# Stochastic Optimization
# Fall 2010
# Coopr Tutorial Project

Hsin-Chan Huang

Kiel Martin

# Table of Contents

## Installing Coopr

We will provide some brief notes and tips to help users install and use Python Optimization Modeling Objects (Pyomo) and Python-based Stochastic Programming (PySP), capabilities of the Common Optimization Python Repository (Coopr). However, for in-depth instructions and tutorials, we refer all users to the Coopr homepage at:

https://software.sandia.gov/trac/coopr

Any user wanting to learn how to use Pyomo or PySP should spend significant time exploring the website and reading the documentation before even attempting to install Coopr. A 'Getting Started' guide provides several convenient ways to install Coopr. However, be careful to read the details as missing any one of the simple instructions provided will cause hours of headache. In particular, as mentioned on the Coopr website, make sure to add the directory of your Coopr installation to your path environment variable BEFORE the system Python path.

One additional note, we found that when installing using Windows 7, the user must open the command line to execute the install as an administrator by right clicking the command line icon and choosing 'run as admin'. This issue was previously unknown to us or the Coopr development team.

After successfully installing Coopr, the first place to go is the examples folder of the installation (C:\coopr\examples). Documentation on the Coopr website provides tutorials using several models in the example folder. To run, view, and edit Python scripts using Pyomo, users may download one of many available Integrated Development Environments (IDEs). We chose to use Eclipse and downloaded the plugin PyDev to enable Eclipse to recognize Python syntax. Also, we downloaded a few other Python packages, SciPy and NumPy, to help us with data generation and reporting statistics. Lastly, Pyomo provides compatibility with several optimization solvers. We used Gurobi and GLPK. Gurobi provides a free academic license with full capability and GLPK is free to download.

## Useful Sites:

| | |
|---|---|
| • Coopr | https://software.sandia.gov/trac/coopr |
| • Eclipse | https://www.eclipse.org |
| • PyDev | http://pydev.org |
| • SciPy/NumPy | http://www.scipy.org |
| • Gurobi | http://www.gurobi.com |
| • GLPK | http://www.gnu.org/software/glpk |

## Scripting Overview

One issue for us as we learned Pyomo and PySP was clearly understanding the scripting necessary to run the models. The best explanation is provided in the Coopr documentation, but we give a short summary. Pyomo is the python module that implements optimization modeling, similar to GAMS, AMPL, etc. After building a model in Python using Pyomo and the associated data (either in a separate data file or in the model itself), we can run the model in one of two ways. We can either run the executable 'pyomo' in command line or we can add additional Python scripting (using Pyomo) to the model itself to run in a Python shell. We primarily used the second method since it provides flexibility to execute more complicated tasks such as running several instances of a model from different datasets and then collecting statistics.

```
Command: pyomo model.py data.dat        (Pyomo method I)

Command: python model.py                (Pyomo method II)
```

PySP is a modeling and solver library for stochastic programming that is based on Pyomo.  Given a Pyomo model file, scenario data files, and a scenario tree structure file, users can use PySP to solve stochastic programming models.  Currently there are two main executables that implement different methods to solve stochastic programming models: 'runph' (Progressive Hedging) and 'runef' (Extended Form).  A significant benefit to using PySP is the ability to use the Progressive Hedging algorithm, 'runph', to more quickly solve large, multi-stage problems than by using a traditional method, 'runef', to solve large-scale deterministic formulations.  In this project, we mostly used 'runef' since our goal was to learn the modeling construct, not the solver algorithm.

```
  Command: runef --model-directory=models –instance-directory=scenariodata
(PySP)
```

All available Python executables contain a significant list of options, accessed by adding '--help' after the command, or available in the documentation on the Coopr website.

## Data Generation

A significant hurdle to implementing the stochastic models was generating the data files.  Each problem requires a given (sometimes large) number of scenarios.  Each scenario requires an individual data file with a realization of the associated stochastic data.  We wrote short Python scripts for each problem that would generate instances of the relevant stochastic data and append this to a template including all of the deterministic data.

## Open Questions & Desired Improvements

This section highlights some difficulties we encountered learning PySP and some desired features. Please note, as novice users, there may already exist solutions to the following issues.

### How do we easily fix first stage variable?

Using stochastic models, often we fix the first stage variable(s) to understand the objective value associated with a given scenario or to understand the performance of a given solution. We did not find a simple way to fix variables and had to create additional constraints. Often, this resulted in model infeasibility due to rounding issues. In GAMS, we would use 'varName.fx' to fix a given variable.

### How do we create conditional sets based on data from *.dat files?

As novice users, we do not know if there are simple ways to manipulate data over conditional sets as in other math programming software such as GAMS. One example of this issue in our report is problem 3-7 in which we implemented the problem over a set containing all possible machine combinations instead of the conditional set of all feasible machine combinations. This caused increased runtimes due to the inefficient set definition.

### Organize PySP output

The output from Pyomo is simple to manipulate. All data fields are easily accessible in the Python data structure hierarchy. However, we do not know how to control the output when running the stochastic optimization routines. For some problems, we had to search through the output files to find the desired results. The user should be able to manipulate the results from PySP in a similar manner as Pyomo.

### Simplify 'ScenarioStructure' file format

It seemed to us that the scenario structure file format could be simplified. When creating this file we had to duplicate the data multiple times in the file, which was cumbersome for large problems. While the flexibility is a nice feature, most problems with common tree structures might benefit from a simplified, shorter format.

# Problem Set Solution Guide

## PS1-7 Coldville

### Description

(Schrage) The city of Coldville has the opportunity to save money by purchasing raw materials (salt and truck fuel) for snow removal prior to the onset of winter. There are two snow removal methods: plowing and salting; the former consumes truck fuel while the latter consumes both truck fuel and salt. Salting is generally more effective than plowing, particularly during warm winters. However, the salvage value at the end of the winter is quite low for excess salt. In contrast, truck fuel may be consumed by other city departments and there is less penalty for stocking too much fuel.

The final column of Table 1 indicates that the cost of operating a truck for one day is $110 in a warm winter and $120 in a cold winter; these figures are in addition to salt and fuel costs. The city's truck fleet can work 5000 truck-days during winter. Salting is more effective than plowing. During a warm winter, one truck salting is equivalent to 1.2 trucks plowing; in a cold winter the efficiency figure is 1.1. If only plowing were used to remove snow, a warm winter would require 3500 truck-days and a cold winter 5100 truck-days.

Coldville's expert weather forecasting team has predicted that the winter will be warm with probability 0:4 and cold with probability 0:6.

| | Salt | Fuel | Cost per truck-day |
|---|---|---|---|
| Summer | 20 | 70 | |
| Warm Winter | 30 | 73 | 110 |
| Cold Winter | 32 | 73 | 120 |
| Salvage Value | 15 | 65 | |

Table 1: Costs and salvage values in summer and winter. All values are in units of dollars per truck-day.

### PS1-7a

Formulate a two-stage stochastic linear program with recourse that models the city's planning problem. Clearly indicate the model's data, random variables, and decision variables as well as the timing of decisions, observations, and recourse decisions.

