Link Reliability

Missing and spurious interactions in complex networks

Paolo Lapo Cerni

20 September 2024

Università degli Studi di Padova

Introduction

- Network science aims to unfold the functional needs of a system by looking at the **interactions between units**.

Guimerà, Roger, and Marta Sales-Pardo. "Missing and spurious interactions and the reconstruction of complex networks." *Proceedings of the National Academy of Sciences* 106.52 (2009): 22073-22078.

Introduction

- Network science aims to unfold the functional needs of a system by looking at the **interactions between units**.
- Unfortunately, the **reliability of network** data is often a source of concern.

Guimerà, Roger, and Marta Sales-Pardo. "Missing and spurious interactions and the reconstruction of complex networks." *Proceedings of the National Academy of Sciences* 106.52 (2009): 22073-22078.

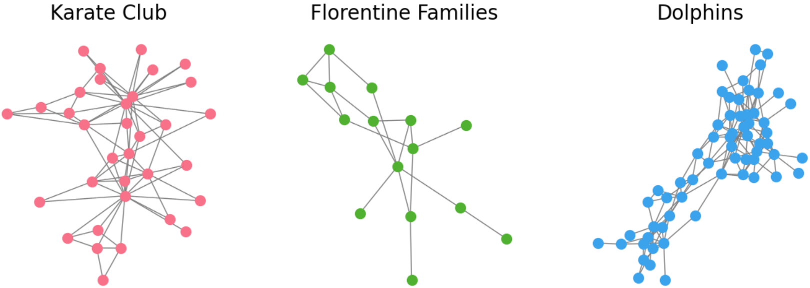
Introduction

- Network science aims to unfold the functional needs of a system by looking at the **interactions between units**.
- Unfortunately, the **reliability of network** data is often a source of concern.
- In this presentation, we will examine a framework to assess the reliability of complex networks, based on **Bayesian inference**.

Guimerà, Roger, and Marta Sales-Pardo. "Missing and spurious interactions and the reconstruction of complex networks." *Proceedings of the National Academy of Sciences* 106.52 (2009): 22073-22078.

Network examples

We compare three **different networks** with different numbers of nodes, structures, and complexities:



SBM (1)

We focus on the family $\mathcal{M}_{BM}$ of the **stochastic block model** (SBM).

SBM (1)

We focus on the family $\mathcal{M}_{BM}$ of the **stochastic block model** (SBM).

This comes with several advantages:

- It is **empirically grounded**: modular structured, role-to-role connected.
- Is is analytically and computationally **tractable**.

Thus, sampling over instances $M \in \mathcal{M}_{BM}$ captures a variety of correlations.

SBM (1)

We focus on the family $\mathcal{M}_{BM}$ of the **stochastic block model** (SBM).

This comes with several advantages:

- It is **empirically grounded**: modular structured, role-to-role connected.
- Is is analytically and computationally **tractable**.

Thus, sampling over instances $M \in \mathcal{M}_{BM}$ captures a variety of correlations.

Stochastic block model

A block model $M = (P, \mathbf{Q})$ is completely defined by the partition P of nodes into groups and the matrix $\mathbf{Q}$ of probabilities of connections between groups.

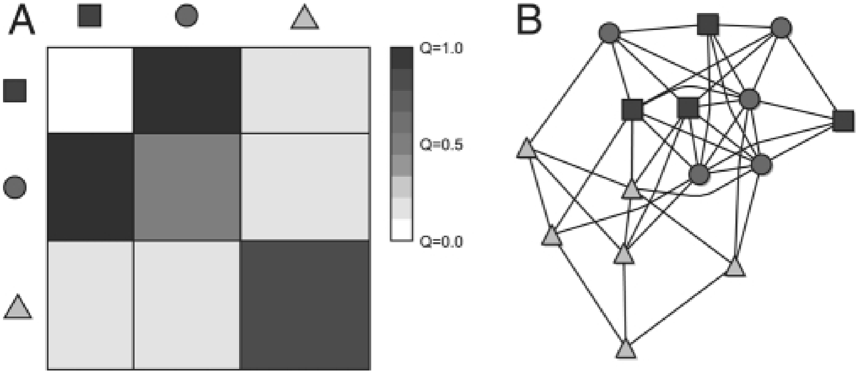


Figure: *(A)* Probability matrix $\mathbf{Q}$ of a SBM $M = (P, \mathbf{Q})$, being $P = (4, 5, 6)$. *(B)* A realization of the model described in A

