Structured-world conjecture: On modules and communities in real-world networks

Lovro Šubelj

University of Ljubljana
Faculty of Computer and Information Science
Slovenia

May 3, 2012
1 Motivation

2 Network structure
   - Degree mixing
   - Clustering mixing
   - Network structures
   - Structured-worlds

3 Structure detection
   - Label propagation
   - General propagation

4 Experimental analysis
   - Synthetic networks
   - Real-world networks
   - Software networks

5 Conclusions
Motivation

**Motivation**

Are there modules that could explain the structure of software networks?

class A extends S implements I {
    F field;
    public A (P parameter) {
        ...
    }
    public R function(P parameter) {
        ...
        return R;
    }
}
Outline

1 Motivation

2 Network structure
   - Degree mixing
   - Clustering mixing
   - Network structures
   - Structured-worlds

3 Structure detection
   - Label propagation
   - General propagation

4 Experimental analysis
   - Synthetic networks
   - Real-world networks
   - Software networks

5 Conclusions
Degree mixing

- Degree mixing coefficient \( r \in [-1, 1] \). (Newman [30])

\[
r = \frac{1}{2m\sigma_k} \sum_{ij} (k_i - k)(k_j - k),
\]

where \( \sigma_k \) is the standard deviation and \( k_i \) degree of node \( i \).

- Assortative mixing refers to \( r > 0 \), and disassortative to \( r < 0 \).

- \( r \) is simply a Pearson correlation coefficient of \( k_i \) at links’ ends.

1) \( s \)-metric [23]  
2) \( \Gamma \) connectivity [38]  
3) Correlation profiles [27]
# Degree mixing (II)

Social networks are assortative, while most other are disassortative!

<table>
<thead>
<tr>
<th>Type</th>
<th>Network</th>
<th>n</th>
<th>m</th>
<th>k</th>
<th>C</th>
<th>D</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration</td>
<td>netsci</td>
<td>1589</td>
<td>2742</td>
<td>3.5</td>
<td>0.638</td>
<td>0.690</td>
<td>0.462</td>
</tr>
<tr>
<td></td>
<td>condmat</td>
<td>27519</td>
<td>116181</td>
<td>8.4</td>
<td>0.655</td>
<td>0.722</td>
<td>0.166</td>
</tr>
<tr>
<td></td>
<td>comsci</td>
<td>239</td>
<td>568</td>
<td>4.8</td>
<td>0.479</td>
<td>0.561</td>
<td>-0.044</td>
</tr>
<tr>
<td>Online social</td>
<td>pgp</td>
<td>10680</td>
<td>24316</td>
<td>4.6</td>
<td>0.266</td>
<td>0.317</td>
<td>0.238</td>
</tr>
<tr>
<td></td>
<td>football</td>
<td>115</td>
<td>613</td>
<td>10.7</td>
<td>0.403</td>
<td>0.419</td>
<td>0.162</td>
</tr>
<tr>
<td></td>
<td>jazz</td>
<td>198</td>
<td>2742</td>
<td>27.7</td>
<td>0.617</td>
<td>0.703</td>
<td>0.020</td>
</tr>
<tr>
<td>Social</td>
<td>dolphins</td>
<td>62</td>
<td>159</td>
<td>5.1</td>
<td>0.259</td>
<td>0.319</td>
<td>-0.044</td>
</tr>
<tr>
<td></td>
<td>karate</td>
<td>34</td>
<td>78</td>
<td>4.6</td>
<td>0.571</td>
<td>0.666</td>
<td>-0.476</td>
</tr>
<tr>
<td></td>
<td>emails</td>
<td>1133</td>
<td>5451</td>
<td>9.6</td>
<td>0.220</td>
<td>0.253</td>
<td>0.078</td>
</tr>
<tr>
<td>Communication</td>
<td>enron</td>
<td>36692</td>
<td>183831</td>
<td>10.0</td>
<td>0.497</td>
<td>0.530</td>
<td>-0.111</td>
</tr>
<tr>
<td>Road network</td>
<td>euro</td>
<td>1039</td>
<td>1305</td>
<td>2.5</td>
<td>0.019</td>
<td>0.025</td>
<td>0.090</td>
</tr>
<tr>
<td>Power grid</td>
<td>power</td>
<td>4941</td>
<td>6594</td>
<td>2.7</td>
<td>0.080</td>
<td>0.100</td>
<td>0.003</td>
</tr>
<tr>
<td>Citation</td>
<td>hepart</td>
<td>27770</td>
<td>352285</td>
<td>25.4</td>
<td>0.312</td>
<td>0.353</td>
<td>-0.030</td>
</tr>
<tr>
<td>Documentation</td>
<td>javadoc</td>
<td>2089</td>
<td>7934</td>
<td>7.6</td>
<td>0.373</td>
<td>0.433</td>
<td>-0.070</td>
</tr>
<tr>
<td>Protein</td>
<td>yeast1</td>
<td>2445</td>
<td>6265</td>
<td>5.1</td>
<td>0.215</td>
<td>0.250</td>
<td>-0.101</td>
</tr>
<tr>
<td></td>
<td>yeast2</td>
<td>2114</td>
<td>2203</td>
<td>2.1</td>
<td>0.059</td>
<td>0.072</td>
<td>-0.162</td>
</tr>
<tr>
<td></td>
<td>javax</td>
<td>1595</td>
<td>5287</td>
<td>6.6</td>
<td>0.381</td>
<td>0.440</td>
<td>-0.120</td>
</tr>
<tr>
<td></td>
<td>jung</td>
<td>317</td>
<td>719</td>
<td>4.5</td>
<td>0.366</td>
<td>0.423</td>
<td>-0.190</td>
</tr>
<tr>
<td>Software</td>
<td>guava</td>
<td>174</td>
<td>355</td>
<td>4.1</td>
<td>0.320</td>
<td>0.375</td>
<td>-0.218</td>
</tr>
<tr>
<td></td>
<td>java</td>
<td>1516</td>
<td>10049</td>
<td>13.3</td>
<td>0.685</td>
<td>0.731</td>
<td>-0.283</td>
</tr>
<tr>
<td>Web graph</td>
<td>blogs</td>
<td>1490</td>
<td>16715</td>
<td>22.4</td>
<td>0.263</td>
<td>0.293</td>
<td>-0.221</td>
</tr>
<tr>
<td>Metabolic</td>
<td>elegans</td>
<td>453</td>
<td>2025</td>
<td>8.9</td>
<td>0.646</td>
<td>0.710</td>
<td>-0.226</td>
</tr>
<tr>
<td>Internet</td>
<td>oregon</td>
<td>767</td>
<td>1734</td>
<td>4.5</td>
<td>0.293</td>
<td>0.317</td>
<td>-0.299</td>
</tr>
<tr>
<td>Bipartite</td>
<td>women</td>
<td>32</td>
<td>89</td>
<td>5.6</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.337</td>
</tr>
</tbody>
</table>
Network clustering coefficient $C = \frac{1}{n} \sum_i c_i$. (Watts and Strogatz [56])

