Community Detection in Networks

Zachary Fine Emaan Hariri Sayan Paul

1. Introduction

Community detection, at its core, attempts to find extra structure among nodes in a graph beyond just vertex-vertex connections—namely, by grouping them into groups of nodes called *communities*. This has wide-reaching applications, from grouping people into social networks to analyzing nodes in a local area network. Given this paradigm, there are two main underlying issues to address:

1. How does one determine the quality of a community assignment?
2. Given some measure of quality, how does one generate a community assignment?

We will address both of these issues in the following discussion. One measure of the quality of a community assignment is *modularity* (Girvan & Newman, 2002). This is an extension of an idea called *assortativity*, where high assortativity means that there are more edges inside communities than between them. Many real-world networks can be classified as having high or low assortativity, so knowing the level of assortativity one expects to see for some problem is helpful in determining the proper method to assign communities.

In this paper, we focus our discussion on two classes of approaches to community detection: the Girvan-Newman algorithm and various augmentations of the stochastic block model. We also explore theoretic bounds on the possibility of exact recovery and an algorithm to do so in a planted bipartite graph, a special case of the stochastic block model.

2. Girvan-Newman Algorithm

We will first examine the more standardized community structure detection algorithm proposed by Girvan and Newman for networks (Girvan & Newman, 2002). The Girvan Newman (GN) algorithm is influenced by the hierarchical divisive clustering method that uses edge betweenness centrality as its linkage criterion. Edge betweenness centrality is an analog of Freeman's definition of vertex betweenness centrality, which refers to the degree to which a vertex lies on the shortest paths between the other vertices (Freeman, 1977). Brandes provides an algorithm for calculating edge betweenness centrality for a graph $\mathcal{G} = (V, E)$ when defined as

$$c_B(e) = \sum_{s,t \in V} \frac{\sigma(s, t \mid e)}{\sigma(s, t)},$$

where $\sigma(s, t)$ is the number shortest paths between vertices $s, t \in V$ and $\sigma(s, t \mid e)$ is the number of those paths that pass through edge $e \in E$ (Brandes, 2008). The general idea is that edge between centrality would be higher in edges that are in between two different communities, so the removal of such edges would ideally separate a graph into its communities. The GN algorithm operates as follows:

1. Calculate edge betweenness centrality of all edges.
2. Remove the edge(s) with highest edge betweenness centrality.
3. Recalculate edge betweenness for all remaining edges.
4. Repeat from step 2 until no edges remain.

Note that the GN algorithm returns a dendrogram similar to those seen in hierarchical clustering rather than an explicit assignment to communities. Calculating edge betweenness centrality takes time $O(nm)$ so the total runtime of the algorithm is $O(nm^2)$.

3. Modularity in Girvan-Newman

The algorithm performs well in finding community structures which are known to exist, but if the communities are not known in advance, then a method must be chosen by which to choose which "slice" of the dendrogram to choose. For example, two possible "slices" of the output of GN on a graph are $[\{1,2,3,4\},\{5,6\}]$ and $[\{1,2\},\{3,4\},\{5,6\}]$, produced in subsequent iterations of edge removal. Newman and Girvan suggest *modularity* as a quantative measurement of community structure. Modularity, denoted Q, measures the fraction of intra-community edges minus the expected number of edges if edges were distributed randomly. This is calculated as

$$Q = \sum_i (E_{ii} - a_i^2) = \mathrm{Tr}(E) - \|E^2\|$$

Where $E \in \mathbb{R}^{k\times k}$ is a symmetric matrix in which E_{ij} gives the fraction of edges in the network that go from community i to community j, $a_i = \sum_j E_{ij}$ is the fraction of edges that connect to community i, and $\|X\|$ is the sum of the entries of X. We see in the above that E_{ii} can be thought of as the probability that a random edge is between two nodes in community i and a_i is the probability that one end of a random edge is in community i. Newman provides another definition for modularity as

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j)$$

where A is the adjacency matrix, k_i is the degree of vertex i, and $\delta(x,y) = 1$ if $x = y$ and 0 otherwise (Newman, 2006a). Here we observe that $k_i k_j/2m$ can be thought of as the probability that a random edge will go between i and j and $1/2m$ is a normalization constant. Notice that as Fortunato points out, the term $k_i k_j/2m$ can be generalized to $P_{i,j}$, which is the expected number of edges between vertices i and j in the *null model*, which is the original graph that is structurally identical without community assignments (Fortunato, 2010).

Here we have an example of the application of modularity with the GN algorithm. Consider a graph generated using the stochastic block model (see figure 1), which we will discuss in more detail below. Applying the GN algorithm, we derive

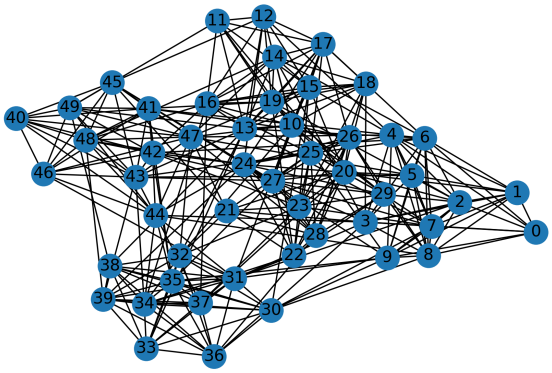


Figure 1. 5-community graph; $p = 0.85.$; $q = 0.15$

a sequence of modularities after every iteration that is seen if a "split" is made at that level of the dendrogram (see figure 2). After applying

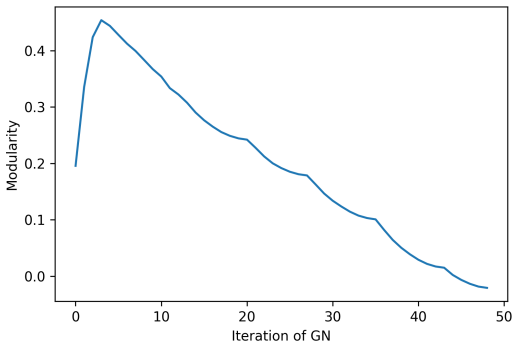


Figure 2. Modularities of graph in figure 1 as GN progresses.

the "split" from iteration five, which achieves the maximum modularity as expected, we derive the assignment seen in figure 3.

