Spatial search by continuous-time quantum walks on complex networks

Master's Thesis
University of Turku
Department of Physics
and Astronomy
Theoretical Physics
November 2020
BSc. Joonas Malmi
Supervisors:
Matteo Rossi
Teiko Heinosaari

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -järjestelmällä.

UNIVERSITY OF TURKU
Department of Physics and Astronomy

MALMI, JOONAS: Spatial search by continuous-time quantum walks on complex networks

Master's Thesis, 70 p.
Theoretical Physics
November 2020

Spatial search by continuous-time quantum walks on complex networks is focused on using a quantum walk in continuous time in order to find a single or multiple marked vertices within the complex network. The specific formalism used here is to consider a coupling constant that shifts the state of the quantum walker from the initial state to the target state, which is the marked vertex.

The thesis begins with establishing the mathematical framework of network theory, quantum walks and numerical methods that will be used in the remainder of the thesis. Then spatial search by continuous-time quantum walk is studied on regular and semi-regular graphs, where most analytical results can be found. This will get us acquainted with spatial search by quantum walk. The complex networks studied are Barabási-Albert graphs and the Internet network on the level of autonomous systems. Different renormalized and pruned versions of the Internet network are studied. The parameters of the quantum walk that are focused on are the optimal values for the coupling constant, success probability, time and search time.

Keywords: continuous-time quantum walk, complex network, spatial search, network theory, quantum computing

TURUN YLIOPISTO
Fysiikan ja tähtitieteen laitos

MALMI, JOONAS: Kvanttikulkujen spatiaalinen etsintä jatkuvassa ajassa kompleksisissa verkoissa

Pro Gradu, 70 s.
Teoreettinen fysiikka
Marraskuu 2020

Kvanttikulkujen spatiaalinen etsintä jatkuvassa ajassa kompleksisissa verkoissa keskittyy yhden tai useamman merkityn solmukohdan löytämiseen kompleksisesta verkosta käyttämällä kvanttikulkua jatkuvassa ajassa. Tässä työssä käytetty formalismi käsittelee kytkentävakiota, mikä siirtää kvanttikulkijan tilan alkutilasta tavoitetilaan, eli merkittyyn solmukohtaan.

Tämä Pro Gradu alkaa matemaattisen viitekehyksen käsittelemisellä, jota tarvitaan lopputyössä. Tämä viitekehys sisältää verkkoteorian, kvanttikulut ja käytetyt numeeriset menetelmät. Tämän jälkeen kvanttikulun spatiaalista etsintää jatkuvassa ajassa tutkitaan säännöllisissä ja miltei säännöllisissä verkoissa, missä analyyttiset ratkaisut on löydettävissä. Tämän tarkoituksena on tutustua spatiaaliseen etsintään kvanttikululla. Barabási-Albert -graafit ja Internet-verkko autonomisten järjestelmien tasolla ovat tässä työssä tutkittavat kompleksiset verkot. Tässä tutkitaan eri renormalisoituja ja karsittuja versioita Internet-verkosta. Kvanttikulun parametrit, joihin keskitytään, ovat optimaaliset arvot kytkentävakiolle, onnistumistodennäköisyydelle, ajalle ja etsintäajalle.

Avainsanat: kvanttikulku jatkuvassa ajassa, kompleksinen verkko, spatiaalinen etsintä, verkkoteoria, kvanttiohjelmointi

Contents

Introduction

Both network theory and quantum walks are quite new fields of study, only having been around approximately since the 1990s, although the exact date can be argued [1, 4]. Network theory has its mathematical background in graph theory and it has since found many applications in almost all fields of science, most notably in our case in physics and computer science. Network theory focuses on utilizing the mathematical tools found in graph theory and beyond, and applying these tools in order to find hidden laws and patterns in real-world networks like social networks, power grids and the Internet. Network theory has received extensive use in many fields of science in the 21st century, since there are networks or network-like structures that can be approximated as such everywhere around us [1].

Network theory as a mathematical discipline can be traced to the 18th century, when Leonhard Euler gave the solution to the *Seven Bridges of Königsberg* problem [1]. This is understood as the beginning of graph theory. After this, graph theory has been an area of study since the 19th century, but it has largely focused on small graphs, since larger ones are exponentially more difficult to analyse. Therefore one of the major reasons why network theory has only become significant recently is because computers have offered an efficient way to analyse large networks.

Quantum walks are the quantum mechanical equivalent to classical random walks, which have been studied for over a century and have found great applicability in many fields of science, for example simulating Brownian motion in physics or estimating the size of the World Wide Web in computer science. Quantum walks have also led to good working results and they also are a viable option for quantum computation [31]. Other uses include for example quantum state transfer and transport and energy transport in biological systems [11, 12].

Quantum walks originally stemmed from classical random walks that have been studied from the beginning of the 19th century onwards. Quantum random walks in discrete time were introduced in 1993 by Aharonov, Davidovich and Zagury [8]. A major result is that the average path length traversed by a quantum walker in

a given time is much larger as opposed to a corresponding classical walker due to quantum interference. Quantum walks in continuous time were introduced in 1998 by Farhi and Gutmann [9]. They also considered the quantum walk on decision trees, so a graph structure was used as the space where the walker was defined. In fact in most cases quantum walks are defined on graphs as they offer a natural way to describe states and allowed transitions between states.

Spatial search by quantum walk was first described by Childs and Goldstone in 2004 [2]. Childs and Goldstone considered this as a continuous-time, physical implementation of Grover's algorithm, which offers a quadratic speed-up for the search of a database when compared to a classical algorithm [3]. In this case a single marked node in the graph is considered, and the quantum walker tries to find the marked node in the graph. As can be seen, the importance of network theory in quantum walks and spatial search by quantum walks is clear. The behaviour of spatial search by quantum walks on complex networks is also a very new area of study with many opportunities and useful applications, as quantum technologies are becoming viable [16–19].

The goal of this thesis is to study the efficiency of spatial search by continuous-time quantum walks on complex networks, especially studying what properties of the networks affect the efficiency of the search. The first chapter introduces the mathematical tools needed for the remainder of the thesis. In the second chapter we look at spatial search by continuous-time quantum walks on regular and semi-regular graphs, so that the behaviour of the spatial search algorithm can be understood on simpler graphs. The third and final chapter contains the results, where the spatial search by quantum walk is studied on complex networks.

Natural units are used throughout the thesis, setting $\hbar = c = k_B = 1$, and $\mathcal{H}$ will always be defined as a complex Hilbert space.

Chapter I

The Mathematical Framework

This chapter introduces the mathematical framework that is required for the thesis. These are split into three main parts: network theory, quantum walks and specific numerical methods used. Network theory is introduced first since it can be discussed completely separately from quantum walks. Quantum walks are introduced as the quantum mechanical equivalent for random walks and the significance of network theory in the context of quantum walks is also discussed. The specific numerical methods are concerned with the computational side of this thesis, i.e. how the results were calculated.

1 Network theory

Network theory is a relatively new branch of mathematics and computer science that deals with systems that are described or can be described with a set of points and lines connecting these points. The points are called vertices or nodes and the lines are called edges or links. An example of a network can be seen in figure 1. For this section on network theory, Newman's book on network theory is used as the main source of information [1].

Network theory has its roots in graph theory, which is a field of mathematics studied since the 18th century. Most of the mathematical formalism used in network theory comes from graph theory, and the difference between the two is not well-defined. Graph theory is strictly mathematical and the graphs studied are hypothetical structures where the graph theorist tries to create proofs using such structures. Networks are typically real-world structures abstracted into graph-like systems, and the parts of a network might have additional information attached to them. In this way network theory is a part of graph theory, but the questions that

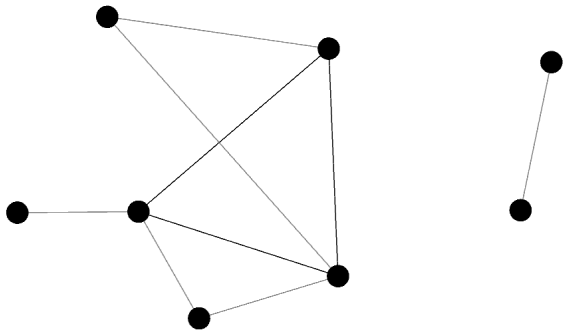


Figure 1: An example of a small undirected network with 8 vertices and 9 edges.

graph theorists and network theorists want to answer are not the same most of the time.

Networks can also be undirected or directed, and unweighted or weighted. In an undirected network the edges between nodes do not have any directionality and they can be traversed to both directions. In a directed network the edges have directionality so just because one can get from one vertex to another that does not necessarily mean that the reverse is possible. In a weighted network each edge has a weight or a value connected to it which can symbolize for example the cost of traversing that edge. In an unweighted network the weights of all edges can be thought of as being unitary. In this paper only undirected and unweighted networks will be considered.

After this point I will use the terms graph and network interchangeably when referring to the abstract mathematical structure.

1.1 Mathematical formalism

In this thesis the number of vertices in a graph is denoted as n or N and the number of edges as m, which is a common notation in the mathematical literature. The vertices of a graph can be denoted as integer labels $1, ..., n$ or as $v_1, ..., v_n$. The ordering of the labels does not matter, as long as they are unique and consistent. Likewise an edge between nodes i and j will be denoted as (i, j).

A graph G is defined as $G \equiv (V, E)$, where V is the set of vertices and E is the set of edges. The structure of the graph can be expressed for example with either

an adjacency list or an adjacency matrix. An adjacency list is a list of lists where each vertex has a list of vertices it is adjacent to. An adjacency matrix is an $n \times n$ -matrix, where rows represent vertices i and columns represent vertices j. Adjacency lists are typically used for computing purposes because they take $\mathcal{O}(n+m)$ space while adjacency matrices take $\mathcal{O}(n^2)$ space. Adjacency matrices are used for greater ease in mathematical formulations and calculations, and in this paper adjacency matrices will be used for the most part.

In an undirected network the adjacency matrix is defined as

$$A_{ij} = \begin{cases} 1, & \text{if there is an edge between vertices } i \text{ and } j, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

In the case of figure 2 the adjacency matrix would be

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

From this it can also be easily seen that the adjacency matrix of an undirected graph will always be symmetric. This is because the edge (i, j) connecting vertices i and j is the same as the edge (j, i). It is also good to note that for graphs with no self-edges, i.e. edges which have both ends at the same vertex, all the diagonal elements of the adjacency matrix are 0.

Let us define the degree of a vertex as the amount of adjacent vertices it has in the case of an unweighted undirected graph. The degree of vertex 1 is $\deg(v_1) = k_1 = 3$ and the degree of vertex 4 is $\deg(v_4) = k_4 = 1$. More generally, the degree of a vertex can be written with the adjacency matrix as

$$k_i = \sum_{j=1}^{n} A_{ij}.$$

1.2 Dynamics in a network and the Laplacian

Dynamics in a network can be thought of for example in terms of diffusion. Let's say that there is some flow of a substance that can move in the network from one vertex to another, and that there is an amount ψ_i of it at vertex i. This amount can also flow along the edges from one vertex to another. Then the flow from vertex j to

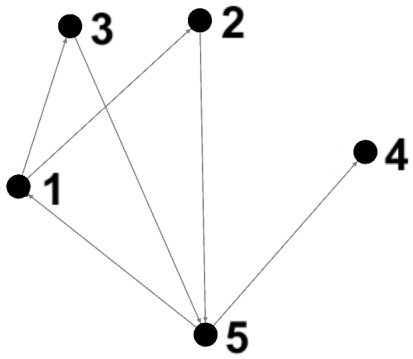


Figure 2: A graph with labels for the vertices.

an adjacent one i can be written as $C(\psi_j - \psi_i)$, where C is some diffusion constant. In an infinitesimal interval of time the flow is then $C(\psi_j - \psi_i)\mathrm{d}t$. The change of the amount of substance in ψ_i is then given as

$$\begin{aligned}\frac{\mathrm{d}\psi_i}{\mathrm{d}t} &= C\sum_j A_{ij}(\psi_j - \psi_i) = C\big(\sum_j A_{ij}\psi_j - \psi_i \sum_j A_{ij}\big) \\ &= C\big(\sum_j A_{ij}\psi_j - \psi_i k_i\big) = C\sum_j (A_{ij} - \delta_{ij}k_i)\psi_j,\end{aligned}$$

where all the adjacent nodes to i are summed over. This can also be written in matrix form as

$$\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}t} = C(\mathbf{A} - \mathbf{D})\boldsymbol{\psi}.$$

From here we can define a matrix known as the Laplacian as[1]

$$\mathbf{L} \equiv \mathbf{D} - \mathbf{A}, \tag{2}$$

where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is the diagonal matrix defined as

$$D_{ij} \equiv \begin{cases} k_i, & i = j \\ 0, & i \neq j. \end{cases}$$

The Laplacian has the following property that will be useful later.

[1]The Laplacian is sometimes also defined as $\mathbf{L} \equiv \mathbf{A} - \mathbf{D}$.

Lemma 1. *Let G be an undirected graph and let L be the Laplacian of this graph. The eigenvalues can then be ordered in such a way that $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ and $\lambda_1 = 0$.*

Proof. To prove this let's create an $m \times n$ –matrix called the edge incidence matrix B_{ev}, where for every edge e and vertex v

$$B_{ij} = \begin{cases} 1, & \text{if } v = i \\ -1, & \text{if } v = j \\ 0, & \text{otherwise.} \end{cases}$$

Now let's consider the sum $\sum_k B_{ki}B_{kj}$. For every $i \neq j$ the value is non-zero if and only if there is an edge connecting i and j. In that case this sum will equal to -1. If $i = j$, then the sum goes over all the connections of i, in which case the sum will equal to the degree k_i. Thus $\sum_k B_{ki}B_{kj} = L_{ij}$ or in matrix form

$$\mathbf{L} = \mathbf{B}^T\mathbf{B}.$$

Now the eigenvalues λ_i of $\mathbf{L}$ can be written in the form

$$\begin{aligned} \lambda_i &= \mathbf{v}_i^T\mathbf{L}\mathbf{v}_i \\ &= \mathbf{v}_i^T\mathbf{B}^T\mathbf{B}\mathbf{v}_i \\ &= (\mathbf{B}\mathbf{v}_i)^T\mathbf{B}\mathbf{v}_i. \end{aligned}$$

All the elements of the real vector $\mathbf{B}\mathbf{v}_i$ are thus squared and summed together, meaning that

$$\lambda_i \geq 0,$$

so $\mathbf{L}$ is positive semi-definite.

Now consider a vector $\mathbf{1} = (1, 1, ..., 1)^T$. Multiplying the Laplacian with it gives

$$\mathbf{L1} = \begin{pmatrix} k_1 - k_1 \\ k_2 - k_2 \\ \vdots \\ k_n - k_n \end{pmatrix} = \mathbf{0},$$

which means that $\lambda_1 = 0$. □

This feature of the Laplacian is the main reason for why the Laplacian will be used in defining the dynamics of continuous-time random walks later in the thesis, as opposed to the adjacency matrix.

1.3 Random graphs

Generally, *random graphs* are network models where specific sets of parameters have fixed values and other parameter values are randomized [1]. Typically random graphs are generated from a set of rules, and depending on these rules, the resulting graph will be a certain type of graph with a certain probability. By changing the set of rules, vastly different graphs can be generated, and for a large number of nodes graphs generated with the same rules will almost always have the same type of overall structure.

One of the most common types of random graphs are generated with the *Erdős–Rényi (ER) model* which was introduced by Paul Erdős and Alfréd Rényi in 1959 [23]. The model can be defined in two separate ways that are equivalent in the types of graphs they generate. The first way is to take a graph $G(n, m)$ with n nodes and m edges, and then to assign these edges randomly between pairs of nodes. The second way is to take a graph $G(n, p)$ with n nodes and a probability p of generating an edge between each pair of nodes. This model generates graphs that are typically very random and do not have any significant overall structure in the case of large n.

Another model with different types of generated graphs is the *Barabási–Albert (BA) model*[2] introduced by Albert-Lâszló Barabási and Réka Albert in 2002 [24]. This model generates graphs via *preferential attachment*, in which a new node introduced to the graph is added to the existing nodes preferentially depending on the degrees of the existing nodes. The model can be defined by taking a graph $G(n, n_0)$ with n_0 initial nodes (these nodes can be either connected or disconnected, in the case of large n this does not make much of a difference). After the initialization, new nodes are introduced to the graph one by one, and each time a new node is connected to $m \leq n_0$ existing nodes using preferential attachment. In this thesis the BA graph generation models always connect a new node to $m = n_0$ existing nodes, which is most common. The probability p_i that a new node is connected to node i is

$$p_i = \frac{k_i}{\sum_j k_j},$$

where k_i is the degree of node i and $\sum_j k_j$ is the sum of all degrees of pre-existing

[2]Typically in literature only graphs generated with models similar to the Erdős–Rényi model are considered random graphs, and graphs generated with models like the Barabási–Albert model are referred to differently, as the method of generation is not completely random. In the case of the BA model, the generated graphs are called scale-free graphs, which will be talked about in Sec. 1.7.

nodes.[3] New nodes are introduced to the graph until there are a total of n nodes. It is quite clear from this that when the generation of a graph progresses, a new node will more likely be attached to nodes with higher rather than lower degrees, and therefore high-degree nodes will continue to be connected to new nodes more likely, creating *hubs*. The concept of hubs is expanded upon in the next section.

There are many different types of graph generating models that produce significantly different graph types. I will not introduce more of them here since the ER and BA models are sufficient for the purposes of this thesis. ER and BA graphs[4] are also significantly different in their respective overall structures, and some of these differences will be discussed later.

1.4 Clusters and hubs

Clusters and hubs are good indicators for the structure of a network. Clusters deal with smaller-scale structures while hubs are bigger and also have a more significant effect on the overall structure and heterogeneity of a graph.

A cluster is defined as a set of three nodes in a network that are all connected to one another. An important value in the study of clustering is the *clustering coefficient*, which can be defined as

$$C = \frac{(\text{number of triangles}) \times 3}{(\text{number of connected triples})}.$$

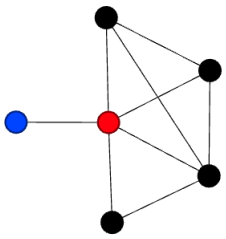
Here a triangle means a cluster, so three nodes u, v and w that are all connected with each other, and a connected triple means three nodes uvw with edges (u, v) and (v, w) [1]. The number of triangles is multiplied by a 3 because a triangle has three connected triples, so all of them need to be taken into account. The clustering coefficient gives the fraction of connected triples that are also clusters. The clustering coefficient can also be given for a single node as the *local clustering coefficient* as

$$C_i = \frac{(\text{number of pairs of pairs of neighbors of } i \text{ that are connected})}{(\text{number of pairs of neighbors of } i)}.$$

[3] If the initial n_0 nodes are disconnected, then the first new node is connected to all the pre-existing nodes.

[4] Saying 'Erdős–Rényi graph' or 'Barabási–Albert graph' is a bit misleading, since the generated graphs themselves are not specifically any one type of a graph as much as they are just outcomes of specific graph generation models. I will however use this shortcut when referring to a graph generated by a specific model.

This coefficient gets values in the interval $[0, 1]$, and it describes how close the neighbors of node i are to all being connected to one another, with value one meaning that all pairs of neighbors are connected and value zero meaning that none of the neighbors are connected. If the node has degree zero or one, the local clustering coefficient is defined to be zero. For example, in the case of the graph



the red node r has a clustering coefficient $C_r = 4/10$ and the blue node b has $C_b = 0$.

Hubs do not have a specific definition as to what is and is not a hub. Hubs are nodes in a graph to which a large portion of the other nodes are connected to. Real-world networks tend to have hubs, which is intuitively understandable, since real-world networks, like social networks, traffic networks, electric networks and the Internet, all tend towards having centralized nodes from which a lot of other nodes can be accessed. Hubs also affect the dynamics on a network quite significantly, as will be seen in the case of continuous-time quantum walks. Figure 3 shows a graph with three hubs, to which all the nodes of the graph are connected to. Here the graph contains 20 nodes and the smallest hub has 9 nodes connected to it, so the hubs have been chosen so that they are connected to at least about half the total amount of nodes.

1.5 Degree centrality and eigenvector centrality

Different centrality measures are used in network theory to quantify the importance of different nodes in a network [1]. Because different measures of importance can be used depending on what is studied, different centrality measures also exist. There are many different centrality measures, but here I only talk about two: *degree centrality* and *eigenvector centrality*.

Degree centrality is a really simple centrality measure because it is just the degree of a node. Sometimes the degree centrality is also defined as the degree of the node divided by the number of nodes in the network, so that it is a value in the interval $[0, 1]$. The degree centrality is quite a crude measure of centrality, but it is still a very useful measure for it's simplicity, as it gives quite a good indication on how central a node is in a network.

