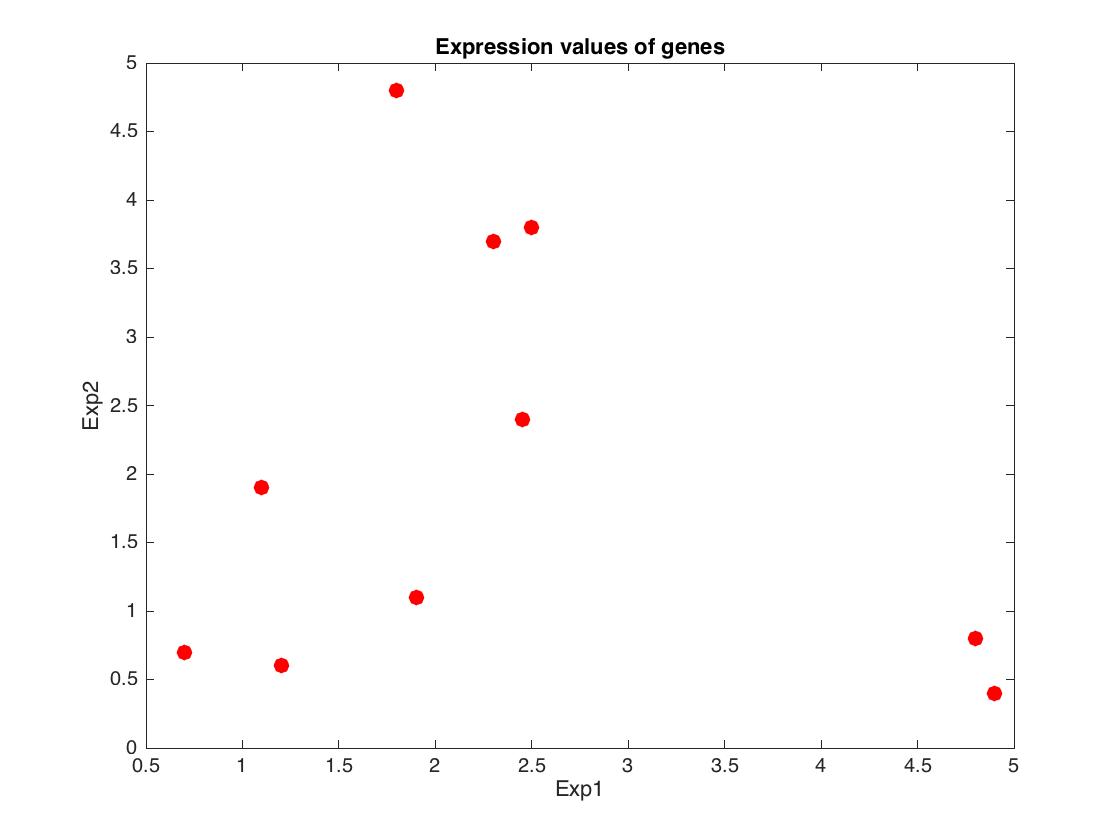
Q1.

1. The only changes applicable would be to equation used for computing similarity matrix, which are as follows:

The (*lambda)* refers to opening gaps along with gap extensions . Since the only modification proposed is the use of an additional constant in penalising the opening gaps, this would not make cause any change to the number of iterations in dynamic program, and hence resulting with no change in computational complexity.

