Introduction to Bioinformatics for Computer Scientists

Lecture 9

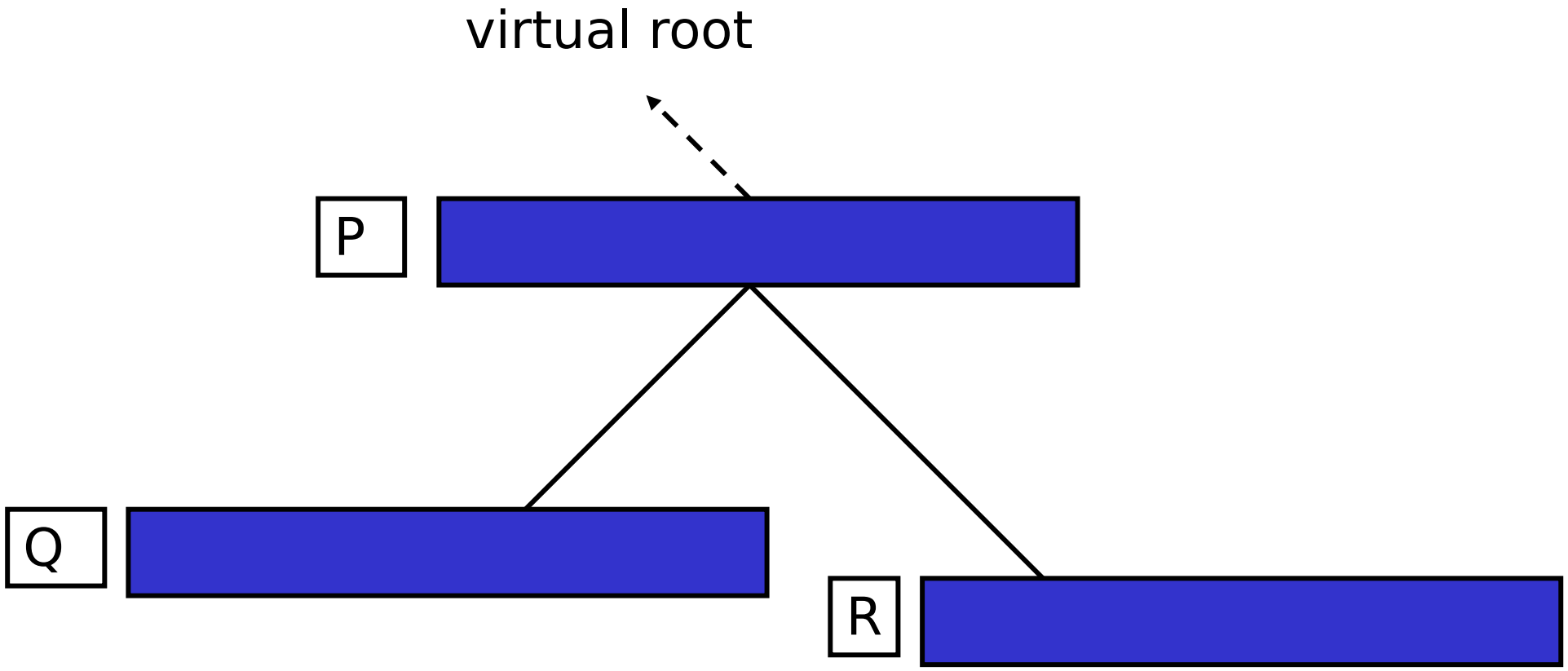
Outline

- Last time:
  - What is hidden in *P(t)* – what do the models look like?
  - How to compute the Maximum Likelihood score on a tree?
  - Advanced substitution models
  - Efficiently computing the Likelihood on trees on a single processor!

Outline

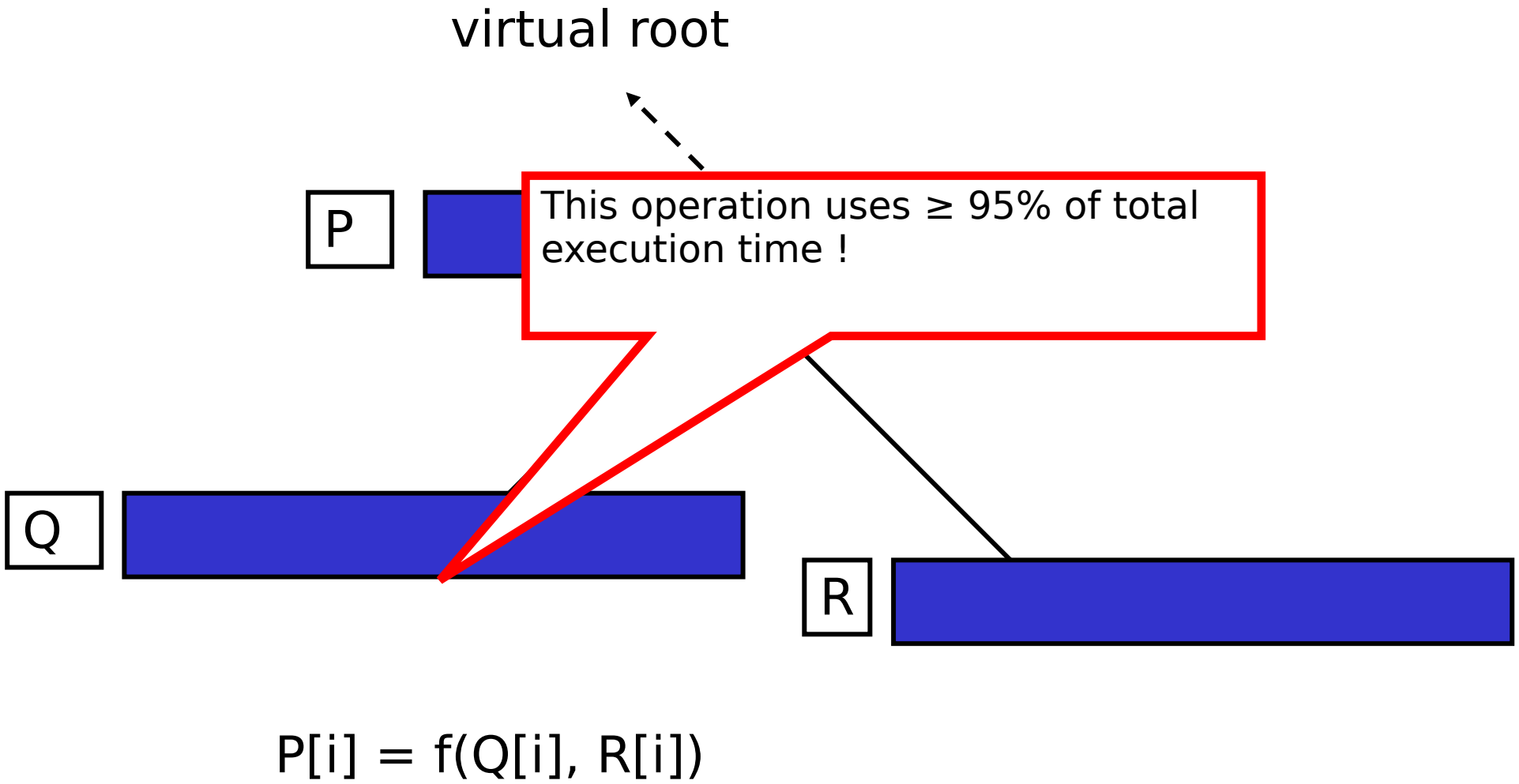
- Last time:
  - What is hidden in *P(t)* – what do the models look like?
  - How to compute the Maximum Likelihood score on a tree?
  - Advanced substitution models
  - Efficiently computing the Likelihood on trees on a single processor!
- Today
  - **Efficiently computing the likelihood in parallel**
  - Bayesien Inference and Markov Chain Monte Carlo

Loop Level Parallelism



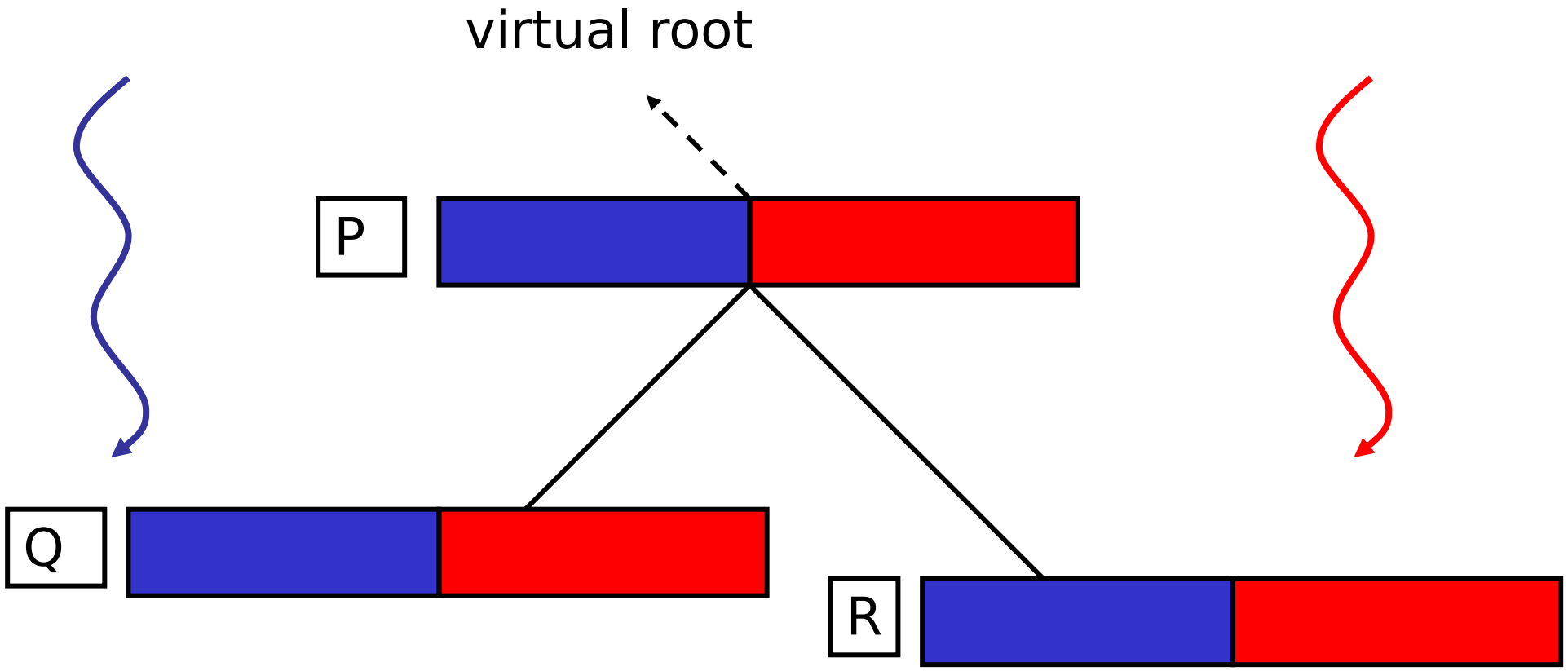
P[i] = f(Q[i], R[i])

Loop Level Parallelism

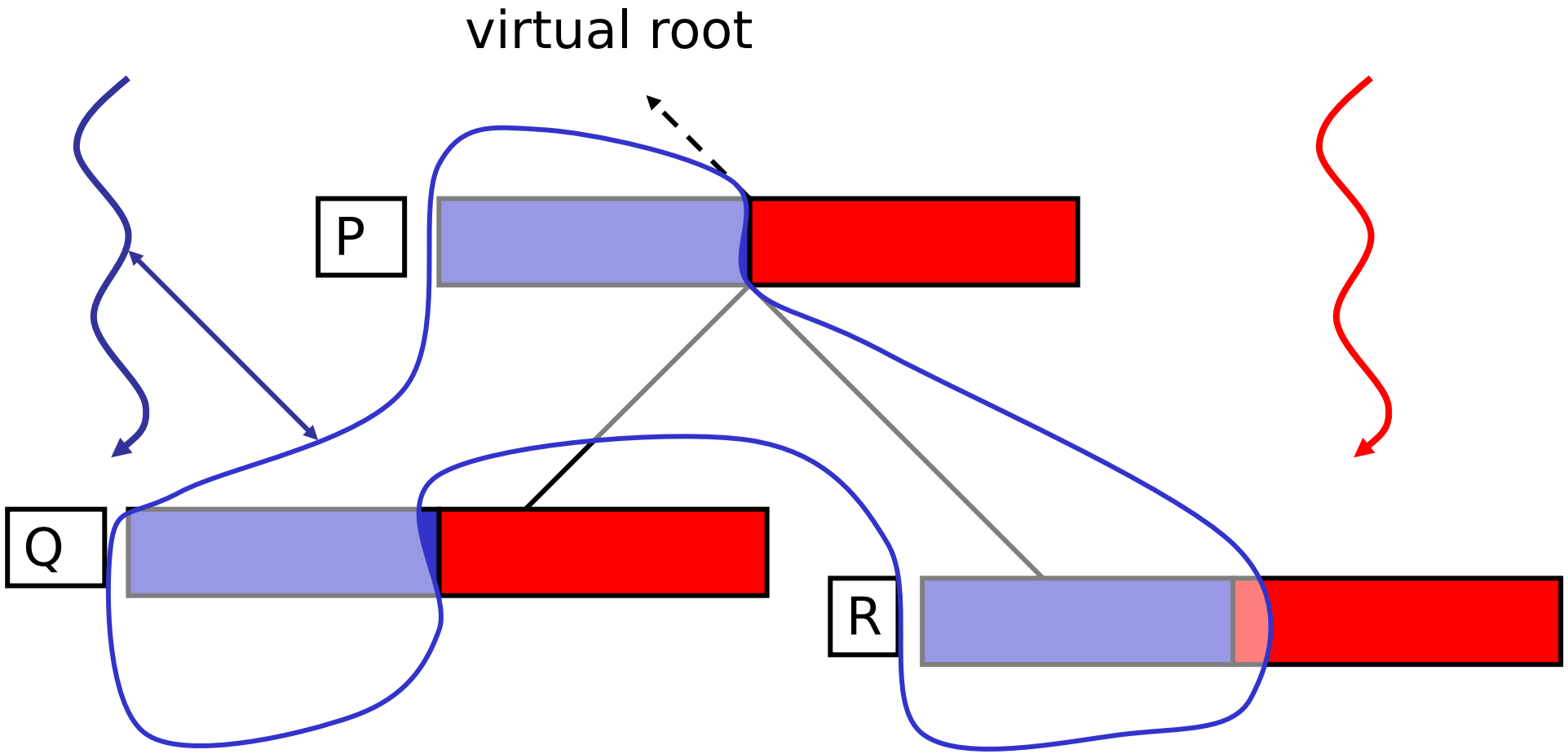


P[i] = f(Q[i], R[i])

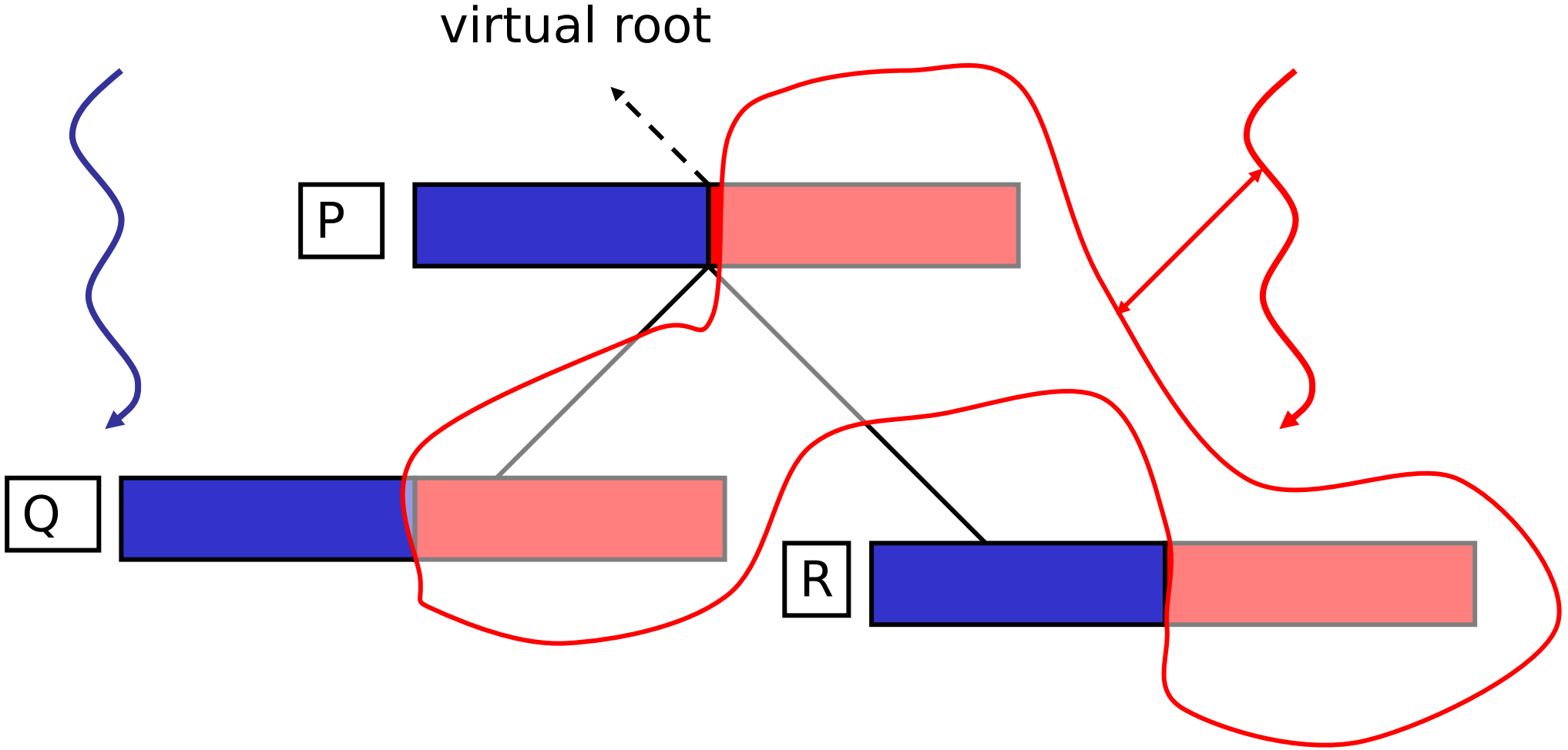
Loop Level Parallelism



Loop Level Parallelism

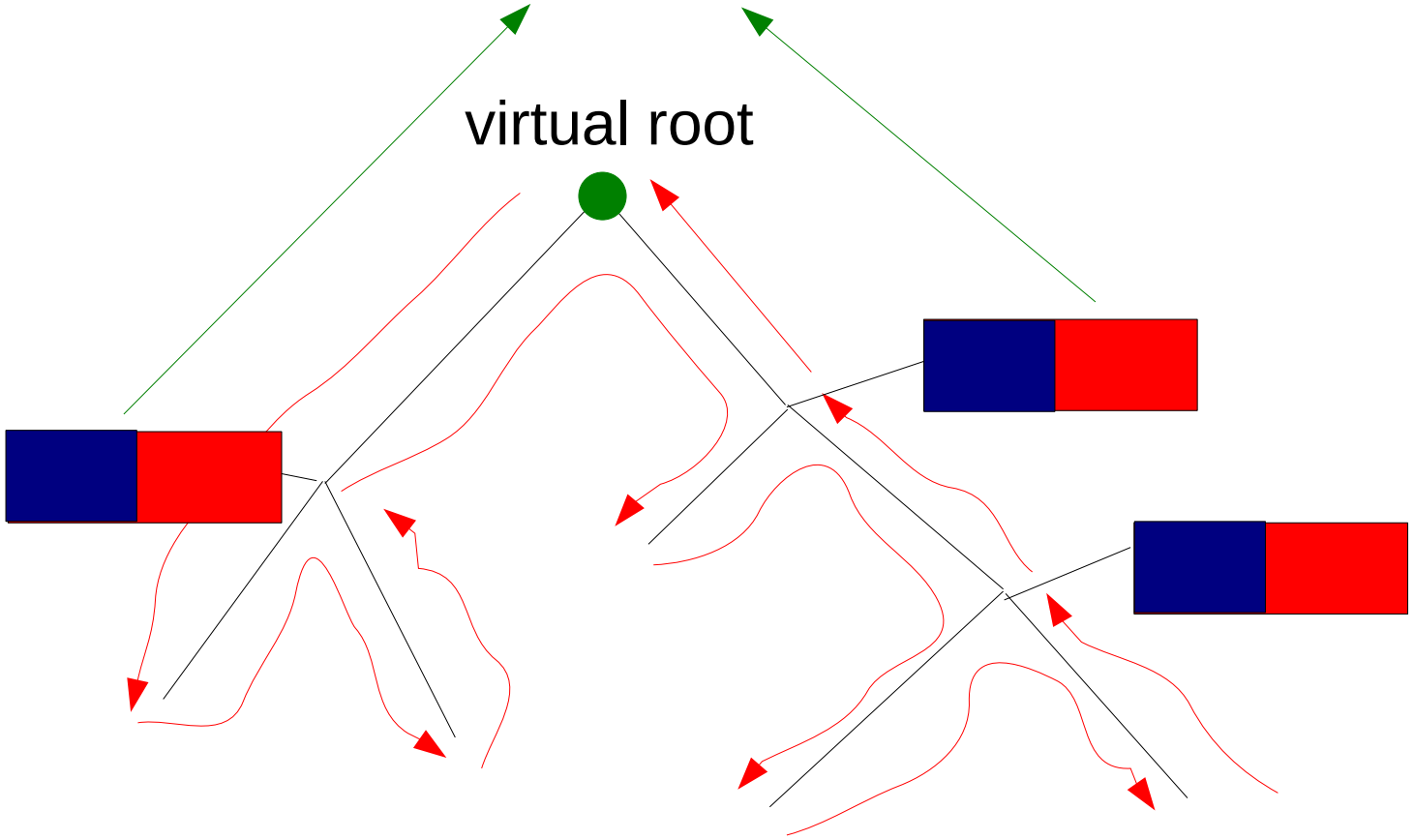


Loop Level Parallelism

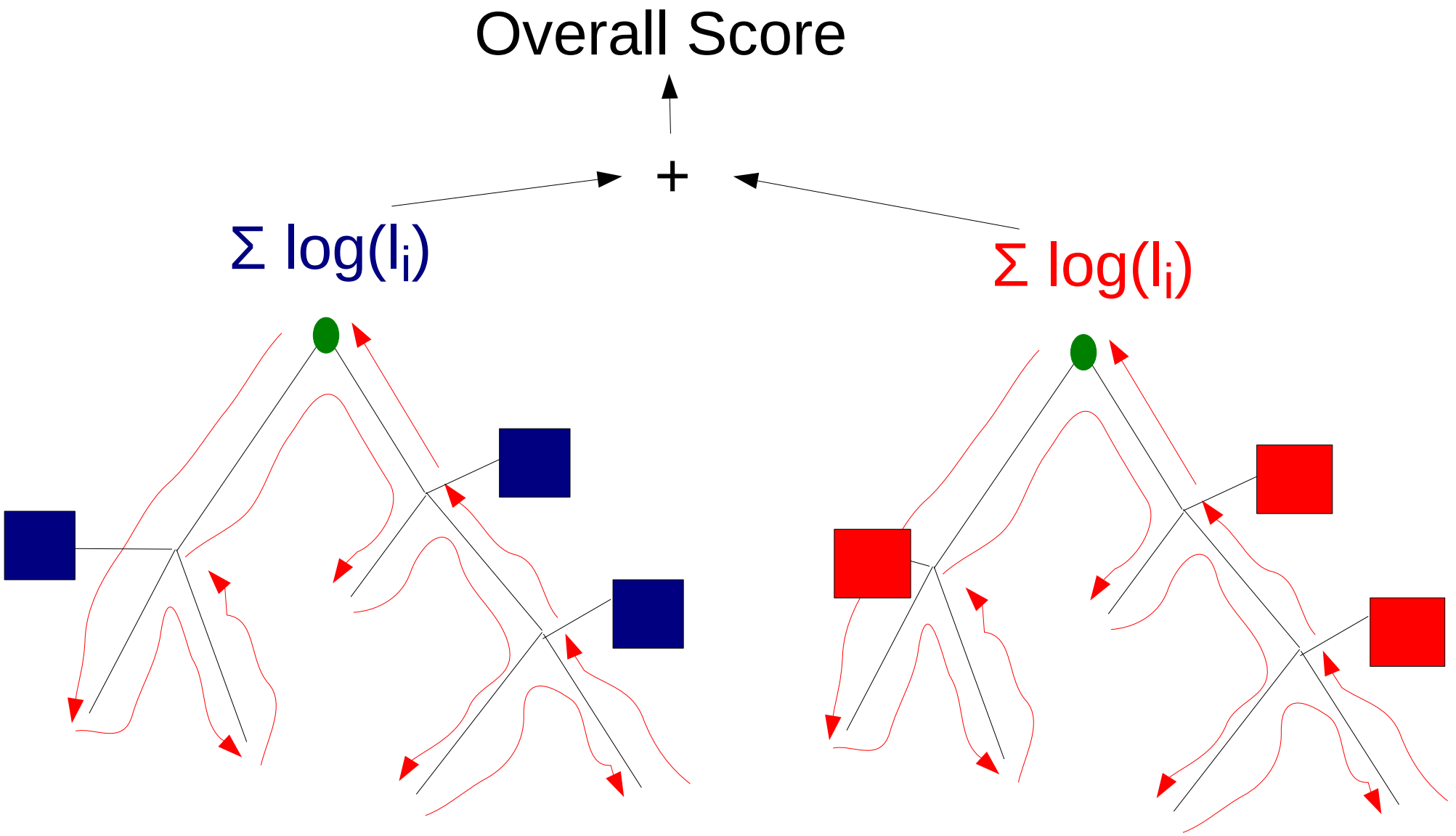


Parallel Post-order Traversal

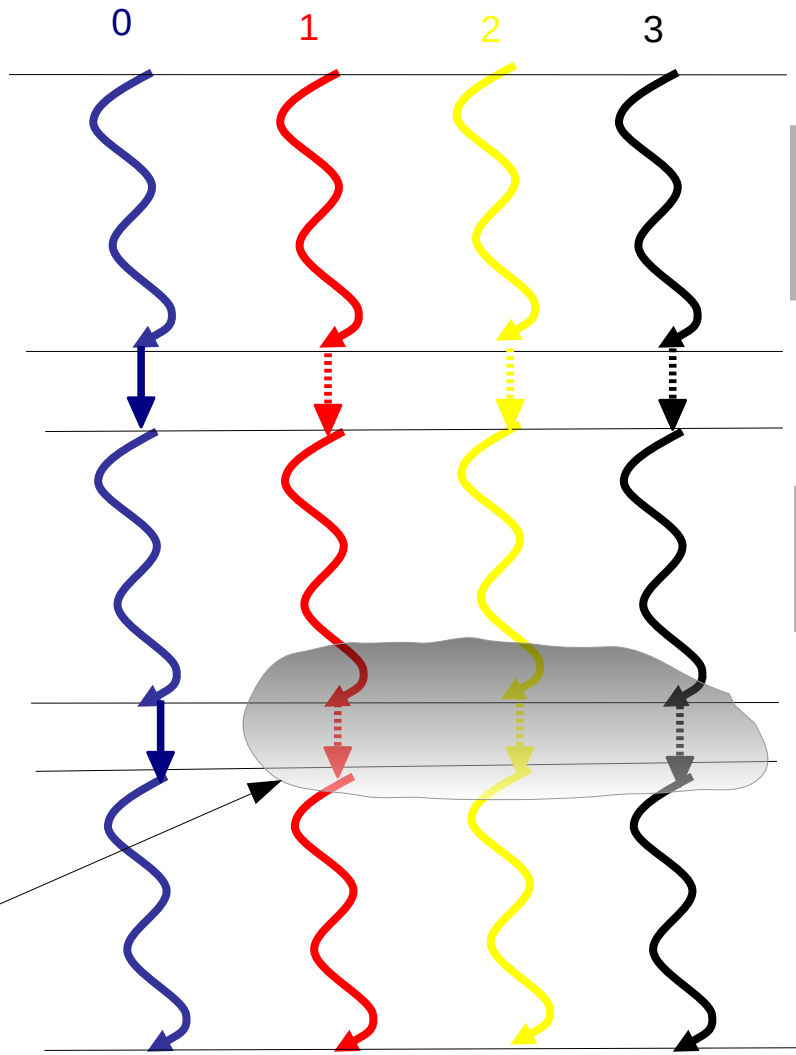
Only need to synchronize at the root
→ MPI_Reduce() to calculate: $\Sigma \log(l_i)$



Parallel Post-order Traversal



Classic Fork-Join with Busy-Wait



Broadcast Traversal — Trigger

Compute all vectors in Traversal descriptor

Reduce $\Sigma \log(l_i)$ — Barrier

Broadcast Traversal — Trigger

Compute all vectors in Traversal descriptor

Reduce $\Sigma \log(l_i)$ — Barrier

Broadcast new α params. — Trigger

Compute all vectors in Traversal descriptor

Reduce $\Sigma \log(l_i)$ — Barrier

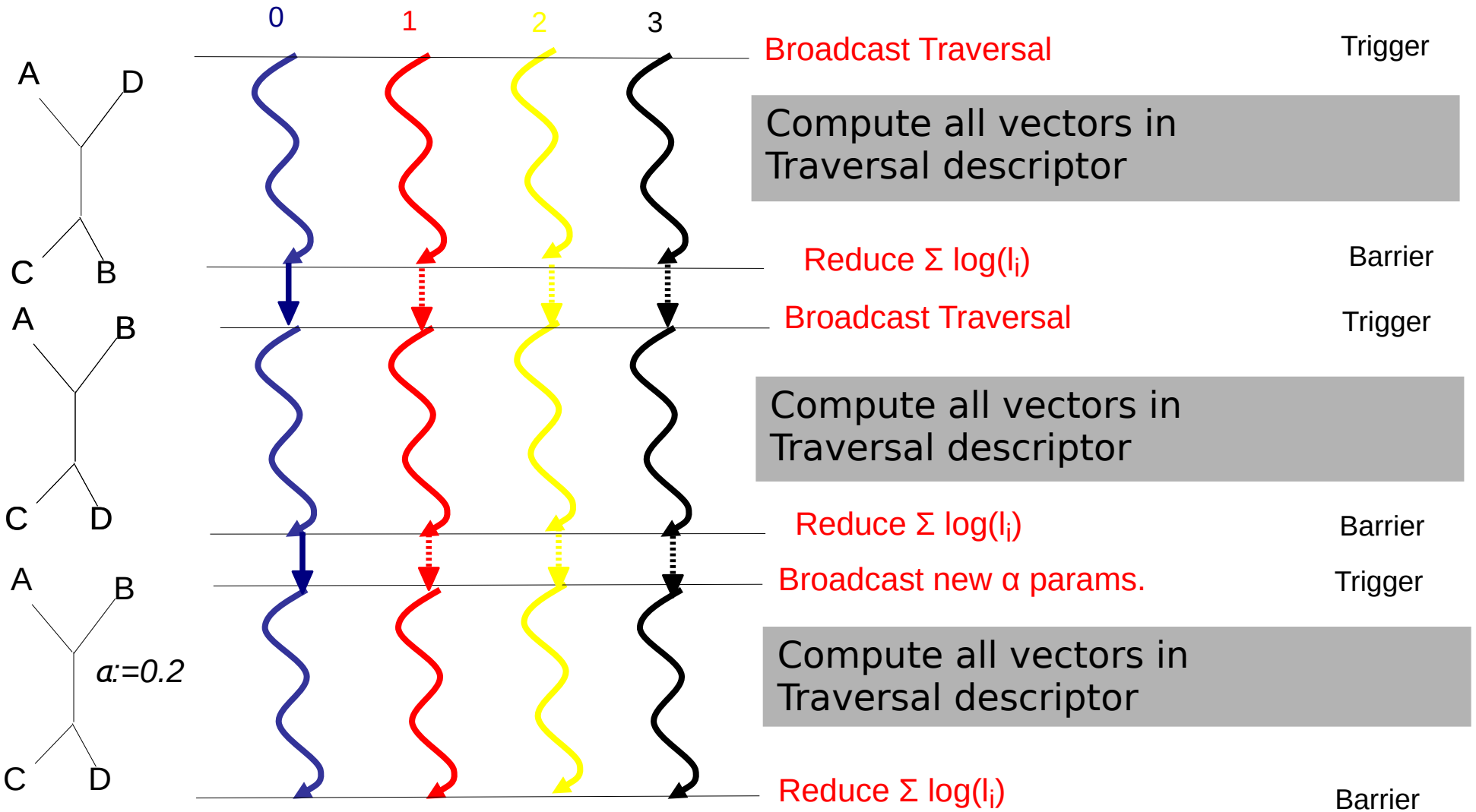
Busy wait

Synchronizations in RAxML with Pthreads

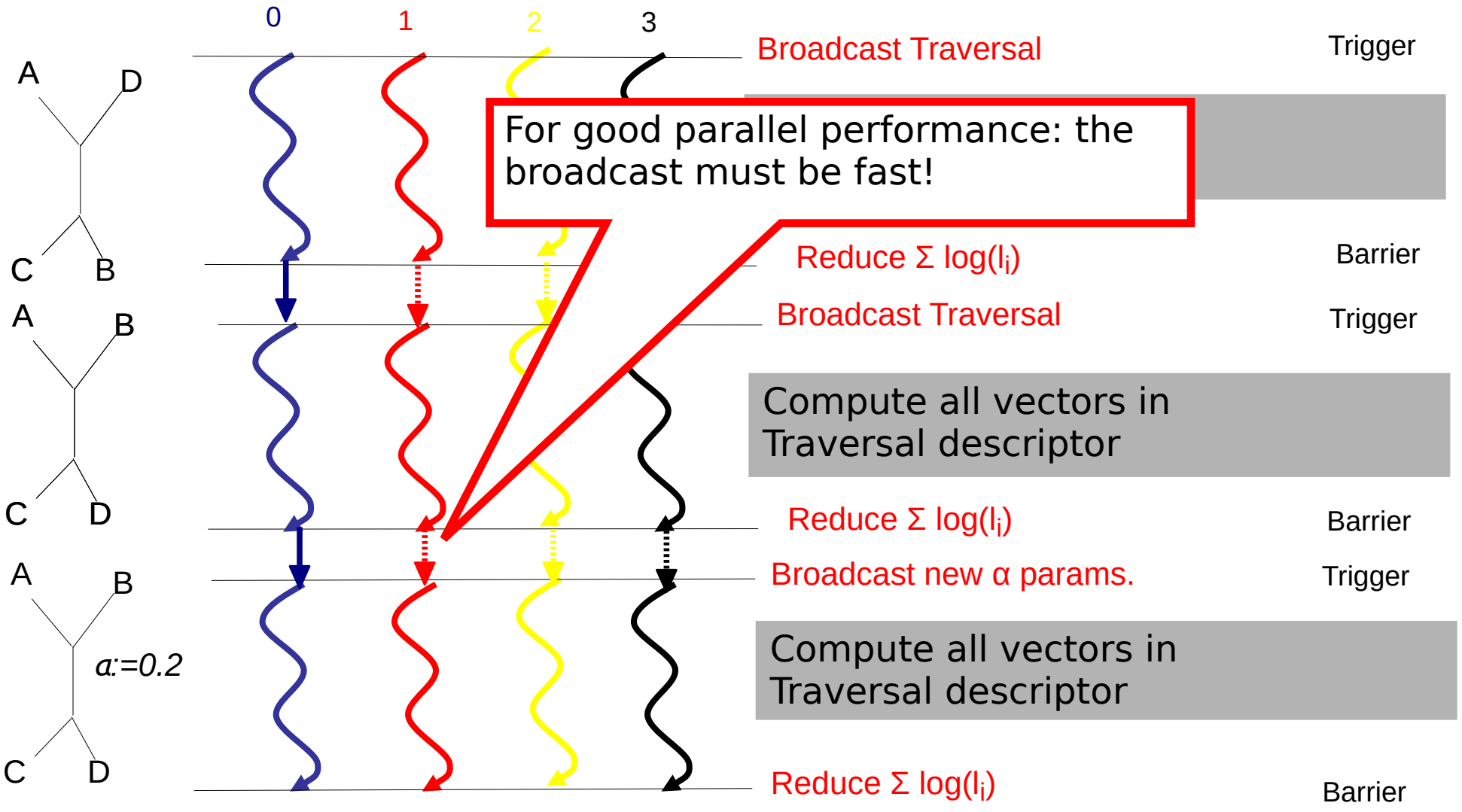
- RAxML Pthreads for a run time of about **10 seconds** on 16 cores/threads
- 404 taxa 7429 sites: **194,000** Barriers
- 1481 taxa 1241 sites: **739,000** Barriers
- A paper on performance of alternative PThreads barrier implementations:

  S.A. Berger, A. Stamatakis: "Assessment of Barrier Implementions for Fine-Grain Parallel Regions on Current Multi-core Architectures", *IEEE Cluster* 2010.

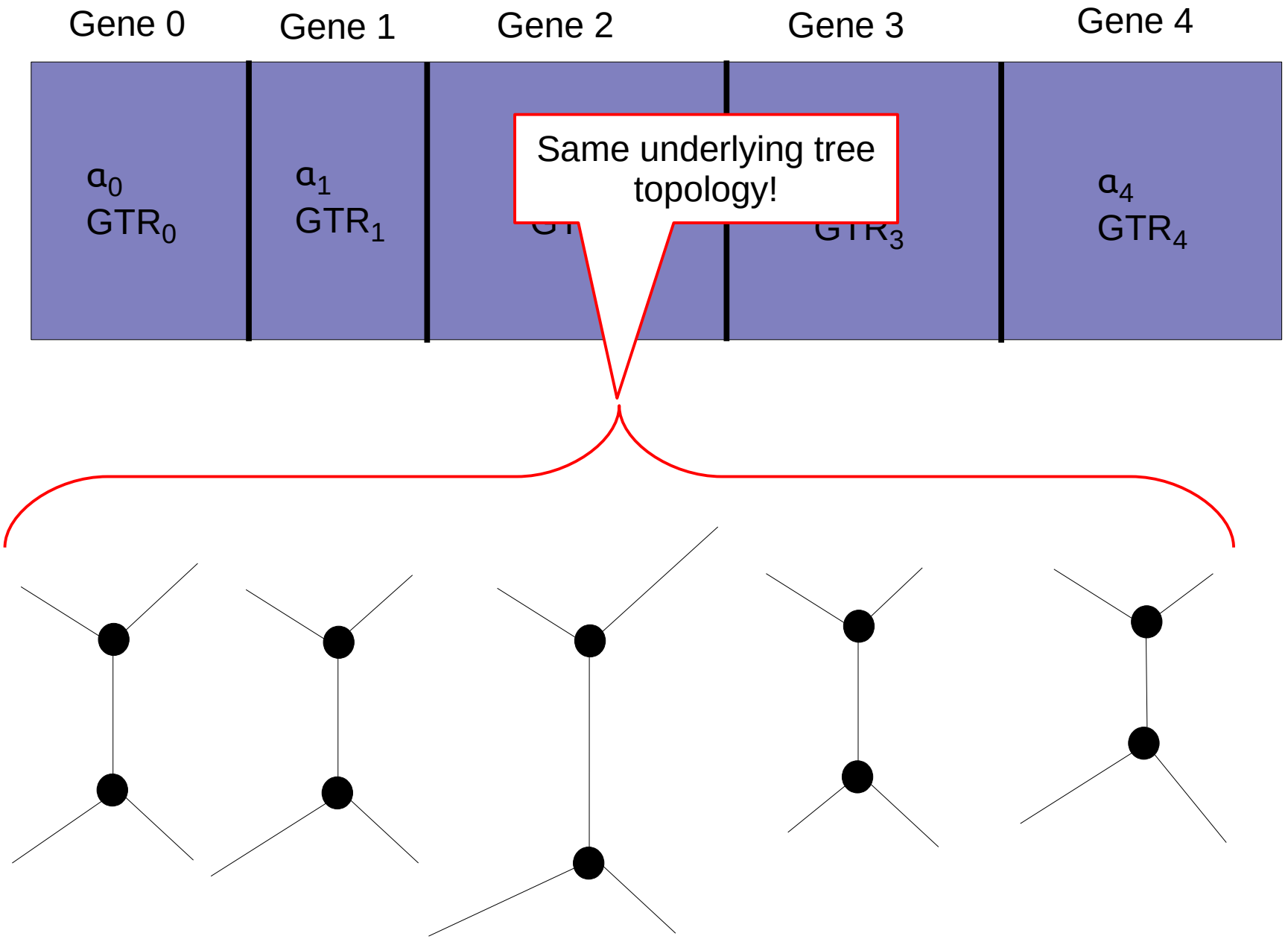
Classic Fork-Join with Busy-Wait (model optimization)



Classic Fork-Join with Busy-Wait (model optimization)



Problems start with partitioned datasets!

