Minimal Free Resolutions of Edge Ideals

David J. Marchette

1 Introduction

We will assume all graphs are undirected and simple: no self-loops or multiple edges. Given a graph, we can define the edge ideal as follows. Assign the labels $\{x_1, \ldots, x_n\}$ to the n vertices of the graph. The edges inherit this labeling: if there is an edge from vertex i to vertex j, this edge now has the label $x_i x_j$. Thus the set of edges is a set of square free quadratic monomials. We will utilize this observation to construct a set of graph invariants, called the graded minimal free resolution of the graph.

The package `mfr` contains code to compute minimal free resolutions of graphs, under certain conditions. For certain classes of graphs, there are algorithms to compute the minimal free resolution (MFR) without requiring more general commutative algebra code. In this vignette we will restrict ourselves to those cases where these algorithms exist. On a Unix machine, the Singular computational algebra system (`http://www.singular.uni-kl.de`, see [Greuel and Pfister(2002)] and decker:2011) allows the `mfr` package to compute the MFR much more generally. This vignette will not make use of Singular. The `mfr` package depends on the `igraph` package, and we will make use of the functionality that this provides us.

To get started, we need a few definitions.

Definition 1.1. *A graph is* chordal *if all induced cycles have at least one chord.*

Figure 1 shows smallest non-chordal graph, a cycle on 4 vertices. This can be made chordal by adding one edge, as in Figure 2.

Definition 1.2. *The* open neighborhood *of a vertex v, denoted $N(v)$, is the set of vertices $\{w \in V | vw \in E\}$. That is, it consists of all neighbors of v. Note that $v \notin N(v)$. The* closed neighborhood *of v, denoted $N[v]$ is the set $N(v) \cup \{v\}$.*

Definition 1.3. *The* trivial graph *is the graph with no edges; $E = \emptyset$.*

Definition 1.4. *The* complete graph *on n vertices K_n is the graph with all possible edges (this is `graph.full` in the package `igraph`). The* complete bipartite graph *$K_{m,n}$ is the graph on $m + n$ vertices in which the vertices are partitioned into two disjoint groups, of size m and n, and all edges are between groups (bipartite) and all possible such edges are present (complete). See `graph.full.bipartite` in `igraph`.*

```
> g <- graph.ring(4, directed = FALSE)
> plot(g, layout = layout.circle)
```

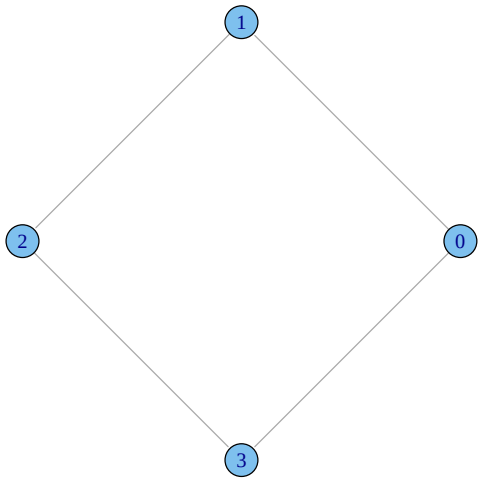

Figure 1: A 4-cycle, a graph which is not chordal.

```
> h <- add.edges(g, c(1, 3))
> plot(h, layout = layout.circle)
```

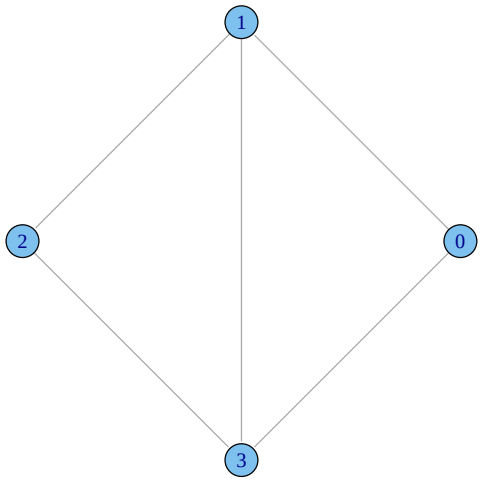

Figure 2: A chordal graph.

