INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Asignación de recursos y diseño de redes de transporte con sistemas distribuídos poco acoplados (Cerebros líquidos)

Algorithms for solving resource allocation and transport networks with low coupled distributed systems (liquid brains)

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A

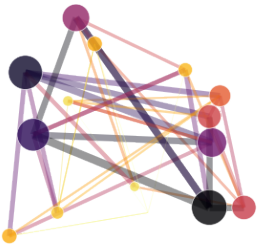
Jerónimo Aranda Barois

ADVISOR: **Carlos Fernando Esponda Darrington**

CIUDAD DE MÉXICO **AÑO 2019**

Content table

1. INTRODUCTION



Computing has turned out to be the very **essence** of things, nature is about information and change, as the highest known thinkers, humans should reflect on **nature** and realize themselves as single **neurons** of a **bigger brain**.

1.1 Context

When reading a thesis on computer engineering or computer science, you may ask himself what is computing? Computing in most cases seems to be a problem related to silicon computers, however, from the scope of information theory computing can be generalized to any information processing task. From this point of view, we can expand our horizons and expect any kind of information processing to appear in this thesis, for example learning, solving or communicating.

This very general definition allows us to use computing as a perfect framework to try to understand any domain information processing situation, however weird or separate from silicon computers it may be, this includes economics, social sciences or even biology.

To start it is important to make some definitions, **distributed systems** in the computational world refer to computing agents that somehow compute and share information, this systems are typically formed by a group of computers and a communication network that connects them, given this definition we can identify neural networks or even human networks as this kind of **distributed system**, not caring if the computations are biochemical, silicon or physical based. Citing **Michael Levin** "It's not about what you're made of, it's about how you compute." [1]. In the case of **neural networks**, **neurons** act as the agents that are able to share and compute information.

Having introduced this **distributed system** definition lets introduce some problems that can be solved by them, from a social science perspective this kind of problem solving can be called collective problem solving [2], which cover Crowd solving, Collective action, Collaborative intelligence, Mass collaboration, Collective wisdom, The Wisdom of Crowds, Distributed knowledge, Online participation, and Group decision-making. These days where we are constantly being attacked by fake news, consensus seems to be really important for any kind of social network, Governments and communities depend on consensus to work well, it determines if the system "trusts" itself and makes the difference for the system to **cooperate**. Society is able to compute whether a piece of news is actually true or false at some extent. As a society it may be important for us to develop or design a methodology where news or anything can be certified, this may remember some people of blockchains, however this thesis pretends to look at the problem from a different approach, it may also try

to solve other relevant problems in the context of low coupled distributed systems (liquid brains), and if possible try to establish the relation between liquid brains and universal Turing machines.

From a biological or ecological point of view, we find ants or bee swarms or whichever biological populations that cooperate. In Economy you may find problems as resource allocation or transport networks. Other interesting problems are calculating maximums between agents, exploring geographic conditions and finding optimums, matching agents (chemical reaction simulations) to enumerate a few.

It is important to bring out the word **brain**, which I recognize as the word for distributed systems in a biological context, a distinction and introduction between **solid brains** and **liquid brains** should be made, *"Some of these networks are describable as static sets of neurons linked in an adaptive web of connections. These are 'solid' networks, with a well-defined and physically persistent architecture. Other systems are formed by sets of agents that exchange, store and process information but without persistent connections or move relative to each other in physical space."[3]*

Liquid brains appear in a very long list of systems, *"biological systems, such as ant colonies, bacterial colonies, slime molds and immune systems, process information using agents that communicate locally while moving through physical space."[4]* Slime molds have very interesting characteristics, they do not possess a nervous system however some state that they compute! This why they can be considered as the primitive precursors of modern animal brains, even when their computation and communicating networks are based in biochemical signaling. "*Evidence mounts that organisms without nervous systems can in some sense learn and solve problems."[4]* These statements are also been held for plants.

1.2 Problem identification

Among all the interesting problems stated before, we will try to give solutions to two of them that are crucial in the economic context, however they may arise in different contexts and its solutions may be biologically inspired, these two problems correspond to resource allocation and transporting networks.

These problems have been widely studied, however, most of the times a centralized approach is taken. This approach can **overoptimize** the system making it non robust and non-adaptable. With overoptimization I mean the following, you are actually minimizing the objective function but at the same time you **miss** restrictions, adding these restrictions will turn the actual optimum **unfeasible**. These restrictions emerge organically when they are not stated centrally but in a **distributed** manner, this happens when each individual contributes or cooperates in order to reach an optimal state for the system.

An example can be the exam scheduling in a school. Assignments could be optimal but not understanding that a student cannot take 4 exams in a row is critical. Not bringing these distributed restrictions to the problem can be dangerous when optimizing human systems.

A resource allocation problem appears when a valuable and finite resource is found and has to be assigned to a given user, whether this resource is gold, CPU time, water or ant food, and the users people, ants or system users. In this broad context, allocation has to be made efficiently, especially when there are time, cost or space constraints. Because of this a well-designed transport network is crucial. The clear problem definitions can be found in the next chapter.

1.3 Objectives

Design an environment capable of simulating the solution of some classic operations research problems through the lenses of **liquid brains**, with it, improve their understanding because they have often proved to generate robust and adaptable solutions. Moreover, I have other objectives, this are to give some understanding of computing in the information theory context, ant to build some cross-domain collaboration between biology and economics through computing and mathematics.

To achieve these objectives, it is crucial to be able to correctly define the problem and model its solution with a liquid brain approach. Stating pros and cons of this approach is very important, in the case of the transport network problem we have a benchmark provided by [6]. Results and analysis will require a thorough mathematical focus.

1.4 Scope

The scope of the problems will be well defined in the following chapters, however it is very important to always consider a complexity analysis in order to correctly conclude and scale the solutions. This thesis is not trying to solve every problem but will try to implement a framework where problems can be modeled and hopefully solved by a computing approach without underestimating the complexity of real problems.

1.5 Methodology

The design methodology for the solution will be based on Karl T. Ulrich book Design Creation of Artifacts in Society [7], were the design process is considered under an information processing lens.

This methodology is based in 4 steps. Finding a gap, defining the problem, exploring alternatives and defining a plan.

The main gap is found in the difference that can arise when using central or distributed scopes when optimizing systems. The problem definition will be well defined in the following chapter. Exploring alternative solutions and its given outcomes will be found in the tests and results chapter.

The chosen plan for this approach will be:

- Research

- Provide design solutions
- Evaluate performance through simulations
- Conclude
- Iterate

Other interesting gaps which this thesis will try to narrow can be seen in the confusing definition of computing most of us understand, which only focuses in its silicon and digital implementations. It will also create links through cross domain collaborations, this day's science and technology tends to be really specialized. Biological inspiration can bring some light in life understanding and help design sustainable and robust systems.

1.6 Document organization

In the introduction you can find the **context** of the problem and some insights in the methodology that is going to be used, the second chapter will be all about clearly defining the **problem** and the **restrictions** involved in the **design** of the solutions. Chapter 3 will try to state clearly the **solution architectures**, which will be then **implemented** in chapter 5. Chapter 6 will be about the **results and tests** and chapter 7 on **ethical implications**. Chapter 4 will include **further analysis** and at last chapter 8 will be about the **conclusions**.

2. ANALYSIS

Analysis can be a tricky word for mathematicians, but it is an everyday practice in the engineering field, in this section we will try to give a general idea on what we will understand as a design solution, hoping it covers the most important sides of the problem. In the design thinking methodology this is one of the most important steps, failing here would trigger bigger failures in the future, especially because of the lack of understanding between **users** and **engineers**.

