Community Detection in the C. elegans Connectome

BY MARK HORENI

The Data

280 neurons (humans have 100 billion)

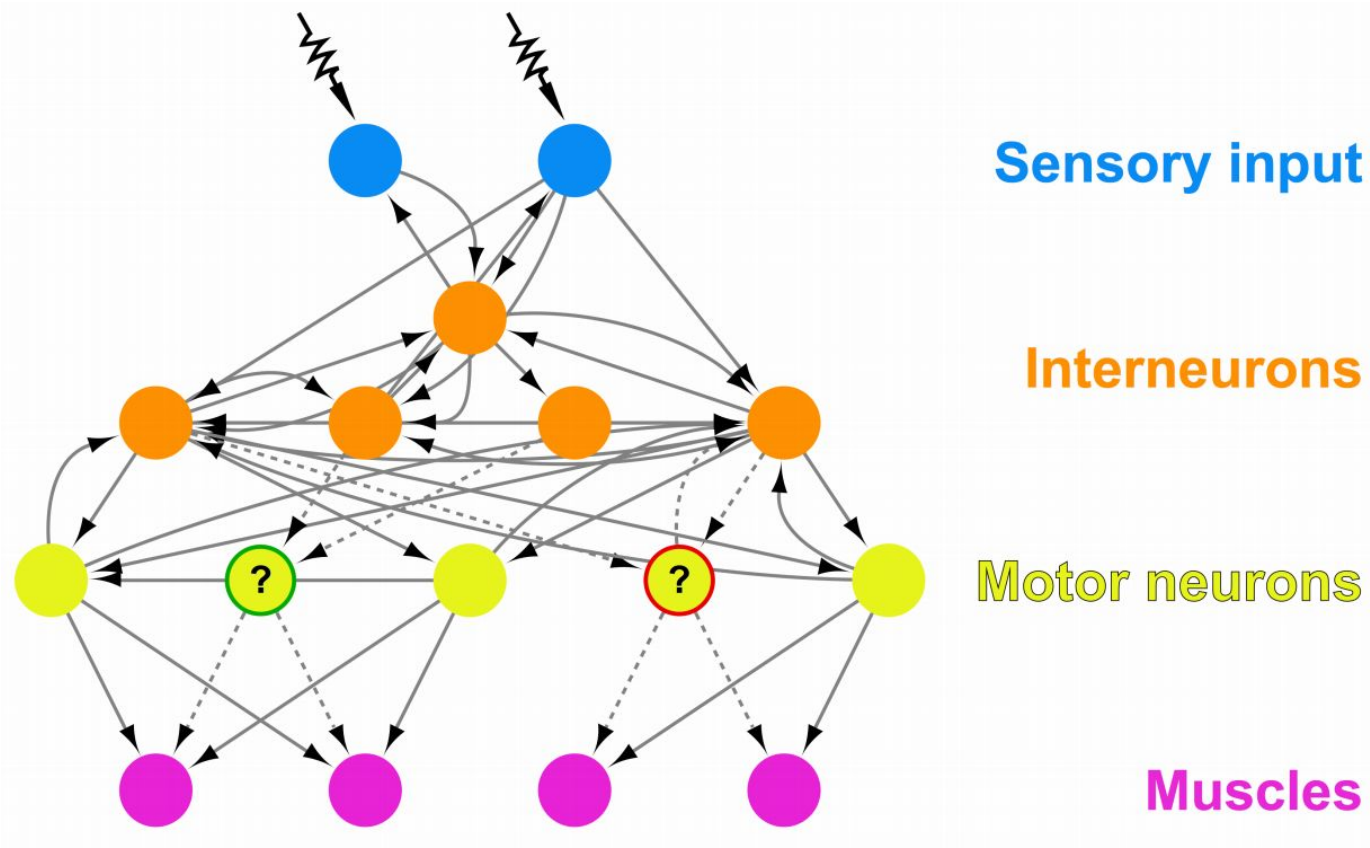
6393 chemical synapses

890 electrical junctions

Neurons can be either Sensory, Inter, Motor, or a combination

Each neuron has a (or multiple) function(s): Touch Chemical, Growth, Locomotion, Feeding, Learning, Light, and Mechanosensensation

