Metropolis-Hastings sampling of paths

Gunnar Flötteröd [*] Michel Bierlaire [*]

June 6, 2011

Report TRANSP-OR 110606
Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
Ecole Polytechnique Fédérale de Lausanne
`transp-or.epfl.ch`

[*]École Polytechnique Fédérale de Lausanne (EPFL), School of Architecture, Civil and Environmental Engineering (ENAC), Transport and Mobility Laboratory (TRANSP-OR), CH-1015 Lausanne, Switzerland, {gunnar.floetteroed, michel.bierlaire}@epfl.ch

Abstract

We consider the previously unsolved problem of sampling cycle-free paths according to a given distribution from a general network. The problem is difficult because of the combinatorial number of alternatives, which prohibits a complete enumeration of all paths and hence also forbids to compute the normalizing constant of the sampling distribution. The problem is important because the ability to sample from a known distribution introduces mathematical rigor into many applications that range from route guidance to the estimation of choice models with sampling of alternatives.

1 Introduction

A path through a network is an essential element of a wide variety of models for transportation-related applications. Although mathematically well-defined, the concept of path is difficult to handle, mainly for two reasons. The first is operational. The combinatorially high number of paths between an origin and a destination in a decently large networks prohibits their enumeration. The second is a behavioral consequence of the first. Travelers, decision-makers, planners do not actually use the concept of a path to make decisions.

The first issue was addressed by researchers in the spirit of the Dijkstra (1959) algorithm based on implicit enumeration, which is the foundation of dynamic programming. This concept has been exploited in the context of traffic assignment (Dial, 1971, Russo and Vitetta, 2003), transit models (Nguyen et al., 1998), pedestrian models (Hoogendoorn and Bovy, 2004) or route choice models (Ben-Akiva and Bierlaire, 2003, Marzano and Papola, 2004).

The second issue has been analyzed by several researchers, introducing concepts such as cognitive mapping (Golledge, 1999) or spatial mental models (Taylor and Tversky, 1992). It has also generated a great deal of discussions in the route choice modeling literature, where the generation of the choice set considered by travelers is still an open problem.

Various choice set generation algorithms have been proposed, based on heuristics (De La Barra et al., 1993, Prato and Bekhor, 2006) or behavioral arguments (Ben-Akiva et al., 1984, Cascetta et al., 2002). However, none of these methods is satisfactory, as they all fail to generate choice sets which include the chosen path (Ramming, 2001, Bekhor et al., 2006, Prato and Bekhor, 2006).

Recently, Frejinger et al. (2009) proposed a new approach to circumvent the limitations of existing algorithms, based on an econometric argument. The choice set is supposed to contain all (possibly cycle-free) paths connecting the origin and the destination. Acknowledging that it does not correspond to a behavioral reality, they argue that it clearly solves the problem of missing the chosen path. Under this assumption, the curse of dimensionality generated by the combinatorially high number of paths in the choice set is relevant and must be addressed. Frejinger et al. (2009) propose to estimate the models based on a sample of alternatives, using the framework introduced by McFadden (1978) for logit models and generalized to multivariate extreme value models by Bierlaire et al. (2008).

As discussed in details by Frejinger and Bierlaire (2010), importance sampling of paths is a powerful method for route choice models but requires to have access to the sampling probabilities in order to obtain a consistent estimator. Frejinger and Bierlaire (2010) review several path generation algorithms and emphasize that deriving the sampling probability is in general infeasible. An important reason is that the derivation of the probability to sample a path usually involves the computation of a normalizing constant, which requires the enumeration of paths. The random walk method introduced by Frejinger et al. (2009) is an exception. Unfortunately, it may generate paths which are circuitous and may not bring much information to the choice model, requiring a great deal of computing efforts to generate a sufficient number of "useful" paths. This excludes its use for moderately large networks, as reported by Schüssler (2010). We refer the reader to Ben-Akiva and Bierlaire (2003), Bovy and Fiorenzo-Catalano (2007), Prato (2009) and Frejinger and Bierlaire (2010) for a review of route choice models and associated issues.

In this paper, we propose an algorithm for sampling cycle-free paths according to a given distribution. It is designed so that the analyst is in full control of the sampling protocol, and it is able to sample paths from a real network (we illustrate it with the Tel-Aviv network composed of about 17 000 links and 8000 nodes). The algorithm is not bound to route choice modeling applications and can be applied in any circumstance where a sample of paths following a predefined distribution is needed.

The method is an instance of the Metropolis-Hastings algorithm (Hastings, 1970). Indeed, this algorithm generates a Markov chain with a predefined stationary distribution, which can be defined in an unnormalized form. The absence of the normalizing constant is the key feature to avoid the enumeration of paths. The method is described in Section 2, and illustrated and validated in Section 3. Finally, Section 4 concludes the article.

2 Framework

Starting with a brief repetition of the generic Metropolis-Hastings algorithm in Subsection 2.1, a family of concrete instances of this algorithm for path generation is developed. Subsections 2.2 and 2.3 define its state space and target weights. The proposal distribution is first procedurally and then formally introduced in Subsections 2.4 and 2.5. Finally, some implementation notes are given in Subsection 2.6.

2.1 Generic Metropolis-Hastings algorithm

The Metropolis-Hastings (MH) algorithm creates a Markov chain (MC) with a predefined stationary distribution. This distribution can be defined in unnormalized form through positive weights $\{b(i)\}_{i \in \mathcal{S}}$ where $\mathcal{S}$ is the MC's finite state space and $b(i)$ is proportional to the stationary probability of state $i \in \mathcal{S}$. The MH algorithm further requires to define an irreducible proposal distribution $Q = (q(i, j))$ that defines the probability of proposing a transition from state i to state j. (Irreducibility is given if every state j can be reached from every state i through one or more transitions.) In every iteration of the algorithm, a proposal transition $i \to j$ is drawn according to Q, and then this proposal is accepted with a certain probability $\alpha(i, j)$ that is specified such that the desired stationary distribution is attained. Algorithm 1 specifies the generic MH algorithm.

Apart from the requirement of irreducibility, operational considerations affect the choice of the proposal distribution Q. If Q generates insufficient variability in that it creates long sequences of similar states, the MC needs

Algorithm 1 Metropolis-Hastings algorithm

```
1. Set iteration counter $k = 0$.
2. Select arbitrary initial state $i^k$.
3. Repeat beyond stationarity:
    (a) Draw candidate state $j$ from $\{q(i^k, j)\}_j$.
    (b) Compute acceptance probability $\alpha(i^k, j) = \min\left(\frac{b(j)q(j,i^k)}{b(i^k)q(i^k,j)}, 1\right)$.
    (c) With probability $\alpha(i^k, j)$, let $i^{k+1} = j$; else, let $i^{k+1} = i^k$.
    (d) Increase $k$ by one.
```

many iterations to cover the relevant part of the state space (where relevance is defined through the weights b). If Q generates too much variability, the MC frequently proposes jumps out of the relevant part of the state space, which results in low acceptance probabilities and long persistence in the same state.

Algorithm 1 requires to compute $q(j, i)/q(i, j)$ for every proposed transition $i \to j$, which should be possible in a computationally efficient manner due to the large number of such computations conducted. The probabilities of "self-loops", i.e., transitions of states to themselves, cancel out and hence need not be computed. This feature will subsequently be exploited in that it allows to forbid certain transitions (i.e, replace them by self-loops) without having to care about the probability with which these transitions occur.