2. a. Euclidean distance between the genes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Gene # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | 1 | 0 | 2.36 | 1.44 | 2.98 | 2.16 | 3.86 | 1.13 | 1.3 | 1.26 | 4.09 | | 2 | 2.36 | 0 | 1.4 | 1.22 | 0.22 | 3.78 | 2.77 | 3.45 | 3.58 | 4.16 | | 3 | 1.44 | 1.4 | 0 | 2.49 | 1.31 | 2.84 | 1.41 | 2.19 | 2.44 | 3.16 | | 4 | 2.98 | 1.22 | 2.49 | 0 | 1.21 | 5 | 3.7 | 4.24 | 4.24 | 5.38 | | 5 | 2.16 | 0.22 | 1.31 | 1.21 | 0 | 3.83 | 2.63 | 3.29 | 3.4 | 4.2 | | 6 | 3.86 | 3.78 | 2.84 | 5 | 3.83 | 0 | 2.92 | 3.61 | 4.1 | 0.41 | | 7 | 1.13 | 2.77 | 1.41 | 3.7 | 2.63 | 2.92 | 0 | 0.86 | 1.26 | 3.08 | | 8 | 1.3 | 3.45 | 2.19 | 4.24 | 3.29 | 3.61 | 0.86 | 0 | 0.51 | 3.71 | | 9 | 1.26 | 3.58 | 2.44 | 4.24 | 3.4 | 4.1 | 1.26 | 0.51 | 0 | 4.21 | | 10 | 4.09 | 4.16 | 3.16 | 5.38 | 4.2 | 0.41 | 3.08 | 3.71 | 4.21 | 0 | |



b. K-medoids clustering

Result after the first epoch/iteration:

|  |  |  |
| --- | --- | --- |
| Cluster # | Gene # | Medoid |
| 1 | 1,7,8,9 | 7 |
| 2 | 2,4,5 | 4 |
| 3 | 3,6,10 | 6 |

Result after the second epoch/iteration:

|  |  |  |
| --- | --- | --- |
| Cluster # | Gene # | Medoid |
| 1 | 1,7,8,9 | 8 |
| 2 | 2,4,5 | 5 |
| 3 | 3,6,10 | 10 |

Result after the third epoch/iteration:

|  |  |  |
| --- | --- | --- |
| Cluster # | Gene # | Medoid |
| 1 | 1,7,8,9 | 9 |
| 2 | 2,4,5 | 2 |
| 3 | 3,6,10 | 3 |

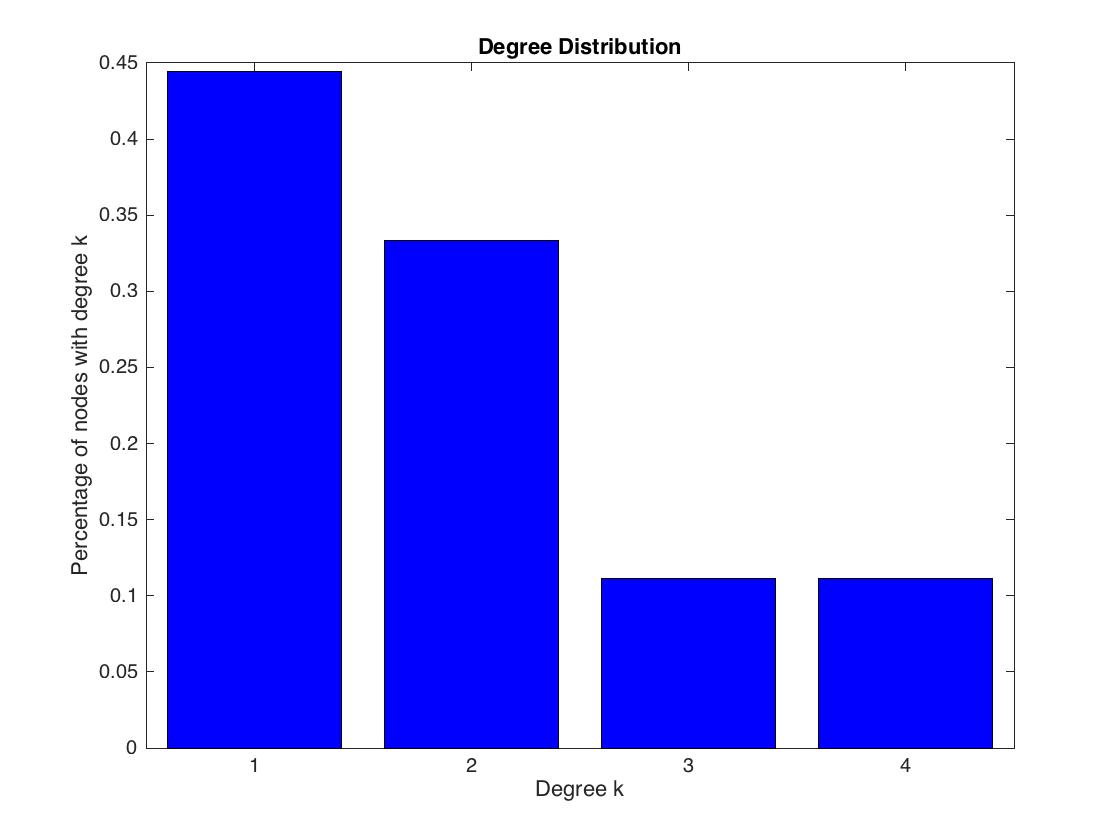
There is a swap in medoid within every cluster, and this is because there is more than 1 data-point, which has the least distance to non-medoid points, and during such case randomness is used to pick one among the data-points with least cost.