Reliability

Considering an **observed network** A^O and a set of generative models $\mathcal{M}$, we can compute the probability $p(X = x|A^O)$ for an **arbitrary network property** X as:

$$p(X = x|A^O) = \int_{\mathcal{M}} dM\, p(X = x|M)\, p(M|A^O)$$

Considering an **observed network** A^O and a set of generative models $\mathcal{M}$, we can compute the probability $p(X = x|A^O)$ for an **arbitrary network property** X as:

$$p(X = x|A^O) = \int_{\mathcal{M}} dM\, p(X = x|M)\, p(M|A^O)$$

Using the **Bayes theorem** we can rewrite

$$p(M|A^O) = \frac{p(A^O|M)p(M)}{\int_{\mathcal{M}} dM' p(A^O|M')p(M')}$$

Considering an **observed network** A^O and a set of generative models $\mathcal{M}$, we can compute the probability $p(X = x|A^O)$ for an **arbitrary network property** X as:

$$p(X = x|A^O) = \frac{1}{Z} \int_{\mathcal{M}} dM \, p(X = x|M) \, p(A^O|M) \, p(M)$$

Reliability

Considering an **observed network** A^O and a set of generative models $\mathcal{M}$, we can compute the probability $p(X = x|A^O)$ for an **arbitrary network property** X as:

$$p(X = x|A^O) = \frac{1}{Z} \int_{\mathcal{M}} dM \, p(X = x|M) \, p(A^O|M) \, p(M)$$

Formally, the block model $M = (P, \mathbf{Q})$ is completely determined by the partition P of nodes into groups and the matrix $\mathbf{Q}$ of probabilities.

Reliability

Considering an **observed network** A^O and a set of generative models $\mathcal{M}$, we can compute the probability $p(X = x|A^O)$ for an **arbitrary network property** X as:

$$p(X = x|A^O) = \frac{1}{Z} \sum_{P \in \mathcal{P}} \int_{[0,1]^G} dQ \, p(X = x|P, Q) \, p(A^O|P, Q) \, p(P, Q)$$

Formally, the block model $M = (P, \mathbf{Q})$ is completely determined by the partition P of nodes into groups and the matrix $\mathbf{Q}$ of probabilities.

Reliability

Considering an **observed network** A^O and a set of generative models $\mathcal{M}$, we can compute the probability $p(X = x|A^O)$ for an **arbitrary network property** X as:

$$p(X = x|A^O) = \frac{1}{Z} \sum_{P \in \mathcal{P}} \int_{[0,1]^G} dQ\, p(X = x|P, Q)\, p(A^O|P, Q)\, p(P, Q)$$

The likelihood of each model M can be written as:

$$p(A^0|P, Q) = \prod_{\alpha \leq \beta} Q_{\alpha\beta}^{l_{\alpha\beta}^O} (1 - Q_{\alpha\beta})^{r_{\alpha\beta} - l_{\alpha\beta}^O}$$

where $l_{\alpha\beta}^O$ is the number of links in A^O between nodes in groups α and β of P, and $r_{\alpha\beta}$ is the maximum number of such links.

Reliability

Considering an **observed network** A^O and a set of generative models $\mathcal{M}$, we can compute the probability $p(X = x|A^O)$ for an **arbitrary network property** X as:

$$p(X = x|A^O) = \frac{1}{Z} \sum_{P \in \mathcal{P}} \int_{[0,1]^G} dQ \, p(X = x|P, Q) \, p(A^O|P, Q) \, p(P, Q)$$

We can assume an uninformative prior, i.e.:

$$p(P, Q) \sim \text{const}$$

Reliability

Considering an **observed network** A^O and a set of generative models $\mathcal{M}$, we can compute the probability $p(X = x|A^O)$ for an **arbitrary network property** X as:

$$p(X = x|A^O) = \frac{1}{Z} \sum_{P \in \mathcal{P}} \int_{[0,1]^G} dQ \, p(X = x|P, Q) \, p(A^O|P, Q) \, p(P, Q)$$

