Finding the most reliable path in a stochastic network

Geert Doek
Student ID: 410547

July 6, 2018

Supervisor: Rolf van Lieshout
Co-reader: Prof. dr. Dennis Huisman

Bachelor Thesis Econometrics and Operations Research
Erasmus School of Economics
Erasmus University Rotterdam

Abstract

This thesis studies the problem of finding the most reliable path between two nodes on a graph, when the exact travel times are unknown. It is desirable that a path is quick on average, while at the same time the arrival time should be easy to predict. Therefore the objective is to minimize a weighted sum of mean and standard deviation. Since this objective function is non-linear and non-convex, standard methods cannot be used. We analyze and implement the algorithm proposed by Xing and Zhou (2011) to find close-to-optimal paths. The algorithm is based on a Lagrangian substitution approach, which results in a sequence of shortest path problems. The Lagrangian multipliers are updated using subgradient optimization. The analysis of Xing and Zhou (2011) is extended in two ways. First, instead of minimizing a weighted sum of mean and standard deviation, we analyze the objective of maximizing on-time arrival probability, in case of a normal distribution. Second, the algorithm is extended to allow for rerouting decisions along the way. For both extensions, examples are presented that illustrate the theory and the algorithms. All methods are tested on a number of randomly generated graphs.

Contents

1 Introduction

The shortest path problem is a well-known problem in operations research. Among other things, this problem is relevant in transportation, when a driver wants to find the quickest route to his or her destination. Furthermore, the shortest path problem is often encountered as a subproblem when solving more complex problems. In the standard shortest path problem the travel times between locations are known beforehand, and this problem is well-understood. In reality however, one does not know the actual travel times when deciding on a route. Travel times depend on many uncertain factors, e.g. demand, weather, and traffic incidents. Therefore, it is more realistic to model the arc weights using a probability distribution. The most desirable route in a stochastic network is not necessarily the one that is shortest on average. Since most people are risk averse, they also want a route to be reliable, i.e. the actual travel time can be predicted accurately. In real life, people often choose a route that is short on average and leave a bit earlier, to account for things like congestion. However, this extra time is costly, so there might be better routes.

Different measures of reliability of routes in a network have been studied. One approach is to maximize the probability of arriving before a pre-determined time (e.g. Fan, Kalaba, & Moore, 2005; Nie & Wu, 2009). Another approach is to find the path with the lowest $(100 - \alpha)\%$ quantile of arrival times, for a given α. These two approaches are shown to be equivalent by Nie and Wu (2009). Sivakumar and Batta (1994) include a constraint that bounds the variance of paths.

Instead of using the probability densities explicitly, we only focus on mean and standard deviation, since the standard deviation is an obvious measure of variation. We want to minimize a weighted sum of the mean and standard deviation, however this objective function is non-linear. Therefore, standard shortest path algorithms cannot be used. Xing and Zhou (2011) develop a heuristic to solve this problem and this thesis follows their approach. The main idea is to introduce a new variable and dualize the defining constraint of this new variable in the objective function. We can then split the problem into two sets of variables, which gives us two easy problems to solve. The resulting lower bound on the optimal objective value is maximized using a subgradient algorithm.

The problem is extended in two ways. First, we develop a method for finding the path that maximizes on-time arrival probability for some given arrival time, under the assumption that travel times are normally distributed. By exploiting the fact that a normal distribution is completely characterized by its mean and variance, the objective can be written in a simple functional form. It turns out that we can make use of the heuristic that finds the path that minimizes a weighted sum of mean and standard deviation, to find the path that maximizes on-time arrival probability. Second, we develop an algorithm that allows for route changes along the way. If link travel times are correlated, actual travel times on visited arcs give additional information on travel times on unvisited arcs. Therefore, it may be beneficial to take a different route than originally planned. To make computations easy, we again rely on the assumption that travel times are normally distributed.

The remainder of this proposal is structured in the following way. In Section 2, a mathematical formulation of the problem is given. In Section 3, we derive and present the algorithm proposed by Xing and Zhou (2011). In Section 4, we test the algorithm and evaluate its performance. Sections 5 and 6 are devoted to extensions of the most reliable path problem.

2 Mathematical formulation

In this section, we give a mathematical formulation of the problem. First, we have to introduce some notation. Let $G = (N, A)$ be a directed graph, where N is the set of nodes and A is the set of arcs. We denote the arc from i to j by (i, j), for $i, j \in N$. Let C_{ij} be the random variable that represents the travel time of arc $(i, j) \in A$. We assume that C_{ij} has mean $\bar{c}_{ij} \geq 0$ and variance σ_{ij}^2.

We also assume that the travel times C_{ij} are mutually independent. This is a strong assumption, but we will elaborate on this assumption in Section 6. Furthermore, let $o \in N$ denote the origin and $d \in N$ denote the destination. For all $(i,j) \in A$, let x_{ij} be a binary variable that equals 1 if (i,j) is used in a solution and 0 otherwise. In order to have a feasible path in G, we impose the following constraint on the variables x_{ij}:

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ij} = \begin{cases} 1, \text{ if } i = o \\ 0, \text{ if } i \in N \setminus \{o,d\} \\ -1, \text{ if } i = d \end{cases}, \qquad \forall i \in N. \tag{1}$$

These constraints are called the flow balance constraints. It is easy to see that a solution $(x_{ij})_{(i,j)\in A}$ that satisfies the flow balance constraints constitutes a path, by starting at the origin and iteratively determining the next node. Since the number of incoming arcs equals the number of outgoing arcs at all intermediate nodes, it is always possible to choose the next node, until one arrives at the destination node. If the goal is to minimize the average travel time, the objective would be given by

$$\min \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij}, \tag{2}$$

which would turn our problem into an ordinary shortest path problem. However, we want to find a solution such that the travel time is short on average and, at the same time, does not vary too much. Since the standard deviation is an obvious measure of variability, our objective function will be a weighted sum of the mean and the standard deviation. Because the travel times are mutually independent, the variance of a path can be computed as

$$\sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij}.$$

Let β be a positive constant, which we will call the reliability coefficient. The objective is given by

$$\min \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij} + \beta \sqrt{\sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij}}. \tag{3}$$

Note that β determines the importance of reliable travel time relative to average travel time. To conclude, the full mathematical formulation of the most reliable path problem $(P1)$ is as follows:

$$(P1) \qquad \min \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij} + \beta \sqrt{\sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij}}. \tag{4}$$

s.t.

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ij} = \begin{cases} 1, \text{ if } i = o \\ 0, \text{ if } i \in N \setminus \{o,d\} \\ -1, \text{ if } i = d \end{cases}, \qquad \forall i \in N \tag{5}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A. \tag{6}$$

In the remainder of this paper, a solution of $(P1)$ is denoted by (x_{ij}). Furthermore, if we talk about a path, we always refer to a feasible solution (x_{ij}).

3 Algorithm

The objective function of $(P1)$ is not linear in the variables x_{ij}. Moreover, the square root makes the objective function concave. Therefore, we cannot use standard linear or convex integer programming solution techniques to solve this problem. Instead, we will exploit the special structure of the problem to derive an algorithm that approximates the solution efficiently.

First of all, we substitute $y = \sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij}$, i.e. y is the variance of the travel time of the solution (x_{ij}). Then we can reformulate $(P1)$ as

$$(P1) \qquad \min \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij} + \beta\sqrt{y} \tag{7}$$

subject to

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ij} = \begin{cases} 1, \text{ if } i = o \\ 0, \text{ if } i \in N \setminus \{o,d\} \\ -1, \text{ if } i = d \end{cases}, \qquad \forall i \in N \tag{8}$$

$$y \geq \sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij}, \tag{9}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A. \tag{10}$$

The inequality sign in (9) is justified, because in an optimal solution, this constraint will always be tight. Let y' be the variance of an optimal solution (x'_{ij}) of the average shortest path problem, as specified by (2) and (1). We claim that in an optimal solution of $(P1)$, we will have $y \leq y'$. To see this, suppose $y > y'$. Then (x'_{ij}) would be a strictly better solution to $(P1)$ than (x_{ij}), since both terms of the objective (7) strictly decrease. Hence, (x_{ij}) cannot be an optimal solution in $(P1)$. For this reason, we can add the constraint

$$0 \leq y \leq y' \tag{11}$$

to $(P1)$, without changing the problem.

