Decentralized routing on spatial networks with stochastic edge weights

Till Hoffmann

Astrophysics Group, Imperial College London, London, United Kingdom

Renaud Lambiotte

Naxys, University of Namur, Namur, Belgium and Department of Mathematics, University of Namur, Namur, Belgium

Mason A. Porter

Oxford Centre for Industrial and Applied Mathematics, Mathematical Institute, University of Oxford, Oxford,
United Kingdom and CABDyN Complexity Centre, University of Oxford, Oxford, United Kingdom

We investigate algorithms to find short paths in spatial networks with stochastic edge weights. Our formulation of the problem of finding short paths differs from traditional formulations because we specifically do not make two of the usual simplifying assumptions: (1) we allow edge weights to be stochastic rather than deterministic and (2) we do not assume that global knowledge of a network is available. We develop a decentralized routing algorithm that provides en route guidance for travelers on a spatial network with stochastic edge weights without the need to rely on global knowledge about the network. To guide a traveler, our algorithm uses an estimation function that evaluates cumulative arrival probability distributions based on distances between pairs of nodes. The estimation function carries a notion of proximity between nodes and thereby enables routing without global knowledge. In testing our decentralized algorithm, we define a criterion that makes it possible to discriminate among arrival probability distributions, and we test our algorithm and this criterion using both synthetic and real networks.

I. INTRODUCTION

One of the most important aspects of many networks is their navigability [1–3], and it is often important to find short paths between pairs of nodes in a network. For example, sending packages across the Internet, attempting to spread ideas through social networks, and transporting people or goods cheaply and quickly all require the ability to find paths with a small number of steps and/or a low cost [4]. Assuming that network topology and the cost of making a step is known, such paths can be found easily [5]. Unfortunately, complete knowledge of network topology (and edge weights) is often unavailable or constitutes an insurmountable overhead [6–9].

Despite the aforementioned difficulties, there is empirical evidence that some networks can be navigated by using only local information.[1] A well-known example is Milgram's small-world experiments [6], which demonstrated that short paths between individuals in social networks exist and that individuals are able to navigate networks without global knowledge of network topology. This observation was put on solid theoretical ground more than 30 years later by Kleinberg [10], who showed that one can find short paths between nodes via decentralized algorithms in certain types of spatially embedded networks [11]. This work has led to both theoretical and numerical studies of routing with limited information [4,12] as well as investigations of routing algorithms in spatially embedded networks [13–17].

---

[1] Our initial motivation for studying our formulation of this problem arose when one of the authors accidentally left his umbrella at University of Bath and had to find a way to get it back to Oxford without further travel on his part.

An important limitation of the above findings is their assumption that the cost of making a step is deterministic. In many situations, it is much more appropriate to model the cost as a random variable. For instance, when a network has varying levels of traffic [18–20], it is unsuitable to model associated costs deterministically. The aim of the present article is to address this important limitation and to develop a decentralized algorithm for routing on networks with stochastic edge weights.

The remainder of this article is organized as follows. We discuss the deterministic shortest-path problem in Sec. II and a stochastic version of this problem in Sec. III. In Sec. IV, we discuss criteria for measuring the quality of a path in a network. In Sec. V, we discuss an adaptive algorithm that will be helpful for trying to solve our stochastic shortest-path problem. We present the notion of an estimation function in Sec. VI, and we discuss our new decentralized routing algorithm in Sec. VII. We show the results of simulations on a synthetic network in Sec. VIII and on a real network in Sec. IX. We discuss our results further in Sec. X and conclude in Sec. XI. We present pseudocode for our decentralized algorithm and discuss additional technical details in appendices.

II. DETERMINISTIC SHORTEST-PATH PROBLEM

A *network* (or *graph*) G consists of a set N of *nodes* labeled by the indices $\{i_1, \ldots, i_n\}$ (and with cardinality $|N| = n$) and a set E of *edges* (with $|E| = m$) labeled by ordered pairs of indices (i,j) that indicate that there is a directed edge from node j to node i. The node j is called a *neighbor* of node i. We associate a *weight* T_{ij}, which represents a cost or travel time, with each edge (i,j). A *path* ℓ with k steps is a sequence

of $k + 1$ nodes $\ell = \{i_1, \ldots, i_{k+1}\}$ that are connected to one another by virtue of an edge between each pair of consecutive nodes in the sequence. Note that we do not require a path to be "simple," so a node can occur multiple times in a path. See Ref. [21] for a discussion of cycles in paths that arise from adaptive routing.

The weight T_ℓ of a path ℓ is given by the sum of the weights of its constituent edges:

$$T_\ell = \sum_{j=1}^{k} T_{i_j i_{j+1}}.$$

The shortest-path problem (SPP) aims to determine the path of smallest total weight from an *origin* node to a *target* (or *destination*) node. In the deterministic shortest-path problem (DSPP), each edge weight T_{ij} is deterministic, and a path with minimal total weight is called *optimal*.

III. STOCHASTIC SHORTEST-PATH PROBLEM

Nondeterministic travel times are a typical feature of transportation networks [18]. Because this is our motivating example, we use the terms "time" and "weight" interchangeably.

