DUCROCQ Romain, M2 SIA

[Management des SI]
Project 1: Reliability

Github: https://github.com/romainducrocq/Reliability_DUCROCQ-Romain

Path to results: code/data/reliability_result.mat

Introduction

The aim of this project was to implement reliability metrics presented in the paper *"Network Reliability and Resilience of Rapid Transit Systems"*, apply them to a 100x100 adjacency matrix and discuss the results. These metrics are used to quantify the reliability of a graph and assess its tolerance to disruptions.
Here, we will study the simplest case, where no additional disruptions are introduced, also called status quo. We will measure Rod, Rnode, Rrange and Rsys, and discuss the resilience of the graph in light of these results.

Methodology

1. Rod Matrix

Rod measures the reliability for each possible path between two nodes, known as origin and destination. To compute all paths from one to the other, the article mentions the use of a k-shortest path algorithm. I have therefore used the Lawler-Yen K-shortest path algorithm, a modification of the original Yen algorithm, which does not explore duplicate paths rather than discarding them afterwards. This decreases the time complexity of the search by a factor n (number of nodes), from $o(kn^4)$ to $o(kn^3)$, and thus supposedly divides the computation time by a 100 times in our case. My algorithm has no fixed K, but loops until the first path with an infinity weight is found, and therefore finds all the k-shortest loopless paths. With this approach, I go through all the 100x100 pairs of origin-destination nodes in approximately 25 minutes of computation time.

The Rod value is based on the probabilities for disjoint events p(Dk) and the probabilities of available paths p(Ek), where the path Ek is composed of the set of links {ei, i = 1:m}, and p(Ek) defined as the product of all p(ei). However, p(ei) is described as the empirical reliability at link i, and we do not have empirical measures for our network. We therefore define p(ei) as *"the probability that it is the ei link that is not 'cut' among all those that make up the path from origin to destination".* This probability is (1 - (1 / m)), with (m = n - 1) for the path Ek. Thus, for a given path Ek, p(e1) = p(e2) = … = p(em), and p(Ek) = (1 - (1 / m)) ** m.

DUCROCQ Romain, M2 SIA

The function RodMat(adjacency, name) in RodMat.m takes an adjacency matrix as input and computes the Rod for all pairs of origin-destination nodes. It then stores the 100x100 resulting matrix in a {name}_rod.mat file.

```
>> load data/adjacency.mat
>> RodMat(adjacency, "data/reliability")
```

Here, we load the adjacency matrix and store it's Rod matrix in data/reliability_rod.mat. It is not advised to replicate this as the computation takes around 25 minutes to complete. Instead, I have added two randomly generated 20x20 test matrices in the folder data/test. data/test/test.mat is dense (p=0.5), and data/test/test2.mat is sparse (p=0.1). Use as follow:

```
>> load data/test/test.mat
>> RodMat(test, "data/test/test")
```

```
>> load data/test/test2.mat
>> RodMat(test2, "data/test/test2")
```

2. Reliability metrics

Rnode, Rrange and Rsys are based on Rod. They all take the Rod Matrix as an input and compute the metrics with it. This way, the measures are obtained very quickly once the Rod Matrix is done, since the Rods are never recomputed.

The function Reliability(name) in Reliability.mat loads the Rod Matrix from the {name}_rod.mat file and computes the metrics: Rnode and Rrange for each pair of origin-destination nodes, and Rsys for the matrix. It then stores the values in the file {name}_result.mat. Use as follow:

Project:

```
>> Reliability("data/reliability")
```

Tests:

```
>> Reliability("data/test/test")
>> Reliability("data/test/test2")
```

3. Display results

The function PrintResult(name, plots, sortMode) in PrintResult.m displays the results obtained at the previous step. It loads all the Rnodes, Rranges and the Rsys from the {name}_result.mat file and displays them depending on the given inputs:

- [plots] 1: plot the Rnodes and Rranges, 0: don't plot (if gnu.plot not installed);
- [sortMode] 1: Rnodes ascending, 2: Rnodes descending, 0: don't sort.

