Path choice models on static graphs with random edge costs *

Anders Karlström†

Mogens Fosgerau‡

April 17, 2011

Abstract

We consider the cost minimising choice of path in a graph where edge costs are functions of edge specific characteristics, parameters to be estimated and edge specific i.i.d. random terms. We provide a way to simulate the likelihood by recursive importance sampling that circumvents the severe curse of dimensionality associated with large graphs. We provide an application to route choice in a road network involving more than 5000+ links.

*We are grateful for financial support the Danish Strategic Research Council. We acknowledge helpful comments from Emma Frejinger and Leonid Engelson. Computational assistance was provided by Per J. Olsson.
†Corresponding author: amail@kth.se
‡mf@transport.dtu.dk
1 Introduction

Our motivating example is a transportation network in which travellers are faced with the choice of route between given origins and destinations. There is a cost for every link in the network given in terms of a cost function. The link cost is a function of link specific observed characteristics, a link specific random term and parameters to be estimated. The link specific terms are i.i.d. across links. The link costs are observed by the travellers but are random from our perspective. The cost of a path connecting the origin to the destination is the sum of link costs over the path. Travellers are assumed to choose the cost minimising paths. Our problem is to infer the parameters of the link cost function. We present a consistent estimator for this purpose.

Our model is not specific to transportation networks. It describes also, e.g., the internet, where selfish agents route packages through a network to some destination server. In general, it applies to any situation where decision makers choose a path through a directed graph. In the language of graph theory, we say that a graph consists of nodes connected by directed edges. The nodes may indeed represent physical locations as in a transportation network. They may also represent locations in space-time, such that a node is a certain physical location at a certain time. Even more abstractly, nodes may represent states or perhaps even states at certain times and locations. Thus the model is quite general and able to describe a variety of situations.

The problem of estimating parameters involving shortest paths in this setting also appears in the context of time use economics and activity based modelling. For instance, daily scheduling decisions (including departure time, activity choice and mode choice) can naturally be formulated as choosing a shortest path in an abstract graph, see for instance Karlström (2006).
Our model is inherently static in that decision makers observe all error terms prior to making their choice of path. This is in contrast to the dynamic discrete choice literature where it is emphasized that agents gather information as they progress through a graph (refer Aguirregabiria and Mira JoE 2010). In return for giving up on the ability of the model to handle dynamics we obtain the ability to handle very large and complicated networks. To the best of our knowledge, we are the first to propose a way to consistently estimate models comprising large graphs.

Our estimator combines recursive importance sampling with a shortest path algorithm. It is the combination of these ideas that allows us to handle the very large choice sets that we encounter. Recursive importance sampling (RIS) is a standard technique for calculating multidimensional integrals, comprising the GHK simulator (ref) as a special case (Vijverberg, 1997). In our context, the recursive structure allows us to approximate the choice probability of an observed path without having to form the universal choice set of all paths between the origin and destination. The crucial feature is that it is only necessary to consider bounds on the error terms for a small number of links near the observed path.

The remainder of the network, further away from the observed path is handled efficiently through the use of a modified shortest path algorithm (Dijkstra). The main insight of this paper is how these ideas complement each other.

The paper also employs a third main idea to generate computational speed, namely conditional maximum likelihood (Cox, 1974). CML allows us to break long observed routes into two or more shorter routes, without sacrificing consistency. Using the Dijkstra algorithm, the computational effort reduces considerably as routes are getting shorter, irrespective of the size of the graph.

As noted above, the choice of route in a road network is an important case for the model discussed here. Route choice models have been the focus of active
research in the transportation engineering literature for many years and they have many applications. Route choice models are applied in transportation economics, e.g., for the evaluation of investments in network capacity or for analysis of congestion pricing. Previously, data collection for such models has been difficult, but this is being changed by the increasing availability of GPS data, which provide almost direct observation of actual route choices.