Based on the results from each step, it is intuitive that the provided dataset does not carry noise, which makes the algorithm converge in the first step itself. The first step refers to associating non-medoid points to closest medoids before any epochs.

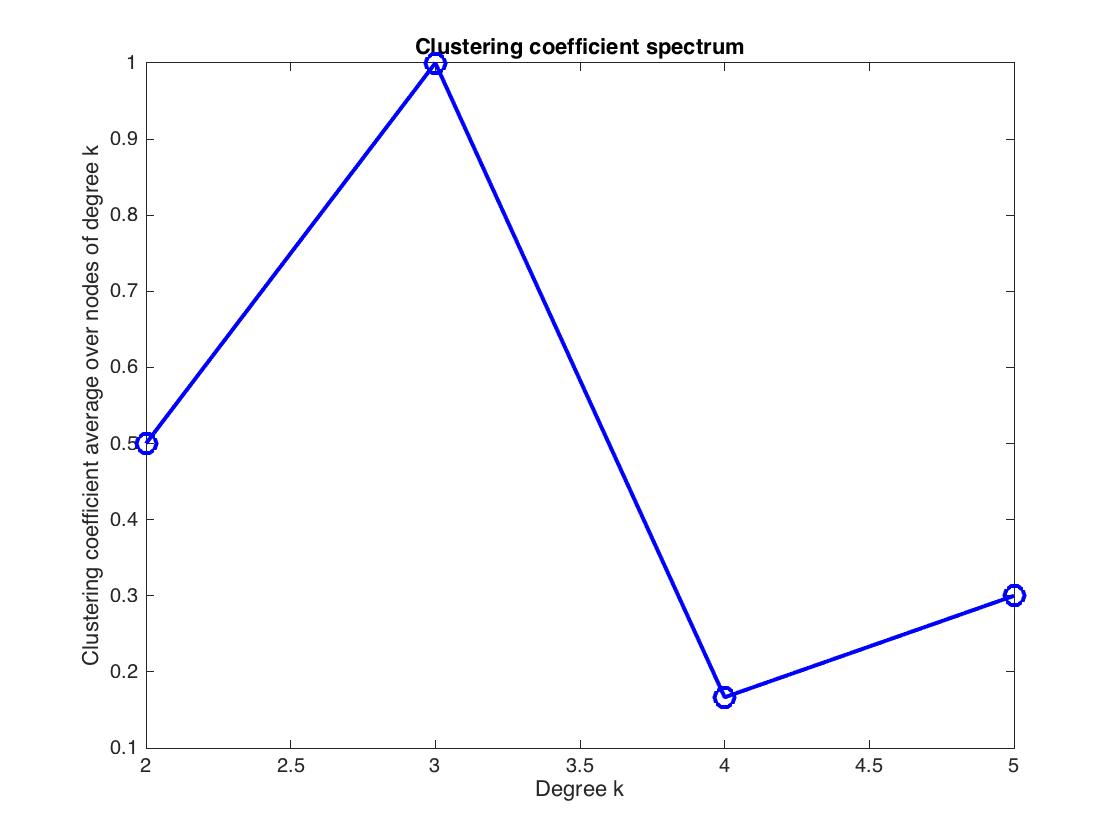
Q2.

