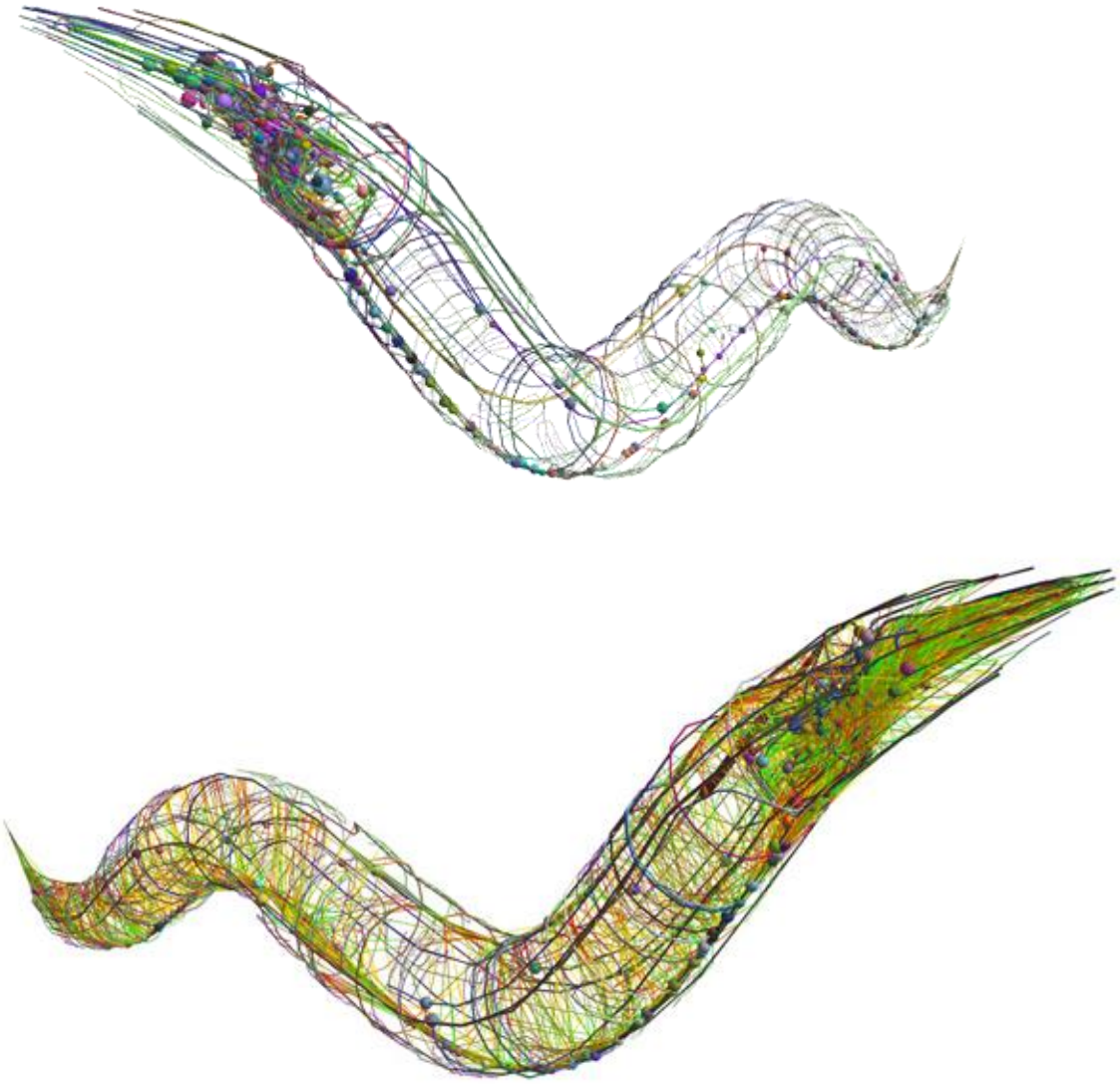

A CONNECTOME

BABAK REZAI, CAROLINE LANCASTER, VANESSA ULLOA, DANIEL CADWELL

Dr. Eric Tao | CST 499: Capstone | CSUMB | June 17, 2016

Executive Summary

Neural networks are modeled on the brain. Brain cells, or neurons, are connected to one another by complex relationships, which is the state of one neuron influenced by the state of many other neurons. As an organism learns through experience, the influence of the various neural connections changes. The benefits of this learning and memory, recorded in relations between neurons, allow the animal or person to use past experiences or predict future events. With more experience, the quality of the prediction improves. Neural Networks seek to combine input information in a flexible way that captures complex relationships among input and output. Most math modeling techniques assume a straightforward relation between input and output. Multiple linear regression, for example, assumes that a straight-line equation adequately links independent variable outputs with an independent variable inputs. Neural networks do not use a single equation to make predictions or explain output. Instead, they use a web of interconnected nodes, similar to the neurons in the brain.

