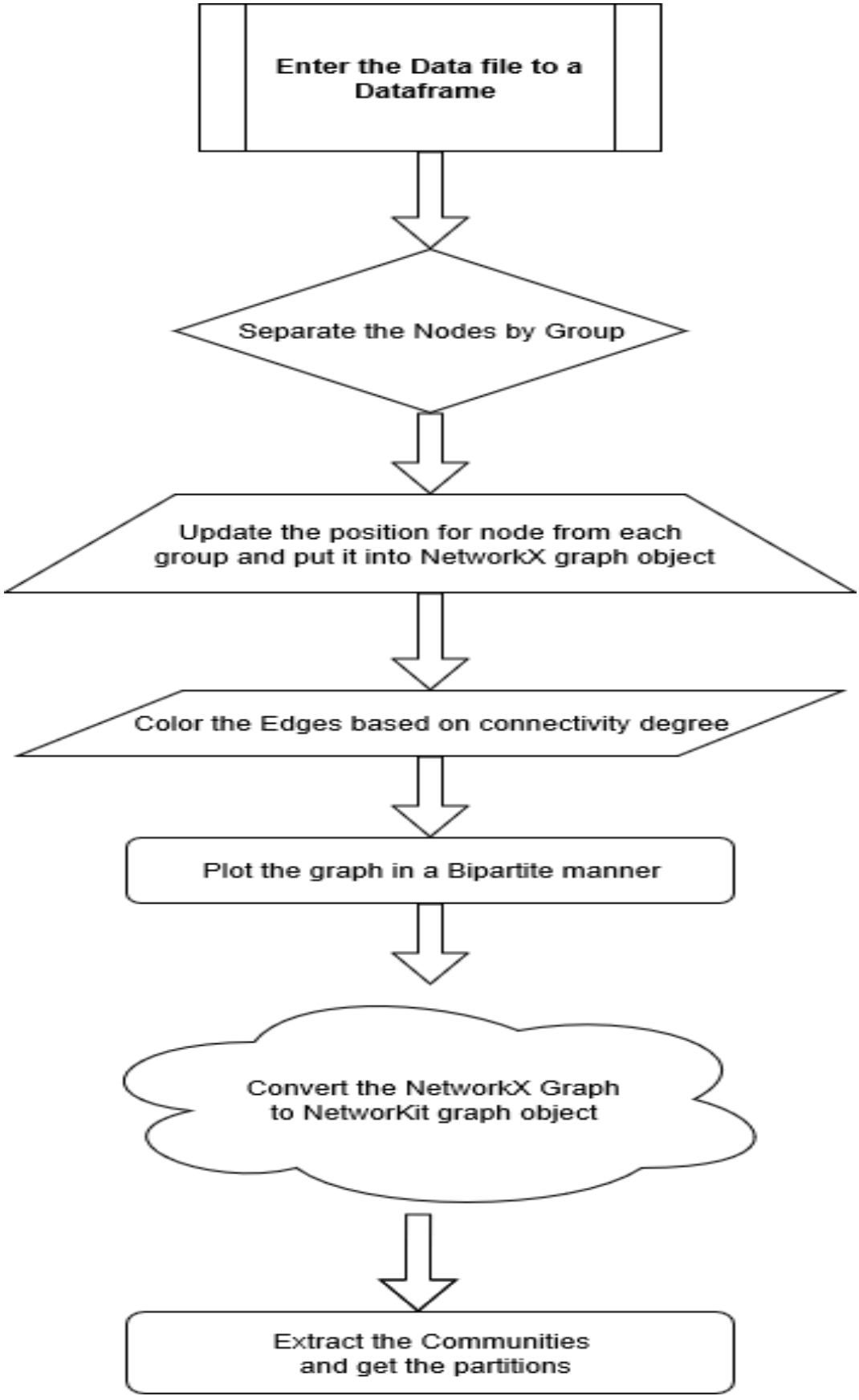
Enter the Data file to a Dataframe
Separate the Nodes by Group
Update the position for node from each group and put it into NetworkX graph object
Color the Edges based on connectivity degree
Plot the graph in a Bipartite manner
Convert the NetworkX Graph to NetworKit graph object
Extract the Communities and get the partitions

