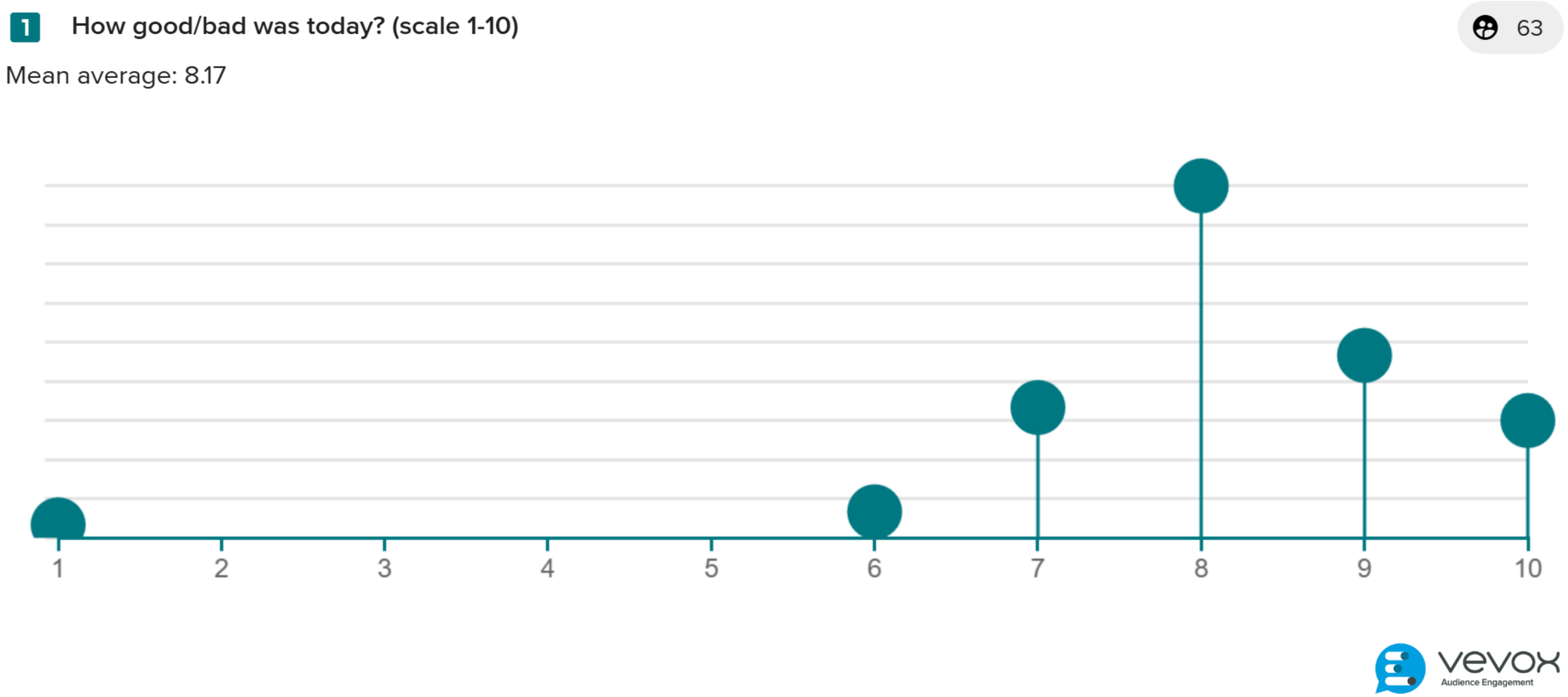
DTU

Decision Making under Uncertainty (02435)

Section for Dynamical Systems, DTU Compute.

Feedback & Follow-up



What you liked

- More interactive lecture
- The proactive teaching and participation of students
- i like the way that we interact with each other.
- Active learning
- Short and sweet!
- Dynamic class, exercises/quizzes help a lot, good to exercise your brain
- Good overview of what we will do in this course.
- Not spending many hours on lectures, just start working.
- The introduction of the course and future lessons were clear. The use of active participation.
- Interactive interesting problems.
- Interesting and dynamic optimization exercises
- I liked the very straight forward way of communicating of what the course will cover and how it is set up. The examples definitely made it easyier to understand.
- Good and clear explanations
- The teaching style is engaging, but I think I need some time to get use to actively engage myself.
- The interactive approach to the lecture
- The recap on optimization
- Very well structured, nice overview.
- I like the conciseness of the lecture No wasted words or time and all info is given
- Giving students the chance to come with a solution before presenting the right answer
- Discovering optimization
- It was very short. This is very good more teachers should keep the first lecture short
- Great lecture
- The inter active lecture That we get to try an anser and that you clearly state what was correct and why (the why is the most important part, that is when I learn from my...
- Good lecture, easy to follow and understand
- The simple explanations on optimization as a recap.
- Loket the interactiveness
- Short and concise, straight to points good structure.
- Active learning part
- Clear examples
- pratical
- Short and precise
- Straight to the point class.
- I like the honesty regarding on how the class may feel like, with the reasoning for that
- That its clear from the beginning what is expected from us to do, and also that the lecture was very well structured.
- It was short and concise
- Great way of active learning and good participation in class, keeping lectures fun and engaging.
- Interactively.
- The example in class and working on them at the same time, it's easier to understand
- You keep making the class to be an interactive class, and it's a fun way to learn for me
- The dynamic interaction and continuous use of questions throughout the lecture made students apply prior concepts and possibly correct misconceptions....
- Really interactive, keep it up!
- The examples were great and works as a refresher
- I like the way how concepts are explained with clear examples.
- Was informative.
- Short and concise
- Good with small simple examples to quickly assess knowledge
- The structure with assignments and no exams.
- The questions
- Engaging lecture, vevox is working well Short lecture
- It was a simple lecture. It is very nice that there is some sort of active learning during lectures. Also, you (lecturer) is good at explaining everything.
- I did, the course seems demanding but interesting
- The interactive part
- Short and to the point, nice outline of the expectations to the course
- Actively participating with instant feedback
- The active learning thing
- Good tempo
- Very clear !
- That you were to the point, and showed examples!
- 7
- The presentation format and the active participation
- Examples

