Synthetic AI Network Analysis: Socially Aware AI Agents

EECS 5414 Information Networks
Project Final report

Raghad El-Shebiny
Depts. of Computer Science and Computational Arts
Lassonde School of Engineering
York University
raghads@my.yorku.ca

Farnaz Beidokhtinezhad
Dept. of Computer Science
Lassonde School of Engineering
York University
fbeid@cse.yorku.ca

Abstract

There have been numerous studies on the construction, online analytical processing, and mining of information networks in multiple fields, including social network analysis, data mining, and communication networks given the prevalence of information networks and their broad applications. Also, there has been a growing interest in analyzing information framed as graphs in recent years, owing to their rich expressiveness and natural ability to depict complex relationships. There are many techniques for analyzing and creating static graphs, that cannot evolve and are capable of representing only a single data snapshot. However, in the real world, networks are repetitively altering. The goal of this project is to develop a method for analyzing synthetic unweighted, directed, dynamic graphs generated from AI simulations. In particular, we defined our network as a large system of agents whose behaviour, social awareness, and geographical awareness are defined by their neural network, as stated in their genetic code analysis on the whole network to evaluate the agents' behaviours and evolution. To the best of our knowledge, this is the first work that focused on creating and analysing a system that has a **unique genetic code** and a neural network that determine behaviour for each agent and their exchange information and evolving over time.

Keywords: Machine Learning and AI, Synthetic Information Networks, AI communities, Machine Learning

1 Introduction

Graphs indeed signify the inter-dependencies by edges or links between the related objects. As a result, a graph network is a strong way to represent a set of objects and their relationships. Most recent researches have concentrated on evaluating the relational structure of data employing graphs as a powerful data representation. The significance of these studies can be seen in a variety of applications, including social networks that connect users through relationships such as transactional communications which is an example of a dynamic network which is constantly changing its structure or characteristics.

In recent years, ARTIFICIAL Neural Networks (ANNs) have been widely employed in a variety of applications due to their evolutionary systems that may be represented as dynamic networks. Many papers discuss how artificial neural networks (ANNs) learn and evolve by running evaluations on each individual ANN and then selecting the fittest based on the evaluation (survival of the fittest methodology), similar to [19] that introduced EPNet, a new evolutionary framework for growing artificial neural networks (ANNs). EPNet's evolutionary algorithm is based on Fogel's evolutionary programming (EP). EPNet evolves ANN's architectures and connection weights (including biases) simultaneously in order to reduce the noise in fitness evaluation. In contrast to most past work on developing ANNs, [19] focuses on evolving ANN behaviours. However, no publications (based on our literature review) discuss ANNs being linked to agents who swap a bit of their genotype (DNA) and have it influence the agent's phenotype (behaviours) via modifying the neural network (decided by the genotype).

In this project, we aim to use the advantage of artificial networks for creating a comprehensive information network that can be modeled as graphs. The main contribution of this work are as follows:

- This study focused on creating a network where each agent (node) has a neural network that controls aspects of their behaviour and awareness of other agents around them which will be analyzed as an Information Network.
- This study discussed the evolving Artificial Neural Networks in Agent Communities (Artificial Information Networks). We want to do this by simulating an information network with AI theories and analysing the results.

According to our information, this is the first effort that focuses on developing and analysing a system with a unique genetic code and a neural network that determines behaviour for each agent as well as their information interchange and evolution through time. Our network construction uses the main advantages of Artificial Neural Networks and dynamic graph properties that have been explained completely in the section 1.1 and section 1.2 correspondingly.

1.1 Artificial Neural Networks

In recent years, artificial neural networks (ANNs) have been extensively employed in a variety of applications. Artificial neural networks are being developed by researchers from a range of scientific fields to tackle a variety of issues in pattern recognition, prediction, optimization, and associative memory.

Artificial neural networks are computational networks that aim to replicate the decision process in networks of nerve cells (neurons) in the biological (human or animal) central nervous system, as their name implies. This is a general cell-by-cell (neurons-by-neurons, element-by-element) simulation [1]. It is based on neuroyeasphysiological understanding of biological neurons and biological neuron networks. It therefore varies from traditional (digital or analogue) computing devices, which aim to replace or accelerate human brain processing without concern for the arrangement of computing parts or their networking [11], [20].

As shown in Figure. 1., artificial neural networks are a technology based on brain and nervous system research. These networks are modelled after biological neural networks, while they only utilize a subset of the principles found in biological neural networks. ANN models, in particular, mimic the electrical activity of the brain and nervous system. Processing elements (also known as a neurode or a perceptron) are linked to one another. The neurodes are usually organised in a layer or vector, with one layer's output acting as the input to the next layer and maybe further layers. A neurode can be linked to all or a subset of the neurodes in the next layer, replicating brain synaptic connections. Weighted data signals entering a neurode imitate nerve cell electrical stimulation and, as a result, information flow throughout the network or brain. A connection weight, $w_{n,m}$, is multiplied by the input values to a processing element, i_n, to replicate the strengthening of neural networks in the brain. In ANNs, learning is simulated by adjusting the strength or weights of the connections.

1.1.1 Cellular Automaton.

Cellular automata (henceforth: CA) are discrete, abstract computing systems that have proven valuable in a range of scientific domains as both general models of complexity and more specialised representations of non-linear dynamics [22]. The main advantages of Cellular automata can be summarized as follows:

- CA are geographically and temporally discrete: they are made up of a limited number of homogeneous, simple elements called atoms or cells. The cells instantiate one of a finite number of states at each time unit. They evolve in discrete time steps in parallel, following state update functions or dynamical transition rules: The state of a cell is updated by taking into consideration the statuses of cells in its immediate vicinity (there are, therefore, no actions at a distance).

