VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
Faculty of Computer Science and Engineering


**Mathematical Modeling (CO2011)**

---

**Group: CC03 — Assignment Report**

# Stochastic Programming and Applications

---

| | | |
|---|---|---|
| **Supervisors:** | Nguyen An Khuong,CSE-HCMUT | |
| | Nguyen Van Minh Man, Mahidol University | |
| | Mai Xuan Toan, CSE-HCMUT | |
| | Tran Hong Tai, CSE-HCMUT | |
| | Nguyen Tien Thinh, CSE-HCMUT | |
| **Students:** | Tran Gia Linh | 2252434 |
| | Che Thanh Vy | 2252937 |
| | Huynh Ngoc Van (*Leader*) | 2252898 |
| | Nguyen Thuy Tien | 2252806 |
| | Nguyen Huu Tri | 2252842 |

Ho Chi Minh City, January 3, 2024

| Name | ID | Tasks | Contribution |
|---|---|---|---|
| Tran Gia Linh | 2252434 | - Report:<br>+ Section 2: Background<br>+ Section 3.3.3: Problem 1 Implementation<br>+ Section 4.2.1: Algorithm 1 in sub-Problem 1<br>- Code:<br>+ Problem 1<br>+ Drawing cost graph, label correcting algorithm<br>+ Successive shortest path algorithm | 24% |
| Huynh Ngoc Van | 2252898 | - Report:<br>+ Section 3.2.2: Problem 1 constraints explanation<br>+ Section 4.2.1: Algorithm 1 in sub-Problem 1<br>+ Section 5: Conclusion<br>- Code:<br>+ Problem 1<br>+ Label correcting algorithm<br>+ Successive shortest path algorithm | 24% |
| Nguyen Thuy Tien | 2252806 | - Report:<br>+ Section 1: Introduction<br>+ Section 4.2.2: Modified Algorithm 1 in sub-Problem 2<br>- Code:<br>+ Problem 1<br>+ Drawing cost graph and evacuation graph in first stage<br>+ Successive shortest path algorithm | 24% |
| Che Thanh Vy | 2252937 | - Report:<br>+ Section 3.1: Problem 1 Description<br>+ Section 4.1: Problem 2 Description | 14% |
| Nguyen Huu Tri | 2252842 | - Report:<br>+ Section 3.2.1: Problem 1 objective fuction explanation<br>+ Section 4.1: Problem 2 Description | 14% |

# Contents

# 1 Introduction

Stochastic programming is a very powerful mathematical optimization technique that solve various problem in real life world where we cannot predict perfectly each situation and the uncertainties always exists. The objective of this report is to explore the application of two-stage stochastic programming in industry manufacturing and evacuation planning. By accounting for uncertainties and risks, decision-makers can make informed decisions that lead to improved operational efficiency, reduced costs, and enhanced risk management. This report also provide the detailed example source code for solving such kinds of problems.

# 2 Background

## 2.1 Probability Spaces and Random Variables

### 2.1.1 Probability Space

In practice, several aspects influencing decisions can be uncertain. For example, in manufacturing, production and material costs may depend on fuel costs. Or transportation plan may depend on the condition of roads and weather conditions. To capture these uncertainty into mathematical models, the concepts of probability and random variables are used.
Before uncertainty is known, it is unknown that which outcome will be the result. The sample space $\Omega$ is the set of all possible outcomes. An event is a subset of the sample space $\Omega$, we denote by $\mathscr{A}$ a collection of random events. And to each event $A \in \mathscr{A}$, $P(A)$ is the *probability*.

**Definition 2.1.** (General definition of probability). A probability space consists of sample space $\Omega$ and a probability function $P$ taking an event $A \subseteq \Omega$ as input and returns a real number $P(A)$, such that $0 \leq P(A) \leq 1$. The function $P$ must satisfy the following axioms:

1. $P() = 0, P(\Omega) = 1$.
2. If $A_1, A_2, ...$ are disjoint events, then $P(\bigcup_{j=1}^{\infty} A_j) = \Sigma_{j=1}^{\infty} P(A_j)$ .

### 2.1.2 Random Variable

A random variable $X$ maps the sample space into the real line. Therefore, a random variable $X$ assigns a numerical value $X(\omega)$ to each possible outcome $\omega$. A discrete random variable can have a finite or countably infinite number of values such that $P(X = x) > 0$. A discrete random variable can be described by its probability mass function PMF.

**Definition 2.2.** (Probability mass function). The probability mass function (PMF) of a discrete random variable is the function $p_X = P(X = x)$.

A continuous random variable can have infinite number of values. It has continuous distribution and can be described by probability density function (PDF).

**Definition 2.3.** (Probability density function) PDF of a continuous random variable is the derivative $f(x)$ of the function $F(x) = P(X \leq x)$. The support of its distribution is the set of all $x$ where $f(x) > 0$.

The expectation $E$ of a random variable $x$ is computed as $\mathbf{E}(X) = \Sigma_{j=1}^{\infty} x_j P(X = x_j)$ and $\mathbf{E}(X) = \int_{-\infty}^{\infty} x f(x)\, dx$ for discretet and continuous cases, respectively. Sometimes the expectation of a random variable may not exist.

### 2.1.3 Bernoulli and Binomial

A random variable X is said have Bernoulli distribution with parameter $p$ if $P(X = 1) = p$ and $P(X = 0) = 1 - p$, where $0 < p < 1$. An experiment that result in either "success" or "failure" follows Bernoulli distribution and $p$ is called the success probability of that experiment. We write $X \sim Bern(p)$
If there are $n$ independent Bernoulli trials with the same success probability $p$ performed, we have $n$ i.i.d (Independent and Identically Distributed) random variables. Let $x$ be the total number of successes. X then follows Binomial distribution with parameters $n$ and $p$, $X \sim \text{Bin}(n, p)$. The PMF and expectation of $X \sim \text{Bin}(n, p)$ are:

1. $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$. for $k = 0, 1, 2, .., n$, and $P(x = k) = 0$ otherwise.
2. $\mathbf{E}(X) = \Sigma_{k=0}^{n} k \binom{n}{k} p^k (1-p)^{n-k} = np$.

## 2.2 Stochastic Linear Programming

### 2.2.1 Introduction to Stochastic Linear Program

Stochastic programs (SLP) are linear programs consists of uncertain data described by random variables. There are two approaches to **deal with uncertainty**:

1. The **wait-and-see** approach: If it is possible to make decisions after the observation of random variables, we can wait until the uncertainty is disclosed and solve the problem using general linear programming.
2. The **no-waiting** approach: Sometimes the decisions need to be made beforehand. For example, a manufacturer needs to buy materials before knowing the demands. To deal with these situations, there are some approaches: guess at uncertainty, probabilistic constraints.

In one-stage stochastic program, we use **no-waiting** approach to deal with uncertainty, i.e making decisions beforehand, and do not take recourse action after.

### 2.2.2 Two-Stage Stochastic Program with Recourse

For two-stage SLP, we take recourse action for decisions making before realization of uncertainty. Recourse are the actions that are taken after the realization of uncertainty $X(\omega)$. Therefore, there are two types of decision variables in two-stage SP:

- First-stage decision: decisions which have to be taken before realization of uncertainty, denoted by vector $\boldsymbol{x}$.
- Second-stage decision: decisions taken after disclosing uncertainty to minimize the expected costs of consequences, denoted by vector $\boldsymbol{y}(\omega)$ or $\boldsymbol{y}(\omega, \boldsymbol{x})$.