The next step is to relax constraint (9) in a Lagrangian fashion, so we remove the constraint and change our objective to

$$\begin{aligned} \mathcal{L}(\mu) &= \min \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij} + \beta\sqrt{y} + \mu \left(\sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij} - y \right) \\ &= \min \sum_{(i,j)\in A} (\bar{c}_{ij} + \mu\sigma_{ij}^2) x_{ij} + \beta\sqrt{y} - \mu y, \end{aligned} \tag{12}$$

subject to constraints (8), (10) and (11). Here μ is a non-negative Lagrangian multiplier, which determines the penalty for violating (9). Observe that the remaining constraints (8), (10) and (11) do not specify any dependence between (x_{ij}) and y. As a result, we can optimize with respect to (x_{ij}) and y separately. Formally, we have

$$\mathcal{L}(\mu) = \mathcal{L}_x(\mu) + \mathcal{L}_y(\mu), \tag{13}$$

where

$$\mathcal{L}_x(\mu) = \min \left\{ \sum_{(i,j)\in A} (\bar{c}_{ij} + \mu\sigma_{ij}^2) x_{ij} : (x_{ij}) \text{ satisfies (8) and (10)} \right\} \tag{14}$$

and

$$\mathcal{L}_y(\mu) = \min_{0 \le y \le y'} \{\beta\sqrt{y} - \mu y\}. \tag{15}$$

For given μ, $\mathcal{L}_x(\mu)$ specifies a shortest path problem with non-negative arc weights $\bar{c}_{ij} + \mu\sigma_{ij}^2$, which can be solved efficiently by a number of algorithms, like Dijkstra's algorithm (Dijkstra, 1959). The mapping $y \mapsto \beta\sqrt{y} - \mu y$ is a univariate concave function, therefore the minimum can be found at the boundaries of the interval $[0, y']$. It follows that $\mathcal{L}_y(\mu) = \min\{0, \beta\sqrt{y'} - \mu y'\}$. We have now found an efficient way to calculate $\mathcal{L}(\mu)$. Note that $\mathcal{L}(\mu)$ is a lower bound to the optimal objective value of $(P1)$, because in the Lagrangian problem, the feasible region is enlarged and the objective value does not change for feasible solutions in $(P1)$. On the other hand, an upper bound on the optimal objective value of $(P1)$ can be given by the objective value of any feasible solution of $(P1)$. In particular, for given μ we can find an upper bound UB_μ by calculating the objective value (7) corresponding to the solution (x_{ij}) found in $\mathcal{L}_x(\mu)$.

To get as much information as possible on the optimal solution, we want to maximize the lower bound of $(P1)$, i.e. we solve the Lagrangian dual problem. A subgradient algorithm is used to find the optimal μ. Given $\mu^{(k)}$, we determine $\mathcal{L}(\mu^{(k)})$ and the corresponding solution $(x_{ij}^{(k)})$. If possible, we also update the best (lowest) upper bound UB. For the next iteration, we set $\mu^{(k+1)} = \mu^{(k)} + \alpha^{(k)} g^{(k)}$. The search direction $g^{(k)}$ is given by

$$g^{(k)} = \sum_{(i,j) \in A} \sigma_{ij}^2 x_{ij}^{(k)} - y^{(k)}, \tag{16}$$

where $(x_{ij})^{(k)}$ and $y^{(k)}$ are optimal solutions in $\mathcal{L}(\mu^{(k)})$. Note that $g^{(k)}$ is the subgradient of $\mathcal{L}(\mu^{(k)})$ with respect to $\mu^{(k)}$. The step size $\alpha^{(k)}$ is given by

$$\alpha^{(k)} = \frac{\lambda^{(k)}(UB - \mathcal{L}(\mu^{(k)}))}{|g^{(k)}|^2}, \tag{17}$$

where $\lambda^{(k)}$ is a suitably chosen scalar.

Given a tolerance parameter $\varepsilon > 0$, the algorithm terminates whenever the relative gap $(UB - \mathcal{L}(\mu^{(k)}))/UB$ is smaller than ε. Additionally, we may also set a maximum number of iterations K, such that the algorithm terminates as soon as $k > K$. When the algorithm terminates, we have found a feasible solution, namely the solution that generated the best upper bound, and we have a bound on the optimality gap. The full algorithm is summarized as follows:

Algorithm 1
Initialize $\mu^{(1)}$, K and ε and set $UB = +\infty$ and $(x_{ij}^*) = 0$
Calculate y' by solving the average shortest path problem
for $k = 1$ to K **do**
  Solve $\mathcal{L}_x(\mu^{(k)})$ using Dijkstra's algorithm and store shortest path $(x_{ij}^{(k)})$
  Set $UB^{(k)} = \sum_{(i,j) \in A} \bar{c}_{ij} x_{ij}^{(k)} + \beta\sqrt{\sum_{(i,j) \in A} \sigma_{ij}^2 x_{ij}^{(k)}}$
  Determine $\mathcal{L}_y(\mu^{(k)}) = \min\{0, \beta\sqrt{y'} - \mu y'\}$
  Set $\mathcal{L}(\mu^{(k)}) = \mathcal{L}_x(\mu^{(k)}) + \mathcal{L}_y(\mu^{(k)})$
  if $UB^{(k)} < UB$ **then**
    Set $UB = UB^{(k)}$ and $(x_{ij}^*) = (x_{ij}^{(k)})$
  end if
  if $(UB - \mathcal{L}(\mu^{(k)}))/UB < \varepsilon$ **then**

break
end if
Set $\mu^{(k+1)} = \mu^{(k)} + \alpha^{(k)} g^{(k)}$
end for
return (x_{ij}^*)

3.1 Example

We provide a small example to illustrate how the algorithm works in detail. Let $G = (N, A)$, where $N = \{1, 2, 3, 4\}$ and $A = \{(1, 3), (1, 4), (3, 2), (4, 2), (3, 4), (4, 3)\}$. We want to find the most reliable path from 1 to 2. The mean and variance of each arc can be found in Figure 1.

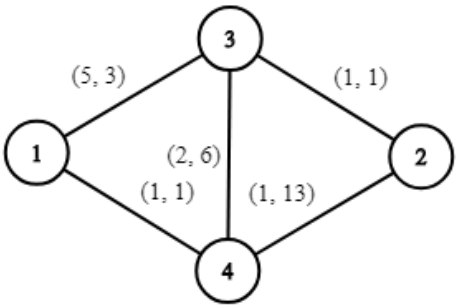

Figure 1: Graph G with mean and variance parameters on the arcs

Since this example is small, we can easily do a path enumeration to find the most reliable path. There are four paths from 1 to 2 without cycles in G, namely $1 \to 3 \to 2$, $1 \to 4 \to 2$, $1 \to 3 \to 4 \to 2$ and $1 \to 4 \to 3 \to 2$. For $\beta = 1$, the objective values equal $8, 5.74, 12.69$ and 6.83, respectively, therefore $1 \to 4 \to 2$ is the most reliable path. In each iteration of Algorithm 1, the solution that is considered depends only on the value of μ. More precisely, a shortest path problem is solved where the arc weights equal $\bar{c}_{ij} + \mu\sigma_{ij}^2$. Therefore, for each μ, we can compute a lower bound and an upper bound on the optimal objective value.

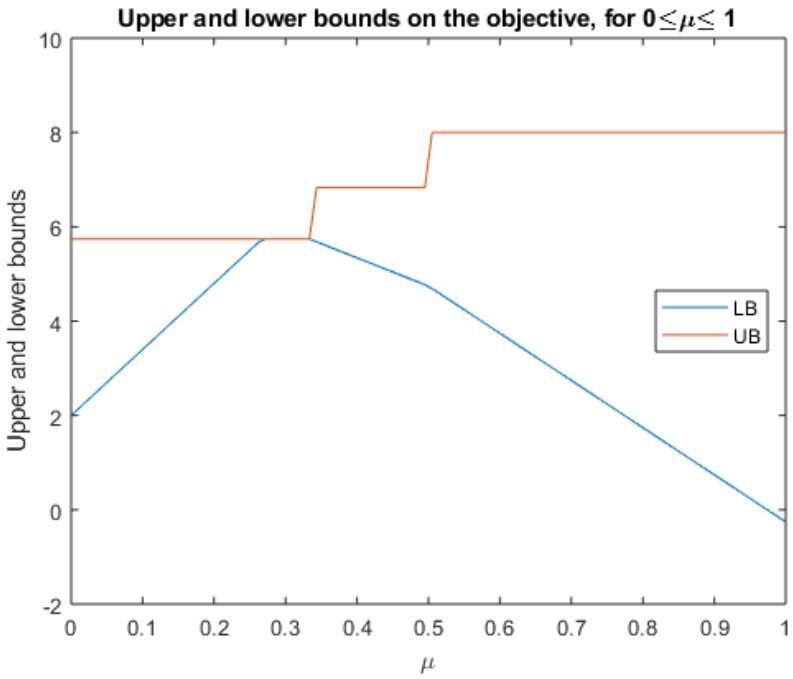

Figure 2: Upper and lower bounds on the optimal objective of the example

Observe from Figure 2 that for all $\mu \leq 1/3$, the upper bound equals the optimal objective value, therefore the shortest path found in the calculation of $\mathcal{L}_x(\mu)$ is in fact the most reliable path. Note that $\mu = 1/3$ is the point of intersection of the lines corresponding to the objective values of the paths $1 \to 4 \to 2$ and $1 \to 4 \to 3 \to 2$. If $\mu \in [\sqrt{14}/14, 1/3]$, the lower bound equals the upper bound, so we know for sure that the solution is optimal. Note that, in general, it is not necessary that the lower bound equals the upper bound for a solution to be optimal. The value of $\sqrt{14}/14 \approx 0.27$ comes about since $y' = 14$ is the variance of the shortest path, and y' is a solution of $\beta\sqrt{y'} - \mu y' = 0$. Therefore, $\mathcal{L}_y(\mu)$ starts decreasing when μ gets below $\sqrt{14}/14$. The algorithm terminates after 3 iterations, as the subgradient procedure gives 1, 0.17, 0.27 as subsequent Lagrangian multipliers, hence the optimality gap vanishes.

