Graphs and Networks Lecture 1

Introduction

Daniel A. Spielman August 29, 2013

1.1 Disclaimer

These notes are not necessarily an accurate representation of what happened in class. They are a combination of what I intended to say with what I think I said. They have not been carefully edited.

On the course web page, I listed readings for today's lecture about clustering coefficients and assortativity. We didn't manage to discuss them today. We will get to them later.

You should be able to find a diary of my Matlab session from today's class. It may reveal computations that do not appear in these notes.

1.2 Introduction

First, let me admit that the title of the course is redundant. "Graphs" and "Networks" are the same things.

Everyone should look at the syllabus. A few important points are:

- The course will be more difficult than when it was last taught.
- There are now more prerequisites. The problem sets will require both writing proofs and doing simple computational experiments. To follow lectures, and to do the problem sets, you should be familiar with graphs, linear algebra, and probability. I will distribute the first problem set after the second lecture.
- In this lecture, and in many others, I will do examples in Matlab. However, you are free code in any language or system that you like.
- There is no exam in this course.
- This course covers a very quickly developing field. As a result, I am actively re-organizing the class. I will probably mess up some lecture. The best lectures will involve experiments, proofs, and presentations of empirical studies.

1.3 How to look at a graph

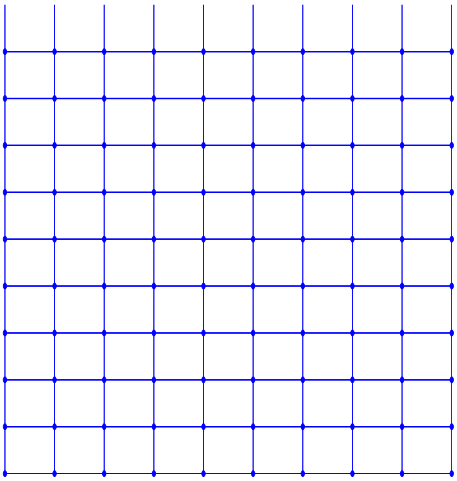
A graph can be a very difficult thing to understand. We often begin to understand a graph by taking some measurements of it. The first things to measure are the numbers of vertices and the numbers of edges. This is assuming that the graph is connected. If it is not, one should first break it into connected components.

In this lecture, we will also examine the distributions of the degrees of the vertices. We will examine this much more carefully a few lectures from now.

Another way to understand a graph is to draw it. For example, we can understand a lot about this graph from a picture of it.

This picture tells us everything we need to know about the grid graph.

```
[a,jnk,xy] = grid2(10,11);
clf; hold on;
gplot(a,xy);
axis off
px = plot(xy(:,1),xy(:,2),'o','MarkerFaceColor','b');
```



Unfortunately, it is a theorem (that we will see later this semester) that it is impossible to make nice drawings of most graphs.

I will discuss more ways to understand graphs as we examine them.

1.4 Where Our Graphs Come From

We will often think about social network graphs. The small ones are gathered by real people asking other real people questions. It will be easier for us to gather electronic social networks, like Facebook graphs.

1.4.1 Web graphs

Another natural graph to consider is the web. Web pages are vertices, and links can be viewed as directed edges. As this is a directed graph, its structure is more complicated. We can partition its vertices into strongly connected components, and these link to each other acyclicly. For some stats on an early web graph computed by a crawl done by Altavista, look at [BKM$^+$00]. Two things that stand out are that there is a very large strongly connected component and that most nodes are not too far from each other.

Thinking about big graphs reveals a methodological problem: it is very difficult to get the whole graph. Instead, a "crawl" produces just a fragment of the graph. The behavior at the boundary can be very different from that at the nodes where the crawl starts, and there is no easy way to discover vertices linking to those we've explored. The situation is worse with graphs gathered by humans (surveys or social scientists): some studies only collect a few of the contacts of each person. So, the fact that a graph does not have a particular edge does not mean that the edges does not exist: it just means that it was not gathered. Issues like these can systematically bias the results of studies. We will examine some cases of this later in the semester.

1.4.2 Technological

The Web should not be confused with the Internet. The Internet is the hardware that hosts the web. It can be viewed as a graph on computers and routers, with edges representing physical connections between them.

