Sociality in Complex Networks

Università degli Studi di Catania

Dipartimento di Ingegneria Elettrica, Elettronica e dei Sistemi

Marco Buzzanca

Abstract

The study of network theory is nothing new, as we may find the first example of a proof of network theory back in the 18th century. However, in recent times, many researchers are using their time to investigate networks, giving new life to an old topic. As we are living in the era of information, networks are everywhere, and their complexity is constantly rising. The field of complex networks attempts to address this complexity with innovative solutions. Complex networks all share a series of common topological features, which revolve around the relationship between nodes, where relationship is intended in the most abstract possible way. Nonetheless, it is important to study these relationships because they can be exploited in several scenarios, like web page searching, recommender systems, e-commerce and so on. This thesis presents studies of sociality in complex networks, ranging from the microscale, which focuses the attention on the point of view of single nodes, to the mesoscale, instead shifts the interest in node groups.

Declaration

I declare herewith, that this dissertation is my own original work. Furthermore, I confirm that:

- this work has been composed by me without assistance;
- I have clearly referenced all sources used in the work;
- all data and findings in the work have not been falsified or embellished;
- this work has not been published.

Contents

Chapter 1

Introduction

In network theory, a network (or graph) is a model that aims at abstracting the symmetric or asymmetric connections among entities. The concept of a network is definitely not new in the field of mathematics, dating back to 1735, when Euler provided what it has been called the first proof of network theory: the solution to the Königsberg bridges problem. Since then a lot has changed, the real world has become increasingly more complex, and in the age of computer science, networks are everywhere. Networks have increased in complexity too: things like web-based social networks or the world wide web are monstrous entities that are made of billions of nodes. But it's not just the amount of nodes that makes the study of these networks more difficult, as the most interesting aspect of these networks are the way the links are distributed. At a first glance, no clear pattern appears, and nodes seem to be randomly interconnected, while in reality their connections are neither regular not random, but somewhere in-between. Basically, there is a sort of regularity in the randomness, as these networks share the same interconnection patterns. A lot of effort has been put into the research of these patterns, which over time has created a specific research field, which is the field of *complex networks*. To

summarize, a network is complex when it displays non-trivial topological features. Some of the features that are shared among complex networks are:

- heavy tail in the degree distribution;
- high clustering coefficient;
- assortativity or disassortativity among vertices;
- hierarchical structure;
- community structure.

Firstly, the degree distribution of complex networks can be heavy-tailed: which means there are many nodes that have a low degree, and few nodes that have an high degree. Nodes also tend to cluster themselves together, resulting in an unusually high clustering coefficient. The way nodes are clustered together depend on the type of network: in social networks nodes tend to form links with nodes that are similar, showing *assortativity*, while in biological networks entities seek diversity in their connections, showing a certain degree of *disassortativity*. Moreover, in networks with a significant amount of clusters, cluster themselves may sometimes be grouped in "clusters of clusters", effectively forming a hierarchical structure. But the last feature in particular has attracted the interest of many researchers, which is a property of the node clusters themselves, the community structure. The definition of community is controversial, but the general consensus is that a set of nodes may be grouped in a community if they are densely interconnected among each other. Community structure inside complex networks suggests that there is a somewhat natural division among the network nodes that emerges from the network itself. Finding these communties has become an hot topic in the field of complex networks, as

the increased complexity presents new challenges and obstacles that need to be addressed (literature review on the matter in Chapter 2).

Note that the list provided is not complete nor mandatory, there are indeed complex networks that, for example, are not heavy-tailed in the degree distribution. Hoewver, most complex networks share at least some of those properties compared to random networks, and they represent a hint that the network under exam is indeed complex. Of all the properties mentioned above, my work has mostly focused on assortativity, community structure and clustering. I found myself studying how nodes arrange themselves in groups, and asking myself why. In a certain sense, my approach was analogous to a sociologist's. After all, some complex networks are indeed models of social networks, but the social behavior also extends to other types of networks as well. If we take the concept of homophily, for example, and extend it to other types of entities in complex networks, we find that it maps quite gracefully to the idea of assortativeness.

There are plenty of other social features and properties in the realm of complex networks: as already mentioned nodes tend to arrange themselves in groups. These groups are hardly static in nature, and they change over time. The dymamics that rule these changes are, again, social-based. My work in the past three years was focused on different things, but the common theme is that I tried to investigate social behaviour in complex networks. There are many real world applications of these studies. In web page ranking, for example, it is important to anaylize how web pages are linked to each other to assess the relevance of a certain web page. Information such as the number of outlinks, the number of inlinks, which pages are pointed and which pages point to our page, is very useful to determine its importance. In the e-commerce scenario, we want to have a successful transaction, so we are looking for trustworthy nodes to interact with. Trustworthiness is assessed by

evaluating what other nodes "think" of the node we wish to contact. It is then essential to abstract the level of trustworthiness that a node places on his neighbours by giving them a score, or trust value. This score is an extremely condensed and simplified measure of how we trust that neighbour, and usually depends on the past history between the two nodes. If we had a history of successful interactions, the resulting score will be higher, and vice-versa. Recommender systems also make use of social interactions to predict which product to suggest. They see which products are commonly bought by similar users and suggest them to the same class of users. This of course requires clustering users that share the same interests in the same group.

My work during the past three years consisted in studying sociality in complex network. The focus was mainly on three facets of sociality: trust, communities, and popularity, intended as the importance of the node in the network. I have studied how to model trust out of metadata information [1, 2], how to raise the importance of a node in the network [3], how to assess the importance of a node [4], and how to evaluate the quality of a specific partition in communities [5, 6].

This thesis is structured as follows. Chapter 2 provide a succint review of the literature about topic inherent to this thesis. Chapter 3 describes research done in the microscale, where the emphasis is on nodes. On the contrary, Chapter 4 gives a broader point of view, focusing on the research done for groups and communities. At last, Chapter 5 reviews the content of this thesis.

Chapter 2

Related Works

This section will provide a succint summary about the state of the art of the three most important topics discussed in this thesis: trust networks, ranking algorithms and community detection.

2.1 Trust networks

Over the last decades, he term *trust* has been characterized with different meanings [7, 8, 9, 10, 11], depending on which context is considered (e.g. sociology, psychology and so on). Although there is no definitive agreement on these various definitions of trust, most researchers agree from the early beginning [12] that it is fundamental whenever an individual takes a risk and there is uncertainty about the outcome [13].

Given that the definition of trust itself is not universal, establishing whenever a certain node trusts another is not an easy task, and requires data aggregation strategies. In [14], the authors provide an overview of the most recent achievements and open challenges about Big Data mining. It is also important to take into account the trustworthiness of the collected data to obtain trust information about the users [15, 16].

Trust modeling goes according to the context: links model trust in trust networks, hyperlinks in website ranking, or buyer-seller relationships in the e-commerce context. In ICT context [17] trust is a leading tool for computers and software agents to discriminate themselves among *good* and *bad* ones, its role in broader scenarios is steadily increasing, in particular when humans communicate with others in virtual environments [18], with on-line services [19, 20], with intelligent pervasive environments [21] and so on [22, 23, 24, 25, 26, 27]. In all these situations indeed people generally have to provide some of their personal information in order to positively fulfill the interaction, and they rely on *trust* as a key factor to establish whether the counterpart is worth to connect to.

Trust network frameworks [28, 10, 29, 30] model the trust network as a graph where nodes are agents (persons) and trusting relationships are directed arcs weighted using a measure of the *direct* trust value according to a given metric. As pointed out by Artz and Gil [31] trust can be intended as a measure of how good the future behavior of a given agent will be based on his past actions, in other words the *reputation* can be considered as an effective approach for trust assessment. Many trust assessment algorithms, such as EigenTrust [32], Powertrust [33] and GossipTrust [34], use the feedback mechanism to evaluate the trustworthiness of an agent.

Despite the importance of neighbours in the definition of trust, existing literature mainly focuses on the assessment of *global* trust, i.e. the unique value for a node that aims at mediating all direct values that express different judgements the node received from others. A work that strengthened the role of *local* trust is TrustWebRank [35], where different trust values can be assigned by distinct nodes to the same one. The need for local trust is supported also by other researchers, e.g. [36] claims that local values are needed when a shared opinion cannot be achieved (controversial nodes). Finally, the local

approach is more precise and tailored to the point of view of each user and also more attack-resistant to malicious peers [37].

2.2 Ranking algorithms

Ranking algorithms are algorithms that process node metadata and topology information, and produce an ordered list of nodes. The order of the nodes has different meanings according to the ranking algorithm employed but, in general, higher positions are assigned to nodes which are more relevant to the specific algorithm. Relevance depends on the scenario being considered: in web searching relevance is measured against a given search query [38, 39, 40], in E-learning resources must be relevant to a given topic [41, 42, 43], in a recommendation network the most reliable nodes are the most relevant [44, 45, 46, 47, 48, 49], which is also true for e-commerce scenarios [50, 51].

One of the most important ranking algorithms is PageRank[52, 53]. PageRank is essentially an application of the random walker model on a Markov chain: the nodes of the Markov chain are the web pages, and the arcs are the links that connect one page to another. The walker represents a generic web surfer which moves from page to page with a certain probability, according to the network structure, and occasionally "gets bored" and jumps to a random node in the network. The steady-state probability vector of the random walker process holds the PageRank values for each node, which can be used to determine the global ranking. Although Pagerank was proposed a long time ago, it still lives as the backbone of many technologies, not limited to the web domain. For example, in [54], personalized PageRank is cited as a possible algorithm to be used in Twitter's "Who To Follow" architecture. In [55], the author shows how the mathematics behind PageRank have been used in a plethora of applications which are not limited to

ranking pages on the web. In [56], another PageRank extension appears as a tentative replacement of the h-index for publications.

PageRank has to face a plethora of competitors in several application domains. In the web domain we have HITS [57], which is not based on the random walker model and is able to provide both an "authority" ranking, which rewards nodes that have many backlinks, and a "hub" ranking, which rewards nodes that have many forward links. SALSA [58] computes a random walk on the network graph, but integrates the search query into the algorithm, which is something PageRank does not do. In the trust networks domain we have PeerTrust [33], which computes the global trust by aggregating several factors, and PowerTrust [59] which uses the concept of "power nodes", which are dynamically selected, high reliability nodes, that serve as moderators for the global reputation update process. The PowerTrust article also describes how the algorithm compares to EigenTrust with a set of simulations that analyse its performance. Several articles feature side-by-side comparisons among PageRank (and its extensions) and other metrics [60, 61, 62]. In particular, [63] focuses on comparing HITS, PageRank and SALSA, and its authors prove that PageRank is the only metric that guarantees algorithmic stability with every graph topology.

2.3 Community detection

Even the community detection problem is not new in the domain of graph theory. The analysis of communities provides a deeper knowledge of the network's structure and the correlation between nodes, which allows the study of the information embedded into networks. Networks concerning healthcare, infection spread, human interactions, economics, transportation, trust and reputation are perfect examples where detecting communities

can help to understand the network's structure.

The definition of a community itself is, again, controversial. Intuitively, it can be defined as a set of entities that are close to others. This notion is quite similar to the concept of *closeness*, which is based on a similarity measure and is usually defined over a set of entities. One of the most acknowledged definitions of community appears in [64]. This definition has given birth to several algorithms for community detection [65] [66] which, for the most part, rely on the optimization of a *validation function* measuring the quality of the community structure. One of the most commonly used functions is the *modularity* function provided by Newman [64, 67]. Despite some limitations [68, 69], the modularity function has been successfully used as a quality measure to evaluate a given network partition and as a cost function to be optimized to uncover communities [70, 71, 65].

Other than modularity based methods, in literature there exists a lot of alternative approaches to solve the problem of community discovering [72, 73]. For example, in Ref. [74] an information-theoretic based method is presented. This method is based on the formulation of a new quality function called map equation[75], which allows to find the optimal description of the network by compressing its information flow. The algorithm is the core of *Infomap* (http://www.mapequation.org/), the search method for minimizing the map equation over possible network partitions.

Community detection has been successfully used to analyze the structure of single-layer networks and for modeling several kinds of interactions, such as social relationships, genetic interactions among biological molecules or trade among countries [76, 77, 78, 79, 80, 81, 82], just to mention a few (a detailed introduction to communities in networks can be found in [72, 73]).

Chapter 3

Microscale

There are two basic units that form a network graph, nodes and edges (or arcs, if the graph is directed). However, edges are merely used to model connections between nodes and, to a certain extent, they can be seen as node metadata. Given this thesis' focus on sociality, it is only natural to start the discussion from the point of view of nodes, the *microscale*: which neighbours should a node have? how does a node model its relationship with them? how do its neighbours impact its popularity in the network? This section will present the answer to these questions through theory, methodologies and simulations.

3.1 Local Weight Assignment

In a social network, the problem of assigning local weights consists in finding what weights should be given to a node's neighbours based on a certain set of metadata. The arc weights are an abstraction of the relationship between the two nodes, and should be based on social metadata in order to accurately model the quality of such relationship. As mentioned, it's difficult to talk about relationships without mentioning trust, and indeed many social networks are grounded on trust relationships. In a certain way, we

could say that social networks are a general case of trust networks.

A trust network is a network where the edges specifically model the level of trust that the two connected nodes share. In the past years, trust networks have grown in popularity as a fundamental precautionary component that helps users in managing virtual interactions with (possibly total) strangers, either real people or virtual entities, in several contexts such as e-commerce, social networks, distributed on-line services and many others [83, 84, 19].

Generally, in trust models and frameworks developed in the past years [28, 10, 29, 30], the trust network is represented as a graph where nodes are agents and trusting relationships (arcs) are weighted against a measure of the *direct* trust value according to a given metric. Sometimes trust networks may even be signed, because a negative weight may be associated to the lack of trust, or *distrust*, even though it's much harder to handle signed graphs due to algorithmic issues.

While most of the existing literature focuses on the assessment of *global* trust, i.e. the unique value for a node that aims at mediating all local (direct) values that express different judgements the node received from others, there is less emphasis on how *direct* trust should be evaluated for each node i.

In many scenarios, the successful, positive interactions between two nodes are taken into account in order to produce a trust value, which will be modeled by the arc weight connecting the two nodes. The well known proposals EigentTrust [32] and GossipTrust [34] compute direct trust following this principle. However, in order to provide a weight assignment model based on real world social criteria, there are two additional factors that need to be taken into account, the *mistrust* as a measure of a lack of trust, and the *popularity* of the node as the total number of received feedbacks. Mistrust may be used to

Symbol	Description
N	Number of neighbors
R	Total number of interactions among all neighbors
r_i	Interactions with neighbor i
r_i^+	Positive interactions experienced with neighbor i
r_i^-	Negative interactions experienced with neighbor i

Table 3.1: Notation in use

balance positive and negative feedbacks when selecting a node, and popularity measures to what extent a certain trust or mistrust rating is relevant. This way, a node which has more positive feedbacks than negative feedbacks it is considered overall trustworthy, and a node with more feedback scores is considered more trustworthy than a node with less feedback scores, if the overall trust rating is the same.

In the paper[2], the authors propose a direct trust assessment model, and prove, through a series of simulations, that it exhibits greater stability compared to EigentTrust and GossipTrust, i.e. if a node changes its behavior, trust and mistrust ratings are not significantly affected, unless this behavior repeatedly occurs, as the proposed model also takes into account the node's history (if a node has received hundreds of positive feedbacks, it takes more than a few incoming negative feedbacks for it to be considered untrustworthy).

3.1.1 Local Trust assessment

Notation and existing approaches

There are several possible intuitive approaches to assign weights to the arcs of a trust network. In this work, some of them will be analyzed for possible shortcomings. Before this, let's introduce a few definitions of the quantities involved for ease of notation, which are reported in Table 3.1. Alongside these definitions, let's introduce three new quantities:

$$p_i = \frac{r_i}{R}, \quad t_i = \frac{r_i^+}{r_i}, \quad m_i = \frac{r_i^-}{r_i} = 1 - t_i \tag{3.1}$$

p_i is the *popularity* of neighbor i seen from the perspective of a certain node n, t_i the *trust* that node n places in neighbor i and m_i the *distrust* that node n places in neighbor i. There are several works about the coexistence of trust and distrust in literature: in [85] propagation of trust and distrust is analyzed, whereas in [86] the authors manage to predict with acceptable accuracy whether a node is going to trust or distrust another node which is not connected to; finally, in [87], an extension of PageRank which works on signed graphs named PageTrust is introduced.

The approach described in [2] attempts to take into account both trust and distrust, incorporating them in a single weight assignment criterion. One of the most intuitive ways to model the node's attitude in a trust network is to give a *Positive Feedback*. It can be obtained by normalizing the positive interactions:

$$w_i^+ = \frac{r_i^+}{\sum_{k=1}^{N} r_k^+} \tag{3.2}$$

However, this solution does not make use of negative interactions. A node which experiences a negative interaction with a neighbor, should alter its attitude towards that neighbor accordingly. Intuitively, an agent wants to avoid more those nodes it had negative interactions with, and equation (3.2) does not take this aspect into account.

Another intuitive way to model the node attitude is to give *Net Feedback*, that tries to incorporate negative interactions into the weight equation, as done by EigenTrust. Naming $f_i^+ = max(0, r_i^+ - r_i^-)$, we have:

$$w_i^+ = \frac{f_i^+}{\sum_{k=1}^{N} f_k^+} \tag{3.3}$$

While this approach does include negative interactions in the arc weight, it has one important shortcoming: it reacts very poorly to feedback changes. Let us suppose we have a neighbour with 100 positive feedback ratings and 99 negative feedback ratings. With this equation, its f_i^+ would be 1. If our node completes another interaction with this neighbour positively, its f_i^+ will change to 2. This feedback gain essentially doubles the previous value, and this doesn't model well the mixed behavior of the node.

Social-based weight assignment

Taking note of issues described so far, the idea is to provide a weight equation that models the attitude according to these features:

- Neighbors with more interactions should be preferred. If two neighbors have the same ratio $\frac{r_i^+}{r_i}$, the one with the largest r_i should be more likely to be contacted.
- The higher negative/total interactions ratio a neighbor has, the more it should be avoided.
- The higher positive/total interactions ratio a neighbor has, the more it should be contacted.
- It should take into account previous interactions.

A node should weight both popularity and trust when making a decision about the trustworthiness of a certain node. A popular, trustworthy node is more likely to yield a successful interaction than a less popular node with the same level of trustworthiness. At the same time, we want to take into account the distrust of other nodes. This is because the more untrustworthy a node is, the more we want to avoid having interactions with it. This criterion to assign weights in a trust-based network, named *social weight*

assignment, sports two different components: the trust towards node i and the average distrust of all neighbors except i; both components are linear dependent on the neighbor popularity.

Definition and meanings

The social weight assignment criteria assigns the following weight to node neighbors:

$$w_i^+ = t_i p_i + \sum_{k=1,k\neq i}^{N} \frac{m_k p_k}{N-1} \tag{3.4}$$

where N is the number of neighbour nodes of node n. It is easy to prove that $\sum_{i=1}^{N} w_i^+ = 1$:

$$\sum_{i=1}^{N} w_i^+ = w_1^+ + w_2^+ + \ldots + w_N^+ = t_1 p_1 + \sum_{i=1,i\neq 1}^{N} \frac{m_i p_i}{N-1} + t_2 p_2 + \sum_{i=1,i\neq 2}^{N} \frac{m_i p_i}{N-1} + \ldots + \\ + t_N p_N + \sum_{i=1,i\neq N}^{N} \frac{m_i p_i}{N-1} = \sum_{i=1}^{N} t_i p_i + \sum_{j=1}^{N} \sum_{i=1,i\neq j}^{N} \frac{m_i p_i}{N-1} \tag{3.5}$$

Note that in the double sum each $m_i p_i$ is repeated $N-1$ times so we may reduce the notation as:

$$\sum_{i=1}^{N} t_i p_i \; + \; \sum_{i=1}^{N} m_i p_i \;\; = \;\; \sum_{i=1}^{N} (t_i \; + \; m_i) p_i \;\; = \;\; \sum_{i=1}^{N} \left(\frac{r_i^+}{r_i} + \frac{r_i^-}{r_i} \right) \frac{r_i}{R} \;\; = \;\; \sum_{i=1}^{N} \frac{r_i}{R} \;\; = \;\; 1 \tag{3.6}$$

This result allows to apply the social weight assignment in conjunction with metrics that require that the sum of the node outlink weights is 1, as all EigenTrust-based proposals. It is also possible to define a dual criterion with the same entities defined in (3.1), where t_i and m_i are swapped:

$$w_i^- = m_i p_i + \sum_{k=1,k\neq i}^{N} \frac{t_k p_k}{N-1} \tag{3.7}$$

These two criteria generate two different networks, a trust network for the social weight assignment, and a distrust network for its dual. These two networks are not complementary, in-fact, while they share the same topology, the arc weights are different: weights of the trust network cannot be derived directly from the weights of the distrust network, and vice-versa.

