Supplementary information for 'maxnodf: an R package for fair and fast comparisons of nestedness between networks'

Christoph Hoeppke[1,2,3] and Benno I. Simmons[1,4]

1 Conservation Science Group, Department of Zoology, University of Cambridge, The David Attenborough Building, Pembroke Street, Cambridge, CB2 3QZ, UK

2 Faculty of Mathematics, Wilberforce Road, Cambridge CB3 0WA, UK

3 Mathematical Institute, Andrew Wiles Building, Woodstock Road, Oxford, OX2 6GG, UK

4 Centre for Ecology and Conservation, College of Life and Environmental Sciences, University of Exeter, Cornwall Campus, Penryn, UK

Corresponding authors:

Benno I. Simmons. Address: Centre for Ecology and Conservation, College of Life and Environmental Sciences, University of Exeter, Cornwall Campus, Penryn, UK. Email: benno.simmons@gmail.com

Christoph Hoeppke. Address: Mathematical Institute, Woodstock Road, Oxford, OX2 6GG Email: christoph.hoeppke@maths.ox.ac.uk

Contents

1 Supplementary figures

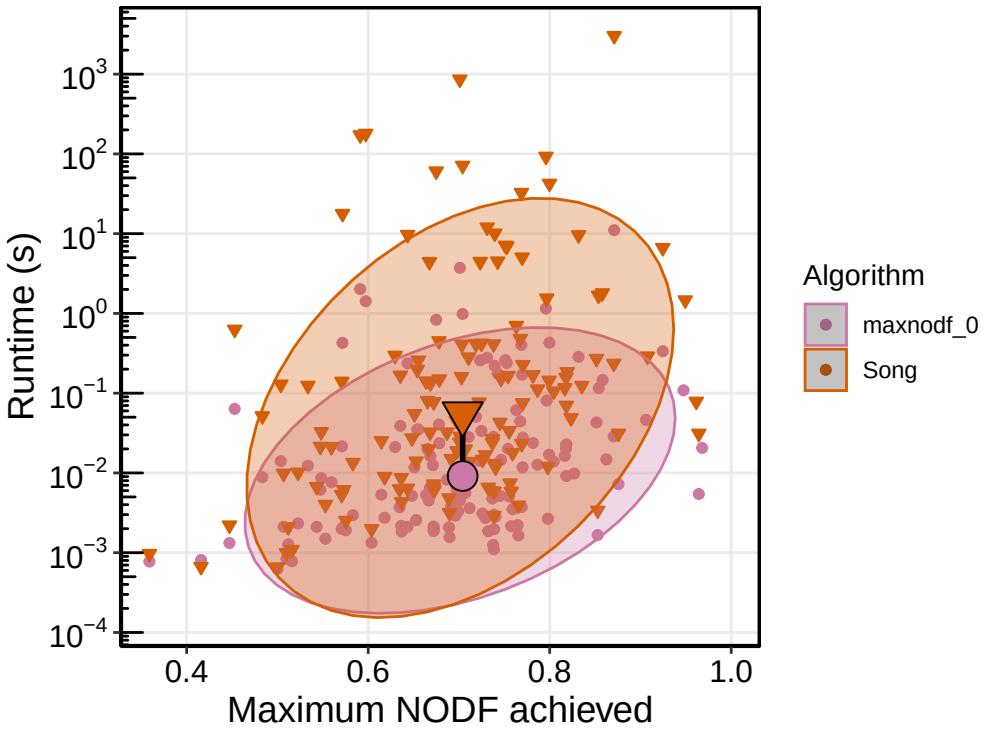

Figure S1: Comparison of the time taken to run, and maximum NODF achieved, for the greedy algorithm in the maxnodf package (quality 0) and the original algorithm proposed by Song et al. (2017). All algorithms were run on an identical set of 135 empirical pollination networks from the Web of Life dataset. Small points represent individual networks; large points represent medians. Ellipses represent 95% confidence intervals.

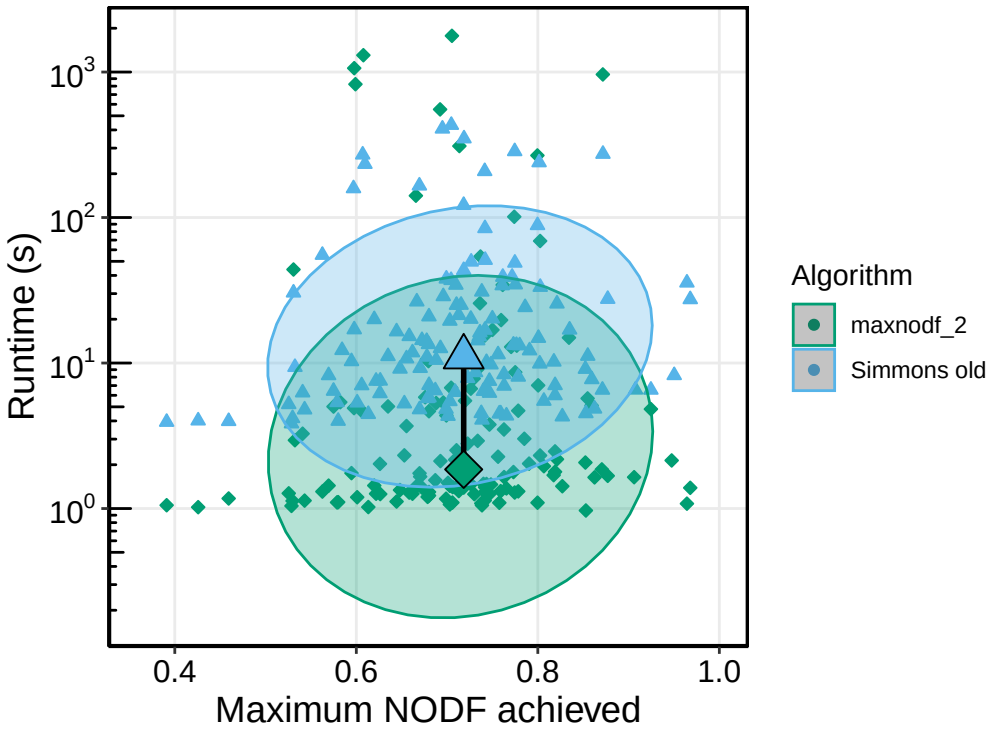

Figure S2: Comparison of the time taken to run, and maximum NODF achieved, for the simulated annealing algorithm in the maxnodf package (quality 2) and the simulated annealing algorithm proposed by Simmons et al. (2019). All algorithms were run on an identical set of 135 empirical pollination networks from the Web of Life dataset. Small points represent individual networks; large points represent medians. Ellipses represent 95% confidence intervals.

