Lab 7 Solutions

CEVE 421-521

Thu., Apr. 4

1 Setup

Start by loading the

```
using Revise
using ParkingGarage
```

and also regular packages

```
using Distributions
using Plots
using StatsBase
Plots.default(; margin=5Plots.mm)
```

2 Bug squash

There was an important error contained in the `calc_construction_cost` function. This generated positive construction costs even if the height increase was zero. I fixed this by adding a conditional statement to check if the height increase was zero:

```
function calc_construction_cost(n_levels, Δn_levels, is_adaptive)
    cost_per_space = Δn_levels == 0 ? 0 : 16_000 * (1 + 0.1 * (n_levels + Δn_levels - 1))
    if is_adaptive
        cost_per_space = cost_per_space * 1.05
    end
    return cost_per_space * 200
end
```

This had a big impact on my analysis, as it made operating the garage much more profitable than otherwise would have been calculated!

3 Uncertainty

First, I generate an ensemble of SOWs. I'll leave the project planning period constant at 25 years. I'll fix the discount rate at 12%, which might be reasonable if I was given fixed lending terms from a bank. I'll sample the demand growth rate from a Normal distribution with a mean of 40 cars per year and a standard deviation of 25 cars per year, representing large uncertainty.

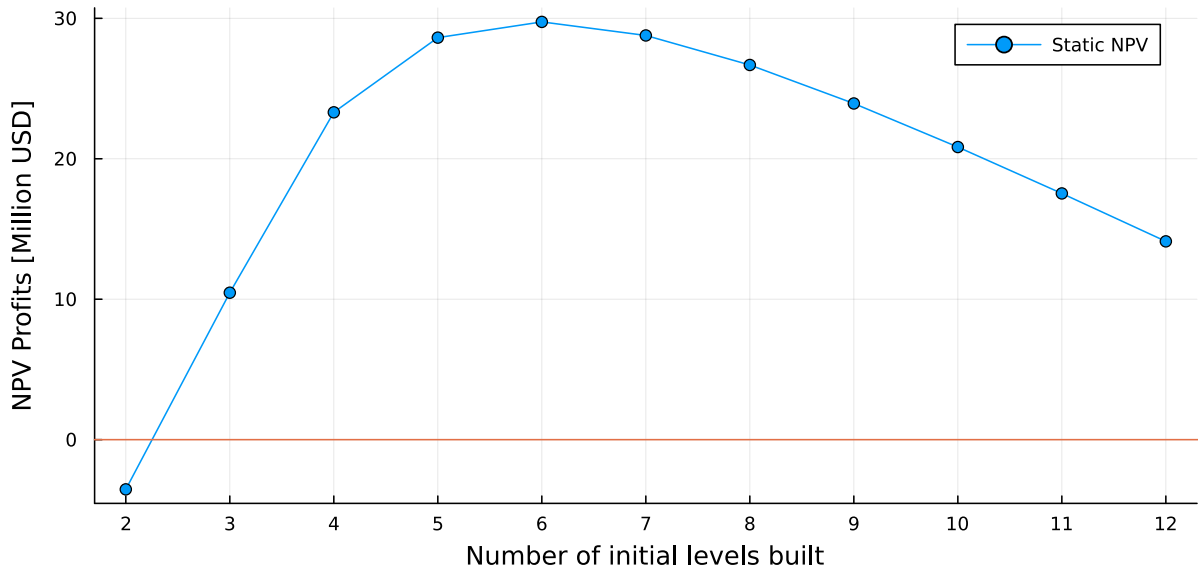
```julia
N_sow = 100_000 # use a lot of SOWs!
function sample_sow()
    return ParkingGarageSOW(;
        demand_growth_rate=rand(Normal(40, 25)), n_years=25, discount_rate=0.12
    )
end
sows = [sample_sow() for i in 1:N_sow]
```

I can calculate the NPV for policy, taking the average across my SOWs.

```julia
n_levels = 2:12
static_policies = [StaticPolicy(i) for i in n_levels]
static_npv = [mean([simulate(sow, policy) for sow in sows]) for policy in static_policies]
```

Finally, I can plot this

```julia
p = plot(
    n_levels,
    static_npv;
    ylabel="NPV Profits [Million USD]",
    xlabel="Number of initial levels built",
    label="Static NPV",
    size=(800, 400),
    marker=:circle,
    xticks=n_levels,
)
hline!(p, [0]; label=false)
```



4 Adaptive case

My code in `sim.jl` looks like:

```
function get_action(x::ParkingGarageState, policy::AdaptivePolicy)
    if x.year == 1
        return ParkingGarageAction(policy.n_levels_init)
    else
        capacity = calculate_capacity(x)
        if x.demand > capacity
            return ParkingGarageAction(1)
        else
            return ParkingGarageAction(0)
        end
    end
end
```

Which implements the adaptive policy. In the first year, it sets the number of levels added to the initial number of levels. In subsequent years, it sets the number of levels added to 1 if the demand exceeds the capacity, and 0 otherwise.

```
policy = AdaptivePolicy(5)
sow = sample_sow()
simulate(sow, policy)
```

