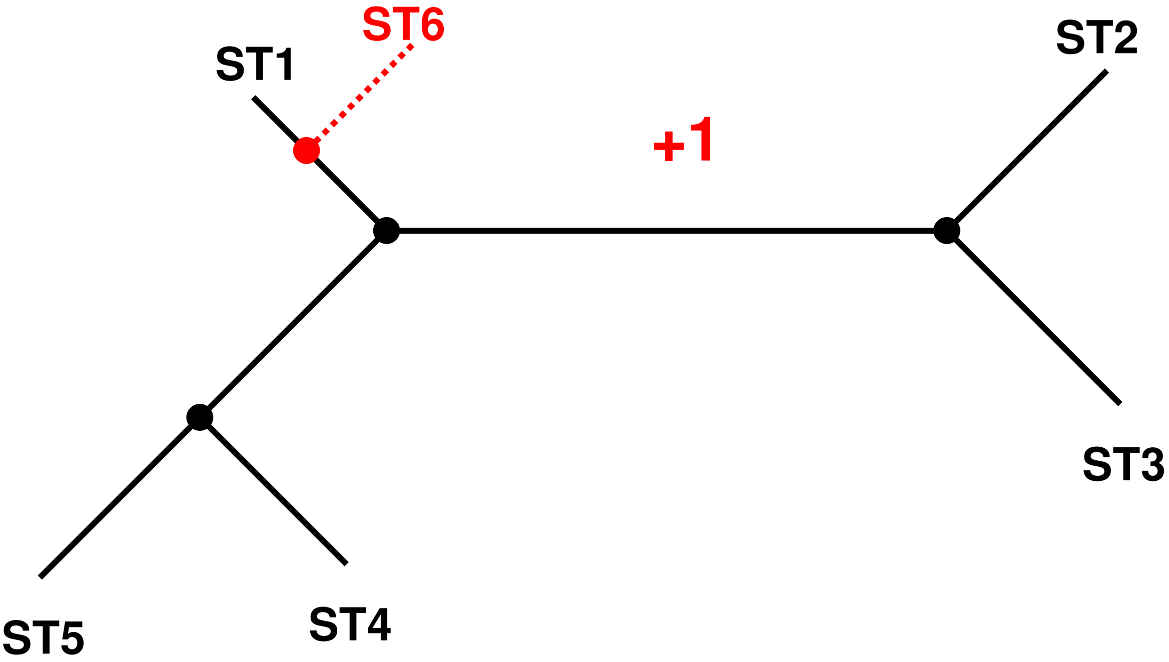
SPR



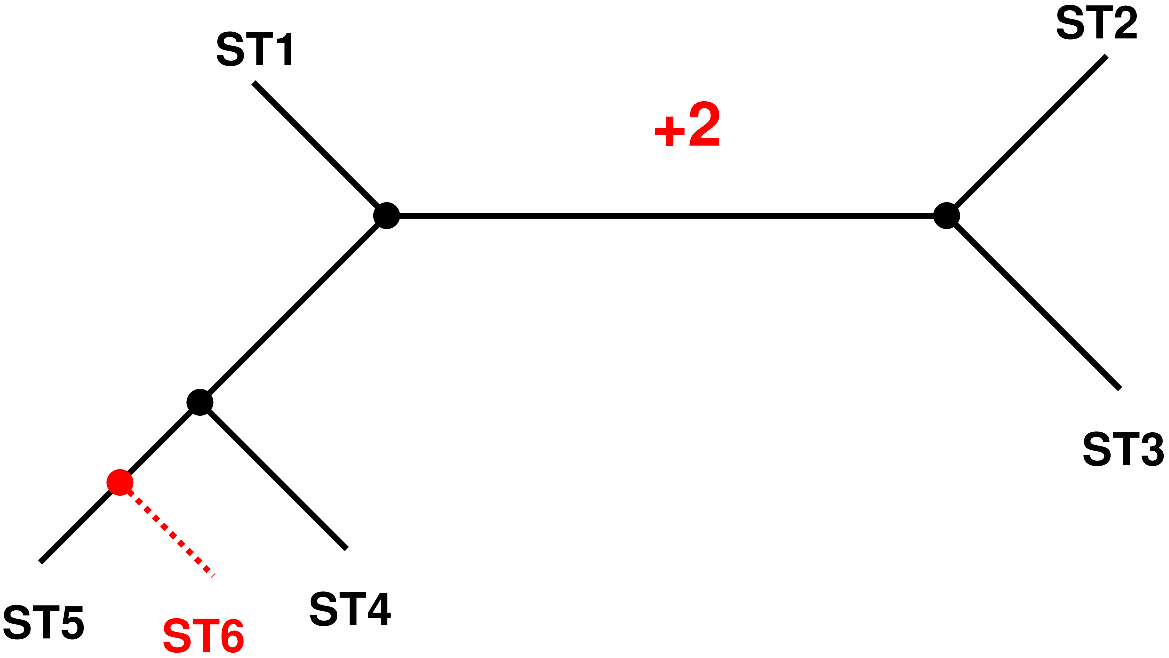
SPR



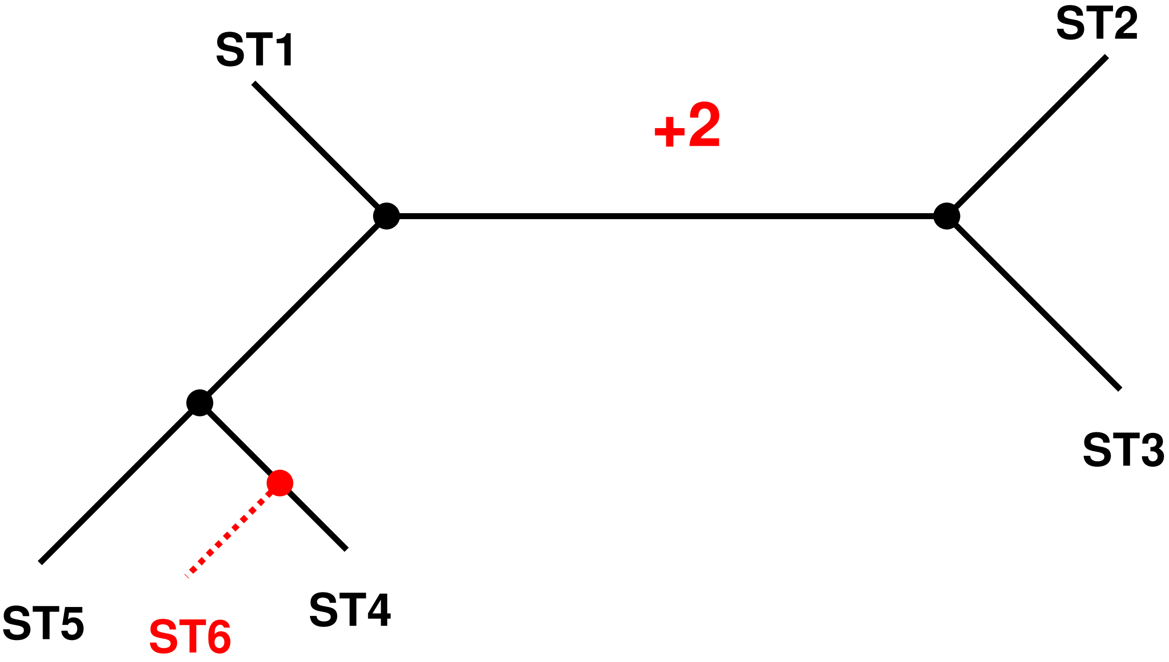
SPR



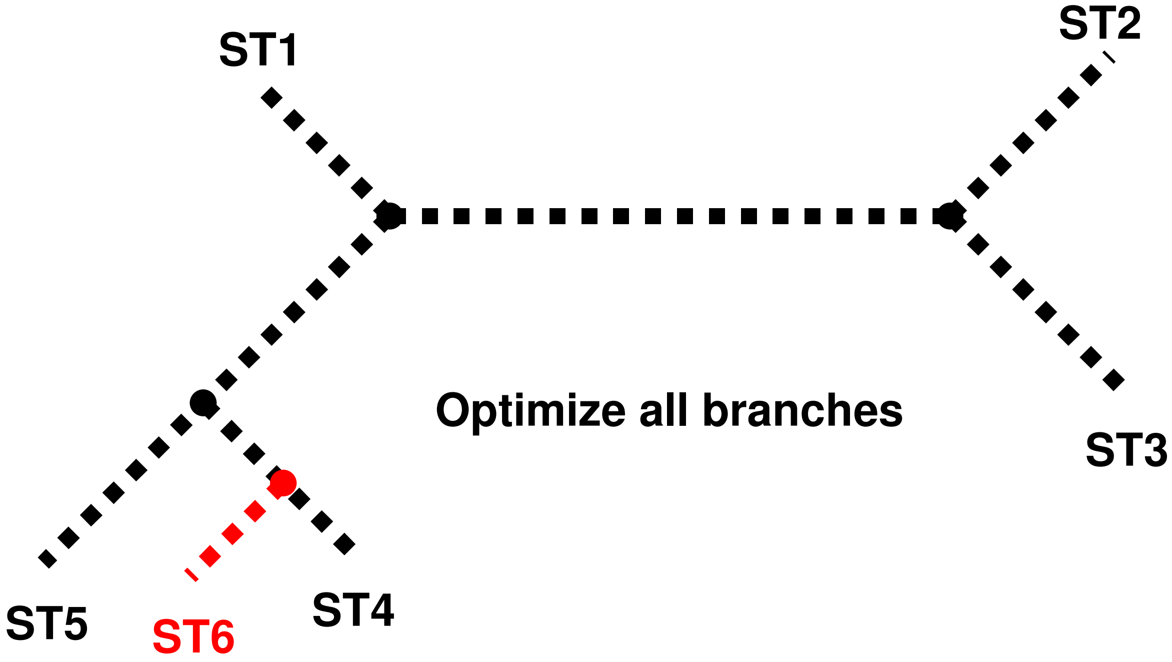
SPR



SPR

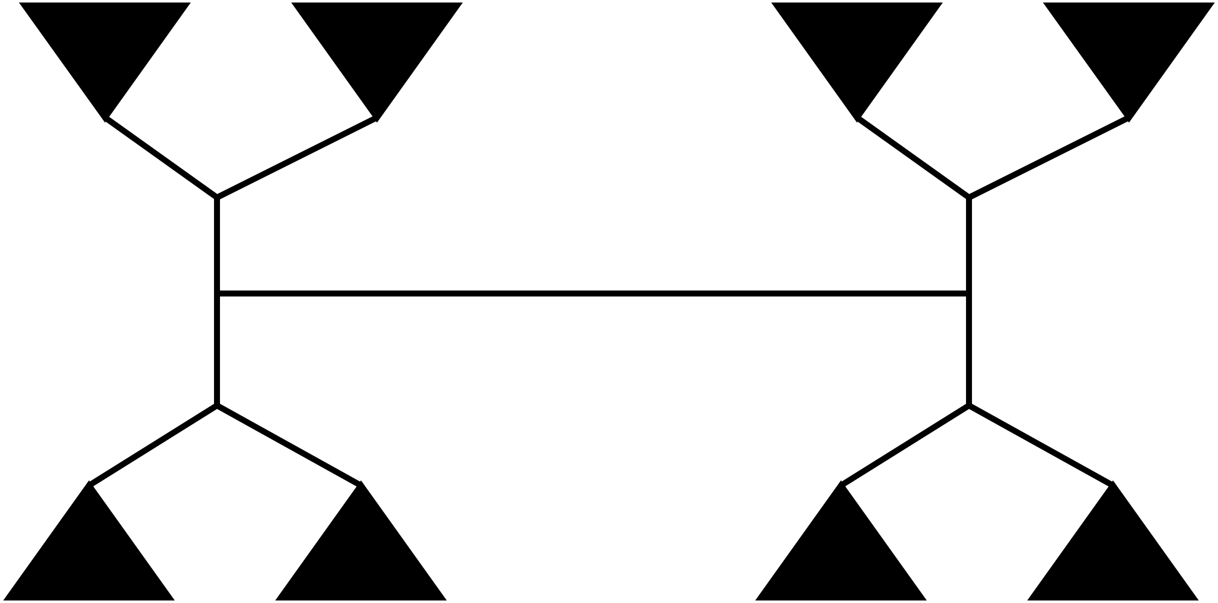


SPR

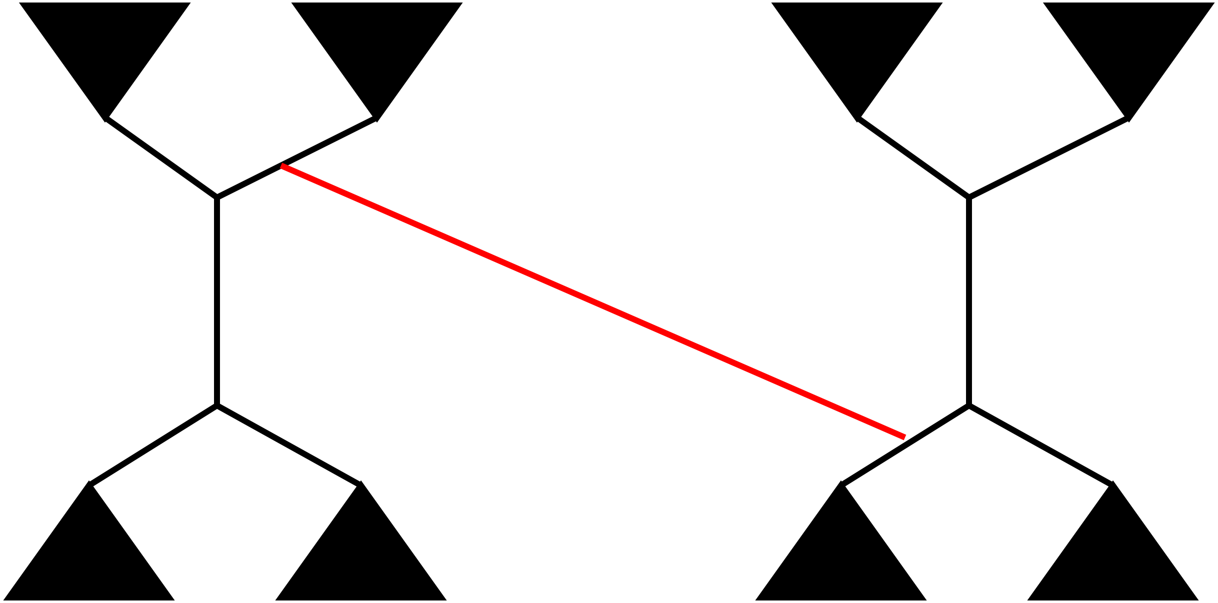


Optimize all branches

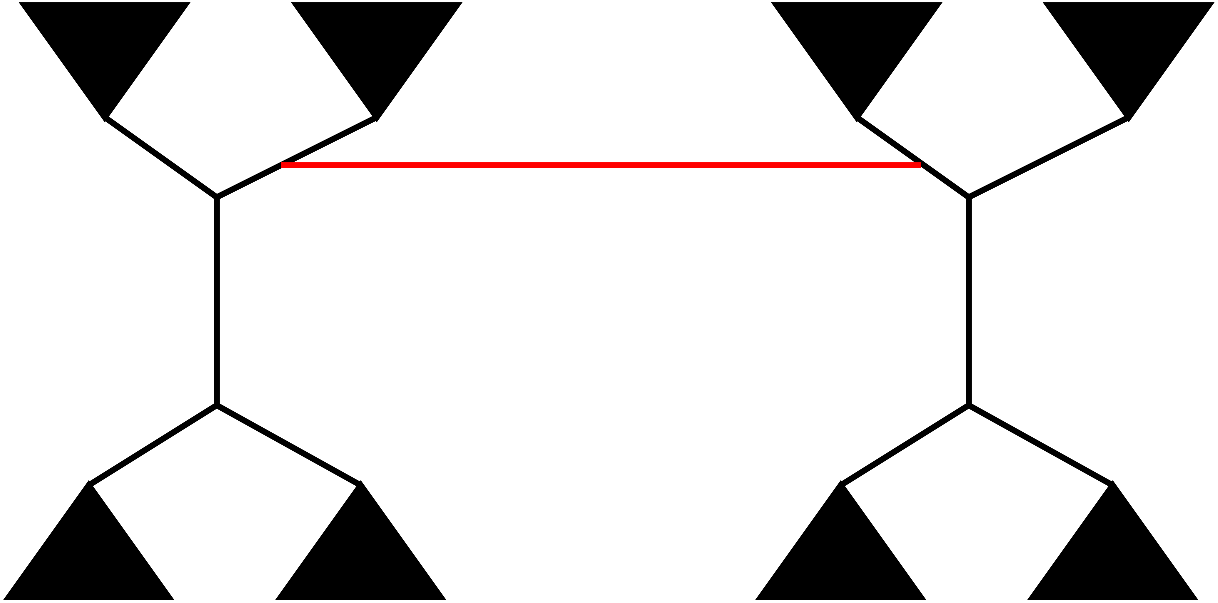
TBR



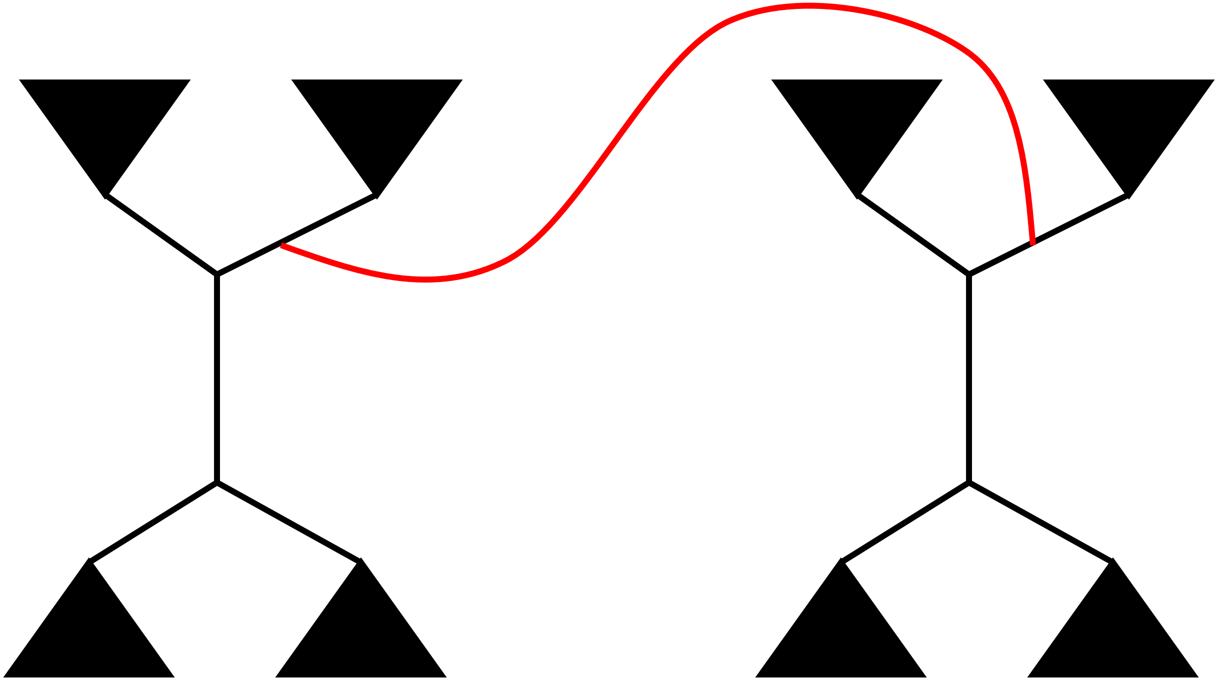
TBR



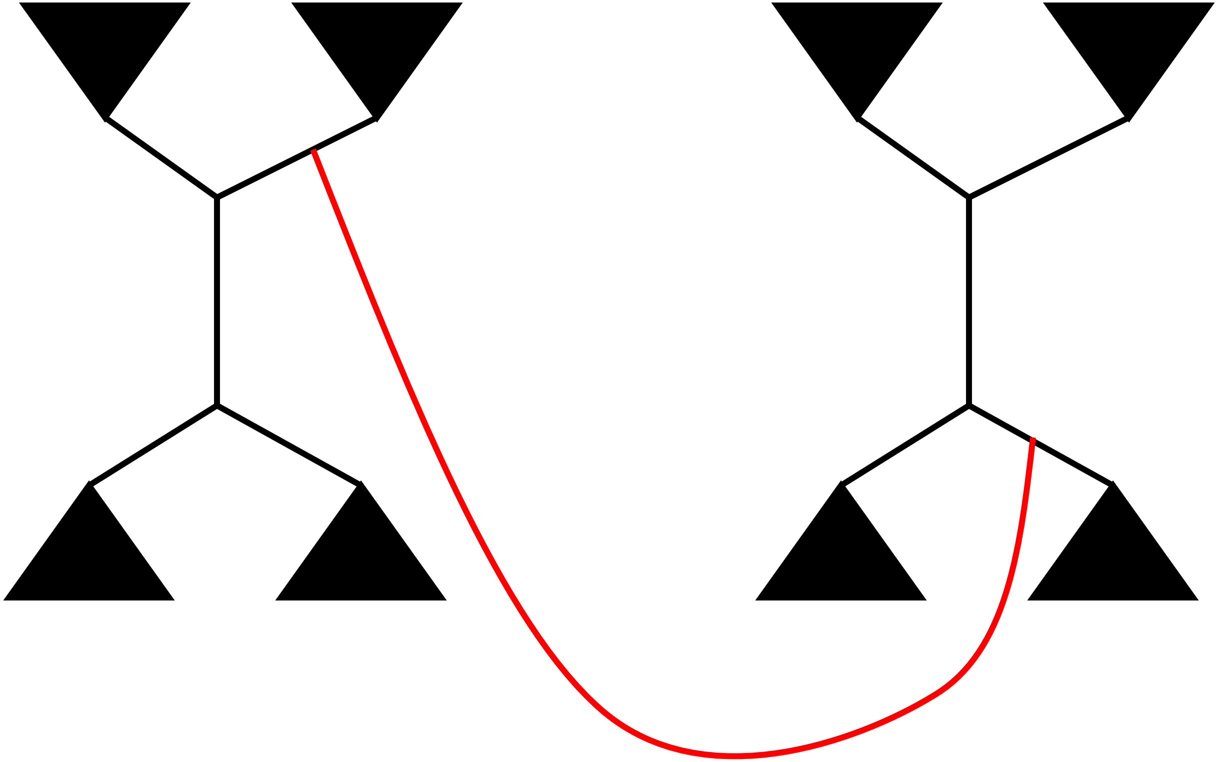
TBR



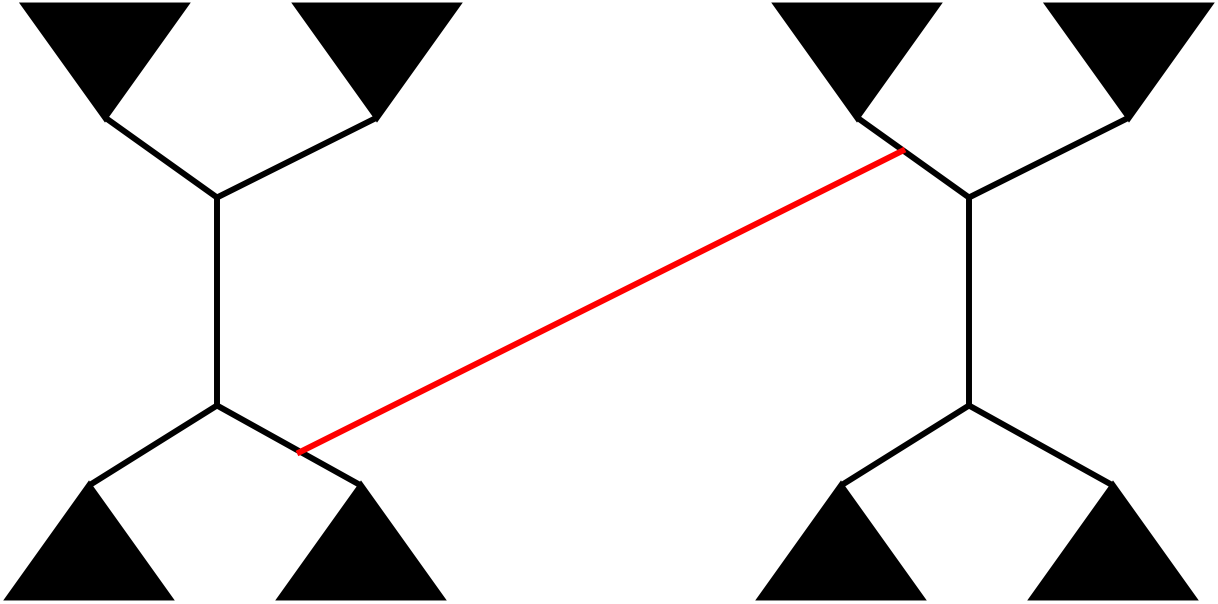
TBR



TBR



TBR



How does RAxML work?