This implies that the node with highest weight in the trust network (the "best node") is not necessarily the node with the least weight in the distrust network. Another consequence of the way the weights are assigned is the fact that the most reliable node isn't necessarily the best node, as its trust weight is multiplied by p_i: the popularity of the node (which essentially means the portion of the node interactions shared with neighbour i) impacts greatly the final weight. This is by design, as the more interactions the node has with a neighbour, the more reliable the trust (or distrust) weight is: a node with 0.9 trust and 0.1 popularity has a lower weight than a node with 0.8 trust and 0.6 popularity in the trust network. This is in line with the behavior of human beings in a social context. An aspect that is interesting to expand upon a bit is how a neighbor is judged in trust and distrust networks. There are three possible situations:

- Trustworthy node: node has higher trust weight than distrust weight. These nodes are obviously the most reliable, especially if they have high popularity.
- Mixed node: node has similar trust and distrust weight. This can happen when the trust weight is near to the distrust weight.
- Non-trustworthy node: node has higher distrust

  weight than trust weight. These nodes should be avoided, regardless of popularity.

It is clear that a node should not react in a symmetric fashion to trust and distrust:

trust is easily shaken, but hard to build up. Future work may define a behavioral pattern for nodes using this weight assignment criteria. In conclusion, the two criteria provide the end-user with different information, which can be used together to make informed decisions about which nodes to trust, and which nodes to avoid.

3.1.2 Modelling the aging of interactions

While the social weight assignment equation (3.4) attempts to model social behavior, it is purely atemporal in meaning, as it doesn't keep track of the age of the interactions. In the realm of social interactions, older experiences and memories tend to have a reduced degree of impact on our behavior towards a certain person. Indeed, older memories get weaker over time, and their decline in strength has been the object of research for mathematical psychologists up to this day [88]. In particular, they strive to bind the memory detention decline to a mathematical function usually called *forgetting curve*.

The aging function proposed in this paper loosely takes inspiration after the forgetting curve modeling effort that has been pursued by researchers. This aging function aims at evaluating the contribution of experiences that have a certain age to the arc weight.

The aging function candidates $a(T)$ should manifest three properties:

- $a : \mathbb{R}_{\geq 0} \to (0, 1]$
- $a(0) = 1$
- monotonically decreasing

Given that the aging function is time-continuous, the age of a certain interaction that occurred at time t_0 can be written as $T = t - t_0$. Finally, if we wanted to know the weight contribution of a specific age T, we would have to calculate the value of $a(T)$. If we name

T_{ij}^+ (T_{ij}^-) the age of the j-th positive (negative) interaction that occurred with neighbor i we may define the following aging parameters:

$$A_i^+ = \sum_{j=1}^{r_i^+} a(T_{ij}^+) \quad A_i^- = \sum_{j=1}^{r_i^-} a(T_{ij}^-) \quad A_i = A_i^+ + A_i^-$$

that can be used to modify (3.1):

$$p_i' = \frac{A_i}{\sum_{i=1}^{N} A_i}, \quad t_i' = \frac{A_i^+}{A_i}, \quad m_i' = \frac{A_i^-}{A_i}, \tag{3.8}$$

The values in (3.8) can then be used in (3.4) to compute the weight of the arc towards neighbor i.

Candidates

We propose two candidate families of functions that manifest the aforementioned properties and could model the aging functions: exponential functions $e(t) = e^{-\beta t}$ and power functions $p(t) = (1 + \alpha t)^{-\beta}$, with $\alpha, \beta \in \mathbb{R}_{>0}$. They both behave similarilly, except that the exponential functions decrease much more quickly after a certain $\bar{t}$, which depends on the choice of parameters. The candidate aging functions as they are have an infimum of 0, as their limit for $t \to +\infty$ is 0. This means that the strength of each interaction fades until it stops to be a significant contribute to the arc weight. This might not be desired in all application scenarios, as older interactions essentially get ignored after a certain amount of time. To prevent this behavior, we changed the candidates in the following way:

$$e(t) = e^{-\beta \cdot \min(t,\gamma)} \quad p(t) = [1 + \alpha \cdot \min(t, \gamma)]^{-\beta} \tag{3.9}$$

Where $\gamma \in \mathbb{R}_{>0}$. This way we have that both candidates have an infimum greater than zero, meaning that after a certain amount of time γ the interactions "stop aging" and keep contributing to the arc weight with a constant value.

Implementation

Regrettably, the implementation of the aging functions is not a trivial task. The main issue comes from the difficulty in finding an appropriate time representation. As said before, the aging functions candidates are time-continuous. This makes them unfeasible candidates as they are, since all software or hardware clocks are time-discrete. The first challenge is then finding a correct quantization step. Intuitively, it should be small enough so that each interaction would lie in a separate time slot. Unfortunately, there's no easy way to predict the expected time of arrival of each interaction in real-world scenarios. In order to solve the issue it's better to discard the idea of a timestamp: each node would now evaluate the passage of the time based on the number of interactions. This virtualization of the notion of time based on the number of past interactions means that having an age T would imply being "T interactions old". This approach also avoids the issue of time synchronization among nodes, which would require a significant effort.
Of course this makes the time completely virtual, so it is not possible to assess the timestamp of a specific interaction among two peers at network level. This can be troublesome e.g. if we want to analyze the network behaviour at a certain time by reading which interactions happened at that time. Due to the virutalization, it's only possible ensure that interactions are accounted for in the proper order, there's no possibility of using a reference time, a sort of wallclock. This is acceptable, however, since social weight assignment criterion operates locally, and is unaware of everything that happens beyond

the line of sight of each node.

In order to correctly implement aging two challenges need to be overcome, updating the age of all interactions whenever a new interaction occurs, and computing the age parameters A_i, A_i^+ and A_i^- to be used in (3.8). With some bit-level manipulation is fairly easy to design a storage-efficient solution to both problems. For each neighbor i, we need to allocate four entities:

- A γ-bit binary register b_i^+ for positive interactions
- A γ-bit binary register b_i^- for negative interactions
- A counter c_i^+ for positive interactions having an age greater than γ
- A counter c_i^- for negative interactions having an age greater than γ

A set bit in position k in registers b_i^+ and b_i^- means that there is an interaction with neighbor i of age k. This way the age increses along with the bit position, so that the MSB is the oldest interaction that can be stored in the register, which always has age $\gamma - 1$. The counters c_i^+ and c_i^- are used to keep count of the interactions which age is equal or greater to γ. Updating the interaction age is then a matter of using appropriate bit-manipulation operators as described in Algorithm 1. We can see that increasing the age of all interactions is done by applying a bit-shift operator on the whole register (lines 5 and 9). However, this destroys the content of the MSB, so we need to transfer its content to the appropriate counter prior to the operation. This is done in lines 2-3 and 6-8. Note that this needs to be done for each neighbor (line 1), even those who don't take part in the current interaction, as age is based on the number of interactions that occur among all neighbors. After the age is updated, we simply check whenever the current interaction is successful, and set the LSB, which has an age of 0, of the counter tied to

the neighbor i (lines 11-15). Algorithm 1 is used to keep track of the virtual time, but it

Algorithm 1 Age update after interaction with neighbor i

```
1: for j ← 1...N do
2:     if b_j^+ & (1 ≪ γ − 1) then
3:         c_j^+ ← c_j^+ + 1
4:     end if
5:     b_j^+ ← b_j^+ ≪ 1
6:     if b_j^− & (1 ≪ γ − 1) then
7:         c_j^− ← c_j^− + 1
8:     end if
9:     b_j^− ← b_j^− ≪ 1
10: end for
11: if interaction is successful then
12:     b_i^+ ← b_i^+ | 1
13: else
14:     b_i^− ← b_i^− | 1
15: end if
```

is also necessary to calculate the three age parameters A_i, A_i^+ and A_i^-. This is described in Algorithm 2. The value in counters c_i^+ and c_i^- stands for the number of interactions that have an age greater than or equal to γ, so it's possible to initialize A_i^+ and A_i^- employing the appropriate counter, like in lines 1-2. Estimating the contribute of fresher interactions require checking each bit of registers b_i^+ and b_i^-: if a bit in position k is set, its contribute to the corresponding register is equal to $a(k)$ (lines 3-10).

Algorithm 2 Computation of A_i, A_i^+ and A_i^-

```
1: A_i^+ ← c_i^+ · a(γ)
2: A_i^− ← c_i^− · a(γ)
3: for T ← 0...γ − 1 do
4:     if b_i^+ & (1 ≪ T) then
5:         A_i^+ ← A_i^+ + a(T)
6:     end if
7:     if b_i^− & (1 ≪ T) then
8:         A_i^− ← A_i^− + a(T)
9:     end if
10: end for
11: A_i ← A_i^+ + A_i^−
```

3.1.3 Simulation

In this subsection, we present a set of simulations that compare the behavior of several weight assignment criteria. The results show that the social weight assignment is resilient to changes, and that it is sensitive to nodes which behave in a different way from the others. We also experimented with the two aging function candidates (3.9) to see which function affects more arc weights evaluation.

The devised simulation scenarios share a common set of rules, related to the weight assignment strategy described by equation (3.4). The networks in which weight assignment techniques are tested are either scale-free or modeled after Erdős-Rényi graphs. They are generated using the software Pajek [89] using the parameters in Table 3.2.

Erdős-Rényi network parameters	
Number of vertices	2000
Min number of arcs for vertex	5
Max number of arcs for vertex	15
Scale-free network parameters	
Number of vertices	2000
Number of line	No constraint
Average degree of vertices	10
Number of vertices in initial ER network	10
Initial probability of lines	0.2
α	0.25
β	0

Table 3.2: Network parameters

After the networks are generated, their evolution is simulated. We divided the simulations in several cycles. For each cycle, each node simulates an interaction with each of its neighbors. This interaction can have a positive, or negative outcome, which is permanently recorded by the node as a positive or negative experience.

The outcome of a transfer is determined at random according to an uniform distribution. The distribution itself depends on the node receiving the interaction, which can be a

"good" or a "bad" node. Whether a node is "good" or "bad" is determined at the beginning of the simulation, according to these simple rules:

- 95% of the nodes have at most a 20% chance of unsuccessfully replying to a transfer request ("good" nodes).
- 5% of the nodes have at least 80% chance of unsuccessfully replying to a transfer request ("bad" nodes).
- The bad nodes are picked at random following a uniform distribution.

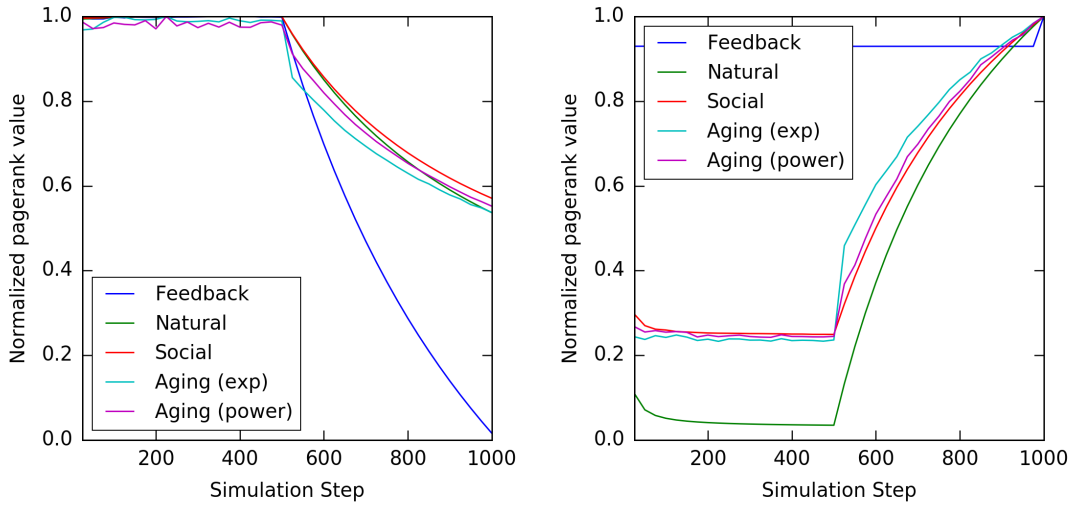
In each simulation, PageRank for a specific node, the "monitored node" is computed, and changes in its PageRank value cycle after cycle are monitored. Once a node is assigned a certain behavior, it does not change it during the course of the simulation. An exception to this is the *monitored node* described below. Also, nodes do not have "favourite" neighbors: each neighbor asking for an interaction is treated equally by the receiving node.

Computing PageRank on the network where the social weight assignment was applied is always possible since the equations guarantee that the outstrength of each node is exactly 1, making the adjacency matrix row-stochastic. The monitored node is always the node which has the highest "topological PageRank", that is, the PageRank calculated when all arcs in the network have unitary weight. The simulation graphs always show this PageRank for each cycle. As already mentioned, the monitored node does not follow the behavioral rules described above.

Finally, the aging functions employed in the simulation are the two candidates defined in (3.9). The parameters used in the simulation for the candidates are shown in in Table 3.3.

	α	β	γ
Exponential	N/A	0.065	64
Power	1	1	64

Table 3.3: Aging functions candidates parameters



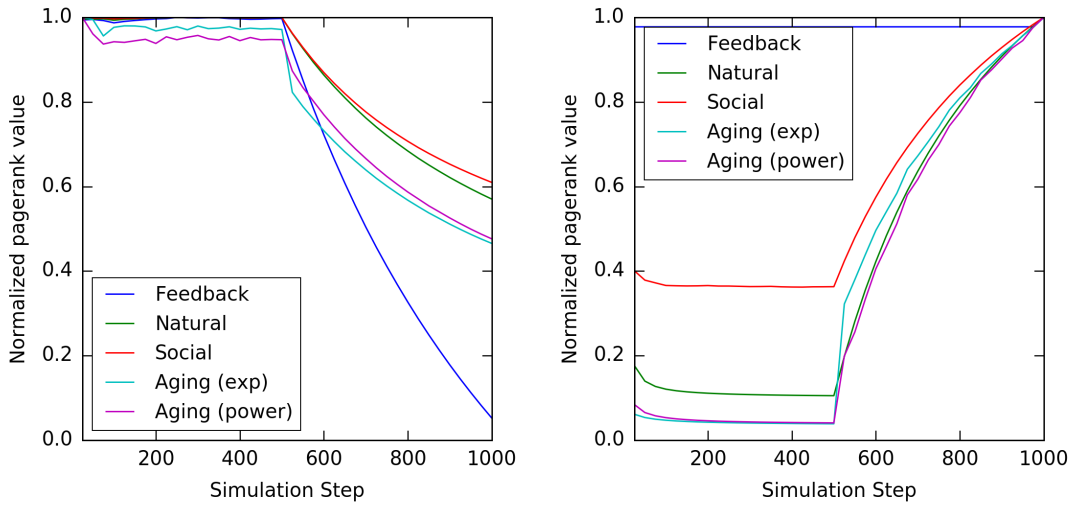
(a) Good to bad

(b) Bad to good

Figure 3.1: Comparison among different weight assignment techniques in a network where some nodes behave poorly.

The goal of the simulations is to compare the behavior of the Social Weight Assignment against the other techniques described in subsection 3.1.1, i.e. the Positive Feedback approach, and the Net Feedback approach, and to establish the effect of aging applied to our Social Weight Assignment. In these simulations, the behavior of the different weight assignment techniques for a 2000 nodes scale-free network is displayed. The normalized PageRank value of the monitored node is plotted against the simulation steps, and the results for all weight assignment criteria are shown in Figure 3.1. The normalized value is used to show the relative difference among the techniques, as the absolute PageRank value by itself is meaningless.

The net feedback approach (blue line) drops much faster in simulation 1a compared to the other weight assignment criteria, and behaves quite poorly in simulation 1b. The behavior of Social (red line) and Natural (green line) Feedback approaches is quite similar, but the



(a) Good to bad

(b) Bad to good

Figure 3.2: Comparison among different weight assignment techniques in a network where all nodes behave properly.

Social approach reacts more slowly to changes. Due to this difference of slope, the two curves cross at a certain simulation step.

It is interesting to highlight the trend of the two aging functions (exponential, in teal, and power, in magenta). They highlight the presence of a *shock* when the node changes behavior, as its PageRank value drops (1a) or raises (1b) immediately. This is due to how the aging model works, as newer experiences can weight several orders of magnitude more than older experiences: this ensures that the model reacts very fast to sudden behavior shifts. While being a consequence of the mathematical model employed, this effect is coherent with the social aspect of the proposed criterion, as it reflects the surprise factor of a person which faces unexpected behavior from a peer they had a certain idea of.

After the shock, the two curves' slopes are comparable to the slope of the non-aging social weight assignment curve. The exponential aging function has a greater effect on the shock phenomenon compared to the power-based aging function.

Simulation 2 has same setup of simulation 1 but the error rate has been set to 0 (that means that all nodes always behave properly). The overall trend of the data analyzed

in Simulation 2 is similar to the trend of data in simulation 1a and 1b, but the slope of the social approach is steeper. This is because the Social weight assignment tends to highlight neighbors which behavior differs from the average. The two curves about the aging effect show no different behavior compared to simulation 1, so the same things can be said concerning their trend.

In conclusion, the simulations show that the social weight assignment criteria is more stable compared to the presented techniques. Moreover, it successfully reacts to sudden behavior changes (shocks) if we add the aging paramenter to the weight assignment. No other difference in terms of behavior are present between the two curves employing the aging function candidates, besides the already mentioned slope steepness.

3.2 Best Attachment

Ordering nodes by rank is a benchmark used in several contexts, from recommendation-based trust networks to e-commerce, search engines and websites ranking. During the past years it emerged in several scenarios, from trust-based recommendation networks [48, 49] to website relevance score in search engines [39, 40], e-commerce B2C and C2C transactions [50, 51]. Within each specific framework, different proposals exist about the meaning of nodes (agents, peer, users) and about the rank assessment algorithm; in all of them, the higher is the rank of a node, the higher is its legitimacy. The rank depends on the set of out-links each node establishes with others [90], and on the set of in-links it receives.

In the mentioned scenarios, the node rank depends on the set of links the node establishes, hence it becomes important to choose appropriately the nodes to connect to. The problem of finding which nodes to connect to in order to achieve the best possible rank

is known as the *best attachment problem*. Given that the network is usually modeled as a directed graph $G(V, E)$, finding the k best attachments for a given node $i \in V$ consists in finding a set $S \subseteq S_{i,k}$ that maximizes the rank P_i by changing its in_i to $in_i \cup S$, where $in_i = \{v \in V : \exists e(v, i) \in E\}$ and $S_{i,k} = \{S \in V : |S| = k, S \cap in_i = \emptyset\}$. Essentially, we need to establish k links from the k nodes that will improve i's rank value up to the highest [91]. Intuitively, we may think that if we select the first k nodes in the ordered PageRank vector we would reach the optimal solution to the problem, but this is not usually the case. The PageRank algorithm is driven by the node backlinks, not forward links. This means that even if i connects to an highly ranked node, unless that node points towards i as well, it is not guaranteed that its Pagerank is positively affected: depending on the network topology, it may even be possible that its Pagerank can decrease. In the general case, the best attachment problem is NP-hard, because the only possible way to predict the new ranking after the node attachment is to calculate $\binom{k}{n}$ PageRanks. More accurately, the problem is actually $W[1]$-hard, as analytically demostrated in [92], which makes it unfeasible to compute the optimal solution in real-life scenarios, as even with small networks we would need to compute the Pagerank millions of times. Because of these computational issues, it is necessary to find an approximation algorithm to choose a solution acceptably close to the optimum in a polynomial time. In [92] however authors also show that there exist both upper and lower bounds for certain classes of heuristics. It is not always possible to calculate these bounds as sometimes the computational complexity of these calculations is NP-hard as well, nevertheless finding bounds of heuristics lets us rank their theoretical accuracy.

In [3], the authors propose several heuristics that aim at providing a near-optimal solution in reasonable time, preserving effectiveness while achieving practical feasibility.

They applied the proposed heuristics to different syntetic networks by simulation, and compared the results. In previous studies [27, 93, 25], the authors attempted a brute-force solution to the best attachment problem, and analyzed the cost of link building and its dynamic. There are however more works in literature concerning the best attachment problem. In [94], the authors use asymptotic analysis to see how a page can control its pagerank by creating new links. A generalization of this strategy to websites with multiple pages is described in [95]. In [96] the authors model the link building problem by using constrained Markov decision processes. In [97] the author demonstrates that by appropriately changing node outlinks the resulting PageRank can be dramatically changed.

3.2.1 Heuristics

As discussed, the problem to find the best set of nodes allowing us to gain the best reputation is not feasible due to computability complexity, therefore a more empirical approach through several heuristics is attempted to approximate the solution. These heuristics have both pros and cons, which will be discussed in this section.

The main goal of the heuristics is to allow a new node (called me), to find a trade-off between the minimization of steps, the cost of new links creration, and the rank position. Note that the cost of creating a link is both the computational effort needed to evaluate the increasing of rank (if any) and the cost needed to prevail on a node ($x \in V$) to create a link with me.

The computational complexity of all stategies depends on the computational complexity of pagerank evaluation, in the following called $\mathcal{O}(PR)$. Using the Gauss method it would require $\mathcal{O}(|V|^3)$, however using iterative approximation it would require $\mathcal{O}(PR) = m*|E|$,

where m is the number of iterations needed to get a good approximation [98].

In the following subsections some heuristiscs based both on naive approach are and on PageRank evaluation are presented, aiming at reducing the complexity in solving the best attachment problem.

Naive Algorithms

The simplest approach to get best attachment problem consists in selecting the nodes k to populate S randomly until we reach the target . This strategy is naive but it is simple to implement, and the cost to create link is proportianal to number of links only. It also serves as a benchmark for computational complexity comparisons, since its complexity only depends on the number of steps me needs to get the best position, i.e. $\mathcal{O}(random) = \mathcal{O}(m)$ where m is the number of steps to converge. The algorithm is detailed in Algorithm 3.

