# Routing on spatiotemporal networks without global knowledge

Till Hoffmann

*Department of Physics, University of Oxford, Oxford, United Kingdom*

The formulation of most shortest-path problems makes two simplifying assumptions: (1) the edge weights are deterministic, and (2) global knowledge of the network is available. I develop a decentralised routing algorithm providing en-route guidance for travellers without global knowledge on spatial networks with stochastic edge weights. The algorithm guides travellers using an estimation function that gauges cumulative arrival probability distributions based on the distance between nodes. The estimation function carries a notion of proximity between nodes and enables routing without global knowledge. I develop a new criterion to discriminate among these distributions and compare decentralised algorithms with centralised ones.

## I. INTRODUCTION

Networks are used to model complex systems consisting of interacting entities [1]. For example, junctions connected by roads, such as the *Chicago Sketch Network* [2] in Fig. 1, can be represented by networks. Over the last decades, the increased availability of large data sets and the development of computational tools that can handle these have led to a surge of activity in the field of complex networks. Modelling social [3] and biological [4] networks or transportation [5–7] and communication [8, 9] systems are some of the applications of network theory.

### A. What is a network?

A *directed network* $G$ is a set $N = \{i_1, \ldots, i_n\}$, whose $n$ elements are called *nodes*, together with a set $E = \{e_1, \ldots, e_m\}$, whose $m$ elements are called *edges*. The nodes represent the entities of the underlying system and are labeled by an index $i \in N$. The edges represent the interactions or connections among the entities and are labeled by an ordered pair of indices $(i, j) \in E$, indicating that node $j$ is connected to node $i$ via a directed edge that starts from $i$ and terminates at $j$. Nodes connected to each other are called *neighbours*. In an *undirected* network, the edges are unordered pairs of indices and the corresponding connections are bidirectional. If real scalars $t_{ij}$, such as the length of a road, are assigned to each edge $(i, j)$, the network is said to be *weighted*.

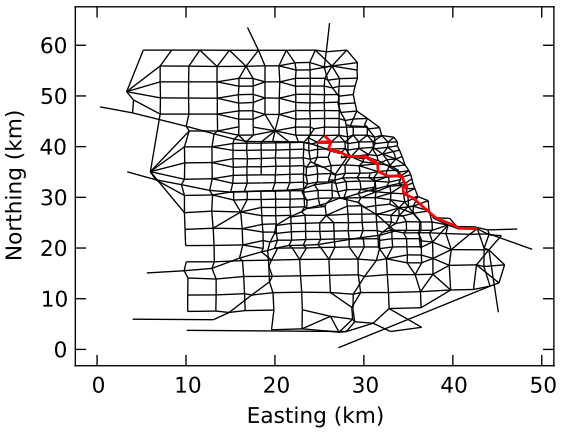


FIG. 1: The Chicago Sketch Network with 542 nodes representing junctions and 1084 edges representing roads. A path is shown in red.

### B. The shortest-path problem

The *shortest-path problem* of finding the path with the smallest weight from an *origin* to a *target* node in a network has received considerable attention because of its great importance for both theory and applications [4, 10–15]. For example, measures used to characterise networks, such as betweenness centrality [12], which is one way to quantify the importance of individual nodes, and the network diameter [1], require the ability to find short paths efficiently.

A *path* $l = \{i_1, \ldots, i_{k+1}\}$ with $k$ steps is a sequence of $k+1$ nodes connected to one another by edges. The first node $i_1$ is called the *origin* and the last node $i_{k+1}$ is said to be the *destination* of the path. The origin and destination of a path do not necessarily have to coincide with the origin and target of a routing process. The weight of a path is given by the sum of the weights of its constituent edges

$$t_l = \sum_{j=1}^{k} t_{i_j i_{j+1}}. \tag{1}$$

The principle that makes efficient pathfinding possible is known as Bellman's principle of optimality [10]. It states that, whatever the initial conditions and previous decisions of a multistage decision process, the remaining decisions have to be optimal with respect to the current state to obtain a globally optimal solution. This implies that the state resulting from previous decisions can be treated as the initial conditions of the same problem. Problems obeying this principle are said to have *optimal substructure* and can be broken down into smaller tasks that can be solved efficiently. In particular, the shortest-path problem is solved by finding shortest subpaths between intermediate nodes and subsequently reconstructing the full path by concatenating the subpaths.

Efficient shortest-path algorithms, such as Dijkstra's algorithm [11], have successfully been applied to a range of real world problems. For example, connecting computers in peer-to-peer (P2P) file-sharing networks [9], navigation of autonomous robots [16], routing information in telecommunication systems [8] and traffic on road networks [5–7] admit natural formulations as shortest-path problems. By employing the same concepts, it is possible to answer questions about the conceptual distance of pieces of information [14], protein sequence alignment [4], or image segmentation [15].

The remainder of this report is organised as follows: Section II provides background about a generalisation of the shortest-path problem using stochastic edge weights and decentralised routing, i.e. routing without global knowledge of the network. In Section III, I develop a new measure to discriminate among paths in the stochastic case, and in Section IV, I introduce a decentralised routing algorithm for networks with stochastic edge weights. I investigate the differences between centralised and decentralised algorithms in Section V and consider the effect of the choice of optimality measure on the routing results based on numerical experiments run on synthetic and real networks. In Section VI, I discuss these results, consider applications of the algorithm developed and prospects for future work in this area.

## II. BACKGROUND

### A. A stochastic shortest-path problem

It is often not sufficient to assign simple scalar weights to edges because of uncertainties about their values. For example, the physical length of a road may be known, but the time it takes to travel this distance may vary considerably depending on traffic [17]. It is natural to generalise the weights to real-valued random variables with probability distribution functions (PDFs) $p_{ij}$ to account for this uncertainty [5, 7, 18–20]. We make the following three assumptions: (1) the random edge weights are independent, (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, e.g. the time taken to travel a road will be known once the next junction is reached. In this report, we consider the example of a transportation network for illustrative purposes; henceforth, the terms 'weights' and 'travel times' will be used interchangeably.