Definition 1.5. *Given a graph G, define* star(G) *to be the graph defined by adding a single vertex v to G, with edges from v to each vertex of G. We will denote by* star(n) *the star on the trivial graph on n vertices. Thus star$(n) = K_{1,n}$ has order $n+1$ and contains one vertex of degree n and n vertices of degree 1.*

Note: our definition of `Star` differs slightly from `graph.star`, and is more general.

```
> graph.star(4, mode = "undirected")

Vertices: 4
Edges: 3
Directed: FALSE
Edges:

[0] 0 -- 1
[1] 0 -- 2
[2] 0 -- 3

> Star(3)

Vertices: 4
Edges: 3
Directed: FALSE
Edges:

[0] 0 -- 1
[1] 0 -- 2
[2] 0 -- 3
```

Note that in this case `Star(n) = graph.star(n+1,mode="undirected")`.

```
> Star(Star(3))

Vertices: 5
Edges: 7
Directed: FALSE
Edges:

[0] 0 -- 1
[1] 0 -- 2
[2] 0 -- 3
[3] 0 -- 4
[4] 1 -- 4
[5] 2 -- 4
[6] 3 -- 4
```

Definition 1.6. *A vertex v is* simplicial *if $N(v) = K_n$ for some n.*

Simplicial vertices will be the main tool we can use to make minimal free resolutions computable for a large number of graphs. Essentially, as we will see, as long a graph contains a simplicial vertex, the MFR of the graph can be computed in terms of the MFRs of two smaller graphs. This gives a recursive method for computing the MFR. We now proceed to define the MFR of a graph.

2 Commutative Algebra

Suppose we have a graph G on n vertices $\{v_1, \ldots, v_n\}$. Let k be a field (we may assume for the purposes of this paper that $k = \mathbb{Q}$). Let $S = k[x_1, \ldots, x_n]$, the ring of polynomials in n variables with coefficients in k. We will use the notation v_i to denote a vertex and x_i the corresponding variable, so that it is clear when we are talking about a graph, and when we are considering the algebraic object S.

If you are not familiar with rings, ideals, modules and exact sequences, you can skip this section and simply think of Betti numbers as invariants of the graph that can be computed via various formulas. See for example [Dummit and Foote(2004), Eisenbud(1994)].

Definition 2.1. *The* edge ideal *of G is the ideal of S generated by the monomials $\{x_i x_j | v_i v_j \in E\}$. We write $\mathcal{I}(G)$ for the edge ideal of G.*

For more information on edge ideals see [Jacques(2004)], [Jacques and Katzman(2005)], [Stanley(1996)], [Villarreal(2001)], [Miller and Sturmfels(2005)].

We can now use the tools of commutative and algebraic geometry to learn about a graph by studying its edge ideal. One such tool, the one we will be focused on in this paper, is the minimal free resolution (MFR).

Definition 2.2. *An* augmented free resolution *of an S-module M is an exact sequence of the form*

$$0 \longleftarrow M \longleftarrow F_0 \longleftarrow F_1 \longleftarrow F_{m-1} \cdots \longleftarrow F_m \longleftarrow 0$$

where each F_i is a free S-module (a direct sum of β_i copies of S). The image of F_i in the sequence is called the i^{th} syzygy module*. Such a resolution is* minimal *if m is minimal over all such, and each β_i is minimal. We define the* minimal free resolution *of the edge ideal to be the minimal free resolution of $I = \mathcal{I}(G)$, in which case the free resolution becomes:*

$$0 \longleftarrow I \xleftarrow{\phi_0} S^{\beta_0} \xleftarrow{\phi_1} S^{\beta_1} \xleftarrow{\phi_2} \cdots \xleftarrow{\phi_{m-1}} S^{\beta_{m-1}} \xleftarrow{\phi_m} S^{\beta_m} \longleftarrow 0$$

The β_i are called the Betti numbers*. The length of the resolution is m. We will refer to these as* total Betti numbers *to distinguish them from their graded versions discussed below.*

It is well known that minimal free resolutions always exist, and are unique up to isomorphism ([Miller and Sturmfels(2005)]). Thus, the length of the minimal free resolution of an edge ideal is well defined.