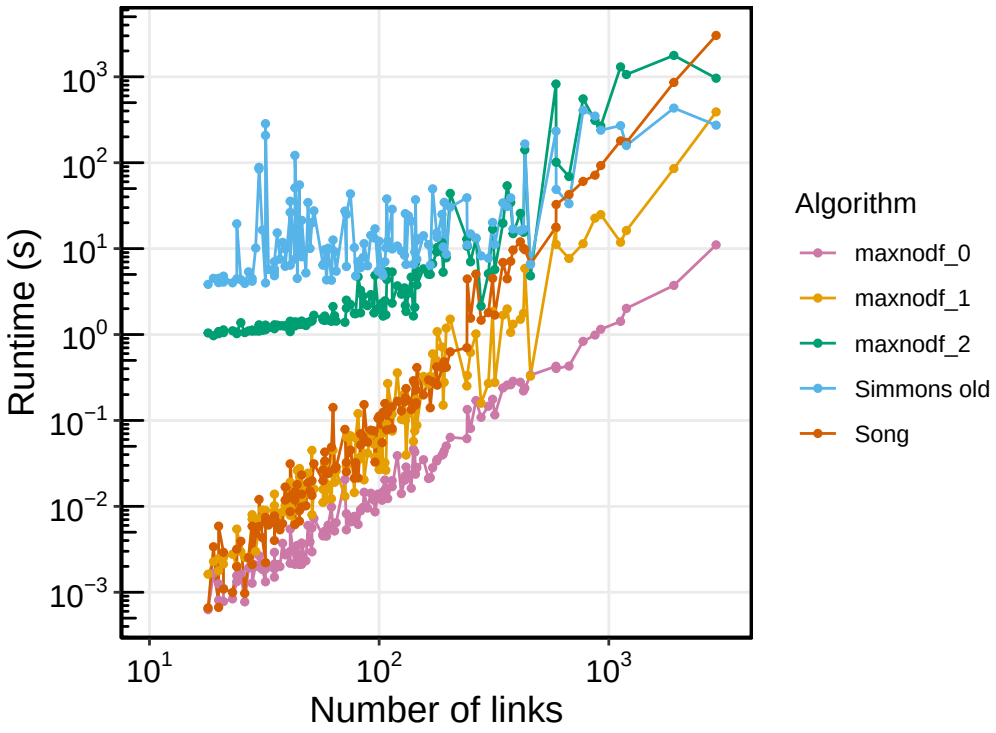

Figure S3: Comparison of how long all the algorithms take to run on networks with different numbers of links. Network data were the same 135 networks as in Figure 1 and Figure S1 and Figure S2. Note that both axes are on a log scale.

2 Appendix S1

Throughout the manuscript we made the statement, that the extended NODF-maximisation problem is NP-hard. We proof this by first reducing the NODF-maximisation problem to an equivalent *mixed integer problem* (MIP). The class of mixed integer problems NODF-maximisation falls into is generally hard. We provide a proof for this by reducing the NP-complete *vertex cover* problem to the generalised NODF-maximisation problem.

Theorem 1 (NODF maximisation problem properties)**.** *Given three integer $N, M, L \in \mathbb{N}_{\geq 1}$, with $N, M > 1$ and $N + M + 1 \leq L \leq N \times M$, the* NODF *maximisation problem can be written as a MIP. The resulting generalised* NODF*-maximisation problem is NP-hard.*

Proof of Theorem 1. To formalise the NODF maximisation problem we first consider the set of bi-adjacency matrices we are optimising over. It is sufficient to consider these bi-adjacency matrices, as each undirected and unweighted bipartite network is uniquely described its bi-adjacency matrix. For a given bipartite graph $G = (V_1, V_2, E)$, where $E \subset \{(v1, v2)|v_1 \in V_1, v_2 \in V_2\}$ is the set of edges in G and V_1, V_2 are vertex sets, the bi-adjacency $A(G)$ is given by

$$A(G) \in \{0,1\}^{|V_1|\times|V_2\|} \tag{1}$$

$$a_{i,j} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

To be consider in the NODF maximisation process a bi-adjacency matrix A has needs to

1. be a binary matrix of the correct shape $A \in \{0,1\}^{N\times M}$,
2. have the correct number of total links $\sum_{i=1}^{N}\sum_{j=1}^{M} A_{i,j} = L$,
3. have no empty rows $\sum_{i=1}^{N} A_{i,k} \geq 1 \ \forall k = 1, \ldots, M$
4. have no empty columns $\sum_{j=1}^{M} A_{k,j} \geq 1 \ \forall k = 1, \ldots, N$.

For a given matrix $A \in \{0,1\}^{N\times M}$ we denote the rows and columns by the column vectors $r_1, r_2, \ldots, r_N$ and $c_1, c_2, \ldots, c_M$ respectively such that

$$A = \begin{bmatrix} r_1^T \\ r_2^T \\ \vdots \\ r_N^T \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \ldots & c_M \end{bmatrix}. \tag{3}$$