What you disliked

- All was good!
- Could be a bit more quick on explanations
- -
- Not sure if i have enough pre requisits for the programming part of the course
- Almost nothing
- i like the way that we interact with each other. But it would be great if you can mention the use of Groubi as seems you only showed python code at course
- I am not yet familiar with optimization programming
- Too fast for me. Could it be little slower so that i can get all like processing time to understand the problem..
- I am missing some deadlines/information on the two assignments and how they are combined with the tasks
- My lack of knowledge on the programming part, hopefully I can make kt up or the learn page might have some simple tutorials about the language?
- Not so much new stuff if you have already had for example intro to OR
- Nothing
- Nothing so far
- None
- Didn't learn much (yet) - maybe focus more in the first lecture on what 'Decision Making Under Uncertainty' even is/give ideas of the methods we will learn to use
- Slides not uploaded in the beginning
- I am not sure there is anything that I dislike.
- Elaborating before hand on the problems that we answered a little bit
- Nothing
- Nothing
- What are we going to do in the assignment?
- Slides were not uploaded before start of the lecture.
- Ppt not available before class
- Not much new material was presented but this is often the case on the first day
- n/a
- Everything was ok. Maybe just put some more time on the code explanation
- Sometimes I felt things were a little rushed, especially for solving problems
- Dont have anything for this lecture that I did not like
- Not having the slides
- Discovering optimization
- Small problem with the mic making loud noises once in a while
- Few time to think on the problems.
- Missed a bit of basic knowledge recap/explanation.
- Nothing. looking forward to the next one.
- Nothing in particular
- Nothing.
- Maybe a little bit more explaining for those who can't understand
- Maybe a time for group formation initiated by the lecturer would be nice
- Lecture was pretty quick but I understand its so that the exercise part is maximized
- I don’t know if you this active learning this is a bit forced, but I want to see how it works later on too
- I have not thought about anything that I dislike
- I am not as ready in linear programming as I thought I guess
- The active learning method.
- We could discuss the optimization research problem with Carl more
- Not much
- It was good:)
- It almost felt to short.
- Slides could have had a better formatting
- A bit to basic for me
- it was ok for first lecture

What you disliked

All was good!

Could be a bit more quick on explanations

-

Not sure if i have enough pre requisits for the programming part of the course

Almost nothing

i like the way that we interact with each other. But it would be great if you can mention the use of Groubi as seems you only showed python code at course

I am not yet familiar with optimization programming

Too fast for me. Could it be little slower so that i can get all like processing time to understand the problem..

I am missing some deadlines/information on the two assignments and how they are combined with the tasks

My lack of knowledge on the programming part, hopefully I can make kt up or the learn page might have some simple tutorials about the language?

Not so much new stuff if you have already had for example intro to OR

Nothing

Nothing so far

None

Didn't learn much (yet) - maybe focus more in the first lecture on what 'Decision Making Under Uncertainty' even is/give ideas of the methods we will learn to use

Slides not uploaded in the beginning

I am not sure there is anything that I dislike.

Elaborating before hand on the problems that we answered a little bit

Nothing

Nothing

What are we going to do in the assignment?

Slides were not uploaded before start of the lecture.

Ppt not available before class

Not much new material was presented but this is often the case on the first day

n/a

Everything was ok. Maybe just put some more time on the code explanation

Sometimes I felt things were a little rushed, especially for solving problems

Dont have anything for this lecture that I did not like

Not having the slides

Discovering optimization

Small problem with the mic making loud noises once in a while

Few time to think on the problems.

Missed a bit of basic knowledge recap/explanation.

Nothing. looking forward to the next one.

Nothing in particular

Nothing.

Maybe a little bit more explaining for those who can't understand

Maybe a time for group formation initiated by the lecturer would be nice

Lecture was pretty quick but I understand its so that the exercise part is maximized

I don't know if you this active learning this is a bit forced, but I want to see how it works later on too

I have not thought about anything that I dislike

I am not as ready in linear programming as I thought I guess

The active learning method.

We could discuss the optimization research problem with Carl more

Not much

It was good:)

It almost felt to short.

Slides could have had a better formatting

A bit to basic for me

it was ok for first lecture

What you disliked

- All was good!
- Could be a bit more quick on explanations
- -
- Not sure if i have enough pre requisits for the programming part of the course
- Almost nothing
- i like the way that we interact with each other. But it would be great if you can mention the use of Groubi as seems you only showed python code at course
- I am not yet familiar with optimization programming
- Too fast for me. Could it be little slower so that i can get all like processing time to understand the problem..
- I am missing some deadlines/information on the two assignments and how they are combined with the tasks
- My lack of knowledge on the programming part, hopefully I can make kt up or the learn page might have some simple tutorials about the language?
- Not so much new stuff if you have already had for example intro to OR
- Nothing
- Nothing so far
- None
- Didn't learn much (yet) - maybe focus more in the first lecture on what 'Decision Making Under Uncertainty' even is/give ideas of the methods we will learn to use
- Slides not uploaded in the beginning
- I am not sure there is anything that I dislike.
- Elaborating before hand on the problems that we answered a little bit
- Nothing
- Nothing
- What are we going to do in the assignment?
- Slides were not uploaded before start of the lecture.
- Ppt not available before class
- Not much new material was presented but this is often the case on the first day
- n/a
- Everything was ok. Maybe just put some more time on the code explanation
- Sometimes I felt things were a little rushed, especially for solving problems
- Dont have anything for this lecture that I did not like
- Not having the slides
- Discovering optimization
- Small problem with the mic making loud noises once in a while
- Few time to think on the problems.
- Missed a bit of basic knowledge recap/explanation.
- Nothing. looking forward to the next one.
- Nothing in particular
- Nothing.
- Maybe a little bit more explaining for those who can't understand
- Maybe a time for group formation initiated by the lecturer would be nice
- Lecture was pretty quick but I understand its so that the exercise part is maximized
- I don't know if you this active learning this is a bit forced, but I want to see how it works later on too
- I have not thought about anything that I dislike
- I am not as ready in linear programming as I thought I guess
- The active learning method.
- We could discuss the optimization research problem with Carl more
- Not much
- It was good:)
- It almost felt to short.
- Slides could have had a better formatting
- A bit to basic for me
- it was ok for first lecture