Then, we can study the property of having a link between nodes i and j, given their groups σ_i and σ_j in P:

$$p(A_{ij} = 1|P, Q) = Q_{\sigma_i \sigma_j}$$

Link reliability

We can write the **link reliability** $R^L_{ij} = p(A_{ij} = 1|A^O)$ as:

$$R^L_{ij} = \frac{1}{Z} \sum_{P \in \mathcal{P}} \left(\frac{l_{\sigma_i \sigma_j} + 1}{r_{\sigma_i \sigma_j} + 2} \right) \exp[-\mathcal{H}(P)]$$

Link reliability

We can write the **link reliability** $R^L_{ij} = p(A_{ij} = 1 | A^O)$ as:

$$R^L_{ij} = \frac{1}{Z} \sum_{P \in \mathcal{P}} \left(\frac{l_{\sigma_i \sigma_j} + 1}{r_{\sigma_i \sigma_j} + 2} \right) \exp[-\mathcal{H}(P)]$$

where the hamiltonian $\mathcal{H}(P)$ is

$$\mathcal{H}(P) = \sum_{\alpha \leq \beta} \left[\ln(r_{\alpha\beta} + 1) + \ln \binom{r_{\alpha\beta}}{l^O_{\alpha\beta}} \right]$$

and Z is the normalization.

Link reliability

We can write the **link reliability** $R^L_{ij} = p(A_{ij} = 1|A^O)$ as:

$$R^L_{ij} = \frac{1}{Z} \sum_{P \in \mathcal{P}} \left(\frac{l_{\sigma_i\sigma_j} + 1}{r_{\sigma_i\sigma_j} + 2} \right) \exp[-\mathcal{H}(P)]$$

In practice, it's impossible to sum over all the possible partitions.
We can treat R^L_{ij} as an **ensemble average** and use the **Metropolis algorithm** to sample the relevant contributions.

Sampling procedure (1)

We start initializing the N nodes into N groups, uniformly at random:

```
# Groups data structure: list of lists
N = A.shape[0]
groups = [[] for _ in range(N)]

# Uniformly random initialization
for i in range(N):
    g = np.random.randint(0, N)
    groups[g].append(i)

# Compute the hamiltonian
H = hamiltonian(A, groups)
```

Sampling procedure (1)

We start initializing the N nodes into N groups, uniformly at random:

```
# Groups data structure: list of lists
N = A.shape[0]
groups = [[] for _ in range(N)]

# Uniformly random initialization
for i in range(N):
    g = np.random.randint(0, N)
    groups[g].append(i)

# Compute the hamiltonian
H = hamiltonian(A, groups)
```

At each step, we select a random node and attempt to move it into a new random group:

```
# Randomly select a node and a group
i = np.random.randint(0, N)
g_prop = np.random.randint(0, N)

# Move the node to another group
groups_prop = swap(groups, i, g_prop)
```

Sampling procedure (2)

Then, we compute $\Delta\mathcal{H}$:

- If $\Delta\mathcal{H} \leq 0$ we accept the change
- Otherwise, the change is accepted with probability $\exp(-\Delta\mathcal{H})$

```python
# Compute the Hamiltonian of the new configuration
H_prop = hamiltonian(A, groups_prop)

# Acceptance probability
if H_prop <= H:
    groups = groups_prop
    H = H_prop
else:
    r = np.random.rand()
    if r < np.exp(H - H_prop):
        groups = groups_prop
        H = H_prop

return groups, H
```

The sampling procedure starts after an **equilibration period**.

```
# Transient
for _ in range(transient):
    groups, H = singleStep(groups, H, A)
```

