Preliminaries for Distributed Natural Computing Inspired by the Slime Mold Physarum Polycephalum

Dissertation

A dissertation submitted towards the degree
Doctor of Natural Sciences (Dr. rer. nat.)
of the Faculties of
Mathematics and Computer Science
of Saarland University

submitted by:

Michael Dirnberger

Saarbrücken, February 2017

Day of doctoral examination: 31st July, 2017

Dean of Faculty: Prof. Dr. Frank-Olaf Schreyer

Head of examination committee: Prof. Dr. Hans-Peter Seidel

Examiner 1: Prof. Dr. Dr. h.c. mult. Kurt Mehlhorn

Examiner 2: Prof. Dr. Martin Grube, Karl-Franzens University of Graz

Examiner 3: Prof. Dr. Hans-Günther Döbereiner, University of Bremen

Academic staff: Dr. Karl Bringmann

Abstract

This doctoral thesis aims towards distributed natural computing inspired by the slime mold *Physarum polycephalum*. The vein networks formed by this organism presumably support efficient transport of protoplasmic fluid. Devising models which capture the natural efficiency of the organism and form a suitable basis for the development of natural computing algorithms is an interesting and challenging goal.

We start working towards this goal by designing and executing wet-lab experiments geared towards producing a large number of images of the vein networks of *P. polycephalum*. Next, we turn the depicted vein networks into graphs using our own custom software called Nefi. This enables a detailed numerical study, yielding a catalogue of characterizing observables spanning a wide array of different graph properties. To share our results and data, i.e. raw experimental data, graphs and analysis results, we introduce a dedicated repository revolving around slime mold data, the Smgr. The purpose of this repository is to promote data reuse and to foster a practice of increased data sharing.

Finally we present a model based on interacting electronic circuits including current controlled voltage sources, which mimics the emergent flow patterns observed in live *P. polycephalum*. The model is simple, distributed and robust to changes in the underlying network topology. Thus it constitutes a promising basis for the development of distributed natural computing algorithms.

Zusammenfassung

Diese Dissertation dient als Vorarbeit für den Entwurf von verteilten Algorithmen, inspiriert durch den Schleimpilz *Physarum polycephalum*. Es wird vermutet, dass die Venen-Netze dieses Organismus den effizienten Transport von protoplasmischer Flüssigkeit ermöglichen. Die Herleitung von Modellen, welche sowohl die natürliche Effizienz des Organismus widerspiegeln, als auch eine geeignete Basis für den Entwurf von Algorithmen bieten, gilt weiterhin als schwierig.

Wir nähern uns diesem Ziel mittels Laborversuchen zur Produktion von zahlreichen Abbildungen von Venen-Netzwerken. Weiters führen wir die abgebildeten Netze in Graphen über. Hierfür verwenden wir unsere eigene Software, genannt Nefi. Diese ermöglicht eine numerische Studie der Graphen, welche einen Katalog von charakteristischen Grapheigenschaften liefert. Um die gewonnenen Erkenntnisse und Daten zu teilen, führen wir ein spezialisiertes Daten-Repository ein, genannt Smgr. Hiermit begünstigen wir die Wiederverwendung von Daten und fördern das Teilen derselben.

Abschließend präsentieren wir ein Modell, basierend auf elektrischen Elementen, insbesondere stromabhängigen Spannungsquellen, welches die Flüsse von *P. polycephalum* nachahmt. Das Modell ist simpel, verteilt und robust gegenüber topologischen Änderungen. Aus diesen Gründen stellt es eine vielversprechende Basis für den Entwurf von verteilten Algorithmen dar.

Acknowledgements

As I start writing these lines intended to thank my *Doktorvater* Kurt Mehlhorn, I struggle to find the words capturing the depth of my gratitude towards him. Kurt, if you read this, please know that I will never forget the role you played in my life.

To me (and to many others) you are, and always will be, a true role model worthy of admiration. You are a just and caring person, an enormous scientist, an unparalleled teacher, a wise mentor and an exemplary leader. I admire you for your humble demeanour, your perfect diligence and your unwavering purpose. Your down-to-earth attitude and your seemingly untiring willingness to help others wherever you can uplifts and inspires everyone around you.

I am grateful for the privilege of being your student. I feel extremely humbled by your unending support and constant encouragement. I am grateful that you bravely trusted me to find myself through my own projects, even if they were risky and out of the ordinary. At times, I didn't feel up to the task myself. Frequently I struggled. Nevertheless, whatever the problem seemed to be, you never failed to help me find the missing pieces. In truth, your generous advice quietly enabled me every step of the way. I will carry with me the numerous lessons you taught me about science, about people and about life. For this I will always be indebted to you. In fact, I hope that one day, I will be able to do for someone, what you have done for me.

In addition to Kurt, there are many others that deserve my gratitude. Without them this thesis would not have been possible either. This is particularly true for my many outstanding co-authors. True to the nature of modern science, most of the results presented in this thesis are the product of various collaborations and many hours of lively discussions amongst colleagues. I would be remiss If I didn't mention Adrian Neumann and Matthias Függer who shared my ideas and supported me actively during the earlier and the later stages of my doctorate respectively. I estimate that roughly two thirds of my sanity have been preserved by interacting with those two individuals and of course, by procrastinating with my office mates.

My final thanks are dedicated to my dear Sarah. I know that the "last hurrah" of completing this thesis was as least as hard on you as it was on me. Thank you for being patient and for overlooking, without complaint, the many late-night sessions it required. Thank you also for always being convinced that my stuff was solid, and that I was solid, and that I would pull through in the end no matter what.

Contents

Contents vii

List of Figures xi

List of Tables xiii

List of Code Listings xv

1. Introduction 1
- 1.1. The Slime Mold Physarum Polycephalum . . . 1
  - 1.1.1. Life of Slime . . . 2
  - 1.1.2. The Plasmodium of P. Polycephalum . . . 5
  - 1.1.3. Overview of Research Focused on P. Polycephalum . . . 7
- 1.2. Natural Computing . . . 8
  - 1.2.1. Computing Inspired by Nature . . . 9
  - 1.2.2. Synthesis of Nature by Means of Computing . . . 11
  - 1.2.3. Computing with Natural Materials . . . 12
- 1.3. Natural Computing with *P. polycephalum* . . . 13
  - 1.3.1. Key Experiments and Observations . . . 13
  - 1.3.2. Natural Computing Approaches . . . 21
- 1.4. Motivation and Outline . . . 31

2. Network Extraction From Images 35
- 2.1. Introduction . . . 35
- 2.2. Network Extraction From Images . . . 37
  - 2.2.1. Preprocessing Collection . . . 41
  - 2.2.2. Segmentation Collection . . . 42
  - 2.2.3. Graph Detection Collection . . . 42
  - 2.2.4. Graph Filter Collection . . . 44
- 2.3. Evaluation . . . 45
  - 2.3.1. Using a Graph Similarity Measure to Evaluate NEFI . . . 45
  - 2.3.2. Definition of the Similarity Measure . . . 46
  - 2.3.3. Evaluation of NEFI's Output . . . 49
  - 2.3.4. Evaluation of Speed Performance . . . 52
- 2.4. Limitations of NEFI . . . 53
- 2.5. Synergies With Other Software . . . 54
  - 2.5.1. Analysis of Graphs . . . 54

2.5.2. Third-party Segmentation Software . . . 54
2.6. Where to Download NEFI and how to Contribute . . . 54
2.7. Discussion . . . 55
2.8. Acknowledgments . . . 55

3. Slime Mold Graph Repository **57**
3.1. Introduction . . . 57
3.2. Repository Concept and Benefits . . . 59
3.3. The KIST Europe Data Set . . . 60
3.3.1. Node Tracking . . . 65
3.4. Sample Usage of the KIST Europe Data Set . . . 69
3.5. Discussion . . . 72
3.6. Acknowledgments . . . 73

4. Network Analysis **75**
4.1. Introduction . . . 75
4.2. Methods . . . 77
4.2.1. Experimental data . . . 77
4.2.2. Graph Representation . . . 77
4.2.3. Statistical Methods . . . 79
4.3. Results . . . 79
4.3.1. Path Properties . . . 79
4.3.2. Face Properties . . . 86
4.3.3. Cut Properties . . . 94
4.3.4. Percolation . . . 97
4.4. Discussion . . . 101
4.5. Acknowledgments . . . 103

5. Modeling Flows **105**
5.1. Introduction . . . 105
5.2. Overview of Modeling Approaches . . . 107
5.3. Continuous Model . . . 110
5.3.1. Putting the Model on a Graph . . . 111
5.4. Basic Properties of the Continuous Model . . . 113
5.5. Discrete Model . . . 116
5.6. Basic Properties of the Discrete Model . . . 117
5.7. Preliminary Simulation Results . . . 119
5.7.1. Cycle of Physarum Elements . . . 121
5.7.2. Diamond of Physarum Elements . . . 123
5.7.3. Paths of Physarum Elements . . . 125
5.7.4. Trees of Physarum Elements . . . 127
5.7.5. Two Linked Cycles of Physarum Elements . . . 127
5.7.6. Physarum Elements and Changing Topology . . . 131
5.8. Discussion . . . 133

6. Summary **135**

A. Guide to Using NEFI **139**
A.1. Properties of Ideal and Non-ideal Images 139
A.2. Dealing With Challenging Images 143

B. Details of Data Acquisition **145**
B.1. Experiments 145
B.2. Graph Extraction 146
B.3. Continued Production of Sclerotia 148

C. Supplementary Figures **151**
C.1. Goodness-of-fit Plots for Section 4.3 151
C.2. Addititonal Illustrations for Section 4.3.3 156

D. Code Listings **159**

Bibliography **177**

List of Figures

1.1. *P. polycephalum* exploring various environments 2
1.2. Life cycle of *P. polycephalum* 4
1.3. Details of apical zone and supporting network 5
1.4. Schematic drawing of the plasmodium of *P. polycephalum* 6
1.5. The goals of natural computing. 9
1.6. Oscillator experiment - Setup 14
1.7. Oscillator experiment - Thickness oscillations 15
1.8. Classic maze experiment with *P. polycephalum* 17
1.9. Network of food sources by *P. polycephalum* 19
1.10. *P. polycephalum* avoiding illumination 20
1.11. Multi-agent *P. polycephalum* - Agent schematic 24
1.12. Multi-agent *P. polycephalum* - Collective behavior of agents 25
1.13. Multi-agent *P. polycephalum* - Evolution of agents 25
1.14. Multi-agent *P. polycephalum* - Approximate MST 26
1.15. Tokyo railway experiment 28
1.16. *P. polycephalum* neurons - Setup 29
1.17. *P. polycephalum* neurons - State transitions 30

2.1. Nefi's pipeline concept 38
2.2. Nefi's pipeline executed 39
2.3. Nefi's graphical user interface 39
2.4. Nefi's output - *Physarum polycephalum* 40
2.5. Nefi's output - *Ajax junius* 41

3.1. Setup for wetlab experiments 62
3.2. Crumbs of *P. polycephalum* sclerotia forming an inoculation line . . 62
3.3. The apical zone advances 62
3.4. The onset of network coarsening 63
3.5. A complex network of veins within a region of interest 63
3.6. Graph extracted from a sample region of interest 63
3.7. Schematic description of node tracking 66
3.8. Demo - Computing an observable across an entire series of graphs . 69
3.9. Demo - Setup for tracking individual edges 71
3.10. Demo - Results for tracking individual edges 72

4.1. Graph representation of a *P. polycephalum* graph 78
4.2. Path length distribution 80

4.3. Path length distribution - Fit parameters . . . 81
4.4. Path length distribution - Fit parameter densities . . . 82
4.5. Path width distribution . . . 83
4.6. Path width distribution - Fit parameters . . . 84
4.7. Path width distribution - Fit parameter densities . . . 85
4.8. Face degree distribution . . . 87
4.9. Face degree distribution - Fit parameter densities . . . 87
4.10. Face area distribution . . . 88
4.11. Face area distribution - Fit parameter densities . . . 89
4.12. Face degree weighted by face area . . . 90
4.13. Face circumference distribution . . . 90
4.14. Face circumference distribution - Fit parameter densities . . . 91
4.15. Face roundness distribution . . . 93
4.16. Average face roundness per face type . . . 93
4.17. Cut properties . . . 96
4.18. Percolation properties . . . 98
4.19. Critical percolation thresholds . . . 99

5.1. The basic *Physarum* element . . . 110
5.2. Definitions for current controlled voltage sources . . . 112
5.3. A node with 3 *Physarum* elements attached . . . 113
5.4. Simulation - Cycles . . . 122
5.5. Simulation - Diamond or Wheatstone graph . . . 124
5.6. Simulation - Paths . . . 126
5.7. Simulation - Trees . . . 128
5.8. Simulation - Dumbbell graph . . . 130
5.9. Simulation - Changing topology . . . 132

A.1. NEFI's caveats - Reflections . . . 140
A.2. NEFI's caveats - Gradients . . . 140
A.3. NEFI's caveats - Non-network objects . . . 141
A.4. NEFI's caveats - Non-network objects . . . 141
A.5. NEFI's strengths - An ideal image of *P. polycephalum* . . . 142
A.6. NEFI's strengths - An ideal image of *A. junius* . . . 142

C.1. Goodness-of-fit plots - Path length . . . 151
C.2. Goodness-of-fit plots - Path width . . . 152
C.3. Goodness-of-fit plots - Face degree . . . 153
C.4. Goodness-of-fit plots - Face area . . . 154
C.5. Goodness-of-fit plots - Face circumference . . . 155
C.6. Horizontal and vertical cuts illustrated . . . 156
C.7. Detail of the apical zone . . . 157

List of Tables

1.1. Computing inspired by *P. polycephalum* 23
1.2. Computing by synthesis of *P. polycephalum* 26
1.3. Computing with live *P. polycephalum* 30

2.1. Nefi's evaluation - Ideal images 50
2.2. Nefi's evaluation - Edges with varying brightness 51
2.3. Nefi's evaluation - Images with background color gradient 52
2.4. Nefi's evaluation - Images with a blur 52
2.5. Nefi's evaluation - Pipeline timings 53

5.1. Hydraulic analogy 109
5.2. Simulation - Initial values 120

List of Code Listings

D.1. Model implementation - Required libraries 159
D.2. Model implementation - Core of the simulation 159
D.3. Model implementation - Update node voltages 160
D.4. Model implementation - Update capacitor voltages 161
D.5. Model implementation - Update current controlled voltage sources . 162

1 Introduction

In this chapter we give an introduction to the basic topics and concepts this thesis is concerned with. We start by familiarizing the reader with the humble yet interesting organisms that is the slime mold *Physarum polycephalum*. After a short excursion to the realm of biology, we proceed by introducing the highly interdisciplinary field of Natural Computing where we highlight various successful algorithms and the natural phenomena that served as their source of inspiration. After illustrating how one may look to nature to drive the development of new algorithms, we discuss how *P. polycephalum* was discovered to be relevant as a medium for Natural Computing. In this context we survey break-through experimental findings as well as the most important Natural Computing approaches realized so far. The chapter closes by giving the motivation of this thesis, namely to pave the way towards novel approaches to distributed Natural Computing with *Physarum polycephalum*.

1.1. The Slime Mold Physarum Polycephalum

Physarum polycephalum is an acellular slime mold belonging to the *myxomycetes*.[1] It is native to the forests of France, Italy, Spain, Romania, North-, Middle- and South America as well as China, Nepal, Southeast Asia and Japan. Its striking bright yellow renders the organism very noticeable, see Figure 1.1a. In the recent past *P. polycephalum* has become increasingly at home in research laboratories and schools across the globe where its extraordinary properties fascinate scientists and students alike, see Figure 1.1b.

In the following we give a concise introduction to *Physarum polycephalum*, discussing its life cycle and scientific importance. We emphasize the plasmodium stage in particular, which is of key importance for all natural computing applications discussed in Section 1.3. We summarize selected material from several sources which may be consulted for more detailed surveys [68], [81], [109], [129], [151], [162]. Our exposition closely follows the structure of [129].

[1] Myxomycetes, myxogastria or myxogastrea are all synonyms and denote a grouping of slime molds containing 5 orders, 62 genera and 888 species [27].

(a) **(b)**

Figure 1.1.: Today striking *P. polycephalum* seems equally at home in the forest (a) and the in the laboratory (b). Figures courtesy of Prof. T. Ueda of Hokkaido University.

1.1.1. Life of Slime

The life cycle of *P. polycephalum* starts with its spores which are propagated in a predominately airborne fashion. After an incubation period of a few days, spores begin to germinate given favorable conditions. In the process the walls of the spores break open to release haploid protoplasmic bodies of 12 µm to 15 µm in diameter. After a short period of quiescence these so-called myxoamoebae become active and start growing and multiplying much like other soil amoebae.

Two reversible processes illustrate the remarkable adaptability of *P. polycephalum* myxoamoebae. First, they have the ability to quickly grow one or two flagellae, i.e. change into myxoflagellates. This enables them to better navigate moist environments such as water films. If moist conditions are followed by dry ones, the organism can change back to its non-flagellate form. Second, both types of myxoamoebae may form dormant micro cysts capable of enduring adverse conditions such as extreme dryness or strong illumination. As soon as conditions become favorable again, myxoamoebae escape their cysts and are ready to continue the life cycle.

In the next stage of the life cycle of *P. polycephalum*, pairwise sexual or heterothallic fusion between two haploid myxoamoebae are observed. This leads to the irreversible formation of a diploid zygote. It is also possible that a single myxoamoeba changes directly into a haploid zygote in an apogamic or selfing fashion.

Both types of zygotes have in common that from now on, nuclear division happens synchronously every 8 h to 10 h without cell devision. This dramatic change signals the onset of a peculiar cellular organization, the so-called *plasmodium*. In this stage *P. polycephalum* lives as an macroscopic brightly yellow mass of protoplasm consisting of up to millions of nuclei contained in a singular cell. Remarkably, the plasmodium stage is unique to myxomycetes and unparalleled throughout all of nature.

In its plasmodium stage *P. polycephalum* is acting as an undifferentiated macroscopic creature, capable of sensing food sources, migrating towards them and feeding on them by means of phagozytosis. Typical food sources encountered by *P. polycephalum* in the field include bacteria, amoebae, algae, common molds and various organic materials. It is also known to feed on spores, hyphen or fruiting bodies of fungi. In the lab *P. polycephalum* can sustain itself on substrates containing solute nutrients. Under continued food intake, the plasmodium can grow to cover large areas of up to several dm^2. Under unfavorable conditions the plasmodium can change into many multi-nucleated dormant macrozysts, forming a crust of dried slime, the so-called *sclerotium*. *P. polycephalum* may reverse back to its plasmodium form in better conditions even after prolonged periods of lying dormant.

Towards the end of the life cycle, triggered by illumination, the plasmodium seeks out dry and preferably elevated locations to begin the irreversible process of sporulation. In a synchronous differentiation process *P. polycephalum* forms colonies of 1 mm to 2 mm tall fruiting bodies. These so-called sporangia are numerous and come with a stem and a head each. Their appearance inspired its species designation *Physarum polycephalum*, the multi-headed. Within the fruiting bodies, spores are maturing which are responsible for species propagation. After maturing, fruiting bodies break open allowing the contained spores to be dispersed by the wind. Thus the life cycle of *P. polycephalum*, given suitable conditions, may start anew.

It is fascinating how this multi-potent development system exists in extremely different forms of live while controlled by one and the same genome in a unique temporal sequence. From spores to amoeba, on to plasmodium and finally to fruiting bodies. From there back to spores and again into more amoeba. The complete life cycle of *Physarum polycephalum* is illustrated in Figure 1.2.

Next we discuss the plasmodium stage of *P. polycephalum* in more detail as it is most relevant to this thesis.

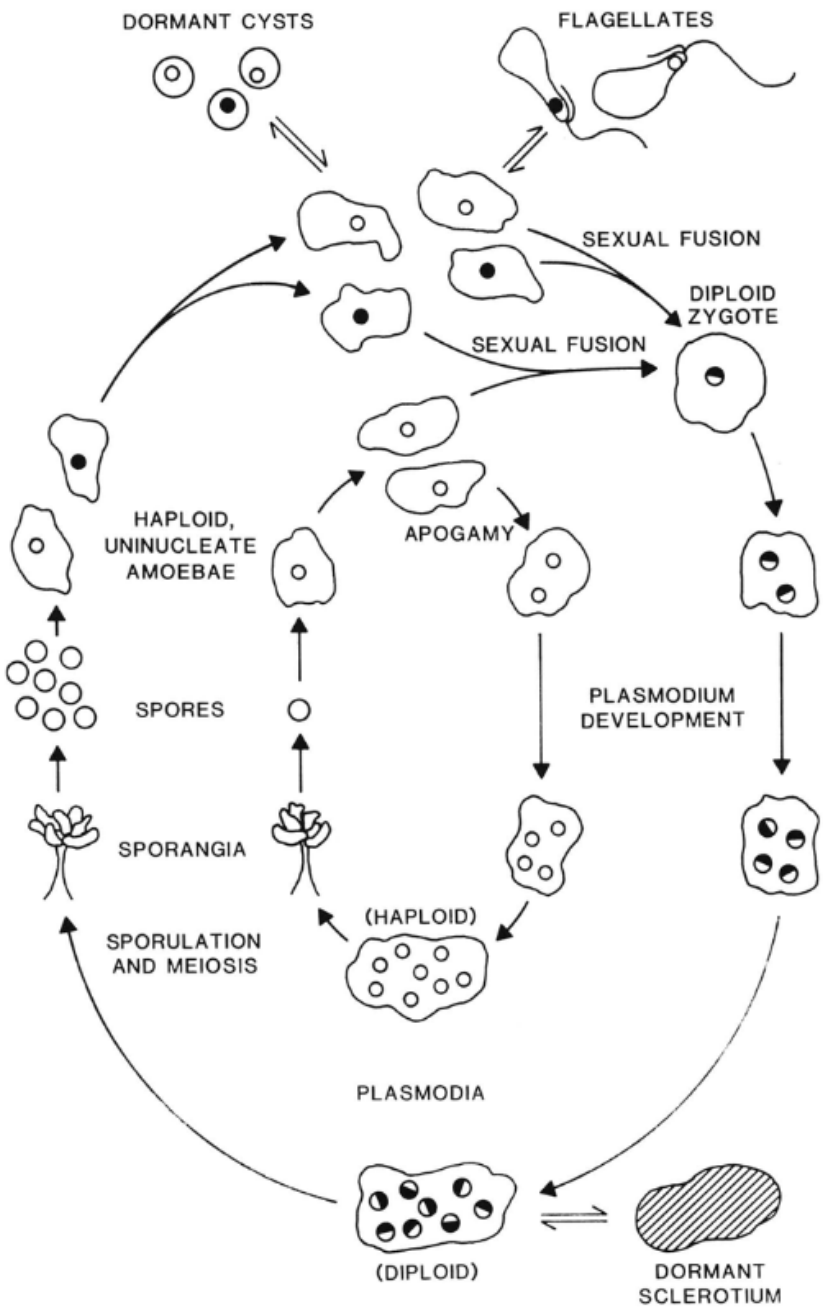
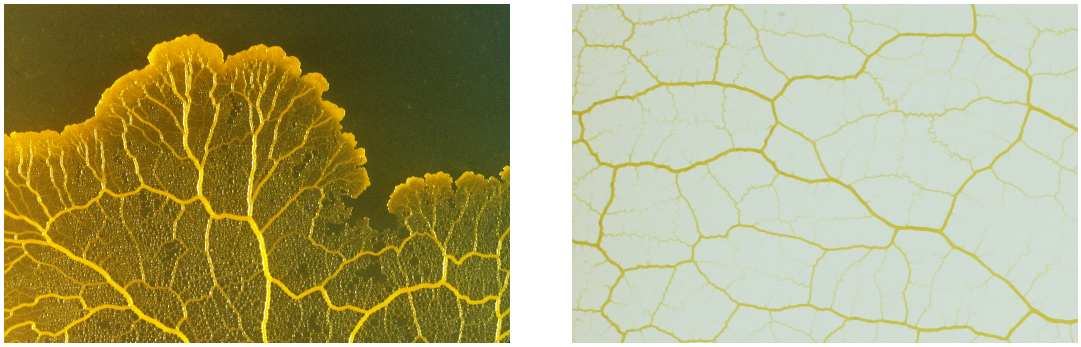


Figure 1.2.: The complete life cycle of *Physarum polycephalum*. Reprint from [151].

1.1.2. The Plasmodium of P. Polycephalum

The typical plasmodium of *P. polycephalum* takes the shape of an extended sheet-like structure capable of moving to explore unvisited territory in search of food sources. Figure 1.1b illustrates that in this stage of its life cycle, the organism consists of a complex vein network (bottom right area of image) which connects to the boundary of the organism, its apical zone or growing front (arcing from bottom left to top right). Figure 1.3a and Figure 1.3b show details of the apical zone and the vein network respectively. The overall shape of the network is highly dynamic and may change drastically in response to changing environmental conditions such as encountered attractants or repellents. This extraordinary functional plasticity allows *P. polycephalum* to navigate its environment successfully.



(a) **(b)**

Figure 1.3.: (a) Detailed image of the apical zone being supported by thick veins. (b) A larger example of the complex vein network. Note the large cycles of thick veins. Smaller veins provide additional connections.

The veins themselves are approximately cylindric with vein walls consisting of a thick connected mesh of Actin and Myosin fibers. They connect to form a complex planar vein network. Within the veins protoplasmic fluid, or protoplasm for short, can freely flow back and forth. Protoplasmic flow transports cell nuclei, nutrients and various other relevant factors. Figure 1.4 shows a schematic drawing of a macro plasmodium.

The protoplasmic flow itself is driven by periodic cross-sectional contractions of the actin-myosin mesh forming the walls of the veins. The contractions cause a peristaltic pumping effect inducing protoplasmic streaming and net material transport. Resulting peristaltic contraction waves can be observed across the entire network inducing complex flow patterns including periodic flow arrests and reversals. Every $(50 \pm 5)\,\mathrm{s}$ the velocity of the protoplasmic flow streaming through a vein decays smoothly until the flow completely arrests. After one or two seconds of standstill, flow velocity quickly accelerates back to normal and the cycle proceeds. Interestingly, after most but not all arrests, the direction of flow is reversed after the flow picks

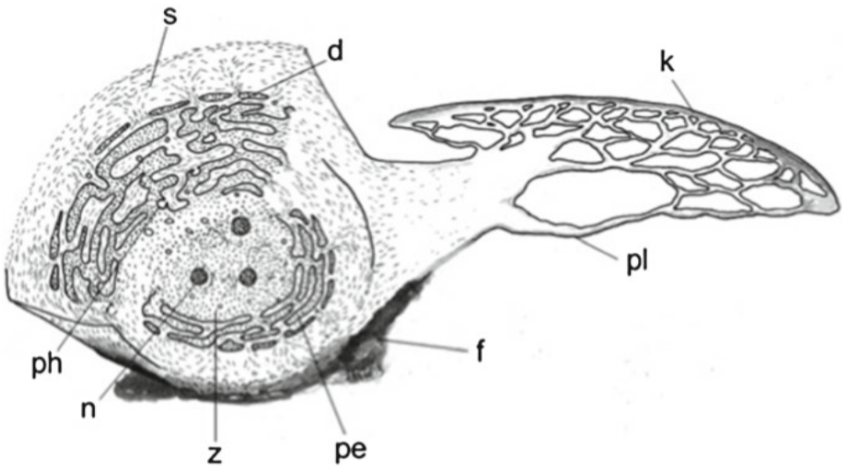


Figure 1.4.: Schematic drawing of slime mold plasmodium with facing side sectioned and upper left section enlarged. `k` moving front, `pl` trailing plasmodial strand, `f` deposited material, `s` slime; `d` vacuole with deposit; `ph` phagocytosis vesicle; `n` nucleus; `z` central plasma; `pe` peripheric membrane stacks. Peristaltic contractions occur in the peripheric plasma, while the central plasma is subject to shuttle streaming. Hand-drawing by Prof. M. Grube, Karl-Franzens University of Graz. Reprinted with permission from [68].

up again. Protoplasmic flow exhibits enormous flow speeds of up to 1000 µm/s.[2] It is believed that the interplay of network topology, peristaltic pumping and complex flow patterns with high flow velocity facilitates efficient transport of nutrients and signaling molecules across the entire organism.

In addition to enabling fluid transport, flow patterns also cause periodic pressure waves arriving at the apical zones of the organism. Each incoming wave extends the organism boundary by a small amount by pushing forward protoplasm that arrives via the vein network. This process is called shuttle streaming. Since the pressure of the streaming protoplasm is very high, material practically shoots out of the supporting veins, resulting in fan-like growing tips which can be seen along the boundary of the organism, see Figure 1.3a. As the advancing growing front leaves behind a network of veins, continued support for expansion is ensured.

Given abundant food supply, *P. polycephalum* rapidly advances a coherent and dense apical zone exploring the available space. The growing front proceeds as one large unit following an exploration strategy aptly termed *phalanx*, see Figure 1.1b or Figure 1.3a. If nutrients are scarce, however, a different strategy is employed and *P. polycephalum* tends to grow several separate distinct growing tips, each advancing their own substantially smaller growing front. This behavior increases the odds of discovering more distant sources of nutrients. Since branches may react individually

[2] Note that 1000 µm = 1 mm. Also note that flow speeds of 2 µm/s to 78 µm/s known for streaming plasma in plants pale in comparison.

to attractants such as food, a more adaptive search is achieved. Once new food sources are secured, the organism can concentrate its movement towards them by reallocating its mass as needed. This strategy has been termed *guerilla*. Figure 1.1a shows a separate growing tip on the right side. The figure also hints at the fact that the plasmodium of *P. polycephalum* can naturally interpolate between the two extreme strategies. The strategies employed by *P. polycephalum* are reminiscent of two well know graph traversal strategies: Breadth-first and Depth-first search. In BSF, the explored region grows uniformly like a wave that expands evenly in all available directions. In contrast to that, DFS chooses one direction in order to explore it to maximum depth first. Only then does it backtrack to resolve other places ignored so far.

Finally, we remark that the plasmodium stage of *P. polycephalum* can be regarded as simple in terms of its biological organization because it lacks any form of brain or nervous system capable of orchestrating complex tasks such as foraging for food. The fact that the organism still displays a level of complex organized behavior sufficient to survive is one of its most fascinating features. Today it is believed that effective global organization emerges from the delicate interplay of local effects such as periodic contractions, topological dynamics and the integration of environmental signals. Shedding light on the details of these dynamics remains one of the major challenges in *P. polycephalum* research.

1.1.3. Overview of Research Focused on P. Polycephalum

P. polycephalum develops exceptionally well when cultured in the lab which makes it an ideal subject for scientific studies. Significant research activity in the beginning of the latter half of the 20th century explored the life cycle of *P. polycephalum* and described the morphology and physiology of its various stages for the first time. Culturing procedures as well as genetic and molecular techniques were developed which allowed a detailed study of its mitotic cycle, cellular motility, differentiation and many other questions of biological interest. Although interest in *P. polycephalum* was at first exclusively fueled by general questions of biology, a wider scientific community soon began to appreciate the value of *P. polycephalum* as a multi-potent model system that could be controlled and studied effectively. Within a short period *P. polycephalum* became a core experimental platform, driving a variety of different research efforts. The study of cell motility is particularly noteworthy in this context. Relevant reviews dating from the 1980s can be found in [4], [56], [150], [151].

After a period of intense research up to the 1980s, the remainder of the century saw a general decline in interest related to *P. polycephalum*. It should not end until the beginning of the 21th century, when exciting new question surrounding the networks formed by the plasmodium of *P. polycephalum* arose.[3] In particular, it was shown that the vein networks formed by the plasmodium of *P. polycephalum* exhibit

[3] Some members of the research community today have been humorously referring to these events as "the second coming" of *P. polycephalum*.

highly localized dynamic oscillatory behavior and a capability to adaptively change their topology. Together with its remarkable chemotactic abilities, these properties play a key role in the foraging behavior of *P. polycephalum*. Despite the lack of any centralized control, they enable the organism to act as an organized unit in order to establish robust and effective vein networks connecting a potentially large set of spatially distributed food sources.

In a similar context, the vein network established by *P. polycephalum* has been shown experimentally to mimic man-made transportation networks such as railway systems or highways [123], [174], [176]. Furthermore, it has been demonstrated that the plasmodium of *P. polycephalum* can establish the shortest path between a pair of food sources placed in a maze [124]. These astounding feats received major interest amongst scientists of many disciplines and simultaneously earned *P. polycephalum* a place in the perception of the general public. Within a short period of time, *P. polycephalum* became the focus of diverse interdisciplinary research efforts, engaging biologists, physicists and computer scientists alike. The renewed multi-disciplinary interest lead to a resurgence of research in *P. polycephalum* that can be observed to this day. For a rare, more recent review, and various recent results see [179] and [5], [120], [155], [167], [175]. Please note that this selection represents a fairly limited part of current research on *P. polycephalum*. In fact, it is focused on results revolving around the network forming plasmodium stage of *P. polycephalum* which are relevant in the context of this thesis.

1.2. Natural Computing

For some time the words *Nature* and *Computing* used to denote terms which could hardly be any more opposite to each other. Starting in the mid 1940s this perception began to fade gradually when researchers started to explore the exciting possibility of looking towards nature for alternative ways of doing computing. The search for new problem solving techniques, novel approaches to synthesize natural phenomena *in silico* as well as novel natural materials capable of realizing computations began. Today, the pursuit of these three distinct, yet interrelated approaches is known as *Natural Computing* [42], [48].

In this section we give a short introduction to the scope of natural computing and mention its most important areas of interest. Our exposition closely follows de Castro [43], who describes natural computing as the extraction of ideas from nature to develop computational systems, or using natural materials to perform computation. He suggests to divide the field into three main areas:

a) Computing inspired by nature

b) The simulation and emulation of nature by means of computing

c) Computing with natural materials

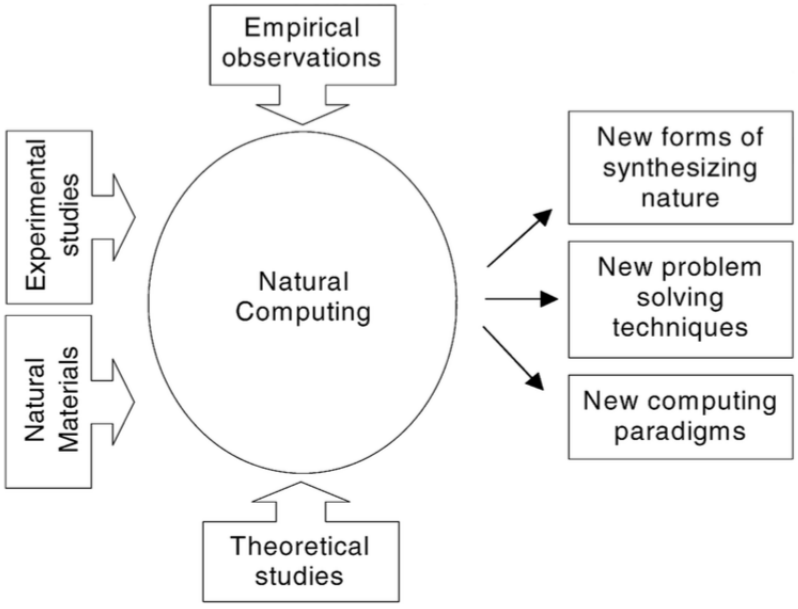


Figure 1.5.: A schematic of the components and goals of natural computing.

Figure 1.5 illustrates the goals of natural computing. The study of these three highly entangled areas seamlessly merge empirical observations and theoretical studies from diverse fields such as biology, physics, chemistry, engineering and computer science amongst others. Thus it is vital for researchers to be open, collaborate and share their ideas and knowledge in order to meet the challenges posed by the highly interdisciplinary field that is natural computing.

What follows is a short description of the main areas of natural computing including some of its most important achievements. For an extensive discussion and a comprehensive collection of references see [43].

1.2.1. Computing Inspired by Nature

In the area of natural computing, the goal is to utilize natural processes and phenomena as a source of inspiration in order to devise novel computational systems and algorithms geared towards solving complex problems. Of particular interest are alternative solution methods for problems for which standard techniques such as linear, non-linear and dynamic programming encounter severe difficulties. Some of the best known representatives of nature inspired computing include artificial neural networks,[4] evolutionary algorithms and swarm intelligence [43].

[4] The field of artificial neural networks deals with computational problems as opposed to the accurate biological modeling of the brain and the nervous system. The latter are questions of computational neuroscience.

Neural networks are formed by *artificial neurons* arranged according to a predefined network architecture, typically consisting of several functional layers of neurons. Triggered by a special *activation function*, each neuron may fire, i.e. mapping its potentially many weighted inputs to a single output. Different types of *learning strategies* may adjust the input weights of each neuron in response to encountered input stimuli. Indeed, some amount of learning, also known as training, is typically required before a neural network becomes competent. Given proper training various problems such as classification, pattern recognition or more general function approximation can be solved without further assisting the neural network. Unfortunately, a deeper understanding of why neuronal networks work the way they do remains elusive. A fact, that does not diminish their usefulness which lead to their large (commercial) success. For references see [26], [59], [77].

Evolutionary Computing and in particular genetic algorithms, try to leverage the power of evolution by simulating a population of individuals with certain traits. Individuals represent points in a search space associated with solutions to a given optimization problem. Each potential solution is evaluated with regards to the objective function of the problem. This process is known as determining the *fitness* of an individual. Individuals are then selected to reproduce, i.e. pass on their traits to the next generation, with a probability proportional to their fitness in an sexual or asexual fashion. In the former case the offspring is determined by a recombination of the traits of the parents. Further genetic variation is introduced by random mutation of traits. If this process is repeated, it evolves towards better solutions in an iterative manner. Typically such algorithms terminate after a predefined number of iterations or when the desired level of fitness is reached. Evolutionary algorithms have found success in problems such as routing, scheduling, packing and various machine learning tasks [13], [160], [188]. Important practical applications include determining optimal design choices in various engineering settings [101], [153].

Swarm Intelligence encompasses techniques for problem solving that are inspired by the collective behavior of human or animal societies. They rely on the social behavior of a population of individuals capable of interacting with the environment or one another in either a direct or indirect fashion [43]. The popular *ant colony optimization* approach uses artificial ants that lay and follow artificial *pheromone trails*. In the shortest path problem for instance, ants travel all possible paths between two distinct nodes at the beginning. However, as time goes by, longer routes have less pheromones associated with them due to pheromones evaporating. Since the probability for an ant to choose a given route is proportional to the present amount of pheromone, ants become less likely to travel long routes and tend to concentrate on the shorter paths, thereby reinforcing them repeatedly. As a result, ants are progressively converging towards the shortest path. This general scheme can be adapted to

various other discrete optimization problems such as the traveling salesman problem and certain vehicle and network routing problems. For references see [30], [31], [32], [66].

1.2.2. Synthesis of Nature by Means of Computing

The synthesis of nature by means of computing aims for full or partial reproduction of phenomena, behaviors or patterns observed in nature [43]. This *in silico* approach allows the exploration of a wide range of questions which may not be tractable using traditional analytic or experimental techniques. Frequently theoretical work suggest theories and models to be studied *in silico* using simulations with the aim of obtaining a deeper understanding of the original phenomenon being modeled. The simulation itself may serve as an evaluation of a proposed model, ideally suggesting further improvements to the model and itself. In addition to the sheer growth in computing power observed in the last few decades, much of the success of fields like computational biology, computational chemistry or computational physics owes to the symbiotic relationship between theory and simulation.

Examples for the synthesis of natural phenomena include *cellular automata* as a model for self-reproduction or *L-systems* modeling the development of multicellular organisms. These concepts were designed to capture and study certain, well-defined rules and associated behaviors. However, over time their elegance and simplicity lead to a variety of unforeseen applications. Cellular automata have played a role in modeling the physics of fluids or gases on lattices [147], and the microscopic study of high-way traffic [117], to name but a few. L-systems have been a successful tool in the study of various properties of *virtual plants* [170]. In addition to that they have found applications outside of biology in formal language theory, the study of tumor growth and even musical composition [42].

Arguably the most exciting field we like to mention in the context of synthesis by means of computing is the study of *artificial life* [105]. Within artificial life, it is proposed that life itself should be regarded as an emergent property resulting from the organization of matter as opposed to a simple property of matter itself. It follows that besides the carbon based chain of life observed on earth, i.e. *life-as-we-know-it*, different materials could lead to alternative life-like organizations aptly termed *life-as-could-be*. While it is debatable what properties such an organization requires to be rightfully termed *alive*, the prospect of new unknown forms of life is extremely exciting.

A well-known example for non-trivial behavior in the context of life-as-could-be is demonstrated by *boids* [144]. Here artificial life is represented merely as virtual agents that are allowed to move in space. They obey the following simple rules: (1) Avoid collision with neighboring boids or obstacles; (2) Mirror velocity and direction of neighboring boids; (3) Stay close to neighboring boids. From these simple rules complex collective behavior emerges allowing a flock of boids to closely imitate the non-trivial behavior seen in flocking birds or shoals of fish [43].

The examples in this section illustrate that much can be learned from attempts to

synthesize nature by means of computing. It is likely that ongoing research efforts will sooner or later drastically alter our understanding of nature in general and life in particular. It is this authors personal opinion that the question of how life arises from inanimate matter constitutes one of the most important scientific questions in the entire history of science.

For extensive reviews on the topic of artificial life, we refer the reader to [25], [29], [105], [171].

1.2.3. Computing with Natural Materials

Computing with natural materials aims to go beyond standard transistor-based computing by utilizing natural materials other than silicon as media for computing. Examples for approaches to computing that are in stark contrast to conventional computing are DNA and quantum computing [43].

In *DNA computing* complex molecules such as DNA strands are used to store and process information. Complex computations can then be realized using a set of basic operations that apply to interacting molecules. These operations are derived from molecular biology and involve fusing, copying or deletion of DNA strands to name but a few. The power of DNA computing derives from the fact that a large number of molecules may interact with each other simultaneously. Despite the potentially slow individual reactions, the inherent massive parallel information processing capability of DNA computing lead to novel approaches to hard combinatorial problems such as the Hamiltonian path problem [3]. Today various other applications are known ranging from matrix multiplication to cryptography [33], [34], [134], [139]. Drawbacks of DNA computing include scaling with problem size as well as retrieving the actual problem solution from the DNA computation [43].

Another promising alternative to classical computing is *quantum computing* [128]. Here the so-called *qubit* replaces the classic binary bit which is limited to the two states, 0 or 1. In contrast to that, the qubit, thanks to its quantum nature, can represent any superposition state of the classical bit states. Thus it is capable of representing all possible combinations of 0 and 1 *simultaneously*. As a result, a quantum computer using n qubits may be in 2^n different states at the same time, very much unlike a classical computer which is in exactly one state at a time.

Elementary operations on qubits take the form of quantum logic gates. Gates may be arranged in such a way that they compute the solution to a given problem. A sequence of gates operating on qubits is then called a *quantum algorithm*. In the final steps of any quantum algorithm the qubit superposition is collapsed in order to obtain the result which is often probabilistic in nature. Exploiting the massive in-build parallelism, dedicated quantum algorithms have been developed that outperform all known classical algorithms for a small set of problems. The two most prominent ones being Shor's integer factorization and Grovers's database search algorithm [67], [156].

In addition to speeding up certain tasks, quantum systems may operate in ways that classical systems cannot. This fact is exploited in *quantum communication* or

quantum cryptography, where protocols have been developed which reduce communication complexity or provide perfect secrecy [43]. A major roadblock on the way of success for quantum computing consists of the physical realization of coherent systems capable of preparing qubits and executing quantum algorithms reliably. Here the very quantum nature which enables extraordinary computation, presents formidable difficulties when it comes to controlling the quantum system. Unfortunately this fact still severely limits the number of qubits that have been used in successful computing applications [128].

1.3. Natural Computing with *P. polycephalum*

Starting with the year 2000 a series of break-through experiments caused the unassuming organisms that are slime molds to become the focus of intense research efforts. In particular, it was demonstrated that in its plasmodium stage, *P. polycephalum* is capable of reacting to environmental conditions in a manner that is reminiscent of optimization processes. Examples include the maximization of food uptake given a small set of food sources, or the minimization of exposure to light known to be harmful to the organism. It is striking that there is no central control coordinating these nontrivial processes. Since *P. polycephalum* lacks any sort of brain or nervous system, it has been speculated that the observed capabilities are emergent as a result of local interactions. Given its abilities, *P. polycephalum* can be seen as a highly distributed system with properties suitable for distributed natural computing.

In order to explain the behavior of *P. polycephalum* and to harness its in-built optimization capabilities, several different approaches were put forward. The most successful amongst those either focus on modeling certain key observations that were discovered experimentally or manipulate the organism *in vivo* in such a way that its behavior can be interpreted as computation. In the next section we survey some of the most influential experiments which paved the way for natural computing with *P. polycephalum*.

1.3.1. Key Experiments and Observations

Emergence of Synchronization in *P. polycephalum*

In the last decade of the 20th century the rhythmic contractions exhibited by the plasmodium of *P. polycephalum* were the subject of a series of experiments aimed at shedding light onto the basic mechanisms governing *P. polycephalum* [115], [121], [165]. Soon it was established that mechanochemical reactions among intracellular chemicals such as ATP and calcium generate periodic cycles of contraction and relaxation of actomyosin fibers, explaining the observed oscillations in the thickness of the plasmodium. Periodic changes in thickness induce pressure differences between different parts of the slime mold which cause protoplasmic fluid to be streaming to

and fro, exhibiting periodic reversals of direction. Given the experimental findings it seemed natural to view the plasmodium of *P. polycephalum* as an ensemble of coupled non-linear oscillators. Using this Ansatz, mutual entrainment of intracellular oscillators may give rise to self-organization and the information processing capabilities of *P. polycephalum*.

In this context, a key experiment was presented in 2000 which uses the plasmodium of *P. polycephalum* to realize a living system of coupled oscillators [166]. To this end a micro-fabricated structure was prepared, consisting of two identical circular reservoirs, connected by a channel of a predetermined width and length. In the experiment, the two reservoirs and the channel are populated with plasmodium. The reservoirs act like two distinct *P. polycephalum* oscillators while the channel ensures a controllable coupling between the two. Here the width of the channel represents the coupling strength while the length of the channel corresponds to the time delay in the coupling. The setup depicted in Figure 1.6 realizes a system of two delay-coupled oscillators whose thickness oscillations were studied as a function of the channel dimensions.

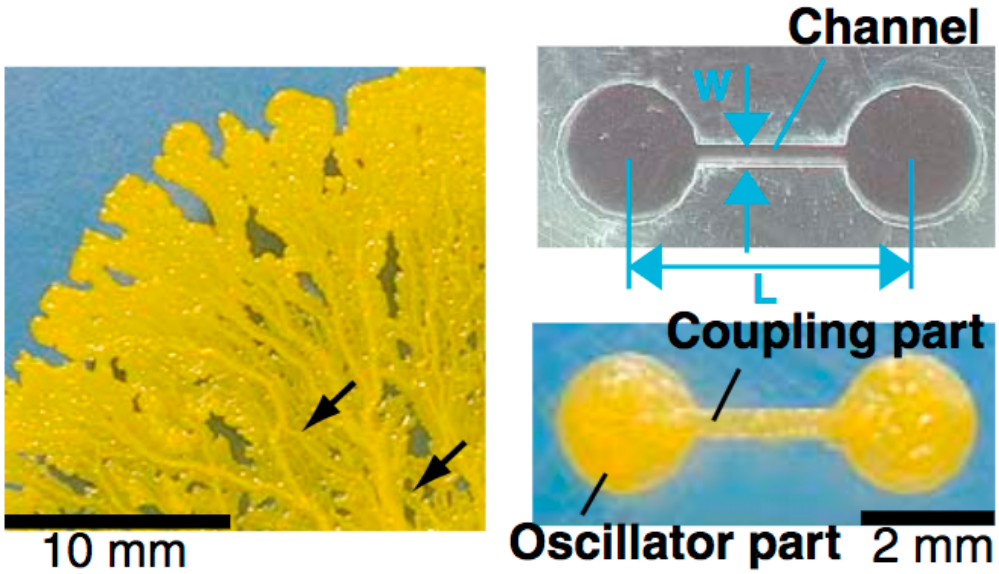
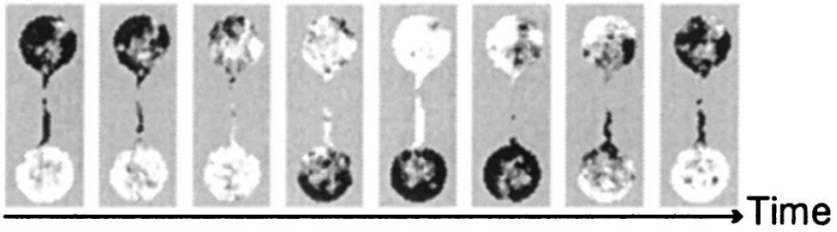
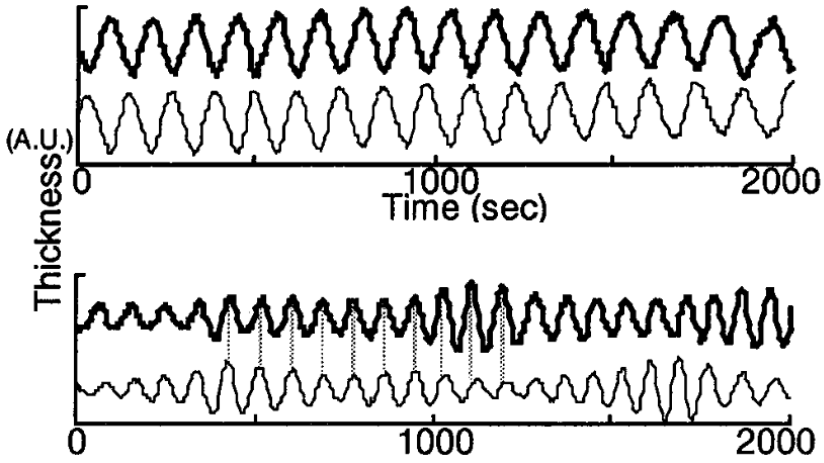


Figure 1.6.: (a) Detail of the growing front of *P. polycephalum*. Arrows indicate the presence of thick veins. (b) Top: A micro-fabricated structure designed to confine *P. polycephalum*. The diameter of the reservoirs was chosen such that they can be regarded as distinct oscillators. Bottom: The dumbbell shaped structure loaded with *P. polycephalum*. Reprinted from [166].

Figure 1.7 shows rich self-synchronizing oscillation patterns exhibiting both in-phase and anti-phase entrainment consistent with theoretical expectations. The experiment shows that the geometry of the channel plays a key role in the generation of thickness oscillations driving peristaltic pumping of protoplasm.



(a)



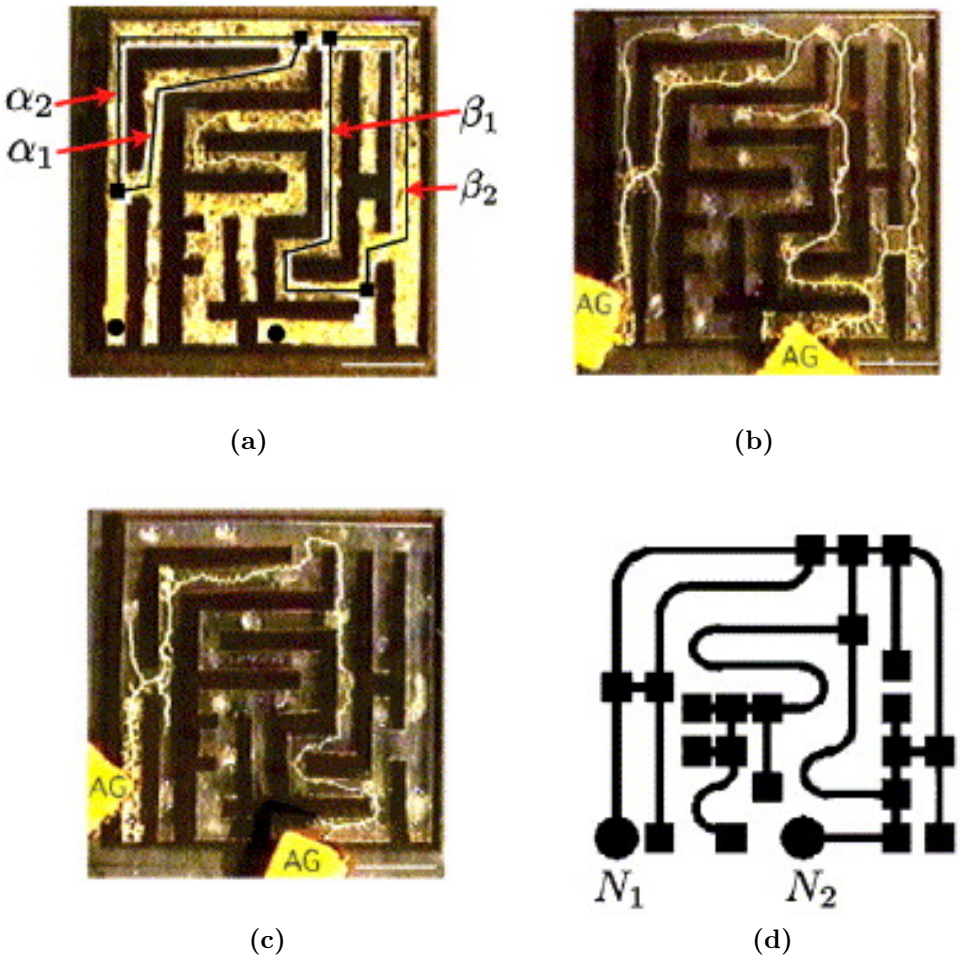
(b)

Figure 1.7.: (a) Visualization of thickness anti-phase oscillations propagating between the two oscillators by means of the transmitted light intensity through the plasmodium. Darker/brighter areas indicate an increase/decrease in thickness. The time interval between consecutive images is 16 s. (b) Time development of thickness oscillations. The thick respectively thin lines represent the thickness of individual oscillators obtained by averaging the transmitted light intensity for each oscillator. Top: Anti-phase oscillations were observed for $W = 0.4000\,\text{mm}$ and $L = 4\,\text{mm}$. Bottom: In-phase oscillations were observed for $W = 0.5000\,\text{mm}$ and $L = 10\,\text{mm}$. Adapted and reprinted from [166].