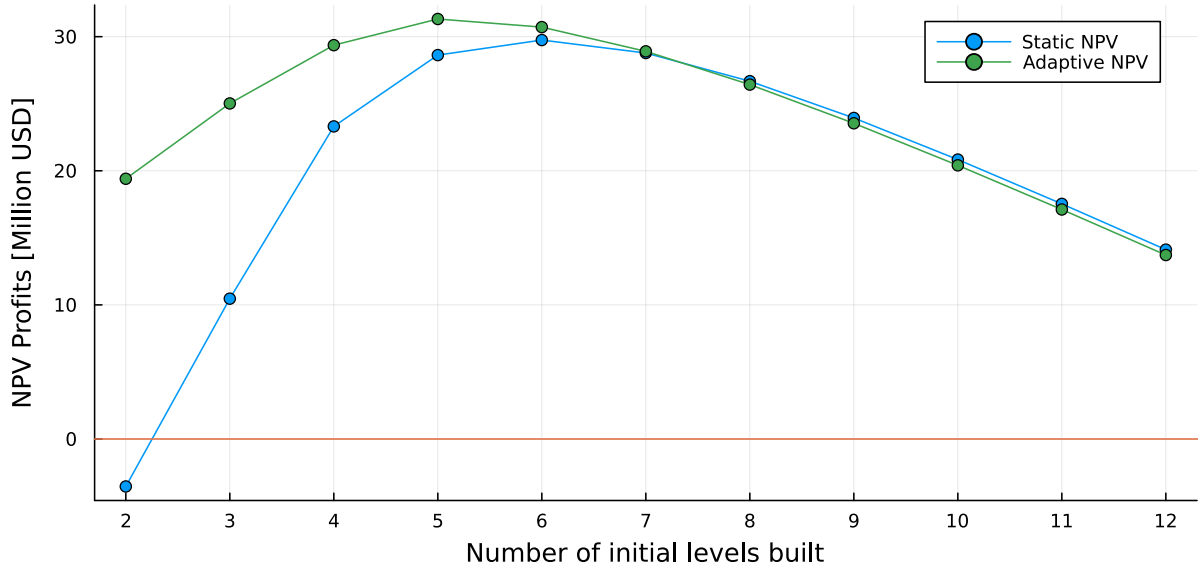
```
22.188979549978026
```

I can now simulate the adaptive policy.

```
adaptive_policies = [AdaptivePolicy(i) for i in n_levels]
adaptive_npv = [
    mean([simulate(sow, policy) for sow in sows]) for policy in adaptive_policies
]
```

and plot the results

```
plot!(p, n_levels, adaptive_npv; label="Adaptive NPV", marker=:circle)
```



We can see that the two policies start to converge above about 8 levels, though the adpatative policy is slightly worse because you have to pay the 5% premium on the construction costs.

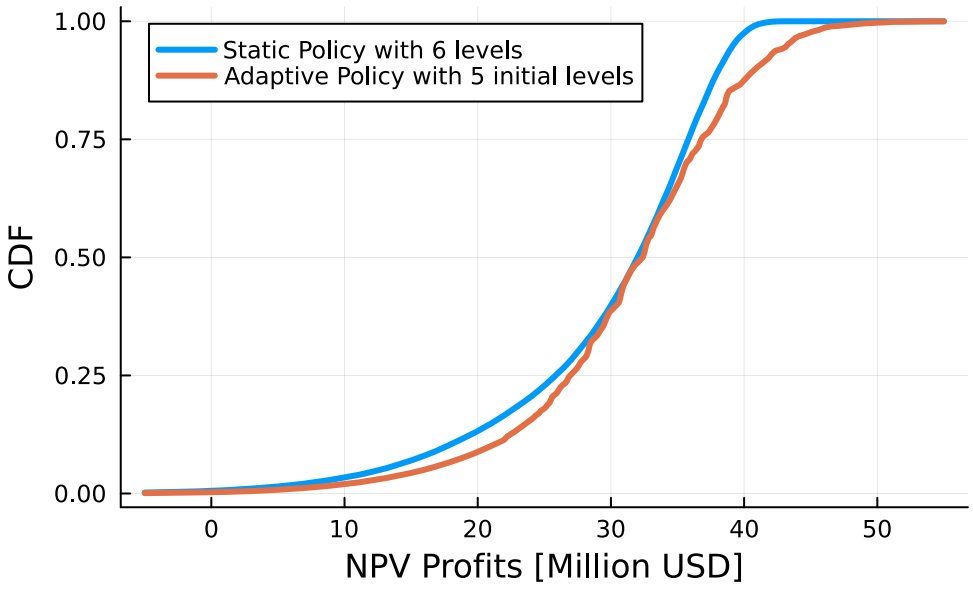
We can also look at the probability distribution of npvs for the best static policy and the best adaptive policy.

```julia
best_static_policy = static_policies[argmax(static_npv)]
best_adaptive_policy = adaptive_policies[argmax(adaptive_npv)]

npvs_static = [simulate(sow, best_static_policy) for sow in sows]
npvs_adaptive = [simulate(sow, best_adaptive_policy) for sow in sows]

# empirical CDFs
cdf_static = ecdf(npvs_static)
cdf_adaptive = ecdf(npvs_adaptive)

plot(
    x -> cdf_static(x),
    -5,
    55;
    label="Static Policy with $(best_static_policy.n_levels) levels",
    xlabel="NPV Profits [Million USD]",
    ylabel="CDF",
    linewidth=3,
)
plot!(
    x -> cdf_adaptive(x);
    label="Adaptive Policy with $(best_adaptive_policy.n_levels_init) initial levels",
    linewidth=3,
)
```



We can see that the adaptive policy is almost purely shifted to the right of the static policy, indicating that it is virtually always better.