Chapter 1

Modularity and Neural Networks

Contributed by Mark Horeni

1.1 Introduction

Community detection is useful unsupervised way to understand more information about a graph. One way to do community detection is by maximizing a global property of the graph known as modularity. Maximizing modularity has been used in a wide variety of applications with some success in not only biological networks, but other social networks and beyond for community detection [5] [6]. Specifically, modulairty maximization techniques have been shown to out perform other community detection algorithms [6].

1.2 The Problem as a Graph

Individual neurons can be thought of as nodes, and each neuron has two types of connectors, either gap junctions or chemical synapses.[7] Chemical synapses as seen in Figure 1.1, can have 1, 2, or 3 directed outputs to another neuron, while similarly, gap junctions can have multiple outputs, but these outputs are undirected as the electrical flow can technically flow either way [7].

1.3 Some Realistic Data Sets

The *c. elegans* is a transparent roundworm that has had all of its neurons mapped along with all of the connections. There are a total of 279 neurons, and between them there are around 6393 chemical synapses and 890 electrical gap junctions[1]. Each neuron has an attribute of whether the neuron itself is either a motor, sensory, or inter neuron (or a combination of), and the distribution of those are roughly equal across neurons [1].
The worm data is the only complete data, but there does exist partial data for other animals including partial data from flies, cats, macaques, mice, rats, and humans.

1.4 Louvain-A Key Graph Kernel

Modularity, Q, is defined as the following [2]

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$

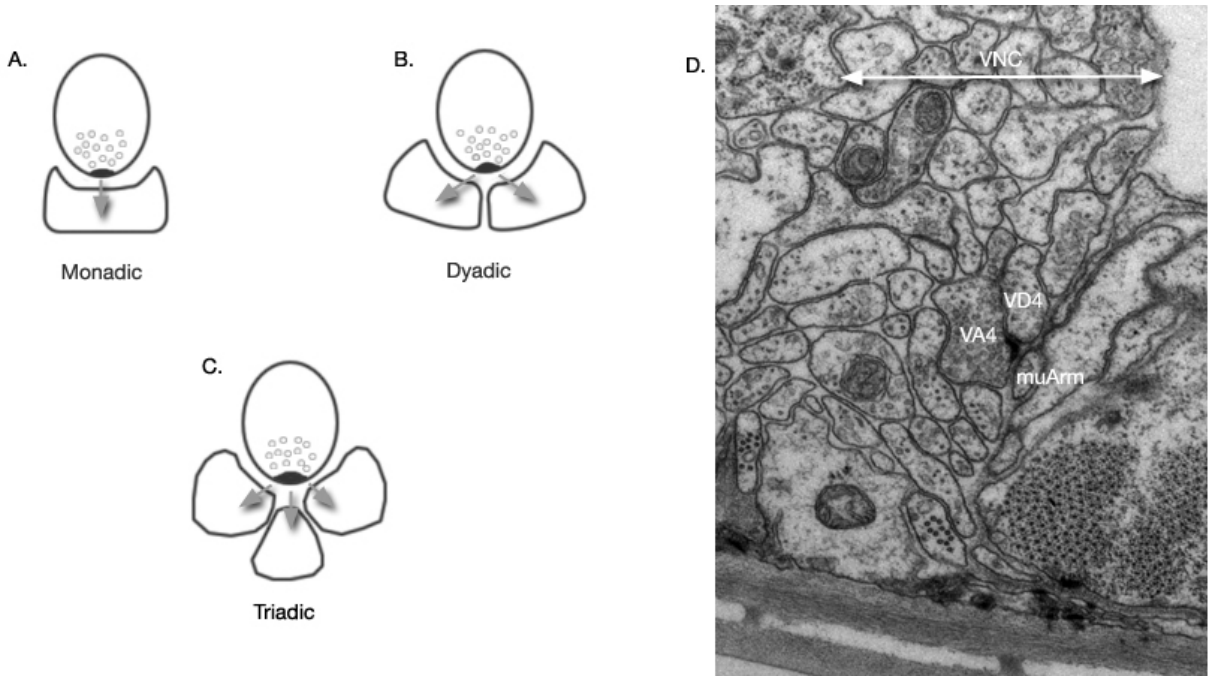

Figure 1.1: A view of the neurons and their chemical synapses

where 2m is the weight of all edges, Aij is the weight between i and j, k_i and k_j are the total weights attached to each i and j, and c_i and c_j are the communities. The goal is to find which combinations of nodes when grouped into certain communities, which combination maximizes modulairty.

1.4.1 Undirected Louvain

Since the goal is to maximize modulairty, the approach of the Louvain algorithm is greedy optimization. To do this, the algorithm first starts with every node in its own community [2]. Next, each node is put into a neighboring community and the change in modularity is calculated by

$$\Delta Q = [\frac{\sum_{in} + k_{i,in}}{2m} - (\frac{\sum_{tot} + k_i}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{k+i}{2m})^2]$$

where $\sum_{in}$ are the total weights inside community C, $\sum_{tot}$ is the sum of edges of the links incident to nodes in C, k_i are the sum of incident links of node i, and $k_{i,in}$ is the sum of weights from i in C with m being the total sum of weights in the network [2].
The other half of the algorithm takes the previous phase, and turns each community into its own node with a self loop with the weight of all the edges of all the nodes inside the community. When this finishes, the process goes back to the first half, and the process is repeated until modularity no longer increases between iterations. This proccess is shown visually in figure 1.2 [4].

