Spatially-Dependent Reliable Shortest Path Problem

Vishal Kanna Anand, Andrew Constantinescu, Sugumar Prabhakaran
Course Instructor: Prof. Jonathan Kelly
2022/04/21

Introduction & Motivation

- **Objective**: solve the spatially-dependent reliable shortest path problem (SD-RSPP)
  - RSPP is variation of shortest path problem where weights have probability distribution
  - numerous approaches exist to solve RSPP (ex. genetic algorithm, lagrangian relaxation, monte-carlo simulation, etc.)
- **Motivation and applications**:
  - SD-RSPP is a better problem formulation for certain real-world applications
  - global route-planning for self-driving cars, Google Maps, network routing
- **Approaches used in our project**:
  - Mixed-Integer Programming (MIP) model
  - Hierarchical Multi-criteria A* (SDRSP-HA*) algorithm

Literature Review

- Frank H (1969) Shortest paths in probabilistic graphs
  - Mirchandani PB (1976) Shortest distance and reliability of probabilistic networks
- Sivakumar RA, Batta R (1994) The variance-constrained shortest path problem.
  - Sen S, Pillai R, Joshi S, Rathi AK (2001) A mean-variance model for route guidance in advanced traveler information systems.
  - Lozano L, Medaglia AL (2013) On an exact method for the constrained shortest path problem.
- Chen A, Ji Z (2005) Path finding under uncertainty
  - Nie YM, Wu X (2009) Shortest path problem considering on-time arrival probability.
  - Ji Z, Kim YS, Chen A (2011) Multi-objective-reliable path finding in stochastic networks with correlated link costs:

Background and Notation

Spatially-Dependent Reliable Shortest Path (SD-RSP) Problem-

uncertainty of a link is dependent on the uncertainty of neighbouring links

Notation

- N: set of nodes, with $N = \{1, 2, \ldots, n\}$
- A: set of links a_{ij}: link from node i to node j
- $T_{ij} = (t_{ij}, \sigma_{ij})$: normal distribution of travel-time for a_{ij}
- r, s: start and end nodes respectively
- α: User defined confidence level:
- ($\alpha > 0.5$: risk-averse, $\alpha = 0.5$: risk-neutral, $\alpha < 0.5$ risk-seeking)
- k: spatial dependency factor

$$
\begin{array}{c c}
 & \begin{array}{cccccc} a_{12} & a_{13} & a_{14} & a_{23} & a_{35} & a_{45} \end{array} \\
\begin{array}{c} a_{12} \\ a_{13} \\ a_{14} \\ a_{23} \\ a_{35} \\ a_{45} \end{array} &
\begin{bmatrix}
2 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 \\
-1 & -0.5 & 1 & 0 & 0 & 0 \\
2 & -1 & -0.3 & \boxed{2} & 0 & 0 \\
0.3 & 1.5 & -0.4 & 2 & 6 & 0 \\
-0.2 & -0.6 & 0.5 & -0.4 & -1.5 & 1
\end{bmatrix}
\end{array}
$$