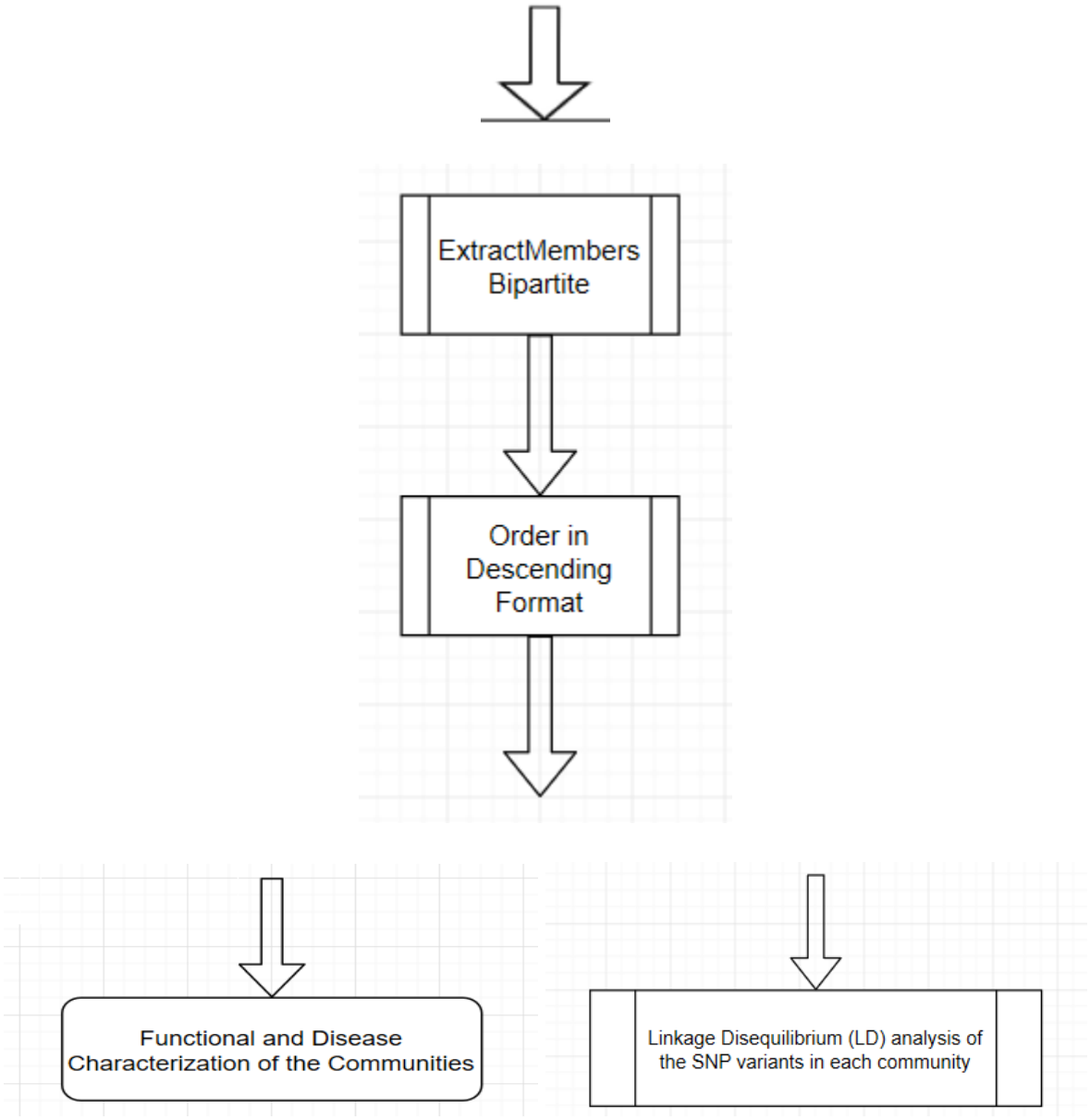

Figure C: Flowchart for workflow of GraphBreak. Plot generated using www.draw.io

GraphBreak PSEUDOCODE

==========

dataframe df <- read_csv(sys.argv[1]) #Here goes the file name which is space or tab separated and 1st row as names of the columns

item1 <- df[sys.argv[2]].unique() #Here goes the first column variable name such as the name of the genes

```
item2 <- df[sys.argv[3]].unique() #Here goes the second column variable name such as the name of the SNPs

edges1 <- df[sys.argv[4]]

edges2 <- df[sys.argv[5]]

edges <- concat([edges1, edges2], axis=1)

edgesArray<- edges.values #Here convert the dataframe into array

B <- networkx.Graph() #Initiate the object for networkx

B.add_nodes_from(item1, bipartite=0) # Add the node attribute "bipartite"

B.add_nodes_from(item2, bipartite=1)#Here we add the second set of data in bipartite manner

B.add_edges_from(edgesArray)


#Separating the nodes by group

r <- Getting the top nodes where bipartite = 0

l = set(B) - r #Getting the lower nodes


#Creating a File where the list of Nodes is written

listOfNodes = list(nodes in graph B)

with open ("nodes.txt", "w") as thefile:

   for item in listOfNodes:

thefile.write<-item

thefile.close()


pos = {} #Here we initialize a variable to store the positions

# Update position for node from each group THis will be needed for two parallel lines as bipartite

pos.update((node, (1, index)) for index, node in lower nodes l

pos.update((node, (2, index)) for index, node in higher nodes r
```

#Put Color to the edges based on number of edges connectedness

color_map = []

for nodeCount in range(len(item1)):

color_map.append('pink') #Item 1 objects colored one color

for nodeCount in range(len(item2)):

color_map.append('green') #Item 2 objects colored second color

#To plot with degrees of association in linear bipartite

nx.draw to plot the bipartite based on the positioning and color and based on degree of connectedness

show the plot

##Community Detection ##

nk.setNumberOfThreads<- N Setting the number of Parallel Threads in OpenMP

nkG = nx2nk(B) #Convert NetworkX to NetworKit graph and store it as a new Graph variable name

communities = nk.community.detectCommunities(nkG) #Call the method to detect communities

#Write the community partitioning

nk.community.writeCommunities(communities)

#Plotting Communities

drawCommunityGraph

#Use functional and disease characterization statistical database tools

#Extract Bipartite members by Prefix

df= read_csv(sys.argv[1]) #Here we read the filename into dataframe

prefix = sys.argv[2] #Storing the prefix variable

mask = df['node'].str.match(prefix)

df[mask].to_csv("communitiesPrefix.csv")

#Order the members in community by descending format

df= pandas.read_csv(sys.argv[1])

df = df.assign(sort_values(['G','community'],ascending=[False,True]).drop('G',1)

df.to_csv("communitiesOrdered.csv”)