Shortest Path in a Maze

Introduced in the year 2000, in the so-called "maze experiment", the plasmodium of *P. polycephalum* is placed in a maze, see Figure 1.8a [124]. After the plasmodium has spread evenly across the entire maze, two food sources are introduced at two specific points in the maze, see Figure 1.8b. The organism reacts to the food source by slowly retracting from areas of the maze that do not coincide with any path connecting them. This process continues for several hours until the plasmodium takes the shape of one single thick vein connecting the food sources. In repeated experiments, it was shown that after sufficient time this vein does not occupy any path, but selects the shortest path ($\alpha_1 + \beta_1$, see Figure 1.8a) in most of the cases. This demonstrates the organisms remarkable ability to iteratively improve the efficiency of fluid flow between the two food sources. By making veins progressively shorter and wider, the hydrodynamic resistance to protoplasmic flow is minimized which is beneficial in terms of nutrient transport[96].

In terms of natural computing, one may interpret the maze as a graph with edge weights equal to the lengths of the associated maze segments and the food sources as two distinct nodes N_1 and N_2, see Figure 1.8d. Thus the slime mold *Physarum polycephalum* can be seen to demonstrate an *in vivo* approximate solution to the $s - t$ shortest path problem with $s = N_1$ and $t = N_2$.



(a) **(b)**

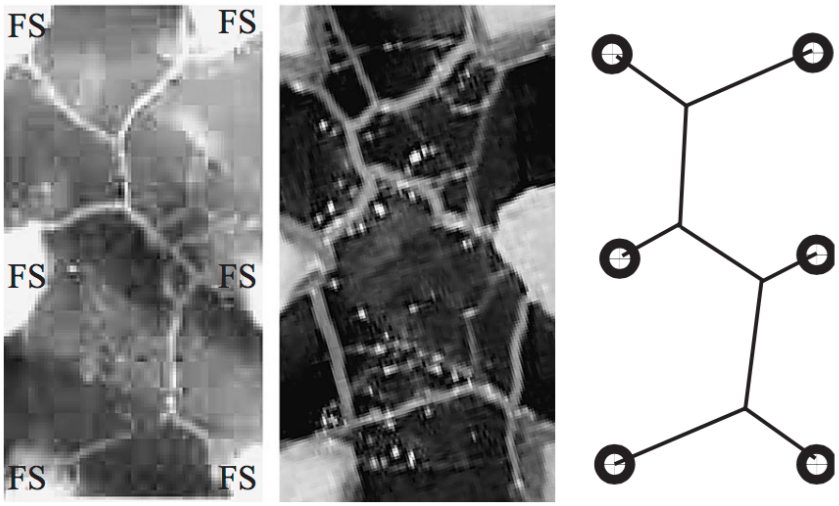
(c) **(d)**

Figure 1.8.: (a) The plasmodium of *P. polycephalum* is evenly spread across a maze. Filled black circles denote two specific points in the maze. Arrows indicate segments of possible paths between them. (b) Two food sources, labeled `AG`, are introduced at the specific points. After a while the plasmodium retreats from areas of the maze that do not intersect with a path connecting the food sources. (c) The plasmodium settles on the shortest path between the two food sources. (d) The maze can be interpreted as an abstract graph with two distinct nodes N_1 and N_2 corresponding to the food sources. The scale bars denote 1 cm. Reprinted from [175].

Efficient networks maximizing food uptake given multiple food sources

In 2004 the plasmodium of *P. polycephalum* was presented with multiple food sources arranged in various regular patters including equilateral triangles and small grids [120], [123]. Independent of the chosen pattern, the organism reacted by changing its shape from a sheetlike plasmodium to a distinct network featuring thick veins. This network was found to balance two competing yet desirable features. Namely, small total length of the tubular network and high tolerance against accidental loss of connectivity. The first feature implies that the biomass required for maintaining veins be small. As a result the bulk of the biomass may concentrate on the food sources themselves in order to improve food uptake. The second feature makes sure that the organism is not disconnected after severing a small number of veins. This allows the organism to keep exploiting obtained food sources and information as a unit. Such behavior is crucial for instance, if a remote part of the vein networks senses an unclaimed food source. Then this information may be processed by the entire organism leading to a potential redistribution of biomass, enabling further improvement of food uptake. If the part which discovers the new food source is disconnected from the main body, the latter cannot react to benefit from the discovery.

Experimentally it was found that the networks formed by *P. polycephalum* resemble minimum Steiner trees with additional cycles contributing a degree of fault tolerance, see Figure 1.9. The authors allow however, that the approximate Steiner points are likely to be a mere byproduct of searching for short connections between food sources. The inherent capability of forming functional networks was demonstrated impressively in another experiment where *P. polycephalum* was set up to approximate the railway area of Tokyo and the greater Kanto area [176].



(a) (b) (c)

Figure 1.9.: (a) A network resembling the minimum Steiner tree connecting six food sources (FS) laid out in a regular pattern. (b) An example of a network with a SMT-like pattern in the upper half and a cycle of veins in lower half. (c) A sample minimum Steiner tree on six nodes. Reprinted from [120].

Minimizing Risk Associated with Illumination

In 2007 the plasmodium of *P. polycephalum* was challenged with a different variant of optimization problem involving the organisms strong tendency to avoid light.

In the experiment the plasmodium is allowed to spread evenly across a rectangular petri dish. Next, two food sources are introduced at two adjacent corners of the container and a fraction of the container is evenly illuminated with white light. When the whole container is illuminated *P. polycephalum* forms a thick vein connecting the two food sources, approximating the shortest path between them. If only a fraction of the container is illuminated, *P. polycephalum* still connects the two food sources forming a thick vein. However, now the organism is trying to minimize its exposure to light by striking a balance between short connection and a short path through the illuminated area. In fact, the behavior of *P. polycephalum* in this experiment is strikingly similar to the behavior of refracting light as given by Snell's law. Figure 1.10 illustrates the experiment.

The illumination experiments yet again demonstrate the uncanny ability of *P. polycephalum* to solve complex problems. Furthermore, it shows that the behavior of *P. polycephalum* can be effectively steered using light. Together with the potential to use chemical attractants and repellents, rich possibilities of controlling the behavior of the slime mold are available.

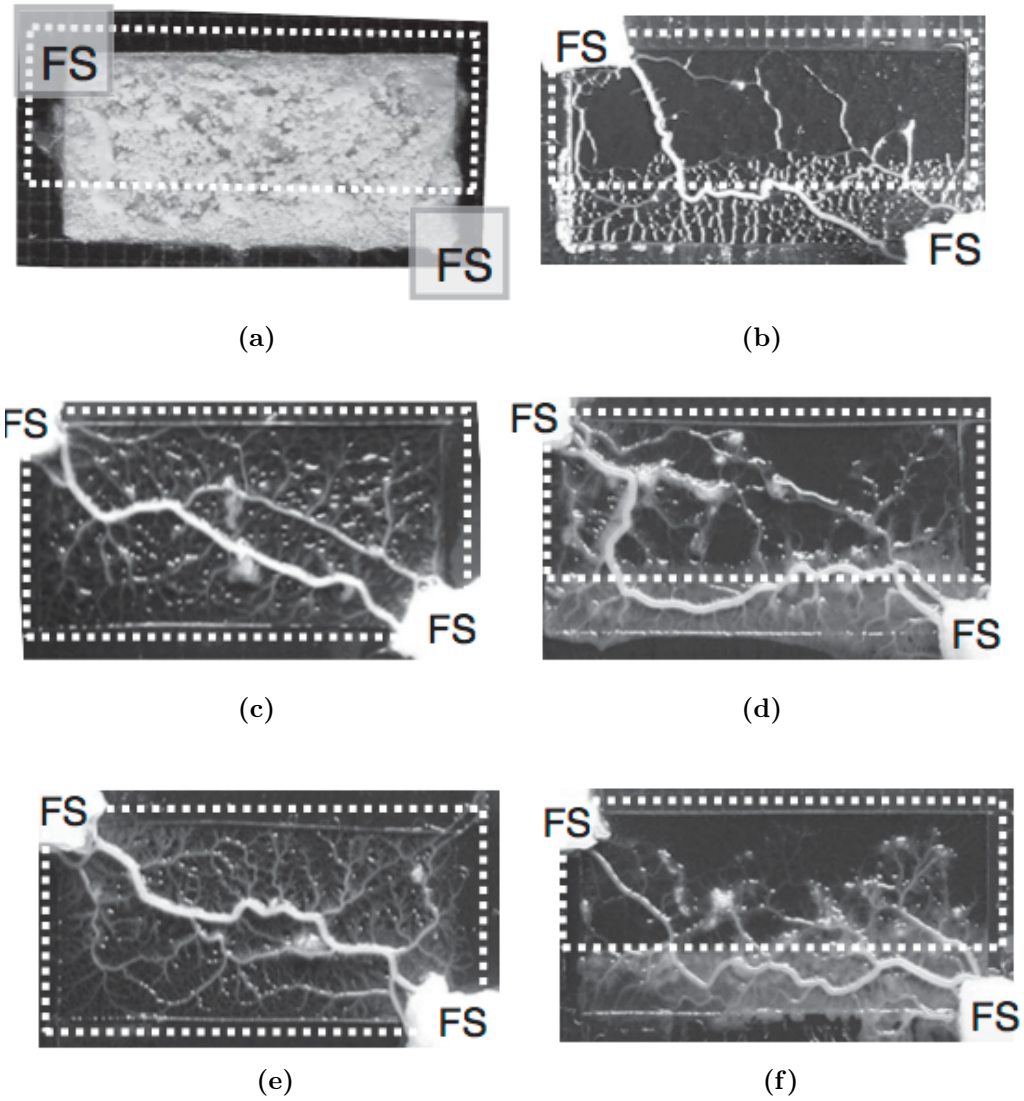


Figure 1.10.: (a) The rectangular sheetlike morphology of the organism immediately before the introduction of two food sources (FS). The region affected by illumination is indicated by a dashed rectangle. In (c) and (d) the entire region was illuminated. As a result an the shortest path between the two food sources is approximated by a thick vein. In (b), (d) and (f) partial illumination leads to a reduced path length through the illuminated area at the expense of an increase in total path length. Note the natural variation in the exhibited paths from experiment to experiment. Reprinted from [118].

1.3.2. Natural Computing Approaches

After a short survey of key experimental observations presented in the previous section, we move on to discuss common strategies to use the plasmodium of *P. polycephalum* as a medium for natural computing. Here we present examples for the different paradigms discussed in the section natural computing. In the interest of brevity, we focus on the most successful approaches and refer the reader to the literature for details.

Computing Inspired by P. Polycephalum

The approach aims at modeling certain aspects of *P. polycephalum* with the goal of using the resulting models for *in silico* computation. The major challenge here is to look past the intricate bio-physical processes that are observed experimentally and find a suitable level of abstraction capable of reproducing the computational abilities of *P. polycephalum*.

Shortly after it was demonstrated that *P. polycephalum* forms networks connecting more than two food sources, Nakagaki et al. went on to explain their own observations on a physiological level in [174].

They claim that the plasmodium of *P. polycephalum* organizes itself through the formation of thick veins through which the protoplasm is driven as a result of periodic contractions of the veins themselves, i.e. peristaltic pumping. Veins are formed when streaming of protoplasm persist in a given direction for a sufficiently long time [125]. On a molecular scale Actomyosin fibers carried by the protoplasmic flow attach themselves to each other forming a mesh of fibers, eventually resulting in tubular structures. The fast flowing protoplasm causes shear stress, exerting a force which stretch-induces a regular orientation of the otherwise arbitrarily oriented fibers along the direction of flow. As a result, veins with a large flux manage to accumulate and orient even more Actomyosin fibers which over time lead to an increase in vein diameter. This in turn further reduces the resistance to flow, since under the assumption that the fluid flow in the veins of *P. polycephalum* is a Poiseuille flow, the theory of hydrodynamics dictates that it be proportional to the forth power of the vein diameter and inverse proportional to the vein length. It follows that veins which are carrying little flow either are long and/or thin or happen to be dead-end veins. The latter are likely to degenerate over time as experimentally demonstrated in various experiments. At the same time, reinforcing thick and short veins increases the efficiency of fluid flow, which in turn enables effective circulation and transport of nutrients, nuclei and other factors.

These considerations became the basis of a concise model presented in 2006, which captures the behavior of the slime mold as demonstrated in the maze experiment and lead to the first natural computing approaches based on *P. polycephalum* [174]. The model can be formulated on a graph $G = (V, E)$, where the node set V denotes the junction points of veins and the edge set E denotes the veins themselves. Each node $u \in V$ has a hydrodynamic pressure associated with it denoted by $p_u(t)$. Each

edge $e = (u, v) \in E$ has a length L_e as well as a time-dependent flow through the vein labeled $Q_e(t)$. Here it is assumed that the flow is a Poiseuille flow such that

$$Q_e(t) = \frac{\pi r_e(t)^4}{8\eta} \frac{p_u(t) - p_v(t)}{L_e}, \tag{1.1}$$

holds. Here $r_e(t)$ is the diameter of an edge and η refers to the viscosity of the protoplasmic fluid. Let $D_e(t) = \pi r(t)^4/8\eta$. Then we can simplify Equation (1.1) to read

$$Q_e(t) = \frac{D_e(t)}{L_e}(p_i(t) - p_j(t)). \tag{1.2}$$

Let us choose two distinct nodes $s, t \in V$ representing the source and sink nodes in the graph. Using this definition we obtain an equation for each edge $e \in E$:

$$\sum_{e=(u,v)} Q_e(t) = \begin{cases} 1 & \text{for } u = s, \\ -1 & \text{for } u = t, \\ 0 & \text{otherwise}. \end{cases} \tag{1.3}$$

Note that the l.h.s is chosen such that flow is conserved everywhere except at the source and the sink where flow is entering respectively leaving the graph. Thus one unit of flow is introduced to the system. Equation (1.3) can be solved to obtain the values of the $Q(t)_e$ and the $p_u(t)$ simultaneously.

Finally the time-dependence of $D(t)_e$ is choosen such, that the positive feedback between $D(t)_e$ and $Q_e(t)$ is as described above. For each $e \in E$ the dimensionless equation for $D(t)_e$ reads

$$\frac{d}{dt} D_e(t) = f(Q(t)_e) - D_e(t), \tag{1.4}$$

which is called adaptation or *evolution equation*. Here f is a monotonically increasing continuous function such that $f(0) = 0$. Equation (1.4) is the key of the model which formalizes the experimental observations reported earlier. If at time t the flux through an edge is such that $f(Q_e(t)) < D_e(t)$ then $D_e(t)$ decreases which further reduces the flux through e. In this scenario an edge e is degenerating. Likewise $f(Q_e(t)) > D_e(t)$ leads to an increase in $D_e(t)$ which further increases the flux. In this case e is reinforced. Only, $f(Q_e(t)) = D_e(t)$ enables a stationary state which leaves $D_e(t)$ and $Q_e(t)$ constant.

The model thus evolves the thickness of veins in *P. polycephalum* according to Equation (1.4) relying on the values of $p_u(t)$ and $Q_e(t)$ which are governed by Equation (1.3). For suitable choices of $f(Q_e)$ this system can be discretized and solved numerically [174]. For the graph depicted in Figure 1.8d a stationary state was found where the $D_e(t)$ of all edges that do not belong to the shortest path between $s = N_1$ and $t = N_2$ vanished. At the same time $D_e(t)$ of edges on the shortest path approached a constant. That is, just like the slime mold, the model has selected the shortest path in the graph between s and t.

Problem	Authors	Year
Transport network design	Tero et al. [176]	2010
Linear programs	Johannson et al. [86]	2012
Learning Bayesian network structure	Schön et al. [152]	2012

Table 1.1.: Various problems and related solution strategies inspired by the original Physarum solver.

After this initial demonstration of the viability of *P. polycephalum* as a medium for natural computing, mathematicians and computer scientists started to analyze the model and its variants contributing a sequence of convergence proofs and complexity bounds. For suitable choices of $f(Q_e)$ convergence was proven for planar graphs at first [113], [114], but shortly thereafter proofs were extended to general graphs for the original model and for variations thereof [23], [35], [36], [83]. The fact that the theoretical properties of the so-called *Physarum solver* are rather well understood is a rarity amongst natural computing algorithms.

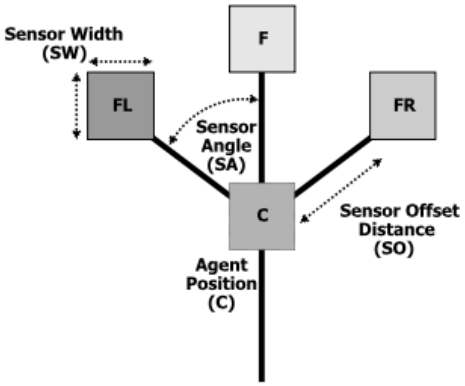
Due to its success and intuitive basic idea, the Physarum solver has received much interest in the scientific community. Subsequently it inspired several other algorithms with varying degree of similarity to the original problem. An incomplete selection is given in Table 1.1.

Synthesis of P. Polycephalum Using Computation

This approach seeks to faithfully synthesize *P. polycephalum* by means of simulation. To do so, it relies on a suitable model capable of mimicking the behavior of the organism. Contrary to the approach presented in the previous section, finding a model that solves a computational problem is not the main goal. However, it has been shown that in the case of *P. polycephalum* accurate modeling of the organism produces a model with problem solving capabilities.

In 2010 Jones suggested to take a closer look towards the foraging behavior of *P. polycephalum*, emphasizing its ability to move towards food based on sensing chemicals signaling its presence [89]. In particular he puts forward the idea of modeling the plasmodium of *P. polycephalum* as a virtual 2D material consisting of a large population of virtual agents. These agents reside on a 2D lattice whose nodes store the concentration of a hypothetical chemical representing the presence of food. Agents are equipped with sensors that allow them to orient themselves towards the locally strongest concentration c of this chemical in the lattice. After reorientation the agent deposits a chemical concentration equal to c at its current location and then takes a step forward. This behavior effectively couples neighboring agents with each other, since they read and react to the trails of chemicals that are being deposited by others. A similarity with the general idea of ant colony optimization is observed. Figure Figure 1.11 depicts a schematic of an agent and its sensors.

Given suitable parameters settings, the collective behavior of all agents leads to



(a)

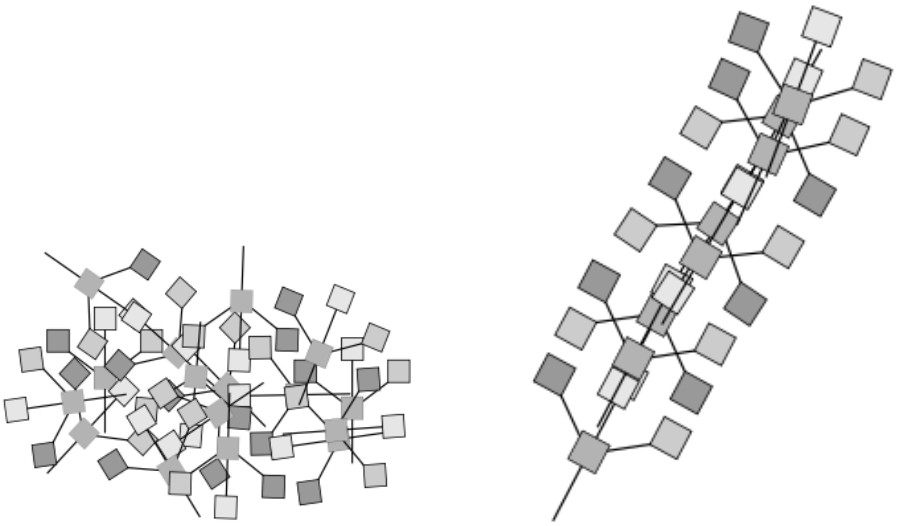
Figure 1.11.: Structure of a virtual agent showing central position and 3 forward-facing offset sensors. Reprinted from [89].

the spontaneous formation of transport networks. The emergent global patterns formed by the collective of agents resemble the actin-myosin mesh structures forming the veins of *P. polycephalum*, while the individual trajectories of the agents can be interpreted as protoplasmic fluid flowing through the veins of the organism, see Figure 1.12. Figure 1.13 shows how an initially randomly distributed collection of agents evolves in time to eventually form a network. In particular, certain parameters of the model can be tuned such that the collective demonstrates minimization processes observed in [21], [90], [91].

In [92] it is pointed out that pattern formation in itself does not yet constitute computation. Thus inspired by previous experimental results, it was suggested to add additional external stimuli to the lattice. These act as the input and the constraints necessary for a controlled computation. They can be attractants or repellents which are projected onto the lattice and create concentration gradients which agents may react to. The placement of additional stimuli acts as constraints to the natural minimization process realized by the virtual material. The constrained process may be interpreted as computation. It is halted once the material has reached a stable configuration which is then interpreted as the solution or the output of the computation.

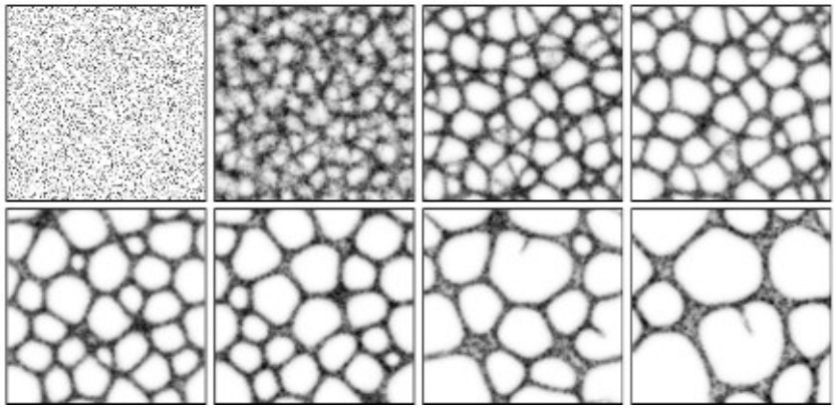
An example of this approach can be seen in Figure 1.14 where six attractants where projected onto the chemo-attractant lattice. They correspond to food sources representing the input to the virtual *P. polycephalum* experiment. When the virtual material is introduced it undergoes morphological changes constrained by the input. After a certain number of iterations the configuration has stabilized, revealing a minimum Steiner tree.

Further examples of problem solving by synthesizing *P. polycephalum in silico* are given in Table 1.2.



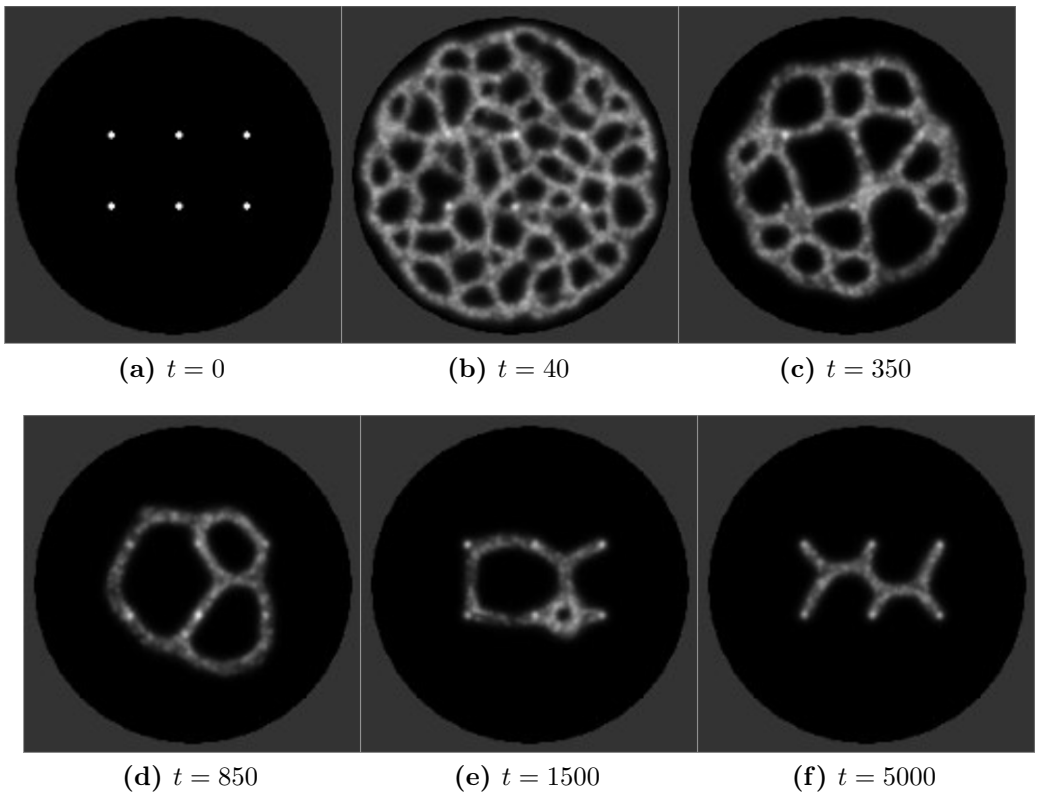
(a) **(b)**

Figure 1.12.: (a) A static collective of agents approximates plasmodium actin-myosin mesh. (b) A mobile stream of agents approximates protoplasmic flow within the plasmodium. Caption and figures reprinted from [89].



(a)

Figure 1.13.: The coupling of agents with each other causes spontenous formation and evolution of transport networks. Reprinted from [92].



(a) $t = 0$ **(b)** $t = 40$ **(c)** $t = 350$

(d) $t = 850$ **(e)** $t = 1500$ **(f)** $t = 5000$

Figure 1.14.: Material computation by constraining pattern formation. (a) A pattern of six nodes inside a circular arena is projected into the diffusive lattice as attractant stimuli. (b) Spontaneous formation of a transport network from random inoculation positions realized with 4000 particles. (c)-(e) The minimization of the network is constrained by the attraction to nodes. (f) The final stable configuration (output) is a Steiner minimum tree. Caption and figure adapted from [92].

Problem	Authors	Year
Cellular automata	Gunji et al. [70], [71]	2008,2011
Voronoi diagrams	Jones et al. [95]	2013
Traveling salesman	Jones et al. [93]	2014
Data smoothing and filtering	Jones et al. [94]	2014
Convex and concave hull	Jones et al. [92]	2016

Table 1.2.: Various problems and related solution strategies obtained via synthesis of *P. polycephalum*.

Computing with Live P. Polycephalum

In order to realize computation with live *P. polycephalum*, the plasmodium may be interpreted as a parallel amorphous biological computing system. Here input is encoded via the spatial configuration of food sources. The slime mold reacts to the presented stimuli in a dynamic process which is interpreted as computation. Finally, once the organism has settled in response to the input, the configuration of its vein network is interpreted as the output of the process. The initial experiments conducted by Nakagaki et al. can be considered the first deliberate realizations of natural computing with *P. polycephalum* as a computing medium. Soon thereafter further possibilities to manipulate and steer the behavior of the plasmodium with chemical attractants, repellents and light where explored.

Two approaches, the perhaps most impressive, deserve to be mentioned separately. The first demonstrates an *in vivo* solution to the problem of planning a transportation network connecting a set of cities. The second shows how a cleverly devised feedback-control system allows *P. polycephalum* to act as an entirely different natural computing system, namely a neural network.

Tero et al. cultivated *P. polycephalum* in a Petri dish designed as a miniature replica of the greater Kanto region where oat flakes were placed at the locations of Tokyo and other major cities. Geographical constraints such as oceans, lakes and mountains were realized by means of a corresponding illumination mask. Since *P. polycephalum* tends to avoid light, growth was restricted to the shaded areas. Under these conditions *P. polycephalum* soon formed a network spanning all the food sources available in the shade. Surprisingly, the networks formed by *P. polycephalum* were found to resemble the actual railway networks serving the Kanto region while balancing efficiency, fault tolerance and cost [176].

The experiment impressively demonstrated that *P. polycephalum* can compute a feasible solution to the complex problem of designing a desirable real-life transport network, see Figure 1.15. Note that properly encoding the input by combining the location and quality of food sources as well as the illumination mask representing geographical constraints is critical. In addition to the *in vivo* computation, Tero et al. also suggested a model based on the Physarum solver suitable for an *in silico* replica of this experiment.

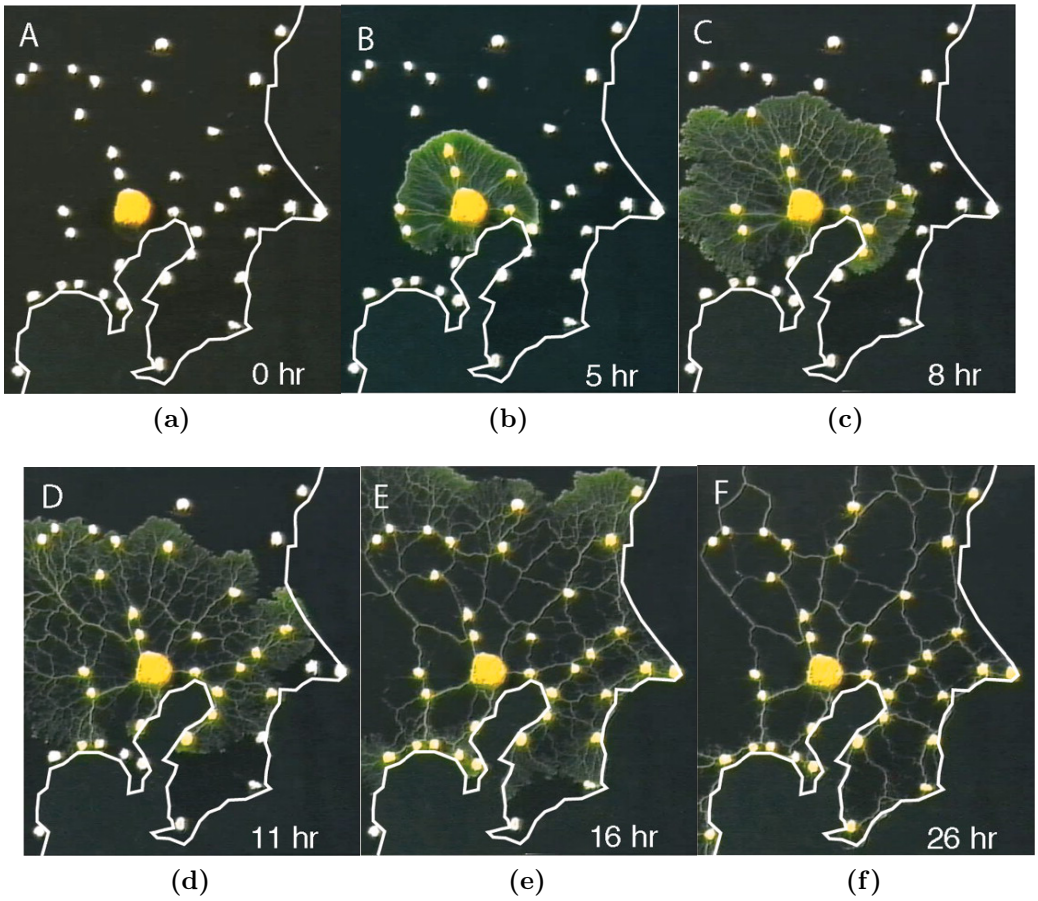


Figure 1.15.: Network formation in *Physarum polycephalum*. (a) At $t = 0$, a small batch of plasmodium was placed at the location of Tokyo in an experimental arena bounded by the Pacific coastline (white border) and supplemented with additional food sources at each of the major cities in the region (white dots). The horizontal width of each panel is 17 cm. (b) to (f): The plasmodium grew out from the initial food source with a contiguous margin and progressively colonized each of the food sources. Behind the growing margin, the spreading mycelium resolved into a network of tubes interconnecting the food sources. Caption and figure reprinted from [176].

While Tero et al. harvested the computing capabilities that follow from the natural foraging behavior and light-sensitivity of *P. polycephalum*, Aono et al. suggest to trick *P. polycephalum* into realizing a neural network [8], [9], [10], [11]. Their scheme is as follows: First, they design a star-shaped container where the "legs" of the container represent 8 identical "neurons", see Figure 1.16.

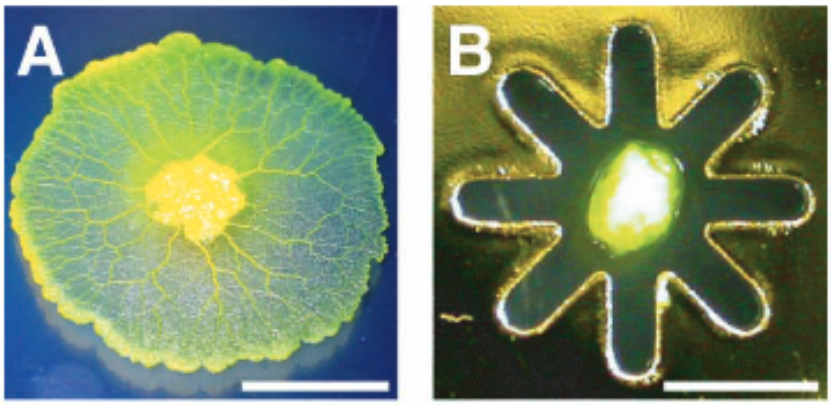


Figure 1.16.: L.h.s.: *P. polycephalum* expanding unrestrictedly from an inoculation site. R.h.s.: *P. polycephalum* placed in a star-shaped container. Scale bar equals 2 mm. Reprinted from [10].

When *P. polycephalum* is placed inside the star it spreads equally across it. A neuron x_i is considered active if more than a quarter of its area is occupied by the organism. Consequently, any neuron can be deactivated by illuminating its branch, which causes the plasmodium to retreat owing to its photo avoidance. The behavior of *P. polycephalum* can be exploited to realize logical functions such as logical `NOR` by enforcing that a neuron x_i be deactivated if any of its neighbors are active.

Thus, the exploring slime mold influences itself through the illumination pattern as it looks to occupy as large an area of possible. Under these conditions the plasmodium is lead towards certain configurations with a stable illumination pattern. When all neurons remain unchanged, the adopted configuration is interpreted as the output of the computation. Figure 1.17 shows various states the experimental setup can take.

Interestingly, the shape changes that the plasmodium undergoes in this setup may also be interpreted as state transitions in a McCulloch-Pitts neural network [10]. Such networks have well studied computational capabilities and are known to be capable of approximating computational problems such as the traveling salesman problem. Exploiting the computational power of neural networks, Aono et al. demonstrated that *P. polycephalum* can be used to realize an *in vivo* computing system. Indeed, using an experimental setup with $N \times N$ branches it is possible to guide *P. polycephalum* towards finding a solution for a TSP problem with $N = 4$ [11].

This astonishing ability to guide and manipulate the behavior of *P. polycephalum* is the key to various *in vivo* computing systems devised so far, see Table 1.3.

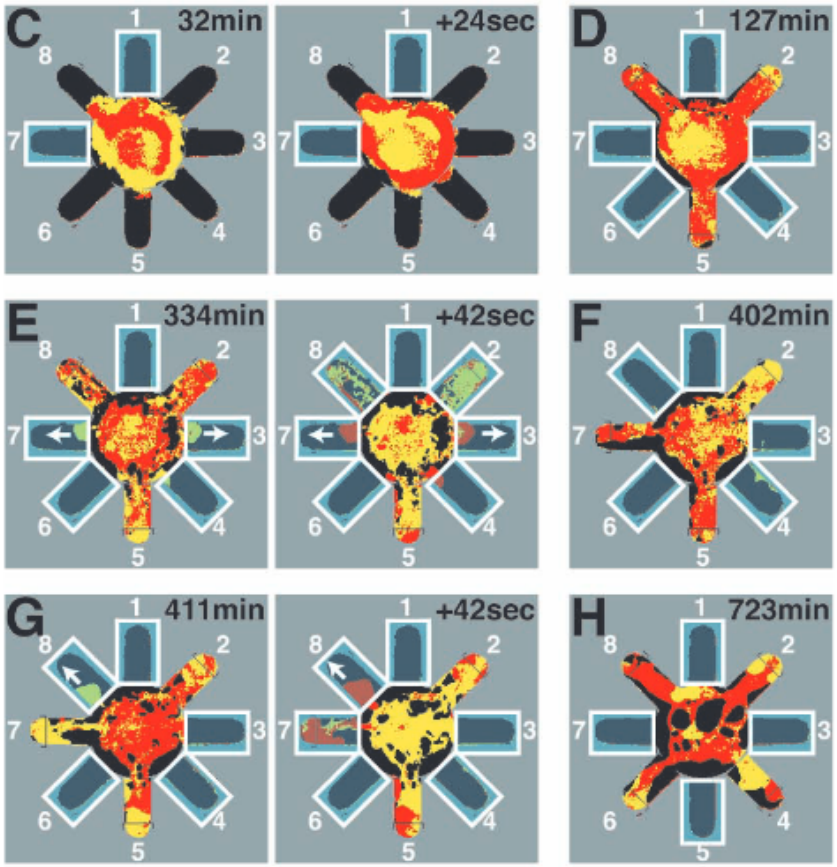


Figure 1.17.: Time development of the *P. polycephalum* neuron system. Illuminated branches of the container are indicated by white rectangles. The thickness oscillation phases are binarized as red and yellow for relaxing (thickness increasing) and contracting (decreasing) states, respectively. The l.h.s. row shows examples of transient states where the plasmodium is still exploring the container. The center row illustrates short term changes in thickness. The r.h.s. shows various stable configurations respecting different illumination patterns. Reprinted from [10].

Problem	Authors	Year
Logic gates	Tsuda et al. [177]	2004
Robot control	Tsuda et al. [178]	2007
PhyChip: Growing computers from Slime Mould	Adamatzky et al. [1]	2012

Table 1.3.: Various applications of *P. polycephalum* of as a living computing system.

1.4. Motivation and Outline

After surveying the most important experimental facts about *P. polycephalum* and discussing the natural computing strategies inspired by them, we are now in a position to give the motivation of this thesis.

Recall for a moment that some of the most successful natural computing approaches are based on modeling the morphological changes observed in *P. polycephalum*. The Physarum solver for instance replicates the changes in vein thickness as a function of throughput while the multi-agent approach mimics the network topology in a dynamic manner. What these and other approaches have in common is that they fail to capture another distinguishing feature of the slime mold: the periodic reversals of the direction of protoplasmic flow through veins. This feature is highly interesting for several reasons.

From a biophysical point of view it may be assumed that flow reversals are a byproduct of the organisms efforts to keep protoplasmic fluid circulating. An efficient circulation of protoplasm is beneficial, possibly necessary, for *P. polycephalum* to survive since it ensures that nutrients, nuclei and other relevant factors are equally available across the entire individual. Note that such a circulation is naturally maintained despite the dynamically changing and growing underlying network of veins. Again we stress the fact that *P. polycephalum* lacks any form of nervous system or brain. As a result, there is no central control to be made responsible for coordinating the apparently coordinated observed behavior. It is intriguing to ask how this organism manages to organize efficient and robust fluid transport in a fully decentralized manner.

From a computing point of view such properties are non-trivial and highly desirable. What *P. polycephalum* seems to produce and maintain naturally is an (approximate) solution to the problem of distributing resources in a dynamically changing planar graph. This is an interesting and complex transport problem with various conceivable practical applications, particularly in the domain of operations research. As a result, it is desirable to model the behavior of *P. polycephalum* with the goal of developing algorithms for this problem. Ideally such an algorithm would have the following properties:

- The algorithm maintains a dynamic circulation of flow including flow reversals mimicking the flows observed in live *P. polycephalum*. Since resources are transported with the flow, they go wherever the flow reaches.

- The algorithm is robust against changes in topology. Neither natural changes of network topology nor accidental disconnection of veins renders *P. polycephalum* in a state from which it cannot recover. The algorithm should share this quality.

- The algorithm is distributed and requires no central control. As a result, complex global coordination of any sort must emerge from local interactions.

- The algorithm has a degree of efficiency. Based on the assumption that a certain degree of efficiency is necessary for *P. polycephalum* to survive, one may hope that models and algorithms mimicking the organism, inherit this efficiency at least to some extent.

Paving the way towards such an algorithm constitutes the main motivation of this thesis. In it, we seek to contribute to the field of natural computing with *P. polycephalum* through the use of experimental and theoretical methods in an interdisciplinary spirit.

Ultimately, we aim to find a model which yields a circulating flow including flow reversals for individual edges as displayed by *P. polycephalum*. Ideally, this model covers the properties listed above and can be used to derive an efficient algorithm which mimics the way *P. polycephalum* distributes resources all across its network of veins by means of peristaltic pumping.

The past has shown that successful modeling of natural phenomena tends to build upon a reliable body of experimental data and results concerning the same. The reason for this is twofold: First experimental results help build intuition which is crucial in the formulation of models. Second, experimental data can be used after models and/or algorithms have been derived to assess the degree to which they resemble the workings of the natural phenomenon in question. Thus, before engaging in attempts to model the flows observed in *P. polycephalum*, this thesis looks towards experimental work.

In this context, the oscillator experiments introduced in Section 1.3.1 are most promising. They suggests to explore an approach based on the behavior of synchronized coupled oscillators. Experimentally it was shown that dynamic flows, including flow reversals, follow from changing pressure gradients induced by organized thickness oscillations in the plasmodium of *P. polycephalum*. Here the experimenters stress in particular that network properties such as the lengths and the thickness of veins critically influence the observed emergent behavior.

To gain insights in the dynamics of the flow, one may a) look at the flows directly, or b) study the vein networks themselves. Experimentally it is possible to track the flow of protoplasm through single veins of the organism for a certain amount of time. However, obtaining this information for large vein networks including thousands of veins lies outside of our capabilities and is thus infeasible. Alternatively, one may study the topology of the networks on a large scale. This is idea forms the starting point of this thesis.

Thus, we conduct a large number of tailored wet-lab experiments designed to produce images of *P. polycephalum* networks. To evaluate and study the obtained experimental data, we develop a software called Network Extraction From Images or Nefi. It is designed to turn images depicting networks into equivalent graphs. After processing these *P. polycephalum* graphs, we obtain a detailed numerical characterization of their properties searching for clues supporting our modeling efforts.

Naturally, we want all our results, i.e. experimental raw data, processed data

as well as our numerical studies, to serve others the same way the serve us. Thus we set up a dedicate repository, the so-called Slime Mold Graph Repository or SMGR, designed to further the reuse and exchange of *P. polycephalum* related data with a focus on slime mold graphs. All results presented in this thesis are readily available to the natural computing community and everyone who is interested online at smgr.mpi-inf.mpg.de and nefi.mpi-inf.mpg.de.

This thesis documents our studies of the networks formed by *Physarum polycephalum* on the road towards novel distributed natural computing approaches inspired by it. In this thesis, we set out on an ambitious interdisciplinary journey with an uncertain destination.

The remainder of this thesis is structured as a sequence of self-contained chapters supported by relevant appendices. They can be read in order or independently:

In Chapter 2 we introduce NEFI and the problem of extracting networks or graphs from images. Network extraction remains a vital precursor to network analysis and the key to making raw data from wet-lab experiments tractable. Chapter 3 describes the experiments we conducted with live *P. polycephalum*. We discuss the experimental setup as well as the data we obtain. Furthermore we introduce the SMGR, a publicly available data repository intended to promote the distribution and reuse of slime mold related data and results. In Chapter 4 we focus on the actual analysis of the data we obtained earlier. Here we present a systematic study of several graph observables characterizing the vein networks formed by *P. polycephalum*. Finally, in Chapter 5 we present our efforts of modeling the complex flow patterns known to occur in said networks through the use of electric elements. In particular, we study the properties of circuits formed by these elements combining analytic and numerical methods. Furthermore, we discuss the potential of our approach and its suitability for developing natural computing algorithms inspired by the behavior of *P. polycephalum*. We close with a general discussion and concluding remarks regarding the tools, services and results presented. In particular we offer suggestions for potential further research.

2 Network Extraction From Images

Networks are amongst the central building blocks of many systems. Given a graph of a network, methods from graph theory enable a precise investigation of its properties. Software for the analysis of graphs is widely available and has been applied to study various types of networks. In some applications, graph acquisition is relatively simple. However, for many networks data collection relies on images where graph extraction requires domain-specific solutions. In this chapter we introduce Nefi, a tool that extracts graphs from images of networks originating in various domains. Nefi provides a novel platform allowing practitioners to easily extract graphs from images by combining basic tools from image processing, computer vision and graph theory. Thus this novel software constitutes an alternative to tedious manual graph extraction and special purpose tools.

In the context of this thesis Nefi is used exclusively to extract graphs that represent vein networks formed by *P. polycephalum* depicted in images acquired in the wet lab experiments discussed in Chapter 3.

This chapter documents joint work with Dr. A. Neumann and Mag. T. Kehl [52].

2.1. Introduction

The study of complex network-like objects is of increasing importance for multiple scientific domains. The mathematical study of networks, Graph Theory, formalizes a network's structure by modeling the constituents of a network as *vertices* and the pairwise relations between them as *edges*.[1] Networks are ubiquitous in everyday life. Examples are as diverse as the Internet, social networks, transportation networks, metabolic networks, blood vessels or the vein networks of leaves. For a comprehensive review see [127].

In situations where the extraction of a mathematical graph from a physical network is easy, the size of graphs that can be analyzed may quickly increase from hundreds to millions of vertices. In such cases it is possible to build large databases of networks which can be explored automatically using software relying on methods from statistics and graph theory. The combination of a large number of available graphs with dedicated methods of analyzing them yielded many results that changed our understanding of large scale network structures. Unfortunately, digitization

[1] Some communities traditionally refer to vertices as nodes or sites and to edges as arcs or links.

of networks is not always easy and remains difficult for many types of networks. Examples include e.g. leaf venations, blood vessels or food webs, and therefore ready-to-analyze datasets are often not available. In these cases, investigation on a larger scale requires tedious and sometimes error prone data acquisition.

In many experimental settings networks are initially available as high quality images obtained under laboratory control. Before any analysis can take place, it is necessary to extract the associated graphs from these images. This requires the identification of vertices and edges within the depicted structure. This process can quickly become very work-intensive even for smaller networks, which makes automated solutions indispensable.

Leveraging advances in computer vision, several authors have proposed and successfully implemented solutions for domain specific graph extraction applications. The authors of [131], [132] consider the mycelial networks of *P. impudicus*, a member of the fungi. They use watershed segmentation in combination with a novel enhancement step designed to highlight curvilinear features in the input networks. Based on the segmented image a skeleton is computed and used to extract the graph representing the input network. The resulting method is designed to be brightness and contrast invariant in order to correctly extract the networks grown by *P. impudicus* from challenging noisy or low contrast images.

Baumgarten et al. [18], [20] investigate the vein networks of *P. polycephalum* using image processing techniques. For segmenting the input image they rely on careful constant thresholding followed by a sequence of restoration algorithms designed to remove artifacts introduced during thresholding. Next, the restored segmented image is used to compute a skeleton. After applying another sequence of correction steps, the skeleton is scanned to extract the graph of the input network.

In [44] a more general algorithm applicable to a variety of problems is proposed. Based on an original stochastic model, the authors use Monte Carlo sampling to obtain junction-points in the input image. This technically involved solution guarantees structural coherence for the resulting graph representation. Further examples include the extraction of road networks [126], retinal blood vessel analysis [103] and the extraction of plane graphs [149].

The three above mentioned algorithmic solutions for the network extraction problem exhibit one or more of the following limitations:

- They do not build on top of well-established computer vision methods and tend to rely on ad-hoc algorithms. As a result the quality of the method and its implementation could likely be improved. In addition, a lot of time is spent on reimplementing algorithms that are already available in common libraries.

- They are not implemented at all or only available on paper as pseudo-code.

- They are implemented but it is clear that the authors never intended for anyone else to use their code. Resulting implementations tend to neglect ease of use, distribution or issues of extendability.

Naturally, we need to keep in mind that the primary objective of the work cited above is not the production of reusable software, but of algorithms and tools for solving a concrete research question at hand. The aim is to get the job done. As a result, the time for researching computer vision libraries and implementations, following best software engineering practice or writing documentation is limited.

From experience we know that when trying to produce an easy-to-use software, a large part of the required work consists of specifying and improving the user-interface as well as working out minor bugs and annoyances. This type of work, while very time consuming, is essential for any software aiming to reach a non-negligible audience. However, efforts like these are hardly attractive to researchers whose focus is on obtaining the next result. While we understand that under these circumstances the aforementioned limitations arise naturally, we strongly believe that it is necessary to overcome those limitations in order to increase the value and the impact of scientific software in general and network extraction software in particular. It is possible to do better.

To this end, we introduce Nefi, a lightweight piece of ready-to-go software intended to enable the non-expert to automatically extract networks from images. Nefi constitutes an extensible framework of interchangeable algorithms accessible through an intuitive graphical user interface.

We emphasize at this point that we do not claim to introduce novel techniques for image processing or computer vision. Instead, our contribution consists of a reusable, flexible and easily extendable toolbox combining well-known methods, which have become standard in their respective fields of origin, in a meaningful way. By introducing Nefi, we hope to make these methods more widely accessible, especially to practitioners that are unfamiliar with them.

Nefi's segmentation is based on a combination of standard routines available in OpenCV [37]. These algorithms are known to perform well on clean and uncluttered images obtained under controlled laboratory conditions. However, on more challenging inputs of low contrast, strong gradients or similar irregularities, their performance is severely reduced. Nevertheless, in these cases more involved algorithms, currently not implemented as part of a reliable library and thus not integrated into Nefi, may still be able to process these images. To help meet this situation, Nefi was designed with extendability in mind. As a result users will hopefully find it easy to build on-top of Nefi's code in order to add their own implementations of more sophisticated methods.

2.2. Network Extraction From Images

Nefi features a collection of image processing routines, segmentation methods and graph algorithms designed to process 2D digital images of various networks and network-like structures. Its main function is executing a so-called *extraction pipeline*, designed to analyze the structures depicted in the input image. An extraction pipeline, for short pipeline, denotes an ordered sequence of algorithms. A successful

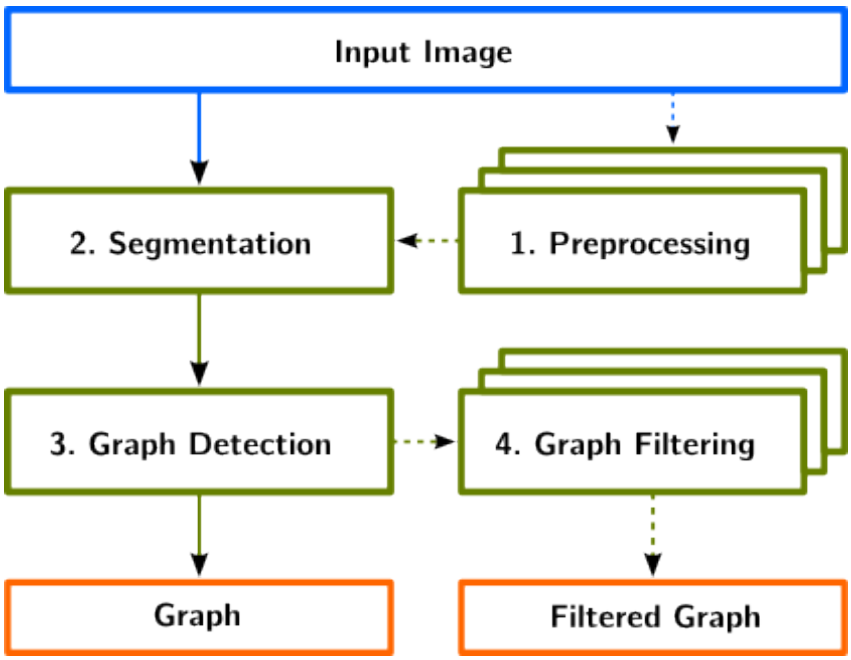


Figure 2.1.: A flow chart illustrating Nefi's pipeline components in green boxes. Dashed arrows depict optional sections of the pipeline. Blue and orange boxes denote Nefi's input and possible outputs respectively.

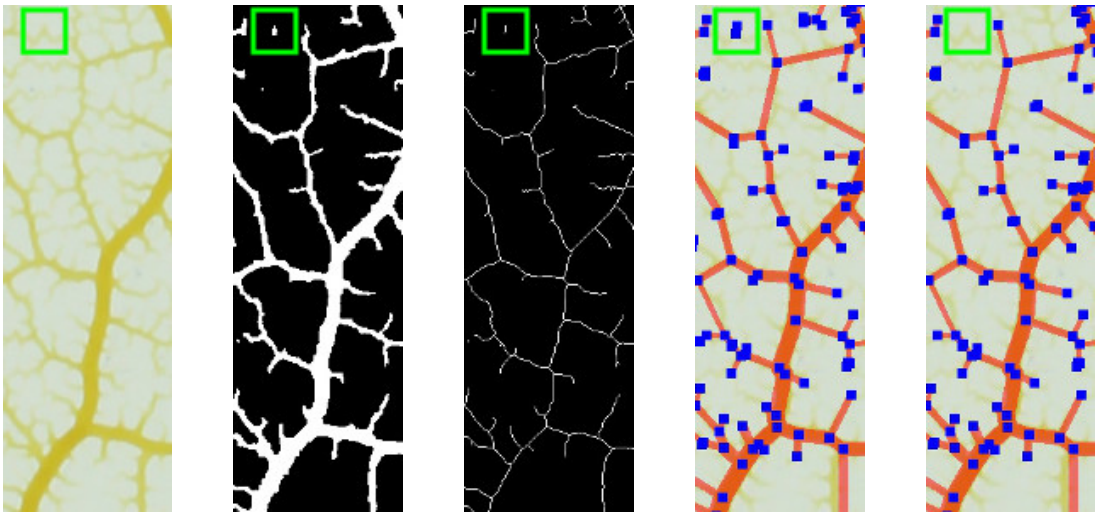
execution will return a representation of the network in terms of an edge-weighted undirected planar graph. Computed weights include edge lengths and edge widths. Once the graph is obtained, available graph analysis software [16], [17], [74], [106], [107], [185] or custom written scripts can be deployed to investigate its properties.

A typical pipeline combines algorithms from up to four different classes: preprocessing, segmentation, graph detection and graph filtering, see Figure 2.1. A more detailed description follows below.

For each pipeline section, Nefi typically offers several interchangeable algorithms to choose from. After executing preprocessing routines, a segmentation algorithm separates foreground from background. Then the foreground is thinned to a skeleton from which the vertices and edges of the graph are determined. In the process various edge weights are computed. Finally, the graph can be subjected to a variety of useful graph filters. Figure 2.2 illustrates the intermediate results of Nefi's pipeline steps listed in the order of their execution. When a pipeline is executed, Nefi makes all intermediate results available via its clean and intuitive graphical user interface, see Figure 2.3.

Via the graphical user interface all basic functions of Nefi can be accessed in an intuitive fashion. To facilitate ease-of-use, most of Nefi's algorithms come with default parameters based on settings in OpenCV [37], which were found to perform well on our test sets as well as on many other images.