Algorithm 3 Random Choice Algorithm

```
1: procedure RANDOM(V, me)          ▷ V is the set of vertices, me is the target node
2:     T = V
3:     S = ∅
4:     while rank_me > 1 do          ▷ Iterate until rank is the best
5:         random_select x ∈ T
6:         T = T − {x}
7:         S = S ∪ {x}
8:     end while
9: return S
10: end procedure
```

Of course *Random* strategy is trivial and does not rely on any of the network properties. However when the topology is almost regular and the distribution of both in- and out-degree- is also regular, this algorithm performs as well as others more complex.

Another simple approach to find a node to be pointed by comes from the degree of target node me, so we can use in-, out- or full-degree of the target node as a selection criteria.

The algorithm is reported in Algorithm 4.

Algorithm 4 Degree Algorithm

```
1: procedure DEGREE(V, me)          ▷ V is the set of vertices, me is the target node
2:     T = V
3:     S = ∅
4:     while rank_me > 1 do          ▷ Iterate until rank is the best
5:         select x ∈ T, where degree(x) is max          ▷ x node with highest degree in T
6:         T = T − {x}
7:         S = S ∪ {x}
8:     end while
9: return S
10: end procedure
```

Note that the complexity of algorithm is $\mathcal{O}(Degree) = \mathcal{O}(m) * \mathcal{O}(select)$, where m is the number of steps and $\mathcal{O}(select)$ is the complexity to find the node having maximum in-, out- or full-degree. This last term depends on the type of data structure used to store T.

Pagerank Based Algorithms

While *Random* approach - being a trivial one - uses no information about network topology and nodes characteristics, the next strategies aim at overcoming this limit using information about the ranking of nodes increasing as little as possible the computational complexity. Since higher rank nodes are the most popular inside the networks we select the in-link node according to its reputation. However, a change of topology due to the new connection could affect the ranking of the nodes, therefore different strategies can be outlined, depending on the how frequently ranking is evaluated, as reported below.

1. *Anticipated Rank* strategy: the rank is calculated just before starting the search and used throughout the whole algorithm to select the (not used) in-link node. Based on the way the ranking is computed we can distinguish two algorithms:

    - *Anticipated_value* (Algorithm 5): the ranking is calculated by ordering nodes

according to the node's pagerank value; the idea here is to select first nodes with higher pagerank value.

- *Anticipated_outdeg* (Algorithm 6): the ranking is computed by ordering nodes according the ratio between node's pagerank value and node's out degree; in this approach, we first select the node that *"transfers"* the highest pagerank value to its out-neighbourhood.

2. *Current Rank* strategy (Algorithm 7): the rank is recalculated each iteration and the best not used node is selected. We always use current rank.

3. *Future Rank* strategy (Algorithm 8): algorithm evaluates all possible connection - one step behind - and select the connection giving *me* the best rank. This strategy should be the more effective but it is the more expensive.

Algorithm 5 Anticipated_value Algorithm

```
1: procedure ANTICIPATED_VALUE(V, me)        ▷ V is the set of vertices, me is the target node
2:     S = ∅
3:     R = pagerank                          ▷ R is the set of n|n ∈ V ordered according to their pagerank value
4:     repeat
5:         x = first(R)|x ∉ S
6:         R = R − {x}
7:         S = S ∪ {x}
8:     until (rank_me = 1)                   ▷ Node will always get best position before R became empty
9: return S
10: end procedure
```

The complexity of the strategy *Anticipated_value* and *Anticipated_outdeg* is almost the same of *Degree* strategy, in fact PageRank is evaluated only once before starting the iteration. However the more nodes are selected, the more often network topology changes therefore the results of initial PageRank evaluation become less accurate.

Current is the second strategy based on pagerank proposed in [3]; it selects the node according to its rank, but re-evalute pagerank at each iteration.

Algorithm 6 Anticipated_outdeg Algorithm

1: **procedure** ANTICIPATED_OUTDEG(V, me) ▷ V is the set of vertices, me is the target node
2: $S = \emptyset$
3: $R = pagerank/out\_degree$ ▷ R is the set of $n|n \in V$ ordered according to the ratio pagerank / degree
4: **repeat**
5: $x = first(R)|x \notin S$
6: $R = R - \{x\}$
7: $S = S \cup \{x\}$
8: **until** $(rank_{me} = 1)$ ▷ Node will always get best position before R became empty
9: **return** S
10: **end procedure**

Algorithm 7 Current Algorithm

1: **procedure** CURRENT(V, me) ▷ V is the set of vertices, me is the target node
2: $S = \emptyset$
3: **repeat**
4: compute pagerank
5: select $x \in V|x \notin S$ where pagerank is max
6: $S = S \cup \{x\}$
7: $E = E \cup (x, me)$ ▷ Connect x to me
8: **until** $(rank_{me} = 1)$
9: **return** S
10: **end procedure**

The complexity of this algorithm is $\mathcal{O}(current) = \mathcal{O}(m) * max[\mathcal{O}(select), \mathcal{O}(PR)]$ and it is quite higher than all previous algorithms, since generally $\mathcal{O}(PR)$ is higher than $\mathcal{O}(select)$. However, this algorithm seems to capture the network dynamics due to creation of new links better than previous ones.

Future is the last algorithm proposed in [3]; it tries to evaluate the best node to be pointed by via evaluating the PageRank that will be obtained by me after the arc creation. This algorithm needs a continue re-evaluation of pagerank and its complexity is $\mathcal{O}(future) = \mathcal{O}(m) * max[\mathcal{O}(select), \mathcal{O}(PR) * \mathcal{O}(|V|)]$, that can be rewritten as: $\mathcal{O}(future) = \mathcal{O}(m) * \mathcal{O}(PR) * \mathcal{O}(|V|)$

At first glance, this algorithm seems to be optimal since it selects the node which gives the best rank, but its complexity is higher than all previous algorithms. To partially overcome this problem, a simpler heuristic that does not iterate over all nodes but selects

Algorithm 8 Future Algorithm

```
1: procedure FUTURE(V, me)                    ▷ V is the set of vertices, me is the target node
2:     S = ∅
3:     repeat
4:         R = ∅
5:         repeat
6:             connect node x ∈ V|(x ∉ R) and (x ∉ S) to node me
7:             calculate pagerank
8:             R = R ∪ {(x, rank^x_me)}
9:             disconnect x from me
10:        until |R| = |V|                    ▷ Iterate over all nodes belonging to V
11:        select (x, rank^x_me) ∈ R where rank^x_me is max
12:        S = S ∪ {x}
13:        E = E ∪ (x, me)                    ▷ Connect x to me
14:    until (rank_me = 1)
15: return S
16: end procedure
```

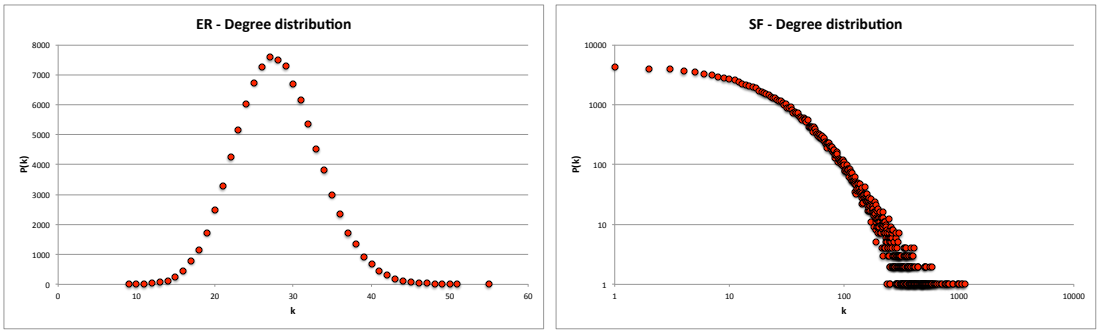
randomly N nodes to which apply the *Future Algorithm* approach can be employed.

3.2.2 Results

To study the performance of the heuristics proposed in the previous subsection, a set of experiments on two well-known family of networks were conducted: the Erdos–Renyi random networks (ER) and scale–free (SF) networks.

A random ER network is generated by connecting nodes with a given probability p. The obtained network exhibit a normal degree distribution [99]. A scale–free network (SF) [100] is a network whose degree distribution follows a power law, i.e. the fraction $P(k)$ of nodes having degree k goes as $P(k) \sim k^{-\gamma}$, where γ is typically in the range $2 < \gamma < 3$. A scale–free network is characterized by the presence of *hub* nodes, i.e. with a degree that is much higher than the average. The scale-free network employed in this work is generated by using the algorithm proposed in [101] as implemented in the Pajek[89] tool.

Simulations have been performed by using 100k nodes networks of both topologies. Ta-



(a) Erdos–Renyi (b) Scale–free

Figure 3.3: Degree distribution for ER and SF networks

ble 3.4 reports the main topological properties of such networks.

Table 3.4: Networks topological parameters

name	**#nodes**	**#links**	**average degree**
ER	100000	1401447	28.028
SF	100000	1394248	27.884

In figure 3.3 the degree distributions of 100K nodes networks is shown. As expected, the ER network (figure 3.3a) exhibits a normal degree distribution, while the SF degree distribution (figure 3.3b) follows a power—law.

Figure 3.4 reports the rank of *me* with respect to the number of in-links and steps for all the algorithms presented in the previous subsection. The best rank is represented by the position 1, so at the beginning *me* has the worst rank, i.e. 100001. The figure shows that *Degree_out* and *Degree_full* are the worst algorithms in terms of performance. *Random*, *Degree_in*, *Anticipated_value* and *Current* exhibits comparable performance. Despite the fact *Random* has the lower computational complexity among the proposed algorithms, it performs as well as more complex algorithms. *Future* and *Anticipated_outdeg* are the best algorithms when applied to ER networks. Surprisingly *Anticipated_outdeg* performs even better than *Future*, mainly during the initial part of the attachment process. In fact, as detailed in the figure inset, *Anticipated_outdeg* permits *me* to rapidly achieve a good rank, even if Future outperforms it after the step number 10. Let's note, however,

that the computational complexity of *Anticipated_outdeg* is far less than *Future*, making the application to very large networks feasible.

In the figure 3.5 the simulation results for SF network is shown. It's clear that *Random*, *Anticipated_outdeg* and *Future* outperform the other algorithms proposed in the previous subsection. As in the case of ER networks, the performance of the *Anticipated_outdeg* algorithm is surprising since its trend is comparable to the more complex algorithm *Future*. In addition, *Anticipated_outdeg* allows the node *me* to reach the best rank in only 18 steps against the 36 required by *Future*. On the other hand, *Random* algorithm gets the best rank in 69 steps, that is a very good figure considering the size of the network (100K nodes) and the random selection strategy. This behaviour is probably due to the presence in SF networks of hubs and authority, that play a central role on the dynamic underlying the PageRank evaluation.

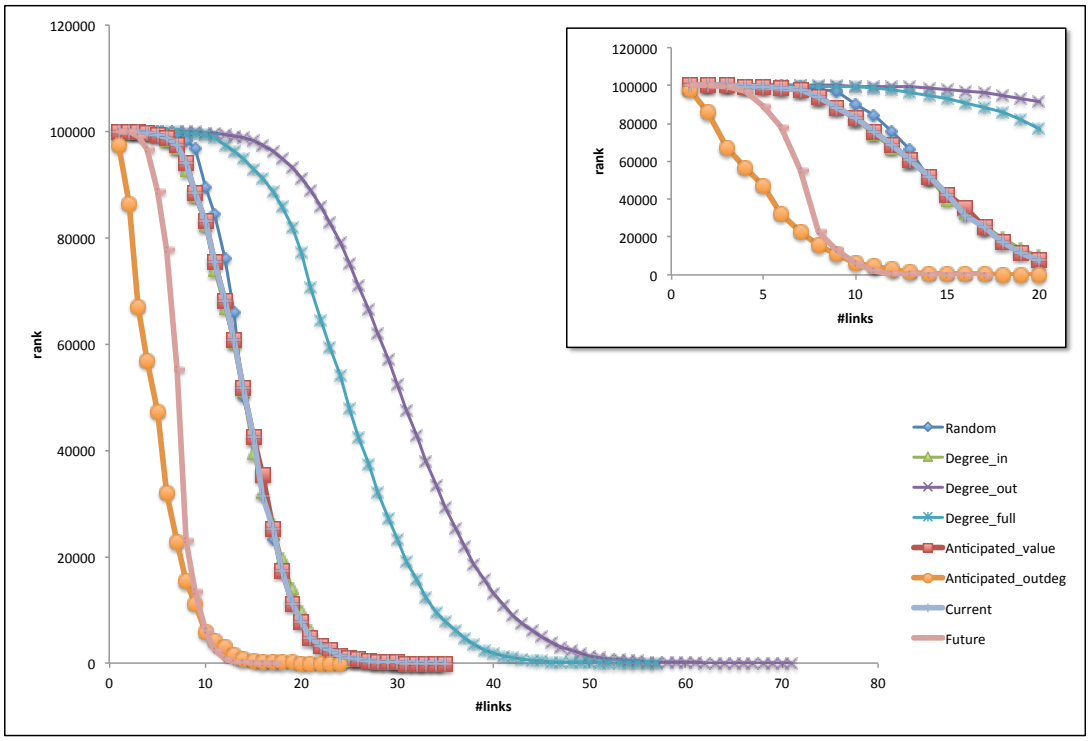


Figure 3.4: Heuristics performance on ER network with 100K nodes

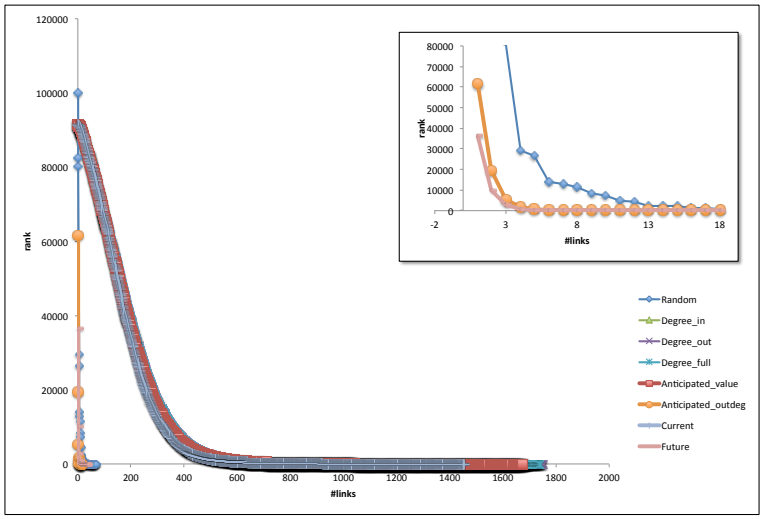


Figure 3.5: Heuristics performance on SF network with 100K nodes

3.3 Black Hole Metric

In 3.2's preamble, some notions concerning node ranking were given in order to describe the problem of *best attachment*. Particular emphasis was given to the PageRank[52, 53] algorithm, which was used as a basis to devise the various heuristics. In this section, an extension of PageRank named *Black Hole Metric* [4] will be presented, which aims at solving some outstanding issues of the 18-years old metric.

In the vast amount of digital data, humans have the need to discriminate those relevant for their purposes to effectively transform them into useful information, which usefulness depends on the scenario being considered. For instance, in web searching we aim at finding significant pages with respect to an issued query [38], in an E-learning context we look for useful resources within a given topic [41, 42, 43], or in a recommendation network we search for most reliable entities to interact with [44, 45, 46, 47]. All these situations fall under the umbrella of *ranking*, a challenge addressed in these years through different solutions. The most well-known technique is probably the PageRank algorithm [52, 53], originally designed to be the core of the Google (*www.google.com*) web search engine. Since it was published it has been analyzed [102, 98, 103], modified or extended for use in

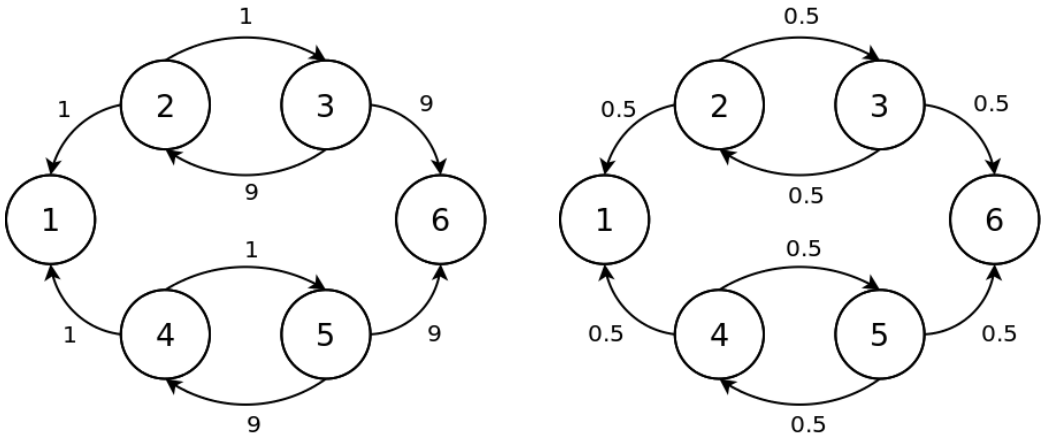
other contexts [104, 54], to overcome some of its limitations, and to address computational issues [105, 106].

PageRank has been widely adopted in several different application scenarios. In many of them though, modificiations to the algorithm are applied in order to adapt PageRank to the specific scenario. The *Black Hole Metric* is a generalization of PageRank whose motivation stems from the concept of trust in virtual social networks. In this context trust is generally intended as a measure of the assured reliance on a specific feature of someone [28, 7, 31], and it is exploited to rank participants in order to discover the *best* entities that is "safe" to interact with. This trust-based ranking approach allows to cope with uncertainty and risks [13], a feature especially relevant in the case of lack of bodily presence of counterparts.

A notable limitation of PageRank when it's used to model social behaviour, is its inability to preserve the absolute arc weights due to the normalization introduced by the application of the random walker. In order to illustrate the problem, we introduce a weighted network where arcs model relationships among entities. Entities may be persons, online shops, computers that in general need to establish relationships with other entities of the same type. Let's suppose to have the network shown in Figure 3.6a, where each arc weight ranges over $[0, 10]$.

Given the network topology, intuition suggests that node 1 would be regarded more poorly compared to node 6 since it receives lower trust values from his neighbors, but, as detailed later, normalizing the weights alters the network topology so much that both nodes are placed in the same position in the ranking. The normalization of the outlink weights indeed hides the weight distribution asymmetry, as depicted in Figure 3.6b.

Moreover, PageRank shadows the social implications of assigning low weights to all of



(a) Before the normalization (b) After the normalization

Figure 3.6: Network with asymmetric trust distribution

a node's neighbours. If we consider the arcs as if they were social links, common sense would tell us to avoid links with low weight, as they usually model worse relationships. If we look at the normalized weights in Figure 3.6b though, we can see that in many cases, the normalized weight changes the relationship in a counter-intuitive way. Consider the arcs going from node 2 or 4 to their neighbours: we can see that their normalized weights are set to 0.5, which, in the range $[0, 1]$ is an average score. However, the original weight of those links was 1, a comparatively lower score considering that the original range was $[0, 10]$.

The *Black Hole Metric* copes with the normalization effect and deals with the issue of the skewed arc weights, detalied in subsection 3.3.2. Note that the metric seamlessly adapts to any situation where PageRank can be used, as it's not limited to trust networks; in the following, they are considered as a simple case study.

3.3.1 PageRank

Definitions and Notation

In order to better understand the mathematics of the Black Hole Metric, let's to clarify the notation and provide a few definitions, which are similar to the notations used in the article of PageRank. Let us suppose that N is the number of nodes in the network. We will call A the $N \times N$ *network adjacency matrix* or *link matrix*, where each a_{ij} is the weight of the arc going from node i to node j. S is the $N \times 1$ *sink vector*, defined as:

$$s_i = \begin{cases} 1 & \text{if } out_i = 0 \\ 0 & \text{otherwise} \end{cases} \quad \forall i \leq N$$

where out_i is the number of outlinks of node i. V is the *personalization vector* of size $1 \times N$, equal to the transposed initial distribution probability vector in the Markov chain model P_0^{T}. While this vector can be arbitrarily chosen as long as it's stochastic, a common choice is to make each term equal to $1/N$. $T = \mathbf{1}_{N\times 1}$ is the *teleportation vector*, where the notation $\mathbf{1}_{N\times M}$ stands for a $N \times M$ matrix where each element is 1.

