Digraph Clustering by Marginal Propagation

CPS 521-321 Duke University
Fall 2022

Haoran Zhang[1]

[1] Duke University, USA

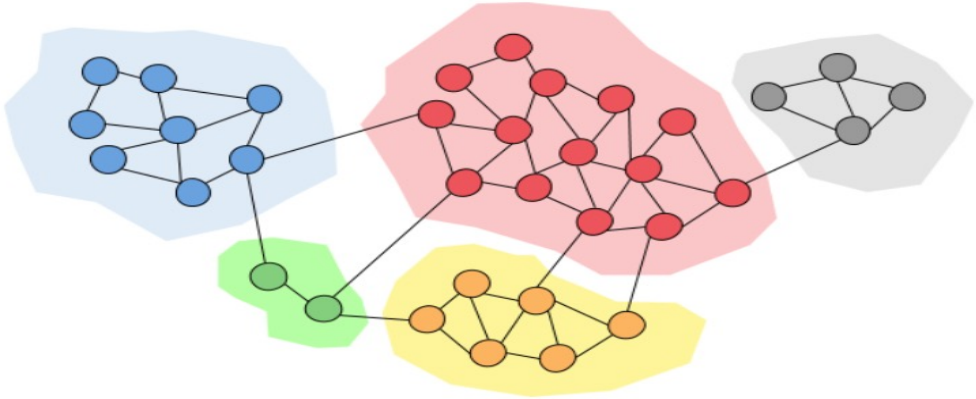
Outline

Introduction

Community within graphs features both **stronger similarities and differences** within and between other communities.

Introduction

Community within graphs features both **stronger similarities and differences** within and between other communities.



Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by “**Marginal Propagation”**

I assume that ……

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

I assume that ……

If diameter equals to **D**, and node i still can't reach j after 1, 2, … **M**, ..., D steps

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

I assume that ……

If diameter equals to **D**, and node i still can't reach j after 1, 2, … **M**, ..., D steps

Then node i is very **unlikely** to be of same community with node j

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

I assume that ……

If diameter equals to **D**, and node i still can't reach j after 1, 2, … **M**, ..., D steps

Then node i is very **unlikely** to be of same community with node j

The **closer** M is to D, the **more unlikely** the membership will be the same

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

I assume that ……

If diameter equals to **D**, and node i still can't reach j after 1, 2, … **M**, ..., D steps

Then node i is very **unlikely** to be of same community with node j

The **closer** M is to D, the **more unlikely** the membership will be the same

And the **closer** node k is to node i, the **more unlikely** same membership, too

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

I assume that ……

If diameter equals to **D**, and node i still can't reach j after 1, 2, … **M**, ..., D steps

Then node i is very **unlikely** to be of same community with node j

The **closer** M is to D, the **more unlikely** the membership will be the same

And the **closer** node k is to node i, the **more unlikely** same membership, too

Therefore we predict the community from **Marginal** nodes by **Propagation**!

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

I assume that ……

If diameter equals to **D**, and node i still can't reach j after 1, 2, … **M**, ..., D steps

Then node i is very **unlikely** to be of same community with node j

The **closer** M is to D, the **more unlikely** the membership will be the same

And the **closer** node k is to node i, the **more unlikely** same membership, too

Therefore we predict the community from **Marginal** nodes by **Propagation**!

proportional to degrees

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by “**Marginal Propagation”**

It is divided into five steps:

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

It is divided into five steps:

1. Remove the self-loops, isolated nodes

Algorithm 1: Remove the self-loops, isolated nodes in the graph (MATLAB)

Data: Graph Adjacency Matrix
Result: Self-loops, isolated nodes free adjacency matrix
(1) Load the data and obtain adjacency matrix as wells as graph object

```
A = Problem.A;
G = digraph(A);
```

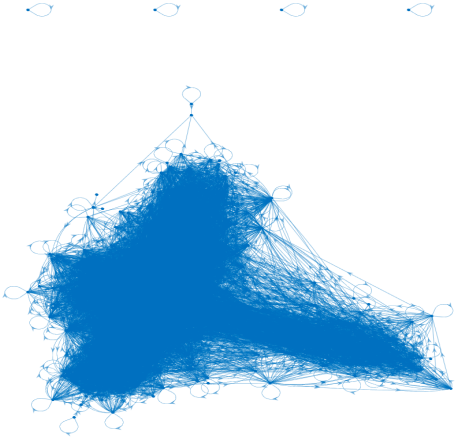
(2) Remove the self-loops for all the nodes

```
diag = diag(diag(A));
A = A - diag;
```