2.1 Functional requirements

A design solution that correctly solves the previously stated problems, **resource allocation** and **transport networks**, should be comparable in performance with other domain knowledge solutions, a solution should clearly **state** what problem is trying to solve and which **performance measures** will be used to evaluate it.

Resource allocation is generally understood as creating a **map** between resources and a given population. In this design we will focus in the problem where both the resources and the population are finite. The most general approach will be of having a function f(x) where x is a position and f(x) is the quantity of resource in that position. A solution of the problem is one that assigns every agent a position or a region in the space. One can ask many questions, of each solution, how fast the solution was found, is the solution 'fair', to mention some.

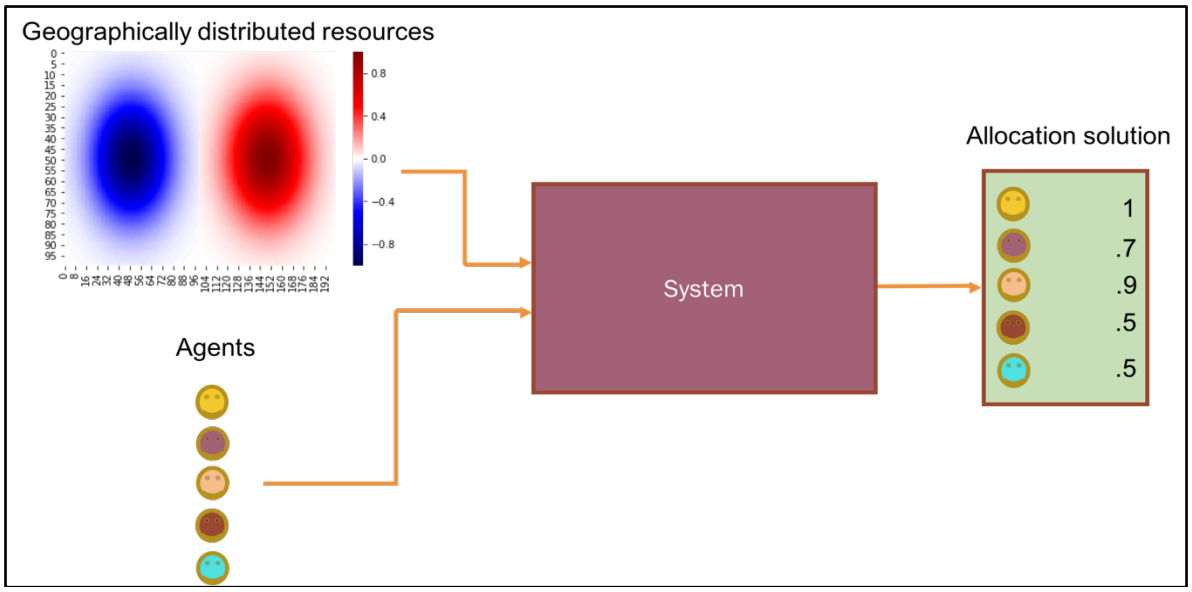


Figure 1. System architecture for solving resource allocation.

The problem of a **transport network** arises when there exist nodes in a spatial context, a solution is a network that has all of its nodes linked. Of course, every solution will have different network properties as the sum of the **cost of the links** for example, and other **flow** properties.

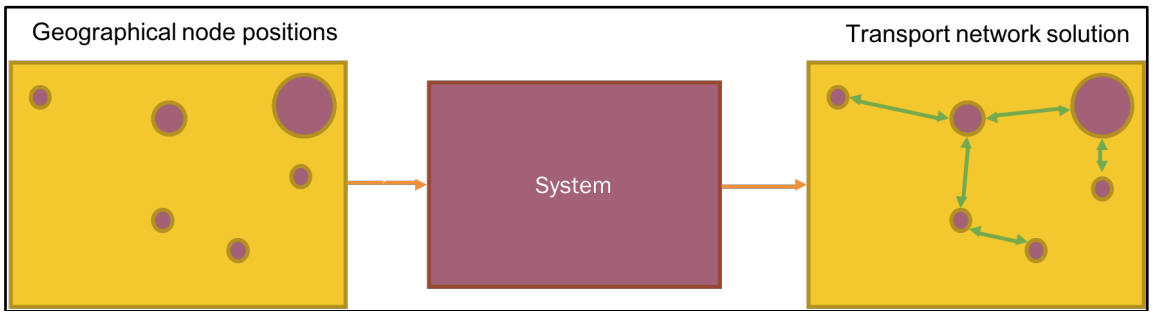


Figure 2. System architecture for solving transport networks.

A solution will be valid if it covers the following requirements, the system **obtains** the previously stated inputs and **produces** the outputs, it should include **performance** metrics which can be general or implementation specific, lastly, a **scaling** analysis of the solution is compulsory.

This thesis academic pursues will only be met if all implementations, experiments and analysis can be completely **replicated**.

2.2 Design and implementation restrictions

As such recurrent problems that appear in a wide range of ambient, from ant colonies to social networks, it has to be clear on the **assumptions** that are taken so that the conclusions that are made can be valid, this assumptions correspond to the **abilities** of the agents and the environments where the agents "**exist**", as well as the computing ambient where the solutions are **simulated**, in other words to understand the implications of the parameters mentioned in the preceding section. As the **goal** is to be able to extrapolate these results to other environments than the ones we simulate.

For the computing implementation restrictions, the models will be run in a MacBook Air (13-inch, 2017), processor 1.8 GHz Intel Core i5, memory 8 GB 1600 MHz DDR3 and graphics Intel HD Graphics 6000 1536 MB. Simulations will be done in **python**. Not being able to run every simulation can be an implementation restriction as some solutions may exceed a single computer capability.

As my background contains a lot of **biological** and **ecological** inspiration solutions may be biased by this kind of ideas. There exist also time constraints as the thesis has to be handed in before the end of the current semester. That is less than 7 months for its **implementation**.

2.3 Related work

Game theory is another approach that has been explored in understanding this type of problems. In [11] results as Nash equilibrium are used in order to **allocate** cloud computing resources to coming users to choose **optimal** strategies. Which in some sense correspond to the same problem, however in this case we wouldn't be allocating a geographical resource

to each agent, but we would be allocating ‘computing time’ to each ‘client’. This approach is also used in the **design** and **deployment** of human transport network infrastructure, for example Wardrop's second principle [12] tries to show that each user behaves cooperatively in choosing his own route to ensure the most efficient use of the whole system.

“At equilibrium the average journey time is minimum.”

This states that drivers cooperate with one another in order to minimize total system travel time. This type of results helps obtain optimal conditions in the deployments of big infrastructure projects.

The design analysis carried out in this thesis is highly computing biased and bio inspired. Other work using similar approaches for algorithm design can be found in [9], some of these algorithms have names as Genetic Bee Colony (GBC) Algorithm, Fish Swarm Algorithm (FSA), to clearly show the **bio-computing** cross domain collaboration.

Some of the other concepts developed in this thesis have already been worked by several people before, for example the book *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds (Complex Adaptive Systems)* introduces some of the concepts, scientific papers as *How does mobility help distributed systems compute?* And *Liquid brains, solid brains* and even in some philosophical works as *Society of mind* by Marvin Minsky and *A new kind of science* by Stephen Wolfram.

3. DESIGN SOLUTION

Transport network design and resource allocation have been popular problems on earth for thousands of years, resource allocation is crucial for life to remain, as lack of resources will for sure mean death. Because of this I find it easy to say that hundreds of solutions and strategies have already been stated, microbes follow chemical gradients, mosquitoes smell mammal breath, ants create paths around the jungle, to mention a few. As humans, we have also found some good solutions to these problems, transport networks made of asphalt and concrete cover then whole planet, as well as railways, telecommunication networks and other kind of transport networks. Inspiring design with solutions found in completely different contexts might not give us the optimal solutions to our special cases but will for sure give us a better understanding of what nature is and how it computes. Our system can be understood as a brain, a complex entity that solves a problem that emerges from the interaction of single and simple entities.