There are various predefined pipelines to get started immediately. Alternatively, users may freely combine the various methods to build custom pipelines. Both



(a) (b) (c) (d) (e)

Figure 2.2.: Direct comparison of NEFI's pipeline steps given a slice of an image depicting veins of a slime mold (*P. polycephalum*). (a) Input image, (b) segmented image, (c) skeletonized image, (d) detected graph and (e) filtered graph. The green square contains a very faint vein which the segmentation did not pick up fully. As a result, the skeleton became fragmented which leading to spurious vertices in the detected graph. By applying a graph filter we remove stray vertices without manipulation of the segmented or the skeletonized image. Similar filtering can remove "dead-ends", i.e. vertices that do not belong to any cycle in the graph.

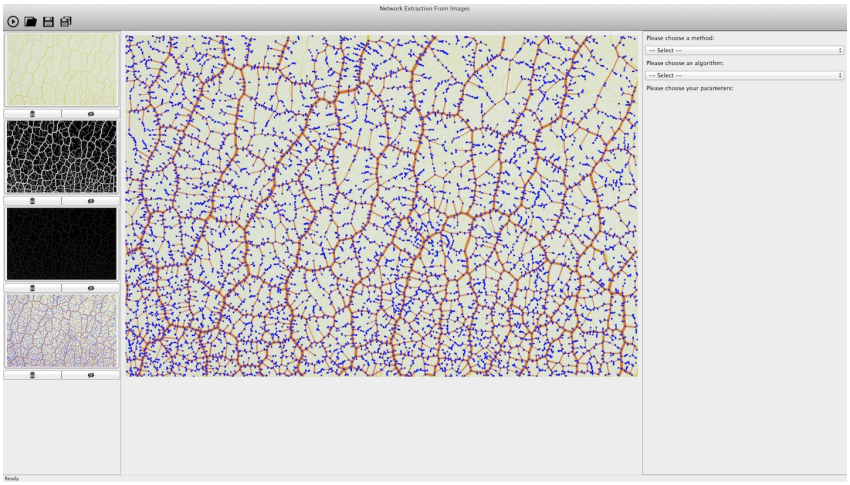
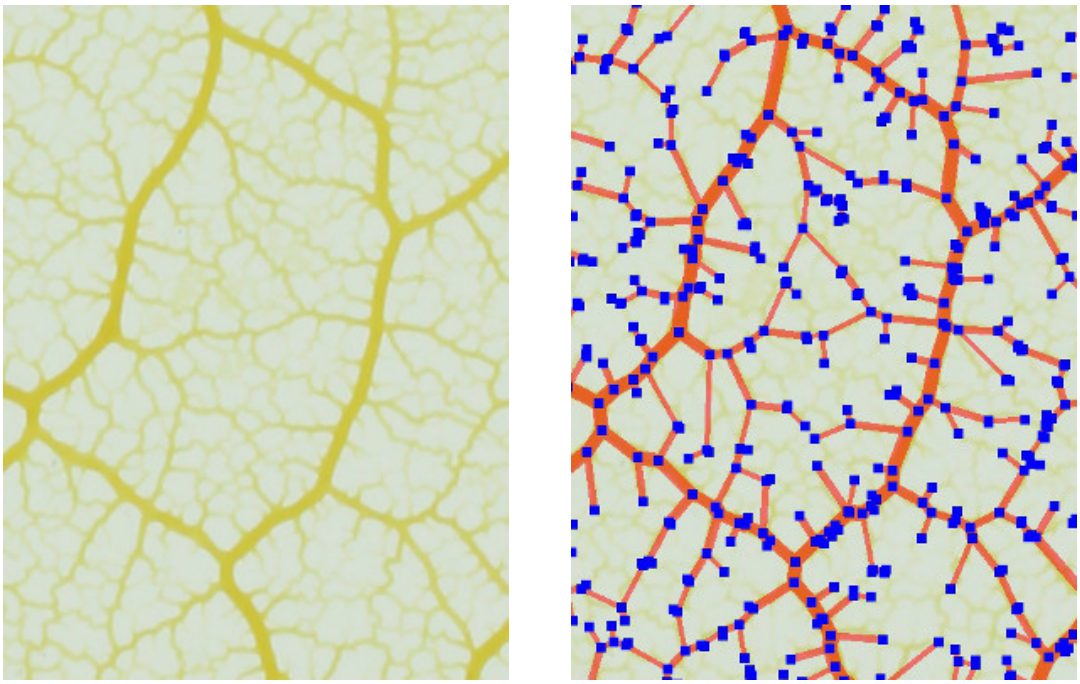


Figure 2.3.: A screenshot of NEFI's GUI running on Mac OS. On the left hand side NEFI lists intermediate results as thumbnails. Bringing the final result to the center workspace allows for direct visual assessment of the quality of the extracted graph. On the right hand side NEFI's pipeline elements can be accessed via convenient drop-down menus.



(a) (b)

Figure 2.4.: Extracted graph of a vein network formed by *P. polycephalum*. (a) Input image depicting the network. (b) The extracted graph drawn on top off the input image for direct comparison. Note, that no filters have been applied. As a result, there are quite a few clumped up nodes and dead-end edges.

approaches allow the user to experiment with the available methods in order to close in on the optimal settings for the data at hand. Once a pipeline is constructed, it can be saved and reused. Nefi's simple pipeline concept together with a self-explanatory graphical user interface make working with Nefi intuitive and straightforward. Nefi also offers a command-line mode, which is suited for batch processing large quantities of input images.

Nefi comes with a number of example images from different domains which we use to produce the figures presented in this chapter. Figure 2.4 and Figure 2.5 show Nefi's output on two images using predefined pipelines. Blue squares denote the vertices and red lines the edges of the detected graph. The thickness of the detected edges corresponds to thickness of the depicted structures. For comparison the extracted graph is drawn on top of the input image. We present a detailed quantitative evaluation in Section 2.3.

We stress that Nefi can deal with a range of inputs from various domains as long as they are of sufficient quality. In addition to the examples shown above, it has been successfully used to process images of natural (e.g. leaf venation, patterns of mud cracks) as well as man-made structures (e.g. tilings). It is also straightforward to add custom extensions. We provide a well documented platform which allows

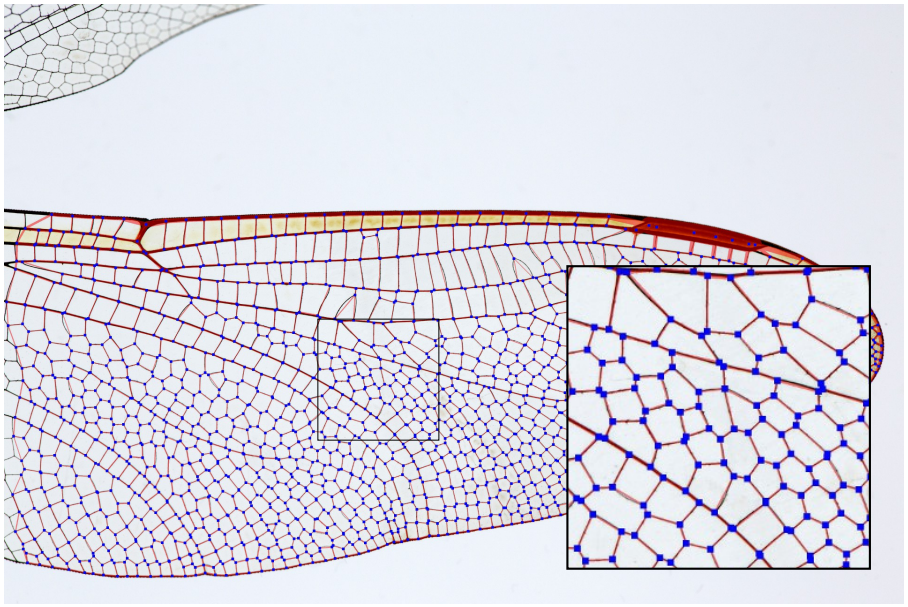


Figure 2.5.: Extracted graph of the vein network exhibited by a wing of a dragonfly (*Ajax junius*). Note, that after the use of various filters an extraordinarily clean graph is obtained. Image courtesy of Pam and Richard Winegar.

programmers to include more specialized segmentation algorithms or additional graph filters. For an overview of alternative graph extraction approaches see for example [49].

Next, we discuss the purpose and design of each major stage of the pipeline and highlight some of NEFI's strong points.

2.2.1. Preprocessing Collection

The preprocessing section of the pipeline offers various standard image processing algorithms intended to be used prior to the segmentation step. Preprocessing methods may be exploited to positively affect the output of the segmentation step. For example, adding a slight blur to an input image may benefit the overall result by reducing the amount of spurious white pixels appearing in the segmented image. However, blurring too much will remove detail and reduce accuracy in determining the thickness of depicted edges. As a result, we recommend to experiment with different approaches and parameter settings in order to decide how to use preprocessing. For images of sufficient quality we found that excellent results can be obtained without preprocessing.

NEFI relies on OpenCV [37] for preprocessing and offers Gaussian and Median Blurring, Denoising as well as Bilateral Filtering.

2.2.2. Segmentation Collection

The goal of the segmentation step is separating the image foreground, i.e. the structures of interest, from the remaining image. Nefi builds on top of OpenCV [37] combining different segmentation algorithms. The algorithms shipped with Nefi are multi purpose and have become standard in image processing. They are very reliable as long as input images are clean, devoid of strong gradients and have good contrast between fore- and background. Conversely, if the input becomes more challenging, the effectiveness of Nefi's segmentation degrades quickly and more complex or domain-specific algorithms become necessary. We defer a quantitative study of how the properties of the input image affect Nefi's performance to Section 2.3.

Nefi's segmentation is designed such that several algorithms can be used interchangeably. We included basic thresholding algorithms like Otsu's method [136], or adaptive thresholding as well as more involved segmentation routines such as guided watershed [111], and the GrabCut algorithm [146]. The last two methods receive as an additional input a so-called marker. The better the markers approximate the foreground, the better these algorithms work. Nefi offers several marker strategies which can be used interchangeably together with the respective marker based segmentation routines.

Interchangeability of the algorithms is a core design principle of all pipeline steps. This design facilitates swift experimentation with different methods. Our own experience shows that *a priori* it is often unclear which method will work best for a given input image. A sensitive method may yield excellent segmented images and very detailed graphs. However its sensitivity increases the chance of false positives for both nodes and edges. Since the ideal choice usually depends on several competing factors, targeted experimentation with different settings coupled with immediate visual feedback from the graphical user interface constitute major strong points of Nefi.

Flexibility is not limited to the algorithms that come with Nefi. Modular software design makes it easy to integrate additional methods and access them via the graphical user interface. It is likely that sooner or later input images will be encountered in practice, too challenging for the core algorithms presently available in Nefi. In these cases a potential user may choose to implement additional, perhaps domain-specific, methods capable of meeting the challenge at hand. When extending Nefi, users may build on existing modules, yielding more reliable code which is easier to understand and to maintain. In this way Nefi is set up to support future development and growth.

2.2.3. Graph Detection Collection

The graph detection collection consists of algorithms that take a segmented image as input and determine the nodes and the edges of the output graph. We offer a colloquial description of the actual algorithm because we do not rely on well-documented library code for this section of the pipeline.

The first step for graph detection is called thinning. Here we reduce the segmented foreground such that every line is only one pixel thick, while preserving the connectivity properties of both the foreground and the background pixels. The result of this process is called the skeleton of the segmented image. To do so we implemented the algorithm by Guo and Hall [72]. It always produces thin results and preserves 8-connectivity of the foreground pixels. The latter is essential for preventing all sorts of erroneous edges and spurious nodes. For this reason we choose Guo-Hall thinning over other available thinning algorithms. A pure Python implementation proved to be fairly slow, hence we chose to implement this function as a C extension.

For fairly thin and network-like foreground structures this method is nearly flawless and finds a skeleton where the lines lie in the center of the foreground areas. However, large foreground structures that do not resemble networks lead to artifacts in the skeleton whose exact shape depends on the noise present at their borders.

Once the skeleton is established, we use it to detect the positions of nodes. For this purpose we adapt criteria from a thinning algorithm by Zhang and Suen [186]. A white skeleton pixel becomes a node in the output graph if its removal creates exactly one or at least three 4-connected white components in its 1-neighborhood. In the former case the pixel forms the end of a path, i.e. a node of degree $d = 1$, in the latter case is the meeting point of at least three edges, i.e. a node of degree $d \geq 3$.

Note that due to this step, the maximum degree of the graphs we detect is limited to four. This is inevitable if nodes are detected at single-pixel locations. For higher degree nodes we will create several nodes of smaller degree that are very close to each other. In this way it is possible to establish the correct large node degree by merging the smaller degree nodes in a later post-processing step.

Given the node positions, it is very simple to find the edges by establishing the paths of pixels between them. We perform a variant of breadth first search on the white pixels in the skeleton, starting from each node simultaneously. Each white pixel around a node gets a unique number and a queue. In each step we iterate over all queues and take out the first pixel. If it is unmarked, we mark it with the unique number of this queue and enqueue all its white neighbors. Otherwise, we have detected an edge, i.e. there is a path along white pixels connecting two distinct nodes.

While walking along the pixels we record the length of the edge in units of pixel. Horizontal and vertical steps count as one unit, diagonal steps count as $\sqrt{2} \approx 1.41$ units.

The diameter of an edge is calculated by computing the distance transform of the segmented image. Next, we assume that the path of skeleton pixels representing an edge approximately goes through the middle of the real edge seen in the input image. Under this assumption, which we found to hold for all our test images, computing the diameters becomes a simple lookup of each edge-pixel from the skeleton in the distance transformed image. Since in doing so, we obtain the diameter for every pixel along an edge, we can easily compute statistical quantities over the recorded diameter values.

For handling the graph in terms of data-structures, we rely on NetworkX [74].

2.2.4. Graph Filter Collection

The graph filter collection offers the possibility to add powerful processing steps that directly apply to the graph obtained after graph detection. Unlike the methods in the preprocessing collection, graph filters do not affect segmentation and graph detection.

Often it is possible to improve the quality of the obtained graph by removing unwanted artifacts caused by segmentation or later processing steps. A common strategy, used for example in [18], [20], consists of "repairing" the errors present in the skeleton due to bad segmentation using heuristics or user assisted methods. While such methods can do well, they carry the potential risk of introducing additional errors.

NEFI pursues a novel approach exploiting knowledge of the structure of the extracted graph together with dedicated filtering methods. We start by using very sensitive segmentation retaining a maximum of structural information. We do not manipulate the resulting segmented image or the skeleton at all. Based on the skeleton, we establish a graph which in general will contain artifacts. For example, if the network in the input image is large and connected, the resulting graph will consist of a large connected component plus a number of spurious smaller components caused by noise and very sensitive segmentation. Such components can easily be spotted by eye when one compares the original images with the output graph. Using one of NEFI's graph filters they can be removed effectively and safely, increasing the degree of similarity between the original network and the extracted graph. Since the effects of filtering the graph can immediately be evaluated by visual inspection, we prefer graph filtering over less transparent approaches aiming to improve the graph before it was established. Figure 2.2 illustrates the use of filtering.

We have used filtering with sensitive segmentation to obtain surprisingly good results. Overly sensitive segmentation picks up fine detail but is also prone to create artifacts due to noise. However, almost all of these artifacts are benign in the sense that they lead to very small components which can easily be removed by filtering. The desired detail will remain mostly unaffected because it is part of the largest component. The graph depicted in Figure 2.5 was obtained using this technique which demonstrates the effectiveness of filtering given good input.

Filtering in general may be used to remove parts of the graph which are not of interest. The following filters are predefined in NEFI. A filter removing everything not in the largest connected component, one smoothing vertices of degree provided no parallel edges are created and finally, a filter which removes all vertices and edges that are not contained in a cycle. Filters may be freely combined with each other in any order to achieve various results. Naturally, the filter collection is designed for extendability which means that users may add their own filters to the existing collection.

Graph filtering and its various applications delivers excellent results. To our knowledge, no other software offers such flexible tools as part of its core work flow.

2.3. Evaluation

To assess NEFI's performance we quantify the quality of the resulting output graphs as a function of input images of varying difficulty. To this end we define a quality measure that captures the degree of similarity between the graphs computed by NEFI and the networks depicted in the input. While to a certain degree said similarity can be determined for a small set of instances by visual inspection, a dedicated measure allows us to systematically and reproducibly investigate whether NEFI's graph extraction is reliable. As an additional metric of interest, we report NEFI's speed which becomes relevant as soon as one starts to process large batches of input images.

2.3.1. Using a Graph Similarity Measure to Evaluate NEFI

Our quality measure needs to quantify the degree of congruence between the graph depicted in the original input image i and the graph computed by NEFI.

Let $A = (V_A, E_A)$ be the *true* graph correctly describing the structure depicted in i, with V_A and E_A denoting its vertex and edge set respectively. We call A the *ground truth*, which is of course not known in general. Furthermore, let B denote the graph obtained by executing one of NEFI's pipelines. Note that, $A, B \in \mathcal{G}$, where $\mathcal{G}$ denotes the set of undirected edge-weighted planar graphs where vertices are labeled with their respective euclidean coordinates in the plane. With these definitions we propose a function s mapping any pair of graphs $A, B \in \mathcal{G}$ onto a number $s \in [0, 1]$. Then, this number s serves as our similarity measure.

To obtain s, we compute a correspondence of vertices in A to vertices in B. Two edges $e \in E_A$ and $f \in E_B$ then correspond if their endpoints correspond. We choose the correspondence such that the following intuitive notions of similarity are optimized.

1. Positions of vertices in V_A are similar to the positions of corresponding vertices in V_B.

2. Edges in E_A including their weights are similar corresponding edges in E_B.

We require that the measure s is maximal if any graph A is compared with itself, that is $s(A, A) = 1$. Consequently, if A is completely different from B we have $s(A, B) = 0$. This minimum value is assumed if no viable correspondence between V_A and V_B can be found. Naturally, the value of $s(A, B)$ increases (decreases) if the similarity between A and B increases (decreases).

An exact definition of s and the notions of *similarity* and *correspondence* is given in the following section. Readers not interested in the details may skip ahead to the results of the evaluation using the measure s given in Section 2.3.3.

2.3.2. Definition of the Similarity Measure

In order to determine the overall quality of NEFI's output we need to quantify the degree of similarity between two graphs $A = (V_A, E_A)$ and $B = (V_B, E_B)$ using a graph similarity measure. Let $A, B \in \mathcal{G}$, where $\mathcal{G}$ is a set of undirected edges-weighted planar graphs where the nodes are labeled with their respective coordinates in the Euclidean plane. Then we may define a similarity measure s as

$$\begin{aligned} s : \mathcal{G} \times \mathcal{G} &\mapsto [0, 1] \,, \\ s(A, B) = \text{matching}(A, B) \qquad &\forall\ A, B \in \mathcal{G} \,, \end{aligned} \tag{2.1}$$

where matching(A, B) denotes the normalized cost of a minimum cost graph matching problem which we will define shortly. Given this definition the degree of similarity between A and B is quantified by the value of $s(A, B)$. The similarity measure we propose has the following desirable properties:

- A compared to itself yields a maximum similarity score of $s(A, A) = 1$.
- If there are no vertices in V_A with corresponding similar nodes in V_B and no edges E_A with corresponding similar edges in E_B the minimum similarity score of $s(A, B) = 0$ is obtained.
- The similarity score $s(A, B)$ increases (decreases) if the similarity between A and B increases (decreases).

We proceed with making the notions of *similarity* and *correspondence* more precise by constructing a minimum cost graph matching problem. Given the two graphs $A, B \in \mathcal{G}$, let n_A, n_B denote the number of nodes and m_A, m_B denote the number edges of the respective graphs. Note that n_A and n_B can differ, as can m_A and m_B.

We can now define a matching between A and B consisting of three parts. First we define a matching between the node sets V_A, V_B, then we define a matching between the edge sets E_A, E_B and finally we introduce a coupling between the two. To obtain a solution to the final problem we rephrase the combined matching problem as an integer linear program and obtain a solution using IBM's CPLEX solver [46]. For ease of exposition, however, we discuss the problem using the language of matchings.

For the node matching, we match each node $i \in V_A$ with at most one node $j \in V_B$. We define a variable x_{ij} that is 1 if we match i with j and 0 otherwise. With a match we associate a cost of $\Delta_V(i, j)$. The function $\Delta_V(i, j)$ returns the euclidean distance between node i and node j. If any node u remains unmatched, we assign a penalty of $p_V(u)$ to it. A node can either be penalized or matched but not both. We thus compute a minimum cost node matching, in which it is favorable for a node $i \in V_A$ to find a match with a node $j \in V_B$ such that i and j are close in distance as measured by $\Delta_V(i, j)$. Two nodes are thus *similar* if their euclidean positions are almost the same. For any given node in V_A there can be many similar nodes in V_B,

all of which are candidates for a match. Amongst all candidates the minimum cost matching selects pairs that are most similar. We call selected pairs of similar nodes *corresponding*.

The edge matching proceeds analogously to the node matching. Each edge $e \in E_A$ is matched with at most one edge $f \in E_B$. We denote this match with a decision variable y_{ef} and associate a cost of $\Delta_E(e, f)$ with it. The function $\Delta_E(e, f)$ returns the sum of the differences in edge weights between edge e and edge f. If any edge d remains unmatched, we assign a penalty of $p_E(d)$ to it. An edge can either be penalized or matched but not both. We thus compute a minimum cost edge matching, in which it is favorable for an edge $e \in E_A$ to find a match with an edge in $f \in E_B$ such that e and f are close in weights as measured by $\Delta_E(e, f)$. Two edges are thus *similar* if their weights are almost the same. For any given edge in E_A there can be many similar edges in E_B, all of which are candidates for a match. Amongst all candidates the minimum cost matching selects pairs that are most similar. We call matching pairs of edges *corresponding*.

As of yet both matchings are independent of each other. In particular this allows an edge $e \in E_A$ to be matched with an edge $f \in E_B$ independently from their respective positions in the plane as long as they have similar weights! A more meaningful matching favors pairing edges where the endpoints of both edges are also similar to each other. Hence, start and end nodes of corresponding edges should be pairwise corresponding themselves. In other words, head and tail nodes of both edges are geographically close respectively.

This additional constraint suggests a dependence between node and edge matching. To enforce it we add the constraint that an edge $e = (a, b) \in E_A$ can only be matched to an edge $f = (a', b')$ if the respective nodes are pairwise matched as well. That is, we require a to be matched to a' and b to be matched to b' or alternatively a to be matched to b' and b to be matched to a'.

By combining node matching, edge matching and the additional coupling constraint we obtain the final minimum cost matching problem. We rephrase the uncoupled matching problem as an integer linear program (ILP) as follows:

$$\begin{aligned}\textbf{Minimize:}\quad f = &\sum_{\substack{i\in V_A\\ j\in V_B}} \Delta_V(i,j)x_{ij} \quad + \sum_{i\in V_A} p_V(i)\overline{x}_i + \sum_{j\in V_B} p_V(j)\overline{x}_j \; + \\ &\sum_{\substack{e\in E_A\\ f\in E_B}} \Delta_E(e,f)y_{ef} + \sum_{e\in E_A} p_E(e)\overline{y}_e + \sum_{f\in E_B} p_E(f)\overline{y}_f \qquad (2.2)\end{aligned}$$

$$\begin{aligned}\textbf{s.t.:}\qquad & x_{ij} \in \{0,1\} \qquad i \in V_A \qquad j \in V_B \qquad (2.3)\\ & y_{ef} \in \{0,1\} \qquad e \in E_A \qquad f \in E_B \qquad (2.4)\end{aligned}$$

$$\textbf{s.t.:} \qquad \forall\, i \sum_j x_{ij} \leq 1 \qquad \forall\, j \sum_i x_{ij} \leq 1 \tag{2.5}$$

$$\forall\, e \sum_f y_{ef} \leq 1 \qquad \forall\, f \sum_e y_{ef} \leq 1 \tag{2.6}$$

$$\textbf{s.t.:} \qquad \overline{x}_i = 1 - \sum_{j \in V_B} x_{ij} \qquad \overline{x}_j = 1 - \sum_{i \in V_A} x_{ij} \tag{2.7}$$

$$\overline{y}_e = 1 - \sum_{f \in E_B} y_{ef} \qquad \overline{y}_f = 1 - \sum_{e \in E_A} y_{ef}\,. \tag{2.8}$$

To introduce the dependence between the vertex and the edge matching, we add an additional constraint as described earlier:

$$\textbf{s.t.:} \qquad 2y_{ef} \leq x_{aa'} + x_{ab'} + x_{ba'} + x_{bb'} \qquad \forall\, e = (a,b), \forall\, f = (a',b')\,. \tag{2.9}$$

From an ILP solution the optimal matching for nodes and edges can be recovered. Thus, we know what matching pairs have been formed and which nodes and edges remained unmatched. We use this information to define true positives, false positives and false negatives as described in the next section.

In addition to that, we obtain the optimal value of the objective function OPT, i.e. the cost associated with the selected optimal matching. The value of OPT includes both the costs incurred by node and edge matches as well as the penalty terms arising from nodes and edges that cannot be matched. Thus, it is a measure of the similarity between the two graphs A and B.

We point out that $\text{OPT} = 0$ if a graph G is matched with itself. Every single node and edge finds its exact copy as a match of cost 0 for itself and thus no penalties arise. The other extreme is realized when there are no vertices in V_A with corresponding similar nodes in V_B and no edges E_A with corresponding similar edges in E_B. In this case the value of OPT attains its maximum $\overline{\text{OPT}}$ because all nodes and edges of both graphs are penalized. Naturally, the value of OPT increases (decreases) with decreasing (increasing) graph similarity. Given these observations it is natural to use OPT and $\overline{\text{OPT}}$ to define a similarity measure betweens two graphs $A, B \in \mathcal{G}$ as

$$s(A,B) = \text{matching}(A,B) = 1 - \frac{\text{OPT}(A,B)}{\overline{\text{OPT}}(A,B)}\,, \tag{2.10}$$

with

$$\overline{\text{OPT}}(A,B) = \sum_{i \in V_A} p_V(i) + \sum_{j \in V_B} p_V(j) + \sum_{e \in E_A} p_E(e) + \sum_{f \in E_B} p_E(f)\,. \tag{2.11}$$

As a practical remark, we note that the number of variables entering the ILP grows like $\mathcal{O}(n_A n_B + m_A m_B)$. As a result, it may become prohibitively large even for graphs of moderate size. To deal with this issue, it is convenient to define a circle with search radius r centered around each node i. We may exclusively consider nodes j found within this search radius as possible matches for node i. By doing so, we dismiss all pairings for i for which we know *a priori* that they cannot be part of the optimal solution. They are simply too expensive. Consequently, we may safely omit the corresponding LP variables x_{ij} from the ILP, thus reducing its size considerably.

Correctly enforcing this idea, requires $r \geq p_V(i) + p_V(j)$. To see this, assume that we have a node i in distance d of node j. If $d = \Delta_V(i,j) < p_V(i) + p_V(j)$ then the ILP matches them. However, to do so it needs to see this match as a possibility and thus $r \geq p_V(i) + p_V(j)$.

Note that, this reasoning conveniently ignores the fact that node matches enable or disable a number of associated potential edge matches. These may come with additional penalties affecting the value of r. In practice, we may circumvent this issue by choosing r generously in relation to the penalties.

The number of variables y_{ef} referring to edge pairs can be reduced by separately applying the previous argument to each of the nodes involved. We enumerate the set of all possible edges $f = (a', b')$ to be matched with edge $e = (a, b)$. The set of edges f is constructed by connecting all nodes a' within a radius r around the node a with all nodes b' within a radius r around node b. This process can create self-loops which we exclude as candidates. If there are no nodes within a distance r from a or b, the associated edge variables y_{ef} can be omitted from the ILP.

Choosing the search radius and adjusting the penalties accordingly in order to control the size of the integer linear program seems more natural to the authors than the other way around. This approach has the additional advantage that a sensible value for the search radius can often be inferred by the looking at the length scales of the structures of interest in the original input image.

As a concluding remark, we stress that the choice of penalties for nodes and edges affects the final value of the similarity measure $s(A, B)$ via the normalization. As a result, when using this measure to compare several graphs with each other, the penalties and tracking radius r must be chosen consistently in order to guarantee the viability of derived comparisons.

2.3.3. Evaluation of NEFI's Output

We proceed with the evaluation of Nefi's output using the above similarity measure. To do so we create a set $\mathcal{A}$ of artificial ground truth graphs such that $\mathcal{A} \subset \mathcal{G}$. We obtain $\mathcal{A}$ by simply turning a set I_0 containing 250 images of the slime mold *P. polycephalum* into an equivalent set of graphs using Nefi.

Next, we turn the ground truth graphs in $\mathcal{A}$ into a test set I_1 of 2D images by simply drawing them. The drawing preserves the euclidean positions of the nodes, the edge lengths and the thickness of the edges. As a result, the image $i \in I_1$ depicts

Method	Similarity score	Sensitivity	Precision
Otsu's method	0.984 ± 0.005	0.970 ± 0.011	0.998 ± 0.001
Adaptive threshold	0.984 ± 0.005	0.970 ± 0.011	0.997 ± 0.001
Watershed (deletion/erosion)	0.980 ± 0.006	0.959 ± 0.013	0.988 ± 0.001
Watershed (distance transform)	0.906 ± 0.121	0.837 ± 0.160	0.998 ± 0.001
Watershed (adaptive)	0.977 ± 0.008	0.956 ± 0.016	0.998 ± 0.001
Grabcut (deletion/erosion)	0.984 ± 0.005	0.970 ± 0.011	0.998 ± 0.001
Grabcut (distance transform)	0.983 ± 0.005	0.967 ± 0.011	0.998 ± 0.001

Table 2.1.: Results for ideal test images I_1.

the graph $A_i \in \mathcal{A}$. In other words, we know the *ground truth* A_i for every image i in the test set I_1.

To compare different segmentation methods, we prepare a set $\mathcal{P}$ of pipelines differing only in the segmentation algorithms used. The parameters of the pipeline where chosen manually for each test set using experimentation and visual inspection.

Given the sets I_1 and $\mathcal{A}$ as well as our similarity measure we can now evaluate NEFI's output. We take an image $i \in I_1$ and process it with a given pipeline $p \in \mathcal{P}$ to obtain a graph $B_i \in \mathcal{B}_1$, i.e. the graph NEFI extracted from the input image. Then we compute the similarity score $s(A_i, B_i)$. To obtain statistical statements, we repeat this procedure for all images and all pipelines.

During the computation of $s(A_i, B_i)$, we record features NEFI failed to detect in i, namely the number of vertices (edges) in A_i which remain without corresponding vertices (edges) in B_i. This is the number of false negatives (FN). Furthermore, we record the number of vertices (edges) in B for which no corresponding vertices (edges) exist in A_i. These are features which NEFI detects in i but which are in fact not present. These are false positives (FP). Finally we record the number of vertices (edges) in B_i that have corresponding elements in A_i. That is, features correctly extracted from the image i, which we count as true positives (TP). Unfortunately, the number of true negatives (TN) is not accessible in a similar fashion. For this reason we restrict ourselves to computing sensitivity ($\frac{TP}{TP+FN}$) and precision ($\frac{TP}{TP+FP}$) in the results of the evaluation. Sensitivity and precision reported in the following Tables combine the respective values for vertices and edges.

Table 2.1 summarizes the results of processing the set I_1. The images in I_1 are ideal inputs for NEFI for which all our segmentation routines produce very good results. Otsu's method and adaptive thresholding yield perfect segmentations. Hence, any difference between NEFI's output and the ground truth cannot originate in the segmentation part of the pipeline but must be attributed to thinning and graph detection. The excellent correspondence between NEFI's output and the ground truth confirms that thinning and graph detection are very reliable. It can be seen that the obtained similarity scores are very close to optimal.

One might question the validity of using graphs which were detected by NEFI as the input set for evaluating NEFI itself. We reply that the approach is valid because

Method	Similarity score	Sensitivity	Precision
Otsu's method	0.868 ± 0.018	0.704 ± 0.028	0.987 ± 0.005
Adaptive threshold	0.941 ± 0.010	0.853 ± 0.034	0.976 ± 0.025
Watershed (deletion/erosion)	0.859 ± 0.018	0.693 ± 0.028	0.984 ± 0.006
Watershed (distance transform)	0.408 ± 0.176	0.239 ± 0.154	0.987 ± 0.007
Watershed (adaptive)	0.966 ± 0.008	0.936 ± 0.016	0.984 ± 0.017
Grabcut (deletion/erosion)	0.864 ± 0.019	0.696 ± 0.029	0.986 ± 0.005
Grabcut (distance transform)	0.858 ± 0.020	0.688 ± 0.030	0.986 ± 0.005

Table 2.2.: Results for images I_2 with edges drawn with random brightness.

the origin of the images in I_1 has no significance regarding Nefi's performance. In other words, they are just as hard or as easy to process as images obtained in any other way. However, note that the perfect images in I_1 do not represent real life input very well. Therefore we produced three more test sets and evaluate them as described above.

For the set I_2 we take the images in I_1 and change the brightness of the edge drawings randomly. As a result the local contrast between foreground and background varies widely across the image. To create set I_3 we take the images in I_1 and insert a color gradient into the background while leaving the foreground unchanged. Set I_4 is obtained by taking the images in I_1 and subjecting them to a global blur.

Table 2.2 summarizes the results of processing the set I_2. We observe that both similarity score as well sensitivity are deteriorating for almost all methods except for adaptive thresholding and watershed based on adaptive thresholding. Adaptive thresholding is still able to compensate the local changes in brightness present in the test images and returns segmented images of high quality. We stress that for images showing more severe irregularities the performance of these methods is expected to suffer. Note that the precision remains comparably high still. This indicates that the vertices and edges that are being detected by Nefi are indeed part of the ground truth.

Table 2.3 summarizes the results of processing the set I_3. We observe that almost all methods, with the exception of adaptive thresholding and watershed based on adaptive thresholding perform very poorly. In particular watershed based on a distance transform marker is completely unable to handle the input images.

Table 2.4 summarizes the results of processing the set I_4. We observe that almost all methods, with the exception of watershed with distance transform, perform reasonably well. Sensitivity and similarity scores are slightly smaller than the results obtained for the optimal test images I_1. This is due to the fact that blurring an image i causes the depicted edges to appear slightly wider. This increase in width translates to the edge weights of the graphs B_i. Since differences in edge weights are penalized, the similarity score $s(A_i, B_i)$ decreases accordingly.

Summarizing the results we conclude that the quality of a graph detected by Nefi depends on the input image and the selected pipeline. We have established

Method	Similarity score	Sensitivity	Precision
Otsu's method	0.737 ± 0.060	0.602 ± 0.065	0.911 ± 0.065
Adaptive threshold	0.984 ± 0.005	0.970 ± 0.011	0.998 ± 0.001
Watershed (deletion/erosion)	0.752 ± 0.047	0.588 ± 0.061	0.977 ± 0.009
Watershed (distance transform)	0.334 ± 0.303	0.240 ± 0.261	0.943 ± 0.043
Watershed (adaptive)	0.982 ± 0.005	0.967 ± 0.012	0.997 ± 0.001
Grabcut (deletion/erosion)	0.733 ± 0.053	0.573 ± 0.066	0.987 ± 0.005
Grabcut (distance transform)	0.742 ± 0.052	0.582 ± 0.065	0.983 ± 0.008

Table 2.3.: Results for images I_3 with a color gradient in the background.

Method	Similarity score	Sensitivity	Precision
Otsu's method	0.953 ± 0.011	0.909 ± 0.022	0.993 ± 0.003
Adaptive threshold	0.947 ± 0.010	0.863 ± 0.028	0.989 ± 0.005
Watershed (deletion/erosion)	0.950 ± 0.010	0.915 ± 0.022	0.981 ± 0.010
Watershed (distance transform)	0.738 ± 0.127	0.575 ± 0.147	0.915 ± 0.059
Watershed (adaptive)	0.954 ± 0.010	0.909 ± 0.329	0.975 ± 0.020
Grabcut (deletion/erosion)	0.950 ± 0.012	0.903 ± 0.024	0.993 ± 0.003
Grabcut (distance transform)	0.918 ± 0.024	0.838 ± 0.046	0.990 ± 0.004

Table 2.4.: Results for blurred test images I_4.

that the major factor determining the quality of the extracted graph is indeed the segmentation step. Errors introduced by thinning and graph detection appear negligible in comparison.

2.3.4. Evaluation of Speed Performance

NEFI was designed to efficiently process large quantities of images. Thus it outsources computationally intensive tasks to highly optimized and reliable libraries such as OpenCV [37] and NetworkX [74]. Table 2.5 illustrates the effectiveness of some of NEFI's algorithms.

Pipeline element	Small image (1152 × 864)	Large image (5760 × 3840)
Watershed	< 1 s	2 s
GrabCut	6 s	160 s
Adaptive threshold	< 1 s	7 s
Guo-Hall thinning	< 1 s	12 s
Vertex detection	< 1 s	5 s
Edge detection	< 1 s	6 s
Computing edge weights	< 1 s	5 s

Table 2.5.: Timings of some of Nefi's pipeline elements on images of different dimensions as given in pixel. The timings were obtained on a Macbook Pro notebook equipped with a 2.4 GHz Intel i5 processor and 8 GB of RAM.

2.4. Limitations of NEFI

In Section 2.3.3 we have seen how the quality of the extracted graph changes depending on the input image and the selected segmentation algorithms. We have established that the major factor determining the quality of the extracted graph is indeed the segmentation step. As a result Nefi's major limitation naturally arises from the limited domain of effectiveness of Nefi's general purpose segmentation collection. It's methods do not work very well if the input contains irregular background or color/brightness gradients, has low contrast or insufficient resolution to detect structures that are either too dense or too fine. For these inputs, domain specific algorithms are necessary and should be implemented as extensions for Nefi. Alternatively, the segmentation step can be entirely outsourced to more specialized third-party software in which case Nefi's pipeline may start directly with graph detection.

In general, however, Nefi's domain of effectiveness is limited to sufficiently clean and uncluttered images such as images produced under controlled laboratory conditions. We refer the reader to Appendix A for a guide on how to use Nefi which also summarizes our experience when dealing with more challenging input.

Another limitation arises due to the nature of Nefi's vertex detection. Since we walk the skeleton in search for junctions corresponding to nodes, no nodes of degree two are detected. Furthermore, nodes of high degree (4 or more) are split into several degree three nodes. The latter can be undone by merging nodes using a suitable graph filter.

As a concluding remark, we stress that Nefi tries to compensate some of its limitations by offering the possibility to work with segmented images from other sources or to integrate additional algorithms with comparably little effort. Nefi should be regarded as a flexible platform suitable for further development rather than a universal solution to the difficult general problem of network extraction.

2.5. Synergies With Other Software

2.5.1. Analysis of Graphs

NEFI is a tool that facilitates data acquisition, which is a necessary precursor to data analysis. To analyze NEFI's output one can either rely on open source graph analysis software [16], [17], [74], [106], [107], [185] or write custom programs. Owing to NetworkX [74], NEFI can output many common graph formats, readable by most popular applications which process graphs. To get the user started immediately, we provide a minimal Python program that illustrates the basic steps required to perform graph analysis. It shows how to read NEFI's output from disk and how to compute a histogram of a given edge attribute. The code can be downloaded from NEFI's project page at the Max Planck Institute for Informatics, nefi.mpi-inf.mpg.de.

2.5.2. Third-party Segmentation Software

NEFI's graph detection takes a segmented image as an input. Such an image need not be produced by using NEFI but can be obtained by relying on arbitrary third-party segmentation algorithms or tools.

In this context an interesting tool called Ilastik [159], was brought to our attention. Ilastik utilizes concepts from machine learning and offers a so-called *classification work flow* were a pixel classifier is trained relying on interactive user inputs. The trained classifier can then be used to automatically segment previously unseen images. Given proper training, the classifier might be able to deal with images that are hard for non-supervised, automatic segmentation as deployed by NEFI. Naturally, the segmented images obtained using Ilastik can directly be turned into graphs using NEFI. By cleverly combining NEFI's graph extraction with with third-party segmentation software NEFI's major weakness can potentially be circumvented completely.

2.6. Where to Download NEFI and how to Contribute

NEFI is an open source Python application and available at its project page at Max Planck Institute for Informatics, nefi.mpi-inf.mpg.de. NEFI's homepage includes a gallery of various use-cases and a comprehensive guide containing instructions on how to download, install and use NEFI on Windows, Mac and Linux. Additionally, a supplementary dataset is available for download there, allowing for a quick evaluation of NEFI's main features. This dataset can be used to reproduce the figures and evaluation results shown in this manuscript.

Ongoing development of NEFI is organized via a dedicated repository which is linked from NEFI's project page. The repository makes the source code of NEFI

available to everyone. People can get their own copies which they can modify, adapt and extend according to their needs. Furthermore it is possible to join the project as a contributor. Additional services such as bug reporting and issue tracking, as well as a small forum revolving around Nefi are also available for public use.

2.7. Discussion

Nefi is a valuable tool which allows scientists from any domain to automate graph extraction from images in an intuitive fashion requiring no expert knowledge. We hope that researchers will be able to spend more time on analyzing their data and less time on processing it. By providing a flexible platform for graph extraction, we invite experts to extend and improve Nefi in order to introduce their contributions to a wider interdisciplinary audience. In the long run we would like Nefi to further the field of network science by promoting the creation of new network databases.

In the context of this thesis Nefi provides the key to the numerical study of properties of the vein networks formed by *P. polycephalum*.

2.8. Acknowledgments

We thank Prof. M. Hauser for his hospitality and the stimulating discussions which sparked our interest in the topic of graph extraction. We thank P. and R. Winegar as well as the Korea Institute of Science and Technology Europe (Kist Europe) for contributing sample data which was critical during the early development stages of Nefi.

A large software project such as Nefi cannot be realized by one individual alone. Its development was a team effort requiring the support of several individuals. In particular we are grateful towards several students from Saarland University for contributing to the code base of Nefi. A full list of acknowledgments is available at Nefi's project page here: nefi.mpi-inf.mpg.de.

3 | Slime Mold Graph Repository

In this chapter we introduce the Slime Mold Graph Repository or SMGR, a novel data collection promoting the visibility, accessibility and reuse of experimental data revolving around network-forming slime molds. By making data readily available to researchers across multiple disciplines, the SMGR promotes novel research as well as the reproduction of original results. While SMGR data may take various forms, we stress the importance of graph representations of slime mold networks due to their ease of handling and their large potential for reuse. Data added to the SMGR stands to gain impact beyond initial publications or even beyond its domain of origin.

We initiate the SMGR with the comprehensive KIST Europe data set focusing on the slime mold *Physarum polycephalum*, which we obtained in the course of our original research. It contains sequences of images documenting growth and network formation of the organism under constant conditions. Suitable image sequences depicting the typical *P. polycephalum* network structures are used to compute sequences of graphs faithfully capturing them. Given such sequences, node identities are computed, tracking the development of nodes over time. Based on this information we demonstrate two out of many possible ways to begin exploring the data. The entire data set is well-documented, self-contained and ready for inspection at smgr.mpi-inf.mpg.de.

The chapter documents an extensive collaboration with Mag. T. Mehlhorn and Prof. K. Mehlhorn [54]. The former was responsible for the realization of all described wet-lab experiments in the laboratories of the Korea Institute of Science and Technology Europe (KIST Europe).

3.1. Introduction

Slime molds are interesting and complex organisms providing a rich substrate for interdisciplinary research. One member of the family, *Physarum polycephalum*, has received increased interest as of late. The resulting intensive research efforts continue to shed light on many aspects of this organism. Of particular interest is its ability to form and maintain complex networks. Efforts to improve our understanding of formation, structure and function of these networks are manifold [5], [19], [22], [108], [176] and ongoing.

A popular two-step approach, not restricted to slime molds, consists of taking images of the networks formed by the organism and converting them to graphs.[1]

[1] In this context graphs denote abstract mathematical objects, subject of Graph Theory, consisting

First, images are obtained by cultivating *Physarum polycephalum* in the wet-lab whilst documenting the development of the organism and its networks. This is a time consuming process which needs to be repeated sufficiently often under constant conditions to acquire a reliable body of observations.

The second part requires methods capable of analyzing an image and deriving a faithful graph representation of the network depicted therein. Such methods have become available recently as convenient software packages [52], see Chapter 2. Once graphs are available, concepts and methods from network science and graph theory directly apply, enabling efficient and detailed investigations of graph properties [20], [78].

We stress that both steps are challenging as they require time, special laboratory resources and expert knowledge. Data acquisition and graph extraction in particular, may quickly become serious obstacles deterring interested researchers from starting to work with networks formed by *P. polycephalum*.

Despite such difficulties, graph-based approaches have been quite successful and various interesting results are available today [19], [22], [61], [62], [84]. However, data used to establish these results, i.e. the graphs and their underlying images themselves, are not nearly as available in many cases. This is unfortunate because due to their ease of handling and their power of abstraction, graphs naturally lend themselves to reuse, potentially gaining impact beyond their initial publications or even beyond their domain of origin.

Our own experience shows, that most researchers are, at least in principle, willing to share their valuable data. However, data sharing can be cumbersome which often constitutes an extra hurdle discouraging data reuse. To combat this, data needs to be collected and made available in an organized fashion. Similar efforts have become best practice for diverse types of data originating in various fields of science. Examples are numerous including collections of images of cells [38], large graphs [106] or experimental data in high energy physics [180], to name but a few.

To the best of our knowledge no such repository exists for data concerned with networks of slime molds. For this reason we decided to set up the *Slime Mold Graph Repository* with the goal of providing an available collection of networks focused on slime molds. Although this is clearly a niche topic, we believe that due to the many open question revolving around the structure and function of such networks and their large interdisciplinary appeal, setting up a small but dedicated repository is of pronounced value.

In the following we discuss the concept and benefits of the SMGR followed by a detailed account of its initial contents, the KIST Europe data set. We start by discussing how the data was obtained experimentally and move on to present our tracking algorithm, a novel method designed to resolve how networks formed by *P. polycephalum* change in time. To illustrate the usefulness of both data and method, we show proof-of-principle applications which suggest avenues for further research. Our exposition aims to promote the SMGR across disciplines and is sufficiently

of vertices and edges. Throughout this thesis we will use both terms interchangeably

detailed and introductory for non-experts to access.

3.2. Repository Concept and Benefits

Today the impact of data sharing as a scientific practice is ever increasing. Currently, two major forms of data collections are commonly encountered.

First, data is being collected in large, well-organized repositories which are holding enormous amounts of information. Such major repositories easily contain thousands of datasets, covering numerous topics across its domain of relevance. Due to their size and complexity, curating and maintaining such repositories requires major resources, typically provided by larger research institutions or non-profit organizations.

Most major data collections, however, have started as small and specialized repositories at some point. Such repositories constitute the second approach to data sharing common today. Small repositories are typically much simpler in structure and contain a smaller number of datasets. They tend to be more specialized and have very close ties to their relevant research communities since the operators of the repository are often researchers themselves. Their small size allows them to operate with minimal infrastructure requirements since changes to the repository are less frequent and data sets are limited in number and space requirements. Due to their minimal implementation they do not offer most of the services frequently provided by larger repositories and focus on but the most essential features.

With the SMGR we seek to establish a collection of research-grade data revolving around network-forming slime molds. Given the highly specialized nature of the topic, we believe that setting up a small stand-alone repository realized as a minimal implementation is advantageous and the correct point to start from. Note that we do not present novel software but rather combine available open source software components to supply a minimal repository service. Keeping the repository simple allows for easier maintenance and more flexibility. As a result it is easier to adapt and evolve based on community feedback. This is why at present only the most basic features necessary for repository operation are offered by the SMGR. However, if sufficient and continued interest is observed, the implementation of additional features may be warranted. For further reasons why we launch the SMGR as a minimal repository, please consult the FAQ on the SMGR project page at smgr.mpi-inf.mpg.de.

In its present form, the idea of the SMGR was first introduced at *PhysNet 2015* in New York and was well received with individuals signaling their willingness to contribute data [51]. We strongly believe that a tight integration with the research community, is key to the future success of the SMGR.

The benefits of the SMGR are manifold. Making data available for everyone increases visibility of contributors, allows original results to be reproduced and puts data in a prime position to become a catalyst for novel research. Given the significant costs, i.e. time and resources, which are typically associated with obtaining high quality data, promoting increased reuse of data is an economical choice. Researchers

and other professionals that do not have the required resources/connections to produce/obtain their own data sets benefit in particular from repositories like the SMGR, since it provides immediate and convenient access to experimental material that would be hard to acquire by other means.

We envision the SMGR to become a mediator between theory and experiment. Theoretical work in biology and biophysics concerned with modeling various aspects of *P. polycephalum* networks, may utilize experimental data as a testbed for model predictions. This approach has been put into practice previously [21], but was exclusive to researchers in possession of relevant data. With the introduction of the SMGR such limitations are removed. Similar statements can be made for other fields, e.g. computer science, which is actively studying *P. polycephalum* in the context of natural Computing, see Section 1.3. Access to experimental data via the SMGR allows to compare theory and experiment and helps build the intuition crucial to theoretical investigations and modeling efforts.

In order for the SMGR to be useful from day one, we initiate the repository by sharing the extensive data that is driving our own slime mold research which was obtained at the *Korea Institute of Science and Technology Europe* in Saarbrücken. It is important to us, that anyone can go the SMGR project page, download our data and immediately start working with it in whichever way he or she chooses. The data set we provide also serves as a possible example for the type of data the SMGR looks to collect and what level of documentation is appreciated.

We stress at this point that the SMGR is not intended to simply provide a permanent data storage service. First and foremost, we seek to collect data that has not yet been fully explored an has a strong potential to be of use to others. The KIST Europe data set described in the next section fulfills both criteria.

Finally, any repository needs a set of instructions, policies and requirements applying to data submission, data usage and various other general aspects of repository operation. For these the SMGR relies on established best practices of data sharing whilst striving to keep things as straight-forward as possible [182]. A detailed account can be found on the SMGR project page, which we consider an integral part of our work. We opt to treat policies, user instructions and other important questions in a comprehensive FAQ on the SMGR project page rather than in this thesis, simply because they are subject to future change.

The SMGR and all available data can be found here: smgr.mpi-inf.mpg.de.

3.3. The KIST Europe Data Set

In the following we present an overview of the KIST Europe data set designed to give the interested reader a high-level understanding of its nature and content. At the same time, we recommend to inspect the data directly using the browsing and download functions provided on the SMGR project page. We refer the expert reader, interested in reproducing all the steps involved in the creation of the data set, to an in-depth exposition given in Appendix B.

The KIST Europe data set contains raw and processed data obtained and derived from 81 identical wet-lab experiments, carefully executed under constant conditions. Figure 3.1 illustrates the experimental setup used. The data was produced using the following procedure:

1. A rectangular plastic dish is prepared with a thin sheet of 1.25 % agar.
2. 1.5 g of dried *P. polycephalum* (HU195xHU200) sclerotia crumbs are lined up along the short edge of the dish, see Figure 3.2. The dish is put into a large light-proof box.
3. After approximately 14 h the plasmodium resuscitates and starts exploring the available space towards the far side of the dish. Typically, the apical zone needs to cover a distance of several centimeters before network formation can be observed properly, see Figure 3.3.
4. For the next 30 h we take a top-view image of the growing plasmodium and the changing network every 120 s from a fixed position. A typical obtained image is seen in Figure 3.4. We stop capturing when the apical zone is about to reach the far side of the dish, which is well outside of the observed area.
5. After obtaining sequences of images showing the characteristic networks of *P. polycephalum*, we use NEFI to compute corresponding sequences of graph representations of the depicted structures within a predefined region of interest [52], see Figure 3.5. The graphs store precise information of the length and width of the edges as well as the coordinates of the nodes in the plane. A typical resulting unfiltered graph is seen in Figure 3.6.
6. Given the resulting sequence of graphs we apply filters removing artifacts and other unwanted features of the graphs. Then we proceed to compute a novel node tracking which encodes the time development of every node taking into account the changing topology of the evolving graphs.

Repeating this experiment we obtain 81 similar sequence of images, which we consider our raw data. We stress at this point that given the inherently uncontrollable growth process of *P. polycephalum*, the obtained sequences differ in length and nature. In some experiments the organism behaved unfavorably, simply stopping its growth, changing direction or even escaping the container. While such sequences are part of the raw dataset, we excluded them partially or completely from the subsequent graph extraction efforts. The removal of such data reduces the number of series depicting optimal network formation to 54.

After obtaining the raw data, we transform the images into equivalent mathematical graphs, thus opening up a wealth of possibilities for data analysis. To this end we deploy NEFI [52], which analyzes a digital image, separates the depicted slime mold network from the background and returns its graph representation. Using this tool effectively requires moderate amounts of image preprocessing. In particular, for

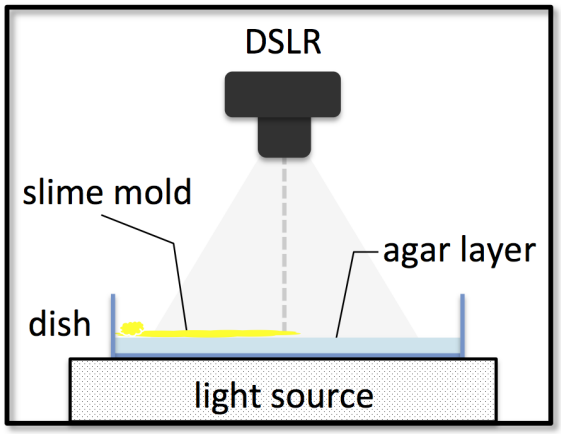


Figure 3.1.: Schematic description of the experimental setup.

Figure 3.2.: Crumbs of *P. polycephalum* sclerotia forming the inoculation line.

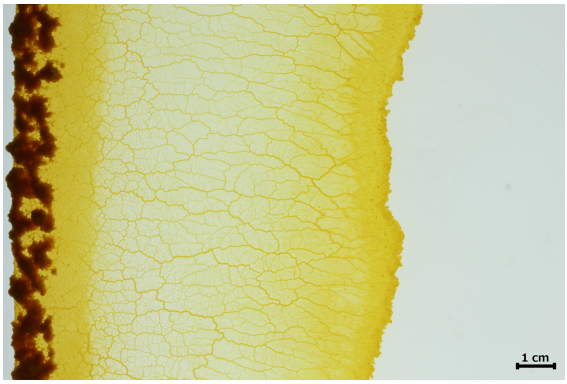


Figure 3.3.: The plasmodium explores the dish. The apical zone advances towards the right side of the dish supported by a complex network that is continuously forming.

