Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Matematica - Curriculum Didattico

QUANTUM WALKS AND COMMUNITY DETECTION

Relatore:
Chiar.mo Prof.
DANIELE TANTARI

Co-Relatore:
Chiar.ma Prof.ssa
PAOLA BOITO

Presentata da:
FEDERICA PETRONGOLO

Sessione VI
Anno Accademico 2021/22

Contents

Introduction

0.1 Introduzione

Il quantum computing è una branca dell'informatica che si sta sviluppando rapidamente negli ultimi anni. A partire da studi e applicazioni di fisici come Richard Feynman e David Deutsch, matematici come Peter Shor e informatici come Lov Grover, si è capito che i principi della meccanica quantistica possono essere sfruttati nella teoria dell'informazione.
Gli algoritmi quantistici possono essere notevolmente più veloci di quelli classici, spesso il loro utilizzo all'interno di altri processi comporta un miglioramento esponenziale della complessità, sono pertanto adatti per i programmi più complessi.

Anche il processo stocastico noto come passeggiata aleatoria, o random walk, è stato riproposto in versione quantistica: il *quantum walk*. Per quanto esso sia direttamente ispirato al random walk, ci sono differenze sostanziali fra le due formulazioni.
Il quantum walk, infatti, consiste in un sistema quantistico, che quindi è regolato dai principi della meccanica quantistica. L'evoluzione del sistema è unitaria e l'intervento di interferenze tende a privilegiare alcuni cammini piuttosto che altri, perdendo così di simmetria.
Grazie al principio di sovrapposizione degli stati quantistici, inoltre, nel quantum walk è possibile percorrere tutti i possibili cammini simultaneamente, finché non viene effettuata una misurazione.
Osserviamo inoltre che, sotto ipotesi ragionevoli, il "*quantum hitting time*" del quantum walk, cioè il minimo numero di passi atteso per essere su un certo nodo, è quadraticamente più piccolo di quello del random walk, questo vuol dire che la diffusione nel quantum walk è quadraticamente più veloce.
Si può definire la passeggiata aleatoria sia a tempo continuo che a tempo discreto, nel lavoro corrente ci concentreremo su quest'ultima. Per descrivere una passeggiata quantistica è necessario adottare un opportuno formalismo, i più frequentemente usati per la versione a tempo discreto sono due: il "coined quantum walk" e il "Szegedy's quantum walk". La presentazione del quantum walk proposta in questo lavoro è basata principal-

mente sull'opera di Renato Portugal [18].

La passeggiata quantistica è utilizzata all'interno di algoritmi, ad esempio algoritmi di ricerca, tra cui l'algoritmo di Grover, e li rende più veloci rispetto alle loro controparti basate sulla passeggiata aleatoria classica. Uno degli ambiti in cui è sfruttata, ad esempio, è la ricerca di comunità nei grafi. Approfondiremo questo aspetto e useremo a tal proposito il formalismo del *coined quantum walk*.
Fissato un grafo, una comunità è un insieme di vertici meglio connessi tra loro che con gli altri vertici del grafo. Per sviluppare l'algoritmo di individuazione di comunità ci riferiremo in particolare all'articolo di Kanae Mukai e Naomichi Hatano [14].
Per prima cosa bisogna introdurre la "probabilità di transizione". Fissato uno stato iniziale, per ciascun nodo del grafo la probabilità di transizione è la probabilità di essere su quel nodo dopo un certo numero di passi. Si dimostra che, per ogni nodo di arrivo, la media rispetto al tempo delle probabilità di transizione converge a una distribuzione limite.

L'algoritmo che useremo consiste per prima cosa nell'ordinare i nodi per grado decrescente. Scegliamo come "centri" delle comunità quelli con grado maggiore. Per ogni nodo si calcola la distribuzione limite relativa al primo centro disponibile come stato iniziale. Se il valore trovato è più alto di una soglia predefinita, si inserisce il nodo in analisi nella comunità del centro corrispondente. Tale processo viene ripetuto al variare dei centri, finché tutti i nodi sono stati assegnati a una comunità. Testeremo quindi l'algoritmo descritto su diversi grafi per mostrarne il funzionamento.
Per l'implementazione useremo `MATLAB`, simulando l'algoritmo quantistico mediante metodi classici, infatti lo sviluppo dei computer quantistici è in fase preliminare e le macchine quantistiche disponibili, oltre ad essere di difficile accesso, hanno capacità limitata e pochissimi qubit.

Successivamente proveremo ad apportare alcune modifiche all'algoritmo e vedremo come queste influiscono sui risultati ottenuti. Analizzeremo, ad esempio, quale sia la miglior scelta per la matrice utilizzata per descrivere l'evoluzione temporale della passeggiata quantistica, in particolare confrontando la matrice di Fourier e quella di Grover. Studieremo poi in quali casi sia conveniente cambiare il criterio utilizzato per selezionare i centri delle comunità, sostituendo la misura di centralità data dal grado dei nodi con quella data dall'esponenziale della matrice di adiacenza. Alla fine vedremo anche un confronto dell'algoritmo di ricerca di comunità basato sul quantum walk con alcuni dei metodi di clustering più conosciuti: il *k-means* e lo *spectral clustering*.

0.2 Introduction

Quantum computing is a new frontier of computer science. Application of the principles of quantum mechanics to information theory began thanks to the theories and the algorithms developed by, among others, the physicists Richard Feynman and David Deutsch, the matematician Peter Shor and the computer scientist Lov Grover. Many researchers then get interested in this new perspective, discovering that quantum algorithms can be considerably faster than classical ones. In some cases, such as, for instance, Shor's algorithm, it is possible to reach exponential speed up. For this reason, quantum algorithms are expected to be especially suitable to large-scale problems.

Random walks are among the classical notions that have been revisited in a quantum framework, yielding the so called "quantum walks". We still have a walk, but with significant differences. A quantum walk takes place in a quantum system, so it obeys to the principles of quantum mechanics.
The evolution of the system is unitary and there are interference phenomena that tend to privilege some paths over others.
Thanks to the superposition of the states, the quantum walk can take all the possible paths simultaneously, until we perform a measurement.
Moreover, under reasonable hypotheses, the "quantum hitting time", that is the minimum expected number of steps to be on a certain node, of the quantum walk is quadratically smaller than the one of the random walk, this means that in the quantum walk diffusion has a quadratic speedup.
We can define the quantum walk for continuous time and for discrete time, and we will focus on the latter in this work. In order to describe a quantum walk, an appropriate formalism must be adopted, the more common models for the discrete-time quantum walks are the "coined quantum walk" and "Szegedy's quantum walk". The presentation of the quantum walk in this work is based mostly on Renato Portugal's work [18].

The quantum walk can be used inside other algorithms, such as search algorithms, for instance the Grover algorithm, and can make them faster. An example is its use for the community detection on graphs. We will see more about this, choosing the model of the coined quantum walk.
Given a graph, a community is a set of nodes better connected each others than to the other ones. In order to implement the algorithm to detect communities we will base mainly on Kanae Mukai and Naomichi Hatano article [14].
Let us briefly describe the setup for this problem. Set an initial state, for each node of the graph the transition probability is the probability to be on that node after a certain number of steps. We can prove that, for each arrival node, the average with respect to the time of the transition probabilities will converge to a limiting distribution.

The algorithm starts in sorting the nodes by decreasing degree. Then we choose as "centers" for the communities the nodes with the highest degree. For each node we compute the limiting distribution related to the first available center as initial state. If this value is greater than a predetermined threshold, the node is labeled as a member of the community of the current center. This operation is iterated over several centers, until all the nodes have been labeled.
We test our algorithm on several networks, to study experimentally its behavior. For the implementation we use `MATLAB`, simulating the quantum algorithm by classical methods, indeed the development of quantum computer is in a preliminary phase and the way to make the quantum algorithms work concretely is still in progress.

Then we also explore some modifications to the original algorithm to see how they affect the results. For example we will analyze which is the best choice for the time evolution operator, comparing the Fourier coin matrix with the Grover matrix, or which is the most convenient criterion to select the centers of the communities, substituting the measure of centrality of the degree of the nodes with the exponential of the adjacency matrix. At the end we will also see a comparison of our community detection algorithm, based on the quantum walks, with some of the most common clustering methods: the k-means and the spectral clustering.

Chapter 1

Quantum Walks

1.1 Quantum Computing

Quantum Computing is a new model of computation, whose operations can harness the phenomena of quantum mechanics. The unit of information is no longer the classical bit, but the "**qubit**", a quantum bit. The use and the behaviour of the qubits are based on the principles of quantum mechanics, so we introduce the four postulates on which quantum mechanics is based [17, 23, 11, 10, 19].

Consider a quantum system.

Postulate 1 (States)**.** *The set of all quantum states of a quantum system is a Hilbert space $\mathcal{H}$, called space of the states. A state of a quantum mechanical system is described by a unit vector $\psi \in \mathcal{H}$ such that $\|\psi\| = 1$.*

For instance, a qubit is described by a finite Hilbert space of dimension equal to 2, that is $\mathbb{C}^2$. The vectors are usually written in the Dirac notation, or "bra-ket" notation, i.e. using the "ket" form $|\psi\rangle$, to denote a vector in $\mathcal{H}$, and the "bra" form $\langle\psi|$, to denote an element in the dual space, which can be identified with the transpose conjugate of the vector. A vector $|\psi\rangle$ can be expressed in terms of its components as:

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix}. \tag{1.1}$$

Given two states $|\psi\rangle$ and $|\phi\rangle$, their inner product is:

$$\langle\psi|\phi\rangle := <\psi, \phi> = \begin{pmatrix} \psi_1 & \psi_2 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}. \tag{1.2}$$

The state of a qubit can be physically represented by a quantum system, such as a photon in one among two paths or the spin of a particle. These two possible states are usually

chosen as the elements of the basis. Choose as basis for $\mathbb{C}^2$ the one composed by the vectors:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1.3}$$

A general state of a qubit is so written as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{1.4}$$

where α, $\beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.
This writing means that, when a measurement is taken, the state of the qubit would be $|0\rangle$ with probability $|\alpha|^2$ or $|1\rangle$ with probability $|\beta|^2$.
The state $|\psi\rangle$ is equivalent to $e^{i\phi}\,|\psi\rangle$, where $e^{i\phi}$ is a unit complex number, called "global phase factor". For example the state $\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$ is equivalent to $e^{i\phi}\frac{1}{\sqrt{2}}\,|0\rangle + e^{i\phi}\frac{1}{\sqrt{2}}\,|1\rangle$. On the other hand, "relative phase factors" of two orthogonal states in superposition, have physical significance: $\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$ is different from $\frac{1}{\sqrt{2}}\,|0\rangle + e^{i\phi}\frac{1}{\sqrt{2}}\,|1\rangle$. Quantum states can be described as equivalence classes of unit vectors: $\{e^{i\phi}\,|\psi\rangle\}$, as $e^{i\phi}$ changes, with $\phi \in [0, 2\pi]$.
A general state $|\psi\rangle$ of a qubit can be written in the form:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,, \tag{1.5}$$

that is a point on the surface of a unit sphere known as *Bloch sphere*.

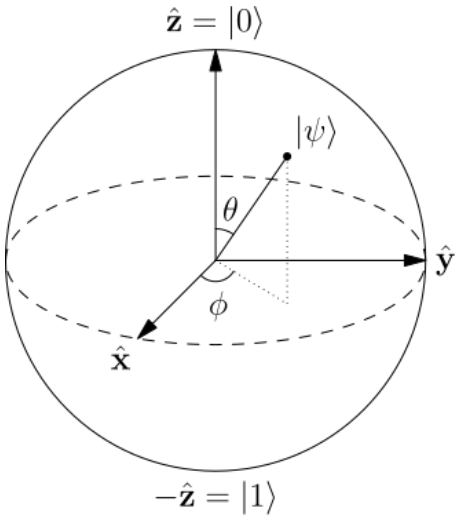


Figure 1.1: State $|\psi\rangle$ on the Bloch sphere

We can indeed see that:

$$\begin{aligned}|\psi\rangle &= \alpha\,|0\rangle + \beta\,|1\rangle = \\ &= |\alpha|e^{i\phi_\alpha}\,|0\rangle + |\beta|e^{i\phi_\beta}\,|1\rangle \equiv \\ &\equiv e^{-i\phi_\alpha}(|\alpha|e^{i\phi_\alpha}\,|0\rangle + |\beta|e^{i\phi_\beta}\,|1\rangle) = \\ &= |\alpha|\,|0\rangle + |\beta|e^{i\phi_\beta - i\phi_\alpha}\,|1\rangle = \\ &= |\alpha|\,|0\rangle + |\beta|e^{i\phi}\,|1\rangle\end{aligned}$$

so there are only three real parameters: $|\alpha|$, $|\beta|$, and $\phi = \phi_\beta - \phi_\alpha$, where $|\alpha|, |\beta| \in [0,1]$ and $\phi \in [0, 2\pi]$. Using the condition $|\alpha|^2 + |\beta|^2 = 1$, the parameters $|\alpha|$ and $|\beta|$ can be written as:

$$|\alpha| = \cos\rho, \quad |\beta| = \sin\rho \tag{1.6}$$

with $\rho \in [0, \frac{\pi}{2}]$. Setting $\theta = 2\rho$, we have:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\phi}\,|1\rangle$$

where $\theta \in [0, \pi]$. In this writing there are only two real parameters, θ and ϕ, that yield coordinates on the Bloch sphere.

The wave function that describes the state of a particle is the solution of the Schrodinger equation:

$$i\hbar\frac{\partial\,|\psi(t)\rangle}{\partial t} = -\frac{\hbar^2}{2m}\Delta\,|\psi(t)\rangle + V(x)\,|\psi(t)\rangle\,. \tag{1.7}$$

Defining the Hamiltonian operator as:

$$\widehat{H} := -\frac{\hbar^2}{2m}\Delta + V(x),$$

the equation can be written in this way:

$$i\hbar\frac{\partial\,|\psi(t)\rangle}{\partial t} = \hat{H}\,|\psi(t)\rangle\,. \tag{1.8}$$

Given an initial state $|\psi_0\rangle = |\psi(0)\rangle$ we would like to know how the system evolves over time.

Postulate 2 (Evolution)**.** *The evolution on time of a quantum system is described by the Schrodinger equation:*

$$i\hbar\frac{\partial\,|\psi(t)\rangle}{\partial t} = H(t)\,|\psi(t)\rangle\,. \tag{1.9}$$

This postulate talks about the evolution on continuous time but we are here interested on discrete steps of time, so we provide a simplified version:

Postulate 3 (Evolution on discrete time)**.** *The evolution on time of a state of a closed quantum system is described by a unit operator U such that, given an initial state $|\psi_1\rangle$, the state after a step of time is $|\psi_2\rangle = U|\psi_1\rangle$.*

Some of the unit operators more frequently used in quantum computing to describe the time evolution of systems are:

1. Pauli matrices:
   - $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$,
   - $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |1\rangle\langle 0| + |0\rangle\langle 1|$,
   - $\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = i|1\rangle\langle 0| - i|0\rangle\langle 1|$,
   - $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$.
2. Hadamard matrix: $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

The third postulate concerns the behaviour of systems composed by more than one qubit interacting each other.

Postulate 4 (Composite systems)**.** *Suppose we have two quantum systems and we want to define the associated composite system. The space of the states is the tensor product of the two spaces of the states: $\mathcal{H}_1 \otimes \mathcal{H}_2$.*
Suppose we know the state of the first system, $|\psi_1\rangle$, and the state of the second system, $|\psi_2\rangle$, than the state of the composite system is $|\psi_1\rangle \otimes |\psi_2\rangle$.

Take two qubits whose states are $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$, the state of the system composed by the two qubits is:

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta \end{pmatrix} = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle,$$

in the notation of the standard basis of $\mathbb{C}^2 \otimes \mathbb{C}^2 = \mathbb{C}^4$:

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix},\ |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix},\ |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix},\ |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

On the other hand, taking a state in a composite system, it can not always be written as product of states of the sub-systems. If it is possible the state is a "**separable state**", otherwise it is called "**entangled**".

An observable, that is a physically measurable quantity of a quantum system, is represented by a self-adjoint operator A on a Hilbert space $\mathcal{H}$. A preparation of a statistical ensemble is described by a state $|\psi\rangle \in \mathcal{H}$ called "pure state" if it is such that, for any observable A, the mean value of the observable can be calculated with the help of $|\psi\rangle$ as:

$$\langle A\rangle_\psi := \langle\psi|A\psi\rangle\,. \tag{1.10}$$

One calls $\langle A\rangle_\psi$ the quantum mechanical expectation value of the observable A in the pure state $|\psi\rangle$.
Let $\mathcal{H}$ be finite-dimensional with a given orthonormal basis, the condition for an operator A to be self-adjoint is equivalent to the condition that A is a Hermitian matrix, i.e. $A = A^*$. By the finite-dimensional spectral theorem, $\mathcal{H}$ has an orthonormal basis such that the matrix of A relative to this basis is a diagonal matrix with entries in the real numbers. Let $\{|\psi_k\rangle\}_k$ be the eigenbasis and $\{\lambda_k\}_k$ the respective eigenvalues of this diagonal representation, the operator A can be written as:

$$A = \sum_k \lambda_k\,|\psi_k\rangle\,\langle\psi_k|\,. \tag{1.11}$$

The expectation value of A is:

$$\begin{aligned}\langle A\rangle_\psi &= \langle\psi|A\psi\rangle = &(1.12)\\ &= \langle\psi|\sum_k \lambda_k\psi_k\rangle\,\langle\psi_k|\psi\rangle = &(1.13)\\ &= \sum_k \lambda_k\,\langle\psi|\psi_k\rangle\,\langle\psi_k|\psi\rangle = &(1.14)\\ &= \sum_k \lambda_k\,|\,\langle\psi|\psi_k\rangle\,|^2. &(1.15)\end{aligned}$$

Indeed, in a measurement process of an observable, the possible results are the eigenvalues of the associated self-adjoint operator. The values $|\,\langle\psi|\psi_k\rangle\,|^2$ are interpreted as the probabilities with which the respective eigenvalue λ_k is observed. For a measurement we can

only determine the probability that the result would be a certain value, but, despite this fact, taking the measurement has an effect on the system. Suppose that the result is the eigenvalue $\lambda_{\overline{k}}$, then the state of the system **collapses** to the corresponding eigenstate $\psi_{\overline{k}}$.

Postulate 5 (Measurement)**.** *In a quantum system the possible measurement values of an observable are given by the spectrum of the self-adjoint operator A associated with the observable. The probability $p_\psi(\lambda)$ that for a quantum system in the pure state $|\psi\rangle \in \mathcal{H}$ a measurement of the observable yields the eigenvalue λ of A is given with the help of the projection $\mathbf{P}_\lambda$ onto the eigenspace of λ as:*

$$p_\psi(\lambda) = \| \mathbf{P}_\lambda |\psi\rangle \|^2. \tag{1.16}$$

A common example of projective measure is the measurement on the computational basis $\{|0\rangle, |1\rangle\}$ of $\mathbb{C}^2$. The possible results are $|0\rangle$ and $|1\rangle$ and the measurement operators are the following ones:

$$\mathbf{P}_0 = |0\rangle\langle 0| = \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix},$$

$$\mathbf{P}_1 = |1\rangle\langle 1| = \begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}.$$

Given a qubit in a general state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the probabilities to have respectively $|0\rangle$ and $|1\rangle$ as results are:

$$p_\psi(|0\rangle) = \langle\psi|\mathbf{P}_0|\psi\rangle = \langle\psi|0\rangle\langle 0|\psi\rangle = \alpha^*\alpha = |\alpha|^2,$$

$$p_\psi(|1\rangle) = \langle\psi|\mathbf{P}_1|\psi\rangle = \langle\psi|1\rangle\langle 1|\psi\rangle = \beta^*\beta = |\beta|^2.$$

The state of the qubit after the measurement process depends on the result obtained, so the options are:

$$\frac{\mathbf{P}_0|\psi\rangle}{\sqrt{p_\psi(|0\rangle)}} = \frac{|0\rangle\langle 0|\psi\rangle}{|\alpha|} = \frac{\alpha}{|\alpha|}|0\rangle,$$

$$\frac{\mathbf{P}_1|\psi\rangle}{\sqrt{p_\psi(|1\rangle)}} = \frac{|1\rangle\langle 1|\psi\rangle}{|\beta|} = \frac{\beta}{|\beta|}|1\rangle.$$

Remember that $\alpha = |\alpha|e^{i\phi}$ and $\beta = |\beta|e^{i\gamma}$, whit $\phi, \gamma \in [0, 2\pi)$, so:

$$\frac{\alpha}{|\alpha|} = e^{i\phi}, \quad \frac{\beta}{|\beta|} = e^{i\gamma},$$

that are phase factors. Then the final states are equivalent respectively to $|0\rangle$ and $|1\rangle$.

An important characteristic of the quantum mechanics is the principle of **superposition**. Suppose that $|\psi_1\rangle$ and $|\psi_2\rangle$ are two possible states of a quantum system, then:

$$|\psi\rangle = \alpha\,|\psi_1\rangle + \beta\,|\psi_2\rangle \tag{1.17}$$

with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$, is still a possible state of the system. A quantum system has hence the property to be simultaneously in more than one state, we say that the system is in a superposition of states.
The density of probability associated to this new state is:

$$\begin{aligned} \|\psi\|^2 = \langle\psi|\psi\rangle = \\ &= \left(\overline{\alpha}\,|\psi_1\rangle + \overline{\beta}\,|\psi_2\rangle\right)\left(\alpha\,|\psi_1\rangle + \beta\,|\psi_2\rangle\right) = \\ &= \overline{\alpha}\alpha\,\langle\psi_1|\psi_1\rangle + \overline{\beta}\beta\,\langle\psi_2|\psi_2\rangle + \overline{\alpha}\beta\,\langle\psi_1|\psi_2\rangle + \overline{\beta}\alpha\,\langle\psi_2|\psi_1\rangle = \\ &= |\alpha|^2\|\psi_1\|^2 + |\beta|^2\|\psi_2\|^2 + 2\mathrm{Re}(\overline{\alpha}\beta)\,\langle\psi_1|\psi_2\rangle \end{aligned} \tag{1.18}$$

where $2\mathrm{Re}(\overline{\alpha}\beta)\,\langle\psi_1|\psi_2\rangle$ is the interference term. This result is an evidence for the fact that when two quantum states are added, we can not say that the density of probability increases, the states in superposition can interfer in a constructive or in a destructive way.

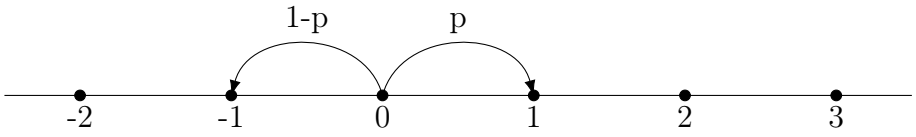
1.2 Classical Random Walks

Let $G = (V, E)$ be a graph, where V is the set of nodes and E is the set of edges, undirected, non-weighted. A discrete-time random walk on G can be visualized as a process where a walker starts from an initial node and at each discrete time step "jumps" from the current node to an adjacent one, according to a predetermined probability distribution [21].
Random walks are stochastic processes. Given a probability space $(\Omega, \mathcal{F}, P)$, where Ω is a sample space, $\mathcal{F}$ is a σ-algebra, and P is a probability measure, and a measurable space (S, Σ), that is a mathematical space S measurable with respect to a σ-algebra Σ, a **stochastic process** is a collection of S-valued random variables indexed by a set T:

$$\{X(t) : t \in T\}.$$

We can see through an elementary example how a random walk can be described as a stochastic process. Consider the one-dimensional random walk, this is the motion of a walker on the integer points of a line: $\ldots, -2, -1, 0, 1, 2, \ldots$ at discrete steps of time $t \in \{0, 1, 2, \ldots\}$. At each step the walker has two possibilities: to move one step right or

one step left. Set 0 as initial node, at any step, the walker has probability p of going to the right and $q = 1 - p$ of going to the left:



To define the walk as a stochastic process, we take independent random variables $X_1, X_2, \ldots$, indexed on discrete time steps. In this case they can be defined on a probability space $(\Omega, \mathcal{F}, P)$ where the sample space is $\Omega = \{\text{right}, \text{left}\}$, the σ-algebra is $\mathcal{F} = \{\{\text{right}\}, \{\text{left}\}\} \subseteq \mathcal{P}(\Omega)$ and the probability function P is defined as $P(\{\text{right}\}) = p$ and $P(\{\text{left}\}) = q$. For any $i = 1, 2, \ldots$, the random variable X_i takes value in $S = \{+1, -1\}$, in particular its value is $+1$ when the walker goes to the right and -1 when it goes to the left. This process can be seen as a Bernoulli process, taking $Y_i = \frac{X_i+1}{2}$, so Y_i is equal to 1 when $X_i = 1$ and to 0 when $X_i = -1$. The position of the walker at time i is described by $Z_i := X_1 + \cdots + X_i$, call its value h_i. Then the sum of the first i Bernoulli variables is:

$$S_i = Y_1 + \cdots + Y_i = \frac{X_1 + \cdots + X_i}{2} + \frac{i}{2} = \frac{h_i + i}{2}. \tag{1.19}$$

The probability for S_i to be equal to $\frac{h_i+i}{2}$ is:

$$P\left(S_i = \frac{h_i + i}{2}\right) = \binom{i}{\frac{h_i+i}{2}} p^{\frac{h_i+i}{2}} (1-p)^{i-\frac{h_i+i}{2}}.$$

Notice, furthermore, that the only positions h_i allowed at time i are the integers such that the following condition holds:

$$i - h_i \equiv 0 \bmod 2. \tag{1.20}$$

We can prove this claim by induction on i.

Proof. At time 0 it holds $h_0 = 0$ and $i - h_i = 0 - 0 \equiv 0 \bmod 2$.
Assuming $i - h_i \equiv 0 \bmod 2$, for $i + 1$ there are two different possibilities.
If $h_{i+1} = h_i + 1$, then:

$$i + 1 + h_{i+1} = i + 1 + h_i + 1 = i + h_i + 2 \equiv i + h_i \equiv 0 \bmod 2;$$

if $h_{i+1} = h_i - 1$, then:

$$i + 1 + h_{i+1} = i + 1 + h_i - 1 = i + h_i \equiv 0 \bmod 2.$$

□

After i steps, the probability for the walker to be on h_i is:

$$P(i,h_i) = \begin{cases} 0 & \text{if} \quad i - h_i \not\equiv 0 \bmod 2 \ \vee \ h_i > i \\ \binom{i}{\frac{h_i+i}{2}} p^{\frac{h_i+i}{2}} (1-p)^{i-\frac{h_i+i}{2}} & \text{if} \quad i - h_i \equiv 0 \bmod 2 \ \wedge \ h_i \leq i \end{cases} .$$

So, at the beginning, it is:

$$P(0,h_0) = \begin{cases} 1 & \text{if } h_0 = 0 \\ 0 & \text{otherwise} \end{cases} ,$$

indeed the walker is on the initial node. After the first step, the probability becomes:

$$P(1,h_1) = \begin{cases} p & \text{if } h_1 = 1 \\ 1-p & \text{if } h_1 = -1 \\ 0 & \text{otherwise} \end{cases} ,$$

indeed the walker can only move to node 1 or to node -1.

A random walk can also be seen as a Markov chain [18]. A **Markov chain** is a memory-less stochastic process: the probability of future actions is not dependent upon the steps that led up to the present state. So it is a sequence of random variables $X_1, X_2, \ldots$ assuming values in a discrete set that satisfy the Markov property, that is to say, for any positive integer n and any possible state $x_1, x_2, \ldots$ of the variables, the following equation holds:

$$\mathbb{P}(X_n = x_n | X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}). \tag{1.21}$$

Such a process can be seen as a directed graph $G = (V, E)$, the nodes $V = \{x_1, \ldots, x_n\}$ represent the states and the edges represent the transitions between states. The evolution on time can be discrete or continuous, but we are here interested into the discrete case. The discrete-time Markov chain, at each step, has an associated probability distribution. The latter can be described through the vector:

$$\begin{pmatrix} p_1(t) \\ \vdots \\ p_n(t) \end{pmatrix}, \tag{1.22}$$

where $p_i(t)$ is the probability for the variable X_i to be on the state x_i at time t. If the Markov chain describes a random walk, this probability is read as the probability of the

walker to be on the node x_i at time t. The probability distribution at time $t+1$ is completely determined by the probability distribution at time t alone:

$$p_i(t+1) = \sum_{j=1}^{n} M_{ij}\, p_j(t). \tag{1.23}$$

Here M is the transition matrix. The elements of M must be non-negative real numbers and their sum on any column must be equal to 1, i.e. M is a column stochastic matrix. When G is an undirected graph, its transition matrix is usually taken as M such that $M_{ij} = \frac{A_{ij}}{d_j}$, where A_{ij} are the elements of the adjacency matrix A and d_j is the degree of the node x_j. When the graph is directed an analogue choice is also possible, it consists in replacing the degree with the outdegree.