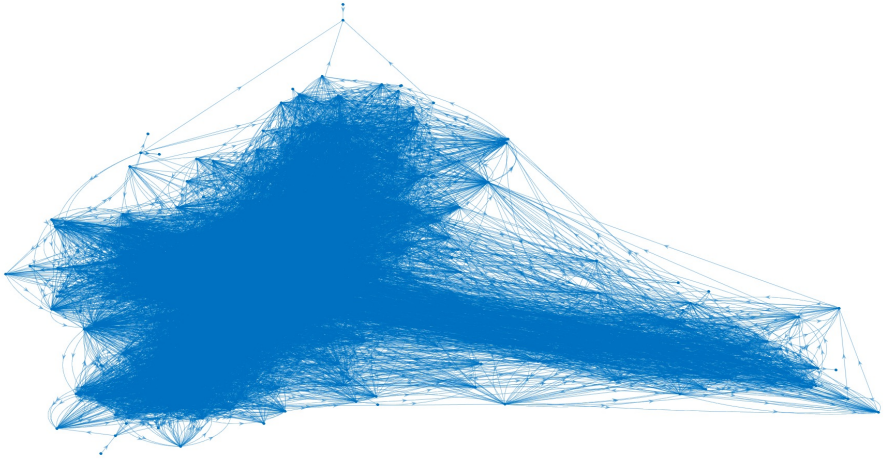
(3) Remove the isolated nodes (indegree + outdegree = 0)

```
outdeg = indegree(dig_A);
indeg = outdegree(dig_A);
mixdeg = indeg + outdeg;
zero_pos = find(~mixdeg);
G = rmnode(G, zero_pos);
A = adjacency(G);
```

Introduction & Method



Introduction & Method



Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

It is divided into five steps:

1. Remove the self-loops, isolated nodes
2. Sum up the step-N (less or equal to the **diameter**) adjacency matrix

Algorithm 2: Generate connectivity adjacency matrix after N steps/hops (MATLAB)

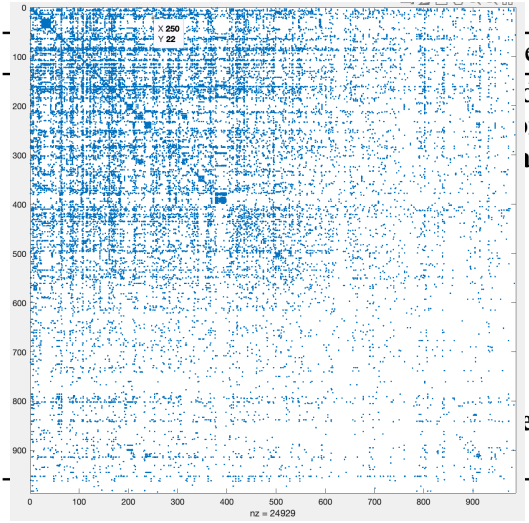
Data: Adjacency matrix without isolated nodes from **Algorithm 1**
Result: Pairwise connectivity matrix from one to N step/hops
(1) Calculate the connectivity matrix after N steps/hops

```
A_org = A;
A_sum = A_org;
while N>0 do
    A = A*A_org;
    A_sum = A_sum + A;
    N = N - 1;
end
```

(2) Get the graph corresponding to the matrix

```
G = digraph(A_sum);
```

Introduction & Method



ency matrix after N steps/hops (MATLAB)