2.2 State space definition

As the purpose of this work is to draw paths, the state space of the MH algorithm should comprise the set of all feasible paths in the considered network. We assume this network to be directed, and we are only interested in cycle-free paths, the finite number of which ensures a finite state space. (The approach is straightforward to generalize for a state space of paths with a given maximum number of cycles, e.g., by computing cycle-free paths in an accordingly expanded network.)

A path is defined as a sequence of nodes. The following notation is used:

$\mathcal{N}$	node set
Γ	a (cycle-free) path
$\|\Gamma\|$	number of nodes in path Γ
$\Gamma(a)$	the ath node of path Γ
$\Gamma^{-1}(v)$	the position of node $v \in \Gamma$ in Γ, between 1 and $\|\Gamma\|$
$\Gamma(a, b)$	sub-path of Γ between node positions $1 \leq a \leq b \leq \|\Gamma\|$
$\Gamma_1 + \Gamma_2$	concatenation of paths Γ_1 and Γ_2 (eliminates node duplications)
$\delta(\Gamma)$	the length (cost) of path Γ
$\mathrm{SP}(v, w)$	shortest path from node v to node w
$\mathrm{SP}_{\mathcal{M}}(v, w)$	shortest path from node v to node w, allowing only for nodes in $\mathcal{M} \subseteq \mathcal{N}$ to be traversed
$\delta_{\mathrm{SP}}(v, w)$	shortest path cost from node v to w
$\delta_{\mathrm{SP},\mathcal{M}}(v, w)$	shortest path cost from node v to w, allowing only for nodes in $\mathcal{M} \subseteq \mathcal{N}$ to be traversed

The representation of a path as a sequence of nodes renders the following requirement necessary.

Requirement 1. *The shortest path (SP) between the upstream node and the downstream node of every link runs across that link.*

This ensures that (i) every link can be covered by an SP and (ii) every pair of nodes is connected by at most one link. If a given network violates this requirement, it can be enforced by adding a dummy node in the center of every problematic link. The special case of two or more SPs of equal cost connecting two nodes is treated further below.

The state space is now defined as follows.

Definition 1. *The state space $\mathcal{S}$ of the MH algorithm consists of tuples (Γ, a, b, c) where Γ is a cycle-free path connecting origin and destination and a, b, c are integer numbers with $1 \leq a < b < c \leq |\Gamma|$.*

There are $\binom{|\Gamma|}{3} = |\Gamma|(|\Gamma| - 1)(|\Gamma| - 2)/6$ configurations of a, b, c for a given path Γ. This means that the ratio of the state space size to the number of paths is in the order of the average path length to the third power. The supplementary integer numbers are not strictly necessary but will turn out helpful to evaluate the proposal distribution Q.

Definition 1 excludes paths of less than three nodes from the state space, which, however, would only affect a single-link connection of origin and destination. Adding one dummy node in the center of that link would move it into the state space as well.

2.3 Target weights

The state space definition requires a proper handling of the target weights. Assume that the target weight for path Γ is $b(\Gamma)$. There are $|\Gamma|(|\Gamma|-1)(|\Gamma|-2)/6$ states that correspond to the path Γ. Hence, if $b(\Gamma)$ was used as the target weight of the MH algorithm, paths with many nodes would be overrepresented in the sample.

In order to eliminate the effect of the index variables (which are nothing but part of the algorithm's internal bookkeeping), the weight associated with state i containing path Γ is defined as

$$b(i) = \frac{b(\Gamma)}{|\Gamma|(|\Gamma|-1)(|\Gamma|-2)/6}. \tag{1}$$

The numerator represents the desired weighting, while the denominator eliminates the effect of the index parameters.

The definition of the target weights is application-dependent. In many circumstances, including the route choice modeling context, a natural choice of the weights b is some exponentially decreasing function $\exp[-\mu\delta(\Gamma)]$ of the path cost $\delta(\Gamma)$ where μ is a non negative scale parameter: $\mu = 0$ results in uniform weights that do not distinguish between paths of different cost, whereas $\mu \to \infty$ enforces that only the SP is considered. In this case, we obtain

$$b(i) = \frac{\exp[-\mu\delta(\Gamma)]}{|\Gamma|(|\Gamma|-1)(|\Gamma|-2)/6} \tag{2}$$

The weight specification (2) is used in all experiments of this article. Note that the cost based on which the weights are computed may be different from the cost that is used in the SP computations specified further below.

2.4 Procedural definition of proposal distribution

For reasons of computational efficiency, the proposal distribution relies heavily on SP computations. This calls for a second requirement.

Requirement 2. *For every node pair* (v, w) *that is connected by multiple SPs, the function* SP(v, w) *reproducibly returns one particular element out of this set. (The analogous statement holds for SPs on reduced node sets.)*

This calls for a unique ordering of paths: if there are multiple paths connecting v and w at the same minimum cost, exactly one of these paths is chosen according to this ordering. The underlying criterion may be arbitrary (e.g., lexicographic) but must yield reproducible results.

In the following, the proposal distribution Q is procedurally defined in terms of two operations, labeled SPLICE and SHUFFLE. These operations are first introduced independently; their combination into a joint proposal distribution is given afterward.

2.4.1 SPLICE operation

Given a current state $i = (\Gamma, a, b, c)$, a proposal state $i' = (\Gamma', a', b', c')$ is generated by the following operations:

1. Identify a set $\mathcal{V}(i)$ of possible "insertion nodes":
   (a) Let $\mathcal{N}_1(i) = \mathcal{N} \backslash \Gamma(1, a-1) \backslash \Gamma(c, |\Gamma|)$. This is the set of all nodes excluding nodes in Γ up to position a (exclusive) and as from position c (inclusive).
   (b) Let $\mathcal{N}_2(i) = \mathcal{N} \backslash \Gamma(1, a) \backslash \Gamma(c+1, |\Gamma|)$. This is the set of all nodes excluding nodes in Γ up to position a (inclusive) and as from position c (exclusive).
   (c) Let $\mathcal{V}(i) = \{v \in \mathcal{N} : \delta_{\mathrm{SP}, \mathcal{N}_1(i)}(\Gamma(a), v) + \delta_{\mathrm{SP}, \mathcal{N}_2(i)}(v, \Gamma(c)) < \infty\}$. This is the set of all nodes through which a path from $\Gamma(a)$ to $\Gamma(c)$ can be found that does not intersect Γ upstream of a or downstream of c.
2. If $\mathcal{V}(i)$ is the empty set:
   (a) Let $i' = i$, i.e., maintain the original state.
   (b) STOP.

If $\mathcal{V}(i)$ has at least one element:

(c) Draw an insertion node v from $\mathcal{V}(i)$ from some distribution $P(v|i)$ that is strictly positive for all $v \in \mathcal{V}(i)$. This distribution should prefer nodes that are "near" to the current path. A concrete instance of $P(v|i)$ is discussed in Section 2.5.1.

(d) Let $\Gamma_1 = SP_{\mathcal{N}_1(i)}(\Gamma(a), v)$, $\Gamma_2 = SP_{\mathcal{N}_2(i)}(v, \Gamma(c))$, and $\tilde{\Gamma} = \Gamma(1, a) + \Gamma_1 + \Gamma_2 + \Gamma(c, |\Gamma|)$. This is the original path with a detour from $\Gamma(a)$ to $\Gamma(c)$ that goes through v.