4 Numerical experiments

4.1 Experimental setup

In this section, we will apply Algorithm 1 to find the most reliable path in a number of networks. As test data, we have randomly generated graphs of different sizes (10, 100, 1,000 and 10,000 nodes) with, for each arc, the mean and variance of the travel time. For each of the four sizes, there are 10 instances with relatively low variance and 10 instances with higher variance. In order to run the algorithm, we need to decide on values for a number of parameters. We set $\varepsilon = 10^{-1}$ and run $K = 50$ iterations. To ensure good performance of the subgradient algorithm, suitable values for $\mu^{(1)}$ and $\lambda^{(k)}$ need to be chosen. After experimenting with different values, it turned out that the Lagrangian multiplier each time converged to the same value, regardless of what values we choose

for $\mu^{(1)}$ and $\lambda^{(k)}$. We decide to set $\mu^{(1)} = 1$ and $\lambda^{(k)} = 2^{-10k/K}$, to make sure the step size does not diminish too quickly. Held , Wolfe, and Crowder (1974) state that the step size in (17) with this choice for $\lambda^{(k)}$ guarantees convergence of the algorithm. By studying traveler's preferences, Noland, Small, Koskenoja, and Chu (1998) propose a value of 1.27 for the reliability coefficient β. We also test the algorithm for $\beta = 4$, to see how this algorithm performs for highly risk-averse travelers. The algorithm was implemented in Java on the Windows 10 platform and was run on a personal computer with Intel Core i7 2.7 GHz CPU and 8GB RAM.

The the most important measure of the performance is the relative optimality gap $r = (UB - LB)/UB$, where LB and UB are the best found lower and upper bound. If the optimality gap equals zero, we have found the optimal solution. If the optimality gap is strictly positive, this does not necessarily mean that we have not found the optimal solution. It might be the case that the upper bound corresponds to an optimal solution, but the lower bound is not tight.

4.2 Results

Figure 3 shows how how the average relative gap evolves for $k = 1, \ldots, K$. We only plot the gaps for $k = 1, \ldots, 20$, since the optimality gap does not change significantly after that point. Observe that the relative optimality gap converges very quickly. Considering all instances together, the average gap tends to 2.1% for $\beta = 1.27$ and to 7.3% for $\beta = 4$. Although the test data is different, these values are comparable to the values found by Xing and Zhou (2011). Presumably, the larger gap for $\beta = 4$ is due to the additional weight put upon the non-linear factor, therefore reducing the accuracy of the solution method. We also plot the standard deviations of the relative gaps in Figure 4, as a measure of the variability between random instances. For $\beta = 1.27$, we find the optimal solution for at least 19 of the 80 instances, and the highest observed relative gap after K iterations is 12%. For $\beta = 4$, we are sure that we find the optimal solution for 6 instances, but the highest relative gap is almost 25%.

Table 1: Average relative gaps for different types

	β=1.27		$\beta = 4$	
	low variance	high variance	low variance	high variance
10 nodes	1.9%	2.9%	5.8%	8.2%
100 nodes	2.5%	3.7%	8.3%	11.1%
1000 nodes	1.0%	2.0%	5.6%	7.8%
10000 nodes	0.9%	4.8%	2.0%	6.5%

From Table 1, we can see that the algorithm seems to be more accurate for large instances. It is possible that, in the larger graphs, there are many reliable paths with approximately the same variance and the algorithm is effective in finding at least one of these paths. The average running time per instance, for each type of graph, is provided in Table 2. The running time increases substantially if we move from 1,000 to 10,000 nodes. It is known that the computation time of a shortest path calculation is $\mathcal{O}(|N| \cdot |A|)$. Since $|A|$ is $\mathcal{O}(|N|^2)$, if we go from 1,000 to 10,000 nodes, the number of computations can increase by a factor of 1,000. This explains the dramatic increase in running times.