Consider two independent random variables $x$ and $y$ with PDFs $f(x)$ and $g(y)$, respectively. The PDF $h(z)$ for their sum $z = x + y$ is given by the convolution of the individual probability distributions [21]:

$$h(z) = \int_0^z dx\, f(x)\, g(z-x) \tag{2}$$

$$\equiv (f * g)(z)\,. \tag{3}$$

Using Eq. (1) and applying the above repeatedly, the probability to arrive at the destination via a path $l$ in the time interval $[t, t+dt]$ is given by

$$p_l(t) = \left(p_{i_1 i_2} * \ldots * p_{i_k i_{k+1}}\right)(t)$$

$$\equiv \left(\overset{k}{\underset{j=1}{*}} p_{i_j i_{j+1}}\right)(t)\,, \tag{4}$$

where the second line is shorthand notation for $k$ convolutions. The probability of arrival along $l$ with weight smaller than or equal to $t$ is given by the cumulative distribution function (CDF):

$$u_l(t) = \int_0^t dt'\, p_l\left(t'\right). \tag{5}$$

Loui [20] approached the stochastic shortest-path problem outlined above by considering expectation values of *cost functions* to discriminate among paths. These functions take the weight

of a path as their argument and return the perceived cost of the path. They are used to construct nontrivial criteria to discriminate among paths, e.g. quadratic cost functions are used to judge the uncertainty associated with paths [18]. Loui showed that the efficient algorithms developed for the deterministic shortest-path problem outlined in Section I B can be applied if and only if the cost functions are linear or exponential. This is equivalent to showing that Bellman's principle holds for routing algorithms using these cost functions on networks with stochastic edge weights [20]. In the framework developed by Loui, it is possible to find the path with the least expected travel time. If the probability to realise a travel time much larger than the expected travel time is considerable, the path is said to be risky. Such paths are a poor choice for travellers who are risk-averse. The precise definition of risk aversion depends on the requirements of the traveller [18, 22]. For example, travellers may favour reliability over small expected travel times if they need to arrive for an interview on time or are transporting perishable goods.

Thus, there is no unique way to generalise the shortest-path problem to account for stochastic edge weights and several criteria have been proposed to differentiate among paths: Frank [19] defined the optimal path to be the one whose arrival CDF surpasses a given threshold $\theta$ within the shortest travel time. Fan [5] suggested choosing the path with maximal arrival probability within a given time budget $\tau$.

In contrast to the deterministic case, routing algorithms on networks with stochastic edge weights can be classified into two groups: (1) *a priori* algorithms, which determine a route before the traveller starts its journey, and (2) *adaptive* algorithms, which re-evaluate the best routing policy at each step to incorporate the weights of the edges that have already been traversed into the decision-making process. Unfortunately, Bellman's principle does not apply to a priori shortest-path problems under the reliability-based criteria above because considering a step to a node $i$ is not equivalent to starting the process at $i$ (Appendix A includes an example of a violation of Bellman's principle). It is thus necessary to enumerate all paths, determine their arrival distributions and compare the paths pairwise [7]. This process is computationally expensive, and even for a small square lattice of one hundred nodes there are more than $10^{24}$ distinct paths connecting diagonally opposite corners [23]. More efficient algorithms have been developed, but their worst case performance is still exponential in the number of nodes of the network [7].

### B. An adaptive algorithm

There is a crucial difference between a priori and adaptive algorithms: a priori algorithms consider making a sequence of steps but do not execute any steps until the full path has been determined. Adaptive algorithms execute a step and then re-evaluate their options before making the next step. Adaptive routing algorithms alleviate the problem of the large computational complexity. Having arrived at a node $i$, the remaining routing process can be formulated as an identical problem starting at node $i$ and Bellman's principle is satisfied.

Modern navigation algorithms that can incorporate traffic jams, accidents and construction work into their decision-making process are examples of adaptive routing algorithms. Such algorithms outperform a priori shortest-path algorithms because they can respond to the observed travel times. However, they require the traveller to be in possession of a device that can evaluate the different options at each stage of the process, which can be problematic for real-world applications.

Fan [5] proposed an adaptive algorithm that obtains a routing table. She considered the maximal probability to reach the target $r$ from all other nodes $i \in N\backslash\{r\}$ in time $t$, where $N\backslash\{r\}$ denotes all elements of $N$ excluding $r$. This amounts to solving a set of nonlinear convolution integral equations given by

$$u_i(t) = \max_{j \in J_i} \left[ \int_0^t p_{ij}\left(t'\right) u_j\left(t - t'\right)\, dt' \right], \quad (6)$$

$$u_r(t) = 1, \quad (7)$$

where $u_i(t)$ is the probability to arrive at the target from $i$ in time t and $J_i$ is the set of neighbours of node $i$. The node $q_i(t)$ that should be chosen to realise the maximal arrival probability is given by

$$q_i(t) = \arg\max_{j \in J_i} \left[ \int_0^t p_{ij}\left(t'\right) u_j\left(t - t'\right)\, dt' \right], \quad (8)$$

where arg max denotes the argument $j \in J_i$ maximising the expression in brackets. Travellers can

then consult the routing table during their journey and make optimal decisions depending on their current location and their remaining budget. In particular, a traveller at node $i$ with remaining budget $\tau$ should choose $q_i(\tau)$ according to Fan's optimality index. A traveller that follows Frank's measure of optimality should choose $q_i(t_c)$, where $t_c$ is the shortest travel time that realises an arrival probability $\theta$ and is defined by $u_i(t_c) = \theta$.

In general, analytical solutions to the convolution integral equations (6) to (8) cannot be found, but they can be solved using iterative approximation techniques [5]. In particular, the CDF $u_i(t)$ to arrive at a target $r$ from node $i \in N\backslash\{r\}$ can be approximated by the sequence

$$\begin{aligned} v_i^{k+1}(t) &= \max_{j \in J_i} \left[ \int_0^t p_{ij}(t')\, v_j^k(t-t')\, dt' \right], \quad (9) \\ v_r^{k+1}(t) &= 1 \end{aligned}$$

with initial conditions

$$\begin{aligned} v_i^0(t) &= 0 \qquad \forall\, i \in N\backslash\{r\}, \quad (10) \\ v_r^0(t) &= 1. \end{aligned}$$

The solution after $k$ iterations $v_i^k(t)$ can intuitively be understood as the probability to arrive at the target from node $i$ within time $t$ if the traveller is not allowed to make more than $k$ steps. Unfortunately, the number of iterations required for the solution to converge is in principle not bounded from above because the traveller can step back and forth between nodes.

Fan and Nie [6] showed that $v_i^k(t)$ is monotonically increasing with the iteration index $k$. The sequence is thus a lower bound for the true CDFs. A sequence $w_i^k(t)$ with the same recursion relation defined in Eq. (9) but different initial conditions

$$w_i^0(t) = 1 \qquad \forall\, i \in N \quad (11)$$

is an upper bound for the true CDFs because it is monotonically decreasing with $k$ [6]. Thus, if $v_i^k(t) = w_i^k(t)$, the two sequences have converged and the true CDF has been obtained. In numerical implementations, we demand that the approximation differs by no more than a positive number $\epsilon$ from the true CDF, i.e. we require that

$$|v_i^k(t) - w_i^k(t)| < \epsilon \qquad \forall\, i \in N, t.$$

### C. Decentralised routing

Milgram [3] investigated the *small-world phenomenon*, i.e. the phenomenon that most people are connected by short chains of acquaintances. He asked a group of people to forward a folder to a target individual using only acquaintances they knew on a first name basis as intermediaries. The individuals at the origins of the paths received basic information about the target individual and were asked to forward the folder to the person in their social circle whom they believed to be closest to the target. Subsequent recipients followed the same prescription, and the folders honed in on their target. Of 160 chains that were started, 44 were completed. The average number of hops required to reach the target was approximately six and gave rise to the popular concept of six degrees of separation.