In the general case the Markov chain built upon the network graph is not always ergodic, so it is not used directly for the calculation of the steady state random walker probabilities. As described in [53], the *transition matrix* M, used in the associated random walker problem, is derived from the link matrix, the sinks vector, the teleportation vector and the personalization vector defined above:

$$M = d(A + SV) + (1 - d)\, TV \tag{3.10}$$

where $d \in [0, 1]$ is called *damping factor* and it is commonly set to 0.85. As we know from

the Markov chain theory, the random walk probability vector at step n can be calculated as:

$$P_n = M^{\mathsf{T}} P_{n-1} \tag{3.11}$$

the related random walker problem can be calculated as:

$$P = \left(\lim_{n\to\infty} M^n\right)^{\mathsf{T}} P_0 = \lim_{n\to\infty} (M^{\mathsf{T}})^n P_0 = M_\infty^{\mathsf{T}} P_0 \tag{3.12}$$

The normalization problem

Let's calculate the PageRank values of the sample network in Figure 3.6a to highlight the flattening effect of the normalization. By applying the definitions in subsection 3.3.1 the network in Figure 3.6b can be described by the following matrices and vectors:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad S = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad V = P_0^{\mathsf{T}} = \frac{1}{6} \cdot \mathbf{1}_{1\times 6} \quad T = \mathbf{1}_{6\times 1}$$

If we calculate the PageRank values for the nodes of the sample network assuming $d = 0.85$ we obtain:

$$p_1 = p_6 = 0.208 \quad p_2 = p_3 = p_4 = p_5 = 0.146$$

Note that the nodes 1 and 6 are both first in global ranking, despite the fact that their in-strength was so different before the normalization.

3.3.2 Black Hole Metric

As mentioned before, the *Black Hole Metric* avoids the flattening effect of the PageRank normalization. *Black Hole Metric* globally preserves the proportions among the arc weights, and ensures at the same time that the outstrength is equal to 1 for each node. This allows compatibility with the random walker model, and it is done by applying a transformation to the original network. The transformation only requires the knowledge of the maximum and the minimum value each weight can assume. This range bounds may be global (each node has the same scale) or local (each node has its own weight scale); in practice, global scale is preferred.

An example of the transformation steps as illustrated in Figure 3.7. In order to obtain the depicted values, formulas (3.13) and (3.16) were used, which will be explained in detail in paragraph 3.3.2, but for now, the transformation will be explained qualitatively. First, *Black Hole Metric* changes the original weights so that they lie in the range $[0, 1]$. The resulting outstrength s_i of each node i is not preserved, but it is guaranteed to be less or equal than 1. Then, it introduces a new node, the *black hole*, and a new arc connecting node i to the *black hole*. The strength of this connection is set to $1 - s_i$, as if the black hole "absorbed" the missing weight amount to reach 1 as the total i's outstrenght. This transformation is applied to all nodes in the network.

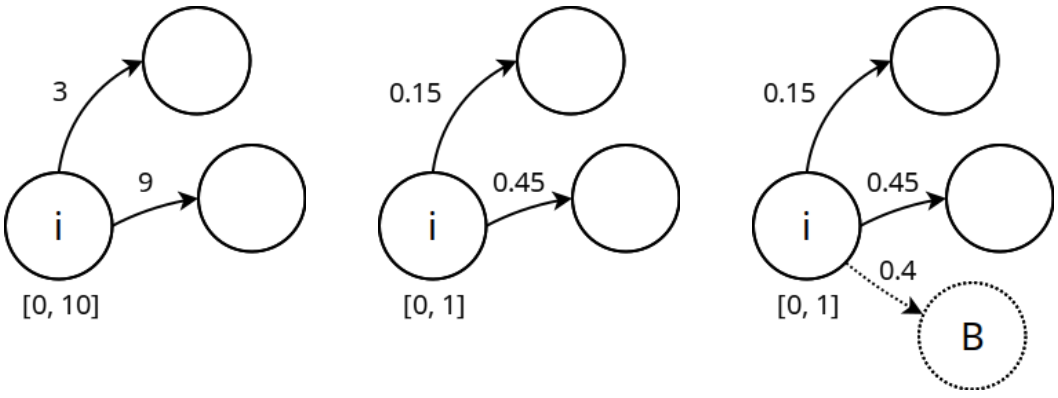


Figure 3.7: Transformation steps

Since the black hole does not have outlinks, it is a sink by construction, and the random walker can only move away because of the teleportation effect. In a network without the black hole, each node would normally have a $1 - d$ chance to teleport to a random node instead of going towards one of its neighbours. We know that moving to the black hole from node i occurs with a $1 - s_i$ chance, and that once in the black hole, the random walker inevitably teleports to a random node. In conclusion, taking both effects into account, each node has a $(1 - d)(1 - s_i)$ chance to teleport to a random node, where d is the damping factor as in (3.10).

It is important to note that not every network has a defined scale for its arc weights. There are networks in which the weights are *unbounded*: an example would be an airline transportation network in which each arc weight is the number of flights connecting two cities. As there is no real "maximum", there is no trivial weight scale that can be used for the transformation. In order to apply such transformation in an unbounded network, we need to somehow infer meaningful scale boundaries exploiting the knowledge of the domain, and the topology of the network. This is usually non-trivial, and for the rest of this section, only bounded networks will be taken into account.

Weight assignment

This subsection will detail how the new weights are calculated. Let i be a generic node in the network. Let the interval $[l_i, h_i]$ be the *local* scale of node i. Let r_{ij} be the weight that node i assigns to the arc pointing towards node j. Let out_i be the number of neighbours of node i. Given that $l_i \leq r_{ij} \leq h_i$, We define the modified weight $\bar{a}_{ij}$ of the arc that goes from i to j as:

$$\bar{a}_{ij} = \frac{r_{ij} - l_i}{out_i(h_i - l_i)} \tag{3.13}$$

which is significantly different from the normalized arc weight required by PageRank:

$$a_{ij} = \frac{r_{ij}}{\sum_{k=1}^{out_i} r_{ik}} \tag{3.14}$$

As mentioned before, the resulting node outstrength is only guaranteed to be less or equal to 1:

$$\sum_{j=1}^{out_i} \bar{a}_{ij} = \sum_{j=1}^{out_i} \frac{r_{ij} - l_i}{out_i(h_i - l_i)} \leq \sum_{j=1}^{out_i} \frac{1}{out_i} = 1 \tag{3.15}$$

We purposely excluded the contribute of the arc from node i to the black hole in (3.15), which is:

$$b_i = \sum_{j=1}^{out_i} \frac{h_i - r_{ij}}{out_i(h_i - l_i)} \tag{3.16}$$

If we include this contribute as well, the weight sum becomes 1 as desired:

$$\sum_{j=1}^{out_i} \bar{a}_{ij} + b_i = \sum_{j=1}^{out_i} \frac{r_{ij} - l_i}{out_i(h_i - l_i)} + \sum_{j=1}^{out_i} \frac{h_i - r_{ij}}{out_i(h_i - l_i)} = \sum_{j=1}^{out_i} \frac{h_i - l_i}{out_i(h_i - l_i)} = 1 \tag{3.17}$$

The weight b_i is ultimately the probability that the node would rather visit a random node rather than one of its neighbours, which is the *amplification* of the teleportation effect operated by the network transformation described before.

Proposal

With the previously mentioned weight assignment, it is now possible to define Black Hole Metric as a generalization of PageRank. Let's start by defining the new link matrix A', the new sink vector S', the new teleportation vector T' and the new personalization vector V'.

For the sake of convenience, let's name B the *black hole vector*, which is the $N \times 1$ vector

that holds the weights of the arcs going from each node to the black hole. The updated link matrix A' is obtained by combining the B vector with $\bar{A} = \{\bar{a}_{ij}\}$ where $\bar{a}_{ij}$ is defined in (3.13):

$$A' = \begin{pmatrix} \bar{A} & B \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} \tag{3.18}$$

In general, $A \neq \bar{A}$. There are other three entities involved in the computation of the transition matrix used by the random walker model: the teleportation vector T', the personalization vector V', and the sink vector S'. We may define T' and V' as follows:

$$V' = P_0'^{\mathsf{T}} = \begin{pmatrix} V & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{N} \cdot \mathbf{1}_{1\times N} & 0 \end{pmatrix} \quad T' = \begin{pmatrix} T \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{1}_{N\times 1} \\ 0 \end{pmatrix} \tag{3.19}$$

Note that we deliberately excluded the *black hole* from the teleportation effect by putting a value of 0 in the corresponding entries of T' and V'. Since the black hole is a sink by construction, going back there as the consequence of a teleportation effect would only trigger another teleportation effect, which is unnecessary.

Regarding the sink vector, we intuitively want to set to 1 the corresponding index in the vector, as the black hole is a sink, but this actually makes the black hole row in the link matrix not stochastic. Let's consider the matrix A' defined above, and let's use the following sink vector to compute the transition matrix:

$$S^* = \begin{pmatrix} S \\ 1 \end{pmatrix} \tag{3.20}$$

We can calculate M' using (3.10):

$$M' = d(A' + S^*) + (1-d)T'V' = d\left[\begin{pmatrix} \bar{A} & B \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} + \begin{pmatrix} SV & \mathbf{0}_{N\times 1} \\ V & 0 \end{pmatrix}\right] +$$

$$+(1-d)\begin{pmatrix} TV & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} = \begin{pmatrix} d(\bar{A} + SV) + (1-d)TV & dB \\ dV & 0 \end{pmatrix}$$

The black hole row in the link matrix is dV, which is not stochastic: the vector V is, but since $d \neq 1$ the product is not. This happened because we excluded the black hole from the teleportation effect by setting its entry to 0 in T', which interferes with the damping factor correction. In order to compensate for this effect, it is sufficient to multiply the black hole entry in the sink vector by a $\frac{1}{d}$ term:

$$S' = \begin{pmatrix} S \\ \frac{1}{d} \end{pmatrix} \tag{3.21}$$

this makes the black hole row in the link matrix V, which is stochastic. Equations (3.18), (3.19) and (3.21) allows us to define the random walker model according to the definition of M in (3.10):

$$M' = d(A' + S'V') + (1-d)\,T'V'$$

We can now partition M':

$$M' = d\begin{pmatrix} \bar{A} & B \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} + d\begin{pmatrix} S \\ \frac{1}{d} \end{pmatrix}\begin{pmatrix} V & 0 \end{pmatrix} + (1-d)\begin{pmatrix} T \\ 0 \end{pmatrix}\begin{pmatrix} V & 0 \end{pmatrix} = \begin{pmatrix} d\bar{A} & dB \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} +$$

$$+\begin{pmatrix} dSV & \mathbf{0}_{N\times 1} \\ V & 0 \end{pmatrix} + \begin{pmatrix} (1-d)TV & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} = \begin{pmatrix} d\left(\bar{A}+SV\right) + (1-d)TV & dB \\ V & 0 \end{pmatrix}$$

If we name $\bar{M} = d\left(\bar{A}+SV\right) + (1-d)TV$ we have:

$$M' = \begin{pmatrix} \bar{M} & dB \\ V & 0 \end{pmatrix} \tag{3.22}$$

Consider now the following partition of the rank vector P':

$$P' = \begin{pmatrix} \bar{P} \\ p_b \end{pmatrix} \tag{3.23}$$

where p_b is the steady-state probability of the black hole. Note that usually $\bar{P} \neq P$. The rank vector at step n, which we named P'_n, can be obtained using (3.11), (3.22) and (3.23):

$$P'_n = M'^{\mathsf{T}} P'_{n-1} \Leftrightarrow \begin{pmatrix} \bar{P}_n \\ p_{b_n} \end{pmatrix} = \begin{pmatrix} \bar{M}^{\mathsf{T}} & V^{\mathsf{T}} \\ dB^{\mathsf{T}} & 0 \end{pmatrix}\begin{pmatrix} \bar{P}_{n-1} \\ p_{b_{n-1}} \end{pmatrix} \Leftrightarrow \begin{pmatrix} \bar{P}_n \\ p_{b_n} \end{pmatrix} = \begin{pmatrix} \bar{M}^{\mathsf{T}}\bar{P}_{n-1} + p_{b_{n-1}}V^{\mathsf{T}} \\ dB^{\mathsf{T}}\bar{P}_{n-1} \end{pmatrix}$$

We split the calculation in two parts:

$$\begin{cases} \bar{P}_n = \bar{M}^{\mathsf{T}}\bar{P}_{n-1} + p_{b_{n-1}}V^{\mathsf{T}} \\ p_{b_n} = dB^{\mathsf{T}}\bar{P}_{n-1} \end{cases} \tag{3.24}$$

The related random walker process (3.12), given the definition of matrix M' (3.22), the definition of the personalization vector $P'_0 = V'^{\mathsf{T}}$, and the definition of the rank vector of

the Black Hole Metric P' (3.23), can be written as:

$$P' = {M'_\infty}^\intercal P'_0 \Leftrightarrow \begin{pmatrix} \bar{P} \\ p_b \end{pmatrix} = \begin{pmatrix} \bar{M}^\intercal & V^\intercal \\ dB^\intercal & 0 \end{pmatrix}_\infty \begin{pmatrix} P_0 \\ 0 \end{pmatrix} \tag{3.25}$$

An important property of the transition matrix M' is that it leads to a converging random walker process no matter the network topology, as it will be clarified in subsection 3.3.2. As a final note, even though in general $A \neq \bar{A}$ and $P \neq \bar{P}$, in subsection 3.3.2 we will introduce a sufficient condition that allows the identity.

Application to example toy network

After defining the necessary entities and describing how to assign weights in the modified network, it is now possible to show how Black Hole Metric behaves in the sample trust network in Figure 3.6a. For this particular network we set that $l_i = l = 0,\ h_i = h = 10\ \forall i \in [1, N]$. It is easy to note that we have only three types of nodes in the network:

1. Nodes which have two links with weight 1 out of 10 (nodes 2 and 4).
2. Nodes which have two links with weight 9 out of 10 (nodes 3 and 5).
3. Sinks (nodes 1 and 6).

We only show the arc weights of node 2, as the same formulas can be used to calculate the outlink weights of the other nodes. Given that $out_2 = 2$ we have:

$$\bar{a}_{21} = \bar{a}_{23} = \frac{r_{21} - l}{out_2 \cdot (h - l)} = \frac{1 - 0}{2 \cdot (10 - 0)} = \frac{1}{20}$$

The black hole arc weight is going to be:

$$b_2 = \frac{h - r_{21} + h - r_{23}}{out_2(h - l)} = \frac{20 - 2}{2 \cdot (10 - 0)} = \frac{9}{10}$$

as expected, $\bar{a}_{21} + \bar{a}_{22} + b_2 = 1$. By applying the formulas to all arcs we create the network in Figure 3.8. The link matrix A' as in (3.18) is:

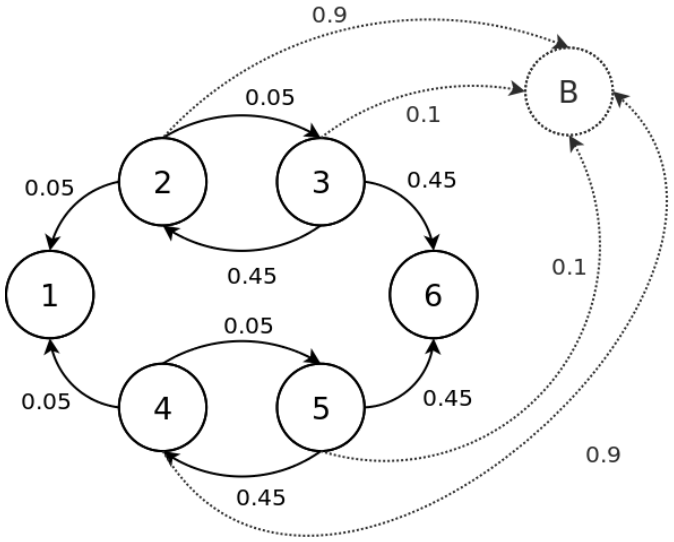


Figure 3.8: The Network in Figure 3.6 with the black hole.

$$A' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.05 & 0 & 0.05 & 0 & 0 & 0 & 0.9 \\ 0 & 0.45 & 0 & 0 & 0 & 0.45 & 0.1 \\ 0.05 & 0 & 0 & 0 & 0.05 & 0 & 0.9 \\ 0 & 0 & 0 & 0.45 & 0 & 0.45 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

while B is:

$$B = \begin{pmatrix} 0 \\ 0.9 \\ 0.1 \\ 0.9 \\ 0.1 \\ 0 \end{pmatrix}$$

Vectors S', V' and T' are obvious from (3.19) and (3.21):

$$S' = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ \frac{1}{d} \end{pmatrix} \quad V' = P_0'^{\mathsf{T}} = \begin{pmatrix} \frac{1}{6} \cdot \mathbf{1}_{1\times 6} & 0 \end{pmatrix} \quad T' = \begin{pmatrix} \mathbf{1}_{6\times 1} \\ 0 \end{pmatrix}$$

If we compute the steady-state probabilities for the random walker process in (3.25) assuming $d = 0.85$, the values calculated for each node (including the black hole) of the network in Figure 3.8 are:

$$p_1 = 0.110 \quad p_2 = p_4 = 0.138 \quad p_3 = p_5 = 0.104 \quad p_6 = 0.178 \quad p_b = 0.228$$

which better models the trust relationships among the nodes, as $p_1 < p_6$. There is also a value p_b for the black hole, which is a consequence of the transformation we operated. Since the black hole is not a real node, this probability does not bear any particular meaning, and it can be discarded.

Complexity assessment

Using (3.24) for direct computation, no matter the method in use, is inefficient in both time and space complexity, therefore we will now introduce a more efficient way to solve the problem. First, let us rewrite $\bar{P}_n$ appropriately:

$$\bar{P}_n = \bar{M}^\mathsf{T}\bar{P}_{n-1} + p_{b_{n-1}}V^\mathsf{T} = d\bar{A}^\mathsf{T}\bar{P}_{n-1} + dV^\mathsf{T}S^\mathsf{T}\bar{P}_{n-1} + (1-d)\,V^\mathsf{T}T^\mathsf{T}\bar{P}_{n-1} + p_{b_{n-1}}V^\mathsf{T}$$

The quantities $T^\mathsf{T}\bar{P}_{n-1} = \bar{t}_{p_{n-1}}$ and $S^\mathsf{T}\bar{P}_{n-1} = \bar{s}_{p_{n-1}}$ are both scalars. In particular, we have:

$$T^\mathsf{T}\bar{P}_{n-1} = \sum_{k=0}^{N}\bar{p}_{k_{n-1}} = 1 - p_{b_{n-1}} \tag{3.26}$$

which allows us to write:

$$\begin{aligned}\bar{P}_n &= d\bar{A}^\mathsf{T}\bar{P}_{n-1} + d\bar{s}_{p_{n-1}}V^\mathsf{T} + (1-d)\,\bar{t}_{p_{n-1}}V^\mathsf{T} + p_{b_{n-1}}V^\mathsf{T} = \\ &= d\bar{A}^\mathsf{T}\bar{P}_{n-1} + [d\bar{s}_{p_{n-1}} + (1-d)\,\bar{t}_{p_{n-1}} + p_{b_{n-1}}]V^\mathsf{T}\end{aligned}$$

The quantity under square brackets can be further simplified using (3.26):

$$d\bar{s}_{p_{n-1}} + (1-d)\,\bar{t}_{p_{n-1}} + p_{b_{n-1}} = d\bar{s}_{p_{n-1}} + (1-d)(1-p_{b_{n-1}}) + p_{b_{n-1}} =$$

$$= d\bar{s}_{p_{n-1}} + 1 - d - \cancel{p_{b_{n-1}}} + dp_{b_{n-1}} + \cancel{p_{b_{n-1}}} = 1 - d(1 - \bar{s}_{p_{n-1}} - p_{b_{n-1}})$$

which allows us to write (3.24) as:

$$\begin{cases}\bar{P}_n = d\bar{A}^\mathsf{T}\bar{P}_{n-1} + [1 - d(1 - \bar{s}_{p_{n-1}} - p_{b_{n-1}})]V^\mathsf{T} \\ p_{b_n} = d\bar{b}_{p_{n-1}}\end{cases} \tag{3.27}$$

$\bar{b}_{p_{n-1}} = B^{\mathsf{T}}\bar{P}_{n-1}$ is also a scalar. The index form of (3.27) is:

$$\begin{cases} \bar{p}_{i_n} = d\sum\limits_{h=0}^{N} \bar{a}_{hi}\bar{p}_{h_{n-1}} + [1 - d(1 - \bar{s}_{p_{n-1}} - p_{b_{n-1}})]v_i \\ p_{b_n} = d\bar{b}_{p_{n-1}} \end{cases}$$

There are three expensive computations in (3.27), which complexity is easily inferrable:

- $\bar{A}^{\mathsf{T}}\bar{P}_{n-1}$. Matrix by vector products usually have a computational complexity of $O(N^2)$. However, in our case, we know that matrix $\bar{A}$ has very few non-zero entries. This number is equal to $|E|$, the total number of arcs in the network, so we can conclude that the average computational complexity is $O(|E|)$ which is less than $O(N^2)$ in the general case.

- $\bar{s}_{p_{n-1}} = S^{\mathsf{T}}\bar{P}_{n-1}$ and $\bar{b}_{p_{n-1}} = B^{\mathsf{T}}\bar{P}_{n-1}$. Inner products among vectors always have complexity $O(N)$. N is less than $|E|$, unless the overall number of arcs in the network is less than the number of nodes itself, which seldom happens.

Then, the overall complexity is $O(|E|)$ in the average case, which is the same as PageRank. Note that $T^{\mathsf{T}}\bar{P}_{n-1}$ does not add to the complexity, as it can be written as the scalar $1 - p_{b_{n-1}}$ and computed offline.

Furthermore, we analyse the memory usage of the entities involved outside the computation:

- Memory usage for adjacency sparse matrix $\bar{A}$ depends on how it is stored. Assuming the storage format is Compressed Column Storage, it is proportional to $2|E|+N+1$.

