A New Method to Build Higher-Order Representation of Graphs

Philip A. Knight and Luce le Gorrec

Mathematics and Statistics Department,

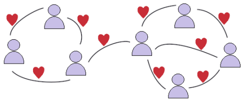
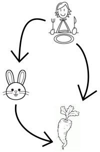
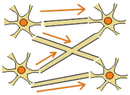
University of Strathclyde

Context – Higher-Order Structures

Royal Academy of Engineering

Graphs: entities (nodes) connected by relations (edges).

✓ Powerful tool to represent complex systems.

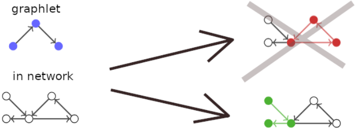
✗ Pairwise relationships: not always enough to capture key properties.

⟹ **Higher-order networks: a hot topic to circumvent this**[1].

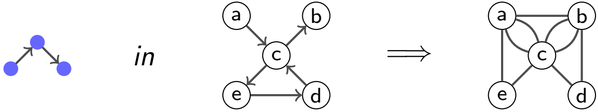
This talk: Two methods to build higher-order networks based on graphlets (motifs).

[1] *Higher-order Network Analysis Takes Off, Fueled by Old Ideas and New Data*, A.Benson, D.Gleich, D. Higham, SIAM News 2021.

- **Graphlet**: Small induced subgraph:



- **Higher-Order Representation**: Weighted motif adjacency matrix[2]:



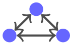
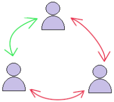
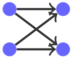
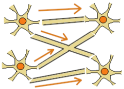
$\Longrightarrow$ **Benson graph** (of some graphlet).

[2] *Tools for higher-order networks analysis*, A. Benson 2017.

Our Rationale

- Graphlets express complex notions within networks:

  3-clique in friendship networks: *bifan in neuronal networks:*

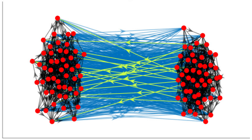





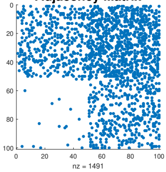
- Benson graph: a non-directed network that conveys this notions.

Our Rationale

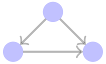
- Graphlets express complex notions within networks.
- Benson graph: a non-directed network that conveys this notions.

$\Longrightarrow$ Of particular interest for partitioning directed networks.

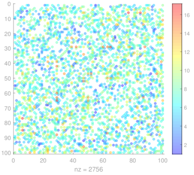




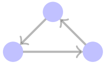
Benson graph of graphlet:



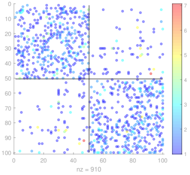
$\Downarrow$



Benson Graph of graphlet:



$\Downarrow$

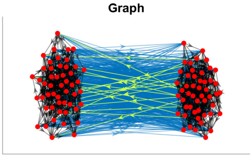


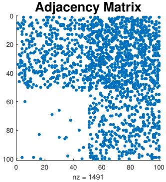
Our Rationale

Royal Academy of Engineering

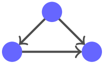
- Graphlets express complex notions within networks.
- Benson graph: a non-directed network that conveys this notions.

$\Longrightarrow$ Of particular interest for partitioning directed networks.

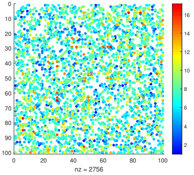




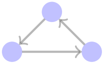
Benson graph of graphlet:



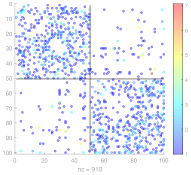
$\Downarrow$



Benson Graph of graphlet:



$\Downarrow$

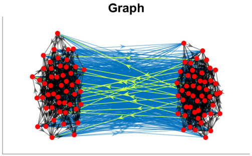


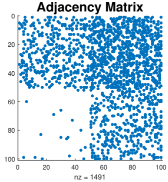
Our Rationale

Royal Academy of Engineering

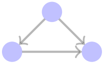
- Graphlets express complex notions within networks.
- Benson graph: a non-directed network that conveys this notions.

