15.094 Homework 1

Due: 03/03/2021

1 Robust Reformulation

Consider the following constraint on x, where z is an uncertain parameter:

$$(2+z)x \leq 1.$$

This constraint is equivalent to the following formulation, when there is no uncertainty in z.

$$(2+z)x + s = 1,$$

$$s \geq 0.$$

Now think of the robust counterparts of these two sets of constraints. The uncertainty set is the interval $\{z : |z| \leq 1\}$.

1.1

Find the feasible sets of the above formulations. Are they equal? What can you say about robust formulation of constraints?

1.2

Now, consider the following reformulation:

$$(2+z)x + s(z) = 1, \quad \forall z : |z| \leq 1,$$

$$s(z) \geq 0, \quad \forall z : |z| \leq 1.$$

Here the difference is that the slack variable introduced is an arbitrary function of uncertain parameter z. Find the feasible set, and compare it with the previous result.

2 MI/LP Modeling

Model the following using LP/MILP compatible big-M constraints. There is no need to indicate values for M, but please indicate the subset of real numbers to which variables belong (eg. integers $\mathbb{Z}$ etc.).

2.1

$|2x_1 - 3x_2| \geq 6$

2.2

$|x_1 + 2x_2| \leq 4$ (Do not use integer variables here.)

2.3

x_1 and x_2 are integer variables. If $x_1 \leq 4$, $x_2 \geq 5$.

2.4

x_1, x_2 are continuous. Either $x_1 + 4x_2 \leq 7$ or $2x_1 + 3x_2 \geq 4$ are satisfied, but not both.

2.5

x is integer, but not equal to 3.

3 Computational Tools

15.094 will have a significant practical component to complement the theory. To facilitate your implementation of RO, we will be leaning on JuMPeR.jl, a modeling language for robust optimization that couples with JuMP.jl.

We strongly urge you to complete these setup instructions in advance of the first recitation, in case issues crop up. Please make sure that you have the specific versions of the required software, since JuMPeR.jl *is incompatible* with more recent versions of most Julia packages. Please install packages in the given order as well.

3.1 Installation

3.1.1 Install Julia-1.0.5.

Download and install julia-1.0.5 by following the instructions here:
https://julialang.org/downloads/

3.1.2 Install Gurobi 8.1.

Register, download and install Gurobi 8.1, with a free academic license:
https://www.gurobi.com/downloads/gurobi-software/

If you have trouble locating your license information, please refer to the Gurobi account page while logged in:
https://www.gurobi.com/account/

3.1.3 Install git.

Install git (if you don't have it already):
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Git will allow us to update environments for you in real time, in case you need new software for the problem sets. If Git is new to you, we encourage you to check out this tutorial: https://git-scm.com/docs/gittutorial

3.2 Installing needed of Julia packages.

Clone this repo into the directory of your choice:
https://github.com/1ozturkbe/15094code

In your terminal, go inside the 15094code/ directory, and open the Julia project using the following command: `julia --project=.`

This will open a Julia REPL and initialize the Julia environment (described by `15094code/Project.toml`) in the current directory, that will allow you to maintain a specific version of the required packages. Type in the following code to instantiate the packages: `using Pkg; Pkg.instantiate()`

Note that, to access the above packages, you will need to always access Julia through `julia --project=directory-to-15094code`. This way, the packages you `Pkg.add` to the generic `julia1.0.5` environment should not interfere with the versions installed in `15094code/Project.toml`. We have initialized versions of packages you may need to complete future problem set questions and the final project. Feel free to update and change them through the semester, but know that you can always revert to this working version.

3.3 Testing the installation.

Please run `examples/portfolio.jl` either inside your REPL or by typing the following into your terminal: `julia --project=.  examples/portfolio.jl`.

This is an example problem from JuMPeR, and will test your Julia environment for 15.094. If run properly, it should return a table of statistics of portfolio returns for a random portfolio optimization problem, where Γ is the risk budget. You will need to debug your installation if you get an error.

3.4 Check out the JuMPeR tutorial.

In `15094code`, we have included an updated version of the JuMPeR tutorial that Iain developed. This will help you get used to the syntax of JuMPeR (if you are not familiar with JuMP already). It is located in `examples/JuMPeR_tutorial.ipynb`, which is a notebook file. You can access the notebook by calling `jupyter notebook` from the code directory.

If the command doesn't work, you may need to do a separate install of Python Anaconda. Otherwise, you should be able to select Julia-1.0.5 as your kernel, and run the code cell by cell.

3.5 Writing a robust knapsack problem.

Using your newly acquired knowledge of JuMPeR, please write a robust version of the binary knapsack problem under ellipsoidal uncertainty.

$$\begin{aligned} \text{maximize } & \mathbf{c}'\mathbf{x} \\ \text{s.t. } & (\mathbf{a} + \tilde{\mathbf{a}})'\mathbf{x} \leq b, \ \forall\ \tilde{a} : \ ||\tilde{a}||_2 \leq \rho \\ & x_i \in \{0, 1\}, \ i = 1, \ldots, n \end{aligned}$$

Please use $n = 20$, $b = 4$, and randomly generated a_i, $c_i \in [0, 1]$. Include your code, as well as a plot of the optima for a relevant range of ρ. Assuming you were using this problem to maximize the revenue of a cargo aircraft subject to weight capacity, please use your intuition and knowledge of Gaussian probabilities to suggest a ρ that provides a good tradeoff between robustness and optimality.

As always, please come to office hours if you have hiccups with the above.