Use as follow:

Project:

```
>> PrintResult("data/reliability", 1, 2)
```

Tests:

```
>> PrintResult("data/test/test", 1, 2)
```

```
>> PrintResult("data/test/test2", 1, 2)
```

DUCROCQ Romain, M2 SIA

NB: Complete instructions

	Project	Test 1	Test 2
Rod Matrix	>> load data/adjacency.mat >> RodMat(adjacency, "data/reliability") **(~ 25 minutes)**	>> load data/test/test.mat >> RodMat(test, "data/test/test")	>> load data/test/test2.mat >> RodMat(test2, "data/test/test2")
Rnodes, Rranges, Rsys	>> Reliability("data/reliability")	>> Reliability("data/test/test")	>> Reliability("data/test/test2")
Display	>> PrintResult("data/reliability", 1, 2)	>> PrintResult("data/test/test", 1, 2)	>> PrintResult("data/test/test2", 1, 2)

Results

1. Hypothesis

Before looking at the results, we should anticipate what kind of measures we can expect. By looking at the adjacency matrix, we observe that the graph is very sparse. When we compute the incidence matrix for our network, we find out that it contains only E = 761 edges. The density of a directed graph is given by (E / (n * (n - 1))), which here equals to D = (761 / 9900) = 0.07. Hence, we can safely assume the graph to be very sparse. In comparison, the graph in Test 1 has a density of D = 0.5, and the graph in Test 2 has a density of D = 0.09.

This first analysis indicates that we will probably have low reliability scores overall. In fact, a very low density indicates that there are on average very few alternative paths between a pair of origin-destination nodes. Any cut has therefore a significant probability to disjoint the given sub-graph.

To get an idea of the result, we can look at the test cases. Test 1, with a density of 0.5, has all its Rnodes equal to 0.99, Rranges lower than 0.001 and an Rsys of 0.99. It is very reliable, and will present a strong tolerance to disruptions. In practice, it would be almost impossible for it to collapse due to random events.
Test 2 has a density of 0.09 and much lower scores. The Rnodes vary from 0.58 to 0.25, and the Rranges have the same order of magnitude. Five nodes have an Rnode of 0, which means that they are disconnected. The Rsys is 0.3. This graph is much less reliable than the first one, and will have a weaker tolerance to disruptions.

Since the density of our 100x100 matrix, D = 0.07, is a bit less than the one in the second scenario, I would expect the results to look overall alike, with slightly inferior values. Thus, great variations in the Rnodes and Rranges, and an Rsys < 0.3.

2. Results

DUCROCQ Romain, M2 SIA

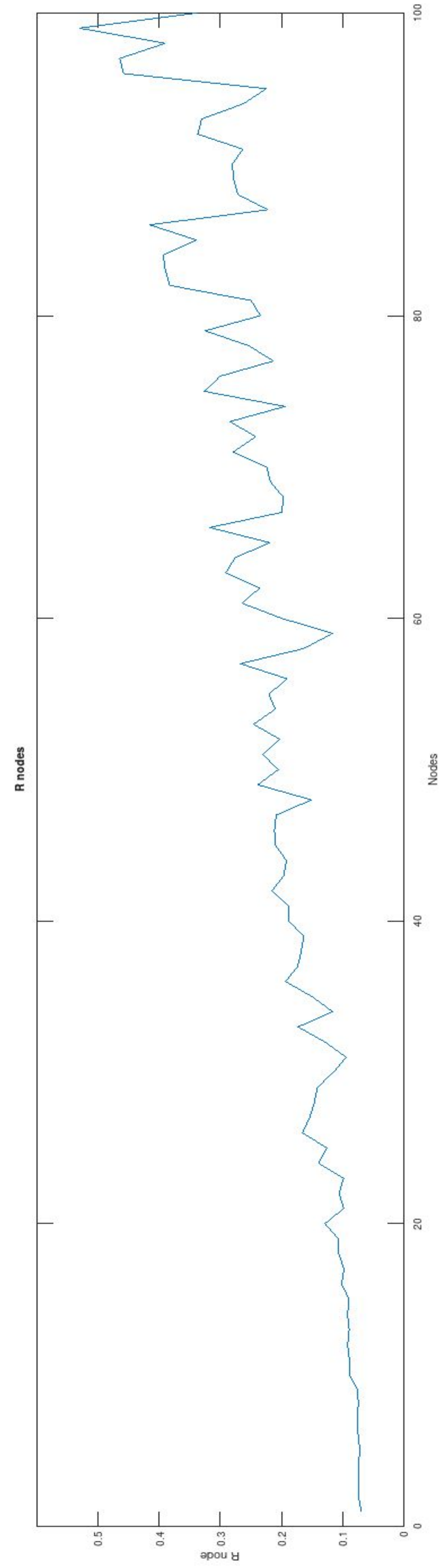
A. Rnode-Rrange pairs sorted by descending Rnode values