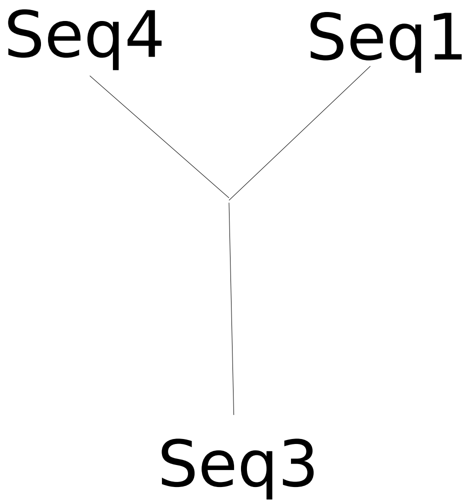
Compute randomized stepwise addition order
Maximum Parsimony tree

Stepwise Addition Order Algorithm

Seq0
Seq1
Seq2
Seq3
Seq4

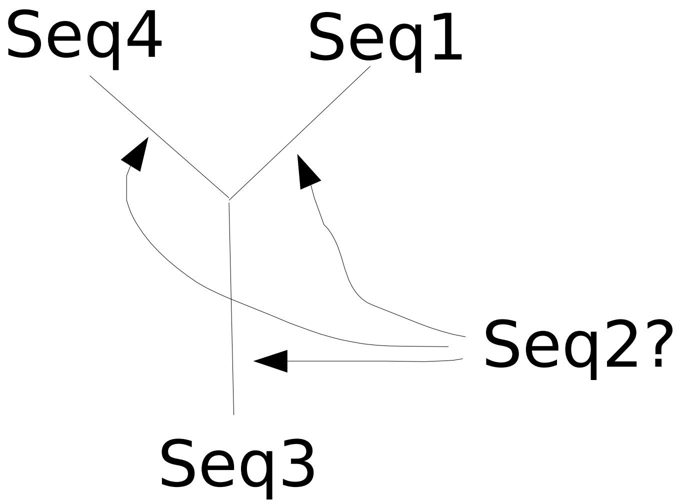
Stepwise Addition Order Algorithm

Seq0
~~Seq1~~
Seq2
~~Seq3~~
~~Seq4~~



Stepwise Addition Order Algorithm

Seq0
~~Seq1~~
~~Seq2~~
~~Seq3~~
~~Seq4~~



Stepwise Addition Order Algorithm

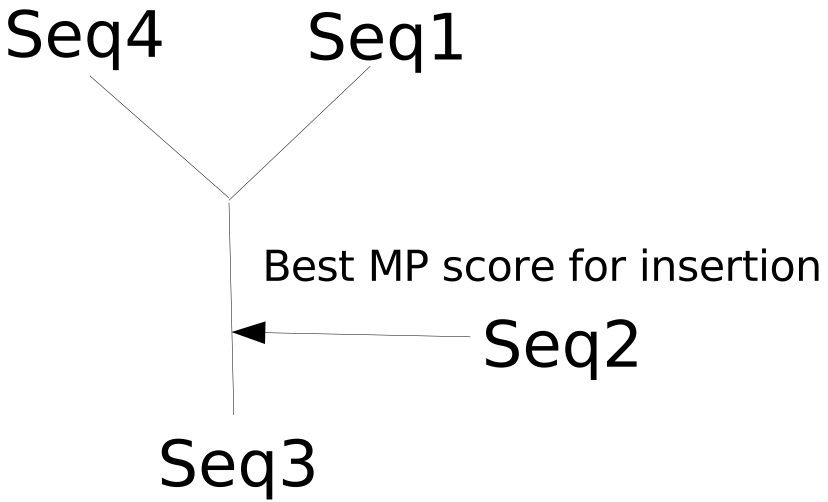
Seq0

~~Seq1~~

~~Seq2~~

~~Seq3~~

~~Seq4~~



Seq4

Seq1

Best MP score for insertion

Seq2

Seq3

Stepwise Addition Order Algorithm

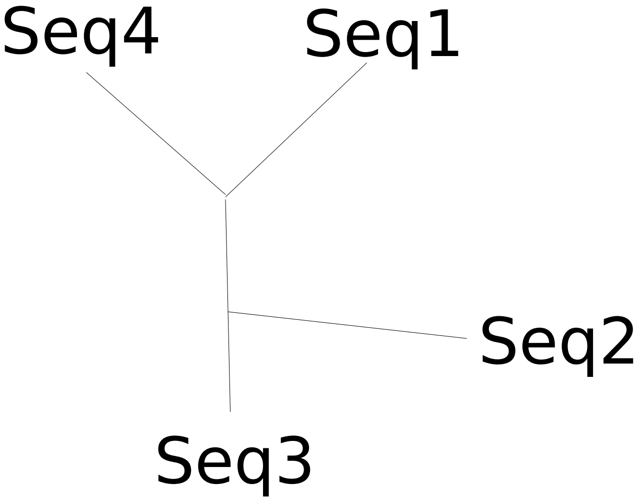
Seq0

~~Seq1~~

~~Seq2~~

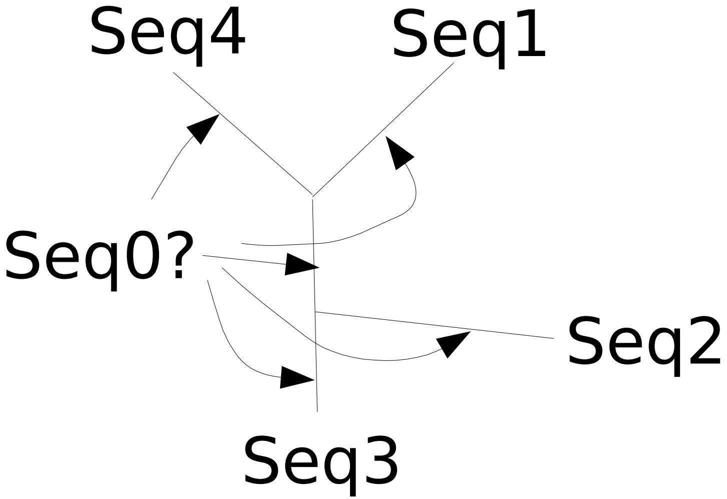
~~Seq3~~

~~Seq4~~



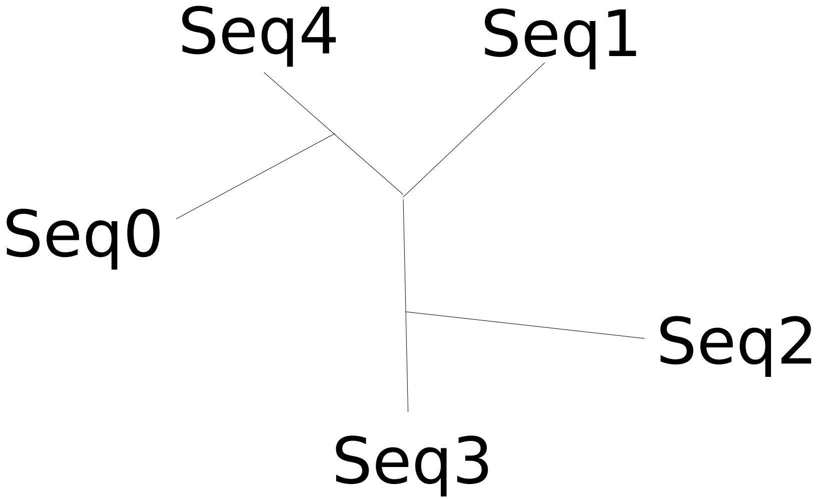
Stepwise Addition Order Algorithm

~~Seq0~~
~~Seq1~~
~~Seq2~~
~~Seq3~~
~~Seq4~~



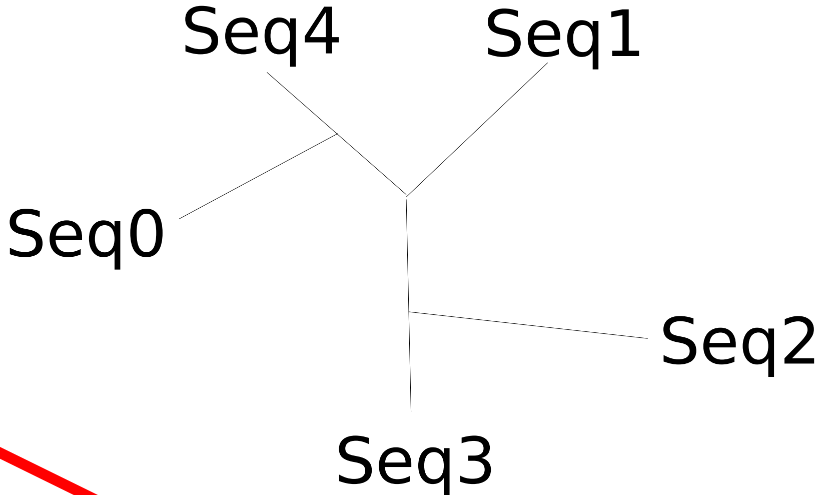
Stepwise Addition Order Algorithm

~~Seq0~~
~~Seq1~~
~~Seq2~~
~~Seq3~~
~~Seq4~~



Stepwise Addition Order Algorithm

~~Seq0~~
~~Seq1~~
~~Seq2~~
~~Seq3~~
~~Seq4~~



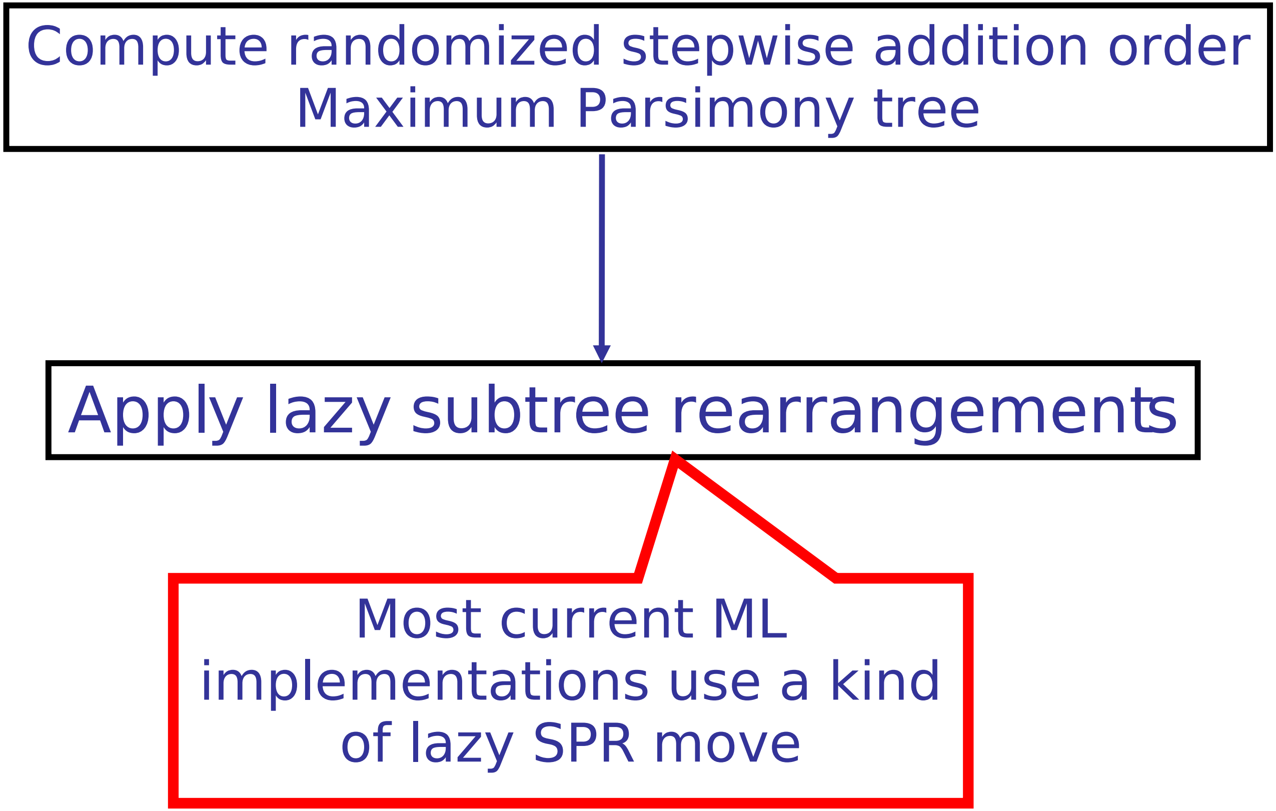
Distinct addition order, e.g., Seq0→Seq1→Seq2→Seq3→Seq4 can yield a different tree

How does it work?

Compute randomized stepwise addition order Maximum Parsimony tree

Advantage of RAxML: search starts from distinct points in search space every time

How does it work?

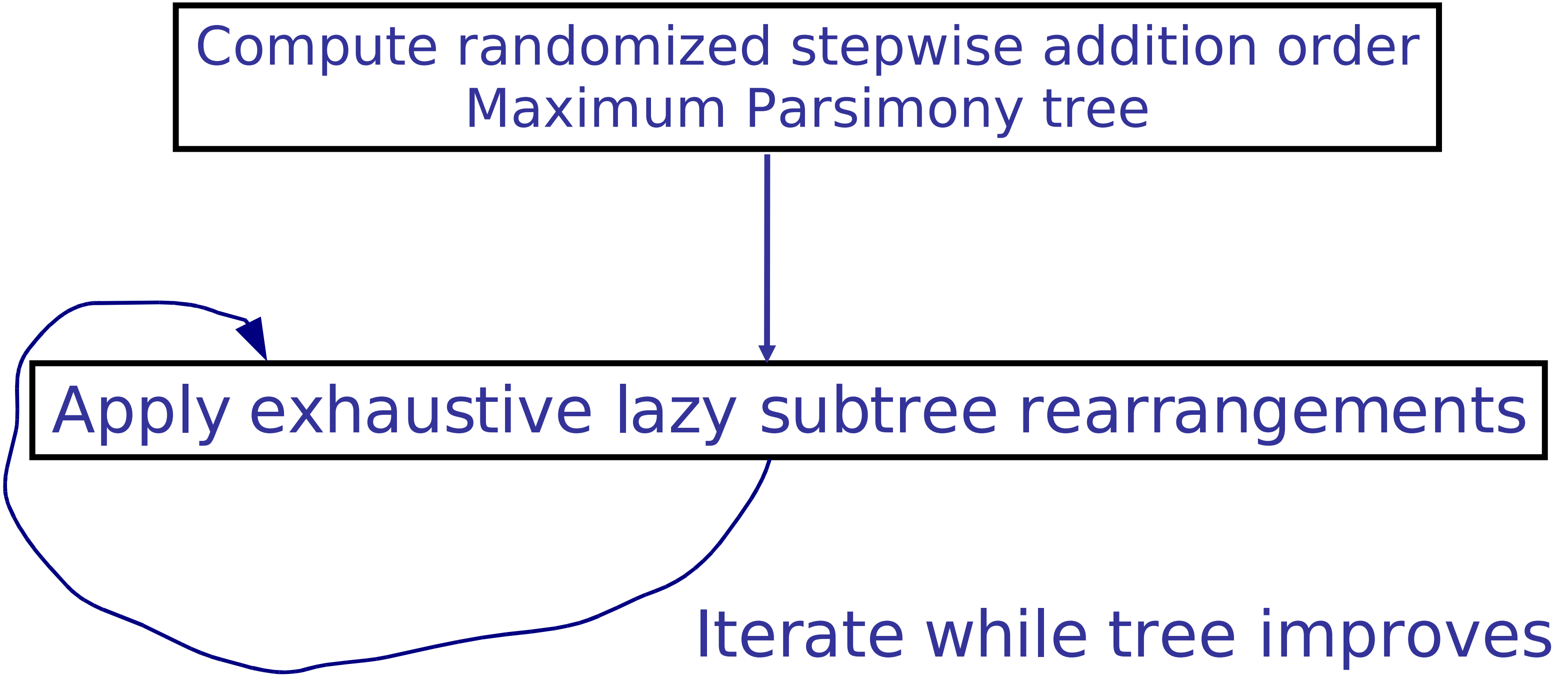


Compute randomized stepwise addition order Maximum Parsimony tree

Apply lazy subtree rearrangements

Most current ML implementations use a kind of lazy SPR move

How does it work?

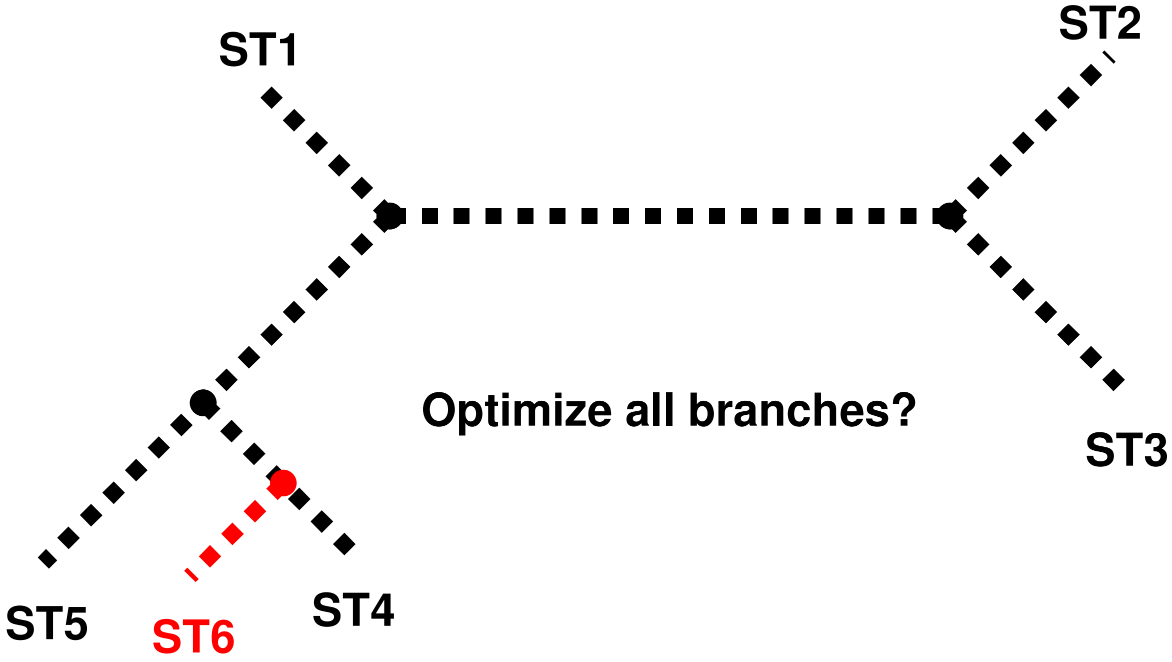


Compute randomized stepwise addition order Maximum Parsimony tree

Apply exhaustive lazy subtree rearrangements

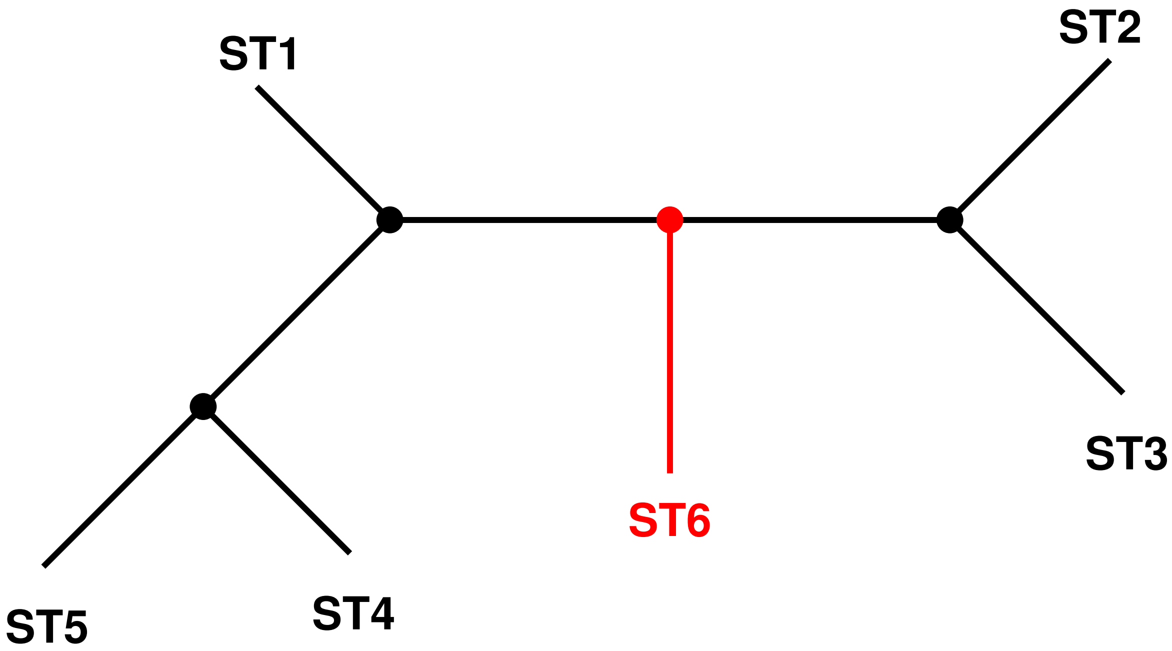
Iterate while tree improves

Subtree Rearrangements

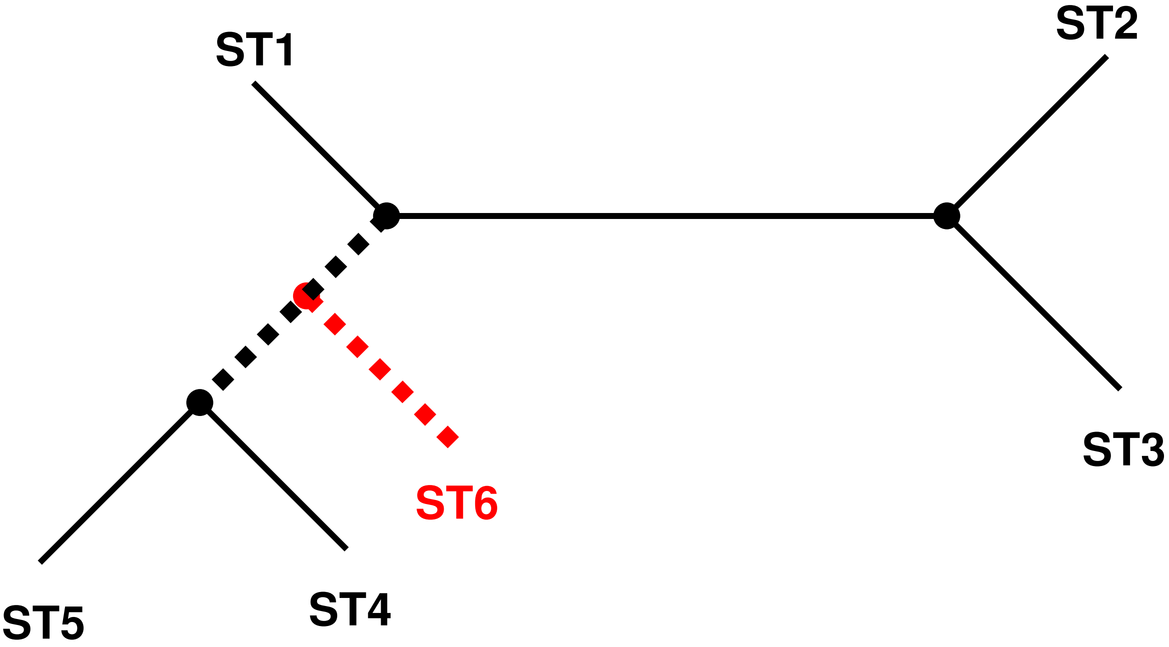


Optimize all branches?

Lazy Subtree Rearrangements



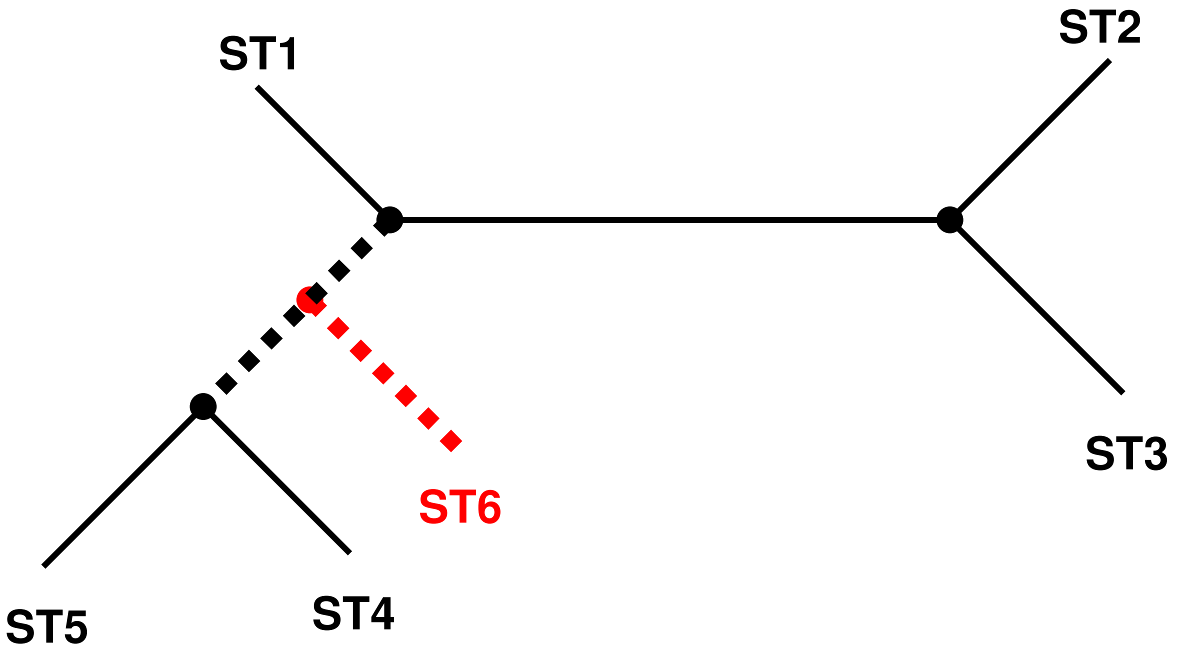
Lazy Subtree Rearrangements



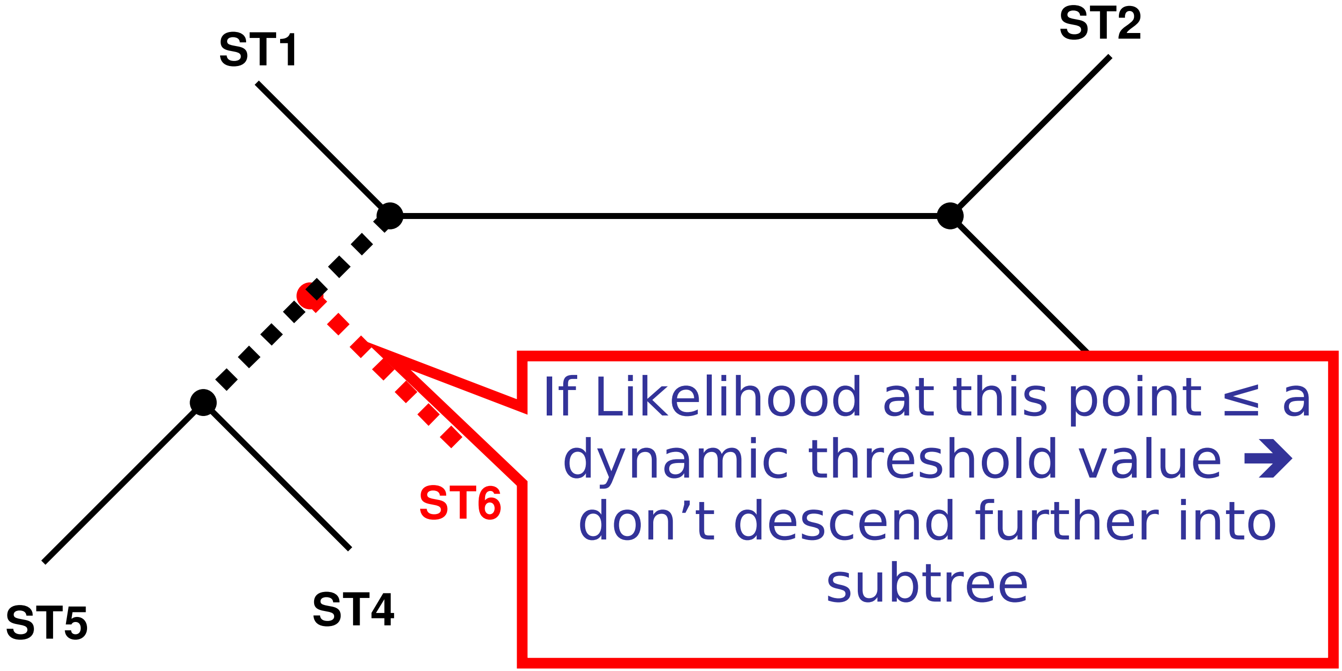
Why does this work?

- Lazy subtree rearrangements:
  - Update less likelihood vectors → significantly faster
  - Allows for higher rearrangement settings → better trees
- Likelihood depends strongly on topology
- Fast exploration of large number of topologies
- Fast pre-scoring of topologies
- Store best 20 trees from each rearrangement cycle
- Full ML optimization of best 20 trees only
- Experimental results justify this mechanism

Likelihood Cutoff

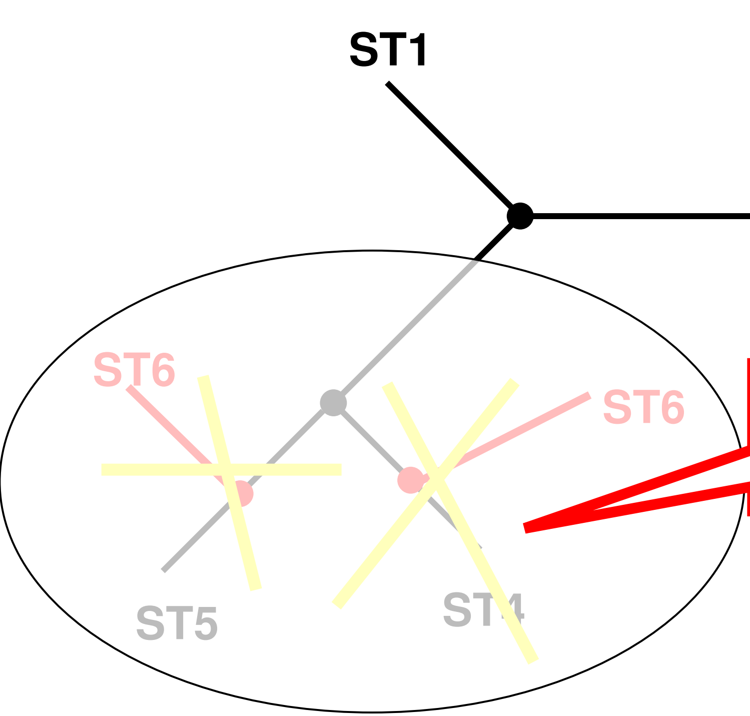


Likelihood Cutoff



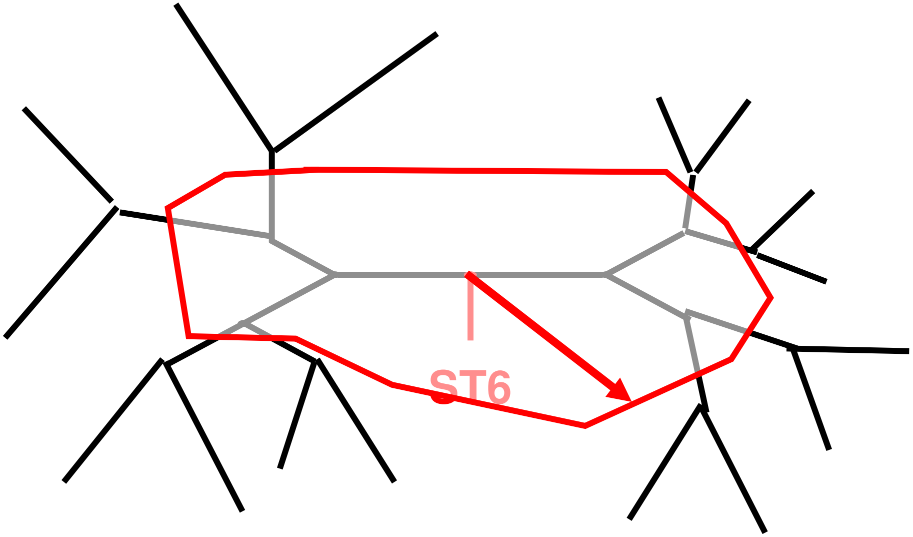
If Likelihood at this point ≤ a dynamic threshold value ➔ don’t descend further into subtree

Likelihood Cutoff

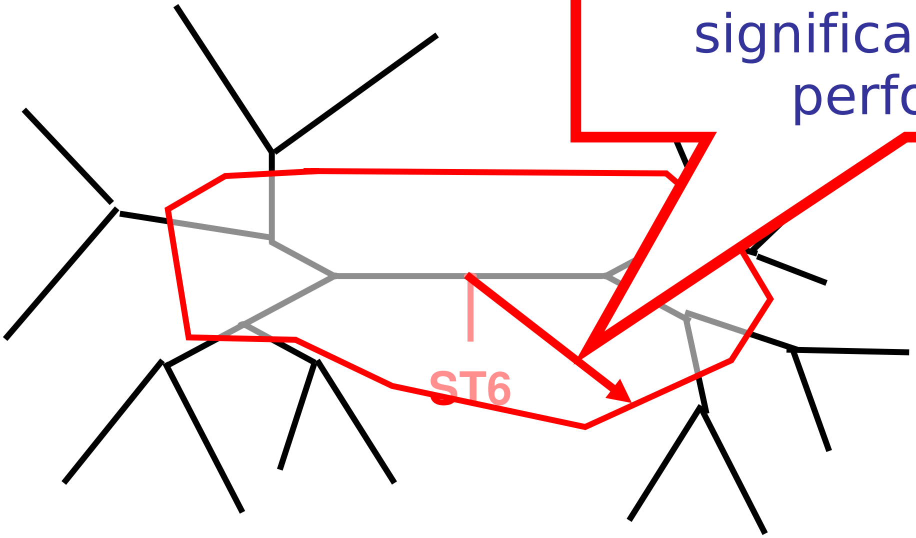


These LSRs are not executed!