3.1 Architecture

Two approaches will be used to solve and understand the given problems, one as a dynamic network problem and one as a liquid cellular automata problem, which at some point could be different approaches of the same structure. The architecture of the system has to be very well described in every implementation, as a minor change may create a chaotic effect when analyzing the system in a macro level.

Plenty of research exists in the study of liquid brains, even though might not have that name, I believe it to range from problems on fluid dynamics to computing agent-based models (ABM). But how is a fluid dynamic problem related to a liquid brain? A fluid dynamic problem is one that models how a liquid behaves in a given environment where some physical constraints exist, the Navier-Stokes equations are the main mathematical model used to represent this fluid behavior, the problem with this approach is complexity, as it is not yet proven that a solution of some of them even exist. Because of this **complexity** issues exact simulations of fluid dynamics do not exist, and the current ones have statistical approaches where strange models like convolutional neural networks can even appear. However, for the case where very few particles interact there exist very good solutions and simulations. If the relation to liquid brains is not yet clear, let's think as a particle of some liquid as an agent, in order to find the next movement, it will perform, it has to do some computing. This computing depends on the properties of each liquid and its environment, to mention some of them we can think of the gradient and texture of the surface where they are, the **momentum** they carry and other properties as **viscosity** and **liquid tension**. Of course, as computing capabilities of each agent increase liquid brains become more interesting, this means they improve their communications, their **sensing** and their **acting**. These improvements are not cheap, they come with many constraints and compromises.

The following **agent-based model** approach can give us some understanding of what liquid cellular automata are, *"We implemented LCA as an ABM that simulates simple interactions such as those found in a robotic swarm or mobile sensor network. LCA consists of N agents moving in a two-dimensional square arena of size L. Each agent is analogous to a cell in a CA, containing an internal state variable that is updated synchronously according to a rule*

table shared by all agents. Instead of a fixed neighborhood, agents have a fixed communication radius r and broadcast their current state to all other agents within that radius."[4] For the design to be successful some simulations will be carried out with some of the parameters varying to detect in which scenarios the agents perform better. These parameters that can be modified are the size of the arena, the speed of the agents or the communication radius, as well as the rule table that describes the behavior of the agents to name a few. A number of simulations will be carried out in order to understand the model better, as well as to visualize the strategies and development provided by each solution. Simulations will be carried out in **Python**.

3.2 Alternative solutions

In this chapter's introduction many different approaches where mentioned, some of them have been better explored than others, the definitions of the problems can vary a lot, in their constraints, variables and topologies of the space where they exist.

"In economics, the area of public finance deals with three broad areas: macroeconomic stabilization, the distribution of income and wealth, and the allocation of resources. Much of the study of the allocation of resources is devoted to finding the conditions under which particular mechanisms of resource allocation lead to Pareto efficient outcomes, in which no party's situation can be improved without hurting that of another party." [10]

Resource allocation solutions can range from portfolio management in finance to wealth distribution in public policy, going through choosing your dinner meal to mention some human resource allocation solutions. Humans have also worked a lot in road transport network design, in this civil engineering problem, plenty of development has been done, in this case the variables of the terrain get more complex with the introduction of soil properties, geographic relief and other metrics that have been crucial to the deployment of efficient and safe road networks.

We have developed infrastructure to cross rivers, climb mountains and reach distant places, human transport networks have even become just spatial channels that planes follow, behaving like crowded bird migration routes. Other transport networks have been deployed by animals such as ants, and some have even been discovered and used by marine animals such as sea currents that bring and move resources in the drifting ocean.

3.3 Standards

Ulrich design thinking methodology is used across the whole thesis project, in order to follow the design development all files and project changes will be hosted in github.com, which is one of the most used repository hosts that uses **GIT** language for version control to track changes.

The solution will be developed in **Python 3.7**, one of the most updated opensource programming languages that aims to imitate English syntax, most of the scripts will be run in **Jupyter notebook**'s, which is an interactive framework for interactive computing that

can run Python as well as other programming languages scripts. Further **hardware** and **software** specs are discussed in the implementation chapter at **5.2**.

As the distributed system will be **simulated** there is no need to state any **network protocols**, further implementations would require very simple and efficient protocols as the ones used in robot swarms so that interactions could happen in real time. Other network explorations can be found in the next chapter **further analyisis.**

4. FURTHER ANALYSIS

With this given approach it can be stated that this kind of **naive** liquid brain, one made of "**water**" can find the closest local minima in a given optimization problem, not just these but with infinite particles of water and infinite time it will easily find the global minima. These turns out to be a problem of handling complexity or being able to have enough computing power even though the algorithm surely converges if there is a minimum.

4.1 Fluid dynamics

Optimization, Gradient descent, global minima.

4.2 Stochastic systems

Adding uniform noise in the minpool operation in which the agent samples the utility function brings some interesting behaviors:

Are this something like **Brownian** movements with **drift**?

4.3 Dynamic networks

In the first problem resource allocation is solved by agents in what I call a naïve brain. This kind of liquid brain has a simple but strange way of communicating between neurons. The only way agents perceive each other is by the impossibility of 2 agents occupying a single cell. This restriction impose some kind of signaling that allows an agent to avoid a certain cell even when this cell is inside the agents local kernel and minimizes the minpool() function on it, but it is somehow 'taken'.

To improve the results in a noisy environment, we propose to create dynamical networks, of course this networks imply really low coupling as they can be created or destroyed anytime.

[Explain the algorithm]

4.4 Network communication dynamics

Now that an incomplete information environment is set, and agents can communicate we can ask the next question :

how can cooperation or in other words, local network communications improve convergence speed?

4.5 Learning.

This neuronal agents do not hold any '**memory'**. Plenty of other questions and dynamics can arise when these agents are able to hold any data. This could be anything, their path,

their ‘name’ or any other relevant information that could be stored to give birth to plenty of new computations and dynamics.

5. IMPLEMENTATION

Hopefully every design solution could be implemented, implementation is when all the abstract ideas finally become material and a solution emerges into the real world. In order to understand the solution, the system will be divided in small pieces. This division will allow understanding each of the modules that build the system, some animations and graphics will be provided to further visualize each piece.

5.1 Implementation description

As both problems will be attacked from the system design solution approach proposed by Ulrich, both implementations will follow the same structure. First every input will be described as the data structures that define them, and the way these inputs will be directed to the system.

Then a simulation kernel will be described, this means explaining the way that every simulation will take place, this means how the system will grab the inputs and produce the outputs. As stated, since the title, these systems will be what is known as a liquid brain, a low coupled distributed system built by interactions between many single agents.

Finally, all outputs will be listed and explained.

5.1.2 Resource allocation with naive agents.

Let's start by explaining the inputs:

Terrain and resources.

The terrain is the spatial way of how we encode our given utility function, this means that every (x,y) position will be assigned with a certain resource quantity, it is important to note that (x,y) positions are discrete and assigned before every experiment and remain static during the whole simulation.

$$F(x, y) \rightarrow \mathbb{R}$$
$$(x, y) \mapsto resources \in [-1, 1]$$

The terrain as the discrete form of the previous function will be restricted to the values in a m*n matrix. The result is what seems to be a 2D utility function that can be visualized as a 3D surface.