The success of Milgram's experiment is due to two components: (1) short paths connecting individuals existed in the network, and (2) individuals were able to navigate the network efficiently despite only having knowledge of their own social circle. The first component was investigated by Watts and Strogatz [24] who devised a network model that can account for the clustering observed in social networks as well as the existence of short paths between individuals.

Kleinberg [25] explored the second component: *decentralised algorithms* that are able to find short paths despite only having local knowledge of the network. A *Kleinberg lattice*, a variant of the Watts-Strogatz model, is a square lattice to which long-range connections of lattice distance $d$ have been added with probability proportional to a power-law distribution $d^{-\alpha}$. The nodes are labeled by their positions in the grid such that $i \to (x_i, y_i)$. The lattice distance between two nodes $i$ and $j$ is

$$d_{ij} = |x_i - x_j| + |y_i - y_j|. \quad (12)$$

Slivkins [26] generalised Kleinberg's ideas by embedding networks in a metric space and defining a distance measure $d : N \times N \to \mathbb{R}$ between any pair of nodes. In principle, decentralised routing is possible by making a traveller choose successor nodes that minimise the distance from the target in the models developed by Kleinberg and Slivkins.

P2P file-sharing networks contain a large number of files. This makes it difficult to maintain a central index. Decentralised search algorithms have been implemented in several P2P

clients to circumvent this problem [9]. Shortest-path problems also arise in robot navigation and decentralised routing is an effective way to provide guidance for robots in unknown environments [16].

Decentralised algorithms are inherently adaptive because they do not identify a path a priori but provide en-route guidance for travellers. In Section V, I will show that they can route travellers on networks with stochastic edge weights given only local knowledge of the network.

## III. A JOINT OPTIMALITY INDEX

Given the current node $i$ occupied by a traveller, the aim of any routing criterion is to identify the neighbour of $i$ that it deems to be an optimal choice. If the remaining budget $\tau$ available to a traveller is sufficiently large, arrival at the target within the budget is almost certain regardless of the choice of neighbour, and we speak of the *large-budget limit*. For example, a budget of one week for the journey from Oxford to Cambridge would be in the large-budget limit. In this regime, Fan's criterion cannot discriminate among the CDFs of neighbours effectively because $u_j(\tau) \approx 1$ for all neighbours $j$. However, Frank's measure can identify the neighbour that should be chosen to realise the shortest travel time once we have specified the arrival probability threshold $\theta$, i.e. how risk-averse we are.

In contrast, if the remaining budget is sufficiently small, the arrival probability is small regardless of the choice of neighbour, i.e. $u_j(\tau) \ll 1$ for all neighbours $j$. In this case, we speak of the *small-budget limit*, and Frank's criterion is not applicable because none of the CDFs surpass the probability threshold $\theta$. However, Fan's criterion can identify the neighbour that should be chosen to have the largest arrival probability at the target. A budget of one hour for the journey from Oxford to Cambridge would be in the small-budget limit.

With the above in mind, we define a new optimality index. If any neighbours' arrival CDFs surpass a given threshold $\theta$ within a time budget $\tau$, we choose the neighbour whose arrival CDF realises the threshold in the shortest time. Otherwise, we choose the neighbour whose arrival CDF is maximal for the given time budget. This closely resembles the decision-making process of real travellers who emphasise reliability when they are running late and are concerned with short travel times when they are planning well in advance [22]. Figure 2 illustrates the differences between the three criteria. Algorithm 1 in Appendix C shows pseudocode for selecting successor nodes according to different criteria.

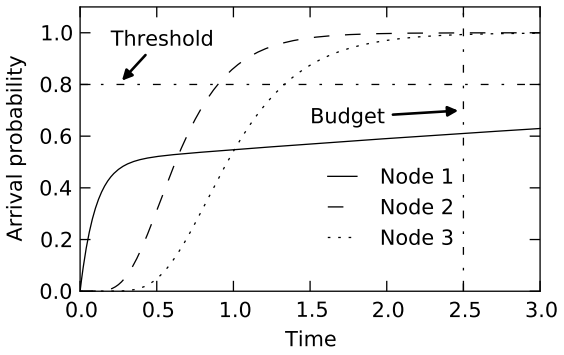


FIG. 2: A set of arrival CDFs of three neighbours of the current node. Fan's criterion prefers 2 and 3 over 1 but cannot discriminate between the arrival CDFs of 2 and 3. Frank's criterion prefers 2 over 3 but is not applicable to 1. The joint criterion is applicable to all CDFs and chooses 2.

## IV. A DECENTRALISED ROUTING ALGORITHM

### A. Estimation function

We follow Slivkins [26] and consider a spatial network [27] embedded in a *metric space* such that the distance between nodes $i$ and $j$ is given by $d_{ij} = d(i,j)$. In the deterministic case, such a distance measure is sufficient to guide travellers without global knowledge to a target [25, 26, 28]. Having generalised the edge weights to random variables, it is necessary to estimate the arrival CDFs between two nodes instead of considering the distance between them. We define an estimation function $f : N \times N \to \mathcal{C}$, where $N$ is the set of nodes and $\mathcal{C}$ is the set of all CDFs. The function $f(i,j;\tau)$ estimates the arrival CDF from node $j$ to $i$ within a time budget $\tau$. It carries a notion of proximity between the nodes and enables routing without global knowledge. We shall first consider the components that are necessary to define the estimation function.

The *network distance* $g_{ij}$ is the shortest distance between nodes $i$ and $j$ if travellers are restricted to the edges of the network. For example, the distance between Oxford and Cambridge as the crow flies corresponds to the Euclidean distance (neglecting the curvature of the Earth) and the distance travelled by a car on the shortest

route corresponds to the network distance. We assume that the distance between nodes can be used to estimate the network distance and that a function $h : \mathbb{R} \to \mathbb{R}$ exists such that $h(d_{ij}) \approx g_{ij}$. This assumption is implicit in all decentralised routing algorithms on spatial networks because the distance between nodes is used to guide travellers [25, 26, 28]. We shall discuss it in more detail in Section V B.