(e) if $\tilde{\Gamma}$ has no cycle:

   i. Let $\Gamma' = \tilde{\Gamma}$, $a' = a$, $b' = \tilde{\Gamma}^{-1}(v)$, $c' = b' + |\Gamma_2| - 1$. That is, take over the new path and adjust the indices such that a' and c' point at the same nodes as before and let b' point at the insertion node v.

   If $\tilde{\Gamma}$ has cycles:

   i. Let $i' = i$, i.e., maintain the original state.

Depending on i, this operation introduces either a detour or a shortcut into Γ, anchored at positions a and c. If neither is feasible, the original state is maintained. Figure 1 provides an example. Assume that the original state i consists of the path $\Gamma = \{A, B, C, D, E\}$ (solid arrows) and the location indices $(a, b, c) = (2, 3, 4)$. The newly drawn insertion node is G. It is connected to node $\Gamma(a) = B$ through $\{B, F, G\}$ and to node $\Gamma(c) = D$ through $\{G, D\}$. Replacing the segment $\{B, C, D\}$ in Γ by the resulting detour (dashed arrows) yields the new path $\Gamma' = \{A, B, F, G, D, E\}$. The new location indices are computed such that (i) the new indices $a' = 2$ and $c' = 5$ still identify the original "boundary" nodes $B = \Gamma'(a')$ and $D = \Gamma'(c')$ and (ii) $b' = 4$ points at the insertion node $\Gamma'(b') = G$.

This definition of the SPLICE operation forbids cycles in two different ways. The insertion node is linked to the original path via SPs on reduced node sets $\mathcal{N}_1(i)$ and $\mathcal{N}_2(i)$. This renders the computation of splice segments Γ_1 and Γ_2 that intersect the original path either upstream of a or downstream of c impossible. The suchlike forbidden "outer cycles" are shown in Figure 2. The solid lines are links of substantial, say unit, cost, and the dashed lines are of very small, say almost zero, cost. If the node sets $\mathcal{N} \backslash \mathcal{N}_1$ and

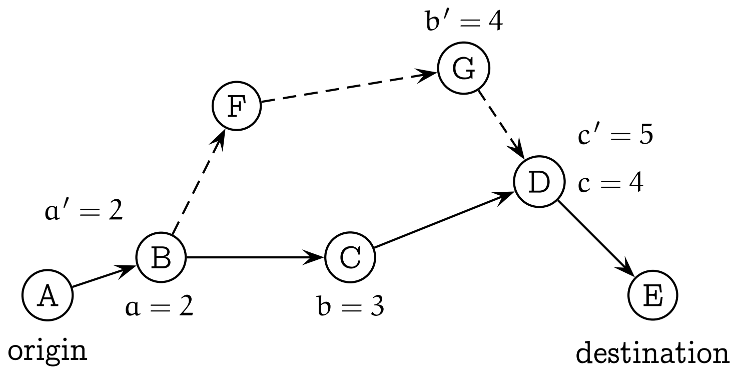


Figure 1: SPLICE example

$\mathcal{N}\backslash\mathcal{N}_2$ were not excluded from the SP calculations connecting a to b and b to c, cycles along the dashed lines could occur.

However, it still is possible that Γ_1 and Γ_2 intersect directly, leading to the "inner cycles" shown in Figure 3. Again, the solid lines are links of substantial, say unit, cost, and the dashed lines are of very small, say almost zero, cost. The exclusion of the node sets $\mathcal{N}\backslash\mathcal{N}_1$ and $\mathcal{N}\backslash\mathcal{N}_2$ from the SP calculations does not exclude cycles that result from a crossing of the inner path segments a to b and b to c along the dashed lines. If an inner cycle is constructed in step 2d, the original path is maintained according to step 2e, such that Γ' is guaranteed to be cycle-free. The rational for these two different treatments of cycles is both computational and conceptual; a detailed discussion is given further below.

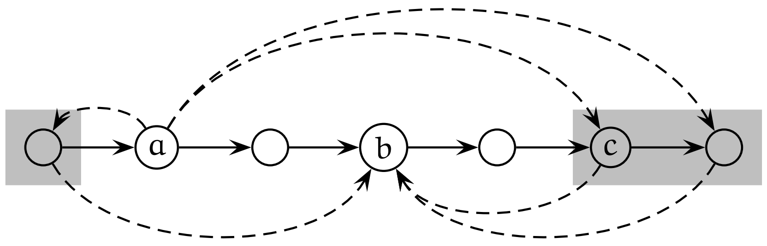
2.4.2 SHUFFLE operation

Given a current state $i = (\Gamma, a, b, c)$, a proposal state $i' = (\Gamma', a', b', c')$ is generated by the following operations:

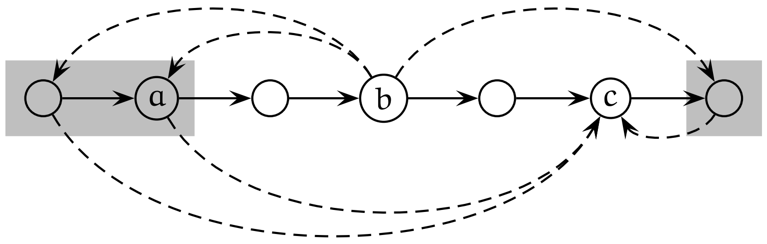
1. Draw a triplet (a', b', c') from some distribution $P(a', b', c'|\Gamma)$ with $P(a', b', c'|\Gamma) > 0$ for all $1 \leq a' < b' < c' \leq |\Gamma|$ and zero otherwise.
2. Let $\Gamma' = \Gamma$.

A concrete instance of $P(a', b', c'|\Gamma)$ is discussed in Section 2.5.2.

This operation merely shuffles the three integer state components but leaves the path unaffected. Its purpose is to randomly change the anchor points for the SP computations of the SPLICE operation.



(a) Outer cycle when going from a to b. Shaded: excluded nodes $\mathcal{N} \backslash \mathcal{N}_1$.



(b) Outer cycle when going from b to c. Shaded: excluded nodes $\mathcal{N} \backslash \mathcal{N}_2$.

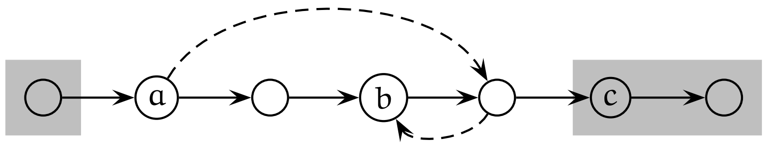
Figure 2: Typology of outer cycles

2.4.3 Combined proposal

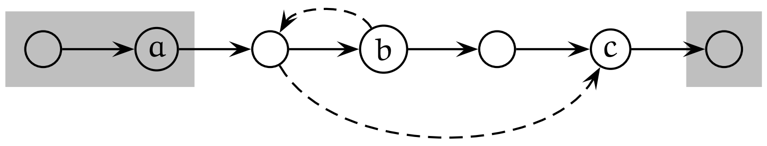
It remains to combine the SPLICE and SHUFFLE operations into one proposal. Since the simulation of any proposal transition $i \to j$ with $q(i,j) > 0$ also requires to compute the inverse transition probability $q(j,i)$ and since the $i \to j$ transition is invariably rejected if $q(j,i) = 0$ (cf. Algorithm 1), it is desirable to ensure that $q(i,j) > 0 \Leftrightarrow q(j,i) > 0$ holds, i.e., that a transition is invertible. This clearly would be the case if the chain was composed only of SHUFFLE operations. The SPLICE operation, however, exhibits this property only in certain situations.

