Estimating Stochastic Shortest Paths

Michael W. Ramsey

Abstract

We consider a special case of the Stochastic Shortest Path problem in which edge costs have a known distribution and we derive the induced shortest path distribution. This case often arises from situations where edge weights result from a noisy measurement or collection process. We model this case by introducing white noise to the edge weights of a standard weighted graph. Using the resulting stochastic graph we derive computational methods for the stochastic shortest paths distribution p^*. We present a new algorithm based on these methods, called Approximate Stochastic Paths (Asp). Preliminiary tests using the NetworkX Python library [1] suggest that Asp is faster and more accurate than the common Monte Carlo simulation-based approach [2].

1 Introduction

In applied problems, graph traversal costs often result from noisy measurement or collection processes, or result from an inherently stochastic cost. For example, a traveler on her daily commute may experiences different travel costs due to varying taffic levels, weather conditions, or incident levels such as work zones, accidents, and vehicle breakdowns. In this case, a road network model with stochastic edge weights provides a more realistic representation of a road network than a deterministic one.

This paper is organized as follows. Section 2 establishes prequisite definitions and results to formally define our problem. Next, in section 3 we present two methods for solving the problem. Section 4 discusses approximation methods for non-negative weights. Section 5 presents the Asp algorithm and provides test results. Section 6 discusses miscellaneous items.

2 Problem Definition

Consider a weighted graph G defined by triple $G = (V, E, C)$ where $V = 1, 2, ..., n$ is a set of nodes, E is the edges subset of $V \times V$, and C is the $n \times n$ matrix of edge weights or costs $c_{(i,j)} \in \mathbb{R}$. We discuss the non-negative case in the next section. Denote edges as $(i, j) \in E$ for $i, j \in V$.

2.1 Stochastic Graphs

We define the stochastic weighted graph induced by G as $G^* = (V, E, C^*)$ with C^* defined as the $n \times n$ matrix of edge cost distirbutions defined by

$$Y_{(i,j)} = X_{(i,j)}c_{(i,j)} + c_{(i,j)} \tag{1}$$

where $c_{(i,j)} \in C$ and $X_{(i,j)} \sim N(0, \sigma^2)$ i.i.d. Thus we perturb the edge weights by a multiple of the original edge weight. The variance σ^2 of the white noise parameterizes the magnitude.

Note that by linearity of expected value and a basic property of variance we immediately obtain

$$Y_{(i,j)} \sim N(c_{(i,j)}, c^2_{(i,j)}\sigma^2). \tag{2}$$

2.2 Path Distributions

Next we define a path $\pi_{i,j}$ as an ordered sequence of nodes $(v_k)_{k=1}^m$ with $v_1 = i$ and $v_m = j$ as source and destination nodes respectively and $(v_k, v_{k+1}) \in E, \forall k$. A path is called simple if $v_k \neq v_l, \forall k, l$. That is, there are no loops or cycles in the path.

Let $i, j \in V$ and define $\Pi_{i,j} = \{\pi_{i,j} \mid \pi_{i,j} \text{ is simple}\}$. Then, for $\pi \in \Pi_{i,j}$ define

$$Z_\pi = \sum_{v_k \in \pi} Y_{(v_k, v_{k+1})} \tag{3}$$

$$\begin{aligned} \hat{Z}_\pi &= \sum_{v_k \in \pi} \mathbb{E}(Y_{(v_k, v_{k+1})}) \\ &= \sum_{v_k \in \pi} c_{(v_k, v_{k+1})} \end{aligned} \tag{4}$$

as the stochastic and expected costs for path π respectively. Note again, that by linearity of expectation and by a basic property of variance we get

$$\begin{aligned} \mathbb{E}(Z_\pi) &= \hat{Z}_\pi \\ &= \sum_{v_k \in \pi} \mathbb{E}(Y_{(v_k, v_{k+1})}) \\ &= \sum_{v_k \in \pi} c_{(v_k, v_{k+1})} \\ \mathrm{Var}(Z_\pi) &= \sum_{v_k \in \pi} \mathrm{Var}(Y_{(v_k, v_{k+1})}) \\ &= \sum_{v_k \in \pi} c^2_{(v_k, v_{k+1})}\sigma^2. \end{aligned}$$

