On Local Behavior and Global Structures in the Evolution of Complex Networks

Dissertation
der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Katharina A. Zweig, geb. Lehmann
aus Hamburg

Tübingen
2007

Tag der mündlichen Qualifikation:	18.7.2007
Dekan:	Prof. Dr. Michael Diehl
1. Berichterstatter:	Prof. Dr. Michael Kaufmann
2. Berichterstatter:	Prof. Dr. Roger Wattenhofer, ETH Zürich
3. Berichterstatter:	Prof. Dr. Tamás Vicsek, Eötvös Lórand University, Budapest

This work is dedicated to my parents, Ursula Lehmann-Buss and Peter-Hannes Lehmann, who always supported me in everything I did, and Winfried Zweig, who is all to me.

CONTENTS

1. ACKNOWLEDGEMENT

I loved the time of scientific research, and the freedom I was given to explore whatever topic I liked in my graduate studies. However, the last few months of this time were overshadowed by fears whether the work is adequate and sheds at least some light on an interesting topic. In this time, many of my friends, my family, and my fiance Winni have supported me with all of their love. Thus, I want to thank my family and Winni for their never-ending prayers and care. Winni has done everything one can do, from listening to alien mathematical problems, to providing excellent food and endless coffee supply, and enduring my nervousness. I also want to thank my second family, Familie Zweig, who heartily welcomed me in their family, and I am happy to be Katharina Zweig in only a few weeks!

The work that is presented in this thesis is inspired by numerous works of others, and often worked out in collaboration with others due to the interdisciplinarity of the field. I want to thank all of my co-authors for our journey into the world of complex systems. Among these the most prominent is of course my advisor, Michael Kaufmann, who was willing to let me go into this adventure in a land of the unknown for both of us, with lots of drawbacks until we found the niche in which this new kind of computer scientist based network analysis was welcomed. I am grateful to have been part of the DFG SPP 1126, a strong community of mostly computer scientists concerned with new aspects of algorithms on complex networks with annual, fruitful meetings. Thanks to my colleagues, Martin Siebenhaller for all the fun and "Ha!", Markus Geyer for his humor and the long hours sitting on an—up to now—untractable problem. Thanks to the students that took part in the work, Hendrik Post (Small Worlds), Sonja Boldt (Simulation of Complex Networks), Karin Zimmermann (Robustness of Networks), Jan Vitense (Network Analysis), Andreas Gerasch (Amazon Crawler), Volker Menrad and Stephan Kottler (Backbones), Valentin Schwamberger (Clustering Algorithms), Ulrike Schöck (c-max Tolerance Graphs), and to my 'Dynamics of Complex Networks'-reading group, Michael, Timo, and Martin. In summary, all our students, the colleagues, and Michael as the head of the group have provided a beautiful working environment with lots of laughter, thank all of you! Thanks also to Marc Begin who has read and corrected, always with a sense of humor, my English (remaining mistakes 'go on my cap', as we say ;-)).

Starting as a biochemist in 1996, I am grateful to this day that I discovered that computer scientists are actually paid for solving the mysteries of mathematics and nature I had always been interested in. I thank God for this beautiful life of mine.

ZUSAMMENFASSUNG

Diese Arbeit beschäftigt sich mit netzwerkerzeugenden Prozessen in komplexen Systemen und ihrer Bedeutung für die Struktur des resultierenden Netzwerkes.
Beginnend 1998 mit einem Artikel von Duncan Watts und Steven Strogatz über sogenannte *Kleinwelten* [242], haben sich Wissenschaftler aus dem Gebiet der Analyse komplexer Systeme zunehmend mit der empirischen Untersuchung von realen Netzwerken beschäftigt. In vielen Publikationen konnte gezeigt werden, dass sich diese realen Netzwerke, also beispielsweise das Internet, das soziale Netzwerk bestehend aus Verwandten, Bekannten und Kollegen aller Menschen, aber auch biologische Netzwerke wie beispielsweise das Netzwerk von miteinander interagierenden Hormonen im menschlichen Körper, in ihrer Struktur stark von den bis 1998 hauptsächlich verwendeten Netzwerkmodellen, den sogenannten Zufallsgraphen, unterscheiden. Im Zuge dieser Untersuchungen wurden zahlreiche strukturelle Maße eingeführt, deren Werte in realen Netzwerken stark von denen in den genannten Modellen abweichen, selbst wenn die Anzahl der Knoten und Kanten in dem zu untersuchenden Netzwerk und dem Modellnetzwerk gleich sind. Zu diesen Maßen gehören der Clusteringkoeffizient [242], die Gradverteilung [21] und die Assortativität [177], um nur die bekanntesten zu nennen. Es wurde in verschiedenen Publikationen nahegelegt, dass Netzwerke mit den gleichen strukturellen Eigenschaften vermutlich auch durch den gleichen Prozess erzeugt wurden: so wurde beispielsweise für die Kleinwelten, die gleichzeitig einen hohen Clusteringkoeffizienten und einen kleinen Durchmesser aufweisen, von Watts und Strogatz ein einfacher Neuverdrahtungsprozess vorgeschlagen, der aus einem gitterähnlichen Netzwerk einen Graphen mit den Kleinwelt-Eigenschaften herstellt; für die sogenannten skalenfreien Netzwerke, in denen die Wahrscheinlichkeit $P(k)$ einen Knoten mit Grad k zu finden proportional zu $k^{-\gamma}$ ist, wurde der *preferential attachment*-Prozess vorgeschlagen, der zu Netzwerken mit ebendieser Gradverteilung führt [21]. Für die meisten Netzwerke ist es schwierig zu verifizieren, ob sie wirklich mit einem der genannten Prozesse erzeugt wurden, für das *preferential attachment*-Modell konnte es aber für die Entwicklung des Internets gezeigt werden [254]. Es wurden teilweise aber auch Prozesse vorgeschlagen, die zwar zu einem Netzwerk mit einer gewünschten Struktur führen können, die aber ganz offensichtlich in dem System, das zu dem zu analysierenden Netzwerk geführt hat, niemals abgelaufen sein können. Zu diesen Modellen gehört beispielsweise der von Ravasz et al. vorgeschlagene Prozess zur Modellierung der Evolution von metabolischen Netzwerken [201]. Die vorliegende Arbeit stellt die These auf, dass der netzwerkerzeugende Prozess für das Verständnis komplexer Netzwerke von großer Bedeutung ist, und zwar hauptsächlich wegen folgender Aspekte:

1. In den letzten Jahren wurden von Informatikern, Soziologen und anderen Wissenschaftlern verschiedene analytische Algorithmen entwickelt, die durch Analyse eines komplexen Netzwerkes eine Aussage über ein durch es beschriebenes komplexes System treffen sollen. Zu diesen Algorithmen gehören insbesondere die Berechnung von sogenannten Zentralitätsindizes und die Berechnung von Clustern, das sind dichte Teilgraphen, in denen die Knoten stark miteinander vernetzt sind. Diese Algorithmen haben einen bestimmten *Kontext*, in dem sie angewandt werden dürfen, nämlich nur dann, wenn die meisten Kanten eines komplexen Netzwerkes nicht-zufällig erstellt sind, sondern bevorzugt solche Objekte miteinander verbinden,

die in einer klaren Beziehung zueinander stehen. Stellt sich also heraus, dass ein Netzwerk zu 50% aus zufällig vorhandenen Kanten zusammensetzt, ist die Anwendung solcher analytischer Algorithmen nicht mehr angeraten. Die Frage, ob diese Art von netzwerkanalytischen Algorithmen auf ein gegebenes Netzwerk angewendet werden darf, kann also nur dann geklärt werden, wenn der netzwerkerzeugende Prozess bekannt ist.

2. Wir postulieren, dass in der Informatik in den nächsten Jahrzehnten Soft- und Hardware für solche Kommunikationsnetzwerke entwickelt werden müssen, die sich aus mehreren autarken und egoistischen Akteuren zusammensetzen, sei es im Bereich Sensornetzwerke [49], Peer-to-Peer-Netzwerke [224] oder dem Internet. Diese Akteure können nicht zu einer bestimmten, für alle im Netzwerk befindlichen Teilnehmer günstigen Handlungsweise gezwungen werden; stattdessen müssen wir annehmen, dass diese Akteure immer versuchen werden, das Netzwerk durch das Knüpfen neuer und das Löschen alter Kanten in einer für sie selbst im Moment günstigen Weise zu verändern. Um ein solches komplexes System aus egoistischen und teilweise auch für den Gesamtzusammenhang und die langfristige Entwicklung blinden Akteuren trotzdem steuern zu können, ist es notwendig, die Beweggründe der Akteure zu verstehen, und ihnen dann im Gesamtsystem Anreize zu bieten, die automatisch das für alle günstigste Verhalten belohnen und damit fördern. Hier ist es also notwendig, den netzwerkerzeugenden Prozess im Gesamtsystem zu verstehen, und dann die richtigen Anreize zu liefern, so dass das lokale Verhalten der Akteure eine global erwünschte Struktur erzeugt.

In dieser Arbeit haben wir uns mit den folgenden Aspekten von strukturbildenden, netzwerkerzeugenden Prozessen beschäftigt:

1. In Kapitel 2 geben wir eine Einleitung in die Themen der Arbeit und einen Überblick über die bisher publizierten Resultate im Bereich *komplexe Systeme*, *Netzwerkanalyse* und *komplexe Netzwerke*, soweit sie für das Verständnis der Arbeit notwendig sind. Dieses Kapitel wird gefolgt von allgemeinen mathematischen Definitionen im Kapitel 3.

2. In Kapitel 4 beschäftigen wir uns mit einem allgemeinen netzwerkerzeugenden Prozess, der die oben genannten Kleinwelten produziert. Dazu haben wir aus den bisher anerkannten Kleinwelt-Modellen, die alle auf leicht unterschiedlichen Definitionen dieser Netzwerkfamilie basieren, eine gemeinsame Basis extrahiert, das von uns sogenannte *Kleinweltphänomen*. Für die Familie von Netzwerken, die dieses Phänomen zeigt, beschreiben wir dann ein allgemeines Netzwerkmodell, das es dem Designer von Netzwerken ermöglicht Kleinwelten mit weiteren gewünschten strukturellen Eigenschaften flexibel zu gestalten. Der wichtigste Aspekt dieser Teil der Arbeit ist, dass wir die maximale Anzahl von Zufallskanten, die nötig sind um den Kleinweltcharakter zu erhalten, in diesem Modell leicht berechnen können.

3. Das in Kapitel 4 entwickelte Modell weicht von anderen Kleinweltmodellen insofern ab, als dass wir für einzelne reale Netzwerke keine Aussage darüber treffen können, ob sie Kleinwelten sind oder nicht, sondern nur für netzwerkerzeugende Prozesse darüber urteilen können, ob sie Netzwerke mit Kleinweltcharakter generieren oder nicht. In Kapitel 5 führen wir daher den Begriff des *netzwerkerzeugenden Systems* und des ihm innewohnenden *netzwerkerzeugenden Prozesses* ein. Die Kleinweltenmodelle, die in Kapitel 4 eingeführt wurden, gehen davon aus, dass die meisten realen Netzwerke zumindestens teilweise von einem Zufallsprozess erzeugt werden. In Kapitel 5 präsentieren wir eine neue Technik, mit der der maximale Anteil von zufälligen Kanten beschränkt werden kann, und wir zeigen an mehreren realen Netzwerken, dass dieser Anteil sehr unterschiedlich groß sein kann. Interessanterweise zeigt diese Technik auch, dass reale Netzwerke eine Struktur haben, die für verschiedene theoretische Probleme, beispielsweise die Berechnung von Zeitplänen [159] oder beim Lösen einer bestimmten Art

von linearen Gleichungssystemen [222], eine bessere worst-case-Laufzeit garantiert. Unseres Wissens ist dies die erste bekannte Struktur von realen Netzwerken, die die Laufzeiten von Algorithmen verbessert, und wir hoffen, dass diese Entdeckung zu weiteren effizienten Algorithmen auf realen Netzwerken führt. Die Technik basiert darauf, in dem gegebenen Graphen einen Spannbaum zu finden, so dass auch die Knoten derjenigen Kanten, die nicht Teil des Spannbaums sind, in dem gegebenen Baum nur eine kleine Distanz haben, so dass also die durchschnittliche Distanz aller Knoten, die im Gesamtgraphen durch eine Kante miteinander verbunden sind, auch im Spannbaum klein ist. Wir werden zeigen, dass es NP-hart ist, den nach diesem Kriterium optimalen Baum zu finden, daher werden wir Heuristiken vorstellen, die zumindestens in den von uns untersuchten realen Netzwerken zufriedenstellende Ergebnisse liefern. In einem letzten Abschnitt dieses Kapitels werden wir kurz skizzieren, wie wir mit Hilfe eines heuristisch optimierten Spannbaumes reale Netzwerke mit mehr als 1.000 Knoten visualisieren, um mehr über die Struktur dieser Netzwerke zu lernen.

4. Kapitel 6 greift dann die oben skizzierte Idee auf, dass sich die Informatik in den nächsten Jahren viel mit dem Design von komplexen Kommunikationsnetzwerken beschäftigen wird, in denen keine direkte Kontrolle auf das Verhalten der einzelnen Akteure ausgeübt werden kann. Wir beschäftigen uns in diesem Gebiet vor allen Dingen mit grundlegenden Fragen, nämlich zuerst mit einem allgemeinen Modell für die Evolution solcher Netzwerke. Da sich schon viele verschiedene Gebiete mit der Frage nach netzwerkerzeugenden Prozessen und deren Modellierung beschäftigt haben, stellen wir hier die wichtigsten dieser Modelle vor, und kombinieren sie zu einem allgemeinen Modell für die Erzeugung der von uns sogenannten *singulären, selbstzentrierten und selbst-organisierten* Netzwerke. Wir zeigen an einem Beispiel, dass die Entwicklung von netzwerkerzeugenden Prozessen für diese Netzwerke sehr schwierig sein kann, da unter Umständen die genaue Formulierung einer netzwerkbildenden Regel den Unterschied zwischen einer exponentiellen und einer polynomiellen Laufzeit bis die gewünschte Netzwerkstruktur erreicht ist ausmachen kann. Wir zeigen aber auch, dass diese selbstzentrierten Netzwerke, in denen kein Akteur einen globalen Überblick hat, in der Lage ist, zwischen zwei globalen Netzwerkstrukturen nach Bedarf zu wechseln, nämlich solchen mit einer Gradverteilung die Poisson-verteilt ist, und einer skalenfreien Gradverteilung. Während erstere Struktur in einem normalen Szenario von Vorteil ist, in dem die bestvernetzten Akteure gezielt attackiert werden, ist die zweitere Struktur immer dann von Vorteil, wenn Akteure nur zufällig ausfallen. Wir zeigen, dass auch diese dezentralen, lokalen Netzwerke ihre Struktur je nach Szenario wechseln können, wenn sie sich in einem Netzwerk befinden, dass es erlaubt, Kanten zu geringen Kosten neu zu knüpfen, falls Nachbarn ausfallen.

Insgesamt weisen wir in dieser Arbeit auf die große Bedeutung von netzwerkerzeugenden Prozessen hin, sei es um festzustellen, welche Art von analytischen Methoden angewendet werden kann, oder um netzwerkerzeugende Prozesse in komplexen Systemen in eine gewünschte Richtung steuern zu können. Natürlich ist dieses Feld sehr weit, und sowohl in der Frage, wie wir herausfinden können, welcher netzwerkerzeugende Prozess zu einem bestimmten Netzwerk geführt hat, als auch in der Frage, mit welchen Regeln wir ein komplexes System ausstatten sollten, damit ein Netzwerk mit einer gewünschten Struktur entsteht, haben wir in dieser Arbeit erst einige wenige Beispiele grundlegend bearbeitet. Wir hoffen aber, dass dieser neue Ansatz zum Verständnis und zur Manipulation komplexer Netzwerke in naher Zukunft Früchte trägt, und neue Algorithmen und Techniken liefert, mit der die steigende Komplexität unserer Umwelt gemeistert werden kann.

2. INTRODUCTION

In the last decade a new interest in network analysis has arisen under the name *complex network science*. By now, the field is quite dominated by empirical observations and model–building with questions such as: What is the structure of real–world networks and how can they be modeled? These questions have mainly been posed and naturally can be very well answered by physicists and other natural scientists using their large arsenal of methods. At first sight, it may seem that this new field that is so strongly empirical and often follows only loosely defined questions, has not much to tell computer scientists, and vice versa. Because this is a doctoral thesis in computer science, we will discuss and explore where this new field of complex network science opens problems that are both interesting for the theoretical computer scientist, and that can be solved with the methods developed in this field, and also where computer scientists can contribute to the field of complex systems science.

Complex network science has developed within *complex systems science*, a field that has rapidly grown in the last decades [156, 181, 236]. Articles from this field have explained or at least shed light on topics as different as the rise and fall of stock market indices [156], earth-quake and snow avalanche analysis [115], the behavior of certain chemical reactions [246], neural network patterns in neuroscience [131], function and structure of biological networks [7, 168, 221], automatic categorization of web sites [124], and swarm behavior [190, 36], to name but a few.

Since complex network science is a subfield of complex systems science, we will first give a rough idea of the term *complex system* in 2.1, together with a short history of the subfield of *complex network science*, which is the main topic of this thesis. In 2.2 we discuss network modeling as an approach for reducing complexity in complex systems and its importance for several applications. In 2.3 we explore the main fields of interest for computer scientists in the realm of complex network analysis, from analysis and modeling of the WWW to the correct use of so–called *contextual algorithms*. We conclude the introduction in 2.4 by an overview of the topics addressed in this thesis.

2.1 Complex Networks in Complex Systems Science

In order to understand the current interest in complex networks, it is easiest to take a short detour to the history and notion of complex systems science. For centuries, it seemed the best approach to understanding natural phenomena by dissecting them into as small parts as possible, and analyzing these parts in isolation, neglecting the interactions they might have with each other. A typical example of this *reductionism* can be observed in the analysis of the behavior of falling objects. Although the behavior of falling objects seems to be complicated to describe, it became much easier to understand after it was recognized that the fall of any object is subject to two different forces—namely, gravity and friction. With this discovery it was easy to think of experiments that could reveal the nature of both forces in isolation. By reducing the whole system to two parts, beautiful and simple laws to describe gravity and friction emerged that could then be combined to understand the overall behavior of falling objects. In general, this reductionistic approach can be used to explain how single objects under multiple forces will react.

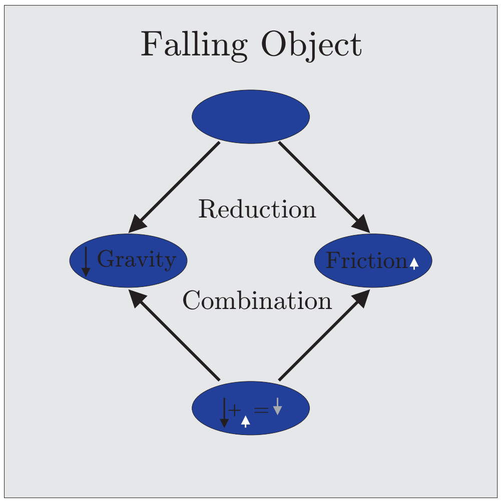
Later, a model that would describe just the opposite, namely, a system composed of multiple objects under the same force, was needed to describe the behavior of gases [16]. It turned out that with gases the properties of the whole system can be calculated by treating the system as a statistical ensemble of single particles, giving this part of physics the name *statistical mechanics* [16]. In detail, the property of the system is a function of the *expected distribution* of the properties of single atoms, like their kinetic or potential energy. Although this model treated for the first time a system composed of many entities and their interactions, i.e., the distribution of energy due to collisions, gases are not generally considered complex systems because the properties of gases scale with their size, obeying the same laws whether one liter, one thousand liters, or a million liters are observed, and most interactions between particles such as attraction or repulsion can be ignored on a broad scale.

But it turned out that some systems consisting of multiple, interacting entities cannot be understood by analyzing the properties of single entities or their distribution, but only by analyzing their properties **and** interactions. A straightforward example of this is the ability of a colony of ants to find a short path from its nest to a food source [36]. The ants do this by first walking randomly until they have found a promising food source. They will then return to the nest, leaving little droplets of pheromone on their way. This pheromone tells other ants to follow the trail, whereby they also deposit some pheromone. Note that the pheromone diffuses into the air approximately radially from the spot, and thus, if the trail has a bend, the concentration of pheromone will be highest on the inside of the bend. Thus, a new ant following the trail will short-cut the bend a bit, and furthermore leave a fresh trace of pheromone spots on a new trail with a shallower bend. This process will eventually lead to straight trails, with a short distance from the nest to the food source [48]. This phenomenon cannot be understood by looking at a single ant. Only the *collective behavior* of the whole ensemble of interacting ants enables this phenomenon to occur, making it a so-called *emergent property of the system.*

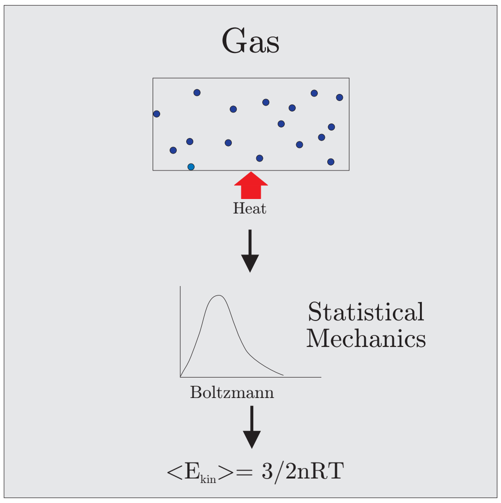
This new kind of phenomena requires a new kind of science:

> ... The world is indeed made of many highly inter-connected parts on many scales, the interactions of which result in a complex behaviour that requires separate interpretations of each level. This realization forces us to appreciate the fact that new features emerge as one moves from one scale to another, so it follows that the science of complexity is about revealing the principles that govern the ways in which these new properties appear. [235]

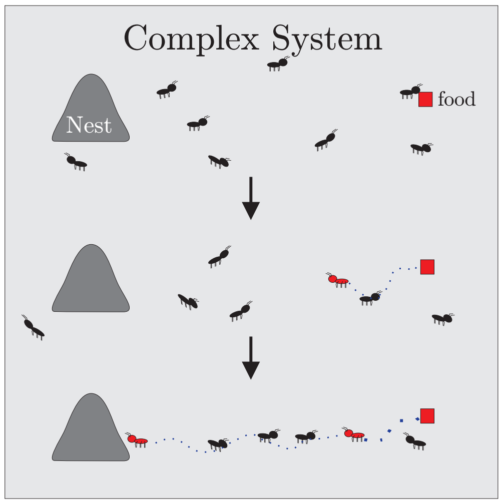
Intuitively, a complex system is one whose pattern and behavior is hard to describe and understand by dissecting it into different units, while disregarding their interactions. Following this intuition, everyone would consider the global finance market or the global climate a complex system whereas the power supply network of a house—be it as complicated as it may—would not qualify as being complex. Despite or maybe also because of this intuitive understanding there is no "single sentence answer" ([184]) to the question of what a complex network is: whenever people tried to rigorously define the notion of a complex system it was either too general, including systems that are not generally considered complex, or too strict, and thus, as Newman points out: "Many definitions of "complex systems" have been proposed over the years, and at present there is none which is universally accepted." [178]. However, this phenomenon of emergent properties induced by the interaction of multiple entities is the common ground of complex systems science, as Newman states: "Most people however would agree that a fundamental property of complex systems is that



(a) reductionism



(b) statistical mechanics



(c) complex system approach

Fig. 2.1: Three different approaches to analyze a system. (a) *Reductionism:* If a system consists of an object subject to different forces, these can be analyzed in isolation. The resulting force on the object is given by the sum of all forces. (b) *Statistical Mechanics:* If a system consists of multiple objects subject to one force and the interactions between them can be neglected, statistical mechanics can be used to deduce properties of the system. (c) *Complex Systems:* If a system consists of multiple objects whose interactions cannot be neglected, a complex systems–approach is needed.

they are composed of a large number of components or "agents", interacting in some way such that their collective behaviour is not a simple combination of their individual behaviours." [178]

Moreover, real complex systems like the aforementioned global finance market are in most cases composed of different kinds of entities that are entangled in multiple kinds of relationships, making the effects of one kind of entity or one kind of relationship hard to understand. To reduce this complexity, *complex networks* concentrate on one kind of entity and mostly only one kind of relationship between them by representing entities by vertices, and their interactions by edges between the vertices. There are mainly two types of perspectives concerning complex network science. The first is concerned with the special structure of complex networks and how it emerges:

> As we are just beginning to realize however, there is a third aspect to these systems which may be even more important and which has so far received little attention, and that is the *pattern* of interaction between agents, i.e., which agents interact with which other. [178]

The second perspective views complex networks as a kind of *skeleton* of the complex system at hand which has to be understood before processes of the whole system using that skeleton can be understood, as the next citations emphasize:

> Why is network anatomy so important to characterize? Because structure always affects function. For instance, the topology of social networks affects the spread of information and disease, and the topology of the power grid affects the robustness and stability of power transmission. [225]

> As it stands, network theory is not a proxy for a theory of complexity - it only addresses the emergence and structural evolution of the skeleton of a complex system. The overall behaviour of a complex system, which we ultimately need to understand and quantify, is as much rooted in its architecture as it is in the nature of the dynamical processes taking place on these networks. [20]

In this sense, the latter kind of perspective views complex networks as the basis of more complex processes in the whole system at hand and will analyze questions on how these two interact. In any case, complex network science is now an accepted field in the realm of complex systems science, with many applications in different disciplines, as stated by the living roadmap of the *Open Network of Excellence in Complex Systems (ONCE-CS)* [184]. In the following we will give a short historical overview of the field of complex networks science.

2.1.1 A Short History of Complex Network Science

In a very broad definition, we will denote by *complex network science*[1] all research that can be subsumed under the following three perspectives:

[1] As will be obvious on the following pages, complex network analysis is quite a young and very interdisciplinary field. The publications on which this thesis is based come from sociology, mathematics, physics, computer science, and biology. Naturally, there is a problem with different languages, and often the same term is used to cover different concepts, and the same concept is known under a hundred names. Discussing the work of others I will always try to translate the concepts into the well-known, traditional notions of graph theory and will consistently use graph theoretic terms even if the original work has introduced a new name for a concept. For clarity, I will mark these 'translations' in footnotes where appropriate.

1. **Complex Network Analysis**: measures and algorithms introduced to understand the special structure of real–world networks by differentiating them from established graph models.
2. **Complex Network Models**: models that capture essential structural properties of real–world networks.
3. **Processes on Complex Networks**: analysis of the outcome of a process or algorithm on a given network structure.

We will give a short overview of the most important findings in each of the three realms as far as they are important for the understanding of this work.

Complex Network Analysis

As *complex network analysis* we understand any approach that maps a real-world problem or situation to a graph, analyzes this graph, and transforms the results of the analysis back to the real-world problem or situation. In this definition, the term *network* is missing, and instead the term *graph* is used. And indeed, in many publications the terms are used interchangeably. In this work, we will also use both terms, but with the following distinction: If we speak of a *graph*, only the mathematical object is meant, i.e., an object composed of vertices and edges that may be directed or undirected, and weighted or unweighted. In the case of network analysis, the graph represents real-world entities and one or more relationships between these entities. If we speak of a *network* we refer to the entities and their relationship represented by the graph. The difference is subtle, so we will illustrate the point with the example of the autonomous system of the Internet [80][2]. An autonomous system is defined as a set of routers within the same domain, specified by a special pattern of the IP addresses. These routers are physically interconnected by different types of cables. The routing table of a router specifies for any given target IP address where the router will send the message to next. These entries thus identify those routers with which the router is physically linked, and by reading the routing tables a **graph** can be built in which every router is represented by a vertex and two vertices are connected by an edge if the corresponding routers are physically linked. In the corresponding communication **network**, each router can be described by a distinct set of properties, e.g., its position, its machine address, or its bandwidth, and it can be in different relationships with other routers, whereas in the **graph** structure, vertices have lost any information about the router they represent except the information about with which other routers it is connected [3]. In this sense, we will also use the terms 'graph' and 'network' (quite) interchangeably, but stick to the term 'graph' as long as only the adjacency information is needed, i.e., the information about which object is related to which other object.

The first application of network analysis—and at the same time the beginning of graph theory—was the famous solution to the seven bridge problem of Königsberg by Euler in 1735, published in 1752 [77]. Legend has it that people were asking themselves whether it was possible to walk over all bridges without crossing at least one of them twice. Euler saw that this problem could be generalized to the question of whether a graph admits a tour that visits all vertices without using any edge twice, a tour which is now called an *Euler tour*. The graph of Königsberg is constructed

[2] The authors of [80] call this graph the *inter-domain topology of the Internet*.

[3] A sentence like 'router A is connected with router B by an edge' intermingles these two settings. However, such a sentence is immediately understandable for everyone and much shorter than the two alternatives: 'The vertex representing router A is connected with the vertex representing router B by an edge' (graph setting), or 'Router A is connected with router B by a copper cable and thus they are related to each other'. Although a clear and strict treatment will not allow for a sentence like 'router A is connected with router B by an edge' we will sometimes identify the vertex with the object it represents to shorten text without losing understandability.

by representing parts of the town by vertices where two vertices are connected if the corresponding parts of the town are connected by a bridge. With this transformation, the solution of the more general problem of finding an Euler tour showed that there is no walk using all bridges that does not use at least one bridge twice. Although Euler did not directly transform this problem into a graph, but rather formulated it as a sequence ordering problem, this is considered to mark the beginning of graph theory [102].

From the first work of Euler, graph theory then flourished as a mathematical field, and many structural properties of graphs were defined and analyzed [33, 101]. Next to the analysis of general graphs, another type of publication was concerned with the structural analysis of certain graph models, foremost random graph models [76, 91, 220]. In this type of graph model, every edge has the same probability of existing, independent of the existence of other edges. Since these graph models are mathematically tractable, many of their structural properties are well-known; for a survey see [34].

Next to understanding graphs in their own right, graph theoretic ideas were also used to understand social and psychological phenomena. For example, one question concerning social systems was whether important people could be determined by looking at social networks, i.e., the question was transformed to finding the most important vertex of a graph. For this purpose, so-called *centrality indices* were developed in the 1940s to the 1970s [13, 28, 87, 208, 212, 247], followed by centrality indices for the analysis of the world wide web in the 1990s [187, 152, 124]. Looking at the journals in which these measures were published, it is already clear that the field of complex network science is extraordinarily interdisciplinary, ranging from sociology (**Human Organizations, Sociometry**), biology, mathematics and physics (**Bulletin of Mathematical Biophysics**), psychology (**Psychometrika**), chemistry (**Journal of the American Chemical Society**), and computer science (**Computer Networks**). These works aimed to understand the structure of real-world networks, where the term *structure* is loosely defined as: 'From the view of social network analysis, the social environment can be expressed as patterns or regularities in relationships among interacting units. We will refer to the presence of regular patterns in relationships as *structure*.' [237] (p. 3)

Until the 1980s, only very small and most often social networks were analyzed by the proposed structural measures, e.g., a set of 21 managers and their consulting relationship, i.e., who would go to whom to ask for advice [134], or the marriage and business ties between 16 families in 15th century Florence, Italy [121]. With the onset of personal computers, much larger data sets could be analyzed and today there is a large set of applications to very different fields, e.g., bioinformatics [84, 116, 166, 188, 201], social network analysis [2, 19, 65, 97, 176, 66], e.g., high school dating networks [29], sexual relationship networks [160, 161], or acquaintanceship and co-worker networks [22, 215], co-purchasing networks [54, 55], and communication networks [242], to name but the most important.

With the analysis of huge data sets it became clearer that real-world networks have a structure that deviates strongly from the structure of random graphs, creating a need for new network models and graph families to explain this structure.

Complex Network Models

For a long time, real–world networks were considered to be at least approximately representable and analyzable as random graphs, an assumption which was rendered invalid in 1998 by the article of Watts and Strogatz [242]. They introduced the so–called *clustering coefficient* whose value strongly deviates from the expected one in random graphs [242]: the clustering coefficient measures for each vertex the probability that two vertices chosen at random from its neighborhood are also connected by an edge. The interpretation of a high clustering coefficient in a network is that the connections are somewhat *local*, and that if A knows B and C it is very likely that B and C are also 'near' each other and thus have a chance to 'know' each other. This is a very intuitive assumption in social and many other networks, where a friend of a person will often know a large portion of the latter's other friends. In [242], Watts and Strogatz could show that the average clustering coefficient is much higher in real-world networks than expected in a random graph that has a comparable number of nodes and edges. Still, the average distance between any two vertices in the real-world network is comparable to those in the corresponding random graph.

This finding required a new model, the so-called small-world model [242]. The name was inspired by a series of experiments conducted by Milgram in the 1960s [165, 231] which aimed to show that any two persons in the United States are linked by a short sequence of acquaintances. The experimental setting was given as follows: randomly chosen participants living in Nebraska or Boston were asked to send a letter to a broker in Boston. The restriction was that the letter could only be given to persons that the sender knew on a first-name basis. The main information known about the target person was his position, his city of birth and the schools he went to, his name, and his address. It turned out that those letters that made it to the target needed on average only six intermediary steps, which led to the famous name 'six degrees of separation' and the term 'small-world network'.

Since there was this discrepancy between the commonly used model and reality, a new graph model was introduced by Watts and Strogatz, which will be discussed in detail in Chapter 4. After its introduction, an immense richness of literature followed with mathematical analyses of the new model and its variants [25, 24, 68, 174, 175], and further empirical analyses of a large set of additional types of real-world networks [10, 84]. Thus, their article clearly marks the beginning of the new field of complex networks science that is dominated by the search for clear structural markers that differentiate real–world networks from simple random graph models.

A second cornerstone of the newly emerging field was the paper by Albert, Jeong, and Barabási that showed the world wide web's link structure to be far from what would be expected in a random graph [8]. In a random graph, the number of edges a vertex has is expectedly Poisson–distributed [34]. In contrast, this so-called *degree distribution* is scale–free (s. 3.2.5) in most real–world networks, signifying that most vertices participate in a low number of edges, and a few participate in many more than ever expected in a random graph[4]. In a second paper, Barabási and Albert provided a model that explained how such a scale-free degree distribution could emerge (s. 3.4.3) [21]. Next to these two most important network models, numerous others have been proposed to model in more detail, e.g., citation networks [129], protein-protein interaction networks [219, 232], the world wide web link structure [31], and metabolic networks [201].

Processes on Complex Networks

In the classical paper of Watts and Strogatz, first questions of how the network structure could affect processes in the whole complex system were already posed and experimentally analyzed. This opened up another field of research where the interaction between structure and function of complex networks was explored. Here we just sketch some of the most important findings to give a feeling for the kinds of questions asked in this field: An interesting aspect of the small-world model was brought up by Kleinberg through the question of how people in a small-world are able

[4] Actually, the same finding was also published in the same year by the Faloutsos brethren [80], but Albert et al.'s paper had the greater impact and was more widely acknowledged.

to **find** a short path. His model assumes that every vertex in the small-world has a globally known position, but that each single vertex only knows the position of itself, the target, and its direct neighbors. Kleinberg could show that there is only a very limited class of graphs that allows a greedy search algorithm with an expectedly poly-logarithmical runtime [125, 126]. This paper thus gives a concise relationship between a possible routing algorithm and the necessary network topology to make it efficient. A further, almost shocking finding was described by Albert, Jeong, and Barabási who showed that networks with the ubiquitous scale–free degree distribution are much more prone to directed attacks than random graphs [6]. This paper preceeded the 9/11-terrorist attack, but it was the first finding in this emerging field that indicated that most of our modern technological, communication, and transport networks had an *architectural flaw* that made them vulnerable, and that showed how important it is to conduct network analysis. On the other hand, Albert et al. also showed that scale-free networks perform much better than random graphs under random failures of vertices. This indicated that a safe system should be able to switch its topology according to the context, i.e., attacks or random failure. In 6.6 we will show that in at least some systems, where the edge costs are low enough to allow for rewiring as a reaction to attacks, this can indeed be accomplished by a simple protocol. Other articles were concerned with disease spreading on different network structures [191, 192], information cascades and cascading failures [240], and efficient routing algorithms and congestion analysis [108, 95].

As we have indicated above, the number of network models for real–world networks proposed to date is actually much higher than the number of different types of real–world networks analyzed so far. It is thus natural to ask why network modeling is so important in complex network science. We will discuss this question in the following section.

2.2 Network Modeling - An Approach to Reduce Complexity in Complex Systems

Network modeling, i.e., finding an abstract graph family that appropriately models the real-world network's structure, began with the seminal paper by Watts and Strogatz. Since it is quite a new scientific approach to the understanding of real-world networks it has to be justified. As Strogatz points out in a review paper:

> Networks are on our minds nowadays. Sometimes we fear their power - and with good reason. On 10 August 1996, a fault in two power lines in Oregon led, through a cascading series of failures, to blackouts in 11 US states and two Canadian provinces, leaving about 7 million customers without power for up to 16 hours. The Love Bug worm, the worst computer attack to date, spread over the Internet on 4 May 2000 and inflicted billions of dollars of damage worldwide. [227]

To understand these processes we need to understand the underlying structure. And to understand any system or structure we need an appropriate modeling as described in an article by Rosenblueth and Wiener on *The Role of Models in Science*:

> No substantial part of the universe is so simple that it can be grasped and controlled without abstraction. Abstraction consists in replacing the part of the universe under consideration by a model of similar but simpler structure. Models, formal or intuellectual on the one hand, or material on the other, are thus a central necessity of scientific procedure. [207]

A model is needed to simplify the matter at hand to make it at the same time intuitively understandable and analyzable in terms of a formal, mathematical analysis. If now a real–world network is given, how can a model be determined that describes it best? The problem is that science is about *falsification* [198], i.e., we can only show that a given model **does not fit** the real-world network. Nonetheless, such negative information is actually interesting: in the properties in which the model **fails** to fit the real-world network there is some hidden structure that needs to be determined and analyzed. Thus, network modeling has made progress by comparing the structure of real-world networks with that of network models, mainly from the field of *random graphs*. The *experiments* have the following form: a measure is computed for real-world networks and for a corresponding graph from the model family, where 'corresponding' normally means that it has the same number of vertices, edges, and maybe other structural properties that are already known about the networks to be analyzed. It is thus very important to find the best model family for comparison with the real-world network, as Rosenblueth and Wiener state:

> An experiment is a question. A precise answer is seldom obtained if the question is not precise; indeed foolish answers—i.e., inconsistent, discrepant or irrelevant experimental results—are usually indicative of a foolish question. [207]

And correct answers can only be achieved if the model is correct. If, e.g., a network is known to be of a special type, the question of how to explore it efficiently can be answered with respect to its structure. We want to illustrate this with the example of the exploration of so–called *protein–protein interaction networks*. To understand a cell's protein architecture, it is necessary to analyze the interaction pattern of its proteins. The experiments that are needed to find out which pairs of proteins interact are extremely expensive and time–consuming. In a widely used experiment, the *yeast two-hybrid* system, two cells have to be genetically engineered for every protein, once as the so-called *bait* and once as the *prey*. These cells compose a so–called *library*, and one experiment has to be done for almost every bait–prey pair. Since building a library as described above is extremely time–consuming and needs a great deal of time and chemical resources, it would be helpful if a theory could guide scientists towards those experiments which would yield the most information and should thus be done first. In [141] the authors show by virtual experiments on already known protein-protein interaction networks that there is a strategy such that only a third of all proteins have to be used as bait to discover up to 90% of all interactions. This strategy works as follows: start with a small set of proteins as bait. After this set is processed, always pick the one protein that interacts with most of the proteins analyzed so far as the next bait. This strategy relies heavily on the idea that protein-protein interaction networks are scale-free networks, and it is thus likely that such a protein will not only interact strongly with the set of proteins already processed but also with the ones that have not yet been used as bait.

But of course, such an algorithm is only as good as the model it is based on, and whether a protein-protein interaction network is really best characterized as a scale-free network was questioned only half a year later [199]. The authors of this article show that protein-protein interaction networks are much better matched by so-called *random geometric graphs* where vertices are positioned uniformly at random in a metric space and every vertex is connected to all vertices within a unit disc around its own position. We will later show that these kind of graphs also have a high clustering coefficient, i.e., neighbors of neighbors tend to be also connected by an edge. If a graph has a high clustering coefficient and a scale-free degree distribution, one can assume that the greedy strategy proposed by [141] is not as successful as claimed because here many of the interactions of the next bait will be with proteins that have already been used as bait, and will thus only confirm interactions that had already been known before.

Another example also comes from biological networks. In the last decade, the search for so-called network motifs in biological networks has been promoted especially by a group around Uri Alon. In their work, the occurrence of small subgraphs with a certain structure is compared with the expected occurrence in a suitable random graph model [168, 166]. If a network motif occurs more often than expected in the chosen random graph model, this motif is assumed to carry a function that gave it an evolutionary advantage. Milo et al. showed—among other things—that certain motifs are more frequent in neural networks than in their random graph model and implied that this is so because of an evolutionary advantage of these motifs for the brain. As [14, 167] discussed, this interpretation is highly dependending on the underlying random graph model. Especially, in the case of neural networks, it is known that these are *local graphs*, i.e., a neuron has a special position in the body of an organism, and it is much more likely to be connected to neurons near by than to neurons far away. Artzy-Randrup et al. show that a very simple local network based on vertices placed on a grid where nodes have a high chance of being connected to near-by nodes will also show a motif frequency pattern that is far from that of a random graph. Interestingly, this very simple *toy network* shows an over-occurrence of some of those motifs that are also more frequent in neural networks than in the corresponding random graph. Artzy-Randrup et al. do not state that their toy network matches the neural network in any sense better than the total random graph, but they state that - if a comparison is needed to decide whether a network motif is frequent - it is important to have a random model that is as closely as possible modeling all structural information we already know about the network. In summary, understanding which network model best captures a real–world network is very important for the design of efficient algorithms and analytical methods, and may save time and effort exploring real–world networks and provide new insights into their formation.

The design of efficient algorithms, e.g., routing and other distributed algorithms [194], but also the analysis of electrical networks [34], or network design as a subfield of chip design [154] has a long history in graph theory and computer science. So, what is the difference to complex network science and why should computer scientists bother? In the following we will discuss the difference between the approaches, and show where the fields overlap and where we think that they can benefit from each other.

2.3 Computer Science and Complex Network Science

Until now, complex network science has been mainly dominated by the empirical analysis of real–world networks and their structures. At this time, it is understood rather less as an engineering science than a natural science that tries to understand and model our surroundings by observation. What are the questions that can be answered by computer science, when computer science is concerned with the question of how to compute something efficiently in a well–defined setting? We see four main points of interest:

1. Analysis of the Internet and the WWW (2.3.1).
2. Algorithms on graph classes as defined by the new network models (2.3.2).
3. Network design for multi-agent systems (2.3.3).
4. Contextual algorithms in complex network analysis (2.3.3).

2.3.1 Analysis of the Internet and the WWW

The analysis of the Internet and its overlaying networks like the WWW or peer-to-peer networks are obvious examples where computer scientists can and have profited from models and analytic tools from the field of complex systems science and complex network theory. As stated in the following quotation, a thorough understanding of the Internet's structure and its network–generating process are crucial for all services based upon it:

> Network generators that capture the Internet's large-scale topology are crucial for the development of efficient routing protocols and modeling Internet traffic. Our ability to design realistic generators is limited by the incomplete understanding of the fundamental driving forces that affect the Internet's evolution. (...) In light of extensive evidence that Internet protocol performance is greatly influenced by the network topology (...), network generators are a crucial prerequisite for understanding and modeling the Internet. Indeed, security and communication protocols perform poorly on topologies provided by generators different from which they are optimized for, and are often ineffective when released. (...) Thus to efficiently control and route traffic on an exponentially expanding Internet, it is important that topology generators not only capture the structure of the current Internet, but allow for efficient planning and long-term network design as well. [254].

But since the Internet and the world wide web are the result of independent entities that added their routers to it or built their websites, these networks cannot be analyzed by the classical, computer scientific approaches, but rather:

> The Internet is also the first object studied by computer scientists that must be approached with humility and puzzlement, and studied by measurement, experiments, and the development of models and falsifiable theories—very much like the cell, the universe, the brain, and the market. [79]

In the case of the self–organized Internet and the design of efficient algorithms on it, only the interdisciplinary approach of complex system science, including game theoretic approaches, and computer science seems to be able to tackle the task.

2.3.2 Algorithms on Special Graph Classes

The above paragraph concerns the quest for efficient algorithms on a specialized kind of network, namely the Internet and the WWW. More generally, classic graph theory is often interested in finding optimal solutions for some problem on a given graph, e.g., the so-called *vertex cover* problem: What is the minimal number of vertices in a graph such that every edge is incident with at least one of them [205]? Whenever such a problem is stated, the first question is: can the problem be solved efficiently, i.e., in a polynomial number of time steps, on a general graph? If the answer is no, a subsequent question is often: can it at least be computed efficiently on special graph classes, such as planar graphs, series–parallel graphs, or hypercubes [101]? These questions try to find the limiting cases where problems become hard, i.e., computationally infeasible.

So far, these special graph classes have been quite restrictive and artificial, and thus unlikely to ever occur in any real application. The new network models capture at least some major structural properties of real–world networks and, especially in the emerging field of algorithm engineering, these can lead to better algorithms for the expected input. Since the new network models are on the one hand much younger than the more established random graph models, and on the other hand often more difficult to analyze, there are still many open questions in this area. A first result in this realm is, e.g., given by Cooper, Klasing, and Zito who were able to compute lower bounds and new algorithms for computing dominating sets in web graphs [56].

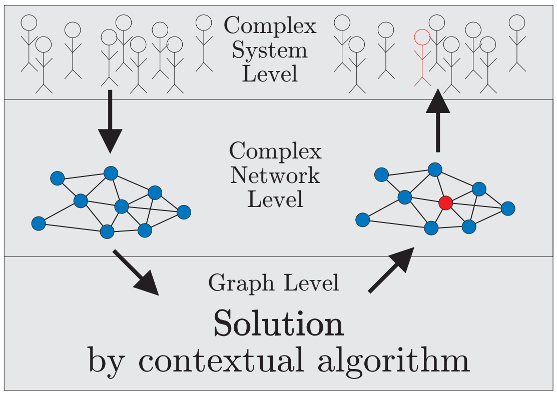
Generalizing the point above, network models from complex network science define interesting, but inherently more difficult graph families for which it is not yet clear whether more efficient algorithms can be designed. In chapter 5 we will introduce one such structure that has already been proven to make algorithms more efficient, and we hope that this will be a starting point for more research in this direction.

2.3.3 Network Design in a Multi-Agent System

Network design problems belong to one of the classical realms in theoretical computer science [205]. In network design a typical question might be: given a set of points in the plane, how can a network be designed such that it connects these points with minimal total edge length?
The design of optimal network processes on a given graph is also a classic problem in computer science, e.g., network flow problems [3]. In the analysis of network processes, a typical question might be: given a network of streets and a set of destinations and targets, how can the transport of goods be scheduled optimally? Both kinds of questions assume that the network structure and the process on it can be centrally supervised and designed, and that there exists a global optimality criterion. The fundamentally different approach in complex network science is that neither network structures nor ongoing processes are centrally organized but stem from the effort of multiple, and possibly selfish and myopic entities. In this sense, classical network design and process problems are posed from the perspective of an engineer who has freedom to construct them whereas the complex network scientist takes the perspective of an observer, trying to reverse–engineer the process that might have led to the system at hand. Here, typical questions are: what kind of network structures emerge in real–world systems and what process does most likely lead to these structures? How are processes directed over the network if every single vertex makes its own decisions locally? Thus, the main difference between the approaches is the amount of control we assume to have over the system at hand.

Both fields overlap whenever networks are analyzed that are built by multiple, independent entities but where at least some control can be exerted, e.g., by law enforcement, and hardware or software capabilities. The Internet is one of them, as already stated above. Another typical example of these kind of networks are peer-to-peer networks, i.e., networks among groups of users, established by the same software, to share data or to chat [23, 224]. Inbuilt in this software are rules governing how a new user is connected to the existing network and how the network changes if he leaves it. It has been shown that due to these rules some peer-to-peer networks will show small–world properties and scale-freeness of the degree distribution [146]. By appropriately designing this software, it is also possible to give incentives for desired behavior, e.g., for sharing the user's bandwidth with others, which will then influence the topology of the network structure and subsequently the flow of information over it. In all of these networks, it is desired that the whole process of network formation occur in a self-organized and decentralized manner but still result in a globally (near–) optimal structure.

It seems that our modern world will become increasingl inhabited by technical communication networks consisting of small, independent, and maybe also mobile units that create, rewire, and remove links to other units, and here, software can be built to guide the network–generating process



(a)

Fig. 2.2: In complex system analysis some questions are answered by transforming the complex system into a complex network, subsequent application of a graph theoretic algorithm like the computation of centrality values or a clustering algorithm, and retransformation of the result into the complex system. Since the choice of the correct algorithm on the network/graph level depends on the context defined by the complex system, these algorithms are called *contextual algorithms*.

towards a desired structure—but only if the mechanisms of decentralized and self-organized network formation have been understood and desired network structures have been identified.

A last point of interest concerns the design of algorithms to answer network analysis questions.

Contextual Algorithms in Complex Network Analysis

In the past four years we have worked with centrality indices and different clustering algorithms, and it has become very clear that in the field of complex network analysis, algorithms have a *context*, as we will briefly discuss here and in more detail in 5.1.2. In complex network analysis, algorithms are often used to give an answer to a question about the *complex system* they represent. As we have sketched above, centrality indices where proposed to answer the question of who is the most important entity in a complex systems. To answer this question, literally dozens of measures have been proposed [132, 133, 113], but—as Borgatti points out in [37]—every one is designed for a special kind of process on the network that determines the notion of importance for this network. We will discuss this point more closely in 5.1.2, but the summary is that not every centrality index *should* be applied to every network although technically it *could* be.

The same is true for *clustering algorithms*, i.e., algorithms that try to find those parts of a network in which the vertices are densely connected, so-called *clusters*. The motivation is that the computation of clusters in a complex **network** will reveal *functional modules* of the complex **system** [93, 188, 201]. Again, numerous measures and algorithms have been introduced to find and evaluate these clusters, all under different assumptions [89]. The most important distinction is between those algorithms that allow [62, 188] or disallow [93, 204, 203] multiple memberships of vertices.

Also here, a computation of clusters can be done on almost any graph with any of the algorithms, but the result may not be reasonable, i.e., the retransformation of the result to a significant answer about the structure of the complex system may not be successful. It should be clear that such an algorithm only makes sense if edges in a network are more likely between those objects that are indeed similar to each other, or fulfill a similar function in the complex system at hand. It is not helpful if the relationship that is represented by the network connects more or less arbitrary entities.

These two examples show that network analysis algorithms can often yield correct results as specified by the problem, but still they do not give a reasonable answer to the question that was posed about the whole system. So, the problem comes from the transformation of a question about a complex system into a mathematically and computationally tractable question about the complex network representing it, and the problematic retransformation of the solution back onto the complex system level (s. Fig. 2.2). Thus, without the context as defined by the whole system the results given by an contextual algorithm are undefined on the higher level. Of course, one could argue that this *contextual information* of where an analytical algorithm is applicable is something outside of the algorithm itself. But we think that, in the days of algorithm libraries and ready–made network analysis tools, the *appropriate context* is necessary information that should accompany every algorithm and thus should be considered a (facultative) *property of algorithms.*

2.4 Overview

From the landscape of possible questions opened by complex network science, we were most intrigued by the interplay of local behavior and global properties of complex networks. As already sketched above, the first cornerstone of complex network science is the paper by Watts and Strogatz on small–worlds. Their model of real–world networks is comprised of two different components, a so–called *local* graph family and a random or *global* part. As we have argued, an understanding of a network's structure and a reliable, and yet mathematically tractable model is most important for finding new algorithms for real–world applications. It turned out that this first article on small–worlds was by no means the only reasonable small–world model, and subsequently at least five others have been proposed to explain the special combination of structural properties seen in real–world networks. Thus, after giving the necessary graph theoretic definitions in chapter 3, we will discuss in chapter 4 the common ground of these models, the so-called *small–world phenomenon* which will then be rigorously defined. We furthermore present a new family of hybrid graph models that show this phenomenon and that can be flexibly tuned to design networks with desired properties.

As we have argued above, the process by which a network is generated defines the *context* for some of the analytical algorithms. If the small–world network model is correct, and real–world networks are at least partly formed by random edges, it is important to bound the maximal number of random edges in a given graph to decide, e.g., whether the application of a clustering algorithm will give reasonable results or not. In chapter 5 we will discuss the general idea of *network–generating systems* and their inbuilt *network–generating processes* that have to be understood as the *context* of a complex network. We will then show how to bound the amount of random edges in complex networks by a new structural measure, the so-called *backbone distance distribution.* It turned out that indeed this distribution distinguishes random graphs from real–world networks and that networks with a steep backbone distance distribution exhibit a markedly different behavior than those with a shallow distribution. However, although it is somehow intuitive that this steep backbone distance distribution is an indicator of a network–generating process that prefers to build local edges, it is actually hard to show a strong correlation between this structure and the proposed generating process, as with other measures that were proposed to measure locality.

However, it turned out that minimizing the sum of backbone distances $Q(T)$ is closely related to an old problem in graph theory, namely that of finding a *minimal length fundamental cycle base*

[61], which has applications in some optimization problems. We will first show that this problem is NP-hard, and then present some heuristics for computing good distributions. We will show empirically that in real–world networks $Q(T)$ is quite near to a simple lower bound, which can be interpreted as an explanation for why some algorithms are easier to compute on real–world networks. We will finish this chapter with a short description of a graph drawing application for drawing large and complex networks.

Whereas chapter 5 on network–generating processes is concerned with how much it is possible to differentiate between local and random edges, chapter 6 is concerned with the question: if a system consists of multiple, independent entities that are selfish, myopic, and prefer to build local edges, how can we design a network–generating process such that the evolution of the network leads to a desired global structure? Similar questions have already been posed and partly answered by related fields of research in game theory or evolutionary computing. Our particular perspective is first on finding a general framework in which these processes can be simulated and analyzed, and second, on analyzing how sensitive such a complex system is with respect to the rules. We can show that indeed two very similar rules that will eventually lead to the same network structure are very different with respect to their expected runtimes, an aspect that, to our knowledge, has not been discussed before. Although this analysis is only done on a toy example it shows that network design in a complex system is a fundamentally different task than in a classical setting, with its own questions and challenges.

A second problem we could solve by an evolutionary rewiring rule is the one that was posed by the article of Albert et al. on the robustness and sensitivity of scale-free networks against random failures and attacks [6]. As stated above, their finding was that scale-free networks will decompose much faster than corresponding random graphs if highly connected vertices are attacked and removed from the system, whereas random graphs turn out to be more sensitive in the case of random failures. It seems that a network must switch its topology in order to be robust in each scenario. This is, of course, a problem because centrally organized networks often have large edge–building and maintenance costs, and decentrally organized networks, e.g., those based on the Internet, are unaware of the scenario they are in because they lack the overview to decide whether vertices are attacked or just fail at random. We will show that in the latter case, where edge–building costs are low and locality is only required to keep the number of exchanged messages low, it is possible to give the vertices a reaction rule that will—without knowing it—switch the network's topology according to the situation.

It is clear that neither the question of how to detect certain network–generating processes in the resulting network, nor the question of how to design efficient evolutionary rules for complex network design can be fully answered in this thesis. Thus, this work will finish with a discussion in chapter 7 of the accomplished results and a list of open problems that we think are interesting for future research.

3. DEFINITIONS

3.1 Sets

Let $S = \{e_1, e_2, \ldots\}$ denote a set of elements. The *cardinality* $|S|$ of a set gives the number of elements in this set. $\mathcal{P}(S)$ denotes the set of all possible subsets of S.

3.2 Graphs

A *graph* G is a pair (V, E) of a set of *vertices* V and a set of *edges* $E \subseteq V \times V$. In cases of ambiguity, we will denote the edge set of G by $E(G)$, and the vertex set of G by $V(G)$. An edge $e = (v, w)$ is said to be *incident* to its *endpoints* v, w and the connected vertices are said to be *adjacent*. A graph can be *weighted* where the weight of an edge is given by a function $\omega : E \to \mathbb{R}$. For compatibility, *unweighted* graphs are often assigned a pseudo-weight function $\omega_1 : E \to 1$, i.e., every edge has weight 1. If all edges of a graph are given as unordered pairs of vertices, the graph is said to be *undirected*, otherwise it is *directed*. We will denote undirected edges e by pairs of vertices in simple brackets (v, w) and directed edges by $(v \to w)$.

3.2.1 Directed Graphs

Let $e = (v \to w)$ be a directed edge, then v is the *source* and w the *target* of this edge. The *out-degree* $deg_o(v)$ of a vertex is defined as the number of directed edges where v is the source, and the *in-degree* $deg_i(v)$ is defined as the number of directed edges where v is the target. The *degree* $deg(v)$ is defined as the number of all edges incident to v, i.e., $deg(v) = deg_o(v) + deg_i(v)$.

3.2.2 Induced Subgraphs

A *subgraph* $G' = (V', E')$ of G is any graph with $V' \subseteq V$ and $E' \subseteq E$, in short denoted by $G' \subseteq G$. A subgraph G' is an *induced subgraph* if, for a given vertex set V', E' is given by:

$$e' = (v, w) \in E' \Leftrightarrow (v, w) \in E \land v, w \in V' \tag{3.1}$$

To simplify notations, we will denote by $G - e$ the graph that results from removing e from $E(G)$, and by $G - v$ the graph that results from removing v from $V(G)$ and all its incident edges from $E(G)$. Analogously, $G + v$ denotes the graph that results by adding v to $V(G)$, and $G + e$ denotes the graph that results by adding an edge $e = (v, w)$ between vertices v, w in $V(G)$.

3.2.3 Set of all Graphs

Let $\mathcal{G}$ denote the set of all possible graphs, and let $\mathcal{G}(n, m)$ denote the subset of all graphs with n vertices and m edges.

3.2.4 Shortest Paths, Distance, and Components

A *path* $P(s,t)$ from vertex s to vertex t is an ordered set of consecutive edges $\{e_1, e_2, \ldots, e_k\} \subseteq E$ with $e_1 = (s, v_1), e_k = (v_{k-1}, t)$ and $e_i = (v_{i-1}, v_i),\ \forall 1 < i < k$. The *length of a path* $l(P(s,t))$ is defined as the sum over the weights of the edges in the path:

$$l(P(s,t)) = \sum_{e \in P(s,t)} \omega(e). \tag{3.2}$$

A path $P(s,t)$ is a *shortest path* between s and t if it has minimal length of all possible paths between s and t. The *distance* $d(s,t)$ between s and t is defined as the length of a shortest path between them. If there is no path between any two vertices, their distance is ∞ by definition. A graph is *connected* if the distance between any two vertices is smaller than ∞. A subgraph is a *connected component* if it is connected.

3.2.5 Degree Distribution

One of the structural properties of a graph is its *degree distribution*, i.e., the number of vertices with degree $deg(v) = k$ in dependence of the degree. It is well known that random graphs (s. 3.4.2) have a Poissonian degree distribution [34], with a mean degree of np and standard deviation of $\sqrt{np}$.

Scale-Free Degree Distribution

It was shown independently by Albert et al. [8] and Faloutsos et al. [80] that the Internet and the world wide web's link structure show a scale-free degree distribution, i.e., the probability $P(k)$ of finding a vertex with degree k is proportional to $k^{-\gamma}$.

3.3 Partition

A *partition* $C = \{C_1, C_2, \ldots, C_k\}$ of a graph G consists of subsets $C_i \subseteq V$ of vertices of G where $C_i \cap C_j = \emptyset$ for all $i \neq j$ and $\cup_{i=\{1,2,\ldots,k\}} C_i = V$.

A partition C' is called a *refinement* of a partition C if for all components C'_i in C' there is a component C_j in C such that $C'_i \subseteq C_j$, and if for at least one component C'_i in C' there exists a component C_j in C such that $C'_i \subset C_j$.

3.4 Graph Families

A *graph family* $\mathcal{G}_A(n, \Pi)$ is a set or graph defined by some algorithm A that gives a description to construct graphs for every given n and - if needed - an additional set of parameters Π. If A

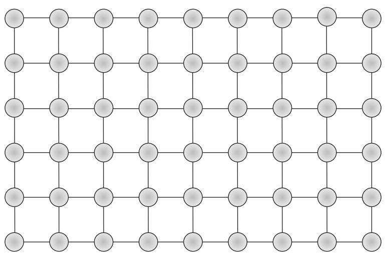


Fig. 3.1: A grid graph with $n = 6$, $m = 9$, $d = 2$, and $r = 1$.

is a *deterministic algorithm* it constructs a single graph, if it is a *non-deterministic* or *stochastic algorithm* it constructs all graphs with n vertices (and maybe additional parameters specified by Π) with a determined probability. The instance that is created from some defined graph family $\mathcal{G}_A(n, \Pi)$ is denoted by $G_A(n, \Pi)$. For graph families, we will often state expected properties with the words: *with high probability*, denoting that any instance $G_A(n, \Pi)$ constructed by A will show property X with a probability higher than $n - 1/n$.