4. Modularity Matrix and Spectral Method

Modularity is a commonly used quality function of community assignments in graphs, since a higher modularity indicates that vertices within a community structure are more connected than expected by chance. We observe the when there are only two communities, 1 and 2, and we wish to assign each vertex to a community, we can rewrite the formula for modularity. Letting $B_{i,j} = A_{i,j} - k_i k_j/2m$ be

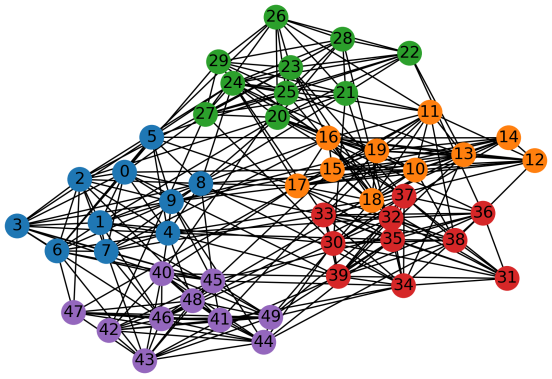


Figure 3. Final assignment to 1 that acheives maximum modularity in GN.

the *modularity matrix*, we write the above as

$$Q = \frac{1}{2m} B_{ij}\delta(c_i, c_j) = \frac{1}{4m} s_i B_{i,j} s_j = \frac{1}{4m} s^\top B s$$

where $s_i = 1$ if i is in community 1 and -1 if i is in community 2. Using this, we see that $\delta(c_i, c_j) = (s_i s_j + 1)/2$, and we can express our optimization problem as

$$\max_{s\in\{-1,1\}^n} \frac{1}{4m} s^\top B s$$

where we can equivalently find

$$\arg\max_{s\in\{-1,1\}^n} s^\top B s$$

Modularity maximization is an NP-Hard problem, and thus a heuristic such as spectral approximation, or agglomerative/divisive clustering is used. A greedy heuristic can be seen when applying modularity to determining the best "slice" when using the GN algorithm (Brandes et al., 2008). The spectral heuristic (different from spectral partitioning) is seen in Newman's leading eigenvector method (Newman, 2006b). We express the modularity in terms of the eigenvalues and unit eigenvectors of B ($\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$ and $u_1, u_2, \ldots, u_n$ respectively, with eigenvalues in decreasing order) as

$$Q = \frac{1}{4m} \sum_{i=1}^{n} (s^\top u_i)^2 \beta_i.$$

We find the eigenvector associated with the largest eigenvector of the modularity matrix B and call this eigenvector u_1. We see that there is no guarantee (in fact it is unlikely) that $u_1 \in \{-1, 1\}^n$, and so we will choose s to be "as parallel as possible" to u_1, which is done by maximizing $s^\top u_1$. This can be done by simply assigning s_i the sign of $(u_1)_i$ ($s_i = \text{sgn}((u_1)_i)$).

Notice however that $\sum_j B_{i,j} = 0$, and so $\mathbf{1} = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^\top$ is the trivial eigenvector associated with eigenvalue 0, much like the Laplacian. Unlike the Laplacian, B is not PSD and so the eigenvalues can possibly be negative. In the event where all the non-trivial eigenvalues are negative, this reveals that the best community structure exists only when all vertices are in the same community (as is the case with $s = \mathbf{1}$). Furthermore, the actual magnitude of the indices of u_1 reveal fundamentally how community-centric each vertex is. In other words, the larger in magnitude $(u_1)_i$ is, the more central i is in its community.

Applying this algorithm to the 2-community (SBM; $p = 0.5, q = 0.1$) seen in figure 4, we derive a sequence of eigenvalues for its modularity matrix (figure 5). We observe the community assign-

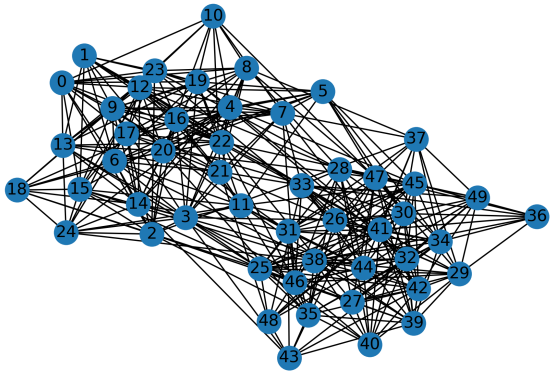


Figure 4. 2-community graph; $p = 0.5$; $q = 0.1$.

ments associated with the smallest and largest eigenvalues (figures 6 and 7, respectively), and clearly see that the assignment associated with the largest eigenvalue captures the community structure better.

We note that the leading eigenvector method can be generalized to more than two communities by one of two ways. First, we can simply recursively apply the algorithm, so when we first determine a s,

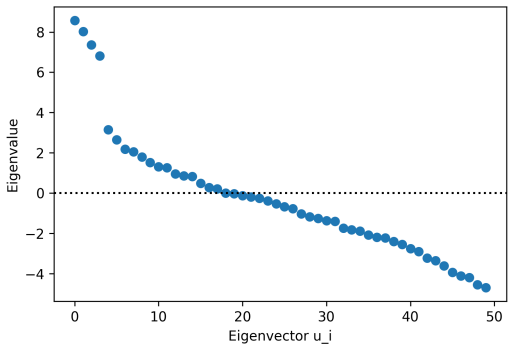


Figure 5. Eigenvalues associated with modularity matrix B of the graph in figure 4 in descending order.

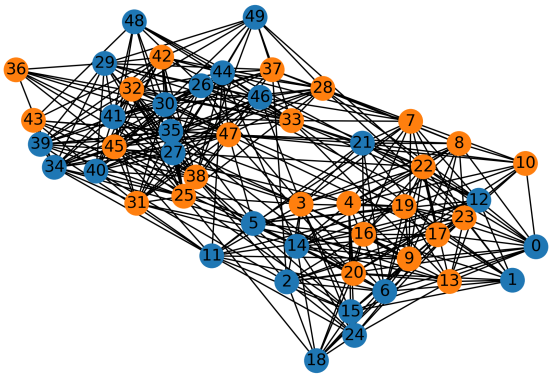


Figure 7. Assignment of $s_i = \text{sgn}(u_{50})_i$ to figure 4.