The sequence of decisions can be summarized as:

$$\boldsymbol{x} \implies X(\omega) \implies \boldsymbol{x}(\omega)$$

A common method to solve two-stage SLP is to consider all possible scenarios and optimize the expected costs. Notice after uncertainty is known, the optimal solution to the model considering all possible scenarios may not provide the optimal minimization for a particular observation. However, two stage SLP is still reasonable since it tackles all scenarios when we have to make

some decisions before knowing the uncertainty.
The classical two-stage SLP model with recourse [1] is:

$$\begin{aligned} &\min z = \boldsymbol{c}^T\boldsymbol{x} + \mathbf{E}_\omega[\min \boldsymbol{q}(\omega)^T\boldsymbol{y}(\omega)] \\ &\text{s.t } \mathbf{A}\boldsymbol{x}=\boldsymbol{b}, \\ &T(\omega)\boldsymbol{x} + W\boldsymbol{y}(\omega) = h(\omega), \\ &\boldsymbol{x} \geq 0, \boldsymbol{y}(\omega) \geq 0. \end{aligned} \tag{1}$$

1. Vector $\boldsymbol{x}$ of size $n_1x1$ are the first stage decision variables. The associated vectors and matrices are:
   - Vector $\boldsymbol{c}$ of size $n_1 \times 1$: cost coefficients. For example, in manufacturing problem, if $x$ represents amount of materials to buy, $\boldsymbol{c}$ can be the cost of one unit of materials.
   - $A\boldsymbol{x}=\boldsymbol{b}$, with size of $A$ and $\boldsymbol{b}$ are $m_1 \times n_1$ and $m_1 \times 1$, respectively, are the first stage constraints.
2. Vector $\boldsymbol{y}(\omega)$ of size $m_2 \times 1$ are the second stage decision variables. The purpose of designing $\boldsymbol{y}(\omega)$ is to perform recourse action (tune, modify) for $\boldsymbol{x}$ after uncertainty $\omega$ is revealed. $T(\omega)\boldsymbol{x} + W\boldsymbol{y}(\omega) = h(\omega)$ are the second stage constraints, and each component of $\boldsymbol{q}, T, h$ is possibly a random variable. $T$ is the transition matrix and $W$ is the recourse matrix.
3. The objective function consists of the first stage objective $\boldsymbol{c}^T$ and the expected value of the second stage objective $\boldsymbol{q}^T(\omega)(\omega)$. Sometimes to emphasize that $\boldsymbol{q}^T(\omega)(\omega)$ taken over all realizations of $\omega$, one may use the deterministic equivalent program notion:

$$\begin{aligned} &\min z = \boldsymbol{c}^T\boldsymbol{x} + \mathbf{E}_\omega Q(\boldsymbol{x}, \omega) \\ &\text{s.t } \mathbf{A}\boldsymbol{x}=\boldsymbol{b}, \boldsymbol{x} \geq 0, \\ &\text{where } Q(\boldsymbol{x}, \omega) = \min\{\boldsymbol{q}(\omega)^T\boldsymbol{y} | W\boldsymbol{y} = h(\omega) - T(\omega)\boldsymbol{x}, \boldsymbol{y} \geq 0\}. \end{aligned} \tag{2}$$

# 3 Problem 1: Multi-product Assembly

## 3.1 Description

| Symbols | Definition |
|---|---|
| $n$ | products |
| $m$ | total parts have to be ordered outside |
| $i$ | product, $i \in [1, n]$ |
| $j$ | part, $j \in [1, m]$ |
| $a_{ij}$ | production coefficient, i product requires $a_{ij}$ units (may be zero) of part $j$ |
| $D$ | random vector for demand for products, $D = (D_1, ...D_n)$ |
| $b_j$ | cost of preordered parts from outside per unit of part $j$ |
| $l_i$ | cost to satisfy a unit of demand for product $i$ |
| $q_i$ | unit selling price of this product |
| $s_j$ | salvage value, $s_j < c_j$ |
| $x_j$ | numbers of parts ordered, $j \in [1, m]$ |
| $z_i$ | numbers of parts produced, $i \in [1, n]$ |
| $y_j$ | numbers of parts left in inventory, $j \in [1, m]$ |
| $d$ | observed value (a realization), $d = (d_1,...,d_n)$ of random vector $D$ |

**Table 1:** *Symbols explanation of Multi-product Assembly problem*

### The interdependence of the first and second stages in two-stage stochastic programming:

- The first stage decisions are made without complete information about the future, often in the face of uncertainty. The second stage decisions are based on the realization of uncertain parameters, and therefore, they depend on the information revealed after the first stage decisions are made.
- The decisions made in the first stage influence the set of feasible options and the conditions under which decisions are made in the second stage. Without the first stage decisions, there would be no need for adaptation because there would be no initial plan to adjust.
- The overall objective of a two-stage stochastic programming problem is to optimize a certain objective function. Optimality in this context requires the coordination of decisions across both stages to achieve the best overall outcome.

In summary, as the *second-stage* problem contains random data (random demand D), its optimal value $Q(x, D)$ is a random variable. The distribution of this random variable depends on the *first-stage* decisions x, and therefore the *first-stage* problems cannot be solved without understanding of the properties of the *second-stage* problem.

## 3.2 Two-Stage Model

In **two-stage stochastic programming problem with recourse** for **Industry Production**, the general formulation is given by the following:

The first stage model is:

$$\min_{x \geq 0} \; b^\intercal x + \mathbb{E}[Q(x, D)]$$

where $\mathbb{E}[Q(x, D)]$ is the optimal solution for the second-stage model:

$$\min_{z,y} (l - q)^\intercal z - s^\intercal y$$

with the constrains:

$$s.t.\ y = x - A^\intercal z,$$
$$0 \leq z \leq d, \quad y \geq 0.$$

In this project, the first stage decision variable, also known as *here-and-now*, is $\boldsymbol{x}$ - the number of advanced ordered materials, and the *second-stage* decision variable or *wait-and-see* is $\boldsymbol{y}$, and $\boldsymbol{z}$, the remaining materials and the number of products, respectively.

### 3.2.1 Objective function description

The main goal of our two-stage stochastic problem is to find the best production plan for the supply demand, specifically minimizing the production cost. From the materials given, this is the general objective function for a supply-chain problem like ours:

$$\min\ b^\intercal x + \sum_{k=1}^{K} p_k[(l - q)^\intercal z^k - s^\intercal y^k]$$

or we can write it in matrix form like this:

$$min \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_5 \end{bmatrix}^\intercal \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_5 \end{bmatrix} + \begin{bmatrix} l_1 - q_1 \\ l_2 - q_2 \\ l_3 - q_3 \\ ... \\ l_8 - q_8 \end{bmatrix}^\intercal \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ ... \\ z_8 \end{bmatrix} - \begin{bmatrix} s_1 \\ s_2 \\ ... \\ s_5 \end{bmatrix}^\intercal \begin{bmatrix} y_1 \\ y_2 \\ ... \\ y_5 \end{bmatrix}$$

The first stage in our problem is before the demand is known, and the second stage is after. Prior to production, the manufacturer may place pre-orders for parts from third-party suppliers. The cost per part $j$, denoted by $b_j$, also serves as the first stage problem's random variable. After the demand $D = (D_1, ..., D_n)$ is revealed, the manufacturer need to decide how much of each product to produce in order to guarantee that there are as few spare components in stock as possible, while fulfilling the demand given. Here is where we are introduced to two random variables of the second-stage problem: $l_i - q_i$ and $s_j$. As stated in the symbols explanation table above, we add $l$ to our objective function due to the fact that this is additional cost to fulfill the demand. Meanwhile, we can deduct $q_i$ from the function because it reflects the money we receive back from selling each product $i$. Since we have to calculate both of these variables for each product $i$, we can combine them into one variable for simplicity. The assessed salvage values $s_j$ represent the unused parts from the initial pre-order. Observing the solution of the problem - vectors $z$ and $y$ - we can see that the second-stage decision variables depend on realization $d$ of the demand vector $D$ as well as the first-stage decision variable $x$.

