INTRODUCTION

Transportation, traffic, communication and energy networks form the backbone of our modern society. To deal with the uncertainty, variation, unpredictability, size and complexity inherent in these networks, we need to develop radically new ways of thinking. The ultimate goal is to build self-organising and intelligent networks. The NWO-funded Gravitation programme NETWORKS started in the Summer of 2014 and covers a broad range of topics dealing with stochastic and algorithmic aspects of networks.

In spring 2023 the fifth “NETWORKS goes to school” event was organised. The aim of the event is to provide secondary education students and teachers a first mathematical introduction on network science. This book collects the material realised for the “NETWORKS goes to school” event.

The content of this book is intended for secondary education students and teachers and aims to provide a first mathematical introduction to network science. In Chapter 1, all the necessary background material that is required for Chapters 2 and 3 is presented. In Chapter 2, we introduce networks theory by showing how to model and analyse a network with mathematical techniques. In this chapter, we will try to show mathematical techniques to find communities in networks. Chapter 3 focuses on road traffic networks, and discusses how navigation systems select routes. Chapter 4 contains exercises on these two topics and in Chapter 5 we provide the corresponding solutions. Chapters 2, 3, 4, and 5 were written with the help of Martijn Gösgens (Eindhoven University of Technology) and Nikki Levering (University of Amsterdam).

For more information and the booklets of the first four masterclasses “NETWORKS goes to school”, please visit networkpages.nl/category/networks-goes-to-school/. This masterclass is sponsored by the Netherlands Organisation for Scientific Research (NWO) through the Gravitation grant ”NETWORKS”, and by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant no. 945045.

On behalf of the NETWORKS programme,

the organising committee of “NETWORKS goes to school”
Mehmet Akif Yildiz (University of Amsterdam)
Nicos Starreveld (University of Amsterdam)

Contents

Chapter 1

Preliminaries

1.1. Basic notation

We start by introducing some notation we will use in the sequel:

(1) $\mathbb{N}$ for the set of natural numbers, that is $\mathbb{N} = \{1, 2, 3, \cdots\}$;
(2) $\mathbb{Z}$ for the set of integer numbers, that is $\mathbb{Z} = \{\cdots, -3, -2, -1, 0, 1, 2, 3, \cdots\}$.
(3) $\mathbb{R}$ for the set of real numbers, that is all integer numbers and all the decimal numbers between them.
(4) We will use the symbol $\leq$ when we want to say "less or equal to". For example, $a \leq b$ means that a is less or equal to b.
(5) We will use the symbol $\approx$ when we want to say " almost equal to". For example. $\pi \approx 3.14159$.
(6) We will often work with subsets of the natural numbers. For a set I, we write I^c or $\mathbb{N} \setminus I$ for it's complement. For example, if $I = \{2, 4, 6, \ldots\}$ contains all even natural numbers, then I^c contains all odd natural numbers.
(7) The number of elements in some set A is denoted by $|A|$.
(8) We will use the notation *max* and *min* for the maximum and minimum value in some set or of some function. For example

$$\max_{i \in \{1,2,3\}} i^2 = 9 \qquad \text{and} \qquad \min_{j \in \{-1,0,1\}} (i+1) = 0.$$

On sums

Mathematicians always want to write down mathematics as compactly as possible. But the notation used should also be clear and representative of what it describes. The typical notation you encounter when doing mathematics involves the symbol used for summation: $\sum$.

Below we show how $\sum$ is used to describe a sum. Suppose you want to use the summation symbol to describe the sum $1+2+3+4+5+6$, then you can write this down compactly as

$$\sum_{k=1}^{6} k = 1+2+3+4+5+6 = 21. \tag{1.1.1}$$

The advantage of this notation is that you can write down large sums very compactly. For example, if you want the sum of the first 100 natural numbers, instead of only up to 6, then you can write this down as

$$\sum_{k=1}^{100} k. \tag{1.1.2}$$

By playing with the value at which the sum starts or ends you see that you can represent many sums or products using this notation. The general notation is the following:

$$\sum_{k=m}^{n} a_k, \tag{1.1.3}$$

where k is the index of summation; a_k are indexed variables representing each term of the sum; m is the lower bound of summation, and n is the upper bounds of summation.

In all the expressions above, either of summations or products, you can remark that the index k in every step increases by one, it starts from a number m, then takes the value $m+1$, $m+2$ until it reaches the number n. It is also possible to choose the indices from some set of values. Say for example that you want to compute the sum of the squares of all even numbers greater or equal to 4 and less or equal to 20. You can define the set of indexes you want to sum over, in this case

$$I = \{N \in \mathbb{N} : 4 \leq N \leq 20 \text{ and } N \text{ is even}\} = \{4, 6, 8, 10, 12, 14, 16, 18, 20\}.$$

Then the desired sum can be written as

$$\sum_{k \in I} k^2 = 16 + 36 + 64 + 100 + 144 + 296 + 324 + 400 = 1380.$$

This sum could also be written compactly as follows

$$\sum_{k \in I} k^2 = \sum_{4 \leq k \leq 20,\, k \text{ even}} k^2.$$

For two quantities x and y, we say x is a *lower bound* for y if $x \leq y$. Similarly, we say x is an *upper bound* for y if $x \geq y$.

1.2. Probability theory

Probability theory is the area of mathematics that studies random phenomena. For example if the experiment is tossing a coin, then there are two possible outcomes, either *heads* or *tails*. Each outcome occurs with probability 0.5. In order to study such a random experiment we use random variables.

Random variable

A **random variable** X is a variable whose possible values are outcomes of a random experiment. We will also use the term *stochastic* as a synonym for random.

We define a random variable by giving the *state space*, i.e. the set of all possible values the variable can take, and the *probability function*, which yields the corresponding probability that a given outcome will occur. For the coin toss for example we can define a random variable by assigning to the outcome *heads* the value 1 and to the outcome *tails* the value 0. In this case we have

$$X(\textit{heads}) = 1 \quad \text{and} \quad X(\textit{tails}) = 0.$$

The probability function for this random variable is given by

$$\mathbb{P}(X = 1) = \mathbb{P}(\textit{heads}) = 0.5,$$

and

$$\mathbb{P}(X = 0) = \mathbb{P}(\textit{tails}) = 0.5,$$

where for a possible set of outcomes A, $\mathbb{P}(A)$ denotes the probability that A occurs. A random variable can be *discrete* or *continuous*.

Discrete random variables

A random variable X is called discrete when it can take countable many values, for simplicity we can just say that its values are the integer numbers, that is $X \in \mathbb{Z}$.

Continuous random variables

A random variable X is called continuous when it can take continuously many values, for simplicity we can just say that its values are the real numbers that is $X \in \mathbb{R}$.

For a discrete random variable, we can write down the probability that it equals a specific value. For a continuous random variable, this is not possible, as there is a continuum of possible values. We can however specify the probability that a continuous random variable falls in a range of values by using the **density function**. The probability that a continuous random variable X assumes values in the interval $[a, b]$ is given by the integral of the density function, denoted by f_X, over that interval:

$$\int_a^b f_X(x)\mathrm{d}x = \mathbb{P}(X \in [a, b]).$$

The result of this integration gives the area delimited by the graph of the density function f_X, the x-axis and the vertical lines given by $x = a$ and $x = b$.

Expectation of a random variable

For a random variable X, discrete or continuous, we define the expectation, or expected value, as the average of all independent realisations of the random variable. We denote the expectation of X by $\mathbb{E}[X]$.

For a discrete random variable its expectation is defined by

$$\mathbb{E}[X] = \sum_{k=0}^{\infty} k\mathbb{P}(X = k). \tag{1.2.1}$$

For a continuous random variable its expectation is defined by

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f_X(x)\mathrm{d}x, \tag{1.2.2}$$

where f_X denotes the density function of the random variable, this means that

$$f_X(x)\mathrm{d}x = \mathbb{P}(X \in \mathrm{d}x). \tag{1.2.3}$$

As we will see in the sequel, if the random variable takes only positive values then the integral in the expectation starts from 0 instead of $-\infty$.

Bernoulli random variable

Bernoulli random variable

A **Bernoulli random variable** describes the outcome of any single random experiment that asks a yes-no question, like tossing a coin.

It takes the value 1 with probability p and the value 0 with probability $1 - p$. Consider for example a coin where one side is heavier, then this is a biased coin where one side is favoured. We will use $B(p)$ to denote a Bernoulli random variable with probability p. A Bernoulli random variable has expectation given by

$$\mathbb{E}[B(p)] = 1 \cdot \mathbb{P}(B(p) = 1) + 0 \cdot \mathbb{P}(B(p) = 0) = p. \tag{1.2.4}$$

Binomial random variable

Binomial random variable

A **binomial random variable** describes the number of successes in a sequence of independent experiments, each asking a yes–no question.

We make the following assumptions:

- the number n of observations is fixed;
- each observation is independent of the other observations;
- each observation represents one of two outcomes: success or failure (yes-no);
- the probability p of success is exactly the same for each trial.

Under these assumptions, we can describe each binomial random variable by using the parameters n and p. We will denote a binomial random variable by $B(n,p)$. The binomial random variable $B(n,p)$ has state space $\{0,1,\dots,n\}$ as it counts the number of successful trials, and the probability that $B(n,p)$ is equal to k is given by

$$\mathbb{P}(B(n,p)=k)=\binom{n}{k}p^k(1-p)^{n-k}$$

where

$$\binom{n}{k}=\frac{n!}{k!(n-k)!}$$

is the **binomial coefficient**. The symbol $\binom{n}{k}$ is read as 'n choose k', as this is the number of ways to choose k different elements from a total of n elements, where the order of elements does not matter. The factorial of n is denoted by $n!$ and it is equal to the product $n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot 1$. The binomial random variable has expectation equal to

$$\mathbb{E}[B(n,p)]=\sum_{k=0}^{\infty}k\mathbb{P}(B(n,p)=k)=\sum_{k=0}^{n}k\binom{n}{k}p^k(1-p)^{n-k}=np. \tag{1.2.5}$$

The exact derivation of this result is far away from the scope of this booklet.

EXAMPLE 1.2.1. Suppose that we have a total of 5 colours, and we wish to know how many combinations there are of 3 different colours, where the order of the colours does not matter. Then $n=5$ and $k=3$, and

$$\binom{5}{3}=\frac{5!}{3!2!}=10.$$

We could also reason in a different way. For the first choice we have a total of 5 possible colours, for the second choice we have 4 possible colours and for the third choice we have 3 possible colours. The total of combinations of three colours is then $5\cdot4\cdot3=5!/2!$. However, the order of colours did not matter so we still have to divide by the number of ways in which we can order 3 colours, which is $3\cdot2\cdot1=3!$.

EXAMPLE 1.2.2. Consider a coin toss, where possible outcomes are heads or tails. Suppose that we have a fair coin, i.e., the probability for heads is the same as it is for tails. If we toss the coin 10 times, then the number of coin tosses that came heads from those ten tosses has a binomial distribution with parameters $n = 10$ and $p = \frac{1}{2}$. The probability of getting exactly four heads is equal to

$$\mathbb{P}(X = 4) = \binom{10}{4}\left(\frac{1}{2}\right)^4\left(1-\frac{1}{2}\right)^{10-4} = \frac{105}{512} \approx 0.205.$$

Normal random variable

Normal random variable

The normal random variable is a continuous random variable that has a symmetric density function. The plot of the density function has a bell-shaped form.

A normal random variable is characterised by two parameters, μ and σ^2. Its probability density function is given by

$$f_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \qquad x \in \mathbb{R}.$$

Figure 1.2.1 shows what this density function looks like for multiple values of μ and σ^2. From this figure, it is clear that the density function is symmetric. Thus, the outcome of a normal random variable with parameters μ and σ^2 is with probability $\frac{1}{2}$ smaller than μ, and with probability $\frac{1}{2}$ it is larger. Hence, it is not surprising that the expectation of such a random variable is equal to the parameter μ.
The parameter σ^2 determines how likely it is that the outcome of a normal random variable deviates from its expectation μ. In other words, if σ is large, the bell shape in Figure 1.2.1 will be much wider, and the actual value of the random variable is likely to be further away from μ. Because of this feature, the parameter σ^2 is also called the variance, and the square root of the variance, namely σ itself, is called the standard deviation.
The bell shape of the normal random variable occurs naturally in a variety of settings. Sample averages, for example the average height of a large group of persons, tend to be approximately normally distributed, which is why normal random variables are often encountered.
The distribution function is however hard to compute. If $\mathcal{N}(\mu,\sigma^2)$ denotes a normal random variable with parameters μ and σ^2, we have

$$\mathbb{P}(\mathcal{N}(\mu,\sigma^2) \leq t) = \int_{x=0}^{t} \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}\,\mathrm{d}x.$$

Since there is no easy way to compute this integral, we often use tables such as Table 4.0.4 at the end of Chapter 4, page 52. This table lists, in case $\mu = 0$ and $\sigma = 1$, the value of

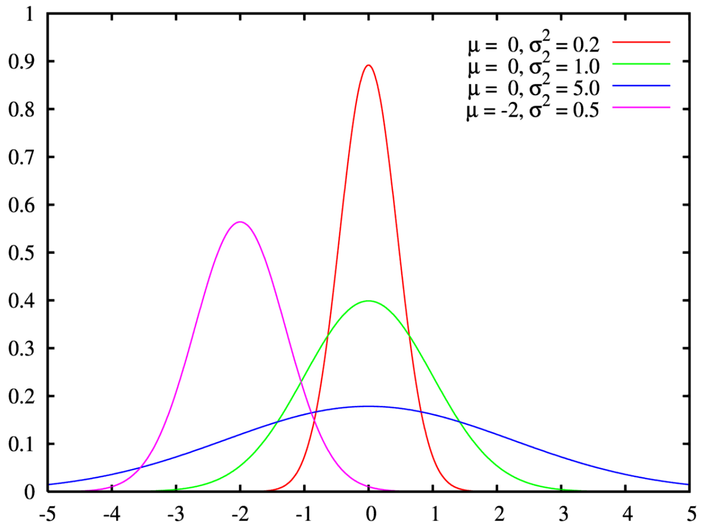


Figure 1.2.1. The density function of a normally distributed random variable with parameters μ and σ^2.