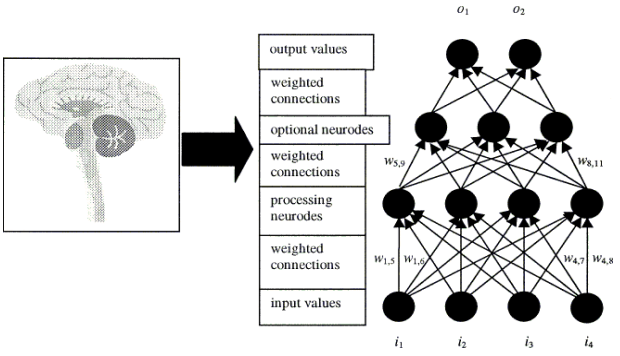

Figure 1. Sample artificial neural network architecture [15]

- CAs are abstract: they may be stated in solely mathematical terms and implemented using physical structures.
- CAs are computational systems that can solve algorithmic issues and calculate functions.

The theory of cellular automata is extremely complex, with basic principles and structures capable of creating a wide range of surprising behaviours. Cellular automata are available in a wide range of forms and sizes. The type of grid on which a cellular automaton is calculated is one of its most essential aspects. A one-dimensional line is the simplest type of "grid." Square, triangular, and hexagonal grids can be regarded in two dimensions.

The key distinction between CAs and our network is that CAs have a set of established (hard coded) rules that are followed by all agents without any uniqueness and evolution, however, our system has a unique genetic code and a neural network that dictates each agent's behaviour, communications and evolve over time.

1.2 Dynamic Graphs

A dynamic graph can be represented as an ordered list or an asynchronous stream of timed events, such as additions or deletions of nodes and edges. Dynamic graphs generation, analysis and extracting embedding from such graphs have attracted great attention.

A conversational dynamics influence in communication network graphs has been studied in [12] by Khrabrov et al. on Twitter as the reference network and data source. They focused on building a replier graph from each user A's messages mentioning another user B , and studying how this graph evolves by computing a pagerank-type score.

A novel class of evolving range-dependent random graphs providing a tractable framework for modelling and simulation has been introduced in [9] by Grindrod et al. In their study, a spectrum technique for calibrating a set of edge ranges from a succession of network images and demonstrate it on some neuro-science data has been designed.

In the research study [13] by Leskovec et al., a wide range of real graphs has been investigated. They found out that most of these graphs densify with time, with the number of edges increasing super-linearly in proportion to the number of nodes. Based on their investigation it can be also observed that the average distance between nodes frequently reduces with time, despite the fact that such distance parameters (such as O (log n)) should rise slowly as a function of the number of nodes.

2 Problem definition

We are creating an artificial network of AI "Agents", that will be analyzed by our analysis software which will be comprised of both preexisting algorithms and our own additions.

Each Agent has:

A genetic code that makes up it's unique Neural Network. The Agent's Neural Network will dictate its behaviour, which includes: motion, spacial awareness (the Agent's ability to locate itself in the world as well as locating the world's boundaries) , as well as social awareness (the Agent's ability to locate other Agents around it, the 'Neighbours').

When an Agent, Agent A, comes 'in contact' or is within range of another Agent, Agent B, a random piece of each Agent's Genetic code will be copied, transferred to the other Agent, and placed in a random position in the other Agent's Genetic Code.

Meaning that if Agent A and Agent B come in contact or are within a specific range of the other (will be specified based on the genetic code of each agent, this is their Social Awareness parameter), Agent A will take a random piece of its own Genetic Code and will copy it and give it to Agent B to be placed randomly in its Genetic Code. All the while, Agent B is doing the exact same thing (if Agent A is within its awareness range), Agent B will take a random piece of its own Genetic Code, copy it, and place it randomly in Agent A's Genetic Code.

The analysis run on this Network can tell us a lot of things about the Network and the Agents and how everything changes over time. Some of the information that can be derived from this network have been mentioned below.

- How the Agents are evolving?
- What is the rate of this evolution?
- Is the development of the Agent making them smarter, worse, or more or less alike?
- Is the evolution making the Agents more or less connected to the rest of the Network as a whole?
- Can we predict the potential evolution of the Agents? of the Network as a whole?

2.1 Potential applications

This work helps us understand how AI Agents learn and what they can learn from each other when put in a limited environment, will they over come any deficits and evolve to better things and continue to evolve or will they reach a point of stagnation where nothing new happens? This work will be very helpful in pushing things forward in the ever developing Machine Learning field.

3 Related Works

Our study may indeed be split into two categories: network generation and network analysis. In this regard, we studied several relevant studies in each of these categories individually, which are highlighted below.

3.1 ANN and Community Network Generation:

- Braitenberg's book Vehicles is about machines that show emotion and react to outside stimuli [5]
- Papert's Mindstorms talks about "turtle talk" and the beginnings of "turtle graphics" and teaching programming to children [16]
- The Infranet installation "a generative artwork realized through a population of artificial lifeforms with evolutionary neural networks, thriving upon open geospatial data of the infrastructure of the city as their sustenance and canvas" [2]
- The Evolving Soft Robots article [6] and video talk about and walk through agent evolution by learning from the previous generation of agents while trying to reach the same goal [3]
- A New Evolutionary System for Evolving Artificial Neural Networks paper talks about building evolving artificial neural networks that learn through an analysis software that gives each network a rating based on the network's behaviour then compares it with the other networks and chooses the one with the highest rating. [21]
- Artificial Neural Networks paper talks about the possibility of having ANNs solve problems that they are given, the current state of artificial neural network technology and the future of ANNs given how societies use technology now. [10]

3.2 Network Analysis related works:

Due to the prevalence of networks in the actual world, graph analysis has received more attention in recent years. Researchers have been able to grasp the diverse network systems in a systematic manner by modelling the interactions between elements as graphs. The following four categories can be used to categorise graph analytic tasks and have been illustrated in 2.

