Just how dense are dense graphs in the real world? A methodological note
Guy Melançon

To cite this version:

HAL Id: lirmm-00091354
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00091354
Submitted on 6 Sep 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ABSTRACT
This methodological note focuses on the edge density of real world examples of networks. The edge density is a parameter of interest typically when putting up user studies in an effort to prove the robustness or superiority of a novel graph visualization technique. We survey many real world examples all being of equal interest in Information Visualization, and draw a list of conclusions on how to tune edge density when randomly generating graphs in order to build artificial though realistic examples.

Categories and Subject Descriptors
H.5.2 [Information Interfaces]: User Interfaces—Evaluation/Methodology, Benchmarking; G.2.2 [Discrete Mathematics]: Graph Theory—Graph Algorithms; G. [Probability and Statistics]: Random generation

Keywords
Interface Evaluation, Information Visualization, Graph Models, Edge Density, Random Generation, Real World Examples

1. INTRODUCTION
Graphs provide a useful mathematical tool for modeling various real world phenomenon. People participating in a same social activity, companies competing or collaborating in a given industrial sector, routers exchanging packets over the internet, or proteins involved in a given process of the living cell are examples of “networks” that can be modeled using graphs. They form a network because of the interactions taking place between the different actors: people, companies, routers or proteins.

In this paper, we will use the term “network” to denote the real world entity that usually maps to a graph after it is modeled. We will reserve the word “graph” to denote the mathematical construction itself. Actors of a network are usually mapped to nodes of the graph. Put differently, nodes are placeholders for actors to which one can attach various attributes (labels, ordinal or numerical values, etc.) in order to reflect properties of the modeled network. Links between actors of the network are mapped to edges, formally defined as pairs of nodes (or ordered couples if direction is relevant). Again, attributes obtained from a description of the studied network can be attached to edges of the graph.

The intent of this methodological note is to look at a particular measure of a graph that is being used when constructing artificial examples of real world networks, namely the edge density. We do not argue about what definition should be adopted, but insist on the way the edge density should be used – whatever the definition may be – when generating artificial example graphs in the context of Information Visualization. The construction of artificial example graphs often follow “random” generation processes. (That is, the randomness of these constructions more often comes from the fact that the underlying generation algorithm is unpredictable. This is radically different from being able to prove formally that the algorithm generates all graphs of the considered class with equal probability.) The class from which graphs are drawn can be controlled by varying parameters such as the number of edges, for instance.

Being able to build artificial examples of networks can be necessary. Algorithms are often designed to be able to deal with any graph possessing a number of given properties. Another way of looking at the question is the following. The list of desired properties defines a class of graphs and the hope is that the algorithm will perform as expected with any candidate graph of the class under consideration. A good example could be that of a drawing algorithm (for planar graphs or directed acyclic graphs, for instance). Testing it against a suite of example graphs may provide confidence on the algorithm’s robustness or performance. In doing so, the designers should take care in testing the algorithm with a selection of cases sampling the set of graphs it has been designed for. One good strategy here is to use a generation algorithm that will draw graphs uniformly at random (that is all graphs of the considered class should be drawn with equal probability).

Another important situation where artificial networks must be constructed is when putting up usability experiments. In order to avoid biases and to make sure that users’ performances are judged equally, one will often prefer an artificial dataset to avoid domain specific knowledge from interfering with the experiment. Testing a new graph navigation technique is a typical example where an artificial example will be required to comply with real world characteristics. For in-
stance, because layout algorithms typically exploit the link structure, it is essential that the constructed graphs have a topology that mimics that of the real world networks for which the navigation technique has been designed. Finally, because real world networks sometimes can be huge, it might be necessary to generate examples on a smaller scale but that nevertheless capture real world properties.

The concept of a random graph, as defined by Erdös and Rényi [13, 14] (see also [4]), has recently been challenged by more focused classes of graphs found in the real world. Indeed, the relevance of “small world networks” and “scale-free networks” has been assessed by the work of Barabási, Watts, Strogatz and others (see [31, 3, 32, 26] for instance; see also [5, 11]). It is nevertheless custom to simply say that “graphs have been generated randomly” when describing the process by which the example graphs have been obtained. What is kept silent here, is that restrictions on the class of all possible graphs have helped the designer to focus on properties of “real-world” examples. One such restriction consists in putting an upper bound on the number of edges of a graph. A useful way to pilot this parameter is to impose the constructed graphs to have a given edge density.