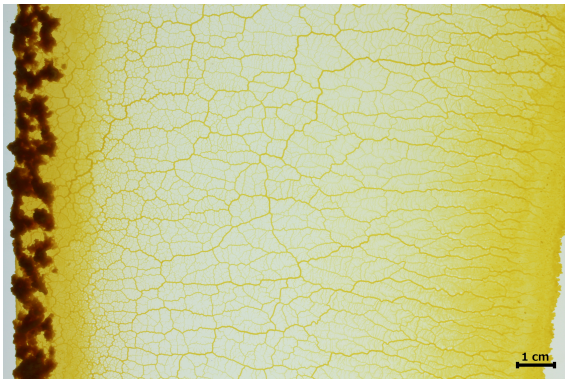


Figure 3.4.: As the apical zone is about to escape the observation region, the coarsening of the network becomes more pronounced.

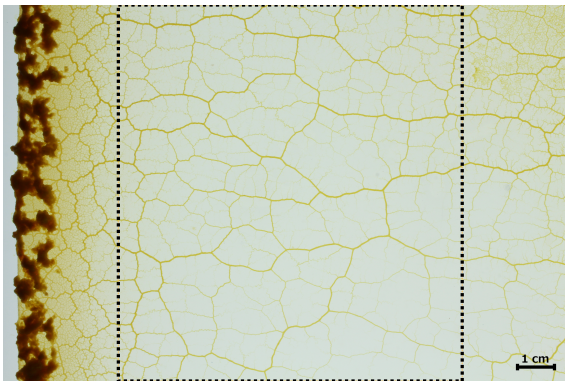


Figure 3.5.: The apical zone has moved on, leaving behind a complex network of veins. The dashed rectangle depicts a typical region of interest relevant for subsequent image analysis and graph detection.

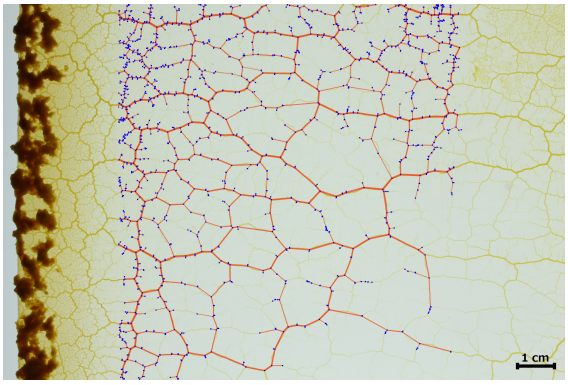


Figure 3.6.: The network within the region of interest has been extracted by NEFI. Note that no filters have been applied. Dead ends and nodes of degree 2 are visible still, leading to small patches of nodes appearing to clump up. Such artifacts can be removed in suitable post-processing steps.

each sequence of images it is necessary to decide on a suitable subsequence to be processed. Here we typically exclude parts of the sequence where the apical zone is still visible. For each such subsequence a suitable region of interest is defined manually. Figure 3.5 depicts a typical choice for the region of interest to be processed by Nefi. The established unfiltered graph can be seen in Figure 3.6. The graph stores the position of the nodes in the plane as well as edge attributes such as edge length and widths for each edge. In addition to the output of Nefi including the unfiltered graphs, the dataset contains Nefi's input, i.e. the selected subsequences of images cropped according to their defined regions of interest.

Note that some parts of the image series showing proper network formation did not yield optimal representations of the depicted networks. This is a result of images exhibiting strong color gradients or other effects rendering them too challenging for automatic network extraction. While such cases may still be handled by tuning the parameters of image processing manually on an image per image basis, we decided to discard affected series from subsequent processing efforts. As a result the number of usable graph sequences of highest quality reduced to 36. To this we apply a set of filters removing artifacts, isolated small components and dead-end paths. Thus we obtain a total of 3134 distinct filtered graphs faithfully reflecting the topology and edge attributes which *P. polycephalum* displayed during the wet-lab experiments.

At this point available graph analysis packages or custom written analysis code can be deployed to investigate the data in various ways, e.g. [16], [74]. The dataset includes the filtered graphs as well as all corresponding graph drawings. The latter enable a quick visual inspection of the graph extraction results.

Given the obtained time-ordered sequences of graphs the development of the entire graph can be investigated. One may also study what happens to single nodes as *P. polycephalum* evolves. Given a graph in a sequence of graphs, let us pick any node u. Can we determine a set of nodes from graphs in the sequence that are equivalent to u, i.e. all nodes in the set are earlier or later versions of u in time? We answer this question by computing a so-called *node tracking* which establishes the time development of all nodes in the graph. Crucially, this tracking takes into account topological changes in the evolving graphs. The result of the tracking is available as node properties of the graphs. Naturally, the program computing the tracking is included in the dataset. To the best of our knowledge, this type of data and algorithm is made available for the first time through the Kist Europe data set.

Finally, in addition to the actual data, i.e. images and graphs, the Kist Europe data set contains scripts and larger programs used to process and evaluate the data. Suitable configuration files specify the used regions of interest and the parameters used with Nefi. Thus it is possible to repeat the entire data production process from the raw images to the obtained filtered graphs including the tracking of nodes. As part of the Smgr, the Kist Europe data set is well-structured and self-contained. In particular, sufficient on-the-fly documentation is available when using the online browsing function of the Smgr.

3.3.1. Node Tracking

In this section we explain how to compute a node tracking. Our approach is a variant of a tracking method introduced previously [99], [184]. Here we give a self-contained account of the tracking technique geared towards non-experts in optimization. For technical details, proofs and an experimental evaluation of the method we refer the interested reader to the original publications.

For each of our experiments, let us denote the respective time ordered sequence of k filtered graphs with $\mathcal{G} = G_1, G_2, \ldots, G_t, \ldots, G_k$ and the union of the respective node sets with $\bar{V} = \bigcup_{i=1}^{k} V_k$. Each node in $\bar{V}$ can be represented as a non-negative integer triple (x, y, t). Here x and y denote a node's pixel coordinates relating to the input image used for graph extraction and t denotes its position in the sequence, i.e. its position in time.

We seek to exploit this information to partition the set $\bar{V}$ into a collection of disjoint, time-ordered paths such that every $u \in \bar{V}$ is part of exactly one path. We call such a path a *track* and assign an unique identifier to it, e.g. a unique color, which is shared by all nodes in the track. Intuitively speaking, rather than thinking of a track as a collection of nodes, one can interpret it as the realization of the same physical node at different points in time.

Let us define the length of a track by the number of nodes it contains. We stress that in general a track need not have length k. Tracks may start at any $t = i$ such that $1 \leq i \leq k$ and end at any $t = j$ such that $i \leq j \leq k$. In particular tracks of length 1 are possible. Crucially, any node in a track has at most one predecessor and at most one successor node.

Let us refer to the edges within a track as *tracking edges*. Tracking edges may connect nodes of temporal distance larger than one, i.e.:

$$e = (u, v) = ((x_u, y_u, t_u), (x_u, y_u, t_v)) \qquad \text{and} \qquad 1 \leq t_u < t_v \leq k\,.$$

Such tracks simply "skip" nodes between the start and the endpoint of the track and correspond to nodes that have not been observed by Nefi for some amount of time. Given the continued changes in the network topology observed during the growth of *P. polycephalum* such tracks are expected to appear frequently.

Let us elaborate on two effects that impact said topology. First, *P. polycephalum* networks tend to coarsen as time goes by. Eventually, this process may cause the thickness of receding veins to slowly drop below the detection threshold of Nefi. Hence, they disappear from the graphs leading to tracks of length much shorter than k. Second, periodic changes in the thickness of veins are observed. It may happen that the thickness of a subset of contracting veins may drop below the detection threshold of Nefi. As a result, those veins disappear from the graph. However, as the contraction cycle of the vein proceeds its thickness may increase again, eventually exceeding the detection threshold and thereby returning to the graph. These periodic changes in the topology of the graphs naturally lead to tracks that skip a certain number of frames periodically. Both effects are present at the same time and can be observed in the graphs we have obtained.

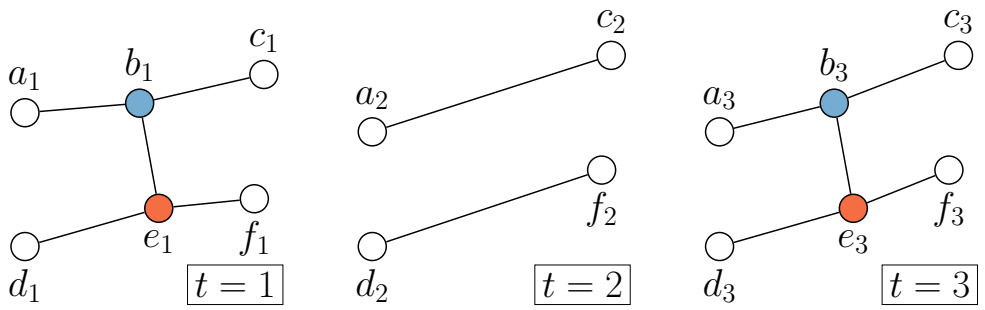


Figure 3.7.: Schematic of a graph and its changing topology with respect to time. At time $t = 2$ the edge (b, e) and its nodes disappear from the graph because the thickness of the corresponding vein in *P. polycephalum* dropped below the detection threshold of NEFI. The colors indicate tracks T_b and T_e for nodes b and e respectively.

We stress at this point that given a time resolution of 120 s, one must not use the graphs contained in the KIST Europe data set to study the quasi-periodic oscillations of edge thickness due to peristaltic pumping. The period of these oscillations is known to be approximately 100 s and thus cannot be properly resolved [163]. However, large changes in topology are on a much slower time scale and are reliably reflected by the graphs.

We proceed to explain how to compute a node tracking. Let us study the graph in Figure 3.7 to gain some insight into the problem. Consider e.g. node a_1. We seek to find a unique track that contains this node, representing its development in time. A straight-forward suggestion would be a track $T_a = [a_1, a_2, a_3]$. However the track $T_a = [a_1, c_2, f_3]$ is just as valid, as is $T_a = [a_3]$ and so forth. The example illustrates that the number of possible tracks and therefore the number of disjoint partitions into tracks grows exponentially with the number of nodes involved.

Ultimately, we seek to find a partition, that faithfully captures the time development of the nodes in $\bar{V}$. To do so we rely on the observation that optimal tracks are likely to consist of nodes that are close in space as well as close in time. In the following we formalize this requirement and construct a linear program (LP) that selects an optimal partition amongst all possible partitions. Once an LP has been defined, solving it becomes a standard task of optimization and can be done efficiently using solvers like Gurobi [73], or Cplex [46].

Let us define the optimization problem we want to solve. First, we construct all possible tracking edges using space partition techniques. Assigning a unique identifier to each edge we obtain the set $\bar{E}$. When the linear program considers edges to include in a possible tracking solution it refers to them using the ids in this set. We define the cost of selecting a particular tracking edge $e = (u, v) = ((x_u, y_u, t_u), (x_v, y_v, t_v))$ as

$$\Delta(e) = \epsilon((x_u - x_v)^2 + (y_u - y_v)^2) + \tau(t_u - t_v)^2\,, \tag{3.1}$$

where ϵ and τ are constants used to control the relative strength of the squared

Euclidean distance in space and time respectively. Note that using this cost function in our example in Figure 3.7, track $T_a = [a_1, a_2, a_3]$ is favored over $T_a = [a_1, c_2, f_3]$. Note also that tracks of length 1 have cost 0 since there are no tracking edges to pay for. As a result the minimum cost solution consists of singleton tracks and is not very useful. To circumvent this problem we assign dedicated costs to any node that starts or ends a track as

$$p(u) = C\,. \tag{3.2}$$

Here it is important that C is selected carefully. It must not be too small since this will force the formation of artificial singleton tracks while too large a value may erroneously suppress such tracks. Going back to Figure 3.7, assuming $\epsilon = \tau = 1$, a choice of $1 < C < 2^2 = 4$ will lead to two singleton tracks for the node b, namely b_1 and b_3 because a tracking edge connecting b_1 and b_3 comes at an edge cost of $\Delta(e) = 4$. Increasing the node cost such that $C > 2^2 = 4$ changes the picture and results in the desired tracks $T_b = [b_1, b_3]$ and $T_e = [e_1, e_3]$. The example illustrates the importance of the constants involved in the cost functions.

Note that, we are still facing an exponential number of tracks to consider. Luckily, we can reduce the size of the set $\bar{E}$, and thus the size of the LP, dramatically by combining the effect of the costs functions with additional assumptions valid for tracking *P. polycephalum* graphs. In particular, the cost functions imply that we favor non-singleton tracks with nodes that are close in time as well as space. As a result, we may discard all options that have a geographical distance larger than a certain R and a temporal distance larger than a certain T. Intuitively, when trying to find all possible tracking edges for a node u, we likely need not consider nodes that are located on the other side of the graph or have an unrealistically large temporal separation. Instead we can restrict the search to tracking edges within a cylinder centered at u with radius R and a height of T. We can exclude all tracking edges outside such a search cylinder, because their resulting costs are such that the linear program will never select any of them.

The problem thus boils down to determining suitable values for R and T. Since the nodes of *P. polycephalum* are well-separated and do not move within a time period of 120 s, a small radius of $R = 30$ pixel is justified. We choose $T = 10$ which amounts to a time period of 1200 s to look for tracking edges in temporal direction. To tie the cost of singleton tracks to the physical features of our experiment, we choose $C = 2R^2 + T^4$. Setting $\epsilon = 1$ and $\tau = 10$ completes the set of constants involved in the tracking problem. With these choices we follow the approach in [99].

We convinced ourselves that these settings are suitable for dealing with *P. polycephalum* graphs by computing node trackings on artificial test graphs which enable a comparison with known ground truths. To obtain one such series we take a real *P. polycephalum* graph and randomly remove a certain fraction of nodes from it to obtain artificial graphs at later times. Furthermore we slightly perturb the coordinates of all the nodes of these graphs. As expected we find the error rate to strongly depend on the strength of the geographical perturbation of nodes. As long as the

perturbation is smaller than the minimum distance between nodes, the computed tracking is identical to the ground truth. When nodes start moving more from frame to frame, there is a danger that they randomly move past each other. As a result they end up swapping their respective tracks, introducing an error. When the perturbation is set such that a node can only move within a radius of $R = 30$ pixel, less than 4 % of the nodes end up in wrong tracks given the settings illustrated above. However, we expect that this error is somewhat pessimistic because nodes in real *P. polycephalum* do not move much compared to our artificial test graphs. Careful visual inspection of the obtained node trackings resulting from real *P. polycephalum* data supports this conjecture.

From the arguments in the previous section it also follows that filtering has a strong impact on the quality of the node tracking. If graphs are filtered such that nodes are well-separated, excellent results can be expected. Thus, we recommend to carefully apply filters if any research questions based on the node tracking are to be tackled. In particular, contracting nodes of degree 2 is strongly advised.

Finally, by combining the considerations discussed above, we are in a position to define the actual linear program computing a node tracking:

$$\textbf{Minimize:}\quad f = \sum_{e\in\bar{E}} \Delta(e)x_e + \sum_{i\in\bar{V}} p(i)y_i + \sum_{j\in\bar{V}} p(j)z_j$$

$$\begin{aligned}
\textbf{s.t.:}\quad & \forall\, e \in \bar{E}:\ x_e \geq 0 \qquad \forall\, i \in \bar{V}:\ y_i \geq 0 \qquad \forall\, j \in \bar{V}:\ z_j \geq 0 \\
& \forall\, i:\ y_i = 1 - \sum_{e=(n,i)\in\bar{E}} x_e \qquad \forall\, j:\ z_j = 1 - \sum_{e=(j,n)\in\bar{E}} x_e\,.
\end{aligned}$$

Note the last two constraints including sums over tracking edges that leave respectively enter a node. They make sure that a given node represented by y_i either has exactly one predecessor or pays the cost $p(i)$ for starting a new track. Likewise, a node represented by z_j either has exactly one successor or pays the cost $p(j)$ for ending an existing track. Thus the LP either pays for selecting a tracking edge or the penalties associated with ending or starting tracks but never both. For a more technical exposition of this optimization problem we refer the reader to [99].

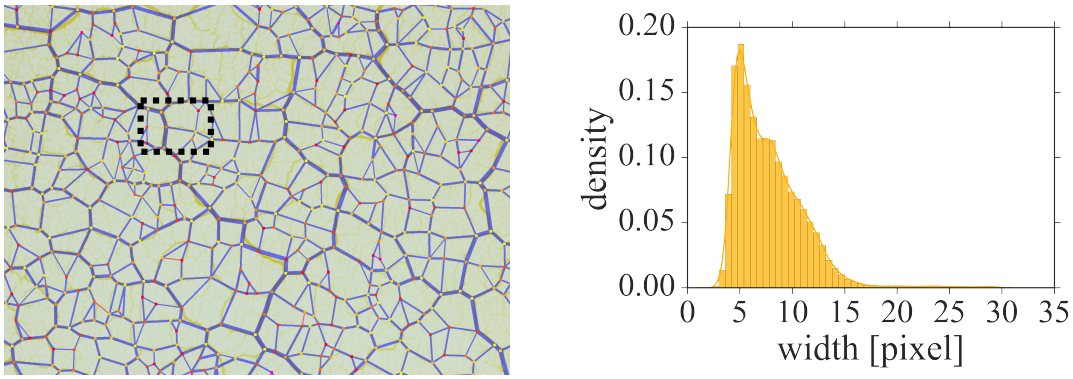
The optimal values of the LP variables x_e, y_i and z_j are then determined by solving the LP. All LP variables are integer values where a selected tracking edge e is encoded by $x_e = 1$, a selected node i is starting a track if $y_i = 1$ and ending it if $z_i = 1$. Naturally, if $y_i = z_i = 1$ we have a singleton track consisting solely of node i.

Partitioning the entire selection into disjoint tracks yields the desired node tracking. As a final step, we assign each track a unique color identifier and propagate this color to each node in the track, i.e. across all graphs in a given series. The color information is part of the graphs of all our datasets and thus readily available for further processing. A program constructing and solving the tracking LP using Cplex [46] is provided as part of the data set.

3.4. Sample Usage of the KIST Europe Data Set

Next we illustrate the principal usefulness of the data contained in the KIST Europe data set set by demonstrating how to compute two different example observables. The python code used to obtain the presented plots is available as part of the data set. As sample data we choose a time series of *P. polycephalum* graphs consisting of 60 graphs. Since the time between two graphs is 120 s, the whole series documents the growth of the slime mold over a period of 2 h. Figure 3.8a shows one of the graphs superimposed on the image from the laboratory it was extracted from. The intricate network formed by *P. polycephalum* is clearly visible. In particular it can be seen that the edges of the graph are of varying thickness. This is to be expected and reflects the variations of thickness observed in the veins of the slime mold.

One may ask the question of how the thickness of veins is distributed in these graphs. Answering this question is very easy, since the graphs in the KIST Europe data set all come with edge weights readily available. Edge thickness or lengths are present for all edges in the graphs. As a result the edge width data can easily be accessed and represented as a histogram. Figure 3.8b shows a cumulative normalized histogram of the edge widths including the edges from all 60 graphs in the series. Note that the width is measured in pixel. Due to the resolution of the original images 37.0625 pixel correspond to 1 mm.



(a) (b)

Figure 3.8.: (a) A sample graph of *P. polycephalum* from the KIST Europe data set. It is part of a series of 60 graphs. The area indicated by the dashed rectangle is magnified in Figure 3.9 (b) Cumulative edge width distribution computed over all 60 graphs.

While computing such histograms is merely an exercise of using the information stored in the graphs, more interesting and insightful observables can be derived. In fact, in Chapter 4 the results of a comprehensive non-trivial study of the KIST Europe data set will be presented [53].

In the context of said study, additional questions that could be explored based on the KIST Europe data set naturally arose. In particular, one may determine

whether there is a similarity between *P. polycephalum* networks and Voronoi graphs. The latter are well-studied and it is interesting to explore a possible connection between their properties and the features of *P. polycephalum*. A different suggestion consists of exploring the question whether *P. polycephalum* graphs are geometric spanners. Spanners have properties that enable efficient communication between different parts of the graph, a feature clearly relevant and desirable for an organism such as *P. polycephalum*.

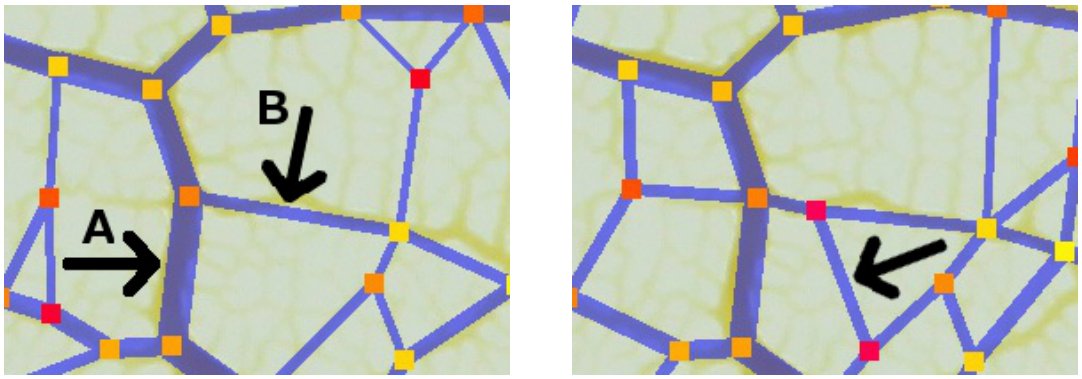
Yet another way to study the KIST Europe data set lies in the information provided by the tracking algorithm presented in this manuscript. Since all the graphs in the data set were tracked already, identifiers are available determining which nodes belong to which unique tracks. For simplicity, we refer to the track identifier as *color*. As a proof-of-principle application, we shall demonstrate how to exploit this information to study the width of selected single edges in time. Since we aim to follow the time development of edges, we first need to establish the concept of tracks for edges. For any edge $e = (u, v)$, we define its color by combining the colors of u and v in a set:

$$\text{color}(e) = \{\text{color}(u), \text{color}(v)\}\,. \tag{3.3}$$

Looking at the tracking example given in Figure 3.7, the edges (b_1, e_1) and (b_3, e_3) are assigned the color $\{blue, orange\}$ signaling that both edges belong to the same edge track and are merely different instances of one and the same edge at different points in time. Note that this edge is missing at $t = 2$ and thus the length of the edge track is reduced by one.

Given this edge color information it trivial to find all the edges that belong to an edge track. Once the tracks are known, it is easy to study how edge properties change within a track. Figure 3.9a gives a detail of the graph shown in Figure 3.8a and highlights two distinct edges which we want to investigate in this example. Edge A was chosen such that the length of its track is equal to the length of the entire time series, i.e. this thick edge is fully accounted for and does not disappear at any point in time. To illustrate the way the tracking algorithm resolves tracks were edges disappear at different points in time, we have selected an edge B which is much thinner than A. There are two ways for an edge to disappear. Either its thickness falls below the detection threshold which means it is not present in the graph anymore, or it is split by a node introduced by the appearance of another edge. Figure 3.9b shows the latter situation where B is split into two new edges by a newly detected third edge. The new edges have colors different from B.

Finally, Figure 3.10 shows the edge widths for edges A and B given their established tracks. Note that the track including the thicker edge A runs across the entire 60 graph in the series. The situation is different for edge B where its track reflects that it is repeatedly split by a third edge which is sometimes present. Eventually, the splitting edge remains established for much of the latter half of the time series, thus superseding edge B permanently. This simple example shows how the tracking information allows one to "zoom in" on what happens with selected edges and study



(a) (b)

Figure 3.9.: Detail of a *P. polycephalum* graph, corresponding to the dashed rectangle in Figure 3.8a. (a) Two edges A and B are selected for tracking. (b) A third edge appears, causing B to split up causing it to disappear from its track.

them in detail.

At the time of writing, the novel information provided by the computed tracking is yet to be explored in a large and systematic study. What can be inferred from the topological changes recorded? Can one identify patterns with certain structural properties? Can topological properties be related to questions of biological relevance? Given the large number of graphs in the SMGR, an investigation of such questions becomes feasible.

The KIST Europe data set is not limited to the use cases and suggestions provided here. Rather, it constitutes a flexible basis to work with as it contains a host of useful data, code and instructions. In particular, potential users are not restricted to working with the graphs that are presently provided. They are encouraged to start from the raw images and determine their own specific data selection, graph extraction and tracking procedures tailored to their particular research agenda. They may use the tools provided by us or deploy entirely different strategies to better suit their needs.

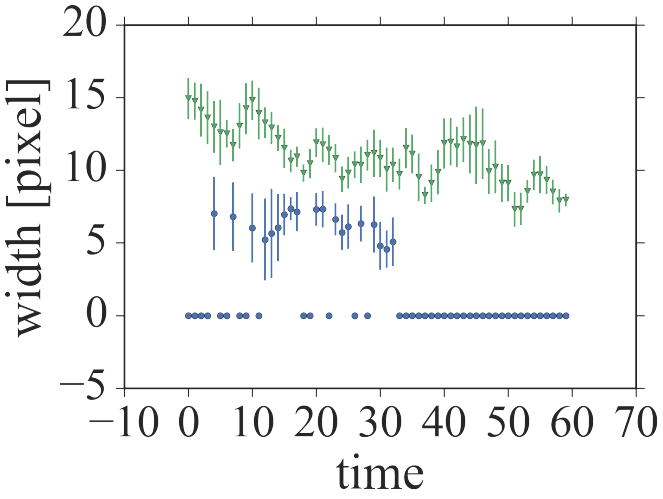


Figure 3.10.: Development of edge widths for edges A (green triangles) and B (blue circles) as given by their established edge tracks. Note that for track B we have set the value of the edge width to zero whenever the edge was missing. This serves to illustrate the fragmentation of track B. The widths are given in units of pixel and time is measured in units of 120 s.

3.5. Discussion

The first and most important step in sharing your data is to share your data [182]. To this end, we introduce the Slime Mold Graph Repository, a novel platform that facilitates the exchange of experimental data revolving around networks formed by slime molds. We believe that by encouraging the reuse of data, the value and visibility of experimental ground work is significantly increased. Not only does the reproduction of results based on publicly available data become much easier, shared data may be put to unforeseen use by researchers from different fields willing to examine it from a new point of view.

To enable this, we initiate the SMGR with an extensive set of raw data, ready-to-use graphs and dedicated methods such as the introduced tracking algorithm. We show the usefulness of time-series of graphs by discussing two out of many possible different approaches to investigate them. First, we discuss how to obtain simple statistical observables for all edges in the graph. In particular, we compute the cumulative distribution of edge widths. Second, we illustrate how the information obtained by the tracking algorithm allows us to explore the time-development of single edges.

Clearly, future investigations are hardly limited to the examples and suggestion given in this manuscript. It is fair to say that any observable defined on a weighted graph, relevant to *P. polycephalum*, can be studied using the KIST Europe data set. In particular, we'd like to stress the implications for evaluating and guiding all sorts of theoretical modeling approaches based on graphs. Any model that produces

a prediction which can be formulated as an observable defined on a graph can immediately be tested on the KIST Europe data set. This includes time dependent observables. Predictions that agree with SMGR data may increase the trust in a given model, while discrepancies between predictions and data hopefully suggest improvements. Thus, data contained in the KIST Europe data set may be used to drive modeling efforts and help bridge the gap between theory and experiment.

We would like the research community to interpret the SMGR as a twofold challenge. First, we challenge people working on slime molds to contribute their data, enable others and to increase the visibility and impact of their experimental work at the same time. Second, we challenge everyone to use the current contents of the SMGR and to come up with new ways to enrich our understanding of the interesting organisms that are slime molds.

3.6. Acknowledgments

We are grateful to Prof. T. Ueda for providing us with sclerotia and for teaching us to the art of culturing *Physarum polycephalum* with infinite patience. We thank Prof. M. Grube and Dr. C. Westendorf for hospitality, support and expert advice on slime molds. We acknowledge Prof. A. Manz and Prof. L. Abelmann and their group at KIST Europe for stimulating discussions. Lastly, we are grateful towards Dr. M. Függer for his encouraging comments and proof-reading.

4 | Network Analysis

We present a systematic study of the characteristic vein networks formed by the slime mold *P. polycephalum*. Our study is based on an extensive set of graph representations of slime mold networks. We analyze a total of 1998 graphs capturing growth and network formation of *P. polycephalum* as observed in 36 independent, identical, wet-lab experiments. Relying on concepts from graph theory such as face cycles and cuts as well as ideas from percolation theory, we establish a broad collection of individual observables taking into account various complementary aspects of *P. polycephalum* networks. As a whole, the collection is intended to serve as a specialized knowledge-base providing a comprehensive characterization of *P. polycephalum* networks. To this end, it contains individual as well as cumulative results for all investigated observables across all available data series, down to the level of single *P. polycephalum* graphs. Furthermore we include the raw numerical data as well as various plotting and analysis tools to ensure reproducibility and increase the usefulness of the collection. All our results are publicly available in an organized fashion in the SMGR introduced in Chapter 3.

This chapter presents joint work with Prof. K. Mehlhorn [53].

4.1. Introduction

Networks are systems consisting of nodes and links. In general, nodes represent abstract entities while links define abstract relationships between them. Once one adopts this powerful concept, networks appear virtually everywhere throughout nature and as systems created by men.

Man-made examples include networks of servers such as the Internet [58], [82], [98], networks of web-pages [2], networks of streets [39], [85], power grids [145], [187] or transportation networks involving trains or air planes [15], [69], [154], [157]. Examples for networks observed in nature include gene regulation networks [6], [112], food webs [116], [142], fungal cord networks [65], [78], or vascularization networks in plants and mammals [100], [102], [148], to name but a few.

A property shared by many networks is that they are *complex*. While an unambiguous definition of the term is elusive, complex networks share some common characteristics. They typically consist of a large number of nodes exhibiting possibly unknown, non-trivial interactions. The network itself as well as the interactions between its constituents are not constant but subject to change induced by internal or external events. Furthermore, self-organization and emergent phenomena are commonly observed in complex networks [14], [28].

Given the intellectual challenges and the importance of technical applications associated with complex networks, research concerned with their structure and function has attracted continuing scientific interest. For extensive reviews and references on the topic consult [7], [28], [127], [164].

The present article focuses on vein-networks formed by the unicellular, multinucleated slime mold *Physarum polycephalum* [81]. This peculiar amoeba-like organism is able to maintain a massive cell body in the form of an extended network of veins. Similar to a mammalian vascular network, the veins carry protoplasmic fluid distributing nutrients and other matter throughout the cell body by means of peristaltic pumping [96]. *P. polycephalum* reacts to changing environmental conditions by dynamically adapting the topology of its network. In the absence of a central nervous system capable of coordinating morphological changes in *P. polycephalum*, it is believed that the ability to reorganize itself to better meet environmental conditions is an emergent property resulting from underlying local interactions [119], [176].

In order to further improve our understanding of the mechanisms governing the morphology of *P. polycephalum* one may directly study the properties of its networks. A popular two-step approach, consists of capturing images of networks and converting them to mathematical graphs, enabling a detailed investigation of their properties [20]. The first part relies on cultivating *P. polycephalum* in the lab whilst taking images that document the development of the organism and its networks. The second part requires dedicated methods capable of analyzing the obtained images and computing graph representations of the networks depicted therein. Such methods have become available as software packages recently [52]. Once graphs have been established, methods from graph theory and network science apply and various quantities of interest can be computed. This approach has been quite successful and various interesting network properties have been established [19], [22], [84], however, many questions remain open.

In this chapter we continue to investigate *P. polycephalum* using the graph-based approach. Our study relies on a publicly available set of *P. polycephalum* graphs which is part of the Smgr [54], see Chapter 3.

Based on the Kist Europe data set, we define and compute a broad collection of observables characterizing *P. polycephalum* graphs. We begin our investigation by reporting a suite of basic graph properties and discuss our findings in relation to results obtained prior to our study [22]. We proceed by introducing a family of observables using concepts from graph theory such as face cycles and cuts. Finally, we study the robustness of *P. polycephalum* networks under random attack by establishing edge and node percolation thresholds.

We seek to establish a broad spectrum of descriptive observables designed to cover various complementary aspects of the organism. The combination of these may be interpreted as a kind of multidimensional "fingerprint" capable of characterizing *P. polycephalum* graphs. Using such fingerprints, it becomes possible to compare individual *P. polycephalum* graphs in a quantitative manner. This can be of particular use when comparing different slime molds with each other or in the

evaluation of models or algorithms producing graphs which claim properties similar to those of *P. polycephalum*. Our approach nicely complements recent efforts to compare different slime molds based on properties of their plasmodia and certain chemotactic responses [181].

We stress that this manuscript only provides an overview of our findings. The complete set of results is available for download and inspection at the SMGR. This includes the full list of observables and associated plots, useful tools as well as instructions that allow the reproduction of all obtained results. Crucially, the provided tools enable the reader to define and compute new observables tailored to their particular research questions.

Establishing a broad body of reliable information and making it available together with useful tools are the major goals of our efforts. We hope that they will serve as a catalyst for future research.

4.2. Methods

4.2.1. Experimental data

To investigate the properties of networks formed by *P. polycephalum* we rely on experimental data publicly available in the Smgr. The Smgr was introduced recently in an effort to improve the visibility and reuse of experimental data with a focus on slime mold networks and their graph representations.

For the present study we work with the SMGR's initial data set, called the Kist Europe data set. In particular we focus on a processed subset of this data tagged *high unfiltered*. For a detailed account on how this data was obtained (experimental setup and procedures), processed (raw images, network extraction and filtering, relevant implementations) and organized we refer to Chapter 3.

Here it suffices to say that the Kist Europe data set contains time series of mathematical graphs derived from images depicting growth and network formation of *P. polycephalum* as observed in 36 independent, identical, wet-lab experiments. Note that no additional food sources where introduced in these experiments. In addition to capturing the topology of *P. polycephalum*, the graphs carry information such as the position of the nodes in the plane and edge lengths and widths. The time between two consecutive graphs within a series is 120 s. The image resolution is such that 370.6250 pixel correspond to 1 cm. The 36 data series include a total of 1998 distinct *P. polycephalum* graphs.

4.2.2. Graph Representation

The graphs we analyze represent *P. polycephalum* networks. They have been processed to ensure that they are connected and that every edge belongs to at least one cycle. In other words, everything not contained in the largest connected component and all dead-end paths were removed. In our graphs, all nodes are of degree 2 or 3.

A significant fraction of the remaining nodes are of degree 2, forming long *paths* which connect pairs of degree 3 nodes. Such paths represent veins of *P. polycephalum*, while degree 3 nodes model veins joining together to form junctions. Figure 4.1 shows a schematic detail of a typical *P. polycephalum* graph.

At this point one could replace the paths by edges that directly connect their start and endpoints, thus obtaining 3-regular graphs. For some observables this is advantageous and natural, since degree 2 nodes have no clear meaning in real *P. polycephalum*. For instance, we ignore degree 2 nodes in our percolation experiments and when computing the degrees of face cycles. For others, however, one may exploit the fact that a better geometric interpolation of the real *P. polycephalum* veins can be achieved by taking into account paths of degree 2 nodes. We make use of this fact when it comes to path and cut properties as well as to computing the area and circumference of face cycles.

In this work, we study the properties of face cycles in *P. polycephalum* [110]. To this end, we compute the dual graphs of all available graphs. Since *P. polycephalum* graphs are planar graphs with a naturally embedding in the plane, their geometric duals can be constructed by placing a (dual) node in each region, including the exterior region. If two regions share an edge, their corresponding (dual) nodes are joined by an (dual) edge. The dual graph of a planar graph is again planar. To compute the duals we rely on a planar face traversal routine described in LEDA [110].

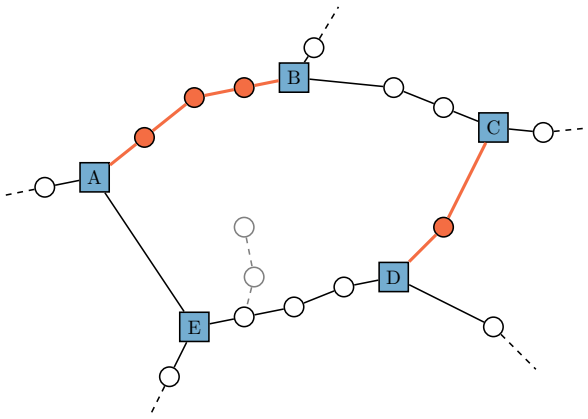


Figure 4.1.: Schematic representation of a *P. polycephalum* graph. Labeled blue squares form a face cycle (region) representative of veins and junctions of the slime mold. The face cycle is of degree 5 as it contains 5 degree 3 nodes. They are connected via 5 distinct paths of varying length, two of which are highlighted in orange. Paths consist of circular nodes of degree 2 which interpolate the veins of *P. polycephalum*. Degree 2 nodes often are the product of removing dead-end paths, one of which is indicated in gray extending into the interior of the face cycle near node *E*.

4.2.3. Statistical Methods

Statistical quantities are reported with their associated errors. Unless noted otherwise in the text, we either show bootstrapped standard errors of the mean or errors reported by fit routines where applicable. For the percolation experiments sample sizes were large enough to estimate the errors directly. Fits reported are least square fits, respectively weighted least square fits when errors where taken into account. For kernel density estimation we either rely on `R` or the `numpy.stats` packages [88], [143].

Statistical distribution testing was done using `R`, in particular using the excellent `fitdistrplus` and `bimodalitytest` packages [50]. The process of determining the theoretical distribution of an observable $\mathcal{O}$ from empirical samples is as follows. For every data series we compute the empirical distributions of $\mathcal{O}$ for the individual graphs as well as the cumulative empirical distribution taking into account all graphs within the given series. Furthermore we fit the obtained empirical distributions with several common distribution functions. To choose suitable candidate distributions we rely on visual inspection of the empirical distributions and their indicative Skewness-Kurtosis plots when no guiding knowledge of the underlying generating processes is available. To evaluate the quality of the resulting fits we rely on four classical goodness-of-fit plots [47]: the density plot, the CDF plot, the Q-Q plot and the P-P plot. For a detailed description see [50]. Relevant plots supporting the results presented in this chapter can be found in Appendix C.

4.3. Results

In the following we discuss a representative selection of results based on the Kist Europe data set. It is designed to illustrate the most important findings and to familiarize the reader with the type of information that is contained in the full set of results which is available for inspection and download at the Smgr project page. The full set contains plots of individual and cumulative results of all computed observables for all available data series, down to the level of singular *P. polycephalum* graphs. Furthermore fit parameters of all applied candidate fit distributions are available together with their goodness-of-fit plots. In addition to that tools and instructions are added which allow the interested reader to investigate aspects of the data that go beyond the scope of this chapter.

4.3.1. Path Properties

Recently, the distributions of basic observables including the length and the width of veins of *P. polycephalum* graphs have been studied [22]. Here we revisit earlier findings in an attempt to reproduce them. Surprisingly, despite (or because of) the increased amount of data available to us compared to earlier work, we were not able to unambiguously determine the underlying distributions of the observables under

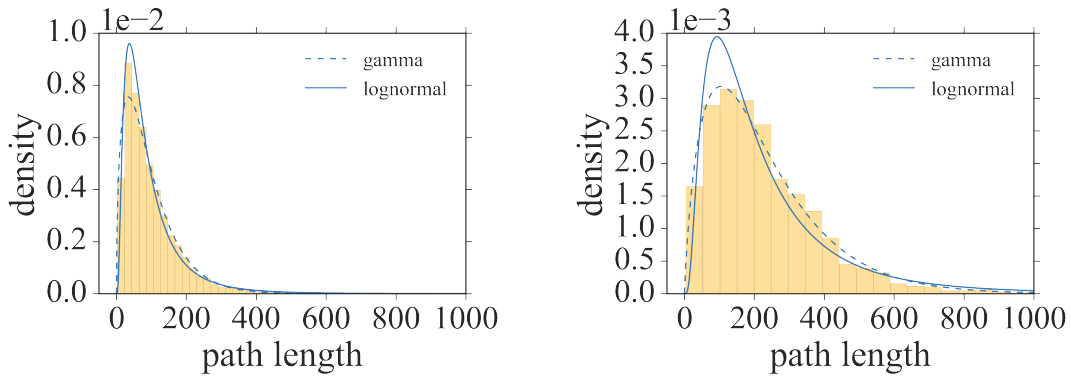
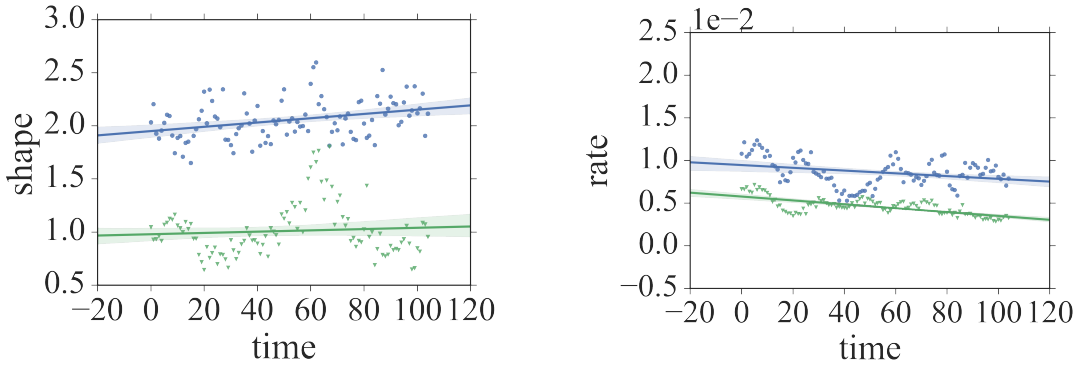


Figure 4.2.: Empirical cumulative path length distributions for `data series 22` (a) and `data series 35` (b). Associated gamma and log-normal fits are shown. Abscissa in units of pixel.

consideration. Instead we consider a set of valid candidate distributions and report on the most promising one in detail.

Let us define the length of a path as the sum of the length of the individual edges it contains. In Figure 4.2 we compare the cumulative empirical path length distributions of `data series 22` and `data series 35`. An inspection of the density histograms of `data series 22`, Figure 4.2a, does not allow a clear distinction between a candidate log-normal or a gamma distribution. At the same time `data series 35`, Figure 4.2b, seems to tend towards a gamma distribution. Note that the distribution of `data series 22` indicates very dense networks dominated by short paths while the probability mass in `data series 35` has started to shift towards longer paths. This is a result of the so-called coarsening process characteristic to *P. polycephalum* [22].

We choose the case of `data series 22` to illustrate the process we apply to deal with the observed ambiguities in detail. To do so we compare the goodness-of-fit plots associated with fitting a gamma, log-normal distribution and other candidate distributions in Figure C.1. Empirical and theoretical CDFs are in excellent agreement for both gamma and log-normal fits. The same holds true for the P-P plot where empirical and theoretical probabilities coincide, illustrating that both fits do not suffer from "lack-of-fit" in the center of the distribution. Looking at the Q-Q plots however, differences between empirical and theoretical quantiles can be seen, indicating that both gamma and log-normal fits are lacking in the right tails of the distributions. The gamma fit could be favored in the end, since it is not as lacking as the log-normal fit when it comes to the right tails, i.e. the longer path lengths. We remark that an exponential distribution was attributed to path lengths in previous work [22]. Our data indicates that it is a possible choice, provided a good description of the right tail is desired and the comparatively weak performance in the center of the distribution is not an issue.



(a) (b)

Figure 4.3.: Time dependence of shape s (a) and rate r (b) for `data series 35` (blue circles) and `data series 58` (green triangles). Linear fits are shown to guide the eye. Abscissa in units of 120 s.

We stress at this point that in order to assess the match between data and fit it is not sufficient to check the agreement between theoretical and empirical distributions alone. Since the latter is given as a density histogram, our perception can easily be mislead by an unfortunate choice of bin sizes. To mitigate this problem we rely on the Freedman–Diaconis rule to determine optimal bin sizes [64]. In combination with the goodness-of-fit plots a reliable judgment of fit quality can thus be achieved.

The example of `data series 22` is illustrative of the difficulties in determining descriptive theoretical distributions we encounter for almost all our data series. Given the available empirical data it is not possible to choose a single distribution from the proposed candidates that captures the tails and the centers equally well for all data series. As a result, depending on what part of the data is of interest one may opt for different theoretical distributions to maximize the usefulness of the description.

It is our impression however, that the gamma distribution yields a good compromise between center and tail accuracy in most cases. Thus we shall proceed with reporting the properties of the gamma fits in this manuscript. Details of properties of other fit options are available in the complete collection of results.

Given a time-ordered data series, one may study the obtained fit parameters as a function of time. For a gamma distribution these are the shape s and the rate r of the distribution. Figure 4.3 compares the time development of the parameters obtained for two long-running data series. Despite the considerable fluctuations, it can be seen in Figure 4.3a that the shape has a tendency to increase, while the rate is slightly declining as shown in Figure 4.3b. Since the mode of the gamma distribution is given by $(s-1)/r$ for $s > 1$, its value is increasing with time for both data series. It follows that the most likely path length to be observed increases in value as time moves on. This is accompanied by a shift of probability mass towards longer paths, which is consistent with network coarsening.

In addition to the time development of the fit parameters within distinct data series, we investigate the obtained values across the entire data set that has been fit. Figure 4.4 shows the scatter-plot of s and r and the associated estimated kernel density. The result illustrates what characteristic ranges of fit parameters, and by extension distributions, one may expect to encounter when it comes to path lengths in *P. polycephalum* graphs. Given the location of the density maximum, we can compute the mode of the associated gamma distribution and obtain 107.2389 pixel.

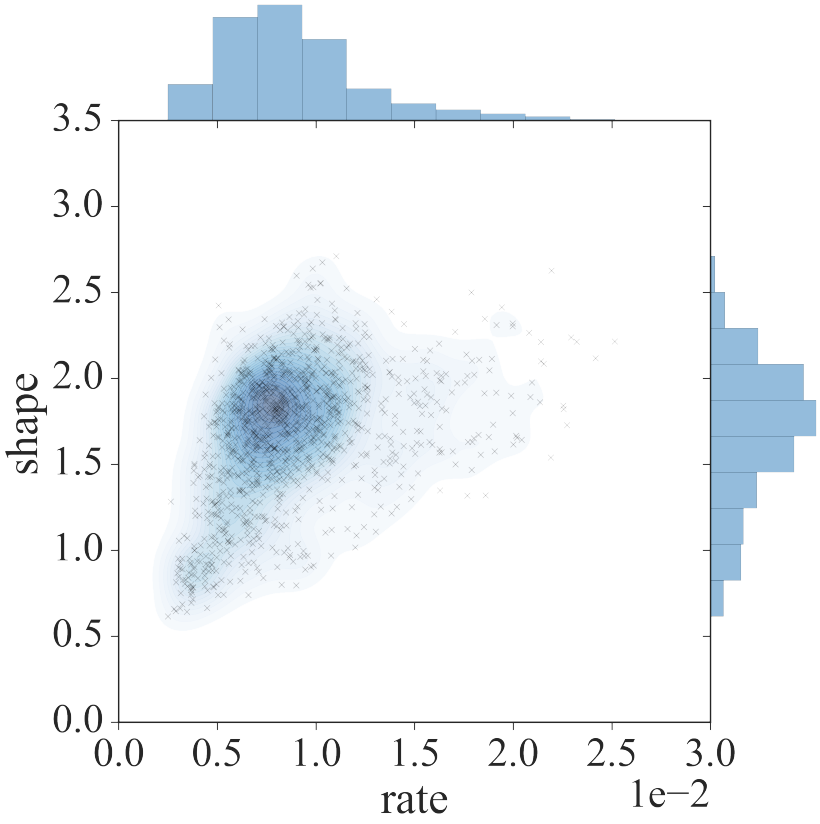


Figure 4.4.: Scatter-plot and kernel density estimates of shape s and rate r for gamma fits of empirical path lengths using the entire dataset. The maximum of the density is located at $s = 1.8403$ and $r \approx 0.7836 \times 10^{-2}$.

After detailing the procedure applied to the path lengths we proceed by investigating the average path widths and the path volume in the same manner. Since the procedure is the same, we keep the description more concise from now on.

Let us define the average path width as the average over the individual edge widths a path contains. In Figure 4.5 we compare the empirical path width distributions of `data series 22` and `data series 35`. Yet again an inspection of the density histograms alone does not allow to decide between the two most promising candidate distributions and we look to the goodness-of-fit plots to learn more. Figure C.2 shows that for `data series 35`, both gamma and log-normal fits do not suffer from lack-of-fit in the tails of the distribution. Furthermore, empirical and theoretical CDFs

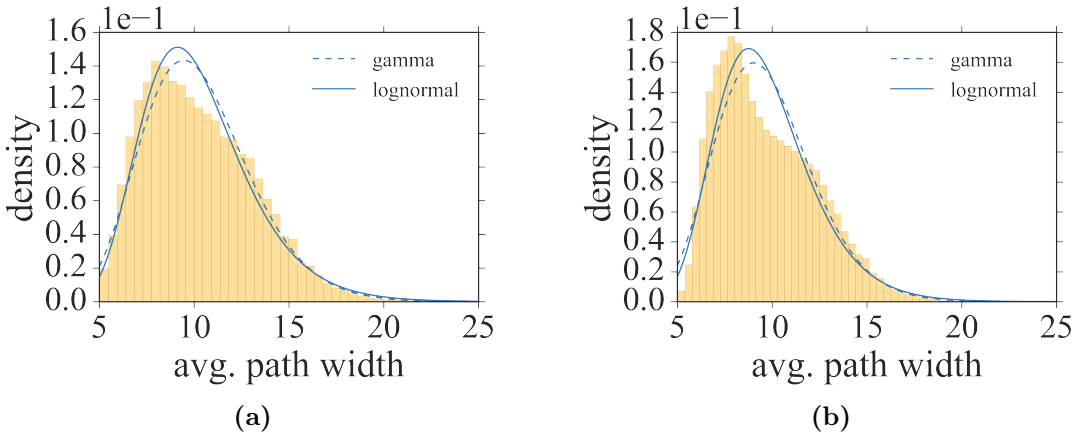


Figure 4.5.: Empirical cumulative distributions of average path widths for `data series 22` (a) and `data series 35` (b). Associated gamma and log-normal fits are shown. Abscissa in units of pixel.

agree. At the same time, the P-P plot indicates that both options are lacking in the center of the distribution. Both gamma and log-normal fits yield almost the same fit quality, even a normal fit yields a usable description. In the end, we favor the gamma fit yet again, because it delivers slightly better Q-Q plots.

Taking into account the whole data set, the theoretical distribution of the average paths widths remains undecidable. The differences between gamma and log-normal fits are minute and thus the fact that earlier work attested a log-normal behavior of the widths is still supported by our findings [22]. We remark that, the pronounced "shoulder" seen in Figure 4.5b suggest to the eye that a bimodal distribution could be responsible for the observations. Thus we attempted to fit the path widths with a Gaussian mixture. However, we failed to satisfy the associated statistical tests conforming that a Gaussian mixture is indeed the correct underlying distribution.

Next we report the gamma fit parameters as a function of time. Figure 4.6 compares the time development of the parameters obtained for `data series 35` and `data series 58`. Note that, for both series a linear relationship between shape s and rate r can be observed. For `data series 35` we find that $s(r) = (8.0634 \pm 0.1160)r + (2.4148 \pm 1.8410)$ holds. At the same time we obtain $s(r) = (9.1390 \pm 0.1420)r + (0.5859 \pm 0.1870)$ for `data series 58`.

Figure 4.7 illustrates that a similar linear relation holds across all the available data. It is given by $s(r) = (8.3761 \pm 0.0400)\, r + (2.0081 \pm 0.0640)$. We believe that this relation follows naturally from the fact that the path widths are relatively small in value and their distributions show only little variation from one data series to the next. In this regards the path widths behave decisively different from the path lengths, which show much larger variation.

We may use the given linear relation to parametrize the gamma distributions as a function of r and study the effect on the mode and the tails of the distribution as r

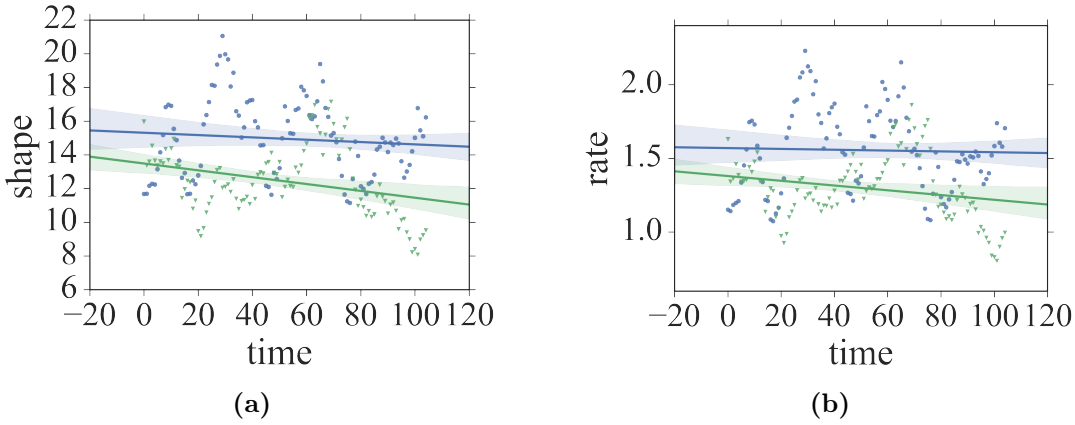


Figure 4.6.: Time dependence of shape s (a) and rate r (b) for `data series 35` (blue circles) and `data series 58` (green triangles). Linear fits are shown to guide the eye. Abscissa in units of 120 s.

assumes values in $[0.5, 3]$. We find that the position of the mode is confined within a relatively small interval of $[8.467, 10.893]$. Furthermore, as r increases the probability mass concentrates around the mode as it is traversing the given interval. At the same time $s(r)$ increases, causing the gamma distribution to approach a normal distribution with $\mu = s/r$ and $\sigma^2 = s/r^2$. Thus the distribution of average path widths smoothly interpolates between a gamma distribution for small r and normal distribution for larger r. For $r = 1.2799$, which is associated with a maximum in the density plot of Figure 4.7, the mode takes a value of 9.1637 pixel. The fact that the gamma distribution naturally captures the transition to a normal distributions increases our confidence that the gamma distribution yields a correct description of the data.

We point out that within the observed range of fit parameters, the probability mass in the right tail of the distribution is largest for small values of r. Thus one would expect the rate to drop as time goes by if larger path widths are to become more probable as the slime mold networks undergo coarsening. Looking at Figure 4.6, `data series 58` does exhibit this behavior. At the same time `data series 35` does not seem to show any long term changes in the rate, which would indicate that the distribution remains stable. Taking into account the entire data set, we repeatedly find both types of behavior.

We remark that earlier work proposed the normal distributions as a suitable description for path widths of coarsened networks that contain a significant number of paths of large width [22]. This is in contrast to the results above, which indicate that within the range of observed parameters, normal distributions are associated with networks where large average path widths are comparatively rare.