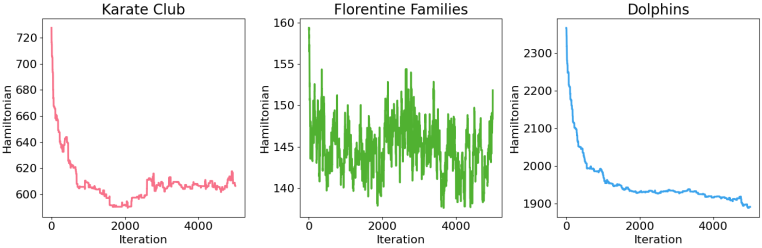


Figure: Transient time

Sampling procedure (4)

The sampling procedure must consider only **uncorrelated partitions**.

```
for k in range(n_samples):
    for _ in range(delay):
        groups, H = singleStep(groups, H, A)
    partitions_set.append(groups)
    hamiltonians_list.append(H)
```

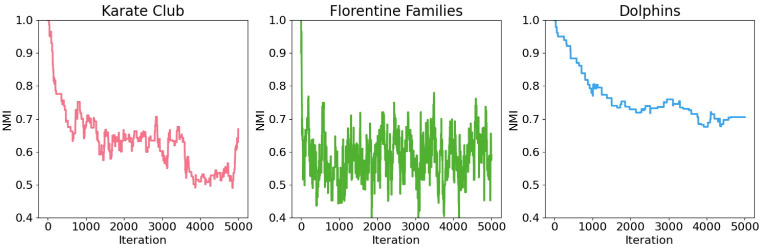


Figure: Normalized Mutual Information

Sampling procedure (5)

Offset

You can always **rescale the hamiltonian** by choosing an offset.

Sampling procedure (5)

Offset

You can always **rescale the hamiltonian** by choosing an offset.

- This is meant to avoid roundoff errors and vanishing information.
- It allows you to work with bigger networks.
- Working with huge networks would require studying the fluctuations around the mean.

```
np.exp(-np.array(hamiltionian_list, dtype=np.float128) + offset)
```

Sampling procedure (6)

The reliability is an ensemble average over independent partitions: one can **parallelize the algorithm** and obtain the partitions concurrently.

Sampling procedure (6)

The reliability is an ensemble average over independent partitions: one can **parallelize the algorithm** and obtain the partitions concurrently.

```
with multiprocessing.Pool(processes=n_cores) as pool:
    results = pool.starmap(samplingBranch, input)
```

Sampling procedure (6)

UNIVERSITÀ DEGLI STUDI DI PADOVA

The reliability is an ensemble average over independent partitions: one can **parallelize the algorithm** and obtain the partitions concurrently.

```
with multiprocessing.Pool(processes=n_cores) as pool:
    results = pool.starmap(samplingBranch, input)
```

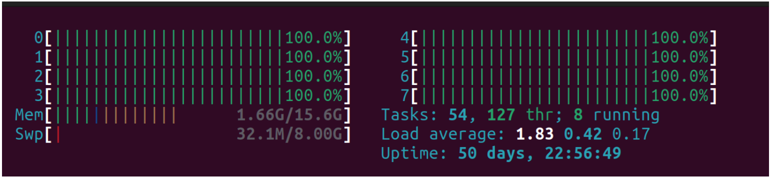


Figure: A snapshot of the system displayed by `htop` in the terminal.

Corrupt a graph

Given a "true" network A^T with E links, we want to generate a hypothetical observation A^O by adding/removing a fraction f of edges from A^T.

Corrupt a graph

Given a "true" network A^T with E links, we want to generate a hypothetical observation A^O by adding/removing a fraction f of edges from A^T.

Missing interactions: A^O is obtained by removing $\lceil f\,E \rceil$ edges from A^T

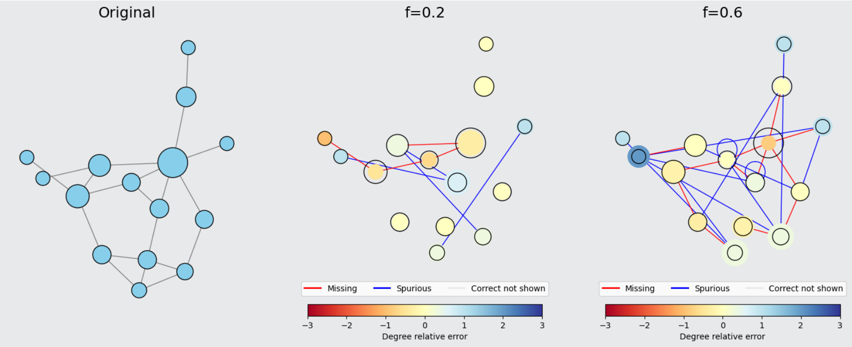
Spurious interactions: A^O is obtained by adding $\lceil f\,E \rceil$ edges to A^T

Corrupt a graph

Given a "true" network A^T with E links, we want to generate a hypothetical observation A^O by adding/removing a fraction f of edges from A^T.