- Memory usage for personalization vector V, sink vector S and black hole vector B is proportional to N.

- No memory usage for teleportation vector T, as it does not appear in (3.27).

Memory usage of PageRank is proportional to $2|E| + 3N + 1$, since the black hole vector B is not present, whilst the memory usage of Black Hole Metric is proportional to $2|E| + 4N + 1$: they only differ by a factor of N.

Proof of convergence

In this subsection, the convergence of the underlying random walker process of the Black Hole metric will be proven. First, let's consider the modified adjacency matrix A'. We know that it is obtained from A by adding a new node (the Black Hole) and by modifying the arcs. It is a well-formed network nonetheless, and it is possible to evaluate its PageRank. We can define the PageRank transition matrix M^* for this network as:

$$M^* = d(A' + S^*V^*) + (1 - d)T^*V^*$$

where S^* is the same as (3.20) and it is the sink vector S with the addition of an extra sink, the entry of the Black Hole. The teleportation vector T^* is easilly constructed:

$$T^* = \begin{pmatrix} T \\ 1 \end{pmatrix} \tag{3.28}$$

V^* must be a non-negative $1 \times N + 1$ vector. The personalization vector controls the per-node teleportation probability, but as long as $\sum_{i=0}^{N+1} v_i^* = 1$, PageRank is guaranteed to converge no matter which nodes get teleported to, so we can arbitrarilly choose V^* as long as such condition is met:

$$V^* = \begin{pmatrix} V & 0 \end{pmatrix} \tag{3.29}$$

Given that the sum of the elements of V is 1, the sum of the elements of V^* is also 1. Because of (3.20) (3.28) (3.29), we rewrite M^* as:

$$M^* = d\left[A' + \begin{pmatrix} S \\ 1 \end{pmatrix}\begin{pmatrix} V & 0 \end{pmatrix}\right] + (1-d)\begin{pmatrix} T \\ 1 \end{pmatrix}\begin{pmatrix} V & 0 \end{pmatrix} =$$

$$= d\left[\begin{pmatrix} \bar{A} & B \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} + \begin{pmatrix} SV & \mathbf{0}_{N\times 1} \\ V & 0 \end{pmatrix}\right] + (1-d)\begin{pmatrix} TV & \mathbf{0}_{N\times 1} \\ V & 0 \end{pmatrix} =$$

$$= \begin{pmatrix} d(\bar{A}+SV) & dB \\ dV & 0 \end{pmatrix} + \begin{pmatrix} (1-d)TV & \mathbf{0}_{N\times 1} \\ (1-d)V & 0 \end{pmatrix} = \begin{pmatrix} d(\bar{A}+SV)+(1-d)TV & dB \\ V & 0 \end{pmatrix} =$$

$$= \begin{pmatrix} \bar{M} & dB \\ V & 0 \end{pmatrix} = M'$$

The last matrix is the definition of the transition matrix M' for the underlying random walker process of the Black Hole Metric of the network with adjacency matrix A and sink vector S.

In conclusion, if we choose T^* and V^* appropriately, the underlying random walker process for the Black Hole Metric and PageRank is the same. The set conditions do not affect the generality of this statement, and since PageRank is guaranteed to converge for every network, it is possible to safely assume that the Black Hole Metric converges as well regardless of the network structure.

Rank equality theorem

In this subsection, the theorem proving that Black Hole Metric is a generalization of PageRank is presented. Before discussing the theorem, however, it is necessary to introduce the lemma(1).

Lemma 1. *If* $B = \mathbf{0}_{N\times 1}$ *then* $M = \bar{M}$*.*

Proof. If every entry in the B vector is 0 it follows that, $\forall i \in [1, N]$, we have from (3.16):

$$\sum_{k=1}^{out_i} \frac{h_i - r_{ik}}{out_i(h_i - l_i)} = b_i = 0$$

Given that $h_i \geq r_{ij}$ and $h_i > l_i$, since the denominator is always greater than 0, the only way the summation can be 0 is if $h_i = r_{ij}\ \ \forall k \in [1, out_i]$. Let's substitute r_{ij} with h_i in (3.13):

$$\bar{a}_{ij} = \frac{r_{ij} - l_i}{out_i(h_i - l_i)} = \frac{\cancel{h_i - l_i}}{out_i(\cancel{h_i - l_i})} = \frac{1}{out_i}$$

and since $r_{ij} = h_i\ \ \forall j \in [1, out_i]$ we have that $a_{ij} = \frac{1}{out_i} = \bar{a}_{ij}$, so $A = \bar{A}$. According to the definitions of the two matrices M and $\bar{M}$ we have that $\bar{M} - M = \bar{A} - A = 0$ so $\bar{M} = M$. □

It is interesting to note that if B is all zeros, the arc weights are all the same, which is obvious since we are assigning maximum score to each neighbour. Knowing that the two matrices M and $\bar{M}$ are the same when the black hole effect is absent, we can easily prove that the values produced by applying both PageRank and Black Hole Metric are the same.

Theorem 1 (of rank equality)**.** *If every entry in the* B *vector is* 0 *then* $p_b = 0$*, and* $P = \bar{P}$*:*

$$B = \mathbf{0}_{N\times 1} \quad \Rightarrow \quad \begin{cases} P = \bar{P} \\ p_b = 0 \end{cases}$$

Proof. Given that $B = \mathbf{0}_{N\times 1}$ then, for the lemma 1, the random walker (3.25) becomes:

$$\begin{pmatrix} \bar{P} \\ p_b \end{pmatrix} = \begin{pmatrix} M^{\mathsf{T}} & V^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix}_{\infty} \begin{pmatrix} P_0 \\ 0 \end{pmatrix}$$

Let's name $V_1 = \mathbf{1}_{R\times 1} \cdot V \in \mathbb{R}^{R\times N}$ and calculate the n-th power of matrix M'^T:

$$\begin{pmatrix} M^{\mathsf{T}} & V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix}^2 = \begin{pmatrix} M^{\mathsf{T}} & V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} \cdot \begin{pmatrix} M^{\mathsf{T}} & V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} = \begin{pmatrix} (M^{\mathsf{T}})^2 & M^{\mathsf{T}}V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix}$$

$$\begin{pmatrix} M^{\mathsf{T}} & V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix}^3 = \begin{pmatrix} (M^{\mathsf{T}})^2 & M^{\mathsf{T}}V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} \cdot \begin{pmatrix} M^{\mathsf{T}} & V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} = \begin{pmatrix} (M^{\mathsf{T}})^3 & (M^{\mathsf{T}})^2V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix}$$

$$\dots$$

$$\begin{pmatrix} M^{\mathsf{T}} & V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix}^n = \begin{pmatrix} (M^{\mathsf{T}})^n & (M^{\mathsf{T}})^{n-1}V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix}$$

the limit for $n \to \infty$ is:

$$\lim_{n\to\infty} \begin{bmatrix} (M^{\mathsf{T}})^n & (M^{\mathsf{T}})^{n-1}V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{bmatrix} = \begin{pmatrix} M_{\infty}^{\mathsf{T}} & M_{\infty}^{\mathsf{T}}V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix}$$

so we may write the random walker as:

$$\begin{pmatrix} \bar{P} \\ p_b \end{pmatrix} = \begin{pmatrix} M_{\infty}^{\mathsf{T}} & M_{\infty}^{\mathsf{T}}V_1^{\mathsf{T}} \\ \mathbf{0}_{1\times N} & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ 0 \end{pmatrix} = \begin{pmatrix} M_{\infty}^{\mathsf{T}}P_0 \\ 0 \end{pmatrix}$$

hence:

$$\begin{cases} \bar{P} = M_{\infty}^{\intercal} P_0 = P \\ p_b = 0 \end{cases}$$

because of (3.12). □

3.3.3 Experiments

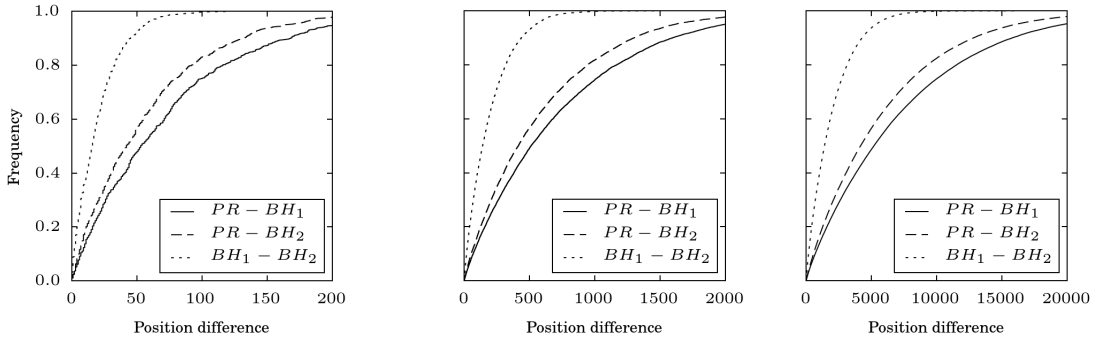
The Black Hole metric was tested against synthetic networks and a real world network. The objective was to study the behaviour of the Black Hole metric using different networks having different size and different topology. The expected result is that the Black Hole Metric should produce a different ranking, but this does not mean that the produced ranking is an improvement over the ranking produced by PageRank, as it is hard to generate or find a network that allows us to clearly highlight the effect mentioned in the toy example. Nonetheless, the possibility exists, and black hole metric still stands as the only solution to this hard to detect issue.

In particular, the results for six different synthetic networks are shown, three weighted directed Erdős-–Rényi random graph networks [107] of size 1000, 10000 and 100000, and three weighted directed scale-free random networks, of size 1000, 10000 and 100000. The chosen networks all differ either in size or in topology, and form a usable set of networks of different characteristics. All Erdős-–Rényi networks were created so that the average outdegree is 10 and in addition all generated the directed scale-free networks following the algorithm described by Bollobás in [108]. Using the same notation of [108], the parameters for the generated networks were chosen as follows: The idea behind the experiments

Parameter	Value	Description
α	0.41	Prob. of adding an edge from a new node to an existing one.
β	0.54	Prob. of adding an edge between two existing nodes.
γ	0.05	Prob. of adding an edge from an existing node to a new one.
δ_{in}	0.20	Bias for choosing nodes from in-degree distribution.
δ_{out}	0	Bias for choosing nodes from out-degree distribution.

we performed is to test the Black Hole metric in a "wary" environment to show that PageRank does not make difference if the weights are all multiplied by a constant factor. Therefore, assuming that the weight range for each arc is $[0, 99]$, the generated weights in each network were set to be in range $[0, 49]$, the lower half of the full range. Then, both PageRank and the Black Hole Metric were applied, and we derived the rank position of each node in the network. This is the first step of the simulation. For the sake of convenience, two result sets are named respectively PR_1 and BH_1. After the first step, the weights were multiplied by a factor of $^{99}/_{49}$, effectively scaling the weights range from $[0, 49]$ to $[0, 99]$. Both metrics were applied again and the result sets were named PR_2 and BH_2. This was the second step of the simulation. As expected, $PR_1 = PR_2$, so, for ease of notation, the PageRank result set for both steps will be simply referred to as PR. The curves describe the cumulative distribution function of the absolute rank position difference. The result sets were compared and the results condensed as shown by Figures 3.9, 3.10 and 3.11. In all the figures, the x-axis stands for the absolute rank position differences between two results sets, while the y-axis stands for the cumulative frequency of appearance. In order to better explain what the axes mean, let's take as an example the solid line in Figure 3.9a, which depicts the frequency of position difference between the result sets PR and BH_1. We can see that for a position difference of 50 there is a frequency of about 0.4. This means that about 40% of the nodes ranked in the result set PR differ by at most 50 from the position they received in the result set BH_1. Since the result sets are always compared pairwise, we will use the notation $R-Q$ to illustrate thex

absolute rank position difference among the result sets of R and Q. To trim the outliers from the result sets, we have restricted the x-axis to 20% of the maximum possible rank position difference (which is equal to the size of the network).



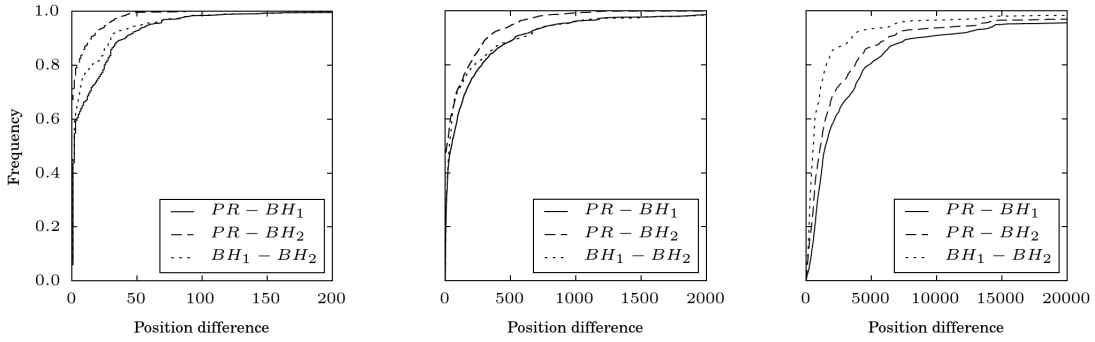
(a) Network of size 1000 (b) Network of size 10000 (c) Network of size 100000

Figure 3.9: Comparison of the empirical CDF of the absolute rank position difference (Erdős-–Rényi networks)

In Figure 3.9 the results of both steps of the simulation of the three Erdős-–Rényi networks are shown. Note that the network size does not affect the shape of the curves; they are very similar for all three network instances. Moreover, the $PR - BH_2$ curve is always above the $PR - BH_1$ curve, which means that the BH_2 result set nearer to the PR result set than BH_1 is. This can be explained if we look at the arc weights: BH_1 comes from a network where the overall weights are lower than BH_2. In a network with lower weights, the black hole steady-state probability is higher, which means that it is more likely that a random walker, from any node, moves to the black hole. But the black hole is a sink so the random walker will teleport after reaching it. This means that the black hole has an higher steady-state probability, and the teleportation effect is amplified.

As specified above, the PageRank result sets are identical in both simulation steps meaning that the PageRank metric fails to capture the effect induced by the different weight distribution. The dotted curve $BH_1 - BH_2$ highlights that the two result sets BH_1 and BH_2 are always different. Note that this curve is steeper than same curve related to the

other two sets, because the difference among the two result sets BH_1 and BH_2 is overall less than the difference between either BH_1 or BH_2 and PR.

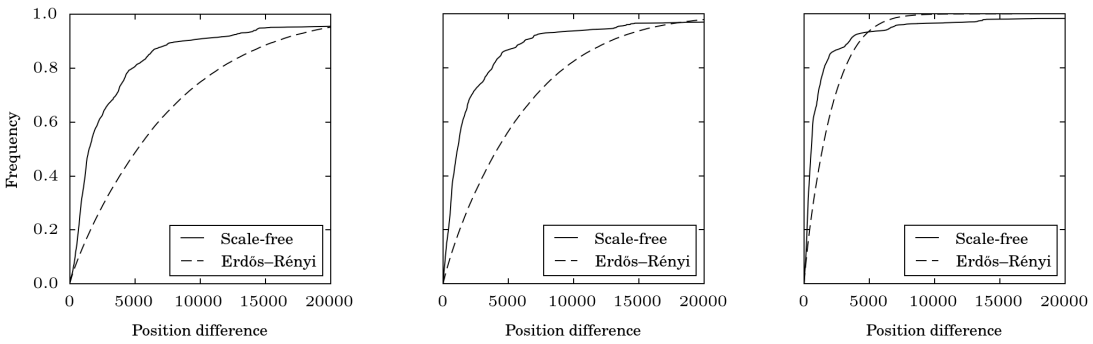


(a) Network of size 1000 (b) Network of size 10000 (c) Network of size 100000

Figure 3.10: Comparison of the empirical CDF of the absolute rank position difference (scale-free networks)

Figure 3.10 compares the result sets related to the three scale-free network. The network size of scale-free networks does not influence much the shape of the curves, however both $PR-BH_1$ and $PR-BH_2$ get smoother when the network increases in size. Despite this difference, the curve $PR-BH_2$ keeps staying above the curve $PR-BH_1$, meaning that the Black Hole metric assesses the difference in wariness of the nodes even in scale-free networks. Finally, the two result sets BH_1 and BH_2 exhibit different behaviour when the network size grows: the curve $BH_1 - BH_2$ is between the other two curves when size is 1000, it almost coincides with $PR - BH_1$ when size is 10000, it is above the other two curves when size is 100000.

At last, in Figure 3.11 we compare Erdős-–Rényi and scale-free networks of size 100000 by grouping together the curves of the same pair of result sets. Note that the scale-free curves are different than the Erdős-–Rényi curves. This effect may depend on the different topology of the two networks. In scale-free networks, nodes with high indegree, which are few in number, are less affected by the weight fluctuations we introduced with our experiments. Nodes with low indegree, which are more, are instead strongly affected by



(a) $PR - BH_1$ (b) $PR - BH_2$ (c) $BH_1 - BH_2$

Figure 3.11: Comparison of the empirical CDF of the absolute rank position difference, same size (100k nodes)

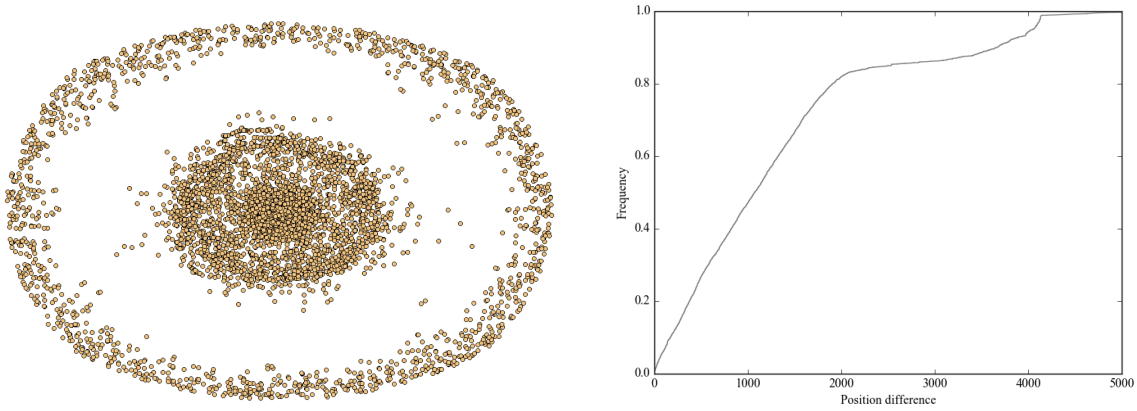
the weight doubling, and their positions change a lot. This causes the scale-free curves to appear steeper compared to the Erdős-–Rényi curves, although the behaviour of the Black Hole metric remains the same.

The second set of experiments we apply the Black Hole Metric to a real world network, Advogato, retrieved from [109].

Advogato (www.advogato.org) is an online community platform for free software developers. As reported in the website of Advogato *"Since 1999, our goal has been to be a resource for free software developers around the world, and a research testbed for group trust metrics and other social networking technologies"*. Here the Advogato [110, 111] trust network is considered, where nodes are Advogato users and the direct arcs represent trust relationships. Advogato names *"certification"* a trust link. There are three different levels of certifications, corresponding to three weights for arcs: *apprentice* (0.6), *journeyer* (0.8) and *master* (1.0). A user with no trust certificate is called an *observer*. The network consists of 6541 nodes and 51127 arcs and it exhibits an indegree and outdegree power law distribution. As in the previously discussed experiments, we compute on this network the PageRank and the Black Hole Metric and compare them using a cumulative distribution graph, where the x-axis represents the possible absolute rank

position difference between the PageRank and the Black Hole Metric of the nodes, while the y-axis represents the cumulative frequency of appearance. To compute Black Hole Metric we set $l_i = 0.6$ and $h_i = 1.0$ for all nodes in the network.

Figure 3.12 shows the results. It is clear that the Black Hole Metric produces different values (and ranks) compared to those computed by using PageRank. In practice, it means that Black Hole Metric produces a different ranking compared to PageRank. For example, in Table 3.5, we report the rank of the first 10 users of Advogato, computed using the PageRank and Black Hole Metric.



(a) The network. (b) $PR - BH$ CDF.

Figure 3.12: The Advogato trust network

As reported in the website of Advogato, in order to assess the certification level of each user they use a basic trust metric computed relatively to a "seed" of trusted accounts. The original four trust metric seeds, set in 1999 when Advogato went online, were: *raph* (Raph Levien), *miguel* (Miguel Icaza), $federico$ (Federico Mena-Quntero) and *alan* (Alan Cox). In 2007 *mako* (Benjamin Mako Hill) replaced $federico$. As we can infer from Table 3.5 both metrics are somewhat able to capture the important role covered by the Advogato trust metric seeds, by putting them in the top positions. However, Black Hole Metric, in our opinion, produces a more appropriate ranking, according to the following

Rank	PageRank		BlackHole metric	
	NodeName	**PageRank Value**	**NodeName**	**BlackHole Value**
1	federico	0.02093458	alan	0.00594131
2	alan	0.00978148	miguel	0.00387012
3	miguel	0.00658376	rms	0.00290212
4	raph	0.00405245	raph	0.00230948
5	rms	0.00381952	federico	0.00176002
6	jwz	0.00274046	jwz	0.00172800
7	davem	0.00262117	rasmus	0.00158964
8	rth	0.00258019	rth	0.00158964
9	rasmus	0.00250191	gstein	0.00138078
10	gstein	0.00230680	davem	0.00135993

Table 3.5: Rank of the first 10 users of Advogato trust network computed by using PageRank and Black Hole Metric.

observations:

- $federico$ is first according to PageRank, while is 5th according to Black Hole Metric. Moreover the PageRank $federico$'s value is also significantly higher compared to *alan* (the second in the chart), which means that $federico$ is steadly in the first position with a wide margin, despite the fact that he has not been a seed since 2007. We believe that lower position that the Black Hole Metric assigns to $federico$ better captures the fact that he was swapped out of the seed set.

