Computing Phylogenetic Trees Based on Mitochondrial DNA (Eve's Tree)

Gaston H. Gonnet
ETH Informatik
Zürich, Switzerland

Speedup 2003, Basel, March 20th, 2003

Abstract

About 12 years ago, a landmark paper was published on the study of human prehistoric migrations based on mitochondrial DNA. The paper concluded that there was a common maternal ancestor of all present-day humans that lived about 200,000 years ago (nicknamed Eve). This paper was later partially discredited due to the lack of rigour and quality of the trees there constructed.

As a matter of fact, building phylogenetic trees with some guarantees of their quality is a much more difficult problem than anticipated. It required significant improvements in the algorithms, significantly faster computers, algorithms and technology to run these computations in parallel, new software systems and a few new ideas. We estimate that we have used about 10^{17} cycles of computing in this problem (cycles = hertz * time). Many of these ideas are applicable to other areas of scientific computation.

We will try to present the points most relevant to high-power computation that we have encountered building these trees.

Outline of the presentation

Problem definition

Pylogenetic Trees

Algorithms to build Trees

Computation

Randomization

Results

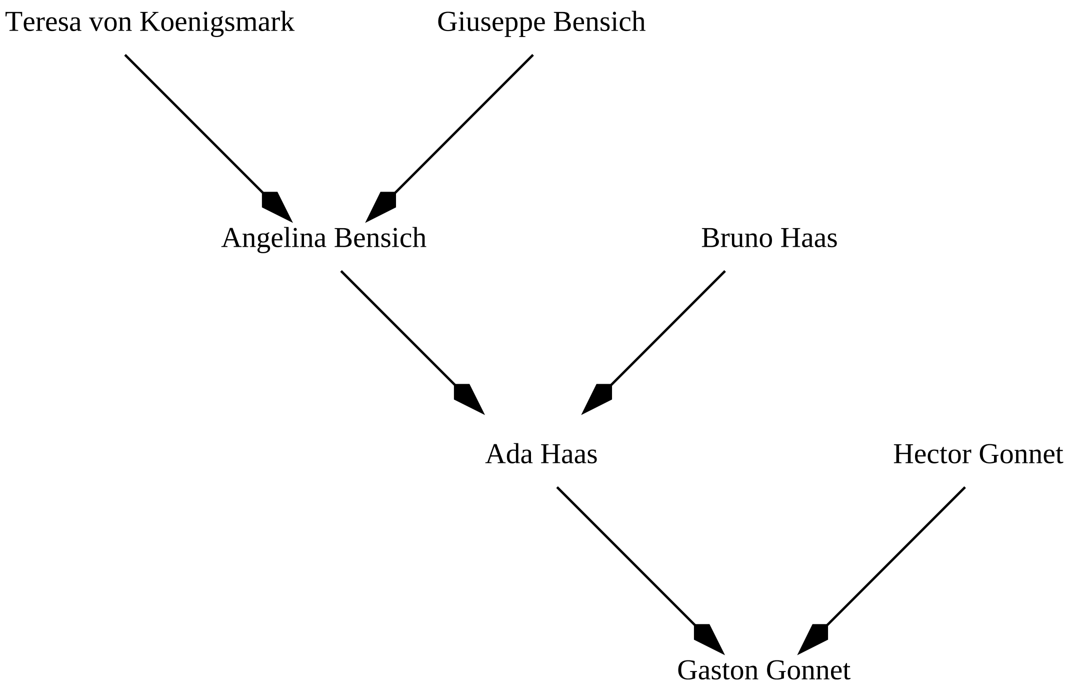
Conclusions

Problem Definition

- Mitochondrial DNA is a circular gene which is fundamental for life (energy), yet not part of our own DNA. Our mitochrondrial DNA comes from our mothers exclusively.

- No transpositions, (no mixture from father/mother). It becomes an ever diverging pure trace of phylogeny.

- 89 sequences from different ethnic groups around the world. About 16570 bases each, extremely well conserved (of all known sequences, only 913 positions have some mutations).


Teresa von Koenigsmark
Giuseppe Bensich
Angelina Bensich
Bruno Haas
Ada Haas
Hector Gonnet
Gaston Gonnet

The Evolution Tracing Dogma

Prehistoric (wo)men, as they migrated around the early world, carried around their MtDNA and this now becomes one of the few traces of their whereabouts.

By finding the common ancestors (and how close they are) we can estimate when two populations separated.

By comparing some of the computed distances with some of the times known and by other techniques, we can conclude the speed of mutations and hence the times of divergence.

Controversy

Quite a bit of controversy was generated around these results when it was shown that the trees that were used were not optimal.

More critiques arose as it was also shown that very similar scoring trees were quite different and could lead to different conclusions.

At the time, there was no hope of constructing (and proving) an optimal tree.

DNA, RNA and proteins

The data is very relevant, as in the studied period (last 200,000 years) there is little knowledge about human migrations.

It is too old for recorded history, and too recent for fossilized records. It is also too old for the more traditional archeological techniques, as man was quite primitive and left very few traces in the environment.

Hence this data fills a void.

DNA, RNA and proteins

Molecular information consists of sequences of DNA/RNA (4 symbols) or sequences of amino acids (20 symbols) normally called proteins.

A DNA sequence:

`ATGATCGTAAATAACACACACGTGCTTACCCTACCACTTTATACCACCAC...`

An RNA sequence and the protein it encodes:

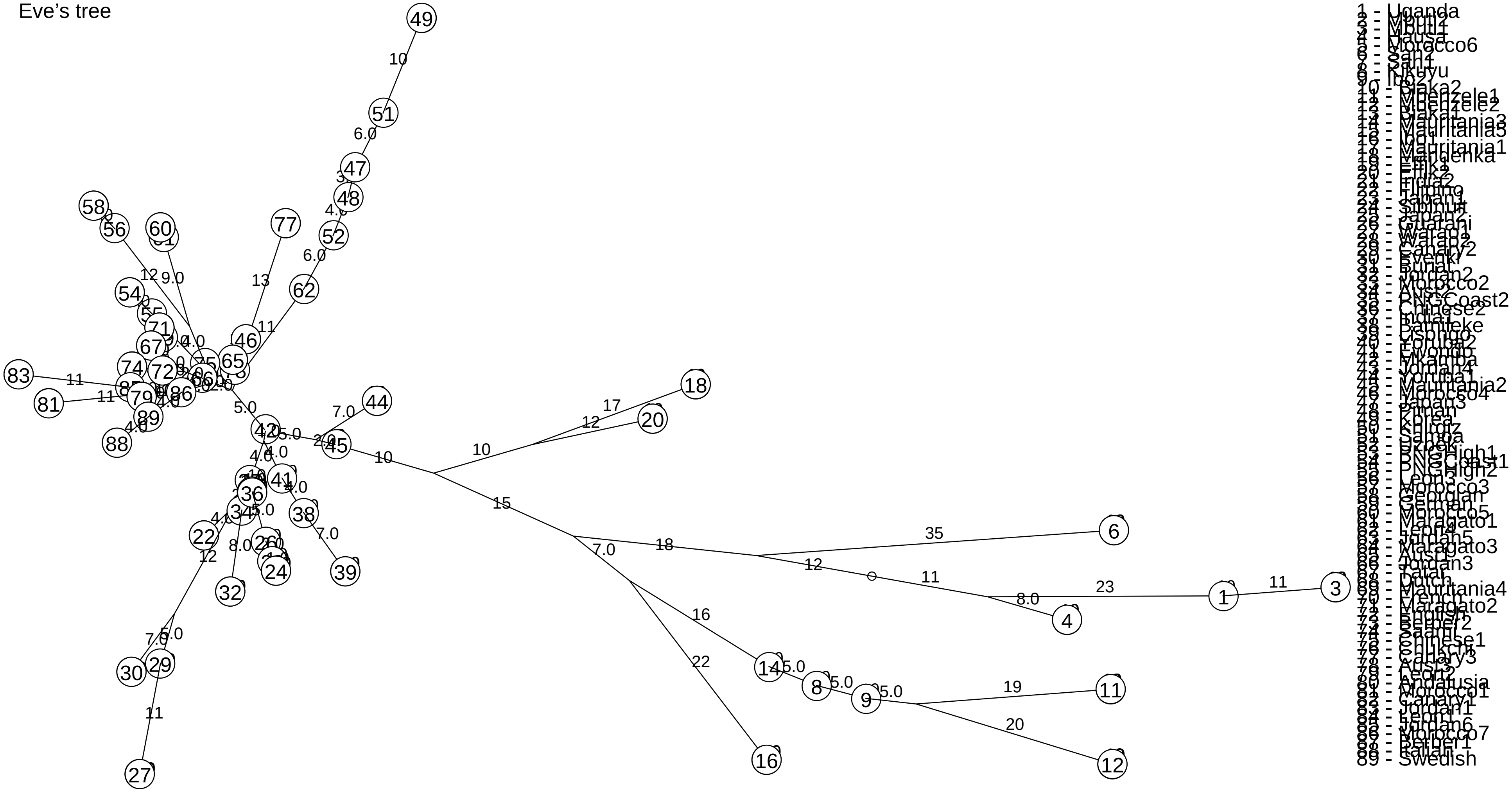
```
AUGAUCGUAAAUAACACACACGUGCUUACCCUACCACUUUAUACCACCAC...
 M  I  V  N  N  T  H  V  L  T  L  P  L  Y  T  T  T...
```

A protein or amino acid sequence:

`MIVNNTHVLTLPLYTTTTCHTHPHLYTDFTYAHGCYSIYHLKLTLLSDSTSLHGPS...`

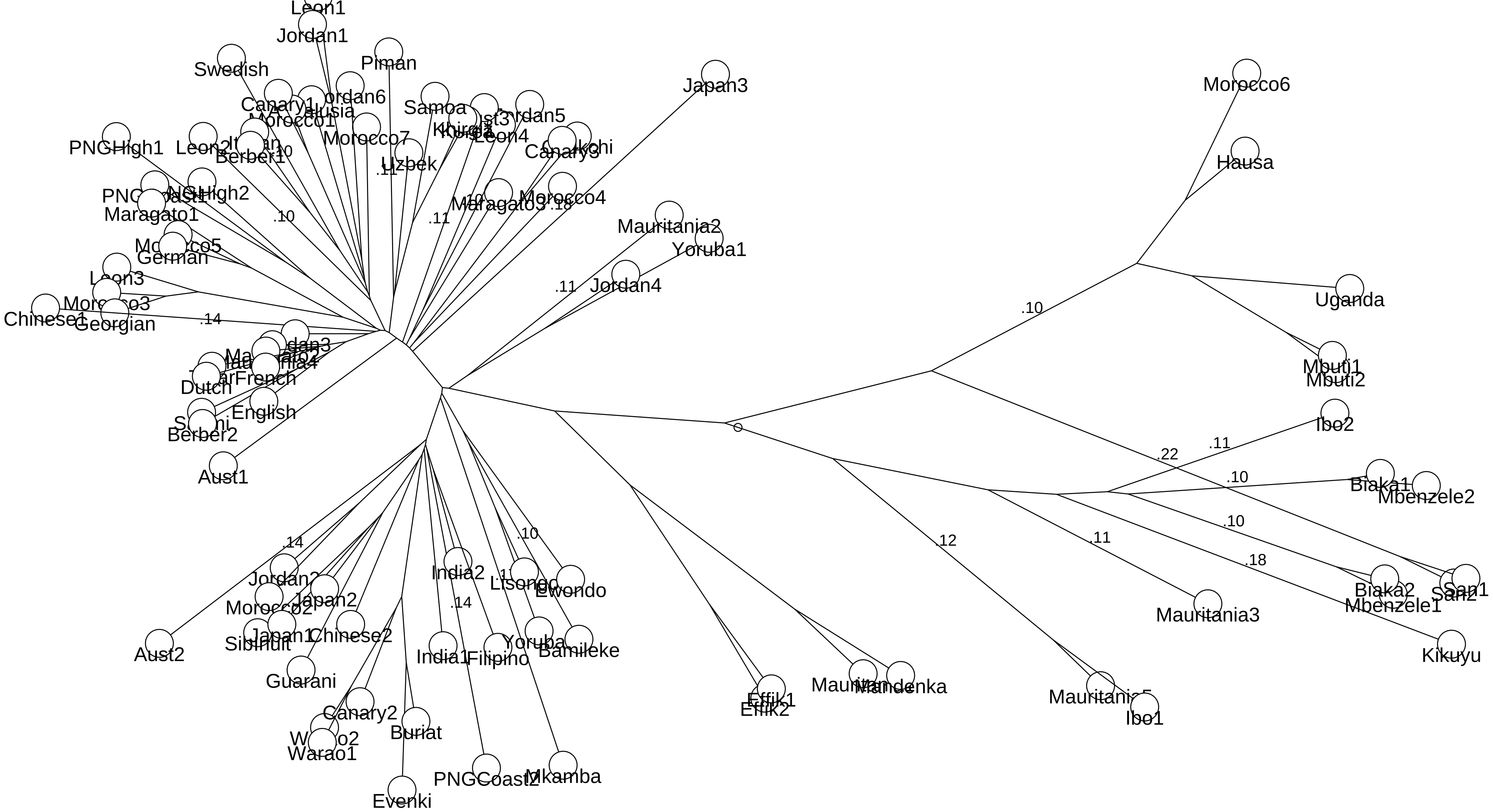
The most parsimonious Eve's tree

Eve’s tree



Best Distance Eve's tree

Eve's tree



Leon1
Jordan1
Swedish
Piman
Japan3
Morocco6
Hausa
Samoa
Morocco7
Uzbek
PNGHigh1
Berber1
Canary3
Morocco4
Maragato1
German
Mauritania2
Yoruba1
Jordan4
Leon3
Georgian
Chinese1
Uganda
Mbuti1
Mbuti2
Dutch
English
Berber2
Aust1
Ibo2
Biaka1
Mbenzele2
Jordan2
Japan2
Morocco2
Sibiriult
Chinese2
India2
Ewondo
Yoruba
Bamileke
India1
Filipino
Aust2
Guarani
Canary2
Buriat
Warao1
PNGCoast2
Mkamba
Evenki
Effik2
Mandenka
Ibo1
Mauritania3
Biaka2
Mbenzele1
San1
San2
Kikuyu
.10
.11
.14
.12
.18
.22

Phylogenetic trees, 3 types

- Distance. A tree which maximizes the agreement between a given matrix of distances and the distances calculated on the tree. The fit is normally a least squares fit.

- Parsimony. A tree which explains the different current species based on a tree which undergoes a minimal number of changes. (Occam's razor principle, the best theory is the simplest one).

- Maximum likelihood. Build a tree with known branch lengths which has the highest probability of happening given the known mutation probabilities.

Types of phylogenetic trees

Method	requires	produces	complexity
Distance	all-all matrix	topology+lengths	normal
Parsimony	Mul Seq Align	topology	complex
Max likel	Mul Seq Align	topology+lengths	very complex

Where is the complexity?

The complexity of the problem comes from the exponential number of topologies that a phylogenetic tree may have. Computing the branch lengths adds even more complexity.

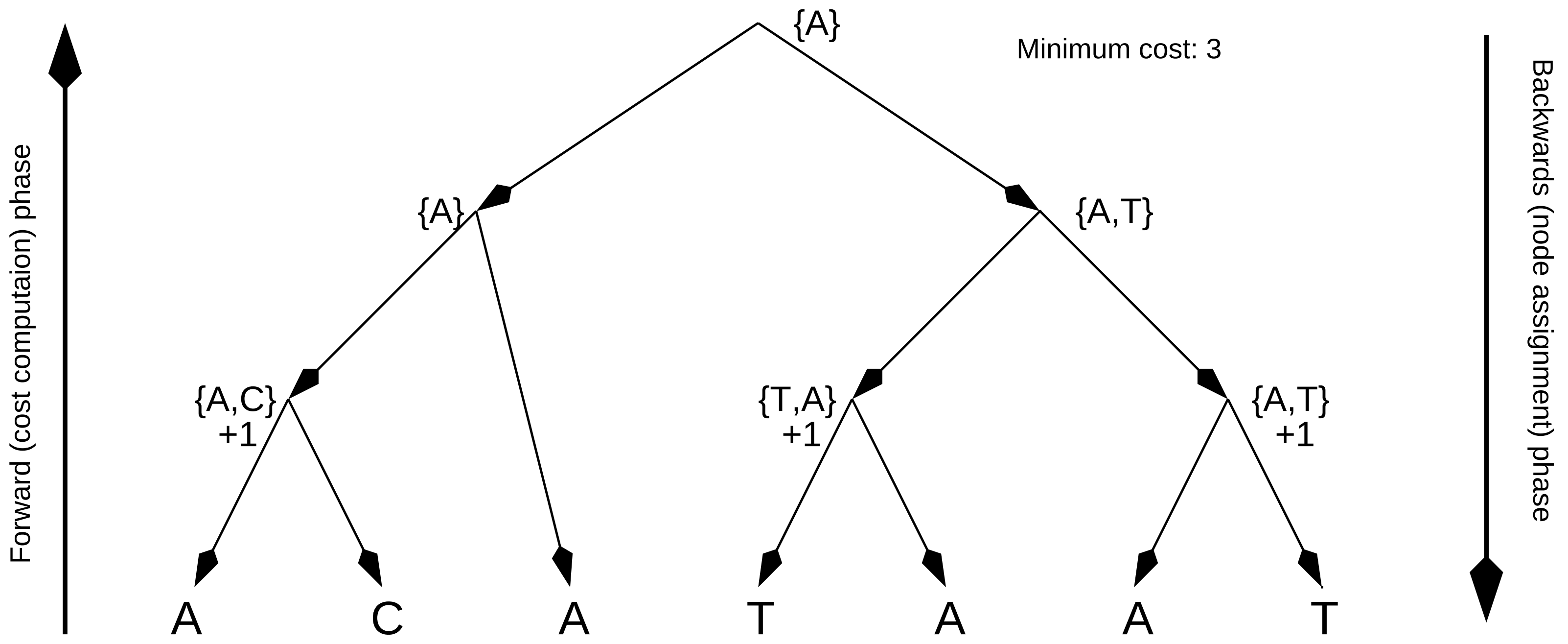
Nodes	3	4	10	89	n
Binary rooted	3	15	34459425	3.44×10^{159}	$(2n-3)!!$
Binary unrooted	1	3	2027025	1.97×10^{157}	$(2n-5)!!$

Parsimony computations

For various reasons, we are interested in computing the most parsimonious tree.

The core of the parsimony algorithm is the computation of the minimum number of mutations required to produce a phylogenetic tree with given leaves. This computation is done by dynamic programming, although in this particular case it is so simple that it is often described in simpler terms.

The following picture shows the essensial steps which have to be followed for each position where there is variation (positions in the multiple sequence alignment which are uniform can be ignored).


{A}
Minimum cost: 3
{A}
{A,T}
{A,C}
+1
{T,A}
+1
{A,T}
+1
A
C
A
T
A
A
T
Forward (cost computaion) phase
Backwards (node assignment) phase

Parsimony computations

In the forward phase for each node we intersect the sets of both descendants. If the intersection is empty, we add 1 to the cost and we assign the union to the node. If it is not empty, we assign the intersection to the node. The cost of the minimal assignment is the sum of all the unions we had to do.

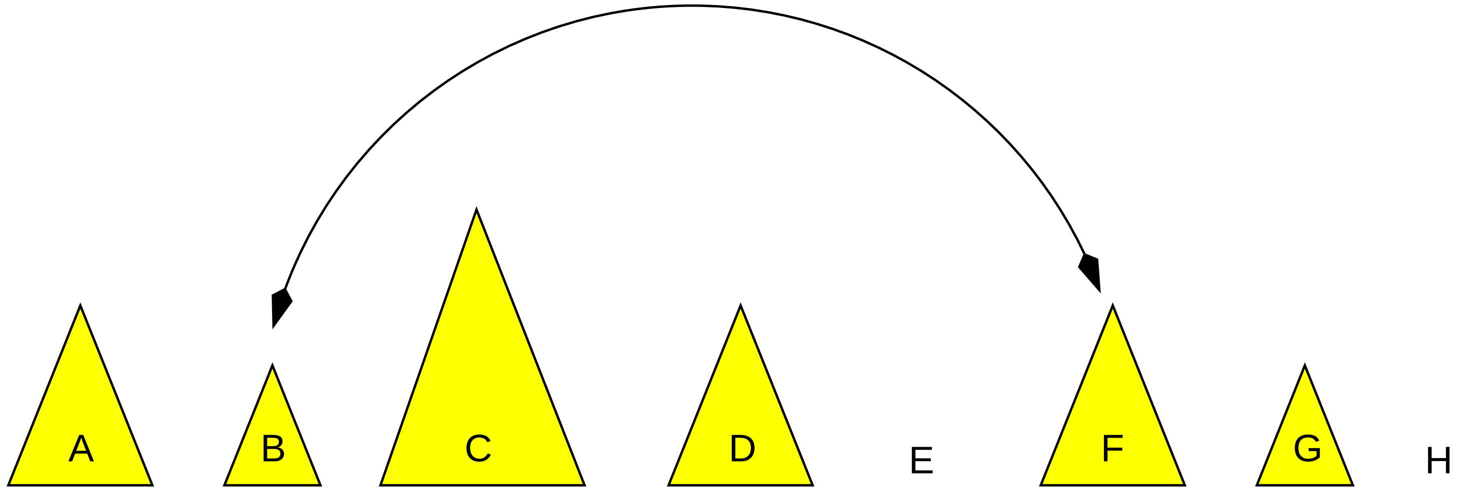
The backwards phase does the node assignments. Normally we are interested in the cost alone, so we will only do the forward phase.

