Mapping Small Worlds
Detecting the Structure of Social Networks

Matteo Dell’Amico
DISI, University of Genoa
dellamico@disi.unige.it

P2P2007
Galway, Ireland
4 September 2007
Outline

1 Small Worlds
   - What “Small World” Means
   - Models

2 From the Network to the Layout
   - Our Problem
   - The Freenet Algorithm
   - Koren: Spectral Graph Drawing

3 The Decentralized Algorithm
   - Adaptation
   - Experimental Results
   - Conclusions
Milgram’s Experiment (1967)

- A random person in USA gets a letter.
- Using only **direct acquaintances**, the letter is sent to a target on the other side of the country.
- “Six Degrees of Separation”: the letters needed on average six **steps** to reach their destination.
It’s a Small World

- **Clustering**: many of my friends know each other.
- Many **triangles** and higher order **cliques**.
- Nodes reachable in a **single step**.
- Nodes reachable in two steps.
- Even if there are many cliques, the number of nodes grows steadily.
Nodes reachable **in three steps**.

The number of nodes grows **exponentially**.
Definition

Small world

- If $n$ is the size of the network:
  - **Clustering** (ratio of triangles over connected triples) is independent of $n$.
  - The **lengths of geodesic paths** grows as the **logarithm** of $n$.

Navigability

- Nodes “know how” to **navigate** the network (e.g., efficiently find short paths).
The Watts - Strogatz Model

- Each node on the ring is connected to the $m$ closest neighbors.
- Every link to a neighbor is replaced, with probability $p$, by a random jump.
- Navigability is not explained yet.
Kleinberg’s Model

- Each node has \( r \) shortcuts.
- \( P(u \text{ gets a shortcut to } v) \propto d(u, v)^{-2} \).
- The position on the layout is the navigability criterion.
  - Routing is done by forwarding to the node which is closest to destination.
- Paths found are \( O\left(\log^2 n\right) \).
The layout represents **affinity** between nodes: closer nodes are more likely to be linked.

Geographical proximity is not the only factor:
- Similarity of tastes
- Habits (work, hobbies...)
- ...

Can we infer information on affinity from the network structure itself?

If we obtain good layouts, we will find **short paths** using them.

Social networks are **structured** networks... but we have to **discover** the structure!
Applications

- Routing in anonymous “friend-to-friend” networks (e.g., Freenet)
- Paths in webs of trust to infer trust for unknown users
- Network analysis:
  - Finding similarity of items (e.g., e-commerce)
  - Personalized ranking
  - Similarity between tags in folksonomies
Sandberg’s Algorithm for Freenet

Idea
- Nodes are **randomly** placed in a $n$-dimensional space.
- Goal: permute positions so that the **total length** of links is **minimized**.

Implementation
- **Metropolis - Hastings**: iterative algorithm.
- Random pairs of nodes are taken in consideration, and switched according to a probability depending on edge length.

Experimental result
- No benefit for $n > 1$. 

**Matteo Dell’Amico**

**Mapping Small Worlds**
Sandberg’s Algorithm for Freenet

Idea
- Nodes are randomly placed in a $n$-dimensional space.
- Goal: permute positions so that the total length of links is minimized.

Implementation
- Metropolis - Hastings: iterative algorithm.
- Random pairs of nodes are taken in consideration, and switched according to a probability depending on edge length.

Experimental result
- No benefit for $n > 1$. 
Sandberg’s Algorithm for Freenet

**Idea**
- Nodes are **randomly** placed in a $n$-dimensional space.
- Goal: permute positions so that the **total length** of links is **minimized**.

**Implementation**
- **Metropolis - Hastings**: iterative algorithm.
  - Random pairs of nodes are taken in consideration, and **switched** according to a probability depending on edge length.

**Experimental result**
- **No benefit** for $n > 1$. 
Graph Drawing

- Algorithms written to obtain **untangled** graphical representations of graphs.
- Some algorithms try to **minimize distances between connected nodes**.
- That’s good for us as well!
Differences

- We want decentralized and scalable algorithms.
- We can have a high number of dimensions.
Koren’s Algorithm

**Goal**
- For a vector of positions $x$, minimizing the energy $E(x)$:

$$E(x) = \sum_{\text{edges}(i,j)} w_{ij} (x(i) - x(j))^2$$

$$\text{Var}(x) = 1$$

**Solution**
- The eigenvector $u_2$ of the **laplacian matrix** $L = D - A$ associated to the smallest nonzero eigenvalue $\lambda_2$.
  - $D$ is the diagonal matrix such that $d_{ii}$ is $i$’s degree, $A$ is the adjacency matrix, $u_1 = 1^n$, $\lambda_1 = 0$.
- To draw in $d$ dimensions, $u_2, \ldots, u_{d+1}$ are used.
- **Degree normalization**: a further optimization to avoid high-degree nodes to be placed in the center.
Koren’s Algorithm (2)

Power iteration step

1. **Orthogonalization**: each \( u_j \) vector is made linearly independent from \( u_i \) vectors with \( i < j \).

\[
    u_i \leftarrow u_i - \frac{u_i^T D u_j}{u_j^T D u_j} u_j
\]

2. **Movement**: each node moves towards the “center of mass” of its neighbors.

\[
    \hat{u}_i(j) \leftarrow \frac{1}{2} \left( u_i(j) + \frac{\sum_{k \in N(j)} w_{jk} u_i(k)}{\deg(j)} \right)
\]

3. **Normalization**: the variance of each vector is set to 1.

\[
    u_i \leftarrow \frac{\hat{u}_i}{\|\hat{u}_i\|}
\]
Results from the OpenPGP web of trust (26350 nodes).
Orthogonalization

\[ u_i \leftarrow u_i - \frac{u_i^T D u_j}{u_j^T D u_j} u_j \forall j < i \]

- The computational cost is **quadratic** with respect to the number of dimensions.
- The power iterations **converge slowly**, since the small \( \mu_i \) eigenvalues are close.
- After a given number of iterations, \( u_i \) is a **linear combination** of the most important eigenvectors.
- We **skip** this step, making sure we stop calculating **before** the algorithm converges.
Movement

\[ \hat{u}_i(j) \leftarrow \frac{1}{2} \left( u_i(j) + \frac{\sum_{k \in N(j)} w_{jk} u_i(k)}{\text{deg}(j)} \right) \]

- This step is already decentralized: we only need each node to communicate with its neighbors.
Normalization

\[ u_i(j) \leftarrow \frac{\hat{u}_i}{\|\hat{u}_i\|} \]

- If the numeric precision is sufficient, we just need to execute the step at the end of the algorithm.
- Decentralization: knowing \( \|\hat{u}_i\| \) is equivalent to knowing the average of \( \hat{u}_i^2 \).
- The movement step converges to the mass center of the graph: we will perform it starting with the \( \hat{u}_i^2 \) values.
Results: Decentralized Algorithm
To make routing better, each node can communicate its neighbors, at the end of the computation, the position of all their own neighbors.

As next step in routing, we choose the node that takes us closest to destination in two steps.
Results: Neighbor-of-Neighbor
Conclusions

- **Clear improvement** over the state of the art.
- **Future work:** resistance to attack.