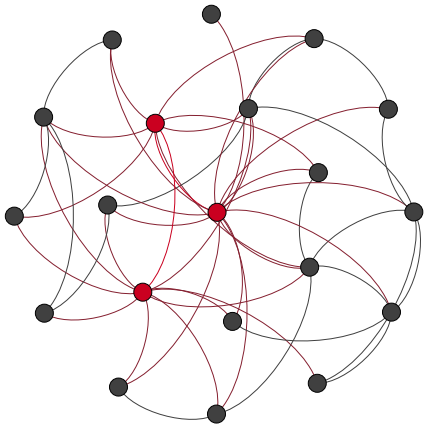


Figure 3: An example of a graph with hubs. The hubs and the edges connected to them are marked in red and here the hubs all have at least 9 nodes connected to them with the graph containing 20 nodes in all.

An extension of the degree centrality is eigenvector centrality, in which the centrality score of a node depends on the degrees of that node's neighbors. This means that instead of just looking at how many connections the node has, the connections of the neighboring nodes are also taken into account. The centrality $\mathbf{x}$ can be given with the eigenvector equation

$$\mathbf{Ax} = \lambda_1 \mathbf{x},$$

where λ_1 is the largest eigenvalue of the adjacency matrix $\mathbf{A}$. The largest eigenvalue is chosen since we require that the centrality score is always non-negative, so $\mathbf{x}$ only contains non-negative entries, and the *Perron-Frobenius theorem* states that the largest real eigenvalue of a real square matrix with non-negative components has a corresponding singular eigenvector with also non-negative entries, and that this is the only eigenvector with all non-negative entries [1]. The eigenvector equation can be rewritten as the eigenvector centrality of node i as

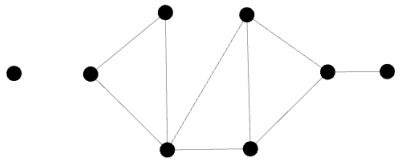
$$x_i = \frac{1}{\lambda_1} \sum_j A_{ij} x_j,$$

so the centrality of i is proportional to the centralities of i's neighbors. This means

that the eigenvector centrality of a node can be large either because it is connected to a lot of nodes or it can be connected to a few important nodes (or both).

1.6 Degree distributions

The distribution of node degrees is one of the most fundamental properties of networks [1]. For example, consider the graph:



There are $n = 9$ nodes, so there is a $1/9$ probability of picking any one node randomly from this graph. Furthermore, there is one node with degree 0, one node with degree 1 and so on, so the probabilities p_k of picking nodes with degrees $k = 0, ..., 4$ are

$$p_0 = \frac{1}{9},\ p_1 = \frac{1}{9},\ p_2 = \frac{2}{9},\ p_3 = \frac{3}{9},\ p_4 = \frac{1}{9}.$$

The probability of picking a node with a degree higher than 4 is zero. This is the *degree distribution* of this graph. The degree distribution can also be given with the total amounts of nodes of different degrees, and this can be given by multiplying the probabilities with the size of the graph, np_k. Figure 4 shows the degree distribution of a network of airports, and it can be seen that there are a great deal more nodes with a low degree than with a high degree.

The degree distributions of different types of networks can differ greatly depending on their structures, and it is a great tool when analysing dynamics on networks. The next section shows how the degree distribution can give information on the structure of the network in the form of *scale-free networks*.

1.7 Power laws and scale-free networks

A lot of real-world networks' degree distributions, like the airport network used in figure 4, social networks and the Internet, follow a *power law*. This means that the degree distribution roughly follows a straight line when plotted on a logarithmic scale, with the network having a lot of small-degree nodes and a few high-degree ones. Mathematically the logarithm of the degree distribution p_k can be given as a function of the degree k as

$$\ln p_k = -\alpha \ln k + c, \tag{3}$$

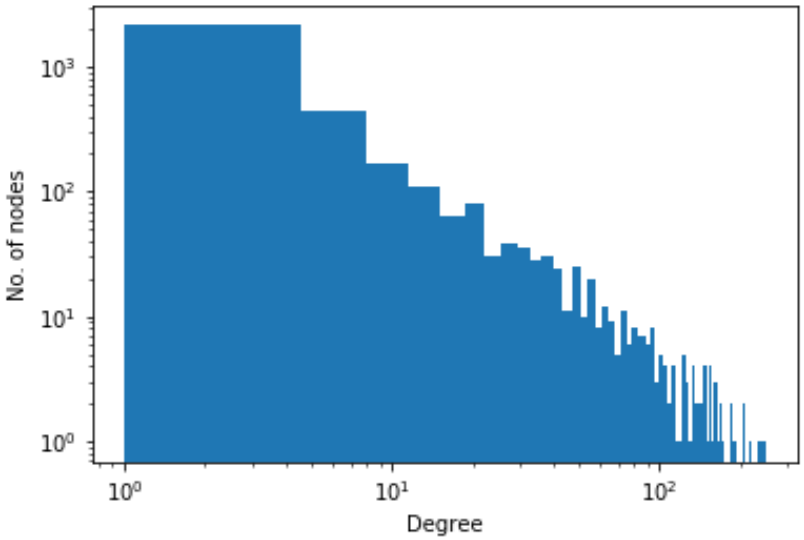


Figure 4: The degree distribution in a logarithmic scale of a network of airports as used in [6].

where α and c are constants [1]. Taking the exponential of this equation gives

$$p_k = Ck^{-\alpha},$$

where $C = e^c$. Degree distributions of this form are power-law distributions. Networks that follow the power law are sometimes called scale-free networks. The value of α is always positive here, equal to minus the slope of the degree distribution histogram on a logarithmic scale. It also typically gets values $2 \leq \alpha \leq 3$, although values beyond these are also possible. Networks that have hubs typically follow the power law, an example of which is the Barabási-Albert graph. This can be understood intuitively because a graph that has few hubs in it will have these nodes be in the tail end of the logarithmic plot, and then there will be many nodes with lower degrees that are connected to these hubs and/or other lower degree nodes. As the Barabási-Albert graph generating model follows preferential attachment, some nodes will become hubs as new nodes are more likely connected to them as their degree rises and most nodes will more likely remain as low-degree nodes.

Networks with power-law distributions also do not always follow the power law exactly at the ends of low k and high k. At the low end there can be significantly more nodes with small degrees, that only connect to a few or a single other node, that differ from the power-law distribution. At the high end there can be hubs that stretch far beyond what the power-law distribution would predict. A scale-free

network therefore is then only to be expected to follow a power-law distribution with respect to the number of nodes asymptotically [1].

1.8 Regular and complex networks

There are not any specific rules about what constitutes a *complex network*. The term complex network is typically there just to give a general direction as to what kind of a network one is dealing with. A *regular network*, on the other hand, has a clear definition: it is any network where all nodes have the same degree. In this thesis I will define a *semi-regular network* as a network where almost all of the node degrees are the same or the network is a completely random ER graph (where also there is not much variance in the degree distribution). A complex network will then differ from both regular and semi-regular networks, where there is more variance in the network structure.

The simplest example of a regular network is a *complete graph*. This is a graph where every node is connected to every other node. Another is a *hypercube graph*, which is a d-dimensional hypercube, where vertices are nodes and the lines between vertices are edges. A *circle graph* is a regular network in which every node is connected to two neighbors, and the nodes and links form a connected circle. An example of a semi-regular network is a *star graph*, where there is one central node that is connected to every other node and all other nodes are connected only to the central node. Examples of all these graphs can be seen in figure 5.

Most real-world networks are complex networks, as they have very complex structures. Generated graphs can also be complex with the right sets of rules, an example of which is the BA graph. In this thesis the behaviour of continuous-time quantum walks is first observed on regular and semi-regular networks, but the main focus is on complex networks.

1.9 Renormalization and pruning

Manipulating a network by making the size of the network smaller can be interesting for studying how much the structure of the network is preserved, and how the dynamics of a quantum walk on the network change. One such way of doing this is by geometrical embedding of the network in order to construct a renormalized version of it's structure by coarse-graining neighboring nodes into supernodes, and therefore rescaling the network. This is called the *geometric renormalization group for complex networks* and it was introduced by Guillermo García-Peréz, Marián

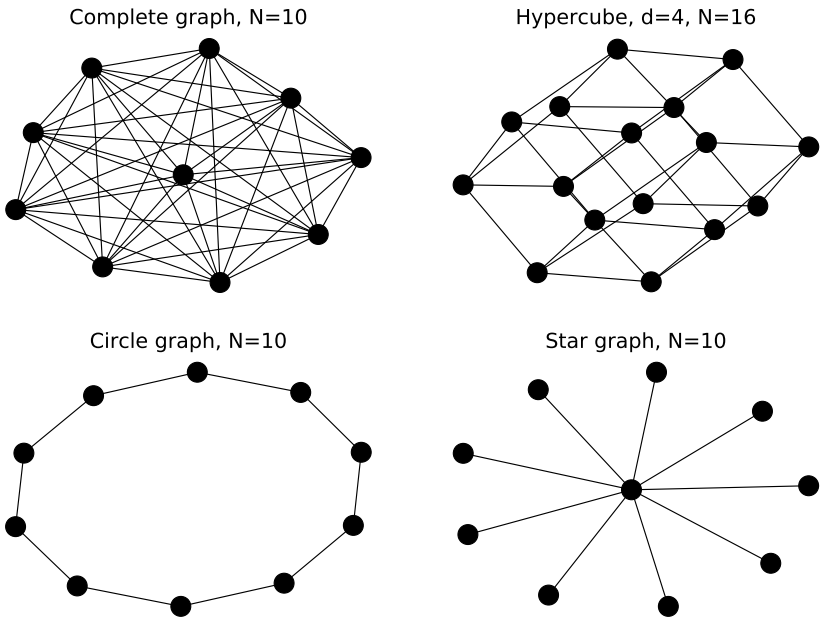


Figure 5: Examples of different simple graph types.

Boguñá and M. Ángeles Serrano in 2017 [6]. It was specifically made for real-world complex networks since they can have complex, not so well-known geometric and topological properties that are important to preserve.

The renormalization can be formulated by first considering a map of the network embedded in a hidden metric space, $\mathcal{M}(T, G)$, where T is the topology and G the geometry of the network. Hidden metric spaces are variations of hidden variables, in which the network nodes reside in and there are hidden distances between all pairs of nodes that satisfy the triangle inequality, and these hidden distances can be arbitrarily large [7]. A geometric renormalization operator $\mathbb{F}_r$ can then be defined, with resolution r, that coarse-grains the network with a factor of r. This gives a new topology T' and geometry G' that conform to the renormalized map $\mathcal{M}'(T', G')$. This can be written as

$$\mathcal{M}(T, G) \xrightarrow{\mathbb{F}_r} \mathcal{M}'(T', G').$$

This operation can be iterated multiple times, starting from the original network at layer $l = 0$,

$$\mathcal{M}^{(l+1)}(T^{(l+1)}, G^{(l+1)}) = \mathbb{F}_r[\mathcal{M}^{(l)}(T^{(l)}, G^{(l)})].$$

This transformation can be thought of as zooming out from the original network by changing the minimum length scale to a larger value.

The simplest hidden metric space to which a network can be embedded is a circle, where each node has a specific angular position, $\{\theta_i \ : \ i = 1, ..., N\}$. In the $\mathbb{S}^1$ *model*, nodes are placed onto the circle by connecting every pair of nodes with a probability that increases inversely to the angular distance and describes the similarity between pairs of nodes. The nodes are then merged together depending on their similarities into supernodes, and the ordering of the nodes/supernodes are preserved on the circle. If $r = 2$, then typically two nodes in layer l are merged into one supernode in layer $l + 1$, and if $r = 4$, then four nodes are merged into one supernode, etc. This means that a network on layer l is r^l times smaller than the original network on layer $l = 0$. Edges are kept in layer $l + 1$ depending on the connections between similar nodes in layer l. Figure 6 shows the renormalization technique for three layers.

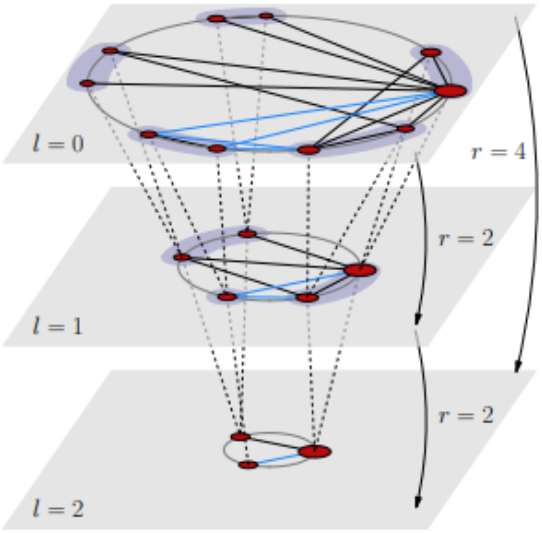


Figure 6: The geometric renormalization technique done for a small network. The original network is on layer $l = 0$ and different layers are obtained after a renormalization step with resolution r. Each node of the network is placed on the $\mathbb{S}^1$ circle at an angular position $\theta_i^{(l)}$. Nodes in layer l are deemed to be similar if they are marked by the grey area, and similar nodes are fused together in layer $l + 1$ into a supernode. If the resolution is $r = 2$, then pairs of nodes are fused together. For a network with an odd number of nodes, one node is not fused with another node. Two supernodes in layer $l + 1$ are linked if and only if at least one node, in layer $l + 1$, in a supernode is connected to at least one node in another supernode. The blue lines give an example of this. Source: [6].

This is the method described for the renormalization of a network. The network can be pruned as well, which means that after a rescaling of the network, every existing edge is individually kept with probability

$$q_{ij}^{(l)} = \frac{p_{ij,\text{new}}^{(l)}}{p_{ij}^{(l)}}, \quad p_{ij,\text{new}}^{(l)} \leq p_{ij}^{(l)}, \tag{4}$$

where $p_{ij}^{(l)}$ is the original probability that an edge exists between nodes i and j, and $p_{ij,\text{new}}^{(l)}$ is the new probability. The probability p_{ij} is given as

$$p_{ij} = \frac{1}{1 + \left(\frac{d_{a,ij}}{\mu\kappa_i\kappa_j}\right)^{\beta}},$$

where μ controls the average degree of the network, β controls the clustering, $d_{a,ij} = R\Delta\theta_{ij}$ is the distance of nodes i and j separated by an angle $\Delta\theta_{ij}$ with R set to $N/2\pi$, and κ_i and κ_j are the hidden degrees proportional to the degrees of i and j. From this it can be seen that the more similar the two nodes are, so the smaller $\Delta\theta_{ij}$, then p_{ij} is higher, and the higher the degrees of nodes i and j, then p_{ij} is also higher. Here it is assumed that the average degree of the renormalized network in layer l, $\langle k^{(l)}\rangle$, is larger than the average degree of the original network, $\langle k^{(0)}\rangle$.[5] The renormalized network is congruent with the hidden metric space, meaning that they have a similar shape and size, with a parameter $\mu^{(l)} = \mu^{(0)}/r^l$, which controls the average degree. The value r is the resolution. The pruning is done so that the average degree is roughly the same in the renormalized network as in the original network, $\langle k_{\text{new}}^{(l)}\rangle = \langle k^{(0)}\rangle$, which also means that a new value $\mu_{\text{new}}^{(l)}$ needs to be found since the pruned network has a lower average degree. For real-world networks, the value of $\mu_{\text{new}}^{(l)}$ is calculated with

$$\mu_{\text{new}}^{(l)} = c\frac{\langle k^{(0)}\rangle}{\langle k^{(l)}\rangle}\mu^{(l)},$$

where c is a correcting factor that is initially set to one. The network is now pruned for every value of c so that if $\langle k_{\text{new}}^{(l)}\rangle > \langle k^{(0)}\rangle$, c gets a new value $c - 0.1u \to c$, where u is a random variable picked from a uniform distribution in the interval $(0, 1)$, and if $\langle k_{\text{new}}^{(l)}\rangle < \langle k^{(0)}\rangle$, then $c + 0.1u \to c$. This process of readjusting the correcting factor continues until $|\langle k_{\text{new}}^{(l)}\rangle - \langle k^{(0)}\rangle|$ is below a required threshold value. For the case of the network used in this thesis, the threshold value is set to 0.1. If a link is removed that makes a node disconnected, then that node can be removed from the pruned network, since it does not contribute to the structure anymore.

[5]Note, that this is typically true for large real-world networks, and not necessarily true for small networks like in figure 6, where the average degree of the original network is larger than the average degree of the renormalized networks.

2 Quantum walks

Quantum walks are the quantum mechanical equivalent to classical random walks, and they have been an area of study since 1993 [8]. Quantum walks are divided into two main categories: discrete-time and continuous-time quantum walks, labelled respectively as DTQW and CTQW. This thesis focuses on the latter type, which was first introduced in 1998 [9].

2.1 Random walks

Before talking about quantum walks, it is good to start from the classical equivalent that are random walks. Random walks are stochastic processes in mathematics that describe a walker which has a certain probability to move in a certain direction in a mathematically defined space. This space can be for example the integer space $\mathbb{Z}$, a d-dimensional real space $\mathbb{R}^d$ or a graph G. Typically random walks are defined as Markov processes or Markov chains, in which the walker's probabilities to move to the next possible states depend only on the walker's current state, meaning that the walker has no memory of what has happened before [13].

A famous example of a random walk is the drunkard's walk [4]. Consider a drunkard who is walking on the street. During each step the drunkard has a probability p to step forward and a probability $1 - p$ to step backward. Where will the drunkard most likely end up after n steps? Also, what is the probability distribution after n steps?

In figure 7 the probability distribution of a drunkard's walk is visible after $n = 100$ where the drunkard has started from $x = 0$ and has had an equal probability to step in both directions. As is visible, the distribution is Gaussian and the highest probability is around the starting point. The probability also decreases rapidly, and although the number of steps taken is 100, it is highly improbable to end up more than 40 steps from the starting point.

The dynamics of a continuous-time random walker on a graph can be described in the same way as explained in Sec. 1.2. The walk can be described with the first-order, linear DE

$$\frac{dp_j(t)}{dt} = -\gamma \sum_k L_{jk} p_k(t), \tag{5}$$

where $p_j(t)$ is the probability of the walker to be at vertex j at time t, γ is the fixed probability per unit time of jumping to an adjacent vertex and L is the Laplacian. The γ is the coupling constant and it will prove to be an important quantity, and

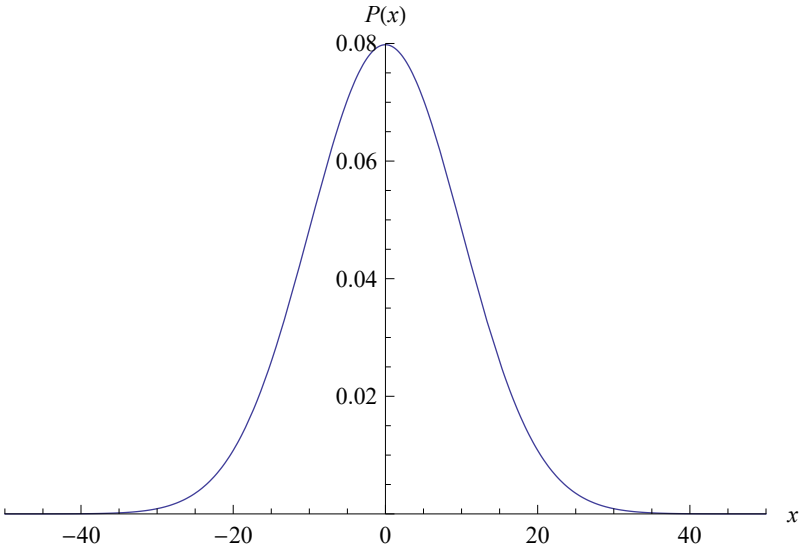


Figure 7: Probability distribution of the drunkard's walk on the set of integers after $n = 100$ steps and $p = 0.5$ in both directions. The starting point is at $x = 0$. Note that with an even number of steps the probability of the drunkard to be at an uneven integer is zero. These probabilities are not explicitly shown in the plot. The probability distribution is also shown to be continuous for simplicity, when in actuality it is discrete.

finding the optimal value for it will be a matter of interest. In equation (3) the probabilities are conserved, since the columns of L sum to zero.

2.2 Continuous-time quantum walks

A continuous-time quantum walk on a graph G is defined in an N-dimensional Hilbert space spanned by the basis states $|j\rangle$, where j is a node in G. A general state $|\psi(t)\rangle$ can now be described with the N complex amplitudes $q_j(t) = \langle j|\psi(t)\rangle$. So instead of using the probability in the dynamical equation, the probability amplitude is used. Therefore, instead of just evolving the probability and directly taking the value $p_j(t)$ to get the probability of the walker being in node j at time t, the probability amplitude's absolute value squared gives the probability, $|\langle j|\psi(t)\rangle|^2$. The dynamics of this walk can now be described with the Schrödinger equation as

$$i\frac{dq_j(t)}{dt} = \sum_k H_{jk}q_k(t), \tag{6}$$

where H is the Hamiltonian and it is defined as $H = \gamma L$. Now the only two differences between equations (3) and (4) is a factor of i and the probability replaced with probability amplitude, which result in radically different behaviour. In the end the reason why quantum walks behave in an extremely different manner compared to classical random walks is caused by the superposition of states and quantum interference.

