Timetabling by Coloring and Clustering by Neuronal Networks

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
MS (by Dissertation) Degree

by

Pranav Niturkar

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

May, 2022

Supervisor: Dr. Collins Assisi

Certificate

This is to certify that this dissertation entitled Timetabling by Coloring and Clustering by Neuronal Networks towards the partial fulfilment of the MS (by Dissertation) Degree at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Pranav Niturkar at Indian Institute of Science Education and Research under the supervision of Dr. Collins Assisi, Dr. Soumen Maity and Dr. M. S. Madhusudhan during the academic year 2021-2022.

Dr. Collins Assisi

Committee:

Dr. Collins Assisi

Dr. Soumen Maity

Dr. M. S. Madhusudhan

This thesis is dedicated to the
Shoonya Vāsini, Trinetrini, Sarva Janani:
Linga Bhairavi

Declaration

I hereby declare that the matter embodied in the report entitled Timetabling by Coloring and Clustering by Neuronal Networks are the results of the work carried out by me at the Indian Institute of Science Education and Research, Pune, under the supervision of Dr. Collins Assisi and the same has not been submitted elsewhere for any other degree.

Pranav

Pranav Niturkar

Acknowledgments

Many thanks to my supervisor, Dr. Collins Assisi, for providing the much needed support at crucial junctures. Thanks to Dr. Soumen Maity for allowing me to learn at my own pace. A special thanks to the Computational Neurobiology Lab for absorbing my nuisance. I also wish to express my gratitude to the int.PhD committee, Dr. Kaneenika Sinha and others, for their patience and understanding.

Thanks to all those who tolerated my loong or irrelevant e-mails. *Many thanks to my parents and sisters for their encouragement, without which this would not be possible.* And last but not the least, thanks to my friends or enemies and random IISERites for the engaging conversations which kept me going.

Abstract

Designing a university course timetable requires assigning events(course lectures, tutorials, colloquia) to locations(on/off-line classrooms) and time slots, while avoiding clashes - for example, lectures of two courses that a particular student subscribes to cannot run concurrently. In designing the timetable, we can consider the events(or classes) as the vertices of a graph and the conflicts between them as edges between the corresponding vertices. This formulation allows us to state the university timetabling problem as a graph vertex coloring problem. Vertices with the same colour, in any colouring of such a graph, can give us the set of events that can share the same time-slot, while different colours represent groups of events that must be assigned different time-slots.

We propose using the dynamics of neuronal networks to solve the graph colouring problem. We will assign neurons to each vertex of the constraint graph and interactions between them will be inhibitory. Inhibitory neurons compete with each other, when one fires, it prevents those connected to it from firing. Therefore, vertices with the same color do not compete and can fire synchronously. In earlier work[1], this idea was used to arrive at solutions of the Sudoku puzzle which can also be mapped to a vertex coloring problem. Here we propose using this approach to solve *a particular instance* of the university timetabling problem.

Contents

Chapter 1

Introduction

1.1 Designing University Timetables

A University Course Timetable can be thought of as an assignment of lectures, tutorials, or exams of courses to a given number of rooms and time-slots satisfying a set of constraints. A timetabling problem can contain a host of different constraints making it complex(or *beautiful*!).

Aimed at generating new approaches to *the timetabling problems* by attracting users from all areas of research, the Second International Timetabling Competition(ITC2007) was organised in 2007, by a group of timetabling researchers from different European Universities. Timetabling can be classified into two categories: Examination Timetabling and Course Timetabling. Further, course timetabling can be based on curriculum or post enrolment[2].

In recent times, significant advancements have been made in research areas by attracting multi-disciplinary approaches and comparing them on a common ground. Another important goal of the competition is to close the existing gap between research and practice within this important area of operational research. Although for the sake of the competition, all aspects of the *real world* problem are not included, recent developments are taken into consideration providing significant depth and complexity[3].

1.1.1 Types of Constraints

The constraints of the timetabling problem can be split into two types: **hard** and *soft*. **Hard** constraints are mandatory, since a teacher or a student can not be present at two different locations at the same time! If all **hard** constraints are satisfied, we have a feasible solution. Whereas *soft* constraints can help us define a criteria from which a timetable could be considered good, enhancing the experiences of the people who will have to use it[2].

Some examples of *soft* constraints:

1. Students should not attend three or more classes consecutively in a day;
2. Students should not attend just one class in a day;
3. They should not be required to attend in the last time-slot of each day.

Note: We will be focusing only on the **hard** constraints.

1.1.2 Two-Stage Approach

Timetabling at IISER Pune is done in two stages, pre-registration and registration.

(1.) Students do pre-registration for the courses they *want* in the upcoming semester.

According to those students choices, the courses are then grouped. The grouping must either satisfy all choices or when that does not happen, minimize the dis-satisfaction.

(2.) These groups are then permuted among the available time-slots for timetabling.

Note: Once the groups are formed, students *can not* register for two courses from the same group.

1.2 IISER Pune Timetabling

	Monday	Tuesday	Wednesday	Thursday	Friday
9 am					
10 am					
11 am					
12 noon					
1 pm			L U N C H		
2 pm					
3 pm					
4 pm					
5 pm					

Fig.1: IISER Timetable consists of 8 x 5 = 40 time-slots.

Fig. *1* shows that IISER Pune has 40 time-slots and since each course has three(or two) lectures per week, we need 13(or 14) groups of courses, occupying 13 x 3 = 39 time-slots + 1 for seminar or colloquium.

Given the students pre-registration(enrolment) data, we wish to group the courses into k(= 13 or 14) groups, such that
(a.) each student gets to choose all the courses that one has done pre-registration for, or
(b.) minimize the dis-satisfaction, or overlap between courses chosen from the same group by students.

The Conflicts Matrix:

A problem instance is given by

- lectures $l = l_1, l_2, ..., l_n$ for n courses,
- time-slots $t = t_1, t_2, ..., t_{40}$,
- set of students $s = s_1, s_2, ..., s_{|s|}$,
- rooms $r = r_1, r_2, ..., r_{|r|}$,
- room features $f = f_1, f_2, ..., f_{|f|}$,
- and room capacity $c(r_i)\ \forall\ r_i \in\ r$.

Attends matrix **A** has $a_{ij} = 1$ if student s_i is due to attend lecture l_j, 0 otherwise;
Room Features matrix **T** has $t_{ij} = 1$ if room r_i has feature f_j, 0 otherwise;
Lecture Features matrix **M** has $m_{ij} = 1$ if lecture l_i requires feature f_j, 0 otherwise;
Lecture Availability matrix **S** has $s_{ij} = 1$ if lecture l_i can be assigned time-slot t_j, 0 otherwise;

and now the *Conflicts matrix* **C** can be defined by $c_{ij} = 1$ if

$$\exists\ s_k \in\ s : (a_{li} = 1 \wedge\ a_{lj} = 1)$$

i.e., two lectures share a student in common; or

$$\nexists\ t_l \in\ t : (s_{il} = 1 \wedge\ s_{jl} = 1)$$

i.e., two lectures have mutually exclusive subsets of available time-slots; or
two lectures require the same room, i.e., they satisfy room capacity and feature requirement;
and 0 otherwise.
(Refer [2], pg. 199-200, for more details)

1.3 Grouping the Courses

There's an underlying *Graph Coloring Problem* for most timetabling problems. This allows us to apply the concepts of graph coloring from the literature of theoretical computer science to the timetabling problem.

A coloring with k colors, which is a solution for the graph colouring problem of the constraints graph, corresponds to k groups of courses satisfying all students choices.

Why? Since assignment of the same color implies absence of an edge between corresponding nodes, which in turn means that no student has chosen both.

One way to approach the problem, is to construct the constraints graph from given student choices and try coloring it. If an algorithm gives us the desired number of groups, great! Otherwise we apply some reductions until we get the size of grouping we want and later merge the output of multiple iterations in a way that minimizes dis-satisfaction.

The Constraints Graph: We will construct *the constraints graph* where each course is represented by a node and there is an edge between two nodes corresponding to two courses, if both of them have been chosen by at least one student.

Further, if multiple students have chosen both the courses, then the number of students doing so is the weight of the edge between corresponding nodes of the courses.

1.3.1 Graph Coloring

Definition 1.3.1. *Let G =(V, E) be a graph; V is a set of n vertices and E is a set of m edges. We wish to assign a color $c(v) \in \{1, 2, ..., k\}$ to each vertex $v \in V$, such that: $c(v) \neq c(u), \forall(v, u) \in E$; with minimum k(number of colors).*

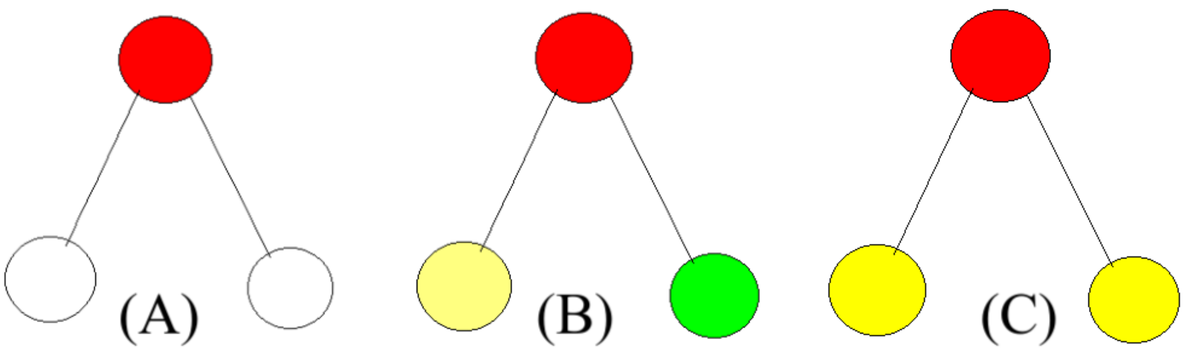

Fig.2: Coloring of a graph with 3 nodes.

***Fig.**2* demonstrates, what we mean by a graph coloring.
(A) shows the node in the centre colored Red. Which implies that its neighbors can't be colored Red and need a different color.
(B) shows one way of coloring the other two nodes. But since there's no edge between them, both of them can get same color(anything other than Red).
(C) shows the coloring with minimum(2) colors as desired.