Definition 2. *A SPLICE operation starting from* $i = (\Gamma, a, b, c)$ *is only invertible if the shortest path from* $\Gamma(a)$ *to* $\Gamma(b)$ *concatenated with the shortest path from* $\Gamma(b)$ *to* $\Gamma(c)$ *equals* $\Gamma(a,c)$*, i.e., if* $\mathrm{SP}(\Gamma(a),\Gamma(b)) + \mathrm{SP}(\Gamma(b),\Gamma(c)) = \Gamma(a,c)$*. Such a state* i *is called "spliceable". (The analogous statement holds for SPs on reduced node sets.)*

The SPLICE operation is only allowed if the chain is in a spliceable state (which is guaranteed to happen occasionally, as discussed in Section 2.5.4).



(a) Inner cycle when going from a to b. Shaded: excluded nodes $\mathcal{N}\backslash\mathcal{N}_1$.



(b) Inner cycle when going from b to c. Shaded: excluded nodes $\mathcal{N}\backslash\mathcal{N}_2$.

Figure 3: Typology of inner cycles

If it is allowed, the SPLICE operation is conducted with probability $w \in (0, 1)$. In every other case, a SHUFFLE is conducted. The combined proposal of going from a feasible state i to a feasible state i' is as follows:

1. If i is spliceable:
   (a) Draw u uniformly from $[0, 1)$.
   (b) If $u \leq w$: $i' = \text{SPLICE}(i)$.
   If $u > w$: $i' = \text{SHUFFLE}(i)$.

   If i is not spliceable: $i' = \text{SHUFFLE}(i)$.

A value of $w = 0.5$ is used in all experiments of this article.

2.5 Formal definition of proposal distribution

This section specifies the proposal probabilities that correspond to the procedural proposal definition given above. While the mere generation of proposals does not require such a specification, the computation of the respective acceptance probabilities is dependent on it.

2.5.1 SPLICE probability

The probability $q_{\mathrm{SPLICE}}(i, i')$ of going from a feasible state i to a feasible state i' with a SPLICE operation is

$$q_{\mathrm{SPLICE}}(i, i') = \begin{cases} P(\Gamma'(b')|i) & \text{if } i \neq i' \text{ are spliceable,} \\ & \Gamma'(1, a') = \Gamma(1, a), \\ & \Gamma'(c', |\Gamma'|) = \Gamma(c, |\Gamma|); \\ \phi_{\mathrm{SPLICE}}(i) & \text{if } i = i'; \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

This states that the probability of going from i to i' equals the probability $P(\Gamma'(b')|i)$ of having selected the node $\Gamma'(b')$ as the insertion node, given that the new path Γ' is composed of segments $\Gamma'(1, a')$, $\Gamma'(a', b')$, $\Gamma'(b', c')$, $\Gamma'(c', |\Gamma'|)$ that are consistent with an according SPLICE operation. The "self-loop" probability $\phi_{\mathrm{SPLICE}}(i)$ of staying in state i is treated further below. Self-loops occur not only when $\Gamma(b)$ happens to be selected as the insertion node but also when no candidate insertion nodes exist or an inner cycle is rejected.

The relative tractability of (3) is owed to three facts. First, if the index variables were not included in the state space but were generated on the fly, then there could be multiple SPLICE operations leading to the same path. The probabilities of all of these proposals would have to be summed up in order to obtain the total proposal probability. Second, if requirement 2 did not call for a unique SP operation, then one out of possibly many SPs would have to be drawn from a well-defined distribution across all SPs, requiring their enumeration. Third, the so far omitted evaluation of $\phi_{\mathrm{SPLICE}}(i)$ would require to sum up the probabilities of all aforementioned events that lead to a self-loop.

A concrete instance of the node selection probability $P(v|i)$ is

$$P(v|i) = \begin{cases} \frac{\exp[-\tilde{\mu}(\delta_{\mathrm{SP},\mathcal{N}_1(i)}(\Gamma(a),v)+\delta_{\mathrm{SP},\mathcal{N}_2(i)}(v,\Gamma(c)))]}{\sum_{w\in\mathcal{V}(i)} \exp[-\tilde{\mu}(\delta_{\mathrm{SP},\mathcal{N}_1(i)}(\Gamma(a),w)+\delta_{\mathrm{SP},\mathcal{N}_2(i)}(w,\Gamma(c)))]} & \text{if } v \in \mathcal{V}(i) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\tilde{\mu}$ is a non-negative scale parameter that controls how distant from Γ the insertion node is drawn. "Distance" corresponds here to the length of the total inserted path segment. The distribution (4) is used in all experiments of this article.

2.5.2 SHUFFLE probability

The probability $q_{\text{SHUFFLE}}(i, i')$ of going from a feasible state i to a feasible state i' with a SHUFFLE operation is

$$q_{\text{SHUFFLE}}(i, i') = \begin{cases} P(a', b', c'|\Gamma) & \text{if } i \neq i',\ \Gamma = \Gamma'; \\ \phi_{\text{SHUFFLE}}(i) & \text{if } i = i'; \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Again, $\phi_{\text{SHUFFLE}}(i)$ is the yet unspecified self-loop probability.

A concrete instance of the index selection probability $P(a', b', c'|\Gamma)$ is

$$P(a', b', c'|\Gamma) = \begin{cases} \frac{1}{|\Gamma|(|\Gamma|-1)(|\Gamma|-2)/6} & \text{if } 1 \leq a' < b' < c' \leq |\Gamma| \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

This uniform distribution is used in all experiments of this article.

2.5.3 Combined proposal distribution

The proposal distribution $Q = (q(i, i'))$ is now defined as

$$q(i, i') = \begin{cases} P_{\text{SPLICE}}(i) \cdot q_{\text{SPLICE}}(i, i') + (1 - P_{\text{SPLICE}}(i)) \cdot q_{\text{SHUFFLE}}(i, i') & \text{if } i \neq i' \\ \phi(i) & \text{otherwise} \end{cases} \tag{7}$$

where

$$P_{\text{SPLICE}}(i) = w \cdot \mathbf{1}(i \text{ is spliceable}) \tag{8}$$

with $\mathbf{1}(\cdot)$ being the indicator function and $w \in (0, 1)$ being 0.5 in all experiments of this article.

This definition is complete apart from the self-loop probability $\phi(i) = P_{\text{SPLICE}}(i) \cdot \phi_{\text{SPLICE}}(i) + (1 - P_{\text{SPLICE}}(i)) \cdot \phi_{\text{SHUFFLE}}(i)$. Since it has already been observed that the probabilities of self-loops are not needed to run the MH algorithm, neither ϕ nor ϕ_{SPLICE} and ϕ_{SHUFFLE} (which only appear in the computation of ϕ) are specified.

2.5.4 Irreducibility

It remains to show that the proposed specification of Q is irreducible. For this, we first consider the following transition sequence from an SP to an arbitrary cycle-free target path Γ^*:

1. Let $\Gamma = \mathrm{SP}(\Gamma^*(1), \Gamma^*(|\Gamma^*|))$.
2. For $\beta = 2 \ldots |\Gamma^*| - 1$:
   (a) SHUFFLE such that $a = \beta - 1$ and $c = |\Gamma|$.
   (b) SPLICE with insertion node $\nu = \Gamma^*(\beta)$.