Figure 8 shows the probability distribution of the continuous-time quantum walker after letting the walk run for $t = 35$. Here the Hamiltonian H_L is just defined with the adjacency matrix of a path graph of size 200, which represents a line, and the state of the walker is then $|\psi(t)\rangle = U_L(t)|0\rangle = e^{-iH_Lt}|0\rangle$, where $|0\rangle$ is the initial state corresponding to the middle node in the graph. For this walk, γ is not needed since that would just change the rate at which the walker evolves. The state of the walker is only measured at the end in order to preserve the superposition of states throughout the course of the walk. The walk can be seen to spread outwards from the initial state more than the classical equivalent, which is caused by quantum interference. Of course this cannot be directly compared with the classical drunkard's walk, since this walk happens in continuous time, but the discrete case functions similarly [4].

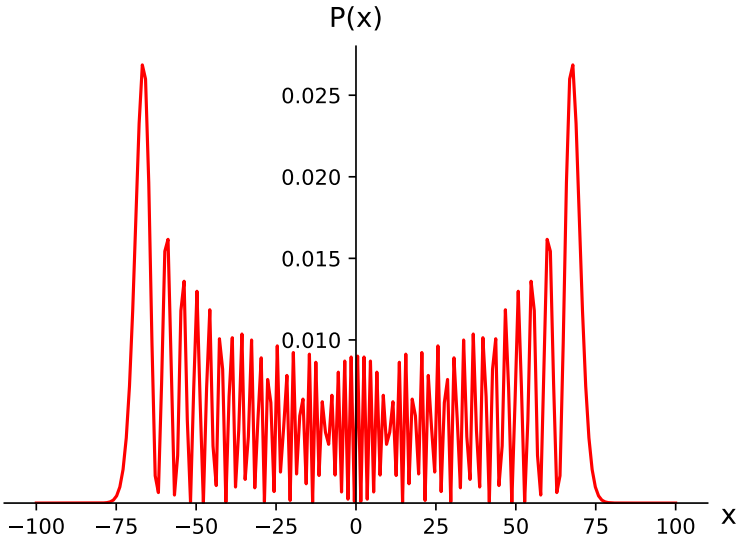


Figure 8: Probability distribution of the CTQW on the set of integers measured after letting the walk run for $t = 35$. The starting point is at $x = 0$.

2.3 Spatial search by CTQW

The main subject of this thesis is using CTQW for spatial search on networks, and the main point of study is finding a specific node in the network. Because of the nature of quantum mechanics and random walks in general, the correct node can only be found with a certain probability. Thus finding the optimal success probabilities for nodes on a network and what rules govern the distribution of these probabilities are of interest.

A paper written by Andrew Childs and Jeffrey Goldstone is the first published paper that describes spatial search by CTQW [2]. This paper also refers to the seminal paper written by Grover, where he presents the quantum search algorithm, which is now known as Grover's algorithm [3]. This algorithm finds with high probability a unique input to a black box function with a particular output in $\mathcal{O}(\sqrt{N})$ evaluations, when N is the size of the function's domain. In other words, given a function $f(x) : \{1, \ldots, N\} \to \{0, 1\}$ that satisfies

$$f(x) = \begin{cases} 0, & x \neq w \\ 1, & x = w, \end{cases}$$

Grover's algorithm finds the value w with high probability using $\mathcal{O}(\sqrt{N})$ queries. Comparing this to a classical search algorithm, it can be seen that the quantum

search algorithm provides a quadratic speed-up. This is because a classical search algorithm takes $\mathcal{O}(N)$ queries because on average it has to go through half of the inputs before finding the correct one.

Grover's algorithm cannot be used to search a physical database, but spatial search by CTQW can. This is because Grover's algorithm is an abstraction of the problem, while quantum walks give the possibility of searching a real-world network. Spatial search by CTQW requires the use of an oracle, which are quite common in classical computing [4]. An oracle is a black box, that performs a specialized task and then the number of calls made to that oracle defines the efficiency of an algorithm. In quantum mechanics this oracle can be introduced in the form of a Hamiltonian. The oracle Hamiltonian can be given as

$$H_w = -|w\rangle\langle w|,$$

following the presentation in [10]. Now, in actuality the oracle Hamiltonian should be given as $H_w = -\omega|w\rangle\langle w|$, where ω would have units of inverse time, but here $\omega = 1$. The target node is now w, and the ground state of H_w is $|w\rangle$ with energy -1. All other states are of energy zero.

Note, that spatial search by quantum walk could also be done with multiple target vertices. For that case the oracle Hamiltonian can be defined more generally as

$$H_w = -\sum_{w\in\text{marked}} |w\rangle\langle w|,$$

and this is discussed more extensively in [16]. In this thesis I focus on a single marked vertex at a time.

Now the full Hamiltonian of a continuous-time quantum walker that searches for a specific node on a graph can be written as

$$H = \gamma L + H_w = \gamma L - |w\rangle\langle w|. \tag{7}$$

Because earlier ω was set to be 1 with dimensionless units, therefore γ is also dimensionless. Now γ is a coupling constant that shifts the Hamiltonian's ground state and the first excited state between the initial superposition $|s\rangle$ and the target state $|w\rangle$.

The initial state of the walker is set to be an equal superposition over all states in the graph, i.e.

$$|s\rangle = \frac{1}{\sqrt{N}}\sum_j |j\rangle. \tag{8}$$

The walk is then made to run for a time T. One of the main goals is to define the optimal γ so that the success probability, that is gotten from the complex amplitude, $|q_w(T)|^2 = |\langle w|\psi(T)\rangle|^2$, is as close to one with the lowest T possible.

The reason why $|s\rangle$ is chosen as the initial state of the walker is because that is the ground state of the Laplacian. Earlier in lemma 1 it was shown that the eigenvalues of the Laplacian can be ordered by increasing value with the first eigenvalue having the value of zero. The corresponding eigenvector is the ground state of the Laplacian, since $L|s\rangle = 0$.

A good question to ask now is why would there be an optimal value for γ, T and the success probability? Confirming the possibility of finding optimal values for these parameters can be done by studying the spectrum of H [2]. First of all, $L|s\rangle$ always equals zero, regardless of the graph. As $\gamma \to \infty$, the ground state of H gets close to $|s\rangle$ because the contribution of H_w to H becomes insignificant. On the other hand, when $\gamma \to 0$, the ground state of H is close to $|w\rangle$. The states $|s\rangle$ and $|w\rangle$ are nearly orthogonal to each other, so the first excited state of H will be close to $|s\rangle$ as $\gamma \to 0$ for large N.[6] With these findings it is reasonable to expect that the ground state of H will switch from $|w\rangle$ to $|s\rangle$ at some intermediate range of γ, and the ground state could also overlap with both states. If this is also true for the first excited state, then the Hamiltonian will effect the transitions for the state of the walker from $|s\rangle$ to a state with substantial overlap with $|w\rangle$ for certain values of γ.

Now let's assume that there is a Hamiltonian H, and two orthogonal states $|a\rangle$ and $|b\rangle$, such that it's action on these states is given by

$$H|a\rangle = x|b\rangle, \quad H|b\rangle = x|a\rangle, \tag{9}$$

where $x \in \mathbb{R}$. Now $|\psi_+\rangle = \frac{1}{\sqrt{2}}(|a\rangle + |b\rangle)$ and $|\psi_-\rangle = \frac{1}{\sqrt{2}}(|a\rangle - |b\rangle)$ are the eigenstates of H, with eigenvalues $\lambda_+ = x$ and $\lambda_- = -x$. The evolution of state $|a\rangle$ with respect to the Hamiltonian can be given as

$$\begin{aligned} e^{-iHt}|a\rangle &= \sum_{n=0}^{\infty} \frac{(-i)^n t^n}{n!} H^n |a\rangle = \sum_{n=0}^{\infty} \frac{(-i)^{2n} t^{2n}}{(2n)!} x^{2n}|a\rangle + \sum_{m=0}^{\infty} \frac{(-i)^{2m+1} t^{2m+1}}{(2m+1)!} x^{2m+1}|b\rangle \\ &= \cos(tx)|a\rangle - i\sin(tx)|b\rangle. \end{aligned}$$

The sum could be split into two parts with Eq. (9). In this case it can be seen that the Hamiltonian drives periodical transitions between the states with respect to time, and the system transitions from state $|a\rangle$ to state $|b\rangle$ in time $t = \pi/(2x) =$

[6]For sufficiently large N, the perturbation caused by H_w to the original Hamiltonian is less significant and the states $|s\rangle$ and $|w\rangle$ will be more closely orthogonal.

$\pi/(2(\lambda_+ - \lambda_-))$. This example will become significant when looking at the case of the complete graph.

Childs and Goldstone showed in the original paper that for a d-dimensional cubic lattice, where d is independent of the size of the graph N, this algorithm for the spatial search by quantum walk gives a success probability of order 1 in time of order $\mathcal{O}(\sqrt{N})$ with $d > 4$, and a success probability of order $1/\log N$ in time of order $\mathcal{O}(\sqrt{N \log N})$ [2]. For $d < 4$, there is no substantial speed-up as compared to a classical algorithm, which is of order $\mathcal{O}(N)$. This is because it can be shown that for the success probability

$$|\langle w|\psi(t)\rangle|^2 \leq 2\sqrt{N}|E_0|,$$

where E_0 is the ground state energy, and for $d < 4$ E_0 always stays small. Childs and Goldstone also showed that this optimality holds true for the complete graph and hypercube as well, although the complete graph is a special case because the quantum walk is optimal and the time is of order $\mathcal{O}(\sqrt{N})$ for all dimensions.

The Hamiltonian can also be defined with the adjacency matrix, and the differences between using either the Laplacian or the adjacency matrix on bipartite graphs is discussed in *Laplacian versus Adjacency Matrix in Quantum Walk Search* by Thomas Wong, Luís Tarrataca and Nikolay Nahimov [15]. The Hamiltonian is defined with the adjacency matrix as

$$H_A = -\gamma A - |w\rangle\langle w|.$$

The only difference between this, and the Hamiltonian being defined with the Laplacian, is that the diagonal matrix D is dropped, so the diagonal of the Hamiltonian does not contain the degrees of each node anymore. Now the initial state cannot be chosen as in Eq. (8), since it is not an eiqenvector of A, and $-\gamma A$ would cause $|s\rangle$ to evolve, even though no oracle is present. A different initial state could be chosen, which would make H_A a viable Hamiltonian for quantum spatial search, but using the Laplacian is more beneficial for the simplicity of the initial state.

Spatial search by quantum walk continues to be an active field of research [16–20, 34]. The research today is focused either on acquiring analytical results in simple graphs with the original search algorithm by Childs and Goldstone expanded upon in different situations, for example with multiple target nodes or if the walker is influenced by noise, or on quantitative results by looking at the behaviour on complex networks. This thesis is focused on the latter.

3 Numerical methods

There are several numerical methods used in the computations for this thesis. Some of them are quite well known and widely used, while others are state-of-the-art algorithms. The purpose of this section is not to explain in detail how these algorithms work, but more to explain generally how and why they were used for the understanding and reproducibility of results.

The success probability for the spatial search by quantum walk can be calculated analytically only for specific classes of graphs, which are regular or semi-regular in structure. In general, numerical methods need to be used, since larger unregular networks become too complex for the spatial search problem to be solved analytically on them. Some analytical results will be shown for regular and semi-regular graphs in the second chapter, but all the results will also be calculated numerically in order to check the validity of the numerical computations.

The programming languages Python version 3.7 and Julia version 1.5 were used for computing the results of this thesis. Originally only Python was used, but the bigger graphs needed faster computation times, so Julia was used for those. As Julia is a compiled language and repeated calculations are performed for all the nodes of large graphs, there is a significant speed-up in using Julia. Julia's syntax is also quite similar to Python's, so the conversion was made easier. All the calculations done to acquire results for all the nodes of non-regular graphs were also done in remote clusters. The clusters belong to the Titan supercomputer in University of Turku. The clusters were accessed remotely through authorized access from a computer's shell. Parallel computing was also used for Julia. Workers are established with the code for the CPU nodes in a cluster, and then a wrapper function distributes the jobs of calculating the results for different graph nodes to different workers. This means that a single worker is responsible of calculating results for a single node at any one time, and once it is finished with that node, the worker receives another node. If a worker crashes, a new worker is used.

In general, the codes for Python and Julia have a slightly different structure. In Python, the spatial search process was written as a class, where the graph and target node are initialized and the Hamiltonian and success probability functions are defined:

```
class SpatialSearch:

  def __init__(self, graph, target):
```

```
        self.graph = graph
        self.target_n = target
        self.n = len(graph)
        self.w = basis_vector(self.n, self.target_n)
        self.s = self.n**-0.5 * np.ones(self.n)

    def hamiltonian(self, gamma):
        H = gamma * nx.laplacian_matrix(self.graph)
        H[self.target_n, self.target_n] -= 1.
        return H

    def success_probability(self, gamma, t, stop=None, steps=None,
                            dense=False):
        H = self.hamiltonian(gamma)
        if stop is not None:
            psi = expm_multiply(-1j*H, self.s, start=t, stop=stop,
                                num=steps, endpoint=True)
        else:
            if dense:
                U = expm(-1j*t*np.asarray(H.todense()))
                psi = U.dot(self.s)
            else:
                psi = expm_multiply(-1j * t * H, self.s)
        return np.abs(psi.dot(self.w))**2
```

In Julia the success probability function is defined on it's own with the Hamiltonian:

```
L = laplacian_matrix(graph)
N = size(L)[1]
psi_0 = 1 / sqrt(N) * ones(ComplexF64, N)

function success_probability(L, node, gamma, t)
    try
        H = gamma * L
        H[node, node] -= 1
        psi_t = expmv(-1im * t, H, psi_0)
        return abs2(psi_t[node])
    catch e
        @error "Problem with node = $node, gamma = $gamma and t = $t"
```

```
      throw(e)
    end
end
```

The reason why there are if-clauses in the Python version of the code is simply because it was used specifically for dense graphs as well (complete graph) and single values were calculated when stop=None. Julia has an error catch so that the computation does not terminate at faulty nodes (a node of degree zero, for example).

The initial choices for γ_{opt} and T are set as $1/k_w$ and 0, where k_w is the degree of the target node. For γ_{opt}, the value is always $\gamma_{\text{opt}} = c/k_w$, where $c \approx 1$. This is because with this value, the Hamiltonian is evenly weighted between the initial and target states, so the walker will rotate most efficiently between the two states. Technically the initial value of T could be chosen to be $\sqrt{N}$ since for most graphs the optimal time is of $\mathcal{O}(\sqrt{N})$ for most graphs, but for some nodes especially with low optimal success probabilities this can cause the algorithm to search for optimal values longer, and nodes with good optimal success probabilities are found rather quickly no matter the initial value as they have smooth curves, so this initial value of T does not make much difference.

3.1 Sparse diagonalization algorithm (implicitly restarted Lanczos method)

The *Lanczos algorithm* is an algorithm designed by Cornelius Lanczos that finds the k most "useful" (highest or lowest) eigenvalues and their corresponding eigenvectors of an $n \times n$ Hermitian matrix [25]. The specific version is the *implicitly restarted Lanczos method*, and this is realized with the ARPACK software, which is a collection of Fortran 77 subroutines designed to solve large-scale eigenvalue problems [26, 27]. This algorithm is used in acquiring the first two algebraically lowest eigenvalues and their corresponding eigenvectors of the Hamiltonian. These eigenvalues correspond to the ground state and first excited state of the Hamiltonian. The Lanczos algorithm works especially quickly if the Hermitian matrix is sparse, which is to be expected from the Hamiltonian of most networks, and especially from the Hamiltonian of complex networks.

The original Lanczos algorithm has a problem with numerical instability since the computed *Krylov subspace* basis loses orthogonality, so that is why a modified version of the algorithm is used. An r-order Krylov subspace generated by an $n \times n$ -matrix M and vector v of dimension n is a linear subspace spanned by the images

of v under the first r powers of M, i.e. $\mathcal{K}_r(M, v) = \text{span}\{b, Ab, ..., A^{r-1}b\}$. Krylov subspaces are used to avoid matrix-matrix operations, and to rather start with a vector v, and then multiply it with M to get the vector Mv, and then multiply again with M to get the vector M^2v, and so on. Methods like the Lanczos algorithm, that use Krylov subspace projection, are collectively called Krylov subspace projection methods.

3.2 Nelder-Mead algorithm

The *Nelder-Mead algorithm* is a numerical method that is used to find the minimum or maximum of a function in multidimensional space [28]. This algorithm is used in the maximization of the success probability $|\langle w|\psi(t)\rangle|^2$, which is given the parameters t and γ, since $|\psi(t)\rangle = e^{-iHt}|s\rangle$. In other words, the function $p(\gamma, t)$ is optimized with respect to the parameters and the output of the function.

The original Nelder-Mead algorithm was written specifically to find the minimum of a function, and a minimizing algorithm is used here as well, but finding the maximum of a function with this is easy with just putting a minus sign in front of the function and solving for the inverted function. The Nelder-Mead algorithm works by first choosing initial test points $\mathbf{x}_1, ..., \mathbf{x}_{n+1}$ for a function $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$. These are then ordered according to their values as $f(\mathbf{x_1}) \leq ... \leq f(\mathbf{x}_{n+1})$. The values are then iteratively manipulated and the algorithm converges to a minimum (possibly local) if it exists.

The Nelder-Mead algorithm is used by first defining a function for the optimal success probability of a node and this function is given initial values for γ and t. This function is then optimized. In Python this was defined as

```
def optimal_success_probability(x):
    gamma = x[0]
    t = x[1]
    return - spatial_search_G.success_probability(gamma, t)

minimize(optimal_success_probability_G, [1/deg, t_0],
           method='Nelder-Mead')
```

3.3 Expokit algorithm

For more complex and bigger networks the more basic sparse diagonalization algorithm takes too long in evaluating the matrix exponentials, so for these the *Expokit algorithm* is used. It was first introduced by Roger B. Sidje in 1998 [29]. The Expokit algorithm is designed for computing matrix exponentials. Specifically it can be used for the computation of a small matrix exponential in full, the action of a large sparse matrix exponential on a vector or the solution of a system of linear ordinary differential equations with constant inhomogeneity. The algorithm is used for the second purpose in this thesis, i.e. the multiplication of the unitary evolution operator $U(t) = e^{-iHt}$ with the initial state of the walker $|s\rangle$.

The sparse routines of the Expokit algorithm are based on Krylov subspace projection methods, which is why it works well for sparse matrices of large dimensions. The advantage of using the Expokit algorithm as opposed to the basic sparse diagonalization algorithm is that the Expokit algorithm is more directly optimized to computing numerically the product of $U(t)$ and $|s\rangle$. This will make a noticeable difference especially in the computation times of larger complex networks.

The Expokit algorithm is used only in Julia for this thesis. The algorithm is used in evaluating the value of $|\psi(t)\rangle$ in the success_probability function.

Chapter II

CTQW on Regular and Semi-Regular Networks

In this chapter, spatial search by CTQW is studied on regular and semi-regular graphs. These graphs include the complete graph, hypercube, star graph and ER graph. In the case of all studied networks from now on, the optimal γ, optimal time and optimal success probability will be observed. It is also interesting to see how the energy gaps and overlaps for different graphs behave and for what reasons.

4 Complete graph

The complete graph is a regular graph, so the behaviour of the quantum walk needs to be studied only for one target node, since the walk will be the same for every target node with the initial superposition. Since the complete graph is a very simple structure, solving the values for optimal γ, time and success probability analytically is quite easy as well. Therefore it is beneficial to use the complete graph to check the validity of the numerical results.

Figure 9 shows the plots for the energy gap and the overlaps as functions of γN, as well as the success probability of getting to the target node as a function of time with the optimal γ and with three suboptimal γ for reference. In the first plot the ground state switching from $|w\rangle$ to $|s\rangle$ and first excited state switching from $|s\rangle$ to $|w\rangle$ can easily be seen. The values of γ have been multiplied with the size of the graph to emphasize the fact that for the complete graph the optimal value for γ comes at right about $1/N$, since the degree of all nodes is very close to N, $k = N - 1$. This is the optimal value for γ, because the energy gap between the ground state and the first excited state is lowest, so the Hamiltonian rotates the

state of the walker from the ground state to the first excited state the fastest.

One of the reasons why the energy gap is so sharp for large N in the first plot in figure 9 is because the energies are actually $N-1$ -degenerate for the complete graph. This means that the Hamiltonian has only two distinct eigenvalues, and consequently the walker is able to rotate form the initial state to the target state with the optimal γ without any disturbance from other states.