$\Longrightarrow$ Of particular interest for partitioning directed networks.

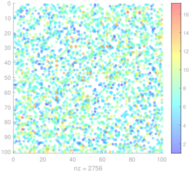




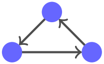
Benson graph
of graphlet:



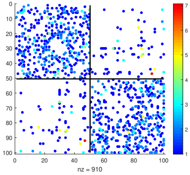
$\Downarrow$



Benson Graph
of graphlet:



$\Downarrow$



Building a Benson Graph

Bottleneck for large networks: enumerating all graphlet instances.

- Techniques for enumerating some specific graphlets.
  - ✗ Often for undirected graphs. Adapted to directed ones to a cost.
  - ✗ Not every graphlets (triangles[3],[4], quadrangles and cliques[4]).

$\Longrightarrow$ What if no prior knowledge about graphlets of interest?

- Techniques for counting some or all graphlets[5],[6].
  - ✗ Enumeration come as a byproduct (post-processing).
  - ✗ More work than needed.

A gap to fill

We propose 2 frameworks based on linear algebra to build the Benson graphs of any 3-and-4-node graphlet.

[3] *Parallel Triangle Counting and Enumeration using Matrix Algebra*, A. Azad et al., 2015.

[4] *Arboricity and Subgraph Listing Algorithms*, N. Chiba, T. Nishizeki, 1985.

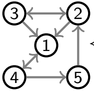
[5] *Efficient Detection of Network Motifs*, S. Wernicke, 2006.

[6] *G-Tries: a Data Structure for Storing and Finding Subgraphs*, P. Ribeiro, F. Silva, 2014.

Our method – Key Notions

Royal Academy of Engineering

A graph $G = (V, E) \iff$ 4 “sparse” matrices that “partition $V \times V$”.

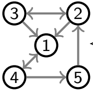


$$\iff \begin{bmatrix} x & & & 1 & \\ 1 & x & 1 & & \\ 1 & 1 & x & & \\ 1 & & & x & 1 \\ & 1 & & & x \end{bmatrix}$$

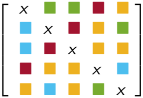
Our method – Key Notions

Royal Academy of Engineering

A graph $G = (V, E)$ $\iff$ 4 "sparse" matrices that "partition $V \times V$".



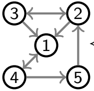
$$\iff \begin{bmatrix} x & & & 1 & \\ 1 & x & 1 & & \\ 1 & 1 & x & & \\ 1 & & & x & 1 \\ & 1 & & & x \end{bmatrix} \iff$$



$$\begin{cases} \text{with:} & \\ \text{if } i \to j \text{ and } i \leftarrow j : & \\ \text{else if } i \to j : & \\ \text{else if } i \leftarrow j : & \\ \text{else :} & \end{cases}$$

Our method – Key Notions

A graph $G = (V, E)$ $\iff$ 4 “sparse” matrices that “partition $V \times V$”.

