Supplementary Information for Sustaining the Internet with hyperbolic mapping

Marián Boguñá
Departament de Física Fonamental, Universitat de Barcelona, Martí i Franquès 1, 08028 Barcelona, Spain

Fragkiskos Papadopoulos
Department of Electrical and Computer Engineering, University of Cyprus, Kallipoleos 75, Nicosia 1678, Cyprus

Dmitri Krioukov
Cooperative Association for Internet Data Analysis (CAIDA), University of California, San Diego (UCSD), La Jolla, CA 92093, USA

Contents

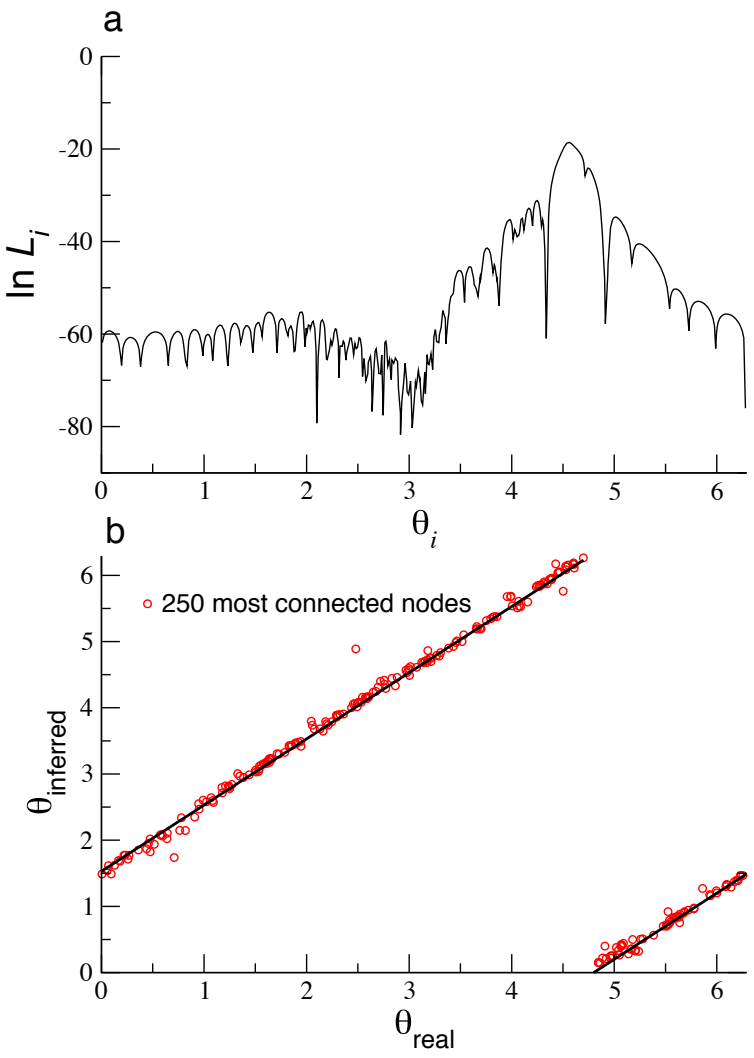
Supplementary Figures 2
Supplementary Figure S1: Testing the embedding method in synthetic networks (I) 2
Supplementary Figure S2: Testing the embedding method in synthetic networks (II) 3
Supplementary Figure S3: Testing the embedding method in synthetic networks (III) 4
Supplementary Figure S4: Measuring β 4
Supplementary Figure S5: Measuring congestion in the Internet 5

Supplementary Methods 6
MAPPING METHOD 6
General theory behind likelihood maximization 7
MLE for expected degrees κ 7
MLE for angular coordinates θ 8
MLE kernels 8
First MLE wrapper 8
Algorithm 1 9
Second MLE wrapper 9
Algorithm 2 10
Parameter estimation and finite size effects 10
Estimating γ 10
Estimating N and $\bar{k}$ 11
Estimating β 12
DEALING WITH NEW-COMING ASs 12
SENSITIVITY TO MISSING LINKS 14
MAPPING ASs TO COUNTRIES 15
GEOGRAPHIC ROUTING 16
TRAFFIC AND CONGESTION CONSIDERATIONS 17

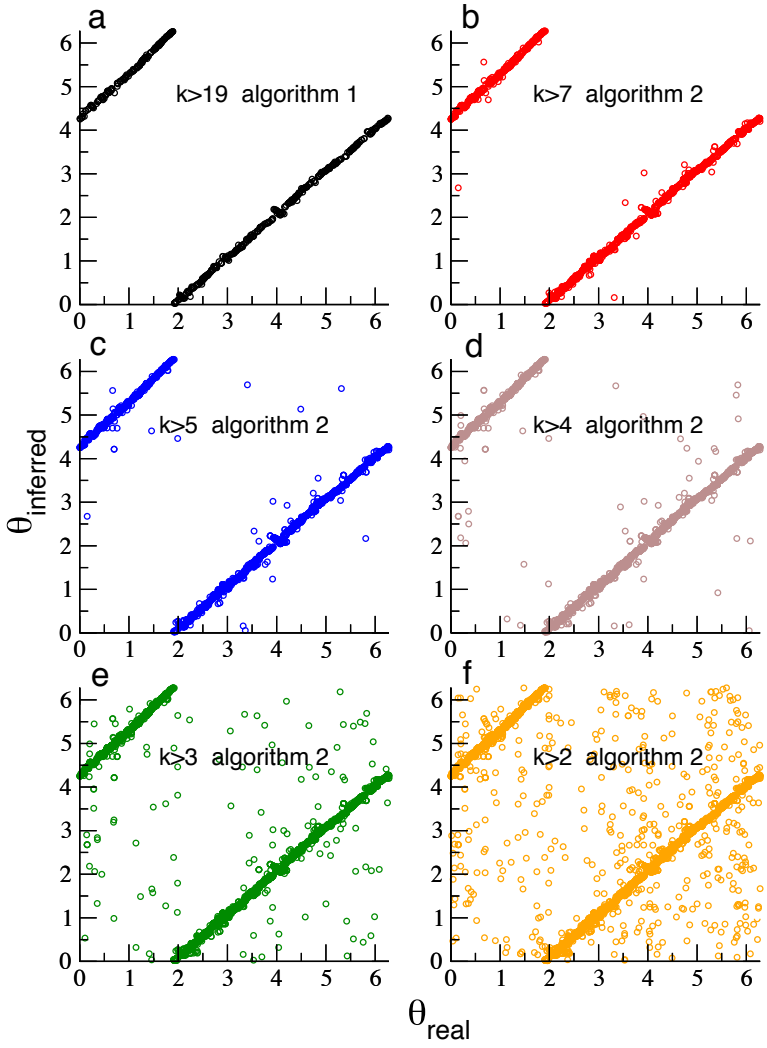
Supplementary Notes 18
AS-LEVEL ROUTING 18