Let

$$\lambda = \frac{1}{m} \sum_{(i,j)\in E} d_{ij} \tag{13}$$

be the mean edge length of the network under consideration, where $m$ is the number of edges. The expected number of steps $\bar{k}$ to reach $i$ from $j$ is thus given by the network distance $g_{ij}$ divided by the mean edge length $\lambda$:

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

where $\lceil x \rceil$ denotes the smallest integer not less than $x$.

Without any further knowledge about the stochastic edge weights associated with each step of the unknown path $\bar{l}$ from $j$ to $i$, we assume that the weight is chosen uniformly at random from one of the edges of the network at each step. The weight of the path is estimated to be

$$t_{\bar{l}} = \sum_{k=1}^{\bar{k}} \bar{t}, \tag{14}$$

where $\bar{t}$ is the weight obtained by chosing an edge weight uniformly at random. The PDF for $\bar{t}$ is a mixture distribution [29], which accounts for chosing edge weights uniformly at random:

$$\bar{p}(t) = \frac{1}{m} \sum_{(i,j)\in E} p_{ij}(t). \tag{15}$$

Following equations (1) and (5), an estimate of the CDF for the weight of $\bar{l}$ is given by

$$f(i,j;t) = \begin{cases} \displaystyle\int_0^t dt' \left( \mathop{*}_{k=1}^{\bar{k}} \bar{p} \right)(t') & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases} \tag{16}$$

Appendix B includes a proof that the order of performing mixtures and convolutions is irrelevant if the weights associated with PDFs are independent as we have assumed here.

The estimation function is not unique, and the choice used in this report indirectly makes use of global knowledge by employing the mean edge length in Eq. (13) and uniform mixture distribution in Eq. (15). This choice is motivated by the requirement for characteristic length scales and travel time distributions which are necessary for $f(i,j;t)$ to be a good estimate of the arrival CDF. In practice, such characteristics can be obtained by sampling the network (e.g. traffic flow measurements) or may be known a priori, so global knowledge is not required.

### B. The algorithm

A decentralised algorithm is inherently adaptive because it does not identify a path a priori. It gathers new information as travellers traverse the graph. The nodes that have been visited $N_V$ and the frontier nodes $N_F$ constitute the discovered subgraph $G_D$. Frontier nodes are neighbours of visited nodes but have not yet been visited themselves. The discovered subgraph has all edges of $G$ that are connected to the discovered nodes $N_D = N_V \cup N_F$. Naïvely stepping towards the node that appears to be the optimal choice without incorporating the journey to date can lead to travellers getting trapped in dead ends. Letting the algorithm have knowledge of $G_D$ enables it to navigate out of such dead ends.

Fan's algorithm [6] can easily be applied to $G_D$ by changing the initial conditions of the two sequences $v_i^k(t)$ and $w_i^k(t)$ defined in Section II B. The frontier nodes are assigned CDFs according to the estimation function:

$$v_j^0(t) = w_j^0(t) = f(j,r;t) \qquad \forall\, j \in N_F,$$

where $r$ is the target of the routing process. The

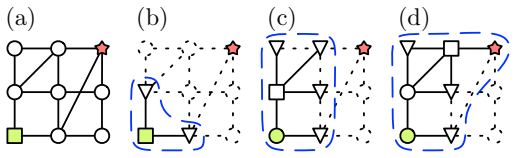


FIG. 3: (a) A Kleinberg lattice with the origin as a green square and the target as a red star. (b) The data available to a decentralised routing process before the first step. The current node is shown as a square and frontier nodes are denoted by triangles. The discovered subgraph $G_D$ is enclosed in the dashed blue contour. (c) and (d) show the available data before the second and third steps, respectively.

visited nodes are initialised according to equations (10) and (11) as before. The two sequences are iterated until they converge to within a chosen tolerance $\epsilon$ and the best successor node can be identified by one of the criteria discussed in Sections II A and III. The traveller subsequently moves to the chosen successor node, and the discovered subgraph is extended by adding the successor node to the visited nodes and its neighbours to the frontier nodes. The remaining budget is reduced by the weight incurred by making the step. The process is repeated until the traveller reaches the target or the budget is exhausted. Figure 3 illustrates a routing process on a Kleinberg lattice.

The algorithm requires that the sequence $w_i^k(t)$ is initialised with an upper bound for the arrival CDF. In the centralised case, this upper bound is necessarily 1 because the target node is part of the network under consideration. However, the upper bound for the arrival CDF of a node in the discovered subgraph is given by the maximal arrival CDF among the frontier nodes

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

Initialising the sequence $w_i^k(t)$ with $f_{\max}(t)$ instead of 1 such that

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

accelerates the convergence of the two sequences because the initial difference between them is reduced. Note that $w_i^k(t)$ remains an upper bound for the arrival CDFs and that $f_{\max}(t) = 1$ if the target is part of the frontier. Algorithm 3 in Appendix C shows pseudocode for the algorithm.

## V. SIMULATIONS

I implemented the decentralised algorithm and the algorithm developed by Fan and Nie [6] for comparison with a centralised algorithm. Noland et al. [17] and references therein found the traffic flow data sets they investigated to have lognormal travel time distributions, and I used this class of distributions for the purpose of the following numerical experiments.

In particular, I assigned travel time distributions to the edges of the networks under consideration by generating the mean $\mu$ and standard deviation $\sigma$ uniformly at random from the interval $[0.5, 1.5]$. The relevant mixture distribution is thus

$$\bar{p}(t) = \iint_{0.5}^{1.5} d\mu\, d\sigma\, p_{\ln}(\mu, \sigma; t),$$

where $p_{\ln}(\mu, \sigma; t)$ denotes the lognormal distribution with mean $\mu$ and standard deviation $\sigma$. Note that the normalisation of the mixture is correct because the size of the interval from which $\mu$ and $\sigma$ were chosen is 1.

I ran four different configurations: the decentralised algorithm and the centralised algorithm both using Fan's criterion and the joint criterion. Frank's criterion was not used in the numerical experiments because it is difficult to determine the size of the integration interval that is required for the CDFs of all nodes to surpass the threshold $\theta$. If there were CDFs that did not surpass $\theta$, Frank's criterion would not be applicable. The convolutions were carried out using a trapezoidal, discretised approximation of Eq. (2) with time steps of width $\delta t = 0.01$. I chose the width of time steps such that reducing them further did not affect the results. Algorithm 2 in Appendix C shows pseudocode for the discretised convolution.