The following notation describes our formulation:

#### *Sets*

| | |
|---|---|
| I | set $i \in I$ of raw materials ($i$=1 salt, $i$=2 fuel) |
| J | set $j \in J$ of removal methods ($j$=1 salting, $j$=2 plowing) |
| $\Omega$ | set $\omega \in \Omega$ of weather scenarios ($\omega$=1 warm, $\omega$=2 cold) |

#### *Data*

| | |
|---|---|
| $p_k$ | probability of given weather scenario (winter temperature) |
| $c_i$ | cost of raw material $i$ in summer per truck-day |

| | |
|---|---|
| $g_i$ | salvage value of raw material *i* per truck-day |
| $b$ | truck-day capacity |

### *Random Variables*

| | |
|---|---|
| $d_i$ | cost of raw material *i* in winter per truck-day |
| $f$ | cost of operating per truck-day |
| $a_j$ | truck-day efficiency of salting or plowing |
| $m$ | truck-day demand |

### *Decision Variables*

| | |
|---|---|
| $x_i$ | amount of raw material *i* purchased in summer |
| $y_i^\omega$ | amount of raw material *i* purchased in winter for weather scenario $\omega$ |
| $t_j^\omega$ | truck-days assigned to removal method *j* in winter for weather scenario $\omega$ |
| $s_i^\omega$ | amount of raw material *i* salvaged after winter for weather scenario $\omega$ |

### *Formulation*

$$\min_{x,y,s,t} \quad \sum_I c_i x_i + \sum_\Omega p_\omega \left( \sum_I \left( d_i^\omega y_i^\omega - g_i s_i^\omega \right) + f^\omega \sum_J t_j^\omega \right) \qquad (1)$$

$$\text{s.t.} \quad \sum_J t_j^\omega \le b \qquad \forall \omega \in \Omega \qquad (2)$$

$$\sum_J a_j^\omega t_j^\omega \ge m^\omega \qquad \forall \omega \in \Omega \qquad (3)$$

$$t_1^\omega + s_1^\omega = x_1 + y_1^\omega \qquad \forall \omega \in \Omega \qquad (4)$$

$$t_1^\omega + t_2^\omega + s_2^\omega = x_2 + y_2^\omega \qquad \forall \omega \in \Omega \qquad (5)$$

$$\{x, y, s, t\} \ge 0$$

The objective minimizes the total cost as the sum of the summer cost and the expected winter cost over all weather scenarios. Constraint (2) enforces capacity. Constraint (3) ensures demand is met. Constraints (4) and (5) ensure the supply of resources is maintained. All variables are non-negative.

### PS1-7b

Implement and solve your model in PySP. Report the solution.

Use runef script to solve the problem. The deterministic model, scenario structure and scenario data are contained in the folder “PS1/1-7b”, in which P(warm winter)=0.4 and P(cold winter)=0.8. To run, use the following command in the appropriate folder.

```
runef –m models –i scenariodata --solver=gurobi –solve>PartBresult.txt
```

The optimal value is Z*=819888.333 and the optimal solution as follows.

| | | | | | |
|---|---|---|---|---|---|
| first stage | D.V. | x_salt | | x_fuel | |
| | opt. sol. | 2916.667 | | 2916.667 | |
| second stage | D.V. | y_salt_warm | y_fuel_warm | y_salt_cold | y_fuel_cold |
| | opt. sol. | 0 | 0 | 0 | 1891.667 |
| | D.V. | t_salting_warm | t_plowing_warm | t_salting_cold | t_plowing_cold |
| | opt. sol. | 2916.667 | 0 | 2916.667 | 1891.667 |
| | D.V. | t_salt_warm | s_fuel_warm | s_salt_cold | s_fuel_cold |
| | opt. sol. | 0 | 0 | 0 | 0 |

## PS1-7c

Resolve the problem assuming, in turn, that the winter will be cold and assuming that the winter will be warm.

Run the Python script to solve the problem. The deterministic model and warm winter scenario data can be found in the folder “PS1/1-7c”. In the Python script (i.e. the model file), notice import opt package and the use of the function “opt.solve()” to solve. The optimal value for the warm scenario is Z*=583333.333 and the optimal solution as follows.

| | | | | | |
|---|---|---|---|---|---|
| first stage | D.V. | x_salt | | x_fuel | |
| | opt. sol. | 2916.667 | | 2916.667 | |
| second stage | D.V. | y_salt_warm | y_fuel_warm | y_salt_cold | y_fuel_cold |
| | opt. sol. | 0 | 0 | 0 | 0 |
| | D.V. | t_salting_warm | t_plowing_warm | t_salting_cold | t_plowing_cold |
| | opt. sol. | 2916.667 | 0 | 0 | 0 |
| | D.V. | t_salt_warm | s_fuel_warm | s_salt_cold | s_fuel_cold |
| | opt. sol. | 0 | 0 | 0 | 0 |

Similar to 7-c1, the deterministic model and cold winter scenario data can be found in the folder “PS1/1-7c”. The optimal value for the cold scenario is Z*=970000 and the optimal solution is as follows.

| first stage | D.V. | x_salt | | x_fuel | |
|---|---|---|---|---|---|
| | opt. sol. | 1000 | | 5000 | |
| second stage | D.V. | y_salt_warm | y_fuel_warm | y_salt_cold | y_fuel_cold |
| | opt. sol. | 0 | 0 | 0 | 0 |
| | D.V. | t_salting_warm | t_plowing_warm | t_salting_cold | t_plowing_cold |
| | opt. sol. | 0 | 0 | 1000 | 4000 |
| | D.V. | t_salt_warm | s_fuel_warm | s_salt_cold | s_fuel_cold |
| | opt. sol. | 0 | 0 | 0 | 0 |

## PS1-7d

Give a verbal interpretation of the solution to the stochastic program [part (b)]. How much of each raw material is bought when? Why? What are the snow-removal policies under the two winter scenarios? Use the solutions to the “known future" problems [part (c)] to help interpret the stochastic solution.

Without knowledge of the weather condition in winter we buy both salt and fuel with equal amount of 2916.667 in summer. Then, we additionally buy 1891.667 fuel in winter to face cold winter. That is exactly the amount used by plowing in cold weather. So, with warm weather we only use salt and with cold weather we use salt and buy additional fuel to plow. If we know the future, we buy exactly what is needed for the given weather in the summer.

## PS1-7e

Similar to PS1-7b, use a runef script to solve the problem. The deterministic model, scenario structure and scenario data in the folder “PS1/1-7e”, in which P(warm winter)=0.3 and P(cold winter)=0.7. To run, use the following command in the appropriate folder.

```
runef –m models –i scenariodata --solver=gurobi –solve>PartEresult.txt
```

The optimal value Z*=858179.167 and the optimal solution is as follows.

| first stage | D.V. | x_salt | | x_fuel | |
|---|---|---|---|---|---|
| | opt. sol. | 2916.667 | | 4808.333 | |
| second stage | D.V. | y_salt_warm | y_fuel_warm | y_salt_cold | y_fuel_cold |
| | opt. sol. | 0 | 0 | 0 | 0 |
| | D.V. | t_salting_warm | t_plowing_warm | t_salting_cold | t_plowing_cold |
| | opt. sol. | 2916.667 | 0 | 2916.667 | 1891.667 |
| | D.V. | t_salt_warm | s_fuel_warm | s_salt_cold | s_fuel_cold |
| | opt. sol. | 0 | 1891.667 | 0 | 0 |

## PS1-7f

Under the original probability distribution, how much (in expected value) do the citizens of Coldville save by purchasing salt and fuel in the summer as opposed to just waiting until winter arrives and then acting optimally?

Again, use a Python script to solve the problem. The associated files can be found in the folder "PS1/1-7f". In the deterministic model, notice the constraint that the first stage variable x is fixed to zero. The optimal value Z*= 621250 for warm winter and 997000 for cold winter. The citizens can save is 26811.667 (= 0.4*(621250-819888.333) + 0.6*(997000-819888.333)).

## PS2-6 Aircraft Allocation

### Description

(Dantzig and Ferguson) Consider the following instance of the aircraft allocation model:

|J| = number of plane types = 4; |R| = number of routes = 5

b = [10; 19; 25; 15] available aircraft

q = [13; 13; 7; 7; 1] bump cost

| Type / route | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| J1 | 18 /16 | 21 / 15 | 18 / 28 | 16 / 23 | 10 / 81 |
| J2 | NA | 15 / 10 | 16 / 14 | 14 / 15 | 9 / 57 |
| J3 | NA | 10 / 5 | NA | 9 / 7 | 6 / 29 |
| J4 | 17 / 9 | 16 / 11 | 17 / 22 | 15 / 17 | 10 / 55 |

Table 2: $c_{jr}$ / $k_{jr}$ for each route-type combination. Certain aircraft cannot y certain routes (NA).

| $d_r$ | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| $[\ell_r, u_r]$ | [200,300] | [50,150] | [140,220] | [10,340] | [580,620] |

Table 3: Ranges for the random demand on each route.