The estimation of route choice models has been a challenge for many years. The fundamental problem is that realistically sized networks are large. The number of links is easily into the thousands and the number of non-trivial paths connecting two points is then easily running into millions. In this paper we will estimate models that assign an error to each link in such networks and consider the choice of route among all possible paths. A notable advantage of such a model is that it naturally leads to a correlation structure across paths. This follows since the costs of overlapping paths are automatically correlated since they share the error terms for the links that they have in common. This is a significant advantage over previous models for the route choice problem. Ways to formulate and consistently estimate route choice models, without resorting to ad hoc reductions of the choice set or unrealistic restrictions on correlation structures, have been sought for at least since the 1970s.

This paper is organized as follows. In Section 2 we define the model to be

---

1As a specific case, the probability of choosing a particular route may be posed as a multinomial probit model (Daganzo, 1977).
2The engineering approach to route choice has a number of problems. Typically, the number of potential paths is arbitrarily reduced via the introduction of a so-called consideration set, which is somehow thought to represent the paths that are actually considered by travellers. Then some discrete choice model is formulated for the choice among these paths, either assuming independent paths or some ad hoc correlation structure.
estimated. In Section 3 we outline the recursive importance sampler estimator and the Dijkstra’s shortest path algorithm, and combine them to arrive at our proposed estimator. The consistency of the estimator is shown. In the following sections we demonstrate the computational feasibility of the estimator, and also show how the computational effort can be reduced by using the idea of partial likelihood. In the concluding section we give suggestions for future research.

2 A model for path choice in a directed graph

A graph $G$ is defined by directed edges $e \in \mathcal{E}$ connecting vertices $v \in \mathcal{V}$. The number of edges is $M = \#(\mathcal{E})$. Each directed edge is defined by its predecessor and successor vertices. Let $p : \mathcal{E} \rightarrow \mathcal{V}$ be the function that assigns a predecessor to each edge and let $s : \mathcal{E} \rightarrow \mathcal{V}$ be the function that assigns successors. We require that there is at the most one edge connecting any two vertices. Denote the set of outgoing edges from a vertex as $\Gamma^+(v) = \{e \in \mathcal{E} | s(e) = v\}$. A path $\pi$ of length $m$ is a sequence of $m$ edges $\pi = \{(e_1, ..., e_m) \in \mathcal{E}^m | s(e_k) = p(e_{k+1}) \forall k < m\}$. To each path $\pi = \{(e_1, ..., e_m)\}$ there is an origin $o(\pi) = p(e_1)$ and a destination $d(\pi) = s(e_m)$. A path $\pi$ may alternatively be described as a set of connected vertices $\{(v_0, ..., v_m) \in \mathcal{V}^{m+1} | v_0 = p(e_1), v_i = s(e_i), \forall e_i \in \pi\}$. Let the set of all paths from origin $o$ to destination $d$ in the graph $G$ be denoted $\Omega(o, d|G)$.

There are $n \in \mathbb{N}$ individuals. Let $z_n \in \mathbb{Z}$ denote a vector of individual specific socioeconomic variables. To each edge $e$ is associated a vector of edge characteristics $x_e \in \mathcal{X}$ and a random component $\epsilon_{en}$. The $\epsilon_{en} \in I = [0, 1]$ are i.i.d. standard uniform. Note that this is no loss of generality since any distribution of the random component can be obtained through the specification of $g$. Define
\( \epsilon_n = \epsilon_{en} \in I^M \) as the vector of all random components for a single individual.

Define an edge cost function

\[
g(\epsilon_{en}; x_e, z_n, \beta) > 0,
\]

where \( \beta \) is a vector of parameters to be estimated. In the following we suppress the notation for the individual specific variables. Then the cost associated with edge \( e \) is \( g(\epsilon_e; x_e, \beta) \).