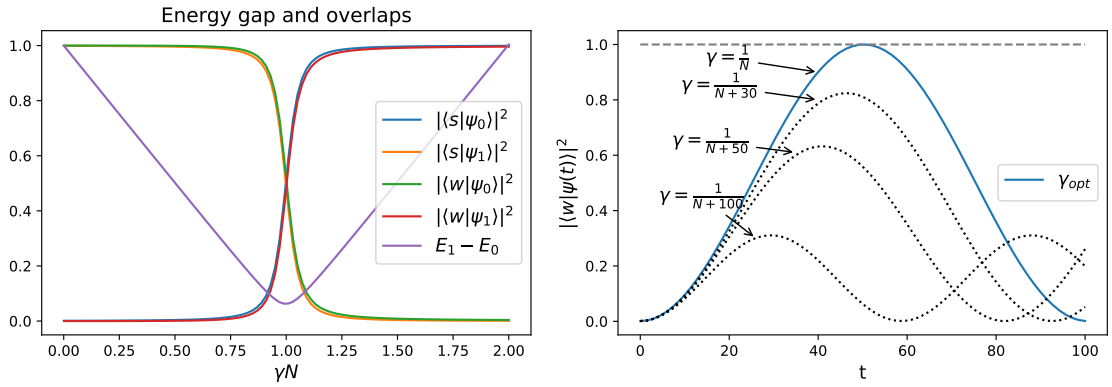


Figure 9: The first plot gives the overlaps of the ground state $|\psi_0\rangle$ and first excited state $|\psi_1\rangle$ with the initial state $|s\rangle$ and the target state $|w\rangle$, as well as the difference between the ground state energy E_0 and first excited state energy E_1, for a complete graph with $N = 1024$ nodes. The second plot gives the success probability as a function of time with the optimal γ for the complete graph with $N = 1024$ nodes, as well as with three different suboptimal γ.

In the second plot in figure 9 the success probability gets close to one at $t \approx 50.27$ and in fact, taking into account the numerical approximations, $|\langle w|\psi\rangle|^2 = 1$ with the optimal γ and T. The suboptimal γ showcase that lowering the value of γ from the optimal γ ends up with a noticeably worse success probability, since the values used for these were $1/(N+30)$, $1/(N+50)$, and $1/(N+100)$. Naturally the same behaviour would be seen if the value of γ would be increased from the optimal γ. The oscillation of the state of the walker to and from the target node can also be seen in the second plot from figure 9, as the success probability of finding that node oscillates.

It is instructive to look at the analytical solution of the spatial search problem for the complete graph. Recalling from Sec. 2.3, for large N $\langle w|s\rangle \approx 0$ and when $\gamma \to \infty$, the ground state of the Hamiltonian ($|\psi_0\rangle$) is close to $|s\rangle$, and when $\gamma \to 0$, $|\psi_0\rangle$ is close to $|w\rangle$. Using degenerate perturbation theory it can be said that the

first excited state of the Hamiltonian ($|\psi_1\rangle$) will be close to $|w\rangle$ when $\gamma \to 0$. The same thing cannot be said about the first excited state when $\gamma \to \infty$, since the Hamiltonian is $N-1$ -degenerate because of L and therefore $|\psi_1\rangle$ can be close to an eigenvector of L. However, the overlap can still be expected to rise compared to when $\gamma \to 0$, when the overlap $|\langle w|\psi_1\rangle|^2$ is close to zero. This is also seen to be true in figure 9. Now it can be assumed that for some intermediate value of γ, the overlaps will be $|\langle s|\psi_0\rangle|^2 \approx \frac{1}{2}$, $|\langle s|\psi_1\rangle|^2 \approx \frac{1}{2}$, $|\langle w|\psi_0\rangle|^2 \approx \frac{1}{2}$ and $|\langle w|\psi_1\rangle|^2 \approx \frac{1}{2}$. Using these together with the orthogonality of $|s\rangle$ and $|w\rangle$, the ground and first excited states can be set with a choice of phase as

$$|\psi_0\rangle \approx \frac{1}{\sqrt{2}}(|w\rangle + |s\rangle), \quad |\psi_1\rangle \approx \frac{1}{\sqrt{2}}(|w\rangle - |s\rangle),$$

or equivalently given in terms of the initial and target states as

$$|w\rangle \approx \frac{1}{\sqrt{2}}(|\psi_0\rangle + |\psi_1\rangle), \quad |s\rangle \approx \frac{1}{\sqrt{2}}(|\psi_0\rangle - |\psi_1\rangle).$$

If the energies are given as $H|\psi_0\rangle = E_0|\psi_0\rangle$ and $H|\psi_1\rangle = E_1|\psi_1\rangle$, then the dynamics with respect to the initial state are

$$\begin{aligned} e^{-iHt}|s\rangle &\approx \frac{e^{-iHt}}{\sqrt{2}}(|\psi_0\rangle - |\psi_1\rangle) = \frac{1}{\sqrt{2}}(e^{-iE_0t}|\psi_0\rangle - e^{-iE_1t}|\psi_1\rangle) \\ &= \frac{e^{-iE_0t}}{\sqrt{2}}(|\psi_0\rangle - e^{-i(E_1-E_0)t}|\psi_1\rangle). \end{aligned}$$

This means that the Hamiltonian drives the transitions between the initial state and the target node, and after a time $t \approx \pi/(E_1 - E_0)$ the walker has transferred from the initial state to the target node. This is because then the term $e^{-i(E_1-E_0)t} = -1$, and $e^{-iHt}|s\rangle$ equals $|w\rangle$ up to a phase factor.

Now, if γ is chosen to be $1/N$ as in the case of the complete graph, the Hamiltonian can be written as

$$H = \frac{1}{N}L - |w\rangle\langle w|,$$

where the Laplacian of the complete graph is assumed for L. Now

$$H|s\rangle = -\frac{1}{\sqrt{N}}|w\rangle, \quad H|w\rangle = -\frac{1}{\sqrt{N}}|s\rangle.$$

This is the same as Eq. (9), so therefore after time $T = \pi\sqrt{N}/2$ the walker has transitioned from the initial state to the target node. This can be seen to be true for the case $N = 1024$, since the numerical solution was $T \approx 50.27$ and the algebraic

solution is $T = \pi\sqrt{1024}/2 \approx 50.27$. The walker reaches the target node with a probability of 1 at this time, and because $T = \mathcal{O}(\sqrt{N})$, the algorithm is optimal for the complete graph. In fact, it is optimal for all cases of N. Farhi and Gutmann showed that the quantum walk on a complete graph is an "analog analogue" of Grover's algorithm [10]. The complete graph can be considered as a fully connected database, and therefore this spatial search algorithm for the complete graph considers the same problem as the Grover's algorithm, just with a different method.

5 Hypercube

The hypercube is also a regular graph, so only one target node needs to be observed. Here I consider a 10-dimensional hypercube, so that means that there are $2^{10} = 1024$ nodes. All of the nodes have a degree of $k = 9$.

The first plot in figure 10 shows the energy gap and overlaps for the 10-dimensional hypercube. If this is compared to the plot by Childs and Goldstone in *Spatial search by quantum walk*, it can be seen that the plot is similar, but the optimal γ is half that of the one given in the paper [2]. This is actually because Childs and Goldstone use a different Hamiltonian for the walker that is defined with the Pauli sigma x operator. Once again it can be seen that the optimal value of γ is about $1/k$ as that is the value of γ with a swift change of the states of the walker and the lowest energy gap. The numerically calculated value for the optimal γ is about 0.1144, so a 0.0033 difference to 1/9. The overlap of the target state with the first excited state can be seen to decrease after the optimal γ, and the energy gap also does not increase as quickly.

The success probability can be seen to not quite reach 1 for the optimal success probability that has been calculated numerically in the second plot in figure 10, but it is still of order 1 so the CTQW is optimal for this graph as well.

6 Circle graph

The circle graph showcases how the spatial search algorithm fails in low dimensions. The circle graph is a 1-dimensional lattice, where the ends are simply connected, thus creating a loop.

The first plot in figure 11 shows how the overlaps and the energy gap behave for a circle graph of size $N = 1024$. There is no drastic change from a significant overlap with $|\langle s|\psi_1\rangle|^2$ and $|\langle w|\psi_0\rangle|^2$ to a significant overlap with $|\langle s|\psi_0\rangle|^2$ and $|\langle w|\psi_1\rangle|^2$, even

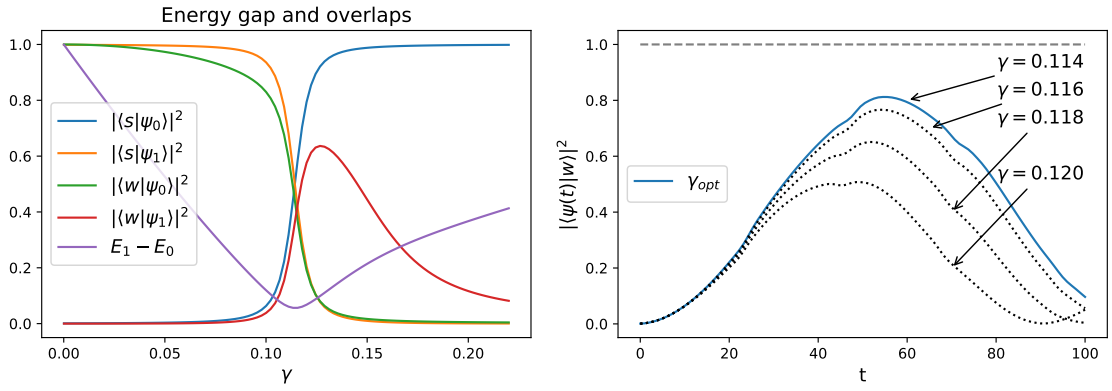


Figure 10: The first plot gives energy gap and overlaps for a 10-dimensional hypercube ($2^{10} = 1024$ nodes). The second plot gives the success probability as a function of time with the optimal and three suboptimal γ for the 10-dimensional hypercube with $N = 1024$ nodes.

if this plot would be continued further to higher values of γ.

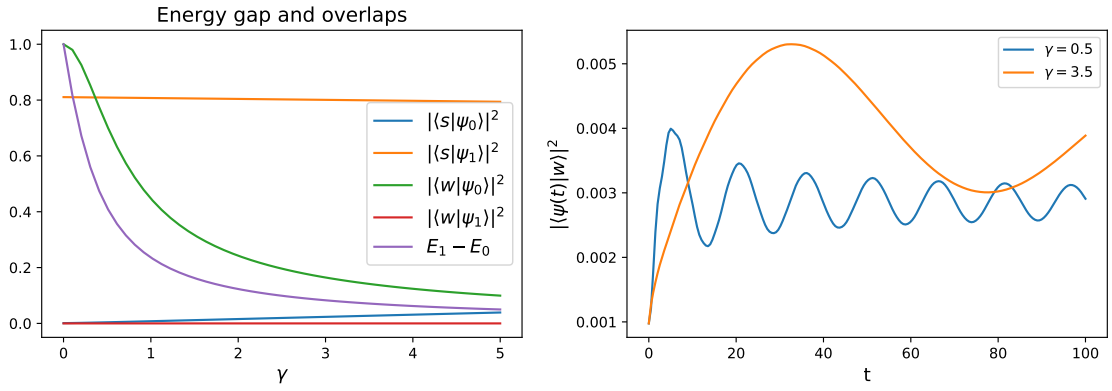


Figure 11: The energy gap and overlaps, as well as the success probability as a function of time with two different γ for the circle graph with $N = 1024$ nodes.

The second plot in figure 11 gives the success probability as a function of time for two different γ for the same circle graph. Here it can be seen that there is not really one particular value for optimal γ, since by changing the value of γ the probability can be higher for an earlier value of time, but for a different γ the success probability can get higher later on. The expected optimal value of γ is $1/2$, but by changing the initial value even a little bit in the numerical computation results in vastly different results for the value of γ. The optimization of time also does not really

work for such low success probabilities, so the resulting time from the numerical computation can be later than where the initial peak is for a certain value of γ. The success probability never reaches the order of 1 so it is not optimal.

7 Star graph

The star graph is a semi-regular graph with two topologically different nodes in the graph, a central node and the outer nodes, so two different nodes will be analysed for this graph type. The spatial search problem on the star graph is studied extensively in [34].

Figure 12 shows the energy gap and overlaps for the central and outer nodes for a star graph of size $N = 1024$. For both cases, there is a drastic change for the overlaps of the ground state of the Hamiltonian with the initial state and the target node. The optimal γ are where they are expected, $\gamma_{\text{opt}} \approx 1/1023$ for the central node and $\gamma_{\text{opt}} \approx 1$ for the outer nodes. A different excited state and energy needed to be used for the central node, since it behaves a little bit differently. Typically, the first two algebraically lowest eigenvalues and their corresponding eigenvectors are picked for the ground and excited energies and states, but for the central node the excited state and energy that refer to the target state correspond to the algebraically largest eigenvalue of the Hamiltonian. The other, algebraically second to lowest eigenvalue is in fact $N-2$ -degenerate. An example can be analysed by taking a star graph with 5 nodes, so 4 outer nodes, and looking at the spectrum of the Hamiltonian, when the target node $|w\rangle = |0\rangle$ is the central node. The eigenvalues are

$$\lambda_1 = \frac{1}{2}(-A + 5\gamma - 1),\ \lambda_2 = \lambda_3 = \lambda_4 = \gamma,$$
$$\lambda_5 = \frac{1}{2}(A + 5\gamma - 1),$$

where $A = \sqrt{25\gamma^2 - 6\gamma + 1}$, and the corresponding eigenvectors are

$$v_1 = \begin{pmatrix} -\frac{-1+3\gamma-A}{2\gamma} \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix},\ v_2 = \begin{pmatrix} 0 \\ -1 \\ 1 \\ 0 \\ 0 \end{pmatrix},\ v_3 = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \end{pmatrix},\ v_4 = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{pmatrix},\ v_5 = \begin{pmatrix} -\frac{-1+3\gamma+A}{2\gamma} \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

The first and last eigenvalues and their corresponding eigenvectors are the ground and highest excited states and energies, respectively. The eigenvalues are given

in rising algebraic order, and the middle eigenvalue is 3-degenerate. In fact, the degenerate eigenvalue and it's corresponding eigenvectors behave the same way for general star graphs with N nodes, with the eigenvectors given as

$$v = (0, -1, \underbrace{0, ..., 0}_{j}, 1, \underbrace{0, ..., 0}_{k})^\top, \quad j, k \in \mathbb{N},\ j + k = N - 3.$$

Now it can be seen that in the star graph with the central node as the target node, both $|s\rangle$ and $|w\rangle$ have zero overlap with the $N-2$ -degenerate eigenspace, and thus the transitions are defined with the ground state and highest excited state.

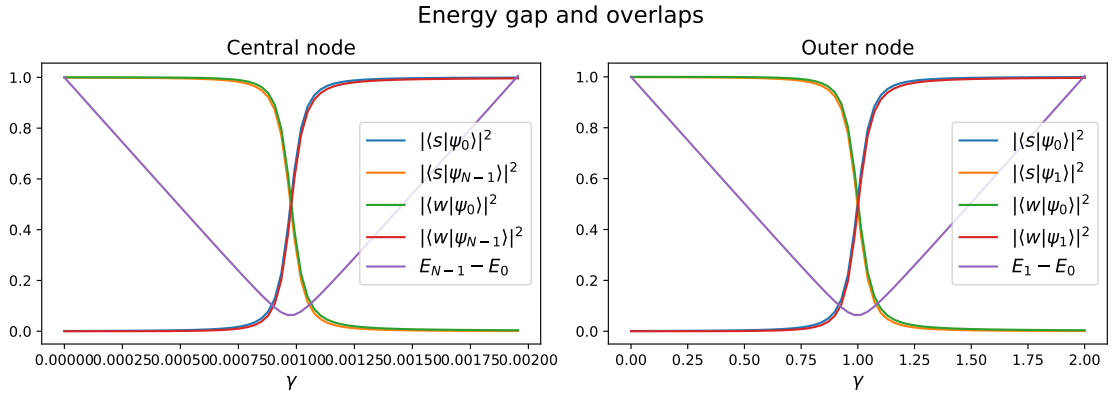


Figure 12: The energy gap and overlaps of a star graph with $N = 1024$ nodes. The walker with the target node as the central node now uses the highest excited state and energy of the Hamiltonian that correspond with the target node.

Figure 13 gives the success probabilities as functions of time for the central and outer nodes of a star graph with $N = 1024$ for the optimal γ. The success probabilities are optimal, reaching one for the optimal values of γ and T, as is to be expected by looking at the overlaps.

8 Erdős–Rényi graph

The Erdős–Rényi graph is a bit different than previous examples. It is a semi-regular graph, so it has nodes with varying degrees as compared to previous graphs, and it is also a random graph, so by changing the variables used in the generation of an ER graph one can get significantly different graphs. Because of the varying degrees, all of the individual overlaps, optimal γ and success probabilities cannot be analysed. This is why I give the overlaps and success probability as a function of time for the optimal

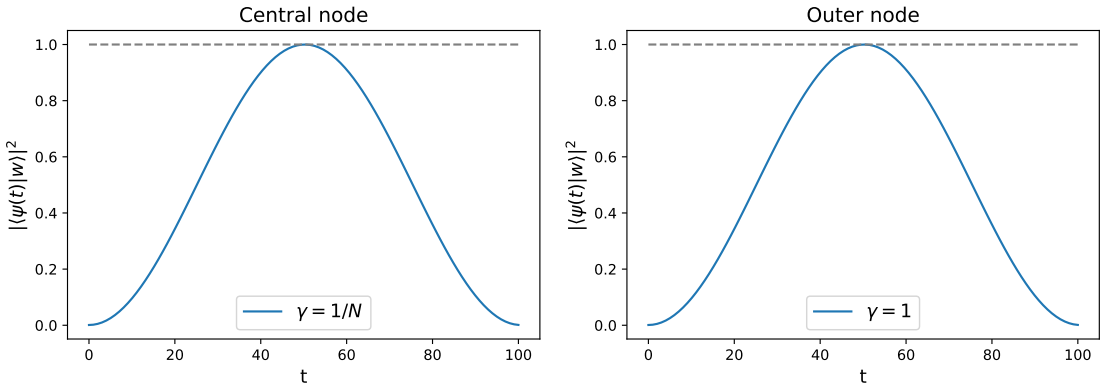


Figure 13: Success probabilities as functions of time with the optimal γ for the central and outer nodes of the star graph with $N = 1024$ nodes.

γ for one node, since for the ER graph the differences between node topologies are typically not that large. After this, the optimal γ and success probabilities of all nodes are plotted, and these results are analysed. For this section I will be using ER graphs with $N = 1024$ nodes and different probabilities of generating a link between each pair of nodes, particularly I will study two ER graphs with probabilities 0.2 and 0.005. The generated graphs are the same throughout this section. Typically, ER graphs with $N = 1024$ and $p = 0.005$ tend to create disconnected nodes, so I generated the other ER graph with extra nodes, and then deleted the disconnected ones, so that the resulting graph was fully connected with 1024 nodes. In the end I will look at how the average optimal success probabilities change with respect to average degrees when the probability of generating links is varied in ER graphs of the same size.

Figure 14 gives the degree distribution of these two ER graphs. For the ER graph with $p = 0.2$ it is roughly a normal distribution, with the peak at around 200. This is because there are $N = 1024$ nodes and a single node will on average be connected to $(N - 1) \times 0.2 \approx 205$ other nodes (the actual average degree of this specific ER graph is $\langle k \rangle = 204.36$). For the ER graph with $p = 0.005$ it is not as much a normal distribution, which is because the average degree is so low, around 5, and there is not as much variance in the degrees of nodes.

The energy gap and overlaps of these ER graphs are given by figure 15 for nodes with degree 202 on the left and degree 6 on the right, and the ER graphs used were generated with probabilities 0.2 and 0.005, respectively. Just as with the complete graph, a sharp change can be seen for the overlaps on the left so that the ground

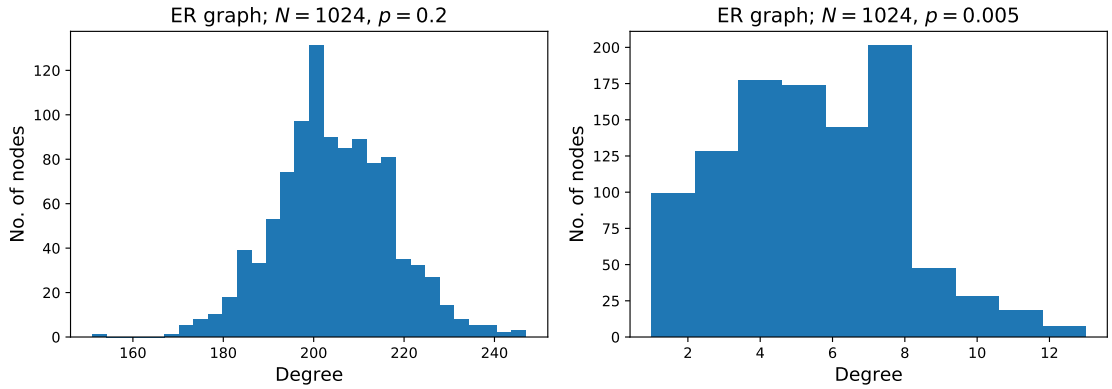


Figure 14: The degree distribution of two ER graphs with $N = 1024$ nodes and $p = 0.2$ or $p = 0.005$ probability of creating a link between each pair of nodes.

state switches from $|w\rangle$ to $|s\rangle$ and first excited state switches from $|s\rangle$ to $|w\rangle$ at the critical value of γ, where the energy gap between the two states is the lowest. The optimal value of γ comes right at about the value $1/k \approx 0.005$. From this plot it can be assumed that the algorithm is optimal for this graph and this node, and that the success probability will also be quite high with the optimal γ.

