**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 leafs, 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 the other a simulation of this to proof the credibility of our methods.
The resulting procedure is as follows:

i) Get the real tree and real data for the leaf nodes.

ii) Get metadata of these for a realisitc the simulation.

iii) Build and run the simulation.

iv) Evaluation of parameters for the simulation and the real problem.

v) Run the resulted algorithm on the original data.

vi) Evalute and interprete results. -> Origins etc...

## 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

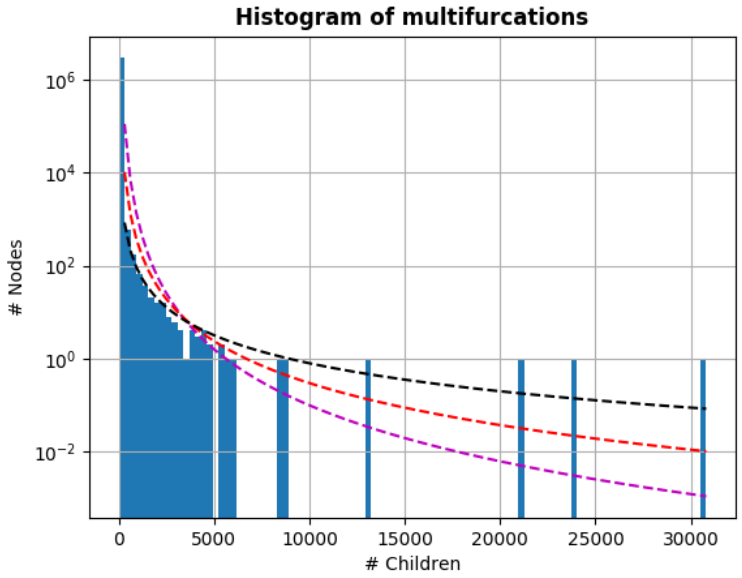
### 2.1.1 countings

Some simple countings...
nr of parasites, free-living and unknown nodes ... from whole tree, subtrees...

### 2.1.2 Multifurcation

One property of the tree is its ridge of multifurcation. For this we collected for every node its number of children (degree $-1$), and plottet this in an histogram, how much nodes we have for every degree. Then we tried to find a function which best discribes this behavior. In figure 2.1 is this plot shown with three functions, which came TODO: close(st) to this goal: $\frac{1000000000000000}{x^4}$ (magenta), $\frac{300000000000}{x^3}$ (red) and $\frac{80000000}{x^2}$ (black).
From the biology we know, that this multifurcation is very clouded. Some parts of the tree

**Figure 2.1:** Histogram of multifurcations and fitting functions




are well known and others have not been very interesting for the research so far. So we also wanted to know how the multifurcations behave in some interesting subtrees.
TODO: subtrees... there is some old stuff...

TODO: other subsections here...

## 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_leaf nodes)
```

From the BaseTree class:

```
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.
"""
```

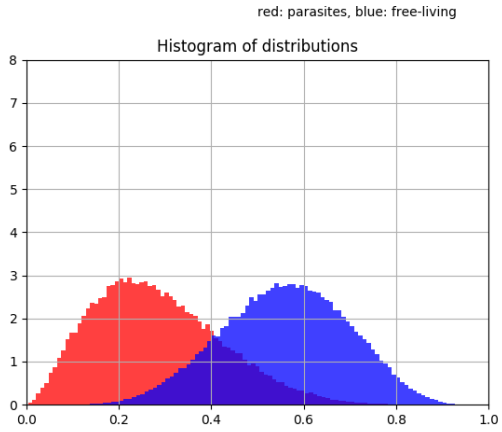
TODO: Zitat von BaseTree und buildTree.py

### 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

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

**Figure 2.2:** 60% Free-living - 40% Parasites




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 leaf 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:
  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 (leaf nodes) and TODO: and probably negligible internal nodes .
To simulate our real tree we save for every node an empty placeholde except for some leaf 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?

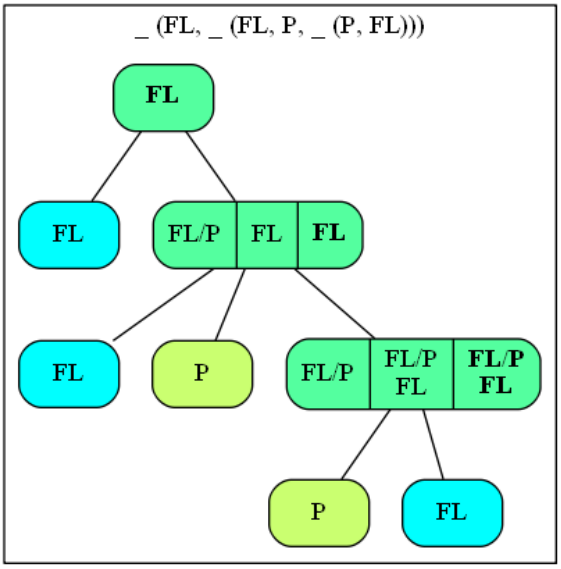
### 2.2.3 multifurcate tree

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 $1 over x$ 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 leaf noc
      i += 1
    else:
      element = Helpers.find_element_in_nodelist(subtree.clades[i
      limit = get_limit(element[1])
      new_random = random.uniform()                    # choose if v
      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
```