1.3 Coined Quantum Walks

Quantum walks are quantum analogues of the random walks. Depending on time evolution, they can be discrete or continuous, here we will talk about the discrete ones. The quantum walks can be described as the evolution of quantum systems, in order to do that we have to choose a formalism. The most common ones for a discrete quantum walk are the Coined Quantum Walk and Szegedy's Quantum Walk. Start talking about the first one [21, 18].
Let $G = (V, E)$ be an undirected graph where the walk is performed, here $|V| = N$, and suppose G is d-regular. A coined quantum walk is the composition of two quantum systems: a coin and a walker, so we need two Hilbert spaces. The space for the coin is $\mathcal{H}_C$, with basis $\{|k\rangle \mid k = 0, 1, \ldots, d\}$, and the space of the positions is $\mathcal{H}_P$, with basis $\{|n\rangle \mid n = 1, \ldots, N\}$. The space of the states is the tensor product $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$, with basis $\{|k\rangle \otimes |n\rangle\}$. A general state is a superposition of the basis states. The evolution of the system can be imagined as the motion of the walker on the nodes of G, along the edges, and its motion is determined by the state of the coin. At the beginning, the walker is on an initial node and a coin is in a certain state. At each step the coin is "tossed" and the walker shifts to one of the adjacent nodes. So we need a *coin operator* C, to toss the coin, and a *shift operator* S, for the motion of the walker. The evolution of the system is described by the operator $U = S \cdot (C \otimes \mathbb{I}_P)$, where $\mathbb{I}_P$ is the identity of $\mathcal{H}_P$. Applying U to a state, first of all the coin is tossed, then the walker shifts. The purpose of the coin operator is to render the coin state a superposition of states, and this influences the motion of the walker and contributes to the randomness of the walk. The previous example of the walk on a line can be replicated in this version. We take the regular graph $G = (\mathbb{Z}, E)$ of the integer points of a line as nodes and the segments between them as edges. All the nodes have degree equal to 2. A state of the system is given by the state of the coin and the position of the walker. The state of the coin

is a vector in $\mathcal{H}_C$, whose basis is $\{|0\rangle, |1\rangle\}$, like heads or tails. The position of the walker is a vector in $\mathcal{H}_P$ with basis $\{|n\rangle \mid n \in \mathbb{Z}\}$. So the basis of $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$ is $\{|k\rangle \otimes |n\rangle \mid k = 0, 1;\ n \in \mathbb{Z}\}$. From now on we will write the basis states as $|k, n\rangle$.
The coin operator is a matrix C of dimension 2×2. The shift operator S acts as follows on the basis states:

$$S|0, n\rangle = |0, n+1\rangle, \qquad S|1, n\rangle = |1, n-1\rangle. \tag{1.24}$$

Indeed S describes the motion of the walker, that from n moves of one step right to $|n+1\rangle$ or one step left to $|n-1\rangle$, for any $n \in \mathbb{Z}$. The equation for S can be derived from its action on the computational basis and it is:

$$S = |0\rangle\langle 0| \otimes \sum_{n=-\infty}^{\infty} |n+1\rangle\langle n| + |1\rangle\langle 1| \otimes \sum_{n=-\infty}^{\infty} |n-1\rangle\langle n|. \tag{1.25}$$

Let $|\psi(0)\rangle = |0, 0\rangle$ be the initial state of the quantum walk. The time-evolution operator is the unitary operator:

$$U = S \cdot (C \otimes \mathbb{I}_P) \tag{1.26}$$

and the state of the system after t steps is:

$$|\psi(t)\rangle = U^t |\psi(0)\rangle. \tag{1.27}$$

A common choice for the coin operator is the Hadamard gate [18]:

$$H = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Using H as coin operator, we can compute the first states of the system:

$$\begin{aligned}
|\psi(1)\rangle &= S \cdot (H \otimes \mathbb{I}_P)|\psi(0)\rangle = \\
&= S \cdot (H \otimes \mathbb{I}_P)|0\rangle|0\rangle = \\
&= \frac{1}{\sqrt{2}} \cdot S \cdot \left(\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} |0\rangle \otimes |0\rangle \right) = \\
&= \frac{1}{\sqrt{2}} \cdot S \cdot (|0\rangle|0\rangle + |1\rangle|0\rangle) = \\
&= \frac{1}{\sqrt{2}} \cdot (|0\rangle|1\rangle + |1\rangle|-1\rangle) = \\
&= \frac{1}{\sqrt{2}}|0\rangle|1\rangle + \frac{1}{\sqrt{2}}|1\rangle|-1\rangle,
\end{aligned}$$

$$
\begin{aligned}
|\psi(2)\rangle &= S \cdot (H \otimes \mathbb{I}_P) \, |\psi(1)\rangle = \\
&= S \cdot (H \otimes \mathbb{I}_P) \cdot \left(\frac{1}{\sqrt{2}} |0\rangle |1\rangle + \frac{1}{\sqrt{2}} |1\rangle |-1\rangle \right) = \\
&= \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} \cdot S \cdot \left(\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} |0\rangle \otimes |1\rangle + \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} |1\rangle \otimes |-1\rangle \right) = \\
&= \frac{1}{2} \cdot S \cdot (|0\rangle |1\rangle + |1\rangle |1\rangle + |0\rangle |-1\rangle - |1\rangle |-1\rangle) = \\
&= \frac{1}{2} \cdot (|0\rangle |2\rangle + |1\rangle |0\rangle + |0\rangle |0\rangle - |1\rangle |-2\rangle) = \\
&= \left(\frac{1}{2} |0\rangle + 0 \, |1\rangle \right) |2\rangle + \left(\frac{1}{2} |0\rangle + \frac{1}{2} |1\rangle \right) |0\rangle + \left(0 \, |0\rangle - \frac{1}{2} |1\rangle \right) |-2\rangle ,
\end{aligned}
$$

and so on. If we want to know the position of the walker at a certain step of time, we have to perform a measurement in the computational basis. Doing this after the first step will give as result $|0\rangle$, with probability $\left| \frac{1}{\sqrt{2}} \right|^2 = \frac{1}{2}$, or $|1\rangle$, with probability $\frac{1}{2}$. When the measurement is done, the state of the system collapses to $|0\rangle$ or $|1\rangle$. Repeating the procedure of applying the time evolution operator and performing the measurement in the computational basis, we obtain a classical random walk. The characteristic behaviour of a quantum system comes out if we do not perform intermediary measurements, because in this way constructive and destructive interferences are generated.

1.4 Szegedy's Quantum Walk

A different formalism for the quantum walk was introduced by Szegedy [18]. Consider a graph $G = (V, E)$, where $V = \{x_i | i \in \mathbb{N}\}$ is the set of the nodes. We can associate to G a bipartite graph B, such that the set of the nodes is the disjoint union of two subsets X and Y and $|X| = |Y|$. This graph is obtained by a duplication process: if $\{x_i, x_j\}$ is an edge of G, then $\{x_i, y_j\}$ and $\{x_j, y_i\}$ are edges of B. Consider a quantum walk on the bipartite graph B.

For any node $x \in X$ and any node $y \in Y$, define the values:

$$p_{xy} := \begin{cases} \frac{1}{\deg(x)} & \text{if } x \text{ is adjacent to } y \\ 0 & \text{otherwise} \end{cases}$$

$$q_{yx} := \begin{cases} \frac{1}{\deg(y)} & \text{if } y \text{ is adjacent to } x \\ 0 & \text{otherwise} \end{cases}$$

such that $\sum_{y \in Y} p_{xy} = 1$ for any x and $\sum_{x \in X} q_{yx} = 1$ for any y. The p_{xy} are the values of the probability that the walker goes from the node x to the node y, and the q_{yx} are the values of the probability that the walker goes from the node y to the node x. Define the matrix $P := (p_{xy})_{x,y}$, called "transition matrix".

The quantum walk on a bipartite graph has an associated Hilbert space $\mathcal{H}^{n^2} = \mathcal{H}^n \otimes \mathcal{H}^n$, where $n = |X| = |Y|$. The computational basis of the first factor is $\{|x\rangle \,|\, x \in X\}$, and the one of the second factor is $\{|y\rangle \,|\, y \in Y\}$, so the basis of $\mathcal{H}^{n^2}$ is $\{|x, y\rangle \,|\, x \in X, y \in Y\}$. Define the operators:

$$\begin{aligned} A : \mathcal{H}^n &\longrightarrow \mathcal{H}^{n^2} \\ |x\rangle &\mapsto A\,|x\rangle = |\alpha_x\rangle \end{aligned} \qquad A = \sum_{x \in X} |\alpha_x\rangle \langle x| \qquad |\alpha_x\rangle = |x\rangle \otimes \left(\sum_{y \in Y} \sqrt{p_{xy}}\, |y\rangle \right)$$

$$\begin{aligned} B : \mathcal{H}^n &\longrightarrow \mathcal{H}^{n^2} \\ |y\rangle &\mapsto B\,|y\rangle = |\beta_y\rangle \end{aligned} \qquad B = \sum_{y \in Y} |\beta_y\rangle \langle y| \qquad |\beta_y\rangle = \left(\sum_{x \in X} \sqrt{q_{yx}}\, |x\rangle \right) \otimes |y\rangle$$

We can compute the products between the columns:

$$\langle \alpha_x | \alpha'_x \rangle = \left(|x\rangle \otimes \sum_{y \in Y} \sqrt{p_{xy}}\, |y\rangle \,, |x'\rangle \otimes \sum_{y' \in Y} \sqrt{p_{x'y'}}\, |y'\rangle \right) = \langle x | x' \rangle \left(\sum_{y \in Y} \sqrt{p_{xy}}\, |y\rangle \,, \sum_{y' \in Y} \sqrt{p_{x'y'}}\, |y'\rangle \right), \tag{1.28}$$

if $x \neq x'$, their inner product is null, so we are interested on the value of the right factor only in the case that $x = x'$. Assuming this condition the right factor becomes:

$$\begin{aligned} \left(\sum_{y \in Y} \sqrt{p_{xy}}\, |y\rangle \,, \sum_{y' \in Y} \sqrt{p_{x'y'}}\, |y'\rangle \right) &= \sum_{y \in Y} \sum_{y' \in Y} \sqrt{p_{xy} p_{xy'}}\, \langle y | y' \rangle = \\ &= \sum_{y \in Y} p_{xy}\, \langle y | y \rangle = \\ &= \sum_{y \in Y} p_{xy} = 1\,. \end{aligned}$$

Substituting this result in 1.28 we obtain that:

$$\langle \alpha_x | \alpha'_x \rangle = \delta_{xx'} .$$

Similarly we can get:

$$\langle \beta_y | \beta'_y \rangle = \delta_{yy'} .$$

Then we can state that $A^T A = \mathbb{I}_n$ and $B^T B = \mathbb{I}_n$. This implies that the action of A and B preserves the norm of vectors.
Define the projectors:

$$\Pi_A := AA^T = \sum_{x \in X} |\alpha_x\rangle \langle \alpha_x| , \qquad \Pi_B := BB^T = \sum_{y \in Y} |\beta_y\rangle \langle \beta_y| . \tag{1.29}$$

We indeed have:

$$(AA^T)^2 = AA^T AA^T = AA^T, \qquad (AA^T)^T = AA^T ;$$

$$(BB^T)^2 = BB^T BB^T = BB^T, \qquad (BB^T)^T = BB^T .$$

Define the reflection operators:

$$\mathcal{R}_A := 2\Pi_A - \mathbb{I}_{n^2}, \qquad \mathcal{R}_B := 2\Pi_B - \mathbb{I}_{n^2}.$$

The operator Π_A projects a vector in $\mathcal{H}^{n^2}$ on the subspace $\mathcal{A} = \text{span}\{|\alpha_x\rangle \,|\, x \in X\}$, while $\mathcal{R}_A$ reflects a vector in $\mathcal{H}^{n^2}$ through $\mathcal{A}$:

- if $|\psi\rangle \in \mathcal{A}$, then $\mathcal{R}_A |\psi\rangle = |\psi\rangle$,
- if $|\psi\rangle \in \mathcal{A}^\perp$ or $|\psi\rangle$ is in the kernel of $\mathcal{A}$, then $\mathcal{R}_A |\psi\rangle = -|\psi\rangle$.

A vector $|\phi\rangle \in \mathcal{H}^{n^2} = \mathcal{A} \oplus \mathcal{A}^\perp$ can be written as a linear combination $|\phi\rangle = |\phi_{\mathcal{A}}\rangle + |\phi_{\mathcal{A}^\perp}\rangle$, so:

$$\mathcal{R}_A |\phi\rangle = \mathcal{R}_A(|\phi_{\mathcal{A}}\rangle + |\phi_{\mathcal{A}^\perp}\rangle) = |\phi_{\mathcal{A}}\rangle - |\phi_{\mathcal{A}^\perp}\rangle .$$

Likewise, Π_B projects a vector on the subspace $\mathcal{B} = \text{span}\{|\beta_y\rangle \,|\, y \in Y\}$ and $\mathcal{R}_B$ reflects a vector through $\mathcal{B}$.
To know the relationship between $\mathcal{A}$ and $\mathcal{B}$ we compute the angles between the vectors of the basis. Define the inner product matrix:

$$C := A^T B$$

such that $C_{xy} = \langle \alpha_x | \beta_y \rangle$. Then

$$\begin{aligned}
\langle \alpha_x | \beta_y \rangle &= \left(|x\rangle \otimes \sum_{y' \in Y} \sqrt{p_{xy'}} \, |y'\rangle \, , \sum_{x' \in X} \sqrt{q_{yx'}} \, |x'\rangle \otimes |y\rangle \right) = \\
&= \left(|x\rangle \, , \sum_{x' \in X} \sqrt{q_{yx'}} \, |x'\rangle \right) \left(\sum_{y' \in Y} \sqrt{p_{xy'}} \, |y'\rangle \, , |y\rangle \right) = \\
&= \sum_{x' \in X} \sqrt{q_{yx'}} \, \langle x | x' \rangle \left(\sum_{y' \in Y} \sqrt{p_{xy'}} \, \langle y | y' \rangle \right)^* = \\
&= \sqrt{q_{yx}} \sqrt{p_{xy}} = \sqrt{p_{xy} q_{yx}}.
\end{aligned}$$

The evolution operator of the quantum walk is:

$$W := \mathcal{R}_B \mathcal{R}_A \, .$$

Let $|\psi(0)\rangle$ be the initial state of the system, then the state at time t is $|\psi(t)\rangle = W^t \, |\psi(0)\rangle$. To analyze this time evolution we need the spectral decomposition of W, that can be computed in term of the singular values and vectors of C. The singular value decomposition theorem states that there are unitary matrices U and V such that $C = UDV^\dagger$, where D is a diagonal matrix with non-negative real entries, called singular values, that are uniquely determined by C. Consider $C^\dagger C$, it is a positive semi-definite Hermitian matrix, so it admits a spectral decomposition. Writing $C^\dagger C$ in the basis of the eigenvectors, then $\sqrt{C^\dagger C}$ is the diagonal matrix of the square roots of the eigenvalues of $C^\dagger C$. Let $\{\lambda_i^2\}$ be the eigenvalues of $C^\dagger C$ and $\{|\nu_i\rangle \, | \, i = 1, \ldots, n\}$ the corresponding eigenvectors, which form an orthonormal basis. Then:

$$C^\dagger C = \sum_{i=1}^{n} \lambda_i^2 \, |\nu_i\rangle \, \langle \nu_i| \, , \qquad \sqrt{C^\dagger C} = \sum_{i=1}^{n} \lambda_i \, |\nu_i\rangle \, \langle \nu_i| \, .$$

If $\lambda_i > 0$, define $|\mu_i\rangle := \frac{1}{\lambda_i} C \, |\nu_i\rangle$. For these vectors we compute:

$$\begin{aligned}
\langle \mu_i | \mu_j \rangle &= \frac{1}{\lambda_i \lambda_j} \, \langle \nu_i | C^\dagger C | \nu_j \rangle = \\
&= \frac{1}{\lambda_i \lambda_j} \sum_{k=1}^{n} \lambda_k^2 \, \langle \nu_i | \nu_k \rangle \, \langle \nu_k | \nu_j \rangle = \\
&= \begin{cases} \frac{1}{\lambda_i \lambda_i} \lambda_i^2 = 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} = \delta_{ij} \, .
\end{aligned}$$

For the eigenvectors in the kernel of $\sqrt{C^\dagger C}$ define $|\mu'_j\rangle := |\nu_j\rangle$. We generally lose the orthogonality between vectors $|\mu_i\rangle$ and $|\mu'_j\rangle$, but we can apply the Gram-Schmidt orthonormalization process to redefine vectors $|\mu'_j\rangle$ such that they are orthogonal to the vectors that do not belong to the kernel, recall them $|\mu_j\rangle$. Now we obtain the matrices U and V using:

$$U = \sum_{i=1}^{n} |\mu_i\rangle \langle i| , \qquad V = \sum_{i=1}^{n} |\nu_i\rangle \langle i| ,$$

$|\mu_i\rangle$ and $|\nu_i\rangle$ are the singular vectors and λ_i the singular values. For $i = 1, \ldots, n$, they respect the equations:

$$C |\nu_i\rangle = \lambda_i |\nu_i\rangle , \qquad C^T |\mu_i\rangle = \lambda_i |\nu_i\rangle$$

So $|\mu_i\rangle$ is called "left singular vector" and $|\nu_i\rangle$ is called "right singular vector". Multiplying them by A and B, respectively, we have:

$$AC |\nu_i\rangle = A\lambda_i |\nu_i\rangle \qquad BC^T |\mu_i\rangle = B\lambda_i |\nu_i\rangle$$
$$\Pi_A B |\nu_i\rangle = \lambda_i A |\nu_i\rangle \qquad \Pi_B A |\mu_i\rangle = \lambda_i B |\nu_i\rangle .$$

Here $B |\nu_i\rangle$ and $A |\mu_i\rangle$ have norm equal to 1 and the projectors preserve or decrease the norm, so the λ_i must satisfy $\lambda \in [0, 1]$. Write them as $\lambda_i = \cos(\theta_i)$, where $\theta_i \in \left[0, \frac{\pi}{2}\right]$ is the angle between $A |\mu_i\rangle$ and $B |\nu_i\rangle$, indeed:

$$(A |\mu_i\rangle , B |\nu_i\rangle) = \langle \mu_i | A^T B | \nu_i \rangle = \langle \mu_i | C | \nu_i \rangle = \langle \mu_i | \lambda_i | \mu_i \rangle = \lambda_i.$$

Recall the evolution operator is $W = \mathcal{R}_B \mathcal{R}_A$. Its characteristic polynomial is:

$$p_W = \det(\lambda \mathbb{I}_N - W) = (\lambda - 1)^{N-2n} (\det(\lambda + 1)^2 \mathbb{I}_n - 4\lambda C^T C).$$

So, there are at least $N - 2n$ eigenvalues equal to 1 and the remaining $2n$ can be obtained from:

$$\det((\lambda + 1)^2 \mathbb{I}_n - 4\lambda C^T C) = 0$$

Using $\mathbb{I}_n = \sum_{j=1}^{n} |\nu_j\rangle \langle \nu_j|$ and $C^T C = \sum_{i=1}^{n} \lambda_i^2 |\nu_i\rangle \langle \nu_i|$ it becomes:

$$\det \left(\sum_{j=1}^{n} ((\lambda + 1)^2 - 4\lambda\lambda_j^2) |\nu_j\rangle \langle \nu_j| \right) = 0,$$

that implies:

$$\prod_{j=1}^{n} \lambda^2 - 2\lambda(2\lambda_j^2 - 1) + 1 = 0.$$

For each $j = 1, \ldots, n$, we obtain two eigenvalues: $\lambda = (2\lambda_j^2 - 1) \pm 2\lambda_j\sqrt{\lambda_j^2 - 1}$, and, using $\lambda_j = \cos(\theta_j)$, they can be written as $\lambda = e^{\pm 2i\theta_j}$.
The following statements are valid:

- if $|\psi\rangle \in \mathcal{A} \cap \mathcal{B} + \mathcal{A}^\perp \cap \mathcal{B}^\perp$, then $W|\psi\rangle = |\psi\rangle$;
- if $|\psi\rangle \in \mathcal{A} \cap \mathcal{B}^\perp + \mathcal{A}^\perp \cap \mathcal{B}$, then $W|\psi\rangle = -|\psi\rangle$;
- if $\dim(\mathcal{A} \cap \mathcal{B}) = k$, then $\dim(\mathcal{A}^\perp \cap \mathcal{B}^\perp) = N - 2n + k$.

The following theorem holds:

Theorem 1 (Szegedy)**.** *The spectrum of W obeys:*

1. *The eigenvalues of W with $\theta_j \in \left(0, \frac{\pi}{2}\right]$ are $e^{\pm 2i\theta_j}$, for $j = 1, \ldots, n-k$, where k is the multiplicity of singular value 1. The corresponding normalized eigenvectors are:*
$$|\theta_j^\pm\rangle = \frac{1}{\sqrt{2}\sin(\theta_j)}\left(A|\mu_j\rangle - e^{\pm i\theta_j}B|\nu_j\rangle\right).$$
2. *$\mathcal{A} \cap \mathcal{B} + \mathcal{A}^\perp \cap \mathcal{B}^\perp$ is the eigenspace related to 1 and $\mathcal{A} \cap \mathcal{B}$ is spanned by the $A|\mu_j\rangle$, for the $|\mu_j\rangle$ left singular vectors of C related to 1.*
3. *$\mathcal{A} \cap \mathcal{B}^\perp + \mathcal{A}^\perp \cap \mathcal{B}$ is the eigenspace related to -1, $\mathcal{A} \cap \mathcal{B}^\perp$ is spanned by the $A|\mu_j\rangle$, for the $|\mu_j\rangle$ left singular vectors of C related to 0, and $\mathcal{A}^\perp \cap \mathcal{B}$ is spanned by $B|\nu_j\rangle$, for the $|\nu_j\rangle$ right singular vectors of C related to 0.*

Note that the above theorem does not describe the $N - 2n + k$ eigenvectors related to the eigenvalue $+1$ that span $\mathcal{A}^\perp \cap \mathcal{B}^\perp$.

1.4.1 Quantum Hitting Time

Szegedy's quantum walk can be applied to search for marked vertices. Start marking some nodes of the graph G, let M be the set of these vertices. Consider the bipartite graph $\mathcal{B}$ obtained by the duplication process, whose vertices set is $X \cup Y$. For any $x \in X$ and any $y \in Y$, define:

$$p'_{xy} = \begin{cases} p_{xy}, & \text{if} \quad x \notin M \\ \delta_{xy}, & \text{if} \quad x \in M \end{cases}.$$

Consider the matrix $P' = \left(p'_{xy}\right)_{x \in X, y \in Y}$ and let its associated evolution operator W'. When we use the operator be W' on the graph $\mathcal{B}$, the probabilities associated with the

marked vertices increase periodically. To find a marked vertex, we must measure the position of the walker as soon as the probability of being in M is high. The **quantum hitting time** tells when we measure the walker's position. Consider as initial state of the quantum walk the following state:

$$|\psi(0)\rangle = \frac{1}{\sqrt{n}} \sum_{x\in X, y\in Y} \sqrt{p_{xy}}\,|x,y\rangle\,. \tag{1.30}$$

The quantum hitting time $H_{P',M}$ of a quantum walk on $\mathcal{B}$ with evolution operator W' and starting from the initial state $|\psi(0)\rangle$ is defined as the smallest number of steps T such that:

$$F(T) \geq 1 - \frac{m}{n}, \tag{1.31}$$

where m is the number of marked vertices, n is the number of vertices of the graph G and $F(T)$ is defined as:

$$F(T) = \frac{1}{T+1} \sum_{t=0}^{T} \| \,|\psi(t)\rangle - |\psi(0)\rangle\, \|^2, \tag{1.32}$$

where $|\psi(t)\rangle$ is the state at step t, i.e. $|\psi(t)\rangle = (W')^t\,|\psi(0)\rangle$.
The initial state can be written in the eigenbasis related to the singular values of C:

$$|\psi(0)\rangle = \sum_{j=1}^{n-k} \left(c_j^+\,|\theta_j^+\rangle + c_j^-\,|\theta_j^-\rangle\right) + \sum_{j=n-k+1}^{n^2-n+k} c_j\,|\theta_j\rangle\,,$$

where $|\theta_j^\pm\rangle$ are the first $2(n-k)$ eigenvectors related to $e^{\pm 2i|\theta_j\rangle}$ and $|\theta_j\rangle$ are the eigenvectors related to 1; the coefficients are given by $c_j^\pm = \langle\theta_j^\pm|\psi(0)\rangle$. It also satisfies the condition:

$$\sum_{j=1}^{n-k} \left(|c_j^+|^2 + |c_j^-|^2\right) + \sum_{j=n-k+1}^{n^2-n+k} |c_j|^2 = 1.$$

The state at step t can so be written as:

$$|\psi(t)\rangle = (W')^t\,|\psi(0)\rangle = \sum_{j=1}^{n-k} \left(c_j^+ e^{2i\theta_j t}\,|\theta_j^+\rangle + c_j^- e^{-2i\theta_j t}\,|\theta_j^-\rangle\right) + \sum_{j=n-k+1}^{n^2-n+k} c_j\,|\theta_j\rangle\,.$$

Computing $|\psi(t)\rangle - |\psi(0)\rangle$, the terms associated to the eigenvalue 1 are eliminated and, since $|c_j^+|^2 = |c_j^-|^2$, denote both with $|c_j|^2$. We obtain that:

$$\| \,|\psi(t)\rangle - |\psi(0)\rangle\, \|^2 = 4 \sum_{j=1}^{n-k} |c_j|^2 (1 - T_{2t}(\cos\theta_j))\,. \tag{1.33}$$

Here T_n is the n-th Chebyshev polynomial of the first kind, defined by the equation $T_n(\cos\theta) = \cos(n\theta)$. Now $F(T)$ can be explicitly calculated:

$$F(T) = \frac{2}{T+1} \sum_{j=1}^{n-k} |c_j|^2 (2T + 1 - U_{2T}(\cos\theta_j)). \tag{1.34}$$

Here U_n is the n-th Chebyshev polynomial of the second kind, defined by the equation $U_n(\cos\theta) = \frac{\sin((n+1)\theta)}{\sin\theta}$. Function $F(T)$ is continuous, select a range $[0, T]$ containing $1 - \frac{m}{n}$ where $F(T)$ can be inverted to obtain the quantum hitting time:

$$H_{P',M} = \lceil F^{-1}\left(1 - \frac{m}{n}\right) \rceil. \tag{1.35}$$

Removing the ceiling function from the above equation, we have a valid definition.
Now determining the running time of the algorithm is fundamental. The probability of finding a marked element has an oscillatory pattern, so if the measurement is delayed, the success of probability may be very low. The quantum hitting time must be close to the time $t_{\max}$ where the probability reaches the maximum for the first time. The state at step t is:

$$|\psi(t)\rangle = |\psi(0)\rangle + \sum_{j=1}^{n-k} \left(c_j^+ (e^{2i\theta_j t} - 1) \, |\theta_j^+\rangle + c_j^- (e^{-2i\theta_j t} - 1) \, |\theta_j^-\rangle \right).$$

The projector on the vector space spanned by the marked vertices is:

$$\mathcal{P}_M := \sum_{x \in M} |x\rangle \langle x| \otimes \mathbb{I} = \sum_{x \in M} \sum_{y} |x, y\rangle \langle x, y| \,. \tag{1.36}$$

The probability of finding a marked element at time t is:

$$\langle \psi(t) | \mathcal{P}_M | \psi(t) \rangle \,.$$

1.4.2 Complete Graphs

As an example we show the computation of the quantum hitting time when G is a complete graph. The number of nodes of G is n, so as computational basis of the Hilbert space $\mathcal{H}^n$ we take $\{|1\rangle, \ldots, |n\rangle\}$. The transition matrix is:

$$P = \frac{1}{n-1} \begin{pmatrix} 0 & 1 & \ldots & 1 \\ 1 & 0 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 0 \end{pmatrix}.$$

Observe that $(n-1)P$ is equal to the matrix with all entries equal to 1 minus the identity matrix. Define the vectors:

$$|u^{(j)}\rangle := \frac{1}{\sqrt{j}} \sum_{i=1}^{j} |i\rangle$$

for a certain value j. We can write the matrix P as:

$$P = \frac{1}{n-1} \left(n \, |u^{(n)}\rangle \langle u^{(n)}| - \mathbb{I}_n \right). \tag{1.37}$$

Choose as marked vertices the last m nodes. The elements of the matrix P' are:

$$p'_{xy} = \begin{cases} \frac{1-\delta_{xy}}{n-1}, & 1 \leq x \leq n-m \\ \delta_{xy}, & n-m < x \leq n \end{cases}.$$

For the quantum hitting time we use the evolution operator W', in order to find its eigenvectors we are interested on the singular values and vectors of the operator C. The entries of C are $c_{xy} = \sqrt{p'_{xy} q_{yx}}$. We here set $q_{yx} = p'_{yx}$. In a complete graph $p_{xy} = p_{yx}$. However, for x and y in M: $p'_{xy} \neq p'_{yx}$. We can write C as:

$$C = \begin{pmatrix} P_{\overline{M}} & 0 \\ 0 & \mathbb{I}_m \end{pmatrix},$$

where $P_{\overline{M}}$ is the matrix obtained from P by eliminating m rows an columns corresponding to the marked vertices. The matrix $P_{\overline{M}}$ can be written as:

$$P_{\overline{M}} = \frac{1}{n-1} \left((n-m) \, |u^{(n-m)}\rangle \langle u^{(n-m)}| - \mathbb{I}_{n-m} \right). \tag{1.38}$$

Its characteristic polynomial is:

$$\det(P_{\overline{M}} - \lambda \mathbb{I}) = \left(\lambda - \frac{n-m-1}{n-1} \right) \left(\lambda + \frac{1}{n-1} \right)^{n-m-1}.$$

Note that, if $m \geq 1$, then 1 is not an eigenvalue of $P_{\overline{M}}$. The eigenvalue $\frac{n-m-1}{n-1}$ has multiplicity 1 and the corresponding eigenvector is $|\nu_{n-m}\rangle = |u^{(n-m)}\rangle$. The eigenvalue $\frac{-1}{n-1}$ has multiplicity $n-m-1$ and the corresponding eigenvectors in this case are $|\nu - i\rangle = \frac{1}{\sqrt{i+1}}(|u^{(i)}\rangle - \sqrt{i}\,|i+1\rangle)$, for $i = 1, \ldots, n-m-1$. The set $\{|\nu_i\rangle, 1 \leq i \leq n-m\}$ is an orthonormal basis of eigenvectors of $P_{\overline{M}}$.
The matrix C is Hermitian, so the non-trivial singular values λ_i of C are given by the absolute value of the eigenvalues of $P_{\overline{M}}$. The right singular vectors are the eigenvectors of $P_{\overline{M}}$: $|\nu_i\rangle$, the left singular vectors are $|\mu_i\rangle = \frac{1}{\lambda_i} C \, |\nu_i\rangle$. If an eigenvalue of $P_{\overline{M}}$ is negative, then the left singular vector is the negative of the corresponding eigenvector

of $P_{\overline{M}}$. These vectors must be increased with m zeros, to fit the dimension of C. Then there is the singular value 1 with multiplicity m, given by $\mathbb{I}_m$, the corresponding singular vectors are $|j\rangle$, with $j = n-m+1, \ldots, n$.
The initial state is:

$$|\psi(0)\rangle = \frac{1}{\sqrt{n}} \frac{1}{\sqrt{n-1}} \sum_{x,y=1}^{n} (1-\delta_{xy}) |x\rangle |y\rangle .$$

The eigenpairs of W' are obtained by the singular values and vectors of C, they are:

- the eigenvalues $e^{\pm 2i\theta_1}$, related to the singular value $\lambda_1 = \cos\theta_1 = \frac{1}{n-1}$, with the corresponding eigenvectors $|\theta_j^{\pm}\rangle = \frac{-(A+e^{\pm i\theta_1}B)|\nu_j\rangle}{\sqrt{2}\sin\theta_1}$, for $j = 1, \ldots, n-m-1$.
- the eigenvalues $e^{\pm 2i\theta_2}$, related to the singular value $\lambda_2 = \cos\theta_2 = \frac{n-m-1}{n-1}$, with the corresponding eigenvectors $|\theta_{n-m}^{\pm}\rangle = \frac{(A-e^{\pm i\theta_2}B)|\nu_{n-m}\rangle}{\sqrt{2}\sin\theta_2}$.
- the eigenvalues 1, related to the singular value $\lambda_3 = \cos\theta_3 = 1$, with the corresponding eigenvectors $|\theta_j\rangle = A|j\rangle$, for $j = n-m+1, \ldots, n$.