1.3.2 Clustering

For $k \geqslant 3$, k colour-ability of a graph is known to be **NP-complete**. And determining the existence of a feasible solution for our timetabling problem with k time-slots, is equivalent to the graph k-colouring problem. Hence, our time-tabling problem is also **NP-hard**.

But a problem being **NP-hard** or practically unsolvable, doesn't mean we're done; it still needs to be solved *somehow!* If not optimally, then at least approximately to whatever extent possible.

Another way of approaching the timetabling problem is to look at the complement of the constraint graph and detect *clusters* in it. In the complement graph, each color group(independent set of nodes) from the original graph, will form an all-to-all connected sub-graph(clique).

But clique detection is **hard**, hence we relax the notion to a cluster and proceed. We tried the Newman's clustering algorithm and a neural network algorithm to cluster the constraint network.

Chapter 2

Greedy Coloring

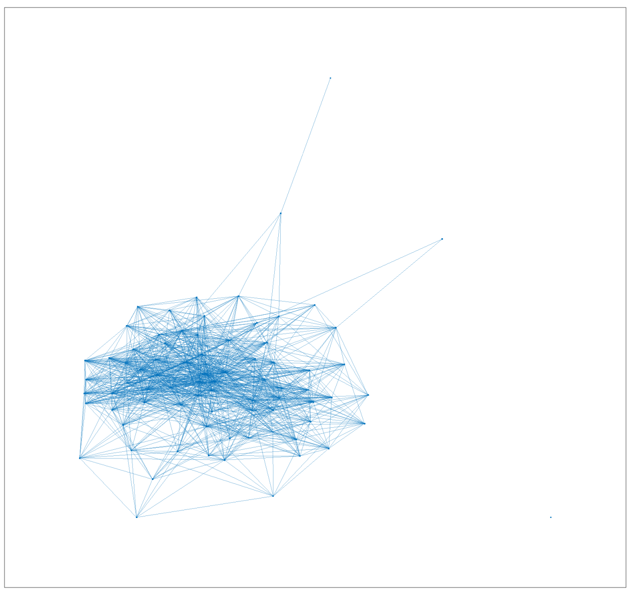

Fig.3: The constraints graph(G) from student choices, January 2022 semester, IISER.

The Greedy algorithm takes vertices one by one, according to some(random) ordering and assigns the first available colour to each vertex. A solution produced by the Greedy algorithm need not be optimal, but it can be made optimal for any graph(yes, *any* graph!) by choosing the*correct* ordering[2]. We will use the ordering obtained by sorting vertices by their degrees in a descending order. We apply this to the constraints graph(See ***Fig.****3*) to obtain a grouping of courses(See ***Fig.****5*).

1	BI3214	16	CH3214	29	EC3214	43	HS3213	50	MT3214	64	PH3214
2	BI3224	17	CH3224	30	EC3224	44	HS3223	51	MT3224	65	PH3224
3	BI3234	18	CH3234	31	EC3234	45	HS3234	52	MT3234	66	PH3234
4	BI3244	19	CH3243	32	EC3243	46	HS3244	53	MT3244	67	PH3244
5	BI3254	20	CH3253	33	EC3253	47	HS3253	54	MT3254	68	PH3253
6	BI3264	21	CH4214	34	EC3264	48	HS3264	55	MT3264	69	PH3264
7	BI3274	22	CH4224	35	EC3274	49	HS3274	56	MT5214	70	PH3273
8	BI3284	23	CHM410	36	EC3284			57	MT5224	71	PH4213
9	BI3294	24	CHM420	37	EC3293			58	MT5234	72	PHY342
10	BI3413	25	CHM423	38	EC4213			59	MTH411	73	PHY420
11	BI3423	26	CHM428	39	EC4224			60	MTH422	74	PHY422
12	BI3433	27	CHM433	40	EC4234			61	MTH423	75	PHY434
13	BI3444	28	CHM442	41	ECS442			62	MTH424	76	PHY441
14	BI5214			42	ECS456			63	MTH426	77	PHY463
15	BIO463									78	PHY464
										79	PHY557

Fig.4: The list of courses offered in January 2022 semester at IISER, Pune.

1	1		24		41			45	51		79	
2	2										66	74
3	3		19						52		68	
4	4								50		70	
5	5				40				59		69	
6	6		27		29	37		46	57	58	67	
7	7							43				
8	8				36						76	
9	9		17								64	75
10	10							47			73	
11	11	14	28		39				56	60		
12	12		22		30	35					65	
13	13											
14	15											
15			16	25	32				53	63		
16			18	23	38			49	55	62	72	78
17			20									
18			21		31			48	54		77	
19			26		33	34	42	44	61		71	
	BIO		**CHM**		**ECS**			**HSS**	**MTH**		**PHY**	

Fig.5: A grouping given by greedy algorithm for the constraint graph G(seen before), indices stand for courses listed in ***Fig.4***.

2.1 Reductions

Applying the greedy coloring algorithm to our constraint graph(G) with decreasing degree sequence, we get a 19 coloring giving us 19 groups, but we need k(= 13).

Note: A k(= 13) coloring *may exist* but we couldn't get it through decreasing degree sequence for Greedy algorithm.

Since we couldn't satisfy all students choices, we now turn our attention towards minimizing dis-satisfaction. We can apply reductions to the graph with some minimization rule up to a point where the reduced graph gives us a k(= 13) grouping.

Chromatic Number and Its Bounds

Definition 2.1.1. *The* Chromatic Number *of a graph* $\chi(G)$ *is the smallest* k *for which* G *can be colored using* k *colours.*

Some well known bounds on the chromatic number are as follows[3]:
Let $\Delta(G)$ = maximum degree of any vertex in G
and $\delta(G)$ = minimum degree of any vertex in G.

$$\chi(G) \leqslant \Delta(G) + 1 \text{ follows from the greedy algorithm;}$$

$$\chi(G) \leqslant \delta(G)$$

for simple connected graphs which are not complete and don't have odd cycle;

$$\text{and } \chi(G) \leqslant 4 \text{ for planar graphs.}$$

2.2 Enumeration

Suppose the highest weight of an edge in the(constraint) graph is 185. But that doesn't mean the graph has edges of all weights from 1 to 185, hence we need to enumerate all edge weights in a variable Enum for applying reduction. However, one can observe that if there are 100 edges of weight 1 and 40 edges of weight 2, then 1 x 100 = 100 is greater than 80 = 2 x 40. We would like to delete minimum value first and we do so by sorting Enum in an ascending order according to the product of weight value and its number of occurrences.

Note: Enum2 is not necessarily better than weight value-wise deletion Enum in general, since the sum of weights of all deleted edges may vary. Refer to ***Fig.****6* for flowchart of Enum(/2) Merge Algorithm.

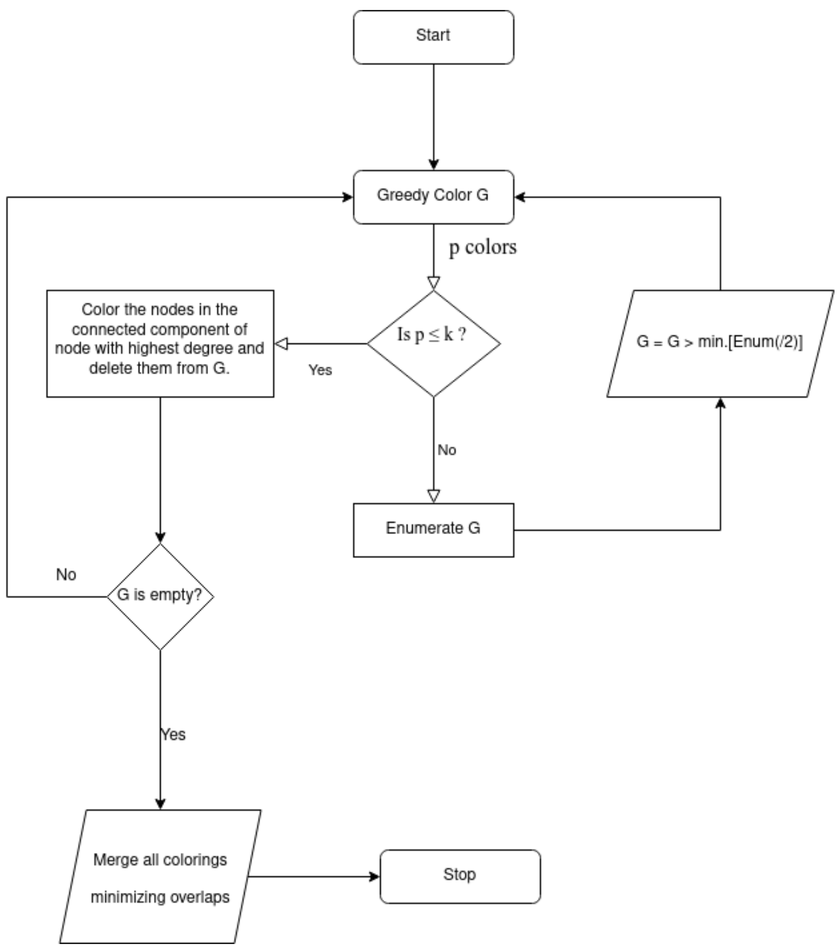

Fig.6: Flowchart for Enum(/2) Merge Algorithm.

Reductions can be done in the following ways: Keep deleting lower weight edges(See ***Fig.****7* *and* ***Fig.****8*) until

1. all connected components of the graph have k(= 13) vertices left, which can then be colored with k(= 13) colors, one each vertex. Later combining outputs from multiple iterations, minimizing overlaps; or

2. every connected component has maximum degree of vertices to be k-1(= 12), then $\chi(G) \leqslant \Delta(G) + 1$ gives a coloring of size $\leqslant$ k(= 13) from the greedy algorithm. Later combining outputs from multiple iterations, minimizing overlaps; or

3. the greedy algorithm with decreasing degree sequence gives a k(= 13) coloring. Later combining outputs from multiple iterations, minimizing overlaps.