In our situation, the problem given has finitely many demand scenarios, each might occur with a positive probability $p_1, ..., p_k$.

$$p_k = Bin(10, \frac{1}{2})$$

For this particular assignment, the probability is chosen to be $\frac{1}{2}$, which transform our general objective function above into:

$$b^\intercal x + \frac{1}{2}[(l-q)^\intercal \vec{z_1} - s^\intercal \vec{y_1}] + \frac{1}{2}[(l-q)^\intercal \vec{z_2} - s^\intercal \vec{y_2}]$$
$$= b^\intercal x + \frac{1}{2}[(l-q)^\intercal (\vec{z_1} + \vec{z_2}) - s^\intercal (\vec{y_1} + \vec{y_2})]$$

### 3.2.2 Constraints description

This is the industry production related problem, therefore, the constraint conditions are about the ordered materials and the number of products depends on the demand in each scenario.

*Firstly,* we consider the constraint related to the ordered materials:

$$y = x - A^\intercal z$$

The above constraint demonstrate the equation of the number of materials left in inventory, denote by $\boldsymbol{m \times 1}$ ***matrix*** $\boldsymbol{y}$. Besides, from "Symbols explanation table" in *section 3.1*, we have already known about the meaning of $\boldsymbol{m \times 1}$ ***matrix*** $\boldsymbol{x}$, $\boldsymbol{n \times 1}$ ***matrix*** $\boldsymbol{z}$ and $\boldsymbol{a_{ij}}$ which is the entries of $\boldsymbol{n \times m}$ ***matrix*** $\boldsymbol{A}$. Furthermore, there is a key point that need to be understand clearly in the constraint, the **multiplication of the transpose of *matrix* $\boldsymbol{A}$ and *matrix* $\boldsymbol{z}$**

$$A = \begin{bmatrix} a_{11} & a_{12} & ... & a_{15} \\ a_{21} & a_{22} & ... & a_{25} \\ ... & & & \\ a_{81} & a_{82} & ... & a_{85} \end{bmatrix}$$

$$A^\intercal z = \begin{bmatrix} a_{11} & a_{21} & ... & a_{81} \\ a_{12} & a_{22} & ... & a_{82} \\ ... & & & \\ a_{15} & a_{25} & ... & a_{85} \end{bmatrix}_{5\times 8} \begin{bmatrix} z_1 \\ z_2 \\ ... \\ z_8 \end{bmatrix}_{8\times 1} = \begin{bmatrix} a_{11}z_1 + a_{21}z_2 + ... + a_{81}z_8 \\ a_{12}z_1 + a_{22}z_2 + ... + a_{82}z_8 \\ ... \\ a_{15}z_1 + a_{25}z_2 + ... + a_{85}z_8 \end{bmatrix}_{5\times 1}$$

As you can easily observe that each row of the matrix $\boldsymbol{A^T z}$ show the number of material j used to produce $\boldsymbol{n}$ products; therefore, let the advance ordered material $x$ minus $\boldsymbol{A^T z}$, we obtain the remaining materials $\boldsymbol{y}$.

*Secondly,* is about the constraint that the number of products must less or equal than the demand so that it ensure the industry does not make a loss. The constraint could be illustrated as following:

$$z \leq D$$
$$or$$
$$\begin{bmatrix} z_1 \\ z_2 \\ ... \\ z_8 \end{bmatrix} \leq \begin{bmatrix} D_1 \\ D_2 \\ ... \\ D_8 \end{bmatrix}$$

## 3.3 Implementation

In this project, Python is used and its library **gurobi** help solve the **deterministic equivalent formulation** of the two stage stochastic problem - *source code*. Also, due to the lack of real world data, the data used in for simulating and solving the problem is simulated based on randomness and convention.

### 3.3.1 Simulation of data

Apart from decision variables of stage 1 and stage 2, i.e $\boldsymbol{x}$ **,**$\boldsymbol{y}$**,** $\boldsymbol{z}$, and n,m, other variables are simulated. Table 2 summarize variables.

| Variables | Data Type | Value or range |
|---|---|---|
| $n$ | $n \in \mathbb{N}$ | n = 8 |
| $m$ | $m \in \mathbb{N}$ | m = 5 |
| $\boldsymbol{b}$ | $\boldsymbol{b} = (b_j) \in M_{m\times 1}(\mathbb{N})$ | $50 \leq b_j \leq 100$ |
| $\boldsymbol{l}$ | $\boldsymbol{l} = (l_i) \in M_{n\times 1}(\mathbb{N})$ | $10 \leq l_i \leq 25$ |
| $\boldsymbol{q}$ | $\boldsymbol{q} = (q_i) \in M_{n\times 1}(\mathbb{N})$ | $q_i = (\boldsymbol{Ab} + \boldsymbol{l})_i + \epsilon_i, 20 \leq \epsilon_i \leq 50$ |
| $\boldsymbol{s}$ | $\boldsymbol{s} = (s_j) \in M_{m\times 1}(\mathbb{N})$ | $s_j = b_j - \epsilon_j, 1 \leq \epsilon_j \leq 10$ |
| $\boldsymbol{A}$ | $\boldsymbol{s} = (a_{ij}) \in M_{n\times m}(\mathbb{N})$ | $1 \leq a_{ij} \leq 10$ |
| $\boldsymbol{D}_1$ | $\boldsymbol{D}_1 = (d_i) \in M_{n\times 1}(\mathbb{N})$ | $d_i \sim$ Bin(10,0.5) |
| $\boldsymbol{D}_2$ | $\boldsymbol{D}_2 = (d_i) \in M_{n\times 1}(\mathbb{N})$ | $d_i \sim$ Bin(10,0.5) |

**Table 2:** *Summary of variables in the model*

Further explanation:

- $\boldsymbol{b}$: this vector is simulated using Discrete Uniform distribution, and the range is from 50 to 100.
- $\boldsymbol{l}$: this vector is simulated using Discrete Uniform distribution, and the range is from 10 to 25.
- $\boldsymbol{q}$: since $\boldsymbol{q}$ is the selling price of the products and some profits are expected, $\boldsymbol{q}$ must be larger than total manufacturing costs. $(\boldsymbol{Ab} + \boldsymbol{l})_i$ is the total manufacturing costs of product i. So a random value $\epsilon_i$ is added to $\boldsymbol{Ab} + \boldsymbol{l}$ to generate $\boldsymbol{q}_i$, and this random value follows Discrete Uniform Distribution with the range from 20 to 50.
- $\boldsymbol{s}$: since $\boldsymbol{s}$ is the salvage value of the material, it must be smaller than $\boldsymbol{b}$ - the costs of materials. Therefore, a random value $\epsilon_j$ is subtracted from $\boldsymbol{b}_j$ to generate $s_i$.
- $\boldsymbol{A}$: this matrix is simulated using Discrete Uniform distribution, and the range is from 1 to 10.