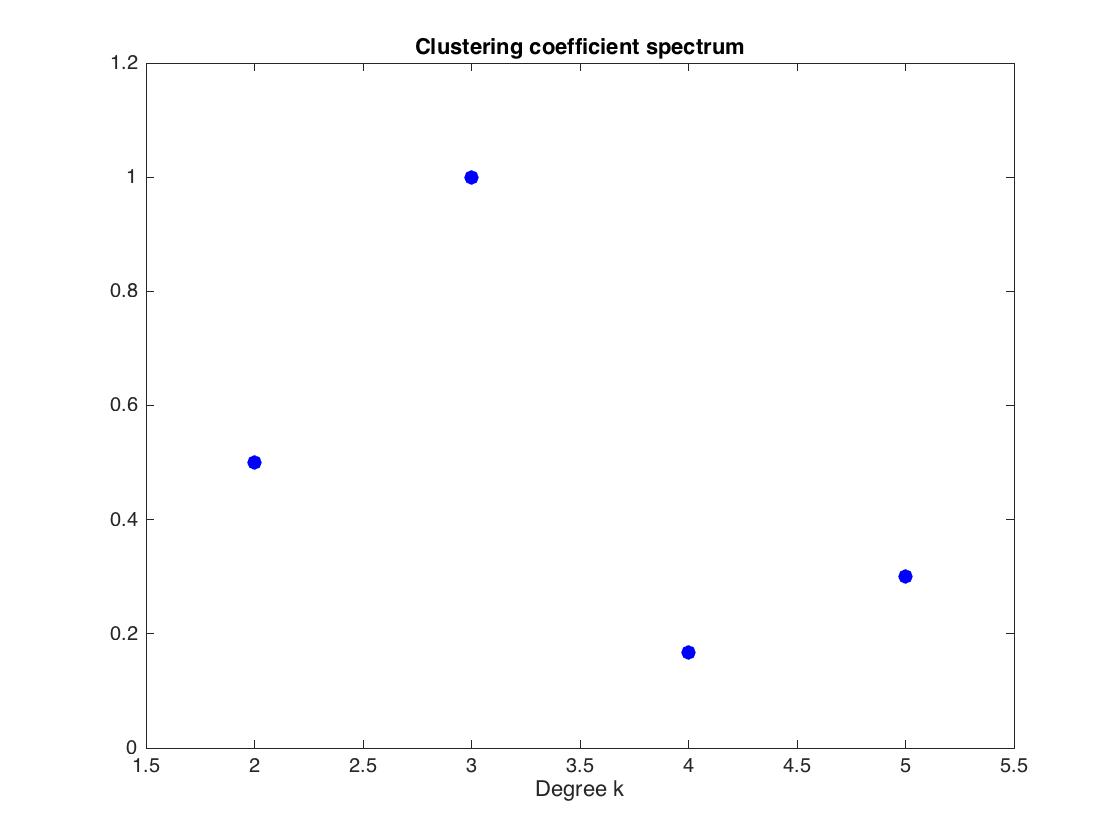
1. In general, proteins do not function in an isolated manner, rather they tend to interact with each other, for instance molecules, which would intervene with metabolic pathways. Metabolic pathways of a cell constitute to the formation of metabolic network, which are interconnected by biochemical reactions that are catalysed by enzymes: *proteins that accelerate chemical reactions* in a cell.