- *Node classification*: It is a typical machine learning problem used to classify nodes in graphs. It involves

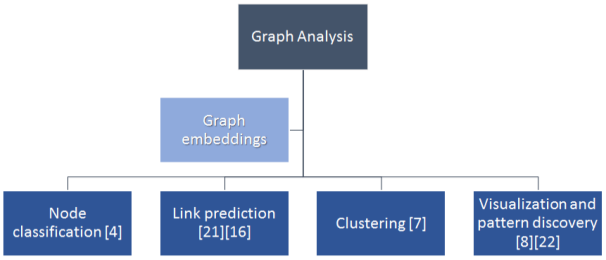

Figure 2. Different categorise of graph analytic tasks

building a model to determine which class a node belongs to. Node Classification models are used to anticipate the existence of a non-existing node characteristic based on the qualities of other nodes [4].

- *Link prediction*: The foundation of recommender systems is link prediction, and typical applications of node embeddings reflect this deep relationship, such as detecting missing friendship linkages in social networks [17], [14].
- *Clustering*: The pairwise similarities between all data items generate a weighted network adjacency matrix that provides all required information for clustering, which is an important application of graph partitioning. Also, node embeddings are a powerful tool for clustering related nodes. On this subject, A novel network partitioning method has been suggested in [7], with an objective function that follows the min-max clustering principle.
- *Visualization and pattern discovery*: The challenge of visualising graphs in a two-dimensional interface has a long history, with applications in data mining, sociology, and biology [8]. Because nodes are translated to real-valued vectors, node embeddings provide a strong new paradigm for graph visualisation: researchers may simply employ current, generic approaches to visualise high-dimensional datasets [18].

Based on the above studies, it can be clearly observed that by applying graph embedding techniques we are able to overcome many graph analysis problems such as link prediction and clustering. Thus in this study we try to work on investigating node properties, graph visualization and investigate its evolution.

4 Methodology

It can be seen that creating and evaluating dynamic networks depicted as graphs, as well as their functionality for extracting relevant information, has attracted a lot of attention, particularly in machine learning and data mining tasks. In this respect, the goal of this project is to present an approach for creating a dynamic network and perform clustering methods for analyzing node behaviours and providing embedding recommendations. Our primary challenge is divided into two sub-tasks: constructing a graph and extracting important information for general machine learning tasks, which are discussed further below.

4.1 Agent Creation Steps Overview

In this step, we aim to create a dynamic network modeled as a graph in NodeJs as follows:

- Creating agents with a simple genetic code and simple neural network that controls behaviour and spatial awareness (of environment and other agents).
- Building a community of 300-500 agents
- Keeping track of all the agents (and their properties) in the system
- Having all the information being outputted at regular intervals to be read and analyzed by the analysis software

4.2 Finalized Agent Creation Methodology

4.2.1 Finalized Agent Network Creation Idea. A lot of papers talk about artificial neural networks (ANNs) learning and evolving by having evaluations run on each individual ANN and then choosing the fittest, based on the evaluation (survival of the fittest methodology) similar to an article titled: "A new evolutionary system for evolving artificial neural networks" from 1997. But no articles (based on our knowledge) talk about the ANN's being connected to agents that exchange a piece of their genotype (DNA) and having it alter the agent's phenotype (behaviours) through altering the neural network (decided by the genotype). This paper talks about just that; in other words, Evolving Artificial Neural Networks in Agent Communities (Artificial Information Networks). We plan to do this by modelling an information network using AI theories and studying the outcomes. A Large system of Agents whose behaviour (motion patterns) and awareness, social (ability to see, recognize, and interact with other agents) and spatial (ability to recognize its own position, world boundaries, and positions of other agents), are defined by their neural network which is described in their genetic code. When each Agent senses another Agent's presence in its "field of view" they connect and give them a random part of their DNA (which alters the recipient Agent's DNA and thus its behaviour) and so on. The last step will be to run an analysis on the whole network to evaluate the agents' behaviours and evolution. There are no specified expectations for results. But the significance of this project is fairly high in terms of where this can go next.

4.2.2 Finalized Network Creation Steps. Steps overview in Codepen.io (a web JS/HTML/CSS processor) rather than NodeJs since it doesn't provide the same functionality and libraries as the online compiler (even though it does provide better scalability).

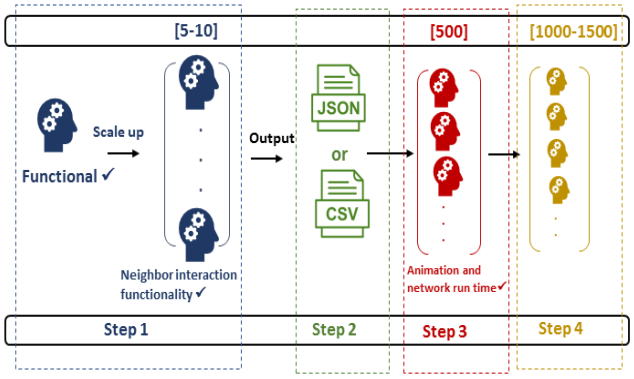

Figure 3. Network Creation Steps

- Creating agents with a simple genetic code and simple neural network that controls behaviour and spatial awareness (of environment and other agents).
- Building a community of 300-500 agents
- Keeping track of all the agents (and their properties) in the system
- Having all the information being outputted at regular intervals to be read and analyzed by the analysis software

Single Agent Creation Steps:

- put together an agent that has a genetic code (genotype) that creates a neural network that decided the agent's behaviour (phenotype)
- agents move around randomly based on phenotype, they detect other agents around them and stop for 3 seconds to "meet" and build a tether that decays over time.
- try to code in genotype exchange during "meeting"

Network Creation Steps: The network creation steps have been demonstrated below and also have been visualized in Figure 3.

- Step 1: When 1 agent is "functional", scale up to 5 then 10 agents to make sure all neighbour interaction functionality is working and that the agent information is being stored correctly
- Step 2: Check if output to .json file or .csv file is working and all information aligns and gets updated. and create a timeout function that re-downloads the file with updated information.
- Step 3: Scale up the whole network to 500 agents and check if there is an effect on animation and network run time.
- Step 4: Finally, if Step 3 works with no issues, scale up further to 1000-1500 agents as web compiler allows.