- Another interesting difference is about the different position of the node *mako*. It is ranked 257^{th} by the PageRank and 142^{th} by Black Hole Metric. This ranking difference suggests that the Black Hole Metric better captures the relevance that *mako* has been assuming inside the Advogato community.

Chapter 4

Mesoscale

One of the most attractive problems in network science deals with the identification of the so-called mesoscale structure of a complex network, a topic of intensive research activity across multiple disciplines [112]. Its importance relies in the ability to unveiling communities of units that, in turn, can be used to explain some hidden behaviours of networks emerging as the result of the complex interaction patterns among nodes (or entities).

Community detection has been successfully used to analyze the structure of single-layer networks and for modeling several kinds of interactions, such as social relationships, genetic interactions among biological molecules or trade among countries [76, 77, 78, 79, 80, 81, 82], just to mention a few (a detailed introduction to communities in networks can be found in [72, 73]).

Despite this intuitive concept, a precise definition of a community is still a topic of debate among network scientists. In this section, two completely different approaches to the detection of communities will be presented, one uses a genetic programming approach to produce a partition quality assessment function, and the other studies the parameter

selection for an information-theoretic algorithm for community detection.

4.1 Genetic Programming

The idea of having machines automatically solve problems has always been central in the domain of artificial intelligence [113, 114], but only recently the technological advancements in the field of computing speed allows to exploit these techniques to solve more complex cases. A relevant problem since the early days of artificial intelligence is however a machine would solve a problem which solution is a computer algorithm itself.

Genetic Programming (GP) attempts to take on such challenge by making use of the concepts of evolutionary computation, which borrows from nature the idea of the survival of the fittest. It aims at generating a feasible algorithm that can solve the specified problem without requiring the user to specify the shape of the solution in advance.

The gist of GP consists in evolving a population of computer programs. Computer programs which participate in the process are named *individuals*. At each iteration of the process, the population is evaluated, and each individual is given a numerical score named *fitness*. The better the fitness, the more likely an individual is a solution to the GP problem. The fitter individuals are then manipulated by the use of *genetic operations* in order to generate a better population for the next iteration. The process continues until an exit condition is satisfied: the fittest individual that was ever bred among all the iterations will be designated as the solution to the problem. This whole process is shown in Figure 4.1.

The GP process is inherently random, and sometimes it produces no meaningful solutions. However, this randomness allows GP to avoid the traditional pitfalls of deterministic search algorithms.

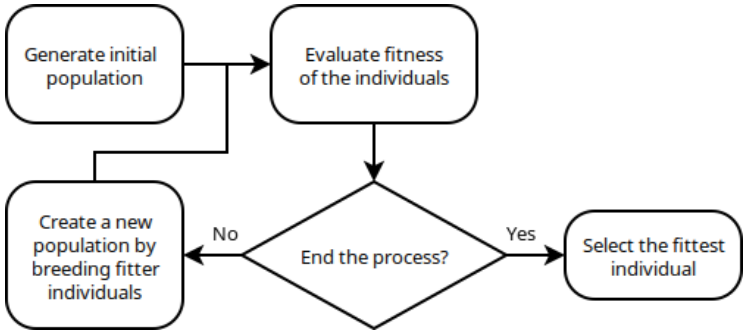


Figure 4.1: Overview of the Genetic Programming process

Setting up a GP problem means specifying how an individual is constructed in terms of terminals and functions, defining a proper fitness function and providing parameters that control the run, including the exit conditions, as summarized in Figure 4.2.

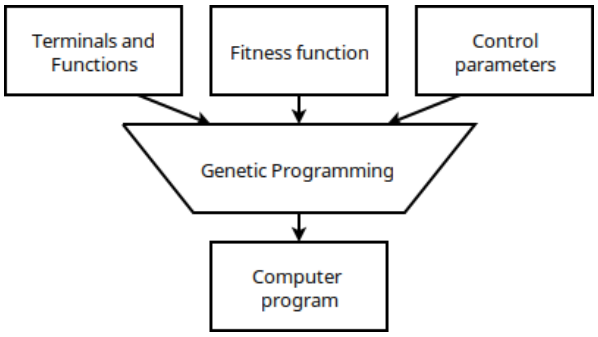


Figure 4.2: Preparatory steps for the Genetic Programming process

Terminal and function sets

The terminal set is the set of values that are used as arguments of the functions in the function set. It may consist of:

- External inputs of the program, typically represented by named variables like `x`.
- 0-ary functions, like `time()`, that may return a different value each time they are run.
- Constants, either determined before the run or created by mutation.

The function set is very dependent on the application domain. In simple numerical problems, it may consist of the four basic arithmetic functions (`+`, `-`, `*`, `/`), but they could be higher level functions: for example, if we are looking for an auto-pilot system for a car, functions could include `steer()`, `accelerate()`, `decelerate()` in a simplest case.

Fitness function

Defining a good fitness function is perhaps the most crucial step when setting up a GP problem. A good fitness function should always return large (small) values for individuals that fit, and small (large) values for individuals less fit, so that the individual which has the highest (lowest) score is the fittest. It is often the sole mechanism to provide a high-level statement of the problem's requirements. For example, if the GP problem consits in finding the closest rational number for any real number `n`, the program `floor(n*100)/100`, is more fit than the program `floor(n*10)/10`, as it gives a more accurate result for all values of `n`, so it should receive a better score.

Control parameters

At last, there are several parameters that need to be configured in order to start the GP search: the termination criterion, the population size, how the initial population is created, the probability of applying a genetic operators and so on. Of all these parameters, the most important two are the population size and the termination criterion. Regrettably, it is not possible to make general recommendations regarding an optimal set of GP parameters, as it strictly depends on the specific application. However, GP is often robust, and many different parameter values may work.

4.1.1 Community structure validation problem

In this section, a technique to solve the problem of community structure validation with a GP approach will be presented, as described in [5]. The solution takes the form of a *validation function*, which is a function that assigns a certain score to a partition of a network in clusters: the closer to the optimal that partition, the better the score. For simplicity's sake, let's say that a partition is better (worse) than another when it's closer (farther) from the optimal partition.

Note that in the general case it's not possible to assume that a better partition always gets an higher score compared to a worse partition. The validation function might assign a higher score to worse partitions, depending on its shape. This is the reason why the terms "lower" or "higher" will not be used when considering the validation function score, but the more generic "better" and "worse" will be employed.

Such a function could be used in conjunction with global optimization methods to find communities: in this case, the goal is to find the partition that yields the best score, or get reasonably close to that. However, let's first describe in more formally what to look for, so that it's possible to determine if a solution to the problem exists.

Let's assume we have an undirected, unweighted network $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. If we name $\mathcal{P}$ the space of all the possible partitions of G, we are looking for a validation function $\beta : \mathcal{T}_{\mathcal{P}} \to \mathbb{R}$ that has a reasonable computational complexity. $\mathcal{T}_{\mathcal{P}}$ is the space of terminal sets obtained from all the partitions in $\mathcal{P}$: its generic element is simply the terminal set of a specific partition $P \in \mathcal{P}$. However, the β function is very difficult to handle as an individual of the GP problem due to the high dimensionality of $\mathcal{T}_{\mathcal{P}}$, as it would require many terminals to provide enough information to describe the whole partition of the network.

Instead of looking for a validation function as a whole, to reduce the dimensionality of the selection function, the function β was fragmented so that it operates on the terminal sets $T_e \in \mathcal{T}_E$, $\forall e \in E$, which have less dimensions:

$$\beta(T_e) = \sum_{i=1}^{r_k} \sum_{e \in E_i} f(T_e) \tag{4.1}$$

$f : \mathcal{T}_E \to \mathbb{R}$ is an individual of the population in the GP problem, and $\mathcal{T}_E$ is the space of terminal sets obtained from all the edges in E, hence a generic $T_e \in \mathcal{T}_E$ is the terminal set of an edge $e = (v, w)$. This terminal set should contain numeric information about the nodes that connects. This includes microscopic parameters like the degree of v and w or their structural difference. However, there are also mesoscopic (related to the communities e belongs to) and macroscopic parameters (related to the whole network) that are worth considering even when evaluating the score of a single edge. For example, it may be worth comparing the degree of v or w to the average degree of the nodes within the same community, or to the average degree of the nodes within the network. Of course it's impossible to know exactly how the terminals will be compared within the function f due to the nature of GP, but the terminal set must offer the opportunity for such comparisons to happen.

Finally, note that all $(v, w) \in E : v \in C_i,\ w \in C_j,\ C_i \neq C_j$ were excluded on purpose. This simplification is necessary to further reduce the dimensionality of the terminal sets and the overall complexity of the GP problem, because including these arcs would imply two problems to be addressed:

- It would be necessary for the f to behave differently for inter-community and intra-community links. This makes the search much harder, so, as far as complexity is

concerned, it is better that all the arcs are of the same type.

- Inter-community links require more terminals than intra-community links, because they bear mesoscopic information about two communities instead of one.

In conclusion, the GP problem consists in finding an individual $f : \mathcal{T}_E \to \mathbb{R}$ that, applied to all intra-community edges, will provide a score to a certain partition $P \in \mathcal{P}$. The terminal set will provide access to microscopic, mesoscopic and macroscopic properties that can be used by the GP algorithm to create a suitable f. In the following subsection the parameters of the GP process will be described in detail.

4.1.2 Methodology

As already mentioned, the parameters that characterize the proposed GP run are illustrated, which are the terminal set, the function set, the fitness function, and all the control parameters such as the population size and the termination criterion.

Terminal set

The terminal set was one of the most challenging parameters to define. On one hand, we want to include several different properties from the network at different levels (microscopic, mesoscopic, macroscopic), on the other hand too many properties would raise the complexity of the GP problem, making the solution harder to search for. As already mentioned in subsection 4.1.1, the original problem in order to make use of a reduced terminal set. If we name the generic edge $e_i = (v_i, w_i) \in E_i$ belonging to the community C_i, the terminal set we decided to make use of is the following:

- Microscopic Parameters

  - degree of node v_i;
  - degree of node w_i;
  - structural equivalence between v_i and w_i;
  - number of arcs of v_i that point to other nodes in C_i;
  - number of arcs of w_i that point to other nodes in C_i.

- Mesoscopic Parameters
  - average degree of nodes in C_i;
  - total number of edges in C_i;
  - total number of nodes in C_i.

- Macroscopic Parameters
  - average degree of nodes;
  - total number of edges;
  - total number of nodes.

The information concerning each level is very abstract and simple by design: the GP run should not be biased with excessively refined mathematical models. The only exception to this could be the structural equivalence, which is computed via the cosine similarity. If the results suggest that the quality of the solution would benefit from a larger terminal set, it is of course always possible to add other parameters in subsequent runs. Likewise, it is also reasonable to remove some of the parameters if it appears that they are unused in the fitter individuals.

Function set

Contrary to the terminal set, the function set is small, and consists of only five functions: $\{+, -, \times, \div, \sqrt{}\}$, which are the binary addition, subtraction, multiplication and *protected division* and the unary square root operator. The protected division operator $\div$ is defined as:

$$a \div b = \begin{cases} 1, & \text{for } b = 0 \\ \frac{a}{b}, & \text{otherwise} \end{cases}$$

The function set is small for two reasons: having a smaller function set decreases the complexity of the algorithm, and since most of the used functions are simple, they have less impact on the overall computation time. Note that there are important terminals and functions that can be derived from a combination of elements from the defined terminal and function sets:

- 0 can be written as $n - n$, $\forall n \in \mathbb{R}$;
- 1 can be written as $n \div n$, $\forall n \in \mathbb{R}$;
- n^2 can be written as $n \times n$, $\forall n \in \mathbb{R}$;
- $|n|$ can be written as $\sqrt{n \times n}$, $\forall n \in \mathbb{R}$.

Fitness

Determining a proper fitness function is also a major challenge, and often the success of a GP search depends on how accurately the fitness functions validates the correct solution. In the case described in this section, the fitness function needs to evaluate how well our validation function β (4.1) behaves. In practice, its behavior is tested by applying it to

the ground-truth partition P^* and d randomly generated partitions. The scores of these randomly generated partitions are then compared against the score of the ground-truth partition. Intuitively, the more a partition P_k is similar to the ground-truth partition, the better score β should yield.

Unfortunately, comparing the scores as they are gives little or no information about how accurate is the validation function in scoring a specific partition. Assuming P^* is the ground-truth partition, how can we say that $\beta(T_{P^*})$ yields the best score if we don't know the maximum value that β can assume? Assuming P is a generic partition, how can we say that $\beta(T_{P_k})$ is better or worse than $\beta(T_{P^*})$ when we don't know the shape of β? This is why the correlation between the difference of the two β scores and the normalized mutual information (NMI) a measure of how different two partitions are, was measured. If the difference is correlated to the NMI, it means that the β function behaves as desired, and it is a good candidate for our solution. Note that it would be equally possible to use any kind of difference measurement: NMI was chosen because it is well-studied and has convenient properties [115].

The aforementioned intuitions were used in order to measure this correlation and obtain the fitness function φ. First, let's define the basic building block for our fitness function, which is the function $\gamma : \mathcal{P} \to \mathbb{R}$, defined as following:

$$\gamma(P) = \frac{|\beta(T_{P^*}) - \beta(T_P)|}{NMI(P^*, P)} \tag{4.2}$$

This function alone does not measure correlation of course. To do that, we need to

consider its *standard deviation* σ_γ:

$$\mu_\gamma = \sum_{i=1}^{d} \frac{\gamma(P_i)}{d} \qquad \varphi(f) = \sigma_\gamma = \sqrt{\sum_{i=1}^{d} [\gamma(P_i) - \mu_\gamma]^2} \tag{4.3}$$

Given the definition of our fitness function, we may conclude that the best individual f is the one that minimizes φ.

Control parameters

Compared to the other settings, determining the optimal control parameters beforehand is usually not possible. Things like population size, crossover ratio, number of generations, are best determined via experimentation. As far as the complexity parameters are concerned, in principle, it is common to start with a small population (about 50 individuals) and an average number of generations (about 30). These numbers may be refined according to the performance of the GP framework in terms of the quality of the results and computing time.

The crossover ratio, which is the chance that crossover occurs between two genes, is also an important factor. Normally, each generation is subject to different genetic operators randomly. Certain individuals will undergo crossovers, other mutation. The crossover ratio indicates what is the chance of two individuals to crossover. A traditional approach [116] is to have a crossover ratio of 0.9, while the mutation ratio is set to the remaining 0.1, and this is what has been used in the experiment.

There is also a variety of different genetic operators and strategies that have to be chosen. For example, it makes sense for certain GP problems to adopt automatically defined functions (ADF), a way to evolve reusable components, but they are most effective in problems which present some degree of regularity. Also, there are many different kinds

of crossover and mutation operators [113], and the problem of determining which kind of operator to use is complex [117].

4.1.3 Experiments

Although it is currently a work in progress, the methodology explained above was applied to a real-world network to see if it would be possible to produce a validation function out of a GP run. The experiments were performed using the DEAP framework [118], a distributed evolutionary algorithm framework written in Python. The GP strategy was tested on the well-studied karate club network [119], as the ground-truth partition in communities is known. As reported in formula (4.3) we also need to choose how many random partitions we need to generate. Using more random partitions makes the validation function more resilient to overfitting, but it makes computing the fitness function more costly.

Moreover, when setting up the framework, special focus was given on how to minimize the running time of the algorithm. This was accomplished by doing precalculations:

- In formula (4.2) the normalized mutual information between the reference partition P^* and the partition P is indipendent of the specific β, so it can be precalculated.
- The structural equivalence, together with most other parameters, can be calculated offline, and stored as node/community/network metadata.

At last, a problem which affects genetic programming is the phenomenon of *bloat*[120]. In a nutshell, bloated individuals are individuals which contain redundant terminals and functions, the GP equivalent of reducible polynoms. The redundant entities do not contribute to the quality of the individual, and they slow down the computation of the fitness function, and use more space in memory. In order to fight bloat, a penalty function

on tree depth was introduced into the computation of the fitness function: this way, smaller individuals are prefered over large, bloated ones.

With the mentioned setup, initial tests were done with 100 different partitions, crossover rate 0.85, mutation rate 0.15, 40 generations and a population of 10000 individuals, but the results were rather poor. In general, the quality of the validation function was found to increase when either the population is larger, or if the number of generations rises. However, with more generations, the validation function tends to overfit the sample partitions: if it is used with a different set of randomly generated partitions, it yields poor results. In the future, strategies to avoid overfitting will be investigated, so that they can eventually be implemented into the framework, which will be made able to yield better results.

4.2 Information Theory

As already mentioned, a community is a group of nodes more densely connected each other inside the group and sparsely connected to nodes outside the group. In literature, there are several attempts at giving meaning to the intuitive concept of a community. Some of them exploit concepts and ideas taken from information theory, for example [74]. This method is based on the formulation of a new quality function called map equation[75], which allows to find the optimal description of the network by compressing its information flow. The algorithm is the core of *Infomap* (http://www.mapequation.org/), the search method for minimizing the map equation over possible network partitions. This section provides an information-theoretic based method that uses an extension of Infomap to detect communities in multi-layer networks.

Multi-layer networks, in a nutshell, are formally equivalent to edge-labeled multigraphs.

They consist in several single-layer networks, which are regular graphs, which are connected by inter-layer edges. Multi-layer complex network are a very useful tool to model the empirical systems which complexity is not captured by single-layer networks, as the ones where units exhibit multiple types of relationships simultaneously. This is the case of social systems, where an individual can have family, business or trust interactions with other individuals, or of transportation systems, where geographical areas might be connected by different transporation means such as bus, tube, rail, so forth and so on.

The suitability of multilayer networks for capturing this higher amount of complexity led to a growing interest in their study [121, 122, 123] and to a more general mathematical framework, which can be used when nodes are connected to each other via multiple types of edges or a network changes in time [124]. In fact, multilayer networks are more adequate to model real world interactions that cannot be aggregated into a single network without a loss, in general, of some important structural or dynamical properties [125, 126].

Many methods and measures developed for single layer networks have been extended to be applicable to multilayer networks [124, 127, 128, 129, 130]. In this context new community detection methods have been devised, mainly by reusing concepts already developed for single layer networks. In Ref. [131], the authors proposed a method based on a generalization of the modularity to multilayer networks. This extended modularity is mainly based on generalized null models obtained by considering a Laplacian dynamics [132, 133] on the multilayer network. To compute communities by using such a generalization of the modularity function, an extension of the Louvain algorithm [134] has been also proposed in Ref. [135].

In Ref. [125] an extension of the map equation to multilayer networks is introduced. It is based on the generalization of random walks to multilayer systems [136], which in

turn are used to generate the corresponding network flow to be compressed in order to identify community flows in multilayer networks. The resulting algorithm – i.e. *Multiplex Infomap* [125] – is the extension of *Infomap* to the case of multilayer networks.

A drawback of community detection algorithms for non-interconnected/edge-colored networks is their dependence on at least one parameter which regulates the structural or dynamical coupling between layers. In the case of Multiplex Infomap, this parameter is known as the *relax rate* r. The relax rate is the parameter responsible of modeling movement among layers. At each random walker step, there is a $1 - r$ probability that the random walker simply moves to a neighbour in the same layer, and an r probability that it changes layer, and then moves to a neighbour on that layer.

The choice of r is crucial and, in general, it depends on the network under analysis. While empirical results suggest that values smaller than 0.5 are generally appropriate for most networks [125], finding the actual optimal value is still an unsolved problem. Moreover, in community detection the concept of an absolute optimal simply does not exist, as it is difficult to ascertain whether the chosen algorithm is able to detect the absolute optimal partition. In fact, a safer approach is to assess that a certain partition can be optimal with respect to a specific algorithm. In this section, the content of [6] is presented, where the goal is to find the value of r which provides the best possible partition with respect to *Multiplex InfoMap* in the case of multilayer systems where the strength of coupling among layers is unknown.

4.2.1 Information-theoretic approaches to parameter selection

Multiplex InfoMap is an algorithm which optimizes the map equation [74], a measure of the information-theoretic duality between data compression and the problem of extracting

significant information from compressed data. Given that its roots lie firmly in the realm of information theory, it is natural to develop an information-theoretic algorithm to determine the relax rate producing an optimal partition, with respect to some criteria. In the domain of information theory, this partition would be the one which retains the most information about the network inside the communities. In literature, there are several attempts at exploiting the concepts of information theory to evaluate the quality of a partition [137, 138, 74, 139], in this section, the suitability of two measures will be investigated: information loss [138] and average inter-community entropy [140].