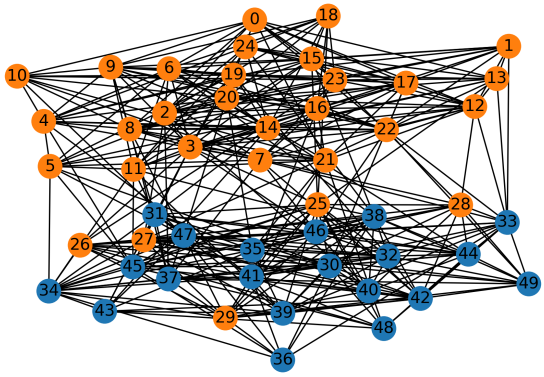


Figure 6. Assignment of $s_i = \text{sgn}(u_1)_i$ to figure 4.

we separate G into G_1 and G_2 and run the leading eigenvector method on each subgraph separately, halting when there is an increase in modularity. Alternatively, we have matrix $S = \begin{bmatrix} s_1 & \dots & s_c \end{bmatrix}$ (each s_i is a column vector, $s_i \in \{0,1\}^n$) where $S_{i,j} = 1$ if i is in community j and 0 otherwise. Now we have $\Delta(g_i, g_j) = \sum_{k=1}^{c} S_{i,k} S_{j,k}$, which results in modularity

$$\begin{aligned} Q &= \sum_{i,j} \sum_{k=1}^{c} B_{i,j} S_{i,k} S_{j,k} = \text{Tr}(S^\top B S) \\ &= \sum_{j=1}^{n} \sum_{k=1}^{c} (s_k^\top u_j)^2 \beta_j. \end{aligned}$$

Our procedure is analogous to the above, but now we choose orthogonal $s_1, \dots, s_c$ proportional to the leading $u_1, \dots u_c$. Again, we only wish to choose those which contribute *positively* to modularity. However, due to $s_i \in \{0,1\}$ we can only get an approximate answer to the number of communities c that we want (we obtain an upper bound).

We find empirically that the above algorithms work fairly effectively. The leading eigenvector method (or modular method) generally works better than spectral partitioning in many networks. The above algorithms and heuristics can be rather time intensive when calculating Q exactly. For very large networks Clauset, Newman, and Moore provide a method which relies on tracking changes to the modularity function ΔQ through iterative updates using a max heap (Clauset et al., 2004).

The Girvan-Newman algorithm and subsequent updates and improvements (e.g. adding spectral heuristic) have found a great deal of popularity in the past decade. Various modifications and speed improvements have been made to the GN algorithm, including the allowment of overlapping clusters (Gregory, 2007), faster approximation of edge betweenness centrality using a *network structure index* (Rattigan et al., 2007), and a pseudo-Monte Carlo estimate of edge betweenness centrality (Tyler et al., 2005). The GN algorithm will likely continue to be used in the analysis of community structure in biological, social, internet, and many other varieties of networks due to its efficiency and effectiveness on these networks.

5. The Stochastic Block Model

The Stochastic Block Model (SBM) is a common formulation of the community detection problem. It has the benefits of rigorous statistical justifica-

tion as well as the ability to approximately model many real-world community structures. We will first define the SBM before discussing some of the limitations of such a model. Finally, because the SBM is generative, we will discuss the problem of Exact Recovery in the SBM (Abbe et al., 2014).

The SBM is defined as follows. Let there be k communities. Then define a $k \times k$ symmetric matrix $\mathcal{X}$ of $[0, 1]$ entries, so that $\mathcal{X}_{ij}$ is the probability of an edge from any member of community i to any member of community j. It is typical for the diagonal elements to be larger than the off-diagonal elements, so that communities are highly clustered and have few edges to other communities. If we have an undirected graph, then $\mathcal{X}$ is symmetric. One clear example of SBM as a model for another regular problem is that of the planted bipartite graph. In this, we have two communities with the probability of intra-community edges $p = 0$ and probability of inter-community edges $q > 0$. This is simply an instance of the SBM with two groups.

6. Exact Recovery in the Stochastic Block Model

The problem of exact recovery is almost never guaranteed to be solvable, due to the random nature of edge connections. Therefore when we discuss successful exact recovery of the model, we do so with high probability (w.h.p.), which is to say that the probability of success is bounded away from zero. In the problem of exact recovery, we will discuss only the case of two communities and their intra- and inter-community edge probabilities p and q respectively. Moreover, we let p and q scale with n, the number of nodes in the graph, so that $p = \alpha \log(n)/n$ and $q = \beta \log(n)/n$. Then the model is defined over the parameters $\mathcal{G}(n, p, q)$. Information theoretic bounds (Abbe et al., 2014) on the probability of exact recovery in the stochastic block model (using maximum likelihood, which by definition is the most accurate recovery technique, but which is also NP-hard in the worst case) have

$$\frac{\alpha + \beta}{2} > 1 + \sqrt{\alpha\beta}$$

as a bound which lets us achieve exact recovery (w.h.p.). This bound is tight, in that if the inequality is not satisfied, the probability of exact recovery tends to zero as n grows large. Approximation algorithms such as the one we are about to look at achieve bounds which are looser but similar. In particular, the threshold achieved in the following section is provably successful w.h.p. in exact recovery when

$$(\alpha - \beta)^2 > 8(\alpha + \beta) + 8/3(\alpha - \beta).$$

This is comparable to the original bound in that

$$\frac{\alpha + \beta}{2} > 1 + \sqrt{\alpha\beta} \text{ iff } (\alpha - \beta)^2 > 4(\alpha + \beta) - 4 \text{ and } \alpha + \beta > 2.$$

So, the bound achievable in a polynomial time algorithm is only slightly looser than the information-theoretic one. One can compare the two bounds using figures 8 and 9, and observe first the intuition that we want α and β to be as separated (different) as possible, so that the groups are more clearly defined, and second, to observe the thinner area on the SDP bound, which indicates that its requirement to retrieve separation is stricter than that of maximum likelihood.

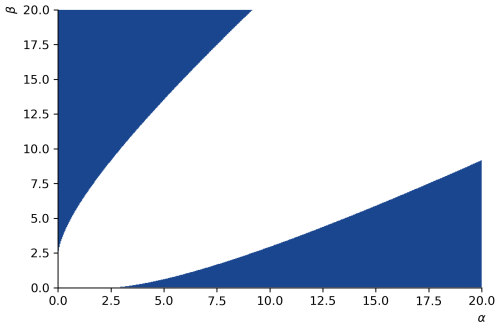


Figure 8. A bound on α and β necessary to achieve exact recovery using Maximum Likelihood w.h.p.

7. Semidefinite Programming Based Exact Recovery in the Stochastic Block Model

Let $\mathcal{G} = (V, E(\mathcal{G}))$ be the observed graph. We seek to maximize the cut of ± 1 assignments to an n-length vector x so that we maximize the number of intra-community edges and minimize the number of inter-community edges. Call these

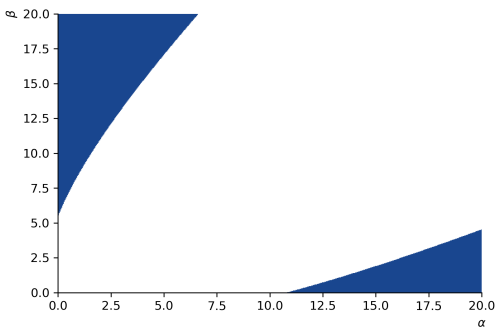


Figure 9. A bound on α and β necessary to achieve exact recovery using Semidefinite Programming w.h.p.