In the case of the ER graph with $p = 0.005$ in figure 15, the change between different overlaps is not as sharp, and after the point of optimal γ the overlap between $|\psi_1\rangle$ and $|w\rangle$ collapses, and also the behaviour of the energy gap changes. This is something that could also be seen to a point with the hypercube, and it is a concern of when the graph is sparse enough.

Figure 16 gives the success probabilities as functions of time with the optimal γ on the left and right for nodes with degree 202 on the left and degree 6 on the right, and the ER graphs used were generated with probabilities 0.2 and 0.005, the same nodes that were used in figure 15. The algorithm can be seen to be optimal with the optimal γ on the left, and this result is very similar to the complete graph case. This type of an ER graph can actually be acquired from the complete graph of size $N = 1024$ by deleting individual links with probability 0.8. The resulting graph remains sufficiently dense and interconnected so that the spatial search by quantum walk on the graph is also optimal. The optimal success probability on the right does not get as high, but it is still of order 1 and optimal.

The upper two plots in figure 17 give the optimal γ of all nodes with respect to the degrees for the two ER graphs in a logarithmic scale. A logarithmic scale is used to highlight the fact that the relation $\gamma_{\text{opt}} = c/k$, with some constant $c \approx 1$, holds

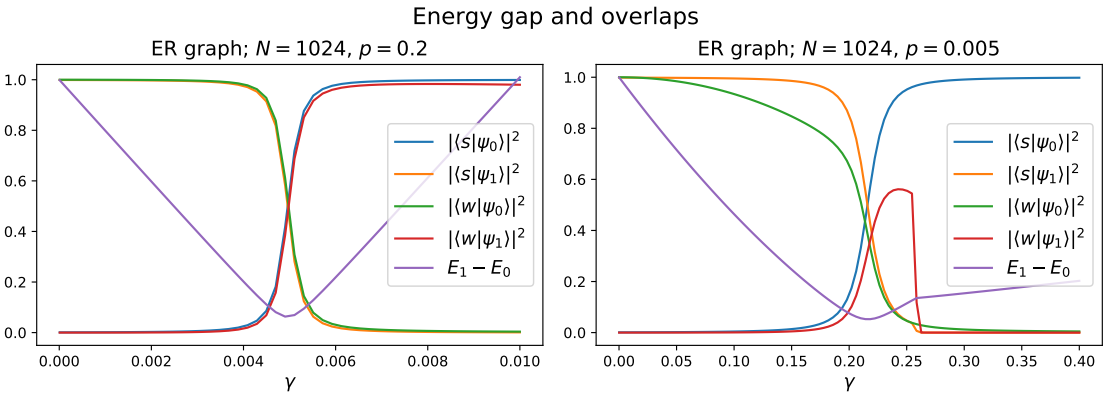


Figure 15: The energy gap and overlaps of nodes with degree 202 on the left and degree 6 on the right. The ER graphs used were generated with probabilities 0.2 and 0.005 of creating a link between each pair of nodes.

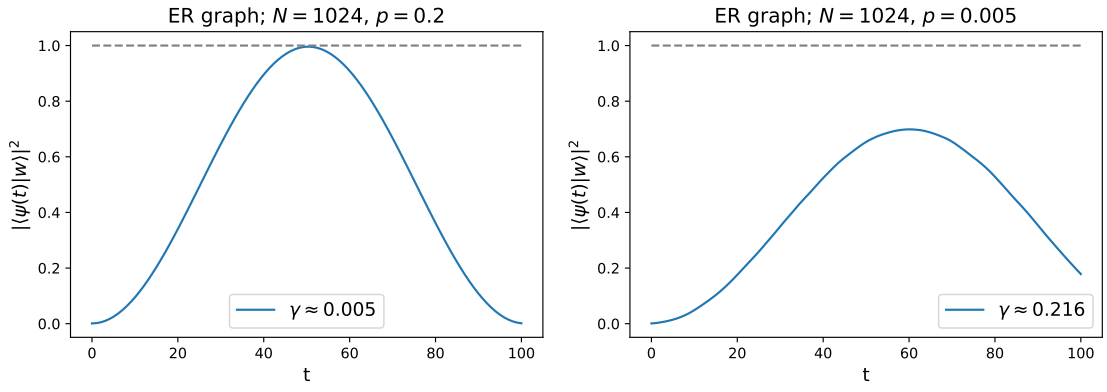


Figure 16: The success probabilities of nodes with degree 202 on the left and degree 6 on the right, and the ER graphs used were generated with probabilities $p = 0.2$ and $p = 0.005$ of creating a link between each pair of nodes.

true, since there is a linear relation with the optimal γ and degree in a logarithmic scale. There is more deviation from the line that can be seen on the right which can be attributed to the fact that individual node topologies are more varied for nodes with similar degrees especially when the degrees are low.

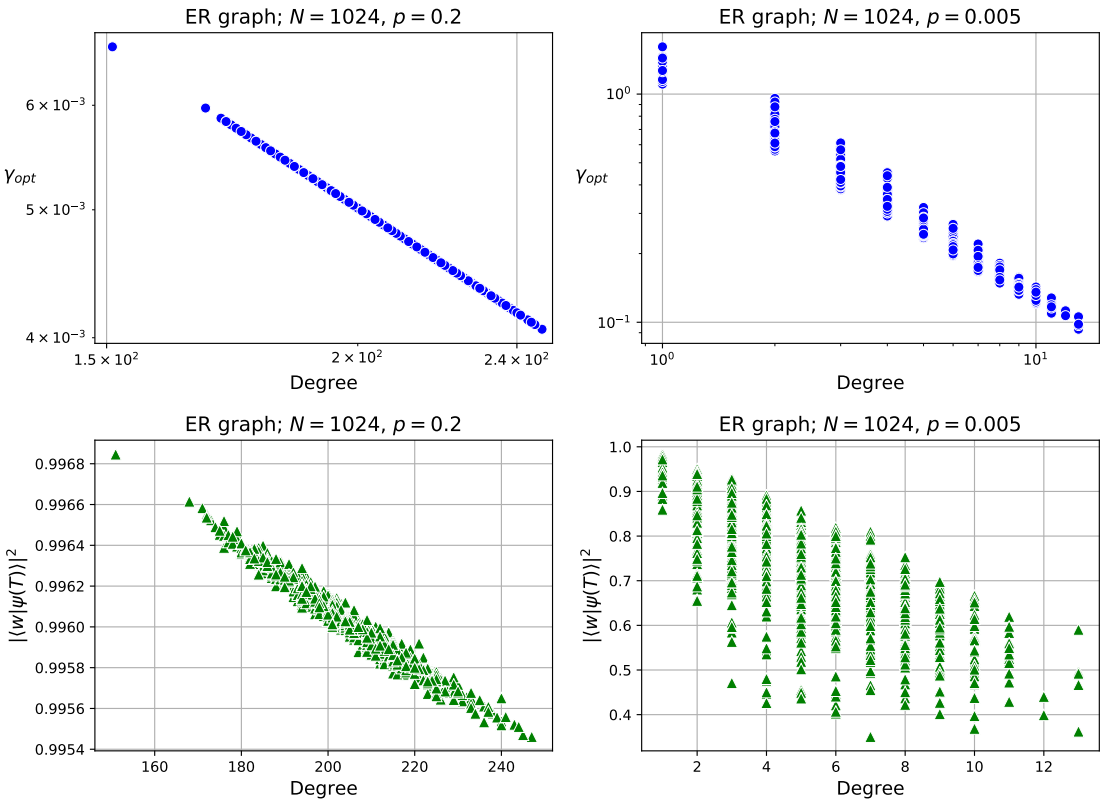


Figure 17: The upper plots give the optimal γ of all nodes with respect to the degrees in ER graphs with $N = 1024$ nodes and probabilities $p = 0.2$ and $p = 0.005$ of creating a link between each pair of nodes. The plots are in a logarithmic scale. The lower plots give the optimal success probabilities of all nodes with respect to the degrees in the same two ER graphs.

The lower two plots in figure 17 give the optimal success probabilities with respect to the degrees of all nodes for the two ER graphs, i.e. the value found with the optimal γ and T for all the individual nodes. The success probability has a roughly linear dependence on the degree of the target node, but there is more variance for the optimal success probabilities with similar values of degree k. The relation between optimal success probabilities and degrees is not linear in general, as will be seen later. This just typically happens to be so for this ER graph. Obviously on the right there is even more variance, and the data points do not form a discernible curve. The

success probabilities are a lot higher for the denser ER graph, they are all very optimal. Also none of the nodes in the sparser ER graph have success probabilities as high as any nodes in the denser one, so even the most easily searchable node in the sparser ER graph can not be found as easily as any node in the denser ER graph.

Figure 18 shows the averages and variances of optimal success probabilities with respect to the average degrees of multiple different ER graphs that all have 1024 nodes. When the probability of creating links between pairs of nodes gets large, the success probabilities also tend to get really close to one. When p goes to zero, the average optimal success probability also goes down and approaches zero, and this is because the graph becomes sparse, where the spatial search by quantum walk is no longer optimal. Chakraborty et al. showed that spatial search by quantum walk is optimal for almost all graphs [17]. 'Almost all graphs' here is defined with ER graphs since the set of ER graphs of size N contain all possible graphs of that size. This is because links between all pairs of nodes are individually generated with probability p. Specifically, they showed that spatial search by CTQW is almost surely optimal for ER graphs of N nodes, as long as the probability $p \geq \log^{3/2}(N)/N$. For the case when $N = 1024$, the probability $p \geq 0.0178$ in which case the average degree is about $\langle k \rangle = 18$. This is also where the average optimal success probability starts to go significantly down in figure 18 so the optimality of the spatial search by CTQW cannot anymore be guaranteed. Note that almost surely optimal means that it will be almost surely optimal for the ER graph generated since the links of the generated graph will most likely be evenly distributed between nodes, and for example the degree distribution of the graph will most likely follow a normal distribution around the average degree, and not some more complex distribution.

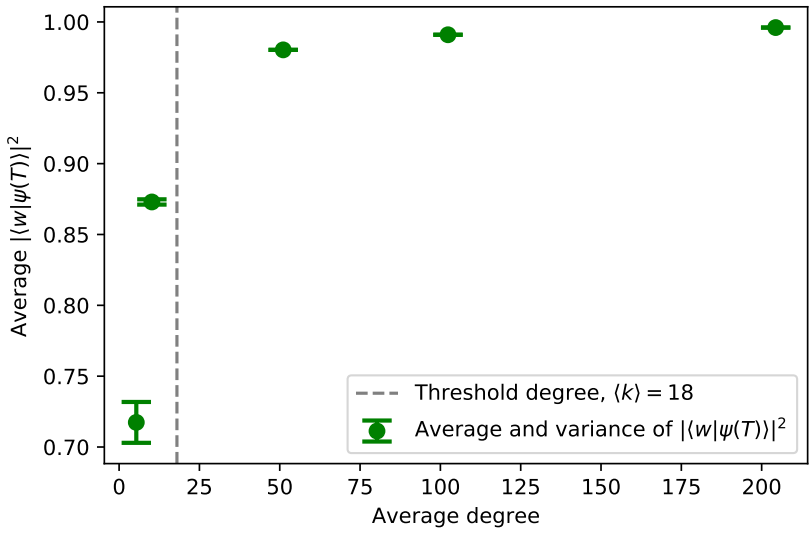


Figure 18: The averages and variances of optimal success probabilities with respect to the averages of degrees for different ER graphs with 1024 nodes. The ER graphs were generated with linking probabilities 0.005, 0.01, 0.05, 0.1 and 0.2, going from left to right. The threshold degree gives the critical average degree of an ER graph of size $N = 1024$ for which higher average degrees the spatial search problem will still be optimal.

Chapter III

CTQW on Complex Networks

Complex networks are quite different from regular and semi-regular ones, since there are quite different node topologies for different nodes in a complex network that can result in some nodes having optimal results for the γ, t and $|\langle w|\psi(t)\rangle|^2$, while other nodes have not in the same network. Therefore, using the previous method of analysis by looking at the overlaps will not be as useful in this case, and looking at the network structure as a whole becomes more important. This is why the method introduced for the Erdős–Rényi graphs is more extensively used, where different properties of the network and the CTQW are compared for all nodes. The energy gap and overlap analysis can still be used to investigate the behaviour for specific nodes, and this can give some direction for success on the whole network, but it will not give as much information for the success of the spatial search by a CTQW on the network as a whole.

Two separate kinds of complex networks are considered for this chapter: Barabási-Albert graphs and *the Internet network* that is used in [6]. The BA graphs are randomly generated, and they follow the power law for the degree distribution, so hubs typically form in them. The Internet network is a real-world network at the level of autonomous systems (AS), in which the autonomous systems are nodes and connections between these are links [30]. This Internet network corresponds to data gathered in 2009. The full Internet network has 23 748 nodes and 58 414 links, but there are also five different layers of renormalized and pruned versions that have been obtained by using the renormalization and pruning techniques introduced in Sec. 1.9, and each layer has about half as many nodes as the layer above it, so that layer $l = 5$ has 743 nodes. On top of analysing these networks in isolation, the results from ER and BA graphs that are made to mimic different layers of the Internet network will also be compared to the Internet networks. The mimicry will

be achieved by setting the number of nodes and links to be the same for the different graphs.

9 Barabási-Albert graph

The same method will be used for this section as for the Erdős–Rényi graph section to begin with. Namely, using two different values of m in the generation of BA graphs so that each new node is linked to m pre-existing nodes using preferential attachment. However, since BA graphs are complex networks and the degree distributions follow a power law, the results for different values of m can vary a lot, especially in the sparse limit, where m is low. Two different BA graphs of size $N = 1024$ will be observed, with values $m = 100$ and $m = 2$.

Figure 19 shows the degree distributions of these BA graphs, and they can be seen to roughly follow the power law. The graph with $m = 100$ has an average degree of 180.5, and the graph with $m = 2$ has 4.0, so these averages are quite close to the graphs used in the Erdős–Rényi section.

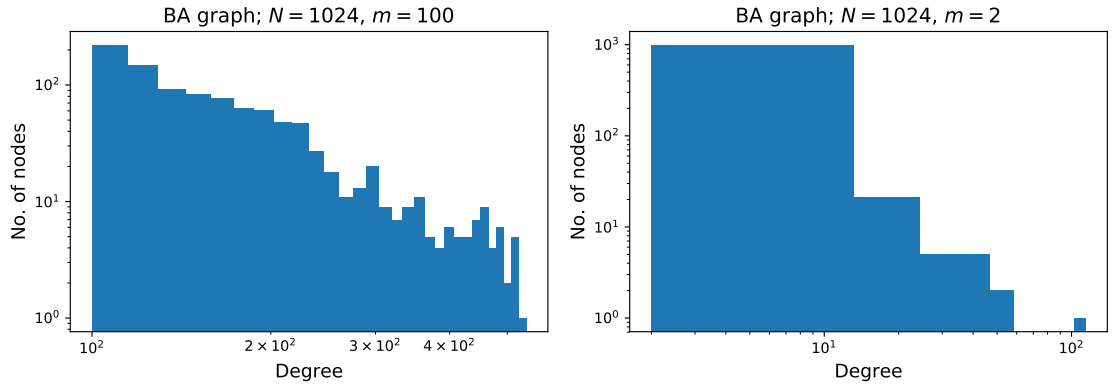


Figure 19: The degree distribution of two BA graphs with $N = 1024$ nodes and with linking each new node to $m = 100$ or $m = 2$ pre-existing nodes using preferential attachment. The histograms are in logarithmic scale in order to see the power law more clearly.

Now, because of the larger variance in the degrees of different nodes and sparser BA graphs have both good and bad optimal success probabilities, the overlaps also vary quite a lot for those BA graphs. Figure 20 shows the energy gaps and overlaps for the two BA graphs, with two different plots for two different nodes in the $m = 2$ case. With both the $m = 100$ BA graph and the low-degree node case of the $m = 2$

BA graph, there is a clear rotation of the states of the walker between the initial and target states, but not for the high-degree node of the $m = 2$ BA graph. The overlaps are pretty much the same for the $m = 100$ BA graph, as will also be seen later from the optimal success probabilities.

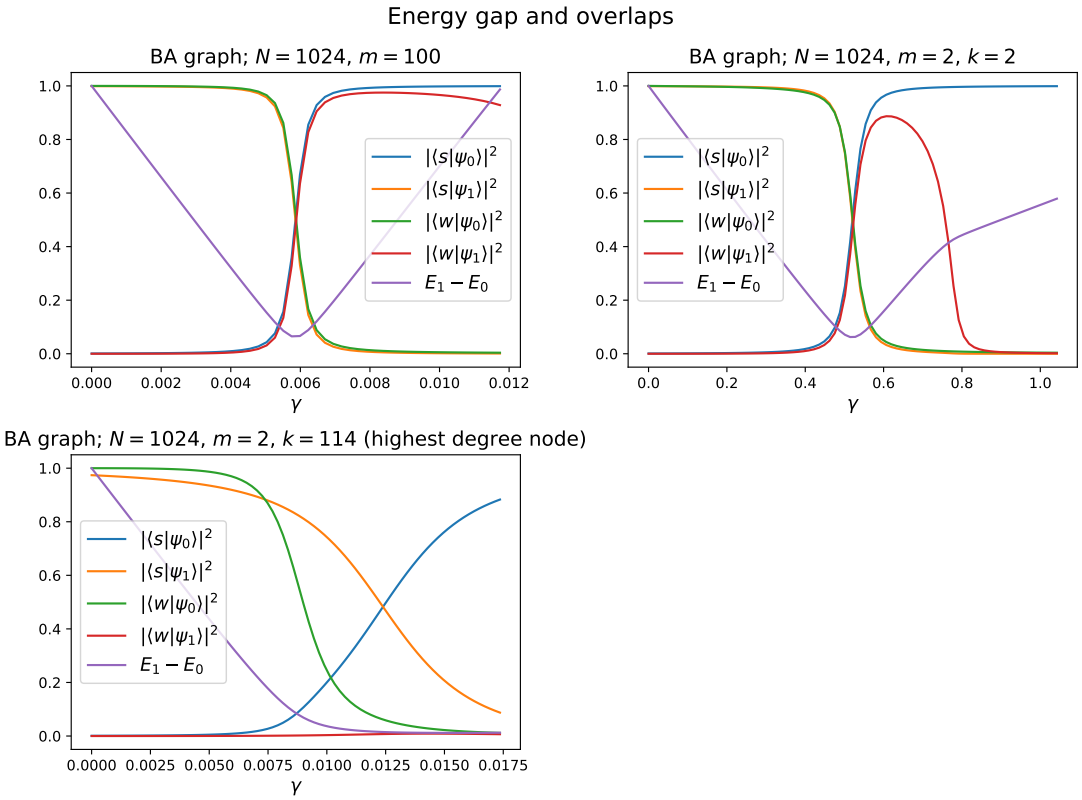


Figure 20: The energy gap and overlaps of two BA graphs with $N = 1024$ nodes and with linking each new node to $m = 100$ or $m = 2$ pre-existing nodes using preferential attachment. The BA graph with $m = 2$ has two different overlap plots given for it, one for a node with degree $k = 2$ and another with $k = 114$, the node with the highest degree in the graph.

The upper two plots in figure 21 give the optimal γ with respect to the degrees of all nodes for the two BA graphs in logarithmic scale. The dense BA graph is completely linear, but the sparse BA graph's optimal γ can be seen splitting into two distinct lines, with the upper line showing the same spread of optimal γ with respect to a certain degree as was seen in the case of the ER graph in figure 17. The lower line does not have a spread, and the related nodes have higher degrees on average. This splitting is a phenomenon that can be seen quite often for sparse complex networks that follow the power law.

