The Robustness of a Food Web

Assessing the robustness of a food web through different attack strategies

Alain Joss

July 17, 2023

Contents

1 Assessing Robustness **2**

1.1 The Overall Idea 2

1.2 Non-Deterministic Perturbations 2

1.3 Node Removal in Food Webs 3

1.3.1 Types of Extinctions 3

1.3.2 The Undirected Approach 3

1.3.3 The Directed Approach 3

1.4 Attack Strategies 4

1.5 Summary 5

2 Addressing Key Concerns **6**

2.1 Preliminary Steps 6

2.2 Correcting Fake Basal Species 6

2.3 Managing Generalist Species 7

2.4 Summary 7

3 The Final Pipeline **8**

1. Assessing Robustness

1.1 The Overall Idea

For assessing the robustness of a network, the idea is to perform some kind of perturbation, and compute different kinds of measures in the process, until a certain state has been reached. The high level algorithm looks as follows:

Algorithm 1 Perturbation

Require: Graph G

Initialize *stop_state* $\leftarrow false$
Initialize *robustness_states* $\leftarrow list()$
while not *stop_state* **do**
    Remove a node from G (according to a predefined attack strategy).
    Compute some metric of robustness of G.
    Save value of metric in *robustness_states*
    if some network state **then**
        stop_state $\leftarrow$ *true*
    end if
end while
Visualize the robustness trend of the perturbation.

By visualizing the trend of robustness across diverse perturbations, we can evaluate the network's resilience under a variety of metrics and attack strategies. In this context, an "attack strategy" refers to the specific probability distribution employed for node removal.

1.2 Non-Deterministic Perturbations

If at any step, a perturbation involves a non-deterministic computation, its results can be skewed by the effect of randomness. The solution is to run multiple times the same perturbation, and average the results across runs:

Algorithm 2 Multiple Perturbations

Require: Graph G, Number of perturbations N

for $i = 1$ to N **do**
    Perform perturbation i on G.
end for
Average the perturbations.
Visualize the robustness trend of the perturbations.

1.3 Node Removal in Food Webs

1.3.1 Types of Extinctions

Food web perturbations can result in two types of node removals: primary and secondary extinctions. Primary extinctions refer to a node's direct removal based on a defined attack strategy. Secondary extinctions occur when a node's elimination, directly or indirectly, results in the removal of another node. Regardless of the attack strategy employed, primary extinctions occur one by one.

1.3.2 The Undirected Approach

Considering the undirected version of a food web, calculating secondary extinctions involves assessing if a node becomes isolated after each primary extinction. Isolation can result from either primary or secondary extinctions. The algorithm for the undirected approach is as follows:

Algorithm 3 Undirected Removal

```
Require: Graph G, Node n
Remove node n from G
Initialize continue_flag ← true
while continue do
    continue_flag ← false
    for each Node u in G do
        if u.degree == 0 then
            continue_flag ← true
            Remove node u from G
        end if
    end for
end while
```

The *continue_flag* ensures we only check for isolated nodes as long as necessary, as the removal of one node can potentially leave others isolated.

1.3.3 The Directed Approach

Given that food webs are typically directed networks, a more appropriate node-removal algorithm would consider the direction of prey-predator relationships. This means when a node is removed, all species feeding on it also go extinct if they have no other food sources. Calculating secondary extinctions here involves checking if a node still receives energy (has incoming edges). The algorithm for this directed approach is as follows:

Algorithm 4 Directed Node Removal

```
Require: Directed Graph G, Node n
k_level_neighbors ← successors(n) in G
Remove node n from G
while k_level_neighbors ≠ ∅ do
    Initialize empty sets new_level_neighbors, removed_neighbors
    for each Node neighbor in k_level_neighbors do
        change_flag ← false
        if G.in_degree(neighbor) == 0 or
          G.degree(neighbor) == 0 or
          G.in_degree(neighbor) == 1 and
          G.has_edge(neighbor, neighbor) then
            Add successors(neighbor) to new_level_neighbors
            Add neighbor to removed_neighbors
            Remove neighbor from G
            change_flag ← true
        end if
        if not change_flag then
            Break
        end if
    end for
    k_level_neighbors ← new_level_neighbors \ removed_neighbors
end while
```

Node removal occurs when:

- "G.in degree(neighbor) == 0": A node has no food source.
- "G.degree(neighbor) == 0": A node becomes completely isolated, including plants, which we assume need pollinators.
- "G.in degree(neighbor) == 1 and G.has edge(neighbor, neighbor)": A cannibalistic node is not receiving energy from the network.