Definition 2.3. *The* projective dimension *of an edge ideal is the length of the minimal resolution.*

Note that $\beta_0 = \text{size}(G)$. There is a natural $\mathbb{N}$ grading on the ring $k[x_1, \ldots, x_n]$, which gives a natural grading on the resolution.

$$0 \longleftarrow I \longleftarrow \bigoplus_j S(-j)^{\beta_{0,j}} \longleftarrow \bigoplus_j S(-j)^{\beta_{1,j}} \longleftarrow \cdots \longleftarrow \bigoplus_j S(-j)^{\beta_{m,j}} \longleftarrow 0,$$

where $S(-j)$ is the shifted module obtained by shifting the degrees by j, so that the corresponding maps remain degree 0. We will be concerned with methods for computing the graded Betti numbers $\beta_{i,j}$. The total Betti number β_i is the sum over j of $\beta_{i,j}$.

3 Splitting

This section provides the main tool we will use to compute the Betti numbers for our graph edge ideal. Theorem 3.1 provides the main recursive algorithm, and Theorem 3.2 shows how to implement the algorithm to compute the minimal free resolutions of chordal graphs.

Definition 3.1. *An edge uv is a* splitting edge *if $N[u] \subset N[v]$ or $N[v] \subset N[u]$. If uv is a splitting edge, we will assume that the vertices are ordered so that $N[u] \subset N[v]$.*

Theorem 3.1 ([Hà and Van Tuyl(2006a)])**.** *If uv is a splitting edge of G, then for all $i \geq 1$ and $j \geq 0$*

$$\beta_{i,j}(\mathcal{I}(G)) = \beta_{i,j}(\mathcal{I}(G \setminus \{uv\})) + \sum_{k=1}^{i} \binom{n}{k} \beta_{i-1-k,j-2-k}(\mathcal{I}(H)), \tag{1}$$

where $n = |N[v]| - 2$, $H = G \setminus N[v]$, $\beta_{-1,0} = 1$ and $\beta_{-1,j} = 0$ for $j > 0$. Recall that we are using the convention that uv is ordered so that $N[u] \subset N[v]$.

Proof. See [Hà and Van Tuyl(2006a)]. □

Lemma 3.1. *A graph G has the property that all (non-trivial) induced subgraphs H contain a splitting edge if and only if G is chordal. Here "non-trivial" refers to the condition that H contain at least one edge.*

Proof. ($\Rightarrow$) Assume G is not chordal. Then there is an induced cycle C_n with $n > 3$ with no chord. But it is easy to see that any such cycle does not have a splitting edge.

($\Leftarrow$) If G is chordal, then it contains a simplicial vertex u. Since $N(u)$ is a clique, uv is a splitting edge for any $v \in N(u)$. Since any induced subgraph of a chordal graph is chordal, we have the result. □

Lemma 3.2. *For any chordal graph G there is a splitting edge e such that $G \setminus \{e\}$ is chordal. In fact, any edge incident on a simplicial vertex may be chosen for e.*

Proof. The only way removing an edge from G can make it non-chordal is if it opens up an induced C_4 with no chord. So in particular, we want to avoid removing chords. Let v be a simplicial vertex. Any edge e incident to v is a splitting edge. Further, it is not the chord of any cycle external to $N[v]$. Since $N[v]$ is complete, removing e cannot result in an induced cycle without a chord. Thus, any edge incident to a simplicial vertex can be removed. □

We call a function s that takes a graph containing splitting edges and returns one splitting edge a *splitting edge selection strategy*, or simply a *strategy*. We will say that Equation (1) is *recursive* for a class of graphs if there is a strategy for which it is recursive. We do not require that any arbitrary strategy will work, only that there is at least one. The following theorem has been stated elsewhere (see [Hà and Van Tuyl(2006b)]), but is usually stated without explicit proof.

Theorem 3.2. *Under the strategy which selects splitting edges incident on simplicial vertices, Equation (1) is recursive for a graph G if and only if G is chordal.*

Proof. This is immediate from the two lemmas and the fact that induced subgraphs of chordal graphs are chordal. □

This shows that the algorithm to apply Equation (1) recursively always works to compute the minimal resolution of G whenever G is chordal, provided the splitting edges are chosen appropriately. It is not the case, though, that it is recursive no matter what splitting edge is chosen at each step. The simplest example of this is shown in Figure 3.