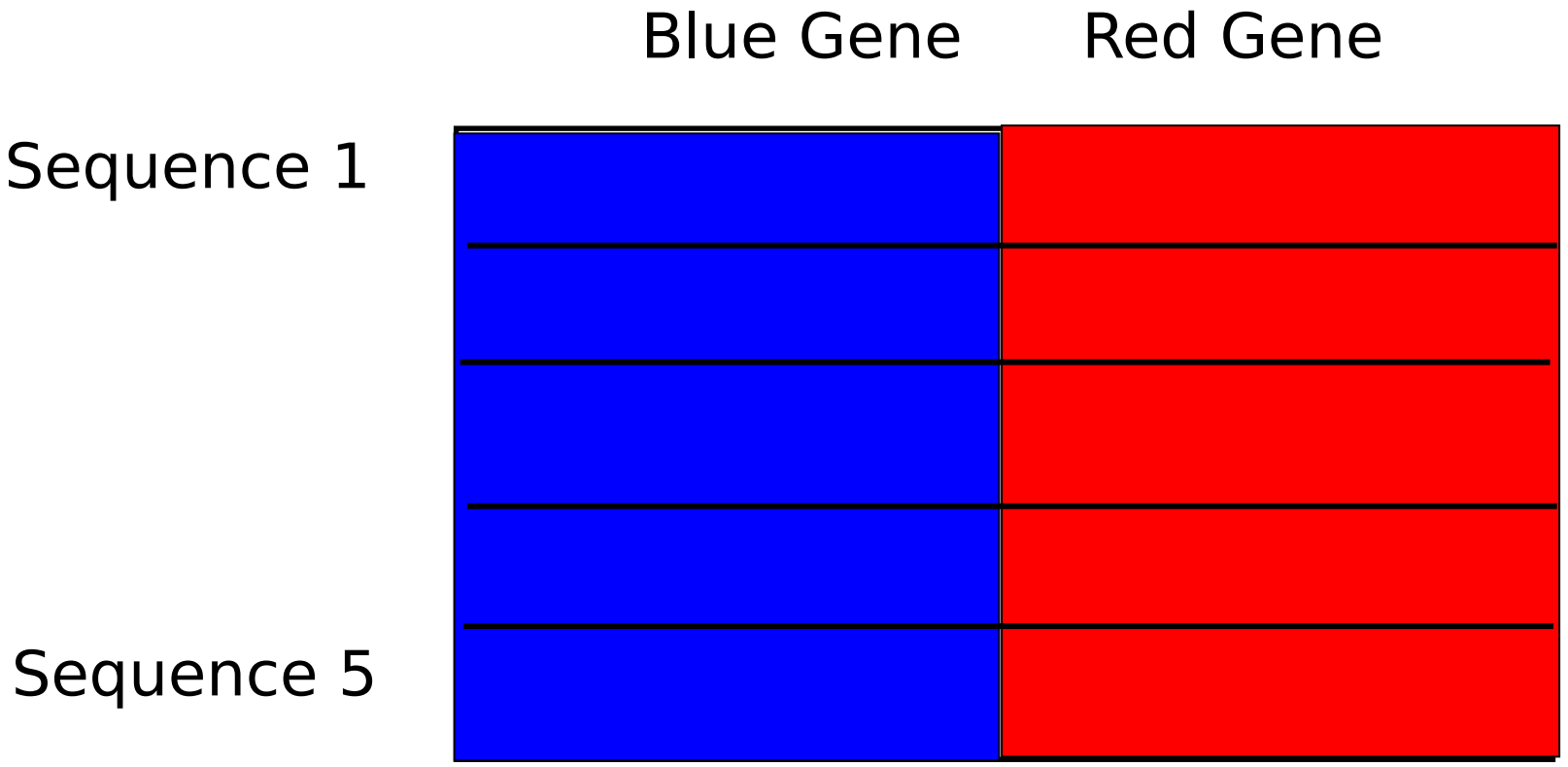


Parallel Performance Problems

- They all start with partitioned datasets!
- How do we distribute partitions to processors?
- How do we calculate parameter changes?
- How much time does our broadcast take?
- Goal: Keep all processors busy all the time

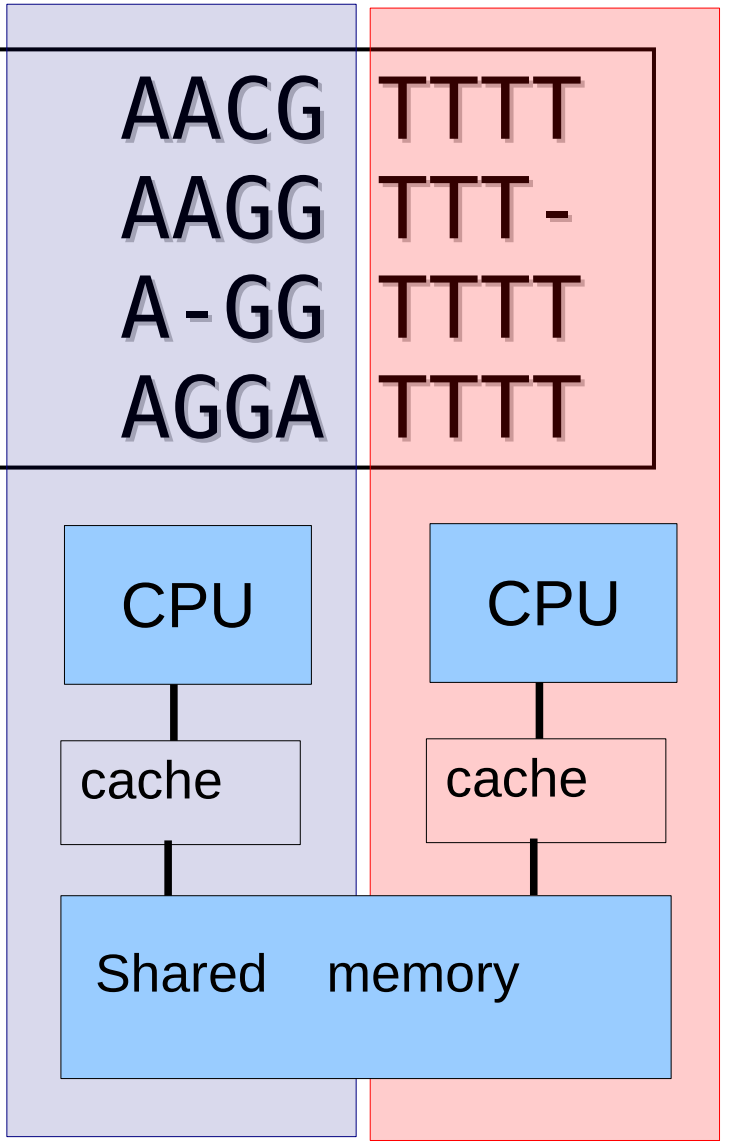
  → minimize communication and synchronization!

Example



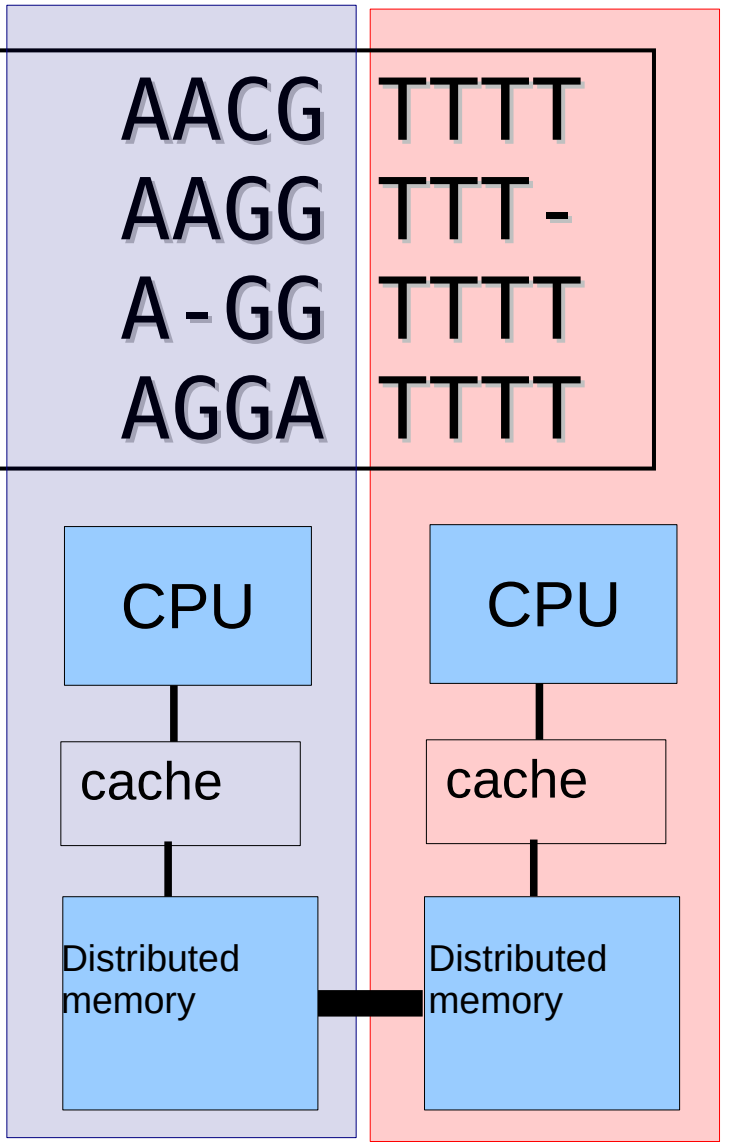
Data Distribution

```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



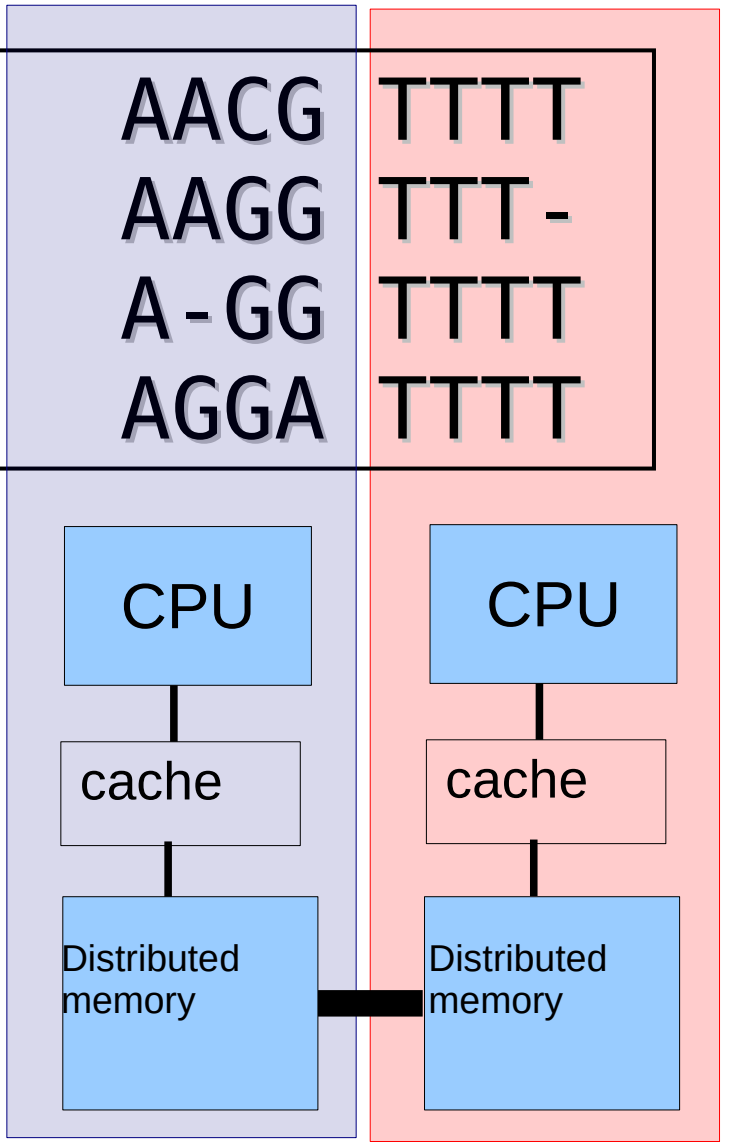
Data Distribution

```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



Data Distribution

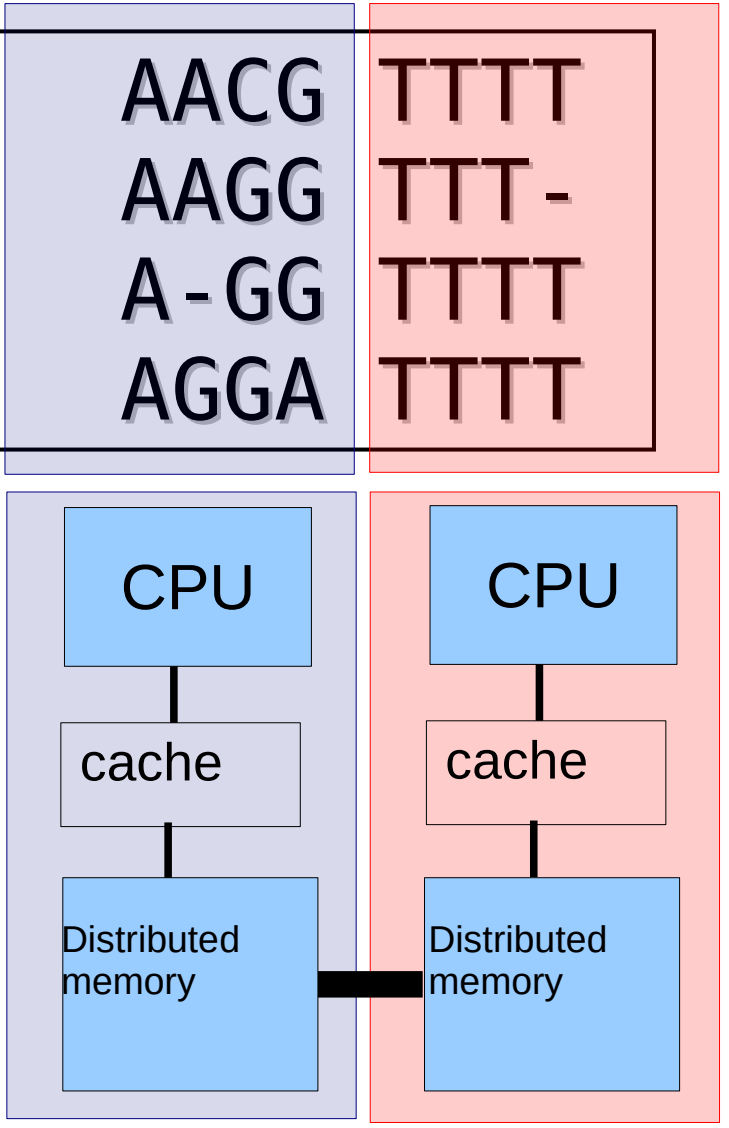
```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



Partitioned data distribution is not that trivial!

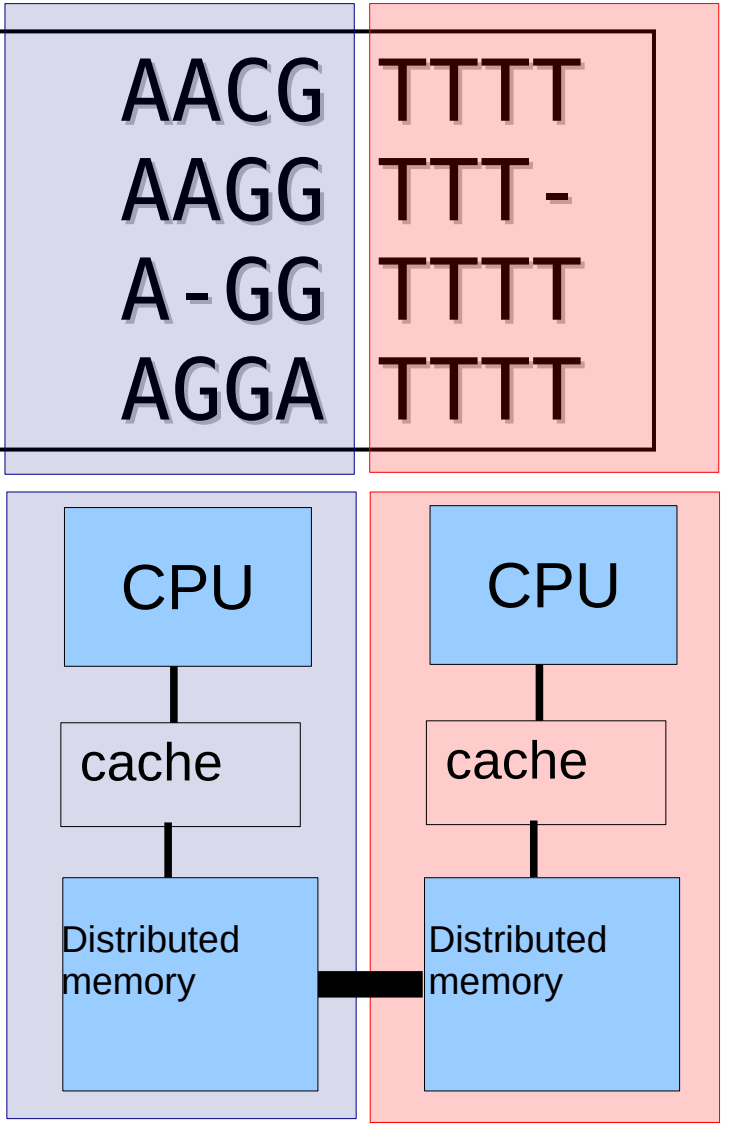
Data Distribution I

```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



Data Distribution I

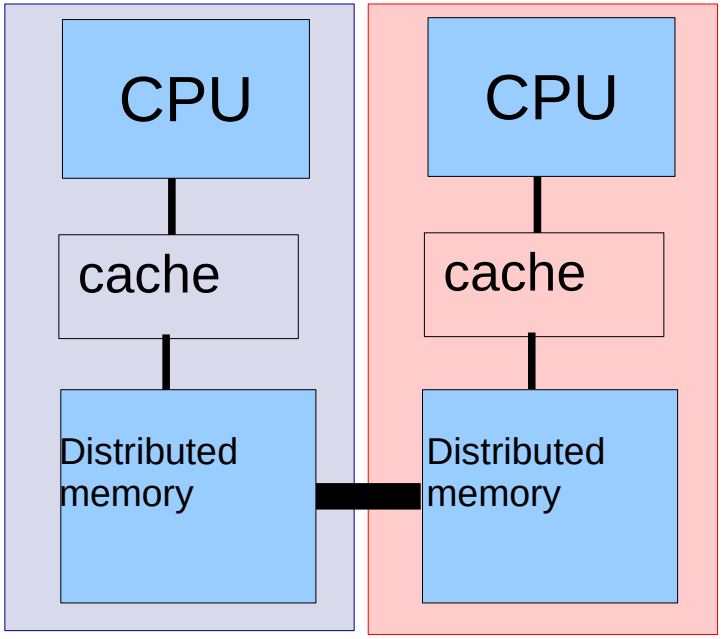
```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



Works well when we have more partitions than processors:
May lead to load imbalance not all processors obtain equal number of sites!
Not all partitions have equal length!

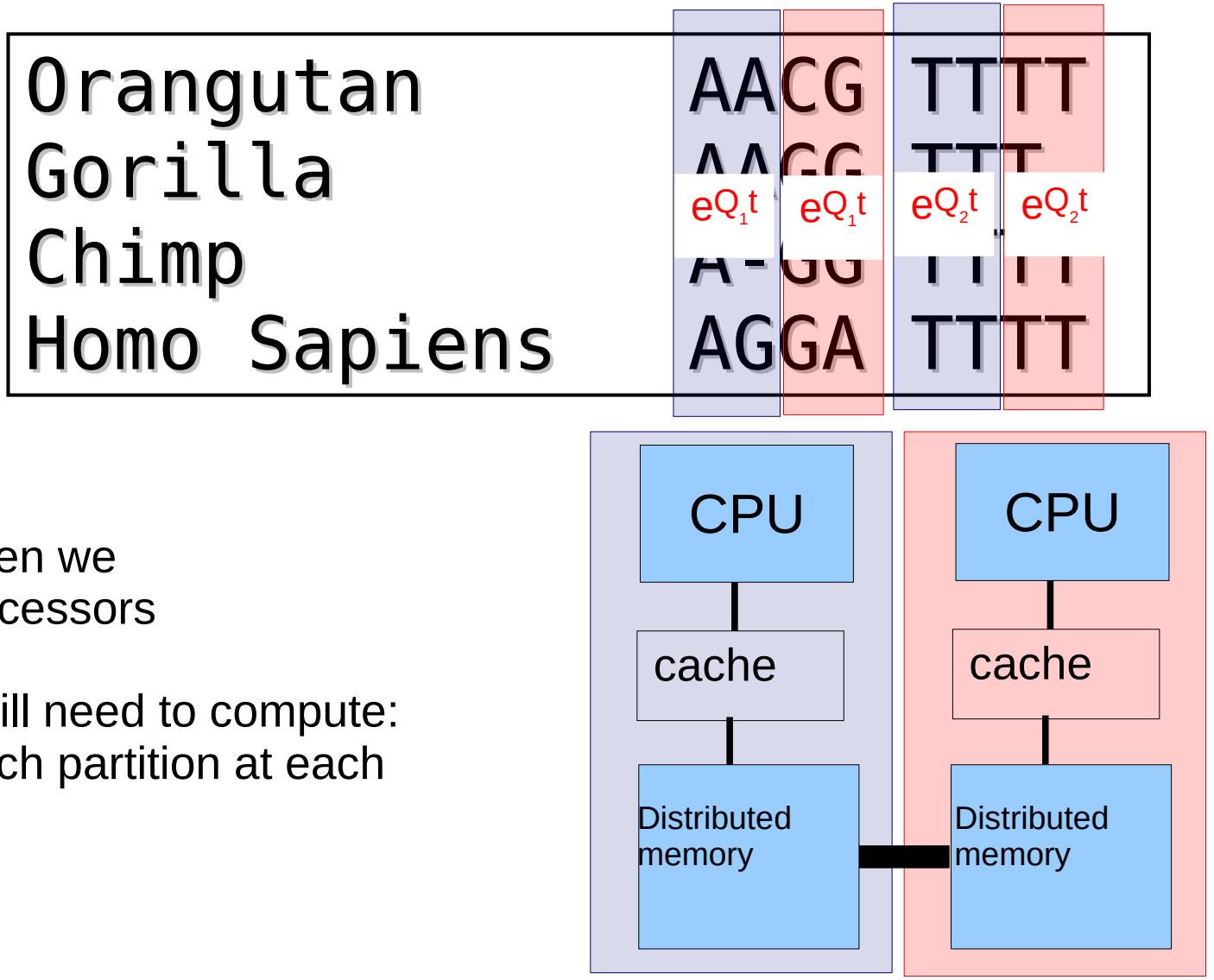
Data Distribution II

```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



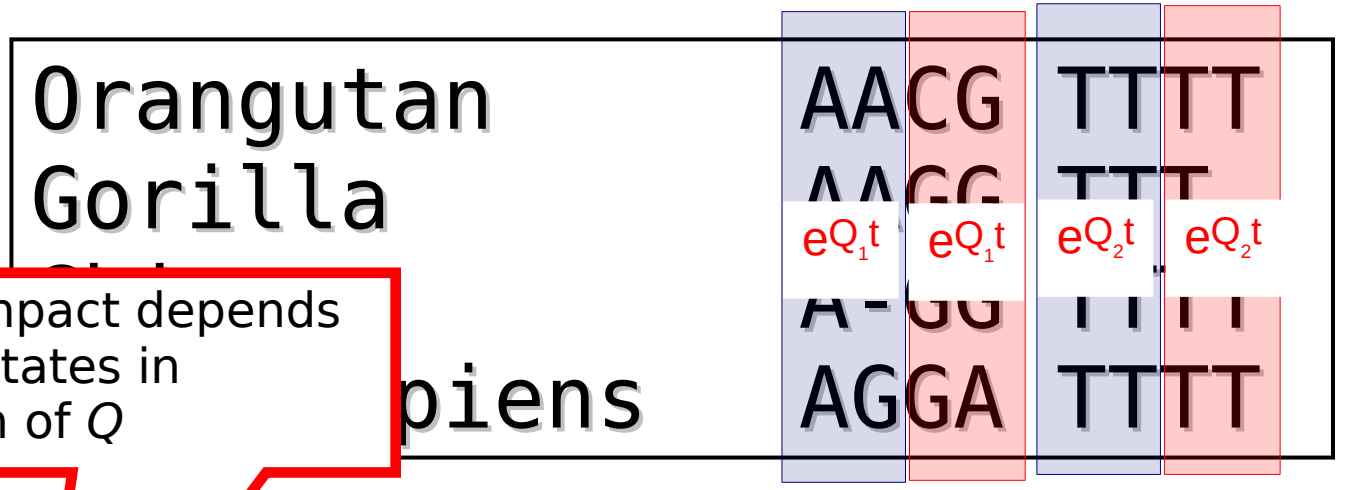
Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

Data Distribution II



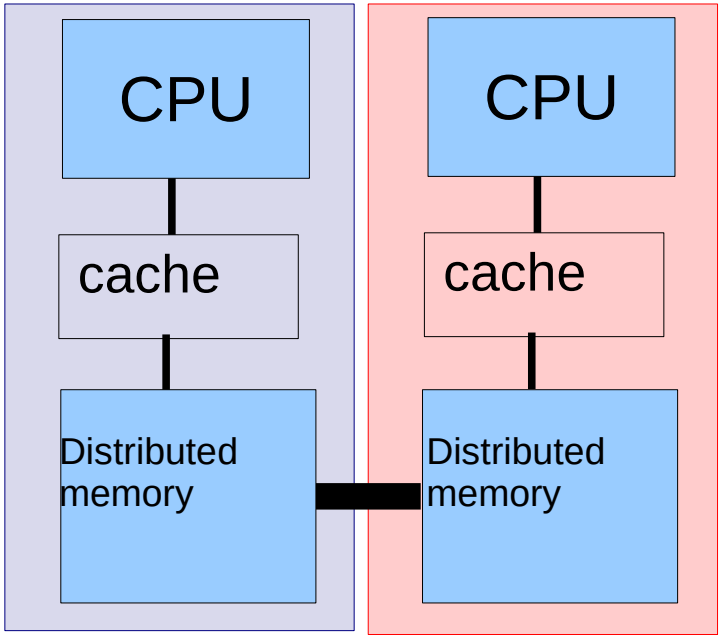
Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

Data Distribution II

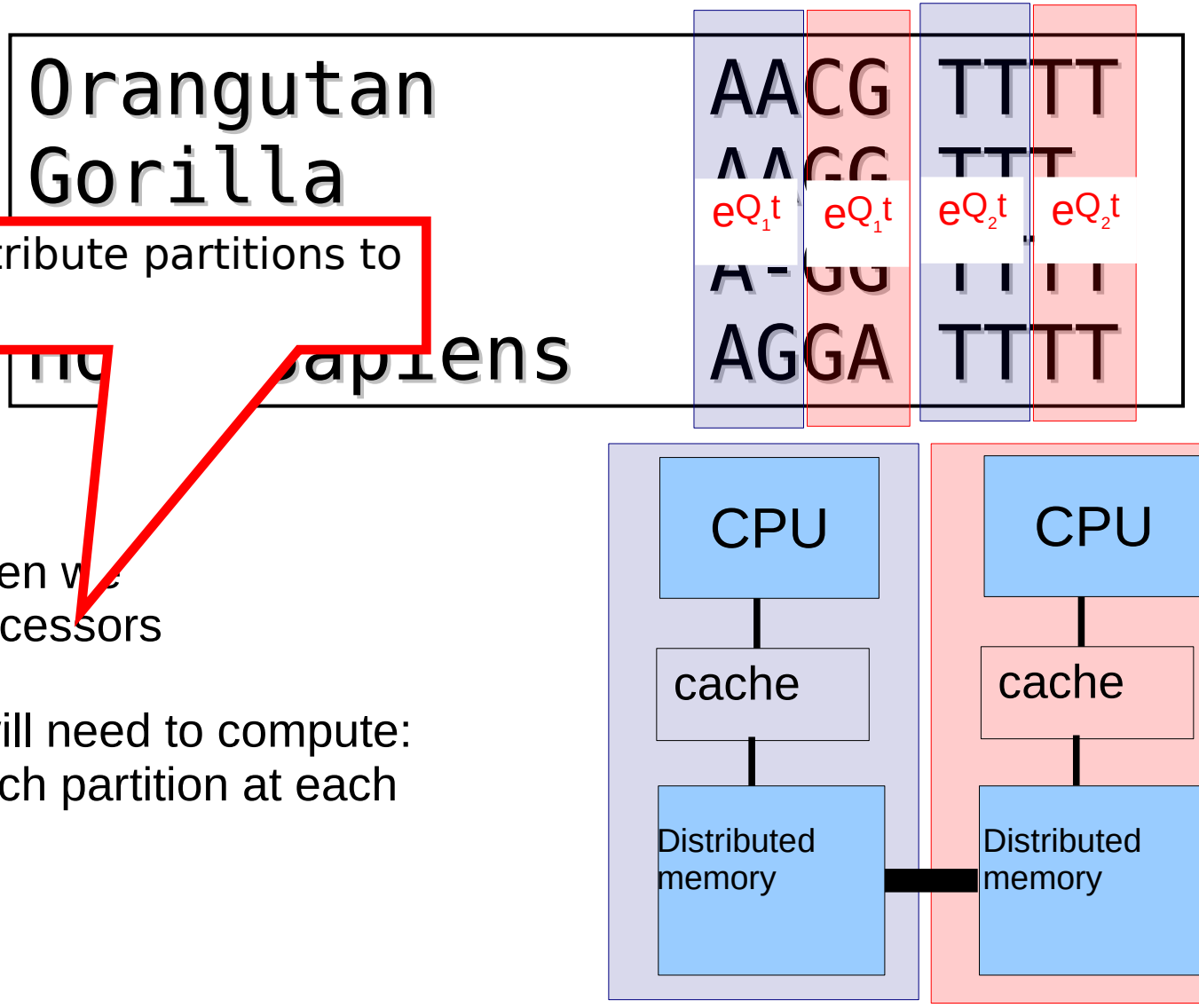


Performance impact depends on number of states in data/dimension of *Q*

Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

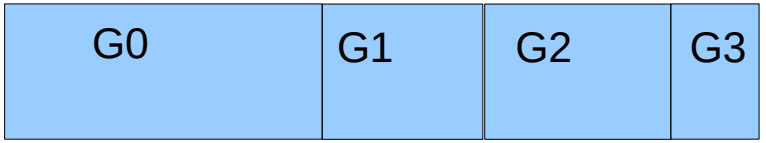


Data Distribution II



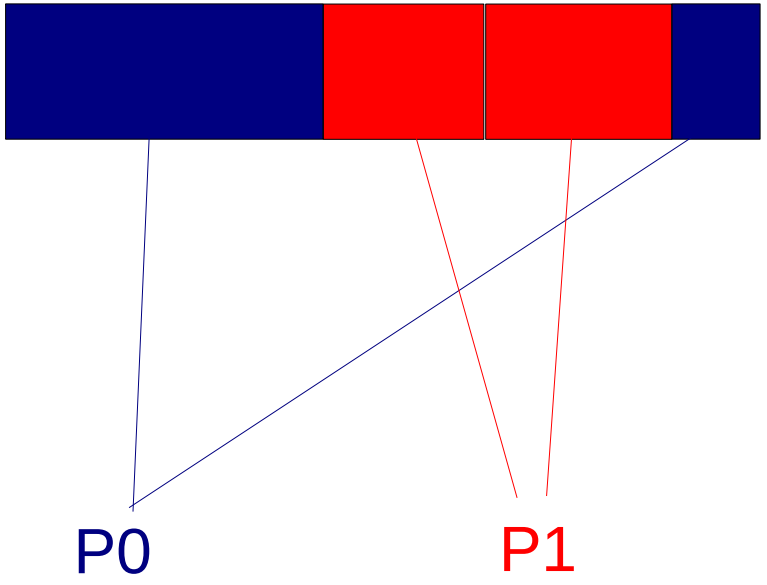
Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

Load Balance I

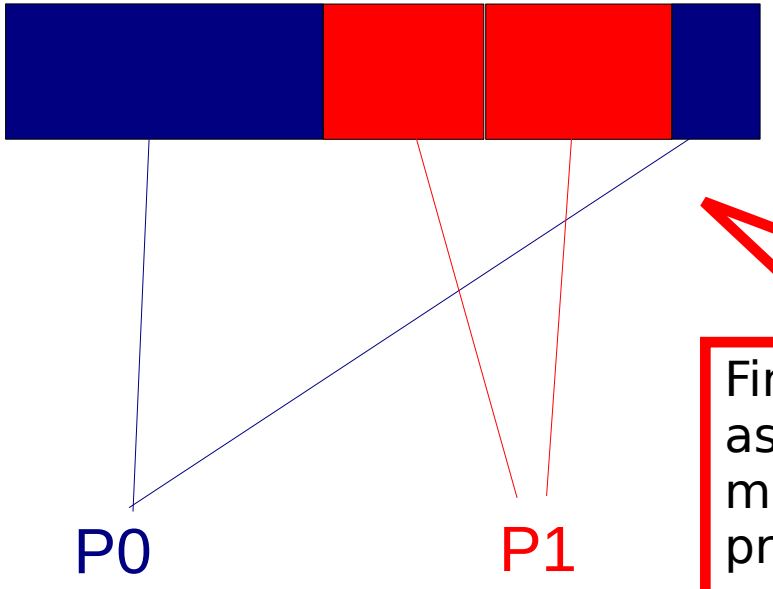


P0 P1

Load Balance I



Load Balance I



Find the partition-to-processor assignment such that the maximum number of sites per processor is minimized
→ this is NP-hard

Load Balance I

- The **multiprocessor job scheduling problem** in phylogenetics
  - Problem when #partitions >> #cores
  - Tested per-site (cyclic/modulo) data distribution versus per partition data distribution
  - We used the Longest Processing Time (LPT) heuristics for assigning partitions to processors
  - 25 taxa, 220,000 sites, 100 genes
    - GAMMA model

      naïve: **613** secs

      LPT: **550** secs
    - CAT model

      naïve: **298** secs

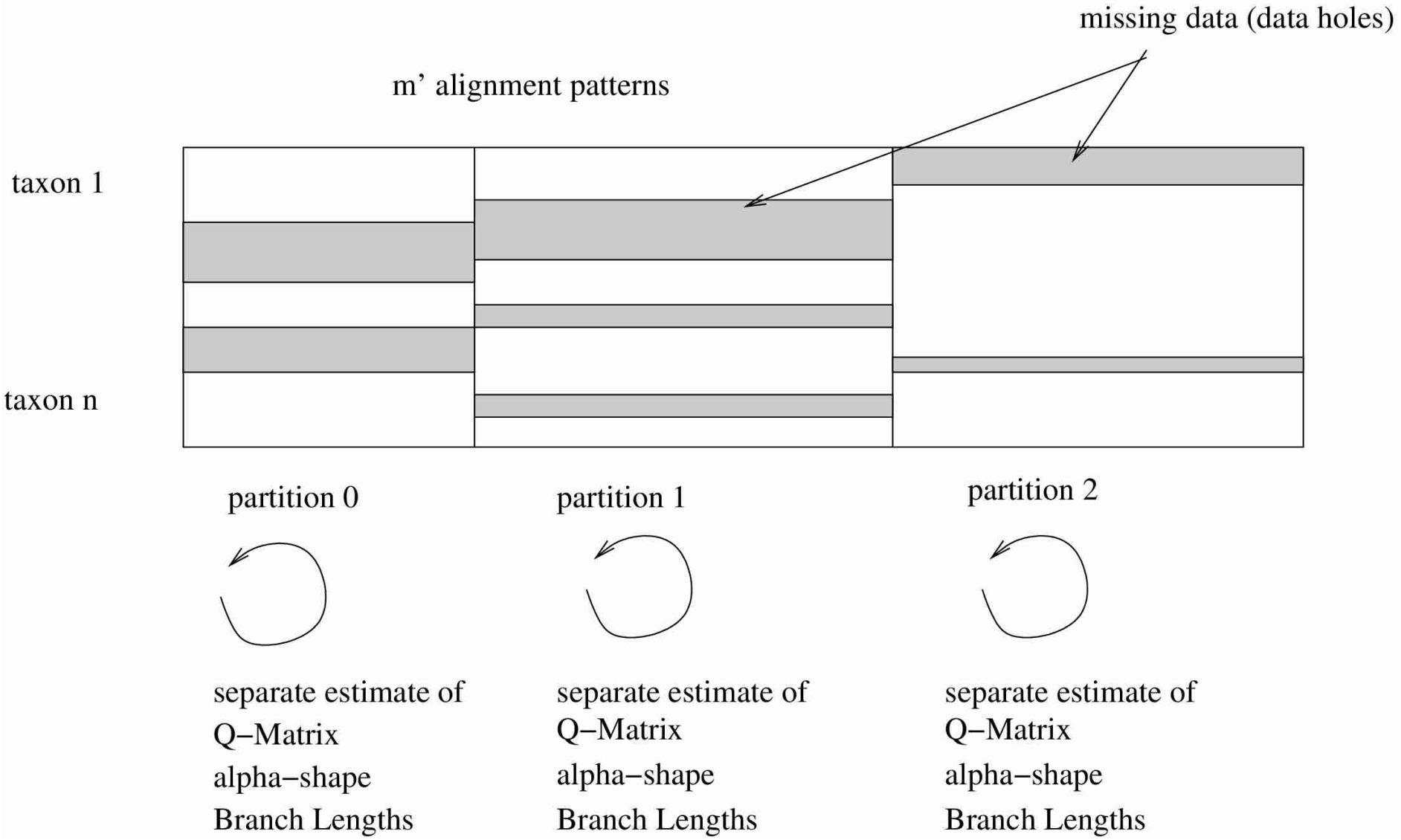
      LPT: **127** secs
  - Larger protein dataset under Γ model of rate heterogeneity: 10-fold performance improvement!

