Analysis of Complex Systems

Lecture 2: Complex systems as graphs

Marcus Kaiser
m.kaiser@ncl.ac.uk
Objectives

- Graphs
- Computational complexity (Time and Space)
- Graph representations
- Patterns in graphs
- (Shortest) Paths
Origin of graph theory: Leonhard Euler, 1736

Bridges over the river Pregel in Königsberg (now Kaliningrad)
Euler tour: path that visits each edge and returns to the origin
Graphs

- Graph: set of nodes and edges (non-directed)
  \[ G = (V,E) \]
- Set of nodes: \( V \) (singular: vertex; plural: vertices)
- Set of edges: \( E \subseteq V \times V \)
- E.g., \( V=\{v1,v2,v3,v4\} \),
  \( E=\{(v1,v2), (v1,v3), (v2,v3), (v3,v4)\} \)
Directed graphs (Digraphs)

- Graph: set of nodes and *arcs* (directed)
- Set of nodes (vertices): $V$
- Set of edges: $E \subseteq V \times V$, the order matters
- E.g., $V=\{v1,v2,v3,v4\}$,
  $$E=\{(v1,v2), (v1,v3), (v2,v3), (v3,v4), (v4,v1)\}$$
Graphs and Networks

In theory (mathematics)
Graph: $G = (V, E)$

Network: $N = (G, s, t, c)$
defined by graph $G$ with
source $s$, sink $t$, and
edge capacity $c$
(examples: electricity/power
grid, water flow, metabolic flux)

In reality (CS, engineering, economics, life and social sciences): term network used throughout (as in this course)
Nodes in graphs

- Isolated nodes
- Degree of a node
- Connected graph
- Average degree of a graph
- Edge density: probability that any two nodes are connected
  \[ d = \frac{E}{\left(\frac{N(N-1)}{2}\right)} \]

- Isolated node: \( v_5 \)
- Degree of a node: \( d(v_1)=2, d(v_4)=1 \)
- Average degree of a graph:
  \[ D = \frac{2+2+3+1+0}{5} = 1.6 \]
- Edge density
  \[ d = \frac{4}{\left(\frac{5 \times 4}{2}\right)} = 0.4 \]
## Examples: edge density

<table>
<thead>
<tr>
<th></th>
<th>nodes</th>
<th>edges</th>
<th>density [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autobahnen</td>
<td>1168</td>
<td>2486</td>
<td>0.18</td>
</tr>
<tr>
<td>Internet</td>
<td>6524</td>
<td>29629</td>
<td>0.0696</td>
</tr>
<tr>
<td>www</td>
<td>325729</td>
<td>1497135</td>
<td>0.0014</td>
</tr>
<tr>
<td>Power Grid</td>
<td>4677</td>
<td>12500</td>
<td>0.0572</td>
</tr>
<tr>
<td>metabolic</td>
<td>422</td>
<td>1972</td>
<td>1.3</td>
</tr>
<tr>
<td>C. Elegans</td>
<td>202</td>
<td>2540</td>
<td>6.3</td>
</tr>
<tr>
<td>macaque</td>
<td>73</td>
<td>835</td>
<td>16</td>
</tr>
</tbody>
</table>

**sparse network** (density ~ 1%)

**dense network** (density > 5%)
Algorithm evaluation: Time and space
Two resources constrain our analysis and simulations: processing *time* and memory *space*.

How much time and space will an algorithm need to deal with a network with N nodes? Are there better algorithms that are faster or use less memory?

Big-O notation gives a *worst-case* approximation of needed resources. Examples:

<table>
<thead>
<tr>
<th>Resource needed</th>
<th>Big-O notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>c*N</td>
<td>O(N) (constant factors are neglected)</td>
</tr>
<tr>
<td>N^2+N+c</td>
<td>O(N^2) (only largest component of a sum is used)</td>
</tr>
</tbody>
</table>
Examples for the time resource

(P – polynomial)

The good:
- time to get an item from a table: \( O(1) \)
- time needed for adding \( N \) numbers: \( O(N) \)

The bad:
- calculate the degree of all nodes: \( O(N^2) \)
- find all shortest paths: \( O(N^3) \)