The Rearrangement Setting



The Rearrangement Setting

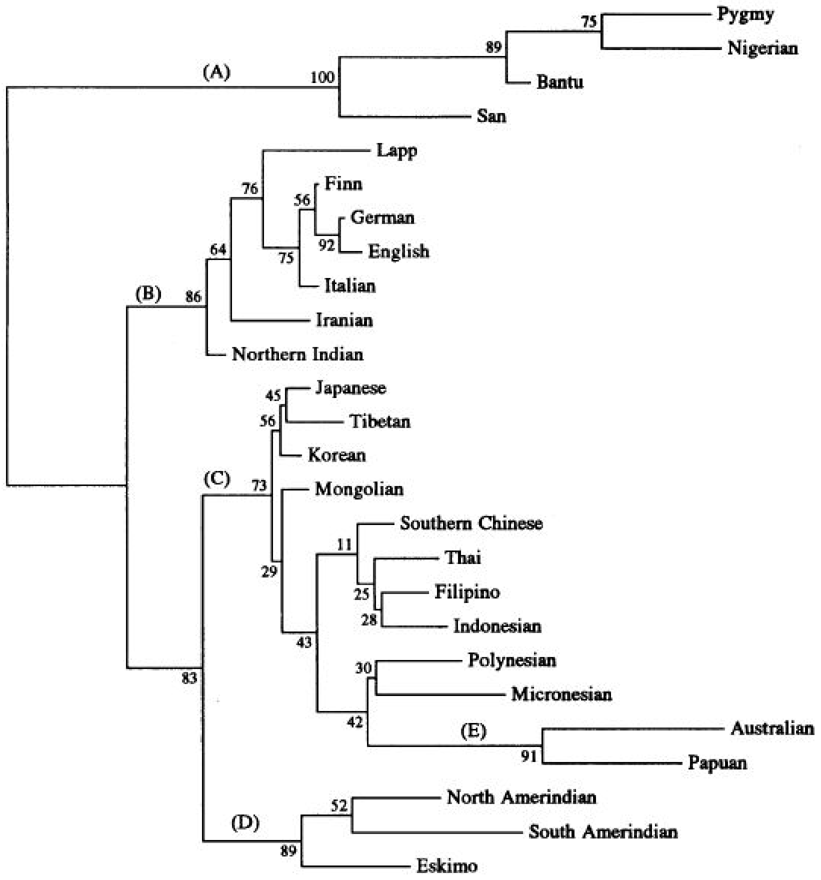


This setting has a significant impact on performance!

Confidence Values

- Tree without node confidence values is mostly useless
- Problem:
  - Confidence value calculation is major computational obstacle
  - ➔ We can compute large trees but not analyze them: compute ≠analyze !

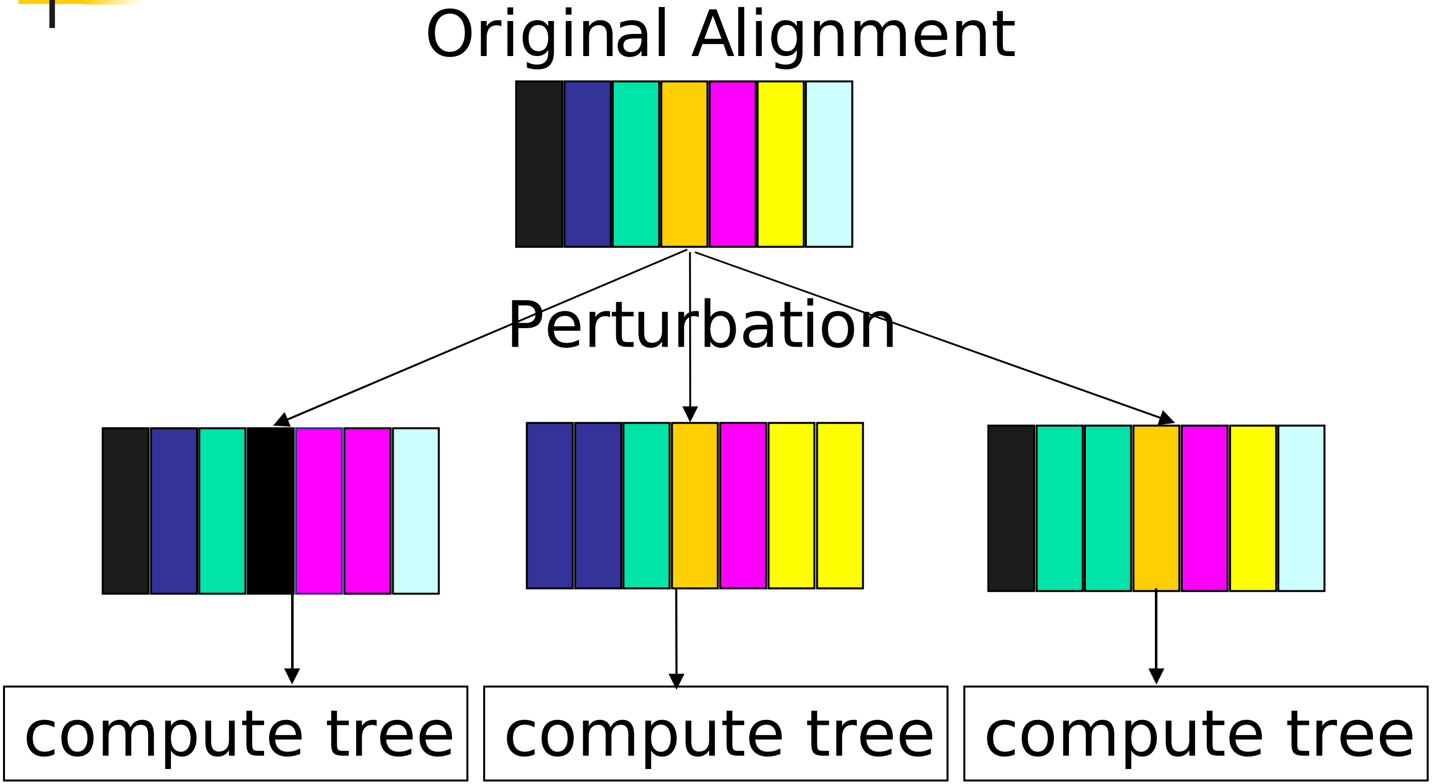
A Tree with Confidence Values



Confidence Values

- Tree without node confidence values is mostly useless
- Problem:
  - Confidence value calculation is major computational obstacle
  - ➔ We can compute large trees but not analyze them: compute ≠analyze !
- Current Slow Methods
  - Sampling with Bayesian methods
  - Non-parametric Bootstrapping

Bootstrapping



Original Alignment

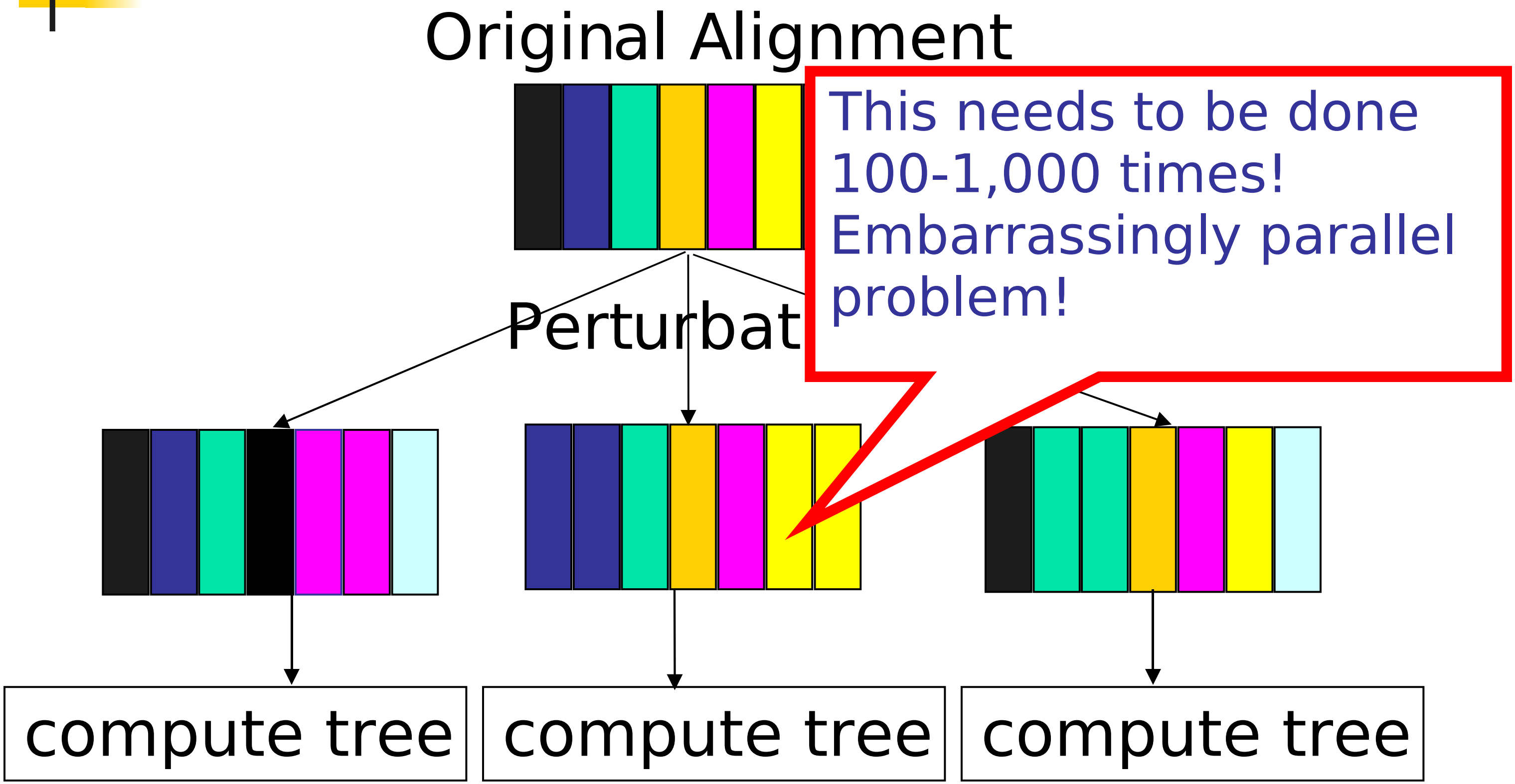
Perturbation

compute tree

compute tree

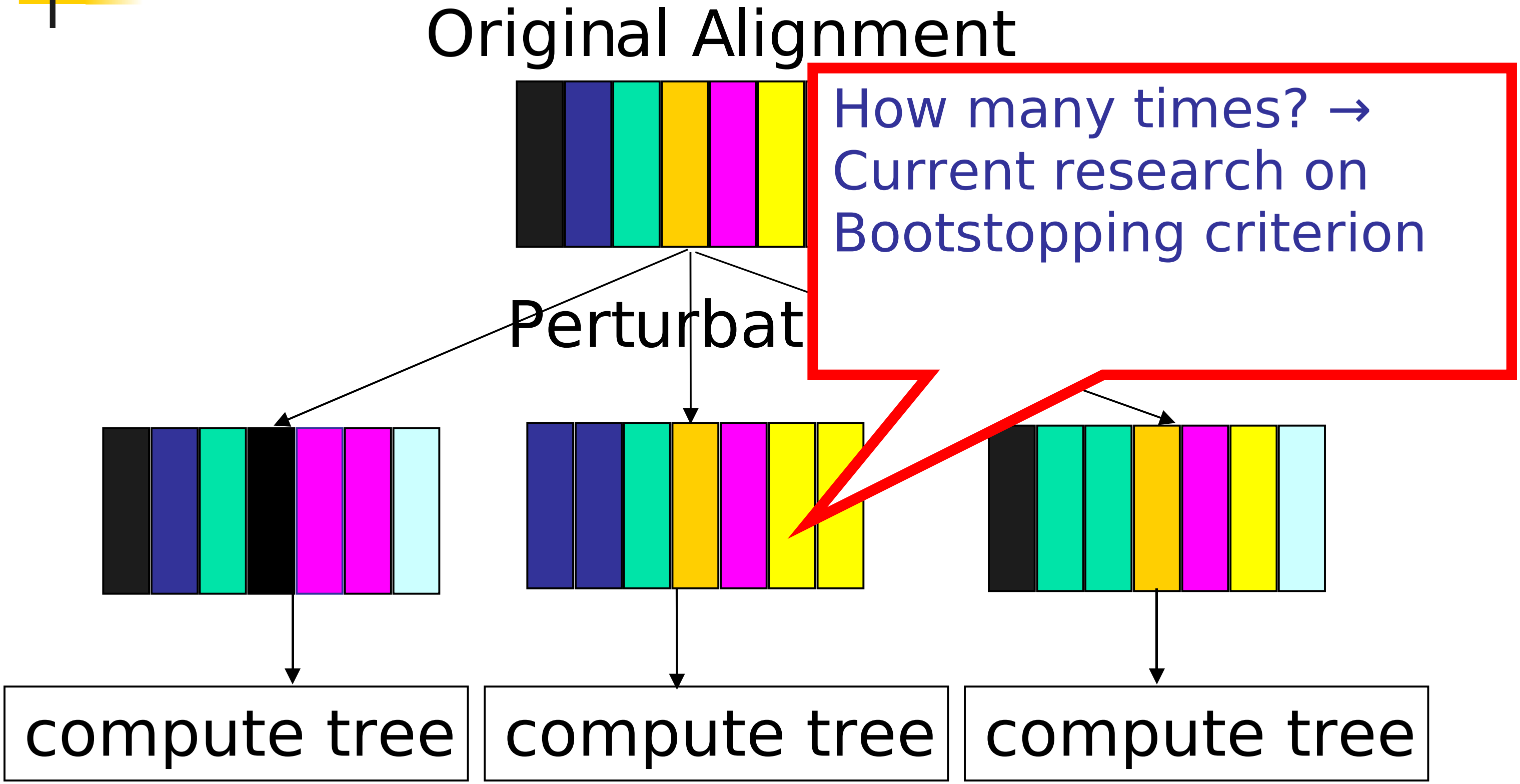
compute tree

Bootstrapping



This needs to be done 100-1,000 times! Embarrassingly parallel problem!

Bootstrapping

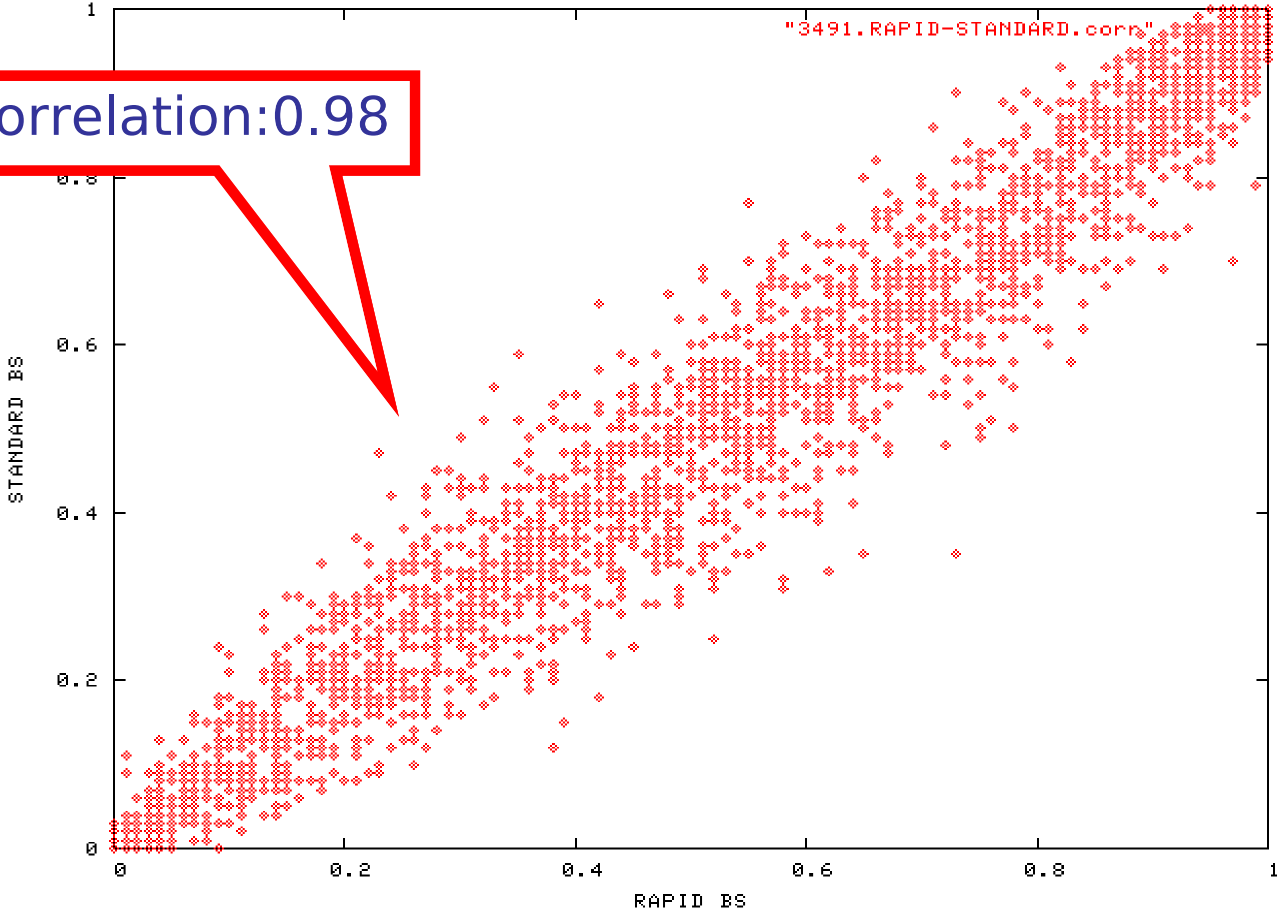


How many times? → Current research on Bootstopping criterion

Rapid Bootstrapping Algorithm: Algorithmic Engineering

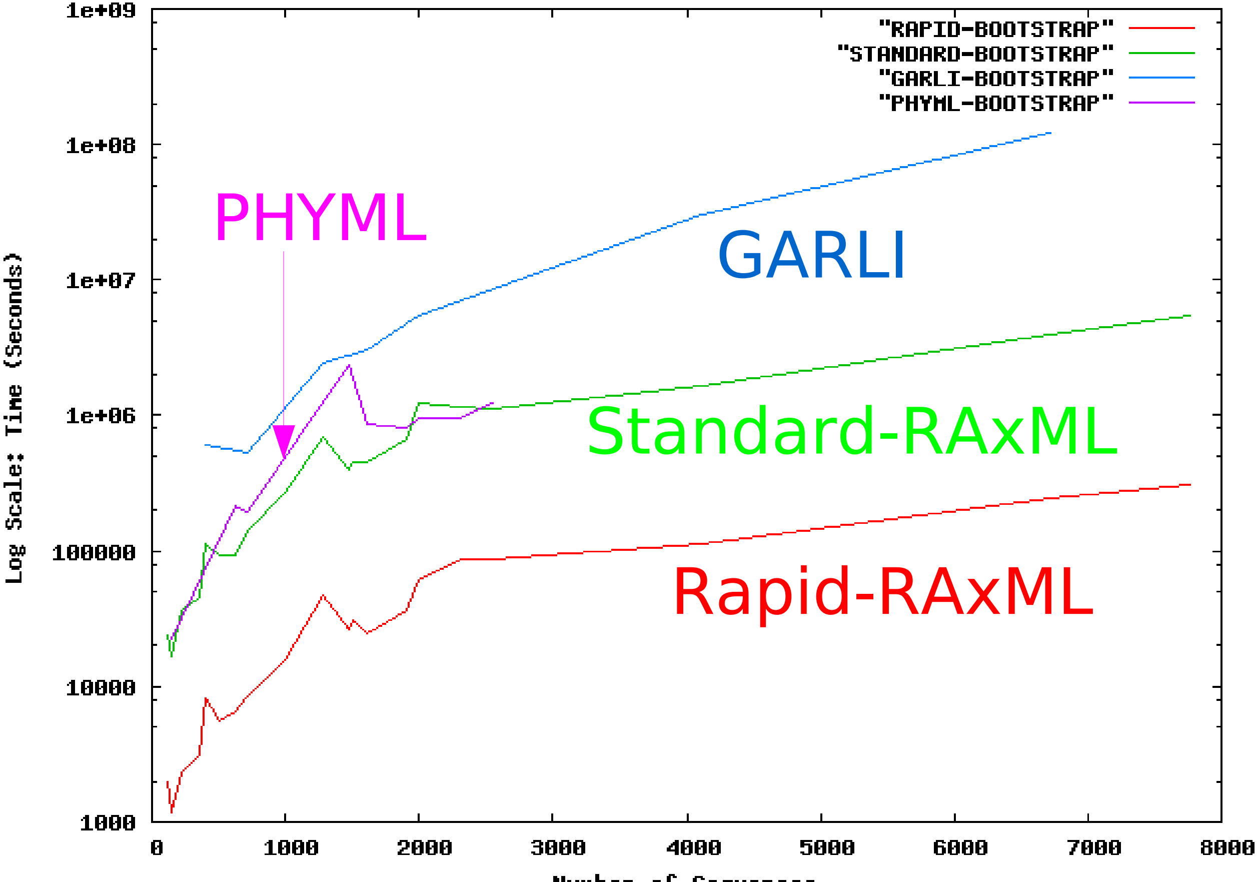
- Tested on **22 diverse** (mammals, bacteria, archaea, grasses, fishes, plants, viral) **real-world** DNA/AA single-/multi-gene datasets containing **125-7,764 sequences**
- Pearson correlation on best-scoring ML trees between RBS (Rapid BS) & SBS (Standard BS) support values 0.95-0.99 (except one dataset at 0.91), **average 0.97**
- Weighted topological distance < 6%, **average 4%**
- **Program Acceleration: 8-20, average ≈ 15**
  - Acceleration by one order of magnitude
  - **Full ML analysis (100BS + ML search) of datasets of up to 5,000 sequences within less than 5 days on your desktop!**
  - Allows for a sufficiently large number of Bootstrap replicates
- Released in January 2008
- To be published in Systematic Biology soon

Relative Accuracy: Correlation on 3,500 rBCL sequences

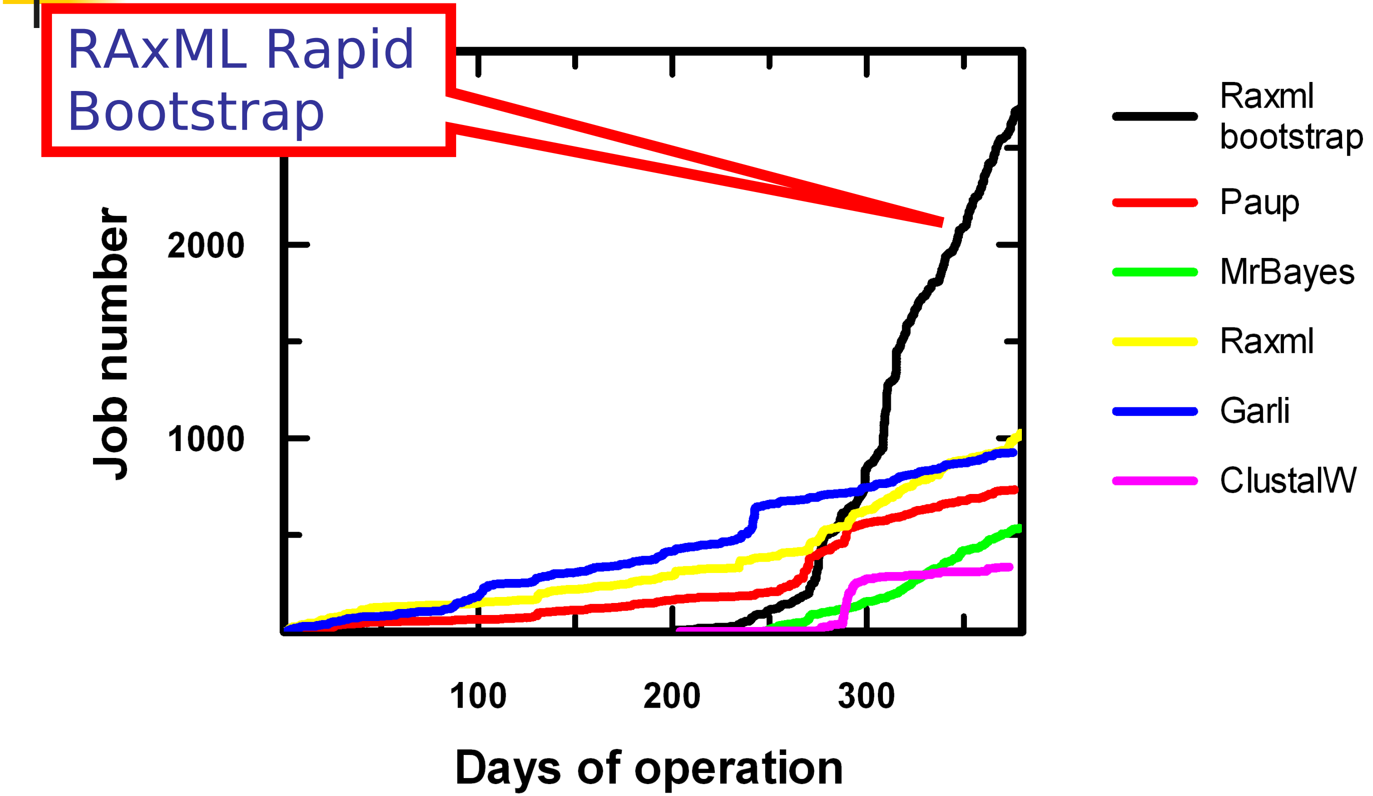


Correlation:0.98

Rapid Bootstrap Inference Times for 100 BS replicates



Impact of Rapid Bootstrap



RAxML Rapid Bootstrap

Courtesy of Mark Miller SDSC

Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Web Servers
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

Optimization of the Likelihood Function

- Likelihood functions (usually 3-4 functions) account for over 95% of total execution time
- Algorithmic Optimization
  - Detection of equal patterns and re-use of previously computed values
  - Special Function version for tip/tip and tip/inner node likelihood vectors
- Technical Optimization
  - Manual loop-unrolling
  - Consider pipeline efficiency
  - Replace x/y by x * 1/y etc
  - Cache efficiency
  - Individual ML implementation for each substitution model

