Appendix A. Supplementary Materials to the paper "Updating and downdating techniques for optimizing network communicability" by Francesca Arrigo and Michele Benzi.

Abstract. In this document we summarize a few supplementary results to the accompanying paper. We give a detailed proof of the bounds on the normalized total communicability obtained via quadrature rules (Theorem 3.1, section 3). We also derive the computational costs for the heuristics introduced. Moreover, this document contains the results of some numerical experiments performed on four small networks to assess the valuability of our techniques. We provide full descriptions four our downdating and updating algorithms and for the updating algorithm developed in [4]. Finally, we briefly recall the approach used to relate the Estrada index of a graph to its Helmholtz free energy.

A.1. Bounds via quadrature rules: a proof of Theorem 3.1.

In this section we give a proof of Theorem 3.1. In order to make this document self-contained, we briefly recall here the technique used to derive bounds via quadrature rules on bilinear forms.

Bounds on bilinear forms $\mathbf{u}^T f(A)\mathbf{v}$ can be derived based on Gauss–type quadrature rules when f is a *strictly completely monotonic* (s.c.m.) function on the interval $[a,b]$ containing the spectrum of A by working on a 2×2 matrix derived from one step of the symmetric Lanczos iteration (see [2, 10]). Recall that a function is s.c.m. on $[a,b]$ if $f^{(2l)}(x) > 0$ and $f^{(2l+1)}(x) < 0$ for all $x \in [a,b]$ and for all $l \geq 0$, where $f^{(k)}$ denotes the kth derivative of f and $f^{(0)} \equiv f$. In order to compute bounds for the normalized total communicability, this means that we need to use $f(x) = e^{-x}$ and therefore we work with the matrix $-A$.

The starting point is to observe that bilinear forms $\mathbf{u}^T f(A)\mathbf{v}$ can be thought of as Riemann–Stieltjes integrals with respect to the spectral measure associated with the symmetric matrix A:

$$\mathbf{u}^T f(A)\mathbf{v} = \int_a^b f(\lambda)dm(\lambda), \quad m(\lambda) = \begin{cases} 0, & \lambda < a = \lambda_n \\ \sum_{k=i+1}^n w_k z_k, & \lambda_{i+1} \leq \lambda < \lambda_i \\ \sum_{k=1}^n w_k z_k, & b = \lambda_1 \leq \lambda \end{cases}$$

where $A = Q\Lambda Q^T$, $\mathbf{w} = Q^T\mathbf{u} = (w_i)$, and $\mathbf{z} = Q^T\mathbf{v} = (z_i)$.

This integral can be approximated by means of Gauss–type quadrature rules, which can be used to obtain lower and upper bounds on the bilinear forms of interest. In particular, our bounds are derived using the Gauss–Radau quadrature rule:

$$\int_a^b f(\lambda)dm(\lambda) = \sum_{j=1}^p c_j f(t_j) + v_1 f(\tau_1), \tag{A.1}$$

where the nodes $\{t_j\}_{j=1}^p$ and the weights $\left\{\{c_j\}_{j=1}^p, v_1\right\}$ are to be determined, whereas τ_1 is prescribed and equal either to a or to b. The Gauss–Radau bounds are then as described in the following theorem.

Theorem A.1 (6.4 in [10]). *Suppose f is such that $f^{(2l+1)}(\xi) < 0$ for all l and for all $\xi \in (a,b)$. Let U_{GR} be defined as*

$$U_{GR}[f] = \sum_{j=1}^p c_j^a f(t_j^a) + v_1^a f(a),$$

c_j^a, v_1^a, t_j^a being the weights and nodes in (A.1) with $\tau_1 = a$, and let L_{GR} be defined as

$$L_{GR}[f] = \sum_{j=1}^p c_j^b f(t_j^b) + v_1^b f(b),$$

c_j^b, v_1^b, t_j^b being the weights and nodes in (A.1) with $\tau_1 = b$. The Gauss–Radau rule is exact for polynomials of degree less than or equal to $2p$ and satisfies

$$L_{GR}[f] \le \int_a^b f(\lambda) dm(\lambda) \le U_{GR}[f].$$

Moreover for all p there exists $\eta_U, \eta_L \in [a,b]$ such that

$$\int_a^b f(\lambda) dm(\lambda) - U_{GR}[f] = \frac{f^{(2p+1)}(\eta_U)}{(2p+1)!} \int_a^b (\lambda - a) \left[\prod_{j=1}^{p} (\lambda - t_j^a) \right]^2 dm(\lambda),$$

$$\int_a^b f(\lambda) dm(\lambda) - L_{GR}[f] = \frac{f^{(2p+1)}(\eta_L)}{(2p+1)!} \int_a^b (\lambda - b) \left[\prod_{j=1}^{p} (\lambda - t_j^b) \right]^2 dm(\lambda).$$