However, the edge density depends on the number of nodes of a graph and must be used with caution to avoid constructing non-realistic examples. We aim at exploring the notion of edge density and provide methodological guidelines to help designers when faced with the problem of “randomly generating example graphs”.

2. EDGE DENSITY OF GRAPHS

2.1 Possible definitions

The density of a graph should be a real number reflecting just how many edges it contains. Let us denote a graph as \( G = (V, E) \) where \( V \) is the set of nodes and \( E \) is the set of edges. We shall denote the number of nodes and edges as \( n = |V| \) and \( m = |E| \). We shall moreover only consider simple graphs, that is undirected graphs without any loop connecting a node to itself. Theoretically speaking, a graph is considered dense if its number of edges is close to \( n^2 \). We shall however adopt a radically different point of view, since graphs with that many edges certainly are virtually absent from reality as far as Information Visualization is concerned. Indeed, simple graphs have \( n(n-1)/2 \) edges on average, only those having a number of edges proportional to \( n \) are taken as test candidates for most layout algorithms. (This statement holds when one considers graphs on \( n \) nodes where each edge is selected with probability 1/2.) More generally, if the probability of selecting an edge is \( p \), then this number is \( p \cdot n(n-1)/2 \). See [4] or [5, Chap. 1].

This assumption is present in the Graph Drawing community and holds for most drawing algorithms producing node-link diagrams. As a particular case, planar graphs can have at most \( 3n - 6 \) edges (see [23, Section 1.5] for instance). The assumption is however irrelevant once we consider other types of representations such as matrix representations that can easily hold graphs having a quadratic number of edges.

Ghoniem et al. [20] argue that, from an information visualization point of view, a better choice is to compute the ratio:

\[
    d = \sqrt{m/n^2}. \tag{2}
\]

Their main argument against \( d_1 \) is:

- although \( d_1 \) is topologically meaningful, it is not scale invariant since the number of potential edges increases in the square of the number of nodes [20, Sect. 3.2] (see also [19, 21]).

Indeed, the density value \( d_1 \) for simple graphs, as defined by Eq. (1), varies over the interval \( [0, \frac{1}{2n}] \) and thus depends on \( n \). The density \( d_2 \) mapped to \([0, n-1]\) for directed graphs or \([0, n] \) if graphs contain loops. This is somehow undesirable as we would prefer the density value to vary on a fixed interval, whatever the number of nodes in the graph is. Eq. (2) achieves this by mapping the edge density of a graph to the interval \([0, \frac{1}{2n}]\) when considering simple graphs. Directed graphs are mapped to \([0, 1]\) or to \([0, 1] \) if loops are allowed. This last remark is of importance when it comes to deciding how the possible \( d \) values should be interpolated.

What we want to emphasize here is that even though Eq. (2) maps the edge density to a fixed interval \([0, \frac{1}{2n}]\) whatever the number \( n \) of nodes in the graph, claiming its scale invariance is somewhat wrong. In other words, the density cannot be used as a tuning parameter independently from \( n \). Any of these two definitions can be used depending on their utility or convenience. We shall refer to \( d_1 \) as the “linear density” and to \( d_2 \) as the “square root density”.

2.2 Looking closer at density

Let us now look at how the number of edges \( m \) in a graph varies according to its density. This is actually a function depending on two parameters, \( d \) and \( n \). The curves in Figure 1 show how the number of edges grow as \( d \) varies on \([0, \frac{1}{2n}]\).

Hence, for instance, a graph on 200 nodes having a density of \( d_1 = 0.1 \) has 400 edges, whereas for the same density a graph with 500 has 2500 edges. When the density goes to \( d_2 = 0.2 \), the number of edges respectively increases to 1600 (for a 200 nodes graph), and 10000 edges (for a 500 nodes graph). Observe that in each case, the number of edges has been multiplied by 4 which was predictable since the number of edges is given by the identity \( m = d^2 \cdot n^2 \) and that \( d_2 = 2 \cdot d_1 \).