The edge cost function \( g \) is strictly increasing in \( \epsilon \). The inverse function \( g^{-1}(\cdot; x, \beta) \) then exists for all fixed \( (x, \beta) \). Furthermore, the edge cost function is such that

\[
\forall C > 0 : Pr(g > C|x, \beta) > 0.
\]

These assumptions ensure that \( g \) exceeds any bound with positive probability and that this happens for large \( \epsilon \).

The cost associated with a path \( \pi \) is given additively as

\[
C(\pi) = \sum_{e \in \pi} g(\epsilon_e; x_e, \beta)
\]

and the cost of the shortest path from origin \( o \) to destination \( d \) in the graph \( G \) is denoted

\[
V(o, d; G) = \min_{\pi \in \Omega(o, d|G)} C(\pi)
\]

Each individual has a given origin and destination and chooses the cost minimising path. We observe this path, the edge characteristics \( x_e \) for all edges in the graph and the individual specific characteristics \( z_n \). Following the standard random utility framework, the random components \( \epsilon_{en} \) are observed by the individuals but not by us. We also observe the chosen path \( \pi^* \).

Given a path \( \pi \) connecting \( o \) and \( d \), define the set

\[
B(\pi) = \{ \epsilon \in I^M | \pi = \arg \min_{\sigma \in \Omega(o, d|G)} C(\sigma) \}.
\]
Then the probability of observing $\pi$ is $P(B(\pi))$. We assume that this model can be consistently estimated by maximum likelihood in principle. The aim of this paper is to address the approximation of $P(B(\pi))$, when the dimensionality of $G$ is large.

## 3 The estimator

### 3.1 Recursive importance sampling

We start with briefly reviewing the basic principle behind recursive importance sampling. Our objective is to approximate

$$ P = \int_B 1d\epsilon, \quad (2) $$

where $B = B(\pi_n^*)$ is the path chosen by some individual. Given some other density $\phi_B$ on the $M$-dimensional unit cube, with $\phi_B > 0$ for $\epsilon \in B$ and $\phi_B(\epsilon) = 0$ for $\epsilon \notin B$, we can write

$$ P = \int \frac{1}{\phi_B(\epsilon)} \phi_B(\epsilon) d\epsilon = E_{\phi_B}[\frac{1}{\phi_B(\epsilon)}]. $$

Now, taking $R$ random draws from the sampling distribution $\phi_B$, the desired integral can be approximated by

$$ \hat{P} = \frac{1}{R} \sum_{r=1}^{R} w_r, \quad (3) $$

where

$$ w_r = \frac{1}{\phi_B(\epsilon_r)} \quad (4) $$

and $\epsilon_j \sim \phi_B$. Note that since $E(w_j) = P$, it follows that $\hat{P}$ converges almost surely to $P$ as $R \to \infty$ (Geweke, 1989).
The issue that we are faced with is that the set $B$ may not be rectangular such that the integral in (2) is complicated to compute. Importance sampling provides the possibility of avoiding the complication of the set $B$ by using a constructed density $\phi_B$. This contributes to computational efficiency by avoiding draws outside $B$. But most importantly, $\phi_B$ may be constructed in a recursive way, which greatly facilitates computation.

Inspired by GHK, we draw sequentially from uniform distributions such that

$$\phi_B(\epsilon_1, \ldots, \epsilon_M) = \prod \phi_e(\epsilon_e),$$  \hfill (5)

where

$$\phi_e^c(x_e) = \frac{1}{Y_e^h - Y_e^l}$$  \hfill (6)

and $Y_e^h$ and $Y_e^l$ are computed based on the previous draws in the sequence such that $\phi_B$ is supported exactly on $B$.

Our recursive importance sampling procedure entails the following steps. (i) First, order the set of vertices in a convenient way depending on $B$. This is described below. (ii) Draw $\epsilon_1$. (iii) Calculate bounds $Y_2^l, Y_2^h$, which may typically depend on $\epsilon_1$. (iv) Draw $\epsilon_2 \in [Y_2^l, Y_2^h]$. (v) Continue sequentially, for each edge calculating the bounds $Y_k^l, Y_k^h$ and drawing $\epsilon_k$. (vi) Repeat steps (ii)-(v) in $R$ replications.