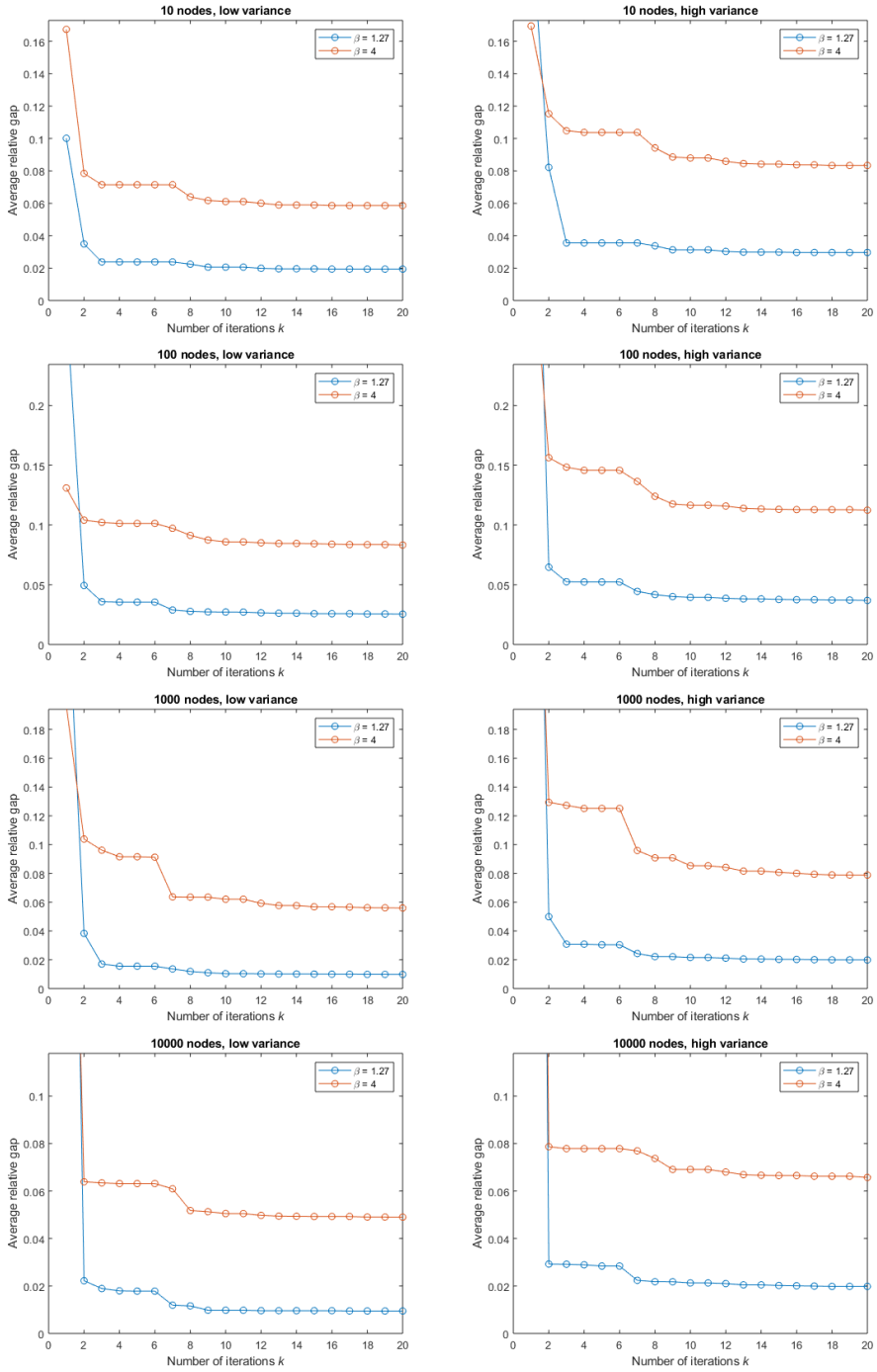

Figure 3: Average relative optimality gap, for a given number of iterations

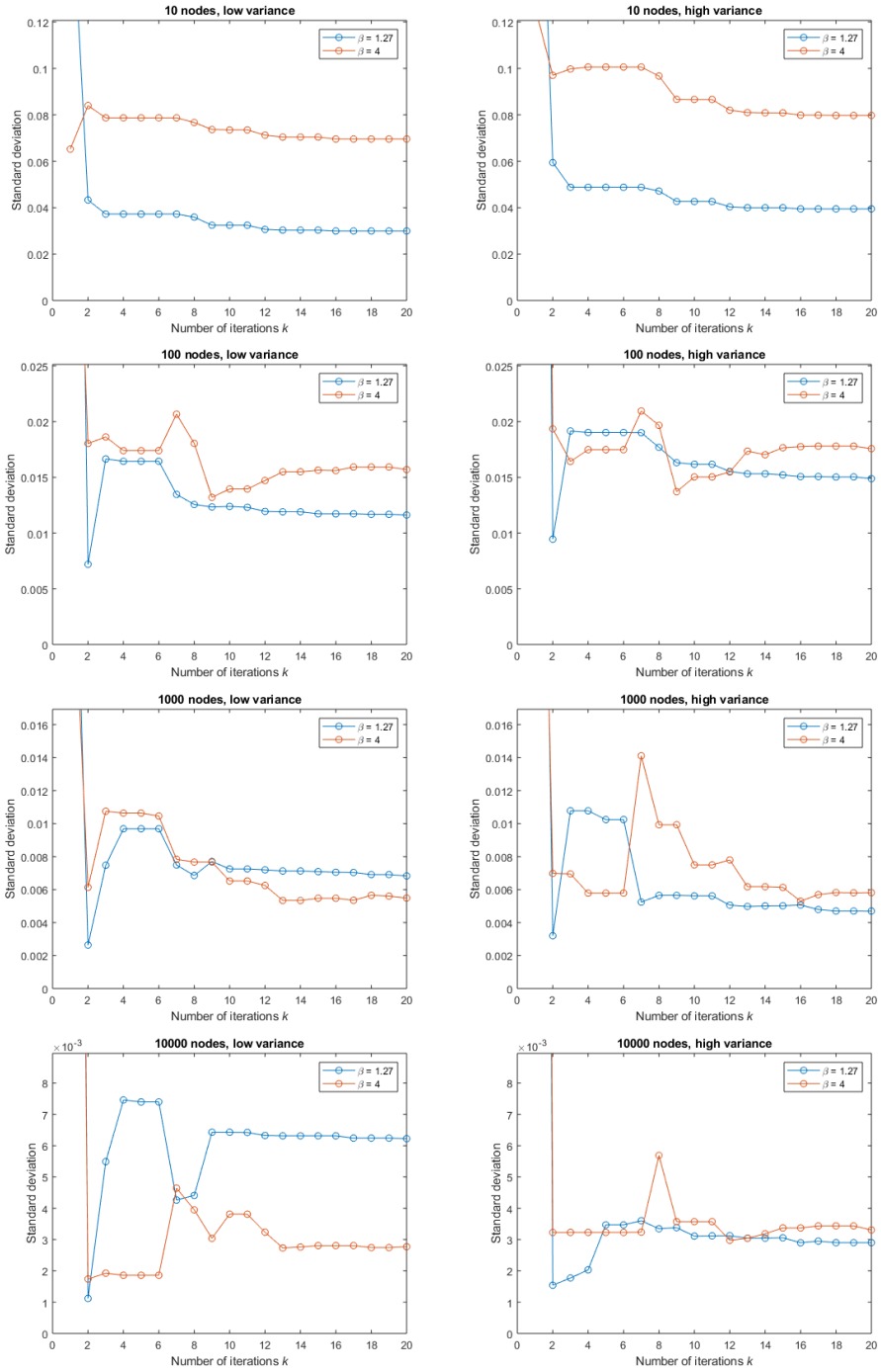

Figure 4: Standard deviation of the relative optimality gap, for a given number of iterations

Table 2: Average running times in seconds, for at most $K = 50$ iterations

	β=1.27		$\beta = 4$	
	low variance	high variance	low variance	high variance
10 nodes	0.012	0.004	0.012	0.004
100 nodes	0.032	0.021	0.034	0.029
1,000 nodes	0.096	0.108	0.120	0.102
10,000 nodes	2.914	3.606	3.629	3.704

Solving a shortest path problem with arc weights equal to the link travel time mean can be used as a heuristic for finding the most reliable path. Since Algorithm 1 required K shortest path calculations, we are interested in how effective a shortest path calculation is in finding the most reliable path. We evaluate this by comparing the reliable path objective S of the solution of the shortest path problem and with the objective R found using Algorithm 1. Obviously, we must have $S \geq R$. The effectiveness is measured by $(S-R)/R$ and the average values of these gaps are provided in Table 3.

Table 3: Average relative optimality gaps of the shortest path heuristic

	β=1.27		$\beta = 4$	
	low variance	high variance	low variance	high variance
10 nodes	1.4%	2.6%	8.2%	11.2%
100 nodes	2.2%	3.8%	11.6%	17.6%
1,000 nodes	0.6%	1.1%	4.2%	7.0%
10,000 nodes	0.5%	1.1%	5.5%	9.7%

It may seem that the shortest path heuristic performs reasonably well for finding the most reliable path on large graphs. However, the relative optimality gap might not be a good measure of the sub-optimality. There are real-life situations where the absolute optimality gap $S - R$ is much more important. Because the paths on the larger graphs are much longer, the absolute optimality gap is still very large.

5 Maximizing on-time arrival probability

For now, we have used a weighted sum of mean and standard deviation as objective function. In general, low mean and low standard deviation are desirable. However, a more direct approach is to maximize the probability of arriving before a specified arrival time, which is what people actually want. This means that we want to find the path such that the cumulative distribution function evaluated at the arrival time is maximal among all paths. In general, this objective depends on more than just mean and standard deviation, since mean and standard deviation do not completely specify the distribution and the functional form of the objective may be different. We will explore the case of the normal distribution, since, as we shall soon see, this case is related to the objective we explored earlier. Fan et al. (2005) already considered a the maximum on-time arrival probability problem for general probability distributions, but this framework leads to a system of integral equations, which might be hard to solve. Srinivasan, Prakash, and Seshadri (2014) assume shifted log-normal travel times, which gives a slightly easier problem to solve. The framework considered in this section contributes to the existing literature in the sense that the relation between the maximum on-time arrival probability problem and the shortest mean-standard deviation path problem is explored.

Let $G = (N, A)$ be a graph and assume that the link travel times are normally and independently distributed with mean $\bar{c}_{ij}$ and variance σ_{ij}^2. In reality, this is not the case, since travel times are positive, however the normal distribution might be a good approximation if the variance is relatively low. Furthermore, the assumption of normally distributed travel times has been made in several research papers (e.g. Chen et al., 2012; Sen, Pillai, Joshi, & Rathi, 2001). The travel time C_p of a path p corresponding to a solution (x_{ij}) is normally distributed with parameters equal to

$$\bar{c}_p = \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij}$$

and

$$\sigma_p^2 = \sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij}.$$

Let t be the desired arrival time at the destination $d \in V$. Then the probability of arriving before time t, given that we take path p, equals

$$P(C_p \leq t) = P\left(\frac{C_p - \bar{c}_p}{\sqrt{\sigma_p^2}} \leq \frac{t - \bar{c}_p}{\sqrt{\sigma_p^2}}\right) = P\left(Z \leq \frac{t - \bar{c}_p}{\sqrt{\sigma_p^2}}\right) = \Phi\left(\frac{t - \bar{c}_p}{\sqrt{\sigma_p^2}}\right),$$

where Φ is the standard normal cumulative distribution function. Since the function Φ is monotonically increasing, maximizing $P(C_p \leq t)$ over all paths is equivalent to maximizing $(t - \bar{c}_p)/\sqrt{\sigma_p^2}$. It is still the case that decreasing the mean or standard deviation of a path gives a better objective, but the objective is not a weighted sum anymore. We formulate the problem mathematically:

$$(P2) \qquad \max \frac{t - \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij}}{\sqrt{\sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij}}} \tag{18}$$

s.t.

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ij} = \begin{cases} 1, \text{ if } i = o \\ 0, \text{ if } i \in N \setminus \{o, d\} \\ -1, \text{ if } i = d \end{cases}, \qquad \forall i \in N \tag{19}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in A. \tag{20}$$

The optimal solutions of $(P1)$ and $(P2)$ are both Pareto optimal if we consider a bi-objective shortest path problem with objectives $\bar{c}_p$ and σ_p^2. This means that in an optimal solution of $(P1)$ or $(P2)$, we cannot decrease the mean without increasing the standard deviation, and vice versa. This observation will help us to solve the problem. First, we will need to analyze the relation between $(P1)$ and $(P2)$ carefully.

5.1 Analysis

In this section, we will always denote the mean of path by x denote the mean of a path and the standard deviation by y. We will analyze the behavior of our objective $(t - x)/y$ on the set of mean-standard deviation pairs of paths. We assume that there is at least one path available with mean x smaller than the arrival time t, so that the on-time arrival probability is bigger than 0.5. Note that (given a normal distribution), any path with $x > t$ will always be worse than any path

with $x < t$, irrespective of the variance y. Therefore, we do not take into account paths with mean $x > t$. We define a function $L : \mathbb{R}^2_+ \to \mathbb{R}$ by

$$L(x, y) = \frac{t - x}{y}.$$

For $x < t$, L is strictly decreasing in x and y. In order to derive interesting properties of L, we will restrict the domain of L in three steps. First, let R denote the set of mean-standard deviation pairs corresponding of all paths in G, that is

$$R = \{(x, y) \in \mathbb{R}^2_+ : \text{ there exists a path } p \text{ in } G \text{ with mean } x \text{ and standard deviation } y\}.$$

Solving $(P2)$ is equivalent to maximizing L on R. Note that R is a finite set, so in particular, the maximum of L on R exists in R. Second, let P be the restriction of R to only the Pareto optimal paths, so

$$P = \{(x, y) \in R : \forall (x', y') \in R \text{ either } x < x' \text{ or } y < y' \text{ or } (x, y) = (x', y')\}.$$

The set P is also called the Pareto frontier. From the definition, it is clear that if $(x, y), (x', y'), (x'', y'') \in P$ and $x < x' < x''$, then $y > y' > y''$. We will prove a few facts about the behavior of L on P.