The network arranges the nodes in a series of layers. The output of each layer becomes the input of the next layer. In the first pass through the data, known inputs are commanded to known outputs. For example, a business analyst might enter several accounting values as input compare with stock prices output. Because the model has not yet been tuned, the model output will not match the known real-world output the model assigns a portion of the error to each link in the network and adjust all of the weight using a complex algorithm. The model has now changed based on the data. That is, the network learns from its experience. A user then run the same data through the model again, further changing and refining the weights describing links between notes. This process is repeated until the new weights no longer changed significantly, from when the error becomes acceptably smaller until the software reaches its threshold on the

number of iterations. The result is a model with predictive power. The user can enter new inputs and should be able to predict a new, unknown output.

At the start of this project, our goal was to further our understanding of machine learning and how living organisms think and learn. We then specifically broke this into four sub-objectives, which were as follows:

1. Assess the current behavior of connectome and its similarities and differences to C. elegans.
2. Test the learning capability and limitations of connectome.
3. Analyze patterns that emerge and identify what/if any algorithms can improve Connectome performance and learning capabilities.
4. Test how some algorithms improve/reduce capabilities and alter behavior.

We configured our network so that we could all talk to the cluster from our individual locations. That being accomplished we came upon the first challenge; we had to learn MPICH, which is a message passing interface used for parallel programing. We started by importing the entire neural network from the CSV file and we were creating dynamically created objects to further assemble into a large graph network of nodes and edges. Each edge is representing the space between each neuron, which is what we know as the synapses. We have some 7000 of them in our cluster.

In order to test the networks, we were using virtual machines running MPI sub processes to transfer messages in a ping pong- like call and response. However, the vast amount of technical material needed to be learned from scratch and then communicated between the team for using the MPICH compared to the time we had to do so was insurmountable. We did,

however, get the software installed and configured and we were able to configure multiple Linux VM's to communicate through MPICH (Hello World C program). In doing this, we have met the general guidelines and accomplishments of our first broad goal of furthering our understanding of machine learning and how organisms think compared to the computer and the implemented neural network. We were unable to meet all four of our sub-goals do to the short amount of time allotted for our research and the unforeseen learning curve of a technology we needed in order to move on to the other three sub-goals.

Table of Contents

Table of Figures and Tables

Introduction

Caenorhabditis elegans is one of the most researched model organisms for biological and neural network development. C. elegans is among one of the simplest organisms having nine-hundred cells and a nervous system, which is comprised of approximately three-hundred static neurons that have been mapped into a connectome, or wiring diagram detailing how each neuron connects to another. A connectome can be described as a map of neural connections whether it is an organism's brain or nervous system. The term is mainly used in scientist's efforts in capturing, mapping and understanding the organization of the neural interactions that exist.

Many companies, public and private, in addition to government organizations are working towards modelling the human brain connectome in attempt to build unparalleled thinking machines, also known as artificial intelligences. Companies, such as Open Worm, are using the wiring diagram for C. elegans as a map to write software and this software they have placed in a Lego robot. Open Worm has one goal in mind, and that goal is to understand the human brain and, in the future, to build a computer model of the worm, C. elegans, with the hope that if Scientists are able to understand the worm's brain, this could lead to monumental discoveries in how diseases like Alzheimer's and Parkinson's originate and progress.

There are many applications of understanding and applying the connectome, or wiring diagram for C. elegans. As mentioned previously, it could lead to further work in building the connectome of a human brain. While the human brain is much more complex, the basic building blocks are essentially the same. The main structure that exists in the nervous system is the neuron. Neurons communicate with one another, depending on their role and function, and different types of neurons include motor neurons (which control movement), sensory neurons

(which collects environmental information such as temperature) and interneurons (neurons that exists between the sensory and motor neurons).

Neurons communicate with other neurons via electrical signals which travel in a space between the neurons called a synapse. That space will also be the home of certain neurotransmitters, or brain chemicals, which can alter the behavior of these signals in the organism. In the case of C. elegans scientists were able to map out all 302 neurons and label them as what type of neuron it was. Unfortunately, there are limitations to what a connectome can help research. There is still no viable way to see how neurons behave in real time or how neurons regulate one another.

Some early research findings were discovered by scientists destroying certain neurons one by one to determine if abilities in C. elegans were retained or lost. In the 1980's a postdoctoral student was able to use this method to accurately map out the specific neural circuits responsible for C. elegans' ability to wriggle backward when poked on the head and move forward when touched on the tail. In future research involving a connectome of a human brain, similar methods can be utilized however the human brain is much more complex as neurons use the flexibility of altering their synaptic strength or signal strength of their connections to allow for memory and learning.

Our research objectives will explore this subject further. We will build the C. elegans connectome in C++ and message passing interface employing the parallelism of the software library and its parallelism. Will the connectome behave like the real worm when stimulated? Can this connectome be trained to solve a machine learning problem? If so, what algorithms can be applied to stimulate neuron growth, and which results in higher accuracy?

This study will help build more of an understanding how the connectome works, as well as if it exhibits behaviors similar to the living worm. Furthermore, this will explore the connectome and its ability to be used in problem solving.