Therefore,

$$Z_\pi \sim N(\sum_{v_k \in \pi} c_{(v_k, v_{k+1})}, \sum_{v_k \in \pi} c^2_{(v_k, v_{k+1})} \sigma^2). \quad (5)$$

2.3 Stochastic Shortest Path

Finally, we define the stochastic shortest path and the expected shortest path as

$$\pi^*_{i,j} = \min_{\pi \in \Pi_{i,j}} Z_\pi \quad (6)$$

$$\hat{\pi}^*_{i,j} = \min_{\pi \in \Pi_{i,j}} \hat{Z}_\pi. \quad (7)$$

Note that $\hat{\pi}^*_{i,j}$ can be computed directly from Dykstra's Algorithm (see [3]) and that $\hat{\pi}^*_{i,j}$ is the mode of $\pi^*_{i,j}$.

We can now give our problem definition:

> Given any weighted graph G and $i, j \in V$, find $p^*(\pi_{i,j}) = P(\pi_{i,j} = \pi^*_{i,j}), \forall \pi \in \Pi_{i,j}$.

This provides the probability that $\pi_{i,j}$ is the shortest path between nodes i, j for a random draw of edge weights from G^* for any path $\pi_{i,j}$.

3 Stochastic Path Distribution

We present two approaches to computing the stochastic path distribution p^*. The first solves for the distribution directly and requires numeric integration. The second estimates the distribution by sampling and aggregating the results. The best choice will depend on application details.

3.1 Analytic Solution

Given any weighted graph G, let $\pi \in \Pi_{i,j}$ for some $i, j \in V$ and $\Gamma_\pi = \Pi_{i,j} - \{\pi\}$. We can recast $P(\pi = \pi^*)$ as $P(Z_\pi < W)$ where $W = \min_{\gamma \in \Gamma_\pi} Z_\gamma$. By conditioning on $Z_\pi = s$ for some $s \in \mathbb{R}$ we obtain:

$$\begin{aligned} P(Z_\pi = s, Z_\pi < W) &= P(Z_\pi = s) P(W > s \mid Z_\pi = s) \\ &= P(Z_\pi = s) \prod_{\gamma \in \Gamma_\pi} P(Z_\gamma > s) \\ &= P(Z_\pi = s) \prod_{\gamma \in \Gamma_\pi} (1 - P(Z_\gamma \leq s)) \\ &= f_{Z_\pi}(s) \prod_{\gamma \in \Gamma_\pi} (1 - F(Z_\gamma(s))) \\ &= f_{Z_\pi}(s) \prod_{\gamma \in \Gamma_\pi} SF(Z_\gamma(s)) \quad (8) \end{aligned}$$

since all the stochastic costs are independent; and where f is the density for Z_π and F and SF are the cumulative distribution function and the survival function for Z_γ. Z_π and Z_γ are distributed according to (5).

To determine $P(Z_\pi < W)$ we simply need to integrate (8) over s:

$$\begin{aligned} P(Z_\pi < W) &= \int_{-\infty}^{\infty} P(Z_\pi = s, Z_\pi < W) ds \\ &= \int_{-\infty}^{\infty} f_{Z_\pi}(s) \prod_{\gamma \in \Gamma_\pi} SF(Z_\gamma(s)) ds \quad (9) \end{aligned}$$

Therefore, given any simple path between any two nodes in a graph with noisy edge weights, (9) gives the probability it is the shortest path.

3.2 Random Sample Solution

Alternatively, by sampling each Z_π directly using (5), we can calculate the frequency that $Z_\pi \leq \min_{\gamma \in \Gamma_\pi} Z_\gamma$ and use this to estimate $p^*(\pi_{i,j})$, $\forall \pi \in \Pi_{i,j}$.