### PS2-6a

Implement and solve the aircraft allocation model as formulated in the notes with each $d_r$ distributed as a discrete uniform random variable with 100 realizations on the ranges shown in Table 3.

The following notation describes our formulation:

#### *Sets*

| | |
|---|---|
| $J$ | set $j \in J$ aircraft types |
| $R$ | set $r \in R$ routes |
| $\Omega_r$ | set $\omega_r \in \Omega_r$ demand scenarios for each route |

#### *Data*

| | |
|---|---|
| $b_j$ | number of aircraft of type $j$ |
| $q_r$ | unit revenue lost by turning away a route $r$ passenger |
| $c_{jr}$ | cost of operating aircraft of type $j$ on route $r$ |
| $k_{jr}$ | passenger capacity of aircraft type $j$ on route $r$ |

#### *Random Variables*

| | |
|---|---|
| $d_r$ | customer demand on route $r$ |
| $d_r^{\omega_r}$ | realization of $d_r$ |

$p_r^{\omega_r} = P(d_r = d_r^{\omega_r})$ [Assume discrete random variables]

### *Decision Variables*

| | |
|---|---|
| $x_{jr}$ | number of planes of type $j$ assigned to route $r$ |
| $y_r^{\omega_r}$ | number of unserved customers (unsatisfied demand) on route $r$ under scenario $\omega_r$ |

### *Formulation*

$$\min_{x,y} \quad \sum_J \sum_R c_{jr} x_{jr} + \sum_R \sum_{\Omega_r} [p_r^{\omega_r} q_r] y_r^{\omega_r} \qquad (6)$$

$$\sum_R x_{jr} \le b_j \qquad \forall j \in J \qquad (7)$$

$$\sum_R k_{jr} x_{jr} + y_r^{\omega_r} \ge d_r^{\omega_r} \qquad \forall r \in R, \omega_r \in \Omega_r \qquad (8)$$

$$\{x, y\} \in \mathbb{R}^+$$

The objective (6) minimizes the cost of operating and the expected cost due to unsatisfied demand over all scenarios. Constraint (7) limits number of aircraft based on the number available. Constraint (8) defines unsatisfied demand as nonnegative difference between demand and available capacity.

Run a runef script to solve the problem. The deterministic model, scenario structure and scenario data is in the folder “PS2/2-6a”. To run, use the following command in the appropriate folder.

```
runef –m models –i scenariodata --solver=gurobi –solve>PartAresult.txt
```

The optimal value is 1831.11 and the solution is as follows.

| $X_{RP}(r,j)$ | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| J1 | 10 | 0 | 0 | 0 | 0 |
| J2 | 0 | 8.15 | 0 | 0 | 0 |
| J3 | 0 | 4.8 | 0 | 10.85 | 20.2 |
| J4 | 7.091 | 0 | 7.909 | 0 | 0 |

## PS2-6b

Repeat part (a) with your improved model.

### *Sets*

| | |
|---|---|
| $J$ | set $j \in J$ aircraft types |
| $R$ | set $r \in R$ routes |
| $\Omega_r$ | set $\omega_r \in \Omega_r$ demand scenarios for each route |

### Data

| | |
|---|---|
| $b_j$ | number of aircraft of type $j$ |
| $q_r$ | unit revenue lost by turning away a route $r$ passenger |
| $c_{jr}$ | cost of operating aircraft of type $j$ on route $r$ |
| $k_{jr}$ | passenger capacity of aircraft type $j$ on route $r$ |

### Random Variables

| | |
|---|---|
| $d_r$ | customer demand on route $r$ |
| $d_r^{\omega_r}$ | realization of $d_r$; assume $d_r^{\omega_r}$ ordered such that $d_r^1 \le d_r^2 \le \ldots$ |

$p_r^{\omega_r} = P(d_r = d_r^{\omega_r})$ [Assume discrete random variables]

$\Delta d_r^{\omega_r} = d_r^{\omega_r} - d_r^{\omega_r - 1}$

### Decision Variables

| | |
|---|---|
| $x_{jr}$ | number of planes of type $j$ assigned to route $r$ |
| $z_r^{\omega_r}$ | capacity installed that is assigned to demand $\Delta d_r^{\omega_r}$ |

### Formulation

$$\min_{x,z} \quad \sum_J \sum_R c_{jr} x_{jr} + \sum_R \sum_{\Omega_r} q_r p_r^{\omega_r} [d_r^{\omega_r} - \sum_{i=1}^{\omega_r} z_r^{\omega_r}] \tag{9}$$

$$\sum_R x_{jr} \le b_j \qquad \forall j \in J \tag{10}$$

$$\sum_R k_{jr} x_{jr} \ge \sum_\Omega z_r^{\omega_r} \qquad \forall r \in R \tag{11}$$

$$0 \le z_r^{\omega_r} \le \Delta d_r^{\omega_r} \qquad \forall r \in R, \omega \in \Omega \tag{12}$$

$$x \in \mathbb{R}^+$$

Run a Python script in a Python shell to solve the problem. The associated deterministic model and scenario data can be found in the folder "PS2/2-6b". Results are identical to (part a).

### PS2-6c

Using the distributions from (a), solve the expected-value model (random demands set at their population means). Calculate the performance of the expected-value aircraft allocation decision in the stochastic model.

Run a Python script in a Python shell to solve the problem. The associated deterministic model and scenario data can be found in the folder "PS2/2-6c1". The optimal value is 1241.33 and the optimal solution is as follows.

| $X_{EV}(r,j)$ | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| J1 | 10 | 0 | 0 | 0 | 0 |
| J2 | 0 | 2.333 | 5 | 11.667 | 0 |
| J3 | 0 | 15.333 | 0 | 0 | 9.667 |
| J4 | 10 | 0 | 5 | 0 | 0 |

Run a runef script to solve the problem. The deterministic model, scenario structure and scenario data can be found in the folder "PS2/2-6c2". In the deterministic model, a constraint is added to fix the first stage variable X to $X_{EV}$. To run, use the following command in the appropriate folder.

```
runef –m models –i scenariodata --solver=gurobi –solve>PartCresult.txt
```

The optimal value is 1934.99 when fixing $X$ at $X_{EV}$ in the stochastic model.

## PS2-6d

Solve the model when the demand random variables are continuous uniform random variables on the ranges shown in Table 2.

The associated deterministic model and scenario data can be found in the folder "PS2/2-6d". Run a Python script to solve the problem using a nonlinear solver. The optimal value is 1824.70.

## PS2-6e

Resolve your improved model (part b) with integer aircraft allocation decision variables.

Run a Python script in a Python shell to solve the problem. The associated deterministic model and scenario data can be found in the folder "PS2/2-6e". The model sets the first stage variables as integers. The optimal value is 1831.97 and the optimal solution is as follows.

| $X(r,j)$ | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| J1 | 10 | 0 | 0 | 0 | 0 |
| J2 | 0 | 8 | 0 | 11 | 0 |
| J3 | 0 | 5 | 0 | 0 | 20 |
| J4 | 7 | 0 | 8 | 0 | 0 |

## PS2-7 Semiconductor Manufacturing

### Description

(Stafford) A semiconductor manufacturer is planning to purchase a number of wafer-fabrication machines of different types, $m$, in the face of uncertain demands for wafers of various types, $w$. In practice, production of a wafer requires routing it through a network of fabrication machines for multiple production steps, $s$. However, because we are building a strategic capacity expansion model we will instead take a simpler and more aggregate view of operations and just ensure that the time required to perform the assigned production steps does not exceed machine time. In order to count a wafer as being produced we must ensure that each of its production steps has been done. The budget allows $B$ machines to be purchased. Each machine will have available production time of $T_m$. The machines must be purchased prior to realizing the demand for each wafer type, $d_w$, and prior to realizing the time to execute a step of type $s$ in the production of wafer $w$ on machine $m$. Assignments of wafer production steps to machines are made after these random variables are known. The goal is to minimize the expected value of a weighted sum of unmet demand, where wafer type $w$ receives weight $\rho_w$.