2. a. Degree Distribution



b. Clustering coefficient spectrum

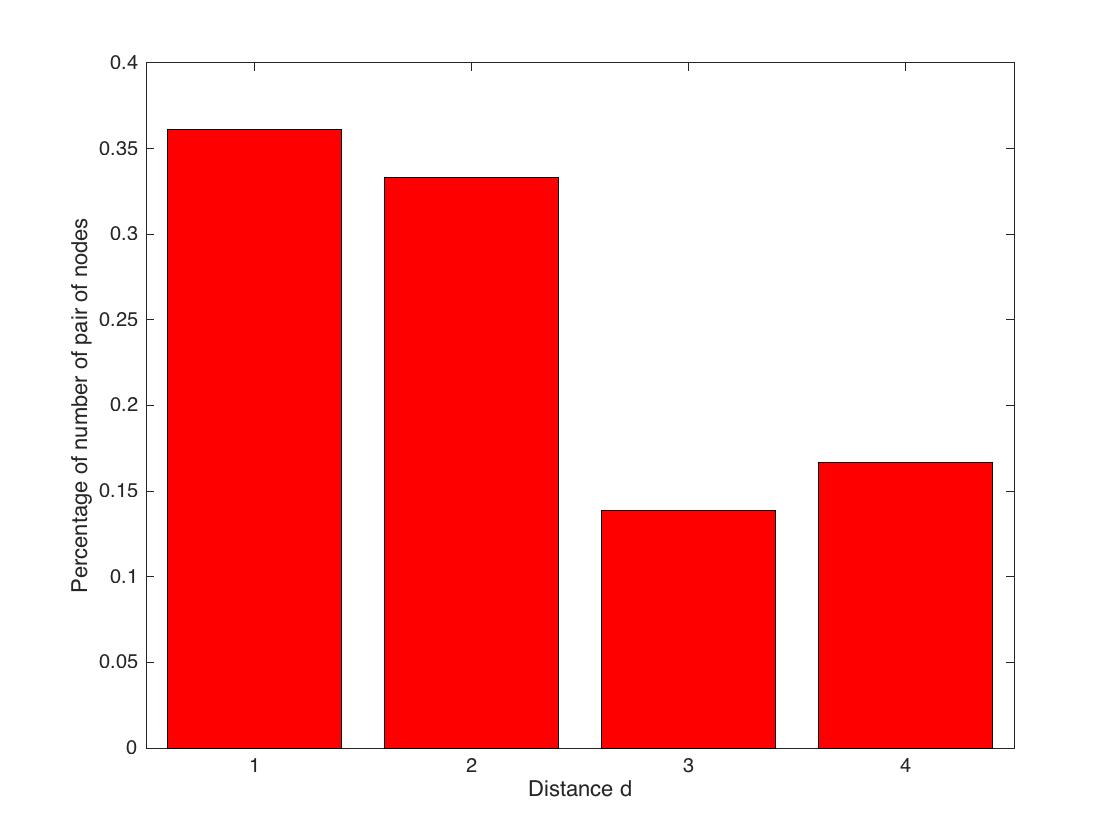




Both the figures are just the same except the second one's bounds were slightly adjusted to make the points more prominent without lines.

Average clustering coefficient of the given network is 0.61

c. Shortest path length distribution



d. Betweenness centrality distribution

e. Graphlets of types and

f. All Automorphisms

All Automorphisms

Automorphism Orbits

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Orbit # | Node(s) | Orbit # | Node(s) | Orbit # | Node(s) |
| 1 | 1,2 | 4 | 4,5 | 7 | 7,8,9 |
| 2 | 1,2 | 5 | 4,5 | 8 | 7,8,9 |
| 3 | 3 | 6 | 6 | 9 | 7,8,9 |

Automorphism Orbits can be easily fetched from the All Automorphism table.

g. Graphlet signatures of nodes 3, 6, 7, and 9 using only the graphlets with 2, 3 and 4 nodes (i.e., omit orbits 15 to 72).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 3 | 4 | 2 | 5 | 1 | 6 | 4 | 0 | 2 | 1 | 0 | 0 | 2 | 0 | 0 | 0 |
| 6 | 5 | 2 | 1 | 3 | 0 | 0 | 0 | 3 | 1 | 0 | 0 | 2 | 0 | 0 | 3 |
| 7 | 3 | 2 | 0 | 3 | 2 | 0 | 1 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 4 |
| 9 | 3 | 2 | 0 | 3 | 2 | 0 | 1 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 4 |

2. What is the number of graphlets of type that a connected graph with 10 nodes and 10 edges can have ? Show the resulting graph:

Matlab code for Task 1

clear;

clc;

close all;

% Adjency matrix...

adj\_mat = zeros(9,9);

adj\_mat(1,2:3) = 1;

adj\_mat(2,[1,3]) = 1;

adj\_mat(3,[1,2,4,5]) = 1;

adj\_mat(4,[3,6]) = 1;

adj\_mat(5,[3,6]) = 1;

adj\_mat(6,[4,5,7,8,9]) = 1;

adj\_mat(7,[6,8,9]) = 1;

adj\_mat(8,[6,7,9]) = 1;

adj\_mat(9,[6,7,8]) = 1;

degree\_vec = sum(adj\_mat,2);

% Degree distribution

deg = unique(degree\_vec);

deg\_vec = zeros(1,length(deg));

for i = 1:length(deg)

n = sum(degree\_vec==deg(i));

deg\_vec(i) = n/9;

end

figure

bar(deg\_vec,'b');

title('Degree Distribution');

xlabel('Degree k');

ylabel('Percentage of nodes with degree k');

Avergage clustering coefficient and its spectrum

cl\_c = zeros(1,9);

for i = 1:9

nodes = find(adj\_mat(i,:) ~= 0);

n = length(nodes);

max\_edges = factorial(n)/(factorial(2)\*factorial(n-2));

m = get\_edges(adj\_mat,nodes);

cl\_c(i) = (m)/max\_edges;

end

avg\_clcoeff = mean(cl\_c);

% plot cl\_c

cl\_dis = zeros(1,length(deg));

for i = 1:length(deg)

node\_indices = find(degree\_vec==deg(i));

% n = length(nodes);

n = length(node\_indices);

cl\_dis(i) = sum(cl\_c(node\_indices))/n;

end

figure

plot(deg,cl\_dis,'.b','LineWidth',2,'MarkerSize',25)

title('Clustering coefficient spectrum');

xlabel('Degree k');

ylabel('Clustering coefficient average over nodes of degree k');

% The below values were estimated manually.

dist\_mat = zeros(9,9);

dist\_mat(1,2) = 1;

dist\_mat(1,3) = 1;

dist\_mat(1,4) = 2;

dist\_mat(1,5) = 2;

dist\_mat(1,6) = 3;

dist\_mat(1,7) = 4;

dist\_mat(1,8) = 4;

dist\_mat(1,9) = 4;

dist\_mat(2,3) = 1;

dist\_mat(2,4) = 2;

dist\_mat(2,5) = 2;

dist\_mat(2,6) = 3;

dist\_mat(2,7) = 4;

dist\_mat(2,8) = 4;

dist\_mat(2,9) = 4;

dist\_mat(3,4) = 1;

dist\_mat(3,5) = 1;

dist\_mat(3,6) = 2;

dist\_mat(3,7) = 3;

dist\_mat(3,8) = 3;

dist\_mat(3,9) = 3;

dist\_mat(4,5) = 2;

dist\_mat(4,6) = 1;

dist\_mat(4,7) = 2;

dist\_mat(4,8) = 2;

dist\_mat(4,9) = 2;

dist\_mat(5,6) = 1;

dist\_mat(5,7) = 2;

dist\_mat(5,8) = 2;

dist\_mat(5,9) = 2;

dist\_mat(6,7) = 1;

dist\_mat(6,8) = 1;

dist\_mat(6,9) = 1;

dist\_mat(7,8) = 1;

dist\_mat(7,9) = 1;

dist\_mat(8,9) = 1;

% get unique distances

u\_dist = unique(dist\_mat);

u\_dist = u\_dist(2:end);

pair\_nodes = zeros(1,length(u\_dist));

for i = 1:length(u\_dist)

pair\_nodes(i) = length(find(dist\_mat==u\_dist(i)));

end

evidence = sum(pair\_nodes);

for i = 1:length(pair\_nodes)

pair\_nodes(i) = pair\_nodes(i)/evidence;

end

figure

title('Shortest distance spectrum');

bar(u\_dist,pair\_nodes,'r')

xlabel('Distance d');

ylabel('Percentage of number of pair of nodes');