$$c_i = \frac{t_i}{\binom{k_i}{2}},$$

where $t_i$ is number of links among $\Gamma_i$, $c_i \in [0, 1]$.  

For many real-world networks $c_i \sim 1/k_i$. [41, 42, 48]

4) Degree assortative

5) Degree disassortative

High degree nodes never have high $c_i$!
DEGREE-CORRECTED CLUSTERING

- Network degree-corrected clustering co. \( D = \frac{1}{n} \sum_i d_i. \) (Soffer and Vázquez [46])
  \[
d_i = \frac{t_i}{\omega_i},
\]
  where \( \omega_i \) is the max. number of links with respect to \( \{k_i\} \), \( d_i \in [0, 1] \).
- Since \( \omega_i \leq \left(\frac{k_i}{2}\right) \), \( d_i \geq c_i \) and \( D \geq C \) by definition.

6) Degree assortative 7) Degree disassortative

- For pseudo-fractal model \( c_i \sim 1/k_i \) implies \( c_i \sim 1/\log k_i \). [46]
**Degree-corrected clustering (II)**

- Most nodes in assortative networks share similar $d_i \gg 0$, whereas 30-55% of nodes in disassortative networks have $d_i \approx 0$!

8) Degree assortative  
9) Degree disassortative

- $d_i$ appear to capture certain characteristics of the underlying domain.
Define clustering mixing coefficients $r_c, r_d \in [-1, 1]$. (Šubelj and Bajec [54])

$$r_d = \frac{1}{2m\sigma_d} \sum_{ij} (d_i - D)(d_j - D),$$

where $\sigma_d$ is the standard deviation. (Similarly for $r_c$.)

Contrary to $r_c$, $r_d \gg 0$ in real-world networks!