Consider the following model instance in which there are 27 machine types (only a subset of which will be purchased), 10 wafer types, and up to 7 production steps for each wafer. For all wafer types $w$ the demands are independent and take values in $\{50+25\ell, \ell=1,\ldots 5\}$ with respective marginal probabilities, 0.1, 0.1, 0.3, 0.4, 0.1 The mean production times, $\mathbb{E}\tilde{t}_{swm}$, are listed in the following table. The production times are independent binary random variables that take values 1.25 $\mathbb{E}\tilde{t}_{swm}$ and 0.75 $\mathbb{E}\tilde{t}_{swm}$ with equal probability. Assume that we currently have one machine of each of the 27 types and we are in a position to purchase 6 additional machines. Let the wafer types have equal weights, $\rho_w$ = 1, and assume $T_m$ = 800 minutes for all $m$.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| w1 | s1 | m1 | 5 | w1 | s1 | m2 | 7 | w1 | s1 | m27 | 4 | w1 | s2 | m3 | 3 |
| w1 | s2 | m4 | 2 | w1 | s2 | m5 | 2 | w1 | s3 | m6 | 5 | w1 | s3 | m7 | 4 |
| w1 | s4 | m8 | 4 | w1 | s4 | m9 | 4 | w1 | s4 | m10 | 4 | w2 | s1 | m1 | 4 |
| w2 | s1 | m2 | 6 | w2 | s2 | m3 | 3 | w2 | s2 | m4 | 2 | w2 | s2 | m5 | 2 |
| w2 | s3 | m11 | 3 | w2 | s3 | m12 | 3 | w2 | s4 | m13 | 6 | w2 | s4 | m14 | 7 |
| w3 | s1 | m15 | 5 | w3 | s1 | m16 | 6 | w3 | s2 | m17 | 4 | w3 | s2 | m18 | 3 |
| w3 | s3 | m6 | 5 | w3 | s3 | m7 | 5 | w3 | s4 | m8 | 3 | w3 | s4 | m10 | 5 |
| w4 | s1 | m1 | 7 | w4 | s1 | m15 | 5 | w4 | s1 | m27 | 4 | w4 | s2 | m3 | 3 |
| w4 | s2 | m18 | 4 | w4 | s3 | m11 | 5 | w4 | s3 | m12 | 3 | w4 | s4 | m13 | 6 |
| w4 | s4 | m14 | 8 | w5 | s1 | m2 | 8 | w5 | s1 | m16 | 7 | w5 | s2 | m3 | 3 |
| w5 | s2 | m4 | 2 | w5 | s2 | m5 | 3 | w5 | s3 | m6 | 5 | w5 | s3 | m7 | 6 |
| w5 | s4 | m8 | 8 | w5 | s4 | m14 | 9 | w6 | s1 | m2 | 4 | w6 | s1 | m15 | 5 |
| w6 | s1 | m16 | 7 | w6 | s2 | m4 | 3 | w6 | s2 | m5 | 2 | w6 | s2 | m17 | 4 |
| w6 | s3 | m19 | 10 | w6 | s3 | m20 | 11 | w6 | s4 | m21 | 3 | w6 | s4 | m22 | 4 |
| w7 | s1 | m15 | 3 | w7 | s1 | m16 | 2 | w7 | s2 | m3 | 5 | w7 | s2 | m4 | 4 |
| w7 | s2 | m5 | 5 | w7 | s3 | m23 | 2 | w7 | s3 | m24 | 2 | w7 | s4 | m21 | 3 |
| w7 | s4 | m22 | 4 | w7 | s5 | m1 | 6 | w7 | s5 | m2 | 5 | w7 | s6 | m17 | 3 |
| w7 | s6 | m18 | 4 | w7 | s7 | m8 | 6 | w7 | s7 | m10 | 5 | w8 | s1 | m1 | 3 |
| w8 | s1 | m2 | 3 | w8 | s1 | m15 | 3 | w8 | s1 | m16 | 3 | w8 | s2 | m23 | 6 |
| w8 | s2 | m24 | 10 | w8 | s3 | m11 | 3 | w8 | s3 | m12 | 7 | w8 | s4 | m8 | 8 |
| w8 | s4 | m9 | 4 | w8 | s4 | m10 | 3 | w8 | s5 | m25 | 6 | w8 | s5 | m26 | 4 |
| w8 | s6 | m21 | 3 | w8 | s6 | m22 | 5 | w8 | s7 | m13 | 5 | w8 | s7 | m14 | 7 |
| w9 | s1 | m1 | 5 | w9 | s1 | m15 | 6 | w9 | s2 | m3 | 3 | w9 | s2 | m17 | 3 |
| w9 | s3 | m6 | 6 | w9 | s3 | m11 | 4 | w9 | s4 | m25 | 7 | w9 | s4 | m26 | 5 |
| w9 | s5 | m3 | 4 | w9 | s5 | m5 | 3 | w9 | s6 | m8 | 4 | w9 | s6 | m14 | 6 |
| w9 | s7 | m21 | 4 | w9 | s7 | m22 | 5 | w10 | s1 | m2 | 9 | w10 | s1 | m16 | 8 |
| w10 | s2 | m4 | 2 | w10 | s2 | m18 | 3 | w10 | s3 | m7 | 5 | w10 | s4 | m11 | 3 |
| w10 | s4 | m12 | 4 | w10 | s5 | m4 | 2 | w10 | s5 | m5 | 4 | w10 | s6 | m25 | 7 |
| w10 | s6 | m26 | 4 | w10 | s7 | m8 | 8 | w10 | s7 | m9 | 7 | w10 | s7 | m10 | 6 |

Table 4: Mean production times, $\mathbb{E}\tilde{t}_{swm}$, on machine $m$ for step $s$ in the production of wafer $w$. The first three entries of the table reads as follows: Step $s1$ in the production of wafer $w1$ can be done on either machine $m1$, $m2$, or $m27$ (the second-stage assignment decision will decide which). The mean production times on these three machines are, respectively, 5, 7, and 4 minutes. In addition, it can be seen that a wafer of type $w1$ requires a total of four steps to complete production.

## PS2-7a

Formulate a general instance of this problem as a two-stage stochastic integer program.