link travel-time
variance: $(\sigma_{ij})^2 = 2$

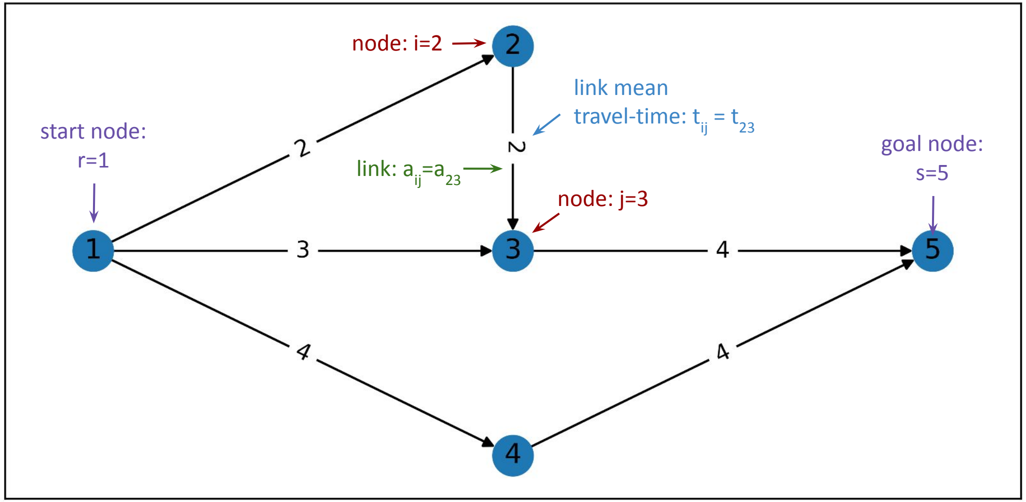

Figure 1: Example Directed-Arc Network Graph, G = (N, A)

Problem Formulation

- The three main equations for the SD-RSPP formulation are:

Mean path travel-time (from node *r* to node *s*):

$$t_u^{rs} = \sum_{m=1}^{\lambda} (t_{ij})^m \tag{1}$$

Standard deviation of path travel-time (*r* to *s*):

$$\sigma_u^{rs} = \sqrt{\sum_{m=1}^{\lambda} (\sigma^m)^2 + \sum_{n=1}^{k} \sum_{m=1}^{\lambda-n} 2\mathrm{cov}(a^m, a^{m+n})} \tag{2}$$

Inverse CDF path travel-time (*r* to *s*):

$$\Phi_{rs,u}^{-1}(\alpha) = t_u^{rs} + z_\alpha \sigma_u^{rs} \tag{3}$$

- Note: the inverse CDF is non-additive and cannot be solved with dynamic programming:

$$(\Phi_{rs}^{-1}(\alpha) \neq \Phi_{rj}^{-1}(\alpha) + \Phi_{js}^{-1}(\alpha))$$

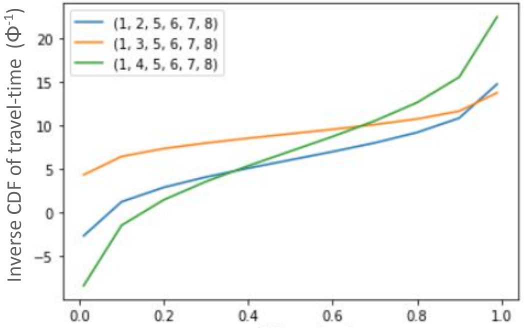

Figure 2: Inverse CDF (Φ^{-1}) vs. confidence (α)

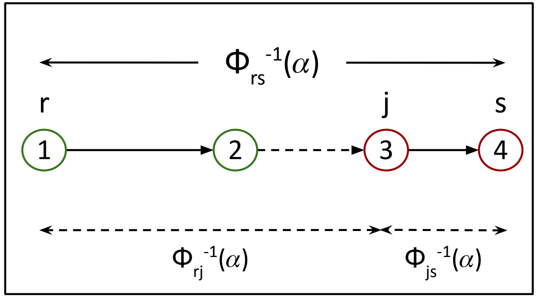

Figure 3: Inverse CDF is non-additive

Mixed Integer Programming (MIP) Model

Objective Function:

$$\min_{x_{ij}} \Phi^{-1}(\alpha) = \min_{x_{ij}} (t_u + z_\alpha \sigma_u) \tag{4}$$

link mean

$$\min_{x_{ij}} \sum_{a_{ij} \in A} (t_{ij} \cdot x_{ij}) + z_\alpha \cdot \sqrt{\sum_{a_{ij} \in A} (\sigma_a)^2 \cdot x_{ij} + 2 \cdot \sum_{a_{ij} \in A} \sum_{b_{kl} \in A} (\sigma_{ab} \cdot x_{ij} \cdot x_{kl})} \tag{4.a}$$

link variance

link covariance

subject to:

Decision Variable:

$$x_{ij} \in \{0, 1\}, \quad \forall a_{ij} \in A \tag{5}$$

Linear Constraints:

$$\sum_{j \in SCS(i)} x_{ij} - \sum_{k \in PDS(i)} x_{ki} = 1, \quad \forall i = r \tag{6}$$

$$\sum_{j \in SCS(i)} x_{ij} - \sum_{k \in PDS(i)} x_{ki} = 0, \quad \forall i \neq r; i \neq s \tag{7}$$

$$\sum_{j \in SCS(i)} x_{ij} - \sum_{k \in PDS(i)} x_{ki} = -1, \quad \forall i = s \tag{8}$$

```python
# INITIALIZE GUROBI MIP MODEL
import gurobipy as gp
model = gp.Model("reliable_shortest_path")
```

```python
    # DECISION VARIABLES
    x = model.addVars(A, vtype=gp.GRB.BINARY)   # link-path incidence variables
```

```python
    # additional variables to linearize objective function
    var = model.addVar(name='var')
    covariance = model.addVar(name='covariance')
    path_var = model.addVar(name='path_var')
    path_std = model.addVar(name='path_std')
```

```python
    # CONSTRAINTS
    #enforce one-way direction of links from node i to node j for all links
    model.addConstrs(gp.quicksum(x[i,j] for j in G.successors(i))-
                     gp.quicksum(x[k,i] for k in G.predecessors(i)) == 1
                     for i in N if i == r)
    model.addConstrs(gp.quicksum(x[i,j] for j in G.successors(i))-
                     gp.quicksum(x[k,i] for k in G.predecessors(i)) == 0
                     for i in N if (i != r and i != s))
    model.addConstrs(gp.quicksum(x[i,j] for j in G.successors(i))-
                     gp.quicksum(x[k,i] for k in G.predecessors(i)) == -1
                     for i in N if i == s)
```

```python
    # OBJECTIVE FUNCTION:  path mean + z*sqrt(var + covar)
    z = norm.ppf(alpha) # inv. cdf of std norm. distr at alpha confidence
    path_mean = gp.quicksum(t_ij[a]*x[a] for a in A)
    var = gp.quicksum(cov_matrix[idx[a],idx[a]]*x[a] for a in A)
    covariance = gp.quicksum(cov_matrix[idx[a], idx[b]]*x[a]*x[b]
                          for a in A for b in A if a !=b)
```

```python
    #linearize sqrt in objective function
    model.addConstr(path_var == var + 2*covariance)
    model.addGenConstrPow(path_var, path_std, 0.5)  # path_std = sqrt(path_var)

    model.setObjective(path_mean+z*path_std, gp.GRB.MINIMIZE)   # obj. function
    model.optimize()                                            # execute
```

SDRSP-HA*

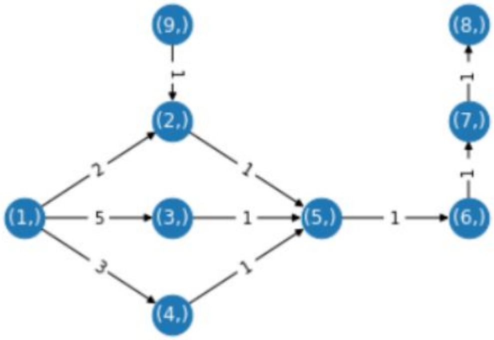

Figure 4.1: Primal network.

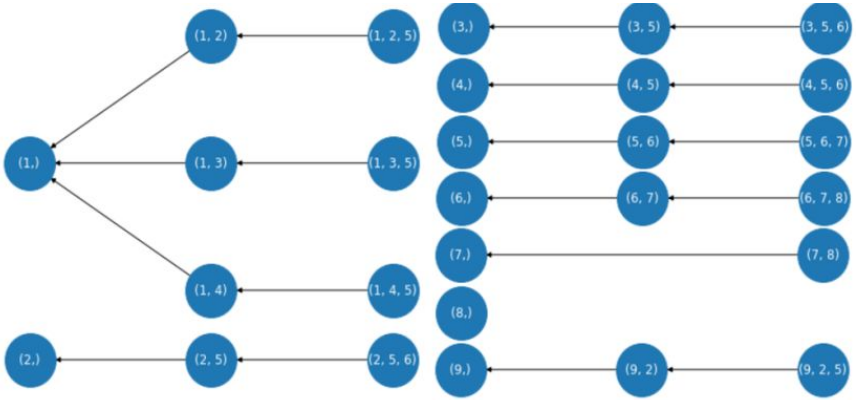

Figure 4.2: Ground hierarchy H_g network for $k = 3$.