Simulated values of variables are:

```
b is    l is        s is        q is            Matrix A is          D_1 is      D_2 is
[[95]   [[11]       [[84]       [[1280]         [[4 5 1 1 5]         [[3]        [[8]
[52]    [13]        [33]        [1644]          [2 8 4 3 5]          [6]         [3]
[78]    [20]        [68]        [2201]          [8 3 5 9 1]          [4]         [4]
[84]    [21]        [74]        [2189]          [8 4 5 7 2]          [5]         [5]
[88]]   [19]        [69]]       [2172]          [6 7 3 2 9]          [2]         [7]
        [16]                    [1697]          [4 6 1 3 7]          [7]         [7]
        [11]                    [1931]          [3 5 5 7 4]          [7]         [1]
        [10]]                   [2185]]         [1 7 5 8 7]]         [2]]        [5]]
```

### 3.3.2 Build Two-stage Stochastic Model

According to the canonical form of equivalent deterministic problem given in section **3.2.1**, the model can be expressed in Python as:

```python
#Decision variable
x = model.addMVar((m,1), vtype = gugu.GRB.INTEGER, name = "x")
y1 = model.addMVar((m,1), vtype = gugu.GRB.INTEGER, name = "y1")
y2 = model.addMVar((m,1), vtype = gugu.GRB.INTEGER, name = "y2")
z1 = model.addMVar((n,1), vtype = gugu.GRB.INTEGER, name = "z1")
z2 = model.addMVar((n,1), vtype = gugu.GRB.INTEGER, name = "z2")

#Objective function
model.setObjective(b.T @ x + 0.5*(l-q).T @ (z1 + z2) - 0.5*(s.T)@(y1+y2))

#Add constraints
model.addConstr(y1 == x - A.T @ z1)
model.addConstr(y2 == x - A.T @ z2)
model.addConstr(z1 <= D_1)
model.addConstr(z2 <= D_2)
model.optimize()
```

### 3.3.3 Optimal solution to the problem

After solving the problem using primal simplex method for both root node and B&B nodes, the program gives the following result:

```
Optimal Solution:
Obj: -1099
x = [[140.]      y1 = [[3.]      z1 = [[3.]         y2 = [[0.]       z2 = [[5.]
     [170.]           [0.]           [3.]               [0.]            [3.]
     [108.]           [0.]           [2.]               [0.]            [4.]
     [155.]           [0.]           [5.]               [0.]            [5.]
     [151.]]          [0.]]          [2.]               [0.]]           [1.]
                                     [7.]                               [7.]
                                     [7.]                               [1.]
                                     [2.]]                              [5.]
```

# 4 Problem 2: Min-cost Flow Problem

## 4.1 Description

When a disaster occurs, the foremost priority is to minimize the loss of life. In situations like earthquakes, hazardous chemical leaks, and hurricanes, individuals in affected areas need to be relocated to safe zones. Li Wang's paper [2] addresses the complexity of decision-making in such scenarios, emphasizing the need for both proactive optimization decisions and adaptive strategies tailored to specific discrete scenarios. This approach involves a two-stage stochastic mixed-integer programming framework, offering an emergency response pre-evacuation strategy applicable to earthquakes and other disasters.

| Symbols | Definition |
|---|---|
| $V$ | the set of nodes |
| $A$ | the set of links |
| $i, j$ | the index of nodes, $i, j \in V$ |
| $(i, j)$ | the index of directed links, $(i, j) \in A$ |
| $s$ | the index of scenario |
| $S$ | the total number of scenarios |
| $v$ | the supply value of source node |
| $\widetilde{T}$ | the time threshold |
| $T$ | the total number of time intervals |
| $u_{ij}$ | the capacity on physical link $(i, j)$ |
| $u_{ij}^s(t)$ | the capacity of link $(i, j)$ in scenario $s$ at time $t$ |
| $c_{ij}^s(t)$ | the travel time of link $(i, j)$ in scenario $s$ at time $t$ |
| $\mu_s$ | the probability in scenario $s$ |
| $x_{ij}$ | the flow on link $(i, j)$ |
| $y_{ij}^s(t)$ | the flow on link $(i, j)$ in scenario $s$ at time t |
| $d_i$ | the flow balance of node $i$ |
| $p_{ij}$ | the penalty on link $(i, j)$ |

**Table 3:** *Symbols explanation of evacuation planning in disaster problem*

The two-stage stochastic programming model with recourse depicts a scenario where both the initial and subsequent stages unfold at distinct time intervals within the same evacuation network. During the first stage, the probabilistic knowledge pertains only to link travel times and capacities. However, it is imperative to evacuate affected individuals from source nodes to other nodes before ascertaining the actual link travel times and capacities in the second stage. In the second stage, the evacuation plan is formulated based on the realized values of link travel times and capacities. It is noteworthy that the first-stage evacuation plan may prove infeasible for the given realization, a challenge addressed by allowing shorter evacuation times in the second stage. Consequently, the objective function encompasses both the first-stage penalty costs and the expected value of the second-stage recourse costs.

The construction of the two-stage stochastic evacuation model involves the introduction of two distinct types of decision variables. In the initial stage, a decision variable $x_{ij}$ is employed to indicate the flow on link $(i, j)$. Additionally, the second decision variable $y_{ij}^s(t)$ is defined to represent the flow on link $(i, j)$ in scenario $s$ at time $t$.