Supplementary References 19

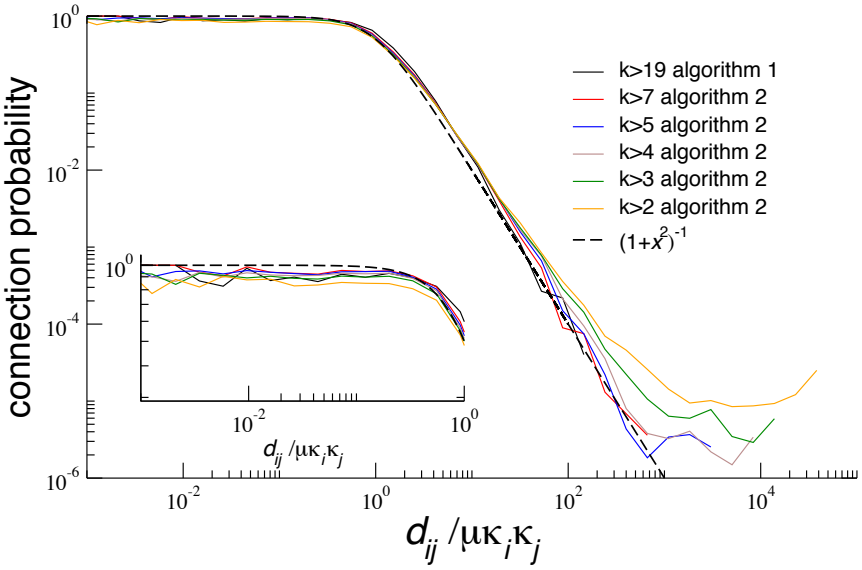
Supplementary Figures



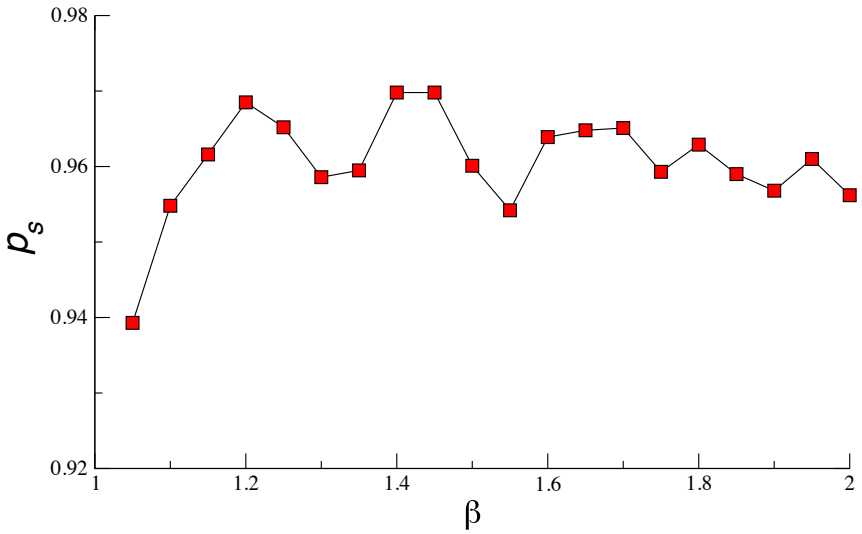
SUPPLEMENTARY FIGURE S 1: **Testing the embedding method in synthetic networks (I).** In **a** we show an example of the local log-likelihood of node i (Eq. (25)), having the coordinates of all other nodes fixed at an intermediate step of the mapping process. In **b** we show inferred angular coordinates vs. real ones for the 250 most connected nodes in a synthetic $\mathbb{S}^1$ network with the same properties as the real Internet: $\gamma = 2.1$, $\beta = 2$, $N \approx 24000$, $\bar{k} \approx 5$. The LMH kernel and Alg. 1 are used for coordinate inference. Notice that both inferred and real angular coordinates correlate up to an arbitrary rotation.



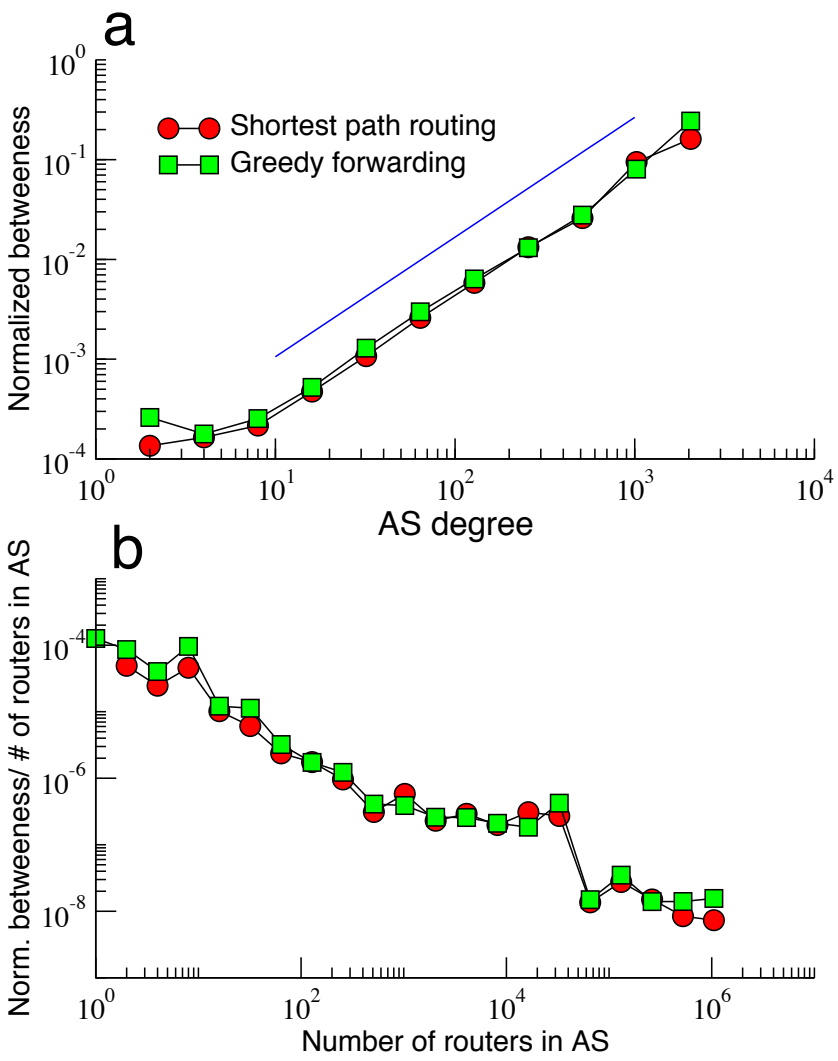
SUPPLEMENTARY FIGURE S 2: **Testing the embedding method in synthetic networks (II).** In **a**, we show the same as in the bottom plot of Supplementary Fig. 1 for nodes of degrees $k \geq 20$. All other plots (**b-f**) show the results of adding layers of lower degree nodes using the second MLE wrapper with $k_{l_{critic}} = 20$. According to Alg. 2, nodes in newly added layers infer their coordinates using the coordinates of nodes in existing layers, without running the MLE kernel, and the coordinates of existing nodes do not change as new nodes are added.