We can thus write the NODF maximisation problem as

$$\max_{A\in\{0,1\}^{N\times M}} \quad \frac{2}{N(N-1)+M(M-1)} \left(\sum_{i,j=1}^{N} [r_i^T\mathbf{1} > r_j^T\mathbf{1}] \frac{r_i^T r_j}{r_i^T r_i} + \sum_{i,j=1}^{M} [c_i^T\mathbf{1} > c_j^T\mathbf{1}] \frac{c_i^T c_j}{c_i^T c_i} \right) \tag{4a}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} A_{i,j} = L, \tag{4b}$$

$$\sum_{i=1}^{N} A_{i,k} \geq 1 \ \forall k = 1, \ldots, M, \tag{4c}$$

$$\sum_{j=1}^{M} A_{k,j} \geq 1 \ \forall k = 1, \ldots, N, \tag{4d}$$

where $\mathbf{1}$ is the ones vector, such that for all vectors $x^T\mathbf{1} = \sum_{i=1}^{d} x_i \ \forall x \in \mathbf{R}^d$. In the following we will establish that the NODF maximisation problem (4) can be equivalently formulated as a linear

mixed integer program. Linear mixed integer programs contain both continuous variables $x \in [a, b] \subset \mathbb{R}$ and discrete variables $y \in \Omega \subset \mathbb{N}$, however all optimisation variables are only involved in the constraints or in the objective function linearly. The NODF maximisation problem as written in (4), while only having linear constraints, contains the nonlinear expression $(r_i^T r_i)^{-1}(r_i^T r_j)$. We commence by introducing binary variables to account for the decreasing fill conditions $[r_i^T \mathbf{1} > r_j^T \mathbf{1}]$ and, will evaluate to 1 if, and only if the i-th row has higher fill as the j-th row. Equivalently we treat the conditions $[c_i^T \mathbf{1} > c_j^T \mathbf{1}]$ for column fills. Let $\delta^r_{i,j}, \delta^c_{i,j} \in \{0, 1\}$ be such that

$$\delta^r_{i,j} = \begin{cases} 1, & \text{if } r_i^T \mathbf{1} > r_j^T \mathbf{1}, \\ 0, & \text{otherwise,} \end{cases}, \qquad \delta^c_{i,j} = \begin{cases} 1, & \text{if } c_i^T \mathbf{1} > c_j^T \mathbf{1}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

We note that the expression

$$M + r_i^T \mathbf{1} - r_j \mathbf{1} \geq 0 \tag{6}$$

satisfies

$$M + r_i^T \mathbf{1} - r_j \mathbf{1} \geq M + 1 \text{ if } r_i^T \mathbf{1} > r_j \mathbf{1}, \tag{7}$$

$$M + r_i^T \mathbf{1} - r_j \mathbf{1} \leq M \text{ otherwise.} \tag{8}$$

As a consequence of $\delta^r_{i,j}, \delta^c_{i,j} \in \{0, 1\}$, we can enforce the conditions (5) with the linear constraints

$$\delta^r_{i,j} \leq \frac{1}{M+1} \left(M + r_i^T \mathbf{1} - r_j^t \mathbf{1} \right) \ \forall i, j = 1, \ldots, N, \tag{9}$$

$$\delta^c_{i,j} \leq \frac{1}{N+1} \left(N + c_i^T \mathbf{1} - c_j^T \mathbf{1} \right) \ \forall i, j = 1, \ldots, M. \tag{10}$$

Next we introduce continuous variables $w^r_{i,j}, w^c_{i,j} \in [0, 1]$ to represent the expressions

$$w^r_{i,j} = \frac{r_i^T r_j}{r_i^T r_i}, \qquad w^c_{i,j} = \frac{c_i^T c_j}{c_i^T c_i}. \tag{11}$$

In the following we will primarily focus on the derivation of the conditions for $w^r_{i,j}$, as the conditions for $w^c_{i,j}$ follow with analogous reasoning. Let us first consider the subexpression

$$r_i^T r_j = \sum_{l=1}^{M} A_{i,l} A_{j,l}, \tag{12}$$

which is quadratic in the entries of the underlying bi-adjacency matrix $A \in \{0, 1\}^{N \times M}$. The fact that we are restricting ourselves to considering only binary variables in A allows us to reformulate this quadratic expression using binary, linearly constraint variables. We introduce $q^r_{i,j;l}, q^c_{i,j;l} \in \{0, 1\}$ with

$$q^r_{i,j;l} \leq A_{i,l}, \tag{13a}$$

$$q^r_{i,j;l} \leq A_{j,l}, \tag{13b}$$

$$q^c_{i,j;l} \leq A_{l,i}, \tag{13c}$$

$$q^c_{i,j;l} \leq A_{l,j}. \tag{13d}$$

$$\tag{14}$$

In the optimisation problem $q^r_{i,j;l}, q^c_{i,j;l}$ will only be multiplied with non-negative variables, consequently we may assume that $q^r_{i,j;l}, q^c_{i,j;l}$ always attain there maximal value at the global optimum. Consequently the conditions introduced in (13), given that A is a binary matrix, ensure that

$$q^r_{i,j;l} = A_{i,l} A_{j,l}, \qquad q^c_{i,j;l} = A_{l,i} A_{l,j}. \tag{15}$$

This lets us write the expression for w as

$$w^r_{i,j} = \frac{1}{r_i^T r_i} \sum_{l=1}^{M} q^r_{i,j;l}, \tag{16}$$

$$w^c_{i,j} = \frac{1}{c_i^T c_i} \sum_{l=1}^{N} q^c_{i,j;l}, \tag{17}$$

which is linear in the q-variables. The remaining non-linearity in equations (16) and (17) is eliminated by introducing the variables binary variables $z^r_k \in \{0, 1\}$, and $z^c_k \in \{0, 1\}$, with

$$z^r_{i;k} \leq 1 + \frac{k-1}{M} - \frac{1}{M} \sum_{l=1}^{M} A_{i,l}, \tag{18}$$

$$z^c_{i;k} \leq 1 + \frac{k-1}{N} - \frac{1}{N} \sum_{l=1}^{N} A_{l,i}. \tag{19}$$

From plot S4 of these constraints on z we can see that they ensure that

$$z^r_{i;k} = \begin{cases} 1, & \text{if } \sum_{l=1}^{M} A_{i,l} < k, \\ 0, & \text{otherwise,} \end{cases}, \text{ and } \qquad z^c_{i;k} = \begin{cases} 1, & \text{if } \sum_{l=1}^{N} A_{l,i} < k, \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

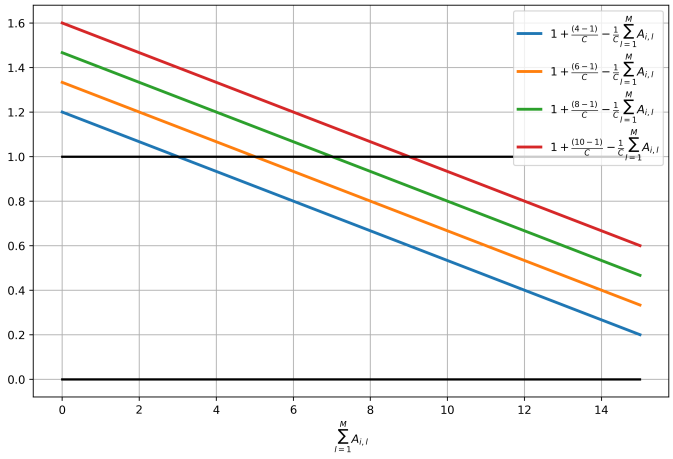

Figure S4: Plot of the constraints on z-variables given in equations (18) and (19).