#Conduct Downstream Over-Representation and LD analysis

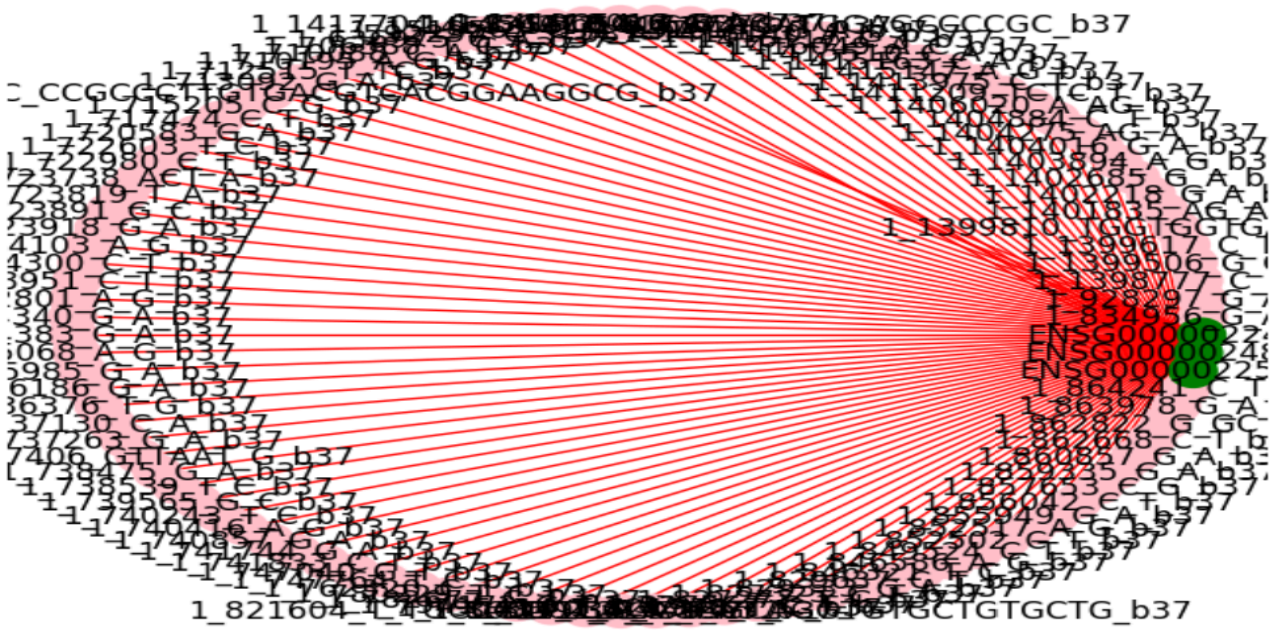

Figure 1 GraphBreak Association plot for 3 genes with the genomic variants

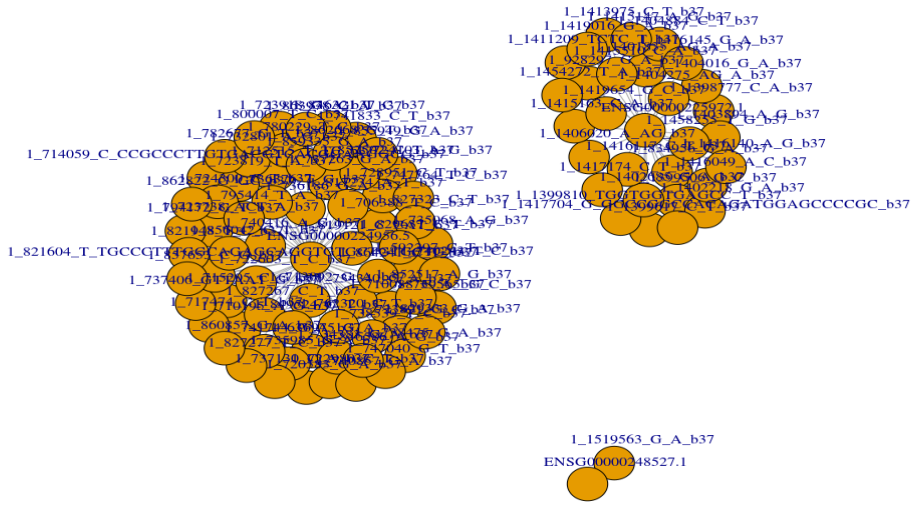

Figure 2 Condor association connections plot of sample 3 genes with the genomic variants