10) Degree assortative  11) Degree disassortative
## Clustering mixing (II)

<table>
<thead>
<tr>
<th>Type</th>
<th>Network</th>
<th>(n)</th>
<th>(m)</th>
<th>(k)</th>
<th>(C)</th>
<th>(D)</th>
<th>(r)</th>
<th>(r_c)</th>
<th>(r_d)</th>
<th>(d_i &lt; p_r)</th>
<th>(d_i &lt; p_c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration</td>
<td>condmat</td>
<td>27519</td>
<td>116181</td>
<td>8.4</td>
<td>0.655</td>
<td>0.722</td>
<td>0.166</td>
<td>0.116</td>
<td>0.291</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>comsci</td>
<td>239</td>
<td>568</td>
<td>4.8</td>
<td>0.479</td>
<td>0.561</td>
<td>−0.044</td>
<td>0.123</td>
<td>0.355</td>
<td>6%</td>
<td>6%</td>
</tr>
<tr>
<td>Online social</td>
<td>pgp</td>
<td>10680</td>
<td>24316</td>
<td>4.6</td>
<td>0.266</td>
<td>0.317</td>
<td>0.238</td>
<td>0.497</td>
<td>0.632</td>
<td>27%</td>
<td>27%</td>
</tr>
<tr>
<td></td>
<td>football</td>
<td>115</td>
<td>613</td>
<td>10.7</td>
<td>0.403</td>
<td>0.419</td>
<td>0.162</td>
<td>0.369</td>
<td>0.385</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>jazz</td>
<td>198</td>
<td>2742</td>
<td>27.7</td>
<td>0.617</td>
<td>0.703</td>
<td>0.020</td>
<td>0.008</td>
<td>0.198</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>Social</td>
<td>dolphins</td>
<td>62</td>
<td>159</td>
<td>5.1</td>
<td>0.259</td>
<td>0.319</td>
<td>−0.044</td>
<td>0.192</td>
<td>0.234</td>
<td>15%</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>karate</td>
<td>34</td>
<td>78</td>
<td>4.6</td>
<td>0.571</td>
<td>0.666</td>
<td>−0.476</td>
<td>−0.229</td>
<td>0.277</td>
<td>3%</td>
<td>6%</td>
</tr>
<tr>
<td></td>
<td>emails</td>
<td>1133</td>
<td>5451</td>
<td>9.6</td>
<td>0.220</td>
<td>0.253</td>
<td>0.078</td>
<td>0.214</td>
<td>0.317</td>
<td>14%</td>
<td>15%</td>
</tr>
<tr>
<td>Communication</td>
<td>enron</td>
<td>36692</td>
<td>183831</td>
<td>10.0</td>
<td>0.497</td>
<td>0.530</td>
<td>−0.111</td>
<td>0.185</td>
<td>0.379</td>
<td>4%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>euro</td>
<td>1039</td>
<td>1305</td>
<td>2.5</td>
<td>0.019</td>
<td>0.025</td>
<td>0.090</td>
<td>0.395</td>
<td>0.499</td>
<td>91%</td>
<td>91%</td>
</tr>
<tr>
<td></td>
<td>power</td>
<td>4941</td>
<td>6594</td>
<td>2.7</td>
<td>0.080</td>
<td>0.100</td>
<td>0.003</td>
<td>0.469</td>
<td>0.653</td>
<td>74%</td>
<td>74%</td>
</tr>
<tr>
<td></td>
<td>hepatic</td>
<td>27770</td>
<td>352285</td>
<td>25.4</td>
<td>0.312</td>
<td>0.353</td>
<td>−0.030</td>
<td>0.132</td>
<td>0.370</td>
<td>6%</td>
<td>6%</td>
</tr>
<tr>
<td>Citation</td>
<td>javadoc</td>
<td>2089</td>
<td>7934</td>
<td>7.6</td>
<td>0.373</td>
<td>0.433</td>
<td>−0.070</td>
<td>0.090</td>
<td>0.440</td>
<td>9%</td>
<td>9%</td>
</tr>
<tr>
<td>Documentation</td>
<td>yeast1</td>
<td>2445</td>
<td>6265</td>
<td>5.1</td>
<td>0.215</td>
<td>0.250</td>
<td>−0.101</td>
<td>0.372</td>
<td>0.534</td>
<td>29%</td>
<td>29%</td>
</tr>
<tr>
<td></td>
<td>yeast2</td>
<td>2114</td>
<td>2203</td>
<td>2.1</td>
<td>0.059</td>
<td>0.072</td>
<td>−0.162</td>
<td>0.576</td>
<td>0.675</td>
<td>68%</td>
<td>68%</td>
</tr>
<tr>
<td></td>
<td>javax</td>
<td>1595</td>
<td>5287</td>
<td>6.6</td>
<td>0.381</td>
<td>0.440</td>
<td>−0.120</td>
<td>−0.041</td>
<td>0.545</td>
<td>17%</td>
<td>17%</td>
</tr>
<tr>
<td></td>
<td>jung</td>
<td>317</td>
<td>719</td>
<td>4.5</td>
<td>0.366</td>
<td>0.423</td>
<td>−0.190</td>
<td>0.092</td>
<td>0.443</td>
<td>21%</td>
<td>21%</td>
</tr>
<tr>
<td>Software</td>
<td>guava</td>
<td>174</td>
<td>355</td>
<td>4.1</td>
<td>0.320</td>
<td>0.375</td>
<td>−0.218</td>
<td>0.075</td>
<td>0.734</td>
<td>34%</td>
<td>34%</td>
</tr>
<tr>
<td></td>
<td>java</td>
<td>1516</td>
<td>10049</td>
<td>13.3</td>
<td>0.685</td>
<td>0.731</td>
<td>−0.283</td>
<td>−0.574</td>
<td>0.536</td>
<td>1%</td>
<td>100%</td>
</tr>
<tr>
<td>Web graph</td>
<td>blogs</td>
<td>1490</td>
<td>16715</td>
<td>22.4</td>
<td>0.263</td>
<td>0.293</td>
<td>−0.221</td>
<td>−0.057</td>
<td>0.308</td>
<td>8%</td>
<td>13%</td>
</tr>
<tr>
<td>Metabolic</td>
<td>elegans</td>
<td>453</td>
<td>2025</td>
<td>8.9</td>
<td>0.646</td>
<td>0.710</td>
<td>−0.226</td>
<td>−0.240</td>
<td>0.183</td>
<td>1%</td>
<td>3%</td>
</tr>
<tr>
<td>Internet</td>
<td>oregon</td>
<td>767</td>
<td>1734</td>
<td>4.5</td>
<td>0.293</td>
<td>0.317</td>
<td>−0.299</td>
<td>−0.231</td>
<td>0.262</td>
<td>35%</td>
<td>70%</td>
</tr>
<tr>
<td>Bipartite</td>
<td>women</td>
<td>32</td>
<td>89</td>
<td>5.6</td>
<td>0.000</td>
<td>0.000</td>
<td>−0.337</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