Algorithms: Neighbour joining

Neighbour joining is a greedy algorithm for building phylogenetic trees. It works with various measures of complexity and it repeats the following steps:

1. Select the pair of nodes (or subtrees) which are closest
2. Join these two nodes (or subtrees) into a single subtree. Compute the distance between the two components (if needed).
3. Recompute the distance between all other nodes/subtrees and the new subtree just built.

Neighbour joining iteration



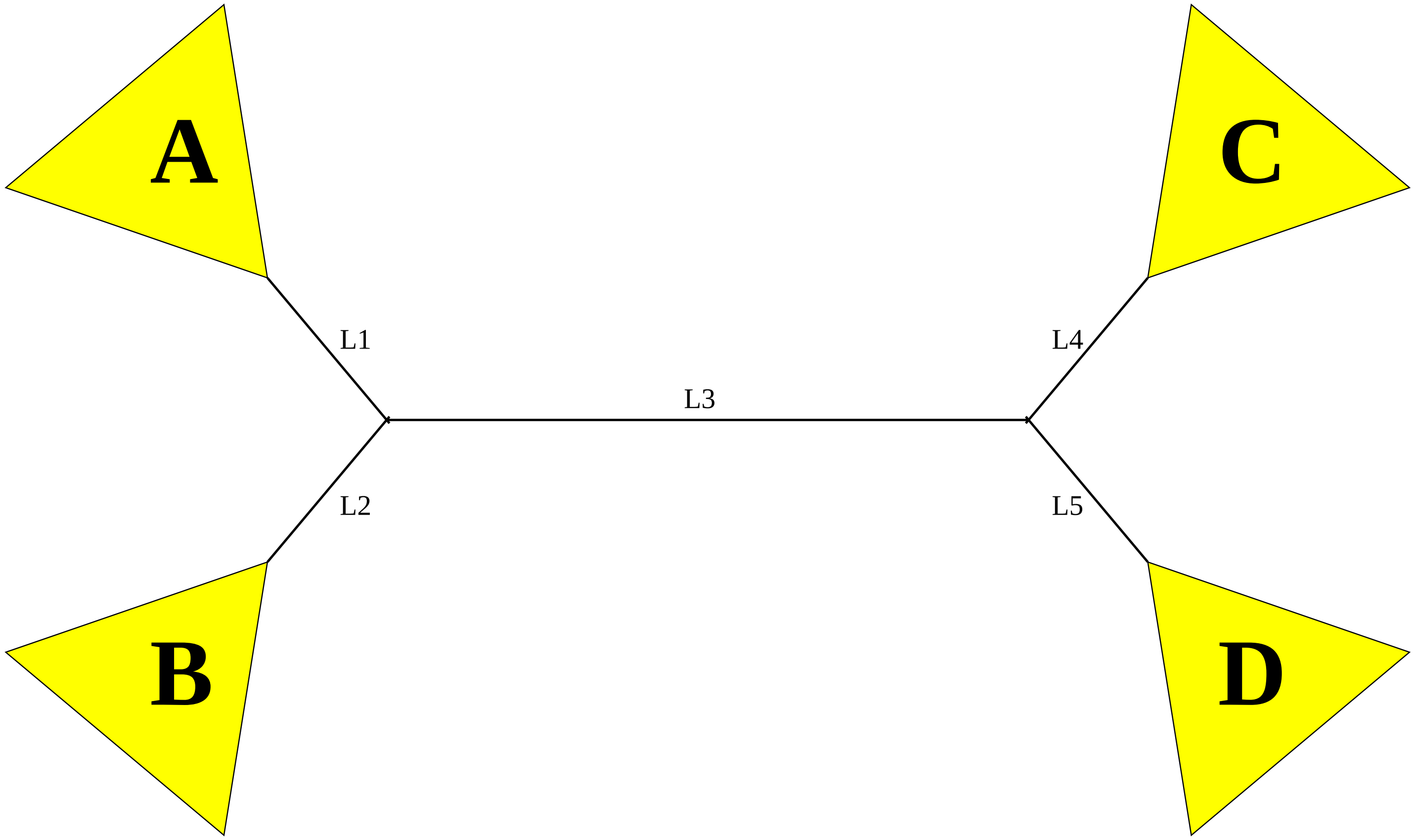
Join the pair with the lowest cost

Heuristic: 4-optim

4-optim is a heuristic to improve phylogenetic trees. It can be applied to all types of trees (measures). It analyses a tree and may produce rearrangements which will improve its cost.

4-optim can be applied repeatedly until no more improvements can be found.

4-optim analyses each internal branch/edge which is not attached to a leaf at any of its ends. Such a branch/edge can be represented as


A
C
L1
L4
L3
L2
L5
B
D

How 4-optim works

For each internal branch/edge there are 2 additional configurations (3 in total) which reattach the 4 subtrees in essentially different ways.

The goal of 4-optim is to check these two alternatives, computing what would be the resulting cost of the new trees. If any of them is better, then changing the tree.

There are $n-3$ internal branches/edges, so each tree has $2n-6$ different trees that will be explored by 4-optim. These $2n-6$ trees will be called the "neighbours" of the tree.

Neighbour searching heuristics

As it turns out to be, 4-optim is a special case of the general neighbour-searching heuristics.

Neighbour searching heuristics start at a point x_0 (in our particular case, a point is a tree), compute the cost of all neighbours of x_0, and in case a better value is found, say at x_1, then x_0 is replaced by x_1 and the search continues.

When we cannot find a neighbour with better cost, the heuristic stops (as we have found a local optimum of the cost).

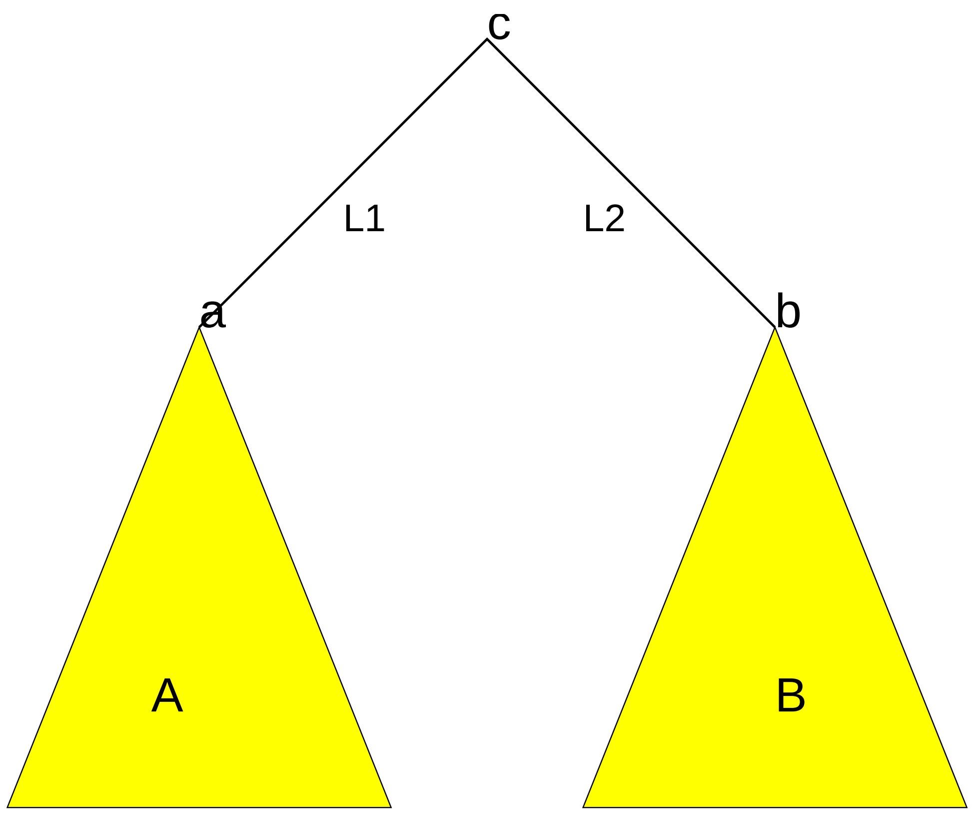
Neighbour searching II

The neighbour searching techniques may be coded with two different strategies:

- Steepest descent - this describes the mode where we explore all the neighbours before changing x_0 and we change it to the neighbour with the best cost.
- Early abort - this corresponds to the case where as soon as we find a better neighbour, we select it.

Neighbour searching III

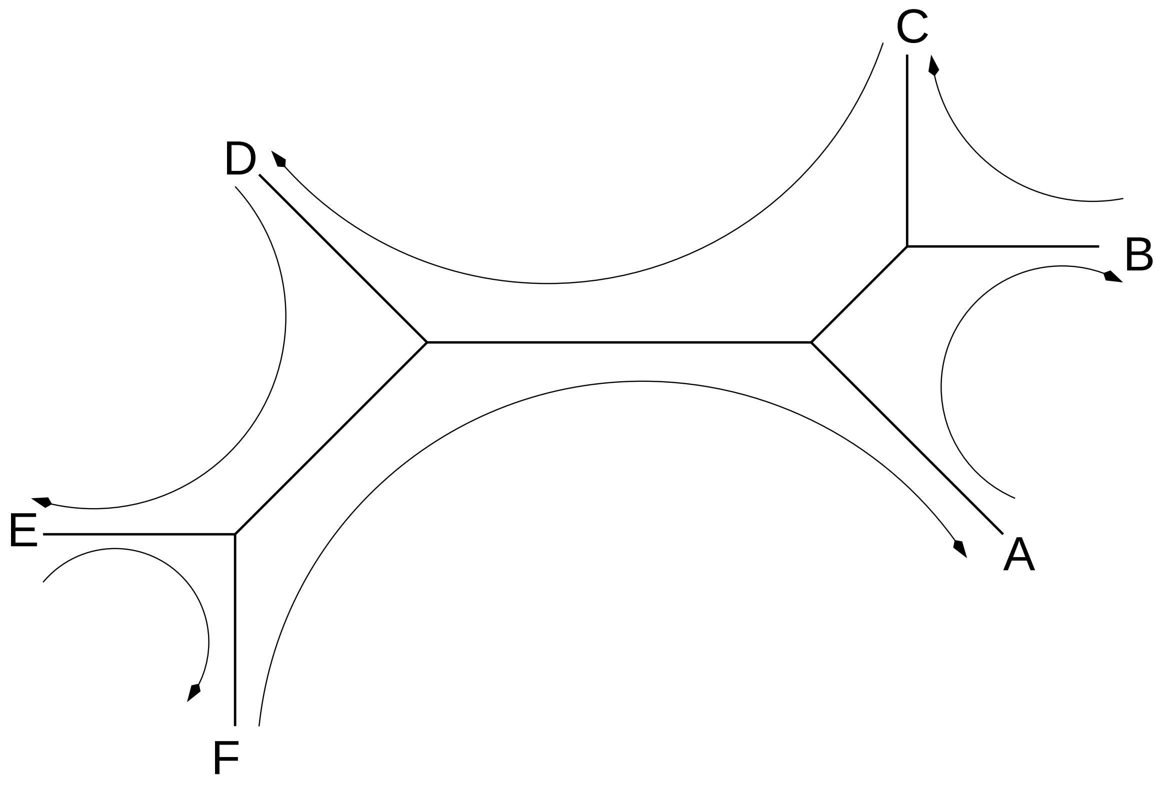
Finally, it is important to code the computation of the cost bottom up (on the tree) in such a way that each node can be computed based on the information of its two descendants. If this can be done, then the computation with neighbour heuristics can be significantly accelerated.