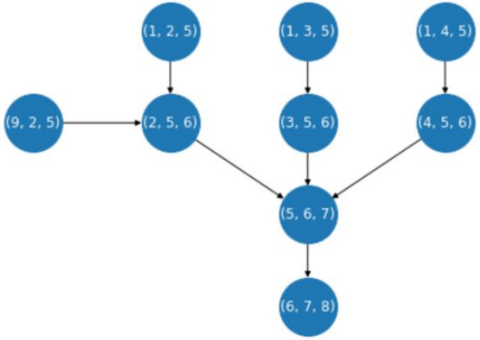

Figure 4.3: Top hierarchy H_t network for $k = 3$.

Case Study

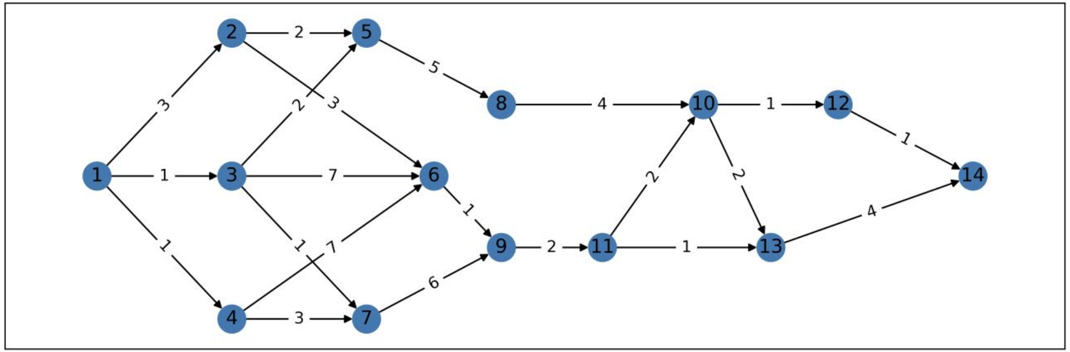

Figure 4.5: Primal network (G) used for algorithm experimentation.

$$
G_\Sigma = \begin{bmatrix}
2 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0.1 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 6 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 2 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 2 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 2 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 9 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1
\end{bmatrix}
$$

SDRSP-HA* Algorithm Results:

Reliable Shortest Path	Probability of on-time expected arrival
$\hat{p}_v^{rs} = (1, 3, 5, 8, 10, 12, 14)$	$0 < \alpha < 0.374$
$\hat{p}_u^{rs} = (1, 2, 6, 9, 11, 10, 12, 14)$	$0.374 \leq \alpha \leq 0.780$
$\hat{p}_w^{rs} = (1, 3, 7, 9, 11, 10, 12, 14)$	$0.780 \leq \alpha \leq 0.942$
$\hat{p}_x^{rs} = (1, 3, 7, 9, 11, 13, 14)$	$0.942 < \alpha < 1$

Gurobi MIP Model Results:

Reliable Shortest Path	Probability of on-time expected arrival
$\hat{p}_v^{rs} = (1, 3, 5, 8, 10, 12, 14)$	$0 < \alpha < 0.374$
$\hat{p}_u^{rs} = (1, 2, 6, 9, 11, 10, 12, 14)$	$0.374 \leq \alpha \leq 0.850$
$\hat{p}_w^{rs} = (1, 3, 7, 9, 11, 10, 12, 14)$	$0.850 \leq \alpha \leq 0.942$
$\hat{p}_x^{rs} = (1, 3, 7, 9, 11, 13, 14)$	$0.942 < \alpha < 1$

Future Work

- How do these methods extend to large road networks?
- How does the spatial-influence factor k affect computational complexity?
- What if we had time-varying travel time distributions?