### A. Kleinberg lattice

I first tested the algorithm on a $10 \times 10$ node Kleinberg lattice [25]. One long-range contact was created for each node with a power-law exponent $\alpha = 2$. Any duplicate edges were discarded. The nodes were labeled by their position in the lattice $i \to (x_i, y_i)$ and I used the lattice distance defined in Eq. (12) to measure the distance between nodes. I let $\lambda = 1$ lattice unit and $h(d) = d$ as an approximation to the network distance between nodes, i.e. the presence of long-range connections was neglected. The origin of the routing process was $(2, 2)$ and the target was $r = (9, 9)$ such that the distance from origin to target was 14 lattice units. I ran each configuration $10^3$ times for a range of budgets and a threshold probability of 80% for the joint criterion. I chose a numerical tolerance $\epsilon = 10^{-3}$ because errors in arrival probabilities smaller than this threshold are not of significance for most real travellers.

The arrival fraction, i.e. the fraction of routing attempts that reached the target within a given budget, increased monotonically with increasing budget as illustrated in the upper panel of Fig. 4. Because centralised algorithms had

global knowledge of the network, they could make better decisions and had larger arrival fractions than decentralised algorithms. For the joint criterion, the arrival probability threshold $\theta$ determines a point of transition from arrival probability maximisation to travel time minimisation. Interestingly, the arrival fraction of travellers following the joint criterion was independent of $\theta$ as shown in the lower panel of Fig. 4. This implies that travel time minimisation is sufficient to achieve the highest possible reliability for a given time budget on the Kleinberg lattice. Such behaviour is unexpected but can be explained. Because the local neighbours of any node were located in the four cardinal directions, the distances from these neighbours to the target were dissimilar. Hence, the associated arrival CDFs were dissimilar and a disagreement between arrival probability maximisation and travel time minimisation was unlikely. Travellers thus chose the same successor nodes irrespective of $\theta$ resulting in the same arrival fraction.

Centralised algorithms had smaller mean arrival times than decentralised ones because the

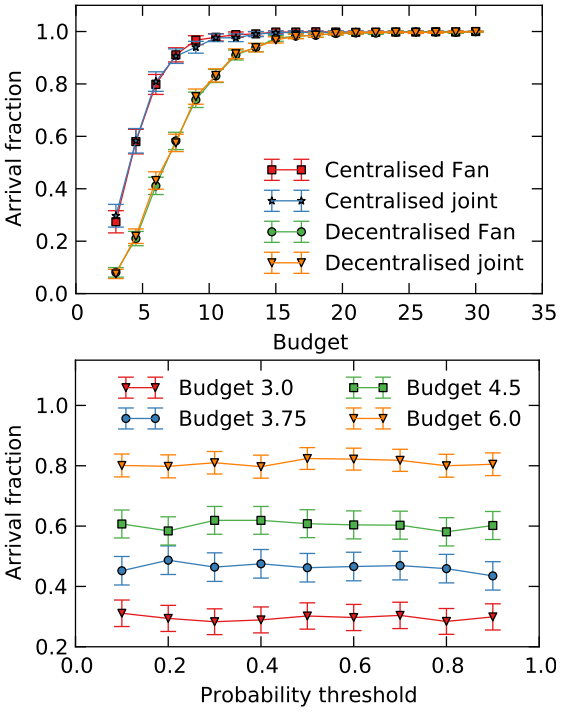


FIG. 4: The first panel shows the fraction of routing attempts that successfully reached the target on the Kleinberg lattice. The second panel shows arrival fractions as a function of probability threshold for four different budgets obtained by the centralised algorithm using the joint criterion. The error bars correspond to three standard deviations of the mean.

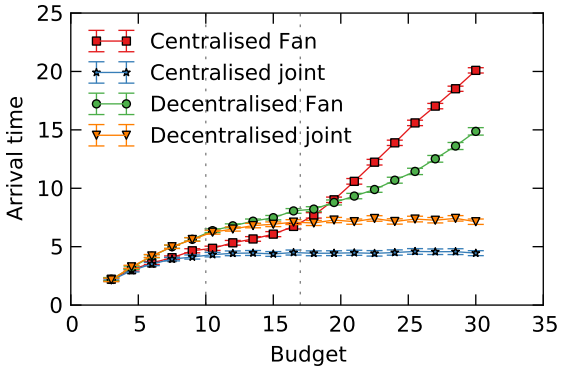


FIG. 5: Mean arrival times of successful routing attempts. The arrival times of algorithms using Fan's criterion rise sharply at large budgets because the criterion cannot guide walkers effectively in this regime. Small, intermediate and large budgets are separated by vertical lines. The error bars correspond to three standard deviations of the mean.

former were aware of all shortcuts in the network, as shown in Fig. 5. The arrival times showed several interesting features. For small budgets $\tau \lesssim 10$, the arrival times of all algorithms increased with increasing time budget. The algorithms chose neighbours to maximise the arrival probability, resulting in longer travel times because it was advantageous to exhaust the entire time budget.

For intermediate budgets $10 \lesssim \tau \lesssim 17$, the arrival times of the Fan and joint algorithms differed. The joint criterion minimised arrival times once arrival CDFs surpassed the threshold. Hence, the arrival times of algorithms using this criterion did not increase further with increasing time budget. Algorithms employing Fan's criterion, however, continued to maximise the arrival probability such that the arrival times continued to grow with increasing time budget.

For large budgets $\tau \gtrsim 17$, Fan's criterion was not able to distinguish between the CDFs of neighbouring nodes as discussed in Section III, and algorithms using this criterion entered an unguided phase: the algorithms stepped to neighbours virtually at random until the remaining budget had decreased sufficiently for Fan's criterion to be able to discriminate among CDFs again. Thus, any additional budget was consumed in the unguided phase and the mean arrival times increased linearly with the available budget. Using bootstrap fitting [30], the best estimate of the slope of a linear function fitted to the tail of the arrival times of the centralised algorithm using Fan's criterion was 1.02(2). For algorithms using the joint algorithm, mean arrival

times approached a steady value for large budgets because the joint criterion was equivalent to Frank's criterion in this regime and minimised arrival times. Note that the mean arrival times were always smaller than the corresponding budget because the budget was an upper bound for the arrival time.

The arrival times of centralised algorithms using Fan's and the joint criterion started to differ at smaller time budgets compared to the arrival times of decentralised ones. In the decentralised case, long-range connections were neglected and $\lambda = 1$ underestimated the mean edge length. Hence, $\bar{k} = \frac{d_{ir}}{\lambda}$ overestimated the number of steps necessary to reach the target $r$ from a frontier node $i$. Consequently, the estimation function $f(r, i; t)$ underestimated the arrival CDF. The decentralised joint algorithm thus erred on the side of caution and the transition from arrival probability maximisation to travel time minimisation occurred at a larger budget $\tau \approx 12$ than in the centralised case $\tau \approx 9$.