All the information of A is in a
All the information of B is in b
c can be constructed from a and b
alone (including L1 and L2).

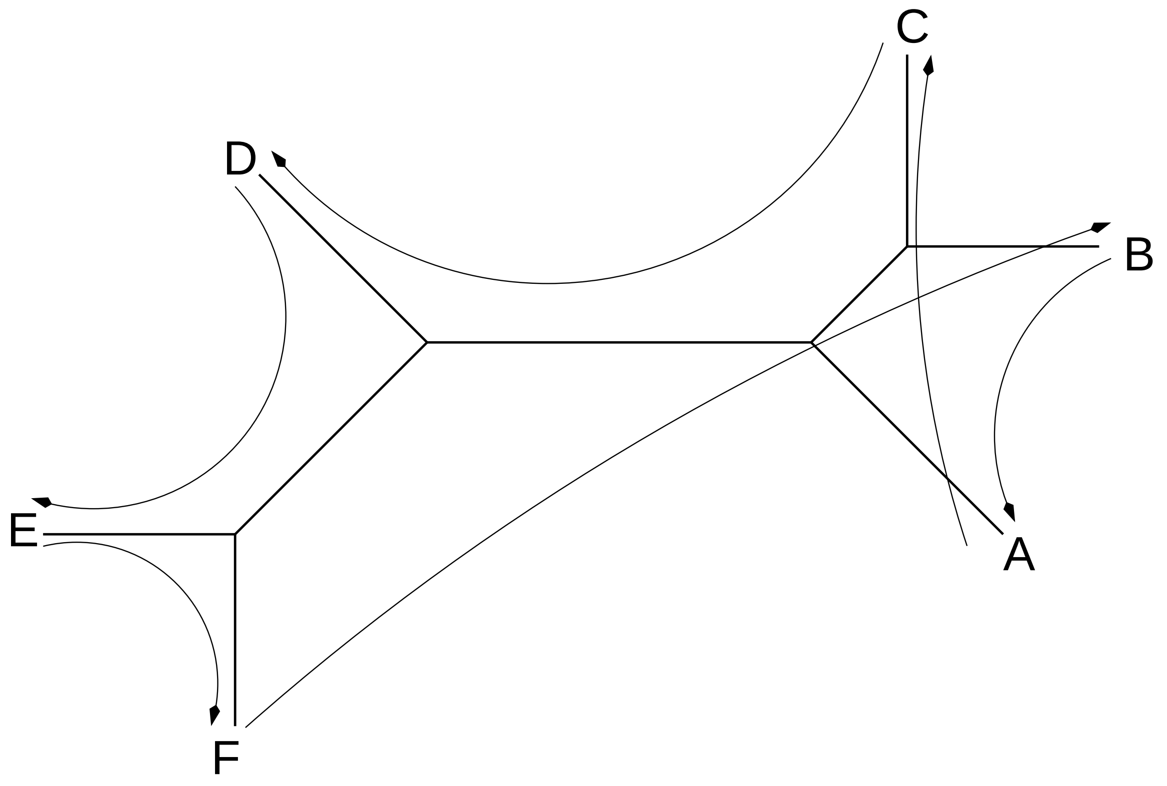
Circular Tour Algorithms

A circular tour of a tree is a clockwise tour of its leaves in one of its planar representations.



Non Circular Tour

Reversing two leaves which are not siblings (e.g. A and B) causes the length of the tour to increase.



Circular Tour Algorithms II

Circular tours can be computed without knowing the tree.

A circular tour is a minimum length tour of all leaves.

We can apply the Traveling Salesman Problem to the all-against-all distance matrix to compute such a tour.

Dynamic programming or k-greedy algorithms

Dynamic programming is a general technique which can be applied to tree construction as well.

Generally, instead of working with the best tree, we will work with a set of good trees.

At every stage we compute all possible variants from the previous set of trees and keep the best k resulting trees.

Computation

- Cluster of PC's (or equivalent) running Linux.
- Goal: cheapest Ghz equivalent.
- As a consequence we need (effective) parallelization (ineffective parallelization will force us to look into other hardware alternatives).

Very long computation paradox

Never start a computation that requires 3 years or longer.

Wait, you will finish it quicker.

How many cycles per \$

Model:

Assume you rent the computer for 18 months (or buy and sell), its half life according to Moore's law. This will cost you half the price.

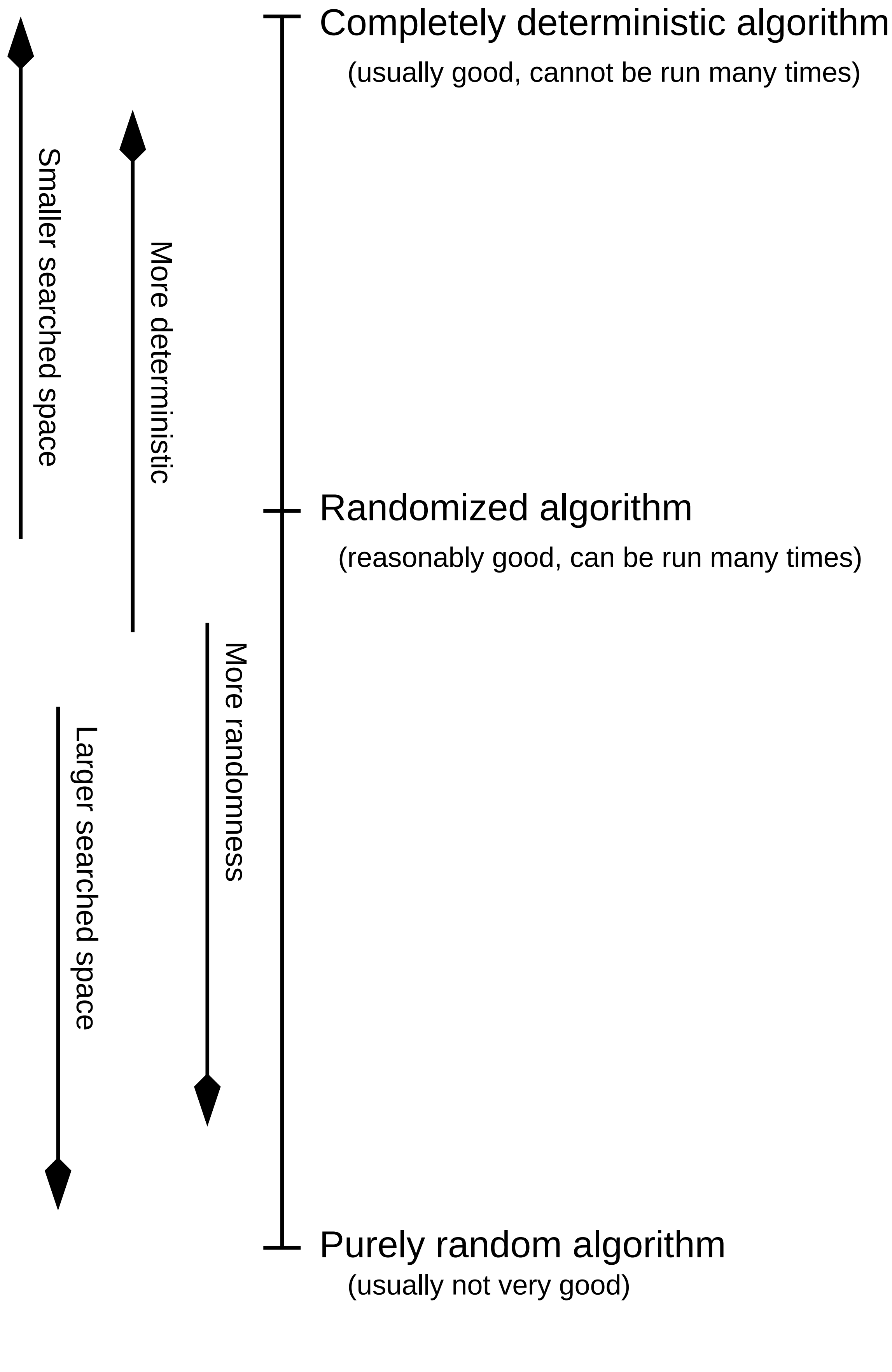
$$\frac{18 \times 30 \times 24 \times 3600 \times 10^9 \times 2.4 \times 2}{1000/2} = 4.479 \times 10^{14}$$

Even simpler, since 18 months is approximately 500 days, the cost of 24 hours of computing is the cost of the computer divided by 1000.

Randomization

The role of randomization, for us, is to allow parallelism. We want:

- algorithms which are competitive with the deterministic algorithms
- can be run multiple times due to their randomness.



Completely deterministic algorithm
(usually good, cannot be run many times)
Randomized algorithm
(reasonably good, can be run many times)
Purely random algorithm
(usually not very good)
Smaller searched space
More deterministic
More randomness
Larger searched space

Randomization II

These techniques are applicable for complicated solution spaces (e.g. the space of all possible phylogenetic trees).

By repeatedly starting at a random point and searching for a local minimum we obtain important information about the functional *landscape*.

Clear advantages over other techniques, like simulated annealing, deterministic, etc. (on top of allowing parallelism).

Controlled randomness

There are many ways of achieving controlled randomness, i.e. a tunable parameter which will continuously map algorithms from deterministic versions to purely random versions.

For example, the neighbour joining algorithms can be randomized with the following strategy for pair selection:

```
pair_to_join := random_pair( subtrees );
for z in all_pairs( subtrees ) do
    if Rand() < p and cost(z) < cost(pair_to_join) then
        pair_to_join := z fi od;
```

Results

We show the results for various randomized algorithms.
Randomized distance trees.
Purely random parsimony trees.
Randomized neighbour joining parsimony trees.
Randomized circular tour parsimony trees.

