Lab 7: Parking Garage Case Study

Anna Delesalle

Fri., Mar. 8

Table of contents

# N\_SOW = 10  
# ParkingGarageSOW(; demand\_growth\_rate=rand()\*100, n\_years=20, discount\_rate=0.12)  
  
# [surge(water) for water in 3:10]  
# [ParkingGarageSOW(; demand\_growth\_rate=rand()\*100, n\_years=20, discount\_rate=0.12) for \_ in 1:N\_SOW]

using Revise  
using ParkingGarage  
using Statistics  
using Core

and also regular packages

using Plots  
Plots.default(; margin=5Plots.mm)

# 1. Uncertainty

1. Generate an ensemble of SOWs. Justify how you are sampling the three parameters (n\_years, demand\_growth\_rate, and discount\_rate). I suggest to keep n\_years as a constant, and perhaps to keep the discount rate constant as well.

# sow = ParkingGarageSOW(; demand\_growth\_rate=rand()\*100, n\_years=20, discount\_rate=0.12)  
# N\_SOW = 10000  
# sows = [  
# ParkingGarageSOW(; demand\_growth\_rate=rand()\*100, n\_years=20, discount\_rate=0.12) for  
# \_ in 1:N\_SOW  
# ] # for 100 SOWs  
  
# (sows[1]).n\_year

1. For each SOW, calculate the NPV for each policy.

# function npv\_average(sows, policy)  
# profits = [simulate(i, policy) for i in sows]  
# return mean(profits)  
# end  
  
# n\_levels = 2:12  
# policies = [StaticPolicy(i) for i in n\_levels]

1. Calculate the average NPV for each number of levels and plot.

# policy\_npv = [npv\_average(sows, policy) for policy in policies]  
  
# let  
# plot(  
# n\_levels,  
# policy\_npv;  
# ylabel="NPV Profits [Million USD]",  
# xlabel="Number of levels",  
# legend=false,  
# title="20 Year Horizon, 0.12 Discount, Varying Demand Growth",  
# size=(800, 400),  
# marker=:circle,  
# xticks=n\_levels,  
# )  
# hline!([0])  
# end

## 1.1 Adaptive case

The static case sheds some light on decision making under uncertainty. However, the point of the (denuefville\_parkinggarage:2006?) paper is to illustrate the value of flexibility in decision making.

To implement this, you’ll need to get your hands a bit dirty with the source code. Specifically, you need to edit the function get\_action(x::ParkingGarageState, policy::AdaptivePolicy) function in ParkingGarage/src/sim.jl. You’ll need to use if…else…end statements to implement the adaptive policy. We’ll talk about this in class!

# x = ParkingGarageState(5, 2024)  
# policy = AdaptivePolicy(2)  
# get\_action(x, policy, 1.2)  
# ParkingGarage.calculate\_capacity(x)

Once you’ve implemented this function, you can simulate the adaptive policy and compare the NPV to the static policy. Compare the fixed and adaptive policies for both the deterministic (single SOW) and stochastic (ensemble of SOWs) cases. Plot the NPV as a function of the number of levels for each case.