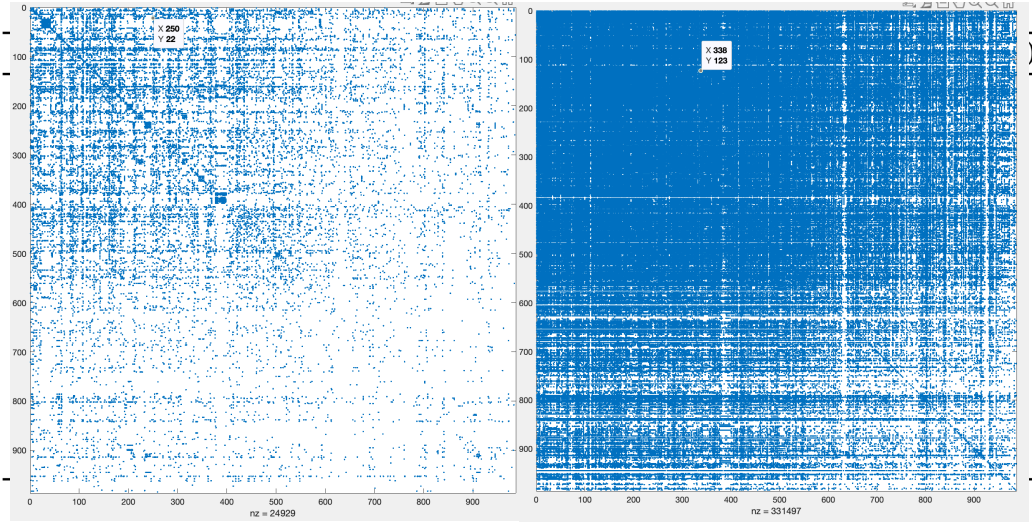
d nodes from **Algorithm 1**

m one to N step/hops

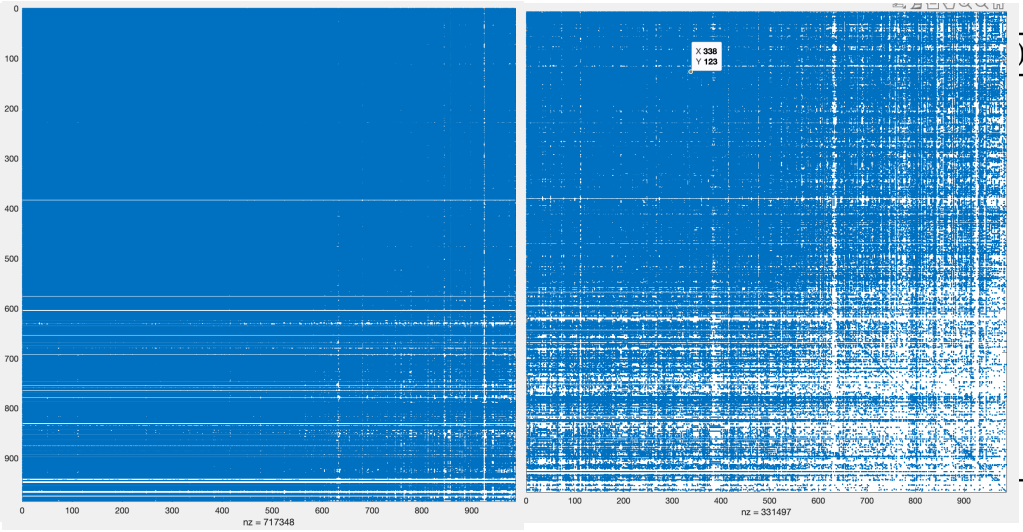
fter N steps/hops

e matrix

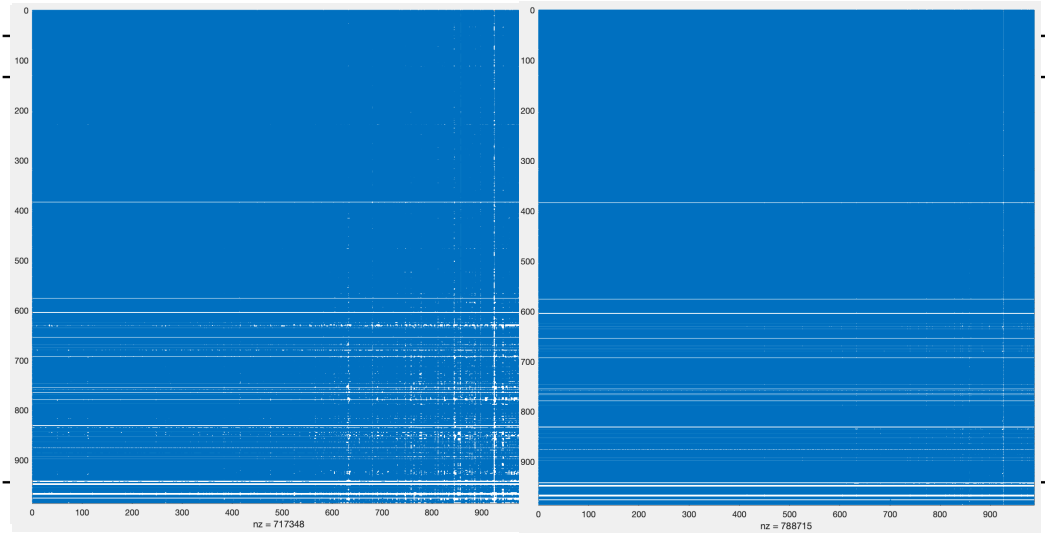
Introduction & Method



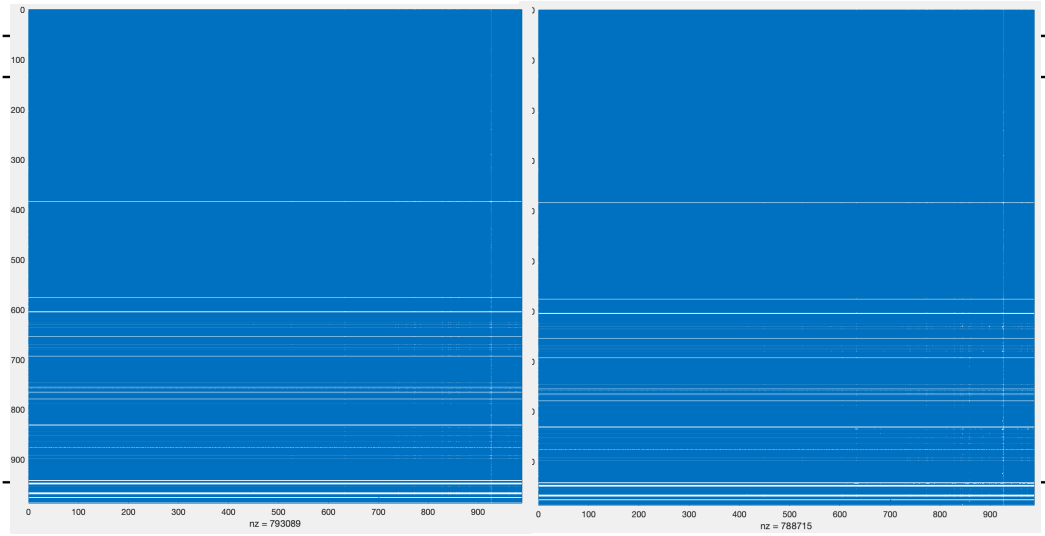
Introduction & Method



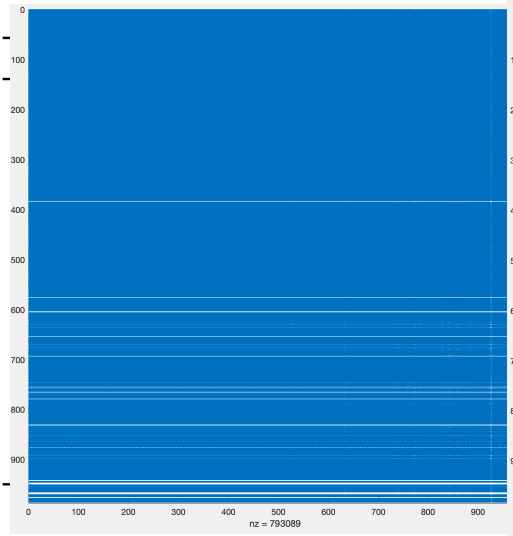
Introduction & Method

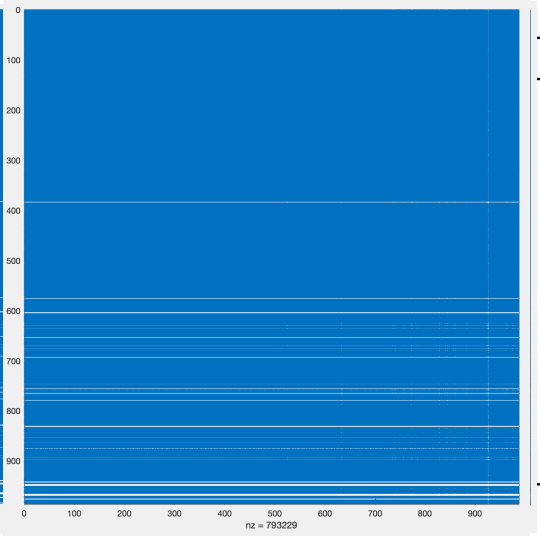


Introduction & Method

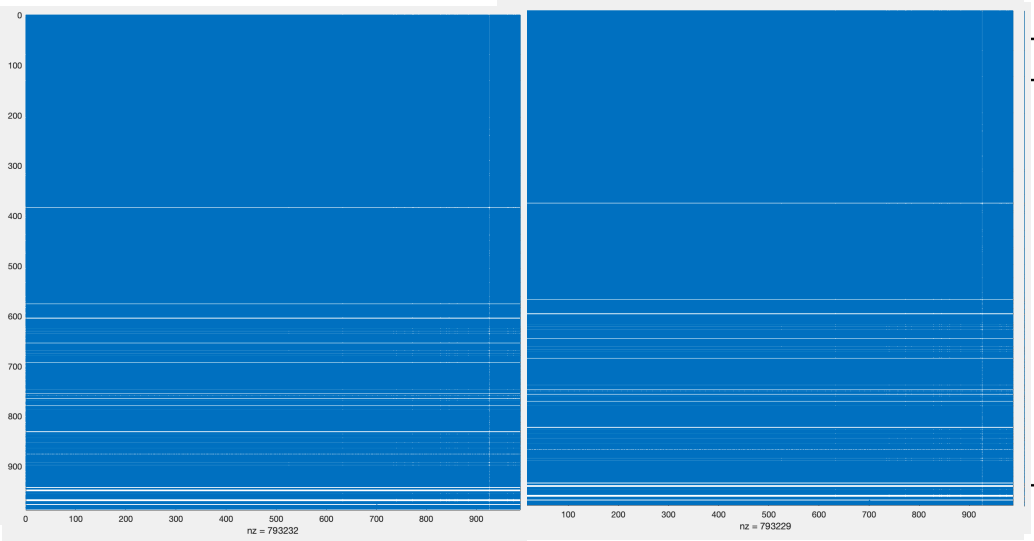


Introduction & Method





Introduction & Method



Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

It is divided into five steps:

1. Remove the self-loops, isolated nodes
2. Sum up the step-N (less or equal to the **diameter**) adjacency matrix
3. Obtain zero values in both degree and whole matrix (after select best N)

Algorithm 3: Find the positions of zeros in both types of degree and whole matrix

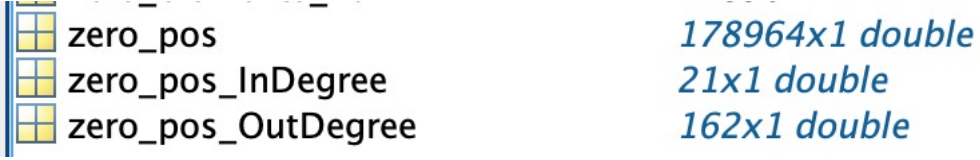
Data: Pairwise connectivity matrix from one to N steps/hops from **Algorithm 2**
Result: Positions of zero values in degrees and whole matrix

```
