The Document

João Morais

October 19, 2019

Contents

1 Mobile Communications: Cellular & Radio Access Networks

Currently, 70 to 2600 MHz is the used band mainly due to propagation characteristics - a notion that we'll reinforce is that the higher the frequency, the higher the attenuation - and due to the size of the antennas we can achieve. They are human scale, from 3m to 10cm.

- In the VHF/UHF band, propagation is characterised by being:
  - almost independent of polarisation and soil electromagnetic properties;
  - essentially done via direct and reflected rays;
  - influenced by the presence of obstacles;
  - almost insensitive to refraction by atmosphere;
  - basically limited by the radio-horizon;
  - basically independent of rain, gases and others.

1.1 3GPP Specifications

These specifications are completely open! Are standards, everyone needs to know what is the standard. You just have to know where to find it.

First, find the Technical Specification Groups (TSGs): 3GPP website Specifications Groups .

What is useful for a 5G thesis is probably the RAN part. The physical layer group is TSG-RAN Working Group 1 (WG1): TSG-RAN WG1 .

In their list of specifications, if one looks closely, the following can be found:

TS 38.201	NR; Physical layer; General description
TS 38.202	NR; Services provided by the physical layer
TS 38.211	NR; Physical channels and modulation
TS 38.212	NR; Multiplexing and channel coding
TS 38.213	NR; Physical layer procedures for control
TS 38.214	NR; Physical layer procedures for data
TS 38.215	NR; Physical layer measurements
TR 38.802	Study on new radio access technology Physical layer aspects
TR 38.812	Study on Non-Orthogonal Multiple Access (NOMA) for NR

Clicking on one of them gets us to a FTP (File Transfer Protocol) page it's just needed to click in the "show all versions of this specification" and choosing the last one. Then is done! You have the study/standard about that matter!

The rest comes with experience ;)

Some free experience:

- How releases work: 3GPP Releases
- How specification numbering works: 3GPP Spec Numbering (Look that Series 38 is for Radio Technology beyond LTE, which is basically all 5G (image above))
- FAQs: 3GPP FAQs
-

2 Telecommunication Networks - Overview

From the courses Internet Networks and Services (RSI in portuguese) and Telecommunication Networks (RTel in pt) I've had an insight about how the whole network is multiplexed into optic fibers and many other interesting topics such as the triple play services and a bunch of protocols that are used in today's world to make everything communicate with everything. Therefore, I propose to write a sum up of the slides and bibliography of RSI and RTel in this section. I'll mainly give importance to RTel since it is what I'm studying at the moment, but I hope to go through the slides of RSI as well.

1. Introduction
   (2 lessons)
2. Fundamentals of networks
   (7 lessons)
3. Ethernet and data centre networks
   (5 lessons)
4. SDH transport networks
   (4 lessons)
5. Optical transport networks
   (4 lessons)
6. Access networks
   (3 lessons)

1. The Internet
2. Quality of Service on the Internet
3. IP Network Models
4. Next Generation Networks
5. The Telephony Network
6. Technologies for data transport
7. MPLS - Multi-Protocol Label Switching

2.1 Introduction

Definition *Telecommunications* : is the transmission of information at a distance through the use of electromagnetic signals.

Definition *Telecom. Network* : collection of nodes and links with the purpose of interchanging these signals in order to have an information flow.

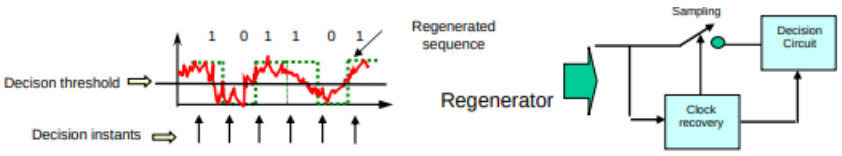
These Telecommunication Networks can be public, owned by Telecom. / Network Operators that use that network to provide services to the general public, or can be private, used by a company to connect infrastructures. Many of these private networks also rely on leased links by public networks.

There are mainly 3 layers in a network: the backbone or core, the metropolitan and the access layer. As expected, the access layer collects the traffic, connections to homes, offices, everywhere the internet is required. The metropolitan area connects different parts of the city that use that network, typically with a ring (made out of optic fiber). The core is the most extensive layer, with a mesh of nodes and very extensive links that connect cities of the whole world. Hundreds or thousands of km's is not atypical.

What makes possible to communicate with everyone connected to the internet is that the networks of both operators are also connected.

As a public service, the public networks must provide fidelity (transmit the information without loss of changes) and reliability (less than 3 minutes down per year).

Nowadays, most of transmission is digital. A series of pulses is transmitted through a channel with attenuation, dispersion, interference from other signals and noise. Therefore what reaches the other side is considerably different and has to be estimated what the original input was.



As having a dedicated physical infrastructure for each service would be far to expensive and messy, the big majority of services share the same channel, the optic fiber. It is easily shared because of the available bandwidth in it. Thus, the signals are multiplexed at the entrance, using different wavelengths (**Wavelength-Division**

Multiplexing (WDM)) and de-multiplexed at the other end, to follow each one to their device that is requiring the service.

Note that WDM is exactly like **Frequency Division Multiplexing (FDM)** but in the optical domain. Technically they are exactly the same as changing the wavelength is nothing more than changing the transmit frequency.

A single mode optical fiber can reach throughputs of 10 Terabits per second.

Remember that there are many ways of scheduling frames in a multiplexer. With time slots or doing it statistically are two ways. Also, there are 2 types of switching: packet switching and circuit switching. Circuit is when a channel is constantly reserved for a certain application even if it is not being used. Packet switching allows a much better share of the resources. Packet switching principle is based on sending packets whenever there's a packet to transmit and use all the resources to do so as fast as possible. Therefore, the "speed" of the internet depends a lot on the amount of people that are accessing it.

Regarding the physical infrastructures for the transmission of data, those go from satellites, well the open space in general as microwave links are also a thing, twisted pairs, optical fibers and even a few more that are less common.

Finally, a look at the tendencies is pertinent. The traffic is increasing constantly, at a rate of 30% a year, therefore the network must be upgraded as the time passes as well, or else it won't be able to handle the traffic of the future. Not only are the links being upgraded since now we have fiber to the home, terminating really in our router, but each node must be upgraded as well to cope with the traffic resulting in new switches, larger datacentres, ect... However, all of this must be standardised to guarantee compatibility between countries, operators, manufacturers and users and to ensure minimum quality of service for all users. This standardisation is done by the International Telecommunication Union (ITU) that has two main sectors of interest: the -T sector regarding telecommunications in general and the -R sector for radiocommunications that is more focused in point-to-point, mobile, satellite links, ect... Additionally, ETSI, ISO, OSI, ANSI, IEEE are some of the main organisations that standardise technologies. IEEE is the best :)

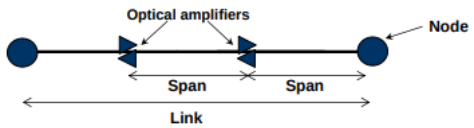
2.2 Networks Fundamentals

A network is composed of nodes and links and can be represented by graphs. However, a clear distinction between networks and graphs has been made in class: a network is a graph with a few more numbers that represent various network parameters. These parameters will be talked later, but can be delay of a link, distance, ...