$\mathbb{P}(\mathcal{N}(0,1) \leq t)$ for various values of t between 0 and 3.4. For example, using the table, we find that

$$\mathbb{P}(\mathcal{N}(0,1) \leq 1.23) = 0.8907,$$

but due to the symmetry also that

$$\begin{aligned}\mathbb{P}(\mathcal{N}(0,1) \leq -0.87) &= \mathbb{P}(\mathcal{N}(0,1) > 0.87)\\ &= 1 - \mathbb{P}(\mathcal{N}(0,1) \leq 0.87)\\ &= 1 - 0.8078 = 0.1922.\end{aligned}$$

But how do we go about finding the distribution function of normal random variables with $\mu \neq 0$ or $\sigma^2 \neq 1$? It turns out that we can then also use this table. For a normal random variable with parameters μ and σ^2, we have for any number u that

$$\mathbb{P}(\mathcal{N}(\mu,\sigma^2) \leq u) = \mathbb{P}\left(\mathcal{N}(0,1) \leq \frac{u-\mu}{\sigma}\right).$$

So, if we want to compute $\mathbb{P}(\mathcal{N}(1,4) \leq 3)$ for example, we use the table and look up the value for $t = \frac{u-\mu}{\sigma} = \frac{3-1}{\sqrt{4}} = 1$, and find that $\mathbb{P}(\mathcal{N}(1,4) \leq 3) = \mathbb{P}(\mathcal{N}(0,1) \leq 1) = 0.8413$.

1.3. Graph theory

An intuitive definition of a network would be a 'collection of objects that are interconnected in some way'. Think for example of a collection of people, who can be interconnected by friendships; or a collection of cities, which can be interconnected by roads. To make this idea precise, we turn to graph theory.

Graph

A **graph** is a pair $G = (V, E)$, where
- V is the set of nodes or vertices;
- E is the set of edges, connecting the nodes.

Typically, we number the nodes from $\{1, 2, 3, \dots, \}$. We denote an edge between two nodes i and j by $\{i, j\}$. To define a graph, we can write down the sets V and E.

EXAMPLE 1.3.1. Consider

$$V = \{1, 2, 3, 4, 5, 6\}, \quad E = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}.$$

Then $G = (V, E)$ is a graph with six nodes and seven edges.

It may be very useful to have a graphical representation of a graph. We do this by typically drawing nodes as a circle with a label in it, and edges as a line between nodes. However, you are free to choose any representation you may like! In fact, the location of the nodes is also arbitrary, it only matters the way in which the edges connect the nodes together.

EXAMPLE 1.3.1 (Continued). In Figure 1.3.1 we see two ways in which the graph G can be drawn.

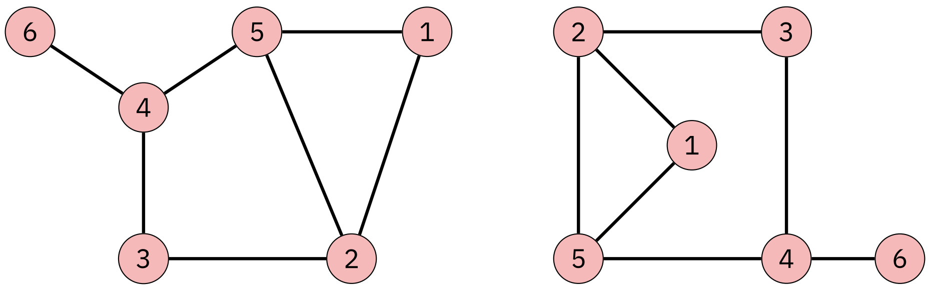


Figure 1.3.1. Two different representations of the graph in Example 1.3.1.

Degree of a node in a graph

The degree of a node v in a graph $G = (V, E)$, denoted by $d(v)$, is the number of neighbors of v. In the graph above for example the degree of node 1 is $d(1) = 2$, and of node 5 is $d(5) = 3$.

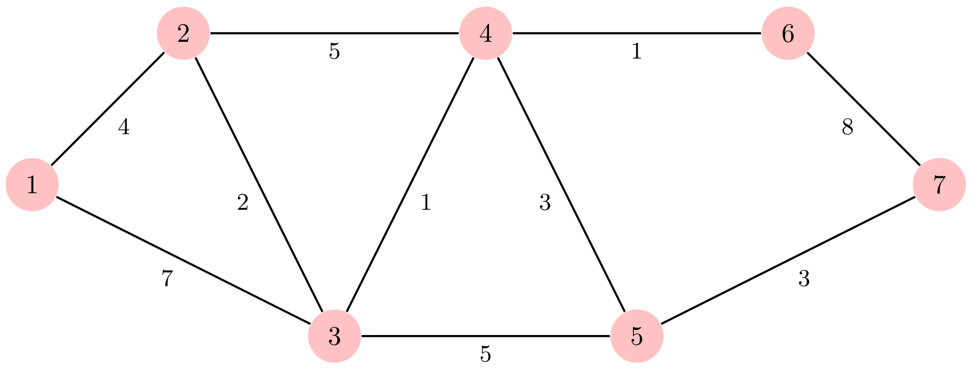


Figure 1.3.2. A network with seven locations

Path between two nodes in a graph

A path between two nodes in a graph, say v and w, is a sequence of edges which joins a sequence of nodes from v to w. In the graph in Figure 1.3.1 for example, the sequence $6 \to 4 \to 3 \to 2$ forms a path from 6 to 2. On the other side, the sequence $6 \to 4 \to 3 \to 1$ is not a path since $\{3, 1\}$ is not an edge in the graph. A shortest path between two nodes is a path using the least amount of edges. The shortest path from node 6 to node 1 for example has length 3.

Finding the shortest route within a network

Now that we have seen how graphs can be used to represent networks, there are many questions that can be asked about those networks. For instance, what is the shortest route from one location in a network to another? This is a question that we ask our favourite route planner on a daily basis.

To answer this question, let us look at the network depicted in Figure 1.3.2. This figure contains a graph, and let us assume that each node in this graph is a location in a network. Furthermore, an edge between two of these nodes represents a road between these two locations. The numbers in the figure represent the length of the roads, let us say in kilometres. Using these numbers, we can calculate the distance of a route in a network. For example, in Figure 1.3.2, it is clear that if we would travel from node 1 to node 4 via node 2, the distance traversed would be 4+5 = 9 kilometres. But, if we were to travel from node 1 to node 4 via node 3 instead, we would have to cover 7+1 = 8 kilometres. And if we were to travel from node 1 to node 4 via node 2 and 3, we would have to cover 4+2+1= 7 kilometres. Thus, ‘the shortest route’ from node 1 to node 4 is the route $1 \to 2 \to 3 \to 4$.

For this particular network, we can thus see in an eye blink what the shortest route from node 1 to node 4 is. But the shortest route from node 1 to node 7 is already harder to de-

nodes	1	2	3	4	5	6	7
Step 1	-	$(4,1)^*$	$(7,1)$	No edge	No edge	No edge	No edge
Step 2	-	-	$(6,2)^*$	$(9,2)$	No edge	No edge	No edge
Step 3	-	-	-	$(7,3)^*$	$(11,3)$	No edge	No edge
Step 4	-	-	-	-	$(10,4)$	$(8,4)^*$	No edge
Step 5	-	-	-	-	$(10,4)^*$	-	$(16,6)$
Step 6	-	-	-	-	-	-	$(13,5)^*$

Table 1.3.1. Dijkstra's shortest route algorithm for the network in Figure 1.3.2.

termine. And then, this is just a network with seven locations. Finding the shortest route in a much larger network by trial and error is simply not doable.
So how do we go about this? We will require a more systematic way of finding the shortest route. Luckily, it exists. We will use an algorithm!

Algorithm

An algorithm is a step-by-step procedure to perform a given task. Algorithms can be executed by computers, but also by persons.

Dijkstra's algorithm. More particularly, we will now consider an algorithm that finds the shortest route in a network. This algorithm was conceived in 1959 by Edsger W. Dijkstra, who was a Dutch systems scientist, programmer, software engineer, science essayist and pioneer in computing science.
The algorithm consists of iteratively performing a number of steps. In each of these steps, preliminary routes will be improved and in each step, a definitive shortest route from the starting node to any of the other nodes will be found. For the bookkeeping of these routes, we will keep records on each node. These records give an upper bound on the distance of the shortest route of the starting node to the corresponding node. In each of the steps, these records will be adjusted, and also one of these nodes is marked as 'permanent', indicating that a definitive shortest route from node 1 to the completed node has been found.
All this is perhaps best demonstrated by means of an example: we want to find the shortest route from node 1 to node 7 in Figure 1.3.2. Since the network in this figure has 7 locations, we will need 7-1 = 6 steps of the algorithm. In Table 1.3.1, we will keep track of all the bookkeeping that the algorithm generates.
To start the algorithm, it is worth noting that we already know the shortest route from node 1 to node 1: this route has distance zero, since we are already there! As such, we mark node 1 as 'permanent', and we will not consider node 1 in the steps we are going to perform, as is reflected in the table by '-'. The algorithm can now be started with node 1 as a permanent node. The rest of the nodes are considered 'non-permanent'. In each of the

steps, we will check which non-permanent nodes can be reached by permanent ones, and mark a non-permanent node as permanent. We do this as follows.

Step 1: Since node 1 is the only permanent node, we see from Figure 1.3.2 that only nodes 2 and 3 can be reached now from permanent nodes. Node 2 can be reached directly from node 1 with a distance of 4, so in the table, we write $(4, 1)$ in the first row (corresponding to Step 1) in the column of node 2. Similarly, we write 7 for node 3 in the first row, as node 3 can be reached directly from node 1 with distance $(7, 1)$.
In the bracket we always write two numbers, the first number is the distance from a permanent node and the second represents the node from which it can be reached. Nodes 4, 5, 6 and 7 can not be reached directly from node 1, and hence we write 'No edge' for these nodes in the first row.
The final part of the step consists of marking a non-permanent node as permanent. We will always mark the node with the lowest distance in the row as permanent. In this case, this is node 2 with distance 4 from node 1, so we make node 2 permanent. We denote this by adding an asterisk to the record of node 1 in the first row. The algorithm now says that the shortest route from node 1 to node 2 now simply is the direct route $1 \to 2$ with distance 4.

Step 2: In Step 2 we check whether routes can be made shorter using node 2 as an intermediate node in the route towards other nodes. For node 3, we know from Step 1 that it can be reached directly from node 1 within distance 7. However, since node 2 now is permanent, node 3 can also be reached from node 2: the edge $\{2, 3\}$ has distance 2, and we know that node 2 itself can be reached with distance 4. Therefore, node 3 can also be reached within distance $2 + 4 = 6$, when going via node 2. Therefore, we write $(6, 2)$ for node 3 in Table 1.3.1 in the row corresponding to Step 2 and in the column corresponding to node 3. We conclude that we have made the route from node 1 to node 3 one kilometre shorter!
Node 4 can now also be reached using edge $\{2, 4\}$ with distance 5. As node 2 itself can be reached within distance 4, node 4 can thus now be reached within distance 5+4 = 9 with preceding node 2. Therefore, we write the record $(9, 2)$ in the table. From nodes 1 and 2, there are still no routes possible to nodes 5,6 and 7, leading to a 'No edge'-record.
Between nodes 3 and 4, node 3 has the shorter distance (namely 6), and therefore we now mark node 3 as permanent with an asterisk.

Step 3: We follow the exact same procedure as the previous steps. Namely, we check whether the now permanent node 3 leads to shorter routes for the other nodes. This is the case for node 4. While in Step 2 we found a distance of 9, we now find a distance 7 via node 3, leading to the record $(7, 3)$. Indeed, node 3 could be reached within distance 6, and the edge $\{3, 4\}$ has distance 1. Node 5 can now be reached via the permanent node 3: namely, node 3 can be reached within distance 6, and edge $\{3, 5\}$

has distance 5, leading to a total distance 6+5=11 and the record $(11, 4)$.
The route to node 4 (distance 7) is now shorter than the route to node 5 (distance 11), so node 4 becomes a permanent node. At this point, nodes 1-4 are permanent, whereas nodes 5, 6 and 7 are still non-permanent. Hence, steps 4-6 will deal with the latter nodes.

Step 4: As node 4 is now a permanent node, we check how this affects the shortest routes of the still non-permanent nodes 5-7. Indeed, the route from node 1 to node 5 can now be made shorter by routing through node 4: we first take the shortest route to node 4 (distance 7 found in step 3) and then use the edge $\{4, 5\}$ (distance 3). This route has distance 7+3=10, which is shorter than the distance 11 in the record for node 5 in step 3. Therefore, the record for node 5 in step 4 becomes $(10, 4)$. However, node 6 can now also be reached, via the new permanent node 4. This route will have distance 7+1 = 8.
Since node 6 now has the shorter of the two found distances of nodes 5 and 6, node 6 will become permanent. The shortest route from node 1 to node 6 has distance 8, and cannot be made shorter in future steps.