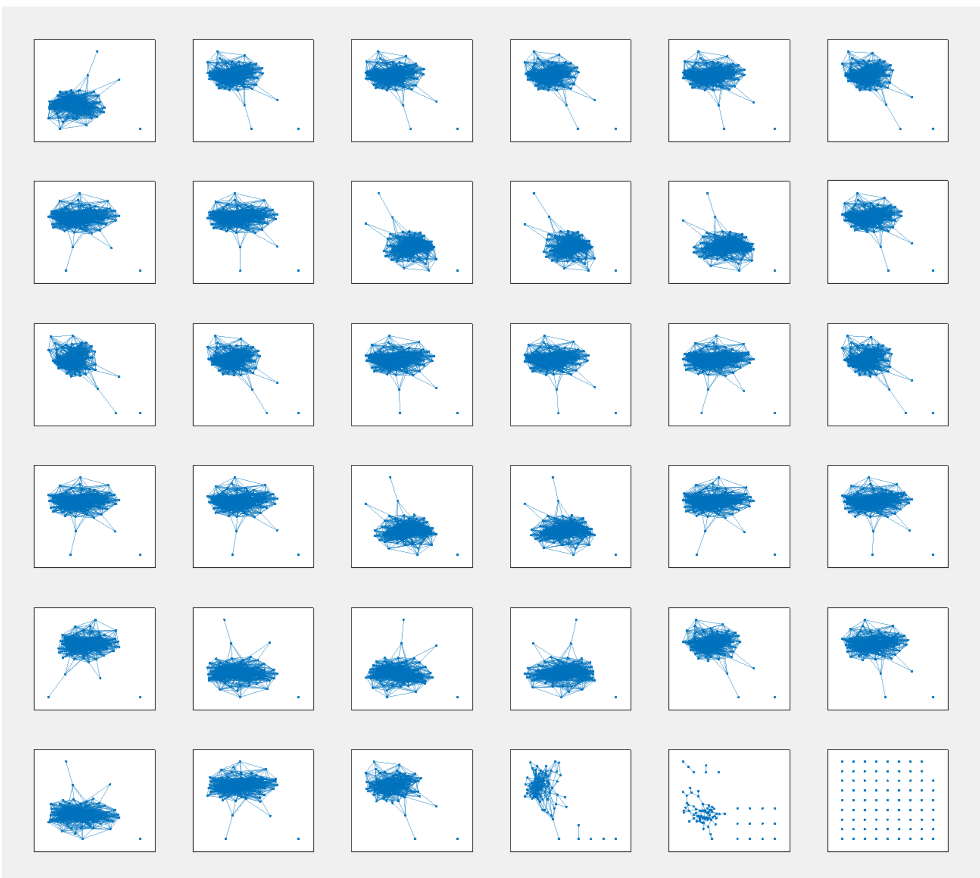

Fig.7: Reduction of the constraint graph G from top left to fewer edges, deleting all edges of a particular weight each iteration, according to the Enum2 list.

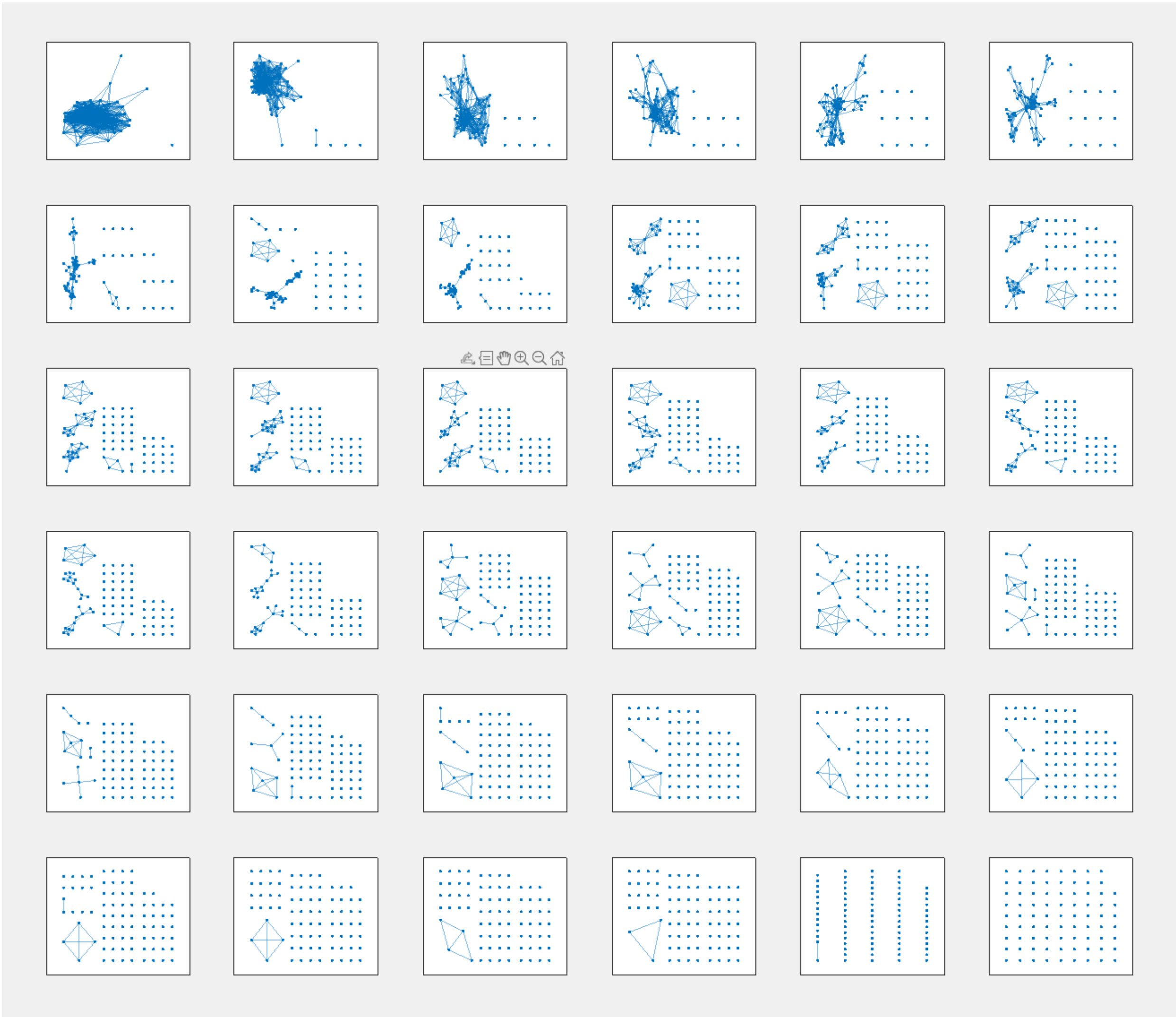

Fig.8: Reduction of the constraint graph G from top left to fewer edges, deleting all edges of a particular weight each iteration, according to the Enum list.

Or

Apply greedy coloring algorithm to the main constraint graph and then merge the groups in a way that minimizes dis-satisfaction of constraints until only k(= 13) groups are left(call it *Greedy Merge Algorithm*). Refer to ***Fig.****9* below for the flowchart of algorithm.

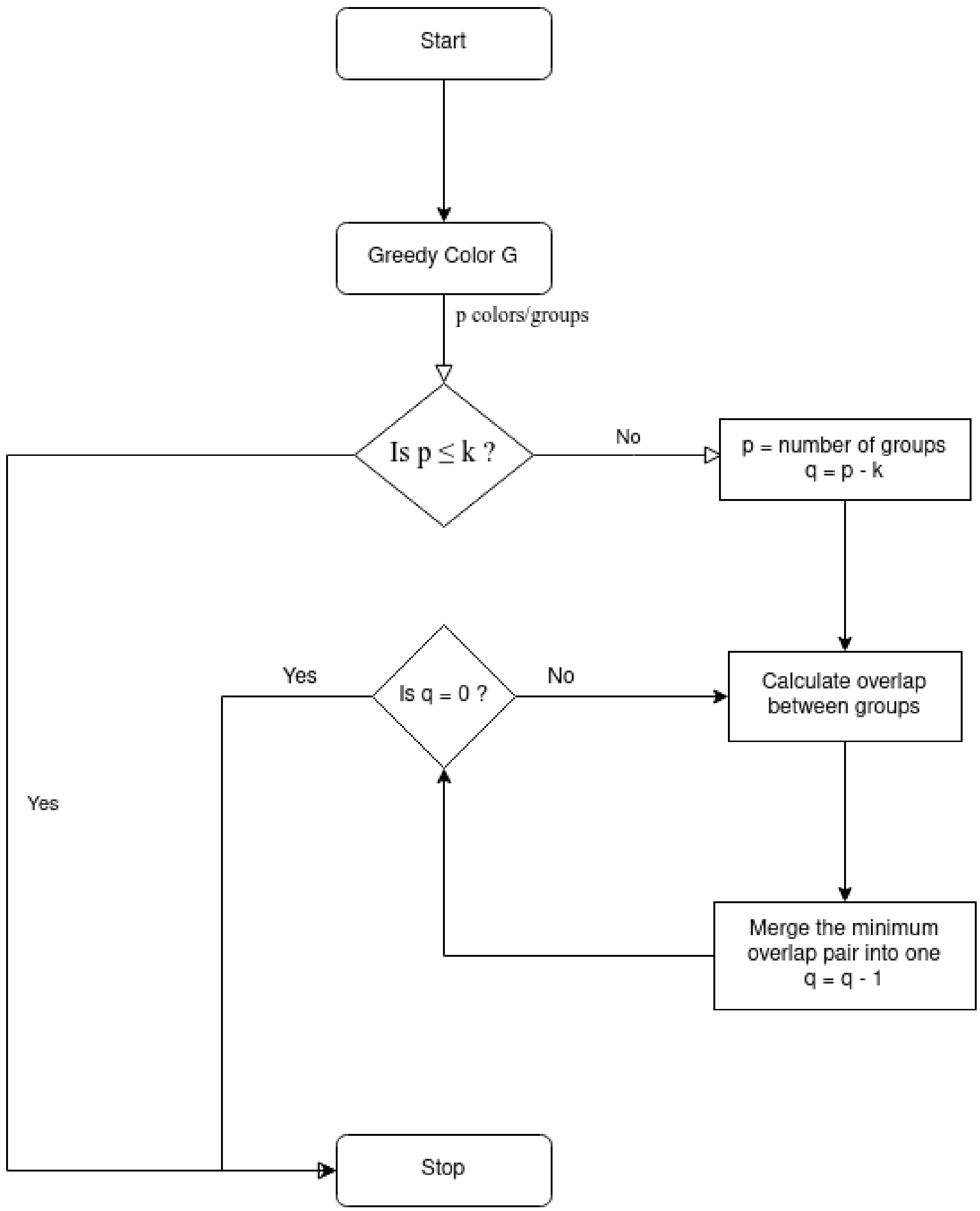

Fig.9: Flowchart for Greedy Merge Algorithm.