To define a stochastic shortest-path problem (SSSP), we let the weights T_{ij} be real-valued random variables that are distributed according to a probability distribution function (PDF) with probabilities p_{ij} [19,20,22–24]. In our SSSP formulation, we make three assumptions: (1) the random edge weights are independent of each other, (2) the PDFs do not change during the routing process, and (3) the weight incurred by traversing an edge becomes known upon completion of the step. (For example, the time taken to travel a road is known once the next junction is reached.) With these assumptions, it follows that the PDF for the weight T_ℓ of a k-step path ℓ to have a value t is given by the convolution of the PDFs of the weights associated with the path [25]:

$$p_\ell(t) = \left(\underset{j=1}{\overset{k}{*}} \, p_{i_j i_{j+1}} \right)(t), \qquad (1)$$

where the right-hand side denotes k consecutive convolutions. The probability to traverse the path ℓ and incur a weight $T_\ell \leqslant t$ is given by the cumulative distribution function (CDF):

$$U_\ell(t) = \int_0^t dt' \, p_\ell(t'). $$

IV. CRITERIA

Because the edge weights are now random variables, we need to reconsider the concept of an optimal path. In particular, there is no longer a unique concept of optimality. For example, Frank [22] defined a path to be optimal if its CDF surpasses a threshold θ within the shortest time, whereas Fan *et al.* [19] suggested maximizing the CDF for a given time budget τ. Each of these criteria has a regime in which it outperforms the other. If the budget τ available to a traveler is large, then there are many paths that result in almost certain arrival at the desired target; that is, $U_{\ell'}(\tau) \approx 1$ for many paths ℓ'. In this regime, the paths are virtually indistinguishable using Fan *et al.*'s criterion. However, Frank's criterion can easily identify the path that it deems to be optimal. In contrast, if the budget is small,
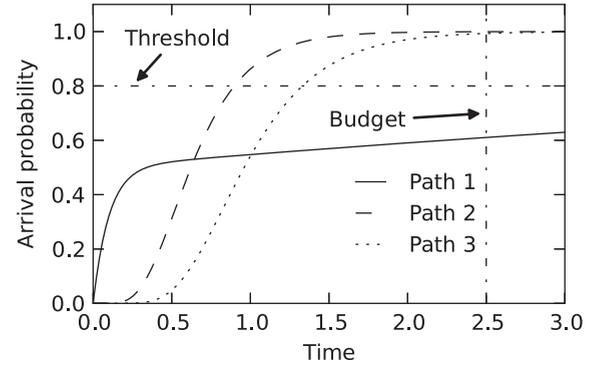

FIG. 1. Comparison of path optimality criteria using CDFs of three paths. Fan *et al.*'s criterion prefers paths 2 and 3 to path 1 but cannot discriminate between the CDFs of paths 2 and 3. Frank's criterion prefers path 2 to path 3 but cannot be applied to path 1. The joint criterion is applicable to all CDFs and selects path 2 as the optimal one.

then arrival at the target within the budget is unlikely; that is, $U_{\ell''}(\tau) \ll 1$ for all paths ℓ''. In this case, Frank's criterion is not helpful because there does not exist a path whose CDF surpasses the threshold. However, Fan *et al.*'s criterion can identify the path with the maximal CDF for the given budget.

We define a joint criterion that takes advantage of both aforementioned criteria. If there are paths whose CDFs surpass a threshold θ within the budget τ, then we choose a path according to Frank's criterion. Otherwise, we choose a path according to Fan *et al.*'s criterion. In Fig. 1, we illustrate the differences between the criteria.

V. AN ADAPTIVE ALGORITHM

Even finding approximate solutions to an SSPP is challenging. An interesting approach was proposed by Fan *et al.*, who utilized an adaptive algorithm that evaluates the available information before each step and accounts for the consequences of previous decisions [19]. They proposed building a routing table by considering the maximal probability to reach the target r from all other nodes $i \in N \backslash \{r\}$. This amounts to solving the following set of coupled nonlinear integral equations:

$$u_i(t) = \max_{j \in J_i} \left[\int_0^t p_{ij}(t') u_j(t - t') \, dt' \right],$$
$$u_r(t) = 1, \qquad (2)$$

where J_i is the set of neighbors of i and $u_i(t)$ is the probability to arrive at node r starting from node i with a total travel time that is no longer than t. The node $q_i(t)$ that one should choose to attain the maximal arrival probability is

$$q_i(t) = \arg\max_{j \in J_i} \left[\int_0^t p_{ij}(t') u_j(t - t') \, dt' \right]. \qquad (3)$$

One cannot find analytical solutions to Eqs. (2) and (3) in general, but one can approximate the CDF $u_i(t)$ using the iterative sequence

$$v_i^{s+1}(t) = \max_{j \in J_i} \left[\int_0^t p_{ij}(t') v_j^s(t - t') \, dt' \right],$$
$$v_r^{s+1}(t) = 1, \qquad (4)$$

with index s and initial conditions

$$v_i^0(t) = 0, \quad \forall i \in N \backslash \{r\}.$$

The sequences $\{v_i^s(t)\}$ give lower bounds for the true CDFs. One obtains upper bounds by using the sequences $\{w_i^s(t)\}$ with the same recursion relation (4) but with different initial conditions [26]:

$$w_i^0(t) = 1, \quad \forall i \in N.$$

In our numerical implementation, we demand that the sequences converge to within a numerical tolerance ϵ for all t. That is, we require that

$$w_i^s(t) - v_i^s(t) < \epsilon, \quad \forall i \in N$$

for sufficiently large s.

VI. ESTIMATION FUNCTION

Fan *et al.*'s algorithm is centralized, because it requires knowledge of the entire network topology and all edge-weight distributions. To build a decentralized algorithm, we define an *estimation function* $f(i,j;t)$, which gauges the arrival probability from node j to node i within time t. Such a function carries a notion of proximity between nodes, and we will use it to guide travelers on a network. We define an estimation function using the following four steps.