Visualization



Modularity

Metric to determine communities

Number between -1 and 1 of how strong communities are

Global Property

Goal: Maximize Modularity

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

Louvain

Each node starts in a community by itself

Put node *i* in a neighboring community

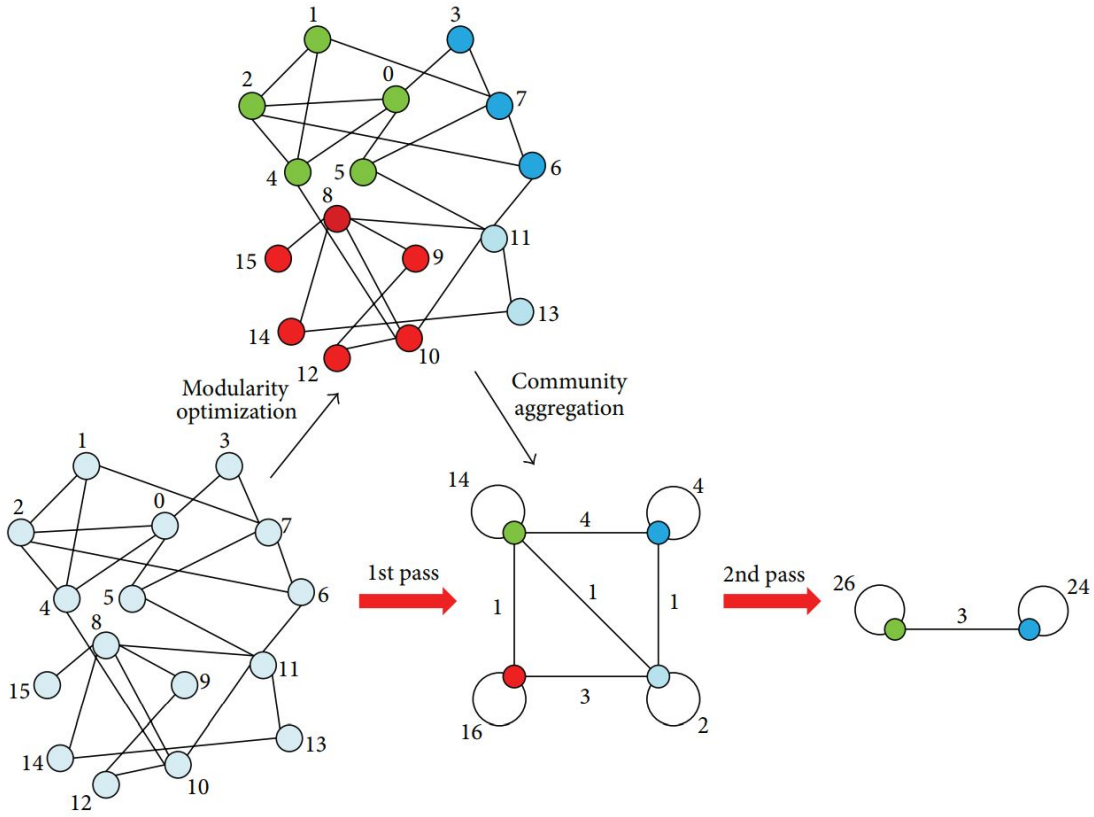
A change in Q is computed: $\Delta Q = \left[\frac{\Sigma_{in} + 2k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$

If the change is positive, *i* becomes part of that community

Once these communities are formed, communities themselves become nodes and intra community weights are treated as self loops, and inter community weights are treated as connections