This process builds the target path Γ^* starting from an SP forwards through the iterations. As from step 1, $\Gamma(1) = \Gamma^*(1)$ and $\Gamma(|\Gamma|) = \Gamma^*(|\Gamma^*|)$. In every iteration, the SHUFFLE leads to a configuration where a is at the downstream end of that part of Γ that already overlaps with Γ^*: $\Gamma(1, a) = \Gamma^*(1, a)$. The SPLICE operation then adds the immediate next target node $\Gamma^*(\beta)$ with $\beta = a + 1$. This operation is always feasible for the following reasons: (i) The current path is always spliceable because b is always located on the SP between a and c; (ii) no inner cycle is possible because a and β are adjacent; (iii) at least the target path constitutes one feasible connection from $\Gamma^*(\beta)$ to $\Gamma(c)$. After $|\Gamma^*| - 2$ iterations, the target path is constructed with positive probability.

The transition from an arbitrary path Γ_1 to another arbitrary path Γ_2 follows from the following observations: (i) the transition $\Gamma_1 \to \Gamma_2$ can always be decomposed into $\Gamma_1 \to \mathrm{SP} \to \Gamma_2$, (ii) the transition $\mathrm{SP} \to \Gamma_2$ has just been demonstrated; (iii) the transition $\Gamma_1 \to \mathrm{SP}$ is just as feasible as the transition $\mathrm{SP} \to \Gamma_1$ because of the invertibility of the proposal distribution. Hence, Q is irreducible.

2.6 Implementation notes

The proposed method is implemented in the Java programming language (`java.sun.com`). This section elaborates on some aspects of its implementation.

2.6.1 Logarithmic probabilities

Given the size of its state space, the MH algorithm deals with extremely small numbers when computing acceptance probabilities from weights and proposal probabilities. These numbers are therefore stored and processed in logarithmic form whenever possible. In consequence, the acceptance probability $\alpha(i,j)$ in Algorithm 1 is computed from

$$\ln \alpha(i,j) = (\ln b(j) - \ln b(i)) + (\ln q(j,i) - \ln q(i,j)) \tag{9}$$

and then compared to the logarithm of a uniform random number. The brackets on the right-hand side collect quantities of similar order of magnitude (weights and proposal probabilities).

2.6.2 SP cost computations

The SPLICE operation requires a large number of SP cost computations, which account for most of the program's computational burden. Implementing SPLICE steps 2c through (4) requires to compute the SP cost from the node at path position a to almost all other network nodes as well as the cost from almost all network nodes to the node at path position c. Since some of the current path's nodes are excluded from these computations according to SPLICE step 1, the resulting SP tree costs are path-dependent and hence can hardly be re-used in later iterations.

For the purposes of this article, a naive implementation is chosen that computes both SP tree cost structures from scratch whenever necessary. An arguably more efficient approach would be to cache the SP tree costs and to merely update them such that they reflect the exclusion of nodes according to SPLICE step 1. Since the outer cycles of Figure 2 that are avoided by the node exclusions occur relatively infrequently if no nodes are excluded, it is conjectured that the subtree that would need to be updated is relatively small as well.

2.6.3 Unique SP ordering

Requirement 2 calls for a unique ordering of paths. Unique SPs in SPLICE step 2d are computed by traversing the upstream and downstream SP tree cost data structures in a unique order while building an SP along the way. (Essentially the same logic is used to check for spliceability.) This unique tree traversal order is established through a unique ordering of the ingoing/outgoing links of every node when looking for the next up-stream/downstream node of the SP under construction.

2.6.4 Network preprocessing

As the length of a path increases, its probability of ever being reached by the chain approaches zero. However, without further precautions, faraway nodes are still accounted for in the computation of the SP tree cost data structures, which implies a large computational overhead for OD pairs that are "close" compared to the network size. In such cases, the network can be preprocessed, trading a minimal loss in precision against a drastic gain in computational performance.

Assuming exponential path weights $\exp[-\mu\delta(\Gamma)]$, the weight ratio ε between a path of cost $\eta\delta_{\text{SP}}$, $\eta \geq 1$, and the SP cost δ_{SP} is $\varepsilon = \exp[\mu\delta_{\text{SP}}(1-\eta)]$. In turn, the coefficient η at which a particular ε threshold is reached is

$$\eta = 1 - \frac{\ln \varepsilon}{\mu\delta_{\text{SP}}}. \tag{10}$$

Based on this number, the network is preprocessed as follows. All nodes ν with $\delta(\text{origin}, \nu) + \delta(\nu, \text{destination}) > \eta \cdot \delta_{\text{SP}}(\text{origin}, \text{destination})$ are removed. Isolated nodes are then also removed in order to ensure full connectivity. Finally, only such links are retained for which both the upstream and downstream node are still in the network. A parameter value of $\varepsilon = 10^{-9}$ is used in all experiments of this article.

3 Experiments

Two experiments are presented. The simple example of Section 3.1 allows for a path enumeration such that the precision of the proposed algorithm can be analyzed. The Tel-Aviv example of Section 3.2 then demonstrates the algorithm's feasibility for large real-world networks.

3.1 Simple example

The first series of experiments is based on the synthetic network of Frejinger et al. (2009). This is a cycle-free network with 64 links, 38 nodes, and 170 paths connecting the origin (node number 1) to the destination (node number 38). It is shown in Figure 4. The loop-free network layout ensures that a random walk reaches the destination in at most 12 steps.

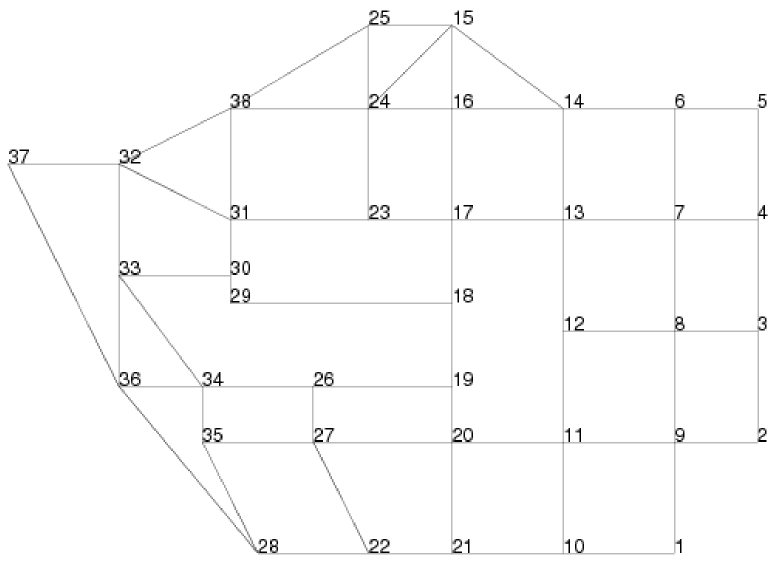


Figure 4: Simple test network

Three experiments are conducted, with $\mu = \tilde{\mu}$ taking values of zero, two, and four. Each experiment runs the MH algorithm for 10^8 iterations. The link cost based on which the weights and the SPs are computed is defined as the geometrical link length. The full network is considered in that the preprocessing of Subsection 2.6.4 is omitted.