The target probability is then approximated by

$$\hat{P} = \frac{1}{R} \sum_{r=1}^{R} \prod_{k=1}^{M} (Y_k^h - Y_k^l)$$  \hfill (7)

### 3.2 Dijkstra shortest path

The previous section left open the issue of how to calculate the bounds $Y_k^l, Y_k^h$. The Dijkstra’s shortest path algorithm is useful for this purpose. It is one of the most important algorithm in computer science.
In the variant that is used in this paper, the algorithm starts at the destination and works itself towards the origin while maintaining a set of vertices $S$ that have been explored. In addition, the algorithm maintains a list of estimates of the cost $c(\cdot)$ of the shortest path from each vertex to the destination. Initially, $S$ is empty and the cost estimates are all infinity except the cost at the destination which is zero, $c(d) = 0$.

- At each stage of the algorithm, the vertex $u$ is chosen from $V \setminus S$ that provides the current least cost estimate. Each predecessor $v$ of $u$ is considered in turn. If the current cost estimate $c(v)$ is greater than $c(u) + g_e$, where $g_e$ is the cost on the edge connecting $u$ to the predecessor, then $c(v)$ is updated with the new value $c(u) + g_e$.

- When all predecessors of $u$ have been updated, $u$ is added to the set $S$ of explored vertices. At this stage $c(u)$ is the shortest path cost to the destination and $u$ will not be reconsidered by the algorithm.

- The algorithm continues until the origin is included in $S$. There may be vertices left in $V \setminus S$, but the shortest path cannot pass these vertices.

There are some caveats that the algorithm must take into account, for example there may not exist any path connecting the origin to the destination. Appendix ? provides a blabla... The following well-known Lemma (can be found in any textbook in Algorithms in Computer Science) describes two useful properties of the set $S$ at any stage of the algorithm.

**Lemma 1** The set of explored vertices $S$ has the following two properties: (i) $\max_{v \in S} c(v) \leq \min_{v \in V \setminus S} V(v)$, where $V(v)$ is the shortest path cost from $v$ to the destination. (ii) For all vertices $v \in S$ there is a shortest path from $v$ to the destination using only vertices of $S$ as intermediates.
3.3 A new combined algorithm

The strategy is now to use the Dijkstra algorithm to order the edges of the graph such that drawing error components from the set $B = B(\pi^*)$ becomes easy. Our algorithm begins by constructing a subgraph $G_0 = (V, E_0)$ where

$$E_0 = \pi^* \cup \left( E \setminus \bigcup_{v \in \mu^0} \Gamma^+(v) \right).$$

In other words, $G_0$ is the subgraph where all nonchosen outgoing edges from vertices on the chosen path are deleted. This is illustrated in Figure 3.3.

The next step is to apply the Dijkstra algorithm to the graph $G_0$ to calculate the set $S$ and the cost of the shortest path from each of the vertices $v \in S$ to the destination. Error components are drawn for the edges that are encountered during the algorithm. The Dijkstra algorithm stops when the origin is added to $S$. In a large network there may therefore be many vertices that are not considered, in particular if the chosen path is short. This is a significant advantage, since it means that the size of the relevant network is much reduced in each replication. We consider a dataset below for a network with more than 5000 edges, which means that we will be evaluating 5000 dimensional integrals. However, the median number of edges on the chosen paths is 15. Hence, the effective dimension of the integrals is much smaller than 5000.

Until this point, as we have applied Dijkstra algorithm to the graph $G_0$, the error components has been drawn under no restriction. That is, the bounds are simply $Y^l_e = 0$ and $Y^h_e = 1$. The chosen path is automatically the shortest path since by construction it is the only path from the origin to the destination in $G_0$. Therefore the draw so far is a point in the projection of $B$ on the dimensions corresponding to $E_0$.