It is clear that all edges in this graph are splitting and that this is a chordal graph. Further, if we remove any edge from the outside cycle, any uv_i or vv_i, the resulting graph is chordal. However, if we remove the diagonal, uv, the resulting graph is not chordal, and furthermore, does not contain any splitting edges. The two degree 2 vertices are simplicial, while the vertices on which the diagonal are incident are not. The minimal free resolution is shown in Table 2.

```
> m <- mfr(h)
> z <- m$graded
> rownames(z) <- c("$\\beta_{1 \\cdot}$", "$\\beta_{2 \\cdot}$")
> colnames(z) <- c("$\\beta_{\\cdot 1}$", "$\\beta_{\\cdot 2}$",
+     "$\\beta_{\\cdot 3}$", "$\\beta_{\\cdot 4}$")

> mat <- xtable(z, caption = "Graded minimal free resolution for h",
+     display = c("s", rep("d", ncol(m$graded))), label = "table:mfrstar")
> print(mat, sanitize.text.function = function(x) {
+     x
+ })
```

```
> g <- graph.ring(4, directed = FALSE)
> h <- add.edges(g, c(1, 3))
> plot(h, layout = layout.circle, vertex.label = c(expression(v[1]),
+     "u", expression(v[2]), "v"))
```

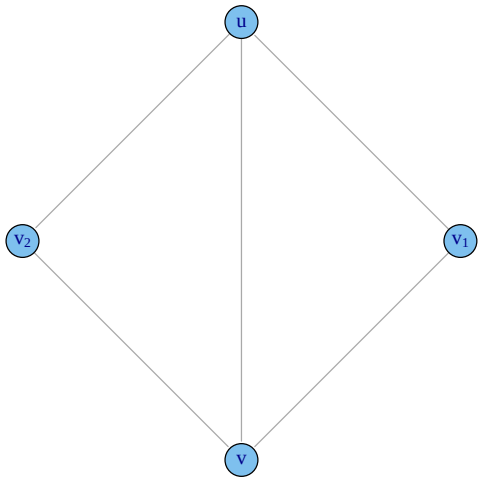

Figure 3: A chordal graph where all edges are splitting.

	$\beta_{\cdot 1}$	$\beta_{\cdot 2}$	$\beta_{\cdot 3}$	$\beta_{\cdot 4}$
$\beta_{1\cdot}$	1	0	0	0
$\beta_{2\cdot}$	0	5	6	2

Table 1: Graded minimal free resolution for h

There are a few things to note about the MFR displayed in Table 1.

1. All the (nontrivial) Betti numbers are on a single row. In this case the resolution is called *linear*. It turns out that whenever the complement of a graph is chordal, the resolution of the graph is linear (and there is a simple formula to compute the MFR). In fact, this is if and only if: the resolution is linear if and only if the complement is chordal (see [Horwitz(2007)] and [Dochtermann and Engstroöm]). In the case we are considering, the complement of the graph is a the graph with the single edge v_1v_2 which is clearly a chordal graph.

2. $\beta_{2,2}$ is the size of the graph (the number of edges). This is always true.

3. Define an angle to be an induced subgraph on three vertices containing exactly two edges. There are two angles ($\angle$) in this graph, the induced subgraphs of $\{u, v_1, v_2\}$ and $\{v, v_1, v_2\}$ and there are two triangles (Δ), $\{u, v_i, v\}$ for $i = 1, 2$. In this case, we see that $\beta_{2,3} = \#\angle + 2\#\Delta$. This is also always the case.

4. Let the graph on four vertices $\{v_1, v_2, v_3, v_4\}$ with edges v_1v_2 and v_3v_4 be called "bars". There are no induced bars subgraphs in this graph, and it turns out the $\beta_{3,3} = \#$bars. In fact, all Betti numbers can be viewed as some combination of counts of various subgraphs, but it is not trivial to determine this association for a given Betti number.