SUPPLEMENTARY FIGURE S 3: **Testing the embedding method in synthetic networks (III).** Empirical connection probability based on the inferred node coordinates, compared to the connection probability used to generate the synthetic network. The inset shows the details in the small distance region.



SUPPLEMENTARY FIGURE S 4: **Measuring β.** Success ratio of greedy forwarding as a function of β for the Internet graph mapping. We chose the value of β that maximises p_s.



SUPPLEMENTARY FIGURE S 5: **Measuring congestion in the Internet.** **a** standard normalized betweenness as a function of the AS degree for shortest path forwarding and greedy forwarding. The solid line is a power law of exponent 1.2. **b** normalized betweenness divided by number of routers in the AS, with source and destination ASs chosen with a probability proportional to the number of routers in them.

Supplementary Methods

MAPPING METHOD

To find our hyperbolic Internet map, we use the equivalence between the Einsteinian–$\mathbb{H}^2$ [40] and the Newtonian–$\mathbb{S}^1$ [41] models. This equivalence establishes a relationship between the expected degree κ of a node in the Newtonian–$\mathbb{S}^1$ model, and its radial coordinate r in the Einsteinian–$\mathbb{H}^2$ model (Eq. (7) in the main text). The angular coordinate θ is the same in both models. Thus, for a given node i we aim to find its expected degree and angular coordinate, $\{\kappa_i, \theta_i\}$, that best match the Newtonian–$\mathbb{S}^1$ model. We then use the κ-to-r mapping to place nodes in the hyperbolic plane according to the Einsteinian–$\mathbb{H}^2$ model.

Thanks to their equivalence, the Newtonian–$\mathbb{S}^1$ and Einsteinian–$\mathbb{H}^2$ models generate statistically the same network topologies. However, the efficiency of greedy forwarding in the Einsteinian–$\mathbb{H}^2$ model is higher, because hyperbolic geodesics are exceptionally congruent with the topologically shortest paths in scale-free networks [40, 42]. The reason for this congruency is that the *effective* distance used as an argument of the connection probability in the Newtonian–$\mathbb{S}^1$ is actually hyperbolic [40], and the Einsteinian–$\mathbb{H}^2$ model simply translates this effective distance to the real hyperbolic one. For these reasons we prefer the Einsteinian–$\mathbb{H}^2$ model for routing purposes, although we use the Newtonian–$\mathbb{S}^1$ one to find the Internet map. We could use directly the Einsteinian–$\mathbb{H}^2$ model for this purpose, but the Newtonian–$\mathbb{S}^1$ model is technically simpler since the statistical inference in it can be performed independently for the two variables κ and θ.

We first recall the Newtonian–$\mathbb{S}^1$ model, which generates networks according to the following steps:

1. Distribute N nodes uniformly over the circle $\mathbb{S}^1$ of radius $N/(2\pi)$, so that the node density on the circle is fixed to 1 [68].

2. Assign to all nodes a hidden variable κ representing their expected degrees. To generate scale-free networks, κ is drawn from the power-law distribution

$$\rho(\kappa) = \kappa_0^{\gamma-1}(\gamma-1)\kappa^{-\gamma}, \quad \kappa \in [\kappa_0, \infty), \tag{8}$$

$$\kappa_0 = \bar{k}\frac{\gamma-2}{\gamma-1}, \tag{9}$$

   where κ_0 is the minimum expected degree, and $\bar{k}$ is the network average degree [69].

3. Let κ and κ' be the expected degrees of two nodes located at distance $d = N\Delta\theta/(2\pi)$ measured over the circle, where $\Delta\theta$ is the angular distance between the nodes. Connect each pair of nodes with probability $p(\chi)$, where the *effective* distance $\chi \equiv d/(\mu\kappa\kappa')$, and μ is a constant fixing the average degree.

The connection probability $p(\chi)$ can be any integrable function. Here we chose the Fermi-Dirac distribution

$$p(\chi) = \frac{1}{1+\chi^\beta}, \tag{10}$$

where $\beta = 1/T$ is a parameter that controls clustering in the network. With this connection probability, parameter μ becomes

$$\mu = \frac{\beta}{2\pi\bar{k}}\sin\left[\frac{\pi}{\beta}\right]. \tag{11}$$

The expected degree of a node with hidden variable κ is $\bar{k}(\kappa) = \kappa$ and, therefore, the degree distribution scales as $P(k) \sim k^{-\gamma}$ for large k.

To go from the Newtonian–$\mathbb{S}^1$ to the Einstenian–$\mathbb{H}^2$ models, we leave the angular coordinate θ unchanged, while the radial coordinate of a node with expected degree κ is given by

$$r = R - 2\ln\frac{\kappa}{\kappa_0}, \tag{12}$$

where the radius of the hyperbolic disk containing all nodes is

$$R = 2\ln\left[\frac{N}{\pi\mu\kappa_0^2}\right]. \tag{13}$$

General theory behind likelihood maximization

We now fit the real AS graph to the model. Specifically, given the measured AS graph, we aim to find the set of coordinates $\{\kappa_i, \theta_i\}$, $i = 1, \cdots, N$, that best match the Newtonian–$\mathbb{S}^1$ model in a statistical sense. To do so, we use maximum likelihood estimation (MLE) techniques. Let us compute the posterior probability, or likelihood, that a network given by its adjacency matrix a_{ij} is generated by the Newtonian–$\mathbb{S}^1$ model, $\mathcal{L}(a_{ij}|\gamma, \beta, \bar{k})$. This probability is

$$\mathcal{L}(a_{ij}|\gamma, \beta, \bar{k}) = \int \cdots \int \mathcal{L}(a_{ij}, \{\kappa_i, \theta_i\}|\gamma, \beta, \bar{k}) \prod_{i=1}^{N} d\theta_i d\kappa_i, \tag{14}$$

where function $\mathcal{L}(a_{ij}, \{\kappa_i, \theta_i\}|\gamma, \beta, \bar{k})$ within the integral is the joint probability that the model generates the adjacency matrix a_{ij}, and the set of hidden variables $\{\kappa_i, \theta_i\}$. Using Bayes' rule, we find the likelihood that the hidden variables take particular values $\{\kappa_i, \theta_i\}$ in the network given by its observed adjacency matrix a_{ij}