(NP – non-polynomial)

The ugly:
- test whether two networks are identical: \( O(N!) \)
- travelling salesman problem: \( O(N^N) \)
Graph representation
Representation of graphs – 1

- Adjacency matrix: \( a(i,j) = 1 \) if there is an edge between nodes \( i \) and \( j \), \( a(i,j)=0 \) otherwise
- In case of non-directed graphs the adjacency matrix is symmetric (same values across the diagonal)
- In case of directed graphs the adjacency matrix may not be symmetric

\[
A = \begin{pmatrix}
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{pmatrix}
\]

**Advantage:** Direct access to each element (O(1) time complexity)

**Disadvantage:** Storage needs for large networks (space complexity \( O(N^2) \) even if most entries are zeros)
Representation of graphs – 2

- Adjacency list (list of existing edges)
- \( E = \{(v_1,v_2,w_{12}), (v_1,v_3,w_{13}), (v_2,v_3,w_{23}), \ldots \} \)

\( w \): weight (e.g. 1)

Can be generated in Matlab with the `sparse` command

Advantage: Low demand for computer memory
\hspace{1cm} (space complexity: \( O(E) \sim O(N) \))

Disadvantage: time to access one matrix element
\hspace{1cm} (time complexity \( O(N) \) instead of \( O(1) \) for adjacency matrices)
Patterns in graphs
Sub-graphs

- Sub-graph = subset of edges and nodes
- E.g., $E' \subseteq E$, $V' \subseteq V$
- Complementary graph = the rest of the graph
Neighbourhood

- Neighbourhood of a node = set of nodes connected to the node
- E.g., $N(v1)=\{v2,v3\}$
- E.g., close functional relationship
Clusters (or modules / communities)

- Dense connections within subgraph but few connections with remaining network
- Nodes within clusters often have similar functions (e.g. protein interaction cluster)
Example: Cat cortical connectivity

- **Visual**
  - DLS
  - VLS
  - ALLS
  - PMLS
  - PLLS
  - En
  - Sb
  - pSe
  - 36
  - 35
  - RS
  - CCA
  - CG
  - IA

- **Auditory**
  - 21a
  - 21b
  - 20a
  - 20b
  - 7
  - AES
  - PS
  - AI
  - AII
  - AAF
  - P
  - VP(ctx)
  - EPP
  - Tem
  - 3a
  - 3b
  - 1
  - 2
  - SII
  - SIV
  - 4g
  - 4
  - 6l
  - 5Al
  - 5Am
  - 5Bl
  - 5Bm
  - 5m

- **Somato-sensory-motor**
  - SSAI
  - SSAo

**Edge:** Existing fibre tract

**Colours:** Area function

**Position:** Nearby on the circle when connections are similar
Trees

- Tree = set of nodes and edges, such that no cycle is included
- Terminology: forest, parent, leaf, root
Bi-partite graphs

- Subclass of n-partite graphs

- Two classes:
  only edges between nodes of a different class are established

- Examples:
  Pairing (male-female; people-guitars)
  Interactions (DNA-Protein)
(Shortest) Paths
Paths in graphs – 1

- Path between nodes
- E.g., signalling pathway
- E.g., $V=\{v_1, v_2, v_3, v_4, v_5, v_6\}$, $E=\{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_4), (v_3, v_5), (v_5, v_4)\}$
  $P(v_1, v_5)=\{(v_1, v_3), (v_3, v_4), (v_4, v_5)\}$
Paths in graphs – 2

- Directed graphs
- E.g., $V=\{v_1,v_2,v_3,v_4,v_5,v_6\}$, $E=\{(v_1,v_2), (v_1,v_3), (v_2,v_3), (v_3,v_4), (v_3,v_5), (v_5,v_4)\}$
- $P(v_1,v_5)=\{(v_1,v_3),(v_3,v_5),(v_5,v_4)\}$
Paths in graphs – 3

- Cycles (non-directed) and circuits (directed)
- E.g., metabolic cycles
- E.g., C=(v1,v2,v3)
- Loop = direct feedback (v1->v1)
Number of paths between two nodes.

E.g., how redundant are protein interaction systems.