We are so interested only on the eigenvectors non orthogonal to $|\psi(0)\rangle$. The $|\theta_j\rangle$ for $j = n-m+1, \ldots, n$ and the $|\theta_j^{\pm}\rangle$ for $j = 1, \ldots, n-m-1$ are orthogonal to the initial state. The remaining eigenvectors are the $|\theta_{n-m}^{\pm}\rangle$ and the ones related to 1, which has not been addressed yet. The initial state can so be written as:

$$|\psi(0)\rangle = c^+ |\theta_{n-m}^+\rangle + c^- |\theta_{n-m}^-\rangle + |\beta\rangle ,$$

where $|\beta\rangle$ is the component in the eigenspace of 1 and the coefficients $c^{\pm}$ are:

$$c^{\pm} = \frac{\sqrt{n-m}(1-e^{\mp i\theta_2})}{\sqrt{2n}\sin\theta_2}.$$

Applying $(W')^t$ to $|\psi(0)\rangle$ we obtain:

$$|\psi(t)\rangle = (W')^t |\psi(0)\rangle = c^+ e^{2i\theta_2 t} |\theta_{n-m}^+\rangle + c^- e^{-2i\theta_2 t} |\theta_{n-m}^-\rangle + |\beta\rangle .$$

Hence:

$$|\psi(t)\rangle - |\psi(0)\rangle = c^+(e^{2i\theta_2 t}-1) |\theta_{n-m}^+\rangle + c^-(e^{-2i\theta_2 t}-1) |\theta_{n-m}^-\rangle ,$$

and

$$\| \, |\psi(t)\rangle - |\psi(0)\rangle \, \|^2 = |c^+(e^{2i\theta_2 t}-1)|^2 + |c^-(e^{-2i\theta_2 t}-1)|^2 =$$

$$= \frac{4(n-1)(n-m)}{n(2n-m-2)} \left(1 - T_{2t}\left(\frac{n-m-1}{n-1}\right)\right).$$

Remember that $F(T) = \frac{1}{T-1}\sum_{t=0}^{T} \| \, |\psi(t)\rangle - |\psi(0)\rangle \, \|^2$ (1.32). Using $\sum_{t=0}^{T} T_{2t}\left(\frac{n-m-1}{n-1}\right) = \frac{1}{2} + \frac{1}{2}U_{2T}\left(\frac{n-m-1}{n-1}\right)$ we obtain:

$$F(T) = \frac{2(n-1)(n-m)\left(2T+1-U_{2T}\left(\frac{n-m-1}{n-1}\right)\right)}{n(2n-m-2)(T+1)}.$$

For $n \gg m$, we can obtain the quantum hitting time $H_{P',M}$ by inverting the Laurent series of the equation $F(T) = 1 - \frac{m}{n}$. The first terms are:

$$H_{P',M} = \frac{j_0^{-1}\left(\frac{1}{2}\right)}{2}\sqrt{\frac{n}{2m}} - \frac{\sqrt{1-\frac{1}{4}j_0^{-1}\left(\frac{1}{2}\right)^2}}{1+2\sqrt{1-\frac{1}{4}j_0^{-1}\left(\frac{1}{2}\right)^2}} + O\left(\frac{1}{\sqrt{n}}\right),$$

where j_0 is a spherical Bessel function of the first kind and $j_0^{-1}\left(\frac{1}{2}\right)$ is approximately 1.9. We now compute the success probability when we use the hitting time. First of all we calculate the vectors $|\theta_{n-m}^{\pm}\rangle$ and $|\beta\rangle$ to explicit $|\psi(t)\rangle$:

$$\begin{aligned}
|\theta_{n-m}^{\pm}\rangle =& \frac{1}{\sqrt{2}\sin\theta_2}(A - e^{\pm i\theta_2}B)\,|u^{(n-m)}\rangle = \\
=& \frac{1}{\sqrt{2(n-m)}\sin\theta_2}\left(\sum_{x=1}^{n-m}|\alpha_x\rangle - e^{\pm i\theta_2}\sum_{y=1}^{n-m}|\beta_y\rangle\right) = \\
=& \frac{1}{\sqrt{2(n-1)(n-m)}\sin\theta_2}\left((1-e^{\pm i\theta_2}\sum_{x,y=1}^{n-m}(1-\delta_{xy})\,|x\rangle\,|y\rangle + \right. \\
& \left. + \sum_{x=1}^{n-m}\sum_{y=n-m+1}^{n}|x\rangle\,|y\rangle - e^{\pm i\theta_2}\sum_{x=n-m+1}^{n}\sum_{y=1}^{n-m}|x\rangle\,|y\rangle\right), \\
|\beta\rangle =& \frac{1}{\sqrt{n(n-1)}}\left(\frac{-m}{2n-m-2}\sum_{x,y=1}^{n-m}(1-\delta_{xy})\,|x\rangle\,|y\rangle + \right. \\
& \left. + \frac{n-m-1}{2n-m-2}\sum_{x=1}^{n-m}\sum_{y=n-m+1}^{n}(|x\rangle\,|y\rangle + |y\rangle\,|x\rangle) + \sum_{x,y=n-m+1}^{n}(1-\delta_{xy})\,|x\rangle\,|y\rangle\right).
\end{aligned}$$

The state at time t is:

$$\begin{aligned}|\psi(t)\rangle =&\frac{1}{\sqrt{n(n-1)}}\left(\frac{2(n-1)T_{2t}\left(\frac{n-m-1}{n-1}\right)}{2n-m-2}\sum_{x,y=1}^{n-m}(1-\delta_{xy})\,|x\rangle\,|y\rangle+\right.\\&+\left(\frac{(n-1)T_{2t}\left(\frac{n-m-1}{n-1}\right)}{2n-m-2}-U_{2t-1}\left(\frac{n-m-1}{n-1}\right)\right)\sum_{x=1}^{n-m}\sum_{y=n-m+1}^{n}|x\rangle\,|y\rangle+\\&\left.+\left(\frac{(n-1)T_{2t}\left(\frac{n-m-1}{n-1}\right)}{2n-m-2}+U_{2t-1}\left(\frac{n-m-1}{n-1}\right)\right)\sum_{x=n-m+1}^{n}\sum_{y=1}^{n-m}|x\rangle\,|y\rangle\right)+|\beta\rangle\,.\end{aligned}$$

The probability of finding a marked element after performing a measurement with projectors $\mathcal{P}_M$ and $\mathbb{I}-\mathcal{P}_M$ is $p_M(t):=\langle\psi(t)|\mathcal{P}_M|\psi(t)\rangle$, that is:

$$\begin{aligned}p_M(t)=&\frac{m(m-1)}{n(n-1)}+\frac{m(n-m)}{n(n-1)}\left(\frac{n-1}{2n-m-2}T_{2t}\left(\frac{n-m-1}{n-1}\right)+\right.\\&\left.+U_{2t-1}\left(\frac{n-m-1}{n-1}\right)+\frac{n-m-1}{2n-m-2}\right)^2.\end{aligned}$$

We can determine the critical points of $p_M(t)$ by differentiating with respect to time. The first maximum point is:

$$t_{\max}=\frac{\arctan\left(\frac{\sqrt{2n-m-2}}{\sqrt{m}}\right)}{2\arccos\left(\frac{n-m-1}{n-1}\right)}$$

and its asymptotic expansion is:

$$t_{\max}=\frac{\pi}{4}\sqrt{\frac{n}{2m}}-\frac{1}{4}+O\left(\sqrt{\frac{m}{n}}\right).$$

The value of the probability in this point so is:

$$p_M(t_{\max})=\frac{1}{2}+\sqrt{\frac{m}{2n}}+O\left(\frac{m}{n}\right).$$

The probability of finding a marked vertex when the measurement is performed at time $t_{\max}$ is greater than $\frac{1}{2}$, for any n or m. The time $t_{\max}$ is lower than the hitting time, then the success probability of an algorithm that uses the hitting time as running time will be smaller than the probability at time $t_{\max}$. Evaluating $p_M(t)$ at time $H_{P',M}$ and taking the asymptotic expansion we obtain:

$$p_M(H_{P',M})=\frac{1}{8}j_0^{-1}\left(\frac{1}{2}\right)^2+O\left(\frac{1}{\sqrt{n}}\right). \tag{1.39}$$

The first term is about 0.45 and does not depend on n or m, this shows that the quantum hitting time is a good parameter for the running time of the searching algorithm.

1.5 Limiting distribution

Let $G = (V, E)$ be a graph and consider a quantum walk on it [1]. The time evolution of a quantum walk is described by a unitary matrix. Let $|\psi(0)\rangle$ be the initial state of the system, the state at time t is $|\psi(t)\rangle = U^t |\psi(0)\rangle$. The evolution operator U is unitary and preserves the norm, so, in general, the limit:

$$\lim_{t\to\infty} |\psi(t)\rangle = \lim_{t\to\infty} U^t |\psi(0)\rangle$$

does not exist.
Denoting the nodes as integers: $V = \{1, 2, \dots\}$, a basis state of the system is $|n \to m\rangle$, $n, m \in \mathbb{Z}$, where m is a node adjacent to n, indeed this is the state of the walker on the node n that is about to hop on the node m. We indicate with $p(\psi(0) \rightsquigarrow n, t)$ the probability for the walker to be on the n-th node at time t, starting from the initial node $|\psi(0)\rangle$. This probability is:

$$p(\psi(0) \rightsquigarrow n, t) = \sum_{m=1}^{k_n} |\langle n \to m|\psi(t)\rangle|^2,$$

and it still does not converge for $t \to \infty$, but its average does. The time average for T steps is:

$$\overline{p_T(\psi(0) \rightsquigarrow n)} = \frac{1}{T}\sum_{t=0}^{T-1} p(\psi(0) \rightsquigarrow n, t)$$

and always has a limit as T goes to infinity.
Let $(\lambda_j, |\mu_j\rangle)_{j=1,2,\dots}$ be the eigenpairs of U and for each node n let k_n be the degree of n. Then the following result holds:

Theorem 2. *For an initial state $|\psi(0)\rangle = \sum_j a_j |\mu_j\rangle$,*

$$\lim_{T\to\infty} \overline{p_T(\psi(0) \rightsquigarrow n)} = \sum_{i,j,m} a_i a_j^* \langle n \to m|\mu_i\rangle \langle \mu_j|n \to m\rangle, \tag{1.40}$$

where the sum is only on pairs i, j such that $\lambda_i = \lambda_j$ and on the nodes m adjacent to n.

Proof.

$$\lim_{T\to\infty} \overline{p_T(\psi(0) \rightsquigarrow n)} = \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} p(\psi(0) \rightsquigarrow n, t) = \tag{1.41}$$

$$= \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^{k_n} |\langle n \to m|\psi(t)\rangle|^2. \tag{1.42}$$

So we compute:

$$\begin{aligned} |\langle n \to m|\psi(t)\rangle|^2 &= |\langle n \to m| U^t |\psi(0)\rangle|^2 = \\ &= \Big|\sum_i a_i \lambda_i^t \langle n \to m|\mu_i\rangle\Big|^2 = \\ &= \sum_{i,j} a_i a_j^* (\lambda_i \lambda_j^*)^t \langle n \to m|\mu_i\rangle \langle \mu_j|n \to m\rangle \ . \end{aligned}$$

For the computation of the time average on T steps, we are interested in the quantity:

$$\frac{1}{T}\sum_{t=0}^{T-1}(\lambda_i\lambda_j^*)^t,$$

because $(\lambda_i\lambda_j^*)^t$ is the only time-dependent term.
We can distinguish two cases:

- if $\lambda_i = \lambda_j$:

$$\frac{1}{T}\sum_{t=0}^{T-1}(\lambda_i\lambda_j^*)^t = \frac{1}{T}\sum_{t=0}^{T-1} 1 = 1,$$

- if $\lambda_i \neq \lambda_j$:

$$\left|\frac{1}{T}\sum_{t=0}^{T-1}(\lambda_i\lambda_j^*)^t\right| = \frac{|1-(\lambda_i\lambda_j^*)^T|}{|1-\lambda_i\lambda_j^*|} \leq \frac{2}{T|\lambda_i - \lambda_j|}$$

  and the latter term converges to zero for $T \to \infty$.

So in 1.41 we can sum only over pairs i, j such that to $\lambda_i = \lambda_j$:

$$\begin{aligned} \lim_{T\to\infty} \overline{p_T(\psi(0) \rightsquigarrow n)} &= \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^{k_n} |\langle n \to m|\psi(t)\rangle|^2 = \\ &= \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^{k_n}\sum_{i,j} a_i a_j^* (\lambda_i\lambda_j^*)^t \langle n \to m|\mu_i\rangle \langle \mu_j|n \to m\rangle = \\ &= \sum_{m=1}^{k_n} \sum_{i,j \text{ s.t. } \lambda_i=\lambda_j} a_i a_j^* \langle n \to m|\mu_i\rangle \langle \mu_j|n \to m\rangle \ . \end{aligned}$$

□

Chapter 2

Community detection

2.1 Complex Networks

A network is a system consisting on nodes, which represent the entities of the system, and edges, which correspond to interconnections between pairs of nodes. It can be formalized as a graph $G = (V, E)$ where V is the set of nodes and E is the set of the edges.
The expression "complex networks" refers to graphs with particular structural properties, often found in real-world systems [16, 15, 3, 4]. Some relevant properties regard the degree of the nodes: for example, there can be many nodes with low degree and a small number of nodes with high degree, or the graph may display a specific connection pattern. These networks often have a hierarchical structure, i.e. the communities at one level are completely contained within the larger communities at all higher levels, and also, almost always have an organisation into communities. We are here interested in detecting these communities.

2.2 Community detection

A community of a graph is a dense-connected subset of nodes. A community can be identified by two conditions: to be a connected subgraph and to have nodes more likely to connect to each other than to other nodes from different communities. A way to divide the nodes into communities is graph partitioning, a grouping of the nodes in disjoint sets, with a predefined number and size of the communities. Another often-used method is community detection, a grouping obtained by an investigation on the structure of the graph, without an initial choice of the number and of the size of the communities. We are here interested into applying the latter method.
Finding the community structure of a network by visual inspection is often impossible. We need to employ other techniques to determine structural features, such as defining

suitable metrics, that can help us understand the network data. An example of an important and useful class of network measures is the class of centrality measures. Centrality quantifies how important vertices (or edges) are in a networked system. A standard example of a centrality measure is the degree. The nodes that are most important, according to this measure, will be called "hubs" in this work, and they are at the center of the different communities.

2.3 Fourier Quantum Walks

Our purpose is to use quantum walks to detect the communities of a network [14]. In particular, we decide to use the formalism of coined quantum walks at discrete time. Let G be a graph with N nodes. The Hilbert space of the system is $\mathcal{H} = \mathcal{H}_1 \oplus \mathcal{H}_2 \oplus \cdots \oplus \mathcal{H}_N$, where each $\mathcal{H}_i$ is the Hilbert space of the i-th node. Any $\mathcal{H}_i$ is spanned by the states $|i \to j_1\rangle, |i \to j_2\rangle, \ldots, |i \to j_{k_i}\rangle$, where k_i is the degree of the node i. A state $|i \to j_h\rangle$ resides on node i and is about to hop to the adjacent node j_h. A basis of $\mathcal{H}$ is the union of the basis of all the $\mathcal{H}_i$ and the dimension of $\mathcal{H}$ is $D := \sum_{i=1}^{N} k_i$. A state of the system at time $t \in \mathbb{N}$ is a normalized vector $|\psi(t)\rangle \in \mathcal{H}$. It can be written as superposition of the basis states:

$$|\psi(t)\rangle = \sum_{i=1}^{N} \sum_{h=1}^{k_i} \psi_{ih}(t) \, |i \to j_h\rangle ,$$

where the coefficients $\psi_{ih}(t)$ are complex numbers and $\langle\psi(t)|\psi(t)\rangle = 1$. The time evolution operator U is the product of the shift operator and the coin operator: $U = S \cdot C$. The shift operator $S : \mathcal{H} \longrightarrow \mathcal{H}$ acts in the following way on the basis states:

$$S \, |i \to j_h\rangle = |j_h \to i\rangle .$$

The coin operator $C : \mathcal{H} \longrightarrow \mathcal{H}$ is defined as $C = C_1 \oplus C_2 \oplus \cdots \oplus C_N$, where $C_i : \mathcal{H}_i \longrightarrow \mathcal{H}_i$. There are several possible choices for C_i, here we take the Fourier matrix, that, for any $i = 1, \ldots, N$, is:

$$C_i = \frac{1}{\sqrt{k_i}} \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & e^{\frac{i2\pi}{k_i}} & e^{\frac{2i2\pi}{k_i}} & \ldots & e^{\frac{(k_i-1)i2\pi}{k_i}} \\ 1 & e^{\frac{2i2\pi}{k_i}} & e^{\frac{4i2\pi}{k_i}} & \ldots & e^{\frac{2(k_i-1)i2\pi}{k_i}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{(k_i-1)i2\pi}{k_i}} & e^{\frac{2(k_i-1)i2\pi}{k_i}} & \ldots & e^{\frac{(k_i-1)(k_i-1)i2\pi}{k_i}} \end{pmatrix} .$$

The transition probability for the walker to start from a node i and reach a node ℓ

at time t is:

$$p(i \rightsquigarrow \ell, t) = \frac{1}{k_i} \sum_{s=1}^{k_\ell} \sum_{h=1}^{k_i} | \langle \ell \to m_s | U^t | i \to j_h \rangle |^2 , \tag{2.1}$$

where j_h are the nodes adjacent to i and m_s are the nodes adjacent to ℓ. The normalized transition probability is:

$$P(i \rightsquigarrow \ell, t) = \frac{p(i \rightsquigarrow \ell, t)}{k_\ell} .$$

The infinite-time average of the transition probability is:

$$\overline{p(i \rightsquigarrow \ell)} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} p(i \rightsquigarrow \ell, t) = \lim_{T\to\infty} \frac{1}{T} \frac{1}{k_i} \sum_{t=0}^{T-1} \sum_{s=1}^{k_\ell} \sum_{h=1}^{k_i} | \langle \ell \to m_s | U^t | i \to j_h \rangle |^2 .$$

We can write it in terms of the eigenvectors of U: $\{|\mu_r\rangle\}_{r=1}^{D}$, thanks to the Theorem 2:

$$\overline{p(i \rightsquigarrow \ell)} = \frac{1}{k_i} \sum_{r=1}^{D} \sum_{s=1}^{k_\ell} \sum_{h=1}^{k_i} | \langle \ell \to m_s | \mu_r \rangle |^2 \, | \langle \mu_r | i \to j_h \rangle |^2 . \tag{2.2}$$

Here we rewrite the proof in this formalism to find the previous equation 2.2:

$$\begin{aligned}
\overline{p(i \rightsquigarrow \ell)} &= \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} p(i \rightsquigarrow \ell, t) = \\
&= \lim_{T\to\infty} \frac{1}{T} \frac{1}{k_i} \sum_{t=0}^{T-1} \sum_{s=1}^{k_\ell} \sum_{h=1}^{k_i} | \langle \ell \to m_s | U^t | i \to j_h \rangle |^2 = \\
&= \lim_{T\to\infty} \frac{1}{T} \frac{1}{k_i} \sum_{t=0}^{T-1} \sum_{s=1}^{k_\ell} \sum_{h=1}^{k_i} \sum_{r,u} e^{i(\theta_{\mu_r} - \theta_{\mu_u})t} \langle \mu_u | \ell \to m_s \rangle \langle \ell \to m_s | \mu_r \rangle \langle i \to j_h | \mu_u \rangle \langle \mu_r | i \to j_h \rangle = \\
&= \frac{1}{k_i} \sum_{s=1}^{k_\ell} \sum_{h=1}^{k_i} \sum_{r=1}^{D} \langle \mu_r | \ell \to m_s \rangle \langle \ell \to m_s | \mu_r \rangle \langle i \to j_h | \mu_r \rangle \langle \mu_r | i \to j_h \rangle = \\
&= \frac{1}{k_i} \sum_{r=1}^{D} \sum_{s=1}^{k_\ell} \sum_{h=1}^{k_i} | \langle \ell \to m_s | \mu_r \rangle |^2 \, | \langle \mu_r | i \to j_h \rangle |^2 ,
\end{aligned}$$

where we used the following calculations:

$$\sum_{h=1}^{k_i} |\langle \ell \to m_s|U^t|i \to j_h\rangle|^2 = \sum_{h=1}^{k_i} \left| \langle \ell \to m_s| \sum_{r=1}^{D} |\mu_r\rangle e^{i\theta_{\mu_r} t} \langle \mu_r| |i \to j_h\rangle \right|^2 =$$
$$= \sum_{h=1}^{k_i} \left| \sum_{r=1}^{D} e^{i\theta_{\mu_r} t} \langle \ell \to m_s|\mu_r\rangle \langle \mu_r|i \to j_h\rangle \right|^2 =$$
$$= \sum_{h=1}^{k_i} \sum_{r,u} e^{i(\theta_{\mu_r} - \theta_{\mu_u})t} \langle \mu_u|\ell \to m_s\rangle \langle \ell \to m_s|\mu_r\rangle \langle i \to j_h|\mu_u\rangle \langle \mu_r|i \to j_h\rangle \ ,$$

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} e^{i(\theta_{\mu_r} - \theta_{\mu_u})t} = \delta_{\mu_r \mu_u} \ .$$

Then, the infinite-time average of the normalized probability is:

$$\overline{P(i \rightsquigarrow \ell)} = \frac{\overline{p(i \rightsquigarrow \ell)}}{k_\ell} = \frac{1}{k_i k_\ell} \sum_{r=1}^{D} \sum_{s=1}^{k_\ell} \sum_{h=1}^{k_i} |\langle \ell \to m_s|\mu_r\rangle|^2 \, |\langle \mu_r|i \to j_h\rangle|^2 \ . \tag{2.3}$$

Our goal is to find a community structure for the graph using the Fourier quantum walk. The idea is to fix a hub and to group the nodes with the highest infinite-time average of the normalized probability starting from this hub in its community; then repeat the same selection for the other hubs. Hubs are selected among the nodes with highest degree.

2.4 Algorithm for community detection

The algorithm that we use to find the community structure of the graph works in this way. The input is the adjacency matrix A of the graph G. It starts sorting the nodes in descending degree. The node of highest degree i is set as first hub. We set a threshold q, chosen as $q = \frac{1}{D}$. For each node ℓ, if the infinite-time average of the normalized probability starting from the hub i to be on ℓ is greater than the threshold:

$$\overline{P(i \rightsquigarrow \ell)} > q \ ,$$

and ℓ is not yet a member of a community, then the node ℓ is set as a member of the community of the hub i. Now this procedure is repeated for the second hub, that is the second-ranked node in terms of degree, then for the third and so on, until all the nodes are included in a community.

We implement this algorithm in MATLAB and write a function c=community_detection(A) that returns the vector c of the hubs of the communities to which each node belongs:

```
function c = community_detection(A)

D=digraph(A);
N=size(A,1);
B = table2array(D.Edges(:,1));

% k for A non-sparse matrix
k=sum(A,2);

% k for A sparse matrix
for i=1:N
    k(i)=size(find(B(:,1)==i),1);
end
k=k';

[deg,h]=sort(k,"descend");
V = [B(:,2) B(:,1)];
M=zeros(size(V,1),1);
for i=1:size(V,1)
    for j=1:size(B,1)
        if(V(i,:)==B(j,:))
            M(i)=j;
        end
    end
end
E = eye(sum(k));

% S:shift operator
S = E(:,M);

% C:coin operator
ind=0;
for s=1:N
    C(ind+1:ind+k(s),ind+1:ind+k(s))=ifft(eye(k(s)))*sqrt(k(s));
    ind=ind+k(s);
end

% U:time-evolution operator
U = S*C;
[Vec, Val]=eig(U);

q=1/sum(k);
c=zeros(N,1);
ell=[1:N];
for i=1:N
```

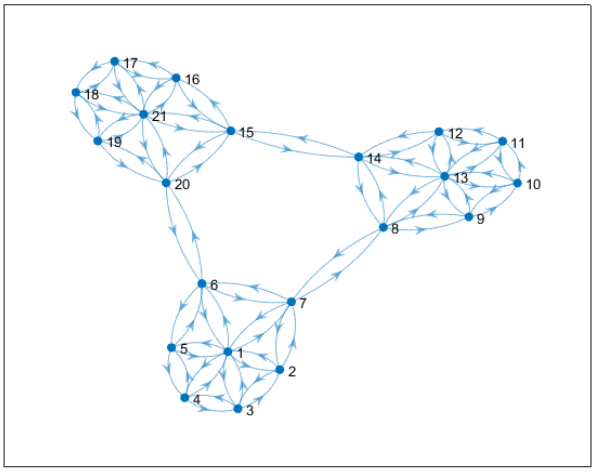
```
    for l=1:N
        if ell(l)~=0
            pil = transprobelem(Vec,k,B,E,h(i),ell(l));
            if pil>q && c(ell(l))==0
                c(ell(l))=h(i);
                ell(l)=0;
            end
        end
    end
end

end
```

In order to compute the infinite-time average of the normalized probability $\overline{P(i \rightsquigarrow \ell)}$ we use the function `pil=transprob(V,k,B,E,i,l)`. It takes in input the matrix of the eigenvectors of the time-evolution operator U, the vector of the degrees of the nodes k, the matrix B, whose rows are the edges of the graph, the identity matrix E of dimension the sum of the degrees, the starting node i and the ending node ℓ:

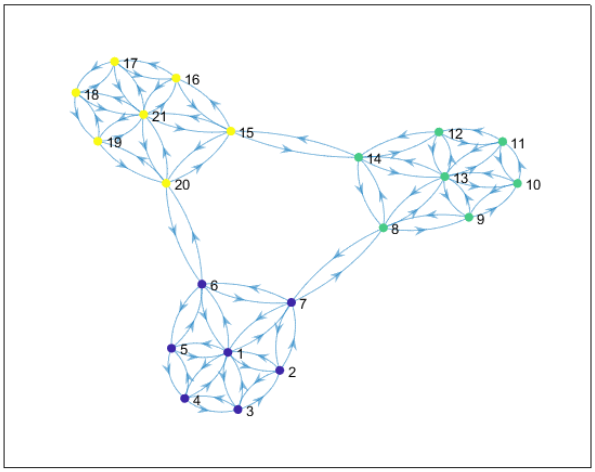
```
function pil = transprob(V,k,B,E,i,l)

pil=0;
ind_1=find(B(:,1)==i);
ind_2=find(B(:,1)==l);

    for j=1:k(i)
        for m=1:k(l)
            for u=1:sum(k)

                pil=pil+(abs(E(:,ind_2(m))'*V(:,u)))^2*(abs(V(:,u)'*E(:,
    ind_1(j))))^2;

            end
        end
    end

pil=pil./(k(i)*k(l));

end
```

2.4.1 A synthetic graph

Consider a prototypical three-community network of 21 nodes and 78 edges [14]:



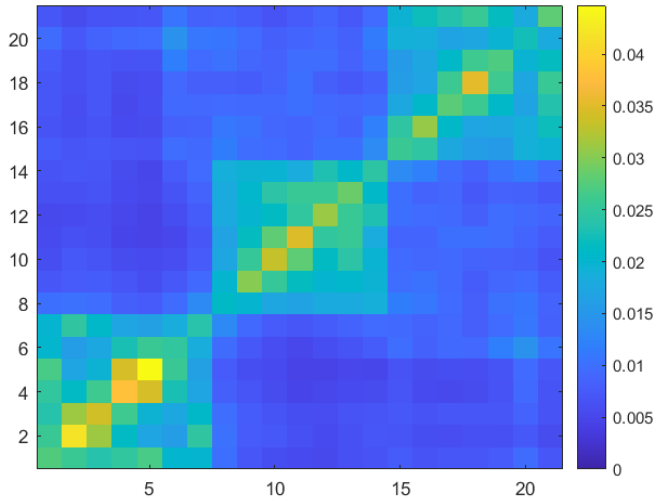
Apply the algorithm to its adjacency matrix. We obtain three communities, with hubs 1, 13 and 21, which are the only nodes of degree equal to 6, the highest. The partition of the nodes into these communities is the following:



Define the matrix P_n as:

$$P_n = \left(\overline{P(i \rightsquigarrow \ell)}\right)_{i=1,\dots,N;\ \ell=1,\dots,N} ,$$

that is the matrix of the infinite-time averages of the normalized probability. The values of P_n for this graph gives us:



We can see the structure in three communities, indeed the values of the average of the probability are higher for the nodes in the same community.

The values of $\overline{P(i \rightsquigarrow \ell)}$ for the hubs $i = 1, 13, 21$ that are over the threshold q reveal which nodes are in the community of each hub i:

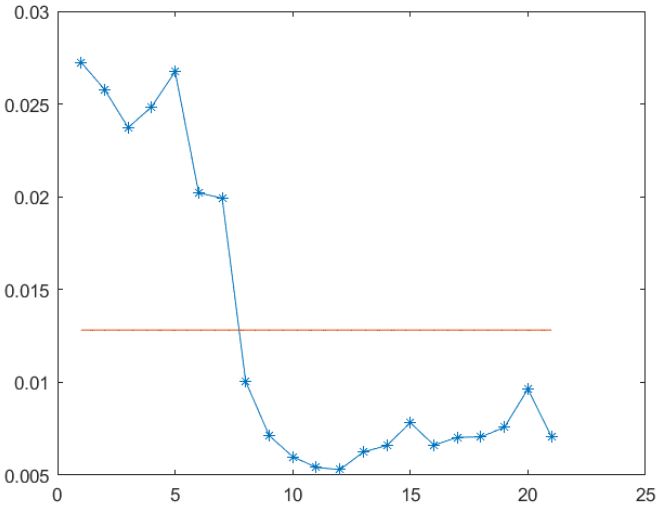


Figure 2.1: $\overline{P(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 3, 4, 5, 6, 7$

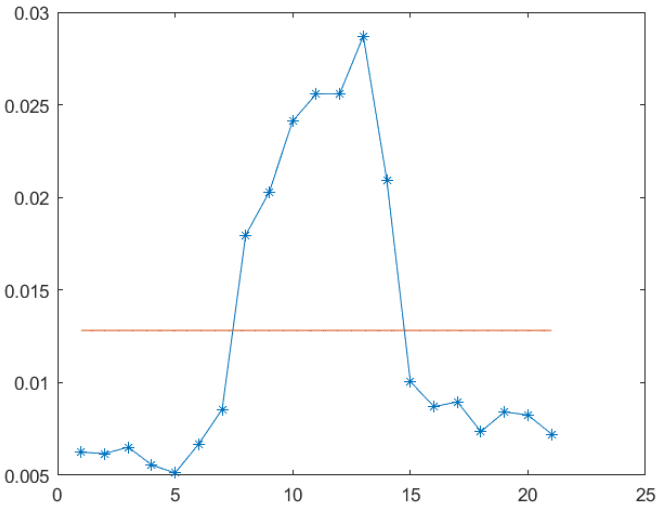


Figure 2.2: $\overline{P(13 \rightsquigarrow \ell)}$,
Nodes: $8, 9, 10, 11, 12, 13, 14$

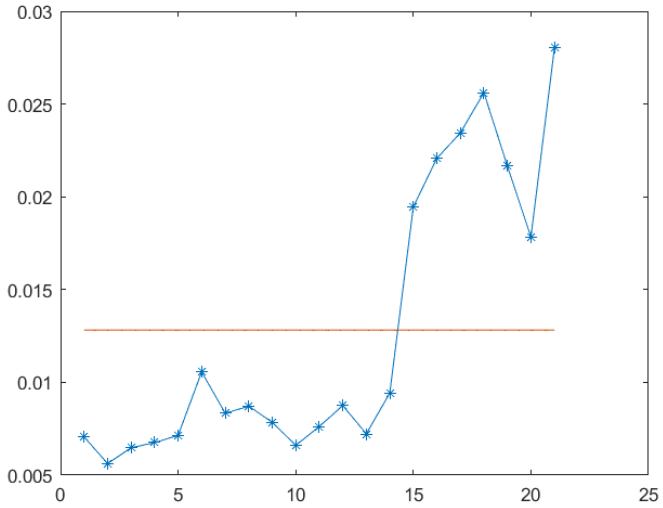


Figure 2.3: $\overline{P(21 \rightsquigarrow \ell)}$,
Nodes: $15, 16, 17, 18, 19, 20, 21$

We notice the presence of a significant gap between probability values of nodes inside and outside each community.