Step 5: There are just two non-permanent nodes left at this point: nodes 5 and 7. There is no direct edge between the most recently permanent node 6 and node 5, so the record of node 5 remains the same as in the previous step: $(10, 4)$. Node 7 can now finally be reached through node 6 (distance 8 found in step 5) and edge $\{6, 7\}$ (distance 8), leading to distance 8+8=16. As a result, we will flag node 5 as permanent, with distance 10 and preceding node 4.

Step 6: Only node 7 is an non-permanent node at this point. We only need to check whether node 5, which we flagged as permanent in the previous step, leads to a shorter route than the one found in step 5 via node 6. This turns out to be the case: if we first go to node 5 (distance 10), and then take the direct edge $\{5, 7\}$ (distance 3), the route will only have distance 13, rather than 16 as found in Step 5. Therefore, the shortest route from node 1 to node 7 has distance with preceding node 5, leading to the record $(13, 5)$ in the table. We finally mark node 7 as permanent, so that there are no non-permanent nodes anymore.

Now that we have performed all the steps of Dijkstra's algorithm, we know that the shortest route from node 1 to node 7 has a distance of 13. To find which route this exactly is, we look at Table 1.3.1, and look at the records with an asterisk (i.e. the records of the nodes when they were marked as a permanent node). In the row of step 6, we see that the preceding node of node 7 is node 5, meaning that the shortest route of node 1 to node 7 coincides with the shortest route of node 1 to node 5, plus the additional edge $\{5, 7\}$. Node 5 was made permanent in step 5 with preceding node 4, meaning that the shortest route from node 1 to node 7 must have the form $1 \to \ldots \to 4 \to 5 \to 7$. Continuing like this, we find the shortest route $1 \to 2 \to 3 \to 4 \to 5 \to 7$.

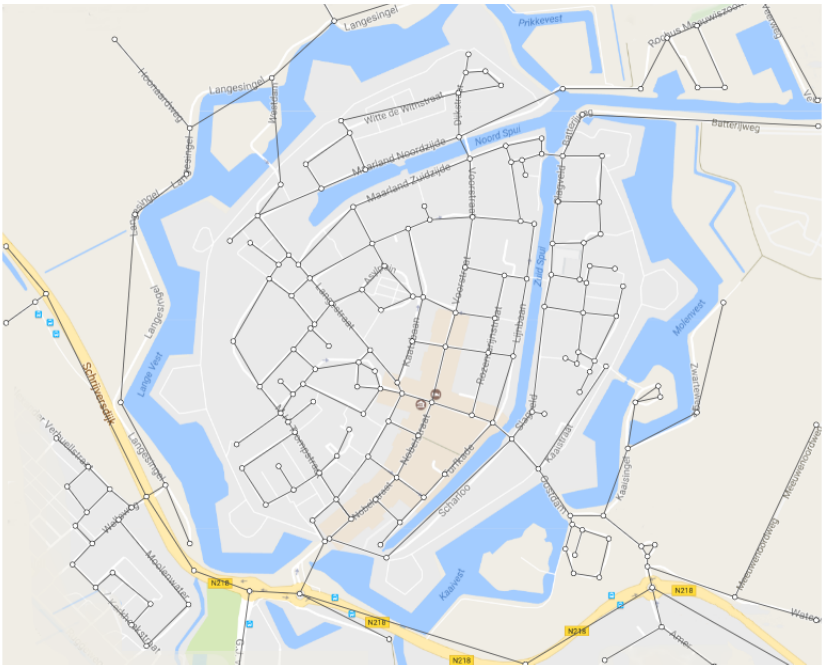


Figure 1.3.3. A real time animation of Dijkstra's algorithm on the map of Brielle, the animation can be found on: networkpages.nl/finding-the-shortest-route-to-your-holiday-destination-dijkstras-algorithm/.

On the Network Pages

For further reading on probability theory, algorithms, networks and graph theory have a look at networkpages.nl/category/basic-notions/!

Chapter 2

Networks Theory - Communities in networks

In many networks, a natural grouping of the network nodes can be discovered. In social networks, for example, friend groups are often visible as groups of nodes that have many internal connections. In network science, such groups of nodes that are more strongly connected to each other than to the rest of the network are referred to as *communities*. Communities are not just limited to social networks. For example, we can consider Wikipedia as a network of pages where two pages are connected if there is a link from one to the other. In this Wikipedia network, pages will have many links to pages with related subject, forming groups that are clearly visible in the network.

There are many different reasons why researchers are interested in communities. In the example of the Wikipedia network, this helps find pages that are likely about a similar subject. The surprising thing is that this can be done without reading the specific Wikipedia pages, but by just looking at the links between them. This can be useful for automatically classifying pages into categories or detecting which pages may be wrongly classified.

Network communities are also relevant in epidemiology, where the presence of such communities influences the way a virus spreads through a population. Diseases spread easily within communities and contacts between people of different communities may lead a virus to enter a new community. If the communities are very closely connected, like households for example, then it may even be more useful to ignore the individual people in the network and instead consider the population as a network of households.

Nevertheless, the main application of community detection is to obtain a better understanding of the network. In modern times, networks tend to be enormous, consisting of millions (e.g., Wikipedia) or even billions of nodes (e.g., Facebook). When one tries to understand such a large network, knowledge of a single node and its connections is less rel-

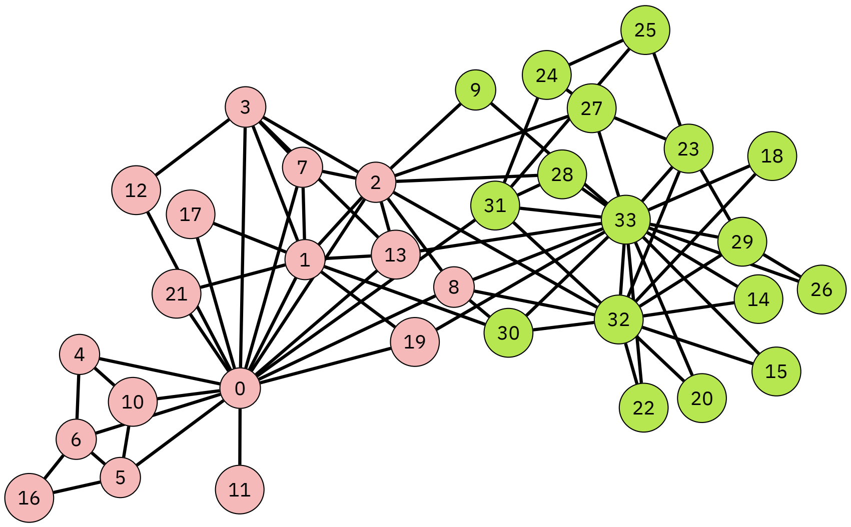


Figure 2.0.1. Zachary's karate club network. Node 0 is the instructor while node 33 is the clubs administrator. The colors indicate the split after the conflict. Source: Wikipedia

evant, while it becomes more useful to look at which groups the nodes form and how these groups are connected to each other.

The karate network. Community detection is usually applied to networks that are so large that it is impossible to get insight into them without using all sorts of advanced algorithms and statistics. However, to understand and experiment with community detection methods, it often helps to try them out on networks that are small and easy to understand. In the field of community detection, one particular network that is often used is *Zachary's karate club network*, a small network of 34 members of a karate club from the 70s. This karate club was studied by an anthropologist, who kept track of which members met each other outside of the club. During this research, a conflict arose among the administrator and the trainer of the club, which led the club to split up into two equal halves. Community detection methods are often 'benchmarked' based on how well they are able to recognize this split based on the network. Figure 2.0.1 shows the network, as well as the split into the two groups.

The FIFA2022 network. We can also consider a tournament as a network of teams, where teams are connected if they have played a match against each other. If we take the football world cup of 2022, for example, we get a network of 32 nodes and 64 edges. The minimal degree of this network is 3, corresponding to the teams that dropped out after the

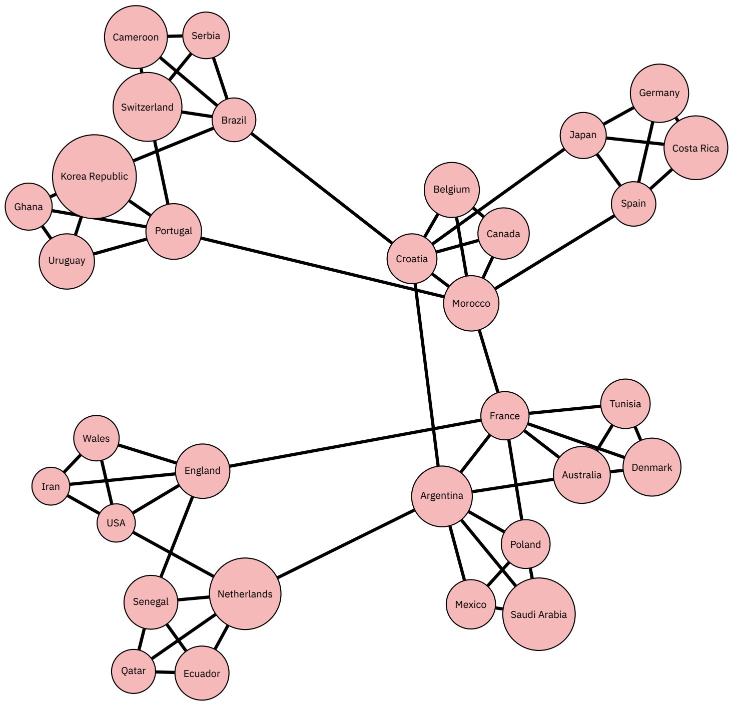


Figure 2.0.2. The network of the FIFA World Cup from 2022. Every node corresponds to a participating country while an edge indicates that the two teams played a match against each other.

group stage, while the tournament champion has maximum degree 7. The network is shown in Figure 2.0.2.

Notation. We denote a network by $\mathcal{N}$, and denote its node set by N and its edge set by E. We denote the number of edges between two subsets of nodes $A, B \subseteq N$ by $\ell(A, B)$ and write $\ell(A) = \ell(A, A)$ for the number of edges inside A. For a set $A \subseteq V$, we denote the sum of degrees by $d(A) = \sum_{i \in A} d(i)$. We write C for a partition into communities, which we represent as a set of sets $C = \{C_1, C_2, \ldots, C_{|C|}\}$, where the r-th community is denoted by C_r and the number of communities is denoted by $|C|$. We denote the community that a node i belongs to by $C(i)$. Thus, $i \in C(i)$ holds by definition for every node i. The number of possible node pairs is given by $\binom{|N|}{2} = |N| \cdot (|N| - 1)/2$, see Exercise 1.

2.1. What do communities look like?

While the concept of a community may sound intuitive at first, there is no clear definition of the word. The most agreed-upon definition of a community is rather vague: a group of nodes that is better-connected to each other than to the rest of the network. The reason that this definition is so vague, is that the exact meaning of 'community' may differ per network and application. For example, if you consider a social network where the communities represent groups of friends, then you expect nearly all people in the same friend group to be connected by a link. In contrast, in the example of Wikipedia pages you wouldn't expect every page to link to every other page of that subject. Because of this, there are many different definitions and the most useful definition may differ based on the context.

A community resembles a clique. The 'ideal' image of a community is a *clique*: a group of nodes where each node has an edge to each other node in the group. Many initial attempts at defining what a community is, have done so by generalizing the definition of a clique in some way. For example, a *k-plex* is a group of nodes so that for each node, the amount of nodes that it is *not* connected to is at most k. In contrast, a *k-core* is a group of nodes such that each node has at least k neighbors inside the k-core. Note that a clique of size n forms a 0-plex and a $(n-1)$-core. An even looser definition of a community is a *strong community*: a group $C \subseteq N$ such that for each node $i \in C$ a majority of the neighbors of i are in C. The above definitions are attempts at describing what a community *must* look like. Given a network and a group of nodes C, these allow to answer the question "Is C a community?" according to these definitions. However, in many networks, such as the Wikipedia network, there exist natural groupings that do not fit any of these criteria. Because of this, the focus moved away from hard definitions of what communities are, towards measures that quantify *how much* a group of nodes is like a community.