intra-community edges "correct edges" and these inter-community edges "incorrect edges." However, this approach as-is would yield us a single community with all the vertices and the other with zero. So we count non-edges (edges which do not exist) against the cut, so that we minimize the number of intra-community non-edges and maximize the number of inter-community non-edges. Call these intra-community non-edges "incorrect non-edges" and these inter-community non-edges as "correct non-edges." We can represent this problem as an integer quadratic program involving a single matrix B where B_{ij} is 1 if $(i,j) \in E(\mathcal{G})$ and -1 if $(i,j) \notin E(\mathcal{G})$. Then we solve

$$\begin{aligned} \max_x \ & x^\top B x \\ & x_i = \pm 1 \end{aligned}$$

Notice that $x^\top B x = \sum_i \sum_j B_{ij} x_i x_j$ is large if $x_i = x_j$ and $B_{ij} = 1$, for many i, j, indicating that the communities have many edges within them. It is also large if $x_i \neq x_j$ and $B_{ij} = -1$ for many i, j, indicating that communities do not have many edges between them. This problem is, unfortunately, NP-hard, but we can find a semidefinite programming relaxation of this as follows:

$$\begin{aligned} \max_X \ & \mathrm{Tr}(BX) \\ & X_{ii} = 1 \\ & X \succeq 0 \end{aligned}$$

Let g be the ground-truth vector consisting of ± 1 assignments to communities. Then we argue that the above SDP, which is solvable in linear time, has solution exactly equal to $gg^\top$ w.h.p. if

$$(\alpha - \beta)^2 > 8(\alpha + \beta) + 8/3(\alpha - \beta).$$

We give a sketch of the proof given in (Abbe et al., 2014) and provide intuition for some of the constructions given within. We will not reprove anything here, as their proofs are quite thorough.

The proof is structured as follows: First, we show that the desired optimum ($gg^\top$) is feasible and find the dual of the SDP. The goal, then, is to use weak duality in addition to showing that there exists a feasible solution to the dual SDP with the same value as the primal. This will imply $gg^\top$ is optimal. We will largely ignore uniqueness here, but the proof in the paper uses the nonnegativity of the second eigenvalue of a particular matrix, given in expression 1.

Now clearly, $gg^\top$ is feasible in the given SDP, since its diagonal values are simply the squares of the entries of g which are all ± 1, and $gg^\top$ is always positive semidefinite. The dual of the given SDP, then, is

$$\begin{aligned} \min \ & \mathrm{Tr}(Y) \\ & Y \succeq B \\ & Y \text{ diagonal.} \end{aligned}$$

And the remainder of the proof is centered around demonstrating that there is a feasible Y s.t. $\mathrm{Tr}(Bgg^\top) = \mathrm{Tr}(Y)$. To do so, we will first analyze the matrix $Bgg^\top$, and show that its diagonal elements are equivalent to the elements of a matrix Y which is feasible in the dual.

Define $\mathcal{G}_+$ to be a subgraph of $\mathcal{G}$ which contains only intra-community edges, and likewise $\mathcal{G}_-$ for inter-community edges so that $\mathcal{G} = (\mathcal{G}_+) + (\mathcal{G}_-)$. Then let $D^+_{\mathcal{G}}$ be a diagonal matrix of degrees of vertices in $\mathcal{G}_+$, and likewise for $D^-_{\mathcal{G}}$.

Recall our notion of correct and incorrect edges and non-edges. Then note that $(Bgg^\top)_{ii}$ is the sum of positive correct edges and non-edges, and of negative incorrect edges and non-edges. That

is,

$$\begin{aligned}(Bgg^\top)_{ii} = &\text{ correct edges + correct non-edges}\\ &-\text{incorrect non-edges - incorrect edges}\\ =&\ (D^+_{\mathcal{G}})_{ii} + \left(\frac{n}{2} - (D^-_{\mathcal{G}})_{ii}\right)\\ &-\left(\frac{n}{2} - 1 - (D^+_{\mathcal{G}})_{ii}\right) - (D^-_{\mathcal{G}})_{ii}\\ =&\ 2((D^+_{\mathcal{G}})_{ii} - (D^-_{\mathcal{G}})_{ii}) + 1\end{aligned}$$

So simply let $Y = 2(D^+_{\mathcal{G}} - D^-_{\mathcal{G}}) + I_n$, which lets $\mathrm{Tr}(Bgg^\top) = \mathrm{Tr}(Y)$. Now all that is left to check is the requirement that $Y \succeq B$. Then let A be the adjacency matrix of $\mathcal{G}$ and define the SBM Laplacian to be

$$L_{\mathrm{SBM}} = D^+_{\mathcal{G}} - D^-_{\mathcal{G}} - A.$$

Then we want

$$\begin{aligned}&2(D^+_{\mathcal{G}} - D^-_{\mathcal{G}}) + I_n \succeq B\\ \Longrightarrow\ &2(D^+_{\mathcal{G}} - D^-_{\mathcal{G}}) + I_n - B \succeq 0\\ \Longrightarrow\ &2(D^+_{\mathcal{G}} - D^-_{\mathcal{G}} - A) + 2A + I_n - B \succeq 0\\ \Longrightarrow\ &2L_{\mathrm{SBM}} + 2A + I_n - B \succeq 0\end{aligned}$$

Now, note that for any entry of A, if $(i,j) \in E(\mathcal{G})$, then $A_{ij} = 1$ and $B_{ij} = 1$. And if $(i,j) \notin E(\mathcal{G})$, then $A_{ij} = 0$ and $B_{ij} = -1$. Therefore $2A - B = 11^\top$, implying

$$2L_{\mathrm{SBM}} + 2A + I_n - B = 2L_{\mathrm{SBM}} + I_n + 11^\top.^1$$

Then the final step is to use the condition that $(\alpha-\beta)^2 > 8(\alpha+\beta) + 8/3(\alpha-\beta)$ to justify

$$2L_{\mathrm{SBM}} + I_n + 11^\top \tag{1}$$

is positive semidefinite with high probability, whose proof is omitted here. The main step of this proof is to construct two matrices, one of which is deterministic and is the expected value of a particular construction, and the other of which is the random matrix corresponding to the zeroed out (in expected value) random assignment of ± 1 edges between vertices. The final step is to use Bernstein's inequality to prove the bound.

[1]The source asserts that we have $2L_{\mathrm{SBM}} + I_n - 11^\top$ instead, and we are unsure why.

8. Degree-Corrected Stochastic Block Model

There are a number of issues with the regular stochastic block model defined in section 5. First, to achieve the best results it requires one to know the number of communities in the graph *a priori*. Second, while the model seemingly places few restrictions on the types of community structures it detects, it does have some bias in practice.