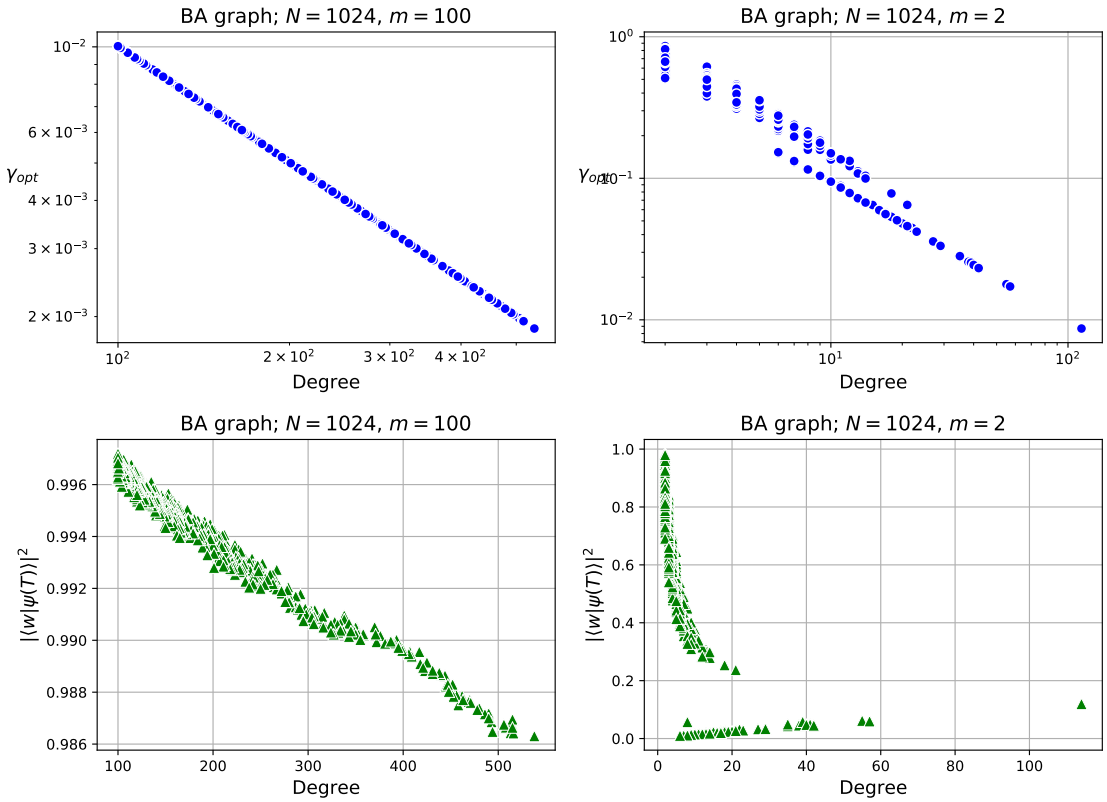


Figure 21: The upper two plots give the optimal γ of all nodes with respect to the degrees in BA graphs with $N = 1024$ nodes in a logarithmic scale, and with linking each new node to $m = 100$ or $m = 2$ pre-existing nodes using preferential attachment. The lower two plots give the optimal success probabilities of all nodes with respect to the degrees for the same two BA graphs.

The lower two plots in figure 21 give the optimal success probabilities with respect to the degrees of all nodes for the BA graphs. The dense BA graph can be seen to be optimal for all nodes, which is to be expected. The sparse BA graph once again has two distinct parts, where the optimal success probabilities behave differently. For low-degree nodes, there is a rapid decrease in the value of the optimal success probability, but there is also another non-optimal part close to zero, where the optimal success probability increases linearly with respect to the degree.

Figure 22 shows the BA graph with $m = 2$, where the sizes of the nodes describe the degree and the colors of the nodes show the success probability of the CTQW on the graph, with red being close to zero and blue being close to one. The nodes have been mapped in such a way that nodes that are connected to a lot of nodes and/or to high-degree nodes are close to the center, and other nodes are on the sidelines. The nodes with the highest success probabilities tend to be low-degree nodes that are connected to high-degree nodes with low probabilities. However, a clear structure cannot be seen for how successful the quantum walk search is for a specific node in a complex graph. This will be further analysed in the next section.

Figure 23 gives the averages and variances of the optimal success probabilities of BA graphs with $N = 1024$ nodes with different values of the linking number m, as well as the averages and variances of the optimal success probabilities of ER graphs from figure 18. The average optimal success probabilities of the BA graphs obviously follow the same curve as those of the ER graphs, with the exception of the variances of BA graphs getting bigger at the low-degree limit, which is to be expected from the power law versus the Gaussian distribution of the degrees.

Figure 24 gives the averages of the optimal success probabilities, times and search times with respect to the sizes of graphs of ER and BA graphs with different sized graphs and with different values of p and m. For the success probabilities, the probability rises with N for all values of p in the ER graphs, asymptotically approaching one as $N \to \infty$. This is because in the ER graph, the number of generated links increases with larger N, and the graph becomes more connected with a larger N for the same value of p. The success probabilities stay the same for the same value of m for the BA graphs, which is because BA graphs create hubs through which nodes in the graph are more closely connected. The averages of optimal times behave quite similarly for both types of graphs, less connected graphs have higher average optimal times and the values of the optimal times get higher with increasing N, as the state of the walker takes longer to rotate from the initial state to the target state. They are also linear in loglog scale, so the times are of order $\mathcal{O}(\sqrt{N})$. The average

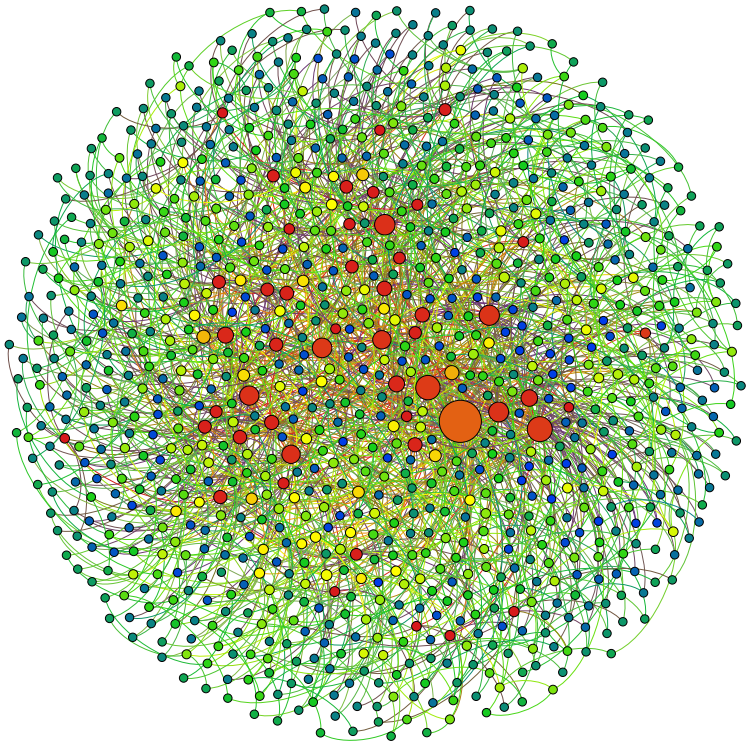


Figure 22: The BA graph with $N = 1024$ and $m = 2$. The sizes of the nodes describe the degree of the node, and the colors describe the optimal success probability of a CTQW with that node as the target node, with red being close to zero and blue being close to one.

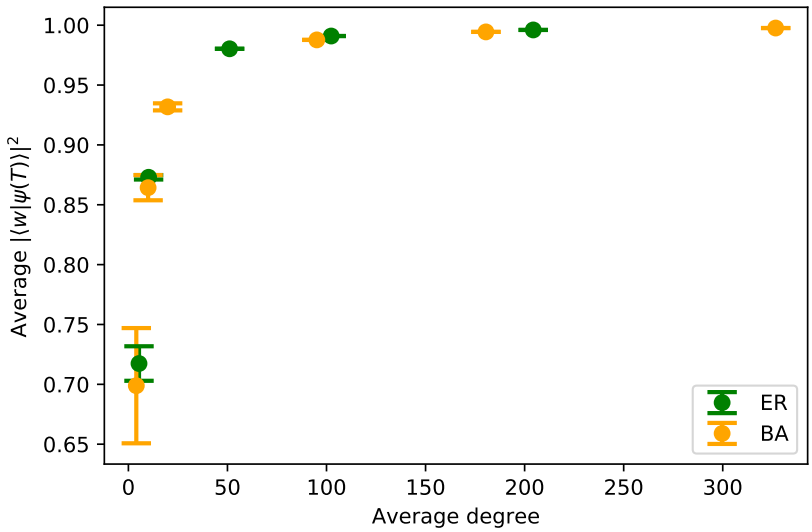


Figure 23: The averages and variances of optimal success probabilities with respect to the averages of degrees for different ER and BA graphs with 1024 nodes. The BA graphs were generated with linking numbers 2, 5, 10, 50, 100 and 204, going from left to right, and the ER graphs are the same as in figure 18.

T for the $p = 0.005$ value goes down for the low N and this is most likely caused by how the graph is created. For the ER graph with $N \approx 342$ and $p = 0.005$ about 50 more nodes needed to be originally generated, because about 50 of the nodes were completely disconnected in the generated graph, and these nodes are then removed. Because such a high fraction of nodes needs to be removed, the resulting graph is actually more connected than would be expected for these values of N and p for the ER graph. This problem is not so prominent in the bigger ER graphs with $p = 0.005$ because there are more chances of connecting nodes to each other, so not as many nodes need to be removed and the fraction of removed nodes is smaller regardless.

The search times $T_w = T/|\langle w|\psi(T)\rangle|^2$ give the actual amount of time that the walk needs to run for so that the target node is reliably reached, since the times T are divided with the success probabilities $|\langle w|\psi(T)\rangle|^2$, so that with a lower success probability the actual search time is longer. The search times in figure 24 were calculated by taking the search time of each node separately, and then the averages of these search times were taken. The results for the search times can not directly be surmised from the other plots in figure 24, since the average of search times can be different from the average of times over the average of success probabilities.

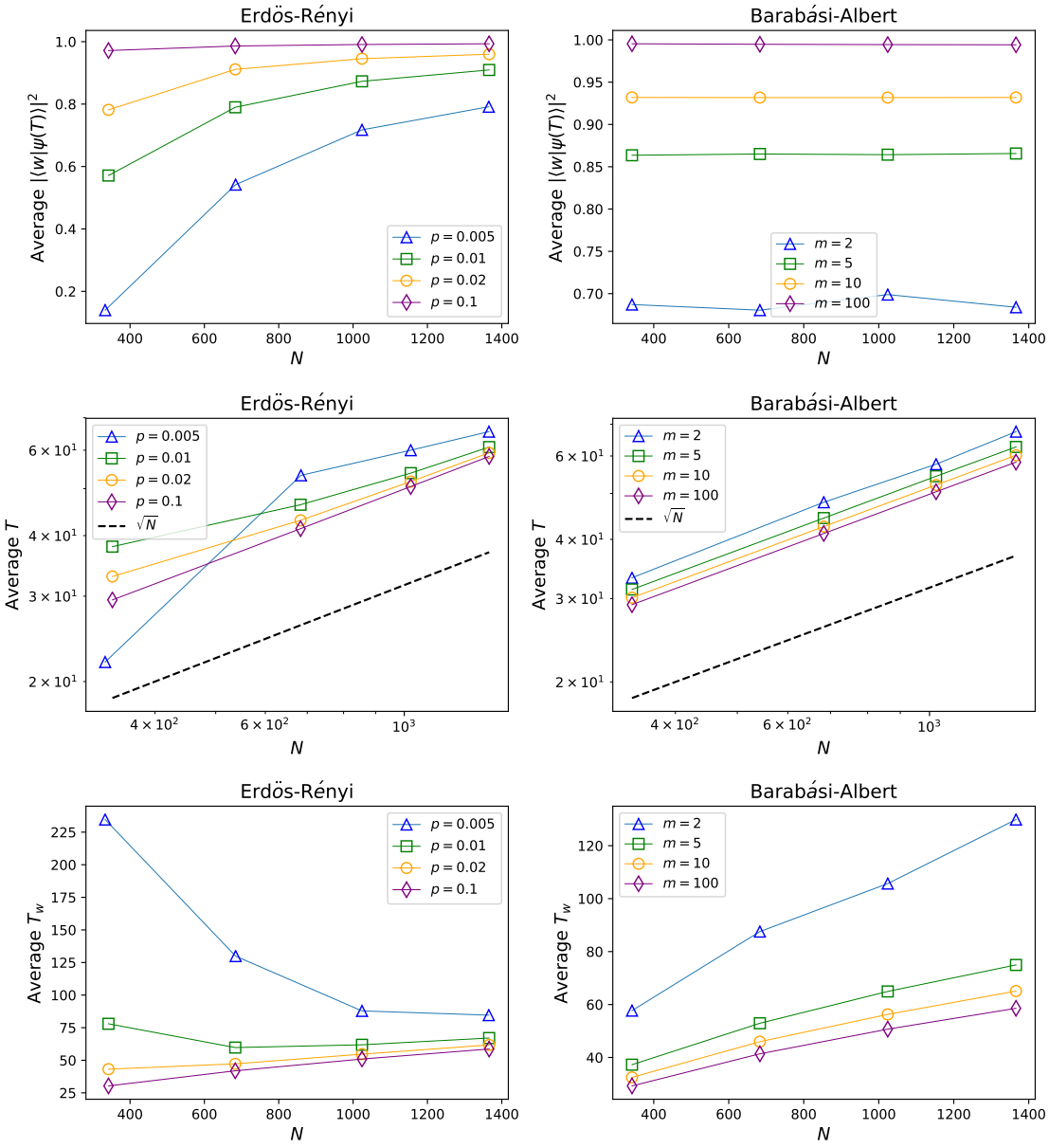


Figure 24: The average optimal success probabilities, times and search times with respect to the sizes of graphs of ER and BA graphs of different sizes with different values of p and m.

10 The Internet

As stated at the introduction of this chapter, the Internet network used here is on the level of autonomous systems and the full network has 23 748 nodes and 58 414 links [30]. The method for obtaining and the subsequent analysis of the renormalized and pruned versions are in [6]. There are five layers of the renormalized and pruned versions and the number of nodes from layer $l = 1$ to $l = 5$ are $N_{1,\mathrm{R}} = 11874$, $N_{2,\mathrm{R}} = 5937$, $N_{3,\mathrm{R}} = 2969$, $N_{4,\mathrm{R}} = 1485$ and $N_{5,\mathrm{R}} = 743$ for the renormalized versions (R for renormalized), and $N_{1,\mathrm{P}} = 11636$, $N_{2,\mathrm{P}} = 5737$, $N_{3,\mathrm{P}} = 2857$, $N_{4,\mathrm{P}} = 1412$ and $N_{5,\mathrm{P}} = 701$ for the pruned versions (P for pruned). Here $N_{l,\mathrm{V}}$, where $l = 1, 2, 3, 4, 5$ and $V = R, P$.

In this section, I will first look at the renormalized and pruned Internet networks of layer $l = 5$, especially focusing on what makes them different and why. After this, all the different layers will be looked at and compared with each other. At the end I will compare the layers from $l = 5$ to $l = 3$ to generated ER and BA graphs with approximately the same amount of nodes and links.

10.1 Renormalized and pruned Internet, layer $l = 5$

Looking at individual overlaps for specific target nodes does not offer that much information at this point, so I will glance over the results for a couple of individual nodes before moving on to the whole networks.

Figure 25 shows the energy gaps and overlaps for two different nodes of both renormalized and pruned Internet networks of layer $l = 5$. The highest-degree nodes of both networks are given, as well as examples of nodes with degree one. The corresponding optimal success probabilities for the renormalized cases are quite high for both, approximately 0.7741 for the $k = 563$ node and 0.9995 for the $k = 1$ node. Figure 26 shows the visualized renormalized $l = 5$ Internet network with colors for different optimal success probabilities. All nodes of degree one have optimal success probabilities close to one. The corresponding optimal success probabilities for the pruned cases are 0.2223 for the $k = 156$ node and 0.1260 for the $k = 1$ node. Figure 27 shows the visualized pruned $l = 5$ Internet network. The $k = 1$ nodes have highly differing optimal success probabilities, and a node with a low probability was chosen for figure 25.

Some of the differences between the renormalized and pruned versions of the $l = 5$ Internet network can be seen from figures 26 and 27. The graphs have been drawn with a force-directed graph drawing algorithm, that tries to make all edges have

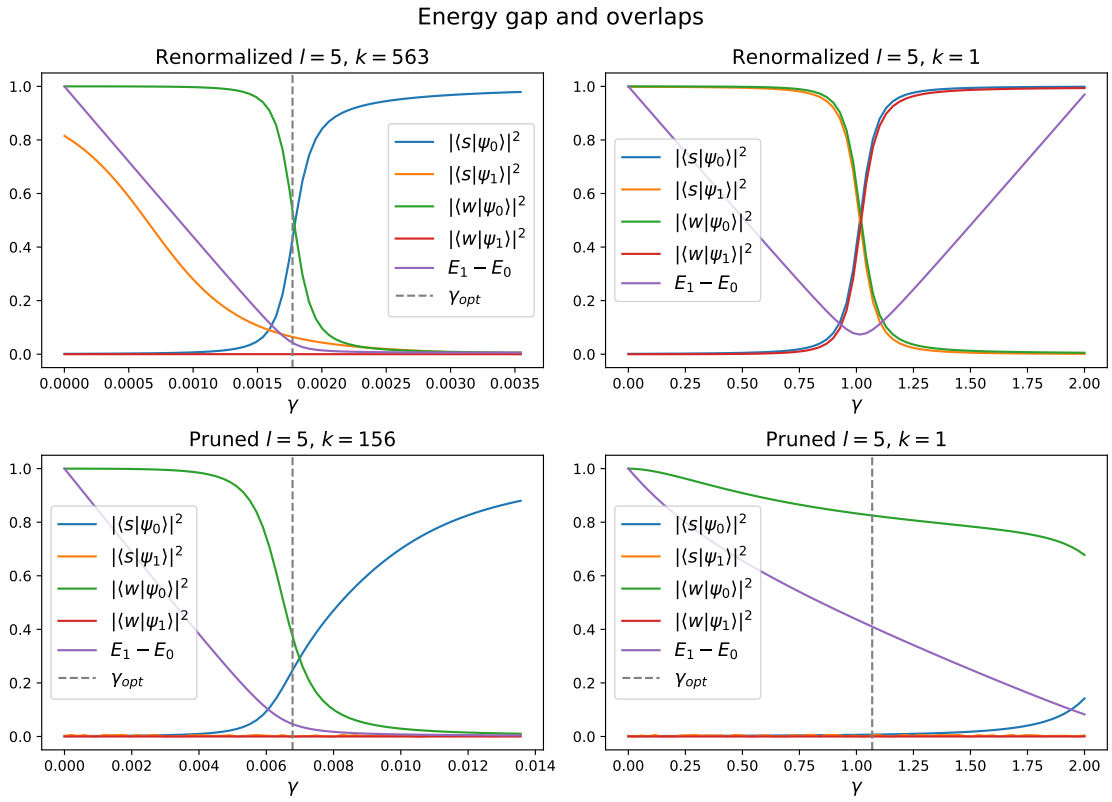


Figure 25: The energy gaps and overlaps for the highest-degree nodes and nodes of degree one of both renormalized and pruned versions of the Internet network of layer $l = 5$.

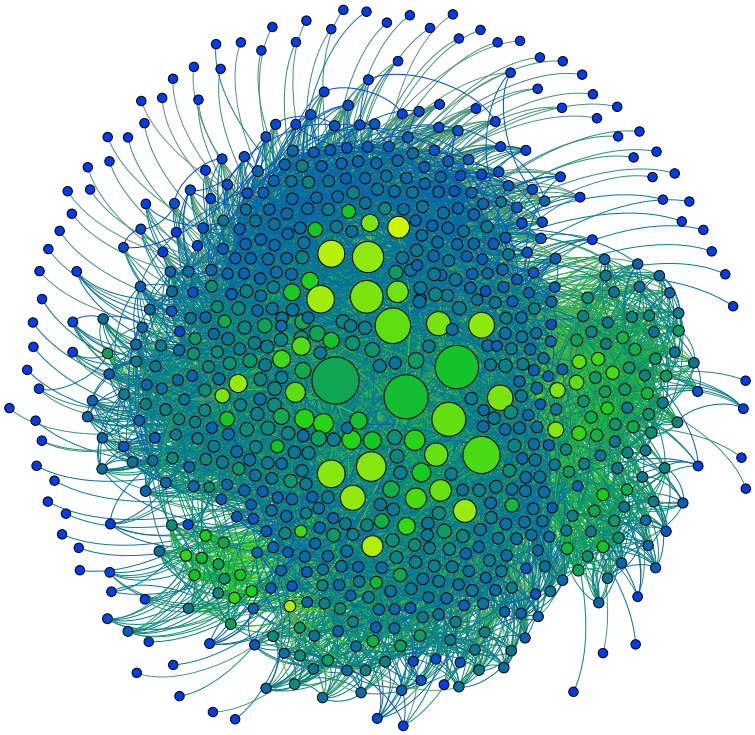


Figure 26: The renormalized Internet graph of layer $l = 5$. The sizes of the nodes describe their relative degrees, while the colours of the nodes describe their optimal success probabilities, blue being close to one and the most yellow node to be approximately 0.42.