Related Work

Studies involving C. elegans and its use to study biological functions can be traced back to the 1960's and 1970's. In 1974, Sydney Brenner published his paper, The Genetics of Caenorhabditis elegans, in which he makes the argument for and establishes the nematode as a model not only for developmental biology from a genetic level, but for neurobiology as well. Brenner began to assemble the first connectome by hand by tracing electron micrographs of the body cross sections.

Later he began to write software that would attempt to reconstruct the cell morphologies from sequential micrographs. It would not be until 1986 structure of the neural network would be determined for the hermaphrodite by the use of electron micrographs. "The hermaphrodite nervous system has a total complement of 302 neurons, which are arranged in an essentially invariant structure." (PMID: 22462104, DOI: 10.1098). As a eutelic organisms they have a fixed number of cells when they reach maturity, the number is invariant for any particular species.

In our study we will be creating a computational model of the C. elegans connection using C/C++ and Argonne National Labs, message passing interface MPICH. This was developed as open source software in a partnership with the University of Chicago and the United States Government. This software provides synchronized communications between processes in a logical graph-like topology. Additionally, it allows for the combination of partial

results as they become available using gather and reduce operations, and synchronization of nodes through barrier operations. Because by their nature networks are constructs in which many neurons can operate in parallel we have elected to build a computational cluster based on a low priced on commodity the Raspberry Pi 2 developer board.

Goals & Objects

A connectome is meant to map the neural pathways of an actual organism. Our research is intended to focus on how close the behaviors of the software actually are to that a living entity. The goal of this research project is to further our understanding of machine learning and how living organisms think and learn. Specifically, over the next several weeks we hope to accomplish several objectives:

1. Assess the current behavior of connectome and its similarities and differences to C. elegans.
2. Test the learning capability and limitations of connectome.
3. Analyze patterns that emerge and identify what/if any algorithms can improve Connectome's performance and learning capabilities
4. Test how some algorithms improve/reduce capabilities and alter behavior.

Within time frame of this study we should be able to determine if connectome is actually moving towards truly mimicking a living organism. Primarily work will focus on the first, and time permitting we will explore the others. Additional goals will attempt to explore if this can be translated to any immediate use outside of attempting to replicate how a living organism thinks.

- Build eleven computer cluster with Arch Linux ARM

- Configure Network
- Write Software using C/ C++ using Argonne National Lab's MPICH library.
- Test connectome for touch response shown in The Neural Circuit for Touch Sensitivity in Caenorhabditis elegans.

APPROACH

Our approach to this research project took a lot of time and planning when first researching neural networks. We were not sure where it would take us and how we would exactly implement it. We did know the materials we would use: Raspberry Pi 2's and Arch Linux. Originally, there were some parts of the approach that were not there but appended later for better testing and understanding of MPI. The following outlines our approach:

- Research existing code in Python that needs to be changed over to C++ and fully understand the connectomes and how the neural networks work.
- The existing code needs to be adapted from Python to C++. We needed to add vectors and we created a .csv file and importing, instead of the neurons being in the main program itself.
- Test the code in C++ to see if the .csv files work correctly when firing neurons and muscles. This is before implementing MPI.
- Build Virtual Machines from scratch using Arch Linux with a master and six nodes. These virtual machines simulated the raspberry pi's.
- Rewrite the code to implement MPICH and install and configure MPI on the Virtual Machines.
- Execute "Hello World" in MPI and "MPI Ring" to test basic MPI commands and the network connectivity between the nodes.

- Add MPI commands to the c++ code like MPI_Send and MPI_Recv.
- MPI_Comm_size (MPI_COMM_WORLD, &world_size); This gives the total number of nodes on the connectome
- MPI_Comm_rank (MPI_COMM_WORLD, &world_rank); This gives the node a rank and the master node is 0.
- MPI_Get_processor_name (processor_name, &name_len); This gives the processor a name from the host file

Background

A connectome can be thought of as a map, in this case it is a map of neural connections and can also be thought of synonymously with a wiring diagram. In the field of neurobiology, a connection is a mapping of all the neural connections within an organism's nervous system. In most scientific efforts the term connectome is used when capturing, mapping and attempting to understand how neural interactions exist. The study of said connectomes and how neurons, synapses and how they interact within a nervous system is called connectomics and this area of neurobiology. Research involving Caenorhabditis elegans has been a huge success for connectomes and the future of connectomics. Our research will be focusing on the connectome of C. elegans and after recreating the connectome we will pursue the answers to the question of will our connectome behave like a real roundworm when stimulate and can a connectome of C. elegans be trained to solve a machine learning problem.

The roundworm may not be large however the existing connectome, created in the 1970's, has led to discoveries of great significance. For example, in 2002 the Nobel Prize was awarded in the area of Physiology or Medicine for "discoveries concerning genetic regulation of

organ development and programmed cell death." (The Nobel Prize in Physiology or Medicine 2002, n.d.) In 2006 the Nobel Prize was awarded in the same discipline for "discovery of RNA interference – gene silencing by double-stranded RNA." (The Nobel Prize in Physiology or Medicine 2006, 2016) Both of these discoveries were assisted by knowledge and research into C. elegans, however connectomes have more to do with neurobiology than genetics. Mapping a connectome and building a functional connectome is a step forward in understanding how neurons and different type of neurons interact with one another. The OpenWorm Project has already succeeded in putting a "[w]orm's [m]ind in [a] Lego [b]ody" (Black, 2014) by utilizing different algorithms to have the neurons interact. This was done using a robot built using Lego Mindstorms and software can be implanted and utilized on different platforms such as Raspberry Pi's as in our research case.