Matlab code written for K-Medoid clustering task

function [ clusters ] = cluster\_v2( X )

% Name: Rajkumar Conjeevaram Mohan

% Coursework: Introduction to Bioinformatics

% Clustering genes into three clusters using Euclidean distance

% Initializing vars....

clc;

[m,~] = size(X);

clusters = cell(1,3);

% Initialising medoids

medoids = [1,2,3];

% Initialize similarity matrix

d\_mat = get\_sim\_matrix(X);

% display('Similarity matrix');

% display(d\_mat);

% Step 1: Initialize datapoints closest to medoids

sim\_ = zeros(size(medoids));

for i = 1:m

if ~sum(medoids==i)

for m\_i = 1:length(medoids)

sim\_(m\_i) = d\_mat(medoids(m\_i),i);

end

[~,I] = min(sim\_);

indx = randi([1 length(I)]);

% In case if there is more than one same minimal value

I = I(indx);

dt\_pts = clusters{I};

dt\_pts(end+1) = i;

clusters{I} = dt\_pts;

end

end

% Iteration beings

max\_epochs = 3;

% Step 2

for epoch = 1:max\_epochs

% Step 2.1

% Recompute the medoids in each cluster

for m\_i = 1:length(medoids)

temp = clusters{m\_i};

% Include also the medoid of this cluster in

% its dataset while re-estimating medoid

temp(end+1) = medoids(m\_i);

new\_medoid = estimate\_medoid(temp);

% if existing medoid is the same as estimated medoid

% then make no change else swap them.

if medoids(m\_i) ~= new\_medoid

clusters{m\_i} = temp(temp~=new\_medoid);

medoids(m\_i) = new\_medoid;

end

end

% Step 2.2 Compute distance between medoid of one cluster and

% non-medoids of other clusters

total\_dtpts = 1:m;

non\_medoids = setdiff(total\_dtpts,medoids);

for nm\_i = 1:length(non\_medoids)

cost = zeros(1,3);

for m\_i = 1:length(medoids)

cost(m\_i) = d\_mat(m\_i,non\_medoids(nm\_i));

end

[~,ob\_medoid] = min(cost);

ex\_medoid = get\_medoid(clusters,non\_medoids(nm\_i));

% if non-medoid dt\_pt's existing cluster is the same

% as obtained, then make no change, else make a swap

if ob\_medoid ~= ex\_medoid

% Now assign the non\_medoid dt\_pt to new cluster

% ob\_medoid

temp = clusters{ob\_medoid};

temp(end+1) = non\_medoids(nm\_i);

clusters{ob\_medoid} = temp;

% Then delete the non\_medoid dt\_pt from old cluster

% to avoid redundancy

temp = clusters{ex\_medoid};

clusters{ex\_medoid} = temp(temp~=non\_medoids(nm\_i));

end

end

end

% Merge medoids with its corresponding cluster bins

for m\_i = 1:length(medoids)

temp = clusters{m\_i};

temp(end+1) = medoids(m\_i);

clusters{m\_i} = temp;

end

end

function medoid = get\_medoid(clusters,dt\_pt)

for i = 1:length(clusters)

temp = clusters{i};

if sum(temp==dt\_pt)==1

medoid = i;

break;

end

end

end

function medoid = estimate\_medoid(cluster)

d\_mat = get\_sim\_matrix(cluster);

d\_mat = sum(d\_mat,2);

[~,I] = min(d\_mat);

% In case if there is more than one same minimal value

indices = randi([1 length(I)]);

medoid = cluster(I(indices));

end

function d\_mat = get\_sim\_matrix(X)

m = size(X,1);

d\_mat = zeros(m,m);

for d1 = 1:m

for d2 = 1:m

if d1 ~= d2

% Manhattan distance

% d\_mat(d1,d2) = sum(abs(X(d1,:)-X(d2,:)));

% Euclidean distance

d\_mat(d1,d2) = sqrt(sum((X(d1,:)-X(d2,:)).^2));

end

end

end

end