J. Zhang, A. Stamatakis: "The Multi-Processor Scheduling Problem in Phylogenetics", 11th IEEE HICOMB workshop (in conjunction with IPDPS 2012).

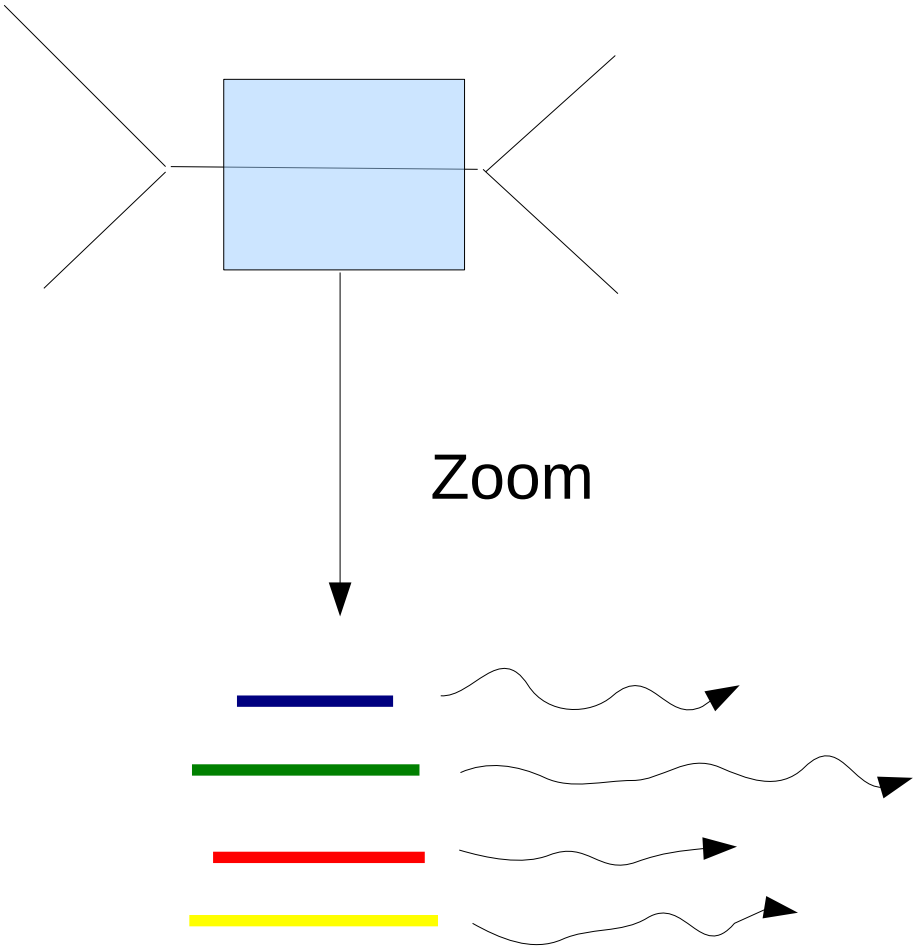
LPT heuristics for multi-processor scheduling

- Sort jobs (partitions) by processing length (partition length) in decreasing order
- Remove a job (partition) from the sorted list and assign it to the processor with the earliest end time (the smallest sum of partition lengths/number of sites)
- Repeat until the sorted list is empty
- Upper bound: *4/3 – 1/(3p) * OPT,* where *p* is the number of processors
- Graham, R. L.: "Bounds on Multiprocessing Timing Anomalies". *SIAM Journal on Applied Mathematics* 17 (2): 416–429, 1969.
- Remark: LPT works surprisingly well (see our paper on the phylogenetic problem where we also tested other heuristics)

Partitioned Branch Lengths & other parameters



Load-Balance II



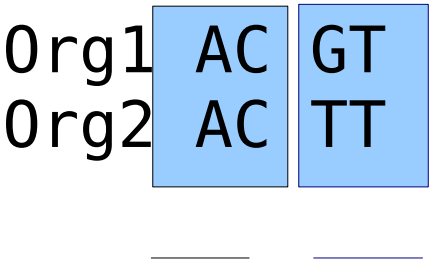
Synchronization Points

- Assume 10 branches
- Each branch requires 10 Newton-Raphson Iterations
- Each NR Iteration requires a synchronization via a reduction operation
- One branch/partition at a time: 100 sync. points, less work (only one partition) per sync. point
- All branches concurrently: 10 sync. points, more work per sync. point
- Branches will need distinct number of operations
- Add convergence state → bit vector

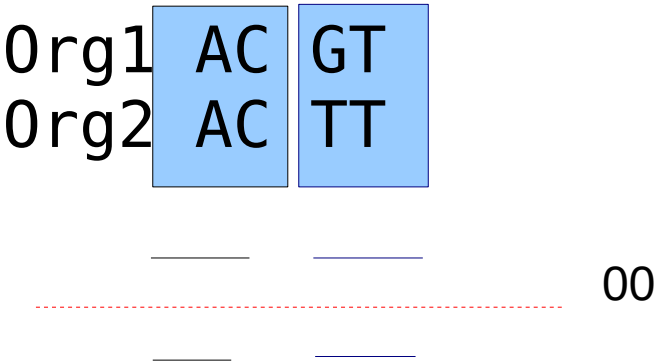
Synchronization Points

```
Org1 AC GT
Org2 AC TT
```

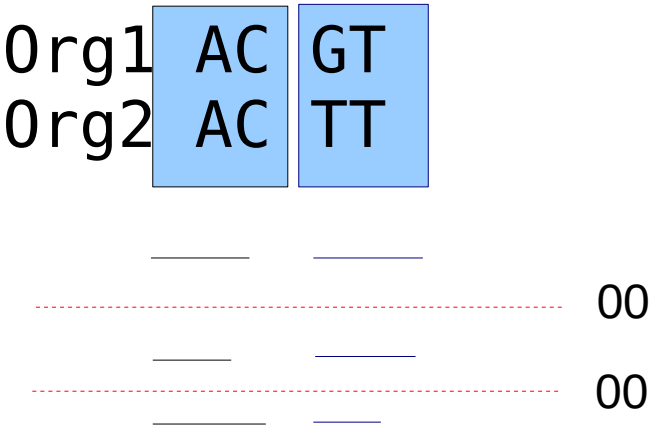
Synchronization Points



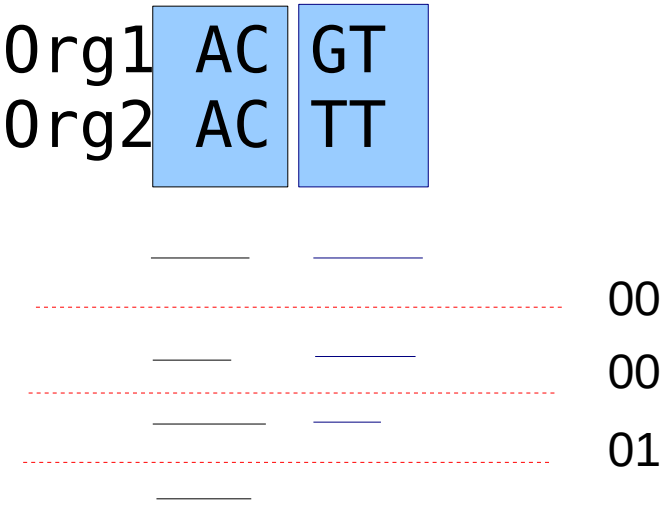
Synchronization Points



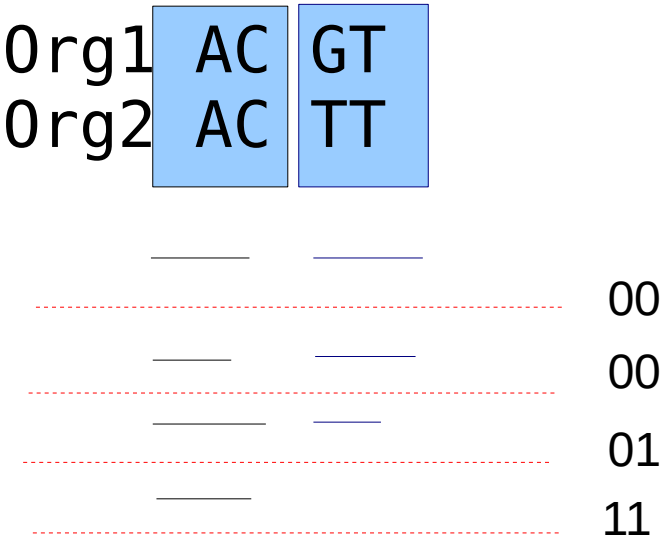
Synchronization Points



Synchronization Points

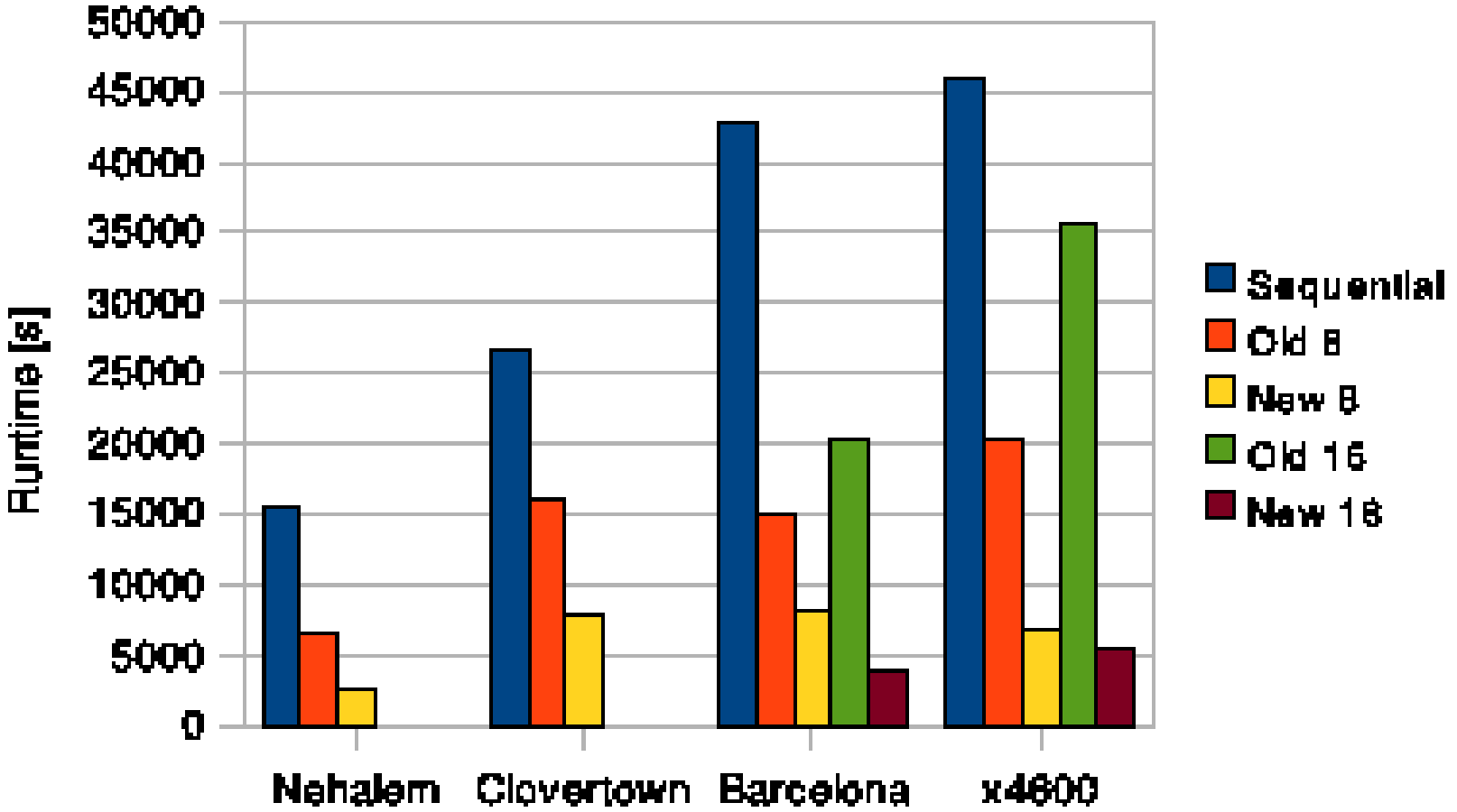


Synchronization Points



In this example: 4 instead of 7 sync points!

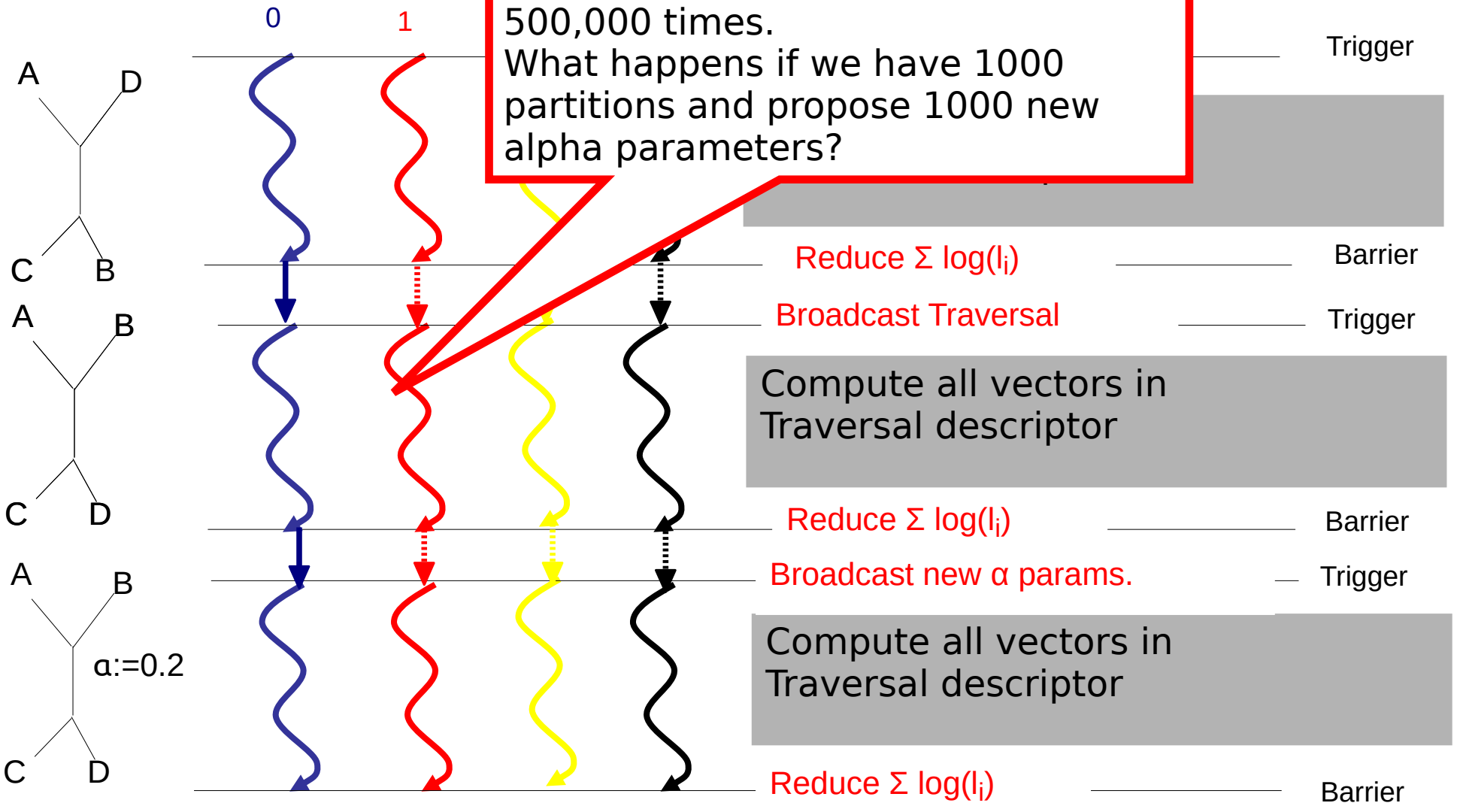
Load Balance II



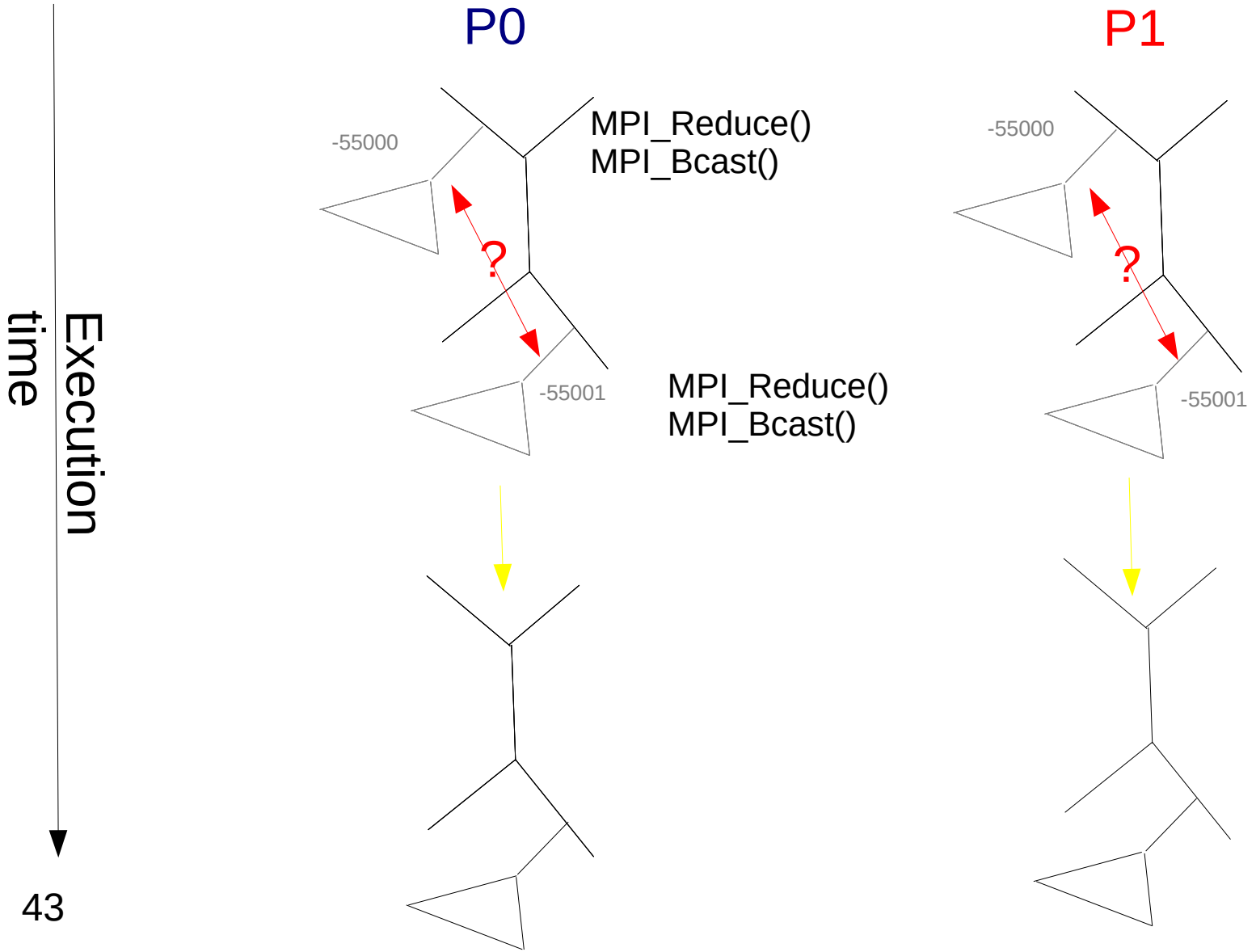
A. Stamatakis, M. Ott: "Load Balance in the Phylogenetic Likelihood Kernel". Proceedings of ICPP 2009, Vienna, Austria, September 2009.

Classic Fork-Join with

For good parallel performance: the broadcast must be fast!
Remember: 10 secs 16 cores approx 500,000 times.
What happens if we have 1000 partitions and propose 1000 new alpha parameters?



Alternative MPI parallelization



Outline

- Last time:
  - What is hidden in *P(t)* – what do the models look like?
  - How to compute the Maximum Likelihood score on a tree?
  - Advanced substitution models
  - Efficiently computing the Likelihood on trees on a single processor!
- Today
  - Efficiently computing the likelihood in parallel
  - **Bayesien Inference and Markov Chain Monte Carlo**

Outline – Bayesian Inference

- Bayesian statistics
- Monte-Carlo simulations
- Markov-Chain Monte-Carlo (MCMC) methods
- Metropolis-coupled MCMC-methods
- Some phylogenetic proposals
- Reversible jump MCMC

Bayesian and Maximum Likelihood Inference

- In phylogenetics Bayesian and ML (Maximum Likelihood) methods have **a lot** in common
- Computationally, both approaches re-evaluate the phylogenetic likelihood *over and over and over* again for different tree topologies, branch lengths, and model parameters
- Bayesian and ML codes spend approx. 80-95% of their total run time in likelihood calculations on trees
- Bayesian methods sample the posterior probability **distribution**
- ML methods strive to find a **point estimate** that maximizes the likelihood

Bayesian Phylogenetic Methods

- The methods used perform stochastic searches, that is, they do not strive to maximize the likelihood, but rather integrate over it
- Thus, no numerical optimization methods for model parameters and branch lengths are needed, parameters are proposed at random
- It is substantially easier to infer trees under complex models using Bayesian statistics than using Maximum Likelihood

A Review of Probabilities

Conditional Probability:

$Pr(A|B) = Pr(A,B) / Pr(B)$

Joint Probability:

$Pr(A,B) = Pr(A|B)\ Pr(B)$

and

$Pr(A,B) = Pr(B|A)\ Pr(A)$

Problem:

If I can compute $Pr(A|B)$ how can I get $Pr(B|A)$?

A Review of Probabilities

Conditional Probability:

$Pr(A|B) = Pr(A,B) / Pr(B)$

Joint Probability:

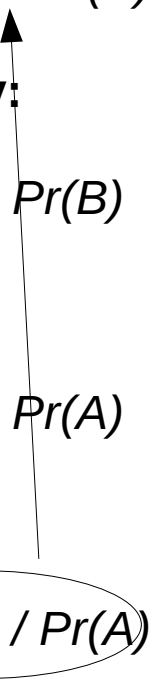
$Pr(A,B) = Pr(A|B)\ Pr(B)$

and

$Pr(A,B) = Pr(B|A)\ Pr(A)$

Bayes Theorem:

$Pr(B|A) = Pr(A,B) / Pr(A)$



A Review of Probabilities

Conditional Probability:

$Pr(A|B) = Pr(A,B) / Pr(B)$

Joint Probability:

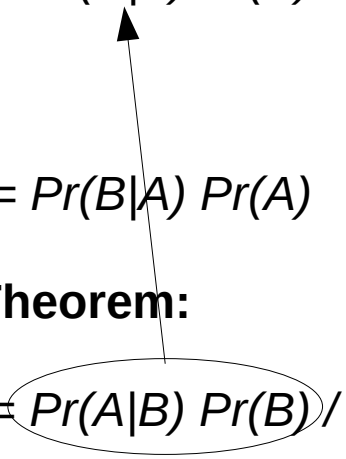
$Pr(A,B) = Pr(A|B)\ Pr(B)$

and

$Pr(A,B) = Pr(B|A)\ Pr(A)$

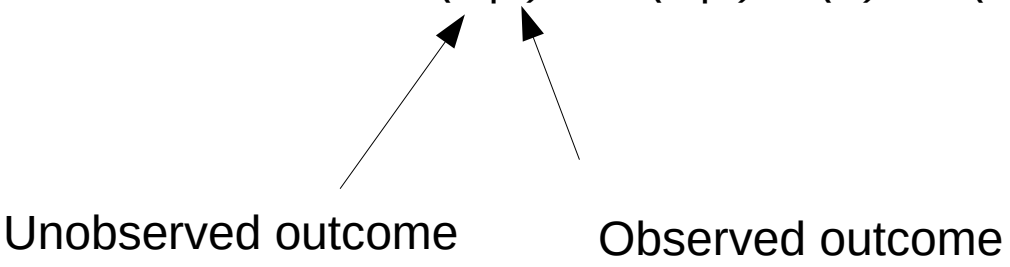
Bayes Theorem:

$Pr(B|A) = Pr(A|B)\ Pr(B) / Pr(A)$



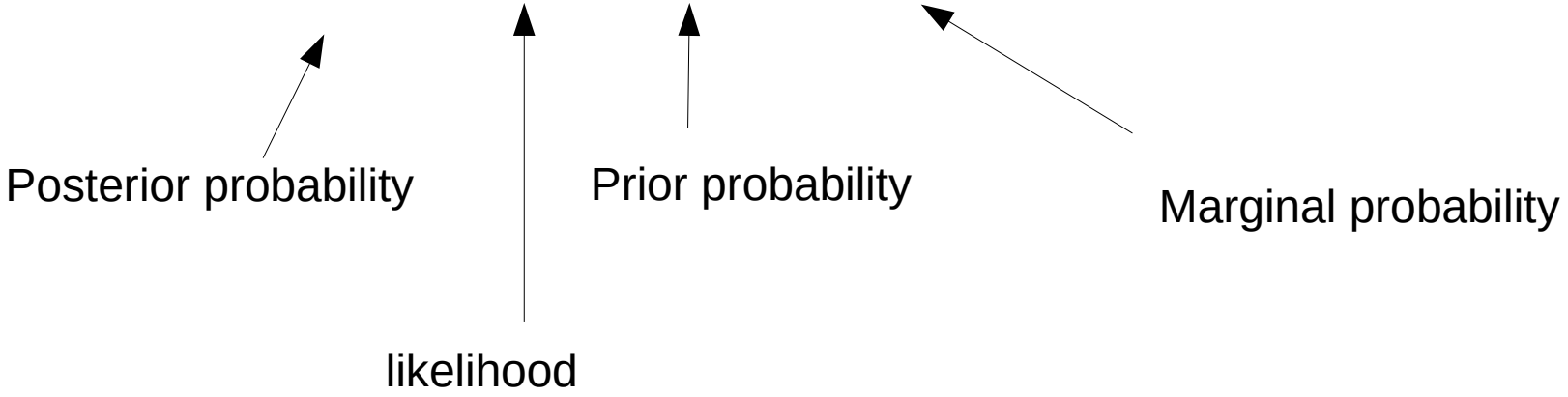
Bayes Theorem

$$Pr(B|A) = Pr(A|B)\ Pr(B) / Pr(A)$$

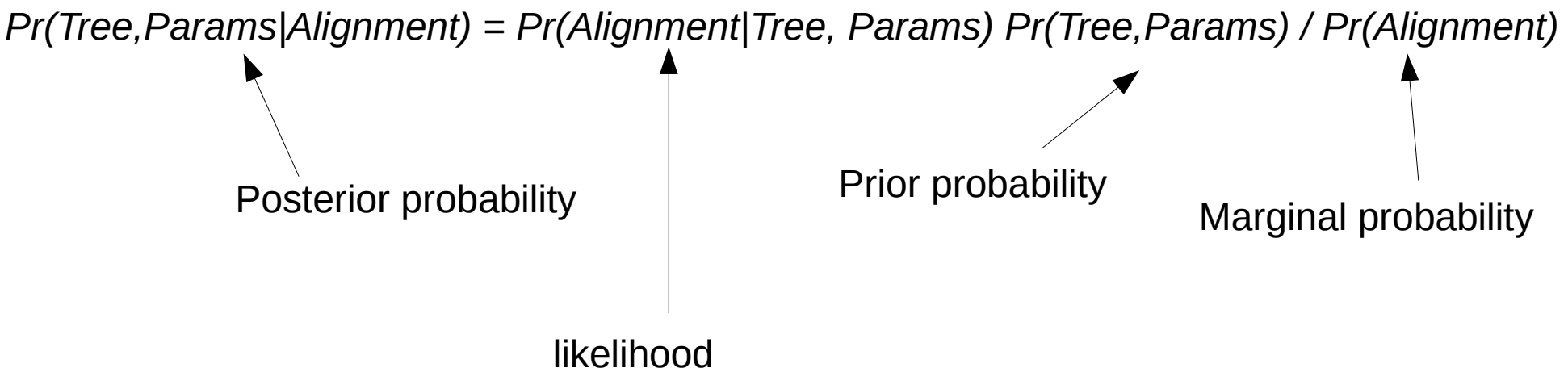


Bayes Theorem

$$Pr(B|A) = Pr(A|B)\ Pr(B) / Pr(A)$$



Bayes Theorem: Phylogenetics

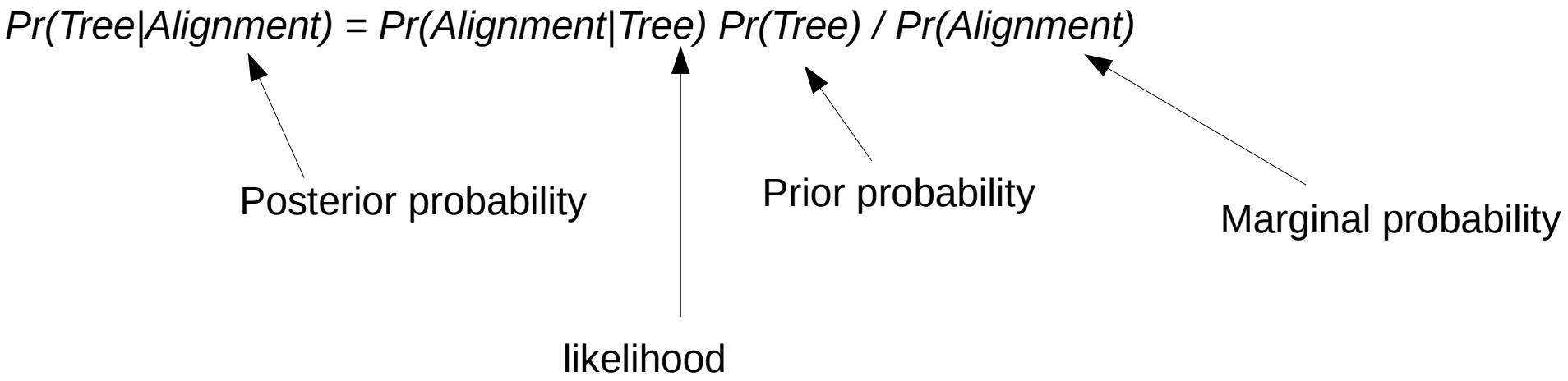


Posterior probability: distribution over all possible trees and all model parameter values
Likelihood: does the alignment fit the tree and model parameters?
Prior probability: introduces prior knowledge/assumptions about the probability distribution of trees and model parameters (e.g., GTR rates, α shape parameter).

For instance, we typically assume that all possible tree topologies are equally probable → uniform prior

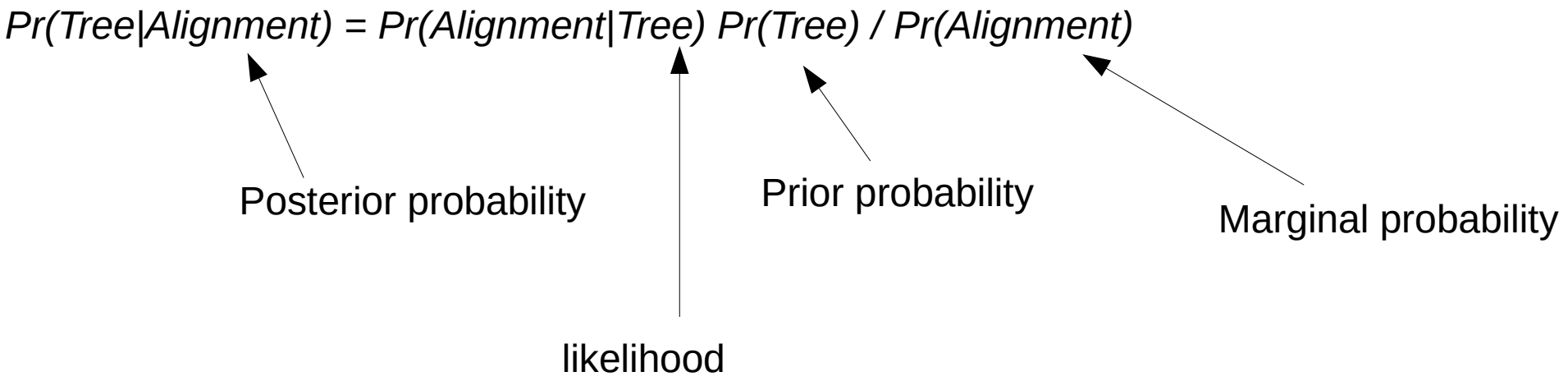
Marginal probability: how do we obtain this?

Bayes Theorem: Phylogenetics



Marginal probability: Assume that our only model parameter is the tree and marginalizing means summing over all unconditional probabilities, thus
Pr(Alignment)
can be written as
$Pr(Alignment) = Pr(Alignment, t_0) + Pr(Alignment, t_1) + \ldots + Pr(Alignemnt, t_n)$
where *n+1* is the number of possible trees!

Bayes Theorem: Phylogenetics



Marginal probability: Assume that our only model parameter is the tree and marginalizing means summing over all unconditional probabilities, thus

Pr(Alignment)

can be written as

$Pr(Alignment) = Pr(Alignment, t_0) + Pr(Alignment,t_1) + \ldots + Pr(Alignemnt, t_n)$

where *n+1* is the number of possible trees!

This can be re-written as

$Pr(Alignment) = Pr(Alignment|t_0)\ Pr(t_0) + Pr(Alignment|t_1)\ Pr(t_1) + \ldots + Pr(Alignment|t_n)\ Pr(t_n)$

Bayes Theorem: Phylogenetics

$$Pr(Tree|Alignment) = Pr(Alignment|Tree)\ Pr(Tree) / Pr(Alignment)$$

Posterior probability

likelihood

Prior probability

Marginal probability

Marginal probability:

$$Pr(Alignment) = Pr(Alignment|t_0)\ Pr(t_0) + Pr(Alignment|t_1)\ Pr(t_1) + \ldots + Pr(Alignment|t_n)\ Pr(t_n)$$

likelihood

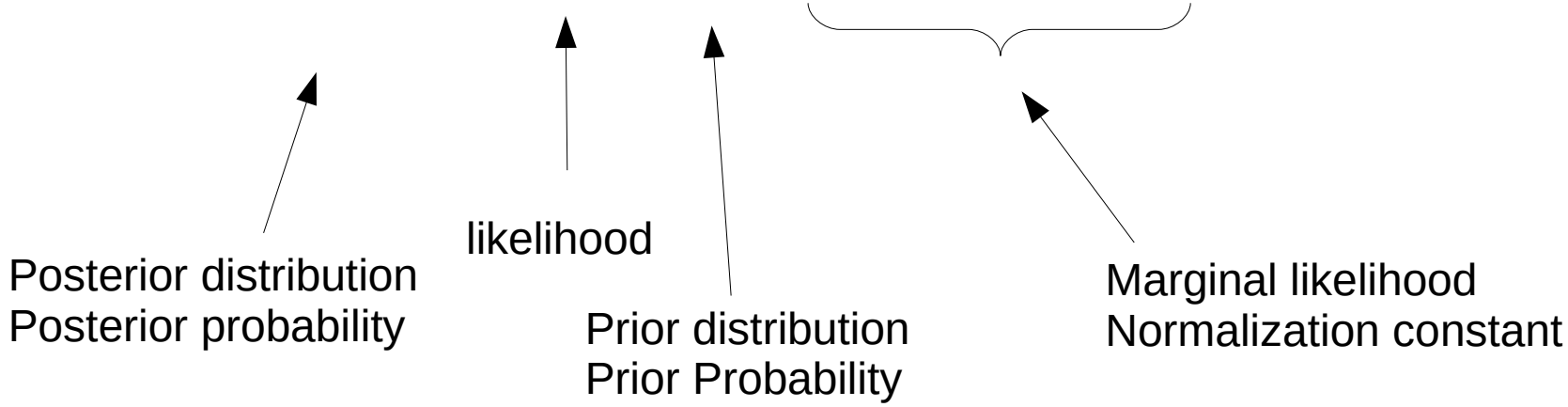
Prior := *1 / (n+1)*
→ this is a uniform prior!

Now, we have all the ingredients for computing *Pr(Tree|Alignment)*, however computing *Pr(Alignment)* is prohibitive due to the large number of trees!