Let $z_{\pi,k}$ be the kth random draw from Z_π out of $N \in \mathbb{N}$ many and $(z_{\pi,k})_{\pi \in \Pi_{i,j}}$ a random draw from all the Z_π. Define $m_k = \min_{\gamma \in \Gamma_\pi} z_{\gamma,k}$. Then the relative frequency that $Z_\pi \leq \min_{\gamma \in \Gamma_\pi} Z_\gamma$ is computed as:

$$\omega_N(\pi = \pi^*) = \frac{\sum_{k=1}^{N} \mathbf{1}_{\mathbb{R}^+}(m_k - z_{\pi,k})}{N} \quad (10)$$

where $\mathbf{1}_{\mathbb{R}^+} \colon X \to \{0, 1\}$ is the indicator function defined as:

$$\mathbf{1}_{\mathbb{R}^+}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x \leq 0. \end{cases}$$

Therefore, by the Law of Large Numbers, as $N \to \infty$:

$$\omega_N(\pi = \pi^*) \longrightarrow P(\pi = \pi^*) \quad (11)$$

where $P(\pi = \pi^*) = \mathbb{E}(1_{\mathbb{R}^+}(m_k - z_{\pi,k}))$.

4 Non-Negative Weights

Often, in applied settings, negative edge weights do not have a meaningful interpretation. Consider time or distance costs. In this case we apply the truncated normal distribution. However, to reproduce our results above we need to know the effect of scaling, shifting, and summing truncated normal random variables.

4.1 Truncated Normal Distribution

The Truncated Normal Distribution is defined in two steps: (1) select parameters μ and σ^2 for a parent normal random variable $\phi \sim N(\mu, \sigma^2)$ with CDF Φ and (2)

select a truncation range (a, b). The truncated normal PDF $\psi(\mu, \sigma^2, a, b; x)$ is defined as:

$$\psi(\mu, \sigma, a, b; x) = \frac{\frac{1}{\sigma}\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}$$

for $a \leq x \leq b$ and 0 elsewhere. If X has PDF $\psi(x; \mu, \sigma^2, a, b)$ we say $X \sim TN(\mu, \sigma^2)$ over (a, b).

Note that the parameters μ and σ^2 are the mean and variance of the parent normal distribution, not of the truncated normal PDF itself. These are more complicated. Let $X \sim TN(\mu, \sigma^2)$ over (a, b), $\alpha = (a - \mu)/\sigma$, and $\beta = (b - \mu)/\sigma$. Then

$$\mathbb{E}(X) = \mu - \sigma\Delta_0$$
$$\mathrm{Var}(X) = \sigma^2\left[1 - \Delta_1 - \Delta_0^2\right]$$

where $\Delta_k = \frac{\beta^k\phi(\beta) - \alpha^k\phi(\alpha)}{\Phi(\beta) - \Phi(\alpha)}$.

By integrating ϕ appropriately, the truncated normal CDF is

$$\Psi(\mu, \sigma, a, b; x) = \begin{cases} 0 & \text{if } x \leq a, \\ \frac{\Phi(\frac{x-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})} & \text{if } a < x < b, \\ 1 & \text{if } x \geq b. \end{cases}$$

4.2 Transformations

Suppose $X \sim TN(\mu, \sigma^2)$ over (a, b) and $c, d \in \mathbb{R}$. Then for $Y = cX + d$ we have

$$Y \sim TN(c\mu + d, d^2\sigma^2) \quad \text{over } (ca + d, cb + d) \tag{12}$$

In general, summing truncated normal random variables does not necessarily produce another truncated normal random variable. Fortunately though, this sum can be approximated by a truncated normal in the case where the summands are truncated symmetrically about their means (see [4]).