It is therefore necessary to evaluate two quadrature rules, one for the upper bound and one for the lower bound. However, the explicit computation of nodes and weights can be avoided. Indeed, the evaluation of the quadrature rules is mathematically equivalent to the computation of orthogonal polynomials via a three–term recurrence, or, equivalently, to the computation of entries and spectral information of a certain tridiagonal matrix via the Lanczos algorithm. In fact, the right hand side of equation (A.1) can be computed from the relation (Theorem 6.6 in [10]):

$$\text{(A.2)} \qquad \sum_{j=1}^{p} c_j f(t_j) + v_1 f(\tau_1) = \mathbf{e}_1^T f(J_{p+1}) \mathbf{e}_1,$$

where

$$J_{p+1} = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \gamma_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{p-1} & \omega_p & \gamma_p \\ & & & \gamma_p & \omega_{p+1} \end{pmatrix}$$

is a tridiagonal matrix whose eigenvalues are the Gauss–Radau nodes (and hence J_{p+1} is built so as to have the prescribed eigenvalue τ_1), whereas the weights are given by the squares of the first entry of the normalized eigenvectors of J_{p+1}. An efficient implementation of this technique is provided in G. Meurant's `mmq` toolbox for Matlab [14]. This toolbox, adapted to handle sparsity, has been used for some of the numerical experiments presented in the paper.

We can now prove Theorem 3.1, which contains the bounds for the normalized total network communicability. The proofs of the subsequent corollaries follow the same line.

Proof. First we derive an explicit expression for the right–hand side of equation (A.2) when $f(x) = e^{-x}$ and J_2 is 2×2 with the help of the Lagrange interpolation formula for the evaluation of matrix functions [12]. Let μ_1 and μ_2 be distinct eigenvalues of a given 2×2 matrix $B = (b_{ij})$, then

$$e^{-B} = \frac{e^{-\mu_1}}{\mu_1 - \mu_2}(B - \mu_2 I) + \frac{e^{-\mu_2}}{\mu_2 - \mu_1}(B - \mu_1 I)$$

where I is the 2×2 identity matrix. It follows that

$$\mathbf{e}_1^T \left(e^{-B}\right) \mathbf{e}_1 = \frac{b_{11}(e^{-\mu_1} - e^{-\mu_2}) + \mu_1 e^{-\mu_2} - \mu_2 e^{\mu_1}}{\mu_1 - \mu_2}.$$

Next, we build explicitly the matrix J_2 and compute its eigenvalues. The values of $\omega_1 = -\mu$ and $\gamma_1 = \sigma$ are derived applying one step of Lanczos iteration to the matrix $-A$ with starting vectors $\mathbf{x}_{-1} = \mathbf{0}$ and $\mathbf{x}_0 = \frac{1}{\sqrt{n}}\mathbf{1}$. We want to compute the value of ω_2 in such a way that the matrix J_2 has the prescribed eigenvalue $\tau_1 = \alpha$ or $\tau_1 = \beta$. Note that $\gamma_1 = 0$ if and only if the graph is regular, i.e., if and only if the nodes in the graph have all the same degree. In such case we simply take $\omega_2 = \tau_1$ and the matrix J_2 is diagonal with eigenvalues $\mu_1 = -\mu$ and $\mu_2 = \tau_1$. Thus, let us assume $\gamma_1 \neq 0$. In order to compute the value for ω_2, we use the three–term recurrence for orthogonal polynomials:

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j)p_{j-1}(\lambda) - \gamma_{j-1}p_{j-2}(\lambda), \quad j = 1, 2, \dots, p,$$

with $p_{-1}(\lambda) \equiv 0$, $p_0(\lambda) \equiv 1$ to impose that $p_2(\tau_1) = 0$ and hence derive $\omega_2 = \tau_1 - \frac{\gamma_1}{p_1(\tau_1)}$. Using the same recurrence, we also find that $p_1(\tau_1) = \frac{\tau_1 - \omega_1}{\gamma_1}$ which is nonzero, since the zeros of orthogonal polynomials satisfying the three–term recurrence are distinct and lie in the interior of $[\alpha, \beta]$ (see [10, Theorem 2.14]).