4.2.3 Borrowed Code Fragments. come from Code built in Prof. Graham Wakefield's course DATT4950/DIGM5950 using the AliceLab library.

- Agents 1: chemotaxis with 2 nostrils - taking environment code fragments.
- Agents 2: random walk, avoidance and cohesion - taking movement fragments
- Agents 3: continuation of flocking - taking neighbour detection, tether creation, and advanced movement code fragments.
- Breeding Biomorphs - taking genotype, phenotype, and gene echange (genetic evolution) code fragments.
- Neuro-Evolution: First Neural Network - and reusable library - taking basic evolving neural network code fragments
- Neuro-Evolution: Tag Game - taking neighbour interaction, node deletion, neuro-evolution code fragments

4.3 Network Analysis Steps Overview

Based on the data set obtained in 4.1 from our proposed complicated network, The following are the primary analytical methodologies that we intend to investigate:

- Visualizing graphs for different time steps and investigate graph evolution
- Running basic network analysis (extracting graph information such number of nodes and edges, in and out degree distribution)
- Node and Edge Importance in Graphs
  - lists of Similar nodes based on: position, genotype, and eye colour
  - PageRank Scores, Edge Betweenness;
- Investigating genetic convergence

It should be noted that some algorithms work only for undirected graphs and others are not well defined for directed graphs. Thus in some cases we had to consider treating a directed graph as undirected for some measurements.

4.3.1 Updated Network Analysis Steps Overview. We aim to use the data set from 4.2 in .json or .csv file formats and use network analysis techniques such as:

- getting graph network measurements: node degree distribution, local and global clustering coefficients, graph diameter (based on the graph snapshots from the timed outputs of the code)
- node importance (based on how many connections and exchanges the agent has made with others)
- link and community prediction of the the agents (based on information provided in the current snapshot compared to the next snapshot of information as this is a dynamic network.)
- community detection (based on similarity of genotype and phenotype in the agents)

to analyze and get more information from the network.

This gives us an insight into the agents' behaviour evolution and the way that the evolution trends. This is very

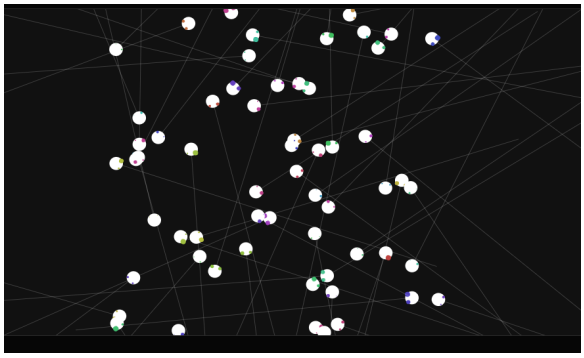

Figure 4. Visualization of generated network evolution

important because it helps us understand better how the network behaves based on the algorithm provided.

5 Evaluation

For our implementation we used JavaScript in Codepen.io for graph generation. The reason behind choosing Codepen.io and not performing codes on our local system is because we are using libraries that are only supported when running JavaScript using a web compiler like Codepen.io. We attempted using NodeJs and other service to help us host and run the code externally but they did not work as needed and it was outside of our scope given the time frame of the project. Also, we used Python for studying graph evolution in Google Colab environmnet. We have used DyNetx and Networkx library for generating our graph in different time steps. Also, for graph visualization and statistics we used Gephi (an open-source network analysis and visualization software package written in Java on the NetBeans platform).

5.1 Network Generation in Codepen.io

The implementation have been performed in Codepen platform and the animation of agents evolving over the time can be found out in thislink. The generated network has been shown in Figure 4.

As can be seen in this figure, each white circle is an agent where the eyes are just another attribute of the agents. We have 300 agents that are moving in the system, and they record everyone that they meet. They move around based on individualized patterns and stop when they meet another agent. Based on their movement they exchanged information that can be defined as genotype.

5.2 Dataset

We have 6 output files from 2 iterations of running our network. Both iterations produced 3 files each for us to run our analysis software on. In each iteration the files are 50 frames apart which is about 2.1 seconds apart if the visualization was run at 24 fps. This gives us an insight on a snapshot of the agents' life and how they evolve over time. The reason we did 2 iterations is because we wanted to see if the simulation would yield the same results when run at different times. This helps us answer are question of whether the agents will always tend towards a specific state or if they will always find a new end state. But given that we were only able to get a small window in the agents' life because of computing limitations we would have to use our limited insight to predict whether the end result would've been different or not based on the results of our genetic convergence analysis.

Table 1. Graph Properties

Time intervals	Graph	# Edges	# Nodes	Diameter
Iteration 1 - Ts 1	*g*	33720	300	2
Iteration 1 - Ts 2	*g1*	55388	300	2
Iteration 1 - Ts 3	*g2*	68742	300	2
Iteration 2 - Ts 1	*g3*	33986	300	2
Iteration 2 - Ts 2	*g4*	68742	300	2
Iteration 2 - Ts 3	*g5*	68708	300	2

5.3 Graph basic information

based on the obtained 6 datasets mentioned earlier, we created six distinct graphs using Networkx Python library in different time intervals. The number of agents (nodes) considered to be fixed and the connections among them (edges) may be varied during all iterations. The detail information about each of these graphs including their graph size and its diameter have been shown in Table 1.

5.4 Graph visualization and statistics

Because the number of edges in our graphs is considerable, Networkx does not offer adequate capabilities for visualizing the graph. Thus we decided to use Gephi which is a free and open-source application for viewing and analysing massive network graphs. Gephi employs a 3D render engine to display graphs in real-time and accelerate exploration. It may be used to explore, analyse, filter, clustering, alter, and export any form of graph.