Note that the above leads one to posit that $\beta_{2,4}$ is the number of induced subgraphs isomorphic to `h`. We can test this by:

```
> k1 <- graph.disjoint.union(h, h)
> m1 <- mfr(k1)$graded
> k2 <- graph.disjoint.union(h, graph.ring(4, directed = FALSE))
> m2 <- mfr(k2)$graded
> k3 <- graph.disjoint.union(h, graph.ring(4, directed = FALSE),
+     graph.ring(4, directed = FALSE))
> m3 <- mfr(k3)$graded
> k4 <- add.edges(k3, c(0, 6))
> m4 <- mfr(k4)$graded
> k5 <- add.edges(k3, c(3, 6))
> m5 <- mfr(k5)$graded

> xtable(m1, caption = "Graded minimal free resolution for the disjoint union of 2 copies o
+     display = c(rep("d", ncol(m1) + 1)), label = "table:mfr1")
```

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	10	12	4	0	0	0
3	0	0	25	60	56	24	4

Table 2: Graded minimal free resolution for the disjoint union of 2 copies of h

```
> xtable(m2, caption = "Graded minimal free resolution for the disjoint union of h and a 4-c
+     display = c(rep("d", ncol(m2) + 1)), label = "table:mfr2")
```

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	9	10	3	0	0	0
3	0	0	20	44	37	14	2

Table 3: Graded minimal free resolution for the disjoint union of h and a 4-cycle

```
> xtable(m3, caption = "Graded minimal free resolution for the disjoint union of h and two 4
+     display = c(rep("d", ncol(m3) + 1)), label = "table:mfr3")
```

	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0	0	0	0
2	0	13	14	4	0	0	0	0	0	0
3	0	0	56	120	98	36	5	0	0	0
4	0	0	0	80	256	344	248	101	22	2

Table 4: Graded minimal free resolution for the disjoint union of h and two 4-cycles

From these examples we might think that $\beta_{2,4} = 2\#h + \#(4-cycles)$, but it is more complicated than this, as seen if Table 5.

```
> xtable(m4, caption = "Graded minimal free resolution for the disjoint union of h and two 4
+     display = c(rep("d", ncol(m4) + 1)), label = "table:mfr4")
```

```
> xtable(m5, caption = "Graded minimal free resolution for the disjoint union of h and two 4
+     display = c(rep("d", ncol(m5) + 1)), label = "table:mfr5")
```

As Tables 5 and 6 make clear, the Betti numbers are computing something that is very dependent on the details of the structure of the graph.

	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0	0	0	0
2	0	14	18	6	0	0	0	0	0	0
3	0	0	56	136	122	48	7	0	0	0
4	0	0	0	64	224	320	240	100	22	2

Table 5: Graded minimal free resolution for the disjoint union of h and two 4-cycles, with an added edge (k4)

	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0	0	0	0
2	0	14	19	8	1	0	0	0	0	0
3	0	0	54	135	130	60	13	1	0	0
4	0	0	0	56	196	282	215	92	21	2

Table 6: Graded minimal free resolution for the disjoint union of h and two 4-cycles, with an added edge (k5)

```
> x <- layout.circle(h)
> x1 <- x
> x2 <- x
> x1[, 1] <- x[, 1] + 3
> x2[, 1] <- x[, 1] + 6
> y <- rbind(x, x1, x2)
```

Figure 4 depicts the three graphs k3, k4 and k5. As can be seen in this Figure and Tables 4–6, very small differences in the graphs can produce large differences in the minimal free resolutions, although the basic structure of the resolutions are very similar.

4 Special Cases

The package contains code to compute the MFR of a number of special types of graphs. The recursive code discussed above applies to all chordal graphs, including trees. Also, the MFR of the disjoint union of graphs is computable as a convolution of the MFRs of the individual components, so the code operates on individual components of the graph separately, then puts the overall MFR together from the individual parts.

