BANA4095: Decision Models – Spring 2021
Linear Optimization, Part 2 (Python)

Dr. Charles R. Sox
Associate Dean - Impact & Partnerships
Professor of Operations and Business Analytics

1

Outline

- Google Colab
- Optimization with Python
- Google OR Tools
- Python Objects
- Simple Sidneyville Example
- Python LP Code Summary

2

Google Colaboratory (Colab)

- Cloud environment for Jupyter Notebooks
- Runs on a remote server
- Requires a Google account
- Save and open files in Google Drive
- Convenient resource when you can't or don't want to run Jupyter on your local computer
- Access through a web browser – recommend Google Chrome

https://colab.research.google.com

3

Optimization with Python

- Different optimization modeling packages available
  - » Glop – free open source, Google OR Tools **we will use this one!**
  - » Gurobi – commercial, free student/academic license
  - » PuLP – fee open source, simple, LP and MILP
  - » Pyomo – free open source, more complex optimization problems
  - » Usually includes one or more solver algorithms
- Different optimization solver algorithms available
  - » Open Source: CBC, GLPK, Glop, SCIP
  - » Commercial: Gurobi, CPLEX, Xpress

4

Installing Google OR Tools

- Must have Python and an Internet connection
- Within a Jupyter Notebook, run the command
  `!pip install ortools`
- Or within a Command Prompt / Terminal Window
  - » Select Run as Administrator option
  - » `python -m pip install --upgrade --user ortools`
- Download may take a few minutes

5

Google OR Tools `pywraplp` Module

- GLOP: Google Linear Optimization Package
- Import `pywraplp` package
  `from ortools.linear_solver import pywraplp as glp`
- For documentation details use the commands
  `help()` and `dir()`

6

New Python Concept – Object

- An object is a collection of related data and code that is used to manipulate that data
- For example, a list is actually an object
- The coded functions associated with an object are called methods. We can use the `dir()` function to find the methods associated with an object. Object methods are invoked using 'dot notation'

```
mylist = ['Clifton', 'Calhoun', 'Ludlow']
dir(mylist)
mylist.append('Corry')
mylist.sort()
```

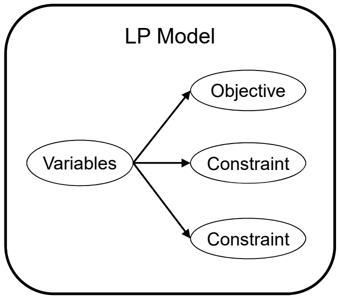
7

New Python Concept – Object

- Other objects we've already used: string, integer, float
- Objects can be nested, that is, an object can be composed of other objects with their own associated methods
- In Glop, an optimization model is a compound object that contains other objects:
  - » Variables
  - » Objective
  - » Constraints

8

Glop LP Optimization Objects



Example – Sidneyville Desk Mfg.

- Allocation/Product Mix Problem
- Produces two types of desk
- Using three types of wood in every desk
  (measured in board feet, b.f.)

Type	Profit/desk
Rolltop	$115
Regular	$90

Wood	Amount Used: Rolltop	Amount Used: Regular	Amount Available
Pine	10	20	200
Cedar	4	16	128
Maple	15	10	220

Sidneyville Linear Programming (LP) Formulation

$\max \quad 115x_1 + 90x_2$ — Maximize Total Profit

s.t. $10x_1 + 20x_2 \le 200$ — Pine

$4x_1 + 16x_2 \le 128$ — Cedar

$15x_1 + 10x_2 \le 220$ — Maple

$x_1, x_2 \ge 0$ — Non-negative

Glop Optimization Objects

Model Object

```
mymodel = glp.Solver('Sidneyville',glp.Solver.GLOP_LINEAR_PROGRAMMING)
```

Variable Objects

```
Roll_Top = mymodel.NumVar(0, mymodel.infinity(), 'Roll_Top Desks')
Regular = mymodel.NumVar(0, mymodel.infinity(), 'Regular Desks')
```

Objective Object

```
Profit = mymodel.Objective()
Profit.SetCoefficient(Roll_Top, 115)
Profit.SetCoefficient(Regular, 90)
Profit.SetMaximization()
```

Constraint Object

```
Pine = mymodel.Constraint(-mymodel.infinity(), 200)
Pine.SetCoefficient(Roll_Top, 10)
Pine.SetCoefficient(Regular, 20)
```

Creating an Optimization Model

- Import `pywraplp` package

```
from ortools.linear_solver import pywraplp as glp
```

- Create the optimization model object

```
mymodel = glp.Solver('Sidneyville', glp.Solver.GLOP_LINEAR_PROGRAMMING)
```

model constructor label model type

- Create the decision variables

```
Roll_Top = mymodel.NumVar(0, mymodel.infinity(), 'Roll_Top Desks')
```

variable lower bound upper bound label

Creating an Optimization Model

- Create the objective function

objective object

```
Profit = mymodel.Objective()
Profit.SetMaximization()
Profit.SetCoefficient(Roll_Top, 115)
```

Set optimization direction

variable coefficient

- Create the constraints

lower bound upper bound

constraint

```
Pine = mymodel.Constraint(-mymodel.infinity(), 200)
Pine.SetCoefficient(Roll_Top, 10)
```

variable coefficient

Displaying the Model

- Download `lptools.py` file from Canvas and place a copy in the local file folder folder
- Import the `lptools` module

```
import lptools as lpt
```

- Use the `lpt.print_model()`function to print a mathematical description of your model

Displaying a Solution

- Solves the model “mymodel” and returns its solution status

```
status = mymodel.Solve()
```

0	Optimal
1	Feasible
2	Infeasible
3	Unbounded
4	Abnormal
5	Not Solved

- Other useful methods

```
Profit.Value()
Roll_Top.solution_value()
Pine.dual_value()
mymodel.ComputeConstraintActivities()
```

- Display the solution
  - » Use `print()` function with formatting commands to display the optimal value, decision variables, and dual/shadow prices

Python Optimization Code Steps

- Import Glop package
- Input parameters
- Create model object
- Create variable objects
- Create objective function and coefficients
- Create constraints and coefficients
- Solve the problem
- Check solution status
- Display optimal value
- Display optimal solution
- Display constraint values

Important Glop Object Methods

- Model Object
  `.NumVar(), .Objective(), .Constraint(), .variables(), .constraints(), .ComputeConstraintActivities(), .Clear()`
- Objective Object
  `.SetMaximization(), .SetMinimization(), .SetCoefficient(), .Value()`
- Variable Objects
  `.name(), .lb(), .ub(), .solution_value(), .reduced_cost()`
- Constraint Objects
  `.SetCoefficient(), .name(), .lb(), .ub(), .dual_value()`

Simple Sidneyville LP Model

- Requires excessive lines of code
- Time consuming to construct and maintain
- Should use Python data structures and loops to generalize the code and make it more efficient to construct and maintain

Summary

- Google Colab
- Optimization with Python
- Google OR Tools
- Python Objects
- Simple Sidneyville Example
- Python LP Code Summary