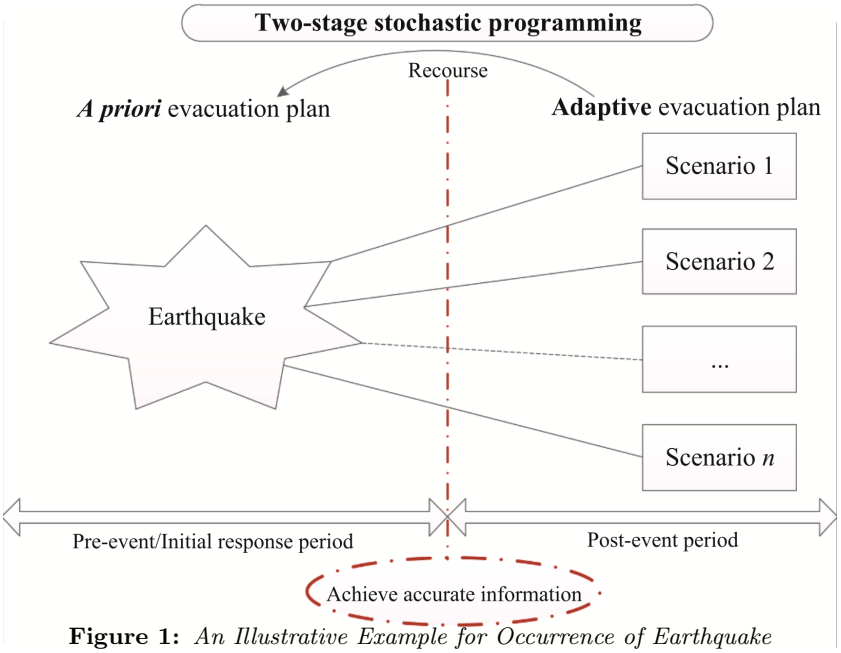



**Figure 1:** *An Illustrative Example for Occurrence of Earthquake*

$$\begin{cases} \min \Sigma_{i,j\in A}\ p_{ij}x_{ij} + \Sigma_{s=1}^{S}(\mu_S \cdot \Sigma_{(i,j)\in A_s} c_{ij}^S(t) \cdot y_{ij}^S(t)) \\ s.t. \\ \Sigma_{(i,j)\in A}\ x_{ij} - \Sigma_{(j,i)\in A}\ x_{ji} = d_i, \forall i \in V \\ 0 \le x_{ij} \le u_{ij}, \forall (i,j) \in A \\ \Sigma_{(i_t,j_{t'})\in A_S}\ y_{ij}^S(t) - \Sigma_{(j_{t'},i_t)\in A_S}\ y_{ij}^S(t') = d_i^S(t), \forall i \in V, \\ t \in \{0,1,...,T\}, s = 1,2,...,S \\ 0 \le y_{ij}^S(t) \le u_{ij}^S(t), \forall (i,j) \in A, t \in \{0,1,...,T\}, s = 1,2,...,S \\ y_{ij}^S(t) = x_{ij}, (i,j) \in A, s = 1,2,...,S \end{cases} \tag{3}$$

Model 3 is an integer programming model including two types of decision variable, i.e., $X := \{x_{ij}\}_{i,j\in A}$ and $Y := \{y_{ij}^S(t)\}_{i,j\in A, t\in\{0,1,...,T\}, S=1,2,...,S}$ are the first-stage variables and second-stage variables, respectively.

***Objective function:*** $\Sigma_{i,j\in A}\ p_{ij}x_{ij}$ is the penalty travel time to avoid potential loops in the network, and $\Sigma_{s=1}^{S}(\mu_S \cdot \Sigma_{(i,j)\in A_s} c_{ij}^S(t) \cdot y_{ij}^S(t))$ is the total travel time. Therefore, if $Y$ can be found such that the sum of these two function is at minimum, the optimal evacuation plan can be found.

***Constraints:***

- $\Sigma_{(i,j)\in A}\ x_{ij} - \Sigma_{(j,i)\in A}\ x_{ji} = d_i, \forall i \in V$: retains the flow balance of the node, calculated by the flow out of node minus the flow move into node. In this problem, the flow balance $d_{source} = v$ (supply value) since there is no flow in the source node, while the node sink has $d_{sink} = -v$ because the destination node does not have flow out. For the nodes between source and sink, the flow balance always $d = 0$

$$d = \begin{cases} v, & i = source \\ -v, & i = sink \\ 0, & \text{otherwise} \end{cases}$$

- $0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A$: the flow on link $(i,j)$ must not be greater than the capacity of that link.
- $\Sigma_{(i_t,j_{t'}) \in A_S}\ y_{ij}^S(t) - \Sigma_{(j_{t'},i_t) \in A_S}\ y_{ij}^S(t') = d_i^S(t), \forall i \in V,$: ensures the flow balance of every node $i \in V$ in the time-dependent network, i.e if some flows travel to node $i \in V/\{s,t\}$ in any period of time, they must travel out of that node, the flows travel out of node $s$ must equal to the supply value, and the flows travel into node $t$ must equal to supply value.
- $0 \leq y_{ij}^S(t) \leq u_{ij}^S(t), \forall (i,j) \in A, t \in \{0,1,...,T\}, s = 1,2,...,S$: ensures that the flow on link $(i,j)$ is not greater than the capacity at any time.
- $y_{ij}^S(t) = x_{ij}, (i,j) \in A, s = 1,2,...,S$: is a hard constraint. It requires that before time threshold , the evacuation plan is identical to the first-stage plan. Consequently, a Lagrangian multiplier is introduced for this coupling constraint. The main idea is bring the hard constraint into the objective function of the Lagrangian relaxation. This relaxation solution then provides the upper bound for the original minimization problem. The hard constraint can be incorporated into the objective function in the following manner:

$$\Sigma_{S=1}^S \Sigma_{t \leq \widetilde{T}} \Sigma_{(i,j) \in A}\ \alpha_{ij}^S(t)(y_{ij}^S(t) - x_{ij}) \tag{4}$$

After Lagrangian relaxation to the objective function, the relaxed model is:

$$\begin{cases} \min\ \Sigma_{i,j \in A}\ p_{ij}x_{ij} + \Sigma_{s=1}^S(\mu_S \cdot \Sigma_{i,j \in A_s} c_{ij}^S(t) \cdot y_{ij}^S(t)) + \\ \Sigma_{S=1}^S \Sigma_{t \leq \widetilde{T}} \Sigma_{(i,j) \in A}\ \alpha_{ij}^S(t)(y_{ij}^S(t) - x_{ij}) \\ s.t. \\ \Sigma_{(i,j) \in A}\ x_{ij} - \Sigma_{(j,i) \in A}\ x_{ij} = d_i, \forall i \in V \\ 0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A \\ \Sigma_{(i_t,j_{t'}) \in A_S}\ y_{ij}^S(t) - \Sigma_{(j_{t'},i_t) \in A_S}\ y_{ij}^S(t') = d_i^S(t), \forall i \in V, \\ t \in \{0,1,...,T\}, s = 1,2,...,S \\ 0 \leq y_{ij}^S(t) \leq u_{ij}^S(t), \forall (i,j) \in A, t \in \{0,1,...,T\}, s = 1,2,...,S \end{cases} \tag{5}$$

It's important to highlight that in the aforementioned relaxed model 5 , the variables $X$ and $Y$ can be disentangled. The relaxed model is broken down into two distinct sub-problems to help separate time-independent variables $X$ and time-dependent variables $Y$. The two sub-problems are min-cost flow problems.

**SubProblem 1: Min-cost Flow Problem**

$$\begin{cases} \text{minSP1}(\alpha) = \Sigma_{i,j \in A}\ (p_{ij} - \Sigma_{S=1}^S \Sigma_{t \leq \widetilde{T}}\ \alpha_{ij}^S(t))x_{ij} \\ s.t. \\ \Sigma_{(i,j) \in A}\ x_{ij} - \Sigma_{(j,i) \in A}\ x_{ij} = d_i, \forall i \in V \\ 0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A \end{cases} \tag{6}$$

We denote the generalized cost of each link by $g_{ij} := p_{ij} - \Sigma_{S=1}^S \Sigma_{t \leq \widetilde{T}}\ \alpha_{ij}^S(t)$. For convenience, the optimal objective value of sub-problem 1 is abbreviated as $Z_{SP1}^*(\alpha)$.

Subsequently, the successive shortest path algorithm is used to solve sub-problem 1. The goal of this algorithm is to calculate the minimum cost flow in the network $G = (V, A, C, U, D)$.