There are relatively few connections between communities. Many of the first attempts at detection communities in networks simply divided the network nodes into groups such that the number of edges between these groups is minimized. The *cutsize* of a set of nodes $C \subset N$ is defined as the number of edges connecting C to the remainder $N \setminus C$. That is,

$$\text{Cut}(C) = \ell(C, N \setminus C). \tag{2.1.1}$$

However, this criteria does not take into account the sizes of the sets C and $N \setminus C$. Thus, it may be that an algorithm minimizing the cutsize, will simply lead to an C consisting of a single node. For example, suppose we have two 1-plexes each of size k and suppose that each node is also connected to one node of the other group. Then the natural division into the two 1-plexes would result in a cutsize of k, while isolating a single node leads to a cutsize of $k-1$, as shown in Figure 2.1.1. The problem with this measure, is that we do not

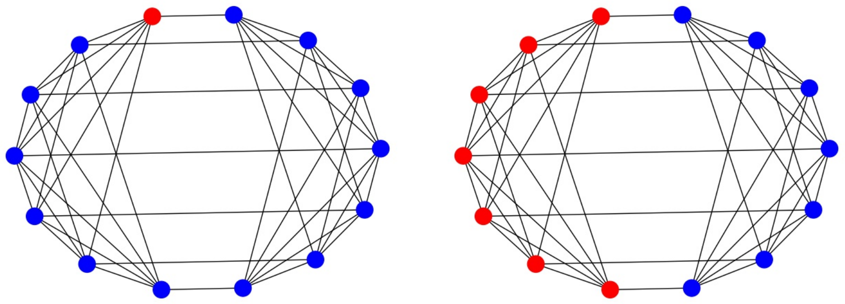


Figure 2.1.1. Two possible cuts for a network consisting of two 1-plexes of size 7 each, where each node has one neighbor in the other 1-plex. The cutsize is minimized by isolating a single node, as shown on the left, while the ratio cut is minimized by spliting the network into the two 1-plexes, as shown on the right.

take into account the sizes of the sets C and $N \setminus C$. Since $|C| = 1$ and $|N \setminus C| = 2k - 1$, the maximum number of *possible* edges is $|C| \cdot |N \setminus C| = 2k - 1$. The *ratio cut* corrects for this by dividing the cutsize by the number of possible edges

$$\text{RatioCut}(C) = \frac{\ell(C, N \setminus C)}{|C| \cdot |N \setminus C|}. \tag{2.1.2}$$

In the above example, the natural division into the two 1-plexes has a ratio cut of $\frac{k}{k \cdot k} = \frac{1}{k}$, which is indeed lower than the value $(k-1)/(2k-1)$ corresponding to isolating a single node.

A community contains many edges. Intuitively, you expect a network to have many edges *inside* communities and few edges *between* communities. The simplest measure of how well-connected a group $C \subseteq N$ of nodes is, is simply the number of edges $\ell(C)$ between them. However, this does not take into account the number of edges that are *absent* in this group, or the number of edges that connect this group to the remainder of the network. The maximal number of edges inside a group of size $|C|$ is $\binom{|C|}{2} = |C| \cdot (|C| - 1)/2$, since each node can connect to the $|C| - 1$ others and this would result in counting each edge twice. The *density* of a community C is simply the fraction of pairs that are connected by an edge. It is given by

$$\text{Density}(C) = \frac{\ell(C)}{\binom{|C|}{2}}. \tag{2.1.3}$$

You would expect communities to have a higher density than the rest of the network. Thus, we need to compare the density of the group to that of the graph as a whole. That is, we expect communities to have a density that is higher than Density(N).

Distances inside communities are short. Suppose you are on a particular Wikipedia page and you want to visit another Wikipedia page, but are too lazy to use your keyboard to search for this page. In this case, you might just click on links to other pages until you reach the desired page. It seems reasonable that a path to the desired page is shorter if it is in the same category as the page you are currently on. Let us denote by $\text{Dist}(i, j)$ the distance between two nodes, i.e., the number of edges in the shortest path. If there is no path between i and j, then the distance is infinite. If S is a connected subnetwork (that is, each node can be reached from each other node), then the *eccentricity* of i in S is given by

$$\text{Ecc}(i, S) = \max_{j \in S} \text{Dist}(i, j). \tag{2.1.4}$$

Similarly to the definitions of the diameter and radius of a circle, the diameter is the maximum eccentricity, while the radius is the minimum eccentricity of S. That is,

$$\text{Diameter}(S) = \max_{i \in S} \text{Ecc}(i, S), \quad \text{Radius}(S) = \min_{i \in S} \text{Ecc}(i, S). \tag{2.1.5}$$

Communities generally tend to have a low diameter and a low radius, compared to the radius and diameter of the whole graph. In Exercises 3, 4, and 6 in Chapter 4 you can practise with these new concepts and compute them in two concrete examples of networks.

Communities are 'hard to escape'. A different but related characterization of communities is that edges between nodes of different communities often form *bottlenecks*. Consider, for example, a city divided by a river. While there may be many roads on either sides, there may only be a few bridges connecting the sides. Thus, if you want to move from a place on one side of the river to the other, you will have to pass over one of the few bridges. A way to quantify this, is to compute the shortest path for each pair of nodes and count for each edge the amount of shortest paths it belongs to. For two nodes $i, j \in N$ that are connected by an edge, we define the *bottleneck-ness* of this edge by

$$\text{Bottleneck}(ij) = |\{(s, t) \ : \ \text{the edge } \{i, j\} \text{ is on a shortest path from } s \text{ to } t\}|. \tag{2.1.6}$$

Edges inside communities are expected to have low bottleneck-ness, as there are many alternative routes through the well-connected communities, while edges between communities tend to have larger bottleneck-ness. In Exercise 9 you can practise with computing the bottleneck-ness in a concrete example of a network.

2.2. Finding communities in networks

In the previous section, we have described what we expect communities to look like. Each of these different ideas about what a community is, can be turned into an algorithm for detecting communities in networks.

For simplicity, we will assume that each node is part of *exactly* one community. Therefore, we will look at algorithms that find a *partition* of the nodes into communities. There also exist algorithms that detect *overlapping* communities, or that simply search for the community around a given node, but we will not discuss these variants of community detection in this masterclass.

2.2.1. Detecting communities with small radius
One of the simplest community detection methods is k*-center* and it is based on the notion that communities should have a small radius. Given a network and a number k (the number of communities we want to detect), we randomly designate k nodes as 'community centers'. Then, for each of the nodes that is not a center, we assign it to the community corresponding to the nearest center. After this, we update the center of each community to the node that has the lowest eccentricity of the community. Such nodes are relatively central because they have the least maximum distance from all other nodes in the community. Note that for each of the nodes that is not a center the nearest center might change after this step. We repeat these last two steps (assigning each node to the nearest center and updating the center) as long as the partition keeps changing. See Exercise 8 in Chapter 4 for an implementation of this algorithm.

2.2.2. Divisive community detection
Divisive community detection is based on the notion that there are relatively few connections between communities, while there are plenty of edges inside communities. Thus, if we eliminate some of the 'less important' edges, we are likely to disconnect communities from each other, while the communities will likely remain connected.
This does, of course, require us to come up with a way to decide which edges are more important than others. An example of a suitable measure for this, is the *bottleneck-ness* that we discussed earlier in (2.1.6): an edge that is on many shortest paths is likely to form a 'bridge' between two communities. After deleting the edge with the highest bottleneck-ness, we can re-compute the bottleneck-ness and delete a new edge. We can keep doing this until we reach some stopping criteria. For example, until we have deleted a predetermined number of edges or until we are left with a certain number of communities.
Another measure that can be used for divisive community detection is the number of common neighbors that two nodes have. If two nodes are connected to many common neighbors, then they are more likely to be part of the same community. Thus, we could at each point eliminate the edge that connects two nodes that have the *least* number of common neighbors.

2.2.3. Maximizing partition quality
The previous algorithms provide procedures to find reasonable partitions of network nodes into communities. However, many of these algorithms involve some randomness, so if we

would run them again, we might find different partitions into communities. The methods themselves do not offer a way of quantifying which of these possible partitions is best. Most community detection methods that are currently popular define a *quality function* that quantifies how well a given partition into communities fits the given network. This way, if we have two algorithms that result in different partitions, we simply select the partition that has the highest quality (according to this measure) and consider this as the best guess of what the community structure of the network is. Furthermore, instead of using any of the previously described procedures, we might as well use an algorithm that optimizes the quality function directly.

While many of the measures that we discussed in Section 2.1 quantified how 'community-like' a group of nodes S is with respect to the network, the quality functions that we discuss in this section quantify how well the *entire partition into communities* fits the network as a whole. Many of these community measures can easily be turned into quality functions by simply summing over the communities. For example, we could simply take the sum of the cut sizes

$$\text{CutQuality}(C) = \sum_{i=1}^{|C|} \text{Cut}(C_i). \tag{2.2.1}$$

If we were to search for the partition that minimizes this, without constraining the number of clusters, the optimal partition would always consist of a single community. Conversely, if we would minimize the sum of the community radii, i.e.,

$$\text{RadiusQuality}(C) = \sum_{i=1}^{|C|} \text{Radius}(C_i), \tag{2.2.2}$$

then the optimum would always be to assign each node to a community of size 1, since every node has distance i to itself. The easiest way to resolve this, is to restrict the number of communities to some fixed number k and to optimize the quality over all partitions consisting of k communities.

Modularity. However, not all quality functions require us to restrict the number of communities in order to avoid 'obvious' optima. *Modularity* is one example of such quality functions and it is currently one of the most widely-used community detection methods. Modularity measures the number of edges inside the communities and compares it to the number of edges we would expect inside communities if they were *randomly placed* according to some model without community structure. The idea behind this, is that if there are much more edges inside the communities than we would expect according to this model without communities, then the found communities must be *significant* in some way. There are different ways to do this random placing, leading to different versions of modularity. The easiest way to do this is to randomly place the $|E|$ edges among the $\binom{|N|}{2}$ possible node pairs. This way, each node pair has a probability $|E|/\binom{|N|}{2} = \text{Density}(N)$ of having an edge after shuffling, so that the expected number of edges inside a community of size s is given by $\binom{s}{2} \cdot \text{Density}(N)$. We refer to the corresponding modularity as *Simple Modularity* and it is given by

$$\text{SimpleModularity}(C) = \frac{1}{|E|}\left(\sum_{i=1}^{|C|} \ell(C_i) - \binom{|C_i|}{2} \cdot \text{Density}(N)\right). \tag{2.2.3}$$

Note that if we place all nodes together into one community, the resulting Simple Modularity is zero since $\ell(C_1) = |E| = \text{Density}(N) \cdot \binom{|C_1|}{2}$. Similarly, if we place each node i into a community of size one (that is, $C(i) = \{i\}$), the Simple Modularity will also be zero since $\ell(\{i\}) = 0 = \binom{1}{2}$. The optimal modularity value, will thus correspond to some partition C with $1 < |C| < |N|$. This shows why modularity does not require us to fix the number of communities.

Taking degrees into account. Another model for the random placement of edges also takes the degrees of the nodes into account. Nodes with higher degrees are more likely to be connected to each other, simply because they have more edges around them. Because of this, algorithms that maximize Simple Modularity have the tendency to group together nodes of high degree. There are many ways to randomly place the edges in a way that the degree of each node in this random graph will roughly correspond to the degree that this node has in the original network. Most of these do (roughly) lead to the same version of modularity, which is why we will explain the simplest version here. Between two nodes i and j, with degrees $d(i)$ and $d(j)$ in the original graph, we place an edge with probability $\frac{d(i)d(j)}{2|E|}$. This way, the expected degree of node i in this random network is

$$\sum_{j \in N \setminus \{i\}} \frac{d(i)d(j)}{2|E|} = d(i)\frac{2|E| - d(i)}{2|E|} \approx d(i),$$

as long as $|E|$ is large compared to $d(i)$. For a group of nodes S, the expected number of edges inside S is given by

$$\frac{1}{2}\sum_{i \in S}\sum_{j \in S \setminus \{i\}} \frac{d(i)d(j)}{2|E|} = \frac{1}{4|E|}\left(d(S)^2 - \sum_{i \in S} d(i)^2\right).$$

The term $\sum_{i \in S} d(i)^2$ does not depend on the partition into communities, so it can be ignored. This leads to the following version of modularity

$$\text{StandardModularity}(C) = \frac{1}{|E|}\left(\sum_{i=1}^{|C|} \ell(C_i) - \frac{d(C_i)^2}{4|E|}\right). \tag{2.2.4}$$

We refer to this version of modularity as *Standard Modularity*, as it is the most widely-used form of modularity. Maximizing this quantity may even be the most widely-used community detection algorithm.

Modularity does have one infamous defect, known as the *resolution limit*. We illustrate this defect in Exercises 10 and 11 using the *ring of cliques* network. These two exercises demonstrate that any modularity-maximizing algorithm will merge together neighboring cliques if the ring is large enough, which is clearly undesirable. This problem is often referred to as the *resolution limit* of modularity. The problem is that when the number of nodes in the network is much higher than the average degree, the number of edges becomes so small that the density becomes negligible. Then, regardless of the way you randomly place the edges, the expected amount of edges between communities becomes negligible, so that *any* edge between communities will already result in 'significantly more edges than expected'.

2.3. Models of communities

In this section we discuss several models of what graphs with community structure look like. Let $C(i)$ denote the community that node $i \in N$ belongs to.

2.3.1. Planted Partition Model

The simplest way to model community structure is to simply assume that nodes of the same community have a higher probability of being connected than nodes of different communities. That is, we say that the probability that two nodes are connected by an edge equals p_{in} if they are of the same community, and $p_{\text{out}} < p_{\text{in}}$ otherwise. That is,

$$\mathbb{P}(\{i, j\} \in E) = \begin{cases} p_{\text{in}} & \text{if } C(i) = C(j), \\ p_{\text{out}} & \text{if } C(i) \neq C(j). \end{cases}$$

This model is called the *Planted Partition Model* and it is often abbreviated as PPM. An example of a small PPM network is shown in Figure 2.3.1.