First, we embed the network under consideration in a metric space by defining a distance measure $d : N \times N \rightarrow \mathbb{R}$ for any pair of nodes. We use the shorthand notation $d_{ij} \equiv d(i,j)$ to denote the *distance* between any pair of nodes i and j in the metric space. The *length* of an edge is equal to the distance between a pair of nodes with a direct connection (i.e., an edge) between them. Note that the length of an edge (e.g., the length of a road) is different from its weight (e.g., the travel time).

Second, we define the *network distance* g_{ij} as the shortest distance between nodes if travelers are restricted to move along edges. Note that the network distance is distinct from the travel time. We assume that network distance between nodes i and j can be estimated from the (metric) distance between i and j. That is, we assume that there exists a function h such that $g_{ij} \approx h(d_{ij})$. Such an assumption is implicit in all decentralized algorithms using a metric for guidance.

Third, we note that the expected number of steps necessary to reach node i from node j is

$$\bar{k}_{ij} = \left\lceil \frac{g_{ij}}{\lambda} \right\rceil \approx \left\lceil \frac{h(d_{ij})}{\lambda} \right\rceil,$$

where λ denotes a characteristic edge length of a network and $\lceil x \rceil$ (which is called the "ceiling" of x) denotes the smallest integer that is at least as large as x.

Fourth, we assume that the weight $\bar{t}$ incurred by making a step towards the target is representative of the network and has a PDF $\bar{p}(t)$. We estimate the weight of the unknown path $\bar{\ell}$ from j to i to be

$$t_{\bar{\ell}} = \sum_{k=1}^{\bar{k}_{ij}} \bar{t}.$$

Using Eq. (1), we obtain an estimate

$$f(i,j;t) = \begin{cases} \int_0^t dt' \left(\overset{\bar{k}_{ij}}{\underset{k=1}{*}} \bar{p} \right)(t'), & \text{if } i \neq j, \\ 1, & \text{if } i = j \end{cases}$$

of the CDF. We thereby use physical distance to evaluate the number of steps between two nodes, and we assume that the random weight associated with each edge is uncorrelated with the length of the edge. In Sec. VIII, we discuss different choices of the characteristic edge length λ and the characteristic distribution $\bar{p}(t)$. In Appendix B, we show that the order of carrying out mixtures and convolutions is irrelevant.

VII. DECENTRALIZED ALGORITHM

Our algorithm explores a network by using local information, and it chooses a locally optimal node according to one of the criteria discussed in Sec. IV. The visited nodes N_V and the "frontier nodes" N_F constitute a known subgraph G_K (i.e., the parts of the graph that the traveler has discovered). Frontier nodes are neighbors of visited nodes but have not yet been visited themselves. The known subgraph includes all edges of G that are connected to the visited nodes N_V. Importantly, naively stepping towards a node that is a locally optimal choice without incorporating information about the journey to date can trap a traveler in a dead end. Developing an algorithm with knowledge of G_K enables a traveler to navigate out of dead ends.

In this local approach, we build on Fan *et al.*'s algorithm [19,26] and apply it to G_K by changing the initial conditions of the sequences $\{v_i^s(t)\}$ and $\{w_i^s(t)\}$ for frontier nodes using the estimation function:

$$v_j^0(t) = w_j^0(t) = f(j,r;t), \quad \forall j \in N_\text{F}.$$

We initialize nodes that have been visited in the same manner as before (so $w_j^s \geqslant v_j^s$ is satisfied for all known nodes $j \in N_\text{F} \cup N_\text{V}$). We iterate the two sets of sequences until they converge to within a chosen tolerance ϵ. The traveler subsequently moves to a successor node that is identified by one of the criteria. We then update G_K and reduce the remaining budget by the weight incurred by making the step. We repeat this process until the traveler reaches the target or the budget is exhausted. Figure 2 illustrates this routing process on a lattice.

We require $w_i^k(t)$ to provide upper bounds for the CDFs. In centralized routing, such an upper bound is equal to 1 because
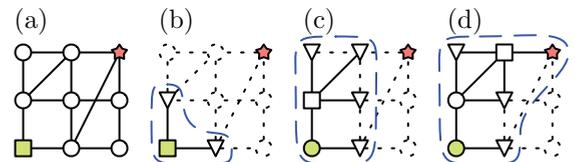

FIG. 2. (Color online) (a) A lattice with the origin node represented by a square and the target node represented by a star. (b) The data available to a decentralized algorithm before the first step. We show a network traveler's current node as a square and the frontier nodes as triangles. We enclose the known subgraph G_K with a dashed contour. Panel (c) shows the available data before the second step, and panel (d) shows the available data before the third step.

the target is part of the network under consideration. In our decentralized situation, the upper bound for the CDF of a node in G_K is

$$f_{max}^r(t) = \max_{j \in N_F}[f(r,j;t)].$$

Changing the initial conditions of visited nodes to

$$w_i^0(t) = f_{max}(t), \quad \forall i \in N_V$$

accelerates the convergence of the two sets of sequences because it reduces the initial differences between them. In Appendix A, we give pseudocode for our decentralized algorithm.

VIII. SIMULATIONS ON A SMALL-WORLD NETWORK

We test our algorithm on a variant of Kleinberg's small-world network [28]. We start with a 10×10 square lattice with undirected edges between neighboring nodes in the grid; we also assign an undirected shortcut edge from each node to exactly one other node. For each shortcut, we determine the destination node using independent random trials so that the probability of such a long-range edge is proportional to $1/D_{ij}^2$, where D_{ij} is the lattice distance between i and j. (In determining shortcuts, we discard duplicate edges.)