While paths are the atomic constituents of *P. polycephalum* their weights do not tell us anything about their arrangement within the graphs. To gain some insight we move on to study the next larger building blocks, namely cycles formed by paths,

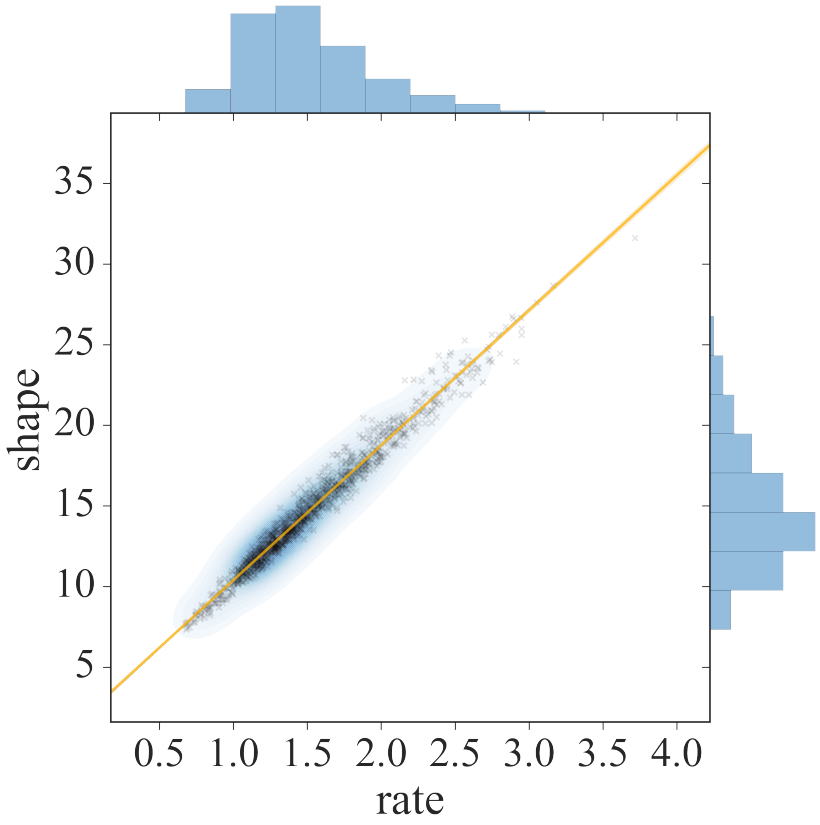


Figure 4.7.: Scatter-plot and kernel density estimates of shape s and rate r for gamma fits of the empirical average path width using the entire dataset. The maximum of the density is located at $s \approx 12.7177$ and $r \approx 1.2799$. A linear fit is shown in yellow.

in the next section.

4.3.2. Face Properties

When cultured under the right conditions on flat surfaces, *P. polycephalum* forms (almost perfect) 3-regular planar graphs that have a natural geometric embedding given by the euclidean coordinates of its nodes in the plane [22]. Each face of the embedding, with the exception of the outer face, is bounded by a cycle of edges. We call such cycles *face cycles* or *faces* for short, see Figure 4.1. In other words, faces can be regarded as basic graph building blocks that correspond to loops formed by the veins of the organism. To characterize the structures formed by *P. polycephalum*, we study various statistical properties of its faces. Given a graph, all of its faces can be obtained in polynomial time by computing its dual [110].

Let us stress at this point that the number of faces in our graphs is significantly smaller than the number of nodes or edges. In fact, using Euler's formula for planar graphs and the fact that *P. polycephalum* graphs are 3-regular it is easy to see that the number of faces f is given by $f = 2 + \frac{n}{2}$. As a consequence any statistical observable we compute over the set of faces suffers from smaller sample sizes as compared to path properties for example. Throughout all of our measurements, we ignore outer faces.

Let us define the degree of a face cycle as the number of degree 3 nodes it contains. The cumulative face degree distribution for `data series 35` is given in Figure 4.8a. It clearly shows that faces of degree 4, 5 and 6 account for the majority of faces. These findings are consistent across all available data sets. With regards to determining the theoretical distributions, gamma and log-normal fits yield comparable goodness-of-fit plots, with gamma fits being slightly favorable, see Figure C.3. Another indication to go with gamma fits is given by similar studies that focus on faces of 2-dimensional Voronoi graphs [80], [169]. Since Voronoi graphs are planar, 3-regular and strongly resembling *P. polycephalum* graphs, gamma fits can be expected to be a good choice. It is interesting to note that for Voronoi graphs faces of degree 6 are known to have the largest expectation. We remark at this point, that the time development of the fit parameters shown in Figure 4.8b shows rather large values of shape, which indicates that the gamma distribution is approaching a normal distribution. Furthermore, a linear relationship between shape s and rate r is apparent.

Figure 4.9 shows that this relation is valid across the entire dataset and is given by $s(r) = (5.3403 \pm 0.0210)r + (0.3508 \pm 0.0370)$.

Given this relation we may parametrize the gamma distribution as a function of r and study the behavior of the mode as r takes on values in $[1, 3]$. Again we find that the possible positions of the mode are restricted to a rather small interval given by $[4.6911, 5.1239]$. For $r = 1.6255$, which is associated with the maximum in the density plot of Figure 4.9, the mode takes a value of 4.941. Rounding these values to the nearest integer suggests that a mode of 5 is most likely to be found for a randomly picked face of a *P. polycephalum* graphs.

Analogous to the discussion of the path widths, for a data series to exhibit a

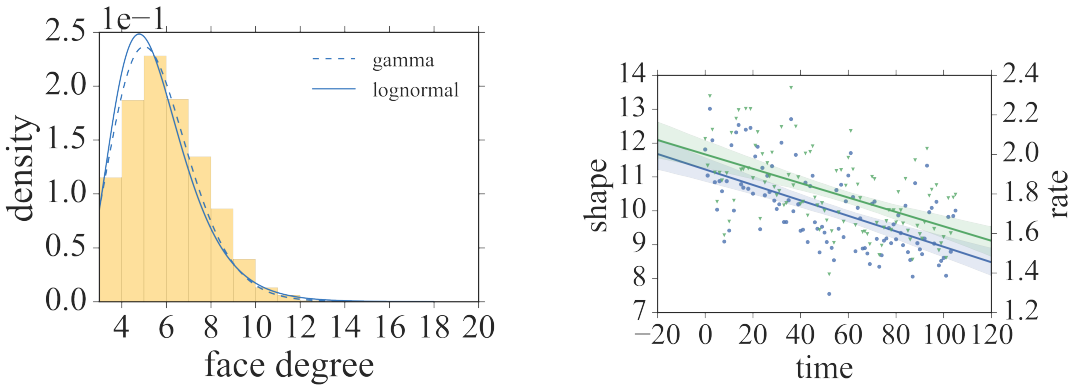


Figure 4.8.: (a): Cumulative empirical face degree distribution for `data series 35`. Candidate gamma and log normal fits are shown. (b): Gamma fit parameters shape (blue circles) and rate (green triangles) as a function of time for the same series. Linear fits are shown to guide the eye. Abscissa in units of 120 s.

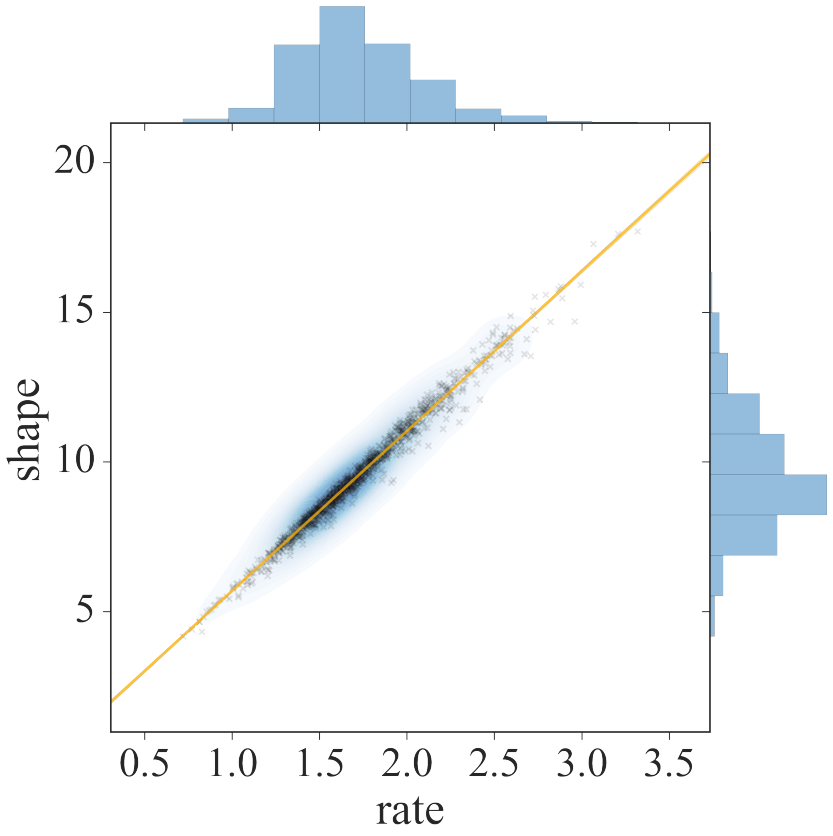


Figure 4.9.: Scatter-plot and kernel density estimates of shape s and rate r for gamma fits of the empirical face degrees using the entire dataset. The maximum of the density is located at $s \approx 9.0387$ and $r \approx 1.6255$.

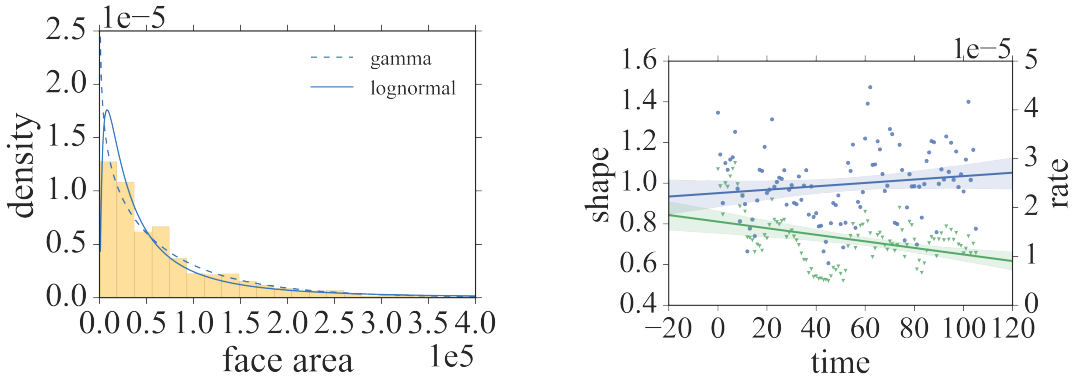


Figure 4.10.: (a): Cumulative empirical face area distribution for `data series 35`. Candidate gamma and log normal fits are shown. Abscissa in units of pixel^2. (b): Gamma fit parameters shape (blue circles) and rate (green triangles) as a function of time for the same series. Linear fits are shown to guide the eye. Abscissa in units of 120 s.

high density of large degree faces, small values of r are necessary. Figure 4.8b indicates that the underlying networks of `data series 35` are actively changing as to increase the number of large degree faces while keeping the distribution mode largely constant. While we find this behavior for many data series, there are others which seem to keep the degree distribution largely stable. The reasons for this are unclear at this point.

Next we investigate the empirical distributions of the areas enclosed by faces in more detail. The cumulative face area distribution for `data series 35` is given in Figure 4.10a. It's goodness-of-fit plots suggest that a gamma fit yields a reliable description of the empirical data, see Figure C.4. The time development of the associated fit parameters is given in Figure 4.10b. Note that the shape values fall below 1 repeatedly, which prevents us from using the mode to characterize the most likely face areas since it requires $s > 1$. However, analogous to earlier results, increasing values of s in combination with decreasing values of r indicate that the probability mass of the associated gamma distribution is shifting towards larger face areas. Given the time development depicted in Figure 4.10b, this is consistent with network coarsening.

We stress that, while a gamma distribution yields a good description of `data series 35`, for numerous other data series gamma fits perform poorly in the center of the distributions compared to a log-normal distribution. The fact that the face area data exhibits a large degree of variance in combination with the reduced size of the available data points hampers our ability to properly select a fitting theoretical description. Results associated with log-normal fits are available in the full data collection.

A global impression of what ranges of fit parameter one can expect to encounter

for gamma fits of the empirical face area distributions is given in Figure 4.11.

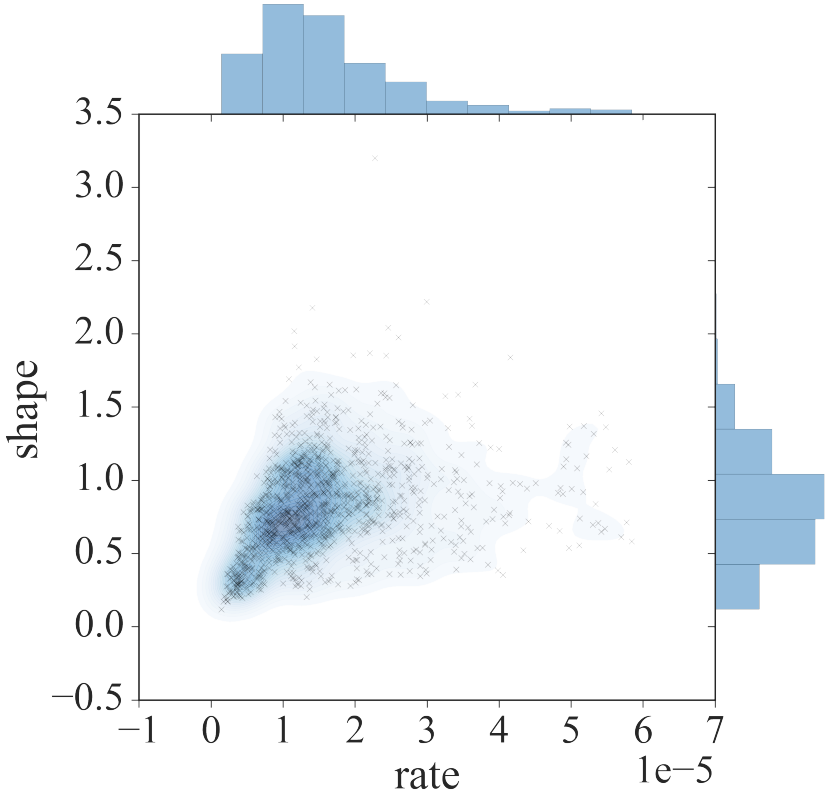


Figure 4.11.: Scatter-plot and kernel density estimates of shape s and rate r for gamma fits of the empirical face areas using the entire dataset. The maximum of the density is located at $s \approx 0.7022$ and $r \approx 1.0381 \times 10^{-5}$.

Finally, Figure 4.12 shows how the sum of the area covered by all faces across all data series is distributed with respect to face degree. It can be seen that faces of degree 6 and 7 account for the majority of the area covered, while faces of degree 5 and 8 play a smaller role in comparison.

Let us now move on to the empirical distributions of the circumferences of faces. The cumulative face circumference distribution for `data series 35` is given in Figure 4.13a. Here, goodness-of-fit plots indicate that a gamma distribution is clearly superior to other candidate fits, see Figure C.5. Surprisingly, this is true for all our datasets. Figure 4.13b shows the time development of rate and shape, indicating that the distribution is shifting towards faces of larger circumference. This time development can be seen for a significant fraction of the available data.

Finally, a global impression of the obtained fit parameters is given in Figure 4.14. Since the shape values are larger than 1, we may obtain the value of the mode of the distribution associated with the global maximum of the density to be 627.1050 pixel.

Let us complete our report on face properties by characterizing the geometric shape of the faces observed in *P. polycephalum*. In particular we want to quantify

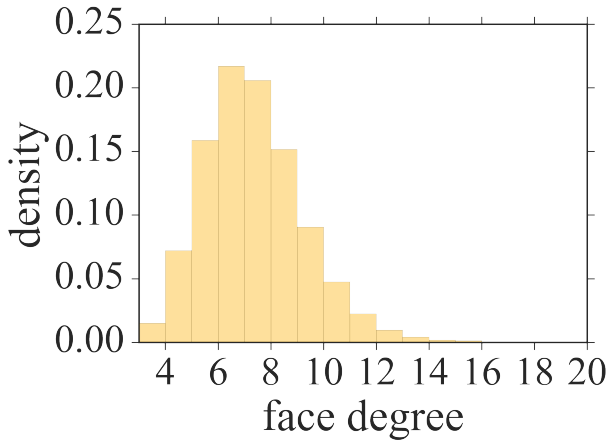


Figure 4.12.: Global cumulative histogram of face degree weighted by face area. The area under the histogram, which is normalized to one, corresponds to the sum of face areas across all observed faces, i.e. the total area covered by 1998 *P. polycephalum* graphs.

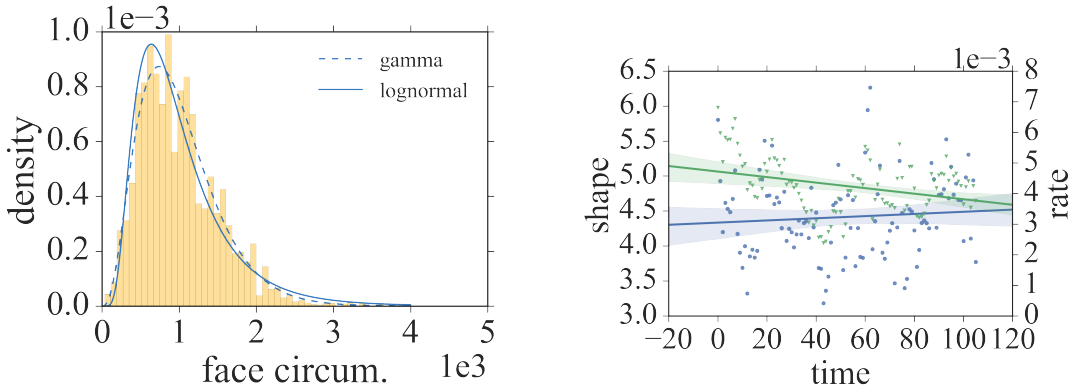


Figure 4.13.: (a): Cumulative empirical face circumference distribution for `data series 35`. Candidate gamma and log normal fits are shown. Abscissa in units of pixel. (b): Gamma fit parameters shape (blue circles) and rate (green triangles) as a function of time for the same series. Linear fits are shown to guide the eye. Abscissa in units of 120 s.

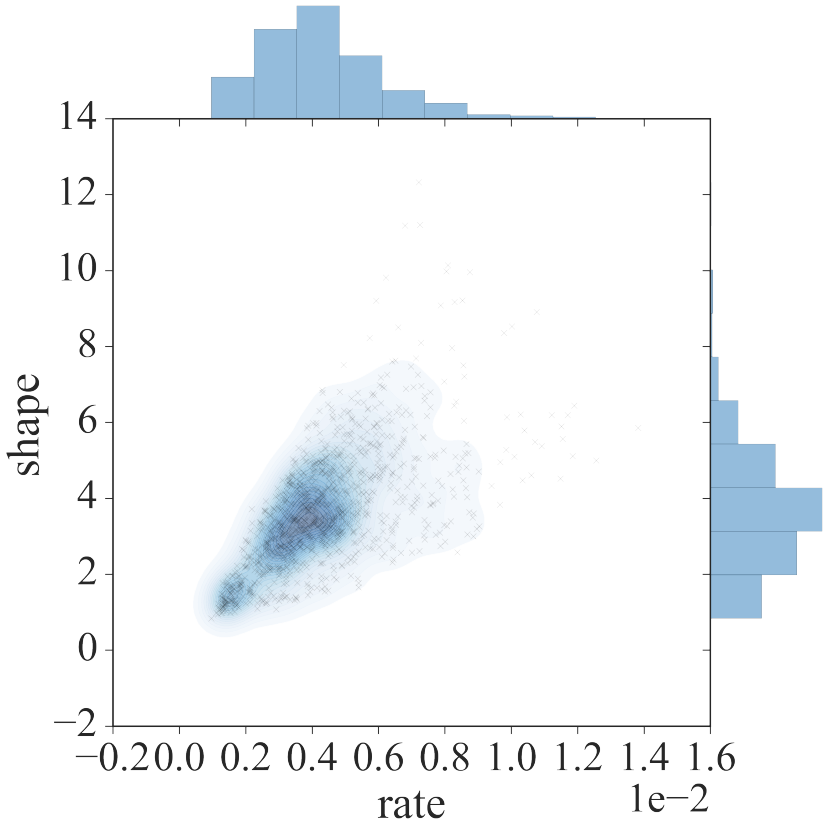


Figure 4.14.: Scatter-plot and kernel density estimates of shape s and rate r for gamma fits of the empirical face circumference using the entire dataset. The global maximum of the density is located at $s \approx 3.4247$ and $r \approx 3.8665 \times 10^{-3}$.

the resemblance between a face of degree k and a regular polygon consisting of k edges. We do so by defining the dimensionless *roundness* $r(f)$ for any face f as

$$r(f) = \frac{4\pi A(f)}{C(f)^2}\,, \tag{4.1}$$

where $A(f)$ and $C(f)$ denote face area and face circumference respectively. For $k \to \infty$, we have $r(f) = 1$ if f forms a circle, and $0 < r < 1$ if f has any other shape. The closer the polygon defined by f is to a circle, the closer r is to 1. To obtain an equivalent statement for faces of finite k we need to normalize $r(f)$ with respect to k. The normalization factor $r_{max}(k)$ is defined as the roundness of a regular polygon with k edges. It is given by

$$r_{max}(k) = \frac{\pi}{k}(tan(\frac{\pi}{k}))^{-1}\,. \tag{4.2}$$

Hence, we obtain the *normalized roundness* of a face f of degree k as

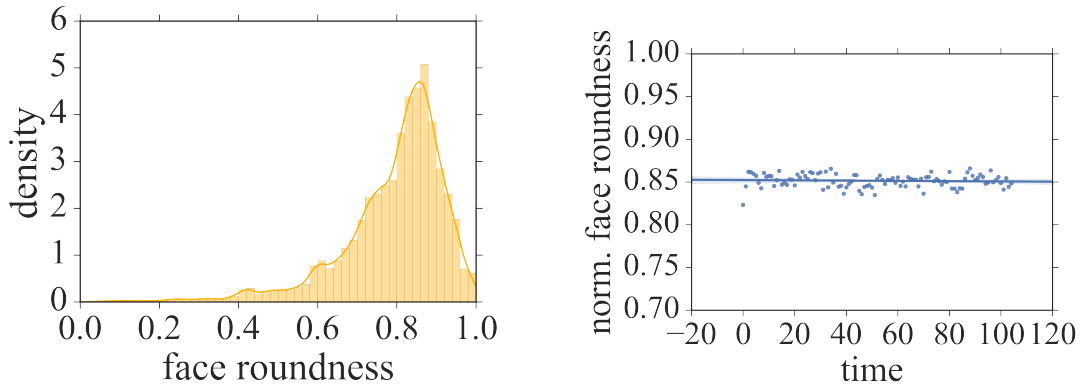
$$R(f,k) = \frac{r(f)}{r_{max}(k)}\,. \tag{4.3}$$

Thus for any k it holds that, the closer $R(f,k)$ is to 1, the closer a face f of degree k is to being a regular polygon consisting of k edges.

With this definition we give the empirical distribution of the $R(f,k)$ for all faces in `data series 35` in Figure 4.15a. The distribution is sharply concentrated around 0.8513 ± 0.0077 indicating a tendency for the faces of the underlying graphs to approach regular polygons. A value of 0.8532 ± 0.0205 is found when the entire dataset is considered. Figure 4.15b indicates that the mode of the roundness shows only small variations in time for `data series 35`. The fact that this behavior is consistent across all datasets makes the roundness a very distinct and robust quantity describing the face structure of *P. polycephalum* graphs.

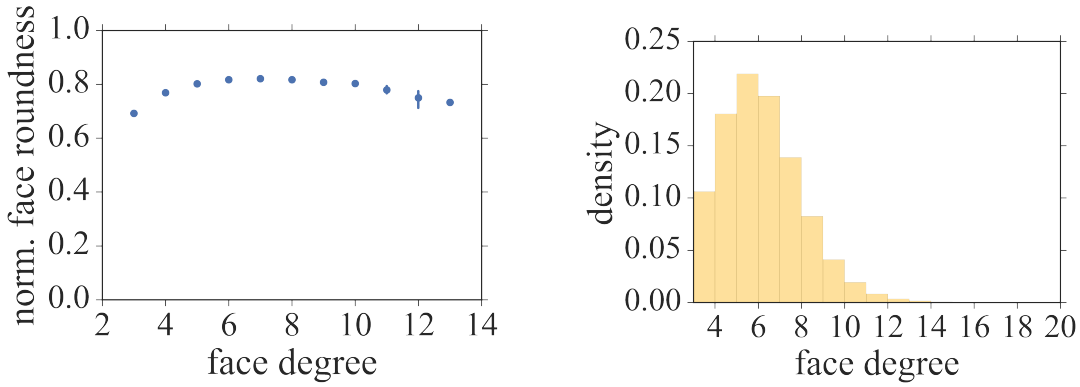
It is interesting to investigate how the roundness is connected to the degree of the faces under consideration. Figure 4.16a shows the roundness values averaged over classes of faces that have the same degree. We find that the roundness steadily increases with k to reach a maximum of 0.8206 ± 0.0014 for $k = 7$. After the maximum is attained the roundness falls off again, albeit slightly. We may also ask which class of faces contributes the most to the global sum of observed roundness values. Figure 4.16b shows the histogram of face degrees weighted by face roundness. It indicates that faces of degree $4, 5$ and 6 account for the majority of the global sum of roundness. Since these faces form a large fraction of all available faces, we can now quantify the degree to which *P. polycephalum* networks are formed by components that resemble regular polygons.

We stress at this point that looking at roundness numbers by themselves is not very instructive. Their usefulness, however, becomes apparent in combination with other face properties when it comes to comparing *P. polycephalum* graphs amongst themselves or with other structures. While the roundness gives a measure of what shapes of faces are present in a graph, face area and circumference add a physical



(a) (b)

Figure 4.15.: (a) Cumulative normalized face roundness distribution for `data series 35`. The kernel density estimation of the cumulative distribution is show as a solid line. The value of the abscissa at the maximum of the KDE serves as an estimate for the mode of the underlying unknown theoretical distribution. (b) The development of the estimates of the mode for the individual networks of `data series 35`. Abscissa in units of 120 s.



(a) (b)

Figure 4.16.: (a) Cumulative normalized face roundness averaged over classes of faces with identical face degree. Classes of size smaller than 30 are excluded as bootstrapping error-bars ceases to be reliable for small sample sizes. (b) Histogram of face degrees weighted by the associated roundness values. Here the area under the histogram, which is normalized to one, corresponds to the sum of roundness values across all observed faces.

dimension to them. Together with the degree distribution of faces a comprehensive description of the entire network of *P. polycephalum* in terms of its basic two dimensional building blocks is available. Since this description can be computed efficiently for any weighted planar graph, a quantitative comparison with other planar networks becomes possible.

4.3.3. Cut Properties

In this chapter we study the extend of spatial homogeneity of *P. polycephalum* networks with respect to two special directions: First, the direction in which the organism expands, i.e. its growth direction and second, the direction perpendicular to it. To do so we use the graph theoretic concept of *cuts*. A cut defines a partition of the nodes of a graph into two disjoint subsets. Furthermore, it determines a *cut-set*, a set of edges with the property that each edge has one endpoint in each of the two node subsets. These are the edges crossing the cut. Since we are interested in the cut-set only, let us refer to it simply as cut from now on.

We distinguish vertical and horizontal cuts, which we determine in the following way. First we take the dimensions of the original images containing the graphs of a series. For each series this yields a rectangle of a certain size. Next we partition these rectangles by slicing them into 100 identical smaller sub-rectangles. Here we distinguish between horizontal and vertical slices. For each image we overlay the resulting horizontal/vertical lines with the graph extracted from the image. Each horizontal/vertical line crosses through a unique set of edges which defines a cut. Thus, for each graph in a series we obtain 100 equidistant horizontal/vertical cuts. Vertical cuts proceed from left to right, i.e. in the direction of growth. Horizontal cuts proceed from top to bottom, i.e. perpendicular to the direction of network expansion, see Figure C.6. Since cuts are defined consistently within a series, we may study how interesting cut properties change as we proceed through the cuts. By comparing the two types of cuts we can shed light on structural properties of the network with respect to its growth direction (vertical cuts) or perpendicular to it (horizontal cuts). We define three such properties: the size, the width and the length of a cut. The size of a cut is given by the number of its cut edges. The width of a cut is the sum of the widths of all the edges in the cut while the length of a cut sums over the lengths of the cut edges.

We begin by discussing the size of cuts for both horizontal and vertical cuts. Figure 4.17a and Figure 4.17b give the results for two characteristic data series. For any given cut, the ordinate shows the value of the cut averaged across all graphs in the respective series while the abscissa enumerates the cuts in a sequence of cuts. Both `data series 45` and `data series 35` exhibit significant fluctuations in the actual cut values. This is to be expected since the network keeps changing as time advances. Linear fits are applied to illustrate trends in the development of the cut values. Looking at `data series 45`, it can be seen that, within fluctuations, vertical cut values remain rather stable while horizontal cuts show only a slight tendency to increase in value. The behavior of the cut values of `data series 45` indicates

that it has a large degree of spatial homogeneity. The situation is different for `data series 35` where vertical cuts show a trend to decrease while horizontal cuts tend to increase at the same time. The fact that the behavior of the lengths and widths of cuts closely mirrors the cut sizes, indicates that the properties of the edges that are being cut are homogeneous throughout the network. Differences in cut lengths or widths are predominately due to differences in cut sizes.

Taking into account the entire data set we find that horizontal cuts remain largely constant, while the vertical cuts frequently show significant changes. The latter include both increasing and decreasing cut values.

The fact that some series exhibit distinctly different trends between horizontal and vertical cuts is curious. Why are the cuts in growth direction more stable than the cuts perpendicular to it? For an attempt to explain this, we take a closer look at the cut widths and their significance with regards to fluid transport. Assume that for the organism to advance its growing front, it needs to actively direct its flow towards the general direction it wants to expand towards. Given the experimental setup underlying the data we explore, this means the organism is advancing along the positive x-axis with an apical zone approximately parallel to the y-axis, see Figure C.7. Thus any fluid flow proceeding along the x-axis must pass trough a sequence of vertical cuts. By virtue of the min-cut max-flow theorem, it is known that the maximum amount of flow that may pass through such a sequence is equal by the value of the minimum capacity cut. Intuitively, any attempt to push flow through a sequence of cuts is always capped by the amount of flow the bottleneck cut admits, see dashed lines in Figure 4.17e and Figure 4.17f. In light of this fact it seems natural for the sequence of vertical cuts to be balanced in width because this configuration allows for efficient fluid transport. We stress that in order to increase the fluid transportation capabilities across a sequence of cuts, only improvements of the minimum cut are effective. Increasing the capacity of any other cut implies higher material costs without any immediate benefits in terms of directed fluid transport. Thus balanced cuts provide the optimal ratio of material cost to transport capacity.

While vertical cuts are well-balanced for almost all series, it is interesting to note that a significant number of series show unbalanced horizontal cuts. We find such cuts across all our data groups, which leads us to speculate that a significant structural difference between the two major directions of fluid transport is present. The fact that *P. polycephalum* favors balancing vertical cuts over balancing horizontal cuts when expanding along the positive x-axis indicates that exploring new territory, i.e. foraging, takes precedence over improving circulation within the existing network.

Note, that our reduction of the complex flow patterns observed in *P. polycephalum* to two cardinal directions is rather crude. However, our results yield a first quantitative indication that the growth direction is indeed intimately connected with distinct structural properties. It is worthwhile to think about more sophisticated methods trying to establish links between network growth and network structure in the future.

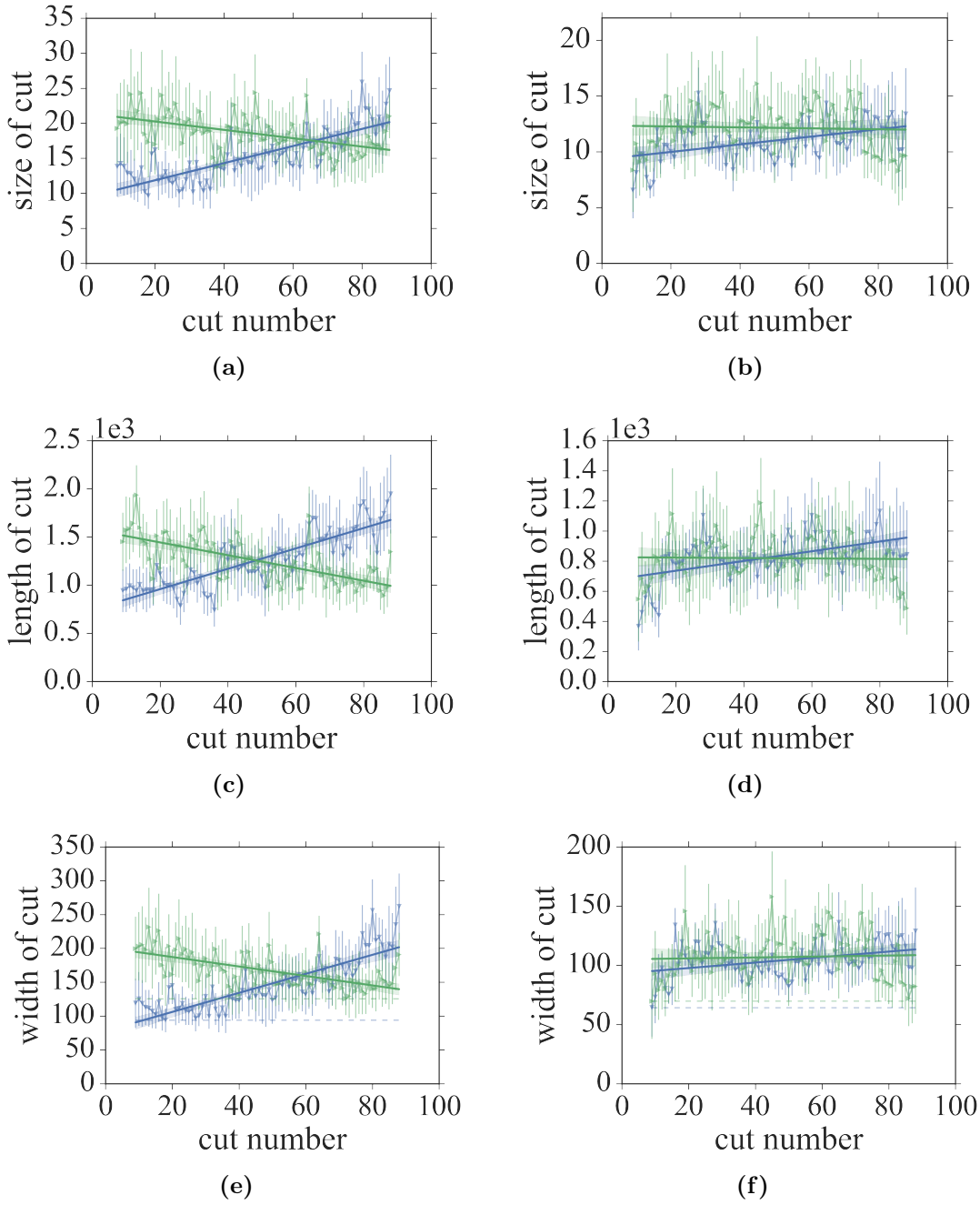


Figure 4.17.: Cumulative cut properties for `data series 35` (l.h.s. row) and `data series 45` (r.h.s. row). The errors denote one sigma standard deviations. Solid lines give linear fits to the data intended to guide the eye. Green triangles pointing downwards belong to horizontal cuts (moving from top to bottom) while blue triangles pointing to the right illustrate vertical cuts (moving from left to right). Dashed lines illustrate the values of minimum width cuts.

4.3.4. Percolation

Previously, concepts from percolation and graph theory where successfully combined to study the assembly of macroscopic *P. polycephalum* networks from microscopic slime mold fragments, called microplasmodia [60], [61], [62]. Here we shall take the reverse approach to add another item to the collection of characteristic *P. polycephalum* properties. To do so, we start with a well-formed *P. polycephalum* network and gradually disassemble or "damage" it until it becomes maximally fragmented. A network G is damaged by removing nodes or edges successively in a random order. Removal continues until the graph is empty. The result is a sequence of graphs $\mathcal{S} = G, G_1, \ldots, G_k$ with a strictly decreasing number of edges or nodes. For each graph in the sequence one may then compute various observables and study their behavior with respect to the inflicted damage. This process has been successfully applied to various networks to determine the extend to which networks can remain operable despite of its components sustaining damage [40], [41].

Let us focus on random node and edge percolation. We define an occupation probability p such that any given node u or edge e respectively is present in the graph with probability p. Equivalently, it is removed with probability $1-p$. For any graph $G_i \in \mathcal{S}$, we study the behavior of two important observables: the sizes of the largest and second largest connected components in G_i as p is varied from 0 to 1. We repeat this measurement 1000 times for each graph in $\mathcal{S}$ and obtain averages and statistical errors for the measured quantities.

Figure 4.18 shows the results for two graphs selected from `data series 22` and `data series 35` respectively. Let us first discuss the size of the largest connected component (LC). From the theory of percolation it is known that this quantity acts as an order parameter, signaling that as a certain critical occupation probability p_c is reached, the system undergoes a (topological) phase transitions. For $p < p_c$ the size of the LC is very small, implying that the graph is fragmented, consisting of a large number of small connected components. Thus, it cannot function as a large connected unit. As we approach p_c the situation changes abruptly and the largest connected component quickly grows to encompass the whole graph. Starting at $p = p_c$ the largest connected component is said to percolate, indicating that the graph may functions as one connected unit. Thus, if one removes less than a $1 - p_c$ fraction of nodes or edges from a *P. polycephalum* network, it is guaranteed to contain a large connected component whose size is comparable to the system size. Hence, the value of p_c can be used to quantify the degree of damage a *P. polycephalum* network can sustain whilst being able to maintain communication between almost all of its nodes.

To determine the value of p_c we study the size of the second largest component (sLC). The theory of percolation predicts that this quantity diverges for $p = p_c$ on graphs of infinite size. On finite size graphs it becomes maximal. To obtain the location of the maximum, we fit the peak with a parabolic function

$$f(p) = a\ (p - d)^2 + c\,, \tag{4.4}$$

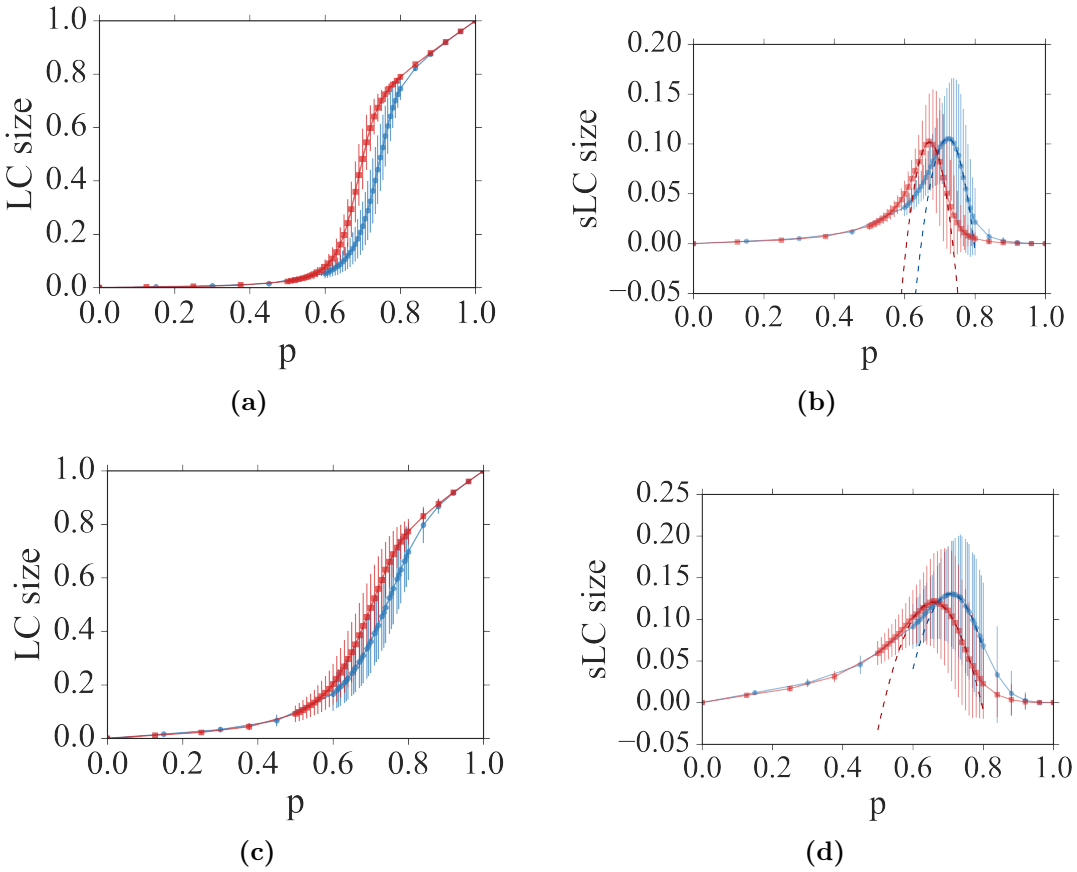


Figure 4.18.: Percolation properties for a sample graph from `data series 22` (top row) and from `data series 35` (bottom row). The abscissa gives the occupation probability p. The ordinate shows the normalized size of the largest connected component (LC, l.h.s. column) or the normalized size of the second largest connected component (sLC, r.h.s. column). Red squares denote random node percolation, blue circles random edge percolation. For the second largest component sizes, parabolic fits approximate the location of the peaks as illustrated by dashed lines. Errors show one sigma standard deviations.

where the value of the fit parameter d approximates the value of the critical threshold p_c. It is also apparent that the critical threshold for random node percolation is larger than the one for random edge percolation. This is a general property of percolation and serves as a sanity check, suggesting that our simulations were performed correctly. It also dictates that *P. polycephalum* graphs are more resilient to the removal of edges than to the removal of nodes.

It is known that the critical threshold p_c depends on the topology of the graph, the dimension of the space and the type of percolation, e.g. node or edge percolation. Figure 4.18 clearly shows that for a sample graph from `data series 22`, containing some of our largest graphs, the size of the LCC increases faster around p_c than for a graph from `data series 35` containing much smaller graphs. At the same time the peak in the size of the sLC is less pronounced for the smaller graphs. These so-called finite size effects are expected and introduce a dependence on the graph size to p_c. As a result, what we are measuring for individual graphs is an effective finite size percolation threshold $p_c(L)$. Here L denotes the number of nodes n in case of node percolation and the number of edges m in case of edge percolation. Naturally, we are interested in the so-called thermodynamic limit, i.e. $p_c(L)$ for $L \to \infty$.

To get an impression of the dependence of the system size L, Figure 4.19 shows the values of p_c for the collective data. It can be seen that the critical values vary with the size of the system as expected. For small L critical values are smaller than for larger L where they seem to become constant. Extracting this constant value for $L \to \infty$ is the objective of finite size analysis. In it, relying on the theory of critical phenomena, a scaling function is derived, which captures finite size corrections as a function of the system size L and a small set of critical exponents.

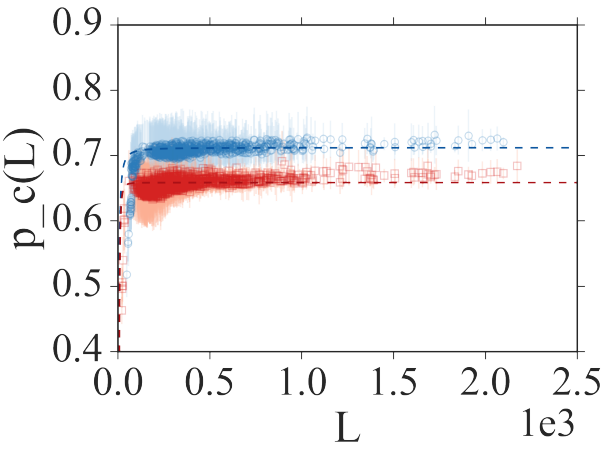


Figure 4.19.: Dependence of the critical threshold $p_c(L)$ for all available *P. polycephalum* graphs on the size of the graph L. Blue circles refer to random node percolation, red squares to random edge percolation. Dashed lines indicate scaling fits.

As a possible scaling law we propose an *Ansatz* of

$$p_c(L) = a + bL^c + dL^e + flog(L)^g\,, \tag{4.5}$$

where we add a logarithmic term to the polynomial expression to allow for subleading corrections, without which scaling fits may fail, particularly on small size systems. Furthermore we restricted the exponents c, e and g to negative numbers to obtain a finite value of $p_c(L)$ as $L \to \infty$. The resulting fits are shown in Figure 4.19. Note that for small L the fit is lacking as it overestimates the values of $p_c(L)$. Furthermore it appears that for large L the available data points are underestimated. The effect is small but seems to be systematic. It likely causes us to underestimate the final critical thresholds in the thermodynamic limit. The thresholds are $p_c = 0.7118 \pm 0.0544$ for random node percolation and $p_c = 0.6584 \pm 0.0217$ for random edge percolation. Note that compared to other numerical percolation results, the error of both quantities is very large and thus the obtained values are to be taken as a rough approximation only.

Let us discuss potential reasons for the large uncertainty in the critical values. First, the graphs we use might be too small. It is known that the reliability of percolation studies increases with larger system sizes. However, the values for $p_c(L)$ in Figure 4.19 appear to be relatively stable in the limit of large L. This does suggest that the graphs are large enough to observe finite size scaling properly. On the other hand, there is only a limited sample of larger graphs which this observation is based on. Also smaller graphs dominate our data in comparison to large graphs may lead to diminished accuracy of the fits. We have experimented with excluding the smaller graphs from the set, but imposing arbitrarily chosen size thresholds did not significantly improve the fits. A real improvement can probably only be achieved by increasing the number of large graphs that enter the analysis.

Second, it is possible that our choice of a scaling law is suboptimal. After searching the literature, we only found one alternative option to our scaling choice in an earlier percolation study focusing on graphs that are topologically similar to *P. polycephalum* networks. While the alternative looked promising on paper, it was not able to satisfyingly fit our data. Thus we had to eliminate this option. An additional downside of the general scaling law we proposed is the resulting difficulty in interpreting the fit results in terms of meaningful critical exponents. Further studies are necessary to obtain reliable results in this regard.

Third, it is conceivable that scaling fails because of a natural variability present in *P. polycephalum* graphs. It is apparent from Figure 4.19 that points with similar L can still differ considerably in $p_c(L)$. This is surprising because, if we assume that there is one complex physical process responsible for the observed topology of the graphs, one would expect it to produce networks that exhibit the same percolation properties for systems of identical size. Either this assumption is incorrect, or we must admit that other, possibly random, effects are responsible for the observed deviations. Candidates are minor variations in the experimental setup during culturing or other unknown environmental effects influencing the network formation of *P. polycephalum*.

Finally, let us compare the obtained approximate critical thresholds with those of two other structures which are planar, 3-regular and well-studied in the context of percolation: The honeycomb lattice with $p_c = 0.6962(6)$ and $p_c = 1 - 2sin(\pi/18) \approx 0.6527$ and the 2D Voronoi Tessellation[1] with $p_c = 0.71410(2)$ and $p_c = 0.666931(5)$ [24], [55]. Cited values denote node and edge percolation respectively. The similarity between the critical thresholds indicates that these structures exhibit a similar degree of resilience against random attacks. Where hexagonal lattices are clearly too regular to resemble *P. polycephalum*, Voronoi graphs are more interesting. It is intriguing to see that the percolation of *P. polycephalum* networks seems almost identical to percolation on Voronoi graphs. Are there similarities between the way *P. polycephalum* grows in search of food and the way Voronoi graphs form? It is worthwhile to investigate these question in detail in the future.

4.4. Discussion

In this chapter we investigate 36 time-series of *P. polycephalum* graphs based on the KIST Europe data set available in the SMGR. This amounts to a total of 1998 graphs under consideration, which to the best of our knowledge, makes this study the largest of its kind available today. We present a investigation of this data leading to a comprehensive collection of various descriptive properties of *P. polycephalum* networks. Let us first discuss the results individually and then comment on their combined usefulness.

Paths can be regarded as the atomic building blocks of *P. polycephalum* networks since they correspond to veins carrying protoplasm. We find that our data does not allow to unambiguously determine the underlying theoretical distribution of path lengths and average path widths, however, fitting gamma distributions slightly outperforms other distribution choices. Relying on gamma fits, we explore the time development of the distributions and find behavior that is consistent with network coarsening for a large number of data series. We determine a range of parameters that captures the distributions of path lengths and average path widths one may expect to observe in *P. polycephalum* networks. In particular we find a linear relation between fit parameters for the average path width. Within the range of observed values, this means that the variation in the path width distribution is small, i.e. the mode of the distribution is confined to a small interval. This is not the case for path lengths, which may display larger variations. With regards to network coarsening, it follows that the tendency to replace short veins with long ones is more pronounced than the tendency to replace thin veins with thick ones.

In order to obtain more information about the structure of *P. polycephalum* networks we introduce the study of its faces which correspond to cycles formed by veins. Here we determine the distribution of physical aspects of faces such as the number of paths in a face (face degree) as well as its area and circumference. Unfortunately,

[1] The point set for the tessellation is Poisson distributed in the plane.

we are not able to determine the underlying distribution with certainty, but find that gamma distributions capture face degree, area and circumference well. Similarly to the path properties we report on the range of parameters that quantify the variance in the observed distribution and explore their time dependence. We find that the majority of faces are of degree 5, while the faces of degree 6 account for the majority of the area covered by *P. polycephalum*. We quantify the similarity in shape between faces and regular polygons and find that the most frequently occurring faces tend to be rather regular.

To study the degree of spatial homogeneity of *P. polycephalum* graphs we cut each network of a given time-series in 100 equidistant horizontal and vertical cuts. For each cut set we compute the following values: Its size and the sum of the lengths, respectively widths, of the edges in the set. Studying the change in cut values as one moves horizontally or vertically allows us to shed light on the degree of homogeneity with respect to these directions. We find that there is a pronounced difference between the two different cut directions. Vertical cuts seem largely stable in value as one moves through the cuts, while horizontal cuts show a pronounced tendency to change. Since vertical cuts are perpendicular to the growth direction of the slime mold, balancing the cut values is beneficial for pushing flow towards the growing front in order to advance it further. At the same time balancing cuts perpendicular to the growth direction seems to be of minor importance for *P. polycephalum*. This indicates that foraging takes precedence over improving circulation. While we establish a first quantitative connection between transport capabilities and growth direction, a more detailed investigation of such a connection suggests interesting future research.

Finally, we look towards a topological property of *P. polycephalum* networks and investigate its resilience to random damage using concepts from percolation theory. We do observe finite size effects and perform a finite size scaling analysis. In the thermodynamic limit of infinite system size, we obtain $p_c = 0.7118 \pm 0.0544$ for random node percolation and $p_c = 0.6584 \pm 0.0217$ for random edge percolation. The results imply that a fraction of $1 - p_c$ of nodes respectively edges can be removed at random before the network of infinite size loses its connectivity. For finite size networks the obtained thresholds yield a reliable approximation of how much damage can be sustained before circulation is critically compromised. It is interesting to note that the obtained critical thresholds are very close to those known for 2D Voronoi tessellations [24]. Thus a closer investigation of the similarities might be rewarding.

While the obtained results are individually interesting and novel, they are selected such that their combination yields a comprehensive characterization of *P. polycephalum* networks and useful base of knowledge to work with. In particular it becomes possible to answer the question whether a given weighted planar network is similar to the networks contained in the KIST Europe data set. Computing the distributions of various path and face properties in conjunction with cut and percolation properties yields a "fingerprint" which can be compared with the data presented in this manuscript. Thus, the combined information is suitable to guide and evaluate research geared towards modeling aspects of *P. polycephalum*. Models

that make predictions in terms of the observables studied here may be put to the test effectively. Furthermore, detailed knowledge of individual graph properties and what ranges of values to expect from them may be useful on their own.

To help facilitate all of this, we make all our results accessible online via the SMGR. This includes the raw numerical data as well as individual and cumulative results of all computed observables for all available data series, down to the level of single *P. polycephalum* graphs. In addition to that, we include tools which allow the interested reader to repeat our measurements as well as to conduct experiments that go beyond the ones presented here.

4.5. Acknowledgments

We are grateful towards Prof. M. Grube and Dr. C. Westendorf for their hospitality, expert advice on slime molds and their encouragement in the early stages of this work. We thank to Prof. H.-G. Döbereiner for valuable feedback and suggestions. We acknowledge Prof. A. Manz and Prof. L. Abelmann and their group at the KIST Europe for stimulating discussions. Lastly, we are grateful towards Dr. A. Neumann, Dr. M. Függer and Prof. A. Maas for their encouragement and valuable technical suggestions.

5 | Modeling Flows

In this chapter we propose to model the oscillatory flows observed in the vein networks formed by the slime mold *Physarum polycephalum* as currents in an electric circuit. Our approach is inspired by the modeling of the human cardiovascular system and relies on electronic elements such as resistors, capacitors and current controlled voltage sources to capture the properties of interacting flow-carrying veins of the organism. In particular we map the effects of oscillatory peristaltic pumping to the behavior of a simple interacting electronic circuit, the so-called *Physarum* element. Connecting these elements according to a given graph, yields an electric *Physarum* network. The currents induced by certain networks exhibit complex emergent flow patterns reminiscent of the flows observed in live *P. polycephalum*. Our model is simple, fully distributed and robust to changes in the topology of the *Physarum* network. Thus it constitutes a promising basis for the future development of distributed natural computing algorithms.

This chapter documents joint work with Prof. M. Grube, Dr. M. Függer and Prof. K. Mehlhorn. The first provided many valuable insights regarding the biology of *P. polycephalum*, while the latter two were the driving force behind the analytic work presented in this chapter.

5.1. Introduction

Slime molds are interesting and complex organisms providing a rich substrate for interdisciplinary research. In the recent past, one member of the family, called *Physarum polycephalum* [81], has been discovered as a suitable medium for natural computing. Its outstanding morphological dynamics have been repeatedly connected to optimization processes capable of solving complex problems. Prominent examples include computing shortest paths [36], [124], [174], designing transport networks [119], [176], or controlling robots [177], amongst others.