Farmer Carl - Model formulation

Decision variables:

$x_C =$ Number of cows, Carl should buy

$x_S =$ Number of sheep, Carl should buy

$$y_S = \begin{cases} 1, & \text{if Carl buys more than 100 sheep} \\ 0, & \text{if Carl buys no sheep} \end{cases}$$

$$y_C = \begin{cases} 1, & \text{if Carl buys more than 10 cows} \\ 0, & \text{otherwise} \end{cases}$$

MIP:

$$\begin{aligned} \text{Max } & \{(400-200)x_C + \\ & \quad (70-30)x_S - 1000y_C\} \\ \text{s.t. } & x_C \leq 50 \\ & x_S \leq 200 \\ & x_C + 0.2x_S \leq 72 \\ & 150x_C + 25x_S \leq 10000 \\ & \boxed{100y_S \leq x_S \leq 200y_S} \\ & \boxed{11y_C \leq x_C \leq 10 + 40y_C} \\ & x_C, x_S \geq 0 \text{ and integer} \\ & y_C, y_S \in \{0, 1\} \end{aligned}$$

Sanity Check!

Pyomo Code Carl.py

```
from pyomo.environ import *

# Create a model
model = ConcreteModel()

# Declare variables
model.xC = Var(bounds=(0, 50), within=Integers)
model.xS = Var(bounds=(0, 200), within=Integers)
model.yS = Var(bounds=(0, 1), within=Binary)
model.yC = Var(bounds=(0, 1), within=Binary)

# Objective function: Maximization of profits
model.profit = Objective(
    expr=(400 - 200) * model.xC + (70 - 30) * model.xS - 1000 * model.yC,
    sense=maximize
)

# Constraints

#Constraint on available acres
model.Acres = Constraint(expr=model.xC + 0.2 * model.xS <= 72)

#Constraint on maximum working hours
model.WorkingHours = Constraint(expr=150 * model.xC + 25 * model.xS <= 10000)

#Minimum of 100 sheep constraint
model.AtLeast100Sheep1 = Constraint(expr=model.xS - 100 * model.yS >= 0)
model.AtLeast100Sheep2 = Constraint(expr=model.xS - 200 * model.yS <= 0)

#Maximum of 10 cows without milk machine
model.MilkMachine1 = Constraint(expr=model.xC - 10 - 40 * model.yC <= 0)
model.MilkMachine2 = Constraint(expr=model.xC - 11 * model.yC >= 0)

# Create a solver
solver = SolverFactory('gurobi')  # Make sure Gurobi is installed and properly configured

# Solve the model
results = solver.solve(model, tee=True)
```

Plan

→ ~~Task 0 : last week and at home~~

→ Today: Task 1
Building an evaluation framework for sequential decision-making methods

Task 4 is about reporting the results from Tasks 2 and 3

→ Weeks 3-4: Task 2
Stochastic Programming

→ Week 5: Assignment Work for Task 2 and Q&A

→ Weeks 6-7: Task 3
Approximate Dynamic Programming

→ Week 8: Assignment Work for Task 3 and Q&A

The process of designing "Decision-making" frameworks

"Decisions"

- ❖Choosing a career path?
- ❖Deciding on which country to move next?
- ❖Stay in a relationship or not?

Define → Design → Validate

define the problem, what are you after? before talking about solutions

The process of designing "Decision-making" frameworks

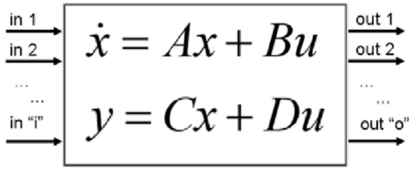
"Decisions"

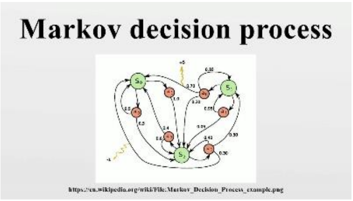
- ❖Choosing a career path?
- ❖Deciding on which country to move next?
- ❖Stay in a relationship or not?

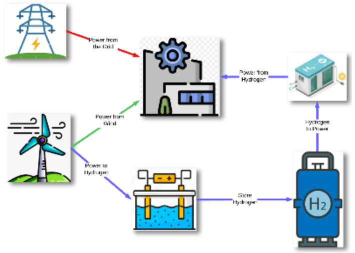
Define → Design → Validate

Coding a simulation Environment to evaluate *any* decision-making policy

how good it is







Agenda for today

1. Defining a problem of Sequential Decisions under Uncertainty
2. Examples
3. Coding an Evaluation Framework for the Assignment (Task 1)

Markov Decision Process

- Stages $t \in \mathcal{T}$
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \dots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \dots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

each statge you need to make a decicion, what information we can get from the system - stat - ex temperature
decision - what can we control
dyn - links the next stage function will be, as function current state and current decicion - deterministic
i take action to take to right but also depends on the wind, current stat wind, and my decision
cost function - can be rewaard, what i want to achieve, each stat i take an action, as function of stat and action i get an imediate reaawerd or cost

Markov Decision Process

- Stages $t \in \mathcal{T}$
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \dots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \dots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

The *State* variables $\boldsymbol{x}_t$ enclose the necessary and sufficient information to model the system's behavior from stage t onwards.

everyting we now from the system should go to the satat vaaible

Markov Decision Process

we need to make the optimal decicion, in markov we need to design a policy, from state to decision, if i counter tis state i do this decision other i do toger decision
policy/method
optimal policy, one that minimizes the expected cost, over time, check if this makes sense
policy is a mapping from states to decisions, sum of cost trough the satates

- Stages $t \in \mathcal{T}$ evry day, every week
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \dots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \dots\}$

  do i show up, do i make a question? something that afect cost function (grade, effort), time spend, how much effort, i put some effort now and more later
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

The *State* variables x_t enclose the necessary and sufficient information to model the system's behavior from stage t onwards.