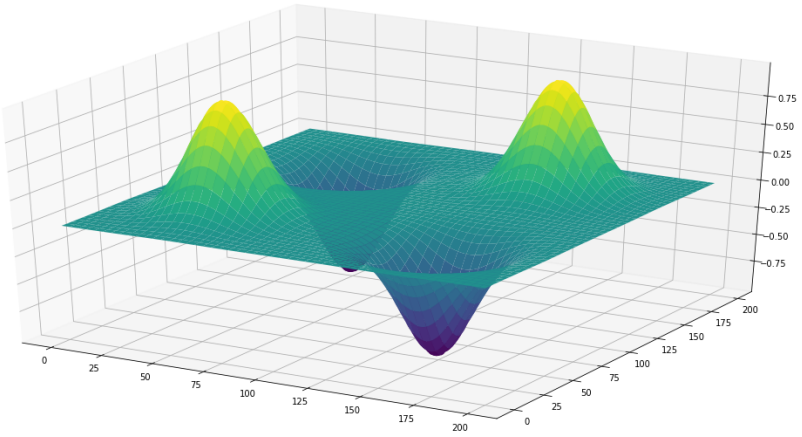


Figure 3. Terrain

Agents

Each agent will be located in a given position in the (x,y) mesh previously defined. In this implementation each agent will be completely defined by their position and their behavior, which will be later explained in the simulation kernel section.

The set of agents will be a list of N positions:

$$[[x1, y1], [x2, y2], [x3, y3], \dots, [xN, yN]]$$

All the input is initialized once the terrain and the set of agents are defined. This will look as the first frame in our simulation where every agent has a given resource quantity assigned depending on their initial position.

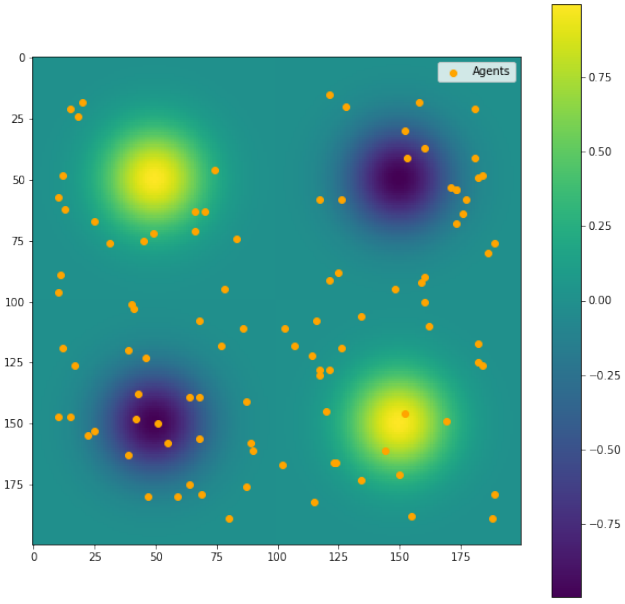


Figure 4. 100 random agents scattered in the previous terrain.

Simulation Kernel

The simulation will be given by the discrete computation of frames starting with the input frame that was stated before. Each step will require computing the next agent position of every agent in the set. The first implementations did not shuffle the agent set before updating each agent position which resulted in different behaviors, each cluster seemed to be following a *head* which was nothing but the first assigned agent in the cluster. This behavior can be reproduced in the videos in the appendix.

The agent behavior is very simple, they will move to the position that minimizes the utility function in a 3*3 kernel centered in the agent's previous position. No two agents can be in the same position, this means that if a contiguous position is already taken, the agent would not be able to move there, even if that position is better.

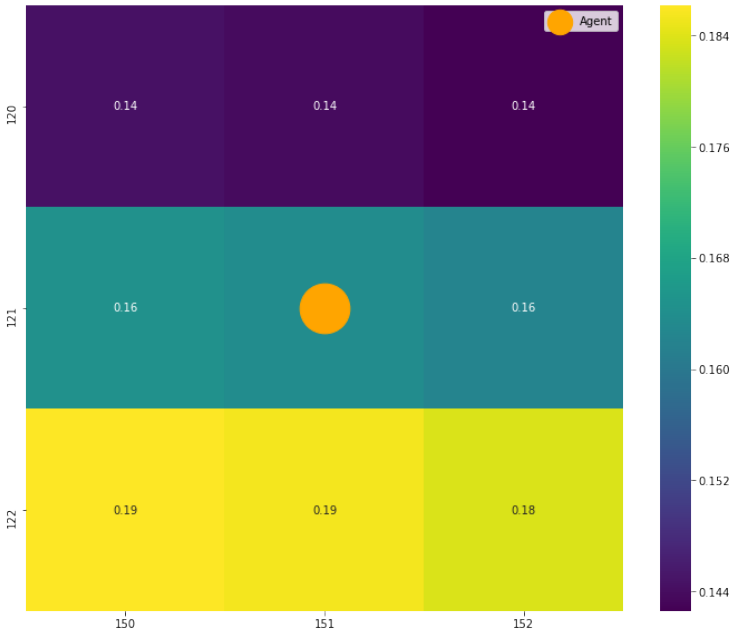


Figure 5. Agent in position [121,151] would move to the upper right position.

Convergence can be guaranteed whenever two sequential frames are exactly the same. This means that no improvement can be done by any agent.

The engine will work by following the next flux diagram:

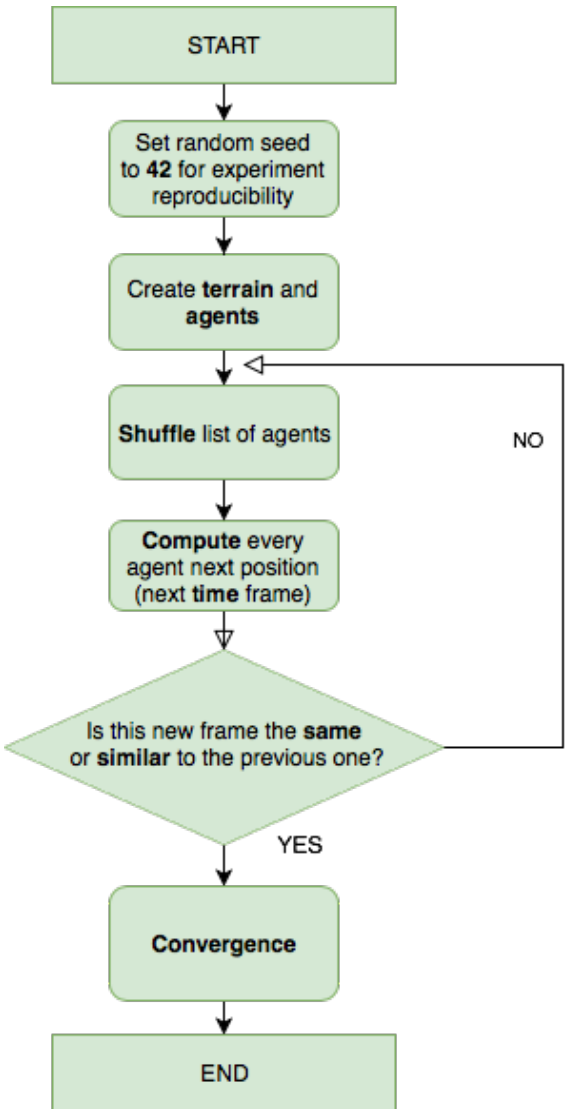


Figure 6. Main engine algorithm.

If convergence is reached, we will have an optimal assignation of resources as output.

5.1.3 Resource allocation in a noisy environment with cooperative agents

Communication networks

The implementation of this networks is done by hierarchical clustering that creates networks by adding edges to any pair of nodes which distance is equal or less than **d**.

