# Origins and losses of parasitism

an analysis of the phylogenetic tree of life with a parsimony-like algorithm

# Abstract

# Contents

# 1 Introduction

This paper is about the further development of parsimony algorithms for non-binary trees, applied to the currently largest phylogeny synthesis tree of Open Tree Of Life, with the application to the ancestral state reconstruction of parasitism.
Researchers of the phylogenies have been dealt with the ancenstral state reconstruction in the 60s. The first methods were only brute force TODO: Quelle, siehe Fitch: Camin and Sokal 1965 . Next came a set of parsimony algorithms such as: Fitch-parsimony [Fit71], Wagner-parsimony [SM87] ... TODO: weitere? .
With more and more data, there is now the possibility to use more information to calculate the probabilities of the ancestral states. In addition to the states of the leaves, algorithms could also use branch lengths. The likelihood based algorithms came more in interest.
Our focus came with another 'data extension'. We wanted to work with the biggest phylogenetic tree that exists at this moment, which goes over all observed species. For most TODO: most? species there is no phylogeny, but only a taxonomic classification. So the biggest 'phylogenetic tree' is a synthesis of phylogenetic trees filled with a taxonomic tree given by Open Tree of Life [HSA+15]. This tree is not binary and therefore the developed algorithms are not directly applicable.
In this work, we have looked at the algorithms that are generally suited to our data, to develop them further for the not binary case, and finally to compare their usability with our sythesis tree.
We have decided to consider only parsimony algorithms since we have no information on branch lengths and no other additional information like different transition probabilities of our states.

# 2 Methods

This work consist of two aspects. One is the application of the algorithm to our question

Diese Arbeit besteht aus zwei Fragestellungen gute Algorithmen zu finden und sie auf unser Anwendungsbeispiel anzuwenden.

Zur Bewertung der gefundenen / entwickelten algorithmen haben wir eine Simulation ausgeführt.

Auf der Anderen Seite haben wir unser reales Problem aufgestellt, für die Algorithmen vorbereitet und diese angewendet.

Metadata -> Simulation -> real Data

## 2.1 Metadata analysis

Properties of real Data - Metadata analysis

There are some Parameters to find out or notice:

- transition probabilities of tags
- multifurcation
- nr of parasites to free-living
- nr of unknown nodes

## 2.2 Simulation

- build random binary trees, tag these (parameters: parasites vs free-living, beta-distribution)
- run fitch-parsimony, wagner-parsimony, our parsimony like algorithm
- build not binary tree (poisson distribution?)
- run new algorithms
- compare trees (distances)

i) build random binary trees

ii) tag tree

iii) multifurcate tree

iv) run maximum parsimony algorithms

  - Fitch
  - Sankoff (Castor package)
  - my algorithm

v) Evaluation

### 2.2.1 random binary tree

To get a random binary tree, I used the Phylo package from biopython. They offer a randomized function which returns a BaseTree [1]:

```
from Bio import Phylo
Phylo.BaseTree.Tree.randomized(number_leafnodes)
```

From the BaseTree class:

[1] https://github.com/biopython/biopython/blob/master/Bio/Phylo/BaseTree.py

```
def randomized(cls, taxa, branch_length=1.0,
        branch_stdev=None):
"""Create a randomized bifurcating tree given a list
        of taxa.
    :param taxa: Either an integer specifying the number
        of taxa to create (automatically named taxon#),
        or an iterable of taxon names, as strings.
    :returns: a tree of the same type as this class.
"""
```

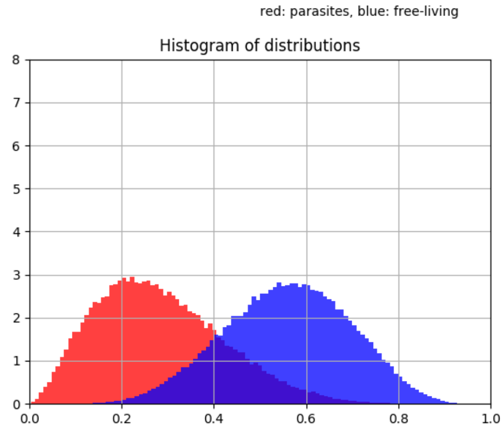
TODO: Zitat von BaseTree und buildTree.py

### 2.2.2 tag tree

At this point we want one fully tagged tree, and one less tagged tree which looks like our real data.
Let's say the first specie (the root node) was free-living (start with a parasite without a host makes no sence). For every transition from a node to his child, we take a random number from the father distribution. We decided that from the biological perspective a beta distribution reflects our transition probabilities best (see Figure 2.1 TODO: ref einfügen ).
For example when our father node was free-living, then we take from the free-living beta distribution. Is the number under the threshold for beeing parasite, we get a change and tag the current node as parasite. Otherwise we tag it as free-living.
With this procedure we traverse through the tree from the root to every leave node. A part of this code you see here:

```
from numpy import random
if father_tag == 0:
  # freeliving_distribution:
  new_random = random.beta(a=A_FL, b=B_FL)
else:
  # parasite_distribution:
```

**Figure 2.1:** 60% Free-living - 40% Parasites




```
  new_random = random.beta(a=A_P, b=B_P)
tag = 0          # -> FL
if new_random < percentage_parasites:
  tag = 1        # -> P
```

---

TODO: Bessere Beschriftung, Plot neu erstellen! U.a. mit threshold We save each tag with the associated node ID in a nodelist, where we can save all information about our nodes, we want.
The real tree has much less information, we have only information from some current species (leave nodes) and TODO: and probably negligible internal nodes .
To simulate our real tree we save for every node an empty placeholde except for some leave nodes. There we save the tags again. The amount of these unknown information is one parameter, which we got from our real tree. Or which we can change to TODO: ... Was hiervon gehört ind Methoden, was schon in Implementierung oder ganz woanders hin?