Missing interactions: A^O is obtained by removing $\lceil f\, E \rceil$ edges from A^T

Spurious interactions: A^O is obtained by adding $\lceil f\, E \rceil$ edges to A^T

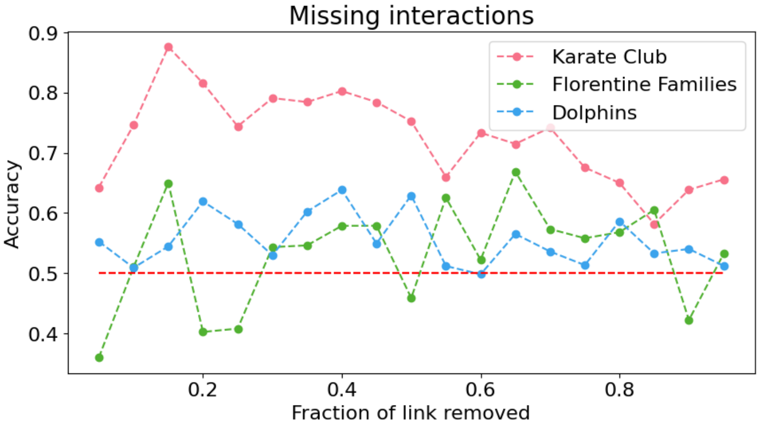


Test link reliability (1)

To test the ability to identify **missing interactions**, we compute the probability that *a false negative has a higher reliability than a true negative*.

Test link reliability (1)

To test the ability to identify **missing interactions**, we compute the probability that *a false negative has a higher reliability than a true negative*.

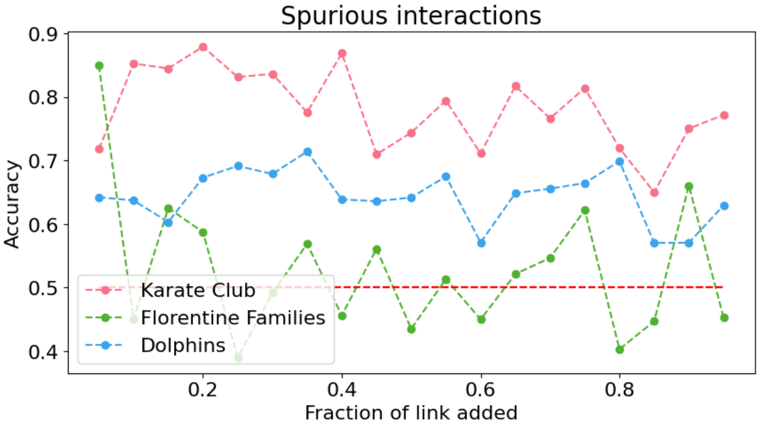


Test link reliability (2)

To test the ability to identify **spurious interactions**, we compute the probability that *a false positive has a lower reliability than a true positive*.

Test link reliability (2)

To test the ability to identify **spurious interactions**, we compute the probability that *a false positive has a lower reliability than a true positive*.