However, although in each case the number of edges has increased identically by a factor of 4, the number of edges when compared to the number of nodes has been affected in a radically different manner. Indeed, for \( d = 0.1 \) a graph of \( n = 200 \) nodes has \( 2n = 400 \) edges. When \( n = 500 \), the number of edges already is equal to 5 times the number of nodes of the graph. When \( d \) increases to \( d = 0.2 \), the number of edges of a 200 node graph goes to 1600, which
is 8 times its number of nodes. A graph of 500 nodes and density \( d = 0.2 \) should have 20 times more edges than nodes which is rarely seen in the real world (see section 3). Even if realistic, such a ratio is rather high and it might be unreasonable to use a node-link layout to visualize such a graph. Put differently, it might be pointless to compare the performance of a navigation technique (or that of a user using the technique) against a node-link layout algorithm on a graph having 20 \( \cdot n \) edges, that is, with \( d_\ell = 20 \). The node-link layout is almost sure to lose (but this, of course, depends on the type of graph being visualized and/or navigated). This statement actually already is true for lower values of \( d_\ell \).

Figure 2 shows snapshots of random simple graphs with fixed density and increasing size (number of nodes). Each drawing has been rescaled (for each value of \( n \)) so that the size of nodes compare. As a consequence, the densities \( d_\ell \) and \( d \) also relates to visual density or opacity inside the circular drawing of the graphs. By contrast, Figure 3 shows graphs with increasing size all having the same density. Images have been scaled so that nodes of all graphs have the same size. As a consequence, any part inside the circular drawing of these graph relatively have the same “visual” density.

We ought to look at how the \( d_\ell \) density is affected by the increase in \( d \). In other words, we wish to look at the number of edges of a graph \( G \) as a multiple of \( n \), that is \( m = k \cdot n \) (or \( d_\ell(G) = k \)) as \( d \) varies. Our argument is simple. In order to test an algorithm or a navigation technique’s robustness with respect to the size and number of edges of a graph, we should consider letting \( m \) grow as \( m_0 = n, m_1 = 2n, m_2 = 3n, \) and so on up to a given order of magnitude. A majority of the examples we collected confirm that indeed networks have a linear number of edges with \( d_\ell \ll n \). Again, it only makes sense to test most layout algorithms on graphs with \( k \cdot n \) edges with rather low values of \( k \) (see section 3). The relation between \( d \) and \( d_\ell \) is straightforward:

\[
d_\ell = d^2 \cdot n \tag{3}
\]

From Eq. (3), we see that in order to map to graphs with \( n, 2n, 3n, \ldots, kn, \ldots \) edges, the “square root” density \( d \) should vary over the set

\[
\left\{ \sqrt{1 \over n}, \sqrt{2 \over n}, \sqrt{3 \over n}, \ldots, \sqrt{k \over n}, \ldots \right\}, \tag{4}
\]

showing that the relevant values for \( d \) actually depend on \( n \). That is, as \( n \) grows, \( d \) should be constrained to lower values in order to map to realistic graphs, as confirmed by Figure 4. Indeed, a graph with 30\( n \) edges has density \( d \sim 0.25 \) when \( n = 500 \) whereas the density goes up to \( d \sim 0.39 \) for a graph with \( n = 200 \) nodes.

### 3. REAL WORLD EXAMPLES

We list here example datasets we have collected from various places and covering as many application domains as possible. We plan to continue this effort and publicize the
Figure 3: The picture gives an impression of how density $d_r$ remains visually “constant” as $n$ grows. The example graphs were built using $d_r = 3$ and $n = 50, 100, 150, 200$.

available statistics on the web\(^1\). Note that not all networks are freely available. Whenever possible, we provide information to locate and get the example networks. In each case, we give the square root density $d$ and the linear density $d_r$ of the graph underlying the considered example network.

