Summary: *Modularity and Community Structure in Networks* by Mark E. J. Newman

**Main Point:** Newman defines the modularity of a graph partitioning as the difference between the number of edges within the partitions found and the number of expected edges in these partitions if the network were produced by placing edges at random between vertices of an equivalent degree distribution. He argues that optimizing modularity is a better approach for community detection than partitional approaches useful to other domains such as load balancing in parallel computing, because we do not know the number of communities present in a network, nor their sizes, in advance. He then describes a spectral method for modularity maximization that shows that finding the principal eigenvector of what he calls the modularity matrix is enough to maximize the modularity of a bisection of any given network, and proves further that a recursive application of this method to the partitions of a network will naturally lead to the number of divisions giving the maximum modularity. Newman shows several experimental results on real networks of up to 27,000 vertices validating that the communities found by maximizing modularity are meaningful and that the spectral method for modularity maximization outperforms previously attempted approaches.

Newman begins by pointing out the difference between partitioning methods useful for applications such as load balancing in parallel computing, where we seek to partition a graph into a set number of partitions with the goal of minimizing the number of edges that cross partitions, and the goal in community detection, where we do not automatically assume that good partitions exist, but we would like to discover any that do. Community detection is therefore more exploratory and algorithms designed for this purpose need not output any communities at all if they do not exist in a network, and they shouldn’t look for a fixed number of communities nor seek communities of a fixed size.

Newman goes on to explain that modularity is a concept very useful for community detection. By dividing a network into groups of maximal modularity, we are partitioning it in such a way that maximizes our “surprise” at seeing such groups in a random graph. The modularity of a partition is defined as the number of edges falling within partitions minus the expected number of edges in an equivalent (same number of vertices) network with edges placed at random, with a multiplicative constant. Suppose a network has a set of $n$ vertices $V$, and two vertices $v_i, v_j \in V$ have degrees $d_i$ and $d_j$. Now, suppose we wanted to build a new random network that we compare the original one to. We want our new network to have the same degree distribution and the same number of vertices, but we want to distribute edges randomly. If we were to pick a pair of vertices from $V$ uniformly at random and place an edge between them, until we had the same number of edges as the original network, the expected number of edges between $v_i$ and $v_j$ is $\frac{1}{2} \frac{d_i d_j}{\sum_{v_k \in V} d_k} = \frac{d_i d_j}{\sum_{v_k \in V} d_k}$. The denominator here is the total number of edges in the network, and the numerator is the number...
of ways to choose one vertex times the number of ways to choose the second vertex when we are picking two vertices to place an edge between. The factors of one half simply account for the fact that the order we pick vertices in doesn’t matter – if we pick vertex \( v_i \) and then \( v_j \) or \( v_j \) and then \( v_i \), we have the same combination of vertices and don’t want to count this pair twice.

Now that we have our random network model, we can define modularity formally in terms of the adjacency matrix \( A \). Suppose \( A_{ij} \) is the number of edges between vertices \( v_i \) and \( v_j \) in the original network, and assume that we have partitioned the network into two groups. Let \( s \) be an indicator vector, where \( s_i = 1 \) if \( v_i \) is in group 1 and \( s_i = -1 \) if \( v_i \) is in group 2. Since \( \frac{1}{2} (s_i s_j + 1) = 1 \) if \( v_i \) and \( v_j \) are in the same group and \( \frac{1}{2} (s_i s_j + 1) = 0 \) otherwise, the modularity, which Newman calls \( Q \), is the sum:

\[
Q = c \sum_{v_i \in V} \sum_{v_j \in V} \left( A_{ij} - \frac{d_i d_j}{\sum_{v_k \in V} d_k} \right) \frac{1}{2} (s_i s_j + 1) = c \sum_{v_i \in V} \sum_{v_j \in V} \left( A_{ij} - \frac{d_i d_j}{\sum_{v_k \in V} d_k} \right) s_i s_j
\]

where \( c \) is a constant that Newman defines as \( \frac{1}{2 \sum_{v_i \in V} d_i} \). This quantity can be re-written as:

\[
Q = s^T B s
\]

where \( s \) is the indicator vector defined above and \( B_{ij} = A_{ij} - \frac{d_i d_j}{\sum_{v_k \in V} d_k} \). \( B \) is the real, symmetric matrix that Newman calls the “modularity matrix”. Since its rows and columns sum to 0, it has an eigenvalue of 0 with the constant eigenvector \((1, 1, \ldots, 1)\). The indicator vector \( s \) can be rewritten as a combination of the normalized eigenvectors \( u_i \) with corresponding eigenvectors \( \beta_i \) of \( B \) so that:

\[
Q = c \sum_{v_i \in V} u_i^T s u_i^T B \sum_{v_j \in V} u_j^T s u_j = c \sum_{i=1}^{n} (u_i^T s)^2 \beta_i
\]

Assuming that the eigenvalues are ordered from largest, so that \( \beta_1 \geq \beta_2 \geq \ldots \geq \beta_n \), if we want to maximize \( Q \) we would want to place the most weight onto \( u_1 \), the eigenvector corresponding to \( \beta_1 \) in the sum above. However, we are limited by the constraint that \( s \) is an indicator vector with elements equal to 1 or -1, so we cannot make \( s \) “parallel” to \( u_1 \), and we must therefore involve terms involving other eigenvalues \( \beta \). To maximize the sum \( Q \), we want to maximize the outer product \( u_1^T s \), we can set all elements of \( s \) to +1 if the corresponding element of \( u_1 \) is positive and we set the elements of \( s \) to -1 if the corresponding element of \( u_1 \) is negative. This gives us a partition of the vertices which maximizes modularity in the case of two partitions.

Newman points out that modularity has the desirable property that it will find good communities when they exist, but if good communities do not exist, making the largest eigenvalue 0, the corresponding eigenvector is constant, placing all the vertices in one group, and so we can not only find communities but we can find out whether any communities actually exist. To divide a network into more than two communities, Newman adopts a recursive bisection approach,
where the network is repeatedly divided into two by maximizing the modularity with each split, accounting for the contribution to the overall modularity of additional splits with a value $\delta Q$, which Newman defines as follows:

$$
\Delta Q = \frac{1}{\sum_{v_i \in V} d_k} \left( \frac{1}{2} \left( \sum_{v_i \in V'} \sum_{v_j \in V'} B_{ij} (s_{ij} + 1) - \sum_{v_i \in V'} \sum_{v_j \in V'} B_{ij} \right) \right)
$$

where $V'$ is the set of vertices that comprise a partition in an earlier split of the recursive bisection algorithm. Newman shows that the same approach can be applied to maximize the value $\Delta Q$ that was applied to the original maximization of $Q$. He notes that $\Delta Q$ gives an intuitive and clear stopping condition for the recursive bisection process – if $\Delta Q$ fails to increase in any given iteration, then the algorithm has found the best modularity possible, and no further divisions will improve this value. Thus, the recursive bisection stops when $\Delta Q$ is less than or equal to 0.

Newman goes on to describe a refinement step in the modularity maximization algorithm which is reminiscent of the Kernighan-Lin algorithm – after a partitioning of the vertices, each vertex is moved once into the partition that results in the greatest increase or smallest decrease in the modularity $Q$. After all vertices have been moved, the partitioning of the vertices that gave the best modularity value in this process is saved.

A few experimental results are shown for real-world networks, and Newman gives two examples illustrating that modularity optimization gives meaningful results (one example considers the communities on Amazon.com that purchase similar politically aligned books, and the other looks at the (two opposing) communities present in political blogs). The spectral algorithm discussed above is found to outperform other modularity optimization algorithms for larger networks but performs about the same for networks on the scale of thousands of vertices.

In the implementation details, Newman notes that if the number of edges $m$ is much smaller than the number of vertices $n$ in a network, the runtime of his algorithm is $O(n^2)$ for the modularity maximization step and $O(n^2 \log n)$ for the entire recursive bisection process in practice (where the process stops when $\Delta Q$ is no longer positive).