\[ p_r = \frac{k}{n-1} \quad \text{and} \quad p_c \leq \frac{(\sum_i k_i^2 - nk)^2}{n^3 k^3}, \quad \text{while percentages ignore nodes with } k_i \leq 1. \]
**Clustering assortativity**

- \( r_d \gg 0 \) in real-world networks! (\( r_c < 0 \) in disassortative networks.)
- \( d_i \approx 0 \) and \( r_d \gg 0 \) imply connected regions with no clustering.

- \( r_d \) captures how well separated are different network structures.
- \( r_d \not\to 0 \) when \( n \to \infty \) in a random graph, however, \( D \approx 0 \).
Network structures

- Let community be a densely linked group of nodes that are sparsely linked with the rest of the network.
  - Result in degree assortativity, when their sizes differ. (Newman and Park [36])

- Recently, communities are a consequence of clustering. (Foster et al. [10])

- There is substantial evidence that communities appear concurrently with high clustering and assortative mixing by degree. [31, 21, 57]

- Non-social real-world networks greatly deviate from this picture!
Most real-world networks still contain at least some communities.

Community extraction: (Zhao et al. [59])

1. generate a pool of candidate communities,
2. extract community $S$ with the highest value of $W$,

$$W = s(n - s) \left( \frac{\sum_{i \in S} k_i^S}{s^2} - \frac{\sum_{i \in S} k_i - k_i^S}{s(n - s)} \right),$$

where $k_i^S$ and $k_i - k_i^S$ are internal and external degree of node $i$.
3. repeat step 1. until $W$ drops below the value expected at random.

- Extract only the links within $S$, but not those between $S$ and $S^C$!

Communities overlaid over original networks and networks after extraction, respectively.
After extraction of communities, $\approx 80\%$ nodes remain!

Network structure beyond communities is characterized by:
- disassortative mixing by degree,
- lower (degree-corrected) clustering,
- short distances between the nodes.