3.4.1 Grid Graphs

As an example of both categories, consider first the family of grid graph $G_d(n, m, r)$ and second the family of random graphs $G_R(n, p)$. A *grid graph* $G_d(\vec{n}, r)$ is a graph with $\vec{n} = \{n_1, n_2, \ldots, n_d\}$ vertices placed in d dimensions. Let $x_i(v)$ denote the position of v in the ith dimension. Each vertex is now connected to all neighbors with Manhattan distance $\leq r$, where the Manhattan distance $d_M(v, w)$ is defined as

$$d_M(v, w) = \sum_{i=1}^{d} |x_i(v) - x_i(w)|, \tag{3.3}$$

i.e., the distance between v and w if each move can change the position by one in one dimension (Fig. 3.1).

A *cube* $Q_d(n, r)$ is a special grid in which all $n_i = n$.

3.4.2 Random Graphs

A *random graph* $G_R(n, p)$ is a graph with n vertices where every (undirected or directed) edge is element of E with probability p. A different but related algorithm for constructing random graphs is the $G_R(n, m)$ family of random graphs that picks m pairs of vertices and connects them with each other. Most of the time an implementation will try to avoid self-loops and multiple edges. Bollobas states that in the limes $n \to \infty$ all expected properties of both families will be the same [34].

3.4.3 Albert-Barabási-Graphs

The Albert-Barabási network model $G_{AB}(n, m, m_o)$ was proposed to generate networks with a scale-free degree distribution. It is a semi-static network model where in each time step t one vertex v_t is added to the already existing graph by the following rule: Let $d_t(v)$ denote the degree of vertex v at time step t. The new vertex v will attach to exactly m different vertices in G_t where G_t denotes the graph existing in time step t. The probability that v_t attaches to vertex $v_{t'}, t' < t$ is proportional to $deg_t(v_{t'})$, a process called *preferential attachment*.

3.5 Spanning and Minimal Spanning Trees

Given a weighted Graph $G = (V, E)$ with $\omega : E \to \mathbb{R}$ the weight function, a *spanning tree* is a subgraph that contains all vertices but only $n-1$ edges, and a *minimal spanning tree* is a spanning tree with lowest total edge length regarding ω.

3.6 Isomorphism

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs with $|V_1| = |V_2|$ and $|E_1| = |E_2|$. G_1 and G_2 are said to be *isomorphic* if there is a bijective function $\phi : V_1 \to V_2$ that maps the vertices from V_1 to V_2 such that:

$$\forall e = (v, w) \in E_1 \Leftrightarrow (\phi(v), \phi(w)) \in E_2 \tag{3.4}$$

We will indicate that two graphs G_1, G_2 are isomorphic by $G_1 \simeq_I G_2$.

3.6.1 Automorphism

Let $\Pi : V \to V$ denote a permutation of V. Π is called an *automorphism* of graph G if

$$(v, w) \in E \Leftrightarrow (\Pi(v), \Pi(w)) \in E, \forall\ (v, w) \in V \times V. \tag{3.5}$$

3.6.2 Structural Index

A *structural index* [41] is any function $\psi : \mathcal{G} \leftarrow \mathbb{R}$ on G such that:

$$\forall\ G_1, G_2 \in \mathcal{G} \text{ with } G_1 \simeq_I G_2 : \psi(G_1) = \psi(G_2) \tag{3.6}$$

In other words, a structural index does only depend on the structure of the graph, given by the edges. We will extend this definition by including functions on sets of graphs $C = \{G_1, G_2, \ldots\}$: A *structural index on a set C of graphs* is any function $\psi : \mathcal{P}(\mathcal{G}) \leftarrow \mathbb{R}$ that yields the same value for any set C' where one or more of the graphs G_i are replaced by isomorphic graphs.

3.7 Stochastic Processes

3.7.1 Chebyshev's Inequality

Let X denote a random variable, $P(X)$ a probability distribution, $E(X)$ the *mean* of this random variable with the given probability distribution, and $\sigma(X)$ the standard deviation. Then

Chebyshev's inequality states that:

$$P(|X - E(X)| \geq t \leq \sigma^2/t^2. \tag{3.7}$$

3.7.2 Chernoff's Inequality

Let $X_1, X_2, \ldots, X_n$ be independent Poisson trials, i.e., every random variable X_i has probability p_i of being 1 and probability $1 - p_i$ of being 0. Let now X be a random variable that sums over X_i, i.e., $X = \sum_i X_i$. The expectation $E[X] = \mu$ of X is given by $\sum_i p_i$. Now, for any $\delta > 0$, the following statement holds:

$$Pr[X > (1+\delta)\mu] < \left[\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right]^\mu \tag{3.8}$$

3.8 Complexity

3.8.1 NP-Hardness

The *complexity* of a problem denotes how its runtime depends on the size of the problem. There are two large classes of problems, P and NP. For problems in P there are known algorithms such that the solution can be computed in polynomial time in the size of the problem. For problems in NP it is possible to check in polynomial time whether a given answer to the problems is actually a solution, i.e., if an answer is produced non-deterministically it is possible to check it in polynomial time. If a problem is at least as hard to solve as all the problems in NP, it is said to be *NP-hard*, if it is in NP, it is said to be *NP-complete*. It is yet undecided whether there are efficient algorithms for the problems in NP, but if only one efficient algorithm for any NP-complete problem is found, all of them can be solved efficiently.

Often a problem P_i is shown to be NP-hard by reducing another NP-hard problem P_j to it, i.e., by defining a polynomial function that transforms P_j into a problem P' of the form P_i such that if P' is solved, also P_j is solved. If the transforming algorithm is itself in P, and there were an efficient algorithm for solving P_i, then P_i would also be in P, in contradiction to the fact that it is an NP-hard problem.

4. THE SMALL–WORLD PHENOMENON

The first real-world network model, the small-world network model introduced in 1998 by Duncan J. Watts and Steven H. Strogatz raised much interest and was followed by a large number of others that were also termed *small-world* network models. These were based partly on different definitions of the term *small world* which triggered our curiosity in the question: "What makes a network model a small-world network model?". We will start this chapter with a review of the classic small-world network model by Watts and Strogatz in 4.1. In 4.2 we will then discuss some of the most important small–world network models that were published later, and analyze the properties common to all of them. Our finding is that all classic small–world network models show the so-called *small–world phenomenon*, the drastic decrease of the average distance in hybrid graphs built out of a graph family with a large diameter and a sparse random graph. This work resulted in a general framework for hybrid graphs showing the small–world phenomenon which is presented in 4.3[1]. The framework, its relation to other models, and possible applications are then discussed in 4.4.

4.1 The Classic Small–World Network Model

To find a general definition that includes all classic small-world network models, we will start with a presentation of the first small-world network model that was introduced by Watts and Strogatz in a seminal paper in 1998 [242, 239].

In this paper, Watts and Strogatz analyzed different kinds of real-world networks with respect to the *average clustering coefficient* $cc(v)$ and the *average distance* $L(G)$[2]. The clustering coefficient $cc(v)$ of vertex v describes the fraction of neighbors of v that are themselves connected by an edge, formally defined as:

$$cc(v) = \frac{e(v)}{deg(v)(deg(v)-1)/2} \;, \tag{4.1}$$

where $e(v)$ denotes the number of edges between the neighbors of v, i.e.,

$$e(v) = |\{(w_i, w_j) \in E | w_i, w_j \in N(v)\}|, \tag{4.2}$$

with $N(v)$ the set of neighbors of v, i.e.,

$$N(v) = \{w | (v, w) \in E\}. \tag{4.3}$$

Note that the denominator in Equ. 4.1 gives the *possible number of edges between neighbors of v*. The clustering coefficient $cc(G)$ of a graph is now simply defined as the average clustering

[1] This project has been conducted together with Hendrik Post and Michael Kaufmann. Part of the following results have been published as a technical report [150], a conference paper [120], and a journal paper [151].

[2] The authors of [242] called it the *characteristic path length*, but in consistency with standard graph theoretic terms we will call it the *average distance* here.

coefficient of its vertices:

$$cc(G) = \frac{1}{n}\sum_{v\in V} cc(v) \tag{4.4}$$

The average distance $L(G)$ is defined as:

$$L(G) = \frac{1}{n(n-1)}\sum_{s\in V}\sum_{t\in V-s} d(s,t). \tag{4.5}$$

These structural measures were chosen by Watts and Strogatz to show that real-world networks have a special structure that is not captured by any of the classic graph families. In general, to show that some kind of data has a special structure it is necessary to compare it with an unstructured instance that is otherwise comparable, e.g., in size. Since random graphs from the $\mathcal{G}(n,p)$ model show no bias towards any special structure, Watts and Strogatz chose them to compare their real-world networks with. A given real-world network with n vertices and m edges is compared with a random graph from the $G(n,p)$ model (3.4.2) with n vertices where p is set to $m/(n(n-1))$ such that the expected number of edges in the instance is m. Henceforth, we will call such a random graph a *corresponding* random graph. Random graphs belong to a well analyzed family of graphs and it is easy to show that the clustering coefficient of such a graph is expectedly p, and Bollobás has shown that the average distance is bound from above by $O(n\log n)$ (s. Theorem 4.2) [34][3].

Surprisingly, in those real-world networks that Watts and Strogatz examined the average clustering coefficient was up to $1,000$ times higher than in the corresponding random graph while the average distances were comparable. Since pure random graph models, i.e., $\mathcal{G}(n,m)$ and $\mathcal{G}(n,p)$, fail to show this combination of a high average clustering coefficient together with a small average distance, the authors introduced a new random model, the so-called *small-world* model. It was designed to scale between the two extremes of total *regularity*[4] and total *randomness* because these were the two opposing extremes defining the range into which all networks seemed to fall, as the authors state:

> Ordinarily, the connection topology is assumed to be either completely regular or completely random. But many biological, technological, and social networks lie somewhere between these two extremes. Here, we explore simple models of networks that can be tuned through this middle ground: regular networks 'rewired' to introduce increasing amounts of disorder. [242]

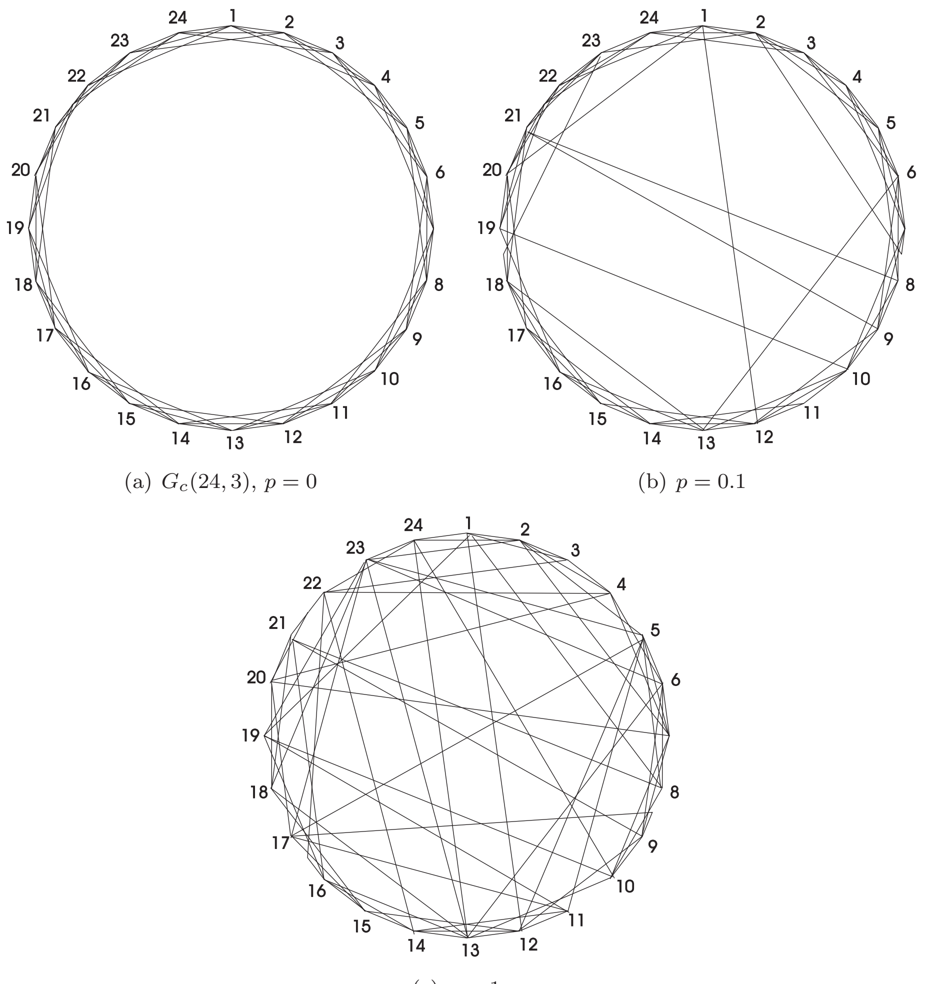
The model starts with a *regular graph family* called *circulant graphs* [102].

Definition 4.1 (Circulant Graph)
A *circulant graph* $G_c(n,k),\ k \leq n$ consists of n vertices, labeled 1 to n, and every vertex i is connected to every vertex j where $(i+z) \bmod n = j,\ z \in \{1,\ldots,k\}$, i.e., for $G_c(n,1)$ the circulant graph is just a ring of n vertices, while $G_c(n,n)$ describes a clique.

In other words, a circulant graph described by the parameters n and k can be constructed by positioning n vertices on a ring where every vertex is connected with its clockwise and counter-clockwise k next neighbors. A sketch of $G_c(24,3)$ is shown in Fig. 4.1(a).

[3] This theorem actually bounds the *maximal distance* in a graph, the so-called *diameter*, which is also an upper bound of the *average distance* in the same graph.

[4] Note that in this case we use the term *regularity* as Watts and Strogatz have used it, but that it has a well-defined meaning in graph theory that cannot possibly be meant by the authors. We will discuss this problem, and the meaning and importance of *regularity* for small-world network models in 4.2.3.



(a) $G_c(24, 3)$, $p = 0$

(b) $p = 0.1$

(c) $p = 1$

Fig. 4.1: **a)** A circulant graph of 24 vertices where each vertex is connected to its three nearest neighbors. **b)** Each edge has been rewired with probability $p = 0.1$. **c)** Each edge has been rewired with $p = 1.0$.

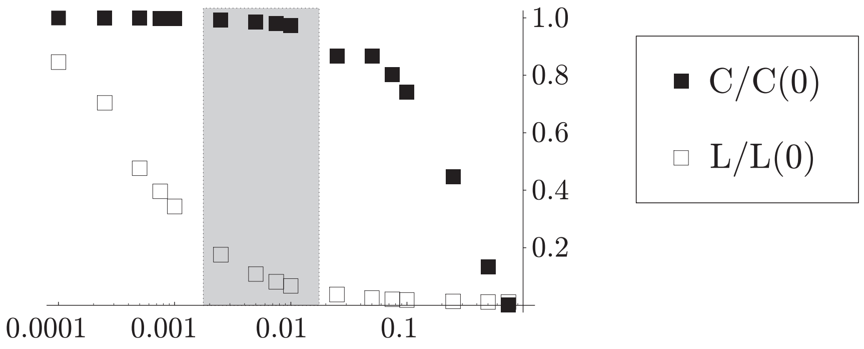


Fig. 4.2: Shown is the average clustering coefficient (denoted by C here) and the average distance (L) in the $\mathcal{G}_{SW}(n,k,p_{rew})$ model, in dependence of p_{rew} and normalized by the resulting value for $p = 0$ ($C(0)$, $L(0)$). The average clustering coefficient is quite stable as long as not too many edges are rewired, while the average distance drops very fast, even at very low values of p_{rew}.

It can be seen quite easily that for $\lim n, k \to \infty$, the average clustering coefficient approaches 3/4 while the diameter and the average distance between vertices in such a graph is in $\Theta(n/k)$, i.e., growing linearly with the number of vertices for fixed k. Thus, this type of graph family also does not show the wanted combination of a high average clustering coefficient and a low average distance. Even more, its structural properties are directly opposed to that of a random graph. By an additional *rewiring* step this circulant graph can be transformed into a special kind of random graph. This rewiring step is governed by a *rewiring probability* p_{rew} which determines the probability that a certain edge will be rewired, i.e., whether the edge will be removed and the source of the edge be connected to a new, randomly chosen target vertex[5]. Figs. 4.1(b) and 4.1(c) show the result of a rewiring step, once with $p_{rew} = 0.1$ and once with $p_{rew} = 1.0$. We will denote the small-world random graph family by $\mathcal{G}_{SW}(n,k,p_{rew})$, and any instance from it by $G_{SW}(n,k,p_{rew})$. By this description it is clear that $\mathcal{G}_{SW}(n,k,0)$ is just the same as a circulant graph $G(n,k)$, and $\mathcal{G}_{SW}(n,k,1.0)$ describes a special kind of random graph family. Note that for any circulant graph $G_c(n,k)$ and any small–world network based on it a corresponding random graph is one with $p = (2nk)/(n(n-1))$ and n vertices.

In this model there is an interval of values for p_{rew} such that the resulting graph $G_{SW}(n,k,p_{rew})$ has a high average clustering coefficient $cc(G_{SW}(n,k,p_{rew}))$ that is much higher than that in the corresponding random graph $G(n, 2nk/(n^2-n))$, and a small average distance $L(G_{SW}(n,k,p_{rew}))$; this interval is shaded grey in Fig. 4.2. Note that in this figure, the values are shown in dependence of p_{rew}, normalized by the value for $p_{rew} = 0$. The networks resulting in this interval where the average clustering coefficient of the resulting graph is much higher than that of a corresponding random graph but the average distance is comparable, i.e., $cc(G_{SW}(n,k,p_{rew})) >> cc(G(n, 2k/(n-1))$ and $L(G_{SW}(n,k,p_{rew})) \simeq L(G(n, 2nk/(n^2-n)))$, are termed *small-world networks*. Note, however, that there is no strict definition of this interval.

We were mostly intrigued by the idea that a network with a large diameter is *compacted* by just a few random edges, and wanted to concentrate on the question of what kind of graph families

[5] Actually, the algorithmic description is too short in [242] to understand whether the edge is totally rewired (new source, new target), or whether only a new target vertex is chosen. The figures and remarks in a follow-up article by Newman and Watts ([175] and [181] (p. 290/1)) indicate that the latter procedure was chosen. Note that this procedure cannot produce the same set of graphs as the $G(n,p)$ family.

would show this *small-world phenomenon*, and how many random edges are needed in dependence of the structure of the underlying graph family to compact the graph. The answer to this question can then be used to build a *general framework for networks showing the small-world phenomenon.*

For this task we first want to define more strictly the terms *small–world network model* and *small–world phenomenon.* Although the first definition of a small-world, or a graph with the small-world effect, was not strict, we think that it is possible to deduce such a definition by examining the many other small-world network models that were proposed and accepted by the scientific community. Thus, we will start this task with a very rare approach in computer science, namely a hermeneutic analysis of texts and models to abstract the common ground of all small-world models.

4.2 Finding a General Definition of Small–World Network Models: The Dependency Between Regularity, Locality, and a High Clustering Coefficient

The article by Watts and Strogatz triggered a vast amount of research and soon many variants of so-called small–world network models were proposed although what a small–world network model is was not strictly defined. As Newman puts it in his review in 2000: "The precise definition of the "small-world effect" is still a matter of debate ..."[173]. The situation had not changed much by 2004 when Evan reviewed the field: "In terms of networks, [small–worlds] have a relatively large clustering coefficient. ... On the other hand, the distances across the network are small, comparable with those obtained from a random graph and much smaller than any regular network. This is then the definition of a small world network". This sentence is followed by a footnote: "Well, except that some people drop the clustering part of the definition. You just can't win with the nomenclature in this field." [78]. In 2006, Newman, Barabási, and Watts stated: "Watts and Strogatz defined a network to be a *small-world network* if it shows both of the properties described above [low average distance and high average clustering coefficient]". This simple statement was also followed by another footnote: "This expression, however, has been used inconsistently by other authors, probably because of confusion about what it means. It might be natural to assume that a "small-world network" would be one that shows the small-world effect, and indeed some authors use the term in this way. Some other authors use it to mean specifically networks taking the form of the Watts-Strogatz model. One should be careful, therefore, in reading the literature on this subject; the use of the term "small-world network" sometimes but not always implies high clustering, and may or may not refer to a specific model." [181].

The question we are trying to answer here is what makes a network model a small–world network model. Are *regularity* and a *high clustering coefficient* necessary ingredients of the small-world phenomenon? Our method is as follows: If many of the accepted small-world models rely on some structural property such as *regularity* we will assume that it is a necessary ingredient; if some or many discard it, we assume that it is not a salient feature of the small-world effect. We start with a discussion of the first paper on small–worlds by Stanley Milgram.

4.2.1 Small–Worlds are Surprisingly Small

The seminal experiments that introduced the term *small world* to describe social networks were conducted by the famous psychologist Stanley Milgram who tested how fast information can flow in human acquaintance networks [165, 231]. He discovered that the number of steps until certain information (in this case, a letter) arrived at a target person thousands of miles away from the starting point was surprisingly small[6]. This fact is *surprising* because people are mainly connected to other people who live not too far away, i.e., connections are mainly *local*. In a purely local setting, where there is strictly no connection between people that live at more than a given distance, the length of a chain should be proportional to the square root of the geographical difference between source and target person, which was not the case. The term small-world was used here to describe the surprising fact that a network that is dominated by *local edges* can be much smaller than implied by its local nature. We will see later that this *surprise* is a salient feature of small–world network models.

This finding of Milgram et al. was the starting point of Watts' and Strogatz' article, and their small–world networks are supposed to model the real-world situation. Although smallness is a required property of a small–world network, most authors argue that it is not sufficient to make a network model a small–world network model, as we will discuss in the following paragraph.

4.2.2 Are All Small Networks Small–Worlds?

An important question to answer is whether there are graph families with a small average distance that are not accepted as small–world network models, e.g., whether a pure random graph should be considered a small–world network model. In the light of the article of Watts and Strogatz this point is clearly rejected, but it is debatable as Newman states:

> The precise definition of "small-world effect" is still a matter of debate, but in the present case a reasonable definition would be that l [the average distance] should be comparable with the value it would have on the random graph ... [173]

This statement is reinforced in the latest book by Newman, Barabási, and Watts, where the authors state that the random graph model does show the *small-world effect* [181]. But they also state that a small-world model is one that shows the small–world effect **and** high clustering. Thus, small-worlds and random graph models are put in opposition to each other and it is reasonable to exclude random graphs from a (yet to be given) small-world definition.

Summarizing, we conclude that the small–world **effect**, defined as cited above, is a necessary but not a sufficient ingredient for most authors to make a network model a small–world network model. The other part of the definition by Watts and Strogatz turns out to be the part that is much harder to define formally: a small–world network is *clustered*. Since we know that their model was supposed to model real-world social networks that are dominated by local relationships, it is reasonable that the clustering coefficient is supposed to measure locality. On the other hand, Watts and Strogatz introduced the clustering coefficient to measure the *regularity* and *cliquishness* of the network. We will now discuss whether *regularity* is necessary for a small-world network model.

4.2.3 Regularity

In graph theory, a graph is *regular* if every vertex has the same degree. It is unlikely that this kind of regularity was meant by Watts and Strogatz, since the clustering coefficient does not measure

[6] The first experiment was quite small, and some say too small to really show that any two people are separated by the well-known *six degrees of separation* [128]. The second study with Travers was based on a larger sample and also here, those chains that reached the target had only 5.2 intermediaries. Still, both experiments might underestimate the real length because they do not consider chains that stop without reaching the target.

this kind of regularity. Another, intuitive meaning of regularity is that every vertex could take the place of every other vertex in the graph. Circulant graphs show different kinds of regularity in the sense of 'structural isomorphisms'. Two of them are the so-called *edge* and *vertex transitivity*, i.e., for any pair of edges a and b (vertices v and w) there is an automorphism (s. 3.6.1) that maps a to b (v to w). If that is the case, the edges (vertices) of a graph are virtually indistinguishable by the adjacency matrix alone, and the graph is regular in the sense that different kinds of mapping operations yield the same graph. Of course, in this latter sense, also random graphs are (at least expectedly) regular, because the expected neighborhood, i.e., the number of vertices in distance k to any given vertex v is exactly the same for all vertices [34]. In any case, the *clustering coefficient* is not able to measure either kind of *regularity* since it is a local measure, but any of these definitions of regularity require a bird's eye view of the graph. Although the small–world network model does scale between the very regular, in the sense of 'structured', circulant graphs and a non-structured randomized network, the clustering coefficient is not able to measure this, and thus we conclude that regularity in the sense of edge or vertex transitivity is not a needed property of small–world network models. This leaves open the question of what is measured by the clustering coefficient and whether a high clustering coefficient is needed for a small–world network model.

4.2.4 The Clustering Coefficient

Many authors of small–world network models consider a high average clustering coefficient a salient feature of a small-world network model [68, 174, 242] while others base their models on graphs that do not necessarily have a high clustering coefficient:

1. Kleinberg used d-dimensional grids with a clustering coefficient of 0 as the basis for his small-world models [126, 125]. On top of these grids, edges $e = (v, w)$ are added with a probability $P(e)$ proportional to $d(e)^{-\alpha}$, where $\alpha > 0$ is some constant and $d(e)$ denotes the distance between v and w in the grid. If the added random graph is sparse, the clustering coefficient of this model tends to 0.

2. Andersen, Chung, and Lu consider a more general concept of clustering by composing a *local* and a *global* (i.e., random graph) into a so-called *hybrid graph* [11, 53]. They offer two different definitions of locality, one based on network flow [11] and another based on counting the number of edge-disjoint paths up to a given length between pairs of vertices [53]. A network is then said to be (k, l)-local if for every pair of vertices connected by an edge there are at least k edge-disjoint paths of length at most l between these vertices. Both of these local graph families do not need to show a high clustering coefficient; they rather generalize the idea of the clustering coefficient to the one of *locality*.

Since all of these models were accepted by the scientific community it seems that a high average clustering coefficient indicates a desired feature of the network, but it is not required itself. As indicated by the many authors we have already cited, the general idea of the clustering coefficient is to measure *locality*. As Newman points out, most social relationships are between people that are somewhat near each other:

> ... most people are friends with their immediate neighbors - neighbors on the same street, people that they work with, people that their friends introduce them to ...[173]

and Watts and Strogatz themselves state that the clustering coefficient was designed to measure the 'cliquishness of a typical neighborhood (a local property)' [242].

We will thus now first discuss how locality in a graph can be defined and then analyze the relationship between the clustering coefficient and locality.

Degrees of Locality

Intuitively, a graph is local if there is a notion of distance between vertices and if the probability that an edge exists is higher the shorter the distance it spans. We will call a network that is generated by a process that prefers short, i.e., local edges, a *local* network. Of course there are different degrees to which local edges can be preferred by the network generating process. In the following we will formally define the notion of *strongly* and *weakly* local graphs.

Given a metric space that defines the distance $\overline{vw}$ between any two vertices v, w and a distribution of vertices in this space, we assume that there is a network generating process that assigns a probability to each edge to be realized. The graph resulting from such a network generating process can be categorized as weakly or strongly local according to the following definitions:

Definition 4.2

1. A graph is called *strongly local* if – given any edge of the graph – it is more or at least as likely that the connected vertices are in distance d than in distance d' for all distances $d < d'$. A special case of a strongly local network generating relationship is given in a *k-next neighborhood graph* where every vertex v is connected to its k_v next vertices. This is the case for circulant graphs (s. Def. 4.1) on which the original small-world model was based, and for (multidimensional) grids later models were based, e.g., [126].

2. A graph is called *weakly local* if two vertices in distance d are more likely to be connected than two vertices in distance d' for all distances $d < d'$.

The difference between the two definitions is explained by introducing two new measures, namely the *absolute* and *relative distance distribution*: For most distributions of vertices in a metric space, there will be many more vertices in distance d' than in distance $d < d'$ for any given vertex v. For example, if infinitely many vertices are placed equidistantly in a two-dimensional space, there are $4 \cdot d$ vertices in distance d of any vertex v. Let $\#_p(d)$ denote the number of *possible edges*, i.e., the number of pairs of vertices in distance d, and let $\#_r(d)$ denote the number of *realized edges* spanning distance d that exist in a given graph. The graph and its network generating process are said to be *strongly local* if there are (expectedly) absolutely more edges between vertices in distance $d < d'$ than in distance d' for all d, i.e., if the *absolute distance distribution* given by the values of $\#_r(d)$ for all d is monotonically non-increasing. A network generating process is *weakly local* if the probability that two vertices are connected increases as the distance between them decreases, i.e., if the *relative distance distribution* defined as $\#_r(d)/\#_p(d)$ is monotonically non-increasing for all d[7]. If the number of edges in greater distances is monotonically non-decreasing, every strongly local graph is also weakly local, but not vice versa.

In the following section we will show that the principle of locality, i.e., the preference to build local edges, is prevalent in many real–world networks.

Local Graphs

It has been shown for some real-world networks where geometrical positions of the vertices are known that indeed local edges are preferred:

[7] For a real–world network where every distance d might occur only a few times it is of course necessary to bin the edges linearly over the whole range.

1. Gastner and Newman could show that in the design of commuter transport networks and sewage systems an intricate balance between the total geometric edge length and the travel time from any vertex to a center vertex along the network paths is achieved [90]. Moreover, in the networks they analyzed the total geometric edge length came close to the total edge length in the *minimal spanning tree* (s. 3.5) of the network, implying that every vertex prefers to be attached by its shortest edge to the growing network.
2. A second example is given by Frenken and van Oort who reviewed literature on the 'geometry of innovation' [88]. They summarize the findings described in the literature, and state that knowledge production and innovation are mainly achieved by groups whose members live in the same region. Additionally, they conducted a co-authorship analysis with respect to the affiliations of authors where two co-authors are considered to have a 'regional' cooperation if their affiliation lies in the same state. With this technique, they analyzed publications in two quite different scientific fields, namely 'aerospace engineering' and 'biotechnology and applied microbiology'. Their result is that scientific cooperations tend to be regional, although the trend has decreased in the last years due to cheaper communication, and that collaborations between academic and non-academic groups are more often regional than pure academic research.
3. Another interesting finding has been achieved by Yook, Jeong, and Barabási on the locality of the Internet, described on the level of routers and autonomous systems (AS) [254]. The authors used data collected by Govindan and Tangmunarunkit that mapped AS addresses to physical locations [99], and measured the probability $P(e)$ that edge e exists as a function of e's geometrical length $d(e)$. Their results clearly show that $P(e)$ is proportional to $1/d$, a result that is explained by the costs of installing a physical link between routers that can be assumed to be mainly growing linearly with its length.

For many other real-world networks that exist between vertices with a fixed position it can be assumed with high certainty that most edges are local if the costs of building an edge are proportional to the distance between the vertices, as for wires, tracks, streets, and also social relationships, although to a lesser extent as anyone can verify from his or her own acquaintanceship network. In the cases discussed so far, there was an explicitly given distance function between the vertices and in these cases it is easy to check whether a graph is strongly or weakly local. In other networks, it can be assumed that edges are more probable between somewhat 'near' or 'similar' objects, i.e., that an edge between two objects indicates that they are similar. Given a graph, it would be very helpful to be able to decide whether there is an embedding of the vertices—and thus a distance function—such that the edge set is strongly or at least weakly local, because only if the network generating process is local in this sense, can network analysis algorithms, especially clustering algorithms, be meaningfully applied to these networks. To understand the importance of this we will first discuss some of these networks where no explicit distance function between the objects is known and the reason we need to understand their structure, and then discuss whether the clustering coefficient is able to measure locality in a given adjacency matrix.

Probably Local Graphs

Although preference of local edges seems to be settled for the above given networks, there are still classes of interesting networks out there where no distance function between the objects is readily available. We want to illustrate this important point with some examples:

1. A protein-protein interaction network represents the proteins of an organism by vertices, and two vertices are connected by an edge if the represented proteins interact biologically

with each other, as introduced in 2.2. Since proteins often exert their function in the cell in tightly packed conglomerates of different types of proteins, the understanding of protein–protein interaction networks is an important step to understand the time and space dependent functionality of cells. We will now explain why they are considered to be local graphs. It is assumed that these networks have at least partly evolved by duplicating certain parts of the genetic code that encodes proteins. Normally, the genetic code of a functional and vital protein is not allowed to change by much and still maintain its functionality. But if it is duplicated, the genetic code of the copy can be mutated without harming the functionality of the original. Thereby the mutated protein may possibly lose some structural properties and gain others [185]. It might also interact with the original itself. This mechanism has been transformed into a dynamic network model that results in networks that are quite similar to the real ones, a good indication that the model captures the essential network generating process [219, 232]. If this mechanism models the evolution of protein-protein interaction correctly, then it is clear that at least some nearby proteins in the protein-protein interaction network are also similar to each other, e.g., on the level of their amino–acid sequence or their $3D$ structure. Under this model, we can assume a certain degree of locality in these networks. Still, the similarity of proteins is not unambiguously defined; measures range from similarity of the structure, to similarity of the amino-acid sequence they are made of, to the similarity of their function in the cell. This makes it very difficult to position proteins in any metric space or to define a coherent metric distance function between any two proteins such that all of these different similarities are captured.

2. We have a similar problem in metabolic networks. Here, all the small molecules produced by the set of enzymes of an organism are represented by vertices, and two vertices are connected if an enzyme catalyzes the transformation of one molecule into the other [85, 116]. Because the one is made of the other, it is clear that they share at least some structural properties and thus, metabolites with a low distance in this network can also be assumed to be structurally similar. And on the other hand, if two metabolites are very dissimilar than it is unlikely that a small number of enzyme catalyzed steps will transform the one into the other. Nonetheless, here it is also highly difficult to denote a metric distance function that captures the similarity between all pairs of metabolites.

3. Krebs [135] and Clauset [54] discuss co-purchasing networks of books where books sold by Amazon are represented by vertices. On the Amazon sites, every item's page contains links under the title: 'customers who bought this book also bought'. In a co-purchasing network two vertices are connected by a (directed) edge if these links point from the one book to the other. As Clauset has shown in his article, a clustering of these networks reveals subgraphs consisting of very similar books, implying that edges are more likely between similar books. But—as can be seen in the comparison between any two libraries—there is no such thing as a unique categorization of books, and we know of no coherent quantitative measure that has been proposed to judge the similarity of two books.

4. As a last example we want to note the web's link structure [46, 8], modeled by networks where websites are represented as vertices and two vertices are connected by a (directed) edge if the one links to the other. Many algorithms have been proposed to harness this network structure to find those pages that are related to each other and to a given query, and their success is, without a doubt, amazing [44, 124, 153, 187]. We can thus safely assume that in this network those pages that are similar by content are also near each other in the network and that those pages that are near each other in the network are similar[8]. Still, it seems to be

[8] Although there might exist different groups of websites concerning the same topic, e.g., if they are in different languages.

impossible to give a precise, metric distance function that quantifies the semantic similarity between any two websites.

For all of the above examples, human experts are often able to agree on the two most similar out of any given three entities, e.g., for proteins or metabolites, books, or websites. Nonetheless, there is no agreed upon distance function between the entities that would allow us to easily check whether these networks are really local as defined by Definition 4.2. We just get indirect hints about this locality, either because an assumed model gives good results in simulations or a clustering algorithm is able to find dense subgraphs of similar entities. But of course, it would be better to first have the information that a network is local, and later apply a clustering algorithm, especially if similarity of the entities is **deduced** from the fact that they end up in the same cluster, as done, e.g., in [188, 201].

We thus assume that there is a similarity space—that is maybe not metric—in which these entities are positioned and that humans are at least partly able to agree on a partial ordering of the entities based on their similarity. We also assume that in the networks sketched above, edges are much more likely between similar entities than between non-similar entities, i.e., our basic assumption is that these systems also prefer to build local edges—but we cannot prove it because we do not know the position of the vertices in the similarity space.

In summary, locality, in the sense of preferring edges between objects that are somewhat near each other, seems to be an important network generating process. The question is now, given the adjacency matrix of a graph, can we detect whether there is some notion of distance between the vertices and whether the network mainly consists of local edges? Most people would say that this is exactly what the clustering coefficient does (although Watts and Strogatz have never claimed this). We will first discuss some findings where the clustering coefficient is correlated with a local network generating process and then show that, nonetheless, a strong correlation between the clustering coefficient and the idea of locality cannot be found, i.e., that not every strongly local graph has a high clustering coefficient.

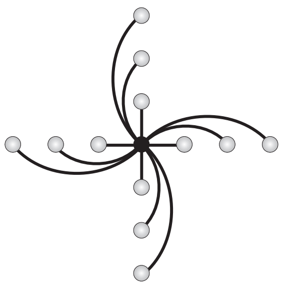
The Correlation Between Locality and the Average Clustering Coefficient

In many settings where vertices are positioned in a metric space and every vertex is connected to its k next neighbors, the average clustering coefficient can be expected to be high:

1. It was already shown that simple circulant graphs have a clustering coefficient of 3/4 in the limes and it is clear that these graphs are strongly local if the graph is embedded on a ring such that every vertex is connected to its k next neighbors.
2. In [173, 174] a d-dimensional grid model is developed in which a vertex at position $\{x_1, x_2, \cdots, x_d\}$ is connected to its k next neighbors in every single dimension i, as sketched for $d = 2$ in Fig. 4.3. With this simple metric, the clustering coefficient is given by [174]:

$$cc(v) = \frac{3(k-2d)}{4(k-d)}, \tag{4.6}$$

   Thus, in the limes of k, the clustering coefficient approaches 3/4 for every number of dimensions d. This kind of graph is also strongly local.
3. A less deterministic, local graph family could also be shown to have a high clustering coefficient: let $G_{ud}(n, r)$ denote the so-called *unit-disk graph* of n vertices that are positioned



(a)

Fig. 4.3: In the model of [174] every vertex is connected to its k next neighbors on each single dimension.

uniformly at random in some 2-dimensional area. Let $d_g(v, w)$ denote the *geometric distance* between any two vertices in this setting. In a *unit-disk graph*, (v, w) is in E if $d_g(v, w) \leq 1$, i.e., a vertex is connected to all vertices that lie within a radius of 1[9]. Since the distance function is symmetric, $(v, w) \in E \Leftrightarrow (w, v) \in E$. In [83] we have shown that a unit-disk graph in $2D$ with a homogeneous distribution of vertices has an average clustering coefficient of 0.58.

On the other hand, d–dimensional grids have a clustering coefficient of 0. Of course, this is only due to the peculiar distribution of vertices in space and a network generating process that stops producing local edges exactly when all vertices are connected to their $2d$ neighbors, i.e., without producing any triangles. Both the distribution and the network generating process are very unrealistic in any real–world network. Thus, we will analyze in the following what happens if every vertex is connected to more than $2d$ neighbors in a d-dimensional grid and we will show that also in this case there is no strong correlation between a high clustering and a local network generating process.

The Clustering Coefficient of Vertices in d-Dimensional Grids

Here we will show that there are strongly local graphs and a metric such that the clustering coefficient approaches $(3/4)^d$ for a fixed dimension d. Let $\vec{x}(v) = \{x_1, x_2, \ldots, x_d\}$ denote the position of v in a d-dimensional grid. For every two vertices, $d_\infty(v, w)$ is given by

$$d_\infty(v, w) = \max\{|x_i(w) - x_i(v)| \; |1 \leq i \leq d\}. \tag{4.7}$$

Let $Q_{d,\infty}(x, k), x \in \mathbb{N}$ denote a *finite, equilateral grid* of $n = x^d$ vertices where every vertex v is connected to all vertices w with $d_\infty(v, w) \leq k$, and let $Q_{d,\infty}(k)$ denote an *infinite grid* in which every vertex v is connected to all vertices w such that $d_\infty(v, w) \leq k$. Note that the use of an infinite grid allows neglecting any boundary effects, which will be discussed later.

We will now determine the clustering coefficient of any vertex v in the infinite grid for a fixed dimension d. To do so, it is enough to concentrate on a subgrid $G_{d,\infty}(2k+1, k)$, i.e., a grid that contains all direct neighbors of one vertex v in this metric. For simplicity, the positions of the vertices in the grid are numbered (i, j), from 1 to $2k + 1$ in every dimension. In the following, we will show that the clustering coefficient of v asymptotically approaches $\left(\frac{3}{4}\right)^d$ for $\lim k \to \infty$.

[9] Such a graph is also called a *random geometric graph* [197]

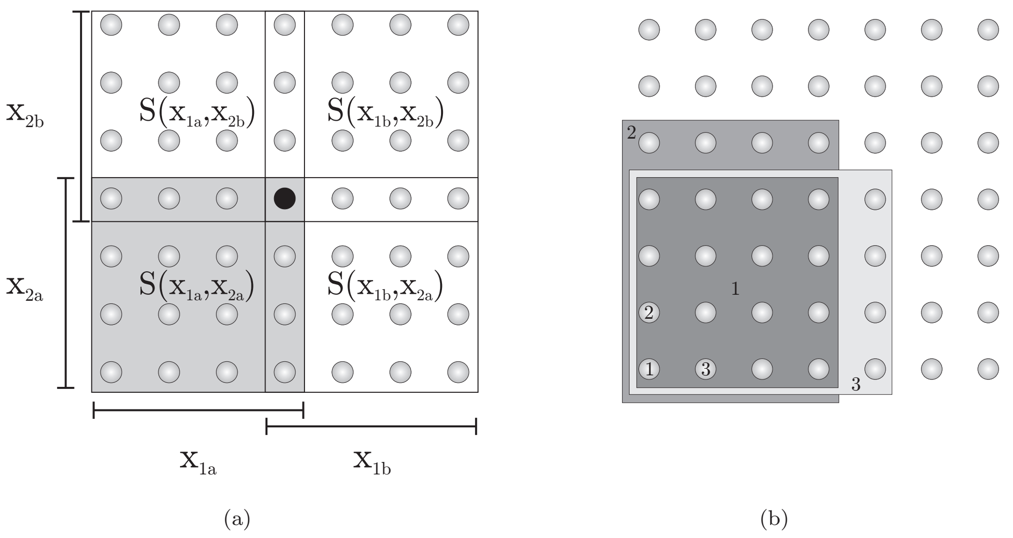


Fig. 4.4: (a) The 2-dimensional grid is divided into 4 subgraphs, each of size $(k+1) \times (k+1)$. (b) The outermost vertex 1 of one subgraph is connected to all vertices in the square of $(k+1) \times (k+1)$ vertices (besides itself). Its direct neighbors $(2,3)$ contribute $(k+2)(k+1)-1$ edges, and in general a vertex at position (i,j) in $S(x_{1a}, x_{2a})$ contributes a degree of $(i+k)(j+k)-1$.

Lemma 4.1
The average clustering coefficient of any vertex in $Q_{d,\infty}(k)$ approaches $(3/4)^d$ for $k \to \infty$.

The special case of $d = 2$ is shown in Fig. 4.4. The degree of the middle vertex is simply given by $(2k+1)^2 - 1$, and by $(2k+1)^d - 1$ for the general case. In the special case of $d = 2$ we divide the grid in every dimension x_i into two intervals x_{ia}, x_{ib} of length $k+1$ that overlap at the middle vertex in that dimension. This results in four subgrids $S(x_{1a}, x_{2a}), S(x_{1a}, x_{2b}), S(x_{1b}, x_{2a}), S(x_{1b}, x_{2b})$ of size $(k+1) \times (k+1)$ for which we now calculate the sum of the degrees of the vertices. To simplify the calculation we will re-number the vertices in each of the subgrids from $(1,1)$ to $(k+1)$. Note that due to this re-numbering vertex $(k+1, j)$ in subgrid $S(x_{1a}, x_{2a})$, is the same vertex as $(1, j)$ in $S(x_{1b}, x_{2a})$, and that v, the vertex at $(k+1, k+1)$ in $S(x_{1a}, x_{2a})$ is the same vertex as $(1, k+1)$ in $S(x_{1b}, x_{2a})$, as $(k+1, 1)$ in $S(x_{1b}, x_{2a})$, and as $(1,1)$ in $S(x_{1b}, x_{2b})$. This observation can be easily generalized for $d > 2$. Let $S(a)$ denote the subgrid $S(x_{1a}, x_{2a}, \ldots, x_{da})$ and let $S(a,i)$ denote the subgrid $S(x_{1a}, \ldots, x_{ib}, \ldots, x_{da})$. Every vertex positioned at $k+1$ in some dimension x_i in subgrid $S(a)$ is also contained in subgrid $S(a,i)$ at position 1 in x_i. Thus, if a vertex is at position $k+1$ in $0 \leq z \leq d$ dimensions in subgrid $S(x_{1a}, \ldots, x_{da})$ it is also contained in $2^z - 1$ other subgrids.

Coming back to the special case of $d = 2$, it is easy to see that every vertex (i,j) in $S(a)$ has edges to all other vertices in a subgrid of size $(i+k) \times (j+k) - 1$ (1 has to be subtracted since the vertex is not connected to itself). Thus, the sum of the degrees of all vertices in subgrid $S(a)$ is given by:

$$\sum_{w \in S(a)} deg(w) = \sum_{i=1}^{k+1} \sum_{j=1}^{k+1} (i+k)(j+k) - 1 \tag{4.8}$$

$$= \sum_{i=1}^{k+1} (i+k) \left((k+1)k + \frac{(k+2)(k+1)}{2} \right) - (k+1)^2 \tag{4.9}$$

$$= \left(\frac{3}{2}k^2 + \frac{5}{2}k + \frac{3}{2}\right)^2 - (k+1)^2 \tag{4.10}$$

$$= \left(\frac{9}{4}k^4 + \frac{15}{2}k^3 + \frac{39}{4}k^2 + \frac{11}{2}k - \frac{5}{4}\right) \tag{4.11}$$

The sum of the degrees of all vertices $w \in V$ in the whole grid is now obtained by multiplying $\sum_{v \in S(a)} deg(v)$ by the number of subgrids (2^2) and subtracting the correct number of degrees of those vertices that were count multiply. Note that if vertex w is at position $k+1$ in $z \in 0,1$ dimensions it is count 2^z times and that every dimension contributes a factor of $x_i + k$ to the degree of w. It thus follows that:

$$\begin{aligned} \sum_{w \in V} deg(w) &= 2^2 \sum_{w \in S(a)} deg(w) - \sum_{j=1}^{k} ((2k+1)(j+k) - 1) - 3((2k+1)^2 - 1) && (4.12) \\ &= 4 \sum_{w \in S(a)} deg(w) - 3k^3 - \frac{9}{2}k^2 - \frac{3}{2}k + k + 1 - 12k^2 - 12k && (4.13) \\ &= 9k^4 + 27k^3 + \frac{45}{2}k^2 + \frac{19}{2}k - 4 && (4.14) \end{aligned}$$

Note that by now every edge in the grid is counted twice and that furthermore also the edges between v and every other vertex are still contained in the sum. To calculate $e(v)$, i.e., the number of edges between neighbors of v, the degree of v has to be subtracted twice and the result has to be divided by 2, yielding

$$e(v) = \frac{9k^4 + 27k^3 + \frac{37}{2}k^2 + \frac{11}{2}k - 4}{2}. \tag{4.15}$$

Thus, the clustering coefficient of v is given by:

$$cc(v) = \frac{9k^4 + 27k^3 + \frac{37}{2}k^2 + \frac{11}{2}k - 4}{16k^4 + 32k^3 + 16k^2} \tag{4.16}$$

which asymptotically approaches 9/16 for $\lim k \to \infty$.

Let now $d > 2$ denote some fixed value. Remember that the degree of v is given by $(2k+1)^d - 1 = \sum_{i=1}^{d} \binom{d}{i} (2k)^i$. The denominator of the clustering coefficient of any vertex v in the infinite grid is given by $deg(v)(deg(v) - 1)$, i.e,

$$deg(v)(deg(v) - 1) = \sum_{i=1}^{d} \sum_{j=1}^{d} \binom{d}{i} \binom{d}{j} (2k)^{i+j} - \sum_{i=1}^{d} (2k)^i. \tag{4.17}$$

To compute the numerator, the grid is again sliced into 2^d subgrids by cutting each dimension in halves of length $k+1$, overlapping at the middle vertices. The numerator is then composed of the d sums over each dimension for each of the 2^d subgrids, where each of the dimensions contributes $\sum_{x_i=1}^{k+1} (x_i + k) = \frac{(k+1)(3k+2)}{2}$, i.e., the numerator is dominated by

$$2^d \frac{((k+1)(3k+2))^d}{2^d} - M = \sum_{i=0}^{d} \sum_{j=0}^{d} \binom{d}{i} \binom{d}{j} 2^{d-j} 3^j k^{i+j} - M, \tag{4.18}$$

where M denotes the sum of the degrees that are multiply count. M is given by:

$$M = \sum_{x=0}^{d} (2^x - 1)(2k+1)^x \left(\frac{(k+1)(3k+2)}{2}\right)^{d-x}. \tag{4.19}$$

Vertices in fixed position $k+1$ in x dimensions contribute a degree of $(2k+1)^x \left(\frac{(k+1)(3k+2)}{2}\right)^{d-x}$ and are counted 2^x times. The largest term in M in dependence of k comes from those vertices that are at position $k+1$ in exactly one dimension. This term has an exponent of k^{2d-1}.

Thus, $cc(v)$ is a fraction of two polynomial functions that are differentiable for $d, k > 0$ for all values of k in the interval from $[1, \infty[$, and—for fixed d—the coefficients of every term are fixed. By the rule of l'Hospital, the limes of the fraction of two polynomials for $x \to \infty$ is given by the fraction of the coefficients of the highest terms of both functions. The highest term in the numerator is clearly given by $3^d k^{2d}$, and the highest term of the denominator is $2^{2d}k^{2d}$. It follows that the asymptotic value of $cc(v)$ in a $Q_{d,\infty}(2k+1, k)$ grid is $\left(\frac{3}{4}\right)^d$. Thus, for any fixed dimension d, and $k \to \infty$, the clustering coefficient of vertices in an infinite grid approaches $(3/4)^d$. Note that infinity is not necessary for this argument: it is valid for any vertex with a distance of at least $k+1$ to the border of the grid; such a vertex will be called a *proper vertex.* For any fixed d and k and $n = (xk)^d, x \in \mathbb{N}$, the fraction of proper vertices is given by $((x-2)/x)^d$. Even if the clustering coefficient of non-proper vertices were 1, for any fixed dimension d and any value $cc > (3/4)^d$ there is a k and an x such that for all $Q' = Q_{d,\infty}(x', k')$ with $x' > x$ and $k' > k$ the average clustering coefficient of Q' is smaller than cc, which concludes the proof.

In summary, for any value $cc > 0$ there is a strongly local graph with an average clustering coefficient below this value as stated in the following theorem.

Theorem 4.1
For every value $cc > 0$ there is a finite, equilateral grid $Q_{d,\infty}(x, k)$ with an average clustering coefficient $cc(Q_{d,\infty}(x, k))$ smaller than cc.

Of course, Watts and Strogatz did not only evaluate the absolute clustering coefficient, but also compared it with that of a corresponding random graph. A corresponding random graph has a clustering coefficient of $p \simeq m/(n^2/2)$. We will now show that the relation between the clustering coefficient of a local graph and that of a corresponding random graph can also grow quite slowly. For this, let $Q_{d,\infty*}(x, k)$ denote an equilateral cube with boundary conditions such that vertices are connected if $d_{\infty*}(v, w) \leq k$ with $d_{\infty*}(v, w)$ defined as:

$$d_{\infty*}(v, w) = \max\{j|(x_i(w) + j) \bmod x = x_i(v), 1 \leq i \leq d\}. \tag{4.20}$$

In such a 'wrapped' grid, every vertex is a proper vertex if k is less than or equal to $x/4$[10]. Let now k be exactly $x/4$. For such a grid, $n = x^d$ and $m \simeq n \cdot (x/2)^d/2$ since every vertex is connected to all vertices in a cube with side length $x/2$, implying that $p \simeq 1/2^d$. Thus, the relation between the clustering coefficient and p for any given $Q_{d,\infty*}(x, k)$ is approximately $(3/2)^d$. Although this is an exponentially growing function, the relation between the values will only be around 50 for $d = 10$. Thus, also the relation between the clustering coefficient in the network and its corresponding value in a random graph cannot be strongly correlated with locality in the sense of Definition 4.2.

In summary, if d is low, the clustering coefficient itself may be high, but there are also cases where the clustering coefficient of a corresponding random graph is high, making it impossible to decide, given only the clustering coefficient, whether the network generating process was random or local.

With this we have shown that *locality* of a graph is not in all cases sufficient to yield a high clustering coefficient or at least a clustering coefficient that is much higher than that of a corresponding random graph.

[10] Otherwise, our above made analysis will not hold because the corners of the neighborhood of one vertex will also be connected to each other due to the boundary condition.

Of course, the graphs for which we have shown this effect are still quite artificial, but since we aimed to answer the question of whether a high clustering coefficient is sufficient or necessary to show locality in a graph, it is sufficient to show one counterexample where the network generating process is local, but the clustering coefficient is not high, or at least not high with respect to that in a corresponding random graph.

The question is now whether a graph with a high clustering coefficient is always local, i.e., if there is always an embedding $\mathcal{E}$ in a metric space such that the network–generating relationship is local. This question is difficult to answer in general since an average clustering coefficient does not give much information about the graph. As already mentioned above, a *unit-disk* graph has an expected average clustering coefficient of 0.58 if the vertices are distributed uniformly at random (s. 9.3.2). However, given the adjacency matrix of a graph, it is NP-hard to decide whether the vertices can be embedded and a radius r can be found such that the edges constitute the corresponding *unit-disk graph* with respect to this embedding and this radius [43]. A more relaxed question, namely, whether there is an embedding of the vertices and two radii r, r' such that all vertices within distance r are connected and all vertices with distance larger than r' are not connected and this set of edges equals the given one, is also NP-hard to decide [138]. Similarly, there is a way to construct something like a *minimal spanning tree* (s. 3.5) in a *unit-disk graph*, the so-called *local minimum spanning tree* [11]. We could show that also here it is NP-hard to decide whether a given graph can be embedded such that the set of edges constitutes the *local minimum spanning tree* of that embedding (s. 9.3.4) [63]. Based on these findings, it seems hard to make a general statement whether a given graph is *strongly* or at least *weakly local* given the adjacency matrix, and much less if it is only known that it has a high clustering coefficient.

In summary, not all small–world network models are based on graphs with a high clustering coefficient. Those models that do not explicitly require a high clustering coefficient, nonetheless focus on the idea of *locality.* Still, as we could show here, there is no strong correlation between the concept of *locality* as defined in 4.2 and the clustering coefficient although for low dimensions it is likely that a local graph, such as a k-next neighborhood graph or *unit–disk* graph, results in a high clustering coefficient. What remains is that the so-called local part of all small–world network models has a larger diameter than the resulting small–world network, after a sparse random graph has been added to it.

4.2.5 Properties of Classic Small–World Network Models

The common ground of all the accepted small-world network models is thus given by the following properties:

1. Every one of these models is **composed of two edge sets** on the same set of vertices, where one of them is a **sparse random edge set** that is created either by rewiring a subset of edges or by adding edges with a certain probabilistic scheme.
2. The resulting graph has a strictly lower diameter and average distance than each of the subgraphs constituted by one of the edge sets.

We conclude that this smallness, induced by adding a sparse random edge set to a graph with a large average distance, is the point most small–world network models have focused on and that

[11] A *local minimum spanning tree (LMST)* is not necessarily a tree [158, 51]. Here, every vertex computes the minimum spanning tree of the subgraph of vertices that are within distance k. In the $LMST^+$ ($LMST^-$) graph, an edge is contained in the $LMST$ if both vertices agree (at least one of the vertices states) that it is in the minimum spanning tree of their (its) local neighborhood.

seems to be the essence of the small-world network models proposed so far. Since *local graphs* are one of the few graph families with a diameter larger than that of a corresponding random graph, they are just a special graph class that shows the small-world effect when combined with some random edges, and the clustering coefficient is one measure that is likely to measure locality.

In light of this review, we suggest using the term *small-world phenomenon* for the drop in diameter that occurs when a certain class of graphs is combined with a sparse random graph. In the following section we will formally define the small-world phenomenon and describe properties of hybrid graphs that show the small-world phenomenon. Furthermore, we will give a formal analysis of the diameter of these combined graphs in dependency of the number of edges in the random edge set.

4.3 Hybrid Graphs Showing the Small–World Phenomenon

Based on the analysis given above, our generalized framework describes graphs that show the *small–world phenomenon*, defined as the substantial decrease in the diameter by building hybrid graphs out of two graph components where each of the components alone has a higher diameter than the hybrid graph in terms of the asymptotic growth. In the following we will restrict ourselves to those hybrid graphs where one component is a random graph from the $\mathcal{G}(n,p)$ family. We will show that the small–world phenomenon can be found in a large family of hybrid graphs with different graph families building the non–random graph component. We introduce a new characteristic for graph families, the so–called *regular decomposability*, and prove that it is sufficient for any hybrid graph to show the small–world phenomenon when the non–random graph family has this property. We also give strong upper bounds on the diameter of the hybrid graph in dependence of p and the structure of the non–random graph component.

In order to do so, we will provide some necessary definitions in 4.3.1, followed by the formal definition of the model in 4.3.2.

4.3.1 Definitions

A graph family $G(n)$ denotes any set of graphs generated by the same algorithm and parameterized by the number n of vertices in it and some extra parameters, if needed (s. 3.4). In the case of deterministic graph families and a fixed set of parameters only one specific graph is generated, whereas in graph families generated partly by probabilistic processes, $G(n)$ is defined as the set of all possible realizations. Statements about $G(n)$ are then interpreted as statements about *expected characteristics* of this set. In this chapter we will use the notation $G(n)$ interchangeably for the whole set or a specific realization of this set.

A special case of grid graphs (3.4.1), a regular d–dimensional, equilateral grid is denoted by $G_d(a)$ and defined as a set of vertices placed on integer positions in d dimensions. $a \in \mathbb{N}$ denotes the number of vertices placed in each of the d dimensions. The number of vertices in this grid is then given by $n = a^d$, where every possible position—identified by a d–dimensional vector $(1 \leq b_1 \leq a, 1 \leq b_2 \leq a, \ldots, 1 \leq b_d \leq a)$—is occupied by one vertex.

We will use the following theorem on the diameter of random graphs $G(n,p)$ [35]:

Theorem 4.2
If $pn/\log n \to \infty$ and $\log n/\log(np) \to \infty$ then $D(G(n,p))$ is asymptotically equal to $\log n/\log(np)$ w.h.p.[12].

[12] The acronym **w.h.p.** stands for *with high probability*, and it states that the probability that the specified event

Note that this theorem implicitly includes that the random graph is connected with high probability. To simplify the following proofs we will use a stricter version of the theorem and require additionally that $p \geq (\log n)^{1+\epsilon}/n$, where $\epsilon > 0, \in \mathbb{R}$.

4.3.2 Formal Definition of the Small–World Phenomenon

The following definition of the small–world phenomenon describes the decrease in the diameter by combining two graph components that each have a higher diameter in terms of asymptotical growth:

Definition 4.3
A hybrid graph family is defined as any combination $G_{LR}(n)$ of some graph family $G_L(n)$ and a random graph family $G_R(n)$. $G_{LR}(n)$ shows the *small–world phenomenon* if the diameter $D(G_{LR}(n))$ is scaling at most polylogarithmically and if the following relations hold for $n \to \infty$:

$$\frac{D(G_L(n))}{D(G_{LR}(n))} \to \infty \quad \text{and} \quad \frac{D(G_R(n))}{D(G_{LR}(n))} \to \infty, \tag{4.21}$$

i.e., if the diameter of each graph component alone grows faster than the diameter of the hybrid graph.

We will now present a rather general proof pattern with which an upper bound on the diameter of certain hybrid graphs can be given. The number of required properties of these hybrid graphs is very small and thus a large family of hybrid graphs falls into this category of graph families showing the small–world phenomenon. For didactic purposes we start with the simple model of a d-dimensional grid combined with a sparse $G(n,p)$ instance as described by Newman and Watts [174]. This model will then be generalized to a large family of hybrid graphs that fall into the above given definition. We will denote such a hybrid graph by $G_d(n,p)$, which is given by the combination of a regular grid $G_{d,\infty}(\sqrt[d]{n}, 1)$, for simplicity denoted by $G_d(\sqrt[d]{n})$ here, and a random graph $G(n,p)$. Note that $a :=^d \sqrt{n}$ denotes the width of the grid in every dimension.

Since the basic network is a d–dimensional grid, the diameter of this component without any added random edges will scale with $a-1$ for a fixed dimension d: $D(G_d(a)) = d \cdot (a-1)$. If the added random graph is built with p greater than or equal to $(\log p)^{1+\epsilon}/n$ then the combined graph will have a diameter that is dominated by the diameter of the random graph and thus is asymptotically at most $\log n/\log(np)$ (Theorem 4.2). Thus, a hybrid graph with a dense random graph component does not show the small–world phenomenon as it is defined above.

In the following we will show what happens in the regime where p lies below $(\log p)^{1+\epsilon}/n$ and describe the regime in which the diameter of the hybrid graph will scale at most (poly–) logarithmically.

4.3.3 The Diameter of $G_d(n,p)$–Graphs

For the above given model of a graph $G_d(n,p)$ the following lemma holds:

will **not** occur is less than $1/n$, where n is a parameter that describes the size of the sample. In this case, **w.h.p.** states that choosing any graph from $\mathcal{G}(n,p)$ at random, the probability that this graph has a diameter **higher** than that given by Theorem 4.2 is less than $1/n$.