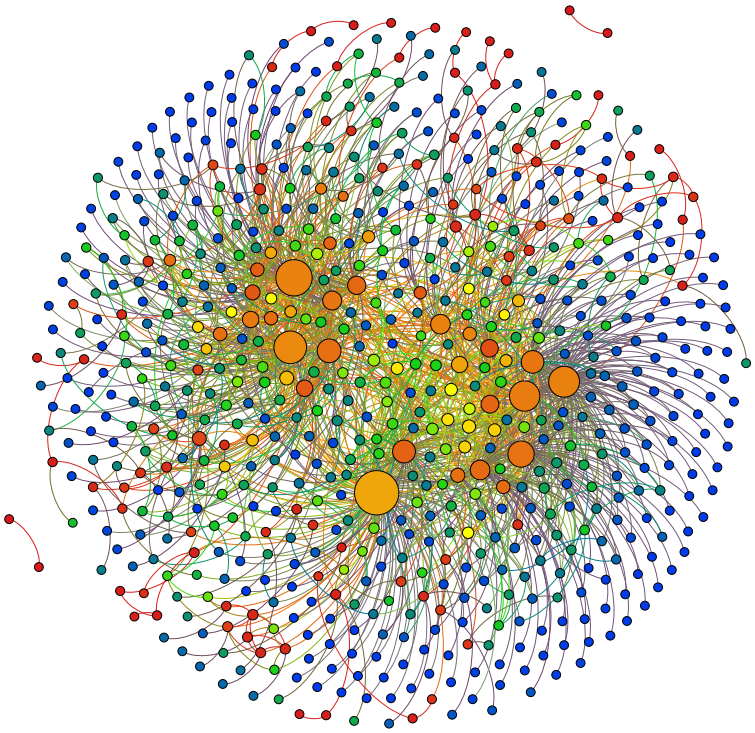


Figure 27: The pruned Internet graph of layer $l = 5$. The sizes of the nodes describe their relative degrees, while the colours of the nodes describe their optimal success probabilities, with red being close to zero and blue being close to one.

roughly the same length and to have as few crossing edges as possible. Naturally, with this kind of a mapping, high-degree nodes tend to be put in the center. A lot of differences can be seen with the renormalized and pruned $l = 5$ Internet networks. The renormalized $l = 5$ Internet has 13 971 links, while the pruned version only has 1 819, so a lot of the links have been removed in the rescaling. The pruned version resembles the BA graph a lot more in its structure, when these are compared to figure 22. The renormalized version also seems to be more heterogeneous in its structure when looking at the graph mapping, there seem to be separate larger-scale structures in the renormalized version. The pruned version on the other hand has two disconnected components from the giant component, where the nodes have really low optimal success probabilities, since the state of the walker cannot fully rotate from the giant component. The other optimal success probabilities of the CTQW in these two networks differ also, similar nodes have similar success probabilities for the renormalized version and the probabilities are much higher, and for the pruned version nodes with similar degrees can have vastly different success probabilities.

Figure 28 gives the optimal γ of the $l = 5$ renormalized and pruned Internet networks in a logarithmic scale. The renormalized version forms a clean line with the data points, even the γ_{opt} for nodes of degree $k = 1$ are all quite accurately the predicted $\gamma_{\text{opt}} = 1$. The pruned version has more variance for γ_{opt} for low degrees, which could also be predicted from the differences in optimal success probabilities for low-degree nodes.

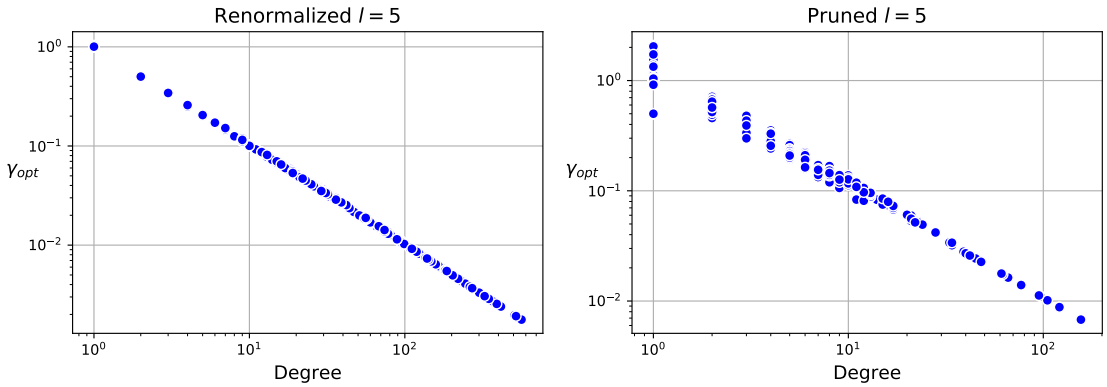


Figure 28: The optimal γ of all nodes with respect to the degrees in both $l = 5$ Internet networks in a logarithmic scale.

Figure 29 gives the optimal success probabilities of nodes for the $l = 5$ renormalized and pruned Internet networks. The same type of behaviour can be seen for

both networks, first the success probabilities go down and there is more variance for nodes of similar degrees, and then there is a linear part for high-degree nodes where the success probabilities get higher. The linear part also has a linear function fitted to the data points. The slopes of the functions for the renormalized and pruned versions are similar, which will be seen for other layers as well.

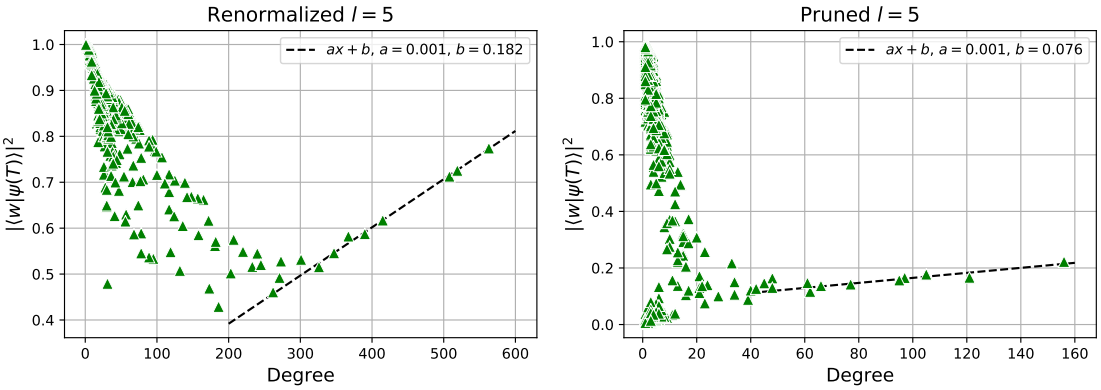


Figure 29: The optimal success probabilities of all nodes with respect to the degrees in both $l = 5$ Internet networks. A linear function has been fitted to the data points corresponding to high-degree nodes.

10.2 Renormalized and pruned layers

The different layers of the renormalized and pruned Internet networks can be studied together in order to see how the behaviour changes.

Figure 30 gives the optimal γ of all nodes for the renormalized and pruned networks with respect to the degree. It's hard to see smaller networks' data points from this plot since even if the data points are made more transparent, there is still so many data points in a small area. Comparing this to figure 28, it can be seen that on larger networks the largest degrees get bigger, and the optimal γ remain linear on a loglog scale. There is also more variance for optimal γ in low-degree nodes in both cases, especially for the pruned versions. For the pruned versions there is also more deviance form the $\gamma_{\text{opt}} = 1/k$ line specifically for $l = 2$ layer on lower-degree nodes. This can be caused by lower success probabilities for these nodes where there is not a clear change in the overlaps.

Figure 31 gives the optimal success probabilities with respect to degrees for renormalized and pruned Internet networks. The success probabilities of pruned layers vary a lot for low-degree nodes for all layers, while for renormalized versions

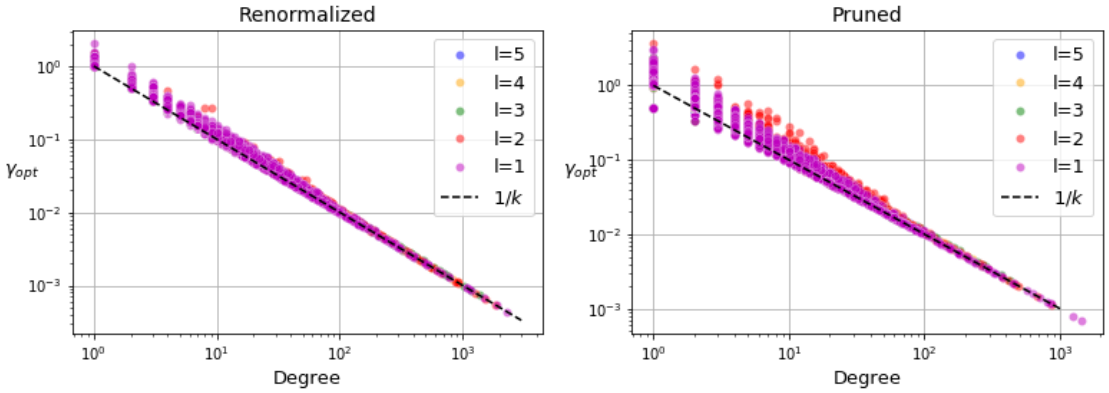


Figure 30: The optimal γ of all nodes with respect to the degrees in both Internet networks of all layers in a logarithmic scale.

they are higher in $l = 5$ and go down from there. This is to be expected since in the pruned versions links are removed during each downscaling with a probability given in Eq. (4). For higher-degree nodes, there is a linear increase of the success probability with respect to the degrees. This is the same type of behaviour that could be seen for the BA graph with $N = 1024$ and $m = 2$ in figure 21. The linear fits of the linear parts for high-degree nodes have similar slopes for the same layer between the renormalized and pruned versions for all layers. This is interesting behaviour and something that can be studied further, but in this thesis it is not focused on.

10.3 Comparisons with ER and BA graphs

The different renormalized and pruned Internet network layers can be compared to similar mimic graphs generated with the ER or BA models. By this I mean that the graphs can be generated in a way that they have approximately the same amount of nodes and links as compared to the layers of Internet networks. The ER graphs can be generated to specifically have n nodes and m links. The BA graphs can be generated by taking n nodes and m links that the original Internet network has, and the linking number is chosen by $\lfloor m/n \rceil$.

Figures 32 and 33 give the success probabilities of renormalized and pruned Internet networks of layers 5-3, as well as the ER and BA graphs made to mimic these graphs. Only these layers are shown since the behaviour is the same for $l = 2$ and $l = 1$ as well, and the plot would just be harder to interpret. The plots have been zoomed in to the parts where the mimic graphs' data points are. The

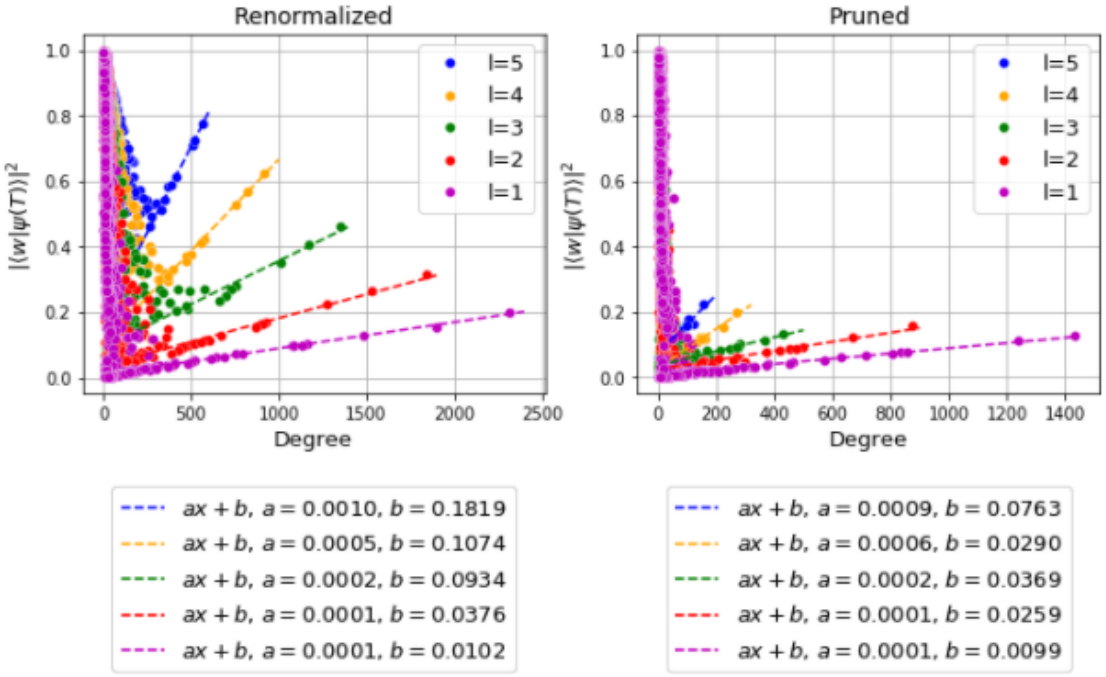


Figure 31: The optimal success probabilities of all nodes with respect to the degrees in both Internet networks of all layers. The linear fit parameters of the linear parts are also given.

renormalized mimic graphs do not overlap much with their corresponding Internet network data points, with the exception of $l = 3$ overlaying a little bit with the highest success probabilities of given degrees. Of course with low-degree nodes the probabilities get closer to one for all graphs. The pruned mimic graphs' data points overlap a lot better with the corresponding Internet network data points, and in fact different layers of the ER and BA mimic graphs overlap as well, there is just more variance for larger graphs for similar degrees. Also, the mimic graphs have similar separations for success probabilities of low-degree nodes as the Internet networks' success probabilities, where some nodes have really low success probabilities and others do not. What is really interesting is that the pruned BA mimic graphs have similar linear parts for the success probabilities of nodes with higher degrees.

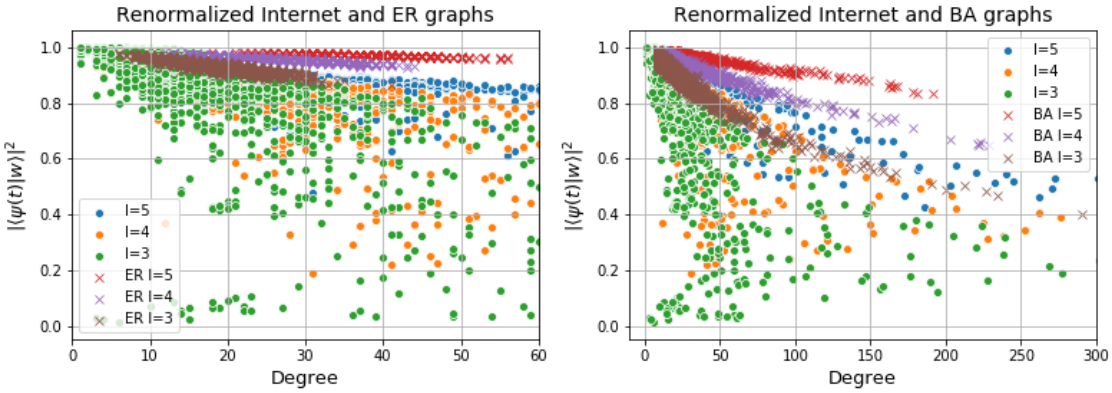


Figure 32: The optimal success probabilities of all nodes with respect to the degrees in both Internet networks of layers 5-3, as well as mimic ER and BA graphs. The graph has been zoomed in to the section where the ER and BA graphs' data points are.

Figure 34 gives the average optimal success probabilities, times and search times of the renormalized and pruned Internet networks of all layers as well as the corresponding ER and BA mimic graphs. The plots for average times and search times have also been plotted in logarithmic scales to see more clearly how they follow the $\sqrt{N}$ curve with respect to N. The success probabilities stay pretty much the same for the different layers of renormalized Internet networks, while for the pruned versions they get larger for larger networks. A spike on layer $l = 3$ can be seen for all averages on the pruned version that is most likely caused by what specific links are deleted with the pruning probability in these transitions between different pruned layers. The average times of both versions follow the $\sqrt{N}$ distribution quite

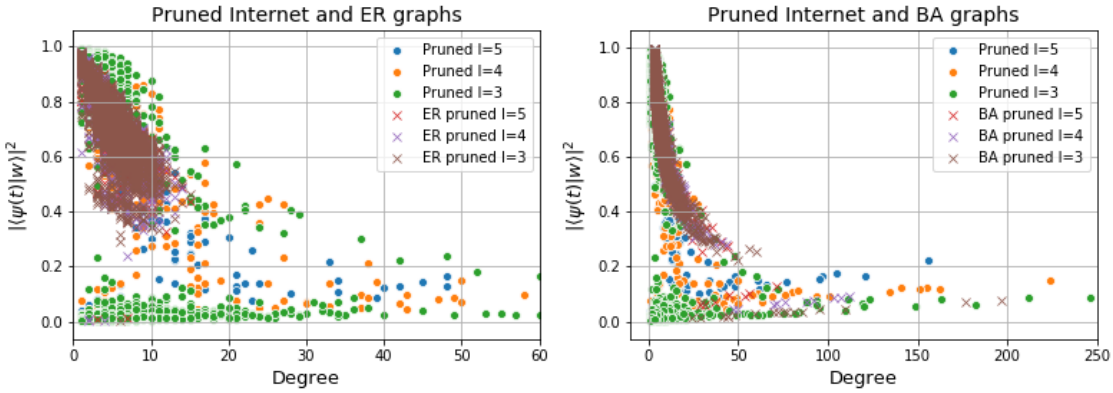


Figure 33: The optimal γ of all nodes with respect to the degrees in both Internet networks of layers 5-3, as well as mimic ER and BA graphs. The graph has been zoomed in to the section where the ER and BA graphs' data points are.

accurately, as well as the renormalized search times. The pruned search times do not change much apart from the spike in $l = 3$. The average search times of the full Internet network is approximately $\overline{T}_w \approx 670.86$, so it does not change much for that either on the pruned version, and for the renormalized version it gets larger and follows the $\sqrt{N}$ distribution. The average search times staying the same for the pruned versions makes sense since the pruning probability removes links when the network is renormalized to a smaller layer, and having less connections between nodes causes the search time to be larger in general. The average success probability and time of the full Internet network are $\overline{|\langle w|\psi(T)\rangle|^2} \approx 0.861$ and $\overline{T} \approx 244.65$, so both versions follow the same patterns for these values to the full network as well.

The ER and BA graphs also behave in a similar manner to the corresponding Internet networks for the renormalized average search times and for both average times. The average times make perfect sense as they follow the order of $\mathcal{O}(\sqrt{N})$ in these high-dimensional graphs. The average search times of the renormalized versions are similar to the mimic graphs, even though the success probabilities change differently for the mimic graphs. However, the probabilities are still quite high throughout, so they do not cause such a high increase in the search times. The average times and search times of the pruned mimic graphs behave similarly to renormalized mimic graphs as the times still follow $\mathcal{O}(\sqrt{N})$ and the success probabilities do not change much.

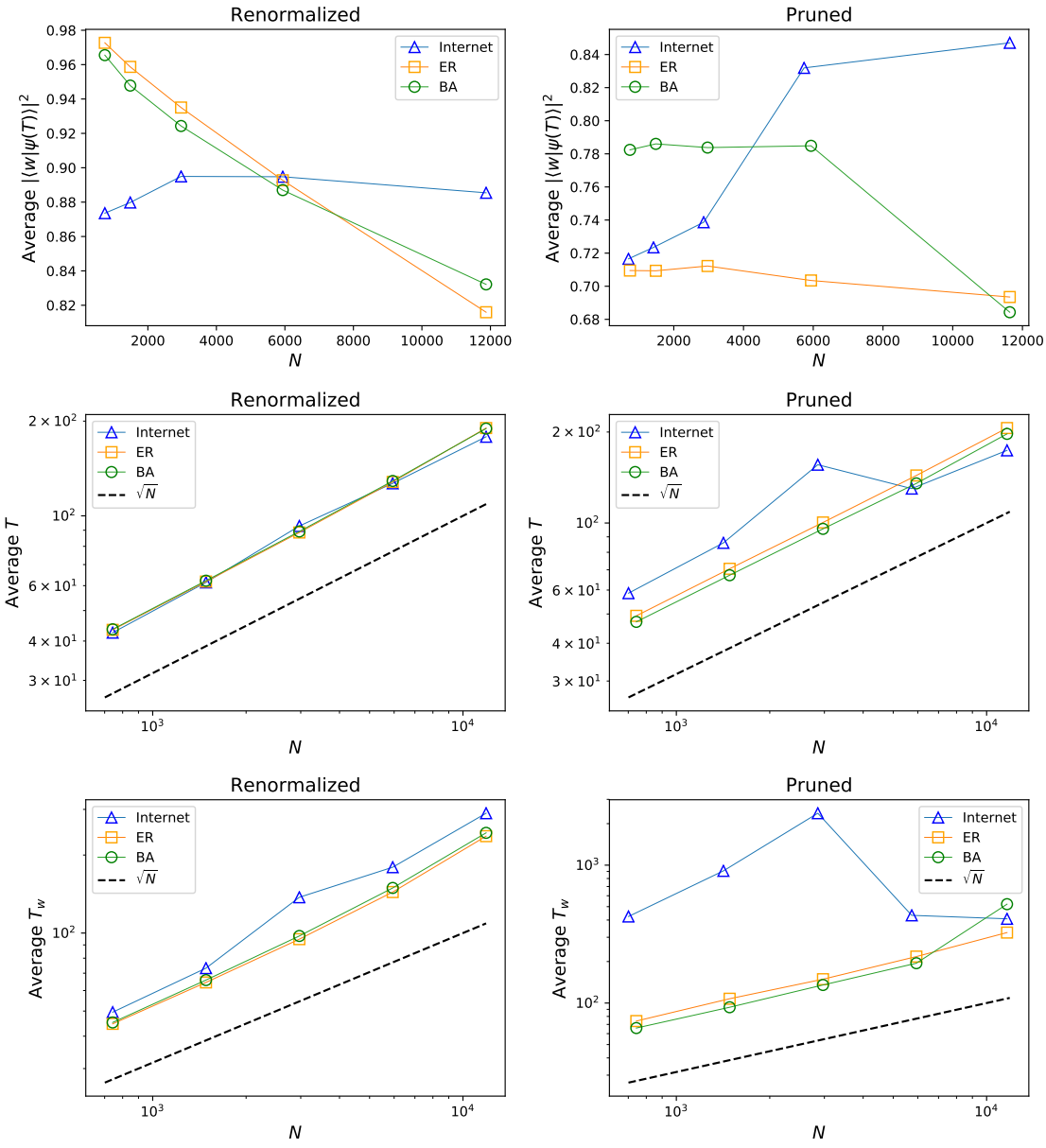


Figure 34: The average optimal success probabilities, times and search times with respect to the sizes of the renormalized and pruned Internet networks, as well as mimic ER and BA graphs.