The following notation describes our formulation:

### *Sets*

| | |
|---|---|
| $M$ | set $m \in M$ machines |
| $W$ | set $w \in W$ wafers |
| $S$ | set $s \in S$ steps |
| $\Omega$ | set $\omega \in \Omega$ demand scenarios |

### *Data*

| | |
|---|---|
| $B$ | total number of machines to be purchased |
| $T_m$ | available production time of machine $m$ |
| $\rho_w$ | weight for wafer $w$ |
| $cap_m$ | current number of machines $m$ |

### *Random Variables*

| | |
|---|---|
| $\tilde{t}_{swm}$ | time of step $s$ to produce wafer $w$ on machine $m$ |
| $d_w$ | demand of wafer $w$ |

$\xi = \{d_w, \tilde{t}_{swm}\}$

### *Decision Variables*

| | |
|---|---|
| $x_m$ | number of machines $m$ bought |
| $y_{swm}^{\omega}$ | number of steps $s$ to produce wafer $w$ on machine $m$ for scenario $\omega$ |
| $z_w^{\omega}$ | unsatisfied demand for wafer $w$, scenario $\omega$ |
| $f_w^{\omega}$ | number of wafers $w$ finished for scenario $\omega$ |

### *Formulation*

$$\min_x \quad \mathbb{E}h(x,\xi) \tag{13}$$

$$\sum_m x_m \le B \tag{14}$$

$$x \in \mathbb{Z}^+$$

$$h(x,\xi) = \min_{f,y,t} \sum_{W} \rho_w \sum_{\Omega} p^{\omega} z_w^{\omega} \tag{15}$$

$$d_w^{\omega} - f_w^{\omega} \le z_w^{\omega} \qquad \forall w \in W, \omega \in \Omega \tag{16}$$

$$\sum_{m} y_{wsm}^{\omega} = f_w^{\omega} \qquad \forall s \in S, w \in W, \omega \in \Omega \tag{17}$$

$$\sum_{W} \sum_{S} y_{swm}^{\omega} t_{swm}^{\omega} \le T_m (cap_m + x_m) \qquad \forall m \in M, \omega \in \Omega \tag{18}$$

$$y \in \mathbb{Z}^+, z \in \mathbb{R}^+$$

The objective (15) minimizes the weighted, expected unsatisfied demand over all scenarios. Constraint (14) limits number of additional machines purchased. Constraint (16) defines unsatisfied demand as nonnegative difference between wafer demand and wafers finished. Constraint (17) defines the finished number of wafers as the sum of all wafers produced on all machines. Constraint (18) ensures the number of wafers produced is less than the available machine time.

## PS2-7b

Implement and solve the linear-programming relaxation of the above instance. Because of the large number of realizations it is impossible to solve this instance exactly. So, draw 100 iid observations of the random parameters and solve that 100-scenario linear program instead.

Run a runef script to solve the problem. The deterministic model, scenario structure and scenario data can be found in the folder "PS2/2-7b". Using our formulation, if we use 100 scenarios the run time is excessive. Therefore, we use only 50 scenarios. To run, use the following command in the appropriate folder.

```
runef –m models –i scenariodata –scenario-tree-downsample-fraction=0.5 --
solver=gurobi --solve>PartBresult.txt
```

Results vary according to the random sample drawn. One such optimal value is 132.2280 and the optimal solution is as follows.

| $X_{RP}$ | |
|---|---|
| M1 | 0.598 |
| M2 | 0.642 |
| M6 | 0.254 |
| M8 | 0.900 |
| M10 | 0.089 |
| M11 | 0.112 |
| M13 | 0.937 |
| M14 | 0.578 |
| M15 | 0.396 |

| | |
|---|---|
| M16 | 0.617 |
| M21 | 0.078 |
| M26 | 0.078 |

## PS2-7c

What is your estimate of the value of the stochastic solution for this model?

First, run a Python script in a Python shell to solve the Expected Value Problem (EV). The associated deterministic model and scenario data can be found in the folder “PS2/2-7c”. The optimal value is 111.418 and the following optimal solution $X_{EV}$ is as follows.

| $X_{EV}$ | |
|---|---|
| M1 | 0.491 |
| M2 | 0.816 |
| M7 | 0.451 |
| M8 | 0.595 |
| M10 | 0.21 |
| M11 | 0.234 |
| M13 | 1.816 |
| M16 | 0.701 |
| M21 | 0.362 |
| M27 | 0.325 |

Second, run a runef script twice to solve the problem, one to estimate the performance of $X_{EV}$ and the other for $X_{RP}$. The deterministic model, scenario structure, and scenario data can be found in the same folder “PS2/2-7c”. In the deterministic model, an added constraint fixes the first stage variable X to either $X_{EV}$ or $X_{RP}$. To run, use the following command in the appropriate folder.

```
runef –m models –i scenariodata --solver=gurobi --solve>PartCresult.txt
```

In the presented solution, only 50 scenarios are used to evaluate the Value of the Stochastic Solution (VSS). to get the optimal values. The optimal value is 144.5337 for $X_{EV}$ and 122.3095 for $X_{RP}$. So, the estimate VSS is 22.2242 (=144.5337-122.3095).

## PS3-7 Vehicle Routing

### Description

(Birge and Donohue) Vehicles initially positioned at nodes 1-5 must traverse the directed network shown in Figure 1 from left-to-right, ending at node 20. A revenue, cost and demand are associated with each link in the network and these are shown in Table 6. Each vehicle can service one unit of demand per arc so that if two vehicles traverse arc (3, 9) when the demand is three then a revenue of 2•255 is collected. (The "demands" are requests for service, i.e., they need not be satisfied.) If four vehicles traverse (3, 9) when the demand is three, the profit for that arc is 3•255 - 73. Table 5indicates that the random demand on arc (3, 9) can take value 1, 2 or 3. Vehicles must be positioned at the five left-most nodes in the network, prior to the demand being realized. After the demand is realized vehicles are routed through the network so as to maximize profit (revenues less costs). Assume the demands on the 46 arcs are independent and are governed by the distributions specified in Table 5. The goal is to position a fleet of 51 identical vehicles so that expected profit is maximized.

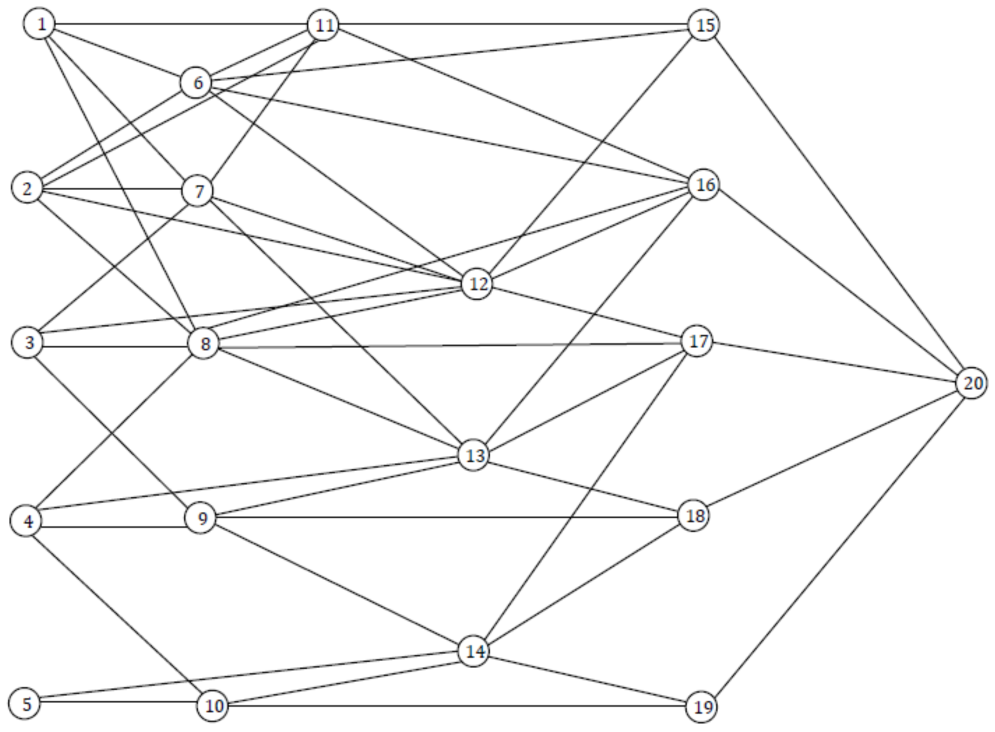

**Figure 1: Transportation network for vehicle allocation problem. In this directed network, flow is left-to-right. Node 20 is an artificial sink node.**

| demand type | $P(\tilde{d}_i = 0)$ | $P(\tilde{d}_i = 1)$ | $P(\tilde{d}_i = 2)$ | $P(\tilde{d}_i = 3)$ | $P(\tilde{d}_i = 4)$ |
|---|---|---|---|---|---|
| 1 | 0.40 | 0.30 | 0.30 | | |
| 2 | 0.40 | 0.25 | 0.20 | 0.15 | |
| 3 | 0.35 | 0.25 | 0.20 | 0.20 | |
| 4 | 0.30 | 0.25 | 0.20 | 0.15 | 0.10 |
| 5 | 0.35 | 0.20 | 0.15 | 0.15 | 0.15 |
| 6 | | 0.45 | 0.35 | 0.20 | |
| 7 | | 0.40 | 0.35 | 0.25 | |
| 8 | | 0.45 | 0.20 | 0.20 | 0.15 |
| 9 | | 0.40 | 0.25 | 0.20 | 0.15 |
| 10 | | | 0.40 | 0.30 | 0.30 |
| 11 | 1.00 | | | | |