Given a graph $\mathcal{G} = (V, E)$, suppose we have some matrix $\mathcal{X}$, where $\mathcal{X}_{r,s}$ denotes the probability of having an edge between a vertex in group r and s. We also have an adjacency matrix $A_{ij} \in \{0,1\}^{n\times n}$. Note that $\mathcal{X}_{rs}$ can also be thought to be the expected value of A_{ij} for all $i \in r, j \in s$. Under the SBM, the probability that our particular graph was generated by this matrix and group assignment g is

$$\Pr[\mathcal{G} \mid \mathcal{X}, g] = \prod_{r\le s\in g} \mathcal{X}_{rs}^{m_{rs}}(1-\mathcal{X}_{rs})^{n_r n_s - m_{rs}}$$

where n_r is the number of nodes in group r and m_{rs} is the number of edges between groups r and s. To maximize this, we can equivalently maximize the log probability:

$$\begin{aligned}&\log\Pr[\mathcal{G} \mid \mathcal{X}, g] =\\ &\quad\sum_{r\le s}(m_{rs}\log(\mathcal{X}_{rs}) + (n_r n_s - m_{rs})\log(1-\mathcal{X}_{rs}))\end{aligned}$$

To find the optimal value for $\mathcal{X}_{rs}$, we check the first-order conditions

$$\begin{aligned}\frac{\partial}{\partial \mathcal{X}_{rs}}\log\Pr[\mathcal{G} \mid \mathcal{X}, g] = \frac{m_{rs}}{\mathcal{X}_{rs}} - \frac{n_r n_s - m_{rs}}{1-\mathcal{X}_{rs}} &= 0\\ \Rightarrow m_{rs} - m_{rs}\mathcal{X}_{rs} &= n_r n_s \mathcal{X}_{rs} - m_{rs}\mathcal{X}_{rs}\\ \Rightarrow \hat{\mathcal{X}}_{rs} &= \frac{m_{rs}}{n_r n_s}\end{aligned}$$

The SBM exactly preserves the expected number of edges between two groups. Since $\mathcal{X}_{rs}$ is the probability of an edge existing between r and s, the expected number of edges between groups r and s is

$$\forall r, s \in g : \sum_{i:g_i=r, j:g_j=s} \mathcal{X}_{rs} = n_r n_s \mathcal{X}_{rs} = m_{rs}.$$

On some level, this is desirable, but let's consider the case where $r = s$, i.e. edges between nodes in the same group. Here, we see that the expected degree of each node, under the maximum likelihood estimate, is exactly the same, namely $n_r \mathcal{X}_{rr} = m_{rr}/nr$, while the actual degree of the vertices may be quite varied. Therein lies the primary issue with the SBM: the expected number of edges between groups is preserved, but the expected degree is not.

We wish to have a more general version of the stochastic block model, where the expected number of edges between groups is preserved *as well as* the expected degree of each node. One such strategy is the *degree-corrected stochastic block model* (Karrer & Newman, 2011).

In this formulation, we implicitly expand the range of graphs considered to include undirected multigraphs with self-edges, and we introduce a set of parameters θ_i which implicitly encode the degree of each node in the graph (shown later). There are some conventions to follow for adjacency matrices in an undirected multigraph. As expected, A_{ij} is the number of edges between nodes i and j. A_{ii}, on the other hand, is twice the number of self-edges from node i to itself.

Since we allow multiple edges between nodes, the Bernoulli distribution for each edge no longer applies. Instead, we say that the number of edges between two nodes i and j has a Poisson distribution with rate $\theta_i\theta_j\mathcal{X}_{g_i,g_j}$ for all $i \neq j$. Analogous to before, we have $\mathbb{E}[A_i j] = \theta_i\theta_j\mathcal{X}_{g_i,g_j}$ When $i = j$, we recall that $\mathbb{E}[A_{ii}] = \mathcal{X}_{ii}$ is twice the number of self-edges from a node to itself. So, we set the rates for those poisson distributions to be $\theta_i\theta_i\mathcal{X}_{ii}/2 = \theta_i^2\mathcal{X}_{ii}/2$.

To see how θ preserves the degree of each vertex, let's find the best θ under maximum likelihood.

$$\Pr[\mathcal{G} \mid \mathcal{X}, g, \theta] = \prod_{i<j\in V} \frac{(\theta_i\theta_j\mathcal{X}_{g_i,g_j})^{A_{ij}}}{A_{ij}!} e^{-\theta_i\theta_j\mathcal{X}_{g_i,g_j}} \cdot \prod_{i\in V} \frac{(\theta_i^2\mathcal{X}_{g_i,g_i}/2)^{A_{ii}/2}}{(A_{ii}/2)!} e^{-\theta_i^2\mathcal{X}_{g_i,g_i}/2}$$

We see that arbitrarily scaling θ_i to $\alpha\theta_i$ has no effect on our MLE. This is because $\alpha\theta_i\alpha\theta_j\mathcal{X}_{g_i,g_j} = \theta_i\theta_j(\alpha^2\mathcal{X}_{g_i,g_j})$, and we have control over $\mathcal{X}_{g_i,g_j}$. So, Karrer and Newman recommend the following constraint on θ_i:

$$\forall r, \sum_i \theta_i \mathbb{1}\{g_i = r\} = 1$$

Essentially, θ_i now represents the probability that an edge from any vertex in g_i goes to node i. With these definitions and constraints, the maximum likelihood estimates for θ_i and $\mathcal{X}_{rs}$ are

$$\hat{\theta}_i = \frac{\deg(i)}{\sum_{j:g_j=g_i}\deg(j)} = \frac{\deg(i)}{\sum_s m_{g_i,s}}$$
$$\hat{\mathcal{X}}_{rs} = m_{rs}$$

We recall that m_{rs} is the number of edges between groups r and s. Let's verify that this still preserves the expected number of edges between groups, using the fact that $\mathbb{E}[A_{ij}] = \theta_i\theta_j\mathcal{X}_{rs}$:

$$\begin{aligned}
\forall r, s \in g : \sum_{i:g_i=r, j:g_j=s} \theta_i\theta_j\mathcal{X}_{rs} &= \sum_{i:g_i=r,j:g_j=s} \frac{\deg(i)\deg(j)}{(\sum_q m_{rq})(\sum_t m_{st})} m_{rs} \\
&= \frac{(\sum_{i:g_i=r}\deg(i))(\sum_{j:g_j=s}\deg(j))}{(\sum_q m_{rq})(\sum_t m_{st})} m_{rs} \\
&= \frac{(\sum_q m_{rq})(\sum_t m_{st})}{(\sum_q m_{rq})(\sum_t m_{st})} m_{rs} \\
&= m_{rs}
\end{aligned}$$