Using the binary z-variables we employ the so called big-M trick, as shown in Parker & Rardin (2014), we can implement constraints that are active only if the corresponding z-variable is 0 or `false`.

The constraints

$$w^r_{i,j} - \frac{1}{k} \sum_{l=1}^{M} q^r_{i,j;l} \leq z^r_k, \tag{21}$$

$$w^c_{i,j} - \frac{1}{k} \sum_{l=1}^{N} q^c_{i,j;l} \leq z^c_k \tag{22}$$

enforce the conditions (16) and (17) as progressively more restrictive conditions are enforced as z-variables are `true` for higher values of k. We note at this point that the constraints (21) and (22) are linear in all programming variables. As a last step we avoid the multiplication of w and δ-variables by including the constraints

$$w^r_{i,j} \leq \delta^r_{i,j}, \tag{23}$$
$$w^c_{i,j} \leq \delta^c_{i,j}. \tag{24}$$

To shorten the notation for writing the complete linear mixed integer programming problem for NODF maximisation, we introduce the shorthand notation $(\delta, \mathbf{q}, \mathbf{w}) \in K$, where K is the feasible set of all auxiliary variables

$$\delta^r_{i,j} \in \{0,1\}\ \forall i,j, \quad q^r_{i,j;l} \in \{0,1\}\ \forall i,j,l, \quad w^r_{i,j} \in [0,1]\ \forall i,j, \tag{25}$$
$$\delta^c_{i,j} \in \{0,1\}\ \forall i,j, \quad q^c_{i,j;l} \in \{0,1\}\ \forall i,j,l, \quad w^c_{i,j} \in [0,1]\ \forall i,j. \tag{26}$$

The linear mixed integer program form of the NODF maximisation problem is therefore given as

$$\max_{A\in\{0,1\}^{N\times M},(\mathbf{delta},\mathbf{q},\mathbf{w})\in\mathbf{K}} \quad \frac{2}{N(N-1)+M(M-1)}\left(\sum_{i,j=1}^{N} w^r_{i,j}+\sum_{i,j=1}^{M} w^c_{i,j}\right) \tag{27a}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} A_{i,j} \le -L, \tag{27b}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M} -A_{i,j} \le -L, \tag{27c}$$

$$\sum_{i=1}^{N} A_{i,k} \ge 1 \ \forall k = 1,\dots,M, \tag{27d}$$

$$\sum_{j=1}^{M} A_{k,j} \ge 1 \ \forall k = 1,\dots,N, \tag{27e}$$

$$\delta^r_{i,j} \le \frac{1}{M+1}\left(M + r_i^T\mathbf{1} - r_j^t\mathbf{1}\right) \ \forall i,j = 1,\dots,N, \tag{27f}$$

$$\delta^c_{i,j} \le \frac{1}{N+1}\left(N + c_i^T\mathbf{1} - c_j^T\mathbf{1}\right) \ \forall i,j = 1,\dots,M, \tag{27g}$$

$$q^r_{i,j;l} \le A_{i,l}, \tag{27h}$$

$$q^r_{i,j;l} \le A_{j,l}, \tag{27i}$$

$$q^c_{i,j;l} \le A_{l,i}, \tag{27j}$$

$$q^c_{i,j;l} \le A_{l,j} \tag{27k}$$

$$z^r_{i;k} \le 1 + \frac{k-1}{M} - \frac{1}{M}\sum_{l=1}^{M} A_{i,l}, \tag{27l}$$

$$z^c_{i;k} \le 1 + \frac{k-1}{N} - \frac{1}{N}\sum_{l=1}^{N} A_{l,i}, \tag{27m}$$

$$w^r_{i,j} - \frac{1}{k}\sum_{l=1}^{M} q^r_{i,j;l} \le z^r_k, \tag{27n}$$

$$w^c_{i,j} - \frac{1}{k}\sum_{l=1}^{N} q^c_{i,j;l} \le z^c_k, \tag{27o}$$

$$w^r_{i,j} \le \delta^r_{i,j}, \tag{27p}$$

$$w^c_{i,j} \le \delta^c_{i,j}. \tag{27q}$$

For an appropriately chosen matrix B and vectors v_x, x_y and b problem (27) can be written in the more general form

$$\max_{x\in\{0,1\}^d, y\in[0,1]^n} \quad x^T v_x + y^T v_y \tag{28a}$$

$$\text{subject to} \quad B\begin{bmatrix} x \\ y \end{bmatrix} \le b. \tag{28b}$$

We prove NP-hardness for this problem by following the well known reduction of the NP-complete minimum vertex cover problem to the above problem. Given a graph $G = (V, E)$, where V is a finite list of graph vertices and $E \subset V \times V$ is a finite set of edges, the minimum vertex cover problem is to find the smallest vertex set $V' \subset V$, so that for every $(u, v) \in E$ at least one of the statements $u \in V'$ or $v \in V'$ is true. The minimum vertex cover was proven to be NP-hard in

Karp (1972). Then NP-hardness of the generalised NODF-maximisation problem follows by the linear time reduction of the minimum vertex cover problem to

$$-\max_{x\in\{0,1\}^V} \quad -x^T\mathbf{1} \tag{29a}$$

$$\text{subject to} \quad x_u + x_y \geq 1 \forall (u,v) \in E. \tag{29b}$$

■

3 Appendix S2

Improving NODF calculation efficiency

The NODF_c metric relies on evaluating $\text{NODF}_n = \frac{\text{NODF}}{max(\text{NODF})}$, where $max(\text{NODF})$ denotes the maximum nestedness that can be obtained by networks in the same class as the original network. We say that two network are in the same class if they share the same number of species in both classes and the total number of links in the network are identical.