We can apply the same algorithm using a time average of the normalized probability on a finite number of time-steps. The normalized probability with starting node i and ending node ℓ on T steps is:

$$\overline{P_T(i \rightsquigarrow \ell)} = \frac{\overline{p_T(i \rightsquigarrow \ell)}}{k_\ell} = \frac{1}{k_\ell}\frac{1}{T}\sum_{t=0}^{T-1} p(i \rightsquigarrow \ell, t)\ . \tag{2.4}$$

We here set $T = 100$ and in the `community_detection` function we substitute the line:

```
    pil = transprobelem(Vec,k,B,E,h(i),ell(l));
```

with the line:

```
    pil = finprobelem(U,k,B,T,E,h(i),ell(l));
```

The function `pfil = finprobelem(U,k,B,T,E,i,l)` has the number of steps T as additional input and the time-evolution operator U instead of its eigenvectors:

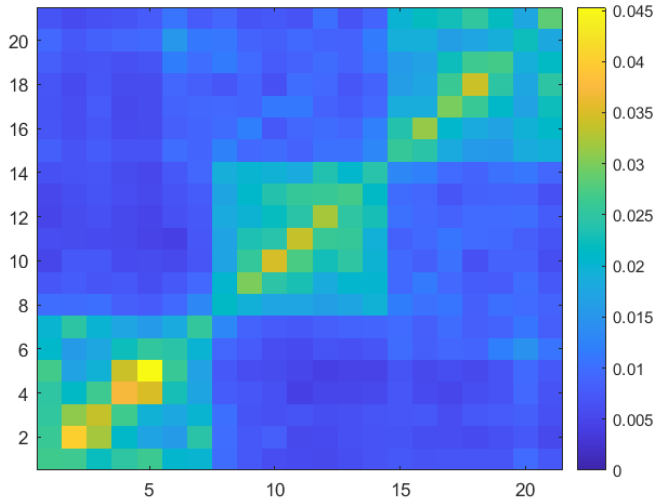
```
function pfil = finprobelem(U,k,B,T,E,i,l)
pfil=0;
aux=U;
ind_1=find(B(:,1)==i);
ind_2=find(B(:,1)==l);
for t=0:T-1
```

```
    for j=1:k(i)
        for m=1:k(l)
            pfil=pfil+(abs(E(:,ind_2(m))'*aux*E(:,ind_1(j))))^2;
        end
    end
    aux=aux*U;
end
pfil=pfil/(k(i)*k(l)*T);
end
```

The output is $\overline{P_{100}(i \rightsquigarrow \ell)}$. Define the matrix P_f as:

$$P_f = \left(\overline{P_T(i \rightsquigarrow \ell)}\right)_{i=1,\ldots,N;\ \ell=1,\ldots,N} ,$$

that is the matrix of the finite-time averages of the normalized probability. The values of P_f for this graph give us:



In this case too we can see the structure in three communities, indeed the nodes in the same community have higher values of the finite-time average of the probability. The values of $\overline{P_{100}(i \rightsquigarrow \ell)}$ for the hubs $i = 1, 13, 21$ that are over the threshold q reveal which nodes are in each community:

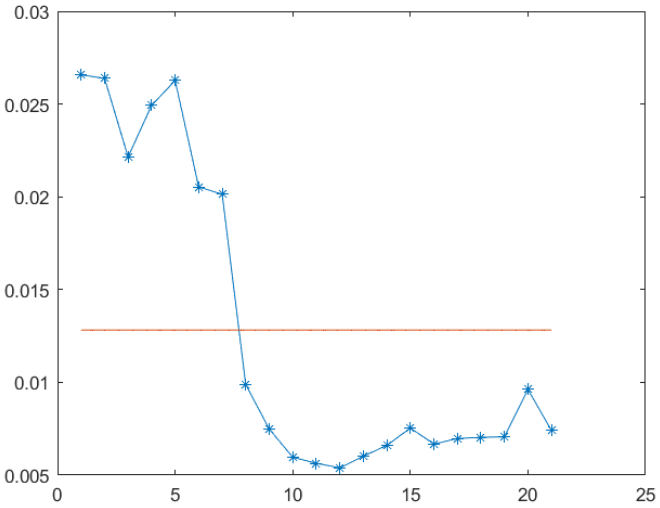


Figure 2.4: $\overline{P_{100}(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 3, 4, 5, 6, 7$

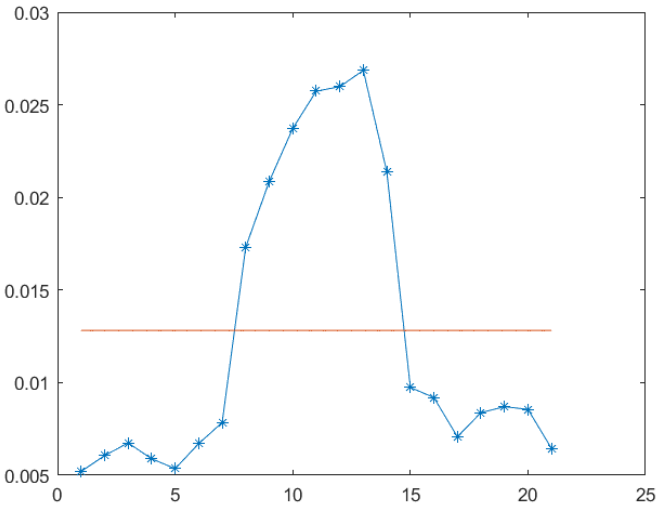


Figure 2.5: $\overline{P_{100}(13 \rightsquigarrow \ell)}$,
Nodes: $8, 9, 10, 11, 12, 13, 14$

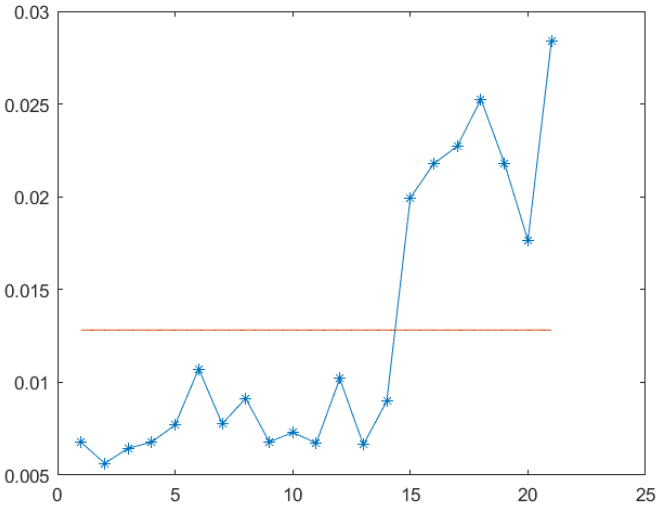
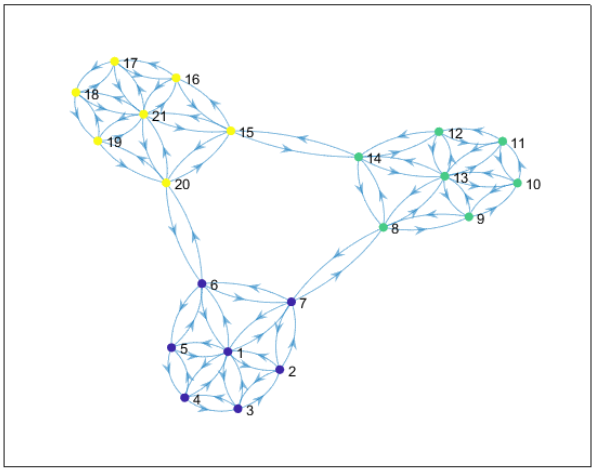


Figure 2.6: $\overline{P_{100}(21 \rightsquigarrow \ell)}$,
Nodes: $15, 16, 17, 18, 19, 20, 21$

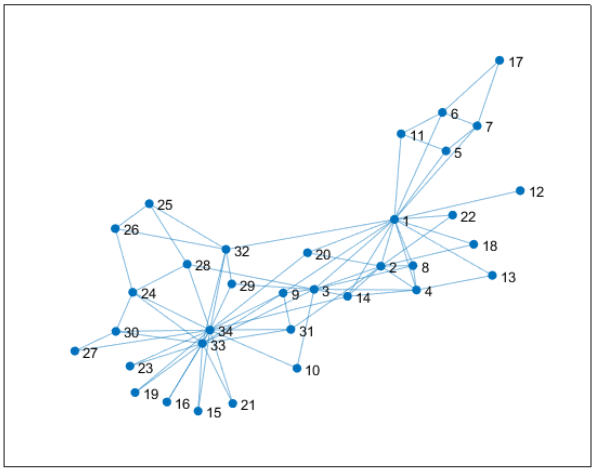
The community organization given by the finite-time average of the normalized probability on the first 100 steps is identical to the previous one:



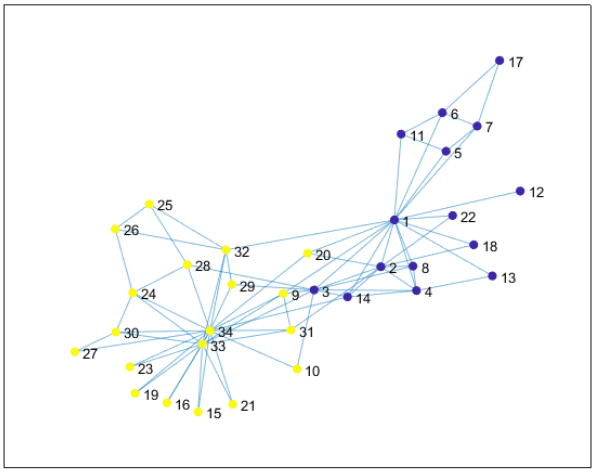
So, in this case, the finite-time average of the probability over 100 time-steps is a good approximation of the infinite one.

2.4.2 Zachary's karate-club network

A social network that has particular attention in the context of community detection is known as Zachary's Karate Club [24]. It is a friendship network in a University karate club, capturing the links between 34 members of a karate club. Each club member knows everyone else because the club has small size. To uncover the true relationships between club members, the sociologist Wayne Zachary documented 78 pairwise links between members who regularly interacted outside the club:



The interest in the dataset is driven by a singular event: a conflict between the club's president and the instructor split the club into two. About half of the members followed the instructor and the other half the president. Today community finding algorithms are often tested based on their ability to infer these two communities from the structure of the network before the split.
We can apply our algorithm to the Zachary's karate-club network and analyse the results. We find two hubs: 34, which has degree equal to 17, and 1, which has degree 16, while all the other nodes have degree lower than 12. The nodes are so divided into two groups: 19 nodes in the community of 34 and 15 nodes in the community of 1:



The values of the infinite-time average of the probability $\overline{P(i \rightsquigarrow \ell)}$ for the hubs $i = 34$ and $i = 1$ reveal the community structure:

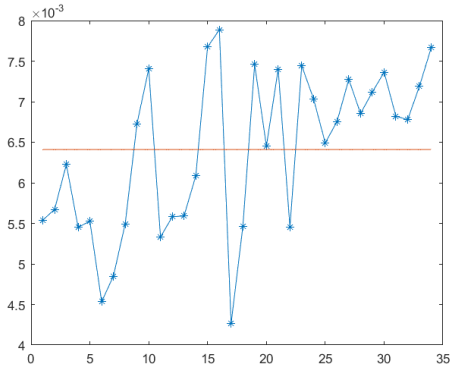


Figure 2.7: $\overline{P(34 \rightsquigarrow \ell)}$,
Nodes: $9, 10, 15, 16, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34$

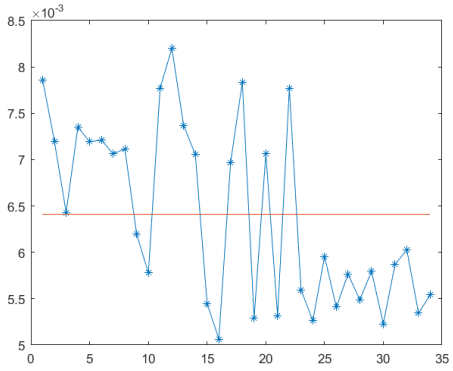


Figure 2.8: $\overline{P(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 22$

Now we apply the algorithm using the finite-time average of the normalized probability over the first 100 steps. The resulting hubs are the same as before and the values of $\overline{P_{100}(i \rightsquigarrow \ell)}$, for $i = 34,\ 1$, reveal that the two communities are identical to the previous ones:

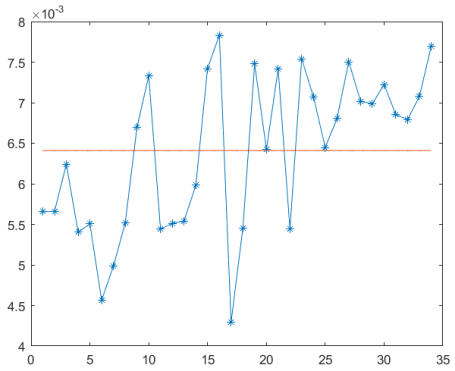


Figure 2.9: $\overline{P_{100}(34 \rightsquigarrow \ell)}$,
Nodes: $9, 10, 15, 16, 19, 20, 21, 23, 24,$
$25, 26, 27, 28, 29, 30, 31, 32, 33, 34$

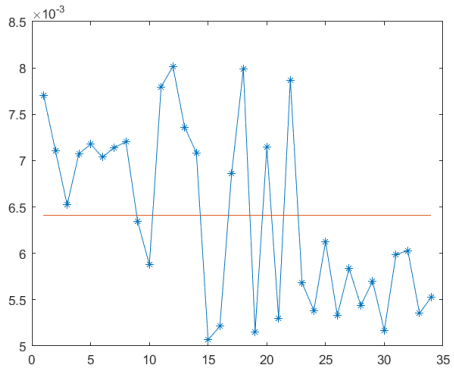
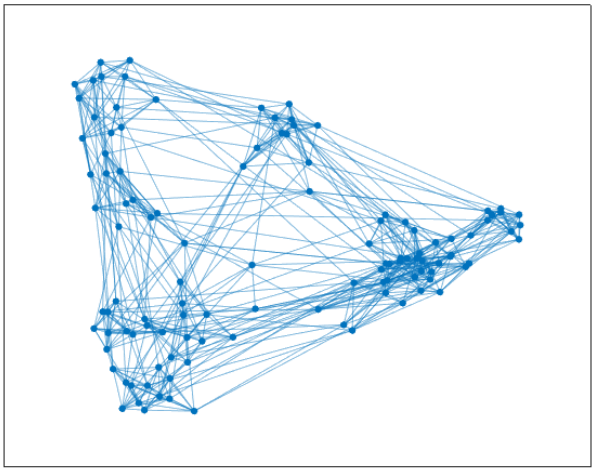


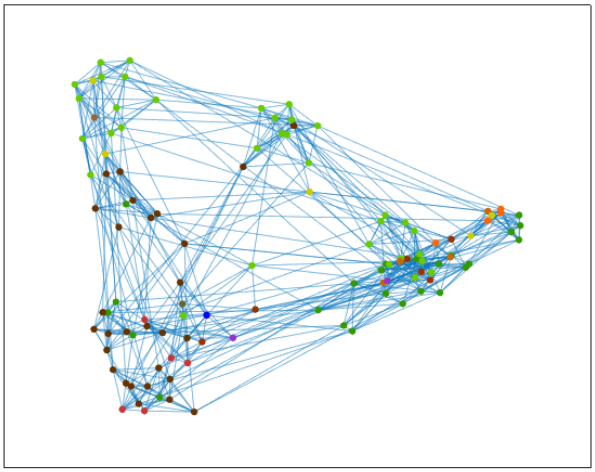
Figure 2.10: $\overline{P_{100}(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 3, 4, 5, 6, 7, 8, 11, 12,$
$13, 14, 17, 18, 22$

2.4.3 College football

Another graph from the same collection of data of the karate club concerns the world of United States college football [8]. This network is a representation of the schedule of Division I games for the 2000 season: vertices of the graph represent the 115 teams and edges represent regular-season games between the two teams they connect.



Applying our algorithm to the graph we obtain 12 communities:



The expected hubs are: 1, 2, 3, 4, 6, 7, 8, 16, 54, 68, 89, 105, because these are the first 12 nodes with the highest degree, but, using the threshold:

$$q = \frac{1}{1226} = 8.1566 \cdot 10^{-04}$$

the nodes 68 and 105 are not selected as hubs, they are replaced by 14 and 18. Among the 12 communities, we can identify three bigger ones, the one related to hub 3, composed by 34 nodes, the one related to hub 2, composed by 28 nodes, and the one related to hub 1, composed by 24 nodes. The values of the $\overline{P(i \rightsquigarrow \ell)}$ for this three hubs reveal the following groups:

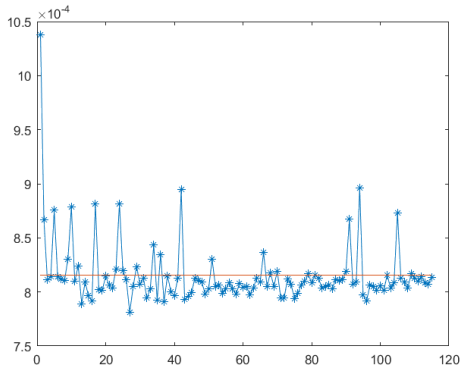


Figure 2.11: $\overline{P(1 \rightsquigarrow \ell)}$,
Hub: 1-BrighamYoung
Nodes: 1, 2, 5, 9, 10, 12, 17, 23, 24, 25, 29, 34, 36, 42, 51, 66, 68, 70, 79, 90, 91, 94, 105, 109

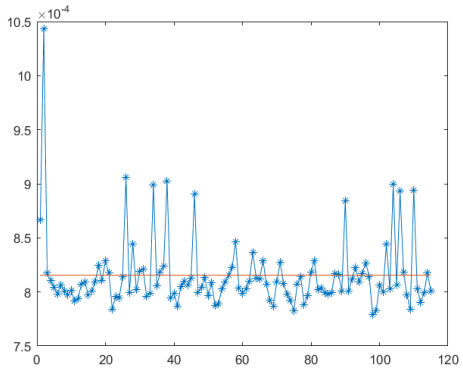


Figure 2.12: $\overline{P(2 \rightsquigarrow \ell)}$,
Hub: 2-FloridaState
Nodes: 3, 18, 20, 21, 26, 28, 30, 31, 37, 38, 46, 57, 58, 63, 71, 80, 81, 87, 88, 93, 95, 96, 102, 104, 106, 107, 110, 114

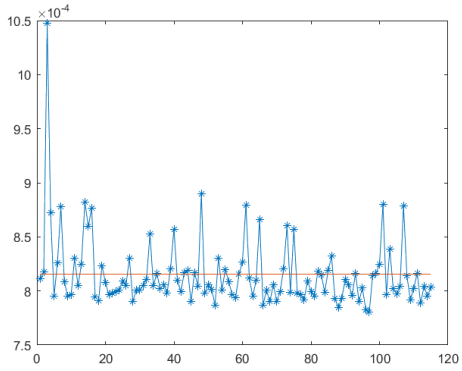


Figure 2.13: $\overline{P(3 \rightsquigarrow \ell)}$,
Hub: 3-Iowa
Nodes: 4, 6, 7, 11, 13, 14, 15, 16, 19, 27, 33, 35, 39, 40, 43, 44, 48, 53, 55, 59, 60, 61, 65, 72, 73, 75, 82, 85, 86, 99, 100, 101, 103, 111

The remaining nine communities are smaller, composed by 1 to 6 nodes. The infinite-time average of the probabilities are the following:

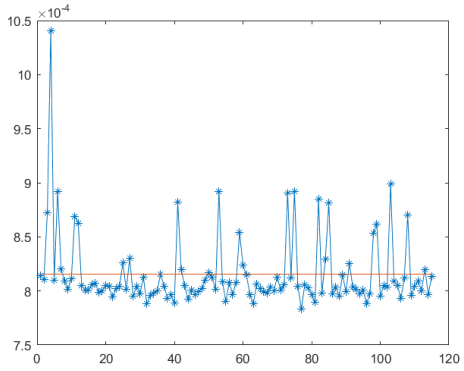


Figure 2.14: $\overline{P(4 \rightsquigarrow \ell)}$,
Hub: 4-KansasState
Nodes: 41, 50, 84, 98, 108, 113

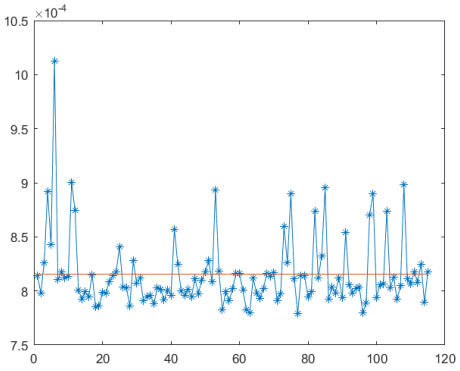


Figure 2.15: $\overline{P(6 \rightsquigarrow \ell)}$,
Hub: 6-TexasTech
Nodes: 8, 54, 74, 115

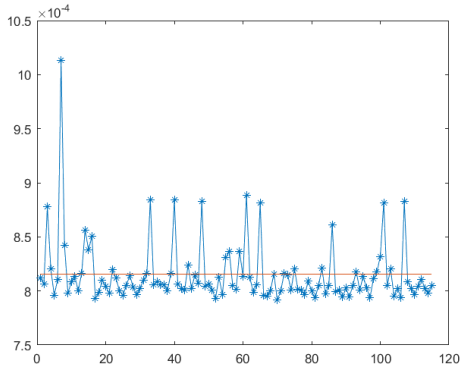


Figure 2.16: $\overline{P(7 \rightsquigarrow \ell)}$,
Hub: 7-PennState
Nodes: 22, 32, 56, 69, 83

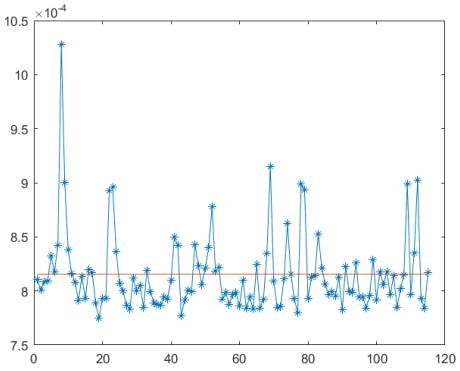


Figure 2.17: $\overline{P(8 \rightsquigarrow \ell)}$,
Hub: 8-SouthernCalifornia
Nodes: 47, 52, 78, 112

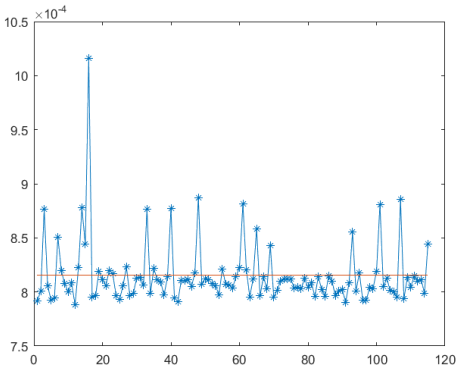


Figure 2.18: $\overline{P(16 \rightsquigarrow \ell)}$,
Hub: 16-Wisconsin
Nodes: 62

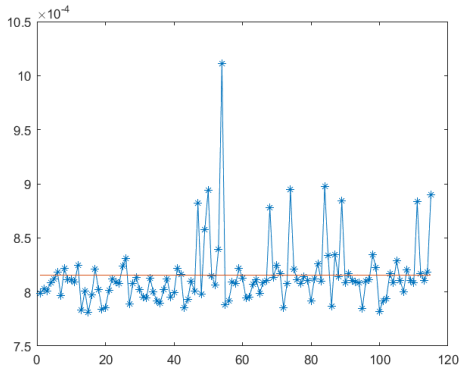


Figure 2.19: $\overline{P(54 \rightsquigarrow \ell)}$,
Hub: 54-SouthernMethodist
Nodes: 49, 89

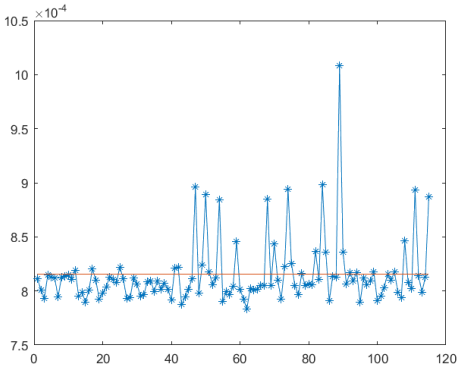


Figure 2.20: $\overline{P(89 \rightsquigarrow \ell)}$,
Hub: 89-Tulsa
Nodes: 92

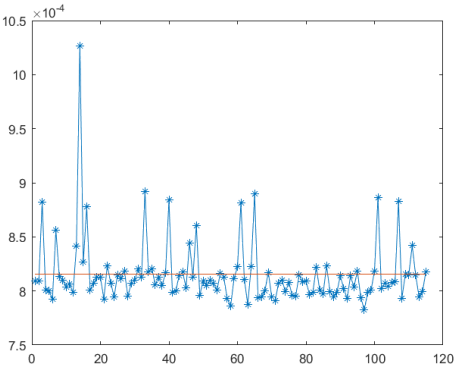


Figure 2.21: $\overline{P(14 \rightsquigarrow \ell)}$,
Hub: 14-Northwestern
Nodes: 64

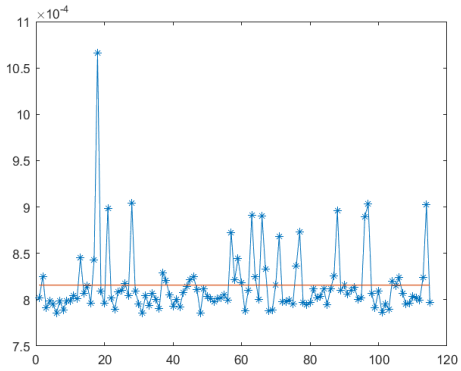


Figure 2.22: $\overline{P(18 \rightsquigarrow \ell)}$,
Hub: 18-Auburn
Nodes: 45, 67, 76, 77, 97

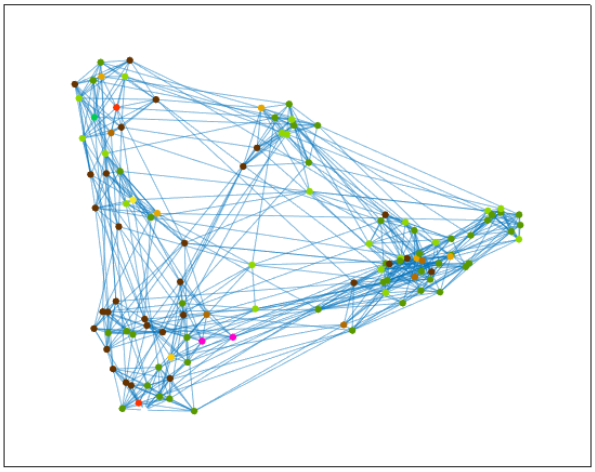
The teams of the Division I are divided into conferences, containing about 8-12 teams each. According to the literature, games are more frequent between members of the same conference. The teams play an average of about seven intraconference games and four interconference games, the last ones are not uniformly distributed, but teams that are geographically closer but belong to different conferences are more likely to play one another than teams separated by large geographic distances. The conferences are the following ones:

1. Atlantic Coast: 2, 26, 34, 38, 46, 90, 104, 106, 110;
2. Big East: 20, 30, 31, 36, 56, 80, 95, 102;
3. Big Ten: 3, 7, 14, 16, 33, 40, 48, 61, 65, 101, 107;
4. Big Twelve: 4, 6, 11, 41, 53, 73, 75, 82, 85, 99, 103, 108;
5. Conference USA: 45, 49, 58, 67, 76, 87, 92, 93, 111, 113;
6. Independents: 37, 43, 81, 83, 91;
7. Mid-American: 13, 15, 19, 27, 32, 35, 39, 44, 55, 62, 72, 86, 100;
8. Mountain West: 1, 5, 10, 17, 24, 42, 94, 105;
9. Pacific Ten: 8, 9, 22, 23, 52, 69, 78, 79, 109, 112;
10. Southeastern: 18, 21, 28, 57, 63, 66, 71, 77, 88, 96, 97, 114;
11. Sun Belt: 12, 25, 51, 60, 64, 70, 98;

12. Western Athletic: 29, 47, 50, 54, 59, 68, 74, 84, 89, 115;

In the communities obtained we can observe this trend for example in the community of hub 18, which has three nodes by the Conference USA and the other two by the Southeastern. In the community of hub 8 three of the four nodes are by the Pacific Ten, in the community of hub 6 three of the four nodes are by the Western Athletic. In the community of hub 1 eight of the twenty-four nodes are by the Mountain West. In the community of hub 2 there are twenty-eight nodes, nine of them are by the Southeastern, six by the Atlantic Coast and six by the Big East. In the community of 3 there are thirty-four nodes, eleven of them are by the Mid-American, ten by the Big Twelve and nine by the Big Ten.

Now use the finite-time average of the normalized probability in the community detection algorithm. In this case the size of the adjacency matrix of the graph is quite large, so we choose to use the average over the first 10 steps. We obtain eleven hubs, that are: 1, 2, 3, 4, 6, 7, 8, 54, 89, 10, 18, so the same as before except 16 and 14, and there is 10 extra.