Finally, the matrix

$$J_2 = \begin{pmatrix} \omega_1 & \gamma_1 \\ \gamma_1 & \tau_1 - \frac{\gamma_1^2}{\tau_1 - \omega_1} \end{pmatrix}$$

has (distinct) eigenvalues $\mu_1 = \tau_1$ and $\mu_2 = \omega_1 + \frac{\gamma_1^2}{\omega_1 - \tau_1}$. This, together with Theorem A.1 and the relation (A.2), concludes the proof. $\square$

A.2. Computational aspects. In this section we describe some technical details that need to be kept in mind when implementing our heuristics. Moreover, we explicitly derive the computational costs of our methods.

There are several important points to keep in mind when implementing the methods described in the paper. First of all, for the downdates, updates, and rewires based on the edge subgraph centrality we need to compute the diagonal entries of e^A. This is the most expensive part of these methods. There are, however, techniques that can be used to rapidly estimate the diagonal entries of e^A and to quickly identify the top ℓ nodes, where $\ell \ll n$; see [1, 3, 9] and references therein. It should be pointed out that very high accuracy is not required or warranted by the problem. We also recall that the same techniques (based on quadrature rules and the Lanczos process) can be used to compute the total communicability $\mathbf{1}^T e^A \mathbf{1}$ quickly (typically in $O(n)$ work), although such computation is actually not required by any of the algorithms tested here except by the `optimal` strategy, which is only used (for small networks) as a baseline method. Such methods can also be used for rapidly estimating the node total communicability centralities, $TC(i) = [e^A\mathbf{1}]_i = \mathbf{e}_i^T e^A \mathbf{1}$.

Secondly, when performing an update or the updating phase of a rewire, it makes sense to work with a subset of the set of all virtual edges $\overline{E}$. Indeed, for a sparse network $\overline{E}$ contains $O(n^2)$ edges and for large n this is prohibitive. Due to the particular selection criteria we want to use, it is reasonable to restrict ourselves to the virtual edges in the subgraph of our network that are incident to a subset S of nodes containing a certain percentage of the top nodes, ranked according to some centrality

Table A.1

Computational costs for the downdating and updating techniques introduced in the accompanying paper.

Method	Downdate	Update
`optimal`	$O(Kn^2)$	$O(Kn^3)$
`subgraph`	$O(Kn)$	$O(K\ell n)$
`eigenvector`	$O(Kn)$	$O(K\ell n)$
`nodeTC`	$O(Kn)$	$O(K\ell n)$
`degree`	$O(Kn)$	$O(Kn)$
`subgraph.no`	$O(n \log n)$	$O((K+\ell)n)$
`eigenvector.no`	$O(n \log n)$	$O((K+\ell)n)$
`nodeTC.no`	$O(n \log n)$	$O((K+\ell)n)$

measure. We found that for the larger networks considered in the paper, using just the top 1% of the nodes ranked using eigenvector centrality yields very good results.

Next, we derive the computational costs for the downdating techniques used in the accompanying paper. Let m be the number of edges in the network and let $K \ll n$, assumed bounded independently of n as $n \to \infty$, be the maximum number of modifications we want to perform; in this paper, the maximum value of K we consider is 2000 (used for the three largest networks in our data set).

In the `optimal` method we remove each edge in turn, compute the total communicability after each downdate, and then choose the downdate which caused the least decrease in $TC(A)$; assuming that the cost of computing $TC(A)$ is $O(n)$, we find a total cost of $O(Kmn)$ for K updates. Since $m = O(n)$, this amounts to $O(Kn^2)$.

Next, we consider the cost of techniques based on subgraph centralities. The cost of computing the node subgraph centralities is not easy to assess in general, since it depends on network properties and on the approximation technique used. If a rank-k approximation is used [9], the cost is approximately $O(kn)$; hence, the cost is linear in n if k is independent of n, which is appropriate for many types of networks. Computing the edge centralities requires another $m = O(n)$ operations, and sorting the edges by their centralities costs approximately $m \ln m$ comparisons. Note that sorting is only necessary in the `subgraph.no` variant of the algorithm; indeed, with `subgraph` we recompute the centralities after each update and instead of sorting the result we only need to identify the edge of minimum centrality at each step, which can be done in $O(m)$ work. Summarizing, the cost of `subgraph` is $O(K(n+m)) = O(Kn)$ for K downdates if we assume the subgraph centralities can be computed in $O(n)$ time, and the cost for `subgraph.no` is $O(n+m) = O(n)$ plus a pre-processing cost of $O(m \ln m)$ $(= O(n \ln n))$ comparisons for sorting the edge centralities. Although the asymptotic cost of `subgraph.no` appears to be higher than that of `subgraph` (due to the $n \ln n$ term), in practice one finds that `subgraph.no` runs invariably much faster than `subgraph` for all cases tested here.