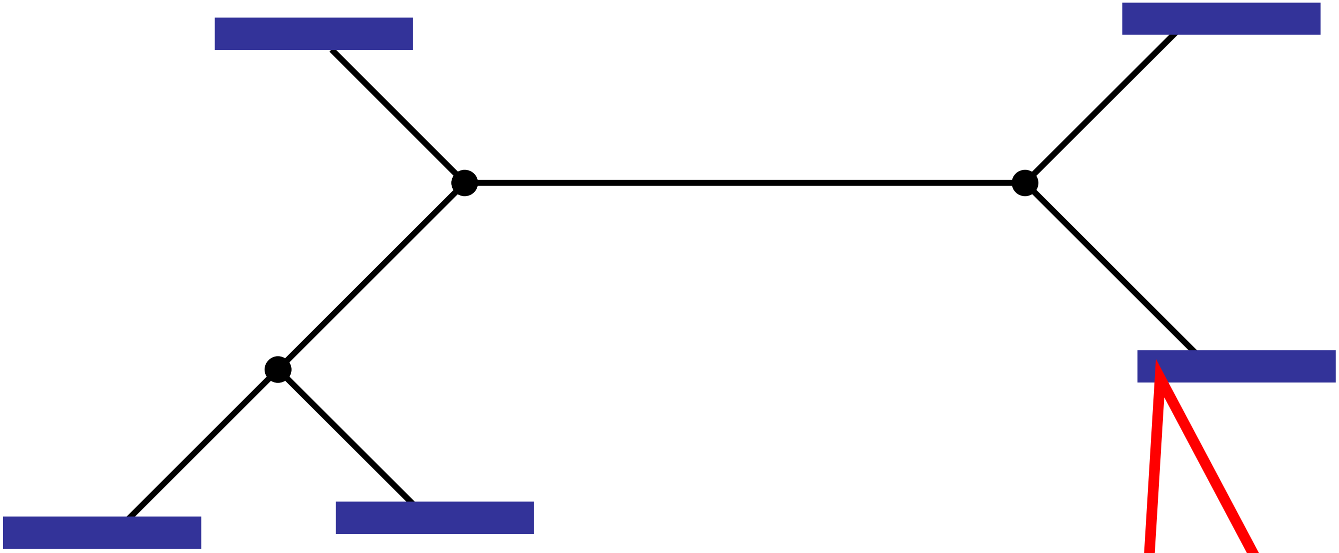
Expensive Likelihood Function: Consequences

- Likelihood Function is expensive
  - Try to reduce # of invocations by algorithmic means
  - Use “cheaper” (in terms of FLOPs) criteria to pre-score alternative trees
  - Problem: These cheaper methods must correlate with the Likelihood function

Model Selection

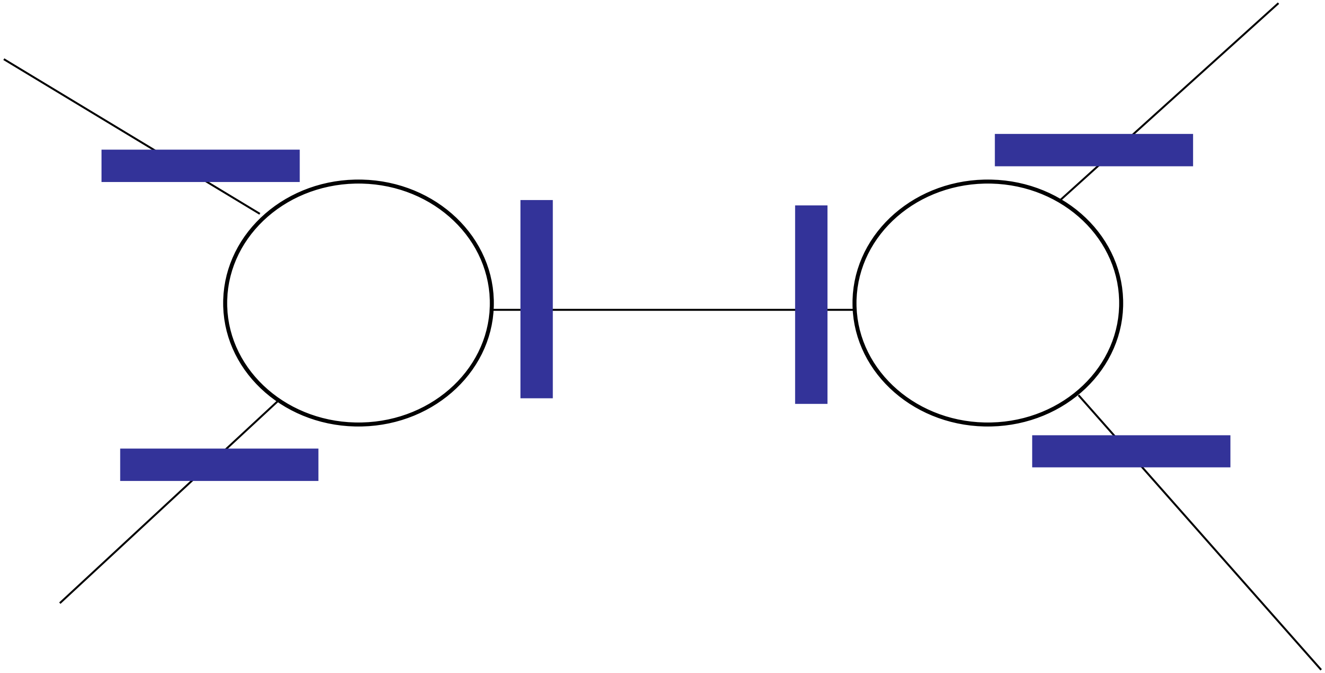
- Different statistical models of evolution
    - Complex models → many FLOPs & good accuracy
    - Simple models → less FLOPs & bad accuracy
- Trade-Off: speed versus accuracy
- Likelihood surface is smooth for complex models → less local maxima

Memory Organization: Tip Vectors

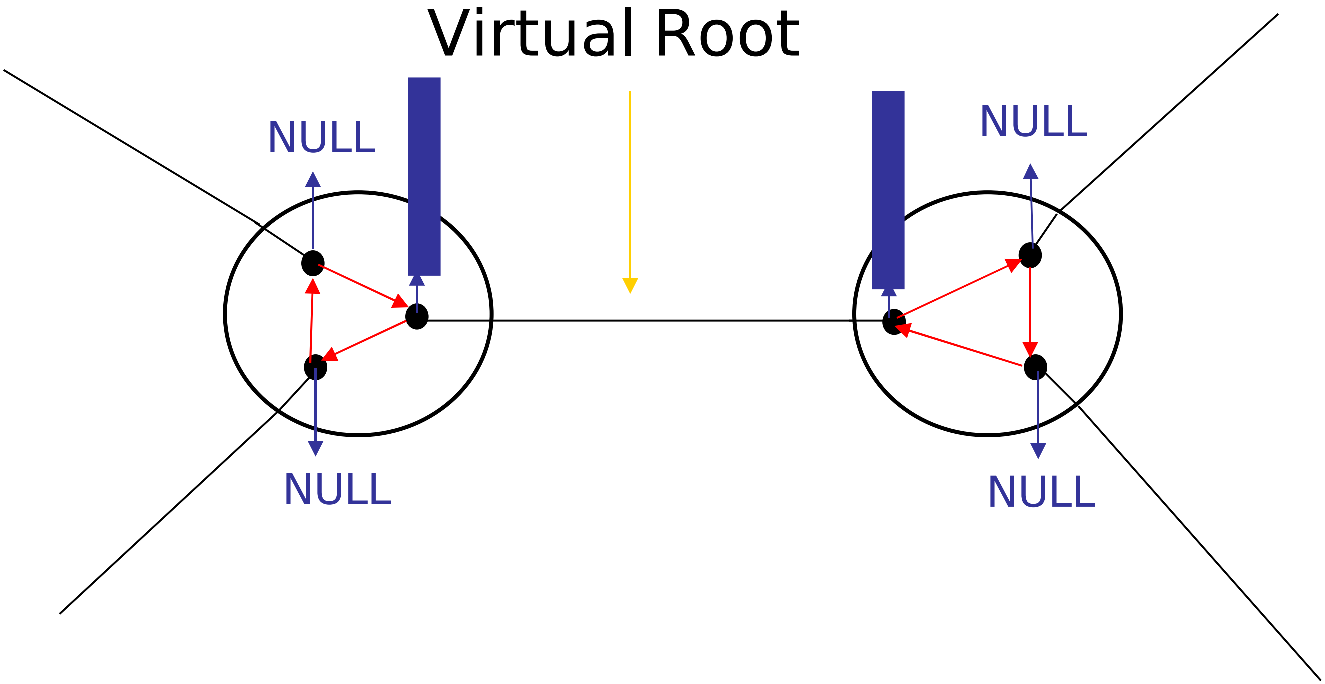


Tip vectors don’t change with topology and are cheap to calculate

Memory Organization: Inner Vectors with Unrooted View

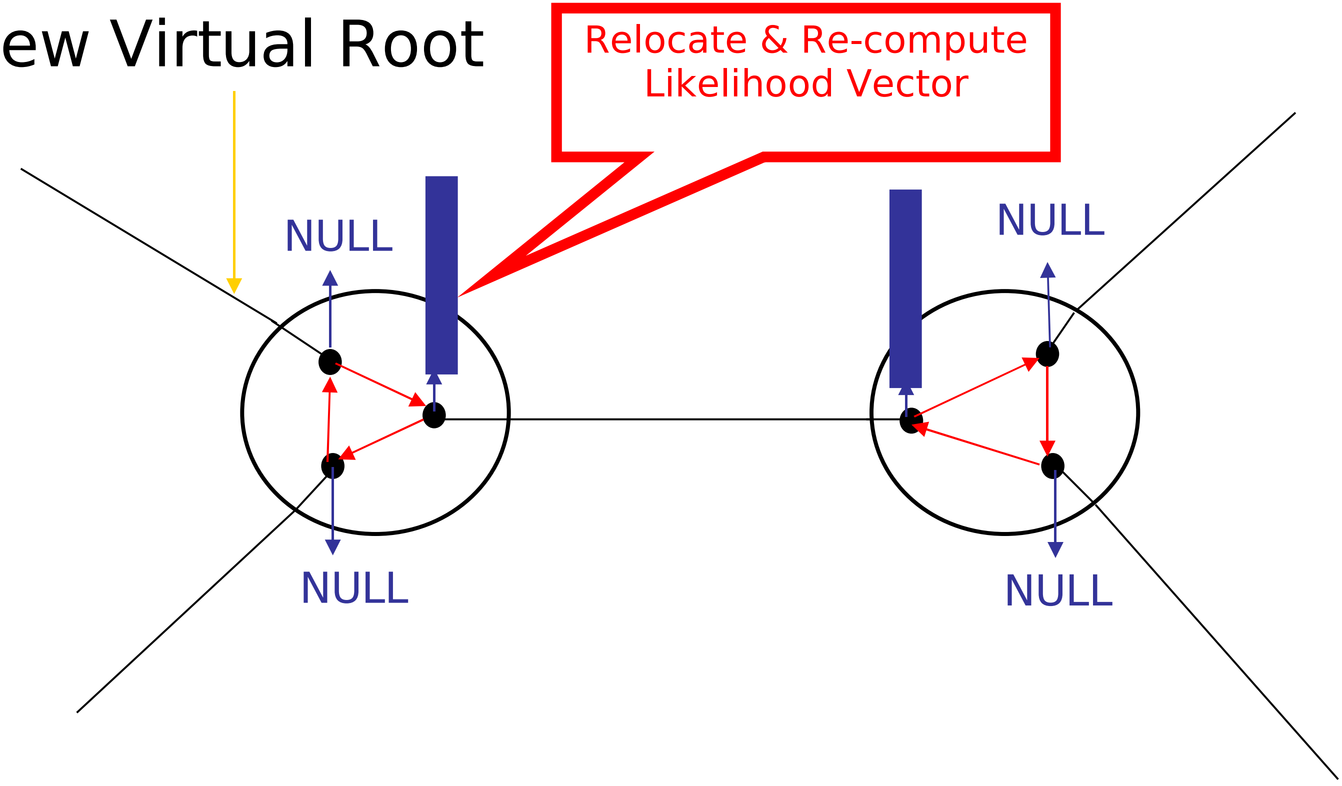


Memory Organization: Inner Vectors with Rooted View



Memory Organization: Inner Vectors with Rooted View

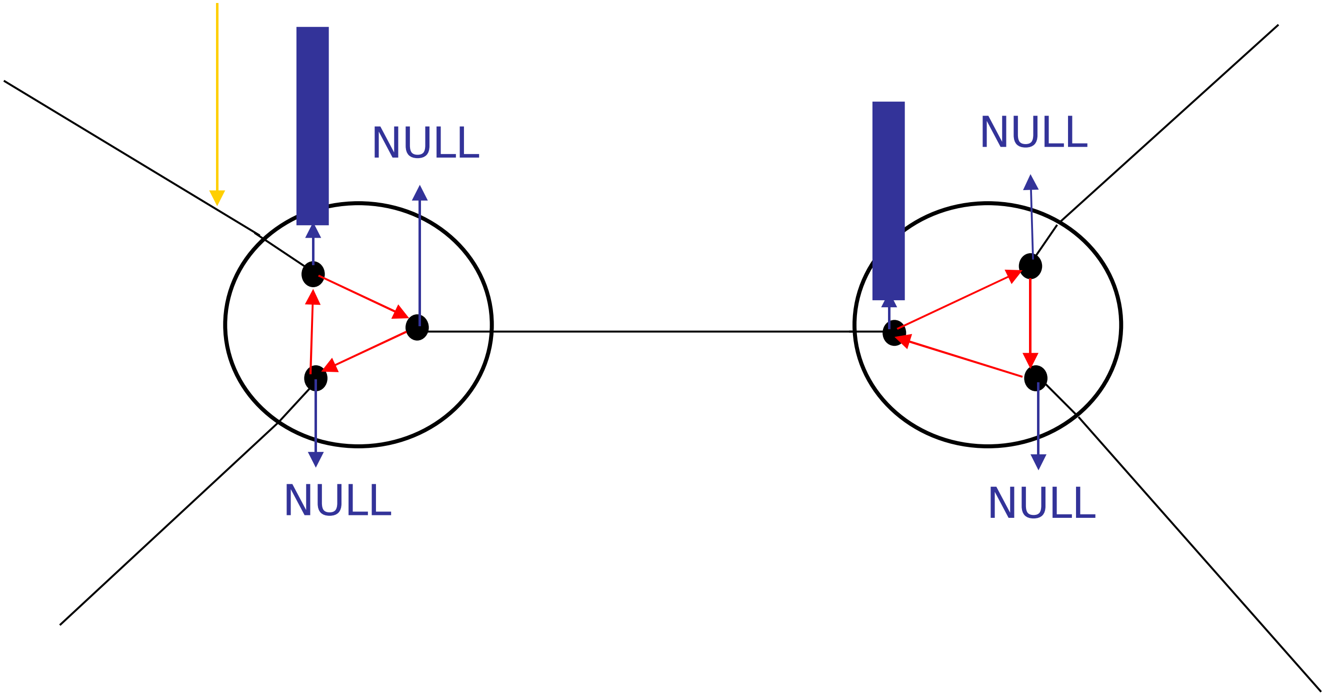
New Virtual Root



Relocate & Re-compute Likelihood Vector

Memory Organization: Inner Vectors with Rooted View

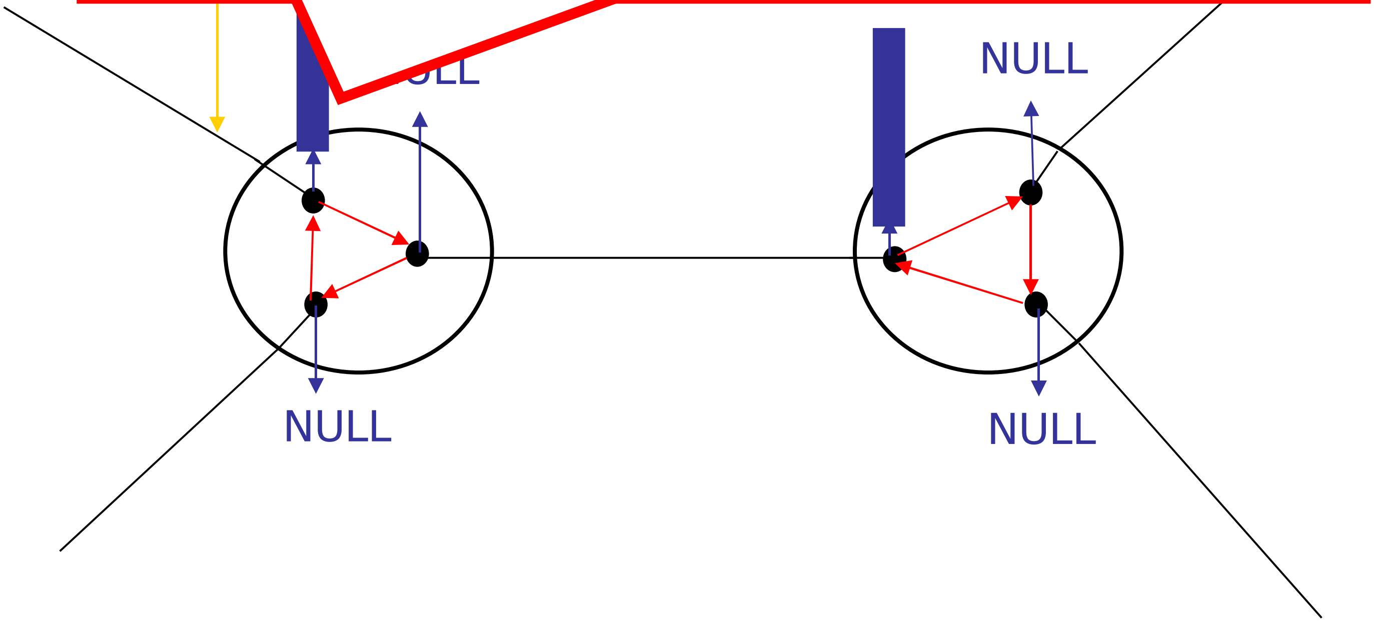
New Virtual Root



Memory Organization: Inner Vectors with Rooted View

New Virtual Root

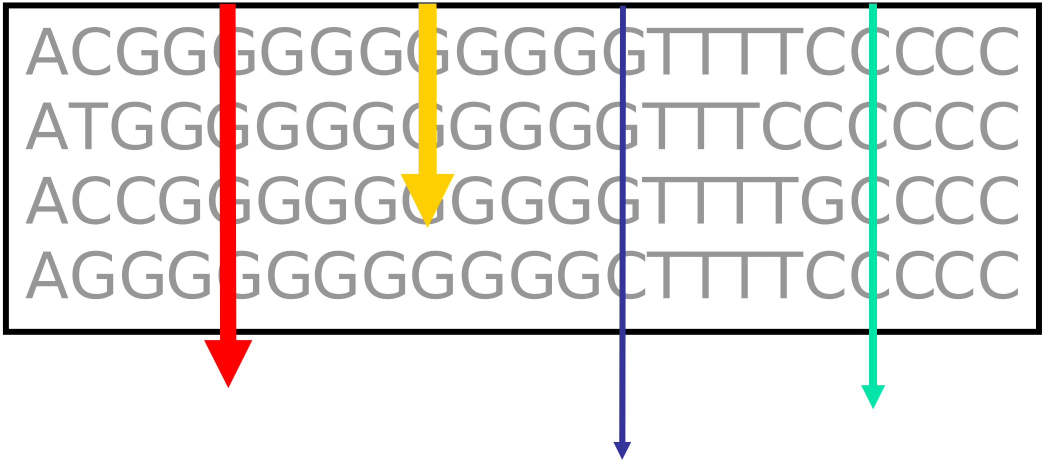
Memory Consumpption =Θ(#seqs*#patterns)



Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Web Servers
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

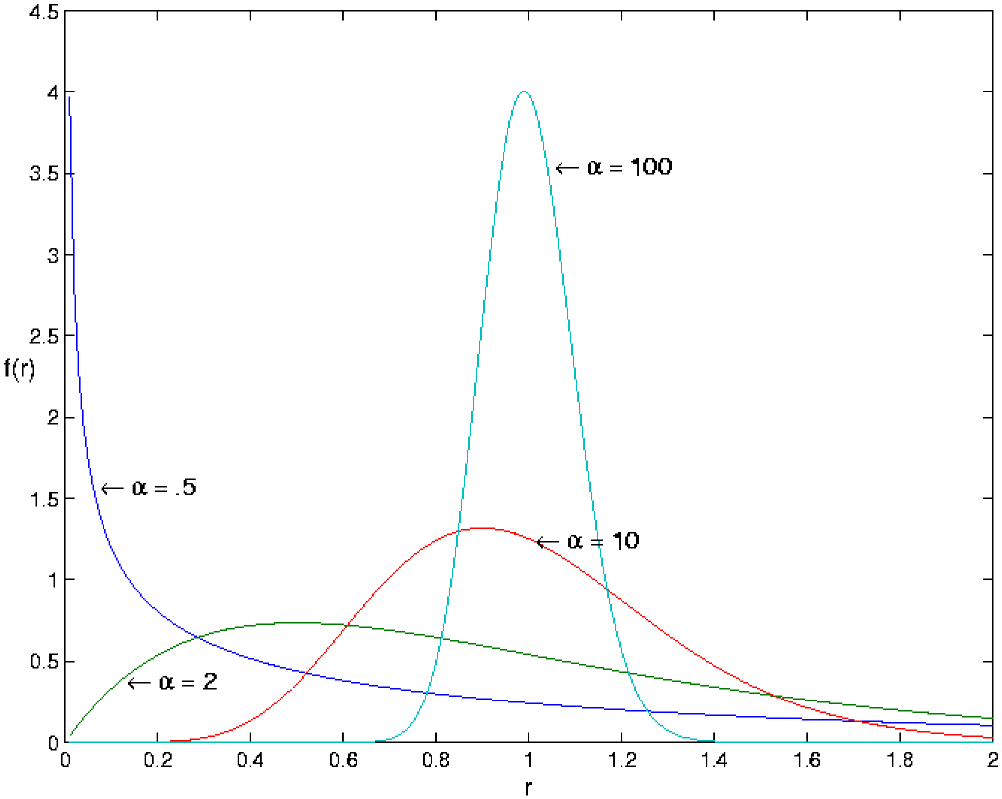
Rate Heterogeneity among Sites



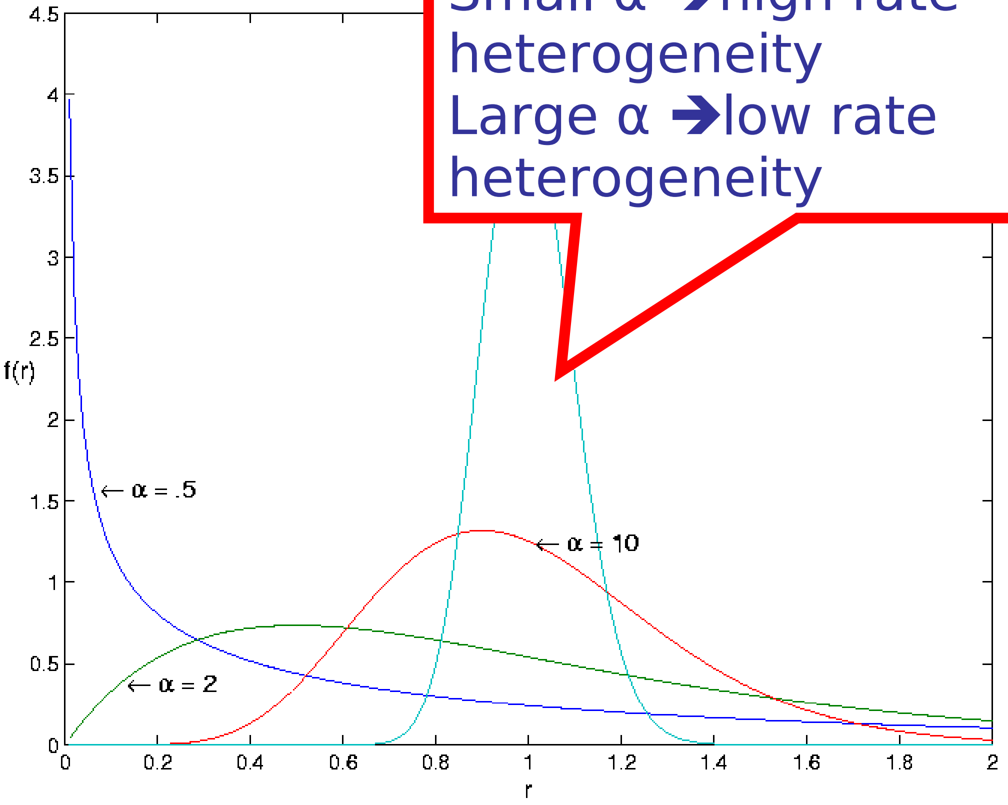
ACGGGGGGGGGGGTTTTCCCCC
ATGGGGGGGGGGGTTTCCCCCC
ACCGGGGGGGGGGTTTTGCCCC
AGGGGGGGGGGGCTTTTCCCCC

- Efficient approximation for the “gold standard” GTR+Γ model of rate heterogeneity among sites
  - execution time improvement: factor 4
  - memory footprint reduction: factor 4
  - returns equally good trees under GTR+Γ

Γ-Distribution

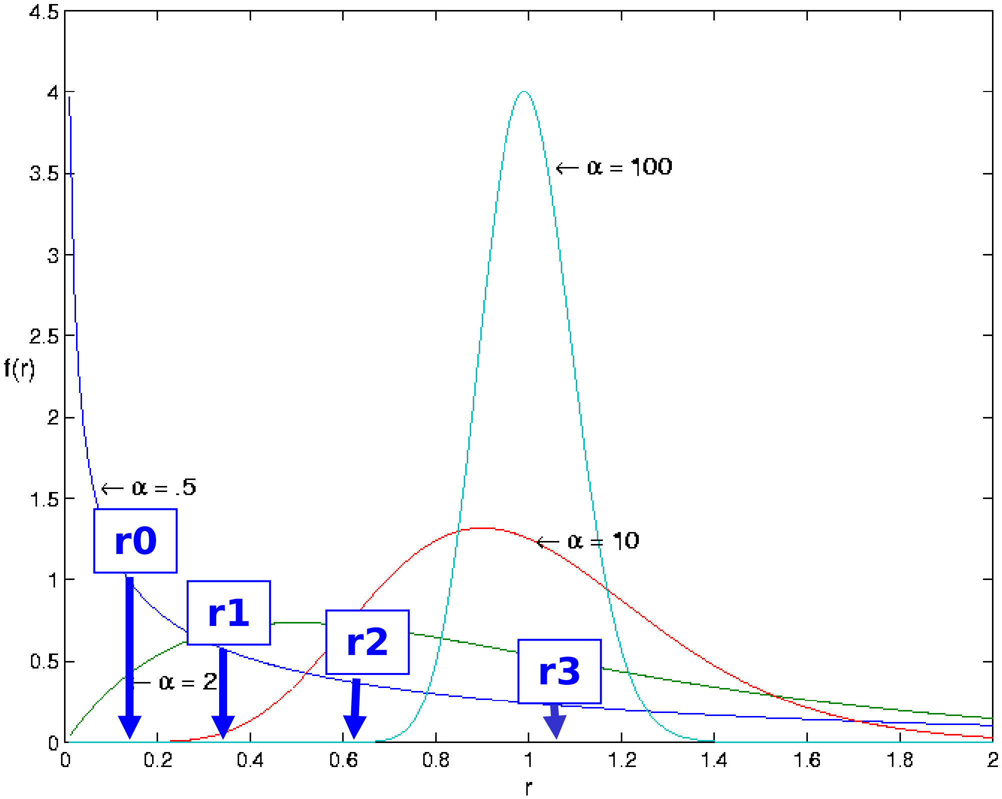


Γ-Distribution



Small α ➔high rate heterogeneity
Large α ➔low rate heterogeneity

Discrete Γ-Distribution



ML-Loops

CAT-LOOP

```
for(i = 0; i < m; i++)
{
   cat = category[i];
   r = rate[cat];
   p[i] = f(q[i], pq, r[i], pr, r);
}
```

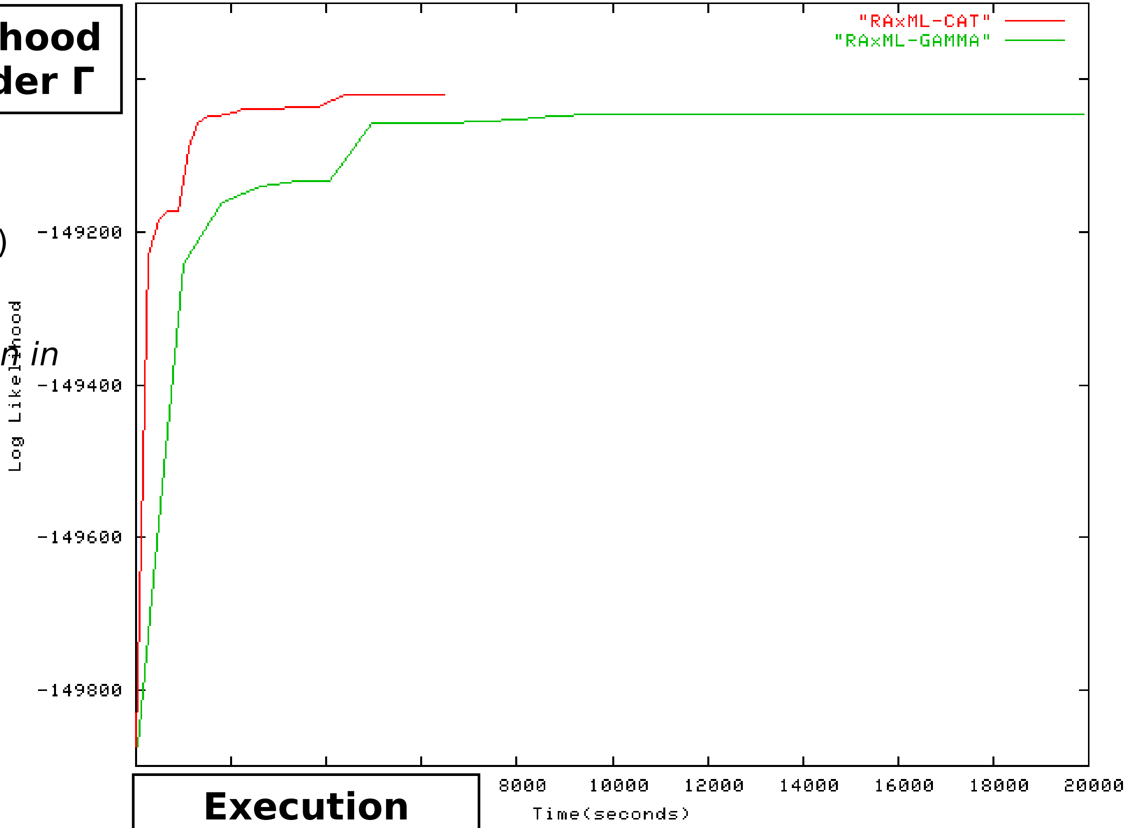
Γ-LOOP

```
for(i = 0; i < m; i++)
{
   p[i].g0 = f(q[i], pq, r[i], pr, r0);
   p[i].g1 = f(q[i], pq, r[i], pr, r1);
   p[i].g2 = f(q[i], pq, r[i], pr, r2));
   p[i].g3 = f(q[i], pq, r[i], pr, r3));
}
```