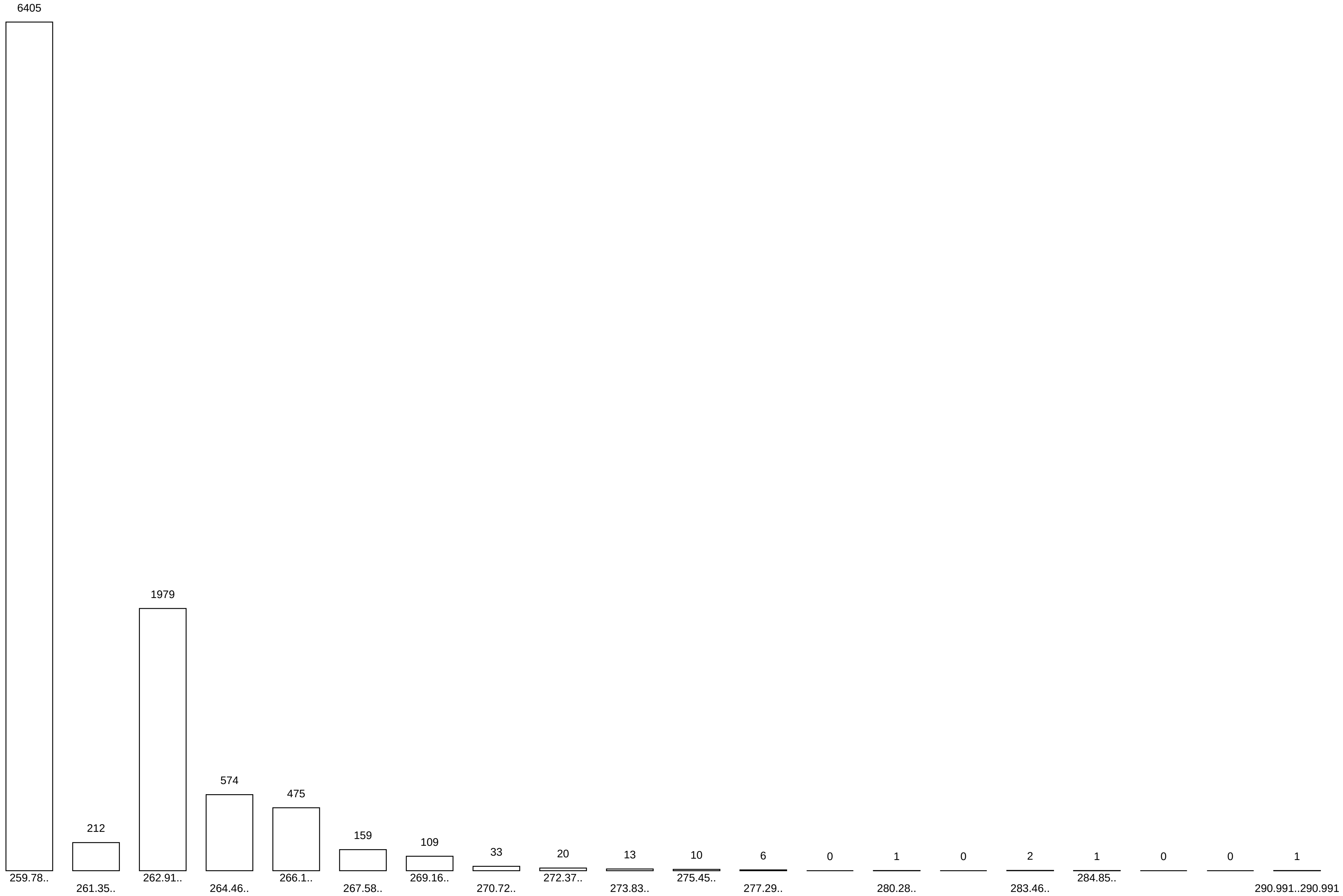
After the initial tree was built, 4/5/6-optim were used until a local minimum was found. In all cases the building algorithm was run 1000s of times. We show histograms of the number of times each local minimum value was achieved.

Results for Distance Trees

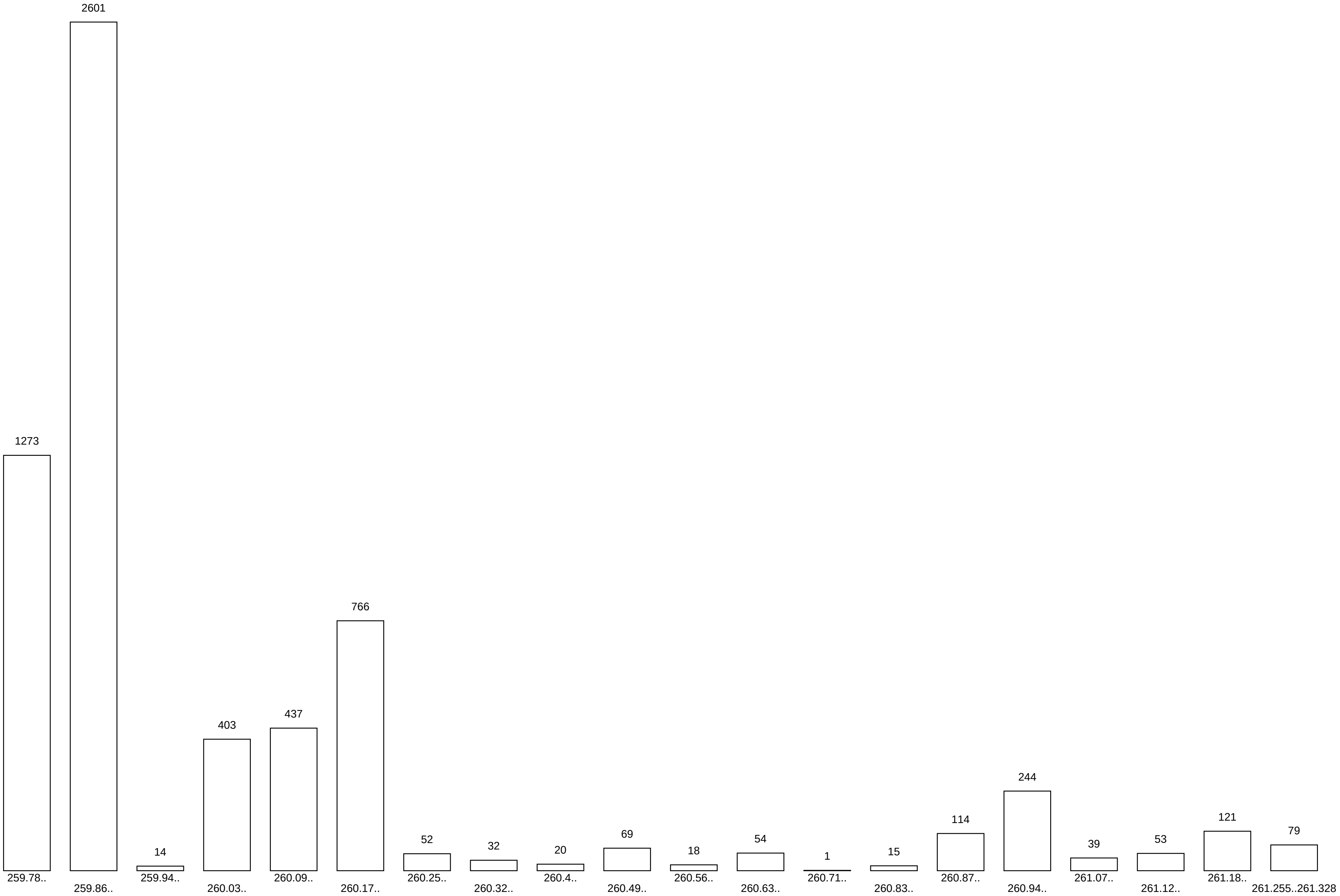
Random distance trees with default settings.

We find 860 different local minima. The smallest occurs 917 times out of 10000 trials.

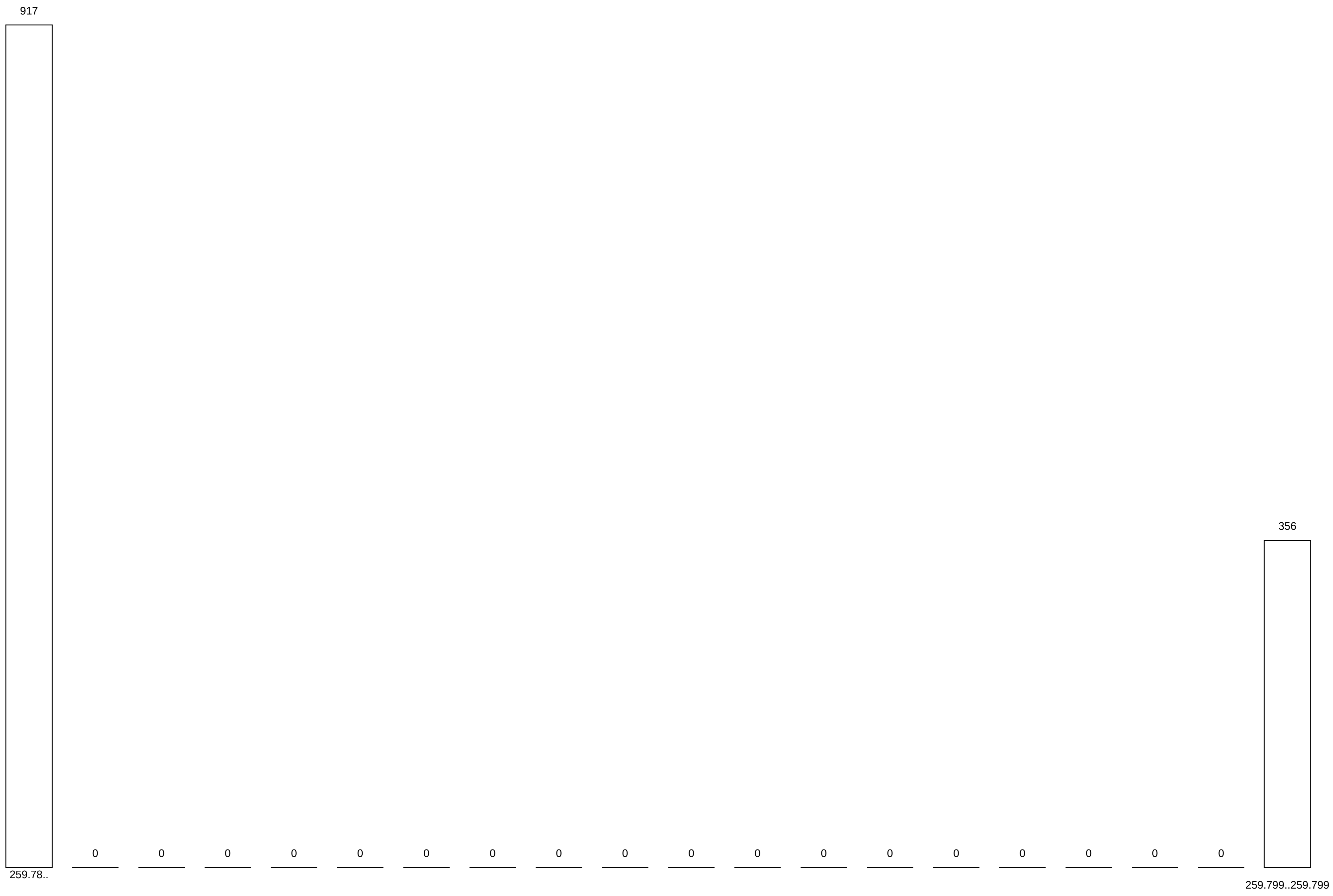
10000 trees from 259.776701 to 290.990677, 860 different minima, 0.141 secs/tree