With continuous parameters the above equation for obtaining the marginal probability becomes an integral. Usually, all parameters we integrate over (tree topology, model parameters, etc.) are lumped into a parameter vector denoted by θ

Bayes Theorem General Form

$$f(\theta|A) = f(A|\theta)\, f(\theta) \,/\, \int f(\theta) f(A|\theta) d\theta$$



We know how to compute $f(A|\theta)$ → the likelihood of the tree

Problems:

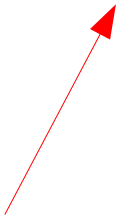
Problem 1: $f(\theta)$ is given a priori, but how do we chose an appropriate distribution?
→ biggest strength and weakness of Bayesian approaches

Problem 2: How can we calculate/approximate $\int f(\theta) f(A|\theta) d\theta$?
→ to explain this we need to introduce additional machinery

However, let us first look at an example for $f(\theta|A)$ *in* phylogenetics

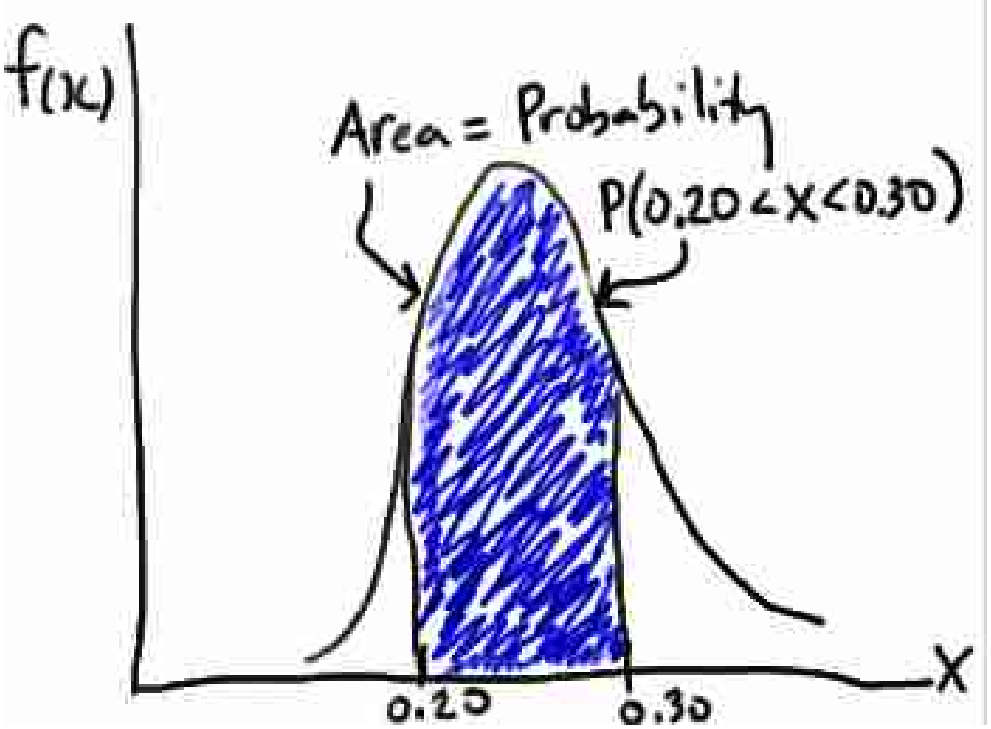
Bayes Theorem General Form

$$f(\theta|A) = f(A|\theta)\, f(\theta) \,/\, \int f(\theta) f(A|\theta) d\theta$$



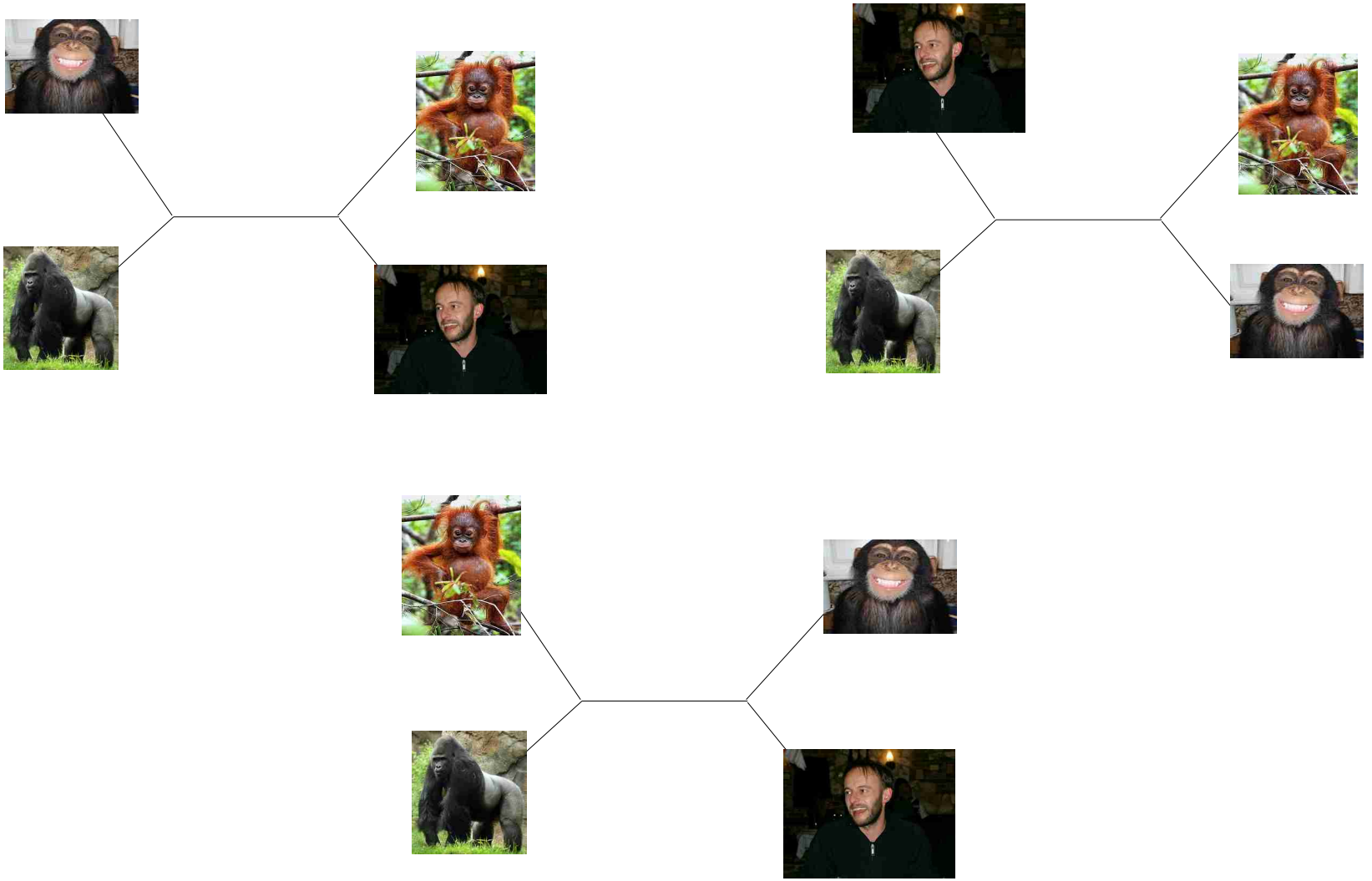
Note that, in the continuous case *f()* is called probability density function

Probability Density Function

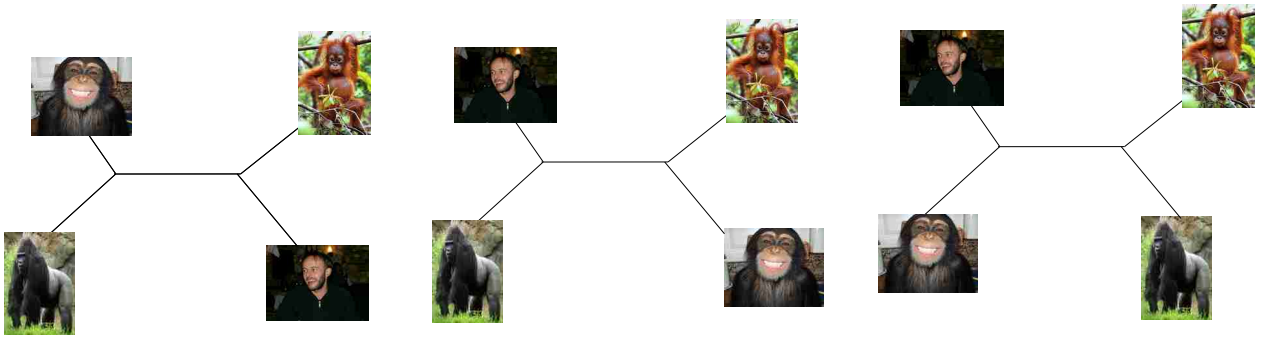


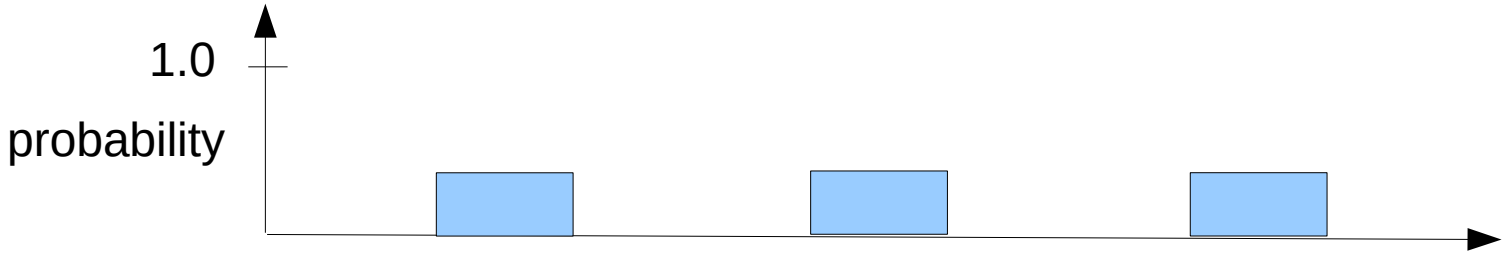
Properties:
1. *f(x) > 0* for all allowed values *x*
2.The area under *f(x)* is *1.0*
3.The probability that *x* falls into an interval (e.g. *0.2 – 0.3*) is given by the integral of *f(x)* over this interval

An Example



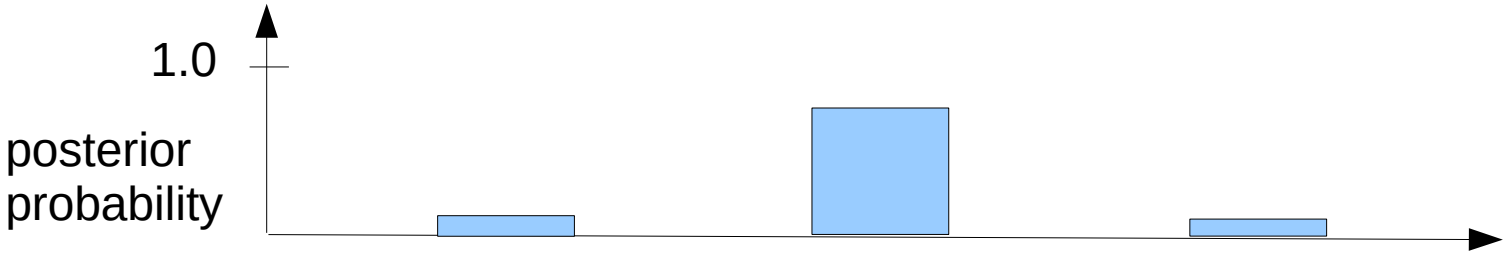
An Example





Prior distribution

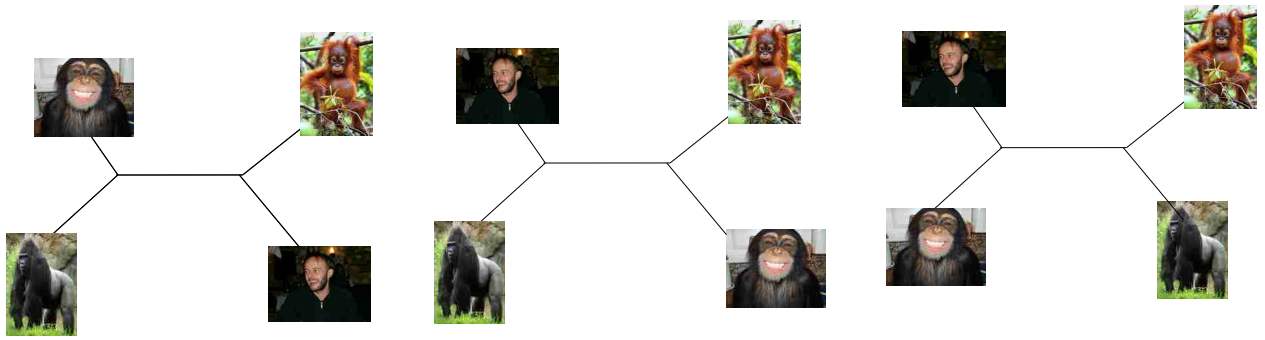
Data (observations → sequences)

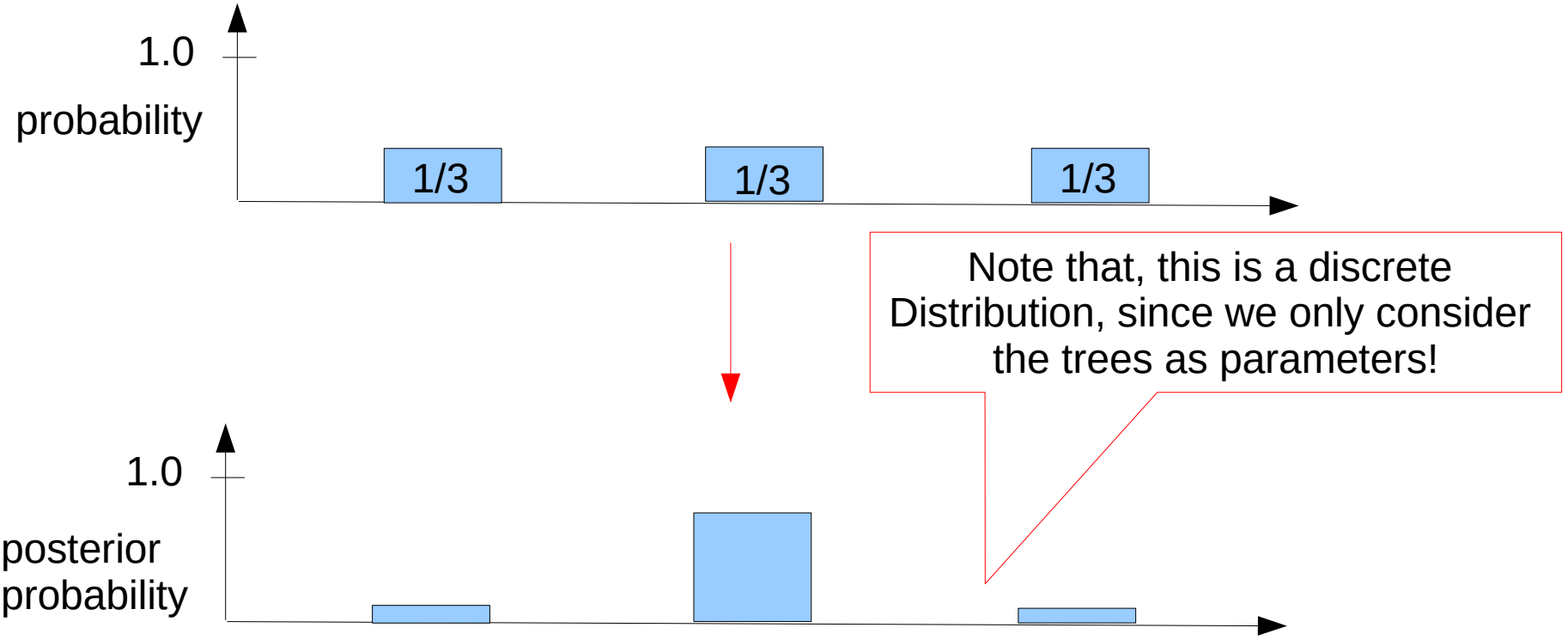


Posterior distribution

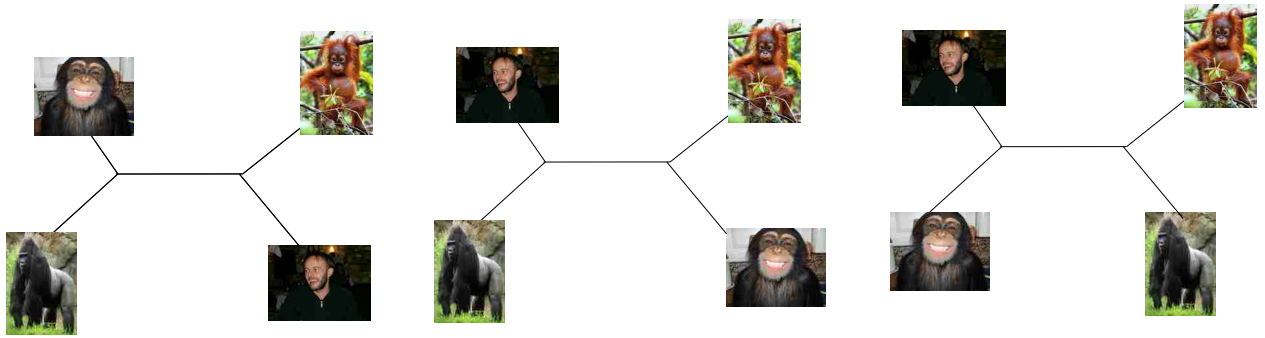
Parameter space → 3 distinct tree topologies

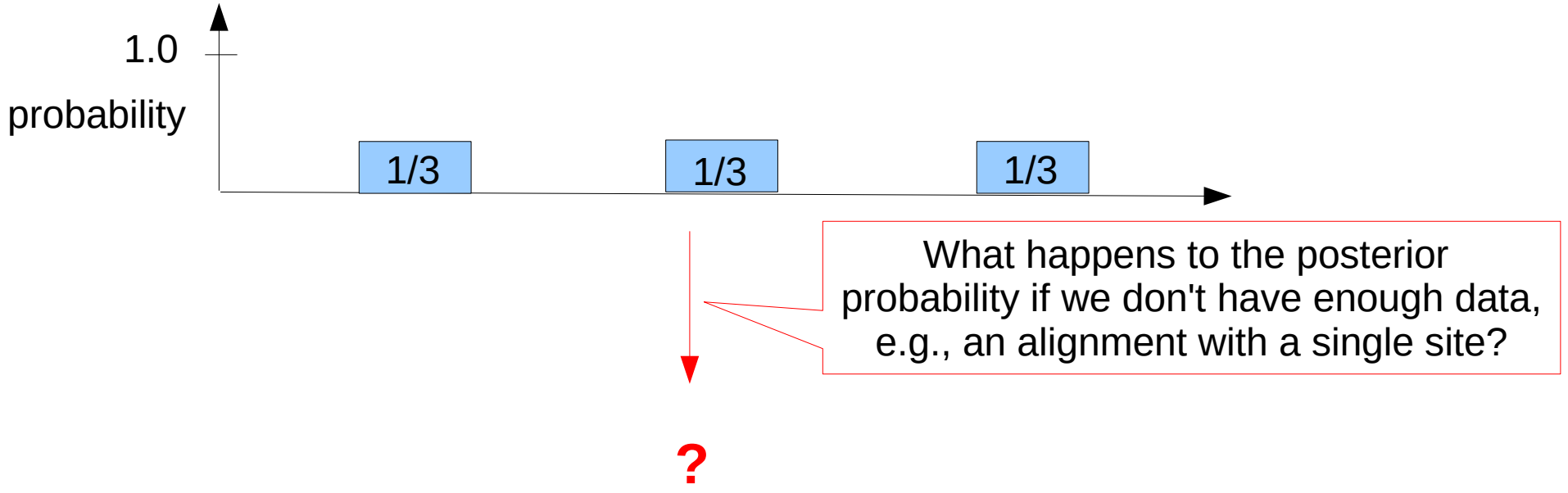
An Example




1.0
probability
1/3
1/3
1/3
Note that, this is a discrete Distribution, since we only consider the trees as parameters!
1.0
posterior probability
Parameter space → 3 distinct tree topologies

An Example



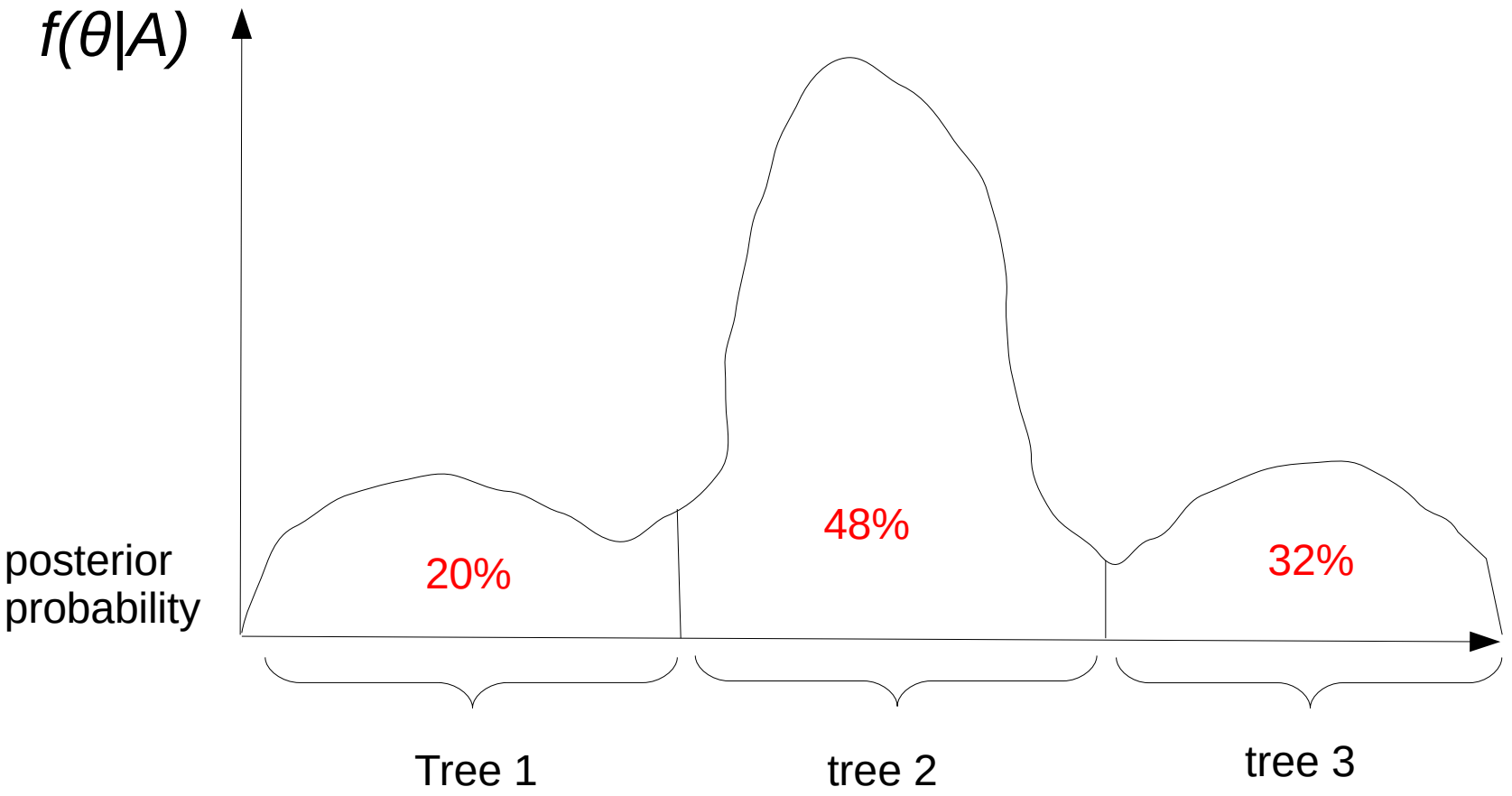

1.0
probability
1/3
1/3
1/3
What happens to the posterior probability if we don't have enough data, e.g., an alignment with a single site?
?
posterior probability

An Example

Include additional model parameters such as branch lengths,GTR rates, and the α-shape paremeter of the *Γ* distribution into the model:
θ = (tree, α, branch-lengths, GTR-rates)

An Example

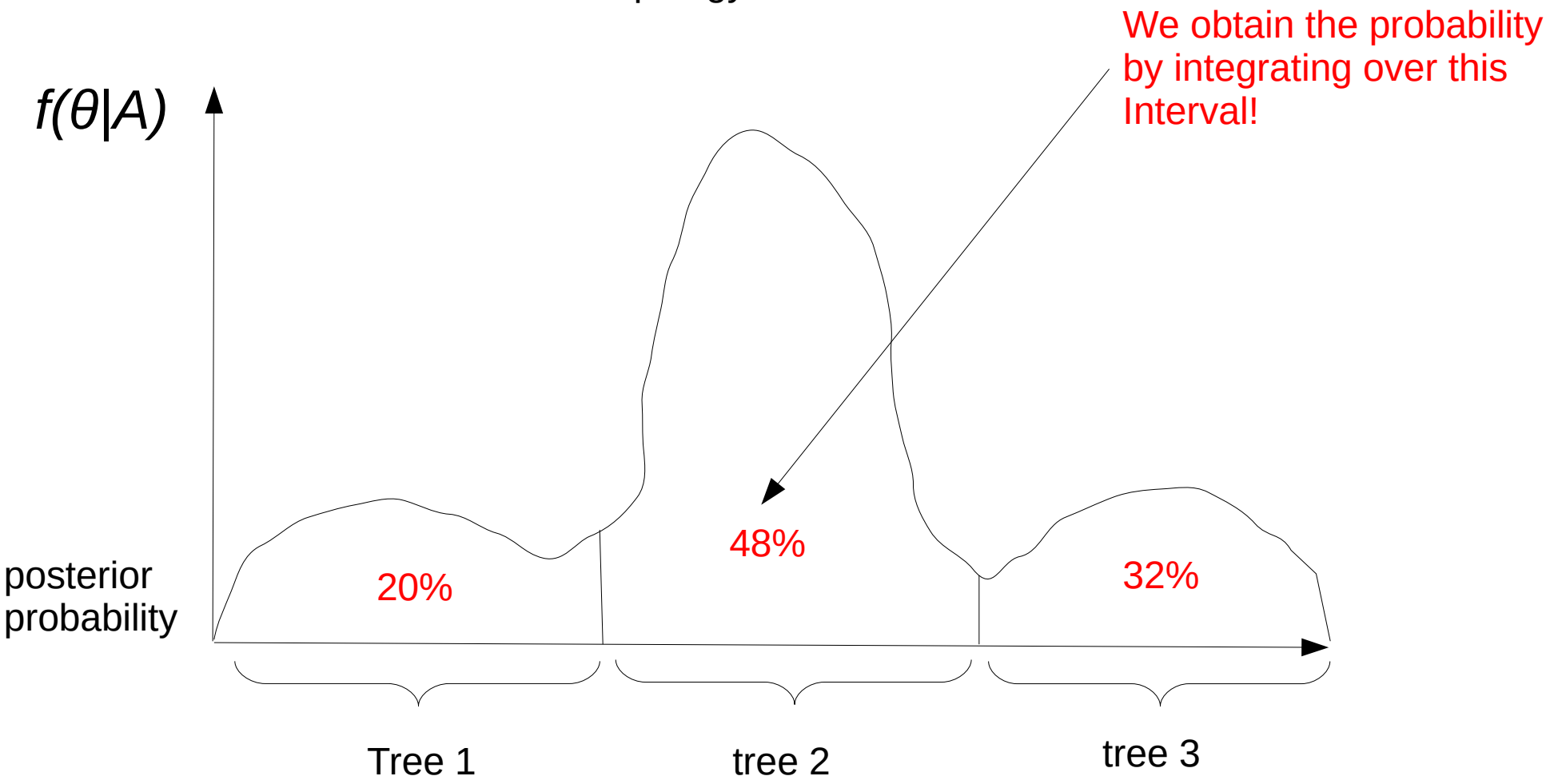
We can look at this distribution for any parameter of interest by marginalizing (integrating out) all other parameters.
Here we focus on the tree topology.



Marginal probability distribution of trees

An Example

We can look at this distribution for any parameter of interest by marginalizing (integrating out) all other parameters.
Here we focus on the tree topology.



Marginal probability distribution of trees

Marginalization

Marginal probabilities of α values

trees

	t_1	t_2	t_3	
$\alpha_1 = 0.5$	0.10	0.07	0.12	0.29
$\alpha_2 = 1.0$	0.05	0.22	0.06	0.33
$\alpha_3 = 5.0$	0.05	0.19	0.14	0.38
	0.20	0.48	0.32	**1.0**

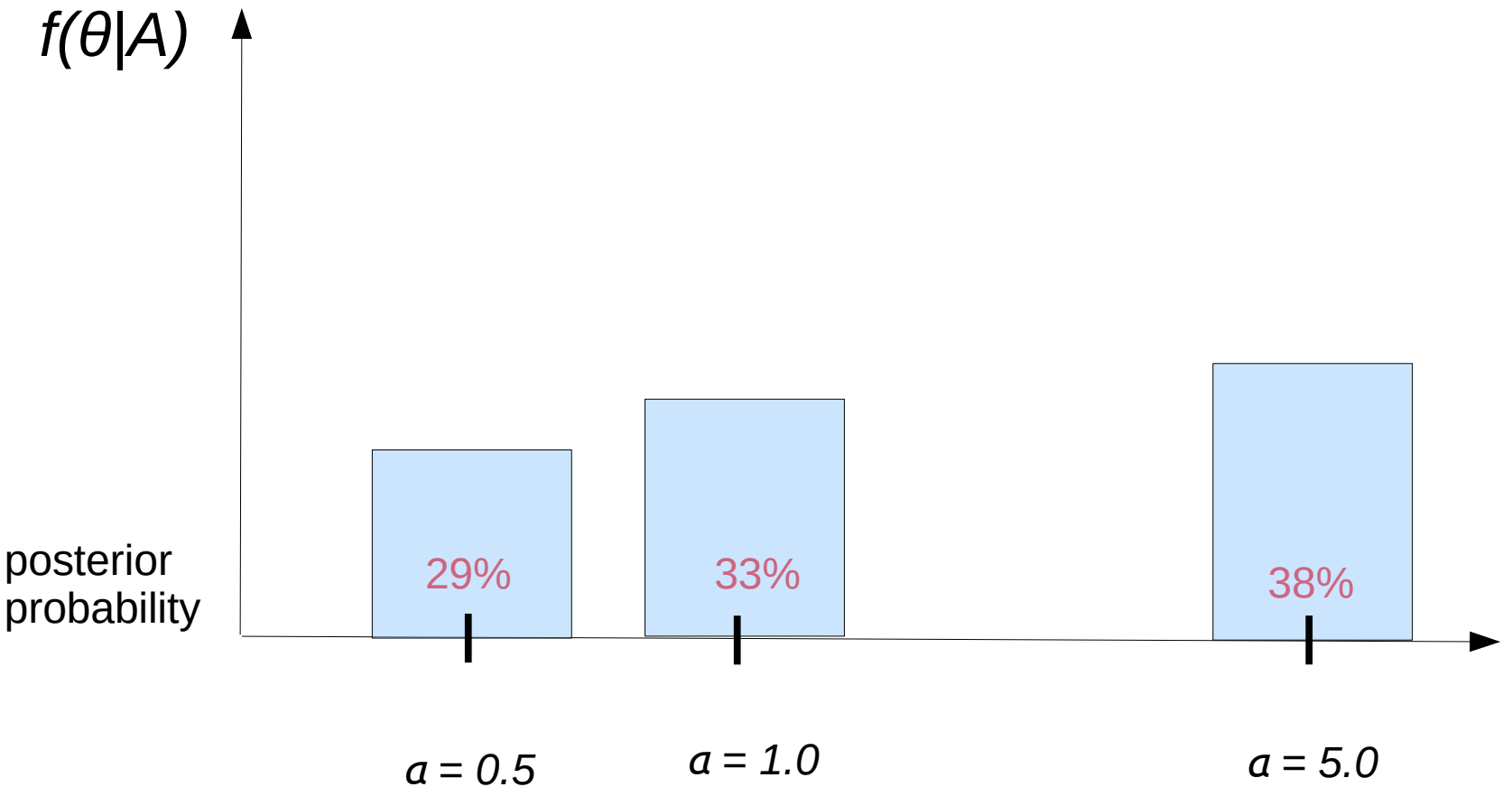
Three discrete Values of the α-shape parameter

Joint probabilities

Marginal probabilities of trees

An Example

We can look at this distribution for any parameter of interest by marginalizing (integrating out) all other parameters.
Here we focus on the three discrete α values.



Marginal probability distibution of *α*

Bayes versus Likelihood

likelihood

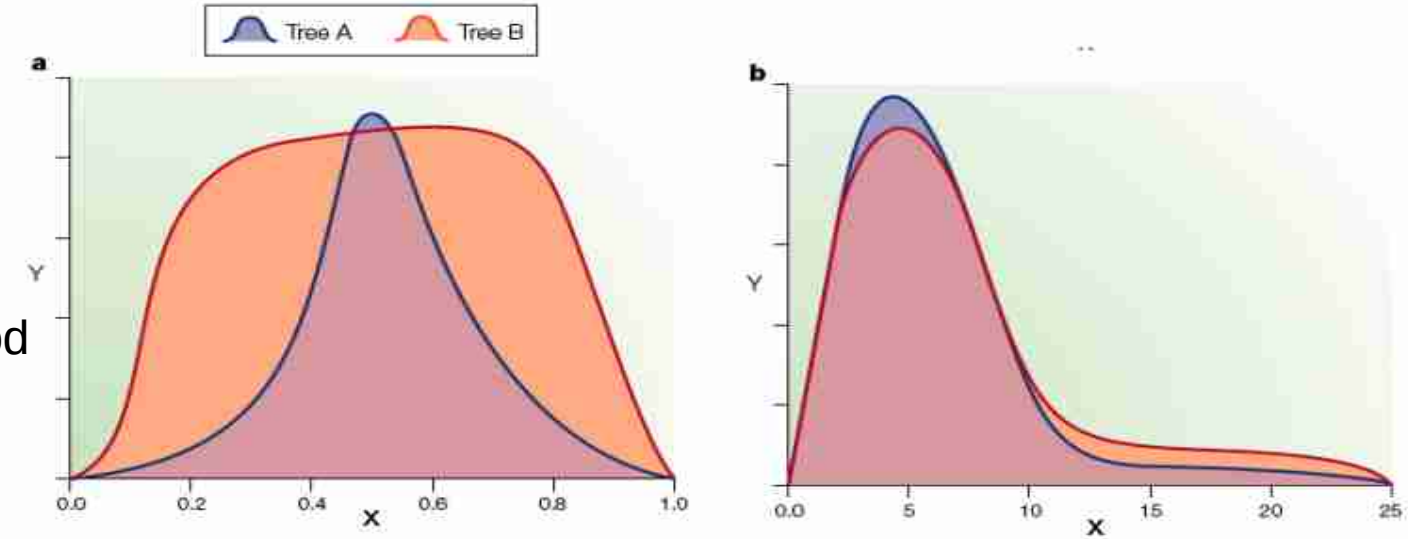


Figure 1 | **Contrast between marginal and joint estimation.** Panels **a** and **b** depict the likelihood profile for two trees versus a hypothetical parameter *x*. The *x* axis represents some nuisance parameter (for example, the ratio of the rate of transitions to the rate of transversions). The *y* axis represents the likelihood in the case of ML, or the posterior-probability density in a Bayesian approach. The area under the likelihood curve for tree A is shown in light blue, the area for tree B is shown in orange. Mauve regions are under the curve for both trees. In both cases, jointly estimating *x* and the tree favours tree A (that is, the highest peak is blue in both cases), but marginalizing over *x* favours tree B (that is, the orange area is greater than the blue area).

ML: Joint estimation
Bayesian: Marginal estimation

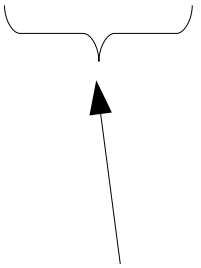
See: Holder & Lewis
“Phylogeny Estimation: traditional & Bayesian Approaches” Link to paper

Outline

- Bayesian statistics
- Monte-Carlo simulation & integration
- Markov-Chain Monte-Carlo methods
- Metropolis-coupled MCMC-methods
- Some phylogenetic proposals
- Reversible jump MCMC

Bayes Theorem General Form

$$f(\theta|A) = (likelihood * prior) / ouch$$



Marginal likelihood
Normalization constant
→ difficult to calculate

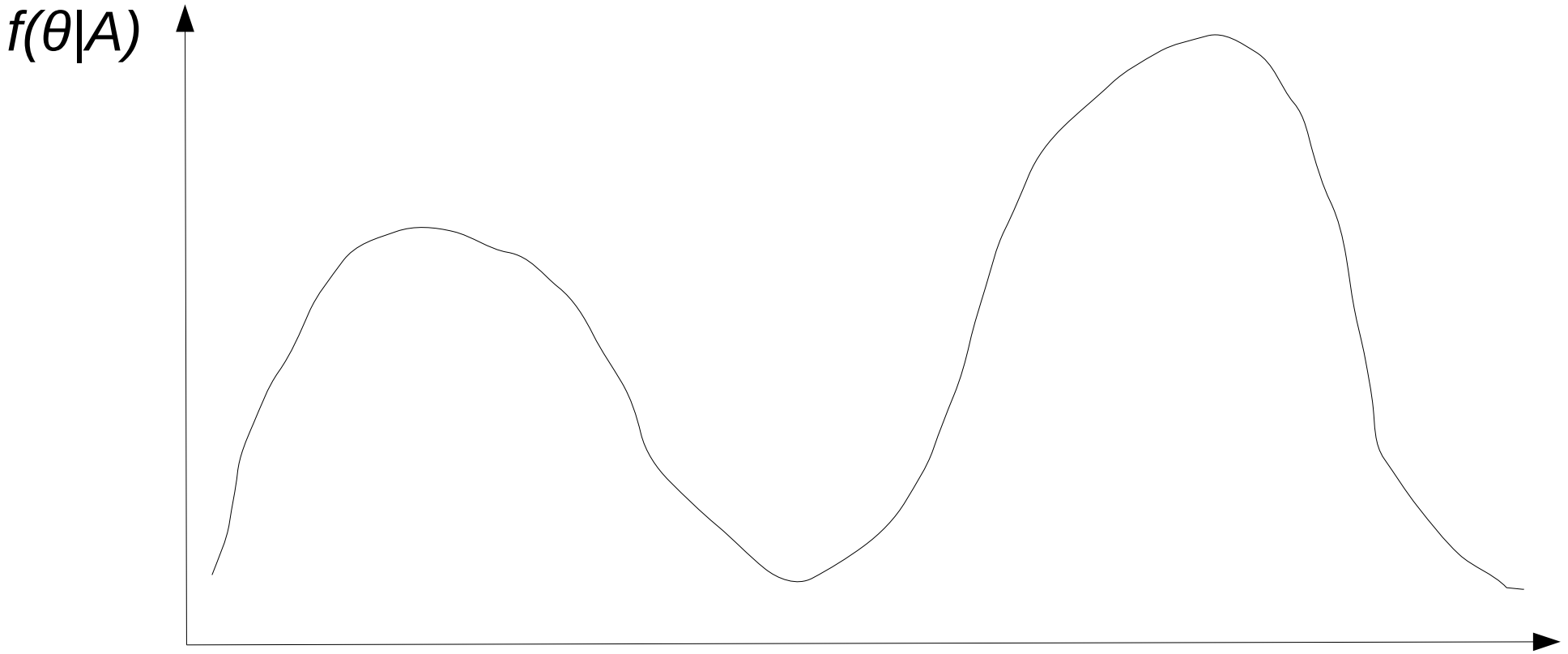
We know how to compute $f(A|\theta)$ → the likelihood of the tree

Problems:

Problem 1: $f(\theta)$ is given a priori, but how do we chose an appropriate distribution?
→ biggest strength and weakness of Bayesian approaches

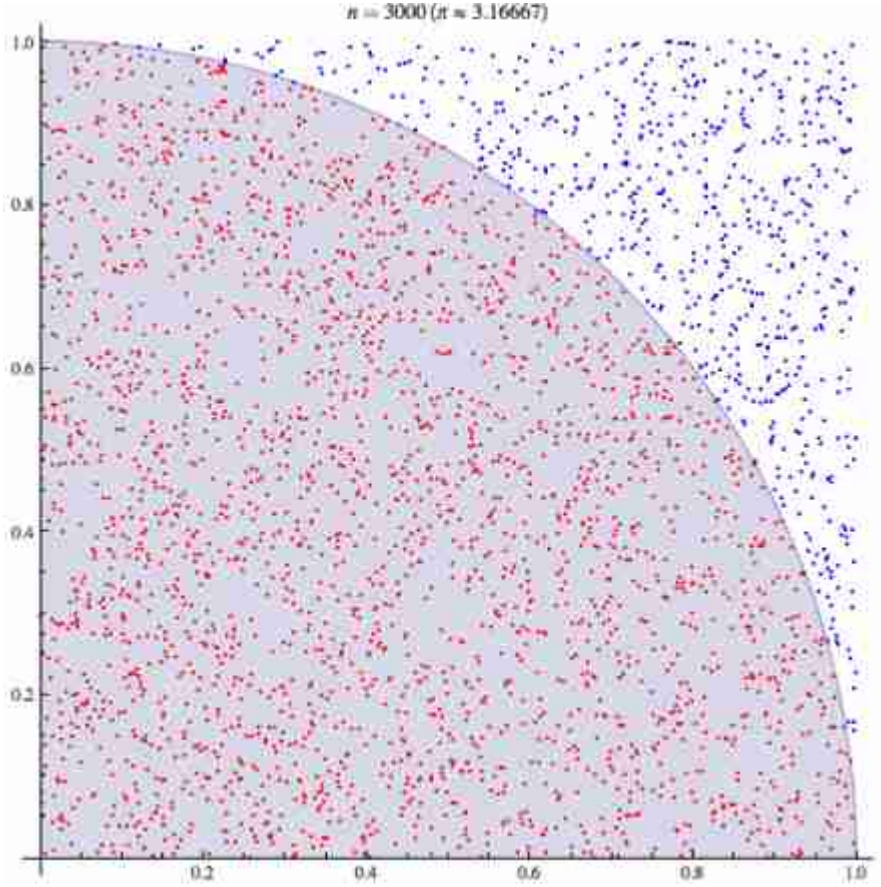
Problem 2: How can we calculate/approximate $\int f(\theta)f(A|\theta)d\theta$
→ to explain this we need to introduce additional machinery to design methods for numerical integration

How can we compute this integral?