2.3.2. PPM with different community sizes

In the above, we chose each community to have the same size. However, if there are communities with vastly different sizes while we keep the values p_{in} and p_{out} constant, then this will lead to nodes that are part of large communities having very high degrees. For example, suppose $p_{\text{out}} = p$ and $p_{\text{in}} = 10p$, and consider a PPM with 11 communities, 10 of size s and one of size $10s$. The expected degree of a node in a small community will be $(10 + 9)sp + 10(s - 1)p \approx 29sp$ (try to prove this formula, the first term corresponds to the average number of neighbors outside the small community, the second term to the average number of neighbors inside the small community), while a node in the large community will have degree $10sp + (10s - 1) \cdot 10p \approx 110sp$ (try to prove this formula using a similar reasoning as above), thus the degrees in the large community are much larger than the degrees

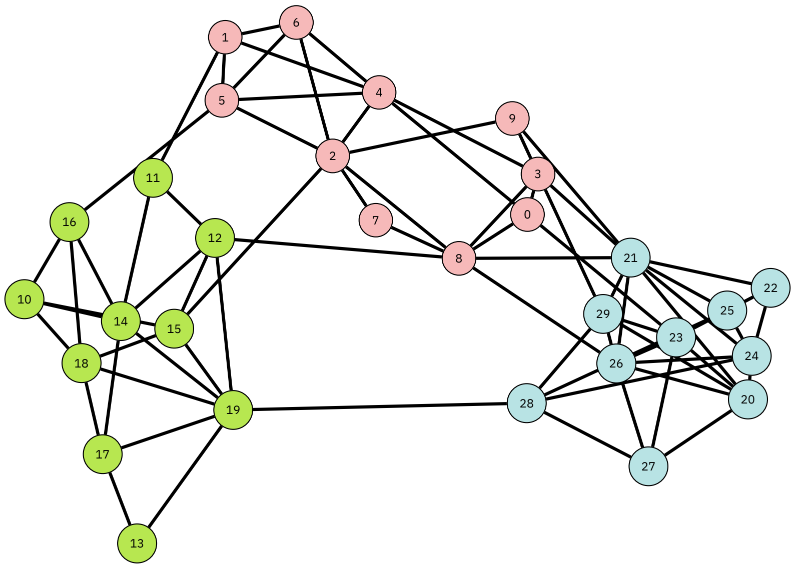


Figure 2.3.1. An example of a PPM consisting of three communities of size 10 each. We take the probabilities $p_{in} = \frac{1}{2}$ *and* $p_{out} = \frac{1}{40}$.

in the small communities. What's perhaps even more problematic, is that a node in a small community is expected to have almost twice as many neighbors outside its community as inside while a node in the large community has almost 10 times more neighbors inside the community than outside. For this reason, we need to modify this model if we want to allow for different community sizes.

To make sure that the expected degree does not depend too much on the size of the community, we change the probability that two nodes i and j inside a community of size s are connected to

$$p_{\text{in}}(s) = \frac{d_{\text{in}}}{s-1},$$

so that each node is connected to d_{in} community members in expectation. The expected number of neighbors outside the community is given by

$$(|N| - s)p_{\text{out}},$$

which unfortunately does depend on the size of the community. However, this is barely noticeable as long as s is relatively small compared to the total number of nodes $|N|$. Usually one chooses

$$p_{\text{out}} = \frac{d_{\text{out}}}{|N|}$$

so that each node is expected to be connected to approximately d_{out} nodes outside the community. We call this model the *Sparse PPM*. The difference between PPM and Sparse

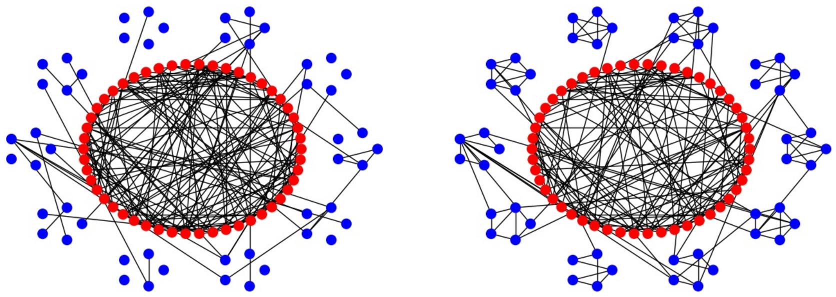


Figure 2.3.2. Comparison of PPM (left) and Sparse PPM (right) for the case where the communities have different sizes. The parameters of the PPM follow the setup as described in Section 2.3.2, for $s = 5$ and $p = \frac{1}{10}$, while the d_{in} and d_{out} of the Sparse PPM are chosen to match the PPM as much as possible.

PPM is demonstrated in Figure 2.3.2, where we have one community of size 50 and 10 communities of size 5. For the PPM, we see that the nodes in the small communities have much smaller degrees than vertices in the large community, and some of the small communities don't even have any edges inside them. For the Sparse PPM, in contrast, we see that the small communities are much better connected.

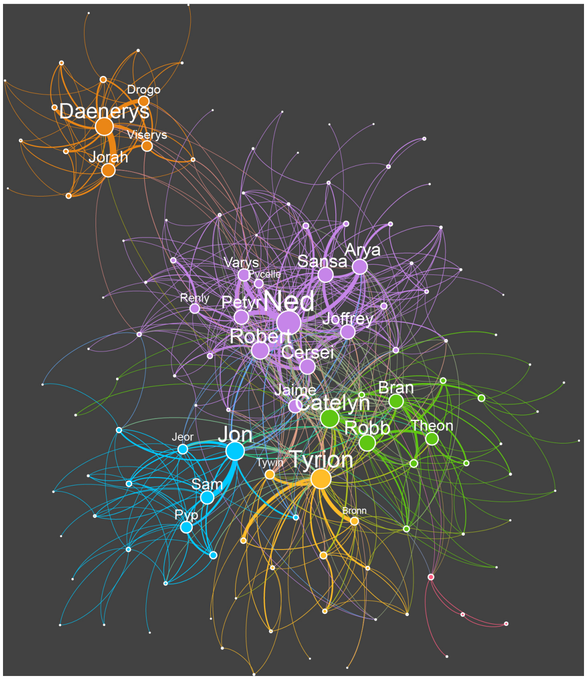


Figure 2.3.3. Mathematical rulers in Game of Thrones! By Clara Stegehuis.

On the Network Pages

For further reading on networks and communities have a look at:

(1) *Mathematical rulers in Game of Thrones* by Clara Stegehuis,

networkpages.nl/mathematical-rulers-in-game-of-thrones/.

(2) *Synchronization in the body clock* by Janusz Meylahn,

networkpages.nl/synchronization-in-the-body-clock//.

(3) *The Network Science of Echo Chambers and Why It Matters* by Maurik Engelbert van Bevervoorde, Maurits Flos, Ana-Maria Olteniceanu, and Riccardo Torlaini

networkpages.nl/the-network-science-of-echo-chambers-and-why-it-matters//.

(4) *How the popular become even more popular* by Remco van der Hofstad,

networkpages.nl/how-the-popular-become-even-more-popular/.

Chapter 3

Road traffic analysis - Route selection in a network

A road traffic network consists of a collection of roads that connect various cities. The users of this network want to travel from their current location, called the *origin*, to some desired *destination*. These users travel in a vehicle and will therefore be called the drivers in the network.

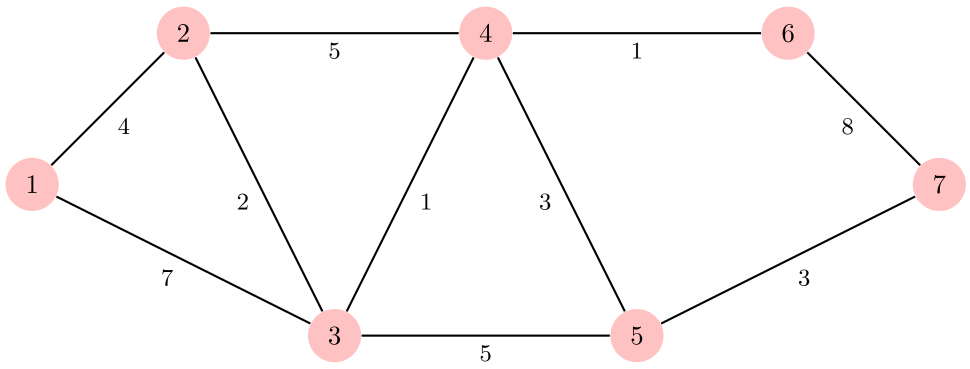
Companies that provide route guidance to the drivers in a network need to keep the potential wishes of these drivers into account. For example, instead of simply using the fastest route, a driver who has ample time and who is concerned about the fuel consumption of his car may prefer the shortest route. Moreover, route guidance companies also need to make sure that the algorithms they use to compute these routes are *efficient*, i.e. the computation can be done in a reasonable amount of time. A driver is not likely to use a route planner if it takes a lot of time for this planner to compute the best route for her. A lot of studies have tried and succeeded in improving Dijkstra's algorithm in either of these aspects:

(1) speeding up Dijkstra's algorithm and
(2) extending Dijkstra's algorithm beyond the simple shortest path (i.e. in distance) setting.

3.1. Efficiency

In Section 1.3, Dijkstra's algorithm is used to compute a (shortest) route in the network of Figure 1.3.2, also given below.

Even though this network contains only seven nodes, execution of the algorithm already takes six steps, each of them consisting of multiple actions. Therefore, one can imagine



A network with seven locations as in Figure 1.3.2

that this algorithm might be less useful in real-life vehicle networks, which are of very large scale. Planners use several speed-up techniques to make sure drivers do not have to wait long before learning their optimal route, the shortest path from the origin to the destination, in such large networks.

Before analyzing two of these speed-up techniques, it is crucial to understand what makes Dijkstra's algorithm limited in large networks. Recall that at the end of each iteration step, we mark the node with the lowest distance in the row that corresponds to the step as permanent. This generally means that nodes that lie closer to the origin will be marked permanent before nodes that lie further away from the origin. Notably, in this procedure, we do not take the location of the destination into account. It is even possible that we may mark some nodes which lie further away from the destination than the origin itself. By explicitly working with the distance of nodes to the destination as well, the two considered speed-up techniques are able to be more efficient than Dijkstra's algorithm.

3.1.1. Bi-directional Dijkstra

In the bi-directional Dijkstra algorithm, we perform the same steps as in Dijkstra's algorithm: starting with the origin as permanent node, iteratively update the distances for the neighbors of the permanent node, each time picking the node with the updated shortest distance as new permanent node. However, at the same time, we also start Dijkstra's algorithm with the destination as first permanent node. Thus, Step 1 of the bi-directional Dijkstra algorithm executes Step 1 of Dijkstra's algorithm twice: once with the origin as starting point, once with the destination as starting point. In terms of bookkeeping, in the example given above, this gives us the first row of Table 3.2.1 and the first row of Table 3.1.2. Then, in each step of the algorithm, we execute Dijkstra's algorithm twice, each time creating a new row in both tables. Thus, besides updating around the origin, we now update around the destination as well.

nodes	1	2	3	4	5	6	7
Step 1	-	$(4,1)^*$	$(7,1)$	No edge	No edge	No edge	No edge
Step 2	-	-	$(6,2)^*$	$(9,2)$	No edge	No edge	No edge
Step 3	-	-	-	$(7,3)^*$	$(11,3)$	No edge	No edge
Step 4	-	-	-	-	$(10,4)$	$(8,4)^*$	No edge
Step 5	-	-	-	-	$(10,4)^*$	-	$(16,6)$
Step 6	-	-	-	-	-	-	$(13,5)^*$

Table 3.1.1. Dijkstra's shortest route algorithm for the network in Figure 1.3.2.

nodes	1	2	3	4	5	6	7
Step 1	No edge	No edge	No edge	No edge	$(3,7)^*$	$(8,7)$	-
Step 2	No edge	No edge	$(8,5)$	$(6,5)^*$	-	$(8,7)$	-
Step 3	No edge	$(11,4)$	$(7,4)$	-	-	$(7,4)^*$	-
Step 4	No edge	$(11,4)$	$(7,4)^*$	-	-	-	-
Step 5	$(14,3)$	$(9,3)^*$	-	-	-	-	-
Step 6	$(13,2)^*$	-	-	-	-	-	-

Table 3.1.2. Backward Dijkstra's shortest route algorithm for the network in Figure 1.3.2.

You may be right to wonder how executing the algorithm twice makes the procedure more efficient. First, let us remark that computers are able to do computations *in parallel*, such that they can simultaneously compute a new row for both tables, without much additional cost as compared to the computation of one new row. Second, it is very important to realize that by comparing the numbers in the two generated tables after each step, we are able to determine the shortest path at an earlier stage, and therefore need lesser steps than the original Dijkstra algorithm.
If we would, for example, consider Tables 3.2.1 and 3.1.2, we see that after Step 2, by combining the entries for node 4 and node 3, we respectively find paths of lengths 15 and 14. Then, after Step 3, we may again combine table entries for the different nodes in the network, and find, via node 3, an even shorter path of length 13. Remarkably, it turns out that, as the sum of the distance entries of the new permanent nodes exceeds 13 (i.e., 7 for node 4 in Table 3.2.1 and 7 for node 6 in Table 3.1.2), there cannot be a path that is shorter than the current shortest path.