The ability of being able to model and simulate a whole organism inside a computer has many applications. The research that connectomics extends to has a greater reach than just understanding how neurons interact with one another. Neurons interact and synapses fire in order to produce behaviors and analysis of these behaviors may be able to answers questions in the fields of biology, synthetic biology, and pharmaceuticals and also push forward in the fields of artificial intelligence. The first question in this activity will be to determine if the software implanted on a computer to simulate the organism C. elegans will behave as expected when stimulated. To prove this, we will utilize known data about expected behaviors of C. elegans and which neural pathways are being activated. In the 1980's, a postdoctoral student was able to explain a specific C. elegans behavior using a wiring diagram. The specific neural circuits were discovered and mapped for "the worm's tendency to wriggle backward when poked on the head and to squirm forward when touched on the tail." (Jabr, 2012) Behaviors of C. elegans, such as

this, that are known and have neural circuits already mapped can be used to verify and answer our question.

So far, the connectome has been used to discover and research C. elegans behaviors and activities, but as the OpenWorm project shows "[w]ith the worm's nose neurons replaced by a sonar sensor and the motor neurons running down both sides of the worm replicated on the left and right motors of the Lego bot, the robot could emulate the worm's biological wiring." (Shadboldt, 2015). This brings up another question, however, as the connectome can now respond to stimuli can the connectome be taught to solve a problem? Algorithms can be written, in this case C or C++ using a Raspberry Pi cluster. Studying the scientific possibility opens up many possible applications of connectomes such as: artificial intelligence, synthetic biology, and even studies into mental illnesses. Out of what was mentioned, artificial intelligence can be considered a concern ethically as there are a range of applications which can also be taken advantage of and used unethically.

Ethical Concerns

Artificial Intelligence has existed in our minds due mainly to science fiction. For example, the Terminator series introduced us to Skynet and the intelligence killing machines, but science fiction also gave us a friendlier form of artificial intelligence with Data on Star Trek: The Next Generation whose quest to understand was displayed as innocent curiosity. However, there are many questions that come with advances in the areas of A.I. In Star Trek, Data struggled with the label of "android" and discussion of whether or not he was, in fact, a sentient being. These questions will also appear in our own development and research into A.I.

While it is true that there are small groups, businesses and individuals researching and experimenting with this type of technology, it is the domain of the giant technology companies. Facebook, Google, IBM, and Amazon among many are investing and spending Millions if not Billions of dollars in a race to build the best systems and reach the ultimate goal of a general purpose artificial intelligence, a machine capable of thinking at a level equivalent to that of a human, or perhaps even beyond.

One question that must be asked and reflected upon is a connectome, a mapping of the neurons in the brain of a living animal, when recreated in code if it exhibits similar or identical behaviors to the living entity alive? What if its reset, or switched off?

More machines are also appearing in everyday life, automation is entering the workforce and this causes more and more people to be displaced as their jobs are replaced by a robot. Writing code for a machine to perform the same task over and over again without fatigue or need to take a lunch is, perhaps, economically better. However, as mentioned, this displaces workers and forces them to seek new employment. Perhaps we must reanalyze and perhaps restructure our human societal constructs, perhaps by implementing a universal income for all to ensure a balance among the corporations that own these machines, and regular citizenry.

In the future with a successful mapping of a human brain connectome, would this be a considered a breach of privacy? Although the diagram will just be a road map of an individual's neural pathways, those pathways are that individual’s memories and who they are. As in any field of science, consent is always requested and required for most procedures and activities however many brain disorders rob individuals of that right to make informed decisions. Brain

disorders such as dementia and Alzheimer's break down a person's memory and cognitive abilities, would a connectome look like for that individual?

Hopefully, research into a connectome would yield results and advances in knowledge into such brain disorders and this can lead to advances in medications and treatment options. However as promising as this may be, just as the debate in Parkinson's treatment using Deep Brain Stimulation or DBS, are we doing more harm than good? At one-point lobotomy was a very viable and sought out treatment for many diseases. In hindsight our knowledge has taught us that there are better treatment options, however will treatment's based on connectome research be viewed similarly years from now?

Evaluation

We evaluated the connectome program output by using print statements when each artificial neuron would reach an assigned activation potential threshold and "fire". For our program, we set our threshold to 15. Each touch response circuit would trigger firing of neurons and into to firing the muscle actuator. For our test platform we built Arch Linux virtual machines with identical software configurations to run the MPI libraries and executables on. However, due to difficulties experienced with using this library we moved to using only one virtual machine and executing the program on it. Given the development time was limited for our capstone, we found ourselves pushing to get one virtual machine to work on resolving issues with the MPI library and use the large parallelism speedup that would be available from the many node Raspberry Pi nodes.