2.3 D.T.F.

Definition 2.3.1. D.T.F., *Distance To Feasibility of a grouping is the sum of the weights of the edges connecting courses in the same group(or same colored vertices).*

The grouping which was implemented for January 2022 semester at IISER Pune(See ***Fig.****11*) has DTF = 67, while the grouping obtained by the Greedy Merge algorithm for the same(See ***Fig.****12*) has DTF = 31. As the following chart(in ***Fig.****10*) shows, *the* ***Greedy Merge Algorithm*** *gives groupings of lowest DTF values* and hence is recommended.

Greedy Colors	Students	Courses	Semester	Constraints		Greedy Merge		Enum Merge		Enum2 Merge	
				Weighted	**Edges**	**13**	**14**	**13**	**14**	**13**	**14**
16	337	79	**Aug2019**	3481	749	7	3	49	49	71	51
19	423	82	**Aug2020**	6818	1291	87	58	385	321	1026	871
18	389	85	**Aug2021**	6069	1202	59	36	368	339	567	596
17	364	92	**Jan2020**	5218	1200	48	34	118	80	357	476
--	---	--	**Jan2021**	----	----	--	--	---	---	---	---
19	346	79	**Jan2022**	4743	1084	48	31	256	110	458	478

Fig.10: DTF values of the groupings for different semesters by different algorithms.

1	1	15	18	23	31		49		54	57	76	
2	2	10			41		45		62		70	79
3	3		28		40				53		77	
4	4		22		42		46		50			
5	5		24		33				59		73	
6	6		21		35		47		55		65	
7	7				29		43	48	58		69	72
8	8		27		36	37			52		68	
9	9		17	26	34		44		61		64	71
10	11	14			39				56	60	78	
11	12		16		30				51			
12	13		25						63		66	74
13			19		38						67	
14			20		32						75	
	BIO		**CHM**		**ECS**		**HSS**		**MTH**		**PHY**	

Fig.11: The grouping which was in implementation for January 2022 semester.

1	1		24		41			45	51		79	
2	2	7						43			66	74
3	3		19						52		68	
4	4	10						47	50		70	73
5	5				40				59		69	
6	6		27		29	37		46	57	58	67	
7	8				36						76	
8	9		17								64	75
9	11	14	28		39				56	60		
10	12		22		30	35					65	
11	13		16	25	32				53	63		
12	15		20	21	31			48	54		77	
13			18	23	38			49	55	62	72	78
14			26		33	34	42	44	61		71	
	BIO		**CHM**		**ECS**			**HSS**	**MTH**		**PHY**	

Fig.12: The Greedy Merge grouping for January 2022 semester.

Hamming Distance

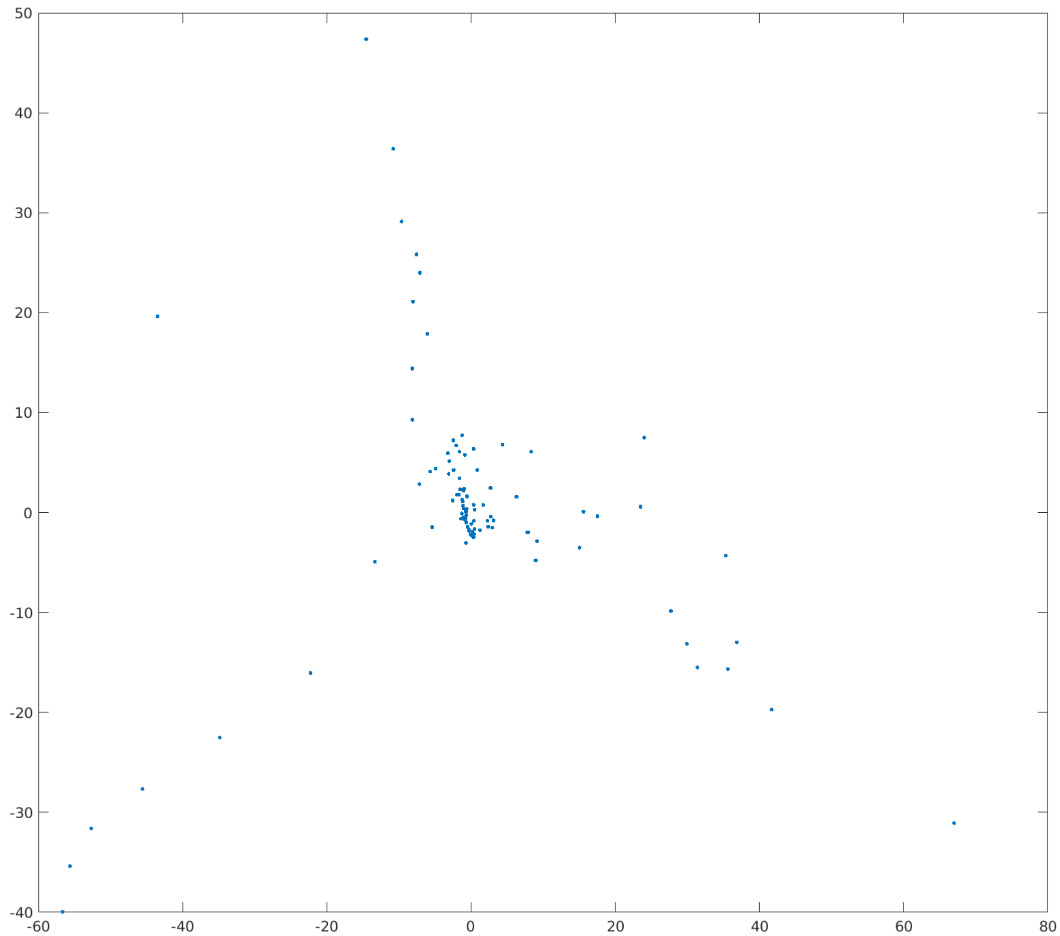

Fig.13: A 2-D MDS plot of hamming distance between courses.

From the pre-registration data, we can obtain a matrix where each row represents a student and each column, a course. If student i has picked course j in the pre-registration, then the(i, j)th entry of matrix is set to 1, otherwise 0. From this matrix, we can compute the hamming distance between any two courses by computing the hamming distance between the corresponding columns. Intuitively, the farther away any two courses are in the 2-D MDS plot(See ***Fig.****13*), the more reliability there is in putting them in the same group.

Choices as Nodes

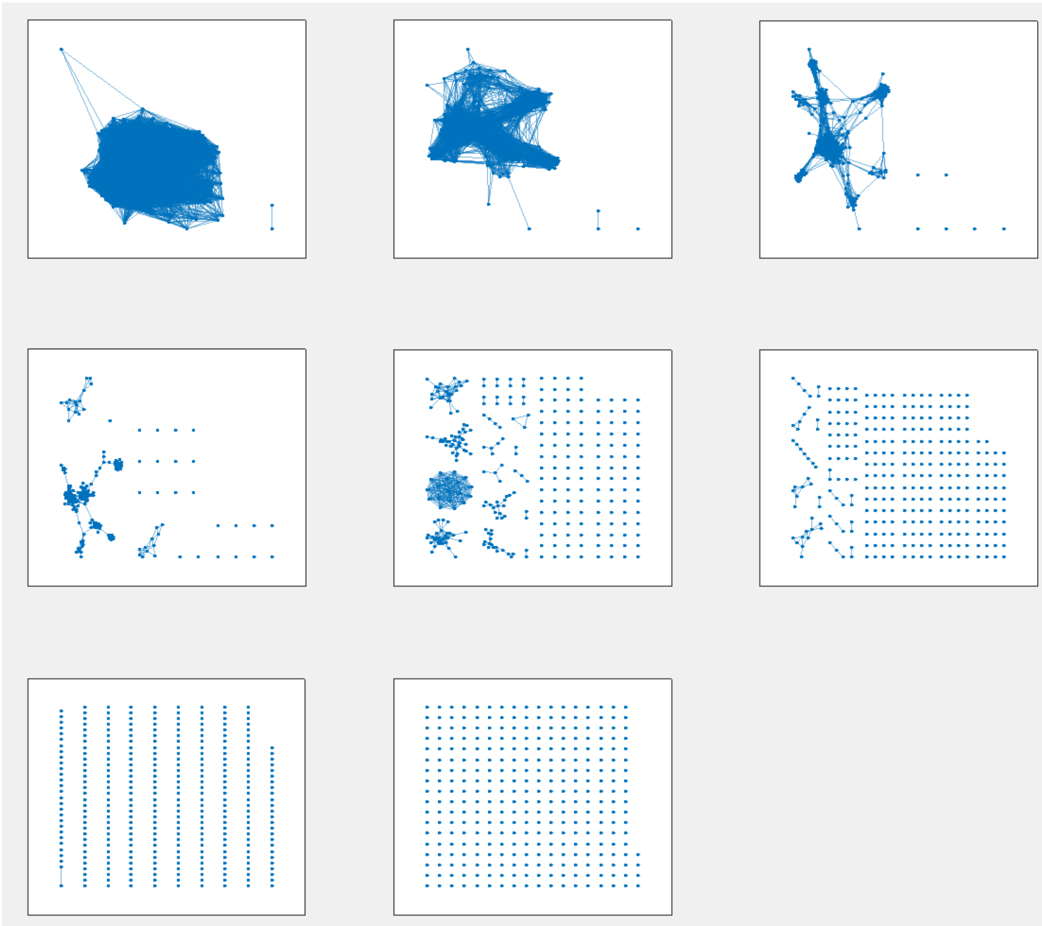

Fig.14: Reductions on the graph with distinct student choices as nodes.