Here, we recall that $\sum_q m_{rq}$ is the sum of all of the edges from group r to other groups, which is equivalent to the sum of the degrees of the nodes in group r. Now, we need to verify that $\hat{\theta}$ and $\hat{\mathcal{X}}$ do in fact preserve the expected degree of each node $i \in V$:

$$\begin{aligned}
\sum_j \theta_i\theta_j\mathcal{X}_{ij} &= \frac{\deg(i)}{\sum_s m_{g_i,s}} \sum_j \frac{\deg(j)}{\sum_s m_{g_j,s}} m_{g_i,g_j} \\
&= \frac{\deg(i)}{\sum_s m_{g_i,s}} \sum_r \sum_{j:g_j=r} \frac{\deg(j)}{\sum_s m_{r,s}} m_{g_i,r} \\
&= \frac{\deg(i)}{\sum_s m_{g_i,s}} \sum_r \frac{m_{g_i,r}}{\sum_s m_{r,s}} \sum_{j:g_j=r} \deg(j) \\
&= \frac{\deg(i)}{\sum_s m_{g_i,s}} \sum_r m_{g_i,r} \\
&= \deg(i)
\end{aligned}$$

9. Regularized Stochastic Block Model

The degree-corrected stochastic block model introduced a set of parameters θ_i that implicitly encode the degree of each node i (Karrer & Newman, 2011). The *regularized stochastic block model* splits this up into two sets of parameters, I_i, which implicitly encodes the degree of a node i within its group g_i, and O_i encodes the degree to all other groups (Lu & Szymanski, 2019). In other words, we set the rates the Poisson distributions for the number of edges between nodes i and j to be

$$\lambda_{ij} = \begin{cases} \mathcal{X}_{g_i,g_j} I_i I_j & g_i = g_j \\ \mathcal{X}_{g_i,g_j} O_i O_j & \text{o.w.} \end{cases}$$

We see that the degree-corrected stochastic block model reduces to this case if we add the constraint that $I_i = O_i$. Lu and Szymanski reformulate these parameters into $f_i = I_i/(I_i + O_i)$, or the ratio of the in-degree to the total degree of a vertex, and $\theta_i = I_i + O_i$, which is analogous to the θ_i from the degree-corrected model. The key point of this, however, is that the log-likelihood for a graph given some $\mathcal{X}$, θ, f, and g has a term that looks like

$$-2 \sum_i \deg(i) H\left(\frac{\text{in-deg}(i)}{\deg(i)}, f_i\right)$$

where "in-deg" is the number of edges from i to nodes in g_i and H is the cross-entropy function. Since this term is negated, this term acts as a regularizer on the maximum likelihood; the further away the ratio between in-degree and total degree is from the prior, the more it penalizes the objective function.

If we make the prior f_i the same for all i, we effectively have a dial on how assortative we expect a community structure to be. Namely, high assortativity means that there are many more edges within a group than between groups. In other words, the ratio between the in-degree and out-degree for each node is high (i.e. close to 1). Conversely, if we set f to be close to 0, then maximum likelihood on the regularized SBM will seek disassortative structures.

10. Appendix

The code the reproduce the figures above is attached to the end of this document.

References

Abbe, Emmanuel, Bandeira, Afonso S., and Hall, Georgina. Exact recovery in the stochastic block model. *CoRR*, abs/1405.3267, 2014. URL `http://arxiv.org/abs/1405.3267`.

Brandes, Ulrik. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145, 2008.

Brandes, Ulrik, Delling, Daniel, Gaertler, Marco, Gorke, Robert, Hoefer, Martin, Nikoloski, Zoran, and Wagner, Dorothea. On modularity clustering. *IEEE Trans. on Knowl. and Data Eng.*, 20(2):172–188, February 2008. ISSN 1041-4347. doi: 10.1109/TKDE.2007.190689. URL `http://dx.doi.org/10.1109/TKDE.2007.190689`.

Clauset, Aaron, Newman, Mark EJ, and Moore, Cristopher. Finding community structure in very large networks. *Physical review E*, 70(6): 066111, 2004.

Fortunato, Santo. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

Freeman, Linton C. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. ISSN 00380431. URL `http://www.jstor.org/stable/3033543`.

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

Gregory, Steve. An algorithm to find overlapping community structure in networks. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 91–102. Springer, 2007.

Karrer, Brian and Newman, M. E. J. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, Jan 2011. doi: 10.1103/PhysRevE.83.016107.

Lu, Xiaoyan and Szymanski, Boleslaw K. Regularized stochastic block model for robust community detection in complex networks. *CoRR*, abs/1903.11751, 2019. URL `http://arxiv.org/abs/1903.11751`.

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

Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006b. doi: 10.1103/PhysRevE.74.036104. URL `https://link.aps.org/doi/10.1103/PhysRevE.74.036104`.

Rattigan, Matthew J, Maier, Marc, and Jensen, David. Graph clustering with network structure indices. In *Proceedings of the 24th international conference on Machine learning*, pp. 783–790. ACM, 2007.

Tyler, Joshua R, Wilkinson, Dennis M, and Huberman, Bernardo A. E-mail as spectroscopy: Automated discovery of community structure within organizations. *The Information Society*, 21(2):143–153, 2005.

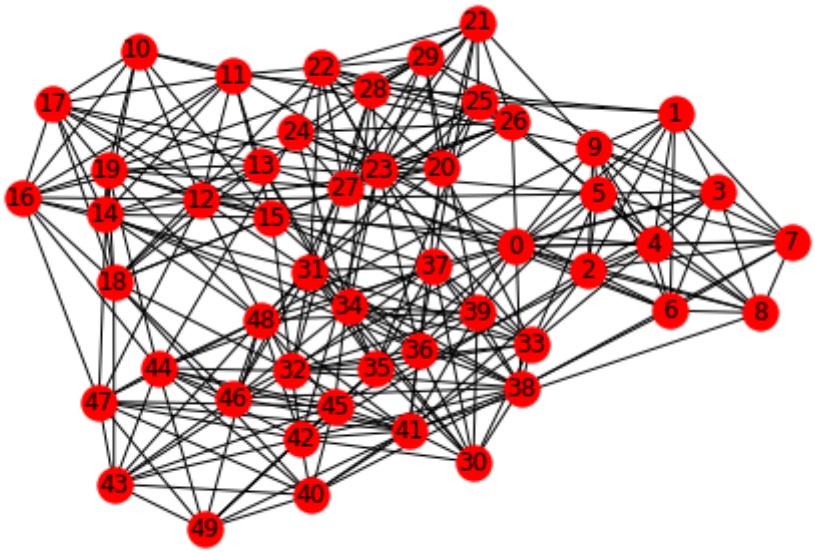
In [1]:
```
import networkx as nx
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import pandas as pd

sbm = nx.stochastic_block_model
gn = nx.algorithms.community.girvan_newman
modularity = nx.algorithms.community.quality.modularity
```

In [2]:
```
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728',
              '#9467bd', '#8c564b', '#e377c2', '#7f7f7f',
              '#bcbd22', '#17becf']
```