$$\mathcal{L}(\{\kappa_i, \theta_i\}|a_{ij}, \gamma, \beta, \bar{k}) = \frac{\mathcal{L}(a_{ij}, \{\kappa_i, \theta_i\}|\gamma, \beta, \bar{k})}{\mathcal{L}(a_{ij}|\gamma, \beta, \bar{k})} = \frac{\text{Prob}(\{\kappa_i, \theta_i\})\mathcal{L}(a_{ij}|\{\kappa_i, \theta_i\}, \gamma, \beta, \bar{k})}{\mathcal{L}(a_{ij}|\gamma, \beta, \bar{k})}, \tag{15}$$

where

$$\text{Prob}(\{\kappa_i, \theta_i\}) = \frac{1}{(2\pi)^N} \prod_{i=1}^{N} \rho(\kappa_i) \tag{16}$$

is the prior probability of the hidden variables given by the model,

$$\mathcal{L}(a_{ij}|\{\kappa_i, \theta_i\}, \gamma, \beta, \bar{k}) = \prod_{i<j} p(\chi_{ij})^{a_{ij}} [1 - p(\chi_{ij})]^{1-a_{ij}} \tag{17}$$

is the likelihood of finding a_{ij} if the hidden variables are $\{\kappa_i, \theta_i\}$, and

$$\chi_{ij} = \frac{N\bar{k}\Delta\theta_{ij}}{\beta \sin(\pi/\beta)\kappa_i\kappa_j}, \tag{18}$$

$$\Delta\theta_{ij} = \pi - |\pi - |\theta_i - \theta_j||. \tag{19}$$

The MLE values of the hidden variables $\{\kappa_i^*, \theta_i^*\}$ are then those that maximize the likelihood in Eq. (15) or, equivalently, its logarithm,

$$\ln \mathcal{L}(\{\kappa_i, \theta_i\}|a_{ij}, \gamma, \beta, \bar{k}) = C - \gamma \sum_{i=1}^{N} \ln \kappa_i + \sum_{i<j} a_{ij} \ln p(\chi_{ij}) + \sum_{i<j} (1 - a_{ij}) \ln [1 - p(\chi_{ij})], \tag{20}$$

where C is independent of κ_i and θ_i.

MLE for expected degrees κ

The derivative of Eq. (20) with respect to expected degree κ_l of node l is

$$\frac{\partial}{\partial \kappa_l} \ln \mathcal{L}(\{\kappa_i, \theta_i\}|a_{ij}, \gamma, \beta, \bar{k}) = -\frac{\gamma}{\kappa_l} - \frac{\beta}{\kappa_l} \left(\sum_{j \neq l} p(\chi_{lj}) - \sum_j a_{lj} \right). \tag{21}$$

The first term within the parenthesis is the expected degree of node l, while the second term is its actual degree k_l. Therefore, the value κ_l^* that maximizes the likelihood is given by

$$\bar{k}(\kappa_l^*) = \kappa_l^* = k_l - \frac{\gamma}{\beta}. \tag{22}$$

Since κ_l^* can be smaller than κ_0 in the last equation, we set

$$\kappa_l^* = \max\left(\frac{\gamma-2}{\gamma-1}\bar{k}, k_l - \frac{\gamma}{\beta}\right). \tag{23}$$

MLE for angular coordinates θ

Having found the MLE values for expected degrees κ, we now have to maximize Eq. (15) with respect to angular coordinates θ. This task is equivalent to maximizing the partial log-likelihood

$$\ln \mathcal{L}(a_{ij}|\{\kappa_i^*, \theta_i\}, \gamma, \beta, \bar{k}) = \sum_{i<j} a_{ij} \ln p(\chi_{ij}) + \sum_{i<j}(1-a_{ij}) \ln\left[1-p(\chi_{ij})\right]. \tag{24}$$

The first term in this equation involves only pairs of connected nodes, whereas the second term accounts for pairs of disconnected ones. Since the connection probability $p(\chi)$ is a monotonously decreasing function of the effective distance χ, the first term in Eq. (24) is large if pairs of connected nodes are placed close to each other. In contrast, the second term is large if pairs of disconnected nodes are far apart. Therefore the optimal MLE solution will balance both effects, and place connected nodes as close as possible while keeping disconnected ones as far as possible.

Unfortunately, the maximization of Eq. (24) with respect to the angular coordinates cannot be performed analytically. We thus have to rely on approximations. At their core lie MLE algorithms, or kernels, which we discuss first. We present two such kernels, standard Metropolis-Hastings (SMH) [43], and our "localized" version of it (LMH).

MLE kernels

In the **standard Metropolis-Hastings (SMH)** algorithm, a node is chosen at random, and given a new angular position chosen uniformly in the interval $[0, 2\pi]$. The change is accepted whenever the likelihood in Eq. (24) computed after the change, $\mathcal{L}_{new}$ [70], is larger than the likelihood computed with the old coordinate, $\mathcal{L}_{old}$. Otherwise, the change is accepted with probability $\mathcal{L}_{new}/\mathcal{L}_{old}$. The SMH algorithm samples the angular phase space, and produces angular configurations with a probability proportional to the likelihood. The SMH's computational complexity depends on a particular system to which SMH is applied. We find that in our case the number of node moves sufficient for SMH to converge is $\mathcal{O}(N^2)$, meaning that the total running time complexity is $\mathcal{O}(N^3)$, since each move attempt involves the $\mathcal{O}(N)$ computation of the likelihood change.

Our **localized Metropolis-Hastings (LMH)** algorithm is not MH *per se*. In fact it bears stronger resemblances to extremal optimization and genetic search algorithms than to MH. We first define the local contribution $\ln \mathcal{L}_i$ of node i to the total log-likelihood $\ln \mathcal{L}$ in Eq. (24):

$$\ln \mathcal{L}_i = \sum_{j\neq i} a_{ij} \ln p(\chi_{ij}) + \sum_{j\neq i}(1-a_{ij}) \ln\left[1-p(\chi_{ij})\right], \tag{25}$$

so that $\ln \mathcal{L} = 1/2 \sum_i \ln \mathcal{L}_i$. We can interpret function $\ln \mathcal{L}_i$ as the fitness of node i, which we can then use to maximize the total likelihood. Specifically, in LMH nodes are visited in rounds, and during each round all nodes are visited one by one. At each node visit, the node is moved to the angular position that maximizes its fitness $\ln \mathcal{L}_i$, having fixed the positions of all other nodes at that particular node visit. An example of the log-likelihood landscape that a node sees during its move is shown in the top plot of Supplementary Fig. S1. The total number of rounds of all-node visits needed for LMH to converge is of the order of the network average degree. Indeed, even though after each node move, the fitness of other nodes changes, the node fitness is mostly affected by changes of coordinates of the node neighbors, whose average number thus roughly determines the number of rounds. The maximization of the fitness of a node takes $\mathcal{O}(N^2)$ time, having fitness $\ln \mathcal{L}_i$ sampled at intervals with $\Delta\theta = 1/N$. Therefore for sparse graphs, the overall computational complexity of LMH is $\mathcal{O}(N^3)$.