State
knowledge you have, skill, how does evolve on stat t,
deterministic - skill_t+1=f(skill_t,effor_t),
stocastic - stress_t+1=g(stress_t,effort_t), stress is more uncertain, skill more deterministic, we could add more stochastic stress_t+1 = g(stree, effort, weather) weather is uncertain, is not a decision, is a state varaible, it has is own dynamics,
weather_t+1 = h(wwather_t) --> exogenues state, not depending on decisions just on the stat

Solution Concept:

Policy π: $\boldsymbol{u}_t = \pi(\boldsymbol{x}_t)$

Optimal Policy:

state t+1 dependes only on t, not in other times
~the skill now i dont need to know the skill before

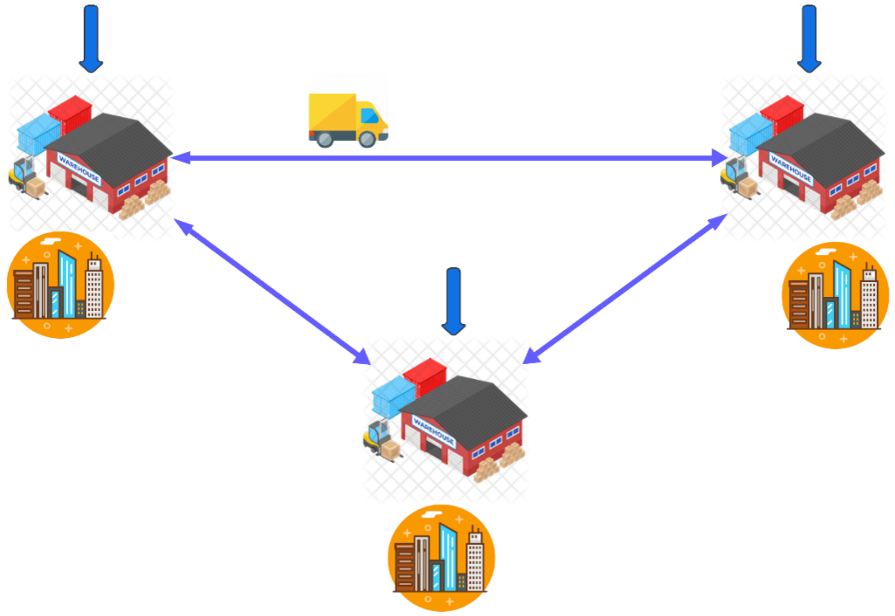
$$\min_{\pi} \left\{ \sum_t E_{\boldsymbol{x}_t \sim \pi}[c_t] \right\}$$

statet=(skill_T, skill_t-1)
state_t+1 = f(state_t)

$$\boldsymbol{u}_t = \pi(\boldsymbol{x}_t), \qquad \forall t$$
$$c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t), \qquad \forall t$$
$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t), \ \forall t$$

in order to preserve, we need to include aall that is relevant into the state

The Lead Example



how many days are we interested

- Stages $t \in \mathcal{T}$
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \ldots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \ldots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

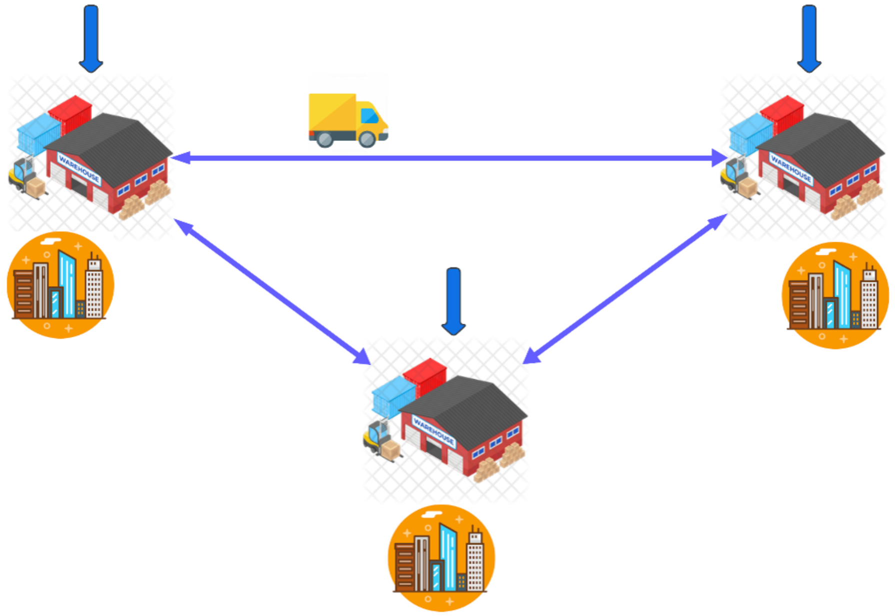
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee between them.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost of b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

The Lead Example



- Stages $t \in \mathcal{T}$
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \ldots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \ldots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

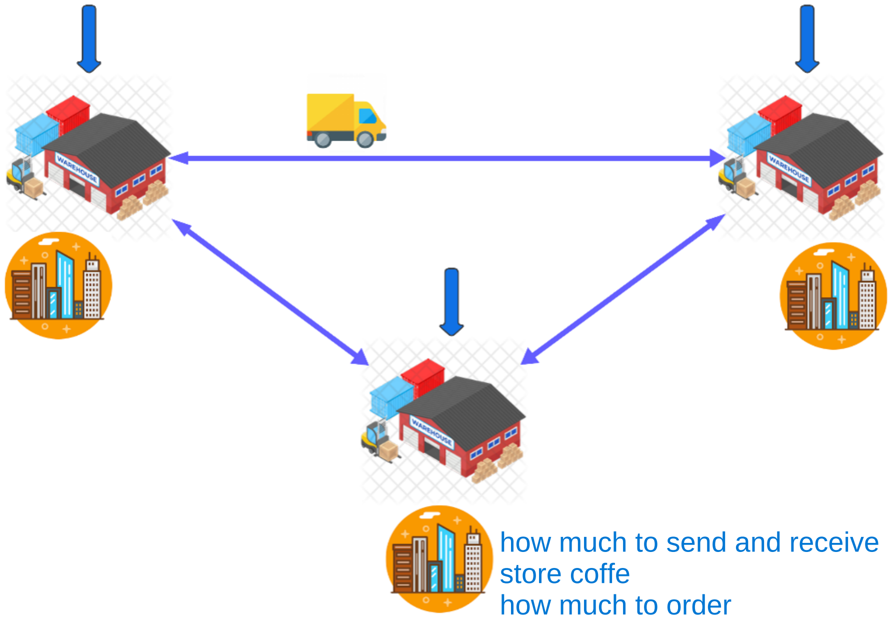
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee between them.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost of b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