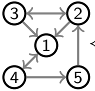


$$\iff \begin{bmatrix} x & & & 1 & \\ 1 & x & 1 & & \\ 1 & 1 & x & & \\ 1 & & & x & 1 \\ & 1 & & & x \end{bmatrix} \iff \begin{bmatrix} x & \color{green}{\blacksquare} & \color{green}{\blacksquare} & \color{red}{\blacksquare} & \color{orange}{\blacksquare} \\ \color{cyan}{\blacksquare} & x & \color{red}{\blacksquare} & \color{orange}{\blacksquare} & \color{green}{\blacksquare} \\ \color{cyan}{\blacksquare} & \color{red}{\blacksquare} & x & \color{orange}{\blacksquare} & \color{orange}{\blacksquare} \\ \color{red}{\blacksquare} & \color{orange}{\blacksquare} & \color{orange}{\blacksquare} & x & \color{cyan}{\blacksquare} \\ \color{orange}{\blacksquare} & \color{cyan}{\blacksquare} & \color{orange}{\blacksquare} & \color{green}{\blacksquare} & x \end{bmatrix} \quad \begin{cases} \text{with:} & & \\ \text{if } i \to j \text{ and } i \leftarrow j: & \color{red}{\blacksquare} & \iff (i,j) \in \mathbf{B} \\ \text{else if } i \to j: & \color{cyan}{\blacksquare} & \iff (i,j) \in \mathbf{U} \\ \text{else if } i \leftarrow j: & \color{green}{\blacksquare} & \iff (i,j) \in \mathbf{Q} \\ \text{else }: & \color{orange}{\blacksquare} & \iff (i,j) \in \mathbf{N} \end{cases}$$

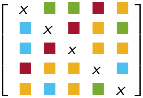
$$\underbrace{\begin{bmatrix} x & \color{green}{\blacksquare} & \color{green}{\blacksquare} & \color{red}{\blacksquare} & \color{orange}{\blacksquare} \\ \color{cyan}{\blacksquare} & x & \color{red}{\blacksquare} & \color{orange}{\blacksquare} & \color{green}{\blacksquare} \\ \color{cyan}{\blacksquare} & \color{red}{\blacksquare} & x & \color{orange}{\blacksquare} & \color{orange}{\blacksquare} \\ \color{red}{\blacksquare} & \color{orange}{\blacksquare} & \color{orange}{\blacksquare} & x & \color{cyan}{\blacksquare} \\ \color{orange}{\blacksquare} & \color{cyan}{\blacksquare} & \color{orange}{\blacksquare} & \color{green}{\blacksquare} & x \end{bmatrix}}_{\mathbf{A}} = \underbrace{\color{red}{\blacksquare} \times \begin{bmatrix} x & & & 1 & \\ & x & 1 & & \\ & 1 & x & & \\ 1 & & & x & \\ & & & & x \end{bmatrix}}_{1\times\mathbf{B}} + \underbrace{\color{cyan}{\blacksquare} \times \begin{bmatrix} x & & & & \\ 1 & x & & & \\ 1 & & x & & \\ & & & x & 1 \\ & 1 & & & x \end{bmatrix}}_{1\times\mathbf{U}} + \underbrace{\color{green}{\blacksquare} \times \begin{bmatrix} x & 1 & 1 & & \\ & x & & & 1 \\ & & x & & \\ & & & x & \\ & & & 1 & x \end{bmatrix}}_{0\times\mathbf{Q}} + \underbrace{\color{orange}{\blacksquare} \times \begin{bmatrix} x & & & & 1 \\ & x & & 1 & \\ & & x & 1 & 1 \\ & 1 & 1 & x & \\ 1 & & 1 & & x \end{bmatrix}}_{0\times\mathbf{N}}$$

Our method – Key Notions

A graph $G = (V, E)$ $\iff$ 4 "sparse" matrices that "partition $V \times V$".

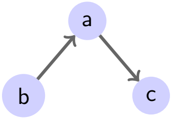


$$\iff \begin{bmatrix} x & & & 1 & \\ 1 & x & 1 & & \\ 1 & 1 & x & & \\ 1 & & & x & 1 \\ & 1 & & & x \end{bmatrix} \iff$$



$$\begin{cases} \text{with:} & \\ \text{if } i \to j \text{ and } i \leftarrow j: & \iff (i,j) \in \mathbf{B} \\ \text{else if } i \to j: & \iff (i,j) \in \mathbf{U} \\ \text{else if } i \leftarrow j: & \iff (i,j) \in \mathbf{Q} \\ \text{else }: & \iff (i,j) \in \mathbf{N} \end{cases}$$