Lemma 4.2
For $p = \frac{1}{cn}$, $c \in \mathbb{R}^+$ the diameter of $G_d(n,p)$ is asymptotically bound by at most

$$d \cdot \left(\left\lceil \sqrt[d]{c \cdot (\log n)^{1+\epsilon}} \right\rceil - 1 \right) \cdot \left(\frac{\log n}{(1+\epsilon)\log\log n - \log 2} + 1 \right) \tag{4.22}$$

The proof proceeds in four steps:

1. The graph $G_d(n,p)$ is partitioned into n_S connected d–dimensional equilateral subgraphs $S_i, 1 \leq i \leq n_S$ with side length l such that each subgraph contains at least $s = l^d \geq c \cdot (\log n)^{1+\epsilon}$ vertices (Figure 4.5).
2. For any a, only $\lfloor a/l \rfloor$ full subgraphs per dimension can be built. n^* denotes the number of all vertices contained in a full subgraph. We will show that the $n - n^*$vertices that are not contained in any full subgraph constitute a vanishing fraction of all vertices for $n \to \infty$. We will thus base our proof on a reduced regular d–dimensional grid of size n^* that contains only the full subgraphs.
3. We construct a supergraph $G_S(n_S) = (S, E')$ where each vertex $v_i \in S$ uniquely represents the subgraph S_i for $1 \leq i \leq n_S$. Edge $e = (v_i, v_j)$ is member of E' iff[13] there is at least one random edge from any vertex in S_i to any vertex in S_j. We will prove that Theorem 4.2 can be applied to $G_S(n_S)$.
4. $G_S(n_S)$ is then expanded to gain a bound on the diameter of the original but reduced graph $G_d(n^*, p)$. The diameter of $G_d(n^*, p)$ is bound by the product of the diameter of the subgraphs $D(S_i)$ and the diameter $D(G_S)$. We will show that there are numerous partitions of $G_d(n,p)$ into n_S subgraphs. Especially, for any pair of vertices v, w there is at least one partition of $G_d(n)$ such that both, v and w, are contained in full subgraphs. Since every supergraph based on a possible partition obeys Theorem 4.2, we will therefore have shown that the whole graph $G_d(n,p)$ obeys Lemma 4.2 and the case is proven.

We will start by partitioning a $G_d(n,p)$ graph. Let $S_i, 1 \leq i \leq n_S$ denote an equilateral subgraph that has a side length of $l = \left\lceil \sqrt[d]{c \cdot (\log n)^{1+\epsilon}} \right\rceil$ in each dimension. The number s of vertices contained in one (full) subgraph is bound by:

$$c \cdot (\log n)^{1+\epsilon} \leq s = \left\lceil \sqrt[d]{c \cdot (\log n)^{1+\epsilon}} \right\rceil^d < 2^d \cdot c \cdot (\log n)^{1+\epsilon} \tag{4.23}$$

$G_d(n,p)$ is partitioned into subgraphs as shown in Fig. 4.5 for a 2-dimensional example. Obviously, incomplete subgraphs exist if a/l is not an integer. The leftover vertices can be placed arbitrarily between full subgraphs as indicated in Fig. 4.5(b). For simplicity we will consider instead of $G_d(n)$ a smaller hyper-cube $G_d(n^*)$ containing all full subgraphs. Note that now a^* with $\sqrt[d]{n^*} = a^* \leq a$ is the maximal integer smaller than a that is a multiple of l. Let $q = a^*/l$ denote the number of subgraphs in one dimension.

The relative fraction of vertices not contained in full subgraphs is approaching 0 for $n \to \infty$:

$$\frac{n - n^*}{n} \leq \frac{(l \cdot (q+1))^d - (l \cdot q)^d}{(l \cdot q)^d} \tag{4.24}$$

$$= \left(\frac{q+1}{q} \right)^d - 1 \tag{4.25}$$

[13] **iff** is used throughout the text as a short-hand for *if and only if.*

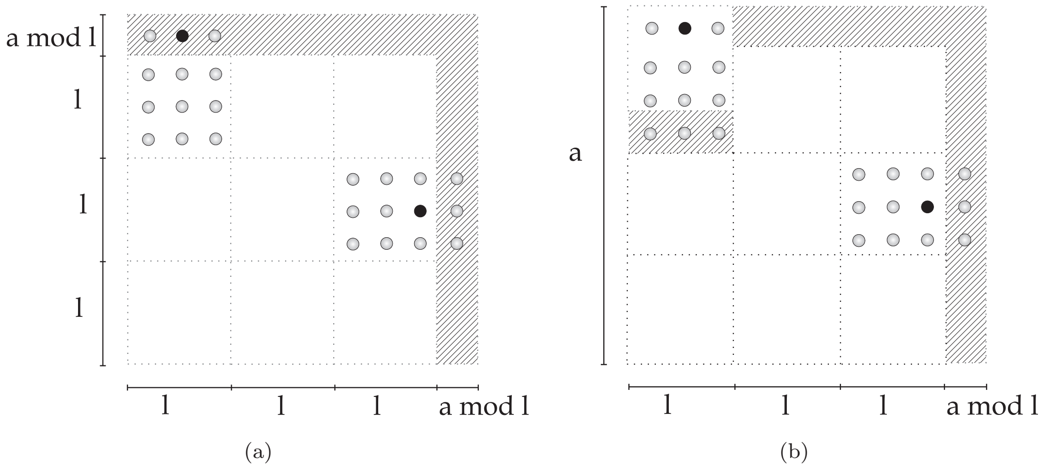


Fig. 4.5: (a), (b): Different valid partitions for a 2–dimensional grid with side length a. Full equilateral subgraphs with side length l may be placed arbitrarily as long as their number is maximal. Therefore numerous partitions exist and for each pair of vertices numerous partitions can be found where both are contained in full subgraphs.

Since $q \to \infty$ for $n \to \infty$, the relative fraction of ignored vertices is asymptotically 0. Note that $n_S = \frac{n^*}{s} \geq \frac{n}{2^d \cdot c \cdot (\log n)^{1+\epsilon}} \to \infty$. Thus for $n \to \infty$ we may safely use

$$n \geq n^* > n/2 \tag{4.26}$$

In $G_d(n^*, p)$ there are s^2 possible random edges between any vertex from subgraph S_i and any vertex from subgraph S_j. Each of these edges exists independently with probability p. It follows that for G_S the probability p_S is exactly $\frac{s^2}{cn}$.

We will now prove that Theorem 4.2 can be applied to $G_S(n)$. A basic observation is that for $n \to \infty$, also $n_S \to \infty$. Additionally, it has to be shown that $\frac{p_S n_S}{\log n_S} \to \infty$ and $\frac{\log n_S}{\log(n_S p_S)} \to \infty$ for $n_S \to \infty$.

Observe that for all $n_S > 1$, $n^* > n/2$ (eq. 4.26) the following two equations hold:

$$\begin{aligned}
\frac{p_S \cdot n_S}{\log n_S} &= \frac{s^2}{cn} \cdot \frac{n^*}{s} \cdot \frac{1}{\log \frac{n^*}{s}} && (4.27)\\
&\geq \frac{s}{2c(\log n^* - \log s)} && (4.28)\\
&\geq \frac{(\log n)^{1+\epsilon}}{2\log n - 2\log s} && (4.29)
\end{aligned}$$

such that $\frac{p_S \cdot n_S}{\log n_S} \to \infty$ for $n \to \infty$ and

$$\begin{aligned}
\frac{\log n_S}{\log(p_S \cdot n_S)} &= \frac{\log \frac{n^*}{s}}{\log \frac{s \cdot n^*}{cn}} && (4.30)\\
&\geq \frac{\log n/2 - \log(2^d \cdot c(\log n)^{1+\epsilon})}{\log(2^d(\log n)^{1+\epsilon})} && (4.31)
\end{aligned}$$

such that also $\frac{\log n_S}{\log(p_S \cdot n_S)} \to \infty$. By Theorem 4.2 we know that G_S has thus a diameter that approaches $\frac{\log n_S}{\log(p_S \cdot n_S)}$ asymptotically. Regarding that $n^*/n > 1/2$ this is bound by

$$\begin{aligned}
D(G_S) &= \frac{\log n_S}{\log(p_S \cdot n_S)} && (4.32)\\
&\le \frac{\log n}{\log \frac{s}{2c}} && (4.33)\\
&\le \frac{\log n}{(1+\epsilon)\log\log n - \log 2} && (4.34)\\
&\le \frac{\log n}{\log\log n}, && (4.35)
\end{aligned}$$

where the last inequality is valid for all n with $\epsilon \log\log n > \log 2$.

We will now expand $G_S(n)$ in order to get an upper bound for the diameter of $G_d(n,p)$. Let v and w be two vertices in the original graph $G_d(n,p)$. First, note that $G_d(n,p)$ can be reduced to $G_d(n^*,p)$ such that v and w are contained in $G_d(n^*,p)$. This implies that there is a path in $G_S(n)$ from subgraph S_i containing v to subgraph S_j containing w with a length of no more than $D(G_S)$. Let the path from subgraph S_i to S_j be denoted by $(e_1, e_2, ..., e_k)$. This path can now be expanded to a path in the original graph $G_d(n,p)$ that connects v with w by first joining the path from vertex v to that vertex v' from S_i that is attached to e_1. This will at most take $D(S_i) = d \cdot (\left\lceil \sqrt[d]{c \cdot (\log n)^{1+\epsilon}} \right\rceil - 1)$ steps. For every edge $e_i = (v_i, w_i)$ that enters subgraph S_x and edge $e_j = (v_j, w_j)$ that leaves this subgraph on the way to subgraph S_j, an additional path is added to connect w_i with v_j. Since both, w_i and v_j, are in the same subgraph, this path is not longer than $D(S_x)$. Thus, the distance of v, w in the original graph $G_d(n,p)$ is asympotically given by at most

$$D(S_i) \cdot (D(G_S) + 1) \le$$
$$d \cdot \left(\left\lceil \sqrt[d]{c \cdot (\log n)^{1+\epsilon}} \right\rceil - 1 \right) \cdot \left(\tfrac{\log n}{(1+\epsilon)\log\log n - \log 2} + 1 \right)$$

With this, Lemma 4.2 is proven.

The simple model was restricted to grid graphs in which every vertex has degree 4. A straight forward generalization allows enlarging the degree of the vertices in the underlying grid graph as we will show now: as stated in Lemma 4.2, the diameter of a $G_d(n,p)$ graph is asymptotically at most $D(S_i) \cdot (D(G_S) + 1)$. Let $G_d(n,k,p)$ denote an extended regular grid, in which every vertex is connected to its k next neighbors as given by the graph theoretic distance in a simple d-dimensional grid, combined with an additional $G(n,p)$ graph. Since the diameter $D(S_i)$ of the subgrids depends only on the degree of the vertices in the underlying grid graph the diameter of the hybrid graph $G_d(n,k,p)$ can simply be reduced by adding more edges to the grid. For example, $D(S_i)$ is reduced to 1 if for $p = \frac{1}{c \cdot n}$ edges from every vertes to its $c \cdot (\log n)^{1+\epsilon}$ next neighbors are added, resulting in a combined graph $G_d(n, (\log n)^{1+\epsilon}, p)$ with a diameter of at most $D(G_S)$.

4.3.4 Generalizing the Family of Hybrid Graphs Showing the Small–World Phenomenon

The simple model allowed only grid graphs as the non–random graph component, and restricted p to be $1/cn$. We will now show that the model can be substantially generalized:

1. The probability p of the added random graph $G(n,p)$ can be as small as $\frac{1}{f(n) \cdot n}$ as long as $(\log n)^{1+(\epsilon/2)} \le f(n) \le n^{1-\delta}$ for some constants $\delta, \epsilon > 0$ and $n \to \infty$.

2. The basic regular d–dimensional grid can be replaced by other graph families, as long as they are *regularly decomposable*, a new property that will be introduced below.

These two points lead to a generalized theorem on the diameter of hybrid graphs showing the small–world phenomenon. We begin by discussing the first point, the generalization of the random graph component.

Generalizing the Random Graph Component

In section 4.3.3 p was restricted to be $1/cn$. To allow for smaller $p = \frac{1}{f(n)\cdot n}$, the size s of the subgraphs has to be chosen larger such that Theorem 4.2 can be applied. Let again n_S denote the number of vertices and p_s denote the probability of an edge in G_S. For simplicity we assume that $n_S = n/s \in \mathbb{N}$ and $p \cdot s \cdot n = (\log n)^{1+\epsilon} \in \mathbb{N}$. The general case follows the argument above. The number of nodes in each subgraph will be chosen such that
$s = \frac{(\log n)^{1+\epsilon}}{p\cdot n} = f(n)\cdot(\log n)^{1+\epsilon}$. Again, Lemma 4.2 requires the validity of

$$\frac{p_S n_S}{\log n_S} \to \infty \tag{4.36}$$

and

$$\frac{\log n_S}{\log(n_S p_S)} \to \infty \tag{4.37}$$

As before $p_S = s^2 \cdot p$. We first analyze the condition given in equation 4.36:

$$\begin{aligned}\frac{p_S n_S}{\log n_S} &= s^2 \cdot p \cdot \frac{n}{s} \cdot \frac{1}{\log n_S} && (4.38)\\ &> s \cdot p \cdot \frac{n}{\log n} && (4.39)\\ &= (\log n)^{\epsilon} && (4.40)\end{aligned}$$

which approaches infinity for increasing n. The second condition (4.37) simplifies to

$$\begin{aligned}\frac{\log n_S}{\log(n_S p_S)} &= \frac{\log\left(\frac{n}{s}\right)}{\log\left(\frac{n}{s}\cdot s^2 \cdot p\right)} && (4.41)\\ &= \frac{\log\left(\frac{\frac{n}{f(n)}}{(\log n)^{1+\epsilon}}\right)}{\log(\log n)^{1+\epsilon}} && (4.42)\\ &= \frac{\log\left(\frac{n}{f(n)}\right)}{\log(\log n)^{1+\epsilon}} - 1 && (4.43)\end{aligned}$$

which tends to infinity for $f(n) \le n^{1-\delta}, \delta > 0$. Therefore both conditions are met and Theorem 4.2 can be applied to G_S. If $f(n)$ is chosen to be lower than $(\log n)^{1+(\epsilon)}$ then the random graph component will be connected and Theorem 4.2 can be applied directly. Such, $f(n)$ is also restricted from below to be larger than $(\log n)^{1+(\epsilon)}$. This result is summarized in the following lemma.

Lemma 4.3
For any function $(\log n)^{1+(\epsilon/2)} \le f(n) \le n^{1-\delta}, \epsilon, \delta > 0$ and $p = \frac{1}{f(n)\cdot n}$, the grid graph within a $G_d(n,p)$ graph can be partitioned into $n_S = \frac{n}{s}$ subgraphs S_i of size $s = f(n)\cdot(\log n)^{1+\epsilon}$ such that

the diameter of $G_d(n,p)$ approaches asymptotically (Eq. 4.43)

$$\underbrace{d \cdot \left(\left\lceil \sqrt[d]{c \cdot (\log n)^{1+\epsilon}} \right\rceil - 1 \right)}_{D(S_i)} \cdot \underbrace{\left(\frac{\log(n/f(n))}{\log(\log n)^{1+\epsilon}} \right)}_{D(G_S)} . \tag{4.44}$$

Possible Replacements of the Regular Grid Graph

In the general proof pattern shown above, the following two properties of regular grid graphs are needed to bound the diameter of the resulting hybrid graph: first, regular grid graphs are partitionable for every n into $\Theta(n/s(n))$ subgraphs of size $s(n)$ for any function $s(n) \leq n$ such that each of these subgraphs is a connected graph; second, for any pair of vertices v, w there must be at least one partition such that v and w are contained in any of the subgraphs.

To abstract from this special graph family to all graph families with these two properties we introduce the following definition:

Definition 4.4
Let $G_L(n)$ be a graph family with the following two properties:

1. $G_L(n)$ is partitionable for every n into $\Theta(n/s(n))$ subgraphs of size $s(n)$ for any function $s(n) \leq n$ such that each of these subgraphs is a connected graph.
2. For any pair of vertices v, w and every n there must be at least one partition such that v and w are contained in proper subgraphs.

$G_L(n)$ is called a *regularly decomposable* graph family. For graph families with a stochastic generating process it is enough to show that such a partition exists *w.h.p.*.

Note that every graph family $G_L(n)$ is *regularly decomposable* for at least $s(n) = 1$. Let $s_{max}(n)$ be that function $s'(n)$ that has the fastest growth of all functions $s(n)$ for which $G_L(n)$ is regularly decomposable. If now $s_{max}(n) = k,\ k \in \mathbb{N}$ for $G_L(n)$ and $G_L(n)$ replaces the regular grid then it is clear that the size of the subgraphs is also at most k to obey $\Theta(n/s)$. This implies that the probability p of the added random graph must be at least $O\left(\frac{(\log n)^{1+\epsilon}}{n}\right)$ in order to achieve a supergraph that obeys Theorem 4.2. It follows that the diameter is dominated by the diameter of a random graph because we add a dense random graph. By definition of the small–world phenomenon, such a graph family will not show this phenomenon in the hybrid graph model.

We conclude this section with a theorem on the diameter of generalized small–world models combining a regularly decomposable graph family with a sparse random graph:

Theorem 4.3
Let $G_L(n,p)$ denote the combination of instances of a *regularly decomposable* graph family $G_L(n)$ and a $G(n,p)$ graph where $p = \frac{1}{f(n) \cdot n}$, $\frac{1}{(\log n)^{1+(\epsilon/2)}} \leq f(n) \leq \frac{1}{n^{1-\delta}}$, and $\epsilon > 0, \delta > 0$. $D(s(n,p))$ denotes the maximal diameter of any subgraph of $G_L(n,p)$ with size $s(n,p) = \frac{(\log n)^{1+\epsilon}}{p \cdot n}$, $\epsilon > 0$, the diameter of $G_L(n,p)$ is asymptotically at most:

$$D\left(s\left(n, \frac{1}{f(n) \cdot n} \right) \right) \cdot \underbrace{\left(\frac{\log(n/f(n)}{\log(\log n)^{1+\epsilon}} \right)}_{D(G_S)} \tag{4.45}$$

4.3.5 Proving Regular Decomposability

We now want to discuss the connection between regularly decomposable graph families and local graph families. As indicated before, the classic small–world models are based on local graph families but our proof pattern is based on the notion of regularly decomposable graph families. We will first introduce the idea of *locally clustered* graphs, and then show that at least some of these locally clustered graphs are at the same time regularly decomposable.

A graph G or a graph family $G(n)$ is *clustered* if (for every n) there is a partitioning such that the fraction of realized edges within the subgraphs given by the partition is much higher than the fraction of realized edges between the subgraphs, i.e., the subgraphs are *locally dense* and *globally sparse*. This definition is quite broad because there are many different measurements that try to quantify how good such a clustering is, e.g., coverage, intra– and inter–cluster conductance (for an overview see [89]), or modularity [93], to name but a few. Here, we just want to concentrate on the *locally dense* vs. *globally sparse* part of that definition that is the intuive basis for most of these measures. An embedded graph or a graph family $G(n)$ is *locally clustered* if at least one partition of the **space** in which the graph is embedded exists such that the subgraphs within the resulting subspaces are locally dense and the graph is globally sparse. If a graph is only given by its adjacency matrix and the vertices have no position in space, the graph is said to be *potentially locally clustered* if it can be embedded in a metric space such that the graph is locally clustered. Note that most of the classic small-world models are based on locally clustered graph families like circulant graphs [32, 242] or d-dimensional lattices [11, 53, 125, 126], and that these are at the same time locally clustered and regularly decomposable.

We will now show under which conditions general locally clustered graph families are also regularly decomposable. The main idea behind the proof is to use the embedding of the vertices in a space and to divide this space into areas that contain connected subgraphs with at least a given number of vertices. Thus, the partition of space induces a partition of the graph. The task that has to be tailored for every specific graph family is to show that there is always a partition of the space that yields $\Theta(n/s(n)))$ connected subgraphs with at least the wanted number $s(n)$ of vertices. Graph families that are locally clustered and show this property are also regularly decomposable. Note that this is not a necessary property of locally clustered graph families. It is possible to construct artificial locally clustered graph families that lack this property, e.g., a graph family $G_5(n)$ which consists of unconnected 5–cliques $C_1, C_2, ..., C_{\lfloor n/5 \rfloor}$ where all members of clique C_i are located at position i in a 1–dimensional space. This family is clearly locally clustered for the partition $\{C_1, C_2, \ldots, C_{\lfloor n/5 \rfloor}\}$, but it is not regularly decomposable because it is not possible to partition the graph into subgraphs with a size $s(n) > 5$.

In the following we will analyze the regular decomposability of a more sophisticated locally clustered graph family, namely that of k–nearest neighborhood graphs (*knn–graphs*). This graph family is an important geometric random graph family that is used mainly in the modeling of sensor and multi-hop communication networks and is already well analyzed [197]. Given a set of vertices distributed in any d–dimensional space, let E be the set of directed edges such that every vertex v is connected to its k nearest neighbors. If the set of k nearest neighbors is ambiguous for any vertex, then any of the possible sets is chosen uniformly at random. Note that the edge relation is not symmetric and therefore the knn–graph is a directed graph where outgoing and ingoing edges of a vertex have to be differentiated. We will now prove the following theorem:

Theorem 4.4
The k–nearest neighborhood graph family is *regularly decomposable* if the vertices of a given instance are distributed uniformly at random in a 2–dimensional unit–square.

The proof proceeds by the following steps:

1. The upper bound on the expected distance to nearest neighbors is determined (4.3.5).
2. We show that a knn–graph is connected with high probability for $k > \log n$ (4.3.6).
3. Finally, we show that a generic partition procedure yields the required $\Theta(n/s(n))$ subgraphs for a given size function $s(n) < n$ (4.3.7).

A Bound for the Maximum Distance of Nearest Neighbors

Let the *knn*–disk of any vertex v be defined as the minimal disk which contains all of its k nearest neighbors. Note that the disk's radius is equal to the maximum distance of any connected nearest neighbor to v. The probability for any vertex v to be placed in some area of size $A \leq 1$ within the unit square is exactly A. Thus, the placement of vertices into a given area is a Bernoulli trial with $p = A$ and $q = 1 - A$. The radius of a disk with expectedly $k = n \cdot A$ vertices is then given by $\bar{r} = \sqrt{\frac{k}{\pi \cdot n}}$. Now, the Chernoff bound (3.7.2) can be applied to yield an upper bound for the diameter of any disk within the unit square that contains at least k vertices:

Observation 4.1
Let $\hat{r} = \sqrt{\hat{c}} \cdot \bar{r}$ denote a knn–disk radius with $\hat{c} = 3 + \sqrt{8}$. Further let $k \geq \log n$. With high probability, no disk with radius $\hat{r}$ around any vertex v exists that does not contain at least k vertices.

Proof 4.1
Let D_v denote a knn–disk around v with an expected number of vertices lying in that disk equal to $\bar{k} = c \cdot k$. X_k denotes the number of vertices lying inside of D_v. Now we apply a relaxed version of the Chernoff inequality for independent Bernoulli trials. With $\mu = c \cdot k$ and $\delta = 1 - \frac{1}{c}$

$$Pr[X_k < (1-\delta)\mu] < e^{-\frac{1}{2}\mu\delta^2} = e^{-\frac{ck}{2}(1-\frac{1}{c})^2} < n^{-\frac{c}{2}(1-\frac{1}{c})^2}. \tag{4.46}$$

The latter inequation is only valid for $k > \log n$. For $c = \hat{c} = 3 + \sqrt{8}$ we yield

$$Pr[X_k < (1-\delta)\mu] < \frac{1}{n^2}. \tag{4.47}$$

Hence, the probability that there is a knn–disk with radius larger than $\hat{r} = \sqrt{3 + \sqrt{8}} \cdot \bar{r}$ in a knn–graph with $k > \log n$ is $< 1/n$.

The interpretation of this result is that it is almost impossible for $n \rightarrow \infty$ that any knn–disk exists with a radius larger than $\hat{r}$. Therefore in our following theorems, we consider the radius of knn–disks to be bound by $\hat{r} = \sqrt{\hat{c}} \cdot \bar{r}$.

Note that these equations are only valid for disks that do not intersect with border of the unit square. If a vertex v_c is positioned in a corner of the unit square, a factor of 2 has to be applied to $\hat{r}$ to yield the correct upper bound on the radius of its *knn*–disk. We will now sketch a sufficient condition such that a knn–graph is connected with high probability.

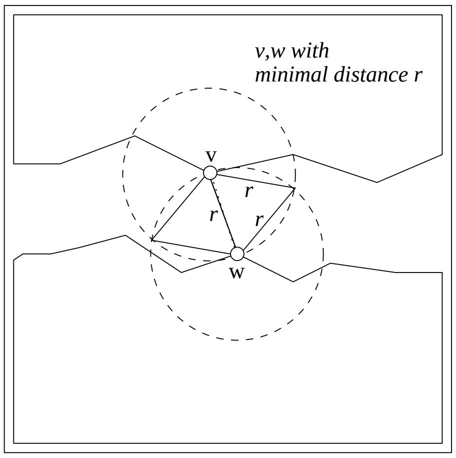


Fig. 4.6: v and w are two vertices from different connected components of a knn–graph having minimal Euclidian distance to each other. A circle is drawn around each of v and w, both having a radius that equals the Euclidian distance between v and w. The figure shows that none of the k–nearest neighbors of either v or w can exist in the intersection of these circles without contradicting the condition that v and w are the pair of vertices from different connected components with minimal Euclidian distance.

4.3.6 Connectedness of knn–Graphs

It can be shown by a simple argument that a *knn*–graph with $k > \log n/\log(3/2)$ is connected with high probability. As sketched in Fig. 4.6, every unconnected *knn*–graph contains one pair of closest vertices lying in different components. It can be shown that there must be an angle of at least 120° in which none of either v's or w's k nearest neighbors is placed. A simple stochastic argument shows that the probability for the existence of any vertex with this property is given by $(2/3)^k$. Equating this with the probability bound of $1/n$ and solving the equation yields the needed k such that w.h.p. not even one vertex with the above mentioned property exists. This leads to the following lemma:

Lemma 4.4
A knn–graph is connected with high probability for $k \geq \frac{\log n}{\log(3/2)}$

Note that the probability of an unconnected knn–graph is smaller than $1/n$ since the existence of at least one vertex with the above given property is only necessary for an unconnected graph, but certainly not sufficient. Nonetheless, this simple argument comes very close to a bound given by Xue and Kumar in a much more involved analysis of which we learned later [252]. They have shown that if k is larger than $5.1774 \log n$ the graph is connected with probability approaching one as n goes to ∞, while our simple analysis comes to the conclusion that it is $5.68 \log n$. However, motivated by empirical results, Xue and Kumar conjecture that 1 is the critical value, i.e., that it is enough if every vertex is connected to its $\log n$ next neighbors such that a tight upper bound is still an open question for further research.

We will now show that connected commensurate partitions can be found.

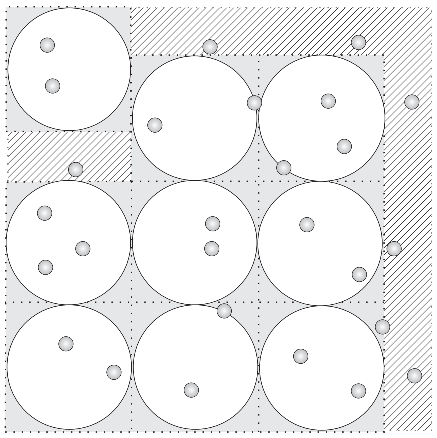


Fig. 4.7: This figure shows the result of the partitioning procedure for $s = 2$ as described in 4.3.7. Each square contains more than $4/\pi \cdot s \approx 2.5$ vertices. Each circle within any quadratic region contains at least $s = 2$ vertices that must form a connected subgraph. Note that the distribution of points is only schematic.

4.3.7 Constructing the Partition

The following procedure constructs partitions as required by the definition of *regularly decomposable* graph families. The needed size s of the subgraphs depends on the probability $p = \frac{1}{f(n) \cdot n}$ of the added random graph $G(n, p)$:

$$s(n) = f(n) \cdot \log n^{1+\epsilon} \tag{4.48}$$

The partitioning algorithm must be capable of finding for each pair of vertices a partition into $\Theta(n/s)$ subgraphs such that both vertices are included in some full subgraph of size s. To guarantee this we construct slightly different partitions for each pair v, w of vertices. For each of them a geometric partition that is based on squares containing at least $4/\pi \cdot s$ vertices is built. The exact positions of the squares are chosen such that both vertices are contained in full subgraphs. Besides this requirement the positions of the squares can be chosen arbitrarily as long as the number of squares placed completely inside the unit square is maximal. Note that a constant relative fraction of vertices may exist that is not contained in any subgraph. Each of the squares covers an area A_s such that with high probability at least $4/\pi \cdot s$ vertices are positioned in each of them. The area is given by $A_s > \frac{4}{\pi} \cdot \pi \cdot \hat{r}^2$, where $\hat{r}$ denotes the maximal expected knn–disk radius (Observation 4.1). The maximal (centered) circle (Fig. 4.7) within each A_s contains only vertices from the same connected component. Otherwise there is at least one vertex with an angle of 120° in which none of its nearest neighbors is placed; this is highly unlikely as was already shown in Lemma 4.4. The area of this circle covers $\pi/4$ of A_s. Therefore, at least $\pi/4 \cdot 4/\pi \cdot s = s$ vertices can be found in the connected component of each A_s. Fig. 4.7 shows an example of a partition for $s = 2$.

Note that the diameter $D(S_i)$ of the subgraphs is expectedly scaling with $O(\sqrt[d]{s})$ as is the case with grid graphs.

With this we have shown that a locally clustered graph family is also regularly decomposable if it is possible to divide the space in which it is located into $\Theta(n/s(n))$ areas that contain connected subgraphs of a required size $s(n)$.

4.4 Summary

In this chapter we have tried to capture the essence of classic small–world network models that were proposed over the last few years. The finding is that a network model qualifies as a small–world network model if it shows the small–world phenomonenon: the decrease in the average distance of a graph with a foremost large diameter after it is combined with only a sparse random graph. We have shown here that all graph families that are regularly decomposable will show this phenomenon, and that many of the network models are local and regularly decomposable at the same time.

4.4.1 Network Design

After so many small–world network models have already been proposed, why were we interested in offering another? Soon after Watts' and Strogatz' model was proposed, many other network properties besides the clustering coefficient were shown to be important structural descriptors for one or another real-world network, e.g., a scale-free degree distribution [8], a certain assortativity [177], or the occurrence of certain subgraphs [168]. In this light, Kleinberg proposed a small–world network model in which a greedy routing algorithm can find short paths [125, 126], and Chung and Lu proposed a small–world network model that shows a scale-free degree distribution [11, 53][14]. Our perspective is that of a network designer, for example a designer of peer-to-peer or other communication networks. It is reasonable that every setting needs special combinations of the above mentioned structural properties, and the small–world effect is often one of the desired features because it allows for efficient broadcasting owing to the small average distance. But of course, since random edges are often long–distance edges, or, in the virtual social space of peer-to-peer networks, would require people of very different interests to connect, the number of these edges should be as small as possible. As long as the other desirable structural properties are mainly local, e.g., a high clustering coefficient or a certain assortativity, these can be simply realized in our model by choosing a regularly decomposable graph family that shows these properties as the $G_L(n)$ part of the model. Theorem 4.3.4 then allows choosing the minimal number of long-distance edges that are needed to assure a certain average distance in the graph.

4.4.2 Modeling Network Generating Processes

One of our main viewpoints on the evolution of real-world networks is that there is always a **system** that contains a **network–generating process** that produces the network. For example, the population living in a certain area sculpts its habitat by building and abandoning villages and cities that are connected by a slowly changing system of streets and paths. It is intuitive that such a network built of streets and paths will—on a large scale—always show the same structural properties, for example, a typical length distribution of the streets between crossroads. This is because the system, built by people with limited resources, will always be under similar constraints and objective functions, e.g., that the system of roads should connect most villages efficiently without becoming too expensive. As already sketched earlier, Gastner and Newman have shown that commuter networks follow such an objective function [90]. From this viewpoint,

[14] From these models, the approach of Andersen, Chung, and Lu and Chung and Lu is most similar to ours since they also build hybrid graphs of a *local* and a *global graph component* [11, 53], although different proof techniques are used to determine the diameter of the hybrid graph. Their global graph component is a power-law random graph that induces a scale-free degree distribution and is already well analyzed [52]. Since their definition of local graphs is different from our definition of *regular decomposability*, how close the two models are remains an open question.

networks are small–world networks if in their network–generating process local edges are preferred, as long as at least some random edges are allowed. This view is much more focused on the network–generating process than the perspective of Watts' and Strogatz' model, and compared with their model, ours fails to identify single networks as small–worlds if the process that built them is unknown. This comes from the definition of the small–world phenomenon that is based on the comparison of the behavior of three graph families for infinite n, i.e., on the analysis of the expected result of the network generating process that constitutes them. We think that network science needs to focus more on network generating processes. We think that one of the most intriguing aspects of the small–world network model is the fact that a globally optimal—in the sense of a small average distance—network structure arises even though most of the edges are local. And this global optimality can be achieved without a central organizing process since the only requirement is that every vertex should build some random edges. In our eyes this is an important, but somewhat hidden aspect, of the small–world network model that was not focused on much in the publications that referred to the article of Watts and Strogatz. We will extend this idea of *global optimality by local behavior* and try to find out what kind of network structures can be achieved in such a setting in chapter 6.

But before that, we will first speak a bit more about *network generating processes*. We have used this term in this chapter to describe locality, saying that a local graph is one whose network generating process more probably realizes edges with shorter distance than those with a greater distance. We have discussed the fact that the clustering coefficient cannot be strongly correlated with a local network generating process, and it seems that there is no single measure that can determine whether a network is local. The reason is that even if a network were built by a network generating process preferring local edges, the resulting adjacency matrix would be the result of two different properties: the distance function between the vertices **and** the preference with which these are realized. And for every locality measure proposed so far it is easy to imagine a certain distribution of the vertices in a metric space such that the measure will be low. Since it is hard to determine whether a network is local, we will instead try in the next chapter to bound the number of edges built by a random network generating process. This technique has led to the discovery of a new network structure that shows a peculiar behavior in real–world networks and is a promising candidate for new efficient algorithms on real–world networks.

5. NETWORK–GENERATING SYSTEMS AND PROCESSES

In this chapter we will introduce the notion of *network–generating systems*, a notion that is similar to the classical description of graph families (s. 3.4), but captures the more general process that builds a network in real life. A network–generating system consists of the *environment* in which a network is built, *network–generating entities* and, if the network–generating entities are not generating a network between themselves, the *objects* that are connected by the entities. To illustrate this difference: *social networks* are of the first type; here, humans are connecting themselves to other humans by getting to know them, working with them, having a sexual relationship with them, giving birth to them, etc. In contrast to this, a *word association network* is built by asking humans which words they associate with a given one. Thus, in the first case, the network is built between the network–generating entities themselves, in the second the network–generating entities are relating other objects with each other. We think that it is important to understand which decision rules make network–generating entities relate themselves to others or relate objects with each other, i.e., the *network–generating process.* Understanding this process is important for computer scientists for two reasons:

1. In the design of technical networks, network–generating systems in which the network–generating entities decide **decentrally** which relations to make become more and more important as we will argue in 5.1.1. Nonetheless it is desired in many cases to steer this decentrally organized network–generating process towards a globally favourable network structure. To do so, it is important to understand the network–generating process that governs the network's design in order to give incentives to the single entities to make the globally 'right' decisions. This topic will be further explored in chapter 6 where we make some first analyses of how to design efficient network–generating rules that are beneficial both locally for the network–generating entity and globally for the whole system.
2. As already sketched in the introduction (2.3.3), some algorithms are *contextual*, especially in network analysis problems, like *centrality indices* and *clustering algorithms.* A *contextual algorithm* is designed to answer a question concerning the complex system by analyzing a network deduced from this system, i.e., the question concerning a complex system is transformed to a question on the complex network level, solved there, and the solutions are then re-transformed to solutions on the complex system level. In the case of centrality indices, the question is how to find the most important object in a complex system, in the case of clustering, the question is how to find groups of objects in the system that are strongly related. Both of these questions have led to dozens of different analytical algorithms that are applied to the networks derived from a complex system [132, 133, 89]. But although every one of these algorithms can be applied to every complex network, the answers they give cannot and should not be re-transformed into an answer on the complex system level in all cases. This problem will be discussed in 5.1.2.

We will start this chapter with a discussion of *network–generating systems* in 5.1, followed by a discussion of the importance of understanding network–generating processes in decentrally organized network structures and a discussion of *contextual algorithms*. These two sections aim to show why it is important to understand the network–generating process that is prevalent in a given network–generating system. Often, this process is hidden, and all we have is the network itself, and — if we are lucky — some data on its dynamic changes in time. Our contribution here is a new technique with which the maximal percentage of random edges in a given graph can be bound. This is important, for example, for the two types of contextual algorithms mentioned above: If the percentage of random edges is high, centrality indices and clustering algorithms should only be applied with great care. Our technique chooses any spanning tree T of the graph and counts how many edges connect vertices that have a distance of k in the tree, the so-called *tree distance distribution*. This technique is discussed in 5.2[1]. We analyzed different real-world networks and most of them have a very peculiar tree distance distribution that is much steeper than expected in, e.g., a random graph, as shown in 5.3. In that section, we will also introduce the notion of *embeddable hierarchically local networks* to enable a characterization of those real–world networks that show a monotonically non-increasing tree distance distribution.

A natural question is whether there is an efficient algorithm to compute an optimal spanning tree. But 'optimality' is difficult to define for a distribution, thus we introduce a simple quality measure of a given spanning tree, namely the sum of the tree distances it induces in 5.4. We will also give some simple lower and upper bounds for this quality measure, and then show that finding an optimal spanning tree is NP-hard. We will then analyze in 5.5 whether a certain kind of simple spanning tree, namely BFS trees, yield a constant factor approximation of the minimal $Q(T)$[2].

It turned out that a steep tree distance distribution gives an upper bound to three interesting theoretical problems, the so-called *lower-stretch spanning tree problem* [73], the *minimal length fundamental cycle/cut basis problem* [61, 211], and the *minimum length cycle basis problem* [30, 109]. There are some problems whose runtimes depend on the sum of tree distances in a spanning tree, and since many real-world networks show a steep tree distance distribution, these problems are easier to solve on real-world networks than on general graphs. To our knowledge, there is so far no result that uses small–world network properties for an efficient algorithm, and there is only one paper that shows how to use a scale–free degree distribution for solving a classical theoretical problem in computer science, the problem of finding a dominating set of a graph [56]. But this newly found structure has already been proven to be important for the design of efficient algorithms and we hope that further algorithms will be developed that use this structure. It is thus a valuable categorization of real–world graph families in algorithm design, as we will discuss in 5.6.

Because finding the optimal backbone is NP-hard, but finding a good spanning tree is important for many applications as sketched above, we will introduce a local optimization heuristic with which a given spanning tree can be improved and some heuristics for promising starting spanning trees for this algorithm in 5.7.

In section 5.8 we will then show how a spanning tree with a steep tree distance distribution can be used to draw a large and complex network[3]. We conclude this chapter with a discussion in 5.9 of the findings.

5.1 Network–Generating Systems: Processes vs. Structural Properties

A graph is simply a mathematical representation of a set of objects and one of possibly many relationships between these objects. In this simple setting, it is easy to define graph families as

[1] This part of the work was partly conducted with Zoran Nikoloski and Michael Kaufmann.
[2] This part of the work was conducted with Michael Kaufmann.
[3] The layout algorithm and the local optimization process was developed together with Stephan Kottler [149].

a function of n, the number of vertices, and m, the number of edges, for example grid graphs, random graphs, or hypercubes [102].

As discussed in the introduction, a network is more than simply a graph: its objects have properties that go beyond the information about whether they are related to the other objects. Why should any property of an object that cannot be directly represented by the graph be correlated with its propensity to build edges? The need to add such properties to vertices was discussed in many papers, for example to explain why and how a later arriving vertex in the preferential attachment model could overtake all others and win the most edges [31]. This is a phenomenon that was observed in the WWW where Google, soon after its release, was the first and best known search engine despite the fact that Altavista and Yahoo had already been there for a long time. To maintain the idea of the preferential attachment model, in which such a behavior would be most unlikely, Bianconi et al. introduced the *fitness* of an object, which is related to the probability with which it will have a relationship with other objects [31]. In this model, a new vertex could simply have an overwhelming fitness that makes it attractive to connect to without the need that it have a high degree to start with.

Another model that required vertices with an additional property is one that tries to model *citation networks*. In this model, a time stamp is associated with each vertex, and the older this time stamp is the more likely it is that the vertex is removed from the set of vertices that are allowed to acquire new edges. Such a mechanism is thought to model the propensity of many articles to be known and cited for a while but after some time to be forgotten and not cited again [129].

These two examples show that real–world networks most often cannot be accurately modeled by simple graph families but rather by more complicated network–generating processes that take additional properties of the objects to be related into account. Furthermore, in all networks there is (or are) some entity (entities) that generate(s) the network, and that decide(s) who is related to whom. It does not have to be a real decision from a real agent, but in any case there is some process that creates the relation between the objects that are represented by the graph. In social networks it is a human person that decides with whom to interact, in citation networks it is an author that decides whom to cite, in protein-protein interaction it is (among others) the time and place where a protein is expressed and its shape that makes it interact with others. And of course the probability of a relationship is influenced by the environment in which the network–generating entities are situated: if a child is taken by her mum to the neighbor it will be compelled to meet the children of that neighbor; an author might be more acquainted with the work of his friends and colleagues and may cite it more often than that of others; a protein's interaction pattern may be influenced by the hormones in the cell in which it is expressed, and so forth.

We will call a network *centrally organized* if there is one single network–generating entity that coordinates the network's building process. A typical centrally organized network is the network of streets in a city, governed by the administration of that city. A network built by many network–generating entities will be called a *decentrally organized network*. A typical example of such a network is the network of flights around the globe since all the airlines decide independently (but influenced by the decisions of others) what flights to offer. We will call a network *self-organized* if the network–generating entities are generating relationships between themselves, as sketched in Fig. 5.1.

In summary, a *network–generating system* consists of four components:

1. the environment;

2. a set of network–generating entities;

3. a set of objects that are related to each other by the network–generating entities. This set may be a subset of the set of network–generating entities;
4. a network–generating process;

The *network–generating process* can be expressed by a mathematical function that describes the probability that a network–generating entity builds a relationship either to others or between two objects as a function of environmental incentives or pressures, its own properties and the properties of others or the objects. If neither environmental properties nor properties of the entities or objects influence the process, we have as a special case the description of a classical graph family. Thus, our model of network–generating systems is a generalization of the more classical descriptions of graph families that allows us to model more closely real–world networks and their building processes.

What we have today is mainly a categorization of real–world networks by their structural properties. Our aim is to change this paradigm into a categorization by the network–generating processes that resulted in this network. In the following we will argue why an understanding of the network–generating process within a network–generating system is important to the application and development of algorithms on real–world networks. We start with a discussion of how knowing the network–generating process in a network–generating system can help to steer complex network evolution.

5.1.1 Steering Complex Network Evolution

In the last chapter we analyzed how to build networks showing the small–world phenomenon. We have indicated that, by choosing the non-random graph part carefully, many different structural properties such as a scale-free degree distribution [21], a certain assortativity [177], or local properties such as the (k, l)-property introduced by Chung et al. [53] (s. 4.2.3), can be designed into the graph. These network models are helpful in traditional network design, where there is full control over the design process of the whole network. One of the main perspectives of this thesis is that this design paradigm is going to shift in many areas. Due to the globalization process, many networks are not (any more) governed by a central authority, e.g., the network of flights governed by dozens of different airlines, or peer-to-peer networks in which every user decides when to participate and when to leave. Nonetheless, the structure of these networks has a global impact of the functionality of the whole system: it has been shown that the network of flights makes our world a small–world with important implications to the spreading of diseases [139, 191]. Similarly, the network structure of peer-to-peer networks influences the overall speed with which news can be transported over the net. This implies that although the networks grow decentrally it might be advantageous to steer the network–generating process by incentives such that the single network–generating entity is guided into making the globally 'right' decisions.

If, for example, it turns out that vertices with a high fitness are more likely to get a new edge and furthermore that their fitness is increased by a new edge, this implies that the resulting network will have a scale–free degree distribution. As is known from a famous paper by Albert et al. [6], scale–free networks are much more prone to attacks than random graphs (s. 6.6.1), and might thus not be a desired network structure for sensible communication networks. A simple example is the international network of flights, where some airports are major hubs, connecting continents with each other, whereas most airports are serving only domestic flights. Here the only choice for reacting to the problematic network structure that could be severly disrupted by just a few attacks is to enhance security on the hubs. In other cases it might be possible to change incentives or pressures to steer the network–generating entities. In peer-to-peer networks there is a similar

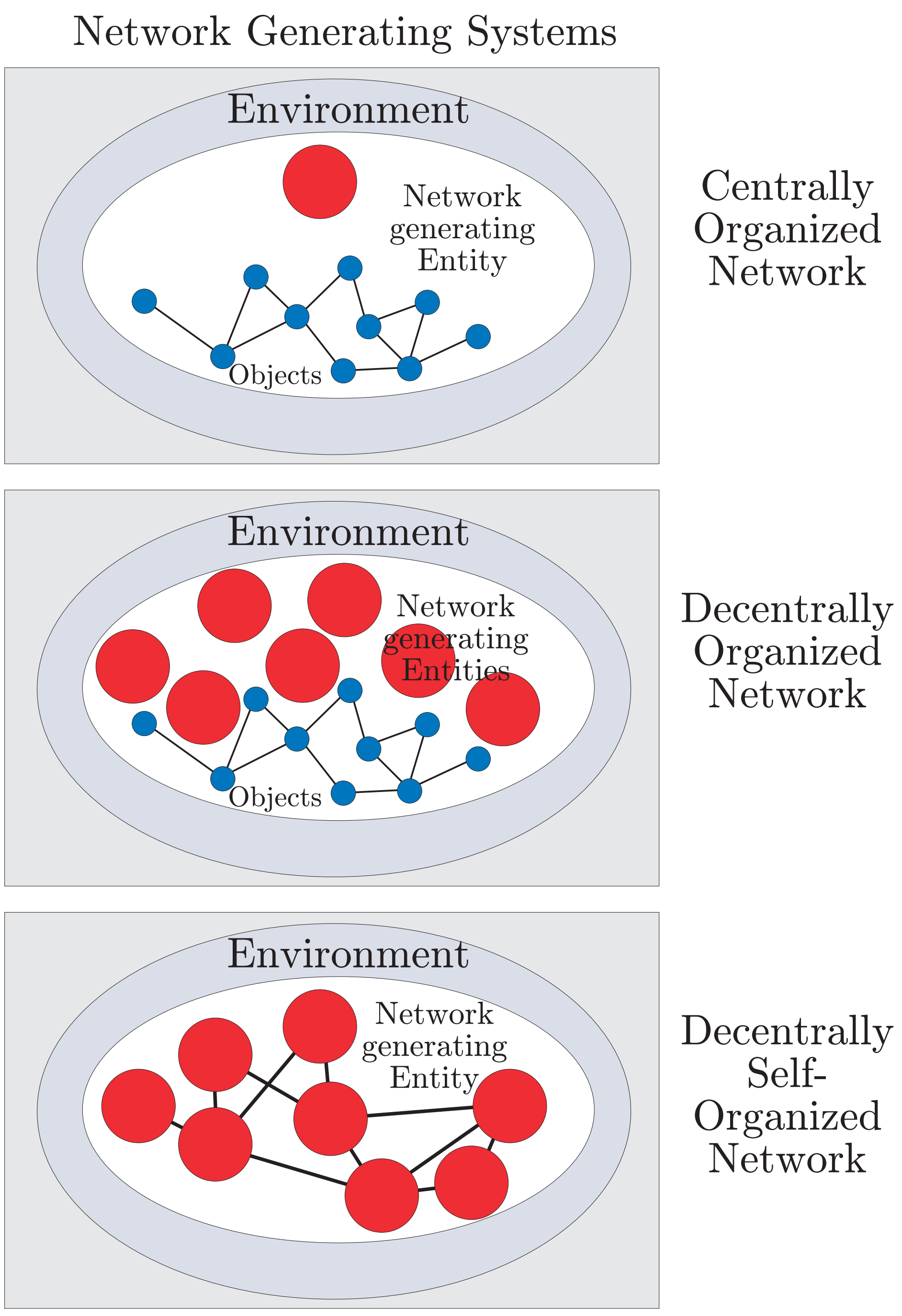

Network Generating Systems
Environment
Network generating Entity
Objects
Centrally Organized Network
Environment
Network generating Entities
Objects
Decentrally Organized Network
Environment
Network generating Entity
Decentrally Self-Organized Network

Fig. 5.1: We differentiate three main types of network–generating systems: **centrally** organized networks where only one entity decides who is connected to whom, and **decentrally** organized networks in which many entities decide independently whom they relate to. If the network connects the entities themselves, we speak of decentrally **self–organized** networks.

problem, namely that of free–riding, i.e., where a user will only download files but never help to distribute them. Here, some protocols try to punish free–riders and push them to the outskirts of the net and maybe even isolate them, making it hard to get any more files out of the system unless they cooperate, e.g., [47].

Our claim here is that the whole network–generating system must be known to come up with the right incentives or pressures to steer the network–generating process into the desired direction, as is the case with all trials to steer a complex system [17, 67, 233]. We will discuss this point more closely for the case of decentrally self-organized networks in chapter 6.

Another point for why understanding the whole network–generating system and its network–generating process is important, is that complex network analysis is often context–sensitive, as we will discuss in the following.

5.1.2 Contextual Algorithms

For a book project [41] we spent some time reviewing the literature on *centrality indices.* Centrality indices were introduced to find out who is the most important actor in a given complex network [237]:

> One of the primary uses of graph theory in social network analysis is the identification of the "most important" actors in a social network. ... a variety of measures designed to highlight the differences between important and non-important actors have been offered by many writers. All such measures attempt to describe and measure properties of "actor location" in a social network. [237]

One question whose answer has been attempted in this ways is: Who is the most important Florentine family in medieval Florence, as indicated by the information regarding which family was connected by marriage with which other family [237]? The question can be answered by looking at the vertex with the highest degree, which is, not surprisingly, the Medici family. This measure is called the *degree centrality.* One can also argue that if marital ties are used to gather and spread information, the importance of a vertex is inversely proportional to its maximal distance to other families in the network. This is computed by the so-called *eccentricity* that will also play a role in chapter 6. Both centrality indices can be applied to the graph of marital ties, but they give different answers to the same question. Thus, which one is 'right'? As we have argued in [133], influenced by a paper by Borgatti [37], centrality indices can only be reasonably applied if the process that made them and the process that uses them is known: if the network displays something like a voting process, certainly the agent with most votes (i.e., edges) is the most important player. If the network displays a hierarchy it might actually be the one with the least number of edges because she is at the top of a pyramid.

Thus, the context, i.e., how the complex **network** represents the complex **system** that is analyzed, is an important part of the input for deciding which algorithm to apply to solve it. Knuth defines an algorithm as a deterministic problem–solving process with five properties: it has *input* and *output*, which stand in a *defined relationship*, it is *finite*, and it is *efficient* [130]. The problem with centrality indices is now that every one of them is defined on undirected, connected graphs, no matter what the *significance* of the edges for the complex system at hand, but not every one can reasonably be applied to solve the question on the complex system level. Thus, we want to introduce a sixth property of — mainly analytical — algorithms, that is, the *context* in which it can reasonably be applied to a complex network. An algorithm that should only be applied in a given context is then defined as a *contextual algorithm.*

A second set of *contextual algorithms* are the so-called clustering algorithms. A *clustering algorithm* computes subsets of vertices that form densely connected subgraphs in a given graph. There are different clustering quality measures for any given partition of a graph (s. [89] for an overview), and dozens of algorithms to compute clusterings, e.g., [180, 93, 188, 204, 201]. All of these methods claim that the subgraphs computed by their method will reveal *functional modules* in the given graphs. And indeed, from metabolic networks [201], to different kinds of social networks [180, 93], or protein networks [188], the computed subsets could be shown to contain vertices that represent somewhat similar objects, e.g., metabolites of a certain metabolic path, scientists of the same community, or proteins with the same function in the cell, e.g., support of the transcription and subsequent translation of DNA to proteins. These are astonishing examples, and it is reasonable that when a new protein is found in an organism, and a clustering algorithm is run, the group into which it falls may predict its function in the cell.

There is also a range of different network models that are built on the assumption that an edge is built mainly between similar or somewhat near objects. Because these models are widely accepted, they also indicate that locality is a main network–generating process and thus that clustering algorithms may be applied to them. We will briefly review some of these models and findings:

1. Since sensors are limited in their transmitting radius, sensor networks are often modeled by *unit-disk graphs*, where each vertex is connected to all vertices within the transmission radius that is common to all, or *knn*-graphs as indicated in the last chapter [49, 197, 75, 74]. These are very strong local network models because there is a threshold distance such that any edge with a greater distance does not exist.

2. Other models assign vanishing probabilities to long-range edges, e.g., with an exponentially decaying probability $P(e)\ e^{-d(e)/d_0}$ for some constant d_0 in the Waxman model that is intensively used for network generators that were proposed to simulate the Internet's network topology [243].

3. Kleinberg analyzes a model in which vertices are embedded in a hierarchy and the probability that they are connected by an edge increases as their distance in that hierarchy decreases [127].

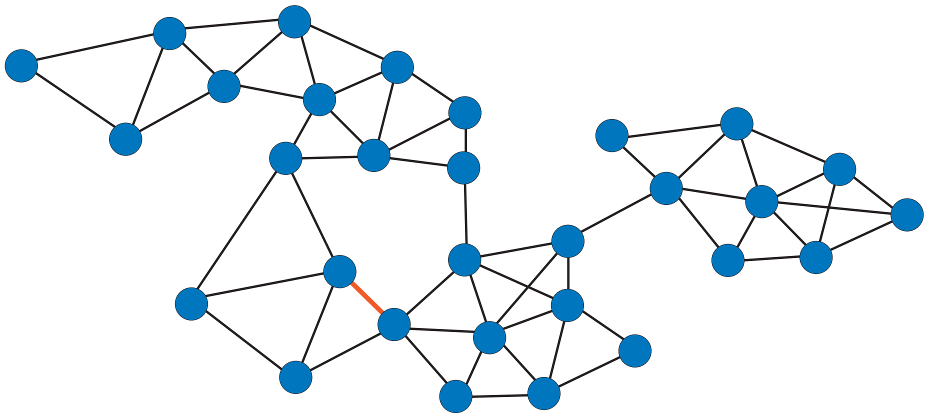
4. As Yook, Jeong, and Barabási showed, it is more likely that the Internet's preference for local edges is less pronounced and more likely to decrease linearly with d [254]. The model proposed by these authors to capture the essential structure of the Internet is then a mixture of the preferential attachment model and a locality preferring model.

5. A similar, more generalized model was also analyzed by Barthélemy who comes to the conclusion that when edge formation is costly, most network structures, e.g., diameter, clustering coefficient, assortativity, are then dependent on the vertex density [26].

But in fact all of the clustering methods will also find a clustering in a random graph, where none of the subsets found will show any common functionality. We do not want to argue that clustering techniques should not be applied to real–world networks, but we do argue that this will only make sense if:

1. there is something such as a *distance* or *similarity* measure between the objects;

2. an edge is the more likely between two objects the more similar they are.



(a)

Fig. 5.2: This is a local graph that is perfectly local in the sense that for the given embedding of the vertices only short edges are built. Still, the graph does not show a high locality in the sense of the (k, l)-measure introduced by Chung and Lu [53, 12]. A graph is said to be (k, l)-local if for every edge there are at least k vertex–independent paths of at most length l that connect their endvertices. In this case, due to the red edge, there is no $(k, 4)$ locality for any k because the smallest alternative to this edge is the path with length 5. For the same edge, there is also only one path with length five, thus prohibiting $(k, 5)$-locality for all values $k > 1$.

The question is now whether given only the adjacency matrix of a network, can we **measure** whether edges are drawn at random or are biased, i.e., that 'short' edges are preferred in a way? As we discussed in the last chapter, it seems difficult to measure *locality* as defined by Definition 4.2. We have already shown that the clustering coefficient can fail to indicate locality for — admittedly artifical — distributions of vertices such that there is no strong correlation between a network–generating process that prefers to build small edges and a high clustering coefficient. Also the measure introduced by Chung and Lu, the (k, l)–locality can be tricked by certain distributions of vertices: If the edge density varies strongly such that big holes emerge without any vertex, there can be pairs of connected vertices that do not have any alternative path smaller than l, as shown in Fig. 5.2. The problem seems to be that to discover locality means to measure two things at the same time: the distribution of vertices in space that determines their distance to each other **and** the propensity to build local edges with respect to this embedding. This seems to be as difficult as to see the real identity of objects in the shadowy interior of Plato's cave.

Since detecting locality seems to be too difficult, we asked ourselves whether it is possible to detect the opposite, namely a random selection of edges that is mixed with some other network–generating process. This idea is discussed in the following section.

5.2 Bounding the Number of Random Edges in a Graph

The setting is as follows. Let $\mathcal{S}$ denote any network–generating system. Let $\mathcal{P}_{SW}$ denote some small–world generating process as described in the last chapter, i.e., one that is composed of two different processes, where $\mathcal{P}_{rand}$ denotes the generation of random edges, each with probability p. Given **only** the adjacency matrix of the resulting network, are we able to measure p? If so,

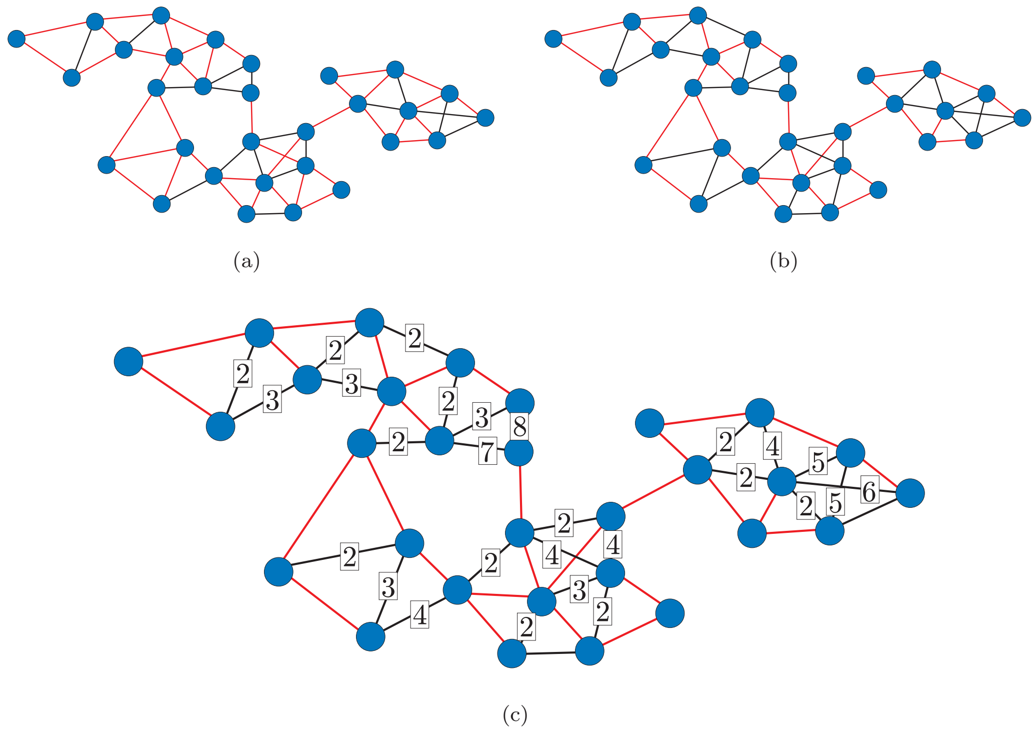


Fig. 5.3: (a) Highlighted by red edges is a subgraph of G. (b) Highlighted by red edges is a spanning tree of G. (c) For the given spanning tree in (b) the distances it defines for the non-tree edges are given.

this is a first step in comparing two given graphs with respect to their locality (or at least bias towards certain edge classes). For example, if two graphs have about the same number of vertices and edges, but one contains a random graph component with $p = 0.1$ and the other contains one with $p = 0.2$ the latter can be said to be 'more' random and thus less local than the other. In the following we will show that it is possible to measure the parameter p of a random edge generating process by introducing a new analytical method, the so-called *backbone distance distribution*.

Let $G(n, p)$ be a connected instance from the $\mathcal{G}(n, p)$ family. Let $G' = (V, E' \subseteq E)$ be any *spanning subgraph* of this graph, i.e., the edge set E' constitutes a connected component on V, as depicted by the red edges in Fig. 5.3.