### B. Chicago Sketch Network

Having demonstrated that a decentralised algorithm is able to find reliable paths on a Kleinberg lattice, I investigated a real network. The Chicago Sketch Network [2] models the larger Chicago metropolitan area, and has 542 nodes representing junctions and 1084 edges representing roads. It is shown in Fig. 1.

In Section IV A, I proposed that a function $h$ exists such that the network distance between two nodes $i$ and $j$ is well approximated by $h(d_{ij})$. We will investigate this claim more closely in this section. Figure 6 shows that 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, the best choice for $h \approx g_{ij}$ is

$$h(d_{ij}) = 0.547(8)\ \text{km} + 1.1176(4) \times d_{ij},$$

where $d_{ij}$ has units of km. The slope of the linear fit was larger than 1 because the Euclidean distance between any pair of nodes is a lower bound for the network distance between the nodes.

I obtained similar results when considering the lattice distance defined in Eq. (12) instead. In this case, the linear fit had a slope smaller than 1 because the lattice distance is an approximate upper bound for the network distance. Complicated paths, such as zigzag paths, may however violate this approximate bound.

The mean edge length of the network is $\lambda = 1.89$ km. The origin and target nodes were chosen uniformly at random such that their Euclidean distance was in the interval between

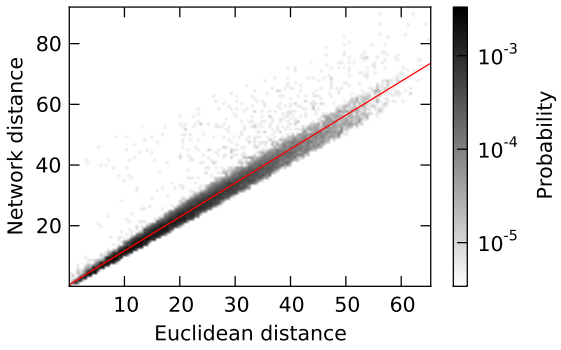


FIG. 6: Logarithmic density plot of the probability density of network distances for a range of Euclidean distances between nodes. The Pearson correlation coefficient of Euclidean and network distances is 0.985, justifying a linear fit shown in red.

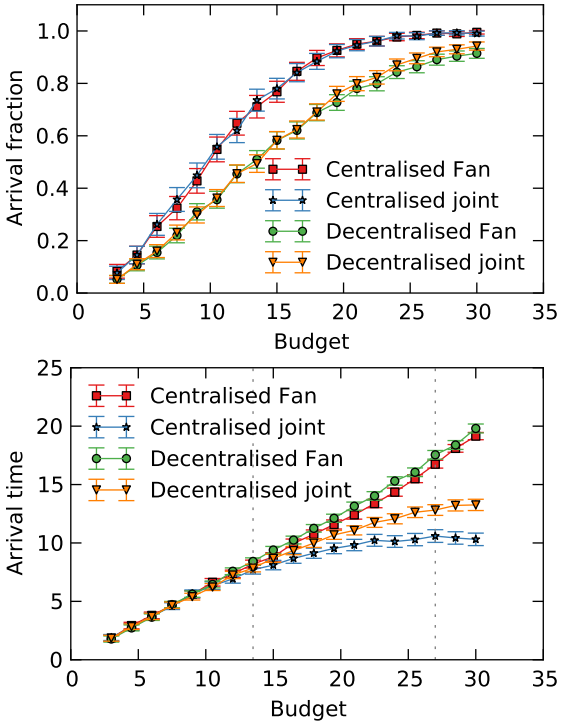


FIG. 7: The first panel shows the fraction of routing attempts that successfully reached the target on the Chicago Sketch Network. The second panel shows the mean arrival times of successful attempts. Small, intermediate and large budgets are separated by vertical lines. The error bars correspond to three standard deviations of the mean.

13.1 km and 16.4 km. This interval corresponds to the most likely distance range between any pair of nodes selected uniformly at random. I ran each configuration $10^3$ times for a range of budgets, a threshold probability of 80% for the joint criterion and a numerical tolerance $\epsilon = 10^{-3}$.

The results show the same qualitative behaviour as on the lattice and are illustrated in Fig. 7. The arrival fraction of algorithms using the joint criterion showed no dependence on the probability threshold $\theta$ in line with the results obtained for the Kleinberg lattice. However, the travel times of the centralised and decentralised algorithms using the Fan and joint criteria started to differ at the same time budget. This occured because $\lambda$ was the mean edge length – an unbiased estimate of the edge lengths – such that the estimation function neither overestimated nor underestimated the arrival CDF on average.

## VI. DISCUSSION

Two oversimplifying assumptions prevalent in the formulation of most shortest path problems have been lifted: (1) routing on networks with stochastic edge weights goes beyond the simplistic approach of treating the weights associated with the edges of a network deterministically [5–7, 17, 19, 20, 31], and (2) decentralised routing algorithms enable navigation on networks without having global knowledge of the connections between nodes [8, 9, 16, 25, 26, 28].

I developed a decentralised routing algorithm that can be applied to a network with stochastic edge weights and a new criterion to discriminate among the CDFs of potential successor nodes for travellers. The new criterion circumvents the limitations of Frank's and Fan's optimality indices, which are not applicable in the small and large budget limits, respectively. Furthermore, it retains the desirable properties of both: it minimises the travel time similar to Frank's measure without de facto sacrificing reliability in comparison to Fan's optimality index. The joint criterion is a more realistic model of the behaviour of real travellers than previous measures of optimality [22]. Future work could investigate under which conditions the reliability of the joint criterion is independent of the probability threshold $\theta$.

I generalised the notion of proximity necessary for decentralised routing algorithms to networks with stochastic edge weights by introducing a CDF estimation function. The estimation function is at the core of the decentralised algorithm and is likely to have significant effects on the routing quality. The function used in this report is simplistic because it assumes that the traveller needs to make a well-known number of steps based on the distance between two nodes. It ignores correlations between the edge length and the edge weight and assumes that the weights of edges are independent. More sophisticated versions could account for the distribution of edge lengths, edge weights and their correlation. Shortest-path problems that consider the least expected travel time for correlated edge weights have been considered [32] but a generalisation of the approach presented in this report to correlated edge weights remains to be investigated. Given that a range of estimation functions can be defined, it needs to be considered whether a unique, optimal estimation function exists.

Because decentralised algorithms only have access to part of a network, they need to process a smaller amount of information than centralised ones. Often the choices made at the current stage of a multi-stage decision process are mostly determined by the local neighbourhood of the current state. For example, a person travelling from Oxford to Cambridge needs to decide whether to go east or west, but whether a particular shortcut exists in London is likely to be irrelevant for a traveller starting in Oxford. Thus, limiting the amount of information available to an algorithm artificially could have performance benefits without jeopardising the results considerably.