One striking feature of *P. polycephalum* is its ability to form and maintain a massive cell body in the form of a dynamic complex network of veins. These networks are highly adaptive and may change drastically in response to changing environmental conditions. This extraordinary functional plasticity allows *P. polycephalum* to navigate its environment successfully in search for food. Efforts to improve our understanding of formation, structure and function of these networks are manifold and ongoing [5], [19], [22], [108], [176].

Similar to the way a typical mammalian vascular network ensures the circulation of blood, the veins of the plasmodium allow protoplasmic fluid to freely flow [97].

The resulting fluid circulation is vital to the organism as a whole since it ensures that nutrients, nuclei and other relevant factors are equally available across an entire individual. Note this circulation is maintained naturally despite the ever changing and adapting underlying network of veins.

The protoplasmic flow itself is driven by quasi-periodic cross-sectional contractions that occur with a period of approximately 100 s [163], [183]. They are generated by a mesh consisting of actin and myosin fibers forming the walls of the tubular veins. Contractions cause a peristaltic pumping effect leading to net fluid transport, the so-called shuttle streaming [96]. Resulting peristaltic pressure waves can be observed across the entire network inducing complex flow patterns [125]. These include periodic flow arrests and reversals in single veins on a time scale of (50 ± 5) s.

It is believed that efficient and robust transport of protoplasm across the entire organism emerges from the interplay of dynamic network topology, quasi-periodic local contractions and complex flow patterns [5], [172]. Unfortunately, the details of such a process remain in the dark.

From a biological point of view it can be argued that evolution has optimized the behavior of *P. polycephalum* at least to the point where fluid transport is efficient and robust enough to survive. Another evolutionary advantage can be seen in the self-organizing character of the slime mold since the evolved neural circuitry, vital to the survival of more complex organism, is simply not necessary to *P. polycephalum*. Indeed, *P. polycephalum* lacks any type central control capable of coordinating its apparently coordinated behavior. It is intriguing to ask how this organism manages to organize efficient and robust fluid transport in a fully decentralized manner.

From a computing point of view such properties are non-trivial and highly desirable. What *P. polycephalum* seems to produce and maintain naturally is an (approximate) solution to the problem of distributing resources in a dynamically changing planar graph. This is an interesting and complex transport problem with various conceivable practical applications, particularly in the domain of operations research. For an overview of similar transport problems and their computational complexity see [12], [57], [79].

As a result, modeling the behavior of the slime mold with the goal of deriving algorithm for transport problems is an interesting proposition. Provided such an algorithm can be obtained, it should have the following properties:

- The algorithm maintains a dynamic circulation of flow including flow reversals mimicking the flows observed in live *P. polycephalum*. Since resources are transported with the flow, they go wherever the flow reaches.

- The algorithm is robust against changes in topology. Neither natural changes of network topology nor accidental disconnection of veins renders *P. polycephalum* in a state from which it cannot recover. The algorithm should share this quality.

- The algorithm is distributed and requires no central control. As a result, complex global coordination of any sort must emerge from local interactions.

- The algorithm has a degree of efficiency. Based on the assumption that a certain degree of efficiency is necessary for *P. polycephalum* to survive, one may hope that models and algorithms mimicking the organism, inherit this efficiency at least to some extent.

In this manuscript we present a model of *P. polycephalum* intended to support the development of distributed natural computing algorithms. It is inspired by the modeling of the human cardiovascular system and yields emergent flow patterns similar to the ones displayed by the organism. This includes flow reversals and anti-phase oscillations. Our model aspires to the properties listed above and aims at replicating the way *P. polycephalum* distributes resources across its vein network. In particular it maps the oscillatory behavior of *P. polycephalum* to simple interacting electronic circuits designed to mimic the effect of peristaltic pumping. The model is fully distributed and exhibits emergent dynamic flow patterns. Due to its relative simplicity it remains amenable to analytic treatment. *In silico* investigations are presented in support of the validity of our approach which demonstrate the behavior of our model.

The presented model may serve as a basis for future investigations in the context of natural computing. Two primary directions discussed in Section 1.2 are possible: a) computing inspired by nature and b) the synthesis of nature by means of computing. For the former, one may utilize the model in an effort to develop novel distributed natural computing algorithms inspired by *P. polycephalum*. The latter may aim towards refining the model with the goal of improving both its accuracy and predictive power. Both approaches present distinct challenges and both approaches deserve future exploration.

5.2. Overview of Modeling Approaches

Before we introduce our approach of modeling the oscillations and flow dynamics observed in *P. polycephalum*, let us survey three classes of existing approaches that provided inspiration.

The first class consists of interpreting the plasmodium of *P. polycephalum* as a living ensemble of interacting oscillators. These can take different forms, ranging from intricate chemical oscillators [158], to various mechanical ones [166], [168], [173].

As an example, we mention interpreting the plasmodium of *P. polycephalum* as a living system of delay-coupled mechanical oscillators [166], see Section 1.3.1 for a detailed description. Here it suffices to say, that a micro-fabricated structure was prepared, consisting of two identical circular reservoirs connected by a channel of variable width and length, see Figure 1.6b. In the experiment, the two reservoirs and the channel are populated with plasmodium. The reservoirs act as two distinct *P. polycephalum* oscillators while the channel ensures a controllable coupling between the two with its width determining the coupling strength, while its length controls

the time delay. In the experiment *P. polycephalum* shows rich self-synchronizing oscillation patterns between the distinct reservoirs with both in-phase and anti-phase entrainment. A strong dependence on the geometry of the channel was observed. These results were found to agree with the theoretical predictions of an equivalent model consisting of a system of two fully distributed delay-coupled oscillators representing the two reservoirs and the channel. The findings confirmed that the behavior of delay-coupled oscillators yields a good description of the oscillation patterns exhibited by *P. polycephalum* in the experiment.

One disadvantage of oscillator approaches is the difficulty of obtaining the actual flow patterns from the knowledge of the oscillation phases. Furthermore, systems with a non-trivial number of interacting oscillators quickly become intractable analytically.

The second class follows a different approach and focuses on fluid mechanics including accurate modeling of peristaltic pumping [5], [172]. Here the central aim is to connect the contractions of vein segments to the resulting hydrodynamic fluid flow.

Recent work along those lines showed that fluid flow and transport through a network of vein segments is optimal if the wavelength of the peristaltic wave is of the order of the size of the network [5]. The flow patterns predicted by a hydrodynamic model, including the effects of peristalsis, showed flow reversals and arrests which were found to be in good agreement with experimental observations.

The subtle downside of this class of models is the fact that they are not fully decentralized. Solving them requires one to fix certain initial conditions for the hydrodynamic equations. In other words the initial states for a selected set of points need to be chosen and the entire system evolution is made to depend the choices made. Note how these choices introduces a central dependence on a small set of distinct points which contrasts truly decentralized systems such as live *P. polycephalum*. For them, no such distinct points exist and the system state is determined by a self-organizing process which is often insensitive to starting conditions.

The third class assumes that the hydrodynamic analogy holds true for veins and networks formed by *P. polycephalum*. Based on this assumption, notions from the theory of electric circuits are used to model *P. polycephalum* as an electrical network rather than a hydraulic one. Table 5.1 translates between both descriptions. We remark that the idea of modeling single viscoelastic tubes and complex networks formed by them as electrical circuits is not novel. It has been introduced and subsequently refined with great success in the context of modeling the human cardiovascular system[63], [76], [104], [138], [161]. Given the apparent similarities between the networks formed by *P. polycephalum* and human vascular networks, it is natural to explore this approach for the modeling of slime molds such as *P. polycephalum*.

Shifting the description of the dynamics of *P. polycephalum* from the hydraulic to the electric world offers several advantages: First, one may dismiss the challenging hydrodynamic equations in favor of simpler electrical ones. Second, electrical circuits are subject to Kirchhoff's circuit laws which can be used to simplify their treatment

Hydrodynamic System	Electrical analogue
Fluid	Charge
Fluid flow	Charge flow, i.e. current
Pressure	Potential
Pressure difference	Potential difference, i.e. voltage
Viscosity	Resistance
Distensibility	Capacitance
Pump	Voltage source
Inert mass	Inductance

Table 5.1.: Illustrating the analogy between hydraulic and electric systems. Adapted and extended based on [138].

analytically. Third, the approach is very flexible and allows one to study different properties of *P. polycephalum* within the same framework.

Extremely recent examples include the study of the mechanisms of information processing in *P. polycephalum* [133].[1] Relevant electronic models have also been formulated to describe various abilities of *P. polycephalum* which were demonstrated in earlier experiments. Examples include the prediction of periodic changes in its environment [140], and the solving of mazes [130].

The difficulty with this class of models lies with the fact that the electronic elements they contain (resistors, capacitors, inductors, current controlled voltage sources) require a number of parameters to be set. If relevant statements with actual predictive power are to be obtained by using such a model, extreme care must be taken when deciding on the parameters. Unfortunately, this task remains difficult.

In the following we present an electric model of *P. polycephalum* inspired by earlier attempts of modeling the human cardiovascular network [138], [161]. Our model focuses on capturing the self-organized oscillatory dynamics of *P. polycephalum* with the goal of obtaining emergent flow patterns similar to those observed in the slime mold. In particular we mimic the effect of peristalsis through the introduction of current controlled voltage sources. Our approach embodies the most desirable features of previous models, namely a direct representation of fluid flows governed by a fully decentralized, tractable dynamics. At this point, we operate our model with a set of parameters which is suitable for exploring its behavior. Determining parameters that allow reliable physical predictions is not our focus at this point.

We begin by defining a continuous time model from which we subsequently derive a discrete version suitable for numerical treatment.

[1] At the time of writing, only the abstract of this work was available.

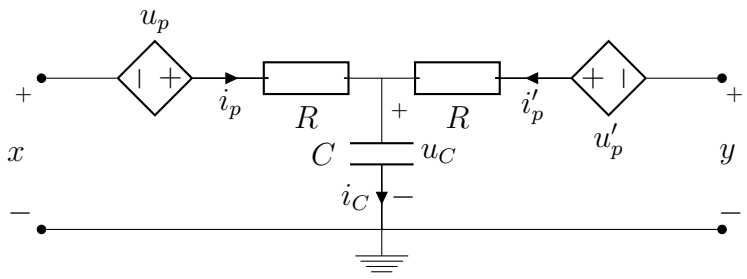


Figure 5.1.: Electrical model of a *P. polycephalum* vein segment. The two resistors R and the capacitor C form the three-element Windkessel model [135]. The two current controlled voltage sources u_p and u'_p augment the model to include the effects of peristaltic pumping.

5.3. Continuous Model

We model a single vein of *P. polycephalum*, respectively a sufficiently short segment of a vein, by an electrical circuit. Here we rely on the hydraulic analogy where current represents protoplasmic flow and voltage refers to differences in fluid pressure. The basis of our model is formed by the three-element Windkessel model [63], [75], comprised of two resistors and a capacitor. The electrical resistance is analogous to hydrodynamic resistance, a result of viscous dissipation inside the tubular veins of *P. polycephalum*. Capacitors accommodate the fact that the veins of *P. polycephalum* are not rigid but exhibit a degree of volume compliance. Their addition also prevents changes in current to propagate through the entire network instantaneously. While this is the case in purely resistive networks it certainly seems unphysical for flows in *P. polycephalum*. At this stage, we ignore the inertia of the protoplasmic fluid which in the Windkessel model is captured by inductors. For a full derivation of the Windkessel model we refer the reader to [135].

We extend the Windkessel model by adding active components, namely current controlled voltage sources. These voltage sources create potential differences leading to a current. In this way, they account for the fact that veins of *P. polycephalum* act as peristaltic pumps capable of creating pressure differences which induce fluid flow. Figure 5.1 depicts the resulting circuit. Note that the circuit is symmetric, ensuring that its behavior does not depend on the sign of the current flowing through it.

Both current controlled voltage sources are identical and we specify their behavior by:

$$u_p^*(t) = \max(\min(\beta \cdot i_p(t), \hat{U}), -\hat{U}) \tag{5.1}$$

$$\frac{du_p(t)}{dt} = \alpha(u_p^*(t) - u_p(t))\,, \tag{5.2}$$

where $\alpha, \beta > 0$ and $\hat{U} > 0$ are constants. To ensure that the currents induced by the current controlled voltage sources are not completely damped by resistances, we

further assume that

$$\beta > R\,. \tag{5.3}$$

Note that Equation (5.1) specifies the limit behavior of the current controlled voltage source. The voltage $u_p^*(t)$ is directly proportional to current $i_p(t)$, but cut off symmetrically at $\hat{U}$ and $-\hat{U}$. Thus the output of the voltage source is capped yielding both negative as well as positive voltages depending on the sign of the current. The constant β controls how sensitive the current controlled voltage source reacts to changes in current, see Figure 5.2a.

These choices were inspired by empirical observations linking the diameter of veins of *P. polycephalum* to the fluid flow through them. If throughput is high/low, the vein diameter is expected to grow/shrink [124]. As a result, effectiveness of the peristaltic pumping is altered since the generated flow is directly proportional to the diameter of the vein. To capture this positive feedback our model is setup such that current is proportional to the voltage of the current controlled voltage source which in turn affects the current. Note that an increase/decrease in vein diameter also decreases/increases the hydrodynamic resistance of the vein. Thus, the electrical resistors R in our model should have a dependency on the current. At this stage of modeling, however, we choose to ignore this dependency in favor of a simpler model. Taking into account the interplay between current and resistance would yield a more accurate model and thus is a strong contender for future augmentations.

Equation (5.2) accounts for the inertia of the current controlled voltage source through a first order differential equation. To model the fact that pumping in *P. polycephalum* does not adapt arbitrarily fast to changes in fluid flow, we chose to delay the reaction of the current controlled voltage source through the use of a low-pass filter, see Figure 5.2b.

5.3.1. Putting the Model on a Graph

In the previous section we have defined the basic electronic building block of our model representing a vein segment of *P. polycephalum*. In the following we shall refer to it as *Physarum* element. Next we define how *Physarum* elements are combined according to a graph G in order to form electronic networks representing arbitrary vein network of *P. polycephalum*.

A *Physarum network* is specified by a directed graph $G(V, E)$ where each edge $e = (i, j) \in E$ represents a *Physarum* element with $i, j \in V$. With respect to the *Physarum* element shown in Figure 5.1, x corresponds to node i and y to node j. We connect all segments accordingly when they share a common node. Note that solely for the purpose of analysis do we assume that edges are directed. By construction, two *Physarum* elements which differ only in orientation will behave the same.

Edges are labeled by a choice for parameters $R, C, \alpha, \beta, \hat{U}$ specifying the electric properties of an *Physarum* element. Note that at this exploratory stage of modeling, any sensible choice of constants allows us to study the model. In particular we may postpone a detailed study of the physical meaning of these constants for now.

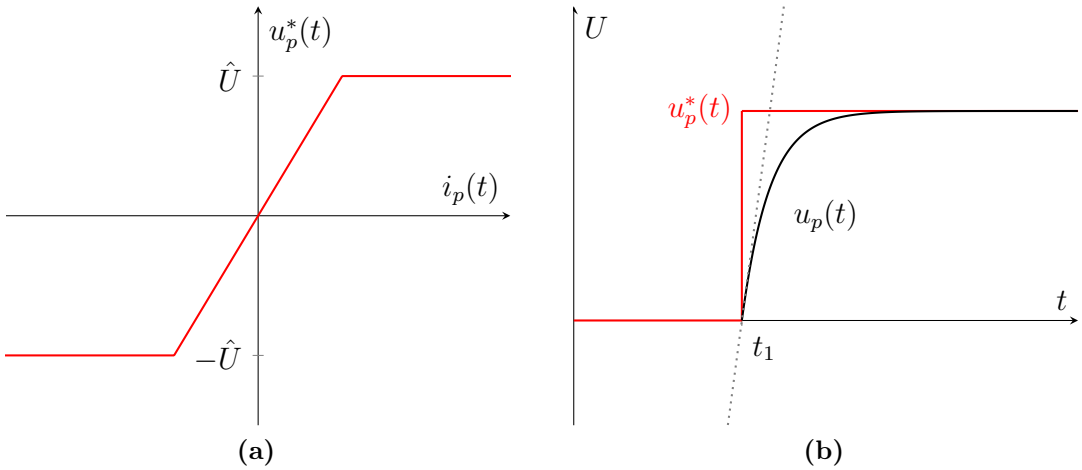


Figure 5.2.: (a) Current controlled voltage $u_p^*(t)$ may not increase/decrease indefinitely but is capped at a constant value. The constant β defines the slope of the red line passing through the origin. (b) A voltage $u_p(t)$ is gained after applying a first order low-pass filter to $u_p^*(t)$. Note that the slope at time t_1, indicated by a dashed gray line, is proportional to α.

Furthermore, we add initial values for the capacitor voltage $u_C(t = 0)$ and the initial current controlled voltage source voltages $u_p(t = 0)$. Our actual choices do not matter since the model is going to adjust these values dynamically in a self-organized manner.

An *execution* of a *Physarum* network is a function that maps each edge in G to a signal $t \mapsto u_C(t)$, i.e. tracking the time development of the capacitor voltages in the network defined by G.

Our exposition follows the notational conventions to refer to the voltage present at a node $v \in G$ by u_v, and adding an index $e \in E$ for variables of the corresponding *Physarum* element. For example, $u_{C,e}$ is the capacitor voltage for *Physarum* element e.

We say a *Physarum* network G *converges* if for its execution we have that $u_{C,e}$ converges for all $e \in G$. We say that a *Physarum* network *dies* if it converges, and for all its edges $e \in G$, $i_{p,e} = 0$.

Figure 5.3 depicts an exemplary *Physarum* network featuring a node $v \in G$ with three incident *Physarum* elements denoted by $E(v)$. Observe, that we may apply Kirchhoff's voltage law to the junction node v, as well as to the paths over each voltage source, resistor, and capacitance of each edge. Furthermore, from Kirchhoff's current law at the junction v, we have that in-flows must be equal to out-flows at v, i.e. charge is conserved.

For arbitrary degree, the voltage at node v at time t is denoted by $u_v(t)$. It is

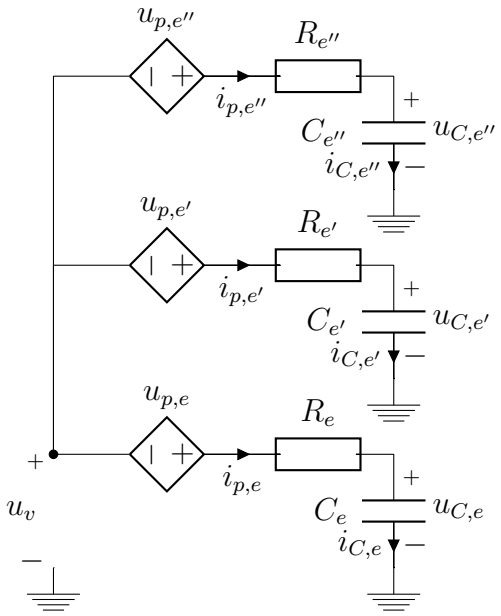


Figure 5.3.: Voltage u_v at a node $v \in G$ of degree 3 with incident edges e, e', and e''. The r.h.s. parts of the *Physarum* elements are not shown because they do not affect u_v.

determined by

$$\forall e \in E(v) : (u_{C,e}(t) - u_{p,e}(t)) + i_{p,e}(t)R_e = u_v(t) \tag{5.4}$$

$$\sum_{e \in E(v)} i_{p,e}(t) = 0\,. \tag{5.5}$$

In the remainder of this work we assume that all *Physarum* elements are identical, i.e. have identical sets of constants. This is a strong simplification, but we believe it is necessary to keep the model simple enough for analytic treatment. Relaxing it is non-trivial and subject of future improvements.

Thus,

$$u_v(t) = \frac{1}{|E(v)|} \sum_{e \in E(v)} (u_{C,e}(t) - u_{p,e}(t))\,. \tag{5.6}$$

5.4. Basic Properties of the Continuous Model

Given the definition of our model, we explore it analytically by establishing some basic properties and investigate *Physarum* networks with simple topologies such as rings and trees.

Let the sum of the charge stored by all capacitors in a *Physarum* network be Q. We show that Q is a conserved quantity.

Lemma 1. *For all Physarum networks, there exists a $Q \in \mathbb{R}$ such that,*

$$\forall t \geq 0 : \sum_{e \in E} C_e u_{C,e}(t) = Q \,. \tag{5.7}$$

Proof. Since all capacitors are connected with one end to ground, we may apply Kirchhoff's current law to the ground. Thus

$$\sum_{e \in E} i_{C,e}(t) = 0 \,,$$

holds. Furthermore the relation between capacitor current I and voltage U is given by $I = C\frac{dU}{dt}$. Thus we get

$$\sum_{e \in E} C_e \frac{du_{C,e}(t)}{dt} = 0 \,,$$

and hence

$$\frac{d}{dt} \sum_{e \in E} C_e u_{C,e}(t) = \frac{d}{dt} Q = 0 \,.$$

□

Next let us study *Physarum* networks with the topology of a cycle. Here *Physarum* elements are chained one after another with the last edge connecting back to the start. The analysis implies that these networks show very interesting behavior.

Lemma 2. *Let G be a Physarum network that is a cycle. If G converges, then it converges to either of three steady states:*

1. *it dies, i.e., for all edges e in G, $i_{p,e} = 0$,*
2. *it does not die, and for all edges e in G, $i_{p,e} = \frac{|E|\hat{U}}{\sum_{e \in E} R_e}$,*
3. *it does not die, and for all edges e in G, $i_{p,e} = -\frac{|E|\hat{U}}{\sum_{e \in E} R_e}$.*

Proof. Let e be an arbitrary *Physarum* element in G. Since we assume that G has converged the capacitor voltage $u_{C,e}(t)$ is constant. Thus we have

$$0 = C_e \frac{du_{C,e}(t)}{dt} = i_{C,e}(t) \,, \tag{5.8}$$

which implies that no current flows into the capacitors. Since charge is conserved and G is a cycle, all current must flow through the resistors instead. Thus, for all edges $e \in G$ we have

$$i_{p,e}(t) = -i'_{p,e}(t) = i^* \,, \tag{5.9}$$

illustrating that all currents have identical strength. In a cycle non-zero current thus flows either clock-wise or counter-clockwise.

Next we substitute i^* in Equation (5.1) and obtain

$$u_{p,e}(t) = u^*_{p,e}(t) = \max(\min(\beta i^*, \hat{U}), -\hat{U}) =: u_p\,, \tag{5.10}$$

indicating that the current controlled voltages are also identical and constant for all $e \in G$. The circuit of the *Physarum* element also implies that $u'_{p,e}(t) = -u_p = u'_p$.

We next apply Kirchhoff's voltage law along the path traversing the whole cycle and obtain

$$0 = \sum_{e\in E}(u_p - u'_p - 2i^*R_e) = 2\sum_{e\in E}(u_p - i^*R_e)\,. \tag{5.11}$$

Next, we distinguish between three cases for u_p, according to Equation (5.10): *Case (1):* $u_p = \beta i^*$, *Case (2):* $u_p = \hat{U}$, and *Case (3):* $u_p = -\hat{U}$.

Case (1): Substituting $u_p = \beta i^*$ into Equation (5.11) yields

$$0 = 2i^*\sum_{e\in E}(\beta - R_e)\,. \tag{5.12}$$

Since $\beta \neq R_e$ by assumption (5.3), the above equation is only fulfilled if $i^* = 0$, which corresponds to case 1 in the lemma.

Case (2): From $u_p = \hat{U}$ and Equation (5.11) we have

$$i^* = \frac{|E|\hat{U}}{\sum_{e\in E} R_e}\,. \tag{5.13}$$

Since, case (2) only applies when $\beta i^* > \hat{U}$, we further obtain

$$\frac{|E|\beta}{\sum_{e\in E} R_e} > 1\,, \tag{5.14}$$

in correspondence with case 2 in the lemma. Note, that the previous statement is implied by assumption Equation (5.3).

Case (3): The proof for case $u_p = -\hat{U}$, corresponding to case 3 in the lemma, is analogous to case (2).

□

Lemma 3. *Let G be a Physarum network that is a tree. If G converges, it dies.*

Proof. Assume that G converges. By definition $u_{C,e}(t)$ is constant for all edges e in G. We will show by induction on the distance of an edge e to the leaf edges, that its currents $i_{p,e}$ and $i'_{p,e}$ are 0.

To begin the induction, consider a leaf edge e. Without loss of generality we may assume that the e is oriented in a way such that node x in Figure 5.1 is of degree 1. From Kirchhoff's current law we then have that $i_{p,e} = 0$. From the fact that G is converged, we know that $i_{C,e} = 0$. By Kirchhoff's current law it follows that $i'_{C,e} = 0$.

As induction hypothesis, assume that all edges e at distance $\ell \geq 0$ have $i_{p,e} = i'_{p,e} = 0$. Consider an edge e' at distance $\ell + 1$. Assuming, again without loss of generality, that e' is oriented in a way such that its node x is incident to an edge at distance ℓ. Observe that by the induction hypothesis, $i_{p,e'} = 0$. By arguments analogous to the start of the induction we find that $i'_{C,e'} = 0$ and the inductive step follows. □

5.5. Discrete Model

In the following we propose a discrete round model, with rounds $r \in \mathbb{N}$. Again, a *Physarum* network is specified by a graph $G(V, E)$ whose edges are labeled as in the continuous time model. For simplicity we assume that all parameters R, C, α, β and $\hat{U}$ are the same for all edges.

The discrete model is derived from the continuous model using forward Euler integration. Let $\varepsilon > 0$ denote the step-size, i.e. the time between two discrete rounds. We refer the reader to Appendix D for a basic implementation of the crucial updates defined in this section.

Intially at $r = 0$: For each edge e in G do:

1. Set $u_{C,e}(r = 0)$ and $u_{p,e}(r = 0)$ as specified by the edge label.

At each round $r \geq 1$:

1. Update node voltages (Code Listing D.3): in the continuous model we determine node voltages according to Equation (5.6). We follow the same approach in the discrete model. Let v be a node in G. Without loss of generality we assume that all edges incident to v are oriented as in Figure 5.1 with v connected to x. Then we have

$$u_v(r-1) = \frac{1}{|E(v)|} \sum_{e \in E(v)} \left(u_{C,e}(r-1) - u_{p,e}(r-1)\right). \tag{5.15}$$

2. Update capacitor voltages (Code Listing D.4): in the continuous model, for each edge $e = (v, w)$ in G, we have

$$\begin{aligned} C\frac{du_{C,e}(t)}{dt} &= i_{C,e}(t) = i_{p,e}(t) + i'_{p,e}(t) \\ &= \frac{u_v(t) + u_w(t) + u_{p,e}(t) + u'_{p,e}(t) - 2u_{C,e}(t)}{R}. \end{aligned} \tag{5.16}$$

By first order discretization, we thus obtain

$$u_{C,e}(r) = u_{C,e}(r-1) + \\ + \frac{u_v(r-1) + u_w(r-1) + u_{p,e}(r-1) + u'_{p,e}(r-1) - 2u_{C,e}(r-1)}{RC\varepsilon^{-1}} . \quad (5.17)$$

3. Update current controlled voltages (Code Listing D.5): for each edge e in G:

$$i_{p,e}(r-1) = \frac{u_v(r-1) + u_{p,e}(r-1) - u_{C,e}(r-1)}{R} , \quad (5.18)$$

$$u^*_{p,e}(r) = \max(\min(\beta i_{p,e}(r-1), \hat{U}), -\hat{U}) , \quad (5.19)$$

$$u_{p,e}(r) = u^*_{p,e}(r)(1 - e^{-\alpha\varepsilon}) + u_{p,e}(r-1)e^{-\alpha\varepsilon} . \quad (5.20)$$

Analogously, for the right voltage source, set

$$i'_{p,e}(r-1) = \frac{u_w(r-1) + u_{p,e}(r-1) - u_{C,e}(r-1)}{R} , \quad (5.21)$$

$$u'^*_{p,e}(r) = \max(\min(\beta i'_{p,e}(r-1), \hat{U}), -\hat{U}) , \quad (5.22)$$

$$u'_{p,e}(r) = u'^*_{p,e}(r)(1 - e^{-\alpha\varepsilon}) + u'_{p,e}(r-1)e^{-\alpha\varepsilon} . \quad (5.23)$$

5.6. Basic Properties of the Discrete Model

In the following we establish the discretized version of Lemma 1. This proof is important since it guarantees that the discretization we introduced in the previous section to solve the model does not break conservation of charge numerically. In particular, we know that the sum of the charges stored in the capacitors is still conserved.

Lemma 4. *For all Physarum networks, there exists a $Q \in \mathbb{R}$ such that*

$$\forall r \geq 0 : \sum_{e \in E} C_e u_{C,e}(r) = Q . \quad (5.24)$$

Proof. Let $r > 0$. By rewriting Equation (5.17) we have

$$u_{C,e}(r) = \left(1 - \frac{2\epsilon}{RC}\right) u_{C,e}(r-1) \quad (5.25)$$
$$+ \epsilon \frac{u_v(r-1) + u_w(r-1)}{RC} + \epsilon \frac{u_{p,e}(r-1) + u'p, e(r-1)}{RC} .$$

We may sum over all edges $e \in E$ on both sides to obtain

$$\sum_{e\in E} u_{C,e}(r) = \left(1 - \frac{2\epsilon}{RC}\right) \sum_{e\in E} u_{C,e}(r-1) \tag{5.26}$$
$$+ \frac{\epsilon}{RC} \left(\sum_{e\in E} u_{v(e)}(r-1) + \sum_{e\in E} u_{w(e)}(r-1) \right)$$
$$+ \frac{\epsilon}{RC} \sum_{e\in E} \left(u_{p,e}(r-1) + u'_{p,e}(r-1) \right) .$$

Here $v(e)$ denotes the tail of edge e and $w(e)$ the head. We apply this distinction when we sum over all edges $e \in E$ in Equation (5.15), to obtain

$$\sum_{e\in E} u_{v(e)}(r-1) = \sum_{e\in E} \frac{1}{|E(v(e))|} \sum_{e'\in E(v(e))} \left(u_{C,e'}(r-1) - u_{p,e'}(r-1) \right), \tag{5.27}$$

$$\sum_{e\in E} u_{w(e)}(r-1) = \sum_{e\in E} \frac{1}{|E(w(e))|} \sum_{e'\in E(w(e))} \left(u_{C,e'}(r-1) - u'_{p,e'}(r-1) \right). \tag{5.28}$$

Where we assume that the edges $E(v(e))$ are out-edges while the $E(w(e))$ are in-edges. This assumption is convenient for the purpose of analysis. however, it has no further implications since *Physarum* elements are symmetric.

Next we substitute (5.27) and (5.28) into Equation (5.26) and get

$$\sum_{e\in E} u_{C,e}(r) = \left(1 - \frac{2\epsilon}{RC}\right) \sum_{e\in E} u_{C,e}(r-1) \tag{5.29}$$
$$+ \frac{\epsilon}{RC} \left(\sum_{e\in E} \frac{1}{|E(v(e))|} \sum_{e'\in E(v(e))} \left(u_{C,e'}(r-1) - u_{p,e'}(r-1) \right) \right)$$
$$+ \frac{\epsilon}{RC} \left(\sum_{e\in E} \frac{1}{|E(w(e))|} \sum_{e'\in E(w(e))} \left(u_{C,e'}(r-1) - u'_{p,e'}(r-1) \right) \right)$$
$$+ \frac{\epsilon}{RC} \sum_{e\in E} \left(u_{p,e}(r-1) + u'_{p,e}(r-1) \right) .$$

Seeking to simplify the double summations appearing in Equation (5.29), we first point out that for the sums regarding the $u_{C,e'}(r-1)$ we have

$$\sum_{e\in E} \frac{1}{|E(v(e))|} \sum_{e'\in E(v(e))} u_{C,e'}(r-1) = \sum_{e'\in E} u_{C,e'}(r-1) \sum_{e\in E; v(e)=v(e')} \frac{1}{|E(v(e))|} \tag{5.30}$$
$$= \sum_{e'\in E} u_{C,e'}(r-1) = \sum_{e\in E} u_{C,e}(r-1) ,$$

where the summations have been interchanged to get rid of one of the sums. Similarly we examine the sums regarding the $u_{p,e'}(r-1)$ and find

$$\sum_{e\in E}\frac{1}{|E(v(e))|}\sum_{e'\in E(v(e))}u_{p,e'}(r-1)=\sum_{e'\in E}u_{p,e'}(r-1)=\sum_{e\in E}u_{p,e}(r-1)\,, \tag{5.31}$$

and analogously for the sums regarding the $u'_{p,e'}(r-1)$ it reads

$$\sum_{e\in E}\frac{1}{|E(w(e))|}\sum_{e'\in E(w(e))}u'_{p,e'}(r-1)=\sum_{e'\in E}u'_{p,e'}(r-1)=\sum_{e\in E}u'_{p,e'}(r-1)\,. \tag{5.32}$$

Finally, we substitute (5.30), (5.31) and (5.32) into Equation (5.29) to arrive at

$$\begin{aligned}\sum_{e\in E}u_{C,e}(r)&=\left(1-\frac{2\epsilon}{RC}\right)\sum_{e\in E}u_{C,e}(r-1) \\ &+\frac{\epsilon}{RC}\left(\sum_{e\in E}u_{C,e}(r-1)-\sum_{e\in E}u_{p,e}(r-1)+\sum_{e\in E}u_{C,e}(r-1)-\sum_{e\in E}u'_{p,e'}(r-1)\right) \\ &+\frac{\epsilon}{RC}\sum_{e\in E}\left(u_{p,e}(r-1)+u'_{p,e}(r-1)\right) \\ &=\sum_{e\in E}u_{C,e}(r-1)\,.\end{aligned} \tag{5.33}$$

Thus we have

$$\sum_{e}u_{C,e}(r)=\sum_{e}u_{C,e}(r-1)=\sum_{e}u_{C,e}(r-2)=\ldots=\sum_{e}u_{C,e}(0)=const.\,. \tag{5.34}$$

From the fact that that all C_e are constant and equal the lemma follows immediately. □

5.7. Preliminary Simulation Results

In this section we present the results of preliminary *in silico* explorations of the discretized model presented in Section 5.5. We investigate several different basic circuits of *Physarum* elements and discuss our findings. The circuits are based on small graphs in the hope that one may still reason about them analytically. This is a preparatory step towards a better understanding of the behavior of our model. Ultimately, the hope is to gain insights useful for deriving an algorithm based on the model.

The following facts suggest, that our implementation is correct: First, we find that during the execution of our model Kirchoff's laws are obeyed. Second, the

Property	value
$u_p = u'_p$	0
$i_p = i'_p$	0
u_C	rand. uniform real $u_C \in [-0.5, 0.5)$
C	10
R	1
$\hat{U}$	1
α	$(RC)^{-1}$
β	$3R$
$x = y$	1

Table 5.2.: Initial settings for discrete simulations at $r = 0$.

implementation maintains the distinct invariants we have established analytically as given by Lemma 1. Finally, for converged cycles of *Physarum* elements we observed precisely the behavior predicted by Lemma 2. This constitutes a nice example of how numerical simulation and analytical work are in an symbiotic relationship. Insights gained in one frequently suggest directions of exploration to the other.

All the topologies we present are drawn as directed graphs. The orientation of an edge $e = (x, y)$ corresponds to the definition of the *Physarum* element given in Figure 5.1. We stress that the orientation has no effect on the results of the simulation since *Physarum* elements are symmetric as are the chosen initial conditions. In the plots presented shortly we always show the value of i_p for all involved *Physarum* elements to illustrate the current flow. Thus if $i_p > 0$ current flows in the direction the edge is pointing to, i.e. from node x to node y. The reverse is true if $i_p < 0$. This information is important when interpreting how the flows split up at junctions of the *Physarum* network.

Unless noted otherwise, all presented simulations were initialized with the same starting values for all *Physarum* elements. The only exception are the initial capacitor voltages which were set at random. They are summarized in Table 5.2.[2] The number of executed rounds can be read off the plots directly.

For some topologies we observe periodic oscillations. To determine the period of these oscillations we explored three different methods of period detection. They are based on signal auto-correlations, FFT and simply determining the distance between subsequent peaks respectively. It turned out that while all three methods usually agree, the simplest method of extracting the period from the peaks directly yielded the most robust results and was thus adopted.

[2] Unfortunately, many plots appear to start at $i_p(r = 0) \neq 0$ instead of $i_p(r = 0) = 0$. This illusion is the product of the circuits instantly jumping from $i_p(r = 0) = 0$ to $i_p(r = 1) \neq 0$ in a single round.

5.7.1. Cycle of Physarum Elements

We begin by investigating the behavior of a *Physarum* network that is a cycle. Exemplary we present the result obtained for a 4-cycle consisting of 4 *Physarum* elements, see Figure 5.4a. Note that together with Figure 5.4b, a color coding is defined which serves as a legend for the remaining plots of Figure 5.4.

For a converged 4-cycle of *Physarum* elements we obtain precisely the results predicted by Lemma 2. Namely, current can either flow clockwise, counter-clockwise or not at all if the *Physarum* network is converged.

These scenarios are reflected in the behavior of the capacitor voltages $u_{C,e}(t)$ given by Figure 5.4d, Figure 5.4f and Figure 5.4h. It can be seen that convergence is rapid for all three depicted cases. Note, that only in Figure 5.4d the values converge to 0. This is the case in which the respective currents $i_{p,e}(t)$ converge to zero as well, see Figure 5.4c. The currents indicate that the flow through the entire cycle dies out. This is expected, since for this case exclusively, $\beta = R$ was chosen as a simulation parameter. As a result, currents are quickly damped to zero as can be seen in the proof of Lemma 2.

Figure 5.4e and Figure 5.4g illustrate the remaining cases treated in said proof. For a given period the *Physarum* elements of the cycle interact to eventually "agree" on a common, constant value of current. This value can either be positive, leading to counter-clockwise flow through the cycle or negative, resulting in clockwise flow. At present, we do not know how the circuit decides on the sign of the current. Either the initial choice of the capacitor voltages determine the sign or it is random. This question deserves further investigation in the future.

We have repeated this experiment for various n-cycles with $n \in [3, 10]$ obtaining identical results. We find that edges agree rapidly on the flow for all tested n. It seems that for larger n the circuit needs progressively longer to converge. Unfortunately, no precise statements about convergence and its dependence on the size of the cycle are available at this point.

Finally, we remark that the above results are only valid for converged cycles. Extreme initial conditions can also be chosen such that the *Physarum* network does not converge. For a 4-cycle anti-phase oscillations between pairs of edges have been forced in this way.

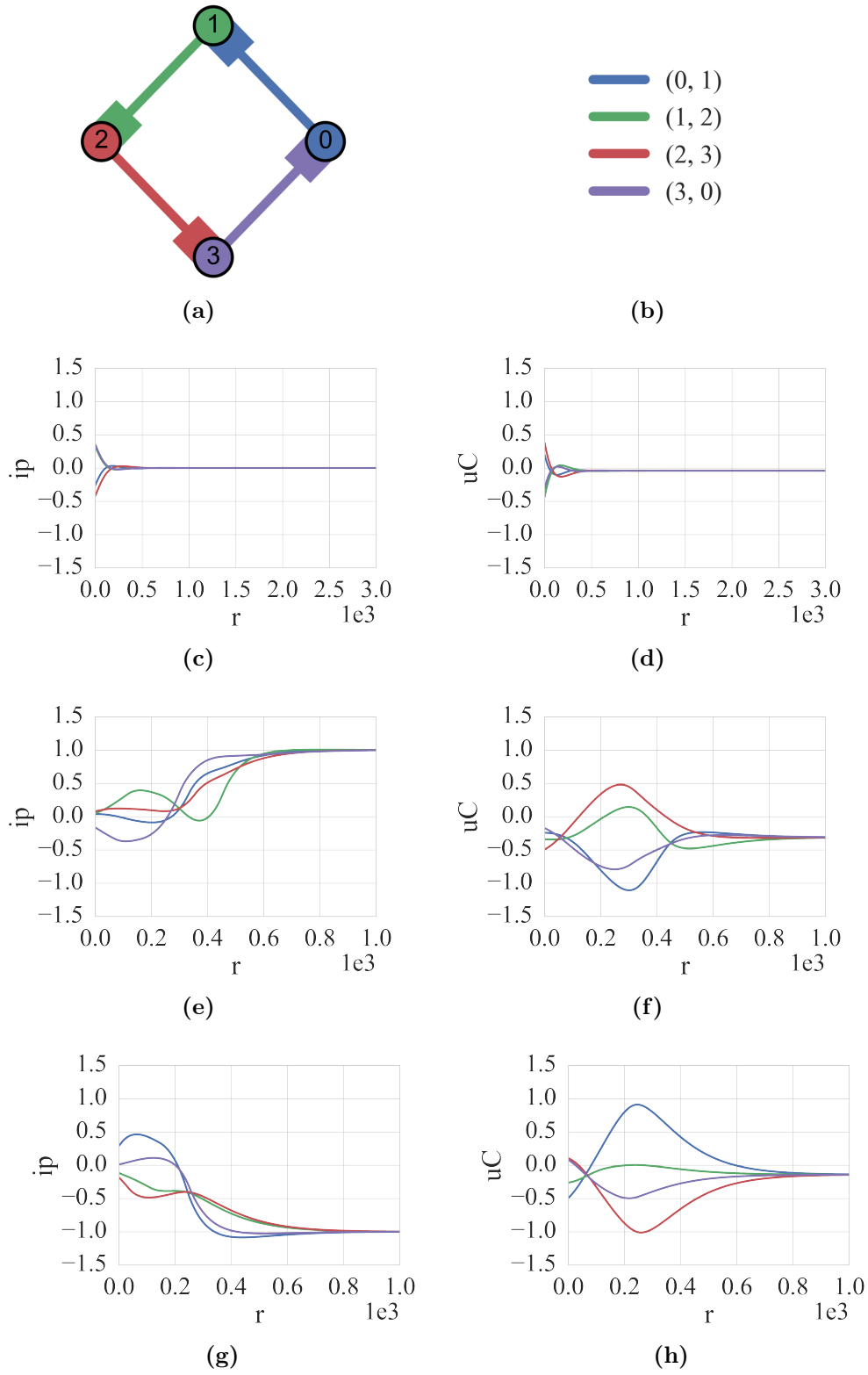


Figure 5.4.: Cycle of 4 *Physarum* elements.

5.7.2. Diamond of Physarum Elements

The graph we investigate next is the so-called diamond graph depicted in Figure 5.5a. This topology is also widely known under the name of Wheatstone graph or Wheatstone bridge. It consists of a 4-cycle with an additional "bridge" edge $(2, 0)$ added. Again Figure 5.5b defines a color coding which serves as a legend for the remaining plots of Figure 5.5. We show the results of three distinct runs of the simulation.

Similar to the 4-cycle explored previously, we investigate several scenarios. In all of them the capacitor voltages $u_{C,e}(t)$ quickly converge, see Figure 5.5d, Figure 5.5f and Figure 5.5h. However, due to the additional edge introduced to the *Physarum* network, edges do not converge to the same values anymore but split up in three distinct groups. The first group is formed by edges $(0, 1)$ and $(1, 2)$. The second group are edges $(2, 3)$ and $(3, 0)$. And the last group is the bridge edge $(2, 0)$. Note that in all scenarios the voltages show that the first two groups are symmetric around the value the center edge converges to. Although the capacitor voltages show identical behavior, very different currents can be accommodated.

Figure 5.5c shows a scenario, where 1.5 units of current flow through the center edge to split evenly at node 0. The split flow then circles back to node 2 showing appropriate signs and values. Figure 5.5e shows the same situation but with the flow on the center edge reversed. Naturally, there is no reason to assume that flow must split up symmetrically. Figure 5.5g shows one of many possible unbalanced scenarios. As in the case of the 4-cycle, we do not know the details of the splitting process nor its dependence on initial conditions at this point.

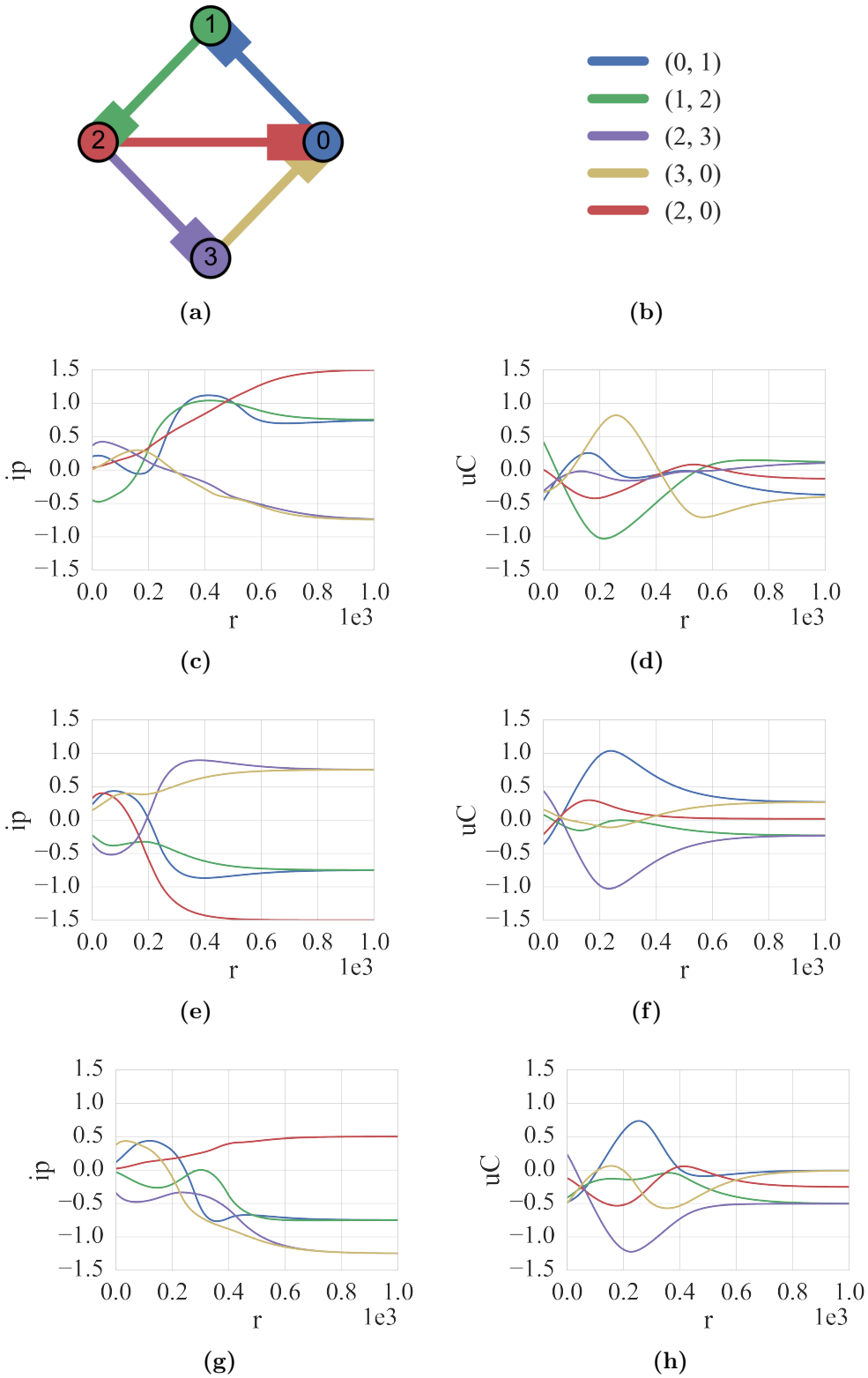


Figure 5.5.: Diamond of *Physarum* elements.

5.7.3. Paths of Physarum Elements

Next we look at paths of *Physarum* elements. Here no circulation is possible but neighboring *Physarum* elements still interact with each other leading to non-zero currents. Figure 5.6a depicts a path consisting of 5 chained *Physarum* elements. Again Figure 5.6b defines a color coding which serves as a legend for the remaining plots of Figure 5.6. We show the results of three different sub-path of Figure 5.6a all starting at node 0. The paths have lengths $l \in [2, 3, 4]$.

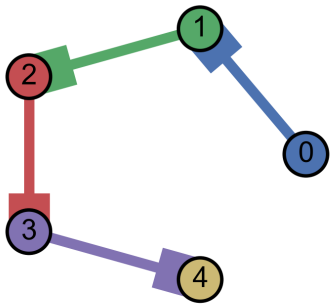
First, note the capacitor voltages of at least one edge does not converge for any of the depicted paths, see Figure 5.6d, Figure 5.6f and Figure 5.6h. Almost all voltages are quasi-periodic and thus none of these *Physarum* elements is converged.

A similar behavior is observed for the currents illustrated in Figure 5.6c, Figure 5.6e as well as Figure 5.6g. Note that for all paths we find that the currents of all but one edge periodically reverse their signs, i.e. flow reversal occurs naturally. At present, we do not know why there remains one edge with zero flow.

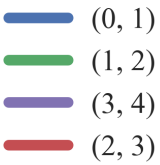
The distinguishing feature of these *Physarum* elements seems to be the period of the oscillations. For paths of length 3 we repeated the simulation 100 times with different random initial conditions. Without fail the *Physarum* network produced a signal with a period of $T = 1087$ rounds in every execution. We thus conclude that the period is invariant under initial conditions within the ranges we explored.

Furthermore, we find that the period of the oscillations decreases with increasing path length. For small path lengths the observed values suggest a linear relation between path length and period as a best fit. To explore the matter further we extended the simulations to paths lengths of up to 10. We found that a simple functional relationship between path length and period cannot be confirmed for all tested lengths. Rather, for path lengths longer than 7 more complicated oscillation patters are found to repeat themselves. Here it appears that higher order oscillations come into play. At present their meaning is not clear to the authors. We conclude, however, that there is a complex relation between the oscillation patterns and the size of the underlying *Physarum* elements that warrants further exploration.

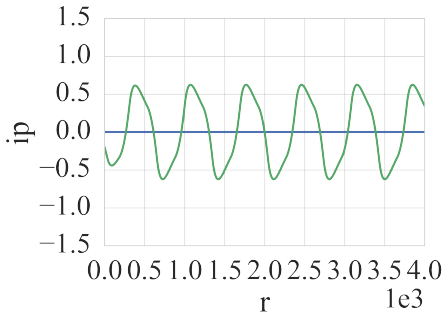
Another aspect awaiting investigation is the nature of the phase shifts observed between the edges of the circuits.



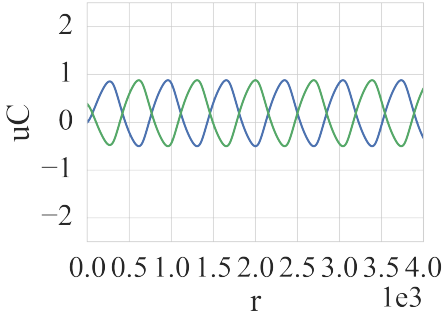
(a)



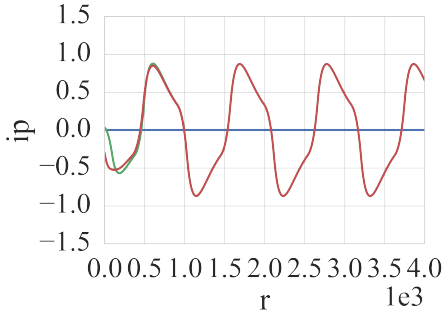
(b)



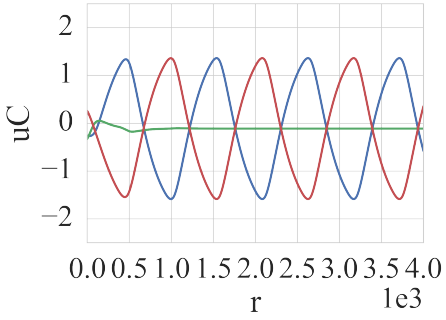
(c) Period $T \approx 694$ rounds.



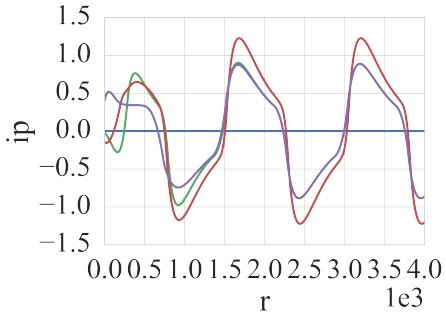
(d) Period $T \approx 694$ rounds.



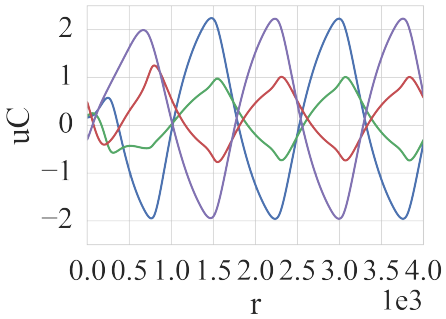
(e) Period $T \approx 1087$ rounds.



(f) Period $T \approx 1087$ rounds.



(g) Period $T \approx 1522$ rounds.



(h) Period $T \approx 1522$ rounds.

Figure 5.6.: Various paths of *Physarum* elements.

5.7.4. Trees of Physarum Elements

Let us study trees of *Physarum* elements next. The situation is similar to paths, with the added possibility of flows splitting up non-trivially at nodes of degree 3. Figure 5.7a depicts a tree rooted at node 0. We stress that the special status of root node exists only in the picture serving for better illustration. From the viewpoint of the model, all nodes are equal and the orientation of the edges matters not due to the symmetry of *Physarum* elements. As before Figure 5.7b defines a color coding which serves as a legend for the plots of Figure 5.7.

Similar to the study of paths, we find that a *Physarum* network with a tree topology does not converge, see the capacitor voltages in Figure 5.7d. Again we observe periodic behavior for both currents and capacitor voltages with a period of $T \approx 1701$ rounds. Of particular interest is the behavior of the currents as given by Figure 5.7c. After a brief period of interaction, the tree exhibits perfect anti-phase oscillation between its subtrees. Looking at the edges $e_1 = (0, 1)$ and $e_2 = (0, 2)$ we see that whenever a positive current flows along e_1 an equal current with opposite sign flows through e_2. The same phase shift can be observed for the subtrees rooted at node 1 and node 2 respectively. This exact type of behavior was observed in 100 independent simulations with different initial conditions.

With regards to the period we observe identical values across all 100 runs. Similar to the path, the period of the tree is robust to varying initial condition given the range we tested. A proof solidifying this statement is missing at this point.

Let us discuss the behavior of the phases. Note that with respect to Figure 5.7a we may state that the currents through the left and the right subtree oscillate in perfect anti-phase. It is striking that the circuit itself suggest that it be viewed as a tree rooted at node 0. It is not clear why it is precisely the subtrees at node 0 that show this anti-phase entrainment. In principle we could choose any other node as the root, leading to different subtrees that should show different behavior. To shed light on these observations further numerical studies of various tree layouts could be useful.