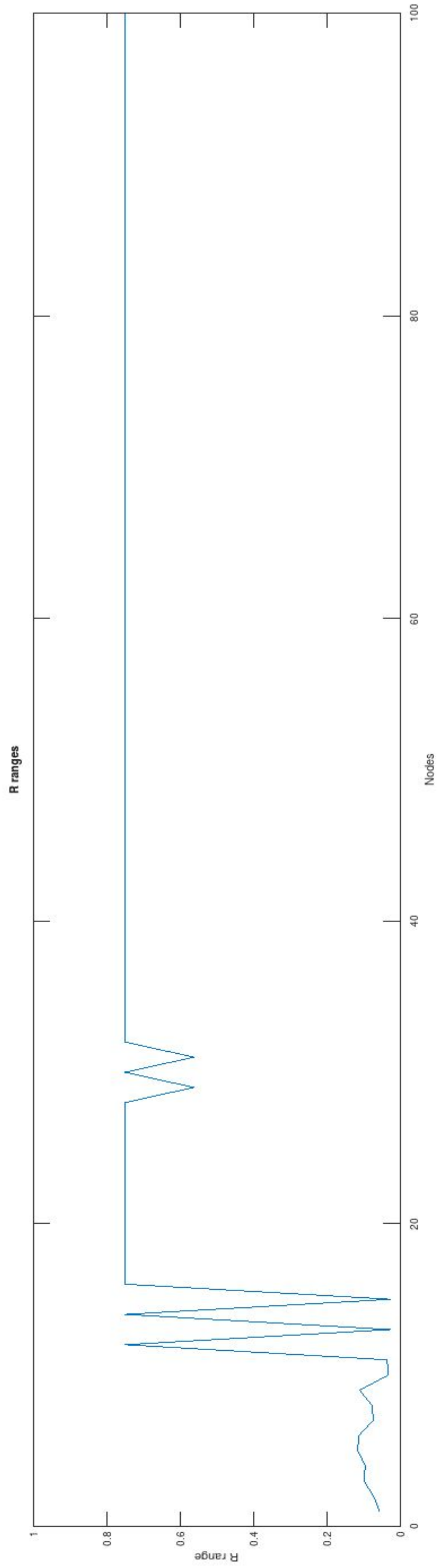
```
Node 99:       Rnode = 0.530816       Rrange = 0.749714
Node 97:       Rnode = 0.464188       Rrange = 0.749694
Node 96:       Rnode = 0.458291       Rrange = 0.749696
Node 86:       Rnode = 0.415525       Rrange = 0.749709
Node 84:       Rnode = 0.393430       Rrange = 0.749678
Node 83:       Rnode = 0.389753       Rrange = 0.749703
Node 98:       Rnode = 0.389623       Rrange = 0.749706
Node 82:       Rnode = 0.382644       Rrange = 0.749689
Node 85:       Rnode = 0.339186       Rrange = 0.749683
Node 100:      Rnode = 0.337487       Rrange = 0.749681
Node 92:       Rnode = 0.337036       Rrange = 0.749698
Node 93:       Rnode = 0.330271       Rrange = 0.749706
Node 75:       Rnode = 0.325987       Rrange = 0.749671
Node 79:       Rnode = 0.324707       Rrange = 0.749686
Node 66:       Rnode = 0.317523       Rrange = 0.749683
Node 76:       Rnode = 0.300048       Rrange = 0.749709
Node 63:       Rnode = 0.291163       Rrange = 0.749660
Node 73:       Rnode = 0.283901       Rrange = 0.749705
Node 90:       Rnode = 0.280311       Rrange = 0.749683
Node 71:       Rnode = 0.278981       Rrange = 0.749685
Node 89:       Rnode = 0.278009       Rrange = 0.749675
Node 64:       Rnode = 0.275745       Rrange = 0.749711
Node 88:       Rnode = 0.271209       Rrange = 0.749705
Node 57:       Rnode = 0.267224       Rrange = 0.749664
Node 61:       Rnode = 0.264524       Rrange = 0.749675
Node 91:       Rnode = 0.263189       Rrange = 0.749542
Node 94:       Rnode = 0.262463       Rrange = 0.749676
Node 78:       Rnode = 0.252616       Rrange = 0.749687
Node 81:       Rnode = 0.249750       Rrange = 0.749657
Node 53:       Rnode = 0.245515       Rrange = 0.749694
Node 72:       Rnode = 0.242071       Rrange = 0.749678
Node 49:       Rnode = 0.238818       Rrange = 0.749693
Node 62:       Rnode = 0.235334       Rrange = 0.749683
Node 80:       Rnode = 0.234287       Rrange = 0.749693
Node 51:       Rnode = 0.230522       Rrange = 0.749632
Node 95:       Rnode = 0.224703       Rrange = 0.749660
Node 70:       Rnode = 0.224114       Rrange = 0.749683
Node 87:       Rnode = 0.221967       Rrange = 0.749676
Node 55:       Rnode = 0.220582       Rrange = 0.749693
Node 65:       Rnode = 0.219475       Rrange = 0.749693
Node 69:       Rnode = 0.217840       Rrange = 0.749665
Node 42:       Rnode = 0.215399       Rrange = 0.749676
Node 77:       Rnode = 0.213307       Rrange = 0.749703
Node 46:       Rnode = 0.211358       Rrange = 0.749674
Node 54:       Rnode = 0.210186       Rrange = 0.749715
Node 45:       Rnode = 0.210017       Rrange = 0.749675
Node 47:       Rnode = 0.208639       Rrange = 0.749640
Node 50:       Rnode = 0.205081       Rrange = 0.749687
Node 52:       Rnode = 0.202568       Rrange = 0.749632
Node 60:       Rnode = 0.199735       Rrange = 0.749673
```

```