12) $\#$ nodes $n$

13) LCC

14) Mixing $r$

15) Distances $l$

16) Clustering $C$

17) Mixing $r_d$
Are there mesoscopic structures that could explain these properties?

Let a module be a group of nodes with common neighbors.

Modules coincide with groups of regularly equivalent nodes.

Such modules should result in:

- disassortative mixing by degree, as long as their sizes differ,
- lower (degree-corrected) clustering (absence of triangles),
- short distances between the nodes (efficient global navigation).
Structured-world conjecture: 
Real-world networks are composed of modules characterizing different functions (roles) within the system and overlaid by communities based on some assortative tendency of the nodes, and noise.

- Modules explain degree disassortativity and efficient long-range navigation, whereas communities increase overall clustering and degree assortativity, and explain efficient short-range navigation.
- Structured-world networks must necessarily be heterogeneous!

Note that degree disassortativity and low clustering are already expected properties of scale-free networks.
Outline

1 Motivation

2 Network structure
   - Degree mixing
   - Clustering mixing
   - Network structures
   - Structured-worlds

3 Structure detection
   - Label propagation
   - General propagation

4 Experimental analysis
   - Synthetic networks
   - Real-world networks
   - Software networks

5 Conclusions
Label propagation

- Let $g_i$ be unknown node (module) labels.
- Label propagation algorithm (LPA): (Raghavan et al. [40])
  1. Initialize nodes with unique labels, $g_i = i$,
  2. Node $i$ adopts the label shared by most in $\Gamma_i$,

$$g_i = \arg\max_g \sum_{j \in \Gamma_i} \delta(g_j, g),$$

3. Repeat step 2. until convergence.

- Algorithm has near linear time complexity $O(m^{1.2})$. (Šubelj and Bajec [51])
Convergence issues for, e.g., overlapping communities. $g_i$ is retained, when among most frequent in $\Gamma_i$.

Oscillation of labels in, e.g., bipartite networks. $g_i$ are updated in a random order (sequentially).

Results can be improved by applying node preferences $f_i$. (Leung et al. [22])

$$g_i = \arg\max_g \sum_{j \in \Gamma_i} f_j \cdot \delta(g_j, g)$$
Balanced propagation algorithm (BPA): (Šubelj and Bajec [50])

\[ g_i = \arg\max_g \sum_{j \in \Gamma_i} b_j \cdot \delta(g_j, g), \]

where \( b_i = \frac{1}{1 + e^{-\eta(i - \lambda)}} \) (or \( b_i = i_i \)) and \( i_i \) is index of \( i \), \( i_i \in (0, 1] \).

Algorithm retains scalability, and improves stability and performance.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>karate</th>
<th>dolphins</th>
<th>books</th>
<th>football</th>
<th>jazz</th>
<th>elegans</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPA</td>
<td>184</td>
<td>525</td>
<td>269</td>
<td>414</td>
<td>63</td>
<td>707</td>
</tr>
<tr>
<td>BPA</td>
<td>19</td>
<td>36</td>
<td>29</td>
<td>154</td>
<td>20</td>
<td>75</td>
</tr>
</tbody>
</table>
Defensive propagation algorithm (DPA): (Šubelj and Bajec [51])

\[ g_i = \arg\max_g \sum_{j \in \Gamma_i} p_j \cdot \delta(g_j, g), \]

where \( p_i \) is the probability of a random walker utilized on \( g_i \).

23) Community cores

24) Defensive and offensive propagation

Defensive and offensive prop. obtain high “recall” and “precision”.
General propagation

- Label propagation can detect only connected (cohesive) structures.
- For modules, labels can be propagated through common neighbors!
- General propagation algorithm (GPA): (Šubelj and Bajec [55])

\[ g_i = \arg\max_g \left( \nu_g \sum_{j \in \Gamma_i} f_j \cdot \delta(g_j, g) + (1 - \nu_g) \sum_{j \in \Gamma_i} \sum_{l \in \Gamma_j \setminus \Gamma_i} \tilde{f}_l / k_j \cdot \delta(g_l, g) \right), \]

where \( \nu_g \in [0, 1] \) are parameters and \( f_i = b_i p_i \) (similarly for \( \tilde{f}_i \)).