The Classic Example

- Calculating π (the geometric constant!) with Monte-Carlo



Procedure:
1. Randomly throw points onto the rectangle n times
2. Count how many points fall into the circle n_i
3. determine π as the ratio n / n_i

→ this yields an approximation of the ratio of the areas (the square and the circle)

Monte Carlo Integration

- Method for numerical integration of *m*-dimensional integrals over *R*:

  $\int f(\theta)d\theta \approx 1/N \sum f(\theta_i)$

  where θ *is from domain* R^m

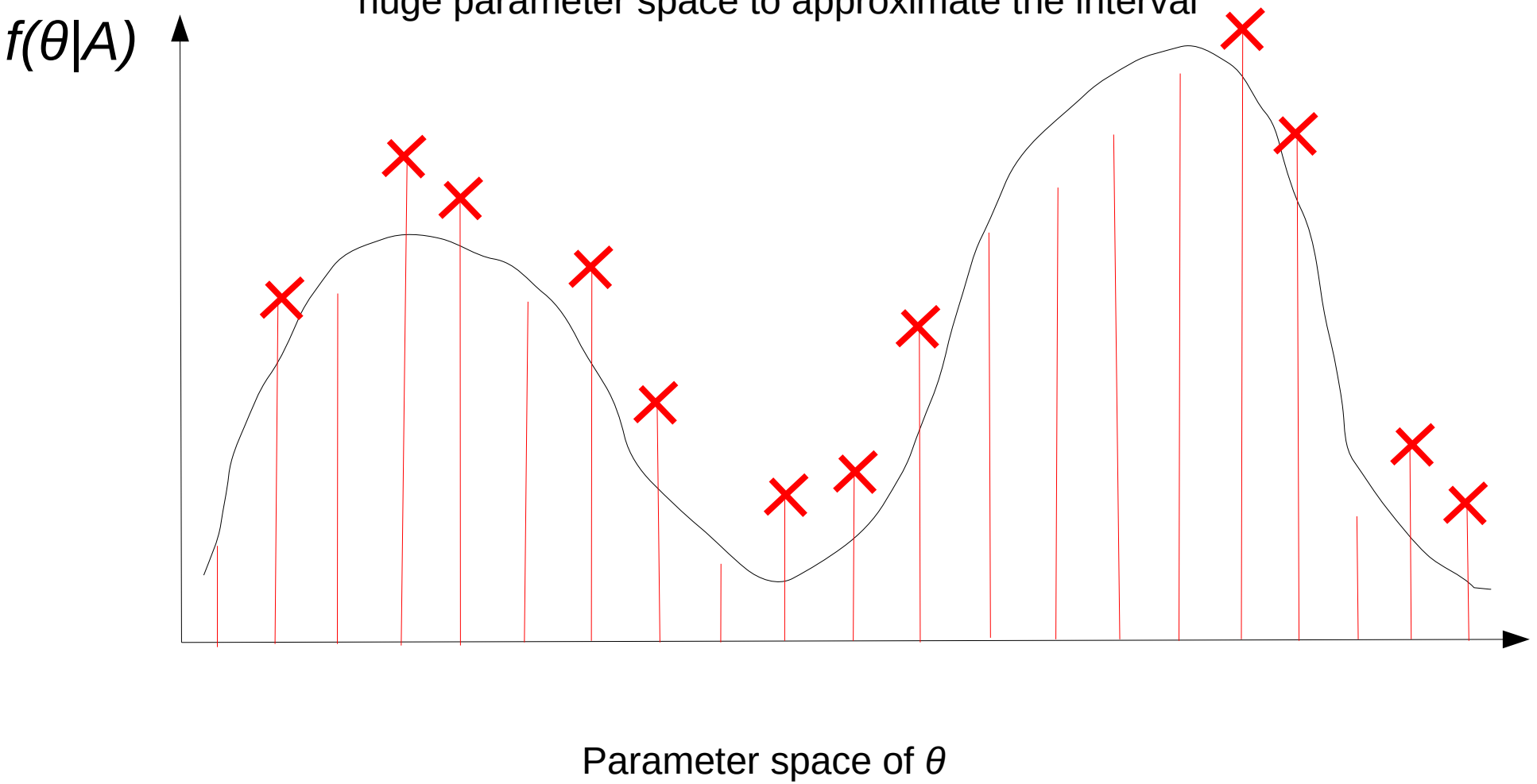
- More precisely, if the integral $\int$ *is* defined over a domain/volume *V* the equation becomes: $V * 1/N * \sum f(\theta_i)$
- Key issues:
  - Monte Carlo simulations draw samples θ_i of function *f()* completely at random → random grid
  - How many points do we need to sample for a 'good' approximation?
  - Domain R^m might be too large for random sampling!

Outline

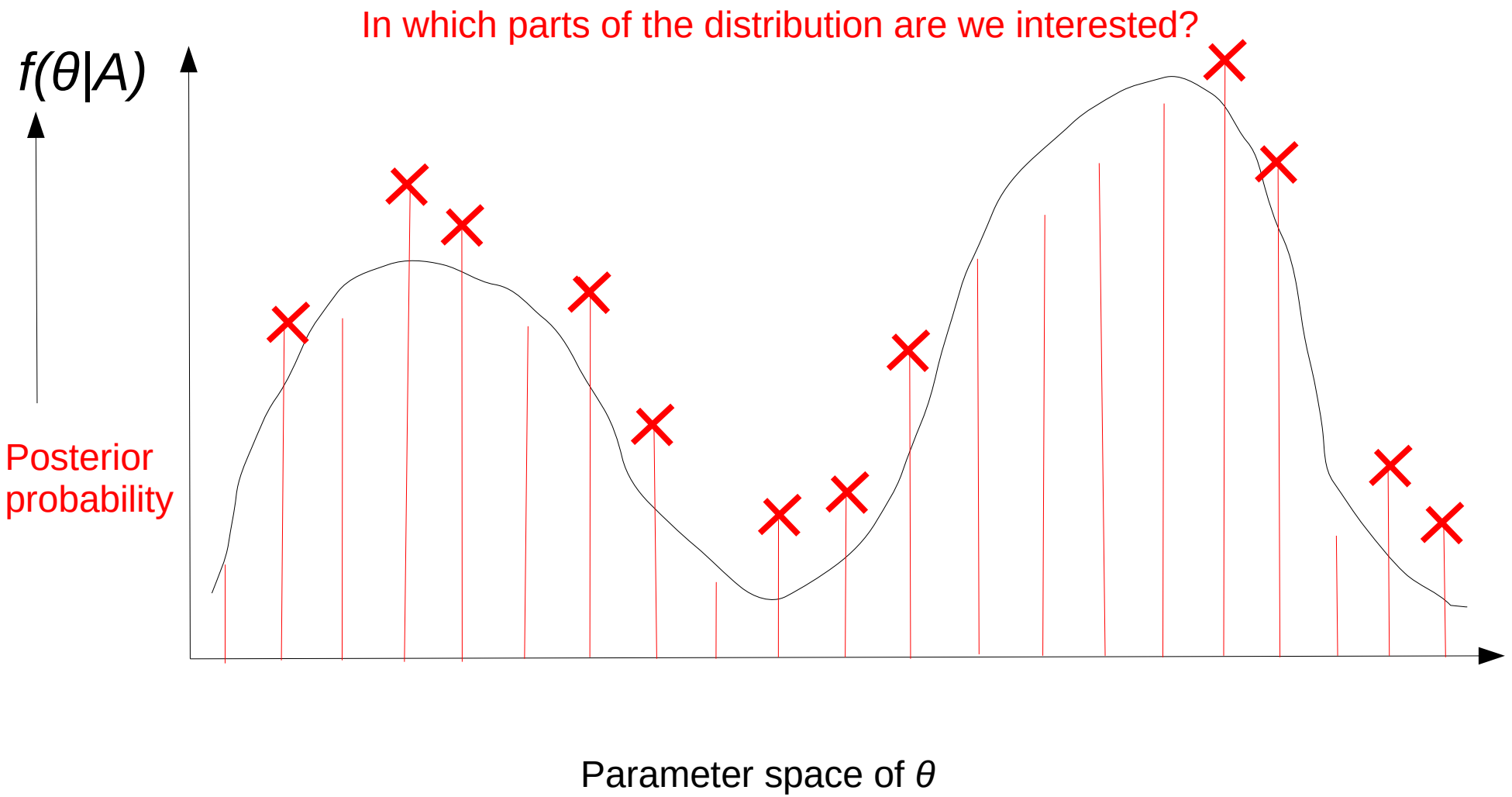
- Bayesian statistics
- Monte-Carlo simulation & integration
- Markov-Chain Monte-Carlo methods
- Metropolis-coupled MCMC-methods
- Some phylogenetic proposals
- Reversible jump MCMC

How can we compute this integral?

Monte-Carlo Methods: randomly sample data-points in this huge parameter space to approximate the interval

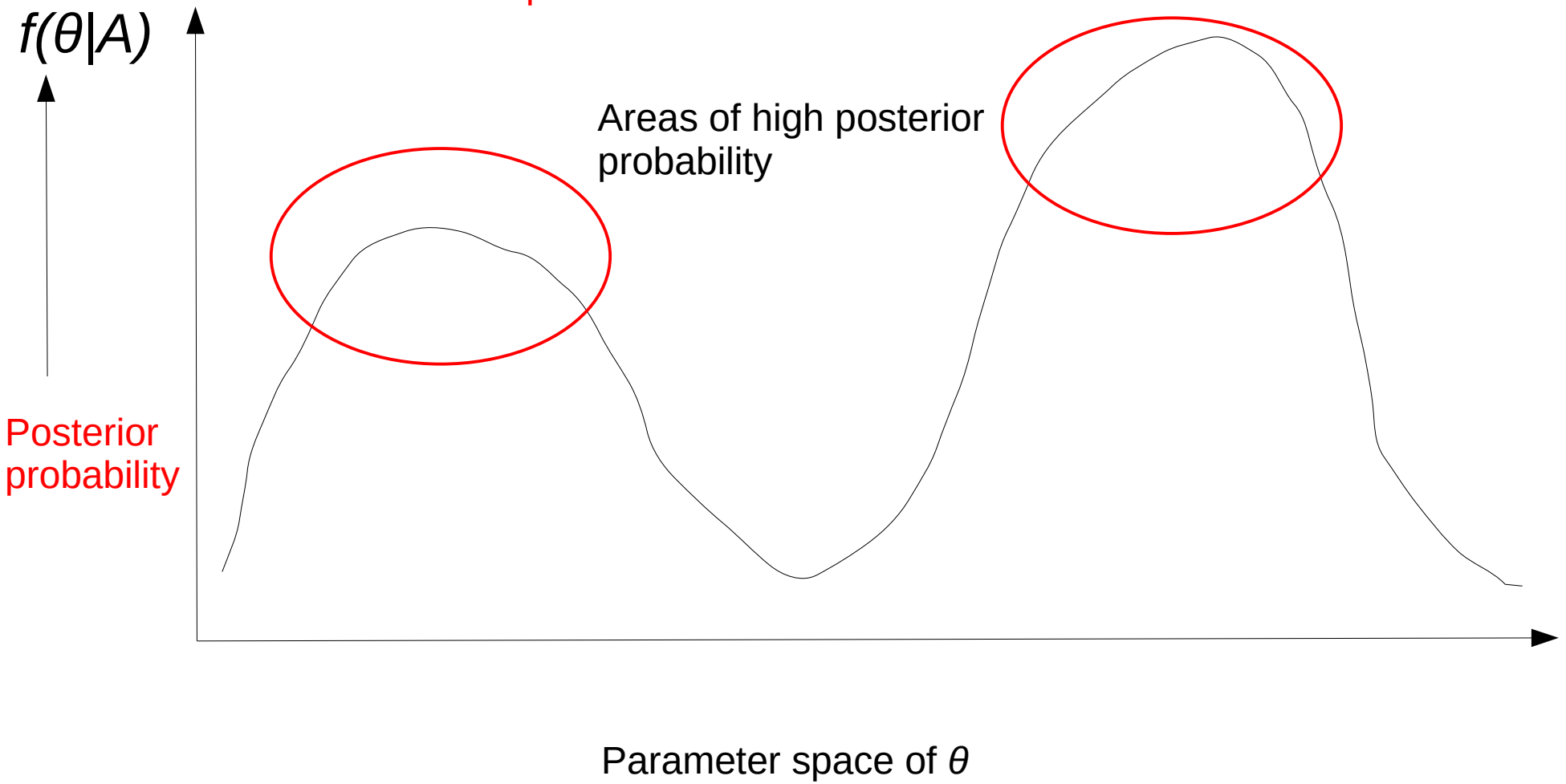


How can we compute this integral?

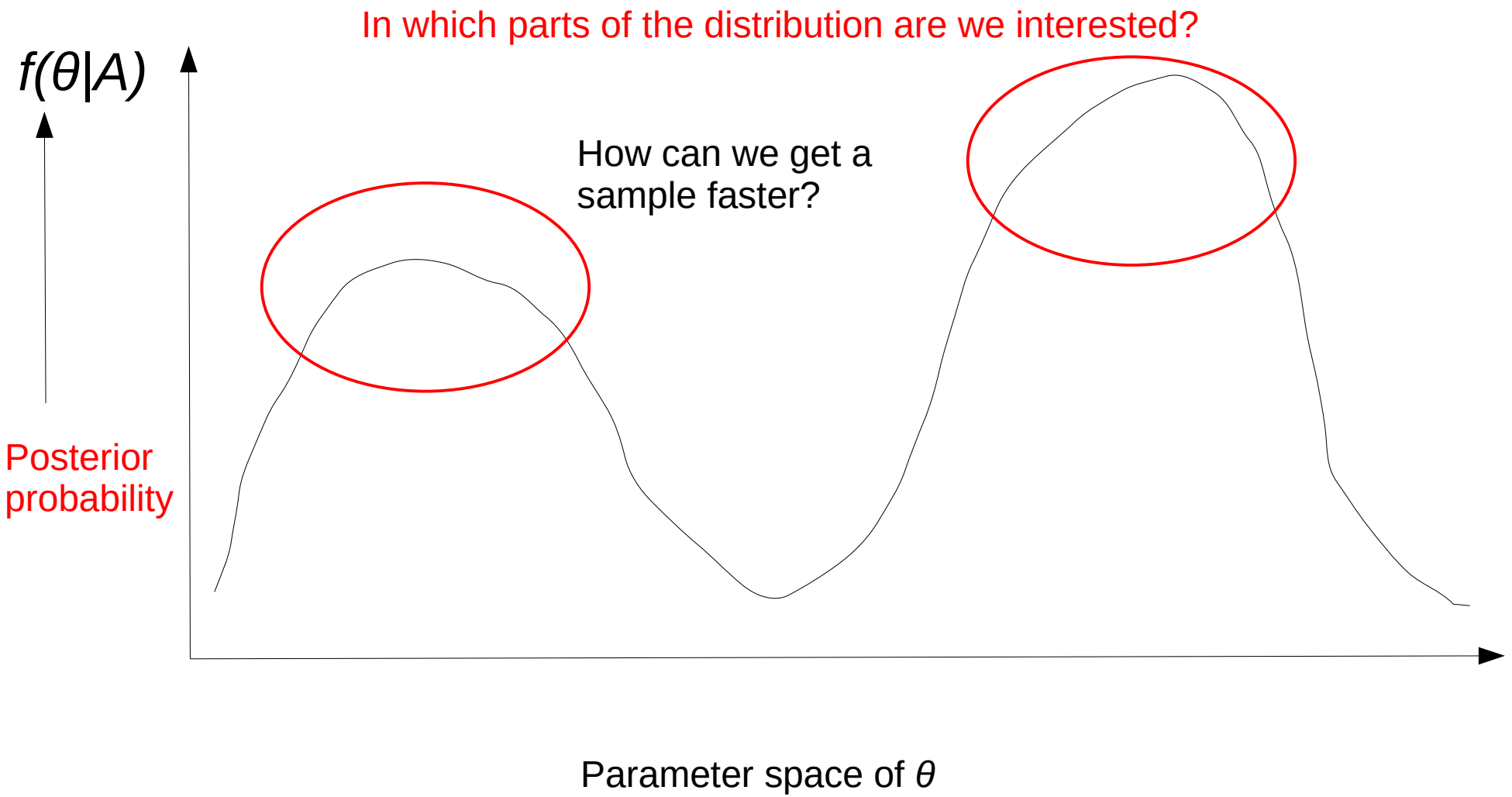


Distribution Landscape

In which parts of the distribution are we interested?

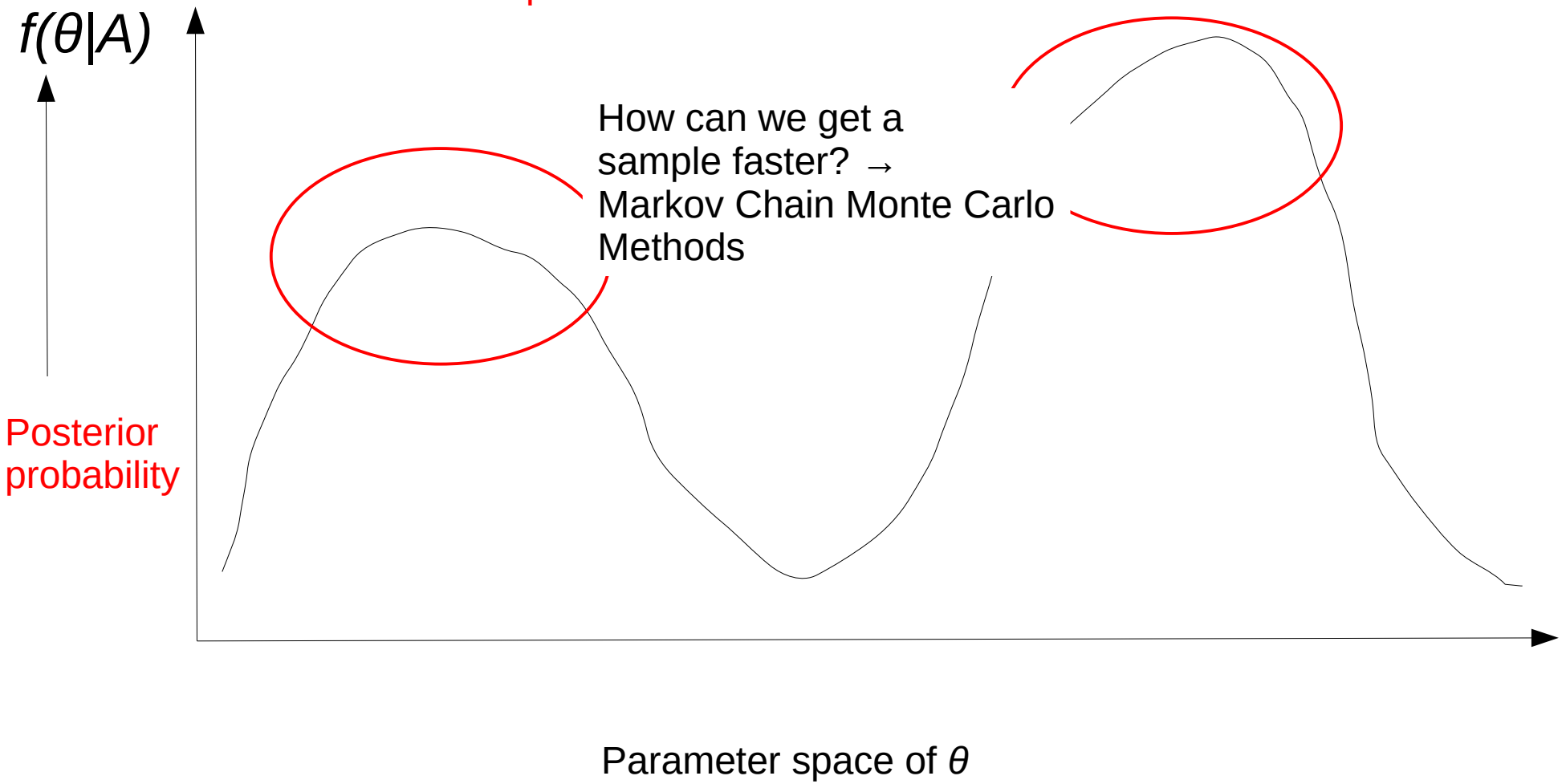


Distribution Landscape



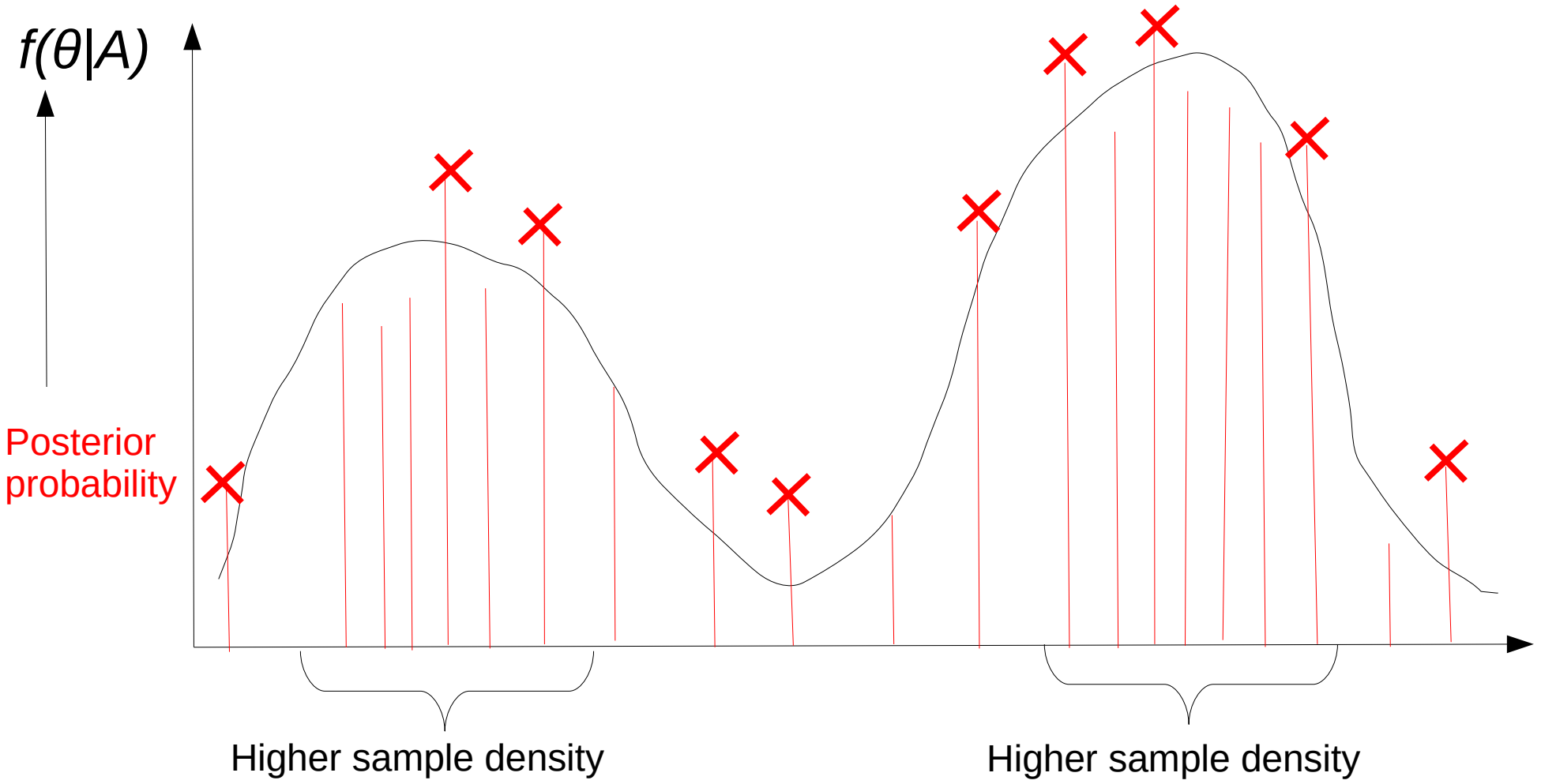
Distribution Landscape

In which parts of the distribution are we interested?



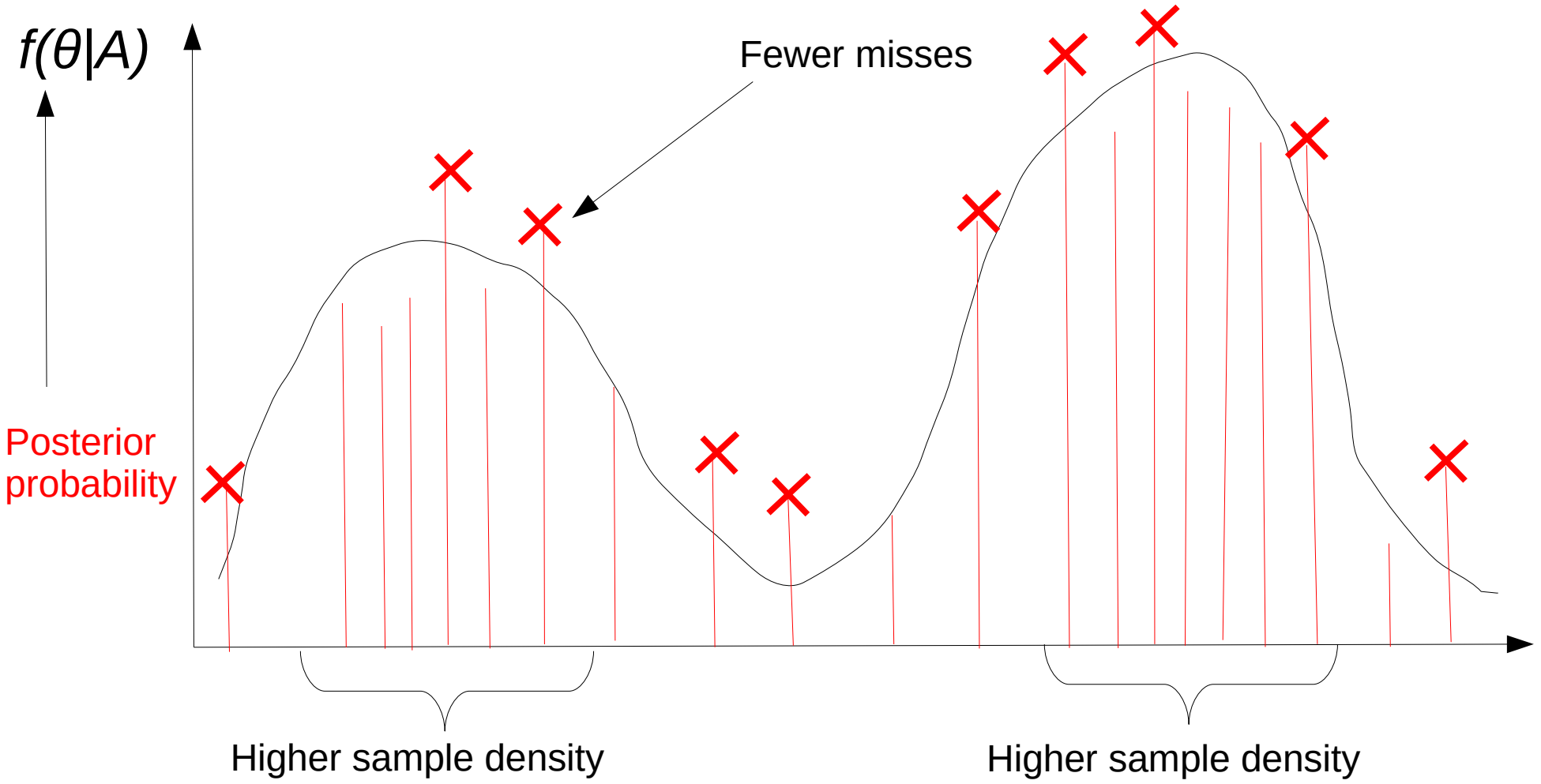
Distribution Landcsape

In which parts of the distribution are we interested?



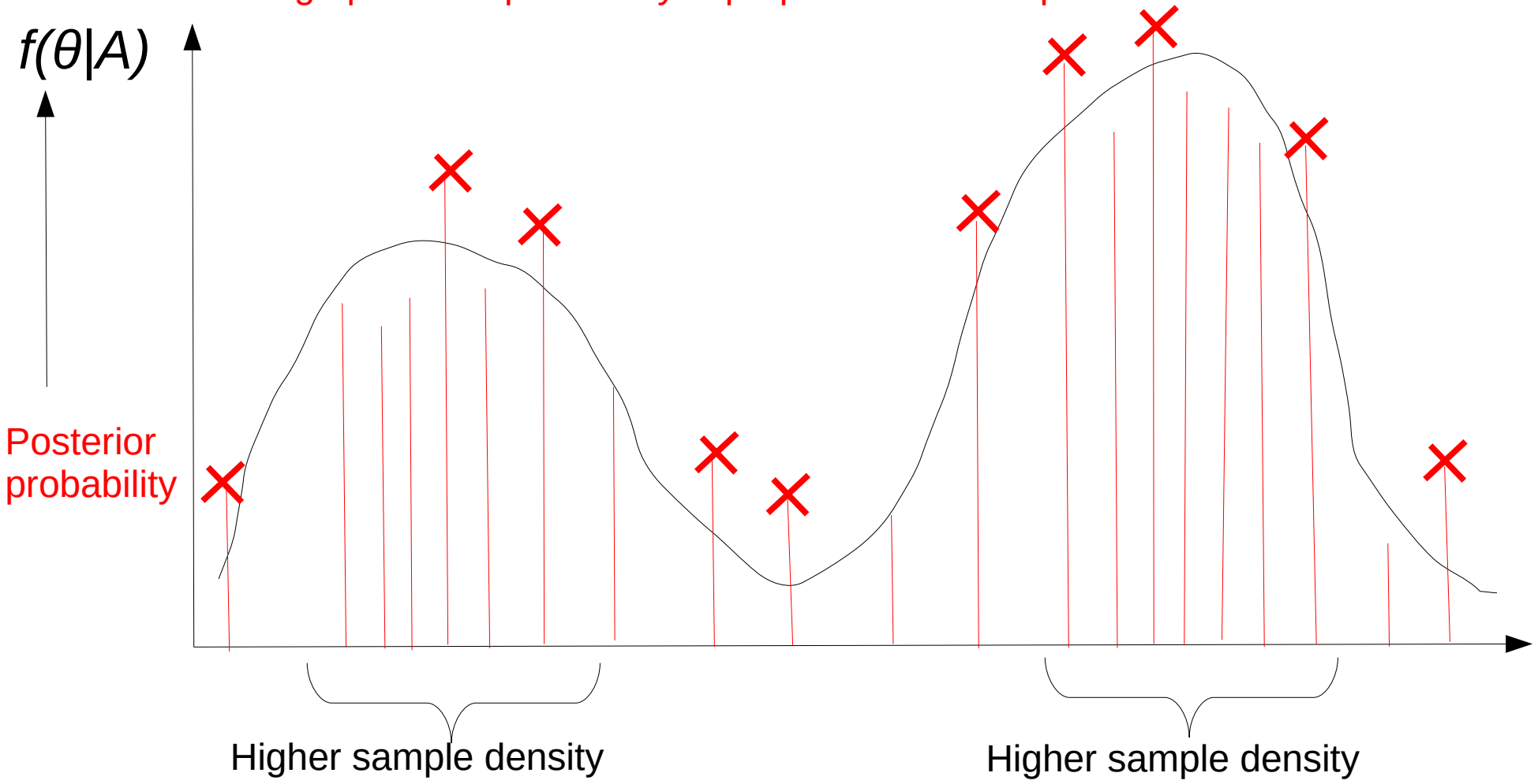
Distribution Landcsape

In which parts of the distribution are we interested?



Markov-Chain Monte-Carlo

MCMC → biased random walks: the probability to evaluate/find a sample in an area with high posterior probability is proportional to the posterior distribution



Markov-Chain Monte-Carlo

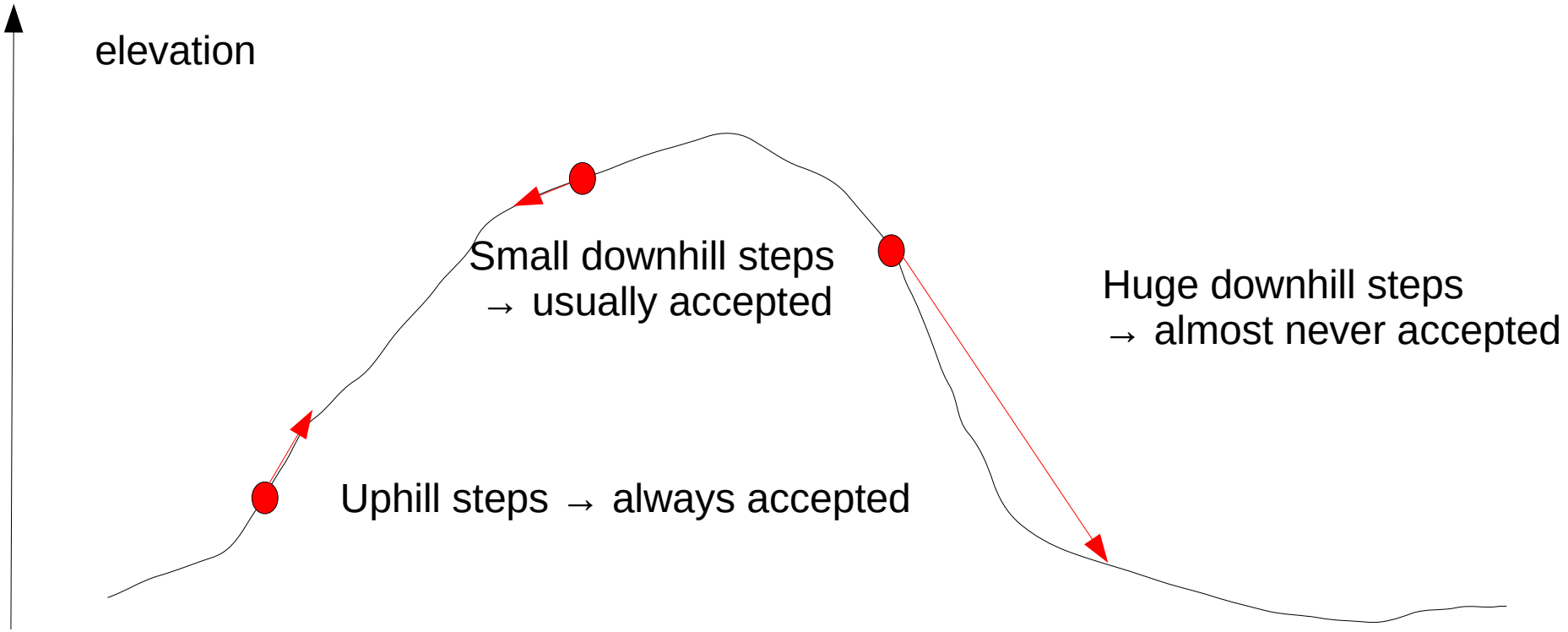
- Idea: Move the grid/samples into regions of high probability
- Construct a Markov Chain that generates samples such that more time is spent (more samples are evaluated) in the most interesting regions of the state space
- MCMC can also be used for hard CS optimization problems, for instance, the knapsack problem
- Note that, MCMC is similar to Simulated Annealing → there's no time to go into the details though here!

The Robot Metaphor

Crete, December 2019

The Robot Metaphor

- Drop a robot onto an unknown planet to explore its landscape
- Teaching idea and slides adapted from Paul O. Lewis



How to accept/reject proposals

- Decision to accept/reject a proposal to go from *Point 1* → *Point 2* is based on the ratio *R* of posterior densities of the two points/samples

  $R = Pr(Point2|data) / Pr(point1|data) =$

  $(Pr(Point2)Pr(data|point2) / Pr(data)) / (Pr(Point1)Pr(data|point1) / Pr(data))$

  $= Pr(point2)Pr(data|point2) / Pr(point1)Pr(data|point1)$

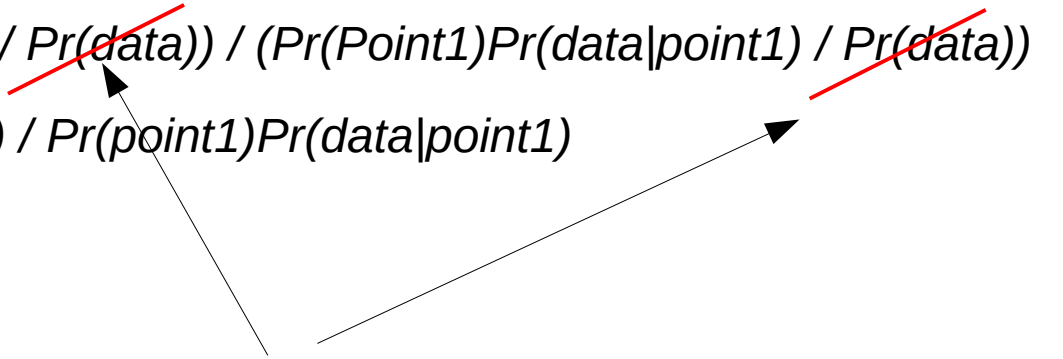
How to accept/reject proposals

- Decision to accept/reject a proposal to go from *Point 1* → *Point 2* is based on the ratio *R* of posterior densities of the two points/samples

R = Pr(Point2|data) / Pr(point1|data) =

(Pr(Point2)Pr(data|point2) / Pr(data)) / (Pr(Point1)Pr(data|point1) / Pr(data))

= Pr(point2)Pr(data|point2) / Pr(point1)Pr(data|point1)



The marginal probability of the data cancels out!
Phew, we don't need to compute it.