Noland *et al.* [18] (and references therein) investigated traffic-flow data sets and reported that travel times have log-normal distributions. For the purpose of our numerical simulations, we let the random edge weights be distributed log-normally with PDF

$$p_{ln}(\mu,\sigma;t) = \frac{1}{x\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2}\left(\frac{\ln x - \mu}{\sigma}\right)^2\right].$$

We assign a random variable to each edge by choosing the parameters μ and σ uniformly at random from the interval $[0.5,1.5]$.

We consider Fan *et al.*'s centralized algorithm and our new decentralized algorithm with the joint criterion that we described previously. Note that the joint criterion reduces to Fan *et al.*'s criterion in the limit $\theta \to 1$. We let d_{ij} be the Euclidean distance between nodes i and j, and we approximate the network distance by $g_{ij} \approx h(d_{ij}) \equiv d_{ij}$.

We investigate two different choices for the characteristic edge length λ and the characteristic distribution $\bar{p}(t)$. First, we let λ be the mean length of all edges in a network. Without further knowledge about a network, we assume that the weight $\bar{t}$ incurred by making a step towards the target node is chosen uniformly at random from the weights associated with the edges. The PDF of $\bar{t}$ is the mixture distribution [27]

$$\bar{p}(t) = \frac{1}{m}\sum_{(i,j) \in E} p_{ij}(t).$$

We call this method *global estimation* (GE), because we make indirect use of global knowledge in the calculation of λ and $\bar{p}(t)$.

Second, we restrict the sample from which we calculate λ and $\bar{p}(t)$ to edges that are connected to visited nodes. In

this way, we use only local information. In particular, we calculate

$$E_V = \{(i,j) \in E \,|\, j \in N_V\},$$
$$\lambda = \frac{1}{|E_V|}\sum_{(i,j) \in E_V} d_{ij},$$
$$\bar{p}(t) = \frac{1}{|E_V|}\sum_{(i,j) \in E_V} p_{ij}(t).$$

We call this method *local estimation* (LE).

Suppose that the origin of a routing process is $(2,2)$ and that the target is $(9,9)$, where (x,y) designates a node using its lattice coordinates. We run each test 10^3 times for several budgets [see Fig. 3(a)], several CDF thresholds [see Fig. 3(b)], and a tolerance of $\epsilon = 10^{-3}$. (An error in arrival probability smaller than a tenth of a percent will not affect real travelers.)

As shown in Fig. 3(a), the arrival fraction—i.e., the fraction of routing attempts that reach the target node within a given budget—increases with increasing budget. Centralized algorithms know the entire network topology and can thus make better decisions, so they have larger arrival fractions. The LE algorithms have the same arrival fraction as the GE algorithms. Thus, it is sufficient to sample the network locally,
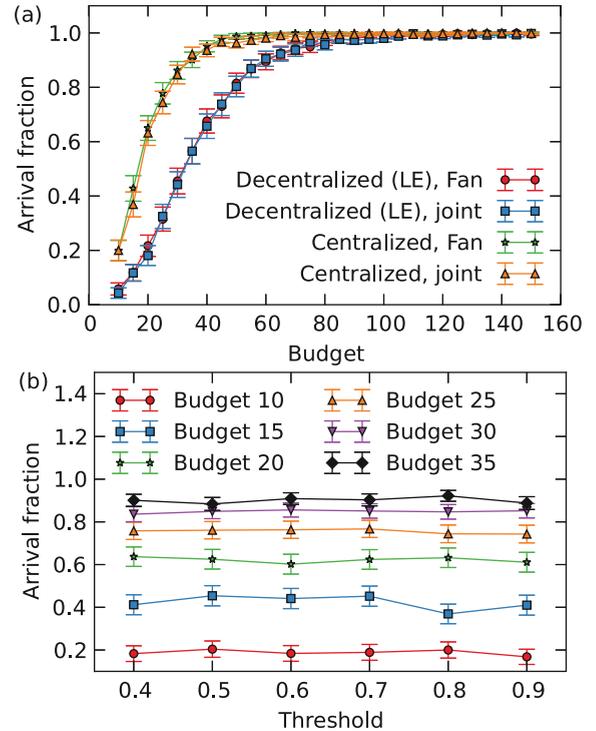

FIG. 3. (Color online) (a) Fraction of routing attempts that successfully reach the target node in (a variant of) the Kleinberg network as a function of budget for a CDF threshold of $\theta = 0.8$ and a tolerance of $\epsilon = 10^{-3}$. (We do not show the results of GE because they agree with those of LE.) (b) Arrival fractions obtained by Fan *et al.*'s centralized algorithm as a function of CDF threshold for a tolerance of $\epsilon = 10^{-3}$ and several different budgets. The error bars in the two panels correspond to three standard deviations from the mean.
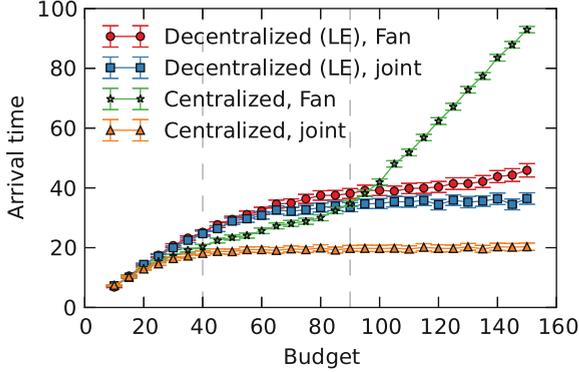
FIG. 4. (Color online) Mean travel time of successful routing attempts on (a variant of) the Kleinberg network as a function of budget. We separate small, intermediate, and large budgets using dashed vertical lines. (We do not show the results of GE because they agree with those of LE.) The error bars correspond to three standard deviations from the mean.

and global knowledge is not required as long as the network is sufficiently homogeneous (i.e., if a sample of the network provides representative estimates of λ and $\bar{p}$). In some cases, some global characteristics of a network might even be known *a priori*, and such information can be used to inform sampling strategies.