Due to the locality of its transition distribution, the MH algorithm typically generates similar states in subsequent iterations, with this similarity ceasing as the iteration distance between two states increases. This effect is quantified through the similarity measure

$$\phi(d) = \frac{1}{K}\sum_{k=1}^{K} \frac{|\Gamma^k \cap \Gamma^{k+d}|}{\frac{1}{2}(|\Gamma^k| + |\Gamma^{k+d}|)} \tag{11}$$

where $|\Gamma^k \cap \Gamma^{k+d}|$ is the number of identical nodes in the paths generated in iterations k and $k + d$. The results are shown in Figure 5. From these plots, it can be concluded that the extraction of every 2500ths path leads to independent samples: $\phi(d)$ decreases monotonically with d, which means that the average similarity of paths decreases with their distance in the chain. In all three subfigures, the similarity stabilizes around 0.5. This is so because the network is quite constrained, and even completely random paths are still likely to overlap. However, since an increase of d beyond, say, 2000, does not result in a notably decreased $\phi(d)$, the extraction of every 2500th path as an independent sample is justified. Since the chain is initialized with the SP, which is already well centered in the target distribution, the first 2500 iterations are discarded before stationarity is assumed and samples are extracted.

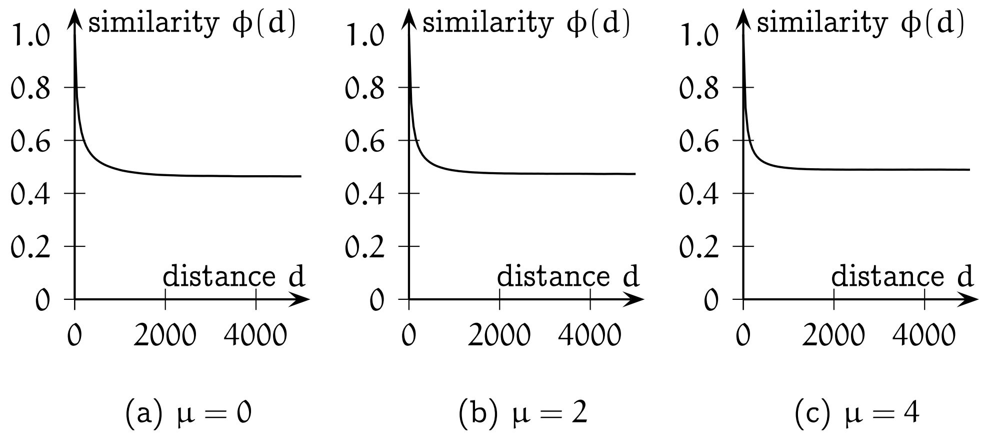


Figure 5: Distance-dependent path similarity in simple example

The ability of the MC to create the desired target distribution is demonstrated in Figure 6. Each subfigure contains 170 points, one per path, where the x-coordinate represents the target probability of the respective path and the y-coordinate represents the frequency of this path in the sample. The points are symmetrically distributed around the main diagonal, which indicates an unbiased reproduction of the target distribution. The remaining scatter is due to the fact that the empirical frequencies are obtained from a finite sample. For $\mu = 0$, all paths have the same target probability, whereas for $\mu = 4$, clear differences are made between paths of different length.

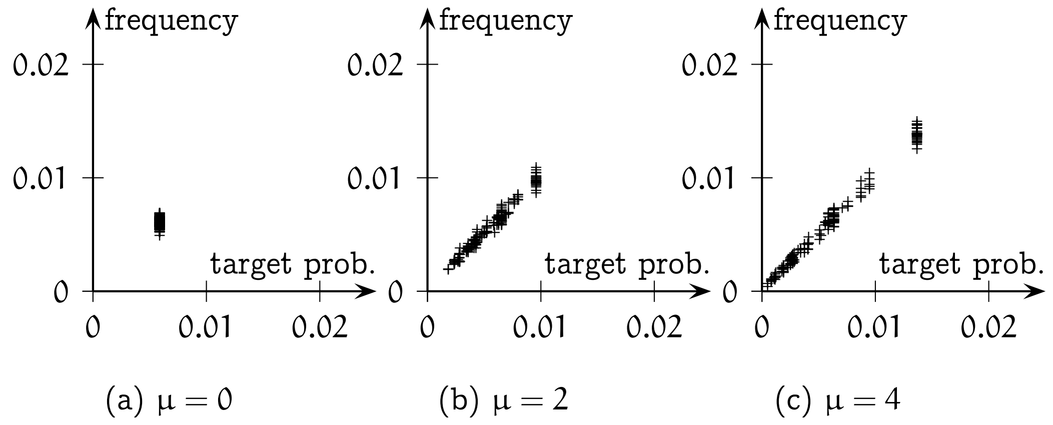


Figure 6: Scatter plots in simple example

The consistency of the samples with the target distribution is quantified through the X^2 statistic

$$X^2 = \sum_{c=1}^{C} \frac{(E_c - O_c)^2}{E_c} \tag{12}$$

where c indexes the C discrete outcomes ($C = 170$ in the present case), E_c is the expected number of realizations of category c, and O_c is the respective observed number. For independent draws, the X^2 statistic is asymptotically χ^2 distributed with $C - 1$ degrees of freedom. The asymptotic approximation is valid if the expected frequency of not too many categories falls below five. In the present experiments, the smallest expected frequency is 19.98, such that the χ^2 distribution is fully justified.

The 90% and 95% quantiles of a χ^2 distribution with 169 degrees of freedom are 192.95 and 200.33. The experimentally obtained X^2 statistics are 156.74, 188.28, and 164.42 for μ values of zero, two, and four. Given that all statistics are below the 90 % percentile of the χ^2 distribution, the hypothesis of consistency between desired and realized distribution can safely be accepted.

3.2 Complex example

These experiments are based on the road network of Tel-Aviv, Israel, which is shown in Figure 7. It consists of 17 118 links and 7 879 nodes. The considered origin/destination (OD) relation is located in the densely meshed network of the city center, which allows for a huge number of possible paths and opportunities for loops. It can be identified in Figure 8.

The MH algorithm is run with parameter settings of $\mu = 0.01$, $\mu = 0.02$, $\mu = 0.04$, $\tilde{\mu} = \mu$, and 10^7 iterations per experiment. Every chain is started with the SP. The network is preprocessed as described in Subsection 2.6. The resulting subnetworks consist of 2987/1627/802 links and 1362/774/395 nodes for $\mu = 0.01/0.02/0.04$.

Figure 8 shows these subnetworks in light gray color. It also shows, for each experiment, the set of all links that are contained in at least one sampled path in black color. The subnetworks are large enough not to bias the path sampling in that the chains hardly get to the subnetwork boundaries, meaning that they are unaffected by the exclusion of nodes and links beyond those boundaries. Some black links appear isolated because of overlapping links; all generated paths constitute proper connections of origin and destination.

The distance-dependent path similarities, again evaluated according to (11), are shown in Figure 9. These diagrams indicate that the extraction of every, say, 10 000th path from the chain yields independent draws. Overall, one observes that the lower μ gets, the larger the domain of relevant paths becomes, and hence the interval between extracted paths needs to become larger in order to ensure independent draws from this increasingly large domain. The similarity of independently drawn paths increases with μ because this reduces the size of the domain of relevant paths, leading to a greater overlap on average.

Due to the enormous number of possible paths in this test case, the generation of scatter plots like those shown in Figure 6 or the computation of X^2 statistics is infeasible.