5.4.1 Graph visualizations. We first obtained the list of nodes and edges and converted them into a Pandas data frame exported as csv for each of the graphs g, g1, g2, g3, g4 and g5 respectively. Then we used Gephi platform and visualized our graphs based on in-degree ranking nodes approach. The layout of these visuaizations have been set to The Fruchterman-Reingold. The Fruchterman-Reingold layout is a force-directed layout algorithm which treats edges like springs that move vertexes closer or further from each other in an attempt to find an equilibrium that minimizes the energy of the system. The visualization for each of the graphs g, g1, g2, g3, g4 and g5 have been illustrated in Figure. 5, Figure. 6, Figure. 7, Figure. 8, Figure. 9, and Figure. 9 respectively. As can be seen in each of these figures, We picked the hue such that the darker colour of each node symbolises the node's higher in-degree count.

5.4.2 Graph statistics. We have also extracted useful information from the generated graphs that have been reported in Table 3. It can be observed that the amount of diameter and radius are equal to 2 for all graphs. Each of these graphs have the same number of

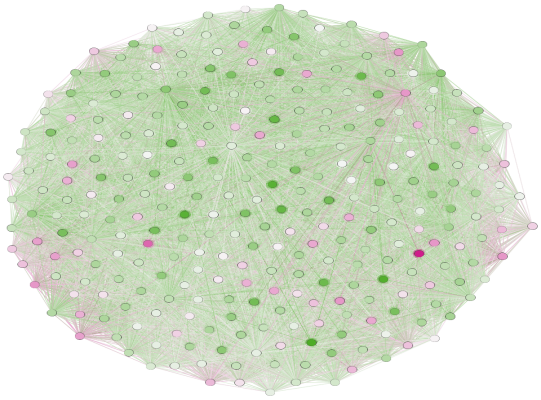

Figure 5. Visualization graph g based on its in-degree rank

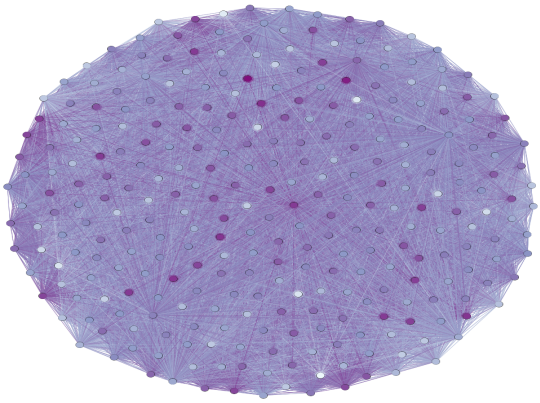

Figure 6. Visualization graph g1 based on its in-degree rank

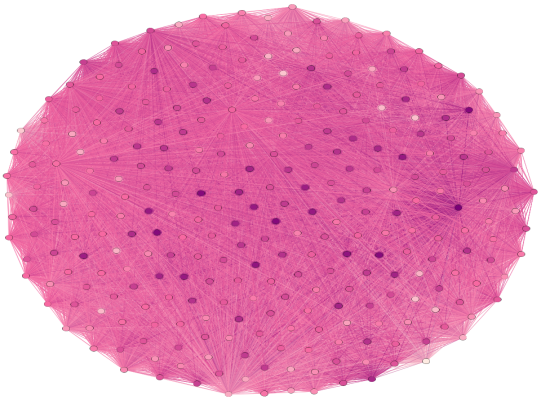

Figure 7. Visualization graph g2 based on its in-degree rank

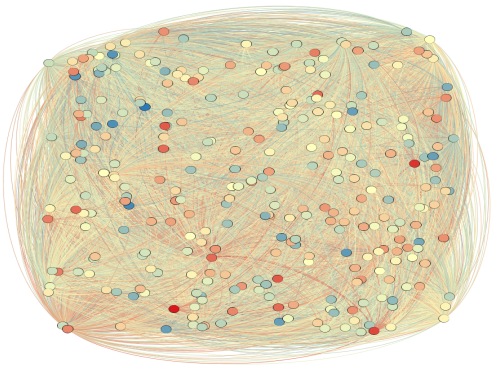

Figure 8. Visualization graph g3 based on its in-degree rank

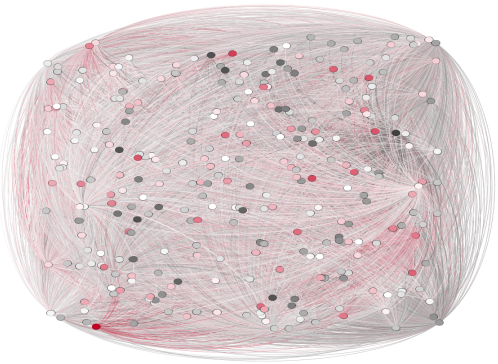

Figure 9. Visualization graph g4 based on its in-degree rank

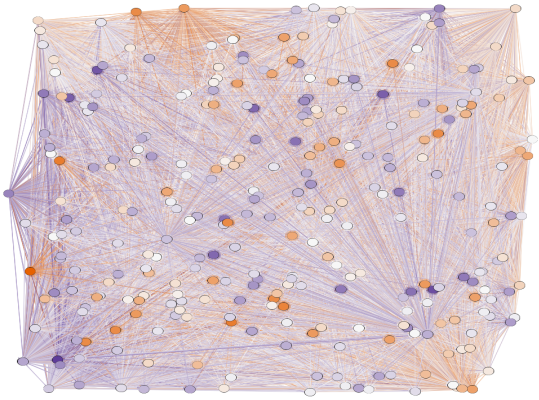

Figure 10. Visualization graph g5 based on its in-degree rank

weakly and strongly connected components that is 1. The Average Clustering Coefficient is the mean value of individual coefficients that have been also reported in this table.

We plotted the node degree distribution of the generated graphs based on the fraction of nodes vs node degrees histogram as shown in Figure 11, 12, 13, 14, 15, 16 for g, g1, g2, g3, g4 and g5 respectively.