As shown in Fig. 3(b), the arrival fraction of Fan *et al.*'s centralized algorithm using the joint criterion is almost independent of the threshold. Because the local neighbors (i.e., the neighbors in the underlying lattice network) of each node are located in the cardinal directions, the arrival CDFs are sufficiently different for CDF maximization and travel time minimization to agree. We obtain the same results using our decentralized algorithm. Travelers thus choose the same successor node irrespective of the threshold θ; this results in the same arrival fraction.

Because centralized algorithms are aware of all shortcuts in a network, they have smaller mean travel times than decentralized ones (see Fig. 4). For small budgets ($\tau \lesssim 40$), the travel times of all algorithms increase with increasing budget. The algorithms choose neighboring nodes to maximize the CDF, which results in longer travel times because it is advantageous to exhaust the budget. For budgets τ that satisfy $40 \lesssim \tau \lesssim 90$, the travel times using Fan *et al.*'s criterion and our joint criterion start to differ. The joint criterion starts to minimize travel times in this regime until they approach a steady value. Fan *et al.*'s criterion, however, continues to maximize CDFs such that travel times grow with increasing budget. For larger budgets ($\tau \gtrsim 90$), Fan *et al.*'s criterion is unable to distinguish between the CDFs of neighboring nodes, and algorithms using this criterion enter an unguided phase (i.e., one can construe a traveler to be "lost"). The algorithm steps to neighbor nodes seemingly at random until the budget decreases sufficiently for Fan *et al.*'s criterion to discriminate among CDFs. As illustrated in Fig. 4, the travel time using Fan *et al.*'s criterion increases linearly with the budget in this regime.

See Ref. [20] for a thorough comparison of arrival fractions for algorithms that treat edge weights as stochastic variables versus algorithms that minimize expected travel time.
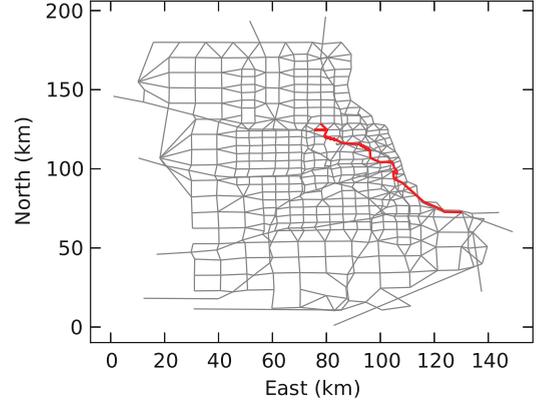

FIG. 5. (Color online) The Chicago sketch network, which has $n = 542$ nodes and $m = 1084$ edges. We show a path in red (thick line).

IX. SIMULATIONS ON THE CHICAGO SKETCH NETWORK

We also test our algorithm on the Chicago sketch network (CSN), which represents an aggregated version of the Chicago metropolitan road network that was developed and provided by the Chicago Area Transportation Study [30,32]. The CSN, which we show in Fig. 5, has $n = 542$ nodes and $m = 1084$ edges.

In Sec. VI, we claimed that there exists a function h such that the network distance between two nodes i and j is well-approximated by $h(d_{ij})$. Our investigation of the CSN allows us to examine this claim more closely. In Fig. 6, we show the joint PDF for the Euclidean distance d_{ij} and network distance g_{ij} for the CSN. To estimate the PDF, we calculate the Euclidean and network distance for all $\frac{1}{2}n(n - 1) = 146\,611$ distinct pairs of nodes and bin the data on a 200×200 grid.

The Euclidean distance

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

is strongly correlated with the network distance. Based on a linear bootstrap fit [31], the best choice for $h(d_{ij}) \approx g_{ij}$ is

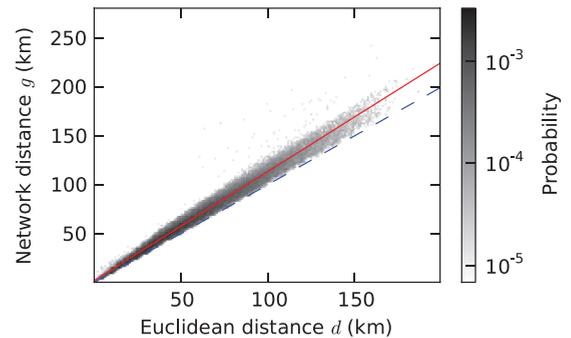$$h(d_{ij}) \approx 1.67(2) \text{ km} + 1.1176(4)d_{ij},$$

FIG. 6. (Color online) Joint PDF for the Euclidean distance d_{ij} and network distance g_{ij} on the Chicago sketch network. The Pearson correlation coefficient between the two distance measures is $\rho \approx 0.985$, which justifies the linear fit (bright red line). The blue dashed line corresponds to the lower bound on the network distance (i.e., $g = d$).

where d_{ij} has units of km. [Recall that "1.67(2)" means that the error bars place the value between 1.65 and 1.69.] The slope of the linear fit is larger than 1 because the Euclidean distance between each pair of nodes provides a lower bound for the network distance between those two nodes.

We obtain similar results when using the lattice distance

$$d_{ij} = |x_i - x_j| + |y_i - y_j| \qquad (5)$$

between nodes i and j. In this case, the slope of the linear fit is smaller than 1 because the lattice distance provides an approximate upper bound for the network distance. Complicated paths, such as zigzag paths, can of course violate this approximate bound.