### 3.1 Small Worlds

Many networks found in the real world share the so-called “small world” property. The expression “small world” refers to the fact that networks often organize into communities, themselves being small worlds organized into sub-communities, and so on. This organization into communities rely on close relationships of people belonging to a same subgroup. In other words, people having common acquaintances are likely to already know each other. Or put more formally, nodes having common neighbors are likely to already be connected by a link. The interested reader will want to browse Watts’ novel [33] or read more specialized literature [32, 27, 5, 11, 29].

We also list here graphs that do not exactly fall within this category, but that more exactly are “scale-free”. That is they are organized around a few nodes with a very high degree. Although it is possible to find scale-free graphs that are also small worlds, some graphs are scale-free but not small world. That is, scale-free graphs can be community-less. This is the case for the yeast protein interaction networks, for example. The interested reader will want to look at [5, 11].

\(^1\)Anyone wishing to share data should feel free to contact us. Note that, as far as edge density is concerned, we do not require the actual dataset but only need to know about the number of nodes and edges.

<table>
<thead>
<tr>
<th>Dataset (source)</th>
<th>$n$</th>
<th>$m$</th>
<th>$d_r$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast protein interaction [30]</td>
<td>985</td>
<td>899</td>
<td>0.91</td>
<td>0.0304</td>
</tr>
<tr>
<td>IV London 2004 Co-authorship graph [8]</td>
<td>1160</td>
<td>2013</td>
<td>1.74</td>
<td>0.0387</td>
</tr>
<tr>
<td>Internet routers [7]</td>
<td>11174</td>
<td>23409</td>
<td>2.09</td>
<td>0.0337</td>
</tr>
<tr>
<td>Word association (web forms) [10]</td>
<td>2975</td>
<td>6981</td>
<td>2.35</td>
<td>0.0281</td>
</tr>
<tr>
<td>Internet routers [7]</td>
<td>10900</td>
<td>31180</td>
<td>2.86</td>
<td>0.0162</td>
</tr>
<tr>
<td>InfoVis Contest 2004 co-citation graph [9]</td>
<td>743</td>
<td>2219</td>
<td>2.99</td>
<td>0.0634</td>
</tr>
<tr>
<td>Yeast protein interaction [35]</td>
<td>3833</td>
<td>11942</td>
<td>3.16</td>
<td>0.0285</td>
</tr>
<tr>
<td>Painters dataset [34]</td>
<td>802</td>
<td>2486</td>
<td>4.95</td>
<td>0.0999</td>
</tr>
<tr>
<td>Co-authorship graph [6]</td>
<td>1506</td>
<td>7766</td>
<td>5.16</td>
<td>0.0585</td>
</tr>
<tr>
<td>Email network [12]</td>
<td>59912</td>
<td>147543</td>
<td>7.47</td>
<td>0.0112</td>
</tr>
<tr>
<td>Word association (nouns in dictionary) [17]</td>
<td>51511</td>
<td>392142</td>
<td>7.61</td>
<td>0.0122</td>
</tr>
<tr>
<td>Peer-to-peer (Kazaa) file sharing [22]</td>
<td>3403</td>
<td>30555</td>
<td>8.98</td>
<td>0.0514</td>
</tr>
<tr>
<td>InfoVis Contest 2004 co-authorship graph [9]</td>
<td>1953</td>
<td>17970</td>
<td>9.20</td>
<td>0.0686</td>
</tr>
<tr>
<td>Word association (verbs in dictionary) [16]</td>
<td>9043</td>
<td>101603</td>
<td>11.24</td>
<td>0.0352</td>
</tr>
<tr>
<td>Social network (IMDB) [2]</td>
<td>419</td>
<td>5651</td>
<td>13.49</td>
<td>0.1794</td>
</tr>
<tr>
<td>Passenger air traffic 2000 [1]</td>
<td>1148</td>
<td>16523</td>
<td>14.39</td>
<td>0.112</td>
</tr>
<tr>
<td>Web pages [18]</td>
<td>329</td>
<td>15120</td>
<td>25.67</td>
<td>0.2088</td>
</tr>
<tr>
<td>Constraint programming [18]</td>
<td>240</td>
<td>6300</td>
<td>26.25</td>
<td>0.33</td>
</tr>
<tr>
<td>Peer-to-peer (web) file sharing [22]</td>
<td>6049</td>
<td>1866271</td>
<td>308.53</td>
<td>0.2258</td>
</tr>
</tbody>
</table>

The table lists the values $n$, $m$, $d$, and $d_r$ for graphs coming from different application domains. All graphs have a single connected component.