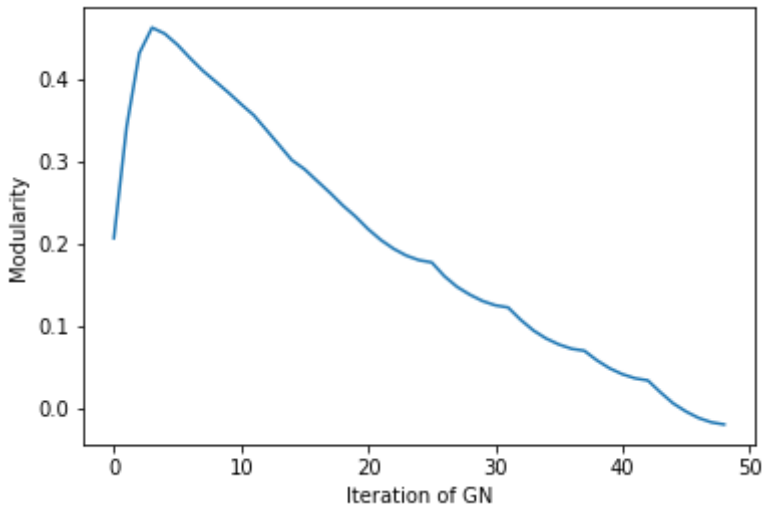
In [3]:
```
P = lambda r, p, q: q*np.ones((r, r)) + (p - q)*np.eye(r)
```

In [4]:
```
n = 10
r = 5
p = 0.90
q = 0.10
G = sbm(tuple([n]*r), P(r, p, q))
nx.draw(G, with_labels=True)
```

```
/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:61
1: MatplotlibDeprecationWarning: isinstance(..., numbers.Number)
  if cb.is_numlike(alpha):
```

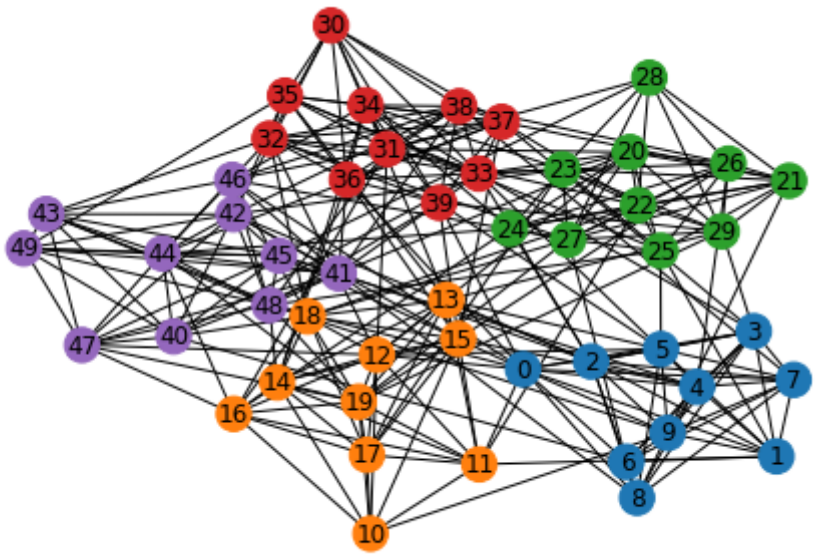


In [5]:
```
dendrogram = nx.algorithms.community.girvan_newman(G)
comm_assigns = list(part for part in dendrogram)
mods = [modularity(G, assign) for assign in comm_assigns]
plt.plot(mods)
# plt.title("Modularity versus iteration of GN")
plt.ylabel("Modularity")
plt.xlabel("Iteration of GN")
plt.savefig("mod.png", dpi=300)
```

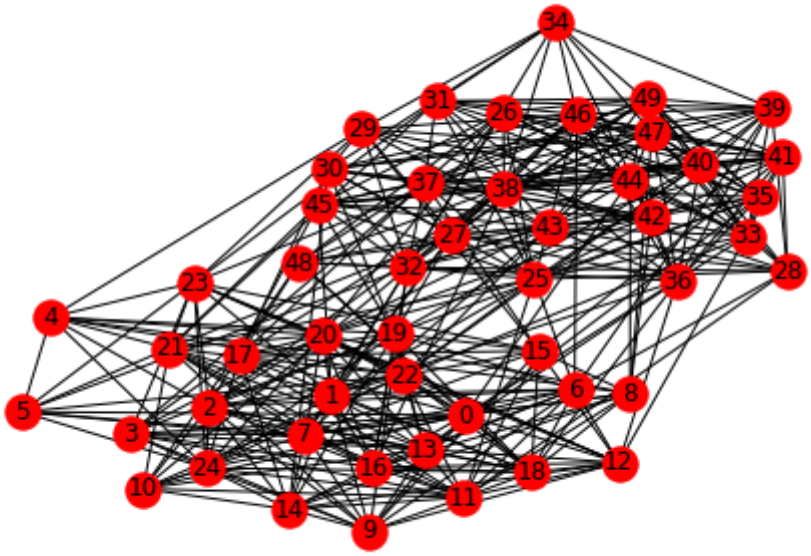


In [6]:
```
final_assign = comm_assigns[np.argmax(mods)]
color_assign = [''] * len(G.nodes)
for comm in range(len(final_assign)):
  for i in final_assign[comm]:
    color_assign[i] = colors[comm]
```

In [7]:
```
nx.draw(G, node_color=color_assign, with_labels=True)
```



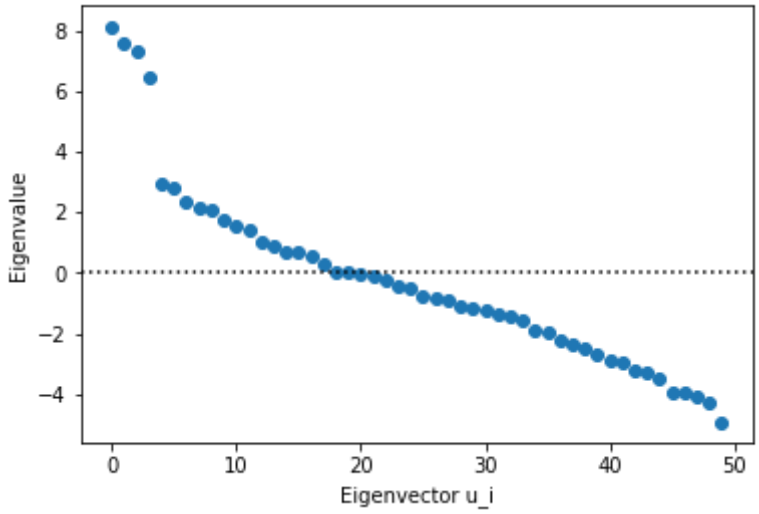
In [8]:
```
n = 25
r = 2
p = 0.50
q = 0.10
G_2 = sbm(tuple([n]*r), P(r, p, q))
nx.draw(G_2, with_labels=True)
```