The physical topology concerns the physical connections that are in place while the logical topology for a certain case concerns the actual flow of information. Even though every computer is connected in a network, maybe the information always flows from one to the others and the graph that represents that has much less links.

A link can be unidirectional or bidirectional. If the link is unidirectional, sometimes is referred to as an arc. $e_1 = (v_1, v_2)$ is the representation of a link, and the order of the nodes matter if it's an arc.

In optical fiber networks, or other networks that require amplifiers, the space between amplifiers (distance the signal has to go attenuating) is called a span.



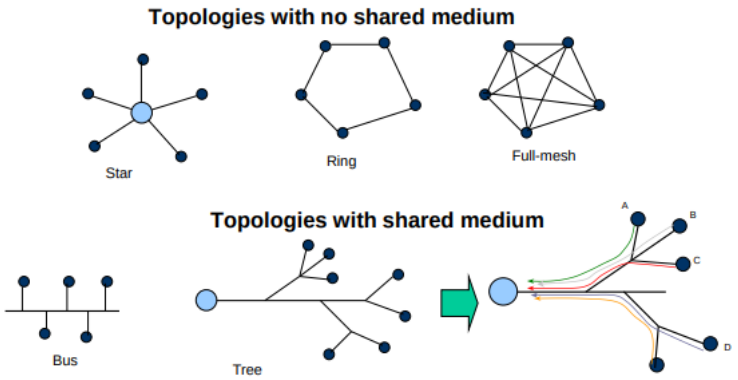
In a graph there's N number of Vertices(v_i), and L number of Edges(e_j). And the degree of the vertex is the number of edges it has. It's called the order of the graph, it's number of vertices, and the size of the graph it's number of edges.

Directed graphs only have unidirectional links, while undirected graphs only have bidirectional links.

The reason to make the distinction between directed and undirected graphs (unidirectional and bidirectional edges): in case of optical fiber, which is what connects most of long distance networking, is required to use more than one fiber. Because and optical emitter can't receive as well (at least in the same fiber). Also, if amplifiers are required, note that they are directed as well.

A path can be represented by a set of links, starting at some node. Source and sink are the names for the first and last vertex of that path.

2.2.1 Network Topologies



Bus, Ring and Star are the main physical topologies. Tree as well.

A tree is simply a graph with no cycles.

2.2.2 Network representative Matrices

A graph can be represented with an **Adjacency matrix (A)** , with $a_{ij} = 1$ if there's a direction from the vertices i to j.

The average node degree is given by the average of each node's degree which won't be more than the sum of all links, times 2 divided by the number of nodes. Times 2 because each link contributes for the degree twice, once at each end.

$$\delta_i = \sum_{j=1}^{N} a_{ij}$$

The average node degree is given by

$$<\delta> = \frac{1}{N}\sum_{i=1}^{N}\delta_i = \frac{2L}{N}$$

The Network diameter (D_R) is the maximum number of links between nodes through the shortest path between them. D_R is the longest of the shortest paths between every node.

Every link can have an associated cost (a function of distance, delay, reliability, actual cost, or other parameters) and a capacity (u_e denotes the capacity of node e).

The link capacity can be measured in any traffic unit that is appropriate for the problem. In packet networks: bit/s; in SDH networks: STM-N; in OTN networks: OTU-k; in WDM networks: number of wavelengths

Despite similar to an Adjacency Matrix, the **Demand matrix (D)** is slightly different. $d_{ij} = 1$ if the traffic flows from the vertex i to the vertex j.

Note that the diagonal of this matrix should be empty, or else it would mean that a certain node would receive information from himself, which makes no sense.

The mean number of demands is the number of demands divided by the number of nodes.

$$<d> = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}$$

The number of unidirectional demands (edges) in a case of full mesh logical topology is $D_1 = N(N-1)$. N nodes x the other N-1 nodes. Note that One other way of seeing it is that the D matrix is $N \times N$ but we need to take N away due to the empty diagonal. $D_2 = \frac{D1}{2}$ is the number of bidirectional demands, which is when only the top triangle of the D matrix is considered. This is usual because with bidirectional links the D matrix will always be symmetric.

Another interesting matrix is the **Traffic matrix (T)** and it's used to denote traffic intensities. It only has entries different from 0 in the exact same places the Demand matrix has. It's used for static traffic designs.

In transport networks the traffic units is the type of client signals: Ex: E3 (34 Mb/s), STM-1 (155.51 Mb/s), GbE (1 Gb/s), 10 GbE (10 Gb/s) ,etc. This traffic units must be converted to traffic units appropriate to be used in network design. These traffic units must be VC-n in SDH networks and in OTN networks ODU-k signals.

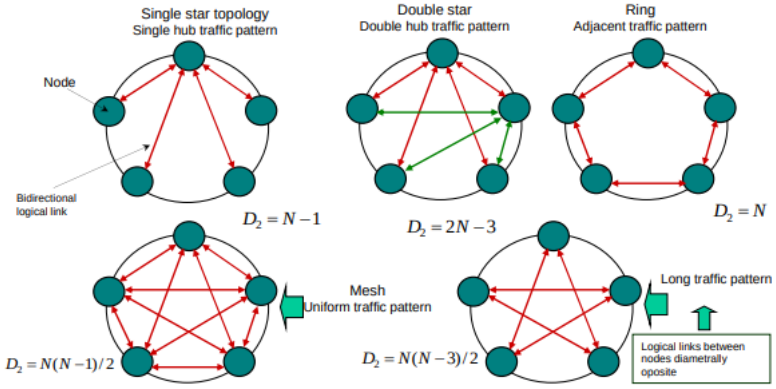
Considering now a Dynamic Traffic case, a few concepts arise:

- Average intensity of data flow between two nodes;
- Traffic bursts are time intervals where the flux of data is considerably higher than the average rate;
- Peak rate is the maximum instantaneous intensity.

Aggregation level or multiplexing level reduces the traffic *burstiness* because there less often flows get fully used and is more likely that the information can flow at a constant pace instead of in bursts.

Note one important distinction between Logical and Physical Topologies that we haven't done yet is that there can be many logical topologies on one physical topology. This can happen in the following way:

- **In a ring network there are diferent ways how the traffic can flow in a network leading to different network topologies (single star, double star, ring, mesh, etc.). *N*: number of nodes; D_2: number of bidirectional (two way) logical links.**



Routing is how a packet travels in a network. Therefore, routing ends up being the map between logical and physical topologies.