A_Connectivity = (A_sum ~= 0);
A_Connectivity_graph = digraph(A_Connectivity);
InDegree = outdegree(A_Connectivity_graph);
OutDegree = indegree(A_Connectivity_graph);
zeros = find(~A_Connectivity);
```

Algorithm 3: Find the positions of zeros in both types of degree and whole matrix

Data: Pairwise connectivity matrix from one to N steps/hops from **Algorithm 2**
Result: Positions of zero values in degrees and whole matrix

```
A_Connectivity = (A_sum ~= 0);
A_Connectivity_graph = digraph(A_Connectivity);
InDegree = outdegree(A_Connectivity_graph);
OutDegree = indegree(A_Connectivity_graph);
zeros = find(~A_Connectivity);
```



Introduction & Method

```
>> zero_pos_InDegree.'

ans =

  Columns 1 through 11

   525   633   735   740   757   772   774   785   828   840   845

  Columns 12 through 21

   857   861   883   923   925   926   961   964   974   977
```

zero_pos	178964x1 double
zero_pos_InDegree	21x1 double
zero_pos_OutDegree	162x1 double

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

It is divided into five steps:

1. Remove the self-loops, isolated nodes
2. Sum up the step-N (less or equal to the **diameter**) adjacency matrix
3. Obtain zero values in both degree and whole matrix (after select best N)
4. Filter unique mutually disconnected nodes

Algorithm 4: Find unconnected nodes indexes after N steps/hops (MATLAB)

```
Data: Indexes of zeroes from Algorithm 3
Result: Mutual disconnected nodes indexes after N steps/hops
(1) Create all zeros matrix with same shape of A
Disc_matrix = zeros(length(A_1), length(A_1));
(2) Calculate the x and y coordinate of zero node
for i = 1:length(zero_pos) do
    x = ceil(zero_pos(i)/length(A_1));
    y = zero_pos(i) - (x-1)*length(A_1);
    if ismember(y, zero_pos_InDegree) | ismember(x, zero_pos_OutDegree) then
        continue;
    else
        Disc_matrix(x,y) = 1;
    end
end
(3) Keep only symmetric nodes in the matrix (i.e. when the disconnectivity is mutual)
total_zeros = sum(Disc_matrix, 'all');
Disc_matrix_sym = floor((Disc_matrix + Disc_matrix.')/2);
L = tril(Disc_matrix_sym);
L_total_num = sum(L, 'all');
Disc_node_id = find(L);
```