715 Sequences under HKY85+Γ

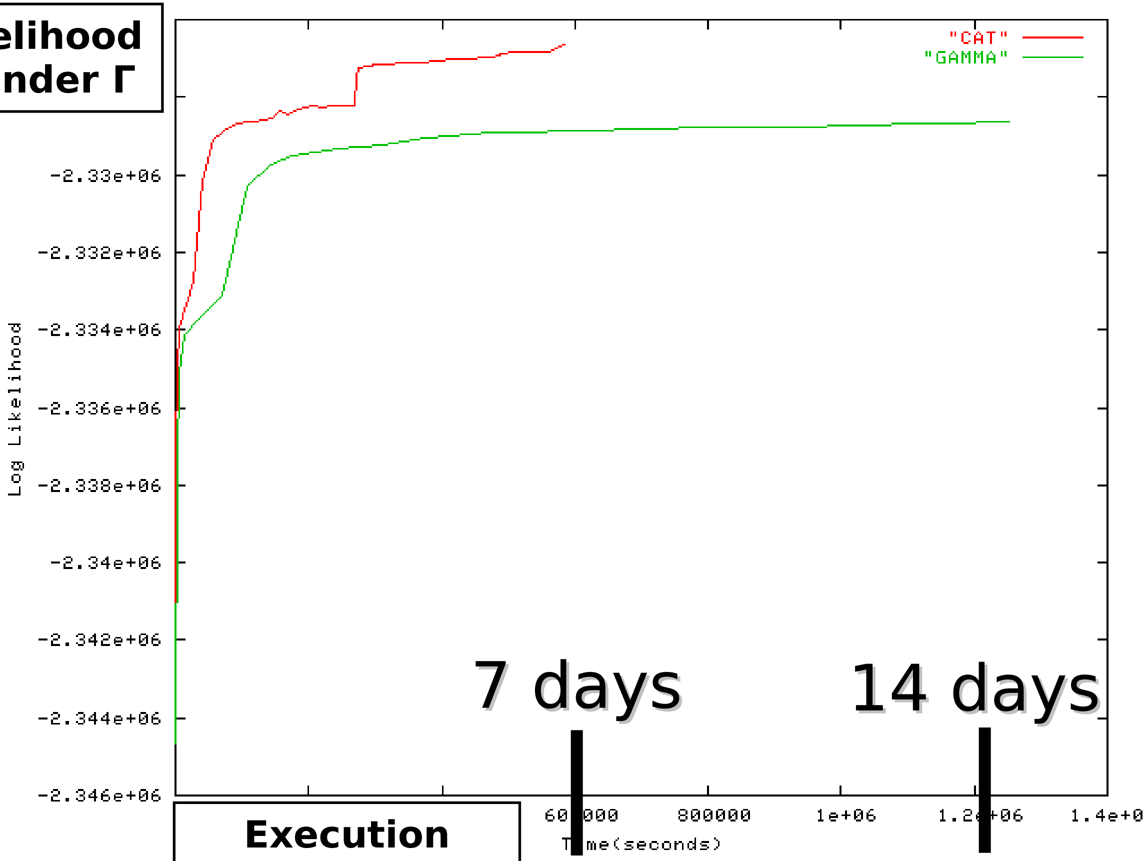
Log Likelihood Score under Γ

C.E. Robertson et al (2005) Phylogenetic diversity and ecology of environmental Archaea, In *Current Opinion in Microbiology*.



Execution Time

8,864 Bacteria under GTR+Γ and GTR+CAT



Log Likelihood Score under Γ

7 days

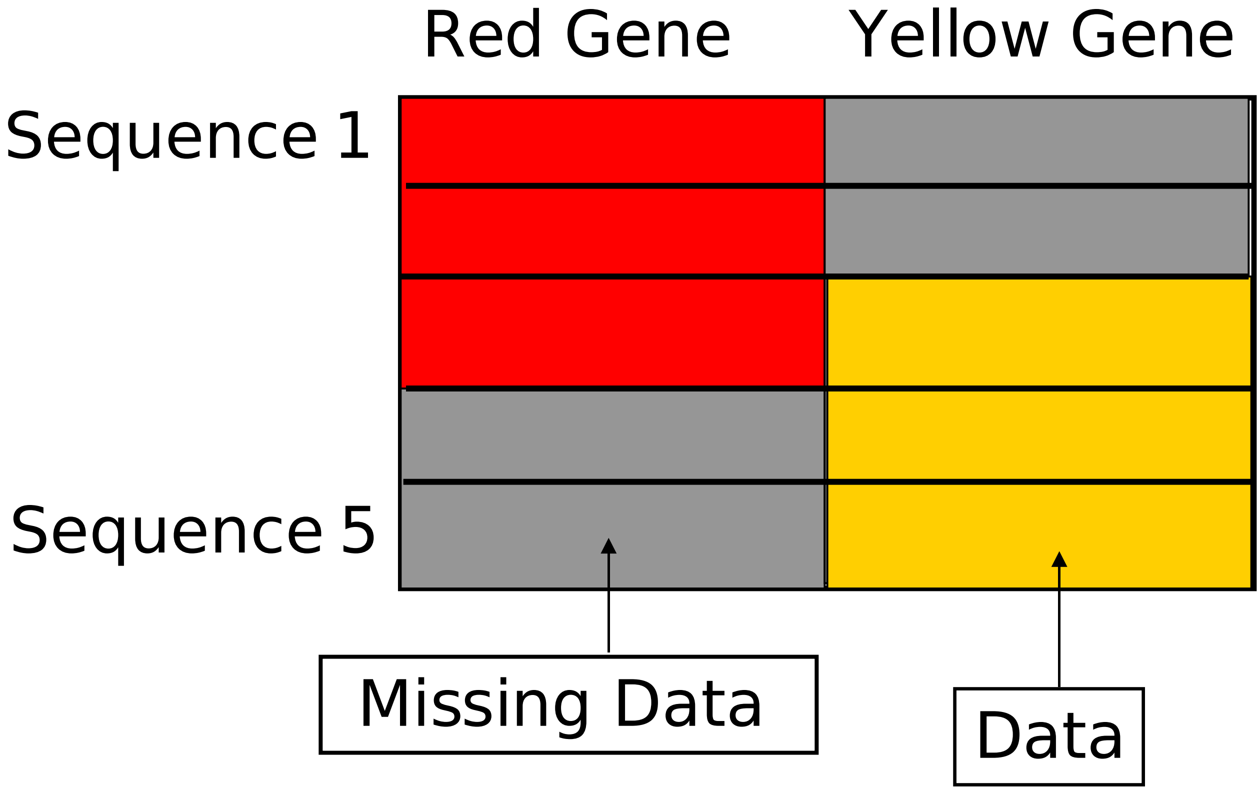
14 days

Execution Time

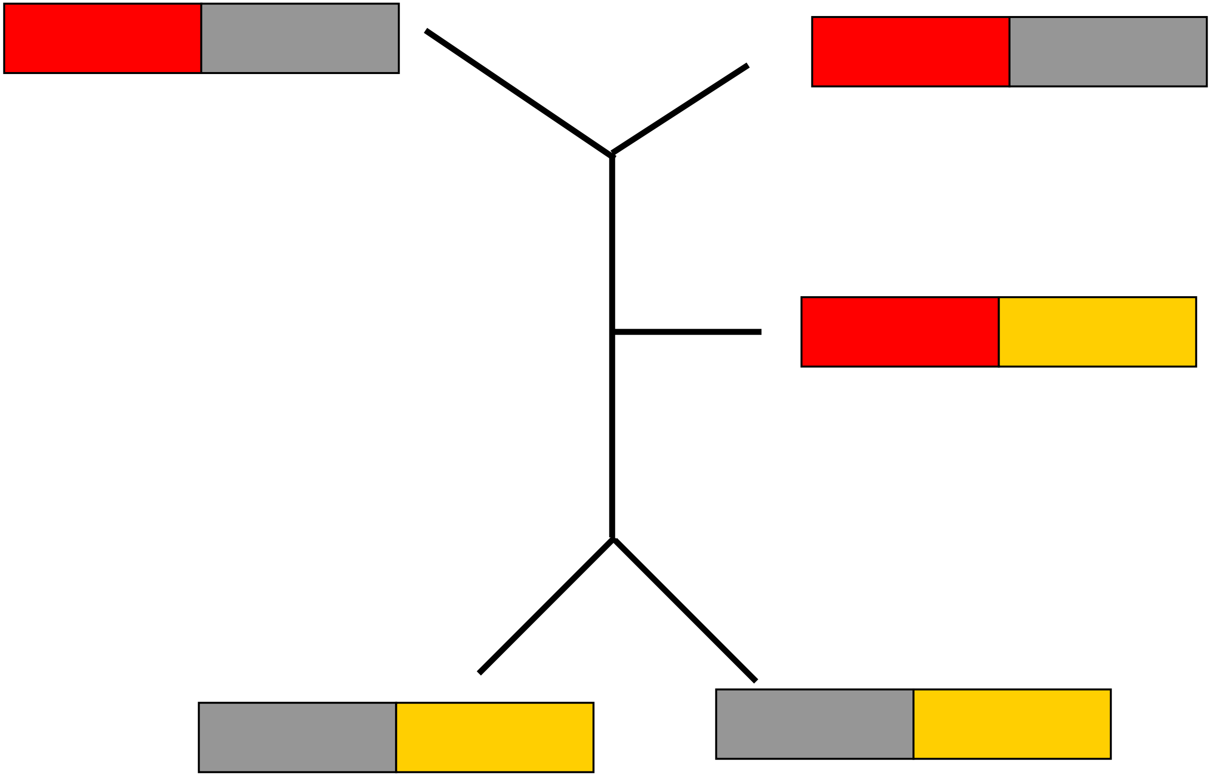
Current Challenge

- Adapt likelihood function and data structures to increasingly common “gappy” multi-gene alignments

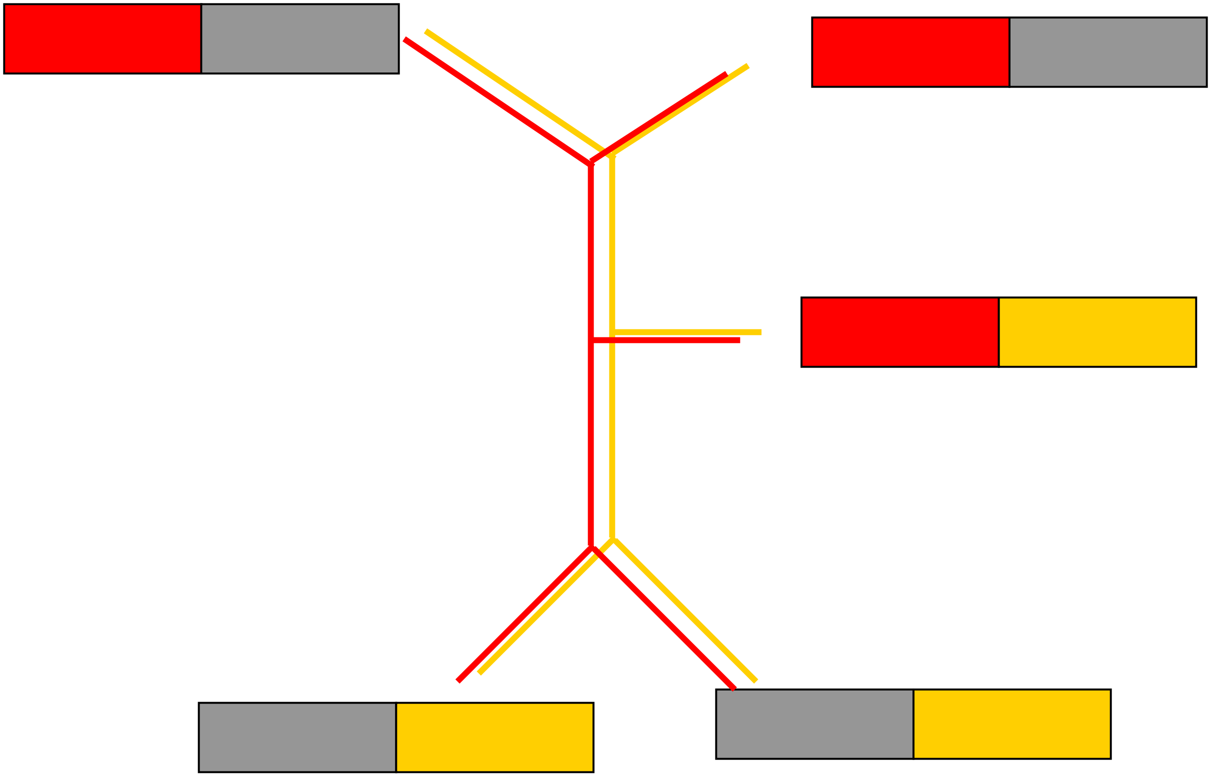
A Current Problem: Handling Multi-Gene Alignments



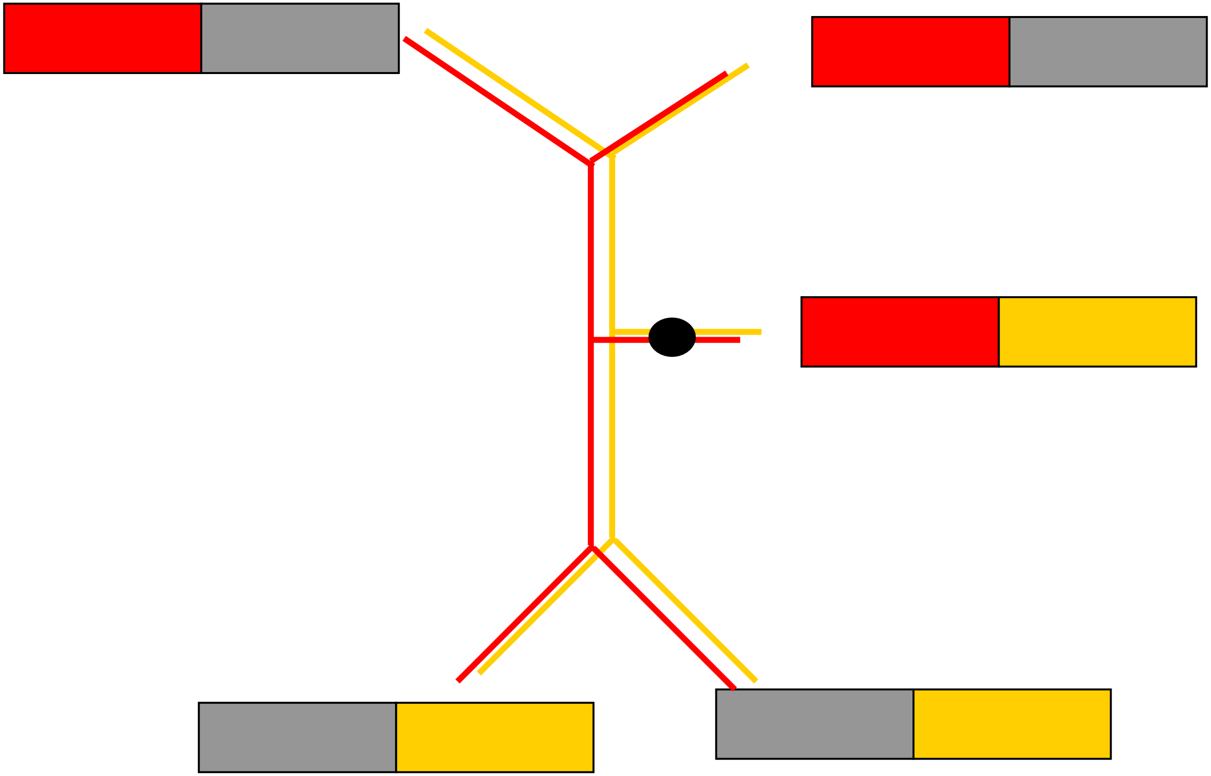
A Multi-Gene Model



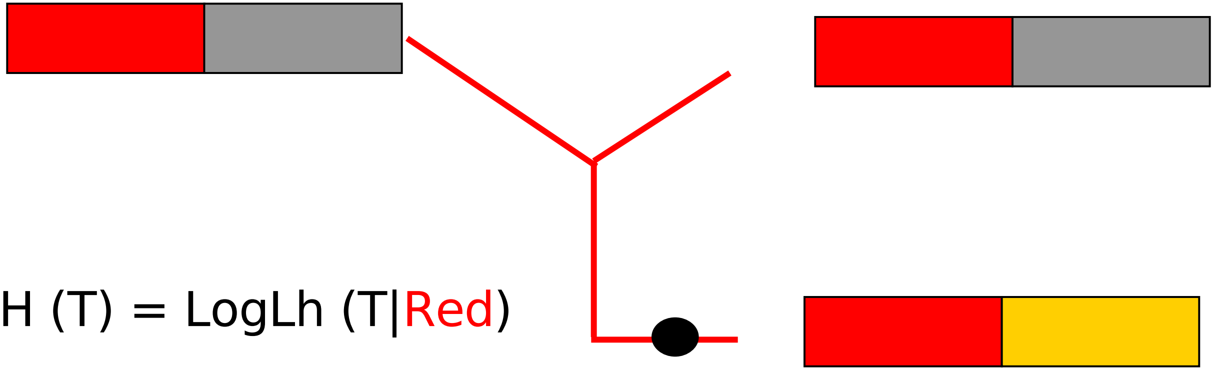
A Multi-Gene Model



A Multi-Gene Model



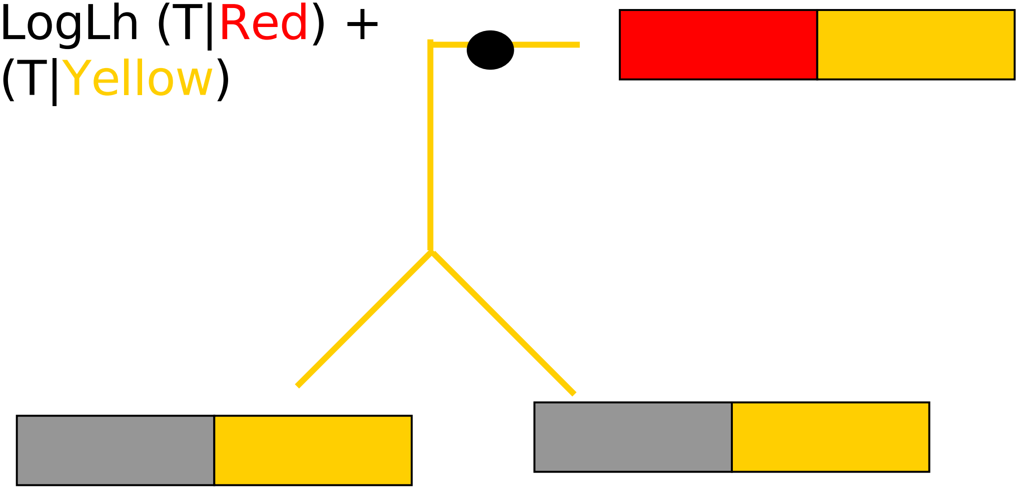
A Multi-Gene Model



LogLH (T) = LogLh (T|Red)

A Multi-Gene Model

LogLH (T) = LogLh (T|Red) + LogLH(T|Yellow)



Initial Results

- 2 datasets
  - 400 sequences, 13,000 base-pairs (alignment columns), 11 genes, gappyness 70%
  - 2,200 sequences, 51,000 base-pairs, 68 genes, gappyness 90% (memory footprint 9GB)
- Full tree traversal (AMD Opteron)
  - 400: 4 times faster
  - 2,200: 13 times faster
- Branch Length Optimization (AMD Opteron)
  - 400: 30 times faster
  - 2,200: 46 times faster
- Initial implementation does not exploit the potential memory footprint reduction

Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Web Servers
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

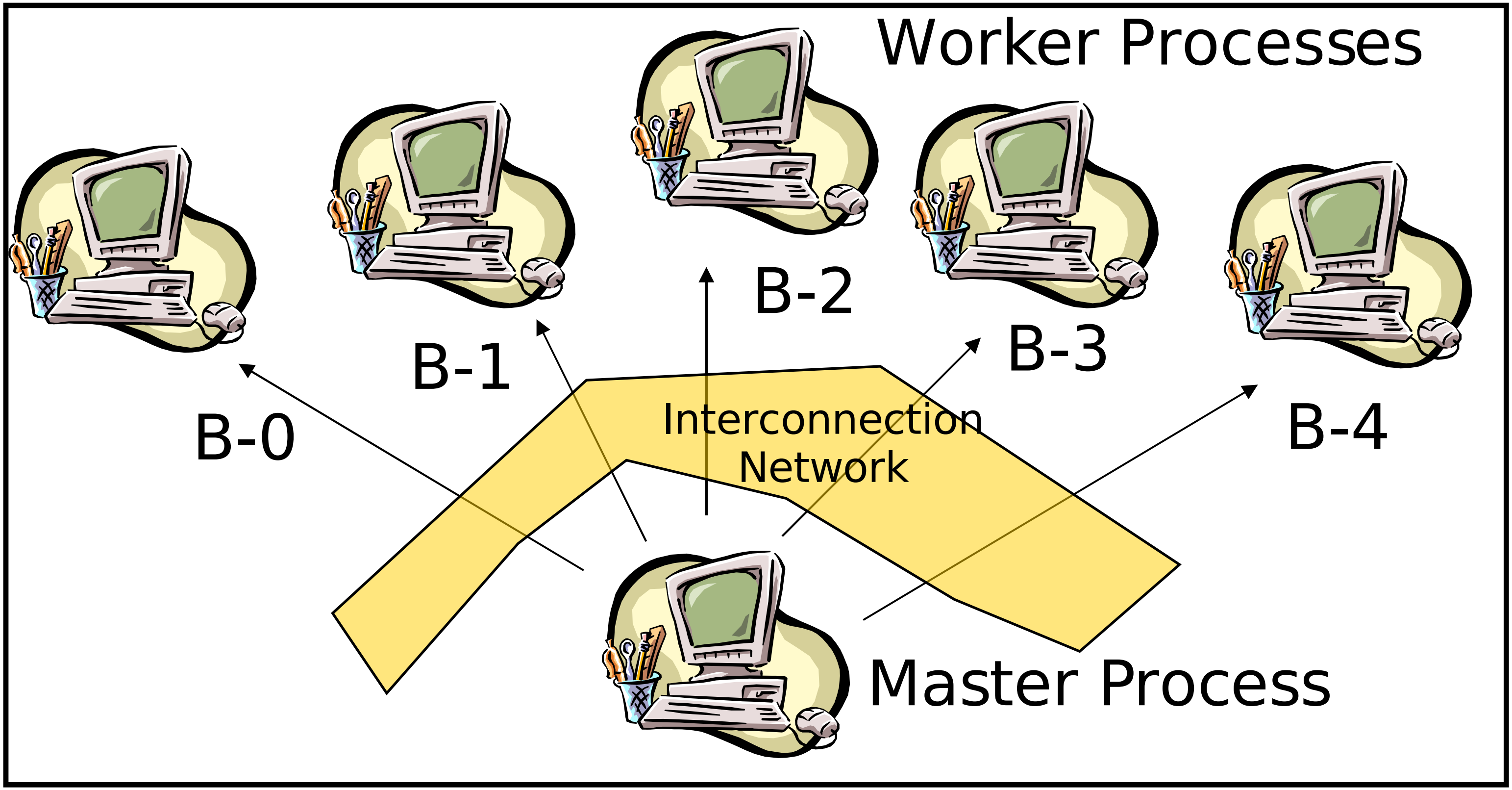
Levels of Parallelism

Embarrassing Parallelism

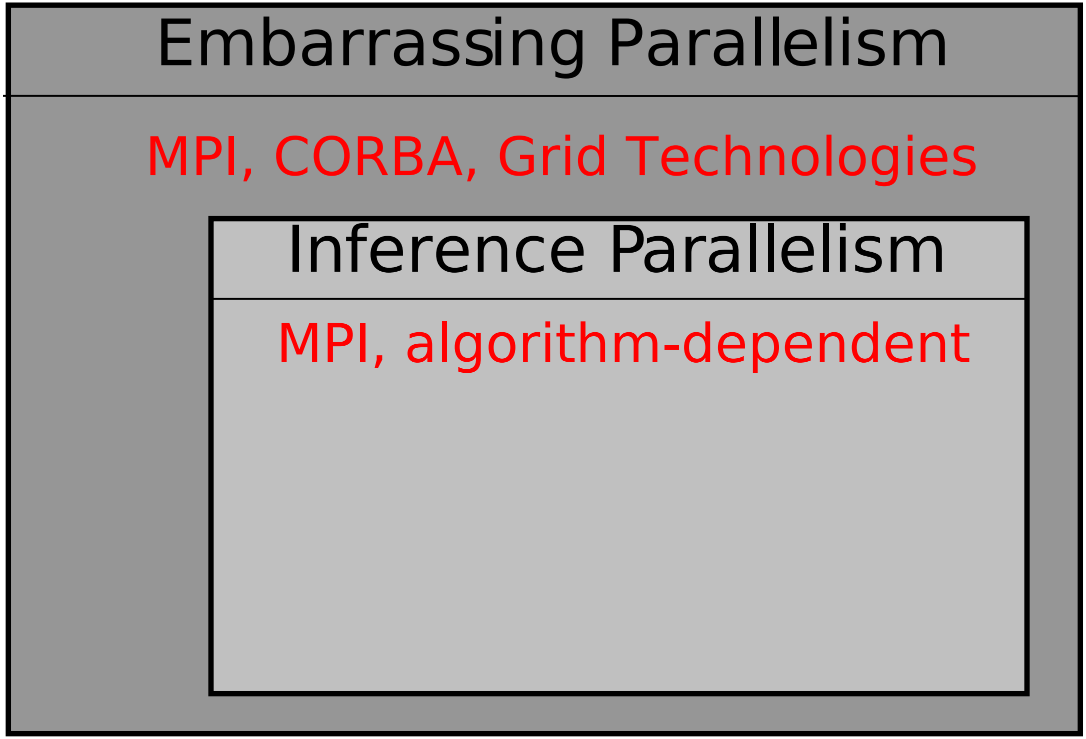
MPI, CORBA, Grid Technologies

Embarrassing Parallelism: MPI Version of RAxML

PC-CLUSTER



Levels of Parallelism



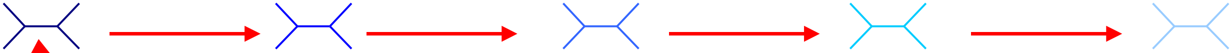
Embarrassing Parallelism

MPI, CORBA, Grid Technologies

Inference Parallelism

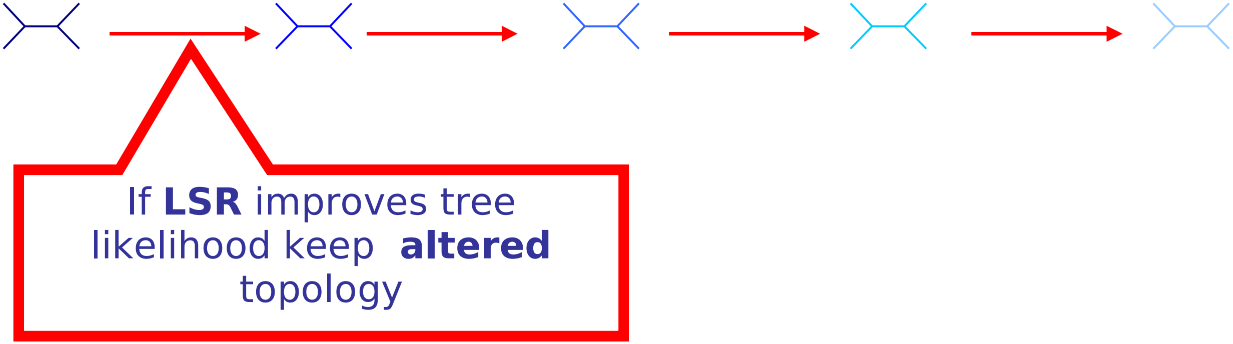
MPI, algorithm-dependent

Inference Parallelism: Dependency Problem in RAxML



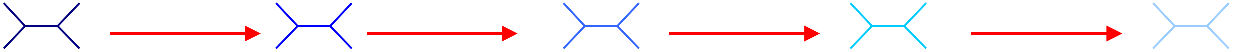
Apply a Lazy Subtree Rearrangement (**LSR**) to currently best tree and evaluate likelihood