The Lead Example



- Stages $t \in \mathcal{T}$
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \dots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \dots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

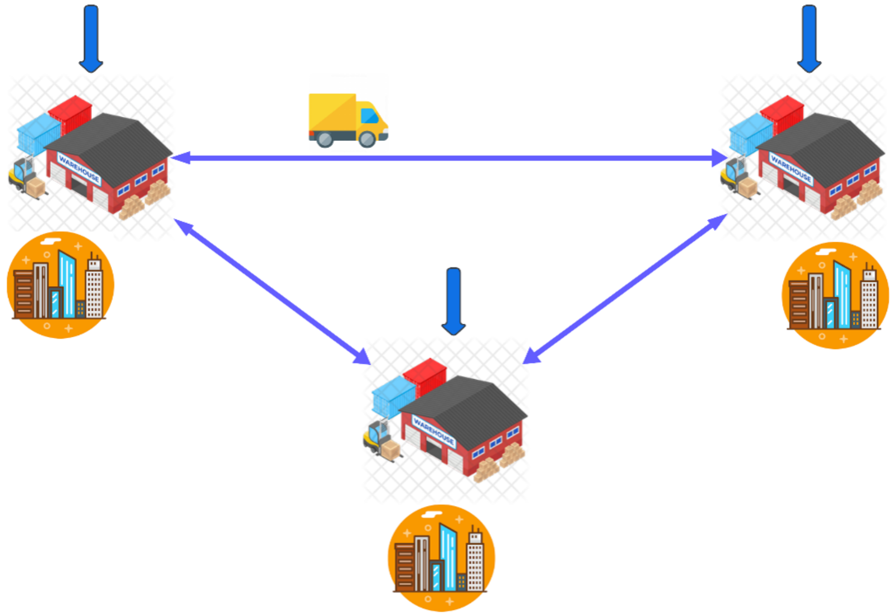
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee between them.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost of b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

The Lead Example



- Stages $t \in \mathcal{T}$

storage level,

- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \dots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \dots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

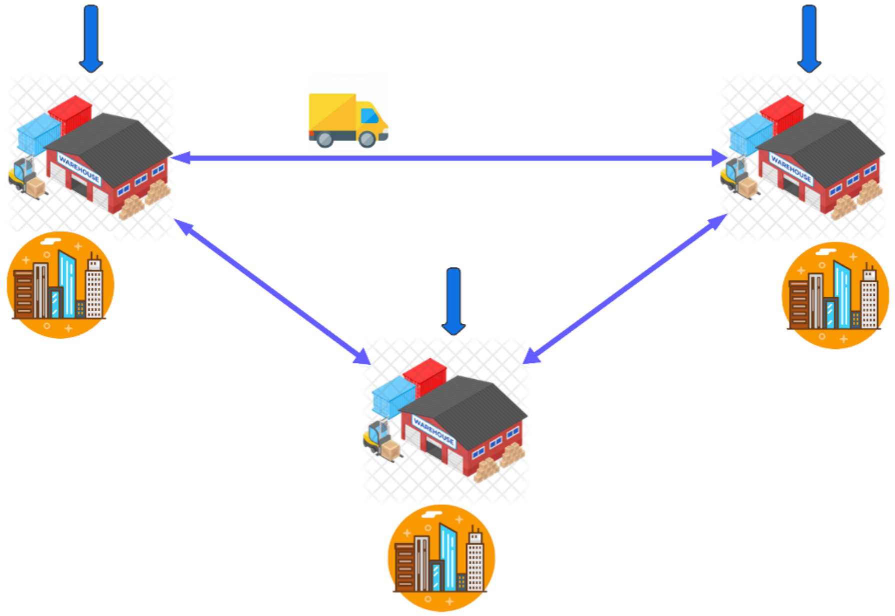
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee between them.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost of b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

The Lead Example



- Stages $t \in \mathcal{T}$
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \dots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \dots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

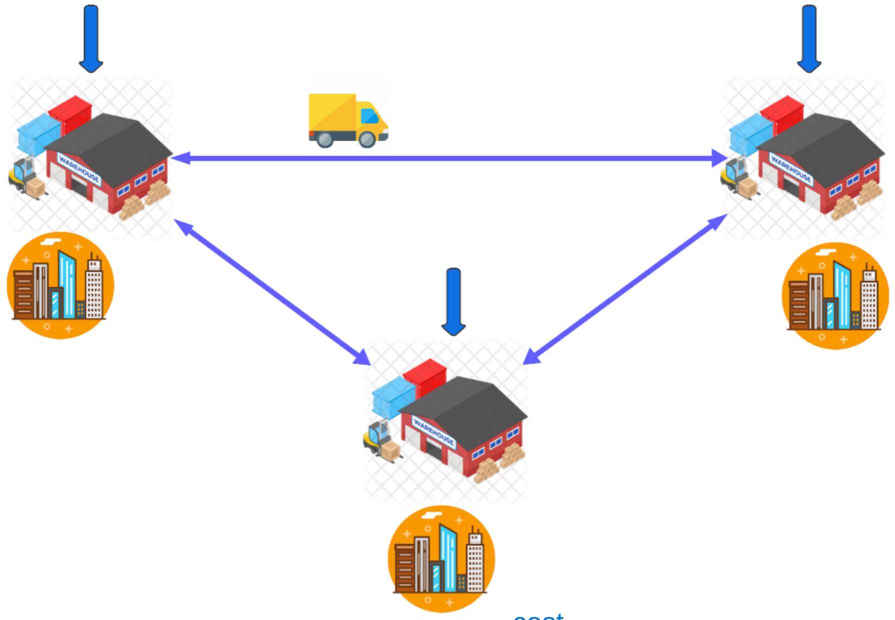
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee between them.
Let $y_{w,q,t}^{rcv}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y_{w,q,t}^{send}$ is the amount sent by w to q.
To send an amount $y_{w,q,t}^{send}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost of b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