We can compute the nestedness of a single network with reasonable efficiency, however approximating $max(\text{NODF})$ requires the application of optimisation algorithms. During the course of such algorithms, it is often the case that NODF is reevaluated repeatedly. The speed at which we can reevaluate NODF thus becomes the determining factor for the performance of optimisation algorithms.

To assess the computational cost of NODF, we revisit the definition provided in Almeida-Neto et al. (2008) and provide an algorithm for its evaluation. Note that other definitions of nestedness exist, with different advantages and disadvantages. We choose to focus on the original definition of NODF here, as it is the most widely used and has been extensively tested by Song et al. (2017), but developing and testing methods for normalising other definitions of nestedness (e.g. Fortuna et al. 2019) is an important area for future work. The computation of NODF relies on evaluating, for all row and column pairs, the paired nestedness $N^{r,i,j}_{\text{paired}}$, $N^{c,i,j}_{\text{paired}}$. We consider the rows $r_1, r_2 \in \mathbb{N}_{\geq 1}$. If $r_2 \geq r_1$ we set $N^{r,r_1,r_2} = 0$. Next we check that the number of links from row r_1 to row r_2 is decreasing. If the number of links is non decreasing, we set again $N^{r,r_1,r_2} = 0$, otherwise we compute the partial overlap between rows r_1 and r_2. The partial overlap is defined as the proportion of links in row r_2, which have matching links in row r_1. The paired nestedness for columns is defined analogous. For a network $A \in \{0,1\}^{N \times M}$ with N rows and M columns, computing the partial overlap between two rows or columns requires $2M$ or $2N$ operations respectively. During the computation of NODF we need to compute N^2 paired nestedness values for rows and M^2 paired nestedness values for columns. This results in a computational cost of $\mathcal{O}(N^2 \times M + M^2 \times N)$ for one NODF evaluation. This cost can also be seen in Algorithm 1, as it results from summation inside of nested for loops. As a result, evaluating NODF for a network of twice the size $\tilde{A} \in \{0,1\}^{2N \times 2M}$ requires 8 times as many operations as evaluating NODF for the original network A. We thus say that the computational cost of NODF grows cubic with network size.

Algorithm 1 NODF evaluation

procedure NODF($A \in \{0,1\}^{N,M}$) ▷ Computes the $NODF$ value of a network A
  $\text{NODF}_r \leftarrow 0$
  $\text{NODF}_c \leftarrow 0$
  for $r_1 \in \{1, \ldots, N\}$ **do** ▷ Compute the row contribution to NODF
    for $r_2 \in \{r_1 + 1, \ldots, N\}$ **do**
      $\alpha_1 \leftarrow \sum_{j=1}^{M} A(r_1, j)$
      $\alpha_2 \leftarrow \sum_{j=1}^{M} A(r_2, j)$
      if $\alpha_1 > \alpha_2$ **then**
        $\beta \leftarrow \sum_{j=1}^{M} A(r_1, j)A(r_2, j)$
        $\text{NODF}_r \leftarrow \text{NODF}_r + \frac{\beta}{\alpha_2}$
      end if
    end for
  end for
  for $c_1 \in \{1, \ldots, M\}$ **do** ▷ Compute the column contribution to NODF
    for $c_2 \in \{c_1 + 1, \ldots, M\}$ **do**
      $\alpha_1 \leftarrow \sum_{j=1}^{N} A(j, c_1)$
      $\alpha_2 \leftarrow \sum_{j=1}^{N} A(j, c_2)$
      if $\alpha_1 > \alpha_2$ **then**
        $\beta \leftarrow \sum_{j=1}^{M} A(j, c_1)A(j, c_2)$
        $\text{NODF}_c \leftarrow \text{NODF}_c + \frac{1}{\alpha_2}$
      end if
    end for
  end for
  $\text{NODF} \leftarrow 2\frac{\text{NODF}_r + \text{NODF}_c}{M(M-1)+N(N-1)}$
end procedure

During optimisation processes we often perform minor modifications, which only alter the network in a single position. We can use the knowledge that all but one entries in the network remain unchanged to accelerate the computation of NODF. Consider a modification of network $A \in \{0,1\}^{N \times M}$ in position (i, j). We only need to consider the N values $A(1, j), \ldots A(N, j)$ to update NODF_r and the M values $A(i, 1), \ldots, A(i, M)$ to update NODF_c. This results in a total complexity reduction from $\mathcal{O}(N^2 \times M + N \times M^2)$ to $\mathcal{O}(N + M)$. We can thus recompute the NODF value for a network which has been modified in a single position in linear rather than cubic time. These efficiency improvements enable the use of more advanced algorithms like, like hill climbing or simulated annealing, to the NODF maximisation problem.

Package functions

The *R* package `maxnodf` contains methods for maximising the NODF metric for ecological networks, subject to the constraint that all species have at least one link. Different applications will require a different trade-off between optimisation quality and performance. To address these different preferences we offer three different quality settings in the `maxnodf` function: 0, 1 and 2, where 0 is the lowest quality and 2 is the highest quality. These three quality settings build upon each other, guaranteeing that the NODF values obtained on a higher quality setting will never be below those obtained at lower quality. The medium quality setting (quality 1) first runs the low quality algorithm (quality 0) before applying further optimisation. Similarly, the high quality setting (quality 2) first runs the medium quality algorithm (quality 1), before applying further optimisation. As each algorithm takes, as its starting point, the solution of the next-lowest quality algorithm below it, higher quality algorithms can never have worse optima than lower-quality algorithms. We note that networks with N rows M columns and a total of L links, where $L \leq N + M$

are necessarily compartmentalised (contain multiple disconnected subnetworks). In such cases nestedness computations should be performed on individual compartments of the network. In these cases `maxnodf` displays a warning message, and the maximisation is instead performed on the set of networks with $L = N + M + 1$ links.