So far we've looked at the constraint graph which had courses as nodes and edges indicating student(s) having them both. However, we now construct a graph in which each distinct student choice represents a node. The number of courses in common between them can be considered as weights of edges between corresponding nodes. ***Fig.****14* shows reduction applied on one such graph for January 2022 semester data of IISER Pune, where lowest weight edges are removed each iteration.

Students(choices) with larger overlap of courses will form clusters as iterations progress. We can gain insights into the spread and distribution of student choices from this graph. For instance, the occurrence of a sub-graph that seems like a clique in the middle plot(***Fig.****14*) has an interesting story to tell!

Chapter 3

Clustering

Definition 3.0.1. Clusters *in a network, are sets of nodes that have greater density of edges within themselves and there are comparatively fewer edges between nodes from different sets. See **Fig.**15 below.*

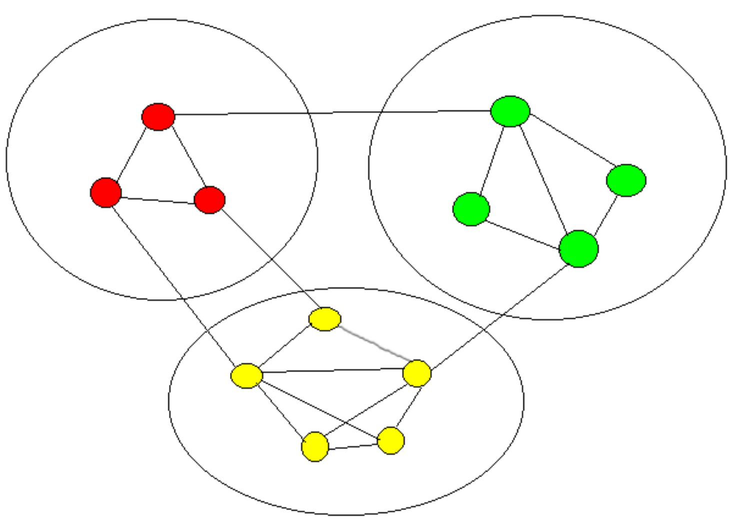

Fig.15: A network with 3 color-coded clusters.

3.1 Community Structure Detection

As it turns out, simply counting edges is not enough for capturing the intuitive concept of community structure. Quoting Newman(2006): *A good division of a network into communities is not merely one in which there are* ***few*** *edges between communities; it is one in which there are* ***fewer than expected*** *edges between communities*[5].

Definition 3.1.1. Modularity *is, up to a multiplicative constant, the difference between the number of edges within groups and*
the expected number of edges in an equivalent network with edges placed at random.

$$Q = \sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})$$

where the summation runs over all pairs of vertices i, j that fall in the same group
A_{ij} *is the adjacency matrix;*
k_i *and* k_j *are degrees of the vertices i and j, respectively*
and the total number of edges is $m = \frac{1}{2}\sum_i(k_i)$.

Q can be either positive or negative. Positive values indicate the possibility of community structure being present. One can search for community structure by looking for the partition of a network with positive(possibly large) values of modularity.

Community structure detection differs from graph partitioning problem since:

1. The number and size of the groups are determined by the network itself; and
2. The methods *may* indicate a lack of good division of the network.

3.2 Newman's Optimal Modularity

We shall now focus on the problem of finding the division of a network into just two communities, if such a division exists.

Definition 3.2.1. The Modularity Matrix $\boldsymbol{B}$ *is given by:*

$$\boldsymbol{B}_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

A_{ij} is the number of edges between vertices i and j; and
when edges are placed at random,
the expected number of edges between vertices i and j are $\dfrac{k_i k_j}{2m}$.

Now, the expression for modularity becomes

$$\mathrm{Q} = \mathbf{s}^T \ \mathbf{B} \ \mathbf{s}$$

where **s** is the column vector whose elements are given by
s_i = 1, if vertex i belongs to group 1 and s_i = -1, if it belongs to group 2.

Newman's Spectral Modularity Maximization method: For maximizing the modularity(Q), we compute the leading eigenvector of the modularity matrix(**B**) and divide the vertices in the network into two groups according to the signs of the elements in this vector, meaning if i^{th} entry is positive, i^{th} vertex goes in group one
and if j^{th} entry is negative, j^{th} vertex goes in group two.

3.3 Application to Karate Club Network

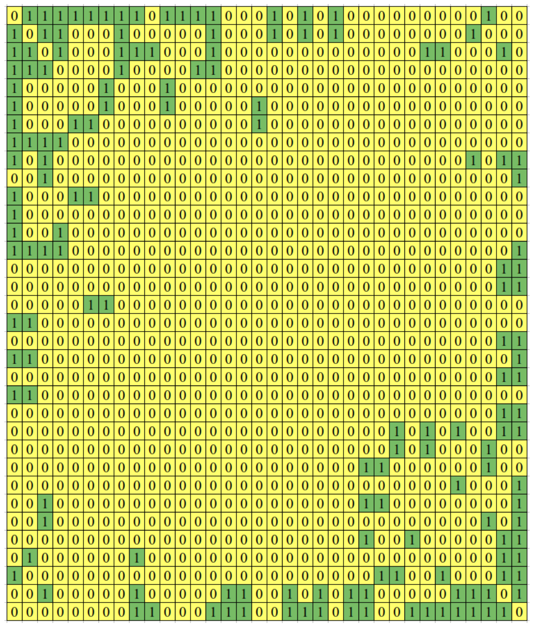

Fig.16: The adjacency matrix of the graph of the karate club network.

The *karate club* network of Zachary(with adjacency matrix in ***Fig.****16*) has become something of a standard test for community detection algorithms. It shows the pattern of friendships between the members of a karate club at an American university in the 1970s[5].

This particular example is interesting because, shortly thereafter the club split in two as a result of an internal dispute. Applying Newman's algorithm to the network, we find the division indicated by the dotted line(See ***Fig.****17*). It coincides exactly with the known division of the club in real life, indicated by the shapes of the vertices.

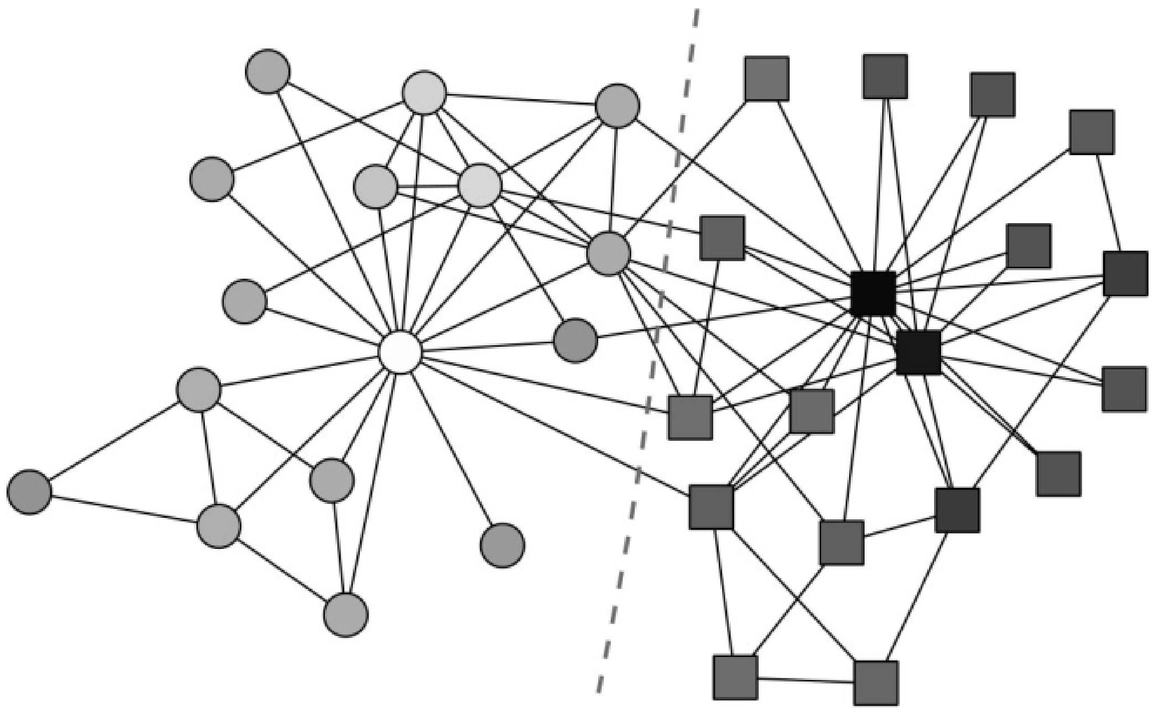

Fig.17: Application of the eigenvector-based method to the karate club network[5].