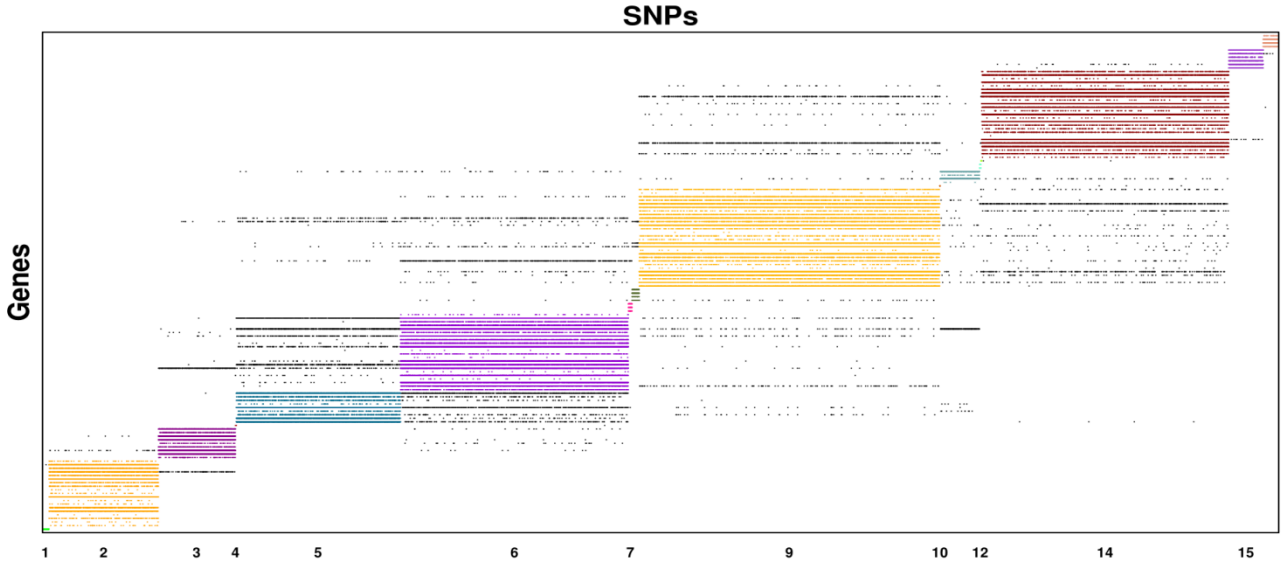

Figure 3 a. Condor communities for sample tissue data Artery-Aorta GTEx V7

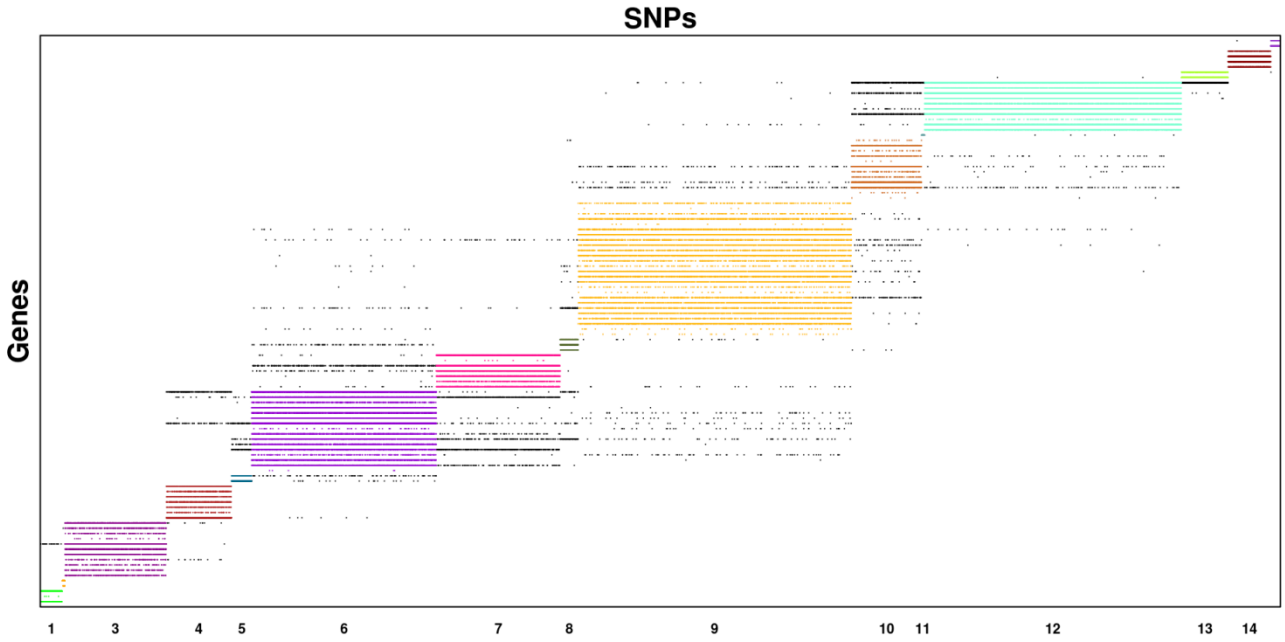

Figure 3 b. Condor communities for sample tissue data Artery-Coronary GTEx V8

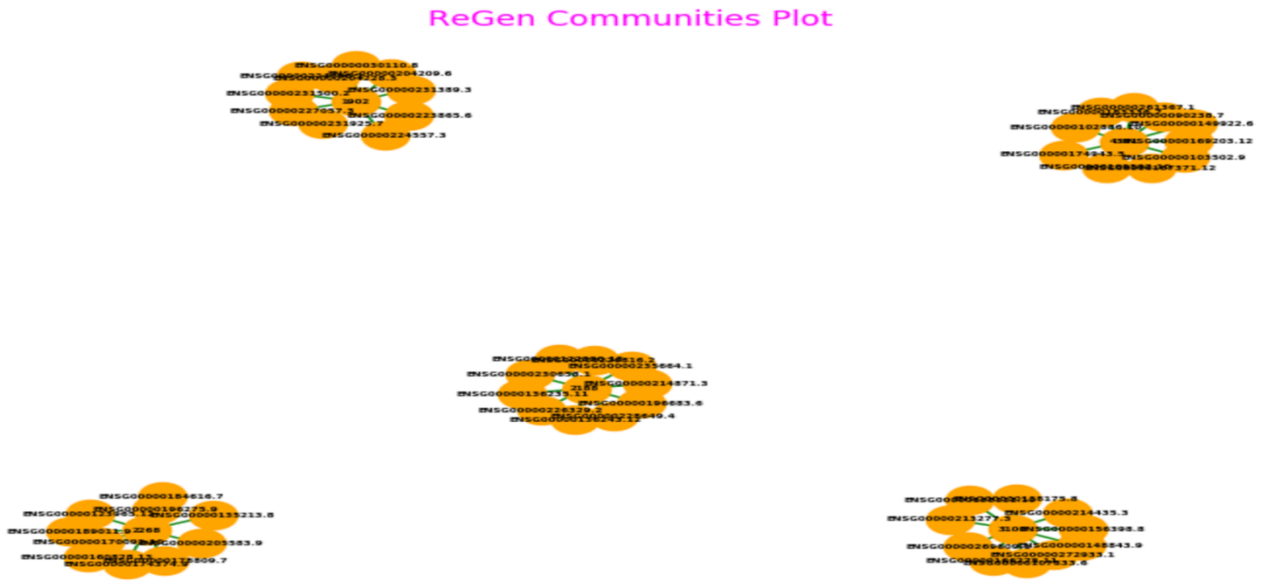