Let $X_k \sim TN(\mu_k, \sigma_k^2)$ over (a_k, b_k) for $k = 1, 2, \ldots n$ such that $\frac{b_k + a_k}{2} = \mu_k$. Then we can approximate the distribution of $Y = \sum_{k=1}^{n} X_k$ by

$$Y \approx TN(\sum_{k=1}^{n}\mu_k, \sum_{k=1}^{n}\sigma_k^2) \quad \text{over } (\sum_{k=1}^{n}a_k, \sum_{k=1}^{n}b_k). \tag{13}$$

4.3 Stochastic Path Distribution

We can use (12) and (13) to retrace the steps in Section 2 for the truncated normal distribution.

Define the truncated stochastic weighted graph induced by G as $G^*_{a,b} = (V, E, C^*)$ with C^* defined as the $n \times n$ matrix of edge cost distirbutions defined by

$$Y_{(i,j)} = X_{(i,j)}c_{(i,j)} + c_{(i,j)} \tag{14}$$

where $c_{(i,j)} \in C$ and $X_{(i,j)} \sim TN(0, \sigma^2)$ over $(-1, 1)$ i.i.d. Thus we perturb the edge weights by a percent of the original edge weight, but since $-1 \leq X_{(i,j)} \leq 1$, then $Y \geq 0$.

Applying (12) to (14) we immediatly obtain:

$$Y_{(i,j)} \sim TN(c_{(i,j)}, c_{(i,j)}^2\sigma^2) \quad \text{over } (0, 2c_{(i,j)}).$$

Let $\Pi_{i,j}$ be defined as in Section 2. Then, for $\pi \in \Pi_{i,j}$ define

$$Z_\pi = \sum_{v_k \in \pi} Y_{(v_k, v_{k+1})}.$$

Next we can apply (14) to Z_π to get:

$$Z_\pi \approx TN(\sum_{v_k \in \pi} c_{(v_k, v_{k+1})}, \sum_{v_k \in \pi} c^2_{(v_k, v_{k+1})}\sigma^2) \quad \text{over } (0, \sum_{v_k \in \pi} 2c_{(v_k, v_{k+1})}). \tag{15}$$

The results of Section 3 then follow analogously substituting (15) for (5).

4.4 Contribution

In addition to generating the performance improvements provided by Asp, detailed in the next section, the above results provide several benefits.

- Numerically soving the integral in (9) provides a accurate, well studied benchmark for validating code and algorithms.
- Detailing path distributions, as in (5), allows for more sophisticated modeling of path selection decision making. For example, depending on the utility function under consideration, path selection can use (5) to pick the path with the least risk rather than lowest expected cost.

5 Computing Distributions

Next, we detail a new algorithm, called Approximate Stochastic Paths (Asp), that combines the above results with a standard k-shortest paths algorithm. Preliminiary tests using the NetworkX Python library suggest that Asp is faster and more accurate than the common simulation-based approach (see [2]).

5.1 The Asp Algorithm

Asp estimates $p^*(\pi_{i,j})$ for any nodes i, j in a weighted graph G in two main steps. The first step, denoted SHORT, uses the top k shortest paths (kSP) between nodes i, j using NetworkX's version of Yen's algorithm (see [1], [5])on the deterministic edge weights $c_{(v_k,v_{k+1})}$, $v_k \in \pi$, from G. The second step, denoted FREQ, estimates $\omega_N(\pi = \pi^*)$ using equation (10). Asp automatically estimates the two required parameters, k and N, using user provided tolerances.

Yen's algorithm computes the k shortest simple paths for a graph with non-negative edge cost. The algorithm employs any shortest path algorithm to find the best path, then proceeds to find $k-1$ deviations of the best path. The NetworkX implementation uses a bi-directional Dijkstra's algorithm and returns a Python generator allowing iterative queries for the next shortest path.