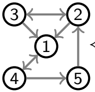
A k-node graphlet $\iff$ a sequence of $k(k-1)/2$ such matrices:



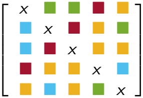
$$\iff \begin{cases} (a, b) \in \mathbf{Q} & \iff \quad \mathbf{Q}(a, b) = 1 \\ (a, c) \in \mathbf{U} & \iff \quad \mathbf{U}(a, c) = 1 \\ (b, c) \in \mathbf{N} & \iff \quad \mathbf{N}(b, c) = 1 \end{cases}$$

Our method – Key Notions

A graph $G = (V, E) \iff$ 4 "sparse" matrices that "partition $V \times V$".



$$\iff \begin{bmatrix} x & & & 1 & \\ 1 & x & 1 & & \\ 1 & 1 & x & & \\ 1 & & & x & 1 \\ & 1 & & & x \end{bmatrix} \iff$$



with:
- if $i \to j$ and $i \leftarrow j$: $\iff (i, j) \in \mathbf{B}$
- else if $i \to j$: $\iff (i, j) \in \mathbf{U}$
- else if $i \leftarrow j$: $\iff (i, j) \in \mathbf{Q}$
- else : $\iff (i, j) \in \mathbf{N}$

A 3-node graphlet $\mathcal{M} = (\{a, b, c\}, E_{\mathcal{M}}) \iff$

$$3 \text{ matrices } \mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \{\mathbf{B}, \mathbf{U}, \mathbf{Q}, \mathbf{N}\} \text{ such that } \begin{cases} (a, b) \in \mathbf{X} \\ (a, c) \in \mathbf{Y} \\ (b, c) \in \mathbf{Z} \end{cases}$$

Our method – Key Notions

Royal Academy of Engineering

A graph $G = (V, E)$, a graphlet $\mathcal{M} = (\{a, b, ...\}, E_{\mathcal{M}})$:

$\mathbf{R}_{a,b} \iff$ the **Role matrix** of (a, b) in G, i.e.:

$$\forall i, j \in V, \mathbf{R}_{a,b}(i, j) = \#i \text{ \& } j \text{ play jointly the role of } a \text{ \& } b\text{, resp.}$$

Our method – Key Notions

Royal Academy of Engineering

A graph $G = (V, E)$, a graphlet $\mathcal{M} = (\{a, b, ...\}, E_{\mathcal{M}})$:

$\mathbf{R}_{a,b} \iff$ the **Role matrix** of (a, b) in G, i.e.:

$$\forall i, j \in V, \mathbf{R}_{a,b}(i,j) = \#i \text{ \& } j \text{ play jointly the role of } a \text{ \& } b \text{, resp.}$$

If $\mathcal{M} = (\{a, b, c\}, E_{\mathcal{M}})$ with $\begin{cases} (a, b) \in \mathbf{X}, \\ (a, c) \in \mathbf{Y}, \\ (b, c) \in \mathbf{Z}, \end{cases}$, then:

$$\boxed{\mathbf{R}_{a,b}(i,j) = |\{u \in V : (i,j) \in \mathbf{X} \& (i,u) \in \mathbf{Y} \& (j,u) \in \mathbf{Z}\}|}$$

Our method – Key Notions

Royal Academy of Engineering

A graph $G = (V, E)$, a graphlet $\mathcal{M} = (\{a, b, ...\}, E_{\mathcal{M}})$:

$\mathbf{R}_{a,b} \iff$ the **Role matrix** of (a, b) in G, i.e.:

$$\forall i, j \in V, \mathbf{R}_{a,b}(i,j) = \#i \text{ \& } j \text{ play jointly the role of } a \text{ \& } b \text{, resp.}$$