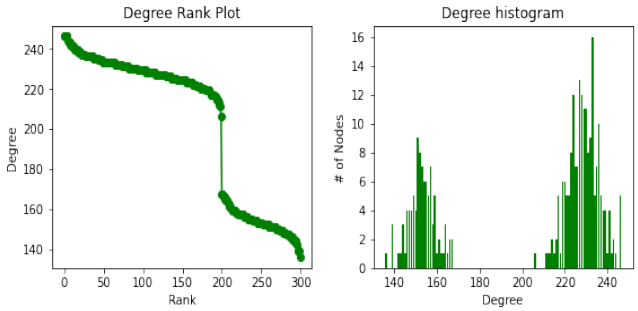

Figure 11. node degree distribution of graph g

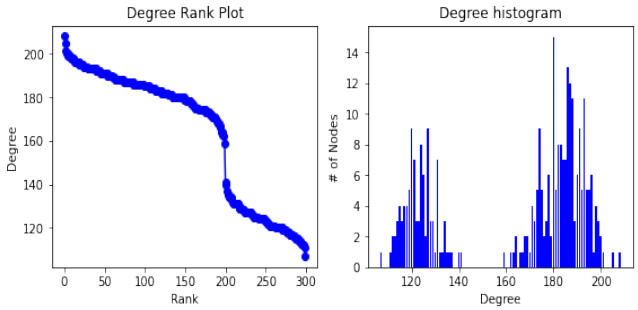

Figure 12. node degree distribution of graph g1

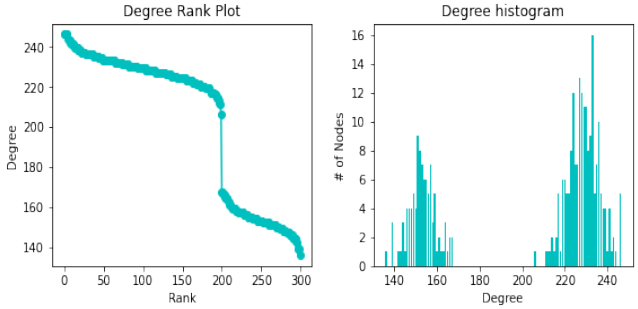

Figure 13. node degree distribution of graph g2

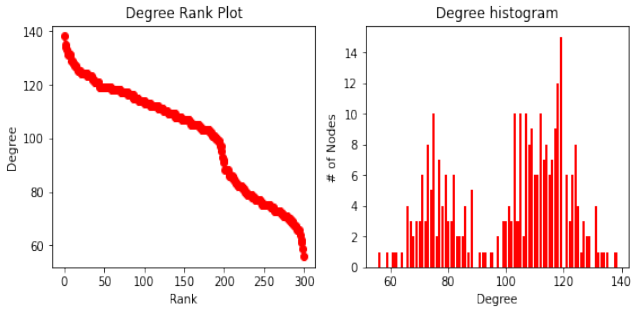

Figure 14. node degree distribution of graph g3

Table 2. Graph statistics results

	Avg Degree	Radius	Average Path length	Density	Average Clustering Coefficient
g	112.400	2	1.6240802	0.376	0.379
g1	184.627	2	1.382519	0.617	0.618
g2	229.140	2	1.233645	0.766	0.767
g3	113.287	2	1.621114	0.379	0.383
g4	229.140	2	1.233645	0.766	0.767
g5	229.027	2	1.234024	0.766	0.766

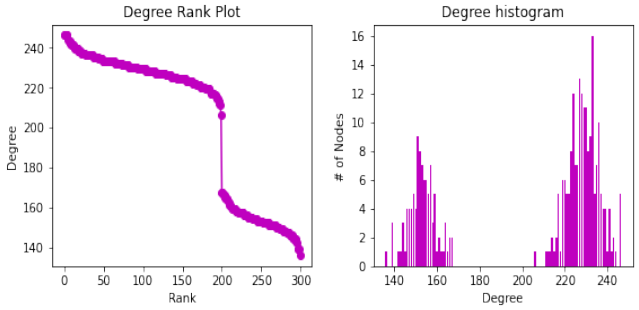

Figure 15. node degree distribution of graph g4

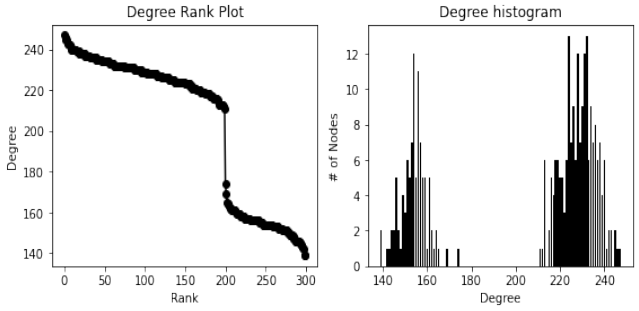

Figure 16. node degree distribution of graph g5

In addition, we performed some analysis such as betweenness centrality distribution, closeness centrality distribution and degree distribution for each the mentioned graphs. However, due to the page number restrictions we only reported the results for the first timestamp of the first iteration graph which is graph g. The betweennness centrality distribution of graph g has been shown in Figure 17 and its closeness centrality distribution has been shown in Figure 18.

5.5 Node and Edge Importance in Graphs

5.5.1 PageRank scores. PageRank calculates the importance of a webpage by mea- suring the quantity and quality of links to that page. The fundamental idea is that more important websites are more likely to be linked to by other websites. It should be noted that the greater a link's PageRank, the more reliable it is. Thus, we used *Networkx* library ("nx.pagerank" command). We considered the inputs of this algorithm to be the generated graphs (g, g1, g2,

