# Decision Making under Uncertainty - Spring 2024
# Assignment A

February 7, 2024

## Deadline for all Tasks: March 24th, 2024

## Problem Description

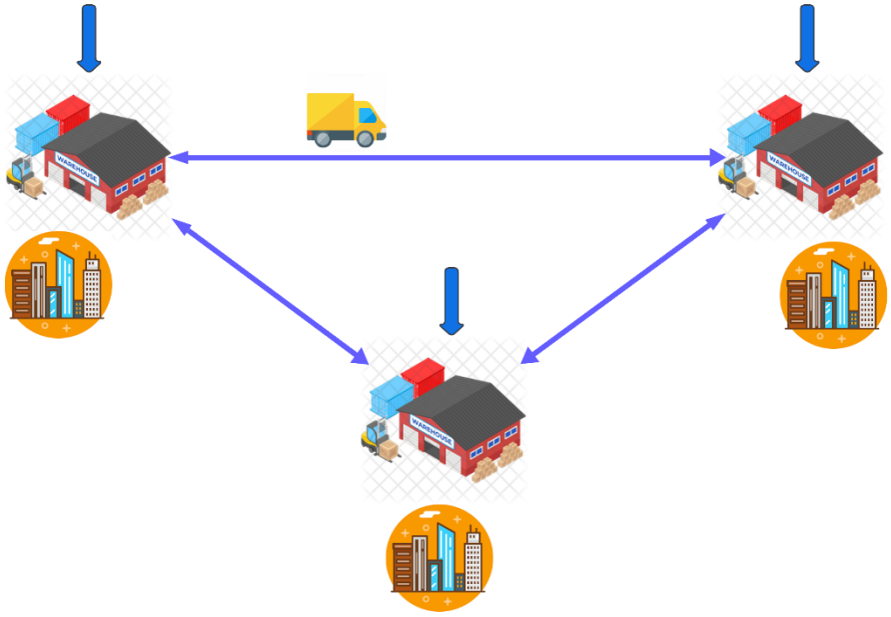

Figure 1: The 3 warehouses' problem

Consider a city divided into three districts. Each district features a dedicated warehouse $w \in \mathcal{W} = \{1, 2, 3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for a continuous good (e.g. coffee), at timeslot/stage $t \in \mathcal{T}$ (e.g. daily), where $\mathcal{T} = \{1, 2, ...\}$. The coffee demand for each warehouse is known and stable at 4, every day.

Each warehouse can store coffee up to a capacity limit $\mathrm{C}_w^{\texttt{storage}}$. Denote the storage level of $w$ at $t$ by $z_{w,t}$. The initial storage level is $z_{w,0} = 2$ for every warehouse. At stage $t$, warehouse $w$ can order an amount $x_{w,t} \geq 0$ of coffee from external suppliers at price $\mathrm{p}_{w,t}$. The price is different for each warehouse and from one day to the next.

Neighboring warehouses can also exchange coffee between them. Let $y_{w,q,t}^{\texttt{receive}}$ denote the amount received by $w$ from a neighboring warehouse $q$, at $t$. Similarly, $y_{w,q,t}^{\texttt{send}}$ is the amount sent by $w$ to $q$. For warehouse $w$ to send an amount $y_{w,q,t}^{\texttt{send}}$ of coffee to a neighboring warehouse $q$ at $t$, warehouse $w$ must already have this amount previously stored (i.e., the coffee ordered by $w$ at $t$, cannot be sent to $q$ at the same day; rather, it can be stored and sent to $q$ from the next day on). The amount that a warehouse can send to a neighbor is restricted by a daily transportation limit $\mathrm{C}_{w,q}^{\texttt{transp}}$. Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.

Finally, failing to meet a district's demand at any day (by a missing amount $m_{w,t}$) comes at a cost $\mathrm{b}_w \cdot m_{w,t}$, where $\mathrm{b}_w > \mathrm{p}_{w,t}$ is a per-unit penalty (e.g. the price charged by a more expensive "emergency" provider that steps in to fulfill the demand).

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses. Your program takes as input the current day (integer), the coffee prices at each district/warehouse

(continuous in $[0, 10]$), and the currently stored quantities (continuous in $[0, \mathrm{C}_w^{\texttt{storage}}]$). It should make a (here-and-now) decision about how much coffee to order at each warehouse and how much to exchange among the warehouses (and consequently how much to store or miss), so that the city's coffee needs are met at the minimum expected cost.

# Task 1: Two-stage problem

**Points: 30**

## Deliverable 1-a: Modelling and Deterministic Optimization

Begin by formulating the problem described, as a deterministic optimization problem, in math mode. You can introduce additional (auxiliary) variables if you must, but try to model the problem using only the variables already defined (they are sufficient): $x_{w,t}, z_{w,t}, y^{\texttt{send}}_{w,q,t}, y^{\texttt{receive}}_{w,q,t}, m_{w,t}$. Assuming prices are known beforehand for a horizon of T days, write a linear program that calculates the optimal decisions:

1. Provide the nomenclature for your model,
2. Write the objective function,
3. Write the set of decision variables,
4. Write the constraints one by one, including a brief verbal description of each constraint.

The deliverable is a pdf file (less than one page) that presents the above.

## Deliverable 1-b: The Expected-Value Benchmark

We now turn to the case where the prices are stochastic. For this task, we only consider a horizon of 2 days: today and tomorrow. Prices are known for today, but not for tomorrow.