Inference Parallelism: Dependency Problem in RAxML



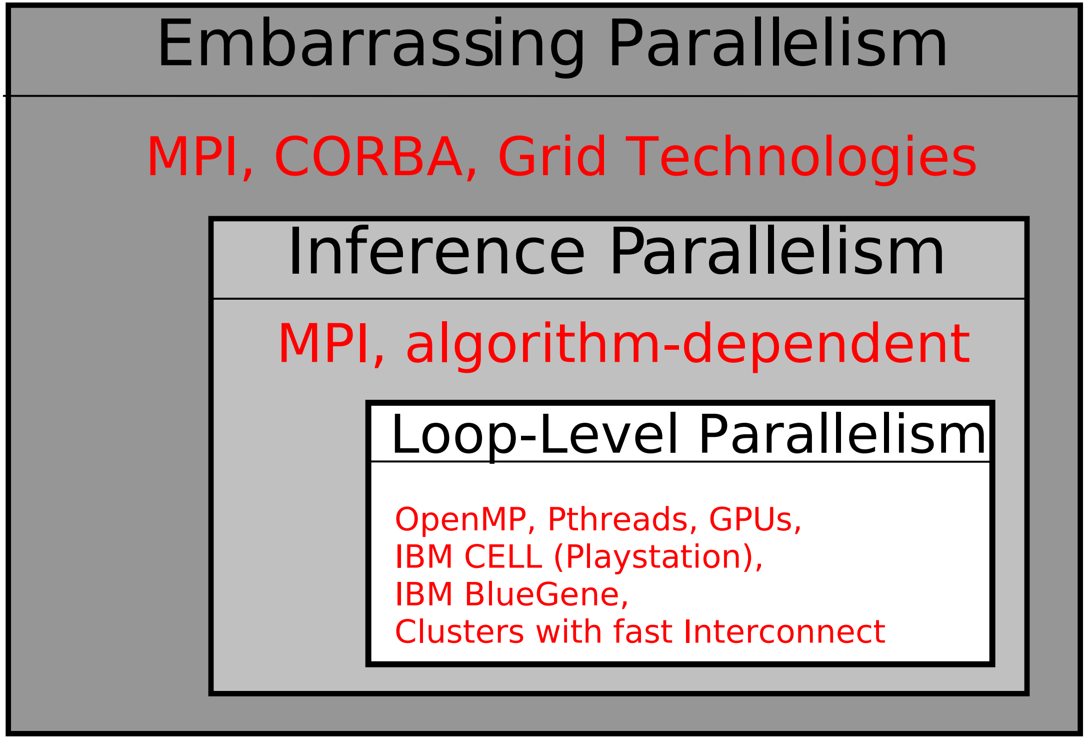
If **LSR** improves tree likelihood keep **altered** topology

Inference Parallelism: Dependency Problem in RAxML



- Each worker process evaluates the rearrangements for **one** subtree at a time
- One optimization cycle consists of 2 * #organisms LSRs
- Many improved topologies are encountered during one cycle
- Many sequential dependencies ➔ hard to parallelize
- Use Non-determinism to solve problem

Levels of Parallelism



Embarrassing Parallelism

MPI, CORBA, Grid Technologies

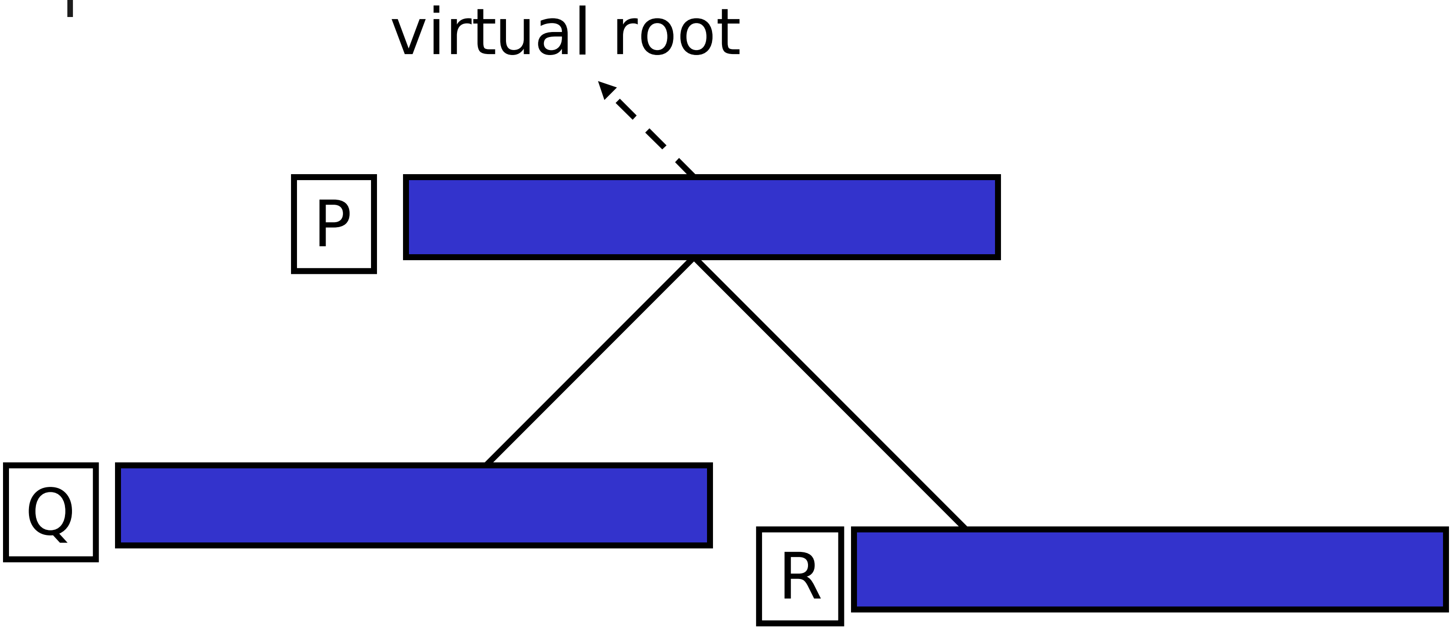
Inference Parallelism

MPI, algorithm-dependent

Loop-Level Parallelism

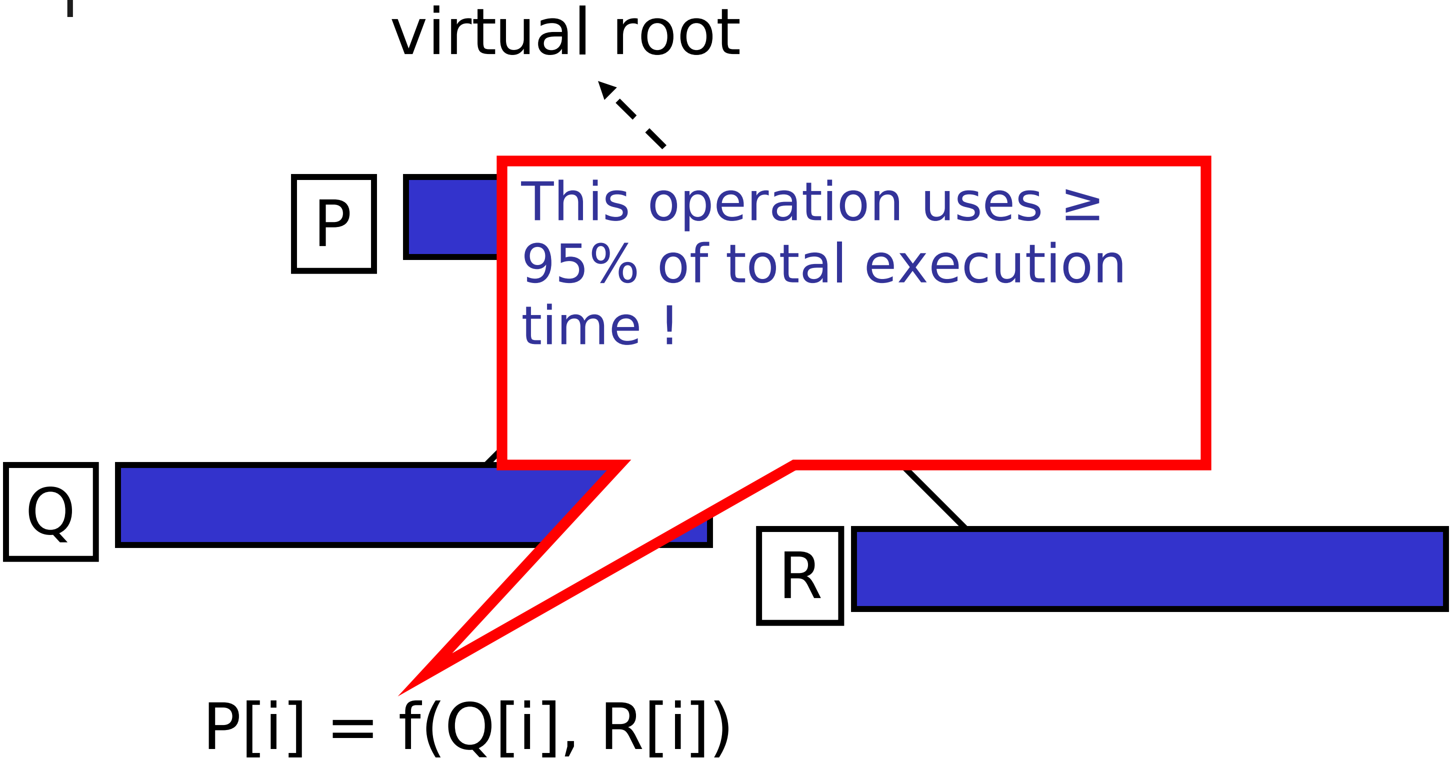
OpenMP, Pthreads, GPUs,
IBM CELL (Playstation),
IBM BlueGene,
Clusters with fast Interconnect

Loop Level Parallelism



P[i] = f(Q[i], R[i])

Loop Level Parallelism



virtual root

P

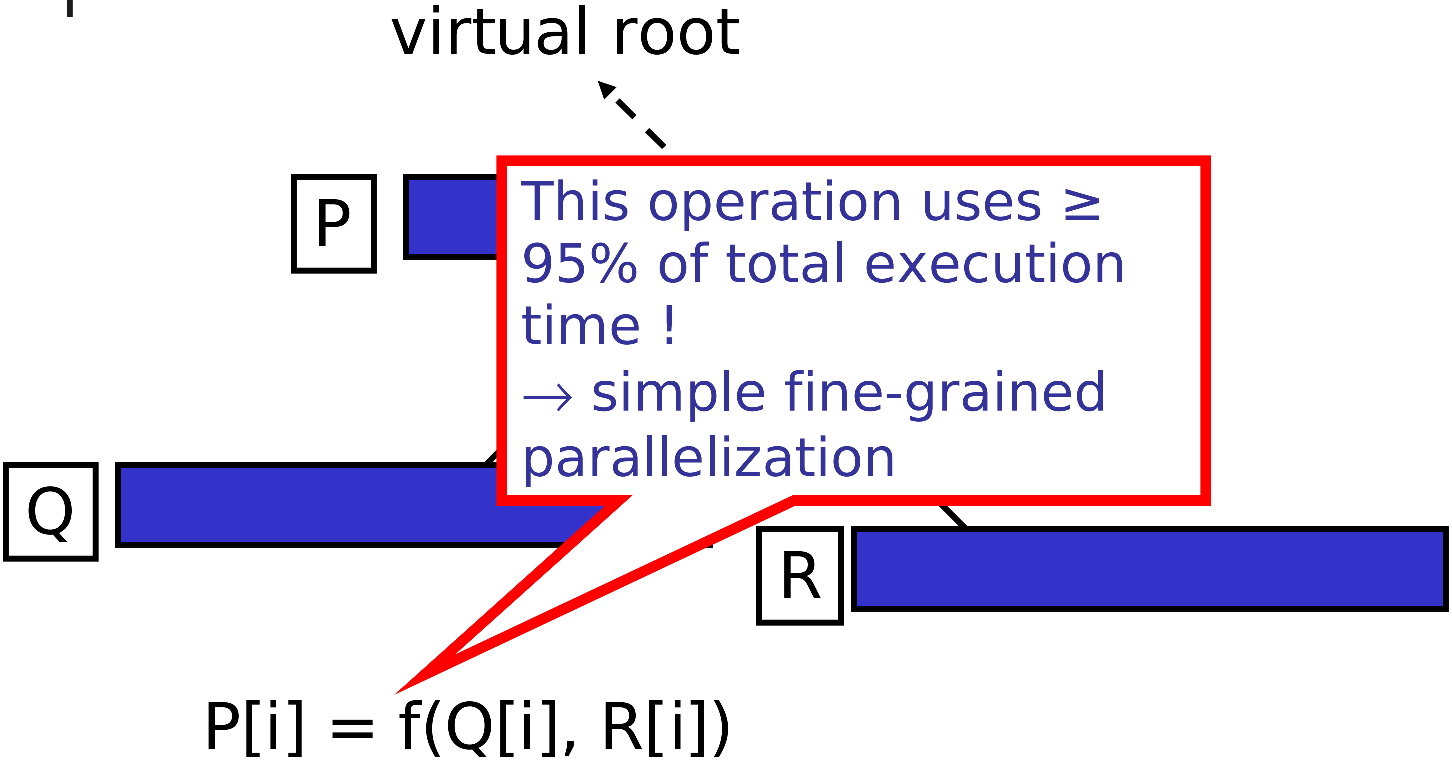
Q

R

This operation uses ≥ 95% of total execution time !

$P[i] = f(Q[i], R[i])$

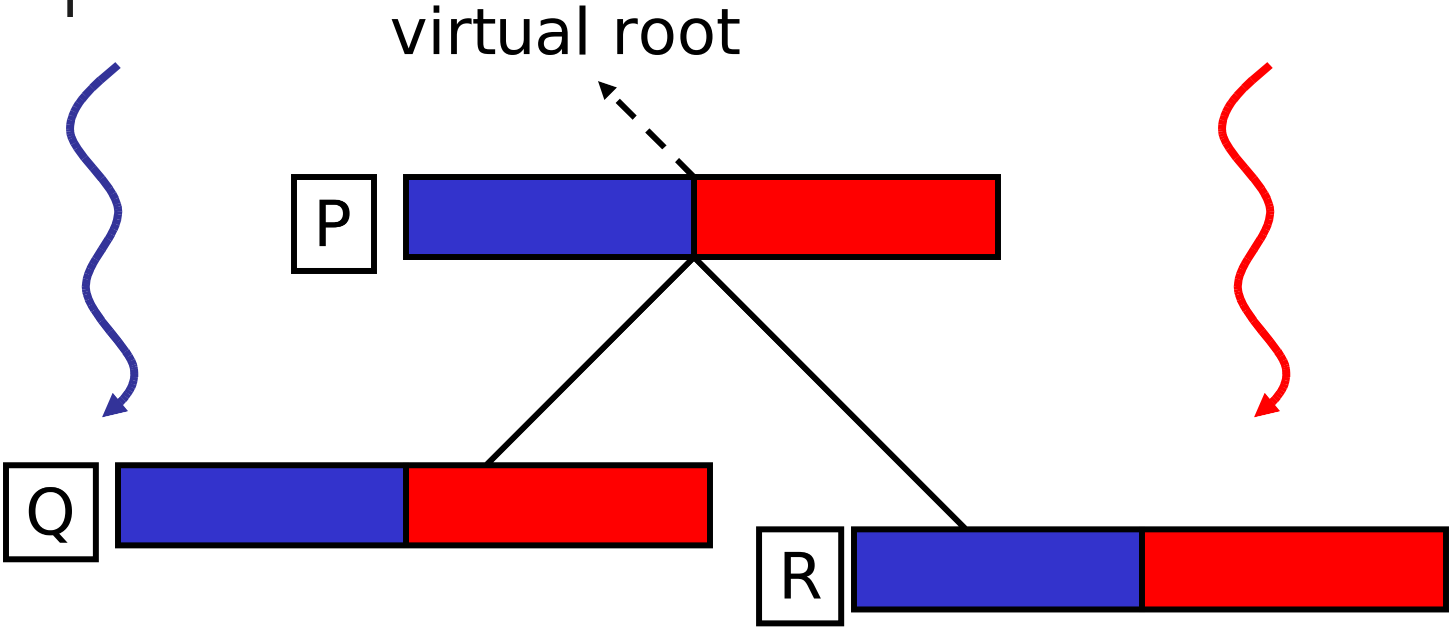
Loop Level Parallelism



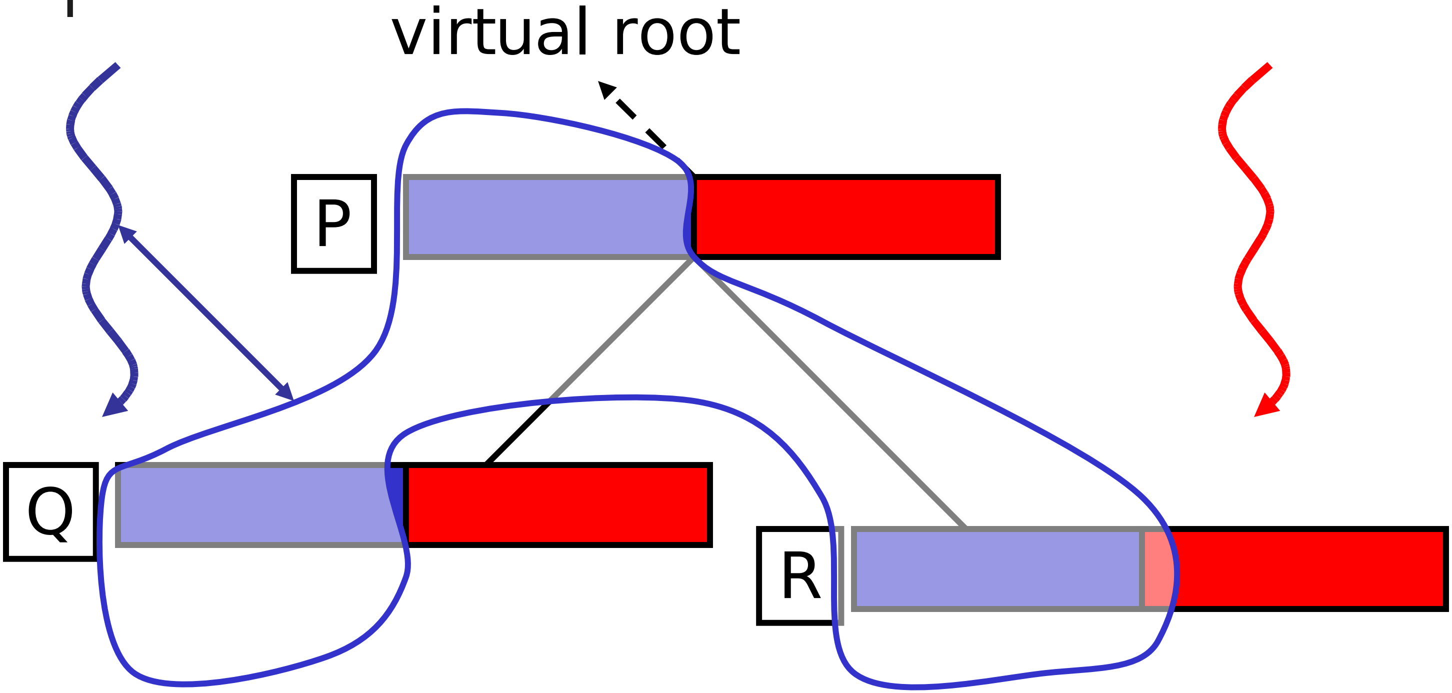
This operation uses ≥ 95% of total execution time !
→ simple fine-grained parallelization

P[i] = f(Q[i], R[i])

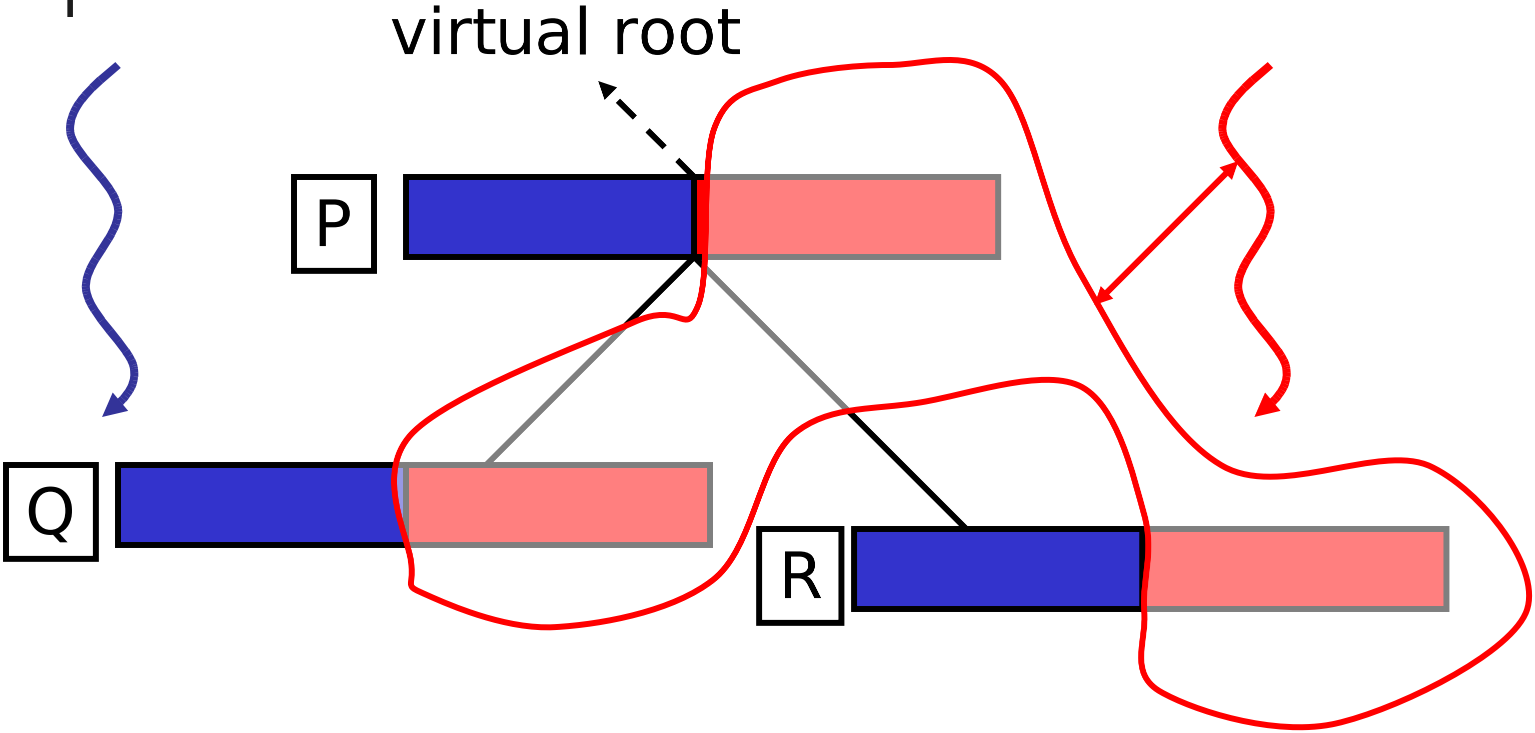
Loop Level Parallelism



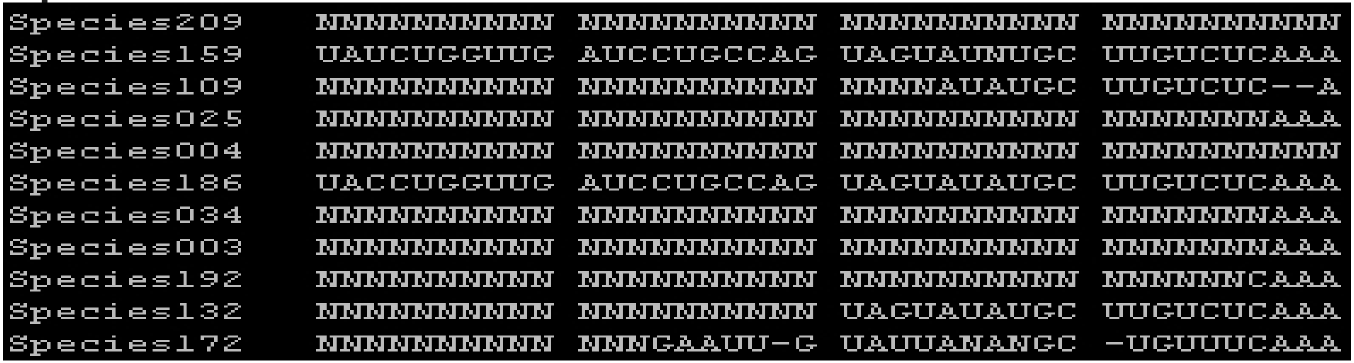
Loop Level Parallelism



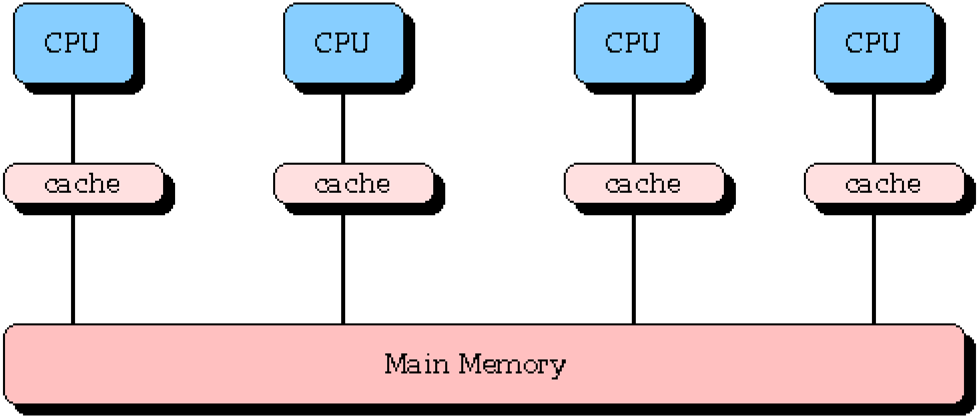
Loop Level Parallelism



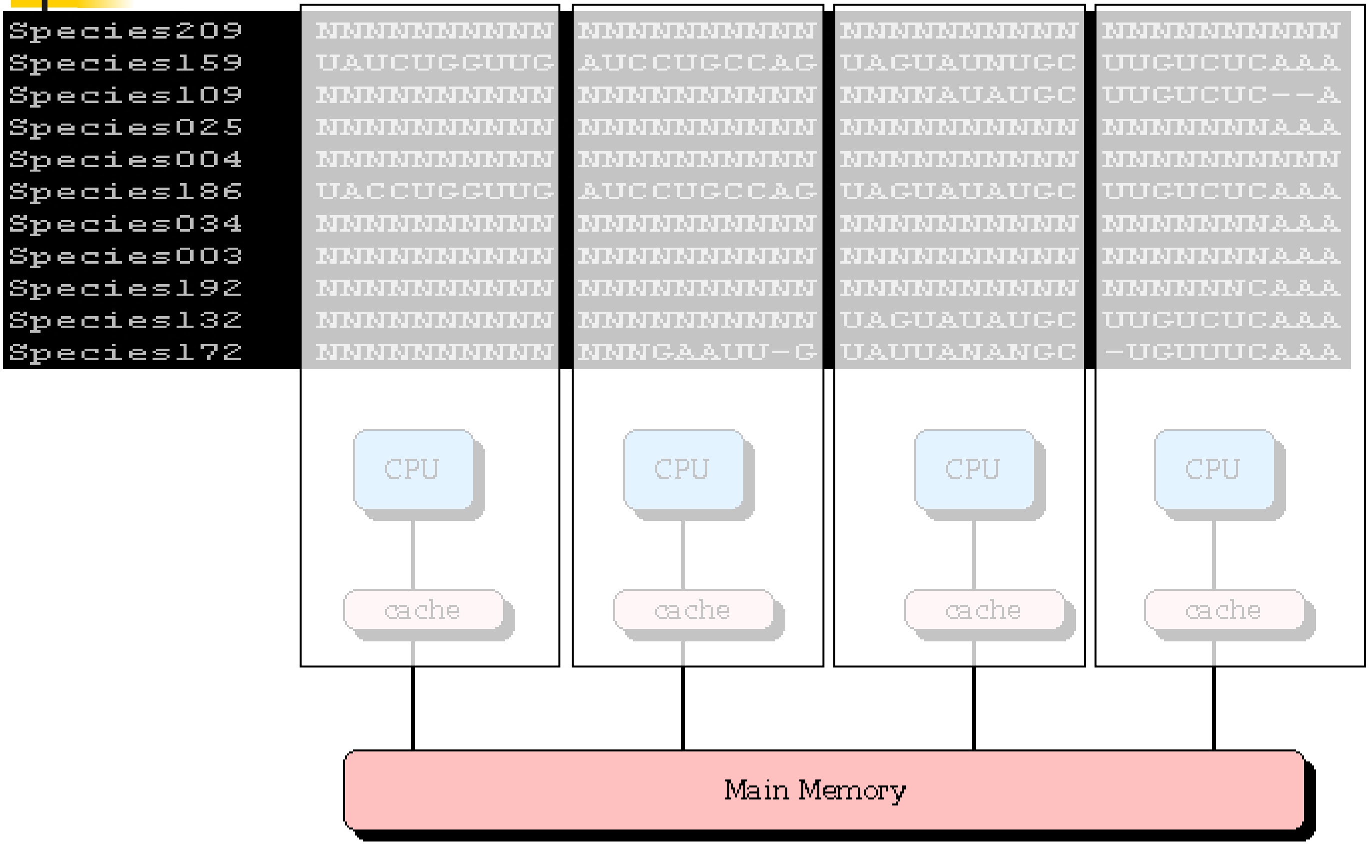
Loop-Level Parallelism



```
Species209     NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN
Species159     UAUCUGGUUG AUCCUGCCAG UAGUAUNUGC UUGUCUCAAA
Species109     NNNNNNNNNN NNNNNNNNNN NNNNAUAUGC UUGUCUC--A
Species025     NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNAAA
Species004     NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN
Species186     UACCUGGUUG AUCCUGCCAG UAGUAUAUGC UUGUCUCAAA
Species034     NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNAAA
Species003     NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNAAA
Species192     NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNCAAA
Species132     NNNNNNNNNN NNNNNNNNNN UAGUAUAUGC UUGUCUCAAA
Species172     NNNNNNNNNN NNNGAAUU-G UAUUANANGC -UGUUUCAAA
```