If $\mathcal{M} = (\{a, b, c, d\}, E_{\mathcal{M}})$ with $\left\{ \begin{array}{lll} (a,b) \in \mathbf{S}, & (a,c) \in \mathbf{T}, & (a,d) \in \mathbf{W} \\ (b,c) \in \mathbf{X}, & (b,d) \in \mathbf{Y}, & (c,d) \in \mathbf{Z} \end{array} \right.$:

$$\boxed{\mathbf{R}_{a,b}(i,j) = \left| \left\{ u, v \in V : \begin{array}{lll} (i,j) \in \mathbf{S}, & (i,u) \in \mathbf{T}, & (i,v) \in \mathbf{W}, \\ (j,u) \in \mathbf{X}, & (j,v) \in \mathbf{Y}, & (u,v) \in \mathbf{Z} \end{array} \right\} \right|}$$

Our method – Key Notions

Royal Academy of Engineering

A graph $G = (V, E)$, a graphlet $\mathcal{M} = (\{a, b, ...\}, E_{\mathcal{M}})$:

$\mathbf{R}_{a,b} \iff$ the **Role matrix** of (a, b) in G, i.e.:

$$\forall i, j \in V, \mathbf{R}_{a,b}(i,j) = \#i \;\&\; j \text{ play jointly the role of } a \;\&\; b \text{, resp.}$$

and the **Benson matrix** Ω of $\mathcal{M}$ in G is

$$\Omega \propto \mathbf{R}_{a,b} + \mathbf{R}_{b,a} + \mathbf{R}_{a,c} + \mathbf{R}_{c,a} + ...$$

$\Longrightarrow$ Our frameworks consist in computing these Role matrices.

3-Node Graphlet Framework

Royal Academy of Engineering

Graph $G = (V, E)$:

Given $\mathcal{M} = (\{a, b, c\}, E_{\mathcal{M}})$ with $(a, b) \in \mathbf{X}, (a, c) \in \mathbf{Y}, (b, c) \in \mathbf{Z}$:

$$\boxed{\mathbf{R}_{a,b} = \mathbf{X} \circ \left(\mathbf{Y} \times \mathbf{Z}^T\right)}$$

✗ **Main limitation:** When $\mathbf{Y}$ or $\mathbf{Z}$ is the dense matrix $\mathbf{N}$.

✓ **Solution:** Working with $\overline{\mathbf{N}} = \mathbf{J} - \mathbf{N}$ instead[1].

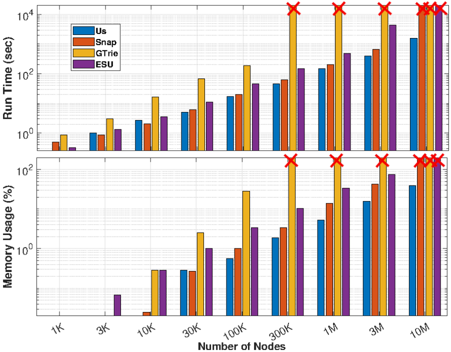
- If $\mathbf{Y} = \mathbf{N}$: $$\mathbf{R}_{a,b} = \mathbf{X} \circ (\mathbf{Y}\mathbf{Z}^T) = \mathbf{X} \circ (\underbrace{\mathbf{J}\mathbf{Z}^T}_{rank1} - \overline{\mathbf{N}}\mathbf{Z}^T).$$
- If $\mathbf{Z} = \mathbf{N}$: $$\mathbf{R}_{a,b} = \mathbf{X} \circ (\mathbf{Y}\mathbf{Z}^T) = \mathbf{X} \circ (\underbrace{\mathbf{Y}\mathbf{J}}_{rank1} - \mathbf{Y}\overline{\mathbf{N}}).$$

NB : No more than 1 matrix among $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ can be equal to $\mathbf{N}$.

[1] $\mathbf{J}$ is the matrix of 1s.

3-Node Graphlet Framework

Royal Academy of Engineering



Task: Finding all 3-node graphlet Benson Graphs on my laptop.

- All in C/C++, 8 threads.
- Us : Our framework (GraphBlas[7]).
- SNAP : Own // of the code from[8].
- GTrie: // version of the GTrie algorithm[9].
- ESU: Own implementation of [10].
- ✗ $\iff$ Unfinished (>3hours).