### 2.2.3 multifurcate tree

Another parameter is the nature and strength of the multifunction of the tree, since we do not have a binary tree in the real case. After several measurements and analyzes, which we explain in TODO: section/chapter x , we decided to use a $1overx$ distribution, where x is the depth of a node. This means, how deaper we are, how less information we have. We traverse through the tree and pick a random number between 0 and 1. If random number is smaller as our limit ($1 over x$), than we forget the node and hang every child to the father node of the current node. TODO: then / than

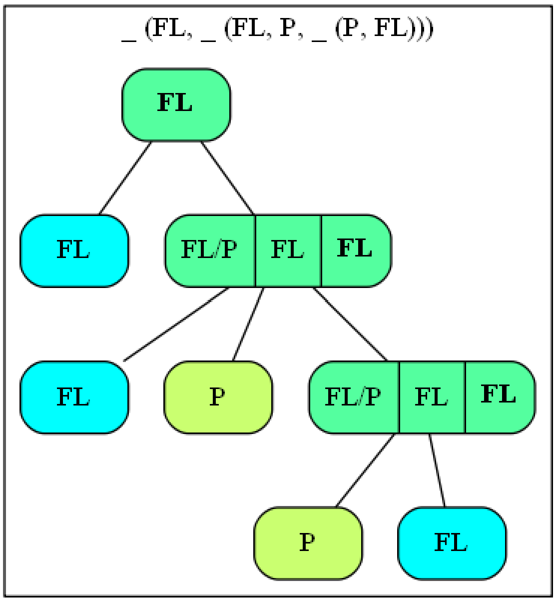
```
from numpy import random
from utilities import Helpers

def get_non_binary_tree(subtree, nodelist):
  i = 0
  while i != len(subtree.clades):
    if subtree.clades[i].is_terminal():                # is leave no
      i += 1
    else:
      element = Helpers.find_element_in_nodelist(subtree.clades[i
      limit = get_limit(element[1])
      new_random = random.uniform()                    # choose if w
      if new_random < limit:                           # or new_rand
        subtree.clades += subtree.clades[i].clades     # add childre
        del subtree.clades[i]                          # delete inte
      else:                                            # if we don't
        get_non_binary_tree(subtree.clades[i], nodelist)
# otherwise the children are in the current clade array
        i += 1
  return

  def get_limit(depth):
    limit = 1 - 1 / ((depth + 3) / 4)
    if limit < 0.1:
      limit = 0.1
    return limit
```

Wir lassen das Limit nicht beliebig klein werden, sondern beschränken es auf 0.1.

**Figure 2.2:** bla




### 2.2.4 maximum parsimony algorithms

#### Fitch maximum parsimony

Described from [COO98] - implemented for multifurcating trees

#### Sankoff

#### my Algorithm

## 2.3 real data analysis

- Import tree
- Import interactions
- run castor algorithm / and others?

- interprete results (leave one out)

## 2.4 Implementation

# 3 Results

# 4 Discussion

Wie gut ist der randomisiert erstellte Baum?
Wie gut kommt unsere Simulation an die echte Datenlage heran.
Fehlerqoute der Daten an sich?
Wie gut ist unsere Datenlage? 3 mio knoten, 1.8 named species (leaf nodes), 200.000 leaf nodes mit Information.
Simulation von subtrees
Welche Teile des Baumes sind gut, an welchen muss noch viel geforscht werden.
Wieviele Origins haben wir gefunden, was bedeutet diese Zahl?

Parameter der Simulation:

- Wie ist die Verteilung der vergessenen internen Knoten? Zum Wurzelknoten hin mehr vergessen?
- Wie sehen die übergangswahrscheinlichkeiten aus von P->FL und andersherum?
- Verteilung Parasiten zu Freilebend zu keine Information

Selecting of the 'right' / best Distribution

# Bibliography


[Fit71] Fitch, Walter M.: Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology. In: *Systematic Biology* 20 (1971), Nr. 4, 406-416. `http://dx.doi.org/10.1093/sysbio/20.4.406`. – DOI 10.1093/sysbio/20.4.406

[HSA+15] Hinchliff, Cody E. ; Smith, Stephen A. ; Allman, James F. ; Burleigh, J. G. ; Chaudhary, Ruchi ; Coghill, Lyndon M. ; Crandall, Keith A. ; Deng, Jiabin ; Drew, Bryan T. ; Gazis, Romina ; Gude, Karl ; Hibbett, David S. ; Katz, Laura A. ; Laughinghouse, H. D. ; McTavish, Emily J. ; Midford, Peter E. ; Owen, Christopher L. ; Ree, Richard H. ; Rees, Jonathan A. ; Soltis, Douglas E. ; Williams, Tiffani ; Cranston, Karen A.: Synthesis of phylogeny and taxonomy into a comprehensive tree of life. In: *Proceedings of the National Academy of Sciences* 112 (2015), Nr. 41, 12764-12769. `http://dx.doi.org/10.1073/pnas.1423041112`. – DOI 10.1073/pnas.1423041112

[SM87] Swofford, David L. ; Maddison, Wayne P.: Reconstructing ancestral character states under Wagner parsimony. In: *Mathematical Biosciences* 87 (1987), Nr. 2, 199 - 229. `http://dx.doi.org/https://doi.org/10.1016/0025-5564(87)90074-5`. – DOI https://doi.org/10.1016/0025–5564(87)90074–5. – ISSN 0025–5564