- \( \nu_g \approx 1 \) and \( \approx 0 \) for communities and modules, respectively.
**General propagation (II)**

- Modeling of $\nu_g$ is of vital importance (guides the algorithm).
  - Dynamic based on conductance $\Phi$. (Šubelj and Bajec [55])
  - Dynamic based on clustering $C$. (Šubelj and Bajec [52])
- Simple model based on clustering $D$ (and mixing $r_d$): (Šubelj and Bajec [54])

$$
\nu_g = \begin{cases} 
1 & \text{for } d_i \geq p_c \ (D \geq p_c), \\
0 & \text{for } d_i < p_c \ (D < p_c), \\
0.5 & \text{otherwise.}
\end{cases}
$$

25) $d_i \geq p_c$ or $d_i < p_c$.

26) $d_i \geq p_c$ and $d_i < p_c$.

- Model seems to ignore most modules (structured-world conjecture)!
Hierarchical propagation

- $k$-partite network on $n$ nodes becomes a clique when $k \to n$ or $n \to k$.
- Modules can become obscure in the presence of communities!
- How community detection algorithms identify network modules?

Dependent modules can be identified as a community, and refined.

- Note that modules must be detected “twice”!
Hierarchical propagation algorithm (HPA):  (Šubelj and Bajec [54])

1. partition the network into communities and modules using GPA,
2. refine each module (step 1.) and accept refinements that increase $\mathcal{L}$,
3. repeat step 1. on a super-network induced by initial structures.

Algorithm reveals entire hierarchy $\mathcal{H}$, where $\mathcal{L}$ is the likelihood of $\mathcal{H}$.

Bottom-most level of $\mathcal{H}$ is reported for structure detection.

Time complexity for each level of $\mathcal{H}$ can be estimated to $\mathcal{O}((km)^{1.2})$. 
Hierarchical propagation (III)

- Single algorithm for communities and modules.
- No prior knowledge is required (e.g., number of structures).
- Algorithm uses only local information (parallelization).
- Relatively simple to extend (e.g., prior knowledge).
- Time complexity is near ideal $O(km)$.
- Relatively simple to implement.
OUTLINE

1 Motivation

2 Network structure
   - Degree mixing
   - Clustering mixing
   - Network structures
   - Structured-worlds

3 Structure detection
   - Label propagation
   - General propagation

4 Experimental analysis
   - Synthetic networks
   - Real-world networks
   - Software networks

5 Conclusions
COMMUNITY DETECTION

Community detection algorithms: greedy modularity \([32, 6]\) (GM), multi-stage modularity \([4]\) (LUV), sequential clique percolation \([18]\) (SCP), Markov clustering \([47]\) (MCL), Infomod \([45]\) (IMD), Infomap \([44]\) (IMP), label propagation \([40]\) (LP) and hierarchical propagation \([54]\) (HP).

27) (Girvan and Newman [11])

28) (Lancichinetti et al. [19]) (small)

29) (Lancichinetti et al. [19]) (big)

30) (Pinkert et al. [39])
31) (ˇSubelj and Bajec [54]) (HN6)
32) (ˇSubelj and Bajec [54]) (HN7)
**REAL-WORLD NETWORKS**


<table>
<thead>
<tr>
<th>Network</th>
<th>NMI</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LUV</td>
<td>MM</td>
</tr>
<tr>
<td>football</td>
<td>0.876</td>
<td>0.823</td>
</tr>
<tr>
<td>karate</td>
<td>0.629</td>
<td><strong>0.912</strong></td>
</tr>
<tr>
<td>jung</td>
<td>0.605</td>
<td>0.662</td>
</tr>
<tr>
<td>women</td>
<td>0.309</td>
<td>0.825</td>
</tr>
</tbody>
</table>

33) Zachary karate net. 34) Davis women net.
Real-world networks (II)

35) jung software network

36) javax software network

37) Amazon web graph

38) Protein interactions
# Real-world networks (III)