Node 67:        Rnode = 0.199624        Rrange = 0.749664
Node 68:        Rnode = 0.196607        Rrange = 0.749664
Node 43:        Rnode = 0.195946        Rrange = 0.749694
Node 74:        Rnode = 0.194115        Rrange = 0.749675
Node 36:        Rnode = 0.193254        Rrange = 0.749676
Node 44:        Rnode = 0.191936        Rrange = 0.749626
Node 56:        Rnode = 0.190218        Rrange = 0.749670
Node 40:        Rnode = 0.188811        Rrange = 0.749492
Node 41:        Rnode = 0.187856        Rrange = 0.749691
Node 37:        Rnode = 0.173346        Rrange = 0.749696
Node 33:        Rnode = 0.173097        Rrange = 0.749662
Node 38:        Rnode = 0.167706        Rrange = 0.749655
Node 26:        Rnode = 0.165664        Rrange = 0.749501
Node 58:        Rnode = 0.163714        Rrange = 0.749703
Node 39:        Rnode = 0.163084        Rrange = 0.749532
Node 27:        Rnode = 0.153900        Rrange = 0.749147
Node 48:        Rnode = 0.150972        Rrange = 0.749670
Node 35:        Rnode = 0.150035        Rrange = 0.749649
Node 28:        Rnode = 0.146194        Rrange = 0.749495
Node 29:        Rnode = 0.141388        Rrange = 0.562210
Node 24:        Rnode = 0.139562        Rrange = 0.749564
Node 20:        Rnode = 0.128959        Rrange = 0.749532
Node 32:        Rnode = 0.128663        Rrange = 0.749649
Node 25:        Rnode = 0.125878        Rrange = 0.749461
Node 59:        Rnode = 0.116437        Rrange = 0.748933
Node 34:        Rnode = 0.116064        Rrange = 0.749477
Node 30:        Rnode = 0.114847        Rrange = 0.749626
Node 19:        Rnode = 0.107038        Rrange = 0.749649
Node 18:        Rnode = 0.106052        Rrange = 0.749626
Node 22:        Rnode = 0.106011        Rrange = 0.749649
Node 16:        Rnode = 0.101449        Rrange = 0.749615
Node 23:        Rnode = 0.098189        Rrange = 0.749147
Node 21:        Rnode = 0.097979        Rrange = 0.748773
Node 17:        Rnode = 0.097867        Rrange = 0.749638
Node 31:        Rnode = 0.094394        Rrange = 0.561700
Node 14:        Rnode = 0.092038        Rrange = 0.749417
Node 12:        Rnode = 0.091764        Rrange = 0.749147
Node 15:        Rnode = 0.089342        Rrange = 0.024676
Node 13:        Rnode = 0.089235        Rrange = 0.026384
Node 11:        Rnode = 0.088884        Rrange = 0.035684
Node 10:        Rnode = 0.088531        Rrange = 0.031350
Node 7:         Rnode = 0.075987        Rrange = 0.072708
Node 9:         Rnode = 0.074907        Rrange = 0.110249
Node 8:         Rnode = 0.074476        Rrange = 0.077060
Node 6:         Rnode = 0.074462        Rrange = 0.112502
Node 2:         Rnode = 0.074408        Rrange = 0.072546
Node 4:         Rnode = 0.074299        Rrange = 0.094677
Node 3:         Rnode = 0.074114        Rrange = 0.098759
Node 5:         Rnode = 0.071741        Rrange = 0.115773
Node 1:         Rnode = 0.069248        Rrange = 0.055933
```