Network reliability

Similarly, one can test the **network reliability** $R_A^N = p(A|A^O)$ that can be written as:

$$R_A^N = \frac{1}{Z} \sum_{P \in \mathcal{P}} h(A; A^O, P) \exp(-\mathcal{H}(P))$$

Network reliability

Similarly, one can test the **network reliability** $R_A^N = p(A|A^O)$ that can be written as:

$$R_A^N = \frac{1}{Z} \sum_{P \in \mathcal{P}} h(A; A^O, P) \exp(-\mathcal{H}(P))$$

where

$$h(A; A^O, P) = \exp\left\{ \sum_{\alpha \le \beta} \left[\ln\left(\frac{r_{\alpha\beta} + 1}{2r_{\alpha\beta} + 1} \right) + \ln\left(\frac{\binom{r_{\alpha\beta}}{l^O_{\alpha\beta}}}{\binom{2r_{\alpha\beta}}{l_{\alpha\beta} + l^O_{\alpha\beta}}} \right) \right] \right\}$$

and

$$\mathcal{H}(P) = \sum_{\alpha \le \beta} \left[\ln(r_{\alpha\beta} + 1) + \ln\binom{r_{\alpha\beta}}{l^O_{\alpha\beta}} \right]$$

Network reconstruction (1)

We want to **reconstruct the network** A^T from its partially corrupted observed version A^O, with both missing and spurious interactions.

Network reconstruction (1)

We want to **reconstruct the network** A^T from its partially corrupted observed version A^O, with both missing and spurious interactions.

Summing over all possible networks to obtain an ensemble average is prohibitive. Thus, we want to **find the network that maximizes** R_A^N

Optimal network

We want to find $A^R = \arg\max_A R_A^N$

Network reconstruction (1)

We want to **reconstruct the network** A^T from its partially corrupted observed version A^O, with both missing and spurious interactions.

Summing over all possible networks to obtain an ensemble average is prohibitive. Thus, we want to **find the network that maximizes** R_A^N

Optimal network

We want to find $A^R = \arg\max_A R_A^N$

Unfortunately, a full scan in the configuration space is unfeasible. We need to define an **heuristic maximization method**.

```python
hyper = (n_samples, delay, transient) # Hyperparameters
A_cur = A_obs.copy() # Initialize the current adjacency matrix

for j in range(5): # This is the maximum number of iterations
    # Compute the network reliability and the link reliability matrix
    R_N = getNetworkReliability(A_cur, A_obs, *generatePartitionsSet(A_obs, *hyper))
    R_L = computeLinkReliabilityMatrix(A_cur, *generatePartitionsSet(A_cur, *hyper))

    # Sort the links (increasing R) and not links (decreasing R)
    sorted_links, sorted_not_links = sortLinkLists(*getLinkLists(A_cur), R_L)

    # Iterate to reconstruct the network
    num_iters = np.min([len(sorted_links), len(sorted_not_links)])
    miss_update = 0
    done_update = False
    for k in range(num_iters):
        # Choose a pair of link and not link in order
        link, not_link = sorted_links[k], sorted_not_links[k]

        # Define the proposed adjacency matrix
        A_temp = A_cur.copy()

        # Swap the links
        A_temp[link[0], link[1]] = 0
        A_temp[link[1], link[0]] = 0
        A_temp[not_link[0], not_link[1]] = 1
        A_temp[not_link[1], not_link[0]] = 1

        # Compute the network reliability with the new adjacency matrix
        R_N_temp = getNetworkReliability(A_temp, A_obs, *generatePartitionsSet(A_obs, *hyper))

        # If the network reliability increases, update the adjacency matrix
        if R_N_temp > R_N:
            A_cur = A_temp
            R_N = R_N_temp
            miss_update = 0
            done_update = True
        else:
            miss_update += 1
            # Break condition for the iteration
            if miss_update > 4:
                break

    # Break condition for the whole loop
    if not done_update:
        break
```

Rationale of network reconstruction

The crucial assumption is that $R^N_{A^T} \gg R^N_{A^O}$.

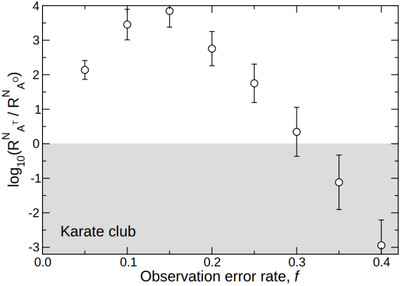


Figure: From the *Supporting Information* of Guimera and Sales-Pardo.

Rationale of network reconstruction

The crucial assumption is that $R^N_{A^T} \gg R^N_{A^O}$.