The Lead Example



cost
price * how much order
+ sent between * proce
+ fail to meet demand * cost

- Stages $t \in \mathcal{T}$
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \ldots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \ldots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee between them.
Let $y_{w,q,t}^{rcv}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y_{w,q,t}^{send}$ is the amount sent by w to q.
To send an amount $y_{w,q,t}^{send}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost of b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

The Lead Example

$$z_{w,t+1} = z_{w,t} - d_{w,t} + o_{w,t} + \sum_{q \in W} y^{rcv}_{w,q,t} - \sum_{q \in W} y^{send}_{w,q,t}$$

- Stages $t \in \mathcal{T}$
- State $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \dots\}$
- Decision $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \dots\}$
- Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$
- Cost $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

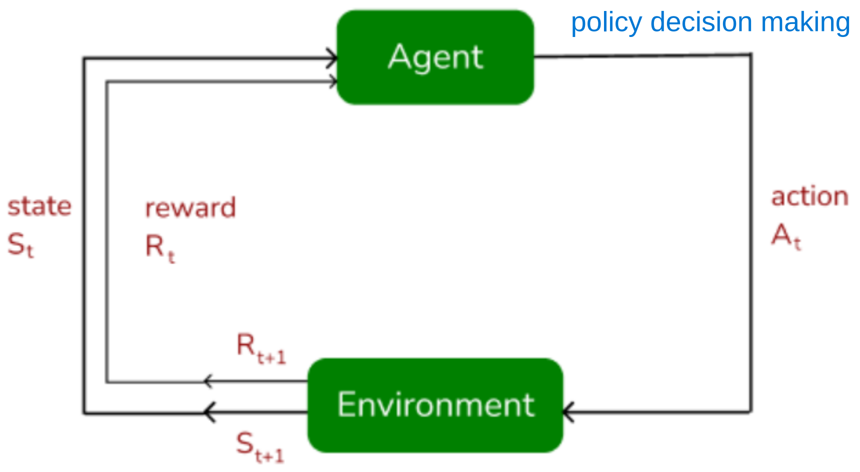
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee between them.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost of b_w. $d_{w,t}$

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

Simulation Environment



task 1
it presents the agent with the curretn state, agent gives an action state, and gives back to environment and the it gives a reward cost based

it takes policy, spits a reward and next stage

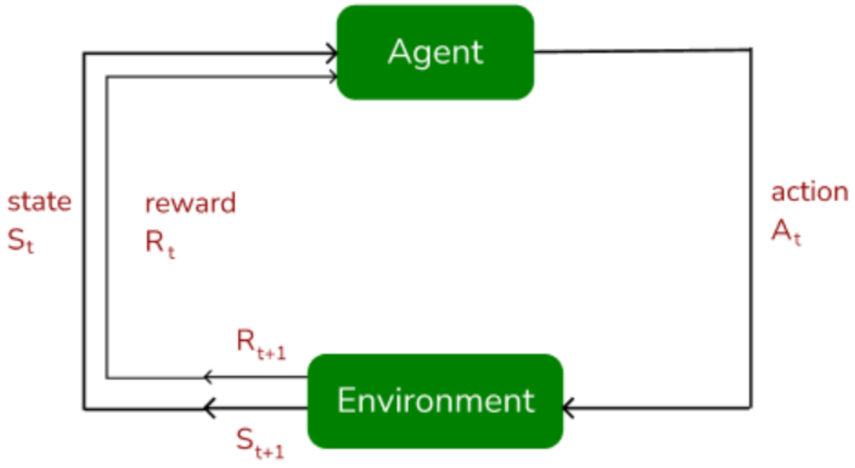
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $D_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $e_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

Simulation Environment



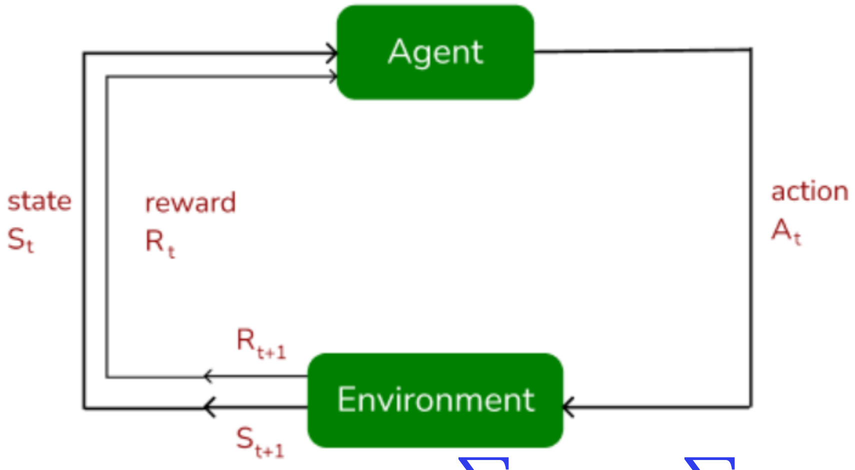
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $D_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $e_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