Repeat

Example


Modularity optimization
Community aggregation
1st pass
2nd pass

1: V: a set of vertices
2: E: a set of edges
3: W: a set of weights of edges, initialized to 1
4: $G \Leftarrow (V, E, W)$
5: **repeat**
6: $C \Leftarrow \{\{v_i\}|v_i \in G(V)\}$
7: calculate current modularity Q_{cur}
8: $Q_{new} \Leftarrow Q_{cur}$
9: $Q_{old} \Leftarrow Q_{new}$
10: **repeat**
11: **for** $v_i \in V$ **do**
12: $Q_{cur} \Leftarrow Q_{new}$
13: remove v_i from its current community
14: $N_{v_i} \Leftarrow \{c_k|v_i \in G(V), v_j \in c_k, e_{ij} \in G(E)\}$
15: find $c_x \in N_{v_i}$ that has $max\Delta Q_{\{v_i\},c_x} > 0$
16: insert v_i into c_x
17: **end for**
18: calculate new modularity Q_{new}
19: **until** no membership change or $Q_{new} = Q_{cur}$
20: $V' \Leftarrow \{c_i|c_i \in C\}$
21: $E' \Leftarrow \{e_{ij}|\forall e_{ij}$ if $v_i \in C_i, v_j \in C_j, and C_i \neq C_j\}$
22: $W' \Leftarrow \{w_{ij}|\sum w_{ij}, \forall e_{ij}$ if $v_i \in C_i$ and $v_j \in C_j\}$
23: $G \Leftarrow (V', E', W')$
24: **until** $Q_{new} = Q_{old}$

Pseudocode

"O(N LOGN)"

Could be $O(N^2)$

Implementation

Python, package included

Changed to be directed

Have to deal with the “resolution limit”

- $$Q_{NL}(t) = (1-t) + \sum_{C \in \mathcal{P}} \sum_{i,j \in C} \left[\frac{A_{ij}}{2m_1} t - \frac{k_i k_j}{(2m_1)^2} \right],$$

Calculated the average Jarccard distance between vector of functions for each community

Results (Resolution = 1)

Number of Nodes In the community	Average Jaccard of Function Vector
38	0.7734204790500803
36	0.6007326004025278
51	0.723304473147914
42	0.6813840153953061
19	0.09090909082644628
42	0.7186147184591744
29	0.3446153845623669
28	0.4953846153084024
53	0.480797

Functions in the Largest Community

Function	Count
Locomotion	28
Growth	5
Chemical	5
Mechanosensensation	4
Touch	4
Feeding	2
Unknown	5

Resolution = 0.75

Number of nodes in the community	Average Jaccard of Function Vectors in a Community
3	0.0
74	0.7364247311085584
12	0.6777777755185185
43	0.7046568624860328
18	0.6190476175736961
28	0.7150793647388513
22	0.5714285711564626