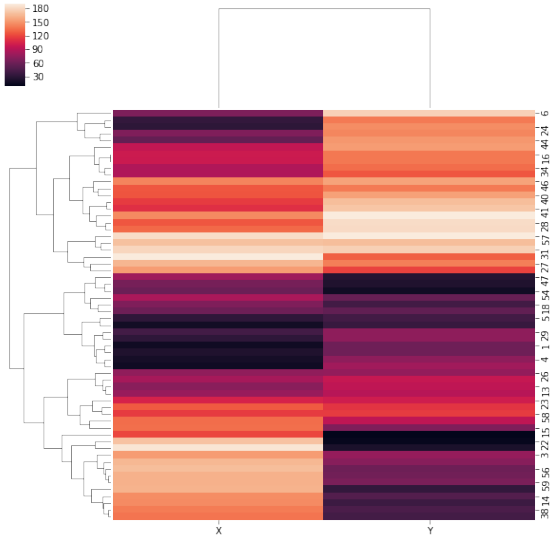


Figure 7 dendrogram with the given x,y positions of each agent.

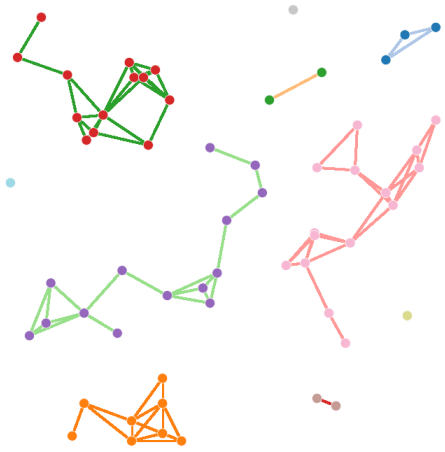


Figure 8 resulting local networks given a dendrogram vertical, choosing **d**.

5.1.4 Transport networks

Let's start explaining the inputs:

Nodes

Let's start again by introducing the inputs. In this problem the only input will be a distribution of nodes, this means again a set of (x,y) positions. The order of this list will be

important as each node will have an *importance* measure assigned to it. The scenario can be visualized as follows, encoding position and *importance* by the size of each node:

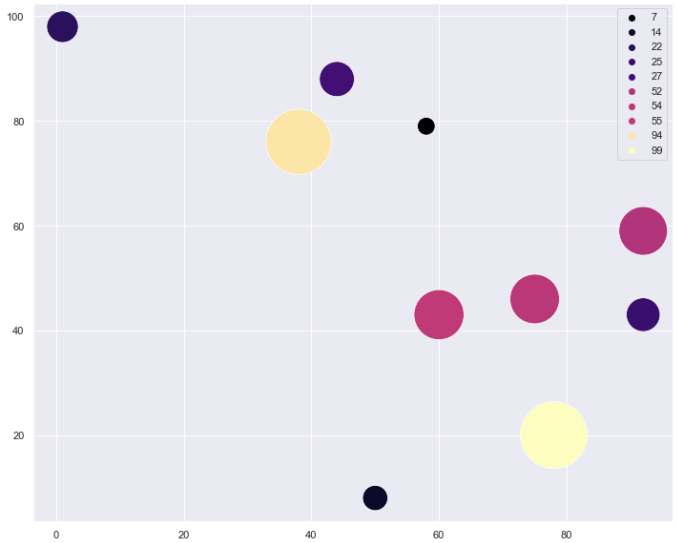


Figure 9. Random assignment of 10 nodes.

Simulation kernel

The previous framework described in problem 1 will be used to compute the solution, except that the terrain will be different. For each node, starting in the one with highest , lets say node i, a specific terrain will be created. Centered in [x,y] position of node i, there will be a positive gaussian kernel, and then for the rest of the nodes there will be negative gaussian kernels.

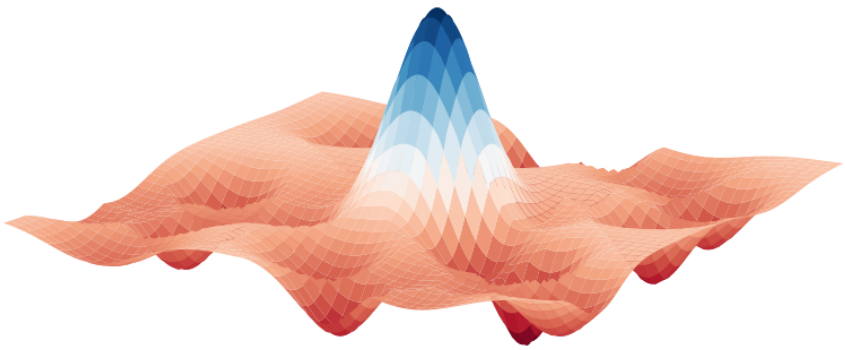


Figure 10. Chosen node as positive gaussian kernel against other nodes

Then Agents will start in node i position and they will have to slide to the minimums. An edge will be set following the path from the maximum at node i to the minimum where it arrived. As if the agents where sliding into the global minima, where the paths will be the edges. After convergence of each simulation, every edge will be saved and as output there will be a network.

5.2 Software and hardware specs

Python 3.7 is used as stated in the standards and all the following libraries are compatible with that, **Matplotlib** is the main backend for the **graphics** rendering, **Seaborn** for the **statistical** plots, **Numpy** for the linear and numerical **algebra** and **Scipy** for the statistical **computing**.

Main library versions are:

- Matplotlib `2.2.3`
- Seaborn `0.9.0`
- Numpy `1.15.1`
- Scipy `1.1.0`

Main **Jupyter notebook** (`5.6.0`):

> https://github.com/jeroaranda/thesis/blob/master/cerebros_liquidos.ipynb

All implementation and analysis **code** and **material** can be found at the following GitHub **repository**:

> https://github.com/jeroaranda/thesis.

As the complexity grows by the number of agents in the first problem multiplied by the number of nodes in the second problem, the memory and the CPU-time explodes easily, different implementations that allow parallelization would perform better. At least in really big scenarios they would certainly beat my MacBook Air (13-inch, 2017) This other implementations could be in **GPUs**, or in even faster **substrates** as it is explored in [13] and [14], a paper called *Pathways to **cellular supremacy** in **biocomputing*** published recently in **nature**, where distributed systems like millions of cells show significant **concurrency computing** gains over other centralized systems and substrates.

6. TESTS AND RESULTS

The way both problems were stated in the first chapters contain infinite different problem configurations. Here we will show the results for experiments carried out in specific configurations. This thesis is based in a digital repository where all implementations, experiments and analysis can be replicated through a Jupyter notebook execution.

6.1 Tests

To be sure the requirements were satisfied let's see the test for convergence in both problems.

For the first problem the starting configuration of the agents will be a random assignment of positions and the number of agents will be 2000. Agents in mesh = (10,195)X(10,195)

Utility function will be constant through tests, further tests should include 3D random surfaces for more robust tests.