3.1.2. A-star algorithm

Intuitively, the A-star algorithm makes sure that the updating of the nodes does not happen solely around the origin, but that the nodes whose values are updated are mostly located in the area between the origin and the destination. To make this happen, the algorithm does not only work with the distances from the origin to the nodes, but also with lower bounds for remaining distances, i.e., the distances between the nodes and the destination. These lower bounds may e.g. be the geographical distances, which are the lengths of the direct lines from the nodes to the destination.

nodes	1	2	3	4	5	6	7
Step 1	No edge	$(5,4)$	$(1,4)^*$	-	$(3,4)$	$(1,4)$	No edge
Step 2	$(8,3)$	$(3,3)$	-	-	$(3,4)$	$(1,4)^*$	No edge
Step 3	$(8,3)$	$(3,3)$	-	-	$(3,4)^*$	-	$(9,6)$
Step 4	$(8,3)$	$(3,3)^*$	-	-	-	-	$(6,5)$
Step 5	$(7,2)$	-	-	-	-	-	$(6,5)^*$

Table 3.1.3. Dijkstra's algorithm for the network in Figure 1.3.2 for a path from 4 to 7.

nodes	1	2	3	4	5	6	7
lower bound	9	8	7	6	4	2	0
Step 1	No edge	$(13,5,4)$	$(8,1,4)$	-	$(7,3,4)$	$(3,1,4)^*$	No edge
Step 2	No edge	$(13,5,4)$	$(8,1,4)$	-	$(7,3,4)^*$	-	$(9,9,6)$
Step 3	No edge	$(13,5,4)$	$(8,1,4)$	-	-	-	$(6,6,5)^*$

Table 3.1.4. A-star algorithm for the network in Figure 1.3.2 for a path from 4 to 7.

We demonstrate the A-star algorithm with an example. Say that we want to travel from node 4 to node 7 in the network of Figure 1.3.2. Then, Table 3.1.3 corresponds to Dijkstra's procedure. As can be seen, the algorithm is inefficient, as it also updates nodes that lie on the other side of the origin as the destination, such as nodes 1 and 2. If we would, however, have lower bounds on the distances from the nodes to the destinations, as given in Table 3.1.4, the efficient A-star algorithm may be executed. This algorithm, whose procedure is shown in the same table, is very similar to Dijkstra's algorithm, in the sense that it is an iterative procedure that marks nodes and updates values of its neighbors. However, for the bookkeeping, different values are used. Specifically, besides the shortest distance to the node and the neighbor via which we should then travel, we store an estimate of the total distance on a path containing this node: the sum of the shortest distance to this node and the known lower bound on the remaining distance to travel. For example, entry $(13,5,4)$ in Step 1 for node 2 arises because the current shortest distance to node 2 is reached via node 4 and has value 5, and adding the lower bound to this distance yields that value 13. Now, in each iteration, the node that is permanently marked is the node with lowest first entry. Again, the algorithm terminates if the marked node is the destination, which already happens after three steps.

3.2. Accuracy

Using (a speed-up version of) Dijkstra's algorithm, it is not difficult to determine the length of the shortest route that connects the origin and the destination, even in the large scale. The length of the individual road segments that make up the route are fixed, and the total travel distance is simply the sum of these segments. We can therefore say that the distance between origin and destination is a deterministic quantity. The travel time between origin and destination, however, is certainly not deterministic. For example, you may get stuck in a traffic jam as a result of an accident which will drastically increase your travel time. One could also think of smaller hindrances, such as having to wait for a bridge that is opening or repeatedly having to stop for red lights. Some routes will be more likely to cause delays than others. These routes are said to bear a higher risk related to the travel time. Therefore, in contrast to travel distances, travel times are stochastic.
Since travel times are stochastic, it is important to realise that the fastest route suggested by your navigation system is only the fastest route in expectation. It may be the case that this route consists of roads that are likely to cause delays. As a result, the actual travel time of this route may be very uncertain. It is because of this uncertainty that the fastest route is not always the most desirable route. We inspect if we can still use Dijkstra's algorithm to find routes that minimise other characteristics.

3.2.1. Reliable routes

Consider a driver that is not interested in finding the shortest route, but who wants to find the route that minimises the probability of getting stuck in a traffic jam. Note that this route may be a big detour from the shortest route and is therefore unlikely to be the fastest route. We call this route the most reliable route. The following is based on Example 6.3-2 in the book "*Operations Research, An Introduction*" (9th Edition) by Hamdy A. Taha.
In Figure 3.2.1 we assigned to each edge $\{i, j\}$ of Figure 1.3.2 a probability p_{ij} of not running into a traffic jam on the road between city i and j. For example, $p_{12} = 0.2$ and $p_{35} = 0.5$. Note that there is no direct edge between city 1 and 7, but the route $1 \to 2 \to 4 \to 6 \to 7$ is a possible route between those cities and the probability of not running into a traffic jam on this route is

$$p_{17} = p_{12} \times p_{24} \times p_{46} \times p_{67} = 0.2 \times 0.8 \times 0.35 \times 0.7 \approx 0.04.$$

Hence, if the driver chooses this route, there is a probability of not running into a traffic jam of only 4%. This does not look very promising indeed! Perhaps we are able to find a route on which it is less likely to get stuck in traffic?
This problem can be formulated as a shortest route model by using a logarithmic transformation. This way, we can convert the product of probabilities into a sum of logarithms

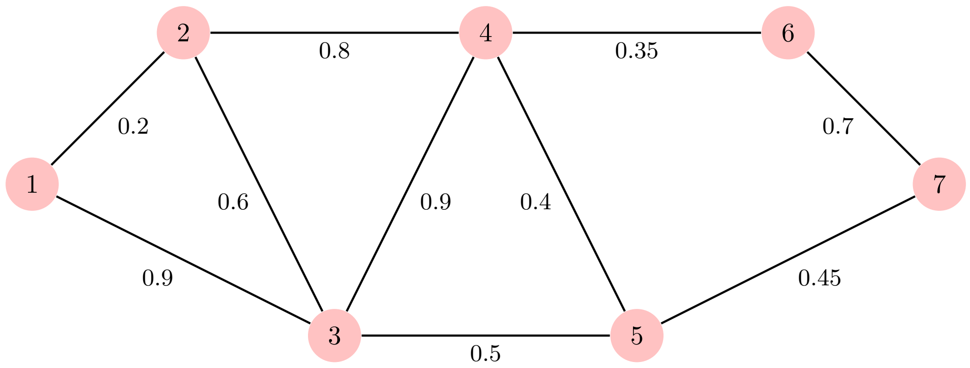


Figure 3.2.1. Most reliable route network model

of probabilities. That is, the probability assigned to our previously suggested route is transformed to

$$p_{17} = p_{12} \times p_{24} \times p_{46} \times p_{67} \implies \log p_{17} = \log p_{12} + \log p_{24} + \log p_{46} + \log p_{67}.$$

If we are able to find a route that maximises $\log p_{17}$, this same route would also maximise the actual probability p_{17} of not running into a traffic jam. This is due to the fact that the logarithm is a strictly increasing function. Note that Dijkstra's algorithm is designed to find a route that minimises a sum instead of maximising it. However, this problem can easily be countered by minimising $-\log p_{17}$. In Figure 3.2.2 we have replaced each p_{ij} by $-\log p_{ij}$. We have now successfully converted our problem to the shortest route problem which we looked at in Section 1.3, since the shortest route of the network in Figure 3.2.2 corresponds to the most reliable route in the sense that this route has the highest probability of not running into a traffic jam.

To find the most reliable route, we thus use the weights in Figure 3.2.2 and act as if they are distances. Making a table for the weights using Dijkstra's algorithm, in the same way as we did in Section 1.3, leads to Table 3.2.1.

nodes	2	3	4	5	6	7
Step 1	$(1.609, 1)$	$(0.105, 1)^*$	No edge	No edge	No edge	No edge
Step 2	$(0.616, 3)$	-	$(0.210, 3)^*$	$(0.798, 3)$	No edge	No edge
Step 3	$(0.433, 4)^*$	-	-	$(0.798, 3)$	$(1.260, 4)$	No edge
Step 4	-	-	-	$(0.798, 3)^*$	$(1.260, 4)$	No edge
Step 5	-	-	-	-	$(1.260, 4)^*$	$(1.597, 5)$
Step 6	-	-	-	-	-	$(1.597, 5)^*$

Table 3.2.1. Dijkstra's shortest route algorithm for the network in Figure 3.2.2.

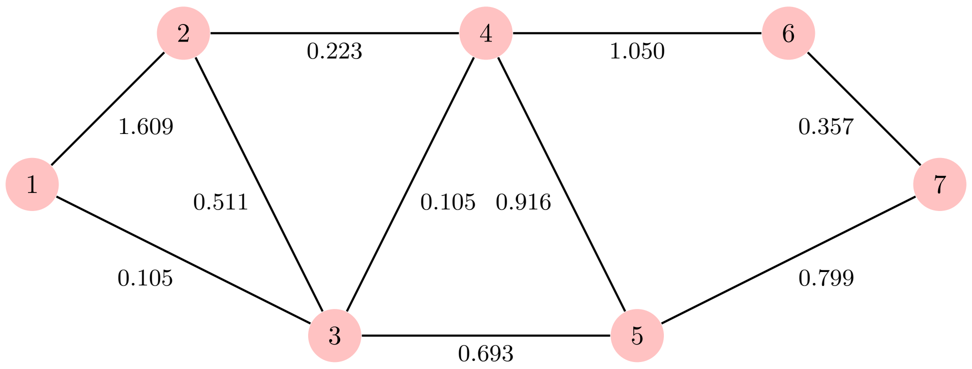


Figure 3.2.2. Most reliable route representation as a shortest route model

From this table, we see that the most reliable route from city 1 to city 7 is the route $1 \to 3 \to 5 \to 7$. As a side note, if we would like to go from city 1 to city 2 instead, the most reliable route is not the direct route $1 \to 2$, but by following the records in the table, the most reliable route in this case would be $1 \to 3 \to 4 \to 2$. When looking at Figure 3.2.1, this seems right: the direct probability 0.2 of not running into a traffic jam is indeed smaller than the probability $0.9 \times 0.5 \times 0.45 = 0.2025$ of the route found by Dijkstra's algorithm!
Note that we could have used the bi-directional Dijkstra algorithm here as well, but that it would not have been possible to use A-star, as we do not have lower bound on the logarithms of the probabilities in the network.

3.2.2. Refinements using probability distributions

So far, we have seen that we can use Dijkstra's algorithm in order to both find the shortest and the most reliable route. However, these routes may both be undesirable for a user of the road traffic network. The shortest route may be a very slow route due to congestion, whereas the most reliable route may be a big detour from the destination, resulting in a long travel time.
A much more realistic objective that a driver may have is that they want to arrive at their destination 'on time'. For example, consider the situation where a driver has a meeting in one hour and they want to maximise the probability of not being late. Perhaps the most reliable route is the best choice, as the low uncertainty ensures that the driver will arrive on time. Another extreme is the situation in which a high risk route, meaning a short route which is likely to be congested, is the only route which gives a chance of arriving on time. Of course, in this case the high risk route would be the best route. But how do we determine the best route mathematically?
Instead of looking at the length or the reliability of a road between two cities, a more relevant quantity would be the travel time between these two cities. We have already argued

before that the travel time is a stochastic quantity. This is where probability distributions come in. Routes that are expected to have a short travel time will have a probability distribution that is centred around a relatively low value. Unreliable routes, i.e. routes on which there is high uncertainty over the travel time, will have a probability distribution that is more spread.

Let's go back to the situation in Figure 3.2.1. The edge between city 1 and 3 indicates that there is a low probability of getting stuck in a traffic jam on this road. Therefore, the travel time to move between city 1 and 3 will be fairly certain and its probability distribution will be relatively concentrated. In contrast, the road between city 1 and 2 is likely to be congested making the travel time between these two cities rather uncertain. Hence, this probability distribution will be more spread, or equivalently, the travel time will have a higher variance.

For simplicity, we will assume that the travel times are distributed according to a normal distribution. An objection to this assumption is that the normal distribution also assigns positive probability to negative values, which does not make sense in our application. However, this distribution is intuitive to work with and it allows us to do some explicit computations without the use of a computer.

In Figure 3.2.3 we once again consider the same road traffic network, but we are now concerned with the random travel times between the cities instead of the probability of getting stuck in a traffic jam. We let the random variable T_{ij} denote the travel time between city i and j and we write $T_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$ to indicate that T_{ij} is normally distributed with mean μ_{ij} and variance σ_{ij}^2. Possible densities for the travel time distribution between city 1 and 3 and city 1 and 2 that would agree with the previous discussion are given in Figure 3.2.4. What is interesting to note here is that even though $13 = \mu_{12} < \mu_{13} = 15$, inspection of these densities seems to imply that

$$\mathbb{P}(T_{12} \geq 20) > \mathbb{P}(T_{13} \geq 20). \tag{3.2.1}$$

It is remarkable that the road between city 1 and 2 has a lower mean travel time compared to the route between city 1 and 3, but at the same time this route is more likely to take more than 20 minutes to traverse. This is of course caused by the higher variance of T_{12}. This example demonstrates that the expected fastest route, which is the route that is most likely to be suggested by a navigation system, is not necessarily the best route for drivers that want to maximise the probability of arriving at their destination on time. If we want to determine the best route for these drivers, it is crucial that we know the probability distribution of the travel times.