In fact, even decentralised routing algorithms on networks with deterministic edge weights could benefit from the ideas presented in this report. The traveller discovers the graph as it approaches the target and is not aware of the presence of dead ends or shortcuts. However, if macroscopic properties or the generative model of the network are known, characteristic length scales and travel time distributions can be derived. Deterministic decentralised routing algorithms could be extended to let travellers specify whether they prefer reliability or short travel times by treating the unknown graph in the same way as in this report.

The results I obtained from numerical simulations show that decentralised routing on networks with stochastic edge weights is possible, but the requirements for such an algorithm to succeed have not been investigated. Slivkins [26] considered the requirements for decentralised

routing to succeed on networks embedded in a metric space. These concepts have to be revisited and combined with the question of whether there is a set of edge-weight distributions that allows for decentralised routing and a set of distributions that does not in order to put decentralised routing on networks with stochastic edge weights on a more solid footing.


### Acknowledgments

I would like to thank my supervisor Mason A. Porter and Renaud Lambiotte for fruitful discussions.


### Appendix A: An example violating Bellman's principle

Consider an a priori routing process using Fan's criterion on the network shown in Fig. 8 (a). For a budget of $\tau = 2.5$ time units, a traveller starting at node 1 should follow the path $\{1, 2, 3\}$ to maximise their arrival probability at node 3 because the arrival probability of this path is maximal for the given budget as shown in the left panel of Fig. 8 (b).

We assume that the travel time from node 3 to 4 is known to be 1.5 time units such that the travel time distribution is $p_{34}(t) = \delta(t - 1.5)$. Thus, the arrival CDFs of the paths $\{1, 3, 4\}$ and $\{1, 2, 3, 4\}$ are obtained by a shift of the CDFs of the paths $\{1, 3\}$ and $\{1, 2, 3\}$ by 1.5 time units as shown in the right panel of Fig. 8 (b). A traveller starting at node 1 should follow the path $\{1, 3, 4\}$ to maximise their arrival probability at node 4. Thus, which node should be chosen in the first step depends on the travel time distributions associated with further steps. It is not possible to consider subpaths independently and Bellman's principle is violated.

### Appendix B: Mixtures of convolutions

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

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

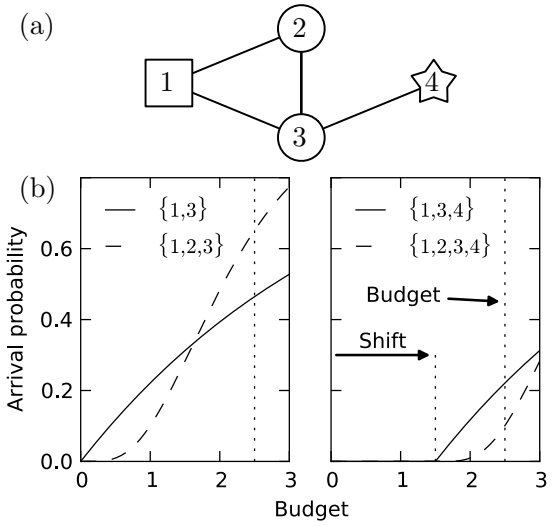


FIG. 8: (a) A network of four nodes. The origin of an a priori routing process is shown as a square and the target is shown as a star. (b) The left panel shows the arrival CDFs to reach node 3 via the paths $\{1, 3\}$ and $\{1, 2, 3\}$. The right panel shows the arrival CDFs to reach node 4 via the paths $\{1, 3, 4\}$ and $\{1, 2, 3, 4\}$. They are obtained by a shift of the CDFs shown in the left panel because $p_{34}(t) = \delta(t - 1.5)$ by assumption.

where $\omega_{f_i}$ and $\omega_{g_j}$ are the independent weights associated with the respective elements of $F$ and $G$. Taking the mixture after performing the convolution of the elements of $F$ and $G$ gives

$$\begin{aligned}
&\sum_{ij} \omega_{f_i}\omega_{g_j} (f_i * g_j)(z) \\
&= \int_0^z dx \sum_{ij} \omega_{f_i} f_i(z - x)\, \omega_{g_j} g_j(x) \\
&= \int_0^z dx\, \bar{f}(z - x)\, \bar{g}(x) \\
&= \left(\bar{f} * \bar{g}\right)(z)
\end{aligned}$$

Thus, provided that the assumption of independent weights holds, 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. On the one hand, carrying out the convolutions of all pairs of probability distributions in the sets first and taking the mixture afterwards requires $\left(\prod_{i=1}^b c_i\right)$ convolutions and additions. On the other hand, carrying out the mixtures first and performing the convolutions afterwards requires $b$ convolutions and $\left(\sum_{i=1}^b c_i\right)$ additions. It is thus much more efficient to compute the mixtures first and subsequently perform the convolutions.

### Appendix C: Pseudocode

This appendix contains pseudocode for three different optimality indices in Algorithm 1. Pseudocode for a discretised approximation to the convolution defined in Eq. (2) is shown in Algorithm 2. Algorithm 3 presents pseudocode for the decentralised routing algorithm on networks with stochastic edge weights introduced in Section IV B.


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

[2] H. Bar-Gera. Transportation network test problems. `http://www.bgu.ac.il/~bargera/tntp/`.

[3] S. Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[4] W.R. Taylor and C.A. Orengo. Protein structure alignment. *Journal of Molecular Biology*, 208(1):1–22, 1989.

[5] Y.Y. Fan, R.E. Kalaba, and J.E. Moore. Arriving on time. *Journal of Optimization Theory and Applications*, 127(3):497–513, 2005.

[6] Y.Y. Fan and Y.M. Nie. Optimal routing for maximizing the travel time reliability. *Networks and Spatial Economics*, 6(3):333–344, 2006.

[7] Y.M. Nie and X. Wu. Shortest path problem considering on-time arrival probability. *Transportation Research Part B: Methodological*, 43(6):597 – 613, 2009.

[8] D. Peleg and E. Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, 1989.

[9] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(2):72–93, 2005.

[10] R.E. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[11] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[12] L.C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[13] B. Brumitt. The road to better path-finding. `http://googleblog.blogspot.co.uk/2007/11/road-to-better-path-finding.html`, 2007.

[14] J.H. Lee, M.H. Kim, and Y.J. Lee. Information retrieval based on conceptual distance in is-a hierarchies. *Journal of Documentation*, 49(2):188–207, 1993.

[15] E.N. Mortensen and W.A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998.

[16] P. Berman. On-line searching and navigation. *Online Algorithms*, pages 232–241, 1998.

[17] R.B. Noland and J.W. Polak. Travel time variability: A review of theoretical and empirical issues. *Transport Reviews*, 22(1):39–54, 2002.