Quality 0: Greedy algorithm

Calling `maxnodf` with a quality parameter of 0 invokes a greedy optimiser, which is the least accurate but fastest optimiser offered in `maxnodf`. We first construct a network $A^{(0)} \in \{0,1\}^{N\times M}$ with $A^{(0)}(1,k) = A^{(0)}(j,1) = A^{(0)}(2,2) = 1\ \forall j = 1,\dots,N, k = 1,\dots,M$. For the case $N = M = 5$ the initial network is given by

$$A^{(0)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{30}$$

Note that by filling the first row and first column of the matrix, our constraint that all species must have at least one link is satisfied in the initial network. We denote the network and its component at step k by $A^{(k)}$ and $a_{i,j}^{(k)}$ respectively. In each step we consider potential new link positions, which border links in the current network $\{(i,j)|a^{(k-1)}(i,j) = 0, a^{(k-1)}(i-1,j) = a^{(k-1)}(i,j-1) = 1\}$ (at no point are links in the first row or column modified, ensuring that all species always have at least one link). We evaluate NODF for each of the potential modifications of $A^{(k-1)}$ and introduce a new link at the NODF-maximising index pair (i^*, j^*). We successively add links to $A^{(k)}$ until the desired number of links is reached. An algorithm description of this procedure is given below:

Algorithm 2 Greedy optimisation

procedure MAXNODF(N, M, L, quality=0) ▷ Greedy maximisation
  $A \leftarrow \mathbf{0} \in \{0,1\}^{N\times M}$
  $A(i,1) \leftarrow 1\ \forall i = 1,\dots N$
  $A(1,j) \leftarrow 1\ \forall j = 1,\dots M$
  $A(2,2) \leftarrow 1$
  for $k \leftarrow N + M + 1,\dots,L$ **do**
    $w \leftarrow -\infty$
    $(i^*, j^*) \leftarrow (-1,-1)$
    for all $(i,j) \in \{(i,j) \in \mathbb{N}\times\mathbb{N} | 1 \le i \le N, 1 \le j \le M, A(i,j) = 0\}$ **do**
      $A(i,j) \leftarrow 1$
      if NODF(A) $> w$ **then**
        $w \leftarrow$ NODF(A)
        $(i*, j^*) \leftarrow (i,j)$
      end if
      $A(i,j) \leftarrow 0$
    end for
    $A(i^*, j^*) \leftarrow 1$
  end for
  return A
end procedure

Quality 1: Greedy algorithm with hill climbing

Calling `maxnodf` with the quality parameter 1 first invokes the same greedy algorithm as quality 0 does. Starting further optimisation from this network ensures that quality 1 will never perform worse than quality 0.

Improvements over the greedy algorithm can be obtained with an algorithm called hill climbing. In hill climbing we perform local modifications to a network by moving a link from its original position to a neighboring position. Algorithm (3) describes how neighbour positions are computed: the neighbours of a focal link are the empty (0) cells immediately above, below, left, and right, of the focal link. For all link positions (i, j), with $i, j \geq 2$ and $A(i, j) = 1$ we test moving the link to all possible neighbor positions, and for each of the resulting networks we compute NODF. In each hill climbing step, the network is modified by executing the NODF-maximising move of a link to a neighboring position. Links located in the first row and first column are excluded from this procedure, ensuring all species always have at least one link, and to avoid compartmentalization effects during this optimisation procedure (i.e. we ensure the network is always one connected component, and never comprises more than one disconnected components). We say that the matrices obtained by moving a link position to a neighboring position are neighbors of the previous matrix. The hill climbing algorithm terminates once no such local modification achieves a strictly higher NODF value. This ensures that we always terminate on a local maximum. Algorithm (4) shows how the hill climbing procedure is used to improve on a previous result.

Algorithm 3 Neighbors

procedure NEIGHBORS($A \in \{0,1\}^{N \times M}$)
  $\nu(A) \leftarrow \emptyset$
  $A(i, 1) \leftarrow 1 \ \forall i = 1, \ldots N$
  $A(1, j) \leftarrow 1 \ \forall j = 1, \ldots M$
  $A(2, 2) \leftarrow 1$
  for all $(i^*, j^*) \in \{(i, j) \in \mathbb{N} \times \mathbb{N} | 2 \leq i \leq N, 2 \leq j \leq M, A(i, j) = 1\}$ **do**
    for all $(i, j) \in \{(i, j) \in \mathbb{N} \times \mathbb{N} | i^* - 1 \leq i \leq i^* + 1, j^* - 1 \leq j \leq j^* + 1, A(i, j) = 0\}$ **do**
      if $2 \leq i \leq N, 2 \leq j \leq M$ **then**
        $A(i, j) \leftarrow 1$
        $A(i^*, j^*) \leftarrow 0$
        $\nu(A) \leftarrow \nu(A) \cup \{A\}$
        $A(i, j) \leftarrow 0$
        $A(i^*, j^*) \leftarrow 1$
      end if
    end for
  end for
  return $\nu(A)$
end procedure

Algorithm 4 Hillclimbing

```
1: procedure HillClimb(A ∈ {0,1}^{N×M})
2:     w ← True
3:     while w == True do
4:         ν(A) ← neighbors(A)
5:         Ã ← argmax_{B∈ν(A)} NODF(B)
6:         if NODF(Ã) > NODF(A) then
7:             A ← Ã
8:         else
9:             w ← False
10:        end if
11:    end while
12:    return A
13: end procedure
```

Quality 2: Greedy algorithm with hill climbing and simulated annealing

For the quality parameter of 2, we first apply the same greedy and hill climbing algorithms as in quality settings 0 and 1. Saving a copy of the starting position, and updating it upon finding a new best network ensures that quality 2 will never perform worse then 0 or 1. Applying the hill climbing algorithm after a new optimal network is found ensures that the algorithm always terminates in a local optimum. As a result, moving a link to a neighboring position cannot improve the NODF value. To further improve on such networks we require algorithms that are able to escape from local optima. Such algorithms necessarily have the property that worse results need to be accepted in certain situations.