The mean edge length of the network is $\lambda \approx 5.77$ km. Note that the mean edge length exceeds the typical length of roads in metropolitan areas because the CSN is aggregated: there is no one-to-one correspondence between nodes and junctions. We choose the origin and target nodes uniformly at random such that their Euclidean distance lies in the interval [40, 50] km. We consider the same four tests as in Sec. VIII and run each test 10^3 times for several budgets, several CDF thresholds, and a numerical tolerance of $\epsilon = 10^{-3}$.

In Fig. 7(a), we illustrate the arrival fractions as a function of budget. We observe the same qualitative behavior as on the (variant of the) Kleinberg small-world network. As with the Kleinberg network, the arrival fraction for algorithms that use
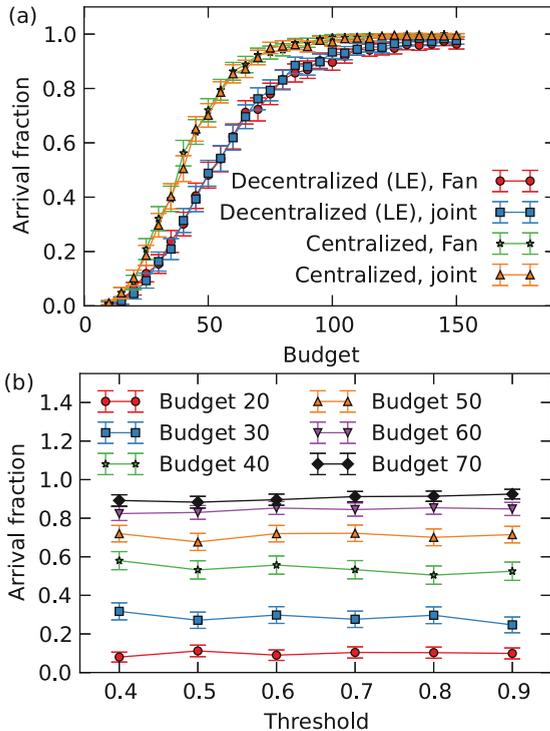

FIG. 7. (Color online) (a) Fraction of routing attempts that successfully reach the target on the CSN as a function of budget for a CDF threshold of $\theta = 0.8$ and a tolerance of $\epsilon = 10^{-3}$. (We do not show the results of GE because they agree with those of LE.) (b) Arrival fractions obtained by Fan *et al.*'s centralized algorithm as a function of CDF threshold for a tolerance of $\epsilon = 10^{-3}$ and several different budgets. The error bars in the two panels correspond to three standard deviations from the mean.
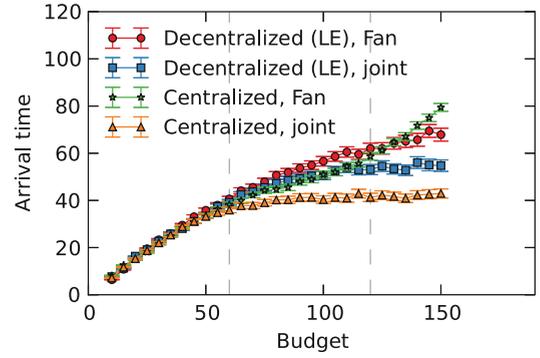

FIG. 8. (Color online) Mean travel time of successful routing attempts on the CSN as a function of budget. We separate small, intermediate, and large budgets using dashed vertical lines. (We do not show the results of GE because they agree with those of LE.) The error bars correspond to three standard deviations from the mean.

the joint criterion depends very little on the CDF threshold θ [see Fig. 7(b)]. In Fig. 8, we show that the travel times of centralized algorithms are smaller than those of decentralized algorithms (because the former know all shortcuts in the network).

X. ADDITIONAL REMARKS

To investigate the computational time of Fan *et al.*'s algorithm and our decentralized algorithm, we perform 100 simulations for several sizes of a Kleinberg small-world network. Fan *et al.*'s algorithm needs to consider the entire network to construct a routing table (see Sec. V). Hence, its computational time increases approximately linearly with the number of nodes (see Fig. 9). However, our decentralized algorithm only considers nodes that are near nodes it has already visited, so its computational time increases sublinearly with the number of nodes. This sublinear scaling illustrates that decentralized routing is possible for networks with stochastic
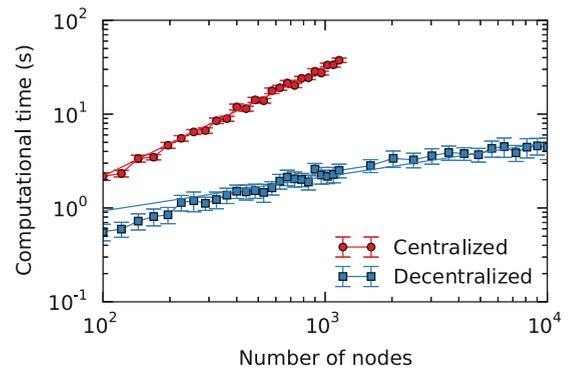

FIG. 9. (Color online) Mean computational times for Fan *et al.*'s centralized algorithm and our decentralized algorithm as a function of the number of nodes in a Kleinberg small-world network. The fits represent power-law scalings with exponents of 1.16 (Fan *et al.*) and 0.36 (decentralized). The error bars correspond to three standard deviations from the mean. We determined the values of these exponents using mean-square regression, and we note that it does not matter whether these curves follow precise power laws.

edge weights because travelers do not explore networks uniformly.