The costs associated with `eigenvector` and `eigenvector.no` scale like those of `subgraph` and `subgraph.no`, assuming that the dominant eigenvector $\mathbf{q}_1$ of a large sparse $n \times n$ adjacency matrix can be approximated in $O(n)$ time. For many real world networks this is a reasonable assumption, since in practice we found that running a fixed number of Lanczos steps will give a sufficiently good approximation of $\mathbf{q}_1$. The prefactors can be expected to be much smaller for the methods based on eigenvector centrality than for those based on subgraph centrality.

The costs for `nodeTC` and `nodeTC.no` are comparable to those for `eigenvector` and `eigenvector.no`, with the same asymptotic scalability.

Table A.2
Description of the Data Set.

NAME	n	m	λ_1	λ_2	$\lambda_1 - \lambda_2$
Zachary	34	78	6.726	4.977	1.749
Sawmill	36	62	4.972	3.271	1.701
social3	32	80	5.971	3.810	2.161
dolphins	62	159	7.193	5.936	1.257

Finally, the cost of `degree` is $O(Km)$ and hence also $O(Kn)$ for a sparse network.

Note that the cost of checking that the connectivity is preserved after each downdate does not affect these asymptotic estimates; indeed, using A^* search [11] this can be done in $O(m)$ time and hence the additional cost is only $O(n)$ for a sparse network. Of course, if the removal of an edge is found to disconnect the network we do not perform the downdate and move on to the next candidate edge.

We consider next the computational cost for the updating strategies. As before we let K, assumed bounded independently of n as $n \to \infty$, be the maximum number of updates we want to perform.

It can be easily shown that the `optimal` method costs $O(Kn^3)$ operations. To estimate the cost of the remaining methods, we assume that the set $S \subset V$ consisting of the top $\ell = |S|$ nodes (ranked according to some centrality measure) is known. The cost of determining this set is asymptotically dominated by the term $O(n \ln n)$, as we saw. As already mentioned, ℓ will be equal to some fixed percentage of the total number of nodes in the network.

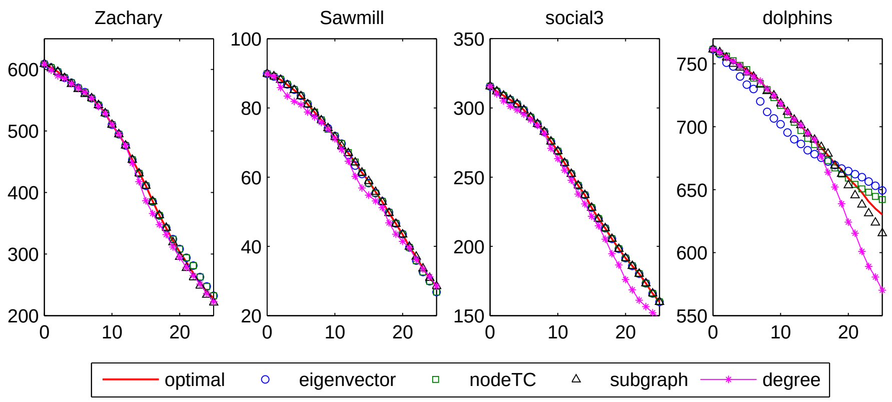
Both `subgraph` and `eigenvector` cost $O(K\ell n)$ operations, provided a low rank approximation (of fixed rank) is used to estimate the subgraph centralities. The same holds for `nodeTC`. Typically, the prefactor will be larger for the former method. Since we assumed that $\ell = O(n)$ (albeit with a very small prefactor, like 10^{-2}) these methods exhibit an $O(n^2)$ scaling. In practice this is somewhat misleading, since the quadratic scaling is not observed until n is quite large.

Finally, `degree` costs $O(K\ell) = O(n)$ while `subgraph.no`, `eigenvector.no` and `nodeTC.no` all cost $O((K + \ell)n)$. Again, the latter cost is asymptotically quadratic but the actual cost is dominated by the linear part until n becomes quite large. We note that we can obtain an asymptotically linear scaling by imposing an upper bound on ℓ, i.e., on the fraction of nodes that we are willing to include in the working subset S of nodes. We stress that because of the widely different prefactors for the various methods, these asymptotic estimates should only be taken as roughly indicative.