More

Newman's Clustering with γ factor

Newman's clustering algorithm has a γ factor variation for greater sensitivity to community structure. The default value of gamma is 1 and setting it to 1, we get back the earlier expression. The modularity matrix is given by:

$$\mathbf{B}_{ij} = A_{ij} - \gamma\,\frac{k_i k_j}{2m}$$

$$\textit{Note: } \sum_j B_{ij} = \sum_j A_{ij} - \frac{k_i}{2m}\,\sum_j k_j = k_i - \frac{k_i}{2m}\,2m = 0$$

γ works like a tuning knob for detection of clusters locally(more clusters, $\gamma > 1$) and globally(fewer clusters, $\gamma < 1$). For application to karate club network, see ***Fig.****18*

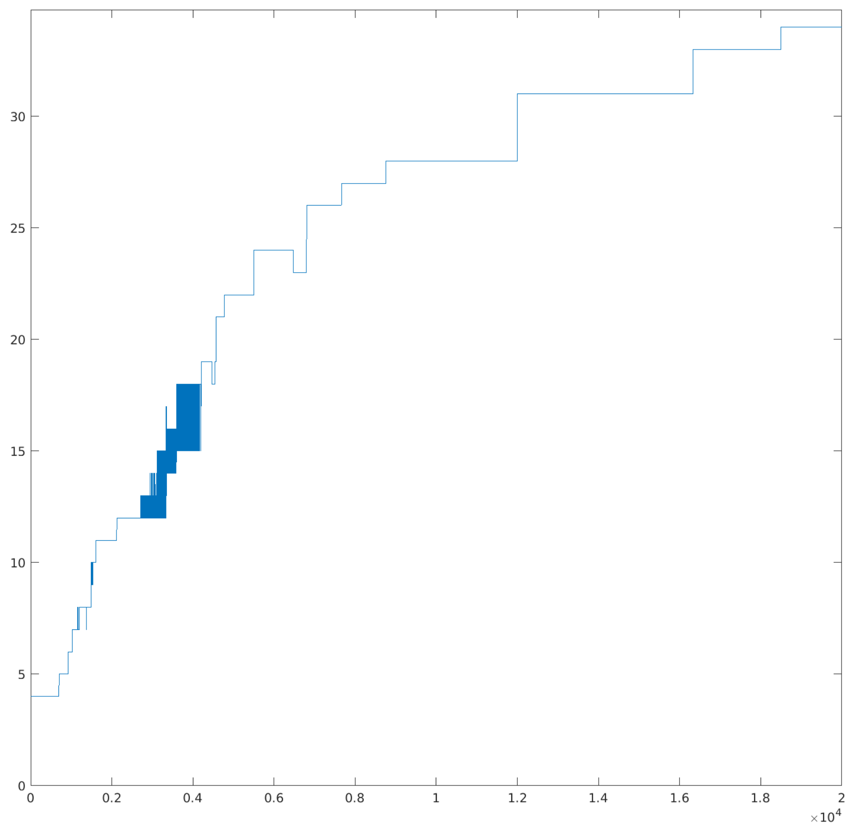

Fig.18: γ(X-axis) vs total number of groups(Y-axis).

Neighborhood Diversity

Neighborhood diversity, as the name suggests, is a classification of vertices based upon their *neighbors*, i.e., the vertices they are adjacent to. *How do we get a grouping?* We check nodes with same neighbors, including or excluding each other. In ***Fig.****19*(A), nodes 33 and 34(in **bold**) have same neighbors listed to their left(in yellow circles), the same goes for nodes 1 and 2. So we merge them with their neighbors and get groups.

(A)

15	16	19	21	23	**33**	**34**
			18	22	**1**	**2**

(B)

1	**2**	3	4	5	6	7	8	10	11	12	13	14	17	18	20	22
9	15	16	19	21	23	24	25	26	27	28	29	30	31	32	**33**	**34**

Fig.19:(A) Groups from the Karate club via neighborhood diversity and(B) 2 groups using Newman's Clustering, $\gamma = 0.5$.

Fig.*19* shows the similarities in grouping of nodes by neighborhood diversity and Newman's clustering. ***Fig.****20* shows the groups by neighborhood diversity, in the network plot.

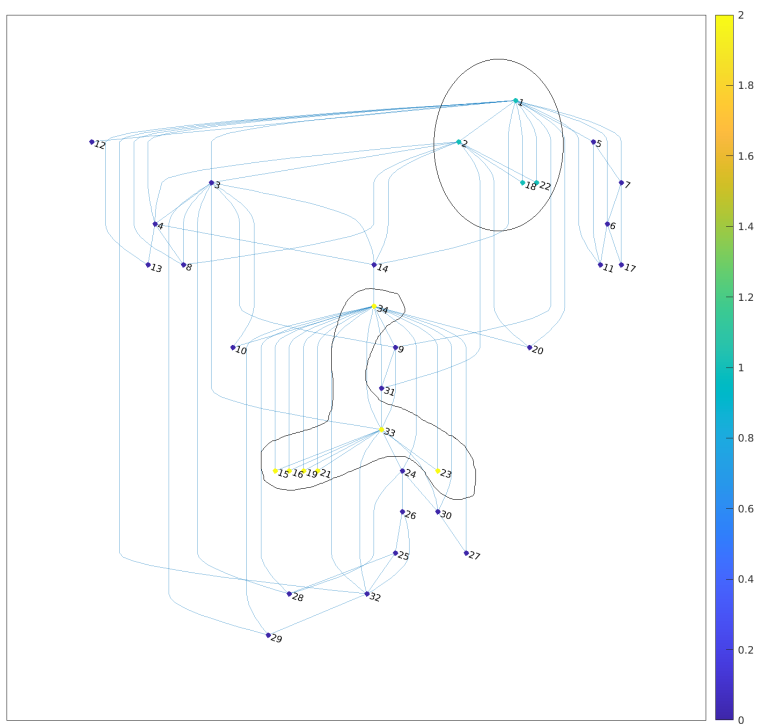

Fig.20: Neighborhood Diversity for karate club network.

Chapter 4

Competitive Neuronal Networks

A bipartite network(See ***Fig.****21*) for which minimum two colors were required to partition the graph, only one 2-coloring was possible. Simulations showed that, when we add inhibitory interactions given by adjacency matrix in ***Fig.****21*(a), the connected neurons did not fire together and nodes within the same partition did not synchronize either[1]. We introduce the *missing* excitatory interactions given by adjacency matrix in ***Fig.****21*(b).

For this network when the ratio of excitatory to inhibitory input was large, the excitation within and across groups drove all the oscillators to fire together(See ***Fig.****21*(c), bottom panel). Whereas lower ratio meaning more inhibition drives the system to chaotic firing(See ***Fig.****21*(c), top panel). And when E-I(Excitation-Inhibition) *balance* happens, we see periodic firing of 2 groups corresponding to 2 colors(See ***Fig.****21*(c), middle panel).

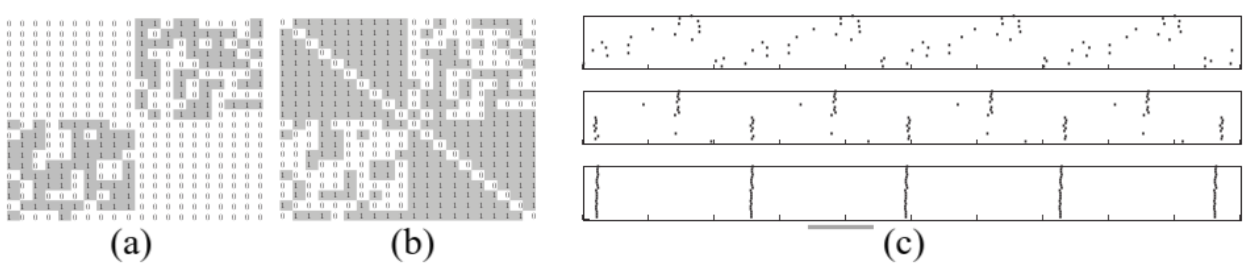

Fig.21:(a) and(b) Adjacency matrices of the inhibitory and excitatory networks;
(c) Response of the bipartite network for low excitation(top panel),
balanced one(middle panel) and for one with high excitation(bottom panel)[1].

4.1 Synchrony and Chaos

Synchrony: Let us consider an integrate-and-fire model of neurons with all connections excitatory. When a given oscillator fires, it pulls up those connected by a fixed amount(ϵ) or makes the corresponding neurons fire, if the threshold is attained or exceeded. It has been observed that for almost all initial conditions, the population evolves to a state in which all the oscillators are firing synchronously. See ***Fig.****22*.

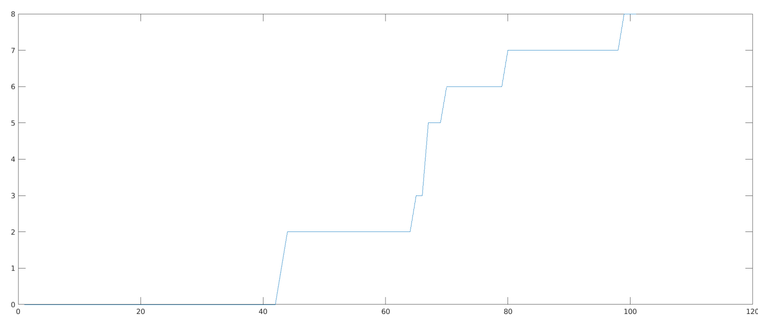

Fig.22: Total number of neurons firing(on Y-axis) vs time(on X-axis).

Chaos: We know that excitatory interactions synchronize the oscillators. Since each excitatory pulse advances the phase of the receiving oscillator towards the threshold until the spike times of both eventually coincide. However a pair of inhibitory oscillators spike at different phases which progressively separate over multiple iterations. See ***Fig.****23*.

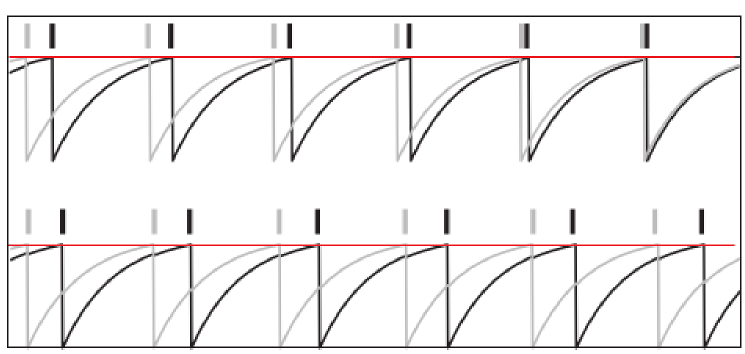

Fig.23: Two oscillators dynamics with excitatory and inhibitory connections. Lines denote the time when pulses are emitted crossing threshold(red line)[1].

4.2 E-I Balance for Karate Club Network

Consider the problem of finding clusters in the Karate Club network seen earlier. For every edge in the graph, we add an inhibitory connection. This means, when a neuron fires, those connected to it will get a downward push(negative ϵ *or* ϵ_{inh}). Whereas, for every two nodes not connected in the graph, there is an excitatory connection between them(positive ϵ *or* ϵ_{exc}). ***Fig.*** *24* shows the periodic firing of neurons in groups when E-I balance occurs.