Applied to the real Internet and synthetic Internet-like networks below, both SMH and LMH yield similar good results. However, we prefer LMH since by its localized nature, it can be implemented in a distributed manner, an important property for deployment in the real Internet. Even more importantly, with LMH, new-coming ASs can compute their coordinates in a distributed manner *without knowing the global Internet topology*. Indeed, to compute its coordinates using Eq. (25), a new-coming AS i has to know the status of connections only to its neighbors; the status of connections between any two ASs other than i does not contribute to $\ln \mathcal{L}_i$ in Eq. (25). All results shown in this paper are for LMH.

First MLE wrapper

If we naïvly applied any MLE kernel to the Internet, we would have to wait forever for good results. We see in the top plot of Supplementary Fig. S1 that the characteristic likelihood profile has abundant local maxima. Therefore an MLE kernel is not guaranteed to converge to the global maximum in a reasonable amount of time. It is thus imperative to find a heuristic procedure, i.e., an MLE wrapper, helping an MLE kernel to find its way towards the global maximum without being trapped in local maxima. This procedure is equivalent to using all available information to make an educated guess of the initial node coordinates.

Our MLE wrapping strategy is based on statistical independence of edges in our graphs. Thanks to this independence, the coordinates of a set of nodes can be inferred based only on the partial information contained in a subgraph formed by the nodes in the set, ignoring the rest of the network. Consider a small subgraph of the whole network, for our purposes made of high degree nodes, and remove all nodes and connections not belonging to this subgraph. Since edges in this subgraph are statistically independent of other edges, we can maximize the likelihood corresponding to the subgraph, and infer the coordinates of the nodes in it based only on this partial information. If the subgraph is small and dense enough, finding the optimal MLE solution is easy. Once this solution is found, we can add more nodes to the network, and use the previously inferred coordinates as the initial configuration for the new MLE problem. However, this method works only if the subgraph forms a single connected component. As explained in the main text, this property holds for synthetic networks in our model, and for the real Internet [41].

Formally, let $k_1, k_2, \cdots, k_m$, with $k_1 > k_2 > \cdots > k_m$, be a set of predefined degrees, and let $\mathcal{G}(k_l)$, $l = 1, \cdots, m$, be the subgraphs formed by all nodes of degrees larger or equal to k_l, plus all connections among them. We thus have $\mathcal{G}(k_1) \subset \mathcal{G}(k_2) \subset \cdots \subset \mathcal{G}(k_m)$, forming a hierarchy of nested subgraphs. The main idea behind our MLE wrapper is to run the MLE kernel, either SMH or LMH, in iterations, starting with the smallest subgraph, and feeding the coordinates inferred at each iteration to the MLE kernel at the next iteration.

This idea must be implemented with care. First, subgraph $\mathcal{G}(k_1)$ is different from other subgraphs. Indeed, in scale-free networks, all nodes of degrees larger than $\sim N^{1/2}$ are connected almost surely. Therefore all such nodes would appear identical to the MLE kernel, which would thus place them all at the same location, something that we have to avoid. To solve this problem, we remove all connections among nodes of degree larger or equal to $k_1 \sim N^{1/2}$ and start the wrapper algorithm with the $\mathcal{G}(k_2)$ iteration. Second, iterating from $\mathcal{G}(k_l)$ to $\mathcal{G}(k_{l+1})$, we still need to specify the initial coordinates of the nodes that belong to $\mathcal{G}(k_{l+1})$ but not to $\mathcal{G}(k_l)$ [71]. While the assignment of random coordinates to new nodes is possible, it is much more efficient to try to maximize the likelihood in $\mathcal{G}(k_{l+1})$ from the very beginning. In other words, we assign to each new node $i \in \mathcal{G}(k_{l+1}) \setminus \mathcal{G}(k_l)$ the coordinate maximizing

$$\ln \mathcal{L}_i \left[\mathcal{G}(k_l)\right] = \sum_{j \in \mathcal{G}(k_l)} a_{ij} \ln p(\chi_{ij}) + \sum_{j \in \mathcal{G}(k_l)} (1 - a_{ij}) \ln \left[1 - p(\chi_{ij})\right] \tag{26}$$

We note that node i uses information contained only in $\mathcal{G}(k_l)$ to get its initial coordinate. After all new nodes corresponding to a given iteration are introduced and assigned initial coordinates, we apply the MLE kernel to the resulting system. This heuristic MLE wrapping procedure is summarized in Alg. 1.

Algorithm 1 First MLE wrapper

```
activate nodes in G(k_1)
assign random angular coordinates to nodes in G(k_1)
remove links among nodes in G(k_1)
for l = 2 to (# of layers) do
    for j = 1 to (# of nodes in G(k_l) not active) do
        i ← label of new node in G(k_l) not active
        if # of connections of i with nodes in G(k_{l-1}) ≥ 2 then
            activate new node i
            assign to node i coordinate θ_i maximizing ln L_i [G(k_{l-1})]
        end if
    end for
    run the MLE kernel on the set of active nodes
end for
```

In the bottom plot in Supplementary Fig. S1 we show the test results for this procedure wrapping the LMH kernel, applied to a synthetic Newtonian–$\mathbb{S}^1$ network generated with the parameters similar to the real AS graph. We observe that the inferred coordinates are very close to the real ones, except for a global phase shift, which can take any value in $[0, 2\pi]$ due to the rotational symmetry of the model.

Second MLE wrapper

As mentioned above, it is not necessary to consider the full graph to infer the coordinates of the most connected nodes. We now use this observation to speed up the mapping process significantly. Specifically, we run our first MLE wrapper up to a subgraph of a certain size, and then add the rest of the nodes assigning to them their coordinates maximizing Eq. (26) *without* subsequent running the MLE kernel, see Alg. 2.

Algorithm 2 Second MLE wrapper

```
activate nodes in G(k_1)
assign random angular coordinates to nodes in G(k_1)
remove links among nodes in G(k_1)
l_critic ← maximum layer with full MLE calculations
for l = 2 to (# of layers) do
    for j = 1 to (# of nodes in G(k_l) not active) do
        i ← label of new node in G(k_l) not active
        if # of connections of i with nodes in G(k_{l-1}) ≥ 2 then
            activate new node i
            assign to node i coordinate θ_i maximizing ln L_i [G(k_{l-1})]
        end if
    end for
    if l ≤ l_critic then
        run the MLE kernel on the set of active nodes
    end if
end for
```

This modification speeds the overall mapping process because once the coordinates of the coordinates of a relative small number of high degree nodes are inferred, the rest of the process takes $\mathcal{O}(N^2)$ steps to complete. This improvement reduces the total running time of the Internet mapping to few hours on a standard computer. Another practically important feature of this second MLE wrapper is that new-coming ASs compute their coordinates *without existing ASs changing their coordinates.* In other words, once the AS coordinates are inferred, they stay static as the Internet grows.