- The data ($n$ and $m$) related to yeast protein interaction was extracted from the cited papers. The number of nodes (proteins) and edges (interactions) strongly
conducted their experiment using graphs with density val-
resentations and other node-link representations should con-
tively small, using a density of \( d \). However, they often
edge between two words if the first appeared in the definition of the other. The other network was built from verbs.

One dataset was used for the InfoVis Contest in 2004. All data is available from the InfoVis Contest Repository site [15]. The graphs considered here were borrowed from [9] (see also www.ca.ubc.ca/~tmn/papers/contest04/entry.html).

The peer-to-peer file sharing data concerns the largest connected component of the graph describing the exchange of a single file between peers.

The graphs are sorted according to their \( d_e \) values in order to make it clear that:

- Most graphs have a rather low linear density value below \( d_e \leq 10 \).
- Most graphs have a \( d \) value below 10\% with an average value of \( 6\% \) for the considered examples.
- The \( d \) density does not necessarily increase with \( d_e \); it all depends on the value of \( n \).
- the example with \( d_e = 308.53 \) stands as an exception.

Ghoniem uses a dataset coming from constraint program-
ing which probably explains why the authors of [20, 21] conducted their experiment using graphs with density values \( d = 0.20.4 \) and 0.6. Note that a value of \( d = 0.6 \) is rather high knowing that \( d \) actually varies over \([0, \frac{1}{n}]\) (and not over [0, 1] as the authors assumed). Also, interpolating at equidistant values over the interval is question-

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
World region & (crawl year) & Service & \( n \) & \( m \) & \( d_e \) & \( d \) \\
\hline
India (2004) & & & 13825999 & 16971769 & 12.23 & 0.00197 \\
United Kingdom (2002) & & & 18820486 & 298113762 & 16.10 & 0.00082 \\
Europe (2005) & & & 862664 & 19225140 & 24.29 & 0.00197 \\
United Kingdom (2005) & & & 39459925 & 93634282 & 23.73 & 0.00077 \\
Indonesia (2004) & & & 7414866 & 194110311 & 26.18 & 0.00197 \\
Italy (2004) & & & 46291594 & 1154725436 & 27.58 & 0.00082 \\
Slovakia (2005) & & & 50636154 & 194912601 & 38.50 & 0.00087 \\
\hline
\end{tabular}
\caption{Networks crawled in different countries and years.}
\end{table}

4. CONCLUSIVE REMARKS

4.1 So what density should I use when building my example graphs?

What we have emphasized here is that when looking at examples from the real world, it seems that the linear density \( d_e \) is a much better descriptor of the complexity of networks with respect to the application domain. That is, \( d_e \) seems relatively constant through all examples of a given application domain when compared to the \([0, \frac{1}{n}]\) theoretical range. For example, graphs extracted from the internet – hyperlinks between web pages or physical connections between internet routers – most of the times have a linear density that can exceed 10 to easily reach 20. Other examples – social networks, co-authorship, graphs from linguistics – all fall below the threshold \( d_e \leq 5 \). It also seems that many examples remain under \( d_e \leq 5 \). Note however that different types of graphs, with varying density, can be considered within a same application domain. Looking at classes of a Java library for instance, graphs with different density can be obtained by looking either at the inheritance structure or call graphs\(^2\). In other words, different questions arising

\(^2\)We wish to thank one of the anonymous referees for pointing out this relevant observation and providing the example. The case of the painters dataset [34] required the design of a specific force model based on the work of Noack [28]. Boutin et al. [6] specifically designed a strategy to filter edges out of a graph in order to help the identification of clusters.

The same type of error was observed in [24]. In order to assess the usability and superiority of their navigation technique, the authors conducted a usability study using an artificial email network. To build their network example, the authors first set its size to \( n = 200 \) and density at \( d = 30\% \). (That is, \( m = 3600 \).