One such algorithm is called simulated annealing and was proposed by Kirkpatrick et al. (1983). In simulated annealing, we accept network modifications which reduce the NODF with a certain probability. This probability is determined by a temperature parameter T and the reduction in NODF which this move has introduced is denoted by $\Delta\,\mathrm{NODF}$. We then accept the move according to the Kirkpatrik acceptance probability function

$$P(\Delta\,\mathrm{NODF}, T) = \begin{cases} 1, & \text{if } \Delta\,\mathrm{NODF} <= 0 \\ \exp\left(-\frac{\Delta\,\mathrm{NODF}}{T}\right), & \text{otherwise} \end{cases}. \tag{31}$$

After each step of the simulated annealing algorithm, the temperature T is reduced by setting $T^{(k)} = \alpha T^{(k-1)}$, where $0 < \alpha < 1$. As result the algorithm is more likely to accept large drops in NODF when T is large, resulting in a fast scan across the space of possible networks at the start of the algorithm, followed by a more concentrated search in the area where we expect the true optimal network, when T is low. Simulated annealing is an established and reliable method within network research, having been used, for example, to obtain the network partition which maximises modularity (Guimera & Amaral 2005). The core of our simulated annealing algorithm follows the same ideas as these past examples. However, our novelty comes from an highly efficient implementation, which results in fast computation speeds, as well as combining simulated annealing with other algorithms – greedy and hill climbing – to produce bespoke algorithms for NODF maximisation, as detailed below.

We combine the simulated annealing with hill climbing by starting a search for local optima whenever the simulated annealing algorithm reaches a new NODF-maximising network. This condition guarantees that the hill climbing algorithm does not reset the algorithm the previous local optimum. The algorithm used by `maxnodf` at quality 2 is given below in pseudocode. Note that neighbour solutions are computed as shown in Algorithm (3), and, as in the previous algorithms,

the first row and column of the network are never changed, ensuring all species always have at least one link.

Algorithm 5 SimulatedAnnealing

procedure MAXNODF($A \in \{0,1\}^{N,M}$, quality=2) ▷ Simulated annealing with hill climbing
  $A_{\text{opt}} \leftarrow A$
  $T \leftarrow T_0$
  while $T > T_{\min}$ **do**
    $\tilde{A} \leftarrow \text{neighbor}(A)$
    $\Delta\,\text{NODF} \leftarrow \text{NODF}(A) - \text{NODF}(\tilde{A})$
    if rand() $> P(\Delta\,\text{NODF}, T)$ **then**
      $A \leftarrow \tilde{A}$
      if $\text{NODF}(A) > \text{NODF}(A_{\text{opt}})$ **then**
        $A_{\text{opt}} \leftarrow \text{HillClimb}(A)$
      end if
    end if
    $T \leftarrow \alpha T$
  end while
  return A_{opt}
end procedure

4 Analysis S1

To test if it might ever be misleading to use the lower quality greedy algorithm, we correlate the maximum NODF achieved by the greedy (quality 0) and simulated annealing (quality 2) algorithms for the 135 networks in our dataset.

We find that the simulated annealing algorithm and the greedy algorithm produce almost identical results when the maximum nestedness is high, but the simulated annealing algorithm produces increasingly better maxima than the greedy algorithm when the maximum nestedness is low (Figure S5). Specifically, we find that the percentage increase in maximum NODF achieved by the simulated annealing algorithm is kept to below 5% when the maximum NODF in the network (assessed by the greedy algorithm) is greater than 0.6 (Figure S5c). When maximum NODF is below 0.6, the simulated annealing algorithm produces maxima that are sometimes greater than a 5% increase relative to the greedy algorithm (Figure S5c).

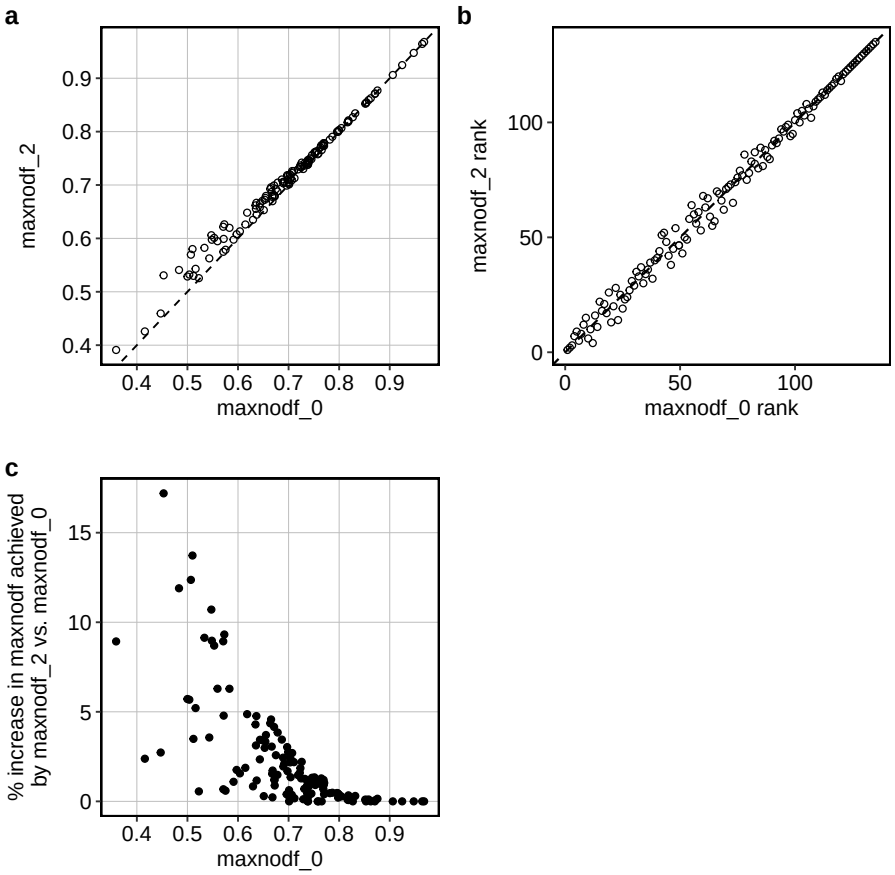

Figure S5: Comparison of the maximum nestedness achieved by the greedy (maxnodf_0) and simulated annealing (maxnodf_2) algorithms. Network data were the same 135 networks as in Figure 1. (a) Relationship between the raw nestedness values produced by the two algorithms. (b) Relationship between the ranks of maximum nestedness values produced by the two algorithms. (c) Relationship between the maximum nestedness calculated by the greedy algorithm and the percentage increase in maximum nestedness achieved by using the simulated annealing algorithm over the greedy algorithm. As long as the greedy algorithm reports a maximum nestedness above 0.6, there is less than a 5% increase in maximum nestedness to be gained by using a simulated annealing algorithm. Thus, if the greedy algorithm reports a maximum nestedness of >0.6, it may not be necessary to use the simulated annealing algorithm.