<table>
<thead>
<tr>
<th>Network</th>
<th>Module</th>
<th>( n )</th>
<th>( 1 - \Phi )</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core community</td>
<td>[jung.visualization.] *(Server</td>
<td>Viewer</td>
<td>Pane</td>
<td>Model</td>
</tr>
<tr>
<td>5-conf. (upper left)</td>
<td>[jung.algorithms.filters.] <em>Filter</em> (3).</td>
<td>3</td>
<td>0.00</td>
<td>*(jung.graph. *(Graph</td>
</tr>
<tr>
<td>5-conf. (upper right)</td>
<td>[jung.] algorithms.generators.<em>Generator (2); algorithms.importance.</em> (4) algorithms.layout.<em>Layout</em> (3); algorithms.scoring.<em>Scorer (2); algorithms.shortestpath.</em> (2); graph.(*(Graph</td>
<td>Tree</td>
<td>Forest) (4); etc. (interfaces)</td>
<td></td>
</tr>
<tr>
<td>5-conf. (central)</td>
<td>[jung.] algorithms.generators.<em>Generator (2); algorithms.importance.</em> (4) algorithms.layout.<em>Layout</em> (3); algorithms.scoring.<em>Scorer (2); algorithms.shortestpath.</em> (2); graph.(*(Graph</td>
<td>Tree</td>
<td>Forest) (4); etc. (interfaces)</td>
<td></td>
</tr>
<tr>
<td>5-conf. (lower left)</td>
<td>[jung.io.graphml.] <em>Parser</em> (10); etc.</td>
<td>13</td>
<td>0.00</td>
<td><em>(jung.) algorithms.cluster.<em>Clusterer</em> (4); algorithms.generators.random.<em>Generator (5); algorithms.importance.<em>Betweenness</em> (3); algorithms.metrics.</em> (3); algorithms.scoring.</em>* (5); algorithms.shortestpath.* (5); graph.util.* (7); etc. (implementations)</td>
</tr>
<tr>
<td>5-conf. (lower right)</td>
<td>[jung.io.graphml.] <em>Metadata</em> (8); etc.</td>
<td>44</td>
<td>0.03</td>
<td><em>(jung.) algorithms.cluster.<em>Clusterer</em> (4); algorithms.generators.random.<em>Generator (5); algorithms.importance.<em>Betweenness</em> (3); algorithms.metrics.</em> (3); algorithms.scoring.</em>* (5); algorithms.shortestpath.* (5); graph.util.* (7); etc. (implementations)</td>
</tr>
<tr>
<td>2-conf. (upper)</td>
<td>[jung.io.graphml.] <em>Parser</em> (10); etc.</td>
<td>13</td>
<td>0.03</td>
<td><em>(jung.io.graphml.] <em>Parser</em> (10); etc.</em></td>
</tr>
<tr>
<td>2-conf. (lower)</td>
<td>[jung.io.graphml.] <em>Metadata</em> (8); etc.</td>
<td>13</td>
<td>0.38</td>
<td><em>(jung.io.graphml.] <em>Metadata</em> (8); etc.</em></td>
</tr>
<tr>
<td>1-conf. (central)</td>
<td>[jung.visualization.control.] <em>Plugin</em> (2).</td>
<td>2</td>
<td>0.00</td>
<td><em>(jung.visualization.control.] <em>Plugin</em> (2).</em></td>
</tr>
</tbody>
</table>

---

**Structured-world conjecture**

May 3, 2012
# Real-world networks (IV)

<table>
<thead>
<tr>
<th>Network</th>
<th>Module</th>
<th>$n$</th>
<th>$1 - \Phi$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core community</td>
<td>179</td>
<td>0.64</td>
<td></td>
<td>[javax.swing.] plaf.<em>UI (24); plaf.basic.Basic</em>UI (42); plaf.metal.Metal<em>UI (22); plaf.multi.Multi</em>UI (30); plaf.synth.Synth*UI (40); etc.</td>
</tr>
<tr>
<td>3-conf. (upper)</td>
<td>193</td>
<td>0.15</td>
<td></td>
<td>[javax.] accessibility.Accessible* (10); swing.J* (41); swing.**(Border</td>
</tr>
<tr>
<td>3-conf. (left)</td>
<td>113</td>
<td>0.11</td>
<td></td>
<td>[javax.] accessibility.Accessible* (6); swing.* (34); swing.event.*Event (8); swing.event.*Listener (13); swing.plaf.*UI (6); etc.</td>
</tr>
<tr>
<td>3-conf. (lower)</td>
<td>44</td>
<td>0.19</td>
<td></td>
<td>[javax.swing.] text.*View (15); text.html.*View (16); etc.</td>
</tr>
</tbody>
</table>
Structure prediction

- How well the model fits the network observed? Not link prediction!