How to accept/reject proposals

- Decision to accept/reject a proposal to go from *Point 1* → *Point 2* is based on the ratio *R* of posterior densities of the two points/samples

R = Pr(Point2|data) / Pr(point1|data) =

(Pr(Point2)Pr(data|point2) / Pr(data)) / (Pr(Point1)Pr(data|point1) / Pr(data)) =

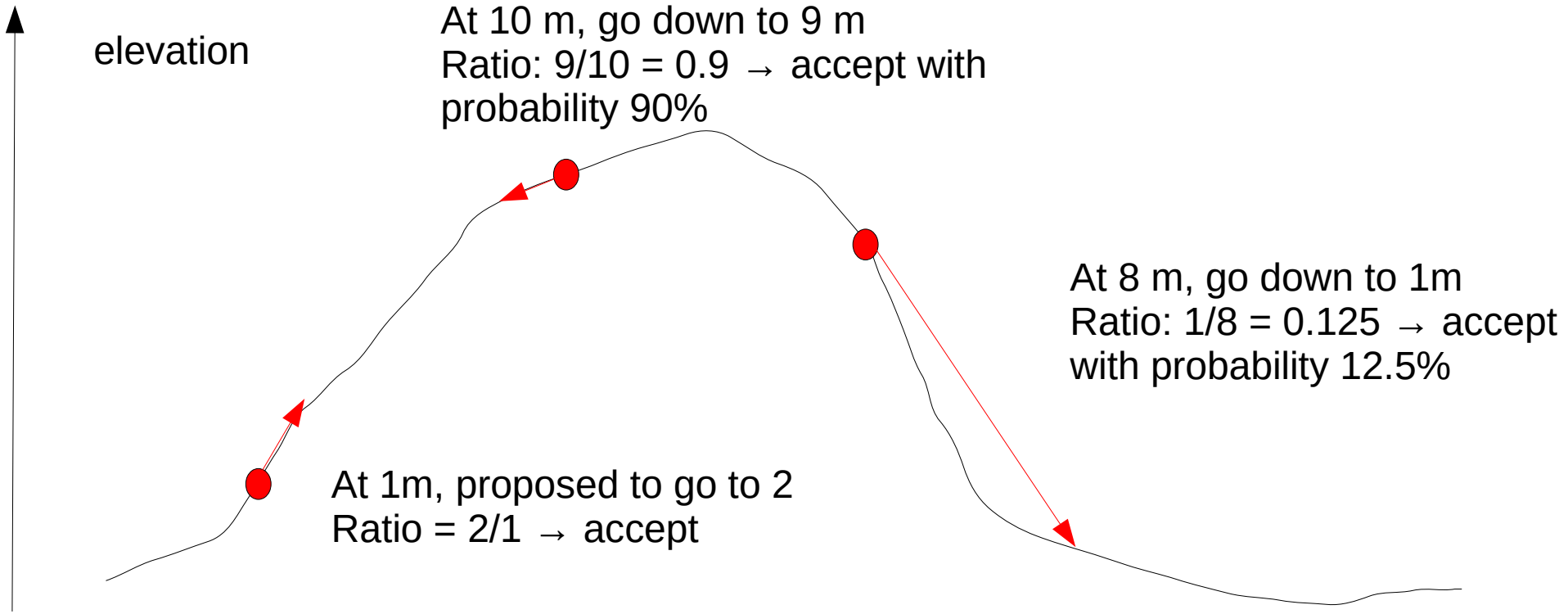
*(Pr(point2)/Pr(point1)) * (Pr(data|point2) / Pr(data|point1))*

Prior ratio: for uniform priors this is 1 !

Likelihood ratio

The Robot Metaphor

- Drop a robot onto an unknown planet to explore its landscape

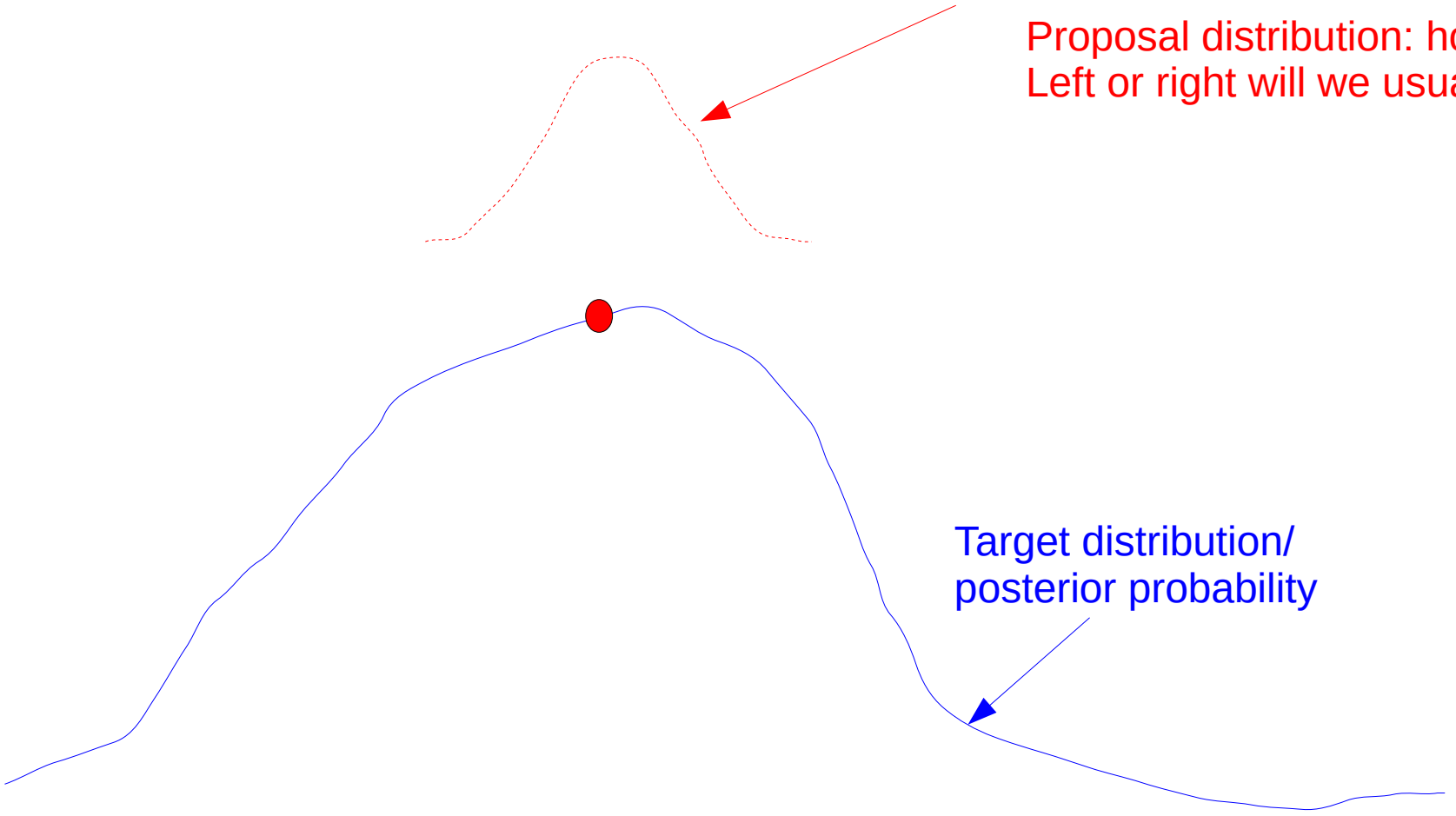


Distributions

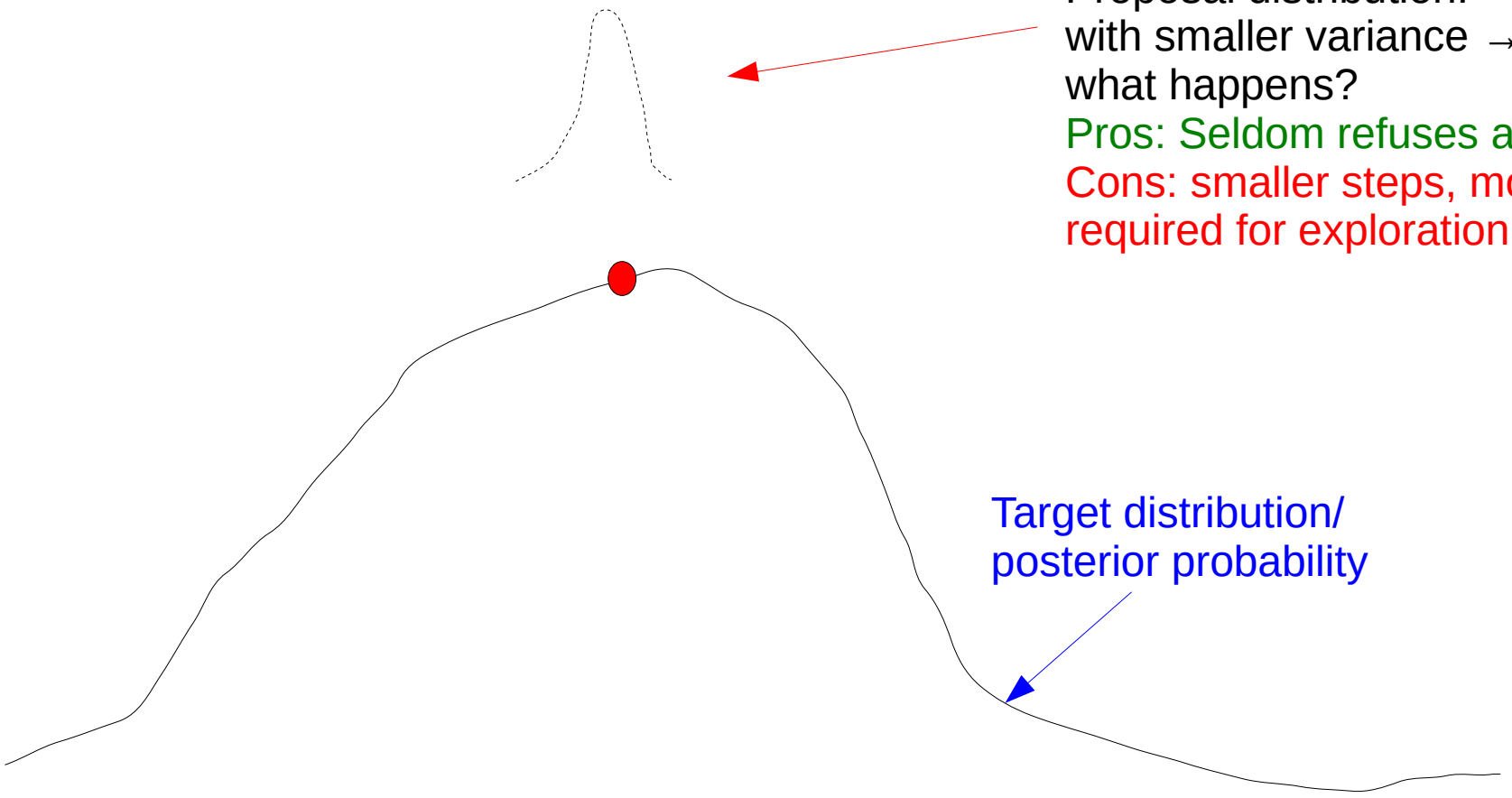
- The target distribution is the posterior distribution we are trying to sample (integrate over)!
- The proposal distribution decides which point (how far/close) in the landscape to randomly go to/try next:

  → The choice has an effect on the efficiency of the MCMC algorithm, that is, how fast it will get to these interesting areas we want to sample

The Robot Metaphor



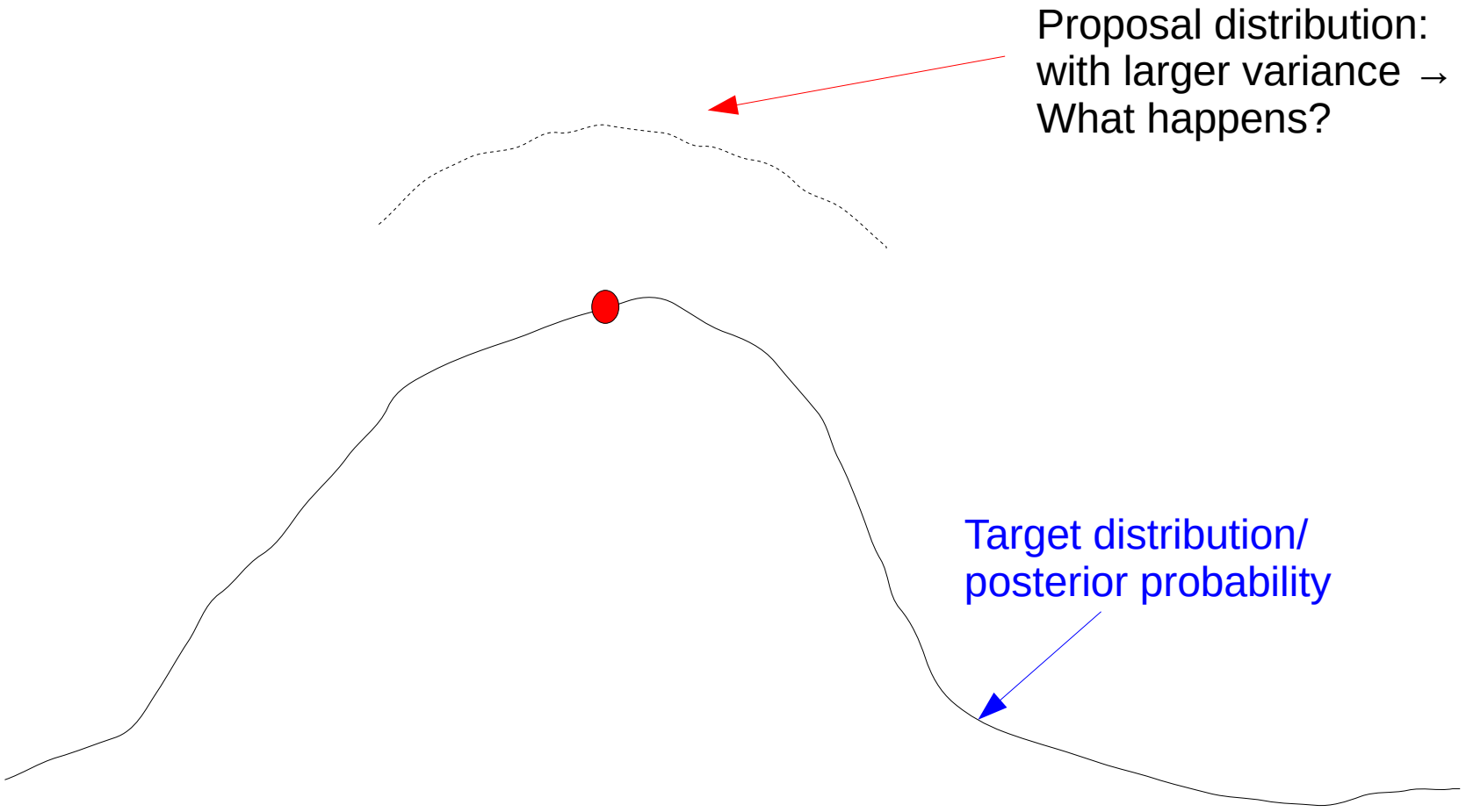
The Robot Metaphor



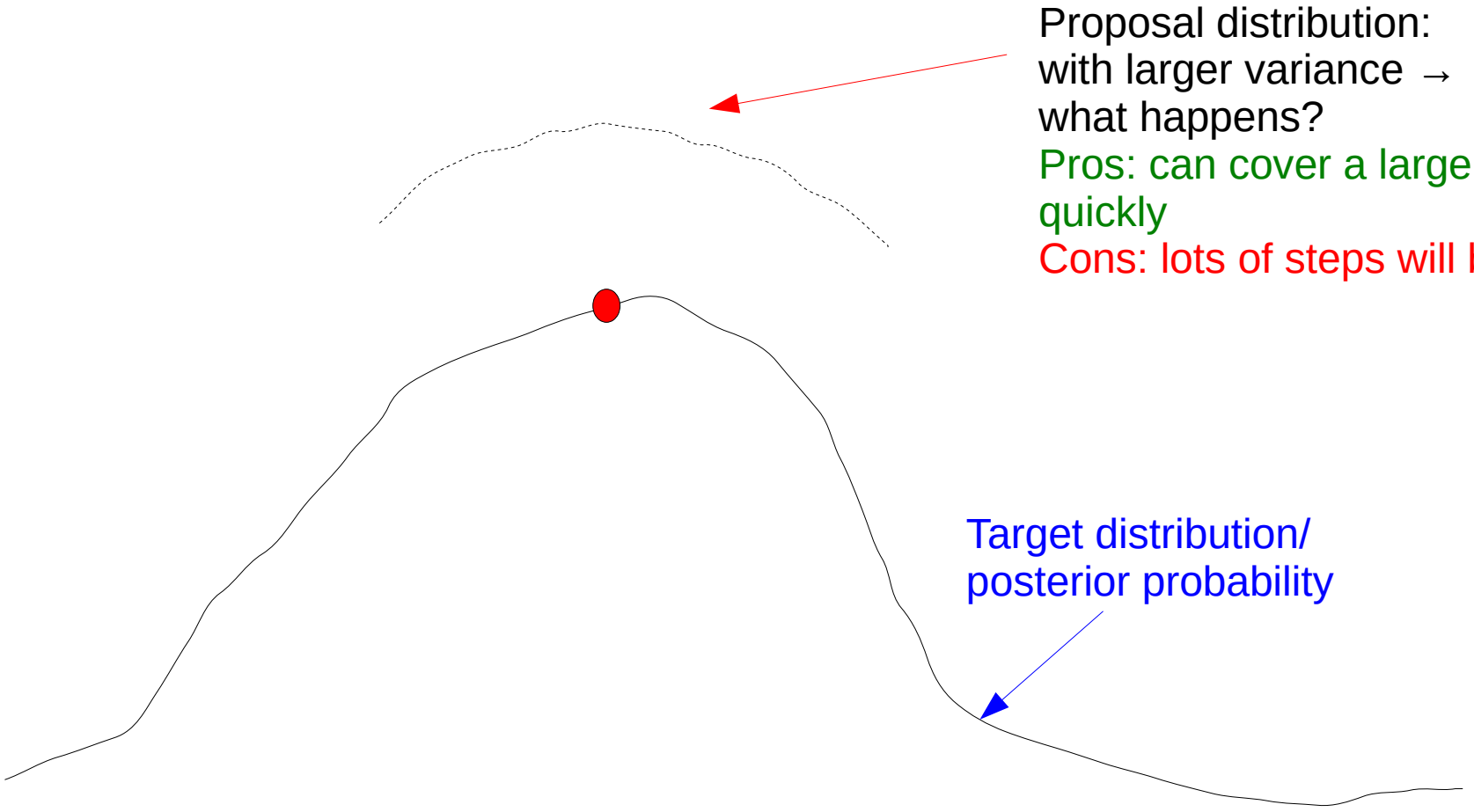
Proposal distribution:
with smaller variance →
what happens?
Pros: Seldom refuses a step
Cons: smaller steps, more steps required for exploration

Target distribution/
posterior probability

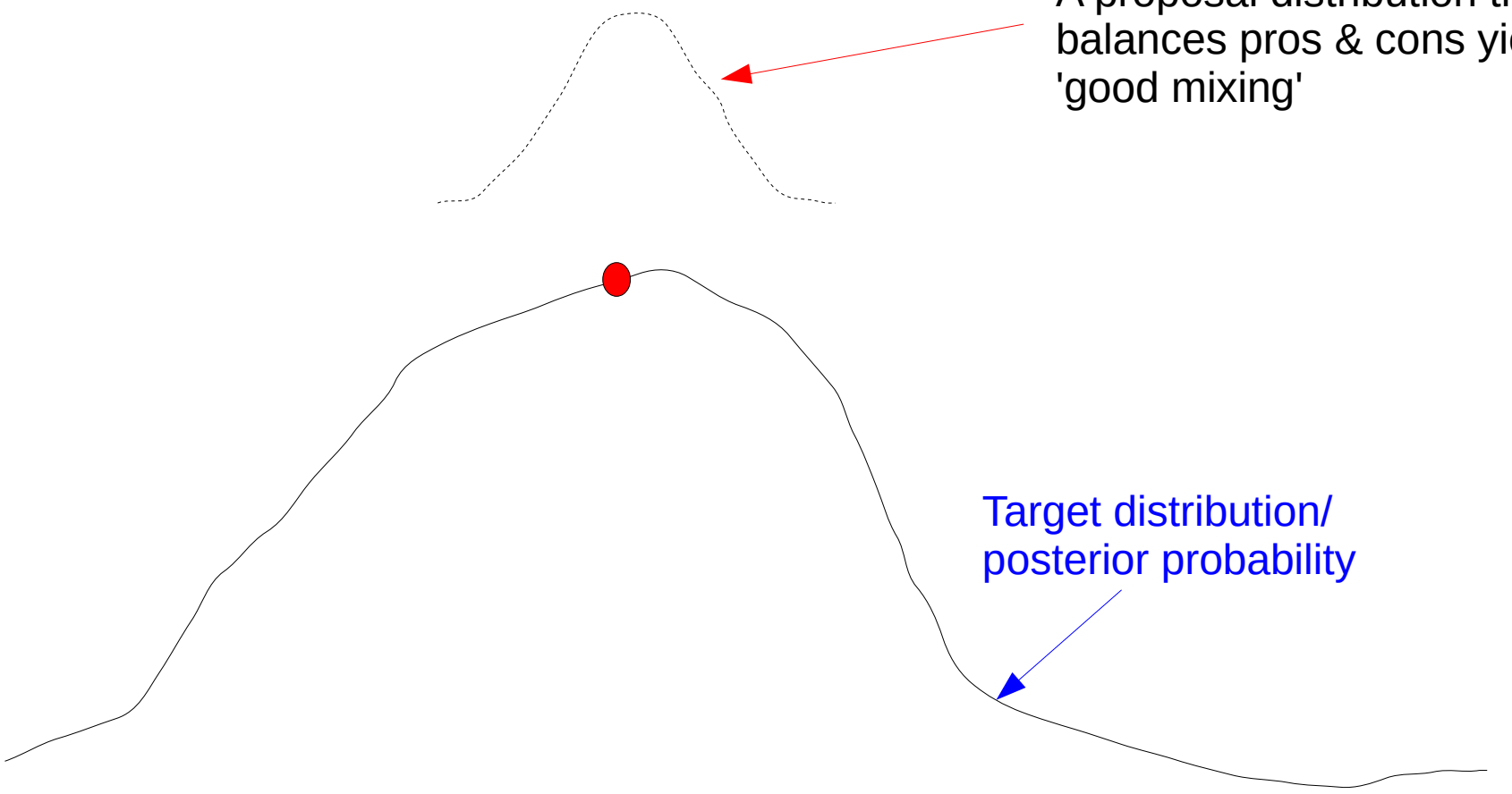
The Robot Metaphor



The Robot Metaphor



The Robot Metaphor

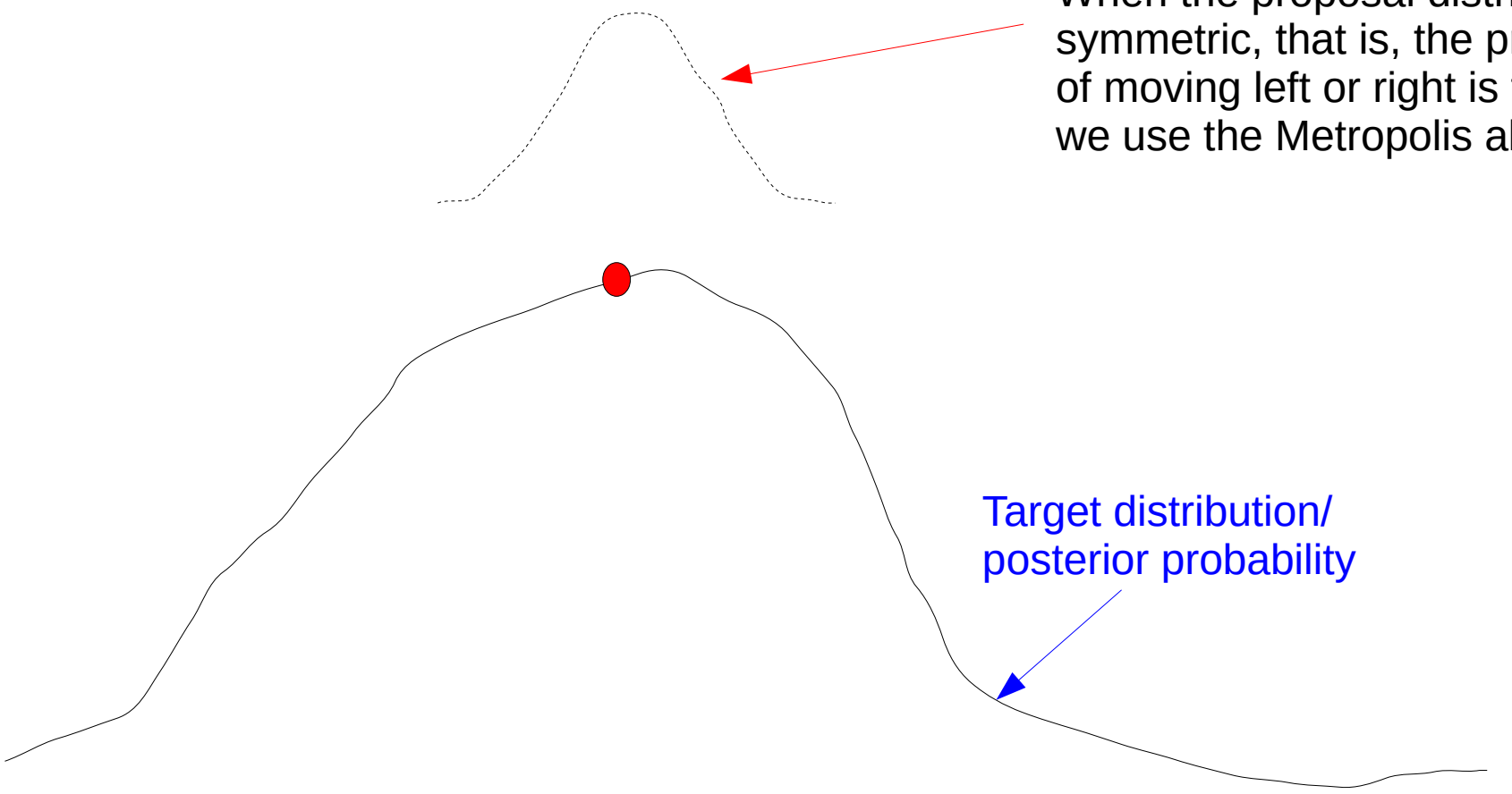


Mixing

- A well-designed chain will require a few steps until reaching convergence, that is, approximating the underlying probability density function 'well-enough' from a random starting point
- It is a somewhat fuzzy term, refers to the proportion of accepted proposals (acceptance ratio) generated by a proposal mechanism → should be neither too low, nor too high
- The real art in designing MCMC methods consists
  - building & tuning good proposal mechanisms
  - selecting appropriate proposal distributions
  - such that they quickly approximate the distribution we want to sample from

The Robot Metaphor

When the proposal distribution is symmetric, that is, the probability of moving left or right is the same, we use the Metropolis algorithm



The Metropolis Algorithm

- Metropolis *et al.* 1953 http://www.aliquote.org/pub/metropolis-et-al-1953.pdf
- Initialization: Choose an arbitrary point θ_0 as first sample
- Choose an arbitrary probability density $Q(\theta_{i+1}|\theta_i)$ which suggests a candidate for the next sample θ_{i+1} given the previous sample θ_i.
- For the Metropolis algorithm, $Q()$ must be symmetric:

  it must satisfy $Q(\theta_{i+1}|\theta_i) = Q(\theta_i|\theta_{i+1})$
- For each iteration i:
  - Generate a candidate θ^* for the next sample by picking from the distribution $Q(\theta^*|\theta_i)$
  - Calculate the acceptance ratio $R = Pr(\theta^*)Pr(data|\theta^*) / Pr(\theta_i)Pr(data/\theta_i)$
    - If R ≥ 1, then θ^* is more likely than θ_i → automatically accept the candidate by setting $\theta_{i+1} := \theta^*$
    - Otherwise, accept the candidate θ^* with probability R → if the candidate is rejected: $\theta_{i+1} := \theta_i$

The Metropolis Algorithm

- Metropolis *et al.* 1953 http://www.aliquote.org/pub/metropolis-et-al-1953.pdf
- Initialization: Choose an arbitrary point θ_0 as first sample
- Choose an arbitrary probability density $Q(\theta_{i+1}|\theta_i)$ which suggests a candidate for the next sample θ_{i+1} given the previous sample θ_i.
- For the Metropolis algorithm, $Q()$ must be symmetric:

  it must satisfy $Q(\theta_{i+1}|\theta_i) = Q(\theta_i|\theta_{i+1})$
- For each iteration i:
  - Generate a candidate θ^* for the next sample by picking from the distribution $Q(\theta^*|\theta_i)$
  - Calculate the acceptance ratio $R = Pr(\theta^*)Pr(data|\theta^*) / Pr(\theta_i)Pr(data/\theta_i)$
    - If R ≥ 1, then θ^* is more likely than θ_i → automatically accept the candidate by setting $\theta_{i+1} := \theta^*$
    - Otherwise, accept the candidate θ^* with probability R → if the candidate is rejected: $\theta_{i+1} := \theta_i$

Conceptually this is the same Q we saw for substitution models and in the Markov Chain lecture!

Phylogenetic Metropolis Algorithm

- Initialization: Choose a random tree with random branch lengths as first sample
- For each iteration *i*:
  - Propose either
    - a new tree topology
    - a new branch length

      and re-calculate the likelihood
  - Calculate the acceptance ratio of the proposal
  - Accept the new tree/branch length or reject it
  - Print current tree with branch lengths to file only every *k* (e.g. 1000) iterations

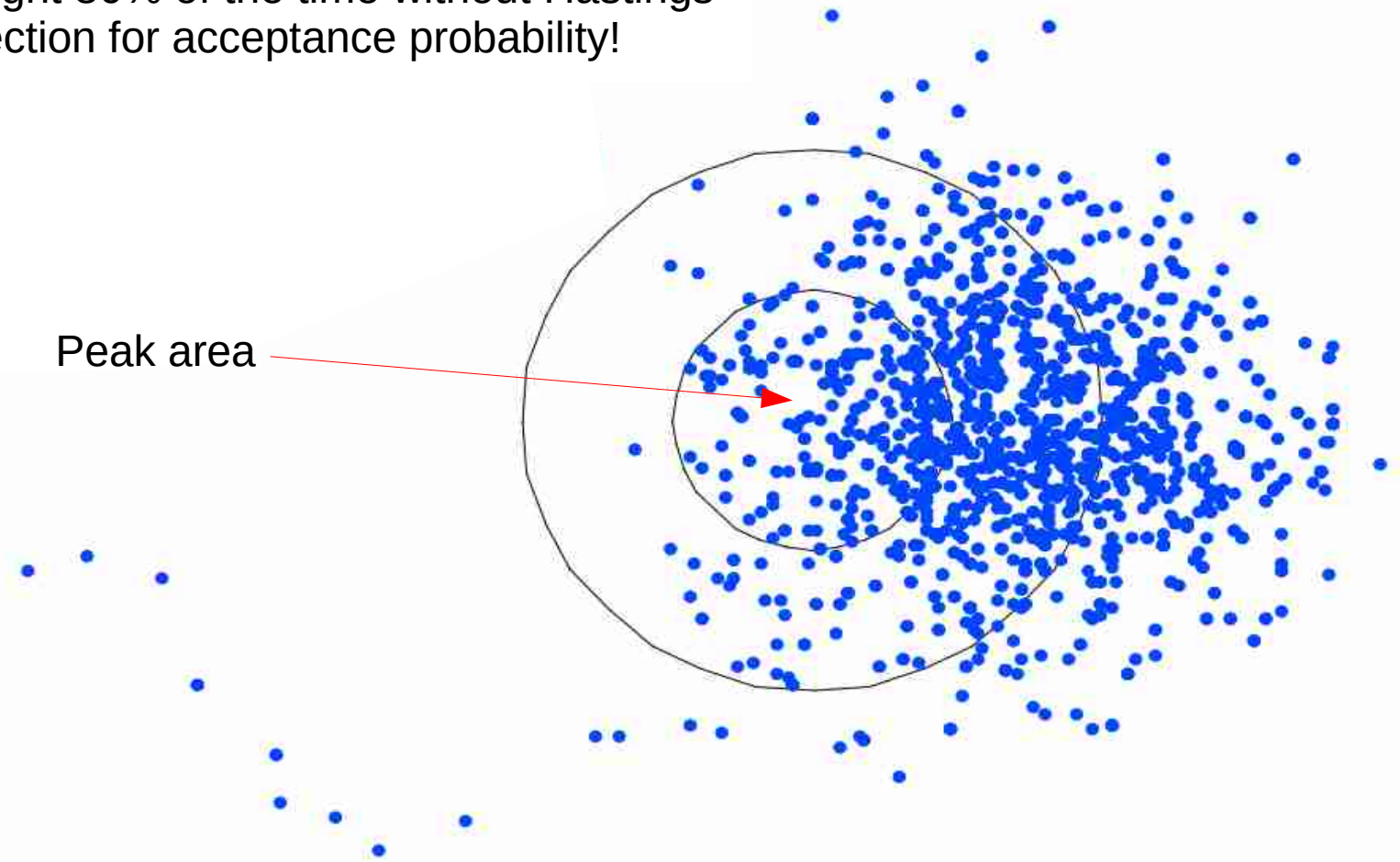
    → to generate a sample from the chain

    → to avoid writing TBs of files

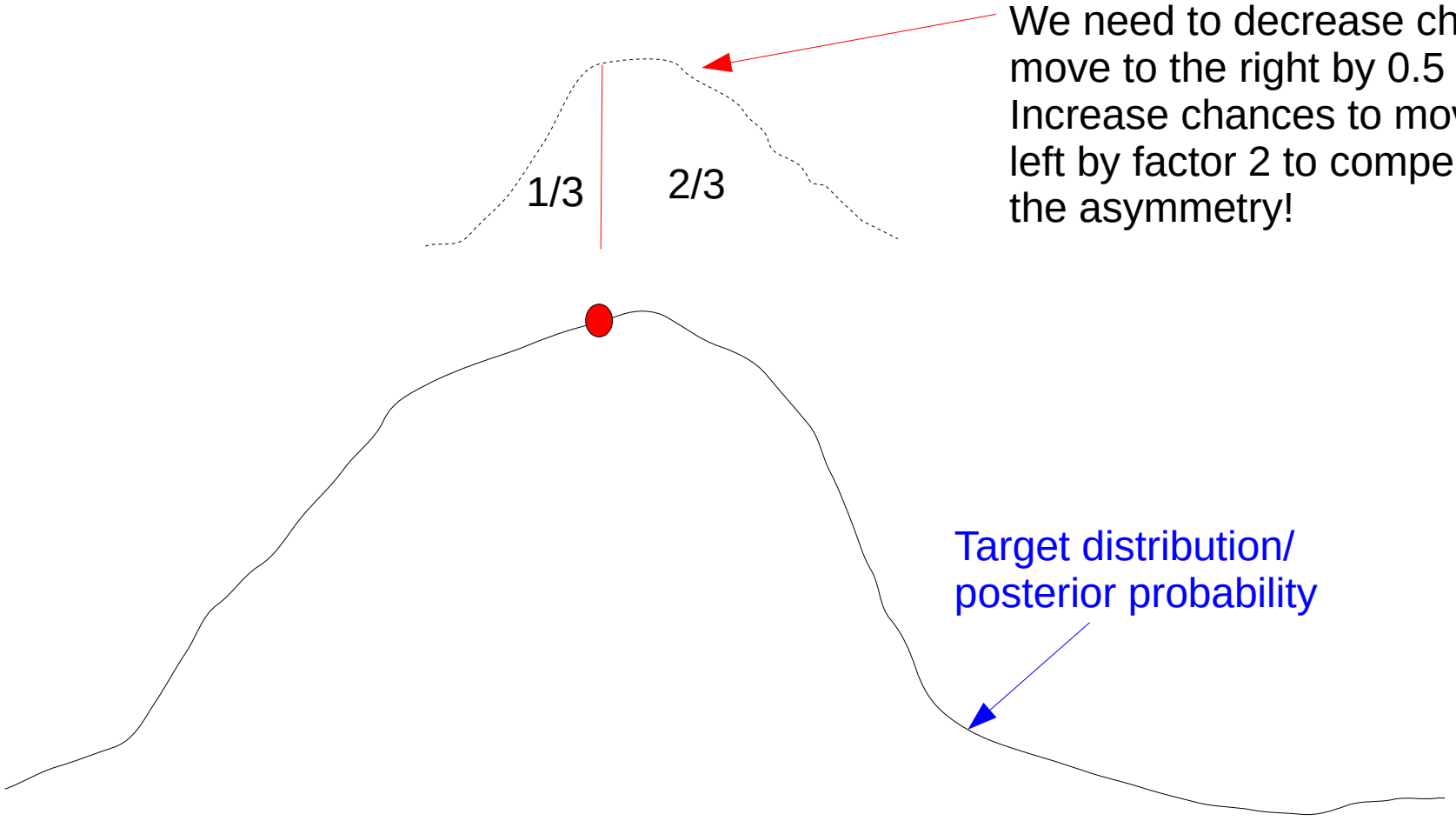
    → also known as thinning
- Summarize the sample using means, histograms, credible intervals, consensus trees, etc.

Uncorrected Proposal Distribution A Robot in 3D

Example: MCMC proposed moves to the right 80% of the time without Hastings correction for acceptance probability!



Hastings Correction



Hastings Correction

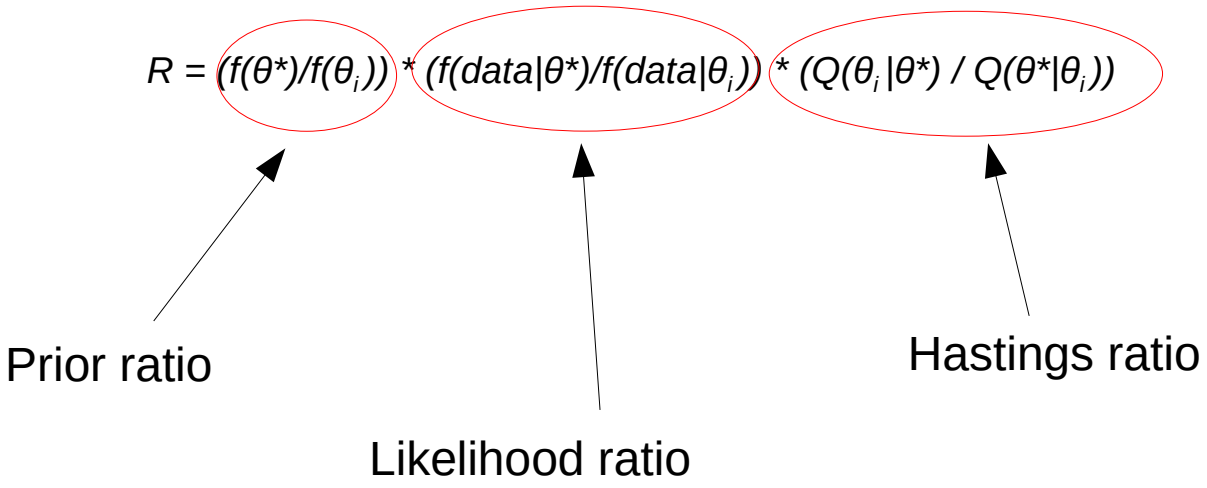
$$R = \frac{Pr(point2)}{Pr(point1)} * \frac{Pr(data|point2)}{Pr(data|point1)} * \frac{Q(point1|point2)}{Q(point2|point1)}$$

Prior ratio: for uniform priors this is 1 !