A.3. Numerical tests: small networks. In this section we present the results obtained when performing numerical tests on four networks of small size. For these networks it is possible to apply the `optimal` strategy and to compare the other, more practical strategies with it. The results of this comparison serve as a justification for the use of our heuristics on larger networks.

The real-world networks used in the tests (see Table A.2) come from a variety of sources. The Zachary Karate Club network is a classic example in social network analysis [17]. The Sawmill and social3 networks were provided to us by Prof. Ernesto Estrada. The Sawmill network describes a communication network within a small enterprise (see [15, 16]), whereas social3 is a network of social contacts among college students participating in a leadership course (see [18]). The network dolphins (see [13]) is in the Newman group from the Florida Sparse Matrix Collection [5] and represents a social network of frequent associations between 62 dolphins in a community living in

FIG. A.1. *Evolution of the normalized total communicability vs. number of downdates performed on small networks.*



the waters off New Zealand. Table A.2 reports the number of nodes (n), the number of edges (m), the two largest eigenvalues, and the spectral gap. We use these networks to test all the greedy methods described in the accompanying paper.

We begin by showing results for the four smallest networks. Figure A.1 displays the results obtained with the downdating methods `optimal`, `eigenvector`, `nodeTC`, `subgraph`, and `degree`. The results for `eigenvector.no`, `subgraph.no`, and `nodeTC.no` are virtually indistinguishable from those obtained with `eigenvector`, `subgraph` and `nodeTC` and are therefore not shown. At each step we modify the network by downdating an edge and we then compute and plot the new value of the normalized total communicability. The tests consist of 25 modifications.

Figure A.1 shows that our methods all perform similarly and give results that are in most cases very close to those obtained with `optimal`, and occasionally even better, as is the case for `eigenvector` (and `eigenvector.no`) on the dolphins network after a sufficient number of downdates have been performed. This result may seem puzzling at first, however, it can be easily explained by noticing that `eigenvector` selects a different edge from that selected by `optimal` at the third downdate step. Hence, from that point on the adjacency matrices on which the methods work are different, and the choice performed by the `optimal` method may no longer be optimal for the graph manipulated by `eigenvector`. Note that even the simple heuristic `degree` seems to perform well, except perhaps on the dolphins network after 15 or so downdate steps. Overall, the methods based on eigenvector and total communicability centrality appear to perform best in view of their efficacy and low cost.

The results for the updating methods are reported in Figure A.2. As for the downdating methods, `subgraph.no`, `eigenvector.no`, and `nodeTC.no` return results that are virtually identical to those obtained using `subgraph`, `eigenvector` and `nodeTC`, therefore we omit them from the figure. Once again we see that the methods based on eigenvector, subgraph, and total communicability centrality give excellent results, whereas `degree` is generally not as effective.

Rewiring results are displayed in Figure A.3. Clearly, the methods making use of edge centrality perform quite well, in contrast to `random` rewiring (which is only included as a base for comparison). Note also the poor performance of `node`, showing that the use of edge centralities (as opposed to node centralities alone) is indispensable in this context.

FIG. A.2. *Evolution of the normalized total communicability vs. number of updates performed on small networks.*

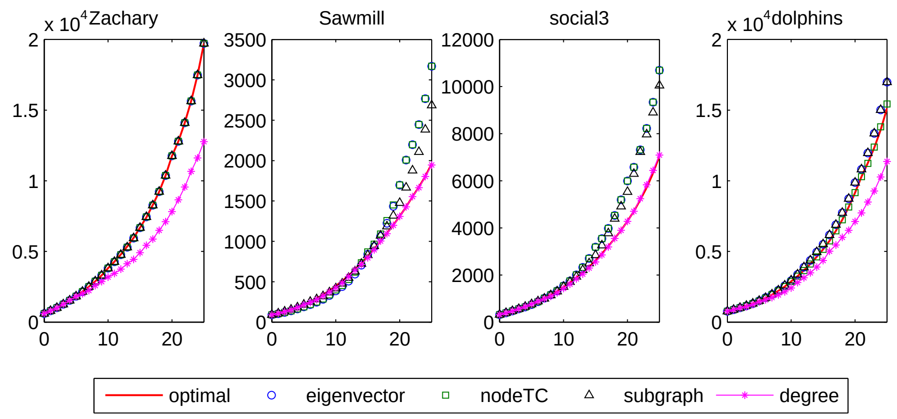
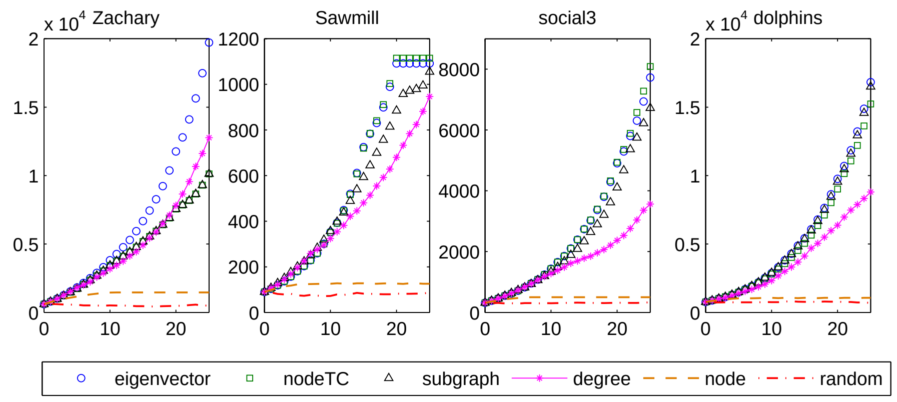