10.4 Other values and properties

Different values of the networks on top of the previous ones were looked at that did not provide substantial results or offer any additional information. Eigenvector centralities, weighted degrees and local clustering were also studied. Eigenvector centralities and weighted degrees were plotted with these same values to see new information. Weighted degree means that the value of the degree of a node is divided with the average degree of that node's neighbours. Both values plotted with optimal success probabilities formed similar plots with high variance in low-degree nodes and linear parts for high-degree nodes, with minor differences. The optimal γ were not as linear with these values, with a lot more scattering for low-degree nodes. The plots of data points for these values and the degree with respect to times and search times were also looked at. The plots were similar to the success probabilities, with the linear parts mostly following the same line for all layers and typically the slope being neutral and the line close to zero. Local clustering did not show any interesting behaviour. Some of these results can be seen in figures 35 and 36.

The mapping of nodes and supernodes between different layers of the Internet networks were also looked at, meaning what nodes of layer l relate to what supernode of layer $l + 1$. Not much structure could be found by looking at the mappings of individual nodes or averages of multiple nodes with similar degrees and their optimal γ, success probabilities and times. The degrees of different node/supernode mappings were also looked at, and even these were not quite consistent. Figure 37 gives an example of the node/supernode mappings of optimal success probabilities of the renormalized Internet network layers. Many more mappings of different types of nodes/supernodes were looked at and the behaviour was always erratic, but more research is needed in checking if this is always so.

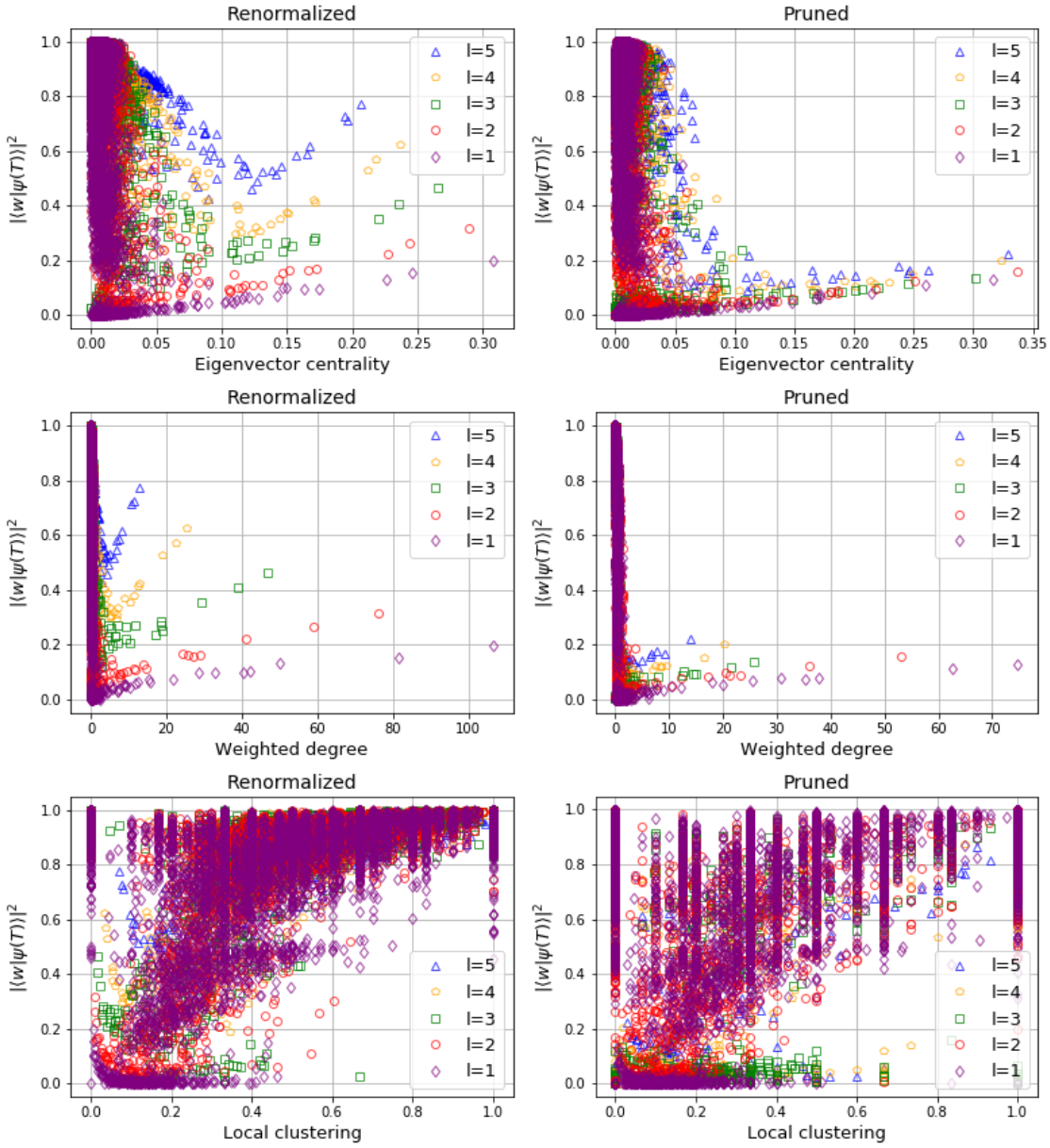


Figure 35: The optimal success probabilities of the renormalized and pruned Internet networks with respect to eigenvector centrality, weighted degree and local clustering.

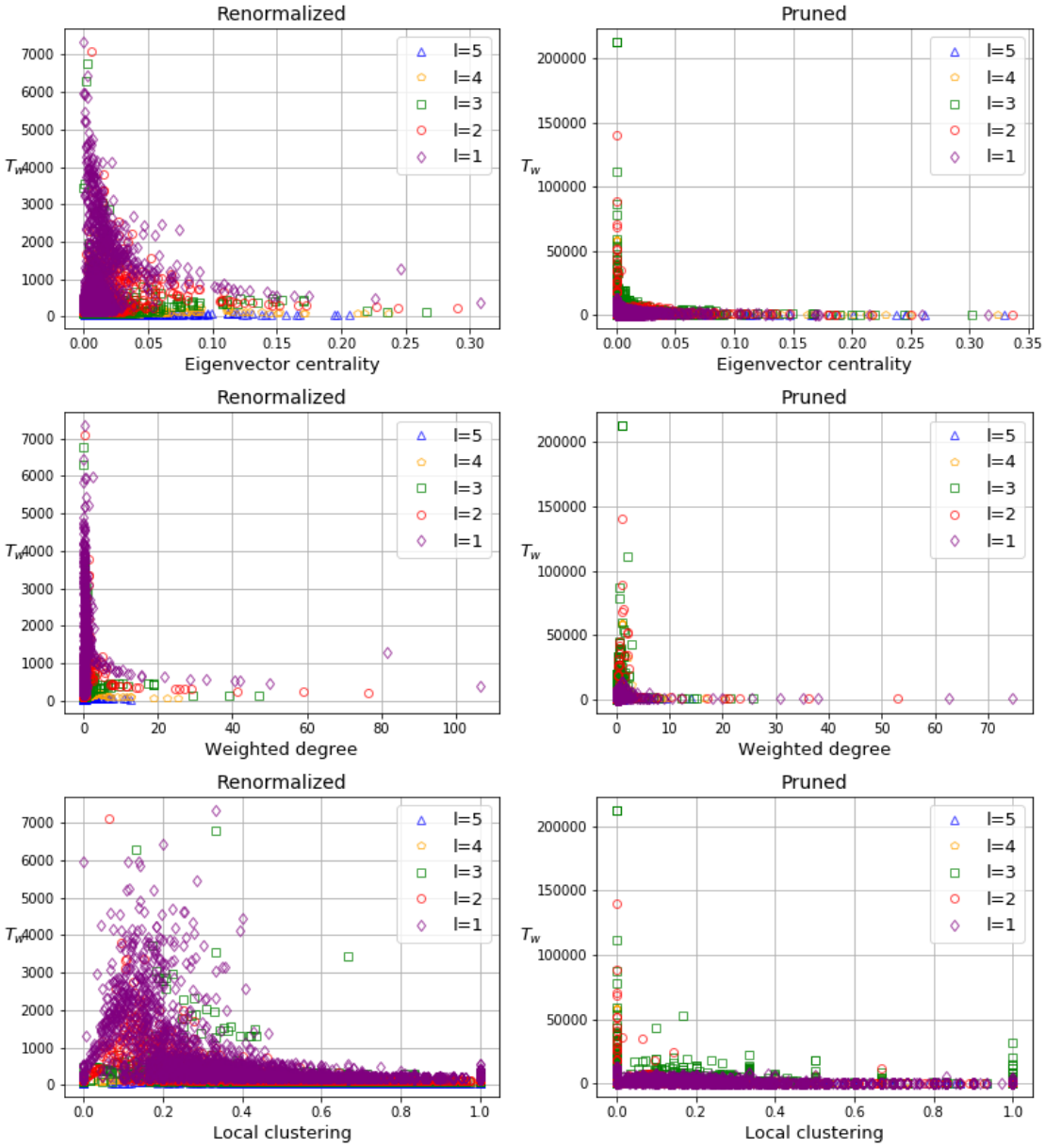


Figure 36: The search times of the renormalized and pruned Internet networks with respect to eigenvector centrality, weighted degree and local clustering.

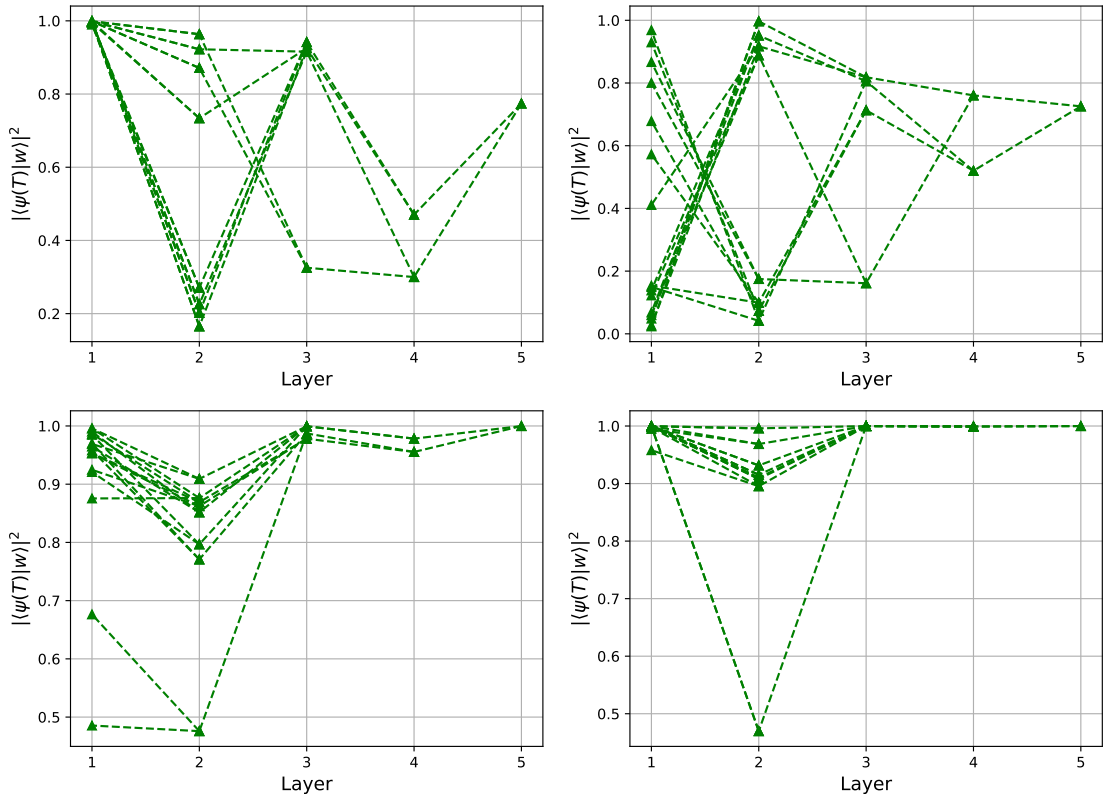


Figure 37: Examples of node/supernode mappings for the optimal success probabilities of renormalized Internet networks of different layers.

Conclusions

The results of this thesis have been quantitative in nature, since the subject is very difficult to study qualitatively as the behaviour of a CTQW on a complex network relies so much on the structure of the network. The field is also very new, so there is not much background information that goes into the specifics of CTQW on complex networks. Therefore the goal was to simply see how the behaviour of the spatial search by CTQW changes depending on the structure of the network, and especially in the case of the complex networks only numerical methods could be used for the analysis.

The goal of this thesis was to study the efficiency of spatial search by CTQW on complex networks, focusing on how the properties of a network affects the search. For this thesis two types of complex networks were considered: the Barabási-Albert graph and different versions of the renormalized and pruned Internet network. The results showed that the density of the network had the most significant effect on the optimality of the search, denser graphs tend to have better success probabilities and search times associated with them. In the future this could be studied further with other complex networks, as the variety in different network types was not the goal for this thesis.

The results do show that the behaviour of a CTQW depends on density, size and larger-scale structure of the network. Also, similar behaviour could be observed on the synthetic BA graphs and the Internet networks, especially for the pruned versions that have fewer links. In these cases the importance of hubs in the network becomes more significant, and it could also be seen that the success probability of a node on a high-degree node is typically lower than on an outlier node. This can also be a point of interest in future research on this subject, how the spectrum of the Hamiltonian looks for low- and high-degree nodes. The relation of the optimal coupling constant γ is very well established to be c/k_w for the target node w, where $c \approx 1$. However, the relation between the optimal success probability and the properties of the node as well as the topology of the network is not well known so

it requires further research.

The renormalization technique introduced by García-Pérez, Boguñá and Serrano has been used presumably for the first time in the context of spatial search by CTQW [6]. This can be a very useful tool in the future, since by reducing the size of the complex network the study of the behaviour of CTQW on the network becomes significantly easier. If it can be shown that the behaviour remains largely the same in renormalized versions, then the spectrum of the Hamiltonian can be more readily analysed in the smaller network, and the results compared to those of a larger network. The renormalized versions of the Internet network especially scale very nicely with respect to the different layers of the network for the average search time T_w. The results from the pruned versions are less beneficial, as the behaviour is more erratic, however the reason for the erratic behaviour can also be looked at more deeply.

What has been done in this thesis can in the future be used for more observations and proper analysis of spatial search by CTQW on complex networks, especially real-life networks. So far most studies of quantum walks on complex networks have focused on synthetic networks [20–22]. For future research, CTQW on real-world networks, such as the Internet network, need to be studied as well to gain more understanding on how quantum walks function on real-world complex structures. The continuous development of quantum computers also brings more importance to this field. More research is needed to see how the structure of a complex network affects the behaviour of a CTQW performing a spatial search on the network, and how well this behaviour could be predicted beforehand from the network itself. In the meantime, spatial search by quantum walk continues to be an active field of research, and currently it appears only to become more relevant [16–22].

Bibliography

[1] M. Newman, *Networks: An Introduction*, Oxford University Press (2010)

[2] A. M. Childs, J. Goldstone, *Spatial search by quantum walk*, Phys. Rev. A **70**, 022314 (2004)

[3] L. Grover, *Quantum Mechanics helps in searching for a needle in a haystack*, Phys. Rev. Lett. **79**, 325-328 (1997)

[4] D. Reitzner, D. Nagaj, V. Buzek, *Quantum walks*, acta physica slovaca **61**, 603-725 (2011)

[5] J. Kempe, *Quantum random walks: An introductory overview*, Contemporary Physics **44**, 307-327 (2003)

[6] G. García-Pérez, M. Boguñá, M. Á. Serrano, *Multiscale of real networks by geometric renormalization*, Nature Phys. **14**, 583-589 (2018)

[7] M. Á. Serrano, D. Krioukov, M. Boguñá, *Self-similarity of complex networks and hidden metric spaces*, Phys. Rev. Lett. **100**, 078701 (2008)

[8] Y. Aharonov, L. Davidovich, N. Zagury, *Quantum random walks*, Phys. rev. A **48**, 1687-1690 (1993)

[9] E. Farhi, S. Gutmann, *Quantum computation and decision trees*, Phys. rev. A **58**, 915-928 (1998)

[10] E. Farhi, S. Gutmann, *An Analog Analogue of a Digital Quantum Computation*, Phys. Rev. A **57**, 2403-2406 (1998)

[11] O. Muelken, A. Blumen, *Continuous-time quantum walks: Models for coherent transport on complex networks*, Phys. Rep. **502**, 37-87 (2011)

[12] M. Mohseni, P. Rebentrost, S. Lloyd, A. Aspuru-Guzik, *Environment-assisted quantum walks in photosynthetic energy transfer* J. Chem. Phys. **129**, 174106 (2008)

[13] S. Karlin, H. Taylor, *A First Course in Stochastic Processes*, Academic Press (2012)

[14] T. Heinosaari, M. Ziman, *The Mathematical Language of Quantum Theory*, Cambridge University Press (2012)

[15] T. Wong, L. Tarrataca, N. Nahimov, *Laplacian versus Adjacency Matrix in Quantum Walk Search*, Quant. Inf. Proc. **15**, 4029-4048 (2016)

[16] T. Wong, *Spatial search by continuous-time quantum walk with multiple marked vertices*, Quant. Inf. Proc. **15**, 1411-1443 (2016)

[17] S. Chakraborty, L. Novo, A. Ambainis, Y. Omar, *Spatial search by quantum walk is optimal for almost all graphs*, Phys. Rev. Lett. **116**, 100501 (2016)

[18] J. Morley, N. Chancellor, S. Bose, V. Kendon, *Quantum search with hybrid adiabatic–quantum walk algorithms and realistic noise*, Phys. Rev. A **99**, 022339 (2019)

[19] A. Glos, A. Krawiec, R. Kukulski, Z. Puchała, *Vertices cannot be hidden from quantum spatial search for almost all random graphs*, Quant. Inf. Proc. **17**, 81 (2018)

[20] T. Osada, B. Coutinho, Y. Omar, K. Sanaka, W. J. Munro, K. Nemoto, *Continuous-time quantum walk spatial search on the Bollobás scale-free network*, Phys. Rev. A **101**, 022310 (2020)

[21] M. Faccin, T. Johnson, J. Biamonte, S. Kais, P. Migdał, *Degree Distribution in Quantum Walks on Complex Networks*, Phys. Rev. X **3**, 041007 (2013)

[22] J. Biamonte, M. Faccin, M. de Domenico, *Complex Networks from Classical to Quantum*, Comm. Phys. **2**, 53 (2019)

[23] P. Erdős, A. Rényi, *On random graphs*, Pub. Math. **6**, 290-297 (1959)

[24] A.-L. Barabási, R. Albert, *Statistical mechanics of complex networks*, Rev. Mod. Phys. **74**, 47-97 (2002)

[25] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards **45**, 255-282 (1950)

[26] D. Calvetti, L. Reichel, D.C. Sorensen, *An Implicitly Restarted Lanczos Method for Large Symmetric Eigenvalue Problems* Electronic Transactions on Numerical Analysis **2**, 1-21 (1994)

[27] R. B. Lehoucq, D. C. Sorensen, C. Yang, *ARPACK Users Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM (1998)

[28] J. A. Nelder, R. Mead, *A simplex method for function minimization*, Computer Journal **7**, 308-313 (1965)

[29] R. Sidje, *Expokit: A Software Package for Computing Matrix Exponentials*, ACM Trans. Math. Softw. **24**, 130-156 (1998)

[30] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, D. Krioukov in CATCH, *http://www.caida.org/projects/ark/* (2009) visited 22.09.2020

[31] A. Childs, *Universal computation by quantum walk*, Phys. Rev. Lett. **102**, 180501 (2009)

[32] A. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, D. Spielman, *Exponential Algorithmic Speedup by a Quantum Walk*, Proceedings of the Thirty-Fifth ACM Symposium on Theory of Computing, 59-68, 2003

[33] Y. Wang, Z.-W. Cui, Y.-H. Lu, X.-M. Zhang, J. Gao, Y.-J. Chang, M.-H. Yung, X.-M. Jin, *Integrated Quantum-Walk Structure and NAND Tree on a Photonic Chip*, Phys. Rev. Lett. **125**, 160502 (2020)

[34] M. Cattaneo, M. Rossi, M. Paris, S. Maniscalco, *Quantum spatial search on graphs subject to dynamical noise*, Phys. Rev. A **98**, 052347 (2018)