Likelihood ratio

Hastings ratio: if Q is symmetric $Q(point1|point2) = Q(point2|point)$ and the hastings ratio is 1 → we obtain the normal Metropolis algorithm

Hastings Correction more formally

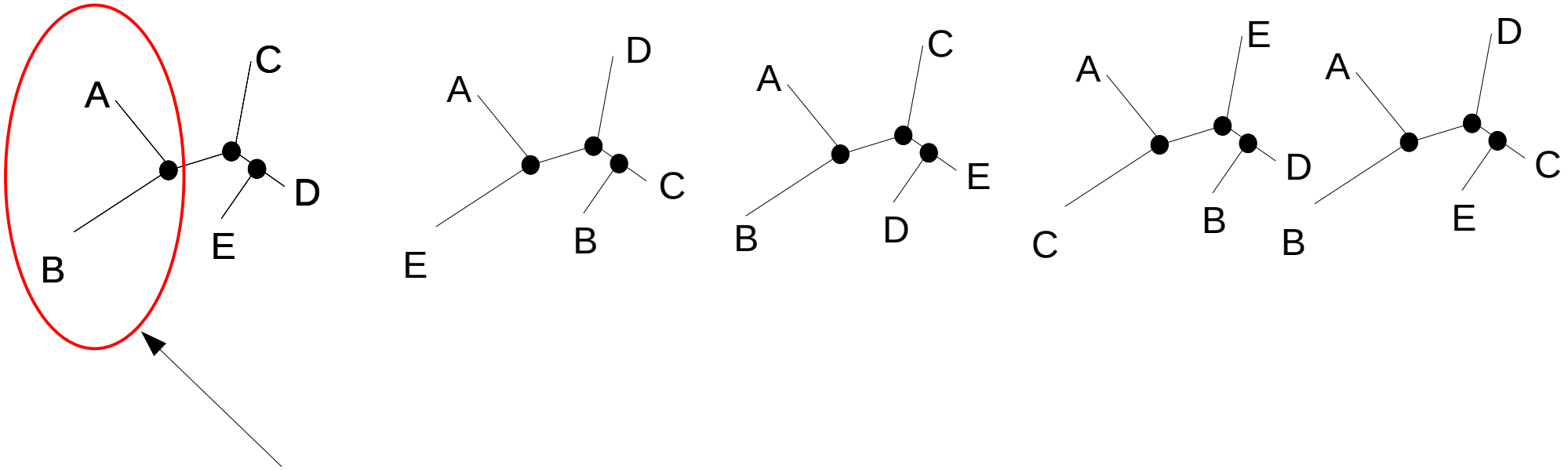


Hastings Correction is not trivial

- Problem with the equation for the hastings correction
- M. Holder, P. Lewis, D. Swofford, B. Larget. 2005. **Hastings Ratio of the LOCAL Proposal Used in Bayesian Phylogenetics.** *Systematic Biology*. 54:961-965. http://sysbio.oxfordjournals.org/content/54/6/961.full

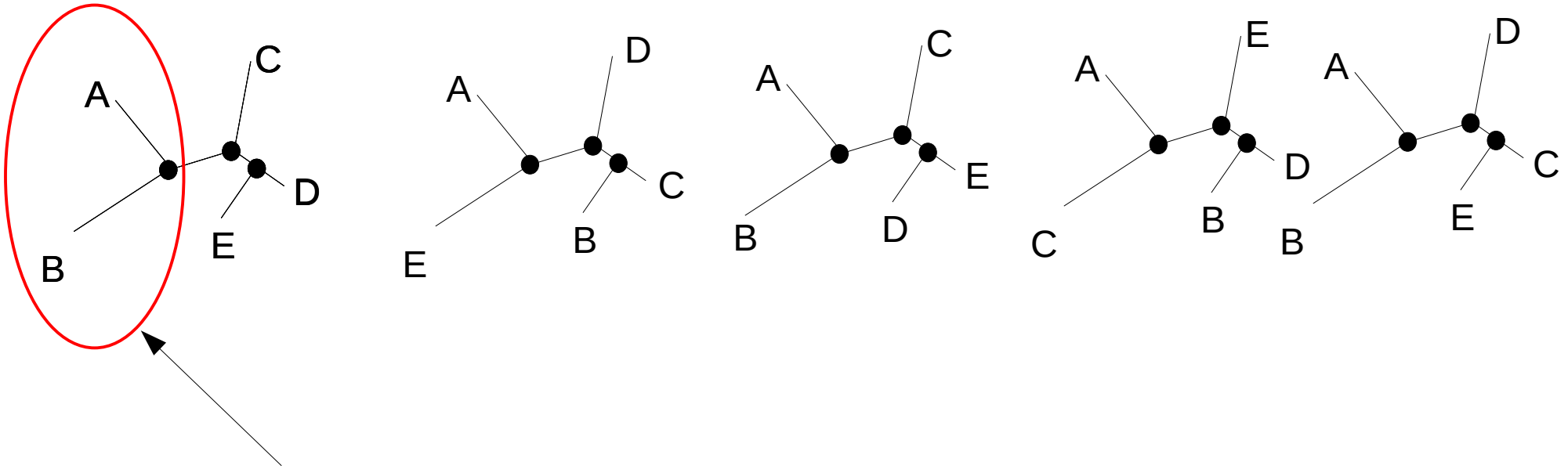
  "As part of another study, we estimated the marginal likelihoods of trees using different proposal algorithms and discovered repeatable discrepancies that implied that the published Hastings ratio for a proposal mechanism used in many Bayesian phylogenetic analyses is incorrect."
- Incorrect Hastings ratio used from 1999-2005

Back to Phylogenetics



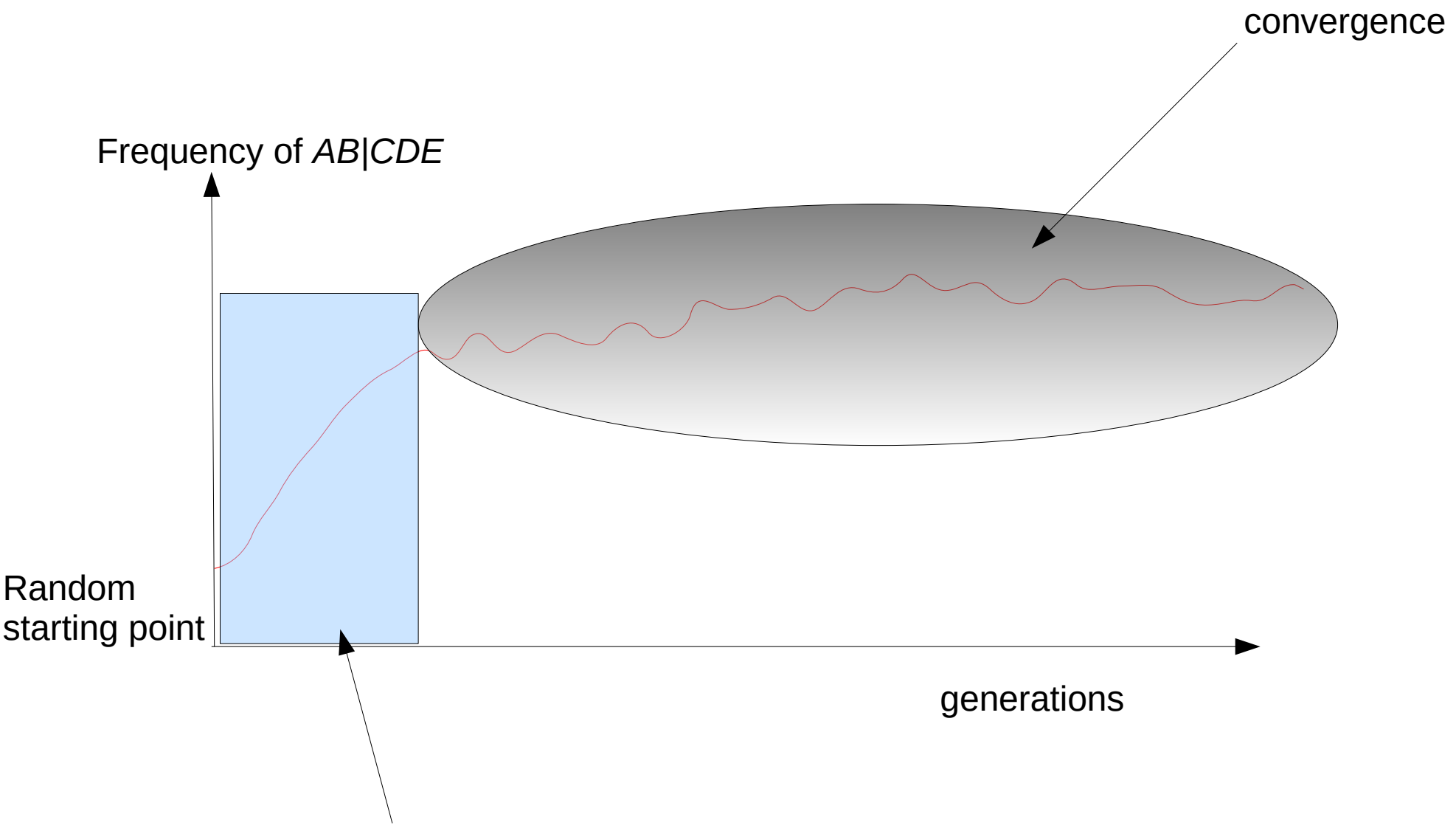
What's the posterior probability of bipartition *AB*|*CDE*?

Back to Phylogenetics



What's the posterior probability of bipartition *AB*|*CDE*?
We just count from the sample generated by MCMC, here it's *3/5 → 0.6*
This approximates the true proportion (posterior probability) of bipartition *AB*|*CDE*
if we have run the chain long enough and **if** it has converged

MCMC in practice



Convergence

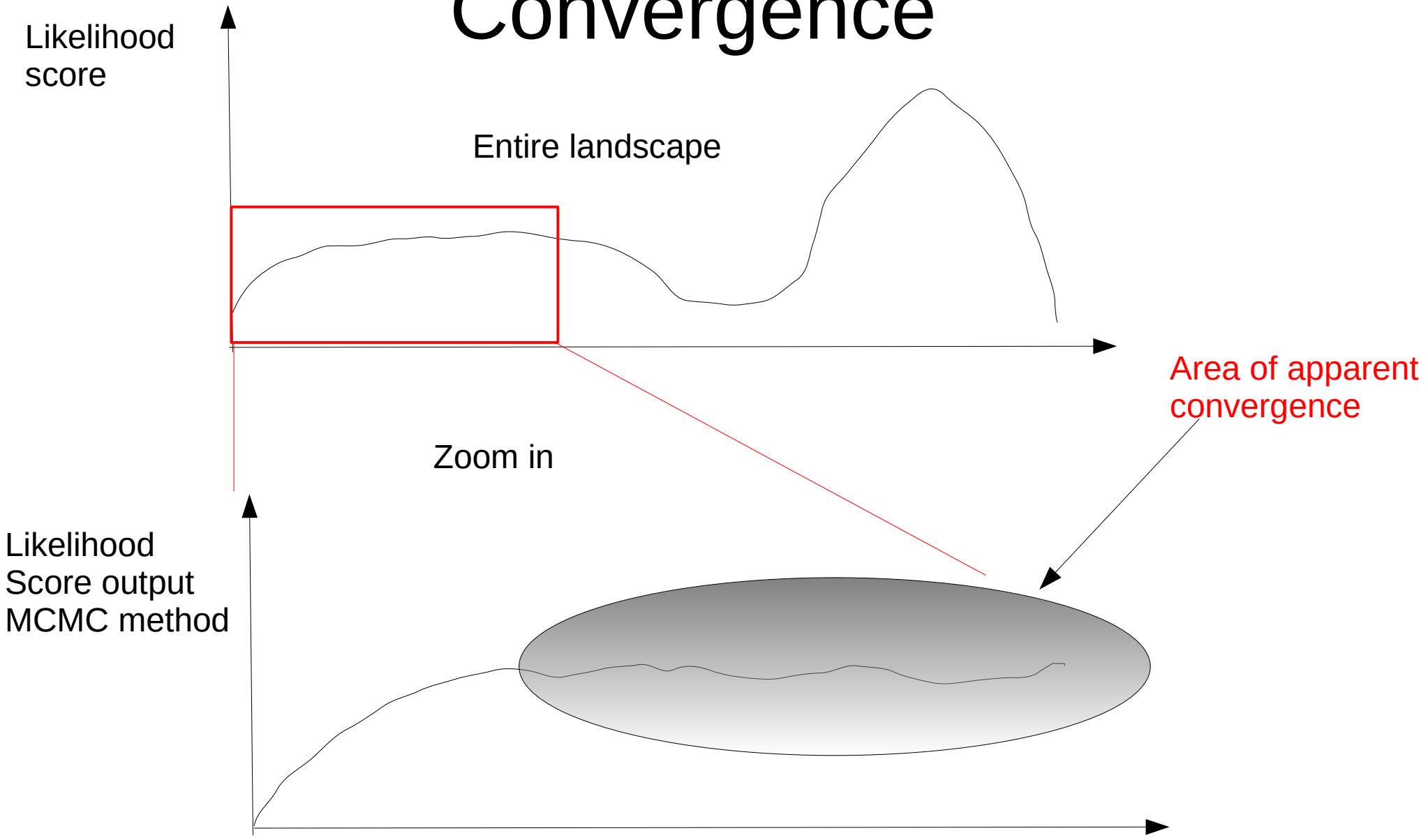
- How many samples do we need to draw to obtain an accurate approximation?
- When can we stop drawing samples?
- Methods for convergence diagnosis

  → we can never say that a MCMC-chain has converged

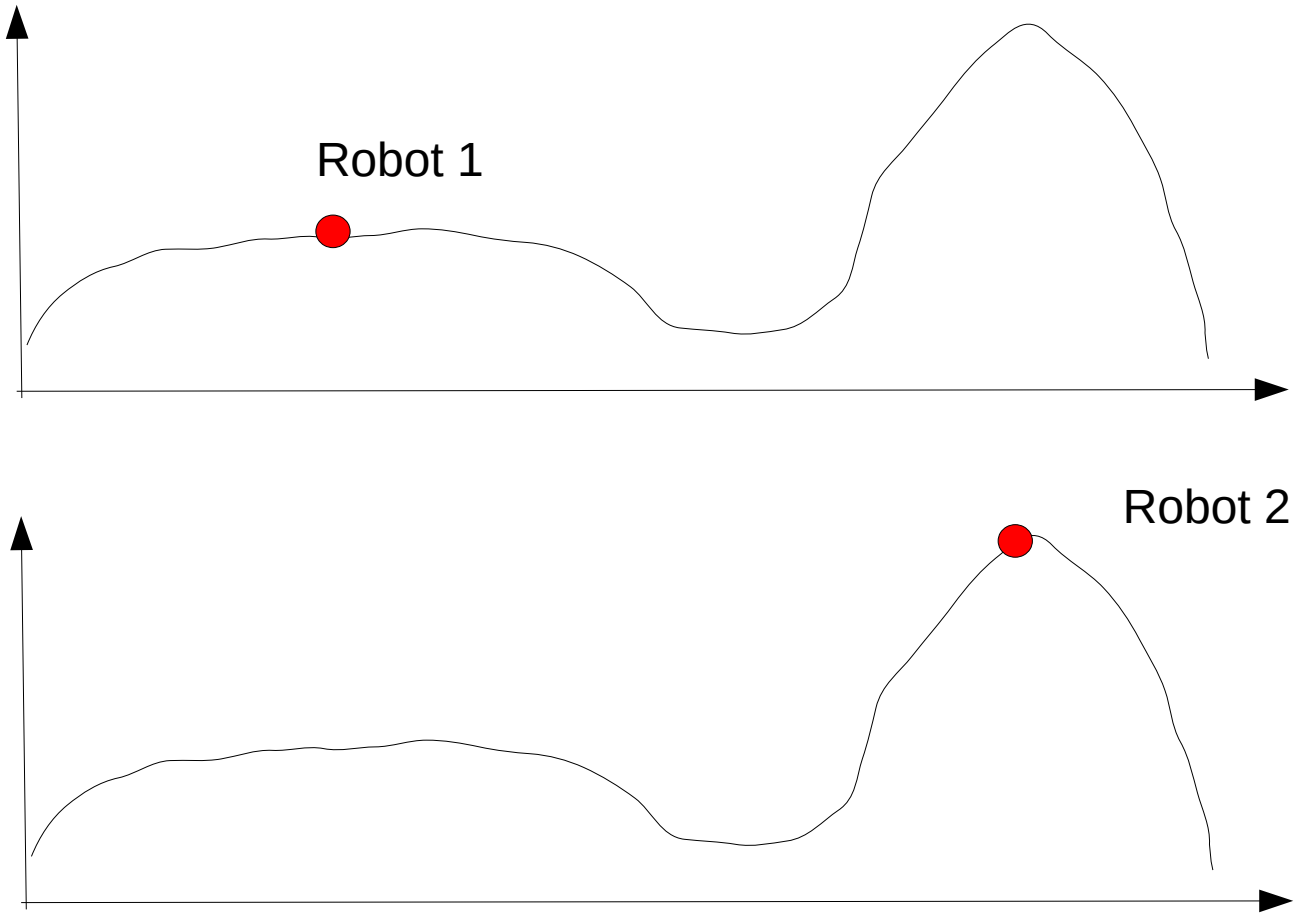
  → we can only diagnose that it has not converged

  → a plethora of tools for convergence diagnostics for phylogenetic MCMC

Convergence



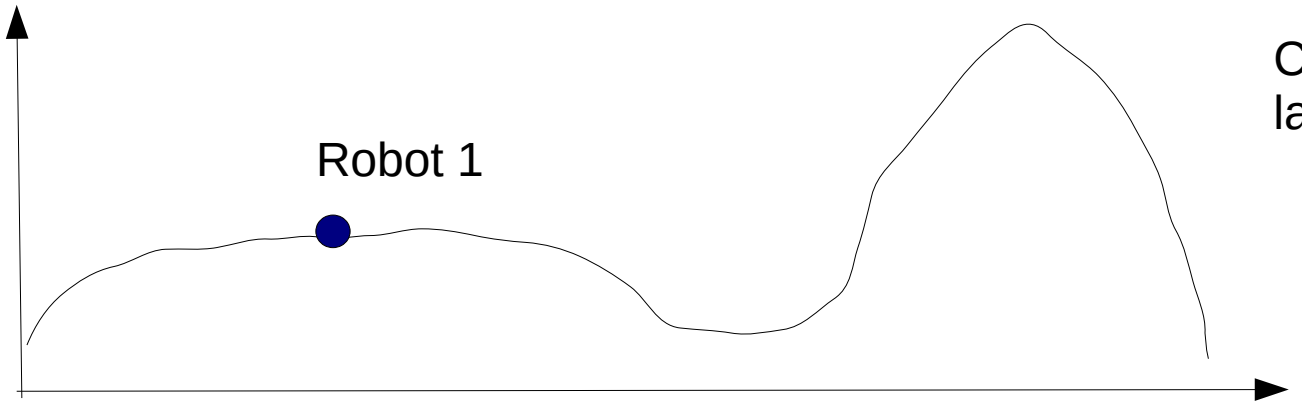
Solution: Run Multiple Chains



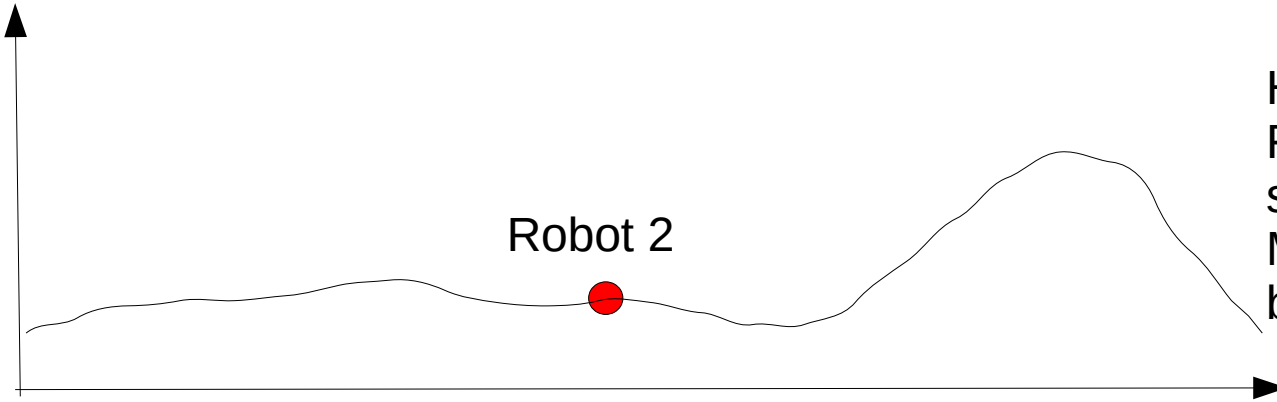
Outline

- Bayesian statistics
- Monte-Carlo simulation & integration
- Markov-Chain Monte-Carlo methods
- Metropolis-coupled MCMC-methods
- Some phylogenetic proposals
- Reversible jump MCMC

Heated versus Cold Chains



Cold chain: sees landscape as is

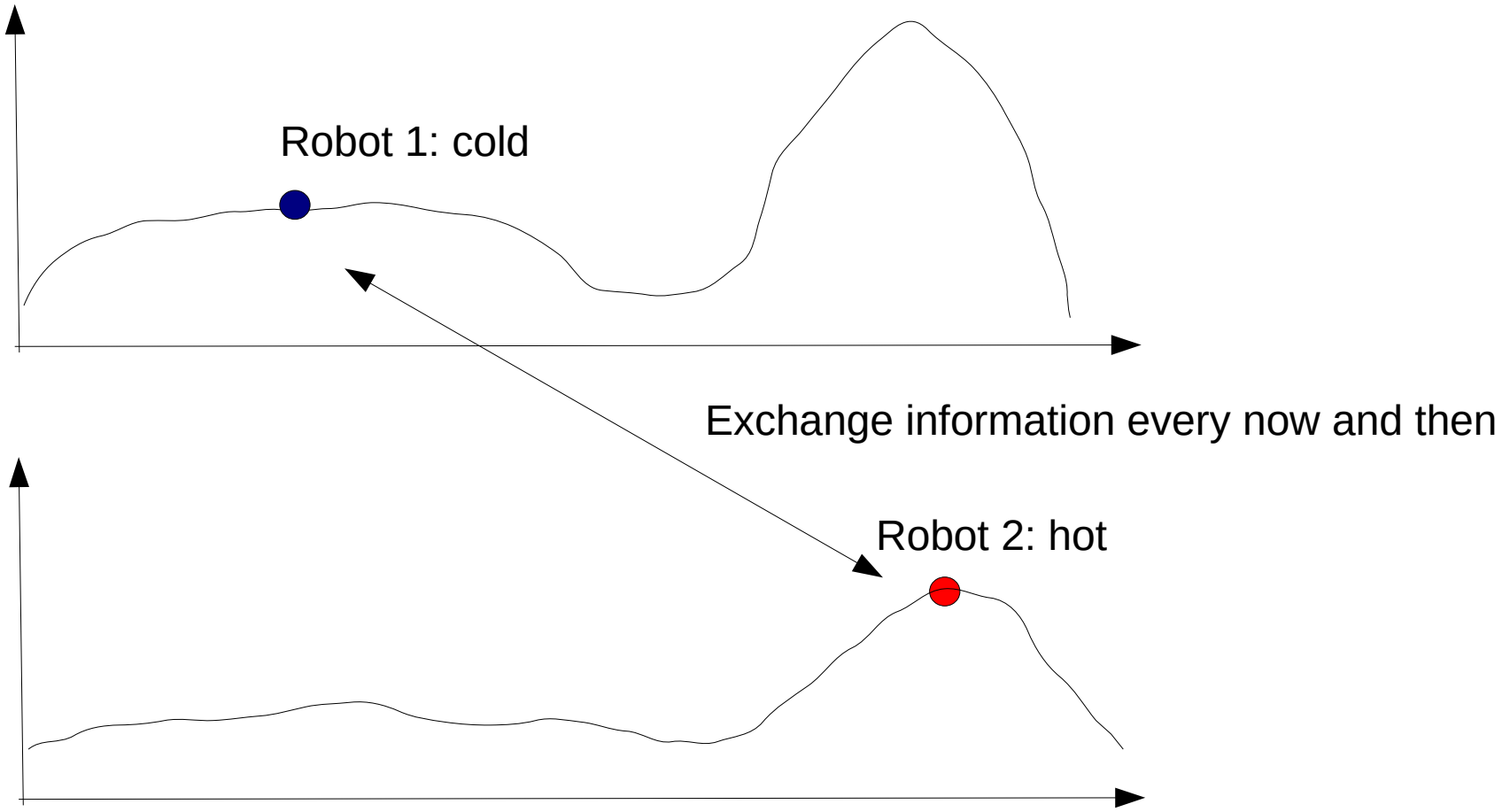


Hot chain: sees a Flatter version of the same landscape → Moves more easily between peaks

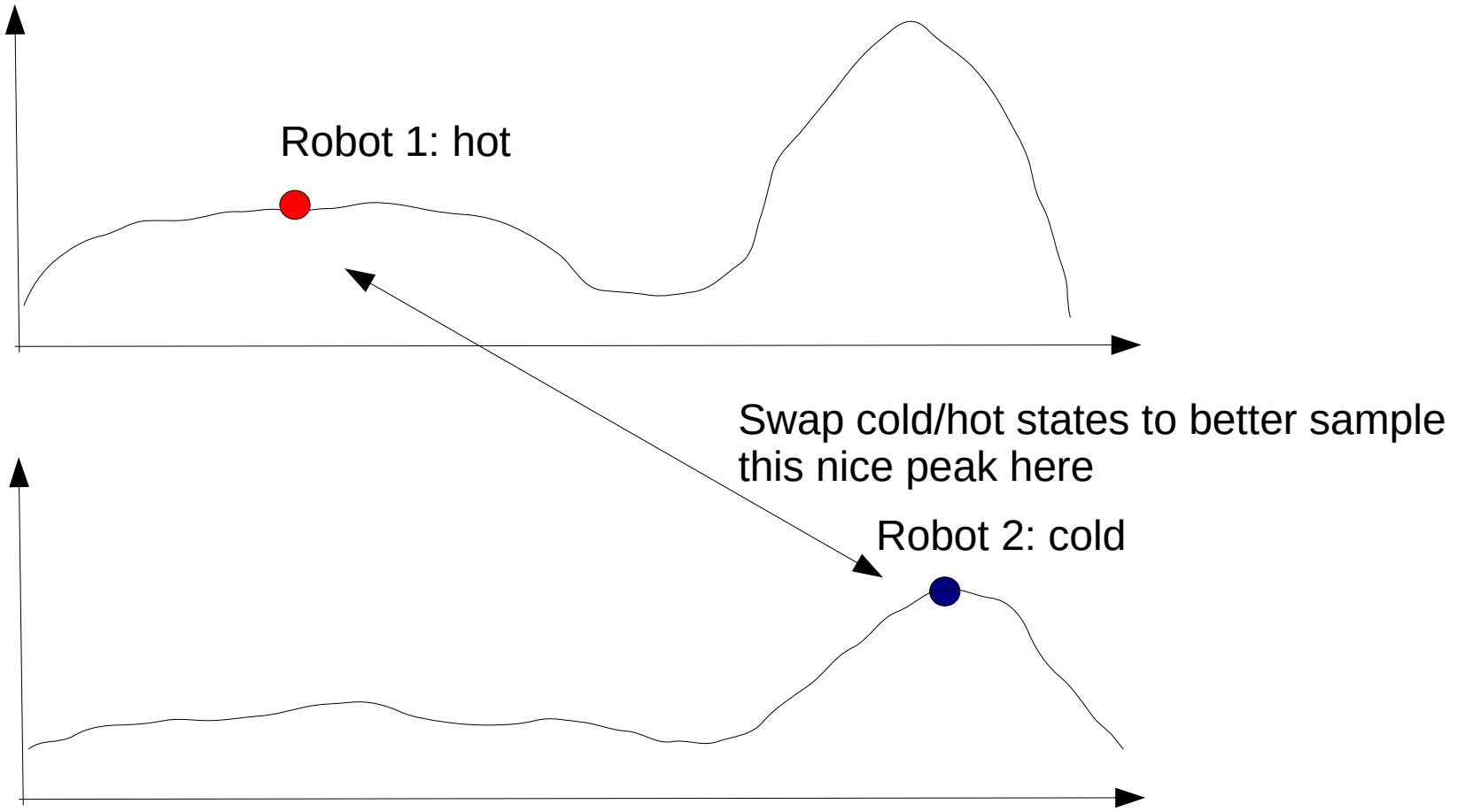
Known as MCMCMC

- Metropolis-Coupled Markov-Chain Monte Carlo
- Run several chains simultaneously
  - 1 cold chain (the one that samples)
  - Several heated chains
- Heated chain robots explore the parameter space in larger steps
- To flatten the landscape the acceptance ratio *R* is modified as follows: $R^{1/1+H}$ where *H* is the so-called temperature
  - For the cold chain *H := 0.0*
  - Setting the temperature for the hot chains is a bit of woo-do

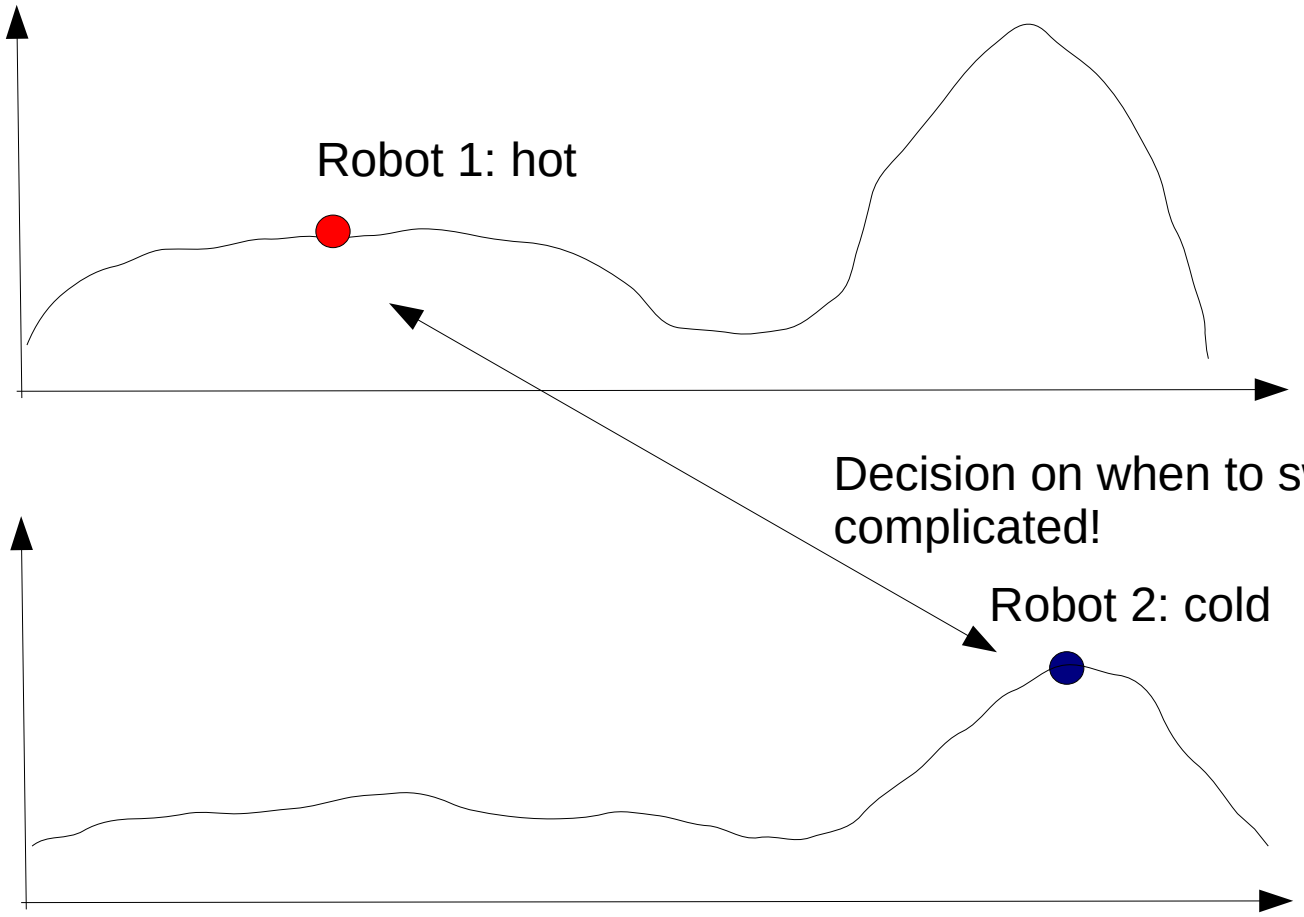
Heated versus Cold Chains



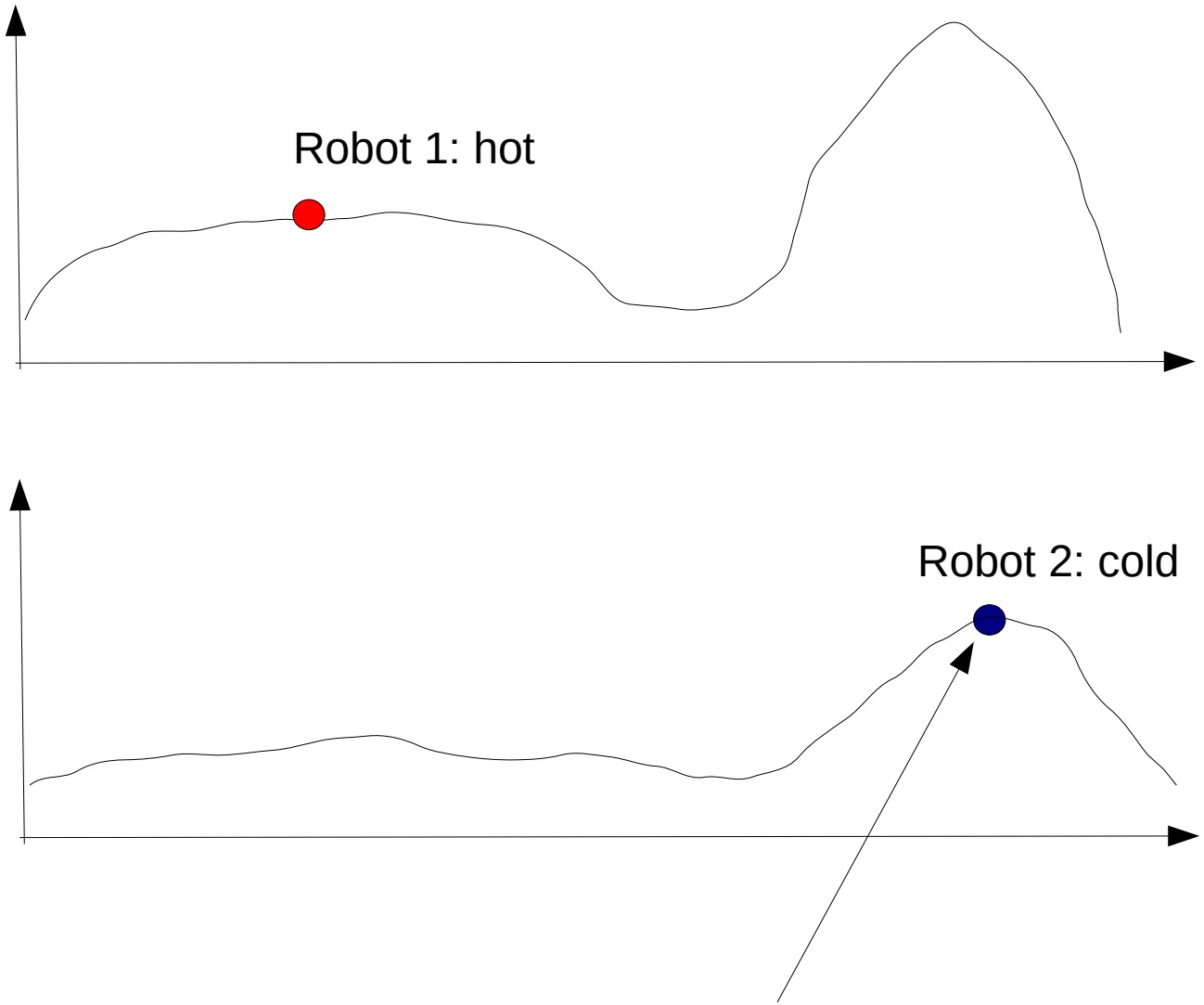
Heated versus Cold Chains



Heated versus Cold Chains



Heated versus Cold Chains



A few words about priors

- Prior probabilities convey the scientist's beliefs, before having seen the data
- Using uninformative prior probability distributions (e.g., uniform priors, also called flat priors)

  → differences between prior and posterior distribution are attributable to likelihood differences
- Priors can bias an analysis
- For instance, we could chose an arbitrary prior distribution for branch lengths in the range [1.0,20.0]

  → what happens if branch lengths are much shorter?