The experiments were conducted on a Dell PowerEdge R710 server with 2 Intel Xeon X5680 processors running at 3.33 GHz. Table 1 summarizes the computational performance of the proposed algorithm in the Tel-Aviv

Table 1: Computational performance of Tel-Aviv test case

μ	nodes in subnetwork	links in subnetwork	computation time for 10^5 iterations	computational share for SPs
0.04	802	395	47 seconds	77 %
0.02	1627	774	109 seconds	80 %
0.01	2987	1362	196 seconds	91 %

network. Overall, the computation time for 10^5 iterations varies from 47 over 109 to 196 seconds for μ values of 0.04, 0.02, and 0.01. The algorithm scales roughly linearly with the size of the considered subnetwork, which depends on μ, as explained in Subsection 2.6.4. Between 77 % and 91 % of the run time are spent computing SPs. This indicates clear potential for further performance tuning based on an SP update procedure that replaces the currently implemented naive re-computation of the SP tree costs in every iteration, cf. Subsection 2.6.2.

4 Conclusion

We have presented an efficient Metropolis-Hastings algorithm to sample cycle-free paths according to an arbitrary distribution. The key methodological contribution is the definition of the state space and the proposal distribution that match the requirement of the MH framework.

The method is validated on a small network, where it is shown that the frequency of the generated paths correspond to the target distribution. Its applicability on a real network has been demonstrated, using a prototype implementation. Given the good computational speed of this prototype and the possible room for improvement of its implementation, in particular in terms of recycling shortest paths, the method is definitely adequate for real applications.

The method is clearly well suited for the generation of choice sets for route choice models. However, its specification is independent of this application, making it appropriate for any context where sampling of paths is necessary.

A typical example appears in map matching algorithms in the presence of

GPS data of poor quality. Newman et al. (2009) propose a method that associate each path in the network with the probability that the data has been generated by a vehicle following this path. A probabilistic map matching algorithm would consist in sampling paths based on this distribution.

The proposed method can also be naturally integrated with iterated simulation approaches to the dynamic traffic assignment (DTA) problem. These simulations typically run a Markov chain until stationarity, where in every iteration a demand simulator and a supply simulator is evaluated (Flötteröd et al., forthcoming). However, the choice set generation in the demand model of all DTA simulators known to the authors is ad hoc, without a clear understanding (let alone control) of the resulting choice sets. If the proposed algorithm was combined with the DTA simulation into one joint Markov chain, choice sets could be generated from a well-defined distribution that even could account for the network conditions upon convergence of the simulation (Nagel and Flötteroöd, forthcoming).

In addition to these applications, future research will involve adapting our method to multi-modal networks, where the underlying structure may be exploited in the definition of the splice operation.

5 Acknowledgment

We would like to thank Shlomo Bekhor to give us access to the Tel-Aviv network data.

References

Bekhor, S., Ben-Akiva, M. and Ramming, M. (2006). Evaluation of choice set generation algorithms for route choice models, *Annals of Operations Research* **144**(1): 235–247.

Ben-Akiva, M., Bergman, M., Daly, A. and Ramaswamy, R. (1984). Modelling inter urban route choice behaviour, *Proceedings of the Ninth International Symposium on Transportation and Traffic Theory, Delft, the Netherlands, 11-13 July 1984*, p. 299.

Ben-Akiva, M. and Bierlaire, M. (2003). Discrete choice models with applications to departure time and route choice, *in* R. Hall (ed.), *Handbook of Transportation Science, Second Edition*, Kluwer Academic Publishers, Boston/Dordrecht/London, pp. 7–37.

Bierlaire, M., Bolduc, D. and McFadden, D. (2008). The estimation of generalized extreme value models from choice-based samples, *Transportation Research Part B: Methodological* **42**(4): 381–394.

Bovy, P. and Fiorenzo-Catalano, S. (2007). Stochastic route choice set generation: behavioral and probabilistic foundations, *Transportmetrica* **3**(3): 173–189.

Cascetta, E., Russo, F., Viola, F. A. and Vitetta, A. (2002). A model of route perception in urban road networks, *Transportation Research Part B: Methodological* **36**(7): 577 – 592.

De La Barra, T., Perez, B. and Anez, J. (1993). Multi-dimensional path search and assignment, *Proceedings of the 21st PTRC Summer Meeting*, pp. 307–319.

Dial, R. (1971). A probabilistic multipath traffic assignment model which obviates path enumeration, *Transportation Research* **5**.

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

Flötteröd, G., Bierlaire, M. and Nagel, K. (forthcoming). Bayesian demand calibration for dynamic traffic simulations, *Transportation Science* **xx**(xx): xx–xx.

Frejinger, E. and Bierlaire, M. (2010). On path generation algorithms for route choice models, *in* S. Hess and A. Daly (eds), *Choice Modelling: The State-of-the-Art and the State-of-Practice*, Emerald Group Publishing Limited, pp. 307–315. ISBN:978-1-84950-772-1.

Frejinger, E., Bierlaire, M. and Ben-Akiva, M. (2009). Sampling of alternatives for route choice modeling, *Transportation Research Part B: Methodological* **43**(10): 984–994.

Golledge, R. (1999). *Wayfinding behavior: Cognitive mapping and other spatial processes*, Johns Hopkins Univ Pr.

Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications, *Biometrika* **57**(1): 97.

Hoogendoorn, S. P. and Bovy, P. H. L. (2004). Pedestrian route-choice and activity scheduling theory and models, *Transportation Research Part B: Methodological* **38**(2): 169 – 190.

Marzano, V. and Papola, A. (2004). A link based path-multilevel logit model for route choice which allows implicit path enumeration, *Proceedings of the European Transport Conference.*

McFadden, D. (1978). Modelling the choice of residential location, *in* A. Karlquist *et al.* (ed.), *Spatial interaction theory and residential location*, North-Holland, Amsterdam, pp. 75–96.

Nagel, K. and Flötteroöd, G. (forthcoming). Agent-based traffic assignment: going from trips to behavioral travelers, *in* C. Bhat and R. Pendyala (eds), *Travel Behaviour Research in an Evolving World*, Emerald Group Publishing, Bingley, United Kingdom.

Newman, J. P., Chen, J. and Bierlaire, M. (2009). Generating probabilistic path observation from gps data for route choice modeling, *Proceedings of the European Transport Conference (ETC) 5-7 October 2009.*

Nguyen, S., Pallottino, S. and Gendreau, M. (1998). Implicit enumeration of hyperpaths in a logit model for transit networks, *TS* **32**(1).

Prato, C. (2009). Route choice modeling: past, present and future research directions, *Journal of Choice Modelling* **2**(1): 65–100.

Prato, C. and Bekhor, S. (2006). Applying branch-and-bound technique to route choice set generation, *Transportation Research Record: Journal of the Transportation Research Board* **1985**(-1): 19–28.

Ramming, M. (2001). *Network knowledge and route choice*, PhD thesis, Citeseer.

Russo, F. and Vitetta, A. (2003). An assignment model with modified logit, which obviates enumeration and overlapping problems, *Transportation* **30**: 177–201. 10.1023/A:1022598404823.

Schüssler, N. (2010). *Accounting for similarities between alternatives in discrete choice models based on high-resolution observations of transport behaviour*, PhD thesis, ETH Zurich, Switzerland.

Taylor, H. A. and Tversky, B. (1992). Spatial mental models derived from survey and route descriptions, *Journal of Memory and Language* **31**(2): 261 – 292.