In optical networks the path between two nodes is usually called "lightpath".

We can also define a **Cost Matrix (C)** where each element c_{ij} represents the costs between nodes i and j.

The path can be performed manually (static routing - Demand matrix is time invariable) or dynamically, through routing algorithms (dynamic routing - Traffic matrix is time dependent, with constant arrival and termination of new demands).

Additionally, if the a given traffic demand(connection) is able to use more than one route, it is called a multipath routing process, else it is a mono-path process. Because there are usually many paths connecting two nodes, some metrics are taken into account when choosing which to follow:

- 1) Minimizing the network cost;
- 2) Minimizing the traffic in the most loaded link;
- 3) Minimizing the number of hops (number of links in the path);
- 4) Minimizing the path distance;
- 5) Maximizing the protection capacity, etc.

Most of the routing strategies incorporate some sort of shortest path algorithm to determine which path minimizes a particular metric, which can be for example 1), or 3), or 4). Note that the same algorithm used to 4) can be applied to 3) making all link distances identical.

From the physical topology, described by a graph G(V,E) and the traffic matrix T, describing all the traffic demands to be routed, one can perform shortest path algorithms such as Dijkstra's algorithm.

Order the demands according to a certain sorting strategy:

- Shortest-first: The demands with the lowest number of nodes in its path come first in the list;
- Longest- first: The demands with the highest number of nodes in its path come first in the list;
- Largest-first: The demands with the highest number of traffic units come first in the list;
- Random ordering: the demands are not known initially.

Route de demands according to the orderings. To break a tie choose the path that minimizes the load in the most loaded link.

Dijkstra's Algorithm

Consider a generic node *i* in a network with *N* nodes from where one wants to determine the shortest path to all the other nodes in the network. Lets l_{ij} be length (cost) of the link between node *i* and node *j* and d_{ij} the length of the shortest path between node *i* and *j*.

Algorithm:

1) Start with the source node i in the permanent list of nodes, i.e. $S=\{i\}$; all other nodes are put in the tentative list labeled S'. Set $d_{ii}=0$ and $d_{ij}=\infty \quad \forall j \neq i$.
2) For all the adjacent nodes to i set $d_{ij} \leftarrow l_{ij} \quad \forall j$ adjacent to i .
3) Identify the adjacent node j (not in the current list S) with the minimum value of d_{ij} (permanent node), add it to the list S ($S=S\cup\{j\}$) and remove it from ($S'=S'\setminus\{j\}$). If S' is empty stop.
4) Consider the list of neighboring nodes of the intermediate node j (but not consider nodes already in S) to check for improvement in the minimum distance path by setting $d_{ik} \leftarrow \min(d_{ik}, d_{ij}+l_{ij})$. Go to Step 3.

Now is pertinent to introduce yet another matrix, the **Hop Matrix (H)** where each element h_{ij} denotes the minimum number of hops from node i to node j.

The average number of hops per demand is nothing more than the sum of the hops of all demands divided by the number of demands. The number of demands was set as the unidirectional links between two nodes. Therefore, if 2 nodes share information between themselves (don't need to have a physical link, a logical one is enough) then there's a demand.

Therefore, the average number of hops can be computed:

$$< h >= \frac{1}{D}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} h_{ij}$$

Note that coherence is key here. If the amount of demands are the bidirectional demands, then the number of hops considered should only be the top half of the hop matrix. **If links are bidirectional, then there will always be a symmetry in these matrices** and we should compute the average with amount that mean the same thing.

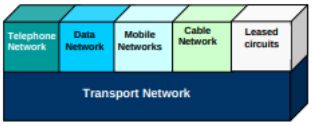
As means of simplifying this calculation, because hops and demands can be dynamic, a way of having a notion on the order of magnitude and get a fairly good approximation, when the number of nodes N is $4 \leq N \leq 100$ and the average node degree $< \delta >$ is $2.5 \leq < \delta > \leq 5$, is by computing the semi-empirical relation:

$$< h > \cong 1.12\sqrt{\frac{N}{< \delta >}}$$

2.2.3 Layers

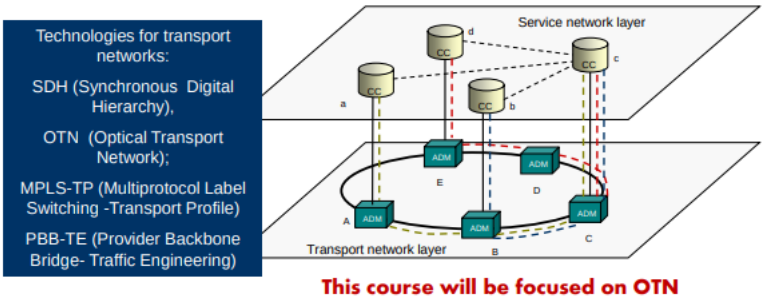
Typically, there's a layered structure in the network. The layer above acts as a client of the layer beneath and each layer appears a black box that supplies a service to the layer above.

The service layer is the one closer to us.



Add/Drop Multiplexing(ADM) are multiplexers controlled by **Control Centres (CC)** that decide what to add and what to drop form the fibre. Note that these don't manage the network, they just use it.

Nowadays, apart from local networks, everything is connected with fibre. Therefore, is pertinent to mention 4 key technologies for transport networks:



This course will be focused on OTN

SDH is also used in Hertzian links, MPLS is Multiprotocol Label Switching and is very useful for routing through different technologies and to choose more carefully the paths. As said above, OTN will be our transport technology.

OTN has become the new standard for a while now and has a few differences compared with SDH. The most important of which is distinction between fixed frame size and fixed rates. SDH has a fixed rate while OTN can increase it's rate to match the client's and this is very important for scalability and being more future proof

A very important distinction between the transport and the service layers is that the representation in the service layer has nodes connected with **logical topologies** while the transport layer has nodes connected with **physical topologies** .

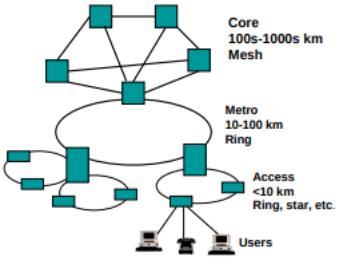
OTN	SONET/SDH
Asynchronous mapping of payloads	Synchronous mapping of payloads
Timing distribution NOT required	Requires right timing distribution across networks
Designed to operate on multiple wavelengths (DWDM)	Designed to operate on multiple wavelengths
Scales to 100Gb/s (and beyond)	Scales to a maximum of 40Gb/s
Performs single-stage multiplexing	Performs multi-stage multiplexing
Uses a fixed frame size and increases frame rate to match client rates	Uses a fixed frame rate for a given line rate and increases frame size (or uses concatenation of multiple frames) as client size increases
FEC sized for error correction to correct 16 blocks per frame	Not applicable (no standardized FEC)

The network management systems sends configurations through the **Data Communication Channel (DCC)** . Moreover, not all parts of the network are the same!