Simulation Environment



$$Z = z_{w,t} - d_{w,t} + o_{w,t} + \sum_{q \in W} y^{rcv}_{w,q,t} - \sum_{q \in W} y^{send}_{w,q,t} \ldots$$

if $Z > C^{store}$:
  then $z_{w,t+1} = C^{store}$
if $Z < 0$:
  then $z_{w,t+1} = 0$, $d_{w,t} = d_{w,t} - |Z|$
else: $z_{w,t+1} = Z$

decision making policy can do order more coffe as much as i like, it is the env to make sure problem is consistent, cannot store more coffe that can store, we need code in the environment, that takes the action and check the previous storage level and calculate new storage level, any extra coffee is trown awaay
we might type police and order negative amounts, we need to take into account crazy things, we dont want t negative amounts
storage 10, i order 2, i dont receive or order, i have 12 available, emand is 14, i have 12, demand goes there and i nneed to set, demand serve as 12 and missing 2

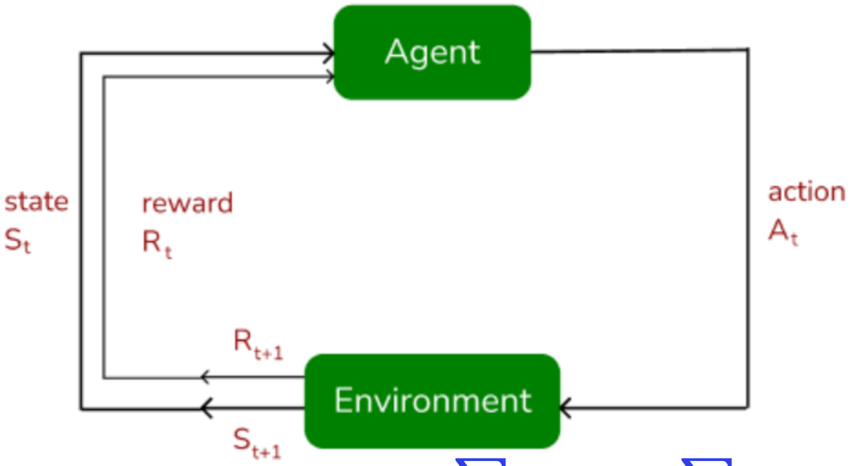
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $D_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $e_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

Simulation Environment



$$Z = z_{w,t} - d_{w,t} + o_{w,t} + \sum_{q \in W} y^{rcv}_{w,q,t} - \sum_{q \in W} y^{send}_{w,q,t} \ldots$$

if $Z > C^{store}$:
  then $z_{w,t+1} = C^{store}$
if $Z < 0$:
  then $z_{w,t+1} = 0$, $d_{w,t} = d_{w,t} - |Z|$
else: $z_{w,t+1} = Z$

if $y^{send}_{w,q,t} > \min\{z_{w,t}, C^{trsnp}\}$:
  then $y^{send}_{w,q,t} = \min\{z_{w,t}, C^{trsnp}\}$

ii cannot send more thatn what is allowed and whaat is previuosly stored

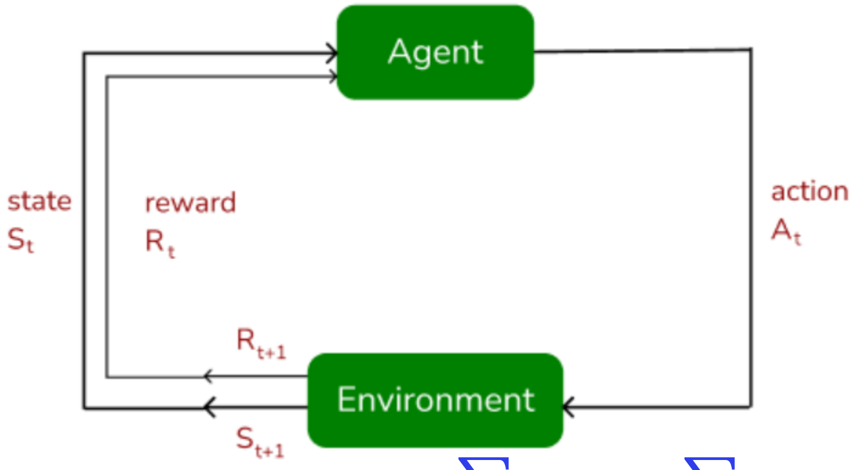
Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $D_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $e_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

Simulation Environment



$$Z = z_{w,t} - d_{w,t} + o_{w,t} + \sum_{q \in W} y^{rcv}_{w,q,t} - \sum_{q \in W} y^{send}_{w,q,t} \ldots$$

if $Z > C^{store}$:
  then $z_{w,t+1} = C^{store}$
if $Z < 0$:
  then $z_{w,t+1} = 0, \; d_{w,t} = d_{w,t} - |Z|$
else: $z_{w,t+1} = Z$

if $y^{send}_{w,q,t} > \min\{z_{w,t}, C^{trsnp}\}$:
  then $y^{send}_{w,q,t} = \min\{z_{w,t}, C^{trsnp}\}$

if $y^{rcv}_{w,q,t} \neq y^{send}_{w,q,t}$:
  then $y^{rcv}_{w,q,t} = y^{send}_{w,q,t}$

what receives should be consistent with what the other sends, we set it equal

Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $D_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $e_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

Simulation Environment

$$c_t = \sum_{w \in W} \left(p_{w,t} o_{w,t} + \mathrm{e}_{w,q} y_{w,q,t}^{send} + \mathrm{b}_w \left(D_{w,t} - d_{w,t} \right) \right)$$

cost function, we need to return to envirnment every time we take a decision

$$Z = z_{w,t} - d_{w,t} + o_{w,t} + \sum_{q \in W} y_{w,q,t}^{rcv} - \sum_{q \in W} y_{w,q,t}^{send} \dots$$

if $Z > \mathrm{C}^{store}$:
then $z_{w,t+1} = \mathrm{C}^{store}$
if $Z < 0$:
then $z_{w,t+1} = 0$, $d_{w,t} = d_{w,t} - |Z|$
else: $z_{w,t+1} = Z$

if $y_{w,q,t}^{send} > \min\{z_{w,t}, \mathrm{C}^{trsnp}\}$:
then $y_{w,q,t}^{send} = \min\{z_{w,t}, \mathrm{C}^{trsnp}\}$

if $y_{w,q,t}^{rcv} \neq y_{w,q,t}^{send}$:
then $y_{w,q,t}^{rcv} = y_{w,q,t}^{send}$

Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee.
Let $y_{w,q,t}^{rcv}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y_{w,q,t}^{send}$ is the amount sent by w to q.
To send an amount $y_{w,q,t}^{send}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost b_w.

Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

Simulation Environment

$$c_t = \sum_{w \in W} \left(p_{w,t} o_{w,t} + \mathrm{e}_{w,q} y^{send}_{w,q,t} + \mathrm{b}_w \left(D_{w,t} - d_{w,t} \right) \right)$$

if $d_{w,t} < 0$:
  then $d_{w,t} = 0$

demadn might go negative, cannot serve more demadn more then ther is

$$Z = z_{w,t} - d_{w,t} + o_{w,t} + \sum_{q \in W} y^{rcv}_{w,q,t} - \sum_{q \in W} y^{send}_{w,q,t} \dots$$

if $Z > \mathrm{C}^{store}$:
  then $z_{w,t+1} = \mathrm{C}^{store}$
if $Z < 0$:
  then $z_{w,t+1} = 0$, $d_{w,t} = d_{w,t} - |Z|$
else: $z_{w,t+1} = Z$

if $y^{send}_{w,q,t} > \min\{z_{w,t}, \mathrm{C}^{trsnp}\}$:
  then $y^{send}_{w,q,t} = \min\{z_{w,t}, \mathrm{C}^{trsnp}\}$

if $y^{rcv}_{w,q,t} \neq y^{send}_{w,q,t}$:
  then $y^{rcv}_{w,q,t} = y^{send}_{w,q,t}$

Consider a city divided into three districts.
Each district features a dedicated warehouse $w \in W = \{1,2,3\}$ which serves the district's demand $\mathrm{D}_{w,t}$ for coffee.
The coffee demand for each warehouse and day is known.

Each warehouse can store coffee up to a capacity limit C^{store}.
Denote the storage level of w at t by $z_{w,t}$.
At stage t, each warehouse w can order an amount $o_{w,t}$ of coffee from external suppliers at price $p_{w,t}$.
The price is different for each warehouse and each day.

Neighboring warehouses can also exchange coffee.
Let $y^{rcv}_{w,q,t}$ denote the amount received by w from a neighboring warehouse q, at t.
Similarly, $y^{send}_{w,q,t}$ is the amount sent by w to q.
To send an amount $y^{send}_{w,q,t}$, warehouse w must already have this amount previously stored.
The amount sent in one stage is restricted by a transportation limit C^{trnsp}.
Each exchange comes at a per-unit transportation cost $\mathrm{e}_{w,q}$.
Failing to meet a district's demand at any day comes at a per-unit cost b_w. $d_{w,t}$

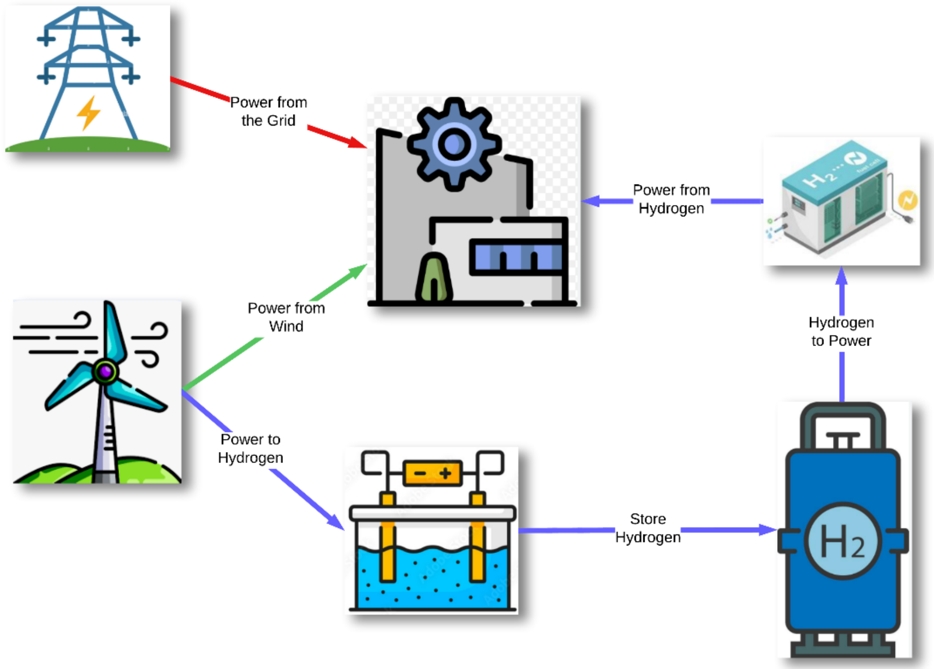
Your job is to build a program that makes the day-by-day decisions for the coffee distribution system of the three warehouses, so that the city's coffee needs are met at the minimum expected cost.

we produce experimetn of wind time series

policy, inputs current state brings a vlaue to the environment

Assignment A, Task 1

if we do a lot of experimetnss, various prices, we will simulaate what decisions will it make, what decisions would occur, this is how the policy would work in real



Deliverable 1: MDP

State variables $\boldsymbol{x}_t = \{x_{1,t}, x_{2,t}, \dots\}$

to simulate wind, simulations of random process of

Decision variables $\boldsymbol{u}_t = \{u_{1,t}, u_{2,t}, \dots\}$

Dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$

Cost function $c_t = g(\boldsymbol{x}_t, \boldsymbol{u}_t)$

if i just use one, it can be an outlier, out of the statistics, we do several to have a good ideaa of the policy costs

Deliverable 2: Policy Evaluation Framework

Input: *policy* (python function that returns decisions)
Initialize state variables
For experiment 1 to E:
  For stage 1 to H:
    decisions = *policy*(state)
    check/correct decisions if inconsistent
    calculate cost for this stage and experiment
    calculate state at next stage
  calculate total cost of policy for this experiment
Return: expected policy cost (average over experiments)

Questions and Survey

- Group selection is open until Friday

random decision is a policy, we can design better
policy is a decision of the state

you design the policy and then each state we just apply policy