Now it remains to calculate the bounds imposed by $B$ on the error components on the remaining edges in $E \setminus E_0$. These bounds are such that the chosen path
Figure 1: The figure shows how one edge at the time is added. The chosen path \( \pi^* = \{1, 2, 6\} \). The greyed out edges are not part of the graph, they are only there as a reference which edges will be subsequently added. Starting with the graph \( G_0 \) in (a), we first add the edge \( e_{23} \); sample \( \epsilon_{23} \) to get the graph \( G_1 \) in (b). Then we add one edge at the time until we have added all edges in (d), graph \( G_3 \).
remains optimal. Our algorithm now proceeds by adding the edges in \( E \setminus E_0 \); one by one in arbitrary order. Consider a particular edge \( e \in E \setminus E_0 \). We will add this edge and draw an error component, making sure that the chosen path is still the shortest from the origin to the destination. This implies only a restriction on how small the error component can be. The upper bound for the error component is always \( Y_e^h = 1 \). There are then three possible cases for the lower bound:

(i) \( s(e) \notin S \). The destination of the edge has not been included in \( S \). Therefore the cost from \( s(e) \) to the destination is at least as high as cost of the chosen path. Consequently, this vertex can not be part of the shortest path, regardless of the value of the error \( \epsilon_e \). The lower bound is then simply \( Y_e^l = 0 \).

(ii) \( s(e) \in S \) and \( c(s(e)) \geq c(p(e)) \). In this case, the cost from the edge’s destination is already higher than the shortest path from the edge’s origin using the chosen path \( \pi^* \). Therefore, the added edge can not be part of the shortest path and there will be no further restrictions on \( \epsilon_e \), hence the lower bound is \( Y_e^l = 0 \).

(iii) \( s(e) \in S \) and \( c(s(e)) < c(p(e)) \). In this case, addition of the edge \( e \) can potentially change the shortest path from the origin to the destination if the error component on \( e \) is sufficiently small. So we require that \( g(e \mid x_e, \beta) + c(s(e)) > c(p(e)) \). This imposes the restriction that \( \epsilon_e > Y_e^l \), where \( Y_e^l = g^{-1}(c(p(e)) - c(s(e)) \mid x_e, \beta) \).

In our small example in Figure 3.3, we add first the edge between vertex 2 and 3. Since the cost from vertex 3 is 4, which is higher than the cost to the destination from vertex 2, we identify case (ii) above, and there are no restrictions on the error component on the edge.

The process continues until all edges in \( E \setminus E_0 \) have been added illustrated in Figure 1(d).
It is important that the relevant lower $Y^l_e$ bounds are not affected as more edges are added and that all shortest path distances are also unaffected. These points are crucial in establishing that our proposed algorithm does in fact provide draws supported on $B(\pi^*_n)$. Theorem 1 below establishes these points and hence that our algorithm is a recursive importance sampler such that estimation of using simulated maximum likelihood does provide consistent estimates.

Note that the value of the error component on a new edge does not matter, provided it is larger than the calculated lower bound. There is therefore no point in actually drawing the new error components.

When all remaining edges have been added, we have reconstructed the original graph $G$, we have drawn a sequence of error components, and computed the bounds that are used to approximate the choice probability in (5). The steps above are repeated $R$ times. Taking the average of these replications we approximate the probability of the chosen path by (7).

Our proposed algorithm is stated in pseudocode in Algorithm 1 in the Appendix. The following theorem establishes formally that our proposed algorithm is indeed a recursive importance sampler. Hence simulated maximum likelihood estimation using this algorithm is consistent.