Asp uses the Python generator's efficiency to retrieve shortest paths in batches of size b_k and stops when the probability, that the last path retrieved would ever be shorter than the first retrieved, is below a user provided threshold t_k. More formally, the stopping condition is $P(Z_\pi < Z_\gamma) < t_k$ where π and γ are the first and last path retrieved respectively. This works because $P(Z_\pi < Z_\gamma)$ is an upper bound on $P(Z_\pi < W)$ where $W = \min_{\gamma \in \Gamma_\pi} Z_\gamma$ and can be computed quickly as:

$$\begin{aligned} P(Z_\pi < Z_\gamma) &= P(Z_\pi - Z_\gamma) < 0 \\ &= 1 - P(Z_\gamma - Z_\pi) \leq 0 \\ &= 1 - CDF_{Z_\gamma - Z_\pi}(0) \\ &= SF_{Z_\gamma - Z_\pi}(0) \end{aligned}$$

where

$$Z_\gamma - Z_\pi \sim N(\mu_\gamma - \mu_\pi, \sigma_\gamma^2 + \sigma_\pi^2)$$

or where

$$Z_\gamma - Z_\pi \approx TN(\mu_\gamma - \mu_\pi, \sigma_\gamma^2 + \sigma_\pi^2) \text{ over } (-2\mu_k, 2\mu_k)$$

Algorithm: Asp - Truncated Normal Version
Data: $G = (V, E, C)$, i, j, σ, b_k, b_N, t_k, t_N, ϵ.
Result: $\{(\omega_N(\pi = \pi^*), (v_k, v_{k+1})) \mid v_k \in \pi \subset \Pi_{i,j}\}$.

begin
  $g = \text{kSP}(G, i, j)$
  SHORT(g, b_k, t_k**)**
    $\Pi = \emptyset$
    begin
      $p = 1$
      while $p > t_k$ **do**
        $\Pi = \Pi \cup g(b_k)$
        $\gamma = \text{First}(\Pi)$
        $\pi = \text{Last}(\Pi)$
        $\bar{\mu} = \sum_\pi c_{(v_k,v_{k+1})} - \sum_\gamma c_{(v_k,v_{k+1})}$
        $\bar{\sigma}^2 = \sum_\pi c^2_{(v_k,v_{k+1})}\sigma^2 + \sum_\gamma c^2_{(v_k,v_{k+1})}\sigma^2$
        $Z_\gamma - Z_\pi = TN(\bar{\mu}, \bar{\sigma}^2)$
        $p = SF_{Z_\gamma - Z_\pi}(0)$
    return Π
  FREQ(Π, σ, b_N, t_N, ϵ**)**
    begin
      $\Omega = \emptyset$
      for $\pi \in \Pi$ **do**
        $\bar{\mu} = \sum_\pi c_{(v_k,v_{k+1})}$
        $\bar{\sigma}^2 = \sum_\pi c^2_{(v_k,v_{k+1})}\sigma^2$
        $Z_\pi = TN(\bar{\mu}, \bar{\sigma}^2)$
      for $\pi \in \Pi$ **do**
        $N = b_N$
        $((z_{\pi,k})_{\pi\in\Pi}) = \text{DRAW}(n, (Z_\pi)_{\pi\in\Pi})$
        $\Gamma_\pi = \Pi - \{\pi\}$
        $m_k = \min_{\gamma\in\Gamma_\pi} z_{\gamma,k}.$
        $\omega_N = \frac{\sum_{k=1}^N \mathbf{1}_{\mathbb{R}^+}(m_k - z_{\pi,k})}{N}$
        $\delta_N = |\omega_N - \omega_{N-\epsilon}|$
        while $\delta_N > t_N$ **do**
          $N = N + b_N$
          $((z_{\pi,k})_{\pi\in\Pi}) = \text{DRAW}(n, (Z_\pi)_{\pi\in\Pi})$
          $\Gamma_\pi = \Pi - \{\pi\}$
          $m_k = \min_{\gamma\in\Gamma_\pi} z_{\gamma,k}.$
          $\omega_N = \frac{\sum_{k=1}^N \mathbf{1}_{\mathbb{R}^+}(m_k - z_{\pi,k})}{N}$
          $\delta_N = |\omega_N - \omega_{N-\epsilon}|$
        $\Omega = \Omega \cup (\omega_N, \pi)$
      return Ω
  return FREQ(SHORT(g, b_k, t_k), σ, b_N, t_N, ϵ)

depending on whether the weights must be non-negative or not.