The MPI Hello Program was a simple program written in C that could be executed on one Node (the "master" node) and sent across to specified hosts or other nodes. The program output

was very simple and since the C code exists on each machine or node a “Hello World” message executed and displayed the name of that Node as well. We used this code to test very basic MPI functionality and also that the Nodes we created with Virtual Machines all existed on the same network.

The MPI Ring Program is another simple program that we wrote in C++. This program differs from the MPI Hello program in that the code utilizes MPI Send and Receive commands to send an integer variable to other Nodes on the network. Using simple logic, the code checks the rank of the machine it is executing on and sends it to the next ranking machine in the network. For example, Node two will send the integer variable to Node three. At the end of the execution the integer variable will have been passed around in a ring to all the Nodes in the network. This program helped us learn about MPI Send and Receive commands and test whether they would be able to send simple MPI data types across to specified machines or Nodes within the network.

Since the code accepts user input (a Neuron) we entered two different neurons to simulate the Anterior and Posterior touch neurons and track which neurons were firing through the program running. Our research gave us an example to follow with these neurons and our results proved that the C++ was firing the appropriate neurons in the appropriate sequence and muscles per the journal's’ findings. (Chalfie, et al., 1985)

Final Implementation

The idea for the code came from a Python code developed by Timothy Busbice and that was posted on Github. The purpose of the code is run a simulated c. Elegans connectome on a robot using a Raspberry Pi as a processor. We used this code as a guide to develop our program in C++ and kept some of the elements such as user input. The code begins by reading two excel files that contain the connectome data—Neurons that are connected and their weights, and a list of all the Neurons (without duplicates) and default weight of 0. These files are read into two vectors named connectome_vector and postsynaptic_vector and these vectors will be used in later functions for comparisons and storage of the synapse weight. A decision was made to declare the vectors globally since they are used in multiple functions throughout the program, this saves having the pass the vectors by reference across multiple function calls. In order to test that the functions to read the excel files were successful, an additional function called testFiles() was written in order to display all values in each vector and the vector size.

```
void testFiles()  {

    /*****  DISPLAY CONNECTOME VECTOR *****/

    cout << "connectome vector size: " << connectome_vector.size() << endl;

    for(int i = 0; i < connectome_vector.size(); i++) {
        cout << "\t" << i << " : " << connectome_vector[i].get_neuronA() << endl;
        cout << "\t" << i << " : " << connectome_vector[i].get_neuronB() << endl;
        cout << "\t" << i << " : " << connectome_vector[i].get_weight() << endl;
        cout << endl;
    }

    /*****  DISPLAY POSTSYNAPTIC VECTOR *****/

    cout << "postsynaptic vector size: " << postsynaptic_vector.size() << endl;

    for(int i = 0; i < postsynaptic_vector.size(); i++) {
        cout << "\t" << i << " : " << postsynaptic_vector[i].get_neuronA() << endl;
        cout << "\t" << i << " : " << postsynaptic_vector[i].get_weight() << endl;
        cout << endl;
    }

}
```

Figure 1. Code snippet of testFiles() function

The vectors took data from the excel files and parsed it based on comma placements, from the first excel file three different pieces of data were taken from each line. The first piece of data is a Neuron, we labeled it neuronA. The second piece of data is also a Neuron and is connected the first neuron, we labeled this Neuron neuronB. The last piece of data is an integer value that is the weight assigned to that connection, this value we labeled weight. The second excel file was read also by parsing the data by the comma placement, each line had 2 different pieces of data. The first value was a Neuron value that we labeled neuronA, the second value in the file is a zero value. Instead of reading this zero value and assigning it as the weight, we set the code to default the weight to zero knowing the pattern of the given excel file.

In order to maintain and store these values properly a class called synapse was created in order to properly assign and manage the values and also create member functions to process this data throughout the program.

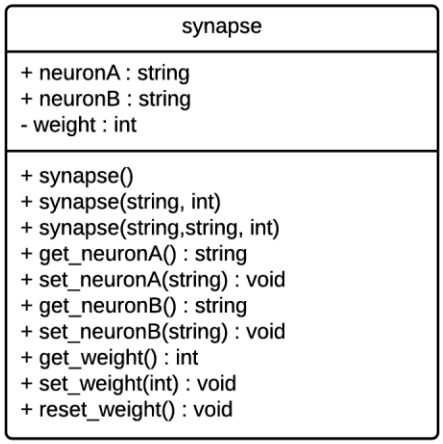

Figure 2. UML Class diagram of the synapse class.

The synapse class has three constructors, two are overloaded constructors. The first overloaded constructor accepts a string value and an integer value and assigns those values to neuronA and weight respectively, the second overloaded constructor assigns two string values and an integer value to neuronA, neuronB and weight. The vectors that were created earlier each hold synapse objects, connectome_vector uses the second overloaded constructor as the objects will each have two Neurons and a weight value while the postsynaptic_vector uses the first overloaded constructor and will hold a Neuron and weight that is default to zero. The synapse class has seven methods which are used to set neuronA, neuronB and weight values and accessors to the variables as well. An additional method was created to reset the weight to zero.