<table>
<thead>
<tr>
<th>Network</th>
<th>Runs</th>
<th>$-\log L$ and # levels</th>
<th>(Clauset et al. [7])</th>
</tr>
</thead>
<tbody>
<tr>
<td>football</td>
<td>$10^4$</td>
<td>1010.9 3 954.8 5 1004.1 3 884.2 11</td>
<td></td>
</tr>
<tr>
<td>karate</td>
<td>$10^5$</td>
<td>174.1 3 172.3 3 173.9 2 73.3 10</td>
<td></td>
</tr>
<tr>
<td>euro</td>
<td>$10^3$</td>
<td>4108.9 6 3883.2 8 3924.4 5</td>
<td></td>
</tr>
<tr>
<td>yeast2</td>
<td>$10^2$</td>
<td>12495.0 6 11611.2 7 11596.4 4</td>
<td></td>
</tr>
<tr>
<td>javax</td>
<td>$10^2$</td>
<td>13020.7 4 12894.1 4 11512.2 3</td>
<td></td>
</tr>
<tr>
<td>jung</td>
<td>$10^3$</td>
<td>2354.5 5 2312.5 4 2272.9 4</td>
<td></td>
</tr>
<tr>
<td>elegans</td>
<td>$10^2$</td>
<td>8734.1 5 8640.9 6 8243.3 5</td>
<td></td>
</tr>
<tr>
<td>women</td>
<td>$10^4$</td>
<td>193.9 2 163.6 1 163.6 1</td>
<td></td>
</tr>
</tbody>
</table>

39) Module hierarchy 40) Binary hierarchy
Hierarchies revealed with CP and HP algorithms, respectively.

41) javax software network  42) elegans metabolic network

Hierarchies and blockmodels revealed with CP and HP algorithms, respectively.
Software networks

- Software network structures coincide with software packages.
- Communities and modules more accurately predict packages than communities alone!

Blockmodels revealed with CP and HP algorithms, respectively.
Software networks (II)

- Software packages can be predicted with \( \approx 80\% \) accuracy, whereas complete hierarchy can be precisely identified for over 60\% of classes!

<table>
<thead>
<tr>
<th>Network</th>
<th>( l )</th>
<th>( l_\infty )</th>
<th>( P )</th>
<th>( P_4 )</th>
<th>( P_3 )</th>
<th>( P_2 )</th>
<th>( P_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>flamingo</td>
<td>2.65</td>
<td>4</td>
<td>0.566</td>
<td>←</td>
<td>0.572</td>
<td>0.793</td>
<td>1.000</td>
</tr>
<tr>
<td>colt</td>
<td>3.35</td>
<td>4</td>
<td>0.654</td>
<td>←</td>
<td>0.756</td>
<td>0.942</td>
<td>1.000</td>
</tr>
<tr>
<td>jung</td>
<td>2.97</td>
<td>4</td>
<td>0.617</td>
<td>←</td>
<td>0.663</td>
<td>0.857</td>
<td>1.000</td>
</tr>
<tr>
<td>org</td>
<td>3.50</td>
<td>7</td>
<td>0.616</td>
<td>0.616</td>
<td>0.714</td>
<td>0.989</td>
<td>1.000</td>
</tr>
<tr>
<td>weka</td>
<td>3.02</td>
<td>6</td>
<td>0.684</td>
<td>0.692</td>
<td>0.736</td>
<td>0.871</td>
<td>1.000</td>
</tr>
<tr>
<td>javax</td>
<td>3.11</td>
<td>5</td>
<td>0.626</td>
<td>0.631</td>
<td>0.816</td>
<td>0.982</td>
<td>1.000</td>
</tr>
</tbody>
</table>

- Networks should not be combined with the core of the language.
# Outline

1. **Motivation**
2. **Network Structure**
   - Degree mixing
   - Clustering mixing
   - Network structures
   - Structured-worlds
3. **Structure Detection**
   - Label propagation
   - General propagation
4. **Experimental Analysis**
   - Synthetic networks
   - Real-world networks
   - Software networks
5. **Conclusions**
Conclusions

- Structured-world conjecture provides a mesoscopic view on the structure of real-world networks!
  - Different structures imply different macroscopic network properties.
  - Clustering assortativity captures how different modules are merged.
  - Conjecture combines scale-free and small-world phenomena.

- Parameter-free algorithm for detection of communities and modules.
  - Algorithm is (at least) comparable to current state-of-the-art.
  - Network properties could be further utilized within the algorithm!
Future work

- How do dependent modules link between each other?
  - Necessary to develop a measure of module quality.
- Results suggest that module complexity is much larger than expected!

- How to utilize degree mixing within the algorithm?
  - Necessary to analyze networks with millions (billions) of nodes.
Thank you.

✉️ lovro.subelj@fri.uni-lj.si
WWW http://lovro.lpt.fri.uni-lj.si/