**Theorem 1** Our proposed algorithm is a recursive importance sampler and $\hat{P} \rightarrow P$ as $R \rightarrow \infty$.\(^3\)

Proof: We need to show that adding the edge $e \in E \setminus E_0$ and drawing an error component $\epsilon_e \in [Y^l_e, Y^h_e]$ does not change any shortest path distance. By construction, adding edge $e$ will not change $c(p(e))$. Consider a vertex $u$. The shortest path from $u$ to $p(e)$ will not include $e$ (otherwise there would be a loop).

\(^3\)It will be useful to know something about the rate of convergence. Is that available?
Hence, the shortest path from $u$ to $p(e)$ will not change. Since the shortest path from $p(e)$ is unaffected by the addition of $e$, so is then the shortest path from $u$.

The bounds $[Y_{e'}^l, Y_{e'}^h]$ for a new edge $e'$ depend only on $c(\cdot)$, which is unaffected by the addition of edges.

**Theorem 2** Under mild regularity conditions, stated in the appendix, simulated maximum likelihood using our estimator of the choice probabilities is consistent and asymptotically normal.

Proof: We appeal to Theorem 1 in McFadden 1989. Conditions A1–A9 are also assumed here. Condition A10 follows by letting the number of draws rise quickly as sample size increases (but how fast?). Condition A11 follows from McFadden’s Lemma 4 by noting that the simulated choice probability is uniformly bounded and almost surely twice continuously differentiable, since no shortest path will change for small variations in the parameter vector $\beta$. 

In some case it might be useful to observe that our model makes a partial likelihood approach naturally available (Cox, 1975). Consider an observed path $\pi^* = (o, ..., v, ..., d)$. $\pi^*$ is the shortest path from $o$ to $d$ for some realisation of the error component vector. Then also $\pi_1^* = (o, ..., v)$ and $\pi_2^* = (v, ..., d)$ are shortest paths, but for the same realisation of the error component vector. Hence maximum partial likelihood estimation replacing the observation $\pi^*$ by the observations $\pi_1^*$ and $\pi_2^*$ will be consistent but a modified estimator of the asymptotic variance may be needed.

### 4 Simulation example

In this section we report Monte Carlo evidence using a small network, drawn from Figure 3 in Frejinger et al (2009). Since loops do not impose any computational
difficulties, our graph has two links between each neighboring vertices, one in each direction.

We use the same origin and destination as Frejinger et al (ibid), and generate simulated data sets with 1000 observations. That is, we generate 1000 observed routes from our origin to the destination. The number of observations are considerably smaller than used in used in Frejinger et al (2009) or Karlstrom et al (2011), both of which use 3000 observations.

We use a simple route choice model specification, such that the cost for link $i$ is given by $C_i = \beta L + \epsilon_i$. The random terms $\epsilon_i$ are drawn from a normal distribution $N(0,1)$, truncated at zero such that all random terms are positive. There is only one parameter $\beta$ to be estimated, associated with the length $L$ of the link. Since the scale cannot be estimated, we will have to fix the scale of the random terms in the estimation. We will here report the scaled value of the estimated value.

We note that, although the number of observations are rather limited, the estimator does an excellent job in replicating the true parameters. However, as is well known, the SML estimator is not unbiased for finite number of draws ($R$). As is evident, the parameter is underestimated if too few draws are used.

<table>
<thead>
<tr>
<th>$R$</th>
<th>value$^a$</th>
<th>std dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.927</td>
<td>0.14</td>
</tr>
<tr>
<td>300</td>
<td>1.942</td>
<td>0.03</td>
</tr>
<tr>
<td>1000</td>
<td>2.017</td>
<td>TBC</td>
</tr>
<tr>
<td>2000</td>
<td>1.992</td>
<td>0.02</td>
</tr>
</tbody>
</table>

$^a$ True parameter: 2.00.