1.4.3 Collaboration

We will sometimes look at collaboration networks. The most popular is the graph on actors. The actors are the vertices, and an edge appears between two of them if they have performed in a movie together. This graph is actually derived from a bipartite graph in which the vertices on one side are the actors and the vertices on the other are movies. An edge appears between an actor and a movie if the actor appeared in that movie. We may look at some of this data if I can get it permission to use it.

For now, we will consider a scientific collaboration network. We start with a bipartite graph with authors on one side and papers on the other. Edges link papers to their authors. We can then turn this into a network on just the authors by connecting each pair of authors who have co-authored a paper. This naturally leads to a graph with weights on its edges: we can weight the edge between

two authors by the number of papers that they have written together.

In fact, I suspect that almost every analysis of a graph can be improved by choosing the right weights for its edges. Or, to put it differently, choosing to treat every edge equally is just as arbitrary a choice as assigning weights intelligently. For almost every graph we will study, there is an intelligent choice of weights for edges. However, sometimes we will not know that that choice is.

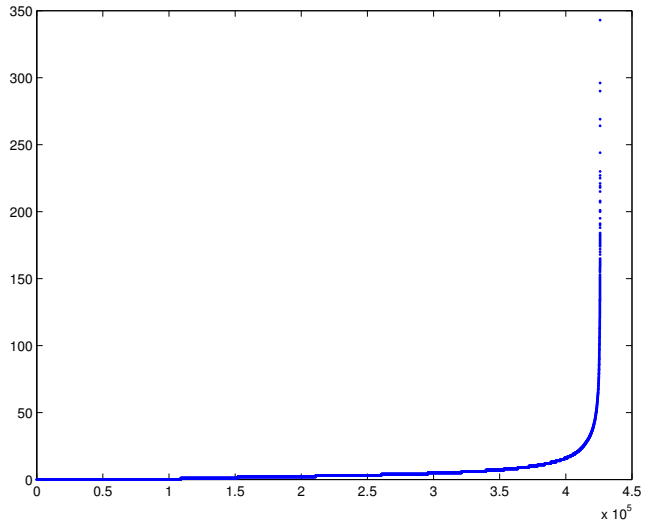
Today, I give you a co-authorship graph called `dblp`, which comes from the SNAP [Les] library.

Let's examine the degrees of the vertices a little. I have stored the graph in a sparse adjacency matrix called `a`.

```
>> size(a)

ans =

      425957      425957

>> degs = sum(a);
>> max(degs)

ans =

   (1,1)      343

>> plot(sort(degs),'.')
```

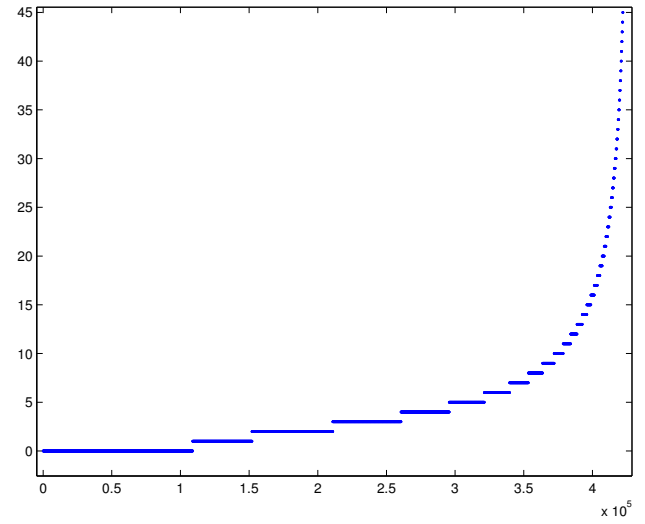
We see that `a` has 425957 vertices, and the largest degree of a vertex is 343. To get a feel for the distribution of the degrees, I will sort them and then plot them.

```
>> plot(sort(degs),'.')
```



As I will often use plots like this to examine lists of numbers, let me explain it some more. Consider the point with horizontal coordinate 4×10^5 and vertical coordinate 16. It means that the 4×10^5-th smallest degree is 16.

Here is a zoom of a part of the plot that better reveals the lower degrees.