6405 trees from 259.776701 to 261.328288, 165 different minima, 0.141 secs/tree



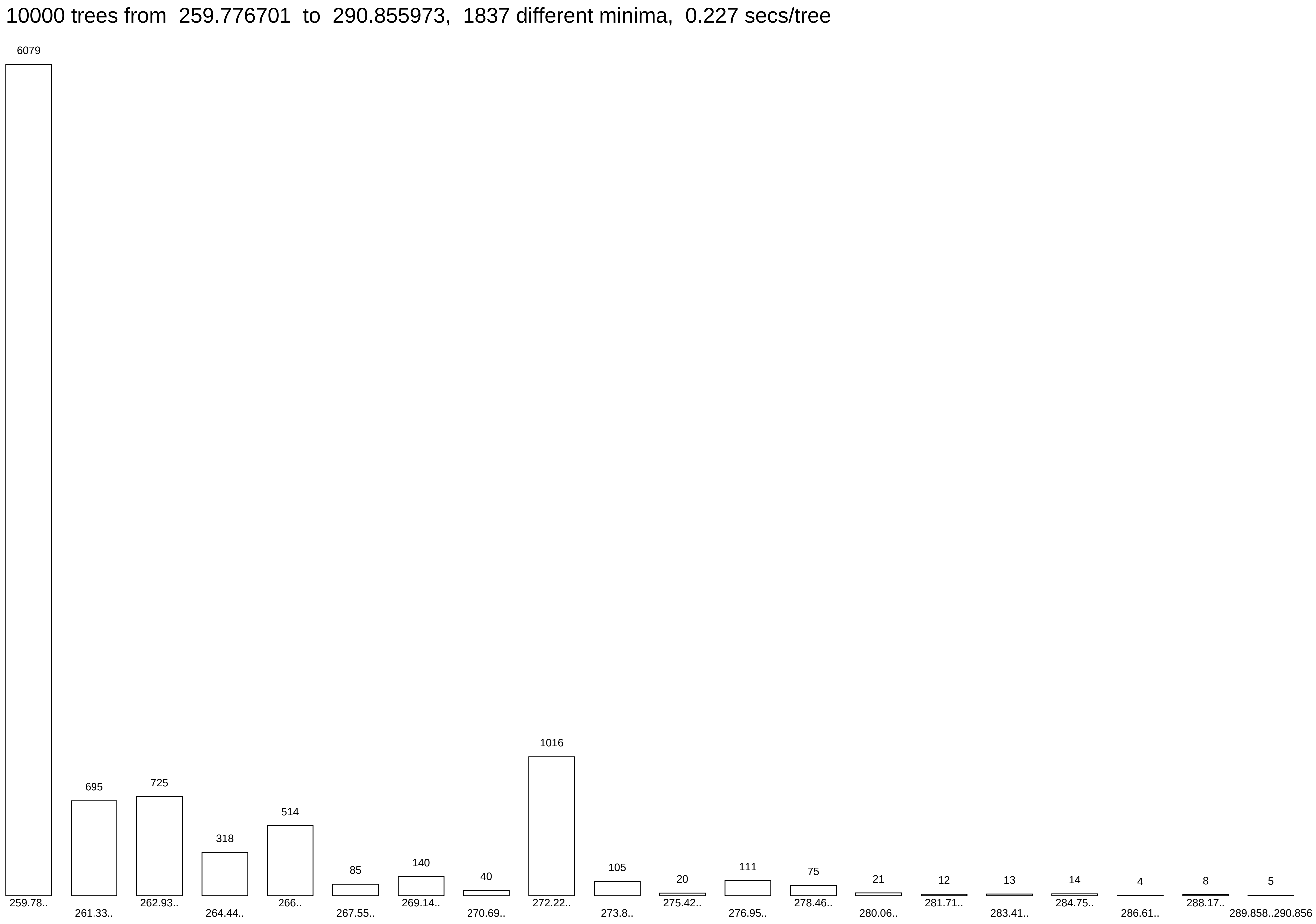
1273 trees from 259.776701 to 259.799397, 2 different minima, 0.141 secs/tree



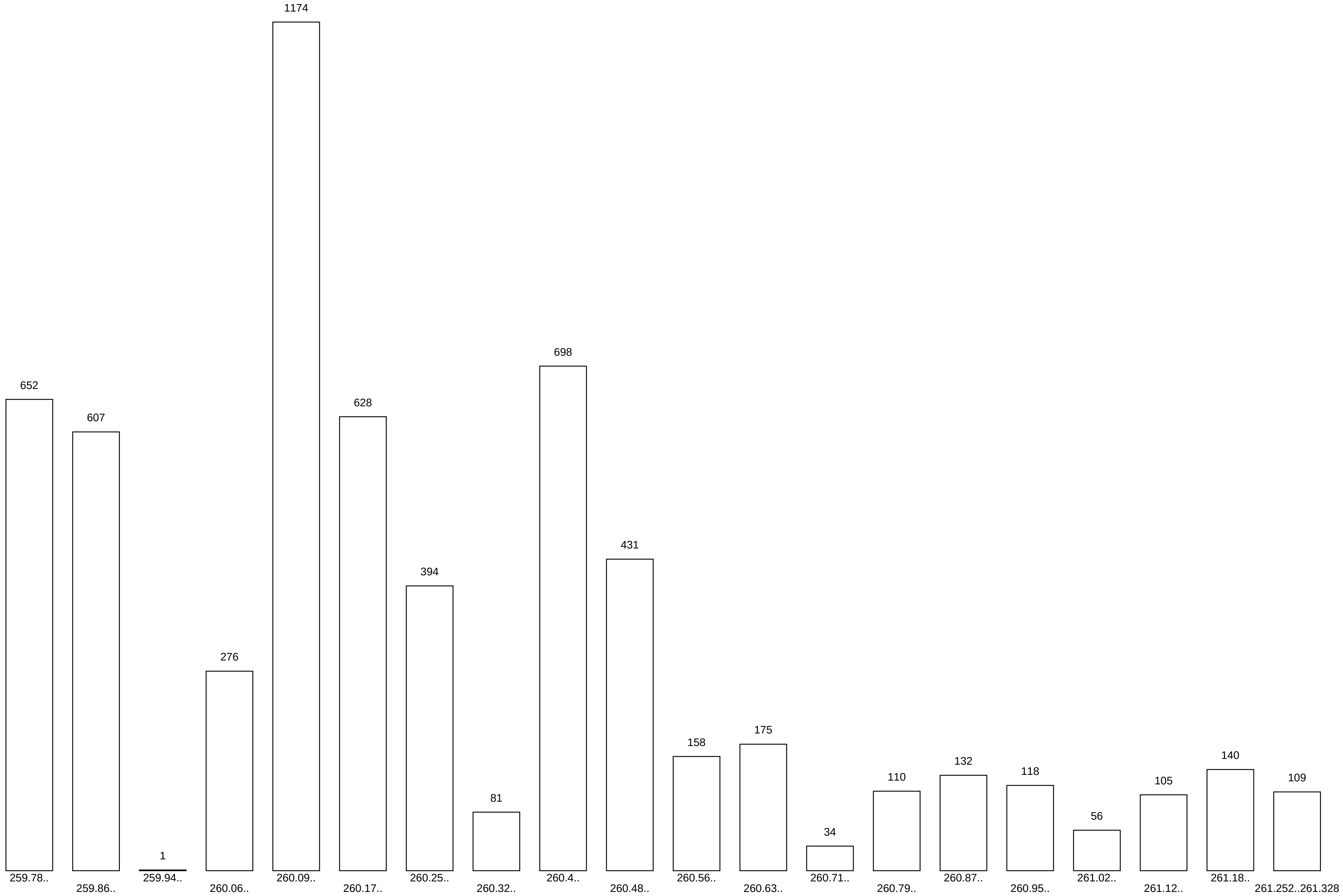
Results for Distance Trees

Random distance trees with increased randomness of initial trees.

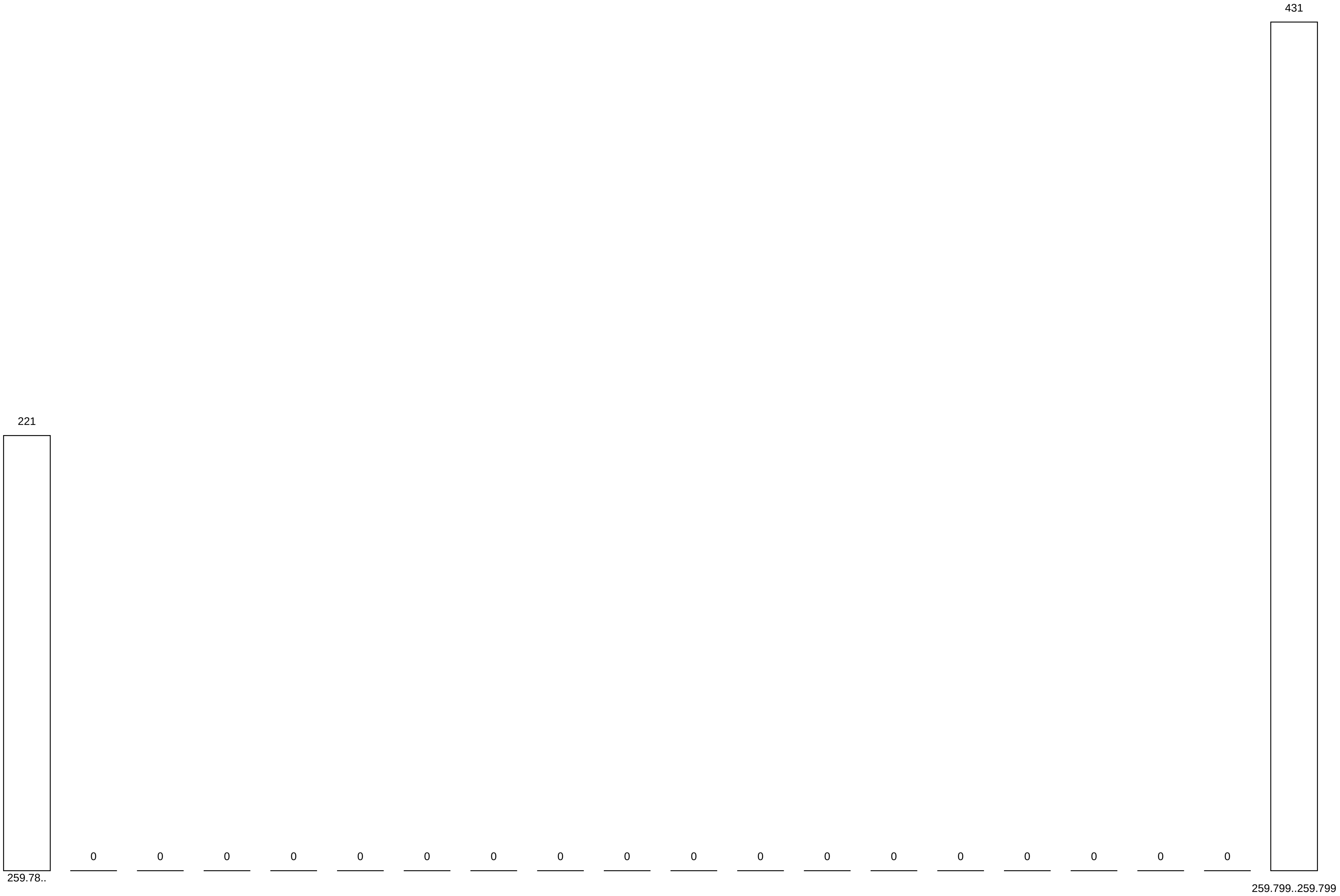
We find 1837 different local minima. The smallest occurs 221 times out of 10000 trials. It requires 61% longer to compute.