We apply this procedure up to nodes of degree 3. Nodes of degree 2 and 1 must be analyzed separately since all nodes of degree 1, and 40% of nodes of degree 2 do not form any triangles. As a consequence, the MLE kernel cannot reliably infer their metric attributes, i.e., their coordinates. Therefore we assign to these nodes the angular coordinate of their (highest-degree) neighbors, which makes sense, especially for nodes of degree 1, since the only path to such nodes is via their neighbors. Forwarding to such nodes is thus equivalent to forwarding to their neighbors.

The test results of this second MLE wrapper are shown in Supplementary Fig. S2. The top left plot shows the inferred vs. real coordinates in the same synthetic network for nodes with degrees $k \geq 20$ using the first MLE wrapper. The other plots show the corresponding coordinates for nodes with degrees larger than or equal to $8, 6, 5, 4, 3$ using the second MLE wrapper with $k_{l_{critic}} = 20$. That is, the MLE kernel is not run for these nodes. We observe that the inference quality does deteriorate for smaller degrees, but it is remarkable that even in the worst case a majority of coordinates are correctly inferred.

As an additional test, we show in Supplementary Fig. S3 the empirical connection probability among nodes in each subgraph using the coordinates inferred by the second MLE wrapper, compared to the connection probability $p(x) = (1 + x^2)^{-1}$ used to generate the network. We observe a good agreement for high degree subgraphs, which slightly deteriorates for low degree nodes located at large effective distances χ.

To map the AS graph, we used the LMH kernel wrapped with the second MLE wrapper with $k_{l_{critic}} = 20$ and the subgraph hierarchy defined by $k_1 = 300$, $k_2 = 200$, $k_3 = 160$, $k_4 = 130$, $k_5 = 110$, $k_6 = 100$, $k_7 = 90$, $k_8 = 80$, $k_9 = 70$, $k_{10} = 60$, $k_{11} = 50$, $k_{12} = 40$, $k_{13} = 30$, $k_{14} = 20$, $k_{15} = 10$, $k_{16} = 9$, $k_{17} = 8$, $k_{18} = 7$, $k_{19} = 6$, $k_{20} = 5$, $k_{21} = 4$, $k_{22} = 3$.

Parameter estimation and finite size effects

Our model has three parameters: the exponent γ of the degree distribution, the average degree $\bar{k}$, and the exponent β of the connection probability.

Estimating γ

We estimate the exponent γ via the direct inspection of the degree distribution, yielding $\gamma = 2.1$.

Estimating N and $\bar{k}$

The estimation of N and $\bar{k}$ is more involved for two reasons. First, the Newtonian–$\mathbb{S}^1$ model generates nodes of zero degree which are included in the computation of the average degree, $\bar{k} = \sum_{k=0} kP(k)$. However, in the real Internet graph all nodes have non-zero degrees. Therefore we first have to estimate the number of nodes N in the model, based on the number of nodes N_{obs} we observe in the real graph. The relationship between the two numbers is

$$N = \frac{N_{obs}}{1 - P(0)}, \tag{27}$$

where $P(0)$ is the probability that a node has zero degree in the model.

The second complication is due to finite size effects. These effects are particularly important when the exponent γ is close to 2, which is the case with the Internet. Suppose we generate a finite size network of N nodes with our Newtonian–$\mathbb{S}^1$ model with parameters γ, $\bar{k}$, and β. Since the network is finite, there is a cut-off value for the expected degree of a node, κ_c, which depends on the size of the network. The first moment of the distribution of expected degrees $\rho(\kappa) = \kappa_0^{\gamma-1}(\gamma-1)\kappa^{-\gamma}$ with this cut-off is

$$\langle \kappa(N) \rangle = \bar{k} \left(1 - \left[\frac{\kappa_0}{\kappa_c} \right]^{\gamma-2} \right) \equiv \bar{k}\alpha(\kappa_0, \kappa_c). \tag{28}$$

In the thermodynamic limit $\kappa_c \to \infty$ and $\alpha(\kappa_0, \kappa_c) \to 1$. However, if γ is close to 2, the approach to these limits is slow, and we have to take care of finite size corrections.

Accounting for these corrections, the expected degree of a node with hidden variable κ becomes

$$\bar{k}_N(\kappa) = \alpha(\kappa_0, \kappa_c)\kappa, \tag{29}$$

with $\bar{k}_\infty(\kappa) = \kappa$. This equation implies that the MLE of the hidden variable κ of a node of degree k changes from Eq. (23) to

$$\kappa^* = \max\left(\frac{\gamma-2}{\gamma-1}\bar{k}, \frac{1}{\alpha(\kappa_0, \kappa_c)} \left[k - \frac{\gamma}{\beta} \right] \right), \tag{30}$$

while the average degree including zero degree nodes in a finite size network becomes

$$\bar{k}_N = [\alpha(\kappa_0, \kappa_c)]^2 \bar{k}. \tag{31}$$

If the average degree observed in the real Internet graph is $\bar{k}_{obs}$, our estimate of the parameter $\bar{k}$ is then

$$\bar{k} = \frac{1 - P(0)}{[\alpha(\kappa_0, \kappa_c)]^2} \bar{k}_{obs}. \tag{32}$$

Therefore, in order to estimate the values of N and $\bar{k}$ for a finite network, we first have to estimate the values of $P(0)$ and $\alpha(\kappa_0, \kappa_c)$. One can check [41] that

$$P(0) = (\gamma-1)[\alpha(\kappa_0, \kappa_c)\kappa_0]^{\gamma-1}\Gamma(1-\gamma, \alpha(\kappa_0, \kappa_c)\kappa_0), \tag{33}$$

where $\Gamma(x, y)$ is the incomplete Gamma function. We can also relate the maximum degree $k_{obs}^{\max}$ observed in the real Internet to the expected degree cut-off κ_c via

$$k_{obs}^{\max} = \alpha(\kappa_0, \kappa_c)\kappa_c. \tag{34}$$

We thus have six unknown values—namely, N, $P(0)$, κ_0, κ_c, $\alpha(\kappa_0, \kappa_c)$, and $\bar{k}$—and the system of six equations (9,27,28,32,33,34) involving them. Substituting into these equations the given values of $N_{obs} = 23752$,

$\bar{k}_{obs} = 4.92$, $k_{obs}^{\max} = 2778$, and $\gamma = 2.1$ observed in the Internet, we compute numerically $\kappa_0 = 0.9$, $\kappa_c = 4790$, $\alpha(\kappa_0, \kappa_c) = 0.58$, and $P(0) = 0.33$, yielding $N = 35685$ and $\bar{k} = 9.86$.