Information loss

Information loss occurs when a certain source of information is compressed in a way that some of the information is discarded as a result of the compression. Since any source of information can be fed to compression algorithms, let's describe how to compress a network, and the information loss that derives from this operation.

Compressing a network X involves finding some representation Y that only keeps part of the available information on connectivity. In [138], the authors compress the network into a representation that preserves the information contained inside the communities in order to evaluate how much information is required to rebuild X given the representation Y. If we name this quantity $H(X|Y)$, given that $H(X)$ is the average amount of information required to describe X, we can compute it from mutual information $I(X;Y)$ between X and Y by

$$H(X|Y) = H(X) - I(X;Y). \tag{4.4}$$

The compressed representation Y is still a graph where each node is a community and

links between nodes are inter-community connections. Hence, we may completely describe Y with the tuple (N, L), with $N = \{n_i\}$, where n_i is the number of nodes of the i-th community, and $L = \{l_{ij}\}$, where l_{ij} is the number of links that go from community i to community j. Note that the definition provided is exactly equivalent to the cross-entropy (or negative log-likelyhood) of the stochastic block model (SBM) [141], a generative model for random graphs. If we assume that there are m communities, in the simplest case of undirected and unweighted networks, Eq. (4.4) reduces to

$$H(X|Y) = \log_2 \left[\prod_{i=1}^{m} \binom{n_i(n_i-1)/2}{l_{ii}} \prod_{j<i} \binom{n_i n_j}{l_{ij}} \right]. \tag{4.5}$$

This formula accounts for all the possible ways to arrange the links that go from nodes of community i to nodes of community j, hence representing all the possible configurations of networks that can be reconstructed knowing Y. The higher the value, the more information is contained in the inter-community links. Extending this formula for directed networks is straightforward, since one should evaluate the possibility that a link can connect two nodes in two different ways (from i to j and vice-versa):

$$H(X|Y) = \log_2 \left[\prod_{i=1}^{m} \binom{n_i(n_i-1)}{l_{ii}} \prod_{j\neq i} \binom{n_i n_j}{l_{ij}} \right]. \tag{4.6}$$

Weights can be included as well, to account for more complex structures. For each link l_{ij} of Y, which represents the total number of links from community i to community j, we have a quantity w_{ij} encoding the sum of the weights of links that go from i to j. Ideally, each configuration reconstructed from Y using (4.6) generates further configurations if we consider all the possible ways to distribute w_{ij} among l_{ij} links. The number of configurations is infinite if the weights are real numbers: given a weight w_{ij}, the problem

is analogous to splitting the interval $[0, w_{ij}]$ in l_{ij} parts, and since any real interval is uncountable, there are infinite ways to make the partition. However, if we impose the restriction that the weights are natural numbers, the number of partitions can be calculated as follows. First, we assign the weight 1 to each one of the l_{ij} links, thus imposing the restriction $w_{ij} \geq l_{ij}$. Since we already distributed l_{ij} out of the total w_{ij}, calculating all possible distributions of the remaining $w_{ik} - l_{ij}$ among l_{ij} links depends on combinations with replacement:

$$C^R(l_{ij}, w_{ij} - l_{ij}) = \frac{(l_{ij} + w_{ij} - l_{ij} - 1)!}{(w_{ij} - l_{ij})!(l_{ij} - 1)!} = \frac{(w_{ij} - 1)!}{(w_{ij} - l_{ij})!(l_{ij} - 1)!} = \binom{w_{ij} - 1}{l_{ij} - 1}. \tag{4.7}$$

Thus, we can update equation (4.6) to include all the possible ways to distribute w_{ij} among l_{ij} links:

$$H(X|Y) = \log_2 \left[\prod_{i=1}^{m} \binom{n_i(n_i - 1)}{l_{ii}} \binom{w_{ii} - 1}{l_{ii} - 1} \prod_{i \neq j} \binom{n_i n_j}{l_{ij}} \binom{w_{ij} - 1}{l_{ij} - 1} \right]. \tag{4.8}$$

A more general formula, accounting for the possibility of self-links (e.g., useful for modeling citation networks) is given by

$$H(H|Y) = \log_2 \left[\prod_{i=1}^{m} \prod_{j=1}^{m} \binom{n_i n_j}{l_{ij}} \binom{w_{ij} - 1}{l_{ij} - 1} \right]. \tag{4.9}$$

Since $H(X|Y)$ represents the information that is lost when compressing the network, a good compression requires $H(X|Y)$ to be as small as possible: hence, our goal is to minimize this quantity.

Average inter-community entropy

In information theory, entropy is a key measures of information content. In fact, entropy can be used to assess the information retained inside and between communities, which together compose the total information contained within the network. In [140], a link-centric definition of entropy is proposed, to tie information content with arc weights. This approach is supported by the fact that most networks can be represented solely by providing their adjacency matrix. We can measure the inter-community entropy of a community C_i as the entropy of all the links that go from nodes in C_i $(i = 1, 2, ..., K)$ to nodes belonging to other communities:

$$H(C_i) = \sum_{j=1}^{K} p_{ij} \log_2(p_{ij}), \qquad p_{ij} = \frac{w_{ij}}{\sum_{j=1}^{K} w_{ij}}, \tag{4.10}$$

where K is the total number of communities and w_{ij} is the total weight of the arcs linking community C_i to community C_j. Inter-community entropy $H(C)$ is calculated by summing up the contribution of each community:

$$H(C) = \sum_{i=1}^{K} H(C_i) \tag{4.11}$$

However, despite its simplicity, this measure is biased towards the number of communities. If the network is partitioned in more communities, the value of $H(C)$ tends to be higher, making difficult to compare values obtained from different partitions, if their number varies. To mitigate this effect, let's compute a weighted average defined by

$$\bar{H}(C) = \frac{\sum_{i=1}^{K} H(C_i) \times n_i}{K}, \tag{4.12}$$

where n_i is the number of nodes contained within community C_i. The rationale behind this weighted average is that large communities with small inter-community entropy are more desirable than smaller communities with the same entropy, because it means that there are more links pointing towards nodes that belong to the same community. Therefore, it is natural to look for the partition which minimizes the average inter-community entropy while varying the relax rate r.

4.2.2 Analysis of synthetic network models

To better understand the suitability and the limitations of the two measures previously described, they were applied to a set of synthetic benchmark networks. The multiplex toy models consist of 256 nodes which are connected in different ways on two layers. The networks are edge-colored because inter-layer connectivity is not given explicitly. We consider three kinds of benchmarks:

- **ER/ER.** This system is generated by combining two Erdos-Renyi (ER) networks with no community structure in either layer. This networks serves the purpose of benchmarking the measures against pure noise.

- **LFR/ER.** This system is generated by combining an Erdos-Renyi layer with one generated by the Lanchichinetti-Fortunato-Radicchi (LFR) benchmark [142]. A community structure is only present in the LFR layer, hence this multiplex network is used to test the impact of coupling noise to a structured population.

- **LFR/LFR.** This system consists of two Lanchichinetti-Fortunato-Radicchi networks, with tunable cross-layer community overlap, generated in the following way. First, a single-layered LFR network is created. Then, the network is duplicated,

and a multiplex network with two layers, where each layer is the generated LFR network, is created. After that, the neighbours of two nodes (eg: the neighbours of node 1 become the neighbours of node 5 and vice versa) are swapped. This effectively changes the community structure of the altered layer by a certain degree. If we divide the count of the nodes which neighbours were swapped by the total number of nodes, we can calculate the ratio of community overlapping across layers, that is, the percentage of nodes that belong in two different communities in different layers. When the desired overlapping ratio is reached, the neighbour swapping is swapped. In this way, three classes of networks are generated, each one corresponding to a different amount of overlapping across layers (100%, 50% and 10%, respectively).

The generated networks all have 256 nodes per layer, an average indegree of 8, and a maximum indegree of 16. LFR networks also require the input of two other parameters: the mixing parameter for the weights μ_w and topology μ_t. Both parameters were set to 0.1. Such toy network models were used to test the ability of our algorithms to detect the most relevant community structure among the ones identified by Multiplex Infomap for varying relax rates.

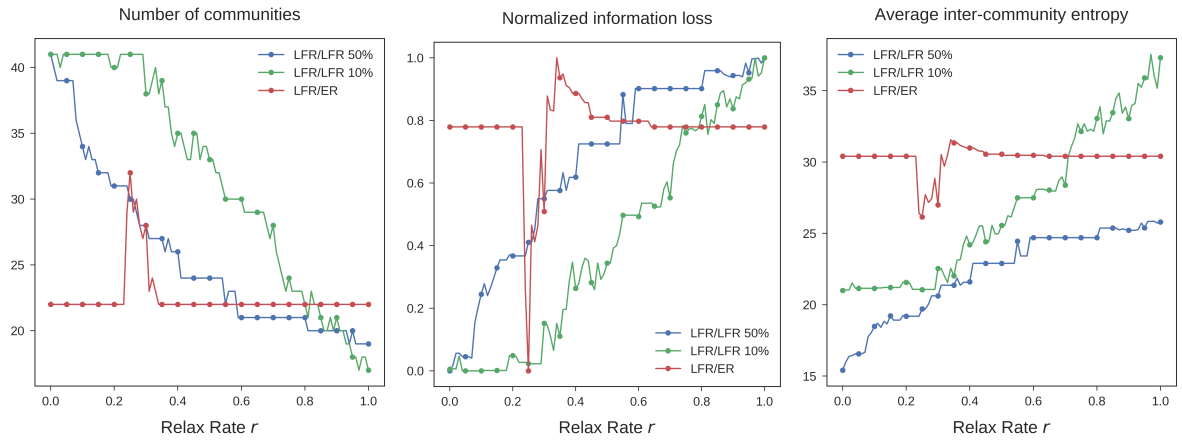
The LFR/LFR system with 100% overlap and the ER/ER network produce the same result, regardless of the relax rate. This is obvious if we look at the meaning of the relax rate: as mentioned before, it is the probability to change layer at each random walker step. In the LFR/LFR multiplex network with 100% overlap, both layers are identical, so whether the random walker changes layer or not it does not have an impact on the final result. In the case of ER/ER network instead, since both layers are completely random, it does not matter if we change layer, InfoMap still fails to find a meaningful community

structure, and lumps all nodes into a single community. Results for these benchmarks are not shown.

Results from the other synthetic network models – shown in Figure 4.3 – are more interesting, because of the increased modeling complexity with respect to the previous cases. For the LFR/LFR networks with 50% and 10% overlapping, the best value of r for both information loss and average community entropy is close to 0. Here, the Multiplex Infomap algorithm prefers to keep the two layers separated since their community structure is different enough that an evident cross-layer pattern is not detected, as expected.

The mixed LFR/ER scenario shows a very prominent change around $r = 0.25$ for both measures, suggesting that the presence of noise introduced by the ER layer forces the random walkers to switch layer more frequently to gain more information. Note that in Figure 4.3 the information loss was normalized for visualization purposes. If we name the information loss for a certain relax rate r $H(X|Y)_r$, we may define its normalized version as:

$$H^*(X|Y)_r = \frac{H(X|Y)_r - \min\limits_{0<r\leq 1} H(X|Y)_r}{\max\limits_{0<r\leq 1} H(X|Y)_r - \min\limits_{0<r\leq 1} H(X|Y)_r} \tag{4.13}$$



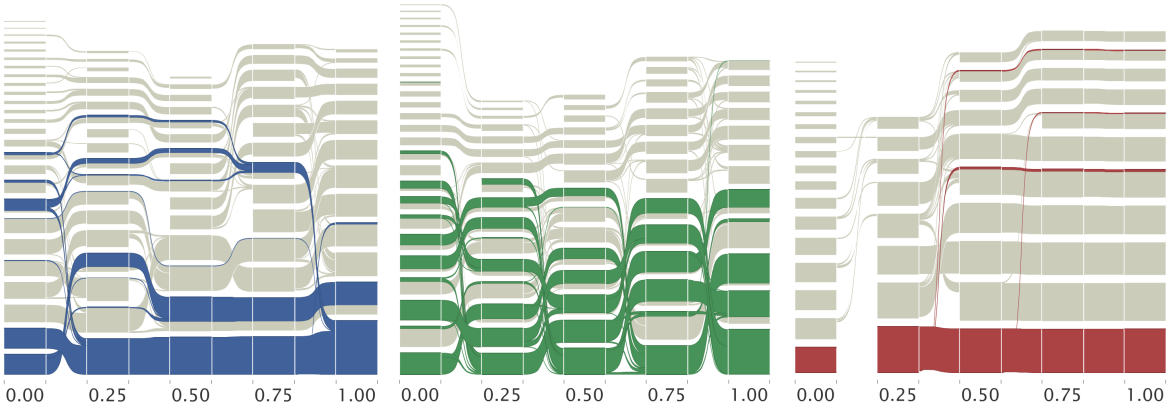
(a) (b) (c)

Figure 4.3: (a) Number of communities, (b) normalized information loss and (c) average weighted inter-community entropy, for different values of the relax rate.

To better understand these results, alluvial diagrams were built (Figure 4.4), showing how communities change when the relax rate increases.

For LFR/LFR benchmarks (Figure 4.4a and Figure 4.4b), the results are similar, with communities reconfiguring themselves more quickly in the scenario with 10% overlap. This is a consequence of the more diverse community structure in the two layers compared to the LFR/LFR network with 50% overlap.

It is of particular interest the scenario LFR/ER. When the switching dynamics is very low, the community structure of the LFR layer dominates. For $r = 0.25$ the algorithm recognizes the whole ER layer as a stand-alone community, complementing the ones detected in the LFR layer. However, when random walker's switching dynamics is too high, corresponding to the case of higher relax rates, the multiplex structure is diluted and inevitably lost, with partitions dominated by noise.



(a) (b) (c)

Figure 4.4: (a) Alluvial plots of the networks LFR/LFR with 50% overlap, (b) 10% overlap and (c) mixed LFR/ER. In each plot, each rectangle represents a community, and its thickness encodes the importance of the community within the network, evaluated by its aggregated PageRank. Flows between communities, from left to right, encode how many nodes are re-assigned from one community to another for increasing relax rate. We highlighted the flows related to the most central community for each value of r.

4.2.3 Analysis of empirical networks

We apply the algorithms to a multilayer network built from real data. In this study we consider sciMAG [143], a data set built by linking the Microsoft Academic Graph (MAG) and the SciMago classification of academic journals. sciMAG is a rather large data set: it contains more than 35 million papers and more than 320 million citations in various knowledge areas. As many other databases used for bibliometrics studies, sciMAG is affected by ambiguities among authors and institutions. The problem of disambiguation is complex and challenging. Despite it has been well-studied in the literature [144, 145, 146, 147] there is no golden standard, and many approaches are tailored around the data sets themselves. Since disambiguation is out of the scope of this paper, we decided to use the data set as it is, with the ambiguous data. We also decided to use a subset of the original data set, because in this study we are interested to show the applicability of our algorithm for identification of an optimal relax rate, rather than to perform a bibliometrics analysis that is out of the scope of the paper.

In order to make the data set usable for our purposes, besides reducing it to a subset of the original data, preprocessing steps were done to build a multiplex network with two layers, encoding citation and collaboration patterns among scholars, respectively. The preprocessing part consists of three different steps:

- *Filtering.* In order to reduce the size of the data to preprocess, all papers from the data set that do not pertain complex networks were removed. This was done by disregarding all papers that do not contain the keyword "complex networks" in their title. In this case, the crudeness of this method does not matter, as any kind of slicing of the data set would have been equally arbitrary. Nevertheless, this choice preserves any community structure that would be inherent the specific field,

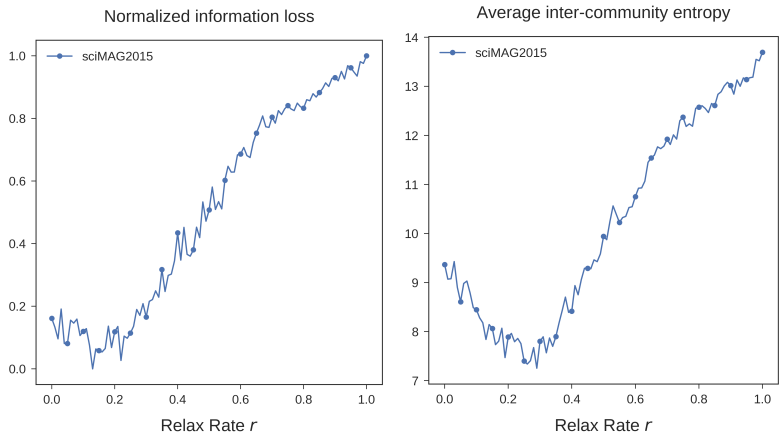
as collaboration and citation ties tend to aggregate nodes in clusters.

- *Building.* The author data with the paper citation network were joined first, in order to build the author citation network, then the paper metadata were exploited to reconstruct the author collaboration network.

- *Pruning.* The resulting network was not yet ideal for analysis, because of many "dangling" nodes – i.e., authors with no citations or collaboration links – and because the two layers do not consist of strongly connected networks. Hence, the largest strongly connected component for both layers was extracted, and all the other nodes removed. Then, nodes from one layer to another were duplicated in order to make sure that each author was present in both layers, ensuring the multiplex structure.

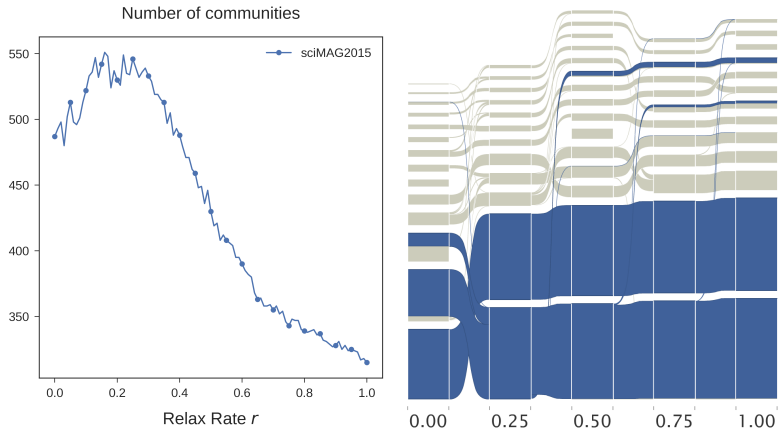
The preprocessed multiplex network consists of about 2500 authors present in both the citation and co-authorship layer. The two measures described in the previous subsections were applied, detecting minima close to 0.2 for both the information loss and the average inter-community entropy. In particular, the information loss has a minimum around 0.15 and the inter-community entropy around 0.3 (Fig. 4.5 and Fig. 4.6a).

The analysis of the corresponding alluvial plot (Fig. 4.6b) suggests the existence of two very central communities (three for relax rate 0.0, i.e., when the two layers are not coupled at all) which remain very stable for increasing r, surrounded by several smaller communities which are less stable. This result suggests that those two communities might play an important role in the research field of complex networks.



(a) (b)

Figure 4.5: (a) Information loss and (b) average weighted inter-community entropy for increasing relax rate.



(a) (b)

Figure 4.6: (a) Number of communities and (b) alluvial plot for the analysis of the sciMAG data set, as a function of the relax rate. Flows related to the most central community are highlighted for each value of r.

Chapter 5

Conclusions

The focus of this thesis has been to explore different aspects of sociality in complex networks. In the microscale, it tackled two issues, the best attachment problem, the weight assignment problem and proposed a new centrality metric, the Black Hole Metric. The best attachment problem, due to algorithmic difficulties, was approached with several heuristics. Of all the heuristics, the most successful ones were the *Anticipated_outdeg*, and the *Future* algorithms, but the *Anticipated_outdeg* is preferrable due to its reduced complexity. The weight assignment problem was approached with a social-based criteria: the neighbours' trust level should be assessed based on the node's personal experience with the neighbours. An aging mechanism was also implemented in order to give more weight to newer experiences. Different weight assignment mechanisms were compared, and the results showed that an aging mechanism makes the trust level more stable. The Black Hole Metric aims to address the normalization issue of PageRank, which flattens the arc weights, disrupting part of the information stored in the network. It has been proved that it is a generalization of PageRank that mantains the same properties, including the guarantee that it always converges. In the mesoscale, two different partition

quality assessment criteria were described: a genetic programming approach, and an information-theoretic approach. The genetic programming approach attempts to discover a validation function out of the network features, without imposing a model. While the results show that the methodology works, it still suffers from the overfitting problem, and it is effectively a work in progress. The information-theoretic approach uses two different information-centric measures, the information loss and the inter-community entropy, to determine which value of relax rate r to choose when employing the multiplex variant of the InfoMap algorithm. The two measures were tested with different networks, and were compared: in the case of the real-world network, the sciMAG database, the two measures almost agree in indicating an optimal value of r close to 0.25.

Bibliography

[1] Marco Buzzanca, Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. *Social-Based Arcs Weight Assignment in Trust Networks*, pages 143–152. Springer International Publishing, Cham, 2016.

[2] M. Buzzanca, V. Carchiolo, A. Longheu, M. Malgeri, and G. Mangioni. Direct trust assignment using social reputation and aging. *Journal of Ambient Intelligence and Humanized Computing*, 8(2):167–175, Apr 2017.

[3] M. Buzzanca, V. Carchiolo, A. Longheu, M. Malgeri, and G. Mangioni. *Dealing with the Best Attachment Problem via Heuristics*, pages 205–214. Springer International Publishing, Cham, 2017.