You are to implement the Expected-Value (EV) program for a horizon of two days and upload a Julia file containing a function ("`Make_EV_here_and_now_decision`") that receives the prices at day one as input, and returns the (here-and-now) decisions at day one (by also reasoning about what is expected to happen in day two). The problem's data (number of warehouses, costs for transportation, missing demand costs, storage and transport capacities, the initial coffee level at each warehouse, and number of simulation periods) can be accessed from the file `V2_02435_two_stage_problem_data.jl`. You also have access to the stochastic process of the coffee prices: function `sample_next`, from file `V2_price_process.jl`, generates a sample of the coffee price at $t$, given the respective price at $t-1$ (it is meant to be used for each warehouse independently). Given initial prices $\mathrm{p}_{w,1}$, you can calculate the expected second-stage prices by averaging over 1000 samples of the given stochastic process.

## Deliverable 1-c: The Optimal-in-Hindsight Benchmark

You are to implement the Optimal-in-Hindsight and upload a Julia file containing a function ("`Calculate_OiH_solution`") that receives the prices for both days and returns the (stage-one *and* stage-two) decisions as well as the system's cost.

## Deliverable 1-d: Two-stage stochastic programming

Upload a Julia file containing the function "`Make_Stochastic_here_and_now_decision`" (same input-output as the Expected-Value program), which makes a stochastic here-and-now decision by using a specified number of scenarios $N$ for the second-stage prices. The function should generate 1000 equally probable scenarios using `sample_next` (given the initial prices), reduce them to $N$ representative ones with appropriate probabilities, and use those to solve a 2-stage stochastic program.

### Deliverable 1-e: Evaluation and Comparison

You are to upload a Julia file in which you evaluate the Expected-Value Program, the Optimal-in-Hindsight solution, and three versions of your Stochastic Program: one for $N$ equals 5, one for $N$ equals 20, and one for $N$ equals 50 scenarios. The programs should be evaluated in 100 experiments (the same 100 experiments for each program).

The evaluation contains the following steps:

- Generate 100 values (experiments) for the initial prices randomly (uniformly from $[0, 10]$).
- For each experiment, call each program to make a here-and-now decision.
- For each experiment, generate the second-stage prices by sampling from the function `sample_next`, from file `V2_price_process.jl`.
- For each experiment, given each program's decisions for stage 1 and the revealed second-stage prices, solve a deterministic (single-stage) program to make the optimal stage-two decisions.
- For each experiment, record the overall effective cost achieved by each program over the two stages.
- Also simulate the Optimal-in-Hindsight solution for the same 100 experiments (as if you knew the prices beforehand, each time).

### Deliverable 1-f

Write a report (around 2-4 pages pdf) where you

- Present your 2-stage stochastic program (using math equations - do not quote code). Follow the same structure as in the deterministic program of Deliverable 1-a.
- Present the Scenario-Reduction method you used.
- Present the evaluation/comparison results for the three Stochastic Programs (each with a different number of scenarios) together with the Expected-Value Program and the Optimal-in-Hindsight solution. Briefly comment/interpret the results.

## Task 2: Multi-stage stochastic programming

**Points: 30**

In this Task, you will try your best to create a good-performing policy for the full multi-stage warehouses problem. Your policy, similarly to before, receives the current prices as input and returns a here-and-now decision, and must make a decision within 1 second give or take, and certainly in no more than 3 seconds. This limit refers to the time between a call of the function and a return. Thus, everything that takes place within the policy (namely scenario generation, reduction, and solving the optimization problem) should be done within around 1 second each time a decision is to be made. Practically, to have fair limits for all groups, **your policy should use no more than 1000 samples for scenario generation, and no more than 6,500 variables for the stochastic program** (hint: there is a trade-off between the number of scenarios and the length of the lookahead horizon).

The problem's data can be accessed from the file `V2_02435_multistage_problem_data.jl` (it is the same as before, but the number of simulation periods is set to 5 instead of 2). The evaluation framework has already been coded for you, and can be accessed from the file `V2_02435_evaluate_policy.jl`: name your Julia file containing your policy as "`02435_multistage_policy`" and the relevant function as "`make_multistage_here_and_now_decision`". Save the file in the same folder as

Given computational limitations, there is not a single "correct" policy. Rather, you are in competition with other groups in terms of your policy's performance. In general, though, it is safe to say that if you can consistently beat the performance of the expected value program (which you also need to develop), you are doing pretty well already!

### Deliverable 2-a:

Upload a Julia file containing your policy in the form of a function.

### Deliverable 2-b:

Prepare a report where you present:

- (1-2 pages) Your policy, including your multi-stage stochastic program in math mode and the way you defined your scenarios, including e.g. the reduction processes. Clearly state the number of samples, and the number of variables that your policy uses, as well as its performance using the standardized evaluation framework

## Task 3: Credit Assignment

Discuss among the group and write a description of what each member did for each of the Tasks and Deliverables. Be specific and honest. It is deemed that all members of the group approve the description of each member's contributions. Take note that a false statement (for yourself or for others) is a breach of academic integrity.

Agree and propose a credit allocation among group members: given a grade of $x$ points for the group's report, propose a credit allocation coefficient $c_m$ for each group member $m$. Namely, it is expected that $c_m \in [0.75, 1.25]$. As a result, your proposed points for member $m$ would be $c_m \cdot x$. Naturally, the average coefficient over members should be 1.

Nuclear option: If you cannot reach an agreement, you can ask for a meeting with the instructor (to be scheduled after your deliverables are evaluated) where each member's contributions and overall understanding of the concepts will be assessed. Use this option only as a last resort; note that it **can** result in reduced points for all members if the assessment reveals weaknesses not discovered through your deliverables alone.