[18] A. Eiger, P.B. Mirchandani, and H. Soroush. Path preferences and optimal paths in probabilistic networks. *Transportation Science*, 19(1):75–84, 1985.

[19] H. Frank. Shortest paths in probabilistic graphs. *Operations Research*, 17(4):pp. 583–599, 1969.

[20] R.P. Loui. Optimal paths in graphs with stochastic or multidimensional weights. *Commun. ACM*, 26:670–676, September 1983.

[21] P.G. Hoel, S.C. Port, and C.J. Stone. *Introduction to probability theory*. Houghton Mifflin series in statistics. Houghton Mifflin, 1971.

[22] K.A. Small. The scheduling of consumer activities: Work trips. *The American Economic Review*, 72(3):pp. 467–479, 1982.

[23] M. Bousquet-Melou, A. J. Guttmann, and I. Jensen. Self-avoiding walks crossing a square. *Journal of Physics A: Mathematical and General*, 38(42):9159, 2005.

[24] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[25] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 163–170, New York, NY, USA, 2000. ACM.

[26] A. Slivkins. Distance estimation and object location via rings of neighbors. *Distributed Computing*, 19(4):313–333, 2007.

[27] M. Barthélemy. Spatial networks. *Physics Reports*, 499:1–101, February 2011.

[28] J. Kleinberg. Complex networks and decentralized search algorithms. In *Proceedings of the International Congress of Mathematicians: Madrid, August 22-30, 2006: invited lectures*, pages 1019–1044, 2006.

[29] C. Forbes, M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions*. John Wiley & Sons, 2011.

[30] B. Efron and R. Tibshirani. *An introduction to the bootstrap*. Monographs on statistics and applied probability. Chapman & Hall, 1993.

[31] C.E. Sigal, A.A.B. Pritsker, and J.J. Solberg. The stochastic shortest route problem. *Operations Research*, pages 1122–1129, 1980.

[32] Y.Y. Fan, R.E. Kalaba, and JE Moore, J.E. Shortest paths in stochastic networks with correlated link costs. *Computers & Mathematics with Applications*, 49(9-10):1549–1564, 2005.

**Algorithm 1** Three different criteria that select a successor node based on the arrival CDF $u(t)$, the neighbour $q(t)$ that realises the maximal arrival probability, a budget $\tau$ and a probability threshold $\theta$.

**function** FAN($u(t), q(t), \tau, \theta$)
  **return** $q(\tau)$
**end function**

**function** FRANK($u(t), q(t), \tau, \theta$)
  $t_c \leftarrow u^{-1}(\theta)$ % $u^{-1}$ denotes the inverse of $u$.
  **return** $q(t_c)$
**end function**

**function** JOINT($u(t), q(t), \tau, \theta$)
  $t_c \leftarrow u^{-1}(\theta)$
  **if** $t_c \leq \tau$ **then**
    **return** FRANK($u(t), q(t), \tau, \theta$)
  **else**
    **return** FAN($u(t), q(t), \tau, \theta$)
  **end if**
**end function**

**Algorithm 2** A naïve convolution algorithm that takes two discrete signals of equal lengths **a** and **b** as input parameters and computes their convolution using the trapezoidal rule.

**function** CONVOLVE($\mathbf{a}, \mathbf{b}$)
  $n \leftarrow$ **length of a**
  $\mathbf{c}[i] \leftarrow 0 \quad \forall i \in \{0, \ldots, n\}$
  **for** $i \in \{0, \ldots, n-1\}$ **do**
    **for** $j \in \{0, \ldots, i-1\}$ **do**
      $\mathbf{c}[i] \leftarrow \mathbf{c}[i] + \frac{1}{2}(\mathbf{a}[j]\mathbf{b}[i-j] + \mathbf{a}[j+1]\mathbf{b}[i-(j+1)])$
    **end for**
  **end for**
  **return c**
**end function**

**Algorithm 3** A decentralised routing algorithm based on the iterative approximation scheme developed by Fan and Nie [6]. The input parameters are a network $G$, an origin and a target, a time budget $\tau$, a probability threshold $\theta$ and a CRITERION to identify successor nodes.

**function** DECENTRALISEDROUTING($G$, origin, target, $\tau$, $\theta$, CRITERION)
  (current, traveltime) $\leftarrow (origin, 0)$
  steps $\leftarrow$ {(current, traveltime)}
  $N_V \leftarrow$ {current}
  **while** traveltime $\leq \tau$ **and** current $\neq$ target **do**
    $N_F \leftarrow i \qquad \forall \{(i,j) \in E : j \in N_V$ **and** $i \notin N_V\}$ % Obtain frontier nodes.
    $v_i^0(t) \leftarrow w_i^0(t) \leftarrow f(i, \text{target})(t) \qquad \forall i \in N_F$ % Initialise frontier nodes.
    $f_{\max}(t) = \max_{i \in N_F} [f(i, \text{target})(t)]$ % Obtain upper bound.
    $v_j^0(t) \leftarrow 0 \qquad \forall j \in N_V$ % Initialise visited nodes.
    $w_j^0(t) \leftarrow f_{\max}(t) \qquad \forall j \in N_V$
    unstable $\leftarrow N_V$
    $k \leftarrow 0$
    **while** current $\in$ unstable **do**
      $v_i^{k+1}(t) = \max_{j \in J_i} \left[\int_0^t p_{ij}(t') v_j^k(t - t')\, dt'\right] \qquad \forall i \in$ unstable % Update the sequences.
      $w_i^{k+1}(t) = \max_{j \in J_i} \left[\int_0^t p_{ij}(t') w_j^k(t - t')\, dt'\right] \qquad \forall i \in$ unstable
      $k \leftarrow k + 1$
      **for** $i \in$ unstable **do**
        **if** $|v_i^k(t) - w_i^k(t)| < \epsilon \qquad \forall t$ **then** % Check for convergence.
          **remove** $i$ **from** unstable
        **end if**
      **end for**
    **end while**
    $q_{\text{current}}(t) = \arg\max_{j \in J_{\text{current}}} \left[\int_0^t p_{\text{current},j}(t') v_j(t - t')\, dt'\right]$
    successor $\leftarrow$ CRITERION($v_{\text{current}}^k(t), q_{\text{current}}(t), \tau -$traveltime, $\theta$) % Obtain the successor node.
    traveltime $\leftarrow$ traveltime + **random sample of** $t_{\text{successor,current}}$ % Update the travel time.
    **add** successor **to** $N_V$ % Extend the discovered subgraph.
    current $\leftarrow$ successor % Make the step to the successor.
    **add** (current, traveltime) **to** steps
  **end while**
  **return** steps
**end function**