Algorithm 5: Record the positions of disconnected nodes and obtain unique node id list (MATLAB)

Data: Unconnected node indexes from **Algorithm 4**
Result: Indexes of unique nodes that are mutually unconnected

```
unique_disconnected_node = [0];
index = 1;
for i = 1:length(Dis_node_id) do
    x = ceil(Dis_node_id/length(A));
    y = Dis_node_id(i) - (x-1)*length(A);
    unique_disconnected_node(index) = x;
    unique_disconnected_node(index+1) = y;
    index = index + 2;
end
unique_nodes = unique(unique_disconnected_node);
```

The total number of communities assumed to be *c*:

```
c = length(unique_nodes);
```

Algorithm 5: Record the positions of disconnected nodes and obtain unique node id list (MATLAB)

```
unique_disconnected_node =

  Columns 1 through 11

   203   415   433   464   559   568   617   632   683   724   728

  Columns 12 through 22

   730   818   820   834   880   885   902   905   924   940   944

  Column 23

   985
```

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

It is divided into five steps:

1. Remove the self-loops, isolated nodes
2. Sum up the step-N (less or equal to the **diameter**) adjacency matrix
3. Obtain zero values in both degree and whole matrix (after select best N)
4. Filter unique mutually disconnected nodes
5. Label the marginal nodes as different communities and propagate

Algorithm 6: Assign nodes into communities and update their probabilities (MATLAB)

Data: Unique disconnected nodes from **Algorithm 5** and adjacency matrix A
Result: Probabilities of each node belong to Community from 1 to c
(1) Store the probabilities of each nodes belonging to community in a $N \times C$ matrix:

```
c_label = 1;
c_prob = zeros(length(A), length(unique_nodes));
for y = 1:length(A) do
    if ismember(y, unique_nodes) then
        c_prob(y, c_label) = 1.0
    else
        c_prob(y, :) = c_prob(y, :) + 1.0/length(unique_nodes);
    end
end
```

Introduction & Method

(2) Add community probabilities to neighbours proportional to its out-degree

```
EPOCHS = E;
for epochs = 1:EPOCHS do
    for node = length(A):-1:1 do
        sucs_nodes = successor(G, node);
        infl_prob = c_(node, :)
        for i = 1:length(sucs_nodes) do
            sucs_node = sucs_nodes(i);
            if ismember(sucs_node, unique_nodes) then
                continue;
            else
                c_prob(sucs_node, :) = c_prob(sucs_node, :) +
                 1.0*infl_prob/length(sucs_nodes)
            end
        end
    end
    for node = length(A):-1:1 do
        c_prob(node, :) = c_prob(node, :) / sum(c_prob(node, :));
    end
end
```

Introduction & Method

198	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
199	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0

```
y = 203 Assign community label 1
y = 415 Assign community label 2
y = 433 Assign community label 3
y = 464 Assign community label 4
y = 559 Assign community label 5
y = 568 Assign community label 6
y = 617 Assign community label 7
y = 632 Assign community label 8
y = 683 Assign community label 9
y = 724 Assign community label 10
```

```
y = 728 Assign community label 11
y = 730 Assign community label 12
y = 818 Assign community label 13
y = 820 Assign community label 14
y = 834 Assign community label 15
y = 880 Assign community label 16
y = 885 Assign community label 17
y = 902 Assign community label 18
y = 905 Assign community label 19
y = 924 Assign community label 20
y = 940 Assign community label 21
y = 944 Assign community label 22
y = 985 Assign community label 23
```

212	0.0435	0.0435	0.0435	0.0435	0.04
213	0.0435	0.0435	0.0435	0.0435	0.04

Introduction & Method

198	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
199	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
200	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
201	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
202	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
203	1	0	0	0	0	0	0	0	0	
204	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
205	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
206	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
207	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
208	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
209	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
210	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
211	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
212	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0
213	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0.0435	0

(3) We normalize all the probabilities to make their sums equal to one

$[c\_prob\_max, c\_label\_pred]$ = max(c_prob, [], 2);

counts = hist(c_label_pred, $[1 : C]$);