Proposition 1. *Let* $(x, y) = \arg\max_{(x', y') \in R} L(x', y')$*, then* $(x, y) \in P$*.*

Proof. Suppose, to the contrary, that L is maximal at $(x, y) \in R \setminus P$. Then $L(x, y) \geq L(x', y')$, for all $(x', y') \in R$. Since $(x, y) \notin P$, there exists $(x', y') \in R$ such that $x' < x$ or $y' < y$. Since L is strictly decreasing in both arguments, we have $L(x', y') < L(x, y)$, which is a contradiction. □

Similarly, all solutions of most reliable path problems lie on P. We will reformulate the objective of $(P1)$ to

$$(P_\alpha) \qquad \min\, (1 - \alpha) \sum_{(i,j) \in A} \bar{c}_{ij} x_{ij} + \alpha \sqrt{\sum_{(i,j) \in A} \bar{\sigma}^2_{ij} x_{ij}}.$$

If $\alpha < 1$, we solve this problem by setting $\beta = \alpha/(1 - \alpha)$ in the 'old' objective. If $\alpha = 1$, we need to find the path with minimum standard deviation. This is equivalent to finding the path with minimum variance, which is a shortest path problem, that we can easily solve. Conversely, all 'old' objectives with reliability coefficient $\beta \geq 0$ are equivalent to (P_α) with $\alpha = \beta/(1 + \beta)$. In particular, we only need to consider $\alpha \in [0, 1]$.

Proposition 2. *Let* (x, y) *be the mean-standard deviation pair of the optimal solution of* (P_α)*. Then* $(x, y) \in P$*.*

Proof. The proof is similar to the proof of Proposition 1. □

Third, we restrict P by only taking into account solutions of a most reliable path problem, so define

$$Q = \{(x, y) \in \mathbb{R}^2_+ : (x, y) \text{ is the mean-standard deviation pair of an the optimal solution of } P_\alpha\}.$$

From Proposition 2, it is clear that $Q \subseteq P$. We will go on to show that mean-standard deviation pairs of paths that maximize L are part of Q. First, we prove an auxiliary lemma that characterizes element of Q in a different way.

Lemma 3. *Let $(x, y) \in P$, then $(x, y) \in Q$ if and only if (x, y) does not lie above a line piece between two elements in P.*

Proof. Let $(x, y), (x', y'), (x'', y'') \in P$ such that $x' < x < x''$ and $y' > y > y''$. By constructing line pieces between (x', y') and (x'', y''), we obtain the following relations:

$$y > y'' + \frac{x - x''}{x' - x''}(y' - y'') \implies \frac{x' - x''}{y' - y''} < \frac{x - x''}{y - y''} \implies \frac{x' - x''}{y'' - y'} > \frac{x'' - x}{y - y''} \tag{21}$$

and

$$y > y' + \frac{x - x'}{x'' - x'}(y'' - y') \implies \frac{x'' - x'}{y'' - y'} > \frac{x - x'}{y - y'} \implies \frac{x' - x}{y - y'} > \frac{x' - x''}{y'' - y'}. \tag{22}$$

For (x, y) to be in Q, we need to find $0 \leq \alpha \leq 1$ such that the following two hold

$$(1 - \alpha)x + \alpha y \leq (1 - \alpha)x' + \alpha y' \implies \alpha(y - y') \leq (1 - \alpha)(x' - x) \implies \frac{\alpha}{1 - \alpha} \geq \frac{x' - x}{y - y'}$$

and

$$(1 - \alpha)x + \alpha y \leq (1 - \alpha)x'' + \alpha y'' \implies \alpha(y - y'') \leq (1 - \alpha)(x'' - x) \implies \frac{\alpha}{1 - \alpha} \leq \frac{x'' - x}{y - y''}$$

Such α exists precisely whenever

$$\frac{x' - x}{y - y'} \leq \frac{x'' - x}{y - y''},$$

but this is impossible because of (21) and (22). This proves the claim. □

The function L has an interesting property if we look at lines in plane.

Lemma 4. *The mapping $L_x : x \mapsto L(x, ax + b)$ is monotone on $\{x \in \mathbb{R} : ax + b > 0\}$.*

Proof. We calculate the derivative

$$\frac{d}{dx} L(x, ax + b) = -\frac{at + b}{(ax + b)^2}.$$

Note that $(ax + b)^2 \geq 0$, for all x. Since $at + b$ does not change sign, the derivative is either positive everywhere or negative everywhere, and the result follows. □

We are now ready to prove a result that is crucial in our solution procedure.

Theorem 5. *Let $(x, y) = \arg\max_{(x', y') \in R} L(x', y')$, then $(x, y) \in Q$.*

Proof. Let $(x, y) \in R \setminus Q$ such that L is maximal at (x, y). From Proposition 1, it follows that $(x, y) \in P$. By Lemma 3, since $(x, y) \notin Q$, there exist $(x', y'), (x'', y'') \in P$ such that $x' < x < x''$ and $y' > y > y''$ and (x, y) lies above the line through (x', y') and (x'', y''). Therefore, we can pick (a, b) on the line piece between (x', y') and (x'', y''), such that $a \leq x$ and $b \leq y$. Then $L(a, b) \geq L(x, y) \geq L(x', y')$ and $L(a, b) \geq L(x, y) \geq L(x'', y'')$. However, this contradicts the monotonicity property of Lemma 4. □

It follows from Theorem 5 that, in order to find the path that maximizes the objective L, we can restrict our attention to Q. We can consider L as a function of α by evaluating L at the mean-standard deviation pair of the optimal path in (P_α). Since all elements of Q are mean-standard deviation pairs of optimal paths in (P_α), there exists $\alpha \in [0, 1]$ such that the optimal path in (P_α) is the path that maximizes L. To find this α efficiently, we will first show that L is unimodal in α. This means that, if we increase α, as soon as L starts decreasing, L will never increase again. Note that x increases with α, because if we increase α, more weight is put upon the standard deviation term, hence we will never get a lower mean. Therefore, it suffices to prove the unimodality property in terms of x.

Theorem 6. *Let $(x, y), (x', y'), (x'', y'') \in Q$ and suppose $L(x, y) > L(x', y')$, $x < x' < x''$ and $y'' < y' < y$. Then we have $L(x'', y'') < L(x', y')$.*

Proof. Suppose, to the contrary, that $L(x', y') < L(x'', y'')$. Since $(x', y') \in Q$, we can pick (a, b) on the line piece between (x, y) and (x'', y''), such that $a \geq x'$ and $b \geq y'$. Note that we have $L(a, b) \leq L(x', y')$, therefore $L(a, b) < L(x, y)$ and $L(a, b) < L(x'', y'')$. We can now apply Lemma 4 to the line through $(x, y), (a, b)$ and (x'', y''). Since $L(a, b) < L(x, y)$, L must be monotonically decreasing in x. However, this contradicts the assumption that $L(a, b) < L(x'', y'')$. Therefore, we must have that $L(x'', y'') < L(x', y')$. □

Let L_α be the function that maps α to the evaluation of L at the mean-standard deviation pair of the optimal path of (P_α), then L_α is unimodular. It is well known that the maximum of strictly unimodular functions can be found using golden-section search. Unfortunately, since L_α is piecewise constant, it is not strictly unimodular, therefore we cannot guarantee to find the maximum in a finite number of steps. For small instances, we can instead do a path enumeration to generate the Pareto frontier P. However, this is not feasible anymore for larger instances. Instead, we propose an algorithm that combines golden-section search with a bisection method.