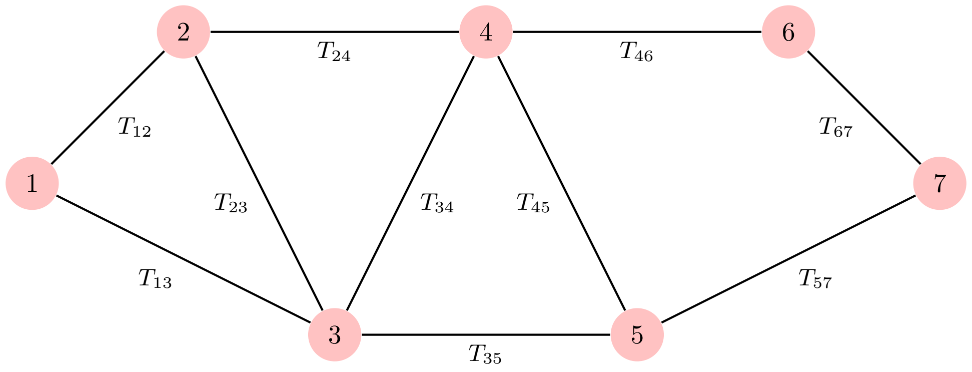


Figure 3.2.3. Road traffic model with stochastic travel times

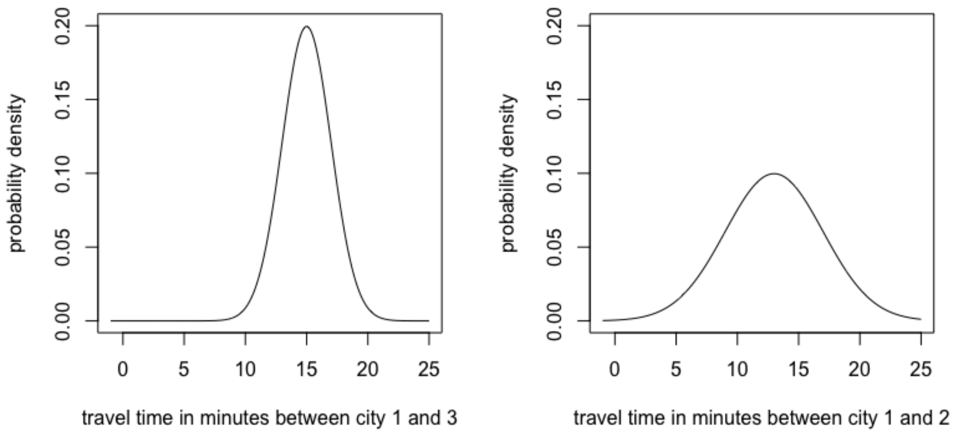


Figure 3.2.4. The density of $T_{13} \sim \mathcal{N}(15, 4)$ (left) and $T_{12} \sim \mathcal{N}(12, 16)$ (right).

3.3. Concluding remarks

We now know that the road traffic network in Figure 3.2.1 does not tell the whole story. Instead of only knowing the expected travel times and the reliability of a road network, knowing the actual probability distribution of the travel times would provide us with much more information. This in turn allows us to answer questions that are of greater relevance for some drivers.

The model we discussed in which the travel times are assumed to be independent and normally distributed could already be fruitful in practice in order to find routes that are both fast and reliable. However, there is room for improvement as the assumptions we made are not very realistic.

For one, we have already argued that travel times cannot be normally distributed. This is due to the normal distribution assigning positive probability to negative values. In other words, if travel times are normally distributed, there is a positive probability of having a negative travel time. Nonsense! Therefore, it is better to assume that the travel times follow some non-negative distribution such as the log-normal distribution or the gamma distribution. Even worse, it can happen that none of the well-known probability distributions provide a good explanation of the actual travel times. In this case we would have to resort to so-called non-parametric methods. Another assumption we made is that the travel times are independent across roads. This means that any information regarding the travel time of one road has no impact on the travel time of any other road. However, one could argue that the level of congestion of a road is positively correlated with the level of congestion of the adjacent roads. This violates the assumption, since the level of congestion clearly has an impact on the travel times. These issues greatly complicate the analysis of the network.

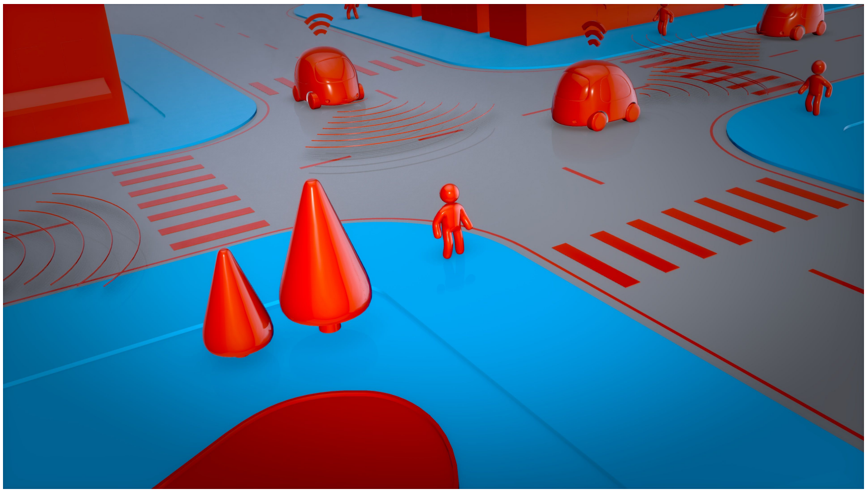


Figure 3.3.1. Reducing travel times can be achieved in multiple ways, finding the optimal route is one of them as we discussed. But there is more! You can also monitor traffic, that is what traffic lights do for example. Have a look at the article of Rik Timmerman below.

On the Network Pages

For more information on algorithms, networks and road traffic analysis have a look at:

(1) *Finding the shortest route to your holiday destination: Dijkstra's algorithm* by Bart Jansen,

networkpages.nl/finding-the-shortest-route-to-your-holiday-destination-iv-dijkstra-algorithm/.

(2) *How to plan Valentine's day using a matching algorithm* by Bart Jansen,

networkpages.nl/how-to-plan-valentines-day-using-a-matching-algorithm/.

(3) *Traffic Congestion: Braess' Paradox* by Peter Kleer,

networkpages.nl/traffic-congestion-iv-braess-paradox/.

(4) *Traffic lights no longer needed: back to the future* by Rik Timmerman,

networkpages.nl/traffic-lights-no-longer-needed-back-to-the-future/.

Chapter 4

Exercises

Exercises on graph theory

EXERCISE 1. Show that in a graph $G = (V, E)$ the maximum number of edges is equal to $\binom{|V|}{2} = \frac{|V| \cdot (|V|-1)}{2}$.

EXERCISE 2. Show that you can construct in total $2^{\binom{|V|}{2}}$ graphs with $|V|$ nodes.

EXERCISE 3. Look at the karate network in Figure 2.0.1. Answer the following questions.

- What are the degrees of points 0, 1, 8, and 33?
- What are the eccentricities (see 2.1.4) of points 0, 1, 8, 15, 16, and 33? For these computations consider S either as the pink or the green community given in the Figure.
- What are the diameter (see (2.1.5)) and the radius (see (2.1.5)) of the two subnetworks? Can you give an intuitive explanation of the values you get?

EXERCISE 4. Look at the FIFA World Cup from 2022 in Figure 2.0.2. Answer the following questions.

- How would you divide the network in communities in order to obtain the lowest cut-size (see (2.1.1))?
- How would you divide the network in communities in order to obtain the lowest RatioCut value between communities (see (2.1.2))?

Exercises on networks and communities

EXERCISE 5. What is the size of the largest clique of the FIFA2022 network of Figure 2.0.2?

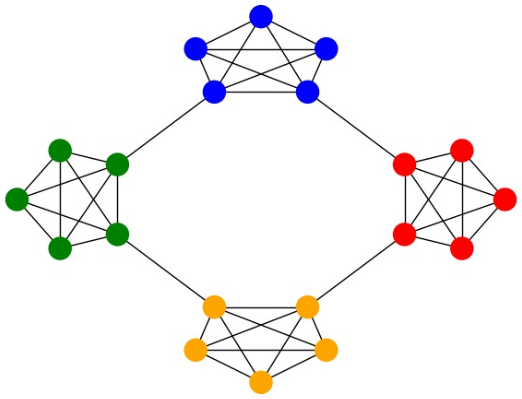


Figure 4.0.1. The ring-of-cliques network for $k = 4$ and $s = 5$.

EXERCISE 6. In the karate network, the conflict that led to the split (called 'true split') was a conflict between nodes 0 and 33 (the instructor and the administrator). Thus, we want to find a split into two communities such that the instructor and the administrator are in different communities. What is the cutsize, see (2.1.1), of the 'true split'? Can you find such a split that has a lower cutsize than the 'true split'?

EXERCISE 7. The diameter of a circle is always twice the radius. Does this also hold for graphs? Experiment with some simple graphs. You may assume that the graph is connected.

EXERCISE 8. Apply the k-center algorithm to the Karate network from Figure 2.0.1. In the first iteration, you can assign nodes 0 and 33 (the instructor and administrator) as the centers. In case you encounter ties (that is, a node that is at equal distance of both centers, or multiple nodes with equal eccentricity), you can resolve them by choosing the node with the lowest index.

EXERCISE 9. Consider a 'line' network of an even number of nodes. That is, nodes are labeled $1, 2, \dots, n$ (where n is even) and nodes are connected if their labels differ by exactly one. Which edge has the highest bottleneck-ness of this network?

In the following exercises, we consider a network with k equally-sized communities $C_1, \dots, C_k$ (each of size $s > 2$). Each of the communities forms a clique while each two neighboring communities (i.e., community C_r and C_{r+1} for $r = 1, \dots, k-1$ or C_k and C_1) are connected by a single edge. We refer to this network as the *ring-of-cliques* network. Figure 4.0.1 shows the corresponding network for $k = 4$ and $s = 5$.

EXERCISE 10. What is the Simple Modularity of this ring of cliques?

EXERCISE 11. Now consider the partition that is obtained by when combining clique C_1 and C_2 into one community of size $2s$. Show that for k sufficiently large, the simple modularity of this new partition is actually *higher* than that of the original, more sensible partition.

EXERCISE 12. For a PPM consisting of k communities each of size s, what is the expected degree of a node? And what is the expected density of the network as a whole?

EXERCISE 13. For a PPM consisting of k communities each of size s, what is the expected value of the Simple Modularity of the 'true' community structure?

EXERCISE 14. For a PPM consisting of k communities each of size s, what is the probability that a given node i is not connected to any of its community members? What is the expected number of nodes in the network that are not connected to any of their community members?

EXERCISE 15. What is the expected radius of a community consisting of s nodes?

EXERCISE 16. What is the expected cutsize between two communities? And what about the ratio cut?

EXERCISE 17. We say that a node s is a common neighbor of i and j it is a neighbor of both i and j. That is, whenever $\{i, s\} \in L$ and $\{s, j\} \in L$. What is the expected number of common neighbors that two nodes have if they are in the same community? And what is the expected number of common neighbors for two nodes that are not in the same community?

EXERCISE 18. Consider a PPM (not sparse) with $p_{\text{in}} = \frac{1}{2}$ with two communities with sizes s_1 and s_2, where $2 \leq s_1 < s_2$. Which of the two communities has a higher expected number of nodes that are not connected to the rest of the community?

Exercises on probability theory – Normal Distribution

EXERCISE 19. Verify the claim in (3.2.1). Recall that $T_{12} \sim \mathcal{N}(12, 16)$ and $T_{13} \sim \mathcal{N}(15, 4)$. Use Table 4.0.4 on page 52.

EXERCISE 20. Let X and Y be continuous random variables in $\mathbb{R}$. Show that

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y].$$

You may use that $\int_{\mathbb{R}} f(x, y)\mathrm{d}y = f(x)$ and $\int_{\mathbb{R}} f(x, y)\mathrm{d}x = f(y)$.

Exercises on road traffic analysis

Dijkstra's algorithm for reliable routes

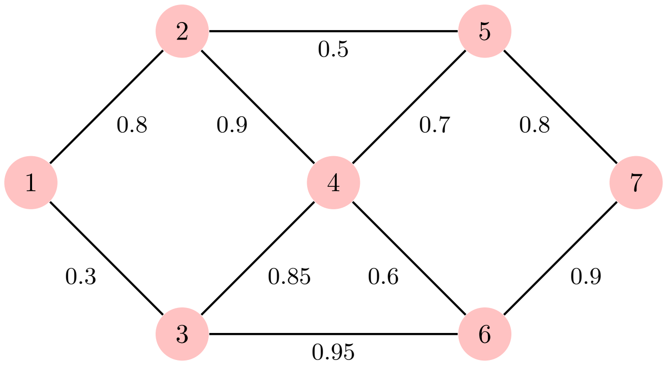


Figure 4.0.2. Road traffic network for Exercise 4

EXERCISE 21. Figure 4.0.2 shows the possible routes to move from city 1 to city 7, and the associated probabilities of not running into a traffic jam. Use Bi-directional Dijkstra's algorithm to find the most reliable route of this road traffic network. What is the probability of not getting stuck in a traffic jam on the most reliable route?

Dijkstra's algorithm for travel times

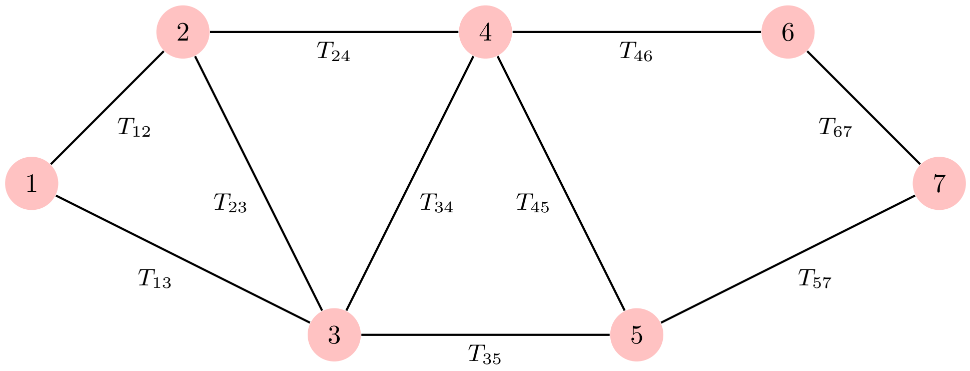


Figure 4.0.3. Road traffic model with stochastic travel times

EXERCISE 22. (1) Assume that the travel time distributions of the road traffic network in Figure 4.0.3 are known and are given in Table 4.0.1 below. Find the route between city 1 and 7 that has the lowest expected travel time.