(3) We normalize all the probabilities to make their sums equal to one

$[c\_prob\_max, c\_label\_pred]$ = max(c_prob, [], 2);
counts = hist(c_label_pred, $[1 : C]$);

```
>> Community_Prob_A(1,:)

ans =

  Columns 1 through 8

    0.0416    0.0630    0.0432    0.0355    0.0500    0.0354    0.0356    0.0630

  Columns 9 through 16

    0.0384    0.0372    0.0649    0.0500    0.0477    0.0421    0.0334    0.0376

  Columns 17 through 23

    0.0570    0.0349    0.0331    0.0441    0.0391    0.0305    0.0426
```

(3) We normalize all the probabilities to make their sums equal to one

$[c\_prob\_max, c\_label\_pred]$ = max(c_prob, [], 2);
counts = hist(c_label_pred, $[1 : C]$);

```
>> Community_Prob_A(1,:)

ans =

  Columns 1 through 8

    0.0416    0.0630    0.0432    0.0355    0.0500    0.0354    0.0356    0.0630

  Columns 9 through 16

    0.0384    0.0372    0.0649    0.0500    0.0477    0.0421    0.0334    0.0376

  Columns 17 through 23

    0.0570    0.0349    0.0331    0.0441    0.0391    0.0305    0.0426
```

Introduction & Method

Community within graphs features both **stronger similarities and differences** within and between other communities.

This project approaches this problem differently as a **semi-supervised learning task**.

where the labels are generated from the graph by "**Marginal Propagation"**

It is divided into five steps:

1. Remove the self-loops, isolated nodes
2. Sum up the step-N (less or equal to the **diameter**) adjacency matrix
3. Obtain zero values in both degree and whole matrix (after select best N)
4. Filter unique mutually disconnected nodes
5. Label the marginal nodes as different communities and propagate
6. Calculate modularity scores on predicted communities

$$Modularity: \quad Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \, \delta \left(C_i, C_j \right)$$

$$Modularity: \quad Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij})\, \delta\,(C_i, C_j)$$

Q being the modularity score and m the total number of edges

$$Modularity: \quad Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij})\, \delta\,(C_i, C_j)$$

Q being the modularity score and m the total number of edges

P_{ij} the expected number of edges between vertices i and j in the null model

$$Modularity: \quad Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij})\, \delta\,(C_i, C_j)$$

Q being the modularity score and m the total number of edges

P_{ij} the expected number of edges between vertices i and j in the null model

The δ-function returns one if vertices i and j are in the same community ($C_i = C_j$), zero otherwise.

Result

The test dataset for this project is:

Result

The test dataset for this project is:

Table 1: Statistics for the test dataset Email EU Core

Dataset Statistics			
Nodes	1005	Edges in largest SCC	24729 (0.967)
Edges	25571	Average clustering coefficient	0.3994
Number of communities	42	Number of triangles	105461
Nodes in largest WCC	986 (0.981)	Fraction of closed triangles	0.1085
Edges in largest WCC	25552 (0.999)	Diameter (longest shortest path)	7
Nodes in largest SCC	803 (0.799)	90-percentile effective diameter	2.9

Result

Predicted Community Distribution:

Result

Predicted Community Distribution:

Table 2: Community Label distribution between the predicted and ground truth for Email EU Core

Dataset	label	#	label	#	Dataset	label	#	label	#	label	#	label	#
Email EU Core	**1**	263	**13**	8	Ground Truth	**1**	111	**13**	27	**25**	12	**37**	3
	2	135	**14**	8		**2**	107	**14**	26	**26**	10	**38**	3
	3	122	**15**	7		**3**	91	**15**	25	**27**	10	**39**	2
	4	82	**16**	5		**4**	56	**16**	24	**28**	9	**40**	1
	5	80	**17**	4		**5**	54	**17**	22	**29**	9	**41**	1
	6	42	**18**	4		**6**	49	**18**	19	**30**	8	**42**	0
	7	40	**19**	4		**7**	39	**19**	18	**31**	8		
	8	39	**20**	3		**8**	34	**20**	15	**32**	6		
	9	38	**21**	2		**9**	31	**21**	13	**33**	6		
	10	35	**22**	2		**10**	29	**22**	13	**34**	5		
	11	35	**23**	1		**11**	29	**23**	13	**35**	4		
	12	27				**12**	28	**24**	12	**36**	4		

Result

Predicted Community Distribution:

Modularity Score!