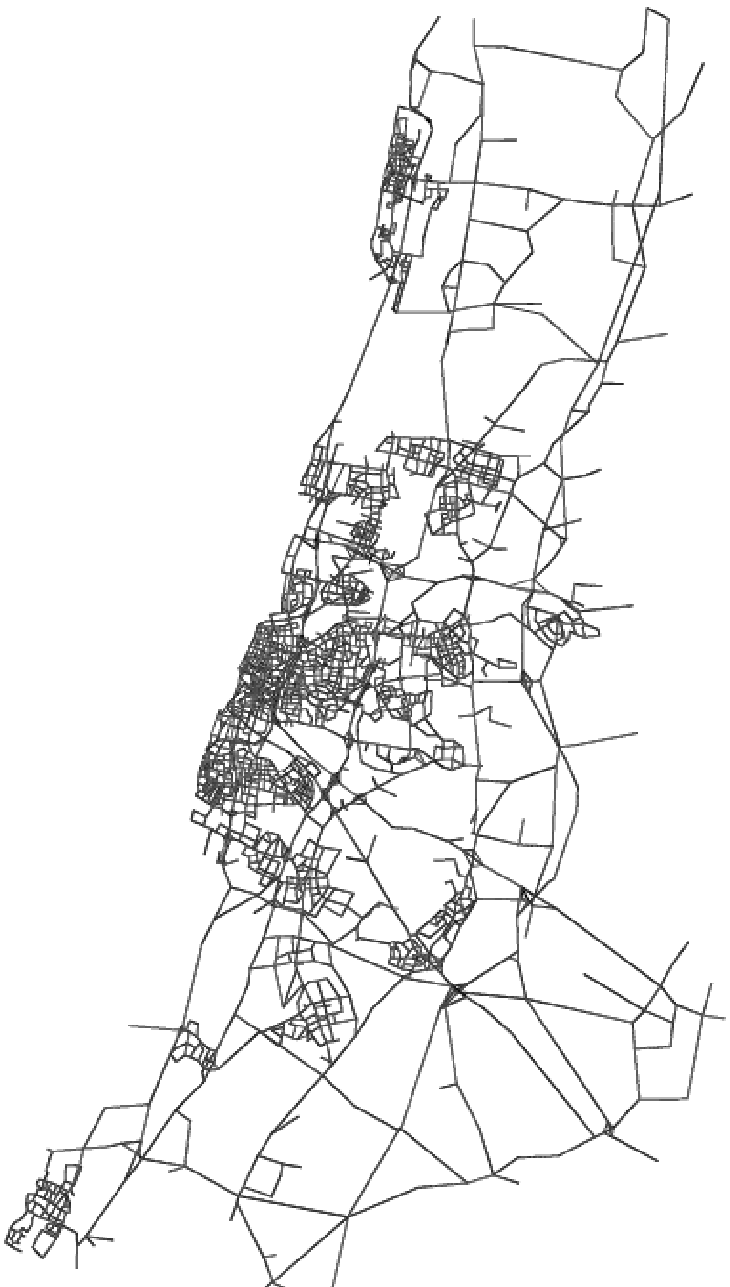
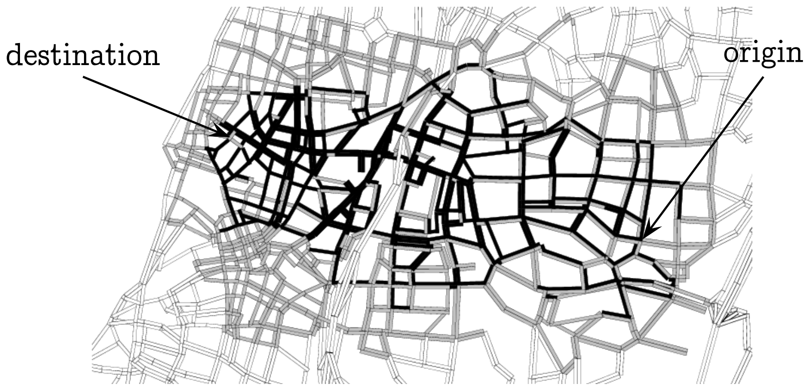
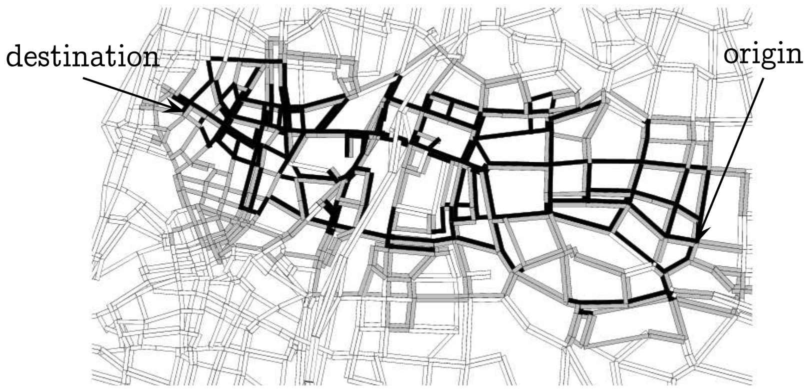


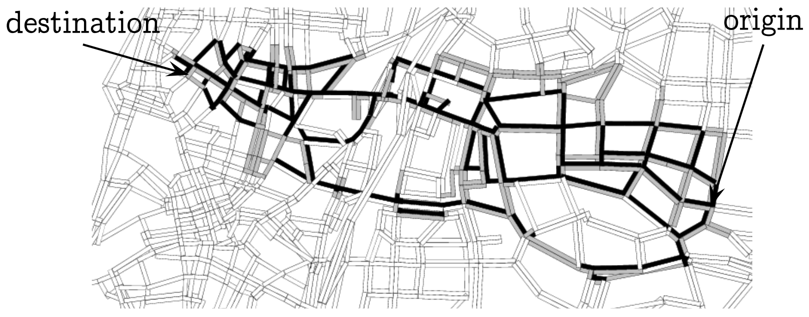
Figure 7: Tel-Aviv network



(a) $\mu = 0.01$

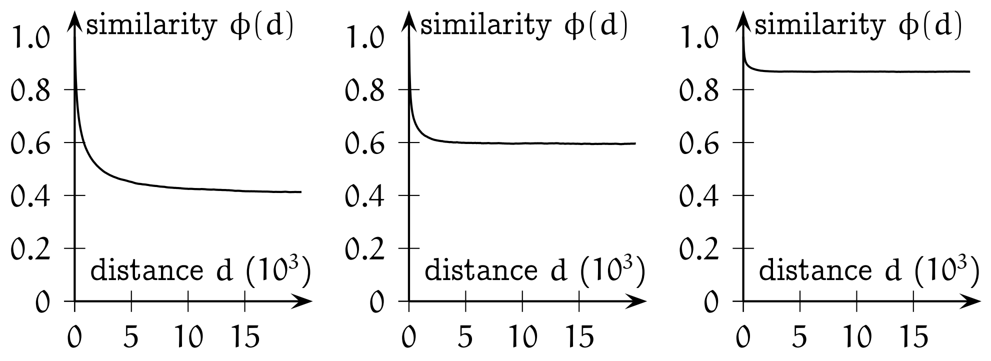


(b) $\mu = 0.02$



(c) $\mu = 0.04$

Figure 8: Link coverage in Tel-Aviv network for different μ values



(a) $\mu = 0.01$ (b) $\mu = 0.02$ (c) $\mu = 0.04$

Figure 9: Distance-dependent path similarity in Tel-Aviv example