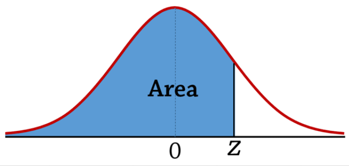
	T_{12}	T_{13}	T_{23}	T_{24}	T_{34}	T_{35}	T_{45}	T_{46}	T_{57}	T_{67}
μ	12	10	2	7	4	4	3	10	19	5
σ^2	1	9	1	9	4	1	1	16	1	4

Table 4.0.1. T_{ij} denotes the travel time between city i and j in minutes and is normally distributed with parameters μ and σ^2.

(2) Suppose you have a job interview in 40 minutes. Find the probability that you arrive on time if you take the route you found in part (1). What is this probability if you take the route $1 \to 3 \to 5 \to 7$? What do you observe?

Hint

Recall that the travel times are independent across the different roads. You can use that if $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$ are independent, it holds that $X + Y \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$.



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998

Figure 4.0.4. Values of distribution function of normal distribution

Chapter 5

Solutions to exercises

Graph theory

(1) Each possible edge in a graph $G = (V, E)$ corresponds to a pair of nodes. In total we have $|V|$ nodes in G, which means we can form in total $\binom{|V|}{2}$ possible pairs of nodes, where $\{i, j\}$ is considered the same as $\{j, i\}$.

(2) We will reformulate the question and then we use Exercise 1. In a graph G there are in total $\binom{|V|}{2}$ possible edges. In a specific graph some of these edges will be present and some not. Hence the total number of graphs with $|V|$ nodes is equivalent to the number of sequences of length $\binom{|V|}{2}$ where each element is either a 0 (edge is not present) or a 1 (edge is present). There are in total $2^{\binom{|V|}{2}}$ such sequences.

(3) $d(0) = 16, d(1) = 8, d(8) = 5, d(33) = 17$. For the eccentricities we compute Ecc$(0, \text{Pink}) = 3$, Ecc$(1, \text{Pink}) = 4$, Ecc$(16, \text{Pink}) = 4$, Ecc$(8, \text{Pink}) = 4$, Ecc$(15, \text{Green}) = 3$, and Ecc$(33, \text{Green}) = 2$. For the diameter and the radius of the two subnetworks we have Diameter(Pink) $= 4$, Diameter(Green) $= 3$, and Radius(Pink) $= 2$, Radius(Green) $= 2$.

Networks and communities

(5) There are six cliques of size 4, corresponding to the group stage of the tournament.

(6) Yes, starting from the ‘true’ split, if we move node 8 to the community of the administrator, we get a cutsize of 10, which is the minimal cutsize for such splits.

(7) No, consider four nodes connected by a line. The diameter is three while the radius is two. The diameter is always at most twice the radius.

(8) See Figure 5.0.1

(9) The edge between nodes $n/2$ and $n/2+1$ has bottleneck-ness $n^2/4$. In general, node k has bottleneck-ness $k \cdot (n-k)$, which is maximized for $k = n/2$.

(10, 11) Note that each clique consists of $\binom{s}{2}$ edges so that there are $k \cdot \binom{s}{2}$ edges inside the cliques and k edges between the cliques. We get

$$\begin{aligned}\text{SimpleModularity}(C) &= \frac{1}{k \cdot (1+\binom{s}{2})} \cdot k \cdot \left(\binom{s}{2} - \binom{s}{2} \cdot \frac{k \cdot (1+\binom{s}{2})}{\binom{k \cdot s}{2}}\right) \\ &= \frac{\binom{s}{2}}{1+\binom{s}{2}}\left(1 - \frac{k \cdot (1+\binom{s}{2})}{\binom{k\cdot s}{2}}\right) \\ &= \frac{\binom{s}{2}}{1+\binom{s}{2}} - \frac{k \cdot \binom{s}{2}}{\binom{k \cdot s}{2}} \\ &= \frac{\binom{s}{2}}{1+\binom{s}{2}} - \frac{s-1}{k \cdot s - 1}.\end{aligned}$$

The number of edges inside the communities increases by 1 (the edge between C_1 and C_2) while the number of intra-community pairs increases by $|C_1| \cdot |C_2| = s^2$. Hence, the *expected* number of edges inside the communities increases with

$$s^2 \cdot \frac{k \cdot (1+\binom{s}{2})}{\binom{k \cdot s}{2}} = s\frac{s+1}{k \cdot s - 1}.$$

Thus, simple modularity increases whenever

$$1 > s\frac{s+1}{k \cdot s - 1},$$

which can be rewritten to

$$k > s + 1 + \frac{1}{s}.$$

Hence, for a sufficiently large number of cliques, merging two adjacent cliques results in an increase in the simple modularity. This shows that maximizing simple modularity does not always result in natural communities.

(12) Expected degree $(s-1)p_{\text{in}} + (k-1)sp_{\text{out}}$. Dividing by $|N|-1$ gives the expected density.

(13) From the previous exercise, we have

$$\text{Density}(N) = \frac{(s-1)p_{\text{in}} + (k-1)sp_{\text{out}}}{|N|-1}.$$

The expected Simple Modularity is given by

$$k\binom{s}{2}(p_{\text{in}} - \text{Density}(N)).$$

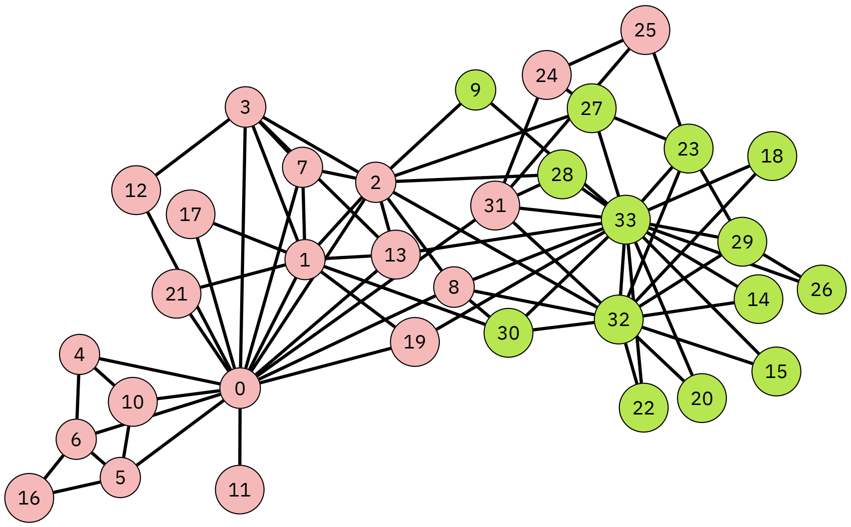


Figure 5.0.1. Answer to Exercise 4.

(14) $(1-p_{\text{in}})^{s-1}$ and $n(1-p_{\text{in}})^{s-1}$

(15) Infinity. Each node has a positive probability $(1-p_{\text{in}})^{s-1}$ of not being connected to any community members, leading to an infinite distance. Thus the expected value is at least $\infty \cdot (1-p_{\text{in}})^{s-1} = \infty$.

(16) $s^2 p_{\text{out}}$ and p_{out}.

(17) $(s-1)p_{\text{in}}^2 + (k-1)sp_{\text{out}}^2$ and $2(s-1)p_{\text{in}}p_{\text{out}} + (k-2)sp_{\text{out}}^2$.

(18) We can take the derivative of $s(1/2)^{s-1}$ w.r.t. s to see that it is decreasing for $s \geq 2$, so the smaller community is expected to have more nodes disconnected from the community.

Probability theory

(19) Since $\mathbb{P}(T \geq t) = 1 - \mathbb{P}(T < t)$, it is sufficient to show

$$\mathbb{P}(T_{13} < 20) > \mathbb{P}(T_{12} < 20).$$

We use that

$$\frac{T_{ij} - \mu_{ij}}{\sigma_{ij}} \sim \mathcal{N}(0,1).$$

Table 4.0.4 on page 52 then gives

$$\mathbb{P}(T_{13} < 20) = \mathbb{P}\left(\frac{T_{13} - 15}{2} < \frac{5}{2}\right) \approx 0.9938$$

and

$$\mathbb{P}(T_{12} < 20) = \mathbb{P}\left(\frac{T_{12} - 12}{4} < 2\right) \approx 0.9772,$$

from which the conclusion follows.

(20)

$$\begin{aligned}
\mathbb{E}[X+Y] &= \int_{\mathbb{R}} \left(\int_{\mathbb{R}} (x+y) f(x,y) \mathrm{d}y \right) \mathrm{d}x \\
&= \int_{\mathbb{R}} \left(\int_{\mathbb{R}} x f(x,y) \mathrm{d}y \right) \mathrm{d}x + \int_{\mathbb{R}} \left(\int_{\mathbb{R}} y f(x,y) \mathrm{d}y \right) \mathrm{d}x \\
&= \int_{\mathbb{R}} \left(\int_{\mathbb{R}} x f(x,y) \mathrm{d}y \right) \mathrm{d}x + \int_{\mathbb{R}} \left(\int_{\mathbb{R}} y f(x,y) \mathrm{d}x \right) \mathrm{d}y \\
&= \int_{\mathbb{R}} x \left(\int_{\mathbb{R}} f(x,y) \mathrm{d}y \right) \mathrm{d}x + \int_{\mathbb{R}} y \left(\int_{\mathbb{R}} f(x,y) \mathrm{d}x \right) \mathrm{d}y \\
&= \int_{\mathbb{R}} x f(x) \mathrm{d}x + \int_{\mathbb{R}} y f(y) \mathrm{d}y \\
&= \mathbb{E}[X] + \mathbb{E}[Y],
\end{aligned}$$

where used in the third line that we can change the order of integration.

Road Traffic Analysis

Dijkstra's algorithm for reliable routes

(21) In order to use Bi-directional Dijkstra's algorithm, we first need to formulate the problem as a shortest route problem. This can be done by replacing the probabilities that are assigned to the edges by the negative of their logarithm. Therefore, the most reliable route of the network in Figure 4.0.2 can be determined by finding the shortest route of the network in Figure 5.0.2.

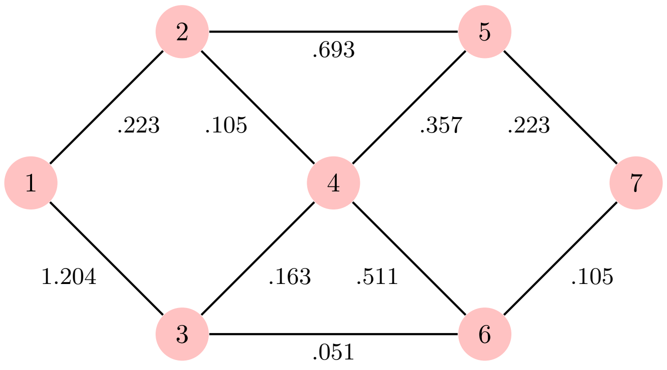


Figure 5.0.2. Most-reliable-route representation as a shortest-route model

Applying Bi-directional Dijkstra's algorithm gives the route

$$1 \to 2 \to 4 \to 3 \to 6 \to 7,$$

from which it follows that

$$\begin{aligned} -\log p_{17} &= -\log p_{12} - \log p_{24} - \log p_{43} - \log p_{36} - \log p_{67} \\ &= 0.223 + 0.105 + 0.163 + 0.051 + 0.105 \\ &= 0.647. \end{aligned}$$

We conclude

$$p_{17} = e^{-0.647} \approx 0.524.$$

Using this route, there is a probability of 52.4% of not getting stuck in a traffic jam.

Dijkstra's algorithm travel times

(22) We learned from Exercise 20 that we can simply apply Dijkstra's algorithm to find the route that minimises the sum of the means. This gives the path

$$1 \to 3 \to 4 \to 6 \to 7,$$

which has an expected travel time of 29 minutes.

Using the hint, we find that

$$T_{17} = T_{13} + T_{34} + T_{46} + T_{67} \sim \mathcal{N}(29, 33).$$

Therefore,

$$\mathbb{P}(T_{17} \leq 40) = \mathbb{P}\left(\frac{T_{17} - 29}{\sqrt{33}} \leq \frac{40 - 29}{\sqrt{33}}\right) \approx 0.9722.$$

The alternative route is distributed as

$$\tilde{T}_{17} = T_{13} + T_{35} + T_{57} \sim \mathcal{N}(33, 11).$$

Therefore, the probability of arriving on time is

$$\mathbb{P}(\tilde{T}_{17} \leq 40) = \mathbb{P}\left(\frac{\tilde{T}_{17} - 33}{\sqrt{11}} \leq \frac{40 - 33}{\sqrt{11}}\right) = 0.9826.$$

We observe that the route with the lowest expected travel time is not the route that gives us the highest probability of arriving on time for our job interview.