E.g., $P(v_1,v_5) = \{(v_1,v_3),(v_3,v_5)\}$, $\{(v_1,v_3),(v_3,v_4),(v_4,v_5)\}$, $\{(v_1,v_2),(v_2,v_3),(v_3,v_5)\}$.
**Paths in graphs – 5**

- Length of path: number of edges in the path
- E.g., $P(v_1,v_5)=\{(v_1,v_3),(v_3,v_5)\}$, length $(P)=2$
- Paths of length 1 $\rightarrow$ entries of the adjacency matrix $A$
- Paths of length 2 $\rightarrow$ entries of $A^2=A \times A$
- Paths of length $k$ $\rightarrow$ entries of $A^k=A \times A^{k-1}$
Matrix multiplication

- Element-wise multiplication (A.*B = C in Matlab)
  
  $\begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix} \ast \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 4 \\ 2 & 0 \end{pmatrix}$

- Matrix multiplication (A*B = C in Matlab)
  
  $A \times B$

  $c_{i,j} = a_{i,1}b_{1,j} + a_{i,2}b_{2,j} + \ldots + a_{i,n}b_{n,j}$

  $C_{i,j} = \sum_{r=1}^{n} A_{i,r}B_{r,j}$
Paths in graphs – 6

\[ A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \]

\[ A^3 = \begin{pmatrix} 2 & 3 & 4 & 1 \\ 3 & 2 & 4 & 1 \\ 4 & 4 & 2 & 3 \\ 1 & 1 & 3 & 0 \end{pmatrix} \]

\[ A^2 = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 3 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \]

\[ A^4 = \begin{pmatrix} 7 & 6 & 6 & 3 \\ 6 & 7 & 6 & 3 \\ 6 & 6 & 11 & 2 \\ 4 & 4 & 2 & 3 \end{pmatrix} \]
Paths in graphs – 7

- Distance between two nodes = length of minimal length path between the nodes
- \( D(v_1,v_4) = \min \{ \)
  
  \[
  \text{length}((v_1,v_3),(v_3,v_4)),
  \text{length}((v_1,v_2),(v_2,v_3),(v_3,v_4))
  \]

  \[
  \} = 2
  \]
Paths and graphs – 8

- Diameter of a graph = maximum of distances between nodes of the graph
- \( D(G) = \max \{d(v1,v2), d(v1,v3), \ldots, d(v3,v4)\} = 2 \)
Average distance of a graph (or average shortest path / characteristic path length)
= average of distances between nodes

Example:
\[ D(G) = \frac{d(v_1,v_2) + d(v_1,v_3) + \ldots + d(v_3,v_4)}{12} \]

Time complexity: \( O(N^3) \); some algorithms achieve \( O(N^2 \log N) \)
Examples: average path length

- Human acquaintance network: 7 steps (“six degrees of separation”) between any two persons in the US (Milgram, Psychology Today, 1963) -> small-world phenomenon

- World-Wide-Web: 19 steps from one web page to any other webpage (Albert et al., Nature, 1999)

- Neural networks:
Summary

- What are graphs and patterns in graphs?
- Which time and space resources are needed to analyse this size of the network? Are there better algorithms?
- How can networks be represented in the computer and what are the benefits and disadvantages?
- What are paths/shortest paths/diameter?
Q&A – 1

1. Is it true that graphs are made of nodes and edges?
2. Is it true that in a directed graph the edge \((v1,v2)\) is equivalent with the edge \((v2,v1)\)?
3. Is it true that it is possible to have more than one path between two nodes of a graph?
4. Is it true that the distance between two nodes is the length of the longest path between the two nodes?
5. Is it true that the matrix A is an adjacency matrix? What about the matrix B?

\[
A = \begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 2 & 0 & 1 \\
1 & 0 & 0 & 0
\end{pmatrix}
\quad
A = \begin{pmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0
\end{pmatrix}
\]
Q&A – 3

6. Is it true that the neighbourhood of a node is the set of nodes connected to the node?

7. What is the time and space complexity of the following tasks:
   - yield all neighbours of one node
   - yield the length of the longest path between one pair of nodes
   - yield the all alternative shortest paths between one pair of nodes