

App Development for Analyzing Biological Networks

Students: William Lopez, Valeria Garcia, Leonardo Villa, Tyler Uwate, Oscar Alvarez

Product Owner: Ananda M. Mondal, **Instructor:** Masoud Sadjadi

Introduction

Using Cyfinder through Cytoscape in order to create various types of subnetworks through the function “Subgraph Finder.”

Methods in use:

- InfomapAlgorithm
- InfomapImplementation
- InfomapTester

Primarily coded in InfomapAlgorithm where we integrated the Coarse Tuning Algorithm (6). Implemented the algorithm within InfomapImplementation and tested it through InfomapTester.

Current Systems

Infomap is a fast search algorithm that focuses on finding modular descriptions of different networks and modules of nodes. In order to do this efficiently all algorithms should be implemented as they improve accuracy and create more desired clusters.

Problem

Current system only had Algorithm 3, 4, and 5. In order to continue implementing algorithms it was pivotal to implement algorithm 6. Algorithm 6 is required for algorithm 2 which is used in all the remaining algorithms.

Algorithm 6

Algorithm 6 focuses on using repeated node aggregation (algorithm 3) on different hierarchies of submodules from existing modules. This is done by honing in on the module itself and treating it as a network with nodes and edges. Once the submodules are clustered they can be treated as modules by Infomap. This is where cluster gets called again but with all modules included.

Solution and Code

```
Map<Node, Integer> previousNodeToModule = new HashMap<>(nodeToModule);
oneModulePerNode(g);
Set<Node> setOfNodes;
List<Node> listOfNodes = new ArrayList<>();
List<Edge> listOfEdges = new ArrayList<>();
Graph tempOriginalGraph = originalGraph;
//Graph subGraph = new Graph("subGraph");
Map<Integer, List<Node>> subMap = new HashMap<>();
ArrayList<HashMap<Integer, List<Node>>> listOfMap = new ArrayList<>();
for (Node Mi : previousNodeToModule.keySet()) {
    //line 3
    setOfNodes = moduleMembers.get(Mi);
    listOfNodes = new ArrayList<>(setOfNodes);
    listOfEdges = Mi.getEdges();
    Graph clusteredGraph = new Graph("clusteredGraph", listOfNodes, listOfEdges);
    //subGraph.union(clusteredGraph);
    originalGraph = clusteredGraph;
    //line 4
    subMap = cluster();
    listOfMap.add(new HashMap<Integer, List<Node>>(subMap));
}
```

- Focused on the submodules within the already existing modules
- Created a “network” or graph in this example in order to cluster the network of submodules
- Added these new submodules into a list in order to create a new network that will be clustered again

```
originalGraph = tempOriginalGraph;
Graph subGraph = new Graph("subGraph");
Graph subGraphModule;
listOfEdges = new ArrayList<>();
for (Map<Integer, List<Node>> entry : listOfMap) {
    for (Integer key : entry.keySet()) {
        listOfNodes.addAll(entry.get(key));
        subGraphModule = new Graph("subGraphModule", listOfNodes, listOfEdges);
        subGraph.union(subGraphModule);
    }
}
originalGraph = subGraph;
return cluster();
}
```

- The graph or network of the submodules is called subGraph which gets inputted into cluster at the end
- In order to create this graph we needed the edges and nodes that each submodule consisted of
- Although not ideal we used a nested for loop to find this critical information

Conclusion and References

As a group we continued working on the Infomap process by implementing a crucial algorithm for future development. The algorithms that are required to complete Infomap have prerequisites and our goal was to check those off. A lot of our contributions were possible because of the “Mapping Higher-Order Network Flows in Memory and Multilayer Networks with Infomap” paper by Daniel Edler, Ludvig Bohlin, and Martin Rosvall.