Algorithm 2
Set $\phi = (1 + \sqrt{5})/2$ and choose $\varepsilon > 0$
Initialize $a = 0, b = 1, c = b - (b - a)/\phi$ and $d = a + (b - a)/\phi$
Calculate $L_\alpha(a), L_\alpha(b), L_\alpha(c)$ and $L_\alpha(d)$
while $|b - a| \geq \varepsilon$ **do**
  if $L_\alpha(c) > L_\alpha(d)$ **then**
    Set $b = d$, $L_\alpha(b) = L_\alpha(d)$ and $d = c$, $L_\alpha(d) = L_\alpha(c)$
    Let $c = b - (b - a)/\phi$ and calculate $L_\alpha(c)$
  end if
  if $L_\alpha(c) < L_\alpha(d)$ **then**
    Set $a = c$, $L_\alpha(a) = L(c)$ and $c = d$, $L_\alpha(c) = L(d)$
    Let $d = a + (b - a)/\phi$ and calculate $L_\alpha(d)$ using Algorithm 1
  end if
  if $L_\alpha(c) = L_\alpha(d)$ **then**
    Search for improvements on $[a, c]$ and $[d, b]$ using bisection
    if improvement is found on $[a, c]$ **then**
      Set $b = c, L_\alpha(b) = L_\alpha(c)$ and calculate $c, d, L_\alpha(c)$ and $L_\alpha(d)$
    end if
    if improvement is found on $[d, b]$ **then**
      Set $a = d$, $L_\alpha(a) = L_\alpha(d)$ and calculate $c, d, L_\alpha(c)$ and $L_\alpha(d)$
    end if

```
      if no improvements found on [a, c] and [b, d] then
        Let p* be the optimal path in (P_α) with α = c
        break
      end if
    end if
  end while
  return p*
```

In the bisection procedure, we keep cutting intervals in half until the length of the interval is smaller than ϵ. Since $(P1)$ is solved by a heuristic, Algorithm 2 is not exact. Khani and Boyles (2015) develop an algorithm to solve $(P1)$ exactly. Their approach for solving $(P1)$ is actually similar to our approach for solving $(P2)$ in some ways. The most reliable path, as measured by a weighted sum of mean and standard deviation, can be found among the paths that minimize a weighted sum of mean and variance. Therefore, a search procedure over the Pareto frontier in terms of mean and variance solves $(P1)$ to optimality. Although we did not implement this procedure in this thesis, Algorithm 2 can be modified to use the exact procedure to solve $(P1)$ instead of using Algorithm 1. In this case, Algorithm 2 is exact, if two additional assumptions hold. First, if $L(x, y) = L(x', y')$, then $(x, y) = (x', y')$. This assumption is not very restrictive in practice, since if we estimate mean and variance from data and assume continuous probability distributions, different paths will have different objective almost surely. Because of this assumption, if $L_\alpha(c) = L_\alpha(d)$, we do not have to look for improvements on $[c, d]$. Second, we need that the optimal path is optimal on an interval for α of length at least ε. This makes sure that we do not skip over the optimal value of α.

5.2 Example

We will illustrate the theory using a small example, that allows us to enumerate all paths and generate the Pareto frontier. Figure 5 depicts the mean and variance parameters on the edges.

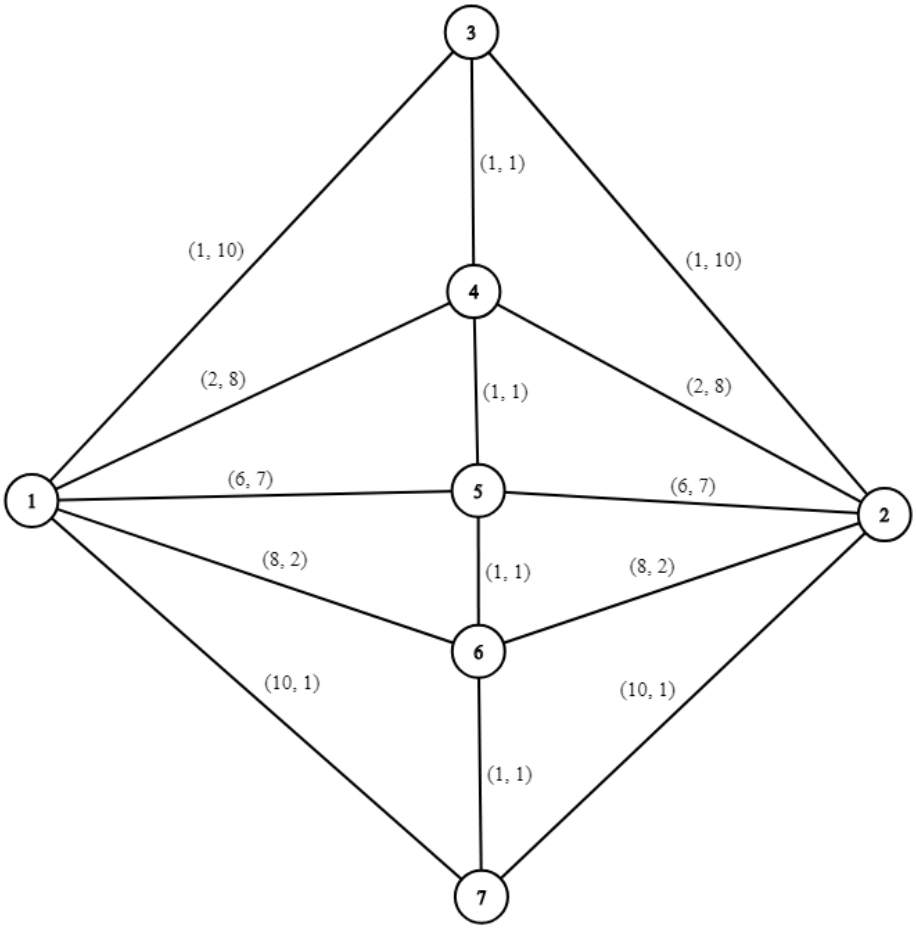

Figure 5: Graph $G = (N, A)$, with mean-variance pairs on the arcs

Our objective is to maximize the probability of arriving at 2 before $t = 25$, starting from 1. The five most interesting paths are those with two arcs, that go from 1 to 2 via one of 3, 4, 5, 6 or 7. Table 4 contains the relevant parameters.

Table 4: Mean, standard deviation, and corresponding objective L for some paths in G

Path	x	y	$L(x, y)$
(1, 3, 2)	2	4.47	5.15
(1, 4, 2)	4	4	5.25
(1, 5, 2)	12	3.74	3.47
(1, 6, 2)	16	2	4.5
(1, 7, 2)	20	1.41	3.54

The set R of mean-standard deviation pairs of all paths in G is plotted in Figure 6. Observe that the Pareto frontier P consists of the mean-standard deviation pairs of all paths contained in Table 4, together with the paths (1, 4, 5, 6, 2) and (1, 5, 6, 2). However, from Lemma 3, we know

that Q consists of only the paths (1, 3, 2), (1, 4, 2), (1, 6, 2) and (1, 7, 2). Therefore, we know from Theorem 5 that L attains its maximum at one of these four paths.

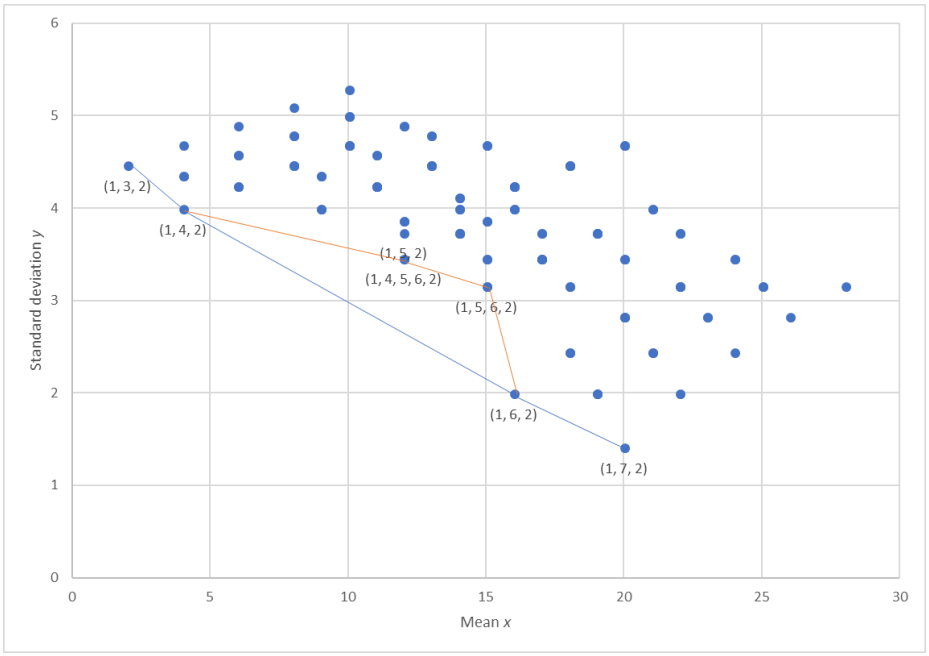

Figure 6: Mean-standard deviation pairs of all paths in G

The optimal path from 1 to 2 is (1, 4, 2). By calculating points of intersection, we obtain that this path is optimal in (P_α) for $0.81 \leq \alpha \leq 0.86$. From Figure 7, we can confirm that this example indeed satisfies all properties that we proved. In particular, L_α is unimodal in α. The value of L_α changes exactly at the kinks in the lower envelope of lines corresponding to objectives of (P_α). In particular, note that the path (1, 5, 2) is never optimal in (P_α). In the first step of the algorithm, we have $L_\alpha(c) = L_\alpha(d)$. The bisection procedure finds an improvement to the right of $d \approx 0.62$. The algorithm then cuts the interval on the left once and on the right once. We now have $c \approx 0.82$, $d \approx 0.85$, and $L_\alpha(c) = L_\alpha(d)$, and no improvements are found to the left or the right, so this is the optimal solution.