Table 5: The mass functions governing the different types of demand distributions.

| arc $(i, j)$ | arc number | revenue | cost | demand type |
|---|---|---|---|---|
| 1.11 | 1 | 327 | 81 | 4 |
| 1. 6 | 2 | 211 | 67 | 10 |
| 1. 7 | 3 | 243 | 70 | 1 |
| 1. 8 | 4 | 268 | 70 | 7 |
| 2.11 | 5 | 301 | 75 | 10 |
| 2. 6 | 6 | 204 | 73 | 3 |
| 2.12 | 7 | 356 | 95 | 8 |
| 2. 7 | 8 | 190 | 63 | 5 |
| 2. 8 | 9 | 225 | 69 | 1 |
| 3. 7 | 10 | 198 | 66 | 1 |
| 3.12 | 11 | 343 | 85 | 2 |
| 3. 8 | 12 | 261 | 80 | 3 |
| 3. 9 | 13 | 255 | 73 | 7 |
| 4. 8 | 14 | 265 | 76 | 3 |
| 4.13 | 15 | 360 | 105 | 9 |
| 4. 9 | 16 | 161 | 60 | 10 |
| 4.10 | 17 | 264 | 78 | 8 |
| 5.14 | 18 | 348 | 86 | 10 |
| 5.10 | 19 | 267 | 78 | 8 |
| 6.11 | 20 | 146 | 54 | 7 |
| 6.15 | 21 | 330 | 86 | 4 |
| 6.16 | 22 | 352 | 93 | 2 |
| 6.12 | 23 | 188 | 61 | 6 |
| 7.11 | 24 | 225 | 74 | 3 |
| 7.12 | 25 | 177 | 63 | 9 |
| 7.13 | 26 | 348 | 91 | 10 |
| 8.12 | 27 | 148 | 51 | 10 |
| 8.16 | 28 | 312 | 76 | 8 |
| 8.17 | 29 | 332 | 83 | 9 |
| 8.13 | 30 | 284 | 75 | 3 |
| 9.13 | 31 | 317 | 76 | 6 |
| 9.18 | 32 | 367 | 103 | 6 |
| 9.14 | 33 | 329 | 81 | 2 |
| 10.14 | 34 | 214 | 65 | 9 |
| 10.19 | 35 | 326 | 78 | 1 |
| 11.15 | 36 | 263 | 93 | 4 |
| 11.16 | 37 | 285 | 98 | 7 |
| 12.15 | 38 | 281 | 99 | 7 |
| 12.16 | 39 | 242 | 87 | 1 |
| 12.17 | 40 | 301 | 108 | 6 |
| 13.16 | 41 | 277 | 93 | 4 |
| 13.17 | 42 | 168 | 77 | 5 |
| 13.18 | 43 | 175 | 78 | 7 |
| 14.17 | 44 | 273 | 94 | 8 |
| 14.18 | 45 | 252 | 87 | 3 |
| 14.19 | 46 | 252 | 88 | 2 |
| 15.20 | 47 | 0 | 0 | 11 |
| 16.20 | 48 | 0 | 0 | 11 |
| 17.20 | 49 | 0 | 0 | 11 |
| 18.20 | 50 | 0 | 0 | 11 |
| 19.20 | 51 | 0 | 0 | 11 |

Table 6: Data for vehicle allocation network. Each arc has a revenue, cost and random demand. The mass functions governing the different demand distribution (types) are given in Table 5.

## PS3-7a

Formulate this vehicle allocation problem as a two-stage stochastic linear program. You may neglect the integral nature of the vehicles.

### *Sets*

| | |
|---|---|
| E | set $e \in E$ of edges in network. $(i,j) = e \in E$ |
| S | set $s \in S$ of source nodes $s = 1,2,\ldots,5$ |
| D | set $d \in D$ of demand nodes $d = 6,7,\ldots,19$ |
| Ω | set $\omega \in \Omega$ of scenarios |

### *Data*

| | |
|---|---|
| $c_e$ | cost on edge $(i,j) = e \in E$ |
| $r_e$ | revenue on edge $(i,j) = e \in E$ |
| $b$ | total number of vehicles |

### *Random Variables*

| | |
|---|---|
| $d_e$ | demand of edge $(i,j) = e \in E$ |

### *Decision Variables*

| | |
|---|---|
| $x_s$ | first stage number of vehicles at source node s |
| $y_e^\omega$ | second stage routing for scenario $\omega$ |
| $z_e^\omega$ | second stage overage for scenario $\omega$ |

### *Formulation*

$$\min \sum_{\Omega,E} \{p^\omega r_e y_e^\omega - (r_e + c_e) z_e^\omega\} \qquad (19)$$

s.t.

$$\sum_s x_s = b \qquad (20)$$

$$\sum_{j \in J,(i,j)\in E} y_{(i,j)}^\omega = x_i \qquad \forall i \in S, \omega \in \Omega \qquad (21)$$

$$\sum_{i \in I|(i,j)\in E} y_{(i,j)}^\omega = \sum_{i \in I|(j,i)\in E} y_{(j,i)}^\omega \qquad \forall j \in D, \omega \in \Omega \qquad (22)$$

$$y_e^\omega - d_e^\omega \le z_e^\omega \qquad \forall e \in E, \omega \in \Omega \qquad (23)$$

$$0 \le \{x, y, z\} \le b$$

The objective (19) is to maximize profit. Constraint (20) ensures all available vehicles are assigned. Constraint (21) enforces flow from the assigned vehicles at the source nodes. Constraint (22) ensures balance of flow at demand nodes. Constraint (23) defines the excess supply at each node. All variable are nonnegative and loosely bounded by the total number of available vehicles.

## PS3-7b

The problem instance described above has $|\Omega| > 3^{46} = 8.8 \bullet 10^{21}$ so we will not attempt to solve it directly. Use Jensen's inequality to derive a bound on the optimal expected profit (denote it $JB$).

Run a Python script to solve the problem. The associated files can be found in the folder "PS3/3-7b". The optimal value for the Jensen's Bound (JB) is 17923.45.

## PS3-7c

Due to the size of $|\Omega|$, the wait-and-see bound ($WS$) on the optimal expected profit cannot be computed exactly. So, instead compute a point estimate and an approximate 95% confidence interval on $WS$. Do so for a sample size of 500.

Run a Python script to solve the problem. The associated files can be found in the folder "PS3/3-7c". Using 20 samples, a point estimate of "wait and see" (WS) = 17872.85 and 95% CI = [16997.45 18748.24].

## PS3-7d

Relate $JB$ and $WS$, both theoretically and through your computations.

If the sample size is increased for the WS solution, then we could calculate a CI and say with a certain confidence that JB > WS. This matches the theoretical expectation since there is randomness only in $\tilde{d}$ (RHS). Thus, $EEV \le RP \le WS \le EV$ .