FIG. A.3. *Evolution of the normalized total communicability vs. number of rewires performed on small networks.*



The values obtained using the updates are in general higher than those obtained using the rewiring strategies, since updating implies the addition of edges whereas in rewiring the number of edges remains the same. For these methods the effects of downdates have a great impact, leading to a decrease by up to nearly 70% of the original value of the total communicability after 25 downdates (cf. Figure A.1). It is noteworthy that the methods based on the edge eigenvector and total communicability centrality appear to be more stable than the others under rewiring and to dampen the effect of the downdates even for small networks.

A.4. Algorithms. This section provides pseudocodes for all the algorithms used in the accompanying paper. Algorithms 1 and 2 implement our downdating techniques with or without the connectivity check, while Algorithm 3 implements our updating heuristics. All these three algorithms can be used with or without the recomputation of the rankings for the edges after each modification has been performed. They require as inputs the initial graph G (typically in the form of its adjacency matrix A) and a budget K, i.e., the number of modifications one wants to perform. The Boolean *greedy* indicates whether the rankings of the edges have to be recomputed after each

Algorithm 1: Downdating algorithm with connectivity check.

```
Data: Initial graph $G$, $K \in \mathbb{N}$, $greedy \in \{0,1\}$
Result: Set $\mathcal{S}$ of $K$ edges to be removed
$\mathcal{S} = \emptyset$;
$c = 0$;
$\mathcal{E}$ = list of edges in the graph;
if $greedy$ then
    while $(c < K)$ && $(|\mathcal{E}| > 0)$ do
        $found\_edge = 0$;
        Compute the centrality measure of interest $\forall (i,j) \in \mathcal{E}$;
        while $(found\_edge == 0)$ && $(|\mathcal{E}| > 0)$ do
            $G' = G$;
            $s$ = element in $\mathcal{E}$ with the smallest centrality;
            Downdate $s$ from $G'$;
            if $G'$ is connected then
                $G = G'$;
                $found\_edge = 1$;
                $\mathcal{S} = \mathcal{S} \cup \{s\}$;
                $c = c + 1$;
            end
            $\mathcal{E} = \mathcal{E} \setminus \{s\}$;
        end
    end
else
    $l = 1$;
    Compute edge centrality measure of interest $\forall (i,j) \in \mathcal{E}$;
    Sort the edges in ascending order;
    while $(c < K)$ && $(l \leq |\mathcal{E}|)$ do
        $G' = G$;
        $s$ = $l$th edge in the sorted array;
        Downdate $s$ from $G'$;
        if $G'$ is connected then
            $G = G'$;
            $c = c + 1$;
            $\mathcal{S} = \mathcal{S} \cup \{s\}$;
        end
        $l = l + 1$;
    end
end
Return $\mathcal{S}$.
```

modification ($greedy = 1$) or not ($greedy = 0$).

For the updating algorithm, it is also required to give in input a subset $S \subset V$ of nodes. The K modifications will be selected among the virtual edges in the subgraph containing the nodes in S and the corresponding edges.

Finally, Algorithm 4 contains a detailed description of the technique introduced in [4]. This algorithm requires as input the adjacency matrix A and a budget K, as our methods do. Moreover, this methods requires as input an integer t, which is the number of leading eigenpairs to be considered and updated during the search for the K modifications.