Suppose that we only know the edge set E' but not the full set of edges E and the parameter p of the random graph. We will now determine the probability with which any pair of vertices that is **not** in E' is in E. Henceforth, a pair of vertices of which we have no knowledge about whether it is connected in G or not, will be called a *possible edge*. The first observation is that in a random graph every possible edge has the same probability of existing; there is no bias towards any of them even if E' is a randomly chosen subset of E. Of course, the new probability p' with which one of those edges exists is diminished for all $p < 1$:

$$p' = \frac{n(n-1)p/2 - |E'|}{n(n-1)/2 - |E'|}. \tag{5.1}$$

Let now the distance of any two vertices v, w in the subgraph G' be denoted by $d_{G'}(v, w)$. To any possible edge $e = (v, w)$ we will assign the distance of its endvertices in the subgraph, i.e.,

$d_{G'}(e) = d_{G'}(v, w)$. If G' is uniquely identified by the context, we will discard the index. Starting at distance $k = 2$, the distribution of the number of possible edges with this distance can now be computed. This distribution is called the *possible distance distribution.* Since every one of these possible edges e has the same probability p' of being realized, the number of realized edges with this distance will be proportional to the number of possible edges in this distance. The corresponding distribution is called the *realized* or *absolute distance distribution.* Dividing the number of realized edges in this distance by the number of possible edges in this distance will thus rediscover probability p'. The corresponding distance distribution is called the *relative distance distribution* for the given subgraph.

To come as close as possible to the 'real' probability p with which the random graph was built, it is of course helpful to reduce the set of edges in E' as far as possible, i.e., by building a spanning tree (3.5); see Fig. 5.3(b). Then, Equ. 5.1 reduces to expectedly

$$p' = \frac{np - 2}{n - 2}. \tag{5.2}$$

If the chosen subgraph is a spanning tree we will in general speak of *(possible/realized/relative) tree distance distributions.* As stated above it is necessary that the spanning tree not be biased in any way. There are several methods to compute spanning trees uniformly at random [200, 248], e.g., by a Markovian process [45]. Here we use a variant of a method that was designed to build minimal spanning trees, called the *Kruskal algorithm* [57]. To construct a minimal spanning tree by this method, edges are sorted non-increasingly by their weight, and recursively added to the growing tree if they do not induce a cycle. If the edge set is instead shuffled, the method is called *randomized Kruskal* and will of course not generate a minimal but just an ordinary spanning tree. Although this method will not generate each spanning tree with the same probability but prefers path–like spanning trees [200] this is not a problem since we just need a spanning tree that will not bias edges within a certain distance. Furthermore, spanning trees with a high diameter yield many tree distance classes which helps to fit a function to the resulting tree distance distribution. Fig. 5.4 shows relative tree distance distributions of some spanning trees in random graphs from the $G(n, p)$ model with different values of the parameter p, ranging from 0.1 to 0.8. The lines indicate p' and it is clear to see that the relative tree distance distribution is very well described by these values.

So far we have shown that the (unknown) parameter of a pure random graph can be computed by this method. In the next section we will show how this method can be used to bound the number of random edges in a hybrid graph model, i.e., one that is composed of a non-random graph component and a random graph component.

5.2.1 Bounding the Number of Random Edges in a Hybrid Graph Model

To show that this method can also help to detect a random graph component superimposed on a different network, we will use a simple hybrid graph model that is based on a grid and a superimposed random graph from the $\mathcal{G}(n, p)$ family. In the experiments, we constructed grids with 20×20 vertices, resulting in 760 edges to which a random graph from the $G(n, p)$ model was added. Fig. 5.5 shows the resulting relative tree distance distributions for the pure grid with $p = 0$ and $p = 0.001, 0.005, 0.01, 0.05$, i.e., expectedly 80, 400, 800, and 4000 added random edges. The relative tree distance distribution of the grid alone is quite steep as can be seen in Fig. 5.5(a), but note that of course no cycles of odd length can emerge. We have thus skipped the even tree distances for this data set to make the fitting meaningful. How then can a random graph on top of this grid be detected by the method? The idea is that the grid alone produces a steeply

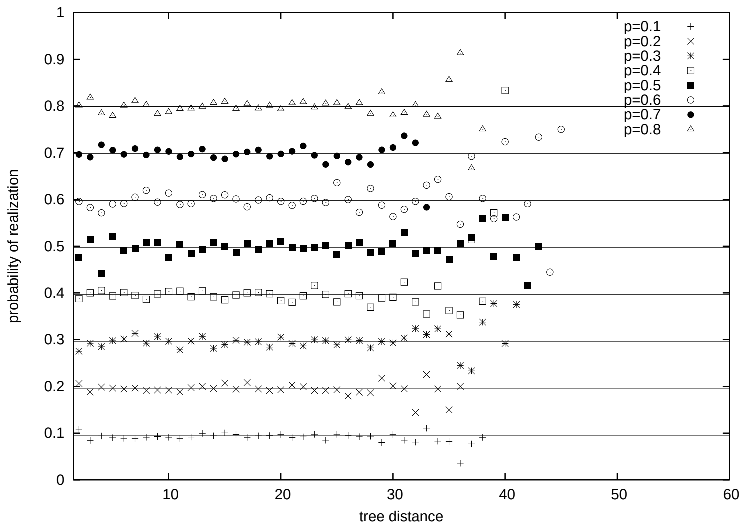


Fig. 5.4: Every data set shows the relative tree distance distribution ($rtdd$) of one random graph with 400 vertices and p as denoted. The lines denote p' as given in Equ. 5.1. The correspondence between the $rtdd$ and p' is very clear, as expected.

declining relative tree distance distribution, while the random graph component adds a constant distribution with value p. This is of course a bit naive since the added random edges will also change the structure of the spanning tree: if a random edge is incorporated into the spanning tree, it is clear that some of the local edges will now have a large tree distance. We thus expect that the offset by which the relative tree distance distribution is elevated is actually larger than p. In any case, the offset will give an upper bound on p.

One method to compute the offset is to fit a function $f(x) = g(x)+b$, $b \in \mathbb{R}$, to the distribution and take the value of b as the offset induced by a random graph component. However, sometimes the standard error of this offset will be in the scale of b itself, and then this method is questionable. We will thus give another, more strict method to compute b. To do so, we need some further definitions: We will call the quotient of edges in a given tree distance k and the possible number of edges in that tree distance the *realization factor* ρ_k of that tree distance. The realization factor in tree distance k is thus nothing else than the parameter p of a random graph that has realized exactly the expected number of edges in this tree distance. Of course, since these graphs are finite, there will always be large tree distances in which nearly no edge is realized but in which the possible number of edges to be realized is also very small. Thus, the realization factor computed in these large tree distances has a large standard deviation compared to the mean and may not be taken as an upper bound on p, implying that it is not reasonable to take the minimum of all these realization factors as an upper bound on p.

Rather, we have to take into account the probability that the *real* realization factor ρ^*, i.e., the real parameter p of the random graph component, is much higher than ρ_k but has nonetheless realized only $\#_r(k)$ of the $\#_p(k)$ possible edges. The random variable X of edges realized in a given tree distance with a realization factor of ρ^* is just the outcome of the simple Bernoulli experiment in which each of the possible edges $\#_p(k)$ is realized with probability ρ^*. We will now see each tree distance as the outcome of a single Bernoulli experiment with an unknown realization factor $\rho^*_k := f \cdot \rho_k$ for some constant $f > 1$. The question we want to answer is then: What is the highest $\rho^*_k > \rho_k$ such that the probability that the deviation of $|X - \rho^*_k \cdot \#_p(k)|$ is greater than

$f \cdot \rho_k \cdot \#_p(k) - \#_r(k) = (f-1)\#_r(k)$ is less than, e.g., 1%? With this question we bound the real realization factor such that it is very unlikely that ρ^*_k is even higher but only produced $\#_r(k)$ edges. And subsequently, the minimal ρ^*_k can then be used as an upper bound on ρ^*.

Let $\#_p(k)$ denote the number of possible edges and $\#_r(k)$ the number of realized edges in tree distance k for a given spanning tree and let ρ_k denote the realization factor of this tree distance, i.e., $\rho_k = \#_r(k)/\#_p(k)$. We will now bound the probability that ρ_k is actually f times as high for some $f > 1$ such that $f \cdot \rho_k \leq 1$, when only $\#_r(k)$ edges were realized. The expected value of realized edges if the realization factor is $f \cdot \rho_k$ is of course $f \cdot \rho_k \cdot \#_p(k) = f \cdot \#_r(k)$ and its standard deviation σ is given by $f \cdot \rho_k(1 - f \cdot \rho_k)\#_p(k) = f(1 - f \cdot \rho_k)\#_r(k)$. By Chebyshev's inequality, the probability that the number X of realized edges deviates by more than $t = f\#_r(k) - \#_r(k) = (f-1)\#_r(k)$ from the expected value $\mu = f\#_r(k)$ is given by:

$$\begin{aligned} P[|X-\mu| \geq t] &\leq \sigma^2/t^2 && (5.3)\\ &= \frac{f(1-f\cdot\rho_k)\#_r(k)}{(f-1)^2\#_r(k)^2} && (5.4)\\ &= \frac{f(1-f\cdot\rho_k)}{(f-1)^2\#_r(k)}. && (5.5)\end{aligned}$$

We equate this with 0.01 and compute the minimal f for each tree distance such that the probability that for $f \cdot \rho_k$ only $\#_r(k)$ edges are realized is at most 1%. From all $f \cdot \rho_k$ values we take the lowest value as an upper bound on ρ^*, the real parameter p of the random graph component. f is the solution of a quadratic equation as follows:

$$\begin{aligned} & \frac{f(1-f\cdot\rho_k)}{(f-1)^2\#_r(k)} = 0.01 && (5.6)\\ \Rightarrow\quad & (f-1)^2\frac{\#_r(k)}{100} - f(1-f\cdot\rho_k) = 0 && (5.7)\\ \Rightarrow\quad & f^2\left(1+\frac{100\rho_k}{\#_r(k)}\right) - f\left(2+\frac{100}{\#_r(k)}\right) + 1 = 0 && (5.8)\\ \Rightarrow\quad & f_{1,2} = \frac{2+\frac{100}{\#_r(k)}}{2\left(1+\frac{100\rho_k}{\#_r(k)}\right)} \pm \sqrt{\left(\frac{2+\frac{100}{\#_r(k)}}{2\left(1+\frac{100\rho_k}{\#_r(k)}\right)}\right)^2 - \frac{1}{1+\frac{100\rho_k}{\#_r(k)}}} && (5.9)\end{aligned}$$

We will first analyze both methods to compute an upper bound on p on simple model introduced above of a grid graph plus superimposed random graph (Fig. 5.5).

To each of the relative tree distance distributions we fitted a function of the form $f(x) = a/x^b + c$. The fit is given as an inset in each of the diagrams, where each parameter a, b, c is accompanied by its standard error. In the case of the pure grid (Fig. 5.5(a)), the offset is 0.01, bounding the number of random edges in this graph to 80, i.e., around 10% of all edges. Of course, this is an overestimation since we know that there are no random edges in this graph. Also for the superimposed random graphs with $p = 0.001, 0.005, 0.01$, the offset overestimates p with 0.0044, 0.0085, and 0.014, bounding the number of random edges to around 350 instead of 80, 680 instead of 400, and 1120 instead of 800. For the densest random graph with $p = 0.05$ that results in more edges than the underlying grid, the offset bounds the number quite correctly to around 4320 edges instead of 4000. In summary, the offset of a fitted function will overestimate the number of random edges, but the higher the random graph part is, the better the approximation. Note also that b grows with the number of added random edges: where the fitted function was approximately inversely proportional to the square of k in the pure grid, it is 3.16 for $p = 0.001$ and 4.31 for $p = 0.005$. For

$p = 0.01$ and $p = 0.05$ the values are 6.72 and 2.31, but in both cases a has also strongly changed, i.e., it looks as if both of these distributions were best fitted by a constant, but have to allow for the first values that deviate strongly from this constant.

In these cases the standard deviation of parameter c was very low compared to c, and thus it can be seen as quite reliable. Nonetheless, we also want to compare the values with that of the minimal ρ^*_k. In the case of the pure random graph, the minimal ρ^*_k value is 0.058, bounding the number of random edges to 4640, i.e., more than the grid graph has. Also in the second graph (Fig. 5.5(b)), the lowest ρ^*_k value is 0.040, overestimating the correct parameter by a factor of 40. For the third graph (Fig. 5.5(c)), the lowest ρ^*_k value is 0.044, overestimating the correct parameter by a factor of 8.8. For the fourth graph with a real parameter of $p = 0.01$, the lowest ρ^*_k value is 0.052, overestimating it by a factor of 5.2, and for the fourth graph with a real parameter of $p = 0.05$, the lowest ρ^*_k value is 0.10, overestimating it by a factor of 2. The ρ^*_k value overestimates by far the number of random edges in all of these cases, although as with the offset given by the fit, the relation between the correct value and the estimation becomes better for large p. In this case, the reason for the failing of the ρ^*_k method is just that the underlying graph is very small with only 760 edges. In every tree distance greater than 17, only 10 edges or less were realized, making the estimation of ρ^*_k difficult, and of course, in the lower distances the edges of the underlying grid are prevalent and thus their ρ^*_k value does not say much about the random graph component. In an underlying grid with $4,900$ edges the lowest ρ^*_k value already drops to 0.0049. If such a graph is superimposed by a random graph with $p = 0.001$, the lowest ρ^*_k value drops to 0.0044, overestimating p by only a factor of 4.4. This is mainly due to the much larger number of possible edges in a large tree distance. If this is large, but only very few of them are realized, Chebyshev's inequation is good enough to bound ρ^*_k reasonably. Thus, for an underlying grid of $19,800$ edges, the lowest ρ^*_k finally drops to $6.69 \cdot 10^{-4}$.

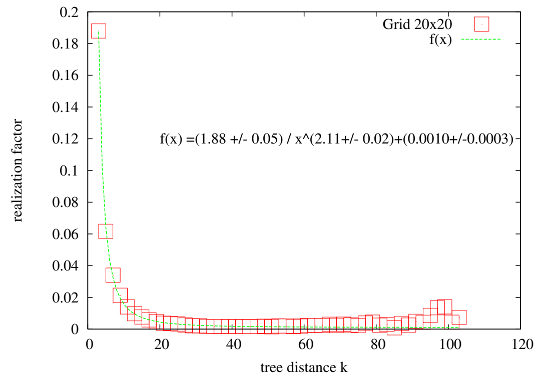
In summary, the parameters of the fitted function can be used as an upper bound if the fit is reasonably close to the distribution as indicated by the standard error. ρ^*_k overestimates the real parameter by more than a reasonable fit, but will in each case yield a valid upper bound on the number of random edges. Although future research is needed to improve the second method, we will now show that in most cases both methods yield reasonable bounds of the number of random edges in real–world networks.

5.3 The Relative and Absolute Tree Distance Distribution of Real–World Networks

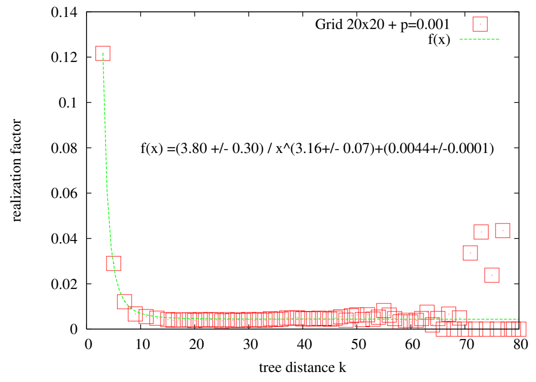
In the following we have analyzed the relative tree distance distribution of various real–world networks by computing 50 randomized Kruskal spanning trees and averaging their tree distance distributions. Figs. 5.6, 5.7 and 5.8 show the relative tree distance distributions of these networks[4] and some fitting function for the data. As can be seen at a first glance, some of the distributions look quite different to the simple ones above, and thus the fitting could not be made with the same function for all graphs.

Fig. 5.6(a) shows the word–association graph that was provided by Palla et al. in their software package *CFinder* [62, 188] (s. 8.1.4). To construct this network, people were given a word and asked for a spontaneous association they had with the given word. The network represents words by vertices and connects them with a directed edge if the word associated with the first vertex was associated with the word represented by the second vertex. The edge is weighted by the number

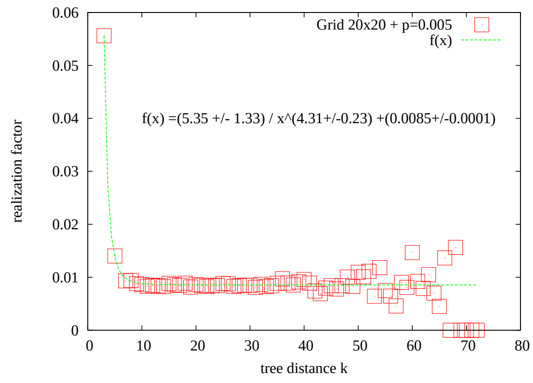
[4] The details concerning this data can be found in 8.1.



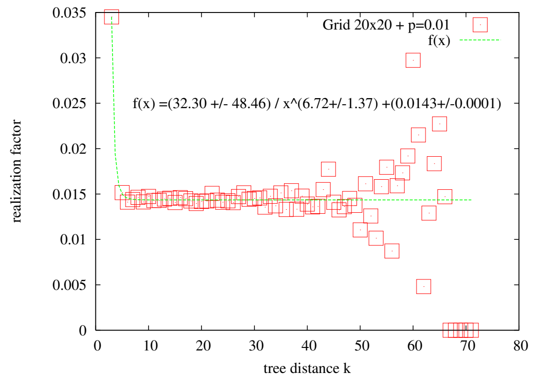
(a) pure grid



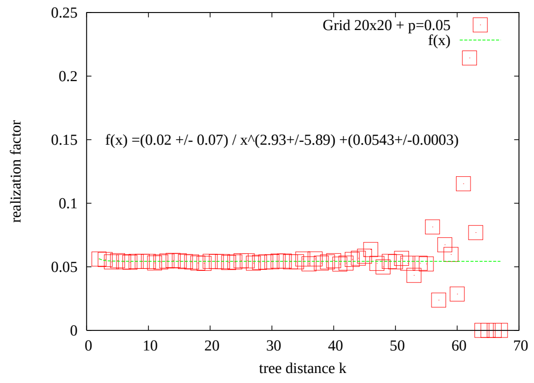
(b) + G(n,p) with p=0.001



(c) + G(n,p) with p=0.005

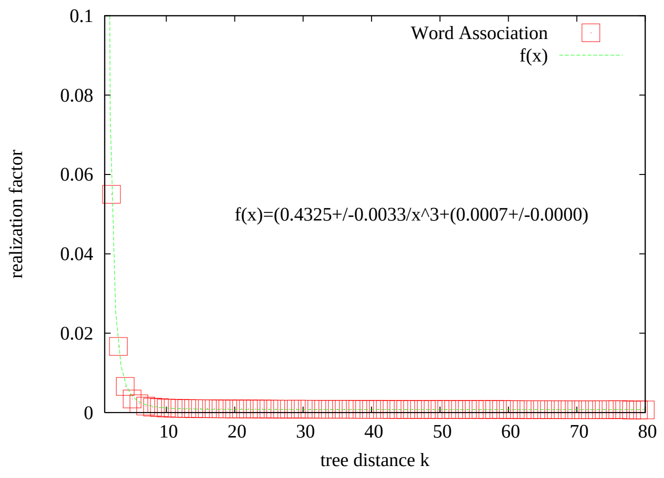


(d) + G(n,p) with p=0.01

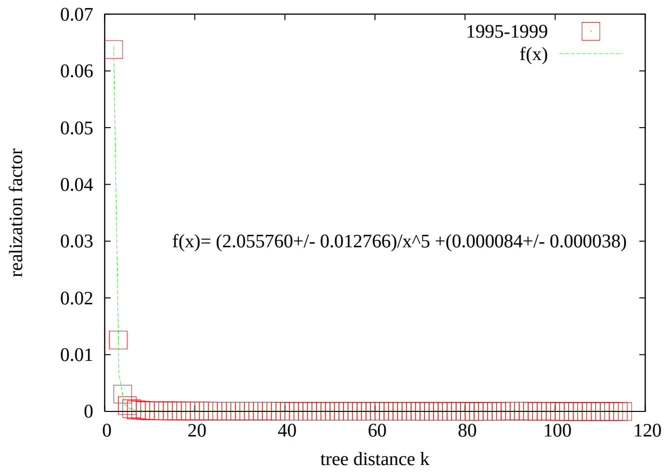


(e) + G(n,p) with p=0.05

Fig. 5.5: Relative tree distance distributions of graphs composed of a grid with 20×20 vertices and a random graph with different values of p as indicated. The higher the number of random edges in the combined graph, the higher the offset from the x-axis, giving an upper bound to the amount of random edges in an unknown graph.



(a) word–association graph

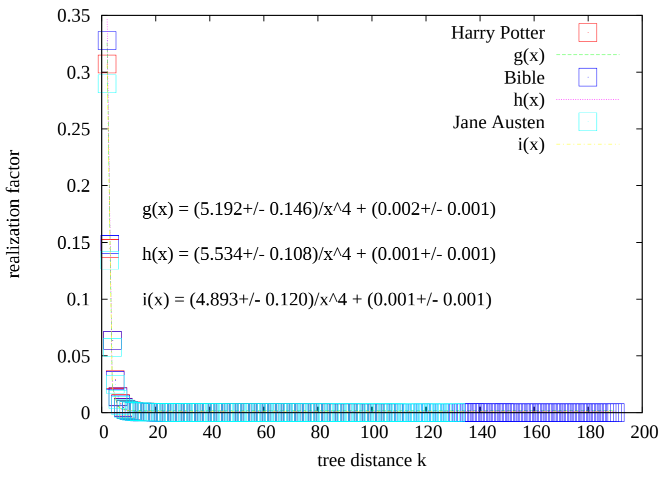


(b) co–authorship network

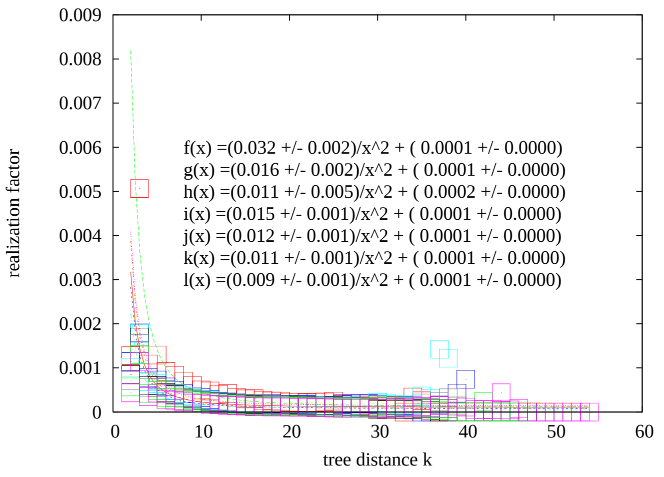
Fig. 5.6: Relative tree distance distributions of various real–world networks (s. 8.1 for details on the data). The findings are discussed in the text.

of these associations. However, here we use the unweighted, undirected version of the network. As can be easily seen, the relative tree distance distribution is far from random and could be very well fitted by a function that is inversely proportional to k^3, with an offset of $7.3 \cdot 10^{-3} \pm 4.2 \cdot 10^{-4}$. Since the standard error is around 6% of that value, this seems to be acceptable as an upper bound to the random graph component, yielding expectedly 18947 random edges, i.e., 60% of all edges. The lowest ρ^*_k value is 0.00125, as expected higher than the value obtained by the fit, bounding the number of random edges to 32445, i.e., more than all the edges that are actually in the graph.

Fig. 5.6(b) displays a co–authorship network, as compiled by Newman and kindly offered for further analysis on his website [176, 179] (s. 8.1.3). Newman analyzed all publications published online at the arXiv [251] in the section *condensed matter*, in the interval from $1995 - 1999$. Vertices represent authors and two vertices are connected by an edge if there is at least one publication in that interval published by them as co–authors. The network shows an extremely steep relative tree distance distribution that is fitted very well by a function that is inversely proportional to k^5. The offset is given by $8.4 \cdot 10^{-5}$, bounding the percentage of random edges to 33% of all edges. Again, the lowest value of ρ_k^* is given by $3.31 \cdot 10^{-4}$ and thus higher than the one given by the offset, bounding the number of random edges to 31797 edges, to at most 70% of all edges in the graph.



(a) co–purchasing network of books



(b) autonomous system network

Fig. 5.7: Relative tree distance distributions of various real–world networks (s. 8.1 for details on the data). The findings are discussed in the text.

Fig. 5.7(a) shows different co–purchasing networks obtained from the online shop amazon.com [1] (s. 8.1.1). We started a crawl at different books and followed the links under the title 'customers who bought this book also bought' until a given depth, in this case 9. The data titled 'Harry Potter' started at 'The Order of the Phoenix', the data titled 'Luther' started at some edition of the Lutheran Bible, the data titled 'Jane Austen' started at some edition of 'Pride and Prejudice'. Here, the fitting function had the form $a/k^4 + b$, with low offset values for all three data sets. According to these values, up to $63\%, 55\%$, and 99%, respectively, of all edges are random in the three data sets. All of these values are problematic since the standard error of the offset is in all cases around 50% of the value itself. In this case, the more rigid method will obtain better bounds because the fit cannot find the offset with a high certainty: for the first data set, the smallest ρ^*_k value is 0.00183, bounding the number of random edges to 4253, i.e., approximately 70% of all edges. For the second data set, the smallest ρ^*_k value is $5.15 * 10^{-4}$, bounding the number of random edges to 7458 edges, i.e., to 35% of all edges. For the third data set, the smallest ρ^*_k value is $5.3 \cdot 10^{-4}$, bounding the number of random edges to 7851, i.e., 48% of all edges. Note, however, that the values obtained by the second method are within the bounds defined by the standard error.

Fig. 5.7(b) shows different snapshots of the autonomous system network of the Internet, taken every six months in the interval from January, 1998 to January, 2001. The fits to the data sets are chronologically sorted, i.e., the first fit describes the first data set from January, 1998. An autonomous system is a group of routers with the same IP-address. Routing is mainly done on this level, and their physical links provide the backbone of the Internet. The exact details of how they are anaylzed and constructed can be found in 8.1.2. The data is fitted best by inversely quadratic functions, as shown in the diagrams, but only if the range of tree distance $2 - 4$ is disregarded. The reason for this is best seen in the earliest data set: tree distance 2, i.e., triangles, are much less realized than expected, then there is a peak at tree distance 3, and again a drop in tree distance 4. Note that tree distance 3 signifies that there is a four-cycle of which three edges are in the tree. Our intuition is that in the early days of the Internet routers were more or less connected grid like and people tried to avoid connecting to other routers that were too near, i.e., they tried to connect autonomous systems such that the overall network had a low average distance. We think that in time different network–generating processes were at work, where the first network–generating process was more centrally organized (avoiding connections between routers already well connected), whereas later connections were more decentrally organized. In any case, these fits have low standard errors and thus we will consider them as an upper bound on p. For each of the data sets, the following bounds on the number of random edges can be obtained:

1. **March 1998**: The offset is $1.15 \cdot 10^{-4}$, bounding the percentage of random edges to merely 9%. The lowest ρ^*_k value is given by $8.22 \cdot 10^{-4}$, thereby bounding the number of random edges to expectedly 5130, i.e., to 64% of all edges.

2. **September 1998**: The offset is $1.36 \cdot 10^{-4}$, bounding the percentage of random edges to 15%. The lowest ρ^*_k value is given by $5.41 \cdot 10^{-4}$, thereby bounding the number of random edges to expectedly 4751, i.e., to 60% of all edges.

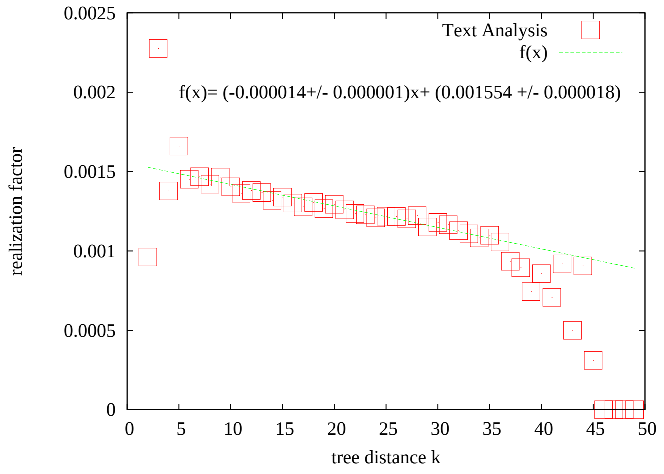
3. **March 1999**: The offset is $1.86 \cdot 10^{-4}$, bounding the percentage of random edges to 23%. The lowest ρ^*_k value is given by $4.66 \cdot 10^{-4}$, thereby bounding the number of random edges to expectedly 5379, i.e., to 58% of all edges.

4. **September 1999**: The offset is $6.5 \cdot 10^{-5}$, bounding the number of random edges to 10%. The lowest ρ^*_k value is given by $4.05 \cdot 10^{-4}$, thereby bounding the number of random edges to expectedly 6706, i.e., to 59% of all edges.

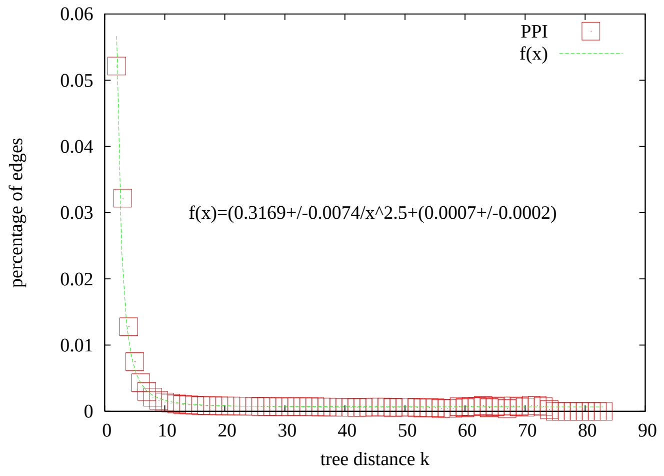
5. **March 2000**: The offset is $1.06 \cdot 10^{-4}$, bounding the percentage of random edges to 18%. The lowest ρ^*_k value is given by $3.55 \cdot 10^{-4}$, thereby bounding the number of random edges to expectedly 8922, i.e., to 60% of all edges.

6. **September 2000**: The offset is $7.8 \cdot 10^{-5}$ bounding the percentage of random edges to 16%. The lowest ρ^*_k value is given by $2.66 \cdot 10^{-4}$, thereby bounding the number of random edges to expectedly 10022, i.e., to 55% of all edges.

7. **March 2001**: The offset is $8.8 \cdot 10^{-5}$ bounding the percentage of random edges to 22%. The lowest ρ_k^* value is given by 0.00021, thereby bounding the number of random edges to expectedly 11733, i.e., to 53% of all edges.



(a) word adjacency



(b) protein–protein interaction graph

Fig. 5.8: Relative tree distance distributions of various real–world networks (s. 8.1 for details on the data). The findings are discussed in the text.

Fig. 5.8(b) shows a *protein–protein interaction* network of the organism *Saccharomyces cerevisiae*; this network is the same as that used by Palla et al. and can be obtained together with the authors' software *CFinder* to find k-cliques [62, 188] (s. 8.1.4). Its relative tree distance is fitted by a function that is inversely proportional to the cube of the tree distance. The offset of this function is $6.56 \cdot 10^{-4}$, thus bounding the percentage of random edges to 31%. Since the standard error of the offset is quite high, it is questionable how tight this upper bound is. The lowest value of ρ_k^* is $2.21 \cdot 10^{-3}$, bounding the number of random edges to 6605 edges which is actually 380 more edges than the graph has. Thus, for this graph our method cannot rule out the possibility that the graph is totally random.

Fig. 5.8(a) shows a special kind of text analysis network, a so–called *word adjacency network* obtained from Uri Alon's website and used in the article of Milo et al. [168]. Here, the whole text of the book 'The Origin of Species' by Charles Darwin was parsed and a list of all words was computed. The network represents these words, and two of these vertices are connected by an edge whenever the corresponding words occur next to each other at least once in the whole text. Also here, we use the unweighted version of the network. The relative tree distance distribution seems to be quite different from the others since it is best fitted by a linear function, at least in the interval between tree distances 10 and 40, but on the other hand, the network is really small and it is hard to tell which function is best. Also here, the smallest $f\rho_k$ value is bounded by $1.63 \cdot 10^{-3}$, bounding the expected number of random edges to 44352 which is 150 edges more than the graph has.

In summary, not surprisingly, none of the real–world networks turned out to be a random graph, although the protein–protein interaction network and the word adjacency network come quite close. However, our method allows for different degrees of random graph contributions to the whole architecture. If a fit can be made with an offset that has a low standard error, this offset may be used as an upper bound on p, the parameter of the contained random graph. The second method overestimates this upper bound by concentrating on only one realization factor. Of course, for every single tree distance class, ρ_k^* could be much higher than ρ_k and still show only $\#_r(k)$ realized edges. But if ρ_k^* were the real parameter of the random graph contained in this then all other tree distance classes should also show expectedly at least $\rho_k^* \#_p(k)$ realized edges, and if this value is higher than $\#_r(k)$ in all tree distance classes, than it is simply unlikely that ρ_k^* is the correct realization factor. Thus, the second method leaves room for improvement but still we get a valid upper bound and thus an impression of how much the graphs might be generated by a random process.

Furthermore, this first analysis shows that very different relative tree distance distributions were obtained by the method. So, what could create a steep relative tree distance distribution in a real–world network? There is one obvious and one intuitive answer: the obvious model is that the vertices are positioned in a hierachical system, as, e.g., the hierarchy in a big company, and an edge is more likely to exist the smaller the distance of the two vertices in the hierarchy is. Such a model has already been proposed and shown to occur in reality in HP labs [2]. In this paper, the relative tree distance distribution was measured as given by the organizational hierarchy, and was fitted by an exponential function e^{-ak} with parameter $a = 0.94$ and k the tree distance. Note that Kleinberg also proposed a model based on a hierarchy where the vertices are more likely connected if they are close within that hierarchy [127]. The difference from the first model is that in this hierarchy the vertices are only leaves and the inner vertices of the hierarchy do not represent vertices, whereas in the hierarchy in a company every position is occupied by one of the vertices.

However, grids also produce a relative tree distance distribution with at least a polynomial decay of the realization factor as a function of k as seen in Fig. 5.5(a) and we know that the vertices are not positioned in a hierarchy but in a 2–dimensional grid. Actually, our intuition is that a steep relative tree distance distribution is positively correlated with a local network–generating process, i.e., if vertices are assigned a position in space and edges with small distance are more likely than long ones, the resulting network will show a steep relative tree distance distribution. But in fact, such a straightforward correlation does not seem to exist. As with the clustering coefficient, and the (k, l)-locality introduced by Chung and Lu (s. 4.2.3), there are embeddings of vertices in a d-dimensional space and local network–generating processes like the unit-distance network–generating process that will lead to a relative tree distance distribution that is not steep, that is not even monotonically non-increasing. Thus, the only thing we know for sure when we find a steep relative tree distance distribution is that the graph cannot contain a large random graph component. However, such a steep relative tree distance distribution does say that there is a hierarchical embedding, i.e., the assignment of positions in a hierarchy, such that the edge set is weakly local. We will later show that indeed these different tree distance distributions can be used to compute certain problems efficiently, and thus we want to formally introduce the notion of a new locality definition.

5.3.1 Hierarchically Embeddable Local Graphs

As discussed in 4.2.4, there are two types of graphs. The first class, in which the vertices are assigned positions in a metric space that defines a (geometric) distance between any two vertices, will be called *embedded* graphs. The second class of graphs contains those where the only information about the structure of the graph is given by the adjacency matrix (*non-embedded graphs*). If there is no information about the position of vertices, i.e., only the adjacency matrix is given, there is so far no measure to determine whether the network–generating relationship is local in the sense of Definition 4.2, but at least one can try to find an assignment of positions such that the edges are local, i.e., find an embedding for the vertices such that the edges are local. If such an assignment can be found, a graph will be called *embeddable local.* We thus generalize the definition of locality of a graph as follows:

Definition 5.1 (Embeddable Local Graphs)
We define a graph to be *embeddable local* if there is an embedding $\mathcal{E} : V \to S$ of the vertices into a metric space[5] S such that the relation given by the set of edges E in the graph is *local* to some degree as defined in Definition 4.2.

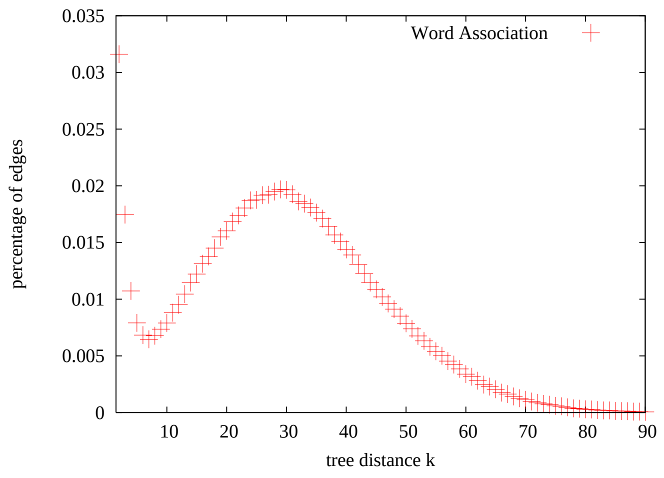
Note that this generalized definition includes those embedded graphs that are local as defined by their embedding. Note also that an embedded graph that is not local as defined by its embedding might be local with respect to another embedding.

The distances between vertices as defined on the basis of an (undirected) spanning subgraph fulfills the requirements of a metric distance measure: The distance between two vertices is 0 iff they are the same vertex, the distance is symmetric, and the triangle inequality holds. Thus, if we can find a spanning tree such that the relative tree distance distribution can be described by a non-increasing monotonic function, the graph can be said to be *embeddable weakly local.* Since the metric space in which the graph is embedded is somewhat peculiar and a spanning tree resembles most closely a hierarchy, we will call such an embeddable weakly local graph a *hierarchically embeddable weakly local graph.* This leads consequently to the question whether there are also *hierarchically embeddable* **strongly** *local graphs.* To answer this question we have computed the absolute tree distance distributions of the same real–world networks as above, shown in Figs. 5.9, 5.10 and 5.11.

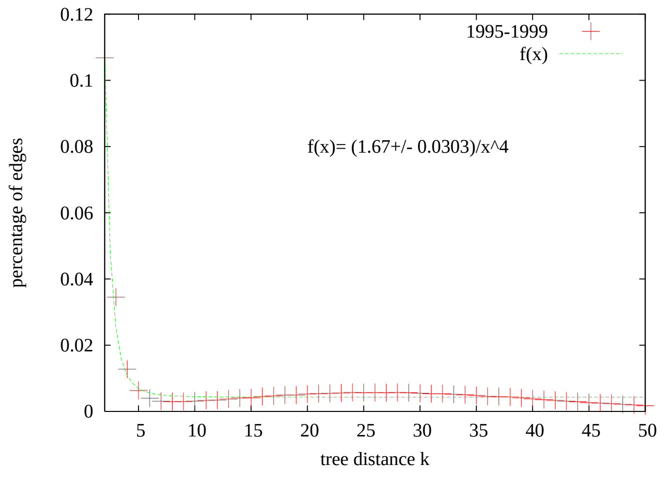
The distributions are given for the same randomized Kruskal spanning trees as above, and show very different distributions, some of which are fittable by simple functions. Note that we have normalized the values by the total number of edges in the graph, i.e., for the co–authorship network

[5] A *metric space* is a non-empty set S of points with a distance function $\delta : S \times S \to \mathbb{R}^+$ such that for all $x, y, z \in S$:

1. $\delta(x, y) = 0$ if and only if $x = y$.
2. $\delta(x, y) = \delta(y, x)$, i.e., δ is symmetric.
3. $\delta(x, y) \leq \delta(x, z) + \delta(z, y)$, i.e., the *triangle inequality* is valid.



(a) word–association graph



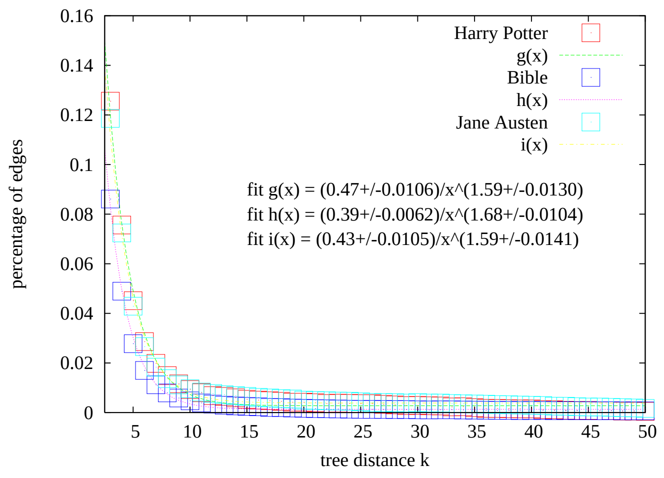
(b) co–authorship network

Fig. 5.9: Absolute tree distance distributions of various real–world networks (s. 8.1 for details on the data). The findings are discussed in the text.

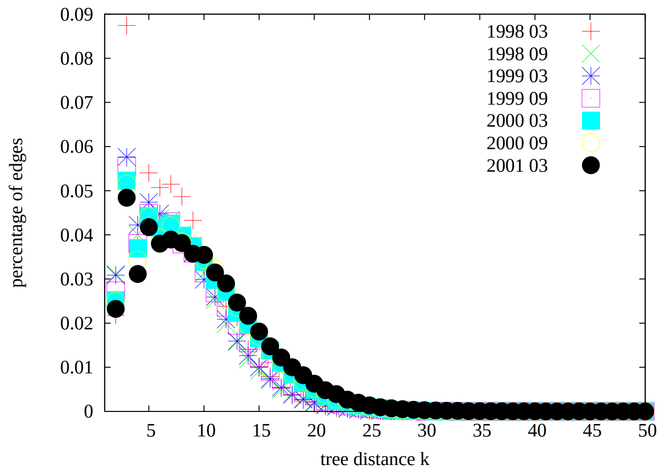
(Fig. 5.9(b)) and the co–purchasing networks in Fig. 5.10(a) more than 10% (9% in the case of the data set starting the Lutheran Bible) of all edges are in tree distance 2.

By this kind of spanning tree, at least the co–purchasing and the co-authorship data can be fitted quite well by a steep and monotonically falling function (although the co-authorship data is somewhat questionable). The protein-protein interaction data could also be called hierarchically embeddable strongly local, while the word–association, the word adjacency, and the autonomous system networks are not hierarchically embeddable strongly local, at least not with this kind of spanning tree.

Of course, the definition given above does only say that a graph is *hierarchically embeddable local* if there *exists* a spanning tree such that the relative and/or absolute tree distance distribution is monotonically non-increasing. We will thus discuss in the next section whether it is possible to



(a) co–purchasing network of books



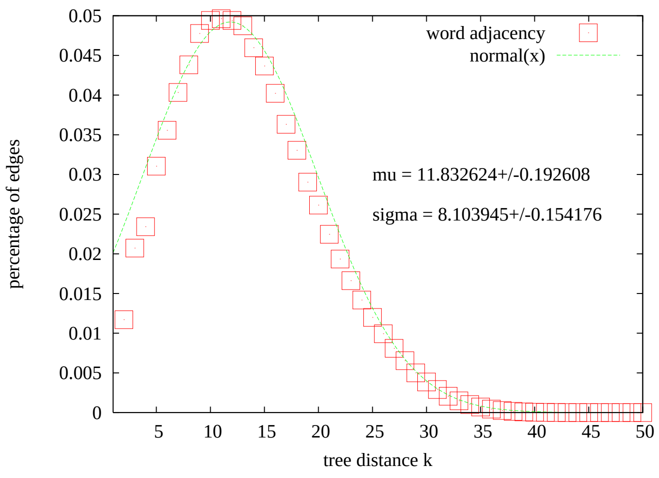
(b) autonomous system network

Fig. 5.10: Absolute tree distance distributions of various real–world networks (s. 8.1 for details on the data). The findings are discussed in the text.

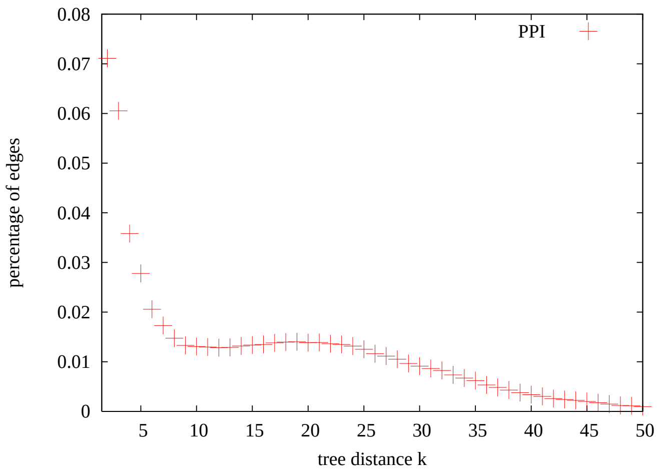
find spanning trees that optimize the absolute tree distance distribution.

5.4 A Quality Measure for Spanning Trees

To show whether a given graph is *hierarchically embeddable strongly local* it is necessary to find one spanning tree that induces a monotonically non-increasing absolute tree distance distribution. It is an unusual approach to optimize a distribution directly since it is hard to parameterize such a distribution, thus we concentrated on the question of whether it is possible to find a spanning tree that minimizes the *sum of the tree distances*. The motivation is that such a spanning tree would somehow capture the *essence of the graph's structure* in the sense that a group of vertices



(a) word adjacency



(b) protein–protein interaction graph

Fig. 5.11: Absolute tree distance distributions of various real–world networks (s. 8.1 for details on the data). The findings are discussed in the text.

that is densely connected in the graph is also near in the spanning tree[6]. A spanning tree with a low sum of the tree distances is thus called a *backbone of the graph*. We will now formally define this measure and give some simple lower and upper bounds in the next section.

[6] There is a huge bibliography on different types of spanning subgraphs. Often, a subgraph is searched such that average or the maximum ratio between the distance in the subgraph and the distance in the graph is lower than a given t. Such a subgraph is called a t spanner and can be used for overlay networks in communication networks [172, 196]. In this sense we search for a spanning tree in which the average distance of all those pairs of vertices that are connected by an edge in the full graph is minimized.

5.4.1 A Quality Measure for Spanning Trees and Some Simple Lower Bounds

With this intuition we define the *quality* $Q(T)$ of a given spanning tree as:

$$Q(T) = \sum_{e \in E} d_T(e) \tag{5.10}$$

A backbone is said to be an *optimal backbone* if it minimizes $Q(T)$. Technically, the problem of finding an optimal backbone is similar to that of finding the "Shortest Total Path Length Spanning Tree" ([205], p. 206, problem [ND3]) that is defined as follows:

Instance: Graph $G = (V, E)$, integer bound $B \in \mathbb{Z}^+$.
Question: Is there a spanning tree T for G such that the sum, over all pairs of $u, v, \in V$, of the length of the paths in T from u to v is no more than k?

The difference between the tree searched in this problem and the optimal backbone is that the latter restricts the sum to those pairs u, v that are connected by an edge in G, i.e., $(u, v) \in E$ instead of $(u, v) \in V \times V$. It is known that the shortest total path length spanning tree problem is NP-hard [205], but this does not necessarily mean that finding the optimal backbone also has to be NP-hard. However, in 5.4.2 we show that this restricted problem is also NP-hard to compute.

We will now give some simple upper and lower bounds on this measure. A trivial lower bound for $Q(T)$ is given by $2(m-n+1)$ since the spanning tree contains $n-1$ edges, and every non-tree edge spans at least tree distance 2. This lower bound is, for example, met by a clique if the spanning tree consists of one vertex and all incident edges. The following procedure computes a non-trivial lower bound that depends on the structure of the given graph: For every edge $e = (v, w)$ the distance $d_{E \setminus \{e\}}(v, w)$ is computed. Let $\Sigma(G)$ denote the sum of the $m - (n-1)$ lowest values of $d_{E \setminus \{e\}}(v, w)$.

Corollary 5.1
$\Sigma(G)$ is a lower bound for $Q(T)$ for any spanning tree T in G.

Proof 5.1
Let T^* denote an arbitrary spanning tree with minimal $Q(T^*)$. Let e be one of the $n-1$ edges in T^*, then its weight does not contribute to $Q(T^*)$. If $e = (v, w)$ is not in T, $d_T((v, w))$ cannot be smaller than $d_{E \setminus \{e\}}(v, w)$. Since we do not know which edges will be in T^*, we disregard the $n-1$ highest values of $d_{E \setminus \{e\}}(v, w)$ and thus, $\Sigma(G)$ is a lower bound for $Q(T^*)$. $\square$

However, for real–world networks both lower bounds often coincide because almost all edges take part in at least one triangle. A simple upper bound is given by $(m - n + 1) \cdot (n - 1)$ since every non-tree edge can at most have $n - 1$ edges in their tree path. In the following we will show that computing a spanning tree with minimal $Q(T)$ is NP-hard.

5.4.2 Finding the Optimal Backbone is Hard

A problem $\mathcal{P}$ is said to be NP-hard if all problems of the complexity class NP can be reduced to $\mathcal{P}$ polynomially, i.e., if every problem $\mathcal{P}'$ in NP can be transformed into an equivalent problem that - if it is solved by an algorithm for the problem $\mathcal{P}$ - also yields a solution for $\mathcal{P}'$. If such a transformation is found, problem $\mathcal{P}'$ is said to be reduced to $\mathcal{P}$. So far, no efficient algorithm has been found for any of the problems in NP. We will show here that finding the optimal backbone of a graph is hard because the well-known NP-hard 'Exact 3 Cover' problem ([205]) can be reduced to it.

Optimal backbone of a graph
Given: A graph G that is simple and undirected.
Solution: A spanning tree T^* with $Q(T) = \min_T \sum_{e \in E \setminus T} d_T(e)$.

where $d_t(e)$ denotes the distance of the end-vertices of e in T.

Theorem 5.1
Finding the optimal backbone is NP-complete.

We will first show that the *Exact* 3 *Cover Problem* can be solved if and only if a corresponding graph G has an optimal backbone T with a uniquely defined $Q(T)$. The *Exact* 3 *Cover Problem* is defined as follows:

Exact 3 Cover
Given: A set X of $3n$ elements $X = \{x_1, x_2, \ldots, x_{3n}\}$ and a set C of triples $c_i \subset X$.
Solution: A subset $C' \subseteq C$ with $|C'| = n$ with $\cup_{c \in C'} c = X$.

If there is at least one element x_k that is not an element of at least one set $c_i \in C$, there is no exact 3 cover and this can be determined in $O(|C|)$. It is thus assumed that every x_i is an element of at least one c_i. W.l.o.g., we will further assume that all triples $c_i \in C$ are unique.

Let now $G = (V, E)$ be built in the following way:

$$\begin{aligned}
V &= \{r\} \\
&\cup \left\{ rc_{ij},\ 1 \le i \le |C|,\ 1 \le j \le \frac{3n(3n-1)}{2} + 1 \right\} \\
&\cup \{c_i,\ 1 \le i \le |C|\} \\
&\cup \left\{ cx_k^{ij},\ 1 \le i \le |C|,\ x_j \in c_i,\ 1 \le k \le \frac{3n(3n-1)}{2} + 1 \right\} \\
&\cup \{x_i,\ 1 \le i \le 3n\}
\end{aligned}$$

$$\begin{aligned}
E &= E_{RC} = \{\{r, c_i\},\ 1 \le i \le |C|\} \\
&\cup\ E_{RRC} = \left\{ \{r, rc_{ij}\},\ 1 \le i \le n,\ 1 \le j \le \frac{3n(3n-1)}{2} + 1 \right\} \\
&\cup\ E_{RCC} = \left\{ \{rc_{ij}, c_i\}, 1 \le i \le n,\ 1 \le j \le \frac{3n(3n-1)}{2} + 1 \right\} \\
&\cup\ E_{CCX} = \left\{ \{c_i, cx_k^{ij}\},\ 1 \le i \le n,\ x_j \in c_i,\ 1 \le k \le \frac{3n(3n-1)}{2} + 1 \right\} \\
&\cup\ E_{CXX} = \left\{ \{cx_k^{ij}, x_j\}, 1 \le i \le n,\ x_j \in c_i,\ 1 \le k \le \frac{3n(3n-1)}{2} + 1 \right\} \\
&\cup\ E_{CX} = \{\{c_i, x_i\},\ x_i \in c_i\} \\
&\cup\ E_{XX} = \{\{x_i, x_j\}\ \exists c_i, x_i, x_j \in c_i\}
\end{aligned}$$

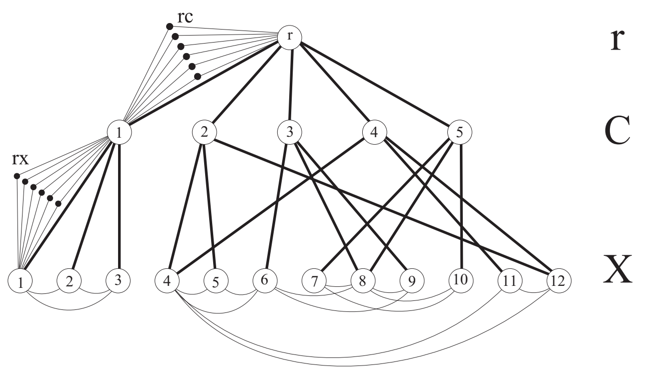


Fig. 5.12: A graph constructed from an **exact** 3 **cover** instance. The vertices in row X represent the elements x_i, and the vertices in row C represent the tripels. An additional root is connected to all subsets c_i, a tripel c_i is connected to all its elements x_i. All pairs x_i, x_j that are contained in at least one c_k, are connected. The vertices rx_{ij}, rc_{ij} (indicated by small, black vertices) are added to ensure that edges between r and c_i and between c_i and x_j are favoured over edges between any pair of x_k, x_l. Note that every edge between r and C and between C and X is shielded in the same way.

The graph thus has

$$|V| \quad = \quad 3n+1+|C|\left(4\cdot\left(\frac{3n(3n-1)}{2}+1\right)+1\right) \tag{5.11}$$

and

$$|E| \quad = \quad |C|\left(4+8\left(\frac{3n(3n-1)}{2}+1\right)+6\frac{3n(3n-1)}{2}+1\right)+|E_{XX}| \tag{5.12}$$

edges. Fig. 5.12 shows an exemplary construction.

Vertices c_i and x_i correspond to the elements of C and X with the same names, and the newly introduced vertex sets RC and RX constitute vertices that are somewhat ‘shielding’the edge between a certain r_i and c_i, and c_i and x_i, respectively, as will be seen in the following proof. Triangles $\{r, rc_{ij}, c_i\}$ and c_i, cx_k^{ij}, x_j are also called *shielding triangles*.

We will first show that every minimal backbone consists only of edges from $E\setminus\{XX\}$ and that no two subsequent shielding edges, neither from r to any c_i nor from any c_i to any x_j, can be in T.

Lemma 5.1
A minimal backbone never contains a pair of edges $e_1 = \{r, rc_{ij}\}$ and $e_2 = \{rc_{ij}, c_i\}$ nor a pair of edges $e'_1 = \{c_i, cx_k^{ij}\}$ and $e'_2 = \{cx_k^{ij}, x_i\}$.

Proof 5.2
Let T be a spanning tree that contains e_1, e_2 for any c_i, and T' the tree that results by replacing one of the edges by the edge $e = \{r, c_i\}$. Then there are four kinds of paths in the tree, containing either only e_1, only e_2, both, or none. Paths containing neither e_1 nor e_2 will not be changed in T'. Paths containing e_1 and e_2 will now be shorter by one edge. Note that paths containing only one of the edges e_1, e_2 are those that start (or end) in rc_{ij}. Note further that the number of paths ending in rc_{ij} containing e_1 is equal to the number of vertices in the subtree rooted at r,

and that the number of paths ending in rc_{ij} containing e_2 is equal to the number of vertices in the subtree rooted at c_i. All the paths containing only $e_i, i = \{1, 2\}$ will be increased by one edge if it is replaced by e but paths containing only $e_{(3-i)}$ are not affected. Thus, by replacing that edge e_i by e that has fewer paths ending in it (or at most the same number), the new tree T' will have a strictly better $Q(T')$ in contradiction to the assumption. An analogous proof holds for edges e'_i, e'_j.
□

It is clear that from every cycle in G at least one edge has to be removed in order to get a spanning tree. We will now show that no edge from E_{XX} can be contained in T.

Corollary 5.2
No minimal backbone contains an edge from E_{XX}.

Proof 5.3
Let $e = \{x_i, x_j\}$ be an edge from E_{XX} that is contained in T. There are two cases:

1. Both x_i, x_j have only one c_k to which they are both connected, i.e., from the triangle c_k, x_i, x_j exactly one edge is not in the tree, say the edge between c_k and x_i that now has a tree distance of 2. It follows that all shielding triangles c_k, cx_{ki}, x_i of the one non-tree edge will then have a tree distance of exactly 3. Let T' denote the tree resulting from exchanging the edge $\{x_i, x_j\}$ with $\{c_k, x_i\}$. Now, the replaced edge and all non-tree edges in the shielding triangles will have tree distance 2. Thus, the new tree would have a lower $Q(T')$ in contradiction to the assumption.

2. The combined neighbourhood of x_i and x_j contains more than one c_k, say c_k and c_l. Thus, there is a cycle in G, containing x_i, c_k, r, c_l, x_j. At least one of the edges in this cycle cannot be in T. Let $\{x_i, c_k\}$ be this edge. Then, from each shielding triangle c_k, rc_{ki}, x_i also one edge cannot be in T, say all edges $\{x_i, rc_{kp}\}, 1 \leq p \leq \frac{3n(3n-1)}{2} + 1$. In the best case where all other edges of the above given cycle are in T, all of these $\frac{3n(3n-1)}{2} + 1$ edges have a tree distance of 5 but they could have had a tree distance of 2 if $\{c_k, x_i\}$ had been in T. The same is true owing to reasons of symmetry if the edge $\{c_l, x_j\}$ is not contained in T. If one of the edges $\{r, c_k\}$ or $\{r, c_l\}$ is missing the tree distance of $\frac{3n(3n-1)}{2} + 1$ non-tree edges is increased from 2 to 5. Observe further that for the combined neighbourhood of x_i, x_j exactly one edge to any of the possible c_k is in T.

   Thus, every edge in E_{XX} that is contained in T will have to counterbalance this amount by decreasing the tree distance of other edges. Because of the above argument, the tree distance of non-tree edges from $E_R \cup E_{RRC} \cup E_{RCC} \cup E_{CCX} \cup E_{CXX} \cup E_{CX}$ will at most be increased, but not decreased. Thus, the only possibility to decrease the tree distance of non-tree edges is provided by edges from E_{XX}. As stated before, any vertex x_i can be connected with all other vertices $x_j, i \neq j$ and these can all be interconnected. If the spanning tree only consisted of edges from $E_R \cup E_{RC} \cup E_{RX}$, none of the non-tree edges from E_{XX} would have a tree distance of more than 5. If all edges from E_{XX} incident to one vertex x_i were in T, then all other edges between the neighbours x_k, x_l would have a tree distance of 2. Even if x_i was incident to all other vertices in X and all of them were pairwise connected, this would reduce the tree distance of no more than $\frac{3n(3n-1)}{2}$ edges from 5 to 2. Since there are $\frac{3n(3n-1)}{2} + 1$ shielding triangles around every edge in RC and CX, it is not possible that any edge from E_{XX} is contained in T.

□

Combining these properties, no edge of E_{XX} can be in T. Furthermore, for every shielding triangle it cannot be that the two shielding edges are part of the tree. Note that if one of the edges in RC is not in T, then at least one edge of E_{XX} has to be in T to make it a spanning tree. It follows directly that all edges of RC have to be in T and that all non-tree edges of the shielding triangles of these edges contribute a tree distance of 2 to $Q(T)$. To ensure that all vertices in X are connected to the spanning tree, every one of them chooses exactly one of its possible edges to a vertex in C, yielding $3n$ vertices from CX that are in T. The non-tree edges of their shielding triangles contribute tree distance 2 to $Q(T)$. The $3|C| - 3n$ vertices from CX that are non-tree edges have a tree distance of 3 and the non-tree edges of their shielding triangles have a tree distance of 4. Since this is fixed for every minimal backbone, the sum of the tree distances for all non-tree edges in $E \setminus |E_{XX}|$ is fixed. The only difference lies in the sum of the tree distances of edges in E_{XX}. These can either have tree distance 5 or 2. An edge in E_{XX} has a tree distance of 2 if and only if both endpoints are connected by a tree edge to the same vertex c_i. Since there are only $3n$ edges between C and X in the tree, there can be at most $3n$ edges that have a tree distance of 2 and this will only happen if there are exactly n vertices in C whose edges to all their elements are in T. If such a tree is found, the solution of the exact cover is given by those vertices in C whose edges to vertices in X are all in T. The result is summarized in the following lemma:

Theorem 5.2
A minimal backbone with $Q(T) = \left(\frac{3n(3n-1)}{2} + 1\right)(20|C| - 12n) + 4|E_{XX}| - 15n$ in G gives a solution to the **exact** 3 **cover** problem.

Conversely: If there is a solution C^* to the **Exact** 3 **Cover** then construct the spanning tree T^* in the following way: All edges in $E_R \cup E_{RC}$ to r are tree edges. All edges incident to a set c_i that is an element of C^* are tree edges. Since the union of all sets in C^* is S, the tree thus defined is a spanning tree. It is easy to check that this spanning tree has the minimal $Q(T)$ as given by Lemma 5.2. This concludes the proof that finding an optimal backbone is NP–hard.