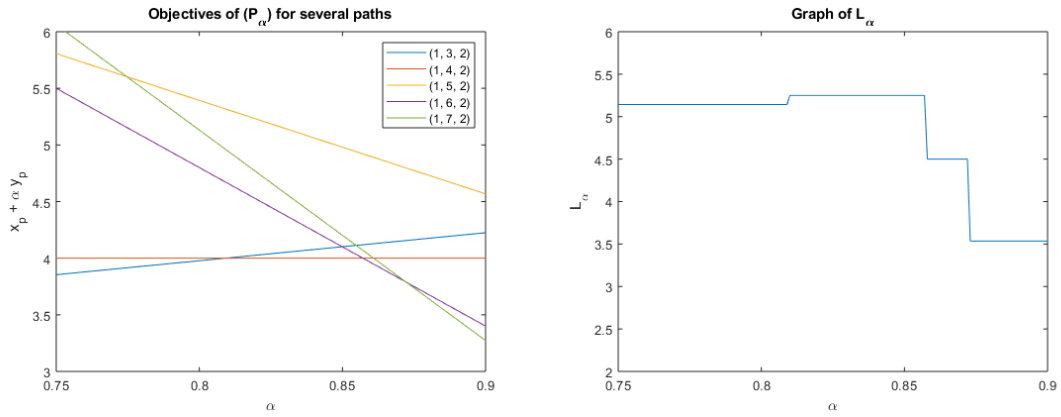

Figure 7: The graph on the left depicts the objectives of P_α for five paths from 1 to 2 in G. The graph on the right shows the objective L_α. In both cases, α runs from 0.75 to 0.90, because there are no changes outside this interval.

5.3 Computational results

Now we have seen how to find the path that maximizes the probability of arriving before a certain time, we go on to test this algorithm on a number of randomly generated graphs, to get an idea of the added value. We test the algorithm on each of the ten instances of 10 and 100 nodes, with low and high variance, as used in Section 4. Let (x_{ij}) be the shortest path in G, with arc weights given by the mean $\bar{c}_{ij}$. We determine the arrival time t using the formula

$$t = \gamma \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij}, \tag{23}$$

where γ is a scalar. To make sure that the theory of this section applies, at least we need $\gamma \geq 1$. If γ is close to 1, it is very likely that the shortest path is also the maximum probability path. On the other hand, if γ is big, almost all paths will have a very high probability of arriving on time, which is not interesting either. Therefore, we choose $\gamma = 1.2$. We compare the on-time arrival probability of the maximum on-time arrival probability path with the shortest path.

Table 5: On-time arrival probability of the shortest path and the path that maximizes on-time arrival probability

		Low variance		High variance	
$\|N\|$	Instance	Shortest	Max. prob.	Shortest	Max. prob.
10	1	0.835	0.835	0.755	0.755
	2	0.855	0.855	0.773	0.773
	3	0.969	0.969	0.906	0.906
	4	0.834	0.834	0.754	0.754
	5	0.827	0.928	0.747	0.849
	6	0.687	0.687	0.635	0.635
	7	0.843	0.843	0.762	0.762
	8	0.782	0.782	0.709	0.709
	9	0.736	0.736	0.672	0.672
	10	0.744	0.744	0.678	0.678
100	1	0.975	0.987	0.917	0.943
	2	0.942	0.942	0.866	0.866
	3	0.934	0.935	0.857	0.858
	4	0.928	0.976	0.850	0.920
	5	0.976	0.996	0.919	0.971
	6	0.943	0.983	0.869	0.934
	7	0.962	0.977	0.895	0.922
	8	0.911	0.999	0.829	0.983
	9	0.965	1.000	0.901	0.995
	10	0.948	0.985	0.875	0.938

For the graphs with 10 nodes, the shortest path maximizes on-time arrival probability on all instances, except instance 5. Presumably, these graphs are too small to have interesting alternative paths. Among the graphs with 100 nodes however, the path that maximizes on-time arrival probability differs from the shortest path on all instances, except instance 2. Therefore, running Algorithm 2 to find the maximum on-time arrival probability path in general offers a definite improvement over finding the most reliable path for some guess of the reliability coefficient β. The average running time of Algorithm 2 equals 0.035 seconds for graphs with 10 nodes and 0.080 seconds for graphs with 100 nodes.

6 Adaptive routing

Depending on the correlation structure, if traveling a link takes longer or shorter than usual, we may have additional information on the link travel times of other links. Correlation between two links may be positive, e.g. if both are affected by the same incident, or negative, when these roads are often used as substitutes. Depending on actual travel times, it may be beneficial to change the route on a trip. Boyles and Waller (2007) model the dependence structure in a way that resembles a Markov process, i.e. the probability distribution on an arc is completely determined by the actual travel time of any preceding arc. They develop a label-setting algorithm to find the optimal path, as measured by a polynomial utility function, where recourse decisions are allowed. In this section, we try to stay close to the most reliable path problem as defined in Section 2, in the sense that the same objective function is used. For convenience, we assume that the means, variances and covariances of all

links are known beforehand. Furthermore, we assume that travel times are (multivariate) normally distributed, because this allows us to specify conditional distributions in terms of the parameters. Let $\bar{c}$ be the vector of means and Σ the corresponding covariance matrix. Let $A = A_1 \cup A_2$, where A_1 denotes the set of visited arcs and A_2 the set of unvisited arcs, so that $A_1 \cap A_2 = \emptyset$. Let $C_2|C_1$ denote the set of random variables that denote the travel times corresponding to the unvisited arcs, conditional on the actual travel times of the visited arcs. We know that $C_2|C_1 = c_1$ is multivariate normally distributed with mean

$$\bar{c}_2|(C_1 = c_1) = \bar{c}_2 + \Sigma_{21}\Sigma_{11}^{-1}(c_1 - \bar{c}_1) \tag{24}$$

and covariance matrix

$$\Sigma_{22}|(C_1 = c_1) = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}. \tag{25}$$

Here we use that we can order the arcs $(i, j) \in A$ in such a way that

$$\bar{c} = \begin{pmatrix} \bar{c}_1 \\ \bar{c}_2 \end{pmatrix}$$

and

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}.$$

Furthermore, unconditionally the component C_1 is multivariate normally distributed with mean $\bar{c}_1$ and covariance matrix Σ_{11} (similar for C_2). To allow for adaptive routing, we can modify the algorithm used for solving $(P1)$ as follows:

Algorithm 3
Let $v \in N$ denote the node we have last visited, initialize $v = o$
Let $G' = (N', A')$ denote the graph we are currently working on, initialize $G' = G = (N, A)$
Let p^* be the current path from o
while $v \neq d$ **do**
  Calculate most reliable v-d path p on G', as specified in $(P1)$, using Algorithm 1
  Move to the next node on p and update v and p^*
  Let $N'' = N' \setminus \{v\}$ and $A'' = A' \setminus \{v\}$ and set $G' = (N', A')$
  Update the mean and covariance parameters on A' using equations (24) and (25)
end while
return p^*

Compared to Algorithm 1, the required number of computation is increased by a factor at most equal to the length of the optimal path. Since we only remove one node in each iteration, the inverse operation in equations (24) and (25) is in fact a division. It is important to note that in each calculation of the most reliable path, we ignore the correlation between arcs, since we solve each reliable path problem using Algorithm 1. That is, the objective is actually not equal to a weighted sum of mean and standard deviation of a path, since the variance of a path is not equal to the sum of the link variances. However, if correlations are small, this will not make a big difference. Moreover, while Xing and Zhou (2011) also derive an algorithm that solves the most reliable path problem with link correlation, they also investigate the use of the algorithm that assumes no link correlation to solve a most reliable path problem with link correlation. Although there may be loss of accuracy due to ignoring correlation, the algorithm that assumes no link correlation is computationally much cheaper and has significantly lower optimality gaps. Therefore, Algorithm 1 can be regarded as a good heuristic to solve the most reliable path problem with correlated link travel times.

6.1 Example

In graphs with low correlation between arcs, it is unlikely that a route is changed along the way, since the available information on unvisited arcs will not change a lot. Furthermore, in smaller graphs there is often a clear optimal path, so route changes will not happen very often. In this section we will look at an example where a route change actually occurs. We consider a graph $G = (N, A)$ with $|N| = 100$ for which the arcs have relatively high variance. (This example is also included in the numerical experiments in Section 4.) Instead of using historical data, we sample the actual travel times from a normal distribution. The nodes $v \in N$ are labeled $1, \ldots, 100$ and we want to travel from 58 to 2. For $\beta = 4$, the most reliable path is found to be

$$58, 95, 54, 74, 13, 64, 49, 37, 94, 38, 2.$$

We will call this path the fixed path. In the adaptive routing algorithm, the first route change occurs after traveling the third arc (54, 74). At the beginning, the mean travel times on the arcs (58, 95), (95, 54) and (74, 13) equal 1.09, 5.75, and 5.33, with variances 1.14, 0.62 and 0.28, respectively. The actual travel times on the arcs equal 2.75, 5.25 and 5.40. The travel time on (58, 95) is very unlikely to happen and seems to have significant impact on the parameters on the other arcs. Algorithm 3 finds the following path, which we will call the adapted path:

$$58, 95, 54, 74, 76, 57, 97, 32, 69, 89, 88, 38, 2.$$

To illustrate how the expected arrival time evolves along the way and to compare the two paths, we will plot confidence of the arrival times, as a function of time. That is, just before traveling each link, we evaluate the current time and estimate the arrival time from the most recent parameters. At time s, we know with 95% probability that the arrival time T satisfies

$$s + \bar{c}_p - 1.96\sqrt{\sigma_p^2} \leq T \leq s + \bar{c}_p + 1.96\sqrt{\sigma_p^2}.$$

Here p is the best found path to the destination for the adapted path, while for the fixed path it is just the tail of that path. Of course, we use the same random draw from the multivariate normal distribution for both paths.