1.4.4 Artificial Graphs

We often use graphs as an abstraction of data that do not naturally occur as a graph. For example, the vertices of the graph `amazon` are products sold by Amazon, and edges appear between products that are often purchased together (this graph also come from the SNAP library). For this graph,

we also have annotations on the vertices. In this case, products are organized into a hierarchy, and we know the sets in the hierarchy. The edges in the graph mostly go between vertices in the same part of the hierarchy.

To see this, let's take one of the larger sets of the hierarchy.

```
% the graph is in the matrix a
% the rows of matrix index products.  the columns index sets in the hierarchy

>> degs = sum(a);
>> [val,ord] = sort(sum(matrix>0));
>> val(end)

ans =

   (1,1)      328

% the largest set in the hierarchy has 328 products

>> ind = ord(end)

ind =

        4876
>> s = logical(matrix(:,ind));
>> sum(degs(s))

ans =

   (1,1)           1403

% there are 1403 edges attached to products in this set

>> nnz(a(s,s))

ans =

        1358

% and, 1358 of them connect to other products in the set
```

So, most of the edges attached to the vertices in s go to other vertices in s.

I will be most interested in studying graphs whose vertices or edges have annotations. It will provide a ground truth for our studies of graphs.

1.4.5 Protein-Protein Interaction Networks

Using BioGrid [SBR+06], I have constructed a graph of the interactions of the proteins in S. Cerevisiae. I did this following the methodology of Song and Singh [SS09]. There are multiple types of interactions: physical, chemical, and detected by various methods. For this graph, I have included all. This graph has 2 connected components. One has only 3 vertices. You can discover this using the routine `components` from the Matlab BGL library.

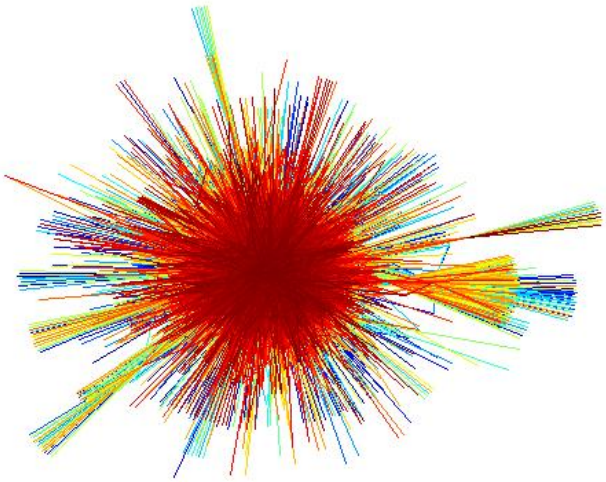
```
>> [ci, sizes] = components(a);
>> max(ci)

ans =

     2

>> sizes

sizes =

        6545
           3
```

While the graph is nice enough, it is much more interesting to see if the structure of the graph is related to the functions of the proteins. To examine this, we will grab annotations of the proteins from the Gene Ontology Project [Con00]. These tell us something like

```
Mitochondrial inner membrane ADP/ATP translocator.
```

Most annotation only appear once or twice. We will consider those that appear more often, say between 3 and 50 times. We will ask if those with the same annotation are nearby in the graph.

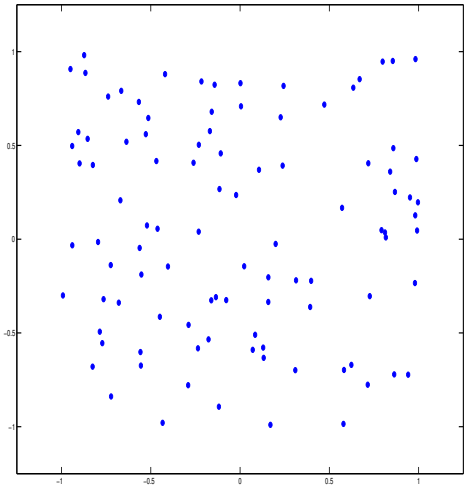
For now, I just give you a picture of larger connected component of the graph. It was generated by calling `sfdp` [Hu05] from the `GraphViz` package. I created the input and output for it using the my Matlab code `dotFile` and `readPlain`. Type `help dotFile` in Matlab to see how to do this.