**Sub-problem 2: Time-Dependent Min-cost Flow Problem**

The second component of the relaxed model 5 pertains to the decision variables $Y$. The optimal objective value for this specific problem is denoted as $Z^*_{SP2}(\alpha)$, and it is expressed as follows:

$$\begin{cases} s\text{minSP2}(\alpha) = \Sigma_{s=1}^{S}\Sigma_{(i,j)\in A}(\Sigma_{t\in\{0,1,...,T\}}\mu_S \cdot c_{ij}^S(t) + \Sigma_{t\leq\widetilde{T}}\ \alpha_{ij}^S(t))y_{ij}^S(t) \\ s.t. \\ \Sigma_{(i_t,j_{t'})\in A_S}\ y_{ij}^S(t) - \Sigma_{(j_{t'},i_t)\in A_S}\ y_{ij}^S(t') = d_i^S(t), \forall i \in V, \\ t \in \{0,1,...,T\}, s = 1,2,...,S \\ 0 \leq y_{ij}^S(t) \leq u_{ij}^S(t), \forall (i,j) \in A, t \in \{0,1,...,T\}, s = 1,2,...,S \end{cases} \tag{7}$$

SubProblem 2 can be further decomposed into a total of S subProblems:

$$\begin{cases} \text{minSP2}(\alpha) = \Sigma_{(i,j)\in A}(\Sigma_{t\in\{0,1,...,T\}}\mu_S \cdot c_{ij}^S(t) + \Sigma_{t\leq\widetilde{T}}\ \alpha_{ij}^S(t))y_{ij}^S(t) \\ s.t. \\ \Sigma_{(i_t,j_{t'})\in A_S}\ y_{ij}^S(t) - \Sigma_{(j_{t'},i_t)\in A_S}\ y_{ij}^S(t') = d_i^S(t), \forall i \in V, \\ t \in \{0,1,...,T\}, s = 1,2,...,S \\ 0 \leq y_{ij}^S(t) \leq u_{ij}^S(t), \forall (i,j) \in A, t \in \{0,1,...,T\} \end{cases} \tag{8}$$

For each scenario $s \in \{1,2,...,S\}$, subProblem (5) has the similar structure with subProblem(3) with the time-dependent link cost $c_{ij}^S(t)$ and link capacity $u_{ij}^S(t)$ and the generalized cost $g_{ij}^S(t)$. This cost can be defined as piecewise function below by two time stages:

$$g_{ij}^S(t) = \begin{cases} \mu_S \cdot c_{ij}^S(t) + \alpha_{ij}^S(t), & \text{if } t \leq \widetilde{T} \\ \mu_S \cdot c_{ij}^S(t), & \text{if } \widetilde{T} \leq t \leq T \end{cases}$$

Since sub-problem 1 is a min-flow problem which does not depend on time, and thus the algorithm 1 should be modified in Step 2. The parameters $A(y(t)), C(y(t))$, and $U(y(t))$ in residual network $N(y(t))$ are defined:

$$A_s(y(t)) = (i_t, j_{t'})|(i_t, j_{t'}) \in A_s, y_{ij}^s < u_{ij}^s \cup (j_{t'}, i_t)|(j_{t'}, i_t) \in A_s, y_{ij}^s > 0$$
$$s = 1,2,...,S$$

$$c_{ij}^s(y(t)) = \begin{cases} c_{ij}^s(t), (i_t, j_{t'}) \in A_s, y_{ij}^s(t) < u_{ij}^s(t), t \in 0,1,...,T \\ -c_{ij}^s(t'), (j_{t'}, i_t) \in A_s, \forall t' \in 0,1,...,T | y_{ji}^s(t') > 0, \\ s = 1,2,...,S \\ T, (j_{t'}, i_t) \in A_s, \forall t' \in 0,1,...,T | y_{ji}^s(t') > 0 \end{cases}$$

$$u_{ij}^s(y(t)) = \begin{cases} u_{ij}^s(t) - y_{ij}^s(t), (i_t, j_{t'}) \in A_s, y_{ij}^s(t) < u_{ij}^s(t), t \in 0,1,...,T \\ y_{ji}^s(t), (j_{t'}, i_t) \in A_s, \forall t' \in 0,1,...,T | y_{ji}^s(t') > 0, \\ s = 1,2,...,S \\ 0, (j_{t'}, i_t) \in A_s, \forall t' \in 0,1,...,T | y_{ji}^s(t') > 0 \end{cases}$$

## 4.2 Algorithm 1: Successive shortest path algorithm for min-cost flow problem

### 4.2.1 Solve sub-problem 1 using the Algorithm 1

To implement successive shortest path algorithm, label-correct algorithm is needed to calculate the shortest paths with the maximum flow. Pseudo codes of algorithm 1 and FIFO label-correct algorithm:

**Algorithm 1** Successive shortest path

Step 1: Take variable $x$ as a feasible flow between any OD and it has the minimum delivery cost in the feasible flows.

Step 2: Terminate if any flow value of $x$ reaches supply value $v$ or there is no minimum cost path in the residual graph $(V, A(x), C(x), U(x), D)$. Otherwise, calculate the shortest path using algorithm 2 and go to step 3.

- $A(x) = \{(i,j)|(i,j) \in A(x)\} \cup \{(j,i)|(j,i) \in A, x_{ij} > 0$
- $C(x) = \begin{cases} c_{ij}, (ij) \in A, x_{ij} < u_{ij} \\ -c_{ji}, (ji) \in A, x_{ji} > 0 \end{cases}$
- $U(ij) = \begin{cases} u_{ij} - x_{ij}, (i,j) \in A, x_{ij} < u_{ij} \\ x_j i, (j,i) \in A, x_{ji} > 0 \end{cases}$

Step 3: Increase the flow by the amount equal to the capacity of the edge with smallest capacity in the min-cost path. If the increased flow value does not exceed supply value $v$, go to Step 2.

**Algorithm 2** FIFO label-correct algorithm

**Input:** Number of nodes, matrix of cost $(c_{ij})$ and matrix of capacity $(u_{ij})$ of graph $G = (V, E)$

**Output:** A shortest path from s to t, smallest capacity MINCAP among edges in the shortest path, total cost UPPER of the shortest path

(Initialization) Set $d_s = 0$, $d_i = \infty$ for all $i \in V \backslash \{s, t\}$.
(Initialization) Set UPPER $=\infty$, MINCAP $= \infty$ and QUEUE $= \{s\}$.
**while** QUEUE is not empty **do**
  Remove node $i$ at the beginning of QUEUE
  **for** each child j of i **do**
    **if** $d_i + a_{ij} < \min(d_j, \text{UPPER})$ **then**
      Set $d_j = d_i + c_{ij}$ and set $i$ to be parent of $j$
      **if** $j \neq t$ **then**
        Place $j$ in QUEUE if $j$ has never been in it
      **end if**
    **else**
      set UPPER $= d_i + c_{i,t}$
    **end if**
  **end for**
**end while**
**for** each node $j$ with parent $i$ in the shortest path **do**
  MINCAP = min(MINCAP,$u_{ij}$)
**end for**

***Data Simulation for Sub-Problem 1:***

For sub-problem 1, we need to simulate a grid network for the min-cost flow problem. The generalized cost of each link for this grid network is $g_{ij} = p_{ij} - \Sigma_{S=1}^{S} \Sigma_{t \leq \widetilde{T}} \alpha_{ij}^{S}(t)$. Table 5 describes how $g_{ij}$ of **each link** is generated. A loop is used to simulate $g_{ij}$ of every link in the grid network.