A routing table depends on network topology and the PDFs of the edge weights (but not on the edge weights themselves). Fan *et al.*'s algorithm is more appropriate than ours when network topology is constant (and completely known) and the PDFs (which must also be completely known) do not change during routing, because one can use the same routing table for all travelers who wish to reach the same destination node. Importantly, note that one can use the same routing table if the edge weights change but the PDFs stay the same. Our decentralized algorithm is more appropriate if edges appear and/or disappear or, more generally, if the PDFs change during the routing process (and, clearly, when information is unknown).[2] It constructs a local routing table in a computationally inexpensive way before each step that a traveler makes. Because of its formulation and flexibility, our

---

[2]Note that letting PDFs change during routing gives a more general formulation of an SSSP than the one that we studied in this paper. Our decentralized algorithm is also appropriate for this more general problem.

decentralized algorithm is a more appropriate match than Fan *et al.*'s algorithm for applications to traveling in real life.

Another interesting problem is routing on a network whose topology is known but whose edge-weight distributions are unknown. Without any knowledge about the PDFs, we assume that all edge weights are independently and identically distributed with PDF $\hat{p}(t)$. The arrival CDF along some path ℓ depends only on the number of steps k and is given by

$$U_k(t) = \int_0^t dt' \left(\overset{k}{\underset{k'=1}{*}} \hat{p} \right)(t').$$

Fan *et al.* [26] showed that $U_k(t)$ is a nondecreasing series in k for a given t [i.e., $U_{k+1}(t) \geqslant U_k(t)$].

Because the criteria in Sec. IV favor larger values of $U_k(t)$, paths with the smallest number of steps are optimal. Hence, the problem reduces to a DSSP unless some information about the edge weights is available.

XI. CONCLUSIONS

We have examined decentralized routing on networks with stochastic edges weights. Our contributions are twofold. First, we have introduced a new criterion to discriminate among the

---

Algorithm 1 Our decentralized routing algorithm (which builds on the iterative approximation scheme developed by Fan and Nie [26]). The input parameters are a network G, an origin node, a target node, a time budget τ, a CDF threshold θ, and a CRITERION to identify successor nodes.

---

    function DECENTRALIZED ROUTING(G, origin, target, τ, θ, CRITERION)
        (current, traveltime) $\leftarrow$ (origin,0)
        steps $\leftarrow$ {(current, traveltime)}
        $N_V \leftarrow$ {current}
5:     **while** traveltime $\leqslant \tau$ **and** current $\neq$ target **do**
        $N_F \leftarrow i \quad \forall\{(i,j) \in E : j \in N_V \text{ and } i \notin N_V\}$ ▷ Obtain frontier nodes.
        $v_i^0(t) \leftarrow w_i^0(t) \leftarrow f(i,\text{target};t) \quad \forall i \in N_F$ ▷ Initialize frontier nodes.
        $f_{\max}^{\text{target}}(t) = \max_{i \in N_F}[f(i,\text{target};t)]$ ▷ Obtain upper bound.
        $v_j^0(t) \leftarrow 0 \quad \forall j \in N_V$ ▷ Initialize visited nodes.
10:    $w_j^0(t) \leftarrow f_{\max}^{\text{target}}(t) \quad \forall j \in N_V$
        unstable $\leftarrow N_V$
        $s \leftarrow 0$
        while current $\in$ unstable **do**
           $v_i^{s+1}(t) = \max_{j \in J_i} \left[\int_0^t p_{ij}(t')v_j^s(t-t')\,dt' \right] \quad \forall i \in$ unstable   ▷ Update the sequences.
15:      $w_i^{s+1}(t) = \max_{j \in J_i} \left[\int_0^t p_{ij}(t')w_j^s(t-t')\,dt' \right] \quad \forall i \in$ unstable
          $s \leftarrow s + 1$
          for $i \in$ unstable **do**
            if $w_i^s(t) - v_i^s(t) < \epsilon \quad \forall t$ **then**   ▷ Check for convergence.
               remove i **from** unstable
20:          **end if**
          end for
        end while
        $q_{\text{current}}(t) = \arg\max_{j \in J_{\text{current}}} \left[\int_0^t p_{\text{current } j}(t')v_j^s(t-t')\,dt' \right]$
        successor $\leftarrow$ CRITERION($v_{\text{current}}^s(t), q_{\text{current}}(t), \tau - \text{traveltime}, \theta$)   ▷ Obtain the successor node.
25:    traveltime $\leftarrow$ traveltime + **random sample of** $T_{\text{successor, current}}$ ▷ Update the travel time.
        add successor **to** N_V ▷ Extend the known subgraph.
        current $\leftarrow$ successor   ▷ Make the step to the successor.
        add (current, traveltime) **to** steps
     end while
30:   **return** steps
    end function

CDFs of paths. Our criterion circumvents the limitations of the criteria proposed by Fan *et al.* and Frank, but it retains the desirable properties of both because it minimizes travel times without sacrificing reliability. It also provides a better caricature of the behavior of real travelers [29]. Second, we have developed a decentralized routing algorithm that is applicable to networks with stochastic edge weights. Our algorithm employs a CDF estimation function that captures a notion of proximity in space and guides network travelers without the need to incorporate global knowledge about a network. Our simulation results demonstrate that decentralized routing on networks with stochastic edge weights is viable.

Our approach appears to be very promising. Investigating both its limitations and the situations in which it is most successful are important topics for future research. In particular, it is important to examine the effects of inhomogeneities and different classes of PDFs on routing performance. Possible improvements of our algorithm include the development of more sophisticated choices of estimation functions that incorporate the dependence of edge weights on edge lengths, correlations between the weights of different edges, and more. We expect such work to be particularly interesting in studies of routing on temporal networks, in which the existence and other properties of edges are time-dependent.