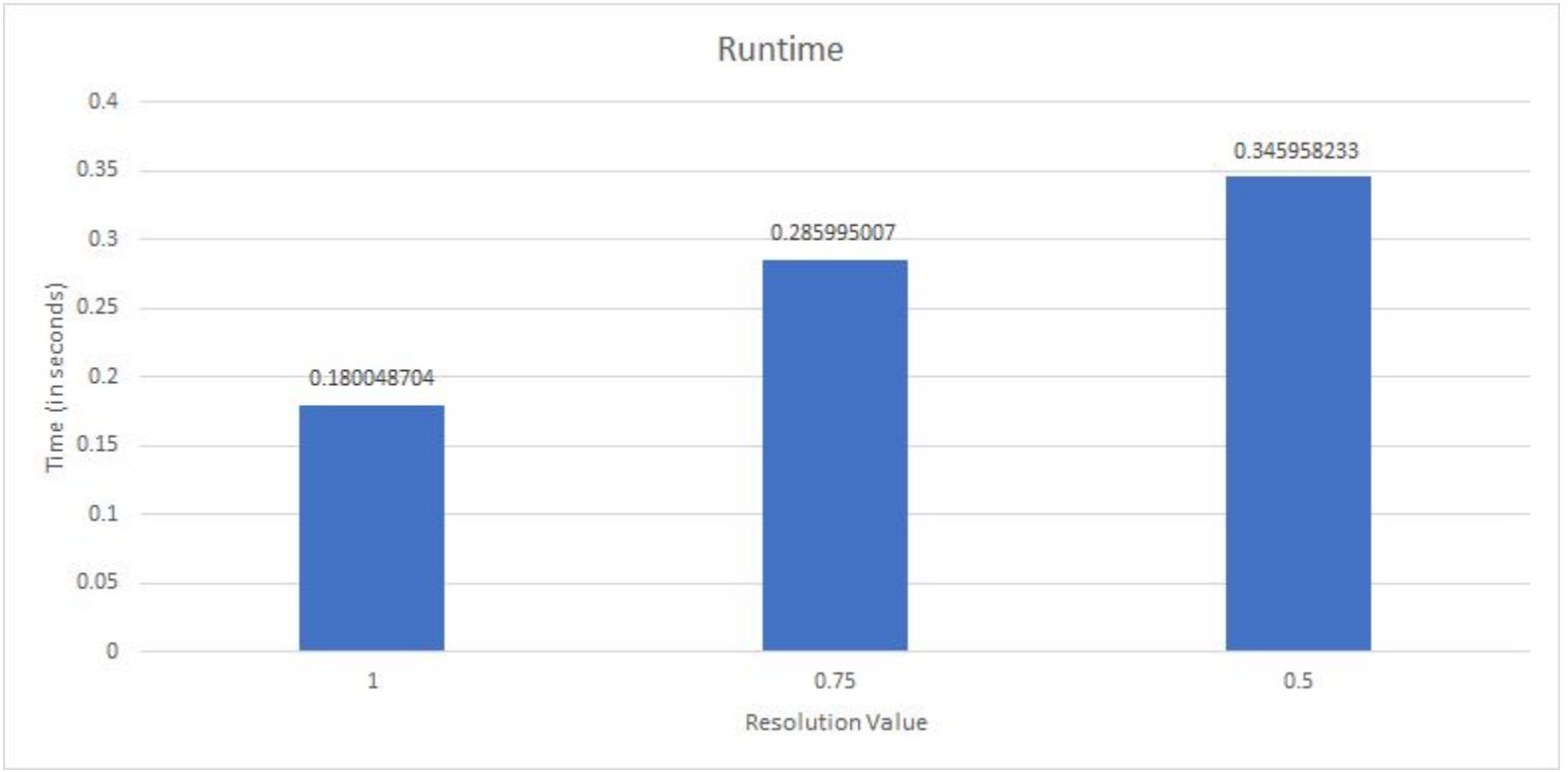
Number of nodes in the community	Average Jaccard of Function Vectors in a Community
12	0.9999999833333336
15	0.8015872996787603
17	0.24761904750113378
11	0.0
29	0.31481481477317264
36	0.6016144348620682
18	0.621212

Resolution = 0.5

Number of nodes in the community	Average Jaccard of Function Vectors in a Community
12	0.6666666650793651
9	0.6666666644444444
38	0.8440170934760576
35	0.6051282047957175
24	0.6924242417947658
20	0.6814814807242797
20	0.3088235292982266
15	0.6666666644444444
7	0.74999999375
9	0.6999999965

Number of nodes in the community	Average Jaccard of Function Vectors in a Community
10	0.0
14	0.3846153843688363
4	0.0
19	0.7323232317684423
13	0.0
14	0.7333333325185186
24	0.7111111103209877
31	0.6318681315209516
12	0.16666666654040405
8	0.142857

Runtime


Runtime
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Time (in seconds)
0.180048704
0.285995007
0.345958233
1
0.75
0.5
Resolution Value

Future Work

Bigger database with Mouse Retina

- No functional metadata (or much of any metadata other than position)
- Better metric

Also bigger database of human connectomes

Use a distributed form of Louvain