The communities found are not so similar to the previous ones, due to the small value chosen for T. After only 10 steps we obtain that a lot of nodes are in the community of hub 1, precisely 47, then 30 nodes in the community of 2 and 20 in the community of 3, all other communities are composed by 1, 2 or 5 nodes. The values of the finite-time average of the probabilities are the following:

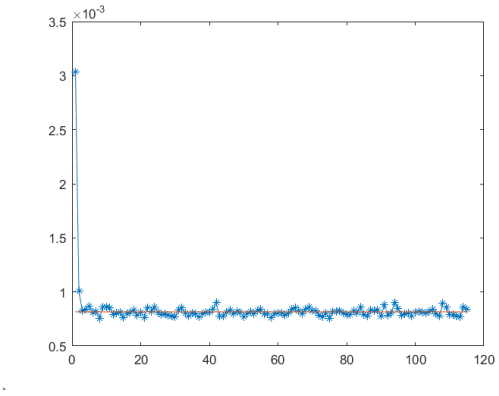


Figure 2.23: $\overline{P_{10}(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 3, 4, 5, 7, 9, 10, 11, 18, 22, 23, 24, 31, 32, 41, 42, 46, 48, 52, 54, 55, 64, 65, 66, 68, 69, 70, 71, 76, 77, 78, 82, 84, 87, 88, 89, 91, 94, 95, 101, 104, 105, 108, 109, 114, 115$

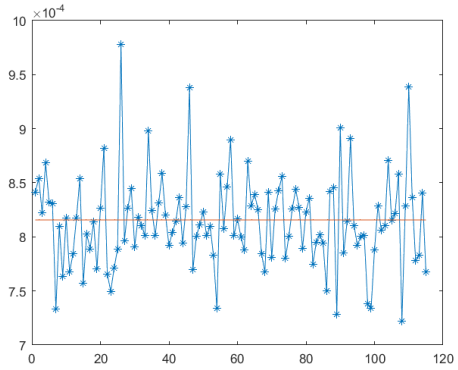


Figure 2.24: $\overline{P_{10}(2 \rightsquigarrow \ell)}$,
Nodes: $6, 13, 14, 20, 21, 26, 28, 29, 34, 35, 37, 38, 39, 43, 45, 50, 57, 58, 60, 63, 72, 73, 80, 81, 90, 93, 106, 107, 110, 111$

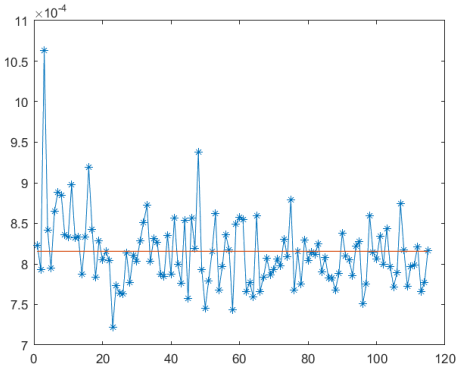


Figure 2.25: $\overline{P_{10}(3 \rightsquigarrow \ell)}$,
Nodes: $8, 12, 15, 16, 17, 19, 33, 36, 44, 47, 53, 56, 59, 61, 75, 79, 83, 98, 103, 112$

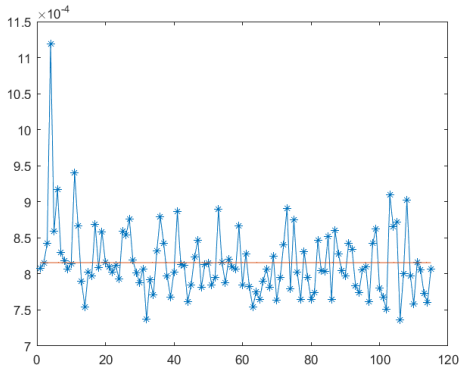


Figure 2.26: $\overline{P_{10}(4 \rightsquigarrow \ell)}$,
Nodes: $25, 27, 85, 92, 99$

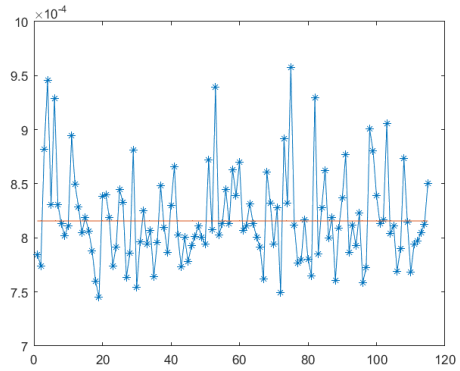


Figure 2.27: $\overline{P_{10}(6 \rightsquigarrow \ell)}$,
Nodes: $40, 51, 74, 100, 102$

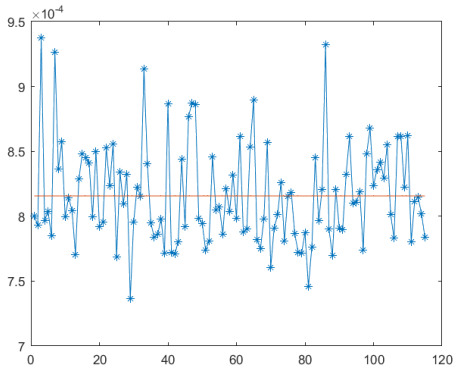


Figure 2.28: $\overline{P_{10}(7 \rightsquigarrow \ell)}$,
Nodes: $86, 96$

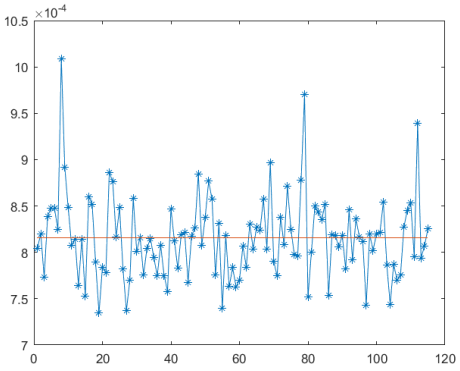


Figure 2.29: $\overline{P_{10}(8 \rightsquigarrow \ell)}$,
Nodes: 67

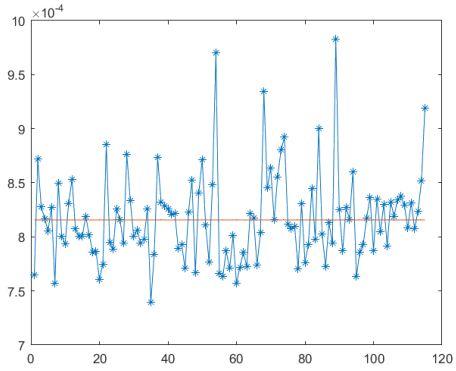


Figure 2.30: $\overline{P_{10}(54 \rightsquigarrow \ell)}$,
Nodes: $49, 113$

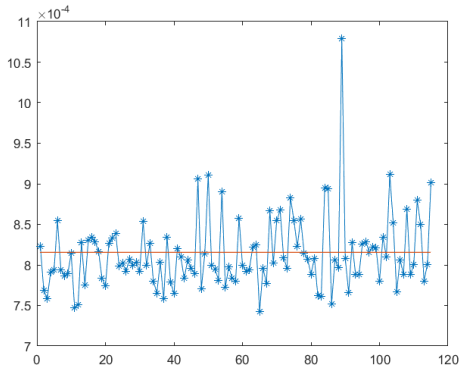


Figure 2.31: $\overline{P_{10}(89 \rightsquigarrow \ell)}$, Nodes: 97

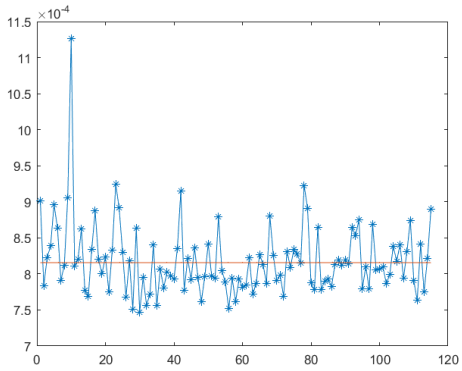


Figure 2.32: $\overline{P_{10}(10 \rightsquigarrow \ell)}$, Nodes: 62

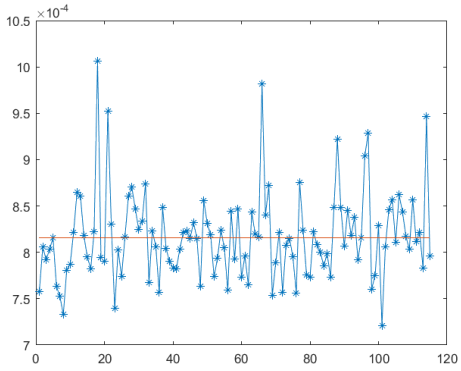


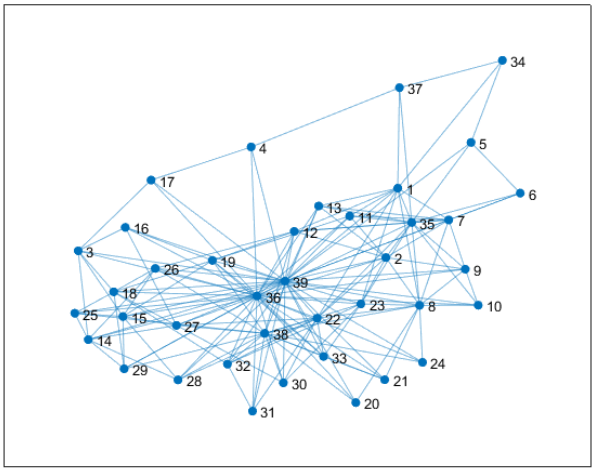
Figure 2.33: $\overline{P_{10}(18 \rightsquigarrow \ell)}$, Nodes: 30

Comparing these results with the previous ones we can state that the finite time $T = 10$ is too small to identify a significant community structure. In the community of hub 1 there are 14 nodes in common for both the methods, 15 in the community of hub 2 and 10 in the community of hub 3. The communities of hubs 4, 7, 8, 89 and 18 are completely different.

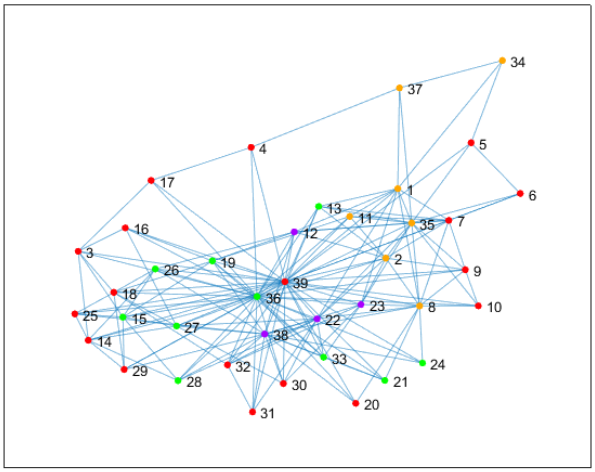
2.4.4 Food Web

Consider the graph of a food web of marine organisms living in the Chesapeake Bay (USA) [2]. The vertices represent the ecosystem's most prominent taxa. Most taxa are represented at the species or genus level, although some vertices represent larger groups

of related species. The edges indicate trophic relationships, i.e. one taxon feeding on another. Although relationships of this kind are inherently directed, we here ignore direction and consider the network to be undirected. The graph has 39 nodes and 170 edges:



Applying the community detection algorithm we find four hubs: 39, 36, 38 and 35, that are the four nodes with highest degree. The four related communities are the following ones:



The values of the infinite-time average of the normalized probability $\overline{P(i \rightsquigarrow \ell)}$ for the hubs $i = 39, 36, 38, 35$ reveal the nodes that belong to each community:

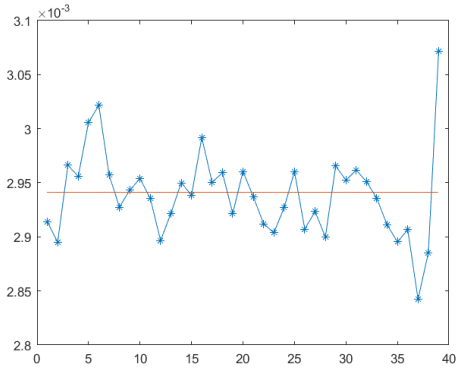


Figure 2.34: $\overline{P(39 \rightsquigarrow \ell)}$,
Nodes: $3, 4, 5, 6, 7, 9, 10, 14, 16, 17, 18, 20, 25, 29, 30, 31, 32, 39$

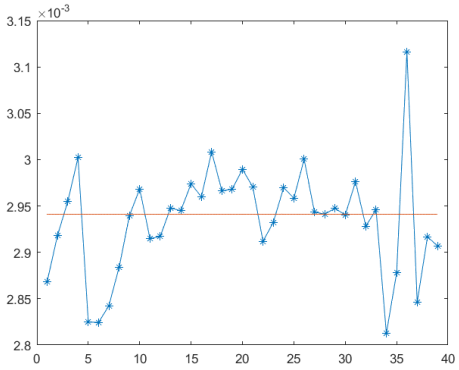


Figure 2.35: $\overline{P(36 \rightsquigarrow \ell)}$,
Nodes: $13, 15, 19, 21, 24, 26, 27, 28, 33, 36$

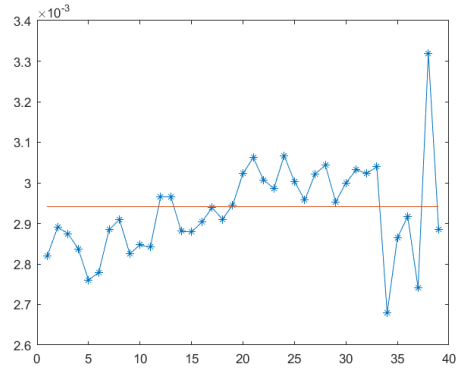


Figure 2.36: $\overline{P(38 \rightsquigarrow \ell)}$,
Nodes: $12, 22, 23, 38$

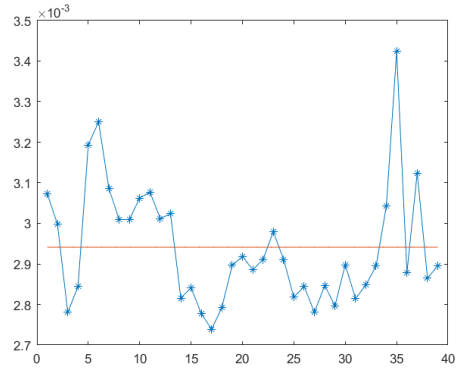
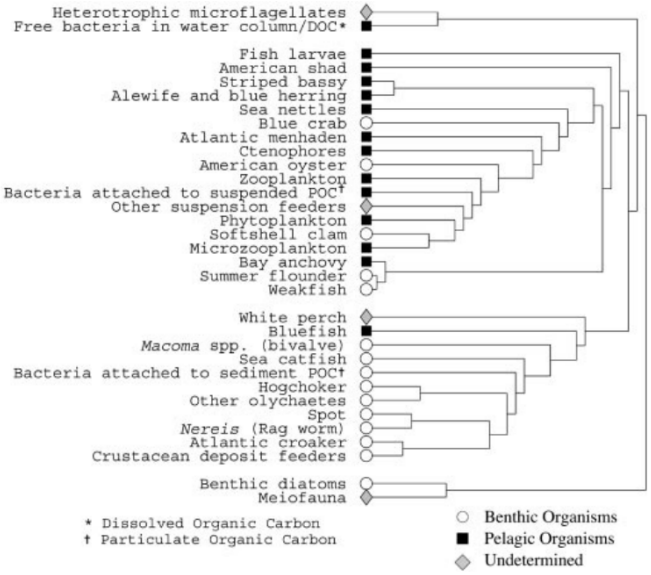
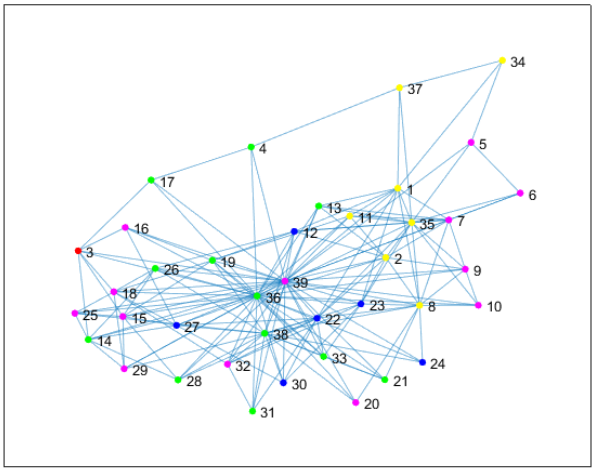


Figure 2.37: $\overline{P(35 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 8, 11, 34, 35, 37$

According to literature, the nodes of the network are divided between pelagic organisms (those that dwell principally near the surface or in the middle depths of the bay) and benthic organisms (those that dwell near the bottom):



Now apply the algorithm using the finite-time average of the normalized probability. Setting $T = 50$, we obtain five communities, so one more than the previous ones. The hub of the fifth community is 15, that is the 12-th node in descending order of degree, and its community consists of only one node, so it is quite irrelevant. The structure found is the following one:



The values of the finite-time average of the probability over 50 steps reveal wich nodes are in each community:

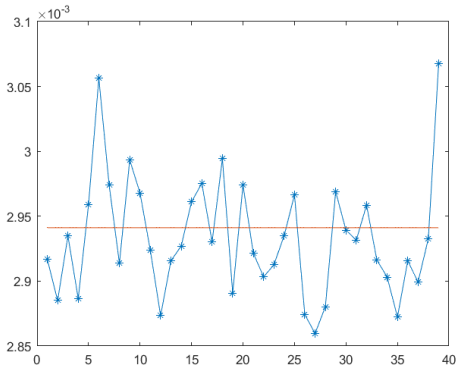


Figure 2.38: $\overline{P_{50}(39 \rightsquigarrow \ell)}$,
Nodes: $5, 6, 7, 9, 10, 15, 16, 18, 20,$
$25, 29, 30, 32, 39$

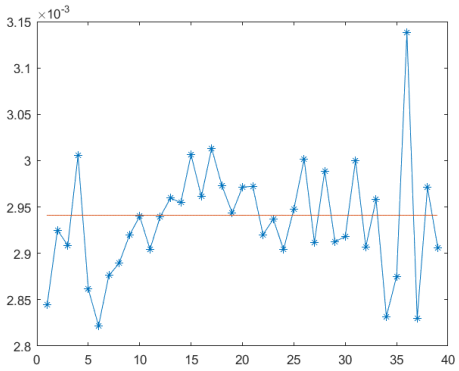


Figure 2.39: $\overline{P_{50}(36 \rightsquigarrow \ell)}$,
Nodes: $4, 13, 14, 17, 19, 21, 24, 26,$
$28, 31, 33, 36, 38$

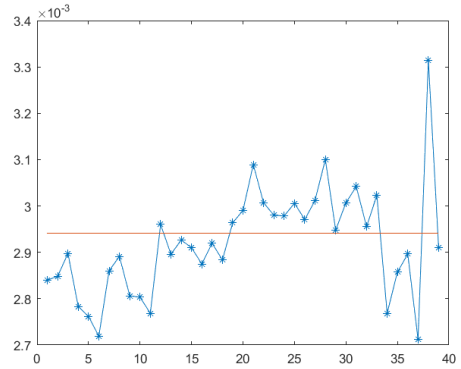


Figure 2.40: $\overline{P_{50}(38 \rightsquigarrow \ell)}$,
Nodes: $12, 22, 23, 24, 27, 30$

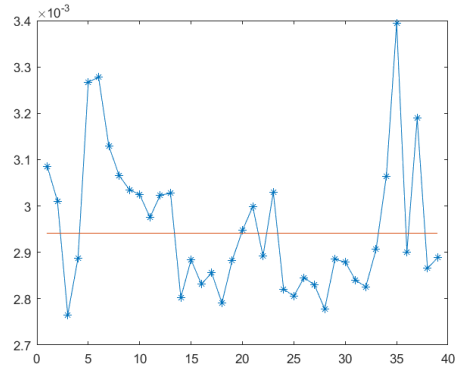


Figure 2.41: $\overline{P_{50}(35 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 8, 11, 34, 35, 37$

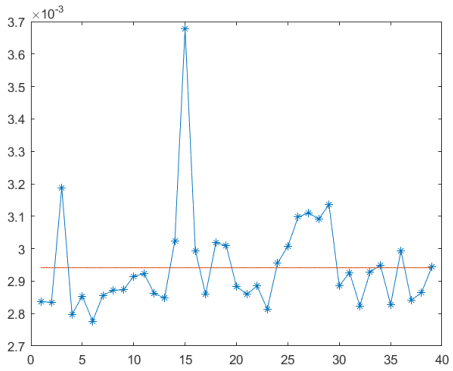
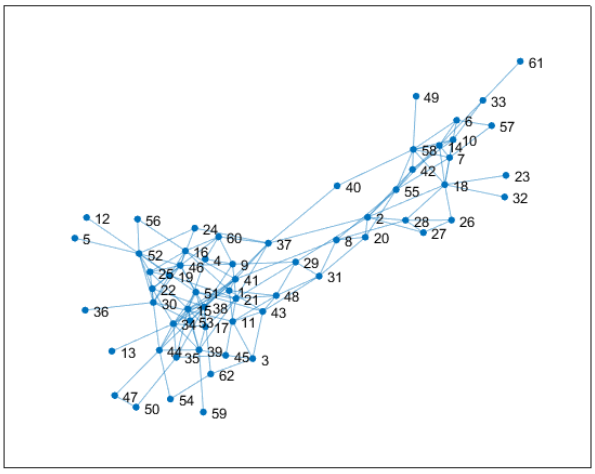


Figure 2.42: $\overline{P_{50}(15 \rightsquigarrow \ell)}$,
Nodes: 3

The community organization of the nodes is very similar to the previous one. The differences we see are that for the community of node 39 we have 15 extra and 3, 4, 14, 17, 30, 31 missing, for the community of node 36 we have 4, 14, 17, 31, 38 extra and 15, 27 missing and for the community of node 38 we have 24, 27, 30 extra and 38 missing.

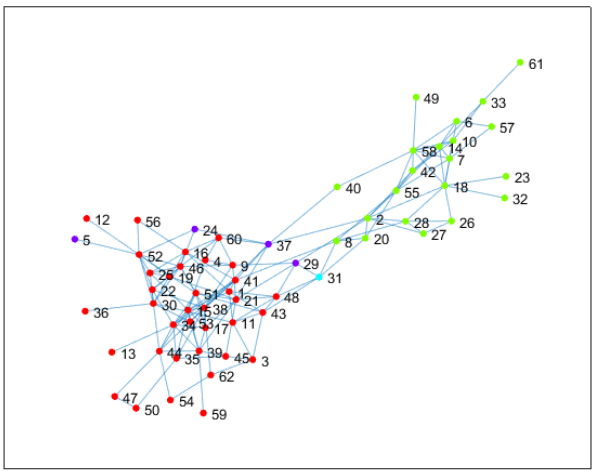
2.4.5 Dolphins

Consider a graph of an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand [12]:



Applying the community detection algorithm to this graph the hubs found are four: 15, 38, 18 and 21, that are among the first seven nodes with highest degrees: 15, 38, 46, 34, 52, 18, 21.

The resulting four communities are the following ones:



The values of $\overline{P(i \rightsquigarrow \ell)}$ for the hubs $i = 15, 38, 18, 21$ reveal the nodes that belong to each community:

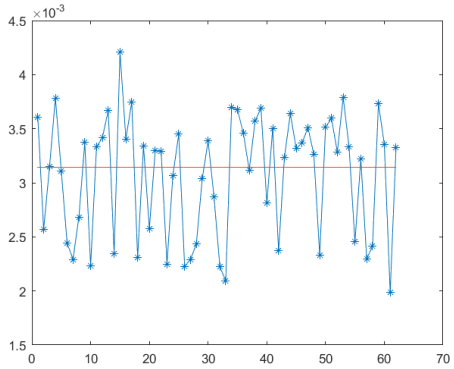


Figure 2.43: $\overline{P(15 \rightsquigarrow \ell)}$,
Nodes: $1, 3, 4, 9, 11, 12, 13, 15, 16, 17, 19, 21, 22, 25, 30, 34, 35, 36, 38, 39, 41, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 56, 59, 60, 62$

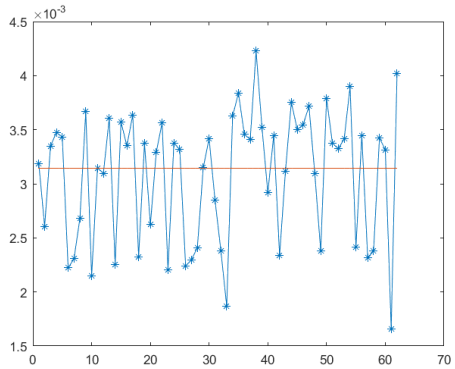


Figure 2.44: $\overline{P(38 \rightsquigarrow \ell)}$,
Nodes: $5, 24, 29, 37$

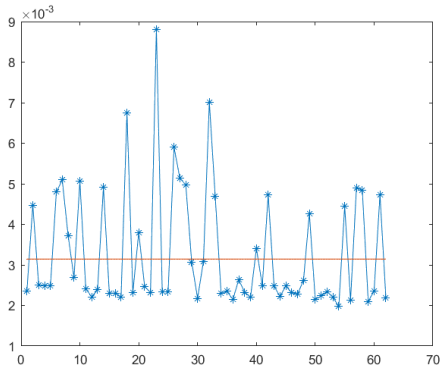


Figure 2.45: $\overline{P(18 \rightsquigarrow \ell)}$,
Nodes: $2, 6, 7, 8, 10, 14, 18, 20, 23, 26, 27, 28, 32, 33, 40, 42, 49, 55, 57, 58, 61$

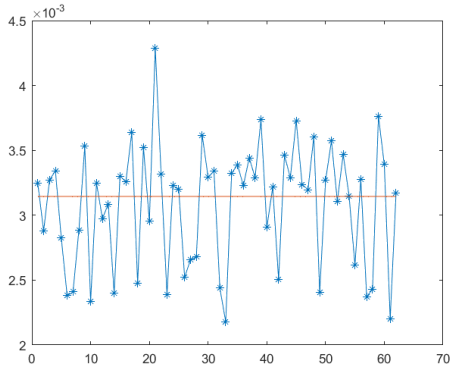
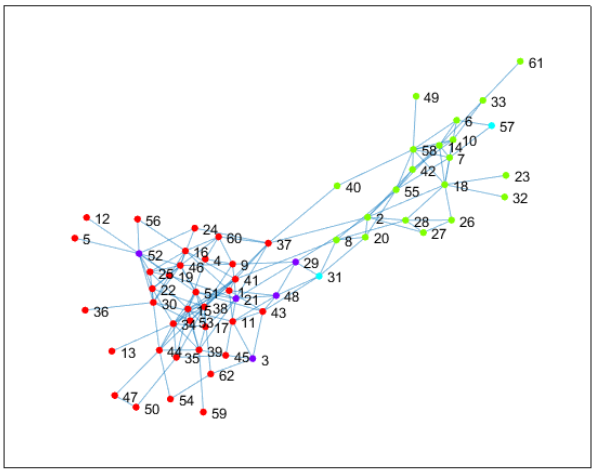


Figure 2.46: $\overline{P(21 \rightsquigarrow \ell)}$,
Nodes: 31

Now we apply the algorithm using the finite-time average of the probability. Set $T = 20$, the resulting hubs are 15, 38, 34 and 18, so there is 34 instead of 21 of the infinite-time method. The four communities obtained are the following ones:



The values of the finite-time average of the normalized probability $\overline{P_{20}(i \rightsquigarrow \ell)}$ for the hubs $i = 15, 38, 34, 18$ reveal the following organization:

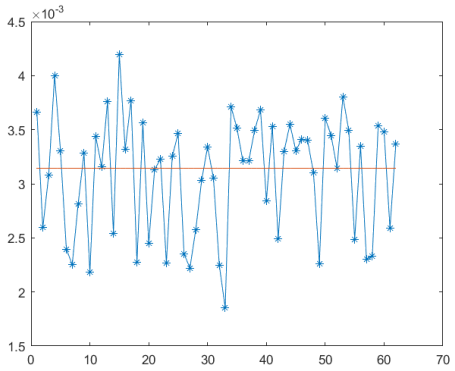


Figure 2.47: $\overline{P_{20}(15 \rightsquigarrow \ell)}$,
Nodes: $1, 4, 5, 9, 11, 12, 13, 15, 16,$
$17, 19, 22, 24, 25, 30, 34, 35, 36, 37,$
$38, 39, 41, 43, 44, 45, 46, 47, 48, 50, 51,$
$53, 54, 56, 59, 60, 62$

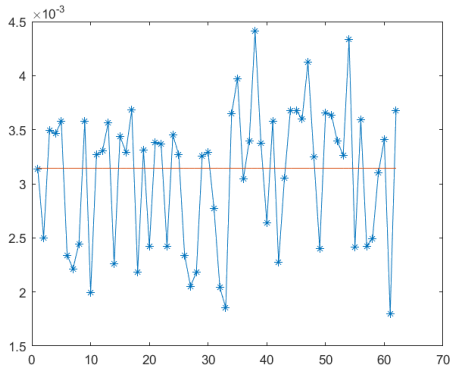


Figure 2.48: $\overline{P_{20}(38 \rightsquigarrow \ell)}$,
Nodes: $3, 21, 29, 48, 52$

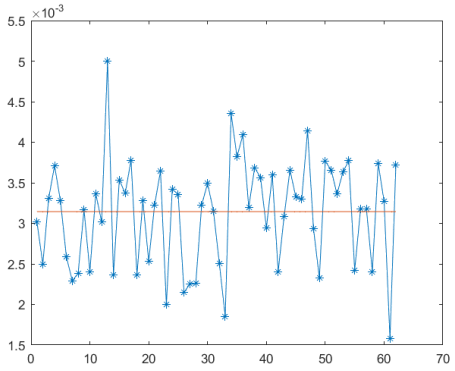


Figure 2.49: $\overline{P_{20}(21 \rightsquigarrow \ell)}$,
Nodes: $31, 57$

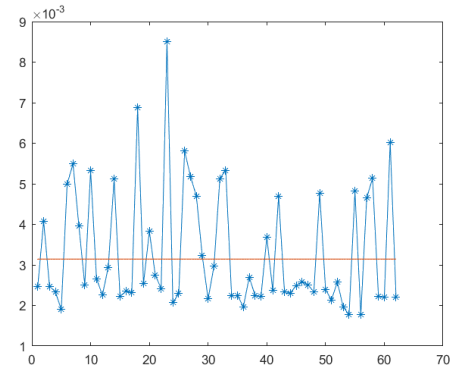


Figure 2.50: $\overline{P_{20}(18 \rightsquigarrow \ell)}$,
Nodes: $2, 6, 7, 8, 10, 14, 18, 20, 23, 26,$
$27, 28, 32, 33, 40, 42, 49, 55, 58, 61$

So we can notice that the communities found are very similar to the previous ones, despite the fact that the value chosen for T is quite small.

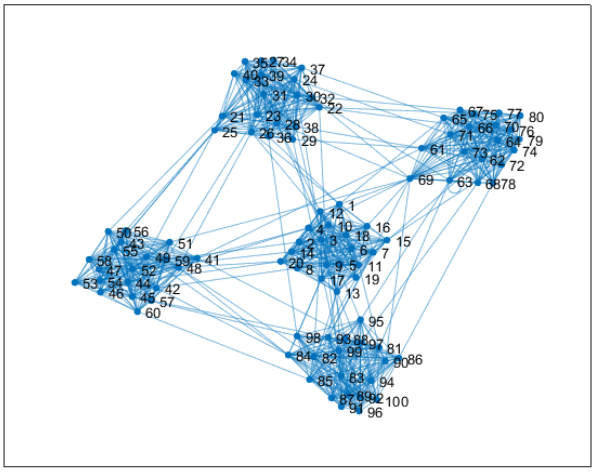
2.4.6 Caveman graph

Caveman graph is a characteristic social network. It is obtained by modifying a set of isolated cliques by removing an edge from each clique and using it to connect to a neighboring clique along a central cycle. Thanks to the help of *NetworkX* [9] we can

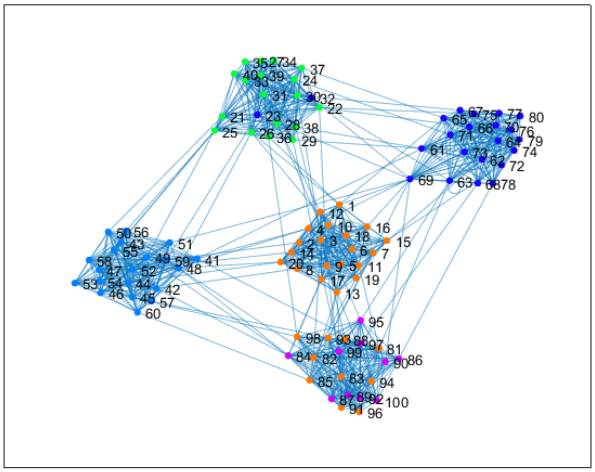
build a relaxed caveman graph, made up of n cliques of size m, where edges are then randomly rewired to link different cliques with probability p. We here set $n = 5$, $m = 20$ and $p = 0.1$ and we can also specify an integer seed, that is an indicator of random number generation state, we here choose seed = 42:

```
G = nx.relaxed_caveman_graph(5, 20, 0.1, seed=42)
```

The graph obtained is the following:



The community detection algorithm finds five hubs: 69, 59, 9, 38, and 81. Node 69 is the one with the highest degree, that is 24, then 59 has degree equal to 22 and then there are seven nodes with degree 21, that are 9, 38, 42, 61, 71, 81 and 84. The resulting community structure is the following one:



Community of 69 is composed by 22 nodes, it consists of the fourth clique, so nodes from 61 to 80, together with two nodes by the first clique: 23 and 32. Community of 59 is strictly the third clique, so nodes from 41 to 60. Community of 9 is composed by the first clique, so nodes from 1 to 20 but also by ten nodes of the fifth clique: 81, 82, 83, 85, 88, 91, 93, 94, 96 and 98. Community of 38 is composed by nodes of the second clique, from 21 to 40, except 23 and 32. Community of 81 is composed by the remaining ten nodes of the fifth clique. Notice that the hubs found are one for each clique and that the resulting communities quite respect the structure in cliques, except for the clique of nodes from 81 to 100, that is split in two different communities, and for two nodes of the clique of nodes from 21 to 40.

2.4.7 Stochastic Block Model

Stochastic block model is a generative model for random graphs that tends to produce graphs containing communities. A stochastic block model is defined by three parameters: the number N of nodes, a partition of the vertex set into r disjoint subsets: $C_1, \ldots, C_r$, called "blocks", and a symmetric $r \times r$ matrix Q of edge probabilities. The edge set is then sampled at random: any two nodes $a \in C_u$ and $b \in C_v$ are linked with probability Q_{uv}. A usual problem is to recover the sets $C_1, \ldots, C_r$ for a given a graph with N nodes and with the edges sampled as described.
Thanks to *NetworkX* [9] we build a stochastic block model graph, giving as input parameters: a list of the sizes of blocks and a list of lists where the v-th element of the u-th list is the density of edges going from the nodes of group C_u to nodes of group C_v, and we can also specify a seed for the random number generation. We here show a couple of examples.