A.5. Free energy in networks. In this section we recall the approach used in [7] to relate the Estrada index of a network with its Helmholtz free energy and, consequently, with its Gibbs entropy.

Consider a network in which every edge is weighted by a parameter $\beta > 0$ and consider its adjacency matrix βA. The eigenvalues of this new matrix are $\beta\lambda_j$ for all

Algorithm 2: Downdating algorithm without connectivity check.

```
Data: Initial graph $G$, $K \in \mathbb{N}$, $greedy \in \{0,1\}$
Result: Set $\mathcal{S}$ of $K$ edges to be removed
$\mathcal{S} = \emptyset$;
$c = 0$;
$\mathcal{E}$ = list of edges in the graph;
if greedy then
    for $iter = 1 : K$ do
        Compute the centrality measure of interest $\forall (i,j) \in \mathcal{E}$;
        $s$ = element in $\mathcal{E}$ with the smallest centrality;
        Downdate $s$ from $G$;
        $\mathcal{S} = \mathcal{S} \cup \{s\}$;
        $\mathcal{E} = \mathcal{E} \setminus \{s\}$;
    end
else
    Compute edge centrality measure of interest $\forall (i,j) \in \mathcal{E}$;
    Sort the edges in ascending order;
    $\mathcal{S}$ = top $K$ elements in the sorted array;
end
Return $\mathcal{S}$.
```

Algorithm 3: Updating algorithm.

```
Data: Initial graph $G$, $K \in \mathbb{N}$, $S \subset V$ nodes in the subgraph, $greedy \in \{0,1\}$
Result: Set $\mathcal{S}$ of $K$ edges to be added
$\mathcal{S} = \emptyset$;
$\mathcal{E}$ = list of virtual edges in the subgraph containing nodes in $S$;
if greedy then
    for $iter = 1 : K$ do
        Compute edge centrality measure of interest $\forall (i,j) \in \mathcal{E}$;
        $s$ = element in $\mathcal{E}$ having the largest centrality;
        Update $s$ in $G$;
        $\mathcal{S} = \mathcal{S} \cup \{s\}$;
        $\mathcal{E} = \mathcal{E} \setminus \{s\}$;
    end
else
    Compute edge centrality measure of interest $\forall (i,j) \in \mathcal{E}$;
    Sort the edges in descending order;
    $\mathcal{S}$= top $K$ elements in the sorted array;
end
Return $\mathcal{S}$.
```

$j = 1, 2, \ldots, n$ and its Estrada index becomes $EE(G, \beta) = \mathrm{Tr}(e^{\beta A})$, where Tr denotes the trace. This index can be interpreted as the *partition function* of the corresponding complex network:

$$Z(G, \beta) := EE(G, \beta) = \mathrm{Tr}(e^{\beta A}).$$

Form the standpoint of quantum statistical mechanics, $\mathcal{H} = -A$ is the system Hamiltonian and $\beta = \frac{1}{k_B T}$ is the inverse temperature, with k_B the Boltzmann constant and T the absolute temperature. It is well known [6, 8] that β can be understood as a measure of the "strength" of the interactions between pairs of vertices; the higher the temperature (i.e., the lower the value of β), the weaker the interactions. The eigenvalues λ_i (for $i = 1, \ldots, n$) give the possible energy levels, each corresponding to a different state of the system.

The probability that the system is found in a particular state can be obtained by

```
**Algorithm 4:** Updating algorithm from [4].
  **Data**: $A$ adjacency matrix, $K \in \mathbb{N}$, and $t \in \mathbb{N}$.
  **Result**: Set $\mathcal{S}$ of $K$ edges to be added
  $\mathcal{S} = \emptyset$;
  Compute the top $t$ eigenpairs $(\lambda_k, \mathbf{q}_k)$ of $A$;
  **for** *iter = 1 : K* **do**
      Compute $d_{\max} = \max(d_i)$, the largest row sum of $A$ ;
      Find the set $C$ of $d_{\max}$ nodes with the highest eigenvector centrality;
      Select the edge $(i^*, j^*) \in \overline{E}$ that maximizes
          $$e^{\lambda_1}\left(e^{2q_1(i)q_1(j)} + \sum_{h=2}^{t} e^{\lambda_h-\lambda_1} e^{2q_h(i)q_h(j)}\right)$$
      and such that $i^*, j^* \in C$, $i^* \neq j^*$;
      $\mathcal{S} = \mathcal{S} \cup \{(i^*, j^*)\}$, $E = E \cup \{(i^*, j^*)\}$;
      Update $A$;
      Update the top $t$ eigenpairs as
          $$\begin{cases} \lambda_k = \lambda_k + 2q_k(i)q_k(j); \\ \mathbf{q}_k = \mathbf{q}_k + \sum_{h\neq k}\left(\frac{q_h(i)q_k(j)-q_k(i)q_h(j)}{\lambda_k-\lambda_h}\mathbf{q}_h\right) \end{cases} \qquad k = 1, 2, \ldots, t;$$
  **end**
  Return $\mathcal{S}$.
```