Several other types of graphs whose MFRs are easily computable using relatively simple formulas are:

- Empty graphs and complete graphs.
- Complete bipartite graphs.

```
> par(mfrow = c(3, 1))
> par(mar = c(0, 0, 0, 0))
> plot(k3, layout = y, vertex.label = "", vertex.size = 1)
> plot(k4, layout = y, vertex.label = "", vertex.size = 1)
> plot(k5, layout = y, vertex.label = "", vertex.size = 1)
> par(mfrow = c(1, 1))
```

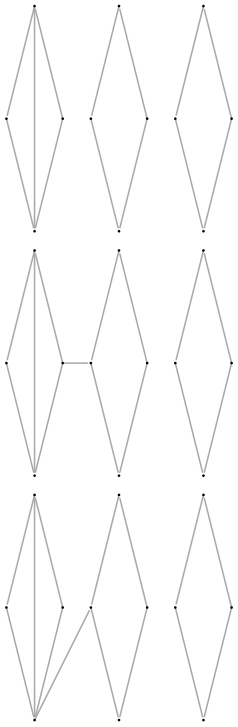

Figure 4: The three graphs k3, k4 and k5.

- Cycles.
- Paths and stars. Although these are also computable using the recursive algorithm for chordal graphs, there are explicit formulas for these special cases which are coded directly.
- Graphs whose complement is a chordal graph.

In addition to this, the package contains a database of graphs for which the minimal free resolutions have been precomputed. If the graph is isomorphic to one of these, the MFR is returned without any calculations. This means that if a graph is not chordal there is still some hope of calculating the MFR. Splitting edges will be removed using the algorithm of Theorem 3.2 until no splitting edges are available. If at any time the graph is one of the special types given above, the MFR is computed using the specialized code. Otherwise the graph is checked against the database, and if found, the MFR is returned.

If none of these approaches is applicable, then the result depends on whether Singular is installed and available on the computer. If so, Singular is called to compute the MFR. From a practical point of view, this will only work (in a reasonable amount of time) if the graph has no more than about 20 vertices and about 40 edges. Much more than this can take an inordinate amount of time.

If Singular is not available, an "approximation" is returned. Here the word "approximation" is not used in a technical sense – there is no theory that tells us how close the resulting estimated MFR is to the true value. Essentially what the algorithm does is to pretend one of the edges is splitting (using a heuristic to pick this) and then proceed as if it were splitting. It is true that sometimes this gives the correct MFR, but there are no guarantees, and as Tables 5 and 6 have shown, very small changes to the graph can have large effects on the MFR, and similarly using non-splitting edges in the recursion can have large effects on the MFR.

Let's look at this "approximation" with Tables 5 and 6 in mind. First we'll compute the MFR of a graph that requires a call to Singular. We'll pick a graph in the database, so the MFR has been precomputed.

```
> z <- mfr(graph.famous("Cubical"))$graded
> mca <- mfr(graph.famous("Cubical"), check.database = FALSE, nocode = TRUE,
+     suppress.warning = TRUE)$graded

The graph is not chordal, and nocode==TRUE
The results are likely to be approximate
```

So now `z` is the true MFR and `mfa` is the approximation. We have told `mfr` not to look in the database (something that one typically only does for these types of examples, or if the database becomes corrupted or needs to be regenerated), and we've also specified that it is not to try to run Singular, even if we have it.

```
> xtable(z, caption = "Minimal free resolution for the Platonic graph of the cube.",
+     display = rep("d", ncol(z) + 1), label = "table:cube")
```

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	12	24	14	0	0	0
3	0	0	6	24	32	16	3

Table 7: Minimal free resolution for the Platonic graph of the cube.

```
> xtable(mca, caption = "Approximate minimal free resolution for the Platonic graph of the c
+     display = rep("d", ncol(mca) + 1), label = "table:cubea")
```

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	12	19	8	0	0	0
3	0	0	15	40	39	17	3

Table 8: Approximate minimal free resolution for the Platonic graph of the cube.

Tables 7 and 8 give a feel for the types of errors that occur when using the "approximation". The bottom line is that if possible one should either stick to graphs that can be processed by the code (chordal graphs, graphs whose complement is chordal, cycles, etc., small graphs contained in the database, etc.) or obtain Singular and make sure it interfaces correctly with the `mfr` code. Future releases of this package will (we intend) have a version of the resolution code that Singular uses, so that the reliance on Singular for some graphs will go away.

5 Scan Statistics

One way to reduce the calculations of the MFR of a graph is to give up on trying to get the MFR of the full graph and instead focus on subgraphs. Suppose that one wishes to use the MFR to make some type of inference on a graph, where the inference in question is related to local structure of the graph rather than global structure. For example, consider the goal of determining whether a graph is "random" (by some given definition), or alternatively that there is a small group of vertices whose induced subgraph has a non-random structure, or at least a different structure than the overall graph as a whole.