In the core and metro networks the physical topology is normally determined by the transport layer.

The access network uses a large variety of technologies and topologies and is responsible for the largest fraction of the operator´s investment on a telecom network.

Transmission technologies in an access network: twisted pair (copper), coaxial cable, optical fibre, wireless solutions (FWA).



In a more abstracted way, one can identify three planes:

- **Data Plane : is concerned with the transmission of the data between the users (forwards the traffic). Assures the physical support.**
  Also called forward or switching plane
- **Control Plane: is responsible for exchanging control information (signalling) between networks elements (nodes), which is used to set up, maintain, and tear-down connections. Examples of control planes: Signalling system nº 7, GMPLS (*Generalized multiprotocol label switching*), SDN (Software-Defined Networks) etc.**
- **Management Plane : Consists of several functions like detecting (alarms) and repairing failures (fault management), network element configuration (configuration management), performance monitoring to ensure to clients quality-of-service (performance management), allowing the system administrator to control personal access (security management).**

As you already know, there are circuit switched or packet switched networks.

Circuit Switched require circuit establishment and tear-down at the beginning and at the end, respectively. However, a distinction is made between physically *switchable* circuits and semi-permanent circuits. The first ones, a physical circuit is easily switched to connect one end to the other, while the second type regards circuits that are more static, that are much more difficult to switch. They are basically assembled.

Circuits can be switched or semi-permanent. The first ones are established by the control plane (by signalling) as the case of phone circuits. The second ones are established by the management plane as it is the case of the electrical paths in SDH networks, or optical channels in OTN networks.

3 Artificial Intelligence / Machine Learning

3.1 Artificial Intelligence

There are some goal states and one initial state. The objective is to find the goal state that is closer (with the shortest path to the root/ with the least search cost). Is given as the solution the steps necessary to perform that path.

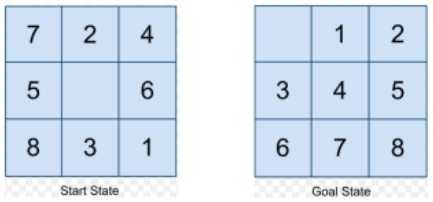
To keep the formulation as general as possible, abstractions are required, keeping in mind that they will need a correspondence when applied to a real problem.

Example of the application to a problem: a vacuum cleaner that needs to clean every square. In this case, only 2 squares are presented, no localization sensors are present, only "rubbish" sensors that tell if the current square is dirty or not.

Therefore:

- States : $< r, d_1, d_2 >$ where r is the robot position, d_1 and d_2 are binary, representing the existence of dirt in each of the rooms.
- Operators/Actions : L (go left), R (go right) or S (suck dirt)
- Goal Test : d_1 = False and d_2 = False. $((d_1 \text{ nor } d_2) == 1)$
- Initial State : $< r, d_1, d_2 >=< 1, T, T >$ is at square 1, and squares 1 and 2 are dirty
- Step Cost : the description of each action cost. In this case, 1 for each one.

8-puzzle



States: specify in a 3x3 matrix the location of the 8 numbered tiles plus the blank one

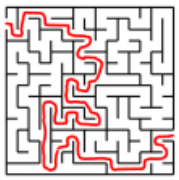
Initial state: any puzzle configuration

Operators: move "blank" to the left, right, up and down

Goal test: checks whether the state matches the goal configuration

Path cost: each step costs 1 (so path cost = number of steps)

Mazes



States: maze configuration plus current location

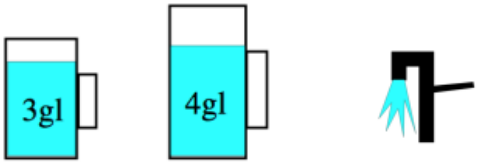
Initial state: any maze location

Operators: move to an adjacent and linked position

Goal test: current location = maze exit

Path cost: number of steps

Water jugs



States: amount of water in each jug (e.g., <1,4>)

Initial state: <0,0>

Operators: fill J4; fill J3;
empty J4; empty J3;
pour water from J4 into J3 until J3 is full or J4 is empty;
pour water from J3 into J4 until J4 is full or J3 is empty;

Goal test: <0,2>

Path cost: amount of water used

Hanoi towers

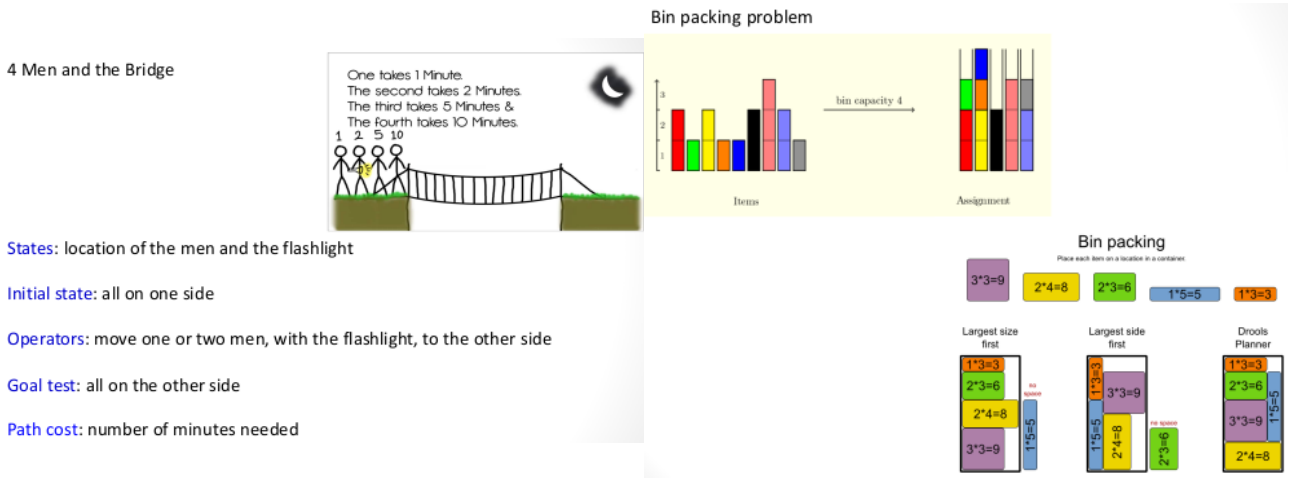
States: location of the disks in the three poles

Initial state: all disks correctly arrange in the left pole

Operators: move a free disk to another pole, which is empty or all disks in it are bigger