A decision was made to also use file output as part of this program to use for debugging and also to store input and output from the program. A file object was created to write to and is stored in a separate folder as part of the file structure. In order to ensure that each output file is unique the file name consisted of the variable neuron (which holds the value of user input) + Day + Month + Year + _ + Hour + Min + Seconds, in the diagram are some example output files from the program runs.

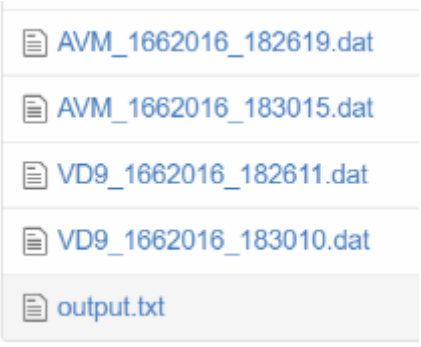

Figure 3. List of filenames for program run outputs

There are three additional functions used in this program:

1. void runconnectome(synapse), which accepts an object of the synapse class and also calls dendriteAccumulate and fireNeuron
2. void dendriteAccumulate(synapse), which accepts an object of the synapse class.
3. void fireNeuron(synapse), which accepts an object of the synapse class.

The function, runconnectome, accepts an object of the synapse type and passes the object to dendriteAccumulate. Afterwards the function loops through the postsynaptic_vector and compares the weight of each synapse object in the vector against a threshold variable with a predetermined integer value. If the condition is true, the synapse object from the vector is passed to the fireNeuron function and the weight is reset.

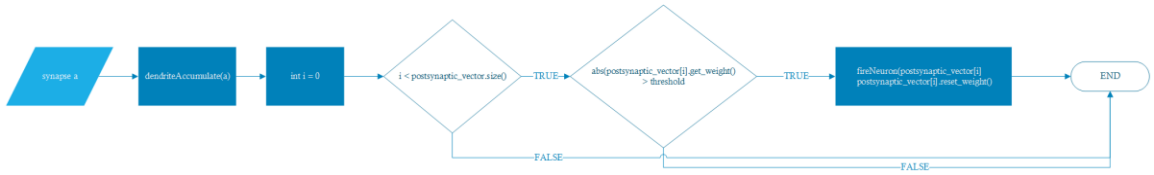

Figure 4. Flow chart of runeconnectome() function

The next function is dendriteAccumulate which accepts a synapse object passed from the runconnectome function. The purpose of this function is to take the synapse object that was passed from runconnectome and compare against the connectome_vector until a match is found for the objects neuronA value. Once a match is found another comparison is initiated to find a match between the neuronA value in the postsynaptic_vector and the neuronB value of the

previously located connectome_vector object. If a match is found the weight of the connectome_vector object is added to the matching postsynaptic_vector object. This function is called at several points in the program to simulate neuron stimulation, the weight is added in order to meet the threshold value.