Although finding the optimal spanning tree is NP-hard, there are some graph classes for which the minimal tree distance sum $Q(T)$ can be characterized tightly. In the following we will give such an analysis for a subset of random graphs from the $G(n, p)$ family.

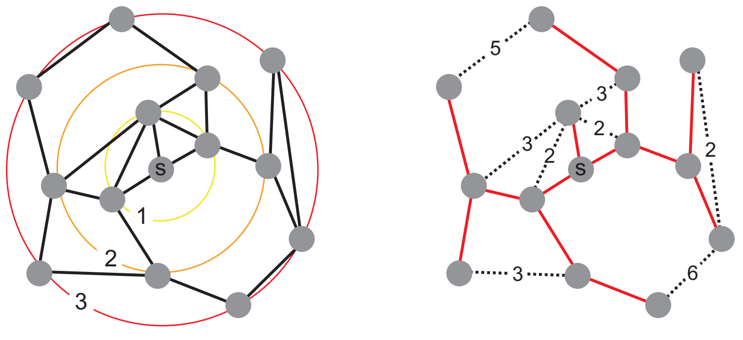
5.5 BFS Trees as an Approximation of Optimal Backbones

In this section we will analyze whether a certain simple class of spanning trees called randomized BFS trees can be used to calculate constant factor approximations of optimal backbones. A randomized BFS tree is constructed by starting a *breadth first search* ([57]) at some vertex r with which the distance of all other vertices to r can be computed. Subsequently, every vertex besides r chooses one of its neighbors with a lower distance to r as its father in the tree. This simple procedure constitutes a spanning tree and needs $O(m)$ time for the construction (Fig. 5.13).

We will first analyze a positive example where indeed a randomized BFS tree gives a constant factor approximation of the optimal backbone in 5.5.1, and then show in 5.5.2 that this is not the case for general graphs.

5.5.1 A Tight Bound for the Optimal $Q(T)$ in Random Graphs

In the following we will show that a simple backbone yields a constant factor approximation to the optimal backbone for a subset of the random graph family. To do so, we first need some further definitions.



(a)

Fig. 5.13: (a) Colored circles denote the vertices in the same distance to root r; (b) Every vertex at any distance i except s itself will mark exactly one incident edge to a neighbor in distance i-1. This results in a spanning tree of the network (red edges). Dashed edges are those that are in the full network but not in the tree (non-tree edges). The numbers assigned to the non-tree edges in the drawing denote the backbone distance of a non-tree edge (v,w), defined as the distance of v and w in the backbone.

The average tree distance is defined as $Q(T)/m$, and the diameter $D(G)$ of a graph G is defined as the maximal distance between any two vertices. To give a tight bound for the tree distance sum in random graphs, we need *Chernoff's inequality* (s. 3.7.2) and a second formulation of Bollobás' result on the diameter of random graphs:

Theorem 5.3 (Diameter of Random Graphs ([34], p. 257))
Let c be a positive constant, $k = k(n) \geq 2$ a natural number, and define $p = p(n, c, k),\ 0 < p < 1$, by

$$p^k n^{k-1} = \log(n^2/c). \tag{5.13}$$

Suppose that $np/log^3 n \rightarrow \infty$. Then, in $\mathcal{G}(n,p)$ we have
$\lim_{n\rightarrow\infty} P(D(G) = k) = e^{-c/2}$ and $\lim_{n\rightarrow\infty} P(D(G) = k+1) = 1 - e^{-c/2}$.

In other words, for these ranges of p and n, the diameter $D(G)$ of almost every graph is either k or $k+1$, and $k = \{1+o(1)\}\frac{\log n}{\log(np)}$. We will now show that for random graphs from this parameter space, $Q(T)$ is in $\Theta(m \log n)$.

It is clear that the expected degree $deg(v)$ of a vertex v is normally distributed around pn and by Chernoff $P[deg(v) > 2pn] < \left[\frac{e}{4}\right]^{np}$ is asymptotically zero. The proof for the average tree distance in an optimal backbone in random graphs makes use of the following lemma:

Lemma 5.2
Given two vertices v, w drawn uniformly at random, the probability that these vertices have a distance of $\geq \frac{1}{2}\left(\frac{\log n}{\log(2np)} - 1\right)$ is $\geq 1/2$.

Proof 5.4
Since we can assume that $deg(v) \leq 2np$ for all $v \in V$, at most $\sum_{i=1}^{j}(2np)^i \leq (2np)^{j+1}$ vertices lie in distance $\leq j$ from v. Let now j^* be chosen such that $(2np)^{j^*} = \frac{n}{2}$ then the first j^* steps can at most contain $n/2$ vertices, and then only in the unlikely case that every vertex has a degree of

$2np$. This means that for a fixed vertex v, half of the choices for vertex w have a distance $\geq j^*$ to v.

$$(j^*)\log(2np) = \log(n/2) \tag{5.14}$$

$$\Leftrightarrow j^* = \frac{\log n - 1}{\log(2np)} \tag{5.15}$$

It is clear that every vertex w has the same chance of being in the set of vertices with distance $> j^*$, i.e., the probability that any vertex w chosen uniformly at random has distance $d(v,w) > j^*$ is $\geq 1/2$ which concludes the proof. $\square$

Let now $e = (v,w)$ be any edge in E. By construction of the random graph, v and w are like any pair of vertices chosen at random from V. Thus, with probability $> 1/2$, the distance between these two vertices in $G\backslash\{e\}$, the graph without edge e, is $> j^*$. For any spanning tree, $m-(n-1)$ edges have to be removed from the graph. By Lemma 5.2 we know that two vertices drawn at random have a distance of more than $\left(\frac{\log n}{\log(2np)} - 1\right)$ with probability $\geq 1/2$ in the full graph, and of course their distance can only increase in the spanning tree. Since the two endpoints of each edge are drawn uniformly at random from the set of all edges, all $m-(n-1)$ non-tree edges have a probability of $\geq 1/2$ of having a distance larger than j^*, and thus the sum over all tree distances $Q(T)$ in a random graph is given by $\Omega((m-n)\frac{\log n}{\log(2np)}$. Since $p \geq \log^3 n$, $\lim_{n\to\infty} m/n \to \infty$ and thus the *average tree distance per edge* is in $\Omega\frac{\log n}{\log(2np)}$.

On the other hand, we know that the diameter of a random graph with the above given restrictions on np is asymptotically given by $\log n/\log(np)$, and thus, any backbone that maintains this diameter induces an upper bound on the tree distance in any tree of two times its diameter, making the bound tight. The result is summarized in the following Theorem:

Theorem 5.4 (Average Tree Distance in Random Graphs)
For $np/\log^3 n \to \infty$ and $p(n,c,k)$ as defined in Theorem 5.3, the average tree distance in an optimal backbone is $\Theta\left(\frac{\log n}{\log(np)}\right)$.

This implies that such a backbone gives a constant factor approximation of the optimal tree with respect to the average tree distance.

Lemma 5.3
Any backbone that maintains the diameter of a random graph $G(n,p)$ with $np/\log^3 n$ yields a constant factor approximation of the optimal average tree distance.

A simple backbone that maintains the diameter of the whole graph can be efficiently computed in $O(m)$ by a *breadth first search* as sketched above. It would of course be convenient if a BFS tree would yield a constant factor approximation of the optimal average tree distance for all graph classes since a BFS tree is so easy to compute efficiently, but we will now show that there are graph classes where this is not the case.

5.5.2 A Lower Bound for the BFS Spanning Tree in a Grid

Here we will show that, in general, BFS trees cannot give a constant factor approximation to the optimal backbone. To do so, we will first introduce the notion of the so–called *BFS–number*. Let

r be any vertex of a graph, called the *root*. A BFS tree is constructed by running the *breadth first search* algorithm [57]. This method assigns consecutive numbers, called the BFS–number, to all vertices in the order it approaches them for the first time, starting with 0 for r. The neighbors of the root are then numbered consecutively in any order, assigned to r as its children in the tree, marked as already visited, and put in a first-in-first-out data structure (FIFO list). From this list, the first element is taken, its neighbors are numbered consecutively (if they were not already visited), assigned to it as its children in the tree, marked as visited, and also put in the FIFO list.

Let $b(v)$ denote the BFS-number of vertex v and let $l(v)$ denote the level (distance to root) of v. It is clear that for a given graph with a given order of the neighbors of any vertex, and a root vertex r, the BFS–numbers constitute a deterministic and bijective mapping from V to the numbers from 1 to n. A *direct* descendant of v is a neighbor w of v with $l(v) = l(w) + 1$, and a *descendant* of v is every vertex w whose shortest path to r contains v. v is called the (direct) ancestor of w. This implies that there is a chain of vertices from v to w such that every two consecutive vertices in the chain are in a direct ancestor-descendant relationship. Note that the father-child relationship is a special ancestor-descendant relationship such that the father of a vertex is one of its direct ancestors, and note that every vertex except the root has exactly one father.

It is known from the literature that the optimal average tree distance in two-dimensional grids is bound from above by $O(\log n)$ ([194], p. 209). We will now show that the average tree distance in a BFS tree of a grid is bounded from below by $\Omega(\sqrt{n})$ and thus a BFS tree does not constitute a constant factor approximation for the optimal average tree distance.

Corollary 5.3
Let v, w be two vertices with $l(v) = l(w)$ and $b(v) < b(w)$. Let x be a descendant of v and y be a descendant of v with $l(x) = l(y)$, then $b(x) < b(y)$.

Proof 5.5
By induction. Since $b(v) < b(w)$, all yet unvisited neighbors of v are assigned their BFS–number first. Since every descendant of v is in a chain of direct ancestor-descendant relationships, this argument can be easily applied in the induction step, finalizing the proof. □

Let the grid be divided into four regions $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ with $\mathcal{A}$ containing all vertices with $x \geq 0$ and $y \geq 0$, $\mathcal{B}$ the vertices with $x \leq 0, y \geq 0$, $\mathcal{C}$ all vertices with $x \leq 0, y \leq 0$, and $\mathcal{D}$ all vertices with $x \geq 0, y \leq 0$. Note that vertices at position 0 in the x- or y-axis are element of two adjacent regions and thus every region contains $i + 1$ vertices in distance i to r. The division is constructed such that every region contains all ancestors of every vertex contained in it, i.e., the groups are closed under the ancestor-descendant relation. Thus, the subtrees of the BFS tree in any region are independent of each other, and in the following we will just observe the subtree that consists of vertices in region $\mathcal{A}$, i.e., those vertices with positive coordinates. All results can be transformed totally analogously to the other regions by a simple coordinate transformation. In the following a vertex v at position (x, y) will sometimes be identified with its position. $v.x$ denotes its position on the x-axis, $v.y$ that on the y–axis.

If the tree in $\mathcal{A}$ contains branches starting at some vertex $v = (x, y)$ such that all vertices (x, y') with $y' > y$ are descendants of this vertex in the tree, then this branch is called a *vertical chain*, and if there is a vertex at (x, y) and all vertices with (x', y) are its descendants in the tree, then this branch is called a *horizontal chain*. Two adjacent vertical or horizontal chains are said to build a so-called *vertical (horizontal) channel*. We will now introduce the notion of a *branching vertex*: a vertex in the subtree of the BFS tree in region $\mathcal{A}$ is called a *branching vertex* if it has two children. A vertex v in the same level as some other vertex w is said to be *above* (*below*) w if $v.y > w.y$ ($v.y \leq w.y$). Note that a vertex v above w is also left of it since the sum of their coordinates is the same, i.e., $v.x + v.y = w.x + w.y$, otherwise the vertices were not in the same level. We will now state a general lemma that explains the correlation between branching vertices and the emergence of channels.

Lemma 5.4

1. In the BFS tree in region $\mathcal{A}$ there is at most one branching vertex $bv(i)$ at position $(bv(i).x, bv(i).y)$ in any level i.
2. For all levels $i > 0$, a branching vertex is always the child of the branching vertex in level $i-1$. Thus, if there is no branching vertex in level i there will be no branching vertex in any of the levels $i' > i$.
3. The branching vertex always has the lowest BFS–number in that level (in that region).
4. The branching vertex starts one vertical chain and is contained in a horizontal chain, or vice versa. This implies that all vertices above the branching vertex are part of vertical chains, all vertices below the branching vertex are part of horizontal chains, and thus, all chains to the left of the branching point constitute vertical channels and all to the right constitute horizontal channels.

Proof 5.6

The proof is done by induction on the number of levels in the BFS. It is clear that at level 0 (the root) the tree in region A consists of the root itself (that is contained in all regions), and its two children with positive coordinates in level 1. Thus, there is only one vertex with two children, and this is the branching vertex. The branching vertex has the lowest BFS–number of that level since it is the only vertex. Now we have to show that it is the root of one vertical and one horizontal chain. As is easy to see, there is only one possible shortest path to vertices at position $(0, y), y > 0$ or $(x, 0), x > 0$, namely through vertex $(0, y-1)$ and $(x-1, 0)$, respectively. Thus, vertex $(0, 0)$ is the ancestor of all vertices $(0, y)$ with $y > 0$ and the ancestor of all vertices $(x, 0)$ with $x > 0$.

For the induction step we assume that in level i of the BFS tree in region $\mathcal{A}$ the lemma is correct. Then there is at most one branching vertex with two children. We will first discuss the case in which there is one branching vertex $bv(i)$. Since it had the lowest BFS–number of all vertices in level i (in region $\mathcal{A}$), its children will also have a lower BFS–number than all other vertices in level $i+1$ as stated in Corollary 5.3, and one of them will have a lower BFS–number than the other, say vertex z. Note that every vertex in a region knows exactly two neighbors in the same region in the next-higher level if it is not on the border of the grid. Since z has the lowest BFS–number its neighbors in the next level will be assigned to it as children, and thus z is the branching vertex of level $i+1$. We will now show that z is either part of one horizontal chain and starts a vertical chain or vice versa. Let z be the child at $(bv(i).x, bv(i).y + 1)$. Since bv is contained in a vertical and a horizontal chain (independent of which one started there, it is contained in both), and z is at $(bv(i).x, bv(i)y. + 1)$, it is contained in the same vertical chain. We thus have to show that it starts a horizontal chain.

Note first that every vertex v at position $(v.x, v.y)$ in region $\mathcal{A}$ is an ancestor of all vertices w with $w.x + w.y = v.x + v.y + (w.x - v.x) + (w.y - v.y)$ due to the definition of shortest paths and the ancestor-descendant relationship. Especially, z is the ancestor of all vertices $(x', z.y)$. All vertices at $(x', z.y)$ have at most two different direct ancestors in region $\mathcal{A}$, namely $(x'-1, z.y)$ and $(x', z.y-1)$. Note that these vertices are in the same level, and that $(x', z.y-1)$ is the descendant of the vertex $(z.x+1, z.y-1)$ that is in the same level as z, and that $(x'-1, z.y)$ is a descendant of z. Because of corollary 5.3 and the fact that the BFS–number of z is lower than every other BFS–number in the same level, the BFS–number of $(x'-1, z.y)$ is also smaller than that of $(x', z.y-1)$, and thus by induction vertex z is the father of $(z.x + 1, z.y)$, and in general $(x', z.y)$ is the father of $(x+1, z.y)$, concluding the proof that if the branching vertex $z = bv(i+1)$ of level $i+1$ is at $(bv(i).x, bv(i).y+1)$ then it is contained in one vertical chain and starts a horizontal chain. It is easy to show that if $bv(i+1)$ is at $(bv(i)x+1, bv(i).y)$, then the opposite will be the case. Note that this part of the lemma implies that every vertex above the branching vertex of a level is contained in a vertical chain and that every vertex below it is contained in a horizontal chain. It is now clear that if a branching vertex has two children where the one with the smaller BFS-number is at the (say, right) border of the grid then this vertex will only have one child in the next level and thus not be a branching vertex. It will nonetheless start a last vertical chain since it is the child of the branching vertex in level i at position $(bv(i).x+1, bv(i).y)$.

Since it is at the right border of the grid, and we know by the assumption that all vertices above the branching vertex in level i are contained in vertical chains and itself starts a last vertical chain, there are now $bv(i).x+1$ vertical chains in parallel. In every level $i' > i+1$ there cannot be more than $bv(i).x+1$ vertices, and thus no further branching vertex can ever emerge in subsequent levels which concludes the proof. □

We have now shown that every BFS tree consists only of vertical and horizontal chains and that these constitute vertical and horizontal channels. We have also shown that every branching vertex (besides the root) is the direct child of the branching vertex of the preceding level and that there is at most one branching vertex in every level. We can thus define the path from the root to the last branching vertex as the *trajectory* of the branching vertex. The trajectory of the branching vertex in one region defines where new channels emerge, and since the trajectory of the branching vertex is determined by the ordering of the neighbors of a vertex, the tree distance sum of a BFS tree in a grid is easy to calculate if this order is given.

Here, we only want to give a lower bound for the BFS tree with minimal $Q(T)$ for any possible neighboring order. For this, we need the notion of a *face* of the grid which is simply any region enclosed by the vertices $(x, y), (x+1, y), (x+1, y-1), (x, y-1)$ with $0 \geq x < \sqrt{n}$ and $0 < y \geq \sqrt{n}$. We will identify each face f by its upper left vertex (x, y). It is clear that at least one of the edges of such a face cannot be in the tree. Furthermore, since the tree allows only for horizontal or vertical channels, it is cheapest for any one face to open to the side of the grid that is closest. Beginning at the outermost faces of the grid, it is thus best to open them directly to the outer face. Faces that are adjacent to the outermost face should open to that face, and so forth. There are $4 \cdot (\sqrt{n}-1)$ outermost faces, $4 \cdot (\sqrt{n}-3)$ that are adjacent to an outermost face, and in general $4 \cdot (\sqrt{n}-2i-1)$ faces in shortest distance i to an outermost face [7]. One of the optimal BFS trees is depicted in Fig. 5.14.

For the computation of the tree distance sum it is easiest to reverse the tree distance assigned to the non-tree edges of the same channel, i.e., to exchange the tree distances of the outermost edge with the innermost edge, and so forth. The motivation behind this idea is that the innermost face in a channel determines the tree distance of the outermost face. Similarly, we exchange the tree distance of the second outermost with the one of the second innermost, and so forth. In this sense, an outermost face causes one non-tree edge with tree distance 3, and in general, a face in distance i from the outer face causes one non-tree edge with tree distance of $2i+1$. The tree distance sum $Q(T)$ for such an optimal BFS tree for odd $\sqrt{n}$ is then given by:

$$Q(T) = 4 \sum_{i=1}^{\sqrt{n}-1} (\sqrt{n} - 2i + 1)(2i + 1) \tag{5.16}$$

[7] For even $\sqrt{n}$. For odd $\sqrt{n}$, there is one additional face in distance $\sqrt{n}/2 + 0.5$

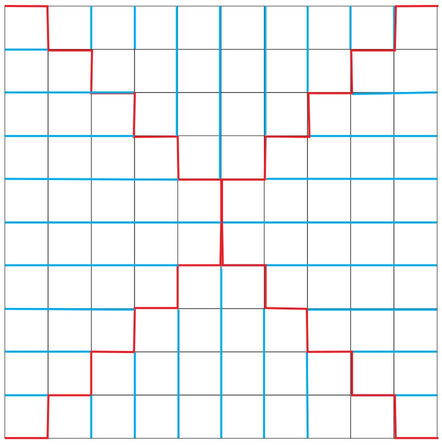


Fig. 5.14: One of the trees with the lowest $Q(T)$ a BFS tree in a grid can have.

$$= \quad 4\sum_{i=1}^{\sqrt{n}-1} 2i\sqrt{n} - 4i^2 + 4i + 1 \tag{5.17}$$

$$= \quad 8\sqrt{n}\sum_{i=1}^{\sqrt{n}-1} i - 4\sum_{i=1}^{\sqrt{n}-1} i^2 + 4\sum_{i=1}^{\sqrt{n}-1} i + \sum_{i=1}^{\sqrt{n}-1} 1 \tag{5.18}$$

$$= \quad 4n(\sqrt{n}-1) - 4\sum_{i=1}^{\sqrt{n}-1} i^2 + 2(n-\sqrt{n}) + \sqrt{n} - 1 \tag{5.19}$$

The lower bound for $Q(T)$ of the best BFS spanning tree is thus in $\Omega(n\sqrt{n})$ and the average tree distance in $\Omega(\sqrt{n})$. This implies that the best possible value of $Q(T)$ obtained by a BFS spanning tree is higher by a factor of $\sqrt{n}$, which concludes the proof that BFS trees do not yield a constant factor approximation in general. $\square$

At about the time we concluded the NP–hardness proof and conducted the first analyses, we learned that the question of finding a minimal backbone had already been posed by Alon, Karp, and Peleg in 1995, although under a totally different perspective and without an NP–hardness proof of the problem [9]. Then we found a connection of the same problem to very old problems in computer science, namely to the *minimum length (fundamental) cycle base problem* and the *minimum length (fundamental) cut base problem* [64, 103]. Since all of these problems have important implications for a series of very different applications, we will review the literature on these problems in the following section.

5.6 Problems Related to Finding the Optimal Backbone

The first to consider the question of finding an optimal backbone were Alon, Karp, Peleg, and West in 1995 [9]. They showed that the optimal backbone in a 2-dimensional grid with $n \times n$ vertices has an optimal average backbone distance of $\Theta(\log n)$, i.e., $Q(T) \in O(m \log n)$. In 2005, Elkin, Emek, Spielman and Teng showed that **every** graph has an optimal backbone such that the average backbone distance is in $O(\log^2 n \log\log n)$ [73]. Using their algorithm, Elkin et al. report that various theoretical problems like solving linear systems by an algorithm presented by Spielman and Teng [222] or finding an approximation to the 'Minimum Communication Cost Spanning Tree Problem' ([205], problem [ND7]) by an algorithm presented by Peleg and Reshef can be improved [195]. An interesting open question is whether the bound can be improved from $O(\log^2 n \log\log n)$ to $O(\log n)$, and whether the construction time of the algorithm can be reduced to $O(m \log n)$. Elkin et al. state that if so, the runtime of the Spielmann and Teng algorithm can be further reduced [73].

However, the problem of finding an optimal backbone is also very closely related to the problem of finding a minimum length fundamental cycle base as defined by a spanning tree, as we will show in the following.

Fundamental Cycle and Cut Bases and Their Relationship to the Optimal Backbone

The first ideas for analyzing graphs with respect to their cycle base was initiated by Kirchhoff to describe eletrical circuits ([123]). We will now give the necessary definitions. Let G be an unweighted, connected graph. A cycle C is a set of edges $\{e_1, e_2, \ldots, e_k\}$ such that the subgraph induced by C is a connected graph in which every vertex has even degree. C can also be described by an m-dimensional vector $\in \{0,1\}^m$ where the jth entry is 1 iff e_j is in C and 0 otherwise. The *cycle space* $\mathcal{C}(G)$ is the set of all cycles C. A *cycle base* $\mathcal{B}$ is a subset of independent cycles such that the vector of every cycle C that is not element of the basis can be constructed by an XOR operation on a subset of vectors of $\mathcal{B}$. The number of cycles in this subset is given by the *cyclomatic number* of a graph, i.e., $m-n+c$ where c denotes the number of connected components in the graph.

A cycle base is called a *fundamental cycle base* if there is an ordering of the cycles in the bases $\{C_1, C_2, ..., C_{m-n+c}$ such that for every cycle C_i the intersection of C_i with the union of all its predecessors $C_j, j < i$ is non-empty. Such a cycle base can be obtained by computing an arbitrary *spanning tree* T in every component of G, since every edge $e \notin T$ now induces a cycle on T, denoted by $C(e)$ [193]. It is easy to see that these are $m-n+c$ cycles, and since every cycle contains at least one edge that is not an element of any other edge, they are independent of each other, thus constituting a cycle base. Furthermore, every ordering of these cycles will satisfy the definition of fundamentality. Note however that not every cycle base and not even every fundamental cycle base can be defined by a spanning tree of the graph [104, 157].

The *length* $L(C(G)$ of a cycle base $C(G)$ of G is defined as the sum over the cardinalities of all cycles in the base, i.e.,

$$L(C(G)) = \sum_{C_i \in C(G)} |C_i(G)|. \tag{5.20}$$

For a fundamental cycle base note that for every $e \notin T$ the cycle it induces includes one more edge than its tree path, i.e., $|C(e)| = td(e) + 1$. The problem of finding a bound for the minimum length $L(C_T(G))$ of a fundamental cycle base induced by spanning tree T has already been stated in 1982 by Deo, Prabhu, and Krishnamoorthy [61], and they could show that it is NP-complete. Of course, a spanning tree that minimizes $L(C_T(G))$ also minimizes $Q(T)$ and vice versa[8].

Astonishingly, computing a minimum length cycle base that is not fundamental can be done efficiently; actually it can be computed by a simple greedy algorithm since it is a matroid. The first algorithm was given by Horton with a runtime of $O(m^\alpha n)$ for a weighted, undirected

[8] Note that the authors used a different reduction than the one we used and that the proof was achieved independently [61].

graph where α denotes the matrix multiplication constant [109]. This runtime was improved to $O(\max\{m^3, mn^2 \log n\})$ by Berger et al. by turning a fundamental cycle base defined by an arbitrary spanning tree into a minimal cycle base [30]. Horton could also show that for an unweighted graph the length of a minimal cycle base is bound by $O(n^2)$. Note however, that the upper bound of $O(m \log^2 n \log\log n)$ obtained by Elkin et al. for $Q(T)$ is also an upper bound on the length of a minimal cycle base and thus improves this bound [73]. Minimum length cycle bases are important for different applications, among others the analysis of electrical networks, for building efficient databases with which chemically similar molecules can be retrieved [111], and periodic event scheduling [159].

Similarly to the cycle space there is also a *cut space* of a graph. A *cut* is a set of edges such that if these edges are removed the graph is decomposed into different components. Again, a cut can be represented by an m-dimensional vector in $\{0,1\}^m$ where the jth field is 1 iff the edge e_j is in the cut. The cut space of G contains all cuts of the graph, and a cut base is again a minimal set of cuts such that all possible subsets of edges of G can be obtained by *xor* combinations of the cuts. Also here, a spanning tree T of G induces a *fundamental cut base* in the following way: Let e be an edge in T, then its removal in T leaves two connected components (in T) whose vertex sets are denoted by V_1 and V_2 [211]. As the induced *cut set* $CS(e)$ we will now assign to e the set of all edges $f = (v, w)$ with $v \in V_1$ and $w \in V_2$ besides e itself, i.e., the set of edges in the graph that have to be removed to disconnect V_1 from V_2. For simplicity, $cs(e)$ denotes the cardinality of e's cut set. We will now cite a proposition that states the fundamental duality between the cut set of a tree edge and the cycle set of a non-tree edge:

Proposition 5.1 ([103], p. 195)
Let T be a spanning tree of a connected, unweighted graph, and let C be a fundamental cycle with respect to a non-tree edge e^*. Then the edge set of cycle C exactly consists of edge e^* and those edges of T whose fundamental cut sets contain e^*.

With this simple proposition it is clear that if non-tree edge e has tree distance k in a given spanning tree T, it is contained in exactly k cut sets of the same spanning tree. Thus, the following equality is valid:

$$Q(T) = \sum_{e \notin T} td(e) = \sum_{f \in T} cs(f). \tag{5.21}$$

We have now shown that all of these problems that are deeply related to the optimal backbone have an interesting variety of applications, and we have shown above for some real–world networks, namely for the co–authorship network (Fig. 5.9(b)) and co–purchasing networks (Fig. 5.10(a)), that these have an absolute backbone distribution that can be fitted by a function in $O(1/k)$, implying that the sum of their backbone distances is in $O(m \log n)$, improving the bounds on minimum length (fundamental) cycle bases for these networks. We thus think that a characterization of different graph classes by the best backbone that can be found in a set of graphs from the same family, i.e., the same network–generating process, may be helpful in the development of efficient algorithms for the above mentioned applications.

Here we will introduce another application that depends on a good backbone to start with, namely the visualization of large real–world networks. Because a good backbone represents a given graph in the way that vertices that are near each other in the full graph are also near each other in the spanning tree, we have used backbones to visualize large and complex real–world networks with more than 1000 vertices, since the layout algorithms that are available in the yFiles graph layout package, a leading software on this area, networks with so many vertices can most often not be drawn in a helpful way but rather result in ball-like structures (Fig. 5.20(a)) [110]. For this it was necessary to find good backbones, a task for which we developed some heuristics that often result in backbones with a surprisingly low $Q(T)$ compared with the lower bounds as defined above (5.4.1). We will introduce these heuristics in the following section.

5.7 Local Optimization of the Backbone

Any given spanning tree T can be optimized by a simple procedure: the main idea is that any edge e that is not in T would induce a cycle if it were added to T. By removing any other edge l of this cycle, a new spanning tree $T'(e,l) := (T \cup e)\backslash l$ results. If no ambiguity is given we will reduce $T'(e,l)$ to T' in the following. If $Q(T')$ is smaller than $Q(T)$, then e should replace l in T. We will call e the *entering edge* and l the *leaving edge*. To analyze whether $Q(T')$ is smaller than $Q(T)$, the following definitions are helpful: Let e be any non-tree edge, then $P_T(e)$ denotes the path in T that connects the endvertices of e, the so-called *tree path* of e.

Proposition 5.2
For all non-tree edges i with $l \notin P_T(i)$, $d_T(i)$ will not be changed.

Proof 5.7
Since all edges of $P_T(i)$ are still in T, $d_T(i)$ cannot be increased. Assume that $d_T(i)$ is decreased by the insertion of e. This means that there is a second path connecting the end vertices of i, violating the tree property of T. □

Let i denote some non-tree edge whose tree path contains at least one of the edges of $P_T(e)$, and let $C_T(i,e)$ denote the set of shared edges:

$$C_T(i,e) := P_T(i) \cap P_T(e) \tag{5.22}$$

If the leaving edge l is in this set, the tree path of i will be altered. To describe the change, the following definitions are needed (Fig. 5.15 **a**): Let $C_T(e)$ denote all edges in the cycle that is introduced by adding e to T. Note that $C_T(e)$ is given by $P_T(e) \cup \{e\}$. Let $\overline{C_T(i,e)}$ denote the complement of $C_T(i,e)$ in cycle $C_T(e)$. The new tree path $P_{T'}(i)$ is then given by

$$P_{T'}(i) = P_T(i) \cup \overline{C_T(i,e)} \setminus C_T(i,e). \tag{5.23}$$

Note that this new tree path is always the same for any fixed non-tree edge i, independent of the identity of the leaving edge l as long as $l \in C_T(i,e)$ (s. Fig. 5.15 **b**). Thus, $\Delta d_T(i,e) := d_{T'}(i) - d_T(i)$ is given by:

$$\begin{aligned} \Delta d_T(i,e) &= |\overline{C_T(i,e)}| - |C_T(i,e)| && (5.24) \\ &= |C_T(e)| - 2|C_T(i,e)| && (5.25) \end{aligned}$$

With $I_e(l)$ denoting the set of non-tree edges i with $l \in C_T(i,e)$, we can now state the following lemma:

Lemma 5.5
For fixed entering edge e and leaving edge l, the difference in $Q(T)$ denoted by $\Delta Q(T,e,l)$ can be computed by:

$$\Delta Q(T,e,l) = \sum_{i \in I_e(f)} \Delta d_T(i,e,l) \tag{5.26}$$

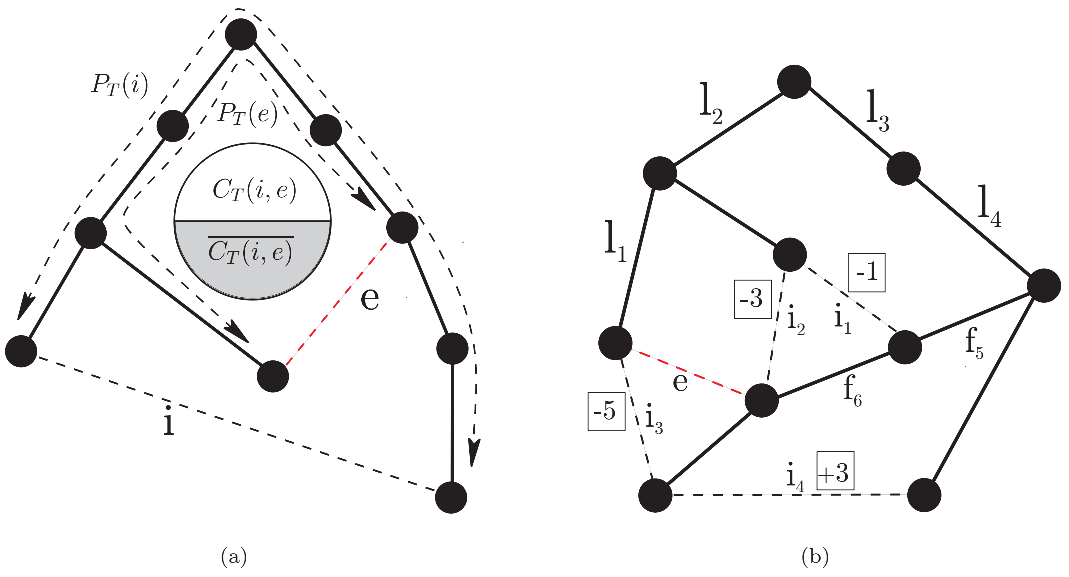


Fig. 5.15: (a) e is the entering edge, and the tree paths $P_T(e)$ and $P_T(i)$ of some other non-tree edge i are indicated by the dotted arrows. Every non-tree edge i with $C_T(i,e) \neq \emptyset$ will have to change its tree path if the leaving edge is element of $C_T(i,e)$. The new tree path is built by removing from the old tree path all edges from $C_T(i,e)$ and adding the complement of the circle, i.e., $\overline{C_T(i,e)}$, to it. (b) Again, e is the entering edge, and i_j are edges that could be affected by choosing some of the possible leaving edges l_i. The boxed numbers give the difference between the new and old tree distance. It follows that for entering edge e, l_2, l_3, or l_4 would yield the best optimization with a value of $\Delta Q(T,e,l)$ of -9.

$\Delta(Q(T,e,l))$ can be computed efficiently by first determining the set $I(e) = \cup_{l \in P_T(e)} I_e(l)$ of all edges i that are depending on at least one edge of $C_T(e)$ in their tree path. This can be done reasonably efficiently if every tree edge l stores $I_e(l)$ in a bit map. A bit map allows space– and time–efficient set operations, e.g., conjunctions and disjunctions. With at most n sets $I_e(l)$, the set $I(e)$ can be computed in $O(nm)$. The tree path $P_T(i)$ of every non-tree edge i is also stored as bits in a bit map. By simple $OR-, XOR-$, and AND-Operations all required sets $C_T(i,e)$, $\overline{C_T(i,e)}$, and $\Delta d_T(i,e)$ can be computed in $O(m)$ for a single non-tree edge i and in $O(m^2)$ for all of them. The leaving edge is the edge f with minimal $\Delta Q(T,e,l)$, which can be computed in $O(nm)$ where ties are broken at random. If there is no leaving edge because all resulting trees T' would be worse, nothing will happen and the next entering edge e is chosen at random. After e and l have been chosen in this way, some updates have to be made that are also computed very efficiently by operations on the bit maps. These updates can then be computed in $O(m^2)$.

Lemma 5.6
A single local optimization step can be computed in $O(m^2)$.

Note however that there is a huge trade-off between memory efficiency and runtime efficiency. The storage of all the bit maps requires $O(m^2)$ of space; if this is too much and the according bit maps have to be computed ad hoc, this will add another factor of $O(m)$ to the runtime.

A backbone is said to be *locally optimized* if no single optimization step with $\Delta Q(T) < 0$ can be found anymore. Note however that a locally optimized network is not necessarily globally optimal. In Fig. 5.16 a spanning tree is shown that cannot be improved by one single step in

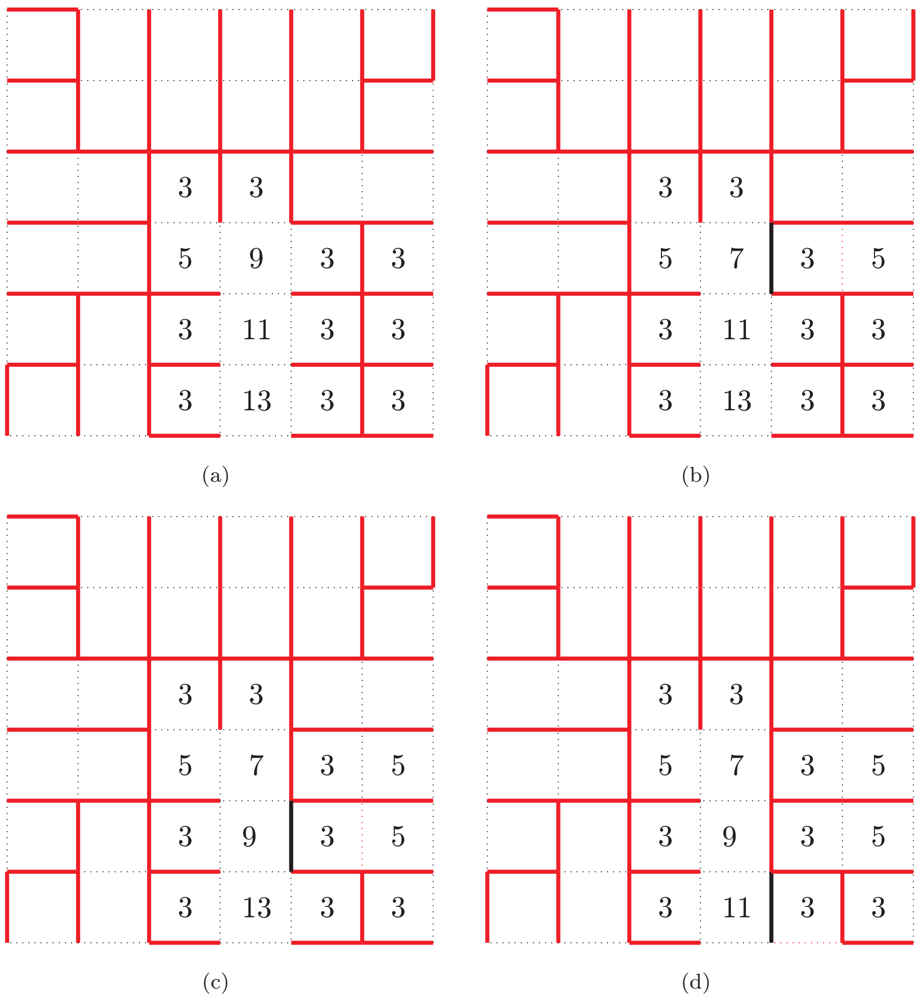


Fig. 5.16: (a) A spanning tree (solid red lines) in a 7×7 grid with $Q(T) = 198$. There is no single edge that could enter the spanning tree to improve $Q(T)$, but, as figures (b) and (c) show, there are spanning trees with the same $Q(T)$ that can subsequently build from the one in (a) such that from the one in (c) there is a single edge that can enter the tree to improve $Q(T)$ to 196. Entering edges are solid black and leaving edges are dotted red.

the local improvement algorithm although a better spanning tree exists. With respect to the local improvement algorithm described in 5.7, this spanning tree is thus a *local minimum*.

As we have seen above, a BFS tree is not always near to the optimal solution. In the following we will thus introduce some heuristics that generate a promising spanning tree, to which the local optimization algorithm can subsequently be applied.

5.7.1 Heuristics for Computing an Initial Backbone

The quality of spanning trees with respect to $Q(T)$ can be very different, a fact that is shown in Fig. 5.17. Since finding the spanning tree with minimal $Q(T)$ is NP-hard, as stated above, we will

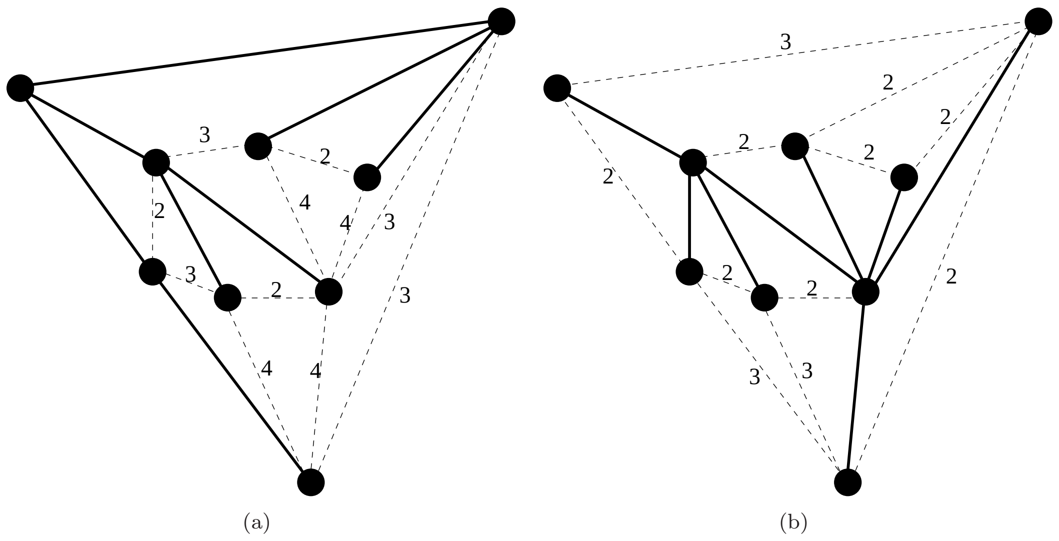


Fig. 5.17: The thick lines denote two different spanning trees T for the given graph. Numbers next to a (dotted) non-tree edge denote the tree distance of this edge. The spanning tree in a) has a quality $Q(T)$ of 34 and the spanning tree in b) has a $Q(T)$ of 25.

now show greedy algorithms that compute reasonable initial backbones that can subsequently be improved by a local optimization heuristic.

To construct a backbone, the most simple idea is to use a BFS tree. Although we could show above that this simple tree will not give a constant factor approximation of the optimal backbone, the quality $Q(T)$ of the resulting backbone is reasonably good compared to the above proposed quality measure Σ_G and the tree can be computed in $O(m)$. We will introduce two other methods that are computationally more involved, but yield much better backbones in practice. Both heuristics grow a spanning tree S incrementally by first choosing the next vertex v to append to S and then choosing the best edge to hook v into S. Both start with one vertex chosen at random. With S the set of vertices already in the tree, let R denote the set of vertices $v \in G \setminus S$ directly connected to at least one vertex in S. The vertex to append next is the vertex with maximal degree of R, where ties are broken in favor of the vertex with maximal number of neighbors in S; remaining ties are then broken at random. The intuition behind this heuristic is that vertices that are appended early to the growing backbone will influence the backbone's structure most. Since a vertex with a high degree will contribute a large sum of backbone distances to $Q(T)$, these vertices should have a large influence and thus be appended as early as possible. A trivial implementation searches for the vertex to append in $O(n)$ in every step, yielding a runtime of $O(n^2)$ for all steps. A more sophisticated data structure that keeps vertices in R sorted in a kind of two-dimensional array of lists, can reduce this runtime to $O(n\ deg^*)$, where deg^* is the maximal degree in the graph. For very large real-world networks this is in most cases a significant improvement.

In general, the chosen vertex v will have more than one neighbor in S and its tree edge will connect it to one of them. These neighbors are the possible *hooks* of v. Note that by choosing one of these edges to be v's tree edge, the tree distances of all the possible tree edges of v are determined. Thus, the first variant, the *minimized inner distance tree*, will choose that hook that minimizes the tree distances of all the other possible tree edges:

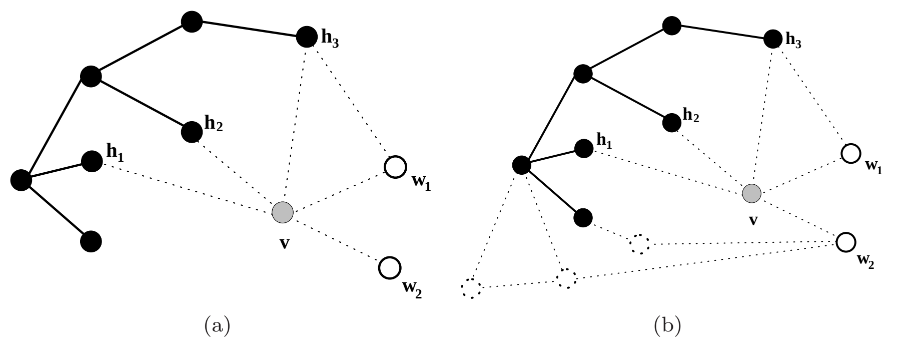


Fig. 5.18: **a)** Minimized Inner Distance Tree: Entering vertex v has three hooks h_1, h_2, h_3. h_2 minimizes the sum of the tree distances of v's edges to h_1, h_3 with a sum of 8, and thus h_2 is h^*. **b)** Minimized Entire Distance Tree: The tree distance of v's edges to w_1, w_2 can be estimated by determining the distance of the hooks to these neighbors. It follows that h_3 has the best sum of distances to all others: $|P(h_3,h_1)| = 4$, $P(h_3,h_2)| = 3$, $|P(h_3,w_1)| = 5$, $|P(h_3,w_2)| = 1$.

Minimized Inner Distance Tree Let $S(v)$ denote the neighbors of the chosen vertex v in S, i.e., the hooks of v. Since only one of the edges incident to a hook can be a tree edge without inducing a cycle in T, it is necessary to choose the one hook h^* that minimizes the tree distances of all the other edges to hooks. Thus, for every hook the distance to all other hooks is summed and the edge to the hook with the minimal distance to all other hooks is chosen as new tree edge (Fig. 5.18(a)).

By holding an array $D(T)$ of size n^2 that keeps the distance $d_T(s,t)$ for all vertices s,t in S, this computation can be done in $O((deg^*)^2)$. After the best hook h^* has been chosen, this data structure has to be updated by adding the distances $d_T(v,w)$ between the newly added vertex v and all other vertices w in S to $D(T)$. Since $d_T(v,w) = d_T(h^*,w) + 1$ for all $w \in S$, this can be done in $O(n)$. Thus, the entire runtime to construct a minimized inner distance tree is given by $O(n(deg^*)^2 + n^2)$.

Lemma 5.7
A minimized inner distance tree for some randomly chosen root vertex can be computed in $O(n(deg^*)^2 + n^2)$.

While this tree only regards those (inner) edges to other vertices in S, the next one tries to estimate the tree distance of the other edges of v as well:

Minimized Entire Distance Tree Let again $S(v)$ denote the neighbors of the chosen vertex v in S, and $N(v)$ denote the full neighborhood of v in G. For those edges of v that do not lead directly to vertices in S, it is hard to estimate their tree distance: It could be that they will later choose v as their hook to the growing tree and in this case an edge will not contribute to $Q(T)$. Since it is unlikely that all of them will use the edge to v as their tree edge, it would be good to choose a hook h^* such that all neighbors w of v have a short alternative path $P'(h^*,w)$ to v: A path $P'(h,w)$ is considered as an alternative if it can be split into two paths, the first using only vertices of S, the second—if necessary—only vertices of $V\backslash S$. In this way, the currently known structure of the tree is used as much as possible and the edges that are not yet known to be in the tree are only used for the last bit to reach w (Fig. 5.18(b)). With this intuition, we will choose the hook $h^* \in S(v)$

that minimizes the following sum:

$$\sum_{w \in N(v)} |P'(h,w)| \tag{5.27}$$

Note that the sum in Equ. 5.27 contains also the sum of the inner distances and thus the name of the tree is justified.

This computation can be done by computing the distance of all vertices to every hook of v which can be accomplished in $O(m\ deg^*)$. It follows that a minimized entire distance tree can be computed in $O(nm\ deg^*)$.

Lemma 5.8
A minimized entire distance tree for some randomly chosen root can be computed in $O(n\ deg^* m)$.

Graph	**BFS**	**Minimized Inner Distance**	**Minimized Entire Distance**	**Optimized**	$\Sigma(G)$
Co–purchasing network $n = 3437$ $m = 9671$	31342 ± 316 73 ± 11 [ms]	21819 ± 57 358 ± 34 [ms]	20654 ± 24 70 ± 0.3[s]	17596 ± 28 20min14s	12468
Live Journal $n = 3763$ $m = 11149$	29615 ± 1332 65 ± 11[ms]	23156 ± 71 284 ± 19[ms]	22058 ± 38 100 ± 0.4[s]	19588 ± 3 22min12s	14774
Co-Authorship Network $n = 12357$ $m = 19448$	52896 ± 1447 131 ± 18 [ms]	52463 ± 222 480 ± 34[ms]	49951 ± 98 337 ± 0.6[s]	34287 ± 47 49min2s	14184

Tab. 5.1: For each network, 10 instances of every kind of spanning tree were computed. Displayed is the average $Q(T)$, its deviation, and the average time and its deviation to compute the tree. Note that the best unoptimized spanning trees already have a quality that is close to the lower bound given by $\Sigma(G)$ that can nonetheless be further reduced by the optimization. Furthermore, every one of those 10 instances started at another, randomly chosen start vertex. The low deviation in $Q(T)$ shows that the method gives a stable $Q(T)$, nearly independent of the choice of the start vertex. The experiments were conducted on a Pentium 4 with 3.2 GHz and 2GB RAM.

Table 5.1 shows a comparison of $Q(T)$ of all three trees for some real-world networks and that of a spanning tree that results from the local optimization heuristic (s. 5.8.2 for a description of the networks). It is clearly visible that the higher computational effort for minimized inner distance and minimized entire distance trees results in much better backbones than the simple BFS tree and come near to the lower bound given by $\Sigma(G)$ (s. 5.4.1). However, the local optimization heuristic can decrease $Q(T)$ reasonably, even for the best starting tree. In the following we will show how such a locally optimized tree can be used to visualize large networks.

5.8 Visualization of Large and Complex Networks with Heuristically Optimized Backbones

At first glance it seems prohibitive to visualize large and complex networks. The idea to represent these networks by suitable spanning trees and draw these trees instead of the whole graph, is a well-known approach, found, e.g., in [50, 82, 106]. In most of these cases it was assumed that the spanning tree was either given by the user or that the graph to draw was hierarchically organized and thus a spanning tree could be easily and more or less unambiguously derived. Here we show that an optimized backbone will also result in helpful visualizations of large networks depending, of course, on their degree of locality and clusteredness. The idea is that a good backbone decomposes the edges of a graph into a set of *local* edges that are likely to be within clusters and a set of *global* edges that are likely to be between clusters. The decomposition into these sets has already been proven useful for drawing power-law graphs where the decomposition is derived by solving a network flow problem [12]. A simple idea for drawing a large graph is to compute a backbone and draw it with a tree layout algorithm, ignoring all non-tree edges. In most cases, this does not result in satisfying drawings. In our approach, non-tree edges will influence the order in which the children of a tree vertex are sorted, depending on the length of these edges in the backbone. This approach results in aesthetic drawings that reveal the large scale structure of the graph.

5.8.1 Using the Backbone for Computing a Layout

As indicated above, a good backbone will try to concentrate the vertices of any cluster on a small, connected subtree. By doing so, the tree also indicates that edges with a high tree distance are more likely to be inter-cluster edges. These properties of the backbone can be used for computing a layout that co-locates the vertices that are supposedly in a dense part of the graph, and simultaneously highlights the inter-connections between these dense parts.

To harness the backbone, our layout approach is based on a tree layout that is adapted towards the needs of a full graph. The layout of the graph can be computed by a variation of the balloon tree layout [50], resulting in a drawing which we will call a *backbone balloon drawing*. In the original balloon drawing of a tree, every subtree is enclosed entirely in a circle that is positioned in a wedge whose end-point is the parent vertex of this subtree. The radius of each circle is proportional to the number of vertices in the subtree.

To adapt this tree layout towards the needs of a full graph, the basic idea is to use the backbone and compute a balloon drawing for it and re-insert all non-tree edges as straight lines. To make this drawing a good drawing for the whole graph, the only parameter left to change is the order of the children of any vertex in the tree. Since all direct neighbors of any vertex in the tree are positioned in a circle, the order of these children can be determined by a variation of the algorithm for crossing reduction in circular layouts [27]. The original algorithm is composed of two phases: In the first phase an initial ordering is heuristically determined. This is optimized by subsequent rounds of local sifting, where each vertex can try to improve the number of crossings by changing its position in the order computed so far. The application of this algorithm in a backbone balloon drawing requires the following two modifications:

1. Every edge between the children of a vertex in the tree can not only cross with each other, but also with the spokes, i.e., the edges from the father to its children. This changes the computation of the resulting number of crossings slightly.
2. Let $T(v)$ and $T(w)$ denote the subtrees rooted at v and w, respectively, and let v and w be children of the same vertex. If the number of edges between these subtrees is large, then v and w should be close in the resulting order, which is not regarded in the original algorithm.

The second point can be dealt with by introducing additional edges between any two children v, w whose subtrees are connected by edges. Additionally, all edges will be assigned weights that present the number of edges between $T(v)$ and $T(w)$. The weight of a crossing between two edges is now given by the sum of the weights of the crossing edges, and the optimization goal is to minimize the sum of the weights of all crossings and not to minimize the number of crossings. The weights of all the edges between any two children can be computed in $O(nm)$. Every round of local sifting in a given circle with at most deg^* vertices can be computed in $O((deg^*)^2)$ as shown in [27]. Since there are at most n circles in the drawing, this sums to $O(n(deg^*)^2)$, which is the largest factor in computing the backbone balloon drawing.

5.8.2 Experiments

We have applied the above presented variant of the balloon layout algorithm to different types of networks, shown in detail in [148]. Here, we show exemplarily one co–purchasing network, starting at one of the 'Harry Potter' books ([181]; Figs. 5.20, 5.21, 5.22). The balloon tree drawing shows discernible clusters connected by long-range edges, that are even more pronounced in the drawing that is based on an optimized backbone with minimized entire distance. This visual impression is supported by the fact that the force-directed drawing has the highest (normalized) total edge length of $434,813$, the one based on the unoptimized backbone has a total edge length of $321,292$, and the one based on the optimized backbone has a total edge length of $220,857$[9].

To show the quality of the different backbone heuristics and the optional optimization step, we have conducted experiments on this Amazon recommendation network and two other networks, shown in Table 5.1. For the creation of the Live Journal network a crawl was started at some participant of www.livejournal.com, following the links to designated friends unto depth 3. The co-authorship network is described in [188]. Fig. 5.19 gives a showcase for the improvements of $Q(T)$ by the optimization heuristic. It is clearly visible that the time spent in this step is worth the effort.

5.9 Summary

In this chapter we have introduced the notion of *network–generating systems* and argued why we think that it is important to know the real process that built a network. Of course, often we do not have any data on the evolution of a network, especially in biological networks, nor any data besides the information of who is connected to whom, i.e., nothing else but the adjacency matrix. Thus, we considered the question of whether we can detect some processes in the resulting adjacency matrix. As we had already argued in chapter 4, locality seems to be hard (and we conjecture, impossible) to detect by looking only at the adjacency matrix. But in this chapter we have shown that it is at least possible to bound the number of edges created by a random graph process in any given network by a new technique. This technique introduced the notion of tree distance distributions and led to the observation that real–world networks often have an astonishingly steep tree distance distribution. It turned out that a steep tree distance distribution—and the low sum of tree distances resulting from it—is important for efficient algorithms in different applications. We thus think that this newly found characterization is one of the first real–world network structures that is interesting for theoretical computer science. So far, the process that leads to a steep tree distance distribution is unclear: it is intuitive that it arises in all networks whose network–generating process prefers local edges, but—as with the clustering coefficient (4.2.4) and the (k, l)-locality definition of Chung an Lu (4.2.3)—there is no strict correlation between the

[9] Note that a printer's resolution is not as good as that of a screen. The drawings look a bit scrambled on paper; the screen version is much more readable.

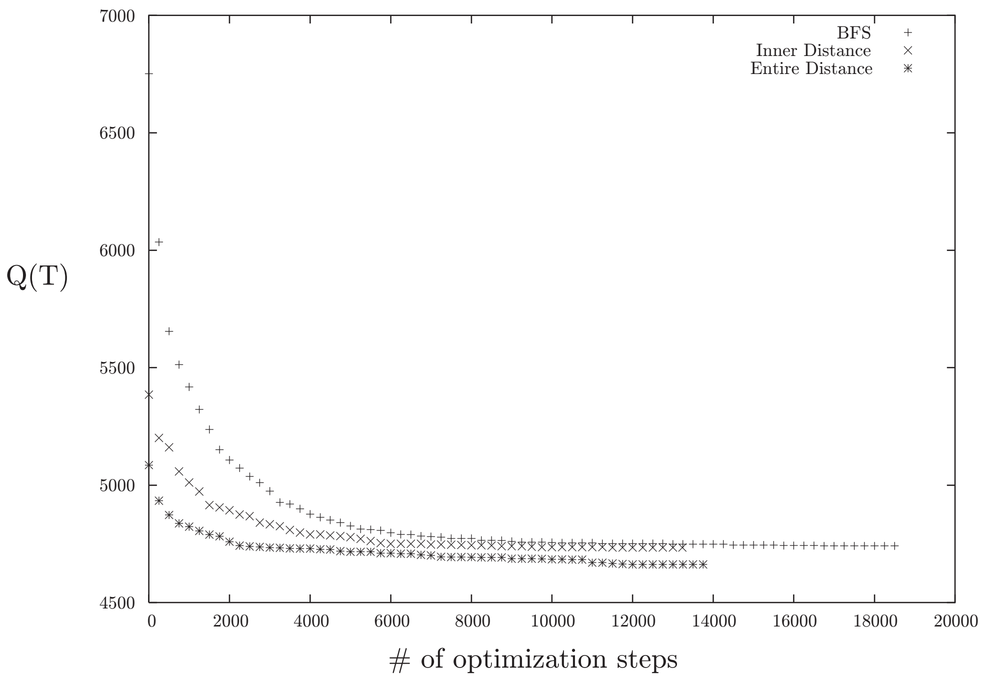


Fig. 5.19: For a smaller Amazon recommendation network with $n = 852$ and $m = 4220$, one BFS, one minimized inner distance and one minimized entire distance tree were computed and improved by the optimization heuristic until no further improvements could be found, i.e., a local minimum is reached. The trees start and end with the following $Q(T)$'s: $6572/4641$ (BFS), $5385/4734$ (Inner), and $5085/4662$ (Entire). Note that $\Sigma(G)$ is 3822.

two concepts. Another process that could lead to a steep tree distance distribution takes place in systems where the vertices are positioned in a hierarchy, and edges are less likely the higher the (hierarchical) distance between them, but so far we have not analyzed whether and how the tree distance distribution and this model correlate. In summary, we think that the tree distance distribution in real–world networks is a helpful characterization of real–world networks that could lead to interesting and efficient algorithms in many applications, and that gives rise to many other questions, e.g., its correlation to the modularity of a graph [93], or the minimal length of an MCB in graphs with a tree distance distribution described by $f(k) \propto k^{-j}$ for some constant j.

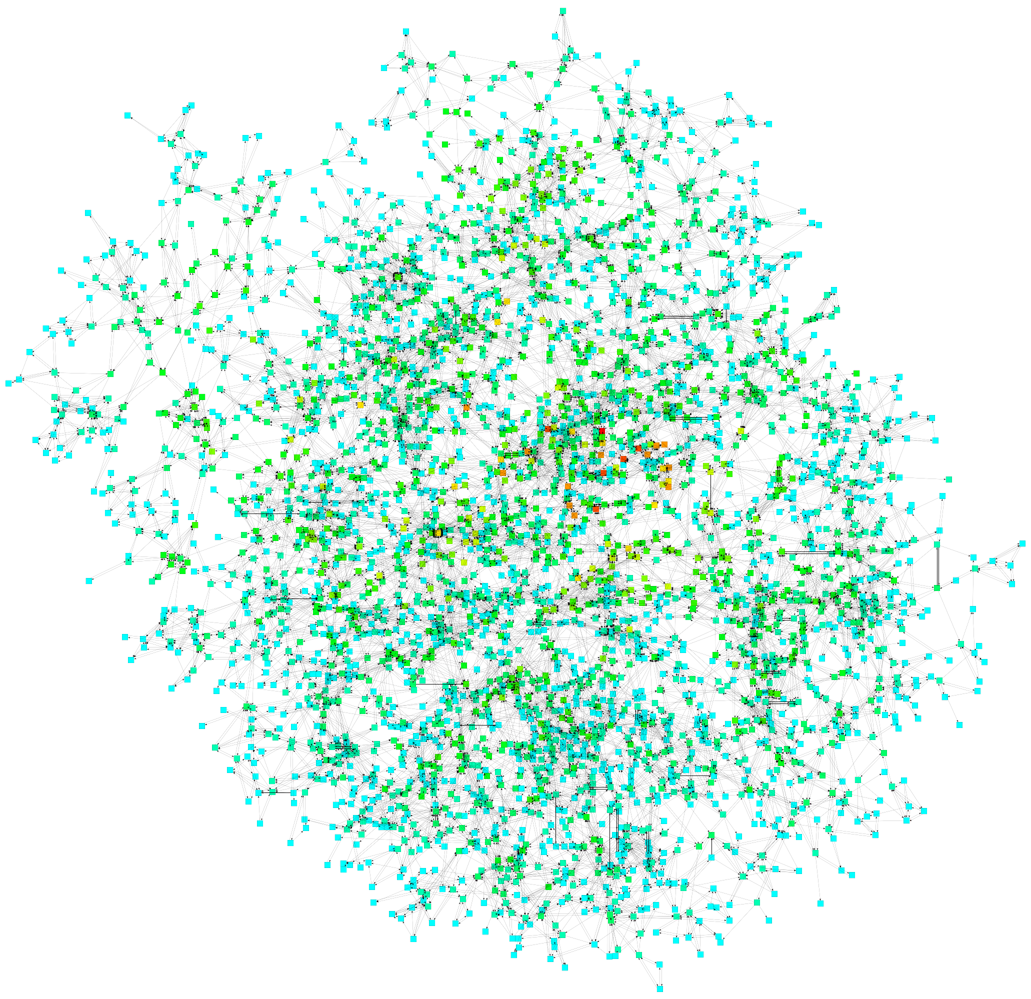


Fig. 5.20: A layout based on a force-directed approach, as implemented by the graph drawing library $yfiles$ [110]. The normalized total edge length of this drawing is $434, 813$. Red edges indicate the spanning tree that was used to compute the embedding.