Figure 4 GraphBreak (also named as ReGen - for Regulatory Genomic work) communities plot for 5 of the communities

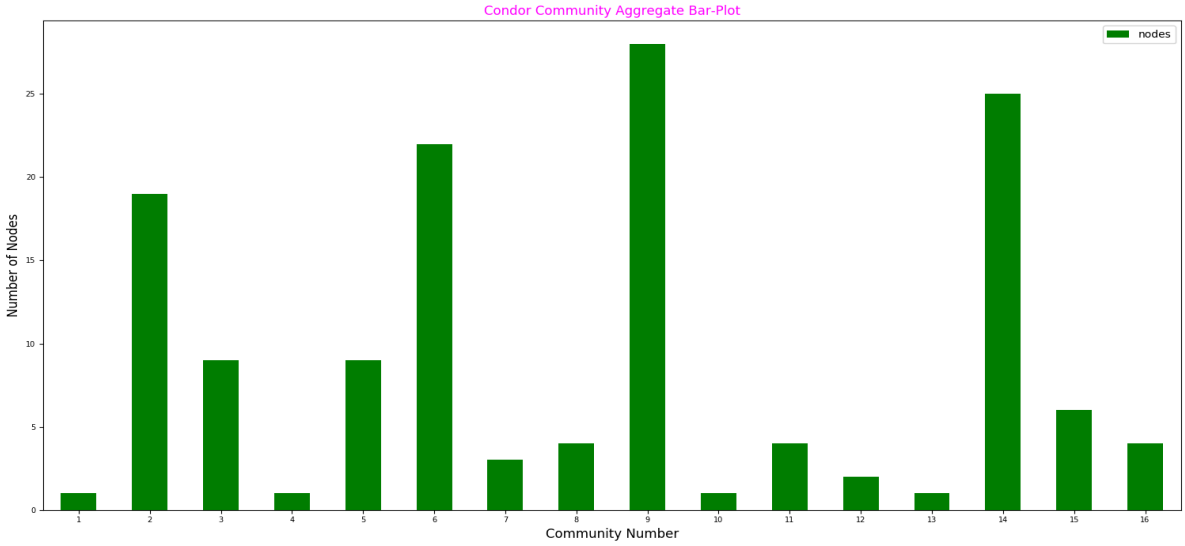

Figure 5 Bar plot of sum of genes in each of the communities obtained by Condor

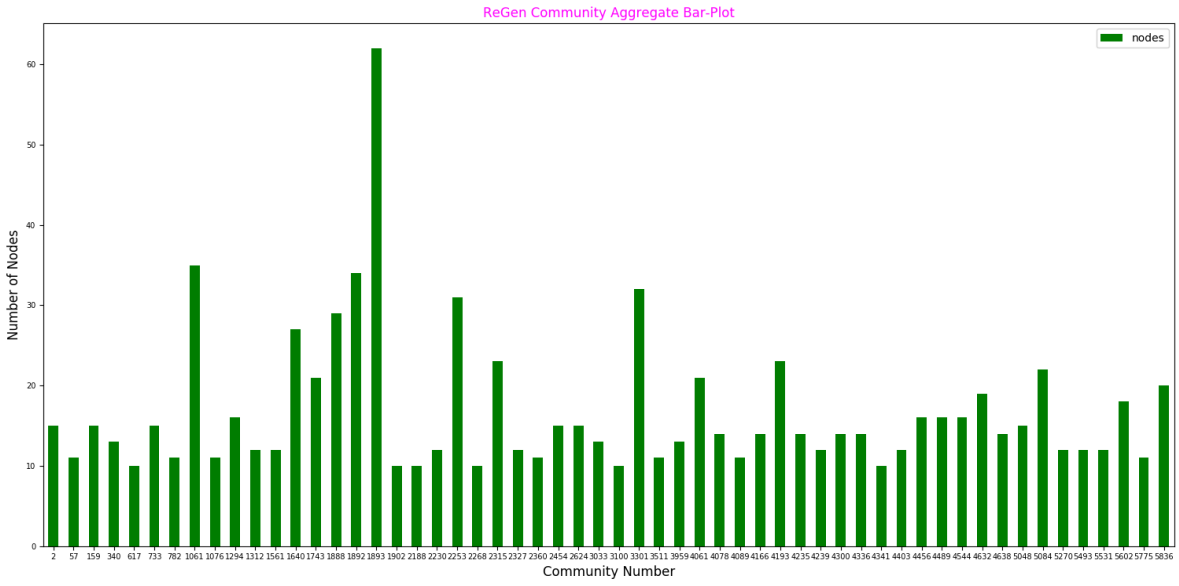

Figure 6 Bar plot for sum of genes in each of the communities obtained from GraphBreak