| Variable | Description | Value or range |
|---|---|---|
| pen | Penalty on link $(i, j)$ | $30 \leq$ pen $\leq 100$ |
| alpha | $\Sigma_{s=1}^{S} \Sigma_{t \leq \widetilde{T}} \alpha_{ij}^{S}(t)$ on link $(i, j)$ | $1 \leq$ alpha $\leq 10$ |
| cost | Generalized cost $g_{ij}$ on link $(i, j)$ | cost = pen - alpha |

**Table 4:** *Data simulation in sub-problem 1 of problem 2*

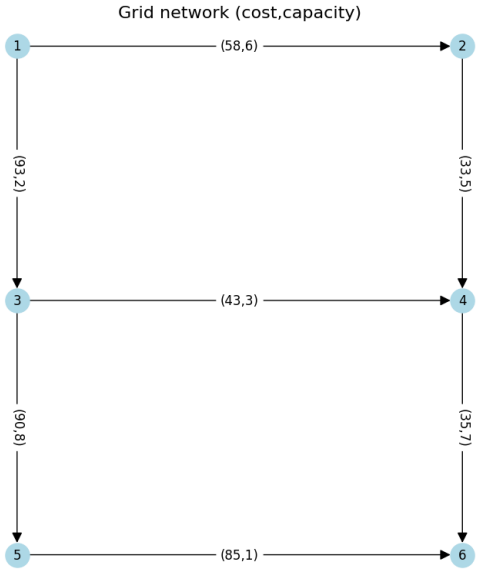
- pen: Since this is the penalty to prevent potential loops, it should be a large value. In ref [2], this value is 2 minutes (unit of time is second).
- alpha: We have difficulties simulating each $\alpha_{ij}^{s}(t)$. Fortunately, this project ignore the update of each $alpha_{ij}^{s}(t)$. Therefore, we choose to simulate the summation instead. The simulation of this summation is re-used in sub-problem 2.

***Example of Min-cost Flow problem:*** *source code*

### a. Input section

```
Enter the number of rows:      3
Enter the number of columns:   2
Enter the number of node:      6
Enter supply value:            10
```

The output is the grid of road network with 3 rows, 2 columns, 6 nodes

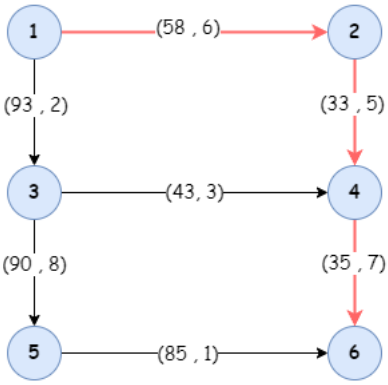



```
Cost Matrix:
    [[ 0 58 93  0  0  0]
     [ 0  0  0 33  0  0]
     [ 0  0  0 43 90  0]
     [ 0  0  0  0  0 35]
     [ 0  0  0  0  0 85]
     [ 0  0  0  0  0  0]]
Capacity Matrix:
    [[0 6 2 0 0 0]
     [0 0 0 5 0 0]
     [0 0 0 3 8 0]
     [0 0 0 0 0 7]
     [0 0 0 0 0 1]
     [0 0 0 0 0 0]]
```

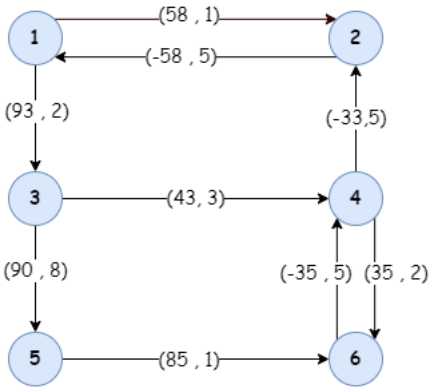
### b. Successive shortest path algorithm

- Iteration 1:

```
shortestPath : 1  2  4  6
smallestCapacity = 5
upper = 630
d = 5
```



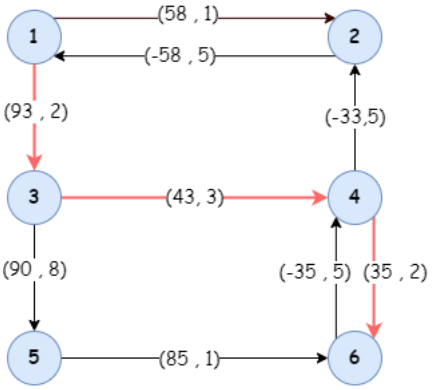

*Shortest path from Iteration 1*



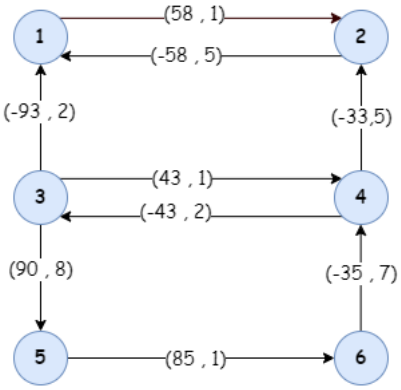

*Residual graph from Iteration 1*

- Iteration 2:

```
shortestPath : 1  3  4  6
smallestCapacity = 2
upper = 342
d = 3
```




*Shortest path from Iteration 2*



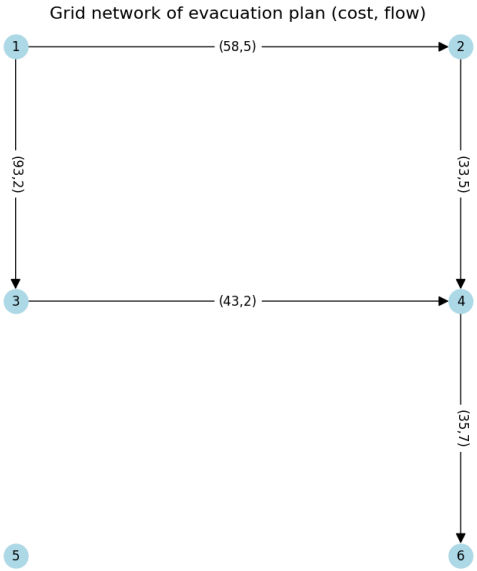

*Residual graph from Iteration 2*

- Iteration 3:

```
shortestPath : 1
smallestCapacity = inf
There is no minimum cost path in the residual network
upper = inf
d = 3
```

The algorithm terminates at iteration 4 because of no minimum cost path in the residual network and remain 4 flows have not transferred.

c. **Output section**




The algorithm returns the result of the maximum flow in the given network, and the total travel time needed to move the number of supply value form the source to the sink.

```
supply value = 10
maxflow = 7
minimum totalcost of maxflow = 972
minimum totalcost to move all = 1350
```

*The beside figure show the number of flows in each edge*

***Comments on Model and Algorithms:*** In the model, there is constraint which ensures that the network successfully evacuates all affected people (supply value) from the disaster areas (source) to the safe areas (sink) in a single run of the model. However, in practice, the capacity of the roads may not allow that to happen. In our example above, the supply value is 10, but the max flow is only 7. Therefore, to move all of the affected people to the safe areas, we might run the model some additional times.

### 4.2.2 Solve sub-problem 2 using modified Algorithm 1

In this sub-problem we will using the Algorithm 1 with some changes in the label-correcting section which can make it able to find the time-dependent min-cost path in the residual graph [3](Ziliaskopoulos & Mahmassani, 1992). By using this modified Algorithm 1, we can solve the problem with time-dependent attribute.