5.7.5. Two Linked Cycles of Physarum Elements

Finally, we combine the features of paths and cycles to obtain a *Physarum* network as shown in Figure 5.8a. Figure 5.8b defines a color coding which serves as a legend for the plots of Figure 5.8.

The proposed topology is meant as a first attempt of exploring the connection between the workings of our model and the behavior of live *P. polycephalum*. We do so by interpreting the *Physarum* network in Figure 5.8a as an abstract representation of the way *P. polycephalum* was modeled as a living system of delay-coupled mechanical oscillators as mentioned in the short survey of existing modeling approaches [166]. See Section 1.3.1 for details. The two 3-cycles represent the reservoirs while the path of length 1 constitutes the channel connecting them. In the original wet-lab experiment *P. polycephalum* showed rich self-synchronizing oscillation pat-

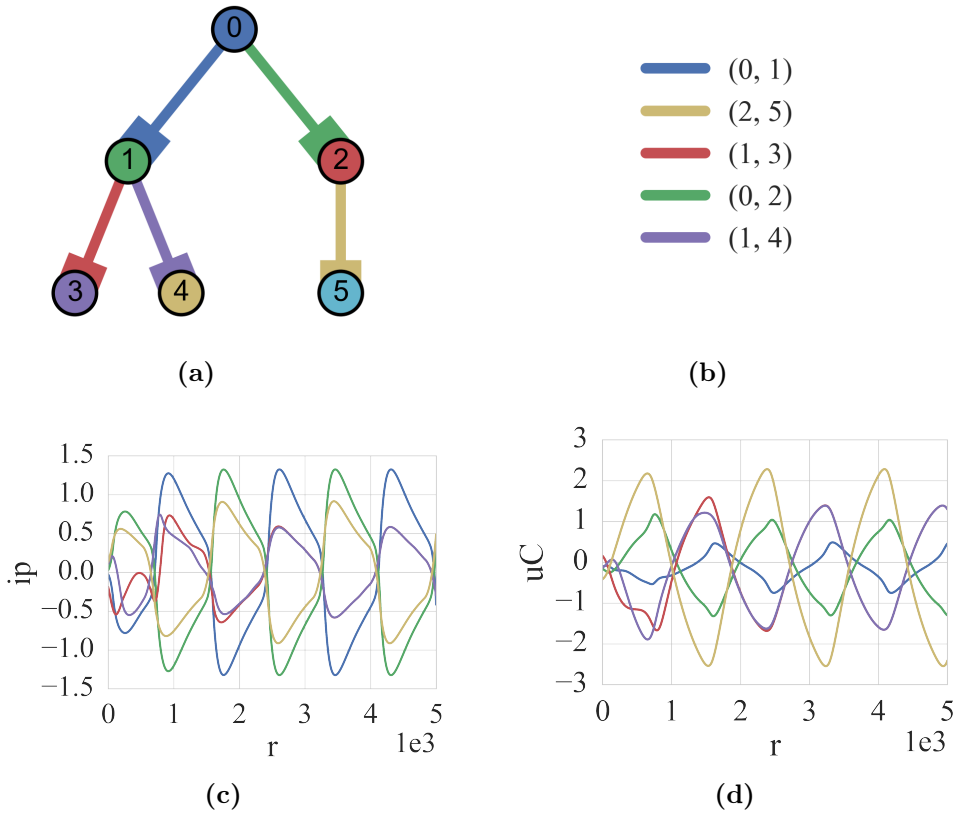


Figure 5.7.: A simple tree of *Physarum* elements.

terns between the distinct reservoirs with both in-phase and anti-phase entrainment. It is interesting to ask, whether our model can reproduce any of the observed effects.

To explore the behavior of our model, we present 3 distinct simulation instances. First we look towards the capacitor voltages to find symmetric oscillations in perfect anti-phase between the two cycles. After a brief period of entrainment, the edges of the two cycles converge symmetrically around the value the channel edges $(0, 5)$ converges to, see Figure 5.8d, Figure 5.8f and Figure 5.8h. The picture is similar to what was observed for the diamond graph, except that the non-bridge edges, i.e. the cycle edges, can now sustain oscillation. While this anti-phase entrainment is reminiscent of the behavior delay-coupled oscillators, we were not able to get our model to show in-phase entrainment. In fact, given the invariants we were able to proof, it is unlikely to be possible.

With regards to the currents flowing in the *Physarum* network, we observe scenarios that are intuitive. The currents in both cycles can be identical and flowing in the same direction, either clockwise, see Figure 5.8e, or counter-clockwise, see Figure 5.8c. Furthermore, the currents can flow in the opposite directions adding two more scenarios. Figure 5.8g shows one of them with the l.h.s. cycle showing counter-clockwise flow while the r.h.s. cycle exhibits clockwise flow. A scenario where currents are damped is not shown. In contrast to cycles, the coupling between the two cycles induces oscillations in the circular flows. Interesting in all scenarios, is the behavior of the channel edge $(0, 5)$ which shows periodic flow reversals with a period of $T \approx 1629$ rounds and large peak flows. Again 100 repetitions of the simulation suggests the period to be independent on the initial conditions within the range we tested.

Clearly, the model we propose yields only a crude approximation of the behavior of *P. polycephalum*. Based on the high degree of abstraction and the relatively simple model, it is surprising that effects like flow reversal and anti-phase entrainment emerge naturally. It is also apparent, that we are limited to simple topologies of *Physarum* elements if we are to make sense of the behavior analytically. Of course this does not prohibit using the model as a black box for more complicated and/or larger graphs.

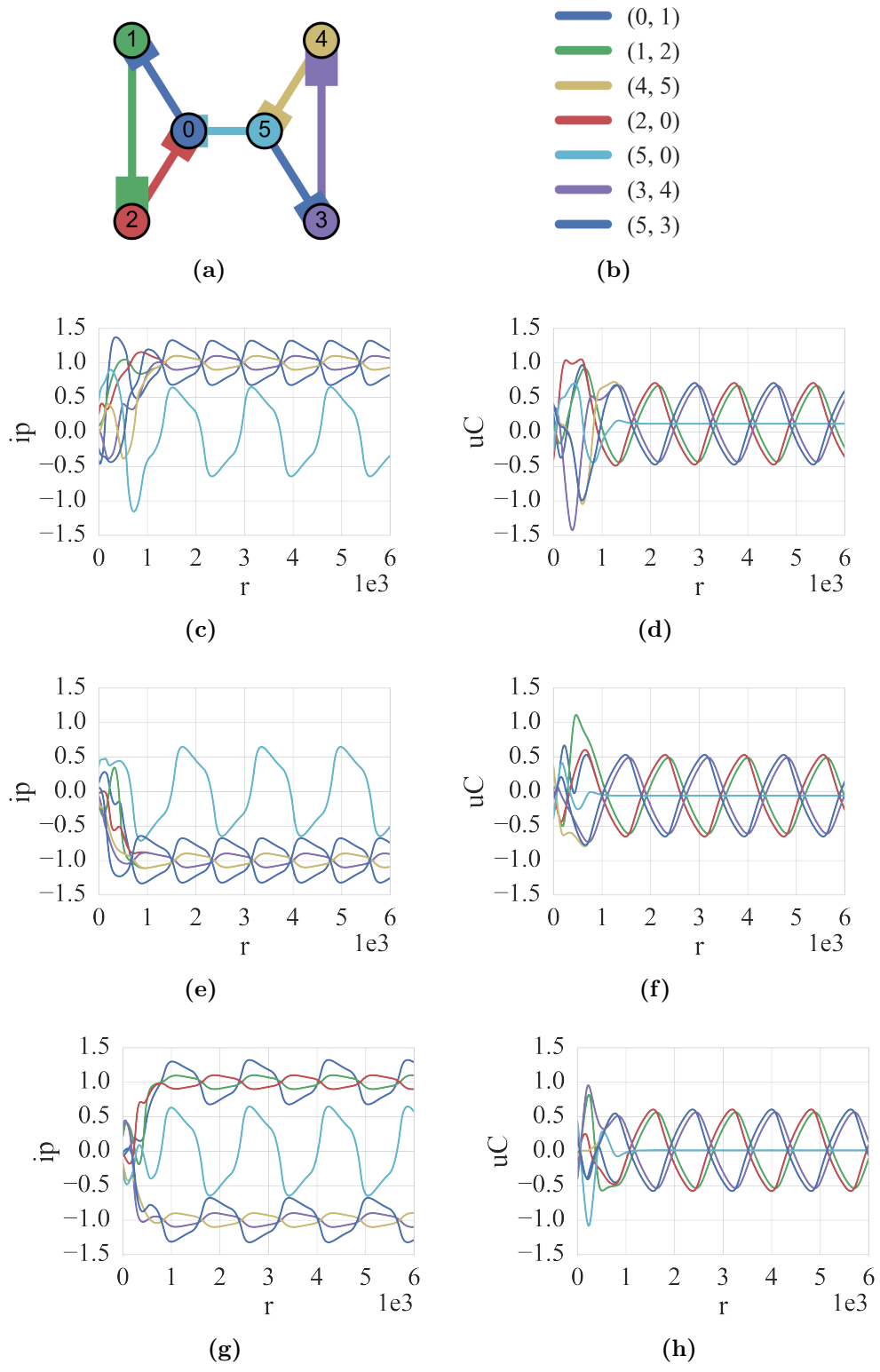


Figure 5.8.: A *Physarum* network designed to mimic aspects of the delay-coupled oscillator model of *P. polycephalum*.

5.7.6. Physarum Elements and Changing Topology

Finally, we show that the model is capable of adapting to changes in the topology of the underlying *Physarum* network. This is not surprising given that at the heart of our model are electronic circuits. It is possible that a formal proof of this property can be obtained.

Here we present numerical evidence based on modifying the graph presented in the previous section, see Figure 5.8a. Needless to say, Figure 5.8b defines a color coding which serves as a legend for the plots of Figure 5.9.

In Figure 5.9a and Figure 5.9b the channel edge $(5, 0)$ was removed from the *Physarum* network at point $r = 4000$ and reinserted at $r = 8000$. The plots illustrate how oscillations cease immediately at $r = 4000$ in favor of steady flow of opposite sign through the two cycles. The same speedy adaptation is seen in reverse when the channel edge returns at $r = 8000$.

The procedure of changes is more complicated in Figure 5.9c and Figure 5.9d. Here we start the simulation with the channel edge $(5, 0)$ as well as one edge from the right cycle $(2, 0)$ removed. As expected the plots show both the behavior of a path of length 2 as well as constant flow in the remaining cycle. At $r = 4000$ we bring back the channel edge. At this point the *Physarum* network consists of a 3-cycle with a path of length 3 attached to it. Note the complex signal shown by the circuit. The influence of the path is clearly visible, inducing oscillations in the cycle. At $r = 8000$ we bring back the remaining edge and observe how the system jumps back to the expected initial behavior.

These preliminary examples illustrate that changes in topology are naturally accommodated by the model. A more systematic investigation is a possible task for the future. Note, that when obtaining formal robustness properties for *Physarum* networks one may start by scanning related literature for classical electronic circuits.

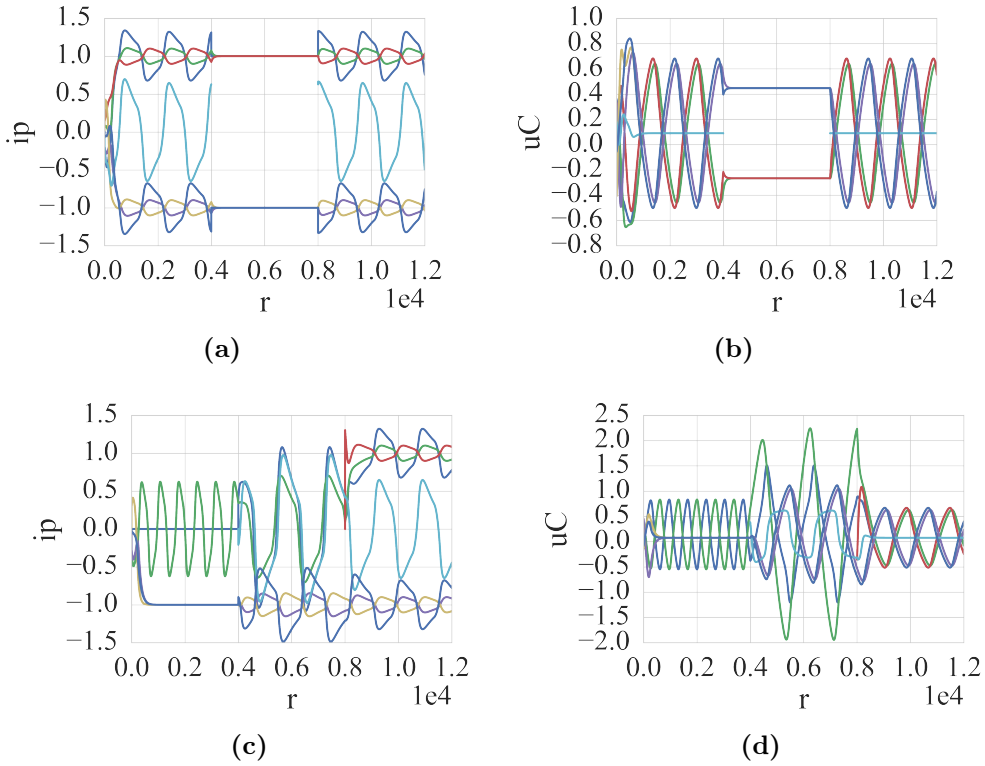


Figure 5.9.: *Physarum* networks changing every 4000 rounds.

5.8. Discussion

We have proposed a preliminary model representing the oscillatory flows observed in networks formed by *P. polycephalum* as currents in electrical networks. Our approach is based on the three-element Windkessel model. This approach has successfully been used in the past to model human cardiovascular networks which strongly resemble the vein networks of *P. polycephalum*. The major difference between the two is the fact that slime molds do not have one powerful central pump, i.e. a heart, capable of producing flows. Rather, each and every vein of the slime mold network may act freely and contribute to the flow as an independent peristaltic pump. Since a vein may be connected to several neighboring veins, non-trivial local interactions arise. To model emergent global oscillations we propose an extension to the Windkessel model, namely current controlled voltage sources. The resulting electronic *Physarum* elements are then connected according to a given topology to form a *Physarum* network. The electric currents induced by such networks exhibit complex emergent flow patterns reminiscent of the flows observed in live *P. polycephalum*.

In a symbiotic fashion we combine analytic and numerical methods to explore the characteristics of the resulting model. We find first and foremost, that our model can be discretized and solved to yield current flows for *Physarum* network of arbitrary topology. In practice we restrict our simulations to simple graph classes of limited size. They illustrate that the model is fully distributed as it requires no central control. It exhibits self-organization leading to coordinated flows and global anti-phase entrainment. In addition to that, we determine that the model is robust to changes of the underlying topology of *Physarum* networks. These are qualities attributed to live *P. polycephalum*. Furthermore, we hope that our model also inherits some of the natural efficiency attributed to the behavior of *P. polycephalum*. We are reduced to hope in this instance, because there is no way of asserting the statement in a systematic way.

When we began thinking about meaningful extensions to the Windkessel model, we noticed that we were frequently faced with decisions about what assumptions to accept during the initial stages of modeling. First and foremost we decided to keep the model simple enough to be able to proof facts about it. This is in line with our desire to obtain a prospective candidate model that would serve as a basis for future attempts of natural computing. Thus we placed significant emphasis on implementing simplifying assumptions. Needless to say, these come at the cost of a less accurate description of live *P. polycephalum*. While this is unfortunate, in the context of computing inspired by nature it is of minor concern. In summary we decided to differentiate the original general model such, that modeling accuracy is sacrificed for ease of analytical treatment.

Naturally, a different approach is necessary if your goal is synthesis of nature by means of computing. Here one seeks to obtain a model providing modeling accuracy and ultimately, predictive power. To do so, trading manageable complexity for a higher degree of modeling accuracy is acceptable. This entails replacing some of

the following simplifying assumptions with more meaningful alternatives in order to obtain a model that is more interesting from a biophysical point of view. Perhaps most influential of all is the assumption that the electric resistance is constant and the same for all edges in the network. Note that the electrical resistance of *Physarum* elements translates to hydraulic resistance of veins in live *P. polycephalum*, a quantity which strongly depends on the width of the veins. In fact, we have established in Section 4.3 that the distribution of widths in real *P. polycephalum* is not constant but is more likely to follow a gamma or log-normal distribution. Clearly, the model could be made more meaningful by incorporating width distributions of *P. polycephalum* graphs. Furthermore, the topologies of real *P. polycephalum* graphs which are conveniently available in the SMGR could directly be used to run the model on.

Incorporating these improvements including the move from simple small graphs to more complex *P. polycephalum* graphs with more degrees of freedom for the edges likely shuts down our hopes of obtaining analytical statements about what is going on. Note that such a model may yet be solvable given the right numerical tools. It could still form a valid basis for natural computing, however the analysis of obtained algorithms is expected to range somewhere between challenging and seemingly impossible.

Note that augmenting the Windkessel model such that modeling accuracy is maximized does not automatically yield a model with predictive power. What remains is to determine a way to set biophysically meaningful parameters for the *Physarum* elements. Unfortunately, the authors have no suggestion as to how to resolve this problem at present. A closer collaboration with biologist and biophysicists seems necessary to tackle this question.

The authors are convinced that both described approaches supported by an augmented Windkessel model are valid and should be explored further. Indeed a manuscript is being prepared documenting our own attempts of deriving a distributed natural computing algorithm based on this model. While it is not necessary for a natural computing algorithm to closely resemble its source of inspiration, it would be nice if a connection between the way such an algorithm works and the way *P. polycephalum* operates could be established. This task will likely require insights obtained in the pursuit of models of higher biophysical relevance and further strengthen the interdisciplinary appeal of natural computing with *P. polycephalum*.

6 | Summary

The present thesis intends to pave the way towards distributed natural computing inspired by the slime mold *Physarum polycephalum*. The networks formed by this humble organism exhibit an extraordinary plasticity and presumably support efficient circulation of protoplasmic fluid through them. For this reason the complex dynamics observed in *P. polycephalum* have repeatedly been compared to optimization processes. Devising models that both capture the natural efficiency of this organism and at the same time form a suitable basis for the development of natural computing algorithms, is an interesting an challenging proposition.

This thesis documents a series of sequential research efforts that build on-top of each other, working towards this goal. These can be subdivided into three broad categories: Experimental work, data analysis and modeling efforts.

In the first category we design and execute a number of wet-lab experiments with *P. polycephalum*. The experimental setup is geared towards producing images which document the topology of *P. polycephalum* vein networks and their time development. Next, we turn the images depicting networks into actual graphs enabling subsequent analysis. To do this we develop a custom software called Nefi. At this point, we make all obtained data, i.e. raw images and corresponding graphs available to everyone. To facilitate this we introduce a dedicated publicly available repository revolving around slime mold data, the Smgr. The goal of this repository is to enable data reuse and to invite others to join in our efforts of fostering a practice of increased data sharing.

In the data analysis category of this thesis we scrutinize the *P. polycephalum* graphs we obtained previously. To this end we design a set of observables aimed at capturing various properties of the vein networks. These include structural information about paths, face cycles as well as graph cuts. Finally we investigate percolation properties gauging the robustness of networks formed by *P. polycephalum*. The results obtained by us form an extensive catalog of characterizing data which help quantify properties of slime mold networks.

The last category of this thesis is concerned with modeling of *P. polycephalum* which is the immediate precursor to deriving natural computing algorithms. Here we rely on electronic elements to model the dynamic flows observed in live the live organism. Our approach is inspired by the modeling of the human cardiovascular system but introduces additional current controlled voltage sources to mimic the effects of peristaltic pumping. These elements are connected according to a given topology in order to form electronic networks representing vein networks of *P. polycephalum*. The model is simple but has the advantage that it analytically tractable

on top of being solvable numerically. A preliminary *in silico* exploration of its properties shows, that just like *P. polycephalum*, it operates fully decentralized and is robust against changes in the underlying network topology. Above all, it yields complex current flows that resemble flow patterns observed in live *P. polycephalum*. In particular, emergent anti-phase entrainment and flow reversals are observed. As such the model we obtain constitutes a promising candidate for the development of distributed natural computing algorithms inspired by *P. polycephalum*. Efforts in this direction are underway and will be reported separately from this thesis.

What started as a fascination with existing natural computing algorithms inspired by *P. polycephalum*, organically developed into a desire[1] to work towards our own natural computing approach. Little did we know when we started that we were embarking on a rather interdisciplinary journey.

Note that at every step of the way we put a strong emphasis on making sure that everything we do, is at least as useful to others as it is to us. Needless to say, this almost always required us to go the so-called "extra mile" investing considerable time and effort. However, the idea of creating additional value for others, on-top of obtaining novel scientific results, is rather satisfying and thus quickly became an underlying principle of our work. Today it can be seen resonating throughout all of this thesis.

Our network extraction software Nefi for instance was designed to process images depicting networks originating from various domains. In particular, it is not limited to the applications presented in this thesis. Rather, our tool and its multi-purpose design has the potential to be of continued use in a variety of unforeseen applications.

The same can be said about our slime mold data repository, the Smgr. Anyone can download its data and start using it right away. At the same time experts are invited to contribute their hard earned data and results to the Smgr in order to increase the reach and impact of their work. We hope that the idea of the Smgr will resonate with the research community concerned with slime molds.

Our efforts of characterizing the networks formed by *P. polycephalum* yielded a catalog of observables spanning a wide array of different graph properties. While the obtained results are not exactly ground-breaking, they have potential implications for evaluating and guiding all sorts of theoretical modeling approaches regarding *P. polycephalum*. Model predictions that agree with data in the catalog increases the trust in a given model. At the same time discrepancies between predictions and catalog data hopefully suggest improvements to the model. Thus, the data and results presented in this thesis may be beneficial to the modeling efforts of others.

Last but not least, there are our own modeling efforts regarding the flows observed in *P. polycephalum*. Here we are working towards computing inspired by nature. By making various simplifying assumptions in the process of abstracting the flow dynamics of the organism, we emphasis low model complexity at the cost of reduced biophysical modeling power. Our aim is to strike a balance between both aspects such that the model remains manageable yet still captures the complex flow

[1] Some would even use the term "obsession".

patters displayed by *P. polycephalum* to a sufficient degree. Since our interests lie in the domain of natural computing, this trade-off is acceptable because a model of manageable complexity is more likely to eventually lead to algorithms which can be analyzed analytically.

Now, one may rightfully ask, why this trade-off cannot lean the other way, leading to a more reliable description of *P. polycephalum* with an emphasis on biophysical accuracy at the expense of increased complexity. In other words placing an emphasis on the synthesis of the behavior of *P. polycephalum* by means of computing. The first steps towards this direction probably start with omitting or improving some of the simplifying assumptions we made during the modeling process. It would be intriguing to investigate to which extend variations of the model are capable of producing meaningful predictions with regards to the behavior of *P. polycephalum*. At present we believe that such an approach could be of pronounced interest regarding questions of biology and biophysics. It certainly appears feasible and suggests itself for future in-depth explorations.

Note that the two different trade-offs between modeling power and complexity we illustrated are geared towards different goals and seem to support disjoint types of research questions. At first glance, they appear to be in stark contrast to each other. However, we propose to adopt a different view of the matter entirely. In fact, we strongly believe that they are merely two of many approaches which can and should support each other in contributing different clues to the study of *P. polycephalum*. Combining various approaches to the study of this humble organism creates additional value but requires an open mind with a strong willingness to work in an interdisciplinary manner. The author tried to live up to this ideal with this thesis.

A | Guide to Using NEFI

Here we document our experiences with using Nefi. As discussed in Chapter 2, the combination of input image quality and Nefi's segmentation algorithms makes or breaks the resulting graph. Let us first discuss the properties of ideal and non-ideal input images. Our goal is to give the prospective user an idea of what to avoid and what to look out for regarding inputs. Furthermore, we add some pointers on how to deal with challenging inputs and what one can try to do in order to improve the output of Nefi.

A.1. Properties of Ideal and Non-ideal Images

Since, the determining factor of the quality of Nefi's graph extraction is the segmentation step, ideal images should enable a nearly perfect separation of foreground and background. Such images have high contrast between the depicted structures of interest and the background. At the same time it is very important that images are free of strong reflections or shadows because such areas are likely to show an even higher contrast to the background than the actual structures of interest. As a result, the segmentation algorithms are prone to identify these regions as foreground causing the actual structures of interest to be ignored. The presence of strong color or brightness gradients can have similar detrimental effects and should thus be prevented if possible. See Figure A.1 and Figure A.2 for examples of challenging images which Nefi will have difficulties working with.

Another factor that influences segmentation, and by extension graph detection, are contaminations of different origin. Examples include parts of the image which might not belong to the object of interest at all. Such regions should be removed before loading the image into Nefi, see Figure A.3.

Figure A.4 depicts an image lacking in a similar way. The image contains objects that are technically part of the network one would be interested to extract, however, they are not suited very well to be represented as a graph. In particular, they will be picked up correctly in the segmentation step yielding four large areas of white pixels. Subsequently, thinning will try to reduce these to lines resulting in more or less unpredictable results. While the remaining parts of the structure will be processed correctly, such images do not constitute ideal candidates for processing with Nefi.

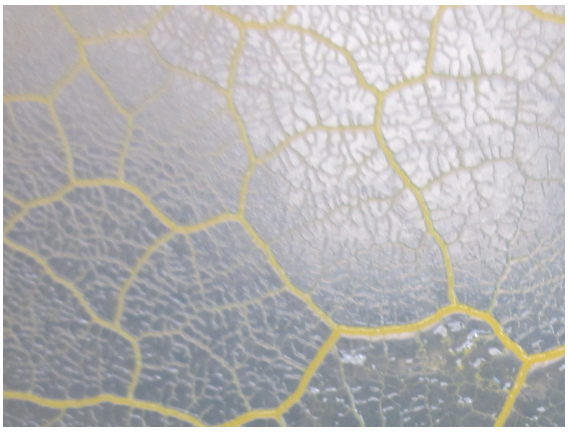


Figure A.1.: This image of *P. polycephalum* contains strong light reflections in the upper right quadrant which will cause NEFI's segmentation algorithms to be lead astray. The fact that the image is not properly focused is less of a problem in comparison.

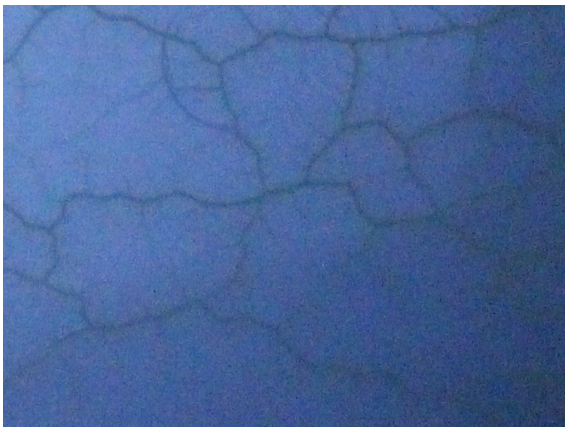


Figure A.2.: This image of *P. polycephalum* was not illuminated evenly from below. As a result it contains a brightness gradient which is detrimental for many of the segmentation algorithms currently implemented in NEFI. The fact that the image contains a lot of visible noise makes it a bad candidate for processing with NEFI.

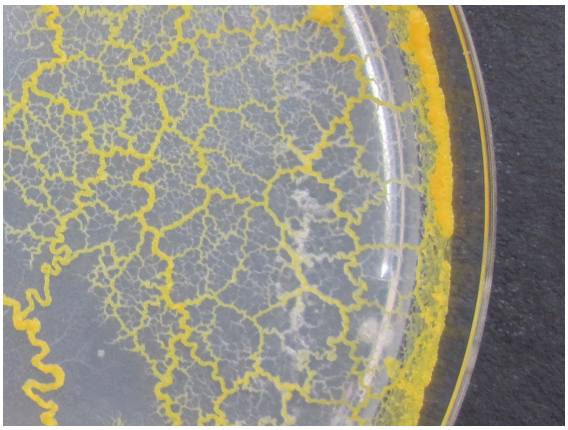


Figure A.3.: In addition to the network of *P. polycephalum* the image contains the edge of a petri dish and pieces of the background on the right hand side (as well as light reflections). Prior to any attempt of processing the image, petri dish and background should be removed.

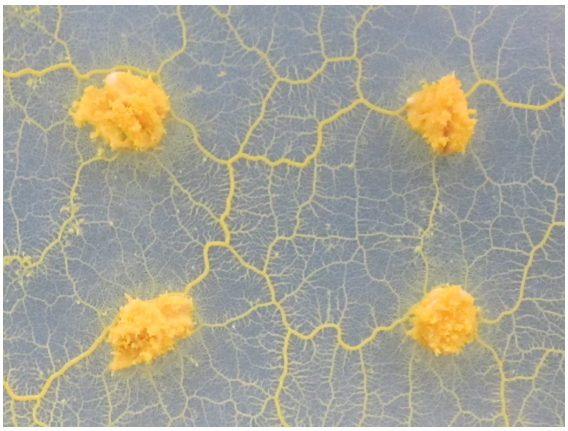


Figure A.4.: The network of *P. polycephalum* depicted in the image contains 4 massive, non-network-like regions (oat flakes completely covered by the mold). After segmentation, thinning will try to reduce these regions to lines leading to unpredictable results. The remaining parts of the network, however, will be extracted correctly.

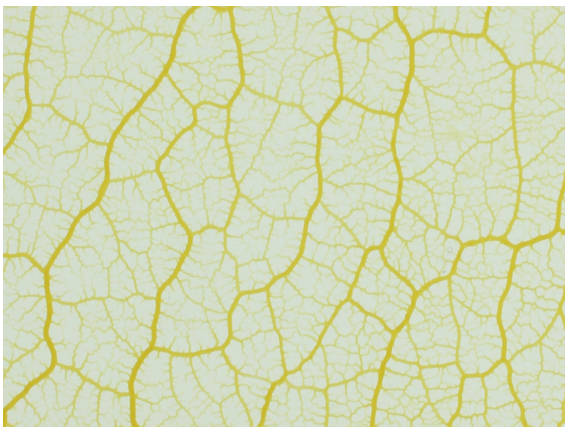


Figure A.5.: An ideal image of *P. polycephalum*. The image has been obtained using bright field illumination and was produced in a collaboration with the KIST Europe.

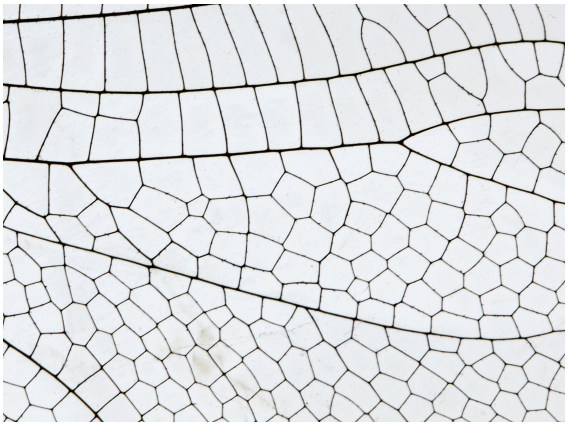


Figure A.6.: An ideal image depicting a detail of the wing of *A. junius*. Image courtesy of Pam and Richard Winegar.

Summarizing this information we have the following desirable properties for an ideal image:

- High contrast between foreground and background.
- Uniform background void of reflections, shadows as well as color or brightness gradients.
- No contaminations that might disrupt the segmentation process.

When producing images to be processed with NEFI one should strive to fulfill these properties whenever possible. See Figure A.5 and Figure A.6 for examples of very good input images.

A.2. Dealing With Challenging Images

Nefi operates best on images produced under controlled laboratory conditions that fulfill the properties described in the last section. However, more difficult inputs may still be processed, but most likely at the cost of reduced quality. Based on our experience when dealing with more challenging input and the results of our evaluation, we are able to formulate the following recommendations for the usage of Nefi:

Otsu's method may be used on noisy or blurred images. It will do reasonably well as long as the image has a high contrast between network and background. Adaptive threshold, watershed based on adaptive threshold, and GrabCut with deletion and erosion may even perform slightly better under these conditions. Otsu's has the advantage that no parameters have to be set in order to get good results. The choice should be based on the desired degree of resolution of the extracted graph. We would recommend these methods to process Figure A.3 after removing areas that are not of interest.

Adaptive threshold and watershed based on adaptive threshold may be used if differences in contrast between foreground and background are local and not too strong. If this is the case, good results may still be obtained. Both methods allow to analyze images that contain a color gradient in the background or that contain edges which have differing levels in brightness. One might try to process images like Figure A.1 and Figure A.2 with these methods while experimenting with different parameter settings. However, such images remain challenging. Nefi's current algorithms may not deliver sufficient results.

Preprocessing methods like Gaussian and Median Blurring, Denoising as well as Bilateral Filtering can be used to remove small artifacts, contaminations or irregularities from the image. Although these methods can reduce the amount of artifacts produced during segmentation and thinning, improvements come at a price. For example, imposing a strong blur causes the depicted edges to appear slightly wider. This effect will propagate through the pipeline causing the edge widths of the final graph to overestimate the true widths depicted in the image. We recommend to use preprocessing with care especially if a high accuracy regarding edge weights is required.

Graph filtering enables the removal of unwanted artifacts and spurious vertices caused by irregularities in the input which propagate through the pipeline. However, filtering can only repair the result up to a given point. While the ability to add custom filters is powerful, it appears pointless to filter a graph established on the basis of a failed segmentation. We encourage users to visually verify the integrity of the established graph and only then to proceed with filtering.

We anticipate users to encounter images NEFI's algorithms will be unable to handle properly. In this situation the user will have to rely on different segmentation solutions.

As a first suggestion, we recommend to user to look for available software specializing in segmentation. In this regard we like to point out the Kitware, ITK Project [87]. It's *C++* code base has been developed by the medical image processing community and serves as the basis for many other specialized tools that deal with image processing as well as segmentation.

After a segmented image has been obtained using specialized third party segmentation software, NEFI can take this image and proceed with graph detection and filtering directly.

If no proper software is available, the user has the option to extend NEFI's segmentation capabilities by adding more sophisticated code. When doing so, one can built on top of existing features already implemented in NEFI. The literature offers a wealth of different approaches to segmentation leading to algorithms of varying complexity. Before diving into any implementation efforts, we strongly recommend to survey existing methods and their domains of effectiveness by consulting [45], [137], [141].

B | Details of Data Acquisition

Let us now explain all steps involved in furnishing the KIST Europe data set in full detail. First, we describe how to setup and execute necessary wet-lab experiments, including the production of sclerotia [81]. Next, we explain how to turn the network structures depicted in the raw images into series of equivalent graphs. Finally, we illustrate how to establish unique node identities and track them within a given series of graphs.

B.1. Experiments

For our experiments we cultivate *P. polycephalum* (HU195xHU200) in a rectangular, 20 cm × 30 cm × 13 cm, translucent plastic dish ontop of a 10 mm layer of 1.25 % agar (Kobe I). To do so we place 1.5 g of dried *P. polycephalum* sclerotia crumbs along the short edge of the dish. We make sure to evenly spread them out such that a continuous and straight line is formed, connecting two adjacent edges of the dish. In the following we refer to this line as the *inoculation line*, see Figure 3.2. This concludes the preparation of the dish.

Since *P. polycephalum* is sensitive to light, we place the dish inside a large light-proof wooden box of 110 cm × 110 cm × 110 cm. Temperature and humidity inside were kept constant at 22 °C and 55 % to 60 % relative humidity. In our setup we rely on dried sclerotia, rather than plasmodium, because the former give exact control over the initial mass of *P. polycephalum* introduced to the dish. We make sure to keep the input masses, the properties of the agar layers and the environment constant to ensure consistent repetition of experiments. For a detailed description on how to produce dried sclerotia from an initial sample we refer the reader to Section B.3.

Inside the box we fix a digital camera (Canon EOS 645D, Lens EFS 18 mm to 55 mm) 16 cm above the dish. The camera is oriented perpendicular to the dish and centered right above it. Each shot captures a large area of 10 cm × 15 cm at a resolution of 5184 pixel × 3456 pixel in JPG format. With these settings 1 cm on the dish corresponds to 370.6250 pixel in the image. Since during graph extraction all lengths and widths measured are stored in units of pixel, this information can be used to map pixels back to centimeters.

To provide the necessary light for the camera to work inside the dark box we opt for bright field illumination using a negatoscope, also known as X-ray film viewer (Planilux, 2 × 15 W, emitting white light). It provides a large area of low intensity illumination which is uniform in space and time. By putting the translucent dish

ontop of the negatoscope the light that passes trough makes the structures formed by *P. polycephalum* visible to the camera overhead. By design we ensure optimal contrast between the networks and the background and eliminate all sources of reflections or shadows in the images. This is particularly desirable as such effects are diminishing the effectiveness of NEFI. This concludes the preparation of the box. A schematic of the complete setup can be seen in Figure 3.1.

After the prepared dish is placed in the box, it takes roughly 15 h for the organism to make its transition from sclerotia to plasmodium. Once the plasmodium begins to spread towards the far side of the dish we start capturing its growth progress by taking an image every 120 s using dedicated software (Motion detection software; Vulpessoft, DSLR Master). We stop capturing when the growing front first hits the adjacent wall of the dish. By doing so we minimize the probability of *P. polycephalum* moving back towards the inoculation line. We do not feed the organism throughout the entire experiment. This concludes one iteration of our experiments.

We repeat this experiment under constant conditions and obtain 81 image series depicting the growth of *P. polycephalum* and the networks it forms. Since there is a natural variability in the growth of the organism, we obtain series of different length and nature. We refer to this data as raw data. It is available in the SMGR.

We are aware of a potential caveat of our approach, namely the light source in the negatoscope emitting the full spectrum of white light. It is well known that *P. polycephalum* reacts to specific parts of the spectrum while it is insensitive to others [122]. Thus, ideally one chooses a light source such that the organism remains undisturbed. However, such a light source was not at our disposal so we decided to minimize the impact of the light by minimizing the time *P. polycephalum* is exposed to it. In particular we couple the triggering of the camera with the power supply of the negatoscope. Thus we ensure that the slime mold is illuminated a mere 1 s every 120 s. Since we did not observe any irregularities in our experiments known to be induced by light, we conclude that our precautions were sufficient.

B.2. Graph Extraction

Given the obtained raw data, we discard all series that do not show proper network formation. Thus the number of usable datasets is reduced to 54. For the remaining series we seek to describe the characteristic *P. polycephalum* networks by equivalent graphs. In addition to capturing the topology of the networks, we want to obtain a precise measure on the length and width of each vein observed. Furthermore, we want to establish the positions of the junctions of the veins in the image. Thus, we want to compute a weighted graph, whose nodes carry the positions of the junctions in the plane and whose edges carry weights corresponding to the length and the width of the observed veins.

To compute such a graph representation we rely on NEFI. This tool takes as input an image from the raw dataset depicting a network and returns a faithful representation of this network in form of a weighted undirected graph. NEFI of-

fers several different algorithms and a variety of settings to do graph extraction. Some experimentation was necessary to find a sequence of algorithms, a so-called pipeline, such that the returned graph representations preserve as much information as possible. The pipeline has been stored and is part of the dataset for reasons of reproducibility. Asserting the effectiveness of a pipeline is convenient and easy, since the tool allows to visually compare the computed graph with the network in the input image by drawing the former ontop of the latter. An example can be seen in Figure 3.6. For a more detailed discussion of the reliability of NEFI and how to use it, we refer to its project page and Appendix A.

The main caveat of NEFI is that, like any form of image processing or computer vision, the quality of the output strongly depends on the quality of the input. To obtain good results with this tool, the input image must be of high contrast and void of strong color gradients and other detrimental effects [52]. Due to the design of our experiments these requirements are largely satisfied. However, due to its implementation NEFI struggles with parts of the image that do not depict networks. In particular it fails to process regions depicting the inoculation line and the apical zone. For the network extraction to succeed these areas must be removed from the images or equivalently a region of interest must be defined excluding such areas. For consistency we define a specific region of interest for each given image series of the raw data set. A typical region of interest is seen in Figure 3.5.

To do so, we visually inspected every single image[1] of every sequence in order to decide on a maximal region of interest containing properly formed networks. It is common that somewhere within an image sequence *P. polycephalum* starts to deviate from "well-behaved" growth, effectively disqualifying the sequence from this point on. Examples include *P. polycephalum* suddenly reversing direction or spontaneously spawning new growing tips within an already established network. Thus we make two choices: First, for each sequence of images we find the longest usable subsequence and second, for each subsequence we decide on one region of interest. We store this information in small configuration files suitable for automated graph extraction. We point out that in general it has been beneficial if the choices of selection are made somewhat defensively, leading to a reduced likelihood of artifacts occurring in the graph detection process.

Given the configuration files and the extraction pipeline, NEFI can be used to batch process sequences automatically. Note that for some series, partially containing strong color gradients in the background, NEFI failed to properly segment the input images resulting in unusable graphs. This situation can be detected easily by inspection of the segmented images or the graph drawings produced by NEFI. The affected series and the resulting graphs are then excluded from further processing. While this reduces the number of usable graph series to 36, the remaining graphs capture the topology of the original *P. polycephalum* networks exceptionally well.

Note that the raw graphs obtained so far are likely to contain artifacts such as

[1] The reader is absolutely right to assume that inspecting several thousand images required an extraordinary amount of patience and time.

isolated nodes and dead-ends, see Figure 3.6. This is to be expected since Nefi cannot reliably resolve structures that are very fine grained, e.g. veins in the network with a width of less than 5 pixel. As a result small structures in the graph break up into several disconnected parts. In a similar fashion spurious isolated nodes can enter the computed graph. We strongly recommend anyone considering to work with the raw graphs to carefully inspect them first in order to assess whether these artifacts need to be removed using filters. In our experience, a moderate amount of filtering is always appropriate and considerably improves quality of the graphs, i.e. the degree to which they resemble the original *P. polycephalum* networks.

To deal with the mentioned artifacts Nefi comes with the possibility to apply filters capable of removing isolated nodes and dead ends. We have filtered all raw graphs to obtain the final set of graphs which we store in several file formats. In particular we removed all edges that are not on a cycle, i.e. dead ends are removed, and kept only the largest connected component. For all but the finest of veins the filtered graphs capture the structure of the original *P. polycephalum* networks extremely well. Furthermore they carry precise information about node positions, edge widths and edge lengths. For a detailed description of how to work with the actual graph files produced by Nefi we refer to its project page.

Lastly, we point out that the process of graph extraction described here is geared towards answering a particular set of research questions, see [53]. For a different set of questions changes may be appropriate and necessary. They can easily be implemented by starting with the raw graphs and applying different filters. Also, it is not difficult to go back even further to the original image sequences and select different regions of interest and different subsequences, leading to different series of raw graphs. Given Nefi and the possibility to use configuration files to automate the graph extraction, it becomes possible to adapt the data in the Kist Europe data set to various particular needs.

B.3. Continued Production of Sclerotia

Here we describe how to continuously produce dried sclerotia [81], from a small initial sample[2] of the same. First, we prepare a 22 cm × 32 cm plastic dish with a layer of 1 % Agar (Kobe I) and line up a generous amount of sclerotia evenly along the short side of the dish, forming an inoculation line. Ideally, the dish is kept in a light-proof box with high humidity at a temperature around 22°. That being said, forest-dwelling *P. polycephalum* is rather forgiving and will make do with room-temperature and room-humidity provided it is properly shielded from light.

After 10 h to 14 h *P. polycephalum* changes from the sclerotia state to the plasmodium stage and starts to explore the dish. From this point on we recommend to feed it with oat flakes every 4 h to 6 h. To do so, distribute a small amount of oat

[2] Starting samples of dried *P. polycephalum* sclerotia can be purchased online or shared via the *P. polycephalum* community, e.g. at the Slime Mold Collective.

flakes evenly across the inoculation line. Given the additional nutrients the organism will rapidly cover the whole dish at a rate of approximately $1\,\mathrm{cm\,h^{-1}}$. Shortly before it reaches the far side of the dish we proceed to transfer the organism to new containers. The mass covering the oat-flakes is moved to a new dish with ample room to keep the growth of the plasmodium going in a continuous fashion. The remainder of the organism is transfered to a bucket allowing it to dry, thus triggering the transition back to the stage of sclerotia. A detailed description of both steps follows.

First, we move the plasmodium at the inoculation line together with the accumulated overgrown oat-flakes to a new $22\,\mathrm{cm} \times 32\,\mathrm{cm}$ dish prepared with agar gel. To do so we cut the agar of the old dish underneath the inoculation line into suitable pieces, which we transfer to the new dish using a small spatula. We arrange the pieces of gel carrying the plasmodium such that a new inoculation line is formed. Soon the plasmodium will proceed to conquer the new dish. From time to time we add a small amount of oat flakes to the starting line to make sure *P. polycephalum* keeps growing steadily. Transferring to a new dish serves to keep the growth of the organism going continuously. This step can be omitted if continued production of plasmodium is not desired. If at some point the growth of the slime mold seems to have come to an unintentional halt, we recommend to make sure it is not to dry by carefully moistening *P. polycephalum* using distillate water and a plant sprayer.

Next, we move the remaining contents of the overgrown gel to a 40 cm high and 25 cm wide ordinary cylindrical plastic bucket. In particular we make sure to transfer the whole growing front since it constitutes a significant part of the biomass of the organism. Before we transfer the organism, we cover the bottom of the bucket with tissue paper and stick moist filter paper to the walls of the bucket such that its inside is completely covered. After a while the plasmodium starts exploring the bucket and is naturally drawn towards the moistened areas. Eventually it will move away from the drier bottom and move along the bucket walls covering the filter paper. As soon as most of the filter paper is occupied by *P. polycephalum*, the paper can be removed, wrapped up and stored to dry, triggering the transition from plasmodium to sclerotia. Letting the wrapped up filter paper dry in a cardboard box is simple and effective. After approximately 3 days, the cell mass is fully dry and in the state of sclerotia. They can easily be handled and stored for later use. Once dried, the sclerotia can be scraped off the filter paper and used with precision in follow-up experiments.

C | Supplementary Figures

This appendix contains figures supplementing Chapter 4. Note that the full set of goodness-of-fit plots for all data series is available online at the SMGR.

C.1. Goodness-of-fit Plots for Section 4.3

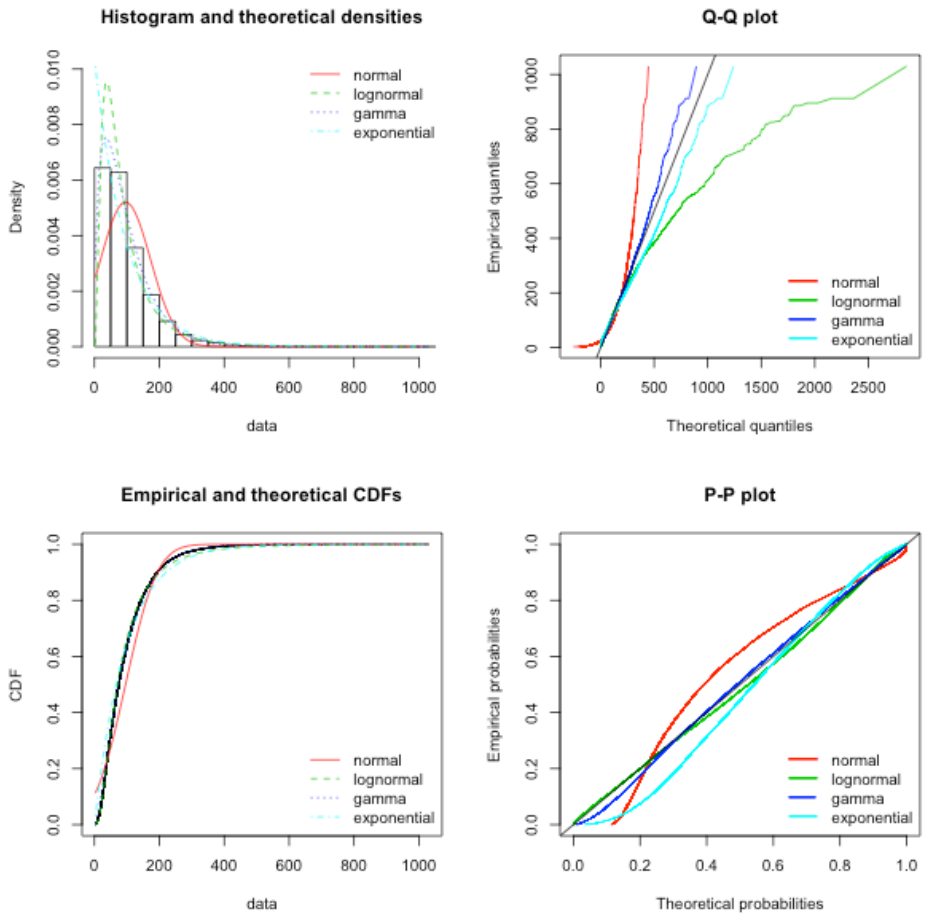


Figure C.1.: Goodness-of-fit plots of fitting the empirical path length distribution of `data series 22` comparing various common theoretical distributions.

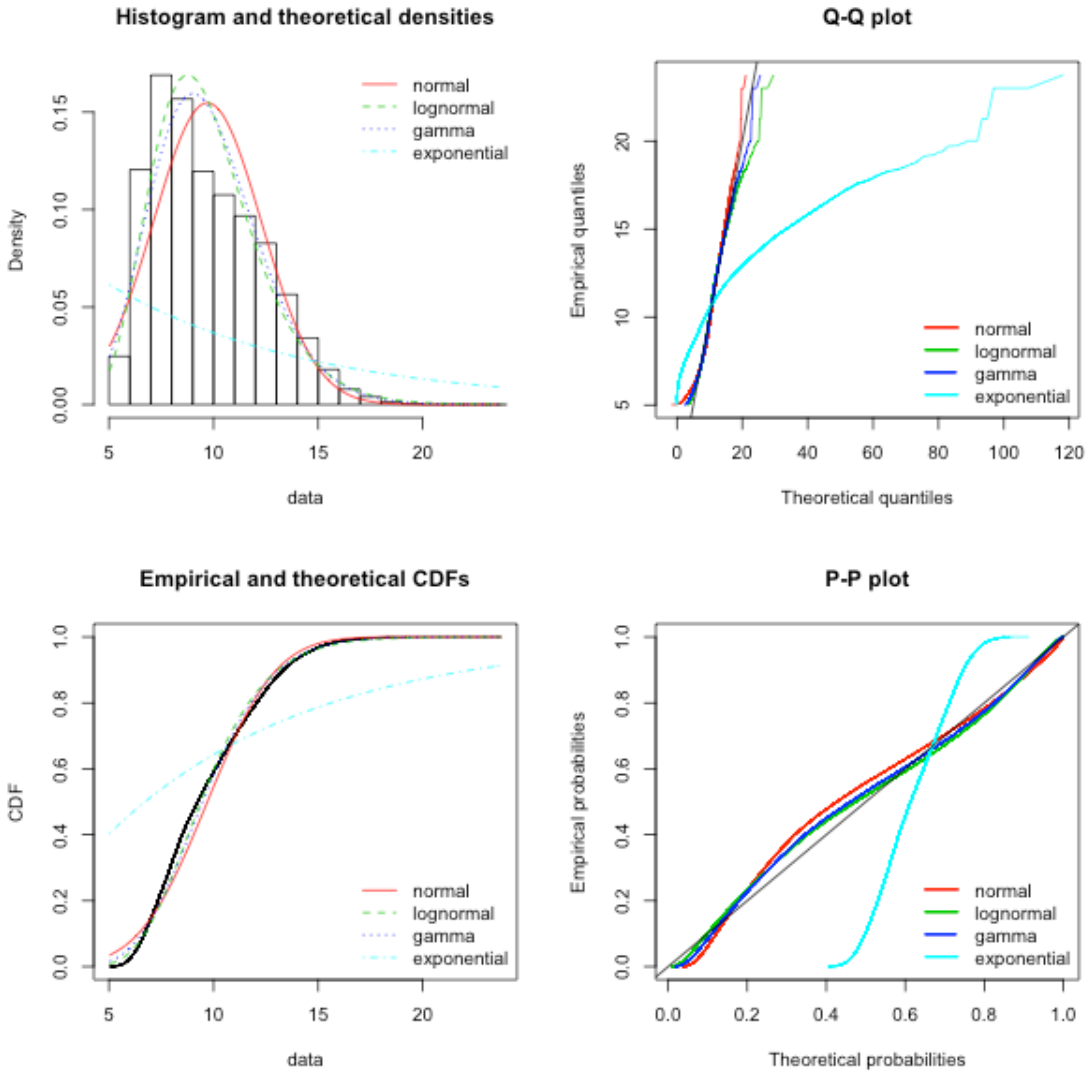


Figure C.2.: Goodness-of-fit plots of fitting the empirical average path widths distribution of `data series 22` comparing various common theoretical distributions.