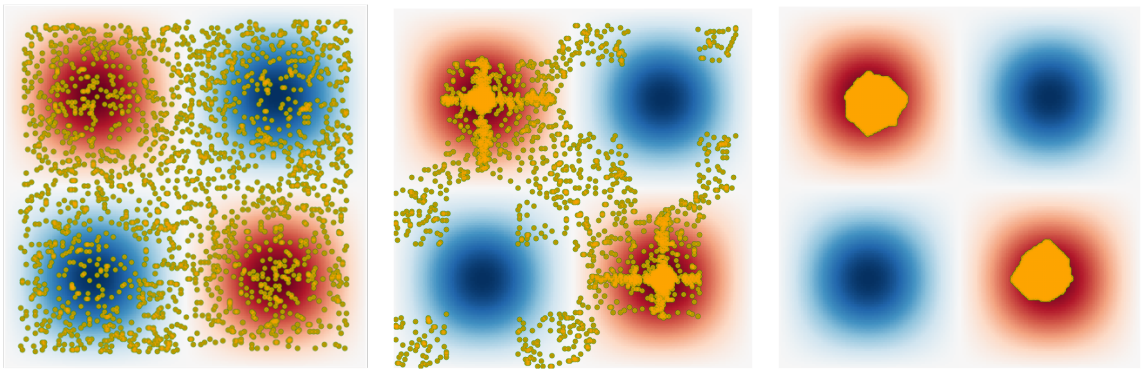


Figure 11. Random initialization of agents, halfway through the simulation and at last convergence. (Frames 0, 50, 99)

When we try for convergence in an ambient of incomplete information it gets a bit tricky as noise doesn't allow agents to be certain if they have reached a local minimum.

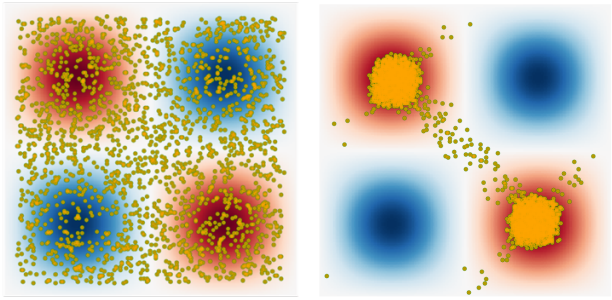


Figure 12. Random initialization of agents and reaching a certain level of "convergence" when sampling with noise. (Frames 0 and 149)

However the incomplete information might make agents jump to better global solutions as they are not always following the minpool() local rule.

Better metrics for convergence should be included here. We could try that two contiguous frames not need to be identical but very similar. In other words, "**convergence**" could be reached whenever the distance between two frames is not greater than a certain epsilon, as discussed in the further analysis section. Being able to choose the right **distance** can be interesting.

For the second problem edges from a given node to others are created, in figure 13 you can see how the agents are created on top of a node, reach other nodes by gliding to them.

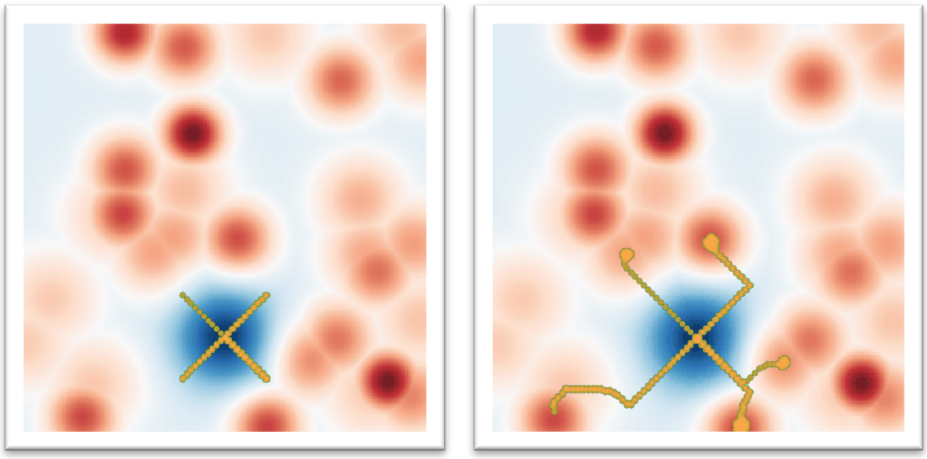


Figure 13. Some frames after initializing some agents in the top and some convergence.

This test guarantees a certain level of convergence for a transport network solution. Find general results in next section.

6.2 Results

An **optimality** and **complexity** comparison analysis is carried out and a **complete** transport network is shown.

6.2.1 Resource allocation

This system finds a way better solution than the one in which agents were initialized. Which can be later upgraded to be **pareto optimal**, as shown in the further analysis section.

In order to understand the **complexity** of this implementation lets look to the following charts for a naïve liquid brain.

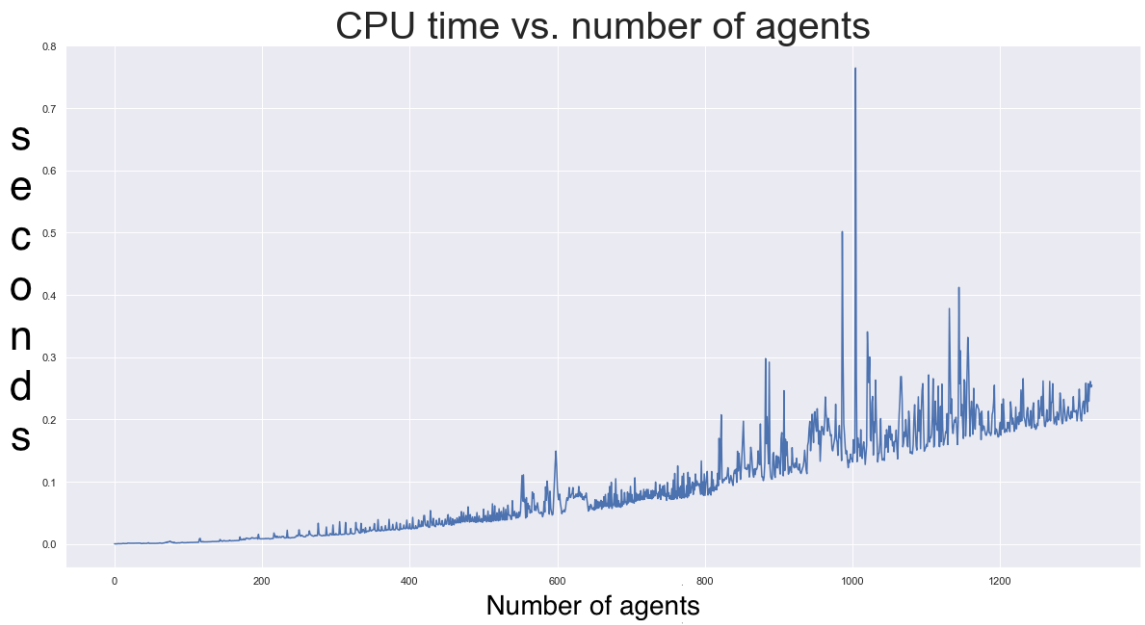


Figure 14. Frame computing time as number of agents increases.