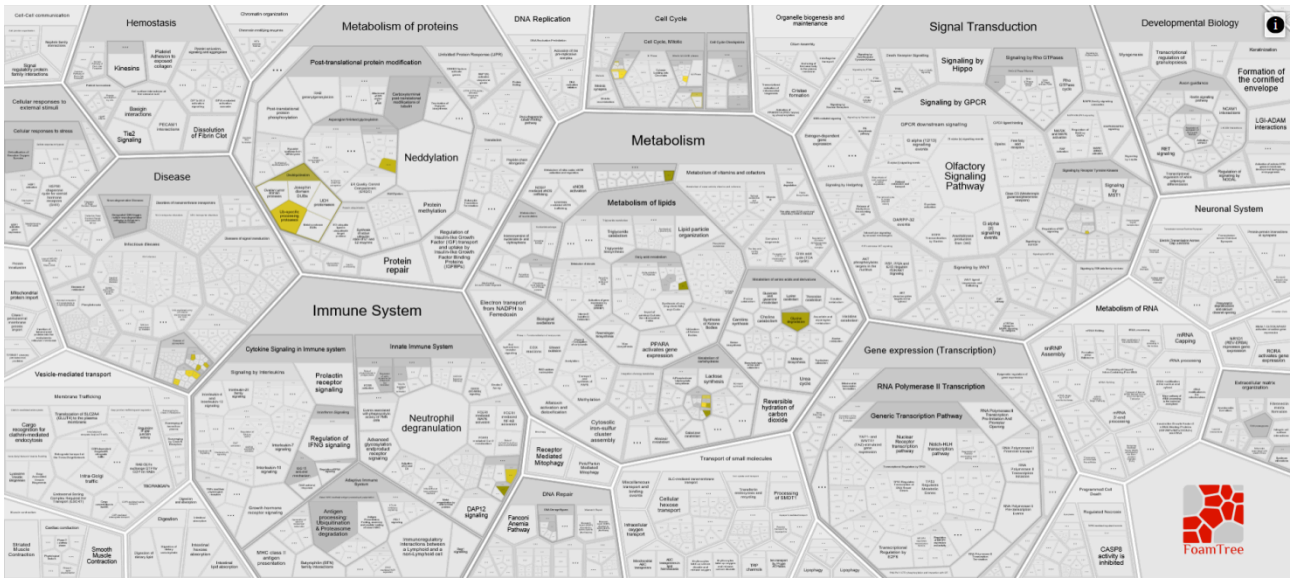

Figure 7 GraphBreak Community 1061 Genes 'Voronoi Diagram' Reacfoam p-value association