Table 2: Community Label distribution between the predicted and ground truth for Email EU Core

Dataset	label	#	label	#	Dataset	label	#	label	#	label	#	label	#
Email EU Core	**1**	263	**13**	8	Ground Truth	**1**	111	**13**	27	**25**	12	**37**	3
	2	135	**14**	8		**2**	107	**14**	26	**26**	10	**38**	3
	3	122	**15**	7		**3**	91	**15**	25	**27**	10	**39**	2
	4	82	**16**	5		**4**	56	**16**	24	**28**	9	**40**	1
	5	80	**17**	4		**5**	54	**17**	22	**29**	9	**41**	1
	6	42	**18**	4		**6**	49	**18**	19	**30**	8	**42**	0
	7	40	**19**	4		**7**	39	**19**	18	**31**	8		
	8	39	**20**	3		**8**	34	**20**	15	**32**	6		
	9	38	**21**	2		**9**	31	**21**	13	**33**	6		
	10	35	**22**	2		**10**	29	**22**	13	**34**	5		
	11	35	**23**	1		**11**	29	**23**	13	**35**	4		
	12	27				**12**	28	**24**	12	**36**	4		

Abundant Features

Abundant Features

Marginal Propagation outputs a N × C matrix with:

Abundant Features

Marginal Propagation outputs a N × C matrix with:

N equals to the total number of nodes

Abundant Features

Marginal Propagation outputs a N × C matrix with:

N equals to the total number of nodes
C equals to the number of communities

Abundant Features

Marginal Propagation outputs a N × C matrix with:

N equals to the total number of nodes
C equals to the number of communities

The probabilities vector can be used to calculate:

Abundant Features

Marginal Propagation outputs a N × C matrix with:

N equals to the total number of nodes
C equals to the number of communities

The probabilities vector can be used to calculate:

Membership Similarity

Abundant Features

Marginal Propagation outputs a N × C matrix with:

N equals to the total number of nodes
C equals to the number of communities

The probabilities vector can be used to calculate:

Membership Similarity & Community Similarity

Similarity Formula

$$Membership\ Similarity\ M_S = V_1 \cdot V_2$$

Similarity Formula

$$Membership\ Similarity\ M_S = V_1 \cdot V_2$$

$$Community\ Similarity = \frac{\sum_i^{C_i} p_j \cdot \sum_j^{C_j} p_i}{N(C_j) \cdot N(C_i)}$$

Similarity Formula

$$Membership\ Similarity\ M_S = V_1 \cdot V_2$$

$$Community\ Similarity = \frac{\sum_i^{C_i} p_j \cdot \sum_j^{C_j} p_i}{N(C_j) \cdot N(C_i)}$$

Overlapping Community Detection

Limitations

The number of communities outputted is restricted by the diameter being selected in Algorithm 2 (D/2)

Limitations

The number of communities outputted is restricted by the diameter being selected in Algorithm 2 (D/2)

If the diameter of the graph is either too large or too short, this method's performance will be severely depreciated

Limitations

The number of communities outputted is restricted by the diameter being selected in Algorithm 2 (D/2)

If the diameter of the graph is either too large or too short, this method's performance will be severely depreciated

Communities can be dominated after long iterations

References

[1] Wikipedia contributors. Directed graph — Wikipedia, the free encyclopedia, 2022. [Online; accessed 12-December-2022].

[2] Arzum Karataş and Serap Şahin. Application areas of community detection: A review. In *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pages 65–70, Dec 2018.

[3] Stuart A. Rice. The identification of blocs in small political bodies. *American Political Science Review*, 21(3):619–627, 1927.

[4] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[5] Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, Quan Z. Sheng, and Philip S. Yu. A comprehensive survey on community detection with deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2022.

[6] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019.

Bibliography II

[7] Tiancheng Liu, Dimitris Floros, Nikos Pitsianis, and Xiaobai Sun. Digraph clustering by the bluered method. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7, 2021.

[8] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[9] E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Phys. Rev. Lett.*, 100:118703, Mar 2008.

[10] Dimitris Floros. Agreement or discrepancy measures. dec 2022.

[11] Santo Fortunato. Community detection in graphs. *Complex Networks and Systems Lagrange Laboratory, ISI Foundation, Viale S. Severo 65, 10133, Torino, I-ITALY.*, 486, feb 2010.

Acknowledgement

Appreciate Prof. Xiaobai Sun who has been supportive throughout the course.

Acknowledgement

Appreciate Prof. Xiaobai Sun who has been supportive throughout the course

Appreciate Janice for providing me with support and continuous encouragement

Acknowledgement

Appreciate Prof. Xiaobai Sun who has been supportive throughout the course

Appreciate Janice for providing me with support and continuous encouragement

Thanks for your attention