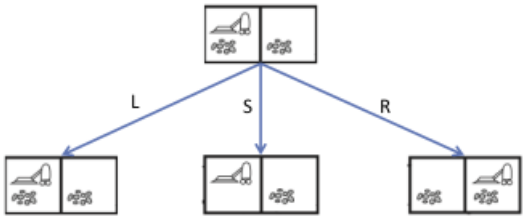
Goal test: all disks correctly arrange in the right pole

Path cost: number of disk moves

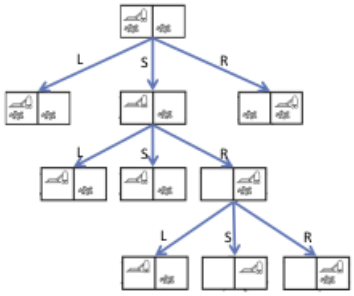


As you can see, every problem is very well defined in terms of the start, the end, the possible moves and what makes a solution better than other. Therefore, the conditions to put the computer thinking how to get from the possible steps to the best solution are assembled. One way the computer should be able to get to the solution is by exploring every move combination and check the state it ended up with.

Given the initial state, use the operators to generate successor states.



Then a choice of which is more "profitable" or more likely to lead to a goal state needs to be made and this cycle continues until the arrival at a goal state.



In this case, the solution would be $\{S, R, S\}$.

However, before diving directly in algorithms, is important to note the characteristics of the Environment and of the Agent. Based on these we'll be able to perform much better choosing the search algorithm that is best for us.

3.1.1 Environment

Observability: how much can the agent know about the environment.

An environment is fully observable if the agent can see everything. Or partial observable: - part of the state is occulted and you simply can't

Deterministic: the effect of the actions are predictable If I move one queen from one place to the other, I can predict the effects, what is attacking what, etc... Instead of Deterministic, it can be stochastic, where the outcomes are functions of probability functions. Therefore, you can't be 100% sure of the outcome. Therefore, you can use probability to make choices...

Neither the environment nor the agent performance change while the agent is deliberating. –¿ Static.

In a dynamic environment, the agent performance can change with time.

Semi-dynamic means that the world is static, but the performance is changing. ex: turn game where the game doesn't change while you are thinking but the more you take, the less point you get.

Continuous or discrete

Sequential or episodic. Episodic means that one episode doesn't influence the next one. It's very related with causality. In episodic environments, there's no influence in consequent problems.

E.g. one game of chess is sequential, but different games are episodic.

The outcomes for all actions are given. –¿ Known environment.

In a case where the less time you spend thinking before answering,

Knowing these is very important because it allows us to best choose the shelf of methods from which we take our algorithms from. Some are best from some things and some can only be used in certain situations as well.

Internally to the agent, you have a way of representing of the world. (Internal representation)

Example when you don't have an internal representation: random vacuum cleaning robots don't know anything about the environment, they just rotate randomly when they see an object.

- Fully observable vs. partially observable
  (partially observable because of noise, inaccurate or faulty sensors, or hidden parts)
- Single agent vs. multiagent (cooperative vs. competitive)
  (in multiagent environments, communication may be a key issue)
- Deterministic vs. stochastic
  (an environment is uncertain if it is not fully observable or not deterministic)
  (nondeterministic environment is when actions have different possible outcomes but no probabilities associated)
- Episodic vs. sequential
- Static vs. dynamic
  (if the environment doesn't change with the time but agent's performance does, it is **semidynamic**)
- Discrete vs. continuous (applies to states, time, percepts, and actions)
- Know vs. unknown
  (depends on the agent's knowledge about how the world evolves – the "laws of physics" of the environment)

	observable	deterministic	episodic	static	discrete
Chess	Y	Y	N	Y	Y
Poker	N	N	N	Y	Y
Taxi	N	N	N	N	N
Image analysis	Y	Y	Y	Semi	N

3.1.2 Agent

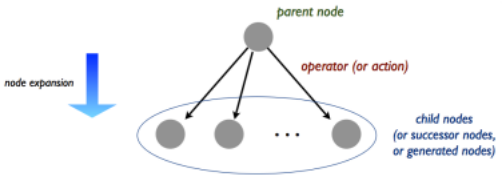
Model-based: typically, when you have partial observability.

Goal-based: when you aim for a goal. You have states and actions and can search for the goal.

Utility-based: are very similar to goal-based but are a bit more, not all goals are the same. There a preference between goals. Utility Theory handles how to translate preferences to numbers.

3.1.3 Search Problems

In essence, is necessary some **Search Terminology** to refer to certain things:



Successor function: given a node, returns the set of child nodes

Open list (or frontier or fringe): set of nodes not yet expanded

Closed (explored) list: set of nodes already expanded

Leaf node: a node without successors

State space can be:

- Tree-based – no repeated nodes in search
- Graph-based – directed cycle graph

General algorithm shape, starting with $open\_list = \{initial_node\}$, iterate over:

1. Select a node from the open_list;
2. Check if it's a goal node (in case it satisfies the goal test). If yes, return solution by backing up to the root;
3. If not, remote it from the open_list, expand it with the successor function and insert the nodes that come from there to the open_list.

Different selection criteria leads to a variety of search methods.

If it's tree based, then no nodes will be repeated and it's not necessary to have a list of the already visited nodes. In a graph however, is needed to have a list of these, or else a cycle is possible.

To evaluate the algorithm, many parameters may be enumerated:

Completeness: guarantee that a solution is found if there is one

Optimality: the solution found minimizes path cost over all possible solutions

Time complexity: how long does it take to find a solution (usually measured in terms of the number of nodes generated)

Space complexity: how much memory is needed to find a solution (usually measured in terms of the maximum number of nodes stored in memory)

Branching factor (b): maximum number of successors of any node

Depth (d) of the shallowest goal node

m = maximum length of any path in state space (may be infinite)

g(n): the cost of going from the root node to node n (path cost function)

Mentioning types of search strategies:

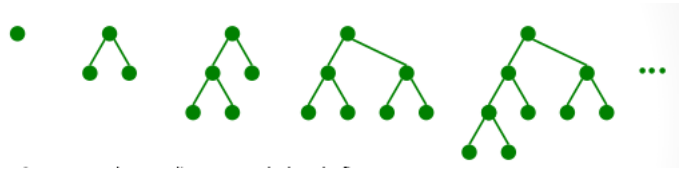
Uninformed (or blind) search: does not use additional (domain-dependent) information about states beyond that provided in the problem definition

Informed (or heuristic) search: uses problem-specific knowledge about the domain to "guide" the search towards more promising paths

3.1.4 Uniformed Search Strategies

A list of **Uniformed search strategies** we'll have a deeper look to:

- **Breadth-first Search** - Select earliest expanded node first - uses a FIFO queue (First In, First out). This leads to opening every node at the safe depth first before moving the deeper nodes.