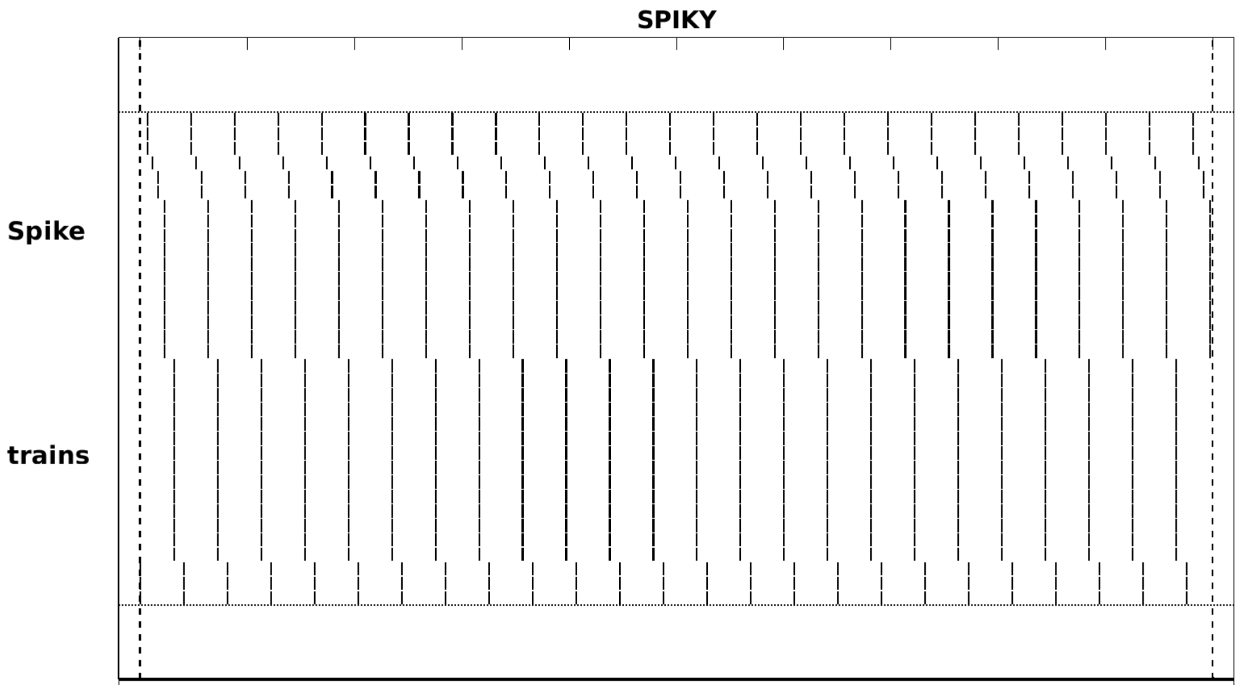

Fig.24: Raster plot for the Karate Club network with $\epsilon_{exc} = 2.2 \text{ x } 10^{-6}$ and $\epsilon_{inh} = 1.06 \text{ x } 10^{-3}$.

What γ does for Newman's algorithm(seen earlier),
ϵ_{exc} and ϵ_{inh} do for our neural network algorithm.

Increasing Excitation: With an increase in ϵ_{exc}, the number of groups decreases as larger groups absorb the smaller ones. ***Fig.****25* below demonstrates one such instance.

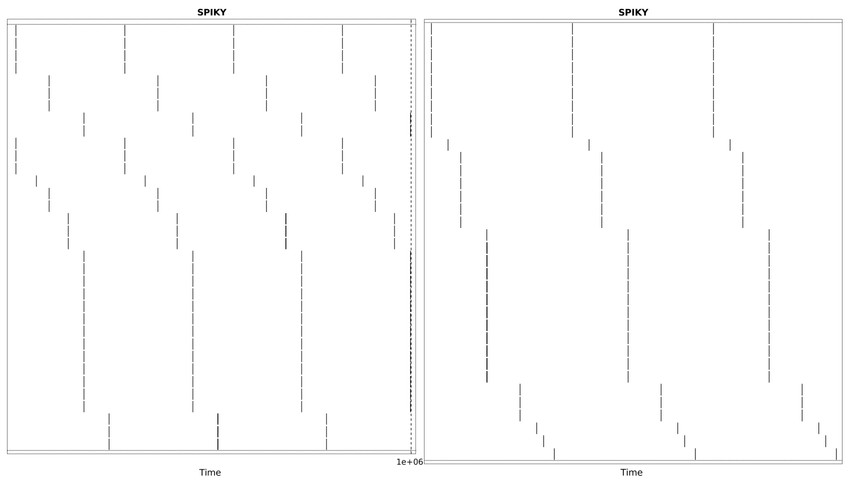

Fig.25: Raster plot for the Karate Club network with
fixed $\epsilon_{inh} = 1.05$ x 10^{-3}, $\epsilon_{exc} = 2.3$ x 10^{-6}(left) and 2.4 x 10^{-6}(right).

Increasing Inhibition: Increasing ϵ_{inh}, as one expects, the number of groups increases as larger groups break down to smaller ones. ***Fig.****26* below demonstrates one such instance.

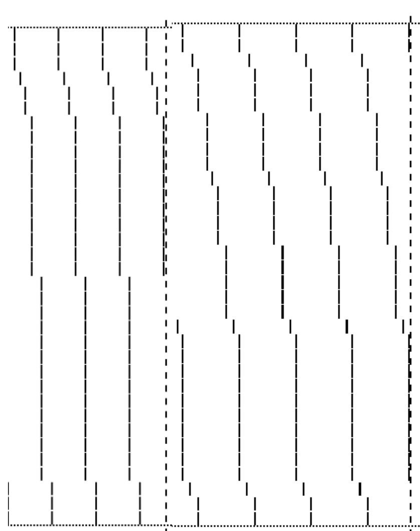

Fig.26: Raster plot for the Karate Club network with
fixed $\epsilon_{exc} = 2.2$ x 10^{-6}, $\epsilon_{inh} = 1.06$ x 10^{-3}(left) and 1.10 x 10^{-3}(right).

4.3 Network Clustering Plots

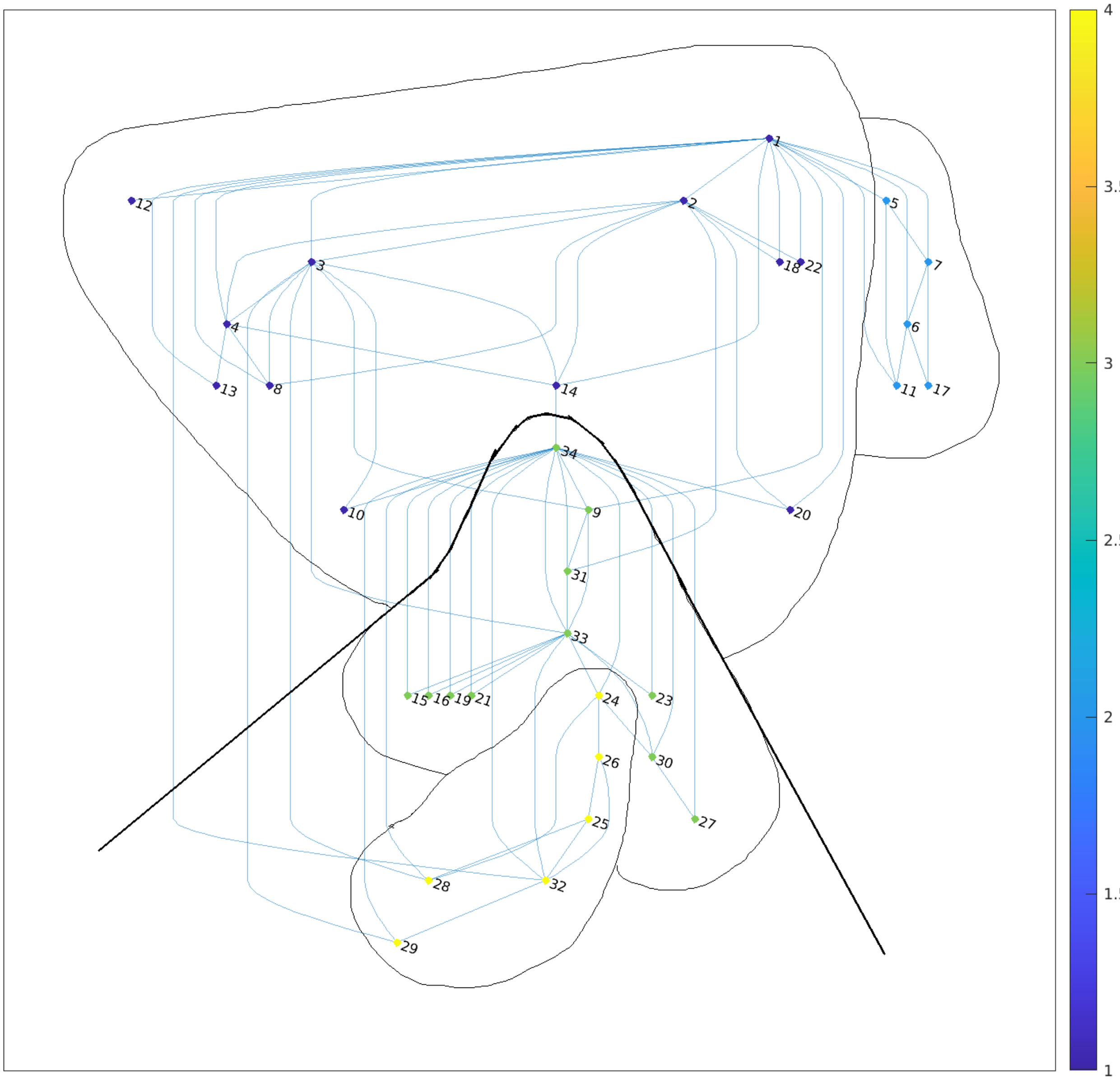

Fig.27: Newman's clustering into 4 groups(lines indicate 2 groups).