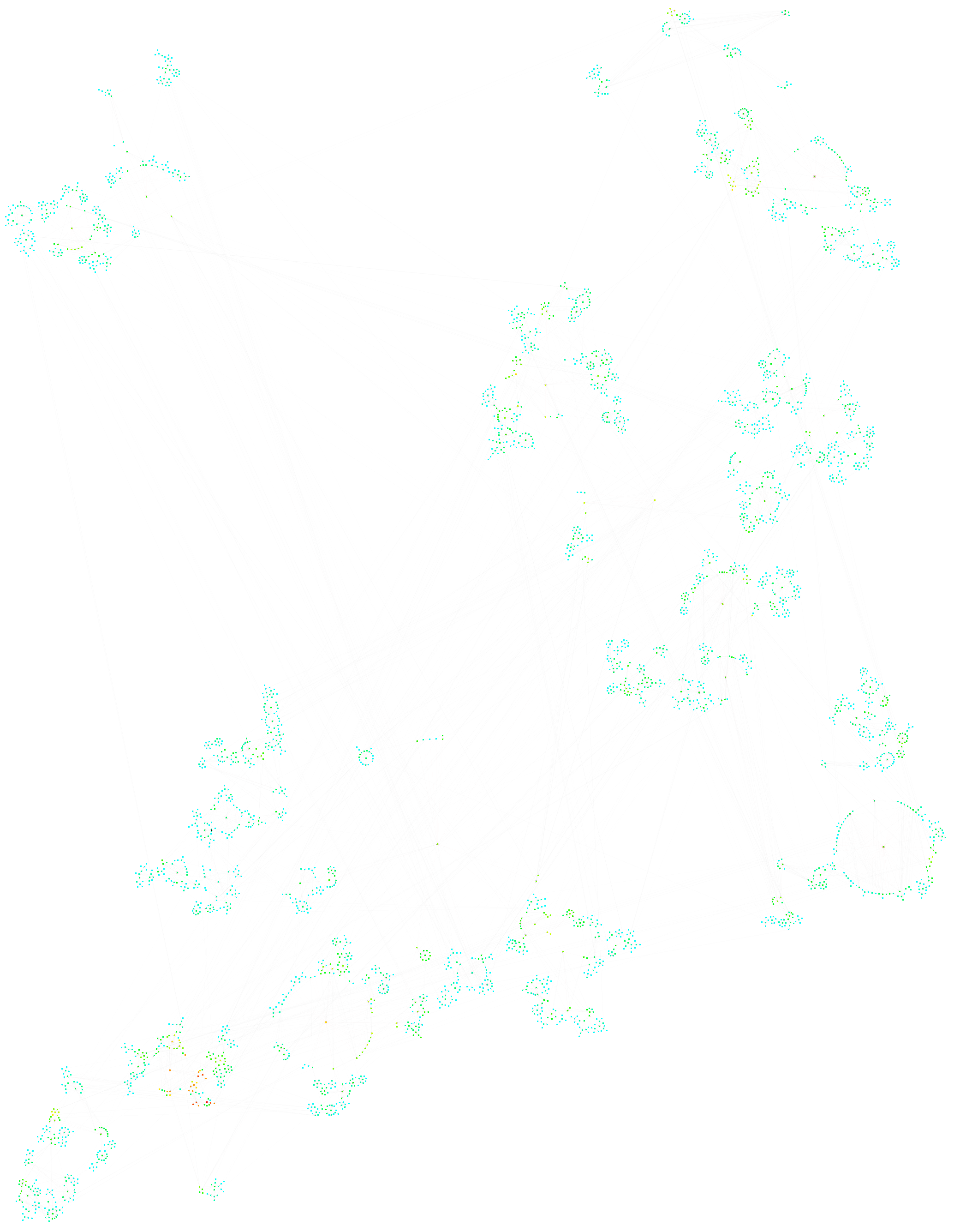


Fig. 5.21: A balloon layout drawing based on a simple, unoptimized backbone with minimized entire distance. The normalized total edge length of this drawing is 321292.

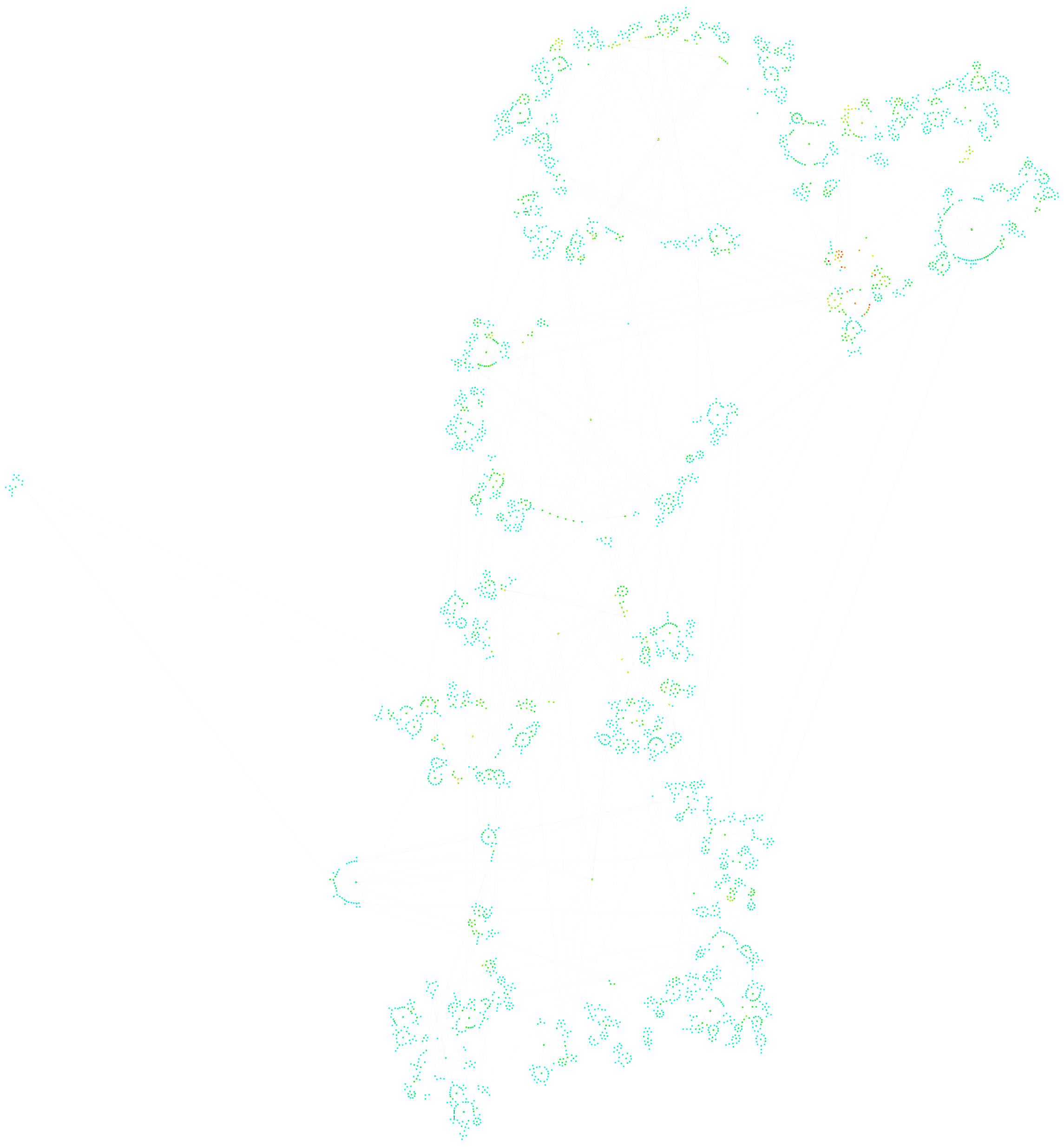


Fig. 5.22: A balloon layout drawing based on an optimized backbone with minimized entire distance. The normalized total edge length of this drawing is 220857. The time to compute this layout was on average around 20 minutes (averaged over 5 drawings).

6. THE PRINCIPLE OF LOCALITY IN THE EVOLUTION OF COMPLEX NETWORKS

As indicated in chapter 4, one interesting aspect of the small–world phenomenon is that the resulting network structure is small and at the same time cost–effective —under the assumption that the cost of building and maintaining an edge is at least weakly correlated with the distance it spans—while no central authority is required to organize it. On the other hand, the network will only be globally small if every vertex behaves in the same way—it does not help if only some of the vertices have one or two random edges. We can additionally safely assume that no single vertex in a social network would alone be willing to pay for a lot of long edges, even though this behavior would also result in a small network. Rather, it is intuitive that in a social network any network structure must be not only globally but also locally in a good cost–value ratio. This implies that every vertex must directly benefit from any edge it builds, and furthermore that it might remove edges that have no direct benefit for it. We conclude that social networks are built in a self-organized fashion by selfish and also myopic agents that will favor a short–term benefit over a long–term investment. Despite those rather severe restrictions on network generation, there seem to be some structural patterns that are stable in such a system, such as the already mentioned ubiquitous small–world effect and the scale–free degree distribution seen in almost any network [80, 8, 72, 46, 116, 176]. These examples show that it is nonetheless possible that a system of selfish and myopic agents builds and stabilizes a network structure that is determined by the behavior of all agents together.

The question is thus: If locality is the main building principle in a dynamic network, what kind of global network structures can be built by the collective behavior of selfish and myopic agents that will only accept local improvements? Vicsek describes *collective behavior* in the following way:

> [Synchronization phenomena] happen in systems consisting of many similar entities interacting in a relatively well-defined manner. These interactions can be simple ... or more complex ... and can occur between neighbours in space or in an underlying network. Under some conditions, transitions can occur during which the objects adopt a pattern of behaviour almost completely determined by the collective effects of the other objects in the system. [234]

We are interested in this latter phenomenon: a pattern of behavior of otherwise independent entities that leads to a globally synchronized—in the sense of globally functional—structure determined only by the interactions and feedback between the entities. The goal here is to find out how the principles that govern the evolution of real-world networks can be transformed and adapted to the needs of technical communication networks, i.e., our focus is on finding interaction and feedback mechanisms that control a certain global network structure such that every single entity in the system has an incentive to use this mechanism.

In this chapter we will provide some starting points to answer the question of how to build local rules to guarantee a desired global network structure. In 6.1, we will start with a general overview of how dynamic real-world networks can be observed and modeled, and how these models can then be analyzed. We then proceed by defining the terms *dynamic* and *evolutionary network model* in 6.2 to differentiate among the different modeling approaches. As mentioned above, we are most interested in those networks in which single vertices are modeled as entities able to decide which edges to build and which to delete in a decentralized manner. We will first introduce a general framework for this kind of dynamic networks in 6.3[1], and then show by some examples how carefully the changing rules have to be designed to assure that the network will evolve efficiently towards a desired structure in 6.4, 6.5, and 6.6[2]. We start with a discussion of how to deduce a network model from a real-world dynamic network and review some methods to deduce the long–time behavior of these models, i.e., the *attractor* of the system.

6.1 Observing, Modeling, and Analyzing Dynamic Networks

We define a real–world network to be a *dynamic network* if it has a continuous identity through history even though its adjacency matrix changes over time because of the addition and deletion of edges and vertices. With this definition, a peer-to-peer network, e.g., gnutella or hyperchord, or the social network of a village is considered a dynamic network although its actual members change all the time due to entry events like logins (or birth, respectively,) and leaving events, e.g., logouts (deaths) of the users (inhabitants) since these networks show a continuous history and a continuous purpose and means of network generation.

Dynamic changes in a network's structure can be depicted as the trajectory of a point in the space of all possible graphs, as sketched in Fig. 6.1. It is helpful to imagine this space of all possible graphs as a *meta-graph* $G(\mathcal{G})$ where any two possible graphs that differ by exactly one vertex or one edge are connected by an edge. The dynamic of any network is then modeled as a path through the meta-graph of all possible graphs. It is intuitive that two graphs that differ by only one element will share many structural properties which implies that graphs that are within a short distance in the meta-graph $G(\mathcal{G})$ constitute groups with the same set of structural properties. In Fig. 6.1 we have colored a two-dimensional sketch of $G(\mathcal{G})$ to indicate that the graphs in a given region have similar structural properties, for example, a small maximal or average distance, or a tendency of the vertices to cluster. If the trajectory homes in on a set of graphs with a common structural property and does not ever leave it afterwards, we will say that this part of the meta-graph is an *attractor* for the dynamic network, and we will call the set of structural properties common to all the graphs in this attractor the *stable network structure*.

If a dynamic network is modeled by a path in $G(\mathcal{G})$ we are interested in the following questions:

1. Is there an attractor in the meta-graph and if so, what are its structural properties?
2. What is the likely path to the attractor?
3. How long will it expectedly take to get to the attractor in dependence of the starting point?

6.1.1 From Real–World Networks to Appropriate Network Models

Of course, the answers to these questions depend on the *changing rules* that govern the path of the dynamic network through the meta-graph. There are mainly two ways in which the changing rules and the resulting stable network structures of a given dynamic network are discovered.

[1] This part of the work was conducted with Michael Kaufmann and has been published in [142, 145], where it won the 'Best Paper in Track award'.
[2] This part of the work was conducted with Karin Zimmermann.

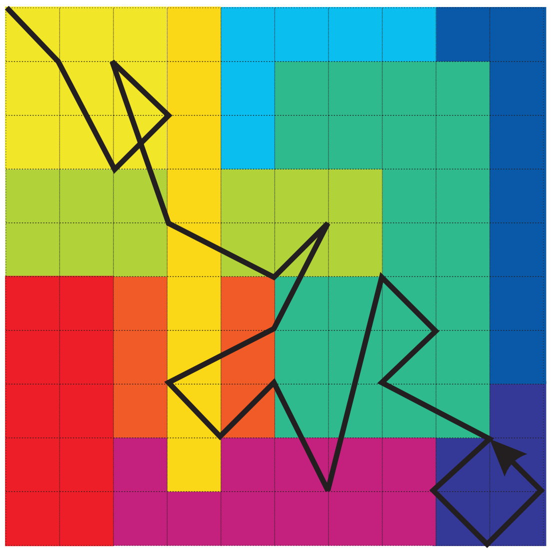


Fig. 6.1: The history of a dynamic network can be depicted as a trajectory through the space of all possible networks where two networks are connected if they differ in only one element. Colors indicate schematically groups of networks with similar structural properties.

1. The first approach studies **different** real-world networks and searches for common structural properties, as it has been done, e.g., in the analysis of the small-world phenomenon [242], scale-free degree distributions [21], and the occurrence of hierarchical clustering [201] or assortativity in real–world networks [177]. The motivation behind this approach is that when a structural property emerges in different networks, it seems to be an important feature of the attractors of different dynamic systems. After a common structural property has been found, different changing rules are then explored to determine whether they will lead to an attractor which has the structural property. This approach is an a posteriori analysis, a kind of *reverse engineering* approach. For this approach methods mainly from (social) network analysis have been used by determining, e.g., vertex centrality distributions [113, 132, 133] like the degree [242] or betweenness centrality[3] distribution [95], or the distribution of certain subgraphs [168]. A detailed overview of these methods can be found in [41, 237].

2. The second approach studies a set of **subsequent** static snapshots of the **same** network and tries to discover how changes between snapshots can be best described, e.g., whether the network preferentially changes by building relationships between vertices that are already neighbors of their neighbors [216, 217, 218], i.e., whether it prefers to build *local edges*. After some statistical relevant changing rule has been found, this can be modeled and the model can subsequently be analyzed to determine its attractor and its structural properties.

In summary, whereas the first approach knows the result and searches for the *process* that will produce it, the second approach knows the process and tries to determine the *result*. Of course, it has also to be shown in the first approach that the proposed process will actually reach and stay in the attractor.

[3] Sometimes also called the *load*. The classic betweenness centrality of a vertex measures the sum of the fractions of shortest paths between all possible pairs of vertices that contain v. Note that sometimes a slightly different definition is used under the same term [95].

When the changing rule is known or guessed, there are basically three different types of network models: *static, semi-dynamic*, and *dynamic* network models that will be discussed in the following.

6.1.2 Static, Semi–Dynamic, and Dynamic Network Models

A *static network model* is one where vertices and edges are created once and not changed thereafter. Many classical graph families such as grids, hypercubes, and random graphs belong to this type of model. A *semi-dynamic network model* allows for a finite number of changes in a basic graph topology. Many of the network models that were proposed to explain the emergence of real-world network structures belong to this class of *semi-dynamic* network models, the most prominent being the small–world network model by Watts and Strogatz, since it only allows at most one rewiring event per edge. But other network models also belong to this class such as the preferential attachment model by Barabási and Albert where every new vertex adds exactly m new edges to the growing network that are not changed thereafter. This network model shows a scale-free degree distribution as shown by [5] and was subsequently varied, e.g., by introducing a *fitness* to the vertices [31], or assigning weights to the edges [255]. Another member of the class of semi-dynamic network models is the modular network model by Ravasz et al. that starts with a small graph structure that is multiplied in every time step, connecting the resulting subgraphs by a special schema to build a larger one [201]. The resulting network shows the same clustering coefficient and dependency $C(k)$ of the average clustering coefficient of vertices with degree k as metabolic networks. Of course, semi-dynamic network models are most suitable to model those networks that are semi-dynamic themselves. For example, *citation graphs* [60][4] and the corresponding *co-authorship networks* [176] described in the last chapter belong to this class. Naturally, an edge, once acquired, will stay forever in these networks and thus they can be considered to be semi-dynamic.

There are fewer fully dynamic network models since these are often hard to analyze. In a fully dynamic network model new vertices can attach to the network and old vertices can be removed from it, and edges are rewired, newly built, or deleted. In 2000, Albert and Barabási added the possibility of deleting or adding new edges and vertices to their preferential attachment model and were able to show that this fully dynamic network model still produces scale-free degree distributions, but with more variation in the exponent γ of the distribution [5, 4]. Another fully dynamic network model for evolving social networks was discussed by Jin et al. [117]. Very interesting evolutionary network models have been proposed by Bornholdt and various co-authors, some based on game-theoretic ideas [70] and many based on genetic algorithms [59, 71, 39, 38, 40]. Both approaches give rise to a set of well-known analytical methods, and thus we will shortly sketch the general idea behind game theoretic models and genetic algorithms here.

1. **Game theoretic–based network modeling:** In a game theoretic network model, each vertex represents an agent that is allowed to build edges to other vertices. Every possible set of edges that an agent can build is called a *strategy*. After every agent has chosen its edge set, they are paid a benefit that is determined by the resulting network's structure and described by the so-called *utility function*. In a game theoretic setting, every agent is assumed to be *rational*, and to have total knowledge of the utility function of every other agent. In this model every agent tries to maximize its benefit regardless of what strategies the other agents choose. It is often assumed that the agents will choose their strategy such that no agent can benefit from choosing another strategy given that the other agents stick with their strategy, and a set of strategies with this property is called an *equilibrium* [186].

[4] In a citation graph, publications are represented by vertices and two vertices are connected if one of the corresponding publications cites the other.

Note that there are also evolutionary game theoretic models in which the agents play more than one game and are allowed to change their strategy as a result of the games played so far [244, 92]. *Network creation* or *network formation* games have been independently introduced by different groups, e.g., Myerson [171], Fabrikant et al. [79], and Skyrims and Pemantle [214], and varied by numerous others, e.g., in a dynamic setting by [238] and an asymmetric, non-cooperative setting by [15].

2. **Network modeling based on genetic algorithms:** Genetic algorithms were introduced in the 1960s by John Holland [107] to find good solutions in a large search space, and were firmly based on the idea of the optimization of a population of solutions by a selecting entity[5]. Holland's approach was a quite fine-grained copy of what happens in biological evolution, e.g., solutions were encoded as *chromosomes* with different *genes* encoding structural properties of the solutions; possible *mutation mechanisms* even modeled *cross-over* events in biological organisms where pairs of chromosomes exchange parts of their information. This first model is a bit too fine-grained for the evolution of networks, since a reasonable simulation of a cross-over event for networks is difficult to imagine. In general, an evolutionary model based on genetic algorithms will thus simply consist of a set of graphs, called a *population*, and in most cases the network structure itself constitutes the *genetic material*, without any differentiation of chromosomes and genes. Different kinds of *mutations* can be applied to the population that alter the structure of the networks and thereby change their fitness. Subsequently, the networks are evaluated by some *fitness function*, and the most fit members of the population are most likely to produce *offspring*, thereby stabilizing functional network structures.

We have sketched both approaches here since the general model of the evolution of so-called S^3 networks proposed in 6.3 integrates these ideas to make them amenable in the design of decentrally organized technical communication networks.

Process– vs. Result–Driven Network Modeling

Even if semi-dynamic network models are able to produce networks with those structural properties seen in a real–world network, such as a specified clustering coefficient or degree distribution, sometimes they do not model the **process** by which the structure is likely built in the system that generated the network. This is especially obvious in the model given by Ravasz et al. sketched above [201]. Their model results in a network that shares some structural properties with metabolic networks, but the process, namely the multiplying of the whole graph in each time step, is highly unlikely to be the biologically correct one. If this process models the evolution of real metabolic networks, this implies that a part of the genetic information encoding the genes involved in metabolism is multiplied. There are some plants that are known to have multiple sets of genetic information (*polyploid plants*), but any deviation from the normal (haploid) number of chromosomes (i.e., aneuploidy) is lethal for higher mammals like humans [105]. In summary, the model of Ravasz et al. has helped us to understand how certain properties like a $C(k)$ distribution proportional to k^{-1} *could* emerge in a dynamic network, but it is unlikely that the proposed mechanism is the one used in the evolution of real metabolic networks. The point that a process resulting in the 'correct' stable network structure has additionally to be able to model the likely way in which the dynamic

[5] As sketched by Melanie Mitchell [169] there were many scientists that used ideas from evolutionary theory to solve technical problems, e.g., Rechenberg, who was the first to use something he called "evolution strategies" for optimizing the design of different devices [202], and in 1966, Fogel, Owens, and Walsh, who described a creative technique to build new computer programs, called "evolutionary programming" [86]. Weicker states that even earlier, Friedman had used ideas from evolution theory in 1958, followed by first steps in evolutionary programming by Friedberg and Friedberg et al. in 1959 ([245], p. 56-58).

network under observation produces it, has also been stressed by other authors, e.g., in the case of social network formation [117, 214] where a constant rewiring of edges seems to be necessary to model short-lived acquaintanceships, as Jin et al. point out:

> New vertices are of course added to social networks all the time ... . However, the timescale on which people make and break social connections, which can be as short as hours or days, is much shorter than the timescale on which vertices join or leave the network, which is typically some years. [117]

This rules out static or semi–dynamic network models for social networks.

Also in the case of (scale–free) citation graphs it was argued that the likely process of their generation is not captured by the classic semi-dynamic network models producing scale–free graphs. For these networks Klemm and Eguíluz suggested assigning vertices a decreasing *activitiy* with age since it was shown that the expected number of citations of a given article per year decreases strongly 3 years after its publication [129].

Interestingly, it can also happen that the assumed network generating process will not lead to network models that capture the structure of the real-world network. If there is another network model that results in more similar networks, this network model can supply hints about the real process, as happened in the modeling of the Internet. In the early years, mainly hierarchical models based on the intuition that new servers are added to the Internet on different levels of organization were proposed. However, it turned out that networks from the preferential attachment model resemble the Internet much more by various measures—implying that the "accepted wisdom of the strong influence of the hierarchical structure" on its network generation was wrong [229].

In summary, semi-dynamic networks mainly model the large-scale addition of vertices to a network, but not the small-scale rearrangements in many evolving networks. If these small–scale rearrangements dominate the network structure of a real–world network, modeling them with a dynamic network model should be preferred over modeling them with a semi-dynamic one. In general we state that a reasonable model for the evolution of a real–world network does not only have to produce a network with the same structural properties, but also has to do so by a process that is possible in the network generating system that produced the real–world network.

6.1.3 Finding Attractors of Dynamic Network Models

Having discussed the pros and cons of different modeling approaches, we will briefly sketch the wide variety of different analytical methods that try to determine the *attractor* of a network model, i.e., methods that, given a network model, determine what kind of networks it will produce.

1. Albert and Barabási have applied methods from *statistical mechanics* to determine that their preferential attachment model results in networks with a scale-free degree distribution [21, 5];
2. Farkas et al. used methods from *linear algebra* (and numerical simulations) to determine the so-called spectrum[6] of random graphs, small-worlds and scale-free graphs for $\lim n \to \infty$ [81];
3. Newman and Watts used *renormalization group analysis* and *scaling theory* to determine the expected average distance in small-worlds [175, 174];

[6] The *spectrum* of a graph is given by the *eigenvalues* of its adjacency matrix.

4. Newman, Strogatz, and Watts used *generating functions* to analyze various properties of random networks with a given degree sequence, e.g., the expected clustering coefficient and average distance [182].

5. *Stochastic models* used in social network analysis try to fit stochastic models governed by a minimum of parameters to the dynamic network data as closely as possible [216, 217, 217, 223]. If the parameters are in a narrow range these can be interpreted to mirror stable network structures.

6. Another approach is to analyze simpler models first. For general dynamic vertex attachment models where in each time step one vertex is added with m edges according to some attachment rule (e.g., preferential attachment), simpler so-called (random) *recursive tree models* have been developed where the new vertex builds only one edge, and thus the growing graph constitutes a tree. On this method a rich body of theories has been based, mainly for random attachment, but also applicable to other attachment rules, e.g., [164, 170, 228, 114, 137].

7. Often, a dynamic network model can be viewed as a *discrete Markov process* in the meta-graph of all possible graphs. A *discrete Markov process* is a stochastic process in a *state space* with defined *transition probabilities*, i.e., $P_i(X, Y)$ describes the probability that the system is in state X in time step i and in state Y in time step $i + 1$. In the case of a dynamic real–world network model, the state space is given by the meta-graph of all possible graphs $G(\mathcal{G})$, and for any given changing rule, the *transition probabilities* can often be easily determined. Markov theory can then be used to determine, e.g., *attractors* (here called *absorbing states*) or—if certain conditions are met—the stationary probability distribution that determines which states will be occupied and expectedly how often for $\lim t \to \infty$ [42, 163, 183].

8. In a game theoretic model, the question for stable network structures is mainly assumed to be equal to the set of *Nash and other types of equilibria*, i.e., those network structures in which no single agent can improve its gain by switching to another strategy [186, 244].

Fig. 6.2 summarizes this perspective on the observation, modeling, and analysis of a dynamic network: A dynamic network follows a trajectory through $G(\mathcal{G})$, guided by a set of changing rules. By collecting static snapshots of the network along this trajectory and analyzing their structural properties, stochastic models can be built to deduce the changing rules of the system. These rules can then be incorporated into a network model which is subsequently analyzed by different mathematical methods as summarized above.

So far, we have used the term *dynamic* network, but one can often find the term *evolving* network or *evolution of networks*. In the following we want to discuss when a *dynamic* network might be called an *evolving network*, and then develop a general framework for *evolutionary network models*. We will argue that evolutionary network models might help to develop decentrally organized networks with a globally desired network structure, and show with some examples how important it is to design changing rules for this purpose carefully.

6.2 Evolution of Complex Networks

The term *evolution of networks* has been used repeatedly for any kind of network that shows a change in its adjacency matrix due to the addition or deletion of vertices or edges, e.g., [5, 255]. But the simple change of the adjacency matrix would rather allow for the term *dynamics of networks* or the characterization as *dynamic networks*. The question is thus why and when the term *evolution*

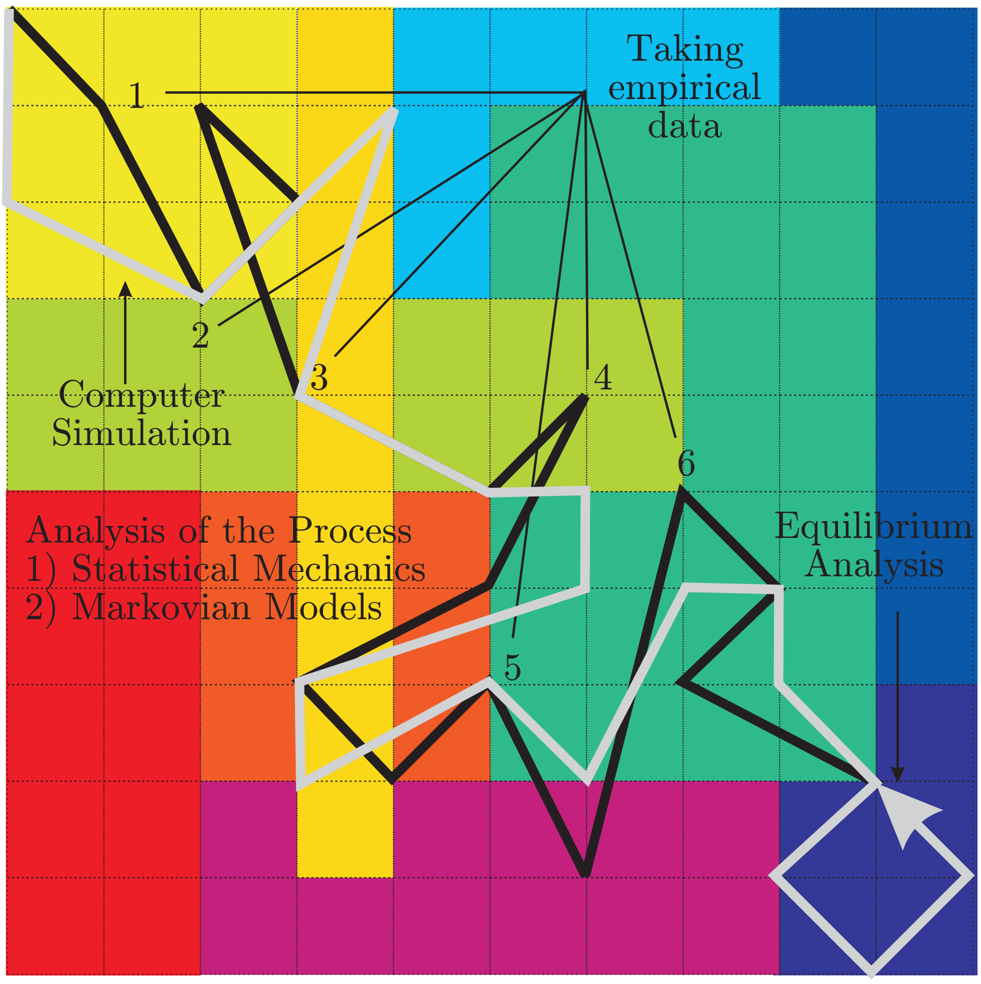


Fig. 6.2: Summary of the target of different analysis methods: Along the trajectory (black) of a dynamic real-world network, data can be collected and analyzed by stochastic models to determine the driving forces and stable structural properties. These can then be transformed into a dynamic graph model. To analyze the long–run behavior of the model, an *equilibrium analysis* can be made. To understand the (likely) trajectory that the evolving graph will take, methods from *statistical mechanics* or *Markov chain models* can be applied. Finally, a *computer simulation* of the model (gray trajectory) can be made to compare the trajectory of the model with a series of real data .

of networks is appropriate. To our knowledge, this rather hermeneutic discussion has not been done so far, but we think that such a discussion will reveal *implicit assumptions* we have on the dynamics of networks that help us to build better models for their dynamics. As Dorogovtsev and Mendes point out in their book "Evolution of Networks"[7]:

> However, the most important natural and artificial networks have a specific architecture (...). Their state is far from equilibrium and their structure cannot be understood without insight into the principles of their evolution. ([69], p. v)

If we need to know about their evolution, what exactly is the *evolution of networks*? In the following we will first determine what kinds of systems are said to be *evolving* in general and then try to adapt the concept specifically for network generating systems.

6.2.1 Evolution of Systems

Of course, the term *evolution* itself is mainly used to describe the development of biological organisms, and is thus not easy to translate to the changes of a network, but on the other hand the principles on which evolutionary theory was based by Darwin [58] can and have been translated to other problems by, e.g., genetic algorithms, as sketched above.

Abstracting from these approaches, a system can be said to *evolve* if it is dynamic, i.e., changing its characteristics, according to the following basic principles:

1. The system generates offspring with mainly the same characteristics as itself (**Inheritance**).
2. The offspring show variations of the parental characteristics (**Mutation**).
3. The system's ability to survive the selection is based on its characteristics (**Fitness**).
4. There is a selection mechanism based on the fitness of the network (**Selection**).

For every evolving system there is a virtual space consisting of all possible system states. If there is a fitness function that assigns a value to each of the possible states, this space is also called the *fitness landscape* of the system [169].

For biological networks like protein-protein interaction networks [199, 219, 232], or metabolic networks [116, 201] it is quite clear that they have *evolved* within this framework. Both types of networks are part of an *organism* (the **system**) and their characteristics are encoded in the genetic code of this organism. Thus, they can be **inherited** by the offspring of the organism whereby they may suffer some **mutation**. Of course, the quality of these networks does not totally determine the **fitness** of the organism, but it does at least influence it. One simple example of the influence that metabolism has on the fitness of an organism is an enzyme called Gulonolactonoxidase that catalyzes the synthesis of vitamin C. This enzyme was lost in humans and guinea pigs, making vitamin C an essential nutrient for these organisms to survive and reducing fitness in those environments where vitamin C is scarce [162].

While the evolution of biological networks can thus be modeled meaningfully as a series of mutation and selection, evolution of other networks such as the social network in a city or the network of streets in a country is not determined by selection since there is no alternative network with which these singular networks have to compete. In analogy to the above given properties of an evolving system we will show why it is still meaningful to speak of the *evolution* of these networks:

[7] Interestingly, the authors do not discuss their usage of the term *evolution* as opposed to *dynamic*.

1. Let G_i and G_{i+1} describe the same network at two subsequent time steps[8]. As long as edges are costly to build and delete we can assume that only a few changes have been made to the adjacency matrix of G_i to result in G_{i+1}. Thus, most of the structural properties of G_i will be preserved in G_{i+1} (**Inheritance**).
2. The changes can be seen as a kind of **mutation**.
3. Under the assumption that edges are costly to build, to maintain, and to delete, it is obvious that no real network is built without the wish to use it for something, e.g., a means of transport, information exchange, or creation of knowledge or wealth. For example, in recent years larger and larger teams and collaborative networks in science have been built to achieve new insights, and seemingly they were built efficient enough to do so [19]. In summary, networks are built with a *purpose* and their structural properties determine their **fitness** for the given purpose.
4. Depending on the fitness of a given network it can be *selected* in two different meanings of the word. If the network is *used* by intelligent agents they can decide whether they want to use it or not. For example, every country has different transportation networks: the network of streets and highways, train networks, and airplane routes. A traveler will make his choice among them depending on the distance, the maximal amount of travel time he wants to spend, money, and the structural properties of these networks. On a second level, the participants or the owner of a network can decide whether they want to leave the network structure as it is in the next time step or whether they want to change it because they are unsatisfied with the performance of the network. Thus on the first level, if there are different singular networks with the same purpose (transport, communication, etc.), agents can decide which one to use. On the second level, agents can decide to change the network's structure according to the fitness of the current structure.

This latter aspect gives rise to an interesting feedback mechanism that is typical for complex adaptive systems [233] and can either make it very difficult to control them or give us a tool by which the system can be made to evolve towards an attractor, a stationary state as in chaotic systems [226].

Note that a very detailed modeling of evolutionary processes that incorporates the ideas of *chromosomes*, *cross-over*, or *genes* as proposed in early genetic algorithms by Holland [107, 169] is not necessary to call a model an *evolutionary* model. Interestingly, the life–long aim of paleontologist Stephen Jay Gould was to decipher the *"Structure of Evolutionary Theory"* [98], and he comes to the conclusion that *natural selection* (together with *heredity*, *variation*, and *overproduction of offspring*) is just the sheer mechanism of evolution, but that a theory must contain, explain, and be based on the following three principles in order to be called an *evolutionary theory* (at least for the advance of biological organisms):

1. *Agency*: Natural selection aims at the level of *organisms*.
2. *Efficacy*: Natural selection is able to create new solutions to a problem.
3. *Scope*: Natural selection is able to create **every** new solution in organisms.

In order to design an *evolutionary model of complex networks* it seems at first glance that there are two different *agencies* for selection: One on the level of the whole network and one on the level

[8] The length of the time interval between these snapshots has to be determined meaningfully in every network or is given by the data at hand.

of single elements of the network. As depicted above, the first type of selection occurs if there is more than one network for the same purpose, allowing people to choose which one to take. The second type of selection occurs whenever the structure of a network needs revision. If an edge contributes to the functionality of a network, it is likely to stay in the network; if a new edge would improve the functionality of the network, it is likely to be built. In essence, the first type of selection will finally also lead to a selection of the elements of the network: If a transport network is not used by the customers, the owner of the network will select those edges to build or to destroy that optimize the functionality. Thus, in order to model the evolution of a complex network, we can disregard the first type of selection and concentrate on the second. Furthermore, we will only concentrate on those models where the central mechanism of selection is able to create new—and all new—structures of the network.

6.2.2 Dynamic vs. Evolutionary Network Models

Summarizing the ideas sketched above on the evolution of biological organisms, we will denote as a *dynamic network* every network that changes its adjacency matrix over time. A network is said to *evolve* if the building and deleting of edges is costly and if, therefore, we can assume that the network is built with a purpose. From this it follows that the network's structure at least partly determines the fitness of the network with respect to the purpose. Furthermore there must be a selection mechanism that makes it more probable that a functional network structure is stabilized and that a less successful network structure is changed in a way that explores the set of all possible graphs for a better solution. With this definition, the small-world model and the preferential attachment model are definitely dynamic network models, but they are not evolutionary network models. The 'purpose', so to speak, of the small–world model is to build small networks. But if the first rewiring steps fail to build a small network by chance, there is no mechanism to rewire these edges again to build a smaller network in a second try. Similarly with the preferential attachment model, where the 'purpose' can be described as building a scale-free degree distribution. But if the network fails to build one there is no mechanism that allows for a rewiring and subsequent selection of those rewiring steps that lead the network towards a more scale-free degree distribution.

Why should one be interested in an evolutionary network model in addition to a pure dynamic network model? The main difference between these types of network models is the following: in a dynamic network model we aim for a certain structure and try to find a changing rule that will result in this network structure. To stay with the picture of the meta-graph $G(\mathcal{G})$: A dynamic network model chooses a narrow trajectory through the meta-graph such that the attractor of this model shows the desired structural property. In such a model, no selection mechanism is needed to guide the trajectory of the system since the changing rule will most likely only explore a small part of the possible search space, as in the small–world model by Watts and Strogatz where the rewiring mechanism results in a small network with very high probability and maintains the high clustering coefficient with high probability for an appropriately chosen rewiring probability p_{rew} [242].

In an evolutionary network model, we aim for a **global structural property** and allow the network to try every solution it can find in $G(\mathcal{G})$, i.e., for a small–world model the model would fix a wanted average clustering coefficient and a wanted average distance. As long as these criteria are not met, the agents would apply a changing rule that changes the network's current structure until they have concertedly achieved a satisfying network structure. Thus, the second type of model is not restricted to a certain part of the fitness landscape, and here the changing rule can be *creative* in finding its own solution to maximize the fitness of the network. Furthermore, depending on the robustness of the changing rule, a self-organized network developed by an evolutionary mechanism implemented in the agents can be hoped to react flexibly to changed environmental properties. Suppose for example that external events destroy a part of the network. Then, an evolutionary network model allows for a rewiring of the structure to deal with the new situation (we will show such an example in 6.6). Of course, to make these models useful in a technical network, it is still required that the changing rule is designed such that *likely solutions* can be easily found by it, as we will show later.

6.2.3 Evolutionary Network Models for Selfish Agents

In this chapter we are especially interested in the evolution of those networks where each of the vertices represents an agent that can decide for itself which edges to build and which to delete, similar to the main assumptions made in game theory as sketched above. It is reasonable that any agent that has to invest in the building or deletion of an edge will only do so if there is an immediate return of interest due to this change, as stated by Fabrikant et al. for the emergence of the Internet's structure:

> The Internet is the first computational artifact that was not designed by one economic agent, but emerged from the distributed, uncoordinated, spontaneous interaction (and selfish pursuits) of many. [79]

On the other hand, every agent that wants to use such a decentrally organized network will benefit from it only if its global structure is functional. It can be shown that the behavior of processes on different kinds of network structures changes dramatically, e.g., the cascading behavior of failures in power grids modeled as small-worlds [174, 240], the spreading of diseases [192], or navigation and routing [125, 126, 108]. One important finding concerned *scale-free* networks, i.e., networks dominated by vertices with a small degree, that also containing some high degree vertices, so-called *hubs*. It was shown that a scale-free network is very stable against random failures and at the same time very vulnerable to directed attacks on the high degree vertices [6]. Thus, for many technical communication networks that have to be decentrally organized for various reasons, it is an interesting task to develop an evolutionary network model that obeys the fact that each change in the network has to bear a benefit for those vertices involved in the change and simultaneously results in a globally functional network such that the overall benefit to the vertices in the network is satisfactory.

In the next section we will introduce a generalized evolutionary network model for these kinds of decentrally organized networks, and show some examples of how efficient changing rules can be designed to guide the evolving network's trajectory to a region of the meta-graph with a desired stable network structure.

6.3 A Generalized Evolutionary Network Model for Singular Networks

In this section we will develop an evolutionary network model for so-called S^3 networks that are *singular, selfish*, and *self-organized* networks. A network is called *singular* if at every timestep there is only one network that is built directly from the one in the time step before. A *singular network* is not evaluated by some entity from outside the network but by the vertices within the network, that are then able to change parts of the network as a result of this evaluation. Thus, the network's topology evolves in a decentralized and *self-organized fashion*. As examples of the evolution of this kind of singular and self-organized networks we want to discuss briefly the evolution of the network of streets and highways and the evolution of the global social network in which all humans participate: Singular networks are built between rational entities, called *agents* and these entities are able to evaluate the network's current performance with respect to their own position in the network, but they are not able to evaluate the global performance of the network. In the case of social networks a person might evaluate her position in the network by how much information she gets from her social environment, while the network of streets and highways might be evaluated from each of the connected cities with respect to the average time it takes to get to any other city. The consequence of such an evaluation process is that vertices unsatisfied with the network's current topology, i.e., the part that is captured by their egocentric evaluation method, will try to adapt the network to improve their situation, using some changing rule. In our examples this might cause a person to make new friends, or a city to build a new freeway. Since most edges in real-world networks come with a cost, the number of changes in each time step can be assumed to be small and it can be additionally assumed that every vertex removes or builds only edges it participates in. Another important point to note is that in most cases it will not be totally clear for the unsatisfied vertex which edges should be changed to improve the performance. This can be easily seen in the case of social networks where a person is only able to survey a small part of the whole network, implying that the change of edges is partially a random process. In summary, singular networks built by selfish agents evolve in a series of evaluation processes from single vertices within the network, making small changes to their local edge sets in a random process defined by some *changing rule*.

The framework that is introduced in the following section provides a way to describe this kind of self-organized evolution and to analyze different changing rules with respect to the expected runtime until a network with a satisfying topology has emerged. This latter analysis is important because only if the expected runtime is known can it be decided whether a given changing rule is efficient enough to be implemented in technical communication networks, as ad-hoc communication networks or peer-to-peer networks. Some first work on how to analyze simple evolutionary algorithms with respect to the expected runtime has already been done, mainly by the group of Ingo Wegener [169, 249, 210], but to our knowledge we are the first to do it for more complex evolutionary algorithms that try to build functional network topologies in a self-organized and decentralized fashion.

6.3.1 The Model

We introduce an evolutionary network model that describes the evolution of networks with the following set of properties:

1. There is only one, singular network at any time step.
2. The vertices evaluate the network's fitness relative to their own position in the network.
3. No vertex knows the adjacency matrix of the whole network.
4. The network evolves so slowly that at any time step only one vertex is active, i.e., the model is *synchronous*.
5. All vertices decide independently which edges to build and which to remove.
6. Each vertex behaves selfishly, i.e., it will build a new edge to improve its own situation, not to improve the situation of others. This implies that a vertex will only insert and remove edges it participates in.

7. There might be external events, e.g., birth, death, failure of or attacks on a subject or object that is represented by a vertex or an edge of the network. These events are modeled by addition or deletion of vertices and edges.

The evolution of networks with these properties will be called evolution of **singular**, **selfish** and **self-organized (S^3)** networks and in the following we will formalize their evolution.

Let $G_t = (V, E_t)$ denote the graph at time step t where V is a set of vertices and E_t is a subset of $V \times V$. All edges $e = (u, v)$ will be regarded as undirected, i.e., $(u, v) \in E_t$ iff $(v, u) \in E_t$. $\mathcal{G}$ denotes the set of all possible graphs on the set of vertices V.

The fitness of a vertex v in graph $G \in \mathcal{G}$ is given by $f : \mathcal{G} \times V \rightarrow \mathbb{R}$, i.e., the performance of a graph is always evaluated with respect to one vertex of the graph. This leaves room for designing a 'selfish' or 'egoistic' evaluation function. Note that the general form in which the model is defined allows the value of $f(G, v)$ to be the same for all vertices v. If this is the case, the function measures the fitness of the graph, independent of a given vertex. But as we understand our model, it is important to use functions that in principle give different values for different vertices.

A minimally required performance is additionally defined for every vertex v by $f_{min} : V \rightarrow \mathbb{R}$. In the following we will use the same constant value f_{min} for all vertices. Let $E_t(v)$ denote the set of edges adjacent to v in time step t: $E_t(v) = \{(u, v)|(u, v) \in E_t\}$. $\mathcal{E}(v)$ denotes the set of all possible edge sets $E(v)$ on $v \times \{V \backslash v\}$: $\mathcal{E}(v) = \{E(v)|E(v) \subseteq v \times V \backslash v\}$. Further, we define two functions that change the set of edges adjacent to v: $C_+ : V \rightarrow \mathcal{E}(v)$ and $C_- : V \rightarrow \mathcal{E}(v)$. These are also called the *changing rules* of the algorithm. Note that every vertex can change only its own vertex set, i.e., it can only build and remove edges attached to itself. This rule - together with the evaluation of the network relative to itself - also reflects the selfishness and limited knowledge of vertices. The latter guarantees that only *local decisions* are made.

The evolution of a network can now be modeled by the following steps (s. Fig. 6.3):

1. Initialize G_0 with (V, E_0).

2. In every time step t choose a subset V' of vertices and evaluate $f(G_t, v)$ for all of them. For all v with $f(G_t, v) > f_{min}(v)$ let v build a new set of adjacent edges $N_{t+1}(v)$ as the result of $C_+(v)$, otherwise the new set of adjacent edges is the result of $C_-(v)$. The new graph G_{t+1} is thus composed of the old edge set without the old neighborhood of v, combined with the new neighborhood of v, i.e.,

$$G_{t+1} = (V, (E_t \backslash \bigcup_{v \in V'} E_t(v)) \cup \bigcup_{v \in V'} N_{t+1}(v)). \tag{6.1}$$

   If not stated otherwise, V' will consist of one vertex chosen at random from V.

3. In a last step, those events that change the network from the outside, can be modeled by adding and/or deleting vertices and/or edges.

Note that the function $f(G_t, v)$ can be seen as an objective function that is personalized to v and that it can be combined with any rules $C_+(v)$ and $C_-(v)$ as long as these are local.

This model describes an iterative process of evaluating and manipulating the network's structure in a decentralized way and thus we consider it to be a kind of evolutionary algorithm [142]. With respect to Gould's definition of an evolutionary theory, we state that the *agency* of selection in this model is the set of edges because the edges are the focus of the changing rules. The selection

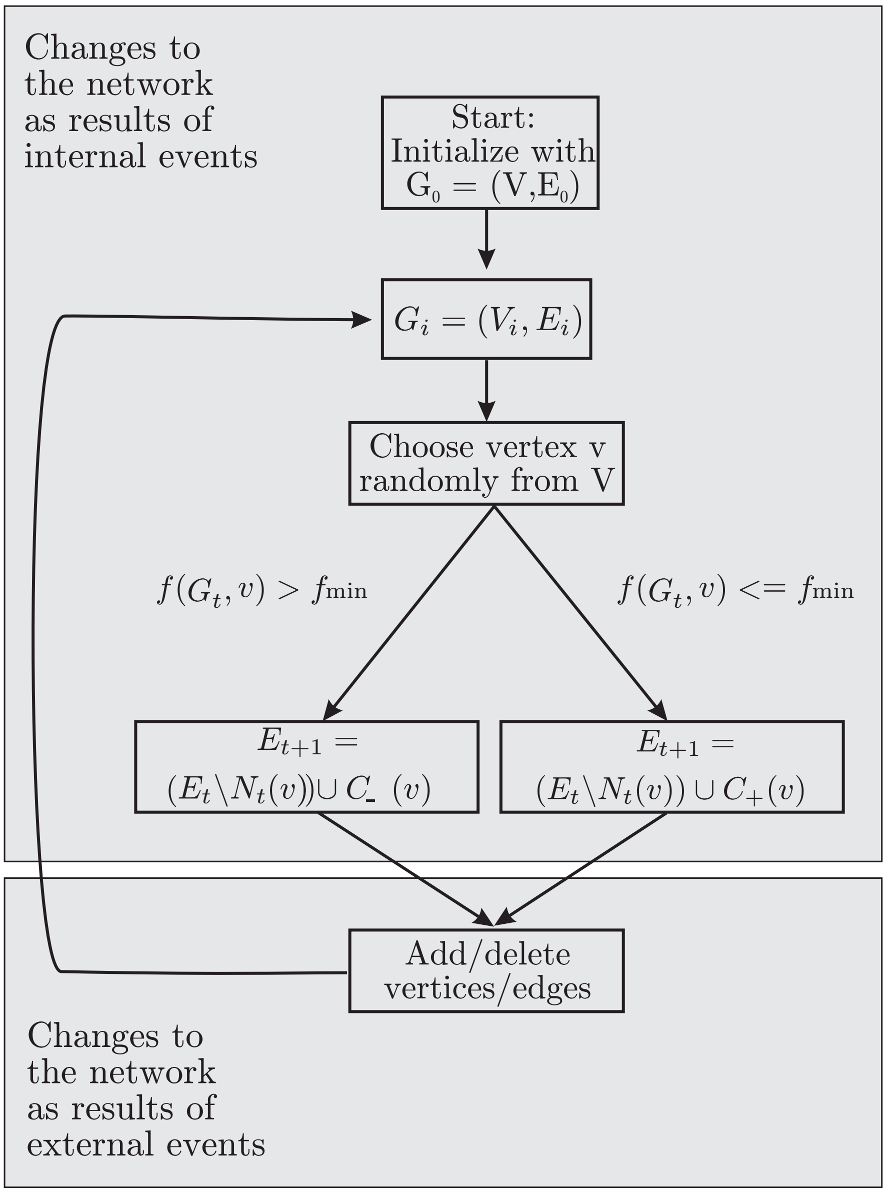

Changes to the network as results of internal events
Start: Initialize with $G_0 = (V,E_0)$
$G_i = (V_i, E_i)$
Choose vertex v randomly from V
$f(G_t, v) > f_{\min}$
$f(G_t, v) <= f_{\min}$
$E_{t+1} = (E_t \backslash N_t(v)) \cup C_-(v)$
$E_{t+1} = (E_t \backslash N_t(v)) \cup C_+(v)$
Add/delete vertices/edges
Changes to the network as results of external events

Fig. 6.3: A general model for the evolution of decentrally organized S^3 networks. The model begins with a graph $G_0 = (V, E_0)$. In each time step t one of the vertices evaluates the fitness of G_t with respect to itself by calculating the value of $f(G_t, v)$. If this is higher than a given minimal value f_{min}, the vertex will change its current neighborhood $N_t(v)$ to a neighborhood given by the changing rule $C_+(v)$. If the evaluation yields a value smaller than or equal to the minimal value f_{min}, the vertex changes its current neighborhood according to $C_-(v)$. In a last step, G is eventually modified as the result of events that happen outside the modeled system, e.g., the death of a person in a social network.

mechanism is given by a combination of $f_{\min}$ that determines what kind of changing rule is applied in a certain situation, and the changing rule itself. Of course, a simple addition or rewiring of edges would be enough to assure *efficacy*, i.e., the ability to create new solutions, and *scope*, i.e., all possible solutions. In the following we will show that changing rules can be a bit more selective in order to reduce the length of the random walk through $G(\mathcal{G})$ to the desired attractor, i.e., they should be efficient.

Efficiency is of course an essential requirement to use the evolutionary network model for building robust technical communication networks. In 6.4 we will present two instances of this evolutionary algorithm for a function $f(G_t, v)$ that is related to the diameter of a network, and where the rules are designed to minimize the diameter. With this example we will show that it is possible to guarantee a global network structure in a local, decentralized model where every vertex is myopic and selfish, but we will also show that the exact design of the changing rules can drastically influence the efficiency with which this structure is evolved: analyzing two sets of rules for minimizing the diameter of a tree, we can show that the expected runtime, i.e., the number of time steps until a favored network structure arises, can be either exponential or polynomial with only minor changes in the rules. Thus, if some of structural properties are required in the decentrally evolved network, the problem is in designing those local changing rules that guarantee that every change to the network is locally favourable and at the same time guides the network towards that part of the search space where the networks show the globally favored network structure.

We will especially show that not every local rule that *could* result in a desired network structure is efficient to do so because the local rule has unwanted and unforeseen side effects. In his book "Der Kobra-Effekt" (The Cobra-Effect) [213] Horst Siebert is concerned with laws designed to reward a desired behavior but that turned out to be ill-posed, as exemplified by the following story Siebert tells: Once, when India still belonged to the Commonwealth, it is told that there was a plague of cobras. The governor promised a reward for every cobra head delivered to him, with the intention that the people would thus be interested in catching the snakes. It is clear that such a reward is in principle able to steer the behavior of people such that they will catch snakes, thereby stopping the plague of cobras. But the reaction of the people was quite unforeseen: Instead of doing the dangerous job of catching the snakes, people started to breed them and bring their heads to the governor, thus resulting in an **increase** in the population of cobras. As illustrated by this legend, a careful design of local changing rules is needed to achieve a global goal, and—coming back to network design—it is necessary to give the right incentives in a network of selfish agents so that they can build a globally functional network.

In summary, our evolutionary network model is applicable to those settings in which the following properties are fixed:

1. vertices are assumed to be myopic and selfish;
2. the global network properties that are desired to guarantee a globally functional network are provided;

and where the local evalution function and the changing rules can be influenced by the system. Primarily, we think of technical communication networks because here the changing rules and the local evaluation function determining when the changing rule will be applied can be implemented by the technical device or software that connects the user to the network. But we think that exploring this model theoretically is also of general value for complex systems science because it shows for a special emergent property, namely the collective network building process, what kind of network structures can be collectively built in a complex system. We will now discuss changing rules for different desired global network properties.

6.4 The Design of Efficient Changing Rules - A First Example

The first task we want to solve by the above given evolutionary network model is to shrink a graph's diameter to a desired size by keeping the number of edges constant[9]. Since the diameter of a network is a *global* network structure, we have to design a local changing rule that considers the selfishness of the vertices and gives them an increased benefit if they find an edge that globally reduces the diameter or seems at least promising to do so in the long run.

The following centrality values are needed, as described in detail in [132]. The *eccentricity* $ecc(G, v)$ of a vertex v in graph G is defined as the maximal distance of v to any other vertex y:

$$ecc(G, v) = \max_{y \in V} d(v, y) \tag{6.2}$$

The *diameter* $D(G)$ of a graph G is defined as the maximal eccentricity of any vertex v in the graph:

$$D(G) = \max_{v \in V} ecc(v) \tag{6.3}$$

Seen from the perspective of the vertex, the eccentricity represents something like a 'personalized diameter' of a graph. The *closeness centrality* $close(G, v)$ of a vertex v in graph G is defined as the sum over all distances from v to any other vertex y:

$$close(G, v) = \sum_{y \in V} d(v, y) \tag{6.4}$$

The *Wiener index* $W(G)$ of a graph G is defined as the sum over the distances between all pairs of vertices u, v. This equals the sum over all closeness centrality values:

$$W(G) = \sum_{v \in V} close(v) \tag{6.5}$$

Equipped with these definitions we can now introduce two instances of the evolutionary algorithm for S^3 networks and analyze their expected runtime behavior.

The goal of the following two algorithms in the framework of the evolutionary network model given in Fig. 6.3 is to provide the vertices in a network with a selection mechanism that enables them to reduce the diameter of the network to a desired value. In this scenario we disregard any external events that could change the network; the only changes to the network are controlled by the vertices within the network. Since every vertex has only a limited view of the network it cannot know which edge will be the best to build. Both algorithms choose one vertex randomly in every time step. If its current eccentricity is greater than the desired diameter it follows that the current network has a diameter greater than the desired diameter. The vertex then tries to rewire one of its edges to improve its own situation.

It is clear that different adaption rules might solve this problem, including even a totally random insertion and removal of edges. Here, we will show that the design of efficient changing rules is essential for the efficiency of the algorithm in constructing a network topology with the desired property. For this, we analyze the *expected runtime* of the two algorithms given below. The *expected runtime* denotes in our case the expected number of time steps t from G_0 to a graph G_t such that for all time steps $t' > t$ $D(G_{t'}) \leq k$ where k is the desired diameter, i.e., the evolutionary process has found some attractor in $G(\mathcal{G})$. We will now discuss the algorithms in detail.

[9] Note that this is a toy example to show how difficult the design of efficient local changing rules is. Of course, to shrink the diameter of a network to a wanted size, especially to 2 as we will do in the following, is no task that would reasonably emerge in a technical communication network. It is just an example to make a point about the design of local changing rules.

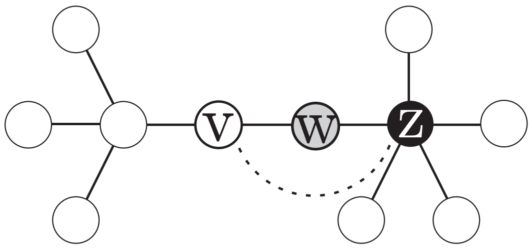


Fig. 6.4: In Algorithm 1, one vertex v is chosen at random in every time step. If its eccentricity is greater than f_{min} it will try to connect to a non-leaf vertex in distance 2 (black vertex). Let z be the second neighbor chosen and w be the vertex connecting both. Then edge (v, w) will be replaced by edge (v, z) if the eccentricity of v does not increase due to this process.

Algorithm 1

Let G_0 be a connected tree on a set of vertices V. The fitness of any vertex in the tree is evaluated by its eccentricity:

$$f(G, v) = ecc(v) \tag{6.6}$$

f_{min} can be set to any value between $n-1$, the maximal possible eccentricity, and 2, the minimal possible diameter for any tree with more than two vertices. In every time step t, choose one vertex v from V at random and calculate $f(G_t, v)$. Changing rule $C_+(v)$ is defined as follows (s. Fig. 6.4):

1. Choose one of the non-leaf vertices z in distance 2 to v at random. Let w be the vertex that is connected to both v and z.
2. Generate a new graph $G_t^*(v, z)$ by replacing edge (v, w) by edge (v, z).
3. If $ecc(G_t^*(v, z), v) \leq ecc(G_t, v)$ then set $G_{t+1} = G_t^*(v, z)$, otherwise set G_{t+1} to G_t, i.e., if the maximal distance of v to any other vertex is not increased due to the rewiring then the new graph is kept; otherwise the old one is restored[10].

Note that we ignore second neighbors z with degree 1 (a leaf) because they will always increase the eccentricity of v if the edge to z were built.

This rule will maintain the graph's connectedness and thus the graph will be a tree at any time. Furthermore, v will only remove edges that it participates in and can only initiate the building of edges it participates in, i.e., C_+ manipulates only the direct neighborhood of v. There will be no change in the graph if $f(G_t, v) \leq f_{min}$, i.e., $C_-(v) = E_t(v)$.

In the following we want to show that the above given evolutionary algorithm will eventually build a tree with a diameter smaller than or equal to f_{min}, independent of the initial tree. It is easy to see that a graph G_t with $D(G_t) \leq f_{min}$ will not change any more, and thus the runtime of algorithm 1 is defined as the number of time steps until such a tree is built. First, we will show that the process can be described as a Markov chain. We define the set S of states $\{s_1, s_2, \ldots, s_k\}$

[10] It is often remarked that the eccentricity itself is not a local network measure. We will deal with this question in 6.4.1.