Because order of depth is followed and every node checked, this search strategy is Complete and Optimal (if the path cost increases - or at least doesn't decreases - with depth). Being **d** the depth of the solution and **b** the branching factor(max number of successors of a node.) then in the worst case, the total number of nodes generated is: $1 + b + b^2 + b^3 + \cdots + b^d$

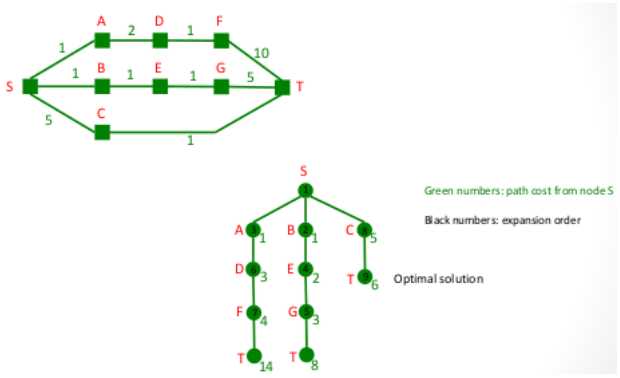
Time Complexity - O(b^d) Space Complexity - O(sum no of nodes) = O(b^d)

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

Note that it makes a difference if you test the node before or after expanding it. If it's tested before, there's no need of expanding the node. If the test is made only after the expansion, the complexities grows to O(b^{d+1}).

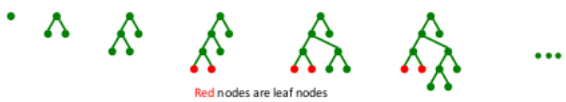
- **Uniform-cost Search** - expands the node that has the smallest cost from the root to it.



Note that each action generates one possible state from the state where the robot was previously.

This search strategy is only complete and optimal if the step costs are strictly positive. Else, it can give several steps that cost nothing to places far away from the solution.

- **Depth-first Search** - Exactly as the name suggests, goes until the deepest node first, and only then looks at the other nodes at the first level. Therefore, it can be very inefficient on a large or infinite tree. Open the last node added to the list (LIFO- Last In, First Out).



- **Backtrack Search** - a variation of depth-first, but expands one node at a time, only stores in memory that only node and the expansion is made by modifying the node, while backtrack is nothing more than undoing the modification..

  It's not complete nor it is optimal... but saves a lot of memory.

- **Depth-limited search** - another variation of depth-first where limiting the search tree to a depth L, is possible to contain the inefficiency. It's complete if the depth of the solution if smaller than L but still not optimal.

  Time complexity: O(b^L) Space complexity: O($b \times L$)

- **Iterative deepening depth-first search** - A variation of the previous one. In this one, the idea will be to run depth-limited search for an increasing L. Run for L=1, L=2, ...

This way, it is complete and it's optimal (if the path cost is a non-decreasing function of depth).

Time complexity: $O(b^d)$ Space complexity: $O(b \times d)$

Node expansion:

Breadth-first

$$1+b+b^2+b^3+\cdots+b^{d-1}+b^d$$

Iterative deepening

$$(d+1)1+(d)b+(d-1)b^2+(d-2)b^3+\cdots+(2)b^{d-1}+(1)b^d$$

Example: b = 10 and d = 5

breadth-first = 111 111

iterative deepening = 123 456 (+ 11%)

- **Bidirectional Search** - Search both from initial node and from the gold node. Note however that can only be used when a goal node is known and when the parent nodes can be computed given its child (through the sets of available actions). It's complete (if breath-first in both directions) and Optimal, if the step costs are equal.

  Time and space complexity: $O(b^{d/2})$

Example: $b = 10$ and $d = 5$

$b^d = 1\,111\,111$

$2\,b^{d/2} = 2\,222$

A summary of the above analysis:

Criterion	Breadth-first	Uniform-cost	Depth-first	Depth-limited	Iterative deepening	bidirectional
Complete?	✓	✓[1]	×	×	✓	✓[3]
Optimal?	✓[2]	✓	×	×	✓[2]	✓[2,3]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(b^m)$	$O(b^L)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(b.m)$	$O(b.L)$	$O(b.d)$	$O(b^{d/2})$

[1] for strictly positive step costs
[2] for path costs a non-decreasing function of depth
[3] for breadth-first in both directions

A **General Search Algorithm** can be formulated in the following way:

function General-search (***problem***, ***strategy***) **returns** a solution or failure
insert the root node into the open list
(the root node contains the initial state of problem)
loop do
if there are no candidate nodes for expansion **then return** failure
choose a node for expansion according to strategy (using strategy function)
if the node contains a goal state (using goal checking function) **then**
return the corresponding solution
else
for each operator in the list of operators (or successor function)
create a child node (for the new child state)
update child node path cost (using g-function)
add the resulting node to the open list [*unless… see graph search versions*]
end

domain (problem) independent

problem argument should include at least:

- initial state (using a specific state representation)
- successor function: new state = succ (current state, operator)
- path cost function (g-function)

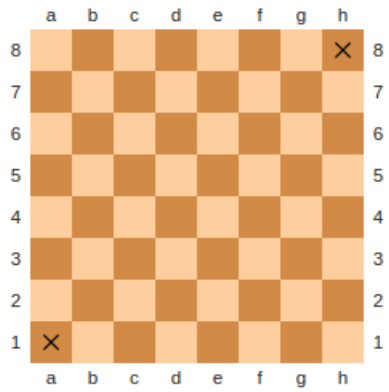
domain (problem) dependent

strategy argument (e.g., FIFO, LIFO, priority queue (by path cost), etc.)

algorithm selection

Finally, there are problem dependent and problem independent details. The search strategies are problem independent, but how we define the actions, states, ect… is problem dependent. And the way we see and think about the problem can be highly related to the way we represent it.

An incredibly well formulated example is the Mutilated Chess Board problem. If we take just the squares of the corners of a chess board, can we still fill the whole board with dominos (each domino takes 2 squares). It becomes slightly harder! If you keep removing squares it gets increasingly harder to figure out by head.



However, if you represent the chess board as the remaining black and white pieces and notice that a domino piece must always cover one black and one white squares, then by taking those two corners then 30 black and 32 white squares will be remaining making it impossible to fill with dominos. As a matter of fact, accordingly with Gomory's Theorem is also possible to say that if 2 square of opposite colours are removed then is always possible to fill the board with dominos! More in: Mutilated chessboard problem

The bottom line is that the representation (problem dependent details) matters.