Estimating β

To estimate β, we first compare clustering in synthetic networks with different β's to the clustering observed in the Internet, keeping all other parameters fixed. This procedure narrows down the possible values of β to $\beta \in [1, 2]$. We then generate Internet maps for different values of β within this range, and perform hyperbolic greedy forwarding in them. Supplementary Fig. S4 shows the success ratio of greedy forwarding as a function of β in this region. We observe that the success ratio increases as β decreases, and then sharply drops at $\beta \sim 1$. The value of β maximizing the success ratio is $\beta = 1.45$, and we used this value in our final Internet map.

DEALING WITH NEW-COMING ASs

In the main text we replay the AS Internet growth from January 2007 to June 2009 similar to [44]. Specifically, we obtain 11 lists of ASs observed in the Internet at different dates as described in [44]. The AS lists are linearly spaced in time with the interval of three months: time $t = 0$ corresponds to January 2007, $t = 1$ is April 2007, and so on until $t = 10$, June 2009. We denote the obtained AS lists by A_t. The number of ASs in A_0 is 17258, while the numbers of new ASs in $A_{t'}$ with $t' = 1, 2, \ldots, 10$, but not in A_0 are 806, 1614, 2389, 3103, 3973, 4794, 5434, 5843, 6207, and 6426. We then take our Archipelago AS topology [45] of June 2009, and for each $t = 0, 1, \ldots, 10$ we remove from it all ASs and their adjacent links that are not in A_t, thus obtaining a time series of historical AS topologies G_t. We then embed G_0 using the SMH kernel (24), but for each subsequent embedding of $G_{t'}$ with $t' > 0$, we keep the hyperbolic coordinates of ASs in G_t with $t < t'$ fixed, and compute coordinates for the new ASs using the LMH kernel (25). That is, once an AS appears at some time $t \geqslant 0$ and gets its coordinates computed, using either the SMH, $t = 0$, or LMH, $t > 0$, computations, the AS then never changes its coordinates for the rest of the observation period. In Fig. 7 of the main text, we show the average success ratio p_s and stretch $\bar{s}$ for greedy forwarding in G_t.

Remarkably, we observe only minor variations of success ratio and stretch over more than 2.5 years of rapid Internet growth. The success ratio does decrease, but by less than 1%. We thus conclude that greedy forwarding using our hyperbolic AS map is quite robust with respect to Internet historical growth. Existing ASs do not have to recompute their hyperbolic coordinates when new ASs join the Internet. Recomputations of all AS coordinates may be executed to improve the greedy forwarding performance, but the time scale for such recomputations exceeds the time scale of Internet historical evolution, i.e., years, thus exceeding by orders of magnitude the time scale of transient dynamics of failing AS links and nodes, i.e., seconds or minutes. That is why the existing AS coordinates are essentially static, and can stay the same for years.

SENSITIVITY TO MISSING LINKS

It is widely known that the existing measurements of the Internet topology miss a number of AS links [46–48]. However, in view of the robustness of greedy forwarding performance with respect to link removals, one could expect that its performance would be robust with respect to missing links as well. Furthermore, if our hyperbolic map is used in practice, then greedy forwarding will see and use those links that we do not see. Therefore one can intuitively expect that, in this case, the efficiency of greedy forwarding will be actually higher than reported in the main text, simply because these links that we miss but greedy forwarding would not miss, would provide additional shortcuts between potentially remote ASs. If so, the routing results presented in the main text should be considered as lower bounds.

To confirm this intuition, we perform the following experiment. It is known that the majority of missing links in the Internet are peer-to-peer links among provider ASs of moderate size [47, 48]. To emulate the missing link issue, we thus remove a fraction (ranging from 0% to 30%) of links among nodes with degree above a certain threshold ($k = 5$) from our AS graph. We then map these graphs with different numbers of emulated missing links to $\mathbb{H}^2$ as described in the Mapping Method section to find hyperbolic coordinates for each AS. Using these maps with missing information, we then consider two different greedy forwarding scenarios for each map:

1. In the first scenario, we navigate an AS graph mapped with a fraction of links removed, and compute the success ratio of greedy forwarding in the graph. This scenario tries to mimic the missing links issue directly. We have incomplete topology measurements of the real Internet, but we have no other option as to use these measurements to map the Internet to its hyperbolic space, and study navigability with this map, which we know miss some information.

2. In the second scenario, we use the hyperbolic map obtained with missing links, but we then add back those removed links, and navigate the complete graph. This scenario is motivated by the observation that even though our map is constructed with some links missing, these missing links will still be used by ASs attached to them to forward information if this map is used in practice.

The results of these two scenarios are shown in Fig. 8 of the main text. As intuitively expected, our mapping is quite robust with respect to missing links: the success ratio decreases by less than 5% even if up to 14% of links are removed from the topology before we map it. Also as expected, the missing links, when added back, increase the success ratio. That is, even though the map has been constructed using partial information, navigability improves when missing links are considered. These results confirm that the routing results reported in the main text are in reality lower bounds for the success ratio that can be achieved if our map is used in practice. In fact, one may somewhat paradoxically expect that the more links are missed in the measured Internet topology we used for mapping, the better the success ratio would be in practice, since according to Fig. 8, the success ratio improvement due to re-adding of removed links tends to increase with the number of removed links.

MAPPING ASs TO COUNTRIES

The AS-to-country mapping is taken from the CAIDA AS ranking project [47]. It uses two methods for this task. The first method is *IP-based*. It splits the IP address space advertised by an AS into small blocks, and then maps each block to a country using [72]. If not all IP blocks of an AS map to the same country, then the other, *WHOIS-based* method is used, which reports the country where the AS headquarters are located according to the WHOIS database [73]. Since large ASs have points of presence in many countries, they tend to map to multiple countries using the IP-based method. Therefore, if we did not apply the WHOIS-based method to them, they would no longer map to a single country. If we ignored such ASs, the angular distributions of the remaining ASs belonging to a given country would be even more localised, including the US, EU, and UK ASs. In our hyperbolic map data, we release the AS-to-country associations using both methods, IP+WHOIS-based and IP-based. The latter has no country information for many ASs with conflicting country mappings.

GEOGRAPHIC ROUTING

To perform standard geographic routing we first map each AS to a collection of geographic locations (characterised by their latitudes and longitudes) using the IP-based method, and then find the centre of mass for each collection. We thus obtain unique geographic coordinates for each AS. We then perform standard greedy forwarding over the AS topology, computing geographic distances between ASs using the spherical law of cosines. For hyperbolised geographic routing, we keep the AS angular coordinates equal to their geographic coordinates, but also, based on the AS degree, we assign to each AS a radial coordinate, according to the relationship between node degrees and radial positions in the three-dimensional Einsteinian model, and then perform greedy forwarding in this three-dimensional hyperbolic space.