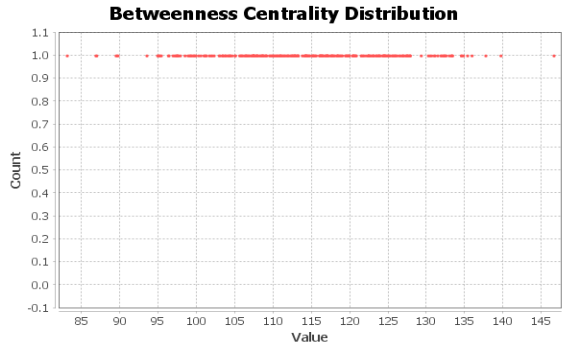

Figure 17. Betweenness centrality distribution of graph g

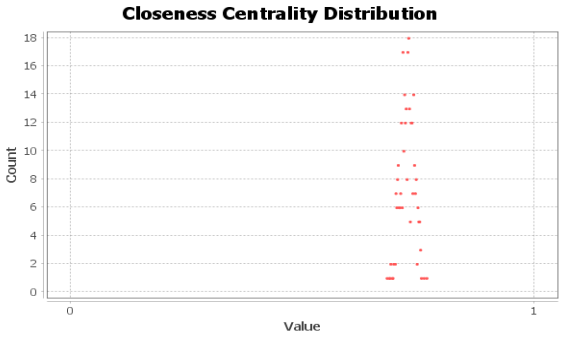

Figure 18. Closeness centrality distribution of graph g

g3, g4 and g5) and alpha = 0.85. Then we sorted our results based on the pagerank scores and reporte the top ten nodes and their scores in Table 3.

5.6 Genetic convergence

In order to investigate the genetic convergence we focused on employing string analysis (string similarity) in Python. We took the output from the .json files with all the agent information and we extracted the genetic code (a list of characters) for each agent into a list. Then we ran through each list and put the individual characters into a string. We then ran a cosine string similarity function that gave each pair of genetic codes (from 2 different agents) a similarity score. We took the similarity scores for each timestamp in each iteration and created frequency histograms. These histograms tell us how often a similarity score shows up which tells us what percentage of agents are similar. Across the timestamps within each iteration we get to see how the agents become either more or less similar genetically.

5.6.1 Iteration 1. As shown in Figure 19 it becomes clear that over time, as captured by the data from each Timestamp in Iteration 1, the agents begin with genetic codes that are relatively similar. Over time, as the agents exchange information and evolve, their genetic codes change to become more and more different from each other. If we were to run this iteration of the simulation for longer and continue to analyze the data, we expect the agents to reach a state where all their genetic codes become even more unique and extremely dissimilar.

Table 3. Pagerank scores of g, g1, g2, g3, g4 and g5 graphs

Iteration	Ts	Graph	Agent	Score
			agent102	0.003942
			agent42	0.003941
			agent166	0.003941
			agent81	0.003938
	Ts 1	g	agent30	0.003934
			agent102	0.004112
			agent58	0.004056
			agent114	0.003982
			agent42	0.003978
	Ts2	g1	agent81	0.003973
			agent102	0.003942
			agent42	0.003941
			agent166	0.003941
			agent81	0.003938
Iteration 1	*Ts 3*	g2	agent30	0.003934
			agent63	0.004384
			agent55	0.004301
			agent2	0.004273
			agent10	0.004246
	Ts 1	g3	agent36	0.004215
			agent102	0.003942
			agent42	0.003941
			agent166	0.003941
			agent81	0.003938
	Ts2	g4	agent30	0.003934
			agent161	0.003950
			agent102	0.003935
			agent189	0.003924
			agent89	0.003922
Iteration 2	*Ts 3*	g5	agent159	0.003894

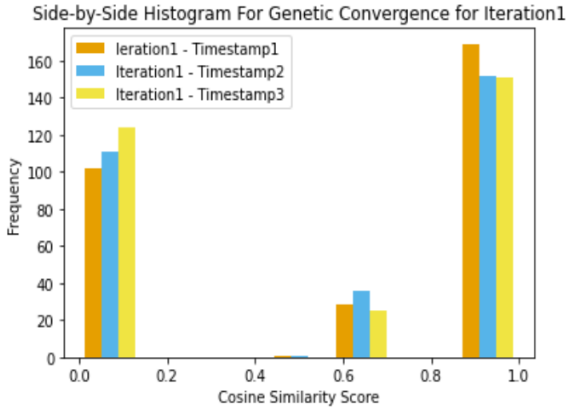

Figure 19. Agents Genetic Convergence for Iteration 1

5.6.2 Iteration 2. As shown in Figure 19 over time, as captured by the data from each Timestamp in Iteration 2, the agents begin with genetic codes that are extremely dissimilar and all their similarity scores equal 0. Meaning that no pair of agents have any similarities in their genetic codes. Two seconds later in Timestamp2, however, the agents genetic codes converge due to their interaction. Another

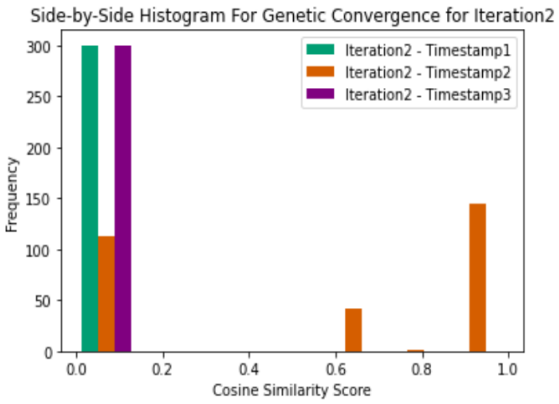

Figure 20. Agents Genetic Convergence for Iteration 1

2 seconds later the agents revert back to not having any similar genetic code with any of their neighbours by Timestamp3. Over time, as the agents exchange information and evolve, we expect their genetic codes to continue this pattern of becoming slightly similar to extremely dissimilar. This makes predicting an end state for this iteration extremely difficult as there does not seem to be a trend either towards or away from convergence given the information analyzed at this time.

5.6.3 Iteration 1 vs. Iteration 2. As expected each iteration yields completely different results and thus will have endlessly unique final states. Having all the different moving parts that make each neural network and agent unique directly translates into the uniqueness of each iteration of the network simulation. Therefore we can always use this same network simulation as a basis for many different analysis that we can run because no matter how many times we rerun the network, the likelihood that we will end up with the same results is slim to none. The more this network evolves the more interesting it gets. Having the ability to add more complexity to the network simulation also provides us with many opportunities to continually evolve the network itself and not just the agents in it. The iterations analyzed above has proven that and given more time and resources we would have run more simulations for longer periods of time to get more information on the evolution of each node in the network as well as, the evolution of the network itself.