1.5 Geometric Graphs

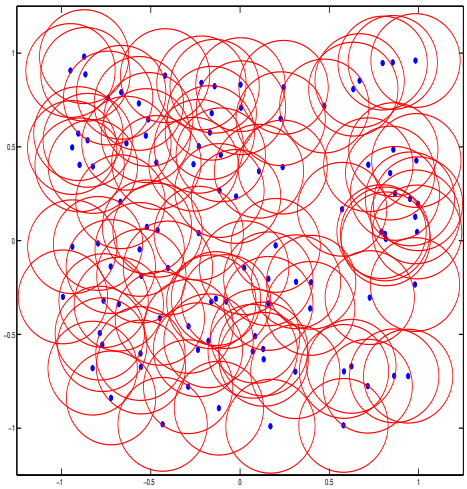
The geometric graphs we will examine come from two sources. The first is from sensor networks. Consider a large number of mobile computational devices (called sensors) whose radios have a limited range. We will draw each as a dot. I choose their locations at random.

```
x = 2*rand(100,2)-1;
rad = .25;
clf;
px = plot(x(:,1),x(:,2),'o','MarkerFaceColor','b');
axis(axis*(1+rad))
```



We now draw a circle around each to indicate the reach of its radio.

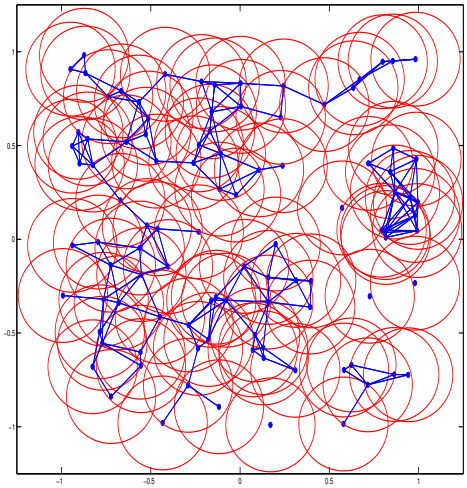
```
hold on
hc = drawCircles(x,rad);
```



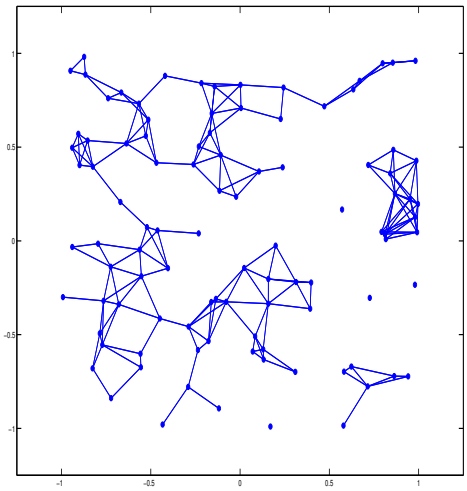
Whenever one sensor is inside the circle of another, they can communicate. We now draw the graph of their allowable communications.

```
a = radGraph(x,rad);
gplot(a,x);
```

It is a little easier to see it without the circles.



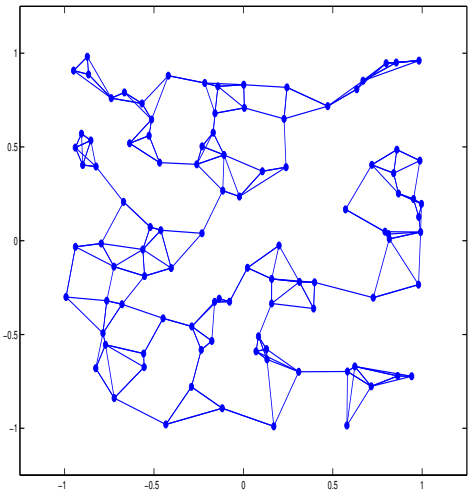
```
for i = 1:length(hc); delete(hc(i)); end
```



Our other source of geometric graphs will be artificial. We will often be interested in representing data by a graph. For example, imagine that each vertex corresponds to some information about a person, like height and weight. We could then connect those that are nearby, and try to see if health outcomes are related to the graph. For this purpose, I prefer using a k-nearest neighbor graph. Such a graph is drawn by connecting each vertex to its k nearest neighbors. In this case, we will choose $k = 3$.

```
clf
```