With this is mind, consider the following example:

Initial state with: bedroom(3), living room(2), kitchen(3), hall(2), truck(0)

a) State: $< pos, n_{bedroom}, n_{living}, n_{kitchen}, n_{hall}, n_{truck} >$

b) Initial State: $< truck, 3, 2, 3, 2, 0 >$

c) m for move, p for push $< m_N, m_S, m_E, m_W, p_N, p_W, p_E, p_W >$. However, is only possible to push in directions where there are rooms (as for walking) and when there are boxes in the current room we are located in.

d) A goal condition could be $n_{truck} = 10$. Another could be the sum of all rooms to be 0. But the first one is more elegant and also seems more general... We don't want to throw boxes out of the window.

Sometimes, just finding the solution is enough. Some times, there's a best solution.

The heuristic (the only difference between informed and uninformed search) is a function of a state that gives us the appreciation we have for that state - how good that state is. If we want the best solution, we must have the heuristic function is key. If you just want a solution, that function can be much simpler or even nonexistent.

3.1.5 Informed Search Strategies

A problem-solving agent is a goal-based agent that acts on the environment, leading him to go through a series of states in order to achieve the desired goal

3.2 Supervised Learning

Extrema Conditions and Hessian Matrix

Videos: 87 - Warm up to the second partial derivative test to... 89 - Second partial derivative test intuition

This sub (...) sub section aims to explain why the conditions imposed on the Hessian matrix - the second derivative matrix - correspond to imposing in 2D what we know already.

In practice, in 2D, to find a extrema we need to guarantee that the first derivative is zero, which in N dimensions is correspondent to guaranteeing that the gradient is zero in that point, because the gradient is nothing more than all partial derivatives in that point. Therefore, setting $\nabla f = 0$ means that, in that point, the function is not increasing or decreasing in any of the N directions. So the first derivative makes sense.

However, when we go to the second derivative, the meanings get a bit more complicated.

In 2D, if the second derivative was 0, it was probably (not certainly) a saddle point, if it was > 0 or < 0 it was, respectively, a local minimum or maximum.

The conditions we should impose in 3D is to have a positive (for finding a minimum) or negative (for finding a maximum) definite Hessian Matrix. While second derivatives in order to just one variable is possible to attribute a sense to it, why do the cross derivatives play a role as well? And, why do they mean really?

Well, first the explanation on why it is needed: there are functions that across multiple dimensions still show that it is an extrema but then there's an inflexion along directions that are not along the axis. So, checking the axis is not enough. Why checking the cross partial derivatives makes it enough?

The Second Derivative Test

$$f_{xx}(x_o, y_o) f_{yy}(x_o, y_o) - f_{xy}(x_o, y_o)^2 \gtrless 0$$

If it is greater than 0, we have a maximum or a minimum and have to check the value of f_{xx} or f_{yy} to be sure. If it is less than 0, we have a saddle point. If it equals 0, then we don't know if it is a saddle point, but it is not a min or max therefore, at least for now, we certainly don't care.

Cross or Mixed partial derivatives can be switched? Yes if the function is C^2. (Boring to prove theorem called: Schwarz' Theorem)

Therefore, we just need to compute one of the cross derivatives.

Also this works because the second derivative test is nothing more than the determinant of the Hessian matrix. The determinant is the product of every eigenvalue of that matrix, therefore it can only be positive if they are both positive or both negative, in which cases there is, respectively, a minimum or a maximum.

But why do eigenvalues tell us this? Because they tell us how the eigenvectors are scaled! And the eigenvectors of such matrix will be the greatest and the least curvatures. Therefore, they either have the same signal / are scaled the same amount, or

Some other links that helped with this:

- Differential Geometry
- Criterior for critical points - Maximum, Minimum or Saddle?
- David Butler - Facts about Eigenvalues

3.2.1 Neural Networks - BackPropagation

Two of the most useful websites to check while trying to demonstrate the backpropagation algorithm:

- all backpropagation derivatives
- Error (deltas) derivation for backpropagation in neural networks
- 3Blue1Brown Neural Networks - Specially the last one, on backpropagation.

Why kernels are important? They facilitate a lot the mapping of features to higher dimensions!

Why kernels?

3.3 Unsupervised Learning

3.3.1 Lagrange Multipliers

Perfect Explanation why it works:

Quora - Intuition on Langrange Multipliers

And a video showing exactly how it's done:

Khan Academy - Lagrange Multipliers Example

3.4 Reinforcement Learning and Decision Making

An agent to make the wisest decision in its situation has to have knowledge on what state he is in, how the environment will evolve, how it will be like if he performs a certain action (taking into account stochastic environments) and a utility function to know how much that state contributes to its happiness.

5 Database work - SQL

5.1 SQL commands

The basic syntax is:

SELECT {Column name } FROM {Table name} WHERE {condition}

From here, there are a lot of variety that can be added to be possible lot of flexibility.

There are also other types of commands, that I haven't used so I won't talk here. Search :)

5.2 Browsing Tool with Filters

To use filters while using DB browsing tool installed on linux from sqlitebrowser.org with:

```
1    sudo apt install sqlitebrowser
```

Consult their wiki on that, it can be found here .

Additionally, is possible to export the result of filters to a csv by selecting the table with the mouse, pasting it in the side window and selecting "Export", not forgetting to add the .csv extension to the file name.

One last important feature: is possible to import and merge tables: if there is a similar database with a table with the same name (and structure) is possible to open that database with the same program, press on the table and Export it to csv. Then when opening the database where we want to import a table or various rows to, we select import from the File menu and select the csv. Note that the columns names in that csv may be in the first row. If this happens, we must check the square box that says "columns in the first row" so that the table can be read appropriately. Then it will ask if we want to merge and the previous table with the same name as the imported table will have the new rows from the imported table. If we want to do some filtering, for instance, *if we just wanted to import certain rows* we can now filter the table, or we could've filtered the table that was imported before importing it.

6 Visual Studio Code: The Environment for Development

This whole document was created with LaTeX on Visual Studio Code.

I ran commands for installing the necessary LaTeX stuff, these can be seen below.

I installed vscode with the software center from the .deb package downloaded throught their website and then installed 2 extensions. The first extensions is Latex Workshop from the extensions market inside vscode. The second I can't remember which one was or even if it did something...

```
1     sudo add-apt-repository ppa:jonathonf/texlive
2     sudo apt update && sudo apt install texlive-full
```

In addition, I configured *Ctrl + .* for compilation opening every shortcut with *Ctrl + K + Ctrl + S* (ctrl + K then lift the K key and press S). However, with the dual window mode, just as I it *Ctrl + S* to save, it automatically compiles which is very pleasant. *Alt + Z* for word wrap is useful too. To finish is possible to have 2 windows side by side in the same vscode instance by right clicking the window on the top bar and selecting "Slit Left/Right".