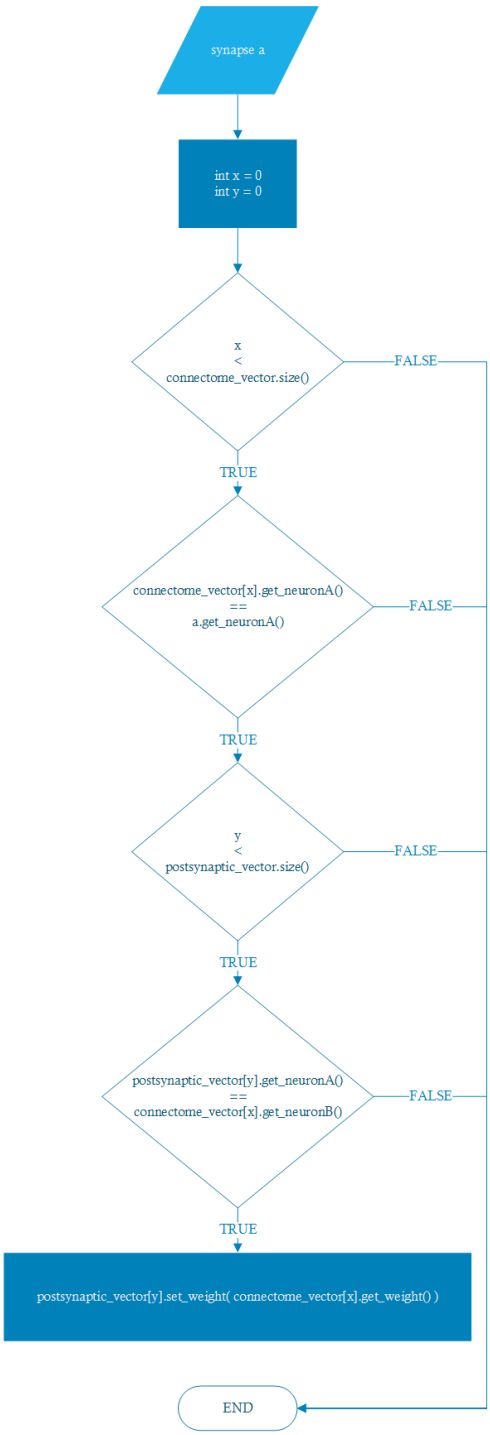

Figure 5. Flow chart of the dendriteAccumulate() function

The last function is fireNeuron which also accepts a synapse object, this function is called with the weight value in the postsynaptic_vector has exceeded the threshold value. In the function the synapse object is once again passed to the dendriteAccumulate function, then there is an additional check to see if the postsynaptic_vector weights have exceeded the threshold value. Lastly, the neuronA value is checked against conditions to determine if this synapse object is a Neuron or a Muscle. If the postsynaptic_vector object has exceeded the threshold and is a Muscle, there is an output message “Fire Muscle”, this output message is also written to the output file, afterwards the weight of the object is reset to zero. If the postsynaptic vector object has exceeded the threshold and is a Neuron, there is an output message “Fire Neuron”, this output message is also written to the output file, afterwards the weight is reset to zero. Within both conditions of the if statement are integer variables muscleFireCount and neuronFireCount, these are incremented with every true condition met and are displayed at the end of the program run for informational purposes. It is important to note that in all comparisons of the vector weight against the threshold the absolute value is compared as some neurons have negative weight values.

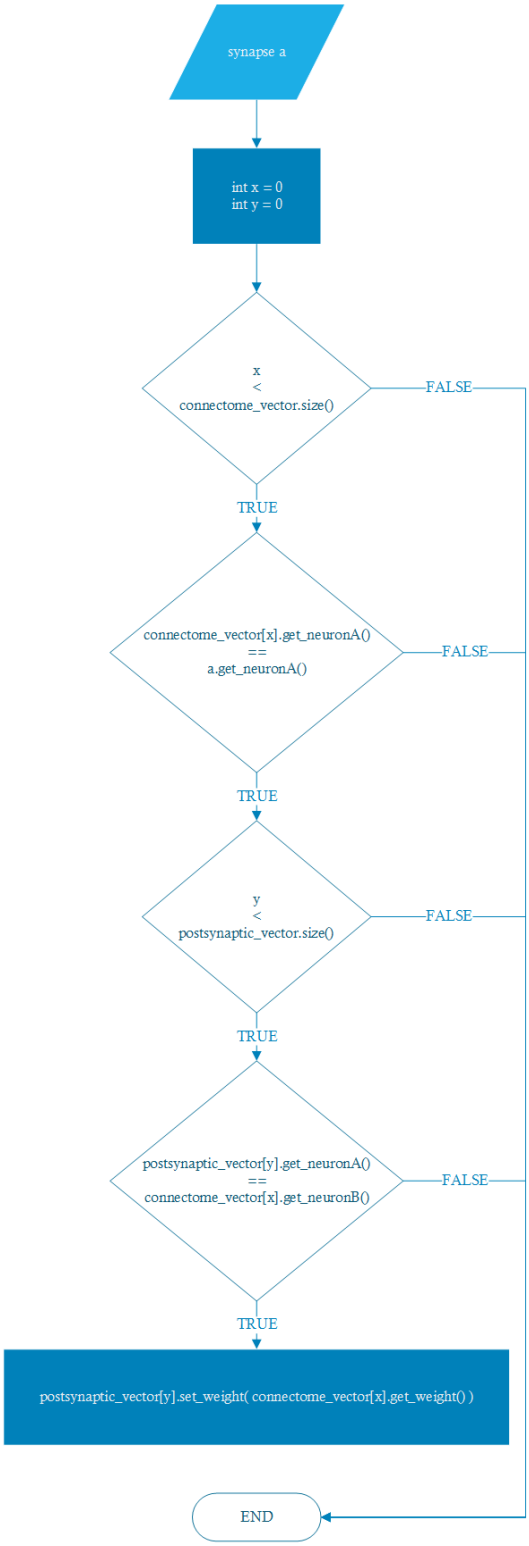

Figure 6. Flowchart of the fireNeuron() function

In the main part of the program there is a loop that takes the user input (neuron) and searches for all instances of that value in the connectome_vector, once all instances have been passed into the runconnectome function and all associated neurons have fired the program ends. At the end of the program run the follow variables are displayed: total neurons fired (neuronFireCount), total muscles fired (muscleFireCount), total run time (clock() value at end of program) in seconds. These values are also written to the output file. While this display is not required for the program to run it is useful to see program run time for different inputs, for example. Two different Neuron inputs can produce radically different run times and results.