The graph in Figure 5 is a kidney-egg graph: the graph is an independent edge random graph, like `erdos-renyi-game`, in which there are two groups of vertices: K containing 25 vertices and E containing 5 vertices. The edge probabilities are $p = 0.1$ for any edge that is not between two vertices in E. For pairs of vertices in E, the edge probability is $q = 0.8$.

```
> set.seed(12345)
> g <- kidney.egg.game(30, 0.1, 5, 0.8)
> plot(g)
```

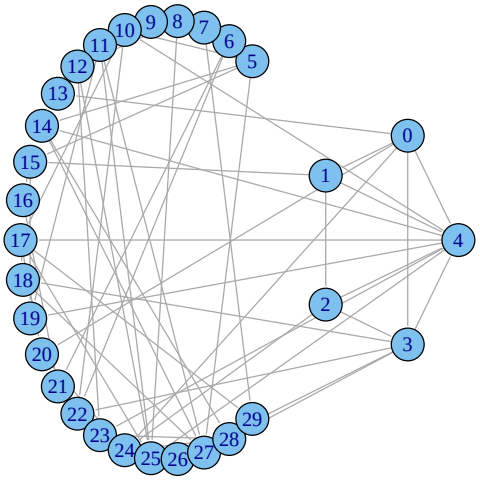

Figure 5: A kidney-egg graph with $n = 30$, $p = 0.1$, $m = 5$, $q = 0.8$.

We cannot compute the MFR of this graph directly, unless we are lucky and find that it is in the database (since the database may grow in time, it may be that this graph can be processed without Singular in your version, but let us assume the contrary, as is the case at the time of writing). We'd like to have a test that can determine whether a graph is Erdös-Renyí or whether there is a small group of vertices with a larger inter-connectivity probability. We could do this by using scan statistics.

The idea of a scan statistic on graphs was introduced in [Priebe et al(2005)]. One computes a statistic (graph invariant) on the induced subgraphs of the neighborhoods of the vertices, and computes the maximum over these. The function `scanMFR` allows us to do this in several ways.

```
> set.seed(12345)
> g <- kidney.egg.game(30, 0.1, 5, 0.8)
> unlist(lapply(neighborhood(g, order = 1), length))

 [1]  7  5  5  8 11  5  3  2  5  1  6  6  4  3  5  5  3  7  4  4  4  3  4  5  5
[26]  8  2  6  5  3

> unlist(lapply(neighborhood(g, order = 1), function(x) ecount(subgraph(g,
+     x))))

 [1]  9  6  6 10 17  5  2  1  6  0  6  6  5  2  5  4  2  8  4  3  4  2  3  7  5
[26] 13  1  7  5  2
```

The neighborhoods are all small, so we are confident we will be able to compute the MFR of any neighborhood. One thing we might do is to pick the neighborhood with the largest number of edges, and compute the MFR of this. We could then compare this with the results of running such a computation on a large number of Erdös-Renyí random graphs. By performing the Monte Carlo on both a large number of Erdös-Renyí random graphs (the null hypothesis) and a large number of kidney-egg graphs (with various parameters) we can determine the Betti numbers which are most powerful for performing the desired inference.

For now, let's just see how to apply `scanMFR`.

```
> set.seed(12345)
> g <- kidney.egg.game(30, 0.1, 5, 0.8)
> h <- erdos.renyi.game(30, 0.1)
> m1 <- scanMFR(g)$graded
> m2 <- scanMFR(h)$graded

> xtable(m1, caption = "Scan minimal free resolution for a kidney-egg graph.",
+     display = rep("d", ncol(m1) + 1), label = "table:scan1")


> xtable(m2, caption = "Scan minimal free resolution for a random graph.",
+     display = rep("d", ncol(m2) + 1), label = "table:scan2")
```

	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	0	0	0	0	0	0	0	0
2	0	17	58	127	211	252	210	120	45	10	1
3	0	0	12	32	31	13	2	0	0	0	0

Table 9: Scan minimal free resolution for a kidney-egg graph.