$\Longrightarrow$ **Our method outperforms the others**, both in terms of runtime and memory usage. SNAP is only slightly worse.

[7] *Parallel GraphBLAS with OpenMP*, M. Aznaveh *et al.*, GrAPL 2019.
[8] *Higher-order Organization of Complex Networks.*, A. Benson *et al.*, Science 2016.
[9] *Parallel subgraph counting for multicore architectures*, D. Aparicio *et al.*, ISPA 2014.
[10] *Efficient Detection of Network Motifs*, S. Wernicke, 2006.

4-Node Graphlet Framework

A Graph $G = (V, E)$:

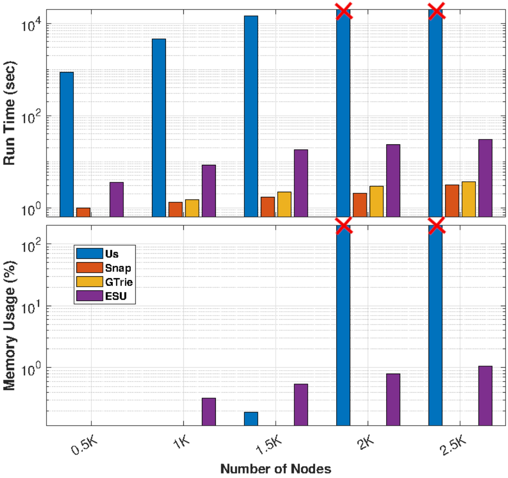
Given $\mathcal{M} = (\{a, b, c, d\}, E_{\mathcal{M}})$ with $\begin{cases} (a,b) \in \mathbf{S}, & (a,c) \in \mathbf{T}, \quad (a,d) \in \mathbf{W} \\ (b,c) \in \mathbf{X}, & (b,d) \in \mathbf{Y}, \quad (c,d) \in \mathbf{Z} \end{cases}$

$$\mathbf{R}_{a,b} = \mathbf{S} \circ \Psi, \qquad \text{with } \Psi \text{ “just” a reshape of}$$

$$\left(\begin{bmatrix} \mathbf{Y} \times \mathcal{D}(\mathbf{W}(1,:)) \\ \vdots \\ \mathbf{Y} \times \mathcal{D}(\mathbf{W}(n,:)) \end{bmatrix} \circ \begin{bmatrix} \mathbf{X} \times \mathcal{D}(\mathbf{T}(1,:)) \times \mathbf{Z} \\ \vdots \\ \mathbf{X} \times \mathcal{D}(\mathbf{T}(n,:)) \times \mathbf{Z} \end{bmatrix} \right) \times \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

which **involves** $|V|$ **matrix-matrix products**.

4-Node Graphlet Framework



Task:

- GTrie and ESU: finding all 4-node graphlets.
- Snap[11] and Us: finding quadrangles.

✗ Our method is extremely slow ($\sim 10^3$sec for a 500-node network).

Why? This version works directly with dense matrix $\mathbf{N}$.

$\Longrightarrow$ Deriving tricks to avoid $\mathbf{N}$ is a work in progress.
$\Longrightarrow$ Good hope it will improve the perf. significantly.

[11] *Arboricity and Subgraph Listing Algorithms*, N. Chiba, T. Nishizeki, SIAM Journal on Computing, 1985.

Take-Home Messages

Royal Academy of Engineering

- **Benson graphs improve the potential of classical networks** by conveying additional information.
- There is **a gap to fill to efficiently build these graphs**.
- We have proposed **two methods** to build them **for any 3-and-4 node graphlets**, with **no prior knowledge** about the graphlet.
- An implementation for **3-node graphlets competitive with state-of-the-art** techniques.
- Room for improvement considering 4-node graphlets.

Codes available soon on `github.com/luleg/`.

Thank you for your attention.
Any question ?