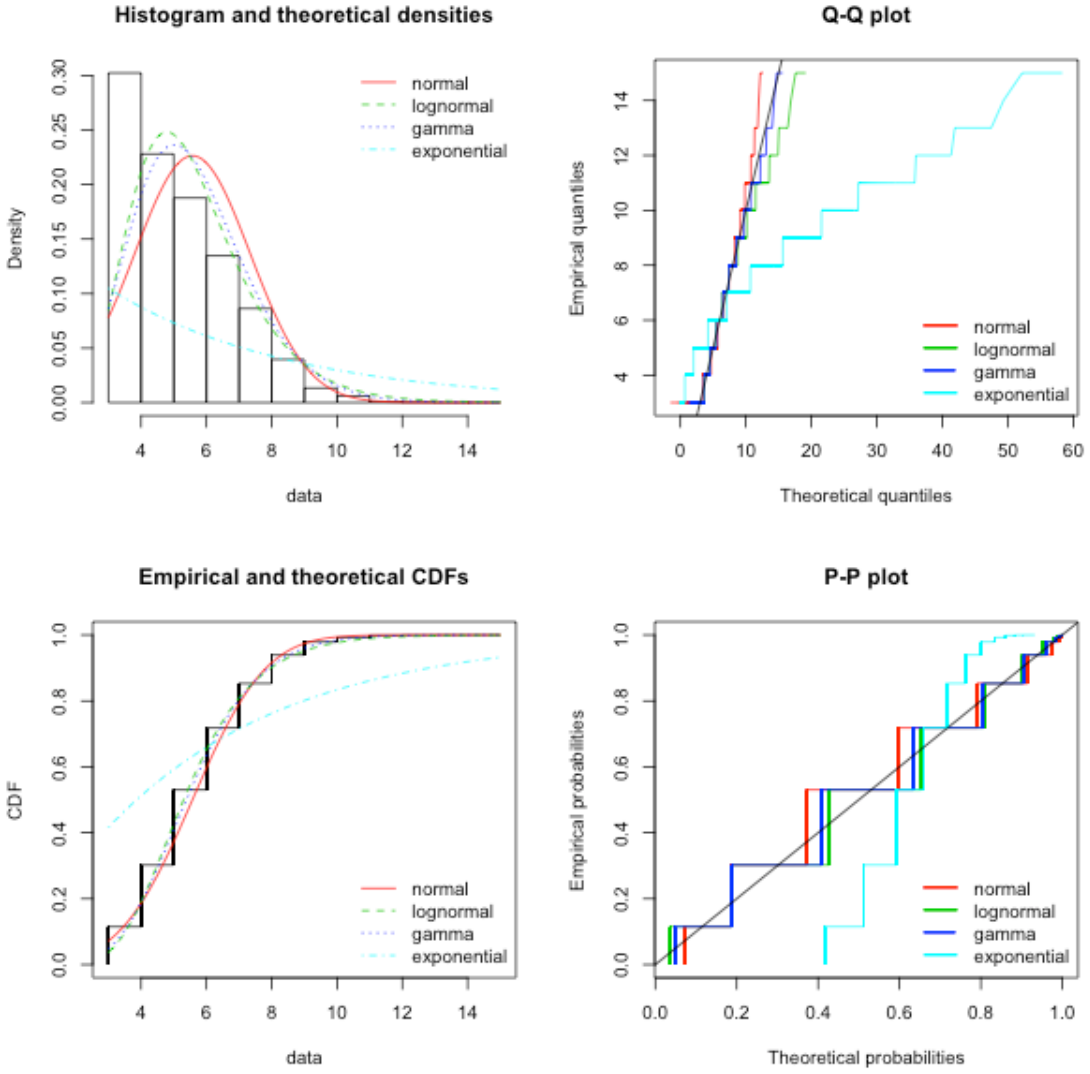


Figure C.3.: Goodness-of-fit plots of fitting the empirical face degree distribution of `data series 22` comparing various common theoretical distributions. Since the degree distribution is a discrete distribution while the fitted distributions are continuous, we obtain the observed steps. We did not look towards fitting discrete distributions.

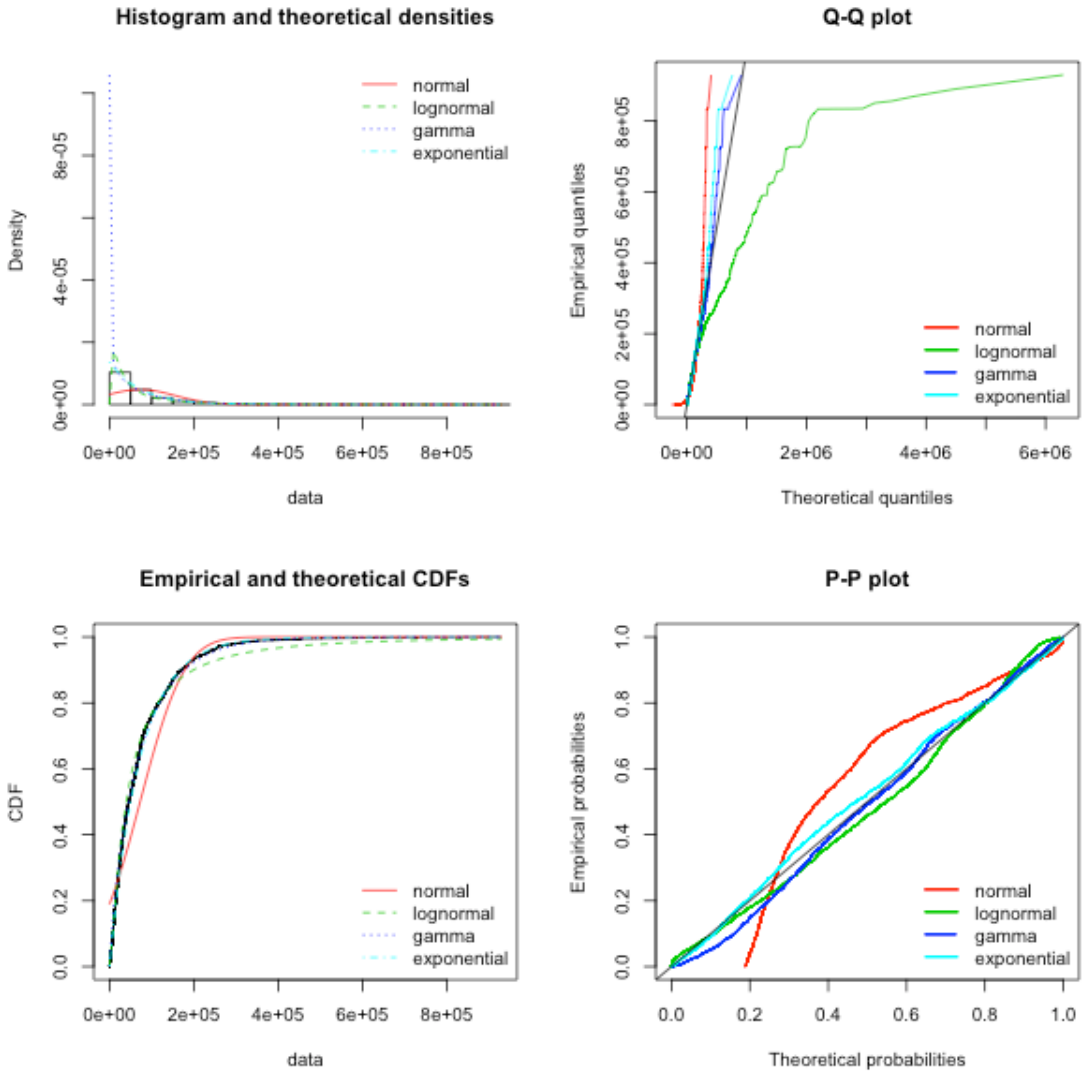


Figure C.4.: Goodness-of-fit plots of fitting the empirical face area distribution of `data series 22` comparing various common theoretical distributions.

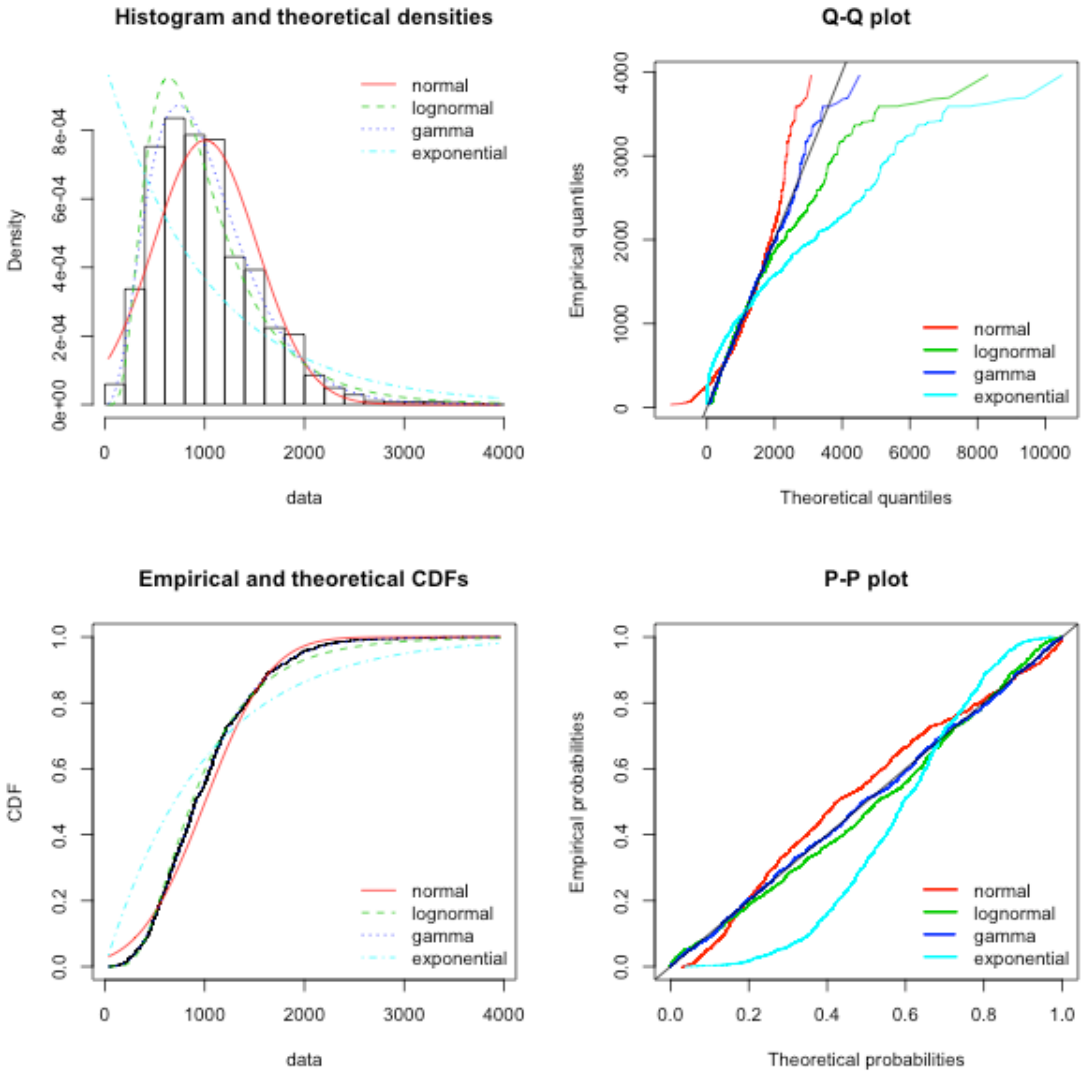


Figure C.5.: Goodness-of-fit plots of fitting the empirical face circumference distribution of `data series 22` comparing various common theoretical distributions.

C.2. Addititonal Illustrations for Section 4.3.3

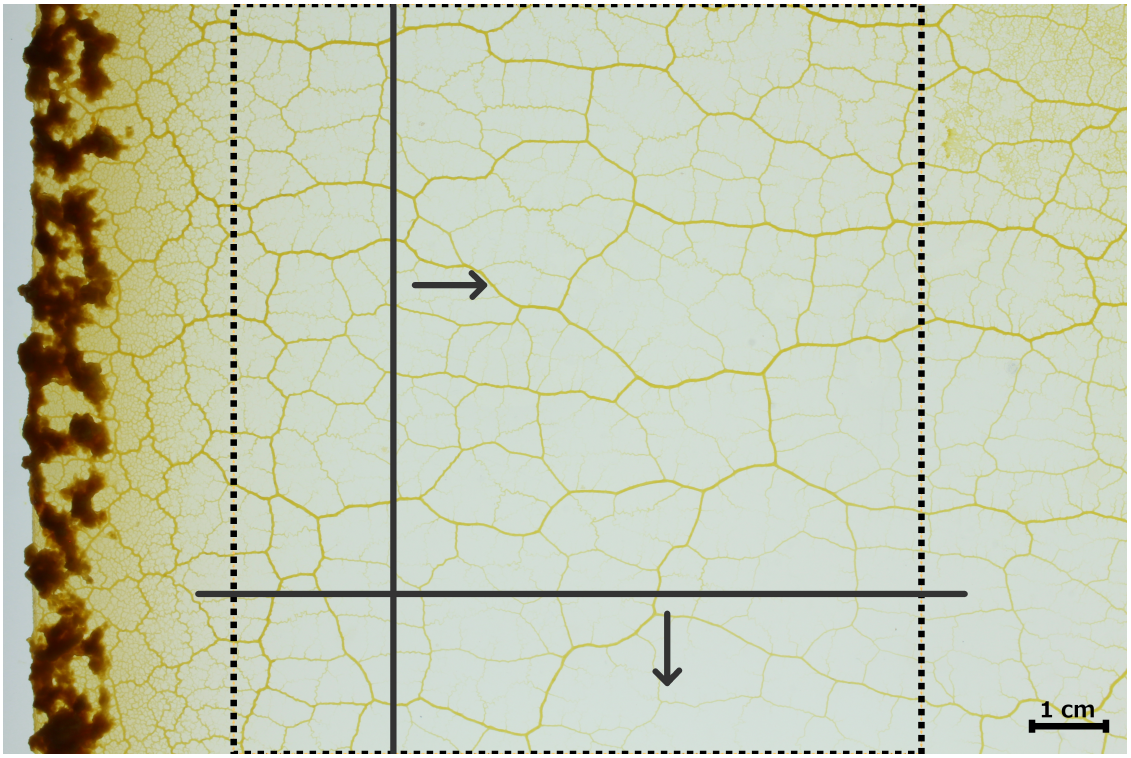


Figure C.6.: After a while the growing front has escaped the area of observation leaving behind its supporting vein network. We consider the network within a predefined region of interest (dashed rectangle). We subdivide this region of interest by defining 100 equidistant vertical and horizontal lines. The intersection of the vein network with these lines define cuts. Horizontal cuts proceed from top to bottom, i.e. perpendicular to the growth direction. Vertical cuts proceed from left to right, i.e. in growth direction. Solid lines show one cut of each type.

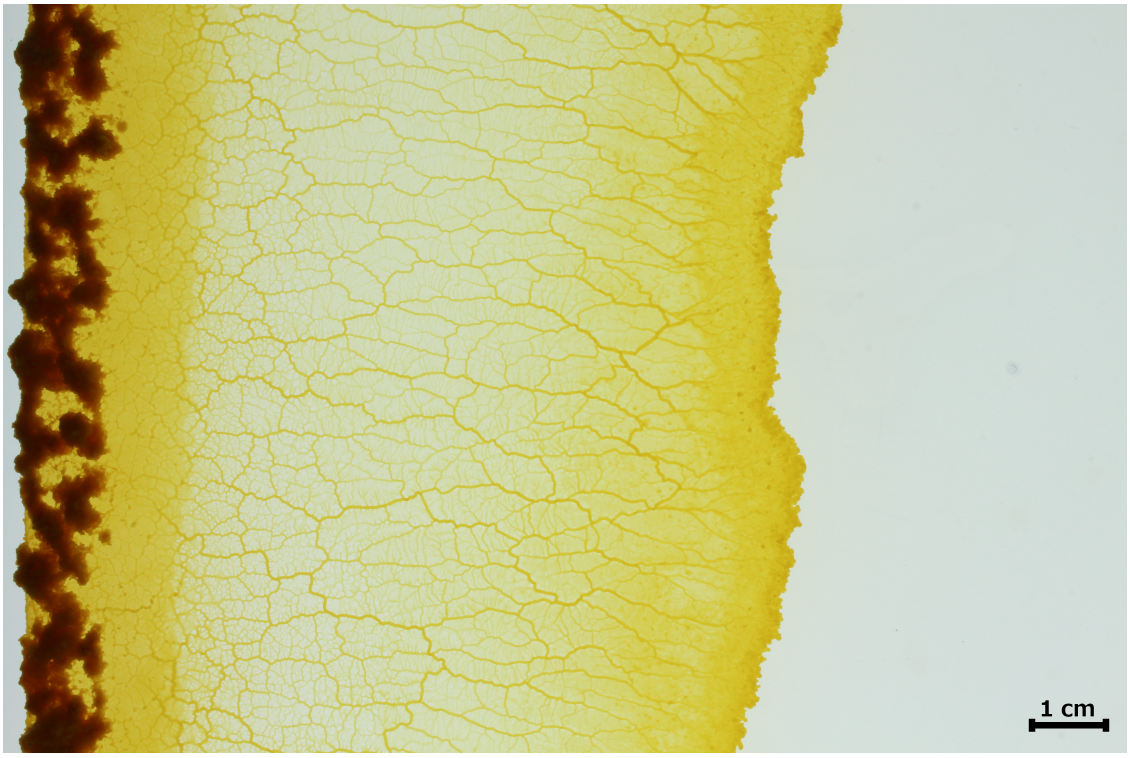


Figure C.7.: *P. polycephalum* advancing its growing front and expanding along the positive x-axis. The front itself is approximately parallel to the y-axis, i.e. perpendicular to the direction of growth.

D | Code Listings

Here we list the basic code snippets which constitute the heart of the numerical simulation of our model. They serve merely as an illustration and written documentation of the simulation setup and are not intended to provide a run-able program.[1] A complete run-able version of our simulation is available for download at the authors page.

Code Listing D.1.: Required python libraries. Here the excellent `networkx` provides graph support [74].

```
from itertools import combinations
import networkx as nx
import numpy as np
```

Code Listing D.2.: Core of the simulation. Note that the sequence of updates is different from Section 5.5. This change simplifies computation but does not affect the results.

```
def simulation(graph, epsilon = 0.01, max_iterations = 3000):

  number_of_iterations = 0

  while True:

    update_node_voltage(graph)
    update_voltage_source(graph, epsilon)
    update_edge_capacitor_voltage(graph, epsilon)

    number_of_iterations += 1

    if number_of_iterations >= max_iterations:
      print "Reached the maximum number of iterations:",
 ↪  max_iterations
      break

  return graph
```

[1] The snippets listed here fully describe the entire non-trivial part of our simulation. We omit code of minor concern such as the creation of test graphs, initialization and data logging.

Code Listing D.3.: Function updating the voltages for all nodes. Implements Equation (5.6). Using the dictionaries `R_map` and `Uc_map` in conjunction with `np.prod` and `combinations` yields a convenient way to compute the required sums and products.

```
def update_node_voltage(graph):
  # d_u["U"] --> voltage at node u
  # d_e["Uc"] --> capacitor voltage at edge e
  # d_e["R?"] --> resistance at edge e
  # d_e["Up_?"] --> current controlled voltage source voltage

  for u, d_u in graph.nodes_iter(data=True):

    R_map = {}
    Uc_map = {}
    edge_counter = 0

    if graph.degree(u) == 1:

      for _,_,d_e in graph.in_edges_iter(u, data=True):
        d_u["U"] = d_e["Uc"] - d_e["Up_2"]

      for _,_,d_e in graph.out_edges_iter(u, data=True):
        d_u["U"] = d_e["Uc"] - d_e["Up_1"]

    elif graph.degree(u) >= 2:

      for _,_,d_e in graph.in_edges_iter(u, data=True):

        R_map[edge_counter] = d_e["R2"]
        Uc_map[edge_counter] = d_e["Uc"] - d_e["Up_2"]
        edge_counter += 1

      for _,_,d_e in graph.out_edges_iter(u, data=True):

        R_map[edge_counter] = d_e["R1"]
        Uc_map[edge_counter] = d_e["Uc"] - d_e["Up_1"]
        edge_counter += 1


      R_product = np.prod( R_map.values() )
      combined_resistance = sum( [ np.prod(e) for e in
↪  combinations(R_map.values(),len(R_map.values())-1) ])
```

```
38
39          current = 0.0
40
41          for key in R_map.keys():
42
43            current += R_product * (float(Uc_map[key])/R_map[key])
44
45          voltage = current / combined_resistance
46
47          d_u["U"] = voltage
```

Code Listing D.4.: Function updating the capacitor voltages for all edges. The quantity `Ucs` is the sum over all capacitor voltages and invariant by Lemma 1 since $C_e = C$ for all $e \in G$ as determined by the initial conditions.

```
1   def update_edge_capacitor_voltage(graph, epsilon):
2     # d_e["I1"] --> current ip at edge e
3     # d_e["I2"] --> current ip* at edge e
4     # d_e["C"] --> capacity of capacitor at edge e
5     # d_e["Up_old_?"] --> previous value of d_e["Up_?"]
6
7     for n in graph.nodes_iter():
8
9       # examine out edges for node n
10      for u,v,d_e in graph.out_edges_iter(n, data=True):
11
12        C = d_e["C"]
13        r1 = d_e["R1"]
14        r2 = d_e["R2"]
15        p1 = d_e["Up_old_1"]
16        p2 = d_e["Up_old_2"]
17
18        Ux = graph.node[u]["U"]
19        Uy = graph.node[v]["U"]
20
21        a = euler_step_size/(r1*C)
22        b = euler_step_size/(r2*C)
23
24        Uc = Ux*a + Uy*b + d_e["Uc"] +
    ↪   euler_step_size/(r1*C)*((d_e["Up_old_1"] + d_e["Up_old_2"]) -
    ↪   2*d_e["Uc"])
25
26        d_e["Uc"] = Uc
27
```

```
28    # the following quantity is an invariant
29    Ucs = sum(nx.get_edge_attributes(graph,"Uc").values())
```

Code Listing D.5.: Updates for current controlled voltage source. Implements Equation (5.18), Equation (5.19) and Equation (5.20) for the l.h.s. and Equation (5.21), Equation (5.22) and Equation (5.23) for r.h.s. voltage source respectively.

```
1  def update_current_controlled_voltage_sources(graph, epsilon):
2    # d_e["beta_?"] --> property beta
3    # d_e["alpha_?"] --> property alpha
4    # d_e["Upmax"] --> cap for current controlled voltage source
5
6    def voltage_delay(Up_target, Up_current, alpha, epsilon):
7
8      c = np.exp((-1)*epsilon*alpha)
9
10     return Up_target * (1 - c) + Up_current * c
11
12     for n, d_u in graph.nodes_iter(data=True):
13
14       for u,v,d_e in graph.out_edges_iter(n,data=True):
15
16         # equation 18, here "Uc" = Uc(r-1) in the text
17         I1 = (d_u["U"] + d_e["Up_1"] - d_e["Uc"])/d_e["R1"]
18         d_e["I1"] = I1
19
20         beta_1 = d_e["beta_factor"]*d_e["R1"]
21         Up_max = d_e["Upmax"]
22         Up_min = (-1)*d_e["Upmax"]
23         alpha = d_e["alpha_factor"]/(d_e["R1"]*d_e["C"])
24
25         # equation 19
26         Up_target = max(min(Up_max, beta_1 * I1), Up_min)
27
28         Up_current = voltage_delay(Up_target, Up_current =
   ↪  d_e["Up_1"], alpha, epsilon)
29
30         d_e["Up_old_1"] = d_e["Up_1"]
31         d_e["Up_1"] = Up_current
32
33       for u,v,d_e in graph.in_edges_iter(n,data=True):
34
35         # equation 21, here "Uc" = Uc(r-1) in the text
36         I2 = (d_u["U"] + d_e["Up_2"] - d_e["Uc"])/d_e["R2"]
```

```
        d_e["I2"] = I2

        beta_2 = d_e["beta_factor"]*d_e["R2"]
        Up_max = d_e["Upmax"]
        Up_min = (-1)*d_e["Upmax"]
        alpha = d_e["alpha_factor"]/(d_e["R2"]*d_e["C"])

        # equation 22
        Up_target = max(min(Up_max, beta_2 * I2), Up_min)
        # equation 23
        Up_current = voltage_delay(Up_target, Up_current =
↪  d_e["Up_2"], alpha, epsilon)

        d_e["Up_old_2"] = d_e["Up_2"]
        d_e["Up_2"] = Up_current
```

Bibliography

[1] A. Adamatzky, V. Erokhin, M. Grube, *et al.*, "Physarum chip project: Growing computers from slime mould.", *IJUC*, vol. 8, no. 4, pp. 319–323, 2012.

[2] L. A. Adamic and B. A. Huberman, "Power-law distribution of the world wide web", *Science*, vol. 287, no. 5461, pp. 2115–2115, 2000.

[3] L. M. Adleman, "Molecular computation of solutions to combinatorial problems", *Nature*, vol. 369, p. 40, 1994.

[4] H. Aldrich, *Cell Biology of Physarum and Didymium V1: Organisms, Nucleus, and Cell Cycle*. Elsevier, 2012.

[5] K. Alim, G. Amselem, F. Peaudecerf, *et al.*, "Random network peristalsis in Physarum polycephalum organizes fluid flows across an individual", *Proceedings of the National Academy of Sciences*, vol. 110, no. 33, pp. 13 306–13 311, 2013.

[6] U. Alon, M. G. Surette, N. Barkai, *et al.*, "Robustness in bacterial chemotaxis", *Nature*, vol. 397, no. 6715, pp. 168–171, 1999.

[7] L. A. N. Amaral and J. M. Ottino, "Complex networks", *The European Physical Journal B*, vol. 38, no. 2, pp. 147–162, 2004.

[8] M. Aono and M. Hara, "Amoeba-based nonequilibrium neurocomputer utilizing fluctuations and instability", in *Proceedings of the 6th International Conference on Unconventional Computation*, Springer International Publishing, 2007, pp. 41–54.

[9] ——, "Spontaneous deadlock breaking on amoeba-based neurocomputer", *Biosystems*, vol. 91, no. 1, pp. 83–93, 2008.

[10] M. Aono, M. Hara, and K. Aihara, "Amoeba-based neurocomputing with chaotic dynamics", *Communications of the ACM*, vol. 50, no. 9, pp. 69–72, 2007.

[11] M. Aono, Y. Hirata, M. Hara, *et al.*, "Amoeba-based chaotic neurocomputing: Combinatorial optimization by coupled biological oscillators", *New Generation Computing*, vol. 27, no. 2, pp. 129–157, 2009.

[12] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, *et al.*, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer International Publishing, 1999.

[13] T. Back, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. IOP Publishing Ltd., 1997.

[14] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks", *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[15] A. Barrat, M. Barthélemy, and A. Vespignani, "Weighted evolving networks: Coupling topology and weight dynamics", *Physical review letters*, vol. 92, no. 22, p. 228 701, 2004.

[16] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks", 2009.

[17] V. Batagelj and A. Mrvar, "Pajek-program for large network analysis", *Connections*, vol. 21, no. 2, pp. 47–57, 1998.

[18] W. Baumgarten and M. J. Hauser, "Detection, extraction, and analysis of the vein network", *Journal of Computational Interdisciplinary Sciences*, vol. 1, no. 3, pp. 241–249, 2010.

[19] ——, "Functional organization of the vascular network of Physarum polycephalum", *Physical Biology*, vol. 10, no. 2, p. 026 003, 2013.

[20] W. Baumgarten and M. Hauser, "Computational algorithms for extraction and analysis of two-dimensional transportation networks", *Journal of Computational Interdisciplinary Sciences*, vol. 3, pp. 107–16, 2012.

[21] W. Baumgarten, J. Jones, and M. J. Hauser, "Network coarsening dynamics in a plasmodial slime mould: Modelling and experiments", *Acta Physica Polonica B*, vol. 46, no. 6, pp. 1201–1218, 2015.

[22] W. Baumgarten, T. Ueda, and M. J. Hauser, "Plasmodial vein networks of the slime mold Physarum polycephalum form regular graphs", *Physical Review E*, vol. 82, no. 4, p. 046 113, 2010.

[23] L. Becchetti, V. Bonifaci, M. Dirnberger, *et al.*, "Physarum can compute shortest paths: Convergence proofs and complexity bounds", in *International Colloquium on Automata, Languages, and Programming*, Springer International Publishing, 2013, pp. 472–483.

[24] A. M. Becker and R. M. Ziff, "Percolation thresholds on two-dimensional voronoi networks and delaunay triangulations", *Physical Review E*, vol. 80, no. 4, p. 041 101, 2009.

[25] M. A. Bedau, J. S. McCaskill, N. H. Packard, *et al.*, "Open problems in artificial life", *Artificial life*, vol. 6, no. 4, pp. 363–376, 2000.

[26] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1995.

[27] E. Boa, "Ainsworth and Bisby's Dictionary of the Fungi", *Plant Pathology*, vol. 47, no. 4, pp. 541–541, 1998.

[28] S. Boccaletti, V. Latora, Y. Moreno, *et al.*, "Complex networks: Structure and dynamics", *Physics reports*, vol. 424, no. 4, pp. 175–308, 2006.

[29] M. A. Boden, *The Philosophy of Artificial Life*. Oxford University Press, Inc., 1996.

[30] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., 1999.

[31] ——, "Inspiration for optimization from social insect behaviour", *Nature*, vol. 406, no. 6791, pp. 39–42, 2000.

[32] E. Bonabeau and G. Théraulaz, "Swarm smarts", *Scientific American*, vol. 282, no. 3, pp. 72–79, 2000.

[33] D. Boneh, C. Dimworth, and R. J. Lipton, "Breaking DBS using a molecular computer", *DNA Based Computing*, vol. 27, p. 37, 1996.

[34] D. Boneh, C. Dunworth, R. J. Lipton, *et al.*, "On the computational power of DNA", *Discrete Applied Mathematics*, vol. 71, no. 1, pp. 79–94, 1996.

[35] V. Bonifaci, "Physarum can compute shortest paths: A short proof", *Information Processing Letters*, vol. 113, no. 1-2, pp. 4–7, 2013.

[36] V. Bonifaci, K. Mehlhorn, and G. Varma, "Physarum can compute shortest paths", *Journal of Theoretical Biology*, vol. 309, pp. 121–133, 2012.

[37] G. Bradski, "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

[38] D. Brandner and G. Withers, *The Cell Image Library*, 2010.

[39] J. Buhl, J. Gautrais, N. Reeves, *et al.*, "Topological patterns in street networks of self-organized urban settlements", *The European Physical Journal B*, vol. 49, no. 4, pp. 513–522, 2006.

[40] D. S. Callaway, M. E. Newman, S. H. Strogatz, *et al.*, "Network robustness and fragility: Percolation on random graphs", *Physical review letters*, vol. 85, no. 25, p. 5468, 2000.

[41] J. M. Carlson and J. Doyle, "Complexity and robustness", *Proceedings of the National Academy of Sciences*, vol. 99, no. 1, pp. 2538–2545, 2002.

[42] L. N. de Castro, "Natural computing", in *Encyclopedia of Information Science and Technology*, IGI Global, 2005, pp. 2080–2084.

[43] ——, "Fundamentals of natural computing: An overview", *Physics of Life Reviews*, vol. 4, no. 1, pp. 1–36, 2007.

[44] D. Chai, W. Forstner, and F. Lafarge, "Recovering line-networks in images by junction-point processes", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2013, pp. 1894–1901.

[45] R. Chellappa and B. Manjunath, "Texture classification and segmentation: Tribulations, triumphs and tributes", in *Foundations of Image Understanding*, vol. 628, Springer International Publishing, 2001, pp. 219–240.

[46] CPLEX, *V12. 1: User's manual for cplex*, 2009.

[47] A. C. Cullen and H. C. Frey, *Probabilistic techniques in exposure assessment: A handbook for dealing with variability and uncertainty in models and inputs*. Springer Science & Business Media, 1999.

[48] L. N. De Castro, *Fundamentals of natural computing: Basic concepts, algorithms, and applications*. CRC Press, 2006.

[49] M. T. Dehkordi, S. Sadri, and A. Doosthoseini, "A review of coronary vessel segmentation algorithms", *Journal of medical signals and sensors*, vol. 1, no. 1, p. 49, 2011.

[50] M. L. Delignette-Muller, C. Dutang, *et al.*, "Fitdistrplus: An R package for fitting distributions", *Journal of Statistical Software*, vol. 64, no. 4, pp. 1–34, 2015.

[51] M. Dirnberger, T. Kehl, T. Mehlhorn, *et al.*, "Towards an open online repository of P. polycephalum networks and their corresponding graph representations", in *The first international workshop on physarum transport networks*, ACM, 2016.

[52] M. Dirnberger, T. Kehl, and A. Neumann, "Nefi: Network extraction from images", *Scientific reports*, vol. 5, 2015.

[53] M. Dirnberger and K. Mehlhorn, "Characterizing networks formed by P. polycephalum", *Journal of Physics D: Applied Physics, submitted*, 2016.

[54] M. Dirnberger, K. Mehlhorn, and T. Mehlhorn, "Introducing the slime mold graph repository", *Journal of Physics D: Applied Physics, submitted*, 2016.

[55] Z. V. Djordjevic, H. E. Stanley, and A. Margolina, "Site percolation threshold for honeycomb and square lattices", *Journal of Physics A: Mathematical and Theoretical*, vol. 15, no. 8, p. L405, 1982.

[56] W. F. Dove and H. P. Rusch, *Growth and Differentiation in Physarum polycephalum*. Princeton University Press, 1980.

[57] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multicommodity flow problems", in *16th Annual Symposium on Foundations of Computer Science*, 1975, pp. 184–193.

[58] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology", in *SIGCOMM computer communication review*, ACM, vol. 29, 1999, pp. 251–262.

[59] L. Fausett, Ed., *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., 1994.

[60] A. Fessel, C. Oettmeier, and H.-G. Döbereiner, "Structuring precedes extension in percolating Physarum polycephalum networks", *Nano Communication Networks*, vol. 6, no. 3, pp. 87–95, 2015.

[61] A. Fessel, C. Oettmeier, E. Bernitt, *et al.*, "Physarum polycephalum percolation as a paradigm for topological phase transitions in transportation networks", *Physical review letters*, vol. 109, no. 7, p. 078 103, 2012.

[62] A. Fessel, C. Oettmeier, and H.-G. Döbereiner, "An analytical model for percolation in small link degree transportation networks", in *Proceedings of the 8th International Conference on Bioinspired Information and Communications Technologies*, 2014, pp. 81–86.

[63] O. Frank, "Die Grundform des arteriellen Pulses", *Z Biol.*, vol. 37, no. 483-526, p. 19, 1899.

[64] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L 2 theory", *Probability theory and related fields*, vol. 57, no. 4, pp. 453–476, 1981.

[65] M. Fricker, L. Boddy, and D. Bebber, "Network organisation of mycelial fungi", in *Biology of the fungal cell*, Springer International Publishing, 2007, pp. 309–330.

[66] Y. Fukuyama, "Fundamentals of particle swarm optimization techniques", *Modern heuristic optimization techniques: Theory and applications to power systems*, pp. 71–87, 2008.

[67] L. K. Grover, "A fast quantum mechanical algorithm for database search", in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ACM, 1996, pp. 212–219.

[68] M. Grube, "Physarum, quo vadis?", in *Advances in Physarum Machines*, Springer International Publishing, 2016, pp. 23–35.

[69] R. Guimera and L. A. N. Amaral, "Modeling the world-wide airport network", *The European Physical Journal B*, vol. 38, no. 2, pp. 381–385, 2004.

[70] Y.-P. Gunji, T. Shirakawa, T. Niizato, *et al.*, "Minimal model of a cell connecting amoebic motion and adaptive transport networks", *Journal of Theoretical Biology*, vol. 253, no. 4, pp. 659–667, 2008.

[71] Y.-P. Gunji, T. Shirakawa, T. Niizato, *et al.*, "An adaptive and robust biological network based on the vacant-particle transportation model", *Journal of Theoretical Biology*, vol. 272, no. 1, pp. 187–200, 2011.

[72] Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms", *Communications of the ACM*, vol. 32, no. 3, pp. 359–373, 1989.

[73] Gurobi, *Gurobi optimizer reference manual*, 2012.

[74] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 2008, pp. 11–15.

[75] S. Hales, "Statical Essays Haemostaticks vol. 2", *London W*, vol. 2, p. 230, 1733.

[76] V. Hardung, "Propagation of pulse waves in viscoelastic tubings", *Handbook of physiology*, vol. 1, pp. 107–135, 1962.

[77] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1998.

[78] L. Heaton, B. Obara, V. Grau, *et al.*, "Analysis of fungal networks", *Fungal Biology Reviews*, vol. 26, no. 1, pp. 12–29, 2012.

[79] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*. Holden-Day, Inc., 1986.

[80] A. Hinde and R. Miles, "Monte carlo estimates of the distributions of the random polygons of the voronoi tessellation with respect to a poisson process", *Journal of Statistical Computation and Simulation*, vol. 10, no. 3-4, pp. 205–223, 1980.

[81] F. L. Howard, "The life history of Physarum polycephalum", *American Journal of Botany*, vol. 18, no. 2, pp. 116–133, 1931.

[82] B. A. Huberman and L. A. Adamic, "Internet: Growth dynamics of the world-wide web", *Nature*, vol. 401, no. 6749, pp. 131–131, 1999.

[83] K. Ito, A. Johansson, T. Nakagaki, *et al.*, "Convergence properties for the physarum solver", *Arxiv:1101.5249*, 2011.

[84] M. Ito, R. Okamoto, and A. Takamatsu, "Characterization of adaptation by morphology in a planar biological network of plasmodial slime mold", *Journal of the Physical Society of Japan*, vol. 80, no. 7, p. 074 801, 2011.

[85] B. Jiang and C. Claramunt, "Topological analysis of urban street networks", *Environment and Planning B: Planning and design*, vol. 31, no. 1, pp. 151–162, 2004.

[86] A. Johannson and J. Zou, "A slime mold solver for linear programming problems", in *How the World Computes: Turing Centenary Conference and 8th Conference on Computability in Europe*. Springer International Publishing, 2012, pp. 344–354.

[87] H. J. Johnson, M. McCormick, L. Ibáñez, *et al.*, *The ITK Software Guide*, Kitware, Inc., 2013.

[88] E. Jones, T. Oliphant, P. Peterson, *et al.*, *Open source scientific tools for Python*, 2001.

[89] J. Jones, "Characteristics of pattern formation and evolution in approximations of physarum transport networks", *Artificial life*, vol. 16, no. 2, pp. 127–153, 2010.

[90] ——, "Influences on the formation and evolution of Physarum polycephalum inspired emergent transport networks", *Natural computing*, vol. 10, no. 4, pp. 1345–1369, 2011.

[91] ——, "Applications of multi-agent slime mould computing", *International Journal of Parallel, Emergent and Distributed Systems*, pp. 1–30, 2015.

[92] ——, "Multi-agent slime mould computing: Mechanisms, applications and advances", in *Advances in Physarum Machines*, Springer International Publishing, 2016, pp. 423–463.

[93] J. Jones and A. Adamatzky, "Computation of the travelling salesman problem by a shrinking blob", *Natural computing*, vol. 13, no. 1, pp. 1–16, 2014.

[94] ——, "Material approximation of data smoothing and spline curves inspired by slime mould", *Bioinspiration & biomimetics*, vol. 9, no. 3, p. 036 016, 2014.

[95] ——, "Slime mould inspired generalised voronoi diagrams with repulsive fields", *Arxiv:1503.06973*, 2015.

[96] N. Kamiya, "Motive Force Responsible for the Protoplasmic Streaming", in *Protoplasmic Streaming*, Springer International Publishing, 1959, pp. 38–52.

[97] N. Kamiya and K. Kuroda, "Studies on the velocity distribution of the protoplasmic streaming in the myxomycete plasmodium", *Protoplasma*, vol. 49, no. 1, pp. 1–4, 1958.

[98] T. Karagiannis, M. Molle, and M. Faloutsos, "Long-range dependence ten years of internet traffic modeling", *Internet Computing*, vol. 8, no. 5, pp. 57–64, 2004.

[99] A. Karrenbauer and D. Wöll, "Blinking molecule tracking", in *Proceedings of the 12th international symposium on experimental algorithms*, Springer International Publishing, 2013, pp. 308–319.

[100] E. Katifori, G. J. Szöllősi, and M. O. Magnasco, "Damage and fluctuations induce loops in optimal transport networks", *Physical review letters*, vol. 104, no. 4, p. 048 704, 2010.

[101] R. Kicinger, T. Arciszewski, and K. De Jong, "Evolutionary computation and structural design: A survey of the state-of-the-art", *Computers & Structures*, vol. 83, no. 23, pp. 1943–1978, 2005.

[102] K. Koizumi, M. Sugiyama, and H. Fukuda, "A series of novel mutants of Arabidopsis thaliana that are defective in the formation of continuous vascular network: Calling the auxin signal flow canalization hypothesis into question", *Development*, vol. 127, no. 15, pp. 3197–3204, 2000.

[103] M. Krause, R. M. Alles, B. Burgeth, *et al.*, "Fast retinal vessel analysis", *Journal of Real-Time Image Processing*, pp. 1–10, 2013.

[104] G. Landes, "Einige Untersuchungen an elektrischen Analogieschaltungen zum Kreislaufsystem", *Z. Biol.*, vol. 101, pp. 418–429, 1943.

[105] C. G. Langton, *Artificial Life: An Overview*. MIT Press, 1995.

[106] J. Leskovec and R. Sosič, *SNAP: A general purpose network analysis and graph mining library in C++*, http://snap.stanford.edu/snap, 2014.

[107] S. Loscalzo and L. Yu, "Social network analysis: Tasks and tools", in *Social computing, behavioral modeling, and prediction*, Springer International Publishing, 2008, pp. 151–159.

[108] W. Marwan, "Amoeba-inspired network design", *Science*, vol. 327, no. 5964, pp. 419–420, 2010.

[109] R. Mayne, "Biology of the Physarum polycephalum plasmodium: Preliminaries for unconventional computing", in *Advances in Physarum Machines: Sensing and Computing with Slime Mould*. Springer International Publishing, 2016, pp. 3–22.

[110] K. Mehlhorn and S. Näher, "LEDA: A Platform for Combinatorial and Geometric Computing", *Commun. ACM*, vol. 38, no. 1, pp. 96–102, 1995.

[111] F. Meyer, "Un algorithme optimal pour la ligne de partage des eaux", *Dans 8me congrès de reconnaissance des formes et intelligence artificielle*, vol. 2, pp. 847–857, 1991.

[112] R. Milo, S. Shen-Orr, S. Itzkovitz, *et al.*, "Network motifs: Simple building blocks of complex networks", *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[113] T. Miyaji, I. Ohnishi, *et al.*, "Mathematical analysis to an adaptive network of the plasmodium system", *Hokkaido Mathematical Journal*, vol. 36, no. 2, pp. 445–465, 2007.

[114] T. Miyaji and I. Ohnishi, "Physarum can solve the shortest path problem on riemannian surface mathematically rigorously", *International Journal of Pure and Applied Mathematics*, vol. 47, no. 3, pp. 353–369, 2008.

[115] Y. Miyake, S. Tabata, H. Murakami, *et al.*, "Environment-dependent self-organization of positional information field in chemotaxis of Physarum plasmodium", *Journal of Theoretical Biology*, vol. 178, no. 4, pp. 341–353, 1996.

[116] J. M. Montoya, S. L. Pimm, and R. V. Solé, "Ecological networks and their fragility", *Nature*, vol. 442, no. 7100, pp. 259–264, 2006.

[117] K. Nagel, D. E. Wolf, P. Wagner, *et al.*, "Two-lane traffic rules for cellular automata: A systematic approach", *Physical Review E*, vol. 58, no. 2, p. 1425, 1998.

[118] T. Nakagaki, M. Iima, T. Ueda, *et al.*, "Minimum-risk path finding by an adaptive amoebal network", *Physical review letters*, vol. 99, p. 068 104, 6 2007.

[119] ——, "Minimum-risk path finding by an adaptive amoebal network", *Physical review letters*, vol. 99, no. 6, p. 068 104, 2007.

[120] T. Nakagaki, R. Kobayashi, Y. Nishiura, *et al.*, "Obtaining multiple separate food sources: Behavioural intelligence in the Physarum plasmodium", *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 271, no. 1554, pp. 2305–2310, 2004.

[121] T. Nakagaki and T. Ueda, "Phase switching of oscillatory contraction in relation to the regulation of amoeboid behavior by the plasmodium of Physarum polycephalum", *Journal of Theoretical Biology*, vol. 179, no. 3, pp. 261–267, 1996.

[122] T. Nakagaki, S. Umemura, Y. Kakiuchi, *et al.*, "Action spectrum for sporulation and photoavoidance in the plasmodium of Physarum polycephalum, as modified differentially by temperature and starvation", *Photochemistry and photobiology*, vol. 64, no. 5, pp. 859–862, 1996.

[123] T. Nakagaki, H. Yamada, and M. Hara, "Smart network solutions in an amoeboid organism", *Biophysical Chemistry*, vol. 107, no. 1, pp. 1–5, 2004.

[124] T. Nakagaki, H. Yamada, and Á. Tóth, "Intelligence: Maze-solving by an amoeboid organism", *Nature*, vol. 407, no. 6803, pp. 470–470, 2000.

[125] T. Nakagaki, H. Yamada, and T. Ueda, "Interaction between cell shape and contraction pattern in the Physarum plasmodium", *Biophysical chemistry*, vol. 84, no. 3, pp. 195–204, 2000.

[126] M. Negri, P. Gamba, G. Lisini, *et al.*, "Junction-aware extraction and regularization of urban road networks in high-resolution sar images", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 10, pp. 2962–2971, 2006.

[127] M. E. Newman, "The structure and function of complex networks", *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.

[128] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011.

[129] W. Nowotny, *Myxomyceten (Schleimpilze) und Mycetozoa (Pilztiere) - Lebensformen zwischen Pflanze und Tier*. 2000.

[130] V. Ntinas, I. Vourkas, G. C. Sirakoulis, *et al.*, "Oscillation-based slime mould electronic circuit model for maze-solving computations", *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2016.

[131] B. Obara, M. D. Fricker, and V. Graua, "Contrast independent detection of branching points in network-like structures", in *SPIE Medical Imaging*, International Society for Optics and Photonics, 2012, pp. 83141L–83141L.

[132] B. Obara, V. Grau, and M. D. Fricker, "A bioimage informatics approach to automatically extract complex fungal networks", *Bioinformatics*, vol. 28, no. 18, pp. 2374–2381, 2012.

[133] C. Oettmeier, J. Lee, and H.-G. Döbereiner, "Hydrodynamic mechanism of information processing in Physarum polycephalum", in *Physics of Physarum polycephalum and Other Slime Molds - Joint Focus Session (BP/DY) organized by Hans-Günther Döbereiner*, 2017.

[134] J. S. Oliver, "Computation with DNA: matrix multiplication", *DNA Based Computers II. Proc of DIMACS*, vol. 44, pp. 113–22, 1998.

[135] M. S. Olufsen, A. Nadim, *et al.*, "On deriving lumped models for blood flow and pressure in the systemic arteries", *Math. Biosci. Eng.*, vol. 1, no. 1, pp. 61–80, 2004.

[136] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[137] N. R. Pal and S. K. Pal, "A review on image segmentation techniques", *Pattern Recognition*, vol. 26, no. 9, pp. 1277–1294, 1993.

[138] L. de Pater and J. van den Berg, "An electrical analogue of the entire human circulatory system", *Medical electronics and biological engineering*, vol. 2, no. 2, pp. 161–166, 1964.

[139] G. Păun, "Computing with membranes", *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[140] Y. V. Pershin, S. La Fontaine, and M. Di Ventra, "Memristive model of amoeba learning", *Physical Review E*, vol. 80, no. 2, p. 021 926, 2009.

[141] D. L. Pham, C. Xu, and J. L. Prince, "Current methods in medical image segmentation 1", *Annual review of biomedical engineering*, vol. 2, no. 1, pp. 315–337, 2000.

[142] S. L. Pimm, J. H. Lawton, and J. E. Cohen, "Food web patterns and their consequences", *Nature*, vol. 350, no. 6320, pp. 669–674, 1991.

[143] R Core team and others, *R: A language and environment for statistical computing*, 2013.

[144] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model", in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, 1987, pp. 25–34.

[145] M. Rohden, A. Sorge, M. Timme, *et al.*, "Self-organized synchronization in decentralized power grids", *Physical review letters*, vol. 109, no. 6, p. 064 101, 2012.

[146] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut - interactive foreground extraction using iterated graph cuts", *ACM Transactions on Graphics (SIGGRAPH)*, pp. 309–314, 2004.

[147] D. H. Rothman and S. Zaleski, *Lattice-gas cellular automata: Simple models of complex hydrodynamics*. Cambridge University Press, 2004, vol. 5.

[148] A. Roth-Nebelsick, D. Uhl, V. Mosbrugger, *et al.*, "Evolution and function of leaf venation architecture: A review", *Annals of Botany*, vol. 87, no. 5, pp. 553–566, 2001.

[149] E. Samuel, C. de la Higuera, and J.-C. Janodet, "Extracting plane graphs from images", in *Structural, Syntactic, and Statistical Pattern Recognition*, vol. 6218, Springer International Publishing, 2010, pp. 233–243.

[150] H. Sauer, *Developmental biology of Physarum*. CUP Archive, 1982, vol. 11.

[151] H. W. Sauer, "Introduction to Physarum", in *The Molecular Biology of Physarum polycephalum*. Springer International Publishing, 1986, pp. 1–17.

[152] T. Schön, M. Stetter, and E. W. Lang, "Structure learning for bayesian networks using the Physarum solver", in *11th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, vol. 1, 2012, pp. 488–493.

[153] H.-P. Schwefel, *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc., 1993.

[154] P. Sen, S. Dasgupta, A. Chatterjee, *et al.*, "Small-world properties of the indian railway network", *Physical Review E*, vol. 67, no. 3, p. 036 106, 2003.

[155] T. Shirakawa and Y.-P. Gunji, "Emergence of morphological order in the network formation of Physarum polycephalum", *Biophysical chemistry*, vol. 128, no. 2, pp. 253–260, 2007.

[156] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[157] J. Sienkiewicz and J. A. Hołyst, "Statistical analysis of 22 public transport networks in Poland", *Physical Review E*, vol. 72, no. 4, p. 046 127, 2005.

[158] D. Smith and R. Saldana, "Model of the Ca2+ oscillator for shuttle streaming in Physarum polycephalum", *Biophysical Journal*, vol. 61, no. 2, pp. 368–380, 1992.

[159] C. Sommer, C. Straehle, U. Köthe, *et al.*, "Ilastik: Interactive learning and segmentation toolkit", in *International symposium on biomedical imaging: From nano to macro*, IEEE, 2011, pp. 230–233.

[160] W. M. Spears, K. A. De Jong, T. Bäck, *et al.*, "An overview of evolutionary computation", in *European Conference on Machine Learning*, Springer International Publishing, 1993, pp. 442–459.

[161] A. Stefanovska, "Physics of the human cardiovascular system", *Contemporary Physics*, vol. 40, no. 1, pp. 31–55, 1999.

[162] S. L. S. Stephenson, L. Steven, *et al.*, *Myxomycetes; a handbook of slime molds*. 1994.

[163] P. A. Stewart and B. T. Stewart, "Protoplasmic movement in slime mold plasmodia: The diffusion drag force hypothesis", *Experimental cell research*, vol. 17, no. 1, pp. 44–58, 1959.

[164] S. H. Strogatz, "Exploring complex networks", *Nature*, vol. 410, no. 6825, pp. 268–276, 2001.

[165] K. Takahashi, G. Uchida, Z.-S. Hu, *et al.*, "Entrainment of the self-sustained oscillation in a Physarum polycephalum strand as a one-dimensionally coupled oscillator system", *Journal of Theoretical Biology*, vol. 184, no. 2, pp. 105–110, 1997.

[166] A. Takamatsu, T. Fujii, and I. Endo, "Time delay effect in a living coupled oscillator system with the plasmodium of *Physarum polycephalum*", *Physical review letters*, vol. 85, pp. 2026–2029, 9 2000.

[167] A. Takamatsu, E. Takaba, and G. Takizawa, "Environment-dependent morphology in plasmodium of true slime mold Physarum polycephalum and a network growth model", *Journal of Theoretical Biology*, vol. 256, no. 1, pp. 29–44, 2009.

[168] A. Takamatsu, R. Tanaka, H. Yamada, *et al.*, "Spatiotemporal symmetry in rings of coupled biological oscillators of physarum plasmodial slime mold", *Physical review letters*, vol. 87, no. 7, p. 078 102, 2001.

[169] M. Tanemura, "Statistical distributions of poisson voronoi cells in two and three dimensions", *Forma*, vol. 18, no. 4, pp. 221–247, 2003.

[170] F. Tardieu, "Virtual plants: Modelling as a tool for the genomics of tolerance to water deficit", *Trends in plant science*, vol. 8, no. 1, pp. 9–14, 2003.

[171] C. Taylor and D. Jefferson, "Artificial Life As a Tool for Biological Inquiry", *Artificial Life*, vol. 1, no. 1-2, pp. 1–13, 1993.

[172] V. Teplov, Y. M. Romanovsky, and O. Latushkin, "A continuum model of contraction waves and protoplasm streaming in strands of Physarum plasmodium", *Biosystems*, vol. 24, no. 4, pp. 269–289, 1991.

[173] A. Tero, R. Kobayashi, and T. Nakagaki, "A coupled-oscillator model with a conservation law for the rhythmic amoeboid movements of plasmodial slime molds", *Physica D: Nonlinear Phenomena*, vol. 205, no. 1, pp. 125–135, 2005.

[174] ——, "Physarum solver: A biologically inspired method of road-network navigation", *Physica A: Statistical Mechanics and its Applications*, vol. 363, no. 1, pp. 115–119, 2006.

[175] ——, "A mathematical model for adaptive transport network in path finding by true slime mold", *Journal of Theoretical Biology*, vol. 244, no. 4, pp. 553–564, 2007.

[176] A. Tero, S. Takagi, T. Saigusa, *et al.*, "Rules for biologically inspired adaptive network design", *Science*, vol. 327, no. 5964, pp. 439–442, 2010.

[177] S. Tsuda, M. Aono, and Y.-P. Gunji, "Robust and emergent physarum logical-computing", *Biosystems*, vol. 73, no. 1, pp. 45–55, 2004.

[178] S. Tsuda, K.-P. Zauner, and Y.-P. Gunji, "Robot control with biological cells", *Biosystems*, vol. 87, no. 2–3, pp. 215–223, 2007.

[179] T. Ueda, "An intelligent slime mold: A self-organizing system of cell shape and information", *Lecture Notes in Complex System*, vol. 3, pp. 221–253, 2005.

[180] G. Watt, M. Whalley, J. Bentham, *et al.*, *The HepData Project*, 2014.

[181] C. Westendorf, C. Gruber, and M. Grube, "Quantitative comparison of plasmodial networks of different slime molds", in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, 2016, pp. 611–612.

[182] E. P. White, E. Baldridge, Z. T. Brym, *et al.*, "Nine simple ways to make it easier to (re) use your data", *Ideas in ecology and evolution*, vol. 6, no. 2, 2013.

[183] K. E. Wohlfarth-Bottermann, "Oscillatory contraction activity in physarum", *Journal of Experimental Biology*, vol. 81, no. 1, pp. 15–32, 1979.

[184] D. Wöll, C. Kölbl, B. Stempfle, *et al.*, "A novel method for automatic single molecule tracking of blinking molecules at low intensities", *Physical Chemistry Chemical Physics*, vol. 15, no. 17, pp. 6196–6205, 2013.

[185] K. Xu, C. Tang, R. Tang, *et al.*, "A comparative study of six software packages for complex network research", in *Communication software and networks, 2010. iccsn '10. second international conference on*, 2010, pp. 350–354.

[186] T. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns", *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[187] M. Zhongwei, L. Zongxiang, and S. Jingyan, "Comparison analysis of the small-world topological model of chinese and american power grids", *Automation of Electric Power Systems*, vol. 15, p. 004, 2004.

[188] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results", *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.