**Figure 2.3:** bla




```
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.

### 2.2.4 maximum parsimony algorithms

## Fitch maximum parsimony

Described from [COO98] + others ... - implemented for multifurcating trees

Fitch algorithm for binary trees:

Der Baum hat die folgende Struktur: Alle inneren Knoten sind leer. In den Blattknoten befindet sich entweder das Tag FL oder P, oder deren Vereinigung, wenn es sich um einen unknown node handelt.

Der Fitch Algorithmus ist aufgeteilt in drei Schritte, in welchen man jeweils durch den Baum traversiert. Schritt 1 beginnt von de Blättern aus, da sich dort zu Beginn die einzige Information befindet. Für jeden Knoten gilt, wenn seine Kinder schon Information enthalten, dann bilde die Schnittmenge der Tags und schreibe diese als Information in den aktuellen Knoten. Ist die Schnittmenge leer, dann schreibe die Vereinigung aller möglichen Tags in den Knoten. Für alle Kinder, die noch keine Information haben, führe diesen Schritt erst für diese aus. Schritt zwei geht von den Kindern der Wurzel bis zu den Vätern der Blätter. Jeder Knoten bekommt einen zweiten Tag, der sich aus der Vereinigung des Tags des Vaterknoten und der Geschwisterknoten zusammensetzt. Ist diese leer, bekommt der Knoten wieder die Vereinigung aller Tags, also $\{FL, P\}$ als Tag.

Hier gibt es einige Möglichkeiten, wie dieser Schritt genau aussieht. 1. Version: Es wird nur der erste Tag vom Vaterknoten genutzt. Außerdem wird von den Geschwisterknoten zuerst der Schnitt gebildet, und danach vom Ergebnis nochmal mit dem Vaterknoten zusammen. (Immer wenn der Schnitt leer ist, ist das Ergebnis die Vereinigung aller Tags, also $\{FL, P\}$. Auch im folgenden...) 