TRAFFIC AND CONGESTION CONSIDERATIONS

We measure a proxy for the amount of traffic that ASs would handle under greedy forwarding. Since greedy forwarding almost always follows shortest paths, we expect the traffic load on an AS under greedy forwarding be almost the same as under shortest path forwarding, which we confirm in Supplementary Fig. S5 where we juxtapose the normalized betweennesses of shortest path and greedy forwarding. To compute those, we select a large number of source/destination AS pairs chosen uniformly at random among all ASs. We then find two paths for each AS pair using shortest path and greedy forwarding. Normalized betweenness of a given AS is then the fraction of all paths going through this AS. We observe in Supplementary Fig. S5 that the normalized betweennesses for shortest path and greedy forwarding are almost identical as expected. We also observe in the top plot, that in agreement with the previous studies on this subject, e.g. [64], the normalized betweenness grows as a power law of the AS degree. This observation may create an impression that high-degree ASs may suffer from traffic congestion problems. However, this impression is wrong not only because of the results in [65], but also because of the following considerations.

In the real Internet, ASs are not singular nodes but differently sized networks composed of (many) routers. The size of an AS, measured by the number of routers in it, is roughly proportional to the AS degree [66, 67]. ASs of different size generate and consume different volumes of traffic. Also, a larger AS can handle larger transit traffic volumes without being congested. These two observations suggest the following modifications to the top plot in Supplementary Fig. S5. First, we model traffic with the more realistic assumption that the amount of traffic an AS generates or consumes is proportional to the AS size. That is, instead of choosing source and destination AS pairs at random, we chose each AS with a probability proportional to the number of routers in the AS using the data from [67]. Second, we divide the normalized betweenness value for each AS by the number of routers in the AS, thus estimating the per-router traffic load. The result shown in the bottom plot of Supplementary Fig. S5 demonstrates that the important large ASs are, in fact, less prone to congestion problems.

Supplementary Notes

AS-LEVEL ROUTING

Our approach belongs to a wide class of approaches proposing to reduce routing granularity to the level of Autonomous Systems [49–61]. The key difference between ours and the existing approaches in this class is that the latter require some form of routing on the dynamic AS graph. As soon as the AS topology changes, new AS routes must be recomputed, so that routing communication overhead is unavoidable in this case. In our case such recomputations are not needed since as we show in the main text, the efficiency of greedy forwarding sustains in presence of failing AS nodes and links, even though ASs do not exchange any information about topology modifications, and do not change their hyperbolic coordinates, i.e., even though they do not incur any communication overhead. A bulk of routing overhead in the Internet today is due to traffic engineering and multihoming in the first place [62, 63]. How the AS-level routing class of approaches helps to deal with and reduce this overhead is discussed in the literature cited above.

Supplementary References

[40] D. Krioukov, F. Papadopoulos, A. Vahdat, and M. Boguñá, Phys Rev E **80**, 035101(R) (2009).
[41] M. Á. Serrano, D. Krioukov, and M. Boguñá, Phys Rev Lett **100**, 078701 (2008).
[42] D. Krioukov, F. Papadopoulos, M. Boguñá, and A. Vahdat, ACM SIGMETRICS Perf E R **37**, 15 (2009).
[43] M. E. J. Newman and G. T. Barkema, *Monte Carlo Methods in Statistical Physics* (Clarendon Press, Oxford, 1999).
[44] A. Dhamdhere and K. Dovrolis, in *IMC* (2008).
[45] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, in *CATCH* (2009), `http://www.caida.org/projects/ark/`.
[46] A. Lakhina, J. Byers, M. Crovella, and P. Xie, in *INFOCOM* (2003).
[47] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, kc claffy, and G. Riley, Comput Commun Rev **37**, 29 (2007), `http://as-rank.caida.org/`.
[48] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang, IEEE ACM T Network (in press) (2010).
[49] I. Castineyra, N. Chiappa, and M. Steenstrup, *The nimrod routing architecture*, IETF, RFC 1992 (1996).
[50] R. Hinden, *New scheme for Internet routing and addressing (ENCAPS) for IPNG*, IETF, RFC 1955 (1996).
[51] F. Kastenholz, *ISLAY: A new routing and addressing architecture*, IRTF, Internet Draft (2002).
[52] P. Verkaik, A. Broido, kc claffy, R. Gao, Y. Hyun, and R. van der Pol, Technical Report TR-2004-1, CAIDA (2004).
[53] R. Gummadi, R. Govindan, N. Kothari, B. Karp, Y.-J. Kim, and S. Shenker, in *HotNets* (2004).
[54] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica, in *SIGCOMM* (2005).
[55] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, in *SIGCOMM* (2006).
[56] D. Krioukov, kc claffy, K. Fall, and A. Brady, Comput Commun Rev **37**, 41 (2007).
[57] R. Oliveira, M. Lad, B. Zhang, and L. Zhang, in *ICNP* (2007).
[58] D. Massey, L. Wang, B. Zhang, and L. Zhang, in *ACM SIGCOMM Workshop on IPv6 and the Future of the Internet* (2007).
[59] B. Zhang, L. Zhang, and L. Wang, *Evolution towards global routing scalability*, IETF, Internet Draft (2009).
[60] D. Farinacci, V. Fuller, D. Oran, D. Meyer, and S. Brim, *Locator/ID separation protocol (LISP)*, IETF, Internet Draft (2009).
[61] C. Shue and M. Gupta, Comput Netw (in press) (2010).
[62] G. Huston, The Internet Protocol Journal **4** (2001).
[63] G. Huston, The Internet Protocol Journal **9** (2006).
[64] M. Barthélemy, Eur Phys J B **38**, 163 (2004).
[65] C. Gkantsidis, M. Mihail, and A. Saberi, in *SIGMETRICS* (2003).
[66] H. Tangmunarunkit, J. Doyle, R. Govindan, S. Jamin, W. Willinger, and S. Shenker, Comput Commun Rev **31**, 7 (2001).
[67] B. Huffaker, A. Dhamdhere, M. Fomenkov, and kc claffy, in *PAM* (2010).
[68] We chose the uniform distribution because we do not have any *a priori* preferred angular coordinate values, and thus expect the network to be isotropic.
[69] Note that the model generates nodes of zero degree that contribute to the total average degree.
[70] From now on we denote the log-likelihood in Eq. (24) by $\ln \mathcal{L}$.
[71] Adding nodes in $\mathcal{G}(k_{l+1})$ but not in $\mathcal{G}(k_l)$, we check if they have at least two connections to nodes in $\mathcal{G}(k_l)$. Otherwise we postpone introducing such nodes to the first iteration when they start satisfying this condition.
[72] Digital Envoy, Netacuity. `http://www.digital-element.net/ip_intelligence/ip_intelligence.html`
[73] WHOIS database. `http://www.whois.net/`