It is rather high considering that it corresponds to a linear density of \( d_e = 18 \), which does not compare to the example reported in [12]. A number of edges equal to \( m = 1500 \) (or \( d = 13.72\% \)) would have been more accurate.

3.2 Web crawls

The following networks have been obtained by web crawlers, in an attempt to get a better view of the traffic taking place over several regions of the world. As expected the networks have a huge size. The incredibly large size of these examples undoubtedly show that \( d \) cannot be used without any reference to \( n \). All data is publicly available from the Laboratory for Web Algorithmics website (see www.lsi.dsi.unimi.it).
in a same application domain might correspond to different types of graphs. This is not really surprising as task and data types indeed are related. We will nevertheless talk about an “application domain” meaning a type of graph associated with a set of relevant tasks.

In other words, when considering a given application domain, you may well select a density \( d \) and work with that density independently from \( n \) – that is, you could use the same \( d \) for all example graphs you generate, on all cardinalities. On the contrary, when using the density measure \( d \), you should select \( d \) depending on the size of the example you wish to generate as indicated by Eq. (4).

However, it may be more convenient to use the density \( d \) to tune the algorithms used to generate example graphs because it varies over a fixed interval. The value to be used should then be chosen according to a target application domain, and should be computed from the adequate \( d \) value. More precisely, suppose one wants to randomly generate a graph to test a technique developed for the navigation of internet graphs while using artificial example graphs of a size smaller than the real world web crawls. Let \( d \) and \( d_t \) be the densities computed from a real world example graph with given \( n \) and \( m \) \( (d = \sqrt{m/n^2}, \ dt = m/n) \). Let \( q \) be such that the size of the example to be generated will be \( N = n/q \). Because we want the linear density to remain constant, \( d^{(N)} = d_t \), we must have \( M = m/q \), so we find \( d^{(N)} = d \cdot \sqrt{q} \). In other words, as we generate examples of size smaller than real world examples we can allow the density \( d \) to increase only by a square root factor.

Figure 5 illustrates how the density \( d \) varies as \( q \) increases when mimicking the 2004 web crawl on India. When generating a graph with \( N = 1000 \) nodes, that is if \( q \approx 10^7 \), we can use a density \( d \sim 0.09 \) close to 30 times that of the original real world example. Starting from Ebel’s email network with \( n = 59912 \) to build a \( N = 200 \) nodes example graph, we compute the above transformation with \( q \sim 300 \) so we can use a density close to \( \sqrt{300} \cdot 0.0112 = 0.194 \), leading to a graph with about 1500 edges.

4.2 Models of random graphs

We have made a point explaining how the density \( d \) or \( d_t \) should be used in order to construct “realistic”, though artificial, example graphs. Properly tuning the density is however often not enough. Different algorithms will typically produce different types of graph because they model different classes of networks. Properly choosing a model has a definite impact on the possible uses of the artificial examples and thus on the conclusion one can draw from the experiment that was conducted using these artificial graphs, or on the robustness of a drawing algorithm.

For example, it is certainly wrong to simply draw graphs at random using the Erdos-Rényi model when designing a user experiment trying to assess of the usability of a visual navigation technique. Indeed, Erdos-Rényi graphs have a more or less constant node degree simply because edges are drawn between nodes without any specific pattern. However, most examples found in the real world are much more inhomogeneous and show properties that radically differ from the Erdos-Rényi graphs. Barabási and others extensively study small world networks or scale-free networks through their degree distribution. Scale-free graphs, for example, typically show power law degree distribution. Barabási suggested that these networks arise from a pattern he calls “preferential attachments”. That is, new actors will preferably get connected to high degree nodes, thus leading to a “rich get richer” pattern. The book by Bornholdt and Schuster [5] surveys Barabási preferential attachment model (with an interesting chapter by Bollabás) and many other useful and “realistic” model that should be preferred to Erdos-Rényi when building test cases. The interested reader will want to look at the recent books [5] and [11]. A comparison of all available models and algorithms for generating random graphs is out of the scope of this short note and merits to be addressed separately. Again, our ambition here was to focus our discussion on the edge density of graphs and on how it should be used when building “realistic” examples.

5. REFERENCES