10000 trees from 259.776701 to 290.855973, 1837 different minima, 0.227 secs/tree
6079
695
725
318
514
85
140
40
1016
105
20
111
75
21
12
13
14
4
8
5
259.78..
261.33..
262.93..
264.44..
266..
267.55..
269.14..
270.69..
272.22..
273.8..
275.42..
276.95..
278.46..
280.06..
281.71..
283.41..
284.75..
286.61..
288.17..
289.858..290.856

6079 trees from 259.776701 to 261.328288, 280 different minima, 0.227 secs/tree



652 trees from 259.776701 to 259.799397, 2 different minima, 0.227 secs/tree

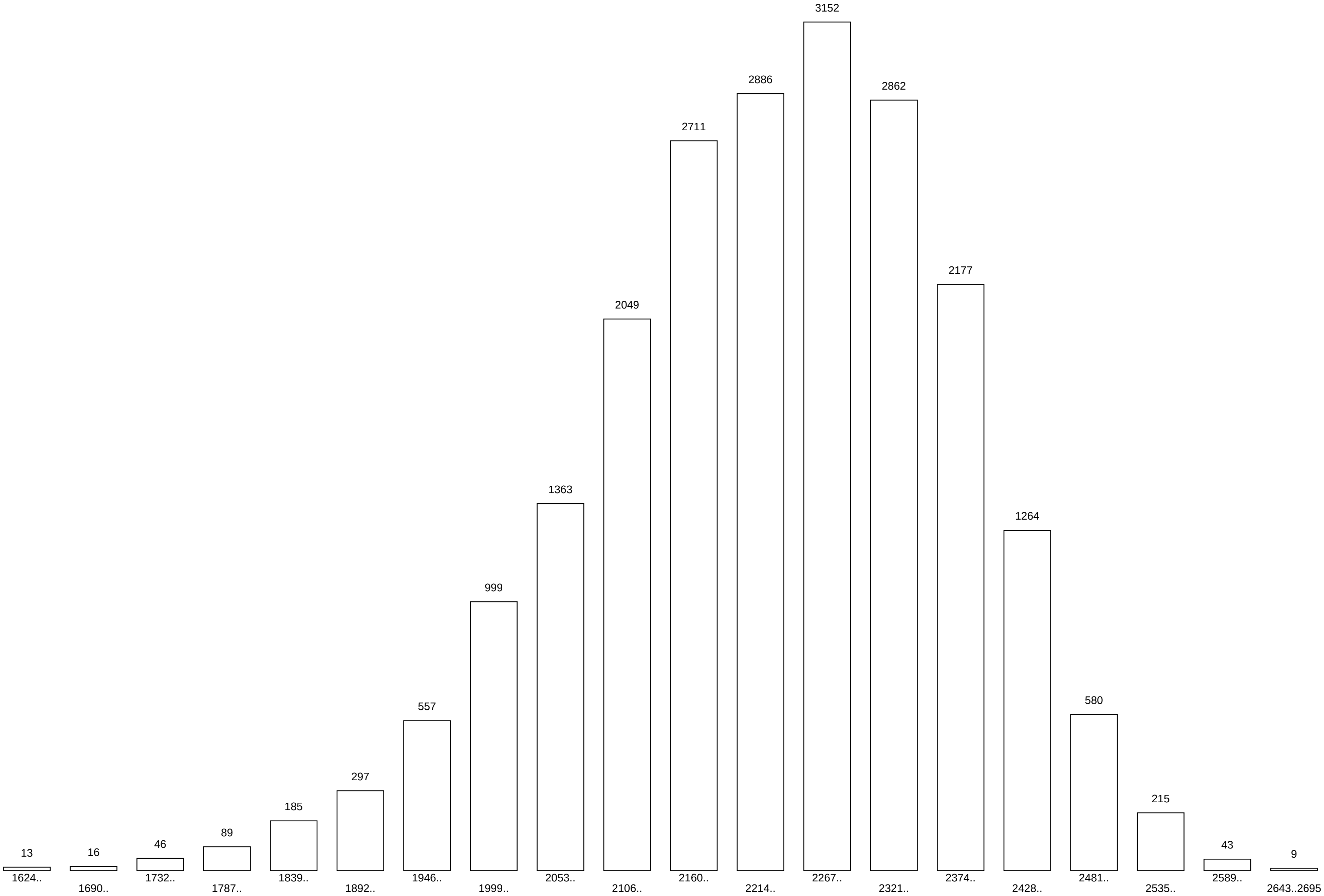


Purely random, parsimony

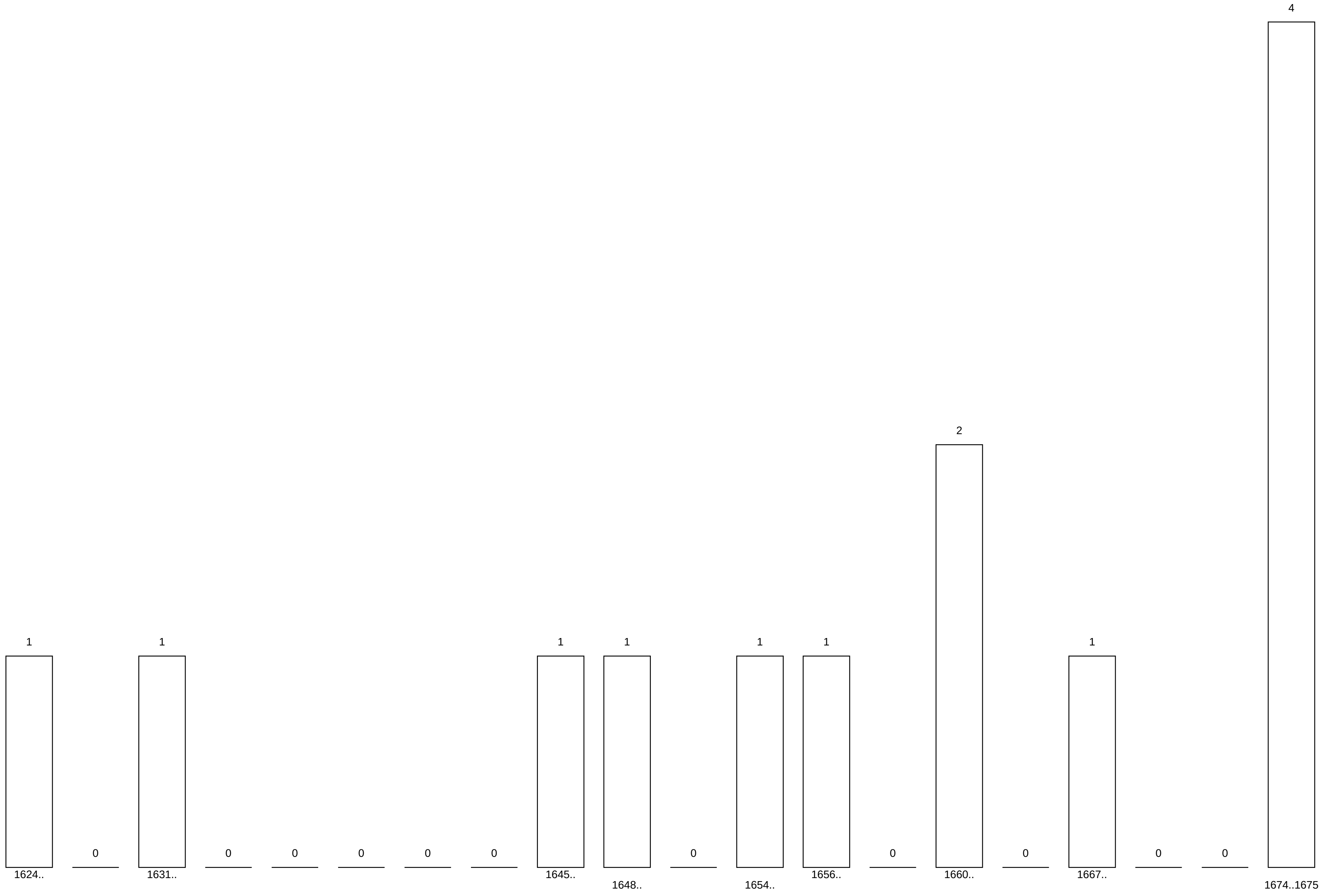
Purely random initial parsimony trees.

We find 885 different local minima (this is over the integers, so we are certainly going to have different trees with the same local minimum value). The smallest occurs 1 time out of 21513 trials.