Loop-Level Parallelism



```
Species209 NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN
Species159 UAUCUGGUUG AUCCUGCCAG UAGUAUNUGC UUGUCUCAAA
Species109 NNNNNNNNNN NNNNNNNNNN NNNNAUAUGC UUGUCUC--A
Species025 NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNAAA
Species004 NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN
Species186 UACCUGGUUG AUCCUGCCAG UAGUAUAUGC UUGUCUCAAA
Species034 NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNAAA
Species003 NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNNAAA
Species192 NNNNNNNNNN NNNNNNNNNN NNNNNNNNNN NNNNNNCAAA
Species132 NNNNNNNNNN NNNNNNNNNN UAGUAUAUGC UUGUCUCAAA
Species172 NNNNNNNNNN NNNGAAUU-G UAUUANANGC -UGUUUCAAA
```

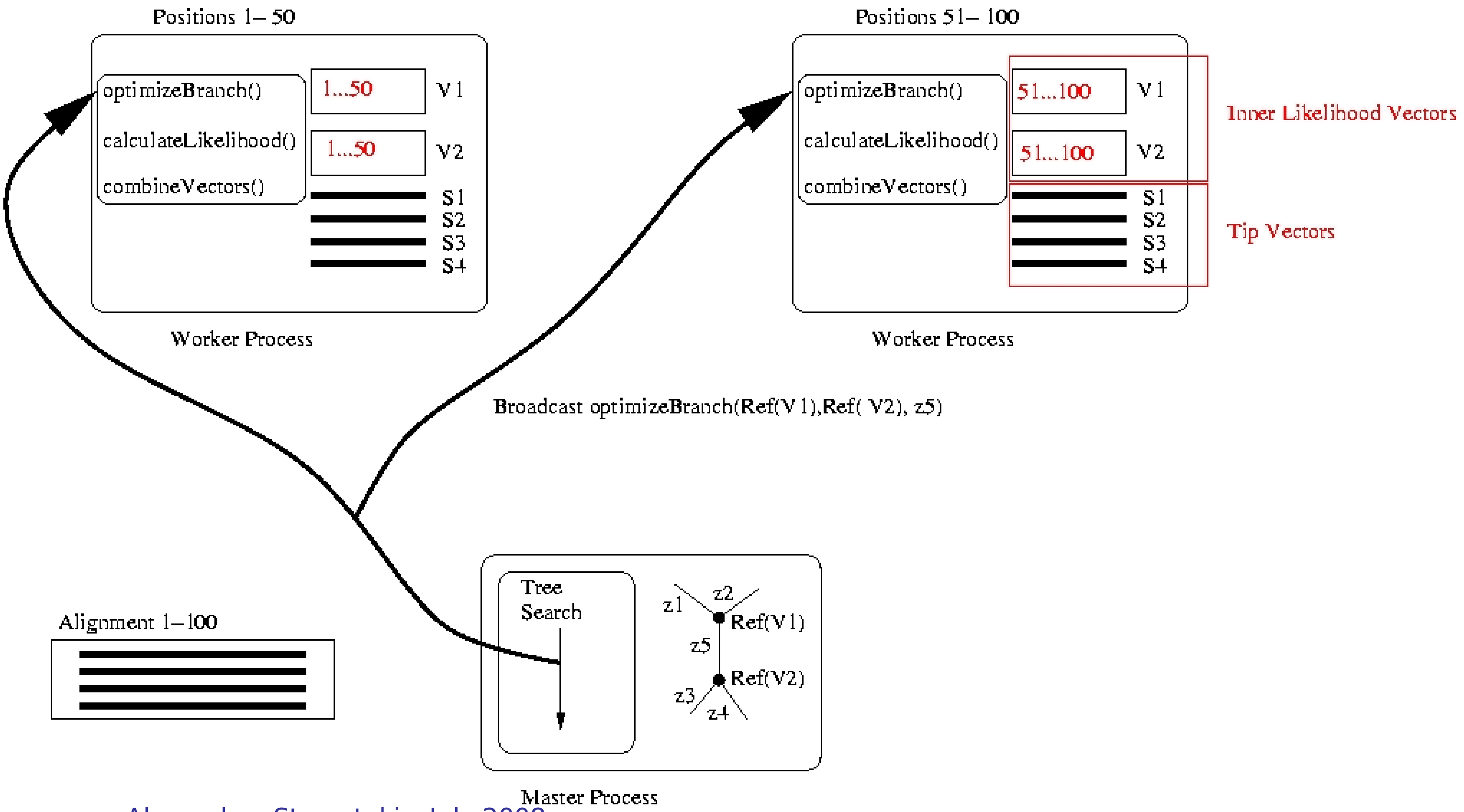
HPC for ML & Bayesian Phylogenetic Inference

- **Proof of Concept & Programming Techniques:**
  - **RAxML on a Graphics Processing Unit (completed)**
  - **RAxML on the IBM CELL & Playstation III (completed)**
- **Production Level Parallelizations:**
  - **RAxML with OpenMP (completed)**
  - **RAxML with MPI (completed)**
  - **RAxML on IBM BlueGene (in progress)**
  - **RAxML with Pthreads on Multi-Core Architectures (in progress)**

Orchestrating the Phylogenetic Likelihood Function on Massively Parallel Machines

- IBM BlueGene/L and BlueGene/P systems dominate the top 500 list www.top500.org
  - 1,024 slow CPUs per rack
  - 512 MB or 1 GB of memory per node
  - High performance interconnect
- Challenges:
  - Distribute tree data structure among CPUs
  - Exploit fast collective communication network

Loop-Level Parallelism on BlueGene



Orchestrating the Phylogenetic Likelihood Function on Current Parallel Architectures

- Handle long memory-intensive datasets
- Processes/threads on cores compete for memory access bandwidth → memory gap problem
- Which is the best parallel programming paradigm/language for ML in terms of
  - Efficiency
  - Usability
  - Portability
  - Programming overhead
  - Program Complexity
- Which is the best multi-core architecture for ML (RAxML)?
- **Integrate all concepts into one piece of code that scales**
  - **From 2 cores up to 1,024 CPUs**
  - **On shared & distributed memory machines**

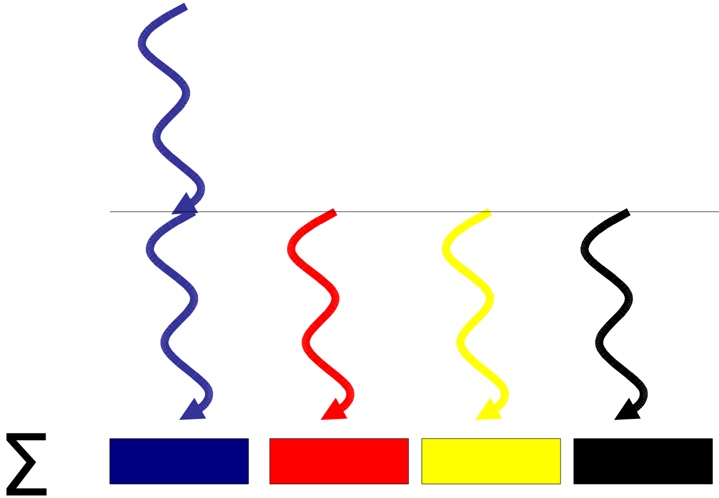
Programming Paradigms
MPI versus OpenMP versus Pthreads

- MPI
  - low level distributed memory programming
  - significant programming overhead (2 weeks)
  - distributed memory model → no joint view of memory space by all processors
  - not easy to compile & install → sys admin required
- OpenMP
  - high level shared memory programming
  - low programming overhead
  - no control over the machine/parallelization details
  - numerical & performance problems due to lack of control
  - not easy to compile & install → specialized compiler required
- Pthreads
  - low level shared memory programming library
  - significant programming overhead (4 weeks)
  - full control over the machine
  - easy to compile → it is starting to get used

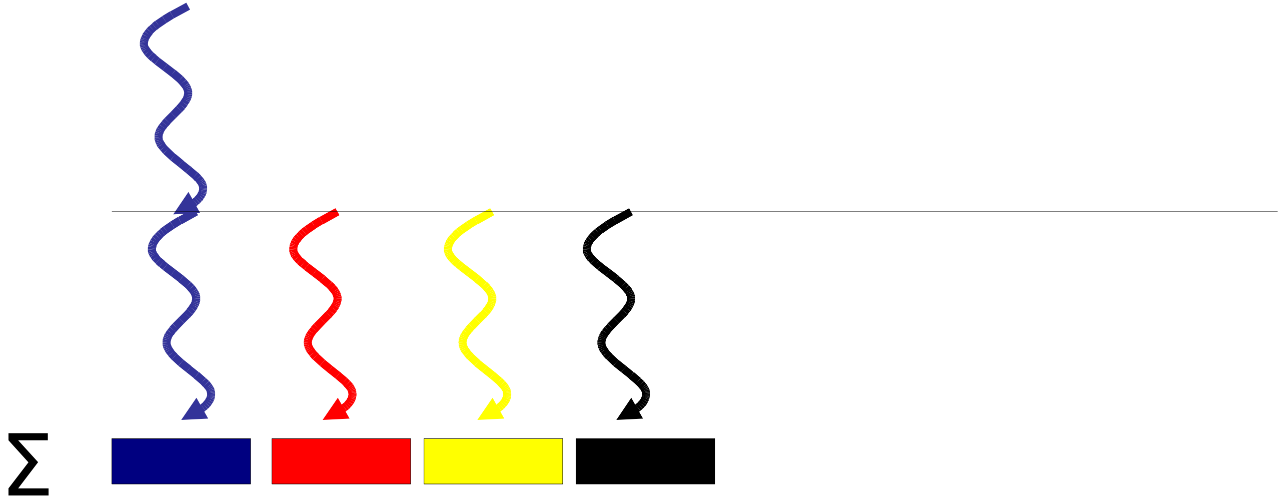
Problems with OpenMP

- Special OpenMP-enabled compiler required → non-expert users will not exploit parallelism
- Reduction operations (see next slide) non-deterministic → numerical operations that should yield identical results yield different results
- Fork-Join model → synchronization required for updating every single likelihood vector
- What does the OpenMP compiler (Intel icc) do?
  - We don't know
  - It automatically fixed a bug and yielded very bad performance
- OpenMP is a little bit like Windows: one does not know what it does and can not influence what is happening
- But it took only 5 hours to parallelize MrBayes from scratch

OpenMP Reduction Operation

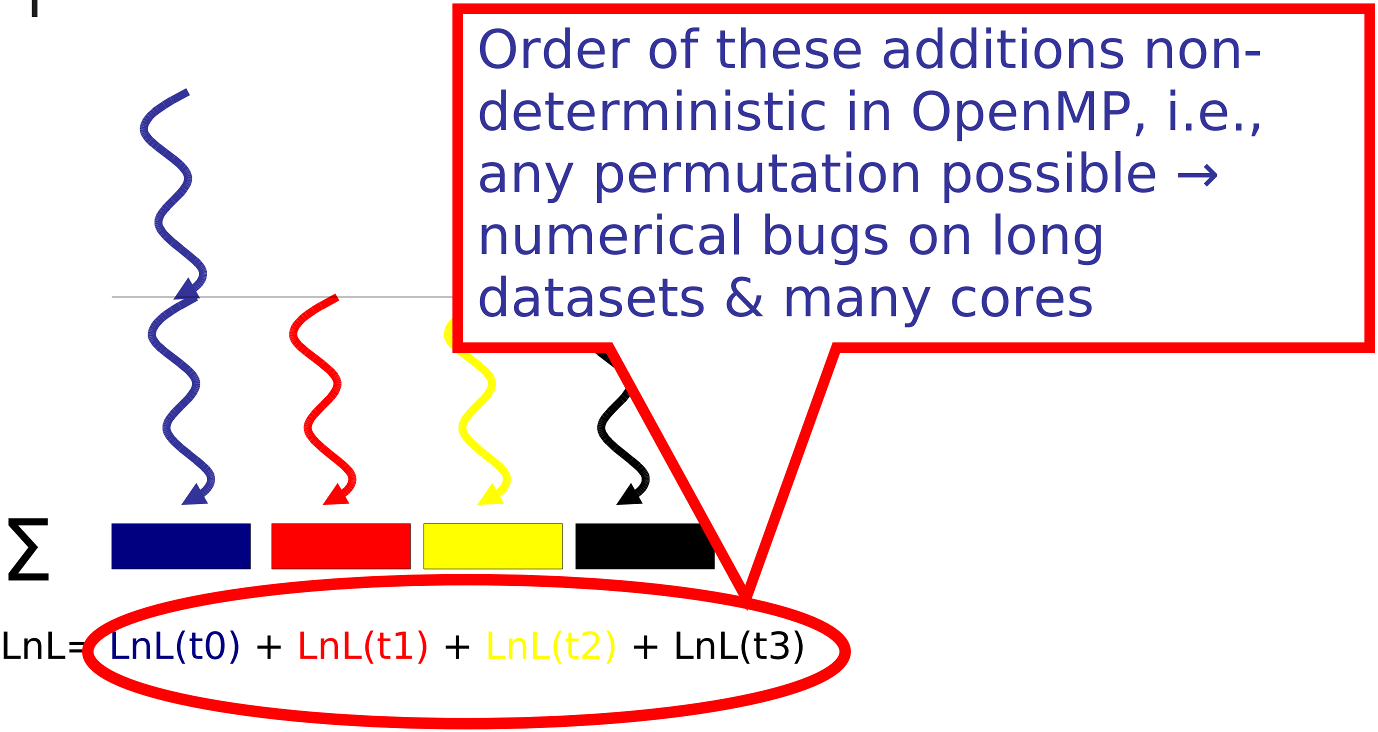


OpenMP Reduction Operation



LnL= LnL(t0) + LnL(t1) + LnL(t2) + LnL(t3) add per-thread sums

OpenMP Reduction Operation



Order of these additions non-deterministic in OpenMP, i.e., any permutation possible → numerical bugs on long datasets & many cores

LnL= LnL(t0) + LnL(t1) + LnL(t2) + LnL(t3)

Thread Pinning

- **Thread pinning/mapping** to cores has **significant impact** on performance if less threads are executed than cores are available!
- This happens if 2 threads are started on the same socket instead of different ones
- Can cause up to 50% run time differences among various mappings of, e.g., 4 threads onto 8 cores
- We compute speedups based on optimal assignments for each configuration

Test Systems

- IBM BlueGene/L distributed memory
  - 1,024 CPUs
- SGI Altix 4700 (LRZ Munich) shared memory
  - Total 9,728 cores, we used up to 256 cores
  - 39 Terabyte of main memory
- Infiniband Cluster
  - 32 4-way SMPs (single cores)
  - Infiniband interconnect (low latency)
- AMD Barcelona
  - 2-way quad core (8 cores)
- Intel Clovertown
  - 2-way quad-core (8 cores)
- Sun x4600
  - 8-way dual core (16 cores)

Test Datasets

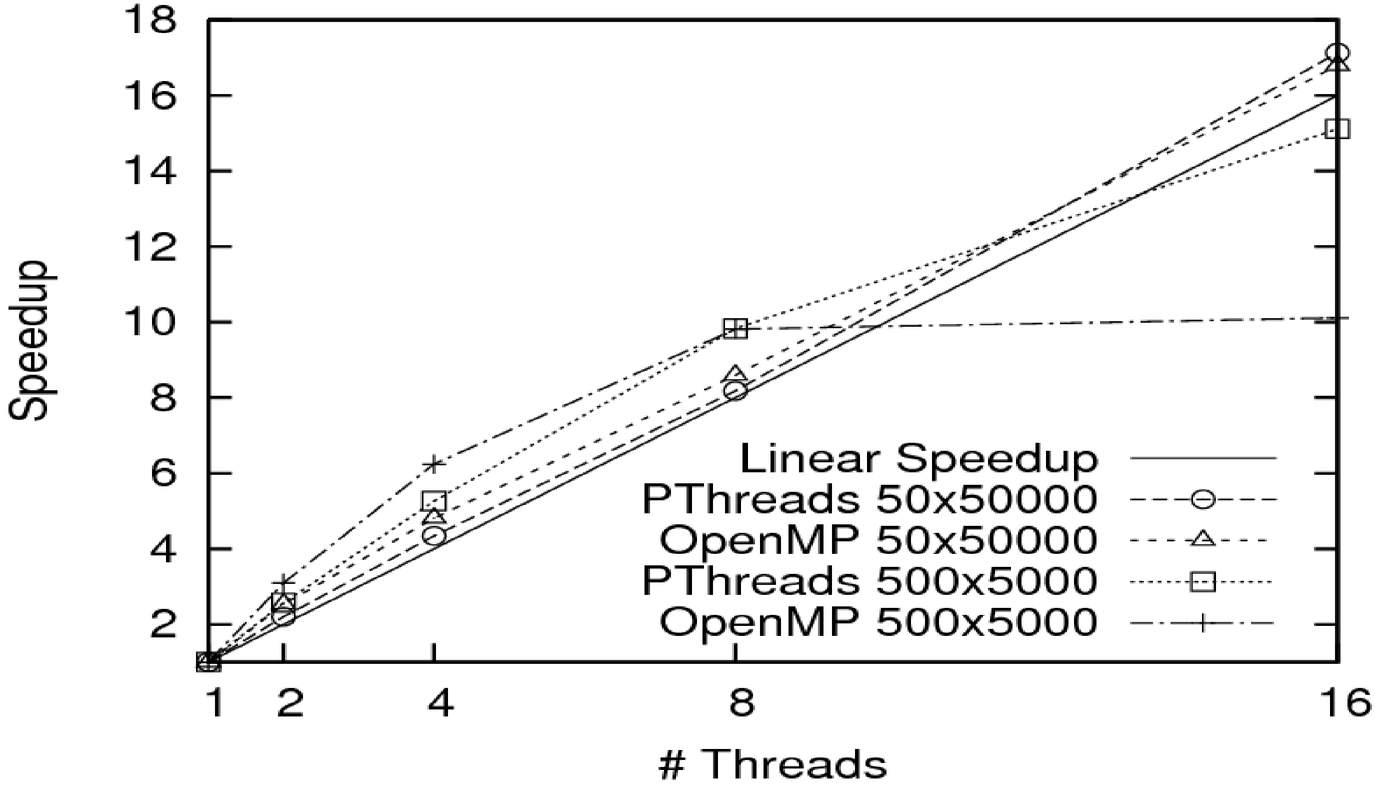
- 50_5000, 50 taxa, 5,000 bp, (3,066 patterns)
- 50_50000, 50 taxa, 50,000 bp (23,285 patterns)
- 50_500000, 50 taxa, 500,000 bp (216,025 patterns)
- 250_500000, 250 taxa, 500,000 bp (403,581 patterns)
- 500_5000, 500 taxa, 5,000 bp (3,829 patterns)

Test Datasets

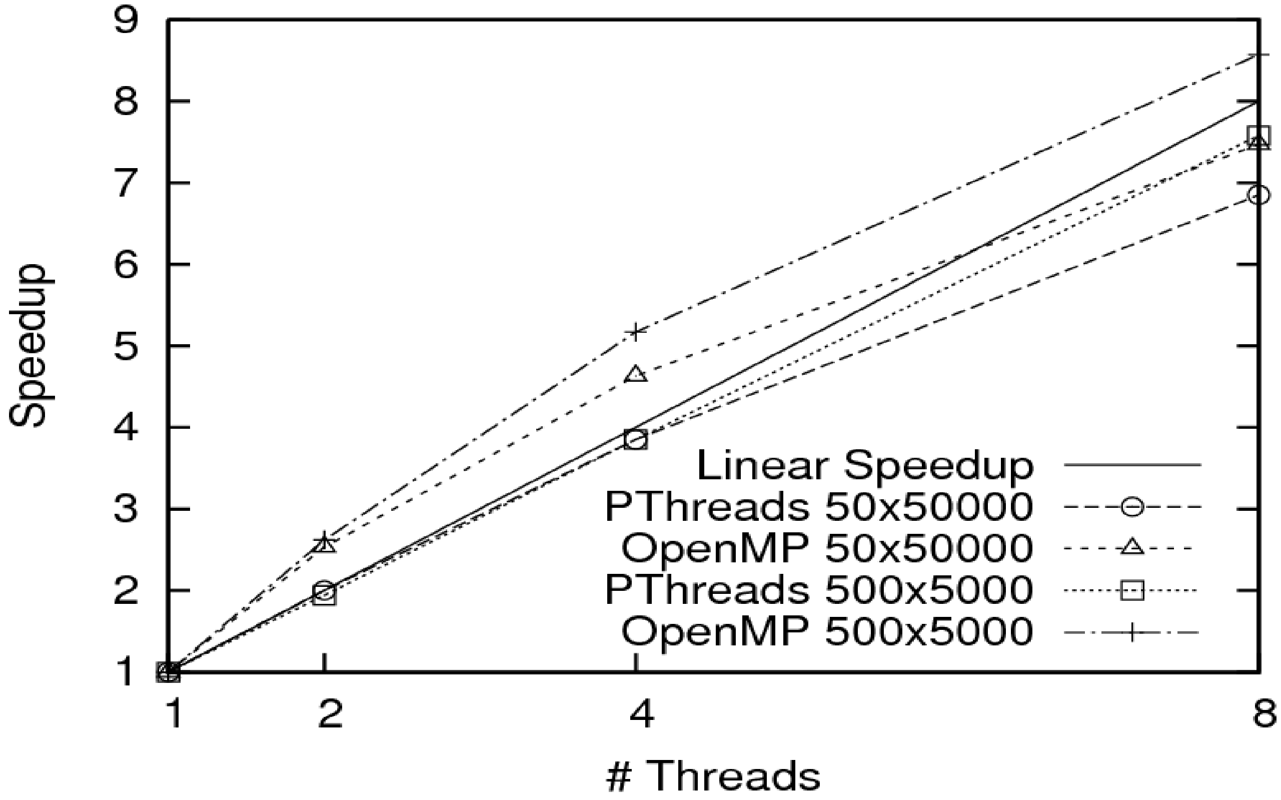
- 50_5000, 50 taxa, 5,000 bp, (3,066 patterns)
- 50_50000, 50 taxa, 50,000 bp (23,285 patterns)
- 50_50000 patterns)
- 250_500000, 250 taxa, 500,000 bp (403,581 patterns)
- 500_5000, 500 taxa, 5,000 bp (3,829 patterns)