6 Conclusion

This project's purpose is to create a technique for assessing synthetic weighted, directed, dynamic graphs created by AI simulations. The first task (graph generation) has been implemented in Codepen.io and the second task (graph analysis) has been performed in google colab environment.

We can describe our program as a success if:

- We get a fully functional Agent network that interacts within itself and it animates all functionality clearly
- We are able to scale our network up to 1000-1500 agents.
- We are able to get all the needed information out in an appropriate file format to be evaluated by the network analysis software.
- We are able to run all the appropriate analyses on the network information and get clear and readable results that make sense (criteria not decided yet) for the kind of network that we have created.

6.1 Task 1: Network Generation Conclusion and Results

We have created a network of 300 agents that each have their own genetic code that builds their neural network, all of which, determines each agent's behaviour. Each agent has a set of information attached to it like it's name, position, velocity, eye colour, memory (list of all agents it has interacted with), genotype, network information, etc. All this information helps us identify the agents and run many kinds of analyses on the network as a whole. As the agents interact with each other and exchange information (genetic code), they evolve and with it the network evolves too. We receive the network information in the form of .JSON files of all the information stored inside each agent at different points in the simulation (which are specified by an interval of frames between the downloads). We chose to use frames as our unit of intervals since, depending on computational specifications, the number of frames per second varies. This ensures that we get the output as regular intervals in the network's life cycle.

6.2 Task 2: Network Analysis Conclusion and Results

We performed many analysis on our generated graphs from our synthetic dataset. We used Gephi platform to visualize our graphs since the Networkx library tool for visualization of such a large amount of edges was not robust enough. We plotted each of our graphs with respect to their time intervals and ranked their color based on their in-degree quantity. In addition, we performed many network analysis methods such as centrality algorithms and node importance approaches. Also, we investigated the convergence of our Genetic patterns and study agents genetic changes similarity using string similarity technique.

7 Future Directions

It was a really difficult process to build the network and then collect the synthetic dataset. Thus, we spent most of our time focusing on network construction and investigate its evolution. However, we would like to extend our work in future by enhancing our graph analysis and experiments, extending our dynamic network (obtaining the maximum number of nodes with respect to our system capabilities) and adding weights to our graph based on the number of times that these agents meet and have interaction with each other.

8 Code repository

Link to our Google Drive with Google Colab files of all our code (also submitted as a .zip file): Network analysis. Also, Link to our network evolution and synthitic data construction implmentations can be found in this link.

References

[1] [n.d.]. *Introduction and Role of Artificial Neural Networks.* Chapter Chapter 1, 1–3. https://doi.org/10.1142/9789811201233_0001 arXiv:https://www.worldscientific.com/doi/pdf/10.1142/9789811201233_0001

[2] 2007. https://artificialnature.net/

[3] 2013. *Evolving Soft Robots with Multiple Materials (muscle, bone, etc.).* Evolving AI Lab. https://www.youtube.com/watch?v=z9ptOeByLA4&ab_channel=EvolvingAILab

[4] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. Springer, 115–148.

[5] Valentino Braitenberg. 1986. *Vehicles: experiments in synthetic psychology*. MIT Press.

[6] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. 2013. *Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding*. http://jeffclune.com/publications/2013_Softbots_GECCO.pdf

[7] C.H.Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and H.D. Simon. 2001. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings 2001 IEEE International Conference on Data Mining*. 107–114. https://doi.org/10.1109/ICDM.2001.989507

[8] M.C. Ferreira de Oliveira and H. Levkowitz. 2003. From visual data exploration to visual data mining: a survey. *IEEE Transactions on Visualization and Computer Graphics* 9, 3 (2003), 378–394. https://doi.org/10.1109/TVCG.2003.1207445

[9] Peter Grindrod and Desmond J. Higham. 2010. Evolving graphs: dynamical models, inverse problems and propagation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 466, 2115 (2010), 753–770. https://doi.org/10.1098/rspa.2009.0456 arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.2009.0456

[10] J.J. Hopfield. 1988. Artificial neural networks. *IEEE Circuits and Devices Magazine* 4, 5 (1988), 3–10. https://doi.org/10.1109/101.8118

[11] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. 1996. Artificial neural networks: a tutorial. *Computer* 29, 3 (1996), 31–44. https://doi.org/10.1109/2.485891

[12] Alexy Khrabrov and George Cybenko. 2010. Discovering Influence in Communication Networks Using Dynamic Graph Analysis. In *2010 IEEE Second International Conference on Social Computing*. 288–294. https://doi.org/10.1109/SocialCom.2010.48

[13] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (Chicago, Illinois, USA) *(KDD '05)*. Association for Computing Machinery, New York, NY, USA, 177–187. https://doi.org/10.1145/1081870.1081893

[14] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. journal of the Association for Information Science and Technology (2007). *Google Scholar Google Scholar Digital Library Digital Library* (2007).

[15] Robert A Meyers. 2002. *Encyclopedia of physical science and technology*. Academic.

[16] Seymour Papert. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc.

[17] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE. *Proceedings of the 24th International Conference on World Wide Web* (May 2015). https://doi.org/10.1145/2736277.2741093

[18] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[19] X. Yao and Y. Liu. 1997. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks* 8, 3 (1997), 694–713. https://doi.org/10.1109/72.572107

[20] X. Yao and Y. Liu. 1997. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks* 8, 3 (1997), 694–713. https://doi.org/10.1109/72.572107

[21] X. Yao and Y. Liu. 1997. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks* 8, 3 (1997), 694–713. https://doi.org/10.1109/72.572107

[22] Edward N. Zalta. 2017. The Stanford Encyclopedia of Philosophy. *IEEE Transactions on Neural Networks* 3 (2017), Stanford, CA 94305–4115.