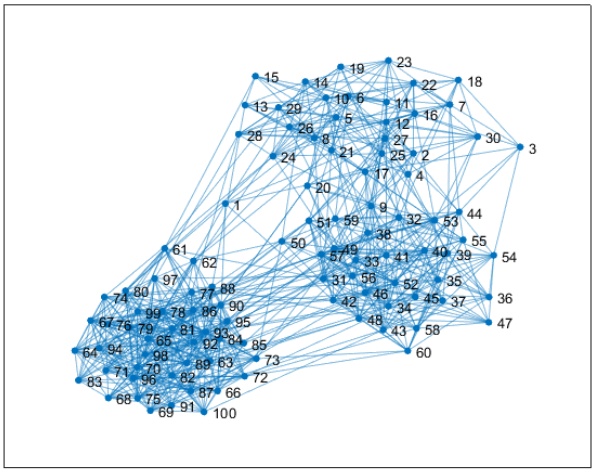
As first example choose a graph with 100 nodes, divided into three sets such that $|C_1| = 30$, $|C_2| = 30$ and $|C_3| = 40$. Set as matrix of edge probabilities:

$$Q = \begin{pmatrix} 0.30 & 0.05 & 0.02 \\ 0.05 & 0.35 & 0.03 \\ 0.02 & 0.03 & 0.40 \end{pmatrix} .$$

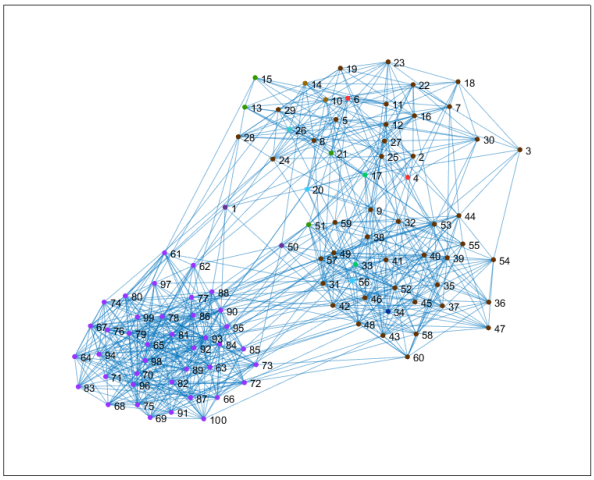
The python code we use is:

```
sizes = [30, 30, 40]
probs = [[0.30, 0.05, 0.02], [0.05, 0.35, 0.03], [0.02, 0.03, 0.40]]
G = nx.stochastic_block_model(sizes, probs, seed=0)
```

The obtained graph is the following:



to which we apply the community detection algorithm. It returns ten hubs: 93, 78, 63, 67, 90, 32, 49, 31, 34 and 52, that are among the first thirteen nodes with highest degree. The resulting community structure is:



There are two big communities, the one of 32 with 44 nodes and the one of 93 with 40 nodes, while other communities are smaller. Community of 31 is composed by 13, 15, 21 and 51. Five communities are composed by two nodes: the one of 67 by 1 and 50, the one of 90 by 20 and 56, the one of 49 by 4 and 6, the one of 34 by 10 and 14 and the one of 52 is composed by 17 and 33. The remaining communities are composed by only one node, the one of 78 by 26 and the one of 63 by 34.

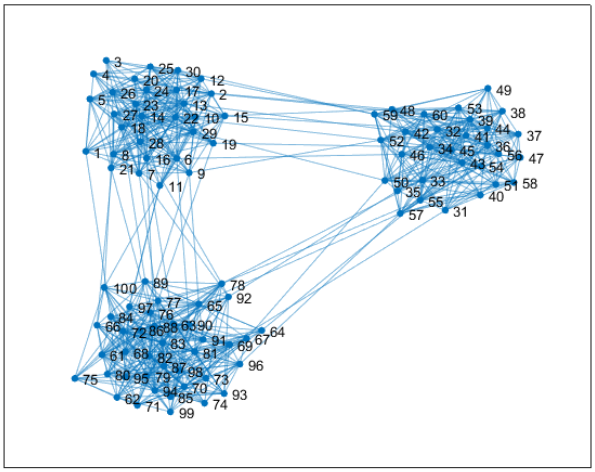
As second example of stochastic block model choose another graph with 100 nodes partitioned as before but with a different matrix of edge probabilities:

$$Q = \begin{pmatrix} 0.50 & 0.01 & 0.01 \\ 0.01 & 0.45 & 0.01 \\ 0.01 & 0.01 & 0.40 \end{pmatrix} .$$

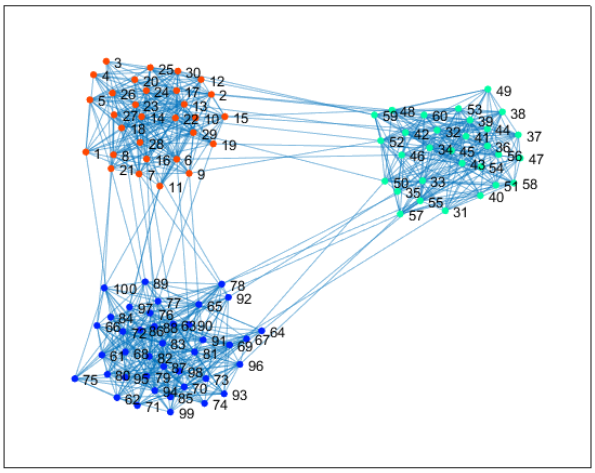
The python code we use is:

```
sizes = [30, 30, 40]
probs = [[0.50, 0.01, 0.01], [0.01, 0.45, 0.01], [0.01, 0.01, 0.40]]
G = nx.stochastic_block_model(sizes, probs, seed=0)
```

The graph obtained is the following:



Applying the community detection algorithm we find three hubs: 65, 45 and 6. The first seven nodes with highest degree are 65, 82, 45, 79, 83, 88 and 6. The resulting community structure is:



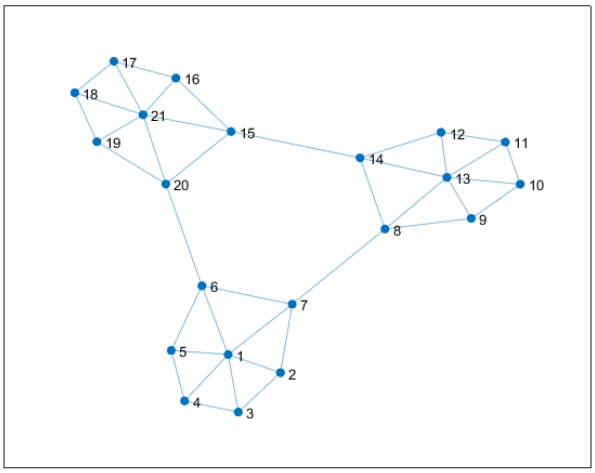
Community of 65 is composed by 40 nodes and communities of 45 and 6 are both composed by 30 nodes. In this case the structure in three blocks such that $|C_1| = 30$, $|C_2| = 30$ and $|C_3| = 40$ is found correctly.

2.5 Grover Quantum Walk to detect communities

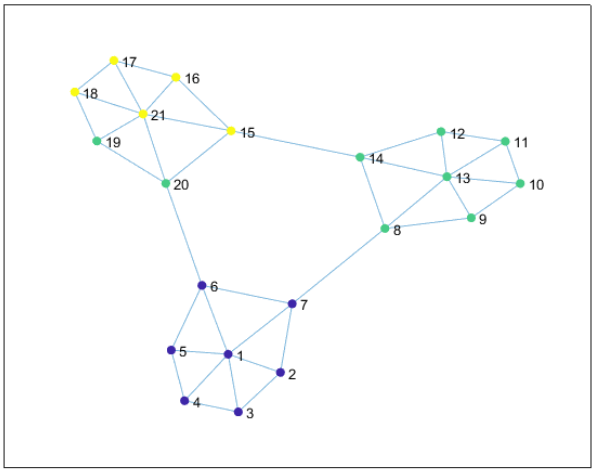
In the community detection algorithm we used until now, we chose a Fourier matrix as coin operator. Keeping the same algorithm, we can change our choice for the coin operator using a different matrix: the Grover matrix. This matrix is the following one:

$$C_G = \frac{1}{k_i} \begin{pmatrix} 2-k_i & 2 & 2 & \dots & 2 \\ 2 & 2-k_i & 2 & \dots & 2 \\ 2 & 2 & 2-k_i & \dots & 2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2 & 2 & 2 & \dots & 2-k_i \end{pmatrix},$$

remember that k_i is the degree of the i-th node of the graph.
Consider the graph we used for the first test of the algorithm with the Fourier matrix:



Repeat the test using the Grover matrix as coin operator. The results that we obtain are a bit worse than those given by the Fourier Walk, indeed there are two nodes associated with the wrong community:



The hubs found are still nodes 1, 13 and 21, but the community structure revealed by the values of $\overline{P(i \rightsquigarrow \ell)}$ is a bit different:

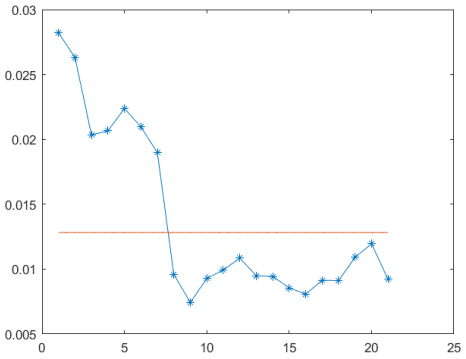


Figure 2.51: $\overline{P(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 3, 4, 5, 6, 7$

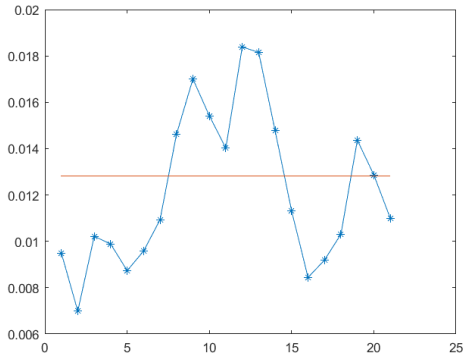


Figure 2.52: $\overline{P(13 \rightsquigarrow \ell)}$,
Nodes: $8, 9, 10, 11, 12, 13, 14, 19, 20$

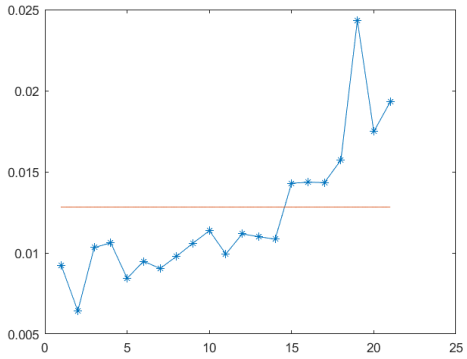
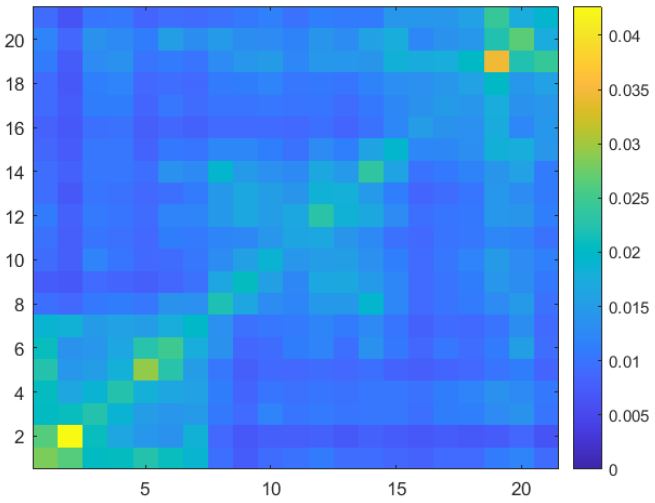


Figure 2.53: $\overline{P(21 \rightsquigarrow \ell)}$,
Nodes: $15, 16, 17, 18, 21$

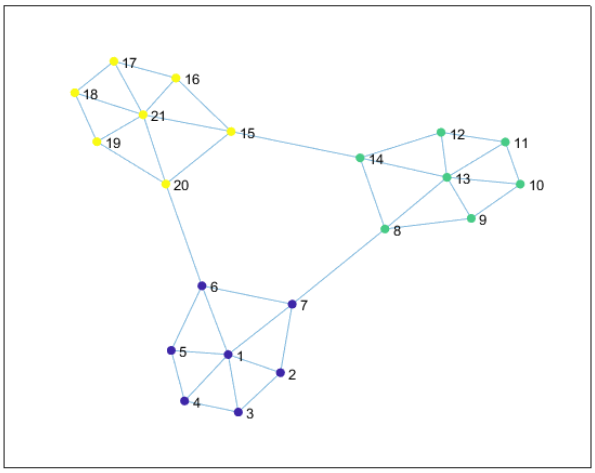
We can see that the values of $\overline{P(13 \rightsquigarrow 19)}$ and $\overline{P(13 \rightsquigarrow 20)}$ are over the threshold $q = \frac{1}{D}$, but that the values of $\overline{P(21 \rightsquigarrow 19)}$ and $\overline{P(21 \rightsquigarrow 20)}$ are over the threshold too. Nodes 19 and 20 are listed in the community of hub 13. This happens because the hubs 1, 13 and 21 have the same degree, so the function `sort` leave them in their indexing order, so the community of 13 is analyzed before the community of 21.

The output matrix of the infinite-time averages of the normalized probability gives:



so we can notice that the organization of the nodes into three communities is not so evident in the Grover Walk.

In the Fourier Quantum Walk we applied the algorithm also in the variant that uses the finite-time average of the probability to detect communities. So we replicate also this test to see what happens with the Grover Quantum Walk. Set $T = 100$, the community structure we obtain is the expected one:



with the usual hubs 1, 13 and 21. The values of the finite-time average of the normalized probability in the hubs are the following:

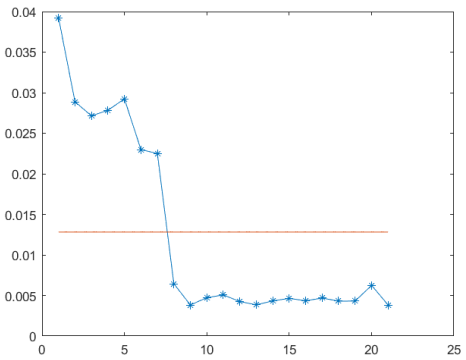


Figure 2.54: $\overline{P_{100}(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 3, 4, 5, 6, 7$

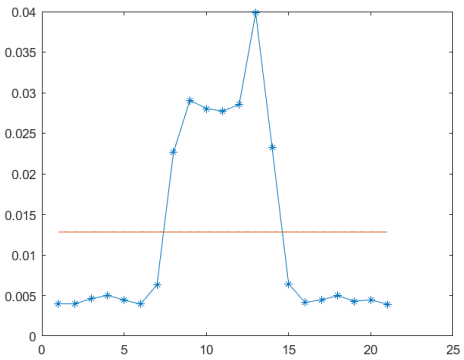


Figure 2.55: $\overline{P_{100}(13 \rightsquigarrow \ell)}$,
Nodes: $8, 9, 10, 11, 12, 13, 14$

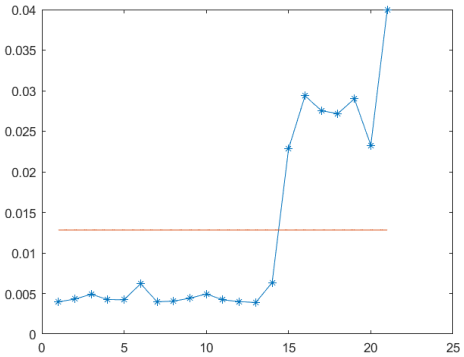
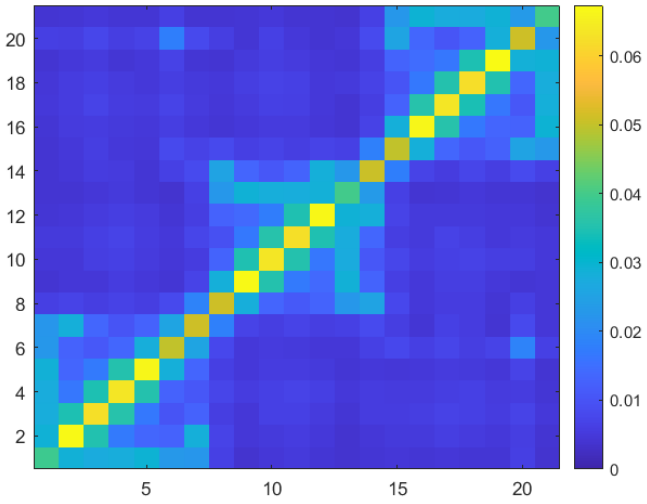


Figure 2.56: $\overline{P_{100}(21 \rightsquigarrow \ell)}$,
Nodes: $15, 16, 17, 18, 19, 20, 21$

The consequence of taking the average of the probability on a finite number of steps is that now the values related to the hub 13 for the nodes 19 and 20 are lower, so they are under the threshold q and the communities are composed by the right nodes.

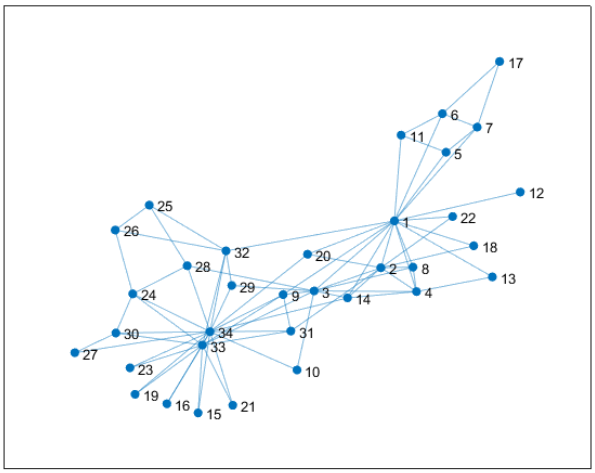
The output matrix of the finite-time averages of the normalized probability gives:



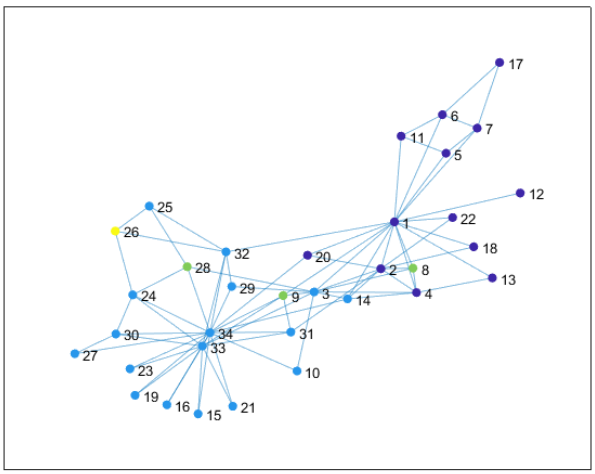
In this case the three communities are clearly recognizable, the values of $\overline{P_{100}(i \rightsquigarrow \ell)}$ are greater between nodes i and ℓ of the same community.

2.5.1 Zachary's karate-club network

We can test the algorithm with the Grover matrix for the karate-club matrix too and study how it works. The graph is the following one:



In this case we obtain four hubs: 34, 1, 3 and 32, so two more than the two hubs 34 and 1 found using the Fourier walk. The corresponding communities are the following ones:



The community structure revealed by the values of $\overline{P(i \rightsquigarrow \ell)}$ is the following:

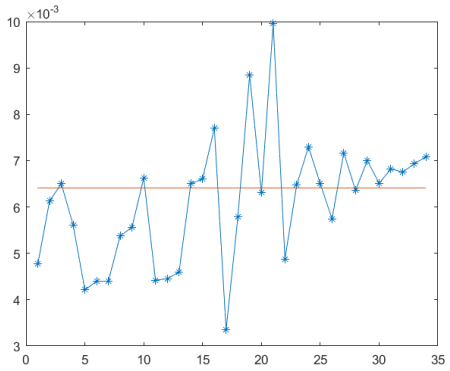


Figure 2.57: $\overline{P(34 \rightsquigarrow \ell)}$,
Nodes: $3, 10, 14, 15, 16, 19, 21, 23, 24, 25, 27, 29, 30, 31, 32, 33, 34$

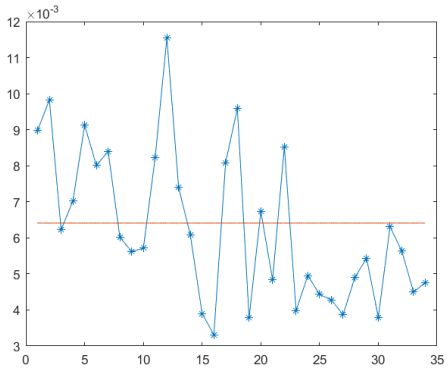


Figure 2.58: $\overline{P(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 4, 5, 6, 7, 11, 12, 13, 17, 18, 20, 22$

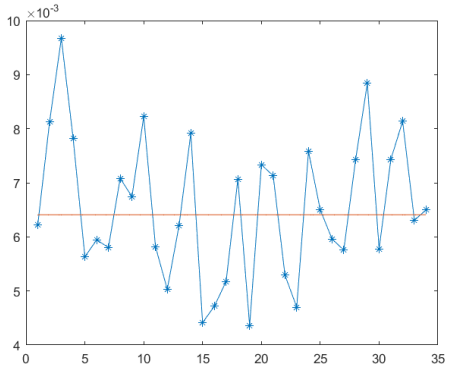


Figure 2.59: $\overline{P(3 \rightsquigarrow \ell)}$,
Nodes: $8, 9, 28$

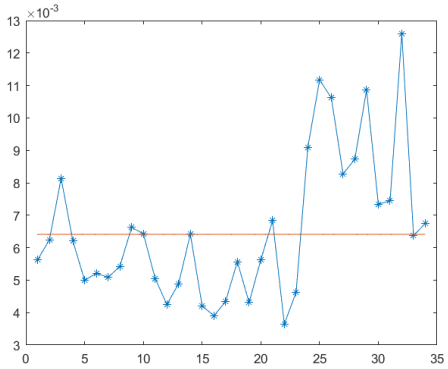
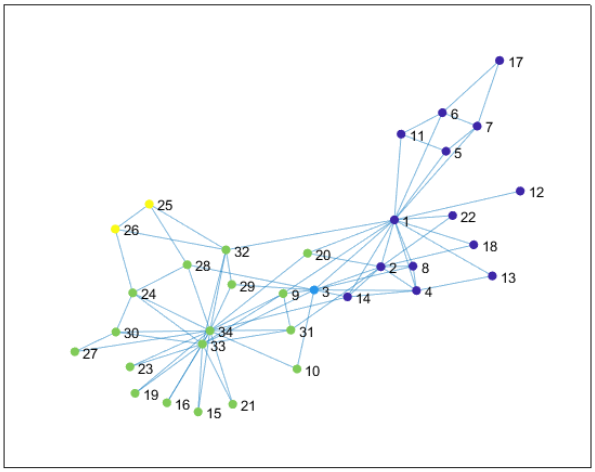


Figure 2.60: $\overline{P(32 \rightsquigarrow \ell)}$,
Nodes: 26

Notice that the two additional communities obtained are small, the one with hub 3 is composed by three nodes and the one with hub 32 by only one node, so they are not so relevant.

Now apply the algorithm using the average of the probability on a finite number of steps. Set $T = 100$, the hubs obtained are still 34, 1, 3 and 32 and the community structure is the following:



The values of the finite-time averages of the normalized probability in the hubs give the grouping of the nodes in the following way:

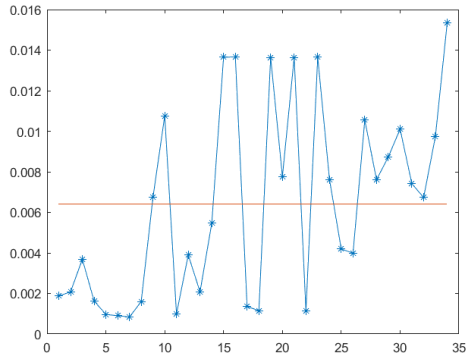


Figure 2.61: $\overline{P_{100}(34 \rightsquigarrow \ell)}$,
Nodes: $9, 10, 15, 16, 19, 20, 21, 23, 24, 27, 28, 29, 30, 31, 32, 33, 34$

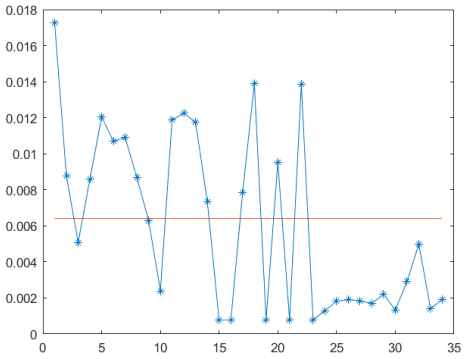


Figure 2.62: $\overline{P_{100}(1 \rightsquigarrow \ell)}$,
Nodes: $1, 2, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 22$

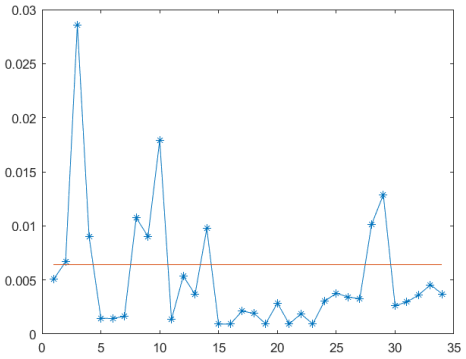


Figure 2.63: $\overline{P_{100}(3 \rightsquigarrow \ell)}$, Nodes: 3

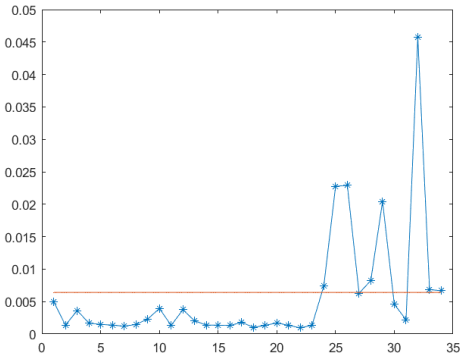


Figure 2.64: $\overline{P_{100}(32 \rightsquigarrow \ell)}$, Nodes: $25, 26$

In this case too the two additional communities are not so relevant and composed by a small number of nodes.

Chapter 3

Further analysis on community detection technique

In this chapter we will analyse some details and modifications of the community detection algorithm. In the previous chapter we mostly performed community detection via an algorithm based on Fourier Walk. At the end of the chapter we introduced the use of the same algorithm but with a different coin operator, the Grover matrix. Here we make a comparison between the choice of the Grover matrix rather than the Fourier matrix and see which one is more suitable for our algorithm. Then we analyse the choice of the degree as a criterion to identify the hubs and we try another measure of centrality, based on the exponential of the adjacency matrix. In the end we show how the most common clustering methods work compared to our algorithm.

3.1 Fourier VS Grover

A question that naturally arises is about the differences between the Fourier Quantum Walk and the Grover Quantum Walk. Compare the two algorithms in order to spot how the two different matrices affect them.
We use the result of the theorem 2 for the infinite-time average of the normalized probability in both the algorithms:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}p(i\rightsquigarrow\ell,t)=\frac{1}{k_i}\sum_{r=1}^{D}\sum_{s=1}^{k_\ell}\sum_{h=1}^{k_i}|\langle\ell\to m_s|\mu_r\rangle|^2\,|\langle\mu_r|i\to j_h\rangle|^2\,.$$

We take the graph $G=(V,E)$, with $|V|=21$, whose adjacency matrix is:

```
A=[ 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0;
    1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1;
    0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0
```

and consider a quantum walk on G using once the Fourier operator and another time the Grover operator.
In the computation of the limiting distribution we used the property:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} e^{i(\theta_{\mu_r} - \theta_{\mu_u})t} = \delta_{\mu_r \mu_u} \ .$$

The eigenvalues of the time-evolution operator U are differently distributed for the two algorithms:

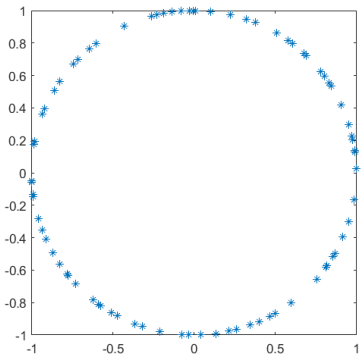


Figure 3.1: Fourier Quantum Walk

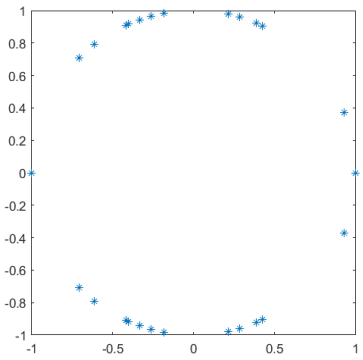


Figure 3.2: Grover Quantum Walk

The eigenvalues of U for the Fourier quantum walk are non-degenerate, while almost half of the eigenvalues for the Grover quantum walk are degenerate at 1 or -1. According to literature [14], this can explain why the Fourier operator is more suitable for the formulation that uses the limiting distribution than the Grover operator.

Our attempt is to explain the preferability of the Fourier operator than the Grover one using a different argument. We run some tests to study the convergence of the sequence of the time average of the normalized probability to their limiting distribution, that is the infinite-time average. These averages depend on the time evolution operator, so the behaviours of the two sequences would be different.

For both algorithms we compute the matrix of the infinite-time averages of the probability:

$$P_{n,F} = \left(\overline{P_F(i \rightsquigarrow \ell)}\right)_{i,\ell=1,\dots,21} \qquad P_{n,G} = \left(\overline{P_G(i \rightsquigarrow \ell)}\right)_{i,\ell=1,\dots,21},$$

where $\overline{P_F(i \rightsquigarrow \ell)}$ is the infinite-time average of the normalized probability obtained using the Fourier matrix and $\overline{P_G(i \rightsquigarrow \ell)}$ is the one obtained using the Grover matrix.
Compute also the matrix of the finite-time average over T steps:

$$P_{f,F} = \left(\overline{P_{T,F}(i \rightsquigarrow \ell)}\right)_{i,\ell=1,\dots,21} \qquad P_{f,G} = \left(\overline{P_{T,F}(i \rightsquigarrow \ell)}\right)_{i,\ell=1,\dots,21}.$$

Then we study the trend of the norm of the difference between them:

$$\|P_{n,F} - P_{f,F}\| \qquad \|P_{n,G} - P_{f,G}\|$$

for different values of T: $T = 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300$.