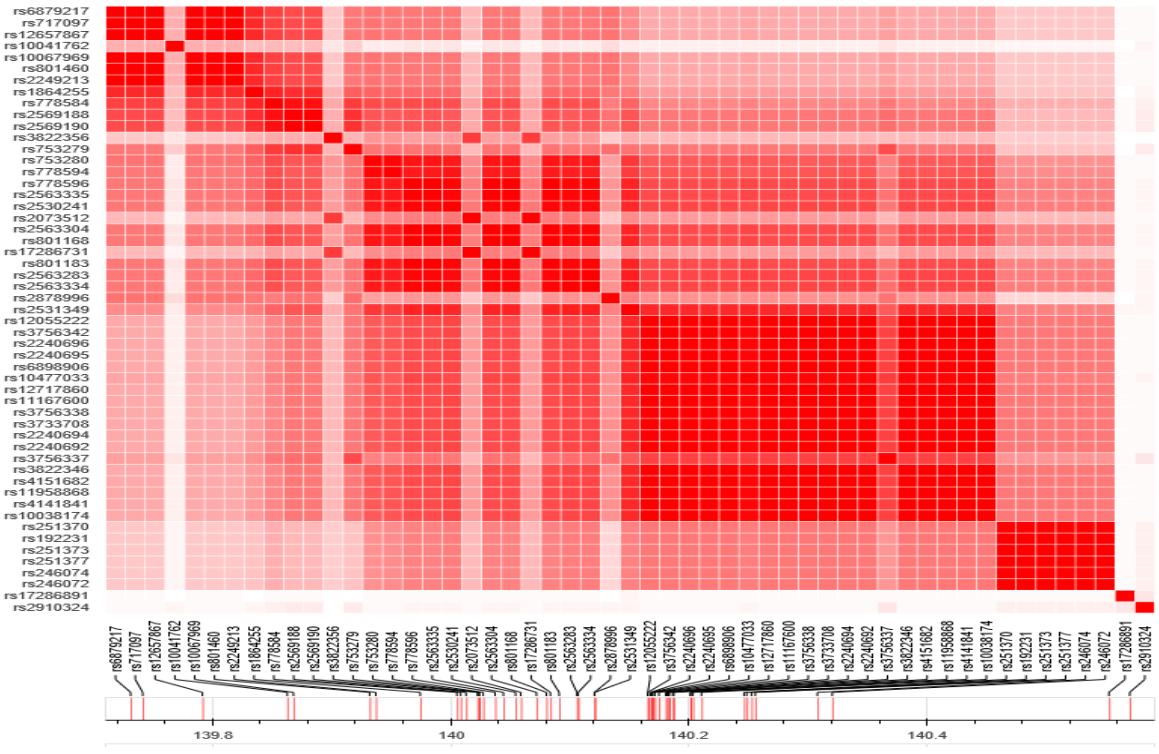

Figure 8 Classifying variants based on their LD associations

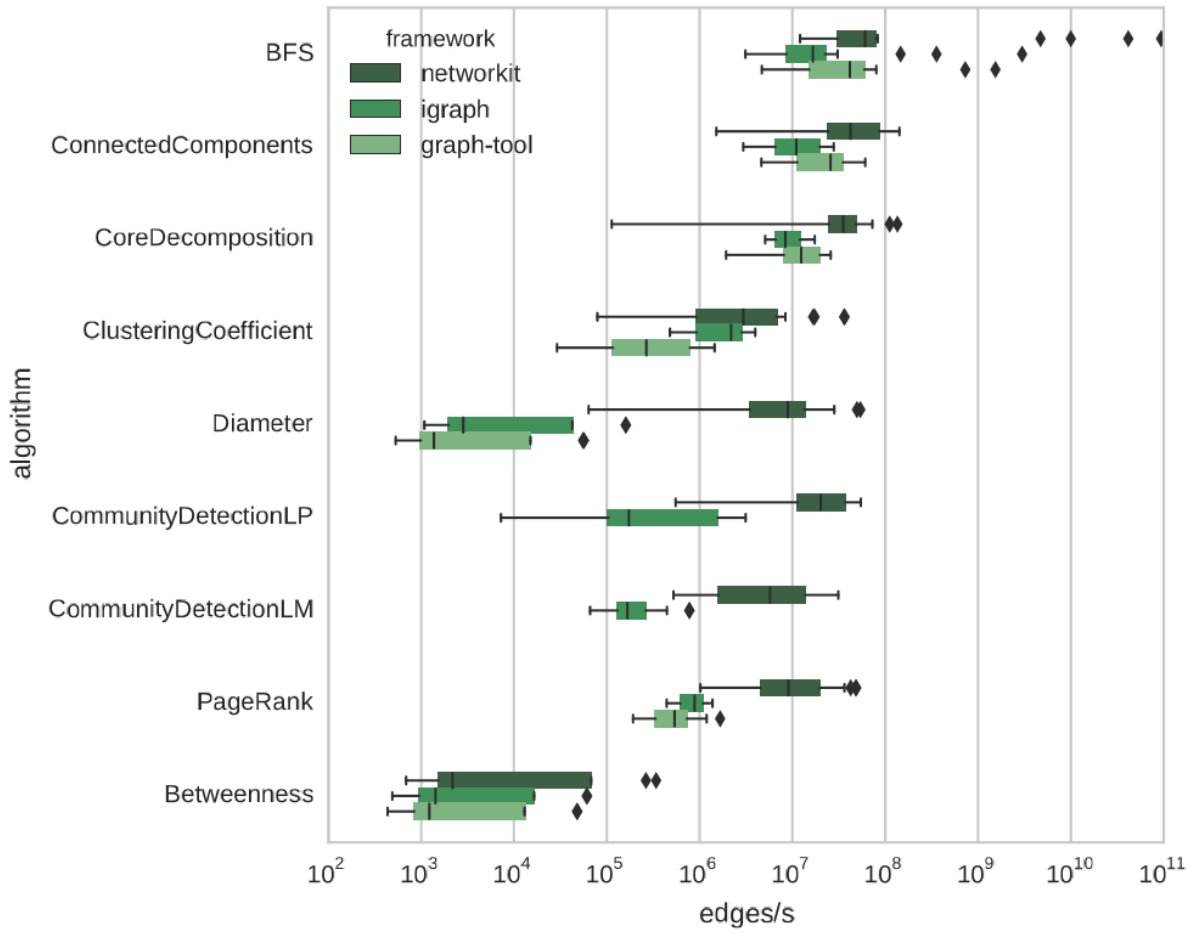

Figure A: Processing rates of typical analytics tasks: NetworKit in comparison with igraph and graph-tool

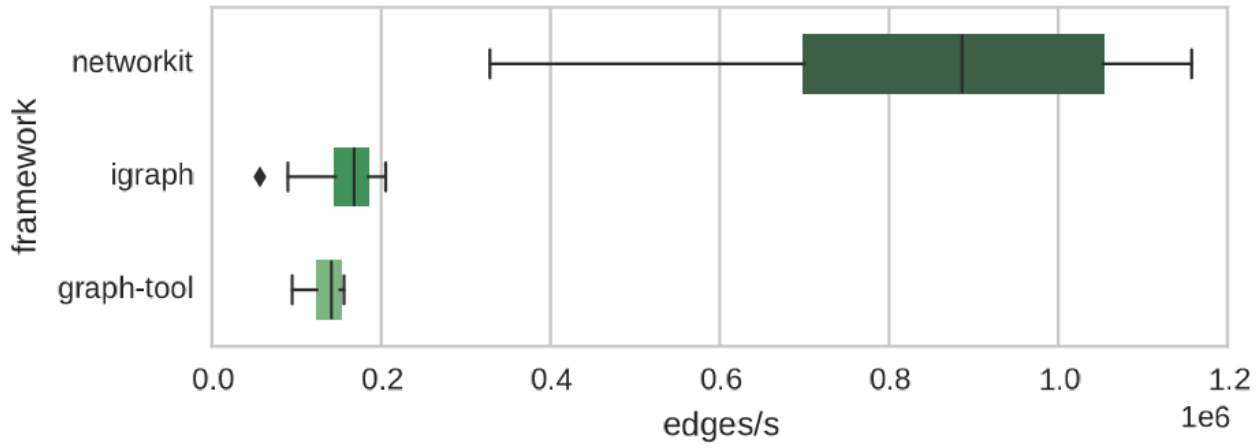

Figure B: I/O rates of reading a graph from a GML file: NetworKit in comparison toigraph and graph-tool