1.4 Attack Strategies

Attack strategies dictate the rules governing the selection of nodes (species) for primary extinction. These nodes are then processed by the node removal algorithm. The implemented strategies include:

- Random Attack: Applies an equal probability for removal to all nodes.
- Centrality Attack: Prioritizes node removal based on a chosen centrality measure, proceeding in descending order. This strategy produces deterministic results.

- Habitat Attack: Nodes associated with a specific habitat have an increased removal probability. The strategy also considers the number of different habitats a node inhabits.
- Threatened Species Attack: Nodes representing species on a threatened species list are prioritized for removal. Within this list, the removal order is random.

1.5 Summary

Putting it all together, the robustness assessment of the food web looks as follows:

Algorithm 5 Robustness Assessment

```
Require: Graph G, Number of perturbations N, Attack Strategy S
Initialize all_robustness_states ← list()
for i = 1 to N do
    Initialize stop_state ← false
    Initialize robustness_states ← list()
    while not stop_state do
        Choose a node according to attack strategy S and remove from G.
        Apply "Directed Node Removal" algorithm.
        Compute some metric of robustness of G.
        Save value of metric in robustness_states
        if some network state then
            stop_state ← true
        end if
    end while
    Add robustness_states to all_robustness_states
end for
Compute the average robustness trend from all_robustness_states.
Visualize the averaged robustness trend of the perturbations.
```

2. Addressing Key Concerns

2.1 Preliminary Steps

In preparation for the network's perturbation analysis, we need to address two concerns as follows:

1. For nodes with an in-degree of zero, that are not basal nodes, we construct randomized inward-links.
2. As the network under consideration is a meta web, rather than a real food web, we eliminate k% of inward links of generalist species.

2.2 Correcting Fake Basal Species

This step rectifies species misrepresented as basal due to data gaps. We create randomized inward-links for these species, considering their diet, habitat, and zone attributes.

Algorithm 6 Randomized Inward-Links Construction

Require: Dataset "species_for_randomized_link_assignment", Dataset "all_species_and_feeding_groups"

for each row in "species_for_randomized_link_assignment" **do**
  Extract the species, its diet, diet rank, habitat, and zone.
  Check the second dataset "all_species_and_feeding_groups" to find species that share the same habitat and diet rank.
  Link the fake basal species to these species.
  Remove duplicate links.
  Retain a subset of potential links.
  Add these links to the overall network graph.
end for

Depending on the species' dietary breadth, we retain different numbers of potential links:

- For Generalized-diet species, we retain 5% of potential interaction links.
- For Specialized-diet species, we randomly select between 1 and 5 potential interaction links.

2.3 Managing Generalist Species

Here, we eliminate p% of inward links for generalist species, guided by the Optimal Foraging Theory (OFT). Generalist species are defined as those with a degree threshold of k. The algorithm is as follows:

Algorithm 7 Prune Network on Degree Threshold

Require: Network G, Degree threshold k, Percentage to remove p

```
nodes ← List of nodes in G
for each node in nodes do
    inDegree ← in-degree of node in G
    if inDegree ≥ k then                                  ▷ The node is a generalist
        inwardEdges ← List of in-edges of node in G
        numEdgesToRemove ← inDegree × p
        edgesToRemove ← randomly sample numEdgesToRemove from inwardEdges
        Remove edgesToRemove from G
    end if
end for
```

Ensure: Pruned network G

In this context, a node's in-degree refers to the number of incoming edges (number of predators). If a node's in-degree equals or exceeds a threshold k, it's deemed a generalist. For each generalist, a proportion p of its inward edges is randomly selected for removal, effectively pruning the network.

2.4 Summary

To summarize, the following steps prepare the network for its robustness assessment:

Algorithm 8 Network Preparation

Require: Graph G, Datasets $D1$, $D2$

```
Apply "Randomized Inward-Links Construction" using D1 and D2 to G.
Apply "Pruning Based on OFT" to G.
```

3. The Final Pipeline

- importing the dataset
- creating new links
- removing links
- create attack strategies S
- perform perturbations for $|S|$ strategies
- visualize all the different perturbation results