1.4.2 Directed Louvain

Although modularity is usually defined for unweighted graphs, in directed graphs it can be defined as

$$Q_d = \frac{1}{m} \sum_{i,j} [A_{ij} - \frac{d_i^{in} d_j^{out}}{m}] \delta(c_i, c_j)$$

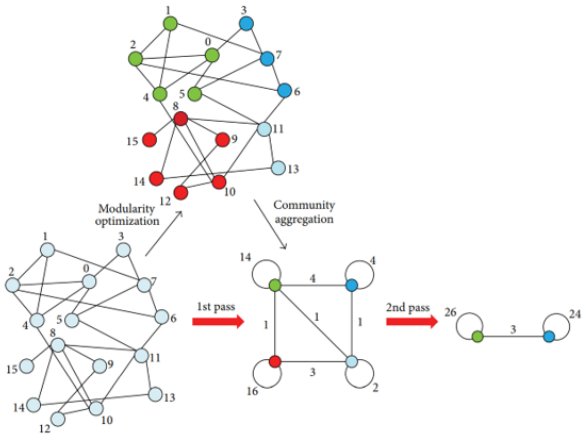

Figure 1.2: Example of the Louvain Algorithm

where the only difference is that m is now the weight of all the arcs (directed edges), and d^{in} stands for the in degree of i while d_j^{out} stands for the out degree of j [3].
Similarly, change in modularity can be defined as

$$\Delta_{Q_d} = \frac{d_i^C}{m} - [\frac{d_i^{out} \sum_{tot}^{in} + d_i^{in} \sum_{in}^{out}}{m^2}$$

where $\sum_{tot}^{in}$ is the sum of all in-going arcs into community C, and $\sum_{tot}^{out}$ are all the out-going arcs out of community C [3].

1.4.3 Psuodcode

The pusdocode of the algorithm that runs with time compleixity $O(n \log n)$ is as follows [5]

1.5 Prior and Related Work

This is space to add in discussion of **prior work** - word on the same problem or kernel that your paper assumes, and **related work** - work on the same application but using different approach or kernel, or a different but similar application..

1.6 A Sequential Algorithm

Discuss here the outlines of a sequential algorithm. What programming paradigms might make the most sense? What are the key data structures? Does the computational complexity differ from that in the Section 1.4?

1.7 A Reference Sequential Implementation

Discuss here your implementation of the basic sequential code. Include what language/paradigm you used for the code.

Algorithm 1 Louvain

V: a set of vertices
E: a set of edges
W: a set of weights of edges, initialized to 1
$G \leftarrow (V, E, W)$
repeat
    $C \leftarrow \{\{v_i\}\} | v_i \in G(V))\}$
    Calculate current modularity Q_{cur}
    $Q_{new} \leftarrow Q_{cur}$
    $Q_{old} \leftarrow Q_{new}$
    repeat
        for $v_i \in V$ **do**
            $Q_{new} \leftarrow Q_{cur}$
            remove v_i from its current community
            $N_{v_i} \leftarrow \{c_k | v_i \in G(V), v_j \in c_k, e_{ij} \in G(E)\}$
            find $c_x \in N_{v_i}$ that has $max \Delta Q_{\{v_i\}, c_x} > 0$
        Calculate new modularity Q_{new}
    until no membership change or $Q_{new} = Q_{cur}$
    $V' \leftarrow \{c_i | c_{\in} C\}$
    $E' \leftarrow \{e_{ij} | \forall e_{ij}$ if $v_i \in C_i, v_j \in C_j$, and $C \not= C_j$
    $W' \leftarrow \{_{ij} | \sum w_{ij}, \forall e_{ij}$ if $v_i \in C_i$ and $v_j \in C_j\}$
until $Q_{new} = Q_{old}$

1.8 Sequential Scaling Results

Discuss here results from your sequential implementation. Include software and hardware configuration, where the input graph data sets came from, and how input data set characteristics were varied. Did the performance as a function of size vary as you predicted?

1.9 An Enhanced Algorithm

Discuss here the outlines of an enhanced algorithm. This could be a parallel code, a code with some significant heuristics, or a code written in a non-traditional programming paradigm. Pseudocode is fine. Discuss what you think is the computational complexity.

1.10 A Reference Enhanced Implementation

Discuss here an implementation of the enhanced algorithm. Include what language/paradigm you used for the code.

1.11 Enhanced Scaling Results

Discuss here results from the enhanced algorithm. Include software and hardware configuration, where the input graph data sets came from, and how input data set characteristics were varied. Ideally plots of performance vs BOTH problem size changes AND hardware resources are desired. Did the performance as a function of size vary as you predicted?

1.12 Conclusion

Summarize your paper. Discuss possible future work and/or other options that may make sense.

1.13 Response to Reviews

This will be included only in the second and third iterations, and will be a summary of what you learned from the reviews you received from the prior pass, and how you modified the paper accordingly.

Bibliography

[1] Beth Li Ju Chen Beckman. Neuronal network of c . elegans : from anatomy to behavior. 2007.

[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008.

[3] Nicolas Dugu and Anthony Perez. Directed louvain : maximizing modularity in directed networks, 2015.

[4] Yong-Hyuk Kim, Sehoon Seo, Yong-Ho Ha, Seongwon Lim, and Yourim Yoon. Two applications of clustering techniques to twitter: Community detection and issue extraction. *Discrete Dynamics in Nature and Society*, 2013:1–8, 2013.

[5] Haewoon Kwak, Yoonchan Choi, Young-Ho Eom, Hawoong Jeong, and Sue Moon. Mining communities in networks: A solution for consistency and its evaluation. pages 301–314, 01 2009.

[6] M. E. J. Newman. Modularity and community structure in networks. *Proc Natl Acad Sci U S A*, 103(23):8577–8582, Jun 2006. 2388[PII].

[7] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 314(1165):1–340, nov 1986.