To select the number N of samples to draw from Z_π, Asp again works in batchs of size b_N and stops when the marginal change δ_N in ω_N falls below a provided threshold t_N. If the marginal change is not below t_N, it draws b_N more, recomputes, tests, and repeats until the threshold is met. Asp computes the marginal change as $\delta_N = |\omega_n - \omega_{n-\epsilon}|$ for some user provided ϵ and for n equal to b_N times the number of completed iterations.

5.2 Simulating p^*

The popularity of estimating p^* via simulation comes from its relative simplicity: for some number of iterations $i = 1 \dots n$, the algorithm computes f_N by running Dykstra's algorithm on G_i^*, where G_i^* is newly generated from G in each iteration.

5.3 Experimental Results

As a preliminary test of Asp, we compare its performace to Monte Carlo simulation on a series of small synthetic graphs where we control the number and length of paths between two points. Each test graph consists of a set number m of disjoint paths with random length and weights drawn uniformly from $(10, 15)$ and $(0, 10)$ respectively. For this study, we used $m = 2 \dots 10$.

To evaluate Asp's accuracy in the absence of groundtruth, we exploit the fact that the simulation estimate of p^* should improve as the number of iterations increases. If Asp is more accurate than the simulation estimates, then as we increase the number of iterations, the simulation estimate should approach Asp's estimate. Therefore, this experiment records the difference between Asp's estimates and the simulated estimates, along with their respective computation times.

In this experiment we performed three trials and averaged the results for each test graph. Each trial consisted of running Asp and the simulation algorithm on each test graph. We ran the simuation for each of 10^2, 10^3, and 10^4 iterations. For each trial we recorded the average difference between Asp's estimates and the simulated estimates, along with their respective computation times. We also set $\sigma = .2$ because simulation took too many iterations and too much time to achieve reasonable probability estimates.

The experimental results, shown in Figure 1 and Figure

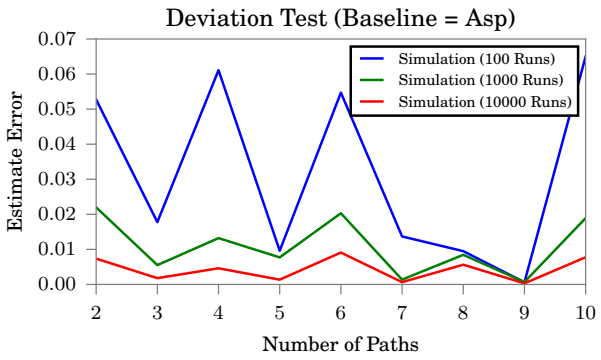

Figure 1: Deviation of simulation estimates from Asp's estimate for each of 10^2, 10^3, and 10^4 iterations.

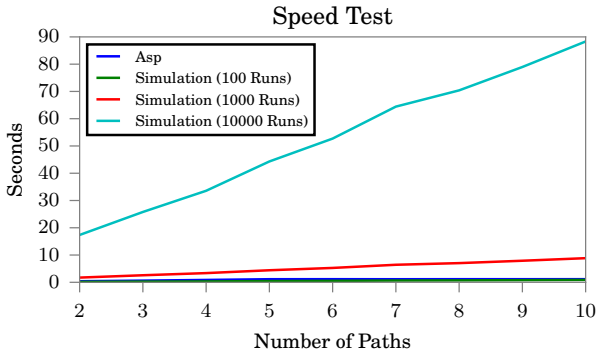

Figure 2: Computing times of Asp and simulation estimates for each of 10^2, 10^3, and 10^4 iterations.