[4] Marco Buzzanca, Carchiolo Vincenza, Alessandro Longehu, Michele Malgeri, and Giuseppe Mangioni. Black hole metric: Overcoming the pagerank normalization problem. *Information Sciences (to be published).*

[5] Marco Buzzanca, Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. *Evaluating the community partition quality of a network with a genetic programming approach*, pages 299–308. Springer International Publishing, Cham, 2017.

[6] Marco Buzzanca, Giuseppe Mangioni, and Manlio De Domenico. Optimal community detection in non-interconnected multiplex networks. *Algorithms (to be published).*

[7] N. L. Chervany D. H. McKnight. The meanings of trust. Technical report, Minneapolis, USA, 1996.

[8] *Trust in Modern Societies*. Polity Press, 1996.

[9] T. Grandison and M. Sloman. A survey of trust in internet application. *IEEE Communication Surveys and Tutorials*, 4(4):2–16, 2000.

[10] Cynthia L. Corritore, Beverly Kracher, and Susan Wiedenbeck. On-line trust: concepts, evolving themes, a model. *International Journal of Human-Computer Studies*, 58(6):737–758, 2003. Trust and Technology.

[11] J. Golbeck. Trust and nuanced profile similarity in online social networks. *ACM Transactions on the Web, to appear*, 2008.

[12] N. Luhmann. *Trust and Power*. Wiley, 1979.

[13] Sini Ruohomaa and Lea Kutvonen. Trust management survey. In *proceedings of ITRUST 2005, number 3477 in LNCS*, pages 77–92. Springer-Verlag, 2005.

[14] Gema Bello-Orgaz, Jason J. Jung, and David Camacho. Social big data: Recent achievements and new challenges. *Information Fusion*, 28:45 – 59, 2016.

[15] Jason J. Jung. Measuring trustworthiness of information diffusion by risk discovery process in social networking services. *Quality & Quantity*, 48(3):1325–1336, 2013.

[16] Dariusz Król. *Intelligent Information and Database Systems: 8th Asian Conference, ACIIDS 2016, Da Nang, Vietnam, March 14-16, 2016, Proceedings, Part I*, chapter Measuring Propagation Phenomena in Social Networks: Promising Directions and Open Issues, pages 86–94. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[17] Christian Damsgaard Jensen. *Trust Management VIII: 8th IFIP WG 11.11 International Conference, IFIPTM 2014, Singapore, July 7-10, 2014. Proceedings*, chapter The Importance of Trust in Computer Security, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[18] S. Song, Kai Hwang, Runfang Zhou, and Yu-Kwong Kwok. Trusted p2p transactions with fuzzy reputation aggregation. *Internet Computing, IEEE*, 9(6):24–34, Nov 2005.

[19] Ye Diana Wang and Henry H. Emurian. An overview of online trust: Concepts, elements, and implications. *Computers in Human Behavior*, 21(1):105–125, 2005.

[20] Radu Burete, Amelia Badica, Costin Badica, and Florin Moraru. Enhanced reputation model with forgiveness for e-business agents. *IJATS*, 3(1):11–26, 2011.

[21] Florina Almenárez, Andrés Marín, Celeste Campo, and Carlos García R. PTM: A pervasive trust management model for dynamic open environments. In *First workshop on pervasive security, privacy and trust PSPT'04 in conjuntion with MOBIQUITOUS*, 2004.

[22] Gregor von Laszewski, Beulah Alunkal, and I. Veljkovic. Toward reputable grids. *Scalable Computing: Practice and Experience*, 6(3), 2005.

[23] Ryan Wishart, Ricky Robinson, Jadwiga Indulska, and Audun Jøsang. Superstringrep: Reputation-enhanced service discovery. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science - Volume 38*, ACSC '05, pages 49–57, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[24] Yanchao Zhang and Yuguang Fang. A fine-grained reputation system for reliable service selection in peer-to-peer networks. *Parallel and Distributed Systems, IEEE Transactions on*, 18(8):1134–1145, Aug 2007.

[25] Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. The cost of trust in the dynamics of best attachment. *Computing & Informatics*, 34(1).

[26] Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. Trust assessment: a personalized, distributed, and secure approach. *Concurrency and Computation: Practice and Experience*, 24(6):605–617, 2012.

[27] Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. Users attachment in trust networks: Reputation vs. effort. *Int. J. Bio-Inspired Comput.*, 5(4):199–209, July 2013.

[28] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, 1994.

[29] Jens Riegelsberger, M. Angela Sasse, and John D. McCarthy. The mechanics of trust: A framework for research and design. *International Journal of Human-Computer Studies*, 62(3):381 – 422, 2005.

[30] Frank E. Walter, Stefano Battiston, and Frank Schweitzer. A model of a trust-based recommendation system on a social network. *Journal of autonomous agents and multi-agent systems*, 16(1):57, 2008.

[31] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71, 2007.

[32] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the Twelfth International World Wide Web Conference, 2003.*, 2003.

[33] Li Xiong and Ling Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. Knowl. Data Eng.*, 16(7):843–857, 2004.

[34] Runfang Zhou, Kai Hwang, and Min Cai. GossipTrust for fast reputation aggregation in peer-to-peer networks. *IEEE Transactions on Knowledge and Data Engineering*, 20(9):1282–1295, 2008.

[35] Frank Edward Walter, Stefano Battiston, and Frank Schweitzer. Personalised and dynamic trust in social networks. In Lawrence D. Bergman, Alexander Tuzhilin, Robin D. Burke, Alexander Felfernig, and Lars Schmidt-Thieme, editors, *RecSys*, pages 197–204. ACM, 2009.

[36] Paolo Massa and Paolo Avesani. Controversial users demand local trust metrics: An experimental study on epinions.com community. In *AAAI*, pages 121–126, 2005.

[37] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24, New York, NY, USA, 2007. ACM.

[38] Antonio J. Roa-Valverde and Miguel-Angel Sicilia. A survey of approaches for ranking on the web of data. *Inf. Retr.*, 17(4):295–325, August 2014.

[39] Bing Pan, Helene Hembrooke, Thorsten Joachims, Lori Lorigo, Geri Gay, and Laura Granka. In google we trust: Users decisions on rank, position, and relevance. *Journal of Computer-Mediated Communication*, 12(3):801–823, 2007.

[40] Vijay Chauhan, Arunima Jaiswal, and Junaid Khan. Web page ranking using machine learning approach. In *Advanced Computing Communication Technologies (ACCT), 2015 Fifth International Conference on*, pages 575–580, Feb 2015.

[41] Vincenza Carchiolo, Alessandro Longheu, and Michele Malgeri. Reliable peers and useful resources: Searching for the best personalised learning path in a trust- and recommendation-aware environment. *Information Sciences*, 180(10):1893 – 1907, 2010. Special Issue on Intelligent Distributed Information Systems.

[42] Jesus Serrano-Guerrero, Enrique Herrera-Viedma, José Olivas, Andres Cerezo, and Francisco Romero. A google wave-based fuzzy recommender system to disseminate information in university digital libraries 2.0. *Information Sciences*, 181:1503–1516, 05 2011.

[43] Jesus Serrano-Guerrero, Francisco Romero, and José Olivas. Hiperion: A fuzzy approach for recommending educational activities based on the acquisition of competences. *Information Sciences*, pages –, 11 2013.

[44] Li Chen, Guanliang Chen, and Feng Wang. Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154, June 2015.

[45] Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. Searching for experts in a context-aware recommendation network. *Computers in Human Behavior*, pages –, 2015.

[46] Young Ae Kim and Rasik Phalak. A trust prediction framework in rating-based experience sharing social networks without a web of trust. *Information Sciences*, 191:128–145, 2012.

[47] Punam Bedi and Pooja Vashisth. Empowering recommender systems using trust and argumentation. *Information Sciences*, 279:569–586, 2014.

[48] Jianshu Weng, Chunyan Miao, Angela Goh, Zhiqi Shen, and Robert Gay. Trust-based agent community for collaborative recommendation. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1260–1262, New York, NY, USA, 2006. ACM.

[49] Xin Liu. Towards context-aware social recommendation via trust networks. In Xuemin Lin, Yannis Manolopoulos, Divesh Srivastava, and Guangyan Huang, editors, *Web Information Systems Engineering - WISE 2013*, volume 8180 of *Lecture Notes in Computer Science*, pages 121–134. Springer Berlin Heidelberg, 2013.

[50] R Fung and M. Lee. Ec-trust (trust in electronic commerce): Exploring the antecedent factors. In *Proceedings of the 5th Americas Conference on Information Systems*, pages 517–519, 1999.

[51] Pathan Sameerkhan, Khan Shahrukh, Arshad Mohammad, Ahmad Amir, and Anand Bali. Comment based grading and rating system in e-commerce. *International Journal of Engineering Research and General Science*, 3(1):1319–1322, 2015.

[52] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*, 1998.

[53] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[54] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. Wtf: The who to follow service at twitter. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 505–514, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[55] David F. Gleich. Pagerank beyond the web. *CoRR*, abs/1407.5107, 2014.

[56] Upul Senanayake, Mahendra Piraveenan, and Albert Zomaya. The pagerank-index: Going beyond citation counts in quantifying scientific impact of researchers. *PLoS ONE*, 10(8):e0134794, 08 2015.

[57] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of ACM*, 46(5):604–632, 1999.

[58] R. Lempel and S. Moran. Salsa: The stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, April 2001.

[59] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Trans. Parallel Distrib. Syst.*, 18(4):460–473, 2007.

[60] Ashutosh Kumar Singh and Ravi Kumar P. A comparative study of page ranking algorithms for information retrieval. 3(4):745 – 756, 2009.

[61] S. Prabha, K. Duraiswamy, and J. Indhumathi. Comparative analysis of different page ranking algorithms. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 8(8):1486 – 1494, 2014.

[62] Marc A. Najork. Comparing the effectiveness of hits and salsa. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 157–164, New York, NY, USA, 2007. ACM.

[63] HyunChul Lee and Allan Borodin. Perturbation of the hyper-linked environment. In Tandy Warnow and Binhai Zhu, editors, *Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 272–283. Springer Berlin Heidelberg, 2003.

[64] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[65] Leon Danon, Jordi Duch, Albert Diaz-Guilera, and Alex Arenas. Comparing community structure identification, 2005.

[66] Santo Fortunato and Claudio Castellano. *Community Structure in Graphs*. Encyclopedia of Complexity and System Science. Springer, 2008.

[67] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[68] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[69] B. H. Good, Y.-A. de Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. 81(4):046106, April 2010.

[70] Jörg Reichardt and Stefan Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Physical Review Letters*, 93(21):218701, 2004.

[71] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2):027104, 2005.

[72] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3–5):75 – 174, 2010.

[73] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[74] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *PNAS*, 105(4):1118–1123, 2008.

[75] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, 2009.

[76] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[77] Roger Guimera and Luis A Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

[78] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[79] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.

[80] Danielle S Bassett, Nicholas F Wymbs, Mason A Porter, Peter J Mucha, Jean M Carlson, and Scott T Grafton. Dynamic reconfiguration of human brain networks during learning. *Proceedings of the National Academy of Sciences*, 108(18):7641–7646, 2011.

[81] Matteo Barigozzi, Giorgio Fagiolo, and Giuseppe Mangioni. Identifying the community structure of the international-trade multi-network. *Physica A: Statistical Mechanics and its Applications*, 390(11):2051 – 2066, 2011.

[82] Matteo Barigozzi, Giorgio Fagiolo, and Giuseppe Mangioni. *Community Structure in the Multi-network of International Trade*, pages 163–175. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[83] Jason Bennet Thatcher, D. Harrison McKnight, Elizabeth White Baker, Riza Ergun Arsal, and Nicolas H. Roberts. The role of trust in postadoption IT exploration an empirical examination of knowledge management systems. *IEEE Transactions on Engineering Management*, 58(1):56–70, 2011.

[84] Jingwei Huang and Mark S. Fox. An ontology of trust: formal semantics and transitivity. In *Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet*, ICEC '06, pages 259–270, New York, NY, USA, 2006. ACM.

[85] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 403–412, New York, NY, USA, 2004. ACM.

[86] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World Wide Web*, pages 641–650, 2010.

[87] C. De Kerchove and P. Van Dooren. The PageTrust algorithm: how to rank web pages when negative links are allowed? In *Proceedings of the of the SIAM International Conference on Data Mining*, pages 346–352, 2008.

[88] Lee Averell and Andrew Heathcote. The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psychology*, 55(1):25 – 35, 2011. Special Issue on Hierarchical Bayesian Models.

[89] Vladimir Batagelj and Andrej Mrvar. Pajek - program for large network analysis. *Connections*, 21(2):47–57, 1998.

[90] Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. Gain the best reputation in trust networks. In F.M.T. Brazier, Kees Nieuwenhuis, Gregor Pavlin, Martijn Warnier, and Costin Badica, editors, *Intelligent Distributed*

Computing V, volume 382 of *Studies in Computational Intelligence*, pages 213–218. Springer Berlin Heidelberg, 2012.

[91] Pavel Berkhin. A survey on pagerank computing. *Internet Mathematics*, 2:73–120, 2005.

[92] Martin Olsen, Anastasios Viglas, and Ilia Zvedeniouk. An approximation algorithm for the link building problem. *CoRR*, abs/1204.1369, 2012.

[93] Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. A heuristic to explore trust networks dynamics. In Filip Zavoral, Jason J. Jung, and Costin Badica, editors, *Intelligent Distributed Computing VII*, volume 511 of *Studies in Computational Intelligence*, pages 67–76. Springer International Publishing, 2014.

[94] K. Avrachenkov and N. Litvak. The effect of new links on google pagerank. *Stochastic Models*, 22(2):319–331, 2006.

[95] Cristobald de Kerchove, Laure Ninove, and Paul Van Dooren. Maximizing pagerank via outlinks. *CoRR*, abs/0711.2867, 2007.

[96] Olivier Fercoq, Marianne Akian, Mustapha Bouhtou, and Stephane Gaubert. Ergodic control and polyhedral approaches to pagerank optimization. *IEEE Trans. Automat. Contr.*, 58(1):134–148, 2013.

[97] Marcin Sydow. Can one out-link change your pagerank? In Piotr S. Szczepaniak, Janusz Kacprzyk, and Adam Niewiadomski, editors, *AWIC*, volume 3528 of *Lecture Notes in Computer Science*, pages 408–414. Springer, 2005.

[98] Monica Bianchini, Marco Gori, and Franco Scarselli. Inside pagerank. *ACM Trans. Internet Technol.*, 5(1):92–128, February 2005.

[99] Reka Albert and Albert-Laszlo Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.

[100] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[101] David M. Pennock, Gary W. Flake, Steve Lawrence, Eric J. Glover, and C. Lee Giles. Winners don't take all: Characterizing the competition for links on the web. *Proceedings of the National Academy of Sciences*, 99(8):5207–5211, 2002.

[102] Luca Pretto. A theoretical analysis of google's pagerank. In AlbertoH.F. Laender and ArlindoL. Oliveira, editors, *String Processing and Information Retrieval*, volume 2476 of *Lecture Notes in Computer Science*, pages 131–144. Springer Berlin Heidelberg, 2002.

[103] Amy N. Langville and Carl D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1:2004, 2004.

[104] A. O. Zhirov, O. V. Zhirov, and D. L. Shepelyansky. Two-dimensional ranking of wikipedia articles. *CoRR*, abs/1006.4270, 2010.

[105] HyunChul Lee and Allan Borodin. Perturbation of the hyper-linked environment. In Tandy Warnow and Binhai Zhu, editors, *Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 272–283. Springer Berlin Heidelberg, 2003.

[106] Mathew Richardson and Pedro Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[107] Paul Erdős and Alfréd Rényi. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959 1959.

[108] Béla Bollobás, Christian Borgs, Jennifer Chayes, and Oliver Riordan. Directed scale-free graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 132–139, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[109] Jérôme Kunegis. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*, pages 1343–1350, 2013.

[110] Advogato network dataset – KONECT, January 2016.

[111] Paolo Massa, Martino Salvetti, and Danilo Tomasoni. Bowling alone and trust decline in social network sites. In *Proc. Int. Conf. Dependable, Autonomic and Secure Computing*, pages 658–663, 2009.

[112] Mark EJ Newman. Communities, modules and large-scale structure in networks. *Nature Physics*, 8(1):25–31, 2012.

[113] Riccardo Poli, William B. Langdon, and Nicholas F. Mcphee. *A field guide to genetic programming*. March 2008.

[114] Wolfgang Banzhaf, Frank D. Francone, Robert E. Keller, and Peter Nordin. *Genetic Programming: An Introduction: on the Automatic Evolution of Computer Programs*

and Its Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[115] Junjie Wu, Hui Xiong, and Jian Chen. Adapting the right measures for k-means clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 877–886, New York, NY, USA, 2009. ACM.

[116] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, 1992.

[117] Sean Luke. A comparison of crossover and mutation in genetic programming. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 240–248. Morgan Kaufmann, 1997.

[118] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[119] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[120] P.A. Whigham and Grant Dick. Implicitly controlling bloat in genetic programming. 14:173 – 190, 05 2010.

[121] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203, 2014.

[122] S. Boccaletti, G. Bianconi, R. Criado, C.I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilnetworks. *Physics Reports*, 544(1):1 – 122, 2014. The structure and dynamics of multilayer networks.

[123] Manlio De Domenico, Clara Granell, Mason A Porter, and Alex Arenas. The physics of spreading processes in multilayer networks. *Nature Physics*, 12:901–906, 2016.

[124] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[125] M De Domenico, V Nicosia, A Arenas, and V Latora. Structural reducibility of multilayer networks. *Nature communications*, 6:6864, 2015.

[126] Marina Diakonova, Vincenzo Nicosia, Vito Latora, and Maxi San Miguel. Irreducibility of multilayer network dynamics: the case of the voter model. *New Journal of Physics*, 18(2):023010, 2016.

[127] Federico Battiston, Vincenzo Nicosia, and Vito Latora. Structural measures for multiplex networks. *Phys. Rev. E*, 89:032804, Mar 2014.

[128] M De Domenico, A Solé-Ribalta, E Omodei, S Gómez, and A Arenas. Ranking in interconnected multilayer networks reveals versatile nodes. *Nature communications*, 6:6868, 2015.

[129] Emanuele Cozzo, Mikko Kivelä, Manlio De Domenico, Albert Solé-Ribalta, Alex Arenas, Sergio Gómez, Mason A Porter, and Yamir Moreno. Structure of triadic relations in multiplex networks. *New Journal of Physics*, 17(7):073029, 2015.

[130] Jacopo Iacovacci, Christoph Rahmede, Alex Arenas, and Ginestra Bianconi. Functional multiplex pagerank. *EPL (Europhysics Letters)*, 116(2):28004, 2016.

[131] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[132] J-C Delvenne, Sophia N Yaliraki, and Mauricio Barahona. Stability of graph communities across time scales. *Proceedings of the National Acdomenico of Sciences*, 107(29):12755–12760, 2010.

[133] Renaud Lambiotte, Jean-Charles Delvenne, and Mauricio Barahona. Random walks, markov processes and the multiscale modular organization of complex networks. *IEEE Transactions on Network Science and Engineering*, 1(2):76–90, 2014.

[134] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[135] Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, and Giuseppe Mangioni. *Communities Unfolding in Multislice Networks*, pages 187–195. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[136] Manlio De Domenico, Albert Solé-Ribalta, Sergio Gómez, and Alex Arenas. Navigability of interconnected networks under random failures. *Proceedings of the National Academy of Sciences*, 111(23):8351–8356, 2014.

[137] Etay Ziv, Manuel Middendorf, and Chris H. Wiggins. Information-theoretic approach to network modularity. *Phys. Rev. E*, 71:046117, Apr 2005.

[138] Martin Rosvall and Carl T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331, 2007.

[139] Tiago P. Peixoto. Parsimonious module inference in large networks. *Phys. Rev. Lett.*, 110:148701, Apr 2013.

[140] Yongli Li, Guijie Zhang, Yuqiang Feng, and Chong Wu. An entropy-based social network community detecting method and its application to scientometrics. *Scientometrics*, 102(1):1003–1017, 2015.

[141] Tiago P. Peixoto. Entropy of stochastic blockmodel ensembles. *Physical Review E*, 85(5), May 2012.

[142] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, October 2008.

[143] Manlio De Domenico, Elisa Omodei, and Alex Arenas. Quantifying the diaspora of knowledge in the last century. *Applied Network Science*, 1(1):15, 2016.

[144] C Lee Giles, Hongyuan Zha, and Hui Han. Name disambiguation in author citations using a k-way spectral clustering method. In *Digital Libraries, 2005. JCDL'05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on*, pages 334–343. IEEE, 2005.

[145] In-Su Kang, Seung-Hoon Na, Seungwoo Lee, Hanmin Jung, Pyung Kim, Won-Kyung Sung, and Jong-Hyeok Lee. On co-authorship for author disambiguation. *Information Processing & Management*, 45(1):84–97, 2009.

[146] Li Tang and John P Walsh. Bibliometric fingerprints: name disambiguation based on approximate structure equivalence of cognitive maps. *Scientometrics*, 84(3):763–784, 2010.

[147] Christian Schulz, Amin Mazloumian, Alexander M Petersen, Orion Penner, and Dirk Helbing. Exploiting citation networks for large-scale author name disambiguation. *EPJ Data Science*, 3(1):11, 2014.