ACKNOWLEDGMENTS

APPENDIX A: PSEUDOCODE FOR OUR DECENTRALIZED ALGORITHM

In Algorithm 1, we give pseudocode for our decentralized routing algorithm for networks with stochastic edge weights.

APPENDIX B: MIXTURE OF CONVOLUTIONS VERSUS CONVOLUTION OF MIXTURES

Let $F = \{f_i(x)\}$ and $G = \{g_j(y)\}$ be two finite sets of probability density functions (PDFs), and let the mixtures of the elements of the sets be given by

$$\bar{f}(x) = \sum_i \omega_{f_i} f_i(x), \quad \bar{g}(y) = \sum_j \omega_{g_j} g_j(y),$$

where ω_{f_i} and ω_{g_j} are, respectively, the independent weights associated with the elements of F and G. Taking the mixture after convolving the elements of F and G gives

$$\sum_{ij} \omega_{f_i} \omega_{g_j} (f_i * g_j)(z) = \int_0^z dx \sum_{ij} \omega_{f_i} f_i(z - x) \omega_{g_j} g_j(x)$$
$$= \int_0^z dx \, \bar{f}(z - x) \bar{g}(x)$$
$$= (\bar{f} * \bar{g})(z).$$

Therefore, as long as the assumption of independent weights holds, it follows that mixing the result of a convolution is equivalent to taking the convolution of two mixtures.

Consider b sets of probability distributions $\{F_1, \ldots, F_b\}$. Let each set F_i have c_i elements. Carrying out the convolutions of all pairs of probability distributions in the sets first and taking the mixture afterwards requires $(\prod_{i=1}^b c_i)$ convolutions and additions. However, carrying out the mixtures first and then performing the convolutions requires b convolutions and $(\sum_{i=1}^b c_i)$ additions. It is thus much more efficient computationally to compute the mixtures first and subsequently perform the convolutions.

[1] A. Clauset and C. Moore (2003), arXiv:cond-mat/0309415.

[2] M. Boguñá, D. Krioukov, and K. C. Claffy, Nature Phys. **5**, 74 (2009).

[3] P. Erola, S. Gómez, and A. Arenas, Int. J. Complex Sys. Sci. **1**, 37 (2011).

[4] M. E. J. Newman, *Networks: An Introduction* (Oxford University Press, Oxford, 2010).

[5] E. Dijkstra, Numer. Math. **1**, 269 (1959).

[6] S. Milgram, Psychology Today **1**, 61 (1967).

[7] P. Berman, in *Online Algorithms: The State of the Art*, Lecture Notes in Computer Science, Vol. 1442, edited by A. Fiat and G. J. Woeginger (Springer Verlag, Berlin, 1998), pp. 232–241.

[8] D. Peleg and E. Upfal, J. ACM **36**, 510 (1989).

[9] D. Krioukov, K. C. Claffy, K. Fall, and A. Brady, SIGCOMM Comput. Commun. Rev. **37**, 41 (2007).

[10] J. M. Kleinberg *et al.*, Nature (London) **406**, 845 (2000).

[11] M. Barthélemy, Phys. Rep. **49**, 1 (2011).

[12] M. Rosvall, P. Minnhagen, and K. Sneppen, Phys. Rev. E **71**, 066111 (2005).

[13] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins, Proc. Natl. Acad. Sci. USA **102**, 11623 (2005).

[14] J. Sun and D. ben-Avraham, Phys. Rev. E **82**, 016109 (2010).

[15] D. J. Aldous and J. Shun, Stat. Sci. **25**, 275 (2010).

[16] S. H. Lee and P. Holme, Physica A **390**, 3996 (2011).

[17] S. H. Lee and P. Holme, Phys. Rev. E **86**, 067103 (2012).

[18] R. B. Noland and J. W. Polak, Transport Rev. **22**, 39 (2002).

[19] Y. Y. Fan, R. E. Kalaba, and J. E. Moore, J. Optim. Theory Appl. **127**, 497 (2005).

[20] Y. M. Nie and X. Wu, Transport. Res. B: Methodological **43**, 597 (2009).

[21] Y. Nie and Y. Fan, Transport. Res. Rec. **1964**, 193 (2006).

[22] H. Frank, Oper. Res. **17**, 583 (1969).

[23] R. P. Loui, Comm. ACM **26**, 670 (1983).

[24] A. Eiger, P. B. Mirchandani, and H. Soroush, Transport. Sci. **19**, 75 (1985).

[25] P. G. Hoel, S. C. Port, and C. J. Stone, *Introduction to Probability Theory* Houghton Mifflin Series in Statistics (Houghton Mifflin, New York, 1971).

[26] Y. Fan and Y. Nie, Netw. Spatial Econ. **6**, 333 (2006).

[27] S. Frühwirth-Schnatter, *Finite Mixture and Markov Switching Models* (Springer Verlag, New York, 2006).

[28] J. Kleinberg, in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00 (ACM, New York, 2000), pp. 163–170.

[29] K. A. Small, Am. Econ. Rev. **72**, 467 (1982).

[30] H. Bar-Gera, http://www.bgu.ac.il/~bargera/tntp/. [The CSN data were first posted in 2001, and we downloaded CSN data from this site in 2013.]

[31] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*, Monographs on Statistics and Applied Probability (Chapman & Hall, New York, 1993).

[32] R. W. Eash, K. S. Chon and D. E. Boyce, Transport. Res. Rec. **994**, 30 (1983).