We therefore recommend that if time or computational constraints prevent running all networks through the simulated annealing algorithm, best practice might involve running all networks in a dataset through the greedy algorithm, and if any of these networks have a maximum nestedness below 0.6, we recommend running these networks through the simulated annealing algorithm.

We believe that the greedy algorithm is the best choice for most questions and think that the sacrifice in quality would rarely change conclusions qualitatively. This is supported by a recent re-analysis which found that using higher-quality simulated annealing algorithms to calculate nestedness did not qualitatively change conclusions (Song et al. 2019). Higher-quality algorithms should therefore only be used if computational time is not a constraint. The highest quality algorithm does technically produce more accurate results and, for a small number of networks, large percentage changes in maximum NODF, but in aggregate this is unlikely to make qualitative difference to results, unless all networks in the dataset have unusually low maximum nestedness values (which can be assessed beforehand using the greedy algorithm).

5 Analysis S2

In their seminal paper on nestedness, Bascompte et al. (2003) normalized nestedness values as $(E-N)/N$, where E is the observed nestedness of a network and N is the mean nestedness across an ensemble of null networks. This is in contrast to z-scores, which are normalized as $(E–N)/\sigma$, where σ is the standard deviation of nestedness across an ensemble of null networks.

We tested for a correlation between null model measures and the $NODF_c$ measure presented in this paper. For the null model measure we normalised by dividing by the average of the null replicates rather than their standard deviation. We ran the analysis for the same 135 networks as used in the benchmarking analysis. We used the 'curveball' algorithm to generate null networks, which preserve degree distribution and connectance (Strona et al. 2014). We found no strong association between our measure $NODF_c$ and the null model measure (correlation coefficient 0.17).

There was no significant relationship between the two measures. Although the P value was approaching significance (0.0512), the R squared was low (0.02). Visually, the two measures appear highly unrelated (Figure S6). Two outliers were removed from this analysis whose Cooks distance exceeded $4/n$, where n is the number of data points.

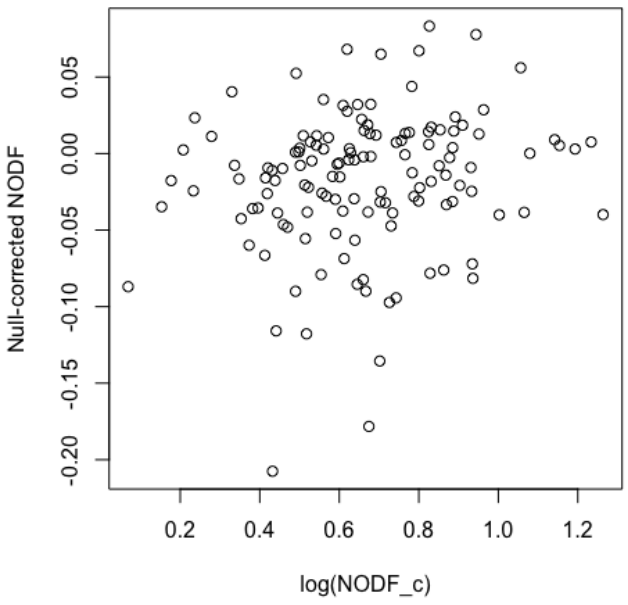

Figure S6: Association between NODFc and null-corrected NODF, where null-corrected NODF is defined as $(E–N)/N$, where E is the observed nestedness of a network and N is the mean nestedness across an ensemble of null networks.

References

Almeida-Neto, M., Guimarães, P., Guimarães Jr, P. R., Loyola, R. D. & Ulrich, W. (2008), 'A consistent metric for nestedness analysis in ecological systems: reconciling concept and measurement', *Oikos* **117**(8), 1227–1239.

Bascompte, J., Jordano, P., Melián, C. J. & Olesen, J. M. (2003), 'The nested assembly of plant-animal mutualistic networks', *Proceedings of the National Academy of Sciences* **100**(16), 9383–9387.

Fortuna, M. A., Barbour, M. A., Zaman, L., Hall, A. R., Buckling, A. & Bascompte, J. (2019), 'Coevolutionary dynamics shape the structure of bacteria-phage infection networks', *Evolution* **73**(5), 1001–1011.

Guimera, R. & Amaral, L. A. N. (2005), 'Functional cartography of complex metabolic networks', *Nature* **433**(7028), 895–900.

Karp, R. M. (1972), Reducibility among combinatorial problems, *in* 'Complexity of computer computations', Springer, pp. 85–103.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *Science* **220**(4598), 671–680.

Parker, R. G. & Rardin, R. L. (2014), *Discrete optimization*, Elsevier.

Simmons, B. I., Hoeppke, C. & Sutherland, W. J. (2019), 'Beware greedy algorithms', *Journal of Animal Ecology* **88**, 804–807.

Song, C., Rohr, R. P. & Saavedra, S. (2017), 'Why are some plant–pollinator networks more nested than others?', *Journal of Animal Ecology* **86**(6), 1417–1424.

Song, C., Rohr, R. P. & Saavedra, S. (2019), 'Beware z-scores', *Journal of Animal Ecology* **88**(5), 808–809.

Strona, G., Nappo, D., Boccacci, F., Fattorini, S. & San-Miguel-Ayanz, J. (2014), 'A fast and unbiased procedure to randomize ecological binary matrices with fixed row and column totals', *Nature Communications* **5**(1), 1–9.