For input Neuron “VD9”:

```
----------
Total Neurons Fired: 1
Total Muscles Fired: 12
Total Run Time: 0 seconds
----------
```

Figure 7. Output results from VD9 as user input

For input Neuron “AVM”:

```
----------
Total Neurons Fired: 125403
Total Muscles Fired: 16915
Total Run Time: 68 seconds
----------
```

Figure 8. Output results from AVM as user input

The adaption of an existing code in Python to C++ was difficult as many data types did not easily translate over. For example, a Python collection was converted (in C++) to a vector of objects in order to hold the same data. The functionality of a class object allowed for methods to

be written specific for those objects and a vector was a flexible way to hold the objects without having to define a size prior to reading the excel files. There were also issues attempting to implement multi-threading into C++ in order to have different threads handle separate amounts of data as a precursor to MPI implementation, unfortunately MPI Send and Receive commands are limited by MPI data types which can support simple types such as integers and characters but not necessarily vectors of objects.

Conclusion

MPICH programming had a steep learning curve and we encountered difficulties with our program crashing. However, we were able to implement the connectome without using MPI methods and were able to trigger chains of synapses in the artificial network that had both clear beginning and ending neurons. Our study was able to verify that the activity and our connectome program matches that captured anterior and post touch response circuits in the physical C. Elegans nematode studied in the "The Neural Circuit for Touch Sensitivity in Caenorhabditis elegans'. (Chalfie, et al., 1985)

Complexity of MPI was higher than expected when first researching it. Configuring MPI on our six different nodes, one master and 5 non-masters, took almost a week to configure correctly. The time needed for MPI, we needed another month to complete with its complexity.

While having physical hardware to work with, due to networking limitations it was very helpful to be able and approximate our build with networked virtual machines that mad with easier to develop and build code on identical software setups and work around network proxies when working from different locations remotely.

In the end, we ended up running our code using AVM and PLMR, which are the anterior and posterior touch response circuits and we were able to verify bother the front and back touch responses were activated. Meaning, we essentially “poked” the connectome and it responded from the “poke”. We completed our first goal in the hypothesis and experienced a breakthrough with technology and furthered the understanding to Artificial Intelligence.

References

Black, L. (2014, November 16). *A Worm's Mind in A Lego Body*. Retrieved from I PROGRAMMER: http://www.i-programmer.info/news/105-artificial-intelligence/7985-a-worms-mind-in-a-lego-body.html

Chalfie, M., Sulston, J. E., White, J. G., Southgate, E., Thomson, N. J., & Brenner, S. (1985). The Neural Circuit for Touch Sensitivity in Caenorhabditis elegans. *The Journal of Neuroscience*, 5. Retrieved from http://www.jneurosci.org/content/5/4/956.full.pdf+html

Jabr, F. (2012, October 2). *The Connectome Debate: Is Mapping the Mind of a Worm Worth It?* . Retrieved April 19, 2016, from Scientific American: http://www.scientificamerican.com/article/c-elegans-connectome/

Shadboldt, P. (2015, January 21). *Scientists upload a worm's mind into a Lego robot*. Retrieved from CNN: http://www.cnn.com/2015/01/21/tech/mci-lego-worm/

The Nobel Prize in Physiology or Medicine 2002. (n.d.). Retrieved April 19, 2016, from The Official Web Site of the Nobel Prize: http://www.nobelprize.org/nobel_prizes/medicine/laureates/2002/

The Nobel Prize in Physiology or Medicine 2006. (2016, April 19). Retrieved from The Official Web Site of the Nobel Prize: http://www.nobelprize.org/nobel_prizes/medicine/laureates/2006/

Appendix

DIVISION OF WORK

Table 1

Task	Assignee	Status
Proposal	Daniel K, Daniel C, Babak, Caroline, Vanessa	COMPLETED
Capstone PowerPoint	Caroline, Daniel K, Babak, Vanessa	COMPLETED
Build Pi Cluster	Babak	COMPLETED
Install Arch Linux	Babak	COMPLETED
Programming/ Research (C++)	Babak	COMPLETED
Programming/ Research (C++)	Caroline	COMPLETED
Programming/ Research (C++)	Vanessa	COMPLETED
Commercial	Dan K, Dan C, Babak, Vanessa, Caroline	COMPLETED
Set up Virtual Machines	Caroline, Vanessa, Babak, Dan C	COMPLETED
Final Video Presentation	Caroline, Daniel C	COMPLETED
Testing	Vanessa, Babak	COMPLETED

BUDGET

Table 2

QTY	Description of Item	Total Cost
11	Raspberry Pi 2 Model B Project Board - 1GB RAM	$416.90
11	3ft Cat6 Snagless Ethernet	$19.80
11	Kingston Digital 16GB microSDHC Class 10	$64.02

1	100 Pack M2.5 Nylon Hex 12mm Standoff	$10.49
2	8 ½ x 11 Inch Acrylic Sheet ⅛ inch thickness	$4.00
8	M2.5 Steel Nuts	$2.40
4	M2.5 Steel Screws	$1.20
1	Sabrent USB 2.0 Ethernet Adapter	$14.99
24	Dupont Male to Female 2.54mm Jumper Cable	$3.60
2	Breadboard Power Rail	$2.00
1	MeanWell Switching Power Supply RT-50C	$20.12
1	D-Link 16-Port Gigabit Switch (DGS-1016A)	$67.02
	TOTAL	**$626.54**

GITHUB LINKS

- https://github.com/vulloa/Connectome_Capstone_MPI