



App Development for Analyzing Biological Networks

Students: David Barahona, Gabriel Zavala, Rigues Brustus
Mentor: Ananda Mondal, Florida International University
Instructor/Faculty: Masoud Sadjadi, Florida International University

INTRODUCTION

Cytoscape, a bioinformatics tool, analyzes biological networks graphically, such as cancer. Its CyFinder plugin adds clustering algorithms for early disease detection. SubgraphFinder, another component, develops and implements these algorithms, enhancing Cytoscape's research capabilities.

CURRENT SYSTEM

The current system allows for the conversion of directed to undirected graphs, offering both additive and transformative approaches. It enables users to set search conditions, choose search algorithms, and order graphs based on specific criteria. The system supports saving and sorting subgraphs, visualizing k-partite graphs, and updating the GUI. Key features include finding maximum cliques and bipartite subgraphs, creating organized layouts, and identifying clusters using various methods. Additionally, it integrates the map equation tool and the Infomap community detection algorithm, enhancing its network analysis and community detection capabilities.

PROBLEM

- When using algorithms to find the best solution, there's a common problem where the algorithm gets stuck on a "good enough" answer, but not the best one. This happens because it keeps looking at the options in the same order.
- Ensures that computational efforts are directed toward meaningful improvements, enhances the quality of the algorithms, and prevents them from overfitting to the specific characteristics of the dataset.

SOLUTION

Enhance algorithm performance by introducing a code length threshold, boosting the efficiency and effectiveness of the clustering() and optimize() functions, and by adding randomization to explore a wider range of solutions, increasing the likelihood of finding the optimal answer.



IMPLEMENTATION

Introduced a codeLengthThreshold variable and modified the optimize and cluster functions to check if bit lengths fall below this threshold. Added randomness using Collections.shuffle() to ensure varied and random node optimization in each iteration..

```
// Check if the new bit length is below the threshold
if (newBitLength < codeLengthThreshold) {
    continue;
}
```

```
private void optimize(Graph g, Map<Node, Integer> nodeToModule) { // algo 4
    List<Node> nodeList = g.getNodeList();
    Collections.shuffle(nodeList); // adds randomization
    boolean shouldContinue = true;
```

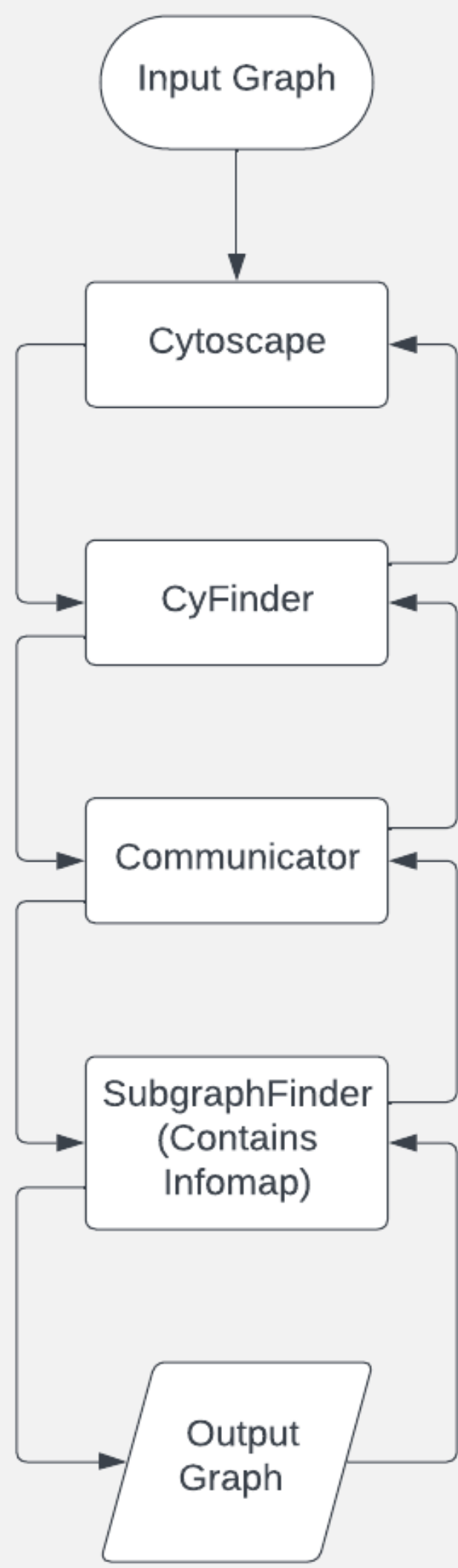
SUMMARY

Our team undertook several tasks, including enhancing algorithms 3 and 4 by integrating a code length threshold and introducing randomization to algorithm 4. We also expanded the functionality of the Map equation and Infomap algorithm to support directed weighted networks. Additionally, we completed the implementation of the Infomap algorithm by incorporating algorithms from the paper "Mapping Higher Order." My primary contribution was adding the code length threshold to algorithms 3 and 4, along with implementing randomization in algorithm 4.

REQUIREMENTS

- Java 11.0
- Apache Ant
- Maven
- CytoScape 3.9.1

SYSTEM DESIGN



REFERENCES

- Rosvall, M., Axelsson, D. & Bergstrom, C. The map equation. Eur. Phys. J. Spec. Top. 178, 13–23 (2009). <https://doi.org/10.1140/epjst/e2010-01179-1>
- Edler D, Bohlin L, Rosvall M. Mapping Higher-Order Network Flows in Memory and Multilayer Networks with Infomap. Algorithms. 2017; 10(4):112. <https://doi.org/10.3390/a10040112>
- Edler, D., Holmgren, A., & Rosvall, M. (n.d.). Simplify and highlight important structures in complex networks. MapEquation. Retrieved April 17, 2023, from <https://www.mapequation.org/>

ACKNOWLEDGEMENT

The material presented in this poster is based upon the work supported by Dr. Ananda Mondal. I thank Dr. Ananda Mondal for his assistance and mentorship that I/we received throughout the senior design project.