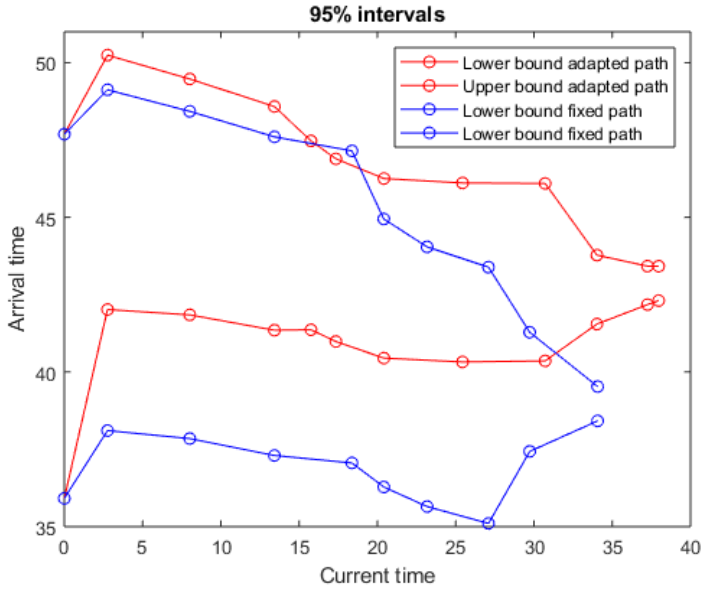

Figure 8: 95% probability intervals for the fixed path and the adaptive path

Observe from Figure 8 that the rerouting decision is already made after the first arc. Since $\beta = 4$, a lot of weight is put upon the variance, and the variance of the new path is smaller. In the end, it turns out that the fixed path has a lower total travel time. This example shows that it can happen that the uncertainty of the final arrival time may actually increase while moving to the next node, in case the actual travel time on this link is unusual.

6.2 Computational results

To get some intuition on how often route changes occur and what the effect of these route changes is, we test Algorithm 3 on the same graphs as in Section 5.3, with reliability coefficient $\beta = 1.27$. For each graph, we randomly generated a covariance matrix. We run Algorithm 3 for 10 samples of travel times, for each graph. Let T_f and T_a denote the actual total travel times of the fixed path and the adapted path, respectively. For each sample, the relative gap r is calculated as

$$r = \frac{T_f - T_a}{T_f},$$

so that a positive gap means that one saves time by using the adapted path. Table 6 contains the average gap over all samples, as well as the number of times where the fixed path and adapted path are different.

Table 6: Average relative gap and the number of samples for which the adapted path differs from the fixed path

		Low variance		High variance	
$\|N\|$	Instance	Gap	# diff.	Gap	# diff.
10	1	0.020	2	0	0
	2	0	0	0	0
	3	0	0	-0.097	8
	4	0	0	0	0
	5	0	0	0	0
	6	0	0	0	0
	7	0	0	0	0
	8	0	0	0	0
	9	0	0	0	0
	10	0	0	0	0
100	1	0.014	3	0.003	6
	2	0	0	0	0
	3	0.015	10	0.008	8
	4	0	0	-0.012	10
	5	0	0	0	0
	6	-0.012	10	0.000	4
	7	-0.018	10	0.020	10
	8	0.012	7	0.014	3
	9	0	0	0	0
	10	0	0	0	0

As expected, more route changes occur in larger graphs. If we measure the performance of Algorithm 3 by the amount of time saved, then the results are mixed. As seen in the example, it is perfectly possible that the actual arrival time increases because of a route changes. However, fundamentally the actual arrival time is a flawed measure in this case, since the objective function also consists of a standard deviation term. The algorithm is constructed in such a way that a route change improves the accuracy of the predicted arrival time along the way. In fact, the amount of time saved might not be important at all if the desired arrival time is fixed. Furthermore, note that the graphs and covariance matrices that we used are very artificial. In real-life situations, where correlation structures are less 'random', the performance of Algorithm 3 might be better. The average running time per sample is 0.012 seconds for graphs with 10 nodes and increases to 4.403 seconds for graphs with 100 nodes. This increase in running time partially comes from the increase in running time of Algorithm 1, but also the construction of the new parameters after each link.

7 Conclusion

In this thesis, we studied the problem of finding the most reliable path in a network, when exact travel times on arcs are unknown. As a first step, we studied the framework introduced in Xing and Zhou (2011). That is, for each arc the mean and variance of the travel time are known, and the objective is to minimize a weighted sum of mean and standard deviation of the travel time of a path. In order to handle this non-linear and non-convex objective, we implemented the algorithm by Xing and Zhou (2011) and tested the algorithm on a number of graphs with different sizes. The

algorithm converged very quickly and the optimality gaps were similar to those found by Xing and Zhou.

We extended the most reliable path problem in two ways. First, we changed the objective to maximizing the on-time arrival probability for a given arrival time at the destination node. If we assume that the travel times are normally distributed, this problem can be solved using the the algorithm for finding the path that minimizes a weighted sum of mean and standard deviation. More specifically, we derived a number of properties about the behavior of the objective on specific domains of mean-standard deviation pairs. This allowed us to propose an algorithm that will always find the path with maximum on-time arrival probability, if some weak conditions hold. We illustrated the theory and the algorithm using an example. By testing the algorithm on a number of randomly generated graphs, it turned out that that the obtained solutions do indeed increase the on-time arrival probability as compared to the average shortest paths, especially on the larger graphs.

Second, we extended the most reliable path problem to allow for changing routes along the way. Depending on the correlation structure, information of actual travel times on visited arcs may give additional information about travel times on unvisited arcs. We proposed an algorithm that, after traveling an arc, updates the parameters on unvisited arcs and solves the most reliable path problem again. Although this algorithm has long running times, it was interesting to see how actual travel times affected the chosen path and the expected arrival time. The route changes that occurred on the test data led to mixed results, in the sense that the actual travel time may just as well increase.

Finding reliable paths can be done in different settings and using different measures of reliability. Therefore, further research should focus on finding efficient algorithms in more general probabilistic settings. One could think, for example, of time-dependent travel times. Furthermore, new research on traffic estimation will help us to find adequate assumptions on travel time distributions.

References

Boyles, S., & Waller, S. (2007). *Online routing with nonlinear disutility with arc cost dependencies.* (Presented at the Sixth Triennial Symposium on Transportation Analysis (TRISTAN VI), Phuket, Thailand)

Chen, B., Lam, W., Sumalee, A., Li, Q., Shao, H., & Fang, Z. (2012). Finding reliable shortest paths in road networks under uncertainty. *Network and Spatial Economics*, *13*, 123-148.

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

Fan, Y., Kalaba, R., & Moore, J. (2005). Arriving on time. *Journal of Optimization Theory and Applications*, *127*(3), 497-513.

Held, M., Wolfe, P., & Crowder, H. (1974). Validation of subgradient optimization. *Mathematical Programming*, *6*, 62-88.

Khani, A., & Boyles, S. (2015). An exact algorithm for the mean-standard deviation shortest path problem. *Transportation Research Part B*, *81*, 252-266.

Nie, Y., & Wu, X. (2009). Shortest path problem considering on-time arrival probability. *Transportation Research Part B*, *43*(6), 597-613.

Noland, R., Small, K., Koskenoja, P., & Chu, X. (1998). Simulating travel reliability. *Regional Science and Urban Economics*, *28*(5), 535-564.

Sen, S., Pillai, R., Joshi, S., & Rathi, A. (2001). A mean-variance model for route guidance in advanced traveler information systems. *Transportation Science*, *35*, 37-49.

Sivakumar, R., & Batta, R. (1994). The variance-constrained shortest path problem. *Transportation Science*, *28*(4), 309-316.

Srinivasan, K., Prakash, A., & Seshadri, R. (2014). Finding most reliable paths on networks with correlated and shifted log-normal travel times. *Transportation Research Part B*, *66*, 110-128.

Xing, T., & Zhou, X. (2011). Finding the most reliable path with and without link travel time correlation: A Lagrangian substitution based approach. *Transportation Research Part B*, *45*(10), 1660-1679.