Table 1: Estimation results. These are preliminary results with 10-20 runs each.
5 Example from a real world network

TBC

6 Concluding remarks

In this paper we present a method for consistently estimate a route choice model in a static network with random edge cost. Such a model is allows for a realistic and flexible error structure across paths, but has previously not been possible to estimate consistently. We show that our proposed estimator is consistent and asymptotically normal, and also demonstrate that it is computational feasible to use on realistic data sets.

For future research, we note imulated maximum likelihood estimates are, in general, not unbiased for finite samples. In our simulations, it appears that the parameter estimates are downward biased in our simulation examples with only one parameter. Since it is costly to increase the number of simulation draws, it may also be useful to explore bias-correction methods, such as Lee (1999, 1997). This is left for future research.

7 Appendix

To show consistency and asymptotic normality of the recursive importance sampler, we have in Proposition 2 referred to Theorem 1 in McFadden (1989). This proof is omitted in this draft.
References


A Dijkstra

**Algorithm 1** Dijkstra shortest path

**Require:** Graph $G = (V, E)$

**Ensure:** SP, dist

1. for each vertex $v \in V$ do
2. \hspace{1em} dist[v] := infinity // Unknown distance function from v to destination
3. \hspace{1em} previous[v] := undefined // Previous node in optimal path to dest
4. \hspace{1em} dist[source] := 0 // Distance from dest to dest
5. \hspace{1em} Q := the set of all nodes in Graph, S = empty // Every vertex will be in either Q or S.
6. \hspace{2em} while source vertex $v_o \notin S$ do
7. \hspace{3em} u := vertex in Q with smallest dist[]
8. \hspace{3em} if dist[u] = infinity:
9. \hspace{4em} \hspace{1em} break // dest is inaccessible from all remaining vertices
10. \hspace{3em} \hspace{1em} remove u from Q
11. \hspace{3em} \hspace{1em} for each neighbor v of u: // where v has not yet been removed from Q.
12. \hspace{4em} \hspace{2em} alt := dist[u] + dist-between(u, v)
13. \hspace{4em} \hspace{2em} if alt < dist[v]:
14. \hspace{4em} \hspace{3em} dist[v] := alt
15. \hspace{4em} \hspace{3em} previous[v] := u
16. \hspace{4em} \hspace{3em} return dist[]
17. \hspace{2em} \hspace{1em} end while
18. \hspace{2em} end for
19. return dist[]
Algorithm 2 Calculate path probability

Require: Graph $G = (\mathcal{V}, \mathcal{E})$, path $\mu^o = (v_0, v_1, ..., v_m)$, $\pi^o = (e_1, \ldots, e_m)$ and then some $g_i$, $x_i$, $z_i$

Ensure: $\text{Prob}\{\mu^o = \arg\min \mu C^o(o(\mu^o), d(\mu^o); x, z)\}$

1: $\mathcal{E}_0 \gets \mathcal{E} \setminus \cup_{v \in \mu^o} \Gamma^+(v) + \mu^o$
2: $G_0 \gets (\mathcal{V}, \mathcal{E}_0)$
3: $SP, \mathcal{V} \leftarrow \text{DIJKSTRA}(\ldots)$
4: for $r \gets 1, 2, \ldots, R$ do
5: for $k \gets 0$ do
6: for each $v = v_{m-1}, \ldots, v_0$ do
7: for each $e \in \Gamma^+(v) | e \notin \pi^o$ do
8: for each outgoing edge not on the chosen path
9: $\hat{Y}^l \gets \max\{Y^{\min}_e, g_e^{-1}(V(v...) \setminus x_e, z_i)\}$
10: $p_k \gets 1 - \Phi(\hat{Y}^l)$
11: Draw $e_e \in [\hat{Y}^l, Y^{\max}]$
12: $k \leftarrow k + 1$
13: end for
14: end for
15: end for
16: $\hat{p} \leftarrow \frac{1}{R} \sum_{r=1}^{R} \prod_{k=1}^{k_{\max}} p_k$
17: return $\hat{p}$