Fig.*27* depicts the four groups obtained by Newman's algorithm with $\gamma\ = 1$, while the bold line splits the graph into two groups obtained by setting $\gamma\ = 0.5$.

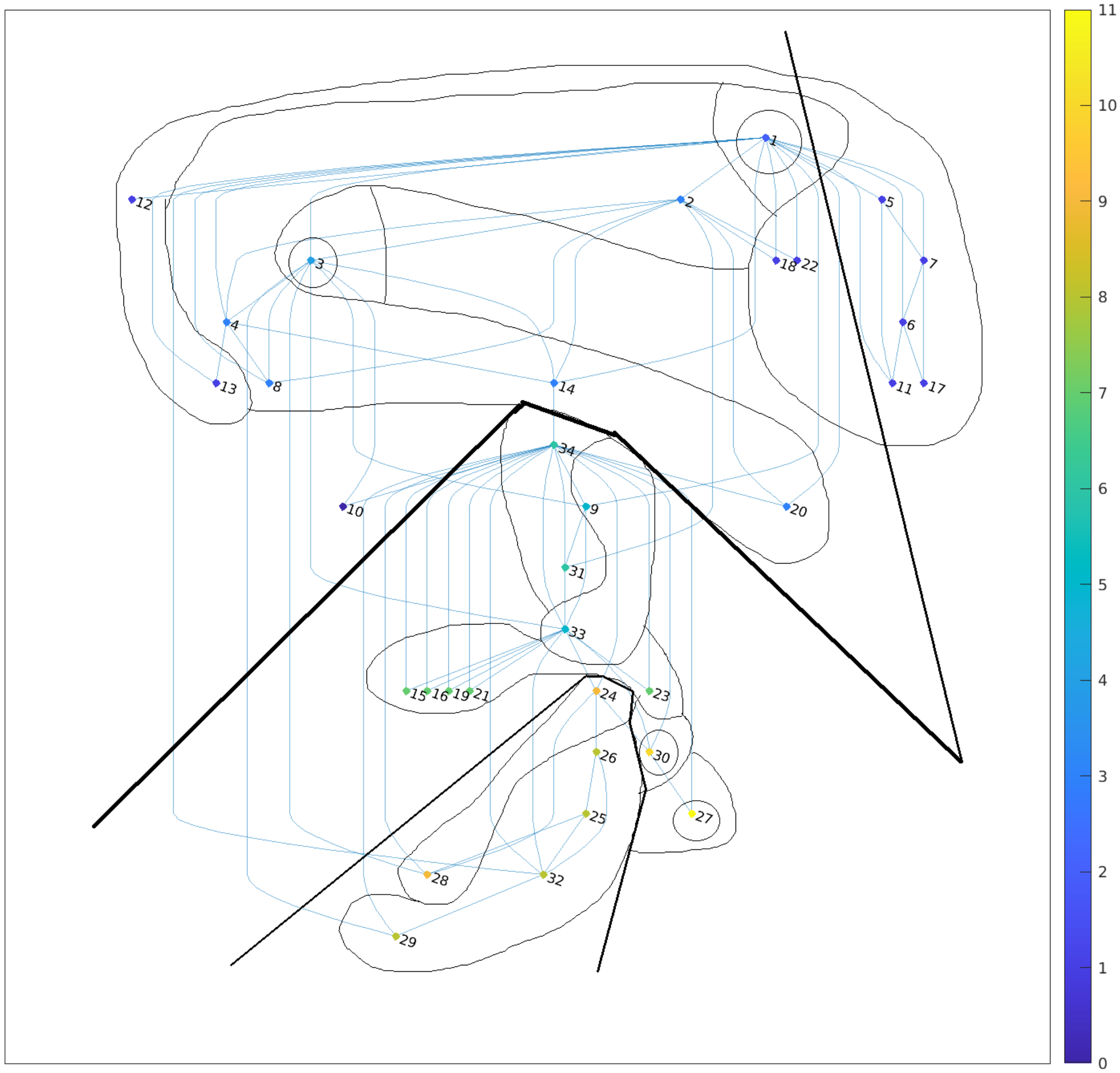

Fig.28: I.F. neuron model grouping for the Karate Club network with $\epsilon_{exc} = 2.2$ x 10^{-6} and $\epsilon_{inh} = 10^{-3}$(lines indicate Newman's clusters).

Fig.*28:* Nodes in circles have corresponding neurons *doing their own thing* and those enclosed in a set are firing together. Neuron for node 10 couldn't get over inhibition, and hence it wasn't firing at all.

The color-bar on the right indicates the sequence of firing of groups through the color-scale. Therefore, this figure shows that our neural network gives decent grouping as compared to those given by Newman's.

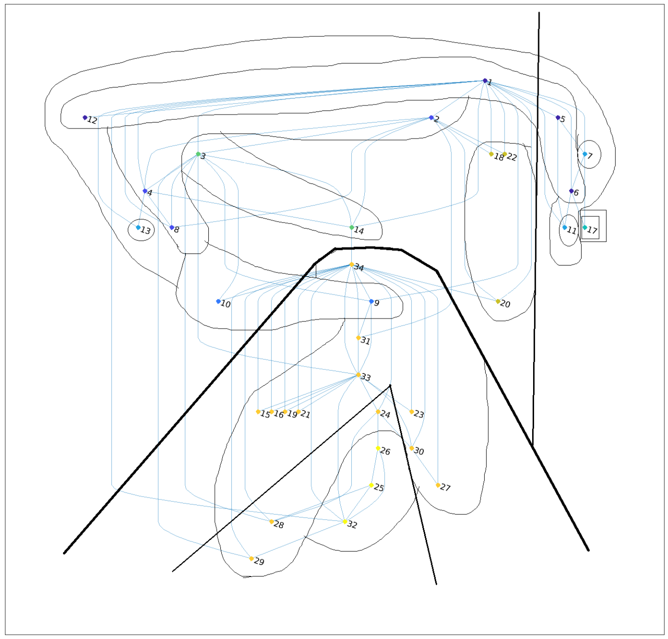

Fig.29: I.F. neuron model grouping for the Karate Club network with $\epsilon_{exc} = 2.3$ x 10^{-6} and $\epsilon_{inh} = 10^{-3}$(lines indicate Newman's clusters).

***Fig.**29 and **Fig.**30 show similar groupings for different values of* ϵ_{inh} *and* ϵ_{exc} *.*

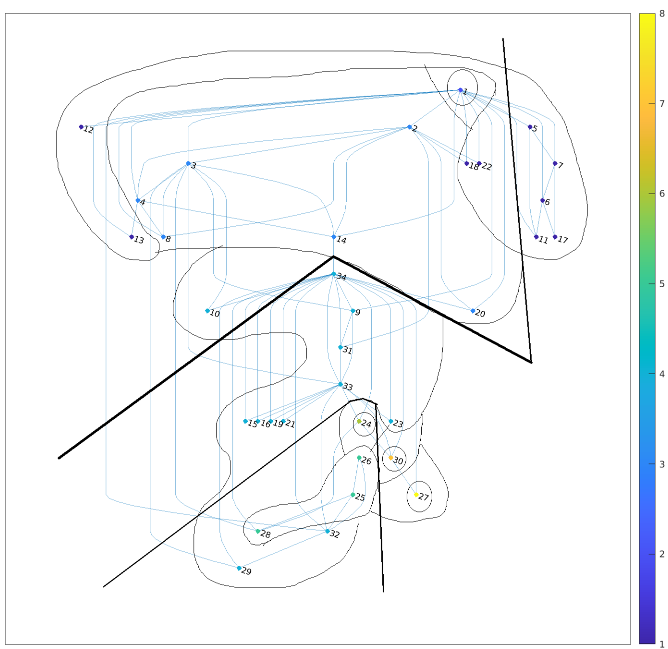

Fig.30: I.F. neuron model grouping for the Karate Club network with $\epsilon_{exc} = 2.4$ x 10^{-6} and $\epsilon_{inh} = 10^{-3}$(lines indicate Newman's clusters).

Chapter 5

Further Discussion

Taking a Closer Look

The constraints graph that we considered has courses as nodes and edge weights account for the number of students choosing both the courses corresponding to those nodes. Here the number of students signing up for a particular course does not show up anywhere.

In the Greedy Merge Algorithm, when there are multiple lowest merging groups, we have chosen the ones which give smallest group size after merging. But considering a possibility that one may desire the courses with higher number of students to stay in different groups, one can modify the criterion for a tie-break accordingly.

Whenever we say Greedy Algorithm, we've used it *only* with the decreasing degree sequence of vertices! Better results *may* be obtained by altering the sequence, or trying out luck by using multiple random ones. However, for the general scenario, there's no guarantee of a better outcome in doing so.

Neuronal Network Dynamics

Taking cue from biological neuronal networks in the olfactory system and the hippocampal formation, we can use oscillatory drive and noise to explore the possible dynamical states of the network. These states, in turn, may be mapped to different colorings of the constraint graph.

Extension to General Setup

In our constraint graph, since we are using a two-stage approach of grouping then permuting, we had only one vertex for each course and that suffices our need. However, in the general setup for the post enrolment-based course timetabling problem(ITC2007) with k time-slots, the groups will be the time-slots themselves.

If a course has three events, viz. lectures and/or tutorials, every week then they will appear as three different nodes in the graph. The graph will have additional edges for constraints like no multiple lectures of the same course are to be allowed on the same day. If a coloring of this new graph of events is a k-coloring, we put those groups in k time-slots and obtain our solution.

Bibliography

[1] Chowdhary, S. and Assisi, C., Maximizing the coding capacity of neuronal networks, BioRxiv 673632, 2019.

[2] Lewis, R.M.R., A Guide to Graph Colouring: Algorithms and Applications, 2016.

[3] PATAT - International Conference on the Practice and Theory of Automated Timetabling(patatconference.org/communityService.html).

[4] Wikipedia, the free encyclopedia(https://wikipedia.org).

[5] M. E. J. Newman, Modularity and community structure in networks, PNAS, 2006.

[6] Renato E. Mirollo and Steven H. Strogatz, Synchronization of Pulse-Coupled Biological Oscillators, SIAM Journal on Applied Mathematics(Vol. 50, pp. 1645-1662), 1990.