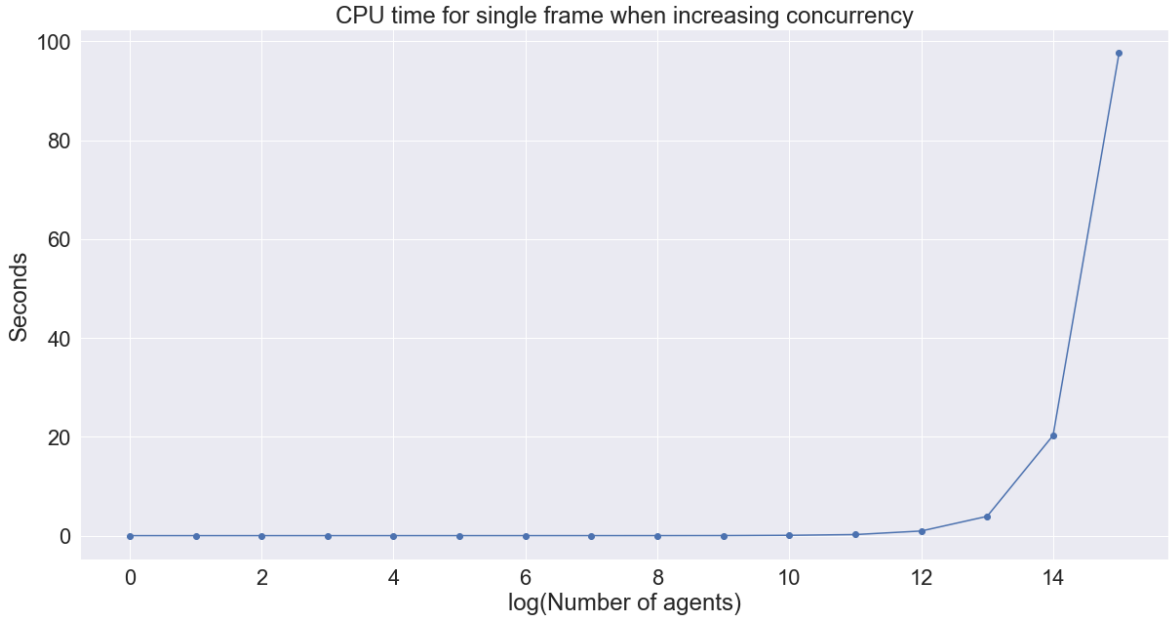


Figure 15. Logarithmic complexity plot.

As we can see the complexity explodes at 2^13, this is where better implementations or different substrates would make a difference.

6.2.2 Transport networks

The strategy to create the **terrain** is the key to having a well-designed network, you can see in figure 14 a **convergent** solution for 10 random nodes. A node is taken from the terrain creation after it gets the chance of being a maximum as explained in the **implementation**.

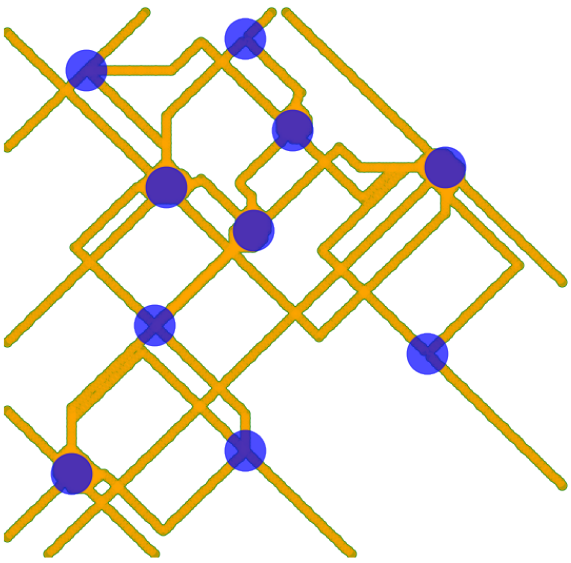


Figure 16. Complete network linking 10 random nodes.

6.2.3 Benchmarks and classical solutions comparison

Performance measures for this kind of solutions range from convergence time, ability to achieve global optima in non-convex terrains to **Pareto efficiency** that states that no improvement can be made without harming the state of other agents.

Resource allocation

As the resource allocation problem is stated at a very low level, there exists no fair comparison to the solutions stated by actual economics, where agents are humans, private companies and governments, however, it can be solved by a **classical operations research** approach, the comparisons will be made through the average agent reward as well as the total reward given by each of the two approaches, liquid brains and operations research.

The **operations research** solution problem is stated as a binary problem as follows:

Given an M*N utility mesh and n agents.

$$\min_{x} \quad z = \sum_{i=1}^{M \times N} U_i x_i$$

$$\text{subject to:} \quad \sum_{i=1}^{M \times N} x_i = n \text{ and } x_i \in [0, 1]$$

Where U is a vector of size M*N containing all the utility values in the mesh. The best possible assignment will be any ordering of the n labels given by {i : xi=1}.
Of course the solution to this problem is trivial as you can order decreasingly the elements of U and just pick the top n, but don't forget that this approach assumes that all U is **known**, or as economists say there exists perfect information.

Lets compare each solution performance for the terrain at figure 3 and simulation for 1000 agents:

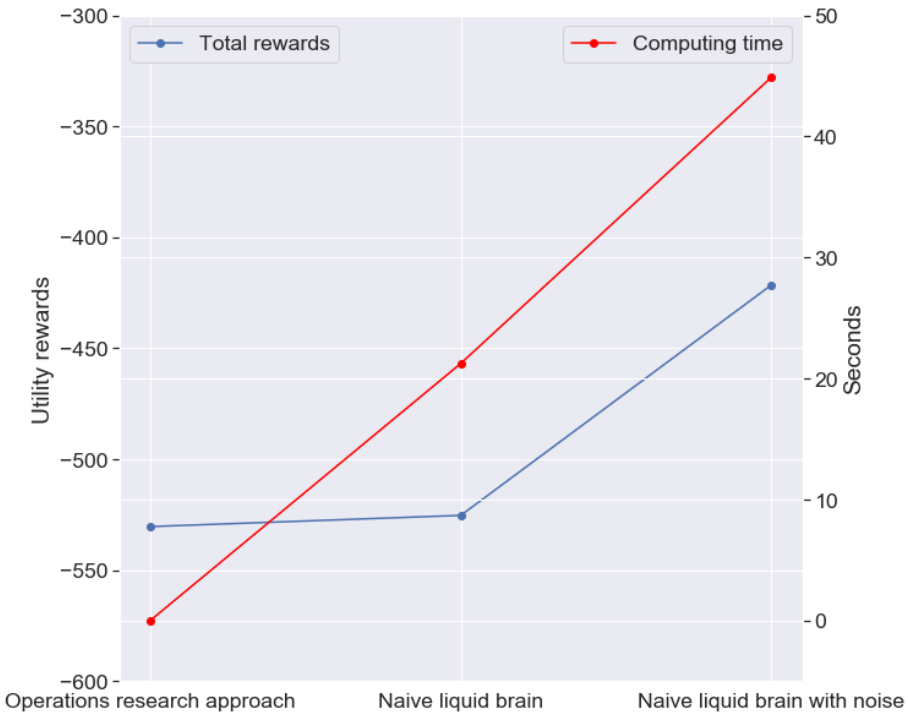


Figure 17. CPU-time and optimality comparison between methods.

As we can see the operations research approach is way faster even though the optimal solutions for the first two methods are similar, it is important to make sense that problems in reality tend to the ones in the right. If each sample of the terrain has a cost, there is an interesting compromise in choosing approach 1 or 2 as this cost changes, in other words, if the cost of perfect information is too high a Naïve liquid brain allocation would be better.

Transport networks

Better looking transport networks are created when there is no maximum created at the nodes where the agents are deployed, however there seems to be a compromise in the connectedness and its efficiency when changing from one technique to the other.

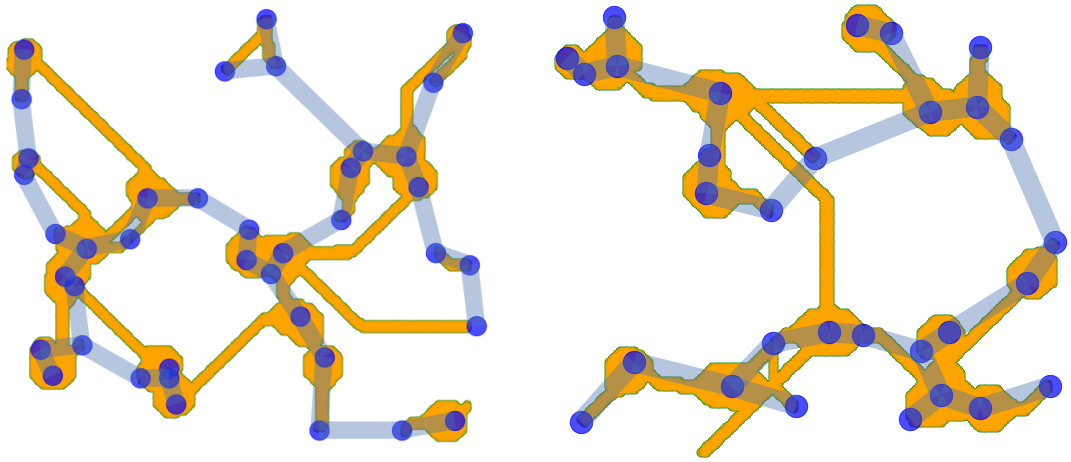


Figure 18. Minimum spanning trees and our **liquid brain** solutions made with 40 and 30 random nodes and 200 agents at each node to explore.