Some useful things to know:

- F1 opens the command pallet. From there, type what you need :P
- Ctrl + Alt + V -¿ open on a window on the right the document that is being edited on the main window;
- Ctrl + Alt + H -¿ go to the same place on the LaTeXdocument as where the source is;
- You can commit and push and all that git stuff with vscode;
- Ctrl + Alt + A -¿ After selecting a word, adds it to user dictionary, therefore is not considered an error anymore.
- Ctrl + T + Ctrl + A -¿ Toggle Activity Side-bar (shows files, ect...)

7 GitHub

So far, what a pain. A globally used tool that takes so much to learn. Here's the short guide.

Crete or clone

It depends who starts it. If you start it, you have to create it. If someone else already created it, then clone.

In the creation part, it can be done in the terminal, with some kind of GUI/in an IDE (Visual Studio Code and Android Studio do this very nicely). Or it can be done on a Git website which is has been necessary for me, due to terminal problems.

Create on Website. Works all the times, then clone it.

How Git works

Other important files

They are: gitignore, README, license...

GitIgnore is made to make git ignore certain files in the directory, so that those aren't included in the repository. Examples of those files are the .log files generated by compilations, build auxiliary files, etc...

About README, there's a whole website about this practice HERE. Normally, markdown is used to do this and that is a fairly easy language that can be fully consulted in HERE.

Another very good guide done on GitHub is Cheatsheet.

Regarding the license, it should be included if the code is to be used by others. A good example of a classical license is the MIT License. The website Choose a License explains everything perfectly.

7.1 After Carolina's help

merge this subsection with the above things.

7.1.1 Start a repository

Git global setup

```
git config --global user.name "João Morais"
git config --global user.email "joao.morais.jumper@hotmail.com"
```

Create a new repository

```
git clone https://gitlab.com/jmoraispk/test.git
cd test
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Push an existing folder

```
cd existing_folder
git init
git remote add origin https://gitlab.com/jmoraispk/test.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Push an existing Git repository

```
cd existing_repo
git remote rename origin old-origin
git remote add origin https://gitlab.com/jmoraispk/test.git
git push -u origin --all
git push -u origin --tags
```

7.1.2 Pull

When you want to pull, two cases may happen:

Either there are no changes in you local repository to the last remote repository and you can simply pull and the changes in the remote repository are implemented in your local one. In this case, simply **pull**

Or, in case you made changes to your local files, a merges must occur! So do, **git stash** to put your changes safely in a stash, then **git pull** to overwrite your local repo. Then **git stash pop** to take you local changes out of the bag and attempt a merge. Usually, it goes well. If it doesn't check the merge section!

7.1.3 Merge

...

7.1.4 SSH key

"You won't be able to pull or push project code via SSH until you add an SSH key to your profile" Is it even necessary? It seems to work fairly well through https.

7.1.5 Delete a repository

It is hard on purpose!

To cut local updates is just to delete the ".git" file.

To delete the remote repo as well (usually not needed), it required to open GitLab/GitHub page, open project repository, settings, general, advanced and delete project.

8 Interesting stuff and People

8.1 ArcXiv

It's pronounce "archive" because the X is read as a χ (Chi, the greek letter).

arxiv.org has more than 1.5 million papers, including the paper published by **Grigori Perelman** , the man who cracked the first millenium problem rejecting the one million euros price and won a Fields medal rejecting the medal as well because he didn't want fame:

> "After 10 hours of attempted persuasion over two days, Ball gave up. Two weeks later, Perelman summed up the conversation as follows: "He proposed to me three alternatives: accept and come; accept and don't come, and we will send you the medal later; third, I don't accept the prize. From the very beginning, I told him I have chosen the third one ... [the prize] was completely irrelevant for me. Everybody understood that if the proof is correct, then no other recognition is needed."

Overall, a place to read freely about what is and what once was the state of the art in science.

8.2 The writings of IST president

He writes well. Very well. After hearing him once talk for less than 5 minutes I did see why he was president.

Some of his writings in Public, one of the most well known portuguese newspapers, can be found in publico.pt/autor/arlindo-ol
.

Just in case he gets retired or something, his name is **Arlindo Oliveira** .

8.3 YIFY/YST release group

A release group so famous for their quality content, speed of delivery and that due to their websites taken down somewhat simultaneously.

Most of movie related piracy would still have their names attached. In quite a few cases, they don't have anything to do with it anymore.

In conclusion, search [YSF][YIFY] when you are searching a movie, it will be found much much faster.

Likewise, searching for yify subtitles will give great results very fast as well.

A good way of organizing movies would be: movie, jpeg with credits for release (yify jpeg), subtitle for yify and torrent source. Then ship all this to the drive.

About formats, in essence BLU ¿ WEB ¿ everyhing else. You can read everything about them in the Wikipedia: Pirated movie release types.

8.4 Interesting links

- https://www.sciencedirect.com/browse/journals-and-books a website with many scientific articles

9 Books

Every book I find interesting, reasons why I want to read it and what I thought after reading it.

9.1 Emotional Inteligence - Daniel Goleman

9.2 The Digital Mind - Arlindo Oliveira

9.3 Inteligência Artificial - Arlindo Oliveira

9.4 12 Rules for Life: An Antidote to Chaos - Jordan Peterson

9.5 Maps of Meaning - Jordan Peterson

9.6 Enlightenment Now: The Case for Reason, Science, Humanism, and Progress - Steven Pinker

9.7 The Better Angels of Our Nature: Why Violence Has Declined - Steven Pinker

9.8 The Beginning of Infinite - David Deutsch

9.9 How We Know What Isn't So - Thomas Gilovic

Another similar to this one is "Thinking, Fast and Slow" by Daniel Kahneman and both of them are important scientists that evaluate to what extent we (humans) don't make the logical decisions every time and what other factors influence ours decision making. We are probably not making the most rational decision because of those factors...

10 A few lessons

10.1 Be professional & make up your mind

Make your decisions. Sometimes in life you have to know what you want to do to make effective choices! Don't let the time pass, don't be a fucking passive cunt that only watches a mess unroll in front of you. Be there, be conscious of the decisions you are making and the impact they'll have and make them anyway because if you don't, you'll just get a random result and you'll probably piss-off and disrespect the people around you that are dependent on your decisions.

10.2 Insure properly

Insurance is the best way of risk transferring. Ensure until the money you would get back is preferable to the contents of the package. The money you are spending in the first place is probably completely irrelevant to make sure you get out of the situation winning.

This lesson came from insuring the package for 100 euro and paying for the shipping 33.5 euro while I could have insure it for 500 euro paying only 36. Of course the package got lost and I totally regret because it had important things that costed much more than that and that I valued much much more than that. Also I am in a phase I would like to buy a 500 euro headphones for motor cortex stimulation during workouts/trainings and I just sold a huge amount of hours of work in notes in TU Delft for 100 euro. Looking back, 500 euro wouldn't even be enough.

Be fucking sure you value your time properly.

10.3 Read

...