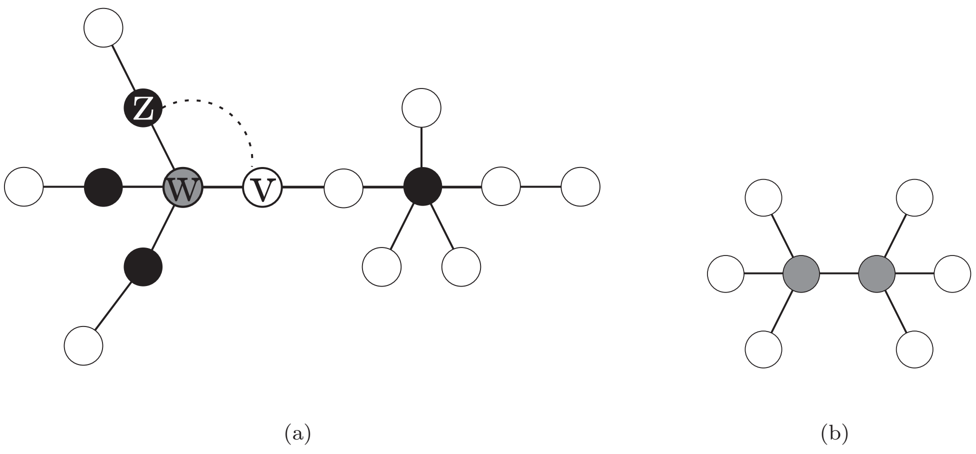


Fig. 6.5: (a) The graph has a diameter of 7, the chosen vertex v has an eccentricity of 4. If v now chooses second neighbor z and replaces edge (v, w) by (v, z), its eccentricity will not change, but the diameter of the graph has actually increased to 8. (b) All vertices in white have an eccentricity of 3. If f_{min} were to be 2, none can change the edge set if strict improvement of eccentricity is required for an edge replacement.

as the set of connected trees on vertex set V. Since any subset of graphs on a given set of vertices is finite, S is also finite. Let every state s_i represent one connected tree $s_i = (V, E_i)$. State s_i is connected to s_j if their edge sets E_i and E_j obey the following relation:

$$\exists\, v, w, z \in V \mid E_j = (E_i \cup (v, z)) \setminus (v, w), \tag{6.7}$$

$$deg(z) > 1 \text{ and } ecc(s_j, v) \leq ecc(s_i, v) \tag{6.8}$$

The weight on this edge equals the probability that a tree s_i in time step t will be changed into tree s_j. This probability is equal to $1/n_2(v)$, where $n_2(v)$ denotes the number of non-leaf vertices w that are in distance 2 to v, also called the *non-leaf second neighbors of v*. Since this number depends only on the structure of E_i, the transition probability between two states is independent of how the network has evolved and is thus constant in time. Note that in the Markov chain the edges are directed, i.e., there is a designated source and target vertex for every edge. We will denote directed edges as $[s_i, s_j]$. Both edges, $[s_i, s_j]$ and $[s_j, s_i]$, exist iff E_i and E_j obey Equ. 6.7 and $ecc((V, E_i), v) = ecc((V, E_j), v)$. All states s_i with $D(s_i) \leq f_{min}$ will be denoted as *final states*.

We will first state that the network will eventually evolve into a network with a diameter smaller than f_{min}.

Lemma 6.1
For $t \to \infty$, $D(G_t) \leq f_{min}$.

Proof 6.1
If $D(G_0) \leq f_{min}$, then $\forall t : D(G_t) = D(G_0)$. Let now $D(G_0)$ be greater than f_{min}. Then, there is at least one vertex v with $ecc(G_0, v) > f_{min}$. We will now show that there is always a path from state s_0, representing G_0, to a tree $G_{t'}$ at time t' with $D(G'_t) \leq f_{min}$. Let z, z' be two vertices with maximal distance in G_t. Let $P(z, z') = \{z, z_1, z_2, \ldots, z'\}$ be the path between z and z'. If now z is chosen, it has an eccentricity greater than f_{min}, and if it itself chooses its second neighbor z_2

then the eccentricity of z will not increase. Thus, edge (z, z_2) will replace edge (z, z_1). If in each time step only vertices with maximal eccentricity are chosen, this process will eventually lead to a tree with a diameter decreased by one and eventually to a tree with the desired diameter. The probability for such a way through the states is small but non-zero.

Since every tree having a diameter of at most f_{min} (final state) will not be changed any more and there is always a path with non-zero probability from every state s_i to some final state, the system will eventually reach one of these states and remain there.

Fig. 6.5 a) shows that the rule given above has some problems: owing to the change of perspective on the evaluation of the network in every step, it can easily happen that a step in the 'right' direction, i.e., to a tree with lower diameter, is reverted in the following step by another vertex that cannot 'see' the improvement of the previous step. This is one problem. Another problem arises if many vertices have to move together in the same direction before the diameter of the whole tree decreases. An example of such a situation is given in Fig. 6.5 b): If $f_{min} = 2$ is required, this can only be fulfilled if $n-1$ vertices are connected to one central vertex, i.e., if the tree constitutes a star. In the example given in Fig. 6.5 this implies that either of the three vertices of one side have to flip to the other side under the given changing rule. The inner vertices will never change their edge set, and any outer vertex v that is chosen to evaluate $f(G_t, v)$ will instantly flip sides because its only non-leaf second neighbor is the opposite inner vertex and because this flip will not increase its eccentricity. After the first time step there will be four vertices on the one side and two on the other. The only way to proceed to the star is if after this step one of the remaining two vertices changes side, and after that that the remaining vertex flips sides. Let x be the number of vertices on one side and $n-2-x$ the number of vertices on the other side. W.l.o.g. let $x \leq n-2-x$. The probability that any vertex of the minority flips sides is thus $x/(n-2)$ and the probability that one of the majority flips sides is $1-(x/(n-2))$. This situation can be described by the famous 'Urn of Ehrenfest' model and we can use Ehrenfest's analysis of the expected number of steps until all vertices of one side have flipped which results in $\frac{1}{n-2}2^{n-2}(1+O(n-2))$ [42].

Theorem 6.1
There is a family of graphs such that with $f_{min} = 2$, the expected runtime of Algorithm 1 is bounded by $\Omega(2^n)$.

Note that for $f_{min}(v) = 2$ any tree will eventually evolve into a member of this family.

From this, it seems obvious that the changing rule $C_+(v)$ is not constructed sensibly: Of course, backward steps could be prevented by allowing only those steps that lead to a strict improvement of the vertex' eccentricity. But as Fig. 6.5(a) shows, this small adjustment will eventually lead to trees in which none of the vertices can change anything despite the fact that the tree still has a diameter that is higher than desired. We will next show *how* the changing rule can be adapted such that it is possible to improve the expected runtime to $O(n^5)$.

Algorithm 2

Algorithm 2 differs from Algorithm 1 only in the formulation of $C_+(v)$:

1. Choose one of the vertices z in distance 2 to v at random. Let w be the vertex that is connected to both v and z.

2. Generate a new graph $G_t^*(v, z)$ by replacing edge (v, w) by edge (v, z).

3. If $close(G^*_t(v,z),v) < close(G_t,v)$ then set $G_{t+1} = G^*_t(v,z)$, otherwise $G_{t+1} = G_t$.

As before, only vertices with an eccentricity higher than f_{min} will be allowed to change the current network. However, the decision whether a new edge will actually replace one of the old edges is based on a comparison of the closeness centrality in the prospective tree with the closeness centrality in the current tree. Note that this time the new value has to be strictly smaller than the previous value. Nonetheless, the change in the formulation of $C_+(v)$ from Algorithm 1 to 2 is actually quite small: Recall that the eccentricity of a vertex v is the *maximal* distance of v to any other vertex w while the closeness centrality is the *sum* over all distances from v to any other vertex w. Thus, in the latter case, we use much more information about the current tree but the calculation of eccentricity and closeness centrality values in a tree takes the same time, namely $O(n)$, with a simple variant of a single-source shortest path algorithm [100]. We will now prove the following theorem:

Theorem 6.2
In Algorithm 2, the expected runtime is bounded by $O(n^5)$.

Proof 6.2
The proof is based on the following two lemmata.

Lemma 6.2
In every connected tree T with $D(T) > f_{min}$ there is at least one vertex v with a second neighbor z such that $close(G^*_t(v,z),v) < close(G_t,v)$

Proof 6.3
Let G_t be a tree with $D(G_t) > f_{min}$. Let v_1 and v_2 denote two vertices with distance $d_t(v_1,v_2) = D(G_t)$. It is clear that these vertices have to be leaves, i.e., vertices with only one edge. Let w_1 and w_2 be the respective neighbor vertices of v_1 and v_2. The path $P(v_1,v_2)$ between v_1 and v_2 will contain w_1 and w_2:
$P(v_1,v_2) = \{v_1,w_1,z_1,\ldots,z_2,w_2,v_2\}$. Since v_1 and v_2 are in distance $D(G_t)$ of each other, all vertices adjacent to w_1 other than z_1 have to be leaves, otherwise v_1 and v_2 could not be in maximal distance within the graph. Let now vertex v_1 be chosen by the algorithm and try to insert an edge to z_1.

As can be seen in Fig. 6.6, the closeness centrality of vertex v_1 in $G^*_t(v_1,z_1)$ is given by:

$$close(G^*_t(v_1,z_1),v_1) = close(G_t,v_1) + |T_{w_1}| - |T_{z_1}|, \tag{6.9}$$

where $|T_X|$ denotes the number of vertices contained in subtree T_X, as depicted in Fig. 6.6. Subtree T_X denotes the subtree containing vertex X that emerges if edges (v_1,w_1) and (w_1,z_1) are removed from G_t. Equ. 6.9 states that the distance from v_1 to vertices from T_{w_1} is increased by one, and that the distance from v_1 to vertices from T_{z_1} is decreased by one in $G^*_t(v_1,z_1)$. If this new closeness centrality value is smaller than $close(G_t,v_1)$ then the new edge is built and the old edge is removed from the network and the case is proven. Let us now assume that the closeness centrality value in $G^*_t(v_1,z_1)$ is not smaller than the one in tree G_t. It follows, that

$$|T_{w_1}| \geq |T_{z_1}|. \tag{6.10}$$

We will now show that in this case any vertex v_2 with $d(v_1,v_2) = D(G_t)$ would decrease its closeness centrality value by replacing edge (v_2,w_2) with (v_2,z_2) where z_2 is a second neighbor of v_2 on the path to v_1 (Fig. 6.7).

$$close(G^*_t(v_2,z_2),v_2) = close(G_t,v_2) + |T_{w_2}| - |T_{z_2}|. \tag{6.11}$$

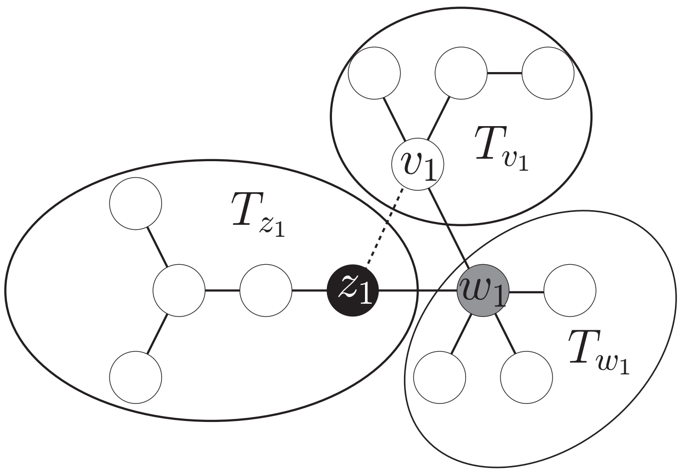


Fig. 6.6: Algorithm 2: In the network depicted above, vertex v_1 has been chosen at random, and afterwards it chooses one of its second neighbors at random, in this case z_1. w_1 is the neighbor of both. v_1 will replace edge (v_1, w_1) by (v_1, z_1) if $|T_{z_1}| > |T_{w_1}|$.

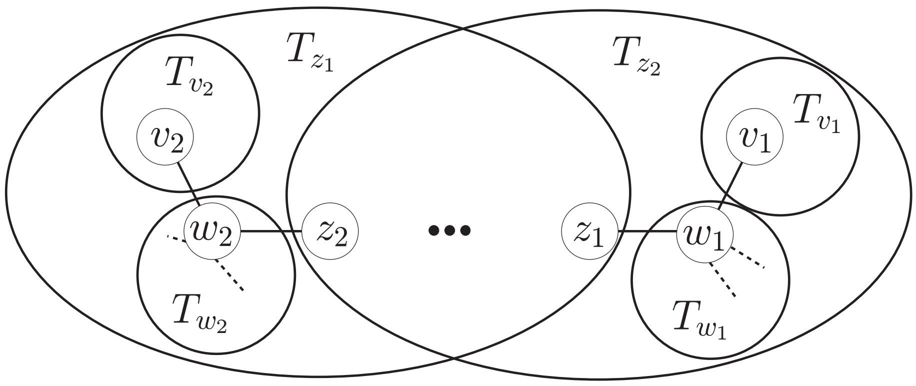


Fig. 6.7: Algorithm 2: Vertices v_1 and v_2 are maximally distant to each other, i.e., $d_t(v_1, v_2) = D(G_t)$. As sketched in Fig. 6.6, the tree can be partitioned into three different subtrees for each pair of vertices v_1, z_1 and v_2, z_2, namely $T_{v_1}, T_{w_1}, T_{z_1}$ and $T_{v_2}, T_{w_2}, T_{z_2}$.

As can be seen in Fig. 6.7, T_{z_2} can be expressed as follows:

$$|T_{z_2}| = |T_{z_1}| - (|T_{w_2}| + |T_{v_2}|) + |T_{w_1}| + |T_{v_1}|. \tag{6.12}$$

Inserting this in Equ. 6.11 and considering that $|T_{v_1}| = |T_{v_2}| = 1$, yields:

$$close(G_t^*(v_2, z_2), v_2) = close(G_t, v_2) + 2 \cdot |T_{w_2}| - (|T_{w_1}| + |T_{z_1}|) \tag{6.13}$$

With $|T_{w_1}| \geq |T_{z_1}| > |T_{w_2}|$ it follows that $close(G_t^*(v_2, z_2), v_2) < close(G_t, v_2)$.

We will now prove the following lemma regarding the non-increasing Wiener index of the evolving trees:

Lemma 6.3
Whenever G_{t+1} has emerged from G_t because of an edge replacement, then $W(G_{t+1}) < W(G_t)$.

Proof 6.4
Let G_{t+1} be evolved from G_t by the replacement of edge (v_1, w_1) with edge (v_1, z_1) according to Algorithm 2. Let again $T_{v_1}, T_{w_1}, T_{z_1}$ denote the subtrees as depicted in Fig. 6.6. It is clear that the distance of vertices has only changed for pairs of vertices x, y where x is either in T_{w_1} or T_{z_1} and y is in T_{v_1}. The distance of vertices from T_{w_1} to vertices in T_{v_1} (and vice versa) has increased by one, and the distance of vertices from T_{z_1} to vertices in T_{v_1} (and vice versa) has decreased by one. Thus, the Wiener index of the graph changes as follows:

$$W(G_{t+1}) = 2 \cdot (|T_{w_1}| - |T_{z_1}|) \cdot |T_{v_1}| + \sum_{x \in V} close(G_t, x). \tag{6.14}$$

Since edge (v_1, w_1) has been replaced by (v_1, z_1) it is clear that $|T_{w_1}| < |T_{z_1}|$ and the lemma is proven.

Combining both lemmata now yields Theorem 2: The first lemma states that there is at least one pair of vertices v, z such that $G_t^*(v, z)$ has a smaller closeness centrality for v. The expected number of time steps until this pair is chosen is bounded from above by $O(n^2)$. The maximal Wiener index is bounded from above by $O(n^3)$ when the tree is arranged as a chain. Since every edge replacement implies a decrease of the Wiener index of at least 1, the expected runtime of Algorithm 2 is thus bounded from above by $O(n^5)$.

The two instances of evolutionary algorithms for the evolution of networks given above serve very well to show how a small change in the adaptation rule can make the difference between having polynomial and exponential expected runtimes. So far, the algorithms suffer from the need to flood the whole network in order to evaluate the eccentricity or closeness centrality of a vertex. This provides a motivation to present a localized version of the algorithm in the next section.

6.4.1 Localized Version

The motivation for a localized version arises from the following question: Will a node in a real network, e.g. a city in a country, really evaluate how far away it is from the maximally distant city or evaluate the sum of the distances to all other cities? This is not very likely. It is much more likely that citizens will complain if some other city that is geographically near is far away in practical terms with respect to the infrastructure. That is, citizens will be unsatisfied with the infrastructure if their local environment is not easily accessible. A new street that replaces another one will thus be evaluated as to whether it reduces the distance to the cities within a given (geographical) radius and/or makes other cities accessible within a given radius.

This analogy prompted a third algorithm that is a localized variant of Algorithm 2. Again, we start with a tree on V, and one of the vertices is chosen at random in every time step. The vertex initiates a restricted broadcast of depth $f_{min} + 1$ and thus tests whether there are vertices in distance $f_{min} + 1$. If there are such vertices, then the vertex knows that its eccentricity is larger than f_{min}. Let again $N_i(v)$ denote the set of vertices in distance i to v, with $i > 0$. Then the following steps apply:

1. Choose one of the vertices z in distance 2 to v at random. Let w be the vertex that is connected to both v and z.
2. Count the number n_{min} of vertices within distance f_{min} and evaluate the *local closeness centrality* $close_{min}(G_t, v)$ defined as the sum of distances to all vertices within distance f_{min}.
3. Generate a new graph $G_t^*(v, z)$ by replacing edge (v, w) by edge (v, z).
4. The sum that defines the *local closeness centrality* can be partitioned into three different parts: First, there is the sum of distances to all vertices that were already accessible in f_{min} steps in G_t. This sum might be increased or decreased because of the new edge. Second, some of the formerly accessible vertices might no longer be accessible (within f_{min} steps). Since they were accessible before, their distance to v is now $f_{min} + 1$. $n^-(v, z)$ denotes the number of these vertices. Then, there is a third group of $n^+(v, z)$ vertices, namely those that are newly accessible within f_{min} steps. Thus, they had a distance of $f_{min} + 1$ before the new edge replaced the old. To make both closeness values comparable they have to be based on the same set of vertices. This can be accomplished by adding
$$close'_{min}(G_t, v) = n^+(v, z) * (f_{min} + 1) + close_{min}(G_t, v) \tag{6.15}$$
and
$$close'_{min}(G_t^*(v, z), v) = n^-(v, z) * (f_{min} + 1) + close_{min}(G_t^*(v, z), v) \tag{6.16}$$
This closeness is called the *balanced local closeness centrality.* If now the new closeness value is smaller than the old one, the new edge is kept, otherwise the old edge is restored.

Experimental results of this localized version are shown in comparison with the global version in Fig. 6.8. For the special case of $f_{min} = 2$ we will now show that the expected runtime until an attractor in $G(\mathcal{G})$ is hit is bounded by $O(n^4)$.

Lemma 6.4
For $f_{min} = 2$ the localized version of Algorithm 2 has an expected runtime of $O(n^4)$.

Proof 6.5
The proof proceeds in three steps:

1. Let G_t be a tree T with a diameter $D(T) > 2$ and let vertex v have an eccentricity $ecc(T, v) > 2$ and a second neighbor z where w is a common neighbor to v and z. We show that $close'_2(T, v) > close'_2(T(v, z), v)$ iff $deg(w) - 1 < deg(z)$.

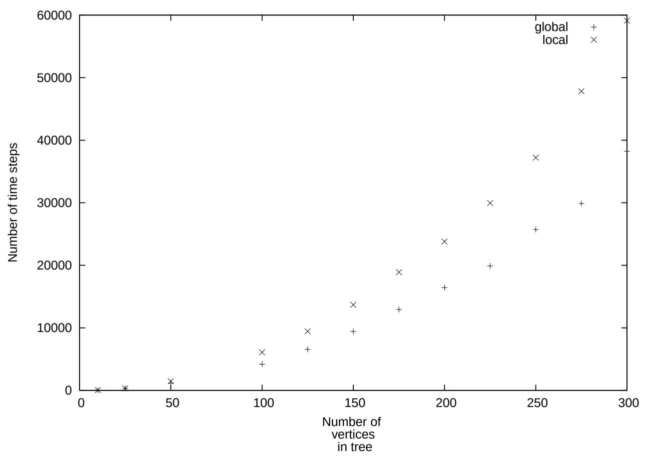


Fig. 6.8: The number of time steps needed by the localized version of Algorithm 2 is given in comparison with the number of time steps needed by Algorithm 2 with global information. f_{min} is set to 2 in both cases, and the starting tree is a chain of n vertices. It shows that the localized version is not much slower than the global version despite the fact that the information on which the decision is based is much smaller.

2. We show that every tree G_t with a diameter $D(G_t) > 2$ will have at least one vertex v with a second neighbor z such that a new edge is built according to the localized version of Algorithm 2.

3. If a new edge is built in time step $t+1$ then the following global objective function is strictly increased:
$$f(T) = \sum_{v \in V} \sum_{(v,w) \in E(T)} deg(w) \tag{6.17}$$
such that $f(G_t) < f(G_{t+1})$.

Let G_t be a tree with a diameter $D(G_t) > 2$ at time t. Its restricted closeness in T is described by:

$$close_2(G_t, v) = deg(v) + 2 \cdot \sum_{\substack{w' \in N_t(v) \\ w' \neq w}} (deg(w') - 1) + 2 \cdot (deg_t(w) - 1), \tag{6.18}$$

where $N_t(v)$ is defined as the set of direct neighbors of v.

Let now v be chosen first and then let v choose z as the second neighbor to which it builds a temporary edge. v's local closeness now corresponds to:

$$close_2 G_t^*(v,z), v) = 1 + 2 \sum_{\substack{w' \in N_t(v) \\ w' \neq w}} (deg(w') - 1) + 2 \cdot deg_t(z). \tag{6.19}$$

To build the *balanced local closeness centrality* it is now necessary to determine the union of the vertices that are counted by at least one of the values $close_2(G_t, v)$ or $close_2(G_t^*(v,z), v)$. The neighbors of z are visible for v in $G_t^*(v,z)$, but they were in distance 3 in G_t, thus we have to add the product of $3 \cdot (deg(z)$ to $close_2(G_t, v)$. On the other hand, the neighbors of w besides z are no longer visible to v in $T^*(v,z)$, thus we have to add the product of $3 \cdot (deg(w) - 2)$. It follows for $close'_2(T,v)$ and $close'_2(T^*(v,z),v)$:

$$\begin{aligned} close'_2(G_t,v) &= deg(v) && (6.20) \\ &+ 2 \sum_{\substack{w' \in N_t(v) \\ w' \neq w}} (deg(w')-1) + 2 \cdot (deg_t(w)-1) + 3 \cdot (deg_t(z)-1) && (6.21) \\ close'_2(G^*_t(v,z),v) &= deg(v) + 2 \sum_{\substack{w' \in N_t(v) \\ w' \neq w}} (deg(w')-1) + 2 \cdot deg_t(z) + 3 \cdot (deg_t(w)-1) && (6.22) \end{aligned}$$

Ad 1: It follows that the new closeness is strictly smaller than the old one only if:

$$\begin{aligned} close'_2(G_t,v) - close'_2(G^*_t(v,z),v) &> 0 \Leftrightarrow && (6.23) \\ deg_t(z) &> deg_t(w) - 1 && (6.24) \end{aligned}$$

Ad 2: Let v be a vertex with an eccentricity equal to $D(G_t)$, and let w denote its only neighbor. Since v has the maximal distance to some other vertex in the graph and $D(G_t) > 2$, w can only be connected to one non-leaf z. Let z' denote some neighbor of z other than w. This vertex has an eccentricity of at least 3 because its distance to v is 3. If now $deg_t(z) \leq deg_t(w) - 1)$ then the edge between v and z will not be built, but if z' were chosen and it chose w as a second neighbor it is clear that $deg_t(z) - 1 < deg_t(w)$ and thus the edge (t,w) would replace (t,z). In summary: While the diameter of the tree is still larger than 2, there is at least one vertex that can enhance its benefit locally.

Ad 3: In the last step we want to show that a local improvement can be translated into a global improvement. For this we analyze the behavior of the following global objective function:

$$f(G_t) = \sum_{v \in V} \sum_{(v,w) \in E_t} deg_t(w) \qquad (6.25)$$

Note that this function is minimal for a chain $C(n)$ of n vertices with a value of $f(C(n)) = 4n - 6$ and maximal for a star $S(n)$ of n vertices with a value of $f(S(n)) = n^2 - n - 1$.

We will now show that every replaced edge results in a strictly increased value of f. Let G_t be the first graph and G_{t+1} be a graph in which edge (v,z) replaces edge (v,w) according to the rules of the localized version of Algorithm 2. Since (v,w) has been replaced by edge (v,z) we know that

$$close'_2(G_t,v) < close'_2(G^*_t(v,z),v), \qquad (6.26)$$

and it follows that

$$deg_t(z) > deg(w) - 1. \qquad (6.27)$$

With this knowledge we can determine how $f(G_{t+1})$ has changed with respect to $f(G_t)$: all vertices that are neighbors of none of v, w, z contribute the same amount to $f(G_t)$ as to $f(G_{t+1})$. All vertices that are neighbors of v except w contribute the same amount as before. v's contribution has changed by $deg_t(z) - deg(w) - 1$. w's contribution is decreased by $deg(v)$, z's contribution has increased by $deg(v)$. All neighbors of w except v and z contribute one less than before, in total this adds up to $-deg(w) + 2$. All neighbors of z in G_t except w contribute one more in G_{t+1}, in total this adds up to $deg(z) - 1)$. This results in the following change of $f(G_{t+1})$ with respect $f(G_t)$:

$$f(G_{t+1}) - f(G_t) = 2 \cdot (deg_t(z) - deg(w)) + 1 \qquad (6.28)$$

In summary, there is at least one pair of vertices v, z in every tree G_t with a diameter $D(G_t) > 2$ such that v increases its local fitness, and this pair will be chosen after an expected number of time steps of $O(n^2)$. Every local improvement increases $f(G)$ by at least 3 and thus it takes at most $O(n^2)$ improvements until the network with maximal global fitness $f(G) = n^2 - n - 1$ has evolved. This results in an expected runtime that is bounded from above by $O(n^4)$. $\square$

6.4.2 Dynamic Version

Another question is how stable the algorithm is if the network is modified by external events, as indicated in the lower part of Fig. 6.3. In this case we wanted to test experimentally how added vertices disturb the process of reducing the diameter. To analyze the stability of the algorithm, we started with a chain of 25 vertices and $f_{min} = 2$. After a given number of time steps a vertex was added by an edge to a randomly chosen vertex already in the tree. The number of time steps needed to reduce the diameter to 2 was measured as well as the resulting number of vertices in the graph. Any run still incomplete by $10,000$ time steps was stopped. Table 6.1 shows the result of this variant. It seems that addition of vertices is tolerated quite well if it occurs only after every vertex has had the chance to get picked. In these cases the runtime was of course longer than a normal run with only 25 vertices, which takes about 44 time steps. On the other hand, it is still faster than a run with a chain of 50 vertices (no added vertices), that would normally take about 1100 time steps. When vertices were added after fewer than 25 time steps, almost none of the 35 runs would stop before $10,000$ time steps, but those that did still showed a good runtime. In summary, a dynamic version of Algorithm 2 is strongly dependent on the frequency with which vertices are added to the system. There seems to be a phase transition such that if vertices are added with a lower frequency the system behaves well and if vertices are added with a frequency higher than the critical value associated to the phase transition the system behaves totally different. How the relationship between this critical frequency and the size n of a chain can be described remains an open question.

Tab. 6.1: The tree started with 25 vertices. Every x time steps one vertex was added to the tree as described in the text. Average number of time steps needed to build a star and the average number of vertices were only evaluated for those runs that stopped before 10,000 time steps. Every experiment includes 35 runs.

#Time steps until vertex was added	Avg #time steps	Avg #vertices	#Trials with $> 10,000$ Time steps
-	44	25	0
37	486	37	0
35	591	41	0
32	540	41	0
30	567	43	1
27	770	52	3
25	649	50	2
22	812	61	22
20	606	55	26
17	665	63	33
14	-	-	35

In the next section we want to present another algorithm that uses a local evaluation function and shows very robust behavior and interesting properties, with possible applications to ad-hoc communication networks.

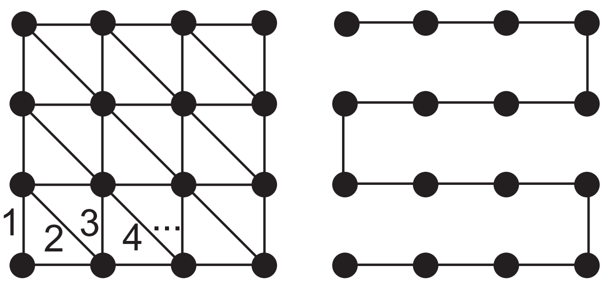


Fig. 6.9: On the left a grid is shown with a diameter of approximately $\sqrt{n}$. If the edges are removed as indicated by the numbers, a tree (right) will emerge with a diameter of $n-1$.

6.5 Reducing the Number of Edges in an Ad-Hoc Communication Network

It is expected that in the near future large *sensor networks* will collect data in environments that are not accessible by humans. A *sensor* is a small device able to measure some predefined property, e.g., the temperature or humidity in a room, with limited energy resources and limited broadcasting abilities for wireless communication. Sensors themselves are static, but the connections between them are often built on demand, and thus they belong to the class of *ad-hoc networks*. Normally, sensors can communicate with every other sensor that is within their transmission radius, but they can decide to ignore some of the messages that they receive and only process and forward those from sensors from a given *buddy list*. The network defined by edges from *buddy lists*, and generally any subgraph of a given graph, is also called an *overlay network* of that graph. If every sensor could communicate directly with every other sensor in the network this would require a very large transmission radius and thus too much energy [49]. To avoid this, a large number of different network structures have been proposed in which each vertex has a limited transmission radius and a limited degree in the resulting overlay network. Furthermore, these networks try to optimize a given objective function like the *area coverage* [253, 230] and/or a low average distance [158, 51]. The question of energy efficient broadcasting becomes even more complicated in mobile ad–hoc networks, e.g., when special devices in cars build ad-hoc communication networks to exchange data about traffic flow that is then submitted to the driver to prevent congestion. Also here, a (sometimes implicitly) reduced edge set is often the starting point on which efficient broadcasting protocols are based [112].

Sensor networks are often modeled as geometric k-next-neighborhood graphs G, where n vertices are placed in a metric space, e.g., Euclidian space, and every vertex is connected to its k geometrically nearest vertices. We have shown that any two neighbors of a vertex have a chance of approximately 58% of also being neighbors to each other [83]. Thus, every vertex participates in a large number of triangles. Since edges in ad-hoc networks can be used to route messages and most devices have only limited battery power, it might be reasonable to reduce the number of edges of a vertex in the communication network to minimize the number of routing requests. On the other hand, the diameter, i.e., the number of devices on the longest communication path, should not be increased too much because every device is a possible point of failure.

In some scenarios, it might be required that every vertex hold as few edges as possible, e.g. to reduce the number of communications needed in a one-to-all broadcast, without increasing the diameter too much. It seems natural that triangles of the sort described above provide a reasonable way to reduce the number of edges in the graph by the following algorithm.

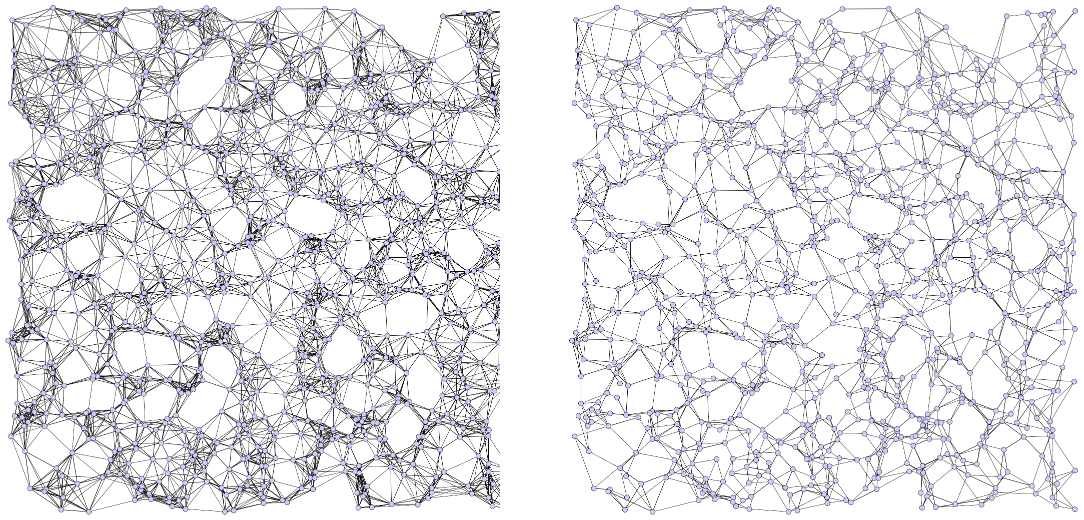


Fig. 6.10: The network on the left is a k-next–neighborhood graph with 1000 vertices, each connected to its ten geometrically next vertices. On the right, the same network is shown after the random removal of triangles as described in algorithm 3. The diameter of the network has increased from 24 to 30 while the average distance between vertices has only slightly increased from 10.4 to 12.5.

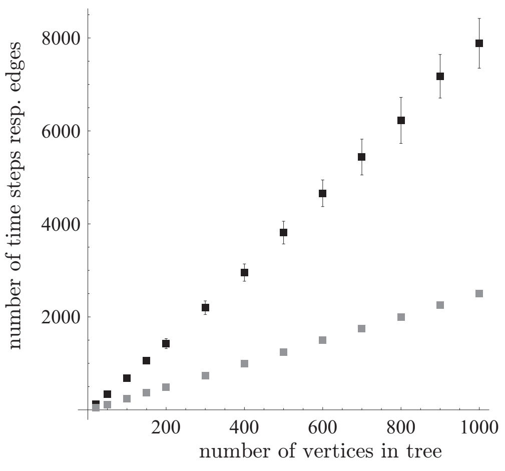
6.5.1 Algorithm 3

Start with G_0, a k-next-neighborhood graph in a metric room. In every time step choose one vertex v at random. The fitness $f(G, v)$ is given by the number of triangles v participates in and f_{min} is set to 0. $C_+(v)$ chooses one of the triangles at random and removes one of the two edges connected to v.

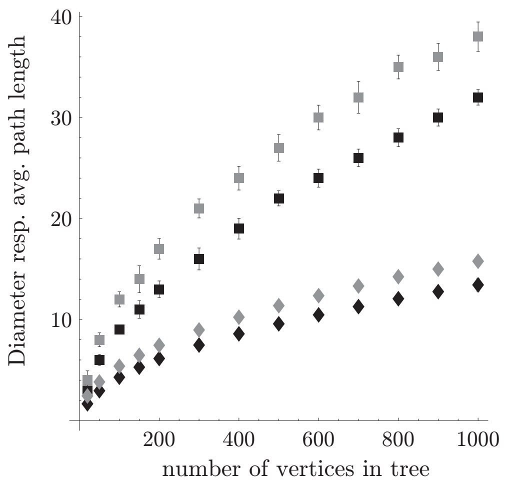
Note that this rule will keep a connected network connected. It is especially appealing because it uses only information from the immediate neighborhood of v. Furthermore, many of the operations can be conducted simultaneously without fearing any inconsistent information on the network. On the other hand, there are worst–case scenarios in which the diameter will increase from $O(\sqrt{n})$ to $\Theta(n)$ (Fig. 6.9) but these scenarios are very unlikely overall. Our empirical results have shown that this simple rule is nonetheless very robust and shows only slight increases of the original diameter. Furthermore, the number of time steps until nearly all vertices have no more triangles in which they participate is quite low (cf. Figs. 6.10 and 6.11).

Some ad-hoc communication networks like sensor networks are modeled as unit-disk networks where every vertex is connected to all other vertices within its unit-disk [138]. If the devices are uniformly distributed this approaches to a k-next-neighborhood graph. To guarantee connectedness, k might be quite large at the beginning, but for most communication protocols it is better to subsequently reduce the number of edges in the graph. Here, the algorithm depicted above might be an interesting protocol to reduce the number of edges without increasing the diameter too much. Of course, the algorithm can easily be altered to guarantee a certain diameter: the chosen vertex could additionally evaluate its current eccentricity and then decide whether it will remove an edge. Similarly, vertices with an eccentricity that is too high could also begin to build new triangles. Further research will have to show which kind of rules are most applicable to which specific situation.

The first algorithms presented so far were concerned with the optimization of a global network



(a)



(b)

Fig. 6.11: Empirical results for algorithm 3. All values have been obtained from 35 samples, each with n vertices, connected as a k-next-neighborhood graph with $k = 7$. Algorithm 3 was stopped after time step t' when in n subsequent steps no change of the edge set had occured; the runtime is thus given by $t = t' - n$. The left diagram shows the runtime (black boxes) of algorithm 3. For fixed k it is linear with n and shows only little deviation. Gray boxes indicate the number of edges after the process has stopped. The deviation is so little that it cannot be seen in this scaling. On average, the number of edges of the beginning graph $(n \cdot k)$ has been reduced to a third. The right diagram shows the diameter of the initial graph (black boxes) and after the process has stopped (gray boxes), and the corresponding average path lengths (black and gray diamonds). The empirical results show that the diameter and average path length is only slightly increased by the procedure despite the fact that approximately two thirds of the edges have been removed.

property in a steady environment. In the following section we will discuss whether another global property, namely the degree distribution, can be altered in reaction to environmental changes.

6.6 Self-Adapting Network Structures for Peer-to-Peer Networks in Random Failure and Attack Scenarios

It is a long-standing observation that a scale-free network structure [21] is more stable against random failures and more sensitive against directed attacks than a corresponding random graph [6]. It would thus be favorable to have a scale-free network structure in the first scenario, and a random graph structure in the second. To change the network structure in reaction to an external event like a random failure or an attack is possible in all those networks that are essentially an overlay network of the Internet such as various kinds of peer-to-peer networks [146]. These networks inflict nearly no cost on the building of an edge independent of its physical length and allow the building of new edges quickly.

It could be shown that many peer-to-peer networks are small-worlds [242] or scale-free networks to start with, due to the nature of their attaching rules [206, 209]. But so far no protocol is known that lets the network's degree distribution change in response to the type of external event, in this case random failures vs. directed attacks against the high-degree vertices or *hubs*. Since most of these communication networks are not centrally managed, this adaption must occur in a decentralized manner. Since external events play a major role in this setting, we will first adapt the above given evolutionary network model and then develop a set of local changing rules that allow for a self-organized shifting of the degree distribution that stabilizes the network's structure in both scenarios, under attack or random failures.

As stated in [6], a random network from the $G(n,p)$ model is less stable than a scale-free network in the case of random failures, but is on the other hand more stable in the case of attacks. The differences between these two network models are mainly based on the difference in their degree distribution: In a random graph every vertex has expectedly the same degree, and every vertex will contribute the same *stability* to the network. In the scale-free architecture, a high degree vertex that is attacked will sever the networks connectivity heavily, but on the other hand it is unlikely that a random failure will hit exactly this vertex.

Many of our modern communication networks, e.g., the Internet, the WWW, peer-to-peer networks, sensor networks, and other multi-hop communication networks, are prone to random failures and would thus benefit from a scale-free or at least a right-skewed degree distribution. Indeed, some of the protocols of peer-to-peer networks result in a scale-free network structure [206, 209]. On the other hand, these networks might suffer from attacks from time to time and in these situations it would be helpful to switch to a network structure in which the degree distribution is normally distributed or at least shows a very small deviation from the average. Since most real-world networks are not centrally organized it is not possible to detect an attack situation from a bird's eye view and change the network's structure in a coordinated manner. One possibility is that every single vertex tries to detect an attack locally and subsequently changes its local neighborhood by connecting to random vertices in the network, as proposed by [122]. This approach is difficult: if every vertex can only see its local neighborhood it will detect an attack only if a large proportion of the network is already attacked. Here, we propose a second possibility, namely a simple reaction scheme that is able to drive the network's structure into the best possible structure independent of whether it is in an attack or random failure scenario.

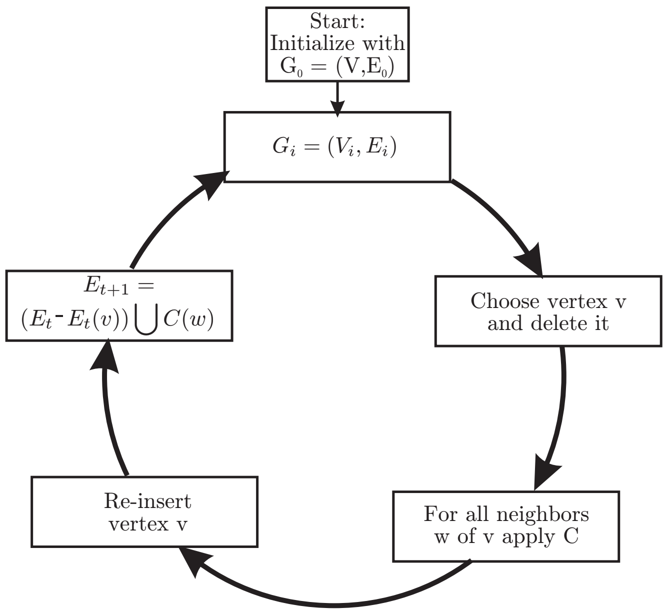

Start:
Initialize with
$G_0 = (V,E_0)$
$G_i = (V_i, E_i)$
Choose vertex v and delete it
For all neighbors w of v apply C
Re-insert vertex v
$E_{t+1} = (E_t - E_t(v)) \bigcup C(w)$

Fig. 6.12: A evolutionary network model similar to the one introduced in Fig. 6.3 that sketches the cycle of attacks, or random failures that remove a vertex from the network, the reaction of its neighbors to this event, and the re-entry of the removed vertex into the network.

6.6.1 An Evolutionary Network Model for Attack and Random Failure Scenarios

The evolutionary network model is a variant of the general evolutionary network model described in Fig. 6.3. This variant is defined as follows: Given a graph $G_i = (V_i, E_i)$ we delete one vertex in every time step $i + 1$. In an *attack scenario* the removed vertex is one of the vertices with highest degree, in a *random failure scenario* it is chosen uniformly at random. This removal is modeled as an *external event* in the model of Fig. 6.3. We assume that both kinds of removals will leave the vertex physically intact and thus we allow it to re-connect to the network by some re-entry rule R. The latter will keep the number of vertices in the system constant and thus allows for easy comparison of the emerging network structures.

We define $f(G_t, v)$ to be the degree of vertex v in G_t, and $f_{\min}$ to be $f(G_{t-1}, v)$, i.e., the degree it had in the previous time step. In every time step, i.e., after every removal, every vertex is allowed to check whether its degree has decreased in comparison with the last time step, and if so, it is allowed to rebuild its neighborhood according to some changing rule C. The model is sketched in Fig. 6.12.

The question is now how the rules R and C should be designed to drive the network towards a normal degree distribution in the case of attacks and in the case of random failures towards a scale-free degree distribution or at least towards a degree distribution with a long tail to the right. Our second goal is to achieve this while keeping the number of edges constant by replacing only those edges that are lost due to the removal of v. To do so, the removed vertex is allowed to re-entry the network with half the edges it had before the removal; this re-enter rule does not need any global information since the degree of a vertex is local information that can be stored by each of the participants and used in the re-entry process. The other half of the edges can then be used by the old neighbors to stabilize the network structure as defined by the changing rule C. The following properties of C are thus desired:

1. C is a local rule, i.e., only information about vertices within a short distance is used.
2. C holds the number of edges (expectedly) constant.
3. C drives a degree distribution towards one with a long right tail under random failures and towards a degree distribution centered sharply around the average degree under attacks.

It is well known that the *preferential attachment* rule will lead to a scale-free network [21]. Unfortunately, such a protocol is only applicable when a new vertex attaches to the network. However, C should show the same *the–rich–get–richer behavior* inducing a right-skewed degree distribution as a preferential attachment rule. The basic construction of C is that a neighbor w of the deleted vertex v will build a new edge with probability 0.5. Since the removed vertex v will re-enter the network with $deg(v)/2$ and the general construction of C will expectedly lead to $deg(v)/2$ edges, the combination of R and C holds the number of edges expectedly constant. To make C local, the new edge of w is built to one of w's second neighbors z in $N_2(w)$, as was the case in the algorithms described before. Since we want to achieve a preferential attachment–like behavior, we define a general form of the probability $P_i(z,w)$ of choosing a vertex z from $N_2(w)$ that depends on the i-th power of the degree of z:

$$P_i(z,w) = \frac{deg(z)^i}{\sum\limits_{z' \in N_2(w)} deg(z')^i}. \tag{6.29}$$

i controls how much the degree of the vertex will influence its probability of being chosen. If i is set to 1, this general form results in the normal preferential attachment probability as described in [21], but restricted to the neighbors in distance 2. If i is set to 0, every neighbor in distance 2 has the same chance of being chosen. Note that the denominator has to take the given form in order to make $P_+(z,w)$ a probability distribution. Note also that by the same argument it does not make any sense to give vertex z a probability of $k \cdot deg(z)$ since such a constant would not alter the *relative probability* that z is chosen, i.e., if z and z' are in $N_2(w)$ the relative probability of z being chosen in relation to z' is still given by $deg(z)^i/deg(z')^i$ since k will be canceled out.

With this general form of $P_i(z,w)$ we can now define our model by defining the re-entry rule R and a set of changing rules C_i:

Definition of the Changing and Re-Entry Rules
Let v be the vertex that is removed from the graph. For every vertex w in $N(v)$ build a new edge with probability 0.5 to some vertex z in $N_2(w)$ chosen with probability $P_i(z,w)$ as defined above. This describes the general form of changing rule C_i. If by chance the neighbor is isolated and does not have any neighbors in distance 2 it will be reconnected at random to the network. Re-insert vertex v with a randomly chosen set of $deg(v)/2$ vertices, where $deg(v)$ denotes the degree of v before its removal.

We have already argued that the combination of R and any C_i will keep the number of edges in the network expectedly constant and it is obvious that C_i considers only local information. We will now have to argue that these rules are able to shift the degree distribution into a shape that is appropriate for the given situation. If we start with a right-skewed degree distribution and the network is attacked, the desired shift to the left towards a normal distribution is mainly organized by the nature of the attack itself. Since the attacker does not allow any vertex to have a degree much higher than average, this behavior introduces a right bound to the degree distribution. Thus, a few attacks will drive the right-skewed degree distribution towards a more balanced, normal distribution.

We will now show that in a random failure scenario a right-skewed degree distribution will emerge because of the changing rule. To do so we will estimate $E[\Delta(deg(w))]$, i.e., the *expected change in the degree of* w. We assume that there is no correlation between the degree of neighbors, i.e., that the so-called *assortativity* of the graph is 0 [177] and thus the graph is *disassortative*. Newman has shown that the preferential attachment and the random graph model are disassortative. Thus, by starting with a scale-free or random graph in G_0 and by re-inserting vertices with a random attachment rule, the resulting graph will still be disassortative. We will now show that neither of the rules C_i causes assortativity of the network. Since G_0 does not show any associativity we also know that the degree of vertices in $N_2(w)$ is not correlated with the degree of w. Even if the new target vertex z is chosen in proportion to its degree (or any higher power of it) this does not introduce any assortativity since the degree of w is chosen uniformly at random from the set of all vertices.

With this assumption we can now determine the probability that any vertex w will gain or lose a new edge in a random failure scenario by determining $E[\Delta(deg(w))]$. Let $P_-(w)$ denote the probability that w loses an edge in the following step, and let $P_+(w)$ denote the probability that w gains an edge, then $E[\Delta(deg(w))] = P_+(w) - P_-(w)$. $P_-(w)$ is given by $0.5 \cdot deg(w)/n$ since there are $deg(w)/n$ ways to remove a direct neighbor of w and with probability 0.5 this edge will not be replaced. $P_+(w)$ is harder to determine. First of all, one of the second neighbors of w has to lose its direct neighbor. The probability for this is

$$\sum_{z \in N_2(w)} \frac{deg(z)}{n} \tag{6.30}$$

If one of the vertices $z \in N_2(w)$ loses a neighbor, it will build a new edge with probability 0.5 and choose w from its second neighbors with probability $P_i(z, w)$. Thus, $P_+(w)$ is given by:

$$P_+(w) = \sum_{z \in N_2(w)} 0.5 \frac{deg(z)}{n} \frac{deg(w)^i}{\sum_{z' \in N_2(z)} deg(z')^i} \tag{6.31}$$

Let now $deg_\emptyset$ denote the *average degree* $2m/n$ in the graph. We will restrict ourselves to the case of $i \in \{0, 1\}$ since here we can approximate $\sum_{z' \in N_2(z)} deg(z')^i$ by

$$\sum_{z' \in N_2(z)} deg(z')^i \simeq deg(z) \cdot deg_\emptyset^{i+1} \tag{6.32}$$

With this we can simplify $P_+(v)$ to:

$$\begin{aligned} P_+(v) &= \sum_{z \in N_2(w)} 0.5 \frac{deg(w)^i}{n \cdot deg_\emptyset^{i+1}} && (6.33)\\ &= 0.5 \frac{|N_2(w)| deg(w)^i}{n \cdot deg_\emptyset^{i+1}} && (6.34)\\ &= 0.5 \frac{deg(w)^{i+1}}{n \cdot deg_\emptyset^{i}}. && (6.35) \end{aligned}$$

The last equation is again derived by the approximation $N_2(w) \simeq deg(w) \cdot deg_\emptyset$.

We will now discuss the special case where $i = 0$. Here, the probability of gaining or loosing an edge is the same, namely 0.5 $deg(w)/n$, implying that $E[\Delta(deg(w))] = 0$. This means that the degree of each vertex makes a random walk with the same probability of increasing or decreasing. Since the degree of a vertex cannot be below 0 this random walk is bounded from the left and will thus create a right-skewed degree distribution. Note however that the right-shift will be very slow. After k steps of a normal random walk, the standard deviation is $\sqrt{k}$. Since every vertex will do one step in the random walk only every $1/n$ steps expectedly, the standard deviation will only grow with $\sqrt{(}k/n)$.

We will now examine changing rule C_1. For C_i the expected difference in the degree of w is given by:

$$E[\Delta(deg(w))] = 0.5\frac{deg(w)}{n}\left(\frac{deg(w)}{deg_{\emptyset}} - 1\right) \tag{6.36}$$

This is an interesting equation that shows that a vertex with a degree higher than average is likely to gain new edges while vertices with lower than average degree tend to lose edges. It is clear that this rule should easily be able to shift the degree distribution of the network to the right.

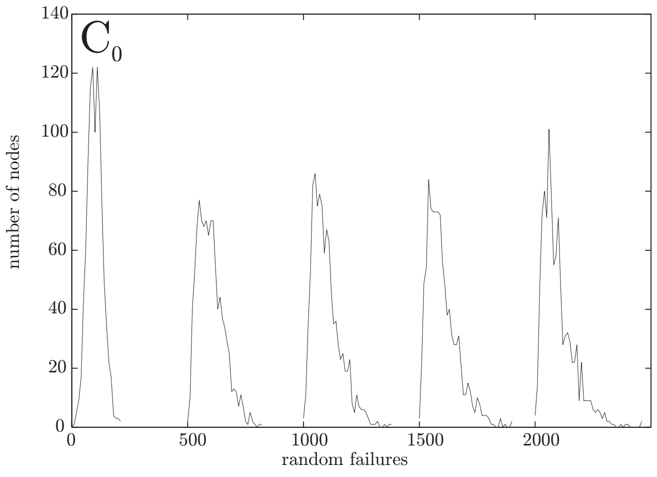
In the following section we will show some empirical results for C_0 and C_1 and show which of the rules is likely to be applicable to a real-world scenario.

6.6.2 Results

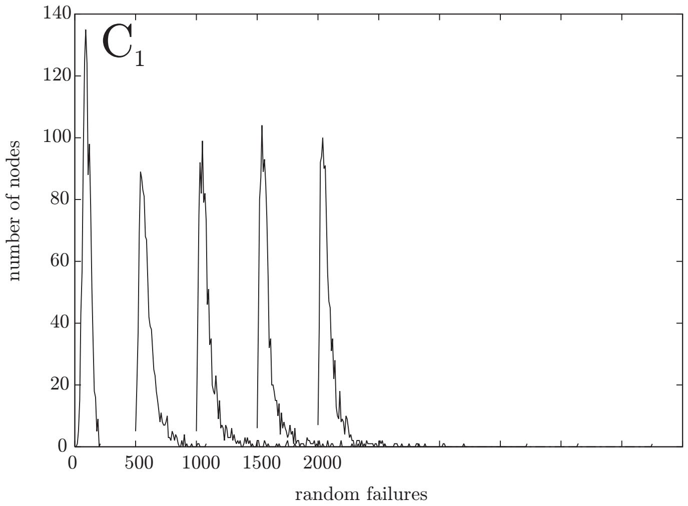
Fig. 6.13 shows that the rules given above are indeed able to change a normal degree distribution (given by a random graph) to a right-skewed degree distribution under random failures, and that the second changing rule in Fig. 6.13(b) changes the degree distribution more quickly into a strongly right-skewed degree distribution.

Note that none of the changing rules can guarantee connectivity, i.e., it is possible that small subgraphs are disconnected from the biggest connected component. With the random attachment insertion rule it is very unlikely that large portions of the network will stay isolated for a long time because every re-inserted vertex has a small chance of picking only vertices from the same component. Fig. 6.14(a) shows that the percentage of vertices in the biggest connected component does not fall below 99% with changing rule C_0, and not below 98% for C_1 in the random failure scenario shown in Fig. 6.13. The evolution of the average path length in this scenario is shown in Fig. 6.14(b). Based on these figures, rule C_0 is slightly better in keeping the graph together, but the average path length increases by approx. 4%, whereas with rule C_1 there are some more isolated vertices, but the average path length actually **decreases** by approx. 4%.

Fig. 6.15 shows the resulting degree distributions of a graph that suffers 5 runs composed of one attack and one random failure series, starting with a scale-free graph or random graph with $n = 1,000$ and $m = 5,000$. Every series contains $1,000$ subsequent removals of each type. It is clearly visible that the degree distribution changes from a sharp degree distribution with a low deviation to a right-skewed distribution and vice versa. Of course, the degree distribution itself is just an indicator of the resulting network's stability. We measure the stability of a network by removing 5% of the vertices without any adjustment of the network and compute the resulting average path length. Again, these 5% are either removed in an attack scenario or in a random failure scenario. This stability measure is denoted by $S_5(G, T)$, where G denotes the type of graph (RG for *random graph*, SF for *scale-free*), and T, the type of removal (A for *attack*, R for *random failure*). For a scale-free graph with $n = 1000$ and $m = 5000$, the empirically observed $S_5(SF, A)$ is 2.9 and $S_5(SF, R)$ is 3.8; for a random graph with $n = 1000$ and $m = 5000$, the empirically observed $S_5(RG, A)$ is 3.4 and the $S_5(RG, R)$ is 3.2.

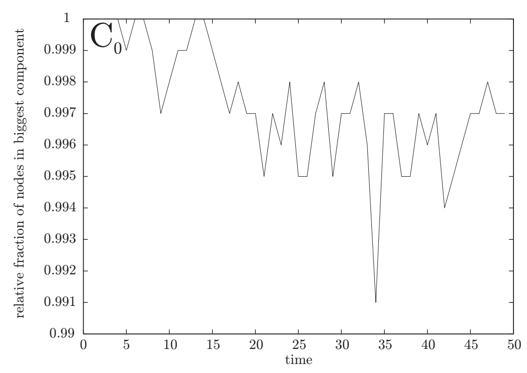


(a)

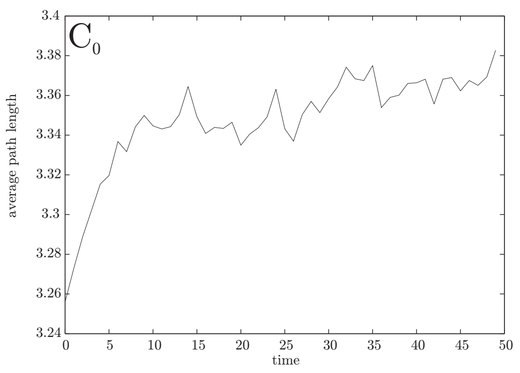


(b)

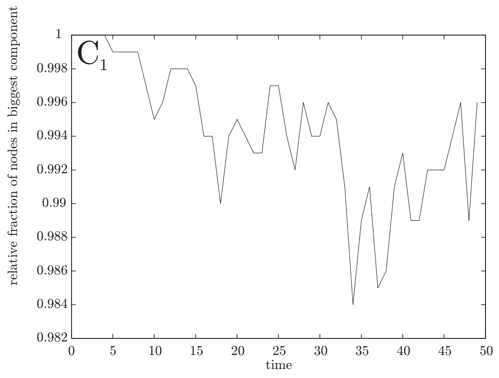
Fig. 6.13: Evolution of the degree distribution of random graphs with $n = 1000$ and $m = 5000$ after 2000 rounds of *attacks*, plotted every 500 removals. (a): changing rule C_0; (b): changing rule C_1. Note that the scales are different because the distributions shown below have a long tail.



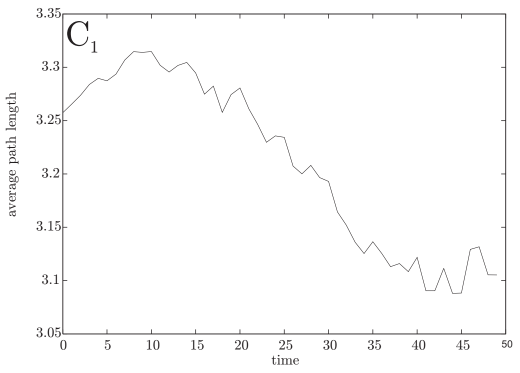
(a) C_1



(b) C_1



(c) C_2



(d) C_2

Fig. 6.14: (a): The percentage of vertices in the biggest connected component, and (b) the average path length in the evolution of the graph shown in Fig. 6.13(a), i.e., under changing rule C_0;(c): The percentage of vertices in the biggest connected component, and (d) the average path length in the evolution of the graph shown in Fig. 6.13(b), i.e., under changing rule C_1.

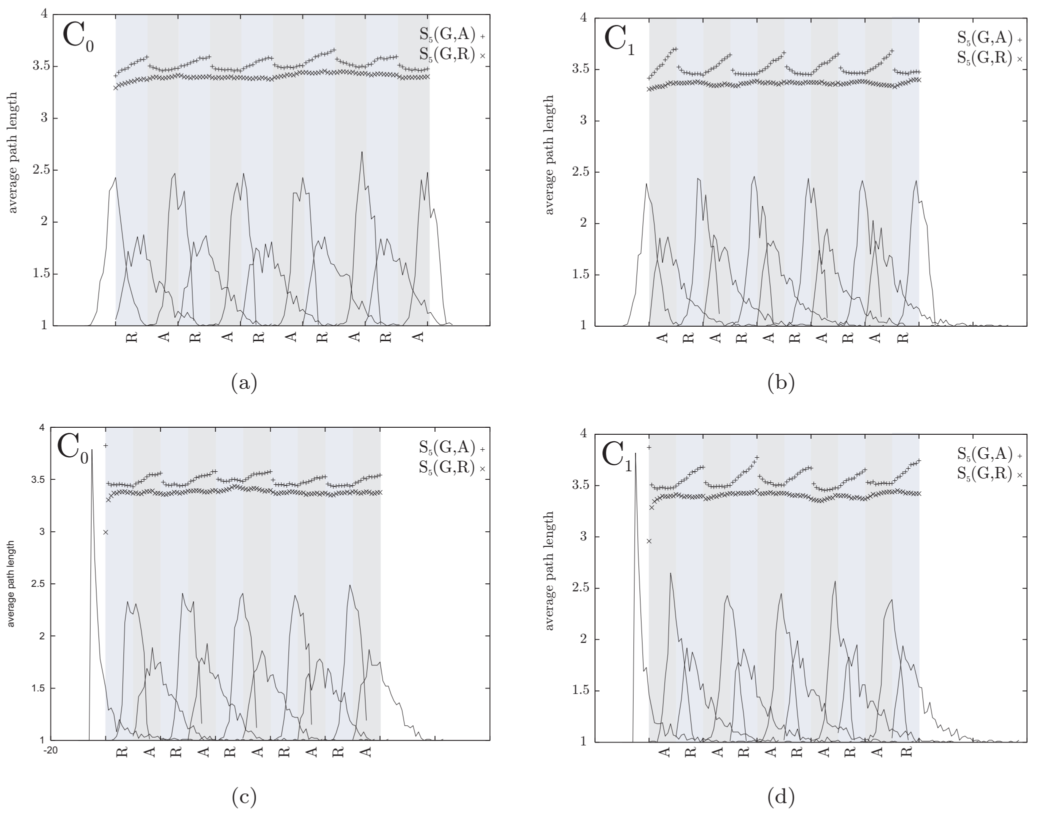


Fig. 6.15: 5 runs, each of $1,000$ attacks and $1,000$ random failures, indicated by the letters A and R. Note that for a random graph we start with a random failure scenario, and for a scale-free graph we start with an attack scenario. The y-axis shows $S_5(G, A)$ and $S_5(G, R)$ of the resulting graphs along the evolution. The degree distributions are shown out of scale to emphasize the correlation between a high $S_5(G, A)$ value and a right-skewed degree distribution. All starting graphs have 1000 nodes and 5000 edges. (a): $G_0 = RG$, $C = C_0$. (b): $G_0 = SF$, $C = C_0$. (c): $G_0 = RG$, $C = C_1$ (d): $G_0 = SF$, $C = C_1$.