The transport network design solution will be perfectly comparable with the results obtained in [6]. Next paragraph has a brief introduction to the comparisons that can be done, MST stands for minimum spanning tree.

"We show that the slime mold Physarum polycephalum forms networks with comparable efficiency, fault tolerance, and cost to those of real-world infrastructure networks—in this case, the Tokyo rail system. […]Thus, for example, Physarum can find the shortest path through a maze […] or connect different arrays of food sources in an efficient manner with low total length (TL) yet short average minimum distance (MD) between pairs of food sources (FSs), with a high degree of fault tolerance (FT) to accidental disconnection […]. The performance of each network was characterized by the cost (TL), transport efficiency (MD), and robustness (FT), normalized to the corresponding value for the MST to give TL_{MST}, MD_{MST}, *and* FT_{MST}. *The TL of the Tokyo rail network was greater than the MST by a factor of ~1.8 (i.e.,* $TL_{MST} \approx 1.8$*), whereas the average* TL_{MST} *for Physarum was* 1.75 ± 0.30 *(n=21) ."*

Running an experiment with the nodes in the same configuration as the ones of Tokyo rail system has plenty of results to compare with, Tokyo rail system, Physarum way of exploring, finally with the results of the given approach used in the cited paper and finally the minimum spanning tree which is the shortest possible network connecting all the city positions.

7. ETHICAL IMPLICATIONS

Taking into account ethical implications of an undergrad thesis should be a compulsory exercise. When I focus in the problems, I am working with I do not see any evident ethical dilemmas, however, we are dealing with the design of brains, very important part in the human experience which if altered could result in big changes. To mention evident ethical implications when dealing with these topics there are some state-of-the-art experiments where brain tissue is already being kept alive in laboratories, these organs show actual thinking activity which is not yet perfectly understood. Controversial dilemmas appear immediately, do these brains actually contain an actual being? This and other questions arise and could result in disastrous answers if rigorous scientific and engineering work is not done under correct ethical frameworks.

University has an enormous responsibility in being able to approve or reject certain designs as it holds a major role as society critic. If we see an undergrad thesis as the professional profile that a student holds it is crucial that all this ethical revision is done. We better start doing this in university where responsibilities are not big, and we are in a safe terrain if we fail.

Only if we are able to state the right problems and find them a solution is how we will be able to evolve to a future where something better than humanity will rise. It is interesting to do a revision of all the ethical schemes that have come out during human history, however, this reflection must be done towards the future, where I believe the most interesting problems live, they are coming around the corner. Big challenges exist already with the possible realities that technology has made possible.

Human and machine have today the blurriest limit that has ever existed between them, and the future looks way blurrier, even though this thesis does not hold any direct implications in ethical dilemmas, my engineering responsibility should transcend my thesis work, this reflection should be a recurrent exercise in any problem that the future brings. Engineering tools should be taken seriously. Humanity already has the power to destroy the whole planet or to take the right path and improve.

8. CONCLUSIONS

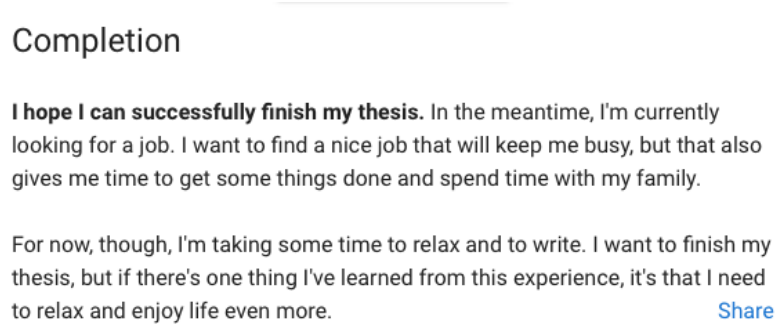


Figure 19. GPT-2 text completion.

The previous text written automatically by one of the most advanced language generation models states very clearly what my experience working in my undergraduate thesis has been.

Liquid brains show a clear benefit when handling **concurrence**, it is convenient to explore them to generate **organic** and **adaptable** solutions, in addition to improving our understanding when approaching more **general** intelligences.

I have also got to know what the limits in our knowledge in biocomputing look like and how continuing this exploration could help expand them by running many other experiments and analysis. Engineering **design** experiences allow us to generate **knowledge**, while creating a product, whose utility is embodied in them forever once they are implemented. This design process has been succesfull and it has sure improved my engineering skills by allowing me to convert research knowledge into succesful implemented solutions.

APPENDIX

REFERENCIAS

[1] https://www.quantamagazine.org/slime-molds-remember-but-do-they-learn-20180709/

[2] https://en.wikipedia.org/wiki/Problem_solving#Collective_problem_solving

[3] Sole´ R, Moses M, Forrest S. 2019 Liquid brains, solid brains. Phil. Trans. R. Soc. B 374: 20190040. http://dx.doi.org/10.1098/rstb.2019.0040

[4] Vining WF, Esponda F, Moses ME, Forrest S. 2019 How does mobility help distributed systems compute? Phil. Trans. R. Soc. B 374: 20180375. http://dx.doi.org/10.1098/rstb.2018.0375

[6] https://www.researchgate.net/publication/41111573_Rules_for_Biologically_Inspired_Adaptive_Network_Design

[7] Ulrich, Karl T. 2011. Design: Creation of Artifacts in Society. University of Pennsylvania. (ISBN 978-0-9836487-0-3)

[9] Ashraf Darwish,
Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications,
Future Computing and Informatics Journal,
Volume 3, Issue 2,
2018,
Pages 231-246,
ISSN 2314-7288,
https://doi.org/10.1016/j.fcij.2018.06.001.

[10] https://en.wikipedia.org/wiki/Resource_allocation

[11] Nezarat A, Dastghaibifard G (2015) Efficient Nash Equilibrium Resource Allocation Based on Game Theory Mechanism in Cloud Computing by Using Auction. PLoS ONE 10(10): e0138424. https://doi.org/10.1371/journal.pone.0138424

[12] Introduction to the Transportation Network Design Assoc. Huseyin Ceylan https://ktu.edu/uploads/files/fakultetai/Statybos%20ir%20architekt%C5%ABros%20fakultetas/files/Introduction%20to%20the%20Transportation%20Road%20Network%20Design.pdf

[13] Mian, I & Rose, Christopher. (2011). Communication theory and multicellular biology. Integrative biology : quantitative biosciences from nano to macro. 3. 350-67. 10.1039/c0ib00117a. http://brown.edu/Departments/Engineering/Labs/Rose/papers/CDI08_9.pdf

[14] Grozinger, L., Amos, M., Gorochowski, T.E. *et al.* Pathways to cellular supremacy in biocomputing. *Nat Commun* **10,** 5250 (2019) doi:10.1038/s41467-019-13232-