Computation to communication ratio about 100 times less favourable

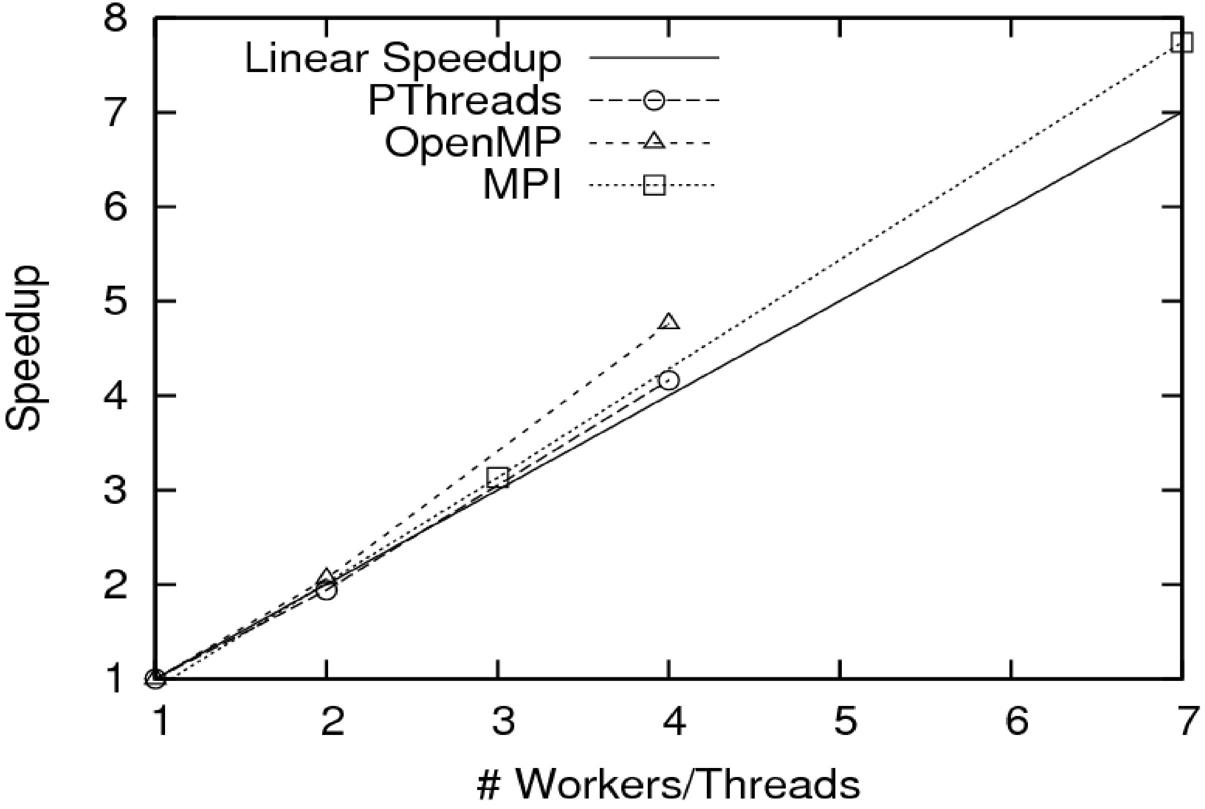
Sun x4600: OpenMP versus Pthreads



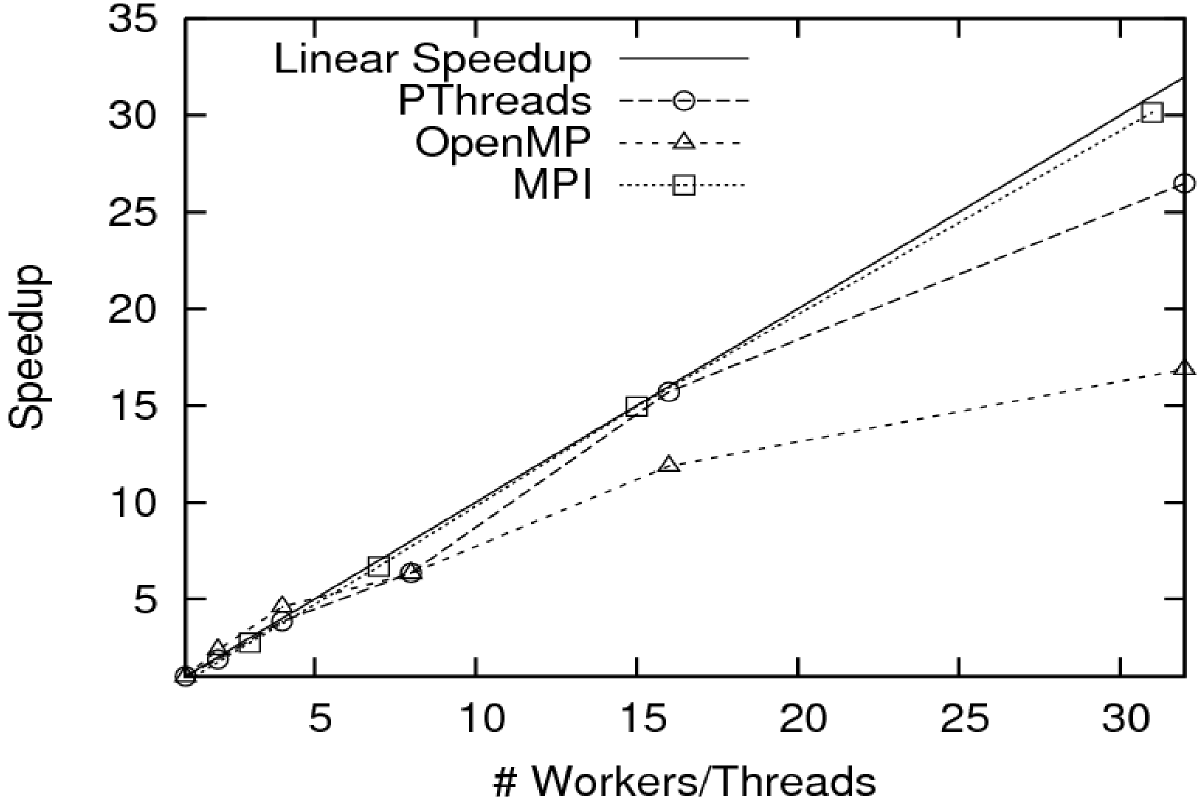
AMD Barcelona: OpenMP versus Pthreads



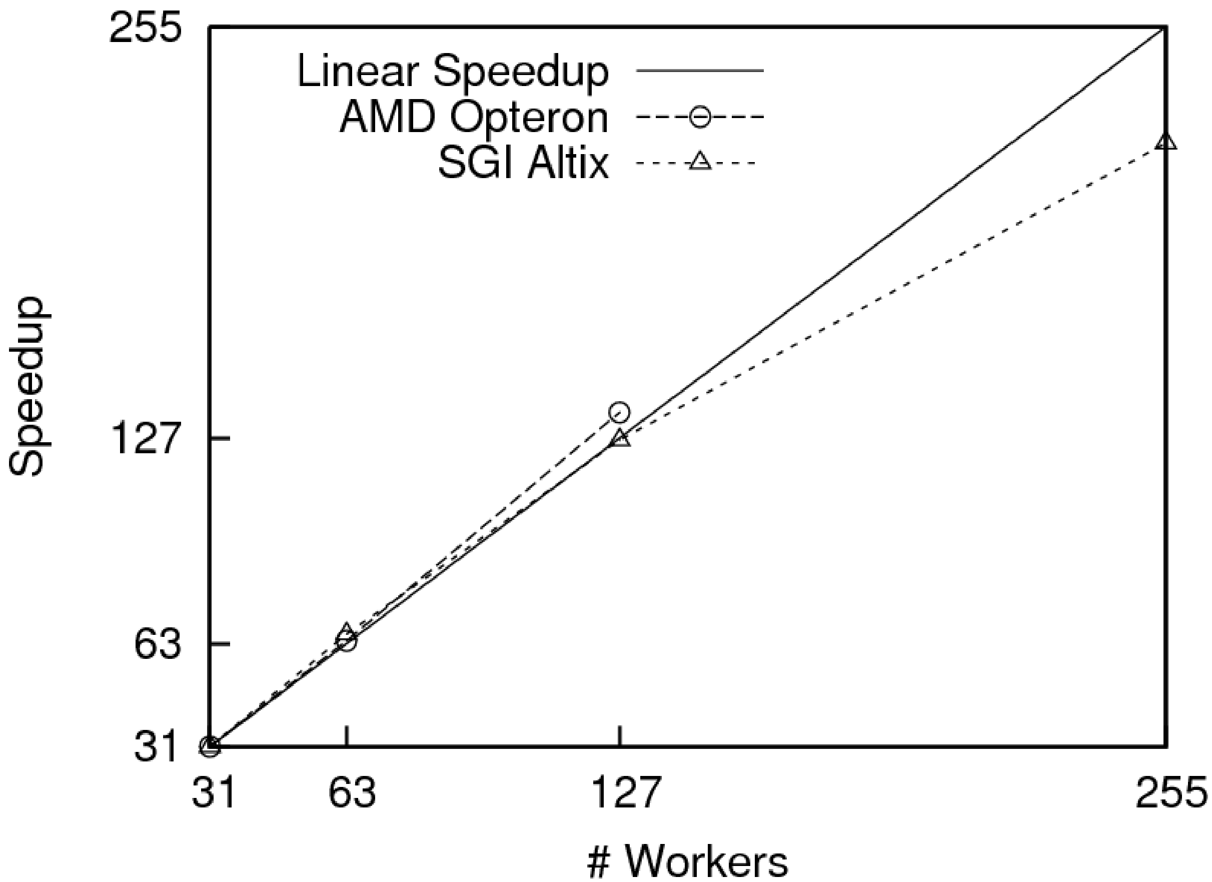
Infiniband Cluster
d50_50000 MPI vs. Pthreads vs. OpenMP



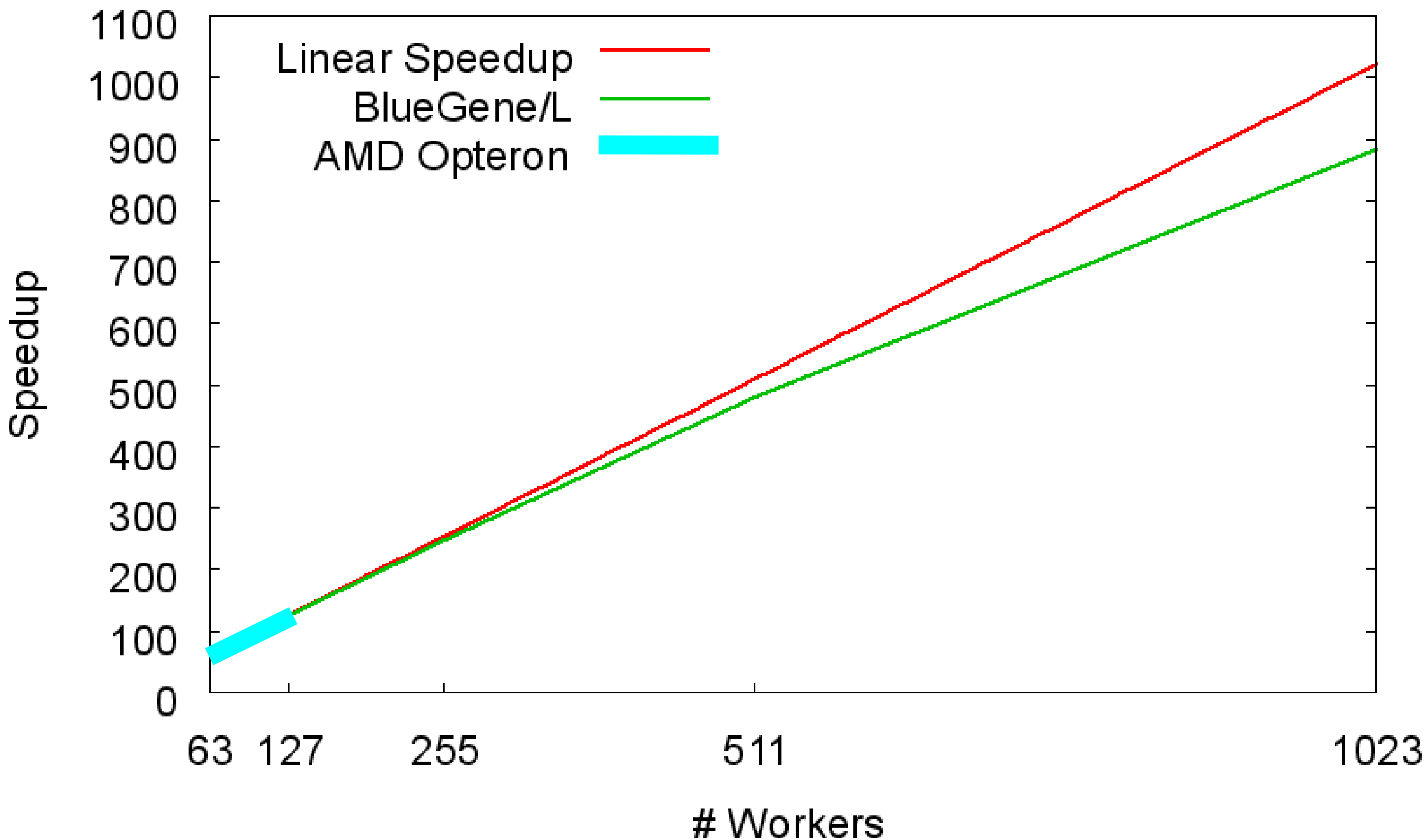
SGI Altix 4700 d50_50000 MPI vs. Pthreads vs. OpenMP



SGI ALTIX & Infiniband Cluster Dataset d250_500000 MPI



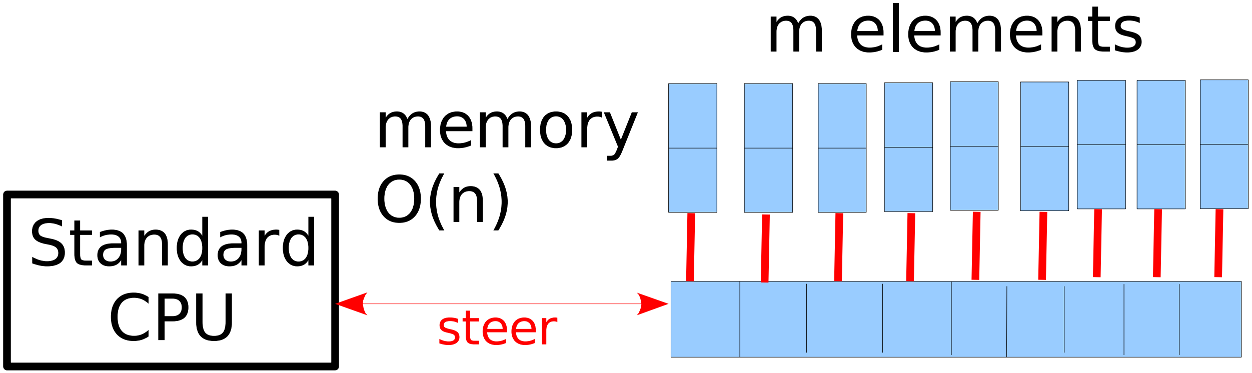
IBM BlueGene/L & Infiniband Cluster d250_500000



Programming Paradigms: Conclusion

- Use MPI and Pthreads:
  - More programming overhead
  - More control
  - Enforce data locality with Pthreads for NUMA and MPI for distributed memory machines
  - Portability (BlueGene, clusters of SMPs)
  - Implement this single complicated parallelization once, use generic interface to access either Pthreads or MPI communication mechanisms
  - Pthreads version can be further optimized
  - Since easy-to-compile Pthreads-based release in January 2008 → Biologists actually use it

The Ideal Architecture for Phylogenetic Inference

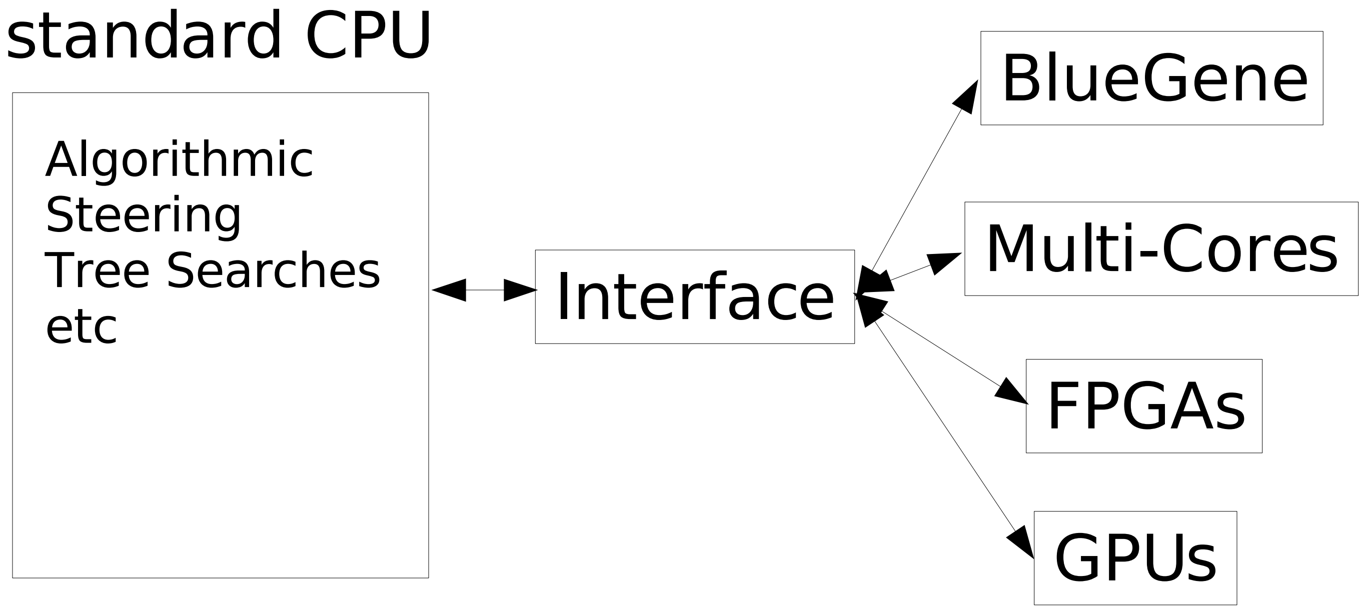


Orangutan	A A C G T T T T -
Gorilla	A A G G T T T - -
Chimp	A - G G T T T T -
Homo Sapiens	A G G A T T T T T

n

m

The Ideal World:
Basic Phylogenetic Subroutines



standard CPU

Algorithmic
Steering
Tree Searches
etc

Interface

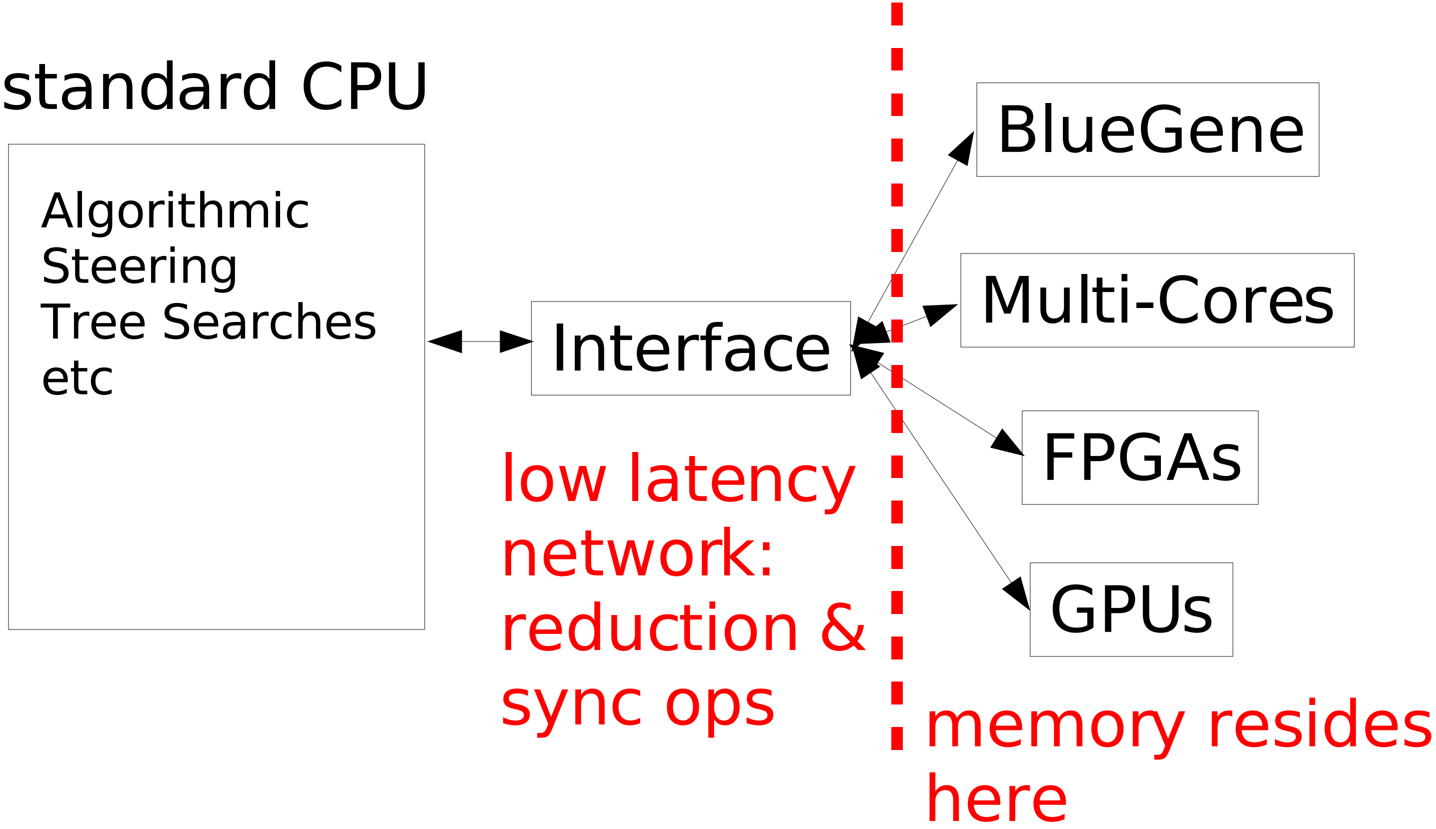
BlueGene

Multi-Cores

FPGAs

GPUs

The Ideal World: Basic Phylogenetic Subroutines



standard CPU

Algorithmic
Steering
Tree Searches
etc

Interface

low latency network: reduction & sync ops

BlueGene

Multi-Cores

FPGAs

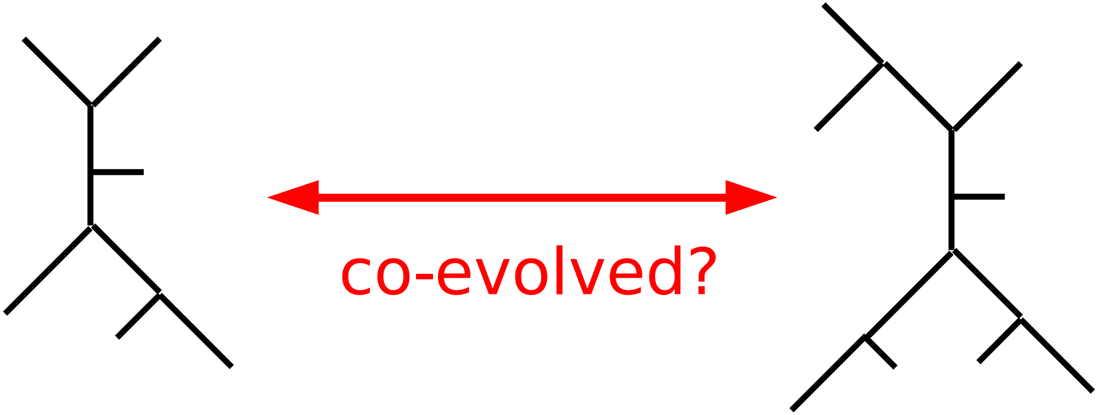
GPUs

memory resides here

Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Impact
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

A brief Detour:
Host-Parasite Co-Evolution



Host-Tree
(Mammals)

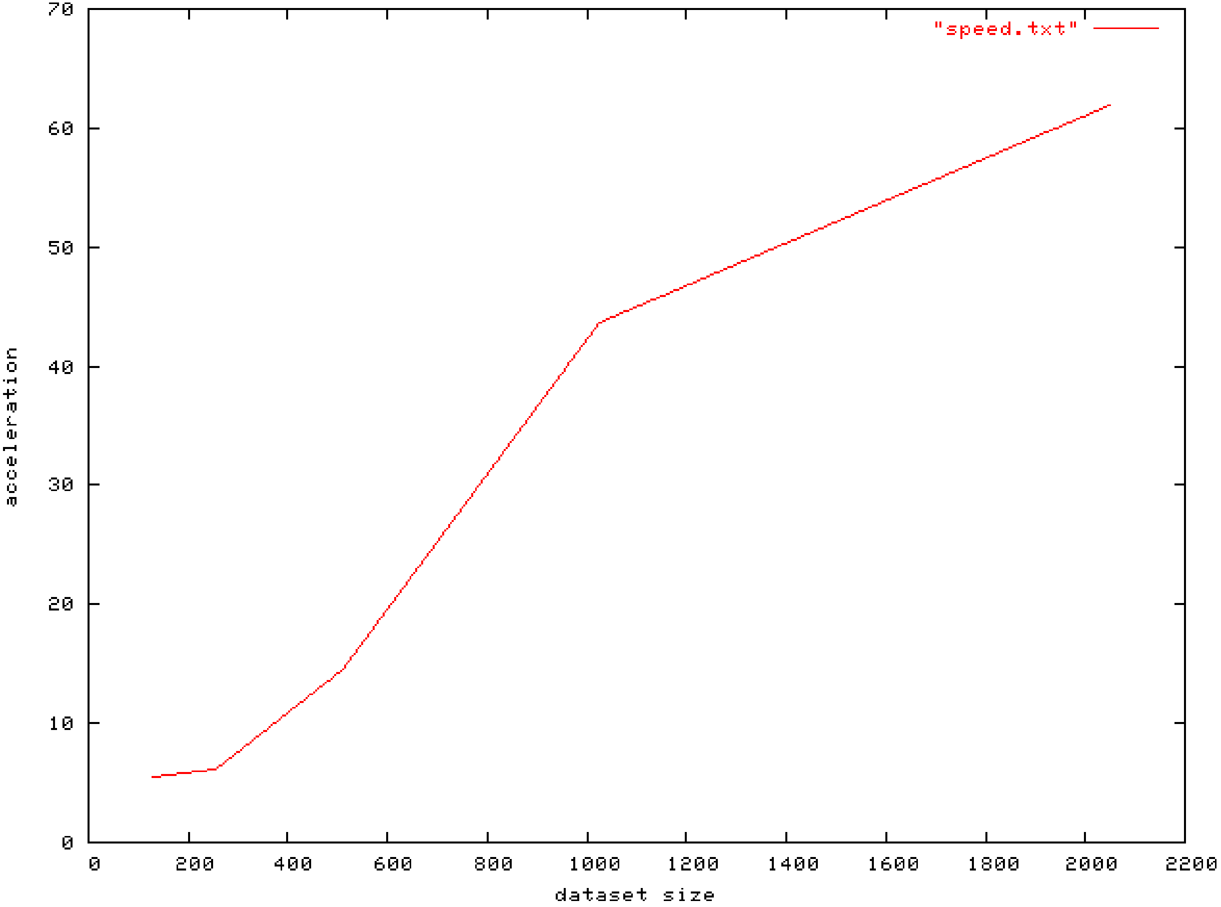
Parasite-Tree
(Lice)

co-evolved?

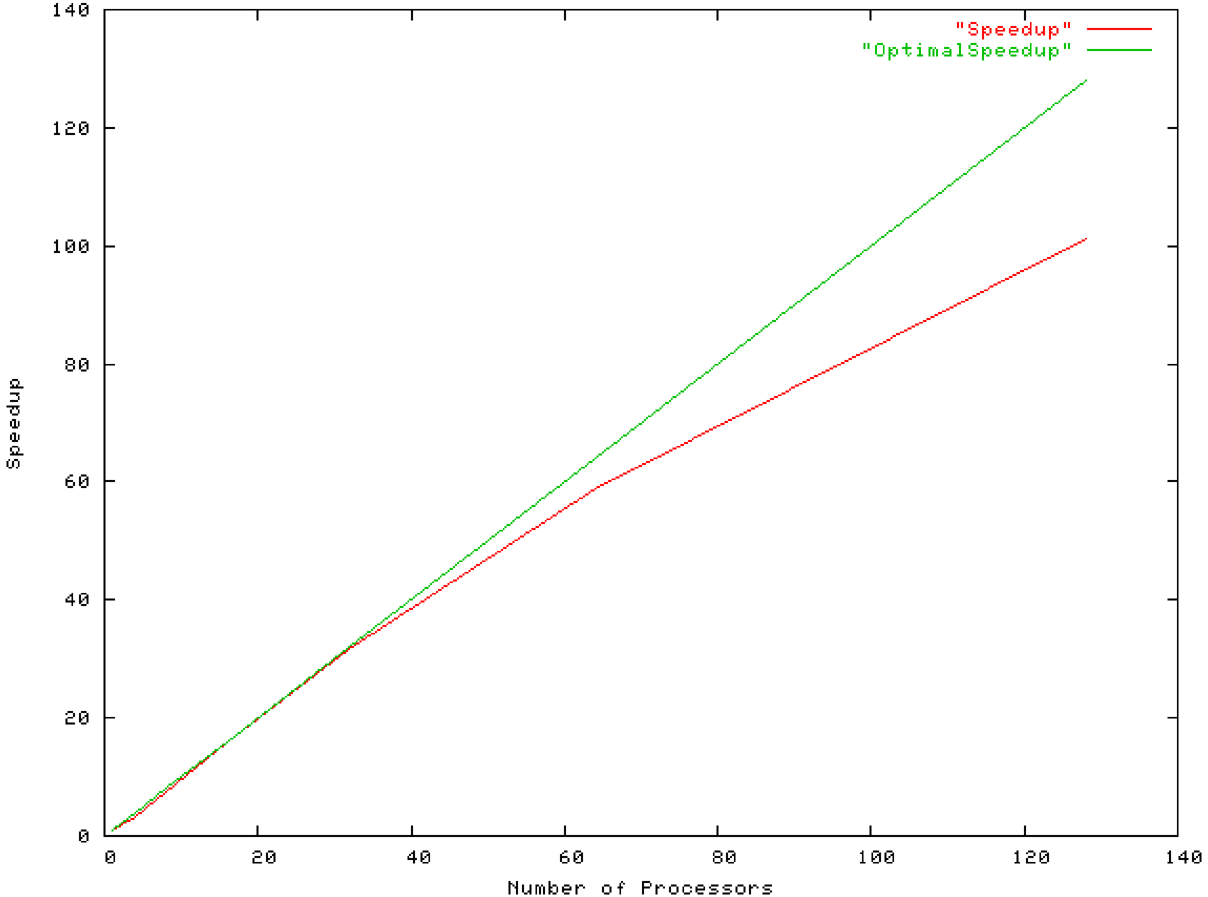
What can HPC do for Bioinformatics? Axelerated Parafit

- "Parafit: statistical test of co-evolution", *Systematic Biology* 2003
- M. Göker (Tübingen): Need for a faster implementation
- AxParafit (Axelerated Parafit)
  - Application of standard HPC techniques: sequential speedup up to factor 67
  - MPI-parallelization
  - Open-Source Code
- Largest co-phylogenetic study to date conducted within 8 minutes instead of 4 weeks:
  - Alexandros Stamatakis, Alexander Auch, Jan Meier-Kolthoff, Markus Göker: "AxPcoords & Parallel AxParafit: Statistical Co-Phylogenetic Analyses on Thousands of Taxa". In *BMC Bioinformatics*, 8:405, 2007.
  - 245 downloads from distinct IPs since October 2007
- Current work
  - Analysis of complete NCBI data
  - SwissGrid-based Web-server

AxParafit: Sequential Performance



AxParafit: Parallel Performance



Outline

- Introduction
  - Computation of Phylogenies
  - Maximum Likelihood
  - Web Servers
- Computing ML Trees:
  - Search Algorithms
  - Optimization of the ML function
  - Model Issues
  - Parallelism
- Related Topics
- Summary of Future Challenges

Future Challenges Summary

- **HPC:**
  - Handle growing alignments
  - Exploit multi-core architectures
- **Algorithms:**
  - Simultaneously compute good trees & support values
  - Phylogenetic classification
- **Models:**
  - Model multi-gene alignments
  - Use structure information
- **Algorithms & Models:**
  - Simultaneous alignment and tree building
- **Vision:** Simultaneous computation of:
  - Alignment
  - Tree
  - Ancestral States
  - Support Values

Acknowledgments

Michael Ott
PhD student
at TUM

PhD Positions in Munich

- Two PhD positions in Phyloinformatics available at Exelixis Lab (LMU Munich)

Thank you for your Attention !

Mount Psiloritis, Crete