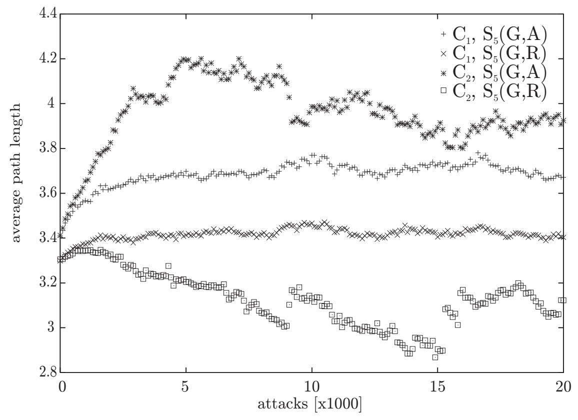


Fig. 6.16: Evolution of $S_5(G, A)$ and $S_5(G, R)$ in a long random failure scenario with $20,000$ events. The starting graph is a random graph with 1000 vertices and 5000 vertices.

As can be seen in Fig. 6.15, independent of the structure of the starting graph, both changing rules C_0 and C_1 quickly result in network structures that show a good stability in the case of attacks, i.e., $S_5(G, A)$ is around 3.5 after the first 200 attacks which is comparable to $S_5(RG, A)$ in a pure random graph. Unfortunately, the emerging right-skewed degree distribution after the first 1000 random failures is not strong enough to decrease $S_5(G, R)$ below 3.4 in the case of C_0 and below 3.3 in the case of C_1. This value is only comparable with the stability of a pure random graph but not with that of a pure scale-free graph. Of course, normally one can assume that random failure phases will be much longer than attack phases. Fig. 6.16 shows $S_5(G, A)$ and $S_5(G, R)$ in the evolution of a random graph with $1,000$ vertices and $5,000$ edges with $20,000$ random failures. It is clear to see that also a long random failure phase does not change the behavior of the network generated by changing rule C_0, and that the resulting network is less stable than a corresponding random graph with $S_5(G, A) \simeq 3.6$ and $S_5(G, R) \simeq 3.4$. However, changing rule C_2 is able to generate a network whose random failure stability $S_5(G, R)$ fluctuates around 3, with a minimum of 2.8.

It follows from these empirical results that changing rule C_0 is too weak to stabilize networks in a random failure scenario if they do not already have a right-skewed degree distribution. But we could show that changing rule C_1 quickly stabilizes an attacked network if the vertices have enough time to rewire the lost edges locally. Furthermore, as long as the network suffers only random failures, the protocol re-establishes a right-skewed degree distribution that then stabilizes the network against random failures.

In summary we have shown that a scale-free network structure is only more vulnerable to attacks if the network is not allowed to react by building new edges to compensate for the ones that were deleted. We have shown that simple, local rules can be implemented that stabilize the network's structure at nearly no communication costs. Furthermore, since these rules are local, they stabilize the local network structure such that subsequent attacks will not enlarge the distance between nearby vertices but only those between distant vertices. Since in most communication networks local communication is much more probable than long-distance communication the newly introduced stability measure *coherence* shows that our network protocol stabilizes networks in a suitable way against attacks and random failures. Of course, these rules have only been shown to work in the artificial setting described above, thus the future will have to show whether they can help to stabilize the communication networks we work with everyday.

6.7 Summary

In this chapter we have shown by a proof of concept that it is possible to guide self-organized networks in an efficient manner to build certain global network structures, although the evolution is controlled by the myopic vertices of the evolving network that can only make local decisions.

As sketched in the introduction of this chapter there are already quite a handful of models for dynamic and evolving graphs. Why do we think that adding another is helpful? Our model is designed explicitly to find those local changing rules that can be implemented in real-world technical communication networks that for various reasons have to be decentrally organized. In peer-to-peer networks it is a political decision, in sensor networks it is a question of investment into the system. Here we want to review briefly the main differences between our model and other models:

1. Every vertex will evaluate only its own situation within the graph and will only try to improve its own situation at any time step (**Egoism**). This is close to the game theoretic approach, but our agents are myopic, not totally rational, and they do not have global knowledge of the utility function of other vertices. Our model is thus **restricted rational in time and space**.
2. The algorithm does not aim for minimality regarding the evaluation function $f(G, v)$ but only that it is low enough, i.e.,
$\sum_{v \in V} f(G, v) \leq n \cdot f_{min}$ (**Satisfying topology**). Regarding this might protect a network from becoming overly adapted to a given environment.
3. A third aspect is that the evaluation of the network is decentralized, which can, in many cases, reduce the number of messages to be sent over the network (**Dezentralized Evaluation**). Note that we define a calculation to be *decentralized* if no vertex needs to know the whole adjacency matrix for its evaluation but only its own neighborhood. Thus, all functions that can be evaluated with a memory space of $O(n)$ and some communication protocol are suitable. It seems necessary to define the term *decentralized* very broadly because even a test for connectedness in a network needs a flooding protocol in which all vertices participate. Of course, the implementation of any rule will be the more interesting in a practical sense the more 'local' it is, i.e., the less communication has to take place in order to evaluate it (s. Sec. 6.5).

Note that, for example, algorithms 1 and 2 can be perfectly modeled by a $1 + 1$ evolutionary algorithm ($1 + 1$ EA): this is a genetic algorithm with memory, in which the population consists of one graph and one offspring that is a copy of the father, mutated as determined by some changing rule. In a $1 + 1$ EA the better graph of the two is then selected for the next round. Although the general framework given above allows to construct models that could also be modeled as $(1+1)EA$s, it is much more general than the family of algorithms modeled by the latter. The changing rule for switching network structure in random failure/attack scenarios is one of the algorithms that can be modeled with the general framework, but does not belong to the class of $1 + 1$ EAs. However, the model can be used to explore more complex and individualized changing rules in empirical simulations, but to deduce some analytical results it seems necessary to stick to simpler rules first.

7. SUMMARY

With the dawn of complex systems science, the world has heard a number of claims of a *new science*, in e.g.: "Six Degrees - The Science of a connected Age" by Watts [241], "Linked - The New Science of Networks" by Barabási [18], or "A new kind of Science" by Wolfram [250]. It is hard to judge whether a totally new kind of science is needed to understand complex systems, but certainly a new kind of *thinking* is needed as a human being in a globalized social network:

> Wir sind uns selbst vorausgeeilt, nun haben wir Mühe, uns wieder einzuholen. Auch unsere Denkstrukturen hinken hinterher. Sie sind geprägt von Epochen, als die Welt noch lokal und übersichtlich war oder es dem Einzelnen zumindest so schien. Bewältigung von Komplexität gehörte nicht zu den überlebenswichtigen Erfordernissen des Alltags. Heute jedoch sind wir ohne diese Fähigkeit verloren.[1] ([94], p. 142)

One of the approaches for dealing with complexity is to reduce a complex system to a complex network by focusing on one type of agent and one relationship, and to analyze these networks with respect to their structures. We have reviewed many structural properties that were found by this approach, but of course, many real–world networks and their structural properties still puzzle us. One of these structural properties, which we can observe, but whose evolution we do not yet know, is the tree distance distribution and its different characteristics that we have found in real–world networks. We think that it is an important structural property that simultaneously describes the locality and clustering of a network, and we hope that if we understand more about network generating systems and their network generating processes, we will better understand the significance of the backbone of a network.

Our belief in this approach can be best explained by the example of cellular automata, of which we give some examples in Fig. 7.1. Looking at the $2D$ picture of cellular automata it seems that some of them are simple to describe (Figs. 7.1(a) and 7.1(b)), and some are very complex to describe (Figs. 7.1(c) and 7.1(d)). But basically, this is an artifact of our perception because we try to find a 2-dimensional pattern in these pictures. All of them are produced by basically the same simple process: every horizontal line represents the state of the system at some time, starting with a white line with one black box in the middle of the line. The state at time $i+1$ is computed from the state in i by looking at each cell and its two neighbors[2]. A rule assigns the resulting cell color to each of the eight possible configurations, and this constitutes the color of all cells in the next state[3]. Looking at the pictures from this perspective, it is still interesting that the same simple process can yield such different results when viewed as $2D$-pictures, but now all of them have the same complexity because their generating process is the same.

Our idea is that the same will happen with complex networks once we find the best perspective, and our conjecture is that this perspective is the one which focuses the network generating process.

[1] We are running in front of ourselves and now it is a pain to catch up. The way we think seems to lag behind. It has been molded by eras in which the world was local and easy to understand, at least seemingly. Mastering complexity did not rank among the capabilities most essential for survival. Today, however, we are lost without

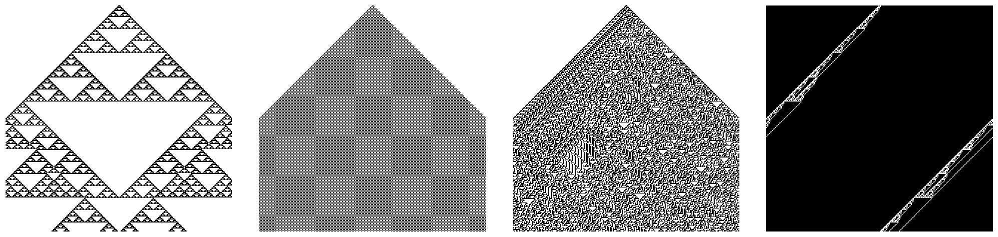

(a) Rule 126 (b) Rule 186 (c) Rule 30 (d) Rule 186

Fig. 7.1: Different 1-dimensional cellular automata in their timely evolution in the y-axis. The rule number gives the assignment of each of the 8 possible configurations of three cells and the state of the middle cell in the next time step [250].

Indeed, in the meantime some first analyses of dynamic network data have been published [254, 155, 189] that may help us to find out more about those network generating processes that build real–world networks. One main point of this perspective is that a complex network is always the result of its evolution in time and that thus every static snapshot holds information about the whole process which may make it difficult to find the simple pattern of how it evolved. A second cornerstone of this perspective is that the *agency* that governs the evolution of a network is important, because we have shown that building networks between selfish agents can make the outcome very sensitive to the applied rule. And of course, a network in which the single vertices build the agency of evolution should not be modeled by a network generating process that requires a central authority and vice versa.

Starting with the question of what a small–world network is we have explored the term *locality* in chapter 4 and our intuition is that it constitutes the most important network generating process. However, as we have shown in the case of the clustering coefficient, and as we have indicated for the (k, l)-locality measure of Chung and Lu and the backbone distance distribution, this seems to be the process that is most difficult to measure in the resulting network. We think that a steep backbone distribution is caused by a network generating process that prefers to build local edges, but to show a strong correlation between the structure and the process was not possible so far. We also see a connection of the backbone distribution to clustering because a steep backbone distribution seems to indicate that the network is clustered, but also here, a strong correlation between the concepts is still missing. Our intuition is that analyzing the network generating process in a network generating system might not be amenable in most cases by an analysis of the complex network alone, but rather will need the traditional tools of the science in whose realm they fall, i.e., sociology in social networks, or biological methods in biological networks. Still, we think that there are many open problems that can be addressed by methods from graph theory and other tools used in computer science:

1. In our small–world model we have given a simple analysis of the needed degree such that a unit-disk graph will be connected with high probability, under the assumption that the vertices are distributed uniformly at random. Our analysis yielded a minimal degree of

it.

[2] The cells are considered to be cyclic, i.e., the last cell is neighbored to the first one.

[3] States are encoded by binary numbers. A white cell is identified by 0, a black cell by 1. The state 'all–white' is thus 000, black–white–white is 100, i.e., 4. The middle cell in such a configuration can either be black or white in the next time step, thus there are $2^8 = 256$ different rules. A rule can now be encoded by a single binary number with eight digits, where the last digit gives the assignment to 000, the second to last gives the assignment to 001, and so on.

$5.68 \log n$, a bound that was improved by Xue and Kumar to $5.1774 \log n$, but they conjecture the threshold should be $\log n$ [252]. This question is still open, to our knowledge.

2. We have shown in chapter 5 how the number of random edges in a hybrid graph can be bound. So far, we have only used simple statistical methods to bound this number. The main problem is that the second method concentrates on the question of how high $\rho*$ could be in every single tree distance class, and thus it overestimates the number of random edges in the graph. We are sure that there are more elaborated methods to determine this bound, and thus state this as the second open problem.

3. The question is also whether there are more network generating processes that can be detected either in the resulting static network or in the dynamic changes of the network. Yook et al. have shown that in the dynamic changes of the Internet's structure, indeed those vertices tend to acquire more new edges that already have a high degree, i.e., they found that newly attached vertices seem to follow the preferential attachment model proposed by Barabási and Albert [254, 21]. Can such a process still be detected if it is mixed with another one, e.g., a random graph process, or an assortative one? In this area, we think that also impossibility theorems would be very interesting. In our eyes, a good candidate for such an impossibility theorem is locality as a network generating principle, but it maybe that our definition of locality is too general to allow for such a proof. In general, we think, that the question of what kind of network generating processes can be detected in the resulting network is of great importance and could lead to a new understanding of the structures we find in real–world networks.

4. As indicated in chapter 5, the runtime of some algorithms depend on $Q(T)$ and are thus more efficient on those real–world networks where $Q(T)$ is in $O(m \log n)$ or even smaller. We see several open problems here:

   (a) We were astonished how simple it was for many real–world networks to find very good spanning trees. Our intuition is that if a network has a steep tree distance distribution this can maybe exploited to find a near–optimal one by an approximation algorithm.

   (b) As we have reviewed, the runtime of some algorithms is depending on $Q(T)$ of a given spanning tree. So far, only those applications were looked at that are directly based on the question of finding a spanning tree with a low $Q(T)$. We assume that also other problems might be more efficiently solvable in networks with a steep tree distance distribution, e.g., the vertex cover problem or routing problems, since these kind of problems rely on the local structure of a network.

   (c) One problem that is also still open is whether the upper bound on $Q(T)$ (and thus also on the minimum length (fundamental) cycle base problem) can be improved to $O(m \log n)$. This open problem was stated by Elkin et al. and is—to our knowledge—not yet solved [73].

5. As already sketched above, it is intuitive that a steep backbone distribution should be correlated with the possibility to find a partitioning of the vertices such that well-known clustering quality measures such as the modularity introduced by Girvan and Newman is high [93]. One open problem in this area is thus whether there is a strong correlation between a steep tree distance distribution and the minimal modularity the optimal partition of a graph will have. A second question is whether a good backbone can be used to find reasonable clusterings.

6. As we have sketched in chapter 5 we do not yet have a good network model that produces a steep tree distance distribution besides the naive, hierarchical model. It would certainly be interesting to find one or more models that generate this special structure.

7. A further open question is how the minimum length fundamental cycle base of a graph and that of a (non–fundamental) cycle base of a graph depend on each other besides the simple fact that the first gives an upper bound on the latter.

The open questions addressed in this list give examples of how complex systems science has opened new fields that are interesting for computer scientists. But we see that complex systems science could also profit from computer science: Complex systems science is interested in the question of how the properties of a complex system emerge due to the interactions of its constituting entities. We have now built a general framework to analyze the question of how network generating entities can interact to build globally optimal networks, one of the many possible emergent properties of a complex system.
In a way, the perspective of complex systems science is based on a reverse engineering approach: an emergent property of a system is observed and then the question arises of how this property is achieved. Computer scientists often analyze models and their capabilities to compute something efficiently by changing parameters of the model in a subtle way to understand when and where limiting cases emerge that make a computation difficult or impossible. We think that this approach could be a complementary way to understand the capabilities of complex systems to build emergent properties. In the concrete case of network formation by selfish and myopic entities, this could mean to explore how efficient these entities can shrink the diameter (or build any other global structure in a network) when different grades of cooperation are allowed or different ranges of locality are assumed. We hope that this classical approach in computer science could help to understand the limits of decentral computations (certainly one of the emergent properties of complex systems), and could also rule out some mechanisms for the generation of a given emergent property. In this perspective we think that the following open problems are interesting:

1. What kind of global structures in a network can be achieved by the collective behavior of the agents? E.g., what if the network's diameter should be decreased to a wanted value but under different constraints such as a restricted maximal length of an edge or a restricted degree? What about other global structures, like finding a good backbone in a decentralized and local manner?

2. What would happen if the changes to the network happen unsynchronized, i.e., different vertices can change the network at the same time? For our analysis it was crucial that every vertex could make a temporary change to the network, evaluate this change, and decide subsequently whether it is better to keep the new edge or restore the old one.

3. All game theoretic models imply that the playing entities will instantly find the equilibrium, but our analyses have shown that if total knowledge cannot be assumed, the runtime is strongly dependent on the concrete design of the changing rules. What would happen if some of the entities have global knowledge? Can this be used to make the process find the desired structure faster?

4. We have given a first (empirical) analysis of how sensitive the process is against external events, such as the addition of new vertices to a network that tries to decrease its diameter. Can this question be formally analyzed?

5. We have also addressed the question of how to reduce the number of edges in a sensor network without increasing its diameter. Can it be shown that the diameter will not increased by much w.h.p. if every vertex is chosen uniformly at random to destroy one of the triangles it participates in?

We thus hope that this new approach to understand what complex systems can achieve in general, by focusing on the kind of global network structures they can generate with a certain local behavior, will help to give a theoretical basis for the understanding of emergent properties in complex systems, and thus eventually help to unravel those patterns in real–world systems that seem to be so complex today.

BIBLIOGRAPHY

[1] www.amazon.com, www.amazon.de.

[2] Lada Adamic and Eytan Adar. How to search a social network. *Social Networks*, 27(3):187–203, 2005.

[3] Ravindra K. Ahuja, Thomas L. Magnati, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[4] Réka Albert and Albert-László Barabási. Topology of evolving networks: Local events and universality. *Physical Review Letters*, 85(24):5234–5237, 2000.

[5] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Review of Modern Physics*, 74:47–97, 2002.

[6] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.

[7] Réka Albert and Hans G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the drosophila segment polarity genes. *Journal of Theoretical Biologie*, 223:1–18, 2003.

[8] Albert-László, Hawoong Jeong, and Réka Albert. The diameter of the world wide web. *Nature*, 401:130, 1999.

[9] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-servers problem. *SIAM Journal on Computing*, 24(1):78–100, 1995.

[10] Luís A. Nunes Amaral, A. Scala, Marc Barthélémy, and H.E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Science*, 97:11149–11152, 2000.

[11] Reid Andersen, Fan Chung, and Lincoln Lu. Analyzing the small world phenomenon using a hybrid model with local network flow. In *Proceedings of the WAW 2004*, LNCS 3243, 2004.

[12] Reid Andersen, Fan Chung, and Linyuan Lu. Drawing power law graphs. In *Proceedings of the 12th Symposium on Graph Drawing (GD'04)*, 2004.

[13] Jacob M. Anthonisse. The rush in a directed graph. Technical report, Stichting Mathematisch Centrum, 2e Boerhaavestraat 49 Amsterdam, 1971.

[14] Yael Artzy-Randrup, Sarel J. Fleishman, Nir Ben-Tal, and Lewi Stone. Comment on "network motifs: Simple building blocks of complex networks " and "superfamilies of evolved and designed networks". *Science*, 305:1107c, 2004.

[15] Venkatesh Bala and Sanjeev Goyal. A noncooperative model of network formation. *Econometria*, 68(5):1181–1229, 2000.

[16] Philip Ball. *Critical Mass.* Arrow Books Ltd, London, 2005.

[17] Yaneer Bar-Yam. *Dynamics of Complex Systems.* Westview Press (Perseus Books Group, Boulder, Colorado, USA), 1997.

[18] Albert-László Barabási. *Linked - The New Science of Network.* Perseus, Cambridge MA, 2002.

[19] Albert-László Barabási. Network theory - the emergence of the creative enterprise. *Science*, 308:639–641, 2005.

[20] Albert-László Barabási. Taming complexity. *Nature Physics*, 1:68–70, 2005.

[21] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[22] Albert-László Barabási, Hawoong Jeong, Erzsébet Ravasz, A. Schubert, and Tamás Vicsek. Evolution of the social network of scientific collaborations. preprint at arXiv: cond-mat/0104162, April 2001.

[23] David Barkai. *Peer-to-Peer Computing.* Intel Press, 2002.

[24] Alain Barrat and M. Weigt. On the properties of small-world networks. *The European Physical Journal B*, 13:547–560, 2000.

[25] Marc Bartheélémy and Luís A. Nunes Amaral. Small-world networks: Evidence for a crossover picture. *The American Physical Society*, 82(15):3180–3183, 1999.

[26] Marc Barthélemy. Crossover from scale-free to spatial networks. arXiv:cond-mat/0212086, Feb 2003.

[27] Michael Baur and Ulrik Brandes. Crossing reduction in circular layouts. In *Proceedings of the 30th Workshop on Graph-Theoretic Concepts in Computer Science (WG'04)*, 2004.

[28] Alex Bavelas. A mathematical model for group structures. *Human Organizations*, 7:16–30, 1948.

[29] Peter S. Bearman, James Moody, and Katherine Stovel. Chains of affection: The structure of adolescent romantic and sexual networks. *American Journal of Sociology*, 110:44–91, 2004.

[30] Franziska Berger, Peter Gritzmann, and Sven de Vries. Minimum cycle bases for network graphs. *Algorithmica*, 40:51–62, 2004.

[31] Ginestra Bianconi and Albert-László Barabási. Competition and multiscaling in evolving networks. *Europhysics Letters*, 54(4):436–442, 2001.

[32] Bé Bollobás and Fan Chung. The diameter of a cycle plus a random matching. *SIAM Journal on Discrete Mathematics*, 1:328–333, 1998.

[33] Béla Bollobás. *Modern Graph Theory*. Springer Verlag, Heidelberg, Germany, 1998.

[34] Béla Bollobás. *Random Graphs*. Cambridge Studies in Advanced Mathematics 73. Cambridge University Press, London, 2nd edition, 2001.

[35] Béla Bollobás and Oliver M. Riordan. *Handbook of Graphs and Networks*, chapter Mathematical results on scale-free random graphs, pages 1–34. Springer Verlag, Heidelberg, 2003.

[36] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence*. Santa Fe Institue, New Mexiko, USA, 1999.

[37] Stephen P. Borgatti. Centrality and network flow. *Social Networks*, 27:55–71, 2005.

[38] Stefan Bornholdt. Less is more in modeling large genetic networks. *Science*, 310:449–451, 2005.

[39] Stefan Bornholdt and Thimo Rohlf. Topological evolution of dynamical networks: Global criticality from local dynamics. *Physical Review Letters*, 84(26):6114–6117, 2000.

[40] Stefan Bornholdt and Kim Sneppen. Robustness as an evolutionary principle. *Proceedings of the Royal Society of London B*, 267:2281–2286, 2000.

[41] Ulrik Brandes and Thomas Erlebach, editors. *Network Analysis - Methodological Foundations*. Springer Verlag, 2005.

[42] Pierre Bremaud. *Markov Chains - Gibbs Field, Monte Carlo Simulation, and Queues*. Springer Verlag, 1999.

[43] Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is np-hard. *Computational Geometry. Theory and Applications*, 9(1-2):3–24, 1998.

[44] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[45] Andrei Broder. Generating random spanning trees. In *Proceedings of the 30th Annual Symposium of Foundations of Computer Science*, 1989.

[46] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stat, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer Networks*, 33:309–320, 2000.

[47] Erik Buchmann and Klemens Böhm. FairNet - how to counter free riding in peer-to-peer data structures. In *Proceedings of Cooperative Information System (CoopIS)*, 2004.

[48] Scott Camazine, Jean-Louis Deneubourg, Nigel R. Franks, James Sneyd, Guy Theraulaz, and Eric Bonabeau. *Self–Organization in Biological Systems*. Princeton Studies in Complexity, 2003.

[49] Jean Carle and David Simplot-Ryl. Energy-efficient area monitoring for sensor networks. *Computer*, pages 40–46, 2004.

[50] Jeromy Carriére and Rick Kazman. Interacting with huge hierarchies: Beyond cone trees. In *Proceedings of the ACM conference on Information Visualization 1995*, pages 74–81, 1995.

[51] Julien Cartigny, Francois Ingelrest, David Simplot-Ryl, and Ivan Stojmenović. Localized LMST and RNG based minimum-energy broadcast protocols in ad hoc networks. *Ad Hoc Networks*, 3:1–16, 2003.

[52] Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *PNAS*, 99(25):15879–15882, 2002.

[53] Fan Chung and Linyuan Lu. The small world phenomenon in hybrid power law graphs. In

Complex Networks (E. Ben-Naim, H. Frauenfelder, Z. Toroczkai (eds.)), pages 91–106, 2004.

[54] Aaron Clauset. Finding local community structure in networks. *Physical Review E*, 72:026132, 2005.

[55] Aaron Clauset, Mark E.J. Newman, and Christopher Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.

[56] Colin Cooper, Ralf Klasing, and Michele Zito. Lower bounds and algorithms for dominating sets in web graphs. *Internet Mathematics*, 2(3):275–300, 2005.

[57] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, 2003.

[58] Charles R. Darwin. The origin of species, 1859.

[59] Jörn Davidsen, Holger Ebel, and Stefan Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical Review Letters*, 88:128701, 2002.

[60] Derek J. de Solla Price. Networks of scientific papers. *Science*, 149:510–515, 1965.

[61] Narsingh Deo, Gurpur Madhav Prabhu, and M.S. Krishnamoorthy. Algorithms for generating fundamental cycles in a graph. *Journal on Transactions on Mathematical Software*, 8:28–42, 1982.

[62] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Phys. Rev. Lett.*, 94:160202, 2005.

[63] Guiseppe DiBattista, P. Francesco Cortese, Francesco Frati, Luca Grilli, Katharina A. Lehmann, Guiseppe Liotta, Maurizio Patrignani, Ian Tollis, and Francesco Trotta. On the topologies of local minimum spanning trees. In *Proceedings of the 3rd Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN'06), Chester, UK*, 2006.

[64] Reinhard Diestel. *Graphentheorie.* Springer Verlag, 2000.

[65] Peter Sheridan Dodds, Roby Muhamad, and Duncan J. Watts. An experimental study of search in global social networks. *Science*, 301:827–829, 2003.

[66] Peter Sheridan Dodds, Duncan J. Watts, and Mark E. J. Newman. Identity and search in social networks. *Science*, 296:1302–1305, 2002.

[67] Dietrich Dörner. *Die Logik des Misslingens - Strategisches Denken in komplexen Situationen.* Rowohlt Taschenbuch Verlag GmbH, Reinbek bei Hamburg, 1992 (15th edition).

[68] Sergei N. Dorogovtsev and Jose F.F. Mendes. Exactly solvable analogy of small-world networks. *Europhys. Lett.*, 50:1–7, 2000.

[69] Sergei N. Dorogovtsev and Jose F.F. Mendes. *Evolution of Networks.* Oxford University Press, 2003.

[70] Holger Ebel and Stefan Bornholdt. Evolutionary games and the emergence of complex networks. arXiv: cond-mat/0211666, November 2002.

[71] Holger Ebel, Jrn Davidsen, and Stefan Bornholdt. Dynamics of social networks. *Complexity*, 8(2):24–27, 2003.

[72] Holger Ebel, Lutz-Ingo Mielsch, and Stefan Bornholdt. Scale-free topology of e-mail networks. *Physical Review E*, 66:035103(R), 2002.

[73] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *Proceedings of STOC'05*, 2005.

[74] David Eppstein, Michael S. Paterson, and Frances F. Yao. On nearest-neighbor graphs. *Discrete & Computational Geometry*, 17(3):263–282, 1997.

[75] David Eppstein and Frances F. Yao. On nearest-neighbor graphs. In *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming*, volume 623 of *LNCS*, pages 416–426, 1992.

[76] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5:17–61, 1960.

[77] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1752.

[78] Tim S. Evans. Complex networks. *EUR PHYS J B*, 56:65–69, 2004.

[79] Alex Fabrikant, Ankur Luthra, Elitza Maneva, Christos H. Papadimitriou, and Scott Shenker. On a network creation game. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing*, 2003.

[80] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *Computer Communications Review*, 29:251–262, 1999.

[81] Illés Farkas, Imre Derényi, Albert-László Barabási, and Tamás Vicsek. Spectra of "real-world" graphs: Beyond the semicircle law. *The American Physical Society*, 64:026704, 2001.

[82] Jean-Daniel Fekete, David Wang, Niem Dang, Aleks Aris, and Catherine Plaisant. Overlaying graph links on treemaps. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'03)*, 2003.

[83] Sandor P. Fekete, Michael Kaufmann, Alexander Kröller, and Katharina A. Lehmann. A new approach for boundary recognition in geometric sensor networks. In *Proceedings of the 17th Canadian Conference on Computational Geometry*, 2005.

[84] David A. Fell and Andreas Wagner. The small world of metabolism. *Nature Biotechnology*, 18:1121–1122, 2000.

[85] David A. Fell and Andreas Wagner. The small world of metabolism. *Nature Biotechnology*, 18:1121–1122, 2000.

[86] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution.* Wiley, 1966.

[87] Linton Clarke Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41, 1977.

[88] Koen Frenken and Frank G. van Oort. The geography of research collaboration in US aerospace engineering and US biotechnology & applied microbiology. In *Conference of the Regional Studies Association*, 2003.

[89] Marco Gaertler. *Network Analysis: Methodological Foundations*, chapter Clustering, pages 178–215. Springer-Verlag, 2005.

[90] Michael T. Gastner and Mark E.J. Newman. Shape and efficiency in spatial distribution networks. *Journal of Statistical Mechanics: Theory and Experiment*, page P01015, September 2004.

[91] E. N. Gilbert. Random graphs. *Anual Math. Statist.*, 30:1141–1144, 1959.

[92] Herbert Gintis. *Game Theory Evolving.* Princeton University Press, Princeton, New Jersey, 2000.

[93] Michelle Girvan and Mark E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99:7821–7826, 2002.

[94] Michael Gleich. *The Web of Life.* Hoffmann und Campe Verlag, Hamburg, 2002.

[95] K.-I. Goh, B. Kahng, and D. Kim. Universal behavior of load distribution in scale-free networks. *Physical Review Letter*, 87(27):278701, 2001.

[96] Martin Golumbic and Ann Trenk. *Tolerance Graphs.* Cambridge University Press, 2004.

[97] M.C. Gőpfert and D. Robert. The web of human sexual contacts. *Nature*, 411:907–908, 2001.

[98] Stephen J. Gould. *The Structure of Evolutionary Theory.* The Belknap Press of Harvard University Press, 2002.

[99] R. Govindan and H. Tangmunarunkit. In *Proceedings of the IEEE INFOCOM'00, Tel Aviv*, pages 1371–1380, 2000.

[100] A. Grama, G. Karypis, V. Kumar, and A. Gupta. *An Introduction to Parallel Computing.* Addison Wesley, 2003.

[101] Jonathan L. Gross and Jay Yellen. *Handbook of Graph Theory.* CRC Press, 2004.

[102] Jonathan L. Gross and Jay Yellen. *Handbook of Graph Theory (Jonathan L. Gross and Jay Yellen (Eds.))*, chapter Introduction to Graphs, pages 1–55. CRC Press, 2004.

[103] Jonathan L. Gross and Jay Yellen. *Graph Theory and Its Applications.* Chapman & Hall/CRC, 2006.

[104] David Hartvigsen and Eitan Zemel. Is every cycle basis fundamental? *Journal of Graph Theory*, 13(1):117–137, 1989.

[105] Wolfgang Hennig. *Genetik.* Springer Verlag, Heidelberg, 1995.

[106] Ivan Herman, Guy Melancon, Maurice M. Ruiter, and Maylis Delest. Latour – a tree visualization system. In *Proceedings of the 7th International Symposion on Graph Drawing*, pages 392–399, 1999.

[107] J. Holland. *Adaptation in Natural and Artifical Systems.* Ann Arbor: The University of Michigan Press, 1975.

[108] Petter Holme. Congestion and centrality in traffic flow on complex networks. *Advances in Complex Systems*, 6:163, 2003.

[109] J.D. Horton. A polynomial–time algorithm to find the shortest cycle basis of a graph. *SIAM J. Comput.*, 16:359–366, 1987.

[110] http://www.yworks.com/en/products_yfiles _about.html.

[111] Wilfried Imrich and Peter F. Stadler. Minimum cycle bases of product graphs. *Australasian Journal of Combinatorics*, 26:233–244, 2002.

[112] François Ingelrest, David Simplot-Ryl, and Ivan Stojmenović. *Resource Management in Wireless Networking*, chapter Energy-Efficient Broadcasting in Wireless Mobile Ad Hoc Networks. Kluwer Academic Publishers, 2004.

[113] Riko Jacob, Dirk Koschützki, Katharina A. Lehmann, Leon Peeters, and Dagmar Tenfelde-Podehl. *Network Analysis - Methodological Foundations*, chapter Algorithms for Centrality Indices. Springer Verlag, 2005.

[114] S. Janson. Asymptotic degree distribution in random recursive trees. *Random Structures and Algorithms*, 26(1–2):69–83, 2005.

[115] Henrik Jeldtoft Jensen. *Self-Organized Criticality.* Cambridge University Press, Cambridge, UK, 1998.

[116] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.-L. Barabsi. The large-scale organization of metabolic networks. *Nature*, 400:107, 2000.

[117] Emily M. Jin, Michelle Girvan, and Mark E.J. Newman. The structure of growing social networks. *Physical Reviews E*, 64:046132, 2001.

[118] Michael Kaufmann, Jan Kratochvíl, Katharina A. Lehmann, and Amarendran Subramanian. Max-tolerance graphs as intersection graphs: Cliques, cylces, and recognition. In *Proceedings of the 17th ACM Symposium on Discrete Algorithms (SODA'06)*, 2006.

[119] Michael Kaufmann, Katharina A. Lehmann, and Andreas Gerasch. Area-optimal hv-like tree drawings with a fixed aspect ratio. In *Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science*, 2005.

[120] Michael Kaufmann, Katharina A. Lehmann, and Hendrik Post. On small-world generating models. In *Proceedings of the 2nd European Conference on Complex Systems (ECCS'05)*, 2005.

[121] D. Kent. *The Rise of the Medici: Faction in Florence.* Oxford University Press, 1978.

[122] Pedram Keyani, Brian Larson, and Muthukumar Senthil. *Web Engineering and Peer-to-Peer Computing: NETWORKING 2002*, chapter Peer Pressure: Distributed Recovery from Attacks in Peer-to-Peer Systems, pages 306–320. Springer Berlin/Heidelberg, 2002.

[123] G. Kirchhoff. Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird. *Annalen der Physikalischen Chemie*, 72(12):497–508, 1847.

[124] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[125] Jon Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.

[126] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[127] Jon Kleinberg. Small–world phenomena and the dynamics of information. In *Proceedings of the 13th Advances in Neural Information Processing Systems*, 2001.

[128] J. S. Kleinfeld. The small world problem. *Society*, 39:61–66, Jan.-Feb. 2002.

[129] Konstantin Klemm and Víctor Eguíluz. Highly clustered scale-free networks. *Physical Review E*, 65:036123, 2002.

[130] Donald Ervin Knuth. *The Art of Computer Programming*, volume 1. Addison Wesley, 3rd edition, 1997.

[131] Christof Koch and Gilles Laurent. Complexity and the nervous system. *Science*, 284:96–98, 1999.

[132] Dirk Koschützki, Katharina A. Lehmann, Leon Peeters, Stefan Richter, Dagmar Tenfelde-Podehl, and Oliver Zlotowski. *Network Analysis - Methodological Foundations*, chapter Centrality Indices. Springer Verlag, 2005.

[133] Dirk Koschützki, Katharina A. Lehmann, Dagmar Tenfelde-Podehl, and Oliver Zlotowski. *Network Analysis - Methodological Foundations*, chapter Advanced Centrality Concepts. Springer Verlag, 2005.

[134] D. Krackhardt. Cognitive social structures. *Social Networks*, 9:109–134, 1987.

[135] Valdis Krebs. The social life of books. http://www.orgnet.com/booknet.html.

[136] Thomas Kropf. *Introduction to Formal Hardware Verification.* Springer Verlag, Heidelberg, 1998.

[137] M. Kuba and A. Panholzer. On the degree distribution of nodes in increasing trees. *Journal of Combinatorial Theory, Series A (to appear)*, 2006.

[138] F. Kuhn, T. Moscibroda, and Roger Wattenhofer. Unit disk graph approximation. In *Proceedings of the DIALM-POMSC'04*, 2004.

[139] Marcelo Kuperman and Guillermo Abramson. Small world effect in an epidemiological model. *Physical Review Letters*, 86(13):2909–2912, 2001.

[140] Olaf Landsiedel, Klaus Wehrle, and Katharina A. Lehmann. T-DHT: Topology based distributed hash tables. In *Proceedings of the 5th International IEEE Conference on Peer-to-Peer-Computing, Konstanz, Germany*, 2005.

[141] Michael Lappe and Liisa Holm. Unravelling protein interaction networks with near-optimal efficiency. *Nature Biotechnology*, 22:98–103, 2004.

[142] Katharina A. Lehmann. Why simulating evolutionary processes is just as interesting as applying them. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'05)*, 2005.

[143] Katharina A. Lehmann. The structure of real-world sat-problems. Master's thesis, University of Tübingen, 2006.

[144] Katharina A. Lehmann and Michael Kaufmann. Decentralized algorithms for evaluating centrality in complex network. Technical report, Technical Report of the Wilhelm-Schickard-Institute, University Tübingen, WSI-2003-10, ISSN 0946-3852, 2003.

[145] Katharina A. Lehmann and Michael Kaufmann. Evolutionary algorithms for the self-organized evolution of networks. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'05)*, 2005.

[146] Katharina A. Lehmann and Michael Kaufmann. *Peer-to-Peer Systems and Applications*, chapter Random Graphs, Small Worlds, and Scale-Free Networks. Springer Verlag, 2005.

[147] Katharina A. Lehmann, Michael Kaufmann, Stephan Steigele, and Kay Nieselt. On the maximal cliques in c-max tolerance graphs and their application in clustering molecular sequences. *Algorithms for Molecular Biology*, 1:9, 2006.

[148] Katharina A. Lehmann and Stephan Kottler. Visualizing large and complex networks. Technical Report WSI–2006–06, ISSN 0946–4852, Wilhelm-Schickard-Institute, University of Tuebingen, 2006.

[149] Katharina A. Lehmann and Stephan Kottler. Visualizing large and complex networks. In *Proceedings of the 14th International Symposium on Graph Drawing (GD'06)*, 2007.

[150] Katharina A. Lehmann, Hendrik Post, and Michael Kaufmann. On small-world generating models. Technical report, Technical Report of the Wilhelm-Schickard-Institute, University Tübingen, WSI-2005-17, ISSN 0946-3852, 2005.

[151] Katharina A. Lehmann, Hendrik D. Post, and Michael Kaufmann. Hybrid graphs as a framework for the small-world effect. *Physical Review E*, 73:056108, 2006.

[152] Ronny Lempel and Shlomo Moran. The stochastic approach for link structure analysis (SALSA) and the TKC effect. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 33:387–401, 2000.

[153] Ronny Lempel and Shlomo Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks: The International Journal of COmputer and Telecommunication Networking*, 33:387–401, 2000.

[154] Thomas Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* Teubner Verlag, 1997.

[155] J. Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters, and possible explanations. In *Proceedings of the 11th ACM SIGKDD*, 2005.

[156] Roger Lewin. *Complexity.* The University of Chigaco Press, 1999.

[157] Josef Leypold and Peter F. Stadler. Minimal cycle bases of outerplanar graphs. Technical report, Department of Applied Statistics and Data Processing, Wirschaftsunversität Wien, 1998.

[158] N. Li, J. Hou, and L. Sha. Design and analysis of an MST-based topology control algorithm. In *Proceedings of the IEEE INFOCOM'03*, 2003.

[159] C. Liebchen and R. Möhring. A case study in periodic timetabling. In *Proceedings of ATMOS*, 2002.

[160] Fredrik Liljeros. Sexual networks in contemporary western societies. *Physica A*, 338:238–245, 2004.

[161] Fredrik Liljeros, Christofer R. Edling, Luís A. Nunes Amaral, H. Eugene Stanley, and Yvonne Åberg. The web of human sexual contacts. *Nature*, 411:907–908, 2001.

[162] Georg Löffler and Petro E. Petrides. *Biochemie und Pathobiochemie*. Springer Verlag, 1990 (5th edition).

[163] Peter Mahlmann and Christian Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *17th ACM Symposium on Parallelism in Algorithms and Architectures*, 2005.

[164] H.M. Mahmoud. Limiting distributions for path lengths in recursive trees. *Probability in the Engineering and Informational Sciences*, 5:53–59, 1991.

[165] Stanley Milgram. The small world problem. *Psychology Today*, 1:61–67, 1967.

[166] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and Uri Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[167] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, and Uri Alon. Response to comment on ”network motifs: Simple building blocks of complex networks ” and ”superfamilies of evolved and designed networks”. *Science*, 305:1107d, 2004.

[168] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303:1538–1542, 2004.

[169] Melanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1997 (3rd printing).

[170] J.W. Moon. The distance between nodes in recursive trees. *London Mathematical Society Lecture Notes Series*, 13:125–132, 1974.

[171] Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, 1991.

[172] Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

[173] M. Newman. Models of the small world: A review. *Journal of Statistical Physics*, 101:819–841, 2000.

[174] M. E. J. Newman and D. J. Watts. Scaling and percolation in the small-world network model. *Phys. Rev. E*, 60:7332–7342, 1999.

[175] Mark E. J. Newman and Duncan J. Watts. Renormalization group analysis of the small-world network model. *Phys. Lett. A*, 263:341–346, 1999.

[176] Mark E.J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences, USA*, 98(2):404–409, 2001.

[177] Mark E.J. Newman. Assortative mixing in networks. *Physical Review Letters*, 89:208701, 2002.

[178] Mark E.J. Newman. The structure and function of networks. *Computer Physics Communication*, 147:40–45, 2002.

[179] Mark E.J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, 2004.

[180] Mark E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the USA*, 103:8577–8582, 2006.

[181] Mark E.J. Newman, Albert-László Barabási, and Duncan J. Watts, editors. *The Structure and Dynamics of Networks*. Princeton University Press, Princeton and Oxford, 2006.

[182] Mark E.J. Newman, Steven H. Strogatz, and Duncan J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64:026118, 2001.

[183] James R. Norris. *Markov Chains*. Cambridge: Cambridge University Press, 1997.

[184] Open Network of Excellence in Complex Systems. Living roadmap for complex systems science (version 1.22). http://www.once-cs.net, March 2006.

[185] S. Ohno. *Evolution by Gene Duplication*. Springer Verlag, Berlin, 1970.

[186] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge Massachusetts, 1994.

[187] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Manuscript, 1999.

[188] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814, 2005.

[189] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446:664–667, 2007.

[190] Julia K. Parrish and Leah Edelstein-Keshet. Complexity, pattern, and evolutionary trade-offs in animal aggregation. *Science*, 284:99–101, 1999.

[191] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(4):3200–3203, 2001.

[192] Romualdo Pastor-Satorras and Alessandro Vespignani. *Handbook of Graphs and Networks: From the Genome to the Internet*, chapter Epidemics and Immunization in Scale-Free Networks. Wiley-VCH, Berlin, 2002.

[193] Keith Paton. An algorithm for finding a fundamental set of cycles of a graph. *Communication of the ACM*, 12(9):514–518, 1969.

[194] David Peleg. *Distributed Computing - A Locality Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[195] David Peleg and E. Reshef. Deterministic polylogarithmic approximation for minimum communication spanning trees. In *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming*, 1998.

[196] David Peleg and A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13:99–116, 1989.

[197] Matthew Penrose. *Random Geometric Graphs*. Oxford Studies in Probability, 2003.

[198] Karl R. Popper. *Conjecture and Refutations*. Routledge and Kegan Paul, 2nd edition, 1965.

[199] N. Pržulj, D.G. Corneil, and I. Jurisica. Modeling interactome: Scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[200] Günther R. Raidl and Bryant A. Julstrom. Edge-sets: An effective evolutionary coding of spanning trees. Technical Report TR-186-1-01-01, Technische Universität Wien - Institut für Computergraphik and Algorithmen, 2002.

[201] E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, and A.-L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297:1551–1553, 2002.

[202] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzoog, 1973.

[203] Jörg Reichardt and Stefan Bornholdt. Partitioning and modularity of graphs with arbitrary degree distribution. arXiv:cond-mat/0606295, Juni 2006.

[204] Jörg Reichardt and Stefan Bornholdt. When are networks truly modular? arXiv:cond-mat/0606220, Juni 2006.

[205] Michael R.Garey and David S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[206] M. Ripeanu and I. Foster. Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer networks. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems and Implications for System Design*, 2002.

[207] Arturo Rosenblueth and Norbert Wiener. The role of models in science. *Philosophy of Science*, 12(4):316–321, 1945.

[208] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31:581–603, 1966.

[209] S. Saroiu, K. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networks*, 2002.

[210] J. Scharnow, K. Tinnefeld, and Ingo Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modeling and Algorithms*, 4(3):349–366, 2004.

[211] Anne Margarete Schwahn. Minimum fundamental cut basis problem. Master's thesis, Technical University of Kaiserslautern, 2005.

[212] Alfonso Shimbel. Structural parameters of communication networks. *Bulletin of Mathematical Biophysics*, 15:501–507, 1953.

[213] Horst Siebert. *Der Kobra-Effekt*. Piper, München Zürich, 2003.

[214] Brian Skyrms and Robin Pemantle. A dynamic model of social network formation. *Proceedings of the National Academy of Science of the USA*, 97(16):9340–9346, 2000.

[215] Reginal D. Smith. The network of collaboration among rappers and its community structure. *Journal of Statistical Mechanics: Theory and Experiment*, page P02006, 2005.

[216] Tom A.B. Snijders. *Sociological Methodology*, chapter The Statistical Evaluation of Social Network Dynamics, pages 361–395. Boston and London: Basil Blackwell, 2001.

[217] Tom A.B. Snijders. Markov chain monte carlo estimation of exponential random graph models. *Journal of Social Structure*, 3(2), 2002.

[218] Tom A.B. Snijders. *Models and Methods in Social Network Analysis*, chapter Models for Longitudinal Network Data, pages 215–247. Cambridge University Press, New York, 2005.

[219] Ricard V. Solé, Romualdo Pastor-Satorras, Eric Smith, and Thomas B. Kepler. A model of large-scale proteome evolution. *Advances in Complex Systems*, 5:43–54, 2002.

[220] Ray Solomon and Anatol Rapoport. Connectivity of random nets. *Bulletin of Mathematical Biophysics*, 13:107–117, 1951.

[221] Ralf J. Sommer. *Handbook of Graphs and Networks*, chapter Cells and Genes as Networks in Nematode Development and Evolution, pages 131–144. Wiley-VCH, Weinheim, Germany, 2003.

[222] Daniel A. Spielmann and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th ACM STOC*, 2004.

[223] Christian Steglich. Actor-driven alternatives to exponential random graph models. In *Paper presented at XXVI Sunbelt Social Networks Conference (Vancouver, Canada)*, 2006.

[224] Ralf Steinmetz and Klaus Wehrle, editors. *Peer-to-Peer Systems and Applications.* Springer Verlag, 2005.

[225] Steven Strogatz. Exploring complex networks. *Nature*, 410:269–276, 2001.

[226] Steven H. Strogatz. *Nonlinear Dynamics and Chaos.* Westview Press, 2000.

[227] Steven H. Strogatz. Romanesque networks. *Nature*, 433:365–366, 2005.

[228] J. Szymanski. On the maximum degree and the height of a random recursive tree. *Random Graphs*, 87:313–324, 1990.

[229] Hongsuda Tangmunarunkit, Ramesh Govindan, Sugih Jamin, Scott Shenker, and Walter Willinger. Network topology generators: Degree based vs. structural. In *Proceedings of the SIGCOMM'02*, 2002.

[230] D. Tian and N.D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proceedings of the 1st ACM Workshop Wireless Sensor Networks and Applications*, 2002.

[231] J. Travers and Stanley Milgram. An experimental study of the small wolrd problem. *Sociometry*, 32:425–443, 1969.

[232] Alexei Vázquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. Modeling of protein interaction networks. *ComPlexUs*, 1:38–44, 2002.

[233] Frederic Vester. *Die Kunst vernetzt zu denken.* Deutscher Taschenbuch Verlag, 2004 (4th edition).

[234] Tamas Vicsek. A question of scale. *Nature*, 411:421, 2001.

[235] Tamas Vicsek. The bigger picture. *Nature*, 418:131, 2002.

[236] M. Mitchell Waldrop. *Complexity.* Touchstone (Simon & Schuster Inc.), New York, USA, 1993.

[237] Stanley Wasserman and Katherine Faust. *Social Network Analysis - Methods and Applications.* Cambridge University Press, Cambridge, revised, reprinted edition, 1999.

[238] Alison Watts. A dynamic model of network formation. *Games and Economic Behavior*, 34(2):331–341, 2001.

[239] Duncan J. Watts. *Small Worlds- The Dynamics of Networks between Order and Randomness.* Princeton Studies in Complexity. Princeton University Press, 1999.

[240] Duncan J. Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9):5766–5771, 2002.

[241] Duncan J. Watts. *Six Degreees - The Science of a Connected Age.* W.W. Norton & Company, New York, London, 2003.

[242] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

[243] Bernard M. Waxman. Routing of multipoint connections. *Journal on Selected Areas of Communication*, 6(9):1617–1622, 1988.

[244] Jörgen Weibull. *Evolutionary Game Theory.* The MIT Press, Cambridge, Massachusetts, 1997.

[245] Karsten Weicker. *Evolutionäre Algorithmen.* Teubner Verlag, Stuttgart, 2002.

[246] George M. Whitesides and Rustem F. Ismagilov. Complexity in chemistry. *Science*, 284:89–92, 1999.

[247] Harry Wiener. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*, 69:17–20, 1947.

[248] David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the 28th annual ACM symposium on Theory of computing*, 1996.

[249] Carsten Witt. An analysis of the (μ+1) EA on simple pseudo-boolean functions. In *Proceedings of the Genetic and Evolutionary Computation (GECCO'04)*, 2004.

[250] Stephen Wolfram. *A New Kind of Science.* Wolfram Media, Inc., 2002.

[251] www.arxiv.org.

[252] Feng Xue and P.R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10:169–181, 2004.

[253] Fan Ye, Gary Zhong, Jesse Cheng, Songwu Lu, and Lixia Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. In *Proceedings of the IEEE International Conference of Network Protocols (ICNP 2002)*, 2002.

[254] Soon-Hyung Yook, Hawoong Jeong, and Albert-László Barabási. Modeling the internet's large-scale topology. *Proceedings of the National Academy of the Sciences, USA*, 99(21):13382–3386, 2002.

[255] Soon-Hyung Yook, Hawoong Jeong, Albert-László Barabási, and Yuhai Tu. Weighted evolving networks. arXiv: cond-mat/0101309, Jan 2001.

8. DATA

8.1 Data sets

Some of the data used in this work have been published by others and were free for use. Here, we will shortly give an acknowledgement for those data sets. The data sets used will be available on request from the author to reproduce the data. Other data were crawled or computed by the author or students of the group. We will briefly note the author of the according software and the procedure by which the data was obtained.

8.1.1 Amazon co-purchasing Networks

The webshop Amazon [1] offers a SOAP interface by which data that is available on their webpages is also freely available for automatic requests. In theory, every information on the websites should be available by this interface but it turned out that the information is not necessarily the same. **Andreas Gerasch** thus built a web bot that crawls the webpages as if it were a normal user. The bot starts at some predefined starting book, indicated by the unique ISBN number. As additional parameters, the number of threads and the depth unto which links are followed are given.

The bot searches for links to other books that are placed directly under the text line: 'Customers who bought this book also bought', the co-purchasing information. There are no more than six links under this title (August 2006), and all of them are stored in a data base. This data base stores each search with a different ID such that it is possible to store the same book multiple times. The tool built by Andreas provides many very helpful features such as building the difference grap given two graphs G_1 and G_2: In a difference graph, every vertex or edge that is only in G_1 is colored red, every vertex or edge that is only in G_2 is colored green, all other vertices and edges are black.

8.1.2 Autonomous System

The National Laboratory for Applied Network Research (**NLANR**) has documented the evolution of the Internet from November 1997 to March 2001 and made this data publicly available at *http://moat.nlanr.net/AS/*. An *Autonomous System* is generally a group of routers and computer networks under the control of one entity.

The raw data provided by the NLANR gives routing information that implicitly contains information on which Autonomous Systems are directly connected. For each month, the data of the first ten days is combined and displayed as a network, where the Autonomous Systems are represented by the vertices and two vertices are connected if there is at least one path in which the two Autonomous Systems are listed consecutively[1]. Note that this data is not perfect: If some paths

[1] Our thanks go to Jan Vitense who provided us with this data

are only rarely used, an edge might emerge in one month, be missing in the next few months, and show up again later. We thus regarded only those edges and vertices as *new* that had never before been seen in any of the earlier networks but where both vertices had already been in at least one of the preceeding networks.

8.1.3 Co-Authorship Network

The data was kindly provided by M.E.J. Newman who used the same networks [176, 179]. These networks are available from his homepage http://www-personal.umich.edu/ mejn/netdata.

The networks represent authors of papers published in the online preprint archive *arxiv*. In the first data set, all papers published between 1995 and 1999 were analyzed, and in the second, all papers published between 1995 and 2003. The networks are weighted but in this analysis we disregarded the weights, and only the biggest connected component is used. The first data set contains 13,861 vertices and 44,619 edges, and the second 27,519 vertices and 116,181 edges. Of those, 57,277 edges are new edges between authors that were already present in the first network. These edges are used to determine the new edge distance distribution.

8.1.4 Word Association and Protein-Protein Interaction Network

The data was kindly provided by Palla et al. who used the same networks in [188]; these networks are available together with the tool *CFinder* at http://angel.elte.hu/ vicsek.

For the word association network, people were asked what word they associate with a given word, and two words are connected by an edge if at least one person associated the two words with each other. The network contains 7,205 vertices and 31,783 edges in the biggest connected component which was used here. Palla et al. name a website from which the data can be obtained that seems to be outdated. The data can now be found starting from page http://w3.usf.edu/FreeAssociation/intro.html.

The protein-protein interaction network presents proteins from the organism *Saccharomyces cerevisiae* that were found to interact with each other in biological experiments. This network contains 2,445 vertices and 6265 edges in the biggest connected component, where self-loops have been removed.

8.1.5 Live–Journal Networks

Live–Journal is an online platform that allows users to build a personal page and to blog. Users can link to other users (or themselves), called friends. For the creation of the Live Journal network a crawl was started at one participant of www.livejournal.com, following the links to designated friends unto depth 3. The network size increases strongly in dependence of the depth since most users have links to between 50–100 ‘friends’.

9. PUBLICATIONS

In this chapter I will give a short abstract for all of my publications that could not be explained in detail in this work but were published during my period of graduate studies.

9.1 Graph Drawing

9.1.1 Area-optimal HV-like Tree Drawings with a Fixed Aspect Ratio

This article describes a new way to visualize trees such that any desired aspect ratio of width and height can be computed in a very efficient way. The paper was published in the proceedings of the 31st conference on "Current Trends in Theory and Practice of Computer Science" [119].

9.2 Centrality Indices

9.2.1 Centrality Indices, Algorithms for Centrality Indices, and Advanced Centrality Concepts

There are four publications centered around the idea of *centrality indices*, three of them in a review-like book on *network analysis* [41] written together with a group of people, where I was one of the main authors in chapter [132] and [133]. The book was supported by the *GI*, the German Society of Computer Science, and published in the *Lecture Notes in Computer Science (LNCS)* series. The idea of these books is to review the latest advances in science that are not yet covered by text books. The first chapter gives the definition of various centrality indices. Since these indices were developed by different communities, and sometimes not even explicitly as centrality indices, we tried to find categories of similar measures in which every individual centrality index finds its place [132]. The second chapter gives an overview of efficient algorithms to compute these centrality measures [113]. The third chapter explains and explores advanced concepts of centrality indices, e.g., the stability of centrality indices against perturbations of the data, or a framework that shows when which of the centrality indices can be applied to find the most central vertex in a given network [133].

9.2.2 Decentralized Algorithms for Evaluating Centrality in Complex Network

This paper is a technical report, written together with Michael Kaufmann where we show that shortest–path–based centrality indices can also be computed in decentralized networks where communication is allowed but no vertex can keep the whole adjacency matrix of the network in memory [144].

9.3 Sensor Networks and Peer-to-Peer Systems

9.3.1 Random Graphs, Small Worlds, and Scale-Free Networks

Similar to the network analysis book described above, Ralf Steinmetz and Klaus Wehrle wrote a book on **Peer-to-Peer Systems and Applications** in the LNCS series. They invited Michael Kaufmann and me to write a chapter on the influence that network topology and network modeling has on current peer-to-peer systems [146].

9.3.2 A New Approach for Boundary Recognition in Geometric Sensor Networks

This publication describes an application of centrality indices to sensor networks, written together with Sandor Fekete, Alexander Kroeller, and Michael Kaufmann. Sensors are very small devices that can communicate, are not too expensive, but therefore have only limited ressources, e.g., battery life and communication radius. The idea is to spread them by the thousands into an area, let them collect data in a self-organized way, and use this data, e.g., to forecast a possible breach in a dyke, or to analyze the dynamics in temperature and salinity of the deep sea. We developed a new centrality index that makes it possible to differentiate between those vertices that lie on the border of an area and those that are surrounded by many other sensors. The article was published in the proceedings of the 17th "Canadian Conference on Computational Geometry" [83].

9.3.3 T-DHT: Topology–Based Distributed Hash Tables

Together with Olaf Landsiedel and Klaus Wehrle we developed a new idea about how to manage data in a sensor network. The main idea is to use the adjacency matrix of a sensor network to develop something like a topology (i.e., a number space) such that two vertices that are assigned two close numbers should not be too far apart in the network. If such a labeling can be found it can be used to store the data in a topology–based distributed hash table. The article was published in the proceedings of the "IEEE Conference on Peer-to-Peer Computing" [140].

9.3.4 On the Topologies of Local Minimum Spanning Trees

This article was the surprising outcome of a one week workshop on graph drawing in Bertinoro, Italy. Together with six co-authors, we worked on the problem of the so-called *local minimum spanning tree (LMST)*, i.e., a subgraph structure that comprises all vertices (*spanning*) but that can be computed locally where every vertex can only use the data available from vertices in distance k to it. My contribution to this article, together with M. Patrignani, was to show that given the adjacency matrix, it is NP-hard to decide whether it could be the LMST-graph of the embedded graph. The article was published in the proceedings of the 3rd "Workshop on Combinatorial and Algorithmic Aspects of Networking" (CAAN'06) [63].

9.4 Network and Structural Analysis of SAT problems

Since I was quite advanced in my graduate studies when I finally wrote my diploma thesis to complete my studies in informatics, network analysis also became part of my diploma thesis. In this work, we tried to apply the ideas of network analysis and network modeling to the structural analysis of so-called *satisfiability problems (SAT problems).* In these problems, a Boolean formula in *conjunctive normal form* is given and the question is whether there is an assignment of Boolean values to the variables such that the whole formula evaluates to true (in Boolean logic). It is long known that so-called *real-world instances*, i.e., those that arise from technical applications like software- or hardware-verification [136], are much easier to solve than expected given the enormous number of variables and constraints. Since SAT problems can be transformed into graphs, our approach was to find special structural properties of real-world SAT problems that differentiate them from random SAT problems. Our findings are published in my diploma thesis and are currently being prepared for publication [143].

9.5 *C*-max Tolerance Graphs

Triggered by a bioinformatic application developed in the working group of Kay Nieselt (Tübingen), we wrote two paper on c-max tolerance graphs that helped to characterize this family of graphs, a longstanding open question in the realm of tolerance graphs [96]. In general, tolerance graphs are based on a given set of intervals where two intervals are connected if they overlap in a defined way. For simple interval graphs in which two vertices are connected if the corresponding intervals overlap to any extent it was known that finding all maximal cliques is in $O(n)$, although it is, in general, NP-hard. In c-max tolerance graphs two vertices are connected if the corresponding intervals overlap such that each covers at least $0 \geq c \geq 1.0$ of the length of the other interval.

9.5.1 Max-Tolerance Graphs as Intersection Graphs: Cliques, Cylces, and Recognition

The first paper deals with the characterization of this family of graphs, written together with Michael Kaufmann, Jan Kratochvíl, and Amarendran Subramanian. We showed that finding all maximal cliques in these graphs is still in P, namely in $O(n^3)$, but therefore harder than in simple interval graphs. Jan Kratochvíl was able to show that the recognition of these graphs is NP-hard, and could answer two open questions regarding c-max tolerance graphs. This paper was published in the proceedings of the 17th "ACM Symposium on Discrete Algorithms" (SODA'06) [118].

9.5.2 On the Maximal Cliques in c-max Tolerance Graphs and Their Application in Clustering Molecular Sequences

The second paper deals with the application of these findings to biological data, namely how to compute all maximal cliques in c–max tolerance graphs of DNA- or amino-acid sequences. The paper was written together with Michael Kaufmann, Kay Nieselt, and Stephan Steigele, and published in **Algorithms for Molecular Biology**, 2006 [147].

INDEX