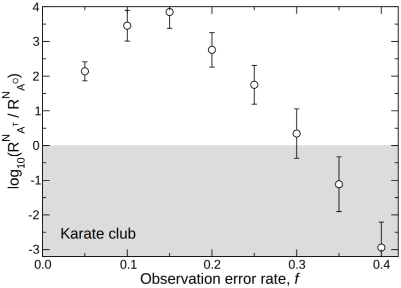


Figure: From the *Supporting Information* of Guimera and Sales-Pardo.

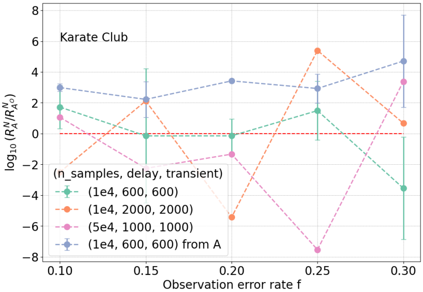
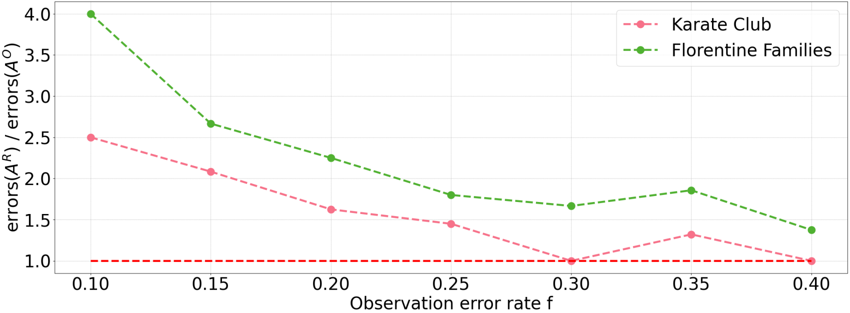


Figure: Preliminary studies on network reconstruction.

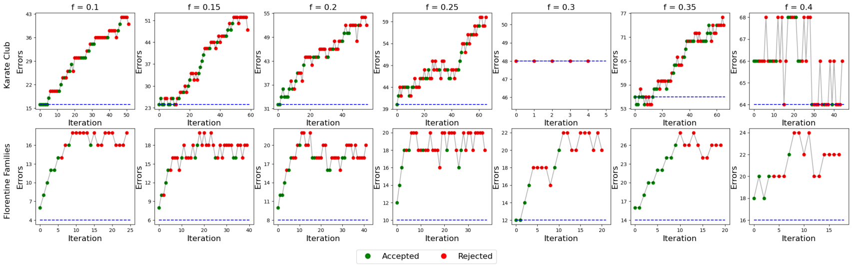
Network reconstruction (4)

This leads to a **failure in the reconstruction** of the true network



Network reconstruction (4)

This leads to a **failure in the reconstruction** of the true network



Possible improvements

The **network reconstruction framework** should be debugged by:

- increasing transient, decorrelation time, and number of samples
- changing the underlying groups' data structure to speed up the computations and the debugging

Possible improvements

Università degli Studi di Padova

The **network reconstruction framework** should be debugged by:

- increasing transient, decorrelation time, and number of samples
- changing the underlying groups' data structure to speed up the computations and the debugging

Moreover, one can study **other optimizations**:

- (partially) shared samples from A^O along the iterations
- dynamic decorrelation time based on NMI

Possible improvements

The **network reconstruction framework** should be debugged by:

- increasing transient, decorrelation time, and number of samples
- changing the underlying groups' data structure to speed up the computations and the debugging

Moreover, one can study **other optimizations**:

- (partially) shared samples from A^O along the iterations
- dynamic decorrelation time based on NMI

Finally, it would be **interesting to study**:

- the role of the network topology with synthetic data
- the impact of possible node metadata

Conclusions

We review this **bayesian framework** to test **interactions in complex networks**.

- We developed the notion of reliability in the context of the SBM.
- We implemented a sampling procedure to compute the ensemble averages.
- We studied the link reliability for missing and spurious interactions.
- We introduced the network reliability.