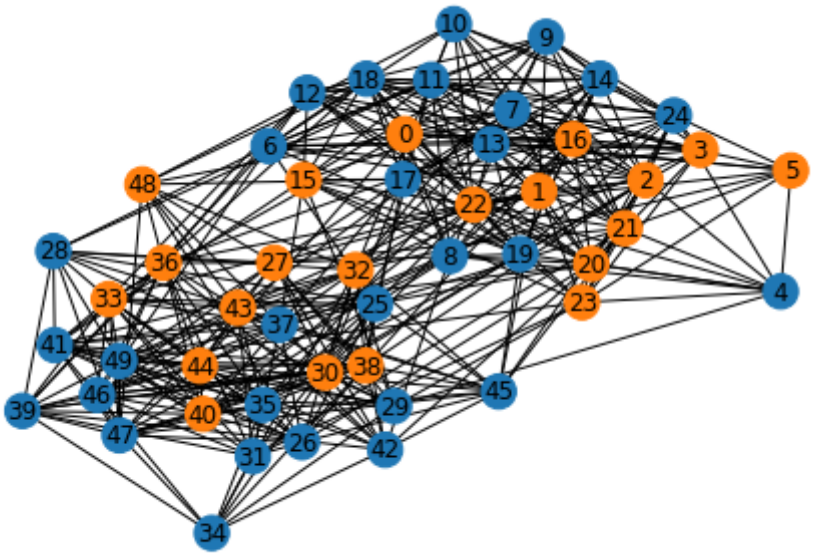
In [9]:
```
B = nx.modularity_matrix(G)
vals, vecs = np.linalg.eig(B)
idx = vals.argsort()[::-1]
vals = vals[idx]
vecs = vecs[:,idx]
```

In [10]:
```
plt.scatter([i for i in range(len(vals))], vals)
# plt.title("Eigenvalues of Modularity Matrix (B)")
plt.xlabel("Eigenvector u_i")
plt.ylabel("Eigenvalue")
plt.axhline(0, color='black', linestyle=":")
```

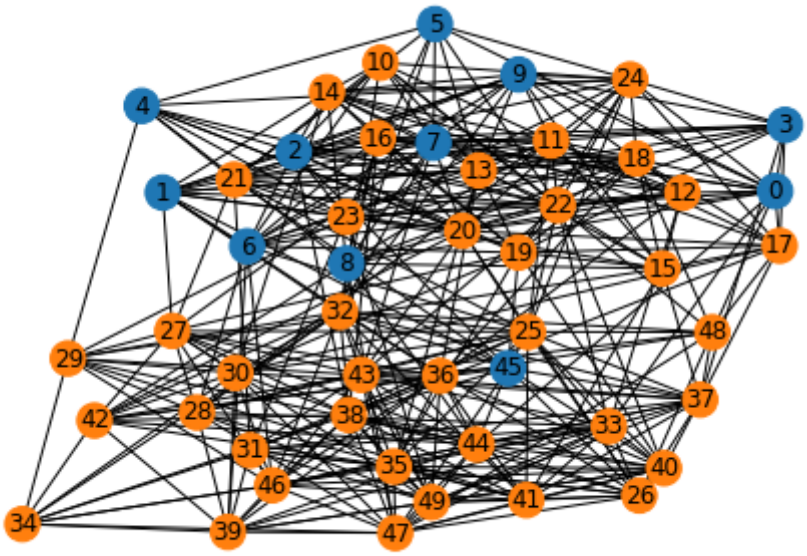
Out[10]: <matplotlib.lines.Line2D at 0xa22d8ad30>



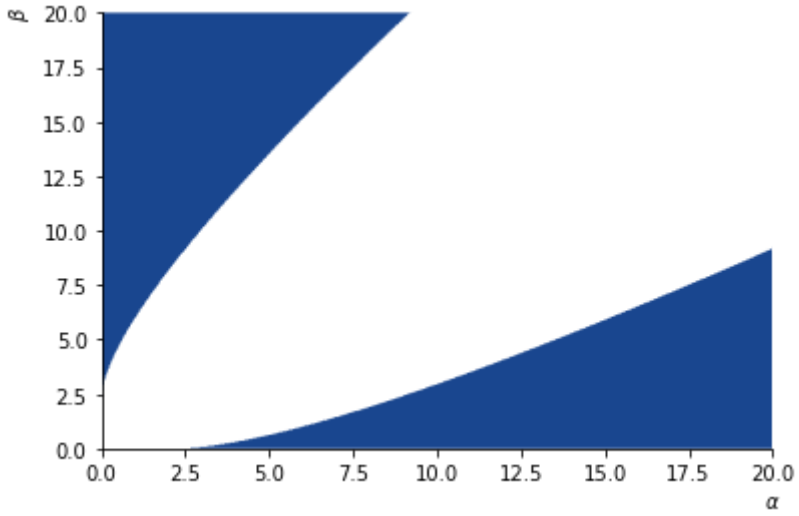
In [11]:
```
worst = vecs[:, -1]
worst_assign = list(np.array((np.sign(worst) + 1)/2, dtype=np.int).flatt
en())
worst_colors = [colors[i] for i in worst_assign]
nx.draw(G_2, with_labels=True, node_color=worst_colors)
```



In [12]:
```
best = vecs[:, 0]
best_assign = list(np.array((np.sign(best) + 1)/2, dtype=np.int).flatten
())
best_colors = [colors[i] for i in best_assign]
nx.draw(G_2, with_labels=True, node_color=best_colors)
```



In [13]:
```
import sympy
import sympy.abc

x, y = sympy.symbols("x y")

x, y = sympy.symbols("x y")

plot = sympy.plot_implicit(
    (x + y) / 2 > 1 + sympy.sqrt(x * y),
    (x, 0, 20),
    (y, 0, 20),
    line_color="#19468f",
    points=1000,
    xlabel=r'${0:s}$'.format(sympy.latex(sympy.abc.alpha)),
    ylabel=r'${0:s}$'.format(sympy.latex(sympy.abc.beta)))
plot._backend.fig.tight_layout()
```



In [14]:
```
plot = sympy.plot_implicit(
    (x - y) * (x - y) > 8 * (x + y) + 8 * (x - y) / 3,
    (x, 0, 20),
    (y, 0, 20),
    line_color="#19468f",
    points=1000,
    xlabel=r'${0:s}$'.format(sympy.latex(sympy.abc.alpha)),
    ylabel=r'${0:s}$'.format(sympy.latex(sympy.abc.beta)),
    figsize=(2,2))
plot._backend.fig.tight_layout()
```