2, indicate that Asp is both faster and it's estimate is more accurate than the simulations. As the number of simulation runs increases, the simulation estimate deviates proportionatly less from the Asp estimate. Given that the similation estimate should be getting better as the iteration runs increase, we conclude Asp is more accurate. Spot checking the results and one-off comparisons to numerical integration estimates based on (9) also supports this conclusion. Further, the computation times of Asp remains constant as the number of paths increases, whereas simulation times increase significantly.

6 Discussion

6.1 Caveats

It is important to stress the limitations of this experiement and note that futher research is required. Experimental results could vary depending on σ, Asp paramenter choices, software implementations, or

graph properties. We did run one variation of the study using kSP as input into the simulation to help prune the computation, but found little performance improvement.

6.2 Edge Traversal Probability

Extending estimates for p^* to edge traversal probabilities arises naturally in the simulation algorithm since edge traversal counts can be easily obtained from path traversals. Such accounting does not come easily with Asp. However, the probability of an edge being traversed can be computed from the probabilities of the paths containing it.

For example, suppose there is a distribution over the nodes giving the probability $P((i,j) = (s,e))$ each pair of nodes would be selected as endpoints (s,e). Then, given any edge (u,v), the probability (u,v) is traversed could be defined as:

$$P((u,v) \text{ traversed }) = \sum_{(u,v)\in\pi_{i,j}} p^*_{\pi_{i,j}} P((i,j) = (s,e)))$$

6.3 Other Distributions

Reproducing our resutls only requires knowing the effect of scaling, shifting, and summing a random variable for any distribution. Therefore, both our derivations and Asp should be extendable to a wide range of edge cost noise models.

6.4 Other Decision Models

This paper focuses on the the global shortest path using a single decision criteria - edge weights. There are many situations where path selection is based on local information, only the next path, or multiple criteria, such as total edge cost and the number to nodes traversed. Possible extentions include investigating multiobjective models and short-sighted decision making (see [6], [7]).

7 Conclusion

We considered a special case of the Stochastic Shortest Path problem in which edge costs have a known distribution and we derive the induced shortest path distribution. We induced a stochastic weighted graph from a standard weighted graph by introducing white noise to the edge weights. Using this graph we derived computational methods for the stochastic shortest paths distribution p^* and presented a new algorithm based on these methods, called Approximate Stochastic Paths (Asp).

Our experimental results indicate that Asp is both faster and it's estimate is more accurate than the more common Monte Carlo simulation approach. As the number of simulation runs increases, the simulation estimate deviates proportionatly less from the Asp estimate. Given that the similation estimate should be getting better as the iteration runs increase, we conclude Asp is more accurate. Spot checking the results and one-off comparisons to numerical integration estimates based on (9) also supports this conclusion. Further, the computation times of Asp remains constant as the number of paths increases, whereas simulation times increase significantly.

In addition to generating these performance improvements, our results provide several benefits. First, Numerically soving the integral in (9) provides a accurate, well studied benchmark for validating code and algorithms. Second, detailing path distributions, as in (5), allows for more sophisticated modeling of path selection decision making. For example, depending on the utility function under consideration, path selection can use (5) to pick the path with the least risk rather than lowest expected cost.

References

[1] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, (Pasadena, CA USA), pp. 11–15, Aug. 2008.

[2] H. Frank, "Shortest paths in probabilistic graphs," *Operations Research*, vol. 17, p. 583599, Jul 1969.

[3] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, dec 1959.

[4] A. M. Andrew N. O'Connor, Mohammad Modarres, *Probability Distributions Used in Reliability Engineering*. Center for Risk and reliability, University of Maryland, 2016.

[5] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, pp. 712–716, jul 1971.

[6] M. Rajabi-Bahaabadi, A. Shariat-Mohaymany, M. Babaei, and C. W. Ahn, "Multi-objective path finding in stochastic time-dependent road networks using non-dominated sorting genetic algorithm," *Expert Systems with Applications*, vol. 42, pp. 5056–5064, jul 2015.

[7] R. W. Hall, "The fastest path through a network with random time-dependent travel times," *Transportation Science*, vol. 20, no. 3, p. 182188, 1986.