2. Version: Es wird nur der erste Tag vom Vaterknoten genutzt. Er wird zusammen mit den Geschwistertags genommen und direkt ein Schnitt aller Mengen gebildet. 3. Version: Es werden alle vorherigen Tags vom Vater genutzt und von diesen ein Schnitt gebildet. Das selbe gilt für die Geschwistertags. Und dann wird ein dritter Schnitt zwischend en Ergebnissen gebildet. 4. Version: Es werden alle Tags genutzt und direkt in einem Schnitt zusammengenommen.

Der Finale Schritt traversiert nochmal über den Baum und Bildet aus den zwei Tags pro Knoten einen finalen Tag, indem wieder der Schnitt der beiden Tags das Ergebnis ist.

Ich habe diese Versionen mit 100 Bäumen mit 10000 Blattknoten und der Verteilung 60% FL zu 40% P simuliert. Bei 90 % unbekannten Knoten lag Version 1 zu 89.67 %, Version 2 zu 89.67 %, Version 3 zu 90.72 % und Version 4 zu 90.74 % richtig.

How to extend Fitch for multifunction?:

## Sankoff

Maximum parsimony algorithm from Sankoff implemented in the R package castor.

**Figure 2.4:** bla

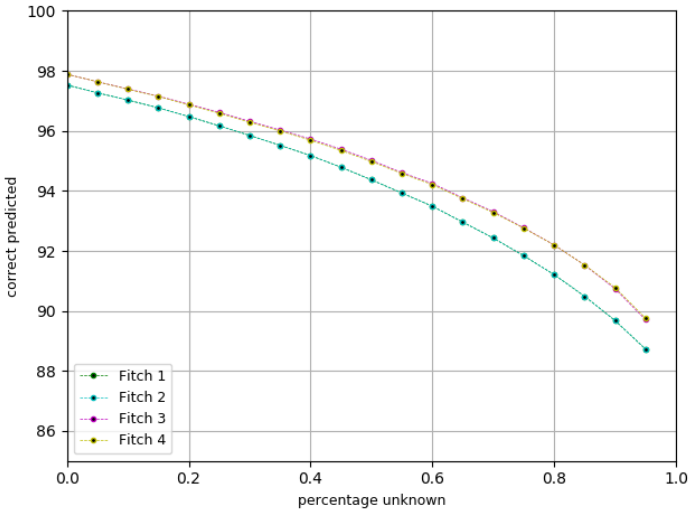
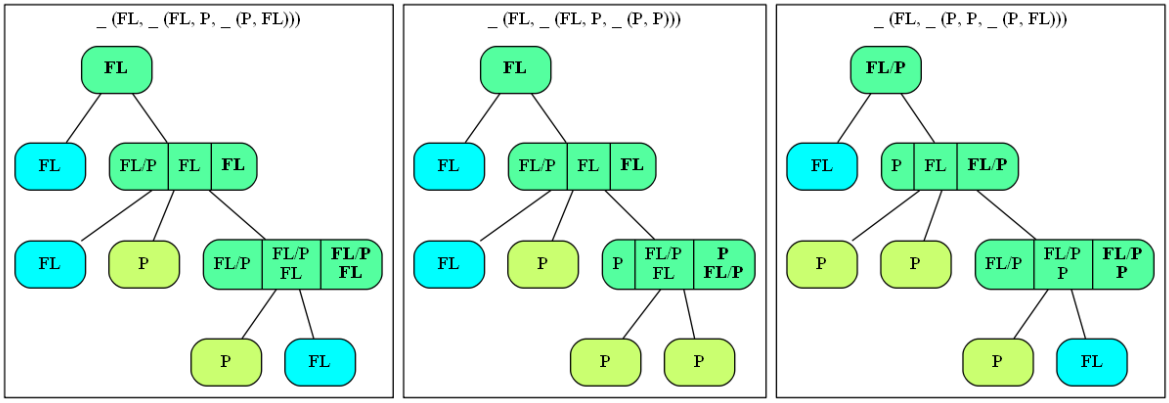



**Figure 2.5:** bla



### my Algorithm

## 2.3 real data analysis

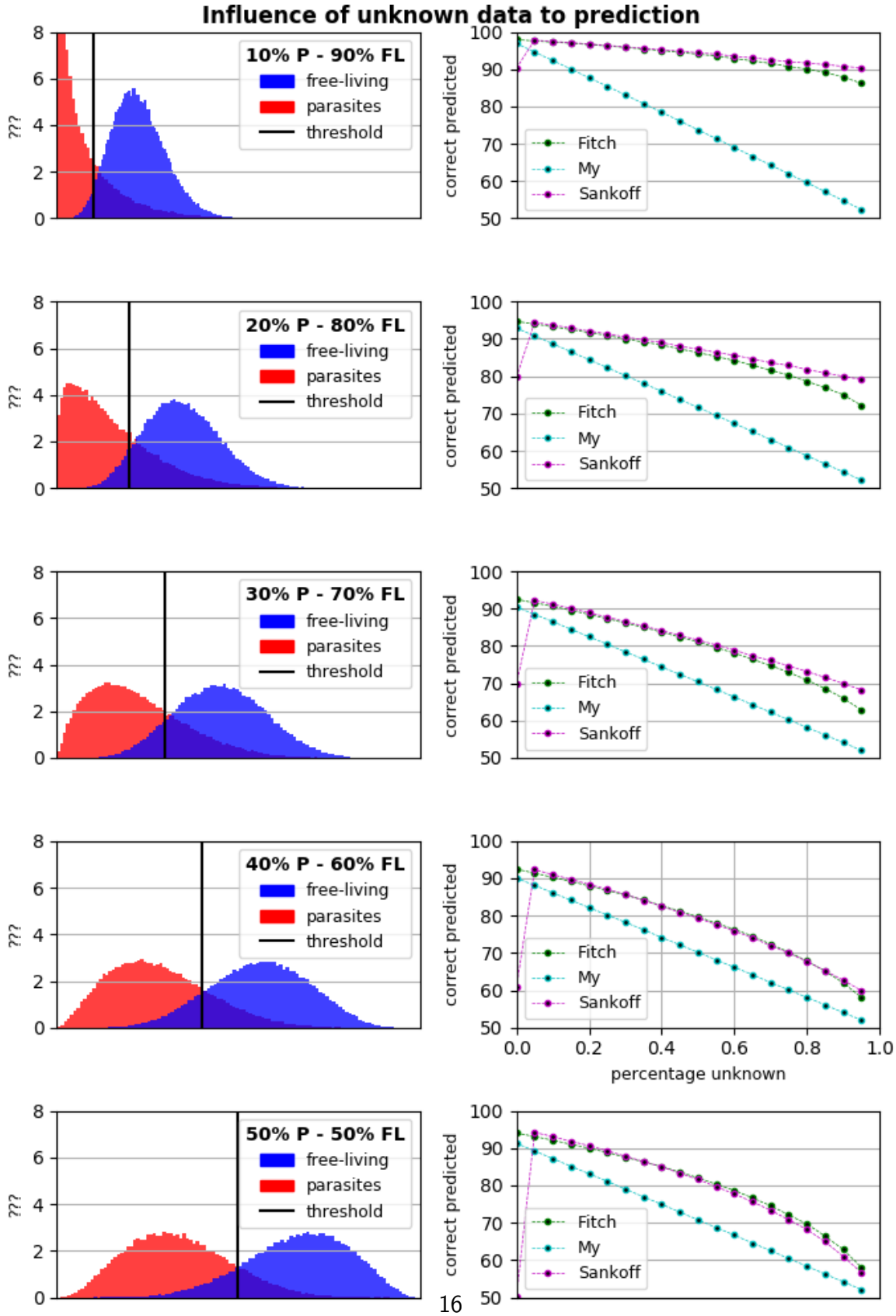
- Import tree
- Import interactions
- run castor algorithm / and others?
- interprete results (leave one out)

## 2.4 Implementation

# 3 Results

## 3.1 Influence of different parameters

**Figure 3.1:** Influence of unknown data to prediction



# 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