## PS3-7e

Use the allocation decision from part (b), $x_{EV}$, to form a sample mean estimate of $EEV$. State the point estimate and the associated approximate 95% confidence interval. Do so for a sample size of 500.

Run a Python script to solve the problem. The associated files can be found in the folder "PS3/3-7e". In the deterministic model, we added the constraint that the first stage variable X is fixed to X_EV. Using 20 samples, we obtained the point estimate of EEV = 16994.15 and 95% CI = [16167.27 17821.03]

## PS3-7f

Based on your calculations in part (c), form an allocation decision, $x_{WS}$. Form a point estimate and approximate 95% confidence interval for $EWS \equiv Eh(x_{WS}, \tilde{d})$. Do so for a sample size of 500.

Run a Python script to solve the problem. The associated files can be found in the folder "PS3/3-7f". In the deterministic model, the constraint is added fixing the first stage variable $x$ to $x_{WS}$ (from PS3-7c). Using 20 samples, a point estimate of EWS = 17726.5 and 95% CI = [16545.32 18907.68]

## PS3-7g

Approximately solve this problem a Monte Carlo sampling-based approach. What is your estimate of $x_{RP}$?

Run a runef script to solve the problem. The deterministic model, scenario structure and scenario data can be found in the folder "PS3/3-7g". To run, use the following command in the appropriate folder.

```
runef –m models –i scenariodata --solver=gurobi --solve>PartGresult.txt
```

However, if we use 500 scenarios, it would take a long time to solve. So, we use the following command to only take a fraction of the scenario tree.

```
runef –m models –i scenariodata --solver=gurobi--scenario-tree-downsample-
fraction=0.5 --solve>PartBresult.txt
```

Using 50 scenarios, the optimal value is 17121.8400 and optimal solution is as follows.

| $X_{RP}$ | |
|---|---|
| X1 | 11 |
| X2 | 13 |
| X3 | 8 |
| X4 | 12 |
| X5 | 7 |

## PS3-7h

Estimate the value of the stochastic solution for this model.

Run a runef script twice to solve the problem, one for estimate the performance of $x_{EV}$ and the other for $x_{RP}$. The deterministic model, scenario structure and scenario data can be found in the same folder "PS3/3-7h". In the deterministic model, a constraint is added fixing the first stage variable $x$ to $x_{EV}$ or $x_{RP}$. However, we did not have enough time to get solutions, although we are sure that the files work with small numbers of scenarios (~10).

## PS4-2 Telecommunications Network Planning

### Description

(Higle and Sen) Figure 2 depicts a small telecommunications planning network for high-capacity services such as televideo conference calls. We wish to "build" this network by installing capacity on each of the seven arcs. The cost of installing capacity on (A, E), (E, D), or (D, C) is 2.9 per unit while the cost of installing capacity on (A, B), (E, B), (D, B), or (C, B) is 1.0 per unit. We have a total budget of 20.0.

The capacity expansion decisions must be made in the face of uncertainty with respect to the future values of point-to-point demand pairs and arc reliabilities. Arcs (A, E), (E, D), and (D, C) are "100%" reliable while arcs (A, B), (E, B), (D, B), and (C, B) operate at full capacity with probability 0.7 and zero capacity with probability 0.3. The point-to-point demand distributions are specified in Table 7 and Table 8. The respective demands and capacity availabilities are modeled as independent random variables.

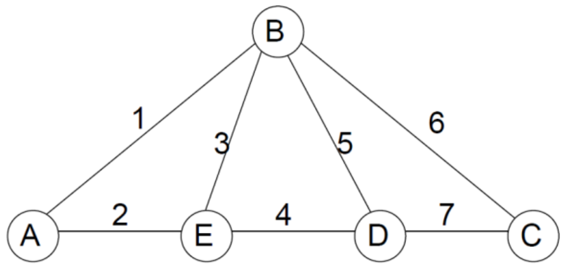

Figure 2: Telecommunications Planning Network

| | B | C | D | E |
|---|---|---|---|---|
| A | type 1 | type 2 | type 2 | type2 |
| B | | type 1 | type 1 | type 1 |
| C | | | type 2 | type 2 |
| D | | | | type 2 |

Table 7: Point-to-point demand distribution types

| Type 1 | | Type 2 | |
|---|---|---|---|
| $d^{\omega}$ | $P(\tilde{d} = d^{\omega})$ | $d^{\omega}$ | $P(\tilde{d} = d^{\omega})$ |
| 0 | 0.05 | 0 | 0.1 |
| 1 | 0.2 | 1 | 0.4 |
| 2 | 0.5 | 2 | 0.4 |
| 3 | 0.2 | 3 | 0.1 |
| 5 | 0.05 | | |

Table 8: Demand distribution

### PS4-2a

Formulate the telecommunications capacity expansion problem as a two-stage stochastic linear program with an objective that minimizes that expected number of blocked calls, i.e., minimizes unmet demand. Do this for a "generic" problem.

The following notation describes our formulation:

## Sets

| | |
|---|---|
| I | set $i \in I$ of demands |
| J | set $j \in J$ of demand levels |
| M | set $m \in M$ of demand types |
| $\Omega$ | set $\omega \in \Omega$ of scenarios |
| | |
| K | set $k \in K$ of arcs, *k* = 1…8 |
| P | set $p \in P$ of paths |

## Data

| | |
|---|---|
| $aa_{k,p}$ | arc path incidence matrix, where $aa_{k,p}$ = 1 if path *p* uses arc *k* |
| $dd_{i,p}$ | demand path incidence matrix, where $dd_{i,p}$=1 if path *p* can satisfy demand *i* |
| *b* | total budget |
| $c_k$ | unit installing cost for arc *k* |

## Random Variables

| | |
|---|---|
| $ppDemand_i$ | the point to point demand for *i* |
| $r_k$ | reliability of arc *k* |

## Decision Variables

| | |
|---|---|
| $x_k$ | first stage capacity allocation by arc *k* |
| $y_{i,p}^{\omega}$ | second stage assignment of demand to path *p* under scenario *u* |
| $z_i$ | second stage demand short |

## Formulation

$$\min E[h(x,\tilde{\xi})] \qquad (24)$$

s.t.
$$\sum_K c_k x_k \leq b \qquad (25)$$

$$x \in \mathbb{R}^+$$

$$h(x,\tilde{\xi}) = \min \sum_i z_i \qquad (26)$$

$$\sum_{\{p:dd(i,p)=1\}} y_{i,p} + z_i = ppDemand_i \qquad \forall i \in I \qquad (27)$$

$$\sum_{\{(i,p):aa_{k,p}=1,dd_{i,p}=1\}} y_{i,p} \le x_k r_k \qquad \forall k \in K \qquad (28)$$

$$\{y,z\} \in \mathbb{R}^+$$

The objective (24) minimizes the expected shortage over all scenarios. Constraint (25) enforces the budget. Constraint (27) ensures assigned demand plus shortage add up to point-to-point demand. Constraint (28) enforces capacity along each arc.

## PS4-2b

Calculate EV.

Run a Python script in a Python shell to solve the problem. The associated deterministic model and scenario data can be found in the folder "PS4/4-2b". The optimal value is 6.1 and the optimal solution is as follows.

| $x_{RP}$ | |
|---|---|
| *x(1)* | 7.07143 |
| *x(2)* | 0 |
| *x(3)* | 5.07143 |
| *x(4)* | 0 |
| *x(5)* | 2.92857 |
| *x(6)* | 4.92857 |
| *x(7)* | 0 |

## PS4-2c

We cannot calculate *WS* exactly because it would involve solving 4.1*$10^7$ variables. Instead, form a sample mean estimate of *WS* based on a sample size of 100. State the point estimate and the associated approximate 95% confidence interval.

Run a Python script to solve the problem. The associated files can be found in the folder "PS4/4-2c". Using 50 samples, a point estimate of *WS* = 6.0 and 95% CI = [5.11 6.89].