Outline

- Bayesian statistics
- Monte-Carlo simulation & integration
- Markov-Chain Monte-Carlo methods
- Metropolis-coupled MCMC-methods
- Some phylogenetic proposals
- Reversible jump MCMC

Some Phylogenetic Proposal Mechanisms

- Branch Lengths
  - Sliding Window Proposal
  - Multiplier Proposal
- Topologies
  - Local Proposal (the one with the bug in the Hastings ratio)
  - Extending TBR (Tree Bisection Reconnection) Proposal
- Remember: We need to design proposals for which
  - We either *don't need to* calculate the Hastings ratio
  - Or for which we *can* calculate it
  - That have a 'good' acceptance rate

    → all sorts of tricks being used, e.g., parsimony-biased topological proposals

Some Phylogenetic Proposal Mechanisms

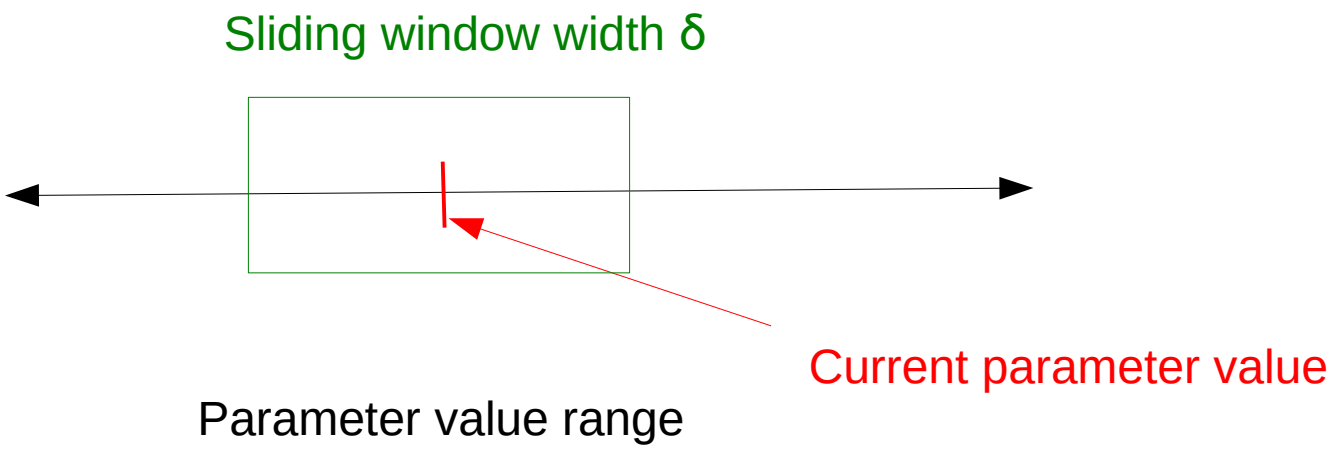
- Univariate parameters & branch lengths
  - Sliding Window Proposal
- Branch lengths
  - Node slider proposal
- Topologies
  - Local Proposal (the one with the bug in the Hastings ratio!)
- Remember: We need to design proposals for which
  - We either *don't need to* calculate the Hastings ratio
  - Or for which we *can* calculate it
  - That have an appropriate acceptance rate

    → all sorts of tricks being used, e.g., parsimony-biased topological proposals

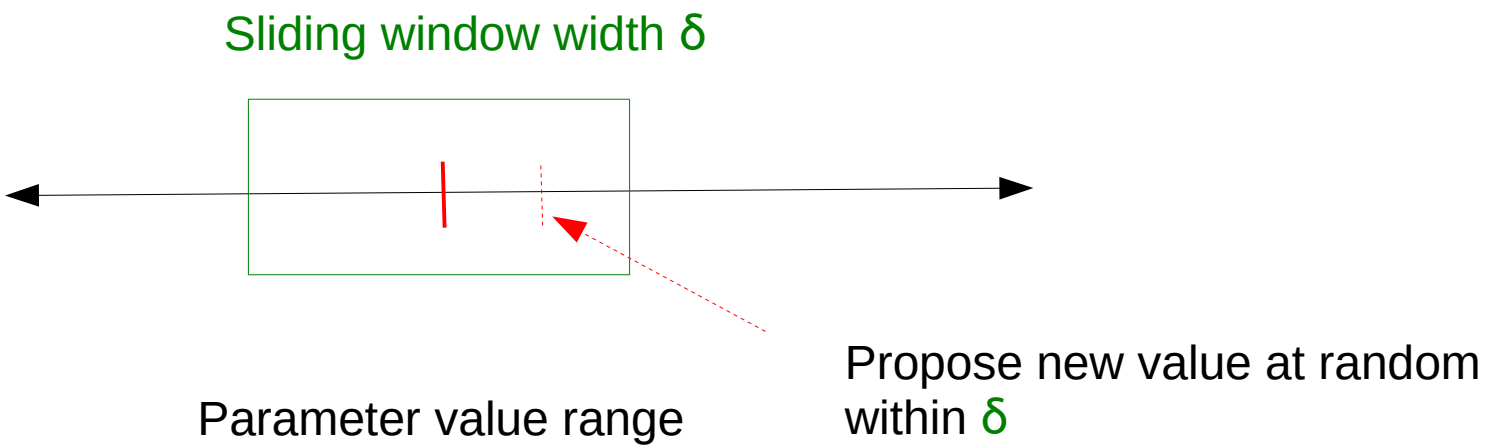
    → acceptance rate should be around 25% (empirical observation)

    → for sampling from a multivariate normal distribution it has been formally shown that an acceptance rate of 23.4% is optimal

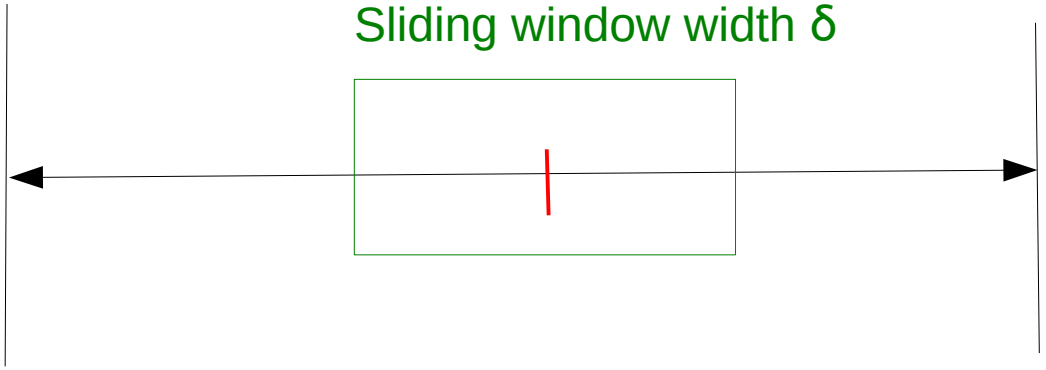
Sliding Window Proposal



Sliding Window Proposal

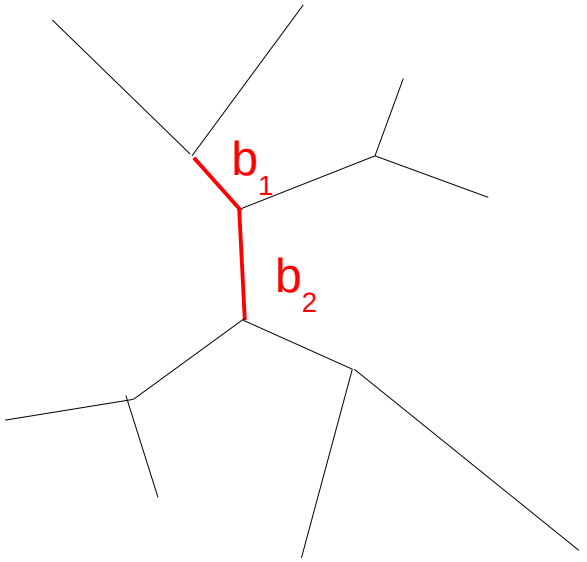


Sliding Window Proposal



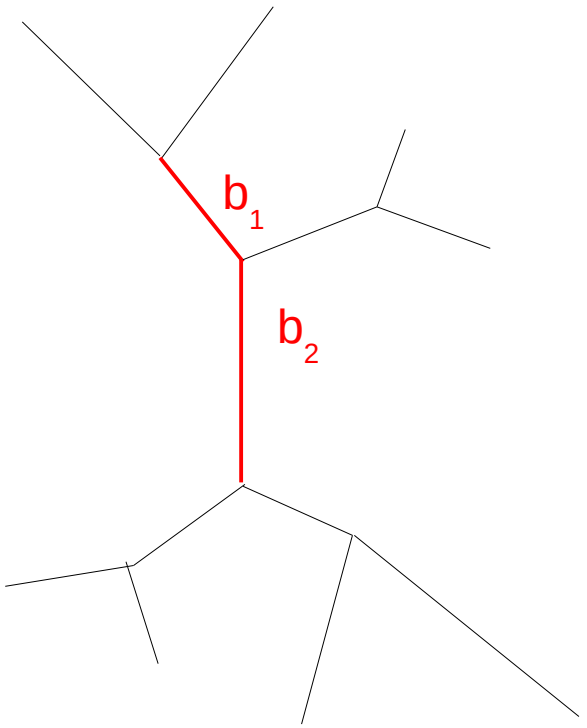
Notes:
1. The hastings ratio of this move is *1*
2. The edge cases can be handled by back-projection
3. The window size δ can be tuned itself (auto-tuning) to obtain an acceptance rate of ≈ ¼
4. This proposal can be used, e.g., for the *α*-shape parameter of the *Γ* function in rate heterogeneity models

The Node Slider Proposal



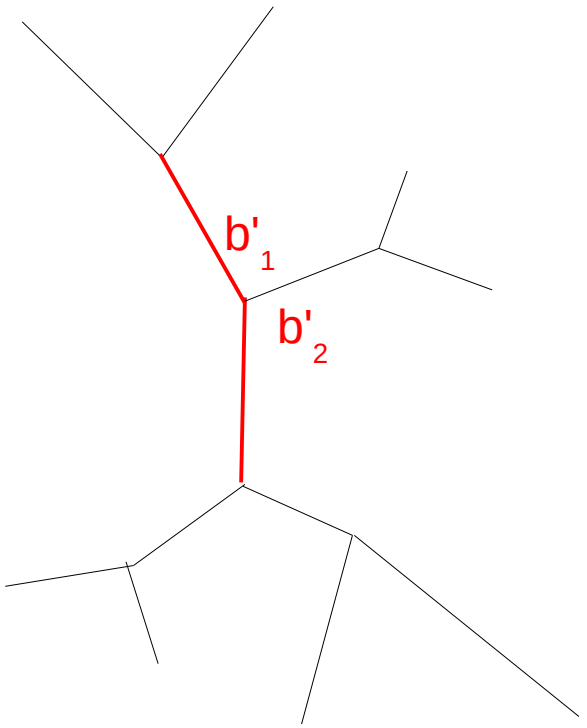
1. Pick *2* contiguous branches at random

The Node Slider Proposal



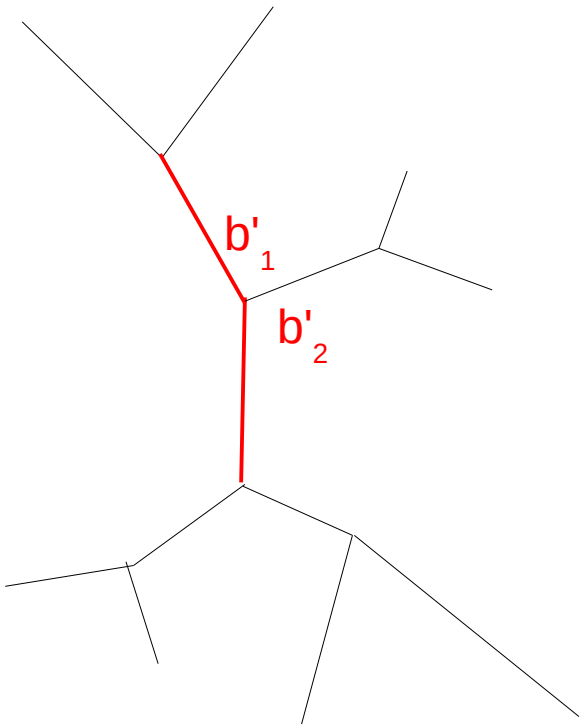
1. Pick *2* contiguous branches at random
2. Multiply the *2* branches by the same random number

The Node Slider Proposal



1. Pick *2* contiguous branches at random
2. Multiply the *2* branches by the same random number
3. Propose a new branch ratio b_1/b_2 at random

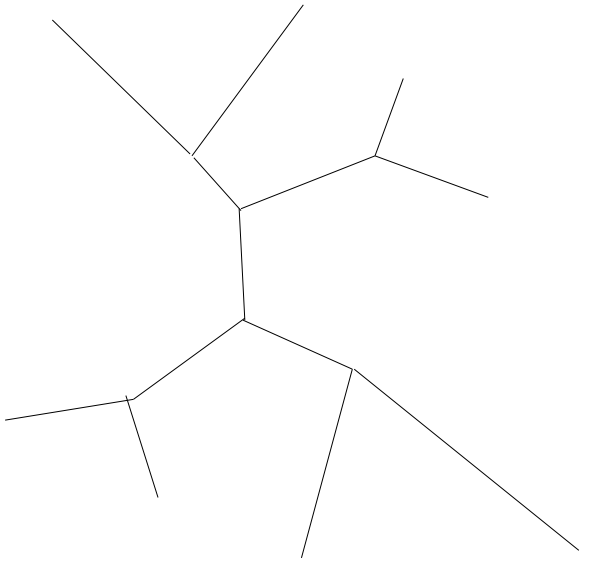
The Node Slider Proposal



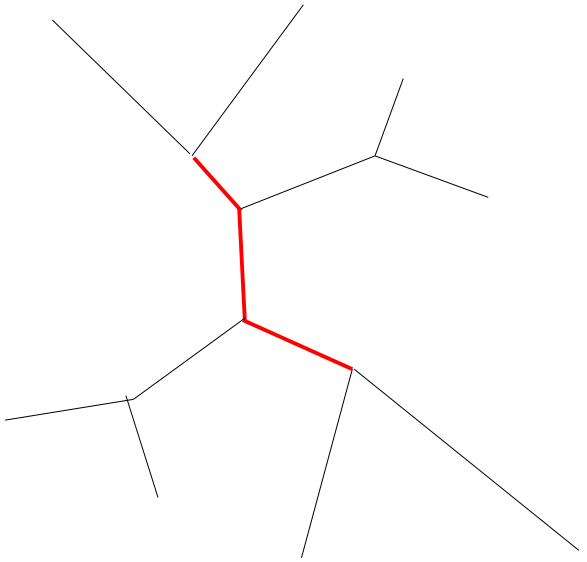
1. Pick *2* contiguous branches at random
2. Multiply the *2* branches by the same random number
3. Propose a new branch ratio b_1/b_2 at random

The Hastings ratio of this move is not *1*!

Moving through Tree Space

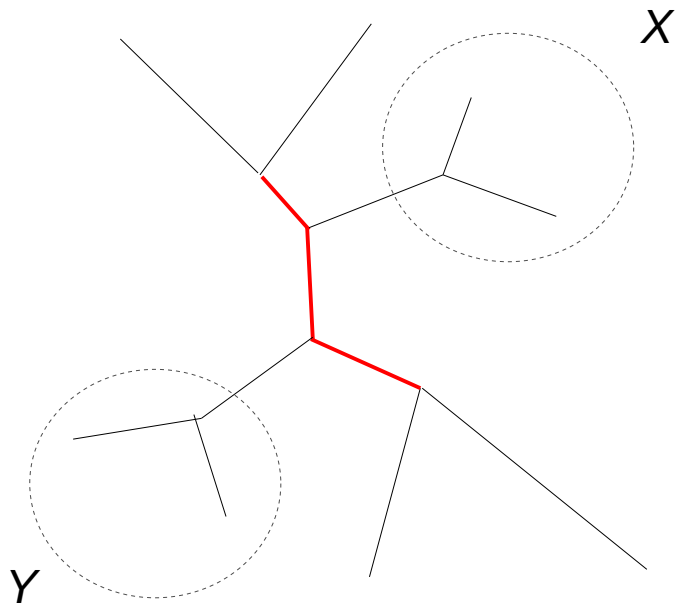


Moving through Tree Space



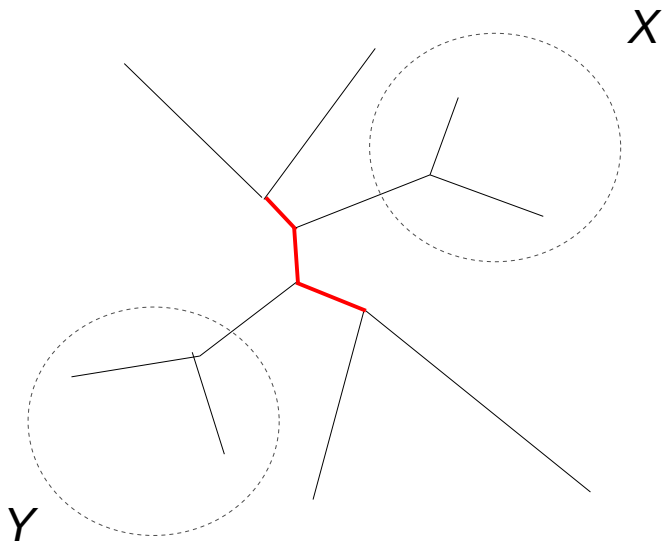
1. Pick *3* contiguous branches at random

Moving through Tree Space



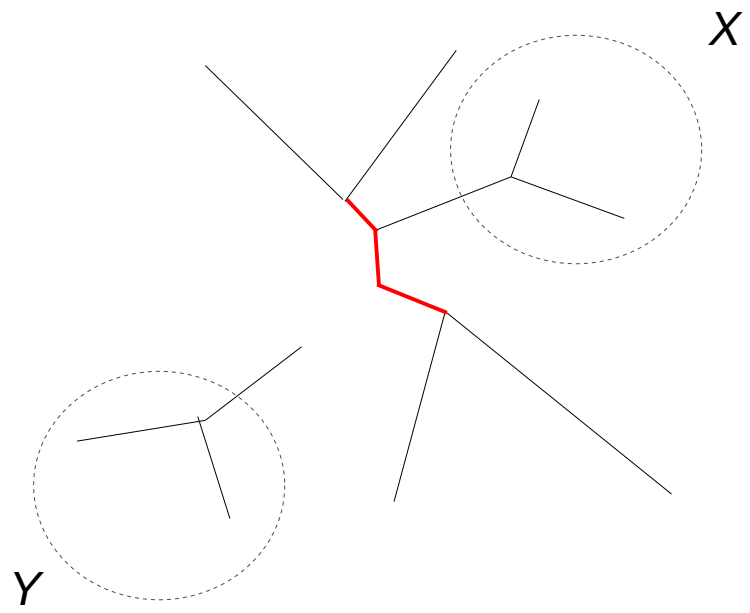
1. Pick *3* contiguous branches at random that define *2* Subtrees *X* and *Y*

Moving through Tree Space



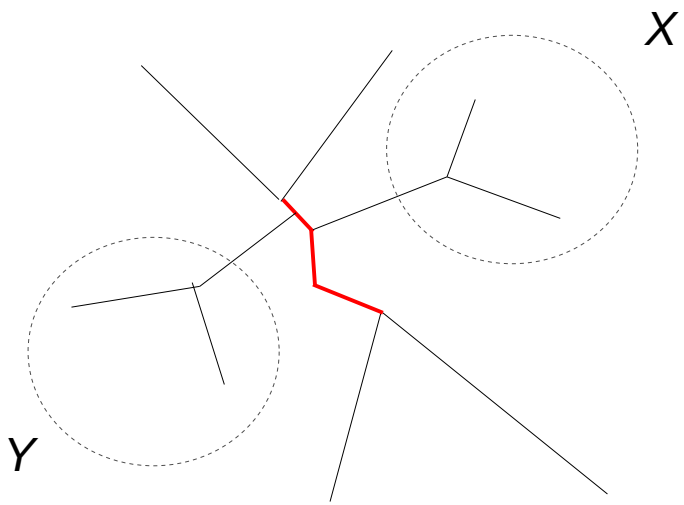
1. Pick *3* contiguous branches at random that define *2* Subtrees *X* and *Y*
2. shrink or grow selected *3* branch segment by a random amount

Moving through Tree Space



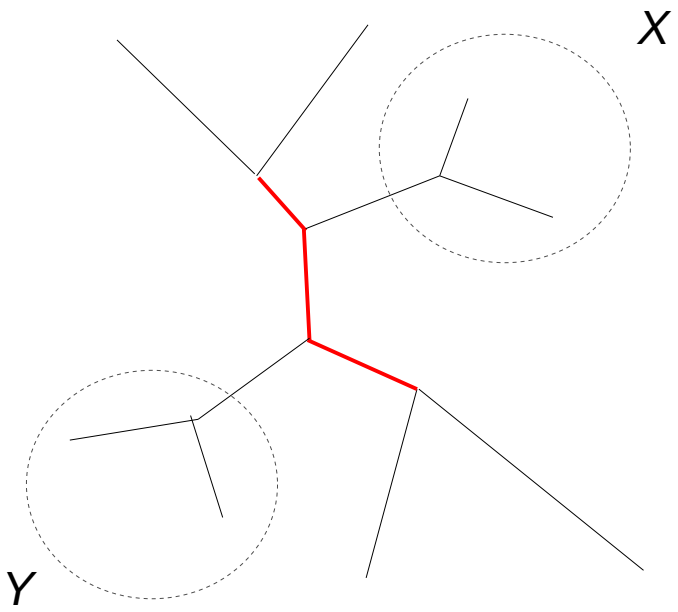
1. Pick *3* contiguous branches at random that define *2* Subtrees *X* and *Y*
2. shrink or grow selected *3* branch segment by a random Amount
3. Chose either *X* or *Y* at random and prune it from the tree

Moving through Tree Space

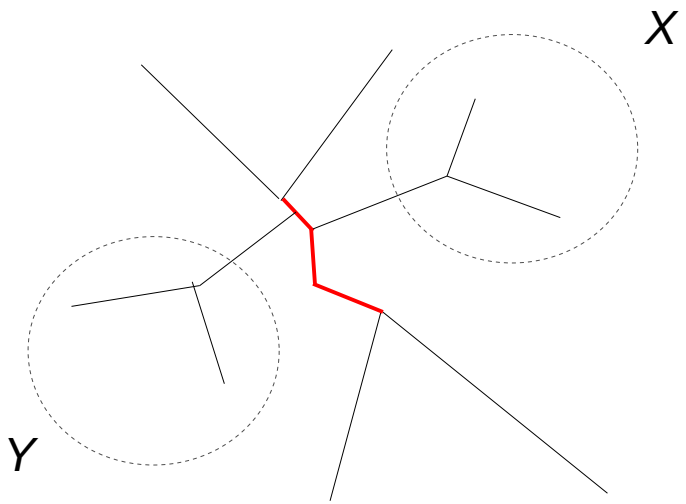


1. Pick *3* contiguous branches at random that define *2* Subtrees *X* and *Y*
2. shrink or grow selected *3* branch segment by a random Amount
3. Chose either *X* or *Y* at random And prune it from the tree
4. Re-insert *Y* at random into The *3* branch segment

Moving through Tree Space



Initial tree t_i

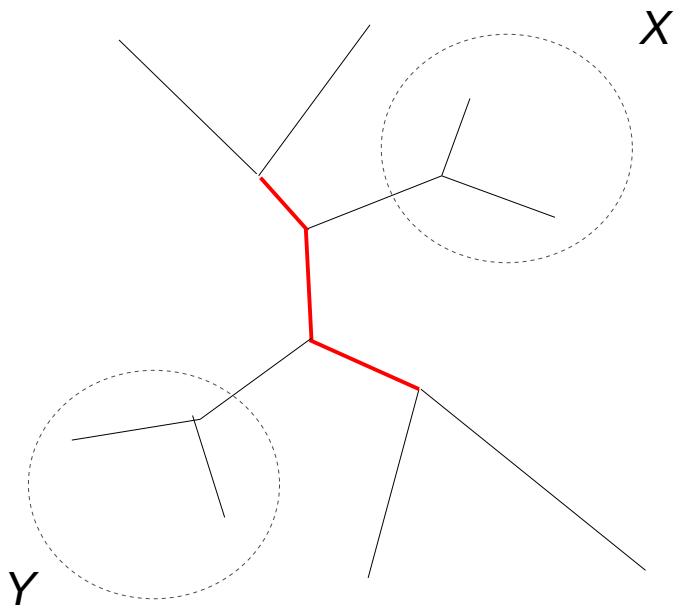


Proposed tree t_{i+1}

Proposed tree: *3* branch lengths changed and one NNI (Nearest Neighbor Interchange) move applied

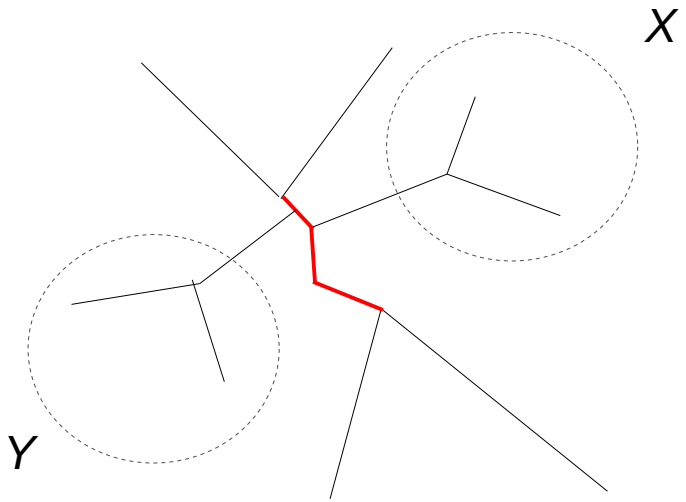
Moving through Tree Space

LnL = -3000



Initial tree t_i

LnL = -2900

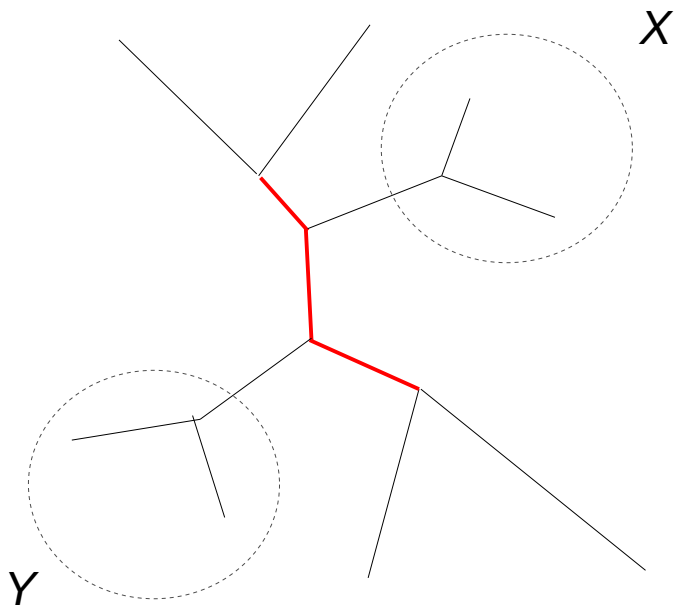


Proposed tree t_{i+1}

The proposed tree has a better likelihood!
Will the proposed tree always be accepted?

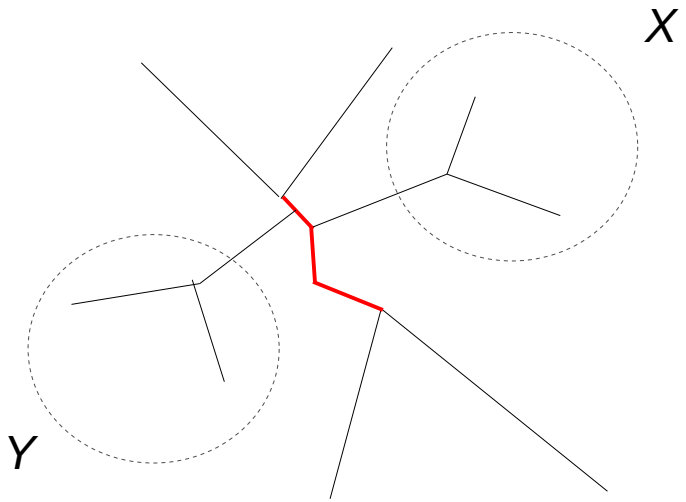
Moving through Tree Space

LnL = -3000



Initial tree t_i

LnL = -2900



Proposed tree t_{i+1}

The proposed tree has a better likelihood!
Will the proposed tree always be accepted?
→ think about Priors and Hastings ratio!

Outline

- Bayesian statistics
- Monte-Carlo simulation & integration
- Markov-Chain Monte-Carlo methods
- Metropolis-coupled MCMC-methods
- Some phylogenetic proposals
- Reversible jump MCMC

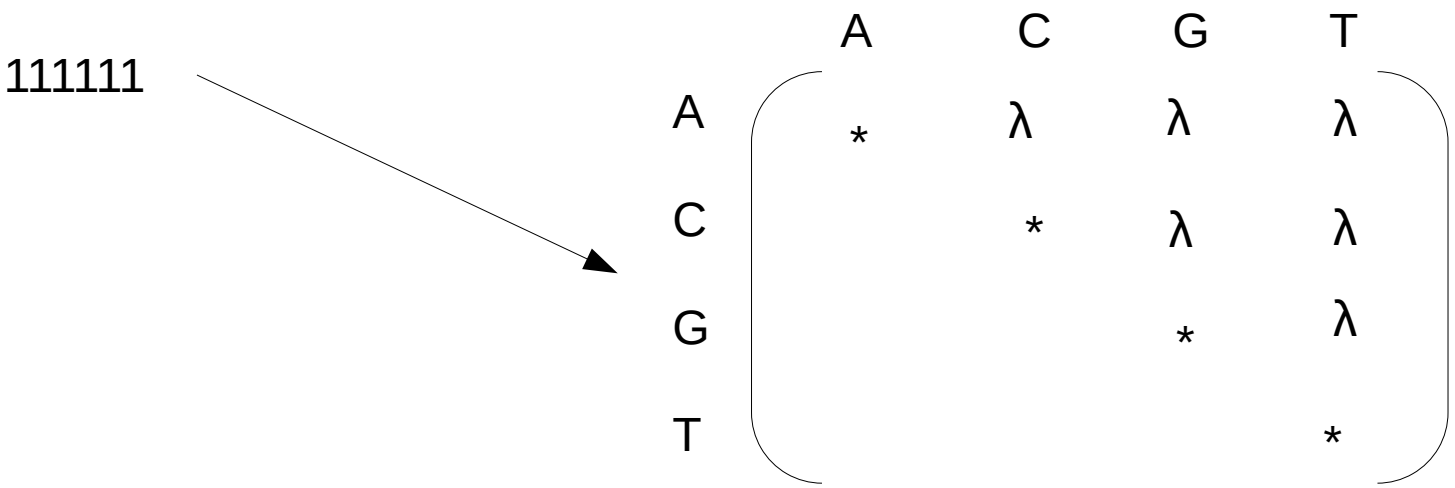
How do we select models using MCMC?

- **Example:** Consider all possible time-reversible nucleotide substitution models ranging from Jukes Cantor (JC, 1 rate) to the General Time Reversible Model (GTR, 6 rates)
- We will denote rate configurations by strings, e.g.,
  - 111111 is the JC model
  - ...
  - 123456 is the GTR model
- Let me explain this further ...

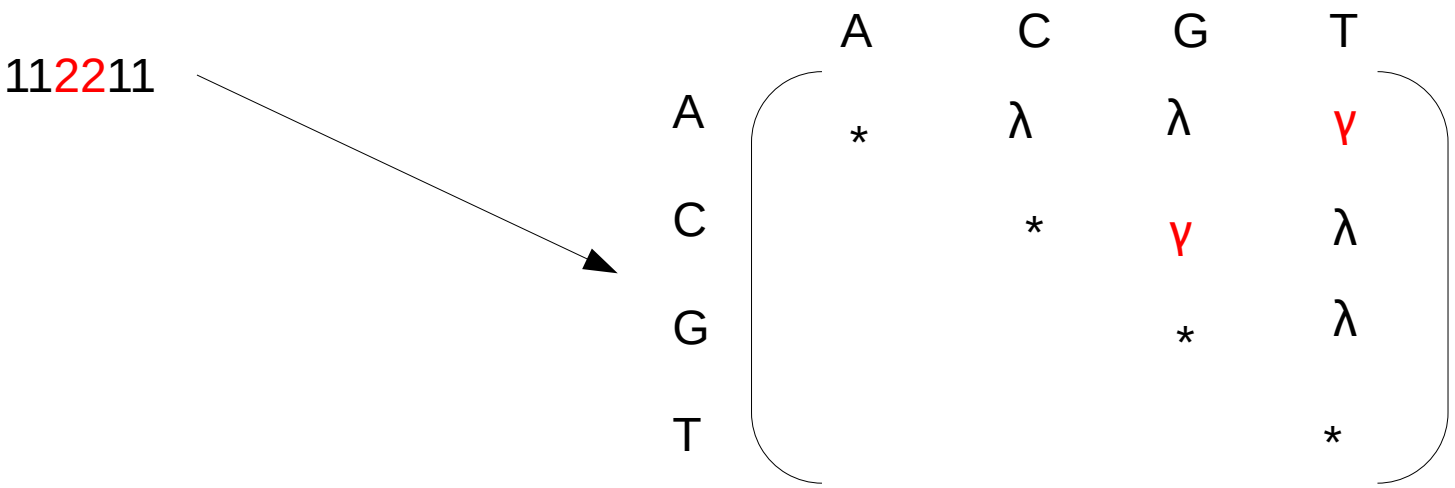
Model Strings

111111

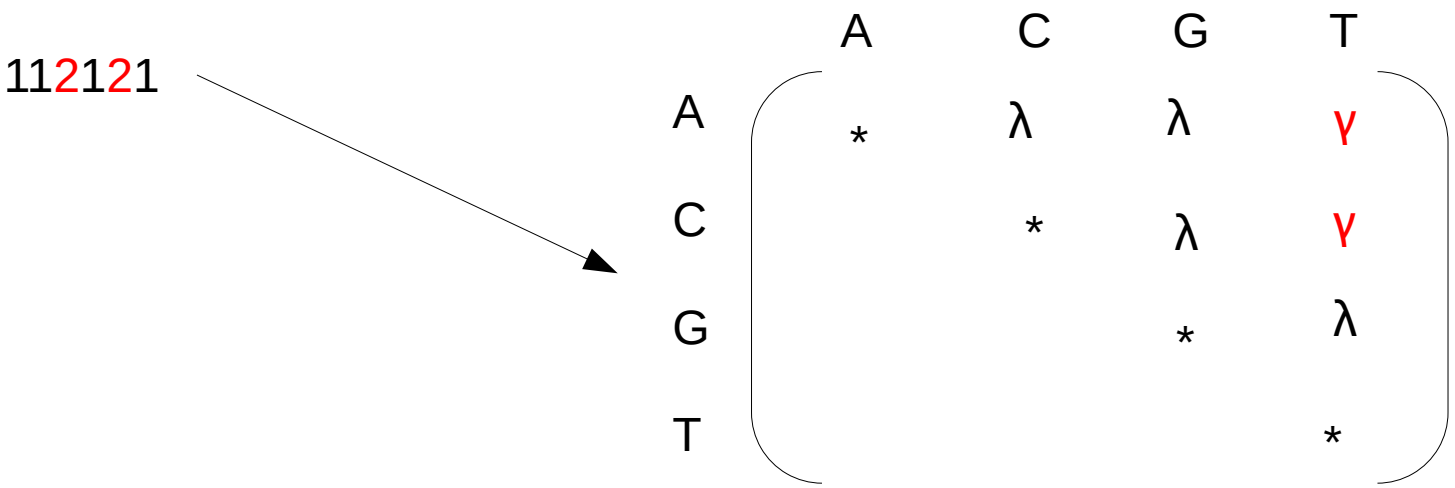
Model Strings



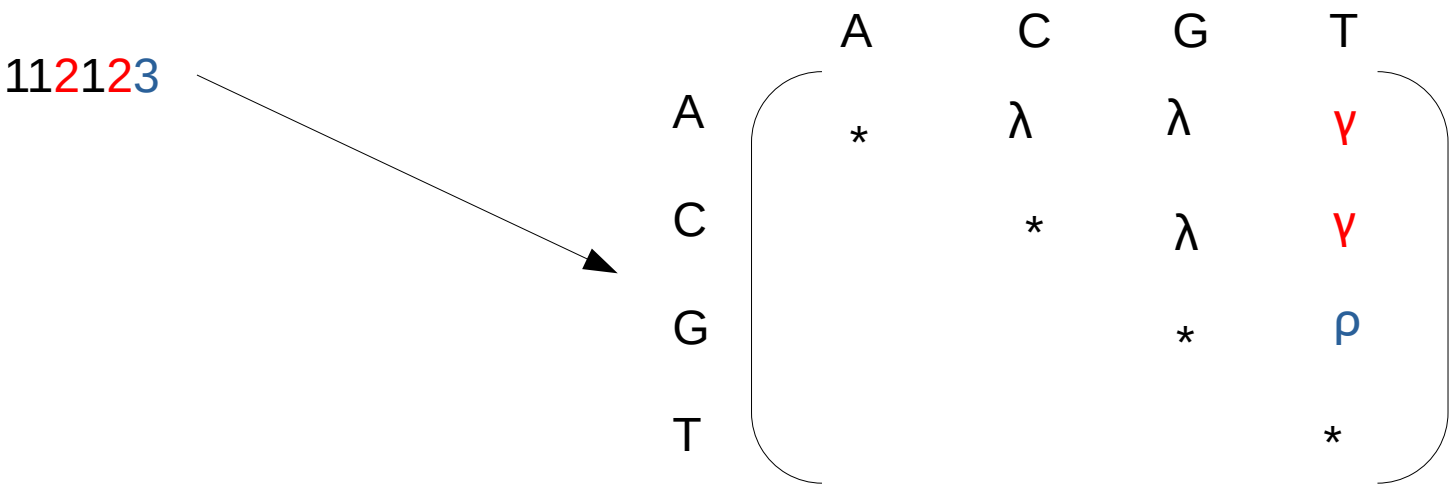
Model Strings



Model Strings



Model Strings



How many time-reversible DNA models are there?

- Number of ways a set with *n* objects can be partitioned into disjoint non-empty sets
- **Example:** the set *{a,b,c}* can be partitioned as follows:

  { {a}, {b}, {c} }

  { {a}, {b, c} }

  { {b}, {a, c} }

  { {c}, {a, b} }

  { {a, b, c} }

- The number of combinations for *n* (*3* in our example) is given by the so-called *Bell* number, for details see https://en.wikipedia.org/wiki/Bell_number

The Bell Numbers

- *n:= 1 → 1*
- *n:= 2 → 2*
- *n:=3 → 5*
- *n:= 4 → 15*
- *n:= 5 → 52*
- *n:= 6 → 203*
- *n:= 7 → 877*
- etc...

What do we need?

- Apart from our usual suspect parameters (tree topology, branch lengths, stationary frequencies, substitution rates, a), we also want to integrate over different models now …
- What are the problems we need to solve?

What do we need?

- Apart from our usual suspect parameters (tree topology, branch lengths, stationary frequencies, substitution rates, α), we also want to integrate over different models now …
- What are the problems we need to solve?
  - Problem #1: we need to design proposals for moving between different models
  - Problem #2: those models have different numbers of parameters, we can not directly compare likelihoods
- Here we use MCMC to not only sample model parameters, but also models

Problem #1
Model Proposals

- Any ideas?

Problem #1
Model Proposals

- Split move

  Chose a set of substitution rates with > *1* member at random

  111222 (two-parameter model)

  and split it randomly into two rates

  111223 (three-parameter model)

- Merge move

  Chose two substitution rate sets at random

  111223

  and merge them into one substitution rate set

  111222

Problem #1
Model Proposals

- Split move

  Chose a set of substitution rates with > *1* member at random

  111222 (two-parameter model)

  and split it randomly in

  111223 (three-param

- Merge move

  Chose two substi[illegible]ion rate sets at random

  111223

  and merge them into one substitution rate set

  111222

Clear to everyone what the respective rate matrix looks like?

Problem #2
Sampling Different Models

- Use reversible jump MCMC (rjMCMC) to jump between models (posterior probability distributions) with different number of parameters (posterior distributions with different dimensions)
- The model proposal moves we designed are reversible jump moves!
- Evidently, we need to somehow modify our proposal ratio calculation …
- In general terms, the acceptance ratio is calculated as:

  r = likelihood ratio * prior ratio * proposal ratio * Jacobian

A Jacobian defines
a linear map from $R^n \rightarrow R^m$
at point *x*, if function *f(x)*
is differentiable at *x*

Problem #2
Sampling Different Models

- Use reversible jump MCMC (rjMCMC) to jump between models (posterior probability distributions) with different number of parameters (posterior distributions with different dimensions)
- The model proposal moves we designed are reversible jump moves!
- Evidently, we need to somehow modify our proposal ratio calculation …
- In general terms, the acceptance ratio is calculated as:

  r = likelihood ratio * prior ratio * proposal ratio * Jacobian

I will not provide further Details; see work by Peter Green (1995, 2003) who developed the rjMCMC methods

rjMCMC - summary

- Need to design moves that can jump back and forth between models of different dimensions (parameter counts)
- Need to extend acceptance ratio calculation to account for jumps between different models
- The posterior probability of a specific model (e.g., *JC* or *GTR*) is calculated as the fraction of time (fraction of samples) the MCMC chain visited/spent time/generations sampling within that model ...