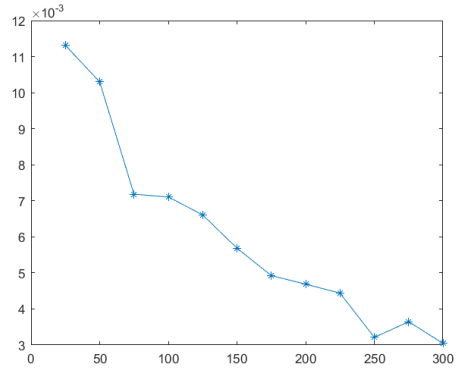


Figure 3.3: Fourier Quantum Walk

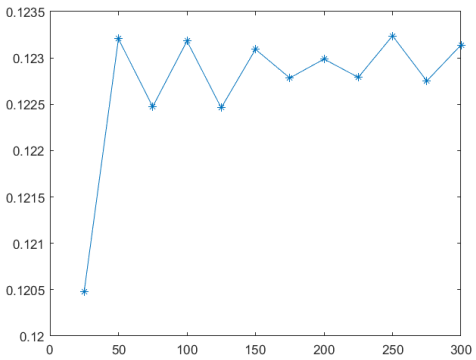


Figure 3.4: Grover Quantum Walk

T	$\|P_{n,F} - P_{f,F}\|$	$\|P_{n,G} - P_{f,G}\|$
25	0.0113123484274876	0.120479119271516
50	0.0102969706819012	0.123205807800115
75	0.00718365373827612	0.122473064925418
100	0.00710635608240139	0.123181816661274
125	0.00660787924686509	0.122461532617391
150	0.00568548661133431	0.123092521300192
175	0.00492577483929084	0.122783602090333
200	0.00468684743635560	0.122987563148248
225	0.00443982102874143	0.122789982513998
250	0.00321227066420356	0.123238108966401
275	0.00364726457604211	0.122751125349677
300	0.00304497474477164	0.123139611043830

We can immediately notice that using the Fourier operator there is an evident convergence, while using the Grover operator the convergence is hardly detectable.
A more precise analysis is given by the behaviour of the sequence of the time average of the probability related to a single couple (i, j) of nodes. Setting an initial node i, we know that the sequence of finite-time averages over T steps of the probability to be on a node ℓ converges to the limiting distribution as T approaches infinity. For example, choose $i = 13$, one of the hubs, and $\ell = 11$, a node in its community, and see the trend of the sequence for both the Fourier and the Grover Walk. Compute the values of the infinite-time average:

$$\overline{P_F(13 \rightsquigarrow 11)} = 0.0255882202661871,$$

$$\overline{P_G(13 \rightsquigarrow 11)} = 0.0140192664905204,$$

and the values of the finite-time averages $\overline{P_{T,F}(13 \rightsquigarrow 11)}$ and $\overline{P_{T,G}(13 \rightsquigarrow 11)}$ for $T = 25$, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300. Then compute the difference between the values of the sequence and the limiting distribution:

$$\left|\overline{P_F(13 \rightsquigarrow 11)} - \overline{P_{T,F}(13 \rightsquigarrow 11)}\right| \qquad \left|\overline{P_G(13 \rightsquigarrow 11)} - \overline{P_{T,G}(13 \rightsquigarrow 11)}\right|$$

and see the trend of the convergence:

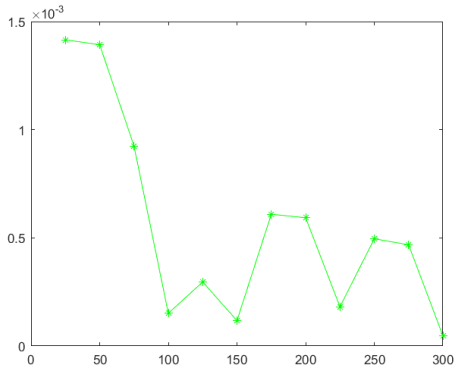


Figure 3.5: Fourier Quantum Walk

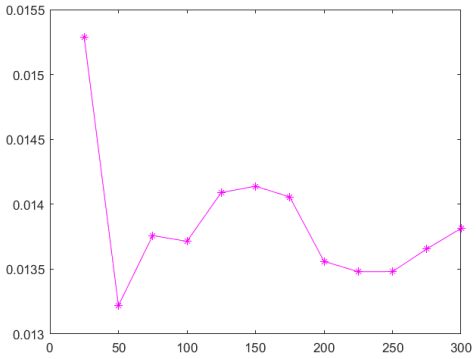
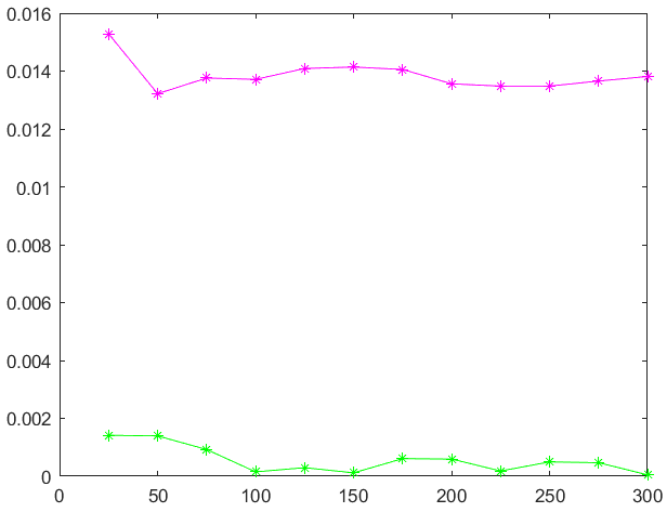


Figure 3.6: Grover Quantum Walk

T	$\lvert\overline{P_F(13 \rightsquigarrow 11)} - \overline{P_{T,F}(13 \rightsquigarrow 11)}\rvert$	$\lvert\overline{P_G(13 \rightsquigarrow 11)} - \overline{P_{T,G}(13 \rightsquigarrow 11)}\rvert$
25	0.0014158	0.015288
50	0.0013929	0.013219
75	0.0009245	0.013759
100	0.00015112	0.013714
125	0.00029628	0.01409
150	0.00011753	0.014139
175	0.00060931	0.014056
200	0.0005931	0.01356
225	0.00018105	0.01348
250	0.00049581	0.013481
275	0.00046786	0.013656
300	4.673e-05	0.013814

The most relevant thing to observe are the different orders of magnitude of the errors for the two algorithms:



Now represent the values of the infinite-time average of the probability $\overline{P(i \rightsquigarrow \ell)}$ for the hubs of the communities, so for $i = 1, 13, 21$, and of the finite-time average $\overline{P_T(i \rightsquigarrow \ell)}$, for $i = 1, 13, 21$ and $T = 25, 50, 100, 150, 200$, in order to see how the values of the sequence become closer to the infinite-time average as T increases. For $i = 1$, using the Fourier operator, we can see that the values of the finite-time average (in blue) become closer to the values of the infinite-time average (in red) as T increases:

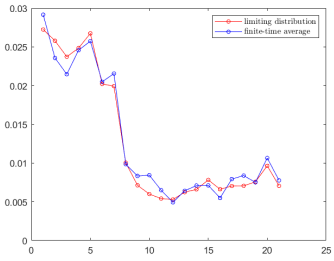


Figure 3.7: $T = 25$

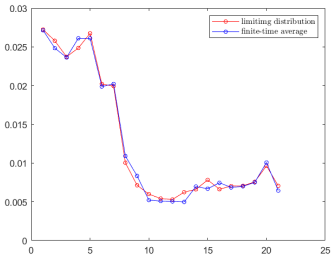


Figure 3.8: $T = 50$

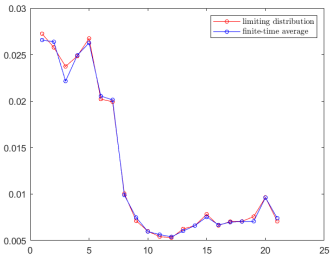


Figure 3.9: $T = 100$

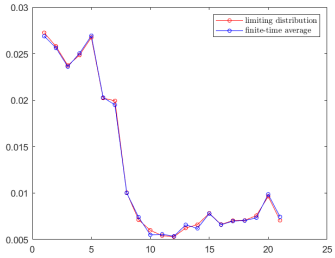


Figure 3.10: $T = 150$

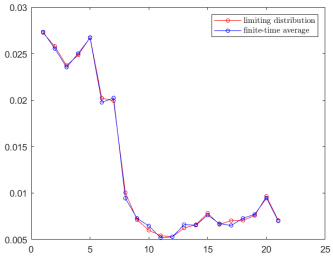


Figure 3.11: $T = 200$

While, using the Grover operator, the values seem to be at the same distance for all the values of T:

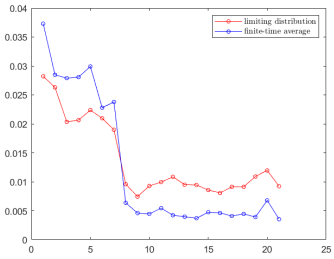


Figure 3.12: $T = 25$

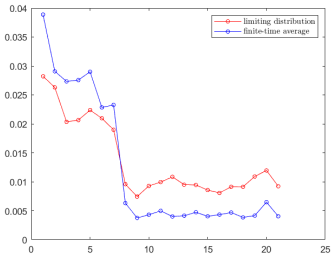


Figure 3.13: $T = 50$

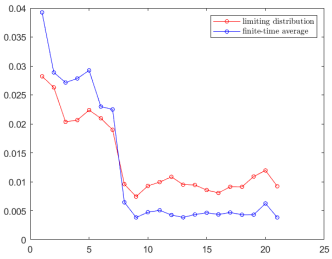


Figure 3.14: $T = 100$

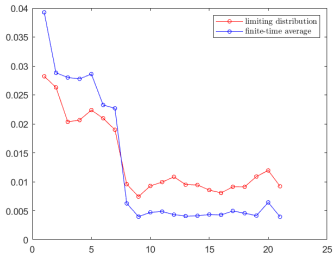


Figure 3.15: $T = 150$

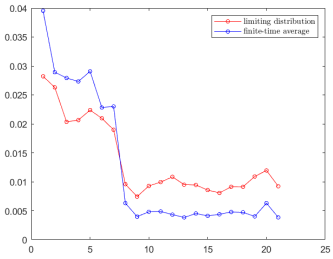


Figure 3.16: $T = 200$

The same observations are still valid for $i = 13$. Using the Fourier operator we have the following trend:

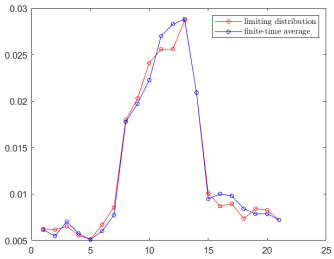


Figure 3.17: $T = 25$

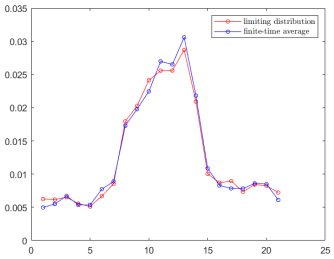


Figure 3.18: $T = 50$

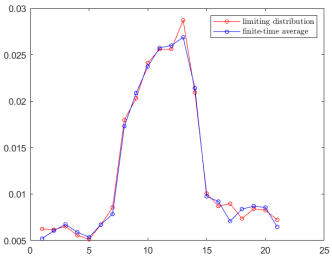


Figure 3.19: $T = 100$

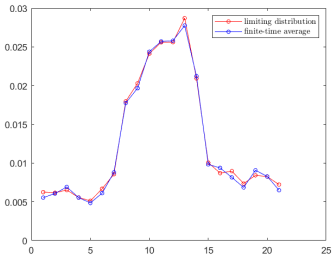


Figure 3.20: $T = 150$

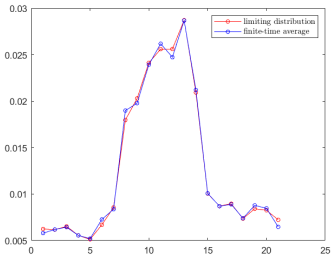


Figure 3.21: $T = 200$

while using the Grover operator we have a very slower trend:

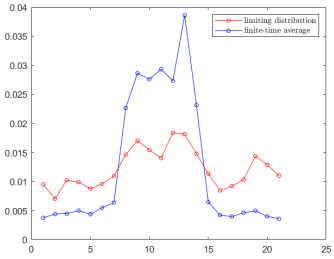


Figure 3.22: $T = 25$

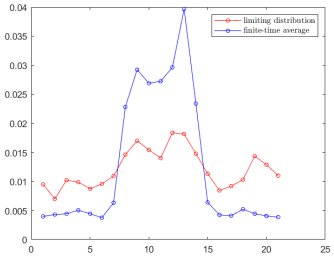


Figure 3.23: $T = 50$

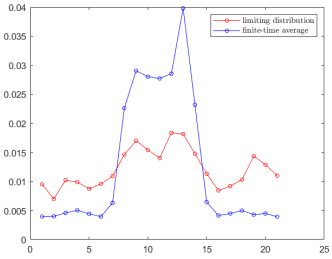


Figure 3.24: $T = 100$

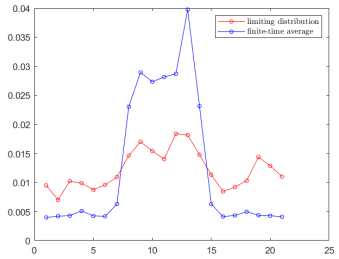


Figure 3.25: $T = 150$

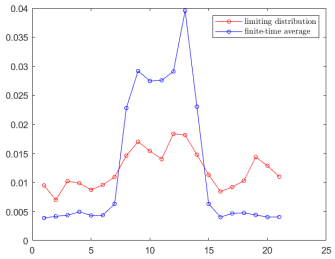


Figure 3.26: $T = 200$

The same happens also for $i = 21$, so using the Fourier operator:

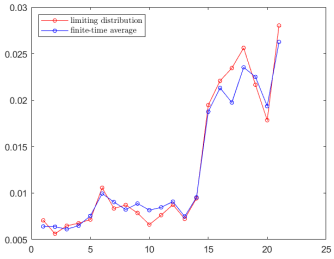


Figure 3.27: $T = 25$

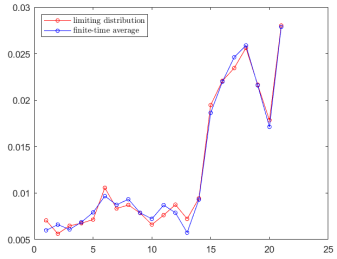


Figure 3.28: $T = 50$

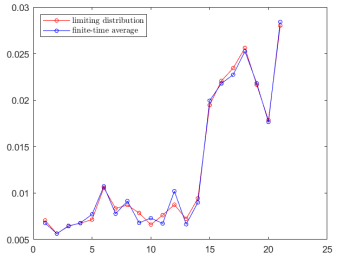


Figure 3.29: $T = 100$

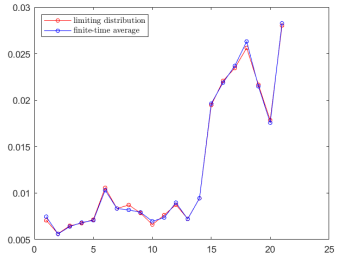


Figure 3.30: $T = 150$

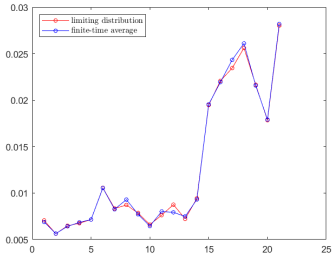


Figure 3.31: $T = 200$

and using the Grover operator:

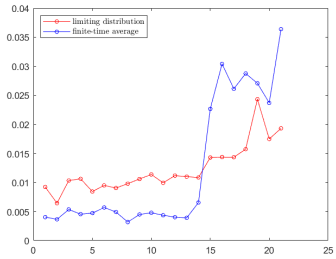


Figure 3.32: $T = 25$

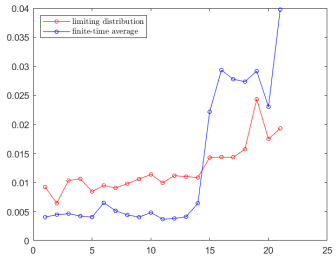


Figure 3.33: $T = 50$

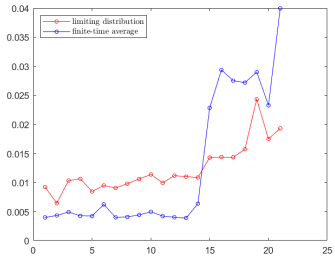


Figure 3.34: $T = 100$

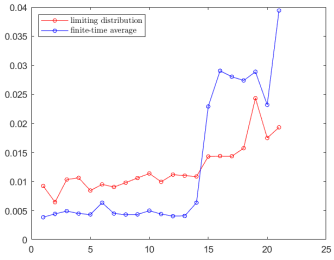


Figure 3.35: $T = 150$

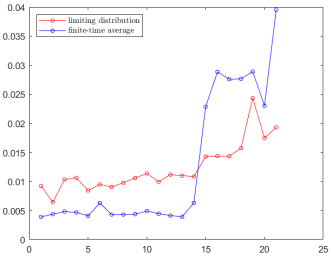


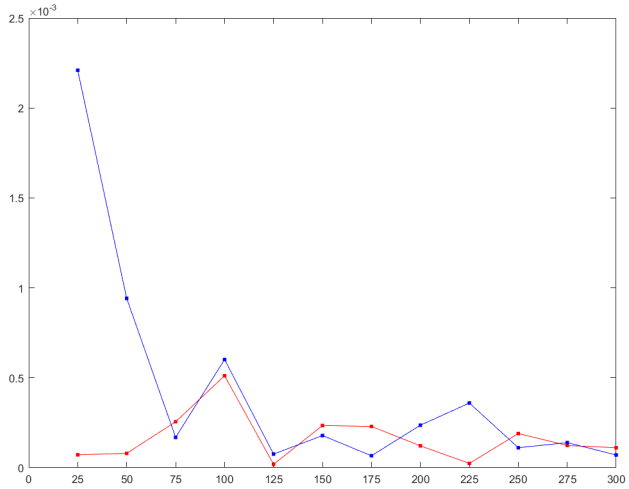
Figure 3.36: $T = 200$

3.1.1 Convergence of the Fourier sequence

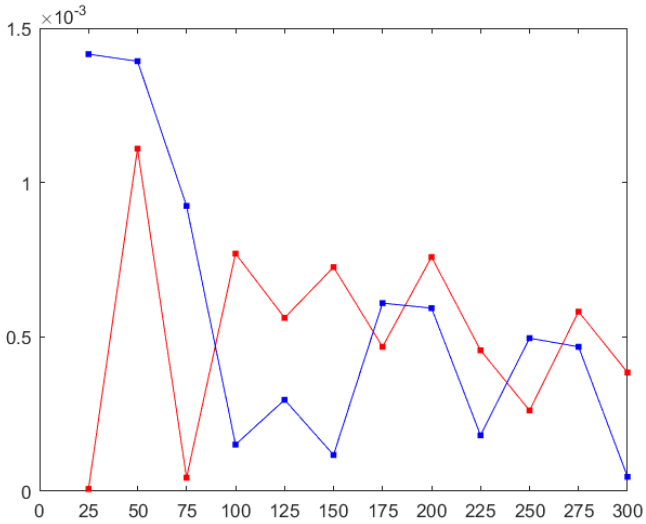
Looking at the Fourier graphs we notice that some points are often closer to the ones of the limiting distribution. This could mean that the finite-time averages: $\overline{P_T(1 \rightsquigarrow \ell)}$, $\overline{P_T(13 \rightsquigarrow \ell)}$ and $\overline{P_T(21 \rightsquigarrow \ell)}$ for some values of ℓ are "more convergent" to the respective infinite-time averages. To find which ones behave this way we compute the difference:

$$\|P_n - P_f\| = \left\| \left(\overline{P(i \rightsquigarrow \ell)}\right)_{i,\ell} - \left(\overline{P_T(i \rightsquigarrow \ell)}\right)_{i,\ell} \right\|$$

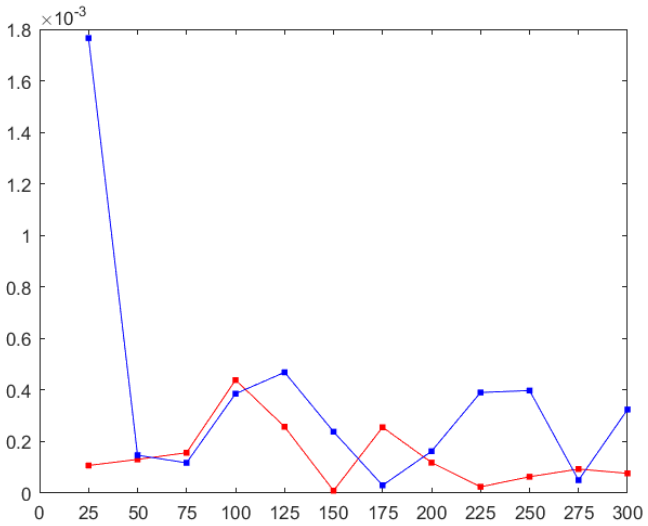
for values of T from 25 to 300, at steps of 25. Then we select only the rows for i equal to 1, 13 and 21 and sort them in ascending order. In this way, for each $i = 1, 13, 21$, we are able to find the nodes ℓ which errors $\|\overline{P(i \rightsquigarrow \ell)} - \overline{P_T(i \rightsquigarrow \ell)}\|$ occur more frequently among the lowest, for different values of T, looking at the ones on the first three positions of the sorted rows. We are also able to find the nodes which errors occur more frequently among the highest, looking at the last three positions. For the hub 1, the "most convergent" node is 19, while the "least convergent" is 2. Plot in red the values of $\|\overline{P(1 \rightsquigarrow 19)} - \overline{P_T(1 \rightsquigarrow 19)}\|$ and in blue the values of $\|\overline{P(1 \rightsquigarrow 2)} - \overline{P_T(1 \rightsquigarrow 2)}\|$, for $T = 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300$:



For the hub 13, the "most convergent" node is 21, while the "least convergent" is 11. Plot in red the values of $\|\overline{P(1 \rightsquigarrow 19)} - \overline{P_T(1 \rightsquigarrow 21)}\|$ and in blue the values of $\|\overline{P(1 \rightsquigarrow 11)} - \overline{P_T(1 \rightsquigarrow 2)}\|$:



For the hub 21, the "most convergent" node is 14, while the "least convergent" is 21. Plot in red the values of $\|\overline{P(1 \rightsquigarrow 14)} - \overline{P_T(1 \rightsquigarrow 19)}\|$ and in blue the values of $\|\overline{P(1 \rightsquigarrow 21)} - \overline{P_T(1 \rightsquigarrow 2)}\|$:



After this analysis, the results obtained are not so satisfying. We can now say that our guess that, for some nodes, the values of the finite-time averages $\overline{P_T(i \rightsquigarrow \ell)}$, $i = 1, 13, 21$, converge rapidly to the correspondent infinite-time averages is not so true. The most important thing to point out is that the sequence of the finite-time averages of the probability is not monotone. This implies that the behaviour of its points is not regular, so it may happen that the finite-time average of the probability gets closer to the limiting distribution but then can get more distant.

3.2 Measures of centrality

In the community detection a measure of centrality is necessary to quantify how "well-connected" nodes are in a networked system. In the algorithm we used until now, nodes are sorted for decreasing degree, so the candidates to be hubs are the nodes with higher degree.
Consider a graph $G = (V, E)$ with $|V| = N$ and adjacency matrix A [7]. Its squared power A^2 has the diagonal composed by the following elements:

$$(A^2)_{ii} = \sum_{j=1}^{N} a_{ij} a_{ji} = \deg(i), \qquad i = 1, \dots, N.$$

This shows that the degree can be characterized through the diagonal of the squared power A^2. The powers A^k, for $k \in \mathbb{N}$, have on their diagonal the elements $(A^k)_{ii}$. For any $i = 1, \dots, N$ the element $(A^k)_{ii}$ is the number of closed walks of length k based on node i. This implies that $\mathrm{diag}(A^2)$, i.e. the degrees, is not the only power of A to provide useful information, but every A^k, for different values of k, brings a meaningful piece of

information.
In order to detect communities, shorter walks are more important, so we can consider the length-based weighted average:

$$\operatorname{diag}\left(\frac{A^2}{2!}+\frac{A^3}{3!}+\cdots+\frac{A^k}{k!}+\dots\right)$$

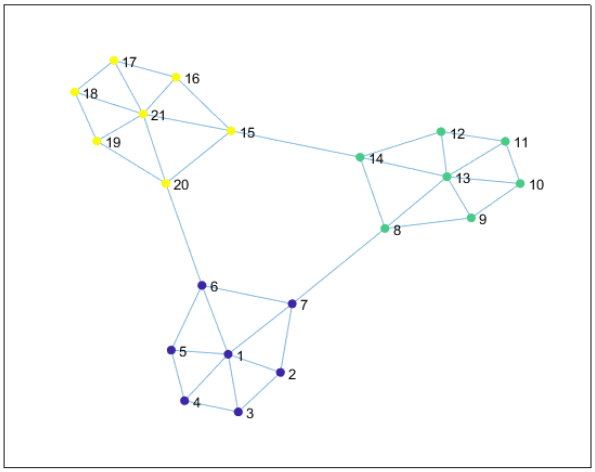
We can bring this to a known form adding the identity matrix I, indeed the relative ordering does not change adding a constant, and we can also add A, indeed we consider only the diagonal elements and they are equal to 0 for the adjacency matrix A. So we take:

$$\operatorname{diag}\left(I+A+\frac{A^2}{2!}+\frac{A^3}{3!}+\cdots+\frac{A^k}{k!}+\dots\right)=\operatorname{diag}\left(\exp(A)\right),$$

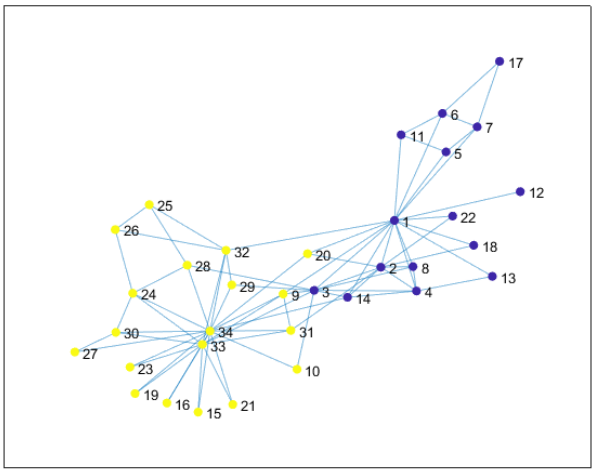
exploiting the fact that this sum is now known, that is the exponential of a matrix.
Here we would see how the previous results change choosing as measure of centrality the matrix exponential instead of the degree. Take the usual synthetic the graph G of 21 nodes whose adjacency matrix is:

```
A=[ 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0;
    1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1;
    0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0
```

and compute the matrix exponential e^A. In the community detection algorithm, until now we sorted the nodes for decreasing degree to set the candidates to be the hubs. Now consider the value $(\operatorname{diag}(e^A))_i$ for any node i and sort the nodes in decreasing order, according to these values. The first nodes would be the best placed to be the hubs of the communities. For this network, the outcoming communities are three, and the first three nodes of the sorting, 1, 13 and 21, are their hubs. The structure obtained is the usual one:



Now test the algorithm with this measure of centrality on other graphs, for example on the Zachary's karate-club network. In this case too the hubs are the same then before, 1 and 34, and the community structure is the usual:



Repeat the experimentation also on the food web of marine organisms living in the Chesapeake Bay. The resulting hubs are: 39, 36, 38, 22 and 35, so one more of the expected ones, the 22. Compare the community structure with the one obtained using the algorithm with nodes sorted according to their degree:

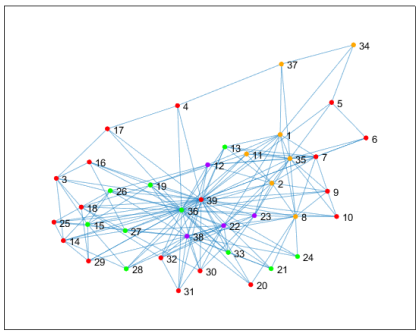


Figure 3.37: Decreasing values of the degree

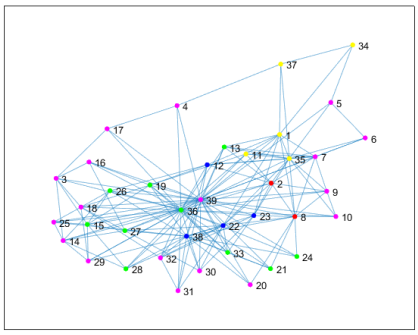


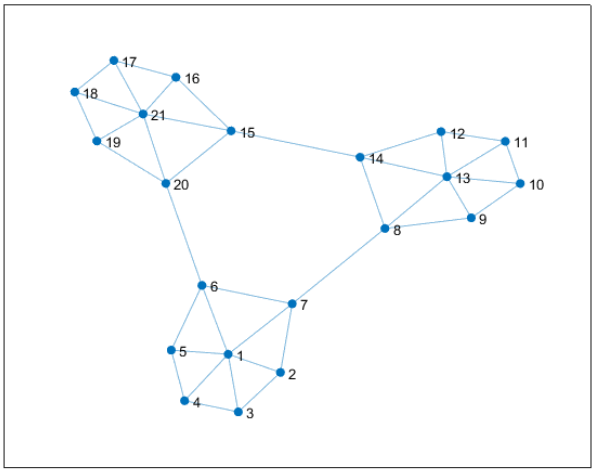
Figure 3.38: Decreasing values of $\mathrm{diag}(e^A)$

We can conclude that, for the examples seen, the algorithm in which nodes are sorted according to the diagonal of the adjacency matrix exponential works quite well. But the proposed graphs do not highlight significant differences due to the chosen sorting method. We can see a better analysis on networks with other characteristics, in order to compare the operation of the two measures of centrality.

3.2.1 Exponential VS Degree

An interesting point is that, using the degree as measure of centrality, we may have cases of more than one node with the same degree. When we run into a situation of equality, the MATLAB function `sort` keep the original ordering, so no meaningful criteria are applied. Using the exponential measure with these networks gives more significant information on the centrality of the nodes and can make the difference in the detection of the communities.