	1	2	3	4	5	6	7	8
1	1	0	0	0	0	0	0	0
2	0	10	25	36	35	21	7	1
3	0	0	2	3	1	0	0	0

Table 10: Scan minimal free resolution for a random graph.

Alternatively, we could compute the MFR of every neighborhood, and take the maximum Betti number across these neighborhoods.

```
> m3 <- scanMFR(g, method = "maximum")$graded
> m4 <- scanMFR(h, method = "maximum")$graded

> xtable(m3, caption = "Scan minimal free resolution for a kidney-egg graph using method=\"m
+     display = rep("d", ncol(m3) + 1), label = "table:scan3")
```

	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	0	0	0	0	0	0	0	0
2	0	17	58	127	211	252	210	120	45	10	1
3	0	0	12	32	31	13	2	0	0	0	0

Table 11: Scan minimal free resolution for a kidney-egg graph using method=”maximum”.

```
> xtable(m4, caption = "Scan minimal free resolution for a random graph using method=\"maxim
+     display = rep("d", ncol(m4) + 1), label = "table:scan4")
```

In this case there is no difference, because the vertex with maximum size and order is unique, and contributes all the large Betti numbers.

References

[Balakrishnan and Ranganathan(2000)] R. Balakrishnan and K. Ranganathan. *A Textbook of Graph Theory*. Springer, New York, 2000.

	1	2	3	4	5	6	7	8
1	1	0	0	0	0	0	0	0
2	0	10	25	36	35	21	7	1
3	0	0	2	3	1	0	0	0

Table 12: Scan minimal free resolution for a random graph using method="maximum".

[Decker et al(2011)] W. Decker, G.˜M. Greuel, G. Pfister and H. Schonemann. *Singular 3-1-3 - A computer algebra system for polynomial computations*. `http://www.singular.uni-kl.de`, 2011.

[Dochtermann and Engstroöm] Anton Dochtermann and Alexander Engstroöm. Algebraic properties of edge ideals via combinatorial topology, the electronic journal of combinatorics 16(2), 2009.

[Dummit and Foote(2004)] David˜S. Dummit and Richard˜M. Foote. *Abstract Algebra*. John Wiley & Sons, Inc, Hoboken, NJ, Third Edition, 2004.

[Eisenbud(1994)] David Eisenbud. *Commutative Algebra: with a view toward algebraic geometry*. Springer, New York, 1994.

[Greuel and Pfister(2002)] David Eisenbud. *A Singular Introduction to Commutative Algebra*. Springer, Berlin, 2002.

[Hà and Van Tuyl(2006a)] Huy˜Tài Hà and Adam Van Tuyl. Splittable ideals and the resolutions of monomial ideals, 2006a.

[Hà and Van Tuyl(2006b)] Huy˜Tài Hà and Adam Van Tuyl. Monomial ideals, edge ideals of hypergraphs, and their minimal graded free resolutions, 2006b.

[Horwitz(2007)] Noam Horwitz. Linear resolutions of quadratic monomial ideals, J. Algebra 318, 2007, 981–1001.

[Jacques(2004)] Sean Jacques. *Betti numbers of graph ideals*. PhD thesis, University of Sheffield, 2004.

[Jacques and Katzman(2005)] Sean Jacques and Mordechai Katzman. The betti numbers of forests, 2005. URL `http://arxiv.org/pdf/math.AC/0501226.pdf`.

[Miller and Sturmfels(2005)] Ezra Miller and Bernd Sturmfels. *Combinatorial Commutative Algebra*, volume 227 of *Graduate Texts in Mathematics*. Springer, New York, 2005.

[Priebe et al(2005)] Carey˜E. Priebe, John˜M. Conroy, David˜J. Marchette and Youngser Park. *Scan Statistics on Enron Graphs*, *Computational and Mathematical Organization Theory, 11, 229–247, 2005.*

[Stanley(1996)] Richard˜P. Stanley. Combinatorics and Commutative Algebra*. Birkhäuser, Basel, second edition, 1996.*

[Villarreal(2001)] Rafeal˜H. Villarreal. Monomial Algebras*, volume 238 of* Monographs and Textbooks in Pure and Applied Mathematics*. Marcel Dekker, Inc., New York, 2001.*