B. Rnodes and Rranges plotted by ascending node index

DUCROCQ Romain, M2 SIA





C. Rsys

```
Rsys = 0.205574
```

3. Discussion

We observe that the results are in adequacy with our hypothesis formulated from the analysis of the density of the graph. The Rnodes vary greatly from a maximum of 0.53 to a minimum of 0.07. The Rranges also have wide disparities, lying between 0.74 and 0.02. And, as expected, the Rsys is low, at around 0.21, which indicates that the network has a low tolerance to disruptions.

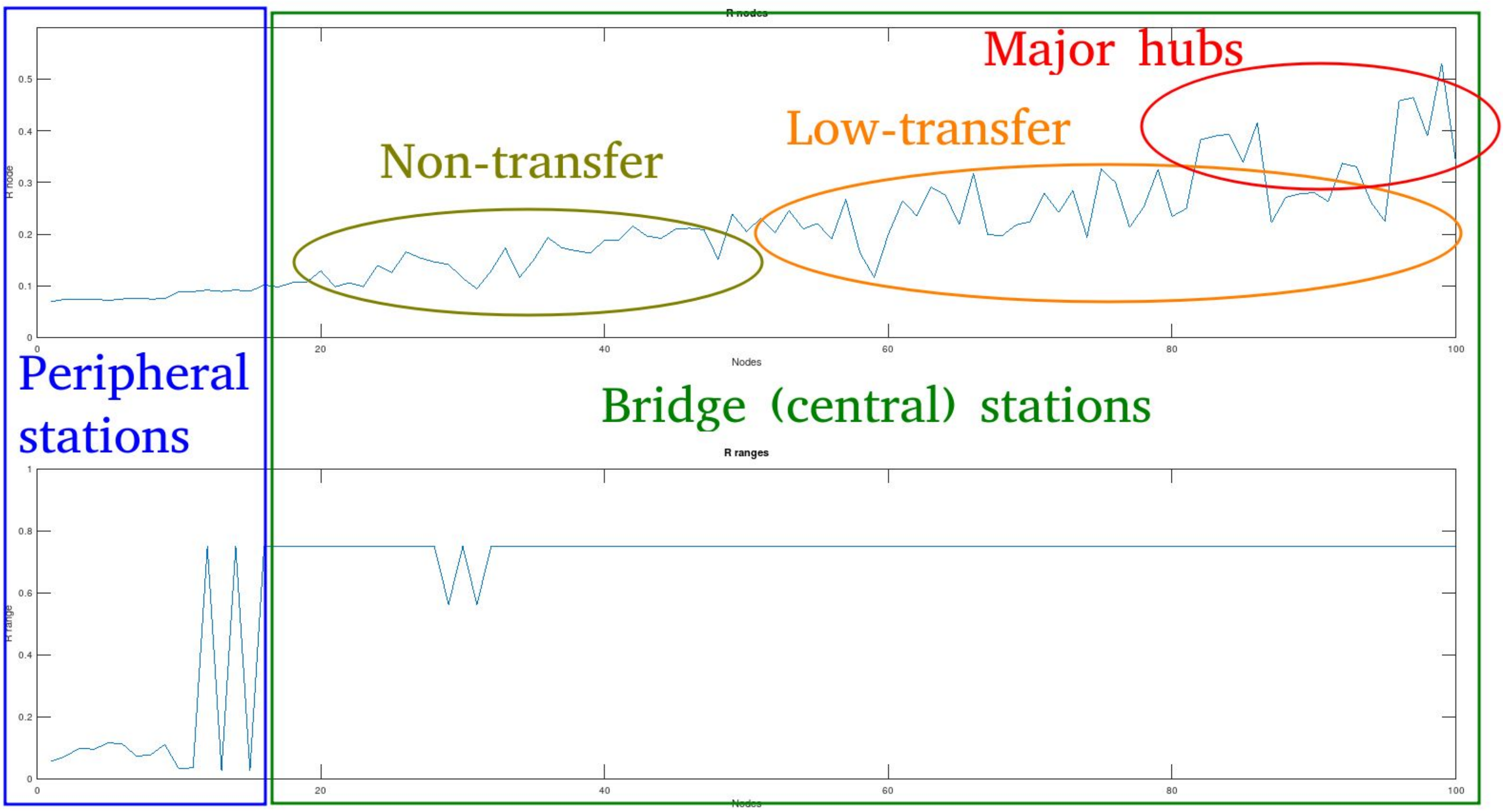
Furthermore, some patterns emerge from the results.
First, we see that the Rnodes are progressively increasing with the node index, and do so very uniformly. All the nodes with an Rnode lower than 0.1 are in the first third of the matrix (node index <= 31), and all the nodes with an Rnode greater than 0.3 are in the last third of the matrix (node index >= 66). This indicates that the matrix happens to be mainly sorted by reliability, going from the least reliable nodes at the beginning towards the most reliable ones at the end. This pattern is also clearly visible in the plot, where the curve smoothly increases with the index.
Secondly, we notice that the Rranges are high for the 87 most reliable nodes. 85 of these have a value of 0.74, while the two others are at 0.56. These overly homogeneous results contrast strongly with the scores of the 13 least reliable nodes, which have low Rranges between 0.12 and 0.02. The curve shows indeed low and disparate Rranges in the first part of the plot, and a quick convergence towards 0.74.
On behalf of these two observations and the paper, a correlation appears between Rnode and Rrange for assessing the reliability of a node. Indeed, nodes with a high Rnode will have a high Rrange, and nodes with a low Rrange will have a low Rnode. However, a low Rnode is not a consistent predictor of the value of Rrange.