***Data Simulation for Sub-Problem 2:***

For sub-problem 2, we need to simulate the adapted plan after a time threshold $\widetilde{T}$. Table 5 describes how capacity $u_{ij}^s(t)$; cost $c_{ij}^s(t)$ after a random time threshold $\widetilde{T}$ is generated. A loop is used to simulate $u_{ij}^s(t)$ and $c_{ij}^s(t)$ of every link in the grid network.

| Variable | Description | Value or range |
|---|---|---|
| $\widetilde{T}$ | Time threshold | $1 \leq \widetilde{T} \leq$ totalcost |
| $\mu_s$ | the probability in scenario $s$ | $1 \leq \mu_s \leq 10$ |
| $u_{ij}^s(t)$ | new capacity on link $(i,j)$ after $\widetilde{T}$ | $u_{ij}^s(t) =$ capacity $-$ randint |
| $c_{ij}^s(t)$ | new cost on link $(i,j)$ after $\widetilde{T}$ | $u_{ij}^s(t) =$ cost $+$ randint |

**Table 5:** *Data simulation in sub-problem 2 of problem 2*

- $\widetilde{T}$ : since this is a time that changes may happen in the process of evacuating, it should be greater than 0 and no bigger than the maximum time *totalcost* according to the original plan.

- $u_{ij}^s(t)$: since this is an evacuation plan for a disaster, The likelihood of increasing the capacity occurring is very low. Hence, we set it equal to the original capacity minus a random positive integer ($1 \leq \text{randint} \leq 10$).
- $c_{ij}^s(t)$: Similar to the capacity, the change is going in a worse way. Thus, we set it equal to the original cost plus a random positive integer ($1 \leq \text{randint} \leq 10$).

***Solving sub-problem 2:***

In order to make the best plan for disaster evacuation when it has uncertainty changing at a time threshold $\widetilde{T}$, we need to know the realization at $\widetilde{T}$. Hence, keep tracking the flow of affected people and the status of infrastructures (capacity $u_{ij}^s(t)$ and time travel $c_{ij}^s(t)$) at each time $t$ is necessary.

Step 1: `Process the evacuation as in the priori plan.`
Since the flow of people evacuating is continuous, there almost immediately another flow go through the link or node after a flow is go out of that link or node. Thus, we need to store the flow of people at each link or node at time $0 \leqslant t \leqslant T$ and the time they are at that link or node. And so, when reaching the time threshold $\widetilde{T}$ we can know the places that affected people are at (how many people are at which link, which node?).

Step 2: `Create super source`
Create a super source and move all affected people to that super source. Then add dummy arcs from super source to each places (node only) with corresponding time travel and capacity. In case it is a link $(i, j)$, then the dummy arcs will go to the node $j$

- People at node: $\begin{cases} u_{0j}^s(t) = \text{number of people at that node} \\ c_{0j}^s(t) = 0 \end{cases}$
- People at link: $\begin{cases} u_{0j}^s(t) = \text{number of people at that link} \\ c_{0j}^s(t) = t_{remain} \end{cases}$

With $t_{remain}$ is the time travel remain need to go to the next node. For people at link: $0 < t_{remain} \leqslant c_{0j}^s(t)$; for people at node: $t_{remain} = 0$

The figure below is an example of this step:

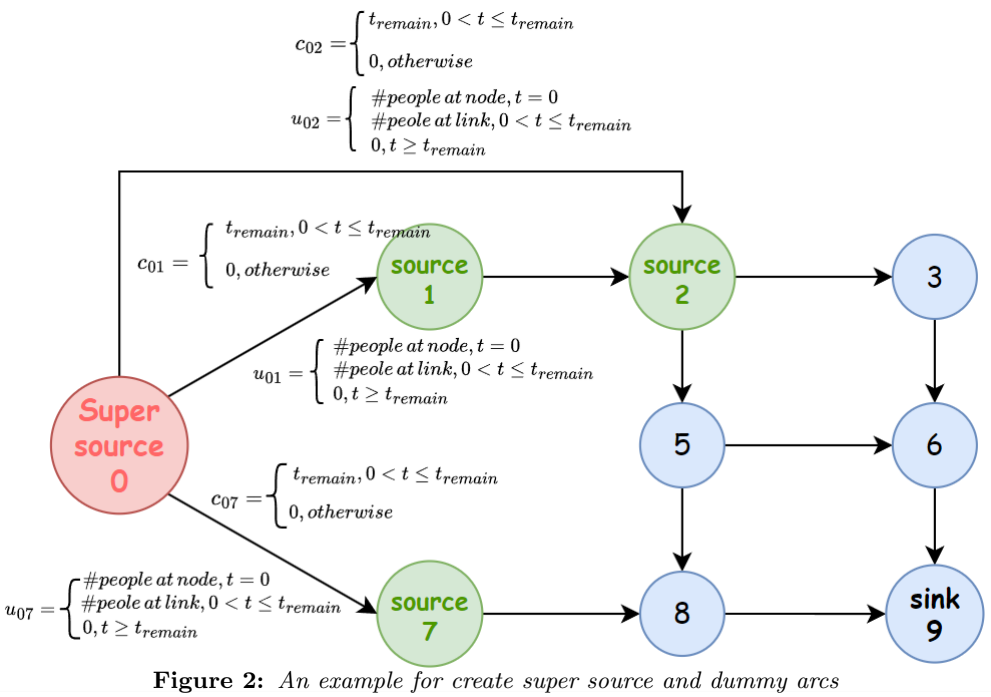



**Figure 2:** *An example for create super source and dummy arcs*

Step 3: `Generate new data and find the new plan`
After having a new graph with super source, we generate new data of time travel and capacity:

- $u_{ij}^s(t)$ = capacity − positive random integer
- $c_{ij}^s(t)$ = cost + positive random integer

Then using the modified Algorithm 1 to detect the best evacuation plan base on the new graph.

# 5 Source Code

The source codes are:
Problem 1: source code
Problem 2: source code

# 6 Conclusion

In summary, base on the concept of stochastic programming, we could obtain the optimal solution for problem with random uncertainties using scenario analysis, which finds the overall solution by considering each scenarios' solution. However, there are still many unsolved parts in this project, particularly problem 2. In the problem 2, we simply generate variable randomly, and the algorithm 1 just provide the lower bound and upper bound to the optimal value of the original model due to the Lagrangian multipliers. Therefore, to obtain the final optimal solution, we need to use "*Lagrangian relaxation-based approach combining with sub-gradient algorithm.*" provided in [2]

# References

[1] G.B. Dantzig. *Linear Programming under Uncertainty*. INFORMS, 1955.

[2] Li Wang. A two-stage stochastic programming framework for evacuation planning in disaster responses. *Computers & Industrial Engineering*, 145:106458, 2020.

[3] A Ziliaskopoulos and HS Mahmassani. *Proc. 5th Advanced Technology Conference, Washington DC*. 1992.

[4] H.W. Hamacher S.A. Tijandra N. *Mathematical Modeling of Evacuation Problems: A state of Art*. Fraunhofer-Institut für Techno- und, 2001.

[5] D. Dentcheva A. Shapiro and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, 2009.

[6] Hwang J Blitzstein J.K. *Introduction to Probability*. CRC Press, 2014.