21513 trees from 1624.000000 to 2695.000000, 885 different minima, 289.739 secs/tree



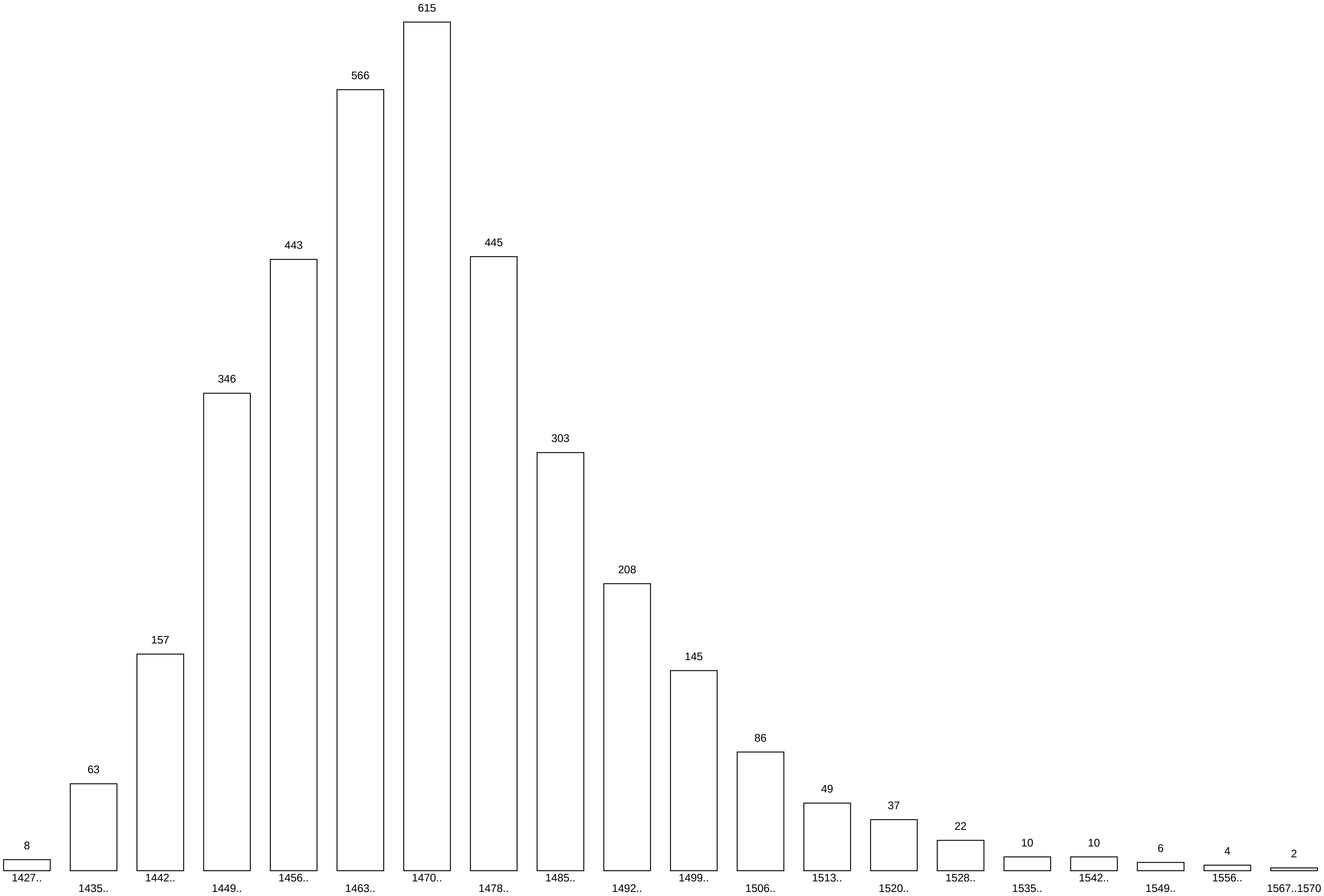
13 trees from 1624.000000 to 1675.000000, 11 different minima, 289.739 secs/tree



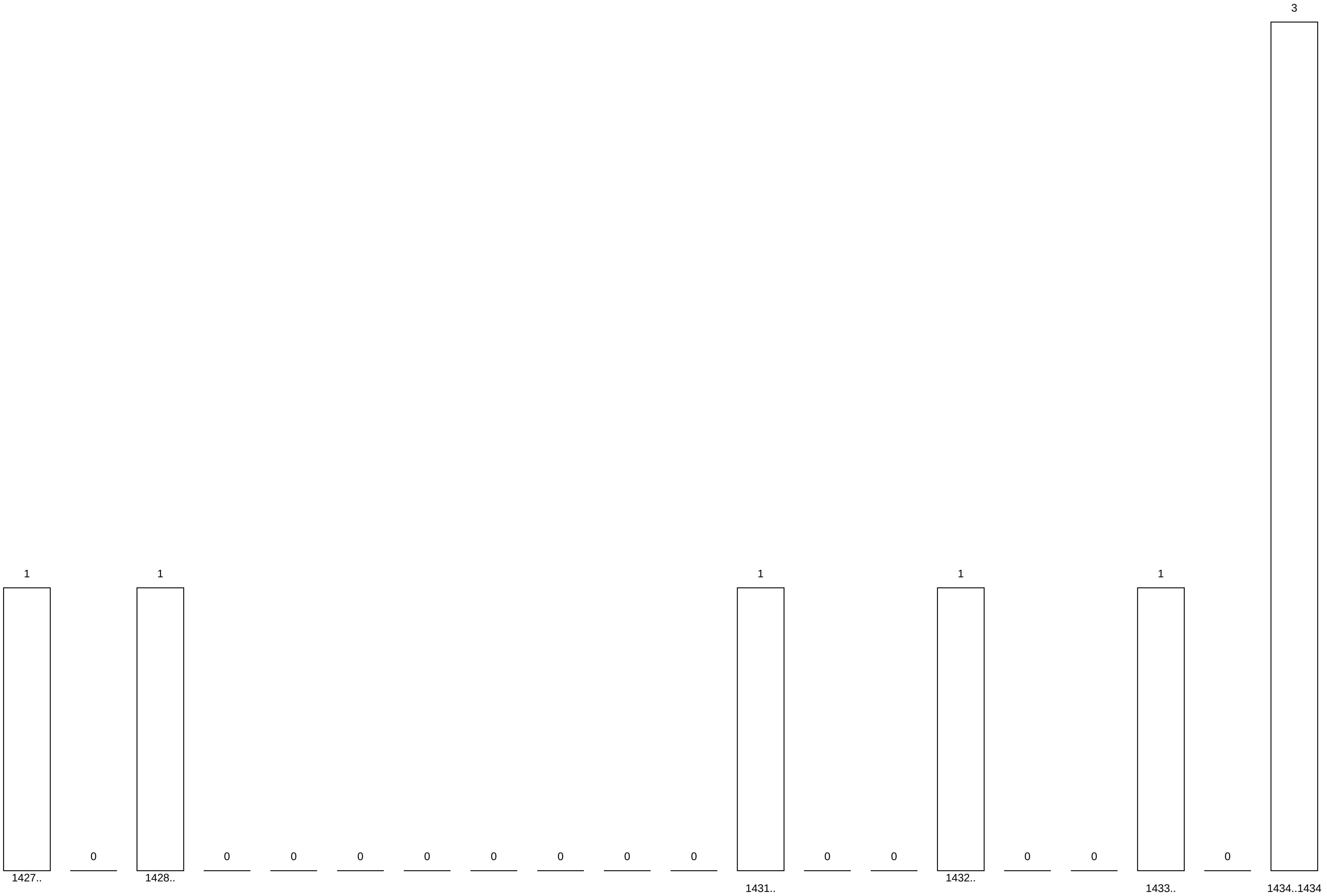
NJ random, parsimony

Random neighbour joining parsimony trees.

3525 trees from 1427.000000 to 1570.000000, 125 different minima, 73.375 secs/tree



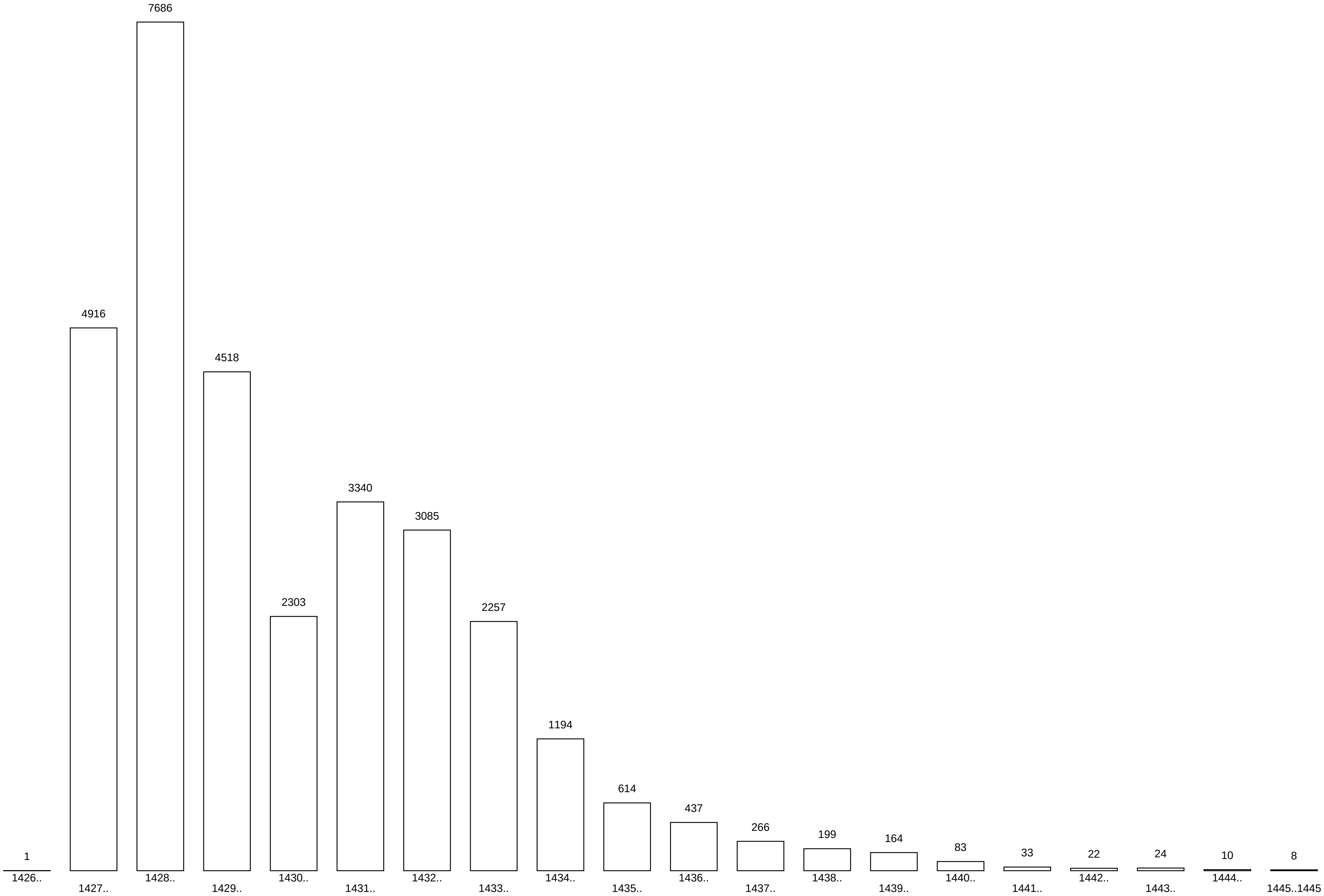
8 trees from 1427.000000 to 1434.000000, 6 different minima, 73.375 secs/tree



Circular tour, parsimony

Randomized circular tour parsimony trees.

31160 trees from 1426.000000 to 1445.000000, 20 different minima, 59.700 secs/tree



Acknowledgement

We should be grateful to MS – by producing lousy software a need for mass-market cheap cycles has been created – and the scientific community benefits.