considering the Maxwell–Boltzmann distribution:

$$p_i = \frac{e^{\beta\lambda_i}}{EE(G,\beta)}, \quad i = 1, \ldots, n.$$

Using this notation and the fact that the Estrada index can be seen as the partition function of the system, in [7] the authors define the *Gibbs entropy* of the network as

$$S(G,\beta) = -k_B \sum_{i=1}^{n} p_i \ln(p_i) = -k_B\beta \sum_{i=1}^{n}(\lambda_i p_i) + k_B \ln(EE(G,\beta))$$

where in the last equality we have used the fact that $\sum_i p_i = 1$.

Using now the standard relation $F = H - TS$ that relates the *Helmholtz free energy* F, the *total energy* of the network H, the Gibbs entropy S, and the absolute temperature of the system T, the authors derive:

$$\begin{cases} H(G,\beta) = -\sum_{i=1}^{n} \lambda_i p_i, \\ F(G,\beta) = -\beta^{-1}\ln(EE(G,\beta)). \end{cases}$$

It is then clear that if we set $\beta = 1$ and let $F := F(G,1)$, we then get $F = -\ln(EE(G))$.

REFERENCES

[1] M. Benzi and P. Boito *Quadrature rule-based bounds for functions of adjacency matrices*, Linear Algebra Appl. 433 (2010), pp. 637–652.

[2] M. Benzi and G. H. Golub *Bounds for the entries of matrix functions with application to preconditioning*, BIT 39 (1999), pp. 417–438.

[3] M. Benzi and C. Klymko, *Total communicability as a centrality measure*, J. Complex Networks, 1(2) (2013), pp. 124–149.

[4] H. Chan, L. Akoglu, and H. Tong, *Make it or break it: manipulating robustness in large networks*, Proceedings of the 2014 SIAM Data Mining Conference (2014), Society for Industrial and Applied Mathematics, pp. 325–333.

[5] T. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices/.

[6] E. Estrada, *The Structure of Complex Networks. Theory and Applications*, Oxford University Press, 2012.

[7] E. Estrada and N. Hatano, *Statistical–mechanical approach to subgraph centrality in complex networks*, Chem. Phys. Lett. 439 (2007), pp. 247–251.

[8] E. Estrada, N. Hatano, and M. Benzi, *The physics of communicability in complex networks*, Phys. Rep., 514 (2012), pp. 89–119.

[9] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez, *Network analysis via partial spectral factorization and Gauss quadrature*, SIAM J. Sci. Comput, 35 (4) (2012), pp. A2046–A2068.

[10] G. H. Golub and G. Meurant, *Matrices, Moments and Quadrature with Applications*, Princeton University Press, Princeton, NJ 2010.

[11] P. E. Hart, N. J. Nilsson, and B. Raphael, *A formal basis for the heuristic determination of minimum cost paths*, IEEE Trans. Systems Sci. Cybernetics, 4 (2) (1968), pp. 100–107.

[12] N. J. Higham, *Function of Matrices, Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2008.

[13] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, *The bottlenose dolphin community in Doubtful Sound features a large proportion of long-lasting associations*, Behavioral Ecology and Sociobiology 54 (2003), pp. 396–405.

[14] G. Meurant, *MMQ toolbox for MATLAB*, http://pagesperso-orange.fr/gerard.meurant/.

[15] J. H. Michael and J. G. Massey, *Modeling the communication network in a sawmill*, Forest Products Journal, 47 (1997), pp. 25–30.

[16] W. de Nooy, A. Mrvar, and V. Batagelj, *Exploratory Social Network Analysis with Pajek*, Cambridge University Press, 2004.

[17] W. W. Zachary, *An information flow model for conflict and fission in small groups*, J. Anthropol. Res., 33 (1977), pp. 452–473.

[18] L. D. Zeleny, *Adaptation of research findings in social leadership to college classroom procedures*, Sociometry, 13 (4) (1950), pp. 314–328