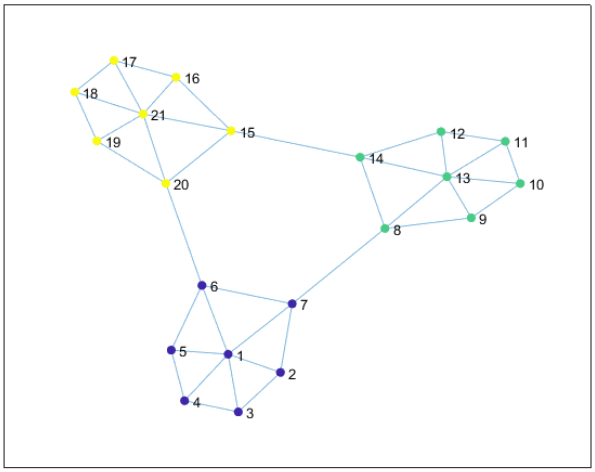
When we introduced the Grover quantum walk (2.5), we saw that the first test falls exactly into this case. The graph we consider is the following:



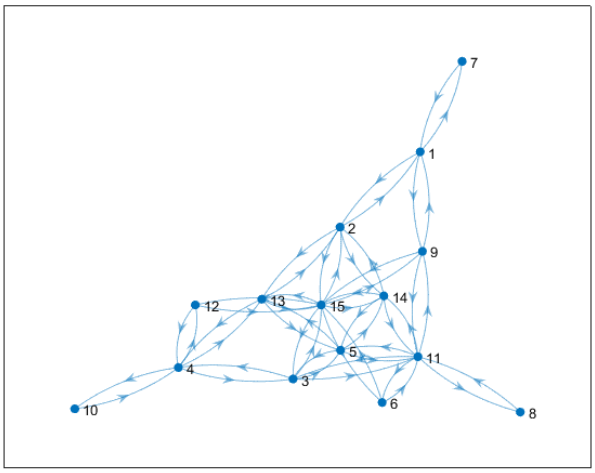
and the hubs found by our algorithm are the nodes: 1, 13 and 21, all with degree equal to 6. The function `sort` orders them following their index order, and we observed that this causes the detection of wrong communities. Repeat the test using the exponential measure instead of the degree to sort the nodes. The hubs found are the same three as before, but the values of diag(e^A) associated to them are different from each other:

Node i	Degree	$(e^A)_{ii}$
21	6	12.384812759051890
1	6	12.384812759051881
13	6	12.384812759051876

So the algorithm analyses the values of the average of the probability related to the hubs in a different order: 21, 1 and 13. Thanks to this, the community structure revealed is the expected one:



Compare now the two measures on another significant network. Consider the data set of the interactions between 15 mine workers in Zambia [22][7], collected by Bruce Kapferer:



The associated adjacency matrix is:

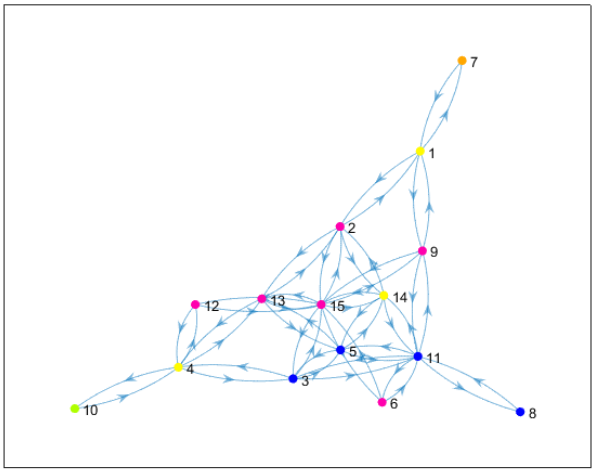
```
A=[0 1 0 0 0 0 1 0 1 0 0 0 0 0 0;
 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1;
 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1;
 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0;
 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0;
```

```
0 0 0 0 0 0 0 0 0 0 1 0 0 0 1;
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;
1 0 0 0 0 0 0 0 0 0 1 0 0 0 1;
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
0 0 1 0 1 1 0 1 1 0 0 0 0 1 0;
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1;
0 1 0 1 1 0 0 0 0 0 0 0 0 0 1;
0 1 0 0 1 0 0 0 0 0 1 0 0 0 1;
0 1 1 0 0 1 0 0 1 0 0 1 1 1 0];
```

There are different groups of nodes with the same degree:

Node	15	11	2	3	4	5	13	14	1	9	6	12	7	8	10
Degree	7	6	4	4	4	4	4	4	3	3	2	2	1	1	1

Six of the nodes have degree equal to 4, that are: 2, 3, 4, 5, 13 and 14, then nodes 1 and 9 have degree equal to 3, nodes 6 and 12 have degree equal to 2 and nodes 7, 8, and 10 have degree equal to 1. The hubs given by the sorting for descending degree are 15, 11, 2, 3 and 1. The nodes 2 and 3 both have degree equal to 4 and they are sorted according to their original ordering. The community structure resulting from the algorithm with the Fourier walk is the following:

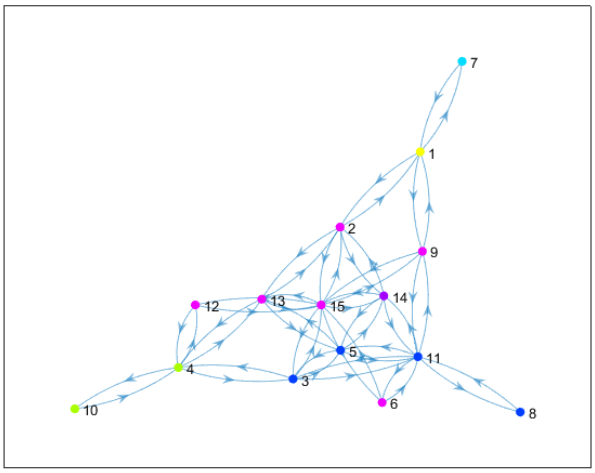


Now compute the exponential of the adjacency matrix and consider $\mathrm{diag}(e^A)$. In this case nodes are sorted in a different order because the values $(e^A)_{ii}$ associated to each of them give additional information and help to understand which ones are more well-connected. Among the nodes of degree 4, the best placed to become hubs are, in order,

14, 5, 3, 2, 13 and 4, that so are no more in their original order. The same happens for 1 and 9, indeed they are swapped. The nodes 6 and 12 keep their order and the nodes of degree equal to 1 are 8, 10 and 7, in order.
The values for each node are the following ones:

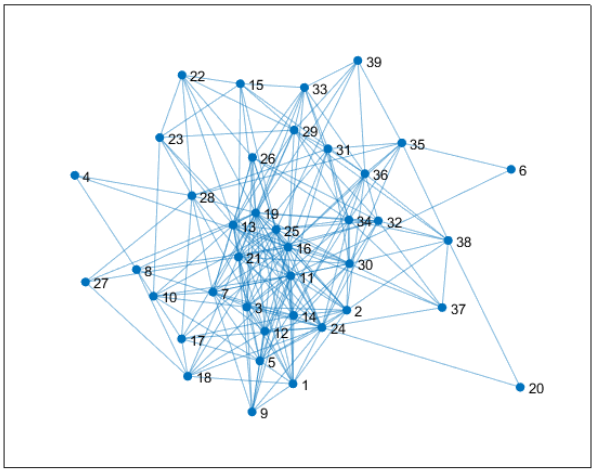
Node i	Degree	$(e^A)_{ii}$
15	7	13.3388924561932
11	6	10.2197287994260
14	4	8.40019924971407
5	4	7.82477238520549
3	4	7.67022191199137
2	4	7.28416384081634
13	4	7.26973652079921
4	4	5.13191229766952
9	3	4.82486611525844
6	2	3.65993426517097
1	3	3.57976905833066
12	2	3.20738138018764
8	1	1.92180548566626
10	1	1.72355485130338
7	1	1.65430072145483

Using the algorithm with the measure given by $\mathrm{diag}(e^A)$, the outcoming hubs are: 15, 11, 14, 3, 2 and 9. The community structure is the following:

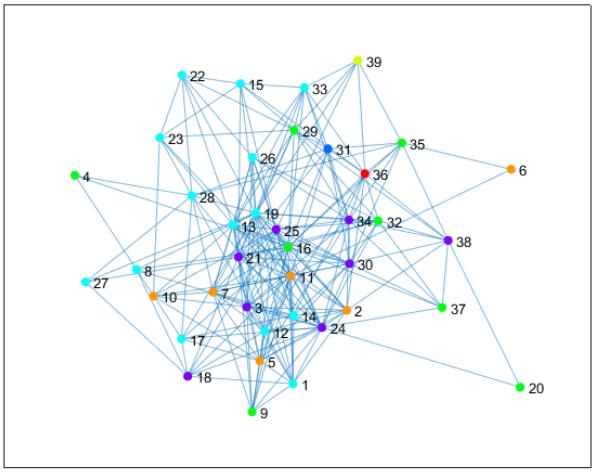


The communities of hubs 15 and 11 are identical to the previous ones: the community of 15 is composed by 2, 6, 9, 12, 13 and 15, the community of 11 by 3, 5, 8 and 11. Then there is an extra community, associated to 14, composed by only the hub itself. The community related to 3 is now analysed before the one of 2 and includes 4 and 10, while, using the degree, it is composed by only 10. The community of 2 is composed by only 1, while, using the degree, it is analyzed first and includes 1, 4 and 14. The community of 1 here is missing, while using the degree 1 is an hub and its community is composed by 7.

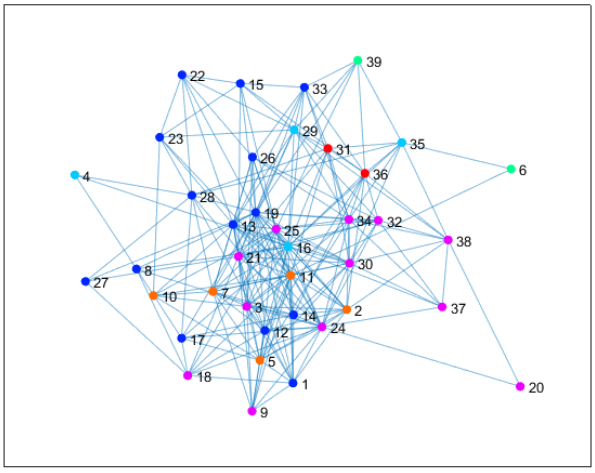
An additional experimentation can be made on another known example, that is a dataset that records assistance-related interactions between individuals in a tailor's shop in Zambia [6][7], also this one collected by Bruce Kapferer:



In this case too there are several nodes with the same degree, how we can see in the table 3.2.1 on page 100.
According to the sort by degree, there are seven hubs: 19, 16, 24, 11, 13, 21 and 30, and 21 and 30 both have degree 16. The communities resulting from the algorithm with the Fourier walk are the following ones:



Using the exponential measure, the hubs found are six, that are: 19, 24, 16, 3, 13 and 30. In this case, using a different measure of centrality, 24 that has degree 21 is analysed before 16, that has degree 22; then 3, with degree 17, is a new hub and is analysed before 13, that has degree 18; node 11, with degree 19, is no more a hub, and about the nodes with same degree, 30 is analyzed before of 21, that also is no more an hub. The community structure now is the following:



The community of 19 is the same as before, it is composed by 1, 8, 12, 13, 14, 15, 17, 19, 22, 23, 26, 27, 28 and 33. The community of 24 is similar to the previous one, that

is composed by 3, 18, 21, 24, 25, 30, 34 and 38, but, in addition to these, it includes also 9, 20, 32 and 37. The community of 16 is composed by 4, 16, 29 and 35, it is smaller than the one found using the degree, that includes also 9, 20, 32 and 37, indeed it is now analysed after the one of 24. The community of 3 is composed by 2, 5, 7, 10 and 11, while 3 is not identified as an hub by the algorithm with the degree. The community of 13 is composed by 6 and 39, while using the degree it is composed by only 39. The community of 30 is composed by 31 and 36, while using the degree it is composed by only 36, and 31 is the only member of the community of 21.

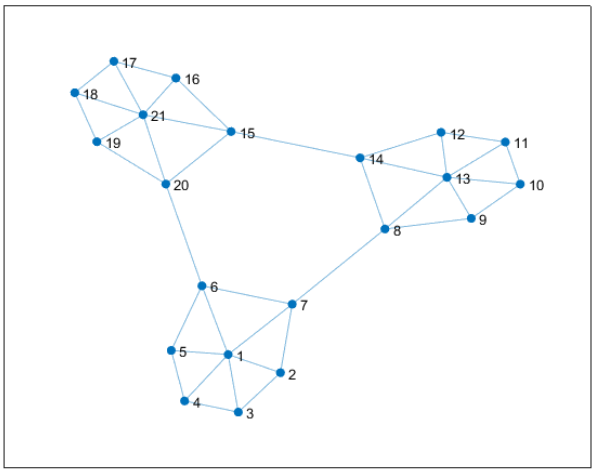
Node i	Degree	$(e^A)_{ii}$
19	25	108655.023752510
16	22	83760.8953482188
24	21	84440.2865696778
11	19	68278.8901353071
13	18	71340.5191722484
25	18	70231.1125523377
3	17	74218.8306446255
30	16	57972.6173042959
21	16	56611.8804772172
34	16	45914.3197365845
7	14	48228.2082108080
12	14	47099.6659372931
5	14	46732.7901795830
29	13	25260.2948379283
2	12	41393.0755130226
1	12	40159.9542733621
14	12	35507.0233394262
32	12	32153.4159002486
36	12	19029.1394388208
28	11	12251.4006242029

Node i	Degree	$(e^A)_{ii}$
26	10	25128.5120712691
31	10	17900.1615939333
33	10	17745.0657277747
9	9	24632.0897224738
18	9	15160.8515254164
38	9	10504.6976223352
35	9	9037.64539795951
17	8	13498.0240099727
15	8	10318.5479344727
10	7	12649.6841419166
22	7	7359.93819932281
23	7	6110.82470777183
37	6	8793.74430507391
8	6	7098.46531158677
39	6	6360.73247568621
27	4	3256.10510590951
4	3	1191.61091212501
20	2	785.279495532616
6	2	644.792659790420

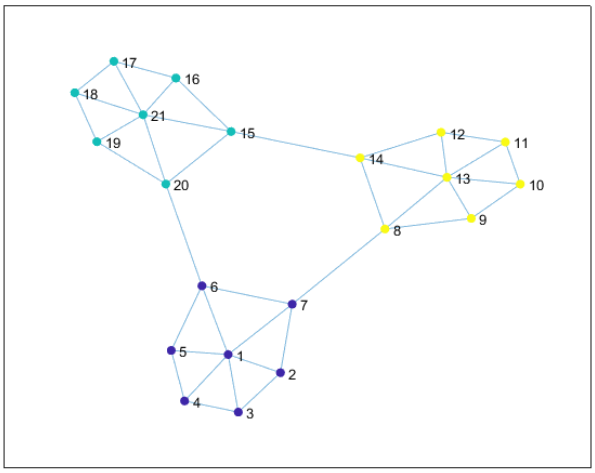
3.3 Clustering

The community detection is strictly connected to the cluster analysis, that is the task of grouping a set of objects in such a way that objects in the same group, called "cluster", are more similar to each other than to those in other groups [13, 20, 5]. Some of the most common clustering algorithms are the **k-means** and the **spectral clustering**. Our attempt here is to apply these algorithms in order to find a community structure in networks and compare the results with the ones obtained by the methods used before.
The **k-means** is a clustering algorithm that makes a partition of the observations into a pre-fixed number k of clusters, around the "centroids". It starts setting k initial centroids and then there is an iterative process. A step consists in associating each observation to the nearest centroid and then updating each centroid, since the assignments no longer change.
The **spectral clustering** is a clustering algorithm that exploits the spectrum of the similarity matrix of the data. Given a dataset, indeed, a similarity graph is associated to it. Each node corresponds to an observation and the weights of the links between the nodes are computed through a measure of their similarity, in this way we can define a similarity matrix S. Now we introduce the Laplacian matrix: $L = D - S$, where D is the diagonal matrix of the degrees of the nodes. Then the algorithm takes the eigenvectors corresponding to the k smallest eigenvalues of L as columns of a matrix V. Now a clustering method, such as the k-means, is applied to V.

We can see the behaviour of these clustering methods on the graph we used as first test for all the previous experimentations:



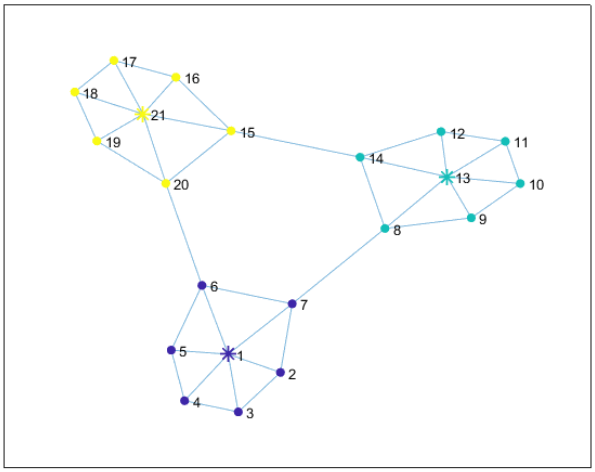
In order to apply the k-means algorithm we can use the appropriate MATLAB function as: `[idx,H] = kmeans(pn,k)`. It takes in input a matrix in which any row is an observation, in this case it is the matrix P_n of the infinite-time averages of the normalized probability, and the desired number of clusters k, here we set $k = 3$. The outputs are a vector `idx` that specifies for each node the cluster it belongs to and a matrix `H` which rows are the centroids. The communities obtained are the following:



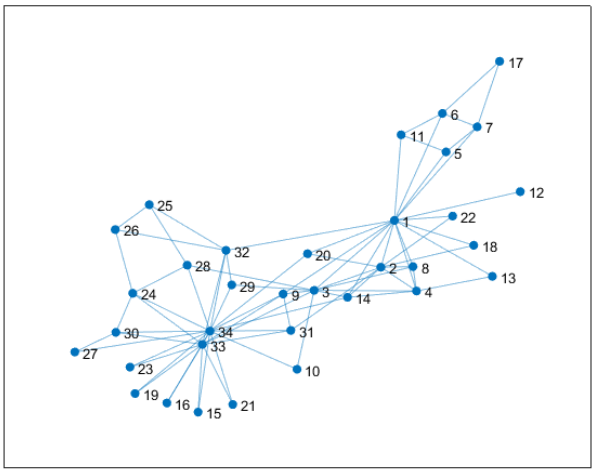
None of the three rows of the outcoming matrix `H` matches with a row of P_n, so the centroids obtained are not nodes of the graph. Computing the distances between any row of `H` and any $\overline{P(i \rightsquigarrow \ell)}$, as i changes, we find the three nodes i such that the corresponding $\overline{P(i \rightsquigarrow \ell)}$ vectors are the nearest ones to the rows of the centroids:

```
for s=1:k
    for i=1:N
        a(i)=norm(H(s,:)-pn(i,:));
        [m,c(s)]=min(a);
    end
end
```

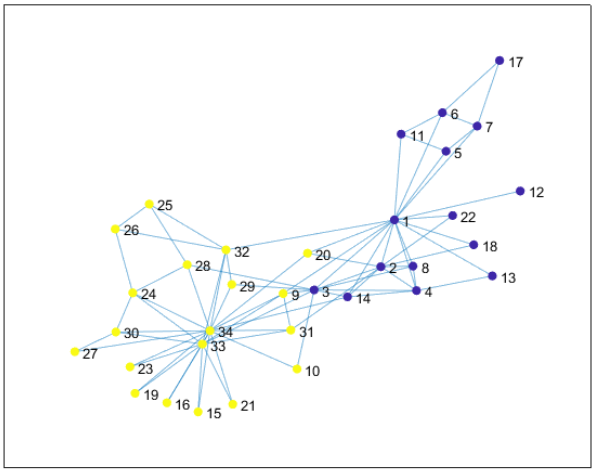
The resulting nodes are 1, 13 and 21, so exactly the usual hubs:



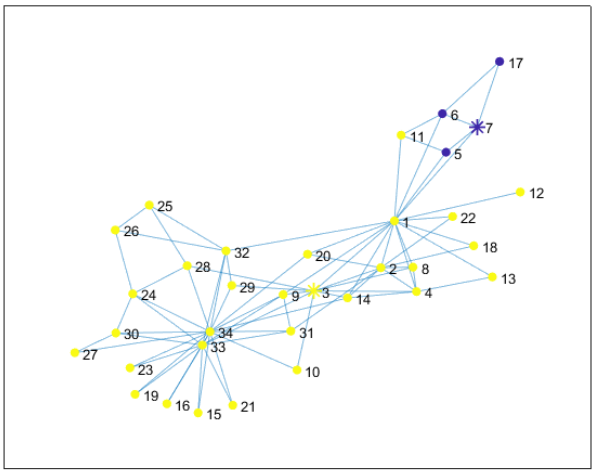
Then we apply the k-means algorithm also to the Zachary's karate-club network:



Previously, as result of the community detection algorithm based on the infinite-time average of the normalized probability, we found the following communities for this graph:



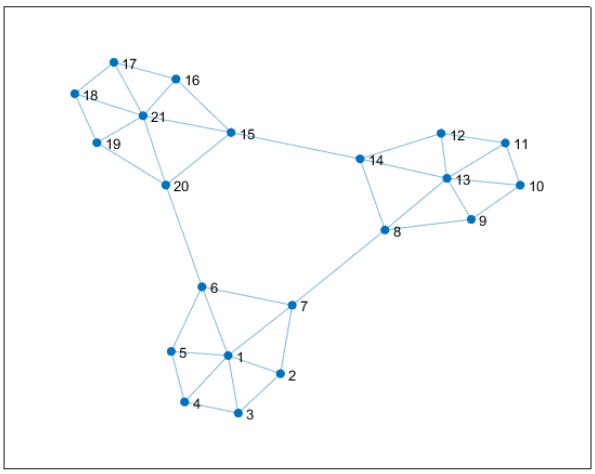
So we now set $k = 2$ to apply the `kmeans` function and the community structure we obtain is the following one:



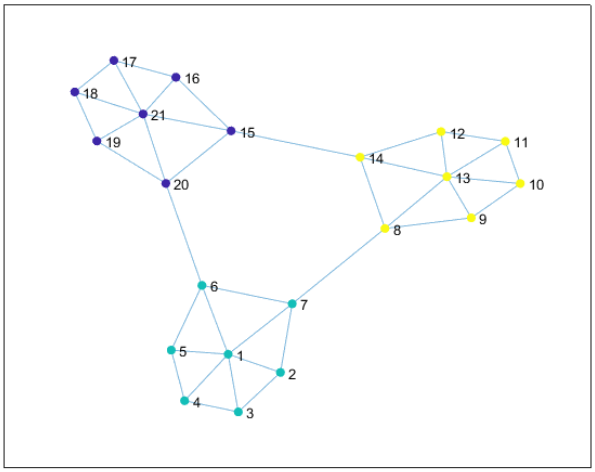
We can see that in this case the k-means does not detect the expected communities.

Now we repeat the same examples using the spectral clustering. Even in this case we can use a MATLAB function: `idx = spectralcluster(pn,k)`, that takes in input a matrix `pn` such that any row is an observation and the number `k` of clusters desired. It works considering a neighborhood for each point of the input matrix and computing the

distances between all the pairs into the neighborhood, then the distances are converted in measures of similarity and the Laplacian matrix L and the matrix of the first k eigenvectors V are computed; now the k-means is applied to V and the outcoming clusters are associated to observations of the input matrix. The output `idx` is a vector that specifies for each node the cluster it belongs to. In this case we use the spectral clustering function where we put the matrix P_n of the infinite-time averages of the normalized probability as input matrix.
At first we apply this function to the matrix P_n related to the synthetic graph:



so we set $k = 3$. The communities obtained are the expected ones:



Then we apply the same function to the Zachary's karate-club network, setting $k = 2$, and we obtain a community structure similar to the expected one. Comparing the one obtained with the one given by the algorithm based on the Fourier Walk, we can see that the only difference are that nodes 3 and 20 are swapped:

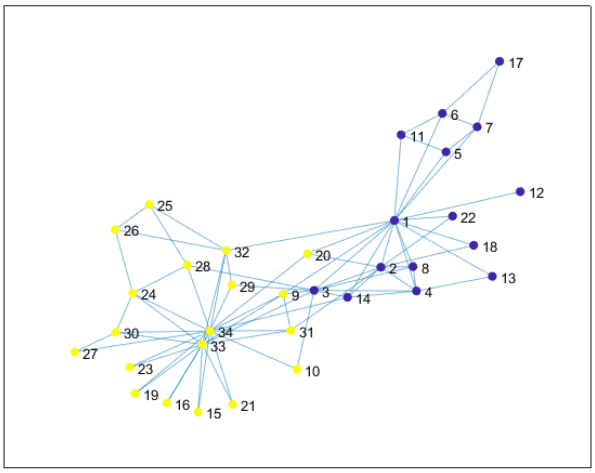


Figure 3.39: Community detection algorithm

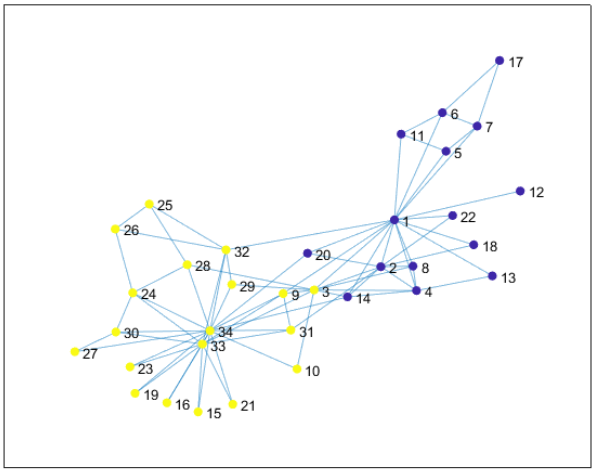
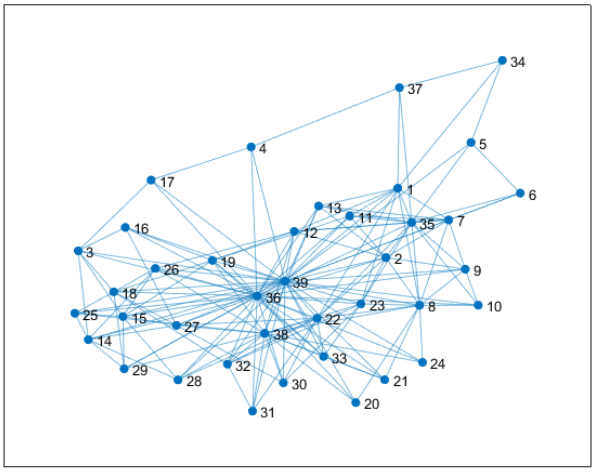


Figure 3.40: Spectral clustering algorithm

At last we apply the spectral clustering to the food web of marine organisms living in the Chesapeake Bay:



Setting $k = 4$, we can compare the community structure obtained to the expected one:

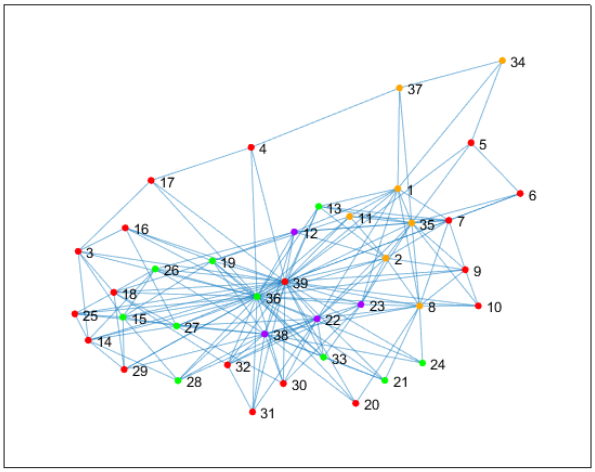


Figure 3.41: Community detection algorithm

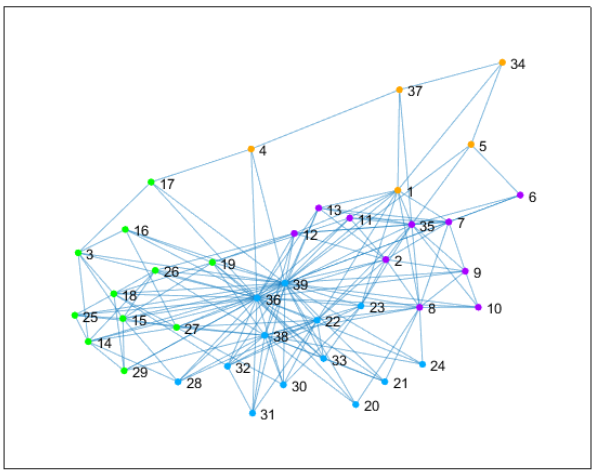


Figure 3.42: Spectral clustering algorithm

We can see that the two community structures are not so similar, looking at the picture 3.42 it seems that the nodes are grouped in clusters of points near to each others.

3.4 Conclusion

In this work we studied an approach to community detection based on quantum walks. At the end we can appreciate the benefits given by a quantum process used within an algorithm of network analysis. We tested this method to detect the communities of the graphs on many examples. We saw how the algorithm behaves over several networks and that it brings out the expected results, exploiting the quantum advantage.
An aspect to point is that we deduced that the Fourier matrix is more effective than the Grover one and, about that, we mentioned the topic of localization, addressed by the article of Kanae Mukai and Naomichi Hatano [14]. While here we focused on an experimental analysis of this phenomenon, it would be interesting to understand better how the eigenvalues of the time-evolution operator affect the preferability of the Fourier walk than the Grover walk.

Bibliography

[1] Dorit Aharonov et al. "Quantum walks on graphs". In: *Symposium on the Theory of Computing*. 2001.

[2] Dan Baird and Robert Ulanowicz. "The Seasonal Dynamics of The Chesapeake Bay Ecosystem". In: *Ecological Monographs* 59 (Dec. 1989), pp. 329–364. DOI: 10.2307/1943071.

[3] A.L. Barabasi. *Network Science*. Cambridge University Press, 2016.

[4] S. Boccaletti et al. "Complex networks: Structure and dynamics". In: *Physics Reports* 424.4 (2006), pp. 175–308. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2005.10.009. URL: https://www.sciencedirect.com/science/article/pii/S037015730500462X.

[5] A. De Mauro. *Big Data Analytics. Analizzare e interpretare dati con il machine learning*. Guida completa. Apogeo, 2019. ISBN: 9788850334780. URL: https://books.google.it/books?id=uAA0vwEACAAJ.

[6] Henrik Dosdall. "Kapferer (1972): Strategy and Transaction in an African Factory". In: Jan. 2019, pp. 289–292. ISBN: 978-3-658-21741-9. DOI: 10.1007/978-3-658-21742-6_66.

[7] Ernesto Estrada and Desmond J. Higham. "Network Properties Revealed through Matrix Functions". In: *SIAM Review* 52.4 (2010), pp. 696–714. DOI: 10.1137/090761070. eprint: https://doi.org/10.1137/090761070. URL: https://doi.org/10.1137/090761070.

[8] M. Girvan and M. E. J. Newman. "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences* 99.12 (June 2002), pp. 7821–7826. DOI: 10.1073/pnas.122653799. URL: https://doi.org/10.1073%5C%2Fpnas.122653799.

[9] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[10] Abhijith J. et al. "Quantum Algorithm Implementations for Beginners". In: *ACM Transactions on Quantum Computing* 3.4 (July 2022), pp. 1–92. DOI: 10.1145/3517340. URL: https://doi.org/10.1145%5C%2F3517340.

[11] Lin Lin. *Lecture Notes on Quantum Algorithms for Scientific Computation*. 2022. DOI: 10.48550/ARXIV.2201.08309. URL: https://arxiv.org/abs/2201.08309.

[12] D. Lousseau et al. "The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations". In: *Behavioral Ecology and Sociobiology* 54 (2003), pp. 396–405.

[13] Ulrike von Luxburg. "A Tutorial on Spectral Clustering". In: (2007). DOI: 10.48550/ARXIV.0711.0189. URL: https://arxiv.org/abs/0711.0189.

[14] Kanae Mukai and Naomichi Hatano. "Discrete-time quantum walk on complex networks for community detection". In: *Physical Review Research* 2.2 (June 2020). DOI: 10.1103/physrevresearch.2.023378. URL: https://doi.org/10.1103%5C%2Fphysrevresearch.2.023378.

[15] M. E. J. Newman. *Networks: an introduction*. Oxford; New York: Oxford University Press, 2010. ISBN: 9780199206650 0199206651. URL: http://www.amazon.com/Networks-An-Introduction-Mark-Newman/dp/0199206651/ref=sr_1_5?ie=UTF8&qid=1352896678&sr=8-5&keywords=complex+networks.

[16] M. E. J. Newman. "The Structure and Function of Complex Networks". In: *SIAM Review* 45.2 (Jan. 2003), pp. 167–256. DOI: 10.1137/s003614450342480. URL: https://doi.org/10.1137%5C%2Fs003614450342480.

[17] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. USA: Cambridge University Press, 2011. ISBN: 1107002176.

[18] R. Portugal. *Quantum Walks and Search Algorithms*. Springer New York, NY, 2013.

[19] Wolfgang Scherer. *Mathematics of Quantum Computing: An Introduction*. 1st. Springer Publishing Company, Incorporated, 2019. ISBN: 303012357X.

[20] Pang-Ning Tan et al. *Introduction to Data Mining (2nd Edition)*. 2nd. Pearson, 2018. ISBN: 0133128903.

[21] S. E. Venegas-Andraca. "Quantum walks: a comprehensive review". In: *Quantum Information Processing* 11 (2012), pp. 1015–1106.

[22] Martin Weißmann and Johannes Zück. "Kapferer (1969): Norms and the Manipulation of Relationships in a Work Context". In: Jan. 2019, pp. 285–288. ISBN: 978-3-658-21741-9. DOI: 10.1007/978-3-658-21742-6_65.

[23] Ronald de Wolf. "Quantum Computing: Lecture Notes". In: (July 2019). arXiv: 1907.09415 [quant-ph].

[24] Wayne W. Zachary. "An Information Flow Model for Conflict and Fission in Small Groups". In: *Journal of Anthropological Research* 33.4 (1977), pp. 452–473. ISSN: 00917710. URL: http://www.jstor.org/stable/3629752 (visited on 11/19/2022).