## PS4-2d

Use the capacity expansion decision from part (b), $x_{EV}$, to form a sample mean estimate of *EEV* with a sample size of 100. State the point estimate and the associated approximate 95% confidence interval.

Run a Python script to solve the problem. The associated files can be found in the folder "PS4/4-2d". In the deterministic model, a constraint is added fixing the first stage variable $x$ to $x_{EV}$. Using 50 samples, a point estimate of *EEV* = 8.3and 95% CI = [7.29 9.31].

## PS4-2e

Based on your calculations in part (c), form a capacity expansion decision, $x_{ws}$. Form a point estimate and approximate 95% confidence interval for $EWS = cx_{ws} + Eh(x_{ws}, \tilde{\xi})$. Again, use a sample size of 100.

Run a Python script to solve the problem. The associated files can be found in the folder "PS4/4-2e". In the deterministic model, a constraint is added fixing the first stage variable $x$ to $x_{ws}$ (which is from PS3-c). Using 50 samples, a point estimate of *EWS* = 7.3and 95% CI = [6.49 8.11].

## PS4-2f

Solve this problem using the Monte Carlo sampling-based approach for assessing solution quality with common random number streams for upper and lower bound estimation. Solve one initial approximating problem in order to generate the candidate solution $\hat{x}$. Use $n_g$ = 20 replications. Choose the sample size, $n$, so that you obtain what you regard as "reasonable" results. State your recommended capacity-expansion decision and a confidence interval for the quality of the proposed solution.

Run a computeconf script to solve the problem, which implements the Mak/Morton/Wood multiple replication procedure to calculate the approximate gap of the stochastic solution. The associated files can be found in the folder "PS4/4-2f". To run, use the following command in the appropriate folder.

```
computeconf–m models –i scenariodata --scenario-tree-downsample-fraction=0.2 --solver=gurobi --solve-ef>PartFresults.txt
```

## PS4-3 Forest Harvesting

### Description

(Gassmann) Fred owns a forest and has a contract to sell timber he harvests to a local paper company. The forest has trees that may be broken into four classes: Class 1 has trees 0-25 years old, Class 2 has trees 26-50 years old, Class 3 has trees 51-75 years old, and Class 4 has trees older than 75 years. The forest currently has 8,000 acres in Class 1, 10,000 acres in Class 2, 20,000 acres in Class 3, and 60,000 acres in Class 4. Each class of timber has a different yield: Class 1 has no yield, Class 2 yields 250 cubic feet per acre, Class 3 yields 510 cubic feet per acre, and Class 4 yields 710 cubic feet per acre. The "danger" in waiting to harvest trees is that fires or bug-infestations can destroy portions of the forest - effectively returning those stands to Class 1. Each 25-year period, it is equally likely that 5%, 10%, or 15% of the trees will be destroyed; these destruction events are independent between time periods but are common to the entire forest (i.e., all four classes) within a time period. Trees that aren't destroyed by fire or bugs, and aren't harvested, progress to the next class in the next time period (e.g., Class 2 trees become Class 3 trees) except that Class 4 trees remain in Class 4. If a class of trees is cut or destroyed then it transitions to Class 1 in the next stage. Fred's contract specifies that over the next 100 years the paper company will pay (in present-dollar terms so no discounting is needed): 10 cents per cubic foot up to 98 million cubic feet, 7 cents per cubic foot between 98-105 million cubic feet, and 5 cents per cubic foot in excess of 105 million cubic feet.

### PS4-3a

Formulate a four-stage stochastic program that will give a harvesting plan for the next 100 years that maximizes expected profit. (Assume that the forest has no value to Fred's family at the end of the 100-year planning period.)

Assume that we harvest in a very short time after the time points (0th, 25th, 50th, 75th, 100th year). Then the model is as follows:

#### *Sets*

| | |
|---|---|
| $\Omega$ | set $\omega \in \Omega$ of scenarios |
| T | set $t \in T$ of time points, $t$ = 1, 2, 3, 4, 5 |
| I | set $i \in I$ of asset classes, $i$ = 1, 2, 3, 4 |
| J | set $j \in J$ of contract pay levels |

#### *Data*

| | |
|---|---|
| $a_i$ | yields per acre for class *i* |
| $b_j$ | contract levels for each pay interval |
| $c_j$ | profit per unit at contract level *j* |
| $p_\omega$ | probability of scenario $\omega$ |

### *Random Variables*

| | |
|---|---|
| $d_t^\omega$ | percent of destruction during *t*-1 and *t* under scenario $\omega$ |
| $x_{i,t}^\omega$ | harvest amount of class *i* at time *t* under scenario $\omega$ |
| $y_{i,t}^\omega$ | available asset of class *i* at time *t* under scenario $\omega$ (i.e. amount before harvest) |
| $z_j^\omega$ | amount sold at contract level *j* under scenario $\omega$ |

### *Formulation*

$$\max \sum_{\Omega} p_\omega \sum_J c_j z_j^\omega \tag{29}$$

s.t.

$$y_{1,t+1}^\omega = \sum_I [(y_{i,t}^\omega - x_{i,t}^\omega)\cdot d_t^\omega + x_{i,t}^\omega] \qquad \forall u \in U \; t = 1,2,3,4 \tag{30}$$

$$y_{2,t+1}^\omega = (y_{1,t}^\omega - x_{1,t}^\omega)\cdot(1-d_{t+1}^\omega) \qquad \forall u \in U \; t = 1,2,3,4 \tag{31}$$

$$y_{3,t+1}^\omega = (y_{2,t}^\omega - x_{2,t}^\omega)\cdot(1-d_{t+1}^\omega) \qquad \forall u \in U \; t = 1,2,3,4 \tag{32}$$

$$y_{4,t+1}^\omega = (y_{3,t}^\omega + y_{4,t}^\omega - x_{3,t}^\omega - x_{4,t}^\omega)\cdot(1-d_{t+1}^\omega) \qquad \forall u \in U \; t = 1,2,3,4 \tag{33}$$

$$x_{i,t}^\omega \le y_{i,t}^\omega \qquad \forall i \in I, t \in T, \omega \in \Omega \tag{34}$$

$$\sum_I \sum_T x_{i,t}^\omega \cdot a_i = \sum_J z_j^\omega \qquad \forall \omega \in \Omega \tag{35}$$

$$0 \le z_j^\omega \le b_j \qquad \forall \omega \in \Omega \tag{36}$$

$$\{x, y\} \in \mathbb{R}^+$$

The objective (29) maximizes expected profit over all scenarios. Constraints (30) - (33) enforce the relationship between available assets and harvests for each time period. Constraint (34) ensures harvest is less than available assets. Constraints (35) and (36) implement the contract pricing levels.

## PS4-3b

Implement and solve your model in PYSP.

Run a Python script in a Python shell to solve the problem. The associated deterministic model and scenario data can be found in the folder "PS4/4-3b". Note that this problem is modeled as 5 stages stochastic optimization problem, so we need to specify every scenario to one of the five stages in "ScenarioStructure.dat". Run the Python script and we get the optimal value = 11366.7072.

## PS4-3c

Report and analyze your solution in a manner Fred can understand.

The harvest plan is as follows:

(1) At time point *t1* (0th year), harvest all of class 3 and 4 (i.e. 20000 and 60000 acres, respectively).

(2) At time point *t2* (25th year), harvest all of class 3 and the amount depends on the damage level of period 1 (*t1*-*t2*).

(3) At time point *t3* (50th year), harvest all of class 3 and the amount depends on the damage levels of period 1 (*t1*-*t2*) and 2 (*t2*-*t3*).

(4) At time point *t4* (75th year), no harvest.

(5) At time point *t5* (100th year), harvest all of class 2, 3, and 4 and the amounts vary with each scenario.