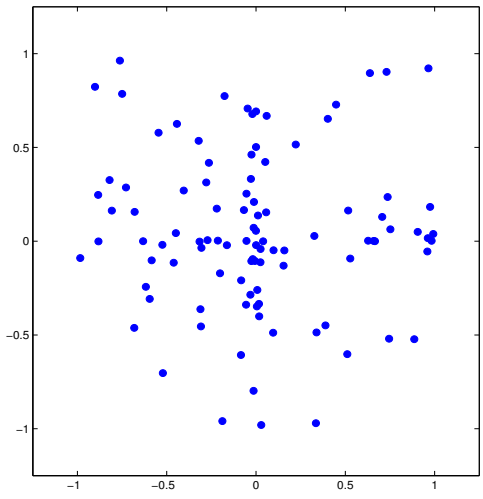
```
px = plot(x(:,1),x(:,2),'o','MarkerFaceColor','b');
hold on
a = knnGraph(x,3);
gplot(a,x);
axis(axis*(1+rad));
```



Even with $k = 3$, these graphs are usually connected. And, they typically have many fewer edges than similar connected graphs in which connectivity is determined by circles.

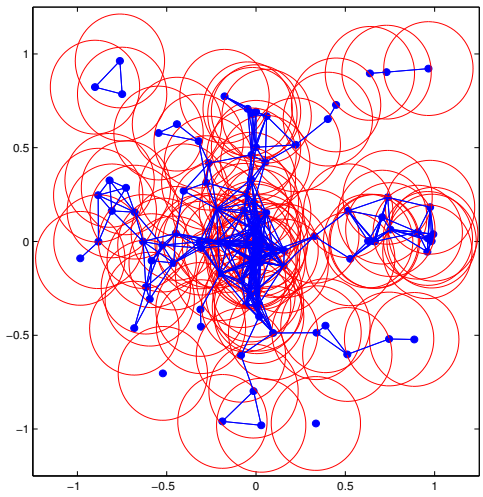
Connecting pairs of vertices below a given distance does not work as well when the typical distances between vertices differ in different parts of the data set. For an example, let's square every coordinate, which will push the vertices towards the axes.

```
xs = sign(x).*(x.^2);
px = plot(xs(:,1),xs(:,2),'o','MarkerFaceColor','b');
axis(axis*(1+rad))
```

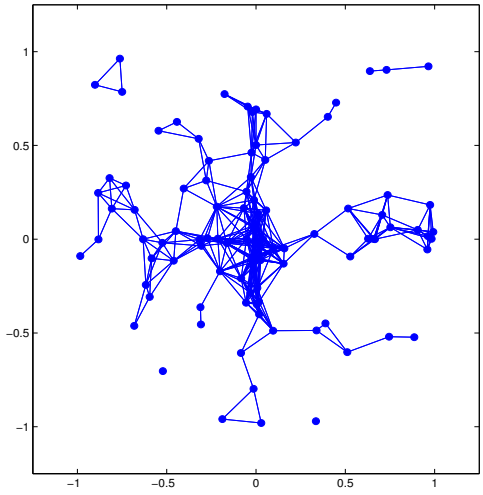


If we choose the same radius as before, we get high density near the axes, but almost no edges near the periphery.

```
hold on;
hc = drawCircles(xs,rad);
a = radGraph(xs,rad);
gplot(a,xs);
```

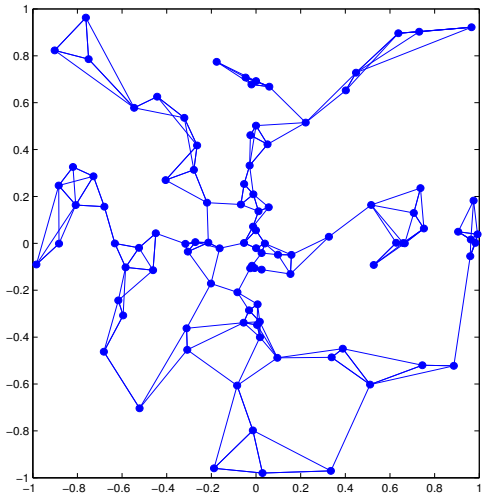



```
for i = 1:length(hc); delete(hc(i)); end
```