If we consider our graph to be a transit system, we can interpret these results as stations in a transport network. While we only have topological information and are not interested in the length of the links nor their positions, we can geographically picture the network as a city, with the number of stations, their connections and the flow of passengers all decreasing with the distance to the center.
On one hand, the nodes with the best Rnodes are hubs and transfer stations in central areas, which offer the best connectivity. The nodes with intermediate Rnodes are mostly non-transfer stations in central areas. And nodes with low Rnodes will be more likely to be non-transfer peripheral stations on transit lines.
On the other hand, the nodes with higher Rranges are bridge stations, which serve as connections between lines and hubs, while nodes with lower Rranges are located at the ends of transit lines, far from the main connections.

We can therefore distinguish three main types of stations:

Station type	Description	Example in Paris
High Rnode & High Rrange	Major hub, transfer and bridge station in central areas.	Châtelet-Les-Halles Gare du Nord
Low Rnode & High Rrange	Non-transfer and bridge station in central areas.	Luxembourg Louvre-Rivoli
Low Rnode & Low Rrange	Non-transfer and non-bridge station in peripheral areas.	Noisy-Champs Melun

When applying this framework to our results, we understand that the 13 first nodes in the matrix, with both low Rnodes and low Rranges, are peripheral stations far from the central connections, which could be disconnected with a single cut.
The other 87 stations with high Rranges are central bridge stations. When we go through this subset of the matrix, we gain reliability and connectivity with the depth, as we have seen that the index corresponds approximately to the Rnode magnitude.
At first, at low Rnodes, we will encounter non-transfer stations, which serve solely as gateways between transfer stations. These would be disconnected with only two cuts
Then, at medium Rnodes, there are minor transfer stations, with low numbers of connections. These would require only a few cuts to be disconnected.
Finally, at the end, the high Rnodes represent the major hubs and transfer stations. These are highly connected, and a great number of cuts would be needed to disconnect them from the network.



DUCROCQ Romain, M2 SIA

Overall, the Rsys = 0.21 indicates that our graph has a low tolerance to disruption scenarios. However, this measure is not that meaningful at status quo alone, since the Rsys is just an average of all Rnodes. Hence, for homogeneous graphs, like randomly generated graphs, the Rsys would give a good approximation at any point, but it would not represent the inner disparities for heterogeneous structures.

The strength of the Rsys metric appears when introducing nodal disruptions in the system. Then, by comparing Rsys for different disruption events, we can determine the critical nodes for system reliability. In fact, the most reliable nodes are not necessarily the ones impacting the most the system in case of disruption.

Therefore, after our status quo analysis, we can assess the reliability of each individual node, have an idea of the structure of our network, and state that the system is relatively unresilient to disruptions. But we can absolutely not establish which critical nodes to protect from disruptions to maintain the level of reliability.