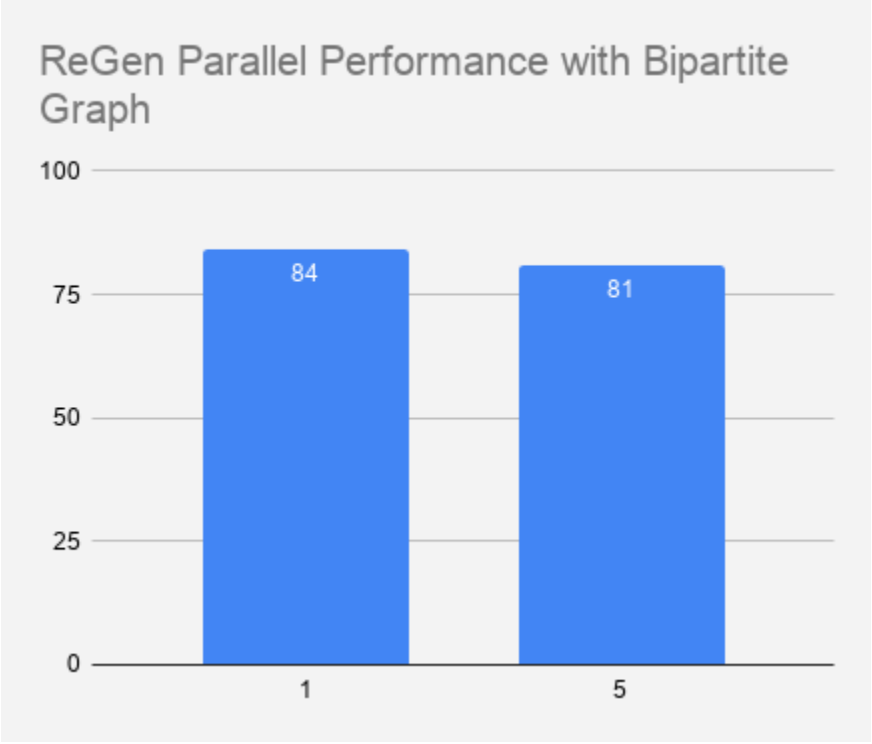

Figure 11: Parallel performance of GraphBreak (in this case regulatory network analysis so also called ReGen) is not greatly benefitted when the bipartite graph also needs to be plotted. X-axis number of computing cores, Y-axis time in minutes. (Data tested: Artery-Coronary tissue eQTL GTEx V8.)

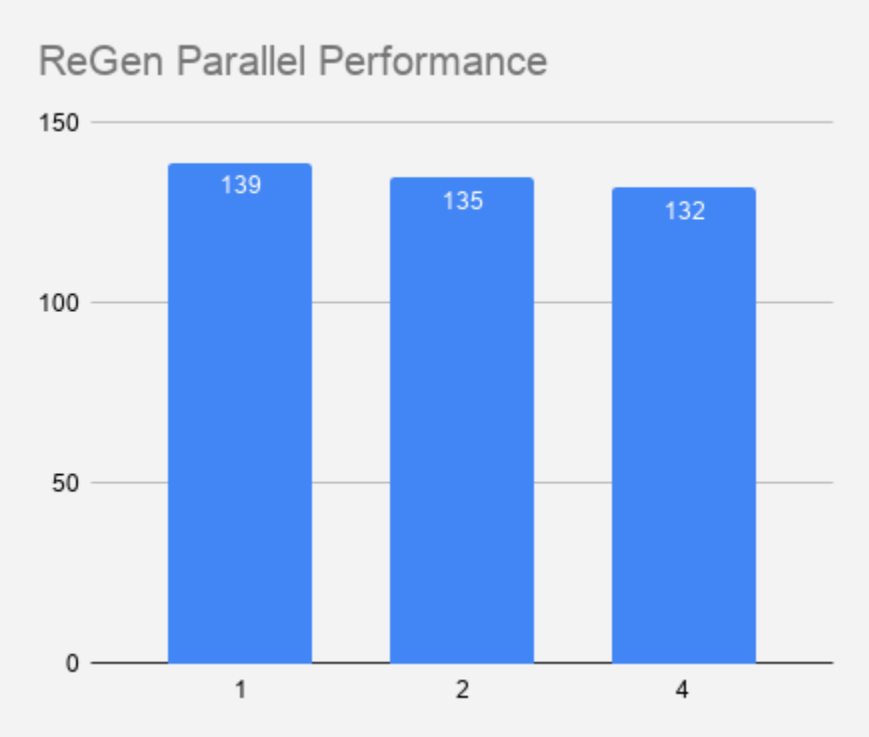

Figure 12: Execution time for GraphBreak (in this case regulatory network analysis so also called ReGen), Y-axis (in seconds), is far less when community detection is done without bipartite plotting. X-axis is number of cores used for computing. (Data tested: Artery-Coronary tissue eQTL GTEx V8.)