But, a 3-nearest neighbor graph still looks good.

```
a = knnGraph(xs,3);
clf; hold on;
px = plot(xs(:,1),xs(:,2),'o','MarkerFaceColor','b');
gplot(a,xs);
```



1.5.1 Graphs from data

We will derive a more interesting example from the MNIST [LCB] dataset of hand drawn digits. Here are some of them.

```
>> load mnist
>> imshow(bigimg)
```

Each image is a 28-by-28 matrix of numbers between 0 and 1. We can turn each into a 784-dimensional vector by concatenating the rows of each. The graph called `a3` in `mnist` was formed from the 3-nearest neighbor graph of these vectors. You can use my code `knnGraph` to compute these on your own. The graph is connected. The images have been labeled with numbers between 0 and 9. Let's look at how many edges stay inside the set of vertices corresponding to images labeled 1, and compare it to how many edges leave.

```
>> s = (labels == 1);

% the number of non-zeros in the submatrix of a3 indexed by rows and columns in s
>> nnz(a3(s,s))

ans =

       30898

% the number of non-zeros in the submatrix of a3 indexed by rows in s and columns not in s.
>> nnz(a3(s,~s))
```

```
ans =

   861
```

1.6 Topics we will cover

Algorithmic problems:

- Clustering
- Learning and Inference
- Ranking, measuring importance

Processes on graphs:

- Percolation, Infection, Cascades
- Random Walks and Diffusion
- Dynamically Changing Graphs

References

[BKM+00] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer networks*, 33(1):309–320, 2000.

[Con00] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

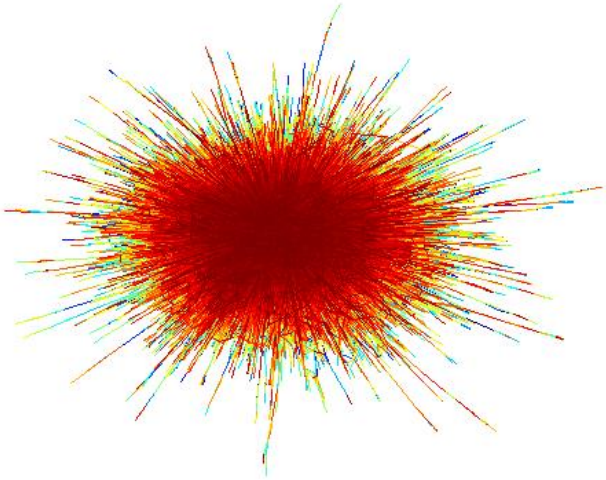
[Hu05] Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[LCB] Yann LeCun, Corinna Cortes, and Christopher Burges. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

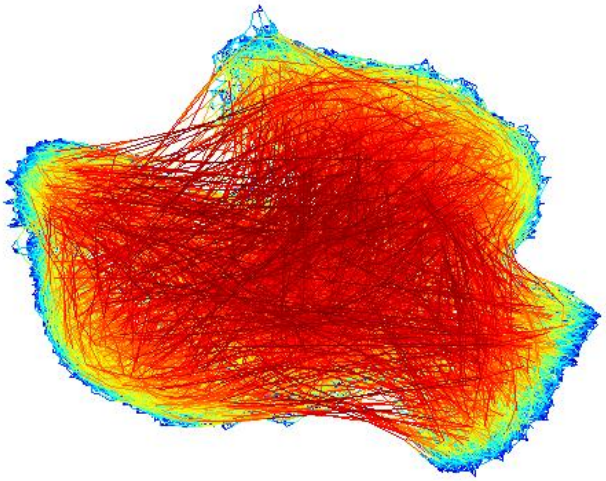
[Les] Jure Leskovec. Stanford large network dataset collection (SNAP).

[SBR+06] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. BioGRID: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl 1):D535–D539, 2006.

[SS09] Jimin Song and Mona Singh. How and when should interactome-derived clusters be used to predict functional modules and protein function? *Bioinformatics*, 25(23):3143–3150, 2009.



A drawing of the graph `dblp`



A drawing of the graph `mnist`